Diss. ETH No. 19970

Probabilistic Role Mining

A dissertation submitted to

ETH Zurich

for the degree of

Doctor of Sciences

presented by

Mario Frank

Dipl. Phys. (University of Heidelberg)

born December 1, 1981

citizen of Germany

accepted on the recommendation of

Prof. Dr. David Basin

Prof. Dr. Joachim M. Buhmann

Prof. Dr. Andrea Montanari

- 2011 -

Abstract

Controlling access in a system with thousands of users and thousands of resources is cumbersome and error-prone if this task is carried out on the level of individual assignments of users to access-permissions. The preferred model, that drastically simplifies administration, is role-based access control (RBAC). An RBAC configuration consists of a user-role assignment relation and a role-permission assignment relation. Thereby, a user is assigned all permissions that are contained in his roles. The migration step from an existing system on the basis of direct user-permission assignments to an RBAC configuration has been identified as one of the costliest aspects of RBAC. Automatically finding an RBAC configuration starting from such a direct user-permission assignment is called 'role mining'. This thesis concerns the role mining problem.

The relevant aspects of role mining are the formal problem definition, algorithmic approaches, as well as methods for validating solutions. These aspects guide the outline of the thesis. Accordingly, we begin by analyzing all established formal definitions of the role mining problem by comparing them with the real-world requirements for RBAC configurations that enterprises typically have. Realizing that established definitions fail to comply with many of these requirements, we redefine role mining as an inference problem. This problem aims at inferring the underlying role structure of the data at hand. An RBAC configuration that solves this problem will most likely be able to endow new, i.e. previously unseen, users with the correct permissions. We identify this generalization ability as the most important quality measure for

role mining. This thesis provides the first comparison and analysis of all established definitions of the role mining problem.

We develop a probabilistic approach to the role mining problem. The majority of this thesis is devoted to defining and analyzing a class of probabilistic models for RBAC. Thereby, we focus on two particular model instances. The first model, called 'multi-assignment clustering' (MAC), supports the assignment of a user to multiple roles. The second model, called 'disjoint decomposition model' (DDM), has a two-level role hierarchy. Users are assigned to 'business-roles', permissions are assigned to 'technical roles', and technical roles are assigned to business roles. Thereby, a user can have only one business-role and a permission can have only one technical role, while the role-role assignment has no constraints. We analyze both models, MAC and DDM, in detail. Additionally, we investigate various model variants for erroneous user-permission assignments, different prior assumptions on the model parameters, as well as different inference algorithms. We experimentally demonstrate that both models yield more precise parameter estimates than combinatorial algorithms. Moreover, we observe that MAC has a superior generalization ability than DDM. Therefore, we recommend the MAC model for role mining.

As an extension of MAC, we propose 'hybrid role mining'. This is a role mining variant where business attributes of the users are provided as additional input. The goal is to find RBAC configurations with intuitive user-role assignments in terms of the business attributes that, at the same time, fit the user-permission assignment. We propose information-theoretic measures to select consistent business attributes. Moreover, we show that there is a trade-off between the two individual objectives of this problem when taking business attributes into account. However, there is a Pareto-optimal point that can be found with our methods. At this point, the learned RBAC configurations generalize well and are intuitive from a business perspective.

Finally, we study the problem of validating solutions of the role mining problem. As generalization ability is the most important quality measure for role mining, we develop a method for estimating the generalization error of RBAC configurations. This method, called the 'minimal

transfer cost' principle, is not limited to role mining but can be used for other unsupervised learning problems as well. We demonstrate this claim on a number of applications. Moreover, we study the framework of 'approximation set coding' (ASC). We explain this novel information-theoretic method for model selection and apply it to two practical tasks. The first task is to select the appropriate model for role mining from a set of candidate models. The second task is to select the optimal cut-off rank for truncated singular-value decomposition (SVD) for denoising binary matrices. Both applications are among the first applications of the ASC framework.

Zusammenfassung

Die Zugriffskontrolle in einem System mit tausenden Benutzern und Ressourcen ist mühsam und fehleranfällig sofern diese Aufgabe auf Basis individueller Zuweisungen von Benutzern zu Zugriffsberechtigungen angegangen wird. Das Modell der Wahl, welches die Administration drastisch vereinfacht, ist daher die rollenbasierte Zugriffskontrolle (RBAC). Eine RBAC-Konfiguration ist durch zwei Relationen definiert: eine Zuweisung von Benutzern zu Rollen und eine Zuweisung von Rollen zu Berechtigungen. Dabei erhält ein Benutzer alle Berechtigungen, die seinen Rollen zugewiesen sind. Die Überführung eines Systems, welches auf einer direkten Zuteilung von Berechtigungen an Benutzer basiert, hin zu einer RBAC Konfiguration wurde als der kostenträchtigste Aspekt von RBAC ausgemacht. Solch eine Konfiguration, startend von Direktzuteilungen, automatisch zu finden wird 'role mining' genannnt. Die vorliegende Arbeit behandelt das role mining Problem.

Die relevanten Aspekte von role mining sind die formale Definition des Problems, Ansätze zur Lösung des Problems, sowie Methoden zur Validierung von Lösungen. Die Arbeit deckt die genannten Punkte in dieser Reihenfolge ab. Zuerst analysieren wir bestehende Definitionen des role mining Problems anhand der Ansprüche, die reale Unternehmen an ein Zugriffskontrollsystem stellen. Wir zeigen auf, dass bestehende Definitionen diesen Ansprüchen nicht vollständig genügen. Deshalb definieren wir role mining neu als ein Inferenzproblem mit dem Ziel, die inhärente Rollenstruktur der gegebenen Daten zu erlernen. Wenn es gelingt, die Struktur zu entdecken, die den Daten zugrunde liegt, dann

können neue Benutzer im System mit den richtigen Berechtigungen ausgestattet werden. Wir identifizieren diese Generalisierungsfähigkeit von RBAC-Konfigurationen als das wichtigste Qualitätsmass für role mining. Unsere Arbeit stellt die erste Gegenüberstellung und Analyse der verschiedenen Definitionen für role mining dar.

Wir wählen einen probabilistischen Ansatz, um das role mining Problem zu lösen. Der grösste Teil dieser Arbeit ist der Herleitung und Analyse einer Klasse von Wahrscheinlichkeitsmodellen für RBAC gewidmet. Dabei legen wir den Fokus auf zwei ausgezeichnete Modellinstanzen. Das erste Modell nennen wir 'Multi-Assignment Clustering' (MAC), da es die Zuweisung von Benutzern und Berechtigungen zu mehreren Rollen erlaubt. Das zweite Modell ist das 'Disjoint Decomposition Model' (DDM). Es hat eine zweistufige Rollenhierarchie mit Business-Rollen, welche Benutzer gruppieren, sowie technischen Rollen, welche Berechtigungen gruppieren. Dabei können Benutzer und Berechtigungen je nur einer Rolle zugewiesen werden, die Zuweisung von Business-Rollen zu technischen Rollen hat jedoch keine Einschränkung. Beide Modelle werden im Detail hergeleitet und analysiert. Dabei untersuchen wir auch unterschiedliche Modellvarianten für fehlerhafte Zuweisungen, unterschiedliche Annahmen an die Modellparameter, sowie unterschiedliche Algorithmen für die Inferenz der Parameter. Wir stellen experimentell fest, dass beide Modellvarianten zu besseren role mining Lösungen führen als kombinatorische Algorithmen. Darüberhinaus stellen wir fest, dass MAC besser generalisiert als DMM. Für die Anwendung in der Praxis empfehlen wir daher das MAC Modell.

Als eine Erweiterung von MAC schlagen wir einen Algorithmus für 'hybrid role mining' vor. Dabei ist die Eingabe des Problems um geschäftsrelevante Benutzerattribute erweitert. Das Ziel ist es, eine RBAC-Konfiguration zu finden, die gleichzeitig der Struktur der Direktzuweisungen entspricht und die Rollenzuteilung aus geschäftlicher Sicht sinnvoll macht. Wir schlagen verschiedene informationstheoretische Masse vor, um eine Auswahl relevanter Attribute zu treffen. Weiter demonstrieren wir, dass die Berücksichtigung solcher Attribute einen Kompromiss zwischen beiden Teilzielen des Problems darstellt, bei dem es jedoch ein Pareto-Optimum gibt. Unsere Methoden eignen sich, dieses Optimum

zu finden und damit RBAC-Konfigurationen zu lernen, die zum einen gut generalisieren und zum anderen aus der Businessperspektive intuitiv sind.

Schliesslich befassen wir uns ausführlich mit dem Problem, Lösungen für role mining zu validieren. Da die Generalisierungsfähigkeit das wichtigste Evaluierungsmass ist, entwickeln wir eine Methode, die den Generalisierungsfehler einer RBAC-Konfiguration schätzen kann. Diese Methode, 'minimal transfer cost' Prinzip genannt, ist nicht auf die Anwendung für role mining beschränkt, sondern kann für weitere Probleme aus der Klasse des unüberwachten Lernens angewendet werden. Wir demonstrieren das an einer Reihe von Beispielanwendungen.

Weiter untersuchen wir die die Modellvalidierungsmethode 'approximation set coding' (ASC). Wir erklären diese neue informationstheoretische Methode und verwenden sie für zwei Anwendungen. Die erste ist die Auswahl des besten Wahrscheinlichkeitsmodells für role mining aus einer Menge von Modellen. Die zweite ist die Auswahl des optimalen Rangs einer Singulärwertzerlegung (SVD) für das Entrauschen binärer Matrizen. Beide Anwendungen gehören zu den ersten praktischen Implementierungen von ASC, die es gibt.

Contents

Chapter 1

Introduction

The amount of data possessed today by governmental organizations and large enterprises is enormous. As it is often challenging for humans to organize large datasets or even to identify relevant pieces of information, machine learning techniques to automate these processes are increasingly popular. There exist two predominant reasons why humans require machine assistance to organize data: i) the large amount of the data at hand and ii) the unintuitive metric of the data space. This thesis concerns the automated analysis of access-control configurations in computer systems. For this security-relevant data, needed for the internal organization of an enterprise, both problems are apparent: i) With (tens of) thousands of users and resources, the size of these datasets usually exceeds what a human can process. ii) The metric relations between objects, populating the corners of a thousand-dimensional hypercube, are unintuitive for humans (unlike images of faces, for instance).

The favored model in information security to organize large access control systems is Role-Based Access Control (RBAC) [23]. In RBAC, rather than assigning users directly to permissions for accessing resources, one introduces a set of roles and defines two relations: a user-role relation that assigns users to roles and a role-permission relation that assigns roles to permissions. This decomposition facilitates the administration of au-

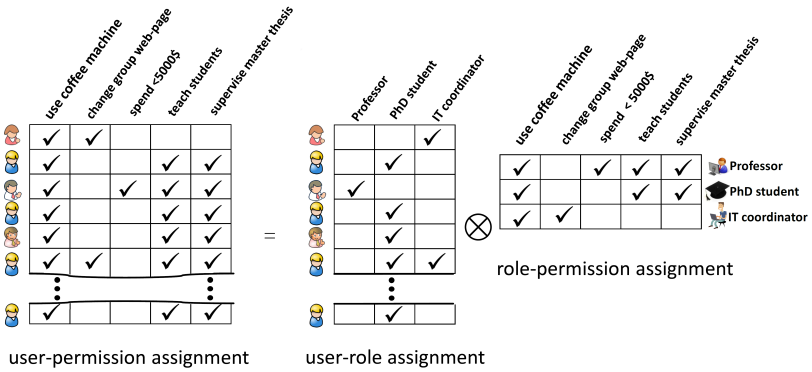


Figure 1.1: Example of an RBAC configuration for a hypothetical research group together with the direct user-permission assignment. This RBAC configuration exactly fits the input data. The Boolean matrix product $\otimes$ encodes that a user gets exactly those permissions that are assigned to his roles.

thorization policies. Since roles are (or should be) natural abstractions of functional roles within an enterprise, the two relations are conceptually easier to work with than a direct assignment of users to permissions.

As most enterprises are older than the concept of RBAC, they must configure RBAC starting from an *existing* access control system. Despite the many advantages of RBAC, this migration step is difficult for large enterprises. This task, called **role engineering** [15], can be approached from two different directions, top-down or bottom-up, each with its own strengths and weaknesses.

Top-down role engineering [21, 61] starts by analyzing an enterprise's business structure. This structure includes business information such as the organizational hierarchy, employees' job descriptions, or their workplace. This information is used to determine the permissions that users should have and to bundle these permissions into roles. The resulting roles are easy to understand from the business perspective as they are

derived from business concepts. However, the business information alone is sometimes insufficient to derive an RBAC configuration that closely corresponds to the existing direct assignment of users to permissions, i.e., the authorizations may change considerably.

In contrast, bottom-up approaches start with the direct assignment of users to permissions. One then analyzes these assignments for patterns, attempting to capture the underlying structure with roles and an assignment of users to roles. Until recently, this process was carried out manually, despite the high incurred costs and the latent security risk due to erroneous assignments to roles. However, bottom-up role engineering can be automated using data mining algorithms. This automated way of role engineering is called **role mining**. Automated approaches are desirable since they have the potential to dramatically reduce both the costs and the security risks.

Research efforts on role mining started in 2003 when data mining techniques for role engineering were proposed in [47]. This research field has expanded rapidly since then. The first approaches all aim at minimizing the RBAC configuration while reproducing the user-permission assignment matrix given as an input. In these early contributions, the problem is defined either as minimizing the distance to the input data for a given number of roles, or as getting below a predefined distance with the minimal number of roles. As these problems are NP-hard [73], the first proposed heuristic algorithms usually create a large set of candidate roles in various combinatorial ways and then minimize the distance of the RBAC configuration to the input data by greedily picking roles from this candidate set.

Three main problems exist with such approaches. First, the closest fit to the input data might contain incorrect user-permission assignments. Practitioners usually assume that some of the existing user-permission assignments are wrong in each system. Ideally, such assignments should not be migrated to RBAC. Second, RBAC configurations achieving the best lossless compression might poorly generalize to new users in the system. It is hard to administrate such configurations if new roles must be designed whenever new users enter the system or when employees change their business role withing the enterprise. Finally, minimizing

the distance between input and RBAC configuration can lead to roles that are hard to interpret from a business perspective. The algorithms might favor synthetic sets of permissions over permissions that resemble business roles, if only the distance to the input is minimized.

We are the first to address such problems using a probabilistic approach to role mining. We define the role mining problem as an inference problem. The task is to find the RBAC configuration that most likely underlies the given access-control data. In experiments with artificially created data and with real-world data, we demonstrate that our approach outperforms existing methods in terms of detecting erroneous user-permission assignments and in terms of generalization ability.

One can distinguish three aspects of role mining:

1. the formal problem definition,

2. the role mining algorithm, and

3. quality measures for the assessment of role mining results.

The first aspect formally defines the goal of "role mining" by specifying what is given, what is assumed, and what must be found. The second aspect concerns the formalization of the approach taken to solve the problem by giving an algorithm. The third aspect addresses how the results are evaluated. In general, all three of these aspects are interrelated and, ideally, they are in agreement. That is, the algorithm should solve the formulated problem in that it returns the best possible result as defined by the quality measure.

This thesis provides a framework for role mining where all three aspects are mutually consistent. We define the role mining problem motivated by the real-world requirements for RBAC. We propose a class of probabilistic models for inferring RBAC configurations from access-control data and analyze inference algorithms. Finally, we address the problem of validating solutions of the role mining problem. We propose quality measures that serve for comparing different solutions and for selecting the appropriate method for role mining.

1.1 Thesis overview

The thesis is organized as follows. We start by defining the role mining problem in Chapter 2. Thereby, we revisit existing definitions and analyze them with respect to the real-world requirements of RBAC. In Chapter 3 we introduce a class of probabilistic models to solve the role mining problem and derive several of its instances. In Chapter 4 and Chapter 5 we introduce two particular model instances. We provide an in-depth analysis of these models and compare them with other methods for role mining. We extend the role mining method from Chapter 4 to hybrid role mining in Chapter 6. This approach combines bottom-up role mining and top-down role engineering in an automated fashion. Finally, in Chapter 7, we propose and analyze several model validation techniques, thereby going beyond the task of role mining.

1.2 Original contributions and publications

Our main contribution is the probabilistic approach to role mining. Individual contributions and publications cover individual aspects of this approach.

- **Role mining as an inference problem:** In [26], we analyzed, for the first time, all established definitions of the role mining problem by relating them to the real-world requirements of RBAC configurations. We redefined role mining as an inference problem and motivated the generalization ability as the primary quality measure for role mining solutions.

- **Probabilistic models for RBAC:** In our first publication on role mining [24], we derived the likelihood of an access-control matrix given a probabilistic RBAC configuration. The starting point of this derivation is the deterministic rule of assigning permissions to users in an RBAC system. Using this likelihood function, we defined a generic class of models for role mining. We investigated a model instance with a two-layer role hierarchy and with the constraint that a user can have only one role.

- **Multi-assignment clustering:** In [71], we solved the technical difficulty of learning a configuration where a user can have multiple roles. The proposed model variant, called multi-assignment clustering (MAC), has no constraints on the number of roles of a user and has a flat role hierarchy. This research was carried out in close collaboration with Andreas Streich who contributed equally in all aspects of the paper.

- **Hybrid role mining:** In [28], we extended the input of the role mining problem to additional business information. Our algorithm uses business attributes of the users to find roles that correspond to these attributes and that optimize the MAC model. This multi-objective optimizer is the first role mining method that learns roles that generalize well and at the same time are intuitive from a business perspective. In this work, Andreas Streich contributed substantially to the experimental analysis and to writing the paper.

- **The transfer cost method:** In [27], we showed how to make cross-validation applicable for unsupervised learning problems like role mining. Using a mapping between training set and validation set enables the data analyst to tune model parameters or select a model from a set of candidate models. This is joint work with Morteza Haghir Chehreghani.

- **First applications of approximation set coding (ASC):** We contributed to the novel framework of ASC in the following way. In [9], we demonstrated how to apply this framework to practical problems like clustering. This is joint work with Morteza Haghir Chehreghani and Andreas Streich. In [25], we demonstrated the first application of ASC to a problem with a continuous and unbounded solution space. Namely, we used ASC for model-order selection for truncated singular-value decomposition.

In all points listed, David Basin and Joachim M. Buhmann greatly contributed by their supervision.

Chapter 2

Definition of the role mining problem

2.1 Motivation

In this first chapter, we provide an introduction to the role mining problem and its different problem definitions. Thereby, we will step back and take stock of some of the central questions in role mining. In particular, *what is the goal of role mining or what should it be?* It might appear that this question is already answered given the growing body of literature on the topic. However, as we will show, no consensus on this fundamental question has been achieved so far and this lack of agreement profoundly effects the way how research in the field is carried out and how results are selected and applied. We will redefine the problem in a way that better reflects the real-world requirements on role mining results. Emphasizing the problem definition is necessary for the following reasons: First, the formal problem definitions suggested in the literature usually only partially reflect actual real-world requirements for role mining. We will review these definitions in Section 2.4 and see that there is a divergence between theory and practice. The problems addressed by researchers are often not the ones faced by actual enterprises migrating to RBAC.

Second, there does not exist consensus on which of the formally defined problems should be solved in practice. In fact, most of the algorithms and quality measures given in the literature often do not fully comply with any existing formal problem definition. This discrepancy indicates that, even though individual researchers have their own understanding of the problem they are solving, a consensus on the problem definition does not yet exist.

Finally, we believe that this lack of consensus limits progress in role mining. It leads to researchers solving different problems and developing different algorithms and quality measures. Not only does this dilute efforts, it makes it difficult to compare approaches. Lacking a common objective, everybody can make his role mining method best by designing his own quality measure! Moreover, the lack of consensus makes it difficult for new researchers to orient themselves in this comparatively immature field.

Given the above, we believe that it is important to start this thesis with a definition of the role mining problem. This chapter is fully devoted to this task. We first introduce the concept of role-based access control (RBAC) in Section 2.2. In Section 2.3, we summarize the most important requirements on role mining. In Sections 2.4 and 2.5 we introduce the existing formal definitions, objective functions, and quality measures and analyze them based on the presented requirements. In Section 2.6, we propose a novel definition of the role mining problem and explain how it reflects the role mining requirements.

2.2 RBAC and notational preliminaries

Role-based access control The basic entities of core RBAC as defined in the NIST-standard for RBAC [23] are:

- $USERS$, the set of users,
- $PRMS$, the set of permissions.
- $ROLES$, the set of roles,

- $UA \subseteq USERS \times ROLES$, a user-role assignment relation,
- $PA \subseteq ROLES \times PRMS$, a role-permission assignment relation, and
- $UPA \subseteq USERS \times PRMS$, a user-permission assignment relation.

As is usual in the context of role mining, we neglect the notion of sessions.

Matrix notation: An assignment relation between entities is conveniently represented by a Boolean matrix where a 1 at the (i, j)-th entry indicates an assignment of i to j and a 0 indicates no assignment. We will denote the matrix representation of a relation A as $\mathbf{M}(A)$. When there are N users, D permissions, and K roles, then $\mathbf{Z} := \mathbf{M}(UA)$ is a $N \times K$ Boolean matrix, $\mathbf{U} := \mathbf{M}(PA)$ is a $K \times D$ matrix, and $\mathbf{X} := \mathbf{M}(UPA)$ is a $N \times D$ matrix. Rows in $\mathbf{X}$ represent users and columns represent permissions. We will use $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{U}$, K, N, and D with this meaning throughout the thesis. Especially in the more mathematical chapters, it will prove useful to only refer to the matrix notation of each entity.

Boolean matrix product: The Boolean product $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ of two matrices $\mathbf{A} \in \{0,1\}^{N \times K}$ and $\mathbf{B} \in \{0,1\}^{K \times D}$ is defined as

$$c_{id} = \bigvee_{k=1}^{K} (a_{ik} \wedge b_{kd}) \ . \tag{2.1}$$

The Boolean matrix product will prove useful since, given a user-role assignment relation UA and a role-permission assignment relation PA, the matrix representation of the equivalent user-permission assignment relation UPA can be expressed as $\mathbf{M}(UPA) = \mathbf{M}(UA) \otimes \mathbf{M}(PA)$.

Role Configurations: We use $RC = (ROLES, UA, PA)$ to denote an *RBAC configuration*. If RC is the output of a role mining algorithm, we will also call it the *role mining result*. Some methods output, in addition to a user-role assignment UA and role-permission assignment PA, a direct assignment relation $DUPA$. For outputs including $DUPA$, we will use the same notation for an RBAC configuration:

$RC = (ROLES, UA, PA, DUPA)$. We will be explicit about which kind of configuration is intended when this distinction is necessary. Direct assignments $DUPA$ are not required for RBAC [23]. However, this relation is frequently used to express assignments that do not fit into the role structure. Moreover, by defining a new role for each of the individual direct assignments, one can, in principle, map such a configuration to standard RBAC (even though such specialized roles are usually not desirable).

δ-Consistency: We will often use the notion of consistency as defined in [73] (in [11] the term "complete" is used instead):

Definition 2.2-1. (**δ-Consistency**)
A user-role assignment UA, role-permission assignment PA, and user-permission assignment UPA are δ-consistent if and only if

$$\|M(UA) \otimes M(PA) - M(UPA)\| \leq \delta \ ,$$

where $M(UA)$, $M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA, and UPA respectively and $\|.\|$ is the L_1 norm for matrices with $\|M\| = \sum_i^m \sum_j^n |M_{ij}|$.
If an RBAC configuration is 0-consistent with the direct assignments UPA, it is usually just called "consistent". Note that this is the original definition of consistency. A better definition would linearly scale δ with $N \cdot D$ to make the criterion scalable to access-control matrices of different dimensions.

2.3 Requirements for role mining

In order to reason about what role mining should achieve, it is necessary to first specify what real-world enterprises require of RBAC configurations. Requirements for role mining are then supposed to favor such configurations. This transition step from real-world requirements to an abstract problem space enables us to analyze and compare different problem definitions with respect to the same frame of reference. In

the following, we briefly explain the main requirements for role mining. Since we aim to achieve a consensus on the goal of role mining, we restrict ourselves to the most fundamental requirements, which we expect most researchers and practitioners to agree with.

Requirement 1: Provisioning. An RBAC configuration must provide the users with all necessary permissions.

On a technical level, this means that an RBAC configuration must provide users with the privileges they need to perform actions on the resources they require.

Requirement 2: Security. An RBAC configuration must be secure.

This means that the configuration should conform to the security policies of the enterprise such that no user can access resources that he is not authorized to access, that separation of duty constraints are fulfilled, etc. As a result, if there are errors in the direct assignment *UPA*, role mining should be able to detect them in order to prevent these errors from being migrated to the RBAC configuration.

From a more practical perspective, RBAC configurations should comply to audit requirements and simplify security audits. The configuration should, for instance, make it easy to answer questions like: "Why does x have access to y?" This is closely related to Requirement 3.

Requirement 3: Maintainability. An RBAC configuration should be as easy to maintain as possible.

Maintainability lowers administration costs and helps administrators to work with the system. Here, three main aspects come into play: **minimality**, **interpretability**, and **generalization ability**.

First, most researchers consider a small RBAC configuration easier to maintain than a large one. Such a configuration has, for instance, few roles or few assignments.

Second, a configuration that is easy to interpret is easier to maintain than one where the roles are artificial and unintuitive. Interpretability can be achieved if the RBAC configuration is in agreement with the

business processes and with the business properties of the users, such as their associated departments, locations, or tasks. Ideally, a user's roles correspond to the business roles that he works in and the two entities can be identified. In this case, one could even use the name of the business role (e.g. "receptionist, head office") as the name of the access-control role. This would enable system administrators and business employees to speak in the same language.

Finally, a configuration that allows administrators to easily add new users is easier to maintain than one where roles must be added or modified whenever new employees are added to the system. Configurations with a high generalization ability are comprised of roles that are stable under employee fluctuations. This substantially increases maintainability since the administrative effort to add new users or to move them within the enterprise is minimal if such actions can be made without requiring the RBAC configuration to be modified. Roles that generalize well are usually in agreement with the business processes of the enterprise in that they endow the user with necessary permissions independent of who is engaged in these processes.

Minimality, interpretability, and generalization ability are all interrelated. As already noted, interpretability increases maintainability. Moreover, it simplifies security audits because provisioning is better supported by meaningful roles. Having roles that conform to business processes requires that the roles contain the permissions needed to carry out these processes, but not more (i.e. least privilege). Since roles that are interpretable and generalize well usually reflect the business processes, they also fulfill the provisioning requirement. Moreover, security requires that the risk that future insecure assignments are granted based on the current RBAC configuration is low. To satisfy this requirement administrators must understand the role configuration, thereby avoiding errors. Hence, an intuitive and maintainable configuration increases security.

Note that minimality cannot always be jointly achieved with interpretability and generalization ability. The smallest system configuration is often not identical to the configuration with the highest interpretability and generalization ability. Hence, one must prioritize. Which aspect is more important? As long as administrators can understand the config-

uration and the configuration can be used with new users without many modifications, it is not problematic if the configuration is larger than the minimized configuration could, in principle, be. If, in turn, minimization comes at the price of low interpretability or low generalization ability, then the system is harder to administer. Hence, finding roles that are interpretable and generalize well should be favored over minimizing the configuration's size. We will therefore give minimality a lower priority, when we analyze problem definitions in terms of the maintainability of their solutions.

2.4 Existing definitions

2.4.1 Role mining definitions in the literature

In this section we review the existing formal definitions of the role mining problem. We restrict our attention to formal definitions in the sense that they have a well-defined mathematical meaning. By distinguishing between problem definition, algorithm, and quality assessment as explained in the introduction, we can also investigate the goals formulated in the literature that are not formally defined but rather implicitly given (see Section 2.5).

The first three formal definitions of the role mining problem are proposed in [73].

Definition 2.4-1. (**Basic Role Mining Problem (RMP)**)
Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles $ROLES$, a user-role assignment UA, and a role-permission assignment PA, consistent with UPA, that minimizes the number of roles k.

Definition 2.4-2. (δ**-approx RMP**)
Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles $ROLES$, a user-role assignment UA, and a role-permission assignment PA, δ-consistent with UPA that minimizes the number of roles k.

Note that Definition 2.4-1 is a special case of Definition 2.4-2 with $\delta = 0$.

Definition 2.4-3. (**min-noise RMP**)
Given a set of users U, a set of permissions $PRMS$, a user-permission assignment UPA, and the number of roles k, find a set of k roles $ROLES$, a user-role assignment UA, and a role-permission assignment PA, minimizing

$$\|M(UA) \otimes M(PA) - M(UPA)\|,$$

where $M(UA)$, $M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA, and UPA respectively and $\|.\|$ is the L_1 norm.

In subsequent work [51], the problem is defined as minimizing the number of assignments of a consistent RBAC configuration.

Definition 2.4-4. (**min-edge RMP**)
Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles, $ROLES$, a user-role assignment UA, and a role-permission assignment PA, consistent with UPA and minimizing $|UA| + |PA|$.

A variant of Definition 2.4-2 that also takes the role hierarchy of an RBAC configuration into account is considered in [36].

Definition 2.4-5. (**Role Hierarchy Mining Problem**)
Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles, $ROLES$, a user-role assignment UA, a role-permission assignment PA, and a complete role hierarchy, $RH = G(V, E)$, that are consistent with UPA and that minimize $k + |E|$.
Here the graph $G(V, E)$ defines the role hierarchy.

We analyze the above definitions in Section 2.4.2 and relate them to associated quality measures in Section 2.5.

Hybrid role mining. Role mining approaches that take business information, also called "top-down information", together with UPA as input are called hybrid role mining methods. Top-down information often includes, for instance, locations, department affiliations, or task descriptions of the users. Alternatively it might refer to the confidentiality levels

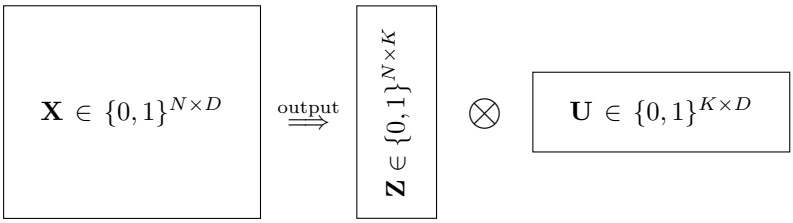


Figure 2.1: Dimensions of input data and output of the problems defined in Definitions 2.4-1–2.4-7.

or the criticality of permissions or to security policies. Generally speaking, top-down information complements the data on *USERS*, *PRMS*, and *UPA*. The representation of top-down information can be of different forms. It can be of a categorical nature, a ranking type, relational, or numerical. For example, the location of users at one of the enterprise branch offices could be modeled as categorical data (e.g. $l_i = p$, if user i has location p). Rank-type top-down information can be found, for instance, in military domains. One example of relational top-down information is a separation-of-duty policy represented by a binary $D \times D$ matrix $\mathbf{S}$, where the entry $S_{d_1 d_2} = 1$ indicates that permissions d_1 and d_2 should never be jointly assigned to the same role. Finally, the budget ceiling of a user (e.g. $b_i = \$10,000$) is an example of numerical top-down information. Throughout this paper, we will denote any kind of top-down information by *TDI*. Depending on what kind of top-down information is available, *TDI* can have specific representations and values.

Besides our own work [26], we did not find a formal problem definition for hybrid role mining in the literature. However, several methods for hybrid role mining have been proposed [12, 28, 57]. Some authors also denote by hybrid role mining those bottom-up methods that are followed by, or used together with, the manual inspection of an expert who knows the business properties (i.e. in [29]). However, this naming convention is not consistent with the use of role mining for fully automated methods. We advocate instead using a term like "machine-assisted role engineering".

2.4.2 Analysis of role mining definitions

The existence of multiple definitions for the role mining problem already suggests that there may be not a universal one. Given this, how do the above definitions differ and in which situations should they be used? Moreover, what advantages and shortcomings do they share?

The above definitions are all based on a notion of minimality of the RBAC configuration. They vary in just two aspects: How configuration size is measured (number of roles, number of assignments, size of the hierarchy, etc.), and whether the RBAC configuration is required to be consistent with the given direct assignments *UPA*.

We first consider the question of whether an RBAC configuration should be 0-consistent with *UPA*. There seems to be no consensus on this question in the literature, as can be seen in the two bottom rows of Table 2.1 given in Section 2.5. Ultimately, the question boils down to two mutually exclusive assumptions on the properties of *UPA*:

1. **Perfect information:** *UPA* grants each user exactly the permissions he needs and no more.

2. **Erroneous assignments:** Some of the assignments in *UPA* might be unnecessary or missing.

Clearly, under the first assumption, consistency should be a part of the definition of role mining. Under the second assumption, consistency should be dropped.

The second assumption appears to be more realistic. Discussions with practitioners indicate that they share this assumption. *UPA* is influenced by human factors. For example, a user may require a privilege that he originally did not posses but which he needs to perform a special, exceptional task. Once given additional permissions, users often keep them because they do not view this as a problem. More concretely, a user may change his position within the enterprise and keep some of his old permissions to help his successor during the transition period. Later, he may forget to have these permissions removed. Errors may also arise because administrators simply make mistakes and wrongly assign privileges.

In the remainder of this thesis, we denote all assignments in *UPA* that are modified due to such exceptions or mistakes as **exceptional assignments** or **noise**. Accordingly, we denote the actions that lead to exceptional assignments as modification processes or **noise processes**. Given that *UPA* can be modified by noise, we advocate using definitions that do not require RBAC configurations to be 0-consistent. Dropping the notion of consistency reduces the set of compatible existing formal definitions to Definitions 2.4-2 and 2.4-3.

Another problem is that the definitions above that do *not* aim for a consistent solution require additional information, which is usually not available at the time the problem is to be solved. For Definition 2.4-2, one must know the number of assignments δ that can be either dropped or additionally given. Knowing δ requires knowledge of the percentage of exceptionally given or missing assignments in *UPA*. For Definition 2.4-3, the number of roles k is needed as input. Both δ and k are difficult to know. The upper-bound of k is the Boolean rank or Schein-rank of $M(UPA)$, which would conform to a consistent role mining solution and cannot be efficiently found [52]. This problem is equivalent to Definition 2.4-1, which is shown to be NP-hard in [73]. If a small δ is required, k can be considerably smaller than the Boolean rank.

All definitions share the notion of minimality. As discussed in Section 2.3, small configuration sizes are beneficial since they can improve maintainability. However, if one tries to minimize the number of roles or the number of assignments (either for consistent RBAC configurations or RBAC configurations with a given approximation error δ), the roles and role assignments to users are then defined such that the roles cover as many of the given direct assignments as possible without granting extra permissions. As a result, the roles can end up quite synthetic and unintuitive. As Vaidya et. al. note [74]: "Minimality is a good notion since it allows one to formally define the problem" but "such a role discovery process can only serve as a guideline to security administrators to launch RBAC."

In summary, each of the above definitions runs contrary to at least one of the basic requirements for role mining as given in Section 2.3. The notion of consistency is not reasonable and setting k or δ in ad-

vance is dangerous: Setting δ too low would result in migrating some of the erroneously granted assignments to the RBAC configuration. This runs contrary to the principle of least-privilege and thus does not fulfill Requirement 2 for security. Setting δ too high would result in an RBAC configuration that does not provide all users their needed permissions. This would violate Requirement 1 for provisioning. Finally, the goal of minimality contradicts the requirement for intuitive roles that generalize well and that correspond to the business properties of the enterprise. Role mining is, after all, not just a compression problem!

2.4.3 Related problems

In this section, we provide an overview of methods for exploratory analysis of Boolean data. The described approaches have been developed within different research areas and have different objectives. However, they all aim to produce a structured representation of given binary data that is similar to an RBAC configuration. The research areas include, among others, association-rule mining, formal concept analysis, clustering, dimension reduction, latent feature models, and database tiling. We distinguish between methods that search for an exact representation of the data and methods that approximate the representation in some way. In the following, we review several related problem formulations and compare the approaches used to solve them.

Exact Boolean matrix decomposition and equivalent problems. These methods aim for an exact Boolean factorization of the input matrix. The earliest formulation of such problems is presumably the set-cover problem (also called set basis problem) presented by [34] and [13].

Definition 2.4-6. (**Set-Cover Problem**)
Given a set of finite sets $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, find a basis $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K\}$ with minimal cardinality K such that each $\mathbf{x}_i$ can be represented as a union of a subset of $\mathbf{U}$.

All sets in $\mathbf{X}$ have a vector representation in a D-dimensional Boolean space, where $x_{id} = 1$ indicates the membership of item d in the respec-

tive set. D is the cardinality of the union of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$. The matrix $\mathbf{Z} \in \{0,1\}^{N \times K}$ indicates which subsets of $\mathbf{U}$ are used to cover the sets in $\mathbf{X}$: $z_{ik} = 1$ indicates that $\mathbf{u}_k$ is used to cover $\mathbf{x}_i$. Using this notation, the set-covering problem is equivalent to finding an exact Boolean decomposition of a binary matrix $\mathbf{X} = \mathbf{Z} \otimes \mathbf{U}$ with minimal K which is the Basic Role Mining Problem in Definition 2.4-1. Therefore, we used the same variable names in Definition 2.4-6. In [7] the set cover problem is shown to be equivalent to Boolean factor analysis, where each factor corresponds to a row of $\mathbf{U}$. The authors in [46] show that the factors together with the objects assigned to them can, in turn, be regarded as formal concepts as defined in the field of Formal Concept Analysis (FCA) [31]. The set-cover problem is NP-hard and the corresponding decision problem is NP-complete [70]. Since the set-cover problem is equivalent to the other problems, Boolean factor analysis, finding the exact Boolean decomposition of a binary matrix, FCA, and the Basic Role Mining Problem, these problems have the same complexity. Approximation heuristics exist and are presented below.

Approximate Boolean matrix decomposition. As often some of the entries in a matrix $\mathbf{X}$ are random, its approximate decomposition is sometimes more useful than an exact one.

Definition 2.4-7. (**LCP: Minimal Deviation for given** K)
For a given Boolean $N \times D$ matrix $\mathbf{X}$ and a number $K < \min(N, D)$, find an $N \times K$ matrix $\mathbf{Z}$ and a $K \times D$ matrix $\mathbf{U}$ such that the deviation $||\mathbf{X} - \mathbf{Z} \otimes \mathbf{U}||$ is minimal.

In this formulation from [54], the size of the matrix $\mathbf{u}$ is fixed and the reconstruction error is to be minimized. This problem is equivalent to the min-noise RMP. The norm in both formulations is usually the Hamming distance. Both problems are NP-hard as shown in [73].

2.4.4 Existing algorithms

Numerous algorithms for role mining exist. In the algorithm proposed in [75], all existing users are initially considered as candidate roles. Thus,

each candidate role consists of all permissions that are assigned to a particular user. Afterwards, candidate roles are picked in a greedy manner to determine the final set of roles.

In [81], the roles are also represented as sets of permissions. Candidate roles are generated and then merged, split, or placed in a role hierarchy, as determined by a small set of rules. A similar procedure is proposed in [67]. But there, roles and permissions are represented as sets of users. Thus, using the notation above, a role k is a column k in $\mathbf{z}$. The initial roles are constructed from existing permissions. Namely, an initial role is the set of users that are assigned to a given permission (a column in $\mathbf{x}$). Again, the initial roles are iteratively merged, split, or placed in a role hierarchy according to the cardinality of the intersections of the roles.

In [47], roles are computed by randomly merging sets of permissions. The merging process is iterated until there is a complete tree structure of role proposals. From this tree, the final roles are selected by analyzing the number of associated users.

The method in [57] applies an algorithm from formal concept analysis (see [31]) to construct candidate roles (rows in $\mathbf{u}$). The method in [11] creates candidate roles using a method from association-rule mining [2]. In both approaches, candidate roles are greedily assigned to users such that a given cost function is minimized. The technique presented in [73] uses an improved version of the database tiling algorithm from [37] which, in contrast to the method presented in [2], avoids the construction of all concepts by using an oracle for the next best concept.

Approaches for related problems. The LCP is NP-hard. Like for role mining, heuristic methods to find approximate solutions usually construct candidate sets for the rows of the matrix $\mathbf{U}$, and then greedily pick candidates such that, in each step, the reconstruction error is maximally reduced. For the set covering problem defined in [13], the candidate set is the set of all possible formal concepts. For the approximate decomposition problem described in [54], candidates are computed using association rule mining as presented in [1]. A predefined number of candidates is then iteratively chosen and assigned to the objects such that,

in each step, the dataset is optimally approximated. We will refer to this algorithm, originally presented in [54], as the Discrete Basis Problem solver (DPBS) and use Miettinen's implementation of DBPS in some of our experiments. In the greedy algorithm proposed in [7], the construction of a large candidate set is avoided by iteratively constructing the next best candidate.

2.5 Existing quality measures

As pointed out in the introduction, there are three aspects that explicitly or implicitly determine the goals of role mining: The formal problem definition, the algorithm employed, and the quality measure used for assessment. A thorough analysis of the role mining problem should consider all three aspects.

In addition to formal problem definitions, many quality measures and algorithms for bottom-up role mining have been introduced in the literature. Often, the associated publications do not include a formal problem definition (see Table 2.1). However, in many of these cases, the quality criterion given provides an implicit problem definition. Note, in this regard, that many quality measures are contradictory and correspond to different (implicit) problem definitions. In this section, we survey all existing quality measures for role mining and the most prominent algorithms. A summary of the most prevalent concepts is provided in Table 2.1.

2.5.1 Problem definition vs. algorithm vs. assessment

First, we would like to compare these three aspects and explain why it is beneficial to distinguish them.

Definition vs. assessment The problem definition and the quality measure used for assessment are related in a simple way: an adequate quality measure quantifies if (or how well) the defined problem was

solved. This relation is undirected: Associated to each given measure is a hypothetical problem that one tries to solve. Hence, it is desirable to have a common definition of the role mining problem. The associated quality measure for solutions then enable us to compare role mining approaches.

Why should one distinguish definition and assessment? First, for some of the problems defined, it is computationally intractable to decide *if* they were solved given a candidate solution. For instance, checking if a role mining result solves the basic RMP from Definition 2.4-1 is NP-complete [73] (this is the decision version of Problem 2.4-1). For computationally intractable problems, one cannot even efficiently check *how good* the result is because evaluating how close one is to the solution requires first knowing the solution. For such problems, one can only compare pairs of RBAC configurations in order to see which one is closer to the solution (in the case of basic RMP, this would correspond to comparing the number of roles). Such a comparative assessment already deviates from the original problem definition. Moreover, there are far more quality measures than problem definitions (compare columns 2 and 3 in Table 2.1) thus making it convenient to group the measures in a distinct category.

Definition vs. algorithm Some role mining approaches are not defined in terms of an algorithm, but rather by defining an objective function or cost function that must be optimized. Since the optimization of a cost function is again an algorithm, we do not further distinguish the two cases. The term "algorithm", as we use it here, shall also cover such approaches.

If a role mining algorithm is based on optimizing an objective function, this function determines the problem being solved. However, often there are deviations between the problem one wants to solve and the algorithm used. First, in most cases, the problems defined are NP-hard (this was shown in [73] for Definitions 2.4-1 – 2.4-3). Therefore the algorithm used often employs a heuristic, which does not actually solve the given problem but rather a computationally tractable approximation. The algorithm thus solves the approximating problem and not the orig-

inal problem. Second, some problem definitions do not directly suggest a way to solve the problem. The definition that we propose provides an example of this, as will be discussed in Section 2.6.1.

Algorithm vs. assessment There is a natural quality measure for role mining algorithms that optimize an objective function $f(RC)$. For a given RBAC configuration RC', one can simply compute the function value $f(RC')$ (or the difference between $f(RC')$ and the optimum, if known) to quantify the solution's quality. This is often used in practice (see Table 2.1). However, since the problem definition and the algorithm often deviate, and since the quality measure should refer to the problem definition rather than to the algorithm, this might not always be the best choice.

Given possible deviations between the three aspects, we put forth that research on role mining approaches should ideally answer all three questions. What is the problem? What algorithm is used to solve it? And how is the solution assessed?

2.5.2 Measures corresponding to existing definitions

There are obvious measures that evaluate how well the five problems defined in Section 2.4.1 are solved. For Definitions 2.4-1 and 2.4-2 this is the number of roles, for Definition 2.4-3 the approximation error δ, for Definition 2.4-4 the number of assignments, and for Definition 2.4-5 the number of roles plus the size of the role hierarchy. These measures are employed in the papers that give the problem definitions [36, 51, 73]. These definitions have the advantage that solutions can be easily assessed. However, the notion of minimality, which is the key criterion for these definitions, should not be the primary requirement for RBAC configurations as explained in Section 2.4.2.

Weighted structural complexity

The most frequently used quality criterion is the *weighted structural complexity* (*wsc*) [50]. The *wsc* criterion measures the size of an RBAC con-

figuration as a linear combination of a set of individual measures of the size (such as number of roles, number of assignments, etc.).

Definition 2.5-1. **Weighted Structural Complexity**
Given $W = (w_r, w_u, w_p, w_h, w_d) \in \mathbb{R}^5$, the weighted structural complexity $wsc(RC, W)$ of an RBAC configuration $RC = (ROLES, UA, PA, DUPA)$ is

$$\begin{aligned} wsc(RC, W) &= w_r|R| + w_u|UA| + w_p|PA| \\ &\quad + w_h|t_{reduce}(RH)| + w_d|DUPA| \,. \end{aligned}$$

Here, RH is the role hierarchy and $t_{reduce(RH)}$ denotes the transitive reduction of RH, which is the minimal set of relationships that encodes the same hierarchy as in RH. Note that, in contrast to the original definition in [50], we have dropped some of the constraints on the weights W (such as being natural numbers) since these parameters must be specified externally anyway. Moreover real weights are also reasonable.

With appropriate weights, $wsc(RC, W)$ equals the quality measures given in Section 2.5.2. Hard constraints, such as consistency, can be imposed by setting the corresponding weights (w_d) to a very large number (in the literature this is often formalized as ∞). Another quality measure that is similar to $wsc(RC, W)$ is the cost function defined in [11]. There, the two terms $|t_{reduce}(RH)|$ and $|DUPA|$ are dropped and, instead, an additional term $\sum_k c(r_k)$ is introduced that penalizes or rewards specific roles r_k with a function $c(r)$ that must be predefined by the role miner.

Note that the quality criterion wsc could be used to formally define the role mining problem. The problem to minimize $wsc(RC, W)$ is similar to the min-edge RMP in Definition 2.4-4. The difference is that the edge measure $|UA| + |PA|$ in the min-edge RMP is replaced by wsc.

The high flexibility of wsc due to the unspecified weights W is, at the same time, its biggest shortcoming. There are different proposals how to set the weights (of wsc and costs) in the literature [11, 12, 20, 50, 57, 58], and, moreover, multiple weight configurations are considered in the same publications. However, to the best of our knowledge, there exists little discussion on how to choose the weight given the input data.

The problem is that without specific weights W, $wsc(RC, W)$ fails to answer the question of what is a good quality criterion. It is obvious

that such a criterion could be any linear combination of the individual quality measures. One has just shifted the question of how the original measures should be defined to the question of how the weights should be chosen. Thus, $wsc(RC, W)$ defines an entire family of quality measures. Moreover, the full space of possible quality measures is not covered since all combinations of nonlinear terms are neglected. Finally, the same considerations on the notion of minimality apply, as discussed in Section 2.4.2.

Comparison with original roles

For experiments with artificially created data, we found four variants of a quality measure that compares the resulting set of roles with the ones that were used to create the data. The method used in [47] checks whether all original roles used to construct the data are discovered by the role mining method. The fraction of roles that equal the original ones is used for assessment in [75] and [76].

A relaxed version of this comparison of the computed set of roles with the original roles is proposed in [58]. There, for each role discovered, the maximal Jaccard-coefficient over all pairings with original roles is taken. The final distance to the original roles $\mathbf{U}_{orig}$ is then

$$d(\mathbf{U}, \mathbf{U}_{orig}) = \frac{1}{K} \sum_{k} \max_{k'} (\text{Jaccard}(\mathbf{u}_{k*}, \mathbf{u}_{k'*})) \ , \tag{2.2}$$

with

$$\text{Jaccard}(\mathbf{u}_{k*}, \mathbf{u}_{k'*}) = \frac{\sum_d u_{kd} u_{k'd}}{\sum_d u_{kd} + u_{k'd} - u_{kd} u_{k'd}} \ . \tag{2.3}$$

In the context of role mining, the Jaccard-coefficient was first used as an optimization criterion for role mining scenarios where some deployed roles are given in advance [36, 74].

In [71], we propose a measure based on the average Hamming distance between the original and the mined roles. In contrast to (2.2), where the $\max_{k'}$ operation holds for each computed role $\mathbf{u}_{k*}$ (such that each role is compared to the most similar original role), in [71] we take a single permutation of all roles that gives a one-to-one mapping between

	definition	solution algorithm	quality measure
size of RBAC configuration (|R|, |UA|, |PA|, ...)	[36, 51, 73]	[36, 51, 67] [72, 81, 74]	[36, 51, 74, 81]
size measure combinations (wsc or combined "costs")		[11, 12, 20] [50, 57, 58]	[11, 12, 20] [50, 57, 58]
comparison with original roles (if known)		[36, 74]: given a few deployed roles	[24, 36, 47, 58] [71, 74, 75, 82]
agreement with top-down information		[12, 28, 57]	[12, 24, 28] [29, 57]
0-consistency with *UPA*: required	[36, 51, 73]	[11, 12, 20, 36] [74, 81, 51]	
0-consistency with *UPA*: not required	[73]	[24, 28, 50, 57] [67, 71, 75, 82]	

Table 2.1: Usage statistics for the most prevalent concepts used for the definition, algorithms, and quality measures of the role mining problem.

the discovered roles and the original roles. This global permutation prevents any two discovered roles $\mathbf{u}_{k_1*}$ and $\mathbf{u}_{k_2*}$ from being compared to the same original role $\mathbf{u}_{k'*}$. We provide a more detailed explanation of this measure in Section 4.4.1.

As we will see later, these four measures could serve as quality criteria that would correspond to the new definition of the role mining problem as advocated in Section 2.6.1. The limitation of such measures is clear: they do not work if the real roles are not known. In real life this is almost always the case!

2.6 Defining role mining as an inference problem

In this section we propose a novel definition of the role mining problem. First, we explain the assumptions underlying our definition. Then we give a general definition that covers the bottom-up role mining problem as well as hybrid role mining. The bottom-up problem will be a special case of the general problem, where the input differs but the goal remains the same. Afterwards, we show that a solution to the problem, as it is

defined here, fits the requirements described in Section 2.3. Moreover, we describe how the problem could be approached and propose quality measures for assessing solutions.

2.6.1 Assumptions and problem definition

The input of the role mining problem is a set of users $USERS$, a set of permissions $PRMS$, a user-permission assignment relation UPA, and, depending on its availability, top-down information TDI. Our problem definition is based on three assumptions about the generation process of UPA and we begin by motivating these assumptions.

Assumption 1: Exceptions exist. In Section 2.4.2, we saw that it is reasonable to assume that there are exceptions in the relation UPA, which arise from modification processes. We formalize this assumption by defining an unknown relation UPA', without exceptions. The relation UPA' is then perturbed by modification processes, leading to the observable relation UPA. In Section 2.4.2, we gave examples of different modification processes that can lead to perturbations. We abstain from making further assumptions on them such as, for instance, the fraction of exceptions δ. We consider the perturbations as unknowns and simply assume their existence.

In practice, one hopes that the perturbations do not fully obscure the data's regularities and thus role mining is still feasible. For given data, the fraction of perturbed assignments influences the difficulty of the problem, but does not change the goal of role mining. Some of the perturbation processes might be deterministic, whereas others might be random. In order to account for random modifications, we assume that UPA is a random variable that is drawn from a probability distribution $p(UPA|UPA')$ that is conditioned on UPA'. This formulation includes the case where modifications are due to a deterministic function $UPA = f(UPA')$, which is expressed by simply setting $p(f(UPA')|UPA') = 1$.

Assumption 2: An underlying RBAC configuration exists. The second assumption is that UPA' was induced by an unknown RBAC configuration $RC^* = (ROLES^*, UA^*, PA^*)$, where "induced" means that

$UPA' = UA^* \otimes PA^*$. This is not a strong assumption. We believe that most researchers and practitioners involved in role mining actually make this assumption implicitly. To search for roles implicitly assumes that they are there to be found. Said another way, searching for roles in direct assignments between users and permissions only makes sense if one assumes that the data could, in principle, be organized in such a structured way. Of course, one might try to find a role structure in assignments that are completely random (i.e., each individual assignment is an exception and there are no structural dependencies between them). But what one finds then are random roles without any business semantics; synthetic sets of permissions will emerge that are found only because exceptional assignments are randomly aggregated in a way that mimics structure.

Assumption 3: Top-down information influences the underlying RBAC configuration. The third assumption is that a relationship exists between RC^* and the top-down information TDI. We assume that RC^* reflects the security policies of the enterprise in the sense that RC^* complies to these policies. Moreover, the business properties of the enterprise influence the generation of RC^* and therefore RC^* reflects these properties. For role mining, not all top-down information might be available. In practice, the role miner has only a subset of this information, for example, the affiliations of users to organizational units. Sometimes one has no TDI at all. Since the unknown part of TDI might still have influenced RC^*, we only assume that RC^* was influenced by *partially* given TDI.

The generation process of UPA, under Assumptions 1-3, is sketched in Figure 2.2. The entities that are input for role mining are drawn in black boxes and the unknown entities are gray.

Given the above assumptions on UPA's generation process, we propose the following definition of the role mining problem.

Definition 2.6-1. **Inference RMP**
Let a set of users $USERS$, a set of permissions $PRMS$, a user-permission relation UPA, and, optionally, parts of the top-down information TDI be given. Under Assumption 1-3, infer the unknown RBAC configuration $RC^* = (ROLES^*, UA^*, PA^*)$.

Our definition, together with the assumptions on *UPA*'s generation process, provides a unified view of bottom-up and hybrid role mining. In both cases, the RBAC configuration RC^* must be discovered. In both cases, RC^* is assumed to induce *UPA* (modulo perturbations) and in both cases RC^* is assumed to be influenced by top-down information. The two cases only differ in terms of the availability of top-down information *TDI*. In hybrid role mining, a non-zero fraction of all top-down properties that influenced RC^* is available. When no *TDI* is given, the problem reduces to bottom-up role mining. Note that, in such cases, the goal still remains the same: the solution of Problem 2.6-1 solves the bottom-up problem as well as the hybrid role mining problem. Only the input differs. The assumption that RC^* is (partially) influenced by *TDI* is also reasonable for the pure bottom-up role mining problem. Whether *TDI* actually influences RC^*, does not depend on the availability of such data for the role miner. Available *TDI* is desirable since it can provide additional evidence of what RC^* might be and thus can be used in role mining.

There are two challenges faced when using this definition in practice. First, it does not provide an objective function to optimize. That is, the problem definition does not itself indicate how to solve the problem. Hence, some creativity is required to devise an algorithm or an objective function for this problem. Second, there is no obvious quality measure that can be easily computed. To validate that the problem was solved, one must know the hidden underlying RBAC configuration RC^*. Except for experimental scenarios with artificially created data, this information is rarely available. However, these challenges do not invalidate the problem definition. In Section 2.6.3, we explain how solutions could be found and we propose a quantitative quality measure that can be even used when RC^* is unknown.

2.6.2 Relation to RBAC requirements

We believe that Definition 2.6-1 is the appropriate definition for the role mining problem. We support this by showing that a solution to Problem 2.6-1, which is the RBAC configuration RC^*, fits well the re-

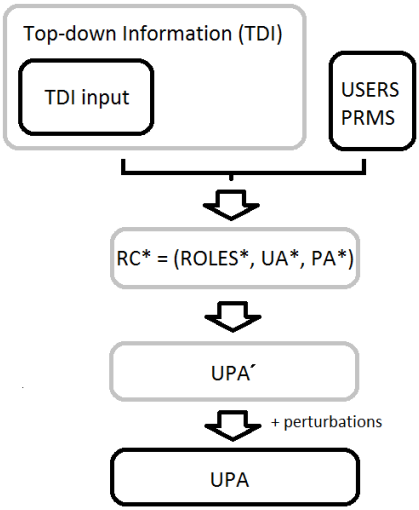


Figure 2.2: Scheme of the assumed generation process of the direct user-permission assignment *UPA*. Grey entities are unknown and black ones are given as input for role mining. For pure bottom-up role mining, no top-down information is given at all.

quirements on role mining given in Section 2.3. Intuitively, since RC^* is generated by the security policies and the business properties of the enterprise, RC^* complies with the security policies and reflects the business properties and thereby naturally fulfills the above requirements.

To explain this in more detail, consider the generation process of *UPA*, sketched in Fig. 2.2. Top-down information influences the generation of RC^* in that the administrators account for TDI when configuring the access-control system and when changing the configuration. The RBAC configuration RC^* underlying the direct assignments thus reflects TDI. As a consequence, if a role mining solution results in RC^*, then this configuration reflects the business processes and the users' business features and complies with the security policies. When RC^* reflects

TDI, then RC^* satisfies Requirements 1-3. Requirement 1, provisioning, is satisfied because when the RBAC configuration reflects the business processes, users can take part in these processes by being assigned to the roles that satisfy the provisioning requirement. RC^* satisfies Requirement 2, security, since it complies with the security policies of the enterprise. Requirement 3, maintainability, is satisfied because when the RBAC configuration reflects the business features of the users and reflects the business processes, the roles are usually intuitive for humans that know these processes and features and the roles also generalize well. Understanding the business semantic of the roles makes the RBAC configuration easy to maintain. Hence, by defining the goal of role mining as inferring RC^*, a solution of the problem satisfies these fundamental role mining requirements.

2.6.3 Associated algorithms and quality measures

In this section, we only briefly point to role mining algorithms that aim at solving the inference role mining problem and to quality measures for validating solutions. The remainder of the thesis will cover these techniques in detail.

A preview of probabilistic role mining The best way to search for the hidden RBAC configuration RC^* is to search for the configuration with the highest probability given UPA, namely $p(RC|UPA)$. This can be achieved using the maximum likelihood principle. We take this approach, for instance, in Chapter 4. For a given UPA, maximum likelihood denotes the RBAC configuration RC_{max} that maximizes the function $p_L(UPA|RC)$, where $p_L(UPA|RC)$ is the probability that UPA would be generated if RC were the underlying RBAC configuration. Maximum likelihood assumes that the RBAC configuration that makes the generation of UPA most likely has the highest probability to be the real underlying RBAC configuration. For a given UPA, this assumption holds if the prior probabilities $p(RC)$ of all possible RBAC configurations

RC are equal. One can see this by employing Bayes rule:

$$p(RC|UPA) \quad = \quad \frac{p_L(UPA|RC)p(RC)}{p(UPA)}\,. \tag{2.4}$$

For maximum likelihood, one must account for the assumption that $p(RC)$ is the same for all RC. Some RBAC configurations might have a higher probability than others per se, that is, without considering a particular UPA. If such prior knowledge is available, then it must be modeled too. In Chapter 5, we use a Dirichlet process prior [5, 22] to model the prior assumption that few roles are more likely than many roles.

Both approaches require the likelihood function $p_L(a|b)$, which must be modeled such that it captures the generation process of the RBAC configuration. In Chapter 3, we derive such a probabilistic model from the logical structure of $RBAC$. The models in Chapter 4 and Chapter 5 are then special cases and extensions of the core model.

Hybrid role mining requires that TDI is available in the first place. Moreover, not all available TDI might be useful for role mining. Consider, for example, the users' home address or gender. TDI that contradicts the role structure present in UPA could be even misleading since it usually renders the underlying optimization problems ill-conditioned. One must therefore understand if (and how) a particular kind of TDI influences RC^*. Ideally, one would model this with a probability distribution $p(RC|UPA, TDI)$ that includes TDI.

To the best of our knowledge, such a model has not been discussed in the literature. All hybrid role mining approaches to date [12, 28, 57] use deterministic cost functions $f(TDI, RC)$ to reward RBAC configurations that agree with TDI. In Chapter 6 we present an approach that optimizes the likelihood $p(RC|UPA)$ for given UPA and, thereby, takes TDI into account via a linear combination of the negative log-likelihood and a deterministic cost function $f(TDI, RC)$. The selection process of a particular kind of TDI is based on an information-theoretic measure of relevance.

Comparison with true roles As already pointed out, the obvious quality measure that corresponds to our definition is the comparison with the hidden RBAC configuration RC^* underlying the given data UPA. If RC^* is not known, as in real-world scenarios, it cannot be compared with the discovered solution RC. However, in experiments with artificially created data this is possible. A comparison requires a similarity measure $d(RC_1, RC_2)$. Counting the number of correctly recovered roles as done in [47, 75, 76] has the disadvantage that tiny deviations to an original role are as severe as a full disagreement. We therefore recommend using more relaxed measures such as the Jaccard-coefficient (used in [58]) or the Hamming distance as we propose in [71]. Both measures have been discussed in Section 2.5.2. The comparison based on the Hamming distance will be defined in detail in Section 4.4.1.

Generalization error Since in the real world, the original RBAC configuration RC^* is unknown, assessment is much more challenging there. However, even in this case one can quantitatively assess how close the solution is to RC^*. The best measure for carrying out this assessment is the generalization error. The generalization error is often used to assess supervised learning methods for prediction [39]. We propose this measure to serve as the evaluation method for our definition of the role mining problem and investigate it further in the experimental sections of this thesis. It is not straightforward how the generalization error for an unsupervised learning problem like role mining should be computed. We devote Section 7.3 to this problem and propose a method that can be used for a wide variety of unsupervised learning problems.

To compute the generalization error, one must randomly split the available dataset UPA along the users. The larger fraction UPA_1 is given to the role mining algorithm that is to be assessed and a smaller fraction UPA_2 is kept secret from the algorithm. After mining an RBAC configuration RC_1 based on UPA_1, one tests how good this configuration generalizes to the remaining users in UPA_2. For such a test, one must transfer the configuration RC_1 to the users in UPA_2. In Sections 4.4.1 and 7.3, we explain how to assign roles from RC_1 to users in UPA_2. The generalization error is the deviation of the permissions, predicted by the

roles, from the actual permissions in UPA_2.

2.6.4 Role mining as a prediction problem

We advocate the generalization error as the appropriate quality measure for solutions of the problem as defined in Definition 2.6-1. In the following, we explain why the most predictive role configuration, i.e. the one with lowest generalization error, is RC^*.

Problem Definition 2.6-1 assumes that UPA was induced by RC^*. If we split UPA in two parts, UPA_1 and UPA_2 (as done when computing the generalization measure), this assumption still holds for both parts: RC^* underlies both UPA_1 and UPA_2. RC^* should thus be the best predictor for assignments in both UPA_1 and UPA_2. One can try to discover RC^* based on one part and then use the discovered roles to predict the permission assignments of the other. The closer the RBAC configuration is to RC^*, the lower the generalization error will be. As a consequence, the solution that generalizes the best is also the best solution for the role mining problem defined here.

Moreover, we emphasize the importance of the generalization ability of the RBAC configuration: The goal is not primarily to compress the existing user-permission assignment, but rather to infer a set of roles that generalizes well to new users. An RBAC system's maintainability and security improves when the roles must not be redefined whenever there is a small change in the enterprise, such as a new user entering the system or users changing positions within the enterprise. Moreover, as previously explained, it is desirable that the role mining step identifies exceptional permission assignments.

Given the above, the generalization error can be used to recast the proposed problem definition in terms of this quality measure. Namely:

> Given $USERS$, $PRMS$, UPA, and, optionally, TDI, find the RBAC configuration RC^* that minimizes the empirical generalization error.

In Section 7.3 we will propose a technique to compute the empirical generalization error for role mining and other unsupervised learning prob-

lems. In the remainder of the thesis we derive and analyze probabilistic models that aim at minimizing the generalization error.

2.7 Summary

In this chapter we have introduced the role mining problem and related problems. We have analyzed several problem definitions and associated quality measures. It turned out that existing definitions fail to account for some of role mining's practical requirements, especially the requirement for **generalization ability**. Therefore, we reformulated the role mining problem as an **inference problem** (Definition 2.6-1). The appropriate measure for assessing how well a method solves this role mining problem is the **generalization error**.

Chapter 3

A class of probabilistic models for role mining

Existing role mining methods aim for approximating the user-permission assignments at hand as *best possible* by finding a minimal set of roles, user-role assignments, and role-permission assignments, using mainly combinatorial methods. As discussed in Section 2.4, such approaches have three major drawbacks. First, the existing assignments may contain errors. If the approach does not allow one to predict how likely it is to observe a particular assignment, these errors cannot be identified as such and therefore are migrated to the RBAC system. Second, role engineering is not just a data compression problem. The roles should be as meaningful as possible with respect to the users assigned to them. They should ideally represent the particular job functions that groups of users have in a domain. Combinatorial methods that aim to minimize differences with the original assignments often result in synthetic roles that are difficult to understand. Finally, roles that do not correspond to the underlying structure of the access-control configurations generalize poorly to new users in that their permissions do not match with the permissions of previously unseen users.

These problems result from the lack of an underlying statistical model

for role mining. In this chapter, we propose a class of probabilistic models that subsumes different bottom-up role engineering scenarios for RBAC. We show how particular instances of our model class can be defined according to the given domain requirements. The advantage of a probabilistic approach is the ability to generalize from observations about existing assignments. This allows us to identify wrong or missing assignments and avoids the migration of these errors. Moreover, generalization facilitates the addition of new users with minimal information about them.

3.1 From a deterministic rule to a probabilistic model

As follows, we derive the core part of the probabilistic model. The derivation starts with the deterministic rule of assigning users to permissions based on a given role configuration. It converts this rule into a probabilistic version, such that one can reason about the probability of observing a particular user-permission assignment matrix given probabilities of users having particular roles and roles entailing permissions. In the subsequent section we will extend the model to role hierarchies. In later chapters we will further refine the data likelihood by adding particular noise models or prior distributions.

The model that we derive here is not limited to role mining. It can be used to factorize any binary matrix that is assumed to be generated from Boolean disjunctions of a set of bit-vectors. Such disjunctions arise whenever one takes into account that several reasons can lead to the same result. Examples are several diseases leading to the same symptoms or several movie preferences leading to the same movie being watched. Therefore, we will use more general terminology in this rather technical chapter. We will speak of objects instead of users, dimensions of the data space instead of permissions, and binary sources or simply bit-vectors instead of roles.

Let the observed data consist of N **objects**, each associated with D binary **dimensions**. More formally, we denote the data matrix by $\mathbf{X}$,

with $\mathbf{X} \in \{0,1\}^{N \times D}$. We denote the i^{th} row of the matrix by $\mathbf{x}_{i*}$, and the d^{th} column by $\mathbf{x}_{*d}$. We use this notation for all matrices throughout the thesis.

We define the generative process of a bit $x_{id} \in \{0,1\}$ by

$$u_{kd} \sim p(u_{kd}|\beta_{kd}) \tag{3.1}$$

$$x_{id} \sim p(x_{id}|\mathbf{u}_{*d}, \mathbf{z}_{i*}) \ . \tag{3.2}$$

Thereby, $a \sim p(a)$ means that a is a random variable drawn from the probability $p(a)$. The latent variable $u_{kd} \in \{0,1\}$ determines dimension $j \in \{1,..,D\}$ of source $k \in \{1,..,K\}$. The parameter $z_{ik} \in \{0,1\}$ encodes whether object i is assigned to source k. In role mining the vector $\mathbf{u}_{k*}$ encodes the permissions of the role with index k and the vector $\mathbf{z}_{i*}$ encodes the set of roles a user is assigned to. We define that $p(u_{kd}|\beta_{kd})$ is a Bernoulli distribution with $p(u_{kd} = 0) := \beta_{kd}$.

$$p(u_{kd}|\beta_{kd}) := \beta_{kd}^{1-u_{kd}}(1-\beta_{kd})^{u_{kd}} \tag{3.3}$$

Throughout this section, we will condition all probabilities on $\mathbf{Z}$. Therefore, we can ignore $p(z_{ik})$ for the moment. We treat it as a model parameter here. In Chapter 5 we will treat z_{ik} as a random variable and describe a particular prior distribution for it. The generative model as we described it so far is illustrated in Figure 3.1.

Note that the probability $p(x_{id}|\mathbf{u}_{*d}, \mathbf{z}_{i*})$ is deterministic in the following sense. Given all source variables $\mathbf{u}_{kd}$ and the assignments to sources $\mathbf{z}_{i*}$, the bit x_{id} is determined by the disjunction rule defined by the Boolean matrix product

$$\mathbf{X} = \mathbf{Z} \otimes \mathbf{U} \ . \tag{3.4}$$

For deriving the likelihood of x_{id}, we must find a way to express such a deterministic formula $x_{id} = f(\mathbf{u}_{*d}, \mathbf{z}_{i*})$ in terms of a probability distribution. In order to be deterministic, the entire probability mass must be centered at the deterministic outcome, i.e. the distribution must be of the form

$$p(x_{id}|\mathbf{u}_{*d}, \mathbf{z}_{i*}) = \begin{cases} 1, & \text{if } x_{id} = f(\mathbf{u}_{*d}, \mathbf{z}_{i*}) \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

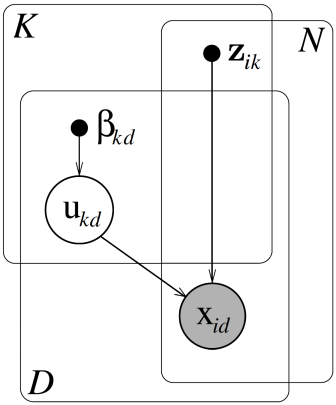


Figure 3.1: Graphical model corresponding to the generation rule of user permission assignments given an RBAC configuration. Semantics: Filled circles account for observable random variables, empty circles are hidden random variables, and points are model parameters. Arrows indicate that one entity influences another one. Entities on a N-plate exist in N different realizations. We will extend this model in the subsequent sections.

The probability distribution

$$p(x_{id}|\mathbf{u}_{*d}, \mathbf{z}_{i*}) = \left(\prod_k (1 - u_{kd})^{z_{ik}}\right)^{(1-x_{id})} \left(1 - \prod_k (1 - u_{kd})^{z_{ik}}\right)^{x_{id}} \tag{3.6}$$

fulfills this requirement as one can see by going through all eight combinations of the binary values of z_{ik}, x_{id}, and u_{kd} (for a single k). It reflects the fact that there are only two possible outcomes of this random experiment. As follows, we exploit this fact and work solely with probabilities for $x_{id} = 0$. The probability for $x_{id} = 1$ can always easily be obtained by assigning the remaining probability mass to this event.

We convert the deterministic formula into a probabilistic version by marginalizing out the latent variables $\mathbf{u}_{kd}$ from the joint distribution for

$\mathbf{u}_{kd}$ and x_{id}. The joint distribution is

$$p(x_{id}=0,\mathbf{u}_{*d}|\boldsymbol{\beta}_{*d},\mathbf{z}_{i*})=p(x_{id}=0|\mathbf{u}_{*d},\mathbf{z}_{i*})\prod_k p(u_{kd}|\beta_{kd})\ . \tag{3.7}$$

As a single bit x_{id} depends on the outcome of a full bitvector $\mathbf{u}_{*d}$ of length K, one must marginalize over all realizations of such bit-vectors.

Let Ω be the set of all possible binary vectors $\mathbf{u}_{*d}$. Then the likelihood for $x_{id}=0$ is

$$\begin{aligned}
&p(x_{id}=0|\boldsymbol{\beta}_{*d},\mathbf{z}_{i*}) && (3.8)\\
&\quad=\sum_{\mathbf{u}_{*d}\in\Omega} p(x_{id}=0,\mathbf{u}_{*d}|\boldsymbol{\beta}_{*d},\mathbf{z}_{i*}) && (3.9)\\
&\quad=\sum_{\mathbf{u}_{*d}\in\Omega} p(x_{id}=0|\mathbf{u}_{*d},\mathbf{z}_{i*})\prod_k p(u_{kd}|\beta_{kd}) && (3.10)\\
&\quad=\sum_{\mathbf{u}_{*d}\in\Omega}\prod_k (1-u_{kd})^{z_{ik}}\left(\beta_{kd}^{1-u_{kd}}(1-\beta_{kd})^{u_{kd}}\right) && (3.11)\\
&\quad=\sum_{\mathbf{u}_{*d}\in\Omega}\left(\prod_{k:z_{ik}=1}(1-u_{kd})^{z_{ik}}\,\beta_{kd}^{1-u_{kd}}(1-\beta_{kd})^{u_{kd}}\right) && (3.12)\\
&\qquad\cdot\left(\prod_{k:z_{ik}=0}\beta_{kd}^{1-u_{kd}}(1-\beta_{kd})^{u_{kd}}\right)
\end{aligned}$$

In the second last step we substituted the individual probabilities with their definitions Eq. (3.3) and Eq. (3.6). In the last step we separated the bit-vectors $\mathbf{u}_{*d}$ into the two cases where $z_{ik}=1$ and $z_{ik}=0$. The first case cancels all contributions of the sum where $z_{ik}=u_{kd}=1$ and for $z_{ik}=1$ and $u_{kd}=0$ only the factor β_{kd} remains. Therefore, it is convenient to introduce a modified set of bit-vectors $\Omega'=\{\mathbf{u}_{*d}\in\Omega\,|z_{ik}=1\Rightarrow u_{kd}=0\}\subset\Omega$, i.e. the entries of $\mathbf{u}_{*d}$ which are relevant for object i are fixed to 0. The likelihood then takes a compact form

$$p(x_{id}=0|\boldsymbol{\beta}_{*d},\mathbf{z}_{i*})=\sum_{\mathbf{u}'_{*d}\in\Omega'}\left\{\left(\prod_{k:z_{ik}=1}\beta_{kd}\right)\left(\prod_{k:z_{ik}=0}\beta_{kd}^{1-u'_{kd}}(1-\beta_{kd})^{u'_{kd}}\right)\right\} \tag{3.13}$$

As derived in Appendix A.1, one can interchange the sum and the product in Eq. (3.13) and thereby simplify it again.

$$p(x_{id} = 0|\boldsymbol{\beta}_{*d}, \mathbf{z}_{i*}) = \prod_k \Big(\beta_{kd}^{z_{ik}} \underbrace{\sum_{u'_{kd} \in \{0,1\}} \beta_{kd}^{1-u'_{kd}} (1-\beta_{kd})^{u'_{kd}}}_{=1 \ \forall k} \Big) \tag{3.14}$$

$$= \prod_k \beta_{kd}^{z_{ik}} \tag{3.15}$$

This is a convenient form of the likelihood. It reflects the intuition that only those sources influence an object to which the object is assigned to. In the context of role mining this means that the permissions of a user are solely determined by the roles the user is assigned to.

As x_{id} can only take two possible values, we have $p(x_{id} = 1|\boldsymbol{\beta}_{*d}, \mathbf{z}_{i*}) = 1 - \prod_k \beta_{kd}^{z_{ik}}$ such that the likelihood of the bit x_{id} is

$$p(x_{id}|\boldsymbol{\beta}_{*d}, \mathbf{z}_{i*}) = \left(\prod_k \beta_{kd}^{z_{ik}} \right)^{1-x_{id}} \left(1 - \prod_k \beta_{kd}^{z_{ik}} \right)^{x_{id}} . \tag{3.16}$$

According to this likelihood, the different entries in $\mathbf{X}$ are conditionally independent given the the parameters $\mathbf{Z}$ and $\boldsymbol{\beta}$. Therefore, the complete data likelihood factorizes over objects and dimensions.

$$p(\mathbf{X}|\boldsymbol{\beta}, \mathbf{Z}) = \prod_{i=1}^{N} \prod_{d=1}^{D} p(x_{id}|\boldsymbol{\beta}_{*d}, \mathbf{z}_{i*}) \tag{3.17}$$

3.2 Role hierarchies

In this section, we extend the core model derived in the last section. We introduce the concept of role hierarchies. From the hierarchy perspective, the core model provides a hierarchy of height 1. There is only one level of roles. In our derivation we introduce one additional level resulting in a hierarchy of height 2. At each level of this modified model there are roles. Roles of the second layer can be sub-roles of roles of the first layer

meaning that the set of permissions from the super-role is a superset of all sets of permissions from the sub-roles.

Our derivation is generic. This means that it can add extra layers to a hierarchical model. With repeated application, probabilistic models with hierarchies of any height can be derived. As we will see, the one-level hierarchy (flat RBAC) and the two-level hierarchy are of particular interest. In Chapter 4 we propose a particular model variant for flat RBAC. In Chapter 5 we investigate a specific instance of a two-level hierarchy.

Again, the hierarchical model derived here is not restricted to role mining. However, we will motivate it by practical considerations of access-control. We assume that there exists a decomposition of the set of users into groups that are not necessarily disjoint: Users are assigned to one or more groups by a Boolean assignment matrix $\mathbf{Z}$. Each row i represents a user and the columns k represent user-groups. In practice, such a decomposition may be performed by the Human Resources Department of an enterprise, for example, by assigning users to divisions in the enterprise according to defined similarities of the employees. If such data is lacking, then the decomposition may just be given by the differences in the assigned permissions for each user. For simplicity, the matrix $\mathbf{Z}$ has the same notation as in the last section. Also there, $\mathbf{Z}$ introduced a (possibly overlapping) grouping of the users.

Now we introduce a new layer. We assume that there is a decomposition of the permissions such that every permission belongs to one or more permission-group. These memberships are expressed by the Boolean assignment matrix $\mathbf{Y}$. Here the lth row of $\mathbf{Y}$ represents the permission-group l and the dth column is the permission d. The assignment of permissions to permission-groups can be motivated, for instance, by technical similarities of the resources that the permissions grant access to. For example, in an object-oriented setting, permissions might be grouped that execute methods in the same class. Alternatively, permissions could be categorized based on the risk that is associated with granting someone a particular permission. Of course, permissions can also be grouped according to the users who own them.

We denote user-groups by **business roles** whereas permission-groups

are referred to as **technical roles**. Business roles are assigned to technical roles. We represent these assignments in a matrix $\mathbf{U}$. To keep track of all introduced variables we list the types of the above-mentioned Boolean assignment matrices:

- Users i to permissions d : $x_{id} \in \{0,1\}$, where $i \in \{1,..,N\}, d \in \{1,..,D\}$.
- Users i to business roles k: $z_{ik} \in \{0,1\}$, where $k \in \{1,..,K\}$.
- Business roles k to technical roles l: $v_{kl} \in \{0,1\}$.
- Technical roles l to permissions d: $y_{ld} \in \{0,1\}$, where $l \in \{1,..,L\}$.

Throughout this section, the indices i, d, k, and l have the above scope and are used to index the above items.

Starting with this extra layer of roles, one can obtain back a flat hierarchy by collapsing the role-role assignment matrix and the role-permission assignments via the disjunction rule

$$u_{kd} = \bigvee_l v_{kl} \wedge y_{ld}. \tag{3.18}$$

Thereby, $\mathbf{U}$ can be understood as the role-permission assignment matrix from the last section. Taking this into account, the final $N \times D$ user-permission assignment matrix $\mathbf{X}$ is determined by two Boolean matrix products

$$\mathbf{X} = \mathbf{Z} \otimes \mathbf{U} \tag{3.19}$$

$$= \mathbf{Z} \otimes \mathbf{V} \otimes \mathbf{Y} \tag{3.20}$$

$$\text{with} \quad x_{id} = \bigvee_k \left[z_{ik} \wedge \left(\bigvee_l v_{kl} \wedge y_{ld} \right) \right]. \tag{3.21}$$

Equation 3.20 expresses whether a user i is assigned to a permission d. There is one Boolean matrix product per role layer. Again, we are interested in the probability of such an assignment. Starting from the logical expression, we derive below how likely it is to observe an assignment of a user i to a permission d.

The deterministic assignment rule for two layers of roles is graphically illustrated in Figure 3.2(a): a user is assigned to a permission if

there is at least one path in the graph connecting them. Since a permission may belong to multiple technical roles and a user may belong to multiple business roles, which in turn may be assigned to multiple technical roles, a user can be assigned to a permission in more than one way (cf. Figure 3.2(a): there may be multiple connecting paths). Therefore, it is easier to express how a user may *not* be assigned to a permission (we denote this by $\neg x := \overline{x}$) rather than computing the union over all possible assignment paths. As in the last section, the remaining event then gets all remaining probability mass.

$$
\begin{aligned}
p\left(\overline{x_{id}}\right) &= p\left(\overline{\bigvee_k \left[z_{ik} \wedge \left(\bigvee_l v_{kl} \wedge y_{ld} \right) \right]}\right), \quad \begin{matrix} 1 \le k \le K \\ 1 \le l \le L \end{matrix} && (3.22)\\
&= \prod_k p\Bigg(\overline{z_{ik} \wedge \underbrace{\left(\bigvee_l v_{kl} \wedge y_{ld} \right)}_{=:u_{kd}}}\Bigg) && (3.23)\\
&= \prod_k \Big(\underbrace{p\left(\overline{z_{ik}} \wedge u_{kd}\right) + p\left(\overline{z_{ik}} \wedge \overline{u_{kd}}\right)}_{=p(\overline{z_{ik}})} + p\left(z_{ik} \wedge \overline{u_{kd}}\right) \Big) && (3.24)\\
&= \prod_k \left(p(\overline{z_{ik}}) + p(\overline{u_{kd}}) p(z_{ik}) \right) && (3.25)
\end{aligned}
$$

Note that in the step from the second to the third line, the correct probability is only obtained when summing over the probabilities of exclusive events (in particular: $\overline{a \wedge b} = \overline{a} \vee \overline{b}$ but $p\left(\overline{a \wedge b}\right) \neq p\left(\overline{a}\right) + p\left(\overline{b}\right)$). Given the generation of u_{kd} in Eq. (3.18), we have that

$$
\begin{aligned}
p(\overline{u_{kd}}) &= \prod_l p\left(\overline{v_{kl} \wedge y_{ld}}\right) \\
&= \prod_l \left[p(\overline{y_{ld}}) + p(\overline{v_{kl}}) p(y_{ld}) \right] . && (3.26)
\end{aligned}
$$

Substituting this back into Eq. (3.25) yields

$$
p(\overline{x_{id}}) = \prod_k \left[p(\overline{z_{ik}}) + p(z_{ik}) \prod_l \left(p(\overline{v_{kl}}) p(y_{ld}) + p(\overline{y_{ld}}) \right) \right] . \qquad (3.27)
$$

We condition this expression on the binary entries of $\mathbf{Y}$ and $\mathbf{Z}$.

$$\begin{aligned} p(\overline{x_{id}} \mid \mathbf{z}_{i*}, \mathbf{y}_{*d}) &= \prod_k 1^{1-z_{ik}} \cdot \left(\prod_l p(\overline{v_{kl}})^{y_{ld}} \cdot 1^{1-y_{ld}} \right)^{z_{ik}} \\ &= \prod_{k,l} p(\overline{v_{kl}})^{z_{ik} y_{ld}} \end{aligned} \tag{3.28}$$

As this expression is independent of other matrix entries in $\mathbf{X}$, we can express the complete likelihood of the user-permission assignment matrix given the business roles and technical roles as a product over users and permissions.

$$\begin{aligned} p(\mathbf{X} \mid \mathbf{Z}, \mathbf{Y}) &= \prod_{i,d} \left[1 - p(\overline{x_{id}} \mid z_{i*}, y_{*d})\right]^{x_{id}} \left[p(\overline{x_{id}} \mid \mathbf{z}_{i*}, \mathbf{y}_{*d})\right]^{1-x_{id}} \\ &= \prod_{i,d} \left[1 - \prod_{k,l} p(\overline{v_{kl}})^{z_{ik} y_{ld}} \right]^{x_{id}} \left[\prod_{k,l} p(\overline{v_{kl}})^{z_{ik} y_{ld}} \right]^{1-x_{id}} \end{aligned} \tag{3.29}$$

If we treat v_{kl} as a Bernoulli variable then this likelihood resembles the one with only one layer of roles. The only difference are the additional binary variables y_{ld} in the exponent that, like z_{ik}, can switch off individual terms of the product. It was up to our choice to condition on $\mathbf{Y}$ and $\mathbf{Z}$ and leave v_{kl} random. As well, we could condition on $\mathbf{V}$ and $\mathbf{Z}$ and infer y_{ld}. This demonstrates how to infer parameters in role hierarchies of arbitrary size. One repeatedly treats the assignment variables in one layer as a random variable and conditions on the current state of the others. We will demonstrate such an alternating inference scheme on a two level hierarchy in Chapter 5.

3.3 Overparametrization and instantiation by introducing constraints

The above model of user-permission assignments in an RBAC environment defines a very general framework. In principle, one can iteratively

introduce additional role layers without changing the outer form of the likelihood. In the derivations, we have avoided any assumptions about the probabilities of the entries of $\mathbf{V}$, $\mathbf{Y}$, and $\mathbf{Z}$. We have only exploited the fact that these variables are Booleans and, therefore, they only take the values 0 or 1. Also, we have avoided any assumptions about the processes that lead to a particular decomposition of the set of users and the set of permissions. Moreover, we have not specified any possible constraints on the user decomposition, the permission decomposition, or the assignments from user-groups to permission-groups.

It turns out that the model with a two-level hierarchy already has more degrees of freedom than required to represent the access control information present in many domains that arise in practice. In particular, when only the data $\mathbf{X}$ is given, there is no indication how to decompose the second role level. This becomes obvious when we think about a one-level decomposition with role-permission assignments $\mathbf{U}$ (as in Eq. (3.4)) and try to convert it into a two-level decomposition. The $\mathbf{Z}, \mathbf{U}$ decomposition has already sufficiently many degrees of freedom to fit any binary matrix. Further decomposing $\mathbf{U}$ into an extra layer of roles $\mathbf{V}$ and assignments from these roles to permissions $\mathbf{Y}$ is arbitrary when there is no additional information or constraint. Therefore, the flat RBAC configuration with only one role layer is the most relevant one. The two-level hierarchy without constraints is over-parameterized. It can be seen as a template for an entire class of models. By introducing constraints, we can instantiate this template to specialized models that fit the requirements of particular RBAC environments and have a similar model complexity as flat RBAC without constraints. These instances of the model class are given by augmenting unconstrained two-level RBAC with assumptions on the probability distributions of the binary variables and giving constraints on the variables themselves. In the following, we will present three relevant model instances and relate them to each other.

3.3.1 Flat RBAC

In this model, each permission is restricted to be a member of only one permission-group and each permission-group can contain only a single

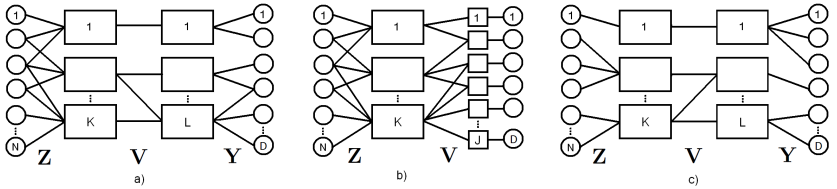


Figure 3.2: Illustration of the structure of three model instances. A user has a permission if there is at least one path connecting them. a) Full model. b) Model with trivial decomposition of the permissions (flat RBAC). c) Disjoint Decomposition Model with only one business role per user and one technical role per permission.

permission. Formally:

$$(\forall l : \textstyle\sum_d y_{ld} = 1) \wedge (\forall d : \textstyle\sum_l y_{ld} = 1) \ . \tag{3.30}$$

The conditioned likelihood then becomes

$$p\left(\mathbf{X} \mid \mathbf{Z}\right) = \prod_{i,d} \left[1 - \prod_k p(\overline{v_{kd}})^{z_{ik}}\right]^{x_{id}} \left[\prod_k p(\overline{v_{kd}})^{z_{ik}}\right]^{1-x_{id}} . \tag{3.31}$$

This "collapsed" model has the same model complexity as flat RBAC without constraints. One can see this by renaming v by u and J by D. As each technical role serves as a proxy for exactly one permission we have $D = J$ anyway. A graphical representation of the structure of this instance is given in Figure 3.2, b). Equivalently, the same constraints could instead be applied to the users, leading to a model with the same structure. As this model already suffices to reconstruct any binary matrix, it is of particular interest for role mining. In Chapter 4 we will provide an in-depth analysis of this model. Thereby, we will add different noise models and compare it with other methods for Boolean matrix decomposition.

3.3.2 Upper-bounded assignments to the user-groups

It is useful in some domains to place an upper bound k_{max} on the number of business roles that a user belongs to: $\forall i : \sum_k z_{ki} \leq k_{max}$. Suppose, for example, that company employees can be classified based on the different business areas that they work in or the different kinds of contracts they have. The business roles can be used to formalize these categories on the set of employees. Moreover, this structure would naturally limit the number of business roles, e.g., to the number of different contract categories. Such a setting would also support the addition of new users to the system by Human Resources. Note that even with a limited number of business roles, a user may have multiple technical roles. Similarly, one could constraint the number of technical roles of a permission and leave the user-role assignment unconstrained.

3.3.3 Disjoint decomposition model

The next model has even stronger constraints. Namely, $k_{max} = 1$ and the number of assigned permission-groups per permission is limited to $l_{max} = 1$. This formalizes that each user belongs to exactly one user-group and each permission belongs exactly to one permission-group. Hence, both users and permissions are partitioned into disjoint business roles and technical roles, respectively. A disjoint decomposition drastically reduces the complexity of a two-level hierarchy while still retaining a high degree of flexibility, since users of a given user-group may still be assigned to multiple permission-groups. We illustrate this model in Figure 3.2, c).

Since both the user and permission groups are disjoint, there is a simple, illustrative representation for data generated or inferred by this model. Namely, the original assignment matrix $\mathbf{x}$ can be drawn with an order on the rows and columns that is given by the assignments to the groups. All rows (users) of the same group are ordered adjacent to each other and all columns (permissions) of a permission group are also ordered adjacent to each other. See Figure 5.2(b) for an example of this representation.

In Chapter 5, we will further introduce prior assumptions on the parameters of the disjoint decomposition model and describe an existing sampling algorithm for inferring its model parameters.

3.4 Summary

In this chapter derived a probabilistic model for RBAC starting from the logical structure of an RBAC configuration. The model can be easily extended to different role hierarchies and different constraints. Therefore, this core model generates an entire class of models for RBAC. We introduced several model instances of particular interest, such as flat RBAC without constraints on the number of roles of a user and a two-level role hierarchy where each user only has one role. In the next two chapters of the thesis we will further extend these two models and analyze their performance for role mining.

Chapter 4

Flat RBAC: Multi-assignment clustering

In this chapter, we investigate the probabilistic model for flat RBAC without constraints. We refine it by adding different noise models that account for the irregularities in the data. This model variant is of particular interest from a technical point of view. If we consider role mining as a clustering task, with the understanding that all users with the same role belong to the same cluster, then an unconstrained RBAC configuration requires that a user can belong to multiple clusters at the same time. Conventional clustering techniques are limited to partitioning the objects of the data set. In our case, the clusters overlap in terms of their objects (thereby, the clusters also overlap with their centroids because the permissions can be part of multiple roles). Note that this differs from 'fuzzy' clustering, where objects are partially assigned to clusters such that these fractional assignments add up to 1. In our approach, an object can be assigned to multiple clusters *at the same time*, i.e. the assignments of an object can sum to a number larger than 1. Membership

in a second cluster does not decrease the intensity of the membership in the first cluster. We call this approach *multi-assignment clustering* (MAC). In principle, it can also be used for other Boolean data that is generated by disjunctions of Boolean vectors. Such data could be, for instance, binary movie preferences, data for market basket analysis, or social networks. We will therefore provide a very general description of our model. However, we will carry out the experimental analysis on access-control data.

The remainder of this chapter is organized as follows. In Section 4.1, we refine the probabilistic model for flat RBAC as defined in Eq. (3.16). We examine various application-specific noise processes and theoretically investigate the relationships among these variants. In Section 4.2 we point to existing clustering techniques that also support assignments to multiple clusters. We describe an annealing method for parameter inference in Section 4.3. In Section 4.4, we present experiments on synthetic and real-world data generated from multiple sources. We demonstrate the ability of multi-assignment clustering to compute more precise parameter estimates than state-of-the art clustering approaches.

4.1 Generative model for Boolean data from multiple sources

In this section, we explain our model of the generation process of binary data, where data may be generated by multiple clusters. The observed data stems from an underlying structure that is perturbed by noise. We first recapitulate the model for the structure as derived in Chapter 3. Afterwards, we model different types of noise processes. Finally, we provide a unifying view of these processes.

4.1.1 Structure model

The structure part of our generative model is the data likelihood defined in Eq. (3.16):

$$p(x_{id}^S|\boldsymbol{\beta}_{*d}\mathbf{z}_{i*}) = \Big(\prod_k \beta_{kd}^{z_{ik}}\Big)^{1-x_{id}^S}\Big(1-\prod_k \beta_{kd}^{z_{ik}}\Big)^{x_{id}^S} \ .$$

and

$$p(\mathbf{X}^S|\boldsymbol{\beta},\mathbf{Z}) = \prod_{i=1}^{N}\prod_{d=1}^{D} p(x_{id}^S|\boldsymbol{\beta}_{*d},\mathbf{z}_{i*})$$

Please note that we added the superscript S to the data x_{id}^S in order to make explicit that this is the model part that accounts for the structure of the data. From a clustering perspective, the assignment of data items to clusters is encoded in the binary assignment matrix $\mathbf{Z} \in \{0,1\}^{N\times K}$, with $z_{ik} = 1$ if and only if a data item i belongs to cluster k, and $z_{ik} = 0$ otherwise. The sum of the assignment variables for a data item i, $\sum_k z_{ik}$, can be larger than 1, which denotes that a data item i is assigned to multiple clusters. This multiplicity explains the name **multi-assignment clustering** (MAC). Each cluster is represented by a Boolean centroid vector. Combinations of these vectors generate the the structure $\mathbf{X}^S \in \{0,1\}^{N\times D}$ of the data. Therefore, we denote the cluster centroids as **sources**. The sources are encoded as rows of $\mathbf{U} \in \{0,1\}^{N\times K}$.

We introduce new notation for the assignment of objects to sources. Let the set of the sources of an object be $\mathcal{L}_i := \{k \in \{1,..,K\} \,|\, z_{ik} = 1\}$. Let $\mathbb{L}$ be the set of all possible **assignment sets** and $\mathcal{L} \in \mathbb{L}$ be one such assignment set. As in Chapter 3, β_{kd} is the probability that source k emits a 0 at dimension d: $\beta_{kd} := p(\overline{u_{kd}})$. Employing the notion of assignment sets, we can write

$$\beta_{\mathcal{L}_i d} := \prod_{k=1}^{K} \beta_{kd}^{z_{ik}} \ . \tag{4.1}$$

The probabilities $\beta_{\mathcal{L}_i *}$ can be interpreted as the probabilistic source of the assignment set $\mathcal{L}_i$. It is a substitute for the combination of all sources

in $\mathcal{L}_i$. However, note that this interpretation differs from a standard disjoint clustering where $L := |\mathbb{L}|$ independent sources are assumed and must be inferred. Here, we only have $K \times D$ parameters β_{kd} whereas in disjoint clustering, the number of source parameters would be $L \times D$, which can be up to $2^K \times D$. The expression $\beta_{\mathcal{L}_i d}$ is rather a 'proxy'-source and we introduce it just for notational convenience. The probability distribution of a x_{id} generated from this structure model given the assignments $\mathcal{L}_i$ and the sources $\boldsymbol{\beta}$ is then

$$p_S\left(x_{id}^S \mid \boldsymbol{\beta}_{*d}, \mathcal{L}_i\right) = \left(1 - \beta_{\mathcal{L}_i d}\right)^{x_{id}^S} \left(\beta_{\mathcal{L}_i d}\right)^{1 - x_{id}^S} . \tag{4.2}$$

Again the value of x_{id}^S is a Boolean disjunction of the values at dimension d of all sources to which object i is assigned. The Boolean disjunction in the generation process of an x_{id}^S results in a probability for $x_{id}^S = 1$, which is strictly non-decreasing in the number of associated sources $|\mathcal{L}_i|$: If any of the sources in $\mathcal{L}_i$ emits a 1 at dimension d then $x_{id}^S = 1$. Conversely, $x_{id}^S = 0$ requires that all contributing sources have emitted a 0 in dimension d. The empty assignment set $\mathcal{L}_i := \{\}$is part of the hypothesis class, i.e. a data item i can be assigned to none of the sources. The corresponding row x_{i*}^S contains only zeros then.

In the following sections, we describe various noise models that alter the output of the structure model. The structure part of the model together with a particular noise process is illustrated in Figure 4.1.

Single-Assignment Clustering. Before we introduce the noise processes we highlight a particular variant of the model described above. In the general case, which is when no restrictions on the assignment sets are given, there are $L = 2^K$ possible assignment sets. If the number of clusters to which an object can be simultaneously assigned is bounded by $|\mathcal{L}_i| = \sum_k z_{ik} \leq M$, this number reduces to $L = \sum_{m=0}^{M} \binom{K}{m}$.

The particular case with $M = 1$ provides a model variant that we call *Single-Assignment Clustering* (SAC). SAC provides a disjoint clustering of the objects. In order to endow SAC with the same model complexity as MAC, we provide it with L clusters. Each of the assignment sets is then identified with one of the clusters. The clusters are

learned independently of each other by computing the cluster parameters $\beta_{\mathcal{L}*}$ for each $\mathcal{L}$, discarding the dependencies in the original formulation. The underlying generative model of SAC, as well as the optimality conditions for its parameters, can be obtained by treating all assignment sets $\mathcal{L}$ as independent and unique sources in the subsequent equations. With all centroids computed according to Eq. 4.1, the single-assignment clustering model then yields the same probability for the data as the multi-assignment clustering model.

4.1.2 Noise models and their relationship

In this section, we present two noise models that appear at first sight to be different, but are actually different representations of the same model. The first noise model, the *mixture model*, interprets the observed data as a mixture of emissions from the structure part and a noise source. Each bit in the matrix is either generated by the structure model or by an independent global noise process. The second model, the *flip model*, flips some randomly chosen bits of the structure matrix $\mathbf{X}^S$ either from 0 to 1 or from 1 to 0. We will describe the two noise models in detail and then unify them. Furthermore, we will show that the flip noise model generalizes the noisy-or likelihood, which allows flips from 0 to 1.

The different noise models have different parameters. We denote the noise parameters of a model α by Θ_N^α. The full set of parameters characterizing the structure and noise are then $\Theta := (\boldsymbol{\beta}, \Theta_N^\alpha)$. As additional notation, we use the indicator function $\mathbf{I}_{\{p\}}$ for a predicate p, defined as

$$\mathbf{I}_{\{p\}} := \begin{cases} 1 & \text{if } p \text{ is true} \\ 0 & \text{otherwise .} \end{cases}$$

Mixture Noise Model

In the mixture noise model, each x_{id} is generated either by the structure distribution or by a noise process. The binary indicator variable ξ_{id} indicates whether x_{id} is a noisy bit ($\xi_{id} = 1$) or a structure bit ($\xi_{id} = 0$). The observed x_{id} is then generated by

$$x_{id} = (1 - \xi_{id}) x_{id}^S + \xi_{id} x_{id}^N \,, \tag{4.3}$$

where the generative process for the structure bit x_{id}^S is either described by the deterministic rule in Eq. 3.4 or by the probability distribution in Eq. 4.2. The random bit x_{id}^N follows a Bernoulli distribution that is independent of the object index i and the dimension d:

$$p_N\left(x_{id}^N \,|\, r\right) = r^{x_{id}^N} \left(1-r\right)^{1-x_{id}^N} \;, \tag{4.4}$$

where r is the parameter of the Bernoulli distribution indicating the probability of a value 1. Combining the structure and noise distributions, the overall probability of an observed x_{id} is

$$p_M^{\text{mix}}(x_{id} \,|\, \mathcal{L}_i, \boldsymbol{\beta}, r, \xi_{id}) = p_N(x_{id} \,|\, r)^{\xi_{id}} \, p_S(x_{id} \,|\, \mathcal{L}_i, \boldsymbol{\beta})^{1-\xi_{id}} \;. \tag{4.5}$$

We assume ξ_{id} to follow a Bernoulli distribution with parameter $\epsilon := p(\xi_{id} = 1)$. We called ϵ the **noise fraction**. The joint probability of x_{id} and ξ_{id} given the assignment matrix $\mathbf{Z}$ and all parameters is thus given by

$$p_M^{\text{mix}}\left(x_{id}, \xi \,|\, \mathbf{Z}, \boldsymbol{\beta}, r, \epsilon\right) = p_M\left(x_{id} \,|\, \mathbf{Z}, \boldsymbol{\beta}, r, \xi\right) \cdot \epsilon^{\xi_{id}} \left(1-\epsilon\right)^{1-\xi_{id}} . \tag{4.6}$$

Since different x_{id} are conditionally independent given the assignments $\mathbf{Z}$ and the parameters θ^{mix}, we have

$$p_M^{\text{mix}}\left(\mathbf{X}, \xi \,|\, \mathbf{Z}, \boldsymbol{\beta}, r\right) = \prod_{id} p_M^{\text{mix}}\left(x_{id}, \xi \,|\, \mathbf{Z}, \boldsymbol{\beta}, r\right). \tag{4.7}$$

The noise indicators ξ_{id} cannot be observed. We therefore marginalize out all ξ_{id} to derive the probability of $\mathbf{X}$ as

$$p_M^{\text{mix}}\left(\mathbf{X} \,|\, \mathbf{Z}, \boldsymbol{\beta}, r, \epsilon\right) = \sum_{\{\xi\}} p_M^{\text{mix}}\left(\mathbf{X}, \xi \,|\, \mathbf{Z}, \boldsymbol{\beta}, r, \epsilon\right) \tag{4.8}$$

$$= \prod_{id} \left(\epsilon \cdot p_N(x_{id}) + (1-\epsilon) \cdot p_S(x_{id})\right) \;. \tag{4.9}$$

The observed data $\mathbf{X}$ is thus a mixture between the emissions of the structure part (which has weight $1-\epsilon$) and the noise emissions (with weight ϵ). We illustrate the model in Figure 4.1.

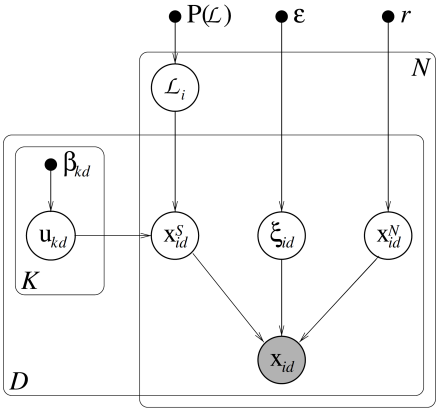


Figure 4.1: The generative model of Boolean MAC with mixture noise. $\mathcal{L}_i$ is the assignment set of object i, indicating which sources from $\mathbf{U}$ generated it. The bit ξ_{id} selects whether the noise-free bit x^S_{id} or the noise bit x^N_{id} is observed. This model is an extension of the model in Fig. 3.1.

Introducing the auxiliary variable

$$q^{\text{mix}}_{\mathcal{L}_i d} := p^{\text{mix}}_M \left(x_{id} = 1 \mid \mathbf{Z}, \boldsymbol{\beta}, r, \epsilon\right) = \epsilon r + (1-\epsilon)\left(1 - \beta_{\mathcal{L}_i d}\right) \tag{4.10}$$

to represent the probability for $x_{id} = 1$ under this model, we get a data-centric representation of the probability of $\mathbf{x}$ as

$$p^{\text{mix}}_M \left(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\beta}, r, \epsilon\right) = \prod_{id} \left(x_{id}\, q^{\text{mix}}_{\mathcal{L}_i d} + (1 - x_{id})\left(1 - q^{\text{mix}}_{\mathcal{L}_i d}\right)\right) . \tag{4.11}$$

The parameters of the mixture noise model are $\Theta^{\text{mix}}_N := (\epsilon, r)$. Since ϵ and r are independent of d and i, we will refer to ϵ and r as parameters of a 'global' noise process.

Flip Noise Model

In contrast to the previous noise model, where the likelihood is a mixture of *independent* noise and structure distributions, we assume the effect of

the noise depends on the structure itself. The data is generated from the same structure distribution as in the mixture noise model. Individual bits are then randomly selected and flipped. In the following, we formalize the generation of data under this model for two different cases. First, we consider the case where the probability ϵ_1 for a 1 in $\mathbf{X}^*$ to be flipped can differ from the probability ϵ_0 for a 0 to be flipped. This is the general flip noise model, which we will denote as the **general bit-flip model**. Afterwards, we will specialize this model to the case where $\epsilon_0 = \epsilon_1$, the **symmetric bit-flip model**.

The generative process for a bit x_{id} is described by

$$x_{id} = x^S_{id} \oplus \xi_{id} \ , \tag{4.12}$$

where $\oplus$ denotes addition modulo 2. Again, the generative process for the structure bit x^S_{id} is described by either Eq. 3.4 (deterministic rule) or Eq. 4.2 (probability). The values of ξ_{id} indicate whether the bits x^S_{id} are to be flipped ($\xi_{id} = 1$) or not (ξ_{id}=0). In a probabilistic formulation, we assume that the indicator ξ_{id} for a bit-flip is distributed according to

$$\xi_{id} \quad \sim \quad p(\xi_{id}|x^S_{id}, \epsilon_0, \epsilon_1) \ . \tag{4.13}$$

Thus, the probability of a bit-flip, given the structure and the noise parameters (ϵ_0, ϵ_1), is

$$p(\xi_{id}|x^S_{id}, \epsilon_0, \epsilon_1) = \left(\epsilon_1^{x^S_{id}} \epsilon_0^{1-x^S_{id}}\right)^{\xi_{id}} \left((1-\epsilon_1)^{x^S_{id}} (1-\epsilon_0)^{1-x^S_{id}} \right)^{1-\xi_{id}} , \tag{4.14}$$

with the convention that $0^0 = 1$. Given the flip indicator ξ_{id} and the structure bit x^S_{id}, the final observation is deterministic:

$$\begin{aligned} p^{\text{flip}}_M(x_{id}|\xi_{id}, x^S_{id}) &= x_{id} \left((1-x^S_{id})\xi_{id} + x^S_{id}(1-\xi_{id}) \right) \\ &\quad + (1-x_{id}) \left(\left(1-x^S_{id}\right)(1-\xi_{id}) + x^S_{id}\xi_{id} \right) \\ &= x_{id}^{\mathbf{I}_{\{\xi_{id} \neq x^S_{id}\}}} (1-x_{id})^{\mathbf{I}_{\{\xi_{id} = x^S_{id}\}}} \ . \end{aligned} \tag{4.15}$$

The joint probability distribution is then given by

$$\begin{aligned} p_M^{\text{flip}}(x_{id}, \xi_{id}, x_{id}^S | \beta, z_{i*}, \epsilon_0, \epsilon_1) \\ &= p(x_{id}|\xi_{id}, x_{id}^S)\; p(\xi_{id}|x_{id}^S, \epsilon_0, \epsilon_1)\; p_S(x_{id}^S|\beta, z_{i*}) \\ &= \epsilon_1^{\xi_{id} x_{id}^S} (1-\epsilon_1)^{(1-\xi_{id}) x_{id}^S} (1-\beta_{\mathcal{L}_i d})^{x_{id}^S} \\ &\quad \cdot \epsilon_0^{\xi_{id}(1-x_{id}^S)} (1-\epsilon_0)^{(1-\xi_{id})(1-x_{id}^S)} \beta_{\mathcal{L}_i d}^{1-x_{id}^S} \; . \end{aligned} \tag{4.16}$$

Marginalizing out ξ_{id} and x_{id}^S, the probability of the data under the flip noise model is

$$\begin{aligned} p_M^{\text{flip}}(x_{id}|\beta, z_{i*}, \epsilon_0, \epsilon_1) &= \sum_{\xi_{id}, x_{id}^S} p(x_{id}, \xi_{id}, x_{id}^S | \beta, z_{i*}, \epsilon_0, \epsilon_1) \\ &= \epsilon_1 \left(1 - \beta_{\mathcal{L}_i d}\right)(1 - x_{id}) + \epsilon_0 \beta_{\mathcal{L}_i d} x_{id} \\ &\quad + (1-\epsilon_1)\left(1 - \beta_{\mathcal{L}_i d}\right) x_{id} + (1-\epsilon_0)\beta_{\mathcal{L}_i d}(1 - x_{id}) \; . \end{aligned} \tag{4.17}$$

In this representation, it is obvious that the observed data is a mixture between the noise emissions (described by the first two terms, for $x_{id} = 0$ and $x_{id} = 1$, respectively) and the emissions of the structure (the last two terms). Under the flip noise model, the probability of $x_{id} = 1$ is given by

$$q_{\mathcal{L}_i d}^{\text{flip}} := 1 - \epsilon_1 - (1 - \epsilon_0 - \epsilon_1)\beta_{\mathcal{L}_i d} \; , \tag{4.18}$$

which yields the data probability

$$p_M^{\text{flip}}(x_{id}|\boldsymbol{\beta}, z_{i*}, \epsilon_0, \epsilon_1) = x_{id} q_{\mathcal{L}_i d}^{\text{flip}} + (1 - x_{id}) \left(1 - q_{\mathcal{L}_i d}^{\text{flip}}\right) \; . \tag{4.19}$$

The flip noise model is parameterized by $\Theta_N^{flip} := (\epsilon_0, \epsilon_1)$, which is, again, global.

Symmetric bit-flip model. If the probability for a bit to be flipped is independent of the bit's value, formally if $\epsilon_1 = \epsilon_0 =: \epsilon$, then the probability for $x_{id} = 1$ under this model becomes

$$q_{\mathcal{L}_i d}^{\text{sym}} = 1 - \epsilon - (1 - 2\epsilon)\, \beta_{\mathcal{L}_i d} \; . \tag{4.20}$$

The only parameter of the symmetric flip noise model is thus: $\Theta_N^{s-flip} := (\epsilon)$.

Noisy-Or. The noisy-OR is, in a way, the opposite case of the symmetric bit-flip model: The noise process can generate extra 1s, but a 1 is never flipped to a 0. Formally, the noisy-or model is characterized by $\epsilon_1 = 0$, and only $\epsilon_0 \in [0, 1]$ remains to be estimated.

Unified Noise Model

In this section, we unify the two noise models presented above (and their special cases symmetric bit-flip and noisy-OR). The overall generation process has two steps:

1. The structure part of the data is generated according to the sources, as described in Section 4.1.1. It is defined by the probability $p_S\left(x_{id}^S \mid \mathcal{L}_i, \boldsymbol{\beta}\right)$ (Eq. 4.2).
2. A noise process acts on the structure $\mathbf{X}^S$ and thus generates the observed data matrix $\mathbf{X}$. This noise process is described by the probability $p^\alpha(x_{id}|x_{id}^S, \theta_N^\alpha)$, where α identifies the noise model and θ_N^α are the parameters of the noise model α.

The overall probability of an observation x_{id} given all parameters is thus

$$p_M^\alpha\left(x_{id}|\mathcal{L}_i, \boldsymbol{\beta}, \theta_N^\alpha\right) = \sum_{x_{id}^S} p_S\left(x_{id}^S \mid \mathcal{L}_i, \boldsymbol{\beta}\right) \cdot p^\alpha\left(x_{id}|x_{id}^S, \theta_N^\alpha\right) \ . \tag{4.21}$$

In the following, we reformulate the two previously described noise models in this framework.

Mixture Noise Model. The mixture noise model assumes that each x_{id} is explained either by the structure model or by an independent global noise process. Therefore, the joint probability of $p^{\text{mix}}\left(x_{id}|x_{id}^S, \theta_N^{\text{mix}}\right)$ can be factored as

$$p^{\text{mix}}\left(x_{id}|x_{id}^S, \theta_N^{\text{mix}}\right) = p_M^{\text{mix}}\left(x_{id}|x_{id}^S, x_{id}^N, \xi_{id}\right) \cdot p_N^{\text{mix}}(x_{id}^N|r) \ , \tag{4.22}$$

with

$$p_M^{\text{mix}}\left(x_{id}|x_{id}^S, x_{id}^N, \xi_{id}\right) = \left(\mathbf{I}_{\{x_{id}^S = x_{id}\}}\right)^{1-\xi_{id}} \left(\mathbf{I}_{\{x_{id}^N = x_{id}\}}\right)^{\xi_{id}} \ . \tag{4.23}$$

	Flip Noise Model $\alpha = \text{flip}$, $\Theta_N^{\text{flip}} = (\epsilon_0, \epsilon_1)$	Mixture Noise Model $\alpha = \text{mix}$, $\Theta_N^{\text{mix}} = (\epsilon, r)$
$p^\alpha(x_{id} = 0 \mid x_{id}^S = 0, \Theta_N^\alpha)$	$1 - \epsilon_0$	$1 - \epsilon \cdot r$
$p^\alpha(x_{id} = 1 \mid x_{id}^S = 0, \Theta_N^\alpha)$	ϵ_0	$\epsilon \cdot r$
$p^\alpha(x_{id} = 0 \mid x_{id}^S = 1, \Theta_N^\alpha)$	ϵ_1	$\epsilon \cdot (1 - r)$
$p^\alpha(x_{id} = 1 \mid x_{id}^S = 1, \Theta_N^\alpha)$	$1 - \epsilon_1$	$1 - \epsilon \cdot (1 - r)$

Table 4.1: Comparison of the influence of the different noise models on the structured data.

$p^S(x_{id}^S|\mathcal{L}_i, \boldsymbol{\beta})$ and $p_N^{\text{mix}}(x_{id}^N|r)$ are defined by Eq. 4.2 and Eq. 4.4 respectively. Summing out the unobserved variables x_{id}^S and x_{id}^N yields

$$
\begin{aligned}
&p_M^{\text{mix}}\left(x_{id}|\mathcal{L}_i, \boldsymbol{\beta}, r, \xi_{id}\right) \\
&\quad = \sum_{x_{id}^S=0}^{1} \sum_{x_{id}^N=0}^{1} p_M^{\text{mix}}\left(x_{id}, x_{id}^S, x_{id}^N|\mathcal{L}_i, \boldsymbol{\beta}, r, \xi_{id}\right) && (4.24) \\
&\quad = p_S\left(x_{id}|\mathcal{L}_i, \boldsymbol{\beta}\right)^{1-\xi_{id}} \cdot p_N^{\text{mix}}\left(x_{id}|r\right)^{\xi_{id}} && (4.25) \\
&\quad = (1 - \xi_{id}) p_S\left(x_{id}|\mathcal{L}_i, \boldsymbol{\beta}\right) + \xi_{id} p_N^{\text{mix}}\left(x_{id}|r\right) \ . && (4.26)
\end{aligned}
$$

Integrating out the noise indicator variables ξ_{id} leads to the same representation as in Eq. (4.5).

Flip Noise Model. Under the assumptions of the flip noise model, a noisy bit depends on the structure bits. Using Eq. 4.16, we derive

$$
p^{\text{flip}}\left(x_{id}|x_{id}^S, \theta_N^{\text{flip}}\right) = \sum_{\xi_{id}=0}^{1} p_M^{\text{flip}}(x_{id}|\xi_{id}, x_{id}^S) \cdot p(\xi_{id}|x_{id}^S, \epsilon_0, \epsilon_1) \ . \quad (4.27)
$$

The probability distributions $p(\xi_{id}|x_{id}^S, \epsilon_0, \epsilon_1)$ and $p_M^{\text{flip}}(x_{id}|\xi_{id}, x_{id}^S)$ are given by Eq. 4.14 and 4.15, respectively. Inserting these expressions and integrating out x_{id}^S as well as ξ_{id}, we again get Eq. 4.17.

Relation between the noise parameters. Our unified formulation of the noise models enables us to compare the influence of the noise

processes on the clean structure under different noise models. These results are summarized in Table 4.1. The probabilities $p^{\text{mix}}(x_{id}|x_{id}^S, \Theta_N^\alpha)$ and $p^{\text{flip}}(x_{id}|x_{id}^S, \Theta_N^\alpha)$, for the cases $(x_{id} = 1, x_{id}^S = 0)$ and $(x_{id} = 0, x_{id}^S = 1)$ respectively, provide the parameters of the flip noise model that is equivalent to a given mixture noise model.

mixture noise model		**flip noise model**
$\Theta_N^{\text{mix}} = (\epsilon, r)$	is equivalent to	$\Theta_N^{\text{flip}} = (\epsilon \cdot r, \epsilon \cdot (1 - r))$

Conversely, we have that

flip noise model		**mixture noise model**
$\Theta_N^{\text{flip}} = (\epsilon_0, \epsilon_1)$	is equivalent to	$\Theta_N^{\text{mix}} = \left(\epsilon_0 + \epsilon_1, \frac{\epsilon_0}{\epsilon_0 + \epsilon_1}\right)$

This shows that the two noise-processes are different representations of the same process. We will therefore use just the mixture noise model throughout the remainder of this paper.

Object-wise and dimension-wise noise processes. In the following, we extend the noise model presented above. Given the equivalence of mix and flip noise, we restrict ourselves to the mixture noise model. Assume a separate noise process for every dimension d, which is parameterized by r_d and has intensity ϵ_d. We then have

$$p\left(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\beta}, \boldsymbol{\theta}\right) = \prod_{i,d} \left(\epsilon_d r_d^{x_{id}} \left(1 - r_d\right)^{1 - x_{id}} + \left(1 - \epsilon_d\right) \left(1 - \beta_{\mathcal{L}_i d}\right)^{x_{id}} \beta_{\mathcal{L}_i d}^{1 - x_{id}} \right) .$$

Now assume a separate noise process for every object i, which is parameterized by ϵ_i and r_i. As before, we have

$$p\left(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\beta}, \boldsymbol{\theta}\right) = \prod_{i,d} \left(\epsilon_i r_i^{x_{id}} \left(1 - r_i\right)^{1 - x_{id}} + \left(1 - \epsilon_i\right) \left(1 - \beta_{\mathcal{L}_i d}\right)^{x_{id}} \beta_{\mathcal{L}_i d}^{1 - x_{id}} \right) .$$

Note that these local noise models are very specialized and could be used in the following application scenarios. In role mining, some permissions are more critical than others. Hence it appears reasonable to assume a lower error probability for the dimension representing, for

example, root access to a central database server than for the dimensions representing the permission to change the desktop background image. However, we experimentally observed that the additional degrees of freedom in these models often leads to an over-parametrization and thus worse overall results. This problem could possibly be reduced by introducing further constraints on the parameters.

4.2 Other probabilistic models for binary data

The likelihood that is most similar to the one we propose is the noisy-OR gate introduced in [63]. Our model allows random flips in *both* directions. The noisy-OR model, which is constrained to random bit flips from zeros to ones, is thus a special case of the noise model that we presented in Section 4.1.2. There are two models using a noisy-OR likelihood. Noisy-OR component analysis (NOCA) [77], is based on a variational inference algorithm [43]. This algorithm computes only the global probabilities $p(\mathbf{u}_{*d} = 1)$, but does not return a complete decomposition. A non-parametric model based on the Indian-Buffet process [35] and a noisy-OR likelihood is presented in [80]. We call this approach the infinite noisy-OR (INO). Our method differs from INO with respect to the data likelihood and with respect to optimization. While our model yields an exact solution to an approximate model, replacing the binary assignments by probabilistic assignments, the inference procedure for INO aims at solving the exact model by sampling. INO is a latent feature model, as described by [33], with Boolean features. Latent feature models explain data by combinations of multiple features that are indicated as active (or inactive) in a binary matrix $\mathbf{z}$. Being member in multiple clusters (encoded in $\mathbf{z}$) is technically equivalent to having multiple features activated.

The binary independent component analysis (BICA) of [44] is a factor model for binary data. The combination of the binary factors is modeled with linear weights and thus deviates from the goal of finding binary decompositions, mentioned above. However, the method can be adapted

to solve binary decomposition problems and performs well under certain conditions as we will demonstrate in Section 4.4.2.

Another model-based approach for clustering binary data is also related to our model, although more distantly. It is a Bayesian non-parametric mixture model including multiple assignments of objects to binary or real-valued centroids and was proposed in [41]. When an object belongs to multiple clusters, the product over the probability distributions of all individual mixtures is considered, which corresponds to a conjunction of the mixtures. This constitutes a probabilistic model of the Boolean AND, whereas in all the above methods mentioned, as well as in our model, the data generation process uses the OR operation to combine mixture components.

In the experiments at the end of this chapter, we compare our approach with the two probabilistic models INO and BICA. Moreover, we investigate the combinatorial algorithm Discrete Basis Problem Solver (DBPS) [54]. We explain DBPS in detail in Appendix C.

4.3 Inference

We now describe an inference algorithm for our model. We learn the parameters according to the maximum likelihood principle and use the optimization method of **deterministic annealing** presented in [10] and [66]. In the following, we specify the deterministic annealing scheme used in the algorithm. In Section 4.3.2 we then give the characteristic magnitudes and the update conditions in a general form, i.e. independent of the noise model. The particular update equations for the individual models are then derived in Sections 4.3.3 and 4.3.4.

4.3.1 Deterministic annealing for clustering

The likelihood of a data matrix $\mathbf{X}$ (Eq. 4.11 and Eq. 4.19 for the mixture and the flip noise model, respectively) is highly non-convex in the model parameters and a direct maximization of this function will likely be trapped in local optima. Deterministic annealing is an optimization method that parameterizes a smooth transition from the convex problem

of maximizing the entropy (i.e. a uniform distribution over all possible clustering solutions) to the problem of minimizing the **empirical risk** R. In our case, the risk is the negative log likelihood. Such methods are also known as continuation methods (see [4]). Formally, the Lagrange functional

$$F := -T \log Z = \mathbb{E}_G [R] - TH \tag{4.28}$$

is introduced, with Z being the **partition function** over all possible clustering solutions (see Eq. 4.32), and G denotes the **Gibbs distribution** (see Eq. 4.30 and Eq. 4.31). F is called the **free energy**. The expectation of the empirical risk in $\mathbb{E}_G [R]$ is computed with respect to the Gibbs distribution. The Lagrange parameter T (called the **computational temperature**) controls the trade-off between entropy maximization and minimization of the expected empirical risk. Minimizing F at a given temperature T is equivalent to constraint minimization of the empirical risk R with a lower limit on the entropy H. In other words, H is a uniform prior on the likelihood of the clustering solutions. Its weight decreases as the computational temperature T is incrementally reduced.

At every temperature T, a gradient-based expectation-maximization (EM) step computes the parameters that minimize F. The E-step computes the risks $R_{i\mathcal{L}}$ (Eq. 4.29) of assigning data item i to assignment set $\mathcal{L}$. The corresponding responsibilities $\gamma_{i\mathcal{L}}$ (Eq. 4.30) are computed for all i and $\mathcal{L}$ based on the current values of the parameters. The M-step first computes the optimal values of the noise parameters. Then it uses these values to compute the optimal source parameters $\boldsymbol{\beta}$. The individual steps are described in Sections 4.3.3 and 4.3.4.

We determine the initial temperature as described in [66] and use a constant cooling rate ($T \leftarrow \vartheta \cdot T$, with $0 < \vartheta < 1$) . We continue decreasing T until the responsibilities $\gamma_{i\mathcal{L}}$ for each data item i are sharply peaked at single assignment sets $\mathcal{L}_i$.

4.3.2 Specific parameter updates

As before, we use α as a placeholder for a particular noise model. Depending on the noise model α, the probability $q^{\alpha}_{\mathcal{L}_i d}$ for $x_{id} = 1$ (Eq. 4.10

and 4.18) differs. However, the outer form of the characteristic magnitudes of the problem is independent of the particular noise model.

Following our generative approach to clustering, we aim at finding the maximum likelihood solution for the parameters. Taking the logarithm of the likelihood simplifies the calculations as products become sums. Also, the likelihood function conveniently factorizes over the objects and features enabling us to investigate the risk of objects individually. We define the empirical risk of assigning an object i to the set of clusters $\mathcal{L}$ as the negative log-likelihood of the feature vector x_{i*} being generated by the sources contained in $\mathcal{L}$:

$$\begin{aligned} R^{\alpha}_{i\mathcal{L}} &:= -\log p(x_{i\cdot}|\mathcal{L}_i, \Theta^{\alpha}) \\ &= -\sum_d \log\left(x_{id}\left(1-q^{\alpha}_{\mathcal{L}d}\right) + (1-x_{id})q^{\alpha}_{\mathcal{L}d}\right) \ . \end{aligned} \tag{4.29}$$

The **responsibility** $\gamma^{\alpha}_{i\mathcal{L}}$ of the assignment-set $\mathcal{L}$ for data item i is given by

$$\gamma^{\alpha}_{i\mathcal{L}} := \frac{\exp\left(-R^{\alpha}_{i\mathcal{L}}/T\right)}{\sum_{\mathcal{L}'\in\mathbb{L}} \exp\left(-R^{\alpha}_{i\mathcal{L}'}/T\right)} \ . \tag{4.30}$$

In this way, the matrix $\boldsymbol{\gamma}^{\alpha}$ defines a probability distribution over the space of all clustering solutions. The expected empirical risk $\mathbb{E}_G[R]$ of the solutions under this probability distribution G is

$$\mathbb{E}_G\left[R^{\alpha}_{i\mathcal{L}}\right] = \sum_i \sum_{\mathcal{L}} \gamma^{\alpha}_{i\mathcal{L}} R^{\alpha}_{i\mathcal{L}} \ . \tag{4.31}$$

Finally, the partition function Z^{α} and the free energy F^{α} for model α are defined as follows.

$$Z^{\alpha} := \prod_i \sum_{\mathcal{L}} \exp\left(-R^{\alpha}_{i\mathcal{L}}/T\right) \tag{4.32}$$

$$\begin{aligned} F^{\alpha} &:= -T\log Z^{\alpha} \\ &= -T\sum_i \log\left(\sum_{\mathcal{L}} \exp\left(-R^{\alpha}_{i\mathcal{L}}/T\right)\right) \end{aligned} \tag{4.33}$$

Given the above, we derive the updates of the model parameters based on the first-order condition of the free energy F^α. We therefore introduce the generic model parameter θ, which stands for any of the model parameters, i.e. $\theta \in \{\beta_{\mu\nu}, \epsilon_0, \epsilon_1, \epsilon, r\}$. Here, μ is some particular value of source index k and ν is some particular value of dimension index d. Using this notation, the derivative of the free energy with respect to θ is given by

$$\frac{\partial F^\alpha}{\partial \theta} = \sum_i \sum_{\mathcal{L}} \gamma^\alpha_{i\mathcal{L}} \frac{\partial R^\alpha_{i\mathcal{L}}}{\partial \theta} \tag{4.34}$$

$$= \sum_i \sum_{\mathcal{L}} \gamma^\alpha_{i\mathcal{L}} \sum_d \frac{(1-2x_{id}) \frac{\partial q^\alpha_{\mathcal{L}d}}{\partial \theta}}{x_{id}\,(1-q^\alpha_{\mathcal{L}d}) + (1-x_{id}) q^\alpha_{\mathcal{L}d}} \ . \tag{4.35}$$

4.3.3 Update conditions for the mixture noise model

Derivatives for the mixture noise model ($\theta \in \{\beta_{\mu\nu}, \epsilon, r\}$) are:

$$\frac{\partial}{\partial \beta_{\mu\nu}} q^{\text{mix}}_{\mathcal{L}d} = (1-\epsilon)\, \beta_{\mathcal{L}\setminus\{\mu\},d}\, \mathbf{I}_{\{\nu=d\}} \mathbf{I}_{\{\mu\in\mathcal{L}\}} \ , \tag{4.36}$$

$$\frac{\partial}{\partial \epsilon} q^{\text{mix}}_{\mathcal{L}d} = 1 - r - \beta_{\mathcal{L}d} \ , \tag{4.37}$$

$$\frac{\partial}{\partial r} q^{\text{mix}}_{\mathcal{L}d} = -\epsilon \ . \tag{4.38}$$

This results in the following first-order conditions for the mixture noise model.
Condition for the source parameter updates:

$$\sum_{\mathcal{L}:\mu\in\mathcal{L}} \beta_{\mathcal{L}\setminus\{\mu\},\nu} \left\{ \frac{\sum_{i:x_{i\nu}=1} \gamma^{\text{mix}}_{i\mathcal{L}}}{\epsilon r + (1-\epsilon)\,(1-\beta_{\mathcal{L}\nu})} - \frac{\sum_{i:x_{i\nu}=0} \gamma^{\text{mix}}_{i\mathcal{L}}}{1-\epsilon r - (1-\epsilon)\,(1-\beta_{\mathcal{L}\nu})} \right\} = 0 \tag{4.39}$$

Condition for the noise parameter updates:

$$\sum_d \left\{ \sum_{\mathcal{L}} \frac{(1-r-\beta_{\mathcal{L}d}) \sum_{i:x_{id}=1} \gamma_{i\mathcal{L}}^{\text{mix}}}{\epsilon r + (1-\epsilon)(1-\beta_{\mathcal{L}d})} - \sum_{\mathcal{L}} \frac{(1-r-\beta_{\mathcal{L}d}) \sum_{i:x_{id}=0} \gamma_{i\mathcal{L}}^{\text{mix}}}{1-\epsilon r - (1-\epsilon)(1-\beta_{\mathcal{L}d})} \right\} = 0 \tag{4.40}$$

$$\sum_d \left\{ \sum_{\mathcal{L}} \frac{\sum_{i:x_{id}=0} \gamma_{i\mathcal{L}}^{\text{mix}}}{1-\epsilon r - (1-\epsilon)(1-\beta_{\mathcal{L}d})} - \sum_{\mathcal{L}} \frac{\sum_{i:x_{id}=1} \gamma_{i\mathcal{L}}^{\text{mix}}}{\epsilon r + (1-\epsilon)(1-\beta_{\mathcal{L}d})} \right\} = 0 \tag{4.41}$$

Here, $\sum_{\mathcal{L}:\mu\in\mathcal{L}}$ is the sum over all assignment sets containing cluster μ and $\beta_{\mathcal{L}\setminus\{\mu\},\nu} = \prod_{k\in\mathcal{L},k\neq\mu} \beta_{k\nu}$. There is no analytic expression for the solutions of the above equations and one must determine the parameters $\beta_{\mu\nu}$, ϵ, and r numerically. We use the Newton method to determine the optimal values for the parameters. We observed that this method rapidly converges, usually within 5 iterations.

The above equations contain the optimality conditions for the single-assignment clustering (SAC) model as a special case. As only assignment sets $\mathcal{L}$ with one element are allowed in this model, we can globally substitute $\mathcal{L}$ by k and get $\beta_{\mathcal{L}*} = \beta_{k*}$. Furthermore, since 1 is the neutral element of the multiplication, we get $\beta_{\mathcal{L}\setminus\{\mu\},\nu} = 1$.

In the noise-free case, the value for the noise fraction is $\epsilon = 0$. This results in a significant simplification of the update equations.

4.3.4 Update conditions for the flip noise model

Derivatives for the asymmetric bit flip model ($\theta \in \{\beta_{\mu\nu}, \epsilon_0, \epsilon_1\}$) are:

$$\frac{\partial}{\partial \beta_{\mu\nu}} q_{\mathcal{L}d}^{\text{asym}} = (1-\epsilon_0-\epsilon_1)\, \beta_{\mathcal{L}\setminus\{\mu\},\nu}\, \mathbf{I}_{\{\nu=d\}} \mathbf{I}_{\{\mu\in\mathcal{L}\}} \ , \tag{4.42}$$

$$\frac{\partial}{\partial \epsilon_0} q_{\mathcal{L}d}^{\text{asym}} = -\beta_{\mathcal{L}d} \ , \tag{4.43}$$

$$\frac{\partial}{\partial \epsilon_1} q_{\mathcal{L}d}^{\text{asym}} = 1 - \beta_{\mathcal{L}d} \ . \tag{4.44}$$

Setting the derivatives of the free energy F with respect to the respective parameters to 0 results in the following update conditions for the optimal

parameter values:

$$\sum_{\mathcal{L}:\mu\in\mathcal{L}} \beta_{\mathcal{L}\setminus\{\mu\},\nu} \left\{ \frac{\sum_{i:x_{i\nu}=1} \gamma_{i\mathcal{L}}^{\text{asym}}}{1-\epsilon_1-\varepsilon\beta_{\mathcal{L}\nu}} - \frac{\sum_{i:x_{i\nu}=0} \gamma_{i\mathcal{L}}^{\text{asym}}}{\epsilon_1+\varepsilon\beta_{\mathcal{L}\nu}} \right\} = 0 \tag{4.45}$$

$$\sum_d \left\{ \sum_{\mathcal{L}} \frac{\beta_{\mathcal{L}d} \sum_{i:x_{id}=0} \gamma_{i\mathcal{L}}^{\text{asym}}}{\epsilon_1+\varepsilon\beta_{\mathcal{L}d}} - \sum_{\mathcal{L}} \frac{\beta_{\mathcal{L}d} \sum_{i:x_{id}=1} \gamma_{i\mathcal{L}}^{\text{asym}}}{1-\epsilon_1-\varepsilon\beta_{\mathcal{L}d}} \right\} = 0 \tag{4.46}$$

$$\sum_d \left\{ \sum_{\mathcal{L}} \frac{(1-\beta_{\mathcal{L}d}) \sum_{i:x_{id}=1} \gamma_{i\mathcal{L}}^{\text{asym}}}{1-\varepsilon\beta_{\mathcal{L}d}} - \sum_{\mathcal{L}} \frac{(1-\beta_{\mathcal{L}d}) \sum_{i:x_{id}=0} \gamma_{i\mathcal{L}}^{\text{asym}}}{\epsilon_1+\varepsilon\beta_{\mathcal{L}d}} \right\} = 0 \tag{4.47}$$

We have used the abbreviation $\varepsilon := 1-\epsilon_0-\epsilon_1$. Again, the values of the parameters $\beta_{\mu\nu}$, ϵ_0, and ϵ_1 must be determined numerically, since there is no general analytic solution to the above equations.

Symmetric flip noise model: By inserting $\epsilon_0 = \epsilon_1 = \epsilon$ into the above update equations, we get the optimality conditions for the symmetric flip noise model:

$$\sum_{\mathcal{L}:\mu\in\mathcal{L}} \beta_{\mathcal{L}\setminus\{\mu\},\nu} \left\{ \frac{\sum_{i:x_{i\nu}=1} \gamma_{i\mathcal{L}}^{\text{sym}}}{1-\epsilon-\beta_{\mathcal{L}\nu}+2\epsilon\beta_{\mathcal{L}\nu}} - \frac{\sum_{i:x_{i\nu}=0} \gamma_{i\mathcal{L}}^{\text{sym}}}{\epsilon+\beta_{\mathcal{L}\nu}-2\epsilon\beta_{\mathcal{L}\nu}} \right\} = 0 \tag{4.48}$$

$$\sum_d \left\{ \sum_{\mathcal{L}} \frac{(1-2\beta_{\mathcal{L}d}) \sum_{i:x_{id}=1} \gamma_{i\mathcal{L}}^{\text{sym}}}{1-\epsilon-\beta_{\mathcal{L}d}+2\epsilon\beta_{\mathcal{L}d}} - \sum_{\mathcal{L}} \frac{(1-2\beta_{\mathcal{L}d}) \sum_{i:x_{id}=0} \gamma_{i\mathcal{L}}^{\text{sym}}}{\epsilon+\beta_{\mathcal{L}d}-2\epsilon\beta_{\mathcal{L}d}} \right\} = 0 \tag{4.49}$$

As in the general case, the parameter values $\beta_{\mu\nu}$ and ϵ must be determined numerically.

4.4 Experiments

In this section, we first introduce the measures that we employ to evaluate the quality of clustering solutions. Then we present results on both synthetic and real data.

4.4.1 Evaluation criteria

We derived our model to solve the inference role mining problem as defined in Definition 2.6-1. As argued in Chapter 2, this determines the quality measures for assessing how well this task is solved. For synthetic data, we evaluate the estimated sources by their Hamming distance to the true sources being used to generate the data. We gain further insight of the investigated algorithms by additionally analyzing the cluster stability. For real-world data, where no true roles are available to us, we analyze the generalization ability of the learned RBAC configurations. The generalization ability of a solution indicates how well the solution fits to the unknown underlying probability distribution of the data. It is therefore the appropriate quality criterion for the role mining problem when the true generating roles are not known. In the following, we explain how to compute these measures, parameter mismatch, cluster stability, and generalization ability.

The following notation will prove useful. We denote by $\hat{\mathbf{Z}}$ and $\hat{\mathbf{U}}$ the estimated decomposition of the matrix $\mathbf{X}$. The reconstruction of the matrix based on this decomposition is $\hat{\mathbf{X}}$, where $\hat{\mathbf{X}} := \hat{\mathbf{Z}} \otimes \hat{\mathbf{U}}$. Furthermore, in experiments with synthetic data, the signal part of the matrix is known. As in Section 4.1, we denote it by $\mathbf{X}^S$.

Parameter Mismatch

Experiments with synthetic data enable us to compare the values of the true model parameters with the inferred model parameters. We report below on the accuracies of both the estimated sources $\hat{\mathbf{U}}$ and the noise parameters.

To evaluate the accuracy of the source estimates, we use the average Hamming distance between the true and the estimated sources. In order to account for the arbitrary numbering of clusters, we permute the source vectors $\mathbf{u}_{k*}$ with a permutation $\pi(k)$ such that the estimated and the true sources agree best. Namely,

$$a(\hat{\mathbf{U}}) := \frac{1}{K \cdot D} \min_{\pi \in P_K} \sum_{k=1}^{K} \left|\left| \mathbf{u}_{k*} - \hat{\mathbf{u}}_{\pi(k)*} \right|\right| \ ,$$

where P_K denotes the set of all permutations of K elements. Finding P_K involves solving the assignment problem which can be calculated in polynomial time using the Hungarian algorithm of [48]. Whenever we know the true model parameters, we will assess methods based on parameter mismatch, always reporting this measure in percent.

Generalization Error

For real world data, the true model parameters are unknown and there might even exist a model mismatch with the true underlying distribution that generated the input dataset $\mathbf{X}^{(1)}$. Still, one can measure how well the method infers this distribution by testing if the estimated distribution generalizes to a second dataset $\mathbf{X}^{(2)}$ that has been generated in the same way as $\mathbf{X}^{(1)}$. To estimate this generalization ability, we compute the transfer costs of the inferred RBAC configuration. This method will be explained in detail in Chapter 7.3 and, particularly for Boolean matrix factorization, in Section 7.3.4. Here we briefly sketch this method.

We first randomly split the dataset along the objects into a training set $\mathbf{X}^{(1)}$ and a validation set $\mathbf{X}^{(2)}$. Then we learn the factorization $\hat{\mathbf{Z}}$, $\hat{\mathbf{U}}$ based on the training set and transfer it to the validation set. For transferring, we compute for each object i in $\mathbf{X}^{(2)}$ its nearest neighbor $\psi_{NN}(i)$ in $\mathbf{x}^{(1)}$ according to the Hamming distance. We then create a new matrix $\mathbf{Z}'$ defined by $\mathbf{z}'_{i*} = \hat{\mathbf{z}}_{\psi_{NN}(i)*}$ for all i. As a consequence, each validation object is assigned to the same set of sources as its nearest neighbor in the training set. The generalization error is then

$$G(\hat{\mathbf{Z}}, \hat{\mathbf{U}}, \mathbf{X}^{(2)}, \psi_{NN}) := \frac{1}{N^{(2)} \cdot D} \left\| \mathbf{X}^{(2)} - \mathbf{Z}' \otimes \hat{\mathbf{U}} \right\| \ , \tag{4.50}$$

$$\text{with } \mathbf{Z}' = \left(\hat{\mathbf{z}}_{\psi_{NN}(1)*}, \ldots, \hat{\mathbf{z}}_{\psi_{NN}(N^{(2)})*} \right)^T \ , \tag{4.51}$$

where $N^{(2)}$ is the number of objects in the validation dataset and $\otimes$ is the Boolean matrix product as defined in Eq.(2.1). This measure essentially computes the fraction of wrongly predicted bits in the new dataset.

As some of the matrix entries in $\mathbf{X}^{(2)}$ are interpreted as noise, it might be impossible to reach a generalization error of 0%. However, this

affects all methods and all model variants. Moreover, we are ultimately interested in the total order of models with respect to this measure and not in their absolute scores. Since we assume that the noise associated with the features of different objects is independent, we deduce from a low generalization error that the algorithm can infer sources that explain — up to residual noise — the features of new objects from the same distribution. In contrast, a high generalization error implies that the inferred sources wrongly predict most of the matrix entries. This behavior indicates overfitting.

Stability

In some of the experiments we will additionally report on the stability of the clustering solutions. The stability measure [49] is based on the requirement that a clustering solution is reproducible. To quantify the degree to which this requirement is satisfied, we separately cluster two i.i.d. data sets $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ to obtain the cluster assignment matrices $\hat{\mathbf{Z}}^{(1)}$ and $\hat{\mathbf{Z}}^{(2)}$. The first data set $\mathbf{X}^{(1)}$ and the cluster assignments $\mathbf{Z}^{(1)}$ are then used to train a classifier $\phi^{(1)}$. In our experiments we use a nearest neighbor classifier with the Hamming distance as a metric.

Let $\phi^{(1)}(\mathbf{x}_{i*}^{(2)})$ be the output of the classifier $\phi^{(1)}$, trained on $(\mathbf{X}^{(1)}, \hat{\mathbf{Z}}^{(1)})$ and applied to $\mathbf{X}^{(2)}$. Ideally, this output corresponds to the clustering solution $\hat{\mathbf{Z}}^{(2)}$ for every object $\mathbf{x}_{i*}^{(1)}$. Note that in our case, we must use a classifier that can handle multi-assignment data. Again, due to the random numbering of clustering solutions, one must find the permutation that minimizes the deviation. Taking this into account, we define the stability as

$$s := 1 - \frac{|\mathbb{L}|}{|\mathbb{L}| - 1} \frac{1}{N} \min_{\pi \in P_K} \left\{ \sum_{i=1}^{n} \mathbf{I}_{\left\{ \pi\left(\phi^{(1)}(\mathbf{x}_{i*}^{(2)})\right) \neq \hat{\mathbf{z}}_{i*}^{(2)} \right\}} \right\} . \tag{4.52}$$

The predicate $\mathbf{I}$ holds if the cluster-assignment of an object i according to the classification outcome $\pi\left(\phi^{(1)}(x_{i*}^{(2)})\right)$ differs from the clustering solution $z_{i\cdot}^{(2)}$ in at least one cluster. In turn, the predicate does not hold if the entire assignment set $\pi\left(\phi^{(1)}(x_{i*}^{(2)})\right)$ and $z_{i*}^{(2)}$ is identical for a given

object i. As a random assignment of data items to one of $|\mathbb{L}|$ assignment sets yields a stability of $\frac{1}{|\mathbb{L}|}$, the factor $\frac{|\mathbb{L}|}{|\mathbb{L}|-1}$ allows us to compare pairs of clustering solutions with different numbers of assignment sets.

The stability criterion can only be computed if the same number of clusters is obtained on $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. For INO, this is not necessarily the case. When a different number of clusters is obtained, we do not report the stability.

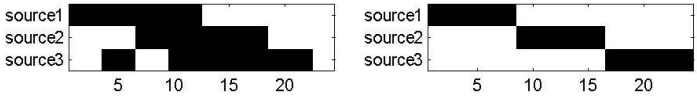
4.4.2 Experiments on synthetic data

This section presents results from several experiments on synthetic data where we investigate the performance of different model variants and other methods. The experiments have the following setting in common. First, we generate data by assigning objects to one or more Boolean vectors out of a set of predefined sources. Unless otherwise stated, we will use the generating sources as depicted in Figure 4.2. Combining the emissions of these sources via the *OR* operation generates the structure of the objects. Note that the sources can overlap, i.e. multiple sources emit a 1 at a particular dimension. In a second step, we perturb the dataset by a noise process.

With synthetic data, we control all parameters, namely the number of objects and sources, the geometry of the Boolean source vectors (i.e. we vary them between overlapping sources and orthogonal sources), the fraction of bits that are affected by the noise process, and the kind of noise process. Knowing the original sources used to generate the dataset enables us to measure the accuracy of the estimators, as described in Section 4.4.1. By varying the problem setting, we investigate how the different methods perform under predefined conditions. We repeat all experiments ten times, each time with different random noise. We report the median (and 65% percentiles) of the accuracy over these ten runs.

Performance of MAC variants

We carry out inference with the MAC model and the corresponding Single-Assignment Clustering (SAC) model, each with and without the



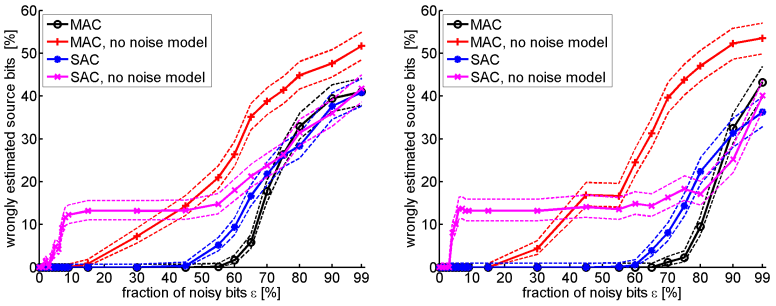
(a) Overlapping Sources (b) Orthogonal Sources

Figure 4.2: Overlapping sources (left) and orthogonal sources (right) used in the experiments with synthetic data. Black indicates a 1 and white a 0 at the corresponding matrix element. In both cases, the three sources have 24 dimensions.

mixture noise model. These model variants are explained in Section 4.1.1. The results illustrated in Fig. 4.3 are obtained using datasets with 350 objects and 3500 objects, respectively. The objects are sampled from the overlapping sources depicted in Fig. 4.2(a). To evaluate the solutions of the SAC variants, we compare the estimated sources against all combinations of the true sources. This is a fair way to compare SAC and MAC because it provides both models the same model complexity.

Influence of noise model. As observed in Figure 4.3(a), the source parameter estimators are much more accurate when a noise model is employed. For a low fraction of noisy bits ($< 50\%$), the estimators with a noise model are perfect, but are already wrong for 10% noise when no noise model is used. When inference is carried out using a model that lacks the ability to explain individual bits by noise, the entire dataset must be explained with the source estimates. Therefore, the solutions tend to overfit the dataset. With a noise model, a distinction between the structure and the irregularities in the data is possible and allows one to obtain more accurate estimates for the model parameters.

Influence of multi-assignments. Multi-Assignment Clustering (MAC) provides more accurate estimates than SAC and the accuracy of MAC breaks down at a higher noise level than the accuracy of SAC. The reason is twofold. First, the ratio of the number of observations per model parameter differs for both model variants. MAC explains the observa-



(a) 350 Objects (b) 3500 Objects

Figure 4.3: Average Hamming distance between true and estimated source prototypes for MAC and SAC with and without noise models respectively.

tions with combinations of sources whereas SAC assigns each object to a single source only. SAC therefore uses only those objects for inference that are exclusively assigned to a source, while MAC, in addition to these objects, also uses objects that are simultaneously assigned to other sources. Second, using the same source in various combinations with other sources implicitly provides a consistency check for the source parameter estimates. SAC lacks this effect as all source parameters are independent. The difference between MAC and SAC becomes apparent when the dataset is noisy. For low fractions of noise, the accuracy is the same for both models.

For noise levels under 40%, the lack of a noise model is more pronounced for SAC: As the number of data items per parameter is smaller than with the MAC model, the effect of a single noisy bit is higher, and a few noisy bits already lead to a wrong estimation. With the multi-assignment clustering model, and thus with more data items per parameter, the effect of the noise is better averaged out than for SAC. For noise levels above a critical value (approximately 45% in the experimental setting considered), this ranking changes: The assumptions underlying the noise-free MAC model are obviously not matched by the

noise, and the effect of this model mismatch becomes more significant than the benefit from a larger sample to average over. The noise-free SAC model, which does not relate different sources with each other, does not incur this model mismatch and therefore yields better results at high noise levels.

Influence of training set size. When the number of samples available for learning increases, the observed effects become more pronounced. This can be seen by comparing the left and right plot in Fig. 4.3. For the methods that contain a noise model (MAC and SAC), there is an interval at high noises where the estimation accuracy breaks down. This interval becomes smaller for MAC and SAC and occurs later (at higher noise levels). The successive accuracy breakdown for MAC occurs in the noise level interval [0.6, 0.8] for 350 objects and in the interval [0.75, 0.9] for 3500 objects. The phase transition for SAC is in both cases in a larger interval at lower noise levels, namely [0.5, 0.8] for 350 objects and [0.65, 0.9] for 3500 objects. Moreover, the difference in estimation accuracy between MAC and SAC around the phase transition becomes more pronounced as the number of samples increases.

For MAC and SAC without a noise model the average accuracy also increases with the number of observations. However, the effects are not as strong as for the noise models.

Comparison with other clustering techniques

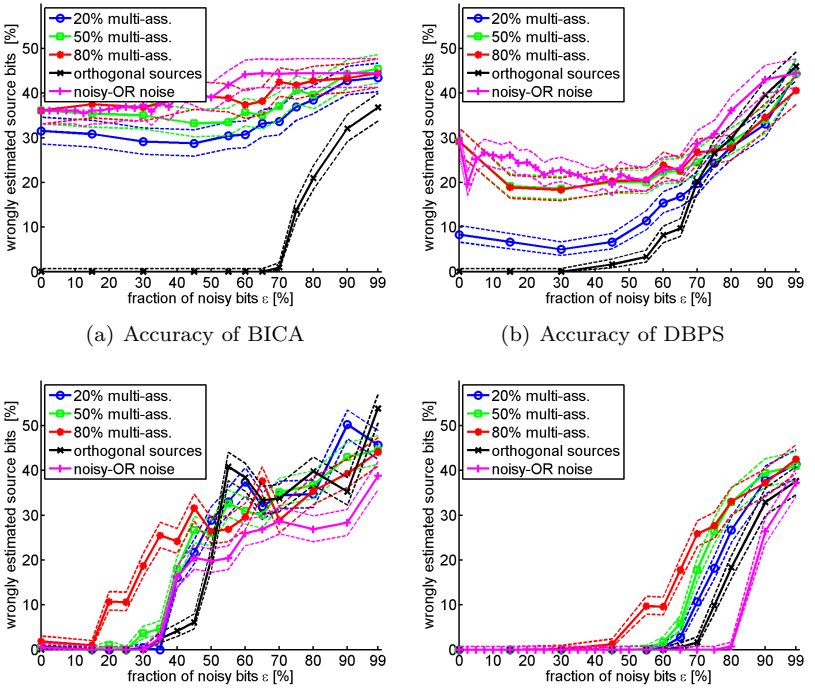
We compare MAC with the probabilistic models Binary Independent Component Analysis (BICA) [44] and Infinite Noisy-Or (INO) [63], as well as with the combinatorial algorithm Discrete Basis Problem Solver (DBPS) [54]. We have already briefly explained the first two models in Section 4.2. A technical description of DBPS is provided in Appendix C. We report the main results of the comparison between the different clustering techniques in Figure 4.4. Each panel illustrates the results of one of the investigated methods under five different experimental setups. We analyze the results of each individual method in separate paragraphs.

Data generation. For generating the data, we use the overlapping sources in Figure 4.2(a) and the orthogonal sources in Figure 4.2(b). We generate 50 data items from each single source as well as from each combination of two sources. Furthermore, 50 additional data items are generated without a source, i.e. they contain no structure. This experimental setting yields 350 data items in total.

In a second step, we randomly perturb the structure by a mixture noise process. The probability of a noisy bit being 1 is kept fixed at $r = 0.5$, while the fraction of noisy bits, ϵ, varies between 0% and 99%. The fraction of data from multiple sources is 50% for the experiments plotted with square markers. Experiments with only 20% (80%) of the data are labeled with circles (with stars). Furthermore, we label experiments with orthogonal sources (Figure 4.2(b)) with 'x'. Finally, we use '+' labels for results on data with a noisy-OR noise process, i.e. $r = 1$.

Binary Independent Component Analysis (BICA). BICA has a poor parameter accuracy in all experiments with data from overlapping clusters. This behavior is caused by the assumption of orthogonal sources, which is not fulfilled for such data. BICA performs better on data that was modified by the symmetric mixture noise process than on data from a noisy-OR noise process. Since BICA does not have a noise model, the data containing noise from the noisy-OR noise process will lead to extra 1s in the source estimators. This effect becomes important when the noise fraction rises above 50%. We observe that, overall, the error rate does not vary much for overlapping sources.

The effect of the source geometry is particularly noticeable. On data generated by orthogonal sources, i.e. when the assumption of BICA is fulfilled, the source parameters are perfectly reconstructed for noise levels up to 65%. Only for higher noise levels, does the accuracy break down. The assumption of non-overlapping source centroids is essential for the performance of BICA as the poor results on data with overlapping sources show. As more data items are generated by multiple, overlapping sources, the influence of the mismatch between the assumption underlying BICA and the true data increases. This effect explains why the source parameter estimators for non-orthogonal centroids become less



(a) Accuracy of BICA

(b) Accuracy of DBPS

(c) Accuracy of INO

(d) Accuracy of MAC

Figure 4.4: Accuracy of source parameter estimation for 5 different types of datasets in terms of mismatch to the true sources. We use (circle, square, star) symmetric Bernoulli noise and overlapping sources with three different fractions of multi-assignment data, (x) orthogonal sources and symmetric noise, and (+) overlapping sources and a noisy-or noise process. Solid lines indicate the median over 10 datasets with random noise and dashed lines show the 65% confidence intervals.

accurate when going from 20% of multi-assignments to 80%.

Discrete Basis Problem Solver (DBPS). Inspecting Figure 4.4(b), we observe that this method yields accurate source parameter estimators

for data generated by orthogonal sources, and, to a lesser degree, for data sets that contain a small percentage of multi-assignment data. As the fraction of multi-assignment data increases, the accuracy of DBPS decreases.

The reason for the low accuracy on multi-assignment data arises from the greedy optimization of DBPS. It selects a new source out of a candidate set such that it can explain as many objects as possible by the newly chosen source. In a setting where most of the data is created by a combination of sources, DBPS will first select a single source that is equal to the disjunction of the true sources because this covers most 1s. We call this effect **combination-singlet confusion**. Lacking a generative model for source-combinations, DBPS cannot use the observation of objects generated by source-combinations to gather evidence for the individual sources. As a consequence, the first selected source estimates fit to the source-combinations and not to the true individual sources. Often, the last selected sources are then left empty, leading to a low estimation accuracy.

Consider, for instance, the following example: a noise-free data matrix generated from two sources and their combination. The corresponding computation steps of DBPS lead to combination-singlet confusion.

$$\begin{pmatrix} \mathbb{1} & \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{1} & \mathbb{1} \end{pmatrix} \overset{\text{unique candidates}}{\longrightarrow} \begin{pmatrix} 1 & \cdots & 1 & \mathbb{0} \\ 1 & \cdots & & 1 \\ \mathbb{0} & 1 & \cdots & 1 \end{pmatrix} \overset{\text{selected sources}}{\longrightarrow} \begin{pmatrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{pmatrix} \tag{4.53}$$

Note the effect of a small amount of noise on the accuracy of DBPS. The clear structure of the association matrix is perturbed, and the candidates might contain 0s in some dimensions. As a result, the roles selected in the second step and subsequent steps are non-empty, making the solution more similar to the true sources. This results in the interesting effect where the accuracy increases when going from noise-free matrices to those with a little bit of noise (for higher noise, it decreases again because of overfitting).

DBPS obtains much more accurate estimators in the setting where the data is generated by orthogonal data (labeled 'x'). Here, the can-

didate set does not contain sources that correspond to combinations of true sources, and the greedy optimization algorithm can only select a candidate source that corresponds to a true single source. Consider, for instance, the following data matrix, generated from two orthogonal sources, and the corresponding computations of DBPS.

$$\begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{1} \\ \mathbb{1} & \mathbb{1} \end{pmatrix} \overset{\text{unique candidates}}{\longrightarrow} \begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{1} \end{pmatrix} \overset{\text{selected sources}}{\longrightarrow} \begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{1} \end{pmatrix} \tag{4.54}$$

Due to this effect, DBPS performs best with respect to source parameter estimation when the generating sources are fully orthogonal. In contrast to BICA, which benefits from the explicit assumption of orthogonal sources, DBPS favors such sources because of the properties of its greedy optimizer.

Infinite noisy-OR (INO). The infinite noisy-OR is a non-parametric Bayesian method. To obtain a single result, we approximate the a posteriori distribution by sampling and then choose the parameters with highest probability. This procedure estimates the maximum a posterior solution. Furthermore, in contrast to BICA, DBPS, and all MAC variants, INO determines the number of sources by itself and might obtain a value different than the number of sources used to generate the data. If the number inferred by INO is smaller than the true number, we choose the closest true sources to compute the parameter mismatch. If INO estimates a larger set of sources than the true one, the best-matching INO sources are used. This procedure systematically overestimates the accuracy of INO, whereas INO actually solves a harder task that includes model-order selection. A deviation between the estimated number of sources and the true number mainly occurs in the mid-noise level (approximately 30% to 70% noisy bits).

In all settings, except the case where 80% of the data items are generated by multiple sources, INO yields perfect source estimators up to noise levels of 30%. For higher noise levels, the accuracy rapidly drops. While the generative model underlying INO enables this method

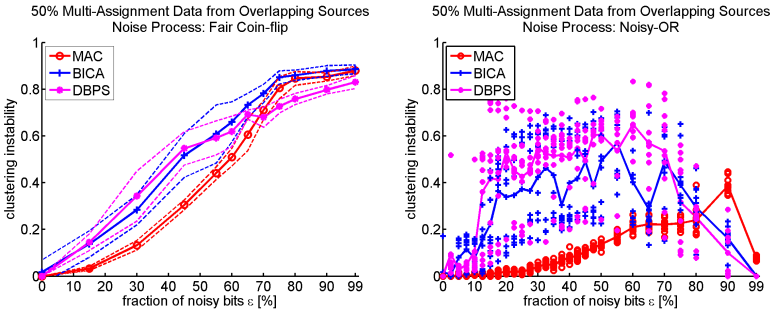
to correctly interpret data items generated by multiple sources, a high percentage (80%) of such data poses the hardest problem for INO.

For noise fractions above approximately 50%, the source parameter estimators are only slightly better than random in all settings. On such data, the main influence comes from the noise, while the contribution of different source combinations is no longer important.

Multi-Assignment Clustering (MAC). The multi-assignment clustering method yields perfect parameter estimators for noise levels up to 40% in all experimental settings considered. The case with 80% of multi-assignment data is the most challenging one for MAC. When only 50% or less of the data items are generated by more than one source, the parameter estimates are accurate for noise levels up to 60% of noisy bits. When few data items originate from a single source, MAC fails to separate the contributions of the individual sources. These single-source data items function as a kind of 'anchor' and help the algorithm to converge to the true parameters of the individual sources. For very high noise levels (90% and above), the performance is again similar for all three ratios of multi-assignment data.

In comparison to the experiments with overlapping sources described in the previous paragraph, MAC profits from orthogonal centroids and yields superior parameter accuracy for noise levels above 50%. As for training data with few multi-assignment data, orthogonal centroids simplify the task of disentangling the contributions of the individual sources. When a reasonable first estimate of the source parameters can be derived from single-assignment data, a '1' in dimension d of a data item is explained either by the unique source which has a high probability of emitting a '1' in this dimension, or by noise — even if the data item is assigned to more than one source.

Interestingly, MAC achieves the best results when the noise is generated by a noisy-OR noise process. The reason is that observing a 1 at a particular bit creates a much higher entropy of the parameter estimate than observing a 0: a 1 can be explained by all possible combinations of sources having a 1 at this position, whereas a 0 gives strong evidence that all sources of the object are 0. As a consequence, a wrong bit being 0 is



(a) Unbiased noise process ($r = 0.5$) (b) Noisy-OR noise process ($r = 1$)

Figure 4.5: Instability of clustering solutions. Data was generated by overlapping sources as depicted in Figure 4.2(a). In (b) we report the individual outcomes of each repetition of the experiment and the trend of the median.

more severe than a wrong 1. The wrong 0 forces the source estimates to a particular value whereas the wrong 1 distributes its 'confusion' evenly over the sources. As the noisy-OR creates only 1s, it is less harmful. This effect could, in principle, also help other methods if they managed to appropriately disentangle combined source parameters.

Stability analysis. We investigate the clustering stability of the different methods on data generated by overlapping sources. We illustrate the results for two different noise processes in Figure 4.5. MAC favors noisy-OR noise over random noise also in terms of stability. Both DBPS and BICA have a higher instability on data with noisy-OR noise than on data with random noise, whereas the results vary much more over the ten datasets with same noise fraction. The high variance is due to the higher influence of the actual positions of the noisy bits, since for $r = 1$ a high fraction of the noise bits x_{id}^N are identical to the original structure bits x_{id}^S. Depending on where the noisy bits are located in the data matrix, the fraction of bits that are actually changed by noise can differ considerably (if there are far more ones than zeros in the data and if

noisy bits are always 1). As a consequence, the instability substantially differs between various runs. This effect is most pronounced for BICA, but is also clearly observable for DBPS.

The instability of all methods drops when the noisy fraction approaches 100%. In this noise regime, most of the bits in the data are 1. Both BICA and DBPS infer one source where all elements are 1 and two sources where all elements are 0. BICA consistently assigns all objects to all three sources. So does MAC, although only at higher noise levels. DBPS assigns all objects to one source. Since all objects are assigned to the same label set, the clustering is, trivially, very stable.

INO repeatedly infers a different number of clusters on the two random subsets of the data. We are therefore not able to compute the instability of the results.

4.4.3 Experiments on role mining data

To evaluate the performance of our algorithm on real data, we apply MAC to mining RBAC roles from access control configurations. We first specify the problem setting and then report on our experimental results.

As explained in Chapter 2, role mining must find a suitable RBAC configuration based on a binary user-permission assignment matrix $\mathbf{X}$. Our MAC model corresponds to flat RBAC (a one-level role hierarchy) without constraints on the assignments. A flat RBAC configuration is a set of K roles and assignments of users to roles. When no constraints are given, a user can have multiple roles, and the bit-vectors representing the roles can overlap. The RBAC configurations inferred by MAC are encoded by the Boolean assignment matrices $(\hat{\mathbf{Z}}, \hat{\mathbf{U}})$.

In our experiments, we use a dataset from a large enterprise containing the user-permission assignment matrix of $N = 4900$ users and $D = 1300$ permissions. A part of this data matrix is depicted in Figure 4.6. The roles are inferred based on the permissions of the first 2400 users. The permissions of the remaining users are used to compute the generalization ability of the role set.

To evaluate the different methods on more complex data with a higher noise level, we generate a second data set $\bar{\mathbf{X}}$ as follows: We combine

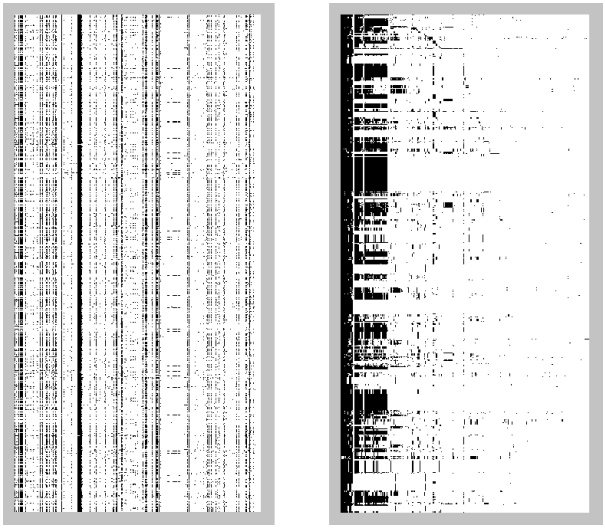
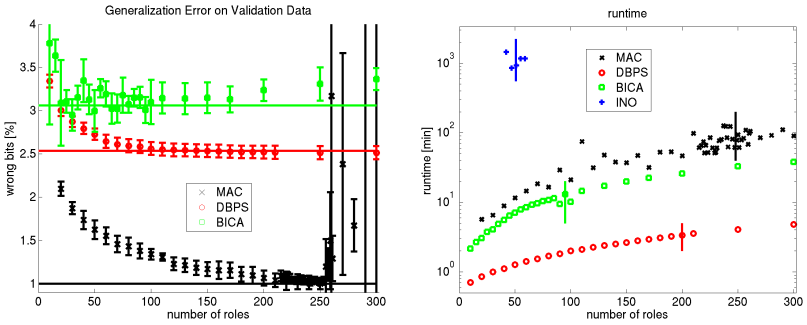


Figure 4.6: A 2400×500 part of the data matrix used for model-order selection. Black dots indicate a 1 at the corresponding matrix element and white dots indicate a 0. The full data matrix has size 4900×1300. Rows and columns of the right matrix are reordered such that users with the same role set and permissions of the same role are adjacent to each other, if possible. Note that there does not exist a permutation that satisfies this condition for all users and permissions at the same time.

the first 500 columns and the second 500 columns of the original user-permission assignment matrix by an element-wise OR operation to give the structure part $\bar{\mathbf{X}}^S$. Afterwards, we replace 33% of the matrix entries by random bits to yield the modified matrix $\bar{\mathbf{X}}$. This matrix exhibits a higher structural complexity and a substantially increased noise level than the original matrix.



(a) Generalization error (b) Runtime

Figure 4.7: Left: Generalization error on the hold-out validation set in terms of wrongly predicted bits versus the number of roles. The other external parameters for BICA and DBPS are determined by exhaustive search. Right: Runtime versus number of roles on a 2400×500 access-control matrix. The selected number of roles is highlighted by vertical lines.

Model-order selection

INO is a non-parametric model that can compute probabilities over the infinite space of all possible binary assignment matrices. It is therefore able to select the number of roles K during inference and needs no external input. For DBPS, BICA, and MAC, the number of roles must be externally selected and for DBPS and BICA, also rounding thresholds and approximation weights must be tuned. The number of roles K is the most critical parameter.

As a principle for guiding these model selection tasks, we employ the generalization error as defined in Section 4.4.1. We use five-fold cross-validation on a subset of 3000 users. Each time we split them into 2400 users for training the model parameters and 600 users for validating them, such that each user occurs once in the validation set and four times in the training set. The number of permissions used in this experiment is 500. The number of roles varies between 10 and 300. For a given number

of roles, we optimize the remaining parameters (of DBPS and BICA) on the training sets and validation sets. For continuous parameters, we quantize the parameter search-space into 50 equally spaced values spanning the entire range of possible parameter values.

To avoid an unnecessary long runtime for the MAC model optimization, we run the algorithm once with a large number of roles, $K = 100$, and check how many of the roles are involved in role combinations. A role that is involved in role combinations is at least once assigned to a user together with at least one other role. In our experiments, 10% of K are used in role combinations and no roles appear in combinations with more than two roles. Therefore, for subsequent runs of the algorithm, we set $M = 2$ and limit the number of roles that can belong to an assignment set to 10%. For large K, such a restriction drastically reduces the runtime. See Section 4.5 for an analysis of the runtime complexity of all investigated methods.

Restricting the number of roles that can belong to an assignment set risks having too few role combinations available to fit the data at hand. However, such circumstances cannot lead to underfitting when K is still to be computed. In the worst case, an unavailable role combination would be substituted by an extra single role.

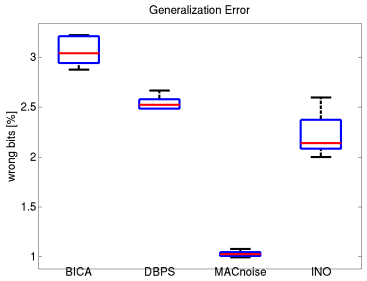
The performance of the three methods MAC, DBPS, and BICA on the validation data is depicted in Figure 4.7(a), left. The different models favor a substantially different number of roles on this data. For MAC, there is a very clear indication of overfitting for $K > 248$. For DBPS, the generalization error monotonically decreases for $K < 150$. As K further increases, the error remains constant. We observed that for a large number of roles, DBPS leaves extra roles empty, as explained by the combination-singlet confusion described in Section 4.4.2. We select for DBPS $K = 200$, where more roles provide no improvement. BICA favors a considerably smaller number of roles, even though the signal is not as clear. We select for BICA $K = 95$, which is the value that minimizes the median generalization error on the validation sets. INO internally selects 50 roles on average.

Results of different methods

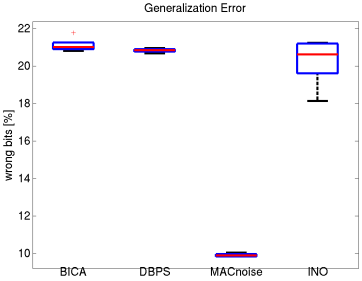
The results of the generalization experiments for the four methods MAC, DBPS, BICA, and INO are depicted in Figure 4.8. Overall, all methods have a very low generalization error on the original dataset. The error spans from 1% to more than 3% of the predicted bits. This result indicates that, on a global scale, the dataset has a rather clean structure. It should be stressed that most permissions in the input dataset are only rarely assigned to users, whereas some are assigned to almost everyone, thereby making up most of the 1s in the matrix (see a part of the dataset in Fig. 4.6). Therefore, the most trivial role set where roles are assigned no permissions already yields a generalization error of 13.5%. Assigning everyone to a single role that contains all permissions that more than 50% percent of the users have, achieves 7.1%. One should keep this baseline in mind when interpreting the results.

INO, DBPS, and BICA span a range from 2.2% generalization error to approximately 3% with significant distance to each other. MAC achieves the lowest generalization error with slightly more than 1%. It appears that INO is misled by its noisy-OR noise model, which seems to be inappropriate for this data. This can be seen from the estimated noise of MAC in Figure 4.10. MAC estimates the fraction of noisy bits by $\hat{\epsilon} \approx 2.8\%$ (Fig. 4.10(a)) and the probability for a noisy bit to be 1 by $\hat{r} \approx 20\%$ (Fig. 4.10(c)). This estimate clearly differs from a noisy-OR noise process (which would have $r = 1$). With more than 3% generalization error, BICA performs worst. As all other methods estimate a considerable role overlap, the assumption of independent (non-overlapping) roles made by BICA seems to be inappropriate here and might be responsible for the higher error.

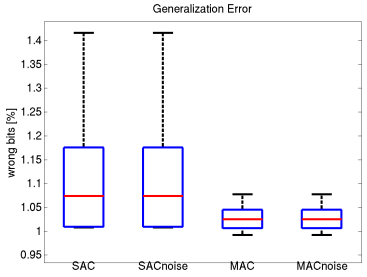
In our experiments on the modified dataset with more structure and a higher noise level, Figure 4.8(b), all methods have significantly higher generalization errors, varying between approximately 10% to 21%. The trivial solution of providing each user all those permissions that more than 50% of the users have, leads to an error of 23.3%. Again, MAC with 10% yields significantly lower generalization error than all the other methods. INO, DBPS, and BICA perform almost equally well each with



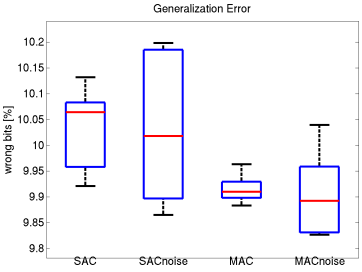
(a) All methods on original data



(b) All methods on modified data



(c) MAC variants on original data

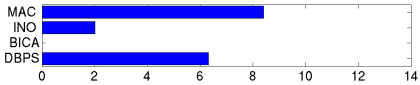


(d) MAC variants on modified data

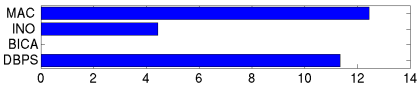
Figure 4.8: Generalization experiment on real data. The Graphs (a)-(d) show the generalization error obtained with the inferred roles.

a median error of 20% to 21%. A generalization error of 10% is still very good as this dataset contains at least 33% random bits, even though a random bit can take the correct value by chance.

Figure 4.9 shows the average role overlap between the roles obtained by the different methods. This overlap measures the average number of permissions that the inferred roles have in common. For BICA, the roles never overlap by the definition of the method. For all other methods, the increased overlap of the data's structure is reflected in the estimated roles. The fact that the difference in performance between BICA and the other models decreases after processing the modified dataset indicates



(a) Average role overlap (%) on original data



(b) Average role overlap (%) on modified data

Figure 4.9: Average role overlap in terms of permissions.

that the main difficulty for models, which can represent overlapping roles, is the increased noise level rather than the overlapping structure. We will return to the influence of the dataset in our discussion of the results of the MAC model variants in the next section.

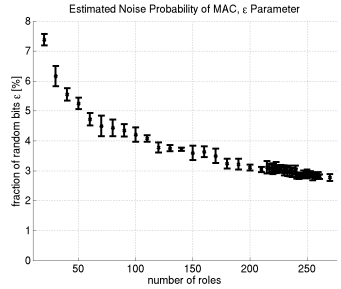
Results of MAC model variants

First, we report on the noise estimates of the MAC model. MAC estimates two noise parameters, the fraction of noisy bits ϵ and the probability r of a noisy bit to be 1, the noise bias. We report these estimates for both datasets and for a varying number of roles in Figure 4.10.

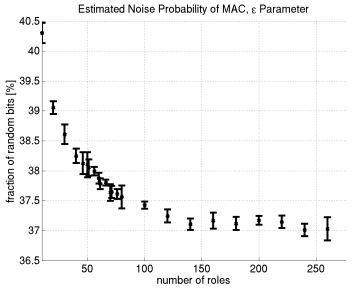
The top row of Figure 4.10 illustrates the trend of the estimated fraction of noisy bits $\hat{\epsilon}$ (please note the different scale). This fraction is monotonically decreasing with the increasing number of roles because additional roles enable the model to explain more of the bits in the data matrix. The model interprets all bits that are not explained by the roles as random bits. At the number of roles selected by the generalization error ($K = 248$) the estimate is $\hat{\epsilon} \approx 2.8\%$. For the modified dataset where the data has been merged by a disjunction and 33% has been added, the estimated noise fraction reflects this increased true noise level.

The noise bias is depicted in the lower row of Figure 4.10. For the original dataset the probability for a noisy bit to be 1 is estimated as $\hat{r} \approx 20\%$. This estimate is shifted to larger values for the modified

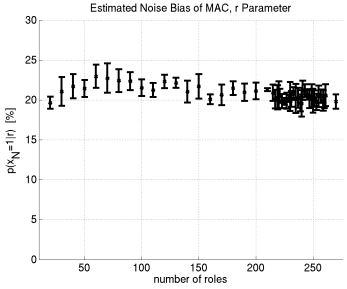
dataset because the noise that we artificially added has a bias of $r = 50\%$.



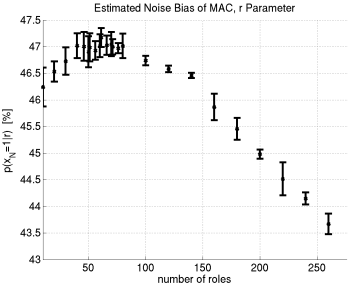
(a) Noise fraction on original data



(b) Noise fraction on modified data



(c) Noise bias on original data



(d) Noise bias on modified data

Figure 4.10: Estimated noise parameters of MAC. The left column accounts for the original dataset and the right column for the modified dataset.

To investigate the influence of the various model variants of MAC, we compare the performance reported above for MAC with i) the results obtained by the single-assignment clustering variant (SAC) of the model and ii) with the model variants without a noise part. The middle row of Figure 4.8 shows the generalization error of SAC and MAC, both with and without a noise model. On the original data set, Figure 4.8(c), all model variants perform almost equally well.

The noise model seems to have little or no impact, whereas the multi-

assignments slightly influence the generalization error. Taking MAC's estimated fraction of noisy bits $\hat{\epsilon} \approx 2.8\%$ into account, we interpret this result by refering to the experiments with synthetic data. There the particular model variant has no influence on the parameter accuracy when the noise level is below 5% (see Fig. 4.3(a)). As we operate with such low noise levels here, it is not surprising that the model variants do not exhibit a large difference on that dataset. On the modified data with more complex structure and with a higher noise level than the original data (Figure 4.8(d)), the difference between multi-assignments and single-assignments becomes more apparent. Both MAC and SAC benefit from a noise part in the model, whereas the multi-assignments have a higher influence.

4.5 Discussion of MAC variants

4.5.1 Model complexity of MAC and SAC

The complexity of the optimization problem is determined by the number of objects and features and by the number of possible assignment sets $L := |\mathbb{L}|$. As L can be large for even a small number of clusters, the complexity is dominated by that number. Let the number of clusters that a data item can simultaneously belong to be limited by the **degree** M, i.e. $\max_{\mathcal{L}\in\mathbb{L}} |\mathcal{L}| = M$. Then the size of the assignment set is bounded by

$$L = \sum_{m=0}^{M} \binom{K}{m} \leq 2^K \ . \tag{4.55}$$

Already for moderately sized K and M, this dependence results in computationally demanding optimization problems both for the inference step as well as for assigning new data items to previously obtained clusters. However, if the data at hand truly exhibits such a high complexity (high K and M) then also a single assignment model needs such a high complexity (otherwise it would underfit). In this case, a SAC model must learn L sources, while the MAC variant learns the L possible combinations out of K sources. The number of responsibilities $\gamma_{i\mathcal{L}}$ (Eq. 4.30) to

be computed in the E-step is the same for both models. However, in the M-step, MAC shares the source parameters while SAC must estimate them separately. We will shortly elaborate on the relationship between MAC and SAC from the inference perspective. Coming back to the complexity, the high number of responsibilities $\gamma_{i\mathcal{L}}$ to be computed for MAC appears to be a model-order selection issue. One can drastically reduce complexity by limiting the number of assignment sets as described in Section 4.4.3.

4.5.2 Relation between MAC and SAC

In the following, we show that MAC can be interpreted as a SAC model with a parameter sharing rule. In the limit of many observations, MAC is equivalent to SAC with proxy-sources substituting MAC's source combinations. In order to understand the parameter sharing underlying MAC, we write the set of admissible assignment sets $\mathbb{L}$ as a Boolean matrix $\mathbf{Z}^{\mathbb{L}} \in \{0,1\}^{L\times K}$. Assuming an arbitrary but fixed numbering of assignment sets in $\mathbb{L}$, $z^{\mathbb{L}}_{lk} = 1$ means that the l^{th} assignment set contains source k, and $z^{\mathbb{L}}_{lk} = 0$ otherwise. Hence, the assignment matrix $\mathbf{Z}$ decomposes into

$$\mathbf{Z} = \mathbf{Z}^{\mathcal{L}} \otimes \mathbf{Z}^{\mathbb{L}} \,, \tag{4.56}$$

where $\mathbf{Z}^{\mathcal{L}} \in \{0,1\}^{N\times L}$ denotes the exclusive assignment of objects to assignment sets ($z^{\mathcal{L}}_{il}$ iff object i has assignment set l, and $\sum_l z^{\mathcal{L}}_{il} = 1$ for all i). Using this notation, the decomposition $\mathbf{X} \approx \mathbf{Z} \otimes \mathbf{U}$ can be extended to

$$\begin{aligned}
\mathbf{X} &\approx \left(\mathbf{Z}^{\mathcal{L}} \otimes \mathbf{Z}^{\mathbb{L}}\right) \otimes \mathbf{U} && (4.57)\\
&= \mathbf{Z}^{\mathcal{L}} \otimes \left(\mathbf{Z}^{\mathbb{L}} \otimes \mathbf{U}\right) && (4.58)\\
&= \mathbf{Z}^{\mathcal{L}} \otimes \mathbf{U}^{\text{SAC}} \,, && (4.59)
\end{aligned}$$

where we have defined $\mathbf{U}^{\text{SAC}} := \mathbf{z}^{\mathbb{L}} \otimes \mathbf{U}$ as the proxy-source parameters of the single-assignment clustering model. The same notion of proxy-sources, substituting the disjunction of individual sources, is used in Eq. 4.1 for the probabilistic source parameters. Asymptotically, the

two models are equivalent. However, SAC must estimate $L \cdot D$ parameters, while the MAC model only uses $K \cdot D$ parameters. By sharing the parameters of the assignment sets, MAC reduces the number of parameters to be estimated and thereby increases the number of data items available per parameter. Moreover, the sharing rule provides a mutual inconsistency check for the involved parameter estimates. This check is not available if parameters are estimated independently. These two points explain the higher accuracy in the parameter estimators, which we observe in the experiments reported in Section 4.4.2.

4.5.3 Runtime

In our experiments on real-world data in Section 4.4.3, we monitored the runtime which is depicted in Figure 4.7(b). Each point represents the runtime for a single run of the different algorithms on an access-control matrix with $N = 2400$ users and $D = 500$ permissions. The number of roles chosen by the respective method is indicated by a vertical line. For INO we report the median number of roles selected. Note that in one run of INO, the model-order selection task is solved 'on-the-fly' while the other methods require multiple runs and an external validation. This overhead is reflected in the runtime. Considerable care is required in interpreting these results since the different methods were implemented by different authors in different languages (Matlab for INO, BICA and MAC, and C++ for DBPS). The DBPS implementation in C++ is impressively fast while the trend of the generalization error over the number of roles is roughly comparable to MAC and BICA. Thus, for large and demanding datasets, one could employ DBPS as a fast 'scout' to obtain an educated guess of the model-order. In conclusion, for all the investigated algorithms the runtime is not a limiting factor in role mining. This computation is only performed once when migrating an access-control system to another one. In this context it is not a problem if the computation takes hours.

4.6 Summary

We have presented and analyzed MAC, a probabilistic method to cluster vectors of Boolean data. Structurally, this model is an extension of the unconstraint flat RBAC model as derived in Chapter 3. Additionally, it provides an explicit noise model. We investigated several noise model variants and found that the mixture noise process is the most general one.

MAC is specifically tailored to role mining but can as well be used for clustering other Boolean vectors. In contrast to the conventional approach of mutually exclusive cluster assignments, our method enables a data item to belong to multiple clusters. We have observed that this enables one to make more accurate parameter estimates than with SAC, a disjoint clustering model of same model complexity. In experiments on real world access-control data, our model achieves significantly lower generalization error than the existing techniques INO, BICA, and DBPS.

Chapter 5

Hierarchical RBAC: Biclustering

In this chapter we investigate another instance of the model class derived in Chapter 3, the disjoint decomposition model (DDM). In comparison with MAC, DDM has one additional layer of roles and the constraints that a user can have exactly one role and a permission can be part of exactly one role. By adding prior assumptions to the model parameters, we convert DDM into an existing nonparametric Bayesian model for biclustering, the infinite relational model (IRM) and adopt the sampling algorithm proposed in [45] to inferring RBAC configurations. In Section 5.3.2, we will relate this model variant to MAC and experimentally compare these two models.

5.1 Disjoint decomposition model

In this section, we briefly recapitulate the disjoint decomposition model and then extend it with prior probability distributions for its parameters.

5.1.1 Revisiting and reformulating DDM

Let us recapitulate the structure of DDM. DDM assumes a three-way decomposition of the data: $\mathbf{X} = \mathbf{Z} \otimes \mathbf{V} \otimes \mathbf{Y}$. In the context of role mining, these entities have the following meaning:

- Users i to permissions d : $x_{id} \in \{0,1\}$, where $i \in \{1,..,N\}, d \in \{1,..,D\}$.
- Users i to business roles k: $z_{ik} \in \{0,1\}$, where $k \in \{1,..,K\}$.
- Business roles k to technical roles l: $v_{kl} \in \{0,1\}$.
- Technical roles l to permissions d: $y_{ld} \in \{0,1\}$, where $l \in \{1,..,L\}$.

Thereby the following constraints hold:

$$\forall i : \sum_k z_{ik} = 1 \tag{5.1}$$

$$\forall d : \sum_l y_{ld} = 1\,. \tag{5.2}$$

This means that a user can have only one role and a permission can be member of only one role. Therefore, the matrix $\mathbf{Z}$ introduces a disjoint clustering of the users and the matrix $\mathbf{y}$ introduces a disjoint clustering of the permissions. This property motivates the name disjoint decomposition model. There are no constraints for the matrix $\mathbf{V}$. Therefore, a user can have multiple technical roles. The relation between the three binary decomposition matrices of DDM is illustrated in Figure 3.2,c).

The property that makes DDM interesting from a technical point of view is that inference is much easier when the business roles and the technical roles are disjoint. This can be seen in the likelihood Eq. (3.29), which, in this case, takes a convenient form:

$$\begin{aligned} p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right) &= \prod_{i,d}\left[1-\prod_{k,l} p(\overline{v_{kl}})^{z_{ik}y_{ld}}\right]^{x_{id}}\left[\prod_{k,l} p(\overline{v_{kl}})^{z_{ik}y_{ld}}\right]^{1-x_{id}} \\ &= \prod_{k,l}\left[1-p(\overline{v_{kl}})\right]^{n_{kl}^{(1)}}\left[p(\overline{v_{kl}})\right]^{n_{kl}^{(0)}}\,. \end{aligned} \tag{5.3}$$

We have introduced the number of active assignments $n_{kl}^{(1)}$ and the number of inactive assignments $n_{kl}^{(0)}$ for each (user-group, permission-group) pair (k, l).

$$n_{kl}^{(1)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \mathbf{I}_{\{x_{ij}=1\}} \ , \tag{5.4}$$

$$n_{kl}^{(0)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \mathbf{I}_{\{x_{ij}=0\}} \ . \tag{5.5}$$

Thus, computing the data likelihood involves only counting.

The convenient form of this likelihood also enables the introduction of prior distributions for $p(\overline{v_{kl}})$. In the next section, we describe an existing model with particular priors for $p(\overline{v_{kl}})$ and the role assignments $\mathbf{Z}$ and $\mathbf{y}$. We investigate a sampling algorithm for inferring model parameters for this particular instance of the model class.

5.1.2 A nonparametric Bayesian model variant

In this section, we introduce prior assumptions for DDM that renders it to the infinite relational model (IRM) [45]. The underlying probability distributions of the binary assignment variables $p(\overline{u_{kj}})$ are assumed to be Bernoulli, i.e.,

$$\begin{aligned} p\{v_{kl} = 0 \mid \mathbf{Z}, \mathbf{Y}, \boldsymbol{\beta}\} &= \beta_{kl} \\ p\{v_{kl} = 1 \mid \mathbf{Z}, \mathbf{Y}, \boldsymbol{\beta}\} &= 1 - \beta_{kl} \, . \end{aligned} \tag{5.6}$$

Thus, knowing the membership of users and permissions, the probability of a user being assigned to a member of the permission-group l is the same for all users of the role k. It is the probability of the assignment v_{kl} of business role k to technical role l. Under this assumption, the data likelihood of DDM is

$$p\left(\mathbf{X}|\mathbf{Z}, \mathbf{Y}, \boldsymbol{\beta}\right) = \prod_{k,l} \left[1 - \beta_{kl}\right]^{n_{kl}^{(1)}} \left[\beta_{kl}\right]^{n_{kl}^{(0)}} \ . \tag{5.7}$$

Additionally, we assume that β_{kl} itself is a random variable that is generated by sampling from a Beta distribution $P_b\left(\beta_{kl}; \gamma, \gamma\right)$ with the

positive hyperparameter γ. This Bayesian perspective enables one to make all prior assumptions on β_{kl} explicit by differently selecting γ. Given the hyperparameter γ, the individual β_{kl} are independent from each other.

The so-called **evidence** term $p(\mathbf{X}|\mathbf{Z},\mathbf{Y})$ is of particular importance for inferring the assignment variables in $\mathbf{Z}$ and $\mathbf{y}$ as we will see later. This term denotes the probability of the permissions of a particular user given the assignments to a user-group (and all assignments of permissions to permission groups). The complete evidence for all users and permissions is

$$p(\mathbf{X}|\mathbf{Z},\mathbf{Y}) = \int p(\mathbf{X},\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y})\, d\boldsymbol{\beta} \tag{5.8}$$

Thereby, $p(\mathbf{X},\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y})$ involves a product of the likelihood as in Eq. (5.7) and the Beta distribution $P_b(\beta_{kl};\gamma,\gamma)$. The Beta distribution is a so-called conjugate prior of the Bernoulli distribution. Therefore this integral can be carried out analytically. We provide this calculation in Appendix A.2. The evidence term becomes

$$p(\mathbf{X}|\mathbf{Z},\mathbf{Y}) \quad = \quad \prod_{k,l} \frac{B(n_{kl}^{(1)}+\gamma, n_{kl}^{(0)}+\gamma)}{B(\gamma,\gamma)}\,. \tag{5.9}$$

Here $B(.,.)$ is the Beta function that appears as a normalization factor in the Beta distribution $P_b(.;.,.)$.

We also treat the assignments $\mathbf{Z}$ and $\mathbf{Y}$ as random variables. They are generated by a Dirichlet process [5, 22]. We do not fully discuss the derivation and all properties of the Dirichlet process here and explain only what is needed for the sampling algorithm. In our application, the Dirichlet process provides the a priori probabilities of the assignments of users to business roles without knowing the permissions for these users. We also use it for the assignments of permissions to technical roles $\mathbf{Y}$ but explain it using $\mathbf{Z}$ only. The permission assignments can be sampled in the same way (even with the same code, technically, by simply transposing the input matrix). Under the Dirichlet process the individual assignment variables z_{ik} are not independent of each other. However,

the z_{ik} are exchangeable meaning that sampling of these variables can be done in arbitrary order. Thereby, the conditional probability of one variable given all the others has a convenient and intuitive form.

$$p\left(z_{i'k} = 1 | \mathbf{Z}_{i \neq i'}, \alpha\right) = \begin{cases} \frac{N_k}{N-1+\alpha} & N_k > 0 \\ \frac{\alpha}{N-1+\alpha} & N_k = 0 \end{cases} \tag{5.10}$$

N is the total number of users and N_k is the number of users that have role k (the cardinality of the role). The Dirichlet process is parametrized by the nonnegative **concentration parameter** α. We use the shorthand notation of $\mathbf{Z}_{i \neq i'}$ for the role assignments of all users except user i'. The Dirichlet process penalizes assignments according to the number of roles and their cardinalities. The probability to be assigned to a particular role k is proportional to the number of users with this role N_k. There is a positive probability, proportional to α, to be assigned to a newly created role that no user has. Therefore, α can be interpreted as the cardinality that each hypothetical role has in advance. The Dirichlet process can, in theory, generate an infinite number of roles, but it penalizes unnecessary roles by weighting these user-group probabilities proportional to their cardinality N_k.

We graphically illustrate the DDM model with all introduced prior assumptions in Figure 5.1. There, we use different hyperparameters α_1 and α_2 for $\mathbf{Z}$ and $\mathbf{Y}$. However, in all calculations (and in practice) we use the same priors $\alpha = \alpha_1 = \alpha_2$.

5.2 Inference

5.2.1 A Gibbs sampling algorithm

We now describe a sampling algorithm that can be used to infer the model parameters of the Disjoint Decomposition Model. The algorithm randomly samples assignments from a Dirichlet process mixture model by using Gibbs sampling for Bayesian models with conjugate priors as presented in [60] (Algorithm 3). Here, we summarize it using our notation.

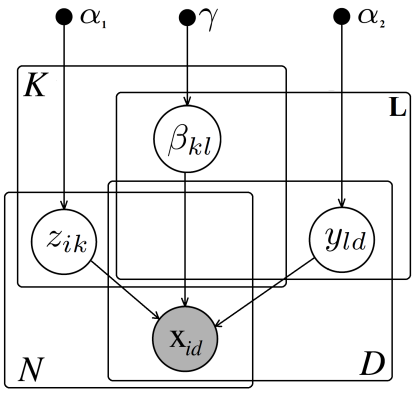


Figure 5.1: Graphical model for DMM with prior distributions. Semantics: Filled circles are observable random variables and empty circles are hidden. Solid points denote fixed parameters of the model. Arrows indicate dependencies. Entities on a N-plate exist in N different realizations.

The main idea behind the inference algorithm is to iteratively assign users to user-groups by altering the entries of $\mathbf{Z}$. In each iteration, each user is randomly assigned to a group according to the Dirichlet process probability that the user belongs to that group (this can also be the group that the user already has been assigned to in the previous iteration). Alternatively, again with a probability that must be computed at each step, the user creates a new user-group with the user being the only member. In later iterations, other users may be assigned to that user-group as well. User-groups that remain empty after an update step are deleted.

After each iteration over the users, the same procedure is performed on the permission assignments $\mathbf{Y}$, while the user-role assignments $\mathbf{Z}$ are kept constant. Theoretically, Gibbs sampling reaches the global optimum in the limit of infinite time. In practice, the algorithm terminates either upon convergence or after reaching a predefined maximal number of iterations. As we randomly sample from this distribution, the

algorithm can also reach suboptimal states. We always keep the state $(\mathbf{Z}^*, \mathbf{Y}^*)$ in memory that has maximal posterior probability $p(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ and return this state as an output of the algorithm.

As described, the behavior of the Gibbs sampler is determined by the probability distributions over the assignments. In each step, the Gibbs sampler successively calculates the probabilities of the following $K + 1$ exclusive events for each user i': assign i' to one of the K existing user-groups k or create a new role with index $k = K + 1$. To randomly assign a user to an existing user group, we only must compare the probabilities of different choices with respect to each other. Hence, we only must compute the probabilities up to a constant factor.

$$p(z_{i'k} = 1 | \mathbf{X}, \mathbf{Z}_{i \neq i'}, \mathbf{Y}) = \text{const} \cdot p(\mathbf{X} | \mathbf{Z}, \mathbf{Y})\, p(z_{i'k} = 1 | \mathbf{Z}_{i \neq i'}) \quad (5.11)$$

This is the product of the evidence term Eq. (5.9) and the Dirichlet prior Eq. (5.10). As a consequence, roles are favored where all other users have similar permissions as user i' (which always perfectly holds for the newly created role) and which many users have (which is never true for the newly created role). When assigning the user i' to the user-group k, the current user-role assignments for all users other than i' are used to compute the probabilities. Note that Eq. (5.11) is written for sampling steps on the users. When the algorithm runs on permissions, then z must be replaced by y, k by l, and i by d, throughout the formula.

In summary, the algorithm assigns rows of the user-permission assignment matrix (users) to groups where the other rows (members of the same group) are similar to that row. For such groups, the assignment probability distributions have very sharp maxima compared to other suboptimal assignments. These random assignments are computed by alternating iterations over the rows and columns, as can be seen in Algorithm 1. Thereby, each sweep clusters the original user-permission assignment matrix $\mathbf{X}$ by alternately moving whole rows and columns.

The matrices in Figure 5.2 illustrate how this procedure looks in practice. Rows are users, columns are permissions, and dots indicate assignments between users and permissions. The assignments in (a) are randomly generated according to the general model (3.29). The roles are then inferred with the Disjoint Decomposition Model (5.3). Since

all business roles and all technical roles are disjoint, the original matrix can be ordered according to the members of their entities, as depicted in (b). As described in Section 5.2.2, erroneous assignments and missing assignments can easily be identified in this arrangement. They could be reported or even automatically removed if desired, as shown in (c).

input : binary matrix $\mathbf{X}$ as defined in Section 3.2,
α, γ, $iter_{\max}$ and $d_{\min}$
output: role assignment matrices $\mathbf{Z}^*$ and $\mathbf{Y}^*$

$\mathbf{Z} \leftarrow \mathbf{0}$; $z_{i1} \leftarrow 1$, for all i
$\mathbf{Y} \leftarrow \mathbf{0}$; $y_{1d} \leftarrow 1$, for all d
$(\mathbf{Z}^*, \mathbf{Y}^*) \leftarrow (\mathbf{Z}, \mathbf{Y})$
for $iter \leftarrow 1$ **to** $iter_{max}$ **do**
    $\mathbf{Z}^{\text{old}} \leftarrow \mathbf{Z}$; $\mathbf{Y}^{\text{old}} \leftarrow \mathbf{Y}$
    for $i \leftarrow 1$ **to** N **do**
        $z_{ik} \leftarrow 0$, for all k
        $z_{ik} \leftarrow 1$, for one k sampled from Eq. 5.11
    end
    for $d \leftarrow 1$ **to** D **do**
        $y_{ld} \leftarrow 0$, for all l
        $y_{ld} \leftarrow 1$, for one l sampled from Eq. 5.11
    end
    if $p(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) > p(\mathbf{X}, \mathbf{Z}^*, \mathbf{Y}^*)$ **then**
        $(\mathbf{Z}^*, \mathbf{Y}^*) \leftarrow (\mathbf{Z}, \mathbf{Y})$
    end
    if $\left(d_{min} < d\left(\mathbf{Z}^{old}, \mathbf{Z}\right)\right) \wedge \left(d_{min} < d\left(\mathbf{Y}^{old}, \mathbf{Y}\right)\right)$ **then**
        break
    end
end

Algorithm 1: Gibbs sampling algorithm for DDM

As outlined in Algorithm 1, in every iteration of the main loop, the function $d(\cdot, \cdot)$ computes the fraction of entries of the current assignment matrices $\mathbf{Y}$ and $\mathbf{Z}$ that differ from their last state. The loop repeats until

either the assignment matrices converge to a stable configuration or it breaks after a predefined maximal number of iterations.

5.2.2 Error detection and subpartitioning

The algorithm just described infers the assignments of users to user-groups and permission to permission-groups. These are the assignment matrices $\mathbf{Z}$ and $\mathbf{Y}$. We now explain how we determine the assignments between business roles and technical roles $\mathbf{V}$ given $\mathbf{Z}$ and $\mathbf{Y}$.

A pair consisting of a user-group and a permission-group (business role and technical role) is called a bicluster. Each bicluster has one empirical estimator $\hat{\beta}_{kl}$ of the probability that the entire user-group is assigned to the permission-group. All members of this bicluster share this assignment probability $\hat{\beta}_{kl} = (n_{kl}^{(1)} + \gamma)/(n_{kl}^{(1)} + n_{kl}^{(0)} + 2\gamma)$. Based on $\hat{\beta}_{kl}$, one must decide whether to set the corresponding u_{kl} to 1 or 0. To solve this, we employ the threshold ϵ,

$$u_{kl} = \begin{cases} 1 & \hat{\beta}_{kl} \geq 1 - \epsilon \\ 0 & \hat{\beta}_{kl} < 1 - \epsilon \,, \end{cases} \tag{5.12}$$

where ϵ is the noise ratio that we expect in the data. This is the estimated percentage of wrong assignments that do exist in the original user-permission assignment matrix $\mathbf{X}$. Given $\hat{\beta}_{kl}$, we can identify erroneous or missing assignments and automatically correct them or alert an administrator to check these inconsistencies. As an example, if we assume 5% noise in the data and find a bicluster with $\hat{\beta}_{kl} = 0.98$, then the 2% of missing assignments are very likely to be errors. Figure 5.2(c) provides an example of an assignment matrix after correcting such errors.

We denote a user-group to permission-group assignment with probability $\hat{\beta}_{kl}$, where $\epsilon < \hat{\beta}_{kl} < 1 - \epsilon$, as an "indifferent bicluster". This means that, for the probability $\hat{\beta}_{kl}$, the deviation of this indifferent bicluster from a pure bicluster cannot be explained by the overall noise in the data. There are instead two alternative explanations:

1. There is actually no structure for the assignments within the bicluster. All assignments are exceptions. All missing assignments are intentionally missing.

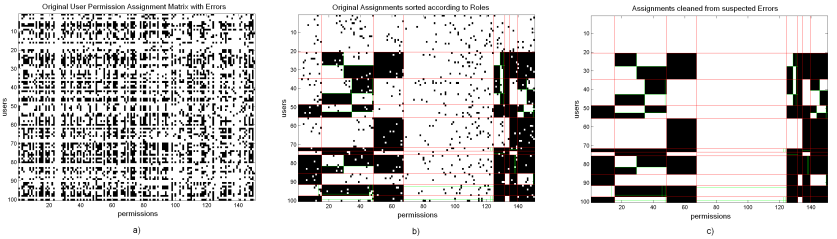


Figure 5.2: a) Illustration of a randomly generated user-permission assignment matrix. b) Users and permissions ordered according to the result of the Gibbs sampling algorithm for the Disjoint Decomposition Model (DDM). c) Possibly erroneous assignments detected.

2. The underlying structure within the bicluster was not discovered by the Gibbs sampler.

The event of not discovering structure can happen due to the following reasons. The assignments of users to user-groups in a given sampling step are randomly drawn according to the probabilities of the user being assigned to that role. These probabilities factor into a product over the evidence (5.9) of all the permission-groups that the permissions of the user are assigned to. Members of indifferent biclusters (k', l') have very high factors in (5.9) for their parts lying within the neighboring biclusters $(k', l \neq l')$. Thus, the probability for them to be reassigned to their user-group k' is very high. As a result, in every iteration, these members are again reassigned, even if a small fraction of their permissions lies within a suboptimal bicluster (k', l').

We graphically illustrate the problems of a disjoint clustering in Figure 5.3, left. In this example matrix solid lines are current partitions and dashed lines are hypothetical partitions. Areas with the same color have the same probabilities β for an assignment. We highlight three situations. i) good partition: The biclusters in region 'h' are perfect because β is homogenous within the partitioned biclusters. ii) unnecessary split: the solid line marked with 'f' was inferred due to the discrepancy between the white and the brownish areas. Thereby, it separates user that

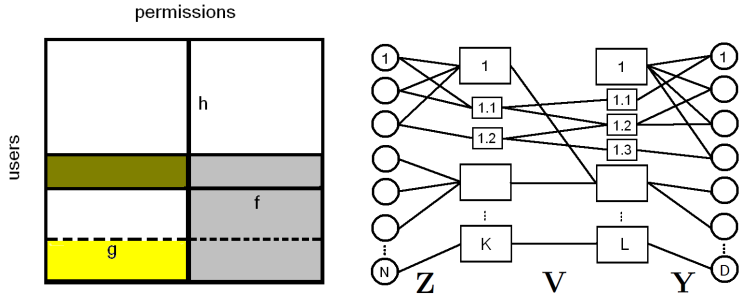


Figure 5.3: Left: Inference problems due to disjoint clustering. See description in text. Right: Extension of the Disjoint Decomposition Model to independent sub-roles. For each pair of business-role and technical-role, the corresponding members may also belong to subroles.

all have the same (the gray) permissions. iii) Inhibited split: The dashed line marked with 'g' is required to separate white and yellow users. However, this split is inhibited by the good match of these users in terms of their gray permissions (resulting in a high evidence term of repeatedly reassigning them to the same big cluster). Error ii) is less likely than Error iii) because the Dirichlet prior supports big clusters and prevents small clusters. We observed that the Gibbs sampler drops states with unnecessary splits after a few steps (sometimes thereby producing an Error of type iii)).

A pragmatic way to check whether a particular bicluster has some hidden structure is to partition the given bicluster into smaller biclusters. This can be done by considering all user-permission assignments within the bicluster as a smaller independent assignment matrix $\mathbf{X}'$ and to run the algorithm again independently on $\mathbf{X}'$. If there is a structure that is hidden by the neighbors, this procedure will discover it. If not, the algorithm will return the identical result as before.

An alternative way to enforce splits would be to choose a very high hyperparameter α for the Dirichlet process. However, this parameter choice renders the rough structure of the data nearly invisible to the

algorithm and results in a high number of very small roles. In contrast, independent subpartitioning preserves the large-scale structure and can *simultaneously* find focused roles for more specialized users (or rarely used permissions).

Extending the model to the concept of subpartitions or focused roles leads to a model structure that differs slightly from the original Disjoint Decomposition Model. Originally, each user is constrained to only one business role. This assumption still holds for the dominant roles themselves, but now additionally for each user-group permission-group pair (for each bicluster) the members may be additionally assigned to (a single!) focused bicluster. Figure 5.3 (right) illustrates this structure. The focused role assignments are inferred independently from the rest of the data successively after the main assignments are found. Therefore, both decompositions are independent of each other and the original constraints still hold within each level of the model. Also, there are no assignments between focused business roles and dominant technical roles and vice-versa.

5.3 Experiments

In this section, we assess the performance of the Gibbs sampler for the Disjoint Decomposition Model (DDM) in comparison with DBPS and MAC. We perform experiments using randomly generated data and real enterprise data. We study two different setups. First, we measure how well the roles of DDM and DBPS represent generated access-control data under the influence of randomly perturbed assignments. As a second experiment, we compare DDM with MAC and investigate their behavior in model mismatch situations.

5.3.1 Inference under noisy conditions

We generate a user-permission assignment matrix $\mathbf{X}$ with 200 users, 200 permissions, 10 business-roles, and 5 technical roles, as follows. For each user, the memberships in one or more business-roles are randomly drawn and indicated in the assignment matrix $\mathbf{Z}$. Similarly, memberships in one

or more technical roles are drawn for each permission (thereby creating $\mathbf{Y}$). Finally, assignments between business roles and technical roles $\mathbf{V}$ are randomly drawn. The final user-permission assignment matrix $\mathbf{X}$ is then $\mathbf{X} = \mathbf{Z} \otimes \mathbf{U} \otimes \mathbf{Y}$. To control the difficulty of the experiment, we add errors to this matrix by randomly flipping a given fraction of the entries of $\mathbf{X}$ from one to zero or from zero to one. This results in both erroneous and missing assignments. In Figure 5.2(a), one can see a user-permission assignment matrix $\mathbf{X}$ that has been generated this way (for illustrative purposes we have chosen a lower number of users and permissions for this figure than in the experiment). In principle, one could also generate structured noise, i.e. complete roles that are wrong or missing. However, since these are exactly the patterns the algorithms are searching for, they would have no chance to discriminate them from valid structures (if there is no side information from other sources) and this in turn would not allow for a comparison of the methods.

On the generated assignment matrix $\mathbf{X}$, we perform role-mining using the algorithm for the Disjoint Decomposition Model (DDM) and the Discrete Basis Problem solver (DBPS). An average run on that data with DBPS, implemented in C++, requires 1.3 seconds. DDM, with a much less efficient MATLAB implementation, needs 40 seconds. An example of a user-permission assignment matrix approximated by DDM is given in Figure 5.2(c).

From the two resulting approximations of the noisy user-permission assignments matrix, we compute the differences to the original noiseless matrix (the ground truth). Thereby, we distinguish between two measures: the relative number of assignments that have been added with respect to the ground truth (the ratio of wrong assignments) and the relative number of assignments that have been removed with respect to the ground truth (the ratio of missing assignments). These two quantities have a different security-relevance. Granting someone a permission that he *should not* have is usually more serious than not assigning a permission that he *should* have. The former may result in a security breach whereas the latter will typically only result in a call to the help desk, when the missing permission is actually needed.

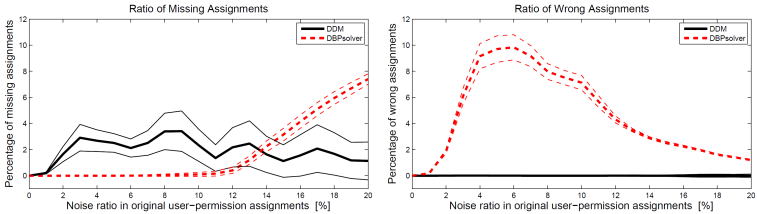


Figure 5.4: Differences to the correct user-permission assignment matrix. The dashed line represents results of DBPS and the solid line results of the Gibbs sampler for DDM. The thin lines give the standard deviation over all runs. Both methods approximate an erroneous assignment matrix with a varying percentage of errors (noise). The left plot gives the ratio of missing assignments after the approximation and the right plot the ratio of wrong assignments.

Discussion In Figure 5.4, we plot the described quality measures versus the noise level in the generated assignment matrix $\mathbf{X}$. This noise level runs from 0% to 20%. Both plots have the same scale. As indicated in the left plot, DBPS can cover every existing assignment to roles up to a noise level of 12%. In doing so, it even covers the assignments that are missing in the original data due to noise. However, this can only be done at the cost of granting wrong permissions, as can be seen by comparison with the right plot: At noise levels where DBPS fits most assignments, there is a high percentage of wrong assignments. Due to this tradeoff, the two curves of DBPS run contrary to each other. The problem is that the algorithm is unable to identify noise. It can only decide between having a very good coverage at the cost of wrong assignments or avoiding wrong assignments at the cost of poor coverage. In contrast to the experiments in Figure 4.4(b), DBPS achieves a perfect fit when there is no noise in the data. This is an effect of the way how the data is generated. While here we use a model with disjoint roles, in Section 4.4.2 we used overlapping roles to generate the data. With overlapping roles the effect of combination-singlet confusion occurs as discussed in Section 4.4.2. Here, we can not observe this effect.

As we see, DDM's results vary only slightly over the entire noise range. One can conclude that it is robust to erroneous data withing the investigated noise range. As one can clearly see in the right plot, DDM produces a strictly conservative approximation of the original assignments. No additional permissions with respect to the ground truth are granted. The only added permissions are those that have been missing in the noisy data. In turn, all wrong permissions of the original data are discovered and set to zero. The ratio of true assignments that are missed by DDM are, on average, 2% over the whole scale. For high noise levels this is superior to DBPS. For DDM there is no tradeoff between coverage and conservativeness. DDM only adds assignments if there is high evidence that an assignment is missing due to an error.

In summary, the results obtained by DDM are more robust to noise than those of DBPS. This is desirable, since, in a real domain, one usually does not know how many errors are in the existing user-permission assignments. Even more important, DDM finds and corrects for wrong assignments while DBPS migrates them. For small noise levels, DBPS achieves higher coverage rates while for high noise levels DDM is superior. Over the entire noise range, DDM is better at avoiding wrong assignments.

5.3.2 Comparison with MAC

In this section, we compare DDM with the model for multi-assignment clustering (MAC). Both models stem from the same core model as derived in Chapter 3. They differ in the following points. First, DDM has one extra layer of roles. This additional layer, encoded in the assignment matrix $\mathbf{y}$, creates a clustering of the permissions. Therefore, DDM has a two-level hierarchy while MAC models flat RBAC. Second, DDM has additional constraints on its assignment matrices $\mathbf{z}$ and $\mathbf{y}$ dictating that the business roles must be disjoint in terms of their users and the technical roles must be disjoint in terms of their permissions. Thereby, the assignment of business roles to technical roles has no constraints. There are no constraints at all for MAC. A user can have multiple roles and permissions can be assigned to multiple roles. The last difference

of the two models are the prior assumptions on the model parameters. While MAC implicitly assumes uniform prior probabilities for its parameters, DDM makes explicit non-uniform assumptions encoded in the Beta priors and the Dirichlet priors.

The most interesting question that arises when going through the list of differences of the two models is how the different properties of the model influence the quality of the inferred RBAC configurations. To examine which of the two model variants is best suited to solve the role mining problem, we design the following experiment. We generate access control data in two different ways. One half of the datasets is generated according to the MAC model, in the way described in Section 4.4.2. As a consequence, some users in these datasets are generated by multiple sources. The second half of the datasets is generated from the DDM probability distribution by repeatedly sampling business roles and technical roles from the Dirichlet process priors and connecting them according to the Beta-Bernoulli probabilities. On both kinds of data sets, we infer RBAC configurations with DDM and with MAC. In this way, the model assumptions always perfectly hold for one of the models while the other one operates in a model-mismatch situation. Moreover, data from the DDM can have an arbitrary number of underlying roles. Therefore, we must train MAC with an additional model-order selection mechanism. As in Section 4.4.3, we use the transfer costs as an external validation criterion.

In Section 2.6.3, we identified the generalization ability of the solution as the most relevant quality measure for the inference role mining problem (Definition 2.6-1). Therefore, we evaluate the inferred roles based on the generalization error. We control the difficulty of the experiments by varying the noise level in the data. At each noise level we sample 30 datasets from each model variant. On each dataset, we run each model 10 times and select the solution with the highest internal quality score, respectively.

Discussion We report the median generalization error of the inferred matrix decompositions and the 25%/75%-percentiles in Figure 5.5. The left plot depicts the generalization error on DDM data and the right plot

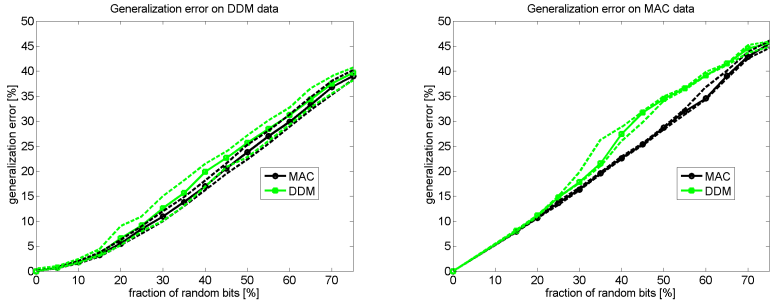


Figure 5.5: Comparison of the generalization error of MAC and DDM on data generated from DDM sources and MAC sources respectively.

depicts the error on MAC data.

As a first observation, we note that the overall trend of both models is similar for both kinds of data. The generalization error increases with increasing noise. This has two reasons. For high noise the problem of estimating the structure of the data becomes increasingly difficult when increasing the noise level. In addition, noisy bits are likely to be wrongly predicted, even when the data structure is learned well.

Our second observation is that on DMM data MAC and DDM generalize almost equally well. MAC is even slightly better than DDM. In contrast, on MAC data DMM achieves a worse generalization error than MAC in the intermediate noise range. One would expect that each model generalizes best on data that is generated according to the model assumptions. This can be observed on MAC data. However, on DDM data MAC is as good as DDM.

The reason is that for DMM data, the model assumptions of MAC are in fact not violated. Even though DMM has an extra layer of roles, this model instance is less complex than flat RBAC (which MAC models). One can see this by collapsing one DDM layer. For instance, let $\mathbf{u}'$ be $\mathbf{v} \otimes \mathbf{y}$. Then permissions can be assigned to multiple roles (because there are no constraints for the business-role to technical-role assignment $\mathbf{v}$). At the same time, $\mathbf{z}$ still provides a disjoint clustering of the users. In

this flat RBAC $(\mathbf{z}, \mathbf{u}')$ the roles overlap in terms of permissions but not in terms of users. This is the same model structure as in single-assignment clustering (SAC). As a consequence, we can interpret DDM as a SAC model with prior probabilities on the model parameters. As we have analyzed in Chapter 4, SAC yields inferior parameter estimates than MAC and, as a result, a larger generalization error.

It turns out that besides the structural difference between DDM and MAC, the other differences, the optimization algorithm and the Bayesian priors, have only a minor influence on the results. It seems like MAC can compensate for a missing internal mechanism for finding the appropriate number of roles, when an external validation step is provided. Also, the Gibbs sampling scheme and the deterministic annealing algorithm perform equally well on the DDM data. Given that the prior Beta distributions for the Bernoulli variables of the DDM provide no improvement over MAC, it seems unnecessary to extend MAC with such priors. Moreover, as we explain in Appendix A.3, Beta priors and the MAC likelihood are hard to combine due to the non-conjugancy of these two distributions. Inference is still possible with non-conjugate priors but is is computationally more demanding. Given the negligible benefit of priors for the DDM, this extra effort is too large.

Given that DDM is structurally equivalent to SAC, we can interpret its generalization error on MAC data using our insights from Chapter 4. There we have seen, that SAC and MAC infer the model parameters almost equally well for low noise levels. This explains the equal generalization error in this range ($\epsilon < 25\%$) as depicted in Figure 5.5, right. For higher noise levels the accuracy of SAC breaks down when MAC still yields good parameter estimates. This is the noise range where the generalization error of DDM and MAC deviate the most ($25\% < \epsilon < 65\%$). For even higher noise levels ($\epsilon > 65\%$), where learnability of the structure is almost impossible, also the accuracy of MAC drops. Then both methods, DDM and MAC, generalize equally bad.

We conclude that MAC is a more general model than DDM. Thereby, the combination of MAC and the minimum transfer cost principle for model-order selection is able to select the appropriate number of roles without making explicit prior assumptions on the distribution of this

number. As a consequence, we recommend using the MAC model for role mining with real-world access-control datasets.

5.4 Summary

In this chapter, we have investigated the Disjoint Decomposition Model. This is a two-level hierarchy RBAC model with the constraints that users have exactly one business role and permissions are assigned to a single technical role. Thereby, a business roles can be super-role of multiple technical roles. As a consequence of the constraints, DDM provides a *disjoint* biclustering of the access-control matrix.

By adding prior distributions for the model parameters, we converted DDM to the nonparametric Bayesian Infinite Relational Model (IRM) for biclustering which was proposed in [45]. We described a Gibbs sampling algorithm for inferring RBAC configurations with DDM and analyzed the impact of the disjointness constraint.

We experimentally investigated two properties of DDM. First, DDM is more robust to overfitting than the combinatorial algorithm DBPS. Second, DDM achieves inferior generalization ability than the multi-assignment model MAC. Interestingly, MAC generalizes better not only on data generated from a MAC distribution but generalizes as well on data that has been generated from a DDM distribution. The reason is that, structurally, DDM is a special case of MAC that is equivalent to Single-Assignment clustering (SAC). This enables MAC to fit all possible realizations of DDM data. Thereby, the advantage of DDM to internally infer the number of roles can be compensated by MAC when using an external model-selection strategy (like the minimum transfer cost principle). We therefore recommend using the MAC model for role mining.

Chapter 6

Hybrid role mining

6.1 Introduction

As it is generally understood, "role engineering" [15] is the task of configuring a Role-Based Access-Control (RBAC) system, i.e., creating roles and assigning users to roles and roles to permissions. This term has been further refined into "top-down" and "bottom-up" approaches to role engineering [21]. Top-down role engineering uses descriptions of business processes, security policies, and other business information to configure an RBAC system. The bottom-up variant uses existing direct assignments between users and permissions, such as access-control lists (ACLs). Both these terms "top-down" and "bottom-up" have also been used in the context of role mining, where "bottom-up role mining" is often abbreviated simply as "role mining". We did so in the preceding chapters.

Top-down role engineering approaches complement bottom-up role-mining methods in terms of their strengths and weaknesses. Role mining algorithms are cheap compared to top-down approaches. They often achieve a good fit with the existing user-permission assignments but sometimes they discover roles that are difficult to interpret from a business perspective and that are cumbersome for administrators to

work with, e.g., to maintain the RBAC configuration as the enterprise evolves. A good RBAC configuration should correspond to the business properties of the enterprise and, at the same time, approximately fit the currently implemented access-control configuration. We included these requirements in our definition of the role mining problem in Section 2.6 by adding the assumption that top-down information TDI influences the underlying RBAC configuration RC^* that must be found.

In this chapter, we propose an approach to *hybrid role mining* that incorporates top-down business information into a bottom-up role-mining process and thereby combines the strengths of both approaches. Our method has two parts, with associated measures, models, and algorithms.

1. Identify business information and determine its relevance for roles with respect to the given access-control data.

2. Incorporate the relevant data into the role mining process itself.

In the first part, we begin by identifying business information that could be relevant for roles. Since roles should represent functions within an enterprise, the enterprise's human resources (HR) department is likely to be a good source of such data, e.g., employees' positions, working groups, locations, etc. But not all such data is equally relevant. Indeed, there is unlikely to be any business information that precisely captures the roles of all system users. For example, employees with the same position (group or location) usually differ in some of their permissions. Moreover, simply using all available data is not the solution: It not only increases the computational cost of hybrid role mining, it can actually lead to worse results, as we will see later (Section 6.2). Hence it is necessary to select the most relevant business information for use in hybrid role mining. To support this selection, we define an appropriate entropy-based notion of *relevance* and show how to compute it.

In the second part, we present a method to incorporate business information into a role mining algorithm based on the MAC model. MAC encodes how likely it is that a particular RBAC configuration underlies a given user-permission assignment. We combine this model with an

objective function on business information that optimizes business relevance: users with the same business attribute should ideally be assigned to the same set of roles.

The remainder of this chapter is organized as follows. In Section 6.2, we introduce our relevance measure for business attributes. In Section 6.3, we develop our model for hybrid role mining and the corresponding algorithm. We report on experimental results obtained with data from a real-world enterprise in Section 6.4. Finally, in Section 6.5, we examine related work in hybrid role mining.

6.2 Entropy-based relevance measures

Enterprises maintain different types of business information about each user. Examples include a user's working address, job code, organization unit, etc. We encode predicates specifying which users i have the business-information attribute s (for example, which users work in the accounting department) as a family of Boolean variables w_{is}. The variable w_{is} has the value of 1 if the user i has attribute s, and 0 otherwise. We shall assume that for each type of business information, each user has a single attribute s, e.g., a user is member of exactly one department.

An abundance of information is usually available in digital form within an enterprise, but most of it is ill-suited for hybrid role mining. To be useful, the data must provide information about the relationship between employees and the permissions they have been granted. In this section, we provide a measure that quantifies to what extent a given type of business information agrees with the direct user-permission assignment. When the agreement is high, we say that the data is relevant because it increases the information about whether a user has a particular permission.

Business information with too little relevance can actually lead to worse role mining results. This deterioration occurs when the objective of agreement between roles and business information conflicts with the objective of finding roles that best explain the direct user-permission assignment. This conflict can be avoided by carefully pre-selecting the

business information. The relevance measure we provide can be used for such a pre-selection.

For the definition of relevance, we first introduce the following quantities. The random variable $X_d \in \{0,1\}$ denotes the assignment of permission d to a generic user. S is the random variable that corresponds to the business attribute of a generic user (e.g. "job code") and let s be one of the actual values that S can take (e.g. "accountant"). Let

$$p(x_d) := 1/N \cdot \sum_i x_{id}$$

be the empirical probability of d being assigned to an unspecified user, and let

$$p(x_d|S = s) := 1/N \cdot \sum_i x_{id} w_{is}$$

be the empirical probability of d being assigned to a user with business attribute s. The natural measure for the information of a random variable A is its entropy $H(A)$ [14], which, in the case of a permission d, is the binary entropy $h(X_d)$. The binary entropy, defined as

$$h(X_d) := -\sum_{x_d \in \{0,1\}} p(x_d) \log_2 \left(p(x_d)\right), \tag{6.1}$$

quantifies the missing information on whether the permission d is granted to some user. The conditional entropy

$$h(X_d|S) := -\sum_{s \in S} p(s) \sum_{x_d \in \{0,1\}} p(x_d|S = s) \log_2 \left(p(x_d|S = s)\right) \tag{6.2}$$

encodes how much of the missing information $h(X_d)$ of X_d remains if S is known. The mutual information

$$I(X_d; S) := h(X_d) - h(X_d|S) \tag{6.3}$$

measures how much the knowledge of S increases the information on X_d. We therefore propose the mutual information $I(X_d; S)$ to measure how much the knowledge of the business information S helps us to predict the assignment x_d of permission d to a generic user.

In order to express this absolute reduction of missing information in a relative way, we define the measure of relevance $\rho_d(S)$ of business information S for permission d to be the relative mutual information ([14], p. 45)

$$\rho_d(S) := \frac{I(X_d; S)}{h(X_d)} = 1 - \frac{h(X_d|S)}{h(X_d)}\,, \tag{6.4}$$

whereas here we use $0/0 := 1$ for the case where $h(X_d) = 0$ ($I(X_d; S)$ will also be 0 then). This number can be interpreted as the fraction of all bits in X_d that are shared with S. Alternatively, $\rho_d(S)$ can be read as the fraction missing information on permission d that is removed by the knowledge of S.

For each kind of business information S that appears potentially useful for role mining, one can now compute $\rho_d(S)$ for all permissions d and examine their distribution (e.g. Fig. 6.2). The larger the overall decrease in entropy of the permissions under the knowledge of the business information S, the better qualified S is as a candidate for hybrid role mining. Given different types of business information that are expected to be helpful for role mining, one can compare them and their combinations according to the proposed measure and pick the most relevant one.

In principle, this relevance analysis can be carried out using any kind of (digitally) available business information. We will give examples in Section 6.4.1.

Limit of few observations per business attribute One should take care to use sufficiently many observations for estimating the relevance $\rho_d(S)$ of a business attribute S. Computing this quantity with too few observations can cause a bias towards high relevance even though the attribute might be completely uninformative for the permission assignments. Imagine the problem of estimating the entropy of a fair coin based on only one observation (being 'heads' without loss of generality). Naïvely computing the empirical probability of heads to be 1 provides an entropy of 0 which differs a lot from the true entropy of a fair coin which is 1. The same effect occurs when one computes the permission entropy conditioned on an irrelevant attribute where only one observation per

attribute value is available. In [59], for instance, the last name of a user was found to be highly relevant!

A practical solution is to compute $\rho_d(S)$ with only those values of S where sufficiently many observations are available. For instance, if more than 10 users have the feature s='Smith', the empirical probability $p(x_d|S = s)$ will give a good estimate of $h(x_d|S = s)$. In our experiments we neglected all attribute values with less than 10 observations.

6.3 Role mining with business attributes

Our goal is to infer user-role and role-permission assignments based on a direct user-permission assignment matrix and additional business information. The basic assumption of role mining is the existence of a role structure underlying the direct assignment. It is this structure that should be discovered by a role mining algorithm. According to Definition 2.6.1, our method searches for the RBAC configuration that is most likely to explain the direct user-permission assignment.

We use the MAC model from Chapter 4 to compute the probabilities of different RBAC configurations. In this section, we show how to combine business information with this model and present an optimization strategy for inferring the parameters of the combined model.

We define the *bottom-up costs* of assigning a given user i to a set of roles $\mathcal{L}$ as the negative logarithm of the likelihood function Eq. (4.11):

$$\begin{aligned} R^{(ll)}_{i,\mathcal{L}} &= -\log\left(\prod_d p_M\left(x_{id} \mid \mathcal{L}, \beta, r, \epsilon\right)\right) \\ &= -\sum_d \log\left(\epsilon \cdot p_U(x_{id}) + (1-\epsilon) \cdot p_S(x_{id}|\mathcal{L}, \beta)\right) . \end{aligned} \tag{6.5}$$

The costs $R^{(ll)}$ for all users are then

$$R^{(ll)} = \sum_{i,\mathcal{L}} z_{i\mathcal{L}} R^{(ll)}_{i,\mathcal{L}} , \tag{6.6}$$

with $z_{i\mathcal{L}} \in \{0, 1\}$ indicating the assignment of user i to the set of roles $\mathcal{L}$.

6.3.1 Combining business information with a likelihood

Optimizing the model parameters with respect to the log-likelihood Eq. (6.6) seeks to find roles and user-role assignments that best explain the given direct-assignment data. Since many different user-role and role-permission assignments can equip the users with their permissions, there are many role configurations with very similar likelihood. Technically speaking, the solution space for a solution with maximum likelihood has many local optima with similar values for the objective function. However, many of these local optima represent RBAC configurations that are unintuitive from a business perspective. A hybrid role mining algorithm that combines the likelihood with business information will find solutions that are more meaningful.

More formally, incorporating business information leads to an optimization problem with two objectives.

1. The RBAC configuration $(\mathbf{Z}, \mathbf{U})$ should accurately approximate a user-permission matrix, both for the current users and for new users.

2. The role assignments should agree with the business information.

These two objectives are weighted and combined to a unified objective function. The weighting allows us to choose the influence of each of the two sub-objectives. Note that if these sub-objectives conflict, the solutions of the joint objective will not be a solution of the single objectives. However, as we will show later, the business meaning of an RBAC configuration can be substantially increased without significantly increasing the bottom-up costs Eq. (6.6). As mentioned above, this behavior is due to the fact that many configurations exist with similarly low bottom-up costs (i.e. similarly high likelihood) but differing degrees of business interpretability.

In the following, we will introduce a cost function $R^{(S)}$ for business information S, reflecting the above assumption. We then define a unified objective function as a linear combination of the business information

costs and the log-likelihood costs Eq. (6.6):

$$R = R^{(ll)}/D + \lambda R^{(S)}\,, \tag{6.7}$$

where $\lambda \geq 0$ is the mixing parameter, weighting the influence of the business information. A weighted linear combination is the easiest way to merge the two cost functions into a single one, and allows a smooth transition from a scenario without business information ($\lambda = 0$) to one that is completely determined by the business information ($\lambda \to \infty$). The term $1/D$ makes the log-likelihood costs independent of the number of permissions D. This enables one to compare with the permission-independent term $R^{(S)}$, which we subsequently give in Eq. (6.8), for arbitrary sized systems.

6.3.2 Objective function for business information

Setting up requirements for an RBAC configuration from the business information perspective is probably the most crucial step in designing a hybrid role-mining technique. Our goal is to make the RBAC configuration as meaningful as possible from the business perspective. This perspective is represented by the business information at hand which could denote, for instance, organizational units or contract types.

Our assumption about the relationship between business information and permissions is as follows: *The business information abstractly describes what users should be able to do.* It plays a role in the generation of the user-permission assignments as we motivated in Section 2.6.1. This assumption implies that two users with the same business attributes will have essentially the same tasks within the company. This assumption, together with the principle of least privileges, which states that users should only have the permissions required for their tasks, therefore implies that users with the same business attributes should have similar permissions. Furthermore, note that only the entire set of roles assigned to a user determines his permissions. Hence, to evaluate if two users of the same business attribute have similar permissions, one must compute a measure of similarity based on their *full* role sets.

Summing up the above considerations, we assume that a role decomposition is meaningful if employees satisfying identical business predicates (i.e., having the same business information attributes) are also assigned to a similar (ideally the same) set of roles.

Given the above considerations, we propose an objective function that compares all pairs of users (i, i') having the business attribute s with respect to their role assignments $(z_{i\cdot},\ z_{i'\cdot})$. Using the Boolean variable $w_{is} \in \{0, 1\}$ to encode whether user i has business attribute s ($w_{is} = 1$) or not ($w_{is} = 0$), the total costs of a role assignment $\mathbf{Z}$ are given as

$$R^{(S)} \;=\; \frac{1}{N} \sum_{s} \sum_{i,i'} w_{is} w_{i's} \sum_{k} z_{i'k} \left(1 - 2 z_{i'k} z_{ik}\right). \tag{6.8}$$

N is the total number of users and $k \in \{1, .., K\}$ is the role index. Each user has a single business attribute s, i.e. $\sum_s w_{is} = 1$, but can be assigned to multiple roles, $1 \leq \sum_k z_{ik} \leq K$. The term $\sum_k z_{i'k} \left(1 - 2 z_{i'k} z_{ik}\right)$ in Eq. (6.8) computes the agreement between the binary assignment vectors $(z_{i\cdot}, z_{i'\cdot})$ for all pairs of users (i, i') having the same attribute s (which is the case iff $w_{is} w_{i's} = 1$). The subterm $(1 - 2 z_{i'k} z_{ik})$ switches the sign of a single term such that agreements $(z_{ik} z_{i'k} = 1)$ are rewarded and differences $(z_{ik} z_{i'k} = 0)$ are penalized.

An alternative to Eq. (6.8) would be to compute the Hamming distance between the two assignment vectors. However, this has the drawback of penalizing pairs with differently sized role sets. We have chosen the dissimilarity function in Eq. (6.8) to avoid such a bias towards equally sized role sets.

For notational convenience, let $N_{sk} := \sum_i z_{ik} w_{is}$ be the number of users that have the business attribute s and are assigned to role k, and let s_i be the attribute of user i. With these auxiliary variables, we simplify the above expression as follows.

$$\begin{aligned} R^{(S)} &= \frac{1}{N} \sum_{s} \sum_{i} w_{is} \sum_{k} \left(N_{sk} - 2 z_{ik} N_{sk}\right) \\ &= \frac{1}{N} \sum_{i} \sum_{k} \left(N_{s_i k} - 2 z_{ik} N_{s_i k}\right) \end{aligned} \tag{6.9}$$

Reorganizing the costs by the role assignments $\mathbf{Z}$ leads to a convenient form:

$$R^{(S)} = \sum_{i,k} (1 - z_{ik}) \frac{N_{s_i k}}{N} - \sum_{i,k} z_{ik} \frac{N_{s_i k}}{N} \ . \tag{6.10}$$

This formulation of the costs is more intuitive: a user i has a business attribute s_i and $N_{s_i k}$ is the number of users with the same attribute that are assigned to role k. User i should be assigned to k if $N_{s_i k}$ is high. The first term in Eq. (6.10) penalizes RBAC configurations *not* assigning i to such roles ($z_{ik} = 0$). The second term rewards solutions with such assignments ($z_{ik} = 1$).

We would like to directly compare this function with the costs $R^{(ll)}_{i,\mathcal{L}}$ of assigning a given user i to a set of roles $\mathcal{L}$. We therefore restate the above expression by substituting z_{ik} by the assignments $z_{i\mathcal{L}}$ from user i to the set of roles $\mathcal{L}$ and the assignments $z_{\mathcal{L}k}$ from role sets to roles. A user gets the roles that are contained in the role set that he is assigned to. Formally, this relation is expressed in the following equation

$$z_{ik} = \sum_{\mathcal{L}} z_{i\mathcal{L}} z_{\mathcal{L}k} \ . \tag{6.11}$$

Using this concept of role sets we can rewrite the top-down costs as

$$\begin{aligned} R^{(S)} &= \sum_{i,k} \left(\left(1 - \sum_{\mathcal{L}} z_{i\mathcal{L}} z_{\mathcal{L}k} \right) \frac{N_{s_i k}}{N} - \sum_{\mathcal{L}} z_{i\mathcal{L}} z_{\mathcal{L}k} \frac{N_{s_i k}}{N} \right) \\ &= \sum_{i,\mathcal{L}} z_{i\mathcal{L}} \left(\sum_{k} \frac{N_{s_i k}}{N} - \sum_{k} z_{\mathcal{L}k} \frac{N_{s_i k}}{N} - \sum_{k} z_{\mathcal{L}k} \frac{N_{s_i k}}{N} \right) \\ &= \sum_{i,\mathcal{L}} z_{i\mathcal{L}} \left(\sum_{k \notin \mathcal{L}} \frac{N_{s_i k}}{N} - \sum_{k \in \mathcal{L}} \frac{N_{s_i k}}{N} \right) \\ &= \sum_{i,\mathcal{L}} z_{i\mathcal{L}} R^{(S)}_{i,\mathcal{L}} \ . \end{aligned} \tag{6.12}$$

In the second line we made use of the fact that a user is only assigned to a single *set* of roles $\mathcal{L}$.

We managed to write the top-down objective function as a sum over users and role sets. In this way we can directly compare it with the log-likelihood costs given by Eq. (6.5).

6.3.3 An annealed inference algorithm

We use the same deterministic annealing scheme for learning the model parameters as in Section 4.3. The free parameters of our model are the same as in the original MAC model. There are the user-role assignments $\mathbf{Z}$, the probabilities of permissions not being assigned to roles $\boldsymbol{\beta}$, and the noise parameters ϵ and r. The only difference is the modified cost function. This cost function depends only on $\mathbf{Z}$ and is independent of the other model parameters. The optimality conditions for the other parameters are therefore not affected by using top-down information and the same update equation as in Section 4.3 can be used. However, the E-step updates of the expectations of the assignments $\mathbf{Z}$ with respect to the Gibbs distribution differ in our modified setting. The costs in these Gibbs distributions are now the combined negative log-likelihood and the top-down costs as defined in Eq. 6.7.

In the following, we explain how to compute the top-down cost function derived above in such an annealing setting. Given an iterative optimization scheme for minimizing an objective function $R^{(S)}$, one faces a computational problem with the above quantities: compute the optimal assignments $z_{i\mathcal{L}}$ from the $N_{s_i k}$, which are, in turn, computed from the $z_{i\mathcal{L}}$ themselves. To make this computation at step t of our algorithm feasible, we use the expected assignments $\gamma_{i\mathcal{L}}^{(t-1)} := \mathbb{E}\left[z_{i\mathcal{L}}^{(t-1)}\right]$ of the previous step instead of the Boolean $z_{i\mathcal{L}}^{(t)}$ to approximate $N_{s_i k}^{(t)}$ by its current expectation with respect to the Gibbs distribution:

$$N_{s_i k}^{(t)} \approx \mathbb{E}\left[N_{s_i k}^{(t-1)}\right] = \sum_{\mathcal{L}} z_{\mathcal{L}k} \sum_{i'} w_{i' s_i} \gamma_{i'\mathcal{L}}^{(t-1)} \,. \tag{6.13}$$

This approximation makes the computation of $R_{i,\mathcal{L}}^{(S)(t)}$ feasible. There-

with, the costs of a user belonging to a set of roles are

$$R_{i,\mathcal{L}}^{(S)(t)} \approx \sum_{k \notin \mathcal{L}} \frac{\mathbb{E}\left[N_{s_i k}^{(t-1)}\right]}{N} - \sum_{k \in \mathcal{L}} \frac{\mathbb{E}\left[N_{s_i k}^{(t-1)}\right]}{N} \quad . \tag{6.14}$$

A description of the algorithm in pseudo-code is given in Algorithm 2.

input : user permission matrix $\mathbf{X}$,
business information S,
parameters λ, T_0, α
output: role assignment matrices $\mathbf{Z}$,
probabilistic role prototypes $\boldsymbol{\beta}$

1: Randomly initialize $\boldsymbol{\beta}$, ϵ, and r
2: $T = T_0$
3: **while** not converged **do**
4: compute $R_{i,\mathcal{L}}$ according to Eq. (6.5) & Eq. (6.14)
5: $c_{i,\mathcal{L}} \leftarrow \exp\left(-R_{i,\mathcal{L}}/T\right)$
6: $\gamma_{i,\mathcal{L}} \leftarrow \frac{c_{i,\mathcal{L}}}{\sum_{\mathcal{L}'} c_{i,\mathcal{L}'}}$
7: Solve Eq. (4.39)–Eq. (4.41) for $\boldsymbol{\beta}$, ϵ, and r, respectively
8: $T \leftarrow \alpha \cdot T$
9: **end while**

Algorithm 2: Probabilistic Hybrid Role Mining

6.4 Experiments

In this section, we report on experimental results on a dataset from an actual enterprise. The dataset contains assignments between 22,352 users and 1,786 permissions. Furthermore we had access to two kinds of business information provided by the company's HR department: each user's organizational unit (OU) and job-code (JC). The organizational unit groups the users based on their division and section within the enterprise. The job-code is a number identifying the kind of employment

contract that the employee has. For example, an employee may be in the division "customer service, overseas" and have job-code 4, indicating that her contract is of the type "head of division". Each employee has a single job-code and a single organizational unit. The enterprise considered has 6,630 OUs and 1,030 JCs.

In the following, we will determine the relevance of these two kinds of business information for role mining using the measures introduced in Section 6.2. Afterwards, we report on experiments that illustrate some of the advantages and drawbacks of hybrid role-mining.

6.4.1 Top-down information analysis

As described earlier, we assume that a user's organizational unit and job-code provide information about his duties and thus his required system permissions. We now test this assumption and measure the information gain using the analytic methods described in Section 6.2.

The top histogram in Figure 6.1 illustrates the distribution of the permission entropy $h(X_d)$ for the direct user-permission assignment. Since the assignment of a permission is either one or zero, the maximum entropy is one bit, which corresponds to a permission that is possessed by exactly half of the users. Permissions possessed by either very few, or almost all, users have low entropy. For the enterprise under consideration, all permissions with low entropy belong to only a few users. To make this distinction clear, in all of the histograms in Figures 6.1 and 6.2, we display counts of permissions shared by less than 2% of the employees in white and the counts of all other permissions in black. As can be seen in the top histogram of Figure 6.1, most of the permissions have low entropy, but a significant number of permissions have very high entropy. The lower three histograms show (in this order) the distribution of the mutual information between permissions and job-codes, organization unit, and the combination of the two.

The results are surprising. Since the job-code provides an abstract high-level job description, one might expect it to be highly relevant. However, the results show that a user's job-code carries only little information about his permissions. The reason is that, in this enterprise,

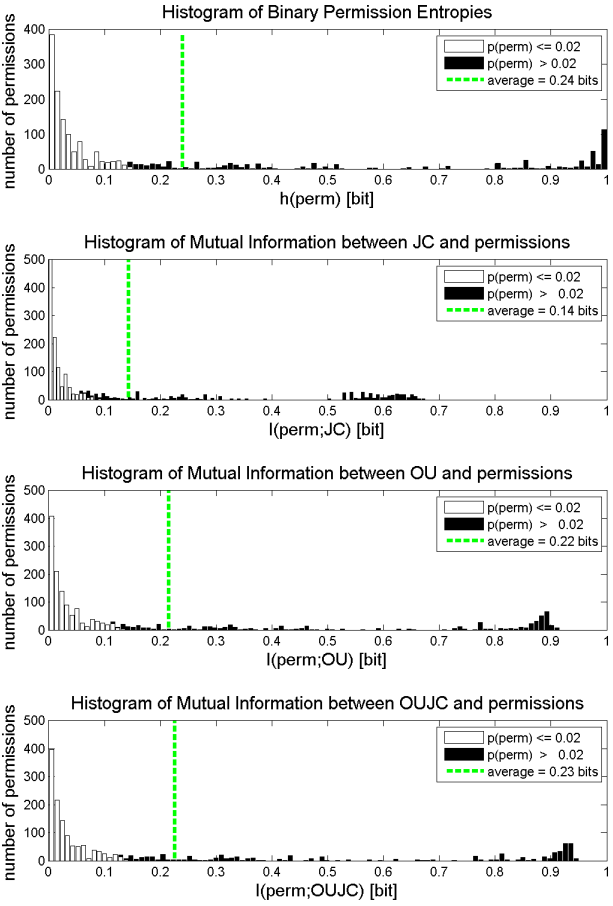


Figure 6.1: Histograms of the mutual information between permissions and different kinds of business information: organization units, jobcodes, and combinations of the two. In the top histogram, the overall entropy of the permissions is shown. The bars for permissions possessed by more than 2% of the users (in black) are stacked on top of the bars for all the other permissions (white).

job-codes are not really abstract *task* descriptions. Instead they express other properties that are interesting for HR, such as the employee's salary class, contract duration, seniority, etc. In contrast, we found that the organization unit is much more relevant for the user's permissions. On average, the organizational unit reduces the entropy by 0.22 bits. Moreover, the entropy of a large number of permissions is even reduced by 0.9 bits (right peak in second lowest histogram of Figure 6.1). Combining the two attributes yields only a slight gain of 0.01 bits on average (see the lower two histograms of Figure 6.1). Hence, we conclude that most of the information gained by using job-codes is already contained in the organizational units.

For the bottom three histograms, note that the bimodal distribution of the permission-entropy histogram is preserved: a high peak at very low entropy and a smaller peak at high entropy. This leads us to a general problem in interpreting mutual information. In many cases, the mutual information $I(X_d; S)$ is low simply because the entropy $h(X_d)$ of the permission d is low. This is the case for permissions that almost all users have (for instance, reading email) or, as is usually the case in this data, permissions that very few users have. In Figure 6.1, we highlighted such permissions in white. This illustrates that almost all permissions whose entropy is not reduced by the knowledge of the given business information have a very low entropy anyway.

To overcome this problem, we weight $I(X_d; S)$ by $1/h(X_d)$ and obtain the relative mutual information $\rho_d(S) = 1 - h(X_d|S)/h(X_d)$ (see Eq. 6.4) as a relevance measure that indicates the *fraction* of entropy that is explained by S (see Fig. 6.2). This relative representation better reveals the difference in information content between organization units and job-codes. Whereas, on average, job codes remove roughly 50% of the uncertainty, knowledge of the organization unit removes 88%. Admittedly, there is no way to really determine if knowledge of S *would* decrease more of the permission entropy if $h(X_d)$ *were* higher. Note that for all permissions with high entropy, the mutual information between business information and permissions is high (compare the white and black bars in the lower three histograms of Fig. 6.1). Therefore, one can reason that knowledge of S might possibly (but not necessarily)

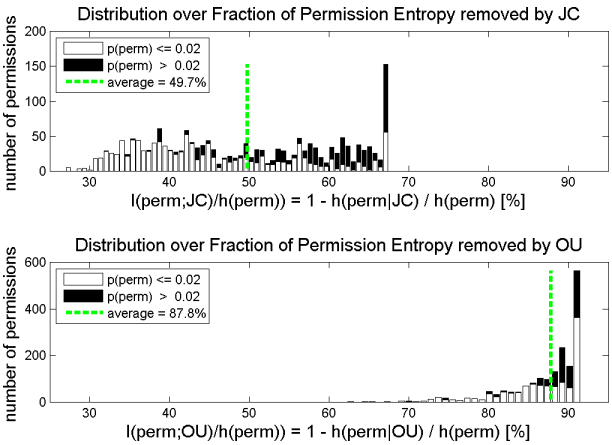


Figure 6.2: Distribution of the measure of relevance Eq. (6.4), the mutual information weighted with inverse permission entropy.

also provide significant absolute information gain on permissions with low-entropy.

Given these findings, we conclude that for the enterprise considered, the organizational unit provides useful top-down information. The information provided by job-codes is already provided by the organizational unit as can be seen from the very small gain in mutual information when both are used together (compare the two lower histograms in Fig. 6.1). Therefore, for the data at hand, it is reasonable to ignore the job-codes and just incorporate the organization unit into the hybrid role-mining process. In the next section, we will report on several experiments with both types of business information.

6.4.2 Experimental results

In this section, we evaluate our algorithm on real-world data. As explained in the introduction, the two criteria of a good hybrid role mining

solution are:

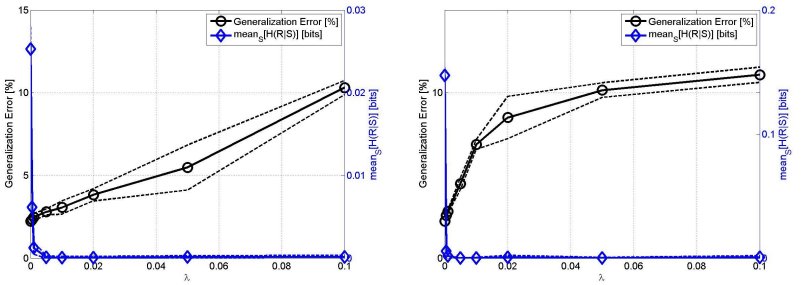
1. the RBAC configuration captures most of the given user-permission assignment matrix without overfitting, and

2. the user-role assignment is easy to interpret from a business perspective.

In order to quantitatively assess a given role mining result, we use two measures: the *generalization ability* and the *interpretability* of an RBAC system. We motivate the generalization ability in Section 2.6.3 and explain how to compute it in Section 4.4.1 and Section 7.3. As follows, we motivate and define the interpretability measure.

Interpretability We formulate the interpretability of the role assignments by the conditional entropy of the role set $\mathcal{L}_i$ of a user i, given his business information s_i, i.e. $h(\mathcal{L}_i|s_i)$. This captures the requirement that all users with the same business attribute should obtain the same set of roles. Thus, the knowledge of the business attribute should, ideally, determine the roles an employee is assigned to. A set of roles, however, might be shared by users with different business attributes. Note that this measure resembles the relevance analysis for business information that we introduced in Section 6.2. There, we required that the given business information should have a high mutual information with the permissions (recall $I(X_d; S) = h(X_d) - h(X_d|S)$) in order to agree with the permission structure (and therefore be useful for role mining). Following the same line of reasoning, we require roles to agree with the business information. RBAC configurations that fulfill this requirement are easy to interpret from the business perspective.

Results We carry out hybrid role mining experiments for two different types of business information. The first experiment uses the organization units as business information, as suggested by our analysis in Section 6.4.1. Our second experiment uses the users' job-codes. In Figure 6.3, we plot the two described measures for both kinds of business



Organization Unit Job Codes

Figure 6.3: Generalization error (black circles) and conditional role entropy (blue diamonds) as a function of the weight of business information λ. Left: hybrid role mining with the organization units of the users. Right: hybrid role mining with job codes. The axes for the generalization error have same scale.

information (OE left, JC right). For the OE-experiment, we directly compare these two measures, as displayed in Figure 6.4.

Figure 6.3 (left) compares, for different factors λ, the two measures on the discovered RBAC configurations. The case with $\lambda = 0$ corresponds to pure bottom-up role mining without business information. While the generalization error slightly increases with λ, the mean conditional role set entropy given the business information decreases drastically if λ is increased from zero to a small value. Hence, the correspondence between the user-role assignment and the users' organizational unit substantially improves as business information is taken into account. Even for small values of λ, the roles can be interpreted as business roles. Since this gain in the business meaning of the user-role assignment comes at the expense of only a small decrease in the generalization ability of the roles, it is a price worth paying. For $\lambda > 0.04$, the entropy $h(\mathcal{L}_i|s_i)$ does not substantially further improve whereas the prediction error rises. In this interval, the two parts of our objective function are antagonistic and hence a further increase of λ is undesirable.

The right part of Figure 6.3 displays the results obtained by using the job-codes as business information. In order to compare the two experiments, Figure 6.3 shows both results for axes with the same scale. Note that the two trends of the conditioned role entropy (blue diamonds) cannot be directly compared since they are computed with respect to the two different types of business information. However, one can reason about the generalization error, which is in both cases computed with respect to the same user-permission assignment. While for the job-codes (right graph) the generalization error converges exponentially with λ to the maximum, it converges only linearly for the organization units. For low λ, it is possible to substantially improve on the interpretability of the role decomposition while preserving good generalization ability.

For hybrid role mining with job-codes this is not possible. The generalization ability increases immediately for even small λ. This result confirms the findings of our analysis of these two types of business information carried out in Section 6.4.1. The job-codes do not agree as well with the direct user-permission assignment as the organizational units do. Hence, using job-codes, it is only possible to trade off generalization ability for business interpretability. However, with organization units, one can improve the business interpretability without substantially increasing the generalization error.

We directly compare the two quality measures with each other for different values of λ in Figure 6.4 for the experiment with the organizational units. The graph shows that it is possible to improve the results of role mining by using our unified objective function to incorporate business information into the role mining process: Changing λ along the straighter parts of the curve gives improved solutions, whereas the more curved parts mark the solutions that are Pareto-optimal with respect to the two measures. In a concrete application, the trade-off between generalization and interpretability must be chosen such that the side conditions are met. For example, one might require that no more than some given percentage of permissions of a new employee are wrongly predicted by the solution. Viewed more generally, optimizing generalization performance and interpretability is a multi-objective optimization problem.

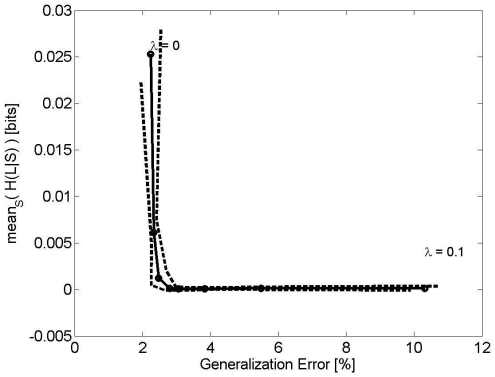


Figure 6.4: Generalization error versus conditional role entropy for the experiment using organization units. The dashed lines represent the standard deviation over ten repeated experiments.

6.5 Related work

Several approaches for top-down role engineering have been previously proposed, all of which are manual. [65] proposed a method to derive roles by analyzing business processes and carried out a case-study on one enterprise as a proof of principle. Similarly, [61] presented an approach based on analyzing business scenarios to find appropriate user-role and role-permission assignments. Both approaches are time consuming for large companies as they require humans to reason about the business processes or scenarios in the enterprise.

Organizational theory is used to define criteria for creating roles in [16]. The criteria are based on a user's position in the enterprise hierarchy, his job function, and the resources he requires for his work. As we demonstrate in this paper, these types of business information can also be used for automated role engineering.

To the best of our knowledge, there are only two other approaches to hybrid role mining. In [57], a candidate set of roles is created using

an algorithm from formal concept analysis [32]. [12] proposes a hybrid approach that extends the bottom-up algorithm proposed in [11]. It is based on an algorithm from [2] for association rule mining.

Both of these approaches work by creating a candidate set of roles based solely on bottom-up data and afterwards using the business information in a post-processing step to select roles in a greedy fashion. In contrast, our algorithm uses business information during the role-creation step. We thereby explore more of the solution space and find solutions that cannot be reached using a preprocessed set of candidate roles. Another difference is that, in contrast to our approach, [57, 12] both lack a probabilistic model whose parameters are to be optimized. The outcome is thus determined by design decisions made for the combinatorial post-processing steps. Moreover, neither approach provides a way to measure the relevance of business information for role mining.

The largest conceptual difference to these two approaches is the objective function. While both approaches demand that users with the same roles have the same business attributes, we demand that users with the same attributes have the same roles. These two objectives are similar on first sight and can be simultaneously satisfied in easy cases. However, in real-world scenarios they usually differ for two reasons.

First, most enterprises have some permissions that are granted to almost all users, such as reading email. Our objective avoids an unnecessarily high number of roles by allowing roles capturing such permissions to be shared among users with different business attributes (e.g. across organizational units). Demanding that users with the same role have the same business attributes would create excessively many roles (e.g. one role for each organizational unit) in such a case.

Second, our objective function differs from the others in terms of role provisioning to new users. We favor solutions where knowledge of the business attributes determines which roles must be assigned, while the other approaches lead to solutions where the roles determine the business attributes. In practice, one usually *knows* the business information of new users and *seeks* the assignment of roles.

6.6 Summary

We have divided the hybrid role mining problem into two parts and provided solutions for them: determining the relevance of business information for role mining, and incorporating this information into a hybrid role mining algorithm. We solved the first problem with an entropy-based measure of relevance. Business information that is relevant for access-control significantly reduces the entropy of user-permission assignments. We approached the second problem by deriving an objective function that combines the MAC model with business information. Our objective function favors RBAC configurations where users with the same business attributes have a similar set of roles.

Experimental results show that our approach finds roles that generalize well and are easy to interpret (i.e., intuitively understandable) from the business perspective. These properties are desirable as they have direct, positive consequences for the administration and maintenance of RBAC-based systems. Generalization facilitates the maintenance of RBAC since new users can be easily equipped with needed permissions without creating new roles. Interpretable roles simplify both the role's life-cycle management and adding new users to the system. We demonstrated that, with a sensible choice of the business attributes, one can drastically improve interpretability at the price of only a small decrease in generalization ability.

Chapter 7

Model selection for unsupervised learning

Clustering and dimensionality reduction are valuable concepts for exploratory data analysis. Both concepts are frequently used in many applications for pattern-recognition, vision, data mining, and other fields. For both concepts one must specify the complexity of solutions. When partitioning a set of objects into clusters, we must select an appropriate number of clusters. Learning a low-dimensional representation of a set of objects, for example by learning a dictionary, involves choosing the number of atoms or codewords in the dictionary. More generally speaking, solving a specific task on some given data with a given set of models requires selecting the model that solves the task best.

The next section serves as a motivation for the model selection problem. We address the task of denoising a given binary matrix by a rounded rank-k approximation of that matrix computed with truncated singular-value decomposition (SVD). We will see that the choice of the SVD model, parameterized by the cut-off rank k, considerably influences the deviation of the denoised matrix from the noise-free ground truth matrix. In the subsequent sections we then propose methods for selecting the appropriate model for a given task. In Section 7.2 we propose a

eigenvalue-based heuristic specifically tailored to the motivating example of rank selection. In Section 7.3 we propose the concept of minimal transfer costs (MTC) for model-order selection in unsupervised learning problems. We demonstrate this technique on a number of different applications. Finally, we describe the theory of approximation set coding in Section 7.4.3. This theory provides a method to model-order selection and model selection in general. We investigate this method on the task to select between different noise models for role mining in Section 7.4.5 and then, in Section 7.4.6, we study it to the rank selection task finally coming back to our motivating application.

7.1 Error-detection as a preprocessing step for role mining

As follows, we describe a SVD-based denoising method for binary matrices. In the context of role mining, it was originally proposed by Molloy et al. in [59]. Let $\mathbf{X}$ be a binary matrix. Molloy et al. compute the SVD decomposition $\mathbf{X} = \mathbf{USV^T}$ and set all but the first k values on the diagonal of $\mathbf{S}$ to 0. This provides a rank-k approximation $\mathbf{U_k S_k V_k^T}$ of $\mathbf{X}$. Then, a function g maps all individual entries higher than 0.5 to 1 and the others to 0.

We demonstrate this procedure on a synthetically generated binary matrix depicted in Figure 7.1, left. The noise-free matrix $\mathbf{X}^*$ is generated out of disjunctions of five Boolean vectors leading to a specific structure. 30% of the matrix entries are replaced by random bits generated by a fair (computational) coin leading to the noisy matrix $\mathbf{X}$. Figure 7.1 shows the rank $k = 5$ approximation of $\mathbf{X}$ in the middle and the rounded output of the procedure on the right. It can be seen that a substantial part of the irregularities of the matrix has been removed while the structure is almost perfectly preserved.

The distance of the resulting denoised matrix $\tilde{\mathbf{x}}_\mathbf{k} = g(\mathbf{U_k S_k V_k^T})$ to the error-free matrix $\mathbf{X}^*$ depends heavily on k. We illustrate the trend of this deviation in Figure 7.2. This curve is computed based on the example matrix depicted in Figure 7.1. In this example, the optimal

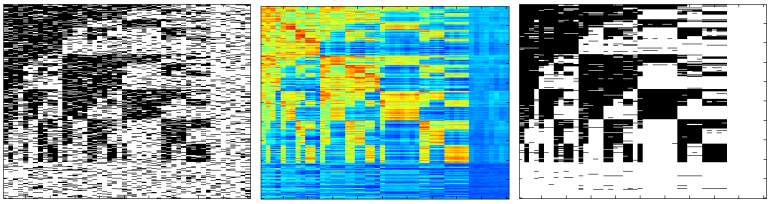


Figure 7.1: Denoising a binary matrix via truncated SVD. From left to right: Original matrix ordered to visualize structure. The rank-k reconstruction with $k = 5$. The reconstruction rounded to $\{0, 1\}$. For this example, $k = 5$ achieves the lowest denoising error.

continuous rank equals the Boolean rank and is $k = 5$. For lower ranks the decomposition fails to capture the structure in the data matrix: it 'underfits'. When k is larger than 5, the decomposition perfectly reconstructs the structure but also 'overfits' to some of the noisy bits.

The reason why, in this example, not all noisy bits are detected for the optimal rank $k = 5$ and why some of the fine structures of the matrix are blurred is due to a model-mismatch. The data has been generated by a Boolean disjunction of source vectors, a highly non-linear process. In contrast, SVD fits the data with a linear combination of continuous values. It can reconstruct the original values by, for instance, subtracting several small continuous factors from a too large factor. Cutting off the contribution of some of such factors sometimes leads to wrongly reconstructed bits. With a generative model like MAC that finds a Boolean decomposition of the data one could, in principle, exactly reconstruct the noise-free matrix. However, we have seen in the previous chapters that Boolean matrix factorization is a hard problem and the optimization of a probabilistic model like MAC is somewhat involved. Therefore, the advantage of a continuous decomposition like SVD is its easy computation.

In this context, it makes sense to distinguish between two tasks: i) finding the irregular bits of a binary matrix and ii) finding a Boolean decomposition of the matrix structure. Task ii) is much harder than

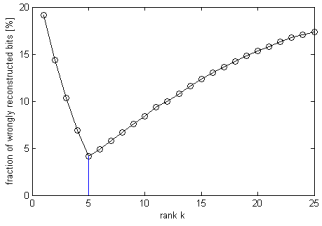


Figure 7.2: Denoising error as a function of the cutoff-rank k of truncated SVD. The example with optimal rank $k = 5$ is depicted in Figure 7.1. For lower ranks, the structure is insufficiently reconstructed (underfitting) and for higher ranks, the decomposition fits to noisy bits (overfitting).

task i) and provides the solution of task i) as a byproduct. Continuous matrix factorizations like SVD solve only task i). This can be seen by trying to obtain the Boolean decomposition by rounding the continuous decomposition $\mathbf{U_k S_k V_k^T}$. It has been observed, for instance in [54], that the resulting reconstruction has nothing in common with the original matrix $\mathbf{X}$ or with the noise-free matrix.

In some applications it makes sense to solve only task i). For instance, as many role mining methods are very sensitive to noise [26], they could benefit a lot from denoising as a preprocessing step. Taking the influence of the rank into account, this task reduces to a model-order selection problem. As in practice one does not know the noise-free ground truth matrix $\mathbf{X}^*$, one cannot select the rank based on a graph like Figure 7.2. Instead, one must find another measure that is computable based on the input matrix $\mathbf{X}$. In the subsequent sections we will propose and investigate different methods for selecting the rank for continuous matrix factorization. Thereby, we will also develop methods that go beyond that task and that can be used for other model selection problems as well. The problem of denoising a Boolean matrix only served to motivate the model-selection problem here. Moreover, it will constitute a running example in the experimental investigation of any model selection method proposed in the subsequent sections.

7.2 A spectrum-based heuristic

We propose a simple heuristic for selecting the cut-off rank of truncated SVD or for selecting the number of principal components in PCA. Our heuristic is based on the following observations. For most matrices, the low eigenvalues account for the irregularities of the matrix and the large eigenvalues account for its predominant structure. As a consequence, a sorted eigenvalue spectrum has a rank k separating the structure eigenvalues from the noise eigenvalues. We further observe that, often, the lower end of the spectrum decreases linearly. Moreover, usually there is a discontinuity at eigenvalue k, either in the spectrum itself or in its first or its second derivation.

Our method exploits the linear trend at the lower end of the spectrum to compute a linear fit and the mean of the residuals. The fit, computed on the lowest $q_1 \in [0, 50]$ percent of the eigenvalues, extrapolates to those larger eigenvalues that also account for noise. These eigenvalues should be in the proximity of the linear fit with a deviation that is similar to the average residual. We select the lowest eigenvalue where the spectrum deviates from the fit with more than $q_2 \in \mathbb{R}_+$ times the average residual. This detector operates on the spectrum and on its first and second derivative, thus proposing three cutoff ranks. Then it selects the median of these three ranks. See Figure 7.3 for an example of the rank selection. The three plots depict the eigenvalue spectrum and its derivatives.

Obviously, there this method has weaknesses. First of all, we replaced the selection of one parameter k by the selection of two other parameters q_1 and q_2. In terms of the number of free parameters, we made the model selection problem even more difficult. However, the sensitivity of the denoising error with respect to k is high as can be seen in Figure 7.2. It is more robust with respect to q_1 and q_2 because for a wide range of these parameters, the heuristic selects the same cut-off rank 'k'. The second point of criticism adresses the assumptions underlying the heuristic. The observed properties described above do not hold for all matrices. Therefore, we will observe that our method sometimes fails to find the optimal rank. In later sections we will describe more elaborate

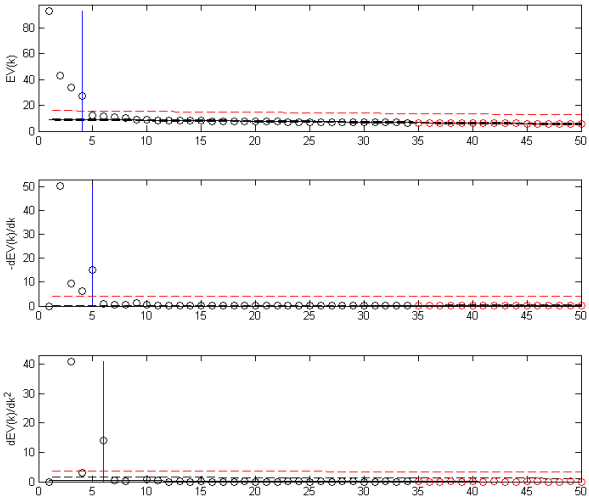


Figure 7.3: Heuristic for detecting the optimal cut-off rank from the singular value spectrum. The top plot depicts the trend of the sorted eigenvalues. The heuristic fits a line to the lowest 30% of the eigenvalues and selects the rank where the eigenvalue substantially deviates from this linear extrapolation (here $k = 4$). Middle: the same procedure on the first derivation of the spectrum and bottom: for the second derivation. The heuristic selects the median rank of the three proposed ranks.

methods. The heuristic proposed in this section will then serve as a baseline method.

7.3 Minimum transfer costs

In this section we deviate a bit from the running example of denoising a binary matrix and address the more general problem of model-order selection for unsupervised learning problems. We develop and advocate the principle of *minimal transfer costs* (MTC). Our method generalizes

classical cross-validation known from supervised learning. It is applicable to a broad class of model-order selection problems even when no labels or target values are given. The MTC principle can be easily explained in abstract terms: A good choice of the model-order based on a given dataset should also yield low costs on a second dataset from the same distribution. We learn models of various model-orders from a given dataset $\mathbf{X}^{(1)}$. These models with their respective parameters are then used to interpret a second data set $\mathbf{X}^{(2)}$, i.e., to compute its costs. The principle selects the model-order that achieves lowest transfer cost, i.e. the solution that generalizes best to the second dataset. Too simple models underfit and achieve high costs on both datasets; too complex models overfit to the fluctuations of $\mathbf{X}^{(1)}$ which results in high costs on $\mathbf{X}^{(2)}$ where the fluctuations are different.

The challenging part of this procedure is related to the transfer of the solution inferred from the objects of the first dataset to the objects of the second dataset. This transfer requires a mapping function which generalizes the conceptually straightforward assignments in supervised learning.

For several applications, we demonstrate how to map two datasets to each other when no labels are given. We select well-known methods such as maximum-likelihood inference, k-means and Gaussian mixture models because the understandability of our principle should not be limited by long explanations of the complicated models it is applied to. Also, we come back to the problem of denoising a binary matrix with SVD. In our real-world applications *image denoising*, *role mining*, and *error detection in access-control configurations*, we pursue the goal to investigate the reliability of the model order selection scheme, i.e. whether for a predetermined method (such as SVD), our principle finds the model-order that performs best on a second test data set.

In the following subsections we first explain the principle of minimal transfer costs and we address the conceptual question of how to map a trained model to a previously unseen dataset. In Sections 7.3.2, 7.3.5, and 7.3.6, we invoke the MTC principle to select a plausible ("true") number of centroids for the widely used Gaussian mixture model and the optimal number of clusters for the k-means algorithm. In Section 7.3.3,

we apply MTC to SVD for image denoising and detecting errors in access-control configurations. In Section 7.3.4, we use MTC for selecting the number of factors for Boolean matrix factorization on role mining data.

7.3.1 The minimum transfer costs principle

Notational Preliminaries. Let O be a set of N objects with corresponding measurements. The measurements can be characterized in several ways: (i) objects can be identified with the measurements and we can use the terms synonymously, i.e., the i^{th} object is described by the vector $\mathbf{X}_i$; (ii) measurements are pairwise (dis)similarities between objects. In the first case, the objects O are directly characterized by the measurements $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathcal{X}$. In the second case, a graph $\mathcal{G}(O, \mathbf{X})$ with (dis)similarity measurements $\mathbf{X} := \{x_{ij}\}$ characterizes the relations for all pairs of objects (i, j), $1 \leq i \leq N, 1 \leq j \leq N$. Furthermore, let $\{O^{(1)}, \mathbf{X}^{(1)}\}$ and $\{O^{(2)}, \mathbf{X}^{(2)}\}$ be two datasets given from the same source. Often in practical situations, only one such dataset is available. Then we randomly partition it into $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$.

A data model is usually characterized as an optimization problem with an associated *cost function*. We denote the potential outcome of optimizing a cost function by the term *solution*. A cost function $R(\boldsymbol{s}, \mathbf{X}, k)$ quantifies how well a particular solution $\boldsymbol{s} \in \mathcal{S}$ explains the measurements $\mathbf{X}$. For parametric models, the solution includes a set of model parameters which are learned through an inference procedure. The number k quantifies the number of model parameters and thereby identifies the model order. In clustering, for instance, k would be the number of clusters of the solution $\boldsymbol{s}(\mathbf{X})$.

Minimum Transfer Costs A cost function imposes a partial order on all possible solutions given the data. Since usually the measurements are contaminated by noise, one aims at finding solutions that are robust against the noise fluctuations and thus generalize well to future data. Learning theory demands that a well-regularized model explains not only the dataset at hand, but also new datasets generated from the same source and thus drawn from the same probability distribution.

Let $\boldsymbol{s}^{(1)}$ be the solution (e.g. model parameters) learned from a given set of objects $O^{(1)} = \{i : 1 \leq i \leq N_1\}$ and the corresponding measurements $\mathbf{X}^{(1)}$. Let the set $O^{(2)} = \{i' : 1 \leq i' \leq N_2\}$ represent the objects of a second dataset $\mathbf{X}^{(2)}$ drawn from the same distribution as $\mathbf{X}^{(1)}$. In a supervised learning scenario, the given class labels of both datasets guide a natural and straightforward mapping of the trained solution from the first to the second dataset: the model should assign objects of both sets with the same labels to the same classes. However, when no labels are available, it is unclear how to transfer a solution. To enable the use of cross-validation, we propose to compute the costs of a learned solution on a new dataset in the following way. We start with defining a mapping ψ from objects of the second dataset to objects of the first dataset:

$$\psi : \; O^{(2)} \times \mathcal{X} \times \mathcal{X} \;\; \rightarrow \;\; O^{(1)} \tag{7.1}$$

$$\left(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}\right) \;\; \mapsto \;\; \psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \tag{7.2}$$

This mapping function aligns each object from the second dataset with its nearest neighbor in $O^{(1)}$. We have to compute such a mapping in order to transfer a solution. Let's assume, for the moment, that the given model is a sum over independent partial costs

$$R(\boldsymbol{s}, \mathbf{X}, k) = \sum_{i=1}^{N} R_i(\boldsymbol{s}(i), \mathbf{x}_i, k). \tag{7.3}$$

$R_i(\boldsymbol{s}(i), \mathbf{x}_i, k)$ denotes the partial costs of object i and $s(i)$ denotes the structure part of the solution that relates to object i. For a parametric centroid-based clustering model $s(i)$ would be the centroid to which object i is assigned to. Using the object-wise mapping function ψ to map objects $i' \in O^{(2)}$ to objects in $O^{(1)}$, we define the **transfer costs** $R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k)$ of a solution $\boldsymbol{s}$ with model-order k as follows:

$$R^T(\boldsymbol{s}^{(1)}, \mathbf{x}^{(2)}, k) \; := \; \frac{1}{N_2} \sum_{i'=1}^{N_2} \sum_{i=1}^{N_1} R_{i'}(\boldsymbol{s}^{(1)}(i), \mathbf{x}_{i'}^{(2)}, k) \; \mathbf{I}_{\{\psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = i\}} \tag{7.4}$$

For each object $i' \in O^{(2)}$ we compute the costs of i' with respect to the learned solution $\boldsymbol{s}(\mathbf{X}^{(1)})$. The mapping function $\psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)})$ ensures

that the cost function treats the measurement $\mathbf{x}_{i'}^{(2)}$ with $i' \in O^{(2)}$ as if it was the object $i \equiv \psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \in O^{(1)}$. In the limit of many observations N_2, the transfer costs converge to $\mathbb{E}[R(\boldsymbol{s}^{(1)}, \mathbf{X}, k)]$, the expected costs of the solution $\boldsymbol{s}^{(1)}$ with respect to the probability distribution of the measurements. Minimizing this quantity with respect to the solution is what we are ultimately interested in. The minimum transfer cost principle (MTC) selects the model-order k with lowest transfer costs. MTC disqualifies models with a too high complexity that perfectly explain $\mathbf{X}^{(1)}$ but fail to fit $\mathbf{X}^{(2)}$ (overfitting), as well as models with too low complexity which insufficiently explain both of them (underfitting).

We would like to emphasize the relation of our method to cross-validation in supervised learning which is frequently used in classification or regression. In supervised learning a model is trained on a set of given observations $\mathbf{X}^{(1)}$ and labels (or output variables) $\mathbf{y}^{(1)}$. Usually, we assume i.i.d. training and test data in classification and, therefore, the transfer problem disappears.

A variant and a special case of the mapping function: In the following, we will describe two other mapping variants. In many problems such as clustering, a solution is a set of structures where the objects inside a structure are statistically indistinguishable by the algorithm. Therefore, the objects $O^{(2)}$ can directly be mapped to the structures inferred from $\mathbf{X}^{(1)}$ rather than to individual objects, since the objects in each structure are unidentifiable. In this way, the mapping function assigns the objects $O^{(2)}$ to the solution $\boldsymbol{s}(\mathbf{X}^{(1)}) \in \mathcal{S}$:

$$\begin{aligned} \psi^s : \; O^{(2)} \times \mathcal{S} \times \mathcal{X} \quad &\rightarrow \quad S(O^{(1)}) \; , \\ \left(i', \boldsymbol{s}(\mathbf{X}^{(1)}), \mathbf{X}^{(2)}\right) \quad &\mapsto \quad \psi\left(i', \boldsymbol{s}(\mathbf{X}^{(1)}), \mathbf{X}^{(2)}\right) \; . \end{aligned} \tag{7.5}$$

The *generative* mapping, another variant of the ψ function, is obtained in a natural way by data construction. Given the true model parameters, we randomly sample pairs of data items. This gives the identity mapping between the pairs in $O^{(1)}$ and $O^{(2)}$ and can be used

whenever the data is artificially generated.

$$\begin{aligned} \psi^G : \; O^{(2)} &\rightarrow O^{(1)} \,, \\ i' &\mapsto \psi(i') = i \end{aligned} \tag{7.6}$$

In practice, however, the data is usually generated in an unknown way. One has a single dataset $\mathbf{X}$ and subdivides it (eventually multiple times) into random subsets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$ which are not necessarily of equal cardinality. The *nearest-neighbor* mapping is obtained by assigning each object $i' \in O^{(2)}$ to the structure or object where the costs of $R(\boldsymbol{s}, \mathbf{X}(O^{(1)} \cup i'), k)$ is minimized. In the cases where multiple objects or structures satisfy this condition, i' is randomly assigned to one of them.

7.3.2 The easy case: density estimation in metric spaces

We start with mixtures of Gaussians (GMM). We will see that for this model, the transfer of the learned solution to a second dataset is straightforward and requires no particular mapping function. This case is still a good example to start with as it demonstrates that cross-validation for unsupervised learning is a powerful technique that can compete with well known model-selection scores such as BIC and AIC.

A GMM solution consist of the centers $\boldsymbol{\mu}_t$ and the covariances $\boldsymbol{\Sigma}_t$ of the Gaussians, as well as the mixing coefficients π_t. The model order is the number of Gaussians k and the cost function is the negative log likelihood of the model

$$R\left((\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathbf{X}, k\right) = -\sum_{i=1}^{N} \ln \left(\sum_{t=1}^{k} \pi_t \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \right) \tag{7.7}$$

As all model parameters are independent of the object index i, it is straightforward to compute the transfer costs on a second dataset. The learned model parameters provide a probability density estimate for the entire measurement space such that the individual likelihood of each new data item can be readily computed. The transfer costs are $R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k) = R(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \mathbf{X}^{(2)}, k)$

We carry out experiments by generating 500 items from three Gaussians. As we increase their variances to increase their overlap, we learn GMMs with varying number of Gaussians k and compute the BIC score, the AIC score, as well as the transfer costs. Two exemplary results are illustrated in Figure 7.4: an easy setting with small overlap of the Gaussians in the upper row and a difficult setting with high overlap in the lower row. In the easy case, each of the four methods selects the correct number of clusters. For increasing overlap, AIC exhibits a tendency to select a too high number of components. At the variance depicted in the lower plots, BIC starts selecting $k < 3$, while MTC still estimates 3 Gaussians. For very high overlap, we observe that both BIC and MTC select $k = 1$ while AIC selects the maximum number of Gaussians that we offered. The interval of the standard deviation where BIC selects a lower number of Gaussians than MTC ranges from 60% of the distance between the centers (illustrated in Figure 7.4 bottom) to 85%. The reason for this discrepancy has to be theoretically explored. As, theoretically, the BIC score is exact in the asymptotic limit of many observations, this discrepancy between MTC and BIC might indicate that MTC is less accurate. However, maybe BIC underfits due to non-asymptotic corrections. We do not have more theoretic insight at the moment. Visual inspection of the data suggests that model selection in this discrepancy regime is non-trivial.

7.3.3 Model order selection for truncated SVD

Denoising access-control matrices In this section, we come back to the running example of denoising a binary matrix with a continuous factorization like SVD. The application domain is the detection of exceptional user-permission assignments in a access-control matrix $\mathbf{X}$.

In [59], SVD and other continuous factorization techniques for denoising $\mathbf{X}$ are proposed. Molloy et al. compute a rank-k approximation $\mathbf{U_k S_k V_k^T}$ of $\mathbf{X}$. Then, a function g maps all individual entries higher than 0.5 to 1 and the others to 0. As we saw in Section 7.1, the Hamming distance of the resulting denoised matrix $\tilde{\mathbf{X}}_\mathbf{k} = g(\mathbf{U_k S_k V_k^T})$ to the error-free matrix $\mathbf{X}^*$ depends heavily on k. The authors propose two

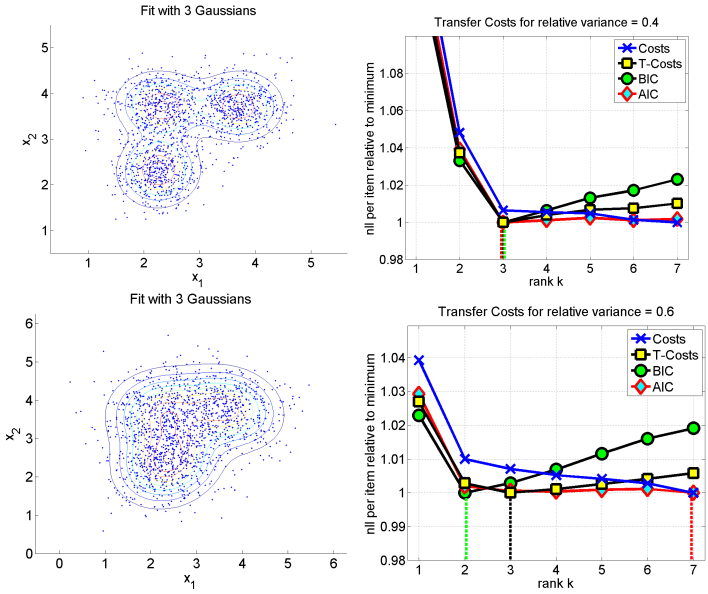


Figure 7.4: Selecting the number of Gaussians k. Data is generated from 3 Gaussians. Going from the upper to the lower row, their overlap is increased. For very high overlap, BIC and MTC select $k = 1$. The lower row illustrates the smallest overlap where BIC selects $k < 3$.

methods for selecting the rank k. The first method takes the minimal rank such that the approximation $\tilde{\mathbf{X}}_\mathbf{k}$ covers 80% of the entries of $\mathbf{X}$ (this heuristic originates from the rule of thumb that 20% of the entries of $\mathbf{X}$ are corrupted). The second method selects the smallest rank that decreases the approximation increment $||(\tilde{\mathbf{X}}_\mathbf{k} - \tilde{\mathbf{X}}_{\mathbf{k+1}})||_1 \Big/ ||\mathbf{X}||_1$ below the *increment-threshold* 0.001.

We also compare with the rank selected by the Bi-crossvalidation method for SVD presented by Owen and Perry [62]. This method, which we will term OP-CV, divides the $N \times D$ input matrix $\mathbf{X}_{1:N,1:D}$ into four submatrices, $\mathbf{X}_{1:p,1:q}$, $\mathbf{X}_{1:p,q+1:D}$, $\mathbf{X}_{p+1:N,1:q}$, and $\mathbf{X}_{p+1:N,q+1:D}$ with

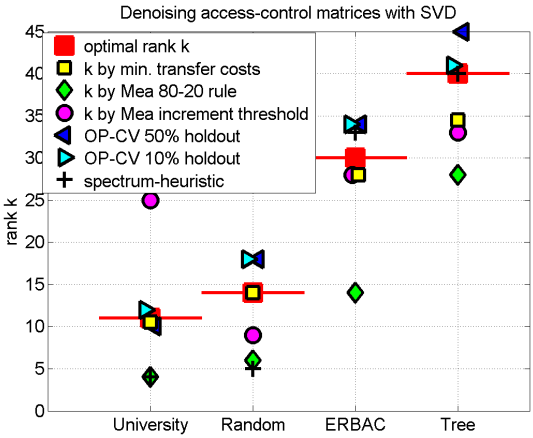


Figure 7.5: Denoising four different access-control configurations via rank-limited SVD. The ranks selected by transfer costs and OP-CV are significantly closer to the optimal rank than the ranks selected by the originally proposed methods [Molloy et al., 2010].

$p < N$ and $q < D$. Let $\mathbf{M}^\dagger$ be the Moore-Penrose inverse of the matrix $\mathbf{M}$. OP-CV learns the truncated SVD $\hat{\mathbf{X}}^{(k)}_{p+1:N,q+1:D}$ from $\mathbf{X}_{p+1:N,q+1:D}$ and computes the error score

$$\epsilon = \mathbf{X}_{1:p,1:q} - \mathbf{X}_{1:p,q+1:D} \; (\hat{\mathbf{X}}^{(k)}_{p+1:N,q+1:D})^\dagger \; \mathbf{X}_{p+1:N,1:q} \; .$$

In our experiments, we compute ϵ for 20 permutations of the input matrix and select the rank with lowest median error.

We compare the rank selected by the described approaches to the rank selected by MTC with nearest-neighbor mapping and Hamming distance. The four different datasets are taken from [59]. The first dataset 'University' is the access control configuration of a department, the other three are artificially created, each with differing generation processes as described in [59]. The sizes of the datasets are (users$\times$permissions) 'University': 493×56, 'Random': 500×347, 'ERBAC': 500×101, and 'Tree': 500×190. We display the results in Figure 7.5.

The rank that achieves the closest reconstruction to the noise-free matrix is plotted as a big red square. This is the optimal rank for denoising. The statistics of the rank selected by MTC is plotted as small bounded squares. We select the median over 20 random splits of the dataset. As one can see, the minimum transfer cost rank is always significantly closer to the optimal rank than the ranks selected by the originally proposed methods. The performance of the 80-20 rule is very poor and performance of the increment threshold depends a lot on the dataset. The Bi-crossvalidation method by Owen and Perry (OP-CV) finds good ranks, although not so reliably as MTC. It has been reported that, for smaller validation sets, OP-CV tends to overfit [62]. We could observe this effect in some of our experiments and also on the University dataset. However, on the 'Tree' dataset it is actually the method with the larger validation set that overfits. Our spectrum-based heuristic proposed in Section 7.2 is very unrelyable on these datasets with sometimes large deviations from the optimal rank. Interestingly, it selects the optimal rank $k = 40$ on the 'Tree' dataset which is the hardest dataset for all other methods.

Image denoising Truncated SVD provides a powerful, yet simple method of denoising images. Given a noisy image, one extracts small $n \times m$ patches from the image (usually squares $m = n$) and computes a rank-limited SVD on the matrix $\mathbf{X}$ containing the ensemble of all patches, i.e. the pixel values of one patch are one row in $\mathbf{X}$. SVD provides a dictionary that describes the image content on a local level. Restricting the rank of the decomposition, the image content is approximated and, hopefully, denoised. SVD has been frequently applied to image denoising in the described way or as part of more sophisticated methods (e.g. [19]). Thereby, selecting the rank of the decomposition poses a crucial modeling choice. In [19], for instance, the rank is selected by experience of the authors and the issue of automatic selection is shifted to further research. Here, we address this specific part of the problem. The task is to select the rank of the SVD decomposition such that the denoised image is closest to the noise-free image. Please note that our goal is not primarily to achieve the very best denoising error

given an image (clearly, better image denoising techniques than SVD exist). Therefore, we do not optimize on other parameters such as the size of the patches. The main goal is to demonstrate that MTC selects the optimal rank for a defined task, such as image denoising, conditioned on a predefined method.

We extract $N = 4096$ patches of size $D = 8 \times 8$ from the image and arrange each of them in one row of a matrix $\mathbf{X}$. We randomly split this matrix along the rows into two sub-matrices $\mathbf{X^{(1)}}$ and $\mathbf{X^{(2)}}$ and select the rank k that minimizes the transfer costs

$$R^T(\boldsymbol{s}, \mathbf{x}, k) = \frac{1}{N_2} \left\| \psi_{NN}(\mathbf{X^{(1)}}, \mathbf{X^{(2)}}) \circ \mathbf{X^{(2)}} - \left(\mathbf{U_k^{(1)}} \mathbf{S_k^{(1)}} \mathbf{V_k^{(1)T}} \right) \right\|_2^2 . \tag{7.8}$$

The mapping $\psi_{NN}(\mathbf{X^{(1)}}, \mathbf{X^{(2)}})$ reindexes all objects of the test set with the indices of their nearest neighbors in the training set.

We illustrate the results for the Lenna image in Figure 7.6 by color-coding the peak-SNR of the image reconstruction. As one can see, there is a crest ranging from a low standard deviation of the added Gaussian noise and maximal rank ($k = 64$) down to the region with high noise and low optimal rank ($k = 1$). The top of the crest marks the optimal rank for given noise (dashed magenta line). The rank selected by MTC is highlighted by the solid black line (dashed black lines are *three times* the standard deviation for improved visibility). The selected rank is always very close to the optimum. At low noise levels where the crest is rather broad, the deviation from the optimum is maximal. There the selection problem is most difficult. However, in this parameter range the choice of the rank has little influence on the error. For high noise, where a deviation from the optimum has higher influence, our method finds the optimal rank.

7.3.4 MTC for Boolean matrix factorization

In this section, we use MTC to select the number of factors in Boolean matrix factorization. As in the last section, the mapping function uses the nearest-neighbor rule with Hamming metric. Here, the MTC score in (Eq. 7.4) measures the number of bits in $\mathbf{x}_{i'*}^{(2)}$ that do not match the

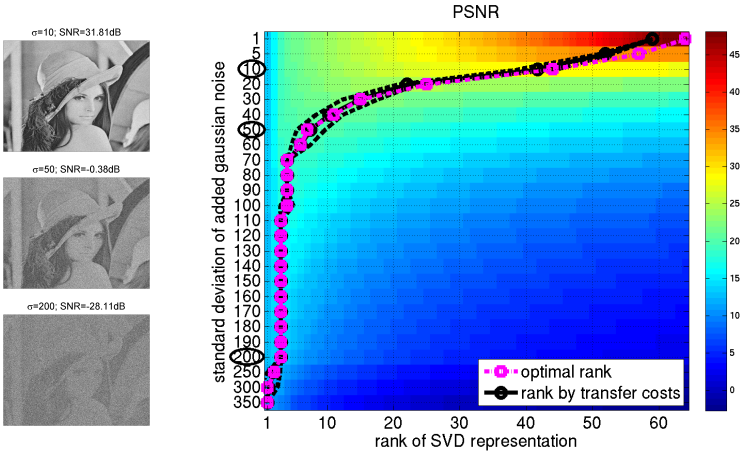


Figure 7.6: PSNR (logarithmic) of a denoised image as a function of the added noise and the rank of the SVD approximation of the image patches. The crest of this error marks the optimal rank at a given noise level and is highlighted (dashed magenta). The rank selected by MTC (solid black) is close to this optimum.

learned decomposition $\boldsymbol{s}^{(1)} = \left(\mathbf{Z}^{(1)}, \mathbf{U}^{(1)}\right)$:

$$R_{i'}(\boldsymbol{s}^{(1)}(i), \mathbf{x}^{(2)}_{i'*}, k) = \sum_j \left| x^{(2)}_{i'j} - \bigvee_{t=1}^{k} (z^{(1)}_{it} \wedge u^{(1)}_{tj}) \right| . \tag{7.9}$$

As in the SVD experiments, the total transfer costs are

$$R^T(\boldsymbol{s}^{(1)}, \mathbf{x}^{(2)}, k) = \frac{1}{N_2} \sum_{i'=1}^{N_2} \sum_{i=1}^{N_1} R_{i'}(\boldsymbol{s}^{(1)}(i), \mathbf{x}^{(2)}_{i'}, k)\ \mathbf{I}_{\{\psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = i\}} .$$

This experiment differs from the previous MTC experiments where the cost function used for optimization on the training data equals the cost function used for the MTC score. This time, the learning algorithm maximizes a particular likelihood Eq. (4.11) to infer the decomposition

$\boldsymbol{s}^{(1)}$, while we cross-validate on the score $R_{i'}(\boldsymbol{s}^{(1)}(i), \mathbf{x}^{(2)}_{i'*}, k)$. In principle, MTC can apply any desired cost function to the hold out dataset. Here this cost function is the Hamming distance of the solution to the hold-out validation data.

Prediction Error vs. Transfer Costs We found that our first approach of computing the generalization error proposed in [71, 26] can be insensitive to overfitting. We denote this older measure of the generalization error by 'prediction error'. As follows, we briefly describe the prediction error and explain where the overfitting problem stems from. We contrast the transfer costs to the prediction error by cross-validating on the real-world dataset used in the experiments in Section 4.4.3.

The computation of the prediction error works as follows. There are, again, two random subsets of the dataset, $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. We train a solution $\boldsymbol{s}^{(1)} = \left(\mathbf{Z}^{(1)}, \mathbf{U}^{(1)}\right)$ on the first dataset $\mathbf{X}^{(1)}$. Let $\mathcal{D}^d$ be a set of randomly chosen dimensions of the data and $\kappa = \left|\mathcal{D}^d\right| / D$ is the cardinality of this set relative to the total number of dimensions D. We report the entries of $\mathbf{X}^{(2)}$ of all dimensions $d \in \mathcal{D}^d$ to the algorithm and hide the entries of the remaining dimensions $\mathcal{D}^u := \{1, \ldots, D\} \setminus \mathcal{D}^d$.

The binary matrix $\mathbf{Z}^{\mathbb{L}}$ encodes the set of possible assignment sets, where each row corresponds to an assignment set. Then, $\hat{\mathbf{U}}^{\mathbb{L}} := \mathbf{Z}^{\mathbb{L}} \otimes \hat{\mathbf{U}}$ encodes all combinations of the estimated sources. For each object i', we choose the assignment set $\hat{\mathcal{L}}_{i'}$ such that the combined source $\hat{u}^{\mathbb{L}}_{\hat{\mathcal{L}}*} = \bigvee_{k \in \hat{\mathcal{L}}_{i'}} u_{k*}$ minimizes the Hamming distance to the disclosed dimensions $\mathcal{D}^d$ of object i':

$$\hat{\mathcal{L}}_{i'} := \arg\min_{\mathcal{L} \in \mathbb{L}} \left\{ \sum_{d \in \mathcal{D}^d} \left| x^{(2)}_{i'd} - \hat{u}^{\mathbb{L}}_{\mathcal{L}d} \right| \right\} . \tag{7.10}$$

We then predict the undisclosed entries of $x^{(2)}_{i'*}$ based on the source combination of these assignment sets compared with the original data. The prediction error of object i' with respect to the undisclosed dimen-

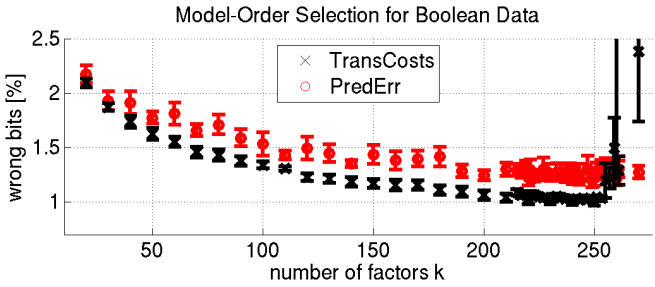


Figure 7.7: Model-order selection for Boolean matrix factorization. The two curves depict two different ways of computing the generalization error on a role mining dataset. While the transfer costs method provides a choice for the number of roles, the prediction error method estimates a monotonically decreasing generalization error and cannot be used for model-order selection.

sions $\mathcal{D}^u$ is then

$$g_{i'} := \frac{1}{|\mathcal{D}^u|} \sum_{d \in \mathcal{D}^u} \left| x_{i'd}^{(2)} - \hat{u}^{\mathbb{L}}_{\hat{\mathcal{L}}_{i'}d} \right| \ . \tag{7.11}$$

Finally, the total prediction error is $N_2^{-1} \sum_{i'=1}^{N_2} g_{i'}$.

Experimental Comparison and Discussion We directly apply both estimates of the generalization error to the role mining problem and use them to select the number of roles for a given real-world access control configuration $\mathbf{X}$ with 3000 users and 500 permissions. As in Chapter 4, we factorize this user-permission matrix into a user-role assignment matrix $\mathbf{Z}$ and a user-permission assignment matrix $\mathbf{U}$ by maximizing the likelihood for multi-assignment clustering derived in Section 4.1. We run five-fold cross-validation with 2400 users in the training set and 600 users in the validation set. In each repetition we compute the transfer costs as well as the prediction error. Prediction error uses $\kappa = 0.5$. The results are depicted in Figure 7.7.

The number of factors with lowest transfer costs is $k = 248$. In the underfitting regime, the transfer costs have low variance. The reason is that solutions that underfit are strongly influenced by the coarse underlying structure of the data. This structure is the same in all random validation sets. In the overfitting regime, the transfer costs vary significantly as, there, the noisy bits in the validation set determine how well the overfitted model matches the data. These bits change over the random validation sets.

The prediction test provides no minimum for selecting the number of roles. It decreases monotonically. This is due to how prediction error is computed. The computation of prediction error and transfer costs is very similar. There is only a small difference in the way of transferring the first part of the solution $\boldsymbol{s}^{(1)} = \left(\mathbf{Z}^{(1)}, \mathbf{U}^{(1)}\right)$ to the hold-out data. Prediction error fixes $\mathbf{U}^{(1)}$, discards $\mathbf{Z}^{(1)}$ and computes a new assignment matrix $\mathbf{Z}$ by selecting the best source combinations $\hat{\mathcal{L}}_{i'}$ for each object in the step described by Eq. (7.10). Transfer costs fixes a mapping between $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ prior to training the model. Then, it transfers the entire solution $\boldsymbol{s}^{(1)}$ to the hold out dataset. As a consequence, the transferred solution is more rigid than the one of prediction error. Prediction error essentially trains one part of the solution to fit the hold out data. For this reason it cannot detect solutions that overfit to the training dataset.

Our experimental data is of low entropy which means that most permissions are either assigned to all users or to nobody. There are only few 'discriminative' dimensions in this dataset (see the analysis of permission entropy in Fig. 6.1, top). With low-entropic data, prediction error is less precise than transfer costs also in the underfitting regime. By transferring the solutions based on only a fraction $\kappa = 0.5$ of all dimensions one decreases the chance to observe these discriminative dimensions. As a consequence, the prediction error is overall higher than the transfer costs. This makes it less sensitive to noisy bits. As the random selection of the dimensions has a big influence on the prediction error, also its variance is higher than the variance of transfer costs.

7.3.5 MTC for non-factorial models

The representation of the measurements plays an important role for optimization. In parametric clustering, the cost function can be written as a sum over independent object-wise costs $R_i(\boldsymbol{s}, \mathbf{x}_i, k)$ as shown in Eq. (7.3). When the measurements are characterized by pairwise similarities, instead of explicit coordinates, then such a form of the costs as in Eq. (7.3) and Eq. (7.4) does not exist.

In such cases we propose to individually transfer the objects of the second dataset to the solution trained on the first dataset. We add each such object to those parts of the solution where the cost increment is lowest. We demonstrate this procedure on a particular cost function. An example of a non-factorial model is the quadratic cost function for correlation clustering [6]. We only provide the original cost function and the transfer costs here. For a detailed experimental analysis we refer to [27].

Let N and k be the number of objects and the number of clusters, respectively. Correlation clustering partitions a graph with positive and negative edge labels. Given a graph $\mathcal{G}(O, \mathbf{X})$ with similarity matrix $\mathbf{X} := \{x_{ij}\} \in \{-1, 1\}^{\binom{N}{2}}$ between objects i and j and a clustering solution $\boldsymbol{s}$, the set of edges between two clusters u and v is defined as $E_{uv} = \{(i,j) \in E : \boldsymbol{s}(i) = u \wedge \boldsymbol{s}(j) = v\}$, where $\boldsymbol{s}(i)$ is the cluster index of object i. $E_{uv}, v \neq u$ are *inter-cluster edges* and E_{uu} are *intra-cluster edges*. The cost function counts the number of disagreements, i.e. the number of positive inter-cluster edges plus the number of negative intra-cluster edges:

$$R(\boldsymbol{s}, \mathbf{X}, k) = \frac{1}{2}\left(\sum_{u=1}^{k}\sum_{v=1}^{u}\sum_{(i,j)\in E_{uv}}(x_{ij}+1) - \sum_{u=1}^{k}\sum_{(i,j)\in E_{uu}}(x_{ij}-1)\right). \tag{7.12}$$

To transfer the clustering solution $\boldsymbol{s}^{(1)}$ to the second dataset $\mathbf{X}^{(2)}$, we add individual objects i' of the second dataset to those clusters v that add the smallest increment to the cost function. This cost increment

computes as

$$H(i', \boldsymbol{s}_v^{(1)}) = \frac{1}{2}\left(\sum_{\substack{u=1\\u\neq v}}^{k}\sum_{j\in\boldsymbol{s}_u}(x_{ij}+1) - \sum_{j\in\boldsymbol{s}_v}(x_{ij}-1)\right), \tag{7.13}$$

where $\boldsymbol{s}_v$ includes the set of objects whose cluster indices are v. The cluster index of object i' is then

$$\boldsymbol{s}^{(1)}(i') = \arg\min_{1\leq v\leq k} H(i', \boldsymbol{s}_v^{(1)}) \ . \tag{7.14}$$

7.3.6 MTC for k-means clustering

In this last example, we investigate the application of k-means to Gaussian data. Selecting the number of clusters in such a scenario is a conceptually difficult task, as we will see. A solution $\boldsymbol{s}$ of k-means is an assignment vector $\boldsymbol{c} \in \{1,..,k\}^N$ and k centroids $\boldsymbol{\mu}_t : t \in \{1,..,k\}$. Thereby, $c(i) = t$ means that object i is assigned to centroid t. The model order is the number of centroids k. The cost function of k-means is the sum of distances between each object and its centroid, i.e. $R(\boldsymbol{s}, \mathbf{X}, k) = \sum_i d\left(\boldsymbol{\mu}_c(i), \mathbf{x}_i\right)$. The distance function d depends on the data type (Hamming, squared Euclidean,...). As k-means provides a disjoint partitioning of the objects into the k clusters, one can rewrite the transfer cost formula:

$$\begin{aligned} R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k) &= \frac{1}{N_2}\sum_{i'=1}^{N_2}\sum_{i=1}^{N_1} d\left(\boldsymbol{\mu}_{c(i)}^{(1)}, \mathbf{x}_{i'}^{(2)}\right)\mathbf{I}_{\{\psi(i',\mathbf{X}^{(1)},\mathbf{X}^{(2)})=i\}} \\ &\approx \frac{1}{N_2}\sum_{i'}\sum_{t} d\left(\boldsymbol{\mu}_t^{(1)}, \mathbf{x}_{i'}^{(2)}\right)\mathbf{I}_{\{\psi^s(i',s^{(1)},\mathbf{X}^{(2)})=t\}}, \end{aligned} \tag{7.15}$$

where ψ is the nearest-neighbor mapping between objects and ψ^s is the mapping of objects to the nearest centroid as defined in Eq. (7.6). We use the fact that the centroids represent the objects which are assigned to them. Therefore, the centroid closest to $\mathbf{x}_{i'}^{(2)}$ is on average approximately as far away as the centroid of the nearest neighbor of $\mathbf{x}_{i'}^{(2)}$ in $O^{(1)}$. For high N_1 and N_2 this approximation becomes very precise.

The setup of the experiment is as follows: We sample 200 objects from three bivariate Gaussian distributions (see for instance Figure 7.8 top right). The task is to find the appropriate number of clusters. By altering the variances and the pairwise distances of the centers, we control the difficulty of this problem and especially tune it such that selecting the number of clusters is hard.

We investigate the selection of k with two mapping functions: the nearest-neighbor mapping of the objects from the second dataset to the centroids $\boldsymbol{\mu}^{(1)}$ and the generative mapping where the two data subsets are aligned by construction. We report the statistics over 20 random repetitions of generating the data.

Our findings for three different problem difficulties are illustrated in Figure 7.8. As expected, the costs on the training dataset monotonically decrease with k. When the mapping is given by the generation process of the data (generative mapping), MTC provides the true number of clusters in all cases. However, recall that the generative mapping requires knowledge of the true model parameters and leaks information about the true number of clusters to the costs.

Interestingly, MTC with a nearest-neighbor mapping follows almost exactly the same trend as the original costs on the first dataset and therefore proposes selecting the highest model-order that we offer to MTC. The higher the number of clusters is, the closer are the centroids of the nearest neighbors of each object. This reduces the transfer costs of high k. The only difference between original costs and transfer costs stems from the average distance between nearest neighbors (the data granularity). Only when the pairwise centroid distances become smaller than this distance, the transfer costs increase again. Ultimately, the favored solution is a vector quantization at the level of the data granularity. This is the natural behavior of k-means, as its cost function has no variances. As we have seen in the first experiments with Gaussian mixture models, fitting Gaussian data with MTC imposes no particular difficulties when the appropriate model (here GMM) is used. The k-means behavior on such data is due to the model mismatch.

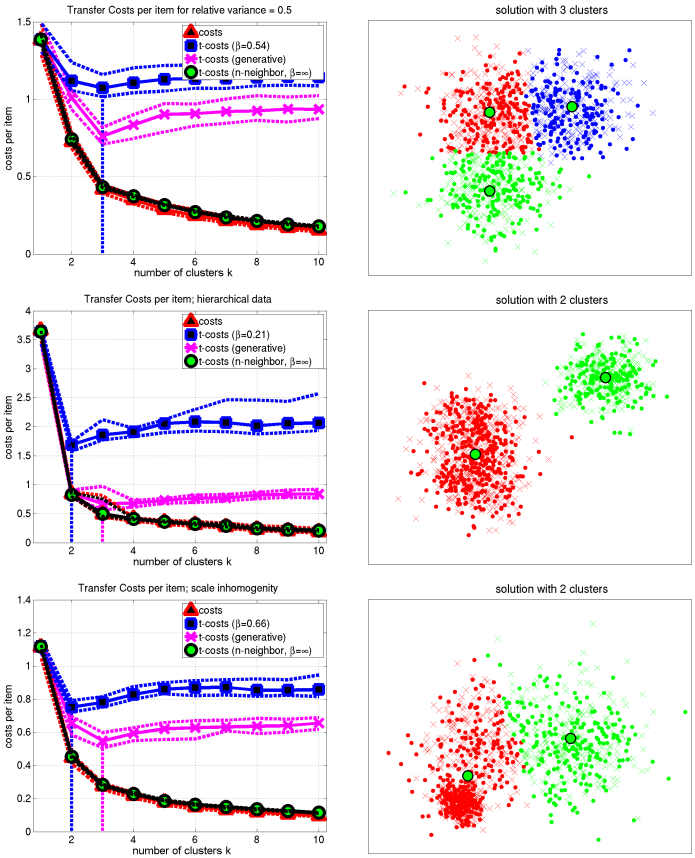


Figure 7.8: Costs and transfer costs (computed with mappings: nearest-neighbor, generative, probabilistic) for k-means clustering of three Gaussians. Solid lines indicate the median and dashed lines are the 25% and 75% percentiles. The right panel shows the clustering result selected by soft mapping MTC. Top: equidistant centers and equal variance. Middle: heterogeneous distances between centers (hierarchical). Bottom: heterogeneous distances and variances.

Probabilistic Mapping: A variant of MTC can be used to still make k-means applicable to estimating the true model order of Gaussian data. As follows, we extend the notion of a strict mapping to a probabilistic mapping between objects. Let $p_{i'i} := p(\psi(i', \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = i)$ be the probability that ψ maps object i' from the second dataset to object i of the first dataset. We define $p_{i'i}$ as

$$p_{i'i} := \frac{1}{Z} \exp\left(-\beta \; d(\mathbf{x}_i^{(1)}, \mathbf{x}_{i'}^{(2)})\right) \;, \qquad (7.16)$$
$$Z = \sum_i \exp\left(-\beta \; d(\mathbf{x}_i^{(1)}, \mathbf{x}_{i'}^{(2)})\right)$$

This mapping distribution is parameterized by the computational temperature β^{-1} and depends on the problem-specific dissimilarity function $d(\mathbf{x}_i^{(1)}, \mathbf{x}_{i'}^{(2)})$. A probabilistic mapping is more general than the deterministic function ψ. When β has a finite value, then objects are mapped to multiple objects. In the case of $\beta \to \infty$, it reduces to a deterministic nearest-neighbor mapping between $O^{(2)}$ and $O^{(1)}$. When $\beta = 0$ then object $i' \in O^{(2)}$ is mapped to all N_1 objects in $O^{(1)}$ with equal probability, thereby maximizing the entropy of $p_{i'i}$.

With this modified mapping, we define the transfer costs $R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k)$ of a factorial model with model-order k as follows:

$$R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k) \quad = \quad \frac{1}{N_2} \sum_{i'=1}^{N_2} \sum_{i=1}^{N_1} p_{i'i} \; R_{i'}(\boldsymbol{s}^{(1)}(i), \mathbf{x}_{i'}^{(2)}, k). \quad (7.17)$$

For k-means, taking the object to centroid approximation, this becomes

$$R^T(\boldsymbol{s}^{(1)}, \mathbf{X}^{(2)}, k) \approx \frac{1}{N_2 Z} \sum_{i'=1}^{N_2} \sum_{t=1}^{k} d\left(\boldsymbol{\mu}_t^{(1)}, \mathbf{x}_{i'}^{(2)}\right) e^{-\beta d(\boldsymbol{\mu}_t^{(1)}, \mathbf{x}_i^{(2)})} \;. \quad (7.18)$$

We fix the inverse temperature by the costs of the data with respect to a single cluster: $\beta = 0.75 \cdot R(\boldsymbol{s}^{(1)}, \mathbf{X}^{(1)}, 1)^{-1}$. This choice defines the dynamic range of the model-order selection problem. When fixing β roughly at the costs of one cluster, the resolution of individual pairwise distances resembles the visual situation where one looks at the entire data cloud as a whole.

Results of probabilistic mapping MTC: The probabilistic mapping finds the true number of clusters when the variances of the Gaussians are roughly the same, even for a substantial overlap of the Gaussians (Figure 7.8, top row). Please note that although the differences of the transfer costs are within the plotted percentiles, the rank-order of the number of clusters in each single experiment is preserved over the 20 repetitions, i.e. the variance mainly results from the data and not from the selection of k.

When the problem scale varies on a local level, fixing the temperature at the $k = 1$ solution does not resolve the dynamic range of the costs. We illustrate this by two hard problems: The middle problem in Figure 7.8 has a hierarchical structure, i.e. the pairwise distances between centers vary a lot. In the bottom problem in Figure 7.8, both the distances and the individual variances of the Gaussians vary. In both cases the number of clusters is estimated too low. When inspecting the middle plot, this choice seems reasonable, whereas in the bottom plot clearly three clusters would be desirable. The introduction of a computational temperature simulates the role of the variances in Gaussian mixture models. However, as the temperature is the same for all clusters, it fails to mimic situations where the variances of the Gaussians substantially differ. A Gaussian mixture model would be more appropriate than modeling Gaussian data with k-means.

7.3.7 Other model selection approaches to unsupervised learning

In this section we point to related work on model selection for unsupervised learning. Models that assume an explicit parametric form are often controlled by a model complexity penalty (a regularizer). Akaike information criterion (AIC) [3] and Bayesian information criterion (BIC) [68] both trade off the goodness of fit measured in terms of a likelihood function against the number of model parameters used. In [56], the model evidence for probabilistic PCA is maximized with respect to the number of components. Introducing approximations for the involved integral, this score equals BIC. In [38] the number of principal compo-

nents is selected by integrating over the sensitivity of the likelihood to the model parameters. Minimum Message Length (MML)[78] and Minimum description length (MDL) [64] select the lowest model order that can explain the data. It essentially minimizes the negative log posterior of the model and is thus formally identical to BIC [39]. It is unclear how to generalize model-based critera like [3, 68, 56, 38] to non-probabilistic methods such as, for instance, correlation clustering, being specified by a cost function instead of a likelihood.

Bayesian model selection as well as ASC have in common that, for a given data instance, one integrates over the entire hypothesis space of the model weighting how well each hypothesis fits the data. In contrast, the loss rank principle (LoRP) [42], fixes the hypothesis and integrates over all possible data realizations that are well fitted by this hypothesis. This method also requires only the cost function and does not depend on prior model assumptions. Algorithmic model complexity or Kolmogorov complexity [69] measures the length of the shortest program that describes the given observations. It is formally related to MML [78] as shown in [79].

For selecting the rank of truncated SVD, probably the most related approach to the transfer cost method is the cross-validation method proposed in [62]. It is a generalization of the method in [30] and was also applied to NMF. We explain it and compare with it in Section 7.3.3. A method with single hold-out entries (i, j) is proposed in [18]. It trains a SVD on the input matrix without row i and another one without column j. Then it combines $\mathbf{U}$ from one SVD and $\mathbf{V}$ from the other and averages their singular values to obtain an SVD which is independent of (i, j). The method in [18] has been reviewed in [62].

In [55], the authors abandon cross-validation for Boolean matrix factorization. They found that i) the method in [62] is not applicable and ii) using the rows of the second matrix of the factorization (here $\mathbf{U}$ in Section 7.3.4) to explain the hold-out data, tolerates overfitting. From our experience, cross-validation fails when only the second matrix is fixed and the first matrix is adapted to the new data. With a predefined mapping to transfer *both* matrices to the new data without adapting them, cross-validation works for Boolean matrix factorization as demonstrated

in Section 7.3.4.

Specialized to selecting the number of clusters in clustering, gap statistics have been proposed in [40]. Stability analysis has also shown promising results [17, 49]. However, stability neglects to account the informativeness of solutions.

An information theoretic model validation principle has been proposed in [8] to determine the tradeoff between stability and informativeness based on an information theoretic criterion called *approximation capacity*. In the next section we will explain this principle and demonstrate two of the first problems it has been applied to.

7.3.8 Summary

In this section, we defined the minimum transfer cost principle (MTC). Our method extends the cross-validation principle to unsupervised learning problems as it solves the problem of transferring a learned model from one dataset to another one when no labels are given. We demonstrated how to apply the principle to different problems such as maximum-likelihood inference, k-means clustering, correlation clustering, Gaussian mixture models, and rank-limited SVD, thereby highlighting its broad applicability. For each problem, we explained the appropriate mapping function between datasets and we demonstrated how the principle can be employed with respect to the specifications of the particular tasks. In all cases, MTC makes a sensible choice of the model order. It finds the optimal rank for image denoising with SVD and for error correction in access-control configurations. Moreover, it enables one to automatically determine the number of roles for role mining.

7.4 Approximation set coding

In this section we describe the framework of approximation set coding (ASC) as first proposed in [8]. For a given model and a given noisy dataset, ASC provides a criterion to determine how precisely the model parameters can be learned at the noise level of the data. For multiple models given, ASC enables one to prioritize the models according to their ability to robustly learn relevant information from the noisy data. It can thus serve as a model selection criterion. As the theory uses many novel concepts, we begin this section by a motivating example. In Section 7.4.3 we then derive the approximation capacity criterion. Finally, in Section 7.4.5 and Section 7.4.6, we demonstrate the ASC framework on two applications.

7.4.1 Notational preliminaries

We start by introducing a set of terms that will be frequently used throughout Section 7.4.

Object & Measurement Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{X}$ be a ordered set of N objects $\mathbf{O}$ and N measurements in a data space $\mathcal{X}$, where the measurements characterize the objects. Throughout this section, we assume the special case of a bijective map between objects and measurements, i.e., the i^{th} object is synonymous with the vector $\mathbf{x}_i \in \mathbb{R}^D$. In general, the (object, measurement) relation might be more complex than an object-specific feature vector. The set of objects is the input of each learning problem.

Hypothesis / solution A hypothesis, i.e. a solution c is defined by the particular output of the algorithm used to solve a problem. The problem of estimating a Gaussian, for instance, has a mean vector and a covariance matrix as a solution. In clustering the solution is defined by the assignments of objects to clusters. Generally, whenever one learns the parameters of a model then the solutions are defined by the learned model parameters.

Hypothesis class / solution space The hypothesis class $\mathcal{C}$ is the set of all solutions. For clustering, the hypothesis class are all object partitionings $\mathcal{C} = \{1, \ldots, K\}^N$ with the number of clusters K.

Learning algorithm A learning algorithm accepts the objects as an input and returns a solution as an output.

Cost function Each learning algorithm is characterized by a cost function or objective function R that assigns a real value to a solution:

$$R \ : \ \mathcal{C} \times \mathcal{X} \to \mathbb{R}, \qquad (c, \mathbf{X}) \mapsto R(c, \mathbf{X}) \ .$$

In order to make explicit that the costs depend on the set of randomly observed objects, we will sometimes call them the empirical costs. Usually, a learning algorithm aims at returning the solution that minimizes the empirical costs. For k-means, for instance, the costs are the sum of squared distances between objects and their centroids. For learning a Gaussian mixture model, one minimizes the negative log-likelihood of this model.

Cost minimizing solution Let $c^{\perp}(\mathbf{X})$ be the solution that minimizes the cost function, i.e. $c^{\perp}(\mathbf{X}) := \arg\min_c R(c, \mathbf{X})$.

7.4.2 The concept of approximation sets

The key concept in ASC is the notion of approximation sets. In this section, we introduce this concept. We motivate the ASC framework by investigating an intuitive learning problem. Imagine the task of estimating the center of a finite set of measurements on the real line. More precisely, given a set $\mathbf{X}$ of 10 real numbers sampled from an unknown Gaussian distribution, the task is to learn the mean μ of this distribution by minimizing the cost function $R(\mu, \mathbf{X})$. When we approach this task with the maximum likelihood principle, the costs are the negative log-likelihood of a Gaussian with given variance σ^2

$$R(\mu, \mathbf{X}) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2 + \log Z \ , \tag{7.19}$$

where Z is the constant that normalizes the Gaussian.

As the 10 measurements $\mathbf{X}$ are random variables, the global minimum $c^{\perp}(\mathbf{X})$ of the empirical costs is a random variable as well. Let $\mathbf{X}^{(q)}, q \in \{1, 2\}$, be two datasets with the same inherent structure (here: with the same true mean) but different noise realizations. In most cases, their global minima differ, i.e. $c^{\perp}(\mathbf{X}^{(1)}) \neq c^{\perp}(\mathbf{X}^{(2)})$. We illustrate this with an example in Figure 7.9. The cross markers and plus markers on the x-axis indicate the 10 objects of $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, respectively. Both datasets are sampled from the same unknown Gaussian distribution. Similarly, one could sample one dataset from the Gaussian and then randomly split it into two equally sized subsets. We repeat the estimation of the mean of this distribution on each of these two datasets and plot both cost functions. The vertical lines indicate the cost-minimizing solution for each dataset. The reader should realize that, in this example, the data space equals the solution space. Both the measurements and the estimated mean populate the real line. This is why one can plot both entities on the same axis. For more sophisticated problems, the data space and the solution space usually differs. For instance, for k-means clustering the measurements are elements of the D-dimensional Euclidean space and the solutions are assignments of objects to clusters (plus the centroids of the clusters).

Investigating the individual cost functions, we first observe that the costs provides a rank-order of the solutions. Parameter estimates that are close to the cost-minimizing solution also have low costs. Estimates that are further away have higher costs. The trend of the costs indicates how much the optimizer commits to a solution. Solutions with high costs would most likely be discarded whereas solutions with low costs are acceptable. In more technical terms, the trend of the costs determines the entropy of the optimizer.

As a second observation, we note that the two estimated means differ from each other. Thereby, the two cost minimizing solutions mutually indicate a cost level that seems to be acceptable given the distance of the two means. Comparing the different optimization outcomes of the two datasets, where would we localize another mean estimate of a third unseen dataset? A good guess would be an interval on the real line

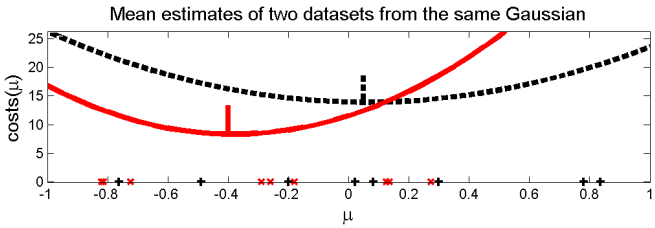


Figure 7.9: Two datasets with 10 observations sampled from the same Gaussian distribution respectively. The two plots depict the negative log-likelihood costs of fitting the mean μ of a Gaussian with fixed variance to each of the datasets. The cost-minimizing fits differ from each other due to the different noise realizations in the data. What would be a good confidence interval for the mean of a third dataset from that distribution?

enclosing the two empirical minima, i.e. a set of solutions approximately minimizing the cost function on *both* datasets.

We formalize this intuitive guess by defining an **approximation set**. This set includes all solutions that are similar in costs to the cost-minimizing solution $c^{\perp}(\mathbf{X}^{(q)})$. i.e.

$$\mathcal{Z}(\mathbf{X}^{(q)}) := \{c(\mathbf{X}^{(q)}) \in \mathcal{C} : R(c, \mathbf{X}^{(q)}) \leq R(c^{\perp}, \mathbf{X}^{(q)}) + \gamma\}, \; q \in \{1, 2\}\,. \tag{7.20}$$

Here, $\gamma \in \mathbb{R}_+$ is the approximation precision that controls the cardinality of the approximation sets $\mathcal{Z}(\mathbf{X}^{(1)})$ and $\mathcal{Z}(\mathbf{X}^{(2)})$. Of particular interest is the intersection of these two sets.

$$\mathcal{Z}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = \mathcal{Z}(\mathbf{X}^{(1)}) \cap \mathcal{Z}(\mathbf{X}^{(2)}) \tag{7.21}$$

This is the **joint approximation set**, the set of solutions that jointly approximates the min-cost solutions for both datasets. For sufficiently large γ, the joint approximation set includes the two cost-minimizing solutions. We illustrate all three sets in Figure 7.10 for two different values of γ. The dashed blue lines are the individual approximation sets and the solid blue line is their intersection.

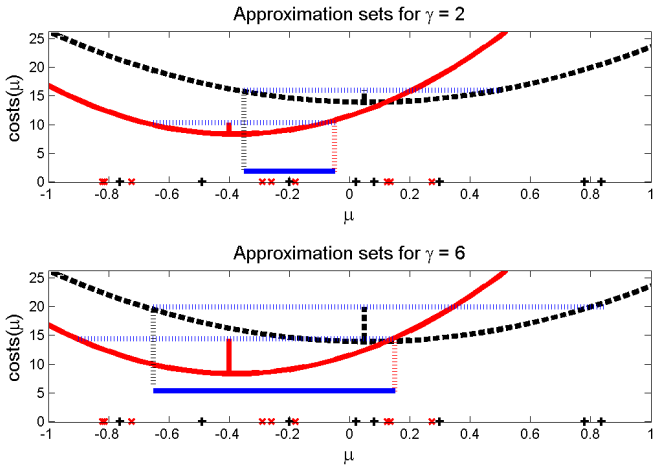


Figure 7.10: Computation of the approximation sets for estimating the mean of the datasets from Fig 7.9. The approximation sets $\mathcal{Z}(\mathbf{X}^{(1)})$ and $\mathcal{Z}(\mathbf{X}^{(2)})$ are depicted by dotted blue lines and the joint approximation set $\mathcal{Z}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)})$ is depicted by solid blue lines.

A positive γ provides robustness. While $c^{\perp}(\mathbf{X}^{(q)})$ changes whenever we optimize on another random subset q of the data, $\mathcal{Z}(\mathbf{X}^{(q)})$ remains comparably stable, if γ is sufficiently large. Therefore, $\mathcal{Z}(\mathbf{X}^{(q)})$ defines the inference resolution of the hypothesis class that is relevant in the presence of noise. High noise in the measurements reduces this resolution and thus coarsens the hypothesis class. As a consequence, the key problem of learning is to optimally control the resolution of the hypothesis class. Using the concept of approximation sets, this problem reduces to selecting the value of the scalar γ. How small can γ be chosen to still ensure identifiability of $\mathcal{Z}(\mathbf{X})$ under noise variations of the data $\mathbf{X}$? Conversely, choosing γ too high yields a too coarse resolution and does not capture the maximal amount of information in the data. For instance, when two datasets from *different* Gaussians are given, we require an estimator of the mean that is sensitive enough to separate the two true

means. At the same time we require the estimates of the individual mean parameters to be robust to noise. Approximation Set Coding (ASC) uses the approximation sets $\mathcal{Z}(\mathbf{X})$ as symbols in a communication scenario to derive a trade-off criterion for an optimal approximation resolution of a learning algorithm. We describe this communication scenario in the next section.

Variance versus approximation precision γ. Before we introduce the communication protocol of ASC, we highlight a special property of our running example. As explained above, for a Gaussian with fixed variance σ^2, the parameter γ controls the approximation precision by modifying the cardinalities of the sets $\mathcal{Z}(\mathbf{X}^{(1)})$, $\mathcal{Z}(\mathbf{X}^{(2)})$, and $\mathcal{Z}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)})$. However, one could as well fix γ and vary σ to modify the cardinalities. We illustrate this in Fig 7.11.

The same cardinality of the approximation sets can either be obtained by tuning γ for a fixed variance σ^2, or by tuning σ for fixed γ. As a consequence, for a fixed cardinality $\left|\mathcal{Z}(\mathbf{X}^{(q)})\right|$, σ and γ are strongly coupled. When σ increases, γ must decrease and vice versa. This inverse relation nicely demonstrates the fundamental connection between noise and approximation precision. While the variance of a Gaussian is a parameter that directly indicates the noise level of the observations sampled from that Gaussian, γ controls how precisely we can narrow down our estimate of the mean.

7.4.3 Coding with approximation sets and approximation capacity

There are two competing requirements for inference that can be described in terms of approximation sets. On the one hand we require that a learning algorithm is able to resolve the relevant information in the data. For instance for datasets from two Gaussians with different means, the algorithm should ideally narrow down the estimate of the means such that they can be discriminated, i.e. the overlap of their approximation sets should be small. On the other hand, noise should not significantly influence the output of a learning algorithm. In our exam-

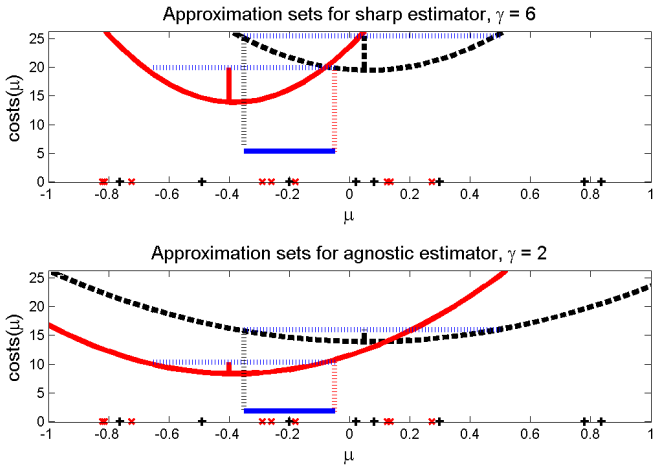


Figure 7.11: Two different cost functions for estimating the mean of the datasets from Fig 7.9. Top: Gaussian estimator with low variance constituting a 'sharp' cost function. Bottom: Gaussian estimator with higher variance, providing an 'agnostic' estimate of the mean μ. Adapting the variance of the Gaussian estimator influences the size of the approximation sets as well as adapting the approximation precision γ.

ple from the last section, this robustness property means that, despite the noise in the data, the overlap of the approximation sets should be large when the two datasets are sampled from the same Gaussian.

We balance the trade-off between these two requirements by selecting the parameter γ in Eq. (7.20). The two extremes of this trade-off are specified by $\gamma = 0$ and $\gamma \to \infty$. As we have seen in the last section, the maximal precision ($\gamma = 0$) cannot be resolved when the data is contaminated with noise. With $\gamma = 0$, only the cost-minimizing solutions would be in the approximation sets and their intersection (the joint approximation set) would be empty. An infinitely large γ would result in a single approximation set covering the entire solution space. In this case, no information can be inferred from data as all datasets would have exactly

the same approximation set. All intermediate choices of γ coarsen the solution space into partially overlapping approximation sets.

We determine the approximation precision γ by means of a hypothetical communication scenario. Distinguishable approximation sets can be used as symbols for communication. They allow us to identify the optimal resolution of the hypothesis class. The communication architecture includes a *sender* $\mathfrak{S}$ and a *receiver* $\mathfrak{R}$ with a *problem generator* $\mathfrak{PG}$ between the two terminals $\mathfrak{S}$, $\mathfrak{R}$ (see Fig. 7.12). The communication protocol is organized in two stages: (i) the design of a codebook and (ii) the communication process.

i) Codebook design To design a coding scheme, we adapt Shannon's random coding scenario, where a codebook of random bit strings covers the space of all bit strings. The sender sends a bit string and the receiver observes a perturbed version of this bit string. For decoding, the receiver has to find the most similar codebook vector in the codebook which is the decoded message. In the same spirit, for our scenario, the sender must communicate solutions to the receiver via noisy datasets. Since we are interested in solutions with low costs, the set of solutions that approximately minimizes the costs (the approximation set) can serve as a message. We generate the other solutions in the codebook by transforming the data $\tau \circ \mathbf{X}^{(1)}$ with the transformation $\tau \in \mathcal{T} := \{\tau_1, ..., \tau_{2^{n\rho}}\}$. The number of codewords is $2^{n\rho}$ and ρ is the rate of the protocol. The choice of such transformations depends on the hypothesis class and they have to be equivariant, i.e., the transformed min-cost solution equals the min-cost solution of the transformed data: $\tau \circ c(\mathbf{X}^{(1)}) = c(\tau \circ \mathbf{X}^{(1)})$.

For our running example, the transformation would be a *shift* vector added to all objects on the real line. Accordingly, the optimal mean would be shifted by the same vector. With applying such transformation vectors to $c^{\perp}(\mathbf{X}^{(1)})$, the entire solution space can be covered. In data clustering, the appropriate transformations for covering the solution space are inter-cluster *permutations* of the indices of the objects. Each cluster assignment matrix $\mathbf{z} \in \mathcal{C}(\mathbf{X}^{(1)})$ can be transformed into another clustering solution by a permutation τ on the rows of $\mathbf{z}$ which equals a permutation of the object indices.

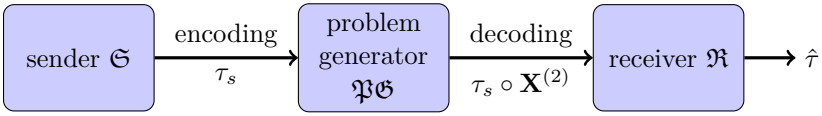


Figure 7.12: Communication process: (1) the sender selects transformation τ_s, (2) the problem generator draws $\mathbf{X}^{(2)} \sim \mathbb{P}(\mathbf{X})$ and applies τ_s to it, and (3) the receiver estimates $\hat{\tau}$ based on $\tilde{\mathbf{X}} = \tau_s \circ \mathbf{X}^{(2)}$.

ii) Communication To communicate, $\mathfrak{S}$ selects a transformation $\tau_s \in \mathcal{T}$ and sends it to a **problem generator** $\mathfrak{PG}$ as depicted in Fig. 7.12. $\mathfrak{PG}$ then generates a new dataset $\mathbf{X}^{(2)}$, applies the transformation τ_s, and sends the resulting data $\tilde{\mathbf{X}} := \tau_s \circ \mathbf{X}^{(2)}$ to $\mathfrak{R}$. On the receiver side, the lack of knowledge on the transformation τ_s is mixed with the stochastic variability of the source generating the data. $\mathfrak{R}$ must estimate the transformation $\hat{\tau}$ based on $\tilde{\mathbf{X}}$. The decoding rule of $\mathfrak{R}$ selects the transformation $\hat{\tau}$ that yields the largest joint approximation set of $\hat{\tau} \circ \mathbf{X}^{(1)}$ and $\tilde{\mathbf{X}}$

$$\hat{\tau} = \arg\max_{\tau \in \mathcal{T}} \mathcal{Z}(\tau \circ \mathbf{X}^{(1)}, \tau_s \circ \mathbf{X}^{(2)}) \tag{7.22}$$

In the absence of noise in the data, we have $\mathbf{X}^{(1)} = \mathbf{X}^{(2)}$, and error-free communication works even for $\gamma = 0$ because the approximation set $\mathcal{Z}(\tilde{\mathbf{X}})$ (containing only the cost minimizing solution) always exclusively overlaps with $\mathcal{Z}(\tau_s \circ \mathbf{X}^{(1)})$, the approximation set of the correct transformation. The higher the noise level, the larger we must choose γ in order to obtain approximation sets that are approximately invariant under the stochastic fluctuations in the measurements thus preventing decoding errors.

Error analysis and approximation capacity. The identifiability of codewords on the receiver side determines the condition of (asymptotically) error free communication. If there is at least one other transformation $\tau_{j \neq s}$ with a larger joint approximation set $\mathcal{Z}_j^{(1,2)} := \mathcal{Z}(\tau_j \circ \mathbf{X}^{(1)}, \tau_s \circ \mathbf{X}^{(2)})$ than the correct one $\mathcal{Z}_s^{(1,2)} := \mathcal{Z}(\tau_s \circ \mathbf{X}^{(1)}, \tau_s \circ \mathbf{X}^{(2)})$, there is a

decoding error. The error analysis measures

$$\mathbb{P}(\hat{\tau} \neq \tau_s | \tau_s, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = \mathbb{P}\left(\bigvee_{j \in \mathcal{T} \setminus \{\tau_s\}} \left| \mathcal{Z}_j^{(1,2)} \right| \geq \left| \mathcal{Z}_s^{(1,2)} \right| \right) \tag{7.23}$$

$$\stackrel{(a)}{\leq} \mathbb{E}_{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}} \left[\sum_{2 \leq j \leq 2^{n\rho}} \mathbf{I}\{ \left| \mathcal{Z}_j^{(1,2)} \right| \geq \left| \mathcal{Z}_s^{(1,2)} \right| \} \right]$$

$$\stackrel{(b)}{\leq} \mathbb{E}_{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}} \left[\sum_{2 \leq j \leq 2^{n\rho}} \frac{\left| \mathcal{Z}_j^{(1,2)} \right|}{\left| \mathcal{Z}_s^{(1,2)} \right|} \right]. \tag{7.24}$$

In step (a), we applied the union bound and left out the first term with $j = s = 1$. In (b), we linearly upper-bounded the step-function

$$\mathbf{I}\{ \left| \mathcal{Z}_j^{(1,2)} \right| \geq \left| \mathcal{Z}_s^{(1,2)} \right| \} \leq \frac{\left| \mathcal{Z}_j^{(1,2)} \right|}{\left| \mathcal{Z}_s^{(1,2)} \right|}. \tag{7.25}$$

The expectations in Eq. (7.24) must be carried out over all noise realizations in $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$. Moreover, in principle, one must compute the error independently of the submitted codeword and integrate $\mathbb{P}(\hat{\tau} \neq \tau_s | \tau_s, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \mathbb{P}(\tau_s)$ over all τ_s. In practice, we will only compute an estimation based one the two datasets at hand and the identity transformation. In the last line of Eq. (7.24) the sum over all joint approximation sets other than the correct one factorizes:

$$\begin{aligned} \sum_{2 \leq j \leq 2^{n\rho}} |\mathcal{Z}_j^{(1,2)}| &= \sum_j \sum_{c \in \mathcal{Z}(\mathbf{X}^{(2)})} \mathbf{I}\{c \in \mathcal{Z}(\tau_s \circ \mathbf{X}^{(2)})\} \mathbf{I}\{c \in \mathcal{Z}(\tau_j^{-1} \circ \mathbf{X}^{(1)})\} \\ &= \sum_j \sum_{c,c'} \mathbf{I}\{c \in \mathcal{Z}(\tau_s \circ \mathbf{X}^{(2)})\} \mathbf{I}\{c' \in \mathcal{Z}(\mathbf{X}^{(1)})\} \mathbf{I}\{c' = \tau_j^{-1} \circ c\} \\ &= \sum_c \mathbf{I}\{c \in \mathcal{Z}(\tau_s \circ \mathbf{X}^{(2)})\} \sum_{c'} \mathbf{I}\{c' \in \mathcal{Z}(\mathbf{X}^{(1)})\} \sum_j \mathbf{I}\{c' = \tau_j^{-1} \circ c\} \\ &= |\mathcal{Z}^{(1)}| \, |\mathcal{Z}^{(2)}| \end{aligned} \tag{7.26}$$

We abbreviated $\mathcal{Z}^{(q)} := \mathcal{Z}(\mathbf{X}^{(q)})$, $q = 1, 2$. In the last step we used that the solutions c' and $\tau_j^{-1} \circ c$ equal for exactly one transformation τ_j. This holds as we constructed the transformations such that one can reach each point in the solution space by applying exactly one unique transformation to one fixed solution.

Substituting back Eq. (7.26) to Eq. (7.24), we see that the probability of decoding error depends on the relation between the individual approximation sets and their intersection.

$$\mathbb{P}(\hat{\tau} \neq \tau_s | \tau_s, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \quad \leq \quad \frac{\left|\mathcal{Z}^{(1)}\right| \left|\mathcal{Z}^{(2)}\right|}{\left|\mathcal{Z}_s^{(1,2)}\right|} \tag{7.27}$$

As derived in [8], an asymptotically vanishing error probability is achievable for bounded communication rates

$$\rho \quad \leq \quad \hat{\mathcal{I}}_\gamma(\tau_j, \hat{\tau}) \quad = \quad \frac{1}{N} \mathbb{E}_{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}} \log \left(\frac{|\{\mathcal{T}\}||\mathcal{Z}_s^{(1,2)}|}{|\mathcal{Z}^{(1)}||\mathcal{Z}^{(2)}|} \right) \tag{7.28}$$

Here, $|\{\mathcal{T}\}|$ denotes the number of possible realizations of the transformation τ_s. This cardinality determines the possible realizations of randomly selected transformations which adopt the role of codebook vectors. This maximum size of the codebook should not be confused with the (smaller) size of the codebook $|\mathcal{T}|$ that can actually be used for communication. The more parameters a cost function has, the larger is $|\{\mathcal{T}\}|$. Therefore, $|\{\mathcal{T}\}|$ directly accounts for the model complexity. The fraction $|\mathcal{Z}_s^{(1,2)}|/(|\mathcal{Z}^{(1)}||\mathcal{Z}^{(2)}|)$ measures the stability of the model under noise fluctuations. This fraction is in $[0, 1]$ and thereby controls the effective size of the codebook.

$\hat{\mathcal{I}}_\gamma(\tau_j, \hat{\tau})$ is the **mutual information** between the true transformations and the decoded transformations. The hat indicates that this quantity involves an expectation over the datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$. This is a quenched average, as the expectation includes the logarithm in Eq. 7.28 (opposed to the annealed average where the expectation affects only the partition functions, see [53], Section 5.4 thereof).

In analogy to information theory, we define the **approximation capacity** as the maximum of $\hat{\mathcal{I}}_\gamma(\tau_j, \hat{\tau})$ with respect to the distribution of

the input of the cannel. In our case, the input (the set of transformations used for coding) is controlled by the approximation precision γ.

$$C(\tau_j, \hat{\tau}) := \max_{\gamma} \hat{\mathcal{I}}_{\gamma}(\tau_j, \hat{\tau})\,. \tag{7.29}$$

For a given cost function and a given dataset, maximizing the mutual information resolves the complexity-robustness tradeoff for this cost function. At capacity, the cost function robustly infers the maximum amount of information that is possible given the noise-level of the dataset. The higher the noise level raises the coarser should the approximation of the optimization problem be chosen, i.e. the larger is the value of γ that achieves capacity and the lower is the capacity itself.

For a given set of cost functions $\mathcal{R}$, we first compute the capacity for each of the cost functions, i.e. we maximize Eq. (7.28) with respect to γ for each individual cost function. Then we select the cost function that achieves highest capacity.

We assume that the data source provides two datasets from the same probability distribution. If only one dataset is given one can always simulate this scenario by sub-sampling or randomly splitting the dataset. The procedure for computing the capacity of a cost function for given data involves the following steps. First, if necessary, randomly split the given dataset $\mathbf{X}$ into two subsets $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. Second, train the min-cost solutions on each of the datasets and use them to compute the mutual information (Eq. 7.28) and maximize it with respect to γ.

The computation of the approximation sets is different for different kinds of data and for different cost functions. Particularly, for each problem one must select the appropriate set of transformations of problem solutions. In the next section we introduce notation that will prove useful in the successive section. In Section 7.4.5 we demonstrate how to apply ASC to the task of model selection for multi-assignment clustering. In Section 7.4.6 we use ASC to select the optimal cutoff rank for denoising a binary matrix with truncated SVD.

7.4.4 Approximation weights

In order to simplify the calculations in the subsequent sections, we introduce a few concepts that are convenient when applying ASC. The concept of approximation sets provides intuition for the problem of selecting the approximation precision of a learning algorithm. $\mathcal{Z}(\mathbf{X}^{(q)})$ provides a hard cutoff between solutions that are acceptable in terms of costs and those solutions that exceed the costs by γ. However, for the computation of the approximation capacity the mathematical concept of a set is a bit cumbersome. It is more convenient to use soft **approximation weights** w instead of hard approximation sets.

$$w : \mathcal{C} \times \mathcal{X} \times \mathbb{R} \to [0, 1] , \qquad (c, \mathbf{X}, \beta) \mapsto w_\beta(c, \mathbf{X}) . \tag{7.30}$$

Thereby, we require that i) weights should be non-negative and ii) solutions with lower costs should have a larger weight, i.e.,

$$R(c, \mathbf{X}) \leq R(\tilde{c}, \mathbf{X}) \quad \Longleftrightarrow \quad w_\beta(c, \mathbf{X}) \geq w_\beta(\tilde{c}, \mathbf{X}) . \tag{7.31}$$

The family of (Boltzmann) weights $w_\beta(c, \mathbf{X}) := \exp(-\beta R(c, \mathbf{X}))$, parameterized by the inverse computational temperature β, fulfills these requirements. These weights define the two **weight sums** $\mathcal{Z}^{(q)}$ and the **joint weight sum** $\mathcal{Z}^{(1,2)}$

$$\mathcal{Z}^{(q)} := \sum_{c \in \mathcal{C}} \exp(-\beta R(c, \mathbf{X}^{(q)})), \; q = 1, 2 \tag{7.32}$$

$$\mathcal{Z}^{(1,2)} := \sum_{c \in \mathcal{C}} \exp(-\beta (R(c, \mathbf{X}^{(1)}) + R(c, \mathbf{X}^{(2)}))), \tag{7.33}$$

where $\exp(-\beta(R(c, \mathbf{X}^{(1)}) + R(c, \mathbf{X}^{(2)})))$ measures how well a solution c minimizes costs on *both* datasets. Note that if the order of the data items in the datasets plays a role for the cost function then the two datasets $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ must be aligned to compute Eq. (7.33). If this is not provided by generation of the data then one can use the mapping functions introduced for the transfer cost principle. The term $R(c, \mathbf{X}^{(2)})$ in Eq. (7.33) equals the transfer costs of solution c.

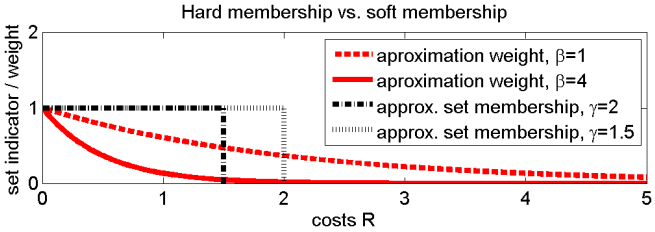


Figure 7.13: Comparison between approximation set membership indicator and approximation weights as a function of the costs. For each γ there is a β such that for both functions the area under curve equals.

The sums (7.32, 7.33) play the role of the approximation sets. In fact, they behave in the same way. If $\beta = 0$, all weights $w_\beta(c, \mathbf{X}) = 1$ are independent of the costs. In this case, $\mathcal{Z}^{(q)} = |\mathcal{C}(\mathbf{X}^{(q)})|$ indicates the size of the hypothesis space, and $\mathcal{Z}^{(1,2)} = \mathcal{Z}^{(1)} = \mathcal{Z}^{(2)}$. For large β, all weights are small compared to the weight $w_\beta(c^\perp, \mathbf{X}^{(q)})$ of the global optimum and the weight sum essentially counts the number of globally optimal solutions (this is the case with $\gamma = 0$ which leads to an approximation set containing only the min-cost solution). For intermediate values of β, $\mathcal{Z}(\cdot)$ takes a value between 0 and $|\mathcal{C}(\mathbf{X}^{(q)})|$, giving rise to the interpretation of $\mathcal{Z}(\cdot)$ as the effective number of patterns that approximately fit the dataset $\mathbf{X}^{(q)}$, where β defines the precision of this approximation.

The approximation sets and the weight sums can be treated equally. In fact, for each approximation precision γ there is a computational temperature β^{-1} such that the cardinality of the approximation set $\left|\mathcal{Z}^{(q)}(\gamma)\right|$ equals the magnitude of the weight sum $\left|\mathcal{Z}^{(q)}(\beta)\right|$. For this reason we denote them by the same symbol $\left|\mathcal{Z}^{(q)}\right|$. For the set concept, the norm $|.|$ is the integral over a hard set membership function (a heavyside step function $\theta(\gamma - R(c, \mathbf{X}^{(q)}))$). For the weight concept the norm has no influence as the sum over all weights already returns a positive real number.

Coming back to the comparison between approximation precision γ and the variance of a Gaussian in Section 7.4.2, we can directly relate the computational temperature β^{-1} with the variance σ^2. In the following

sections we will always use the weight sums to compute the approximation capacity.

7.4.5 ASC for choosing the noise model

In this section we use approximation capacity to prioritize a set of different probabilistic models for multi-assignment clustering for Boolean data. In the same way as in Section 4.4.2, we generate data from a set of Boolean source vectors. With available ground truth one can rank the candidate models according to their parameter estimation accuracy. We investigate whether ASC reproduces this ranking or not. Our hypothesis is that, at a given noise level, the model with highest approximation capacity equals the model that yields the most accurate source parameter estimates.

Data We generate datasets as in Figure 4.1. Here, we briefly recapitulate the generation process. Each data item $i \in \{1, .., N\}$ is assigned to a set of sources $\mathcal{L}_i$. These sources generate the D Boolean measurements $\mathbf{x}_{i*}$ of the data item. The probabilities β_{kd} of source k to emit a zero in dimension $d \in \{1, .., D\}$ parameterize the sources. We sample one D-dimensional bit-vector from each source in $\mathcal{L}_i$. Then we combine these vectors by a disjunction, leading to $\mathbf{x}_{i*}$. Finally, a noise process modifies $\mathbf{x}_{i*}$ by randomly selecting a fraction of ϵ elements and replacing them with random values. Each item represents a row of the matrix $\mathbf{x} \in \{0, 1\}^{N \times D}$.

Model variants Following the generative process of the data, the negative log-likelihood is $R_{\text{tot}} = \sum_i R(\boldsymbol{\beta}_{\mathcal{L}_i *}, \mathbf{x}_{i*}^{(q)})$, where the costs of assigning data item i to source set $\mathcal{L}$ are

$$R(\boldsymbol{\beta}_{\mathcal{L}*}, \mathbf{x}_{i*}^{(q)}) = -\sum_{d=1}^{D} \log \left[(1-\epsilon) \left(1 - \beta_{\mathcal{L}d}\right)^{x_{id}} \beta_{\mathcal{L}d}^{1-x_{id}} + \epsilon \, r^{x_{id}} \left(1-r\right)^{1-x_{id}} \right] \tag{7.34}$$

The model parameter ϵ is the mixture weight of the noise process, and r is the probability for a noisy bit to be 1. Fixing $\epsilon = 0$ corresponds to

a generative model without noise process. We call models with finite ϵ 'mix' models.

Further model variants concern the source combinations. MAC supports the simultaneous assignment of one data item to more than one source, i.e. a source set may contain more than one element ($|\mathcal{L}| \geq 1$), while SAC has the constraint $|\mathcal{L}_i| = 1$ for all i. Hence, when MAC has K sources and L different source combinations, the SAC model requires L independent sources for an equivalent model complexity. For MAC, $\beta_{\mathcal{L}d} := \prod_{\lambda \in \mathcal{L}} \beta_{\lambda d}$ is the product of all source parameters in assignment set $\mathcal{L}$, while for SAC, $\beta_{\mathcal{L}d}$ is an independent parameter of the cluster indexed by $\mathcal{L}$. SAC thus must estimate $L \cdot D$ parameters, while MAC uses the data more efficiently to only learn $K \cdot D$ parameters. In summary, there are four model variants:

- 1) MACmix: $|\mathcal{L}| \geq 1$, $\boldsymbol{\beta} \in [0,1]^{K \cdot D}$, $\epsilon \in [0,1[$
- 2) SACmix: $|\mathcal{L}| = 1$, $\boldsymbol{\beta} \in [0,1]^{L \cdot D}$, $\epsilon \in [0,1[$
- 3) MAC no Noise: $|\mathcal{L}| \geq 1$, $\boldsymbol{\beta} \in [0,1]^{K \cdot D}$, $\epsilon = 0$
- 4) SAC no Noise: $|\mathcal{L}| = 1$, $\boldsymbol{\beta} \in [0,1]^{L \cdot D}$ and $\epsilon = 0$

Computation of the approximation capacity. In order to compute the approximation capacity, we must identify the hypothesis space of the models. For these models the hypothesis space is spanned by all possible assignments of objects to source combinations. A solution (a point in this hypothesis space) is encoded by the N assignment-set indicators $\mathcal{L}_i \in \{1, .., L\}, i \in \{1, .., N\}$. Each of these indicators can take any out of L values. We explained in the last paragraph that L has the same magnitude for all four model variants. Therefore, the hypothesis space of all four models equals in cardinality.

The appropriate transformation in clustering problems is the permutation of objects. Albeit a solution contains the cluster assignments *and* cluster centroids, the centroid parameters contribute almost no entropy to the solution. With given cluster assignments the solution is

fully determined as the objects of each cluster pinpoint the centroids to a particular vector. With the permutation transformations one can construct all clusterings starting from a single clustering. However, as the mutual information in Eq. (7.28) is estimated solely based on the identity transformation, one can ignore the specific kind of transformations when computing this estimate. The sum over all possible transformations would be required to carry out the exact computation of the error probability in the first line of Eq. 7.23). In Eq. (7.28) the transformations have no influence anymore.

As the probabilistic model factorizes over the objects (and therefore the costs are a sum over object-wise costs $R(\boldsymbol{\beta}_{\mathcal{L}_i*}, \mathbf{x}_{i*}^{(q)})$ in Eq. (7.34)) we can conveniently sum over the entire hypothesis space by just summing over all possible assignment sets for each object. Therefore, the weight sums are

$$\mathcal{Z}^{(q)} = \prod_{i=1}^{N} \sum_{\mathcal{L}=1}^{L} \exp\left(-\beta R(\boldsymbol{\beta}_{\mathcal{L}*}, \mathbf{x}_{i*}^{(q)})\right), \; q = 1, 2 \tag{7.35}$$

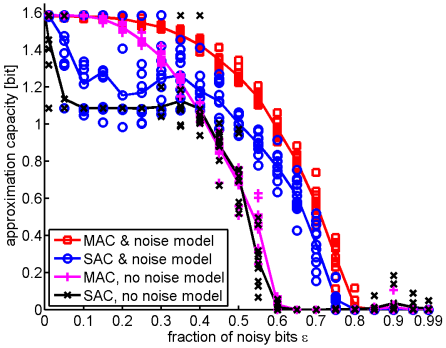
$$\mathcal{Z}^{(1,2)} = \prod_{i=1}^{N} \sum_{\mathcal{L}=1}^{L} \exp\left(-\beta (R(\boldsymbol{\beta}_{\mathcal{L}*}, \mathbf{x}_{i*}^{(1)}) + R(\boldsymbol{\beta}_{\mathcal{L}*}, \mathbf{x}_{i*}^{(2)}))\right) \tag{7.36}$$

where the two datasets must be aligned before computing $R(\boldsymbol{\beta}_{\mathcal{L}*}, \mathbf{x}_{i*}^{(2)})$ such that $\mathbf{x}_{i*}^{(1)}$ and $\mathbf{x}_{i*}^{(2)}$ have a high probability to be generated by the same sources. This cost term for the second dataset is equivalent to the transfer costs measure. In this particular example of Boolean vectors, we use the Hamming distance to compute a mapping that aligns the two datasets.

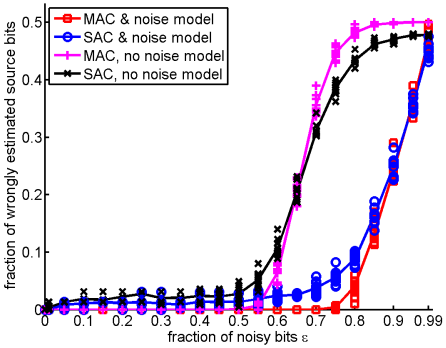
The weight sums of the four model variants differ only in the combined source estimates $\boldsymbol{\beta}_{\mathcal{L}*}, \forall \mathcal{L}$. We train these estimates on the first dataset $\mathbf{x}^{(1)}$ prior to computing the mutual information Eq. (7.28). Having the formulas for the weight sums, one can readily evaluate the mutual information as a function of the inverse computational temperature β. We maximize this function numerically for each of the model variants. In Appendix B.1 we provide formulas for gradient ascent. Of course, one can as well only run a sweep search starting from $\beta = 0$.

Experiments. We investigate the dependency between the accuracy of the source parameter estimation and the approximation capacity. We choose a setting with 2 sources and draw 100 samples from each single source as well as from the combination of the two sources. The sources have 150 dimensions and a Hamming distance of 40 bits. To control the difficulty of the inference problem, we vary the fraction ϵ of random bits between 0 and 0.99. The parameter of the Bernoulli-noise process is $r = 0.75$. The model parameters are then estimated by MAC and SAC both with and without a noise model. We use the true model order, i.e. $K = 2$ and $L = 3$ and infer the parameters by deterministic annealing.

The mismatch of the estimates to the true sources and the approximation capacity are displayed in Figures 7.14(b) and 7.14(a), both as a function of the noise fraction ϵ. Each method has very precise estimates up to a model-dependent critical noise level. For higher noise values, the accuracy breaks down. For both MAC and SAC, adding a noise model shifts this performance decay to a higher noise level. Moreover, MACmix estimates the source parameters more accurately than SACmix and shows its performance decay at a significantly increased noise level. The approximation capacity (Fig. 7.14(a)) confirms this ranking. For noise-free data ($\epsilon = 0$), all four models attain the theoretical maximum of the approximation capacity, $\log_2(3)$ bits. As ϵ increases, the approximation capacity decreases for all models, but we observe vast differences in the sensitivity of the capacity to the noise level. Two effects decrease the capacity: inaccurately estimated parameters and (even with perfect estimates) the noise in the data that favors uniform object-source assignment probabilities (for $\epsilon = 1$ all clusters are equally probable). In conclusion, the approximation capacity confirms the ranking by parameter accuracy. We emphasize that parameter accuracy does require knowledge of the true source parameters while ASC requires only the data at hand.



(a) Approximation Capacity



(b) Source Estimation Mismatch

Figure 7.14: Relation between error of source parameter estimation and approximation capacity. The rank order of the model variants is identical for both measures. Computing the capacity does not require knowledge on the ground truth, whereas parameter estimation error depends on this information.

7.4.6 ASC for truncated SVD

In this section we apply the ASC framework to select the cut-off rank k for SVD. We regard truncated SVD as a model that is parametrized by k. Therefore, the model selection task is finding the rank k^* with maximal approximation capacity of the corresponding model. This search involves maximizing Eq.(7.28) for all ranks that we investigate. We identify the following entities to be relevant for applying ASC to truncated SVD. The input of the problem is a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$. The cost function is the Frobenius norm:

$$R(\mathbf{X}, \mathbf{U}, \mathbf{S}, \mathbf{V}) = \left\| \mathbf{X} - \mathbf{U}\mathbf{S}\mathbf{V}^T \right\|_F^2 \tag{7.37}$$

$$= \sum_{i,j} \left(x_{ij} - \sum_{t=1}^{k} u_{it} s_{tt} v_{tj} \right)^2 \tag{7.38}$$

An optimizer for a problem with data $\mathbf{X}^{(q)}$ returns a decomposition $c^{\perp}(\mathbf{X}^{(q)}) = \mathbf{U}^{(q)}\mathbf{S}^{(q)}\mathbf{V}^{(q)}$, where all but the first $k \leq \min(N, D)$ diagonal entries of $\mathbf{S}^{(q)}$ are zero due to truncation (we will denote the third matrix by $\mathbf{V}$ instead of $\mathbf{V}^T$ for convenience). The upper index k of the sum in Eq.(7.37) parametrizes the model order.

The solution $c^{\perp}(\mathbf{X}^{(q)})$ gives the closest rank-k approximation of $\mathbf{X}^{(q)}$ with respect to the Frobenius norm. In the case of SVD, the hypothesis $c = (\mathbf{U}, \mathbf{S}, \mathbf{V})$ is a particular decomposition of the input matrix (this contrasts our clustering application in the last section where c is a relation that assigns objects to sets of clusters or sources). The $k \times D$ matrix $\mathbf{W}$ with the entries $w_{tj} := s_{tt} v_{tj}$ is a new basis and $\mathbf{U}$ provides the linear weights needed to represent the data $\mathbf{X}$ in this basis. When the empirical mean of $\mathbf{X}$ is the origin, this representation corresponds to principal components analyis (PCA).

We define the hypothesis space for truncated SVD with cut-off rank k as follows. For a fixed basis $\mathbf{W}$, the hypothesis space is spanned by all $N \times k$ matrices $\mathbf{U}$. For a given dataset $\mathbf{X}^{(q)}$, $\mathbf{U}^{(q)}$ is the cost-minimizing solution. We parameterize the different transformations $\tau \in \mathcal{T}$ for encoding messages in datasets by $\tau \circ \mathbf{X}^{(q)} = \mathbf{X}^{(q)} + \mathbf{U}_\tau \mathbf{W}^{(q)}$. Accordingly, the identity transformation is $\mathbf{U}_{\mathrm{id}} = 0$. We will revisit this

definition of the hypothesis space later.

The application of ASC to models with continuous solution space like SVD turns out to be more difficult than the application to clustering problems. The challenges of computing the weight sums Eq. (7.32) and Eq. (7.33) are twofold. First, in a small volume of a continuous hypothesis space, there are infinitely many transformations. Second, transformations can have infinite distance to each other such that the receiver can always distinguish an infinite subset of $\mathcal{T}$ even when the datasets are noisy. This fact renders the union bound in the derivation of the error-probability in Eq. (7.24) inadequate. For these reasons, calculating the capacity under the assumption that the hypothesis space involves any real $N \times k$ matrix $\mathbf{U}$ fails. In the following, we will first demonstrate the effect of this assumption by providing the naïve analytical calculation of the mutual information in Eq. (7.28). Then we introduce constraints on the hypothesis space such that Eq. (7.28) can be computed.

Unconstraint hypothesis space. Here we investigate the mutual information of SVD for an infinite hypothesis space. There are no constraints on the possible transformations used for coding. We analytically compute the weight sums for such a scenario in Appendix B.2 and substitute them to Eq. (7.28). This provides the mutual information

$$\begin{aligned} I(\beta) &= \frac{1}{N}\left(\log\left(\left|\mathcal{Z}^{(1,2)}\right|\right) - \log\left(|\mathcal{Z}_1|\right) - \log\left(|\mathcal{Z}_2|\right)\right) \\ &= \frac{Dk}{2}\log\left(\frac{\beta}{\pi}\right) + \frac{1}{2}\sum_j D_j^{(2)} - \frac{\beta}{4N}\sum_{ij}\left(x_{ij}^{(1)} - x_{ij}^{(2)}\right)^2 . \end{aligned} \tag{7.39}$$

The first order condition provides the optimal computational temperature

$$1/\beta^* = \frac{1}{2NDk}\sum_{ij}\left(x_{ij}^{(1)} - x_{ij}^{(2)}\right)^2 . \tag{7.40}$$

Note that the temperature monotonically increases with the distance of the two datasets, as one would expect: with more noise, the precision for

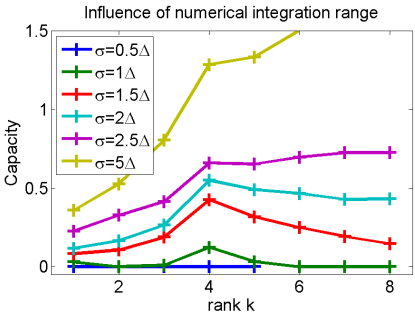


Figure 7.15: Numerically computed approximation capacity for various sizes of the hypothesis space. The optimal rank is $k = 4$. This rank can be determined for spaces with an effective diameter σ that is comparable to the distance Δ between the two cost minimizing solutions for datasets $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$.

approximating the optimal solution must decrease. However, unexpectedly, the temperature decreases with k suggesting that a higher rank always stabilizes the solutions. This misconception is a consequence of the unconstrained hypothesis space, as discussed earlier, and indicates that constraints for $\mathbf{U}$ are necessary. Also, we neglected the temperature-independent term $|\{\mathcal{T}\}|$ in Eq. (7.28) which is infinitely high for an unconstraint continuous hypothesis space.

Finite and bounded hypothesis space. In the discussion above, we identified two problems: an infinitely large capacity due to i) an infinitely large transformation space $\mathbb{R}^{N\times k}$ (or a negative one if we disregard the infinitely many possible codewords in $|\{\mathcal{T}\}|$) and ii) due to the existence of infinitely many transformations in an arbitrary small volume of this space. For a practical implementation of the approximation capacity criterion for SVD we must i) bound the hypothesis space and ii) quantize the set of transformations to a finite set of representative hypotheses. This renders the integrals for computing the weight sums to finite sums which must be explicitly computed.

In our experiments, we use two ways of summing over the hypothesis space. First, the transformations populate the hypothesis space on an equispaced grid in a hypercube of finite size. Second, we randomly sample transformations from an isotropic Gaussian. In both cases the set of transformations is centered around the cost minimizing solution $\mathbf{U}^{(1)}$ (the identity transformation $\mathbf{U}_{\text{id}}$). For both ways, one must choose the boundaries (the size of the grid or the variance of the Gaussian) as well as the number of transformations. In Appendix B.1 we provide some formulas that are useful for numerical maximization of ASC.

We experimentally investigate the influence of the integration range and the number of sample points. First, we study the influence of the integration range on the capacity. We create data $\mathbf{X}^{(q)}$ from a mixture of 4 Gaussians with isotropic noise, leading to an optimal rank 4. We compute the approximation capacity by sampling transformations from a Gaussian sphere around the cost-minimizing SVD solution $\mathbf{U}^{(q)}$ with standard deviation σ. Our experimental findings for various magnitudes of σ are illustrated in Figure 7.15. We write σ in units of $\Delta := 1/N \sum_{i=1}^{N} \left\| \tilde{\mathbf{u}}_i - \mathbf{u}_i^{(1)} \right\|$, where $\tilde{\mathbf{U}}$ is the matrix that satisfies $\mathbf{X}^{(2)} = \tilde{\mathbf{U}}\mathbf{W}^{(1)}$. In the regime where $1/N \left\| \mathbf{U}_\tau \right\| \approx \Delta$, a transformed dataset $\tau_j \circ \mathbf{X}^{(2)}$ could possibly be confused with $\mathbf{X}^{(1)}$. When the transformations are smaller than Δ, none of the transformations could possibly be used in a codebook as they are all indistinguishable from the identity transformation. As a result, the obtained capacity is too low. On the contrary, for a too high integration range, the capacity converges to the naïve analytical solution because infinitely many transformations could serve as distinguishable codewords.

The second experiment studies the influence of the number of transformations on the mutual information. This time, we use a grid of fixed size and vary the density of grid points. In order to sufficiently cover the hypothesis space, we increase the number of transformations by a factor of 2 when increasing k. While this increment is still too low to preserve the transformation density, it already imposes a computational challenge. In our experiments with larger datasets, we sample the hypothesis space more sparsely. The influence of the number of trans-

formations is illustrated in Figure 7.18. The results demonstrate that this number only affects the stability of the computation and not the maximum of the capacity.

Continuous and bounded hypothesis space. The numerical experiment on the influence of the number of transformations suggests that for a defined transformation density, the analytical solution should provide the desired result if only the integration range is properly defined. We calculate the mutual information as in Section 7.4.6 but, this time, we weight the integrand with an isotropic Gaussian around the identity $u_{it}^{(1)}, \forall\, i,t$ to suppress the contribution of heavy transformations.

$$\mathcal{Z}^{(q)} = \int_{-\infty}^{\infty} \exp\left(-\beta R_q\right) \exp\left(-\frac{1}{2\sigma}\sum_{i,t}(u_{it}-u_{it}^{*})^2\right) d\mathbf{U},\ q \in \{1,2\} \tag{7.41}$$

$$\mathcal{Z}^{(1,2)} = \int_{-\infty}^{\infty} \exp\left(-\beta R_{\Delta}\right) \exp\left(-\frac{1}{2\sigma}\sum_{i,t}(u_{it}-u_{it}^{*})^2\right) d\mathbf{U} \tag{7.42}$$

The derivation of the mutual information is provided in Appendix B. The simulations depicted in Figure 7.16 illustrate that for an analytically computed mutual information (see Eq. (B.35)), the width σ influences the capacity much more than in the numerical computation (compare with Fig. 7.16). However, if a maximum exists (square markers in Fig. 7.16), it is at the correct rank.

Experiments

We study how well ASC and other model-order selection methods select the appropriate rank for approximating a noisy dataset via rank-limited SVD. We compare with the following methods: 'Laplace' and 'BIC' approximate the marginal likelihood (the evidence) for probabilistic PCA [56]. The first method applies Laplace approximation to the involved integral. The well-known BIC score [68] further assumes that the likelihood exhibits the same sensitivity to all model parameters. This drastically

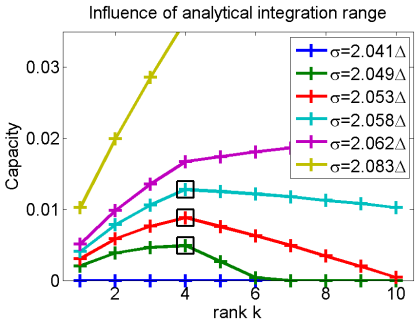


Figure 7.16: Analytically computed approximation capacity integrated over an isotropic Gaussian sphere with varying standard deviation.

simplifies the involved integral. The third method to compare with is the minimum transfer cost principle ('MTC') as introduced in Section 7.3.

We create objects from a number of centroids with a defined separation from each other and add Gaussian noise. The difficulty of the problem is controlled by altering the variance relative to the separation of the centroids. To enable a comparison with the PCA methods, the data mean is shifted to the origin.

For a given true number of generating components and a given noise level relative to the centroid separation, there exists one SVD rank that yields the reconstruction with the minimal deviation from the noise-free matrix. For very noisy data or a high number of components and dimensions, this optimally denoising rank is smaller than the true number of generating components. Inspecting Figure 7.17, one can see that all methods select a rank between the best denoising rank and the true rank. For low noise learning the rank is easy. For a high noise level, the learning problem becomes hard. There exists a transitional regime where all methods start selecting a lower rank than the true one. For very high noise levels, all methods select $k = 1$. In our experiments we zoom in to this transition interval with a high resolution and report the results in Figure 7.17. The left panel of Figure 7.17 depicts the selected rank of all methods and the rank that leads to the smallest distance to

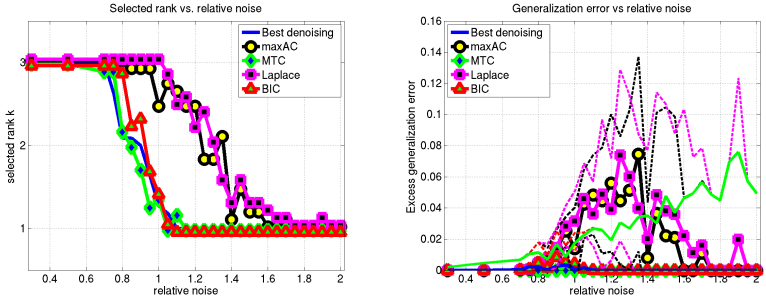


Figure 7.17: Left: Selected rank for data generated from a mixture of three Gaussians. The noise level relative to the separation of the Gaussians varies. The plot zooms in at the noise interval where the different methods provide different choices of k. All methods select a rank between the true number of components and the rank that minimizes the distance to the noise-free matrix ('Best denoising'). For readability, we omitted plotting the variances. Right: Transfer costs of each method relative to minimum transfer costs.

the noise-free matrix ('Best denoising').

The cutoff rank that achieves the lowest denoising error is probably the most favorable one. Except for a transitional phase, all investigated methods select this rank for all learning problems. In the right panel we report the transfer costs of the selected solutions of all methods together with the 85%-percentiles. As the minimum transfer cost principle (MTC) minimizes this score we plot all transfer costs as the difference to the transfer costs of the MTC method. Please note the trend of the 85%-percentile of MTC (solid green line). This provides insight on how sensitive the transfer costs are to the choice of k at a particular noise level. At high variance the transfer costs are rather insensitive to k. As the high variance interval covers the transitional interval of the rank selection, we reason that selecting the rank in this noise interval is a difficult task.

7.4.7 Summary

In this section we provided an introduction to approximation set coding (ASC), a general framework for model selection. It can be used on two levels. i) For a given model, ASC determines the precision of estimating the model parameters that yields the optimal tradeoff between complexity and robustness. The measure that ASC maximizes is the mutual information in a hypothetical communication scenario Eq. (7.28). The optimum is called **approximation capacity**. ii) For a given set of models, ASC selects the model with highest approximation capacity.

We applied ASC to two model selection problems. The first problem is the selection of an appropriate noise model for role mining with the MAC model. We demonstrated that ASC selects the model variant that yields the most accurate parameter estimation for given noisy access-control matrices. The second problem is the selection of the optimal cut-off rank for truncated SVD for denoising Boolean matrices such as, for instance, access-control configurations. The Euclidean geometry and the infinite solution space in this second problem renders the computation of the approximation capacity more difficult than for the first problem. With a suitable sampling scheme and the introduction of a constrained subspace one can solve these technical difficulties.

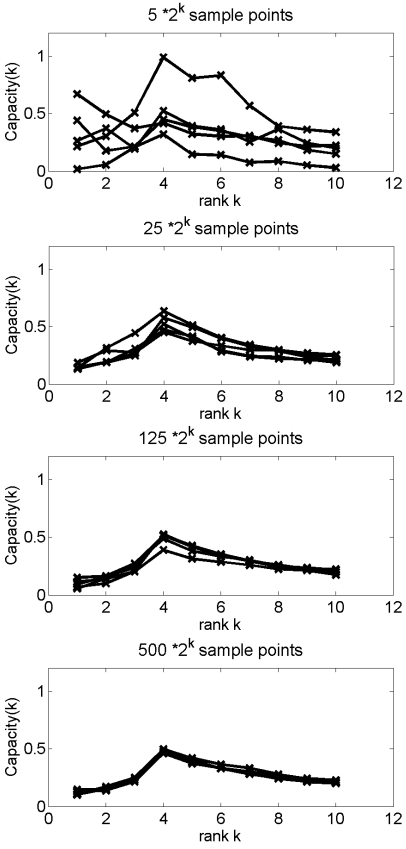


Figure 7.18: Approximation capacity against rank for various numbers of sample points. Even though the capacity varies a lot for a small number of sample points, the optimal model-order is already apparent. With increasing number of transformations the calculations stabilize.

Chapter 8

Conclusion

In this thesis, we described the probabilistic approach to role mining. We first motivated that the goal of role mining is to find the RBAC configuration that underlies the given access-control data. Therefore, we defined role mining as an inference problem. When the true underlying RBAC configuration is known, then one can assess role mining methods based on the accuracy of their parameter estimates. In real-world scenarios, where the true configuration is unknown, the appropriate quality measure is the generalization ability of the inferred RBAC configurations.

We proposed a class of probabilistic models for role mining and investigated several of its model instances. Especially, we analyzed the disjoint-decomposition model (DDM) and the model for multi-assignment clustering (MAC). DMM has a two-layer role hierarchy with the constraint that each user can be assigned to only one role and each permission can be assigned to only one role. MAC has a flat hierarchy and no constraints on the number of roles of a user or a permission. In the direct comparison of these two model instances it turned out that MAC achieves superior generalization ability than DDM in scenarios with high noise while at low noise levels their performance equals.

In experiments with artificially created data and real-world access-

control data, we found that MAC outperforms combinatorial algorithms as well as other probabilistic models in terms of parameter estimation accuracy and in terms of generalization ability. We therefore recommend to use MAC for role mining.

Computing the generalization ability of role mining solutions requires one to define a mapping between the users of two access-control datasets. We generalized this way of transferring solutions between datasets to the class of unsupervised learning problems. Our method, called minimum transfer cost principle (MTC), enables cross-validation for problems where no labels are given. We demonstrated our method on singular-value decomposition (SVD), Gaussian mixture models, correlation clustering, k-means, and role mining with MAC. Based on the simplicity of our method, and based on our experimental results, we believe that MTC will prove useful in many practical applications of unsupervised learning algorithms.

The last part of this thesis concerns approximation set coding (ASC), a novel framework for model validation. For the first time, we applied ASC, to model selection problems such as selecting the noise model for MAC or selecting the cut-off rank for truncated SVD. We consider the application of ASC to well studied problems as a sanity check of this framework. Our experimental findings demonstrate that, in the cases that have been studied, ASC succeeds. This gives hope that the framework is also applicable to many other problems. Due to the generality of the theory of ASC, this framework has the potential to solve validation problems for partially random data that can not be solved yet with established validation techniques. Examples include the selection of the approximation precision of algorithms for combinatorial problems like the traveling salesman problem with noisy travel-times or the knapsack problem with noisy weights. Another application domain is approximate sorting of items with noisy priorities. Application of ASC to such problems is a promising path for future work.

Appendix A

Derivations for the probabilistic model

A.1 Derivation of the data-likelihood

In this Section, we provide the detailed derivation of the data likelihood Eq.(3.16). As follows, we will convert the deterministic formula in Eq. (3.6) into a probabilistic version by marginalizing out the latent variables $\mathbf{u}_{kd}$ from the joint distribution for $\mathbf{u}_{kd}$ and x_{id}.

The joint distribution is

$$p(x_{id} = 0, \mathbf{u}_{*d}|\beta_{*d}, \mathbf{z}_{i*}) = p(x_{id} = 0|\mathbf{u}_{*d}, \mathbf{z}_{i*}) \prod_k p(u_{kd}|\beta_{kd}) \ . \quad \text{(A.1)}$$

Let Ω be the set of all possible binary vectors $\mathbf{u}_{*d}$, then the likelihood

for $x_{id} = 0$ is

$$
\begin{aligned}
& p(x_{id} = 0 | \beta_{*d}, \mathbf{z}_{i*}) && \text{(A.2)} \\
&= \sum_{\mathbf{u}_{*d} \in \Omega} p(x_{id} = 0, \mathbf{u}_{*d} | \beta_{*d}, \mathbf{z}_{i*}) && \text{(A.3)} \\
&= \sum_{\mathbf{u}_{*d} \in \Omega} p(x_{id} = 0 | \mathbf{u}_{*d}, \mathbf{z}_{i*}) \prod_k p(u_{kd} | \beta_{kd}) && \text{(A.4)} \\
&= \sum_{\mathbf{u}_{*d} \in \Omega} \prod_k (1 - u_{kd})^{z_{ik}} \left(\beta_{kd}^{1-u_{kd}} (1 - \beta_{kd})^{u_{kd}} \right) && \text{(A.5)} \\
&= \sum_{\mathbf{u}_{*d} \in \Omega} \left(\prod_{k: z_{ik} = 1} (1 - u_{kd})^{z_{ik}} \, \beta_{kd}^{1-u_{kd}} (1 - \beta_{kd})^{u_{kd}} \right) && \text{(A.6)} \\
&\qquad \cdot \left(\prod_{k: z_{ik} = 0} \beta_{kd}^{1-u_{kd}} (1 - \beta_{kd})^{u_{kd}} \right)
\end{aligned}
$$

In the second last step we substituted the individual probabilities with their definitions Eq. (3.3) and Eq. (3.6). In the last step we separated the bits in $\mathbf{u}_{*d}$ into the two cases where $z_{ik} = 1$ and $z_{ik} = 0$. The first case cancels all contributions of the sum where $z_{ik} = u_{kd} = 1$ and for $z_{ik} = 1, u_{kd} = 0$ only the factor β_{kd} remains. Therefore, it is convenient to introduce a modified set of bitvectors $\Omega' \subset \Omega$ where $\Omega' = \{\mathbf{u}_{*d} \in \Omega \,|u_{kd} = 0 \; \forall k \text{ with } z_{ik} = 1\}$, i.e. the entries of u_{kd} which are relevant for object i are fixed to 0. Then, the likelihood takes a compact form

$$
p(x_{id} = 0 | \beta_{*d}, \mathbf{z}_{i*}) = \sum_{\mathbf{u}'_{*d} \in \Omega'} \left(\prod_{k: z_{ik} = 1} \beta_{kd} \right) \left(\prod_{k: z_{ik} = 0} \beta_{kd}^{1-u'_{kd}} (1 - \beta_{kd})^{u'_{kd}} \right) \tag{A.7}
$$

The sum in Eq. (A.7) has $|\Omega'|$ terms. For a k'' with $z_{ik''} = 0$, one half of these terms has $u_{k''j} = 1$ the other half has $u_{k''j} = 0$, whereas the remaining bits $k \neq k''$ equal in both halves. Therefore, we can factor out the term with k'' in Eq. (A.7). This reduces the number of terms in the sum by a factor of 2 such that the sum reaches now over the modified

set $\Omega'' \subset \Omega'$ where all bits are varied except for k'' or $z_{ik} = 1$. In the following steps we recursively factor out such terms. This successively interchanges the sum and the product in Eq.(A.7) and makes it easy to see (in Eq.(A.10)) that finally all contributions with $z_{ik} = 0$ sum up to 1.

$$p(x_{id} = 0|\beta_{*d}, \mathbf{z}_{i*}) \tag{A.8}$$

$$= \beta_{k''j}^{z_{ik''}} \underbrace{((1 - \beta_{k''j}) + \beta_{k''j})}_{=1 \ \forall k''} \tag{A.9}$$

$$\sum_{\mathbf{u''}_{*d} \in \Omega''} \left(\prod_{\substack{k \,:\, z_{ik} = 1 \\ k \neq k''}} \beta_{kd}^{z_{ik}} \right) \left(\prod_{\substack{k \,:\, z_{ik} = 0 \\ k \neq k''}} \beta_{kd}^{1-u''_{kd}} (1 - \beta_{kd})^{u''_{kd}} \right)$$

$$= \left(\prod_k \beta_{kd}^{z_{ik}} \right) \left(\prod_{k : z_{ik} = 0} \underbrace{\sum_{u'_{kd} \in \{0,1\}} \beta_{kd}^{1-u'_{kd}} (1 - \beta_{kd})^{u'_{kd}}}_{=1 \ \forall k} \right) \tag{A.10}$$

$$= \prod_k \beta_{kd}^{z_{ik}} \tag{A.11}$$

As x_{id} can only take two values, we have $p(x_{id} = 1|\beta_{*d}, \mathbf{z}_{i*}) = 1 - \prod_k \beta_{kd}^{z_{ik}}$ such that the likelihood of the bit x_{id} is

$$p(x_{id}|\beta_{*d}, \mathbf{z}_{i*}) = \left(\prod_k \beta_{kd}^{z_{ik}} \right)^{1-x_{id}} \left(1 - \prod_k \beta_{kd}^{z_{ik}} \right)^{x_{id}} . \tag{A.12}$$

A.2 Derivations for DDM

Here, we derive all necessary distributions for the Gibbs sampling algorithm for DDM in Section 5.2.1. We start by collecting all distributions that define the model. The data likelihood of the model is

$$p\left(\mathbf{X}|\mathbf{Z}, \mathbf{Y}, \boldsymbol{\beta}\right) = \prod_{k,l} (1 - \beta_{kl})^{n_{kl}^{(1)}} \beta_{kl}^{n_{kl}^{(0)}} , \tag{A.13}$$

with the counters

$$n_{kl}^{(1)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \mathbf{I}_{\{x_{ij}=1\}} \ , \tag{A.14}$$

$$n_{kl}^{(0)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \mathbf{I}_{\{x_{ij}=0\}} \ . \tag{A.15}$$

The parameters β_{kl} are random variables themselves. They follow the Beta distribution

$$P_b\left(\beta_{kl};\gamma,\gamma\right) = \frac{\Gamma(2\gamma)}{2\Gamma(\gamma)}\left(\beta_{kl}(1-\beta_{kl})\right)^{\gamma-1} \tag{A.16}$$

$$= B(\gamma,\gamma)^{-1}\left(\beta_{kl}(1-\beta_{kl})\right)^{\gamma-1} \ . \tag{A.17}$$

Thereby, $B(.,.)$ is the beta function, also known as "Euler integral of the first kind".

In each sampling step one must sample new assignments of user i' to roles from the distribution

$$p\left(z_{i'k}=1|\mathbf{X},\mathbf{z}_{i\neq i'*},\mathbf{Y}\right) \quad = \quad \text{const}\cdot p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right)p\left(z_{i'k}=1|\mathbf{z}_{i\neq i'*}\right) \tag{A.18}$$

Sampling assignments from permissions to roles (updating $\mathbf{Y}$) has the same form, with only $\mathbf{Y}$ and $\mathbf{Z}$ interchanged. In order to compute this term for a particular user i' one must compute the evidence term $p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right)$ and the Dirichlet process prior $p\left(z_{i'k}=1|\mathbf{z}_{i\neq i'*}\right)$ for all available roles $k\in\{1,..,K\}$ and for a potential new role with index $K+1$.

We introduced the Dirichlet process prior [5, 22] in Section 5.1.2. Here, we derive the evidence term.

$$p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right) = \int p\left(\mathbf{X},\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y}\right)d\boldsymbol{\beta} \tag{A.19}$$

$$= \frac{p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y},\boldsymbol{\beta}\right)p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y}\right)}{p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y},\mathbf{X}\right)} \tag{A.20}$$

So far, we only applied Bayes' rule. The nominator is the product of terms that we already have: the data likelihood in Eq. (A.13) and the

Beta distributions

$$p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y}\right)=\prod_{k,l}P_b\left(\beta_{kl};\gamma,\gamma\right)\ . \tag{A.21}$$

The term in the denominator is the posterior probability $p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y},\mathbf{X}\right)$ (the probability of $\boldsymbol{\beta}$ *after* having observed the data $\mathbf{X}$). This term is (again with Bayes)

$$p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y},\mathbf{X}\right)=\frac{p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y},\boldsymbol{\beta}\right)\prod_{k,l}P_b\left(\beta_{kl};\gamma,\gamma\right)}{p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right)} \tag{A.22}$$

$$=\text{const}\cdot p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y},\boldsymbol{\beta}\right)\prod_{k,l}P_b\left(\beta_{kl};\gamma,\gamma\right) \tag{A.23}$$

$$=\text{const}\cdot\prod_{k,l}\left(1-\beta_{kl}\right)^{n_{kl}^{(1)}}\beta_{kl}^{n_{kl}^{(0)}}\left(\beta_{kl}(1-\beta_{kl})\right)^{\gamma-1} \tag{A.24}$$

$$=\text{const}\cdot\prod_{k,l}\left(1-\beta_{kl}\right)^{n_{kl}^{(1)}+\gamma-1}\beta_{kl}^{n_{kl}^{(0)}+\gamma-1} \tag{A.25}$$

The constant term serves for normalization. It substitutes $p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right)$ and constant contributions from the Beta distributions. Comparison with the Beta distribution makes it easy to identify how this probability must be normalized. Therefore, we can analytically compute the posterior:

$$p\left(\boldsymbol{\beta}|\mathbf{Z},\mathbf{Y},\mathbf{X}\right)=\prod_{k,l}\frac{\Gamma(n_{kl}^{(1)}+n_{kl}^{(0)}+2\gamma)}{\Gamma(n_{kl}^{(1)}+\gamma)\Gamma(n_{kl}^{(0)}+\gamma)}\left(1-\beta_{kl}\right)^{n_{kl}^{(1)}+\gamma-1}\beta_{kl}^{n_{kl}^{(0)}+\gamma-1} \tag{A.26}$$

$$=\prod_{k,l}B(n_{kl}^{(1)}+\gamma,n_{kl}^{(0)}+\gamma)^{-1}\left(1-\beta_{kl}\right)^{n_{kl}^{(1)}+\gamma-1}\beta_{kl}^{n_{kl}^{(0)}+\gamma-1}\ . \tag{A.27}$$

The Beta distribution is a conjugate prior of the Bernoulli distribution. As a consequence, the posterior of a Bernoulli likelihood and a Beta prior has again the form of a Bernoulli distribution as we just observed in the last derivation.

Substituting the posterior back to Eq. (A.20) results in the analytic expression of the evidence term.

$$p\left(\mathbf{X}|\mathbf{Z},\mathbf{Y}\right) = \prod_{k,l} \frac{B(n_{kl}^{(1)}+\gamma, n_{kl}^{(0)}+\gamma)}{B(\gamma,\gamma)} \tag{A.28}$$

Please note that computation of this term only involves updating the two counters.

A.3 Non-conjugancy of the MAC model with the Beta process

The likelihood of MAC is

$$\begin{aligned} p(x_{id}|\beta_{\cdot,d}, \mathbf{z}_{i,\cdot}) &= \left(\prod_k \beta_{kd}^{z_{ik}}\right)^{(1-x_{id})} \left(1-\prod_k \beta_{kd}^{z_{ik}}\right)^{x_{id}} && \text{(A.29)} \\ &= \beta_{\mathcal{L}d}^{(1-x_{id})} \left(1-\beta_{\mathcal{L}d}\right)^{x_{id}} && \text{(A.30)} \end{aligned}$$

By using proxy sources $\beta_{\mathcal{L}d} = \prod_k \beta_{kd}^{z_{ik}}$, the probability distribution can be represented as in (A.30) which is a Bernoulli distribution with parameter $\beta_{\mathcal{L}d}$.

The Beta distribution is a conjugate prior distribution to the Bernoulli distribution. This implies that a Bernoulli variable with a Beta prior has a posterior probability distribution that can be analytically integrated out.

Assuming that the proxy parameters $\beta_{\mathcal{L}d}$ themselves are drawn from a Beta distribution would allow to integrate out the posterior. However, the proxy parameters are just auxiliary variables that simplify writing down the model. The actual model parameters are the β_{kd}. We must assume that they are individually generated by a Beta distribution. Unfortunately, our model as defined above is not conjugate to the Beta distribution. The problem arises from

$$\left(1-\prod_k \beta_k\right) \neq \prod_k \left(1-\beta_k\right) \tag{A.31}$$

This problem holds for all i and d and all values of z_{ik}. Here, these unnecessary indices are omitted and we just consider general variables $\beta_k \in [0,1]$. As follows, the relation between the two terms in (A.31) is investigated. We would like to express the right-hand side by

$$\prod_k (1-\beta_k) = \left(1 - \prod_k \beta_k\right) + t(k) \tag{A.32}$$

Where $t(K)$ is the deviation from the left-hand side in (A.31).

$$\begin{aligned}
&\prod_k (1-\beta_k) \\
&\quad = (1-\beta_1)(1-\beta_2)..(1-\beta_K) && \text{(A.33)}\\
&\quad = \underbrace{(1-\beta_1+\beta_2-\beta_3+\beta_1\beta_2+\beta_1\beta_3+\beta_2\beta_3-\beta_1\beta_2\beta_3)}_{a_3}(1-\beta_4)..(1-\beta_K) && \text{(A.34)}\\
&\quad = (a_3 - \beta_4 + \beta_1\beta_4 + \beta_2\beta_4 + \beta_3\beta_4 - \beta_1\beta_2\beta_4 - \beta_1\beta_3\beta_4 - \beta_2\beta_3\beta_4 \\
&\qquad +\beta_1\beta_2\beta_3\beta_4) \prod_{k=5}^{K}(1-\beta_k) && \text{(A.35)}\\
&\quad = 1 - \sum_k \beta_k + \sum_k \sum_{k'>k} \beta_k\beta_{k'} - \sum_k \sum_{k'>k} \sum_{k''>k'} \beta_k\beta_{k'}\beta_{k''} + ... && \text{(A.36)}\\
&\quad = 1 + \sum_{q=1}^{K} (-1)^q \sum_{k^{(1)}} \dots \sum_{k^{(q)}>k^{(q-1)}} \prod_{p=1}^{K} \beta_{k^{(p)}} && \text{(A.37)}\\
&\quad = 1 + \sum_{q=1}^{K} (-1)^q a_q \ , && \text{(A.38)}
\end{aligned}$$

with $a_q = \sum_{k^{(1)}} \dots \sum_{k^{(q)}>k^{(q-1)}} \prod_{p=1}^{K} \beta_{k^{(p)}}$. The last term in the sum of Eq.(A.38) is $\pm \prod_k^K \beta_k$ depending on whether K is even or odd. Therefore, it holds that

$$\prod_k (1-\beta_k) = 1 - \prod_k^K \beta_k + t(K) \tag{A.39}$$

with

$$t(K)=\sum_{q=1}^{K}(-1)^q \sum_{k^{(1)}} \dots \sum_{k^{(q)}>k^{(q-1)}} \prod_{p=1}^{K} \beta_{k^{(p)}} + 2\mathbf{I}(K \text{ even}) \prod_{k}^{K} \beta_k \quad \text{(A.40)}$$

This is an alternating series. In principle, one could use the Leibniz criterion in order to show how much one deviates from the original term when omitting all but the first few terms. However, the β_k have unspecified values in $[0, 1]$, such that no precise arguments can be made. If we neglect the cases where all β_k are exactly 0 or 1, the product over all β_k is smaller than their individual values. However, particular combinations of many β_k can still be larger than a particular individual β_k.

Since our motivation for the introduction of continuous β parameters was to model probabilities of binary or Boolean variables we would like to emphasize what the inequality (A.31) means for the original Boolean variables. With binary values u_k the inequality is expressed by

$$\left(1 - \prod_k u_k\right) \neq \prod_k (1 - u_k) \quad \text{(A.41)}$$

Using Boolean variables this corresponds to

$$\neg \bigvee_k (u_k) \neq \bigvee_k \neg (u_k) \ . \quad \text{(A.42)}$$

which can be intuitively understood.

Appendix B

Derivations for approximation set coding

B.1 Numerically maximizing mutual information

There are several ways of numerically maximizing the mutual information in Eq. 7.28 with respect to the temperature. The simplest way is to compute I_β for several values of β and pick the maximum. This creates a quantization error of the optimum that depends on the step-size of the temperature scale.

Using a gradient-descent method like Newton iterations provides more precise results. In the following, we report the first and the second derivation of I_β which are needed for the Newton updates. The derivation of the mutual information I_β (Eq. 7.28) with respect to the inverse

temperature β is

$$\frac{\partial I(\beta)}{\partial \beta} = 1/N \frac{\partial}{\partial \beta} \left(\log\left(|\Delta C_\gamma|\right) - \sum_{q=1}^{2} \log\left(\left|C_\gamma^{(q)}\right|\right) \right) \tag{B.1}$$

$$= 1/N \left(\frac{1}{|\Delta C_\gamma|} \frac{\partial}{\partial \beta} |\Delta C_\gamma| - \sum_{q=1}^{2} \frac{1}{\left|C_\gamma^{(q)}\right|} \frac{\partial}{\partial \beta} \left|C_\gamma^{(q)}\right| \right) \tag{B.2}$$

$$= 1/N \left(\sum_{q=1}^{2} \left(\frac{\int_{-\infty}^{\infty} R_q \exp\left[-\beta R_q\right] d\mathbf{U}}{\int_{-\infty}^{\infty} \exp\left[-\beta R_q\right] d\mathbf{U}} \right) - \frac{\int_{-\infty}^{\infty} R_\Delta \exp\left[-\beta R_\Delta\right] d\mathbf{U}}{\int_{-\infty}^{\infty} \exp\left[-\beta R_\Delta\right] d\mathbf{U}} \right) \tag{B.3}$$

$$= 1/N \left(\sum_{q=1}^{2} \mathbb{E}[R_q]_{p_G(R_q)} - \mathbb{E}[R_\Delta]_{p_G(R_\Delta)} \right) \tag{B.4}$$

Where $p_G(R_\Delta) = Z^{-1} \exp(-\beta R_\Delta)$ is the Gibbs distribution with normalization constant $Z = \int_{-\infty}^{\infty} \exp\left[-\beta R_\Delta\right] d\mathbf{U}$. These integrals can readily be computed either for a finite set of transformations or with a continuous integral.

Accordingly, the second derivative is:

$$\frac{\partial^2 I(\beta)}{\partial \beta^2} = 1/N \left(\sum_{q=1}^{2} \left(\frac{\partial}{\partial \beta} \frac{\int_{-\infty}^{\infty} R_q \exp\left[-\beta R_q\right] d\mathbf{U}}{\int_{-\infty}^{\infty} \exp\left[-\beta R_q\right] d\mathbf{U}} \right) - \frac{\partial}{\partial \beta} \frac{\int_{-\infty}^{\infty} R_\Delta \exp\left[-\beta R_\Delta\right] d\mathbf{U}}{\int_{-\infty}^{\infty} \exp\left[-\beta R_\Delta\right] d\mathbf{U}} \right) \tag{B.5}$$

$$= 1/N \left(\sum_{q=1}^{2} \left(\mathbb{E}[R_q]^2_{p_G(R_q)} - \mathbb{E}[R_q^2]_{p_G(R_q)} \right) \right) - 1/N \left(\mathbb{E}[R_\Delta^2]_{p_G(R_\Delta)} - \mathbb{E}[R_\Delta]^2_{p_G(R_\Delta)} \right) \tag{B.6}$$

B.2 Analytical calculation of mutual information with infinite hypothesis space

In the following calculations, we will use the abbreviations

$$\begin{aligned} R_q &:= R(\mathbf{X}^{(q)}, \mathbf{U}, \mathbf{S}^{(q)}, \mathbf{V}^{(q)}) \\ R_\Delta &:= 1/2 \left(R(\mathbf{X}^{(1)}, \mathbf{U}, \mathbf{S}^{(1)}, \mathbf{V}^{(1)}) + R(\mathbf{X}^{(2)}, \mathbf{U}, \mathbf{S}^{(1)}, \mathbf{V}^{(1)}) \right) \ . \end{aligned} \tag{B.7}$$

The weight sums require an integration over the hypothesis space of our problem. Here, this is the space of all linear combinations $\mathbf{U}$.

$$\begin{aligned} \mathcal{Z}^{(q)} &= \int_{-\infty}^{\infty} \exp\left[-\beta R_q\right] d\mathbf{U} \ , \quad q \in \{1,2\} \\ &= \prod_{ij} e^{-\beta {x_{ij}^{(q)}}^2} \int_{-\infty}^{\infty} e^{-\beta \left(\left(\sum_t^k u_{it} w_{tj}^{(q)} \right)^2 - 2 x_{ij}^{(q)} \sum_t^k u_{it} w_{tj}^{(q)} \right)} d^k \mathbf{u_{i*}} \ , \end{aligned} \tag{B.8}$$

$$\begin{aligned} \mathcal{Z}^{(1,2)} &= \int_{-\infty}^{\infty} \exp\left[-\beta R_\Delta\right] d\mathbf{U} \\ &= \prod_{ij} e^{-\frac{\beta}{2}\left({x_{ij}^{(1)}}^2 + {x_{ij}^{(2)}}^2 \right)} \int_{-\infty}^{\infty} e^{-\beta \left(\left(\sum_t^k u_{it} w_{tj}^{(1)} \right)^2 - \left(x_{ij}^{(1)} + x_{ij}^{(2)} \right) \sum_t^k u_{it} w_{tj}^{(1)} \right)} d^k \mathbf{u_{i*}} \end{aligned} \tag{B.9}$$

These Gaussian integrals can be evaluated analytically which yields the solution

$$\int_{-\infty}^{\infty} e^{-1/2 \sum_t^k \sum_{t'}^k A_{tt'} u_t u_{t'} + \sum_t^k b_t u_t} = \sqrt{\frac{(2\pi)^k}{\det(\mathbf{A})}} e^{1/2 \mathbf{b}^T \mathbf{A}^{-1} \mathbf{b}} \ ,$$

where in our case the solutions are

$$\begin{aligned} \mathbf{A}_{(\mathbf{j})}^{(\mathbf{q})} &= 2\beta \ \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T} \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})} \ , & \mathbf{b}_{(\mathbf{j})}^{(\mathbf{q})} &= 2\beta \ x_{ij}^{(q)} \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T} \ , \\ \mathbf{A}_{(\mathbf{j})}^{(\mathbf{\Delta})} &= \mathbf{A}_{(\mathbf{j})}^{(\mathbf{1})} \ , & \mathbf{b}_{(\mathbf{j})}^{(\mathbf{\Delta})} &= \beta \left(x_{ij}^{(1)} + x_{ij}^{(2)} \right) \mathbf{w}_{*\mathbf{j}}^{(\mathbf{1})^T} \ . \end{aligned} \tag{B.10}$$

Substituting back the solutions of the integrals, the weight sums yield

$$\begin{aligned}\mathcal{Z}^{(q)} &= \prod_{i,j} e^{-\beta x_{ij}^{(q)2}} \sqrt{\frac{(2\pi)^k}{\det\left(2\beta\ \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})}\right)}} e^{1/2\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}} \\ &\overset{(i)}{=} \left(\frac{\pi}{\beta}\right)^{\frac{NDk}{2}} \prod_j \left(D_j^{(q)}\right)^{\frac{-N}{2}} e^{-\beta\sum_{ij}\left(x_{ij}^{(q)2}\left(1-\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T}(\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T})^{-1}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})}\right)\right)} \\ &\overset{(ii)}{=} \left(\frac{\pi}{\beta}\right)^{\frac{NDk}{2}} \prod_j \left(D_j^{(q)}\right)^{\frac{-N}{2}} \qquad \text{(B.11)}\end{aligned}$$

$$\mathcal{Z}^{(1,2)} = \left|C_\gamma^{(1)}\right| \prod_{ij} e^{-\beta/4\left(x_{ij}^{(1)}-x_{ij}^{(2)}\right)^2} . \qquad \text{(B.12)}$$

We abbreviated $D_j^{(q)} := \det(\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})\mathbf{T}}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})})$. In step (i), we used that for a $n \times n$ matrix $\mathbf{M}$ and a scalar p, it holds that $\det(p\mathbf{M}) = p^n \det(\mathbf{M})$. In step (ii), we used that $\mathbf{F}^{(\mathbf{q})} := \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})}(\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})})^{-1}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T} = 1$. Substituting the weight sums to Eq. (7.28) provides the mutual information:

$$\begin{aligned}I(\beta) &= \frac{1}{N}\left(\log\left(\left|\mathcal{Z}^{(1,2)}\right|\right) - \log\left(|\mathcal{Z}_1|\right) - \log\left(|\mathcal{Z}_2|\right)\right) \qquad \text{(B.13)} \\ &= \frac{Dk}{2}\log\left(\frac{\beta}{\pi}\right) + \frac{1}{2}\sum_j D_j^{(2)} - \frac{\beta}{4N}\sum_{ij}\left(x_{ij}^{(1)} - x_{ij}^{(2)}\right)^2 .\end{aligned}$$

The first order condition provides the optimal computational temperature

$$1/\beta^* = \frac{1}{2NDk}\sum_{ij}\left(x_{ij}^{(1)} - x_{ij}^{(2)}\right)^2 . \qquad \text{(B.14)}$$

B.3 Analytical calculation of mutual information with bounded integration range

We derive the mutual information Eq. (B.35) when the transformations are weighted with a Gaussian centered around the identity transformation $u_{it}^{(1)}$. Except for this modification the derivation is analog to the

derivation of the unconstraint mutual information in Eq. (7.39). The approximation sets and the joint approximation set are

$$\left|C_\gamma^{(q)}\right| = \int_{-\infty}^{\infty} e^{-\beta R(\mathbf{X}^{(q)},\mathbf{U},\mathbf{S}^{(q)},\mathbf{V}^{(q)})} e^{-\frac{1}{2\sigma}\sum_{i,t}(u_{it}-u_{it}^{(1)})^2} d\mathbf{U} \;, q \in \{1,2\} \tag{B.15}$$

$$= \prod_{ij} e^{-\beta x_{ij}^{(q)2}} e^{-\frac{1}{2\sigma D}\sum_t u_{it}^{*2}} \tag{B.16}$$

$$\cdot \int_{-\infty}^{\infty} e^{-\beta\sum_t^k u_{it}\left(\frac{1}{\sigma\beta D}u_{it}^{(1)} - 2x_{ij}^{(q)} w_{tj}^{(q)}\right)}$$

$$\cdot e^{-2\beta\sum_t^k u_{it}\left(w_{tj}^{(q)2}u_{it} + 2w_{tj}^{(q)}\sum_{t'\neq t}^k u_{it'}w_{t'j}^{(q)} + \frac{1}{2\sigma\beta D}u_{it}\right)} d^k\mathbf{u}_{\mathbf{i}*}$$

and

$$|\Delta C_\gamma| = \int_{-\infty}^{\infty} e^{-\frac{\beta}{2}\left(R(\mathbf{X}^{(1)},\mathbf{U},\mathbf{S}^{(1)},\mathbf{V}^{(1)}) + R(\mathbf{X}^{(2)},\mathbf{U},\mathbf{S}^{(1)},\mathbf{V}^{(1)})\right)} e^{-\frac{1}{2\sigma}\sum_{i,t}(u_{it}-u_{it}^{(1)})^2} d\mathbf{U} \tag{B.17}$$

$$= \prod_{ij} e^{-\beta/2\left(x_{ij}^{(1)2} + x_{ij}^{(2)2}\right)} e^{-\frac{1}{2\sigma D}\sum_t u_{it}^{*2}} \tag{B.18}$$

$$\cdot \int_{-\infty}^{\infty} e^{-\beta\sum_t^k u_{it}\left(w_{tj}^{(1)2}u_{it} + 2w_{tj}^{(1)}\sum_{t'\neq t}^k u_{it'}w_{t'j}^{(1)} + \frac{1}{2\sigma\beta D}u_{it}\right)}$$

$$\cdot\, e^{-\beta\sum_t^k u_{it}\left((x_{ij}^{(1)} + x_{ij}^{(2)})w_{tj}^{(1)} - \frac{1}{\sigma\beta D}u_{it}^{(1)}\right)} d^k\mathbf{u}_{\mathbf{i}*} \;.$$

The characteristic terms of the integrals are

$$\mathbf{A}_{(\mathbf{j})}^{(\mathbf{q})} = 2\beta\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})}, \tag{B.19}$$

$$\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{q})} = 2\beta\, x_{ij}^{(q)}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T} - \frac{1}{\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})} \;, \tag{B.20}$$

$$\mathbf{A}_{(\mathbf{j})}^{(\mathbf{\Delta})} = \mathbf{A}_{(\mathbf{j})}^{(\mathbf{1})} \;, \tag{B.21}$$

$$\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{\Delta})} = \beta\,\left(x_{ij}^{(1)} + x_{ij}^{(2)}\right)\mathbf{w}_{*\mathbf{j}}^{(\mathbf{1})^T} - \frac{1}{\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})} \;, \tag{B.22}$$

with

$$\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})} := \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})^T}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})} + \frac{1}{2\sigma\beta D}\mathbb{1} \;. \tag{B.23}$$

These terms determine the cardinalities of the approximation sets:

$$\left|C_\gamma^{(q)}\right| = \prod_{i,j} e^{-\beta x_{ij}^{(q)2}} e^{-\frac{1}{2\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T}} \sqrt{\frac{(2\pi)^k}{\det\left(2\beta\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})}\right)}} e^{\frac{1}{2}\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{q})T}\mathbf{A}_{(\mathbf{j})}^{(\mathbf{q})-1}\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{q})}} \tag{B.24}$$

$$= \left(\frac{\pi}{\beta}\right)^{\frac{NDk}{2}} \prod_{i,j} \det\left(\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})}\right)^{-1/2} \tag{B.25}$$

$$e^{-\beta x_{ij}^{(q)2} - \frac{1}{2\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T} + \frac{1}{4\beta}(2\beta\, x_{ij}^{(q)}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})T} - \frac{1}{\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})})^T\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})-1}(2\beta\, x_{ij}^{(q)}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})T} - \frac{1}{\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})})}$$

$$= \left(\frac{\pi}{\beta}\right)^{\frac{NDk}{2}} \left(\prod_j \det\left(\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})}\right)\right)^{-N/2} \tag{B.26}$$

$$\cdot \prod_{i,j} e^{-\beta x_{ij}^{(q)2}\left(1 - \mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})}\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})-1}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{q})T}\right) - \frac{1}{2\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T} - \frac{1}{4\beta\sigma^2 D^2}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{a}_{(\mathbf{j})}^{(\mathbf{q})-1}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T}}$$

and

$$|\Delta C_\gamma| = \prod_{i,j} e^{-\frac{\beta}{2}\left(x_{ij}^{(1)2} + x_{ij}^{(2)2}\right)} e^{\frac{-1}{2\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T}} \sqrt{\frac{(2\pi)^k}{\det\left(2\beta\mathbf{a}_{(\mathbf{j})}^{(\mathbf{1})}\right)}} e^{\frac{1}{2}\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{\Delta})T}\mathbf{A}_{(\mathbf{j})}^{(\mathbf{1})-1}\mathbf{b}_{(\mathbf{i},\mathbf{j})}^{(\mathbf{\Delta})}} \tag{B.27}$$

$$= \left(\frac{\pi}{\beta}\right)^{\frac{NDk}{2}} \prod_j \det\left(\mathbf{a}_{(\mathbf{j})}^{(\mathbf{1})}\right)^{-N/2} \tag{B.28}$$

$$\cdot \prod_{i,j} e^{-\frac{\beta}{2}\left(x_{ij}^{(1)2} + x_{ij}^{(2)2} - \frac{1}{2}\left(x_{ij}^{(1)} + x_{ij}^{(2)}\right)^2 \mathbf{w}_{*\mathbf{j}}^{(\mathbf{1})}\mathbf{a}_{(\mathbf{j})}^{(\mathbf{1})-1}\mathbf{w}_{*\mathbf{j}}^{(\mathbf{1})T}\right)}$$

$$\cdot \prod_{i,j} e^{-\frac{1}{2\sigma D}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T} - \frac{1}{4\beta\sigma^2 D^2}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})}\mathbf{a}_{(\mathbf{j})}^{(\mathbf{1})-1}\mathbf{u}_{\mathbf{i}*}^{(\mathbf{1})T}}$$

The number of random transformations depends on the size of the Gaussian

sphere b :

$$b = \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma}\sum_{i,t}(u_{it}-u_{it}^{(1)})^2} d\mathbf{U} \tag{B.29}$$

$$= \prod_i \sqrt{\frac{(2\pi)^k}{\det(\sigma^{-1}\mathbb{1})}} \tag{B.30}$$

$$= (2\pi\sigma)^{\frac{Nk}{2}} \ . \tag{B.31}$$

Define the auxiliary variables $\mathbf{F}^{(\mathbf{q})}_{(\mathbf{j})} := \mathbf{w}^{(\mathbf{q})}_{*\mathbf{j}} \mathbf{a}^{(\mathbf{q})^{-1}}_{(\mathbf{s})} \mathbf{w}^{(\mathbf{q})^T}_{*\mathbf{j}}$. Then, the mutual information is

$$I(\beta) \tag{B.32}$$

$$=1/N\left(|\{\mathcal{T}\}| + \log(|\Delta C_\gamma|) - \sum_{q=1}^{2}\log\left(\left|C_\gamma^{(q)}\right|\right)\right) \tag{B.33}$$

$$=\frac{k}{2}\log(2\pi\sigma) + \frac{Dk}{2}(\log(\beta) - \log(\pi)) + \frac{1}{2}\sum_j \log\left(\det\left(\mathbf{a}^{(\mathbf{2})}_{(\mathbf{j})}\right)\right) \tag{B.34}$$

$$-\frac{1}{N}\sum_{ij}\left[\frac{\beta}{2}\left(x_{ij}^{(1)2}+x_{ij}^{(2)2}-\frac{1}{2}\left(x_{ij}^{(1)}+x_{ij}^{(2)}\right)^2\mathbf{F}^{(\mathbf{1})}_{(\mathbf{j})}\right)+\frac{1}{2\sigma D}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}+\frac{1}{4\beta\sigma^2D^2}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{a}^{(\mathbf{1})^{-1}}_{(\mathbf{j})}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}\right]$$

$$+\frac{1}{N}\sum_{ij}\left[\beta x_{ij}^{(1)2}\left(1-\mathbf{F}^{(\mathbf{1})}_{(\mathbf{j})}\right)+\frac{1}{2\sigma D}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}+\frac{1}{4\beta\sigma^2D^2}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{a}^{(\mathbf{1})^{-1}}_{(\mathbf{j})}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}\right]$$

$$+\frac{1}{N}\sum_{ij}\left[\beta x_{ij}^{(2)2}\left(1-\mathbf{F}^{(\mathbf{2})}_{(\mathbf{j})}\right)+\frac{1}{2\sigma D}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}+\frac{1}{4\beta\sigma^2D^2}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{a}^{(\mathbf{2})^{-1}}_{(\mathbf{j})}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}\right]$$

$$=\frac{k}{2}\log(2\pi\sigma) + \frac{kD}{2}\log\left(\frac{\beta}{\pi}\right) + \frac{1}{2}\sum_j \log\left(\det\left(\mathbf{a}^{(\mathbf{2})}_{(\mathbf{j})}\right)\right) \tag{B.35}$$

$$+\frac{1}{2\sigma N}\sum_i \mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*} + \frac{1}{4\beta\sigma^2D^2N}\sum_{ij}\mathbf{u}^{(\mathbf{1})}_{\mathbf{i}*}\mathbf{a}^{(\mathbf{2})^{-1}}_{(\mathbf{j})}\mathbf{u}^{(\mathbf{1})^T}_{\mathbf{i}*}$$

$$+\frac{\beta}{2N}\sum_{ij}\left[x_{ij}^{(1)2}\left(1-\frac{3}{2}\mathbf{F}^{(\mathbf{1})}_{(\mathbf{j})}\right)+x_{ij}^{(2)2}\left(1-2\mathbf{F}^{(\mathbf{2})}_{(\mathbf{j})}+\frac{1}{2}\mathbf{F}^{(\mathbf{1})}_{(\mathbf{j})}\right)+x_{ij}^{(1)}x_{ij}^{(2)}\mathbf{F}^{(\mathbf{1})}_{(\mathbf{j})}\right] \ .$$

Appendix C

The Discrete Basis Problem Solver

We provide a description of the Discrete Basis Problem Solver proposed in [54]. A $N \times D$ matrix $\mathbf{X}$ must be approximated by the $N \times K$ matrix $\mathbf{Z}$ and the $K \times D$ matrix $\mathbf{U}$ via the Boolean matrix product $\mathbf{X} = \mathbf{Z} \otimes \mathbf{U}$, for a given K. The rows of $\mathbf{X}$ are interpreted as sets of items (here, users characterized by their sets of permissions), the rows of $\mathbf{U}$ are the basis sets (the roles), and the rows of $\mathbf{Z}$ indicate which basis sets are used to approximate a row of $\mathbf{X}$ (the roles a user is assigned to).

In an initial step, the algorithm computes a set of candidate roles using association rule mining [1]: An $D \times D$ association matrix $\mathbf{A}$ is computed whose entries $A_{j_1 j_2}$ are the pairwise associations between permissions j_1 and j_2: $A_{j_1 j_2} := \langle x_{*j_1}, x_{*j_2} \rangle \,/\, \langle x_{*j_1}, x_{*j_1} \rangle$ with the inner product $\langle ., . \rangle$. In our notation, $A_{j_1 j_2}$ is the empirical probability for a user to have permission j_2 given that he already has permission j_1. Before starting with a greedy algorithm, all entries of $A_{j_1 j_2}$ higher than a threshold τ are set to 1 and the others are set to 0.

Starting from this point, the rows of the role-matrix $\mathbf{U}$ are iteratively filled by the rows of $\mathbf{A}$ and the user-role assignments in $\mathbf{Z}$ are set. In the kth step, both a row from $\mathbf{A}$ is picked as $\mathbf{u}_{k*}$ and the entries of $\mathbf{z}_{*k}$

are set such that the objective function

$$R\left(\mathbf{X},\mathbf{Z},\mathbf{U},w^{+},w^{-}\right)=w^{+}\left|\left\{(i,j):x_{ij}=1\wedge\left(\mathbf{Z}\otimes\mathbf{U}\right)_{ij}=1\right\}\right| - w^{-}\left|\left\{(i,j):x_{ij}=0\wedge\left(\mathbf{Z}\otimes\mathbf{U}\right)_{ij}=1\right\}\right|$$

is maximized. This objective function aims at best covering the existing assignments while avoiding additional assignments. The parameters w^{+} and w^{-} penalize missing assignments and additional assignments respectively.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *Int Conf on Management of Data*, 22(2):207–216, 1993.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[3] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716 – 723, dec 1974.

[4] E. L. Allgower and K. Georg. Simplicial and continuation methods for approximations, fixed points and solutions to systems of equations. *SIAM Review*, 22:28–85, 1980.

[5] Charles E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, November 1974.

[6] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Machine Learning*, pages 238–247, 2002.

[7] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20, 2010.

[8] Joachim M. Buhmann. Information theoretic model validation for clustering. In *ISIT*, 2010.

[9] Joachim M. Buhmann, Morteza Haghir Chehreghani, Mario Frank, and Andreas P. Streich. Information theoretic model selection for pattern analysis. In *JMLR: Workshop and Conference Proceedings 7, 1-8*, 2011.

[10] Joachim M. Buhmann and H. Kühnel. Vector quantization with complexity costs. In *IEEE Trans on Information Theory*, volume 39, pages 1133–1145. IEEE, 1993.

[11] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. A cost-driven approach to role engineering. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2129–2136, New York, NY, USA, 2008. ACM.

[12] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in rbac systems. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, SACMAT '09, pages 85–94, New York, NY, USA, 2009. ACM.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 2nd ed.* MIT Press, 2001.

[14] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

[15] Edward J. Coyne. Role engineering. In *RBAC '95*, page 4, New York, NY, USA, 1996. ACM.

[16] Robert Crook, Darrel Ince, and Bashar Nuseibeh. Towards an analytical role modelling framework for security requirements. In *Proc. of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02)*, pages 9–10, 2002.

[17] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome biology*, 3(7), June 2002.

[18] H.T. Eastment and W.J. Krzanowski. Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24(1):73–77, 1982.

[19] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736 –3745, Dec. 2006.

[20] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *SACMAT '08*, pages 1–10, New York, NY, USA, 2008. ACM.

[21] P. Epstein and R. Sandhu. Engineering of role/permission assignments. In *ACSAC '01*, page 127, Washington, DC, USA, 2001. IEEE Computer Society.

[22] Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973.

[23] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Systems Security*, 4(3):224–274, 2001.

[24] Mario Frank, David Basin, and Joachim M. Buhmann. A class of probabilistic models for role engineering. In *CCS '08*, pages 299–310, New York, NY, USA, 2008. ACM.

[25] Mario Frank and Joachim M. Buhmann. Selecting the rank of truncated svd by maximum approximation capacity. 2011.

[26] Mario Frank, Joachim M. Buhmann, and David Basin. On the definition of role mining. In *SACMAT '10: Proceeding of the 15th*

ACM symposium on Access control models and technologies, pages 35–44, New York, NY, USA, 2010. ACM.

[27] Mario Frank, Morteza Haghir Chehreghani, and Joachim M. Buhmann. The minimum transfer cost principle for model-order selection. In *ECML PKDD '11: Machine Learning and Knowledge Discovery in Databases*, volume 6911 of *Lecture Notes in Computer Science*, pages 423–438. Springer Berlin / Heidelberg, 2011.

[28] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In *CCS '09*, pages 101–111, New York, NY, USA, 2009. ACM.

[29] Ludwig Fuchs and Günther Pernul. Hydro — hybrid development of roles. In *ICISS '08*, pages 287–302, Berlin, Heidelberg, 2008. Springer-Verlag.

[30] K. Gabriel. Le biplot-outil d'exploration de données multidimensionelles. *Journal de la Societe Francaise de Statistique*, 143:5–55, 2002.

[31] Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer, 2005.

[32] Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626. Springer, 2005.

[33] Zoubin Ghahramani, Thomas L. Griffiths, and Peter Sollich. Bayesian nonparametric latent feature models. *Bayesian Statistics 8. Oxford University Press*, pages 201–225, 2007.

[34] James F. Gimpel. The minimization of spatially-multiplexed character sets. *Commun. ACM*, 17(6):315–318, 1974.

[35] Thomas L. Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *Conf on Neural Information Processing Systems*, pages 475–482, 2005.

[36] Qi Guo, Jaideep Vaidya, and Vijayalakshmi Atluri. The role hierarchy mining problem: Discovery of optimal role hierarchies. In *ACSAC '08*, pages 237–246, Washington, DC, USA, 2008. IEEE Computer Society.

[37] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM.

[38] L.K. Hansen and J. Larsen. Unsupervised learning and generalization. In *IEEE International Conference on Neural Networks*, volume 1, pages 25 –30, Jun 1996.

[39] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 2001.

[40] Trevor Hastie, Guenther Walther, and Robert Tibshirani. Estimating the Number of Clusters in a Dataset via the Gap Statistic. *Journal of the Royal Statistical Society, Series B*, 63:411–423, 2000.

[41] Katherine A. Heller and Zoubin Ghahramani. A nonparametric bayesian approach to modeling overlapping clusters. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-2007)*, 2007.

[42] Marcus Hutter and Minh-Ngoc Tran. Model selection with the loss rank principle. *Computational Statistics and Data Analysis*, 54(5):1288 – 1306, 2010.

[43] Tommi S. Jaakkola and Michael I. Jordan. Variational probabilistic inference and the qmr-dt network. *J. Artif. Int. Res.*, 10(1):291–322, 1999.

[44] Ata Kaban and Ella Bingham. Factorisation and denoising of 0-1 data: A variational approach. *Neurocomputing*, 71(10-12):2291 – 2308, 2008.

[45] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Nat Conf on Artificial Intelligence*, pages 763–770, 2006.

[46] Ales Keprt and Václav Snásel. Binary factor analysis with help of formal concepts. In *Proc. of CLA 2004*, pages 90–101, 2004.

[47] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining – revealing business roles for security administration using data mining technology. In *SACMAT '03*, pages 179–186, New York, NY, USA, 2003. ACM.

[48] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[49] Tilman Lange, Volker Roth, Mikio L. Braun, and Joachim M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16:1299–1323, 2004.

[50] Ninghui Li, Tiancheng Li, Ian Molloy, Qihua Wang, Elisa Bertino, Seraphic Calo, and Jorge Lobo. Role mining for engineering and optimizing role based access control systems. Technical report, Purdue University, IBM T.J.Watson Research Center, November 2007.

[51] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. Optimal Boolean matrix decomposition: Application to role engineering. In *ICDE '08*, pages 297–306, Washington, DC, USA, 2008. IEEE Computer Society.

[52] G. Markowsky. Ordering d-classes and computing Schein rank is hard. *Semi-group Forum, 44*, pages 373–375, 1992.

[53] Marc Mézard and Andrea Montanari. *Information, Physics and Computation*. Oxford University Press, Oxford, 2008.

[54] Pauli Miettinen, Taneli Mielikäinen, Aris Gionis, Gautam Das, and Heikki Mannila. The Discrete Basis Problem. In *Proc. of Principles*

and Practice of Knowledge Discovery in Databases, pages 335–346, 2006.

[55] Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In *17th ACMSIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011.

[56] T. P. Minka. Automatic choice of dimensionality for PCA. In *NIPS*, page 514, 2000.

[57] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *SACMAT '08*, pages 21–30, New York, NY, USA, 2008. ACM.

[58] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *SACMAT '09*, pages 95–104, New York, NY, USA, 2009. ACM.

[59] Ian Molloy, Ninghui Li, Yuan (Alan) Qi, Jorge Lobo, and Luke Dickens. Mining roles with noisy data. In *Proceeding of the 15th ACM symposium on Access control models and technologies*, SACMAT '10, pages 45–54, New York, NY, USA, 2010. ACM.

[60] Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[61] Gustaf Neumann and Mark Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *SACMAT '02*, pages 33–42, New York, NY, USA, 2002. ACM.

[62] B. Owen, Patrick, and O. Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization 1. *Annals of Applied Statistics*, 3(2):564–594, 2009.

[63] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[64] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[65] Haio Roeckle, Gerhard Schimpf, and Rupert Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In *RBAC '00*, pages 103–110, New York, NY, USA, 2000. ACM.

[66] Kenneth Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2239, 1998.

[67] Jürgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *SACMAT '05*, pages 168–176, New York, NY, USA, 2005. ACM.

[68] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 1978.

[69] R. J. Solomonoff. A preliminary report on a general theory of inductive inference. 1960.

[70] Larry J. Stockmeyer. The set basis problem is NP-complete. *Report RC5431, IBM Watson Research*, 1975.

[71] Andreas P. Streich, Mario Frank, David Basin, and Joachim M. Buhmann. Multi-assignment clustering for Boolean data. In *ICML '09*, pages 969–976, New York, NY, USA, 2009. ACM.

[72] Romuald Thion. Découverte automatisée de hiérarchies de rôles pour les politiques de contrôle d'accès. In Hermes, editor, *INFORSID'07*, pages 139–154, May 2007.

[73] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *SACMAT '07*, pages 175–184, New York, NY, USA, 2007. ACM.

[74] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Nabil Adam. Migrating to optimal RBAC with minimal perturbation. In *SACMAT '08*, pages 11–20, New York, NY, USA, 2008. ACM.

[75] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. Roleminer: mining roles using subset enumeration. In *CCS '06*, pages 144–153, New York, NY, USA, 2006. ACM.

[76] Jaideep Vaidya, Vijayalakshmi Atluri, Janice Warner, and Qi Guo. Role engineering via prioritized subset enumeration. *IEEE Transactions on Dependable and Secure Computing*, 99, 2008.

[77] Tomáš Šingliar and Miloš Hauskrecht. Noisy-or component analysis and its application to link analysis. *J. Mach. Learn. Res.*, 7:2189–2213, 2006.

[78] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computing*, 1968.

[79] C. S. Wallace and D. L. Dowe. Minimum message length and kolmogorov complexity. *Computer Journal*, 42:270–283, 1999.

[80] Frank Wood, Thomas L. Griffiths, and Zoubin Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *Conf on Uncertainty in Artificial Intelligence*, pages 536–543, 2006.

[81] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In *SACMAT '07*, pages 139–144, New York, NY, USA, 2007. ACM.

[82] Dana Zhang, Kotagiri Ramamohanarao, Tim Ebringer, and Trevor Yann. Permission set mining: Discovering practical and useful roles. In *ACSAC '08*, pages 247–256, Washington, DC, USA, 2008. IEEE Computer Society.

Acknowledgements

I would like to express my gratitude to my advisors David Basin and Joachim M. Buhmann for their great supervision and for the freedom they gave me at the same time. Also, I'd like to thank Andrea Montanari for serving as a co-referee.

I had the pleasure to work with Claudiu Duma, Gritta Wolf, and Günter Karjoth. Thanks for your collaboration and support!

Thanks to my first room-mate Patrick Pletscher who gave me a kick-start in nonparametric Bayesian models and sampling algorithms. Also, I like to thank Andreas Streich. I greatly benefited from working with him. It was a pleasure to collaborate with Morteza Haghir Chehreghani, a man with no pain and with an impressively bright mind.

Thanks to my many room-mates over the last years and different rooms, Patrick Pletscher, Ludwig Busse, Samuel Müller, Morteza Haghir Chehreghani, Andreas Streich, Eugen Zalinescu, Christian Dax, and Michèle Feltz. Also, I'd like to thank Barbara Geiser and Rita Klute for their friendly support.

I want to thank my parents, Marie's parents, and our brothers and sisters for keeping up contact and for driving the distance from time to time. Thanks to Jonathan and Hanna for their unbiased perspective on the world! Finally, I thank my wife Marie who instructed me in advance not to write sentimental words here. Many such words would be appropriate, though.

.