Master Thesis

Bayes Risk for Large Scale Hierarchical Top-K Image Classification

Author(s):
Paulin, Mattis

Publication Date:
2013

Permanent Link:
https://doi.org/10.3929/ethz-a-009767867

Rights / License:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.
Bayes Risk for Large Scale Hierarchical Top-$K$ Image Classification

Mattis Paulin

Supervisor: Joachim Buhmann
Advisor: Patrick Pletscher
Group: Machine Learning
Universitätstrasse 6
8092 Zürich Switzerland
Abstract

Despite numerous efforts and recent progress, image classification remains a challenging problem, where computers are still outperformed by humans. In particular, the recent trend of large-scale image classification (thousands of images, hundreds of classes, high dimensional features), made popular by the ImageNet dataset [13] has recently received growing interest. Yet, the standard evaluation protocol, which reports only the misclassification rate, fails to produce well-behaved classifiers. The inherent difficulty of large-scale datasets, causes human beings to sometimes fail at the classification task (the three classes of ImageNet “softball”, “hardball” and “professional baseball” are for instance almost indistinguishable). Even when failing, humans nevertheless always predict an output semantically similar to the correct one. A hierarchy between concepts was therefore introduced to define an inter-class distance, to penalize classifiers outputing far-fetched labels, as for instance in the 2010 and 2011 editions of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). Another recent trend in classification protocols is to allow classifiers to output several guesses, only taking the best one into account. Also in use in ILSVRC, this leniency allows for datasets imperfections and ambiguous images.

The purpose of this master thesis is twofold. In a first part, we introduce Minimum Bayes Risk prediction to solve the problem of large-scale hierarchical top-K classification. Using an approximation of a submodular score and posterior class-probabilities given by a Logistic Regression, we get significant improvements over the naive prediction. In a second part, we report a preliminary work on improving the determination of the posterior probabilities with a new classifier called the Bayes Risk Machine. We report good improvements on top-1.
# Contents

1 Introduction ................................................. 3  
   1.1 Image Classification ...................................... 3  
   1.2 Contributions ........................................... 5  
   1.3 Experimental Settings .................................... 6  

2 Large-scale Classification .................................... 7  
   2.1 Descriptors and Feature Extraction ......................... 8  
   2.2 Making Use of Class Similarity ............................ 8  
      2.2.1 Losses Examples .................................... 9  
      2.2.2 Related Work ..................................... 10  
   2.3 Top K output ............................................. 10  

3 Standard Classification Approaches .......................... 12  
   3.1 Logistic Regression ...................................... 12  
   3.2 Structured Support Vector Machines ....................... 14  
      3.2.1 Two-Class Classification ............................ 14  
      3.2.2 General Case ..................................... 14  
   3.3 Maximum Score Prediction ................................ 15  

4 Minimum Bayes-Risk Prediction .............................. 16  
   4.1 Brute-Force Method ...................................... 17  
   4.2 Greedy algorithm ....................................... 17  
   4.3 Lazy Algorithm ......................................... 19  
   4.4 Experiments .............................................. 20  

5 Bayes-Risk Training .......................................... 23  
   5.1 Top-1 Bayes Risk Machine ................................ 23  
   5.2 Top-K Bayes Risk Machine ................................ 24  
   5.3 Resolution ............................................... 24  
   5.4 Experiments .............................................. 25  
      5.4.1 Top K ................................................ 25  

6 Conclusion .................................................. 28  

7 Acknowledgments ............................................. 29
Chapter 1

Introduction

The learning rate of a human mind has not yet ceased to impress. Recent studies \[32\] have shown that a child can learn concepts with little material, making strong generalizations with only a few examples. The field of image recognition is extremely challenging, as we strive to understand a generally tridimensional concept from its bidimensional projection, and has lately received a fair amount of attention, mainly for its usage in robotics, surveillance, and in general, task automation. The many possibilities offered by visual recognition can be easily envisioned (see for instance Fig. 1.1), yet the underlying complexity of the problem has up to now prevented us from meeting our expectations, though decent results are already available (Fig. 1.2).

![Figure 1.1: Image recognition, as envisioned 20 years ago (Terminator II).](image1)

1.1 Image Classification

In this thesis, we focus on the particular task of image classification, where the goal is to learn to assign one and only one label to an image (Fig 1.3). This task can be relevant in itself for search engines (though one would probably rank rather than classify) or as a second step of a sliding window detection protocol \[21\]. It was recently made popular by the PASCAL challenge \[14\], and its large-scale version: ILSVRC \[11\].

The general approach to designing a classifier is to rely on an annotated database, split between a training part, whose images and labels are known, and a testing part, whose labels are to be inferred. To evaluate a classifier’s performance, the predicted testing labels are compared to the ground truth, and a particular score measures the output deviation. Designing classification
databases is not an easy task. Two main methods exist: using a search engine to find images that answer to a particular set of labels (e.g. Tiny Images [33]), or using humans to carefully select and sort images (ImageNet[13], Caltech101 [15], Caltech256 [16]). The second method gives cleaner datasets but requires far more time and resources. Yet, even when human-annotated, databases show a bias that usually prevents information from being transferable from one to an other, even when they share classes.

Although seemingly well-defined, the classification problem faces a large number of methodological issues, most of which are usually ignored. Beyond the loss of information during the camera projection, it is also sometimes impossible for humans to attribute the correct label to an image. Some images may be ambiguous, as they represent several different concepts, as in Fig. 1.4, some others may contain several different concepts (a “bird” in a “forest”, under a “cloudy sky”). The problem of the level of expertise also arises when not knowing at which family level a concept should belong. For instance, should a sparrow be labeled as its exact subspecies, as “sparrow”, as “bird”, or as “animal” (Fig: 1.5)?
1.2 Contributions

In this thesis, we start by formally defining the classification problem in chapter 2. In chapter 3, we give examples of standard approaches to the problem. Our main contribution, described in chapter 4, is to introduce a new prediction method based on the Minimum Bayes Risk. Our method operates under the assumption that the posterior probabilities of descriptors belonging to a class are known, and designs a score to predict a shortlist of possible labels. Using submodular properties of this score, we propose an efficient approximation and get significant improvements compared to the brute-force case, as well as having theoretical guarantees. We propose several experiments on ImageNet [13] subsets that shows our method performs better than standard approaches. In a last part (chapter 5), we show preliminary results on improving the posterior probabilities estimate, to better fit our prediction score. In some cases, we are again able to improve the global performance of our classification.
1.3 Experimental Settings

For our image experiments, we use parts of the ImageNet dataset \cite{10}. In particular, we consider the subtree with root “Athletic Games”, which contains 51 classes, and around 40 000 images. We split it 50%-50% in train and test sets. We use 3-fold cross-validation to determine our exogenous parameters. For larger scale experiments, we use the subset determined by the ImageNet Large Scale Visual Recognition (ILSVRC) 2010 competition, which contains 1 000 classes, around 500 000 images in the train set, 150 000 in test set, and 5 000 in a validation set.
Chapter 2

Large-scale Classification

Throughout this thesis, we denote $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ the dataset, formed of pairs of features and labels, with $x_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, C\}$. We adopt the standard pipeline for supervised image classification. Images are in a first offline step preprocessed into descriptors. A classifier (i.e., a map $\mathbb{R}^d \rightarrow \{1, \ldots, C\}$) is then learned, using the training part of the dataset. The said classifier is in turn used to infer the missing labels of the test set. The detailed pipeline is illustrated in Fig. 2.1.

![Classification pipeline](image)

Figure 2.1: Classification pipeline: the training part of a dataset is given as an output to the classifier, which solves a minimization problem involving a weight parameters, made to fit data by the mean of a loss function, but prevented from overfitting by a regularizer. The weight is used to determine a predictor that estimates the testing set’s labels, which are in turn compared to the ground truth. A score aggregating these deviations measures the global performance of the classifier.
2.1 Descriptors and Feature Extraction

An important part of a classification process relies on the initial step of abstracting images into vectors of a space where they can be easily discriminated. This step is not always used, as recent progress in deep learning [1, 20] have been quite successful in the latest challenges (ILSVRC [11]).

We settle for the Fisher Vector image descriptor [28], which aggregates local features into a global image descriptor. We start by extracting $T$ local descriptors (e.g. SIFT [22], color descriptors [23],...) on a subsample of our images, then learn a diagonal-variance Gaussian Mixture Model (GMM, see [2, 6]). Let $\mathcal{L} = \{\ell_1, \ldots, \ell_T\}$ be the extracted local descriptors, using $k$ centroids, we get the following likelihood:

$$u_\lambda(\ell) = \sum_{i=1}^{k} A_i u_{\mu_i, \sigma_i}(\ell),$$

where $\lambda = \{(A_i, (\mu_i), (\sigma_i))\}$, with $A_i$ the mixture weights that sum to one and $u_{\mu_i, \sigma_i}$ the Gaussian distribution function with mean $\mu_i$ and variance $\sigma_i$:

$$u_{\mu_i, \sigma_i}(\ell) = \frac{1}{\sqrt{2\pi\sigma_i}} \exp\left(-\frac{(\ell - \mu_i)^2}{2\sigma_i^2}\right)$$

Given an image to encode, we use the gradient of the log-likelihood as a measure of its deviation to the previously learned model

$$G^\mathcal{L} = \nabla_\lambda \log u_\lambda(\mathcal{L}).$$

Assuming the descriptors are iid, the distribution splits accordingly. The derivatives over the GMM weights gives no relevant information, we therefore only keep the ones over $\mu_i$ and $\sigma_i$:

$$G^\mathcal{L}_{\mu_i} = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\mu_i} \log A_i u_{\mu_i, \sigma_i}(\ell_t)$$

$$G^\mathcal{L}_{\sigma_i} = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\sigma_i} \log A_i u_{\mu_i, \sigma_i}(\ell_t).$$

To later on be able to compare descriptors, with respect to the $L_2$ norm, we multiply $G^\mathcal{L}$ by the left side Cholesky decompositions of the Fisher information matrix (see [2]). We only use the $\mu$ part of the Fisher Vector, as the $\sigma$ part is not found to carry much additional information. The Fisher Vectors are then normalized by taking the square root of every component, as in [27]. In the end, the input $x$ of our classifier are the concatenated $G^\mathcal{L}_{\mu_i}$ and $G^\mathcal{L}_{\sigma_i}$ for all $i \in \{1, \ldots, k\}$. For a good survey showing performance comparisons for these descriptors, see [18]. Fig. 2.2 illustrates the feature extraction pipeline.

For our experiments, we use Fisher Vectors with 64 GMM centroids. We densely extract both SIFT [22] and color descriptors as in [27], reducing them by PCA to respectively 64- and 48-dim, concatenating them into a 7168-dimensional descriptor.

2.2 Making Use of Class Similarity

Most of the time, raw misclassification is used as a performance evaluation, meaning that the returned label of a classifier can either be right or wrong. Yet, when the result is wrong, it gives
Figure 2.2: Fisher description pipeline. In a first step (a), local features are extracted from the image (in this case, SIFT [22]). The distribution of these local descriptors is learned through a GMM process (b). To compute the global descriptor of an image, the deviation of its local features with respect to the learned model is used.

no indication of how wrong it is. It is for instance preferable to mistake a pear for an apple rather than for a whale. The dataset we use in most of our experiments, ImageNet [13], already has this type of information, as a global taxonomy of concepts.

We introduce a class loss term, $\Delta(y', y)$, which quantifies the error made when misclassifying $y'$ for $y$. Note that this loss is not necessarily symmetric. In particular, an image with “animal” for ground truth could be assigned to any animal without making an error, while the reverse does not hold.

When given a taxonomy of classes as a tree (as in ImageNet), we can derive several possible class losses. In our work, we explore three different losses.

2.2.1 Losses Examples

Node Distance Loss. It is defined as the distance between the two classes through their lowest common ancestor in the tree (Fig. 2.5). This loss is symmetrical and scales logarithmically with the number of classes. It is used to classify documents in [5].

Ranking Distance Loss. It is defined as the size of the subtree whose root is the lowest common ancestor of two classes. It scales linearly with the number of classes. To the best of our
knowledge, no publication makes use of this loss.

![Figure 2.4: Illustration of the Ranking Distance Loss. Here, $\Delta(y', y) = 11$.](image)

**ILSVRC Loss** As in the popular ImageNet Large Scale Visual Recognition Contest [11], $\Delta(y', y)$ is defined as the height of $y'$ in the subtree originating from the lowest common ancestor between $y$ and $y'$.

![Figure 2.5: Illustration of the ILSVRC Loss. Here, $\Delta(y', y) = 3$.](image)

The choice of a particular loss is application dependant and is usually determined by a domain expert.

### 2.2.2 Related Work

In the domain of textual documents, hierarchical classification have already received some attention. [30] defines a hierarchical kernel used inside the structured support vector machine framework, while [5] proposes a new classifier that use a feature map that reflects hierarchy.

In computer vision, hierarchy has received little attention. In early works on large-scale classification, [12] use a variant of the Bayes Risk to improve the hierarchical score.

### 2.3 Top $K$ output

To partially solve the possibility for an image to contain several concepts, we allow classifiers to produce several outputs. When comparing to the ground truth, only the best match is taken
into account for the final score. Formally, for top-$K$ classification, if $y = \{y_1, \ldots, y_K\}$ is the output of a classifier and $y$ the ground truth for the same image, the score is:

$$\Delta(y, y) = \min_{k \in \{1, \ldots, K\}} \Delta(y_k, y).$$  \hspace{1cm} (2.5)

Not many authors treat of top-$K$ classification. The authors of [17] also work on predicting several outputs, yet only for flat classification error. Instead of having one multiple output classifier, they jointly train $K$ of them.
Chapter 3

Standard Classification Approaches

We start by reviewing some of the popular methods for tackling large-scale classification. The first method, logistic regression (section 3.1), focuses on computing posterior class probabilities \( p(y|x) \) (i.e., the probability that descriptor \( x \) belongs to class \( y \)), while structured support vector machines (SSVM, section 3.2) compute the best separator between classes. Both these classifiers are used with the maximum score predictor (section 3.3).

3.1 Logistic Regression

We can express the probability of a point \( x \) to belong to a given class using Bayes’ rule:

\[
p(y = k | x) = \frac{p(x | y = k) p(y = k)}{\sum_j p(x | y = j) p(y = j)}. \tag{3.1}
\]

Equation (3.1) can be rewritten as:

\[
p(y = k | x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \tag{3.2}
\]

with \( a_k = \ln[p(x | y = k) p(y = k)] \).

The idea behind logistic regression is to assume that the class-conditional densities are Gaussian with the same covariance matrices:

\[
p(x | y = k) \propto e^{(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)}. \tag{3.3}
\]

We can therefore write \( a_k \) as a quadratic function:

\[
a_k = x^T \Sigma^{-1} x + w_k x + w_0. \tag{3.4}
\]

As all covariance matrices are assumed to be equal, the quadratic terms can be simplified in the formula for \( p(y = k | x) \), and \( a_k \) therefore becomes affine. Furthermore, adding a dimension to the feature vectors allows us to get rid of the constant term. In the following, we denote in matricial form \( w = (w_k)_{k=1}^K \).

Given a dataset \( D \), we can evaluate its likelihood with respect to our model, assuming its samples are iid:
\[ \mathcal{L}(\mathcal{D}|w) = \prod_{x,y \in \mathcal{D}} p(y|x, w). \] (3.5)

Rather than maximizing \( \mathcal{L} \), as usual we instead minimize the negative log-likelihood:

\[
\ell(\mathcal{D}|w) = -\sum_{x,y \in \mathcal{D}} \left[ w_y x - \ln \sum_k \exp(w_k x) \right]. \tag{3.6}
\]

In order to prevent \( w \) from taking too large values, we add to the likelihood of the dataset the prior that \( w \) is Gaussian distributed, with a mean of zero and a covariance of scale \( \lambda \).

The resulting objective becomes, after simplification:

\[
\ell(\mathcal{D}|w) = \sum_{x,y \in \mathcal{D}} \left[ \ln \sum_{k \neq y} \exp(w_k x) \right] + \frac{1}{2} \lambda \|w\|^2. \tag{3.7}
\]

The intuition behind this formula is that for each datapoint \((x, y)\), \( w \) needs to “discriminate” \( x \) well, which means that the misclassification \( w_k x \) for \( k \neq y \) should be small. This is exactly what the first term of the loss achieves. For more details, see [2].

**Figure 3.1:** An example of partitioning using logistic regression. Points are drawn from three different normal distributions.
3.2 Structured Support Vector Machines

We here give a quick insight into the support vector machine theory, initially due to Vapnik [7].

3.2.1 Two-Class Classification

Let us briefly consider the case \( C = 2 \). A natural way to classify a set of such examples would be to try to find a hyperplane that splits datapoints into the two classes. This approach suffers from two pitfalls: such a hyperplane may not exist and if it does, it may not be unique. The remedy to the latter is to select the hyperplane with the largest “margin”, which is the perpendicular distance between the hyperplane and the closest of the datapoints [6]. For a dataset \( D = \{(x_i, y_i)\}_{i=1}^N \), with \( y_i \in \{-1, 1\} \), the degree of freedom in the norm of \( w \) allows us to fix \( y_i(wx_i + b) = 1 \) for the points at the minimal distance. We therefore need to maximize the geometric margin \( \frac{1}{\|w\|} \) or equivalently minimize \( \frac{1}{2}\|w\|^2 \). The optimization problem can be formalized as:

\[
\begin{align*}
\minimize_\{w\} & \quad \frac{1}{2}\|w\|^2 \\
\text{s.t.} & \quad y_i(wx_i + b) \geq 1
\end{align*}
\] (3.8)

In the case of a non linearly-separable dataset, we allow datapoints to lie on the wrong side of the separating hyperplane, by introducing a penalty term \( \xi_i \):

\[
\begin{align*}
\minimize_{w,\xi_i} & \quad \sum_{i=1}^n \xi_i + \frac{\lambda}{2}\|w\|^2 \\
\text{s.t.} & \quad y_i(wx_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0
\end{align*}
\] (3.9)

By looking at the constraints, we can see that if \( y_i(wx_i + b) \geq 1 \), \( \xi_i \) would have no reason to be anything else than zero, otherwise, a trivially better solution to the minimization problem would be \( \xi_i = 0 \). In the same fashion, if \( y_i(wx_i + b) < 1 \), the logical choice for \( x_i \) is \( 1 - y_i(wx_i + b) \). We can therefore cast the problem into:

\[
\arg \min_{w} \sum_{i=1}^n \max(0, 1 - y_i(wx_i + b)) + \frac{\lambda}{2}\|w\|^2
\] (3.10)

3.2.2 General Case

It is not straightforward to transform this loss into a multiclass version. Many methods have been devised to tackle the problem of multi-class max-margin classification, including approaches based on the two-class problem, such as the popular one-versus-rest (teaches \( C \) classifiers to separate one class from the others) or one-versus-one (a classifier is trained for each pair of classes). Yet, it is not easy in these methods to see how the outputs of each classifier should be combined. Instead, we modify the loss to take the \( C \) classes into account, following a scheme devised by Crammer and Singer [9].

The idea is to combine features and labels in a vector by the mean of a feature map \( \Phi : \mathbb{R}^d, \{1, \ldots, K\} \rightarrow \mathbb{R}^m \), and define the prediction function \( f \) for a given \( w \) as

\[
f_w(x) = \arg \max_y \{w, \Phi(x, y)\}. \tag{3.11}
\]
In the case of classification, the standard feature map is taken as the concatenation of \(\{\delta_{r,y_i}, x_i, r \in [1,K]\}\), resulting in a \(dK\)-dim vector. Note that writing \(w\) as \((w_1, \ldots, w_k)\), we have \(<w, \Phi(x_i, y_i)> = <w_y, x_i>\), where \(w_k\) is the \(k\)-th row of \(w\).

To find the best candidate \(w\), the margin for a training sample is defined as \(\ell(x_i, y_i|w) = \max_y (\Delta(y_i, y) - <w_y, x_i> + <w_y, x_i>)\), and corresponds to an upper bound of \(\Delta(y_i, f(x_i|w))\), as shown in [26]. If we forget for a moment about the \(\Delta(y_i, y)\) term, a parallel can be drawn between the Max-Margin loss and the previously seen Logistic loss. For the latter, the log sum exp function acts as a soft-max, which can be very close to the max if one value stands out.

As is the case with \(C = 2\), jointly maximizing the margin and minimizing the penalty for misclassification gives us:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{\lambda} \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|w\|^2 \\
\text{s.t.} & \quad \max_y (\Delta(y_i, y) - <w_y, x_i> + <w_y, x_i>) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0.
\end{align*}
\] (3.12)

After the same reasoning as before, this can be written as

\[
\begin{align*}
\arg \min_{w,b} \sum_{i=1}^{n} \max(0, \max_y (\Delta(y_i, y) - <w_y, x_i> + <w_y, x_i>) + \frac{\lambda}{2} \|w\|^2)
\end{align*}
\] (3.13)

### 3.3 Maximum Score Prediction

Surprisingly, both logistic regression and support vector machine uses the same standard prediction, called maximum score. For the former, the posterior probabilities are

\[
p(y|x) = \frac{\exp(w_y, x)}{Z(x)}.
\] (3.14)

The maximum likelihood prediction is therefore, when \(K = 1\).

\[
y^{\text{PRED}} = \arg \max_y p(y|x) = \arg \max_y \exp(w_y, x).
\] (3.15)

The SVM loss optimizes for this precise predictor. For general \(K\), the maximum score prediction is defined as the \(K\) maximal elements of the score \(w_y, x\).
Chapter 4

Minimum Bayes-Risk Prediction

As seen previously, when given posterior probabilities $p(y|x)$ for a sample $x$ to belong to class $y$, the optimal score for the 0-1 loss (i.e., the flat misclassification rate: 0 for the same class, 1 for the rest) is

$$y^{MS} = \arg \max_y p(y|x) \quad (4.1)$$

When the loss is no longer 0-1, the expected score of a prediction $y$ is

$$s(y, x) = \sum_{y' = 1}^{C} \Delta(y, y') p(y'|x). \quad (4.2)$$

As we in the end report the sum of the scores $\Delta(y^{\text{pred}}, y^{\text{groundtruth}})$, the prediction

$$y^{MBR} = \arg \min_y s(y, x) \quad (4.3)$$

is optimal.

Note that when dealing with the 0-1 class loss, this prediction is equivalent to Maximum Score. Yet, we expect this predictor to perform better in the hierarchical evaluation case, as we truly optimize for the $\Delta$-error.

For Top $K$ prediction, the first naive choice that comes to mind would be to use MS or MBR and take the $K$ extrema of the score function. This approach is optimal in the MS case, but not for MBR prediction. With hierarchical evaluation, we are interested in having a broader coverage of the possible class, and aim for diversity. Taking the $K$ minima of the MBR score usually leads to taking several classes from the same branch of the tree. We therefore adapt our MBR score to the top-$K$ case.

When predicting several labels, the prediction that matters is the one that is the closest (in the $\Delta$ sense) to the ground-truth. We therefore define the $\Delta$-error for a prediction $y$ as

$$\Delta(y, y) := \min_{y' \in Y} \Delta(y', y). \quad (4.4)$$

The Bayes Risk score of a multiple prediction $y = \{y_1, ..., y_K\}$ is therefore

$$s(y, x) = \sum_{y' = 1}^{C} \Delta(y, y') p(y'|x). \quad (4.5)$$

And as before,

$$y^{MBR} = \arg \min_y s(y, x). \quad (4.6)$$
4.1 Brute-Force Method

The difficulty in this problem is that even for small $K$, the size of the set of possible outputs becomes extremely large: $\binom{C}{K}$. The brute-force algorithm solving for this problem is given in Alg. 1.

**Algorithm 1** Brute-Force MBR prediction

```plaintext
function PredictBruteForce(x, p, Δ)
    y := (1, 2, $\cdots$, K)
    y∗ := y, s∗ = ∞
    while $y[1] \leq C - K$ do
        $s = \sum_{y=1}^{K} \Delta(y, y)p(y|x)$
        if $s < s^*$ then
            $s = s^*$
            $y^* := y$
        end if
        $i := 0$
        while $y[K - i] = C - i$ do
            $i += 1$
        end while
        $y[K - i] += 1$
        for $j = K - i$ to $K$ do
            $y[j] = y[K - i] + j - K + i$
        end for
    end while
    return $y^*$
end function
```

The $\binom{C}{K}$ possible combinations are generated, resulting in an exponential complexity.

4.2 Greedy algorithm

To make the problem tractable, we resort to an approximation. We start by noticing that minimizing the objective function $\sum_{y=1}^{C} \Delta(y, y)p(y|x)$ is equivalent to maximizing over

$$s(y, x) = \sum_{y=1}^{C} \Delta(y, y)p(y|x),$$  \hspace{1cm} (4.7)$$

where

$$\Delta(y, y) = \max_{(y', y'')} \Delta(y', y'') - \Delta(y, y).$$  \hspace{1cm} (4.8)$$

We call $\Delta$ the class correlation; it measures the semantic closeness of two classes.

Having rewritten the objective in this form, we observe that it is submodular [25].

**Definition 1.** A set function $f : A \rightarrow B$ is called submodular if for all $E \subset F \subset A$ and $x \in A \setminus F$,

$$f(E \cup \{x\}) - f(E) \geq f(F \cup \{x\}) - f(F).$$  \hspace{1cm} (4.9)$$
Theorem 1. The objective $s(y, x)$ is submodular in $y$.

Proof. We first show that
$$\Delta(y, y') = \arg \max_{y' \in y} \Delta(y, y')$$
(4.10)
is submodular in $y$.

Let $y_1 \subset y_2$ be two sets of labels. Obviously, $\Delta(y_1, x) \leq \Delta(y_2, x)$. Let $y$ be a new label. $y$ can be a new maximizer of $\Delta(y_1 \cup \{y\})$, in which case it is also a new maximizer of $\Delta(y_2 \cup \{y\})$ and $\Delta(y_1 \cup \{y\}) = \Delta(y_2 \cup \{y\})$, which proves the inequality (4.9) with the previous point. Alternately, if $y$ does not maximize $\Delta(y_1 \cup \{y\})$, then it also does not maximize $\Delta(y_2 \cup \{y\})$, and both quantities are equal.

It is easy to see that the set of submodular functions over a given set is closed under positive linear combination. Our objective $s(y, x)$ is therefore submodular in $y$.

The submodularity of the objective function allows us to use the greedy algorithm (Alg. 2) which has a complexity of $KC^2$, and to lower-bound the error it makes, as in Theorem 2.

Algorithm 2 Greedy MBR prediction

<table>
<thead>
<tr>
<th>function PREDICTGREEDY(x, p, $\Delta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = \emptyset$</td>
</tr>
<tr>
<td>for $i = 1, \ldots, K$ do</td>
</tr>
<tr>
<td>$y = \arg \max_{y' \in y} \sum_{y' \in y'} \Delta(y \cup {y}, y')p(y', x)$</td>
</tr>
<tr>
<td>$y = y \cup {y}$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>return $y$</td>
</tr>
<tr>
<td>end function</td>
</tr>
</tbody>
</table>

Theorem 2. Let $y^{\text{GREEDY}}$ be the output of the greedy algorithm and $y^{\text{EXACT}}$ the one of the brute-force algorithm. The following approximation holds:
$$s(y^{\text{GREEDY}}, x) \geq (1 - 1/e)s(y^{\text{EXACT}}, x)$$
(4.11)

Proof. For the sake of simplicity, we denote just for this proof $y := y^{\text{GREEDY}}$, $y^* := y^{\text{EXACT}}$, and $s(y) := s(y, x)$.

Let $y = \{y_1, \ldots, y_K\}$, and $y_i = \{y_1, \ldots, y_i\}$. We start by noticing that because $\Delta$ is non-negative, $s(y^*) \leq s(y^* \cup y_i)$. Using a telescoping sum, we rewrite the latter as:
$$s(y^* \cup y_i) = s(y_i) + \sum_{j=1}^{K} \left( s(y_i \cup \{y_1, \ldots, y_j\}) - s(y_i \cup \{y_1, \ldots, y_{j-1}\}) \right).$$
(4.12)

Because $s$ is submodular (Theorem 1),
$$s(y_i \cup \{y_1, \ldots, y_j\}) - s(y_i \cup \{y_1, \ldots, y_{j-1}\}) \leq s(y_i \cup \{y_j^*\}) - s(y_i)$$
(4.13)

Because the greedy algorithm maximizes at each step the next label, we have the following simplification:
$$s(y_i \cup \{y_j^*\}) - s(y_i) \leq s(y_{i+1}) - s(y_i)$$
(4.14)
Putting everything back together, we get

\[ s(y^*) - s(y_i) \leq K(s(y_{i+1}) - s(y_i)) \quad (4.15) \]

Defining \( \delta_i := s(y^*) - s(y_i) \), we have

\[ \delta_i \leq K(\delta_i - \delta_{i+1}) \quad (4.16) \]

and therefore

\[ \delta_i \leq (1 - 1/K) \delta_{i-1} \quad (4.17) \]

We can now see that

\[ \delta_K \leq (1 - 1/K)^K \delta_0 \leq (1 - 1/K)^K s(y^*) \quad (4.18) \]

and therefore, as \((1 - 1/K)^K \leq e\), rearranging terms proves Theorem 2.

The original proof is due to Nemhauser et al. [25], we merely adapted it to our settings. Fig 4.1 shows that in practice, the bound is tighter than the theoretical one.

![Figure 4.1: Different minimum Bayes risk prediction approaches for top-K classification. Naive is the standard approach and simply predicts the classes with the K smallest risk. Optimal considers the dependence of the different predictions and "diversifies" the top-K list, but is computationally expensive. Greedy is the computationally efficient approximation to the optimal strategy (with theoretical suboptimality guarantees). Data obtained using the 10 first images of the 51 classes of our Athletic dataset.](image)

### 4.3 Lazy Algorithm

The complexity of the greedy algorithm grows with \( KC^2 \). It is however possible to improve the running time, using an algorithm due to Minoux [24] and detailed in Alg. 3. The idea is to maintain an ordered list of the marginal improvements, and to check for updates in the given
Algorithm 3 Lazy MBR Prediction

```
function LazyPredict(x, p, Δ)
    Priority heap $H$
    Initialize $H$ with $s(y, x)$, for $y \in \{1, \ldots, C\}$
    Initialize $y$ with $\{\arg\max s(y, x)\}$
    for $k = 1, \ldots, K$ do
        while true do
            Pop from $H$ element $y$ with highest gain
            Evaluate gain $s(y \cup \{y\}, x) - s(y, x)$
            if gain stays highest then
                $y = y \cup \{y\}$
                break
            else
                Reinsert gain in $H$
            end if
        end while
    end for
    return $y$
end function
```

order. The higher $K$ is, the fewer function evaluation are necessary, compared to the greedy algorithm.

**Theorem 3.** The lazy greedy implementation returns the same prediction as the plain greedy algorithm.

**Proof.** We prove by induction that every step of the lazy algorithm picks the same label as its counterpart. Initialization is covered by the fact that both algorithms have the same first step. If $y$ is picked, it means that $s(y \cup \{y\}, x) - s(y, x)$ is higher than all previous gains. Yet, because $s$ is submodular, a gain with a subset of $y$ is always higher than with a full $y$. We can therefore be sure that $y$ maximize $s(y \cup \{y'\}, x)$ for all $y'$.

A comparison of both greedy and lazy algorithms can be found in Fig. 4.2.

### 4.4 Experiments

We then proceed to run evaluation on both Athletic and ILSVRC datasets. We use the ILSVRC class loss for the latter, and all three possible for the former. Scores are reported on the test set, after determining $\lambda$ with the two validation methods we previously described. Standard deviation is obtained by using five different train-test partitions.

As we can see, MBR performs better with hierarchical loss, but worse with flat 0-1 loss.
Figure 4.2: Speed of greedy (Alg. 2) and lazy (Alg. 3).

<table>
<thead>
<tr>
<th>Athletic</th>
<th>Hierarchy</th>
<th>0-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 2</td>
</tr>
<tr>
<td>MS prediction</td>
<td>2.07 ± 0.015</td>
<td>1.36 ± 0.008</td>
</tr>
<tr>
<td>MBR prediction</td>
<td>2.02 ± 0.008</td>
<td>1.30 ± 0.011</td>
</tr>
</tbody>
</table>

Table 4.1: Results on the Athletic dataset with the Node distance class loss. The second table is given to provide a more concrete evaluation, as well as to show that MBR non-hierarchical performances are close to MS ones with respect to this measure. MS prediction is expected to perform better.

<table>
<thead>
<tr>
<th>Athletic</th>
<th>Hierarchy</th>
<th>0-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 2</td>
</tr>
<tr>
<td>MS prediction</td>
<td>18.29 ± 0.12</td>
<td>11.61 ± 0.08</td>
</tr>
<tr>
<td>MBR prediction</td>
<td>17.71 ± 0.18</td>
<td>10.36 ± 0.11</td>
</tr>
</tbody>
</table>

Table 4.2: Results on the Athletic dataset with the ranking class loss.
Table 4.3: Results on the Athletic dataset with the ILSVRC class loss.

<table>
<thead>
<tr>
<th>Athletic</th>
<th>Hierarchy</th>
<th>0-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 2</td>
</tr>
<tr>
<td>MS prediction</td>
<td>1.02 ± 0.0018</td>
<td>0.63 ± 0.0013</td>
</tr>
<tr>
<td>MBR prediction</td>
<td>0.75 ± 0.0014</td>
<td>0.50 ± 0.0020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Athletic</th>
<th>0-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
</tr>
<tr>
<td>MS prediction</td>
<td>0.54 ± 0.004</td>
</tr>
<tr>
<td>MBR prediction</td>
<td>0.58 ± 0.004</td>
</tr>
</tbody>
</table>

Table 4.4: Results on the ILSVRC 2010 dataset. Our descriptors are far from being comparable to the ones used in the competition, yet we see that for comparable flat results, we significantly outperform the hierarchical loss.

<table>
<thead>
<tr>
<th>ILSVRC</th>
<th>Hierarchy</th>
<th>0-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 2</td>
</tr>
<tr>
<td>Log-Loss</td>
<td>7.21</td>
<td>5.81</td>
</tr>
<tr>
<td>MBR prediction</td>
<td>6.97</td>
<td>5.45</td>
</tr>
</tbody>
</table>

22
Chapter 5

Bayes-Risk Training

The framework for the prediction being laid out, the question of getting the posterior probabilities can now be considered. The obvious choice would be to go for the standard logistic regression, but we also propose a new algorithm, called the Bayes Risk Machine, that aims to optimize for our new prediction.

5.1 Top-1 Bayes Risk Machine

Let us consider at first the case where we predict a single input. The standard Structured Support Vector Machine theory \[9\] (SSVM), relies on a linear score function

\[
s_{sv}^w(y, x) = \langle w, \phi(x, y) \rangle, \tag{5.1}\]

where \(\phi\) is a mapping of \(\mathbb{R}^d \times \{1, \ldots, K\}\) in \(\mathbb{R}^{Kd}\).

The prediction chooses the label that maximizes \(s_{sv}^w(y, x)\), and the training algorithm therefore trains \(w\) to make it discriminate classes with respect to this score. In the SVM framework \([3, 8]\), this means maximizing the margin, ie the score difference between right and the wrong choices. When having a class loss function, the “wrongness” of a choice can be quantified, and we aim to minimize

\[
\frac{1}{N} \sum_{(x, y) \in Z} -s_{sv}^w(y, x) + \max_{y'} [s_{sv}^w(y', x) + \Delta(y', y)] \tag{5.2}
\]

As in the logistic loss, a regularization term, usually equal to \(\frac{1}{2} \|w\|^2\) is added.

\[
f_{SSVM}(Z|w) = \frac{1}{N} \sum_{x, y \in Z} -(w, \phi(x, y)) + \max_{y'} [(w, \phi(x, y')) + \Delta(y', y)] + \frac{1}{2} \lambda \|w\|^2 \tag{5.3}
\]

The first term in equation \(5.2\) is the response of the score to the ground-truth, the second is the prediction, penalized by the class loss. Minimizing \(5.2\) forces \(w\) to return a high score to matching features and labels; the class loss term ensures that a class that is poorly correlated with the ground-truth does not output a high score.

When predicting using the Bayes Risk, we can actually replace the support vector score by the Bayes score. When using the logistic loss, our probabilities are

\[
p(y|x, w) = \frac{1}{Z(x, w)} \exp(w, \phi(x, y)), \tag{5.4}\]
where $Z$ is the partition function, used to normalize everything to one.

We modify slightly the previous Bayes Risk score by dropping $Z$ and applying a logarithm. Both these operations do not change the actual maximum and using a logarithm helps with numerical issues.

$$s_{w}^{\text{BR}}(y, x) := \log \left( \sum_{y' = 1}^{C} \bar{\Delta}(y, y') \exp\{w, \phi(x, y')\} \right)$$  \hspace{1cm} (5.5)

As in Eq. 5.2 our objective is:

$$\sum_{(x, y) \in Z} -s_{w}^{\text{BR}}(y, x) + \max_{y'} [s_{w}^{\text{BR}}(y', x) + \Delta(y', y)]$$  \hspace{1cm} (5.6)

The loss can in turn be explicitly decomposed in

$$\ell^{\text{BRM1}}(D|w) = -\frac{1}{N} \sum_{(x, y) \in Z} \log \sum_{y'} \bar{\Delta}(y, y') \exp\{w, \phi(x, y')\} +$$

$$\frac{1}{N} \sum_{(x, y) \in Z} \max_{y''} \left[ \log \sum_{y'} \bar{\Delta}(y'', y') \exp\{w, \phi(x, y')\} + \Delta(y'', y) \right]$$

$$+ \frac{\lambda}{2} \|w\|^2. \hspace{1cm} (5.7)$$

### 5.2 Top-K Bayes Risk Machine

We can also apply our learning formulation to the top-$K$ case. However we lack the proper ground-truth, as we only know the true label of an example. The trick is here to treat the $K - 1$ remaining unknown labels as latent variables [34].

As in (5.5), the multiple output Bayes Risk score is modified to

$$s_{w}^{\text{BR}}(y, x) := \log \left( \sum_{y' = 1}^{C} \bar{\Delta}(y, y') \exp\{w, \phi(x, y')\} \right)$$  \hspace{1cm} (5.8)

The new loss becomes

$$\ell^{\text{BRM1}}(D|w) = -\frac{1}{N} \sum_{(x, y) \in Z} \max_{y'|y'(0) = y} \log \sum_{y'} \bar{\Delta}(y'', y') \exp\{w, \phi(x, y')\} +$$

$$\frac{1}{N} \sum_{(x, y) \in Z} \max_{y''} \left[ \log \sum_{y'} \bar{\Delta}(y'', y') \exp\{w, \phi(x, y')\} + \Delta(y'', y) \right]$$

$$+ \frac{\lambda}{2} \|w\|^2. \hspace{1cm} (5.9)$$

### 5.3 Resolution

The main drawback from switching to the Bayes Risk score is the loss of linearity, which in turns implies the surrogate loss is no longer convex. The easiest way to solve this problem is to notice
that in both equations (5.7) and (5.9), the first sum term is concave, whereas the second and third are convex. This setting calls, as in the latent-SSVM case, for a Concave-Convex procedure [35]. This optimization algorithm assumes the objective \( f \) can be decomposed as \( f = g + h \), where \( g \) is concave and \( h \) is convex. It iterates between linearizing the concave part, and solving for the new objective, which becomes convex. Alg. 4 describes the procedure.

**Algorithm 4 Concave-Convex Procedure**

```plaintext
function CCCP(g, h, x₀)
    for n = 0, ... do
        \( g'_n = (\nabla g)_{x_n} \)
        \( x_{n+1} = \arg \inf_x [h(x) + \langle g'_n, x \rangle] \)
    end for
end function
```

Because we deal with a non-convex objective, we can only guarantee a local minimum. However, theorem 4 guarantees the output improves on the initial guess. Because Logistic Regression already gives a good estimate, the output of our algorithm should give good results.

**Theorem 4.** The output of Alg. 4 improves on the initial guess \( x₀ \).

**Proof.** At step \( n \), rewriting the linearized objective as

\[
    f_L(x) = h(x) + g(x_n) + \langle g'_n, (x - x₀) \rangle
\]

does not change \( x_{n+1} \). Because \( g \) is concave, The true objective \( h(x) + g(x) \) is upper-bounded by \( f_L \). Because\( f_L(x_n) = f(x) \), we can be sure that for all \( n \), \( f(x_{n+1}) \leq f(x_n) \).

To solve the linearized objective, a theoretically correct solver would be a subgradient descent [31], as we deal with a piecewise smooth function. Yet, LBFGS [4, 29], although not guaranteed to end up on a minimum, offered a better convergence rate in practice, and we therefore chose it.

### 5.4 Experiments

We start by running some experiments on a synthetic dataset. We randomly sample Gaussian classes with various means and variances. We apply a slight non-linear transform to prevent logistic regression to be too competitive. A simple example with 5 classes and two dimensions is given on Fig. 5.1. We investigate the difference between a totally random class loss (although still observing \( \Delta(y, y) = 0 \) for all \( y \)) and one that makes relative sense: the simple distance between class centroids.

We summarize our synthetic experiments in Table 5.1. We report the same prediction schemes: MBR and MS evaluated with hierarchical loss, as well as, for reference, the misclassification rate. As we can see, our algorithm is able to improve significantly on the LogLoss baseline when the class loss makes sense (distance between centroids).

On Athletic, we were able to improve results with all the proposed class losses.

#### 5.4.1 Top \( K \)

Unfortunately, none of the good results obtained with MBR prediction and top 1 Bayes Risk Machines were reproduced when using BRM top \( K \). Training error with \( \lambda = 0 \) even failed to
overfit, and results stayed higher or equal to those obtained by LogLoss. As of now, we do not know if the optimization algorithm was not powerful enough, or if the data simply did not fit the model. We plan on investigating this in future work.
Table 5.1: First array shows results on several synthetic datasets, described in the second. Values in themselves and across settings are not meaningful, what really matters is to see which, between LogLoss and BRM has the higher MBR hier score.

<table>
<thead>
<tr>
<th>Settings</th>
<th>MBR hier</th>
<th>MS hier</th>
<th>Error (%)</th>
<th>MBR hier</th>
<th>MS hier</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>4.66</td>
<td>4.81</td>
<td>25.6</td>
<td>4.09</td>
<td>4.20</td>
<td>23.9</td>
</tr>
<tr>
<td>(2)</td>
<td>8.69</td>
<td>9.27</td>
<td>56.8</td>
<td>7.91</td>
<td>8.57</td>
<td>8.57</td>
</tr>
<tr>
<td>(3)</td>
<td>303</td>
<td>305</td>
<td>50.4</td>
<td>257</td>
<td>268</td>
<td>44.6</td>
</tr>
<tr>
<td>(4)</td>
<td>409</td>
<td>415</td>
<td>72.4</td>
<td>361</td>
<td>389</td>
<td>66.6</td>
</tr>
<tr>
<td>(5)</td>
<td>0.94</td>
<td>1.10</td>
<td>25.6</td>
<td>1.04</td>
<td>0.87</td>
<td>33.0</td>
</tr>
<tr>
<td>(6)</td>
<td>2.31</td>
<td>2.83</td>
<td>56.8</td>
<td>2.51</td>
<td>2.36</td>
<td>66.4</td>
</tr>
<tr>
<td>(7)</td>
<td>1.94</td>
<td>2.40</td>
<td>50.4</td>
<td>1.78</td>
<td>1.46</td>
<td>55.5</td>
</tr>
<tr>
<td>(8)</td>
<td>3.12</td>
<td>3.66</td>
<td>72.4</td>
<td>3.11</td>
<td>2.64</td>
<td>75.5</td>
</tr>
</tbody>
</table>

Table 5.2: Results showing good performance of the Top-1 Bayes Risk Machine on Athletic, with our three proposed losses.

<table>
<thead>
<tr>
<th>Delta-Loss</th>
<th>MS</th>
<th>MBR</th>
<th>BRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>2.06</td>
<td>2.01</td>
<td>1.92</td>
</tr>
<tr>
<td>Ranking</td>
<td>18.29</td>
<td>17.71</td>
<td>17.37</td>
</tr>
<tr>
<td>ILSVRC</td>
<td>1.02</td>
<td>0.75</td>
<td>0.72</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusion

During this master’s thesis, we have studied image classification through the popular ImageNet dataset, focusing on a part of its related challenge (ILSVRC), which received little attention: the hierarchical top-$K$ classification. Using logistic regression to compute an estimate of the posterior class probabilities, we derive an theoretically optimal prediction called Minimum Bayes Risk. For $K > 1$, we use a greedy approximation and a fast implementation to tackle the computational difficulty. Results were shown to significantly improve over the usual Maximum Score prediction scheme.

In a second part, we focused on improving our initial probabilities estimates, by designing a new classifier: the Bayes Risk Machine, to work conjointly with our prediction. Using a loss based on the structured SVM one, we cast the problem into an unconstrained optimization of a piecewise-smooth non-convex objective, which we solve efficiently using the Concave-Convex procedure. Results were promising for $K = 1$, but failed to improve for larger $K$.

With more time, we would have focused on improving the resolution of our Top-$K$ MBR optimization problem, in particular to be able to run with increased speed on larger datasets (computing on 1000 classes took around a week). Also, not being able to find a better minimum with subgradient descent does not necessarily mean that the solution outputed by LBFGS was optimal. It may be worthwhile to investigate this issue in order to get the results we longed for. It may also be possible that our features, or possibly our initial problem does not fit with our BRM model. Working on documents or other data could also have helped us understand the underlying problem.
Chapter 7

Acknowledgments

The end of my master thesis at ETH is near, and I would like to warmly thank my advisers, Patrick Pletscher for his patience over my mistakes and delays, and David Balduzzi for having taken over in the middle of a work right when it was stalling and refusing to be tamed. I would also like to thank my roommates for being supportive and listening to my confuse explanations about my work. As time comes for me to return to my motherland, my thoughts will be with those who desperately tried to teach me Schwizertütsch, I still have a few weeks to become fluent.
Bibliography


