# Minimization of calibrated loss functions for image classification

Wafa Bel Haj Ali

UNIVERSITY OF NICE - SOPHIA ANTIPOLIS
DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

# P H D T H E S I S

to obtain the title of

**PhD of Science**

of the University of Nice - Sophia Antipolis
**Speciality : Computer Science**

Defended by

Wafa Bel haj ali

# Minimisation de Fonctions de Perte Calibree pour la Classification des Images

Thesis Advisor: Michel Barlaud

prepared at I3S Sophia Antipolis, MediaCoding Team

defended on October 11, 2013

**Jury :**

| | | |
|---|---|---|
| *Reviewers :* | Florent Perronnin | Research Manager XRCE, Grenoble |
| | Patrick Perez | DR INRIA-Technicolor, Rennes |
| *Advisor :* | Michel Barlaud | PR Emerite UNS, Sophia Antipolis |
| *Examinators :* | Sherif Makram-Ebeid | Research Fellow Philips Research, Paris |
| | Frédéric Precioso | PR UNS, Sophia Antipolis |
| | Cordelia Schmidt | DR INRIA, Grenoble |
| *Invited :* | Thierry Pourcher | CEA / UNS / CAL, Nice |

*This dissertion is dedicated to my mother, to the memory of my father, because of their faith and pride, because of all the wonderful things they did for me...*

A special dedication to my lovely husband who always supports me and to my cute son who gives me the joy. A special feeling to my brothers and "sisters", who always encourage me, and to my mother-in-law and my father-in-law for their love. Finally, a special thank to all my friends, whith whom I spent those years and shared wonderful things. I do not want to list them, because the list could be long and I am afraid to forget someone.

# Acknowledgments

First of all, I would like to thank Professor Michel Barlaud for his support, his advises, his presence and above all his faith. Working with him during three years of thesis was a great honor.

I would adress a great thank to Professor Richard Nock and Frank Nielsen for their collaboration and their precious contributions. I wish also to thank and acknowledge the contribution of the team of biologists Tiro and especially Professor Thierry Pourcher and Philippe Pognonec. I do not forget Eric Debreuve, who helped me for long times and gave me precious advises.

Finally, I thank every one who supported me and helped me during those years: a thought to all the administrative persons who are making our needs easier.

# Contents

Chapter 1

# General introduction

## 1.1 Introduction

The classification task consists in predicting category membership of an unlabeled data based on its content. Classifying images is a challenging task in computer vision, since it involves different fields and applications. In fact, two main fields are being studied to perform image classification and pattern recognition: the first, which belongs to the image processing field, deals with extracting the features from data. A way to encode images with less complex structures that best describes the information contained in the image. While the second one is a machine learning task defining the classification rule.

In computer vision tasks, image features are usually considered either as local or as global descriptors. Both of them have been shown to be efficient. Gist global feature [Oliva & Torralba 2001, Oliva & Torralba 2006] for example represents a whole scene in a unique descriptor, while the scale invariant feature transform (SIFT) [Lowe 2004] or the histogram of oriented gradients (HOG) [Dalal & Triggs 2005] represent local information in the image allowing the description of significant objects in the scene independently. Local features are relevant for image description. In computer vision, they are well adapted for objects detection and image retrieval: they give a sparse representation and cover a wide range of visual features in the image. However, for classification task, we almost need global feature description, since we compare categories and not only pairs of images. Hence, we usually encode local features into global ones using statistical models. This global representation describes the occurrence of relevant visual features in the image. State of the art Bag of features/words (BoF/BoW) [Sivic & Zisserman 2006] are the most common approaches in this context. Recently an efficient feature called fisher vectors (FV) [Perronnin et al. 2010] was extensively used for large scale image classification.

Getting efficient descriptors is not sufficient to perform categorization. Robust classification algorithms should be designed to accomplish such challenging task. For most state of the art methods, the task of image classification is addressed as a learning problem. Within this context, we distinguish two major approaches, depending on wether we have or have not a knowledge about the categories and about the labels of a set of data. On the one hand, unsupervised approaches, like clustering, tend to group data according to their visual content similarities. On the other hand, supervised learning uses an already labeled training set to learn classifiers (categories boundaries) and then labels non-annotated images subsequently. For the second kind of learning, three or four main standard methods are often used.

The kernel based algorithms and more precisely the Support Vector Machine (SVM) [Cristianini & Shawe-Taylor 2000] are robust classification methods. The boosting based algorithms such as Adaboost [Freund & Schapire 1999] are scalable, have low computational complexity and still reliable. Nearest Neighbors approaches are fast, simple and scalable, but still poorly effecient in accuracy. Recently, a stochastic gradient descent (SGD) algorithm was introduced by [Bottou 2010], a robust and non complex method for large scale data.

To state a supervised classification problem, we need to define our classifier first. In fact, the classification rule is a function mapping between the data features and their predicted labels. Among state of the art classifiers, we can cite k-Nearest Neighbors, linear or kernel based classifiers. However, despite the nature of a classification rule, it is often defined by a set of parameters. Therefore, we set a learning process to reach the optimal rule. Indeed, given a set of already annotated data, we tend to estimate the optimal parameters by minimizing the classification error rate.

This thesis deals with supervised learning approaches for image classification. Especially, we are interested in the minimization of a criterion based on some specific loss functions (*Calibrated losses*) for different kind of classification rules. In a first part, we are interested in k-NN classifiers. A first approach, revisits and expands a leveraged k-NN rule by minimizing the risk criterion in a boosting framework. In the same context, a second approach deals with fast convergence Newton based leveraged Nearest Neighbors rule. In a second part, we design a fast low rank Newton descent algorithm of criterion minimization for learning scalable linear classifiers. This latter is a robust algorithm especially for big datasets and shows high computational performance and precision towards state of the art approaches. In a final part, this thesis presents an application of image categorization to an interesting field: bio-medial imaging. In a first step, we design a specific descriptor for such application: a multiscale contrast based feature, well adapted for cell images. Then, we report examples of experiments on two different applications of biological cells classification.

## 1.2 Setting the problem

We first provide some generalities that define our supervised learning scheme. Our setting is that of multiclass, multilabel classification. In supervised learning, we have access to an *annotated* input set of $m$ *observations*, $\mathcal{S} \doteq \{\boldsymbol{o}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{x}_i \in \mathbb{X}$ is a feature data where $\mathbb{X}$ denotes the feature space. We adopt the mainstream one-vs-all classification scheme. Then, vector $\boldsymbol{y}_i \in \{-1, +1\}^C$ encodes class memberships, assuming $y_{ic} = +1$ means that sample $\boldsymbol{x}_i$ belongs to class $c$ and $y_{ic} = -1$ otherwise.

The goal is to learn a classifier $H$ which is a function mapping observations in $\mathbb{X}$ to vectors in $\mathbb{R}^C$. Given some sample $\boldsymbol{x}$, the sign of coordinate $c$ in $H(\boldsymbol{x})$ ($H_c(\boldsymbol{x})$) gives whether $H$ predicts that $\boldsymbol{x}$ belongs to class $c$, while its absolute value may be viewed as a confidence in classification (or score).

To define the classifier $H$, we will minimize the *Empirical (or Hamming) Risk* $\varepsilon^{0/1}(H, \mathcal{S})$ which computes over classes and observations the missclassification rate of $H$ :

$$\varepsilon^{0/1}(H, \mathcal{S}) \doteq \frac{1}{C}\sum_{c=1}^{C}\frac{1}{m}\sum_{i=1}^{m}[(y_{ic}H_c(\boldsymbol{x}_i)) < 0] \ , \tag{1.1}$$

where [.] is the indicator function equal to 1 if the condition is true and 0 otherwise and which represents here the 0/1 or empirical loss. We denote this loss $F^{0/1}$. Unfortunatly, the minimization of such problem is not tractable since the 0/1 loss function is not convex.

A common alternative to minimize (1.1) is to rather minimize an upperbound of this empirical risk, known as the *Surrogate Risk*. Lets denote this later $\varepsilon_F$. This *surrogate* sums over observations and classes a strictly convex loss function $F : \mathbb{R} \to \mathbb{R}$ that satisfies $\forall x \in \mathbb{R}, F^{0/1}(x) \leq F(x)$.

$$\varepsilon_F(H, \mathcal{S}) \doteq \frac{1}{C}\sum_{c=1}^{C}\frac{1}{m}\sum_{i=1}^{m}F(y_{ic}H_c(\boldsymbol{x}_i)) \ . \tag{1.2}$$

The loss function $F$ is based on the functional margin $y_{ic}H_c(\boldsymbol{x}_i)$ or what we call *the edge of classification* and denote by $\rho(H_c, o_{i,c})$. Obviously, the minimization of (1.2) leads to a close form solution of the initial problem (1.1).

The consistency of classification rules is crucial properties without which the minimization of the loss brings no strong statistical guarantee: the risk of classification should get close to the lowest possible risk with a large probability (Bayes rule). To satisfy this property, a set of loss functions relevant for learning is often used and called Calibrated Losses [Bartlett et al. 2006].

# Part I

# Learning weighted k-NN Classifiers with Calibrated Losses

Chapter 2

# Universal Nearest Neighbors algorithm: UNN

## 2.1 Introduction

The nearest neighbors (NNs) rule belongs to the oldest, simplest and still most widely studied classification algorithms [Devroye et al. 1996]. It relies on a non-negative real-valued "distance" function. This function measures how much two observations differ from each other, and may not necessarily satisfy the requirements of metrics.

k-NN classification has proven successful, thanks to its easy implementation and its good generalization properties [Shakhnarovich et al. 2006]. A major advantage of the k-NN rule is to not require explicit construction of the feature space and be naturally adapted to multi-class problems. Moreover, from the theoretical point of view, straightforward bounds are known for the true risk (error) of k-NN classification with respect to Bayes optimum, even for finite samples ([Nock & Sebban 2001]). In fact, it is yet a challenge to reduce the true risk of the k-NN rule, usually tackled by data reduction techniques [Hart 1968].

We propose in this chapter an optimization of a generalized solution to the problem of boosting k-NN classifiers in the general multi-class setting, and for general classes of losses, not restricted to Adaboost's exponential loss, built upon the works of [Piro et al. 2012, Nock & Nielsen 2009, Nock & Nielsen 2008]. Namely, we propose a *leveraged* nearest neighbor rule that generalizes the uniform k-NN rule, and whose convergence rate is guaranteed for many *classification calibrated* losses, encompassing popular choices, such as the logistic loss or the matsushita loss. The voting rule is redefined as a strong classifier that linearly combines weak classifiers of the k-NN rule.

The remaining of the chapter is organized as follows: Section 2.2 brievly introduces the basic notions about k-NN classifiers and about the calibrated loss functions used latter in the learning framework. Section 2.3 presents the Universal Nearest Neighbors algorithm for leveraging the k-NN classifier and Section 2.4 gives details about the optimizations brought on this algorithm and the implementation of the method. Finally, Section 2.5 shows experimental results of our method against standard/uniform k-NN and SVM methods on large images datasets.

| crit | calibrated loss F | annotation |
|---|---|---|
| A | $\exp(-x)$ | $exp$ |
| B | $\ln(1+exp(-x))$ | $log$ |
| C | $-x+\sqrt{1+x^2}$ | $mat$ |

Table 2.1: The strictly convex losses that are used in UNN. From top to bottom, losses are exponential, logistic and matsushita's loss.

## 2.2 Basic notions and annotations

### 2.2.1 The k-NN classifier

We let $j \rightarrow_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, or simply example $j$, belongs to the $k$ NNs of observation $\boldsymbol{x}$. We shall abbreviate $j \rightarrow_k \boldsymbol{x}_i$ by $j \rightarrow_k i$ — in this case, we say that example $i$ belongs to the *inverse neighborhood* of example $j$. To classify an observation $\boldsymbol{x}$, the $k$-NN rule $H(\boldsymbol{x})$ computes the sum of class vectors of its nearest neighbors. The coordinate $c$ in $H(\boldsymbol{x})$ is :

$$H_c(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} y_{jc} \ . \tag{2.1}$$

### 2.2.2 Calibrated losses

Classification calibrated losses are surrogates suitable for classification. To be classification-calibrated, loss $F : \mathbb{R} \rightarrow \mathbb{R}$ is required to be convex, differentiable and such that $F'(0) < 0$ [Bartlett et al. 2006] (Theorem 4), [Vernet et al. 2011].

In this chapter, we are interested in a subset of the calibrated losses called *Strictly Convex Losses* (SCL). This set includes, in addition to the exponential loss, the logistic, the matsushita and the squared loss. The strictly convex losses F we are intrested in are given in Table 2.1.

## 2.3 UNN, Leveraging the k-NN classifier

As previously introduced, a leveraged k-NN rule is a non-uniform voting among the k-Nearest Neighbors defined like below:

$$H_c(\boldsymbol{x}_i) \doteq \sum_{j \rightarrow_k i} \alpha_{jc} y_{jc} \ . \tag{2.2}$$

The classifier $H_c$ is defined as a sum among a set of $T$ weak classifiers. We call those laters *prototypes*. So, given a set $\mathcal{S} \doteq \{\boldsymbol{o}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$, one prototype, denoted by the index $j$, is a training sample $\in \mathcal{S}$ defined by its feature vector $\boldsymbol{x}_j$, label $y_{jc}$ and later by its leveraging weight $\alpha_{jc}$. Those weights are determined by fitting the classifier $H_c$ into the supervised learning scheme previously described in (1.2).

### 2.3.1 Learning leveraged k-NNs in a boosting framework

Voting weights $\alpha_{jc}$ in (2.2) are solutions of the minimization of the following average surrogate risk:

$$\varepsilon_F(H, \mathcal{S}) \doteq \frac{1}{C}\sum_{c=1}^{C} \underbrace{\frac{1}{m}\sum_{i=1}^{m} F(y_{ic}H_c(\boldsymbol{x}_i))}_{\varepsilon_F(H_c,\mathcal{S})} . \tag{2.3}$$

Since we are in the one-vs-all learning scheme we can minimize the per-class risk $\varepsilon_F(H_c, \mathcal{S})$ corresponding to $H_c$. To to so, one alternative is to use a boosting like approach and then minimize each surrogate $\varepsilon_F(H_c, \mathcal{S})$ iteratively. In fact, at each iteration we pick one prototype $j \in \mathcal{S}$ for which the classification rule is defined as the following *weak* classifier:

$$h_{jc}(\boldsymbol{x}_i) \doteq \alpha_{jc}y_{jc} \quad ; \quad j \rightarrow_k i \tag{2.4}$$

such that:

$$H_c(\boldsymbol{x}_i) \doteq \sum_{j\rightarrow_k i} h_{jc}(\boldsymbol{x}_i) . \tag{2.5}$$

Thus the local risk (of the weak classifier) is the sum of losses due to $h_{jc}$ over the training set $\mathcal{S}$:

$$\varepsilon_F(h_{jc}, \mathcal{S}) \doteq \frac{1}{m}\sum_{i=1}^{m} F(y_{ic}h_{jc}(\boldsymbol{x}_i)) . \tag{2.6}$$

Note that the classifier $h_{jc}$ follows the leveraged k-NN rule and then only a subset of $\mathcal{S}$ for which sample $j$ is a k-NN are concerned by the voting of $j$. We denote this subset by $\mathcal{R}_j \subseteq \mathcal{S}$ which is exactly the set of inverse nearest neighbors of $j$ and which cardinality is equal to $n_j$. Hence we reduce once again the risk function that should be minimized to this following:

$$\varepsilon_F(h_{jc}, \mathcal{R}_j) \doteq \frac{1}{n_j}\sum_{i=1}^{n_j} F(y_{ic}h_{jc}(\boldsymbol{x}_i)) . \tag{2.7}$$

We need to find optimal voting weight that minimizes the risk function in (2.7). To do so, we iteratively update the leveraging weight of the actual weak classifier / prototype $j$ in a boosting like procedure. Hence, we give samples weights of classification denoted by $w_{ic}$ and progressively update them according to the missclassification of $h_{jc}$. That is, weights of badly classified samples should be enhanced and those of well classified ones will be narrowed. We consider the following updating rules for prototypes weights $\alpha_{jc}$, classification rule $h_{jc}$ and for training samples weights $w_{ic}$:

$$\alpha_{jc}^{t} = \alpha_{jc}^{t-1} + \delta_{jc} . \tag{2.8}$$

$$h_{jc}^{t}(\boldsymbol{x}_i) \;=\; h_{jc}^{t-1}(\boldsymbol{x}_i) + \delta_{jc} y_{jc} \;. \tag{2.9}$$

$$w_{ic}^{t} \;=\; -F'(y_{ic} h_{jc}^{t}(\boldsymbol{x}_i)) \tag{2.10}$$

Actually, at each iteration $t$ we should minimize $\varepsilon_F(h_{jc}^t, \mathcal{R}_j)$ according to $\delta_{jc}$. Let us first replace $h_{jc}$ in (2.7) by its expression in (2.9). Then, the risk function becomes

$$\varepsilon_F(h_{jc}^t, \mathcal{R}_j) \;\doteq\; \frac{1}{n_j} \sum_{i=1}^{n_j} F(y_{ic} h_{jc}^{t-1}(\boldsymbol{x}_i) + y_{ic}\delta_{jc} y_{jc}) \;. \tag{2.11}$$

and its first derivative according to $\delta_{jc}$ is expressed like follows:

$$\frac{\partial \varepsilon_F(h_{jc}^t, \mathcal{R}_j)}{\partial \delta_{jc}} \;=\; \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ic} y_{jc} F'(y_{ic} h_{jc}^{t-1}(\boldsymbol{x}_i) + y_{ic}\delta_{jc} y_{jc}) \tag{2.12}$$

$$\;=\; \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ic} y_{jc} F'(F'^{-1}(-w_{ic}^{t-1}) + y_{ic}\delta_{jc} y_{jc}) \tag{2.13}$$

Finally, finding $\delta_{jc} = \arg\min \Big(\varepsilon_F(h_{jc}^t, \mathcal{R}_j)\Big)$ amounts to solving the following general equation based on the surrogate loss F:

$$\sum_{i=1}^{n_j} y_{ic} y_{jc} F'(F'^{-1}(-w_{ic}^{t-1}) + y_{ic}\delta_{jc} y_{jc}) \;=\; 0 \;. \tag{2.14}$$

### 2.3.2 Step by step algorithm

The different steps of UNN are detailed in the general algorithm 1. The step **[I.0]** in the algorithm consists in choosing the prototype $j \in \{1, 2, ..., m\}$ (weak classifier). In fact, at each iteration, the index to leverage $j$, is obtained by a call to a *weak index chooser* oracle WIC(.,.,.). The selection of the index $j$ of the next weak classifier could be done randomly, or using some criterion. In the second case, we pick $T \geq m$, and let $j$ be chosen by WIC($\{1, 2, ..., m\}, t, c$) such that $\delta_j$ is large enough. Each $j$ can be chosen more than once or one can restrict this index to be chosen only once.

The demonstration of the computation of $\delta_j$ solution of (2.15) and $w_i$ in (2.16) will be detailed later. Those expressions are given in Table 2.2 respectively for each of the considered loss in Table 2.1. $W_{jc}^+$ and $W_{jc}^-$, used in Table 2.2, are respectively the sum of weights of positif (good) inverse-NNs and that of negatif (bad) ones:

$$W_{jc}^{+} \;=\; \sum_{i=1}^{n_j} [y_{ic} y_{jc} > 0]\, w_{ic} \;; \tag{2.17}$$

$$W_{jc}^{-} \;=\; \sum_{i=1}^{n_j} [y_{ic} y_{jc} < 0]\, w_{ic} \;; \tag{2.18}$$

**Algorithm 1:** Algorithm Universal Nearest Neighbors UNN($\mathcal{S}$, F)

**Input**: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$, loss F;
**for** $c = 1, 2, ..., C$ **do**
  Let $\alpha_{jc} \leftarrow 0, \quad \forall j$;
  Let $w_i \leftarrow -\mathrm{F}'(0) \in \mathrm{R}^m_{+*}, \quad \forall i$;
  **for** $t = 1, 2, ..., T$ **do**
    **[I.0]** Let $j \leftarrow \text{Wic}(\{1, 2, ..., m\}, t)$;
    **[I.1]** Let $\delta_j \in \mathrm{R}$ solution of:

$$\sum_{i=1}^{n_j} y_{ic} y_{jc} \mathrm{F}'(\mathrm{F}'^{-1}(-w_{ic}) + y_{ic}\delta_{jc} y_{jc}) = 0 \ ; \tag{2.15}$$

    **[I.2]** $\forall i \ : \ j \sim_k i$, let

$$w_i \leftarrow -\mathrm{F}' \left( y_{ic}\delta_{jc} y_{jc} + \mathrm{F}'^{-1}(-w_i) \right) \ ; \tag{2.16}$$

    **[I.3]** Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;

**Output**: $h_c(\boldsymbol{x}) = \sum_{i \sim_k \boldsymbol{x}} \alpha_{ic} y_{ic}, \quad \forall c$ ;

For now, we will give some details about the demonstration getting to the expressions in table 2.2. We will consider first the exponential loss function A in Table 2.1 which is a special case since it leads to a close form solution of $\delta_{jc}$. Then we will explain how to solve the problem for general cases. Lets consider the equation (2.14) corresponding to the exponential risk function, then:

$$\sum_{i=1}^{n_j} y_{ic} y_{jc} (-\exp(-(-\ln(w_{ic}^{t-1}) + y_{ic}\delta_{jc} y_{jc}))) = 0 \tag{2.19}$$

$$\sum_{i=1}^{n_j} y_{ic} y_{jc} \exp(\ln(w_{ic}^{t-1})) \exp(-y_{ic}\delta_{jc} y_{jc}) = 0 \tag{2.20}$$

$$\sum_{i=1}^{n_j} y_{ic} y_{jc} w_{ic}^{t-1} \exp(-y_{ic}\delta_{jc} y_{jc}) = 0 \tag{2.21}$$

$$\sum_{i=1}^{n_j} [y_{ic} y_{jc} > 0] \, w_{ic}^{t-1} \exp(-\delta_{jc}) - \sum_{i=1}^{n_j} [y_{ic} y_{jc} < 0] \, w_{ic}^{t-1} \exp(\delta_{jc}) = 0 \ ; \tag{2.22}$$

In expression (2.22) we split the sum on the inverse-NNs such that we separate the set $\mathcal{R}_j$ into $\mathcal{R}_j^+$ and $\mathcal{R}_j^-$ where $\mathcal{R}_j^+$ denotes the good inverse NNs (i-NN with the same label as $j$) and $\mathcal{R}_j^-$ denotes the bad ones (i-NNs which does not have same label as $j$). Then, using definitions (2.17) and (2.18) we get:

$$W_{jc}^+ \exp(-\delta_{jc}) - W_{jc}^- \exp(\delta_{jc}) = 0 \ ; \tag{2.23}$$

| F | $\delta_{jc}$, see (2.17) and (2.18) | $g : w_i \leftarrow g(w_i)$ |
|---|---|---|
| *exp* | $\frac{1}{2}\ln\frac{W_{jc}^+}{W_{jc}^-}$ | $w_i \exp(-y_{ic}y_{jc}\delta_{jc})$ |
| *log* | $\ln\frac{W_{jc}^+}{W_{jc}^-}$ | $\frac{w_i \exp(-y_{ic}y_{jc}\delta_{jc})}{1-w_i(1-\exp(-y_{ic}y_{jc}\delta_{jc}))}$ |
| *mat* | $\frac{2W_{jc}-1}{2\sqrt{W_{jc}(1-W_{jc})}}$ | $1-\frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}}$ |

Table 2.2: Computation of $\delta_{jc}$ and the weight update rule of our implementation of UNN, for the strictly convex losses in Table 2.1. UNN leverages example $j$ for class $c$, and the weight update is that of example $i$ (See text for details and notations).

which leads to the following final expression of $\delta_{jc}$:

$$\delta_{jc} = \frac{1}{2}\ln\left(\frac{W_{jc}^+}{W_{jc}^-}\right) . \tag{2.24}$$

Therefore, the iterative update of boosting weights $w_{ic}^t$ in (2.10) as a function of $\delta_{jc}$ is expressed like bellow:

$$\begin{aligned} w_{ic}^t &= \exp\left(-y_{ic}h_{jc}^t(\boldsymbol{x}_i)\right) && (2.25)\\ &= \exp\left(-y_{ic}h_{jc}^{t-1}(\boldsymbol{x}_i) - y_{ic}y_{jc}\delta_{jc}\right) && (2.26)\\ &= w_{ic}^{t-1}\exp\left(-y_{ic}y_{jc}\delta_{jc}\right) && (2.27)\end{aligned}$$

For the remaining loss functions, it is not possible to directly solve (2.15). Then we will assume that $\mathrm{F}'(\mathrm{F}'^{-1}(-w_{ic}) + y_{ic}\delta_{jc}y_{jc}) \simeq -w_{ic}\mathrm{F}'(y_{ic}\delta_{jc}y_{jc})$. Therefore, the equation (2.14) becomes:

$$\begin{aligned} \sum_{i=1}^{n_j} y_{ic}y_{jc}w_{ic}^{t-1}\mathrm{F}'(y_{ic}\delta_{jc}y_{jc}) &= 0 && (2.28)\\ \sum_{i=1}^{n_j}[y_{ic}y_{jc}>0]\, w_{ic}^{t-1}\mathrm{F}'(\delta_{jc}) - \sum_{i=1}^{n_j}[y_{ic}y_{jc}<0]\, w_{ic}^{t-1}\mathrm{F}'(-\delta_{jc}) &= 0 && (2.29)\\ W_{jc}^+\mathrm{F}'(\delta_{jc}) - W_{jc}^-\mathrm{F}'(-\delta_{jc}) &= 0 . && (2.30)\end{aligned}$$

Replacing $\mathrm{F}'$ in (2.30) and (2.10) by its expression corresponding to each of the considered losses will directly lead to the Table 2.2. The convergence proof and the theoretical properties of UNN are detailed in [Nock et al. 2012].

## 2.4 Implementation details and optimizations

### 2.4.1 Implementation

Since we are dealing with classification topic for large scale image datasets, UNN should overcome some numerical problems that could arise.

The first one is that, we can face unbalancing problem especially because we are considering a one-vs-all framework. To cope with such problem we use adaptive weights $w_{ic}$. That is: initially, $w_{ic}^0$ are weighted, according to wether they belong or do not belong to the class "$c$", by the proportion of positive (respectively negative) samples in this class such that the sum of weights is equal to 1. Then, at each iteration, we normalize weights $w_{ic}, i = 1..m$, to unity after the update in (2.16).

Note that when $W_j^+$ and/or $W_j^-$ is zero, $\delta_{jc}$ in Table 2.2 is not finite. We suggest a simple alternative to cope with this issue: we use $(W_j^+ + \varepsilon)$ instead of $W_j^+$ and $(W_j^- + \varepsilon)$ instead of $W_j^-$.

Then, for the choice of the prototype $j$ in step **[I.0]** of Algorithm 1, we adopt the next scheme: we pick $T \leq m$, consider the $m$ samples, choose $j$ such that $\alpha_{jc}$ is large enough and enable each example to be chosen only once.

### 2.4.2 Metric setting

Two major issues arise when implementing our UNN algorithm in practice. The first one concerns the distance (or, more generally, the *dissimilarity* measure) used for the k-NN search. The second one consists in setting the value of $k$ for both training and testing our prototype-based classifiers (see section 2.4.3).

In fact, defining the most appropriate dissimilarity measure for k-NN search is particularly challenging when dealing with very high-dimensional feature vectors like the ones commonly used for categorization. Indeed, the standard metric distances may be inadequate when such vectors are generated by sophisticated pre-processing stages (*e.g.*, vector quantization or unsupervised dictionary learning), thus lying on complex high-dimensional manifolds. In general, this should require an additional distance learning stage in order to define the optimal dissimilarity measure for the particular type of data at hand. In this respect, our UNN method has the advantage of being fully complementary with any metric learning algorithm [Bel Haj Ali et al. 2010], acting on the top of the k-NN search (see Appendix A). Furthermore, since we use here BoF based on normalized histograms, we prefer use standard $L1$ distance and then avoid expensive computational tasks.

### 2.4.3 Parameters and optimization

Selecting a good value for $k$ amounts to learning parameter-dependent weak classifiers, where the parameter $k$ specifies the size of the voting neighborhood in classification rule (2.2). From the theoretical standpoint, a brute-force approach is possible with boosting: one can define multiple candidate weak classifiers per example, one for each value of $k$, i.e., for each neighborhood size, and then learn prototypes by optimizing the surrogate risk function over $k$ as well. This strategy has the advantage of enabling direct learning of $k$ at training time. However, training several weak classifiers per example without computation tricks would potentially severely impair the applicability of the algorithm on huge datasets. The solution we propose is subtler: we have modified the classification phase of UNN, and tried a *soft* solu-

| # of categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| k-NN BoF | 76.38 | 57.28 | 45.00 | 40.27 | 36.09 | 32.30 | 24.67 |
| SVM BoF | 83.85 | 67.65 | 58.21 | **53.45** | **47.81** | **44.09** | **35.31** |
| AdaBoost BoF | 75.37 | 58.21 | 45.57 | 37.75 | 32.41 | 29.01 | 26.72 |
| $\text{UNN}_s$ BoF | **84.28** | **70.44** | **58.49** | 51.07 | 46.34 | 41.80 | 31.61 |

Table 2.3: Classification performances of the different methods we tested in terms of the average accuracy or mAP as a function of the number of categories.

tion which, to classify new observations, convolutes weighting with a simple density estimation suggested by boosting. Typically, we consider a logistic estimator for a Bernoulli prior which vanishes with the rank of the example in the neighbors, thus decreasing the importance of the farthest neighbors:

$$\hat{p}(j) = \beta_j = \frac{1}{1 + \exp(\lambda(j-1))} \ , \tag{2.31}$$

with $\lambda > 0$. The shape prior is chosen this way because it was shown that boosting, as carried out in a number of algorithms — not restricted to the induction of linear separators [Nock & Nielsen 2009] — locally fits logistic estimators for Bernoulli priors. The soft version of UNN we obtain, called $\text{UNN}_s$ (for "Soft UNN"), replaces (2.2) by:

$$h_c^\ell(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \beta_j \alpha_{jc} y_{jc} \ . \tag{2.32}$$

Notice that it is useless to enforce the normalization of coefficients $\beta_j$ in (2.31), because it would not change the classification of $\text{UNN}_s$. Notice also that the $\beta_j$ in (2.32) are used only to classify new observations: the training steps of $\text{UNN}_s$ are the same as UNN, and so $\text{UNN}_s$ meets the same theoretical properties as UNN described in [Nock et al. 2012].

## 2.5 Experiments

In this section, we present experimental results of UNN for image categorization. Our experiments aim at carefully quantifying and explaining the gains brought by boosting on k-NN voting on real image databases. In particular, we propose in this section precision and accuracy comparison between UNN vs k-NN, SVM and AdaBoost using Bag-of-Features (BoF) as descriptors. Here, we extracted 2500 SIFT [Lowe 2004] per image to form a codebook of 500 visual words. BoF, of a dimension 500, are then computed by vector quantizing the local features SIFT using this codebook.

We selected 100 categories from the SUN database [Xiao et al. 2010]. We kept all the images of each category and the inherent unbalancing of the original database. We randomly chose half images to form a training set, while testing on the remaining ones. The average accuracy or mAP (Mean average precision) was computed by

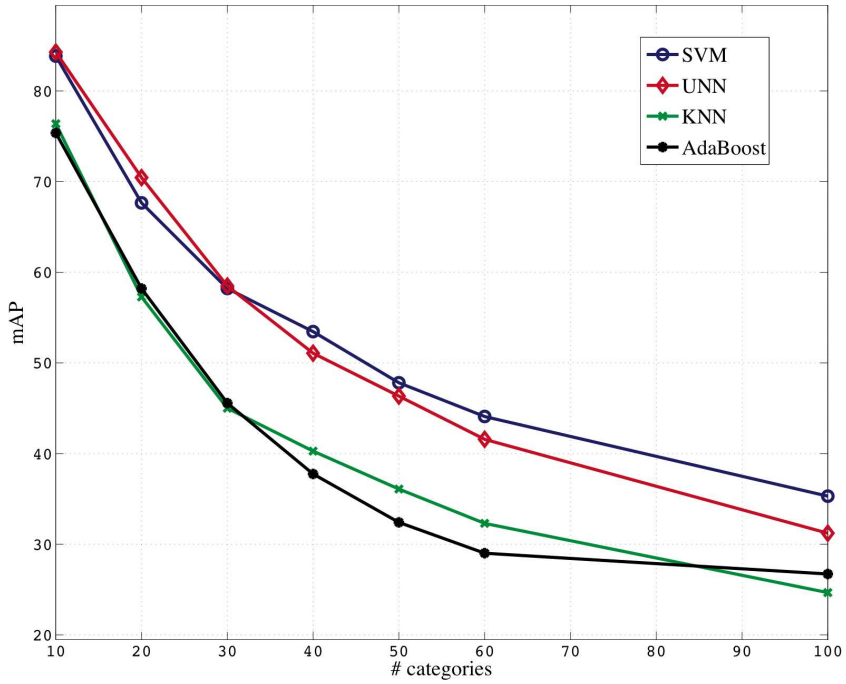


Figure 2.1: Classification performances of the tested methods as a function of the number of image categories.

averaging classification rates over categories (diagonal of the confusion matrix) and then averaging those values after repeating each experiment 10 times on different folds. To speed-up processing time, we used Yael toolbox[1] for a fast implementation of k-NN. Furthermore, we also developed an optimized version of our program, which exploits *multi-thread* functionalities. We denote this version as UNN$_s$(MT.) All the experiments were run on an Intel Xeon X5690 12-cores processor at 3.46 GHz.

We compared UNN$_s$, SVM with Gaussian RBF Kernel, and AdaBoost with decision stumps[2] (i.e., decision trees with a single internal node), using BoF descriptors. In particular, we followed the guidelines of [Hsu et al. 2003] for carrying out the SVM experiments, thus carrying out cross-validation for selecting the best parameters values for SVM.

In Table 2.3 we report the accuracy for each classification method. Results in these tables are provided as a function of the number of image categories. The most relevant results obtained are also displayed in Figure 2.1 (mAP as a function of the number of categories) and Figures 2.2 and 2.3, for the training and classification times, respectively.

Accuracy results display that UNN$_s$ dramatically outperforms AdaBoost (and k-NN as well); this result, which somehow experimentally confirms that UNN successfully exploits the boosting theory, was quite predictable, as UNN builds a piece-

[1]Code available at `https://gforge.inria.fr/frs/?group_id=2151`

[2]For AdaBoost, we used the code available at `http://www.mathworks.com/matlabcentral/fileexchange/22997-multiclass-gentleadaboosting`.

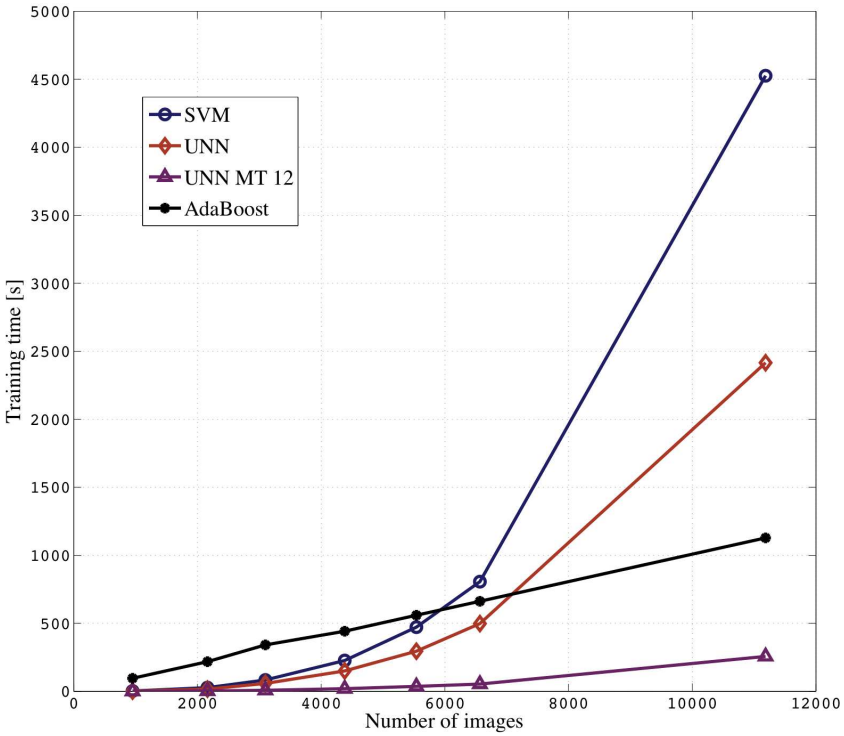


Figure 2.2: Training time as a function of the number of image categories.

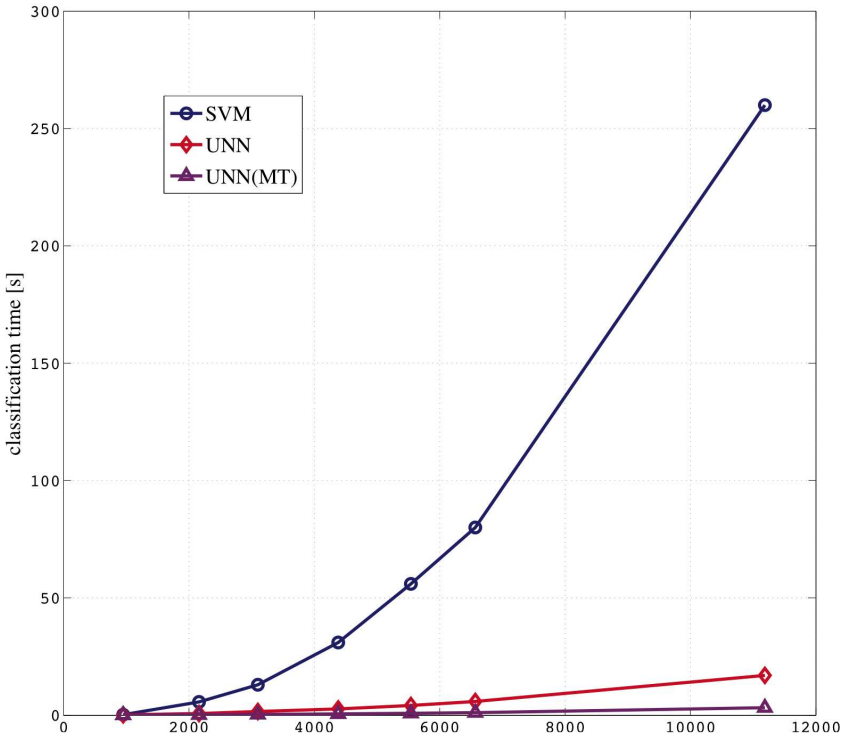


Figure 2.3: Classification time for UNN($_s$) vs SVM as a function of the number of image categories with BoF.

| # categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| # training images | 951 | 2,162 | 3,099 | 4,381 | 5,540 | 6,568 | 11,186 |
| k-NN | 0 | | | | | | |
| SVM | 2.4 | 27 | 83 | 226 | 472 | 806 | 4526 |
| AdaBoost | 96 | 218 | 341 | 442 | 559 | 662 | 1128 |
| $UNN_s$ | 1.7 | 16 | 58 | 150 | 295 | 498 | 2146 |
| $UNN_s$(MT) | 0.3 | 2.5 | 7.8 | 19 | 36 | 53 | 257 |

Table 2.4: Computation time [s] for the training phase.

| # categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| # test images | 951 | 2,162 | 3,099 | 4,381 | 5,540 | 6,568 | 11,186 |
| k-NN | 0.20 | 1.0 | 2.0 | 4.0 | 6.0 | 9.0 | 22.0 |
| SVM | 0.25 | 5.7 | 13 | 31 | 56 | 80 | 260 |
| AdaBoost | 0.02 | 0.1 | 0.25 | 0.43 | 0.67 | 0.95 | 2.74 |
| $UNN_s$ | 0.21 | 0.72 | 1.6 | 2.7 | 4.2 | 5.9 | 17 |
| $UNN_s$(MT) | 0.08 | 0.2 | 0.37 | 0.58 | 0.84 | 1.11 | 3.25 |

Table 2.5: Computation time [s] for the testing phase.

wise linear decision function in the initial domain $\mathbb{X}$, while AdaBoost builds a linear separator in this domain. SVM, on the other hand, have access to non-linear fitting of data, by lifting the data to a domain whose dimension far exceeds that of $\mathbb{X}$. Yet, SVM testing results are somehow not as good as one might expect from this clearcut theoretical advantage over UNN, and also from the fact that we carried out SVM with significant parameters optimization [Hsu et al. 2003]. Indeed, $UNN_s$ even beats SVMs over 10 to 30 categories, being slightly outperformed by them on more categories.

In Table 2.4 and 2.5 we report the corresponding computation time (in seconds) for the training and classification phase, respectively. Obviously, the computation times over training and testing are also a key for exploiting the experimental results. Table 2.4 displays that, while the training time of AdaBoost is linear, $UNN_s$ is a logical clearcut winner over SVM for training: it achieves speedups ranging in between two and more than seventeen over SVM. Thus, UNN provides the best precision/time trade-off among the tested methods, which suggests that UNN might well be more than a legal contender to classification methods dealing with huge domains, or domains where the testing set is huge compared to the training set, which is the case, for instance, for cell classification in biological images. Finally, we have only scratched experimental optimizations for UNN, and have not optimized UNN from the complexity-theoretic standpoint, so we expect room space for further significant improvement of its training/testing times.

## 2.6 Conclusion

In this chapter, we contribute to fill an important void of NN methods, showing how boosting can be transferred to k-NN classification, with convergence rates guarantees for a *large* number of surrogates. UNN, which builds upon the works of ([Piro et al. 2012]), generalizes classic k-NN to weighted voting where weights, the so-called leveraging coefficients, are iteratively learned by UNN. We prove that this algorithm converges to the global optimum of many surrogate risks in competitive times under very mild assumptions. Compared to [Piro et al. 2012], we enlarge the set of formal boosting flavors of UNN, from a singleton associated to the exponential loss to a set encompassing popular losses like the logistic and matsushita loss.

Our approach is also the first extensive assessment of UNN to computer vision related tasks. Comparisons with k-NN, support vector machines and AdaBoost, using Bag-of-Feature descriptors, on real domains, display the ability of UNN to be competitive with its contenders, achieving high accuracy in comparatively reduced training and testing times.

An optimization approach using metric learning was not reported in this chapter, since it does not concern our learning framework, is reported in Appendix A ([Bel Haj Ali et al. 2010]). It includes blending UNN with an approach that learns more sophisticated metrics over data.

Chapter 3

# Newton Nearest Neighbor algorithm: $N^3$

## 3.1 Introduction

Large scale image classification implies satisfying tight time, memory and numerical processing requirements. Coping with them involves in general two kinds of approaches. For the first one, scalability goes hand in hand with simplification: algorithms are built around sophisticated, state-of-the art approaches that are simplified to fit into these requirements, such as Support Vector Machines (SVM) with linear kernels [Shalev-Shwartz et al. 2007], or (Ada)Boosting with weight clipping and simple stumps as weak classifiers [Ali et al. 2011].

The second kind of approaches use as core very simple algorithms that already fit into these requirements, and then, from this basis, elaborate more complex approaches with improved performances: this is the case for the $k$-nearest neighbor (NN) classifier, or the nearest class mean classifier embedded with metric learning [Mensink et al. 2012, Weinberger & Saul 2009]. From the experimental standpoint, these latter approaches obtain surprising competitive results with respect to the former ones. In fact, they may have another advantage: while theoretical guarantees barely survive extreme simplification, elaborating on a core makes it perhaps easier to preserve its theoretical properties, such as its statistical consistency (*e.g.* for k-NN [Devroye et al. 1996]).

Our algorithm belongs to the second category of approaches, as we elaborate on the ordinary k-NN classifier. Our approach is different but complementary to metric learning approaches, as we choose to adapt k-NN to the boosting framework. It is in the same line of works as UNN algorithm introduced in chapter 2, but the present one is of Newton-Raphson type, and then more adapted for large scale classification.

Our high-level contribution is threefold: a novel Adaptive Newton-Raphson scheme to leverage k-NN, called $N^3$, an extensive theoretical analysis of the approach, and fine-grained experimental validations on three large and challenging domains: SUN and Caltech. To be more specific, the novelty of our method includes:
(*i*) a proof of the boosting ability of $N^3$, the first boosting-compliant convergence rates for a Newton-type approach to convex loss minimization to the best of our knowledge;
(*iii*) a proof that the output of $N^3$ directly yields efficient estimators of posteriors;
(*iv*) a divide and conquer algorithm to compute these estimators and cope with the

curse of dimensionality with low memory requirement;
$(v)$ experimentally optimized core-processing stages for N$^3$ with linear cost per boosting iteration.

Experimental results display that N$^3$ manages to challenge accuracy of sophisticated approaches while being faster, and requires low memory.

The remaining of the chapter is organized as follows: Section 3.2 states basic definitions. Section 3.3 presents classification-calibrated losses. Section 3.4 presents N$^3$ algorithm. Section 3.5 discusses its theoretical properties. Section 3.6 presents experiments, and section 3.7 concludes the chapter.

## 3.2 Basic definitions

We first provide some basic definitions. Our setting is multiclass, multilabel classification. We have access to an input set of $m$ examples (or prototypes), $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, +1\}^C$ encodes class memberships, assuming $y_{ic} = +1$ means that observation $\boldsymbol{x}_i$ belongs to class $c$. A classifier $H$ is a function mapping observations to vectors in $\mathbb{R}^C$. Given some observation $\boldsymbol{x}$, the sign of coordinate $c$ in $H(\boldsymbol{x})$ gives whether $H$ predicts that $\boldsymbol{x}$ belongs to class $c$, while its absolute value may be viewed as a confidence in classification.

The nearest neighbors (NNs) rule belongs to the oldest, simplest and still most widely studied classification algorithms [Devroye et al. 1996]. It relies on a non-negative real-valued "distance" function. This function measures how much two observations differ from each other, and may not necessarily satisfy the requirements of metrics. We let $j \rightarrow_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, or simply example $j$, belongs to the $k$ NNs of observation $\boldsymbol{x}$. We shall abbreviate $j \rightarrow_k \boldsymbol{x}_i$ by $j \rightarrow_k i$ — in this case, we say that example $i$ belongs to the *inverse neighborhood* of example $j$. To classify an observation $\boldsymbol{x}$, the $k$-NN rule $H(\boldsymbol{x})$ computes the sum of class vectors of its nearest neighbors, that is: $H_c(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} y_{jc}$ is the coordinate $c$ in $H(\boldsymbol{x})$. A *leveraged* k-NN rule [Nock et al. 2012] generalizes this to:

$$H_c(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} \alpha_{jc} y_{jc} \ , \tag{3.1}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ leverages the classes of example $j$. Leveraging nearest neighbors raises the question as to whether there exists efficient inductive learning schemes for these *leveraging coefficients*.

To learn them, we adopt the framework of [Bartlett et al. 2006, Vernet et al. 2011], and focus on the minimization of a *total calibrated risk* which sums per-class losses:

$$\varepsilon_F(H, \mathcal{S}) \doteq \frac{1}{C} \sum_{c=1}^{C} \underbrace{\frac{1}{m} \sum_{i=1}^{m} F(y_{ic} H_c(\boldsymbol{x}_i))}_{\varepsilon_F(H_c, \mathcal{S})} \ . \tag{3.2}$$

| crit | transfer function $f$ | calibrated loss $F$ |
|---|---|---|
| A | $\frac{1}{1+\exp(-x)}$ | $\ln(1+\exp(-x))$ |
| B | $\frac{1}{1+2^{-x}}$ | $\ln(1+2^{-x})$ |
| C | $\frac{1}{2}\left(1+\frac{x}{\sqrt{1+x^2}}\right)$ | $\exp\sinh^{-1}(-x)$ |
| D | $\frac{1+\max\{0,x\}}{2+\lvert x\rvert}$ | $\max\{0,-x\}-\ln(2+\lvert x\rvert)$ |

Table 3.1: Calibrated losses that match (3.3) for several transfer functions. From top to bottom, losses are the logistic loss, binary logistic loss, Matsushita's loss, calibrated linear Hinge loss.

To be classification-calibrated, loss $F : \mathbb{R} \to \mathbb{R}$ is required to be convex, differentiable and such that $F'(0) < 0$ [Bartlett et al. 2006] (Theorem 4), [Vernet et al. 2011]. The recent advances in the understanding and formalization of (multiclass) loss functions suitable for classification have essentially concluded that classification calibration is mandatory for the loss to be Fisher consistent or proper [Bartlett et al. 2006, Vernet et al. 2011]. These are crucial properties without which the minimization of the loss brings no string statistical guarantee with respect to Bayes rule (such as universal consistency).

## 3.3 Classification-calibrated losses

In this chapter, we are interested in a subset of classification-calibrated functions, namely those for which:

$$F(x) \doteq -x + \int f \ , \tag{3.3}$$

for some continuous *transfer function* $f : \mathbb{R} \to [0,1]$, increasing and symmetric with respect to $(0, 1/2 = f(0))$. Intuitively, a transfer function brings an estimate of posteriors: it is a bijective mapping between a real-valued prediction $H_c(\boldsymbol{x})$ and a corresponding posterior estimation for the class, $\hat{p}[y_c = +1|\boldsymbol{x}]$, mapping which states that both values are positively correlated, and establishes a tie for $H_c = 0$ to which corresponds $\hat{p}[y_c = +1|\boldsymbol{x}] = 1/2$. Transfer functions have a longstanding history in optimization [Kivinen & Warmuth 2001], and the set of $F$ that match (3.3) strictly contains balanced convex losses, functions with appealing statistical properties [Nock et al. 2012] (and references therein). Table 3.1 provides four example of such losses on which we focus. Another example of losses that meet (3.3) is the squared loss, for transfer $f = \min\{1, \max\{0, x + 1/2\}\}$.

To carry out the minimization of (3.2), we adopt a mainstream 1-vs-rest boosting scheme which, for each $c = 1, 2, ..., C$, carries out separately the minimization of

**Algorithm 2:** Algorithm Newton Nearest Neighbors N$^3(\mathcal{S}, \text{crit}, k)$

**Input**: Sample $\mathcal{S}$, criterion crit $\in \{\text{A}, \text{B}, \text{C}, \text{D}\}$, $k \in \mathbb{N}_*$;
Let $\boldsymbol{\alpha}_j \leftarrow \mathbf{0}, \forall j = 1, 2, ..., m$;
**for** $c = 1, 2, ..., C$ **do**
  //Minimize $\varepsilon_F(H_c, \mathcal{S})$
  Let $w_i \leftarrow \frac{1}{\|\mathbf{1} + y_{ic}\boldsymbol{y}_i\|_1}, \ \forall i$;
  **for** $t = 1, 2, ..., T$ **do**
    **[I.0]**//Choice of the example to leverage
    Let $j \leftarrow \text{Wic}(\mathcal{S}, \boldsymbol{w})$;
    **[I.1]**//Leveraging update, $\delta_j$
    Let $\eta(c, j) \leftarrow \sum_{i: j \rightarrow_k i} w_{ti} y_{ic} y_{jc}$;
    Let $n_j \leftarrow |\{i : j \rightarrow_k i\}|$;
    Compute $\delta_j$ following Table 3.2, using crit;
    **[I.2]**//Weights update
    $\forall i : j \rightarrow_k i$, update $w_i$ as in Table 3.2, using crit;
    **[I.3]**//Leveraging coefficient update
    Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;

**Output**: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j \rightarrow_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$

$\varepsilon_F(H_c, \mathcal{S})$ in $\varepsilon_F(H, \mathcal{S})$. To do so, it fits the $c^{th}$ coordinate in leveraging coefficients by considering the two-class problem of class $c$ versus all others.

## 3.4 N$^3$: Adaptive Newton Nearest Neighbors

### 3.4.1 Algorithm

We now present algorithm N$^3$, which stands for "Newton Nearest Neighbors". N$^3$ updates iteratively the leveraging coefficients of an example in $\mathcal{S}$, example picked by an oracle, Wic for "Weak Index Chooser" oracle. We detail below the properties and implementation of Wic. The technical details of the N$^3$ are given in Table 3.2. N$^3$ follows the boosting scheme, with iterative updates of leveraging coefficients followed by an iterative re-weighting of examples. Before embarking into formal algorithmic and statistical properties for N$^3$, we first show that N$^3$ is of Newton-Raphson type.

**Theorem 1** N$^3$ *performs adaptive Newton-Raphson steps to minimize* $\varepsilon_F(H_c, \mathcal{S})$, $\forall c$.

**Proof sketch:** The key to the proof, which we explore further in subsection 3.4.2, is the existence of a particular function $g_F$, strictly concave and symmetric with respect to $1/2$, which allows to rewrite the loss as:

$$F(x) \quad = \quad (-g_F)^\star(-x) \ , \tag{3.4}$$

where $\star$ denotes the (Legendre) convex conjugate. Convex conjugates have the property that their derivatives are inverses of each other. This property, along with (3.4), allows to simplify the computation of the derivatives of the loss, for any example $i$ in the inverse neighborhood of $j$:

$$\begin{aligned}
\frac{\partial F(y_{ic}H_c(\boldsymbol{x}_i))}{\partial \delta_j} &= y_{ic}y_{jc}F'(y_{ic}H_c(\boldsymbol{x}_i)) \qquad (3.5)\\
&= -y_{ic}y_{jc}((-g_F)^\star)'(-y_{ic}H_c(\boldsymbol{x}_i))\\
&= -y_{ic}y_{jc}((-g_F)')^{-1}(-y_{ic}H_c(\boldsymbol{x}_i))\\
&= -y_{ic}y_{jc}(1-(g_F')^{-1}(-y_{ic}H_c(\boldsymbol{x}_i)))\\
&= -y_{ic}y_{jc}(g_F')^{-1}(y_{ic}H_c(\boldsymbol{x}_i))\\
&= -K_F w_i y_{ic}y_{jc}\ . \qquad (3.6)
\end{aligned}$$

Eq. (3.6) holds because we can also rewrite the weights update (Table 3.2) as:

$$w_i \leftarrow \frac{1}{K_F}(g_F')^{-1}\left(\delta_j y_{ic}y_{jc} + g_F'(K_F w_i)\right)\ , \qquad (3.7)$$

where $(g_F')^{-1}$ is the inverse function of the first derivative of $g_F$, and $K_F$ is a normalizing constant: it is respectively $\ln(2), 1, 1/2, 1$ for A, B, C and D in Table 3.3. From (3.5), it also comes $\partial^2 F(y_{ic}H_c(\boldsymbol{x}_i))/\partial\delta_j^2 = F''(y_{ic}H_c(\boldsymbol{x}_i))$, where $F''$ denotes the second derivative. Considering the whole inverse neighborhood of $j$, the Newton-Raphson update for $\delta_j$ is (with $\eta(c,j) \doteq \sum_{i:j\rightarrow_k i} w_{ti}y_{ic}y_{jc}$ in N$^3$):

$$\delta_j \leftarrow \lambda_F \times \frac{K_F\eta(c,j)}{\sum_{i:j\rightarrow_k i} F''(y_{ic}H_c(\boldsymbol{x}_i))}\ , \qquad (3.8)$$

for learning rate $0 < \lambda_F \leq 1$. Matching this expression with the updates in Table 3.2 brings learning rate:

$$0 < \lambda_F = \frac{L_F \sum_{i:j\rightarrow_k i} F''(y_{ic}H_c(\boldsymbol{x}_i))}{K_F n_j} \leq \frac{L_F F''(0)}{K_F} = 1\ ,$$

for each criteria A, B, C and D, where $L_F$ is respectively $4\ln(2), 4/\ln^2(2), 1/2, 4$, and $n_j \doteq |\{i : j \rightarrow_k i\}|$ in N$^3$. The inequalities come from the fact that $F'' > 0$ and takes its maximum in 0 for all criteria. We then check that $F''(0) = K_F/L_F$ for A, B, C and D. □

### 3.4.2 A key to the properties of N$^3$

The duality between real-valued classification and posterior estimation which stems from $f$ (See Section 3.3) is fundamental for the algorithmic and statistical properties[1] of N$^3$. To simplify the statement of results and proofs, it is convenient to make the parallel between our calibrated losses $F$ and functions elsewhere called permissible[2],

[1] See Appendix B for details on statistical properties of N$^3$.

[2] The usual definitions are more restricted: for example the generator of the calibrated linear Hinge loss would not be permissible in the definitions of [Kearns & Mansour 1999, Nock et al. 2012].

| crit | leveraging update, $\delta_j$ | weight update $g : w_i \leftarrow g(w_i, \delta_j, y_{ic}, y_{jc})$ |
|---|---|---|
| A | $\frac{4\ln(2)\eta(c,j)}{n_j}$ | $\frac{w_i}{w_i \ln 2+(1-w_i \ln 2)\times\exp(\delta_j y_{ic} y_{jc})}$ |
| B | $\frac{4\eta(c,j)}{\ln^2(2) n_j}$ | $\frac{w_i}{w_i+(1-w_i)\times 2^{\delta_j y_{ic} y_{jc}}}$ |
| C | $\frac{\eta(c,j)}{2n_j}$ | $1-\frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_j y_{ic} y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_j y_{ic} y_{jc}}}$ |
| D | $\frac{4\eta(c,j)}{n_j}$ | $\frac{1+\max\left\{0,-\left(\delta_j y_{ic} y_{jc}+\frac{1-2w_i}{\mathrm{err}(w_i)}\right)\right\}}{2+\left\lvert\delta_j y_{ic} y_{jc}+\frac{1-2w_i}{\mathrm{err}(w_i)}\right\rvert}$ |

Table 3.2: Leveraging and weight updates in $N^3$ corresponding to each choice of calibrated loss in Table 3.1.

| crit | generator $g_F$ |
|---|---|
| A | $-x\ln x-(1-x)\ln(1-x)$ |
| B | $-x\log_2 x-(1-x)\log_2(1-x)$ |
| C | $\sqrt{x(1-x)}$ |
| D | $\ln(2\mathrm{err}(x))+1-2\mathrm{err}(x)$ |

Table 3.3: Generators corresponding to calibrated losses in Table 3.1.

that is, functions defined on $(0,1)$, strictly concave, differentiable and symmetric with respect to $x = 1/2$. It can be shown that for any of our choices of $F$, there exists a permissible $g_F$, that we call a *generator*, for which the relationships (3.7) and (3.4) used in the proofsketch of Theorem 1 indeed hold. Furthermore, the generator is also useful to write the transfer function itself, as we have:

$$f(x) \quad = \quad (-g_F)'^{-1}(x) \ . \tag{3.9}$$

Table 3.3 provides the four generators corresponding to choices A, B, C and D. The permissible generator of the calibrated linear Hinge loss makes use of the error function:

$$\mathrm{err}(x) \quad \doteq \quad \min\{x, 1-x\} \ . \tag{3.10}$$

Permissible functions (as well as (3.10)) are used in losses that rely on posterior estimation rather than real-valued classification. Such losses are the cornerstone of decision-tree induction and other methods that directly fit posteriors [Devroye et al. 1996]. Hence, (3.4) establishes a duality between the two kinds of losses, duality which appears as a watermark in various works [Bartlett et al. 2006, Friedman et al. 2000]. The writing of the weight update using $g_F$ in (3.7) is also extremely useful to simplify the proofs of the following Theorems. Finally, there is a synthetic writing for the weights, which sheds light on their interpretation: unraveling the weight update (3.7) and using (3.9), we obtain that $w_i$ satisfies:

$$w_i \quad \propto \quad 1 - f(y_{ic} H_c(\boldsymbol{x}_i)) \ . \tag{3.11}$$

Hence, weights and estimated posteriors are in opposite linear relationship. According to (3.11), examples "easier to classify" (receiving large estimated posteriors) receive small weight. This is a fundamental property of boosting algorithms, that progressively concentrate on the hardest examples.

## 3.5 Algorithmic properties of N$^3$

The first result is a direct follow-up from Table 3.2.

**Lemma 2** *With choice D (calibrated linear Hinge loss),* N$^3$ *may be implemented using only rational arithmetic.*

**Comments on Lemma 2**: In the light of the boosting properties of N$^3$, this result is important in itself. Most existing boosting algorithms, including UNN, AdaBoost, Gentle AdaBoost and spawns [Nock et al. 2012, Friedman et al. 2000] make it necessary to tweak or clip the key numerical steps, including weights update or leveraging coefficients [Ali et al. 2011], at the possible expense of failing to meet boosting's convergence or accuracy. Rational arithmetic still requires significant computational resources with respect to floating point computation, but Lemma 2 shows that whenever these are accessible, formal boosting may be implemented *virtually without any loss in numerical precision.*

Let us now shift to the boosting result on $N^3$, which is stated under the following weak learning assumption:

> There exist constants $\gamma_u > 0, \gamma_n > 0$ such that at any iterations $c, t$ of $N^3$, index $j$ returned by WIC is such that $n_j > 0$ and the following holds: (i) $\frac{\sum_{i:j \to_k i} w_i}{n_j} \geq \frac{\gamma_u}{K_F}$, and (ii) $|\hat{p}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \to_k i] - 1/2| \geq \gamma_n$.

Requirement (ii) corresponds to the usual weak learning assumption of boosting: it postulates that the current *normalized* weights in the inverse neighborhood of example $j$ authorize a classification different from random by at least $\gamma_n$. Requirement (i) states that *unnormalized* weights must not be too small. This is a necessary condition as unnormalized weights of minute order do not necessary prevent (i) to be met, but would obviously impair the convergence of $N^3$ given the linear dependence of $\delta_j$ in the unnormalized weights. The following Theorem states that $N^3$ is a boosting algorithm.

**Theorem 3** *Suppose* $N^3$ *is ran for* $T$ *steps for each* $c$*, and that the weak learning assumption holds at each iteration of* $N^3$*. Denote* $\mathcal{J}$ *the whole multi-set of indexes returned by* WIC*. Then for any criterion A, B, C, D, the total calibrated risk does not exceed some* $\varepsilon \leq F(0)$ *provided:*

$$\sum_{j \in \mathcal{J}} n_j \quad = \quad \Omega\left(\frac{(C + |\varepsilon|)m}{\gamma_n^2 \gamma_u^2}\right) \quad . \tag{3.12}$$

*Remark*: requirement $\varepsilon \leq F(0)$ comes from the fact that a leveraged NN with null leveraging vectors would make a total calibrated risk equal to $F(0)$.

**Comments on Theorem 3**: to the best of our knowledge, no formal convergence rate has been established to date for Newton approaches to boosting, including the popular Gentle AdaBoost [Friedman et al. 2000]. Theorem 3 gives several rules of thumb to run $N^3$ and implement WIC. The first is that WIC should choose examples whose inverse neighborhood is not too small. For example, assume that boosted examples have inverse neighborhood's size not smaller than the average, implying $(1/T)\sum_{j \in \mathcal{J}} n_j \geq k$. Then, omitting constants in the big omega of (3.12), we obtain that (3.12) is satisfied as soon as the number of iterations ($T$) meets:

$$T \quad \geq \quad \frac{(C + |\varepsilon|)m}{k \gamma_n^2 \gamma_u^2} \quad .$$

This inequality suggest to choose $k$ (i) proportional to $C$ and (ii) moderately increasing in $m$. These two choices imply, under the weak learning assumption, that $N^3$ is a *sparse* boosting algorithm: we only need to boost a *subsample* of $\mathcal{S}$ to reach a desired upperbound for the calibrated risk.

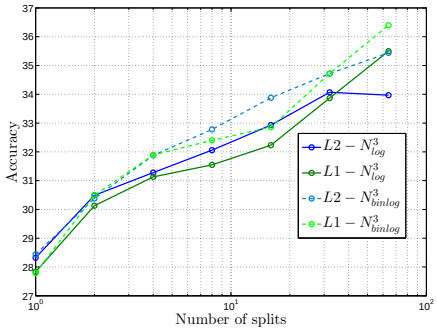


Figure 3.1: The $x$-axis is the number of splits of the FV, on CAL. The $y$-axis reports, using $L1$ or $L2$ normalization, the top1 accuracy of $N^3$. Posteriors were combined with the harmonic mean.

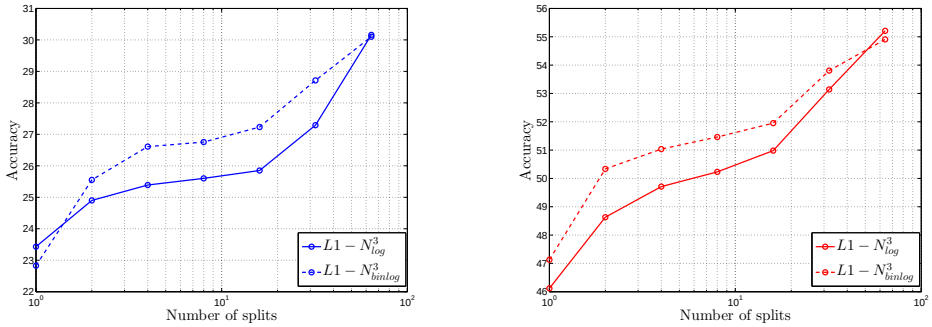


Figure 3.2: The $x$-axis is the number of splits of the FV, on SUN. The $y$-axis reports, using $L1$ or $L2$ normalization, the top1 accuracy (left) and top5 accuracy (right) of $N^3$. Posteriors were combined with the harmonic mean.

## 3.6 Experimental Evaluation

### 3.6.1 Settings: contenders, databases and features

We mainly report and discuss experiments of $N^3$ versus k-NN and support vector machines (SVM) implemented with Stochastic Gradient Descent SGD which represents the state of art among the classifiers on large scale datasets [Perronnin et al. 2012].

We abbreviate $N^3_{log}$, $N^3_{binlog}$, $N^3_{mat}$, $N^3_{hinge}$ the four flavors of $N^3$ corresponding respectively to rows A, B, C, D in Table 3.1. In $N^3$, WIC chooses the example with the largest current $\delta_j$.

The **datasets** used in this chapter, Caltech256, and SUN are among the most challenging datasets publicly available for large scale image classification:

• Caltech256 [Griffin et al. 2007] (CAL): This dataset is a collection of 30607 images

| | | k-NN | $N^3_{log}$ | $N^3_{binlog}$ | $N^3_{hinge}$ | $N^3_{mat}$ |
|---|---|---|---|---|---|---|
| ACC | $L1$ | 25.58 | 35.50 | 36.40 | 33.62 | 34.40 |
| | $L2$ | 25.90 | 33.97 | 35.44 | 32.87 | 33.55 |

Table 3.4: Top1 accuracy on CAL (64 splits, $L1$ or $L2$ normalization).

| | k-NN | $N^3_{log}$ | $N^3_{binlog}$ | SGD |
|---|---|---|---|---|
| Top1 ACC | 20.92 | 30.16 | 30.10 | 28.59 |
| Top5 ACC | 42.67 | 55.21 | 54.90 | 57.08 |

Table 3.5: Top5 accuracy on SUN (64 splits, $L1$ normalization).

of 256 object classes. Following classical evaluation, we use 30 images/class for training and the rest for testing.

• SUN [Xiao et al. 2010] (SUN): This dataset is a collection of 108656 images divided into 397 scenes categories. We set the number of training images per class to 50 and we test on the remaining.

We adopted for the **features** the Fisher vectors (FV) [Perronnin et al. 2010] encoding to represent images. Fisher Vector are computed over densely extracted SIFT descriptors ($FV_s$) and local color features ($FV_{sc}$), both projected with PCA in a subspace of dimension 64. Fisher Vectors are extracted using a vocabulary of 16 Gaussian and normalized separately for both channels and then combined by concatenating the two features vectors ($FV_{s+sc}$). This approach leads to to a 4K dimensional features vector.

To **compare** algorithms, we adopt the top1 and top5 accuracies (ACC), defined respectively as the proportion of examples that was correctly labelled and the proportion of those for which the correct class belongs to the top5 predicted patterns [Mensink et al. 2012]. We also report processing times on a 2 X Intel Xeon E5-2687W 3,1GHz and analyse the convergence and the cost of $N^3$. But first, we propose a divide and conquer algorithm that optimizes classification using posteriors.

### 3.6.2 A divide and conquer algorithm to cope with the curse of dimensionality with low memory requirement

It is well known that NN classifiers suffer of the curse of dimensionality [Beyer et al. 1999], hubs [Radovanović et al. 2010], so that the accuracy can decrease when increasing the size of descriptors. This may also affect $N^3$. FV are extremely powerful descriptors but they generate a space with about $4K$ dimension for 32 gaussians that could impair $N^3$ performance.

Our approach relies on nice property of minimizing classification-calibrated losses: we can easily compute the posteriors from the score using $N^3$ (see [D'Ambrosio et al. ear]). Thus, we propose a three step splitting method :

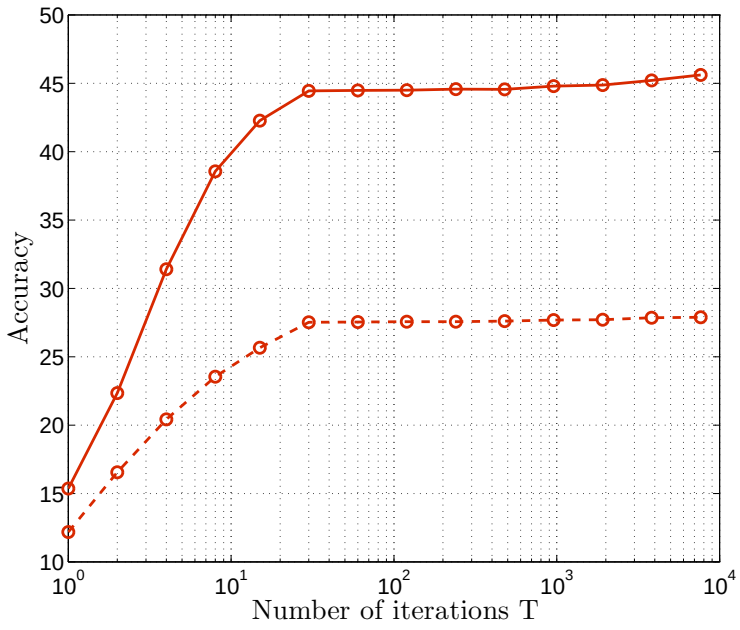


Figure 3.3: Top1 and top5 accuracies (with 1 split) on CAL as a function of the number of iterations $T$.

• split FV in a regular set of $n^* \in \{2, 4, 8, 16, 32, 64\}$ sub-descriptors and normalize with $L1$ or $L2$ norm;
• compute posteriors for each sub-vector (Table 3.1);
• combine these probabilities using a generalized average: arithmetic, geometric or harmonic.

### 3.6.3 Analysis on accuracy and convergence

First, figures 3.1 and 3.2 validate the divide and conquer approach, as increasing the number of splits on FV clearly improves performances. Also, as seen from the left plot, $L1$ normalization tends to outperform $L2$ normalization. The "optimal" number of splits (64) is then used in Table 3.4 which displays that $L1$ normalization of FV slightly improves classical $L2$ normalization. $\mathrm{N}^3_{binlog}$ is also better than all other flavors of $\mathrm{N}^3$, and overall all flavors of $\mathrm{N}^3$ very significantly outperform k-NN.

We have also compared $\mathrm{N}^3$ against SGD and k-NN on the SUN data set [Xiao et al. 2010]. Results using $T = 50$ *iter* for $\mathrm{N}^3$ and 1000 *iter* for SGD are displayed in Table 3.5. One sees that $\mathrm{N}^3$ significantly beats $\mathrm{N}^3$ and approaches the accuracy of SGD. Note that memory requirement for $\mathrm{N}^3$ is divided by the number of splitting (i.e. twice the number of Gaussian of the Fischer Vector).

Figures 3.3 and 3.4 shows the convergence of $\mathrm{N}^3$ on CAL and SUN. One sees from the plots that the convergence of the Newton approach in $\mathrm{N}^3$ is extremely fast

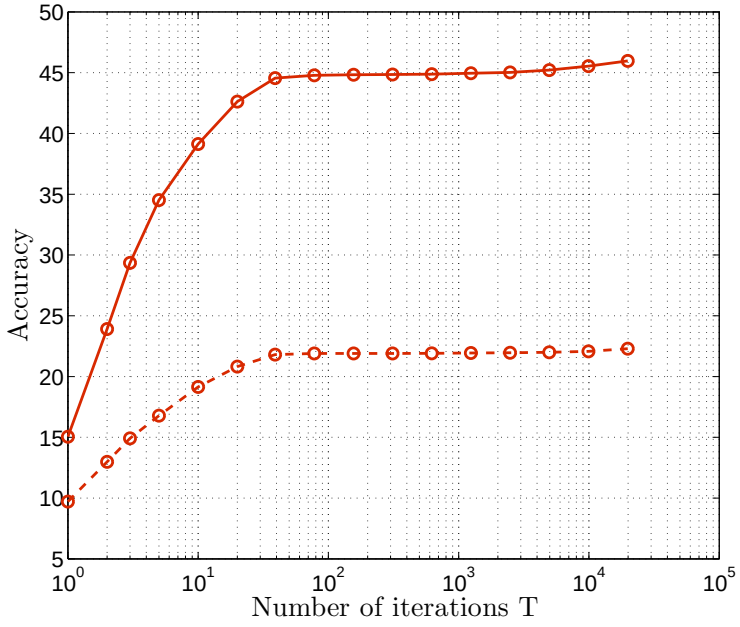


Figure 3.4: Top1 and top5 accuracies (with 1 split) on SUN as a function of the number of iterations $T$.

and requires only few iterations — this is not the case for the non-Newton approach UNN [Nock et al. 2012], which requires a larger number of iterations. The fast convergence in N$^3$ results in sparse prototype selection ($T \ll m$), well adapted for large scale datasets, and suggests to choose $T$ as a function of the number of images in the corresponding class (inner loop of N$^3$), such as $T = \mathcal{O}(m/C)$. Hence, we end up with a complexity depending on $T \ll m$.

## 3.7 Conclusion

In this chapter we have proposed a novel Newton-Raphson approach to boosting k-NN. We show that it is a boosting algorithm, with several key algorithmic and statistical properties. In fact, the specific set of calibrated loss functions allows us to estimate the posteriors from the classification scores of N$^3$, and use them in a divide and conquer scheme to cope with the k-NN's curse of dimentionality. Experiments display that although accuracy results are similar to state of the art approaches like SGD, our N$^3$ requires limited memory since we split the features and use each part independently. This makes our approach suitable for very large scale image classification problems.

# Part II

# Learning Linear Classifiers with Calibrated Losses

Chapter 4

# Stochastic Low-Rank Newton Descent algorithm: SLND

## 4.1 Introduction

Large scale image classification requires computational efficiency. To cope with these issues, current standard approaches involves high dimensional features like Fischer Vectors [Perronnin et al. 2010] or super vectors [Zhou et al. 2010] and Support Vector Machines (SVM) with linear kernels for training [Wang et al. 2010]. The classical approach introducing SVM first states dual formulation [Vapnik 1998] where the task is to minimize empirical risk in the dual space with a regularization term. The first alternative approach on primal optimization [Keerthi et al. 2006] used conjugate gradient or cutting plane algorithms [Joachims 2002]. Recent state of the art papers rather focus on the more efficient "Stochastic Gradient Descent" algorithm (SGD) [Zhang 2004, Bottou & Bousquet 2008] and the "PEGASOS" algorithm [Shalev-Shwartz et al. 2007], with linear complexity in the number of samples.

Although SGD methods perform as well as batch solvers at a fraction of cost, first order SGD methods still suffer from slow convergence. Two approaches were recently proposed in order to cope with this issue; The first is the natural gradient approach, which incorporates the estimation of the Riemannian metric tensor using Fisher information [Amari 1998]. The second alternative approaches are based on a stochastic version of the quasi Newton Broyden-Fletcher-Golfarb-Shanno (BFGS) optimization algorithm. The first one is a low memory stochastic version of the BFGS quasi Newton method [Schraudolph et al. 2007]. Although their BFGS method reduces the number of iterations, each iteration requires a multiplication by a low rank matrix. Unfortunately this computational complexity is often larger than the gains associated with the quasi-Newton update as pointed in [Bordes et al. 2009]. In order to cope with this complexity [Bordes et al. 2009, Bordes et al. 2010] proposed a "SGD-QN" algorithm with an update using the diagonal of the Hessian matrix. Unfortunately there are no proof of convergence of their "SGD-QN" algorithm.

Our high-level contribution is a new stochastic Low-Rank Newton scheme with theoretical proofs and experimental validations on three large and challenging domains: SUN, Caltech256 and ImageNet. To be more specific, the novelty of our approach includes:

(i) A new Stochastic Newton descent algorithm, SLND, which approximates the

inverse Hessian by a low-rank matrix which we prove is the best according to the squared Frobenius norm. SLND minimizes any *classification calibrated risk*, that may ensure convergence towards Bayes rule;

(ii) The proof of convergence of SLND which provides rates of convergence and working set of parameters for the experiments, including the step size parameter $\eta_t$;

(iii) Experimental results display that SLND has linear complexity both in term of the number of samples and the dimension of the features and challenges the accuracy of SGD while being a magnitude faster.

The remaining of the chapter is organized as follows: section 4.2 summarizes the general framework, section 4.3 provides our new algorithm SLND with several key steps for its core optimization, section 4.4 presents experiments on large datasets and finally section 4.5 presents convergence proof of our new algorithm SLND.

## 4.2 Reminder

### 4.2.1 Framework

We first remind some definitions. Our setting is multiclass, multilabel classification. We have access to an input set of $m$ samples, $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, +1\}^C$ encodes class memberships, assuming $y_{ic} = +1$ means that observation $\boldsymbol{x}_i$ belongs to class $c$. A classifier $h$ is a function mapping observations to real-valued vectors in $\mathbb{R}^C$. Given some observation $\boldsymbol{x}$, the sign of coordinate $c$ in $h(\boldsymbol{x})$, $h_c$, gives whether $h$ predicts that $\boldsymbol{x}$ belongs to class $c$, while its absolute value may be viewed as a confidence in classification.

To learn this classifier, we focus on the minimization of a total risk which sums over classes and over samples the loss of classification according to $h$;

$$\varepsilon_{\mathrm{F}}(h, \mathcal{S}) \doteq \frac{1}{C} \sum_{c=1}^{C} \underbrace{\frac{1}{m} \sum_{i=1}^{m} \mathrm{F}(y_{ic} h_c(\boldsymbol{x}_i))}_{\varepsilon_{\mathrm{F}}(h_c, \mathcal{S})} . \tag{4.1}$$

$\varepsilon_{\mathrm{F}}(h_c, \mathcal{S})$ is the per-class risk, and F is a surrogate loss function.

### 4.2.2 Calibrated risks

Recent advances in classification allow to precisely define constraints with whom losses F in (4.1) have to comply, to meet statistical and computational properties particularly desirable in handling large, complex and noisy classification problems [Bartlett et al. 2006, Nock & Nielsen 2008, Vernet et al. 2011]. There are three constraints: F is convex, differentiable and such that $\mathrm{F}'(0) < 0$. We restrict our interest to losses that also meet the following property:

$$\mathrm{F}(x) = -x + \int f \ , \tag{4.2}$$

| crit | transfer function $f$ | calibrated loss $F$ |
|---|---|---|
| A | $\frac{1}{1+\exp(-x)}$ | $\ln(1+\exp(-x))$ |
| B | $\frac{1+\max\{0,x\}}{2+\lvert x\rvert}$ | $\max\{0,-x\}-\ln(2+\lvert x\rvert)$ |

Table 4.1: Calibrated losses $\mathrm{F}^{crit}$ and their respective transfer functions. A is the logistic loss and B is the calibrated linear hinge loss.

where $f : \mathbb{R} \to [0,1]$ is increasing and symmetric with respect to $(0, 1/2 = f(0))$. The fundamental intuition is that $f$ directly maps a real valued prediction $h_c$ to a posterior estimation for class $c$ (see [D'Ambrosio et al. ear]). This last constraint ensures that the loss at hand F is Fisher consistent and proper, properties with which convenient form of convergence to Bayes rule are accessible through minimizing (4.1). We call losses that meet these constraints, and the total risks by extension, as *classification calibrated*. Examples of classification calibrated losses include the squared and the logistic losses. In this chapter, we first consider the logistic loss:

$$\mathrm{F}^{log}(x) \doteq \ln(1+\exp(-x)) \ . \tag{4.3}$$

Then, we consider the calibrated linear Hinge loss, previously introduced in chapter 3, as:

$$\mathrm{F}^{hinge}(x) \doteq \max\{0,-x\} - \ln(2+|x|) \ . \tag{4.4}$$

Table 4.1 gives the considered losses F and their corresponding transfer function $f$. Figure 4.1 shows the logistic loss and the calibrated linear Hinge loss. We also plot Hinge loss and the exponential loss for comparison. Note that $0 < \mathrm{F}''(x) \leq \mathrm{F}''(0)$ and $\mathrm{F}''(0) = 1/4$ for the calibrated losses (4.3) and (4.4).

## 4.3 SLND: Stochastic Low-Rank Newton Descent

### 4.3.1 Computing gradient update

To carry out the minimization of (4.1), we adopt a mainstream 1-vs-rest training scheme which is more efficient among different approaches [Perronnin et al. 2012, Weston et al. 2011]. For each class $c = 1, 2, ..., C$, we carry out separately the minimization of $\varepsilon_{\mathrm{F}}(h_c, \mathcal{S})$ in $\varepsilon_{\mathrm{F}}(h, \mathcal{S})$. To do so, it fits the $c^{th}$ component of $h$ by considering the class $c$ versus all others. **In what follows, we thus drop "$c$" to simplify notations.**

In this approach we focus on the classical linear classifier defined as:

$$h(\boldsymbol{x}_i) \doteq \boldsymbol{w}^\top \boldsymbol{x}_i \ . \tag{4.5}$$

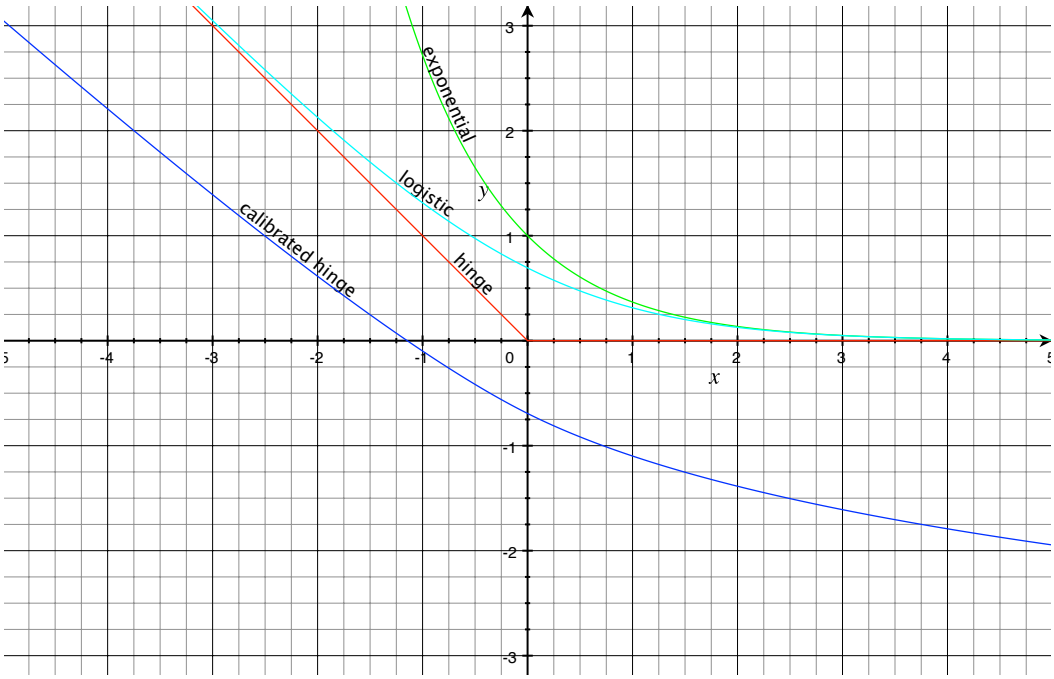


Figure 4.1: Calibrated losses $F$: the logistic and calibrated linear Hinge losses considered for SLND algorithm.

The goal is to learn $w$ for each class $c = 1, 2, ..., C$ minimizing the following criterion, after replacing $h_c$ in $\varepsilon_{\mathrm{F}}(h_c, \mathcal{S})$ by its expression in 4.5 :

$$\varepsilon_{\mathrm{F}}(\boldsymbol{w}, \mathcal{S}) \doteq \frac{1}{m} \sum_{i=1}^{m} \mathrm{F}(y_{ic} \boldsymbol{w}^{\top} \boldsymbol{x}_i) \ . \tag{4.6}$$

**Remark**: there is no regularization term in (4.6) (and in (4.1) in general), which is quite non-standard if we refer to the classical SVM or SGD approaches [Bordes et al. 2009].

To approximate the optimal $\boldsymbol{w}$, we carry out an iterative stochastic Newton algorithm. In general, this aims at updating at each iteration $t$, the current $w$ noted $\boldsymbol{w}_t$, according to a randomly picked sample $\boldsymbol{x}_i \in \mathcal{S}$ as follows :

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \left( \frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \boldsymbol{x}_i)}{\partial^2 \boldsymbol{w}_t} \right)^{-1} \frac{\partial \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \boldsymbol{x}_i)}{\partial \boldsymbol{w}_t} \ , \tag{4.7}$$

where $\eta_t > 0$ controls the strength of the update. In such case, the first derivative or the gradient $\nabla$ is:

$$\frac{\partial \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \boldsymbol{x}_i)}{\partial \boldsymbol{w}_t} = y_i \mathrm{F}' \left( y_i \boldsymbol{w}_t^T \boldsymbol{x}_i \right) \boldsymbol{x}_i \ , \tag{4.8}$$

and the second derivative, or the Hessian $\mathcal{H}$, is:

$$\frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \boldsymbol{x}_i)}{\partial^2 \boldsymbol{w}_t} = \mathrm{F}'' \left( y_i \boldsymbol{w}_t^T \boldsymbol{x}_i \right) \boldsymbol{x}_i \boldsymbol{x}_i^T \ . \tag{4.9}$$

Unfortunately it is well known that the Hessian matrix typically varies as the samples $\boldsymbol{x}_i$ changes. Thus, instabilities arise quickly if we try to estimate it for each sample [Bordes et al. 2009]. To circumvent these problems, statistic optimization approaches consider instead an averaging of the Hessian. For example, [Ljung & Söderström 1983] rewrite the stochastic Newton algorithm as follow :

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \left( \mathbb{E}[\frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}_t)}{\partial^2 \boldsymbol{w}_t}] \right)^{-1} \frac{\partial \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \boldsymbol{x}_i)}{\partial \boldsymbol{w}_t} \ , \tag{4.10}$$

where $\mathcal{S}_t \subseteq \mathcal{S}$ is the set of samples $\boldsymbol{x}_i$ picked until the iteration $t$. The update of the averaged Hessian in (4.10) is quite expensive in the case of huge datasets and large scale features. Hence, we follow [Li 1992, Cook 1998] who average the Hessian once and approximate it by the covariance matrix. We consider $\mathbb{E}[\frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}(m'))}{\partial^2 \boldsymbol{w}_t}]$, with $\mathcal{S}(m')$ a subset of $m' \leq m$ random examples from $\mathcal{S}$, instead of $\mathbb{E}[\frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}_t)}{\partial^2 \boldsymbol{w}_t}]$ in (4.10). Let recall that $0 < \mathrm{F}''(x) \leq \mathrm{F}''(0)$ for the calibrated losses (4.3) and (4.4). Then, we will consider the following approximation :

$$\mathbb{E}[\frac{\partial^2 \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}(m'))}{\partial^2 \boldsymbol{w}_t}] = \frac{1}{m'} \sum_{i \in \mathcal{S}(m')} \mathrm{F}'' \left( y_i \boldsymbol{w}_t^T \boldsymbol{x}_i \right) \boldsymbol{x}_i \boldsymbol{x}_i^T \tag{4.11}$$

$$\approx \mathrm{F}''(0) \frac{1}{m'} \sum_{i \in \mathcal{S}(m')} \boldsymbol{x}_i \boldsymbol{x}_i^T \ , \tag{4.12}$$

Consequently, computing $\mathcal{H}^{-1}$, requires only once the principal hessian direction (p.h.d.) using eigenvectors for the eigenvalue decomposition of the covariance matrix.

For sometypically small $k > 0$, we compute a low-rank pseudo-inverse, *i.e.* a low-rank approximation of its inverse, $\mathcal{H}^*$, as follows, where $rank(\mathcal{H}^*) = k$ is user-fixed. First, we perform a diagonalization of $\mathcal{H} = \mathcal{P}\mathcal{D}\mathcal{P}^\top$ where (non-negative) diagonal values are ordered in decreasing order, $d_{11} \geq d_{22} \geq ... \geq d_{uu} = 0 = ...d_{nn}$, where $u \geq k$. Denote $\mathcal{P}_{|k}$ the $m \times k$ matrix containing the first $k$ columns of $\mathcal{P}$, and resp. $\mathcal{D}_{|k}$ as the $k \times k$ diagonal matrix of their eigenvalues. We finally compute $\mathcal{H}^*$ only once:

$$\mathcal{H}^* = \mathcal{P}_{|k} \mathcal{D}_{|k}^{-1} \mathcal{P}_{|k}^\top \ . \tag{4.13}$$

The update (4.7) becomes:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t y_i F' \left( y_i \boldsymbol{w}_t^T \boldsymbol{x}_i \right) \mathcal{H}^* \boldsymbol{x}_i \ . \tag{4.14}$$

### 4.3.2 Core optimization

Since we use 1-vs-rest training scheme, the training set is usually highly unbalanced when the number of class increases, examples not in class $c$ outnumbering those in class $c$, for any $c$. When class $c$ is a minority class, this is even more dramatic. To dampen the negative consequences, we follow the sampling balancing approach

proposed by [Perronnin et al. 2012]. When learning class $c$ against the rest, we use all examples from class $c$ (the positives), while sampling a subset of the rest of the other classes (the negatives) of the same size.

Furthermore, in order to optimize computational complexity, once $\mathcal{H}^*$ is computed, we precompute for all the training set a weighted preprocessing of the features:

$$\boldsymbol{x}_i^* = \mathcal{H}^* \boldsymbol{x}_i \ . \tag{4.15}$$

Notice that this is done only once for a given $\mathcal{H}^*$. This saves significant training time and the computational complexity of each iteration in SLND is basically of the same order as classical SGD [Bordes et al. 2009]. The final update in SLND is:

$$\boldsymbol{w}_{t+1} \quad = \quad \boldsymbol{w}_t - \eta_t y_i F' \left( y_i \boldsymbol{w}_t^T \boldsymbol{x}_i \right) \boldsymbol{x}_i^* \ . \tag{4.16}$$

Finally, the tuning of $\eta_t$ is a non-trivial problem for gradient or Newton approaches [Bordes et al. 2009]. We prove an explicit convergence rate for SLND in Theorem 5 hereafter which provides us with expressions for $\eta_t$ typically in the order $\Omega(1/m)$ and $O(1/\sqrt{m})$. The values we have chosen in our implementation of SLND belong to this range and are thus compatible with the formal convergence rates shown for SLND.

### 4.3.3 Remarks

There are several comparisons to make about SLND with respect to other prominent approaches. First, SLND is not related to (linear) SVM, as there is no regularization term in the criterion (4.6), which explains the difference between the right hand-side term in $\boldsymbol{w}_t$ in (4.6) and the term in $(1-\lambda)\boldsymbol{w}_t$ which would follow from the classical linear SVM cost function, where $\lambda$ controls the strength of regularization [Bordes et al. 2009]. Also, SLND is significantly different from dimensionality reduction techniques like PCA or general non-linear manifold learning, which would carry out dimensionality reduction as a preconditioning *on data* and on $w$, thus working on the reduced domain. Notice also that (4.15) is not a preconditioning of data, as each iteration in (4.16) makes use of both $\boldsymbol{x}_i$ and $\boldsymbol{x}_i^*$. In addition, SLND is also different from the quasi newton (L)BFGS family [Nocedal 1980] [Schraudolph et al. 2007] as the approximation to the Hessian inverse is carried out in a different way. Moreover SLND differs from quasi-Newton methods for SVM [Bordes et al. 2009] since we do not restrict the Hessian approximation to be diagonal (thus omitting all covariance terms). Finally, SLND is not a natural gradient approach (which incorporates Riemannian metric tensor [Amari 1998]) and thus SLND does not require the computation of the Fisher information matrix.

## 4.4 Experimental evaluation

### 4.4.1 Settings

We mainly report and discuss experiments of SLND versus SGD which represents the state of art among the classification methods on large scale datasets [Zhang 2004, Bottou & Bousquet 2008] , [Shalev-Shwartz et al. 2007], [Perronnin et al. 2012].

We use Fisher vectors (FV) [Perronnin et al. 2010] as efficient **features** to represent images. Fisher Vectors are computed over densely extracted SIFT descriptors ($\text{FV}_s$) and local color features ($\text{FV}_{sc}$), both projected with PCA in a subspace of dimension 64. Since the goal of the chapter is to compare SLND versus SGDwe use Fisher Vectors using a vocabulary of only 16 Gaussian to limit memory requirement. Each Fisher Vectors are normalized separately for both channels and then combined by concatenating the two features vectors ($\text{FV}_{s+sc}$). This approach leads to to a 4K dimensional features vector.

We report experimental results on three **datasets**, Caltech256, SUN and ImageNet which are among the most challenging datasets publicly available for large scale image classification:

- Caltech256 [Griffin et al. 2007]: This dataset is a collection of 30607 images of 256 object classes. Following classical evaluation, we use 30 images/class for training and the rest for testing.

- SUN [Xiao et al. 2010]: This dataset is a collection of 108656 images divided into 397 scenes categories. We set the number of training images per class to 50 and we test on the remaining.

- ImageNet [Deng et al. 2009]: We use the dataset of the ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC2010)[1] with its 1000 categories. It provides $1.2M$ of images for training step and $150K$ for testing.

To **compare** algorithms, we use top1 and top5 accuracies (ACC), defined respectively as the proportion of examples that was correctly labelled and the proportion of those for which the correct class belongs to the top5 predicted images [Mensink et al. 2012]. We first analyse parameter of SLND and then the convergence of SLND.

### 4.4.2 Tuning parameters of SLND

Our algorithm requires the tuning of only three parameters: the step size parameter $\eta_t$ , the rank $k$ and the number of sample $m'$ for the computation of the covariance matrix. The step size parameter $\eta_t$ is typically in the order $\Omega(1/m)$.
Let us study the influence of parameters $k$ and $m'$.

[1] http://image-net.org/challenges/LSVRC/2010/index

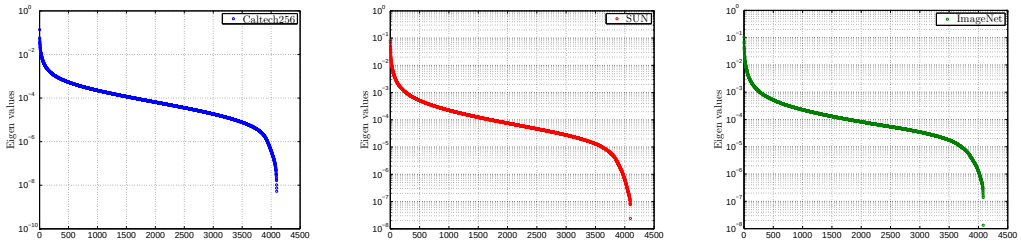


Figure 4.2: Eigenvalues of the covariance matrix on Caltech256 (left), SUN (center) and ImageNet (right).

Fig 4.2 shows the eigenvalues of the covariance matrix, ordered from the largest to the smallest. All curves have the same sigmoid shape, and our choices of $k$ ensure that we get all the significantly large eigenvalues. Recall that although the covariance matrix is positive-definite, the condition number is very large resulting in an ill-conditioned problem.

In order to cope with this issue, we study the accuracy as a function of the rank of the inverse of the Hessian: Fig 4.3 shows that accuracy curve has its max for a large rank plateau, and furthermore this plateau shape is similar regardless of the domain.

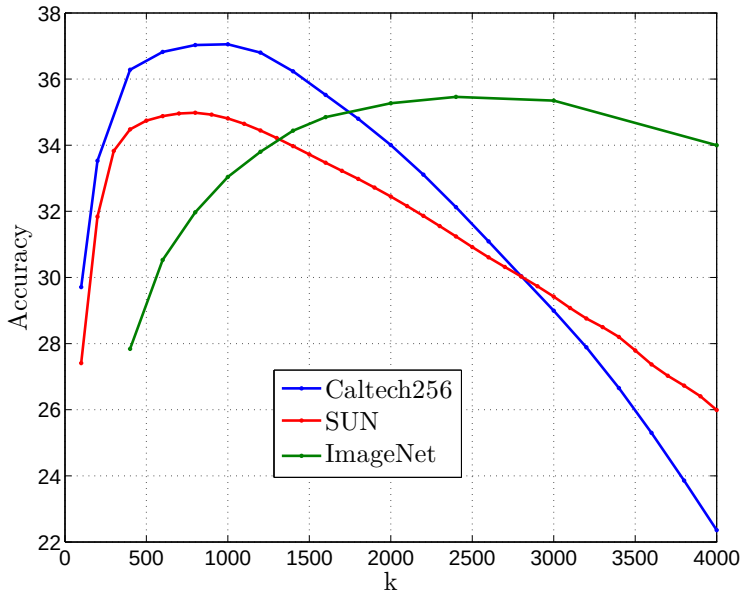


Figure 4.3: Accuracy as a function of the rank of the Hessian matrix on Caltech256 (blue), SUN (red) and ImageNet (green).

Fig 4.4 shows the accuracy as a function of samples $m'$ used for computing the covariance matrix. Fluctuations of $m'$ imply fluctuations in the accuracy, but the range of the accuracy is not very large for reasonable values of $m'$.

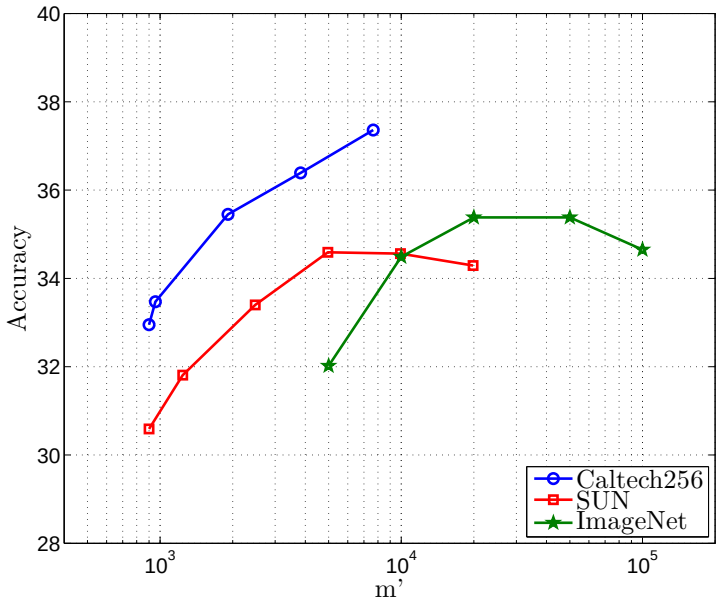


Figure 4.4: Accuracy as a function of the number of samples used for the computation of the Hessian matrix on Caltech256 (blue), SUN (red) and ImageNet (green, see text).

To summarize, the eigenvalues curve, the curve accurary as a function of the rank $k$ and to a lesser extent the curve accuracy as a function of $m'$ have the same behavior for all databases. Thus, based on the above-experiments, both rank $k$ and $m'$ in SLND are easily tuned.

### 4.4.3 Convergence rate analysis

Training time and convergence of algorithms are very important for large scale data base processing. We plot on fig 4.5 and 4.6 the convergence of SGD with logistic loss, SLND both for Logistic Loss and calibrated linear Hinge Loss and SGD-QN for logistic Loss on Caltech256 and SUN data bases. One sees from the plots that the convergence of our Stochastic Low-Rank Newton approach SLND is a magnitude faster than the classical SGD. Note that accuracy of Logistic Loss and calibrated linear Hinge Loss SLND are very similar. Accuracy of SGD-QN is very close to SGD on Caltech256 and SUN and slightly better on ImageNet; We get similar results when using only a diagonal approximation of the Hessian matrix in our SLND method.

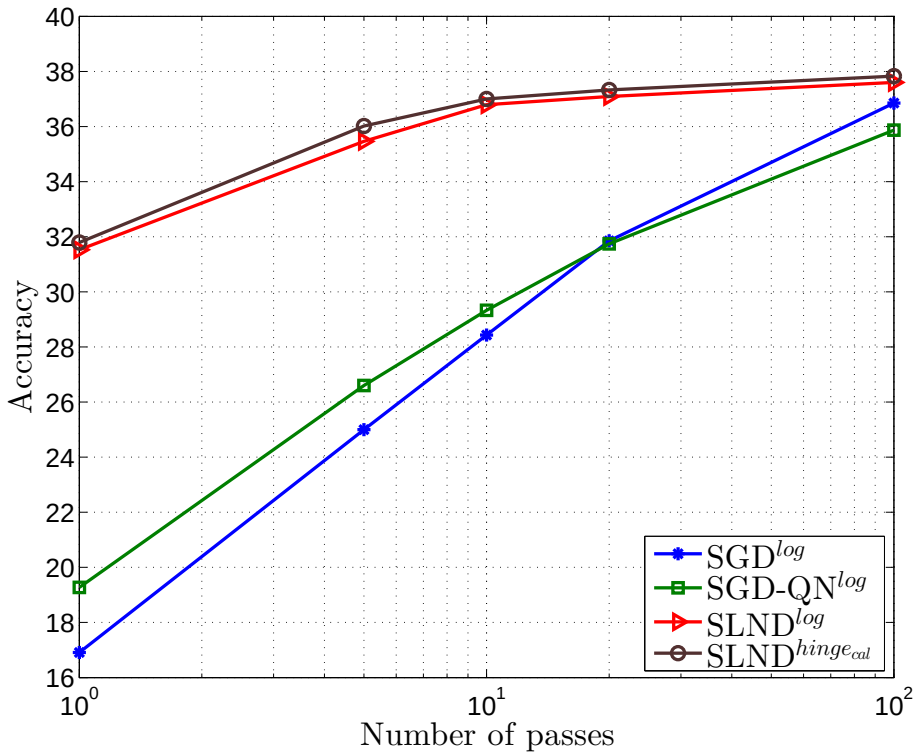


Figure 4.5: Top1 accuracies as a function of number of passes (iterations / skips) for SGD and SLND on Caltech256

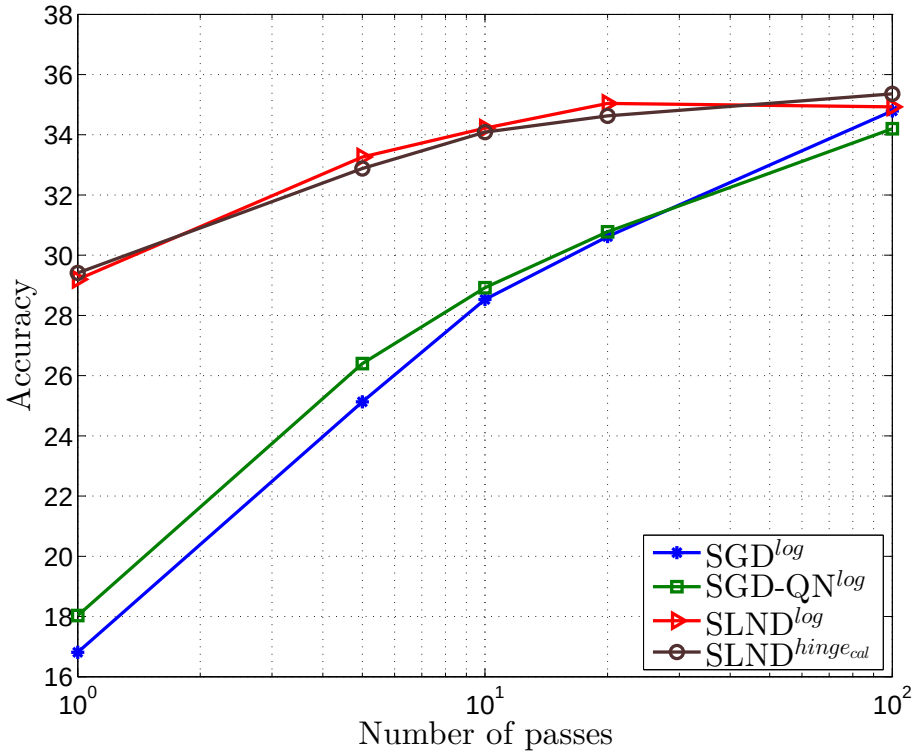


Figure 4.6: Top1 accuracies as a function of number of passes (iterations / skips) for SGD and SLND on SUN.

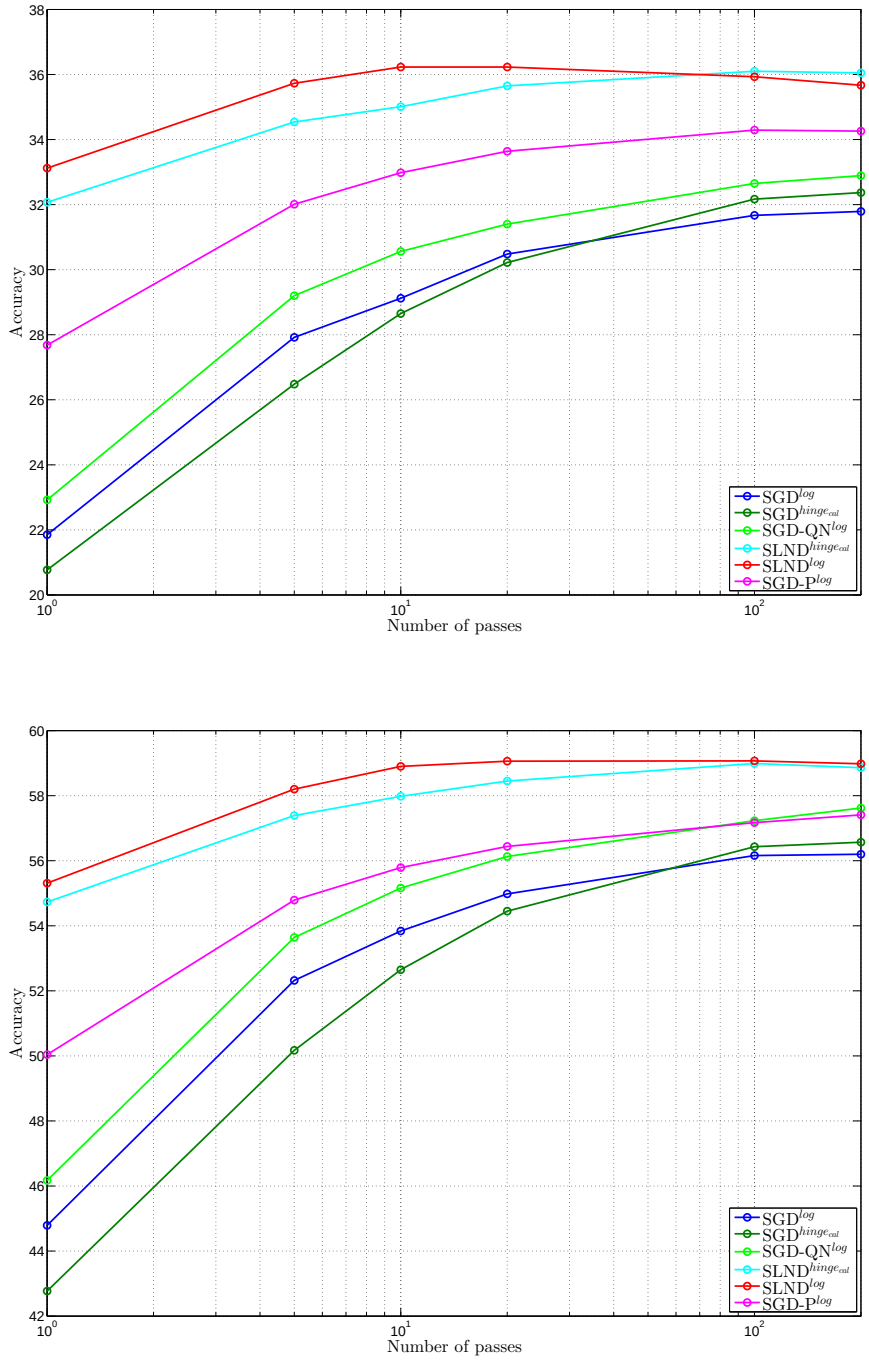


Figure 4.7: Accuracies as a function of number of passes for SGD and SLND on ImageNet. On top, the top-1 accuracy and at the bottom the top-5 accuracy.

Plots of convergence in Fig 4.7 on ImageNet shows again that SLND is faster

of a magnitude than classical SGD both for the top-1 accuracy and top-5 accuracy. SLND requires few iterations to converge: we only need *one* iteration (on 2000 samples) to get the same top-1 accuracy as SGD with 200 iterations. Moreover, we achieve top-1 accuracy equal to 36.23% (respectively top-5 accuracy equal to 59.06%) with 10 or 20 iterations of SLND, wich outperforms the best accuracies of SGD by 4% (respectively 3.5%) and SGD-QN by 3.3% (respectively 2.4%). Note that accuracy of SGD-QN is slightly better than SGD on ImageNet. We report also in Fig 4.7 on ImageNet results of SGD using preconditioning of the data (noted SGD-P) [LeCun et al. 1998]. Although preconditioning the data improves classical SGD, SLND still outperform all SGD methods. Training using SLND on ImageNet requires only one CPU hour. Training SGD for the same accuracy requires at least 20 CPU hours on a 2 X Intel Xeon E5-2687W 3,1GHz and 64 GB of RAM. Thus fast convergence of SLND results in sparse training set requirement well adapted for large scale image classification.

## 4.5 SLND Theoretical convergence analysis

### 4.5.1 Best rank $k$ approximation

We first show that $\mathcal{H}^*$, as computed in (4.13), is the best rank $k$ approximation of the inverse of $\mathcal{H}$ according to squared Frobenius norm.

**Lemma 4** *$\mathcal{H}^*$ satisfies:*

$$\mathcal{H}^* = \min_{\mathcal{H}' \in \mathbb{R}^{m \times m}, rank(\mathcal{H}')=k} \|\mathcal{I} - \mathcal{H}\mathcal{H}'\|_F^2 \tag{4.17}$$

**Proof:** We use the fact that $\mathcal{H} = \mathcal{P}\mathcal{D}\mathcal{P}^\top$, $\mathcal{P}\mathcal{P}^\top = \mathcal{I}$ and trace tr is cyclic invariant, and we have: $\|\mathcal{I} - \mathcal{H}\mathcal{H}'\|_F^2 = \mathrm{tr}((\mathcal{I} - \mathcal{H}\mathcal{H}')(\mathcal{I} - \mathcal{H}\mathcal{H}')) = \mathrm{tr}(\mathcal{P}\mathcal{P}^\top(\mathcal{I} - \mathcal{H}\mathcal{H}')\mathcal{P}\mathcal{P}^\top(\mathcal{I} - \mathcal{H}\mathcal{H}')) = \mathrm{tr}(\mathcal{P}^\top(\mathcal{I} - \mathcal{H}\mathcal{H}')\mathcal{P}\mathcal{P}^\top(\mathcal{I} - \mathcal{H}\mathcal{H}')\mathcal{P}) = \mathrm{tr}((\mathcal{I} - \mathcal{D}(\mathcal{P}^\top\mathcal{H}'\mathcal{P}))^2)$, out of which is comes that $\mathcal{P}^\top\mathcal{H}'\mathcal{P}$ is diagonal, and so $\mathcal{H}'$ can be diagonalized in the same basis as $\mathcal{H}$. Finally, to minimize the squared Frobenius norm, the non zero entries in its diagonal must equal the $k$ greatest non-zero entries in $\mathcal{D}$. □

### 4.5.2 A Weak Separability Assumption

We now prove a convergence result on SLND. For this objective, we define $p_{tj} \doteq -\mathrm{F}'(y_j \boldsymbol{w}_t^\top \boldsymbol{x}_j) \geq 0$ as a weight over the examples. For any classification calibrated loss, $-\mathrm{F}'$ is decreasing. Hence, weight $p_{tj}$ is all the *smaller* as example $j$ is all the *better* classified by $\boldsymbol{w}_t$. Intuitively, an example gets better classified as $y_j$ agrees with the sign of $\boldsymbol{w}_t^\top \boldsymbol{x}_j$ and the magnitude $|\boldsymbol{w}_t^\top \boldsymbol{x}_j|$ is large. We let $p_t \in \mathbb{R}^m$ be the vector of weights. We let $\boldsymbol{x}_j^\circ \doteq (\mathcal{P}_{|k}\sqrt{\mathcal{D}_{|k}^{-1}})^\top \boldsymbol{x}_j$ denote vector $\boldsymbol{x}_j$ expressed in the normalized eigenvectors' basis of $\mathcal{H}^*$ (4.13). Finally, we define $s_t \in \mathbb{R}^m$ as the vector whose coordinates are:

$$s_{tj} \doteq y_j \boldsymbol{x}_j^\top \mathcal{H}^* \boldsymbol{x}_{i_t} = y_j (\boldsymbol{x}_j^\circ)^\top \boldsymbol{x}_{i_t}^\circ \ , \forall j \ , \tag{4.18}$$

where example $i_t$ is the one chosen to update $\boldsymbol{w}_t$ in (4.16).

Our result relies on the following *Weak Separability Assumption*:

- (WSA) There exists $\gamma > 0$ a constant such that for any iteration $t$ in SLND,

$$\frac{p_t^\top s_t}{\|s_t\|_1} \geq \gamma \ . \tag{4.19}$$

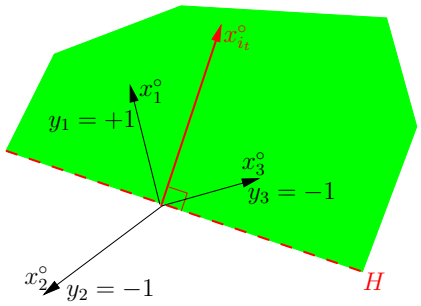


Figure 4.8: $\boldsymbol{x}^\circ_{i_t}$ is a better classifier for examples 1 and 2 ($s_{t1}, s_{t2} > 0$) than it is for example 3 ($s_{t3} < 0$).

To interpret WSA and see why it is indeed a Weak Separability Assumption, consider the interpretation of $s_t$ and assume $\boldsymbol{x}^\circ_{i_t}$ is used as a linear classifier. Then, $s_{tj} \geq 0$ iff the class $y_j$ agrees with the sign of this classifier, and it is all the larger as the classifier's output is large. On the other hand, $s_{tj} \leq 0$ iff the class $y_j$ disagrees with the sign of the classifier, and it is all the smaller as the classifier's output is large. Hence, $s_{tj}$ quantifies the goodness of fit of classifier $\boldsymbol{x}^\circ_{i_t}$ on $\boldsymbol{x}_j$ (see Figure 4.8). Thus, $p_t^\top s_t$ is a weighted average of this goodness of fit, in which weights are all the larger as examples have received a bad fitting so far by $\boldsymbol{w}_t$. Hence, WSA implies that $\boldsymbol{x}_{i_t}$ must contribute to classify better at least a small fraction of the examples with respect to $\boldsymbol{w}_t$. To see why it is "Weak", informally, picking $\boldsymbol{x}_{i_t}$ at random in any set satisfying mild constraints would make an expected value of $p_t^\top s_t$ equal to zero. So, we require the choice of $\boldsymbol{x}_{i_t}$ in SLND to beat a random linear classifier by at least a small amount. For the informed reader, the WSA parallels in our setting the popular weak learning assumptions in boosting algorithms [Freund & Schapire 1997].

### 4.5.3 Convergence theorem

The following Theorem shows that under the WSA, there exists a guaranteed decrease rate of the calibrated risk at each iteration, and this holds for whichever of the logistic and calibrated linear Hinge loss chosen to run SLND. The result would also hold for various other possible choices of classification calibrated losse, including the squared loss.

**Theorem 5** *Assume WSA is satisfied at each step of* SLND. *Then, for any* $\varepsilon \in (0,1)$ *there exists a value of* $\eta_t$ *in* $\Omega(1/m)$ *and* $O(1/\sqrt{m})$ *such that the following rate of decrease is guaranteed for the calibrated risk at hand:*

$$\varepsilon_{\mathrm{F}}(\boldsymbol{w}_{t+1}, \mathcal{S}) \quad \leq \quad \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}) - \frac{2\gamma^2\varepsilon(1-\varepsilon)}{m\mathrm{F}''(0)} \ , \forall t \ . \tag{4.20}$$

Since SLND is initialized with $w_0 = 0$, the null vector, to guarantee $\varepsilon_{\mathrm{F}}(\boldsymbol{w}_T, \mathcal{S}) \leq \mathrm{F}^\circ$ for any chosen real $\mathrm{F}^\circ \leq \mathrm{F}(0)$ such that $\mathrm{F}^\circ$ is in the image of F, it is enough to make

$$T \quad \geq \quad \frac{(\mathrm{F}(0) - \mathrm{F}^\circ)\mathrm{F}''(0)}{2\gamma^2\varepsilon(1-\varepsilon)} \times m = \Omega\left(\frac{m}{\gamma^2}\right)$$

iterations of SLND. In order not to laden the chapter's body, a proofsketch of the Theorem is provided in Appendix C. The proof exhibits and discusses the expression of $\eta_t$ which guarantees (4.20).

## 4.6 Conclusion

In this chapter we have proposed a new Stochastic Low Rank Newton descent algorithm (SLND) for the minimization of calibrated risk with linear complexity both in term number of samples and dimension of the features. SLND performs update of the current classifier with pseudo-inverses of the Hessian that are the most accurate low-rank approximations of the inverse according to Frobenius norm. We show the convergence of SLND using a Weak Separability Assumption which states that each example chosen to update the classifier must provide a weighted margin at least larger than some (possibly small) constant $\gamma > 0$. Under this weak assumption, SLND guarantees that its classifier has reached some fixed upperbound on the claibrated risk at hand after $\Omega(m/\gamma^2)$ iterations. No convergence rates are known to date for SGD-like approaches. Furthermore, the theory provides us with a set of working parameters for the experiments, including a step parameter $\eta_t$ typically in the order $\Omega(1/m)$.

We validate these theoretical properties by benchmarking it against state-of-the-art SGD algorithm on three challenging domains: Caltech256, SUN and ImageNet. The results on large scale image classification display that SLND improves significantly accuracy of the SGD baseline while being faster by orders of magnitude. Experiments also display that the parameters of SLND may be easily fixed and used from a domain onto another.

# Part III

# Bio-Inspired features for biological cells classification

Chapter 5

# Bio-Medical cells classification

## 5.1 Introduction

High-content cellular imaging is an emerging technology for studying many biomedical phenomena. Pathologists establish their diagnostics by studying tissue sections, blood samples or punctures. Related cellular image analysis generally requires to classify many cells according to their morphological aspect, staining intensity, subcellular localization and other parameters. In general, samples are stained with various dyes to visualize cell cytoplasm and nucleus. In addition, immunohistochemistry is used to study specific protein expression. Using these approaches, pathologists observe tissue damage or cell dysfunction like for example, inflammation, neoplasia or necrosis. Abnormal nuclei allow determining cancer grades. Pathologists recognize aberrant shapes of whole cells, organelles, nuclei or staining allowing the classification of the cells. Classical quantification is based on visual counting. New powerful fully motorized microscopes are now able to produce thousands of multiparametric images for several experimental conditions. Consequently, large numbers of cell images have to be analysed. Such analysis by one (or several) experimenter is time-consuming and above all poorly reproducible. In fact, humans are limited in their ability to classify due to the huge amount of image data. Visual counting is consequently performed on a small portion of the sample. A Computer Aided Diagnosis (CAD) system will allow reliable quantification and therefore be a precious tool in diagnostics.

In this chapter we present an application of UNN algorithm to biological cellular image classification. First we introduce our specific bio-inspired descriptors, using contrast information distributions on the already segmented cells: a region based descriptor that shows its efficiency to describe cellular images. Those bio-inspired features (BIF) are sometimes more than 10% more accurate than standard descriptors for such images. Then, we report two biological applications of cells classification using BIF descriptor.

## 5.2 Region based bio-inspired descriptor

For better understanding the image content, it can be useful to get inspiration from the way our visual system operates to analyze the scene. The first transformation undergone by a visual input is performed by the retina.

In fact, ganglion cells, that are the final output of the retina, are first simulated by the local changes of the *illumination*. This information is captured by their

receptive fields and transformed to *luminance contrast* intensities. Those receptive fields are like center-surround models (see Fig. 5.1). They react to the illumination

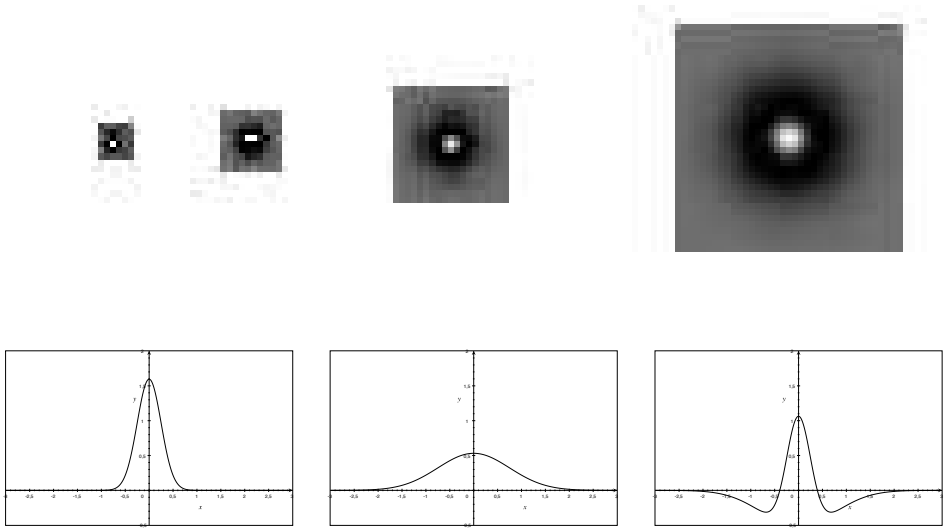

Figure 5.1: Top, receptive fields in the retina modeled by DoGs for 4 scales. Bellow, the model of the response of those retinal cells.

of either the center or the surround of the ganglion cells and are disabled when illuminating the other one. Such behavior, similar to an edge detector, is modeled by a centered two-dimensional *Difference of Gaussians* (5.1).

$$DoG_\sigma(x,y) = G_\sigma(x,y) - G_{\alpha\cdot\sigma}(x,y) \tag{5.1}$$

Moreover, ganglion cells react to the luminance in different scales, thus adding multiscale aspect and allowing us to use DoG filters in a scale space (Fig. 5.1).

The basic idea is to compute features inspired from the visual system model and specially from the main characteristics of the retina processing. Such was the case in [Bel haj ali et al. 2011], where we represented the image using features based on *contrast* information on square blocs.

Such descriptor is well adapted in the case of our cells images since the most discriminative visual feature between categories is the *luminance contrast* in subcellular regions. Thus, we define cell descriptors based on the *local contrast* in the cell, that we call Bio-Inspired Features, BIF. The *local contrast* is obtained by a filtering with *Differences of Gaussians* (DoGs) centered at the origin. So that the contrast $C_{Im}$ for each position $(x, y)$ and a given scale $s$ in the image $Im$ is as follows:

$$C_{Im}(x,y,s) = \sum_i \sum_j (Im(i+x, j+y) \cdot DoG_{\sigma(s)}(i,j)) \tag{5.2}$$

We use the DoG described by [Field 1994] where the larger Gaussian has three times the standard deviation of the smaller one. After computing these contrast

coefficients in (5.2), we apply a non-linear bounded transfer function, named neuron *firing rates*, used in [Van Rullen & Thorpe 2001]. This function is written as:

$$R(C) = G \cdot C/(1 + Ref \cdot G \cdot C), \tag{5.3}$$

where $G$ is named the contrast gain and $Ref$ is known as the refractory period, a time interval during which a neuron cell *reacts*. The values of those two parameters proposed in [Van Rullen & Thorpe 2001] to best approximate the retinal system are $G = 2000\,Hz \cdot contrast^{-1}$ and $Ref = 0.005\,s$.

Firing rate coefficients $R(C)$ are encoded on an already segmented cell region. Then, they are quantified into normalized $\mathcal{L}1$ histograms of $n$-bins for each scale and finally concatenated. Thus our global descriptor's dimension is a multiple of $n$.

Note that state of the art classical methods such as SIFT descriptors encode gradient directions on square blocks [Lowe 2004]. and Gist features encode average energies of filters coefficients on square blocks too [Oliva & Torralba 2001].

## 5.3 Application to the localization of NIS protein in the cells of the thyroid gland

In the present work, we perform cellular image classification to study the pathways that regulate plasma membrane localization of the sodium iodide symporter (NIS for Natrium Iodide Symporter). Those biological experiments are part of the research project of TIRO team from the faculty of medecine of Nice. NIS is the key protein responsible for the transport and concentration of iodide from the blood into the thyroid gland. NIS-mediated iodide uptake requires its plasma membrane localization that is finely controlled by poorly known mechanisms. For decades, the NIS-mediated iodide accumulation observed in thyrocytes has been a useful tool for the diagnosis (thyroid scintiscan) and treatment (radiotherapy) of various thyroid diseases. Improvements in radioablation therapy might result from promoting targeting of NIS to the plasma membrane in the majority of thyroid cancers or metastases. NIS has also been described as a promissing therapeutic transgene promoting metabolic radiotherapy (*i.e.*, 131I uptake by cancer cells ectopically-expressing NIS) in many different studies. An important improvement of this approach should benefit from a better understanding of the post-transcriptional regulation of NIS targeting to the plasma membrane, Previously, we observed that mouse NIS catalyses higher levels of iodide accumulation in transfected cells compared to its human homologue. We showed that this phenomenon was due to the higher density of the murine protein at the plasma membrane. To reach this conclusion, biologists classified several hundreds of cells [Dayem et al. 2008]. We have also demonstrated, using a set of monoclonal antibodies, that human NIS is not expressed intracellularly in thyroid and breast cancer [Peyrottes et al. 2009], as was proposed by other groups. The team of biologists is now focussing on the analysis of NIS phosphorylation that most probably plays an important role in the post-transcriptional regulation of the NIS. Using site-directed mutagenesis of previously-identified consensus sites,

we have recently shown that direct phosphorylation of NIS alters NIS targeting to the plasma membrane, as well as NIS recycling, causing retention of the protein in intracellular compartments such as the Golgi apparatus, the endoplasmic reticulum or the early endosomes. We used a high-content cellular imaging to study the impact of the mutation of several putative phosphorylation sites on the subcellular distribution of the protein.

### 5.3.1 Experiments settings

In our experiments, expert biologists individually expressed different NIS proteins mutated for putative sites of phosphorylation. The effect on the protein localization of each mutation was studied after immunostaining using anti-NIS antibodies as described in [Dayem et al. 2008]. Immunocytolocalization analysis revealed mainly two cell types with different subcellular distributions of NIS: at the plasma membrane or in intracellular compartment (mainly endoplasmic reticulum) which we will refer to by *Mb*; throughout the cytoplasm (with an extensive expression) which we will call *ER*. An example of *Mb* and *ER* cells are shown respectively in Figures 5.2(a) and 5.2(b).

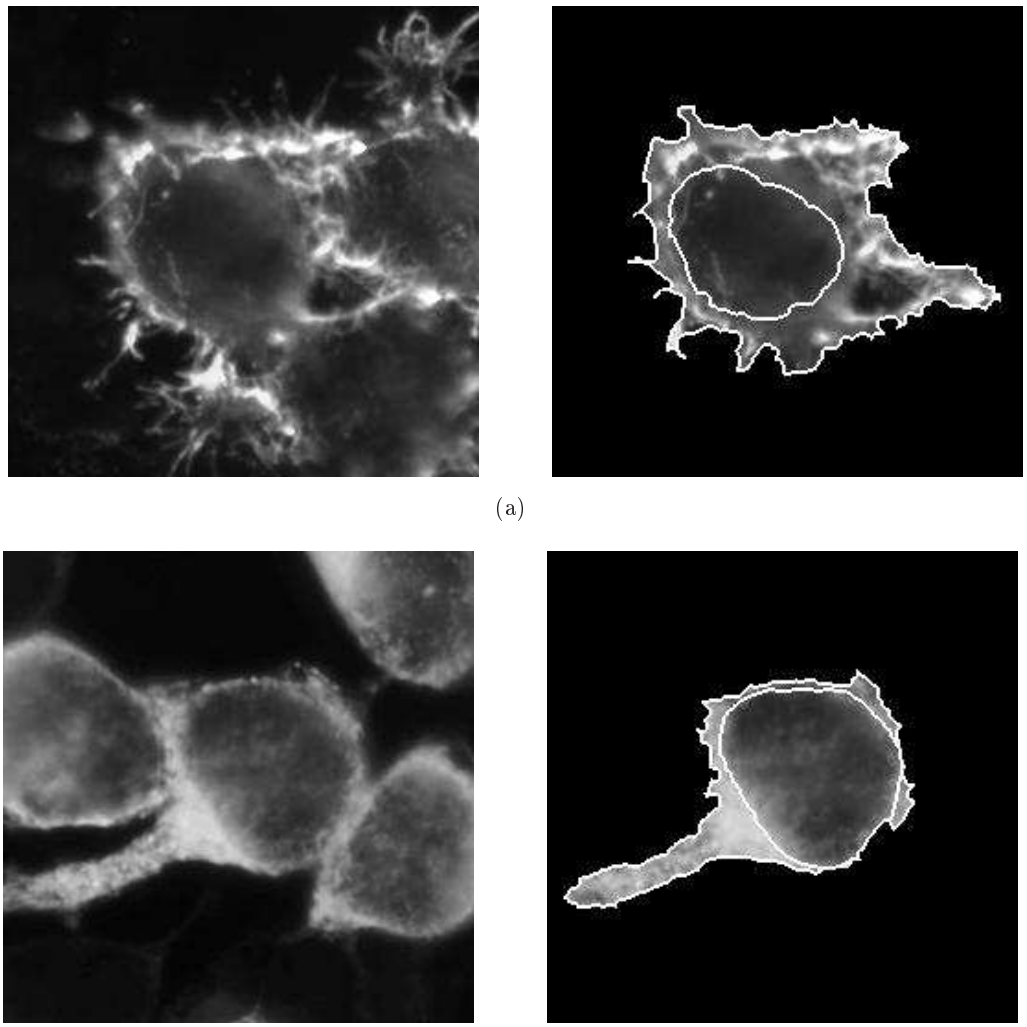

Figure 5.2: An *Mb* (a) and an *ER* (b) extracted cells and their two segmented regions of interest.

The goal of such experiments is to establish statistics on the different mutations of cells. Our application aims to assign automatically for each cell one of the previously numbered patterns according to its staining aspect. The approach is depicted into two main steps: image segmentation to seperate cells, extracting descriptors and classification task.

### 5.3.2 Cells detection and segmentation

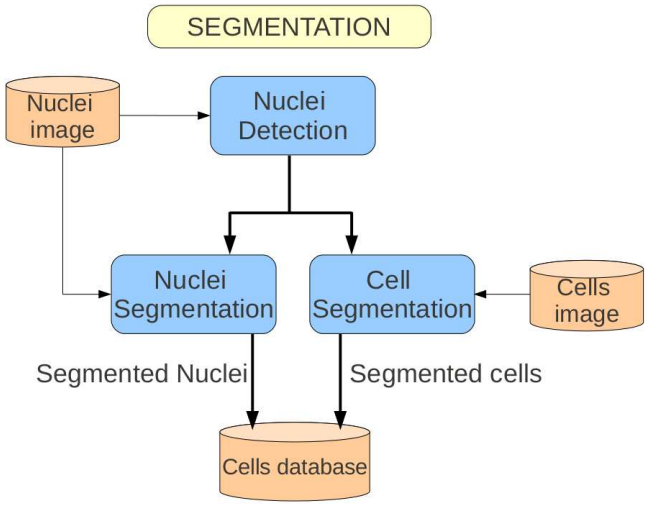


Figure 5.3: Block diagram of the proposed method for automatic cell segmentation.

The first step is a pre-processing segmentation of cells from the main microscopic images. The database consist of two distinct parametric fluorescence images. The first one, called *nucleus image*, shows the nucleus and the second called *global image*, shows the staining of the protein. The two images are only two different acquisitions (with two different wavelength) of the same experiment. We consider images at 40-fold magnification that was acquired by means of a fully fluorescence microscope (Zeiss Axio Observer Z1) coupled to a monochrome digital camera (Photometrics cascade II camera). We note that nuclei images are used for the only purpose to help to segment global cells. But never used for feature extraction. The information that is used to define classes is the staining aspect in the global cells images. Nuclei are identified from the nuclei image and used as a prior for whole cell segmentation of the *global image*. An example of nuclei and global cells segmentation result are given in Figures 5.4 and 5.5.

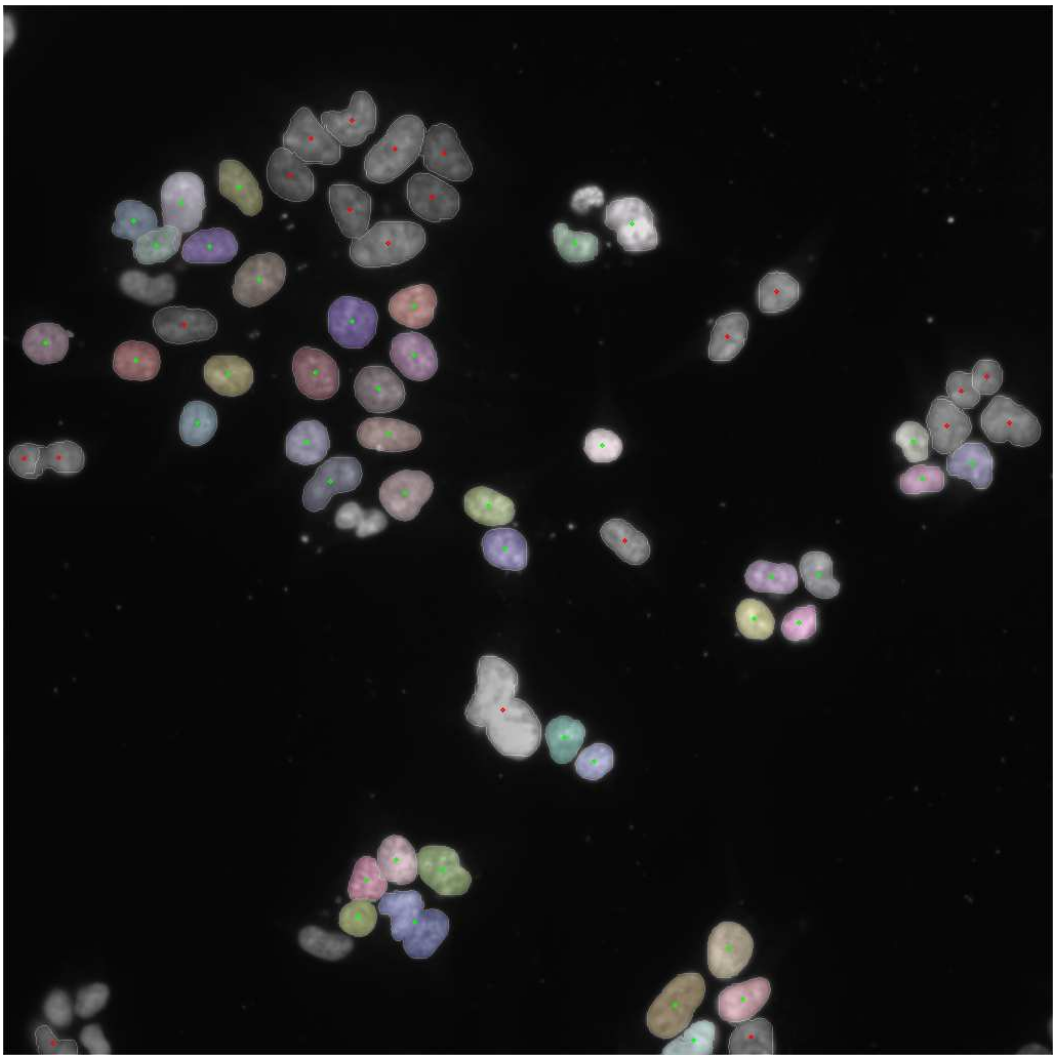

Figure 5.4: An example of nuclei segmentation. Each nucleus is identified with a different color. The green point shows the nucleus center.

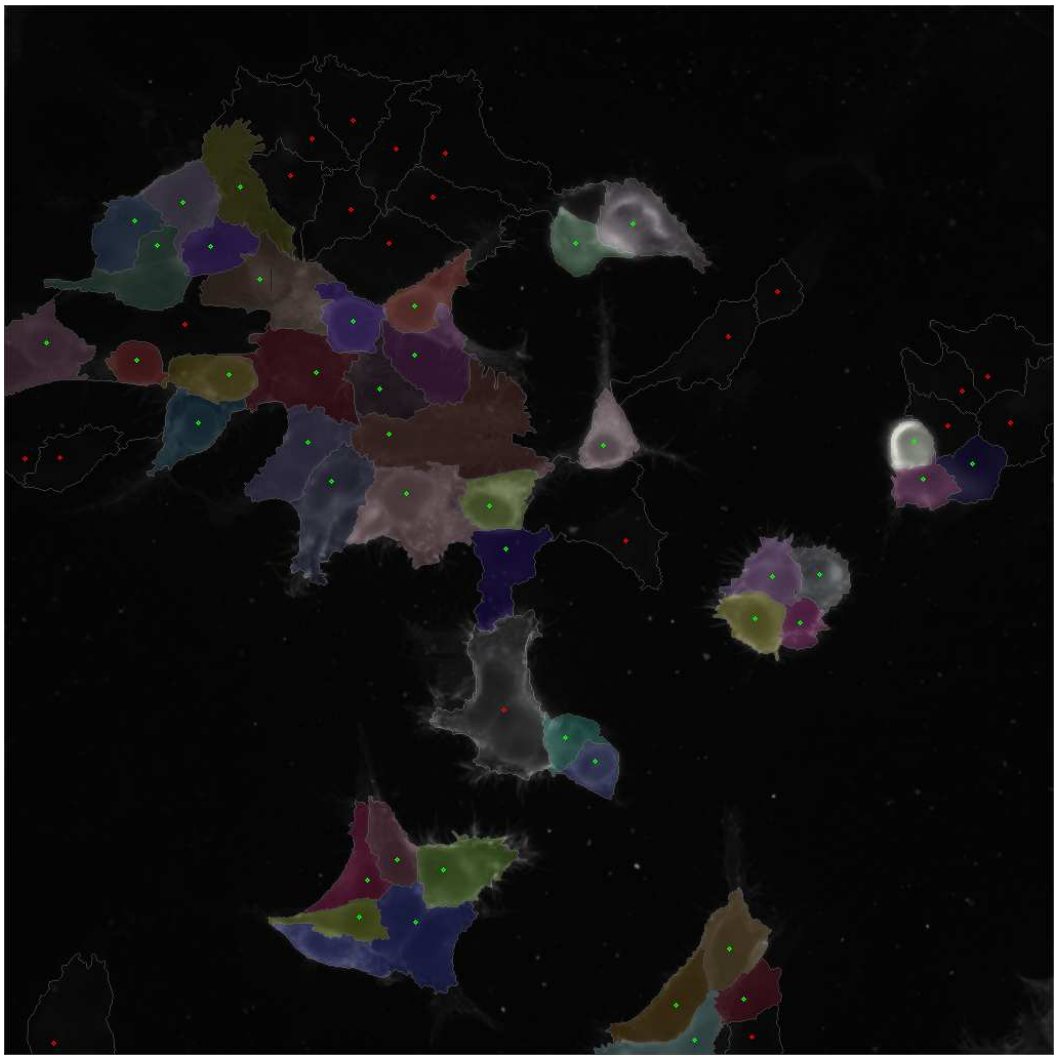

Figure 5.5: An example of cells segmentation. Each cell is marked with a different color.

The segmentation process is described in the diagram of Figure 5.3. In fact, nuclei locations are detected by the mean of morphological operators and used to segment nuclei and get their masks.
Those latters are then used as markers to segment the global cells. The output of the segmentation step corresponds to three images for each cell: a sub-image that bounds the cell, a binary image for cell's mask and a second binary image for nucleus mask.

### 5.3.3 Features and classification

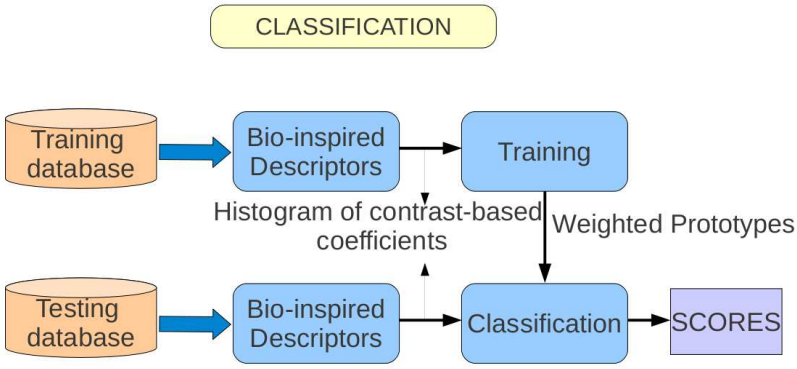


Figure 5.6: Block diagram of the proposed method for automatic cell classification: descriptor extraction and classification process.

Once cells are segmented, we apply our classification method (see diagram 5.6); First we compute bio-inspired region descriptors, extracting contrast-based features for each of the segmented cells. These descriptors are then used in a supervised learning framework to define the classifier to be used to predict the class of unlabeled cells.

For this purpose, we collected 489 cell images of such biological experiments and manually annotated them according to three classes, that are denoted in the following as *Mb* (389 cells), *ER* (100 cells) and Round (8 cells) (dead cells).

Since round cells are very easy to classify (very high contrast everywhere in the cell), we focus on the two category classification: Membrane (*Mb*) and *ER*.

To extract our descriptors, we use *masks* on cell images on which we encode firing rate coefficients (5.3): according to the visual aspect of cells, we split each cell into two *regions of interest* (see figure 5.2), corresponding to nucleus and external part, by using previously computed masks (the external region is the remaining of the substitution of the nucleus mask from the global one). For both of the two regions, firing rate coefficients are quantified into normalized $\mathcal{L}1$ histograms of 32-bins then concatinated, thus giving our global descriptor with a dimension equal to 64 per scale.

An important parameter for our bio-inspired descriptors is the scale on which we compute the local contrast. In fact, the standard deviations of the DoG are dependant of this parameter as follows: $\sigma 1 = 0.5 \cdot 2^{scale-1}$ and $\sigma 2 = 3 \cdot \sigma 1$. We made a cross validation on 100 experiments to choose the most relevant scale parameter. According to those evaluations, next experiments are performed using $scale = 5$ for descriptor extraction.

Once we get descriptors of all the cells in the database, we ran our UNN algorithm by training on 50% of the images, while testing on the remaining 50%. In order to get robust performance estimation, we repeated the evaluation 100 times

over different random training/testing folds. Note that we used a fast and effecient tool for the k-NN search provided in the Yael toolbox[1].

Our classification algorithm $\text{UNN}_s$ was evaluated in a first step using a *uniform* regularization by the mean of the parameter $\gamma$ that compensates the class imbalance. In a second step, we focused in an *adaptive* regularization according to majority and minority classes and we denote this approach by $\text{UNN}_{s\_adaptive}$. This approach allows to have automatically a balance number of trained prototypes per class (see Tab. 5.1) and visibly decrease misclassification.

| | $N_t$ | $N_{Mb}$ | $N_{ER}$ |
|---|---|---|---|
| $\text{UNN}_s$ | 69.24% | 50.20% | 19.03% |
| $\text{UNN}_{s\_adaptive}$ | 47.69% | 28.58% | 19.11% |

Table 5.1: This table shows the percentage of prototypes number selected from the training set by both $\text{UNN}_s$ and $\text{UNN}_{s\_adaptive}$ : We report the total number ($N_t$), the one in the class *Mb* ($N_{Mb}$), and in the class *ER* ($N_{ER}$). The distribution of selected prototypes on both classes is more balanced using $\text{UNN}_{s\_adaptive}$.

| | mAP | | AP for Mb | | AP for ER | |
|---|---|---|---|---|---|---|
| | $\mu(mAP)$ | $\sigma(mAP)$ | $\mu(AP)$ | $\sigma(AP)$ | $\mu(AP)$ | $\sigma(AP)$ |
| k-NN | 84.22 | 2.56 | 94.81 | 2.02 | 73.64 | 5.63 |
| $\text{UNN}_s$ | 86.04 | 2.54 | 94.48 | 1.90 | 77.60 | 5.46 |
| $\text{UNN}_{s\_adaptive}$ | 87.67 | 1.93 | 89.27 | 2.26 | 86.08 | 3.78 |
| SVM | 76.46 | 4.55 | 95.58 | 2.38 | 57.34 | 10.67 |

Table 5.2: Global average precision (MAP), average precision for *Mb* and average precision for *ER* for different classifiers.

We report the average classification results and the classification rate of each class in Tab. 5.2. Remark that we achieve a mean average precision (MAP) greater than 87.5% when using $\text{UNN}_{s\_adaptive}$, which is a very promising result for our cell descriptor and classification method. Our classification approach improves the MAP of the k-NN classifier of more 3% and the SVM with gaussian kernel of more than 11%. Moreover some misclassification arises on the minority class (*ER*) using k-NN , thus giving an average precision (AP) of about 73% (see Tab. 5.2). Using $\text{UNN}_{s\_adaptive}$ classification improved MAP of the minority class up to 86% thus 13% better than k-NN. For the SVM classification, the result in Tab. 5.2 shows that there is an important classification error on *ER* cells where the AP is about only 57%.

[1]Source code can be downloaded in the following link: https://gforge.inria.fr/projects/yael

## 5.4 Application to Immuno-Fluorescence cells

In autoimmune diseases, targets of autoantibodies are characterized by indirect Immunofluorescence (IIF) on human cultured cells. Then, stained compartments of cells are identified by experts.

In this context, we evaluate our BIF features and our UNN classification on the HEp-2 Cells dataset [Foggia et al. 2010] provided by University of Salerno and Campus Bio-Medico of Roma[2]. This database contains 721 images divided into *six* categories as shown in Fig. 5.7.

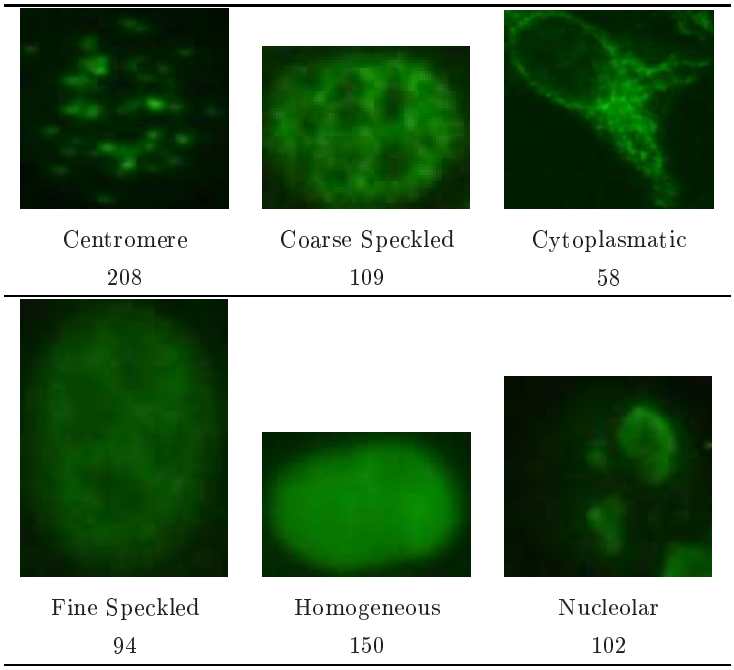


Figure 5.7: Sample images and the number of elements for each category in the dataset.

Cells are already segmented (manually) and both hole images and their corresponding masks are provided in the dataset.

In a first step, we extract *Bio-Inspired* features for each *manually* segmented cell according to the cell mask. This version of our feature will be denoted as $\text{BIF}^s$. In a second experiment, we extracted BIF on the whole image of the cell (without segmentation) to test the robustness of those features. We will refer to this version by $\text{BIF}^a$. To better adjust some parameters, such as the dimension, we performed a cross validation system on the number of scales and the number of quantification bins, and we choose using 4 scales with a number of bins equal to 256. Our global features are the concatenation of histograms of 256-bins for each scale. The final dimension of descriptors is then equal to $4 \times 256$. We compare our approach to the state of the art SIFT descriptor. We use classical Bag-of-

[2]Data available at: http://mivia.unisa.it/hep2contest/index.shtml

Features [Sivic & Zisserman 2006], with the same dimension 1024, on the dense SIFT provided by [Vedaldi & Fulkerson 2008] which encode gradient directions on a grid of small square blocks of the cellular image.

| | $\text{UNN}_{exp}$ | $\text{UNN}_{log}$ | $\text{UNN}_{mat}$ |
|---|---|---|---|
| Accuracy | 96.16 | 95.46 | 94.72 |
| AUC | 96.32 | 95.78 | 95.25 |

Table 5.3: Classification results using the $\text{BIF}^s$ descriptor for the three proposed versions of UNN. The first row indicates the True Positive rate or accuracy, and the second one is about the Area Under the roc Curve (AUC).

For the classification task we performed cross validations on 10 random folds. Each fold corresponds to a random split of the dataset such that we train on 50% of the images, while testing on the remaining ones. We evaluated the different versions of UNN in Tab.5.3.

| | UNN | | | SVM | | |
|---|---|---|---|---|---|---|
| | $\text{BIF}^a$ | $\text{BIF}^s$ | SIFT | $\text{BIF}^a$ | $\text{BIF}^s$ | SIFT |
| Centromere | **96.05** | 96.15 | 85.00 | **97.01** | 97.40 | 88.07 |
| Coarse Speckled | **99.62** | 97.59 | 69.81 | **95.00** | 97.03 | 71.29 |
| Cytoplasmatic | **100.0** | 100.0 | 99.65 | **100.0** | 100.0 | 97.93 |
| Fine Speckled | **93.82** | 95.95 | 61.27 | **94.25** | 94.46 | 58.93 |
| Homogeneous | **90.26** | 91.20 | 91.86 | **93.46** | 94.00 | 88.93 |
| Nucleolar | **97.45** | 96.07 | 87.25 | **97.64** | 97.45 | 88.03 |
| average Accuracy | **96.20** | 96.16 | 82.47 | **96.23** | 96.72 | 82.20 |

Table 5.4: Evaluations of UNN and SVM using both $\text{BIF}^a$(on whole images), $\text{BIF}^s$ (on *manually* segmented cells) and SIFT Bag-of-features. Here, we give the Accuracy for each class. The last row shows the average Accuracy. The best performance for each category is given in blue and the second one in green.

We compared performances of $\text{UNN}_{exp}$with those of standard SVM, using both BIF and SIFT Bags-of-features (see Tables 5.4 and 5.5). The reported results of UNN refer to setting $k = 10$ for both training and testing. This value refers to the best performances according to a cross validation on the training set. The same experiment was performed to choose the parameters for the *gaussian* SVM. Note that for the k-NN search we used the same fast and efficient software as previously. For BIF descriptor we report experiments on both $\text{BIF}^s$ and $\text{BIF}^a$ versions of our features. Although $\text{BIF}^a$ version performs similar results to $\text{BIF}^s$ version, the comparison with SIFT Bags-of-features becomes fair enough to conclude that Bio-Inspired Features are more adapted to such images. In fact, results on tables 5.4 and 5.5 display the high discriminative ability of the proposed Bio-Inspired Feature, which allows for classification precision generally larger than 90%, up to

| | UNN | | | SVM | | |
|---|---|---|---|---|---|---|
| | BIF$^a$ | BIF$^s$ | SIFT | BIF$^a$ | BIF$^s$ | SIFT |
| Centromere | **95.48** | 95.68 | 92.63 | **97.86** | 96.62 | 92.03 |
| Coarse Speckled | **98.54** | 97.24 | 86.70 | **94.23** | 95.40 | 79.00 |
| Cytoplasmatic | **99.64** | 99.73 | 97.82 | **99.39** | 99.02 | 93.15 |
| Fine Speckled | **93.54** | 95.61 | 63.35 | **89.56** | 91.82 | 59.26 |
| Homogeneous | **93.42** | 94.79 | 91.06 | **97.04** | 97.78 | 91.39 |
| Nucleolar | **97.74** | 94.89 | 92.35 | **94.94** | 98.66 | 92.59 |
| average AUC | **96.39** | 96.32 | 87.32 | **95.50** | 96.55 | 84.57 |

Table 5.5: Evaluations of UNN and SVM using both BIF$^a$(on whole images), BIF$^s$ (on *manually* segmented cells) and SIFT Bag-of-features. Here we present the Area Under the roc Curve (AUC) for each class. The last row shows the average AUC. The best performance for each category is given in blue and the second one in green.

almost 100% (on the "Coarse Speckled" and "Cytoplasmatic" classes). In addition, the precision obtained with such specific descriptor outperforms the standard SIFT bag-of-features by at least 14% in terms of True Positive rate (TP rate) and 9% in terms of Area Under the roc Curve (AUC). Furthermore, the most interesting results are those obtained using BIF$^a$, since in real cases an automatic segmentation process on cellular images is poorly reproducible. Those results (columns in bold in tables 5.4 and 5.5) shows not only the efficiency of the feature but also the precision of our UNN algorithm which remains relevant (in terms of TP rate and AUC), comparable and even better than state-of-the-art SVM. For instance, notice the improvement of UNN over SVM on the "Coarse Speckled" class (4.5% of gap), while SVM is the best performing method on the "Homogeneous" class (3% of gap).

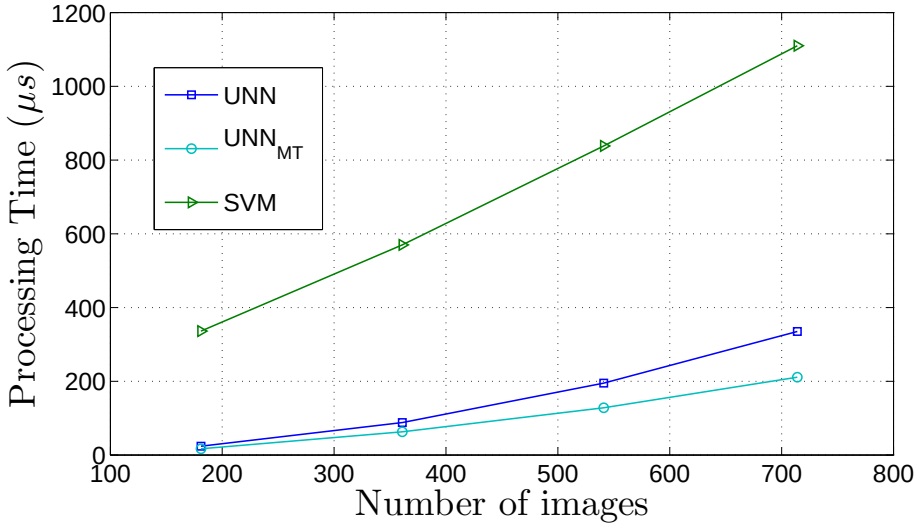


Figure 5.8: Processing time of the training step for both UNN, SVM and multi-thread version of UNN.

Besides comparing very favorably with state-of-the-art approaches, our UNN method enables much faster classification. Fig. 5.8, shows typical processing time for UNN and SVM and UNN achieves speedups of roughly 3 to 5 over SVM. UNN benefits from straightforward multi-thread implementation (UNN$_{MT}$) in addition to the fast k-NN search algorithm. This makes the processing furthermore faster. Therefore our Bio-Inspired UNN algorithm provides the best Precision/Time trade-off.

## 5.5 Conclusion

As a first application in this chapter, we have presented a novel algorithm for automatic segmentation and classification of cellular images based on different subcellular distributions of the NIS protein. First of all, our method relies on extracting highly discriminative descriptors based on bio-inspired histograms of Difference-of-Gaussians (DoG) coefficients on cellular regions. Then, we applied UNN algorithm for learning the most relevant prototypical samples that are to be used for predicting the class of unlabeled cellular images. We notice that this application is currently being integrated in a software designed for biological cells identification. A second application, that deals with immunofluorescence cellular imaging, was presented in this chapter. We used the same algorithm UNN to evaluate our experiments on an unbalanced dataset of cells that were manually segmented. Although being the very early results of our methodology for such a challenging application, performances are really satisfactory (average global precision of 87.5% and MAP of the minority

class up to 86%) and suggest our approach as a valuable decision-support tool in cellular imaging.

Chapter 6

# General conclusion

In this thesis, we deal with a specific supervised learning scheme for image classification based on a set of calibrated surrogates. In this context, we designed three three learning algorithms for different kind of classifiers. The first one is a generalization and an optimization of a leveraged k-NN algorithm, UNN. This latter is based on learning voting weigths in a boosting framework using the minimization of our classification criterion. In fact, we enlarge the set of losses often used in boosting and restricted to the singleton associated to the exponential loss to a more generalized set containing the logistic and matsushita losses. The UNN algorithm shows high performances in competitive computation times.

The second algorithm, $N^3$ is a Newton-Raphson approach for boosting k-NN voting weights. We prove that our $N^3$ method has consistent convergence properties within the set of considered losses and provide several interesting statistical properties like the estimation of posteriors of the classification. In the experimental standpoint, this algorithm shows a fast convergence on quite large datasets of real images like the SUN and Caltech256. Furthermore, $N^3$ shows that it is possible to cope with k-NN's curse of dimentionality. In fact, based on the posteriors of the classification, we use $N^3$ in a low memory divide and conquer method.

The third algorithm is a novel approach based on stochastic low rank newton descent, SLND for linear classifiers. It consists on the minimization of calibrated losses using a Newton update of the classifier. The Newton update, known by its fast convergence, becomes a complex problem in high dimentional features space. We present in this work an approximation that overcome this complexity. In addition, experiments on very large datasets show the high performances of SLND that outperform the state of the art methods.

This work, presents at the end, an interesting application to biomedical cells classification. For this purpose, we designed a bio-inspired descriptor, based on histograms of contrast, that are well adapted for those microscopic cellular images. Testing UNN algorithm for such applications shows promising high performances.

# Part IV

# Appendices

Appendix A

# UNN optimization with metric learning

## A.1 Introduction

A study of the accuracy of metric learning algorithms [Nielsen & Sérandour 2009] has compared some methods, and has shown that those metrics are mostly dependent on the data type. Several researches were concentrated on Mahalanobis distance aspect like [Davis et al. 2007] who tend to define a distance given constraints on training set and boundaries between similarity and dissimilarity. This can be seen as a projection in a new space that fits with an a priori knowledge on categories.

In this appendix, we present an optimization approach tested on UNN algorithm. First, we include metric learning process introduced by [Davis et al. 2007] to adapt distances between features. This latter replaces the $L1/L2$ norm used for the k-NN search. Metric choice in the context of NN classifiers could be a critical problem [Guillaumin et al. 2009] in the way that a wrong choice can lead to the failure of the classification method. Then, we evaluate this approach on Gist features [Oliva & Torralba 2001, Oliva & Torralba 2006] reduced in the space of their principal components. In fact, global descriptors like Gist are well appropriate for classification tasks. However, those descriptors are usually high dimensional and therefore costly in similarity measuring.

In a first section, we detail the proposed approach. In the second one, we explain parameter settings and expose evaluation results of the different steps.

## A.2 Proposed approach

### A.2.1 Descriptor and dimension reduction

Global features like Bag-Of-Features (BOF) or Visual Words are often used for categorization because they represent the global content of an image. Therefore, this point makes descriptors very high dimensional. In this paper we use the same Gist global features as in [Oliva & Torralba 2001] to have comparable classification results[1].

For the method we use, Gist are extracted on $4\times4$ subregions of 4 scales from gray images, and considering the 8 dominant directions. Thus we have image descriptors

[1] http://people.csail.mit.edu/torralba/code/spatialenvelope/

of 512 dimensions. As we will deal with metric learning later, we need to reduce the features dimension in first step.

We studied the effect of principal component analysis (PCA) on Gist, and we noticed that a reduction up to 16 and even 32 times of the original dimension does not affect the classification rate. Results will be presented later in experiments section (A.3.3).

Throughout the following work, we use Gist with a 32 dimension where the principal components were already computed on training dataset.

### A.2.2 Metric learning

For classification task, partial information provided by real scene images may be misleadingly. In fact, in high dimensional feature spaces, image descriptors may be, in the $L^2$ sense, similar within different categories and dissimilar under the same one. For example, points that are near the class border or that are in an area of overlap with other classes are constrained to be metrically similar but semantically not.

In this case, we use metric learning to adjust the similarity measure so that it increases inter-class variability and decreases intra-class one.

In [Davis et al. 2007], authors propose an Information-Theoretic Metric Learning (ITML) approach that generalize the Mahalanobis distance. This metric considers pairs of similar and dissimilar points, and trains a matrix $A$ to build a distance function that will make close elements in the same class and far those in distinct ones. This distance for a given couple of points $(x_i, x_j)$ is expressed in (A.1).

$$d_A(x_i, x_j) = (x_i - x_j)^T A (x_i - x_j) \tag{A.1}$$

The approach is an iterative algorithm that tends to approximate a positive definite matrix $A$ using a minimization under constraints task.

$$\min_A KL(p(x; A_0) \| p(x; A)) \tag{A.2}$$

subject to

$$d_A(x_i, x_j) \leq u \quad (i,j) \in S, \tag{A.3}$$

$$d_A(x_i, x_j) \geq l \quad (i,j) \in D. \tag{A.4}$$

where $KL$, Kullback Leibler, is a Bregman divergence (statistical distance between distributions). $S$ and $D$ are sets of similar and dissimilar pairs and $u$ and $l$ denotes threshold distances between points respectively in $S$ and $D$. An a priori knowledge of some parameters is needed for learning process. For the algorithm version we use, a constraint matrix $c$ is considered for this process, and the problem is formulated like it follows:

$$\min_A D_{ld}(A, A_0) + \gamma \cdot D_{ld}(diag(\xi), diag(\xi_0)) \tag{A.5}$$

subject to

$$d_A(x_i, x_j) \leq \xi_{c(i,j)} \quad (i,j) \in S, \tag{A.6}$$

$$d_A(x_i, x_j) \geq \xi_{c(i,j)} \quad (i,j) \in D. \tag{A.7}$$

where $\xi$ is threshold matrix for similarity and dissimilarity, $c(i,j)$ is the index of the constraint corresponding to the pair $(i,j)$, $\gamma$ controls the tradeoff between satisfying the constraints and minimizing the LogDet divergence between $A$ and $A_0$: $D_{ld}(A, A_0)$, which was induced from

$$KL(p(x; A_0) \| p(x; A)) = \frac{1}{2} D_{ld}(A, A_0) \tag{A.8}$$

## A.3 Experiments

### A.3.1 Dataset

For our experiments we use the database proposed in [Oliva & Torralba 2001], composed of outdoor natural scenes divided into the following categories: coast, mountain, forest, open country, street, inside city, tall buildings and highways. This base includes 2688 color images with $256 \times 256$ pixels.


Figure A.1: Natural scenes from the outdoor database of Torralba.

### A.3.2 Settings

For experiments, we need two separate datasets: the first one for train and the second for tests. We divide our database so in a random way. For the results presented here, we use 2000 images for train (of about 250 images per category) and 688 as queries. For evaluations in (A.3.4) and (A.3.5), Gist features are used here in 32 dimensions.

We evaluate UNN against the standard k-NN method considering different numbers of prototypes (classifiers). For the k-NN, it is trivial that we should consider all training set as classifiers to be as robust as possible. However, our goal in this paper is to optimize classification taking into consideration scalability rule. Hence, sets of prototypes $P \subset S$ tested on here contains respectively 10, 20, 30, 40, or

50 percent of train set $S$. For k-NN, prototypes are chosen randomly, whether for UNN classifiers are those of highest learned coefficients.

In a second part of this section, ITML is compared to the standard Euclidean distance often used with Gist features. The ITML algorithm needs an a priori knowledge on some parameters as indicated in the previous section. We use the same parameters as [Davis et al. 2007], so we initialize the matrix A to the identity matrix, then we consider the training set as samples to compute threshold distances and constraints matrix. Only, due to computing cost, we consider the same parameters as [Davis et al. 2007] to choose randomly $20 \times C^2$ constraint pairs of features from the training set. Consequently, the approximation of the matrix $A$ is non-deterministic. This is why we consider the mean classification result under 10 different evaluations. We do the same thing for the k-NN method since prototypes are taken randomly.

### A.3.3 Robustness to dimension reduction

Our tests aim to classify unlabeled queries based on trained classifiers. We evaluate results of classification using the mean Average Precision (mAP) value, which is the mean of the right classification rate of all categories.

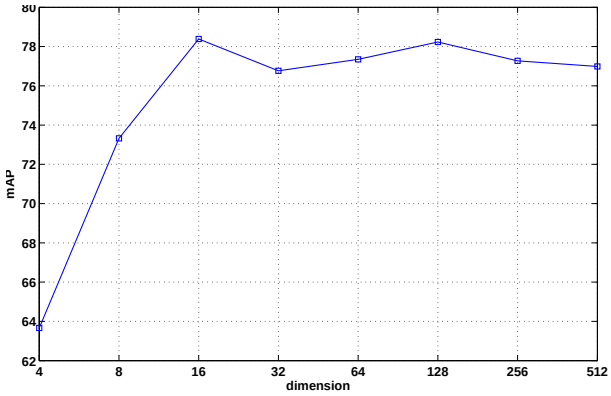


Figure A.2: Mean average precision curve for classification depending on Gist dimension.

First, to prove that dimension reduction does not affect the robustness of Gist features we evaluate classification using those descriptors in different dimensions. The Fig.A.2 presents the evolution of the mean average precision in function of the dimension of the features. We vary dimension from 4 to 512 and we notice that mAP value is practically constant from dimension 16. This shows that dominant information is located on the first 16 components of the descriptor. Consequently, we use Gist on 32 dimensions instead of 512 which makes a huge difference in computation time.

### A.3.4 Boosting k-NN results and comparison to the k-NN classification method

UNN is an approach based on nearest neighbors framework. Therefore, it uses boosted coefficients to choose a set $P$ of best classifiers from the train set, and that is sufficient to reach a best categorization results. Next, we evaluate the influence of trained prototypes number on classification rate respectively for UNN approach and for k-NN one. We test on 10, 20, 30, 40 and 50 percent of the training set as prototypes to compare the two previous approaches. The evaluation in Fig.A.3

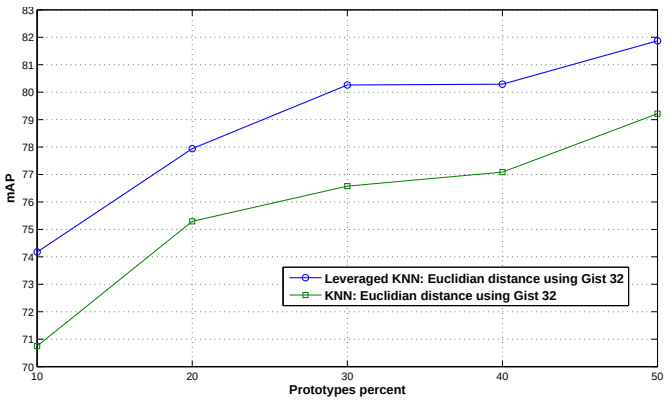


Figure A.3: Comparaison between k-NN classification and UNN one.

shows that UNN classification is more efficient than standard k-NN, and using only 50 percent of $S$ as prototypes we reach a significant precision rate. In other evaluations not reported here, we had to consider all training set as classifiers for the uniform (standard) k-NN method to achieve the same efficiency as UNN algorithm.

Notice that this mAP is comparable to the result of Torralba [1] based on SVM method, except that with the UNN approach we are scalable.

### A.3.5 Evaluation of the metric learning process

For more efficiency, we substitute euclidean distance by ITML to adapt the metric to features. Results in Fig.A.4 indicate that with this metric we can get more robustness with fewer learned classifiers, which is really important when dealing with large datasets.

As reported in Fig.A.4, for 400 prototypes (20% of the training data), we have already more than 81% of precision when using UNN with ITML metric. And for only 600 classifiers (30%) we come to the same precision rate as using the euclidean distance with 1000 classifiers (50%).

We also test the optimization of ITML on k-NN method as shown in Fig.A.4 and evaluation conduct to the same conclusion as with UNN algorithm.

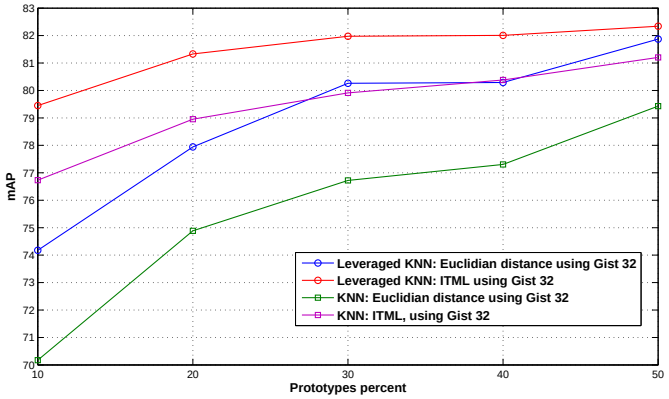


Figure A.4: UNN classification using Euclidean distance and ITML.

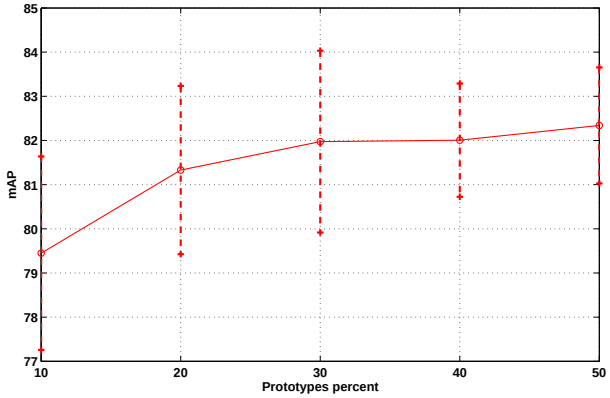


Figure A.5: MAP variance of UNN classification with ITML distance over 10 evaluations.

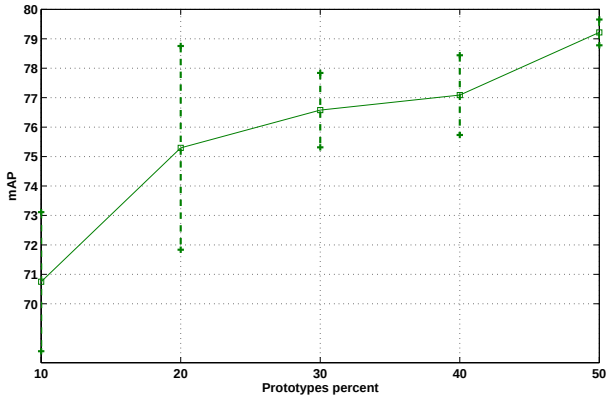


Figure A.6: MAP variance of k-NN classification over 10 evaluations

We remind that because of the randomness of the learning process of the matrix

$A$, previous results on ITML are averaged over 10 evaluations. Fig.A.5 shows the variance of these results.

The same process is applied to k-NN method as prototypes are chosen randomly, and a preview of the Fig.A.6 shows the detailed results.

Appendix B

# Convergence proof of N$^3$ and statistical properties

## B.1 Proof of Theorem 3

The proof is sketched for the calibrated Hinge loss, and so consider row D in Tables 3.1, 3.2, 3.3. For the sake of simplicity, let us name $F_D$ the calibrated Hinge loss, and suppose we are at the beginning of step $t$ and class $c$ in N$^3$, with $j$ the index returned by Wic. The current leveraged NNis denoted $H_c^t$ and the current weights are denoted $\boldsymbol{w}_t$. For any $i$ in the inverse neighborhood of $j$, let us denote

$$\tilde{H}_c^t(\boldsymbol{x}_i) \doteq H_c^t(\boldsymbol{x}_i) + y_{ic} g'_{F_D}(K_{F_D} w_{1,i}) \ . \tag{B.1}$$

Classifier $\tilde{H}_c^t$ is the leveraged NNto which we add a constant term which depends on the initialization weight of example $i$. We now focus on establishing a convergence property for $\varepsilon_{F_D}(\tilde{H}_c, \mathcal{S})$, which will then be translated to $H_c$. First, we upperbound the variation between two successive values of $\varepsilon_{F_D}(., \mathcal{S})$. After several derivations, we obtain:

$$\begin{aligned} &\varepsilon_{F_D}(\tilde{H}_c^{t+1}, \mathcal{S}) - \varepsilon_{F_D}(\tilde{H}_c^t, \mathcal{S}) \\ &= -\frac{1}{m} \sum_{i:j \to_k i} \Delta_{-g_{F_D}}(w_{t+1,i} \| w_{t,i}) - \frac{\tilde{\eta}\delta_j}{m} \ , \end{aligned} \tag{B.2}$$

with $\Delta_{-g_{F_D}}$ the Bregman divergence with generator $-g_{F_D}$ [Kakade et al. 2009], and $\tilde{\eta} \doteq \sum_{i:j \to_k i} w_{t+1,i} y_{ic} y_{jc}$. Notice that $\tilde{\eta}$ is not measured on the same weights as $\eta(c, j)$.

Using the fact that $F_D$ is $F''_D(0) = 1/4$ is strongly smooth and Theorem 6 in [Kivinen & Warmuth 2001], we obtain that $-g_{F_D} - 2x^2$ is convex. Considering its Bregman divergence computed between $w_{t+1,i}$ and $w_{t,i}$, summing for all $i$ in the inverse neighborhood of $j$ and rearranging terms, we obtain:

$$\sum_{i:j \to_k i} \Delta_{-g_{F_D}}(w_{t+1,i} \| w_{t,i}) \geq 2 \sum_{i:j \to_k i} (w_{t+1,i} - w_{t,i})^2 \ .$$

After remarking that $\sum_{i:j \to_k i} (y_{ic} y_{j_c})^2 = n_j$, Cauchy-Schwartz inequality yields

$$n_j^2 \sum_{i:j \to_k i} (w_{t+1,i} - w_{t,i})^2 \geq (\tilde{\eta} - \eta(c, j))^2 \ ,$$

that is:

$$\sum_{i:j\rightarrow_k i} \Delta_{-g_{F_D}}(w_{t+1,i}\|w_{t,i}) \quad \geq \quad \frac{2(\tilde{\eta}-\eta(c,j))^2}{n_j} \ .$$

Plugging this into (B.2) yields after few more derivations the left-hand side inequality of:

$$\begin{aligned} &\varepsilon_{F_D}(\tilde{H}_c^{t+1},\mathcal{S}) - \varepsilon_{F_D}(\tilde{H}_c^{t},\mathcal{S}) \\ &\leq \quad -\frac{2\eta(c,j)^2}{mn_j} \leq -8\frac{n_j\gamma_n^2\gamma_u^2}{m} \ . \end{aligned} \tag{B.3}$$

The right-hand side inequality of (B.3) comes from the weak learning assumption. Hence, for some $\varepsilon < F(0) = -\ln(2)$, to obtain $\varepsilon_{F_D}(\tilde{H}_c^{T+1},\mathcal{S}) \leq \varepsilon$, it is sufficient that:

$$\sum_{j\in\mathcal{J}} n_j \quad \geq \quad \frac{(-\varepsilon-\ln(2))m}{8\gamma_n^2\gamma_u^2} \ , \tag{B.4}$$

where $j$ spans the indexes of $\mathcal{J}$. To finish the proof, we shift the analysis to $H_c^{T+1}$, and obtain from (B.1) and the expressions of $F_D$ and $g_{F_D}$:

$$\begin{aligned} &\forall i : j \rightarrow_k i \ , F_D(y_{ic}H_c^{T+1}(\boldsymbol{x}_i)) \\ &= \quad F_D\left(y_{ic}\tilde{H}_c^{T+1}(\boldsymbol{x}_i) - g'_{F_D}(K_{F_D}w_{1,i})\right) \\ &= \quad F_D\left(y_{ic}\tilde{H}_c^{T+1}(\boldsymbol{x}_i) - \|\mathbf{1}+y_{ic}\boldsymbol{y}_i\|_1\right) \ , \qquad \text{(B.5)} \\ &\leq \quad F_D(y_{ic}\tilde{H}_c^{T+1}(\boldsymbol{x}_i)) + C \ . \qquad \text{(B.6)} \end{aligned}$$

There remains to combine (B.4) and (B.6) to obtain the statement of the Theorem.

## B.2 Statistical properties of N$^3$

The first property consists in a justification of the weight initialization in N$^3$. Following the terminologies of [Bartlett et al. 2006, Vernet et al. 2011], we want the total calibrated risk to be pointwise Fisher consistent: this implies that for any observation, when $p[y_c = +1|\boldsymbol{x}] = 1/C, \forall c$, the optimal constant real prediction for $\boldsymbol{x}$ is $z = 0$ [Bartlett et al. 2006, Vernet et al. 2011]. Notice that each example in $\mathcal{S}$ participates to $C$ classification problems. Consider example $i$ which meets the conditions above, and let $w^+$ (resp. $w^-$) denote its weight for the classification problem for one class to which it belongs (resp. does not belong) vs all others in N$^3$. A constant real prediction $z$ brings for this example a contribution to the total calibrated risk proportional to $w^+\|\mathbf{1}+\boldsymbol{y}_i\|_1 F(-z) + w^-\|\mathbf{1}-\boldsymbol{y}_i\|_1 F(z)$. Given the definition of $F$ (3.3), the optimal $z$ to this contribution is found to be $z = f^{-1}\left(w^+\|\mathbf{1}+\boldsymbol{y}_i\|_1/(w^+\|\mathbf{1}+\boldsymbol{y}_i\|_1 + w^-\|\mathbf{1}-\boldsymbol{y}_i\|_1)\right)$. Because $f(0) = 1/2$, we have to ensure that $w^+\|\mathbf{1}+\boldsymbol{y}_i\|_1 = w^-\|\mathbf{1}-\boldsymbol{y}_i\|_1$, which is the case in N$^3$.

The second property establishes that $\mathrm{N}^3$ brings a convenient estimation of posteriors. It confirms the intuition that the transfer function links real valued prediction to the estimation of posteriors (See Section 3.3).

**Theorem 6** *For any $c$, $f(H_c(\boldsymbol{x}))$ is an efficient estimator of $p[y_c = +1|\boldsymbol{x}]$.*

The proof, given in appendix, calls to the representation of exponential families. It is interesting in itself as it shows that the duality between labels prediction and posteriors estimation born from the transfer function (Section 3.3) implies a duality between the classification calibrated risk — which depends upon labels — and the log-likelihood of some exponential family — which is parameterized by posteriors —.

The third property shows that $\mathrm{N}^3$ is weakly universally consistent. It makes use of the definition of the empirical risk of $H$ on $\mathcal{S}$ ($1_{\cdot}$ is the indicator variable):

$$\varepsilon_{0/1}(H, \mathcal{S}) \doteq \frac{1}{C}\sum_{c=1}^{C}\frac{1}{m}\sum_{i=1}^{m} 1_{y_{ic}H_c(\boldsymbol{x}_i)<0} \ . \tag{B.7}$$

**Theorem 7** *Suppose that examples in $\mathcal{S}$ are drawn i.i.d. according to some unknown but fixed distribution $\mathcal{D}$. Let $R_{m,T} \doteq \mathbb{E}_{\mathcal{S}:|\mathcal{S}|=m}[\varepsilon_{0/1}(H, \mathcal{S})]$ denote the expectation, over the sampling of size-$m$ samples following $\mathcal{D}$, of classifier $H$ built by $\mathrm{N}^3$ after $T$ rounds of boosting for each class. Then, as $k \to +\infty$, provided $k = \mathcal{O}(T)$ and $T = \mathcal{O}(m)$, $\mathrm{N}^3$ is weakly universally consistent: regardless of $\mathcal{D}$,*

$$\lim_{m\to+\infty} R_{m,T} = R^* \ , \tag{B.8}$$

*where $R^*$ is Bayes risk.*

**Comments on Theorems 3 and 7**: the Theorems provide sets of choices for parameters that make it possible for $\mathrm{N}^3$ to perform consistent *and* sparse boosting. For example, $k = \mathcal{O}(m^\mu), T = \mathcal{O}(m^\nu)$, with $0 < \mu, \nu < 1$ and $\mu + \nu > 1$.

## B.3 Proof of Theorem 6

We focus on class $c$ and remove for the sake of readability the reference to $c$ in all notations. We let $y \in \{-1, +1\}$ denote the membership to the class and $\hat{y}_\varepsilon \doteq (1/2 - \varepsilon)y + 1/2 \in \{\varepsilon, 1 - \varepsilon\}$, for some $\varepsilon \in [0, 1)$. Letting for short $H \doteq H(\boldsymbol{x})$, we

have:

$$\begin{aligned}
&\Delta_{-g_F}(1-\varepsilon\|f(H)) \\
&\doteq\ (-g_F)(1-\varepsilon) - (-g_F)(f(H)) \\
&\quad -(1-\varepsilon-f(H))(-g_F)'(f(H)) \\
&=\ (-g_F)(1-\varepsilon) + (-g_F)^\star(H) + \varepsilon H - H\ , \qquad \text{(B.9)} \\
&\Delta_{-g_F}(\varepsilon\|f(H)) \\
&\doteq\ (-g_F)(\varepsilon) - (-g_F)(f(H)) \\
&\quad -(\varepsilon-f(H))(-g_F)'(f(H)) \\
&=\ (-g_F)(\varepsilon) + (-g_F)^\star(H) - \varepsilon H\ , \qquad \text{(B.10)}
\end{aligned}$$

where $\Delta_{-g_F}$ is the Bregman divergence with generator $-g_F$ [Kivinen & Warmuth 2001]. To derive (B.9) and (B.10), we have used (i) the fact that convex conjugates have derivatives that are inverse of each other, (ii) $f = (-g_F)'^{-1}(x)$ from (3.3) and (3.4), and (iii) the convex conjugate of some strictly convex differentiable function $h$ is $h^\star(x) = xh'^{-1}(x) - h(h'^{-1}(x))$. Since $g_F$ is permissible, $(-g_F)(\varepsilon) = (-g_F)(1-\varepsilon)$, and we remark that $F(-x) = F(x) + x$, so that (3.3) and (3.4) become:

$$\begin{aligned}
\Delta_{-g_F}(1-\varepsilon\|f(H)) &= u(\varepsilon) + \varepsilon H + F(H)\ , \\
\Delta_{-g_F}(\varepsilon\|f(H)) &= u(\varepsilon) - \varepsilon H + F(-H)\ ,
\end{aligned}$$

where $u \doteq (-g_F)(\varepsilon) = (-g_F)(1-\varepsilon)$. We end up having:

$$F_\varepsilon(yH) = \Delta_{-g_F}(\hat{y}_\varepsilon\|f(H)) - u(\varepsilon)\ , \qquad \text{(B.11)}$$

with $F_\varepsilon(x) \doteq F(x) - \varepsilon x$. Now,

$$\begin{aligned}
&\Delta_{-g_F}(\hat{y}_\varepsilon\|f(H)) \\
&=\ -\log p_{((-g_F)^\star,H)}[\hat{y}_\varepsilon = 1-\varepsilon|\boldsymbol{x}] + \log v(\boldsymbol{x})\ , \qquad \text{(B.12)}
\end{aligned}$$

where $p_{((-g_F)^\star,H)}$ is the pdf of the exponential family parameterized by $(-g_F)^\star$, with natural parameter $H$ and expectation parameter $f(H)$ [Banerjee et al. 2005]. Hence, $f(H)$ is an estimator of:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}}[\hat{y}_\varepsilon] &= (1-\varepsilon)p[y=+1|\boldsymbol{x}] + \varepsilon p[y=-1|\boldsymbol{x}] \\
&= p[y=+1|\boldsymbol{x}] + \varepsilon(p[y=-1|\boldsymbol{x}] - p[y=+1|\boldsymbol{x}])\ .
\end{aligned}$$

In fact, $f(H)$ is the *only* efficient estimator of $\mathbb{E}_{\boldsymbol{x}}[\hat{y}_\varepsilon]$ [Müller-Funk et al. 1989]. Plugging (B.11) and (B.12) together, we get:

$$-\log p_{((-g_F)^\star,H)}[\hat{y}_\varepsilon = 1-\varepsilon|\boldsymbol{x}] = F_\varepsilon(yH) + r\ ,$$

where $r$ does not depend upon $H$ or $T$. Hence, minimizing $\varepsilon_{F_\varepsilon}(H,\mathcal{S})$ amounts to a maximum likelihood fitting of $f(H)$. There remains to take $\varepsilon = 0$ for A, B, C to conclude the Theorem. For D, since $g_F$ is not defined in $\{0,1\}$, we can pick $\varepsilon = 1/T^2$. Since weights are finite, leveraging coefficients are finite. Thus, $|H| = o(T^2)$, and so $F_{1/T^2}(yH) = F(yH) + o(1)$. There remains to take the limit in $T$ to conclude.

Appendix C

# Convergence proof of SLND

## C.1 Proofsketch of Theorem 5

The proofsketch of Theorem 5 involves there steps:

- Bregman divergence estimation.
- Calibrated loss properties.
- Weak Separability Assumption.

We first make some simplifications in notations. We remove the $c$ subscript and make the analysis for class $c$, and thus focus on the analysis of $\varepsilon_{\mathrm{F}}(h_c, \mathcal{S})$, noted for short $\varepsilon_{\mathrm{F}}(h, \mathcal{S})$. To avoid confusion, we also rename example chosen at iteration $t$ in (4.16) as example $i_t$, so that (4.16) reads:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t y_{i_t} F'\left(y_{i_t} \boldsymbol{w}_t^T \boldsymbol{x}_{i_t}\right) \boldsymbol{x}_{i_t}^{\circ} \ . \tag{C.1}$$

### Bregman divergence estimation

Let us define the Legendre conjugate and the notion of Bregman divergence. $\tilde{\mathrm{F}}(x) \doteq \mathrm{F}^{\star}(-x)$, where $\star$ denotes the Legendre conjugate ($\mathrm{F}^{\star}(x) \doteq x(\mathrm{F}')^{-1}(x) - \mathrm{F}((\mathrm{F}')^{-1}(x))$), and $D_{\tilde{\mathrm{F}}}(u \| v) \doteq \tilde{\mathrm{F}}(u) - \tilde{\mathrm{F}}(v) - (u - v)\tilde{\mathrm{F}}'(v)$ denotes the Bregman divergence with generator $\tilde{\mathrm{F}}$ [Nock & Nielsen 2008].

We get the following equality

$$\begin{aligned}
&\varepsilon_{\mathrm{F}}(\boldsymbol{w}_{t+1}, \mathcal{S}) - \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}) \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathrm{F}(y_{ic} \boldsymbol{w}_{t+1}^{\top} \boldsymbol{x}_i) - \frac{1}{m}\sum_{i=1}^{m} \mathrm{F}(y_{ic} \boldsymbol{w}_t^{\top} \boldsymbol{x}_i) \\
&= -\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\mathrm{F}}}(p_{(t+1)i} \| p_{ti}) \\
&\quad -\frac{\eta_t}{m}\sum_{i=1}^{m} p_{(t+1)i} y_i y_{i_t} \pi(i_t, i) \ ,
\end{aligned} \tag{C.2}$$

where

$$\pi(i, i_t) \doteq p_{ti_t} \boldsymbol{x}_{i_t}^{\top} \mathcal{H}^{\star} \boldsymbol{x}_i = p_{ti_t} (\boldsymbol{x}_i^{\circ})^{\top} \boldsymbol{x}_{i_t}^{\circ} \ , \tag{C.3}$$

### Calibrated loss properties

Since $\mathrm{F}''(x) \leq \mathrm{F}''(0)$ for the classification calibrated losses we consider, we also have the following quadratic lower-bound which can be obtained following [Kakade et al. 2009]:

$$\sum_{i=1}^{m} D_{\tilde{\mathrm{F}}}(p_{(t+1)i} \| p_{ti}) \quad \geq \quad \frac{1}{2\mathrm{F}''(0)} \sum_{i=1}^{m} (p_{(t+1)i} - p_{ti})^2 \ . \tag{C.4}$$

Cauchy-Schwartz inequality brings:

$$\sum_{i=1}^{m} (y_i y_{i_t} \pi(i_t, i))^2 \sum_{i=1}^{m} (p_{(t+1)i} - p_{ti})^2 \tag{C.5}$$

$$\geq \quad \left( \sum_{i=1}^{m} y_i y_{i_t} \pi(i_t, i)(p_{(t+1)i} - p_{ti}) \right)^2 \ . \tag{C.6}$$

Define for short $v_t \doteq \sum_{i=1}^{m} p_{(t+1)i} y_i y_{i_t} \pi(i_t, i)$, $e_t \doteq \sum_{i=1}^{m} p_{ti} y_i y_{i_t} \pi(i_t, i)$ and $\Pi_t \doteq \sum_{i=1}^{m} \pi^2(i_t, i)$. Plugging (C.4) and (C.6) into (C.2) and simplifying, we obtain:

$$\begin{aligned} &\varepsilon_{\mathrm{F}}(\boldsymbol{w}_{t+1}, \mathcal{S}) - \varepsilon_{\mathrm{F}}(\boldsymbol{w}_t, \mathcal{S}) \\ &\leq \underbrace{-\frac{(v_t - e_t)^2}{2\mathrm{F}''(0) m \Pi_t} - \frac{\eta_t v_t}{m}}_{\doteq \frac{\Delta_t(v_t)}{m}} \ . \end{aligned} \tag{C.7}$$

• $\Delta_t(v_t)$ takes its maximum for $v_t = v^\circ = e_t - \mathrm{F}''(0)\eta_t \sum_{i=1}^{m} (y_i y_{i_t} \pi(i_t, i))^2 = e_t - \mathrm{F}''(0)\eta_t \Pi_t$, for which we have:

$$\Delta_t(v^\circ) \quad = \quad \frac{\mathrm{F}''(0)\eta_t \Pi_t}{2} \times \left( \eta_t - \frac{2e_t}{\mathrm{F}''(0)\Pi_t} \right) \ .$$

Assume we pick, for some $\varepsilon \in (0, 1)$:

$$\eta_t \quad \doteq \quad \frac{2(1-\varepsilon)e_t}{\mathrm{F}''(0)\Pi_t} \ . \tag{C.8}$$

For this choice of $\eta_t$, we have:

$$\Delta_t(v^\circ) \quad = \quad -\frac{2\varepsilon(1-\varepsilon)}{\mathrm{F}''(0)} \rho(i_t, \mathcal{H}^*) \ , \tag{C.9}$$

with

$$\rho(i_t, \mathcal{H}^*) \quad \doteq \quad \frac{\left( \sum_{i=1}^{m} p_{ti} y_i (\boldsymbol{x}_i^\circ)^\top \boldsymbol{x}_{i_t}^\circ \right)^2}{\sum_{i=1}^{m} ((\boldsymbol{x}_i^\circ)^\top \boldsymbol{x}_{i_t}^\circ)^2} \ .$$

### Weak Separability Assumption

Now, the Weak Separability Assumption implies $|\sum_{i=1}^m p_{ti} y_i (\boldsymbol{x}_i^\circ)^\top \boldsymbol{x}_{i_t}^\circ| \geq \gamma \|s_t\|_1 \geq \gamma \|s_t\|_2 = \gamma \sqrt{\sum_{i=1}^m ((\boldsymbol{x}_i^\circ)^\top \boldsymbol{x}_{i_t}^\circ)^2}$, which leads to $\rho(i_t, \mathcal{H}^*) \geq \gamma^2$.
Finally, the fact that $\Delta_t(v_t) \leq \Delta_t(v^\circ)$ and (C.9) imply:

$$\Delta_t(v_t) \quad \leq \quad -\frac{2\gamma^2 \varepsilon(1-\varepsilon)}{\mathrm{F}''(0)} \ .$$

Plugging this into (C.7) achieves the proof of the theorem.
**Remarks on** $\eta_t$ (C.8) gives, under the WSA:

$$\begin{aligned} \eta_t &= \frac{2(1-\varepsilon)\sum_{i=1}^m p_{ti} y_i y_{i_t} \pi(i_t, i)}{\mathrm{F}''(0)\Pi_t} \\ &= \frac{2(1-\varepsilon)\gamma' \|s_t\|_1}{\mathrm{F}''(0)|p_{ti_t} y_{i_t}| \|s_t\|_2^2} \ , \end{aligned}$$

for some $\gamma' \geq \gamma > 0$ as in the WSA. Because $\|s_t\|_2 \leq \|s_t\|_1 \leq \sqrt{m}\|s_t\|_2$, it comes:

$$\frac{2(1-\varepsilon)\gamma'}{\mathrm{F}''(0) p_{ti_t} \|s_t\|_1} \quad \leq \eta_t \leq \quad \frac{2(1-\varepsilon)\gamma'\sqrt{m}}{\mathrm{F}''(0) p_{ti_t} \|s_t\|_1} \ .$$

Letting $\mu_t \doteq (1/m)\sum_{i=1}^m |(\boldsymbol{x}_i^\circ)^\top \boldsymbol{x}_{i_t}^\circ|$ denote the average value of $|s_{tj}|$, we obtain:

$$\frac{2(1-\varepsilon)\gamma'}{m\mathrm{F}''(0) p_{ti_t} \mu_t} \quad \leq \eta_t \leq \quad \frac{2(1-\varepsilon)\gamma'}{\sqrt{m}\mathrm{F}''(0) p_{ti_t} \mu_t} \ .$$

Hence, omitting $p_{ti_t}$ in big-Oh notations to simplify the analysis, the value $\eta_t$ which guarantees the rate of convergence of Theorem 5 is indeed roughly between $\Omega(1/m)$ and $O(1/\sqrt{m})$.

# Bibliography


[Ali et al. 2011] K. Ali, D. Hasler and F. Fleuret. *FlowBoost - Appearance learning from sparsely annotated video.* In Proc. of the $24^{th}$ IEEE CVPR, pages 1433–1440, 2011. (Cited on pages 19 and 25.)

[Amari 1998] S.-I. Amari. *Natural Gradient works efficiently in Learning.* Neural Computation, vol. 10, pages 251–276, 1998. (Cited on pages 33 and 38.)

[Banerjee et al. 2005] A. Banerjee, S. Merugu, I. Dhillon and J. Ghosh. *Clustering with Bregman Divergences.* JMLR, vol. 6, pages 1705–1749, 2005. (Cited on page 82.)

[Bartlett et al. 2006] P. Bartlett, M. Jordan and J. D. McAuliffe. *Convexity, classification, and risk bounds.* JASA, vol. 101, pages 138–156, 2006. (Cited on pages 3, 8, 20, 21, 25, 34 and 80.)

[Bel Haj Ali et al. 2010] W. Bel Haj Ali, P. Piro, E. Debreuve and M. Barlaud. *From descriptor to boosting: Optimizing the k-NN classification rule.* In Content-Based Multimedia Indexing (CBMI), 2010 International Workshop on, pages 1–5, 2010. (Cited on pages 13 and 18.)

[Bel haj ali et al. 2011] Wafa Bel haj ali, Eric Debreuve, Pierre Kornprobst and Michel Barlaud. *Bio-Inspired Bags-of-Features for Image Classification.* In International Conference on Knowledge Discovery and Information Retrieval (KDIR 2011), 2011. (Cited on page 52.)

[Beyer et al. 1999] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan and Uri Shaft. *When Is "Nearest Neighbor" Meaningful?* In Proc. of the $7^{th}$ ICDT, pages 217–235, 1999. (Cited on page 28.)

[Bordes et al. 2009] Antoine Bordes, Léon Bottou and Patrick Gallinari. *SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent.* J. Mach. Learn. Res., vol. 10, pages 1737–1754, December 2009. (Cited on pages 33, 36, 37 and 38.)

[Bordes et al. 2010] Antoine Bordes, Léon Bottou, Patrick Gallinari, Jonathan Chang and S. Alex Smith. *Erratum: SGDQN is Less Careful than Expected.* J. Mach. Learn. Res., vol. 11, pages 2229–2240, August 2010. (Cited on page 33.)

[Bottou & Bousquet 2008] Léon Bottou and Olivier Bousquet. *The tradeoffs of large scale learning.* In NIPS*20, pages 161–168, 2008. (Cited on pages 33 and 39.)

[Bottou 2010] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent.* In Yves Lechevallier and Gilbert Saporta, editeurs, Proceedings

of the 19th International Conference on Computational Statistics (COMPSTAT'2010), pages 177–187, Paris, France, August 2010. Springer. (Cited on page 2.)

[Cook 1998] R. Dennis Cook. *Principal Hessian Directions Revisited.* Journal of the American Statistical Association, vol. 93, no. 441, pages 84–94, 1998. (Cited on page 37.)

[Cristianini & Shawe-Taylor 2000] Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines: and other kernel-based learning methods. Cambridge University Press, New York, NY, USA, 2000. (Cited on page 2.)

[Dalal & Triggs 2005] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893 vol. 1, 2005. (Cited on page 1.)

[D'Ambrosio et al. ear] R. D'Ambrosio, R. Nock, W. Bel Haj Ali, F. Nielsen and M. Barlaud. *Boosting nearest neighbors for the efficient estimation of posteriors.* In Proc. of the 23 $^{rd}$ ECML-PKDD, 2012 (to appear). (Cited on pages 28 and 35.)

[Davis et al. 2007] J. Davis, B. Kulis, P. Jain, S. Sra and I. Dhillon. *Information-theoretic Metric Learning.* In ICML'07, 2007. (Cited on pages 71, 72 and 74.)

[Dayem et al. 2008] Manal Dayem, Cécile Basquin, Valérie Navarro, Patricia Carrier, Robert Marsault, Patrick Chang, Sylvaine Huc, Elisabeth Darrouzet, Sabine Lindenthal and Thierry Pourcher. *Comparison of expressed human and mouse sodium/iodide symporters reveals differences in transport properties and subcellular localization.* Journal of Endocrinology, vol. 197, no. 1, pages 95–109, 2008. (Cited on pages 53 and 54.)

[Deng et al. 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. *ImageNet: A Large-Scale Hierarchical Image Database.* In CVPR'09, 2009. (Cited on page 39.)

[Deng et al. 2010] Jia Deng, Alexander C. Berg, Kai Li and Li Fei-Fei. *What does classifying more than 10,000 image categories tell us?* In Procs of the 11 $^{th}$ ECCV, pages 71–84, 2010. (Not cited.)

[Devroye et al. 1996] L. Devroye, L. Györfi and G. Lugosi. A Probabilistic Theory of Pattern Recognition. Springer, 1996. (Cited on pages 7, 19, 20 and 25.)

[Field 1994] D. J. Field. *What Is the Goal of Sensory Coding?* Neural Computation, vol. 6, no. 4, pages 559–601, 1994. (Cited on page 52.)

[Foggia et al. 2010] P. Foggia, G. Percannella, P. Soda and M. Vento. *Early experiences in mitotic cells recognition on HEp-2 slides.* In IEEE Symp. on Comp.-Based Medical Systems, pages 38–43, 2010. (Cited on page 61.)

[Freund & Schapire 1997] Y. Freund and R. E. Schapire. *A Decision-Theoretic generalization of on-line learning and an application to Boosting.* JCSS, vol. 55, pages 119–139, 1997. (Cited on page 46.)

[Freund & Schapire 1999] Y. Freund and R. Schapire. *A short introduction to boosting.* J. Japan. Soc. for Artif. Intel., vol. 14, no. 5, pages 771–780, 1999. (Cited on page 2.)

[Friedman et al. 2000] J. Friedman, T. Hastie and R. Tibshirani. *Additive Logistic Regression : a Statistical View of Boosting.* AOS, vol. 28, pages 337–374, 2000. (Cited on pages 25 and 26.)

[Griffin et al. 2007] G. Griffin, A. Holub and P. Perona. *Caltech-256 Object Category Dataset.* Rapport technique 7694, California Institute of Technology, 2007. (Cited on pages 27 and 39.)

[Guillaumin et al. 2009] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek and Cordelia Schmid. *TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation.* In International Conference on Computer Vision, sep 2009. (Cited on page 71.)

[Hart 1968] P. E. Hart. *The Condensed Nearest Neighbor rule.* IEEE Trans. IT, vol. 14, pages 515–516, 1968. (Cited on page 7.)

[Hsu et al. 2003] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin*et al. A practical guide to support vector classification*, 2003. (Cited on pages 15 and 17.)

[Joachims 2002] Thorsten Joachims. *Optimizing search engines using clickthrough data.* In KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. (Cited on page 33.)

[Kakade et al. 2009] S. Kakade, S. Shalev-Shwartz and A. Tewari. *Applications of strong convexity–strong smoothness duality to learning with matrices.* Rapport technique CoRR abs/0910.0610, CoRR, 2009. (Cited on pages 79 and 84.)

[Kearns & Mansour 1999] M.J. Kearns and Y. Mansour. *On the boosting ability of top-down decision tree learning algorithms.* JCSS, vol. 58, pages 109–128, 1999. (Cited on page 23.)

[Keerthi et al. 2006] S Sathiya Keerthi, Olivier Chapelle, Dennis DeCoste and P Bennett. *Building support vector machines with reduced classifier complexity.* JMLR, vol. 7, no. 7, pages 1493–1515, 2006. (Cited on page 33.)

[Kivinen & Warmuth 2001] J. Kivinen and M. Warmuth. *Relative loss bounds for multidimensional regression problems.* MLJ, vol. 45, pages 301–329, 2001. (Cited on pages 21, 79 and 82.)

[LeCun et al. 1998] Yann LeCun, Léon Bottou, Genevieve B Orr and Klaus-Robert Müller. *Efficient backprop.* In Neural networks: Tricks of the trade, pages 9–50. Springer, 1998. (Cited on page 45.)

[Li 1992] Ker-Chau Li. *On Principal Hessian Directions for Data Visualization and Dimension Reduction: Another Application of Stein's Lemma.* Journal of the American Statistical Association, vol. 87, no. 420, pages 1025–1039, 1992. (Cited on page 37.)

[Ljung & Söderström 1983] L. Ljung and T. Söderström. Theory and Practice of Recursive Identification. The MIT Press, Massachusetts and London, 1983. (Cited on page 37.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, vol. 60, no. 2, pages 91–110, 2004. (Cited on pages 1, 14 and 53.)

[Mensink et al. 2012] Thomas Mensink, Jakob Verbeek, Florent Perronnin and Gabriela Csurka. *Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost.* In Procs of the 12 $^{th}$ ECCV, 2012. (Cited on pages 19, 28 and 39.)

[Müller-Funk et al. 1989] U. Müller-Funk, F. Pukelsheim and H. Witting. *On the attainment of the Cramér-Rao bound in $L_r$-differentiable families of distributions.* AOS, pages 1742–1748, 1989. (Cited on page 82.)

[Nielsen & Sérandour 2009] Frank Nielsen and Aurélien Sérandour. *Accuracy of distance metric learning algorithms.* In KDD Workshop on Data Mining using Matrices and Tensors, 2009. (Cited on page 71.)

[Nocedal 1980] E. Nocedal. *Updating quasi-Newton matrices with limited storage.* Mathematics of Computation, vol. 35, pages 773–782, 1980. (Cited on page 38.)

[Nock & Nielsen 2008] R. Nock and F. Nielsen. *On the efficient minimization of classification-calibrated surrogates.* In NIPS*21, pages 1201–1208, 2008. (Cited on pages 7, 34 and 83.)

[Nock & Nielsen 2009] R. Nock and F. Nielsen. *Bregman divergences and surrogates for learning.* IEEE PAMI, vol. 31, no. 11, pages 2048–2059, 2009. (Cited on pages 7 and 14.)

[Nock & Sebban 2001] R. Nock and M. Sebban. *An improved bound on the Finite Sample Risk of the Nearest Neighbor Rule.* Pattern Recognition Letters, vol. 22, pages 413–419, 2001. (Cited on page 7.)

[Nock et al. 2012] Richard Nock, Paolo Piro, Frank Nielsen, Wafa Bel Haj Ali and Michel Barlaud. *Boosting k-NN for Categorization of Natural Scenes.* IJCV, vol. 100, pages 294–314, 2012. (Cited on pages 12, 14, 20, 21, 23, 25 and 30.)

[Oliva & Torralba 2001] A. Oliva and A. Torralba. *Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope.* International Journal of Computer Vision, vol. 42, pages 145–175, 2001. 10.1023/A:1011139631724. (Cited on pages 1, 53, 71 and 73.)

[Oliva & Torralba 2006] Aude Oliva and Antonio Torralba. *Building the gist of a scene: the role of global image features in recognition.* Progress in brain research, vol. 155, pages 23–36, 2006. (Cited on pages 1 and 71.)

[Perronnin et al. 2010] F. Perronnin, J. Sánchez and T. Mensink. *Improving the fisher kernel for large-scale image classification.* In Procs of the 11 $^{th}$ ECCV, pages 143–156, 2010. (Cited on pages 1, 28, 33 and 39.)

[Perronnin et al. 2012] F. Perronnin, Z. Akata, Z. Harchaoui and C. Schmid. *Towards good practice in large-scale learning for image classification.* In CVPR'12, pages 3482–3489, 2012. (Cited on pages 27, 35, 38 and 39.)

[Peyrottes et al. 2009] Isabelle Peyrottes, Valerie Navarro, Alejandro Ondo-Mendez, Didier Marcellin, Laurent Bellanger, Robert Marsault, Sabine Lindenthal, Francette Ettore, Jacques Darcourt and Thierry Pourcher. *Immunoanalysis indicates that the sodium iodide symporter is not overexpressed in intracellular compartments in thyroid and breast cancers.* European Journal of Endocrinology, vol. 160, no. 2, pages 215–25, 2009. (Cited on page 53.)

[Piro et al. 2012] Paolo Piro, Richard Nock, Frank Nielsen and Michel Barlaud. *Leveraging k-NN for generic classification boosting.* Neurocomputing, vol. 80, no. 0, pages 3 – 9, 2012. <ce:title>Special Issue on Machine Learning for Signal Processing 2010</ce:title>. (Cited on pages 7 and 18.)

[Radovanović et al. 2010] M. Radovanović, A. Nanopoulos and M. Ivanović. *Hubs in space: Popular nearest neighbors in high-dimensional data.* JMLR, vol. 11, pages 2487–2531, 2010. (Cited on page 28.)

[Schraudolph et al. 2007] Nicol N. Schraudolph, Jin Yu and Simon Günter. *A Stochastic Quasi-Newton Method for Online Convex Optimization.* In AISTATS'07, pages 436–443, San Juan, Puerto Rico, 2007. JMLR. (Cited on pages 33 and 38.)

[Shakhnarovich et al. 2006] Shakhnarovich, Darrell and Indyk. Nearest-neighbors methods in learning and vision. MIT Press, 2006. (Cited on page 7.)

[Shalev-Shwartz et al. 2007] Shai Shalev-Shwartz, Yoram Singer and Nathan Srebro. *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM.* In ICML

'07, pages 807–814, New York, NY, USA, 2007. ACM. (Cited on pages 19, 33 and 39.)

[Sivic & Zisserman 2006] Josef Sivic and Andrew Zisserman. *Video Google: Efficient Visual Search of Videos.* In Jean Ponce, Martial Hebert, Cordelia Schmid and Andrew Zisserman, editeurs, Toward Category-Level Object Recognition, volume 4170 of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin / Heidelberg, 2006. 10.1007/11957959_7. (Cited on pages 1 and 62.)

[Van Rullen & Thorpe 2001] R. Van Rullen and S. J. Thorpe. *Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex.* Neural Comput, vol. 13, no. 6, pages 1255–1283, June 2001. (Cited on page 53.)

[Vapnik 1998] V. Vapnik. Statistical learning theory. John Wiley, 1998. (Cited on page 33.)

[Vedaldi & Fulkerson 2008] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms.* http://www.vlfeat.org/, 2008. (Cited on page 62.)

[Vernet et al. 2011] E. Vernet, R.-C. Williamson and M. Reid. *Composite multiclass losses.* In NIPS*24, pages 1224–1232, 2011. (Cited on pages 8, 20, 21, 34 and 80.)

[Wang et al. 2010] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang and Yihong Gong. *Locality-constrained Linear Coding for image classification.* In CVPR'10, pages 3360–3367, 2010. (Cited on page 33.)

[Weinberger & Saul 2009] K.-Q. Weinberger and L.-K. Saul. *Distance Metric Learning for Large Margin Nearest Neighbor Classification.* JMLR, vol. 10, pages 207–244, 2009. (Cited on page 19.)

[Weston et al. 2011] Jason Weston, Samy Bengio and Nicolas Usunier. *WSABIE: Scaling Up to Large Vocabulary Image Annotation.* In IJCAI'11, pages 2764–2770, 2011. (Cited on page 35.)

[Xiao et al. 2010] J. Xiao, J. Hays, K.-A. Ehinger, A. Oliva and A. Torralba. *SUN database: Large-scale scene recognition from abbey to zoo.* In CVPR'10, pages 3485–3492, 2010. (Cited on pages 14, 28, 29 and 39.)

[Zhang 2004] Tong Zhang. *Solving large scale linear prediction problems using stochastic gradient descent algorithms.* In ICML '04, pages 116–, New York, NY, USA, 2004. ACM. (Cited on pages 33 and 39.)

[Zhou et al. 2010] Xi Zhou, Kai Yu, Tong Zhang and ThomasS. Huang. *Image Classification Using Super-Vector Coding of Local Image Descriptors.* In

Kostas Daniilidis, Petros Maragos and Nikos Paragios, editeurs, Computer Vision – ECCV 2010, volume 6315 of *Lecture Notes in Computer Science*, pages 141–154. Springer Berlin Heidelberg, 2010. (Cited on page 33.)

**Abstract:**

**Minimization of Calibrated Loss Functions for Image Classification**

Image classification becomes a big challenge since it concerns on the one hand millions or billions of images that are available on the web and on the other hand images used for critical real-time applications.

This classification involves in general learning methods and classifiers that must require both precision as well as speed performance. These learning problems concern a large number of application areas: namely, web applications (profiling, targeting, social networks, search engines), "Big Data" and of course computer vision such as the object recognition and image classification.

This thesis concerns the last category of applications and is about supervised learning algorithms based on the minimization of loss functions (error) called "calibrated" for two kind of classifiers: k-Nearest Neighbours (kNN) and linear classifiers.

Those learning methods have been tested on large databases of images and then applied to biomedical images.

In a first step, this thesis revisited a Boosting kNN algorithm for large scale classification. Then, we introduced a new method of learning these NN classifiers using a Newton descent approach for a faster convergence. In a second part, this thesis introduces a new learning algorithm based on stochastic Newton descent for linear classifiers known for their simplicity and their speed of convergence.

Finally, these three methods have been used in a medical application regarding the classification of cells in biology and pathology.

**Résumé:**

## Minimisation de fonctions de perte calibrées pour la classification des images

La classification des images est aujourd'hui un défi d'une grande ampleur puisque ça concerne d'un cote les millions voir des milliards d'images qui se trouvent partout sur le web et d'autre part des images pour des application temps réel critiques.

Cette classification fait appel en général à des méthodes d'apprentissage et à des classifieurs qui doivent répondre à la fois à la précision ainsi qu'à la rapidité. Ces problèmes d'apprentissage touchent aujourd'hui un grand nombre de domaines d'applications: à savoir, le web (profiling, ciblage, réseaux sociaux, moteurs de recherche), les "Big Data" et bien évidemment la vision par ordinateur tel que la reconnaissance d'objets et la classification des images.

La présente thèse se situe dans cette dernière catégorie et présente des algorithmes d'apprentissage supervisé basés sur la minimisation de fonctions de perte (erreur) dites "calibrées" pour deux types de classifieurs: k-Plus Proches voisins (kNN) et classifieurs linéaires.

Ces méthodes d'apprentissage ont été testées sur de grandes bases d'images et appliquées par la suite à des images biomédicales.

Ainsi, cette thèse reformule dans une première étape un algorithme de Boosting des kNN et présente ensuite une deuxième méthode d'apprentissage de ces classifieurs NN mais avec une approche de descente de Newton pour une convergence plus rapide. Dans une seconde partie, cette thèse introduit un nouvel algorithme d'apprentissage par descente stochastique de Newton pour les classifieurs linéaires connus pour leur simplicité et leur rapidité de calcul.

Enfin, ces trois méthodes ont été utilisées dans une application médicale qui concerne la classification de cellules en biologie et en pathologie.