SCALABLE AND ROBUST
SUPERVISED LEARNING THROUGH
RANDOM APPROXIMATIONS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZÜRICHT
(Dr. sc. ETH Zürich)

presented by

GABRIEL CHRISTOF RAFFAEL KRUMMENACHER

MSc ETH Inf.-Ing., ETH Zürich
born on 26.05.1983
citizen of Zürich, Switzerland

accepted on the recommendation of

Prof. Dr. Joachim M. Buhmann, examiner
Prof. Dr. Thomas Hofmann, co-examiner
Prof. Dr. Cheng Soon Ong, co-examiner

2016
ABSTRACT

Scalable machine learning methods are becoming more and more important due to the growth in the size of the data sets. The large scale of modern machine learning data sets has enabled recent success on many real-world problems. The size of these data sets necessitates innovation in learning methods that can scale to the large number of data points and features.

In this thesis we introduce several novel approximation algorithms to solve different machine learning problems for large-scale data sets. As real-world data is also often noisy or unreliably annotated, we contribute new ways of handling uncertainty in data or label space.

We leverage random projections to approximate important aspects of the data by small representations. These representations allow us to design scalable algorithms for linear regression, stochastic optimization and distributed ridge regression.

First, we propose a method to solve large-scale regression problems approximately by subsampling relevant data points from the data set. To make the method robust against noise or outliers in the measurements, we advocate the use of “influence” as a subsampling weight. We show that this measure allows to distinguish corrupted from non-corrupted points in the data and develop a scalable approximation to it. We theoretically bound the estimation error of our method in different settings. We show theoretically and empirically that our method outperforms state-of-the-art approximation methods for large-scale regression on simulated and real-world data sets.

One of the reasons for the recent success and renaissance of neural networks are stochastic optimization methods, since they enable optimization on massive data sets such as image and speech data. Adaptive stochastic optimization algorithms have proven to be among the most successful methods for training these deep neural networks. We contribute a novel method for adaptive stochastic optimization based on random projections of the gradients, that captures important dependencies in the data or network structure. We show empirically on large-scale learning tasks with convolutional neural networks and recurrent neural networks that our method improves convergence. The-
oretically we show that the regret of our method is bounded by the approximation quality of the random projection and can be significantly lower in a low-rank setting.

For distributed optimization we contribute an algorithm that allows to split the data points across the features over multiple machines. Communicating only a small random projection of the data per machine once, the algorithm has low-communication cost and is able to preserve important dependencies between the features. We bound the approximation error for ridge regression relative to the solution on the full data set and demonstrate practical performance and near-linear speedup.

For large data sets it becomes increasingly expensive to obtain ground truth labels. Collectively annotating a group of data points with one label has proven to be an effective strategy to limit this cost. Within this weak labeling approach, the framework of multiple instance learning has gained particular interest. To make multiple instance learning scalable we propose an ellipsoidal approximation to a group of data points and develop a large margin method for learning with ellipsoids.

As our final contribution we solve the real-world problem of wheel defect detection on freight trains, an important task in train safety monitoring. To this end we develop novel wavelet based time series features and kernel and deep neural network architectures that directly model invariance in the measurements.

In dieser Thesis stellen wir mehrere neue Methoden zum Lösen von verschiedenen Problemen des Maschinellen Lernens für umfangreiche Datensätze vor. Da “real-world” Daten oft verrauscht oder unzuverlässig annotiert sind, steuern wir neue Wege zum handhaben von Unsicherheit im Daten- oder Labelraum bei.

Wir setzen Zufallsprojektionen ein um wichtige Aspekte der Daten durch kleinere Representationen zu approximieren. Das erlaubt uns skalierbare Algorithmen für lineare Regression, stochastische Optimierung und verteilte Ridge-Regression zu entwerfen.


in den Daten oder Netzwerkstruktur erfasst bei, basierend auf Zufallsprojektionen der Gradienten. Wir zeigen empirisch auf umfangreichen Lernaufgaben mit Convolutional Neural Networks und Recurrent Neural Networks, dass unsere Methode die Konvergenz verbessert. Theoretisch zeigen wir, dass der Regret unserer Methode durch die Approximationsqualität der Zufallsprojektion begrenzt ist und in einem Szenario mit niedrigem Rang signifikant kleiner sein kann.


The following publications are included in parts or in an extended version in this thesis:


Christina Heinze, Brian McWilliams, Nicolai Meinshausen, Gabriel Krummenacher, and Hastagiri P. Vanchinathan (2014). „LOCO: Distributing Ridge Regression with Random Projections.” In: NIPS Workshop on Distributed Machine Learning and Matrix Computations.

The following Master’s thesis was conducted in relation to this thesis.


Furthermore, the following publication was part of my PhD research, is however not covered in this thesis. The topics of this publication are outside of the scope of the material covered here:

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my supervisor Joachim Buhmann for his advice and support during my PhD studies. I’m grateful for the freedom he gave me to explore research topics of my interest.

Besides my advisor, I would like to thank the rest of my thesis committee Thomas Hofmann and Cheng Soon Ong for agreeing to read my thesis and for providing helpful comments and corrections. I also want to thank Thomas along with Martin Jaggi for organizing the reading groups on random projections and stochastic optimization that sparked my interest in what became two important aspects of this thesis.

My profound thanks go to Cheng Soon Ong for his support, guidance and mentorship. He was an extraordinarily important source of wisdom and advice throughout my PhD and remains to this day.

I would also like to express my deepest gratitude to Brian McWilliams. I profited greatly from his ambition, knowledge and creativity.

My sincere thanks go to my collaborators Stefan Koller at SBB, Mario Lucic, Nicolai Meinshausen, Christina Heinze, Yannic Kilcher, Alberto Busetto and Seijin Kobayashi.

I want to thank all my colleagues in the Information Science and Engineering Group and at the Institute for Machine Learning, that I had the pleasure to work with: Stefan, Dima, Olivier, Alexey, Patrick, Sharon, Hasta, Dave, Sasha, Victor, Nico, Aytunc, Hamed, Carsten, Florian, Aurelien, Andy, Octavian, Paulina, Hadi, Celestine, Jason, Sabrina and Rita.

I thank my parents for their unconditional love and support for anything that I do in life and for babysitting before important deadlines.

Last but most importantly I would like to thank my wife for her amazing support during my whole PhD studies.

To Eliza and Anna.

This thesis was written using a LaTeX template kindly provided by Patrick Pletscher.
CONTENTS

1 INTRODUCTION 1
1.1 Strategies for large-scale data .......................... 2
1.2 Uncertainty .............................................. 4
1.3 Contributions ............................................. 5

2 BACKGROUND 7
2.1 Empirical Risk Minimization .......................... 7
2.2 Maximum Margin Classifiers .......................... 10
2.3 Linear Regression ....................................... 11
2.4 Reproducing Kernel Hilbert Space .......................... 12
  2.4.1 Kernel Embedding of Distributions ................. 13
2.5 Stochastic Optimization .................................. 15
2.6 Random Projections for Machine Learning ....................... 16
  2.6.1 Johnson-Lindenstrauss Lemma .......................... 16
  2.6.2 Fast Random Projections .............................. 18

3 SCALABLE REGRESSION FOR CORRUPTED MODELS 21
3.1 Introduction .............................................. 21
3.2 Statistical model ........................................... 23
3.3 Diagnostics for linear regression .......................... 25
  3.3.1 Statistical leverage ..................................... 26
  3.3.2 Influence ............................................. 27
3.4 Analysis of Leverage and Influence .......................... 28
  3.4.1 Bounds for non-corrupted observations ............... 30
  3.4.2 Bounds for corrupted observations .................... 30
3.5 Related work: Randomized least squares algorithms ............... 31
  3.5.1 Randomized leverage computation ..................... 32
3.6 Proposed method: Influence weighted subsampling .................... 33
  3.6.1 Randomized approximation algorithms .................. 34
3.7 Theoretical analysis ...................................... 36
  3.7.1 Estimation error in the sub-Gaussian model ............ 37
  3.7.2 Corrupted observation model .......................... 38
3.8 Experimental results ...................................... 43
  3.8.1 Non-corrupted data ..................................... 44
  3.8.2 Corrupted data ........................................ 44
  3.8.3 Larger scale experiments on corrupted data ............ 48
  3.8.4 Airline delay dataset ................................. 51
3.9 Conclusions ........................................ 51
Appendix 3.A Proofs for the bounds on leverage and influence 53
Appendix 3.B Theoretical Background .................. 54

4 DISTRIBUTED RIDGE REGRESSION ..................... 57
4.1 Introduction .......................................... 57
4.2 Problem Setting ....................................... 59
4.3 Algorithm ............................................ 62
4.4 Analysis ............................................... 63
4.5 Related work .......................................... 66
4.6 Experimental Results ............................... 67
  4.6.1 Data generation .................................. 67
  4.6.2 Implementation details ........................... 68
  4.6.3 Simulated data ................................... 69
4.7 Discussion ............................................ 73

5 SCALABLE ADAPTIVE STOCHASTIC OPTIMIZATION ...... 75
5.1 Introduction .......................................... 76
  5.1.1 Related work ..................................... 77
  5.1.2 Problem setting ................................... 78
5.2 Stochastic optimization in high dimensions .......... 79
5.3 Approximating ADA-FULL using random projections . 82
  5.3.1 Randomized low-rank approximation ............... 82
  5.3.2 Proof of Theorem 5.3 ............................. 83
  5.3.3 Computational complexity of ADA-LR ............. 87
  5.3.4 RadaGrad: A faster approximation ................ 87
  5.3.5 Practical algorithms .............................. 88
5.4 Experiments on low rank data ....................... 91
  5.4.1 Effective low-rank data .......................... 91
  5.4.2 Non-convex optimization in neural networks .... 92
5.5 Discussion ............................................ 96

6 ELLIPTOIDAL MULTIPLE INSTANCE LEARNING .......... 97
6.1 Introduction .......................................... 97
6.2 Background .......................................... 98
  6.2.1 Robust Classification ............................. 98
  6.2.2 Multiple Instance Learning ....................... 99
6.3 Detection with ellipsoids ........................... 101
  6.3.1 Optimization problem ............................ 101
  6.3.2 Ellipsoid estimation ............................. 105
  6.3.3 Confidence regions .............................. 106
  6.3.4 A Kernel Test for Normality ..................... 106
CONTENTS

6.3.5 Probabilistic multiple instance learning .......................... 109
6.3.6 Relation to robust classification ................................. 112
6.4 Solving eMIL .............................................................. 113
   6.4.1 Difference of Convex Functions ............................... 113
   6.4.2 Solving eMIL with CCCP .................................... 114
   6.4.3 Solving eMIL with BFGS .................................... 116
6.5 Benchmark data sets .................................................. 117
   6.5.1 Feature space corresponding to kernels .................... 119
   6.5.2 Results ........................................................... 120
6.6 Defective wheel classification ........................................ 120
   6.6.1 Experimental protocol ........................................ 121
   6.6.2 Results on the wheel data .................................... 122
6.7 Discussion ............................................................... 122

7 FREIGHT TRAIN SAFETY MONITORING ................................. 125
7.1 Introduction .............................................................. 126
   7.1.1 Contribution ....................................................... 127
   7.1.2 Related Work ..................................................... 128
7.2 Measurement System and Defect Types .............................. 128
   7.2.1 Wheel Load Checkpoint ...................................... 128
   7.2.2 Preprocessing .................................................... 130
   7.2.3 Railway Wheel Defects ....................................... 135
7.3 Data Sets ................................................................. 136
   7.3.1 Data Set 1: Calibration Run ................................ 137
   7.3.2 Data Set 2: Reprofile Events ............................... 137
7.4 Label ambiguity .......................................................... 138
   7.4.1 True Error Rates ............................................... 140
   7.4.2 Multiclass and Performance Scores ......................... 142
7.5 Time Series Representation for Defect Detection .................... 143
   7.5.1 Wavelet Transform ............................................. 144
   7.5.2 Wavelet Features for Defect Detection .................... 145
   7.5.3 Time Series Kernel ............................................. 146
   7.5.4 Load Normalized Features .................................... 147
   7.5.5 Measurement Site .............................................. 148
   7.5.6 Load ............................................................... 148
   7.5.7 Dynamic Coefficient ........................................... 149
7.6 Automatic Representation Learning .................................... 149
   7.6.1 2-Dimensional Time Series Representation .................. 150
   7.6.2 Convolutional Neural Network ............................... 150
   7.6.3 Cyclic Permutation Network ................................. 152
| Figure 2.1 | The SRHT is composed of a subsampling matrix \( S \), a Walsh-Hadamard matrix \( \Theta \) and a diagonal matrix \( D \) with random signs. It projects a vector \( x \in \mathbb{R}^p \) to \( \tilde{x} \in \mathbb{R}^\tau \). |
| Figure 3.1 | Random rows of the matrix \( X \) are corrupted with probability \( \pi \) by measurement noise \( W \) and result in the observations \( Z \). |
| Figure 3.2 | Comparison of the distribution of the influence and leverage for corrupted and non-corrupted points. The \( \ell_1 \) distance between the histograms is shown in brackets. |
| Figure 3.3 | Gaussian data: Comparison of (a) mean estimation error and (b) root mean square prediction error on Gaussian data for increasing number of sub-sampled points. |
| Figure 3.4 | Student-\( t \) 3 data: Comparison of (a) mean estimation error and (b) root mean square prediction error on the T-3 data set for increasing number of sub-sampled points. |
| Figure 3.5 | Student-\( t \) 1 data: Comparison of (a) mean estimation error and (b) root mean square prediction error on the T-1 data set for increasing number of sub-sampled points. |
| Figure 3.6 | 5% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 5% corruptions. |
| Figure 3.7 | 10% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 10% corruptions. |
List of Figures

Figure 3.8  30% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 30% corruptions. .............................. 47

Figure 3.9  5% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 5% corruptions. .............................. 49

Figure 3.10 10% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 10% corruptions. .............................. 49

Figure 3.11 30% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 30% corruptions. .............................. 50

Figure 3.12 Airline delay: Comparison of the root mean square prediction error on the airline delay data set for increasing number of sub-sampled points. .............................. 50

Figure 4.1 Schematic for the approximation of a large data set in a distributed fashion using random projections. The random features can either be concatenated or added. .............................. 61

Figure 4.2 Relative prediction error against time for \( p = 131,072 \). .............................. 69

Figure 4.3 Relative speedup comparison between Loco and HOGWILD! for \( p = 131,072 \). .............................. 70

Figure 4.4 Prediction error against time for \( p = 1.05M \). .............................. 71

Figure 4.5 Relative speedup for \( p = 1.05M \). .............................. 72

Figure 4.6 Relative additional dimensions for \( p = 1.05M \). .............................. 73

Figure 5.1 Comparison of: (a) loss and (b) the largest eigenvalues (normalised by their sum) of the proximal term on simulated data. .............................. 91

Figure 5.2 Comparison of training loss (a) and test accuracy (a) on the MNIST data set. .............................. 92

Figure 5.3 Comparison of training loss (a) and test accuracy (a) on the CIFAR data set. .............................. 92

Figure 5.4 Comparison of training loss (a) and test accuracy (a) on the SVHN data set. .............................. 93
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>Comparison of training loss (a) and and test loss (b) on language modelling task with the LSTM.</td>
</tr>
<tr>
<td>6.1</td>
<td>Derivation of the prediction function. The following three cases are distinguished: (a) ellipsoid is fully in the negative-half space, then the distance is the minimal distance of a point on the ellipsoid to the hyperplane; (b) ellipsoid is fully in the positive half-space, then the distance is the maximal distance from the hyperplane and (c) ellipsoid intersects the hyperplane, then the distance is to the point maximally in the positive half-space. This means that for all three cases the distance is always to the point on the ellipse, that is maximally in the direction of the hyperplane normal vector ( \beta ) and the sign is the sign of the half-space where that point lies in.</td>
</tr>
<tr>
<td>6.2</td>
<td>Two negative (red) and two positive (green) ellipsoids with separating hyperplane and margin, notice the different hyperplane and margin for eMIL (a) and robust SVM (b).</td>
</tr>
<tr>
<td>7.1</td>
<td>Multiple vertical wheel force measurements of a train wheel by the four sensors of one measurement bar. The wheel is affected by a discrete defect that manifests itself in the measurement of the first sensor. The remaining sensors do not directly observe the defect.</td>
</tr>
<tr>
<td>7.2</td>
<td>Diagram of one sensor on a measurement bar of the WLC. The strain gauges are attached to the side of the wheel between two sleepers and cover 28cm of vertical wheel force of the wheel rolling on the track.</td>
</tr>
<tr>
<td>7.3</td>
<td>Two out of four measurement bars, one on the left and one on the right side of the tracks on a Wheel Load Checkpoint.</td>
</tr>
</tbody>
</table>
List of Figures

Figure 7.4 Sensor bias in June for site “Brugg”. Each line gives the bias of each sensor q\textsubscript{1} to q\textsubscript{4} for both sides left and right. Each point is averaged over one week, the number of measurements per average is given in brackets. (b) shows bias of measurements corrected with the bias computed with measurements from March to May. 132

Figure 7.5 Sensor bias in June for site “Gellert”. Each line gives the bias of each sensor q\textsubscript{1} to q\textsubscript{8} for both sides left and right. Each point is averaged over one week, the number of measurements per average is given in brackets. (b) shows bias of measurements corrected with the bias computed with measurements from March to May. 133

Figure 7.6 Speed dependence of the bias between February and April on two different measurement sites. Each line shows the bias at different speeds for all the sensors on each side. The histogram on top of the figures show the number of axles measured for each speed. 134

Figure 7.7 Box plot of the speed dependence (a) and corrected measurements (b) for trains measured on site “Geneva” between March and August 2011. 136

Figure 7.8 Box plot of the speed dependence (a) and corrected measurements (b) for trains measured on site “NH855” between March and August 2011. 137

Figure 7.9 Picture of a serious flat spot on a train wheel of SBB (a) and the resulting idealized wheel load measurement (b). (Picture taken from Wikipedia/Bobo11) 138

Figure 7.10 Signals and wavelet coefficients of a defective (right) and non-defective (left) wheel. The power in the high frequency coefficients D\textsubscript{1}-D\textsubscript{2} reveal the defect. 144

Figure 7.11 Structure of the cyclic permutation network that automatically learns cyclic shift invariant features. 152

Figure 7.12 Top layer filters (a) and features (b) learned by the 1-dimensional defect detection network for flat spots for a measurement of a defective (right) and non-defective (left) wheel. 154
Figure 7.13  Example of a top layer filter (a) and corresponding features of the signal of a non-defective (b) and defective (flat spot) (c) wheel learned on 2D representations. ........................................ 154

Figure 7.14  Structure of the MIL defect detection network for flat spots. The network consists of one CNN per measurement with weights shared across the networks. The defect likelihood of the whole wheel is given by the maximum defect likelihood across sensors. ........................................ 156

LIST OF TABLES

Table 5.1  Comparison of computational complexity in big O notation between ADA-FULL, ADA-LR and RADA-GRAD. .................................................. 88

Table 6.1  Classification accuracy on the MUSK data sets (top block), image annotation (middle block), and TREC9 data. See Section 6.2.2 for description of previous approaches. Performance of those were obtained from the respective papers. ....... 118

Table 6.2  Classification accuracy on the wheel defect data set. Rank gives the average rank. If two methods are equal, they both get the same rank. See text for details. .................................................. 121

Table 7.1  Results on data set 1 ........................................ 158

Table 7.2  Results on data set 2 ........................................ 159

Table 7.3  Performance of the deep models on flat spots in data set 2. Averaged over 3 folds for the deep models, and 10 folds for the SVM. .................. 161

Table 7.4  Performance of the deep models on non-roundness in data set 2. Averaged over 3 folds. ........... 162
INTRODUCTION

Machine learning data sets are growing rapidly in size. Technological advances have made it cheap and easy to collect vast amounts of data. This growth in available data has enabled recent successes in image classification, speech recognition, and reinforcement learning. But the large scale of these data sets also poses challenges for estimation and learning. While traditionally computational and storage restrictions were not an issue for finding predictive models, on modern data sets these limitations are now the main concern when solving large-scale optimization or estimation tasks. These challenges have sparked the research of approximate methods for machine learning.

Additionally to the challenge of learning with large-scale data, real world problems are rarely well behaved in the sense of the data following nice distributional assumptions. Faulty measurement devices might corrupt some of the observed data for instance. Also obtaining annotations for all the measurements is usually a bottleneck.

In the standard supervised learning setting training data comes in pairs consisting of an observation and a label. The task of learning is now to find a function that predicts a label given a new data point, based on annotated training examples. When the training examples are sampled independently from the same distribution as the test data, increasing the number of annotated training examples reduces the error on the test data under some assumptions on the complexity of the function class.

Deviations from the ideal fully supervised setting with its strong model assumptions lead to additional uncertainty in the process of finding reliable predictive models. The effects of noise in the measurements and weak supervision are in addition to error inherent to learning with large data sets due to the necessity of approximations.

1 We will use the terms ‘observation’, ‘data point’ or ‘feature vector’ to refer to the input to the prediction function interchangeably in this thesis.
2 The output of the prediction function is also referred to as ‘annotation’ when part of the training data.
This thesis is concerned with solving (weakly-)supervised learning problems by finding a prediction function from data point to label. This is done via minimizing a loss function on a large-scale training set where either the number of data points or the number of features or both are large. Additionally there is some form of uncertainty or noise in the observations of the data or the labels. We formalize the learning problem and other important concepts in the next chapter (Chapter 2) and the different forms of uncertainty in the respective chapters. In this first chapter we introduce strategies for learning on noisy large-scale data sets.

It is important to note that the algorithms developed in this thesis work on dense data sets. Since this means that the data matrix does not contain large numbers of entries that are zeros, the data sets are truly large when the dimensionality is large. For sparse data sets on the other hand, the number of actual non-zero entries might be low which makes devising parallel optimization methods or adaptive gradient descent easier.

1.1 Strategies for large-scale data

We will look at three main strategies to deal with large-scale data sets: (1) subsampling a smaller, manageable data set; (2) stochastic optimization; (3) distributing the data across multiple machines. In this section we will briefly review these three ways of obtaining an approximate solution to an otherwise infeasible learning task and look at related work in more detail in the respective chapters.

1. To improve the scalability of linear regression, randomized approximation algorithms subsample “typical” data points from the full data set. This smaller data set is then used for least squares estimation. Robustness against outliers is achieved through importance sampling with respect to a statistical leverage measure that quantifies how untypical a data point is with respect to the data distribution. Random projections are used to enable quick estimation of these leverage scores and algorithms based on this idea work well under the assumption of sub-Gaussian or fixed design setting. One problem of this approach is that real world data rarely meets these assumptions. When parts of the data are
corrupted or are outliers with respect to the model, subsampling based on leverage breaks down.

2. One of the most popular approximation algorithms for learning with large-scale data sets is stochastic gradient descent (SGD). Instead of minimizing the loss on the full data set, a random data point is selected at each step in the iteration and loss is evaluated only with respect to this point. This strategy facilitates fast updates of the model parameters, as the loss on the full data set does not need to be evaluated at each step. Due to methods based on SGD the renaissance of neural networks and recent successes in image classification, self-driving cars and natural language processing have been made possible. One of the main challenges with using SGD still remains the selection of an appropriate learning rate (schedule). The frequent parameter updates based on only a crude estimate of the full loss function lead to high variance in the objective function. For large learning rates this leads to fluctuations around the minimum or even divergence. If the learning rate is set too small, convergence is slow. Adaptive gradient methods have been developed to rectify this problem. They adaptively set the learning rate per parameter proportional to the sum of the squared gradients seen so far. This heuristics empowers these methods to better adapt to the geometry of the loss surface and improves convergence. While these methods work well for sparse data sets, their performance deteriorates on dense data sets when there are correlations between the features or when optimizing neural networks that explicitly correlate the parameters like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

3. The next step in making machine learning on large-scale data sets scalable is to parallelize computation across the compute cores of a single machine or even distribute the data over multiple machines. The latter is a natural setting when a data set is so large that it can not be held by a single machine anymore. An important question in this setting is how the data is distributed and when communication between the machines occurs. For SGD it is natural to distribute data over the observations or rows of the data matrix since independence between the data points is assumed. The distribution over rows however leads to costly synchronization
between the machines to prevent the overwriting of intermediate results. When data is distributed over the features or columns of the data matrix then important dependence between features is lost. Since each machine only has access to a fraction of the features, correlation between features on different machines can not be captured and thus simply combining the results of the individual machines is not possible.

1.2 Uncertainty

In addition to the noise due to the approximations described above that are necessary when solving large-scale learning problems, we will look at other forms of uncertainty in the measurements or annotations in this work.

1. The first form of uncertainty that we encounter is caused by training examples that violate the assumption of being sampled from the same underlying distribution as the test data. When some of the measurements are corrupted by an additional noise the standard methods to solve linear regression become inaccurate. Traditionally to solve this problem the noise of the corruptions is estimated and corrected. In the large-scale setting this becomes infeasible as it would require to solve the full regression problem multiple times.

2. While the acquisition of unlabelled training data points is usually automated, annotating these training examples is still mostly done manually by human experts. This leads to the trade-off of more training data enabling better prediction functions but also incurring higher cost due to the bottleneck of annotating data. This lead to the research on weakly supervised learning, where cheaper but less reliable annotations are obtained. Among this framework multiple instance learning has become especially popular. In this setting whole groups of data points are given a binary label. Since the labels of the individual data points (instances) in the groups are unknown during training this leads to a combinatorial optimization problem that is cumbersome to solve.
To each of the main approximation strategies and models of uncertainty from above we contribute an algorithm that improves performance.

1. We show that under a generalised “error-in-variables” model the influence measure is able to detect corrupted data points and develop a randomized approximation of influence for large-scale learning. Theoretically we show that an algorithm based on subsampling with respect to influence reduces bias and variance compared to the ordinary least squares (OLS) estimate under this model. Two practical algorithms based on subsampling with respect to an approximation to influence or the residual based on random projections are proposed. Empirically we show on simulated and real-world data sets that these algorithms outperform state-of-the-art methods for large-scale linear regression.

2. We propose a low-communication method for distributed large-scale ridge regression where the data is distributed across the features. The method preserves dependencies between the features on different machines by communicating a small approximation of the data once in the beginning. Due to this the method has a bounded approximation error with respect to the exact ridge solution on the full data set. Near-linear speed up and good predictive accuracy of the method on large-scale simulated data is demonstrated.

3. We contribute two adaptive SGD methods that are able to capture dependencies between features. Our methods work by incorporating second-order information with a time-varying proximal term similar to the popular AdaGrad algorithm. Instead of a diagonal approximation to the gradient outer product we consider a low-rank approximation based on random projections. The error added to the regret due to this approximation is bounded and for effective low-rank data this results in a regret bound with smaller dependence on the dimensionality compared to the diagonal approximation. We show extensive experimental evidence that our algorithm outperforms AdaGrad for training CNNs and RNNs on image classification and language modelling. We also
combine our algorithm with variance reduction to achieve even better results.

4. We propose a novel method for multiple instance learning (MIL) where we approximate a bag of instances by an ellipsoid and generalise the standard Support Vector Machine (SVM). This allows us to simplify the combinatorial optimization problem inherent to MIL. Two ways of obtaining optimal scaling factors for the ellipsoids based on different assumptions on the instance distributions are shown. The ellipsoid approximation is the most robust solution for any instance distribution with finite mean and covariance. Our method outperforms state-of-the art methods on benchmark data sets. In this chapter we also contribute a novel hypothesis test for the normality of a sample based on the kernel mean map.

5. We solve the real-world problem of wheel defect detection on freight trains with two novel machine learning methods based on multiple vertical force measurements of the trains running in full operational speed. Time series features based on multi-scale wavelet decomposition are developed and generalised to a time series wavelet kernel. We also design novel Deep Neural Networks (DNNs) for classification with a sequence of time series data. For each wheel there are multiple measurements and we propose different ways to encode this structure: MIL, a MIL-DNN and a cyclic-shift invariant DNN.

**Structure** In Chapter 2 important concepts and methods that will be used in this thesis are explained. The next three Chapters (3, 4, 5) all use random projections (Section 2.6) in some way to design scalable algorithms on noisy large-scale problems. In Chapter 3 an algorithm to solve large-scale linear regression under measurement noise is introduced, Chapter 4 deals with distributed ridge regression and Chapter 5 introduces a scalable adaptive stochastic gradient method to train deep neural networks. In the last two chapters (6, 7) the uncertainty in the data is with respect to the label space. In Chapter 6 a scalable method for general multiple instance learning (MIL) is proposed and in Chapter 7 the real-world problem of wheel defect detection is studied.
BACKGROUND

In this chapter important concepts and notation, that are used throughout the thesis, are introduced and discussed. We also formalize the concept of learning through optimization that was discussed in the introduction (Section 1).

First we briefly review the concept of Empirical Risk Minimization (ERM, Section 2.1) as it gives the theoretical motivation to how most of the tasks in the following chapters will be solved. ERM together with the idea of maximum-margin classification (Section 2.2) and kernels (Section 2.4) directly leads to a specific optimization problem called SVM that we will encounter later on in Chapters 6 and 7.

A popular way to solve ERM in large-scale data problems is SGD that we review in Section 2.5 and is a building block of our algorithms in Chapters 4 and 5. In the last section we discuss random projections, powerful dimensionality reduction techniques that are important for the methods in Chapters 3 through 5. We also briefly review the mean map in this chapter (Section 2.4.1) that we will use in Section 6.3.4.

Throughout the thesis vectors will be denoted as bold lowercase letters and matrices as bold capital letters. For more information on notation also see the section on page 183.

2.1 EMPIRICAL RISK MINIMIZATION

At the heart of many machine learning problems lies mathematical optimization. The goal of finding a prediction function that produces little error on unseen test data is usually formalized by a minimization problem of the expected error over the data distribution.

We assume that each data point $x \in \mathbb{R}^p$ and label $y$ is independently distributed according to the unknown joint probability distribution $P(x, y)$. We are mainly looking at two problems: regression where $y$ is a real number and binary classification where $y$ is $+1$ or $-1$. We can now write the expected risk $R(\cdot)$ of the prediction function $f(\cdot)$ as an expectation over this distribution (Vapnik 2013).
Definition 2.1. Expected Risk

\[ R(f) = \mathbb{E}[\ell(y, f(x))]. \]

The simplest form that such a prediction function can take is just a linear combination of all the entries in the vector \( x \). We will parameterize such a linear function by the weights \( \beta \) and offset \( b \).

\[ f(x) = \beta^\top x + b \quad (2.1) \]

In a supervised setting the loss function \( \ell(\cdot, \cdot) \) is a function of the true label sampled from \( P(x, y) \) and the label predicted by \( f(x) \). For classification one of the most common loss functions is the hinge loss (Definition 2.2) that leads to maximum-margin classifiers (Section 2.2). For regression the common loss function is the squared loss (Definition 2.3)

Definition 2.2. Hinge loss
For true label \( y_t \in \{-1, +1\} \) and prediction \( p \in \mathbb{R} \) the hinge loss is defined as

\[ \ell(y_t, p) = \max(0, 1 - y_t \cdot p). \]

Definition 2.3. Squared loss
For true label \( y_t \in \mathbb{R} \) and prediction \( y_p \in \mathbb{R} \), the squared loss is defined as

\[ \ell_2(y_t, y_p) = (y_t - y_p)^2. \]

To find the best prediction function on data sampled from \( P(x, y) \) one would ideally want to minimize the expected risk \( R(f) \) over all possible measurable functions \( \mathcal{G} \).

\[ f^* = \arg \min_{f \in \mathcal{G}} R(f). \quad (2.2) \]

The optimum \( f^* \) is the function with the minimal expected risk.

Since the distribution \( P(x, y) \) is unknown, the usual approach is to instead look at the empirical risk (Definition 2.4) over a fixed set of \( n \) data points: the training set \((X, y)\) with \( X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n \) or \( y \in \{-1, +1\}^n \).

Definition 2.4. Empirical Risk

\[ \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_{it}, f(x_i)) \]
If the data points in the training set are sampled independently from $P(x, y)$ this is an unbiased estimate of the expected risk.

The prediction function that minimizes this empirical risk is called the empirical risk minimizer:

$$\hat{f}_n^* = \arg \min_{f \in \mathcal{F}} \hat{R}_n(f).$$ \hfill (2.3)

Here we restrict the optimization problem to a family of functions $\mathcal{F}$. This restriction prevents overfitting to the training set. In the following we also use the minimal risk of a function from this family with the respective minimizer given by

$$f^*_\mathcal{F} = \arg \min_{f \in \mathcal{F}} R(f).$$ \hfill (2.4)

These two deviations from the original goal of minimizing the risk over the underlying data distribution introduce an excess error over the risk of the best possible function $f^*$. This error can be decomposed into the approximation error $\mathcal{E}_{app}$ due to restricting the function class and the estimation error $\mathcal{E}_{est}$ due to minimizing empirical risk instead of expected risk (Bousquet et al. 2008).

$$\mathcal{E} = \mathcal{E}_{app} + \mathcal{E}_{est}$$ \hfill (2.5)

$$= \mathbb{E}[R(f^*_\mathcal{F}) - R(f^*)] + \mathbb{E}[\hat{R}_n(\hat{f}_n^*) - R(f^*_\mathcal{F})]$$ \hfill (2.6)

where the expectation is over the random choice of the training set. The estimation error depends on the number of samples $n$ in the training set and on the capacity of $\mathcal{F}$ (Vapnik et al. 2015). There is a tradeoff in choosing the size of the family $\mathcal{F}$ as larger families have a smaller approximation error but lead to a higher estimation error (Bousquet et al. 2004). To control this tradeoff between $\mathcal{E}_{app}$ and $\mathcal{E}_{est}$ explicitly we introduce a Tikhonov regularizer $\Omega(f)$ that measures the complexity of $f$. Now by minimizing the regularized empirical risk

$$\hat{f}_n^* = \arg \min_{f \in \mathcal{F}} [\hat{R}_n(f) + \lambda \Omega(f)]$$ \hfill (2.7)

the tradeoff between estimation and approximation error can be controlled by the Lagrange multiplier $\lambda$. Usual regularizers are the squared-$\ell_2$-norm $\Omega(f) = \|f\|^2$ or the $\ell_1$-norm $\Omega(f) = |f|$. 

9
Minimizing the regularized empirical risk with the hinge loss, linear classification functions and $\ell_2$-regularization is called a SVM. This widely used classification method can also be derived from maximizing the margin between classes, which we discuss in the next section.

2.2 Maximum Margin Classifiers

Maximum margin classifiers are binary classifiers that separate the data into two half spaces by means of a separating hyperplane with maximum margin (separation between classes). This hyperplane is found by maximizing the distance of each point to the hyperplane and enforcing that the half-spaces induced by the hyperplane are homogeneous with respect to the label of the points. If the hyperplane is given by the normal vector $\beta$ and offset $b$, this is achieved by the following optimization problem, called hard margin SVM (Equation 2.8).

$$\min_{\beta,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(\beta^T x_i + b) \geq 1 \quad \forall i$$

(2.8)

If the data is not linearly separable we instead look at the hinge loss (Definition 2.2), where we set $p = \beta^T x_i + b$ and solve the optimization problem in Equation 2.9, called soft margin SVM. The hinge loss is 0 for points on the correct side of the margin and proportional to the distance from the margin if on the wrong side.

$$\min_{\beta,b} \left[ \frac{1}{n} \sum_{i=1}^{n} \ell \left( y_i, \beta^T x_i + b \right) \right] + \frac{\lambda}{2} \|\beta\|^2$$

(2.9)

The soft-margin SVM optimization problem in Equation 2.9 can be rewritten as a constrained optimization problem with continuous objective (Equation 2.10) by introducing slack variables $\xi_i$.

$$\min_{\beta,b} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2} \|\beta\|^2 \quad \text{s.t.} \quad y_i(\beta^T x_i + b) \geq 1 - \xi_i \quad \forall i \quad 0 \leq \xi$$

(2.10)

Both formulations of a soft margin SVM are going to be useful for the derivation of our ellipsoidal multiple instance learning classifier in Section 6.
To allow for non-linear decision boundaries the *kernel trick* can be employed to non-linearly map the data into a Reproducing Kernel Hilbert Space, which we discuss in Section 2.4.

### 2.3 Linear Regression

Linear regression is used to find predictive models based on linear prediction functions. Given a vector of responses (labels) $y \in \mathbb{R}^n$ and a data matrix $X \in \mathbb{R}^{n \times p}$ of $n$ observations (data points) with $p$ independent variables (features), the model is given by Equation 2.11.

$$ y = X\beta + \epsilon $$

(2.11)

$\beta \in \mathbb{R}^p$ are the unknown parameters of the linear function that we seek to find. $\epsilon \in \mathbb{R}^n$ are random errors from measurements or random fluctuations. The errors $\epsilon$ are assumed follow a joint Normal distribution with zero mean and homoscedastic constant variance with no correlations.

The data points in $X$ are assumed to be exactly known without errors (*fixed design*) or sampled from a known probability distribution (*random design*).

Using the principle of ERM (Section 2.1) with the squared loss (Definition 2.3) for this model (Equation 2.11) leads to the optimization problem in Equation 2.12 called ordinary least squares (*OLS*).

$$ \hat{\beta} = \arg \min_{\beta} \|y - X\beta\|^2 = \left( X^\top X \right)^{-1} X^\top y. $$

(2.12)

Minimizing regularized empirical risk (Equation 2.7) with an $\ell_2$-penalty instead, we get the *Ridge regression* problem in Equation 2.13.

$$ \min_{\beta \in \mathbb{R}^p} L(\beta) := n^{-1} \|y - X\beta\|^2 + \lambda \|\beta\|^2 $$

(2.13)

The first term is the squared error loss and the second term is the ridge penalty which regularizes the size of the coefficient vector according to the regularization parameter $\lambda$. Ridge regression also has a closed-form solution $\hat{\beta}^{rr} = (X^\top X + n\lambda I)^{-1} X^\top y$. 

11
2.4 REPRODUCING KERNEL HILBERT SPACE

A Reproducing Kernel Hilbert Space (RKHS) is defined as a Hilbert space of functions that has a reproducing kernel. We define the notion of a reproducing kernel and other related concepts in the following.

**Definition 2.5. Kernel (Steinwart et al. 2008)**
Let \( \mathcal{X} \) be a non-empty set. A function \( k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \) is a **kernel** on \( \mathcal{X} \) if there exists a map \( \phi : \mathcal{X} \mapsto \mathcal{H} \) to a Hilbert space \( \mathcal{H} \), such that for all \( x, x' \in \mathcal{X} \)

\[
k(x, x') = \langle \phi(x'), \phi(x) \rangle.
\]

\( \phi(\cdot) \) is called the **feature map** and \( \mathcal{H} \) the **feature space** of \( k(\cdot, \cdot) \).

**Definition 2.6. Reproducing kernel (Berlinet et al. 2011)**
Let \( \mathcal{H} \) be a Hilbert space of \( \mathbb{R} \)-valued functions defined on a non-empty set \( \mathcal{X} \). A function \( k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \) is a **reproducing kernel** of \( \mathcal{H} \) if it satisfies

1. \( \forall x \in \mathcal{X} : k(\cdot, x) \in \mathcal{H}, \)
2. \( \forall x \in \mathcal{X}, \forall f \in \mathcal{H} : \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x). \)

The second condition is called the **reproducing property**. Lemma 4.19 in Steinwart et al. (2008) now shows that a reproducing kernel is a kernel and the so called canonical feature map is given by \( \phi(x) = k(\cdot, x) \).

**Definition 2.7. Reproducing kernel Hilbert space (Berlinet et al. 2011)**
\( \mathcal{H} \) is a reproducing kernel Hilbert space if and only if \( \mathcal{H} \) has a reproducing kernel.

The most common kernel function is the Gaussian radial basis function given by Definition 2.8.

**Definition 2.8. Gaussian kernel**

\[
k(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)
\]

When the SVM (Equation 2.10) is written in the Lagrangian dual every dot product can be replaced with a kernel evaluation and a non-linear classification function can be learned (Boser et al. 1992).

In the next sub-section we briefly introduce the kernel mean map, that allows for the mapping of distributions instead of single points into a RKHS.
2.4 REPRODUCING KERNEL HILBERT SPACE

2.4.1 Kernel Embedding of Distributions

The kernel embedding of distributions (mean map) embeds a whole probability distribution into a RKHS. The embedding is given by the following definition.

**Definition 2.9.** Mean map (Song et al. 2013)

\[
m_X := \mathbb{E}_X[\phi(X)] = \int_{\Omega} \phi(x) dP(x),
\]

where \(X\) is a random variable with domain \(\Omega\) and distribution \(P(X)\), \(x\) is an instantiation of \(X\) and \(\phi(\cdot)\) a feature map as in Definition 2.5.

An important property of the mean map is that the mapping is injective under the condition that the kernel is characteristic (Sriperumbudur et al. 2010). This means that two different probability distributions will be mapped to two distinct elements in \(\mathcal{H}\). Many commonly used kernels, like the Gaussian kernel, are characteristic.

To get an empirical estimate of the mean map if we do not have direct access to the distribution \(P(X)\) we can use a finite sample average to estimate its embedding.

**Definition 2.10.** Empirical mean map (Song et al. 2013)

\[
\hat{m}_X := \frac{1}{n} \sum_{i=1}^{n} \phi(x_i).
\]

The empirical mean map \(\hat{m}_X\) converges to the mean map \(m_X\) with a rate of \(O\left(n^{-1/2}\right)\) in RKHS norm \(\|\hat{m}_X - m_X\|_{\mathcal{H}}\) (Smola et al. 2007).

The mean map has been used in many different scenarios. Among them are a generalization of the Support Vector Machine to distributions (Muandet et al. 2012), domain adaptation (Zhang et al. 2013) and distribution regression (Szabó et al. 2014).

The most obvious use of the mean map is a distance-measure between distributions \(P(X)\) and \(Q(Y)\), the maximum mean discrepancy (MMD).

**Definition 2.11.** Maximum mean discrepancy (Gretton et al. 2012) For two random variables \(X\) and \(Y\) defined on the same topological space and distributed according to \(P\) and \(Q\) respectively, the maximum mean discrepancy is given by

\[
MMD^2(P, Q) = \|m_X - m_Y\|_\mathcal{H}^2.
\]
Unlike other measures like the Kullback-Leibler divergence, the MMD does not require density estimation first, but can easily be estimated as an empirical mean in RKHS. The name maximum mean discrepancy comes from the fact that it can be written as maximizing the distance over functions in a RKHS:

\[
MMD(P, Q) = \sup_{\|f\|_H \leq 1} \mathbb{E}_X[f(X)] - \mathbb{E}_Y[f(Y)].
\]  

(2.14)

If the mean maps \(m_X\) and \(m_Y\) in Definition 2.11 are replaced by the empirical mean map (Definition 2.10) the following empirical estimate of the MMD is obtained (Definition 2.12). Here the kernel trick was used by replacing the feature map \(\phi(\cdot)\) with kernel functions. Definition 2.12 thus allows for the explicit computation of a finite sample estimate of the mean discrepancy between two distributions.

**Definition 2.12.** Empirical maximum mean discrepancy (Gretton et al. 2012) For \(n\) samples from \(X\) and \(m\) samples from \(Y\) defined as in Definition 2.11, the empirical maximum mean discrepancy is given by

\[
\hat{MMD}^2(P, Q) = \frac{1}{n^2} \sum_{i,j=1}^{n} k(x_i, x_j) + \frac{1}{m^2} \sum_{i,j=1}^{m} k(y_i, y_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j). 
\]

Based on this empirical estimate of the MMD a test statistics can then be obtained where the null hypothesis is that both samples come from the same distribution. Gretton et al. (2012) call this the kernel two-sample test and proof the following bound on \(\hat{MMD}\).

**Theorem 2.13** (Bound on \(\hat{MMD}\) (Gretton et al. 2012)). For \(P\) and \(Q\) as in Definition 2.11, \(X, X'\) independent random variables distributed according to \(P\) and \(0 \leq k(x, y) \leq K\). Then if \(m = n\) and \(P = Q\),

\[
\hat{MMD}(P, Q) \leq n^{-\frac{1}{2}} \left[ \frac{2 \mathbb{E}_{X,X'}[k(X, X) - k(X, X')] + \varepsilon}{B_1} \right] + \varepsilon 
\]

\[
\leq (2K/n)^{1/2} + \varepsilon.
\]

(2.15)

We will use this bound on the empirical mean map in Section 6.3.4 where we derive a Normality test. There we can compute \(B_1\) in Equation 2.15 explicitly.

In the next section we will again discuss minimizing (regularized-) empirical risk and describe stochastic gradient descent.
2.5 STOCHASTIC OPTIMIZATION

The standard method to find $\hat{f}_n^*$ by solving (2.7) or (2.3) is to use a form of batch gradient descent. We assume that our family of functions $F$ is parameterized by a vector $\beta \in \mathbb{R}^p$ and write the function corresponding to $\beta$ as $f_\beta$. Now the method iteratively computes the gradient of $\hat{R}_n$ with respect to the parameter $\beta$ and moves in the direction of steepest descent.

$$\beta_{t+1} = \beta_t - \eta \nabla_\beta \hat{R}_n(f_\beta)$$

$$= \beta_t - \eta \frac{1}{n} \sum_{i=1}^n \nabla_\beta \ell(y_i, f_\beta(x_i)),$$  \hspace{1cm} (2.16)

where $\eta$ is a learning rate, that potentially depends on $t$. To solve regularized empirical risk minimization (2.7) instead, only the gradient of $\Omega(f_\beta)$ needs to be added to the equation.

Running batch gradient descent (2.16) is often computationally expensive, especially for large training sets, and only gives an approximation to the problem we actually want to solve: minimizing expected risk (2.2). Therefore it might not be necessary to solve empirical risk minimization (2.3) to precision. In fact it has been shown in Bousquet et al. (2008) that for large data sets only solving (2.7) approximately leads to faster convergence of (2.2).

The prototypical stochastic approximation method is stochastic gradient descent (Robbins et al. 1951; Bottou 2010; Bottou et al. 2016), where the gradient with respect to $\beta$ is computed only for one randomly drawn data point $x_i$ at a time and $\beta_{t+1}$ is updated immediately.

**Definition 2.14.** Stochastic Gradient Descent (SGD)

$$\beta_{t+1} = \beta_t - \eta \nabla_\beta \ell(y_i, f_\beta(x_i)),$$  \hspace{1cm} (2.17)

This leads to faster computable iterates, but adds noise to the convergence behaviour. There have been many approaches developed to improve the convergence behaviour of the SGD method (Definition 2.14). One popular direction is to reduce the variance of the updates by either averaging over past gradients (SAGA, (Defazio et al. 2014; Hofmann, Lucchi, Lacoste-Julien, et al. 2015)), or by computing full batch gradients at certain points in the iteration (SVRG, (R. Johnson et al. 2013)). Another way to speed up convergence is to incorporate second order
BACKGROUND

information (Duchi et al. 2011; Schraudolph et al. 2007; Byrd et al. 2016a; Lucchi et al. 2015).

Even though most theoretical properties of SGD are shown for (strongly-) convex functions, the method enjoys considerable popularity for the optimization of non-convex functions. It is the main method to find the parameters of deep neural networks (LeCun et al. 1998).

2.6 RANDOM PROJECTIONS FOR MACHINE LEARNING

This thesis is concerned with solving large-scale learning problems with uncertainty in the data or label space (Chapter 1). In the next three chapters we will look at different problems where both the number of data points $n$ and the number of dimension $p$ are large and there is uncertainty in the data space. In this chapter we will describe random projections, powerful dimensionality reduction techniques that allow us to approximately solve these problems by embedding the data in a lower-dimensional subspace.

More formally, random projections are low-dimensional embeddings $\Pi : \mathbb{R}^p \to \mathbb{R}^\tau$ which preserve – up to a small distortion – the geometry of a subspace of vectors (e.g. Ailon et al. (2009)). We make clear in the following section what is meant by this.

The linear projection $\Pi$ can be applied to a set of data points via matrix-matrix multiplication from the left or the right side. Throughout the data is arranged as a matrix of $n$ row vectors of dimension $p$: $X \in \mathbb{R}^{n \times p}$. Thus applying the random projection from the left side, $\Pi X$ with $\Pi \in \mathbb{R}^{\tau \times n}$, reduces the number of data points to $\tau$ and applying it from the right side, $X \Pi$ with $\Pi \in \mathbb{R}^{p \times \tau}$ reduces the number of features per data point to $\tau$. Depending on the application one or the other (or both) are used.

2.6.1 Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma (W. B. Johnson et al. 1984) is the main theoretical foundation behind random dimensionality reduction. It states that a set of points in a high-dimensional Euclidean space can be linearly projected onto a lower-dimensional space while approximately preserving the pairwise $\ell_2$ distances between the points. The distortion of the distance between the projected points depends on the
2.6 Random Projections for Machine Learning

dimensionality of the lower-dimensional space and logarithmically on
the number of points. Interestingly the quality of the embedding does
not depend on the dimensionality of the original space.

**Lemma 2.1** (Johnson-Lindenstrauss lemma (W. B. Johnson et al. 1984)). For $0 < \epsilon < 1$, a global constant $c$, any set $X$ of $n$ points in $\mathbb{R}^p$ and $\tau > \frac{c \ln(n)}{\epsilon^2}$, there exists a linear map $f : \mathbb{R}^p \to \mathbb{R}^\tau$, such that for all $u, v \in X$:

$$(1 - \epsilon) \|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon) \|u - v\|^2. \quad (2.18)$$

While Lemma 2.1 only guarantees the existence of a linear projection, there is a similar version of the lemma, that directly gives a way on how to quickly find such a projection by choosing a random $\tau$-dimensional subspace.

**Lemma 2.2** (Distributional J-L lemma (W. B. Johnson et al. 1984; Frankl et al. 1988)). For $0 < \epsilon < 1/2$, a global constant $c$, any set $X$ of $m$ points in $\mathbb{R}^p$ and $\tau \geq \frac{c \ln(m)}{\epsilon^2 - \frac{2}{3} \epsilon^3}$, and a random orthonormal projection matrix $\Pi$, the following property holds with probability at least $\frac{1}{2}$ for all $u, v \in X$:

$$(1 - \epsilon) \|u - v\|^2 \leq \|\Pi u - \Pi v\|^2 \leq (1 + \epsilon) \|u - v\|^2. \quad (2.19)$$

Lemma 2.1 is a direct consequence of the following lemma (Lemma 2.3). This can be seen by noting that since the random projection $f(\cdot)$ or $\Pi$ is linear, the distance in the target subspace $(f(u) - f(v))$ can be replaced by the projected distance in the original space $f(x)$, $x = u - v$. Since the statement of Lemma 2.1 is invariant under scaling it can be assumed that $\|x\|_2 = 1$ without loss of generality. Now setting $\delta = \frac{1}{n^2}$ and applying the union bound on all the distances gives the statement of Lemma 2.1. See for instance Matousek (2002) for a detailed proof of Lemma 2.1 using 2.3.

**Lemma 2.3** (W. B. Johnson et al. (1984)). For $0 < \epsilon, \delta < 1/2$ positive integer $\tau$, there exists a distribution $D$ over $\mathbb{R}^{\tau \times p}$ for $\tau = O(\epsilon^{-2} \log(1/\delta))$, such that for any $x \in \mathbb{R}^p$ with $\|x\|_2 = 1$:

$$\mathbb{P}_{\Pi \sim D} \left[ \|\Pi x\|_2^2 - 1 > \epsilon \right] < \delta. \quad (2.20)$$

While Lemma 2.3 holds for any random $\tau$-dimensional subspace, there exist several constructive proofs of Lemma 2.3 with $D$ being a specific distribution. For instance for distributions over matrices with
independent Gaussian or Bernoulli elements (Dasgupta et al. 2003; Indyk et al. 1998).

It follows a sharp concentration bound for a Gaussian random projection (modified from Vempala (2005)).

Lemma 2.4 (Vempala (2005)). For $0 < \epsilon$, $G \in \mathbb{R}^{p \times \tau}$ a matrix with each entry chosen independently from $\mathcal{N}(0,1)$, $\Pi = \frac{1}{\sqrt{\tau}} G^\top$ and $x \in \mathbb{R}^p$.

\[
\mathbb{E}[\|\Pi x\|^2] = \|x\|^2. \tag{2.21}
\]

\[
P[\|\Pi x\|^2 - \|x\|^2 \geq \epsilon \|x\|^2] < 2 e^{-(\epsilon^2 - \epsilon^3) \frac{\tau}{2}}. \tag{2.22}
\]

If we now assume $\|x\| = 1$ and set $\tau = \frac{4 \log(\frac{2}{\delta})}{\epsilon^2 - \epsilon^3}$ we get the statement of Lemma 2.3 with the distribution over matrices with standard Gaussian entries as witness distribution $D$.

As mentioned, it can be shown (Achlioptas 2003; Matousek 2008) that Lemma 2.3 also holds for random projections with each entry $\Pi_{i,j} \sim \mathcal{U}\{+1, -1\}$, where $\mathcal{U}\{+1, -1\}$ is the Rademacher distribution, and for $\Pi_{i,j}$ distributed according to any sub-Gaussian distribution.

2.6.2 Fast Random Projections

The downside of the dense random projections in the previous section is that they have storage and computational complexity of $O(p\tau)$ (Liberty et al. 2008). Therefore alternative fast Johnson-Lindenstrauss transformations (FJLT) have been developed that substantially reduce the time and storage requirements of applying a random projection while still obeying the J-L property (Lemma 2.1).

We concentrate on the class of structured random projections, among which the Subsampled Randomized Hadamard Transform (SRHT) has particularly attractive properties (Tropp 2010; Boutsidis et al. 2012; Halko et al. 2011). We discuss the SRHT in detail in the following section. Note that the SRHT is mainly used for analysis. For implementation we will use a very closely related transform, the Subsampled Randomized Fourier Transform (SRFT) (Section 2.6.2.2) that shares the properties of the SRHT.

2.6.2.1 Subsampled Randomized Hadamard Transform

The SRHT consists of a preconditioning step after which $\tau$ columns of the new matrix are subsampled uniformly at random. In more detail,
it consists of a projection matrix, \( \Pi = \sqrt{\frac{\tau}{\tau}} S \Theta D \) (Halko et al. 2011; Boutsidis et al. 2012) with the definitions:

- \( S \in \mathbb{R}^{\tau \times p} \) is a subsampling matrix.
- \( \Theta \in \mathbb{R}^{p \times p} \) is a normalized Walsh-Hadamard matrix which is defined recursively as
  \[
  \Theta_p = \begin{bmatrix} \Theta_{p/2} & \Theta_{p/2} \\ \Theta_{p/2} & -\Theta_{p/2} \end{bmatrix}, \quad \Theta_2 = \begin{bmatrix} 1 & 1 \\ +1 & -1 \end{bmatrix}.
  \]
  We set \( \Theta = \frac{1}{\sqrt{p}} \Theta_p \) so it has orthonormal columns.
- \( D \in \mathbb{R}^{p \times p} \) is a diagonal matrix whose entries are drawn independently from \{\(-1,1\)\}.

For the Hadamard transform, \( \tau \) must be a power of two but other transforms exist for which similar theoretical guarantees hold and there is no restriction on \( \tau \). See the next section for details on the SRFT that uses the discrete Fourier transform instead of the Hadamard transform.

The SRHT has similar \( \ell_2 \) distance preserving properties as sub-gaussian random projections but has the added benefit of a fast \( O(p\tau) \) matrix-vector product due to the recursive nature of \( \Theta \) (Ailon et al. 2008). Note that the SRHT can also be combined with an i.i.d. Gaussian random vector to obtain a Gaussian matrix with approximately independent entries in the same computational time (Le et al. 2013).

An important result by Tropp (2010) states that the SRHT preserves the geometry of an entire subspace of vectors.
Theorem 2.15 (The SRHT preserves geometry (Tropp 2010; Halko et al. 2011)). Fix an \( p \times k \) matrix \( \mathbf{V} \) with orthonormal columns, and draw a random \( \tau \times p \) SRHT or SRFT matrix \( \mathbf{\Pi} \) where the embedding dimension \( \tau \) satisfies

\[
4 \left[ \sqrt{k} + \sqrt{8 \log(kp)} \right]^2 \log(k) \leq \tau \leq p.
\]

Then, except with probability \( O(k^{-1}) \),

\[
0.40 \leq \sigma_k(\mathbf{\Pi} \mathbf{V}) \quad \text{and} \quad \sigma_1(\mathbf{\Pi} \mathbf{V}) \leq 1.48.
\]

The symbol \( \sigma_j(\cdot) \) denotes the \( j \)th largest singular value of a matrix.

For the SRFT below the same statement holds (Halko et al. 2011).

2.6.2.2 Subsampled Randomized Fourier Transform

The SRFT is defined analogous to the SRHT but instead of using a Hadamard matrix, a discrete Fourier transform matrix is used. We thus set \( \mathbf{\Pi} = \sqrt{\frac{p}{\tau}} \mathbf{S} \mathbf{\Theta} \mathbf{D} \) with \( \mathbf{\Theta} \in \mathbb{R}^{p \times p} \) a unitary discrete Fourier transform (DFT) matrix. This formulations yields very fast implementations using the fast Fourier transform (FFT), for example using the popular FFTW package\(^1\). It is well known that applying the FFT to a \( p \)–dimensional vector can be achieved in \( O(p \log \tau) \) time. In particular, similar structured random projections have gained popularity as a way to speed up (Mahoney 2011) and robustify (McWilliams, Krummenacher, et al. 2014) large-scale linear regression and for distributed estimation (Heinze et al. 2014).

For the SRFT the geometry preserving property also holds (Theorem 2.15 (Halko et al. 2011)). This property is the basis for Theorem 2.15 that will be used in Chapter 5 to prove a regret bound for an algorithm based on the SRFT.

\(^1\) http://www.fftw.org/
SCALABLE REGRESSION FOR CORRUPTED MODELS

Subsampling methods have been recently proposed to speed up least squares estimation in large-scale settings. However, these algorithms are typically not robust to outliers or corruptions in the observed covariates.

The concept of influence that was developed for regression diagnostics can be used to detect such corrupted observations as shown in this chapter\(^1\). This property of influence – for which we also develop a randomized approximation – motivates our proposed subsampling algorithm for large-scale corrupted linear regression which limits the influence of data points since highly influential points contribute most to the residual error. Under a general model of corrupted observations, we show theoretically and empirically on a variety of simulated and real data sets that our algorithm improves over the current state-of-the-art approximation schemes for ordinary least squares.

In this chapter the uncertainty arises from noise in the measurement process. Since a fraction of the data points are corrupted by noise, a naive estimator will be biased. The problems we consider are too large-scale to compute an estimator on the whole data set. Therefore we will use random projections (Section 2.6) to approximate the full data set and embed it in a lower-dimensional space.

3.1 INTRODUCTION

To improve scalability of the widely used ordinary least squares (OLS) algorithm, a number of randomized approximation algorithms have recently been proposed. These methods, based on subsampling the data set, reduce the computational time from $O(np^2)$ to $o(np^2)$\(^2\) where $n$ is the number of data points and $d$ is the dimensionality (Mahoney 2011). Most of these algorithms are concerned with the classical fixed design setting or the case where the data is assumed to be sampled i.i.d. typically from a sub-Gaussian distribution (Dhillon et al. 2013). This is

\(^1\) Parts of this chapter appear in Krummenacher* et al. (2014)

\(^2\) Informally: $f(n) = o(g(n))$ means $g(n)$ grows faster than $f(n)$. 
known to be an unrealistic modelling assumption since real-world data are rarely well-behaved in the sense of the underlying distributions.

We relax this limiting assumption by considering the setting where with some probability, the observed covariates are corrupted with additive noise. This scenario corresponds to a generalised version of the classical problem of “errors-in-variables” in regression analysis which has recently been considered in the context of sparse estimation (Loh et al. 2012). This corrupted observation model poses a more realistic model of real data which may be subject to many different sources of measurement noise or heterogeneity in the data set.

A key consideration for sampling is to ensure that the points used for estimation are “typical”\(^3\) of the full data set. “Typicality” requires the sampling distribution to be robust against outliers and corrupted points. In the i.i.d. sub-Gaussian setting, outliers are rare and can often easily be identified by examining the statistical leverage scores (Definition 3.3) of the data points.

Crucially, in the corrupted observation setting described in Section 3.2, the concept of an outlying point concerns the relationship between the observed predictors and the response. Now, leverage alone cannot detect the presence of corruptions. Consequently, without using additional knowledge about the corrupted points, the OLS estimator (and its subsampled approximations) are biased. This also rules out stochastic gradient descent (SGD) – which is often used for large-scale regression – since convex cost functions and regularizers which are typically used for noisy data are not robust with respect to measurement corruptions.

This setting motivates our use of influence (Definition 3.6) – the effective impact of an individual data point exerts on the overall estimate – in order to detect and therefore avoid sampling corrupted points. We propose an algorithm which is robust to corrupted observations and exhibits reduced bias compared with other subsampling estimators.

**Outline and Contributions.** In Section 3.2 we introduce our corrupted observation model before reviewing the basic concepts of statistical leverage and influence in Section 3.3. In Section 3.5 we briefly review two subsampling approaches to approximating least squares

---

\(^3\) In this work we use “typical” in the statistical sense of not being an outlier with respect to the data distribution. Not to be confused with the information theoretic notion of typicality.
based on structured random projections and leverage weighted importance sampling. Based on these ideas we present influence weighted subsampling (IWS–LS), a novel randomized least squares algorithm based on subsampling points with small influence in Section 3.6.

In Section 3.7 we analyse IWS–LS in the general setting where the observed predictors can be corrupted with additive sub-Gaussian noise. Comparing the IWS–LS estimate with that of OLS and other randomized least squares approaches we show a reduction in both bias and variance. It is important to note that the simultaneous reduction in bias and variance is relative to OLS and randomized approximations which are only unbiased in the non-corrupted setting. Our results rely on novel finite sample characteristics of leverage and influence which we defer to Section 3.4 and on known concentration bounds on sub-Gaussian random variables which we give in Section 3.B. Additionally, in Section 3.7.1 we prove an estimation error bound for IWS–LS in the standard sub-Gaussian model.

Computing influence exactly is not practical in large-scale applications and so we propose two randomized approximation algorithms based on the randomized leverage approximation of Drineas, Magdon-Ismail, et al. (2011). Both of these algorithms run in $o(np^2)$ time which improve scalability in large problems. Finally, in Section 3.8 we present extensive experimental evaluation which compares the performance of our algorithms against several randomized least squares methods on a variety of simulated and real data sets.

### 3.2 Statistical Model

In this chapter we consider a variant of the standard linear model from Section 2.3.

$$y = X \beta + \epsilon,$$  \hfill (3.1)

where $\epsilon \in \mathbb{R}^n$ is a noise term independent of $X \in \mathbb{R}^{n \times p}$. However, rather than directly observing $X$ we instead observe $Z$ where

$$Z = X + UW.$$  \hfill (3.2)

$U = \text{diag}(u_1, \ldots, u_n)$ and $u_i$ is a Bernoulli random variable with probability $\pi$ of being 1. $W \in \mathbb{R}^{n \times p}$ is a matrix of measurement corruptions. The rows of $Z$ therefore are corrupted with probability $\pi$ and not corrupted with probability $(1 - \pi)$. See Figure 3.1 for an illustration.
Figure 3.1: Random rows of the matrix $X$ are corrupted with probability $\pi$ by measurement noise $W$ and result in the observations $Z$.

If we would observe $X$ directly, i.e. there are no corruptions, then the OLS estimate given by Equation (3.3) would be unbiased.

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|^2 = \left( X^T X \right)^{-1} X^T y. \quad (3.3)$$

Under the corrupted model (Equation (3.2)) the OLS solution becomes biased. Traditionally overcoming this bias was attempted by estimating the noise covariance and correcting $\hat{\beta}$ (Fox 1997). This requires multiple iterations of solving a least squares problem and is unfeasible if the data is large-scale as already computing $\hat{\beta}$ once is computationally prohibitive.

**Definition 3.1 (Sub-Gaussian matrix).** A zero-mean matrix $X$ is called sub-Gaussian with parameter $(\frac{1}{n} \sigma^2_X, \frac{1}{n} \Sigma_X)$ if (a) Each row $x_i^T \in \mathbb{R}^p$ is sampled independently and has $\mathbb{E}[x_i; x_i^T] = \frac{1}{n} \Sigma_X$. (b) For any unit vector $v \in \mathbb{R}^p$, $v^T X_i$ is a sub-Gaussian random variable with parameter at most $\frac{1}{\sqrt{p}} \sigma_X$.

We consider the specific instance of the linear corrupted observation model given by Equation (3.1), and Equation (3.2) where

- $X, W \in \mathbb{R}^{n \times p}$ are sub-Gaussian with parameters $(\frac{1}{n} \sigma^2_X, \frac{1}{n} \Sigma_X)$ and $(\frac{1}{n} \sigma^2_w, \frac{1}{n} \Sigma_w)$ respectively,
- $\epsilon \in \mathbb{R}^n$ is sub-Gaussian with parameters $(\frac{1}{n} \sigma^2_e, \frac{1}{n} \sigma^2 \mathbf{I}_n)$,

and all are independent of each other.
The key challenge is that even when $\pi$ and the magnitude of the corruptions, $\sigma_w$ are relatively small, the standard linear regression estimate is biased and can perform poorly (see Section 3.7). Sampling methods which are not sensitive to corruptions in the observations can perform even worse if they somehow subsample a proportion $rn > \pi n$ of corrupted points. Furthermore, the corruptions may not be large enough to be detected via leverage based techniques alone.

The model described in this section generalises the “errors-in-variables” model from classical least squares modelling. Recently, similar models have been studied in the high dimensional ($p \gg n$) setting in Chen et al. (2012), Chen, Caramanis, and Mannor (2013), Chen and Caramanis (2013), and Loh et al. (2012) in the context of robust sparse estimation. The “low-dimensional” ($n > p$) setting is investigated in Chen et al. (2012), but the “big data” setting ($n \gg p$) has not been considered so far.\footnote{Unlike Chen and Caramanis (2013) and Loh et al. (2012) and others we do not consider sparsity in our solution since $n \gg p$.
}

In the high-dimensional problem, knowledge of the corruption covariance, $\Sigma_w$ (Loh et al. 2012), or the data covariance $\Sigma_x$ (Chen and Caramanis 2013), is required to obtain a consistent estimate. This assumption may be unrealistic in many settings. We aim to reduce the bias in our estimates without requiring knowledge of the true covariance of the data or the corruptions, and instead sub-sample only non-corrupted points.

Nevertheless the theoretical results on the estimation error of our algorithms in the sub-Gaussian and corrupted setting build on results from Chen and Caramanis (2013) (and the corresponding extended version (Chen et al. 2012)).

### 3.3 Diagnostics for Linear Regression

In practice, the sub-Gaussian linear model assumption is often violated either by heterogeneous noise or by a corruption model as in Section 3.2. In such scenarios, fitting a least squares model to the full data set is unwise since the outlying or corrupted points can have a large adverse effect on the model fit. Regression diagnostics have been developed in the statistics literature to detect such points (see e.g. Belsley et al. (1981) for a comprehensive overview). Recently Mahoney (2011) proposed
subsampling points for least squares based on their leverage scores. Other recent works suggest related influence measures that identify subspace (McWilliams and Montana 2014) and multi-view (McWilliams et al. 2012) clusters in high dimensional data.

### 3.3.1 Statistical leverage

Remember from Section 2.3 the least squares solution to the standard linear model in Equation (3.1) is given by

$$\hat{\beta} = \left( X^\top X \right)^{-1} X^\top y. \quad (3.4)$$

The “hat matrix” $L$ in Definition 3.2 maps the response $y$ to the fitted values $\hat{y} = X\hat{\beta}$.

**Definition 3.2** (Hat matrix).

$$L = X(X^\top X)^{-1}X^\top \quad (3.5)$$

Since $\hat{y}_i = \sum_j L_{ij}y_j$ and $L$ is symmetric and idempotent it is natural to define the diagonal elements of $L$ as the statistical leverage scores of the $i^{th}$ sample (Definition 3.3). They measure how strongly the $i$-th data point $x_i$ determines the fitted value $\hat{y}_i$.

**Definition 3.3** (Leverage score).

$$l_i = L_{ii} = x_i^\top \left( (X^\top X)^{-1} \right) x_i \quad (3.6)$$

Leverage scores quantify to what extent a particular sample is an outlier with respect to the distribution of $X$. This means that a point with high leverage score is atypical with respect to the distribution of $X$.

The leverage of a data point is directly related to the Mahalanobis distance between the point and the sample mean relative to the sample covariance.

An equivalent definition from Mahoney (2011) which will be useful later, concerns any matrix $U \in \mathbb{R}^{n \times p}$ which spans the column space of $X$ (for example, the matrix whose columns are the left singular vectors of $X$). The statistical leverage scores of the rows of $X$ are the squared row norms of $U$, i.e. $l_i = \|U_i\|^2$. 

---

26
Although the use of leverage can be motivated from the least squares solution in Equation (3.4), the leverage scores do not take into account the relationship between the predictor variables and the response variable y. Therefore, low-leverage points may have a weak predictive relationship with the response and vice-versa. In other words, it is possible for such points to be outliers with respect to the conditional distribution $P(y|X)$ but not the marginal distribution on $X$.

### 3.3.2 Influence

A concept that captures the predictive relationship between covariates and response is influence. Influential points are those that might not be outliers in the geometric sense, but instead adversely affect the estimated coefficients.

One way to assess the influence of a point is to compute the change in the learned model when the point is removed from the estimation step (Belsley et al. 1981). We can compute a leave-one-out least squares estimator by straightforward application of the Sherman-Morrison formula (Equation 3.7). We first show the derivation under the standard model without corruptions.

**Proposition 3.4** (Derivation of $\hat{\beta}_{-i}$ (Belsley et al. 1981)). Defining $e_i = \hat{y}_i - y_i$ and $\Sigma = X^T X$

$$
\hat{\beta}_{-i} = \left(\Sigma - x_i x_i^T\right)^{-1} \left(X^T y - x_i y_i\right)
= \left(\Sigma^{-1} + \frac{\Sigma^{-1} x_i x_i^T \Sigma^{-1}}{1 - l_i}\right) \left(X^T y - x_i y_i\right)
= \hat{\beta} - \Sigma^{-1} x_i \left(y_i + \frac{x_i^T \Sigma^{-1} X^T y - x_i^T \Sigma^{-1} x_i y_i}{1 - l_i}\right)
= \hat{\beta} - \Sigma^{-1} x_i \left(y_i + \frac{\hat{y}_i - l_i y_i}{1 - l_i}\right)
= \hat{\beta} - \Sigma^{-1} x_i \left(y_i + \frac{e_i}{1 - l_i} - \frac{y_i (1 - l_i)}{1 - l_i}\right)
= \hat{\beta} - \Sigma^{-1} x_i e_i \frac{1}{1 - l_i}
$$
Where the first equality comes from the Sherman-Morrison formula (Sherman et al. 1950; Bartlett 1951) given by

\[
\left( A + uv^\top \right)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.
\]  

(3.7)

Under the corrupted model (Equation 3.2) setting we can derive a leave-one-out estimator where the point we removed is corrupted.

**Proposition 3.5** (Derivation of \( \hat{\beta}_{-m} \)). By proposition 3.4. Defining

\[
e_m = \hat{y}_m - y_m = (x_m + w_m)^\top \hat{\beta} - y_m \quad \text{and}
\]

\[
l_m = (x_m + w_m)^\top \Sigma^{-1}(x_m + w_m)
\]

where \( \Sigma = Z^\top Z \), we have that

\[
\hat{\beta}_{-m} = \hat{\beta} - \frac{\Sigma^{-1}(x_m + w_m)e_m}{1 - l_m}.
\]

Defining the influence as the change in expected mean squared error we arrive at Definition 3.6. The expression we use is also called Cook’s distance (Belsley et al. 1981).

**Definition 3.6** (Influence).

\[
d_i = \left( \hat{\beta} - \hat{\beta}_{-i} \right)^\top X^\top X \left( \hat{\beta} - \hat{\beta}_{-i} \right) = \frac{e_i^2 l_i}{(1 - l_i)^2}.
\]

Points with large values of \( d_i \) are those which, if added to the model, have the largest adverse effect on the resulting estimate. Since influence only depends on the OLS residual error and the leverage scores, it can be seen that the influence of every point can be computed at the cost of a least squares fit. In the next section we will see how to approximate both quantities using random projections.

### 3.4 Analysis of Leverage and Influence

Here we provide finite sample bounds for the leverage and influence terms under the sub-Gaussian design and corrupted design models. We also show empirically that the distribution of leverage scores for corrupted points is very similar to the distribution for non-corrupted
3.4 Analysis of Leverage and Influence

(a) Influence (1.1)

Approximate Influence

Corrupted

Non-Corrupted

(b) Leverage (0.1)

Approximate Leverage

Corrupted

Non-Corrupted

Figure 3.2: Comparison of the distribution of the influence and leverage for corrupted and non-corrupted points. The $\ell_1$ distance between the histograms is shown in brackets.
points. The distribution of influence on the other hand differs significantly for corrupted points. We give a theoretical explanation of this phenomenon in Remark 1 and Remark 2.

Figure 3.2 shows the difference in distribution of influence and leverage between non-corrupted points (top) and corrupted points (bottom) for a data set with 30% corrupted points. The distribution of leverage is very similar between the corrupted and non-corrupted points, as quantified by the \( \ell_1 \) difference. This suggests that leverage alone cannot be used to identify corrupted points.

On the other hand, although there are some corrupted points with small influence, they typically have a much larger influence than non-corrupted points.

We give the proofs of the bounds in this section in the appendix in Section 3.A. We also collect relevant Lemmas from related work that we need for our analysis in Section 3.B in the appendix.

3.4.1 Bounds for non-corrupted observations

**Lemma 3.1** (Leverage). The leverage of a non-corrupted point is bounded by

\[
l_i \leq \sigma_x^2 \cdot O \left( \frac{(p/\sqrt{n})^2}{p} \right) \quad (3.8)
\]

where the exact form of the \( O \left( \frac{(p/\sqrt{n})^2}{p} \right) \) term is given by

\[
\left( \frac{p + 2 \log p + 2\sqrt{p \log p}}{\sqrt{n} - C \sqrt{p} - \sqrt{\log p}} \right)^2,
\]

and \( C \) is a constant, see the proof in Section 3.A.

**Lemma 3.2** (Influence). Defining \( E := \|\hat{\beta}_{OLS} - \beta\| \), the influence of a non-corrupted point is

\[
d_i \leq C_i \left( \sigma_x \sigma_e + 2 \sigma_x^2 E \right), \quad (3.9)
\]

where \( C_i = \frac{\|\Sigma^{-1}\|}{1 - l_i} \sqrt{p \log p} \) and \( \Sigma^{-1} = X^T X \).

3.4.2 Bounds for corrupted observations

**Lemma 3.3** (Leverage of corrupted point). The leverage of a corrupted point is bounded by

\[
l_m \leq (\sigma_x^2 + \sigma_w^2) \cdot O \left( \frac{(p/\sqrt{n})^2}{p} \right), \quad (3.10)
\]
where the exact form of the $O\left((p/\sqrt{n})^2\right)$ term is given by

\[
\left(\frac{p + 2\log p + 2\sqrt{p\log p}}{\sqrt{n} - C\sqrt{p} - \sqrt{\log p}}\right)^2,
\]

and $C$ is a constant, see the proof in Section 3.A.

Remark 1 (Comparison of leverage). Comparing this with Equation (3.8), when $n$ is large, the dominant term is $O((p/\sqrt{n})^2)$ which implies that the difference in leverage between a corrupted and non-corrupted point – particularly when the magnitude of corruptions is not large – is small. This suggests that it may not be possible to distinguish between the corrupted and non-corrupted points by only comparing leverage scores.

Lemma 3.4 (Influence of corrupted point). Defining $E := \|\hat{\beta}_{OLS} - \beta\|$, the influence of a corrupted point is

\[
d_m \leq C_m \left(\sigma_x^2 \sigma_w + \sigma_w^2\right)\|\beta\| + \left(\sigma_x^2 + 2\sigma_x \sigma_w + \sigma_w^2\right)E \\
+ \left(\sigma_x + \sigma_w\right)\sigma_e,
\]

where $C_m = 2\left\|\Sigma^{-1}\right\|_1 \sqrt{p} \log p$ and $\Sigma^{-1} = X^\dagger X$.

Remark 2 (Comparison of influence). Here, $C_m$ differs from $C_i$ in Lemma 3.2 only in its dependence on the leverage of a corrupted instead of non-corrupted point and so for large $n$, $C_i \approx C_m$. It can be seen that the influence of the corrupted point includes a bias term similar to the one which appears in Equation (3.20). This suggests that the relative difference between the influence of a non-corrupted and corrupted point will be larger than the respective relative difference in leverage. All of the information relating to the proportion of corrupted points is contained within $\|\hat{\beta}_{OLS} - \beta\|$.

3.5 RELATED WORK: RANDOMIZED LEAST SQUARES ALGORITHMS

We briefly review two randomized approaches to least squares approximation: the importance weighted subsampling approach of Drineas et al. (2006) and the dimensionality reduction approach (Mahoney 2011). The former proposes an importance sampling probability distribution according to which, a small number of rows of $X$ and $y$ are drawn and
used to compute the regression coefficients. If the sampling probabilities are proportional to the statistical leverages, the resulting estimator is close to the optimal estimator (Drineas et al. 2006). We refer to this as LEV–LS.

The dimensionality reduction approach can be viewed as a random projection step followed by a uniform subsampling. The class of Johnson-Lindenstrauss projections – e.g. the SRHT – has been shown to approximately uniformize leverage scores in the projected space. See Section 2.6 for an overview of random projections and their properties in general and Section 2.6.2.1 for a description of the SRHT.

In addition to the results in Section 2.6, the rows of the transformed matrix $\Pi X$ have approximately uniform leverage scores.

To see this consider that $(DHx)_1 = \sum_{i=1}^{n} h_{ii} \xi_i x_i$ where $\xi_i$ are the diagonal entries of $D$. The variance of this zero-mean sum is $n^{-1}$ since entries of $H$ have magnitude $n^{-1/2}$ by design. The magnitude of this sum can then be bounded using Hoeffding’s inequality (see (Tropp 2010)) and since this argument applies to all rows of the transformed matrix, it implies that they have approximately uniform leverage. The SRHT also has the property that rank$(X) = \text{rank}(\Pi X)$ (Tropp 2010).

Uniformly subsampling the rows of the projected matrix proves to be equivalent to leverage weighted sampling on the original data set (Mahoney 2011). We refer to this as SRHT–LS. It is analysed in the statistical setting by Dhillon et al. (2013) who also propose ULURU, a two step fitting procedure which aims to correct for the subsampling bias and consequently converges to the OLS estimate at a rate independent of the number of subsamples (Dhillon et al. 2013).

With $\Pi$ an SRHT projection matrix, the SRHT–LS algorithm solves

$$
\hat{\beta}_{SRHT} = \arg \min_{\beta} \| \Pi y - \Pi X \beta \|^2
$$

which for an appropriate subsampling ratio, $r = \Omega(\frac{p^2}{\rho^2})$ results in a residual error, $\hat{e}$ which satisfies

$$
\| \hat{e} \| \leq (1 + \rho) \| e \|	ag{3.12}
$$

where $e = y - X \hat{\beta}_{OLS}$ is the vector of OLS residual errors (Mahoney 2011).

3.5.1 Randomized leverage computation

Recently, a method based on random projections has been proposed to approximate the leverage scores based on first reducing the dimension-
3.6 PROPOSED METHOD: INFLUENCE WEIGHTED SUBSAMPLING

ality of the data using the SRHT followed by computing the leverage scores using this low-dimensional approximation (Drineas, Magdon-Ismail, et al. 2011; Ma et al. 2014; Drineas, Mahoney, Muthukrishnan, and Sarlós 2011; Drineas et al. 2006).

The leverage approximation algorithm of Drineas, Magdon-Ismail, et al. (2011) uses a SRHT, \( \Pi_1 \in \mathbb{R}^{r_1 \times n} \) to first compute the approximate Singular Value Decomposition (SVD) of \( X \),

\[
\Pi_1 X = U_\Pi X \Sigma_\Pi X V_\Pi^T.
\]

Followed by a second SHRT \( \Pi_2 \in \mathbb{R}^{p \times r_2} \) to compute an approximate orthogonal basis for \( X_R^{-1} = V_\Pi X \Sigma_\Pi^{-1} \Pi_2 \in \mathbb{R}^{p \times r_2} \).

(3.13)

The approximate leverage scores are now the squared row norms of \( \tilde{U} \), \( \tilde{l}_i = \|\tilde{U}_i\|^2 \).

From Mahoney (2011) we derive the following result relating to randomized approximation of the leverage

\[
\tilde{l}_i \leq (1 + \rho_l) l_i,
\]

where the approximation error, \( \rho_l \) depends on the choice of projection dimensions \( r_1 \) and \( r_2 \).

3.6 PROPOSED METHOD: INFLUENCE WEIGHTED SUBSAMPLING

In the corrupted observation model, OLS and therefore the random approximations to OLS described in Section 3.5.1 obtain poor predictions. To remedy this, we propose influence weighted subsampling (IWS-LS) which is described in Algorithm 3.1. IWS-LS subsamples points according to the distribution, \( P_i = c / d_i \) where \( c \) is a normalizing constant so that \( \sum_{i=1}^{n} P_i = 1 \). OLS is then estimated on the subsampled points. The sampling procedure ensures that points with high influence are selected infrequently and so the resulting estimate is less biased than the full OLS solution. Several approaches similar in spirit have previously been
proposed based on identifying and down-weighting the effect of highly influential observations (Welsch 1980).

Obviously, IWS-LS is impractical in the scenarios we consider since it requires the OLS residuals and full leverage scores. However, we use this as a baseline and to simplify the analysis. In the next section, we propose an approximate influence weighted subsampling algorithm which combines the approximate leverage computation of Drineas, Magdon-Ismail, et al. (2011) and the randomized least squares approach of Mahoney (2011).

Algorithm 3.1 Influence weighted subsampling (IWS-LS).

Input: Data: $\mathbf{Z, y}$

1: Solve $\hat{\beta}_{\text{OLS}} = \arg\min_{\beta} \| \mathbf{y} - \mathbf{Z}\beta \|_2^2$
2: for $i = 1 \ldots n$ do
3: \hspace{0.5cm} $e_i = y_i - \mathbf{z}_i^\top \hat{\beta}_{\text{OLS}}$
4: \hspace{0.5cm} $l_i = \mathbf{z}_i^\top (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{z}_i$
5: \hspace{0.5cm} $d_i = e_i^2 l_i / (1 - l_i)^2$
6: end for
7: Sample rows $(\tilde{\mathbf{Z}}, \tilde{\mathbf{y}})$ of $(\mathbf{Z}, \mathbf{y})$ proportional to $d_i^{-1}$
8: Solve $\hat{\beta}_{\text{IWS}} = \arg\min_{\beta} \| \tilde{\mathbf{y}} - \tilde{\mathbf{Z}}\beta \|_2^2$

Output: $\hat{\beta}_{\text{IWS}}$

3.6.1 Randomized approximation algorithms.

Using the ideas from Section 2.6.2.1 and Section 3.5.1 we obtain the following randomized approximation to the influence scores

$$d_i = \frac{\tilde{e}_i^2 \tilde{l}_i}{(1 - \tilde{l}_i)^2}, \quad (3.15)$$

where $\tilde{e}_i$ is the $i^{th}$ residual error computed using the SRHT-LS estimator. Since the approximation errors of $\tilde{e}_i$ and $\tilde{l}_i$ are bounded (inequalities (3.12) and (3.14)), this suggests that our randomized approximation to influence is close to the true influence.

Basic approximation. The first approximation algorithm is identical to Algorithm 3.1 except that leverage and residuals are replaced by their randomized approximations as in Equation (3.15). We refer to this
algorithm as Approximate influence weighted subsampling (aIWS-LS). Full details are given in Algorithm 3.2. Note that Step 3 can be replaced with the QR decomposition.

Algorithm 3.2 Approximate influence weighted subsampling (aIWS-LS).

Input: Data: $Z$, $y$

1. Solve $\hat{\beta}_{SRHT} = \arg\min_\beta \| \Pi_1 \cdot y - \Pi_1 \cdot Z \beta \|^2$
2. SVD: $U, \Sigma, V = \Pi_1 \cdot Z$  # Compute basis for randomized leverage approximation.
3. $R^{-1} = V \Sigma^{-1}$
4. $\hat{U} = Z R^{-1} \cdot \Pi_2$
5. for $i = 1 \ldots n$ do
6. $\bar{I}_i = \| \hat{U}_i \|$
7. $\bar{e}_i = y_i - z_i^\top \hat{\beta}_{SRHT}$
8. $\bar{d}_i = \bar{e}_i^2 \bar{I}_i / (1 - \bar{I}_i)^2$
9. end for
10. Sample rows ($\tilde{Z}, \tilde{y}$) of ($Z, y$) proportional to $\bar{d}_i^{-1}$
11. Solve $\hat{\beta}_{aIWS} = \arg\min_\beta \| \bar{y} - \tilde{Z} \beta \|^2$

Output: $\hat{\beta}_{aIWS}$

Algorithm 3.3 Residual weighted subsampling (aRWS-LS)

Input: Data: $Z$, $y$

1. Solve $\hat{\beta}_{SRHT} = \arg\min_\beta \| \Pi \cdot (y - Z \beta) \|^2$
2. Estimate residuals: $\bar{e} = y - Z \hat{\beta}_{SRHT}$
3. Sample rows ($\tilde{Z}, \tilde{y}$) of ($Z, y$) proportional to $\bar{e}_i^{-2}$
4. Solve $\hat{\beta}_{RWS} = \arg\min_\beta \| \bar{y} - \tilde{Z} \beta \|^2$

Output: $\hat{\beta}_{RWS}$

Residual weighted sampling. Leverage scores are typically uniform (Ma et al. 2014; Dhillon et al. 2013) for sub-Gaussian data. Even in the corrupted setting, the difference in leverage scores between corrupted and non-corrupted points is small (see Section 3.7). Therefore, the main contribution to the influence for each point will originate from the residual error, $e_i^2$. Consequently, we propose sampling with probability inversely proportional to the approximate residual, $\bar{e}_i^{-2}$. The resulting algorithm Residual Weighted Subsampling (aRWS-LS) is
detailed in Algorithm 3.3. Although $aRWS-LS$ is not guaranteed to be a good approximation to $IWS-LS$, empirical results suggests that it works well in practise and is faster to compute than $aIWS-LS$.

**Computational Complexity.** Clearly, the computational complexity of $IWS-LS$ is $O(np^2)$. The computation complexity of $aIWS-LS$ is $O(np \log n_{subs} + np r_2 + n_{subs}p^2)$, where the first term is the cost of SRHT-LS, the second term is the cost of approximate leverage computation and the last term solves OLS on the subsampled data set. Here, $r_2$ is the dimension of the random projection detailed in Equation (3.13). The cost of $aRWS-LS$ is $O(np \log n_{subs} + np + n_{subs}p^2)$ where the first term is the cost of SRHT-LS, the second term is the cost of computing the residuals $e$, and the last term solves OLS on the subsampled data set. This computation can be reduced to $O(np \log n_{subs} + n_{subs}p^2)$. Therefore the cost of both $aIWS-LS$ and $aRWS-LS$ is $o(np^2)$.

### 3.7 Theoretical Analysis

In this section we will prove an upper bound on the estimation error of $IWS-LS$ in the random sub-Gaussian and in the corrupted model. To bound the error in the corrupted model, we first show that the OLS error consists of two additional variance terms that depend on the size and proportion of the corruptions and an additional bias term. We then show that $IWS-LS$ can significantly reduce the relative variance and bias in this setting, so that it no longer depends on the magnitude of the corruptions but only on their proportion. We compare these results to recent results from Loh et al. (2012) and Chen et al. (2012) suggesting that consistent estimation requires knowledge about $\Sigma_w$. More recently, Chen and Caramanis (2013) show that incomplete knowledge about this quantity results in a biased estimator where the bias is proportional to the uncertainty about $\Sigma_w$. We see that the form of our bound matches these results.

Inequalities are said to hold with high probability (w.h.p.) if the probability of failure is not more than $C_1 \exp(-C_2 \log p)$ where $C_1, C_2$ are positive constants that do not depend on the scaling quantities $n, p, \sigma_w$. The symbols $\precsim$ and $\gtrsim$ mean that we ignore constants that do not depend on these scaling quantities. Thus, for example, writing $n \gtrsim f(n, p, \sigma_w)$ means $n \gtrsim cf(n, p, \sigma_w)$ for a positive universal constant $c$ that does not
scale with \( n, p, \sigma_w \). Unless otherwise stated, \( \| \cdot \| \) denotes the \( \ell_2 \) norm for vectors and the spectral norm for matrices.

### 3.7.1 Estimation error in the sub-Gaussian model

Using the definition and bounds on influence (Lemma \( 3.2 \)) from Section \( 3.3.2 \) and \( 3.4 \), we can state the following theorem characterising the error of the influence weighted subsampling estimator in the sub-Gaussian design setting.

**Theorem 3.7** (Sub-gaussian design influence weighted subsampling). Defining \( E = \| \hat{\beta}_{OLS} - \beta \| \) for \( n \gtrsim \frac{\sigma_e^2}{\lambda_{\min}(\Sigma_x)} p \log p \) we have

\[
\| \hat{\beta}_{IWS} - \beta \| \lesssim \frac{1}{\lambda} \cdot \frac{\sigma_e}{\lambda_{\min}(\Sigma_x)(\sigma_e + 2\sigma_x E)} \cdot \sqrt{\frac{1}{rn}}
\]

where \( 0 \leq \lambda \leq \lambda_{\min}(\Sigma_{\Theta x}) \) and \( \Sigma_{\Theta x} \) is the covariance of the influence weighted subsampled data and \( r = \frac{n_{\text{subs}}}{n} \).

**Proof.** Through subsampling according to influence, we solve the problem

\[
\hat{\beta}_{IWS} = \arg \min_{\beta} \| \Theta y - \Theta X \beta \|^2
\]

where \( \Theta = \sqrt{\frac{n}{n_{\text{subs}}}} \mathbf{S} \mathbf{D} \). \( \mathbf{S} \) is a subsampling matrix, \( \mathbf{D} \) is a diagonal matrix whose entries are \( \sqrt{\frac{P_i}{n}} = \sqrt{\frac{c}{d_i n}} \) where \( c \) is a constant which ensures \( \sum_{i=1}^{n_{\text{subs}}} P_i = 1 \).

\[
D_{ii}^2 \propto \left( \frac{\| \Sigma^{-1} \|}{1 - l_i} \left( \sigma_x \sigma_e + 2\sigma_x^2 \| \beta - \hat{\beta} \| \right) \sqrt{p} \log p \right)^{-1}. \quad (3.16)
\]

Setting \( \hat{\gamma} = (\Theta X)^\top y \), \( \hat{\Sigma} = (\Theta X)^\top (\Theta X) \), by Lemma 3.6 the error of the influence weighted subsampling estimator is given by

\[
\frac{1}{\lambda} \| \hat{\gamma} - \hat{\Sigma} \beta \| = \| (\Theta X)^\top (\Theta y) - (\Theta X)^\top (\Theta X) \beta \|
\]

\[
= \frac{1}{\lambda} \| (\Theta X)^\top (\Theta \epsilon) + (\Theta X)^\top (\Theta X) \beta - (\Theta X)^\top (\Theta X) \beta \|
\]

\[
= \frac{1}{\lambda} \| (\Theta X)^\top (\Theta \epsilon) \| \quad (3.17)
\]
Now, by Lemma 3.7 we have
\[ \|X^\top \epsilon\| \leq \sigma_x \sigma_\epsilon \sqrt{\frac{p \log p}{n}} \]
and so defining \( E = \|\beta - \hat{\beta}\|, \)
\[
\|(\Theta X)^\top \Theta \epsilon\| \leq \left\| \frac{1}{rn} \sum_{i=1}^{rn} p_i \right\| \cdot \|(SX)^\top S \epsilon\|
\leq \left\| \frac{1}{rn} \sum_{i=1}^{rn} (1 - l_i) \right\| \left\| \frac{\sigma_x \sigma_\epsilon \sqrt{p \log p / rn}}{\|\Sigma^{-1}\| (\sigma_x \sigma_\epsilon + 2\sigma_x^2 E) \sqrt{p \log p}} \right\|
\leq \frac{\sigma_x \sigma_\epsilon \sqrt{p \log p / rn}}{\|\Sigma^{-1}\| (\sigma_x \sigma_\epsilon + 2\sigma_x^2 E) \sqrt{p \log p}}
\leq \frac{\sigma_\epsilon \sqrt{1/rn}}{\lambda_{\min}(\Sigma_x)(\sigma_\epsilon + 2\sigma_x E)}
\] (3.18)
where the third inequality uses the fact that \( \sum_{i=1}^{n} (1 - l_i) \leq n. \)

Define \( \Sigma_{\Theta x} = \mathbb{E} \left[ (\Theta X)^\top (\Theta X) \right]. \) Now, when \( n \gtrsim \frac{(\sigma_x^2) p \log p}{\lambda_{\min}(\Sigma_{\Theta x})} \)
using Lemma 3.8 with \( \lambda = \lambda_{\min}(\Sigma_{\Theta x}) \) we have w.h.p. \( \lambda_1((\Theta X)^\top (\Theta X) - \Sigma_{\Theta x}) \leq \frac{1}{54} \lambda_{\min}(\Sigma_{\Theta x}). \) It follows that
\[
\lambda_{\min}((\Theta X)^\top (\Theta X)) = \inf_{\|v\| = 1} v^\top \left( \Sigma_{\Theta x} + (\Theta X)^\top (\Theta X) - \Sigma_{\Theta x} \right) v
\geq \lambda_{\min}(\Sigma_{\Theta x}) - \lambda_1((\Theta X)^\top (\Theta X) - \Sigma_{\Theta x}))
\geq \frac{1}{2} \lambda_{\min}(\Sigma_{\Theta x}).
\] (3.19)

Using (3.18) and (3.18) in Equation (3.17) completes the proof.

\[ \square \]

Remark 3. Theorem 3.7 states that in the non-corrupted sub-Gaussian model, the influence weighted subsampling estimator is consistent. Furthermore, if we set the sampling proportion, \( r \geq O(1/p), \) the error scales as \( O(\sqrt{p/n}). \) Therefore, similar to ULURU there is no dependence on the subsampling proportion.

3.7.2 Corrupted observation model

As a baseline, we first investigate the behaviour of the OLS estimator in the corrupted model.
Theorem 3.8 (A bound on \( \| \hat{\beta}_{OLS} - \beta \| \)). If \( n \gtrsim \frac{\sigma_x^2 \sigma_w^2}{\lambda_{\min}(\Sigma_x)} p \log p \) then w.h.p.

\[
\| \hat{\beta}_{OLS} - \beta \| \lesssim \left( \sigma_x \sigma_x + \pi \sigma_x \sigma_w + \pi \left( \sigma_w^2 + \sigma_w \sigma_x \right) \| \beta \| \right) \sqrt{\frac{p \log p}{n} + \pi \sigma_w^2 \sqrt{p} \| \beta \|} \cdot \frac{1}{\lambda},
\]

(3.20)

where \( 0 < \lambda \leq \lambda_{\min}(\Sigma_x) + \pi \lambda_{\min}(\Sigma_w) \).

Proof. 

Remark 4 (Scaling by \( \pi \)). In the following, with some abuse of notation we will write \( U \tilde{W} \) as \( W \). Now,

\[
\| W \| := \| U \tilde{W} \| \\
\leq \pi \| W \|.
\]

Setting \( \hat{\gamma} = Z^\top y, \hat{\Sigma} = Z^\top Z \) we have

\[
\| \hat{\gamma} - \hat{\Sigma} \beta \| = \| (X + W)^\top y - (X + W)^\top (X + W) \beta \| \\
= \| X^\top (X \beta + \epsilon) + W^\top (X \beta + \epsilon) - X^\top X \beta - W^\top W \beta - X^\top W \beta - W^\top X \beta \| \\
= \| X^\top \epsilon + W^\top \epsilon - X^\top W \beta - W^\top W \beta \| \\
\leq \| X^\top \epsilon \| + \| W^\top \epsilon \| + \| X^\top W \beta \| + \| W^\top W \beta \|.
\]

From Lemma 3.7 and Remark 4 we have w.h.p.

\[
\| X^\top \epsilon \| \leq \sigma_x \sigma_e \sqrt{\frac{p \log p}{n}} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
Now, when \( n \gtrsim \frac{\sigma_x^2 \sigma_w^2 p \log p}{\lambda_{\min}(\Sigma_x)} \) using Lemma 3.8 with \( \lambda = \lambda_{\min}(\Sigma_x) \) we have w.h.p. \( \lambda_1(Z^T Z - (\Sigma_x + \Sigma_w)) \leq \frac{1}{54} \lambda_{\min}(\Sigma_x) \). It follows that

\[
\lambda_{\min}(Z^T Z) = \inf_{\|v\|=1} v^T \left( \Sigma_x + \Sigma_w + Z^T Z - (\Sigma_x + \Sigma_w) \right) v \\
\geq \lambda_{\min}(\Sigma_x) + \lambda_{\min}(\Sigma_w) - \lambda_1(Z^T Z - (\Sigma_x + \Sigma_w)) \\
\geq \frac{1}{2} \lambda_{\min}(\Sigma_x) + \pi \lambda_{\min}(\Sigma_w).
\]

Using Lemma 3.6 with Equations (3.21-3.24) and the above bound for \( \lambda = \lambda_{\min}(Z^T Z) \) completes the proof.

\[ \square \]

Remark 5 (No corruptions case). Notice for a fixed \( \sigma_w \), taking \( \lim_{\pi \to 0} \) or for a fixed \( \pi \) taking \( \lim_{\sigma_w \to 0} \) (i.e. there are no corruptions) the above error reduces to the least squares result (see for example Chen et al. (2012)).

Remark 6 (Variance and Bias). The first three terms in (3.20) scale with \( \sqrt{T/n} \) so as \( n \to \infty \), these terms tend towards 0. The last term does not depend on \( \sqrt{T/n} \) and so for some non-zero \( \pi \) the least squares estimate will incur some bias depending on the fraction and magnitude of corruptions.

We are now ready to state our theorem characterising the mean squared error of the influence weighted subsampling estimator.

**Theorem 3.9** (Influence sampling in the corrupted model). For \( n \gtrsim \frac{\sigma_x^2 \sigma_w^2}{\lambda_{\min}(\Sigma_{\Theta x})} p \log p \) we have

\[
\| \hat{\beta}_{\text{IWS}} - \beta \| \lesssim \left( \sigma_x + \frac{\pi \sigma_e}{\sigma_w + 1} + \pi \| \beta \| \right) \sqrt{\frac{p \log p}{n_{\text{subs}}} + \pi \sqrt{p} \| \beta \|} \cdot \frac{1}{\lambda}
\]

where \( 0 < \lambda \leq \lambda_{\min}(\Sigma_{\Theta x}) \) and \( \Sigma_{\Theta x} \) is the covariance of the influence weighted subsampled data.

**Proof.** When \( n \gtrsim \frac{\sigma_x^2 \sigma_w^2}{\lambda_{\min}(\Sigma_{\Theta x})} p \log p \) using Lemma 3.8 with \( \lambda = \lambda_{\min}(\Sigma_{\Theta x}) \) we have w.h.p. \( \lambda_1((\Theta Z)^T (\Theta Z) - \Sigma_{\Theta x}) \leq \frac{1}{54} \lambda_{\min}(\Sigma_{\Theta x}) \). It follows that

\[
\lambda_{\min}((\Theta Z)^T (\Theta Z)) = \inf_{\|v\|=1} v^T \left( \Sigma_{\Theta x} + (\Theta Z)^T (\Theta Z) - \Sigma_{\Theta x} \right) v \\
\geq \lambda_{\min}(\Sigma_{\Theta x}) - \lambda_1((\Theta Z)^T (\Theta Z) - \Sigma_{\Theta x})) \\
\geq \frac{1}{2} \lambda_{\min}(\Sigma_{\Theta x}).
\]
From the bound in Lemma 3.6 we have
\[ \|\hat{\gamma} - \hat{\Sigma}\beta\| \leq \| (\Theta X)^T (\Theta \epsilon) \| + \| (\Theta W)^T (\Theta \epsilon) \| + \| (\Theta X)^T (\Theta W) \beta \| + \| (\Theta W)^T (\Theta W) \beta \|. \]

We now aim to show that the relative contribution of the corrupted points is decreased under the influence weighted subsampling scheme. To show this, we first multiply both corrupted and non-corrupted points by
\[ \|\Sigma^{-1}\| \left(\sigma_x \sigma_e + 2\sigma_x^2 \|\beta - \hat{\beta}\|\right) \log p \sqrt{p}. \]

This is equivalent to multiplying the non-corrupted points by the subsampling matrix \( S \) and scaling and subsampling the corrupted points by the following term \( \Theta_M = \sqrt{\frac{n}{n_{subs}}} S D_M \) where \( D_M \) has squared diagonal entries proportional to
\[ D_M^2 \propto \frac{1}{n} \cdot \frac{\sigma_e \sigma_x + 2\sigma_x^2 E}{2(\sigma_w^2 + \sigma_w \sigma_x) \|\beta\| + 2(\sigma_w^2 + \sigma_w \sigma_x + \sigma_x^2) E + 2(\sigma_w + \sigma_x) \sigma_e}. \]

Now we have
\[ \|\hat{\gamma} - \hat{\Sigma}\beta\| \lesssim \| (SX)^T (S \epsilon) \| + \| (\Theta_M W)^T (\Theta_M \epsilon) \| + \| (\Theta_M X)^T (\Theta_M W) \beta \| + \| (\Theta_M W)^T (\Theta_M W) \beta \|. \]

Applying Lemma 3.7 we have w.h.p.
\[ \| (SX)^T (S \epsilon) \| \lesssim \sigma_x \sigma_e \sqrt{\frac{p \log p}{rn}} \quad (3.25) \]
\[ \| (\Theta_M W)^T (\Theta_M \epsilon) \| \lesssim \frac{\pi \cdot (\sigma_e + 2E) \pi \sigma_w \sigma_e \sqrt{\frac{p \log p}{rn}}}{2(\sigma_w^2 + \sigma_w \sigma_x) \|\beta\| + 2(\sigma_w^2 + \sigma_w \sigma_x + \sigma_x^2) E + 2(\sigma_w + \sigma_x) \sigma_e} \quad (3.26) \]
\[ \| (\Theta_M X)^T (\Theta_M W) \beta \| \lesssim \frac{\pi \cdot (\sigma_e + 2E) \sigma_x \sigma_w \|\beta\| \sqrt{\frac{p \log p}{rn}}}{2(\sigma_w^2 + \sigma_w \sigma_x) \|\beta\| + 2(\sigma_w^2 + \sigma_w \sigma_x + \sigma_x^2) E + 2(\sigma_w + \sigma_x) \sigma_e} \quad (3.27) \]
\[ \| (\Theta_M W)^T (\Theta_M W) \beta \| \lesssim \frac{\pi \cdot (\sigma_e + 2E) \sigma_w^2 \left( C \sqrt{\frac{p \log p}{rn}} + \sqrt{p} \right) \|\beta\|}{2(\sigma_w^2 + \sigma_w \sigma_x) \|\beta\| + 2(\sigma_w^2 + \sigma_w + 1) E + 2(\sigma_w + 1) \sigma_e}. \quad (3.28) \]
We observe that each of the quantities in Equations (3.26 - 3.28) are scaled by a term proportional to
\[
\frac{\pi \cdot (\sigma_e \sigma_x + 2 \sigma^2 E)}{2(\sigma^2_w + \sigma_w \sigma_x) \| \beta \| + 2(\sigma^2_w + \sigma_w \sigma_x + \sigma^2_x) E + 2(\sigma_w + \sigma_x) \sigma_e}. \tag{3.29}
\]

Taking the limit of large \( E \) of the above (see remark 8) and setting \( \sigma_x = 1 \) we get
\[
\pi^* = \lim_{E \to \infty} \frac{\pi}{(\sigma^2_w + \sigma_w)}. 
\]
Replacing the scaling factor in Equation (3.29) with \( \pi^* \) completes the proof. \( \square \)

Remark 7. Theorem 3.9 states that the influence weighted subsampling estimator removes the proportional dependance of the error on \( \sigma_w \) so the additional variance terms scale as \( O(\pi / \sigma_w \cdot \sqrt{p/n_{subs}}) \) and \( O(\pi \sqrt{p/n_{subs}}) \). The relative contribution of the bias term is \( \pi \sqrt{p} \| \beta \| \) compared with \( \pi \sigma_w^2 \sqrt{p} \| \beta \| \) for the OLS or non-influence-based subsampling methods.

Remark 8 (Taking \( \lim_{\| \hat{\beta}_{OLS} - \beta \| \to \infty} \)). Intuitively, when \( E = \| \hat{\beta}_{OLS} - \beta \| \) is small, this suggests that the effect of the corruptions is negligible and the full (or subsampled) least squares solution is close to optimal. Alternatively, when \( E \) is large, the corruptions have a large effect on the estimate and so influence subsampling should work well. Note that here the size of \( E \) is dependent on \( \sigma_w \) and \( \pi \). If we send \( E \to \infty \) by allowing many points to be corrupted, the relative performance of IWS-LS compared with OLS worsens. However if we allow \( \sigma_w \) to be large, the relative performance of our method improves.

Comparison with fully corrupted setting. We note that the bound in Theorem 3.8 is similar to the bound in Chen and Caramanis (2013) for an estimator where all data points are corrupted (i.e. \( \pi = 1 \)) and where incomplete knowledge of the covariance matrix of the corruptions, \( \Sigma_w \) is used. We reproduce Corollary 4 from Chen and Caramanis (2013) as Lemma 3.5 below.
Lemma 3.5 (Originally Corollary 4 from Chen and Caramanis (2013)). If \( \Sigma_w \) is known and \( n \gtrsim \frac{(1+\sigma_w^2)2}{\lambda_{\min}(\Sigma_x)p \log p} \). Then w.h.p., plugging the estimator built using \( \hat{\Sigma} = Z^\top Z - \Sigma_w \) and \( \hat{\gamma} = Z^\top y \) into Lemma 3.6, satisfies

\[
\| \hat{\beta} - \beta \| \lesssim \frac{(\sigma_w^2 + \sigma_w)\| \beta \| + \sigma_c \sqrt{1 + \sigma_w^2}}{\lambda_{\min}(\Sigma_x)} \sqrt{\frac{p \log p}{n}}. \tag{3.30}
\]

When only an upper bound \( \Sigma_w \succeq \Sigma_w \) is known then

\[
\| \hat{\beta} - \beta \| \lesssim \frac{\left[ (\sigma_w^2 + \sigma_w)\| \beta \| + \sigma_c \sqrt{1 + \sigma_w^2} \right]}{\lambda_{\min}(\Sigma_x) - \lambda_{\max}(\Sigma_w - \Sigma_w)} \sqrt{\frac{p \log p}{n}} + \frac{\lambda_{\max}(\Sigma_w - \Sigma_w)\| \beta \|}{\lambda_{\min}(\Sigma_x) - \lambda_{\max}(\Sigma_w - \Sigma_w)}. \tag{3.31}
\]

We can compare these two statements with our result from Theorem 3.8. Equation (3.30) is similar to the bound we have from Theorem 3.8 up to the bias term assuming \( \pi = 1 \) (i.e. all of the points are corrupted). Since we do not use knowledge of \( \Sigma_w \) we can compare our result with Equation (3.31) which has a bias term which is related to the uncertainty in the estimate of \( \Sigma_w \) which in our case is \( \sigma_w^2 \). It is clear from Lemma 3.5 that the only way to remove this bias completely is to use additional information about the covariance of the corruptions.

3.8 Experimental Results

We compare IWS-LS against the methods SRHT-LS (Mahoney 2011), ULURU (Dhillon et al. 2013). These competing methods represent current state-of-the-art in fast randomized least squares. Since SRHT-LS is equivalent to LEV-LS (Drineas et al. 2006) the comparison will highlight the difference between importance sampling according to the two difference types of regression diagnostic in the corrupted model. Similar to IWS-LS, ULURU is also a two-step procedure where the first is equivalent to SRHT-LS. The second reduces bias by subtracting the result of regressing onto the residual. The experiments with the corrupted data model will demonstrate the difference in robustness of IWS-LS and ULURU to corruptions in the observations. Note that we do not compare with SGD. Although SGD has excellent properties for large-scale linear regression, we are not aware of a convex loss function which is robust to the corruption model we propose.
We assess the empirical performance of our method compared with standard and state-of-the-art randomized approaches to linear regression in several different scenarios. We evaluate these methods on the basis of the estimation error: the $\ell_2$ norm of the difference between the true weights and the learned weights, $\|\hat{\beta} - \beta\|$. Additionally we compare the root mean squared prediction error (RMSE) on the test set.

For aIWS-LS and aRWS-LS we used the same number of sub-samples to approximate the leverage scores and residuals as for solving the regression. For aIWS-LS we set $r_2 = p/2$ (see Equation (3.13)). The results are averaged over 100 runs.

### 3.8.1 Non-corrupted data.

We first compare performance in three different leverage regimes taken from Ma et al. (2014): uniform leverage scores (multivariate Gaussian), slightly non-uniform (multivariate-t with 3 degrees of freedom, T-3), highly non-uniform (multivariate-t with 1 degree of freedom, T-1). Full details of the data simulating process can be found in Ma et al. (2014).

The size of the data set is: $n_{\text{train}} = 10'000$, $n_{\text{test}} = 1'000$, $p = 50$.

Figures 3.3, 3.4 and 3.5 show the estimation error and the RMSE for the simulated data sets described in Ma et al. (2014). The results for the T-3 data are similar to the Gaussian data. The slightly heavier tails of the multivariate $t$ distribution with 3 degrees of freedom cause the leverage scores to be less uniform which degrades the performance of uniform subsampling relative to SRHT-LS and IWS-LS. The RMSE performance is similar to that of the statistical estimation error.

### 3.8.2 Corrupted data.

We investigate the corrupted data noise model described in Equations (3.1)-(3.2). We show three scenarios where $\pi = \{0.05, 0.1, 0.3\}$. $X$ and $W$ were sampled from independent, zero-mean Gaussians with standard deviation $\sigma_x = 1$ and $\sigma_w = 0.4$ respectively. The true regression coefficients, $\beta$ were sampled from a standard Gaussian. We added i.i.d. zero-mean Gaussian noise with standard deviation $\sigma_e = 0.1$.

Figure 3.6, Figure 3.7 and Figure 3.8 show the estimation error and the mean squared prediction error for different subsample sizes. In all settings influence based methods outperform all other approximation
3.8 Experimental Results

Figure 3.3: Gaussian data: Comparison of (a) mean estimation error and (b) root mean square prediction error on Gaussian data for increasing number of sub-sampled points.

Figure 3.4: Student-$t$ 3 data: Comparison of (a) mean estimation error and (b) root mean square prediction error on the T-3 data set for increasing number of sub-sampled points.
Figure 3.5: Student-\(t\) 1 data: Comparison of (a) mean estimation error and (b) root mean square prediction error on the T-1 data set for increasing number of sub-sampled points.

Figure 3.6: 5\% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 5\% corruptions.
3.8 Experimental Results

Figure 3.7: 10% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 10% corruptions.

Figure 3.8: 30% corruptions (small-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on small-scale corrupted data set with 30% corruptions.
methods. For 5% corruptions for a small number of samples ULURU outperforms the other subsampling methods. However, as the number of samples increases, influence based methods start to outperform OLS. For > 3000 subsamples, the bias correction step of ULURU causes it to diverge from OLS and ultimately perform worse than uniform.

For 10% corruptions, aIWS-LS and aRWS-LS converge quickly to IWS-LS. As the number of corruptions increase further, the relative performance of IWS-LS with respect to OLS decreases slightly as suggested by Remark 8.

For 30% corruptions, the approximate influence algorithms achieve almost exactly the same performance as IWS-LS. Even for a small number of samples all of the influence methods far outperform OLS. As the proportion of corruptions increases further, the rate at which the approximate influence algorithms approach IWS-LS slows and the relative difference between IWS-LS and OLS decreases slightly. In all cases, influence based methods achieve lower-variance estimates. Here, ULURU converges quickly to the OLS solution but is not able to overcome the bias introduced by the corrupted data points.

These experimental results confirm that OLS is biased in the corrupted setting and that sampling by leverage does not mitigate this as we both have shown theoretically in Section 3.7. IWS-LS on the other hand consistently achieves lower error as it samples non-corrupted points with high probability and the approximate methods aIWS-LS and aRWS-LS quickly approach the performance of IWS-LS.

3.8.3 Larger scale experiments on corrupted data.

We now present results on larger scale simulated data. We used the same experimental setup as in Section 3.8.2 but we increase the size of the data to $n = 100,000$ and $p = 500$.

Figure 3.9, Figure 3.10 and Figure 3.11 show the estimation error and RMSE for 5%, 10% and 30% corrupted data points respectively. In this setting, computing IWS-LS is impractical (due to the exact leverage computation) so we omit the results but we notice that aIWS-LS and aRWS-LS quickly improve over the full least squares solution and the other randomized approximations. The general trend in this setting is the same as with the smaller experiments, however for 5% corruptions the improvement of aIWS-LS and aRWS-LS over OLS happens with a much smaller subsampling ratio than with smaller data sets.
For 30% corruptions for a small number of samples ULURU outperforms the other subsampling methods. However, as the number of samples increases, influence based methods start to outperform OLS. Here, ULURU converges quickly to the OLS solution but is not able to overcome the bias introduced by the corrupted data points.

Figure 3.9: 5% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 5% corruptions.

Figure 3.10: 10% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 10% corruptions.
Figure 3.11: 30% corruptions (large-scale): Comparison of (a) mean estimation error and (b) root mean square prediction error on large-scale corrupted data set with 30% corruptions.

Figure 3.12: Airline delay: Comparison of the root mean square prediction error on the airline delay data set for increasing number of sub-sampled points.
3.8.4 Airline delay dataset

The data set consists of details of all commercial flights in the USA over 20 years. Selecting the first \( n_{\text{train}} = 13,000 \) US Airways flights from January 2000 (corresponding to approximately 1.5 weeks) our goal is to predict the delay time of the next \( n_{\text{test}} = 5,000 \) US Airways flights. The features in this data set consist of a binary vector representing origin-destination pairs and a real value representing distance (\( p = 170 \)).

The data set might be expected to violate the usual i.i.d. sub-Gaussian design assumption of standard linear regression since the length of delays are often very different depending on the day. For example, delays may be longer due to public holidays or on weekends. Of course, such regular events could be accounted for in the modelling step, but some unpredictable outliers such as weather delay may also occur. Results are presented in Figure 3.12, the RMSE is the error in predicted delay time in minutes. Since the data set is smaller, we can run IWS-LS to observe the accuracy of aIWS-LS and aRWS-LS in comparison. For more than 3000 samples, these algorithm outperform OLS and quickly approach IWS-LS. The result suggests that the corrupted observation model is a good model for this data set. Furthermore, ULURU is unable to achieve the full accuracy of the OLS solution.

3.9 Conclusions

We have demonstrated theoretically and empirically under the generalised corrupted observation model that influence weighted subsampling is able to significantly reduce both the bias and variance compared with the OLS estimator and other randomized approximations which do not take influence into account. Importantly our fast approximation, aRWS-LS performs similarly to IWS-LS. We find ULURU quickly converges to the OLS estimate, although it is not able to overcome the bias induced by the corrupted data points despite its two-step procedure. The performance of IWS-LS relative to OLS in the airline delay problem suggests that the corrupted observation model is a more realistic modelling scenario than the standard sub-Gaussian design model for some tasks.

\[ Data \text{ set along with visualisations available from } \text{http://stat-computing.org/dataexpo/2009/}. \]
APPENDIX

3.1 PROOFS FOR THE BOUNDS ON LEVERAGE AND INFLUENCE

In this section we proof Lemma 3.1, Lemma 3.2, Lemma 3.3 and Lemma 3.4 from Section 3.4.

Proof of Lemma 3.1. From the Eigen-decomposition of $\Sigma = XX^\top$, $\Sigma = \Lambda V V^\top$. Define $A = \Lambda^{-1/2}V$ such that $A^\top A = \Sigma^{-1}$. We have

$$l_i = x_i^\top \Sigma^{-1} x_i = \|Ax_i\|^2$$

Since $x$ and $w$ are sub-Gaussian random vectors so the above quadratic form is bounded by Lemma 3.9, setting the parameter $t = \log p$. We combine this with the following inequalities

$$\sqrt{\text{tr}\left(\Sigma^{-2}\right)} = \|\Sigma^{-1}\|_F \leq \sqrt{p}\|\Sigma^{-1}\| = \sqrt{p}\sigma_1(A)^2$$

and

$$\text{tr}\left(\Sigma^{-1}\right) = \|A\|_F^2 \leq (\sqrt{p}\|A\|)^2 = p\sigma_1(A)^2$$

which relate the Frobenius norm with the spectral norm. We also make use of the relationship $\sigma_n(Z)^{-1} = \sigma_1(A)$ where $Z = X + W$ to obtain

$$\|Ax\|^2 \leq \sigma_x^2 \sigma_n(Z)^{-2} \left(p + 2\log p + 2\sqrt{p\log p}\right)$$

which holds with high probability.

In order for this bound to not be vacuous in our application, it must be smaller than 1. In order to ensure this, we need to bound $\sigma_n(Z)^{-1}$ using Lemma 3.10 and setting $\tau = \sqrt{c_0 \log p}$ to obtain the following which holds with high probability

$$\|Ax\|^2 \leq \sigma_x^2 \left(\frac{p + 2\log p + 2\sqrt{p\log p}}{\sqrt{n} - C\sqrt{p} - \tau}\right)^2 \leq \sigma_x^2 \left(\frac{p + 2\log p + 2\sqrt{p\log p}}{\sqrt{n} - C\sqrt{p} - \sqrt{\log p}}\right)^2.$$
Proof of Lemma 3.2. Defining $\Sigma = Z^\top Z$, the lemma states
\[
\|\widehat{\beta}_i - \bar{\beta}\| \leq \frac{\|\Sigma^{-1}\|}{1 - l_i} \left( \sigma_x \sigma_\epsilon + 2\sigma_x^2 \|\beta - \bar{\beta}\| \right) \sqrt{p \log p}.
\]
Using Proposition 3.4 we have
\[
\|\widehat{\beta}_i - \bar{\beta}\| = \frac{1}{1 - l_i} \|\Sigma^{-1} x_i e_i\| \\
= \frac{1}{1 - l_i} \|\Sigma^{-1} x_i \left( \epsilon + x_i^\top \left( \beta - \bar{\beta} \right) \right)\| \\
\leq \frac{1}{1 - l_i} \|\Sigma^{-1}\| \|x_i \epsilon + x_i^\top (\beta - \bar{\beta})\| \\
\leq \frac{1}{1 - l_i} \|\Sigma^{-1}\| \left( \|x_i \epsilon\| + \|x_i x_i^\top (\beta - \bar{\beta})\| \right).
\]
Using Corollary 3.10 to bound $\|x_i \epsilon\|$ and $\|x_i x_i^\top (\beta - \bar{\beta})\|$ (since for these terms $n = 1$ and so Lemma 3.7 does not immediately apply) completes the proof.

Proof of Lemma 3.3. The proof follows from rewriting $l_m = \|A(x_m + w_m)\|^2$ and following the same steps as the proof of Lemma 3.1 above.

Proof of Lemma 3.4. From Proposition 3.5 and following the same argument as Lemma 3.2 we have
\[
\|\widehat{\beta}_m - \bar{\beta}\| = \frac{1}{1 - l_m} \|\Sigma^{-1}(x_m + w_m)e_m\| \\
\leq \frac{1}{1 - l_m} \|\Sigma^{-1}(x_m + w_m)\left( (x_m + w_m)^\top (\beta - \bar{\beta}) + w_m^\top \beta + \epsilon \right)\| \\
\leq \frac{1}{1 - l_m} \|\Sigma^{-1}\left( \|x_m w_m \beta\| + \|w_m^\top w_m \beta\| + \|x_m \epsilon\| + \|w_m \epsilon\| \\
+ \|x_m x_m^\top + w_m^\top w_m + 2x_m^\top w_m \right) (\beta - \bar{\beta})\|).
\]
Applying the triangle inequality followed by Corollary 3.10 and noting that $(\sigma_x \sigma_w + 2\sigma_w^2) \leq 2(\sigma_x \sigma_w + \sigma_w^2)$ completes the proof.

3.8 THEORETICAL BACKGROUND

Here we collect results which are useful in the statements and proofs of our main theorems.
Lemma 3.6 (A general bound on $\|\hat{\beta} - \beta\|$ (Chen and Caramanis 2013)). Suppose the following strong convexity condition holds: $\lambda_{\text{min}}(\hat{\Sigma}) \geq \lambda > 0$. Then the estimation error satisfies

$$
\|\hat{\beta} - \beta\| \lesssim \frac{1}{\lambda} \|\hat{\gamma} - \hat{\Sigma}\|.
$$

Where $\hat{\gamma}$, $\hat{\Sigma}$ are estimators for $E[X^\top y]$ and $E[X^\top X]$ respectively.

To obtain the results for our method in the non-corrupted and corrupted setting we can simply plug in our specific estimates for $\hat{\gamma}$ and $\hat{\Sigma}$.

Lemma 3.7 (Originally Lemma 25 from Chen et al. (2012)). Suppose $X \in \mathbb{R}^{n \times k}$ and $W \in \mathbb{R}^{n \times m}$ are zero-mean sub-Gaussian matrices with parameters $(\frac{1}{n} \Sigma_x, \frac{1}{n} \sigma_x^2)$, $(\frac{1}{n} \Sigma_w, \frac{1}{n} \sigma_w^2)$ respectively. Then for any fixed vectors $v_1, v_2$, we have

$$
\mathbb{P} \left[ |v_1^\top \left( W^\top X - E[W^\top X] \right) v_2| \geq t \|v_1\| \|v_2\| \right] 
\leq 3 \exp \left( -cn \min \left\{ \frac{t^2}{\sigma_x^2 \sigma_w^2}, \frac{t}{\sigma_x \sigma_w} \right\} \right) \quad (3.32)
$$

in particular if $n \gtrsim \log p$ we have w.h.p.

$$
|v_1^\top \left( W^\top X - E[W^\top X] \right) v_2| 
\leq \sigma_x \sigma_w \|v_1\| \|v_2\| \sqrt{\frac{\log p}{n}}
$$

Setting $v_1$ to be the first standard basis vector, and using a union bound over $j = 1, \ldots, p$, we have w.h.p.

$$
\| \left( W^\top X - E[W^\top X] \right) v \|_{\infty} \leq \sigma_x \sigma_w \|v\| \sqrt{\frac{\log p}{n}}
$$

holds with probability $1 - c_1 \exp(-c_2 \log p)$ where $c_1, c_2$ are positive constants which are independent of $\sigma_x, \sigma_w, n$ and $p$.

Corollary 3.10 (Modification of Lemma 3.7 for $n = 1$). Suppose $X \in \mathbb{R}^{n \times k}$ and $W \in \mathbb{R}^{n \times m}$ are zero-mean sub-Gaussian matrices with parameters $(\frac{1}{n} \Sigma_x, \frac{1}{n} \sigma_x^2)$, $(\frac{1}{n} \Sigma_w, \frac{1}{n} \sigma_w^2)$ respectively. Then for any fixed vector $v_1$ and $n = 1$ we have w.h.p.

$$
\| \left( W^\top X - E[W^\top X] \right) v \|_{\infty} \leq \sigma_x \sigma_w \|v\| \log p.
$$
Proof. Setting $t = c_0\sigma_x\sigma_w \log p$, $n = 1$ and $v$ as the first standard basis vector in Inequality (3.32) in Lemma 3.7 and applying a union bound over $j = 1, \ldots, p$ yields the result. \hfill \Box

Lemma 3.8 (Originally Lemma 11 from Chen and Caramanis (2013)). If $X$ and $W$ are zero-mean sub-Gaussian matrices then

$$\Pr \left[ \sup_{\|v_1\| = \|v_2\| = 1} \left| v_1^\top \left( W^\top X - \mathbb{E}[W^\top X] \right) v_2 \right| \geq t \right] \leq 2 \exp \left( -cn \min \left\{ \frac{t^2}{\sigma_x^2 \sigma_w^2}, \frac{t}{\sigma_x \sigma_w} \right\} + 6(k + m) \right)$$

In particular, for each $\lambda > 0$, if $n \gtrsim\max \left\{ \frac{\sigma_x^2 \sigma_w^2}{\lambda^2}, 1 \right\} (k + m) \log p$, then w.h.p.

$$\sup_{v_1, v_2} \left| v_1^\top \left( W^\top X - \mathbb{E}[W^\top X] \right) v_2 \right| \leq \frac{1}{54} \lambda \|v_1\| \|v_2\|.$$

Lemma 3.9 (Quadratic forms of sub-Gaussian random variables. Theorem 2.1 from Hsu et al. (2012)). Let $A \in \mathbb{R}^{n \times n}$ be a matrix, and let $\Sigma := A^\top A$. $x$ is a mean-zero random vector such that, for some $\sigma \geq 0$,

$$\mathbb{E} \left[ \exp(\alpha^\top x) \right] \leq \exp(\|\alpha\|^2 \sigma^2 / 2)$$

for all $\alpha \in \mathbb{R}^n$. For all $t > 0$

$$\Pr \left[ \|Ax\|^2 > \sigma^2 \left( \text{tr}(\Sigma) + 2\sqrt{\text{tr}(\Sigma^2)} t + 2\|\Sigma\|t \right) \right] \leq e^{-t}. $$

Lemma 3.10 (Extremal singular values of a matrix with i.i.d. sub-Gaussian rows. Theorem 5.39 of Vershynin (2010)). Let $A$ be an $n \times p$ matrix whose rows $A_i$ are independent sub-Gaussian isotropic random vectors in $\mathbb{R}^p$. Then for every $\tau \geq 0$, with probability at least $1 - 2 \exp(-c\tau^2)$ we have

$$\sqrt{n} - C\sqrt{p} - \tau \leq \sigma_n(A) \leq \sigma_1(A) \leq \sqrt{n} + C\sqrt{p} + \tau$$

where $C$ and $c$ are constants which depend only on the sub-Gaussian norm of the rows of $A$. 

56
DISTRIBUTED RIDGE REGRESSION

In this chapter we propose Loco, a distributed algorithm which solves large-scale ridge regression. Loco randomly assigns variables to different workers which do not communicate\(^1\). Important dependencies between variables are preserved using random projections which are cheap to compute. We show that Loco has bounded approximation error compared to the exact ridge regression solution in the fixed design setting. Experimentally, in addition to obtaining significant speedups Loco achieves good predictive accuracy on a variety of large-scale regression problems. Notably Loco is able to solve a regression problem with 5 \textit{billion} non-zeros distributed across 128 workers in 25 seconds.

The distribution of the features during training introduces uncertainty as each worker only sees a subset and the relationship between the features on the different machines is lost. As in Chapter 3 we will use random projections (Section 2.6) to approximate important aspects of the data set. Where in Chapter 3 we approximated the influence of a data point on the regression problem, here each worker approximates his subset and communicates it to all the other workers. This allows us to cheaply preserve the dependencies between the distributed features and only needs little communication between the workers once in the beginning of the training procedure.

4.1 \textsc{Introduction}

Two obvious questions arise: (1) how should the data and processing tasks be distributed among processing units (workers) and (2) how and what should each worker communicate. The choice of learning algorithm affects both of these points. Stochastic gradient descent methods (SGD, Section 2.5) are suited to parallelization over the rows (observations) of the data (Zinkevich et al. 2010). However, synchronization of results to ensure each worker is updating the current gradient becomes expensive. This has motivated recent asynchronous approaches.

\(^1\) Parts of this chapter appear in Heinze et al. (2014) and Heinze (2014)
to parallel SGD (Niu et al. 2011; Duchi et al. 2013). Parallelizing over the columns (features) of the data has been proposed for coordinate descent optimization which is commonly used to solve $\ell_1$ penalized problems (Bradley et al. 2011). To ensure convergence, the inherent dependence between features is typically assumed to be small between blocks which are operated on in parallel. Fewer methods have been proposed for distributed optimization on a cluster of machines where communication costs must be considered (Richtarik et al. 2013).

Specifically, we limit our focus to $\ell_2$ penalized linear regression for large-scale estimation tasks when the size of the data is such that a single multi-core processor is insufficiently fast. We therefore wish to distribute the problem in a way which allows computation to be shared across many machines which do not share memory. Ideally this is done in such a way that synchronization and communication between workers are kept to a minimum. Another setting which motivates distributed estimation is one of privacy preservation. In this framework, no single worker may see all of the features and so even when the data size is not massive, sharing memory and data between workers is not permitted.

For linear regression, it can be shown that the least squares solution computed on a random projection (Section 2.6) of either the row (Mahoney 2011) or column (Lu et al. 2013; Kabán 2014) space of the data matrix results in a solution which is close to optimal. An obvious downside to dimensionality reduction is that the solution obtained is no longer in the original space. Therefore, the estimated coefficients are difficult to interpret with respect to the observed features – a task often as important as prediction accuracy. Furthermore, in order to compute the projection, a single machine is assumed to have access to the entire data set.

In this chapter we propose and analyze Loco a simple, low-communication distributed algorithm to approximately solve $\ell_2$ penalized least squares which crucially requires no synchronization between workers. Loco assigns features to workers by randomly partitioning the data into $K$ blocks (alternatively, this may be part of the problem specification). In each block, a small number of cheaply computed random projections are used to approximate the contribution from the remaining columns of the data. Each worker then simply optimizes the objective independently on this compressed data set, the size of which is proportional to the size of the random projection and the total number of workers. The
solution vector returned by Loco is constructed by collecting the estimates for the respective unprojected “raw” features from each worker such that it lies in the original space.

OUTLINE AND CONTRIBUTION. In Section 4.2 we formally describe our estimation problem and the distributed setting which we consider. In Section 4.3 we describe Loco, our algorithm for distributed ridge regression. In Section 4.4 we show in the fixed design setting that the error between the coefficients estimated by Loco and the optimal ridge regression coefficients is bounded, under natural assumptions about the problem setting – that some proportion of the signal lies in the top principal components. Importantly, unlike other approaches to parallelizing or distributing optimization, we make no assumptions on sparsity in the data. In Section 4.5 we place our contribution in context of recently proposed related approaches to distributed optimization. In Section 4.6 we provide implementation details and empirical evaluation of our algorithm on large-scale simulated and real data sets. Loco typically exhibits near-linear speedups with the number of workers with little loss in prediction accuracy.

4.2 PROBLEM SETTING

As in the last chapter (Chapter 3) we consider the linear regression problem in a large-scale setting. Here we add a regularization term to the least-squares term from the ordinary least squares estimator in Equation 3.3 and concentrate on ridge regression, a ubiquitous tool for high-dimensional data analysis (Hastie et al. 2009). Given a matrix of features $X \in \mathbb{R}^{n \times p}$ and a corresponding vector of responses, $y \in \mathbb{R}^n$ where the dimensionality $p$ and sample size $n$ are very large, we are interested in solving the following estimation task

$$
\min_{\beta \in \mathbb{R}^p} L(\beta) := n^{-1} \left\| y - X\beta \right\|^2 + \lambda \left\| \beta \right\|^2 \tag{4.1}
$$

The first term is the squared error loss and the second term is the ridge penalty which regularizes the size of the coefficient vector according to the tuning parameter, $\lambda$. Ridge regression has a closed-form solution $\hat{\beta}_{rr} = (X^\top X + n\lambda I)^{-1} X^\top y$, but clearly when the dimensionality of the data is large, constructing and inverting the $p \times p$ covariance matrix
DISTRIBUTED RIDGE REGRESSION. We now consider the case where we distribute the features across \( K \) different workers. Formally, let \( \mathcal{P} \) be the set of indices \( 1, \ldots, p \). We partition this into \( K \) non-overlapping subsets \( \mathcal{P}_1, \ldots, \mathcal{P}_K \) of equal size, \( p_k = p/K \) so \( \mathcal{P} = \bigcup_{k=1}^{K} \mathcal{P}_k \) and \( |\mathcal{P}_1| = |\mathcal{P}_2|, \ldots, = |\mathcal{P}_K| = p_k \). This is for simplicity of notation only, in general the partitions can be of different sizes.

A naive attempt at parallelizing (4.1) would simply be solving the minimization problem on each subset of features \( \mathcal{P}_k \) independently. However, without sparsity in the data set to guide the partitioning process, important dependencies between features in different blocks would not be preserved.

We can rewrite (4.1) making explicit the contribution from block \( k \).

Letting \( X_k \in \mathbb{R}^{n \times p_k} \) be the sub-matrix whose columns correspond to the coordinates in \( \mathcal{P}_k \) (the “raw” features of block \( k \)) and \( X_{(-k)} \in \mathbb{R}^{n \times (p - p_k)} \) be the remaining columns of \( X \), we have

\[
L(\beta) = n^{-1} \| y - X_k \beta_{\text{raw}} - X_{(-k)} \beta_{(-k)} \|^2 + \lambda \| \beta_{\text{raw}} \|^2 + \lambda \| \beta_{(-k)} \|^2. \quad (4.2)
\]

The idea behind our approach is to replace \( X_{(-k)} \) in each block with a low-dimensional approximation. Since the regularizer is separable across blocks, we only require that the contribution from \( X_{(-k)} \beta_{(-k)} \) to the loss in (4.1) is preserved.

Let \( \tilde{X}_k \in \mathbb{R}^{n \times (K-1)\tau_k} \) be the matrix whose columns are a low-dimensional approximation to \( X_{(-k)} \), i.e. to the columns of \( X \) not in \( X_k \), and \( \tau_k \ll p_k \). Note that each of the other \( K - 1 \) blocks provides an approximation of its respective \( p_k \) raw features of size \( \tau_k \). \( \tau_k \) needs to be small enough that \( \tilde{X}_k \) can quickly be communicated to all the workers once in the beginning of the procedure and added to their raw features. We shall call the columns in \( \tilde{X}_k \) the “random” features of block \( k \). See Figure 4.1 for a description of this procedure. Defining the sub-problem that worker \( k \) solves as

\[
L_k(\beta_k) = n^{-1} \| y - X_k \beta_{\text{raw}} - \tilde{X}_k \beta_{k,\text{rp}} \|^2 + \lambda \| \beta_{\text{raw}} \|^2 + \lambda \| \beta_{k,\text{rp}} \|^2, \quad (4.3)
\]

we require the approximation \( \tilde{X}_k \) to be such that the risk of the estimator which minimizes Equation (4.3) is similar to the risk of the minimizer
of Equation (4.2) (we formalize this in Section 4.4). In order to achieve this we construct the approximation using random projections (see Section 2.6). Remember that the SRHT (Section 2.6.2.1) is defined as the projection matrix $\Pi = \sqrt{p_k/\tau_k}S\Theta D$ with the matrices $S\Theta D$ as defined in Section 2.6.2.1.

For moderately sized problems, random projections have been used to reduce the dimensionality of the data prior to performing OLS (Kabán 2014) and ridge regression (Lu et al. 2013). However after projection, the solution vector is in the compressed space and so interpretability of coefficients is lost. Furthermore, for large problems the running time of the SRHT presents a large constant overhead.
DISTRIBUTED RIDGE REGRESSION

4.3 ALGORITHM

Our procedure, Loco for distributed ridge regression is presented in Algorithm 4.1. We describe the steps in more detail below.

Algorithm 4.1 Loco

**Input:** Data: $X, y$, Number of blocks: $K$, Parameters: $\tau_k, \lambda$

1. Partition $P = \{1, \ldots, p\}$ into $K$ subsets $P_1, \ldots, P_K$ of equal size, $p_k$.
2. for each worker $k \in \{1, \ldots, K\}$ in parallel do
3. Compute and send random projection $\hat{X}_k = X_k \Pi_k$.
4. Construct $\bar{X}_k$
5. $\hat{\beta}_k \leftarrow \text{SolveRidge}(\bar{X}_k, y, \lambda)$
6. $\hat{\beta}_k = [\hat{\beta}_k]_{1:p_k}$
7. end for

**Output:** Solution vector: $\hat{\beta} = [\hat{\beta}_1, \ldots, \hat{\beta}_K]$

**INPUT.** As well as the usual regularization parameter $\lambda$, Loco requires the specification of the number of workers $K$ and the random projection dimension $\tau_k$.

**Steps 1 & 3.** We first randomly partition the coordinates into $K$ subsets. Then each worker computes a random projection, via the SRHT (Section 2.6.2.1), of its respective block which we denote by $\hat{X}_k = X_k \Pi_k \in \mathbb{R}^{n \times \tau_k}$.

**Step 4.** Each worker $k$ constructs the matrix

$$\bar{X}_k \in \mathbb{R}^{n \times (p_k + (K-1)\tau_k)} = \left[X_k, \left[\hat{X}_k'\right]_{k' \neq k}\right]$$

which is the column-wise concatenation of the raw feature matrix $X_k$ and the random approximations from all other blocks. Without loss of generality the raw features will always occupy the first $p_k$ columns of $\bar{X}_k$. The last $(K-1)\tau_k$ columns of $\bar{X}_k$ are a good approximation of the $(K-1)$ blocks of the full data matrix not in $X_k$ and so solving (4.1)

---

2 Alternatively, when $\Pi$ is defined explicitly, summing the $p_k \to \tau_k$-dimensional random projections of $K-1$ blocks is equivalent to computing the $(p-p_k) \to \tau_k$-dimensional random projection in one go which allows for a much smaller dimensional but also less accurate representation.
using $\tilde{X}_k$ obtains a solution which is close to the optimal solution using $X$. We make this explicit in Section 4.4.

**Steps 5 & 6.** The function $\text{SolveRidge}(\bar{X}_k, y, \lambda)$ returns a vector

$$\bar{\beta}_k \in \mathbb{R}^{pk+(K-1)\tau_k} = \arg\min_{\beta_k} \frac{1}{n} \| y - \bar{X}_k \beta_k \|^2 + \lambda \| \beta_k \|^2$$  (4.4)

In practice, any fast algorithm which returns an accurate solution to Equation (4.4) can be used here. The final solution vector $\hat{\beta}$ is the concatenation of the first $p_k$ coordinates of each $\bar{\beta}_k$ and so lives in the same space as the original data.

**Computational, Memory and Communication Costs.** The cost of computing a fast random projection of the $p_k$ features in each block is $O(p_k \log \tau_k)$. Assuming a solver which scales linearly with the problem dimension (i.e. stochastic gradient descent) is used in $\text{SolveRidge}(\bar{X}_k, y, \lambda)$, the part of the computational cost which is dependent on the dimension scales with $O(p_k \log \tau_k + p_k + (K-1)\tau_k)$.

Each machine only needs to store a copy of its block of raw features and a random projection of the remaining features which is $O(p_k + (K-1)\tau_k)$. This is substantially smaller than the original dimensionality $p$. Each worker must communicate its random projection once to all other workers (or to a shared location where the other workers can read it). Aside from this information exchange there is no further communication between workers. The small size of the projection ensures that for appropriately sized problems, each worker is able to store its relevant features in local memory.

### 4.4 Analysis

In this section we will show that in the fixed design setting the coefficients estimated by Loco are close to the full ridge regression solution. The results here are developed for the case where the random features in $\tilde{X}_k$ result from concatenating the SRHT projections of all other blocks and throughout we shall assume that the columns of $X$ and $\tilde{X}_k$ are standardized.

Consider the model

$$y = X\beta^* + \varepsilon,$$  (4.5)
DISTRIBUTED RIDGE REGRESSION

with fixed $X \in \mathbb{R}^{n \times p}$ and true parameter vector $\beta^* \in \mathbb{R}^p$. Assumption 1 below will formalize our assumptions on the noise, $\varepsilon$. Let $\hat{\beta}^{rr}$ denote the ridge estimate for $\beta^*$, so $\hat{\beta}^{rr}$ is the solution which results from solving the ridge regression problem in the original space, stated in Equation (4.2).

In order to formulate our result, we require the following risk function.

**Definition 4.1 (Risk).** Let $\hat{b}$ be an estimator for $\beta^*$ and define the risk of $\hat{b}$ with fitted values $\hat{y} = X\hat{b} \in \mathbb{R}^n$ as

$$R(X\hat{b}) = n^{-1}E_\varepsilon \|X\beta^* - X\hat{b}\|^2.$$ 

In the derivation of Theorem 4.2 we make use of the fact that we can rewrite the regularized optimization problems in Equation (4.2) and Equation (4.3) as constrained optimization problems with a monotonic relationship between the regularization parameter $\lambda$ and the constraint $t$ which upper-bounds the $\ell_2$ norm of the solution. In the original space we have

$$\min_{\|\beta\|^2 \leq t} n^{-1}\|y - X\beta\|^2$$  \tag{4.6}$$

while each worker solves

$$\min_{\|\beta_k\|^2 \leq t} n^{-1}\|y - X_k\beta_k\|^2$$  \tag{4.7}$$

in a compressed space. Recall that $\hat{\beta}^{rr}$ minimizes Equation (4.6) and $\beta_k$ minimizes Equation (4.7).

Before we state our main theorem, we make the natural assumption that the main contribution to the $\ell_2$ norm of the true parameter vector – i.e. most of the important signal – lies in the direction of the first $J$ principal components of $X$. This merely formalizes the conditions under which ridge regression yields good results. Since ridge regression applies more shrinkage in directions associated with smaller eigenvalues (Hastie et al. 2009), if Assumption 1 does not hold we might expect a different estimator to be more appropriate.

**Assumption 1.** Let $w^*$ be the true parameter vector after rotating $X$ to the Principal Component Analysis (PCA) coordinate system. There exists $1 \leq J \leq \min\{n, p\}$ and $c \in (0, 1/2)$ such that

(A1) the $J$-th largest eigenvalue of the covariance matrix $\lambda_J > 0$, 

64
(A2) the ridge constraint is active: \( t \leq (1 - c) \sum_{j=1}^{J}(w_j^*)^2 \),

(A3) the errors \( \varepsilon_i, i = 1, \ldots, n \) have zero mean, are independent and their variances are bounded by \( \sigma^2 > 0 \).

To shed some light onto Assumption (A2), consider the noiseless case where the entire signal lies in the first \( J \) principal components. Then \( c = 0 \) implies no shrinkage, while increasing \( c \) means that the amount of regularization becomes larger.

If Assumptions (A1) and (A2) do not hold, then ridge regression may not be a suitable estimator for \( \beta^* \) in Equation (4.5), independent of how we choose the size of the constraint. If, on the other hand, (A1) and (A2) do hold, the amount of required regularization can differ. In problem settings where the signal-to-noise ratio is low, \( p > n \) or where the covariance matrix of \( X \) is otherwise close to singular, the ridge constraint is active and (A2) covers the relevant section of the regularization parameter. The ridge estimator will then shrink less along directions associated with large variance. If the data are full rank and the noise value is very low, shrinkage may be unnecessary and the OLS estimator may be more appropriate. This issue is discussed in Heinze et al. (2014) were a similar bound for OLS is derived, which is the relevant bound if the ridge constraint is not active and (A2) does not apply.

We now present Theorem 4.2 which states that the expected difference between the coefficients \( \hat{\beta} \) returned by Loco and the full ridge regression solution is bounded.

**Theorem 4.2.** Under Assumption 1, \( \exists n_0(\xi) \) for all \( \xi > K(\delta + (p - \tau) / \epsilon^r) \) such that for all \( n \geq n_0 \) with probability at least \( 1 - \xi \)

\[
\mathbb{E}_e(\|\hat{\beta}^{rr} - \hat{\beta}\|^2) \leq \frac{5K}{c\lambda_J} \left( \frac{1}{(1 - \rho)^2} - 1 \right) R(\hat{X}\hat{\beta}^{rr})
\]

where \( \rho = C \sqrt{r \log(2r/\delta) / (K-1)^2} \), \( r = \text{rank}(X) \), \( \lambda_J \) denotes the \( J \)th largest non-zero eigenvalue of the covariance matrix and \( R(\hat{X}\hat{\beta}^{rr}) \) is the risk of the ridge estimator (see e.g. (Lu et al. 2013)). The expectation is conditional on the random projection as the uncertainty coming from the SRHT is captured in the probability with which the statement holds.

The exact value of \( n_0 \) depends on \( \xi \) and the exact form is given in Heinze et al. (2014) together with the proof.
The bound above scales with the number of workers, $K$ and inversely with $(1 - \rho)^2$, which measures the quality of the random feature representation. The latter is improved (for a fixed $\tau_k$) by increasing $K$ although this has the additional effect of increasing the computational overhead per worker which scales as $O((K - 1)\tau_k)$.

4.5 RELATED WORK

Recently, several methods have been proposed for parallelizing convex optimization. Among these, hogwild! (Niu et al. 2011), ASyncDA and ASyncAdaGrad (Duchi et al. 2013) have shown that large speedups are possible with asynchronous gradient updates when data is sparse. These methods rely on the idea that if the number of non-zero coordinates in each stochastic gradient evaluation is small compared to $p$, workers updating the same solution vector in parallel will rarely propose conflicting updates. As such each worker is allowed to update the solution asynchronously without the need for locking, provided the delay of any processor is not too great.

Similar requirements on sparsity have been used to parallelize coordinate descent (Richtarik et al. 2013). Other approaches rely on alternative, but related conditions which require the spectral norm of the data – which captures the size of dependencies between features – to be small (Bradley et al. 2011; Scherrer et al. 2012).

Whilst sparsity is a natural and common feature of large data sets, in some fields the data collected is dense with many correlated features. Furthermore, in the setting where $n \ll p$, SGD can converge slowly. Under these conditions we might expect the performance of the above mentioned approaches to suffer. Notably, Loco makes no assumptions about sparsity since each block sees a representation of the remaining features such that updates to the individual solution vectors are not independent of the rest of the data set. Loco does not require synchronization between workers since each worker may only update its own part of the solution vector. Although a bound on the maximum delay is not required or assumed, the total running time of Loco will naturally be dictated by the slowest worker.

Most of the above mentioned approaches implement parallelism on a multi-core architecture with shared memory. For distributed optimization, communicating results between workers introduces overhead. Several communication strategies for distributed coordinate descent
were discussed in Richtarik et al. (2013). In contrast, Loco requires computing random projections for each block and communicating them once and so no additional synchronization or communication are required until prediction time.

Parallel estimation has also been considered in the case of kernel ridge regression (Y. Zhang et al. 2013). It has been shown that randomly splitting and distributing the samples among workers and averaging their estimates achieves a superlinear speedup whilst retaining optimality up to a number of workers which is problem dependent. Conceptually, Loco is perhaps most similar to this approach, although clearly the usual i.i.d. assumption on the observations does not hold for partitioning the features.

4.6 EXPERIMENTAL RESULTS

Typically parallel optimization methods are evaluated on extremely sparse data sets (Niu et al. 2011; Duchi et al. 2013) or uncorrelated simulated data (Richtarik et al. 2013; Agarwal et al. 2011; Peng et al. 2013) which fulfills the types of assumptions on sparsity or low-correlations between features necessary to obtain theoretical results.

Since we do not make these assumptions, we aim to show that Loco is robust to correlations between features which can be accounted for using random projections.

4.6.1 Data generation

We generated each observation in the following way. First, an $r$-dimensional standard Gaussian vector is generated

$$z \in \mathbb{R}^r \sim \mathcal{N}(0, I_r).$$

We then sample a diagonal matrix of eigenvalues,

$$\Lambda \in \mathbb{R}^{r \times r} = \text{diag}(\lambda_1, \ldots, \lambda_r)$$

where we can choose each $\lambda_i$ so that it has a flat (i.e. $\lambda_1 = \ldots = \lambda_r$) or decaying (i.e. $\lambda_1 > \ldots > \lambda_r$) spectrum, along with an orthonormal matrix of eigenvectors $Q \in \mathbb{R}^{p \times r}$ with columns

$$Q = [q_1, \ldots, q_r]$$
which have the additional structure that only one entry in each row of $Q$ is non-zero. $Q$ can easily be generated from the QR decomposition of a $p \times r$ random matrix. For example, in MATLAB this is achieved by

$$ [Q, \sim] = \text{qr}(\text{randn}(p,r),0); $$

Now, we construct $x \in \mathbb{R}^p = QA^{1/2}z$ so that $\mathbb{E}[zz^\top] = QAQ^\top$. This ensures that features are correlated according to the non-zero entries in each column of $Q$. In other words, the features indexed by the non-zero entries of each $q$ are correlated with each other. A feature $j$ indexed by $q_r$ and a feature $j'$ indexed by $q_r'$ are uncorrelated. If we arrange the non-zero entries in each $q$ such that they form contiguous regions, $\mathbb{E}[zz^\top]$ would be a block-diagonal covariance matrix.

We initialize the true regression vector as $\beta^* = 0$. We then choose $r$ indices, each corresponding to a randomly chosen coordinate indexed by each of the columns of $Q$. Denoting these indices by the subscript, $\beta^*_Q$, we have $\beta^*_Q \sim \text{Unif}[-2, 2]$. The remaining $p-r$ coordinates are drawn from $\mathcal{N}(0,0.2)$. This ensures that only a single coordinate from each of the blocks of correlated features contributes a large amount to the response. Finally the response is, $y = x^\top \beta^* + \sigma^2$.

Using this routine, we can quickly generate very high dimensional and dense vectors $x$ which have an interesting, low-dimensional structure and dependencies between features.

4.6.2 Implementation details

We implemented Loco in Python making use of fast packages for random projections and ridge regression. For the random projection we used the DCT implemented in FFTW\(^3\). The ridge regression solver is implemented in scikit-learn which outperforms SGD for $p \gg n$. We ran Loco on the BRUTUS cluster\(^4\) where each worker “communicates” by writing its random projections to a Lustre file system to which each worker is connected via InfiniBand network and as such enables very fast simultaneous reads and writes. We consider each machine to be a worker and since the cluster is heterogeneous we do not exploit any further local parallelization. For comparison, we modified the loss and gradient computations of HOGWILD\(^5\) to perform ridge regression.

\(^3\)[http://www.fftw.org]
\(^4\)[http://en.wikipedia.org/wiki/Brutus_cluster]
and ran it on a single Xeon E3-1275 V with 32GB of RAM (4 cores, comparable but slightly slower than the test machine of (Niu et al. 2011)). The difference between setups makes a comparison of absolute timing inaccurate but relative speedup for each method is comparable.

4.6.3 Simulated data

We consider two large-scale simulated problems. The data is generated from a Gaussian distribution with mean zero and a block-wise covariance matrix such that the features are not independent and the block structure is not known to the algorithm a priori. The data simulation method is fully described in Section 4.6.1. The matrix of simulated data is fully dense, so there is no sparsity that could be exploited for speedups which directly handicaps Hogwild!.

The first scenario we consider is \( n = 1000, p = 131,072 \) (200M non-zeros, testing set is the same size as the training set) and has rank \( r = 20 \). The purpose of this experiment is to compare Loco against Hogwild! in a setting where memory is not a limiting factor and so Hogwild! can comfortably parallelize the problem and run on a single
Figure 4.3: Relative speedup comparison between Loco and Hogwild! for \( p = 131,072 \).

machine. We aim to compare the speedup of both methods when the data is dense. The behaviour of Hogwild! should also be similar to the behaviour of AsyncDA (Duchi et al. 2013).

According to Theorem 4.2, increasing \( \tau_k \) will improve the prediction error but also increase the computational time. To give a meaningful comparison between the methods we vary \( \tau_k \) and report the time spent training the model and the prediction error achieved. Since for different number of workers, \( p_k = p/K \) is different, the random projection dimension, \( \tau_k \) is chosen relative to \( p_k \), i.e. \( 0 \leq \tau_k \leq 0.1p_k \). As a baseline we show the results for a standard ridge regression solver. Prediction error is normalized by this result to obtain relative prediction error.

Figure 4.2 shows the average results over 5 trials for \( K = \{2, 4, 8, 16\} \). Comparable test error to full ridge regression (horizontal dotted line at 1) is obtained with very low-dimensional random projections, even for \( K = 16 \). The largest error for each setting occurs when \( \tau_k = 0 \), i.e. the data set is partitioned but no random features are added, and increases as \( K \) increases. This highlights the importance of accounting
for dependencies between blocks of features and justifies our random projection approach to distributing features across workers.

Figure 4.3 compares the relative speedup for increasing $K$ for Loco and Hogwild!. Hogwild! exhibits linear speedup for up to 4 threads but no speedup when more are added. Loco exhibits an almost linear speedup with the number of workers – with $K = 16$ we obtain a $6 \times$ speedup over $K = 2$. The absolute running time of Loco was faster and the timings for Hogwild! ignore a large constant overhead for file loading.

We must stress that this experiment is designed so that the theoretical assumptions underpinning Hogwild! are violated. Clearly if the data were very sparse, Hogwild! would exhibit large speedups as investigated in Niu et al. (2011) and Duchi et al. (2013). However, for very large data sets it would still be limited by the number of cores and amount of memory available to a single machine. In such scenarios it may be advantageous to combine Loco and Hogwild!.

Figure 4.4: Prediction error against time for $p = 1.05M$. 
The second scenario we consider is $n = 5000$, $p = 1.05M$ and $r = 2000$. Since the data is fully dense, there are over 5 billion non-zeros\(^6\). Now the size of the data starts to become impractical for a single machine (training data is $\geq 40$GB) and the distributed nature of Loco is advantageous. We just report the performance of Loco for $K = \{64, 128, 256\}$.

Figure 4.4 shows the normalized mean-squared prediction error against time. Loco takes just 25 seconds to achieve a prediction error of 0.11 for $K = 128$. Figure 4.5 shows that we obtain an almost $2\times$ speedup when increasing from 64 to 256 workers. The reason for the small speedup can be observed in Figure 4.6. The x-axis plots the additional overhead in terms of the original data set size required to obtain a particular error. Since the size of the random feature representation for each worker scales as $(K - 1)\tau_k$, for $K = 256$, we require almost $20p$ features to match the performance of $K = 64$. This illustrates the natural tradeoff between parallelism and additional computational cost

---

\(^6\) Comparable in number with the experiment size of Richtarik et al. (2013) and Peng et al. (2013) with the key difference that we do not impose block sparsity in the data like Richtarik et al. (2013) and we do not simply sample $X$ from $\mathcal{N}(0, 1)$ like Peng et al. (2013).
from adding enough random features to ensure a good representation as discussed in Section 4.4.

4.7 DISCUSSION

In this chapter we have presented Loco, a simple algorithm for distributed ridge regression – requiring minimal communication and no synchronization – based on random projections. We have shown theoretically and empirically that Loco achieves small additional error compared with the optimal ridge regression solution. It obtains near linear speedups with the number of workers without making any additional assumptions about sparsity in the data. Furthermore, in principle any fast ridge regression solver can be used in conjunction with Loco to further speed up learning in each individual worker according to whichever computing architecture or data assumptions apply. It should be noted that Loco is a complementary rather than competing approach to methods such as HOGWILD!. In particular, a large but sparse problem could be solved on a cluster of multi-core machines extremely quickly by combining Loco with HOGWILD!.
Although the results in this chapter are specific for ridge regression, the same principles can be generalized to other convex optimization problems. In Heinze, McWilliams, and Meinshausen (2016) Loco is extended to an algorithm that also applies to other smooth, convex \( \ell_2 \) penalized loss minimization problems like logistic regression.

As mentioned in the introduction, distributed optimization – where no single worker sees all of the data – is a natural paradigm when preserving privacy is required. Additionally, the class of J-L projections that we use have been shown to preserve differential privacy (Blocki et al. 2012). The privacy preserving properties of Loco are explored further in Heinze et al. (2015).
SCALABLE ADAPTIVE STOCHASTIC OPTIMIZATION

The size of modern data sets for machine learning prohibits the direct minimization of the loss on the full training set. Therefore, stochastic gradient descent methods (SGD, Section 2.5), that only consider the gradient at one data point per iteration are used. This leads to inherent noise in the gradients as only an approximation of the full loss is evaluated. Therefore the noise considered in this chapter is directly related to the large-scale of the data set.

To improve the performance of stochastic optimization, so-called adaptive stochastic optimization algorithms have gained popularity for large-scale convex and non-convex optimization problems. Among these, ADAGRAD (Duchi et al. 2011) and its variants have received particular attention and have proven among the most successful algorithms for training deep networks. Although these problems are inherently highly non-convex, recent work has begun to explain the success of such algorithms (Balduzzi 2016).

The most commonly used and studied variant maintains a diagonal matrix approximation to second order information by accumulating past gradients which are used to tune the step size adaptively. In certain situations the full-matrix variant of ADAGRAD is expected to attain better performance, however in high dimensions it is computationally impractical. We present ADA-LR and RADA GRAD two computationally efficient approximations to full-matrix ADAGRAD based on randomized dimensionality reduction (Section 2.6). They are able to capture dependencies between features and achieve similar performance to full-matrix ADAGRAD but at a much smaller computational cost. We show that the regret of ADA-LR is close to the regret of full-matrix ADAGRAD which can have an up-to exponentially smaller dependence on the dimension than the diagonal variant. Empirically, we show that ADA-LR and RADA GRAD perform similarly to full-matrix ADAGRAD. On the task of training convolutional neural networks as well as Recurrent

\footnote{Parts of this chapter appear in Krummenacher et al. (2016)}
Neural Networks (RNNs), RadaGrad achieves faster convergence than diagonal AdaGrad.

5.1 INTRODUCTION

AdaGrad adaptively sets the learning rate for each feature by means of a time-varying proximal regularizer. The most commonly studied and utilised version considers only a diagonal matrix proximal term. As such it incurs almost no additional computational cost over standard SGD. However, when the data has low effective rank the regret of AdaGrad may have a much worse dependence on the dimensionality of the problem than its full-matrix variant (which we refer to as Ada-full). Such settings are common in high dimensional data where there are many correlations between features and can also be observed in the convolutional layers of neural networks. The computational cost of Ada-full is substantially higher than that of AdaGrad -- it requires computing the inverse square root of the matrix of gradient outer products to evaluate the proximal term which grows with the cube of the dimension. As such it is rarely used in practice.

In this chapter we propose two methods that approximate the proximal term used in Ada-full drastically reducing computational and storage complexity with little adverse affect on optimization performance. First, in Section 5.3.1 we develop Ada-lr, a simple approximation using random projections. This procedure reduces the computational complexity of Ada-full by a factor of $p$ but retains similar theoretical guarantees. In Section 5.3.4 we systematically profile the most computationally expensive parts of Ada-lr and introduce further randomized approximations resulting in a truly scalable algorithm, RadaGrad. In Section 5.3.5 we outline a simple modification to RadaGrad -- reducing the variance of the stochastic gradients -- which greatly improves practical performance.

Finally we perform an extensive comparison between the performance of RadaGrad with several widely used optimization algorithms on a variety of deep learning tasks. For image recognition with Convolutional Neural Networks (CNNs) and language modeling with Recurrent Neural Networks (RNNs) we find that RadaGrad and in particular its variance-reduced variant achieves faster convergence.
5.1 Introduction

5.1.1 Related work

Motivated by the problem of training deep neural networks, very recently many new adaptive optimization methods have been proposed. Most computationally efficient among these are first order methods similar in spirit to AdaGrad, which suggest alternative normalization factors (Kingma et al. 2014; Neyshabur et al. 2015; Dauphin et al. 2015). Several authors propose efficient stochastic variants of classical second order methods such as limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) (Byrd et al. 2016b; Keskar et al. 2015). Efficient algorithms exist to update the inverse of the Hessian approximation by applying the matrix-inversion lemma or directly updating the Hessian-vector product using the “double-loop” algorithm but these are not applicable to AdaGrad style algorithms. In the convex setting these methods can show great theoretical and practical benefit over first order methods but have yet to be extensively applied to training deep networks.


Finally Gonen et al. (2015) propose a randomized preconditioner for SGD. However, their approach requires access to all of the data at once in order to compute the preconditioning matrix which is impractical for training deep networks. Luo et al. (2016) propose a theoretically motivated algorithm similar to Ada-LR and a faster alternative based on Oja’s rule to update the SVD but without corresponding guarantees.
5.1.2 Problem setting

The problem we consider is online stochastic optimization where the goal is, at each step, to predict a point $\beta_t \in \mathbb{R}^p$ which achieves low regret with respect to a fixed optimal predictor, $\beta^{\text{opt}}$, for a sequence of (convex) functions $F_t(\beta)$. After $T$ rounds, the regret can be defined in the following way

$$R(T) = \sum_{t=1}^{T} F_t(\beta_t) - \inf_{\beta} \sum_{t=1}^{T} F_t(\beta)$$

$$= \sum_{t=1}^{T} F_t(\beta_t) - \sum_{t=1}^{T} F_t(\beta^{\text{opt}}). \quad (5.1)$$

We will consider functions $F_t$ of the form $F_t(\beta) := f_t(\beta) + \varphi(\beta)$ where $f_t$ and $\varphi$ are convex loss and regularization functions respectively. Throughout, the vector $g_t \in \nabla f_t(\beta_t)$ refers to a particular subgradient of the loss function. Standard first order methods update $\beta_t$ at each step by moving in the opposite direction of $g_t$ according to a step-size parameter, $\eta$. The AdaGrad family of algorithms (Duchi et al. 2011) instead use an adaptive learning rate which can be different for each feature. This is controlled using a time-varying proximal term which we briefly review. Defining

$$G_t = \sum_{i=1}^{t} g_i g_i^\top$$

and

$$H_t = \delta I_p + (G_{t-1} + g_t g_t^\top)^{1/2},$$

the Ada-Full proximal term is given by

$$\psi_t(\beta) = \frac{1}{2} \langle \beta, H_t \beta \rangle.$$

The update at time $t + 1$ is given by the following optimization problem

$$\beta_{t+1} = \arg \min_{\beta} \left\{ \eta \langle \bar{g}_t, \beta \rangle + \eta \varphi(\beta) + \frac{1}{t} \psi_t(\beta) \right\}, \quad (5.2)$$

with $\bar{g}_t = \frac{1}{t} \sum_{i=1}^{t} g_i$, the average gradient vector at time $t$. 

78
Clearly when $p$ is large, constructing $G$ and finding its root and inverse at each iteration is impractical. In practice, rather than the full outer product matrix, AdaGrad uses a proximal function consisting of the diagonal of $G_t$,

$$\psi_t(\beta) = \frac{1}{2} \left\langle \beta, (\delta I_p + \text{diag}(G_t)^{1/2}) \beta \right\rangle.$$ 

Although the diagonal proximal term is computationally cheaper, it is unable to capture dependencies between coordinates in the gradient terms. Despite this, AdaGrad has been found to perform very well empirically. One reason for this is modern high-dimensional data sets are typically also very sparse. Under these conditions, coordinates in the gradient are approximately independent. In fact, under certain assumptions about data sparsity, AdaGrad has been shown to achieve optimal regret in a minimax sense (Duchi et al. 2013).

### 5.2 Stochastic Optimization in High Dimensions

AdaGrad has attractive theoretical and empirical properties and adds essentially no overhead above a standard first order method such as SGD. It begs the question, what we might hope to gain by introducing additional computational complexity. In order to motivate our contribution, we first present an analogue of the discussion in Duchi et al. (2013) focussing on when data is high-dimensional and dense. We argue that if the data has low-rank (rather than sparse) structure Ada-Full can effectively adapt to the intrinsic dimensionality. We also show in Section 5.3.1 that Ada-LR has the same property.

First, we review the theoretical properties of AdaGrad algorithms.

**Proposition 5.1** (Regret of AdaGrad: Corollary 6 from Duchi et al. (2011)). AdaGrad achieves the following regret:

$$R_D(T) \leq 2\|\beta^{opt}\|_\infty \cdot \text{tr} \left( \text{diag}(G_T)^{1/2} \right) + \delta\|\beta^{opt}\|_1$$

**Proposition 5.2** (Regret of Ada-Full: Corollary 11 from Duchi et al. (2011)). Ada-Full achieves the following regret:

$$R_F(T) \leq 2\|\beta^{opt}\|_2 \cdot \text{tr} \left( G_T^{1/2} \right) + \delta\|\beta^{opt}\|_2$$

The major difference between $R_D(T)$ and $R_F(T)$ is the inclusion of the final full-matrix and diagonal proximal term, respectively. Under
a sparse data generating distribution AdaGrad achieves an up-to exponential improvement over SGD which is optimal in a minimax sense (Duchi et al. 2013). While data sparsity is often observed in practice in high-dimensional data sets (particularly web/text data) many other problems are dense. Furthermore, in practice applying AdaGrad to dense data results in a learning rate which tends to decay too rapidly. It is therefore natural to ask how dense data affects the performance of Ada-full.

For illustration, consider when the data points \( x_i \) are sampled i.i.d. from a Gaussian distribution \( P_X = \mathcal{N}(0, \Sigma) \). The resulting variable will clearly be dense. A common feature of high dimensional data is low effective rank defined for a matrix \( \Sigma \) as 
\[
\text{r}(\Sigma) = \frac{\text{tr}(\Sigma)}{\| \Sigma \|} \leq \text{rank}(\Sigma) \leq p.
\]
Low effective rank implies that \( r \ll p \) and therefore the eigenvalues of the covariance matrix decay quickly. We will consider distributions parameterised by covariance matrices \( \Sigma \) with eigenvalues \( \lambda_j(\Sigma) = \lambda_0 j^{-\alpha} \) for \( j = 1, \ldots, p \).

Functions of the form \( F_t(\beta) = \frac{1}{2} (y_t - \beta^\top x_t)^2 \) have gradients \( \| g_t \| \leq M \| x_t \| \). For example, the least squares loss \( F_t(\beta) = \frac{1}{2} (y_t - \beta^\top x_t)^2 \) has gradient \( g_t = x_t (y_t - x_t^\top \beta) = x_t \epsilon_t \), such that \( \| \epsilon_t \| \leq M \). Let us consider the effect of distributions parameterized by covariance matrices \( \Sigma \) on the proximal terms of full, and diagonal AdaGrad. Plugging \( X \) into the proximal terms of (5.3) and taking expectations with respect to \( P_X \) we obtain for AdaGrad:

\[
\mathbb{E} \text{tr}\left( \text{diag}(G_T)^{1/2} \right) \leq \sum_{j=1}^p \sqrt{M^2 \mathbb{E} \sum_{t=1}^T x_{t,j}^2} \leq p M \sqrt{T}, \quad (5.5)
\]

where the first inequality is from Jensen and the second is from noticing the sum of \( T \) squared Gaussian random variables is a \( \chi^2 \) random variable. For Ada-full we obtain

\[
\mathbb{E} \text{tr}\left( \left( \sum_{t=1}^T g_t g_t^\top \right)^{1/2} \right) \leq M \sqrt{T \lambda_0} \sum_{j=1}^p j^{-\alpha/2}. \quad (5.6)
\]

We can consider the effect of fast-decaying spectrum: for \( \alpha \geq 2 \), \( \sum_{j=1}^p j^{-\alpha/2} = O(\log p) \) and for \( \alpha \in (1, 2) \), \( \sum_{j=1}^p j^{-\alpha/2} = O(p^{1-\alpha/2}) \).

When the data (and thus the gradients) are dense, yet low rank, Ada-full is able to adapt to this structure. On the contrary, although AdaGrad is computationally practical, in the worst case it may have exponentially worse dependence on the data dimension (\( p \) compared
with $\log p$). In fact, the discrepancy between the regret of ADA-FULL and that of ADAGRAD is analogous to the discrepancy between ADAGRAD and SGD for sparse data.

Algorithm 5.1 ADA-LR

Input: $\eta > 0$, $\delta \geq 0$, $\tau$

1: for $t = 1 \ldots T$ do
2: Receive $g_t = \nabla f_t(\beta_t)$.
3: $G_t = G_{t-1} + g_t g_t^\top$
4: Project: $\tilde{G}_t = G_t \Pi$
5: $Q R = \tilde{G}_t$ # QR-decomposition
6: $\tilde{B} = \tilde{Q}^\top G_t$
7: $\tilde{U}, \tilde{S}, \tilde{V} = \tilde{B}$ # SVD
8: $\beta_{t+1} = \beta_t - \eta \tilde{V}(\tilde{S}^{1/2} + \delta I)^{-1} \tilde{V}^\top g_t$
10: end for
Output: $\beta_T$

Algorithm 5.2 RadaGrad

Input: $\eta > 0$, $\delta \geq 0$, $\tau$

1: for $t = 1 \ldots T$ do
2: Receive $g_t = \nabla f_t(\beta_t)$.
3: Project: $\tilde{g}_t = \Pi g_t$
4: $\tilde{G}_t = \tilde{G}_{t-1} + g_t \tilde{g}_t^\top$
5: $\tilde{Q}_t, \tilde{R}_t \leftarrow qr\text{-update}(\tilde{Q}_{t-1}, \tilde{R}_{t-1}, g_t, \tilde{g}_t)$
6: $\tilde{B} = \tilde{G}_t^\top \tilde{Q}_t$
7: $\tilde{U}, \tilde{S}, \tilde{W} = \tilde{B}$ # SVD
8: $\tilde{V} = \tilde{W} \tilde{Q}^\top$
9: $\beta_{t+1} = \beta_t - \eta \tilde{V}(\tilde{S}^{1/2} + \delta I)^{-1} \tilde{V}^\top g_t$
10: end for
Output: $\beta_T$
5.3 Approximating ADA-Full Using Random Projections

It is clear that in certain regimes, ADA-Full provides stark optimization advantages over ADAGrad in terms of the dependence on \( p \). However, ADA-Full requires maintaining a \( p \times p \) matrix, \( G \) and computing its square root and inverse. Therefore, computationally the dependence of ADA-Full on \( p \) scales with the cube which is impractical in high dimensions.

A naïve approach would be to simply reduce the dimensionality of the gradient vector, \( \tilde{g}_t \in \mathbb{R}^\tau = \Pi g_t \). ADA-Full is now directly applicable in this low-dimensional space, returning a solution vector \( \hat{\beta}_t \in \mathbb{R}^\tau \) at each iteration. However, for many problems, the original coordinates may have some intrinsic meaning or in the case of deep networks, may be parameters in a model. In which case it is important to return a solution in the original space. Unfortunately in general it is not possible to recover such a solution from \( \hat{\beta}_t \) (L. Zhang et al. 2012).

Instead, we consider a different approach to maintaining and updating an approximation of the ADAGrad matrix while retaining the original dimensionality of the parameter updates \( \beta \) and gradients \( g \).

5.3.1 Randomized low-rank approximation

As a first approach we approximate the inverse square root of \( G_t \) using a fast randomized Singular Value Decomposition (SVD) Halko et al. (2011). We proceed in two stages: First we compute an approximate basis \( Q \) for the range of \( G_t \). Then we use \( Q \) to compute an approximate SVD of \( G_t \) by forming the smaller dimensional matrix \( B = Q^\top G_t \) and then compute the low-rank SVD \( \tilde{U} \Sigma \tilde{V}^\top = B \) and set \( U = QU \). This is faster than computing the SVD of \( G_t \) directly if \( Q \) has few columns.

An approximate basis \( Q \) can be computed efficiently by forming the matrix \( Y = G \Pi \) by means of a structured random projection and then constructing an orthonormal basis for the range of \( Y \) by QR-decomposition. The randomized SVD allows us to quickly compute the square root and pseudo-inverse of the proximal term \( H_t \) by setting \( \tilde{H}_t^{-1} = \tilde{V}(\tilde{S}^{1/2} + \delta I)^{-1}\tilde{V}^\top \). We call this approximation ADA-LR and describe the steps in full in Algorithm 5.1.
In practice, using a structured random projection such as the SRFT leads to an approximation of the original matrix, \(G_t\) of the following form
\[
\|G_t - QQ^\top G_t\| \leq \epsilon,
\]
with high probability (Halko et al. 2011) where \(\epsilon\) depends on \(\tau\), the number of columns of \(Q\); \(p\) and the \(\tau^{th}\) singular value of \(G_t\). Briefly, if the singular values of \(G\) decay quickly and \(\tau\) is chosen appropriately, \(\epsilon\) will be small (this is stated more formally in Theorem 5.3). We leverage this result to derive the following regret bound for Ada-LR.

**Theorem 5.3.** Defining \(\epsilon = \sqrt{1 + 7p/\tau \cdot \sigma_{k+1}}\) Ada-LR achieves the following regret with failure probability at most \(O\left(k^{-1}\right)\)
\[
R_{LR}(T) \leq 2\|\beta^{opt}\| \text{tr}\left(G_t^{1/2}\right) + (2\tau \sqrt{\epsilon} + \delta)\|\beta^{opt}\|,
\]
(5.7)
where \(\sigma_{k+1}\) is the \(k^{th}\) largest singular value of \(G_t\), and \(4\left[\sqrt{k} + \sqrt{8 \log(kn)}\right]^2 \leq \tau \leq p\).

Due to the randomized approximation we incur an additional \(2\tau \sqrt{\epsilon}\|\beta^{opt}\|\) compared with the regret of Ada-Full (Equation 5.3). So, under the earlier stated assumption of fast decaying eigenvalues we can use an identical argument as in Equation (5.6) to similarly obtain a dimension dependence of \(O\left(\log p + \tau\right)\).

### 5.3.2 Proof of Theorem 5.3

The following proof is based on the proof for Theorem 7 in Duchi et al. (2011). The key difference is that instead of having the square root and (pseudo-)inverse of the full matrix \(G_t: G_t^{1/2}\) and \(S^\dagger_t\) we have the approximate square root and inverse based on the randomized SVD (Halko et al. 2011): \(\hat{S}_i = (QQ^\top G_i)^{1/2}\) and \(\hat{S}_i^\dagger = (QQ^\top G_i)^{-1/2}\). Essentially we use the proximal function \(\psi_i = \langle x, \hat{S}_i x \rangle\) or \(\psi_i = \langle x, \hat{H}_i x \rangle\) where we set \(\hat{H}_i = \delta I + \hat{S}_i\). Here \(Q\) is the approximate basis for the range of the matrix \(G_t\) (Halko et al. 2011).

We first state results from Duchi et al. (2011) that we will need in our analysis. We also prove two lemmas that are used in the proof of Theorem 5.3.
5.3.2.1 Supporting Results

**Lemma 5.1** (Proposition 2 from Duchi et al. (2011)).

\[
R(T) := \sum_{t=1}^{T} f_t(\beta_t) + \varphi(\beta_t) - f_t(\beta_{opt}) - \varphi(\beta_{opt})
\]

\[
\leq \frac{1}{\eta} \psi_T(\beta_{opt}) + \frac{\eta}{2} \sum_{t=1}^{T} \| f'_t(\beta_t) \|_{\psi^{-1}}^2
\]

**Lemma 5.2** (Lemma 8 from Duchi et al. (2011)). Let \( B \succeq 0 \). For any \( \nu \) such that \( B - \nu g g^\top \succeq 0 \) the following holds

\[
2 \text{tr} \left( (B - \nu g g^\top)^{1/2} \right) \leq 2 \text{tr} \left( B^{1/2} \right) - \nu \text{tr} \left( B^{1/2} g g^\top \right)
\]

**Lemma 5.3** (Lemma 9 from Duchi et al. (2011)). Let \( \delta \geq \| g \|_2 \) and \( A \succeq 0 \), then

\[
\langle g, (\delta I + A^{1/2})^{-1} g \rangle \leq \langle g, ((A + g g^\top)^{1/2})^{-1/2} g \rangle
\]

We also proof the following facts about the relationship between \( G \) and \( \tilde{G}^{-1/2} \).

**Lemma 5.4.** Defining \( \tilde{G}^{-1/2} = (Q Q^\top G)^{-1/2} \) we have

(I) \( \tilde{G}^{-1/2} G = (G^{-1}(Q Q^\top) G^2)^{1/2} \),

(II) \( \text{tr} \left( (G^{-1}(Q Q^\top) G^2)^{1/2} \right) = \text{tr} \left( \tilde{G}^{1/2} \right) \).

**Proof.** By direct computation we have for (I)

\[
\tilde{G}^{-1/2} G = (Q Q^\top G)^{-1/2} G
\]

\[
= ((Q Q^\top G)^{-1} G^2)^{1/2}
\]

\[
= (G^{-1}(Q Q^\top)^{-1} G^2)^{1/2}
\]

\[
= (G^{-1}(Q Q^\top) G^2)^{1/2}.
\]

and for (II)

\[
\text{tr} \left( (G^{-1}(Q Q^\top) G^2)^{1/2} \right) = \text{tr} \left( (Q^\top G Q)^{1/2} \right)
\]

\[
= \text{tr} \left( (Q Q^\top G)^{1/2} \right)
\]

\[
= \text{tr} \left( \tilde{G}^{1/2} \right).
\]
We also require the following Lemma which bounds the sequence of proximal terms by the trace of the final $\tilde{G}^{-1/2}$.

**Lemma 5.5** (Based on Lemma 10 in Duchi et al. (2011)).

$$\sum_{t=1}^{T} \langle g_t, \tilde{G}_t^{-1/2} g_t \rangle \leq 2 \sum_{t=1}^{T} \langle g_t, \tilde{G}_T^{-1/2} g_t \rangle = 2 \text{tr} (\tilde{G}_T^{1/2}). \quad (5.8)$$

**Proof.** We set up the following proof by induction. In the base case:

$$\langle g_1, \tilde{G}_1^{-1/2} g_1 \rangle = \text{tr} (\tilde{G}_1^{-1/2} g_1 g_1^\top) = \text{tr} (\tilde{G}_1^{1/2}) \leq 2 \text{tr} (\tilde{G}_1^{1/2}),$$

where we have used (II).

Now, assuming that the lemma is true for $T - 1$, we get:

$$\sum_{t=1}^{T} \langle g_t, \tilde{G}_t^{-1/2} g_t \rangle \leq 2 \sum_{t=1}^{T} \langle g_t, \tilde{G}_{T-1}^{-1/2} g_t \rangle + \langle g_T, \tilde{G}_T^{-1/2} g_T \rangle.$$

Now using that $\tilde{G}_T^{-1/2}$ does not depend on $t$ and (II):

$$\sum_{t=1}^{T-1} \langle g_t, \tilde{G}_{T-1}^{-1/2} g_t \rangle = \text{tr} (\tilde{G}_{T-1}^{-1/2} G_{T-1}) = \text{tr} (\tilde{G}_{T-1}^{1/2}).$$

Therefore we get

$$\sum_{t=1}^{T} \langle g_t, \tilde{G}_t^{-1/2} g_t \rangle \leq 2 \text{tr} (\tilde{G}_{T-1}^{1/2}) + \langle g_T, \tilde{G}_T^{-1/2} g_T \rangle. \quad (5.9)$$

We can rewrite

$$\text{tr} (\tilde{G}_{T-1}^{1/2}) = \text{tr} \left( \left( Q_{T-1} Q_{T-1}^\top G_T - Q_{T-1} Q_{T-1}^\top g_T g_T^\top \right)^{1/2} \right) \quad (5.10)$$

Now since range$(Q_{T-1}) \subset$ range$(Q_T)$ and Proposition 8.5 in Halko et al. (2011) we can use Lemma 5.2 with $\nu = 1$ and $g = g_t$ to obtain:

$$2 \text{tr} (\tilde{G}_{T-1}^{1/2}) + \langle g_T, \tilde{G}_T^{-1/2} g_T \rangle \leq 2 \text{tr} (\tilde{G}_T^{1/2}) \quad (5.11)$$

\( \square \)

We are now ready to prove Theorem 5.3.
Proof of Theorem 5.3

Proof of Theorem 5.3. Inspecting Lemma 5.1:

\[ R(T) \leq \frac{1}{η} \psi_T(\beta^{\text{opt}}) + \frac{1}{η} \sum_{t=1}^{T} \|f'_t(\beta_t)\|_{\psi_{T-1}^*}^2 \]

we first bound the term \( \sum_{t=1}^{T} \|f'_t(\beta_t)\|_{\psi_{T-1}^*}^2 \).

From Duchi et al. (2011, Proof of Theorem 7) we have that the squared dual norm associated with \( \psi_t \) is

\[ \|x\|_{\psi_t^*}^2 = \langle x, (\delta I + (QQ^T G_t)^{1/2})^{-1} x \rangle \]

and thus it is clear that \( \|g_t\|_{\psi_t^*}^2 \leq \langle g_t, (QQ^T G_t)^{-1/2} g_t \rangle \). Lemma 5.3 shows that \( \|g_t\|_{\psi_{t-1}^*}^2 \leq \langle g_t, S_t g_t \rangle \) as long as \( \delta \geq \|g_t\|_2 \). Lemma 5.5 then implies that

\[ \sum_{t=1}^{T} \|f'_t(\beta_t)\|_{\psi_{t-1}^*}^2 \leq 2 \text{tr} \left( \tilde{G}_T^{1/2} \right). \]

We now bound \( 2 \text{tr} \left( \tilde{G}_T^{1/2} \right) \) by \( 2(\text{tr} \left( G_T^{1/2} \right) + \tau \sqrt{\epsilon}) \):

\[ \text{tr} \left( \tilde{G}_T^{1/2} \right) - \text{tr} \left( G_T^{1/2} \right) = \text{tr} \left( \tilde{G}_T^{1/2} - G_T^{1/2} \right) \]

\[ = \sum_{j=1}^{τ} \left( \lambda_j(\tilde{G}_T^{1/2}) - \lambda_j(G_T^{1/2}) \right) - \sum_{j=τ+1}^{p} \lambda_j(G_T^{1/2}) \]

\[ \leq \sum_{j=1}^{τ} \left( \lambda_1(\tilde{G}_T^{1/2}) - \lambda_1(G_T^{1/2}) \right) \] (5.14)

since \( \lambda_j(\tilde{G}_T) = 0, \forall j > τ \).

Now, using the reverse triangle inequality and Theorem 2.15 we obtain

\[ \sum_{j=1}^{τ} \left( \lambda_1(G_T^{1/2}) - \lambda_1(G_T^{1/2}) \right) \leq \sum_{j=1}^{τ} \|G_T^{1/2} - G_T^{1/2}\|_2 \]

\[ \leq \sum_{j=1}^{τ} \sqrt{ε} \] (5.16)

\[ \leq τ \sqrt{ε}. \] (5.17)
It remains to show that \( \psi_T(\beta^\text{opt}) \) in Lemma 5.1 is bounded by \( (\delta + \sqrt{\epsilon} + \text{tr} \left( G_T^{1/2} \right)) \| \beta^\text{opt} \|^2 \) to get the statement of Theorem 5.3:

\[
\psi_T(\beta^\text{opt}) = \langle \beta^\text{opt}, \delta I + \left( QQ^T G_T \right)^{1/2} \beta^\text{opt} \rangle \\
\leq \| \beta^\text{opt} \|^2 \| \left( QQ^T G_T \right)^{1/2} \|_2 + \delta \| \beta^\text{opt} \|^2 \\
\leq \| \beta^\text{opt} \|^2 \left( \sqrt{\epsilon} + \| G_T^{1/2} \| \right) + \delta \| \beta^\text{opt} \|^2 \\
\leq \| \beta^\text{opt} \|^2 \left( \sqrt{\epsilon} + \text{tr} \left( G_T^{1/2} \right) \right) + \delta \| \beta^\text{opt} \|^2
\]

where we again use the reverse triangle inequality and Theorem 2.15 as above.

Finally, plugging this into the statement of Lemma 5.1 and setting \( \eta = \| \beta^\text{opt} \|_2 \) (as in Corollary 11 in Duchi et al. (2011)) we get the expression for the regret of Ada-LR as stated in Theorem 5.3. \( \square \)

### 5.3.3 Computational complexity of Ada-LR

Approximating the inverse square root decreases the complexity of each iteration from \( O(p^3) \) to \( O(\tau p^2) \). We summarize the cost of each step in Algorithm 5.1 and contrast it with the cost of Ada-Full in Table 5.1. Even though Ada-LR removes one factor of \( p \) form the runtime of Ada-Full it still needs to store the large matrix \( G_t \). This prevents Ada-LR from being a truly practical algorithm. In the following section we propose a second algorithm which directly stores a low dimensional approximation to \( G_t \) that can be updated cheaply. This allows for an improvement in runtime to \( O(\tau^2 p) \).

### 5.3.4 RadaGrad: A faster approximation

From Table 5.1, the expensive steps in Algorithm 5.1 are the update of \( G_t \) (line 3), the random projection (line 4) and the projection onto the approximate range of \( G_t \) (line 6). In the following we propose RADAGrad, an algorithm that reduces the complexity to \( O(\tau^2 p) \) by only approximately solving some of the expensive steps in Ada-LR while maintaining similar performance in practice.

To compute the approximate range \( Q \), we do not need to store the full matrix \( G_t \). Instead we only require the low dimensional matrix \( \tilde{G}_t = G_t \Pi \). This matrix can be computed iteratively by setting \( \tilde{G}_t \in \)
Table 5.1: Comparison of computational complexity in big O notation between ADA-FULL, ADA-LR and RADAGrad.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Line</th>
<th>ADA-FULL</th>
<th>ADA-LR</th>
<th>RADAGrad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi g_t$</td>
<td>3</td>
<td>$p \log \tau$</td>
<td>$p^2$</td>
<td>$p^2$</td>
</tr>
<tr>
<td>$G_t = g_t \Sigma$</td>
<td>4</td>
<td>$p^2 \log \tau$</td>
<td>$p^2 \log \tau$</td>
<td>$p^2 \log \tau$</td>
</tr>
<tr>
<td>QR-decomp</td>
<td>5</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
</tr>
<tr>
<td>$Q^\top G_t$</td>
<td>6</td>
<td>$\tau p^2$</td>
<td>$\tau p^2$</td>
<td>$\tau p^2$</td>
</tr>
<tr>
<td>SVD</td>
<td>7</td>
<td>$p^3$</td>
<td>$\tau^2 p$</td>
<td>$\tau^3$</td>
</tr>
<tr>
<td>$QW$</td>
<td>8</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
<td>$\tau^2 p$</td>
</tr>
<tr>
<td>$\beta_{t+1} =$</td>
<td>9</td>
<td>$p^2$</td>
<td>$\tau p$</td>
<td>$\tau p$</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$p^3$</td>
<td>$\tau p^2$</td>
<td>$\tau^2 p$</td>
</tr>
</tbody>
</table>

$\mathbb{R}^{n \times \tau} = \tilde{G}_{t-1} + g_t (\Pi g_t)^\top$. This directly reduces the cost of the random projection to $O(p \log \tau)$ since we only project the vector $g_t$ instead of the matrix $G_t$, it also makes the update of $\tilde{G}_t$ faster and saves storage.

We then project $\tilde{G}_t$ on the approximate range of $G_t$ and use the SVD to compute the inverse square root. Since $G_t$ is symmetric its row and column space are identical so little information is lost by projecting $\tilde{G}_t$ instead of $G_t$ on the approximate range of $G_t$. The advantage is that we can now compute the SVD in $O(\tau^3)$ and the matrix-matrix product on line 6 in $O(\tau^2 p)$. See Algorithm 5.2 for the full procedure.

The most expensive steps are now the QR decomposition and the matrix-matrix multiplications in steps 6 and 8 (see Algorithm 5.2 and Table 5.1). Since at each iteration we only update the matrix $G_t$ with the rank-one matrix $g_t \Sigma$, we can use faster rank-1 QR-updates (Golub et al. 2012) instead of recomputing the full QR decomposition. To speed up the matrix-matrix product $G_t^\top Q$ for very large problems (e.g. backpropagation in CNN), a multithreaded BLAS implementation can be used.

5.3.5 *Practical algorithms*

Here we outline several simple modifications to the RADAGrad algorithm to improve practical performance.

---

2 This idea is similar to bilinear random projections (Gong et al. 2013).
Projected Gradient Update. Whereas the regret guarantees for AdaGrad are derived for the randomized dual averaging update (Equation 5.2), in practice the $\ell_2$-norm in the projected gradient stochastic update (Zinkevich 2003)

$$\beta_{t+1} = \arg \min_\beta \|\beta - (\beta_t - \eta g_t)\|_2^2$$

is generalized to a Mahalanobis norm

$$\beta_{t+1} = \arg \min_\beta \|\beta - (\beta_t - \eta G_t^{-1/2} g_t)\|_{G_t^{1/2}}^2.$$  

For AdaGrad this leads to the update

$$\beta_{t+1} = \beta_t - \eta G_t^{-1/2} g_t.$$  

For empirical comparison we also implement Ada-lr and RADA GRAD in this way, replacing $G_t$ by $\tilde{G}_t$. See Algorithm 5.1 and Algorithm 5.2.

Corrected Update. The random projection step only retains at most $\tau$ eigenvalues of $G_t$. If the assumption of low effective rank does not hold, important information from the $p - \tau$ smallest eigenvalues might be discarded. RADA GRAD therefore makes use of the corrected update

$$\beta_{t+1} = \beta_t - \eta \tilde{V} (\tilde{S}^{1/2} + \delta I)^{-1} \tilde{V}^\top g_t - \gamma_t,$$  \hspace{0.5cm} (5.18)

where

$$\gamma_t = \eta (I - \tilde{V} \tilde{V}^\top) g_t.$$  \hspace{0.5cm} (5.19)

$\gamma_t$ is the projection of the current gradient onto the space orthogonal to the one captured by the random projection of $G_t$. This ensures that important variation in the gradient which is poorly approximated by the random projection is not completely lost. Consequently, if the data has rank less than $\tau$, $\|\gamma\| \approx 0$. This correction only requires quantities which have already been computed but greatly improves practical performance.

Variance Reduction. Variance reduction methods based on SVRG (R. Johnson et al. 2013) obtain lower-variance gradient estimates by means of computing a “pivot point” over larger batches of data. Recent work has shown improved theoretical and empirical convergence in
non-convex problems (Allen-Zhu et al. 2016) in particular in combination with AdaGRAD.

We modify RADAGrad to use the variance reduction scheme of SVRG. The full procedure is given in Algorithm 5.3. The majority of the algorithm is as RADAGrad except for the outer loop which computes the pivot point, $\mu$ every epoch which is used to reduce the variance of the stochastic gradient (line 4). The important additional parameter is $m$, the update frequency for $\mu$. As in Allen-Zhu et al. (2016) we set this to $m = 5n$. Practically, as is standard practise we initialise RADA-vr by running AdaGrad for several epochs.

**Algorithm 5.3 RADA-vr**

**Input:** $\eta > 0, \delta \geq 0, \tau, S$ number of epochs, $m$ iterations per epoch, initial $\beta^1_0$

1: for $s = 1 \ldots S$ do  
2: $\mu = \nabla \sum^n_{i=1} f_i (\beta^s_0)$  
3: for $t = 1 \ldots m - 1$ do  
4: Compute VR gradient: $\tilde{g}_t = \nabla f_t (\beta^s_t) - \nabla f_t (\beta^s_0) + \mu$  
5: Project: $\tilde{G}_t = \Pi \tilde{g}_t$  
6: $\tilde{G}_t = \tilde{G}_{t-1} + \tilde{g}_t \tilde{g}_t^\top$  
7: $\tilde{Q}_t, \tilde{R}_t \leftarrow qr\_update (\tilde{Q}_{t-1}, \tilde{R}_{t-1}, \tilde{g}_t, \tilde{g}_t)$  
8: $\tilde{B} = \tilde{G}_t^\top \tilde{Q}$  
9: $\tilde{U}, \tilde{S}, \tilde{W} = \tilde{B} \# SVD$  
10: $\tilde{V} = WQ^\top$  
11: $\beta^s_{t+1} = \beta^s_t - \eta \tilde{V}(\tilde{S}^{1/2} + \delta I)^{-1}\tilde{V}^\top \tilde{g}_t - \eta \cdot \gamma_t$  
12: end for  
13: $\beta^{S+1}_0 = \beta^s_{t+1}$  
14: end for

**Output:** $\beta^S_m$

We study the empirical behaviour of Ada-lr, RADAGrad and its variance reduced variant in the next section.
5.4 Experiments on Low Rank Data

5.4.1 Effective low-rank data

We compare the performance of our proposed algorithms against both the diagonal and full-matrix ADAGrad variants in the idealised setting where the data is dense but has low effective rank. We generate binary classification data with \( n = 1000 \) and \( p = 125 \). The data is sampled i.i.d. from a Gaussian distribution \( \mathcal{N}(\mu, \Sigma) \) where \( \Sigma \) has with rapidly decaying eigenvalues \( \lambda_j(\Sigma) = \lambda_0 j^{-\alpha} \) with \( \alpha = 1.3, \lambda_0 = 30 \). Each of the two classes has a different mean, \( \mu_c \).

For each algorithm learning rates are tuned using cross validation. The results for 5 epochs are averaged over 5 runs with different permutations of the data set and instantiations of the random projection for ADA-LR and RADAGrad. For the random projection we use an oversampling factor so \( \Pi \in \mathbb{R}^{(10+\tau) \times p} \) to ensure accurate recovery of the top \( \tau \) singular values and then set the values of \( \lambda_{[\tau:p]} \) to zero (Halko et al. 2011).

Figure 5.1a shows the mean loss on the training set. The performance of ADA-LR and RADAGrad match that of ADA-FULL. On the other hand, ADAGrad converges to the optimum much more slowly. Figure 5.1b shows the largest eigenvalues (normalized by their sum) of the proximal matrix for each method at the end of training. The spectrum of \( G_t \) decays rapidly which is matched by the randomized approximation.

Figure 5.1: Comparison of: (a) loss and (b) the largest eigenvalues (normalised by their sum) of the proximal term on simulated data.
This illustrates the dependencies between the coordinates in the gradients and suggests $G_t$ can be well approximated by a low-dimensional matrix which considers these dependencies. On the other hand the spectrum of AdaGrad (equivalent to the diagonal of $G$) decays much more slowly.

The learning rate, $\eta$ chosen by RadaGrad and Ada-full are roughly one order of magnitude higher than for AdaGrad.

5.4.2 Non-convex optimization in neural networks

Figure 5.2: Comparison of training loss (a) and test accuracy (a) on the MNIST data set.

Figure 5.3: Comparison of training loss (a) and test accuracy (a) on the CIFAR data set.
In this section we compare RADAGrad and RADA-vr against AdaGrad and the combination of AdaGrad and SVRG on the task of optimizing CNN on the well-known MNIST, CIFAR-10 and SVHN data sets. Furthermore we provide a comparison of RADA-vr against AdaGrad+svrg on the task of optimizing a RNN for word prediction.

**Implementation.** We implemented RADAGrad within Lasagne\(^3\), a lightweight python library to build and train neural networks in the popular open-source deep learning framework Theano. The random projection is implemented as a Cython package that uses FFTW\(^4\) for the Fourier transform. We also added Theano operations for QR-update, SVD and the SRFT.

**Convolutional neural networks.** We used modified variants of standard CNN architectures for image classification.

For MNIST, CIFAR-10 and SVHN our architecture consists of three convolutional layers, each followed by a $2 \times 2$ max-pooling. At the end of the network we use a dense softmax layer with 10 output units for classification. Convolutional layers have a filter size of $5 \times 5$, generating 32 channels. All nonlinearities employed were ReLU. As objective, we minimize the categorical cross entropy.

We used a batch size of 8 and trained the networks without momentum or weight decay, in order to eliminate confounding factors. Instead,

---

\(^3\) [https://github.com/Lasagne/Lasagne](https://github.com/Lasagne/Lasagne)

\(^4\) [http://www.fftw.org](http://www.fftw.org)
we used dropout regularization \((p = 0.5)\) in the dense layers during training. Step sizes were determined by coarsely searching a log scale of possible values and evaluating performance on a validation set. We found RADAGrad to have a higher impact with convolutional layers than with dense layers, due to the higher correlations between weights. Therefore, for computational reasons, RADAGrad was only applied on the convolutional layers. The last dense classification layer was trained with AdaGrad.

In this setting Ada-full is computationally infeasible. The number of parameters in the convolutional layers is between 50-80k. Simply storing the full \(G\) matrix using double precision would require more memory than is available on top-of-the-line GPUs. We found that RADAGrad performed 5-10\times slower than AdaGrad in terms of wall clock time. This can be attributed to the lack of GPU-optimized SVD and QR routines. However these numbers are comparable with other similar recently proposed techniques (Luo et al. 2016).

The results of our experiments can be seen in Figures 5.2-5.4, where we show the objective value during training and the test accuracy. We find that both RADAGrad variants consistently outperform both AdaGrad and the combination of AdaGrad+svrg on these tasks. In particular combining RADAGrad with variance reduction results in the largest improvement for training although both RADAGrad variants quickly converge to very similar values for test accuracy.

For all models, the learning rate selected by RADAGrad is approximately an order of magnitude larger than the one selected by AdaGrad. This suggests that RADAGrad can make more aggressive steps than AdaGrad, which results in the relative success of RADAGrad over AdaGrad, especially at the beginning of the experiments.

Recurrent Neural Networks. We trained a long-short term memory (LSTM) network (Hochreiter et al. 1997) for language modelling, which consists of the following task: Given a sequence of words from an original text, predict the next word of the original text. We used pre-trained GloVe embedding vectors (Pennington et al. 2014) as input to the LSTM layer and a softmax over the vocabulary (10k words) as output. As a cost function, we chose the mean categorical cross-entropy between the output distribution and the true labels. The memory size of the LSTM units was set to 256. We trained and evaluated our network on the Penn Treebank data set (Marcus et al. 1993). We subsampled
5.4 Experiments on Low Rank Data

Figure 5.5: Comparison of training loss (a) and test loss (b) on language modeling task with the LSTM.
strings of length 20 from the data set and asked the network to predict each word in the string, given the words up to that point. Learning rates were selected by searching over a log scale of possible values and measuring performance on a validation set.

We compared RADA-vr with AdaGrad+svrg. The results of this experiment can be seen in Figure 5.5. We found that RADA-vr consistently outperforms AdaGrad+svrg in training the LSTM network. Again, we found that the selected learning rate is an order of magnitude higher for RADA-vr than for AdaGrad+svrg. The relative improvement of RADA-vr over AdaGrad+svrg in the training loss is also reflected in the test loss.

5.5 Discussion

We have presented Ada-lr and RADAGrad which approximate the full proximal term of AdaGrad using fast, structured random projections. Ada-lr enjoys similar regret to Ada-full and both methods achieve similar empirical performance at a fraction of the computational cost. Importantly, RADAGrad can easily be modified to make use of standard improvements such as variance reduction. Using variance reduction in combination in particular has stark benefits for non-convex optimization in CNNs and RNNs. We observe a marked improvement over widely-used techniques such as AdaGrad and SVRG, the combination of which has recently been proven to be an excellent choice for non-convex optimization (Allen-Zhu et al. 2016).

Furthermore, we tried to incorporate exponential forgetting schemes similar to RMSProp and Adam into the RADAGrad framework but found that these methods degraded performance. A downside of such methods is that they require additional parameters to control the rate of forgetting.

Optimization for deep networks has understandably been a very active research area. Recent work has concentrated on either improving estimates of second order information or investigating the effect of variance reduction on the gradient estimates. It is clear from our experimental results that a thorough study of the combination provides an important avenue for further investigation.
ELLIPSOIDAL MULTIPLE INSTANCE LEARNING

In many applications of supervised learning, the cost of obtaining ground truth labels is a significant bottleneck. This has led to research on weakly labeled data, among which the framework of multiple instance learning (MIL) has shown promising results. In parallel, there has been developments in robust optimization, where data uncertainty is taken into account.

In this chapter we will investigate a novel algorithm for the MIL problem: Instead observing a label for each data point in the training set, we only observe one label for a whole group of data points (instances). If the label is positive (+1) we only know that at least one of the instances has a positive label, if the label of the group is negative (−1) all the instances have negative label.

Whereas in the previous three chapters uncertainty was with respect to the data space and came either indirectly from the infeasibility to process the data set as a whole (Chapter 5), from corruptions in the observations (Chapter 3) or from not observing all the features on each compute node (Chapter 4), MIL solves a problem where the uncertainty lies in the label space.

6.1 INTRODUCTION

We will describe a large margin method for asymmetric learning with ellipsoids, called ellipsoidal multiple instance learning (eMIL), suited to multiple instance learning. We derive the distance between ellipsoids and the hyperplane, generalising the standard support vector machine (Section 2.2). Negative bags in MIL contain only negative instances, and we treat them akin to uncertain observations in the robust optimization framework (Section 6.2.1). However, our method allows positive bags to cross the margin, since it is not known which instances within are positive. We relax the function over the set of instance labels and approximate a bag by the first and second moment of the empirical

1 Parts of this chapter appear in Krummenacher et al. (2013)
distribution. I.e., we take the arithmetic mean and empirical covariance matrix of the within bag instances. Neglecting higher order moments gets rid of the combinatorial optimization problem and enforces regularisation.

Our proposed method (eMIL) results in a modular two stage algorithm: (1) estimate the ellipsoids, and (2) optimize the generalised large margin algorithm. An additional benefit is that our approach gives an instance level classifier (instead of a bag level classifier), which may be important in some applications. We first derive the optimization problem to find a maximum-margin type classifier (see Section 2.2 for ellipsoids, where one class of ellipsoids can overlap with the decision boundary (Section 6.3). Two different ways of scaling the empirical covariance matrix for different distributional assumptions on the bags are presented in Section 6.3.3 and Section 6.3.5. We show in Section 6.3.5 that solving this optimization problem is equivalent to treating each bag as a random variable and robustly maximizing the margin between instances distributed according to this random variable under asymmetric probabilistic constraints over all distributions with finite mean and covariance. To solve the resulting non-convex optimization problem a quasi Newton method and a decomposition of the objective into the difference of two convex functions is presented in Section 6.4. The method compares favourably to state of the art MIL methods with respect to accuracy on benchmark data sets (Section 6.5). Finally we briefly introduce our motivating application: a safety critical real world problem of detecting wheel defects, and show that eMIL has better accuracy than recent methods (Section 6.6). This data set is described in more detail in Chapter 7.

6.2 Background

In this section we briefly review multiple instance learning and robust classification. For an introduction to support vector machines, kernels and the mean map we will refer to the respective sections in Chapter 2.

6.2.1 Robust Classification

In robust classification, the goal is to find a classifier which is robust to random perturbations in the feature space (Ben-Tal et al. 2009). If we
assume an ellipsoidal uncertainty set \( U_i = \{ \mathbf{u}_i : \mathbf{u}_i^\top \mathbf{P}_i \mathbf{u}_i \leq 1 \} \) and seek to find a maximum-margin classifier, we get the formulation of a robust SVM (Sra et al. 2011):

\[
\begin{align*}
\min_{\beta, b, \xi} & \quad \frac{1}{2} \| \beta \|^2 + C \cdot \sum_{i=1}^{N} \xi_i \\
\text{s.t.} & \quad y_i (\beta^\top \mathbf{q}_i + b) \geq 1 - \xi_i + \| \mathbf{P}_i^{1/2} \beta \| \\
& \quad 0 \leq \xi_i \quad \forall i.
\end{align*}
\]

(6.1)

The idea of ellipsoidal uncertainty and the resulting optimization problem are similar to our proposed ellipsoidal multiple instance learning algorithm. See Section 6.3.6 for important differences in the optimization problem and what that means for the resulting classifier.

6.2.2 Multiple Instance Learning

The multiple instance learning setting was introduced by Dietterich et al. (1997). The authors formulate the problem as having a set of feature vectors per object and it is unknown which vector is responsible for the classification of the object. They introduce the now canonical classification rule for MIL (Definition 6.1), where the label \( y_i \) of each group is based on the labels \( y_{ij} \) of its instances and given by the following rule.

**Definition 6.1.** MIL rule

\[
y_i = \begin{cases} 1, & \text{if } \exists j: y_{ij} = 1 \\ -1, & \text{else.} \end{cases}
\]

Here and in the following the index \( i \) refers to the \( i \)th bag and the indices \( ij \) to the \( j \)th instance in the \( i \)th bag. In contrast to the standard binary supervised classification setting, the labels \( y_{ij} \) of the training examples (the instances) are not observed. The examples are grouped into bags however and we observe a label \( y_i \) for each bag. According to the MIL labeling rule in Definition 6.1 there are many different instance label assignments possible that are consistent with the bag labels.

We will use the following notation for bags and instances. The data set consists of \( B \) bags of examples, where each bag

\[
\left\{ \left\{ (\mathbf{x}_{i1}, \ldots, \mathbf{x}_{ij}, \ldots, \mathbf{x}_{in_i}, y_i) \right\} \right\}_{i=1}^{B}
\]

consists of \( n_i \) instances and only one label.
The MIL setting is very natural in many applications such as text classification (Andrews et al. 2002), image retrieval (Gehler et al. 2007) and object detection (Viola et al. 2006). For example, content based image retrieval represents an image as a bag containing image patches (examples $x_{ij}$) and for a particular query, one is interested in returning images (bags $\{x_{i1}, \ldots, x_{ijn_i}\}$) that contain the object, instead of solving the more complex problem of labeling every patch in the image.

Unfortunately, due to the weak labeling, it is unclear during training time of MIL methods how to allocate the positive label. Since any number of examples in a positive bag may be positive, one would naively have to look at all possible labelings that includes at least one positive label. For learning a classifier, the bag label is traditionally inferred as the max over the classification of all instances in the respective bag: $y_i = \text{sgn} \max_{j=1}^{n_i} (\langle \beta, x_{ij} \rangle + b)$ (Andrews et al. 2002). This results in a non-convex optimization problem for finding a maximum-margin hyperplane for classification due to the negative max function not being convex. It also results in potentially expensive computations which involve optimising a combinatorial problem. Recently, there has been several proposals of making some assumptions about the structure of the bag such as using Markov random fields (Warrell et al. 2011) and low dimensional manifolds (Babenko et al. 2011).

In Section 6.5, we compare our method to the following algorithms for MIL: Two traditional approaches to solve the MIL problem, the earliest one being the method of axis-parallel rectangles (APR) (Dietterich et al. 1997), that was specifically designed for the MUSK1 and MUSK2 data sets and an extension of the diverse-density algorithm (Maron et al. 1998), EMDD, using Expectation-Maximization to find a positive witness (Q. Zhang et al. 2002). We also compare to two extensions of a support vector machine microscopic Multiple-Instance SVM (mi-SVM) (maximizing instance margin) and macroscopic Multiple-Instance SVM (MI-SVM) (maximizing bag margin), that lead to mixed-integer programs (Andrews et al. 2002); deterministic annealing methods to solve mi-SVM (AL-SVM) and MI-SVM (AW-SVM) (Gehler et al. 2007); a convex semi definite programming to the maximum instance margin problem (SDP) (Guo 2009); MICA, an algorithm that uses convex combinations of positive bag instances (Mangasarian et al. 2008); and a more recent approach developing a Gaussian process by building bag likelihood models from the GP latent variables (GPMIL) (Kim et al. 2010).
6.3 Detection with Ellipsoids

We derive a maximum-margin type classifier for the problem of learning with positive and negative ellipsoidal examples. Our aim is to exploit the structure of a bag in the MIL setting and not just treat the instances as individual separate points. We capture this bag structure by the empirical mean and the empirical covariance matrix of all the instances in a bag. This naturally leads to the interpretation of a bag as an ellipsoid. The notion that a positive bag label only guarantees one instance to be positive, is represented by letting ellipsoids with positive label overlap the negative half space. Negatively labeled ellipsoids on the other hand are required to be maximally distant from the decision surface, since we know that all instances in a bag with negative label are indeed negative.

Recall that an ellipsoid in $\mathbf{x} \in \mathbb{R}^d$ is given by a positive semidefinite covariance matrix $\mathbf{P} \in S^+_d$, and a central vector $\mathbf{q} \in \mathbb{R}^d$ by

$$
(x - q)^\top P^{-1}(x - q) = 1.
$$

(6.2)

Given a set of examples and corresponding labels $\{(\mathbf{P}_i, \mathbf{q}_i), y_i\}_{i=1}^B$, we would like to find a linear separating hyperplane with $\beta \in \mathbb{R}^d$ and $b \in \mathbb{R}$

$$
\beta^\top \mathbf{x} + b = 0
$$

(6.3)

which follows the maximum margin principle. Therefore for the predictor based on ellipsoid $i$ given by $f(\mathbf{P}_i; \mathbf{q}_i)$, we would like to solve the following regularized ERM

$$
\min_{\beta, b} \sum_{i=1}^B \ell(y_i, f(\mathbf{P}_i; \mathbf{q}_i)) + \frac{\lambda}{2} \|\beta\|^2
$$

(6.4)

where $\lambda$ is the regularisation parameter and $\ell(y_i, p)$ is the hinge loss (See Equation 2.2).

6.3.1 Optimization problem

The prediction function $f(\mathbf{P}_i, \mathbf{q}_i)$ should give the signed distance of the ellipsoid to the hyperplane. By reasoning about the geometry of the problem (refer to Figure 6.1), we get the following distance for any ellipsoid to the hyperplane.
Proposition 6.2. Given an ellipsoid, Equation (6.2), and a hyperplane, Equation (6.3), and taking the asymmetry of positive and negative ellipsoids into account, the signed distance from the ellipsoid to the hyperplane is given by

\[
\frac{1}{\|\beta\|} \left( \sqrt{\beta^\top P \beta + \beta^\top q + b} \right).
\]  

(6.5)
**6.3 Detection with Ellipsoids**

*Proof.* We would like to minimize the squared distance between a point \( x \) on the hyperplane, and a point \( z \) on the ellipsoid. This can be expressed as the following constrained optimization problem:

\[
\begin{align*}
\min_{x,z} \quad & \|z - x\|^2 \\
\text{s.t.} \quad & (z - q)^\top P^{-1}(z - q) = 1 \\
& \beta^\top x + b = 0
\end{align*}
\]

We form the Lagrangian, using multiplier \( \eta \) for the ellipsoidal constraint and \( \gamma \) for the hyperplane.

\[
\mathcal{L}(x, z, \eta, \gamma) = \|z - x\|^2 + \eta (z - q)^\top P^{-1}(z - q) - \eta + \gamma \beta^\top x + \gamma b \quad (6.6)
\]

Taking the gradient of Equation (6.6) with respect to \( x \) and \( z \) respectively, and setting it to zero gives

\[
\begin{align*}
2(z - x) &= \gamma \beta \quad (6.7) \\
2(z - x) + 2\eta P^{-1}(z - q) &= 0 \quad (6.8)
\end{align*}
\]

By substituting Equation (6.7) into Equation (6.8), we obtain that

\[
z = -\frac{\gamma}{2\eta} P\beta + q \quad (6.9)
\]

and using this in Equation (6.7) gives

\[
x = -\frac{\gamma}{2\eta} P\beta + q - \frac{\gamma}{2} \beta \quad (6.10)
\]

By substituting Equation (6.9) and (6.10) into the Lagrangian (Equation (6.6)) we obtain an expression only in the dual variables.

\[
\mathcal{L}(\eta, \gamma) = -\frac{\gamma^2}{4} \|\beta\|^2 + \eta \left( \frac{\gamma}{2\eta} P\beta \right)^\top P^{-1} \left( \frac{\gamma}{2\eta} P\beta \right) \\
- \eta + \gamma \beta^\top \left( -\frac{\gamma}{2\eta} P\beta + q - \frac{\gamma}{2} \beta \right) + \gamma b \\
= -\frac{\gamma^2}{4} \|\beta\|^2 - \frac{\gamma^2}{4\eta} \beta^\top P\beta - \eta + \gamma \beta^\top q + \gamma b
\]

We would like to maximize the dual with respect to \( \eta \) and \( \gamma \), and this point is achieved at the stationary points

\[
\frac{\partial \mathcal{L}}{\partial \gamma} = -\frac{\gamma}{2} \|\beta\|^2 - \frac{\gamma}{2\eta} \beta^\top P\beta + \beta^\top q + b = 0 \quad (6.11)
\]
ellipsoidal multiple instance learning

and

$$\frac{\partial L}{\partial \eta} = \frac{\gamma^2}{4\eta^2} \beta^T P \beta - 1 = 0 \tag{6.12}$$

Equation (6.12) implies

$$\eta = \pm \frac{\gamma}{2} \sqrt{\beta^T P \beta} \tag{6.13}$$

Substituting the expression for \(\eta\) (Equation (6.13)) into the stationary condition for \(\gamma\) (Equation (6.11)) gives

$$-\frac{\gamma}{2} \|\beta\|^2 \pm \sqrt{\beta^T P \beta + \beta^T q + b} = 0 \tag{6.14}$$

Observe from Equation (6.7) that the distance from the ellipsoid to the hyperplane is given by \(\frac{\gamma}{2} \|\beta\|\) which from Equation (6.14) is given by

$$\frac{\gamma}{2} \|\beta\| = \frac{1}{\|\beta\|} \left( \pm \sqrt{\beta^T P \beta + \beta^T q + b} \right)$$

When the ellipsoid intersects the hyperplane, we would like the point on the ellipsoid furthest away from the hyperplane, which is given by the solution of the following constrained optimization problem.

$$\max_{x,z} \|z - x\|^2 \quad \text{s.t.} \quad (z - q)^T P^{-1} (z - q) = 1 \quad \beta^T x + b = 0$$

Since the only difference is from finding the minimum to finding the maximum, the above derivation remains identical and the proposition follows.

Therefore the prediction function is given by

$$f(P; q) = \sqrt{\beta^T P \beta + \beta^T q + b}. \tag{6.15}$$

Substituting Equation (6.15) into Equation (6.4) we obtain the following optimization problem, which we call ellipsoidal multiple instance learning (eMIL).

$$\min_{\beta, b} \sum_{i=1}^B \ell \left( y_i \left( \sqrt{\beta^T P_i \beta + \beta^T q_i + b} \right) \right) + \frac{\lambda}{2} \|\beta\|^2. \tag{6.16}$$
6.3 DETECTION WITH ELLIPSOIDS

Equation (6.16) is subtly different from robust optimization problems. We discuss this further in Section 6.3.6. Note that the optimization problem given by Equation (6.16) is non-convex. This is due to the term $-\sqrt{\beta^\top P_i \beta}$ in the hinge loss for positive bags

$$\max \left( 0, 1 - \left( \sqrt{\beta^\top P_i \beta} + \beta^\top q_i + b \right) \right),$$

which is a concave function in $\beta$. It can also be observed that the problem is not a second order cone program by decomposing the hinge loss:

$$\min_{\beta, b} \frac{\lambda}{2} \| \beta \|^2 + \sum_{i=1}^{B} \xi_i$$

s.t. $\| A_i \beta \| \geq -\xi_i - \beta^\top q_i - b + 1, \forall i: y_i = +1$ (6.17)

$\| A_i \beta \| \leq \xi_i - \beta^\top q_i - b - 1, \forall i: y_i = -1$

$0 \leq \xi.$

Where we have used $P_i = A_i^\top A_i$. The constraints for positive ellipsoids are not second order cone constraints.

6.3.2 Ellipsoid estimation

We model the $i$th bag, $\{x_{i1}, \ldots, x_{in_i}\}$ with $n_i$ instances, by the empirical mean and covariance of the instances, given by

$$q_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}, \quad P_i = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (x_{ij} - q_{ij})(x_{ij} - q_{ij})^\top.$$ (6.18)

When the number of instances per bag $n_i$ is larger than the dimensionality of the feature space $d$, the covariance $P_i$ is of full rank and strictly positive definite. However, in many data sets $n_i < d$, resulting in a low rank $P_i$. This is usually the case in the test data sets we consider. The average number of instances in a bag is much lower than $d$, resulting in a semidefinite covariance $P_i$.

The covariance matrix gives the shape of the ellipsoid. To find the volume, we derive two types of scaling factors for the covariance matrix under two different distributional assumptions in the following sections.
6.3.3 Confidence regions

Under the assumption of approximately Gaussian distributed instances per bag we can use the following fact. Recall that for a random variable \( x \) distributed as a \( p \)-dimensional Gaussian \( \mathcal{N}_p(\mu, \Sigma) \), the quadratic form \((x - \mu)^\top \Sigma^{-1} (x - \mu)\) is distributed as \( \chi^2 \) with \( p \) degrees of freedom. This implies that the ellipsoid

\[(x - \mu)^\top \Sigma^{-1} (x - \mu) \leq \chi^2_p(\alpha)\]

contains \( 1 - \alpha \) of the total probability mass.

For an ellipsoid \((q, P)\) to cover \( 1 - \alpha \)-percent of a \( p \)-dimensional multivariate Gaussian distribution with a covariance matrix \( \Sigma \) we set \( P = \Sigma \cdot \frac{1}{\chi^2_p(1 - \alpha)} F^{-1}_{\chi^2_p(q)} \), where \( F^{-1}_{\chi^2_p(q)} \) is the quantile function (inverse cdf) of \( \chi^2_p \) at quantile \( q \). We can use this fact to scale the empirical covariance matrix, estimated from the bag instances.

In the next section we describe a method to test for the assumption of Gaussian distributed instances by providing a kernel test for the Normality of a data set.

Then in Section 6.3.5 a more general scaling factor for non Gaussian distributions is derived.

6.3.4 A Kernel Test for Normality

The maximum mean discrepancy (MMD) (Definition 2.11) discussed in Section 6.3.4 allows us to derive a hypotheses test for the normality of a sample. To test how likely it is that a data set is a sample from a multivariate normal distribution (MVN) we derive a test statistics based on the MMD. The test statistic is given by the distance between the empirical distribution of the sample and a multivariate normal distribution in feature space.

For a data set \( X = \{ x_1, \ldots, x_n \} \) with empirical covariance \( \Sigma \) and mean \( \mu \) the test statistics is

\[
\| \hat{m}_X - m_y \|_{\mathcal{H}}^2 = \| \sum_{i=1}^n \phi(x_i) - \mathbb{E}_y[\phi(y)] \|_{\mathcal{H}}^2,
\]

(6.19)

where \( y \sim \mathcal{N}(\mu, \Sigma) \), \( m \) refers to the mean map (Definition 2.9), \( \hat{m} \) to the empirical mean map (Definition 2.10) and \( \phi(\cdot) \) is a feature map. See Section 6.3.4 for detail of these concepts.
If we use a Gaussian kernel $\langle \phi(x), \phi(y) \rangle = k_\sigma(x, y) = \exp(-\sigma \|x - y\|^2)$ for the mean map, the expectation over $y$ can be evaluated analytically and we arrive at the expression in Theorem 6.3.

**Theorem 6.3** (Normality test statistics).

$$ Tb(X, \sigma) = |4\sigma \Sigma + I|^{(-1/2)} + \frac{1}{n^2} \sum_x \sum_{x'} k_\sigma(x, x') $$

$$ - \frac{2}{n} \sum_x |2\sigma \Sigma + I|^{-1/2} \cdot \exp\left(-\frac{1}{2} (x - \mu)^\top (\Sigma + \frac{1}{2\sigma} I)^{-1} (x - \mu)\right), $$

where $|\cdot|$ is the determinant.

**Proof.** To derive the expectations analytically we use the following Equation from Ahrendt (2005, Equation 5.2 and 5.6).

$$ \int_{\mathbb{R}^d} N(\mu_a, \Sigma_a) \cdot N(\mu_b, \Sigma_b) dx = \int_{\mathbb{R}^d} z_c N(\mu_a, \Sigma_a) dx $$

$$ = z_c $$

(6.21)

$$ z_c = |2\pi(\Sigma_a + \Sigma_b)|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (\mu_a - \mu_b)^\top (\Sigma_a + \Sigma_b)^{-1} (\mu_a - \mu_b)\right) $$

(6.22)

To be able to use this fact, we rewrite the Gaussian kernel function as the pdf of a MVN. After setting $A = \frac{1}{2\sigma} I$

$$ k(x, y) = e^{-\sigma \|x - y\|^2} $$

(6.23)

$$ = e^{-\sigma (x - y)^\top (x - y)} $$

(6.24)

$$ = e^{-\frac{1}{2} (x - y)^\top A^{-1} (x - y)} $$

(6.25)

$$ = \sqrt{2\pi A} \cdot N_x(y, A) $$

(6.26)

Now computing the expectation over $x$ is straightforward:

$$ \mathbb{E}_x[k(x, y)] = \sqrt{2\pi A} \int_{\mathbb{R}^d} N_x(y, A) N_x(\mu, \Sigma) dx $$

(6.27)

$$ = \sqrt{2\pi A} \cdot |2\pi(A + \Sigma)|^{-\frac{1}{2}} $$

$$ \cdot \exp\left(-\frac{1}{2} (y - \mu)^\top (A + \Sigma)^{-1} (y - \mu)\right) $$

(6.28)

$$ = |2\sigma \Sigma + I|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (y - \mu)^\top (A + \Sigma)^{-1} (y - \mu)\right) $$

(6.29)
To compute the expectation over $y$, we note that:

$$
\mathbb{E}_x[k(x, y)] = \sqrt{2\pi A} \cdot \mathcal{N}_y(\mu, A + \Sigma) \quad (6.31)
$$

Now

$$
\mathbb{E}_y[\mathbb{E}_x[k(x, y)]] = \sqrt{2\pi A} \int \mathcal{N}_y(\mu, A + \Sigma) d\mathcal{N}_y(\mu, \Sigma) dx
$$

$$
= \sqrt{2\pi A} \cdot |2\pi(A + 2\Sigma)|^{-\frac{1}{2}} \exp \left( -\frac{1}{2}(\mu - \mu)^\top(A + 2\Sigma)^{-1}(\mu - \mu) \right) \quad (6.34)
$$

$$
= |4\sigma\Sigma + I|^{-\frac{1}{2}} \quad (6.35)
$$

We are now ready to proof the statement of the theorem

$$
T_b(X, \sigma) = \|\hat{m}_X - m_y\|_{\mathcal{H}}^2 \quad (6.36)
$$

$$
= \| \sum_{i=1}^n \phi(x_i) - \mathbb{E}_y[\phi(y)] \|_{\mathcal{H}}^2 \quad (6.37)
$$

$$
= \mathbb{E}_{y' \mid y}[\mathbb{E}_y[k_{\sigma}(y, y')]] + \frac{1}{n^2} \sum_x \sum_{x'} k_{\sigma}(x, x') - \frac{2}{n} \sum_x \mathbb{E}_y[k_{\sigma}(x, y)] \quad (6.38)
$$

Using Equation (6.30) and setting $y = x, x = y$ and Equation (6.35) with $y = y', x = y$ we get the statement in Equation (6.3).

The normality test statistics $T_b(X, \sigma)$ (Definition 6.3) can now be used in a statistical hypothesis test. We first derive a uniform convergence bound on $T_b(X, \sigma)$ and then use this bound to derive a hypothesis test.

**HYPOTHESIS TEST BASED ON BOUND ON TB**

We can directly use Theorem 2.13 to bound $T_b(X, \sigma)$ by setting $B_1 = \sqrt{2/n} \cdot \sqrt{1 - |4\sigma\Sigma + I|^{-\frac{1}{2}}}$ and arrive at

$$
T_b(X, \sigma) \leq \sqrt{2/n} \cdot \sqrt{1 - |4\sigma\Sigma + I|^{-\frac{1}{2}}} + \varepsilon
$$

with probability at least $1 - \exp(-\frac{\varepsilon^2 n}{4})$. 108
Since we can compute $E_{x,x'}[k(x, x) - k(x, x')]$ in Theorem 2.13 analytically we can use the tighter bound based on $B_1$ unlike Gretton et al. (2012).

Using the above bound we can write

$$P[Tb \leq B_1 + \epsilon] \geq 1 - e^{-\frac{\epsilon^2 n}{4}}$$  \hspace{1cm} (6.39)

$$P[Tb \geq B_1 + \epsilon] < 1 - e^{-\frac{\epsilon^2 n}{4}}$$  \hspace{1cm} (6.40)

Now setting $\alpha = 1 - e^{-\frac{\epsilon^2 n}{4}}$, the Type I error rate, we can compute the acceptance region of the test:

$$Tb < \sqrt{\frac{2}{n}} \cdot \sqrt{1 - |4\sigma \Sigma + I|^{-\frac{1}{2}}} + \sqrt{\frac{4 \log(\frac{1}{1-\alpha})}{n}}$$  \hspace{1cm} (6.41)

### 6.3.5 Probabilistic multiple instance learning

In this section we show that \texttt{eMIL} maximizes the margin between instances from any within bag data distribution with finite mean and covariance with high probability, while enforcing the asymmetry inherent to multiple instance learning. We use similar minimax techniques to robust optimization approaches (Lanckriet et al. 2002; Shivaswamy et al. 2006), that are based on a multivariate Chebyshev’s inequality (Bertsimas et al. 2001).

Assuming the instances in a bag are drawn from a probability distribution with mean $q_i$ and covariance $\Sigma_i$, we want to find a hyperplane that maximises the margin between instances from the two classes. Remember, that for instances in a negative bag we know the label, for instances in a positive bag we only know at least one instance is positive. We formalize this with the following optimization problem:

$$\min_{\beta,b} \frac{\lambda}{2} \|\beta\|^2 + \sum_{i=1}^{B} \xi_i$$

$$\text{s.t.} \inf_{x_i \sim (q, \Sigma)} \Pr[y_i(\langle \beta, x_i \rangle + b) \geq 1 - \xi_i] \geq 1 - \alpha_i, \forall y_-$$

$$\sup_{x_i \sim (q, \Sigma)} \Pr[y_i(\langle \beta, x_i \rangle + b) \leq 1 - \xi_i] \geq \alpha_i, \forall y_+$$

$$0 \leq \xi_i,$$  \hspace{1cm} (6.42)
where $\sup / \inf_{x_i \sim (q_i, \Sigma_i)}$ means supremum/infimum over all distributions for $x_i$ having mean $q_i$ and covariance $\Sigma_i$. The constraints in Equation (6.42) differ for instances from positive bags and instances for negative bags. For instances from negative bags we want the smallest (over all distributions with mean $q_i$ and covariance $\Sigma_i$) probability of correct classification to be higher than $\alpha_i \in (0, 1]$. For small $\alpha_i$ this gives high worst-case probability of correct classification for all instances in a negative bag. For instances from positive bags the highest (again over all distributions with mean $q_i$ and covariance $\Sigma_i$) probability of negative classification needs to be larger than $\alpha_i$, in other words, there exists a distribution with negative classification higher than $\alpha_i$. This may seem counter-intuitive at first, but recall that for a positive bag to be classified correctly only one instance needs to be classified correctly, i.e. many of the instances will be classified negative. Here for small $\alpha_i$ this gives low negative classification probability.

We show that considering the worst case distribution with finite mean and covariance results in the same constraints as making an assumption of ellipsoidal bags.

**Proposition 6.4.** The optimization problem in Equation (6.42) is equivalent to eMIL (Equation (6.17)).

To prove Proposition 6.4 we use the following Lemmas:

**Lemma 6.1.**

$$
\sup_{x_i \sim (q_i, \Sigma_i)} \Pr[y_i (\langle \beta, x_i \rangle + b) \leq 1 - \xi_i] = \frac{1}{1 + d^2}, \text{with} \quad (6.43)
$$

$$
d^2 = \inf_{x_i | y_i (\langle \beta, x_i \rangle + b) \leq 1 - \xi_i} (x_i - q_i)^\top \Sigma_i^{-1} (x_i - q_i). \quad (6.44)
$$

Where $x_i$ is a random vector and the supremum is over all distributions for $x_i$ with mean $q_i$ and covariance matrix $\Sigma_i$.

**Proof.** We make use of the following multivariate Chebyshev’s inequality to proof Lemma 6.1

**Theorem 6.5.** (Marshall et al. 1960), (Bertsimas et al. 2001), (Lanckriet et al. 2002)

$$
\sup_{y \sim (\bar{y}, \Sigma_y)} \Pr[y \in S] = \frac{1}{1 + d^2}, \text{with} \quad (6.45)
$$

$$
d^2 = \inf_{y \in S} (y - \bar{y})^\top \Sigma_y^{-1} (y - \bar{y})
$$
Where \( y \) is a random vector, the supremum is over all distributions for \( y \) with mean \( \bar{y} \) and covariance matrix \( \Sigma_y \) and \( S \) is a given convex set.

Now, setting \( S = \{ x_i | y_i(\langle \beta, x_i \rangle + b) \leq 1 - \zeta_i \} \) we get the claimed equality.

**Lemma 6.2.** For \( x_i \sim (q_i, \Sigma_i) \):

\[
\inf_{x_i \mid y_i(\langle \beta, x_i \rangle + b) \leq 1 - \zeta_i} \frac{(x_i - q_i) \top \Sigma_i^{-1} (x_i - q_i)}{\beta \top \Sigma_i \beta} = \frac{\max(0, y_i(\langle \beta, q_i \rangle + b) - 1 + \zeta_i)^2}{\beta \top \Sigma_i \beta}.
\]

**Proof.** We follow the proof in (Lanckriet et al. 2002) to find a closed form expression for \( \inf_{x_i \mid y_i(\langle \beta, x_i \rangle + b) \leq 1 - \zeta_i} \frac{(x_i - q_i) \top \Sigma_i^{-1} (x_i - q_i)}{\beta \top \Sigma_i \beta} \).

If \( y_i(\langle \beta, q_i \rangle + b) \leq 1 - \zeta_i \) then we can just set \( x_i = q_i \) and the infimum becomes 0.

To show the other case of \( y_i(\langle \beta, q_i \rangle + b) \geq 1 - \zeta_i \) we write \( d^2 = \inf_{(c,k) \geq f} \langle k, k \rangle \), where \( k = \Sigma_i^{-1/2} (x_i - q_i) \), \( c \top = -y_i \beta \top \Sigma_i^{1/2} \) and \( f = y_i(\langle \beta, q_i \rangle + b) - 1 + \zeta_i \geq 0 \). We form the Lagrangian:

\[
L(k, \lambda) = \langle k, k \rangle + \lambda(f - \langle c, k \rangle)
\]

and maximize it with respect to the dual variable \( \lambda \geq 0 \) and minimize with respect to the primal variable \( k \). At the optimum we get \( 2k = \lambda c \) and \( f = \langle c, k \rangle \). So, \( \lambda = \frac{2f}{\langle c, c \rangle} \) such that indeed \( \lambda \geq 0 \) because \( f > 0 \).

Also, \( k = \frac{fc}{\langle c, c \rangle} \). This yields

\[
\frac{(y_i(\langle \beta, q_i \rangle + b) - 1 + \zeta_i)^2}{\beta \top \Sigma_i \beta}
\]

Combining both cases \( y_i(\langle \beta, q_i \rangle + b) \leq 1 - \zeta_i \) and \( y_i(\langle \beta, q_i \rangle + b) \geq 1 - \zeta_i \) we get the right hand side of Lemma 6.2:

\[
\frac{\max(0, y_i(\langle \beta, q_i \rangle + b) - 1 + \zeta_i)^2}{\beta \top \Sigma_i \beta}
\]

**Proof of Proposition 6.4.** Since the objective function in both optimization problems are the same, it is sufficient to show that the constraints are equivalent.
Ellipsoidal Multiple Instance Learning

For the probabilistic constraint of negative bags

$$\inf_{x_i \sim (q_i, \Sigma_i)} \Pr[y_i(\langle \beta, x_i \rangle + b) \geq 1 - \xi_i] \geq \alpha_i,$$

we rewrite it as

$$\sup_{x_i \sim (q_i, \Sigma_i)} \Pr[y_i(\langle \beta, x_i \rangle + b) \leq 1 - \xi_i] \leq 1 - \alpha_i. \quad (6.47)$$

Then we can use Lemma 6.1 to rewrite the constraints as

$$\frac{1}{1 + d^2} \leq \alpha_i, \forall i: y_i = -1 \quad (6.48)$$

$$\frac{1}{1 + d^2} \geq \alpha_i, \forall i: y_i = +1,$$

where $d^2$ is defined as in Lemma (6.1).

Next we use Lemma 6.2 and rearrange the terms to finally get the constraints

$$-\langle \beta, q_i \rangle - b \geq 1 - \xi_i + \kappa(\alpha_i) \sqrt{\beta^T \Sigma_i \beta}, \forall y_- \quad (6.49)$$

$$\langle \beta, q_i \rangle + b \geq 1 - \xi_i - \kappa(\alpha_i) \sqrt{\beta^T \Sigma_i \beta}, \forall y_+,$$

where $\kappa(\alpha_i) = \sqrt{\frac{1 - \xi_i}{\alpha_i}}$.

Now, if $A_i$ in Equation (6.17) is set to $A_i = \kappa(\alpha_i) \Sigma_i^{1/2}$ the equivalence can be seen.

This shows that by considering the worst case distribution (not necessarily Gaussian) with finite mean and covariance scales the ellipsoid by a factor $\kappa(\alpha_i)^2$.

6.3.6 Relation to robust classification

As explained in Section 6.2.1 the robust SVM with ellipsoidal uncertainty set (Equation 6.1) has a similar looking optimization problem. The difference to eMIL (Equation (6.17)) is the fact that $\|P_i^{1/2} \beta\|$ is not multiplied with the label $y_i$. This leads to a hyperplane that separates ellipsoids, whereas in eMIL the positive ellipsoids can overlap the hyperplane. See Figure 6.2b for an illustration.
6.4 solving emIL

6.4.1 Difference of Convex Functions

We propose two approaches to optimize the resulting non-convex optimization problem (6.16). We derive a Concave-Convex Procedure (CCCP) in the following subsection, and a quasi-Newton approach (L-BFGS) in Section 6.4.3. While CCCP gives consistently lower optimal values on all the data sets that we tried, the gradient based method is usually much faster, especially in very high dimensional problems. However, the lower objective value typically also does not translate into significant improvements on test accuracy. For both approaches, we initialize by setting

$$
\beta_0, b_0 = \arg \min_{\beta,b} \sum_{i=1}^{B} \max(0, 1 - y_i (\beta^\top q_i + b)) + \frac{\lambda}{2} \|\beta\|^2,
$$

(6.50)

which is the maximum-margin hyperplane that separates the means of the bags. This can be seen as a first order approximation of the within-bag distribution.
**Algorithm 6.1 eMIL: Sequential SOCP**

Initialise $(\beta_0, b_0)$ according to Equation (6.50)

while $\ell(\beta_k, b_k) - \ell(\beta_{k+1}, b_{k+1}) > \epsilon$ do

Find the optimal solution $(\beta_{k+1}, b_{k+1})$ of Equation (6.56), given $(\beta_k, b_k)$.

end while

---

6.4.2 **Solving eMIL with CCCP**

We can express the objective function (6.16) as a difference of convex functions and use CCCP to solve it, by solving a series of convex programs. See Yuille et al. (2003) for the introduction of the CCCP, Sriperumbudur et al. (2009) for its convergence proof, and Le Thi et al. (2005) for an overview on difference of convex functions algorithm. The solution of eMIL with CCCP is shown in Algorithm 6.1.

The decomposition of (6.16) into the difference of two convex functions $g(\beta, b), h(\beta, b)$ is as follows:

$$
\min_{\beta, b} \sum_y \max(0, 1 + f(P_i; q_i)) \\
+ \sum_{y_+} \max(0, -1 + f(P_i; q_i)) + \frac{\lambda}{2} \|b\|^2 \\
- \sum_{y_+} (-1 + f(P_i; q_i)) ,
$$

(6.51) (6.52)

where the first three lines (Equation (6.51)) correspond to $g(\beta, b)$ and the last line (Equation (6.52)) to $-h(\beta, b)$. Given the decomposition, CCCP proceeds by linearizing the concave part $-h(x, b)$ at $\beta_k, b_k$, solving the resulting convex optimization problem Equation (6.53), obtaining the optimal value $\beta_{k+1}, b_{k+1}$ and repeating until convergence. The linearisation of $-h(x, b)$ (Equation (6.52)) at $\beta_k, b_k$ is given by: $-\langle \beta, \partial h(\beta_k, b_k) \rangle$. 
By taking the sum over the positive examples out of the inner product we arrive at:

\[
\min_{\beta, b} \sum_{y} \max (0, 1 + f(P_i; q_i)) \\
+ \sum_{y} \max (0, -1 + f(P_i; q_i)) + \frac{\lambda}{2} \|\beta\|^2 \\
- \sum_{y} \left( \langle \beta, \frac{P_i\beta}{\sqrt{P_i^T P_i} \beta} + q_i \rangle + b \right).
\]

(6.53)

By introducing slack variables \(\xi\), using \(P_i = A_i^T A_i\) and finally converting the remaining objective function into second order cone constraint, we can now rewrite Equation (6.53) to get the equivalent (in terms of optimal solution \((\beta, b)\)) constrained optimization problem Equation (6.56), which is a Second order cone program (SOCP).

**Convex sub-problem as SOCP**

First we write \(\sqrt{\beta^T P_i \beta}\) as \(\|A_i \beta\|\) with \(P_i = A_i^T A_i\).

Then we replace the hinge-loss type part of the objective function in Equation (6.53) with the following constraints, by introducing slack variables \(\xi_i\):

\[
\min_{\beta, b, \xi} \frac{\lambda}{2} \|\beta\|^2 - \sum_{y} \left( \langle \beta, \frac{P_i\beta}{\sqrt{P_i^T P_i} \beta} + q_i \rangle + b \right) + \sum_{i=1}^B \xi_i \\
\text{s.t.} \quad \|A_i \beta\| + \beta^T q_i + b \leq \xi_i - 1, \quad \forall i: y_i = -1 \\
\|A_i \beta\| + \beta^T q_i + b \leq \xi_i + 1, \quad \forall i: y_i = +1 \\
0 \leq \xi
\]

(6.54)

Where \(\sum_{y_i}\) means sum over all \(i\) for which \(y_i = +1\).
Next replace the remaining objective function with $θ$ and add it as a constraint:

$$\min_{\theta, \beta, b, \xi} \theta$$

subject to

$$\frac{\lambda}{2} \|\beta\|^2 - \sum_{y_+} \left( \left\langle \beta, \frac{P_i \beta_k}{\beta_k \top P_i \beta_k} + q_i \right\rangle + b \right) + \sum_{i=1}^{B} \xi_i \leq \theta$$

(6.55)

$$\|A_i \beta\| + \beta^\top q_i + b \leq \xi_i - 1, \quad \forall i: y_i = -1$$

$$\|A_i \beta\| + \beta^\top q_i + b \leq \xi_i + 1, \quad \forall i: y_i = +1$$

$$0 \leq \xi$$

Finally we see that this quadratic constraint is equivalent to the SOCP constraint in Equation 6.56.

$$\min_{\theta, \beta, b, \xi} \theta$$

subject to

$$\left\| \left( \frac{\gamma}{\sqrt{\frac{\lambda}{2}} \beta} \right) \right\| \leq -\gamma + 1$$

(6.56)

$$\|A_i \beta\| + \beta^\top q_i + b \leq \xi_i - 1, \quad \forall i: y_i = -1$$

$$\|A_i \beta\| + \beta^\top q_i + b \leq \xi_i + 1, \quad \forall i: y_i = +1$$

$$0 \leq \xi.$$

Where $\gamma$ is just a placeholder for

$$\frac{1}{2} \left[ 1 - \sum_{y_+} \left( \left\langle \beta, \frac{P_i \beta_k}{\beta_k \top P_i \beta_k} + q_i \right\rangle + b \right) + \sum_{i=1}^{B} \xi_i - \theta \right].$$

### 6.4.3 Solving eMIL with BFGS

Another way of solving eMIL is to find a local minimum with a gradient based method. We use the quasi-Newton method L-BFGS (Byrd et al. 1995). To get a gradient of eMIL we use a smoothed version of the hinge-loss similar to (Chapelle 2007; L. Wang et al. 2008), which has the following form $\ell_\delta(P_i; q_i, y_i, \beta, b) =$

$$\begin{cases} 
\frac{(1-y_i \cdot f(P_i; q_i))^2}{2\delta} & \text{if } 1 - \delta < y_i \cdot f(P_i; q_i) \leq 1 \\
1 - y_i \cdot f(P_i; q_i) - \frac{\delta}{2} & \text{if } y_i \cdot f(P_i; q_i) \leq 1 - \delta \\
0 & \text{if } y_i \cdot f(P_i; q_i) > 1,
\end{cases}$$
where we choose an appropriately small $\delta$. The gradient is given below.

**Gradient of the smoothed hinge-loss**

The gradient of the smooth hinge loss with respect to $\beta$ and $b$ is given respectively by

$$
\frac{\partial}{\partial \beta} \ell_\delta(x_i, y_i, \beta, b) = \begin{cases} 
\frac{1-y_i f(x_i)}{\delta} \cdot -y_i \frac{\partial}{\partial \beta} f(x_i) & \text{if } \Phi \\
-y_i \cdot \frac{\partial}{\partial \beta} f(x_i) & \text{if } \Psi \\
0 & \text{if } \Omega
\end{cases}
$$

(6.57)

$$
\frac{\partial}{\partial b} \ell_\delta(x_i, y_i, \beta, b) = \begin{cases} 
\frac{1-y_i f(x_i)}{\delta} \cdot -y_i \frac{\partial}{\partial b} f(x_i) & \text{if } \Phi \\
-y_i \cdot \frac{\partial}{\partial b} f(x_i) & \text{if } \Psi \\
0 & \text{if } \Omega
\end{cases}
$$

(6.58)

Where

$$
\Phi \equiv 1 - \delta < y_i \cdot f(P_i; q_i) \leq 1 \\
\Psi \equiv y_i \cdot f(P_i; q_i) \leq 1 - \delta \\
\Omega \equiv y_i \cdot f(P_i; q_i) > 1
$$

And where the gradient of the ellipsoid predictor

$$f(q; P) = \sqrt{\beta^T P \beta} + \beta^T q + b$$

is given by Equation (6.59) and Equation (6.60).

$$
\frac{\partial}{\partial \beta} f(q; P) = q + \frac{P \beta}{\sqrt{\beta^T P \beta}}
$$

(6.59)

$$
\frac{\partial}{\partial b} f(q; P) = 1
$$

(6.60)
Table 6.1: Classification accuracy on the MUSK data sets (top block), image annotation (middle block), and TREC9 data. See Section 6.2.2 for description of previous approaches. Performance of those were obtained from the respective papers.

<table>
<thead>
<tr>
<th>Data set</th>
<th>eMIL</th>
<th>APR</th>
<th>EMDD</th>
<th>MI-SVM</th>
<th>mi-SVM</th>
<th>MICA</th>
<th>AL-SVM</th>
<th>AW-SVM</th>
<th>SDP</th>
<th>GPMIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musk 1</td>
<td>84.5</td>
<td>92.4</td>
<td>84.8</td>
<td>77.9</td>
<td>87.4</td>
<td>84.4</td>
<td>85.7</td>
<td>85.7</td>
<td>69.5</td>
<td>89.5</td>
</tr>
<tr>
<td>Musk 2</td>
<td>86.0</td>
<td>89.2</td>
<td>84.9</td>
<td>84.3</td>
<td>83.6</td>
<td>90.5</td>
<td>86.2</td>
<td>83.8</td>
<td>61.3</td>
<td>87.3</td>
</tr>
<tr>
<td>Tiger</td>
<td>88.8</td>
<td>-</td>
<td>72.1</td>
<td>84.0</td>
<td>78.4</td>
<td>82.0</td>
<td>78.5</td>
<td>83.0</td>
<td>73.6</td>
<td>87.4</td>
</tr>
<tr>
<td>Elephant</td>
<td>84.0</td>
<td>-</td>
<td>78.3</td>
<td>81.4</td>
<td>82.2</td>
<td>82.5</td>
<td>79.5</td>
<td>82.0</td>
<td>74.8</td>
<td>83.8</td>
</tr>
<tr>
<td>Fox</td>
<td>58.3</td>
<td>-</td>
<td>56.1</td>
<td>57.8</td>
<td>58.2</td>
<td>62.0</td>
<td>63.5</td>
<td>63.5</td>
<td>56.8</td>
<td>65.8</td>
</tr>
<tr>
<td>TST1</td>
<td>95.9</td>
<td>-</td>
<td>85.8</td>
<td>93.9</td>
<td>93.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>92.7</td>
<td>94.4</td>
</tr>
<tr>
<td>TST2</td>
<td>79.2</td>
<td>-</td>
<td>84.0</td>
<td>84.5</td>
<td>78.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>75.1</td>
<td>85.3</td>
</tr>
<tr>
<td>TST3</td>
<td>86.8</td>
<td>-</td>
<td>69.0</td>
<td>82.2</td>
<td>87.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.3</td>
<td>86.1</td>
</tr>
<tr>
<td>TST4</td>
<td>84.0</td>
<td>-</td>
<td>80.5</td>
<td>82.4</td>
<td>82.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>77.7</td>
<td>85.3</td>
</tr>
<tr>
<td>TST7</td>
<td>80.4</td>
<td>-</td>
<td>75.4</td>
<td>78.0</td>
<td>81.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>72.5</td>
<td>80.3</td>
</tr>
<tr>
<td>TST9</td>
<td>69.0</td>
<td>-</td>
<td>65.5</td>
<td>60.2</td>
<td>67.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>59.9</td>
<td>70.8</td>
</tr>
<tr>
<td>TST10</td>
<td>83.4</td>
<td>-</td>
<td>78.5</td>
<td>79.5</td>
<td>79.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.4</td>
<td>80.4</td>
</tr>
</tbody>
</table>
We compare the performance of eMIL on the following data sets: The MUSK1 and MUSK2 data sets described in (Dietterich et al. 1997), three image annotation data sets (Elephant, Fox, Tiger) introduced in (Andrews et al. 2002) and 7 splits of the TREC9 data set (TST1, TST2, TST3, TST4, TST7, TST9 and TST10) also described in (Andrews et al. 2002). The TREC9 data sets are extremely high-dimensional and sparse, having 66000 to 67000 features of which only a maximum of 30 are non-zero per instance. On average the MUSK1 and MUSK2 data sets contain approximately 6 and 60 instances per bag respectively. The average bag sizes for the image annotation and TREC9 data are 7 and 8 respectively.

We minimize the regularized empirical risk function, Equation (6.16) using L-BFGS (Section 6.4.3).\(^2\) To avoid numerical problems, when the ellipsoids are low rank we add a tiny positive constant to the diagonal of \(P_i\). This preserves the shape of the ellipsoid and makes \(P_i\) positive definite. We also experimented with different scaling factors (see Section 6.3.3 and Section 6.3.5), but could not generally improve test accuracy compared to simply using the estimated covariance matrix. By setting \(\kappa(\alpha_i) = 1\) we implicitly use \(\alpha_i = 0.5\) for all bags.

To be able to compare the performance of our method with previous methods we follow (Andrews et al. 2002) and employ the following procedure on all of the data sets: We use 10-fold cross-validation and search coarsely for an optimal regularization parameter \(\lambda\). This procedure is repeated 10 times on random permutations of the data and the results are averaged.

### 6.5.1 Feature space corresponding to kernels

For the MUSK data sets we use a Gaussian kernel with \(\gamma = 10^{-6}\). Since we optimize eMIL in the primal, we use kernel PCA to project the infinite dimensional feature vector to a lower-dimensional subspace. For MUSK2 we additionally restrict the number of basis vectors to 2500 to save memory.

To be able to optimize in the primal, we explicitly compute a finite dimensional representation of the features corresponding to the kernel. Following Zien et al. (2007), we use kernel PCA (Schölkopf et al. 2002) to find a \(d\) dimensional representation of the data from the kernel \(k(x_i, x_j)\). Since the representer theorem ensures that the optimal solution \(\beta\) lies

\(^2\) eMIL is available at [http://gabriel.kru.li/software.html](http://gabriel.kru.li/software.html)
in a finite dimensional subspace, we first find a basis for this subspace and then represent the instances in terms of this basis.

The basis needs to satisfy two criteria: (1) each basis vector has to be expressed in terms of the feature maps, and (2) the basis vectors should be orthonormal. Hence for a kernel matrix $K$, we need to find a set of coefficients in a matrix $A$ such that $A^\top KA = I$. One way to do so is to compute the eigenvalue decomposition of $K = V\Lambda V^\top$ and set $A = V\Lambda^{-\frac{1}{2}}$.

6.5.2 Results

On the musk data sets (Table 6.1, top) eMIL shows comparable performance to the methods motivated by finding a witness instance for positive bags (MI-SVM, MICA, AL-SVM) and the instance level maximum margin methods (mi-SVM, AW-SVM). The good accuracy of APR on the musk data sets can be explained by the fact that the hypothesis class of axis-parallel rectangles was specifically developed for this particular data set. For the image annotation data sets (Table 6.1, middle) eMIL has the best accuracy on the tiger and elephant data set and beats the MIL-SVM methods on the Fox data set. Our method achieves highest accuracy for some of the TREC9 data sets (Table 6.1, bottom), and comparable accuracy to the best method (GPMIL or mi-SVM) on all the TREC9 data sets, apart from TST2.

6.6 Defective Wheel Classification

In this section we apply eMIL to a real world MIL problem: Detecting defective wheels of freight trains from multiple dynamic vertical wheel force measurements. Late or undetected wheel defects on railway vehicles result in increased infrastructure maintenance due to damage of the railway infrastructure, like track systems or civil engineering works, and reduced availability of the vehicle pool, maintenance compounds and infrastructure. Most importantly, wheel defects are the major source of noise and vibration emissions of rail traffic. This makes the automatic, reliable and timely detection of wheel defects an essential part of any railway infrastructure safety monitoring system.

In the next chapter (Chapter 7) we will look closer at this data set, different types of defects and feature construction. In this section we
Table 6.2: Classification accuracy on the wheel defect data set. Rank gives the average rank. If two methods are equal, they both get the same rank. See text for details.

<table>
<thead>
<tr>
<th>Acc</th>
<th>eMIL</th>
<th>ALSVM</th>
<th>AWSVM</th>
<th>ALPSVM</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>0.70</td>
<td>0.64</td>
<td>0.68</td>
<td>0.63</td>
<td>0.67</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>0.05</td>
<td>0.08</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>Rank</td>
<td>1.5</td>
<td>3.5</td>
<td>2.7</td>
<td>3.5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

use one specific type of wheel defect from this data set, flat spots, since they naturally fit the MIL learning setting. For the purpose of comparing the different MIL methods we are not interested in absolute performance for detecting defective wheels, but only in the relative performance between the methods and the improvement of using MIL over binary classification. See Section 7.8.1 for a method optimized for maximal performance of detecting flat spots.

For detecting flat spots, we are given eight measurements per wheel, obtained by eight sensors installed on the tracks as the train runs over the measurement site in full operational speed (see Section 7.2.1 for in depth description and illustrations). A defect only impacts a measurement if it hits the part of the track where the sensor is installed directly. This results in eight measurements per wheel, with usually one measurement affected by a defective wheel. By considering all measurements from different sensors for the same wheel as a bag a natural setting for MIL is obtained. The labeled data was obtained by running a test train with a known configuration of wheel defects, resulting in 100 positive and negative bags.

6.6.1 Experimental protocol

Each measurement (instance) consists of a time series of the vertical wheel force. We use the Global Alignement (GA) kernel for time series, described in Cuturi et al. (2007) and Cuturi (2011). The GA kernel can be seen as a generalization of dynamic time warping (DTW) with a soft-max over all the alignments. To optimize eMIL in the primal we again project the features corresponding to this kernel to a lower dimensional subspace (see Section 6.5.1).
We compare our algorithm (eMIL) to the three deterministic annealing methods described in Gehler et al. (2007) in Table 6.2. We chose these methods because they solve the mi-SVM and MI-SVM formulations of MIL and our method could be seen as a generalization of the maximum bag margin method MI-SVM. Furthermore, an implementation was readily available. ALP-SVM is a balancing extension of AL-SVM, it needs to know an estimate of the fraction of positive points in a positive bag \( p^* \) a priori. For ALP-SVM, we provide extra information by setting \( p^* \) to 1/8 because we expect one sensor on average to see a defect. In addition, we compare a baseline method using a standard Support Vector Machine (SVM). To convert from bag labels to instance labels, we set all instance labels to the bag label.

The reported accuracy is averaged over 10 random permutations of the following two stage evaluation scheme: Half of the data is split for model selection and half for evaluation. On the first half of the data the optimal parameter for regularization is searched over \( \lambda \in 10^{[-2,\ldots,-5]} \). This is done with estimating test error with 5-fold cross validation for all values of \( \lambda \) and the \( \lambda \) with lowest test error is kept. This \( \lambda \) is then used to train the classifier on the full first half of the data set and test error is computed on the second evaluation half of the data set. If multiple parameter values give the same test accuracy, the one closest to the average is kept for training.

6.6.2 Results on the wheel data

From Table 6.2 we see that eMIL has the highest average accuracy on the test set. However, due to the large variation between the different splits of the data, the standard deviation is large. We also compared the ranks of the methods for each split, with the method with highest accuracy obtaining rank 1, and the lowest rank 5. We see in Table 6.2 that eMIL has average rank 1.5, which is the best among all considered methods. Interestingly, the naive SVM approach performs well.

6.7 Discussion

Motivated by the real world application of detecting wheel defects from multiple dynamic force measurements, we derive an ellipsoidal algorithm to solve MIL, resulting in a classifier that optimises a class
conditional distance between an ellipsoid and a hyperplane. We show that representing bags as ellipsoids amounts to finding a robust solution. Using only the assumption that the instances are samples from a distribution with finite mean and covariance, we derive an appropriate scaling factor for eMIL. We propose two approaches to solve the optimization problem: a CCCP approach which results in a sequential SOCP, and a quasi-Newton method based on L-BFGS.

Our algorithm results in state of the art performance on benchmark MIL data sets, demonstrating the effectiveness of the method. For classifying defective wheels with multiple instance time series data eMIL consistently outperforms AL-SVM, AW-SVM and ALP-SVM, which are recent improvements to SVM type MIL approaches.

In the next chapter we will look closer at the wheel defect classification data set.
Wheel defects on railway wagons have been identified as an important source of damage to the railway infrastructure and rolling stock. They also cause noise and vibration emissions that are costly to mitigate. We propose two machine learning methods to automatically detect these wheel defects, based on the wheel vertical force measured by a permanently installed sensor system on the railway network\(^1\). Our methods automatically learn different types of wheel defects and predict during normal operation if a wheel has a defect or not. The first method is based on novel features for classifying time series data and it is used for classification with a Support Vector Machine (SVM). To evaluate the performance of our method we construct multiple data sets for the following defect types: flat spot, shelling and non-roundness. We outperform classical defect detection methods for flat spots and demonstrate prediction for the other two defect types for the first time.

Motivated by the recent success of Deep Neural Networks (DNNs) for image classification we train custom artificial neural networks with convolutional layers on two-dimensional representations of the measurement time series. This neural network approach allows us to improve performance on wheels with flat spots and non-roundness due to encoding the structure of the measurement system and characteristics of the defect types as part of the neural network.

In this chapter we again encounter uncertainty or noise in the data. There are two sources of this. The first source is the system that measures the wheel vertical force of the trains. It consists of multiple sensors that measure different parts of the same wheel. For the defect type flat spot these parts can be modeled naturally as a multiple instance learning problem as in the previous chapter (Chapter 6). For other defect types the model is more complicated and we are going to express the uncertainty about the alignment of the wheel and the sensors in a cyclic shift invariant neural network.

The second form of uncertainty stems from the way that the labels for the training and testing are constructed. There is no information

\(^1\) Parts of this chapter appear in Kobayashi (2016)
about the orientation of a wagon as it is observed by the measurement system. This leads to uncertainty about the label of each wheel during training and testing. We deal with this by enlarging the training set to account for different orientations of the wagons, by filtering the data set using external information and by deriving theoretical bounds on the true error rates given the corrupted estimates.

7.1 Introduction

Early detection of serious wheel defects on freight trains are an essential part in preventing damage to the railway infrastructure and in providing the train operators with timely information on necessary repairs, that can prevent further deterioration of the wheels.

Wheel defects of railway vehicles directly cause an increase in attrition of and damage to the railway infrastructure, e.g., the track systems or the civil engineering works, thereby adding additional costs to maintenance and repair and leading to a reduced lifetime and availability of rolling stock. The life span of the railway infrastructure is significantly shortened by the negative effects of wheel defects. The life span of railway bridges for instance is calculated with an assumed maximal dynamical load of 21 tons. Due to wheel defects the actually occurring dynamical load can be up to 50 tons, or 270% higher than the theoretically assumed maximum, thus shortening the life span. Wheel defects also accelerate crack-growth on the rail tracks and lead to premature failure of the rail system.

Another important effect caused by wheel defects are ground vibration and noise emissions. In the European Union (EU) Project “Railway Induced Vibration Abatement Solutions” (RIVAS) 27 partners from nine countries investigated the source and mitigation measures for noise and vibration emissions. They found that reducing wheel defects by wheel maintenance significantly reduces vibration and noise emissions directly (Müller et al. 2013). Therefore, it is recommended to use timely and targeted maintenance of train wheels as an economic means to reduce emissions (Huber et al. 2015). This measure is all the more important as the density and usage of modern railway networks is steadily increasing and failures quickly disrupt operation of the whole network.

2 http://www.rivas-project.eu
or parts of it. Since 2008, all states in the EU are advised to employ noise emission ceilings. Switzerland started a noise abatement program based on emission ceilings that requires the infrastructure manager to curb emissions above the ceiling. This abatement programme leads to total costs of 1.5 billion CHF (Verheijen et al. 2015).

In this chapter we propose a method of detecting defective wheels. This classification method promises to increase the reliability of the railway infrastructure, to reduce the cost of freight train operation and to save additional investments on noise protection measures. To reach this goal without the costly construction of further measurement sites or newly built sensors, we propose the use of statistical methods that allow us to automatically inspect the existing data and extract the information about defective wheels that is already present.

Our proposed methods do neither require a model of the measurement system, nor of train dynamics or wheel defects. The methods enable us to predict defects on wheels where there is no prior understanding of how these defects manifest themselves in the measurements. The methods detect and classify different types of defects based on measurements during normal operation where the trains pass the measurement sites in full operational speed. The features that we have developed for the use in supervised learning are general and can in principle be used for any time series data and are not restricted to specific defect types. In a second step we automatically learn features directly from the raw measurement signal.

7.1.1 Contribution

Our main contribution are two methods for automatic railway wheel defect detection and classification through vertical force measurements of trains running in full operational speed. For the first method we design novel wavelet features for time series data from multiple sensors and we learn a classifier using a support vector machine. For the second method we design and train Convolutional Neural Networks (CNNs) for different wheel defect types by deep learning.

We evaluate our novel and other classical methods for wheel defect detection on two labeled data sets with different types of wheel defects, that we have constructed from calibration runs and from maintenance reports.
The algorithms and data sets contributed in this chapter lead to more accurate predictions of flat spots and enable the classification of two defect types that was not possible with traditional methods.

7.1.2 Related Work

To our knowledge machine learning methods for railway wheel defect detection have not been developed so far.

There has been some research on sensor systems for wheel defect detection on freight trains. In Nenov et al. (2011), the authors analyse the signal from acceleration sensors and demonstrate that they can visually see a difference between the measurements of wheels with flat spots and good wheels but they do not propose a method for detection. Another related work (Ho et al. 2008) advocates the use of Fibre Bragg Gating sensors for defect detection of rails to monitor track conditions. The authors investigate the wavelet decomposition of pressure signals but they do not propose a method or threshold for automatic defect detection. Jianhai et al. (2002) use continuous wavelet analysis of acceleration sensor data to visually inspect the measurements and conclude that there is a difference in the coefficients for wheel with flat spots and defect-free wheels.

Different kinds of track scales are in use in the field. They can in principle be used to detect flat spots. But to our knowledge they do not use machine learning to train a defect classifier. A general advantage of our proposed system is that the measurement system is relatively inexpensive, but we can show that it can still be used to detect wheel defects, thanks to our proposed machine learning methods.

7.2 Measurement System and Defect Types

7.2.1 Wheel Load Checkpoint

The infrastructure division of the Swiss railway operator SBB operates and maintains one of the most heavily used railway network of the world. In 2010, 95.4 km of trains travelled one kilometer of track on average; this value documents the highest utilisation of network capacity in the world (Badran et al. 2011). Automatically monitoring trains and network are thus important to minimise the risk of incidents
7.2 Measurement System and Defect Types

Figure 7.1: Multiple vertical wheel force measurements of a train wheel by the four sensors of one measurement bar. The wheel is affected by a discrete defect that manifests itself in the measurement of the first sensor. The remaining sensors do not directly observe the defect.

that quickly affect the scheduling of trains on the network. SBB infrastructure operates an integrated wayside train monitoring system that controls safety relevant aspects of the railway traffic and infrastructure.

As part of this system, the wheel load checkpoints (WLCs) measure vertical force through strain gauges installed on the rails. These devices are used for observing maximal axle load, maximal train load, load displacement and grave wheel defects. Our study investigates the use of machine learning methods to detect and classify wheel defects based on the data obtained through these wheel load checkpoints.

Each WLC consists of four 1m long measurement bars with four strain gauges (referred to as sensors in the following) per measurement bar. Since on each side two measurement bars with 4 sensors are installed, each wheel that runs over the WLC is measured eight times at different parts of the wheel. Figure 7.1 shows schematically the measurement of one wheel by one measurement bar. In this example a defect is directly observed by the measurement of the first sensor.

See Figure 7.2 for a diagram of one sensor. The strain gauges are installed perpendicular on the centerline of the railroad track and they
are combined into one vertical wheel force measurement. One sensor covers approximately 30 cm of the wheel circumference.

Figure 7.3 shows a picture of two measurement bars on the tracks installed on a WLC in the SBB network.

The wheel load checkpoints are installed on multiple strategic sites on the railway network: Ten on the border to Switzerland at the entrance to the railway network maintained by SBB and a dozen within the network.

7.2.2 Preprocessing

Getting reliable estimates of the vertical wheel force from the wheel load checkpoints is important for detecting defective wheels and for estimating train and axle load. When analyzing the measurements from the WLCs we found that there was a systematic difference (bias) in the measurements of the different sensors per site. We also found that the speed of the train during the measurements affects the measured
Figure 7.3: Two out of four measurement bars, one on the left and one on the right side of the tracks on a Wheel Load Checkpoint.

wheel force systematically. To correct the measurements we compute the sensor bias and speed dependence of each site on historic data and then correct the current measurements.

7.2.2.1 Sensor Bias

For wheels that do not have a wheel defect, the force measured by each sensor should be the same. If there are systematic differences between the sensors we can correct this by estimating the bias and correcting the measurements. We consider two types of bias, first a difference between the eight sensors on one side of the tracks (sensor bias), second a difference between all the sensors on the left side of the track and all the sensors on the right side of the track (left-right bias).

These biases are always defined as an average over the measurements of many axles. Since we correct the bias per side of the tracks and also correct for a difference between sides we add the subscript \(lr\) to mark the left or right side. A measurement of axle \(i\) from sensor \(s\) on side \(lr\) is thus given by \(x_{i,s,lr}\). We can now define the bias of one sensor with

\[
sensor_{bias}(s, lr) = \frac{\frac{1}{n} \sum_{i=1}^{n} x_{i,s,lr}}{\frac{1}{n} \sum_{i=1}^{n} \frac{1}{8} \sum_{s=1}^{8} x_{i,s,lr}}.
\]

First we correct this bias in the measurement to get
Figure 7.4: Sensor bias in June for site “Brugg”. Each line gives the bias of each sensor q1 to q4 for both sides left and right. Each point is averaged over one week, the number of measurements per average is given in brackets. (b) shows bias of measurements corrected with the bias computed with measurements from March to May.

\[ x_{i,s,l,r}^c = \frac{x_{i,s,l,r}}{\text{sensor\_bias}(s, l,r)}. \]

We then determine the bias between the left and the right side.

\[ lr\_bias(l,r) = \frac{1}{8} \sum_{s=1}^{8} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \sum_{l_r=\text{left}}^{\text{right}} x_{i,s,l,r}^c. \]

The bias that we then use to finally correct the measurement is a combination of sensor\_bias und lr\_bias:

\[ \text{total\_bias}(s, l,r) = \text{sensor\_bias}(s, l,r) \cdot lr\_bias(l,r). \]

This total bias is now used to obtain a corrected measurement \( x_{i,s,l,r}^{corr} \)

\[ x_{i,s,l,r}^{corr} = \frac{x_{i,s,l,r}}{\text{total\_bias}(s, l,r)} \]

Figure 7.4 and Figure 7.5 show bias of uncorrected and corrected measurements for trains running over two different measurement sites. They show that the bias on data not used to compute the bias can be significantly reduced by correcting the measurements with the bias obtained on older data.
Figure 7.5: Sensor bias in June for site “Gellert”. Each line gives the bias of each sensor q1 to q8 for both sides left and right. Each point is averaged over one week, the number of measurements per average is given in brackets. (b) shows bias of measurements corrected with the bias computed with measurements from March to May.

The strength of the bias depends on the speed of the train running over the sensors. This is due to differences in the subsoil below the tracks having different effects on the measurements at different speeds. Figure 7.6 shows that the bias is dependent on the speed of the train during measurement. On the site “Brugg” the bias increases starting from speeds around 45 km/h. The effect is especially pronounced in the 4th sensors of both sides (“q4 l” and “q4 r”). On site “Gellert” where the trains run on higher speeds, the bias is higher at lower speeds.

To get the best correction on average we only compute the bias using measurements obtained from trains with a typical speed. This is defined as the speed bracket of 20 km/h that most trains fall into.

### 7.2.2.2 Speed Dependence

In addition to the sensor bias, we found that the same locomotive (with a static weight) is measured differently at different speeds. As in the last section we compute this systematic difference and correct it per site.

The dependence of the measurement depends piecewise linear on the speed. As for the sensor bias this dependence is different for the
Figure 7.6: Speed dependence of the bias between February and April on two different measurement sites. Each line shows the bias at different speeds for all the sensors on each side. The histogram on top of the figures show the number of axles measured for each speed.

different measurement sites as it is an effect of the installation and subsoil below the tracks.

To determine the speed dependence of a site, we use the relative vehicle weight ($w_{rel}$). This refers to the vehicle weight given by one measurement relative to the average weight of all the measurements of the same vehicle. $w_i$ refers to the measured weight of the $i$th measurement and $lok(i)$ identifies the locomotive measured by the $i$th measurement. For $n$ measurements of different locomotives, we compute the relative weight in the following way. $1$ is the indicator function and $speed$ the speed at which the vehicle was measured.

$$w_{rel}^i = \frac{w_i}{\sum_{j=1}^{n} l_{lok(i)=lok(j)} w_j} \quad (7.1)$$

For linear speed dependence the dependence can be fully characterised by the slope $a$ and intercept $b$ of the line that fits the data best. We compute this fit by minimizing the sum of the squared errors.

$$speed\_dependence = \arg \min_{a,b} \left\| w_{rel} - a \cdot speed - b \right\|^2$$
7.2 MEASUREMENT SYSTEM AND DEFECT TYPES

PIECEWISE LINEAR DEPENDENCE A piecewise linear function with two pieces is given by three support points \((c_1, d_1), \ldots, (c_3, d_3)\) and can be expressed by the two linear functions

\[
\begin{align*}
f_1(x) &= d_1 + \frac{d_2 - d_1}{c_2 - c_1}(x - c_1) \\
f_2(x) &= d_2 + \frac{d_3 - d_2}{c_3 - c_2}(x - c_2)
\end{align*}
\]

We set \(c_1 = \min_{i=1}^{n}(\text{speed}_i)\) and \(c_3 = \max_{i=1}^{n}(\text{speed}_i)\), the upper and lower end of the speed range of all measurements. The mid point \(c_2\) and the \(d_k\) can now be obtained again by least squares minimization. The slope \(a_k\) and intercept \(b_k\) of segment \(k \in (1, 2)\) are now given by the following equations.

\[
\begin{align*}
a_k &= (d_{k+1} - d_k) / (c_{k+1} - c_k) \\
b_k &= d_k - (a_k \cdot c_k)
\end{align*}
\]

correction To correct a measurement \(w\) of a site for speed dependence we can now use the following formula, given the previously computed parameters of the (piecewise-)linear speed dependence.

\[
w^{\text{corr}} = w - (a_k \cdot \text{speed} + b_k - 1) \cdot w
\] (7.2)

Figure 7.7 shows the measurements from a site with piecewise-linear speed dependence of the weight measurements and the respective corrected measurements. Figure 7.8 shows a site with simple linear speed dependence. Due to correcting the speed dependence the deviation of the relative vehicle weight can be reduced by up to a factor of 1.84 on certain measurement sites.

7.2.3 Railway Wheel Defects

A relatively well understood wheel defect type is the flat spot or wheel flat. This defect occurs when the wheel stops rotating (for instance during an emergency brake) and is dragged along the track. Figure 7.9 shows an image of a flat spot on a railway wheel of SBB and the corresponding idealized measurement obtained by the WLCs if the flat spot directly hits a sensor of the measurement system. Grave wheel flats can be detected by looking at simple statistics (c.f. Section 7.5.7) of the
measurement if the defect hits the sensor perfectly. To be able to detect flat spots that are less grave or that do not hit a sensor directly, more advanced machine learning methods are required. We demonstrate such cases on our first data set in Section 7.8.1.

Apart from flat spot, other common wheel defects on railway vehicles are non-roundness and shelling (Nielsen et al. 2000; Nielsen 2009). Wheels with non-roundness have a high influence on the vibration and noise emitted by a passing train and, therefore, they are an important type of defect to detect (Müller et al. 2013; Nielsen 2009). Non-roundness, in contrast to shelling and flat spot, is a non-discrete type of defect. This characterization means that the defect affects a large part of the wheel and changes its shape in a non-local way. We create an additional data set that contains the defect types flat spot, non-roundness and shelling (Section 7.3.2) and then, we compare the performance of our two machine learning methods in predicting these three defect types.

7.3 DATA SETS

Two data sets from different sources are assembled to evaluate the performance of different methods for wheel defect detection and classification and to train various classifiers. For both data sets the signals that we use to predict a wheel defect are measured by the wheel load checkpoint (Section 7.2.1). The annotations or labels that provide the
information about the defectiveness and defect class of a wheel are collected from different sources. These data sets contain information about different types of defects as described in the following.

### 7.3.1 Data Set 1: Calibration Run

To acquire training data for flat spots, two wheels on different wagons were artificially damaged. The wagons were then added to a calibration train that was run over different measurement sites with different velocities and from both directions to calibrate the wheel load check points.

### 7.3.2 Data Set 2: Reprofile Events

To generate data for training and testing a classifier that can predict additional types of wheel defects, we aggregated the time and date of re-profile events and linked them to railway wagons. We used two sources for these events: The protocols of repair workshops of freight trains and the regular maintenance measurements of passenger trains. These were annotated with a defect class by an expert before re-profiling the defective wheels. We then categorized measurements of the wheel load checkpoints of the same wagons around the date of re-profiling. Measurements up to a week before re-profiling were considered defective.
Figure 7.9: Picture of a serious flat spot on a train wheel of SBB (a) and the resulting idealized wheel load measurement (b). (Picture taken from Wikipedia/Bobo11)

(according to the class label given by the expert), while measurements up to a week after re-profiling were considered defect free.

One complication of this data set arises from the lack of knowledge if the wagon passes the wheel load checkpoint with the same orientation as the wheels were annotated in the workshop. This lack of information leads to uncertain labels for the class of defective wheels, as not all wheels on a wagon necessarily share a defect. For the class of non-defective wheels this uncertainty does not pose a problem, since all wheels of a wagon are re-profiled and therefore are non-defective in our data set. We deal with this problem by adding both possible orientations of each wagon to the data set for the defective class of wheels. This augmentation of the data set introduces additional noise to the learning problem during training as non-defective wheels might be labeled defective. Nonetheless, we are able to train classifiers with high accuracy for all three types of defects (flat spot, non-roundness, shelling) based on data generated from this source. Since during testing the same uncertainty exists and actually non-defective wheels might have a defect class assigned, the error rate of the classifier is over-reported. We formalize this uncertainty in the next section.

7.4 LABEL AMBIGUITY

The uncertainty in the orientation of the wagon during measurement of the wheels can be understood as a contamination of the class conditional
distribution for defects. We model the observed data distribution for defects $P^{corr}$ and non-defects $Q^{corr}$ as a mixture of the true distribution of defects $P$ and non-defects $Q$ (Menon et al. 2015).

\[
P^{corr} = (1 - \alpha) \cdot P + \alpha \cdot Q \\
Q^{corr} = Q,
\]

where the mixture weight $\alpha \in [0, 1]$. The observed non-defective distribution $Q^{corr}$ correspond directly to the true distribution $Q$ since there is no ambiguity in this case (c.f. Section 7.3.2). To find the mixture weight $\alpha$ for the distribution of defects we distinguish between two slightly different assumptions about the label assignment process. Under the first assumption an expert looks at train wheels independently only assigns a label to certain wheels. This is reflected in $\alpha_1$ (7.5) below, where a wheel with label defective is either truly defective or from the joint distribution of both classes. We use $\pi$ to refer to the true proportion of positive labels (defects). Under the second assumption all wheels of a wagon are labeled, and therefore we do not need to estimate the true underlying base rate $\pi$ but have a 50% chance of true defectiveness for wheels sampled from $P^{corr}$. This leads to $\alpha_2$ (7.6). For both mixture weights we use that there are only two possible orientations for a wagon to pass the measurement system, and assume that both orientations are equally likely.

\[
\alpha_1 = \frac{1}{2} (1 - \pi) \\
\alpha_2 = \frac{1}{2},
\]

If we now set the parameter $\alpha_1$ or $\alpha_2$ into Equation 7.3 we get the contaminated distribution of defective wheels as

\[
P_{1}^{corr} = \frac{1}{2} P + \frac{1}{2} (\pi P + (1 - \pi) Q), \text{ or} \\
P_{2}^{corr} = \frac{1}{2} P + \frac{1}{2} Q,
\]

depending on the assumption from above.

While in general the base rate $\pi$ is unidentifiable (Scott et al. 2013), in this case we can estimate $\pi$ from the observed, corrupted proportion, because the mixture weights of the corrupted distributions are known.
7.4.1 True Error Rates

Following Menon et al. (2015) we can now use the values of $\alpha$ to estimate the true false positive rate (FPR) and false negative rate (FNR). We first define these rates in Definition 7.1. In the following when speaking about false or true positives or negatives we always mean the entries in the confusion matrix given below.

**Definition 7.1.** Confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Ground truth condition positive</th>
<th>Ground truth condition negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

We can now define the following rates.

\[
\text{False positive rate} \quad \text{FPR} = \frac{FP}{FP + TN} \\
\text{False negative rate} \quad \text{FNR} = \frac{FN}{FN + TP} \\
\text{True negative rate} \quad \text{TNR} = \frac{TN}{FP + TN}
\]

Using these definitions we relate the true rates FPR and FNR to the observed corrupted rates $\text{FPR}^{\text{corr}}$ and $\text{FNR}^{\text{corr}}$.

**Proposition 7.2** (Label ambiguity).

\[
\begin{bmatrix}
\text{FPR} \\
\text{FNR}
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
-\alpha & 1 - \alpha
\end{bmatrix}^{-1} \begin{bmatrix}
\text{FPR}^{\text{corr}} \\
\text{FNR}^{\text{corr}} - \alpha
\end{bmatrix}.
\]

This is a special case of Menon et al. (2015, Proposition 16) that is itself a result from Scott et al. (2013).
Proof. The proposition follows directly from the definition of the mixture distributions $P^{\text{corr}}$ and $Q^{\text{corr}}$ and linearity of the expectation.

\[
\begin{align*}
\text{FNR}^{\text{corr}} &= \mathbb{E}_{X \sim P^{\text{corr}}} [f(X) = -1] \\
&= \mathbb{E}_{X \sim (1-\alpha)P + \alpha Q} [f(X) = -1] \\
&= (1-\alpha) \cdot \mathbb{E}_{X \sim P} [f(X) = -1] + \alpha \cdot \mathbb{E}_{X \sim Q} [f(X) = -1] \\
&= (1-\alpha) \cdot \text{FNR} + \alpha \cdot \text{TNR} \\
&= (1-\alpha) \cdot \text{FNR} + \alpha - \alpha \cdot \text{FPR}
\end{align*}
\]

\[
\begin{align*}
\text{FPR}^{\text{corr}} &= \mathbb{E}_{X \sim Q^{\text{corr}}} [f(X) = 1] \\
&= \mathbb{E}_{X \sim Q} [f(X) = 1] \\
&= \text{FPR}
\end{align*}
\]

We can now write this as

\[
\begin{bmatrix}
\text{FPR}^{\text{corr}} \\
\text{FNR}^{\text{corr}}
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
-\alpha & 1 - \alpha
\end{bmatrix} \begin{bmatrix}
\text{FPR} \\
\text{FNR}
\end{bmatrix} + \begin{bmatrix}
0 \\
-\alpha
\end{bmatrix},
\]

and re-arrange to get the statement of the proposition.

We can estimate the true false negative rates under the different settings of independently labeled wheels ($\text{FNR}_1$) or fully labeled wagons ($\text{FNR}_2$).

\begin{align*}
\text{FPR} &= \text{FPR}^{\text{corr}} \\
\text{FNR}_1 &= \frac{1 - \pi}{1 + \pi} \left( \text{FPR}^{\text{corr}} + \frac{2}{1 - \pi} \text{FNR}^{\text{corr}} - 1 \right) \\
\text{FNR}_2 &= \text{FPR}^{\text{corr}} + 2\text{FNR}^{\text{corr}} - 1.
\end{align*}

In both settings the relationship between the expected true rates and the observed rates is not trivial because of the contribution of the observed false positive rate to true false negative rate.
7.4.2 Multiclass and Performance Scores

In this section we generalize the framework from above under the assumption of completely labeled wagons ($\alpha_2$) to the multiclass setting with $k$ classes. Instead of a distribution for defective and non-defective wheels ($P$ and $Q$) we now look at true distributions $P_i, i = 1, \ldots, k$ and observed, corrupted distributions $P_{j\text{corr}}, j = 1, \ldots, k$. As for the binary case the $P_{j\text{corr}}$ are a mixture of all the true class distributions ($P_1, \ldots, P_k$). The mixture weights are given as the matrix $\alpha$.

\[
P_{j\text{corr}} = \sum_{i=1}^{k} \alpha_{ij} \cdot P_i
\]

\[
\sum_{i=1}^{k} \alpha_{ij} = 1 \quad \forall j
\]

\[
\alpha_{ij} \geq 0, \quad \forall j, i.
\]

The binary setting is a special case with $k = 2$:

\[
P_{1\text{corr}} = \alpha_{11} \cdot P_1 + \alpha_{21} P_2
\]

\[
P_{2\text{corr}} = \alpha_{12} \cdot P_1 + \alpha_{22} P_2
\]

\[
\alpha_{11} + \alpha_{21} = 1
\]

\[
\alpha_{21} + \alpha_{22} = 1
\]

\[
\alpha_{ij} \geq 0, \quad \forall j, i.
\]

With $P_1 = P$ and $P_2 = Q$ and simplifying the $\alpha$ using the constraints (7.10) we get the original binary framework back.

Similarly to the proof of Proposition 16 (A.3) in Menon et al. (2015, Supplementary), which is based on a result from Scott et al. (2013) we can now derive true testing scores. We derive the full true confusion rate matrix that can then be used to compute multi-class scores.

**Proposition 7.3** (Multi-class confusion rate matrix). *Given the matrix $C_{\text{corr}}$ with $C_{ij\text{corr}}$ being the expectation of classifying a sample from the corrupted class $i$ with class $j$ and the mixture-weight matrix $\alpha$ we can compute the matrix $C$ with $C_{mj}$ the expectation of confusing class $m$ with $j$ for the true class distributions.*

\[
C = \alpha^{-1} C_{\text{corr}}
\]
7.5 Time Series Representation for Defect Detection

Proof.

\[
C_{ij}^{corr} = \mathbb{E}_{X \sim P_j^{corr}}[f(X) = i]
\]

\[
= \mathbb{E}_{X \sim \sum_{m=1}^{k} \alpha_{mj} P_m}[f(X) = i]
\]

\[
= \sum_{m=1}^{k} \alpha_{mj} \cdot \mathbb{E}_{X \sim P_m}[f(X) = i]
\]

\[
= \sum_{m=1}^{k} \alpha_{mj} \cdot C_{mj}.
\]

Here. We can write this as a matrix equality:

\[
C^{corr} = \alpha C
\]  \hspace{1cm} (7.11)

After inversion we get the expression for the true confusion rate matrix \( C \) in the proposition. \( \square \)

7.5 Time Series Representation for Defect Detection

An important step in any machine learning method is finding a representation of the original measurements that supports discrimination between different classes. For instance: The mean of the measurement signal of a wheel with or without a flat spot coincide if the weight of the axle is the same and the defect perfectly hits a sensor. The standard deviation on the other hand differs significantly, since the force exerted on the track is much higher for a wheel with the flat spot than for non-defective wheels. For other types of defects like shelling this observation does not hold, as the variance of the measured force does not significantly differ from a non-defective wheel, but there is a clear difference in higher frequency bands of the measurement, c.f. Figure 7.10. These observations suggest to decompose the signal by a multiscale wavelet analysis in order to extract indicative frequency features for time series data.
Figure 7.10: Signals and wavelet coefficients of a defective (right) and non-defective (left) wheel. The power in the high frequency coefficients D1-D2 reveal the defect.

7.5.1 Wavelet Transform

The Discrete Wavelet Transform (DWT) decomposes a signal over an orthonormal basis of dilated and transformed wavelets (Mallat 1999):

\[
\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \psi \left( \frac{t - k2^j}{2^j} \right),
\]

(7.12)

where \( \psi \) denotes mother wavelet, \( j \) and \( k \) the scale and shift parameters.

The orthogonal wavelets given by definition (7.12) at different scales \( 2^j \) resolve the original signal at different resolutions. The DWT can
7.5 Time Series Representation for Defect Detection

Thus be employed to construct a multiresolution signal approximation (Mallat 1999). An equivalent way of calculating the DWT is by passing the original signal through a series of appropriate high- and low-pass filters and subsampling operations, where at each level the output of the high-pass filter is stored as the detail coefficients for that level and the output of the low-pass filter is decomposed further at the next level until level \( \log(n) \) is reached, where \( n \) is the length of the original signal. If the high- and low-pass filters in this filter bank are derived from the child wavelets in Equation 7.12, the detail coefficients correspond exactly to the wavelet coefficients.

The wavelet transform has been extensively used in fields ranging from biomedical signal processing (Unser et al. 1996), geosciences (Kumar et al. 1997) to image compression (Skodras et al. 2001). Since weight measurement signals and the defect effects on the signal are both localized in time and frequency the wavelet transform explicitly encodes this local perturbation and, therefore, has an advantage over the fourier transform in our application. The weight measurement signals also show a self-similar behavior which suggests the wavelet transformation as an adapted set of basis functions with approximately the same amount of power per frequency band.

7.5.2 Wavelet Features for Defect Detection

To extract features from the measurement signals of the wheels, we first compute the wavelet decomposition of each signal. Each time series is now represented by the distributions of the wavelet coefficients at the different levels of the multiscale decomposition. To represent the distribution of the coefficients, \( n \) moments of the empirical distribution of the coefficients are computed. This representation captures higher order behaviour while still maintaining invariance to shift or scale of the defects as measured by the sensors. The procedure is summarized by Algorithm 7.1.

As explained in Section 7.2.1, we observe eight signals for each wheel that we want to classify. To compute features for one wheel, we first concatenate the measurements of all the sensors and then compute the wavelet features on this single time series. When we are processing localized defects, like a flat spot, that are observable as a change in vertical force on one sensor, the specific information, which sensor has observed a defect, does not play a role due to the scale
invariance of our feature construction method. For each sensor, the measurement signal can be divided into the regions of no load, raising slope, load measurement window and falling slope. Even though the load measurement window is relatively small we can still observe wheel defects that manifest themselves in one of the slopes or during the no load phase before and after the load measurement. To capture this information, a window of size three times the measurement window is used for feature construction. In all our experiments we use the Daubechies-5 wavelet family as basis functions (Daubechies 1992).

7.5.3 Time Series Kernel

In this section we generalize the wavelet features for time series (Section 7.5.2) that are used for the classification of wheel defects to a kernel function. Recall that the features were computed by looking at different moments of the empirical distribution of the coefficients in a multi-scale wavelet decomposition (Algorithm 7.1). Using the kernel embedding of distributions (Smola et al. 2007) (Section 2.4.1) we can represent the whole empirical distribution of the wavelet coefficients as an element in a Reproducing Kernel Hilbert Space (RKHS). This representation can then be used directly, without having to do density estimation, in kernel methods like the SVM that only operate on inner products.
The wavelet mean map kernel function uses the kernel mean map on the different levels of a discrete wavelet transform. Each level in the wavelet transform is represented by an element in Hilbert space. This way the distribution of coefficients on each level is captured.

The wavelet mean map kernel is computed by first computing the discrete wavelet transform for both time series. Then on each level the empirical mean map (Definition 2.10) of the coefficients is computed. As a last step the mean maps of each level are summed up.

**Definition 7.4.** Wavelet mean map kernel

Given two time series \( x \) and \( y \), and the \( T \)-level wavelet transform \( \psi(\cdot) \) (where \( \psi(\cdot)_{ti} \) gives the \( i \)th wavelet coefficient on the \( t \)th level of the decomposition, \( C_t \) the number of coefficients on level \( t \) and \( C \) the total number of coefficients), and a characteristic kernel \( k_1 \) (and respective feature map \( \phi_1 \)), the wavelet mean map kernel is given by the following function.

\[
k_W(x, y) := \sum_{t=1}^{T} \langle \hat{m}_{\psi(x)}, \hat{m}_{\psi(y)} \rangle = \sum_{t=1}^{T} \left( \sum_{i=1}^{C_t} \phi_1(\psi_{ti}(x)), \sum_{i=1}^{C_t} \phi_1(\psi_{ti}(y)) \right) \]

\[
= \frac{1}{C^2} \sum_{t=1}^{T} \sum_{i=1}^{C_t} \sum_{j=1}^{C_t} k_1(\psi_{ti}(x), \psi_{tj}(y)).
\]

\( k_W \) is a kernel function since \( k_1 \) is a kernel and the sum of kernels is a kernel. In principle \( k_1 \) does not need to be the same for all levels and different kernels can be used. If for instance a Gaussian kernel is used, different optimal scale parameters for the different levels could be found. This can either be done by cross-validation or as a computationally cheaper heuristic, the empirical MMD (Definition 2.12) between the two classes can be maximized over the kernel parameters.

### 7.5.4 Load Normalized Features

In addition to the wavelet features computed on the full concatenated signals of all the sensors we also compute wavelet features for each sensor separately. Whereas the feature construction based on the full signal pursued the strategy to capture as much information as possible,
the goal here is to construct features that are normalized with respect to the load measurement.

To this end, we first subtract an idealized measurement curve from the signal of each sensor and then compute wavelet features with Algorithm 7.1 on the difference. Additionally we add the mean squared error of the signal to the measurement template as a feature per sensor.

7.5.5 Measurement Site

Each wheel load checkpoint exhibits different physical characteristics due to small differences in the ground below the tracks and the curvature of the track before the checkpoint. These characteristics change the wheel load measurement. Small unevenness in the tracks also manifest themselves as noise or small bumps in the signal. Therefore, we add the site of the wheel load checkpoint as additional feature to enable different predictions based on the origin of the measurement site. We encode this information as a unary code or a one-hot vector, where every dimension represents a site and is 1 only for measurements from that site.

7.5.6 Load

A train with different load, but the same wagons results in different wheel measurements for the same defect types, since the weight of the train plays a significant role how the defect exerts its pressure on the sensors. Another important reason to add information about the load to the feature set arises from the following observation: Certain defect classes like non-roundness mostly change the average of a sensor reading, but only marginally affect higher order information. An oval wheel for instance will result in higher load measured by some of the sensors and lower load by others, but will not be detected as a defect wheel by individual load normalized measurements. The mean load of all the sensors, standard deviation over the mean load per sensor and the mean load for each sensor are added to the feature set.
7.5.7 Dynamic Coefficient

A conservative method to detect defective wheels and protect the infrastructure of SBB is based on the dynamic coefficient of the wheel load measurement. We use a simplified version of this coefficient, for each sensor it is given by

\[
d_{BW}(x) = \frac{\max(x)}{\text{mean}(x)},
\]

where max and mean refer to the maximum and average value of a sequence of measurements \(x\), respectively.

An alarm is raised when the dynamic coefficient of any of the sensors exceeds the threshold \(\theta = 3\).

7.6 AUTOMATIC REPRESENTATION LEARNING

An alternative to predefined feature representations are provided by deep neural networks that learn the features from data in a task specific way to maximize correct classification. In this section we introduce a learning method to automatically infer a representation of the measurements for the classification of wheel defects based on Deep Neural Network (DNN). These models have gained considerable popularity in recent years, mostly due to their success in image classification and segmentation tasks (Szegedy et al. 2015; Deng et al. 2014), in speech recognition (Mohamed et al. 2012) and quite recently in reinforcement learning for playing Go (Silver et al. 2016). The renaissance of artificial neural networks can be attributed mostly to four recent developments: (1) The availability of large amounts of supervised data that enable to train models with millions of parameters while still avoiding the risk of overfitting (Russakovsky et al. 2015); (2) The availability of fast compute architectures like GPU clusters that provide the computing speed to train these large models on large data sets; (3) Innovations in the design of artificial neural networks, specifically the design of deep networks and the stacking of multiple convolutional layers (Krizhevsky et al. 2012; Szegedy et al. 2015). (4) Innovations in optimizing deep neural networks like batch normalization, dropout and adaptive learning rate schemes (Duchi et al. 2011; Bengio 2012; Sutskever et al. 2013; Srivastava et al. 2014; Krummenacher* et al. 2016).

DNN for wheel defect detection alleviates the burden of the modeller to manually construct features and allows to learn representations from
time series directly. Another benefit is the flexibility that comes with designing decision functions as stacked activation layers. This flexibility allows us to design a network specifically for certain defect types.

### 7.6.1 2-Dimensional Time Series Representation

Motivated by the success of CNN on image classification tasks (Krizhevsky et al. 2012) we propose the use of 2D representations of the measurement signals for wheel defect detection. Recently Gramian Angular Fields (GAFs) have been proposed (Z. Wang et al. 2015) as a 2-dimensional encoding of time series data. This representation has been shown to capture cross-temporal dependencies and to enhance classification performance when used as input to a convolution network. A GAF is constructed by first transforming the time series to polar coordinates and then computing trigonometric sums between all points (See Z. Wang et al. (2015) for details of the construction).

In addition to the GAF encoding we also considered simply transforming the time series into the image of its 2D graph. This procedure is motivated by the fact that a human expert would also look at such a two-dimensional representation to classify wheel defects. The addition of the the value of the signal as the second dimension allows the network to learn different filters for different values of the signal at the same point in time (the first dimension). The procedure is summarized in Algorithm 7.2.

### 7.6.2 Convolutional Neural Network

The basic building block of our defect detection networks for wheel defects is a Convolutional Neural Network (CNN). Each CNN consists of multiple fully connected, convolutional layers. A convolutional layer is further composed of filtering and pooling layers.

**Fully Connected Layer** Neurons in a fully connected layer have full connections to all units in the previous layer. The layer outputs biased linear combination of its input, followed by a non-linearity. As a non-linearity we use the hyperbolic tangent (tanh) function.
**Algorithm 7.2** Compute 2D time series representation

**Input:** $X = (X_t)_{1 \leq t \leq N}$: time series.

**Input:** $r > 0$: resolution.

**Input:** $[V_{min}, V_{max}]$: window.

1: $h = \left\lceil \frac{V_{max} - V_{min}}{r} \right\rceil$
2: $M = 0_{h \times N}$
3: $X = \frac{X - V_{min}}{V_{max} - V_{min}}$
4: for $t = 1 \ldots N$ do
5: $M_{\left\lfloor \frac{X_t}{r} \right\rfloor, t} = 1$
6: end for
7: for $m = 1 \ldots N - 1$ do
8: Set entries between $M_{\left\lfloor \frac{X_t}{r} \right\rfloor, t}$ and $M_{\left\lfloor \frac{X_{t+1}}{r} \right\rfloor, t+1}$ to 1.
9: end for

**Output:** $M$: 2D graph of time series $X$.

**Convolutional Layer** A convolutional layer is a combination of a number of filtering layers, each followed by a non-linearity and a pooling layer. The settings chosen for each of these layers are specified below.

The filtering layer outputs convolutional products of the input by learnable filters with a fixed receptive field.

Every filter layer is followed by an activation function. We use a Parametric Rectified Linear Unit (PReLU), as it better back-propagates the gradient compared to the tangent hyperbolic or sigmoid functions, which can easily saturate. The PReLU non-linearity also prevents neurones from “dying out” as can be the case for the popular ReLU units, by introducing a learnable non-zero slope to the negative side of the input (He et al. 2015).

$$PReLU(x) = \max(0, x) + a \cdot \min(0, x),$$

(7.14)

where $a$ is an adaptable parameter.

The pooling layers reduce the resolution of the input time series and the learned features at each layer of the deep neural network. This max-pooling emphasizes the signal part of the weight measurements,
thereby robustifying the classification to small variations of learned features at each layer.

In all of our convolutional layers, we used a pooling layer with filters of size $2 \times 2$ applied with downsampling ratio of two, taking the maximum value among the four pixels in its receptive field.

### 7.6.3 Cyclic Permutation Network

![Cyclic Permutation Network](image)

**Figure 7.11:** Structure of the cyclic permutation network that automatically learns cyclic shift invariant features.

The cyclic permutation network (Figure 7.11) is designed to learn cross-sensor features invariant to a cyclic permutation of the eight recordings. Depending on its phase, a given wheel can generate a set of possible recordings, which is approximately stable by cyclic permutation of the eight recordings. This network architecture serves the purpose to encode this characteristic of cyclic invariance. Similar to the Defect Detection Network, features are extracted within each sensor recordings using a weight-shared **CNN**. However, extracted features
from all eight recordings are then concatenated together. The subsequent fully connected layer learns cross-sensor features to classify the wheel instead of using each sensor in an independent way. Moreover, the cyclic permutation network performs the concatenation in eight different ways corresponding to the eight cyclic permutation of the recordings. The fully connected layers across the possible permutations are weight-shared, and eventually, the maximum defect probability across all permutations is outputted. This forces the network to learn cross sensor features invariant by cyclic permutation of the sensors that detects defects.

7.6.4 Top Layer Features learned by the DNN

In this section we look at the features learned by the DNN and compare the filters learned by the network on the 1-dimensional or 2-dimensional time series representation. The results in this section were obtained by training on data set 2 (Section 7.3.2) and defect type flat spot. Examples of top-layer filters learned by the DNN directly on the 1-dimensional time series, as well as the features extracted by them are shown in Figure 7.12. We can observe that the network has been trained to detect a short quick oscillation in the time series. The extracted features on the defective input clearly shows the successful training of the model in detecting defect regions.

Figure 7.13 shows the top layer filters learned by the DNN on the 2-dimensional representation of the time series, and their respective extracted features on a defective and non-defective wheel. In general, the filters learned on the 2D representation encode high gradients in intensity, qualitatively presenting clear white/black delimitation. This suggests that the model focuses on 2D shape recognition rather than 1D pattern recognition as seen in filters learned on the time series directly.

7.7 Classification of Wheel Defects

Our ultimate goal is to find a function $f(\cdot)$ that given the wheel vertical force measurement $x$ of a wheel returns if the wheel is defective or not. To achieve this we use sets of measurements of wheels of a certain defect type and of non-defective wheels. We then use this training set of measurements and labels (the type of defect) to automatically find
Figure 7.12: Top layer filters (a) and features (b) learned by the 1-dimensional defect detection network for flat spots for a measurement of a defective (right) and non-defective (left) wheel.

Figure 7.13: Example of a top layer filter (a) and corresponding features of the signal of a non-defective (b) and defective (flat spot) (c) wheel learned on 2D representations.

a function that is expected to predict the defects of wheels not seen during training accurately.
One of the most popular models to find such a function are Support Vector Machine (SVM), see Section 2.2 for an introduction to maximum margin classification and SVM. Remember than an SVM finds a linear function parameterised by \((\beta, b)\) that separates two classes well. To allow non-linear decision functions, we use a Gaussian radial basis kernel function (Definition 2.8) on the feature vectors \(x\). We can now express the minimization problem in Equation 2.9 from Section 2.2 in the dual and employ the kernel trick (Section 2.4) to learn parameters \(\alpha_i\) and get the classification rule

\[
\text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b\right).
\]

To find the optimal regularization \((\lambda)\) and scale \((\gamma)\) parameters we minimize accuracy on cross-validation folds.

### 7.7.1 Classification with DNN

If we replace the hinge loss function in the softmargin SVM (Equation (2.9) in Section 2.2) with the logistic loss function \(\log (1 + \exp(-y_i\beta^T x))\) we get regularized logistic regression. This optimization problem has the advantage that optimization algorithms estimate probabilities of the class likelihoods in addition to the binary labels. Using the softmax function instead of the logistic loss this benefit can be generalized to an arbitrary number of classes. We will use these probability estimates through a SoftMax-layer in our DNN to combine the output of multiple classifiers for different measurements of the same wheel.

For a given input and \(N\) classes, its log-likelihood for belonging to class \(i\) equals

\[
p(x|i) = \log \left( \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)} \right), \tag{7.15}
\]

where \((x_i)_{1 \leq i \leq N}\) are the top-layer features of the network.

Unlike the previous section, where the classification function \(f(\cdot)\) was modeled as a linear function in a Hilbert space, that takes a fixed representation of the measurements, DNNs model this function as a hierarchical structure (layers) of linear combinations and activation functions (non-linearities) directly on the time series of the measurement (Section 7.6).
Figure 7.14: Structure of the MIL defect detection network for flat spots. The network consists of one CNN per measurement with weights shared across the networks. The defect likelihood of the whole wheel is given by the maximum defect likelihood across sensors.

7.7.2 Defect Detection Network for Flat Spots

We propose the following DNN (Figure 7.14) for flat spot detection. The network is designed using weight sharing CNN for the time series of each sensor (Section 7.6). The output of these networks is then fed to a special layer that combines the estimated defect/non-defect probabilities for each of the eight measurements per wheel based on the defect type that we want to detect. In the following we describe the DNN architecture and time series representations used in this research project.

In Figure 7.14 we depict the structure of the DNN that we use to train a model for the detection of flat spots. For each wheel the time series representation of each of the eight sensors is fed to a convolutional neural network given by the red and blue boxes. The convolutional
layers are shown in the red boxes and the fully connected layers are shown in the blue boxes. All the parameters are shared between the individual CNN. The output is then combined by what we call a MIL-Layer to estimate the final probabilities of the defectiveness of the wheel.

The MIL-layer combines the log-likelihoods of multiple Log-SoftMax layers by returning the maximal log-likelihood for the defect class and the minimal log-likelihood for the non-defective class.

Given a set of $s$ log-likelihoods for binary classification from $s$ sensors $x = (P^D_i, P^N_i)_{1 \leq i \leq s}$, where $P^D_i$ is the likelihood for defect and $P^N_i$ for non-defect from sensor $i$. Since $P^N_i = 1 - P^D_i$:

$$MIL(x) = (\max_i (1 - P^D_i), \max_i (P^D_i)). \quad (7.16)$$

Since a flat spot is a discrete defect and usually manifests itself only in one sensor reading, the MIL-Layer makes sure that if one measurement of the wheel captures the defect, the probability of the wheel having a defect is high. If defects are not “seen” by any sensor this probability will be low. This logic was already used for the SVM based MIL flat spot classifier in Krummenacher et al. (2013).

Zhou et al. (2002) also use neural networks to solve multiple instance learning.

### 7.8 Experimental Results

Our two machine learning methods for wheel defect detection are evaluated on the benchmark data sets. On the first data set we compare the Wavelet-SVM with benchmark flat spot prediction methods. We show that it greatly outperforms prior art based on thresholding methods and also on multiple instance learning with dynamic time warping. The thresholding methods measure statistics of the time series and trigger classification decisions in a non-adaptive or weakly adaptive way.

The second data set serves to demonstrate that the Wavelet-SVM can accurately classify all three defect types. We also compare the performance of the deep learning model on different time series representations by showing that the cyclic permutation network outperforms the simpler neural networks and also the Wavelet-SVM for non-roundness. For flat spots, the neural network with features learned on the 2d time series representation also outperforms the Wavelet-SVM.

For performance evaluation of the methods we compute three metrics: accuracy, precision and recall. Whereas accuracy gives the total
fraction of correctly classified wheels, precision measures the fraction of correctly predicted defects out of all predicted defects and recall the fraction of correctly predicted defects out of all defects (Rijsbergen 1979).

### 7.8.1 Data Set 1

This is the data set used in Chapter 6 to empirically demonstrate the effectiveness of a new algorithm (Krummenacher et al. 2013) for multiple instance learning (MIL) (Dietterich et al. 1997). Krummenacher et al. (2013) beat state-of-the-art MIL algorithms on this data set and get a classification accuracy of 70%. In this study features based on the Global Alignment (GA) kernel for time-series (Cuturi 2011; Cuturi et al. 2007) were used.

Using the features described in Section 7.5 with a SVM (Section 2.2) we were able to improve accuracy to 92% (Table 7.1).

We also consider another method to detect flat spots in this data set, that is not based on machine learning. It is a conservative threshold on the dynamic coefficient: a general measure of spread within one time series (Section 7.5.7).

With a threshold on the dynamic coefficient an accuracy of 60% is achieved. This is relatively low, as with random guessing already 50% accuracy could be achieved. It is thus important to note that the precision of this method is perfect with 100% of reported wheels being defective. So even though the method misses defective wheels it never raises a false alarm.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet-SVM (ours)</td>
<td>92</td>
<td>94</td>
<td>93</td>
</tr>
<tr>
<td>Dynamic coeff.</td>
<td>60</td>
<td>100</td>
<td>22</td>
</tr>
</tbody>
</table>
7.8 Experimental Results

Table 7.2: Results on data set 2

<table>
<thead>
<tr>
<th>Defect</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat spot</td>
<td>87 ± 3</td>
<td>89 ± 4</td>
<td>86 ± 6</td>
</tr>
<tr>
<td>Shelling</td>
<td>92 ± 2</td>
<td>92 ± 3</td>
<td>93 ± 3</td>
</tr>
<tr>
<td>Non-roundness</td>
<td>87 ± 6</td>
<td>87 ± 10</td>
<td>89 ± 4</td>
</tr>
</tbody>
</table>

7.8.2 Data Set 2 - SVM

Equipped with our general method of constructing features from multiple wheel vertical force measurements (Section 7.5) and learning a classifier from them (Section 7.7) we are now ready to predict other types of wheel defects as well. We also evaluate the DNN based method (Section 7.6) in this section.

The SVM classifier (Section 7.7) are trained on the labels obtained by this method for the defect types flat spot, shelling and non-roundness. First the data set is balanced to contain the same proportion of defective and non-defective measurements by over-sampling the smaller class. 1836 measurements are evaluated for flat spot detection, where 588 cases are classified as defective. For shelling, we received 6070 measurements, with 2678 being defective. For the non-roundness defect class, 688 cases out of 920 measurements are defective.

To estimate classifier performance we reserve 20% of the data set for testing and these “set aside” data are not used for training nor model validation. On the remaining data, three-fold cross-validation is performed to find the optimal hyper-parameters of the SVM and the Gaussian rbf kernel. This procedure is repeated 10 times on random permutations of the data set to get estimates of the variance. In Table 7.2 the performance on the reserved test set is reported for each defect type including standard deviation over the permutations. The performance on shelling is the best out of the three defect types. This observation can be explained by the fact that the training set for this defect type was by far the largest, so we were able to train a classifier with rel. This defect type also affects the wheel globally, so it’s harder to miss for the sensors as a flat spot. To improve the performance on flat spot and non-roundness we trained custom DNN and give the results in the next section.
For the defect type non-roundness, the load normalized features based on the load observed by individual sensors (c.f. Section 7.5) substantially contributed to an increase in accuracy. This effect can be explained by the observation that wheel non-roundness errors do not cause a large variation on the within measurement time series since they are a non-discrete type of wheel defects. They do introduce variations between the different measurements per wheel on the other hand and so features based on averages per measurement sequence are important. We will improve the classification performance for flat spot and non-roundness in the next section by using a custom DNN that is cyclic-shift invariant for classification of these defect types.

7.8.3 Data Set 2 - Deep Learning

Using the same data set as in the previous section we evaluate the deep learning method (Section 7.6) on the two defect types flat spot and non-roundness. To simplify the experiments we do not include additional features like speed, measurement site or template fit, but only consider the wheel vertical force measurements from the WLC sensors. Therefore, the performance of the SVM is slightly worse compared to the previous section.

As in the previous section the data sets were artificially balanced through up-sampling the under represented classes. Since the training of the DNN is considerably more time intensive than for the SVM, we only computed performance and standard deviation over 3 different random splits. As above 20% of the data were set aside to compute test error. From the training set another 10% were set aside as a validation set to benchmark performance online and decide on when to stop training.

To compute the 2D image of the time series the following parameters were used: $V_{\min} = -4, V_{\max} = 6$ as the window captures more than 99.9 percent of all the values, and $r = \frac{10}{N}$ to generate square pictures of size $N \times N$. We then divided the picture size by 4 by averaging every $2 \times 2$ non-overlapping pixels for computational reasons.

To prevent overfitting to the training set and to enable the model to explore a larger parameter space, we augmented the data by adding Gaussian noise and by randomly shifting and re-scaling the time series before applying image transformations.
Learning the parameters of a deep neural network relies on efficient optimization and substantial regularization to avoid overfitting. We have applied dropout regularization (Srivastava et al. 2014) on all the fully connected layers. To further improve generalization, we added an additional $\ell_2$ weight regularization penalty term in the cost function ("weight decay") to encourage smooth solutions by favouring small weights. We have employed stochastic gradient descent with Nesterov Momentum (Sutskever et al. 2013) to accelerate convergence. The gradient of the network is computed by using the backpropagation algorithm (Williams et al. 1986). Generally, the learning rate was set to decay inversely proportional to the number of epochs.

7.8.3.1 Flat Spots

Table 7.3: Performance of the deep models on flat spots in data set 2. Averaged over 3 folds for the deep models, and 10 folds for the SVM.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep 1D</td>
<td>88 ± 1</td>
<td>96 ± 2</td>
<td>79 ± 3</td>
</tr>
<tr>
<td>Deep 2D</td>
<td>89 ± 1</td>
<td>93 ± 2</td>
<td>85 ± 2</td>
</tr>
<tr>
<td>Deep GAF</td>
<td>87 ± 2</td>
<td>91 ± 1</td>
<td>81 ± 5</td>
</tr>
<tr>
<td>Wavelet-SVM</td>
<td>87 ± 3</td>
<td>87 ± 2</td>
<td>86 ± 5</td>
</tr>
</tbody>
</table>

In Table 7.3 we compare the performance of the different DNN models and the Wavelet-SVM. The only deep model that is able to out-perform the accuracy of the Wavelet-SVM is based on the 2D image of the time series. All of the deep models have smaller standard deviation and higher precision.

7.8.3.2 Non-roundness

With the DNN we can specifically model the large-scale effects of the non-discrete defect type non-roundness. In a similar way as for the class flat spot we take the maximum probability of defectiveness over multiple inputs. Instead of looking at just the maximum probability per sensor, here we determine the maximum probability over all cyclic permutations of the sensors. As the region of the wheel that rolls over the first sensor is arbitrary we want to be able to be invariant to a
specific way of shifting the sensors. Thanks to the symmetric way and the distances at which the sensors are installed we can look at cyclic shifts of the concatenated signal of all sensors to simulate different scenarios. The DNN trained to learn these cyclic shift invariant features is described in Section 7.6.3.

Table 7.4: Performance of the deep models on non-roundness in data set 2. Averaged over 3 folds.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy ±</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep MinMax</td>
<td>81 ± 1</td>
</tr>
<tr>
<td>Deep Concat</td>
<td>81 ± 2</td>
</tr>
<tr>
<td>Deep Cyclic</td>
<td>88 ± 1</td>
</tr>
<tr>
<td>Wavelet-SVM</td>
<td>84 ± 9</td>
</tr>
</tbody>
</table>

In Table 7.4 we compare the performance of this cyclic DNN with the DNN used for flat spot prediction (Deep MIL), a DNN that is trained on the concatenation of all the sensors (Deep Concat) and the Wavelet-SVM. Remember that the MIL-DNN used for flat spot prediction is trained by looking at the time series of each sensor individually and computing the loss on the sensor with highest probability of observing the defect. The performance of the different methods on the test set shows that MIL is an inadequate model for this type of defect since a wheel with a non-roundness defect can not be reliably identified on the basis of only one sensor measurement. This non-local behavior is in contrast to the challenge of predicting flat spot. Concatenating the sensors as is and not looking at the possible cyclic permutations resulted in training set accuracy similar to the cyclic shift network, but performance on the test set is significantly worse (Table 7.4). Intuitively ignoring the permutations leads to over-fitting as the measurements in the test set might be shifted arbitrarily.

In comparison with the Wavelet-SVM the cyclic DNN shows higher accuracy and precision and reduced variance. Unlike the DNN for flat spot we only trained the cyclic DNN for non-roundness directly on the 1D time series, as the increase in parameters due to the concatenation of measurements of the sensors prohibited efficient training of the model on the 2D representation.
We have presented two machine learning methods for defect detection on railway train wheels. The methods analyse multiple time series of the vertical force of a wheel under operational speed and output if a wheel has a defect or not. Both methods are trained automatically on measurements gathered from defective and non-defective wheels. The first method is based on novel general wavelet features for time series. The second method employs deep CNN to automatically learn features from the time series directly or from a 2-dimensional representation. We design cyclic shift invariant artificial neural networks for the detection of wheel flats and non-round wheels that model the relationship between the measurements inherent to these defects. To evaluate our methods we collect three data sets from different sources and demonstrate improved performance for predicting flat spot, shelling and non-roundness.

For this study we always optimized for accuracy as this simplifies the comparison of the different methods. A practitioner implementing such a method might be more interested in optimizing recall given a fixed high precision level to keep the number of false alarms low. We also balanced all the data sets to equal proportions of measurements from defective and non-defective wheels (see Section 7.8.2). This balancing leads to easily interpretable performance metrics, the baseline for accuracy is 50% for instance, and again makes comparing the methods simpler. But training on balanced data can also lead to a high false alarm rate if deployed in practice where defective wheels occur rarely. Therefore we recommend that for practical implementations recall and precision are controlled and instead of a binary prediction a numerical severity score is reported. This score could for instance be obtained through Platt-scaling (Platt 1999).

The methods that were developed in this chapter are currently being implemented as part of the SBB wayside train monitoring system. To improve the quality of the training and test data RFID tags will be deployed to enable perfect association between defect labels and measurements. Further future work consists of integrating external features into the deep learning models, optimizing for precision and predicting severity scores for the defects. For the prediction of severity scores we obtained promising preliminary results on regressing the flat spot length using support vector regression (Smola et al. 2004) and the wavelet features.
DISCUSSION

In this thesis we present novel machine learning methods for scalable learning on large noisy data sets. We contribute a method for large-scale regression with corrupted data, a distributed low-communication ridge regression algorithm and an adaptive stochastic optimization method for deep learning. We also propose a novel method for MIL and show how to solve the real-world problem of wheel defect detection through machine learning.

Below we discuss ideas and potential directions for extensions and future work.

• The algorithm presented in Chapter 3 is designed for linear regression. It would be interesting to extend the method to logistic regression to solve classification problems. Generalizations of regression diagnostics for logistic regression based on one-step approximations of the hat matrix exist (Pregibon 1981). Scalable approximations to these diagnostic measures would need to be developed.

In the context of pool based active learning (Ganti et al. 2012) leverage or influence could be used as measure to guide the selection of candidate examples for labelling. Similarly to information density (Settles et al. 2008) this would allow the rejection of outliers as candidates. Here the challenge would lie mostly in obtaining sample complexity bounds as the measure does not directly lead to a reduction in the size of the hypothesis class (Beygelzimer et al. 2009).

• For the adaptive stochastic optimization method in Chapter 5 different opportunities for distributed optimization arise. The main computational complexity is due to the approximation of the matrix square root and inverse of the gradient outer product. This work can be split across different workers or the data might be already held by different machines. If different sets of data points are held by different machines, a fast approximation to
the SVD similar to the distributed PCA algorithm in Liang et al. (2014) could be used. Secondly, a Loco-style algorithm as in Chapter 4 admits communication efficient distributed computation of covariance matrices when the data is split across the features.

A remaining open question is whether Ada-Full achieves optimal regret under distributions of the form in Section 5.2. Further theoretical questions open for exploration are a regret bound for RADAGrad or an error bound for the stochastic instead of the adversarial setting for adaptive gradient methods.

Given enough computational resources it would be interesting to see the performance benefits of RADAGrad on the state-of-the-art image classification architectures as they have many convolutional layers (He et al. 2015) and similar properties to RNNs (Simonyan et al. 2014). A challenge to here is implementing the SVD and QR-decomposition on a GPU to get fast algorithms.

- The results in Chapter 4 can be extended to other convex optimization problems like SVM and Lasso. In Heinze, McWilliams, and Meinshausen (2016) this is done for smooth, convex $\ell_2$ penalized problems. Additionally the algorithm preserves privacy which is shown in Heinze et al. (2015).

- A similar setting to MIL in Chapter 6 is learning with label proportions, here instead of the Boolean OR rule for label aggregation per bag, the proportion of labels is given. Yu et al. (2013) derive an SVM to learn in this setting. An ellipsoidal bag approximation similar to eMIL could prove to be useful also for learning with label proportions.

Other extensions would be to compute a minimum-volume enclosing ellipsoid instead of the empirical covariance and the usage of shrinkage estimators to reduce the variance of the covariance estimation (Chen et al. 2009).

- The measurements for the wheel defect detection problem in Chapter 7 come in a sequence of time series. At the moment this structure is modelled with MIL, wavelet-based features that are invariant to the order of the measurements or cyclic-shift invariant CNNs. A promising alternative would be to use RNNs to encode this structure.


Boutsidis, Christos and Alex Gittens (2012). „Improved matrix algorithms via the Subsampled Randomized Hadamard Transform.“ In:


Bibliography


Heinze, Christina, Brian McWilliams, and Nicolai Meinshausen (2015). „DUAL-LOCO: Preserving privacy between features in distributed estimation.“ In: NIPS 2015 workshop on Learning and privacy with incomplete data and weak supervision (cit. on pp. 74, 166).
Heinze, Christina, Brian McWilliams, Nicolai Meinshausen, Gabriel Krummenacher, and Hastagiri P. Vanchinathan (2014). „LOCO: Distributing Ridge Regression with Random Projections.“ In: NIPS Work-


shop on Distributed Machine Learning and Matrix Computations (cit. on pp. viii, 20, 57, 65).
Bibliography


Lanckriet, Gert R. G., Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan (2002). „A Robust Minimax Approach to Classifica-


Bibliography


Zhang, Qi, Sally A. Goldman, Wei Yu, and Jason E. Fritts (2002). „Content-Based Image Retrieval Using Multiple-Instance Learning.“ In: ICML (cit. on p. 100).


Bibliography

## NOTATION

### GENERAL

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>Label</td>
</tr>
<tr>
<td>$x \in \mathbb{R}^p$</td>
<td>Data point</td>
</tr>
<tr>
<td>$X \in \mathbb{R}^{n \times p}$</td>
<td>Data set</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of data points</td>
</tr>
<tr>
<td>$p$</td>
<td>Number of features / dimensions</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Dimensionality of down-projected space</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>Random projection</td>
</tr>
<tr>
<td>$\ell(\cdot, \cdot)$</td>
<td>Loss function</td>
</tr>
<tr>
<td>$\beta \in \mathbb{R}^p$</td>
<td>Model parameters</td>
</tr>
<tr>
<td>$\phi(\cdot)$</td>
<td>Feature map</td>
</tr>
<tr>
<td>$k(\cdot, \cdot)$</td>
<td>Kernel</td>
</tr>
<tr>
<td>$\psi(\cdot)$</td>
<td>Wavelet transform</td>
</tr>
<tr>
<td>$\mathbb{E}_X[f(X)]$</td>
<td>expected value of $f(X)$, w.r.t. distribution of $X$</td>
</tr>
<tr>
<td>$\mathbb{I}_{\text{expr}}$</td>
<td>indicator function returning 1 if the Boolean expression $\text{expr}$ is true and 0 otherwise.</td>
</tr>
<tr>
<td>$\mathcal{N}(\mu, \Sigma)$</td>
<td>Normal distribution with mean $\mu$ and covariance $\Sigma$</td>
</tr>
</tbody>
</table>
**LINEAR ALGEBRA**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|x|$</td>
<td>$\ell_2$-norm</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$|X|$</td>
<td>Spectral norm</td>
</tr>
<tr>
<td>$|X|_F$</td>
<td>Frobenius norm</td>
</tr>
<tr>
<td>$</td>
<td>X</td>
</tr>
<tr>
<td>$\sigma_1(X)$</td>
<td>Largest singular value</td>
</tr>
<tr>
<td>$\sigma_{\text{min}}(X)$</td>
<td>Smallest singular value</td>
</tr>
<tr>
<td>$\lambda_1(X)$</td>
<td>Largest Eigenvalue</td>
</tr>
<tr>
<td>$\lambda_{\text{min}}(X)$</td>
<td>Smallest Eigenvalue</td>
</tr>
<tr>
<td>$X^{1/2}$</td>
<td>Matrix square root</td>
</tr>
<tr>
<td>$X^{-1}$</td>
<td>Matrix inverse</td>
</tr>
<tr>
<td>$X^\dagger$</td>
<td>Pseudo-inverse</td>
</tr>
</tbody>
</table>
**ACRONYMS**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>Empirical Risk Minimization</td>
</tr>
<tr>
<td>RKHS</td>
<td>Reproducing Kernel Hilbert Space</td>
</tr>
<tr>
<td>MMD</td>
<td>maximum mean discrepancy</td>
</tr>
<tr>
<td>OLS</td>
<td>ordinary least squares</td>
</tr>
<tr>
<td>RMSE</td>
<td>root mean squared prediction error</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>MIL</td>
<td>multiple instance learning</td>
</tr>
<tr>
<td>eMIL</td>
<td>ellipsoidal multiple instance learning</td>
</tr>
<tr>
<td>mi-SVM</td>
<td>microscopic Multiple-Instance SVM</td>
</tr>
<tr>
<td>MI-SVM</td>
<td>macroscopic Multiple-Instance SVM</td>
</tr>
<tr>
<td>L-BFGS</td>
<td>limited-memory Broyden-Fletcher-Goldfarb-Shanno</td>
</tr>
<tr>
<td>CCCP</td>
<td>Concave-Convex Procedure</td>
</tr>
<tr>
<td>SGD</td>
<td>stochastic gradient descent</td>
</tr>
<tr>
<td>SOCP</td>
<td>Second order cone program</td>
</tr>
<tr>
<td>SRHT</td>
<td>Subsampled Randomized Hadamard Transform</td>
</tr>
<tr>
<td>SRFT</td>
<td>Subsampled Randomized Fourier Transform</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>DWT</td>
<td>Discrete Wavelet Transform</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
</tbody>
</table>
Bibliography

RNN  Recurrent Neural Network
LSTM long-short term memory
WLC  wheel load checkpoint
GAF  Gramian Angular Field
CURRICULUM VITAE

Name               Gabriel Krummenacher
Date of birth      26. Mai 1983 in Zürich

2005 – 2008        Bachelor of Science ETH in Computer Science, ETH Zürich
2008 – 2010        Master of Science ETH in Computer Science, ETH Zürich
2011 – 2016        Doctoral studies, Institute for Machine Learning, ETH Zürich
COLOPHON

This document was typeset in \LaTeX using the typographical look-and-feel classicthesis. The bibliography is typeset using biblatex.