Ecole polytechnique
Promotion X2012
WEIL Michael

RAPPORT DE STAGE DE RECHERCHE
Stage Apprentissage Statisique
Los Alamos National Lab / Ernst & Young

Département de Mathématiques Appliquées
MAP 594 : Modélisation probabiliste et statistique
Directeur de stage : Stéphane Gaiffas
mars 2015-juillet 2015 Los Alamos New Mexico USA

Déclaration d'intégrité relative au plagiat

Je soussigné Michael Weil, certifie sur l'honneur :

- **1.** Que les résultats décrits dans ce rapport sont l?aboutissement de mon travail.
- **2.** Que je suis l'auteur de ce rapport.
- **3.** Que je n'ai pas utilisé des sources ou résultats tiers sans clairement les citer et les référencer selon les règles bibliographiques préconisées.

Je déclare que ce travail ne peut être suspecté de plagiat.

Date : le 1er Septembre 2015 Signature:

Abstract

In this paper, we are considering a new kind of statistical learning classification problem in the presence of label noise. A small training set is trained by adding a sample of examples with noisy labels at a constant noise rate ϵ. To deal with this problem, we are using an algorithm derived from AdaBoost, especially from its relationship with Function Gradient Descent algorithms. Specifically, the algorithm called Stochastic Core AdaBoost is originated from a Stochastic Gradient Descent method applied on a particular loss function. Through simulations we prove that this new method seems to be more robust towards noise than the original AdaBoost applied on this specific classification problem. This paper also introduces a new method to estimate the noise rate ϵ. This method is based on dividing the sample into subsamples, thanks to nearest neighbors estimation, in order to estimate local noise rates.

Résumé

Dans cet article, nous considérons, dans le cadre de l'apprentissage statistique, un nouveau problème de classification en présence de labels bruités. Un ensemble d'apprentissage de petite taille est entraîné en ajoutant un échantillon d'exemples contenant des labels bruités avec un taux constant ϵ. Ce problème est abordé à l'aide d'un algorithme issu d'AdaBoost, notamment grâce à son lien avec les algorithmes de descente de gradient. Plus précisément, l'algorithme du nom de Stochastic Core AdaBoost provient d'une méthode de descente de gradient stochastique appliquée sur une certaine fonction de perte. Nous prouvons à partir de simulations que le nouvel algorithme est plus robuste face au bruit qu'AdaBoost. Cet article présente aussi une nouvelle méthode d'estimation d'ϵ. Cette méthode est basée sur la subdivision de l'échantillon, grâce entre autres à l'estimation des plus proches voisins, afin de calculer des taux de bruit locaux.

Contents

1 Introduction

One of the issues that classification problems may encounter is the presence of label noise. Random Label noise is a real problem likely to happen in every concrete learning process. Being noisy means that each variable may have its label changed with a certain probability $\epsilon < \frac{1}{2}$. For instance, we consider labels that can only take two values -1 and 1. Noise labelling means here that a label -1 is likely to be changed into 1, and vice versa.
In this paper we consider another classification problem. Instead of having a noisy set, the dataset here is divided into two subsets. One subset is called the clean set, where the examples are not subject to noise. The size of the clean sample is smaller than the size of the other subset called the noisy set. In this noisy set, the examples can be mislabeled at a constant noise rate ϵ. The problem here is, in a way, similar to semi-supervised learning. But instead of improving the classification of a "small" learning set thanks to unlabeled data, the labeled data are trained by adding mislabeled ones.
One of the main aspects of classification is transforming the problem into a convex minimization problem using loss functions. Under certain assumptions, the minimizer of the risk of the loss function minimizes the classification risk. For our problem, the loss functions will take different expressions on the clean and noisy sets.
Boosting is one of the main learning techniques originated from convex minimization. It minimizes the risk using a Gradient Descent method. It is an algorithm aimed to improve the accuracy of any classifier called weak learner, provided that it is not a poor predictor in the sense that their prediction errors are below 50%. This iterative method gives a more accurate classifier called strong learner. Among Boosting methods we can cite the AdaBoost [6] where the loss function is the exponential function.
However, Boosting is generally not robust against noise labelling. The poor performance of AdaBoost on data with noisy labels is a prime example showing the failure of boosting to deal with these kinds of data. The main reason for Boosting algorithms such as AdaBoost to fail in the presence of label noise can be explained by the bad effect of the iterative reweigthing of the noisy examples.
Many papers have focused on improving robustness of Boosting towards Random label noise. Algorithms such as GentleBoost [7], MAdaBoost [11], LogitBoot [7] or BrownBoost [5] prove to be noise tolerant. This paper introduces a new Boosting algorithm derived from AdaBoost that takes into account the presence of clean labels. The new Boosting method is obtained from a Stochastic Gradient Descent algorithm on a specific loss function adapted to examples with clean and noisy labels. This paper is organized as follows. Section 2 re-introduces the concept of Boosting, especially the fact of being a Function Gradient Descent algorithm. Section 3 presents the first Boosting algorithm called Core AdaBoost. Section 4 gives the final algorithm by introducing stochastic approximation. Finally, Section 5 introduces a method to estimate the label noise rate ϵ.

2 Boosting

2.1 Definition

In the context of classification, Boosting is an ensemble method aimed to improve a single classifier. It constructs an aggregation by voting of the learners from reweighted data, each learner's vote is related to its accuracy.
The first Boosting was designed by Freund & Schapire with AdaBoost [6] . The algorithm is defined below.

Data: (X_i, Y_i) for $i = 1...N$ $Y_i \in \{-1, 1\}$, T
Result: Classifier $\hat{f}_{AdaBoost}(.)$
initialization : $\omega_i = \frac{1}{N}$ for $i = 1...N$, $\hat{h}_{AdaBoost}(.) = 0$;
for $t = 1...T$ **do**
  Fit (X_i, Y_i) with weights ω_i using weak learner $\hat{h}^{[t]}(.)$;
  $err = \sum_{i=1}^{N} \omega_i \mathbb{1}(Y_i \neq \hat{h}^{[t]}(X_i))$;
  if $err \geq \frac{1}{2}$ **then**
    STOP;
  end
  $\alpha = \frac{1}{2} log(\frac{1-err}{err})$;
  $\hat{h}_{AdaBoost}(X_i) = \hat{h}_{AdaBoost}(X_i) + \alpha \hat{h}^{[t]}(X_i)$;
  $\omega_i = \omega_i exp(-\alpha Y_i \hat{h}^{[t]}(X_i))$;
  $\omega_i = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i}$;
end
$\hat{f}_{AdaBoost}(.) = sign(\hat{h}_{AdaBoost}(.))$;
return $\hat{f}_{AdaBoost}(.)$;

Algorithm 1: AdaBoost

The weights are initially all equal : $\omega_i = \frac{1}{N}$. At each iteration, we train the dataset using a single classifier. For instance, in this paper, the decision-tree classifiers are boosted. However, the classifier h is defined in order to minimize the weighted error $err = \sum_{i=1}^{N} \omega_i \mathbb{1}(Y_i \neq h(X_i))$, which is the sum of the weights that it misclassifies. So the value of each weights shows the influence of its instance. A weight ω_i that has high value means that the example (X_i, Y_i) is important in the sense that its misclassification by the classifier would truly increase the misclassification error err. Also the weights are always normalized so that every weight's value is relative to the other ones. After learning, the classifier's vote is defined as follow : $\alpha = log(\frac{1-err}{err})$. As told before, it is related to the learner's accuracy. The lower the error err is, the higher the vote α is. Also, the weights are updated according to the classifier's performance. If an instance is misclassified, then its weight will be increased for the next round by being multiplied by $\exp(\alpha)$. On the contrary, if it is well predicted, its weight will be divided by $\exp(\alpha)$. The algorithm gives importance to the misclassified examples, hopping to be well predicted for the next round. Notice that the algorithm stopped as soon as the classifier predicts poorly, meaning that the weighted error err is greater than $\frac{1}{2}$. To respect this condition, α must be positive. The number of rounds T must be chosen wisely to avoid over-fitting.

2.2 Boosting and Loss Function : convexification

Boosting can be represented as a steepest descent algorithm [3] that minimizes the empirical expectation of a function called the loss function.This minimization problem prove to be easier because this expected value is convex whereas the usual classification, where the empirical is minimized, is not a convex minimization problem. The goal of the classification problem is to minimize the empirical misclassification error $R_n(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(-Y_i f(X_i) \geq 0)$, but this cannot be practically done because R_n is not convex. To get rid of the issue of non-convexity, the problem has to be 'convexified'. We replace the function $x \mapsto \mathbb{1}(x \geq 0)$ by a convex surrogate function which is the loss function. The empirical risk of the loss function is convex as a sum of convex functions. In the case of AdaBoost, the loss function is $x \mapsto exp(x)$. Besides being convex, the loss function L must satisfy some other conditions. It has to be greater than the classification function : $L(x) \geq \mathbb{1}(x \geq 0)$. Besides it has to satisfy $L(0) = 1$. The last condition $\lim_{x \to -\infty} L(x) = 0$ emphasizes the fact that good predictions are not penalized. The other loss functions (Figure 1) mainly used are the Logit loss $L(x) = \log_2(1+\exp(x))$ related to the LogitBoot [7], the Hinge loss $\max_x(1+x, 0)$ related to SVM. Within the conditions , Zhang inequality [12] tells that minimizing the empirical misclassification error results in minimizing the empirical expected value of the loss function $R_L^n(f) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i f((X_i))$.

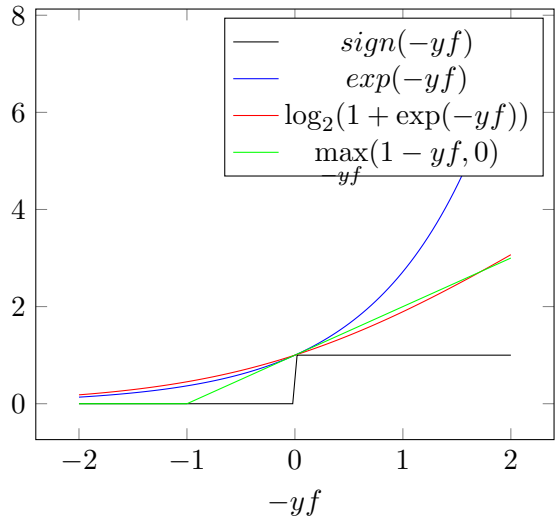


Figure 1: Loss functions

3 Boosting with noisy & clean labels

3.1 Clean & noisy labels

In real-world classification problems, the sample labels may have been corrupted because of noise. Boosting is very sensitive to these labelling errors. In the case of AdaBoost, the algorithm spends indeed efforts to classify mislabeled samples as their weights are increased exponentially. Some papers like BOOTKRAJANG and KABAN (2013)[2] try to conceal Boosting with label noise, but our goal is to deal with a distribution of both clean and noisy labels.

We consider the distribution (X_i, Y_i) for $i = 1...N$, where $Y_i \in \{-1, 1\}$. $(Y_i)_{i=1..N}$ are the genuine sample labels. The noise is here represented by a probability $\epsilon < \frac{1}{2}$ for a label to flip. The set $I = [\![1; k]\!]$ defines the "clean" set whereas the other sample points in $\complement I$ are subject to random noise.

Thus the new distribution is defined like this : $(X_i, \tilde{Y}_i)$ for $i = 1...N$, where :

$$\tilde{Y}_i = \begin{cases} Y_i \text{ if } i \in I \\ -Y_i \text{ with probability } \epsilon \text{ if } i \in \complement I \\ Y_i \text{ with probability } 1-\epsilon \text{ if } i \in \complement I \end{cases}$$

Notice that the noise is symmetric. The probability for a sample to to have its label 1 changed into -1 is the same for a sample with label -1 to be changed into 1.

3.2 Defining the new Loss Function

The goal of the paper is to adapt the Boosting method on a training set with clean and noisy labels. The clean set has a too small sample size to make a performant training set itself. So we assume that there are much more noisy instances than clean ones.
For the clean set I let's consider a loss function L used for Boosting. For instance we can use the loss function for AdaBoost $L(X_i, Y_i, f) = exp(-Y_i f(X_i))$. The idea is to have a loss function for the noisy set $\complement I$ so that the expected values of the two sets are the same. Let's call L_ϵ the loss function for the noisy set.
L_ϵ satisfies

$$\mathbb{E}(L_\epsilon(\tilde{Y}, f)) = \mathbb{E}_Y(L_\epsilon(\tilde{Y}, f) \mid Y) \tag{1}$$
$$= (1-\epsilon)\mathbb{E}_Y(L_\epsilon(Y, f) \mid Y) + \epsilon \mathbb{E}_Y(L_\epsilon(-Y, f) \mid Y) \tag{2}$$

Hence for $Y \in \{-1, 1\}$

$$\mathbb{E}(L(Y, f)) = \mathbb{E}(L_\epsilon(\tilde{Y}, f)) \Leftrightarrow \mathbb{E}(L(Y, f)) = (1-\epsilon)\mathbb{E}(L_\epsilon(Y, f)) + \epsilon \mathbb{E}(L_\epsilon(-Y, f)) \tag{3}$$

As $Y = 1$ or $Y = -1$, this gives a system of two equations :

$$\begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix} \begin{pmatrix} \mathbb{E}(L_\epsilon(1, f)) \\ \mathbb{E}(L_\epsilon(-1, f)) \end{pmatrix} = \begin{pmatrix} \mathbb{E}(L(1, f)) \\ \mathbb{E}(L(-1, f)) \end{pmatrix} \tag{4}$$

By solving this equation we get:

$$\mathbb{E}(L_\epsilon(Y, f)) = \frac{1-\epsilon}{1-2\epsilon}\mathbb{E}(L(Y, f)) - \frac{\epsilon}{1-2\epsilon}\mathbb{E}(L(-Y, f)) \tag{5}$$

One of the most obvious loss functions L_ϵ that satisfy this equation is simply :

$$L_\epsilon(\tilde{Y}, f)) = \frac{1-\epsilon}{1-2\epsilon}L(\tilde{Y}, f) - \frac{\epsilon}{1-2\epsilon}L(-\tilde{Y}, f) \tag{6}$$

For instance, for the exponential loss $L(X_i, Y_i, f) = exp(-Yf(X))$ we have a loss function

$$L_\epsilon(\tilde{Y}, f)) = \frac{1-\epsilon}{1-2\epsilon}\exp(-\tilde{Y}f(X)) - \frac{\epsilon}{1-2\epsilon}\exp(\tilde{Y}f(X))$$

By looking at the new loss function (Figure 2), we notice that it is not necessary convex and it does not converge to 0 in $-\infty$, and yet we still have $L_\epsilon(0) = L(0) = 1$. Consequently, the assumptions for the Zhang inequality are not all valid for the loss function on the noisy set. Moreover, by considering the loss function ϕ on the whole distribution, we notice that ϕ is not continuous for it has two expressions : $\phi = L_\epsilon$ or $\phi = L$. Yet, the interesting fact is that ϕ and L have the same risk :

$$R_\phi(f) = \mathbb{E}(\phi(-\tilde{Y}f(X))) = \mathbb{E}(L(-\tilde{Y}f(X))) \tag{7}$$
$$\text{i.e } R_\phi(f) = R_L(f) \tag{8}$$

Thanks to (8), to minimize the risk $R_\phi(.)$ is similar to minimize the risk $R_L(.)$, so that the Zhang inequality still holds.To approximate the risk $R_\phi(.)$ we minimize the empirical risk

$$R_\phi^n(f) = \frac{1}{n}\sum_{i=1}^{n}\phi(Y_i, f((X_i)) \tag{9}$$
$$= \frac{1}{n}\sum_{I} L(\tilde{Y}_i, f((X_i)) + \frac{1}{n}\sum_{\complement I} L_\epsilon(\tilde{Y}_i, f((X_i)) \tag{10}$$

As we perform a Boosting algorithm, R_ϕ^n is minimized over the following dictionary $\mathcal{F} = \left\{ f = \sum_{j=1}^T \theta_j h_j : \sum_{j=1}^T |\theta_j| \leq 1 \right\}$, where h_j are the base learners. $\mathcal{F}$ is by definition the L_1 ball of $Span\left\{h_1, ..., h_T\right\}$, so it is compact. As $f \in \mathcal{F} \mapsto R_\phi^n(f)$ is continuous, the minimizer $\hat{f}_\phi^n = \arg\min_{f \in \mathcal{F}} R_\phi^n(f)$ exists.
But this is not enough because this minimizer has to be consistent. To prove its consistency, we are looking at the gobal error $R_\phi(\hat{f}_\phi^n) - R_\phi^*$, where $R_\phi^* = \inf_f R_\phi(f)$. By introducing $\inf_{f \in \mathcal{F}} R_\phi(f)$, the global error can be interpreted as the sum of the stochastic error and the approximation error :

$$R_\phi(\hat{f}_\phi^n) - R_\phi^* = \underbrace{R_\phi(\hat{f}_\phi^n) - \inf_{f \in \mathcal{F}} R_\phi(f)}_{\text{stochastic error}} + \underbrace{\inf_{f \in \mathcal{F}} R_\phi(f) - R_\phi^*}_{\text{approximation error}} \tag{11}$$

To prove the consistency of $\hat{f}_\phi^n$, the stochastic error has to decrease as n goes to infinity. Under some assumptions on ϕ we can explicit an upper bound of the stochastic error.

Proposition 1. *If* $\phi(0) = 1$ *and* $\forall$ *x,x'* $\in [-1;1]$ $|\phi(x) - \phi(x')| \leq L \mid x - x'|$ *with* $L > 0$, *then*

$$\mathbb{E}(R_\phi(\hat{f}_\phi^n)) - \inf_{f \in \mathcal{F}} R_\phi(f) \leq 8L\sqrt{\frac{2\log(2T)}{n}}$$

The proof requires the Ledoux-Talagrand contraction inequality [8] and the use of the Rademacher complexity[1] of $\mathcal{F}$.
Now let us verify that ϕ respects the conditions of Proposition 1.
As seen before, ϕ respects the condition : $\phi(0) = 1$. For the second condition, let us take x and x' in $[-1;1]$ with $x \leq x'$. Regarding the expression of ϕ, $|\phi(x) - \phi(x')|$ can take two forms :

$$(i) = |\exp(x) - \exp(x')|$$

$$(ii) = |\frac{1-\epsilon}{1-2\epsilon}\exp(x) - \frac{\epsilon}{1-2\epsilon}\exp(-x) - \frac{1-\epsilon}{1-2\epsilon}\exp(x') + \frac{\epsilon}{1-2\epsilon}\exp(-x')|$$

The two forms (i) and (ii) both verify the condition because the functions are Lipschitz continuous.
Consequently, this oracle inequality in Proposition 1 proves that $\hat{f}_\phi^n$ is a consistent estimator of $f^* = \arg\min_f R_\phi(f)$.

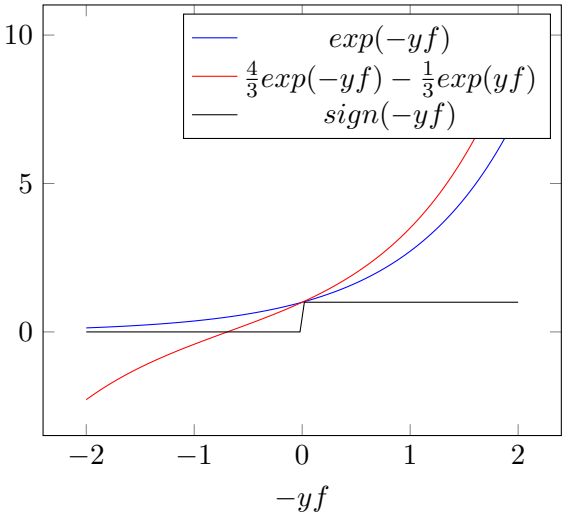


Figure 2: New loss function with $\epsilon = \frac{1}{5}$

3.3 Core Boosting algorithm

The original Boosting algorithm results from the Function Gradient Descent (FGD) algorithm on the empirical risk of the loss function. In the problem with data with clean and noisy labels, the loss function is not the same in the clean set and the noisy one, and yet we can apply the FGD to obtain an algorithm similar to the Boosting algorithm. Here we are showcasing the new Boosting algorithm applied on the loss function $x \mapsto L(x) = exp(x)$ (Algorithm 2). The proof of this new AdaBoost called Core AdaBoost is given in the Appendix. The weights on the clean set are the same weights from AdaBoost, but for the instances in $\complement I$, $\omega_i = (\omega_{1-\epsilon})_i + (\omega_\epsilon)_i$ where

$$\begin{cases} (\omega_{1-\epsilon})_i = \dfrac{1-\epsilon}{1-2\epsilon} exp(-\tilde{Y}_i \hat{h}_{AdaBoost}(X_i)) \\ (\omega_\epsilon)_i = \dfrac{\epsilon}{1-2\epsilon} exp(\tilde{Y}_i \hat{h}_{AdaBoost}(X_i)) \end{cases}$$

If we draw the two kind of weights as a function of $-\tilde{Y}_i \hat{h}_{AdaBoost}(X_i)$ (Figure 3), we notice that these two weights both penalize the same way a wrong prediction. Moreover, the weights $\omega_{\complement I}$ may be increased after good classifications. Then these news weights seem not to behave better than the AdaBoost weights towards noise labelling. Like the original AdaBoost, the Core AdaBoost also applies an aggregation by voting. The vote α is almost the same as the one from the original AdaBoost. The weighted misclassification error err still has to be smaller than $\frac{1}{2}$ in order to get $\alpha > 0$.

Data: $(X_i, \tilde{Y}_i)$ for $i = 1...N$. $Y_i \in \{-1, 1\}$. Clean set $I = [\![1; k]\!]$, Noisy set $\complement I = [\![k+1; N]\!]$. T
Result: Classifier $\hat{f}_{AdaBoost}(.)$
initialization : $\omega_i = \frac{1}{N}$ for $i = 1...N$;
$(\omega_{1-\epsilon})_i = \frac{1-\epsilon}{1-2\epsilon}$ for $i \in \complement I$;
$(\omega_\epsilon)_i = \frac{\epsilon}{1-2\epsilon}$ for $i \in \complement I$;
$\hat{h}_{AdaBoost}(.) = 0$;
for $t = 1...T$ **do**
  Fit $(X_i, \tilde{Y}_i)$ with weights ω_i using weak learner $\hat{h}^{[t]}(.)$;
  $err = \sum_{i=1}^{N} \omega_i \mathbb{1}(\tilde{Y}_i \neq \hat{h}^{[t]}(X_i))$;
  $a_\epsilon = \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i)) = \sum_{i=k+1}^{N} (\omega_\epsilon)_i$;
  if $err \geq \frac{1}{2}$ **then**
    STOP;
  end
  $\alpha = \frac{1}{2} log(\frac{1-err-a_\epsilon}{err-a_\epsilon})$;
  $\hat{h}_{AdaBoost}(X_i) = \hat{h}_{AdaBoost}(X_i) + \alpha \hat{h}^{[t]}(X_i)$;
  $\omega_i = \omega_i exp(-\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in I$;
  $(\omega_{1-\epsilon})_i = (\omega_{1-\epsilon})_i exp(-\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in \complement I$;
  $(\omega_\epsilon)_i = (\omega_\epsilon)_i exp(\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in \complement I$;
  $\omega_i = (\omega_{1-\epsilon})_i + (\omega_\epsilon)_i$ for $i \in \complement I$;
  $\omega_i = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i}$ for $i \in [\![1; N]\!]$;
end
$\hat{f}_{AdaBoost}(.) = sign(\hat{h}_{AdaBoost}(.))$;
return $\hat{f}_{AdaBoost}(.)$;

Algorithm 2: Core AdaBoost

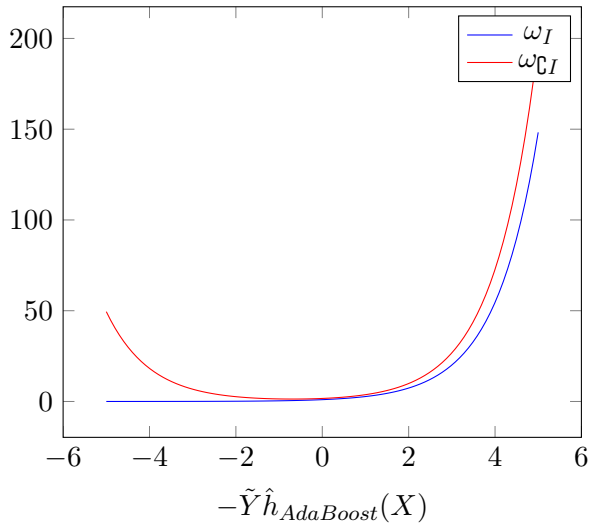


Figure 3: Weights for the Core AdaBoost with $\epsilon = \frac{1}{5}$

4 Stochastic Core Boosting

4.1 Stochastic gradient descent

The algorithm defined in 3.3 is however not reliable. First of all, the new weights $\omega_{\complement I}$ are still increased exponentially in case of wrong prediction. $a_\epsilon = \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i))$ can also be greater than $err = \sum_{i=1}^{N} \omega_i \mathbb{1}(Y_i \neq \hat{h}^{[t]}(X_i))$ or $1 - err$. So the optimized step size $\alpha = log(\frac{1-err-a_\epsilon}{err-a_\epsilon})$ may not always be defined.
The solution to avoid these issues is not to select a sequence of optimal step sizes α_n anymore. The new sequence can be chosen so that $\omega_{\complement I}$ do not "explode" in case of both good and bad prediction (see Figure 3). For instance, we can consider a sequence of steps that decrease to zero such as $\alpha_n = \frac{1}{n^2}$.
With the new problem on deciding which learning rate sequence (α_n) comes the idea of using the field of Stochastic Approximation, especially the theory of Stochastic Gradient Descent. In the early development of stochastic approximation, Robbins and Monro [10] give assumptions on the steps to have a convergent method.

Proposition 2 (Robbins-Monro (1951))**.** *Suppose that the following assumptions are true :*

Assumption 1. $(\alpha_n) \geq 0$

Assumption 2. $\sum_{n=1}^{\infty} \alpha_n = \infty$

Assumption 3. $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$

Then the stochastic approximation method converges.

The step sequence (α_n) must follow these assumptions in order to have a convergent Stochastic FGD method. In this paper, we take $\alpha_n = \frac{1}{n}$.

4.2 Bootsrapping

Stochastic Gradient Approximation is a solution to prevent Core Boosting from crashes because of an undefined value of α. But choosing an sequence of steps (α_n) that respect the conditions given by Monro and Robbins is not enough because the Boosting algorithm lacks of a random variable. To make it stochastic, we apply bootstrapping. It means that

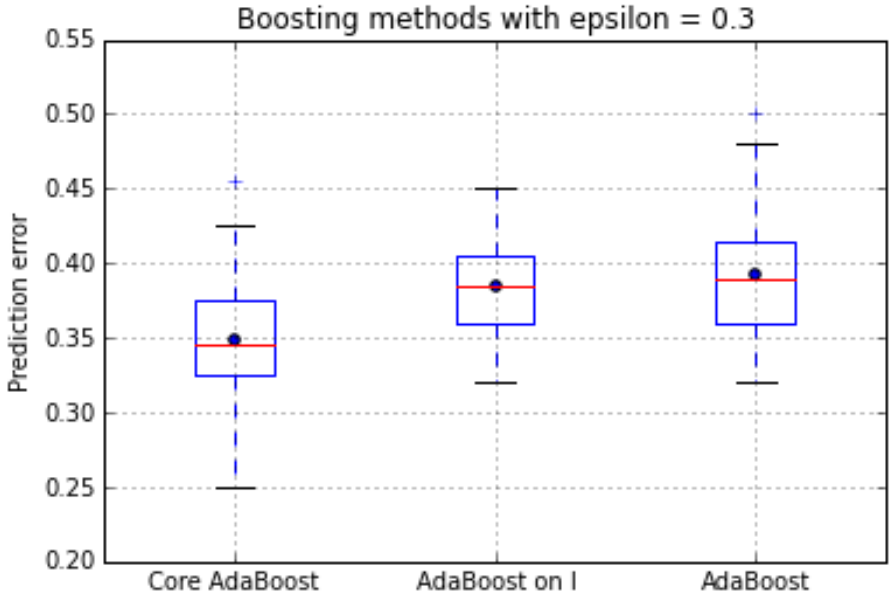
at each iteration of the Boosting, we randomly sample the training set with replacement. This bootstrapping gives us a new algorithm called the Stochastic Core Boosting. The Stochastic Core AdaBoost is defined below.

Data: $(X_i, \tilde{Y}_i)$ for $i = 1...N$. $Y_i \in \{-1, 1\}$. Clean set $I = [\![1; k]\!]$, Noisy set $\complement I = [\![k+1; N]\!]$. T
Result: Classifier $\hat{f}_{AdaBoost}(.)$
initialization : $\omega_i = \frac{1}{N}$ for $i = 1...N$;
$(\omega_{1-\epsilon})_i = \frac{1-\epsilon}{1-2\epsilon}$ for $i \in \complement I$;
$(\omega_{\epsilon})_i = \frac{\epsilon}{1-2\epsilon}$ for $i \in \complement I$;
$\hat{h}_{AdaBoost}(.) = 0$;
for $t = 1...T$ **do**
  $\mathcal{B}$ random sample of size N of $(X_i, \tilde{Y}_i)$ with replacement ;
  Fit $(X'_i, \tilde{Y'}_i) \in \mathcal{B}$ with weights ω_i using weak learner $\hat{h}^{[t]}(.)$;
  $\alpha = \frac{1}{t}$;
  $\hat{h}_{AdaBoost}(X_i) = \hat{h}_{AdaBoost}(X_i) + \alpha \hat{h}^{[t]}(X_i)$;
  $\omega_i = \omega_i exp(-\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in I$;
  $(\omega_{1-\epsilon})_i = (\omega_{1-\epsilon})_i exp(-\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in \complement I$;
  $(\omega_{\epsilon})_i = (\omega_{\epsilon})_i exp(\alpha \tilde{Y}_i \hat{h}^{[t]}(X_i))$ for $i \in \complement I$;
  $\omega_i = (\omega_{1-\epsilon})_i + (\omega_{\epsilon})_i$ for $i \in \complement I$;
  $\omega_i = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i}$ for $i \in [\![1; N]\!]$;
end
$\hat{f}_{AdaBoost}(.) = sign(\hat{h}_{AdaBoost}(.))$;
return $\hat{f}_{AdaBoost}(.)$;

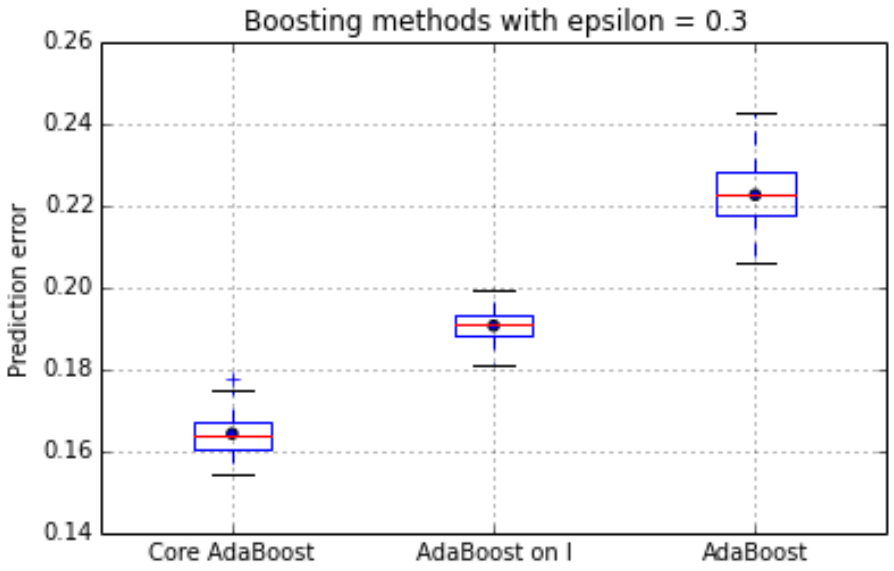
Algorithm 3: Stochastic Core AdaBoost

4.3 Experimental results

To highlight the performance of the Stochastic Core Boosting we conduct experiments on two different sets of data.
The first set randomly generated 100000 points in $[-1; 1]^2$. The labels are also attributed at random. We choose 1000 points among the set to define the clean set I. The other points are being noised at a noise rate $\epsilon = 0.3$. The test set is composed by 200 points of I. We apply the Core AdaBoost with $T = 100$ and compute the prediction error. The weak learners are decision tree classifiers. We also compute the prediction error for the AdaBoost and the AdaBoost on the clean set I only. This operation is done 100 times, each time with a different subset I. The results are shown in the Figure 4.a. We notice that the Core AdaBoost slightly performs better than the two other methods. This experiment is expected not to give outstanding results in terms of prediction error because, as the dataset is generated randomly, it is not suited for classification.
The second data set is taken from the UCI machine learning datasets repository [4]. This set is called the Adult dataset. It has 14 covariates and contains a training set of 48843 examples and a test set of 16281 instances. Again, we compare the prediction errors of the Core AdaBoost, the AdaBoost and the AdaBoost on I for different I, with $|I| = 4440$ (Figure 4.b). Compared with the previous dataset, the classification on the Adult dataset gives rather good prediction rates.The results show that the prediction errors for the Core AdaBoost are much lower than the ones from the two other methods. Also, the poor performance of the AdaBoost algorithm against noise labelling is truly emphasized in this example.



(a) Prediction error on the generated sample



(b) Prediction error on the Adult Dataset

Figure 4: Experimental results of Core AdaBoost

5 Estimation of the noise rate ϵ

5.1 Bernoulli distribution

With the Stochastic Core Boosting, it is now possible to learn on a small training set by adding a bigger but noisy dataset. However, in real-world problems, the flipping label noise rate ϵ is unknown. Few methods have been implemented to estimate the noise rate, among them, we can cite Liu & Tao (2014) [9]. Our method unlike the other ones takes into account both clean and noisy labels.But first of all, to estimate ϵ requires to make assumptions on the distribution $.(X_i, Y_i)$. Indeed, let's assume that all labels (Y_i) have a

Bernoulli distribution with the same probability p :

$$\begin{cases} P(Y_i = 1) = p \\ P(Y_i = -1) = 1 - p \end{cases}$$

The expected value of the noisy labels $(\tilde{Y}_i)$ depends on the rate ϵ :

$$\mathbb{E}(\tilde{Y}_i) = (1 - \epsilon)(p - (1 - p)) + \epsilon(-p + 1 - p) \tag{12}$$

$$\mathbb{E}(\tilde{Y}_i) = (1 - 2\epsilon)(2p - 1) \tag{13}$$

As we know that $\mathbb{E}(Y_i) = 2p-1$, we have a relationship between these two expectations

$$\mathbb{E}(\tilde{Y}_i) = (1 - 2\epsilon)\mathbb{E}(Y_i) \tag{14}$$

So that

$$\epsilon = \frac{1}{2}\frac{\mathbb{E}(Y_i) - \mathbb{E}(\tilde{Y}_i)}{\mathbb{E}(Y_i)} \tag{15}$$

5.2 Simulation & results

As seen previously, the label noise rate ϵ can be described as a function of the sample distribution. The problem, however, is that we have only one set of sample, thus the Bernoulli distribution can not be estimated. We provide an alternate solution by dividing the distribution sample. We separate the set into localized M subsamples $(\mathcal{A}_k)_{k=1..M}$, the $\mathcal{A}_k$ do not have to be a partition of the sample .Each subsamples contains examples that are in I and $\complement I$. Then for every subsample $\mathcal{A}_k$, we define the empirical means of the clean and noisy labels :

$$\mathbb{E}_I^k = \frac{1}{|\mathcal{A}_k \cap I|} \sum_{\mathcal{A}_k \cap I} Y_i$$

$$\mathbb{E}_{\complement I}^k = \frac{1}{|\mathcal{A}_k \cap \complement I|} \sum_{\mathcal{A}_k \cap \complement I} \tilde{Y}_i$$

Ideally, ϵ satisfies the relation (14) in every subset $\mathcal{A}_k$. It is mostly not the case in practice. To have an estimation of ϵ we use a least square approximation :

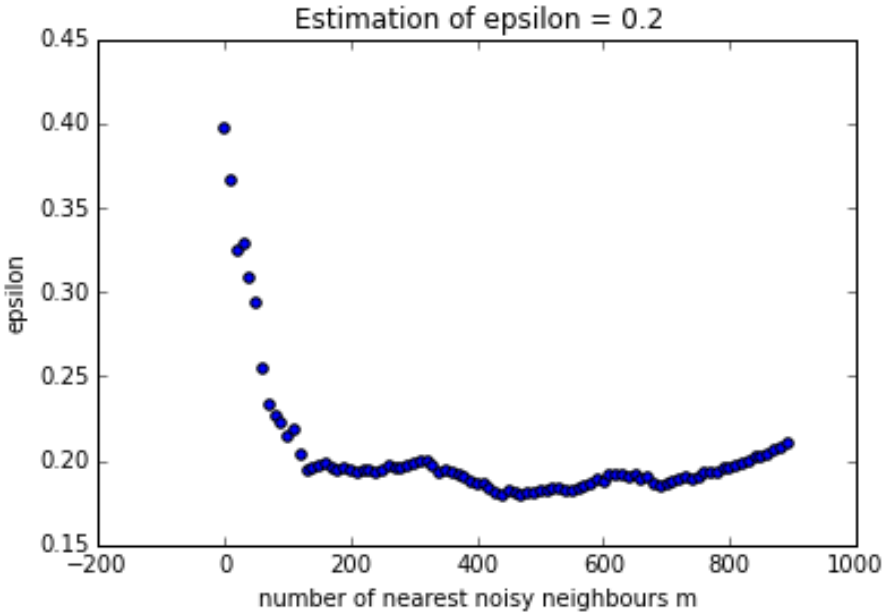
$$\epsilon = \arg\min_x \sum_{k=1}^{M} (\mathbb{E}_{\complement I}^k - (1 - 2x)\mathbb{E}_I^k)^2 \tag{16}$$

The sum in the relation above is just a second degree polynomial. Hence we can have explicit form of ϵ :

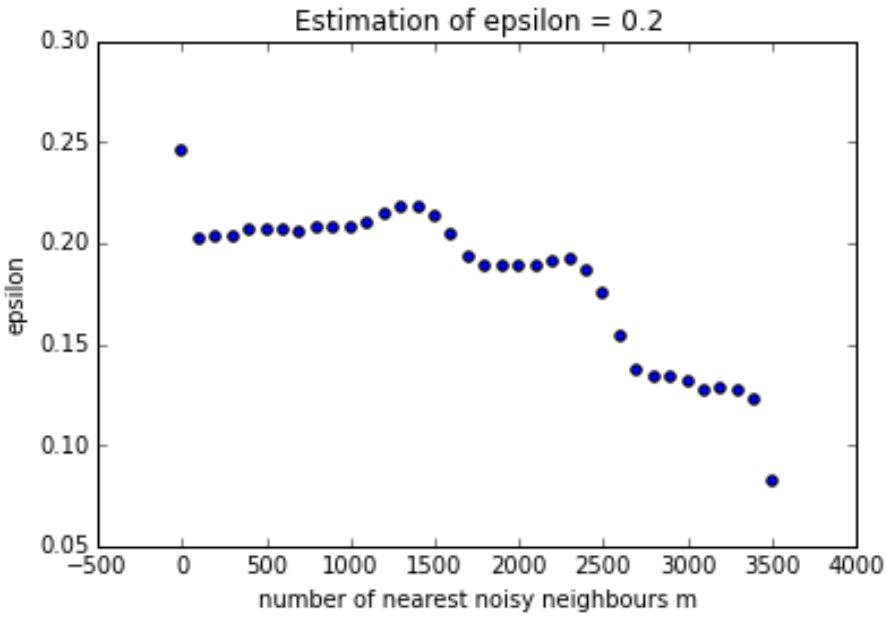
$$\epsilon = \frac{\sum_{k=1}^{M} \mathbb{E}_I^k(\mathbb{E}_I^k - \mathbb{E}_{\complement I}^k)}{\sum_{k=1}^{M} (\mathbb{E}_I^k)^2} \tag{17}$$

The question of the choice of the subdivision $(\mathcal{A}_k)_{k=1..M}$ remains. In our paper, we decide to take $M = |I|$. For an fixed integer m, $(\mathcal{A}_k)$ will be the nearest neighbors of $Y_k \in I$ such as there are exactly m nearest neighbours that are in $\complement I$. To know which value of m to pick to get a good approximation of the noise noise rate, we plot ϵ as a function of m. The method is tested on two sets (Figure 5). The first set is a sample of 1000 points generated with a Bernoulli distribution. One tenth of the set remains clean, the other points are noised with a noise rate $\epsilon = .2$. The graph in Figure 5.5a shows that epsilon is well estimated from a certain number of nearest noisy neighbours m.
Now, let the method be test on a real data set. We take the Banana dataset from the UCI repository [4]. The two thirds of the sample points are being noised with $\epsilon = 0.2$. We can see that the Figure 5.5b is a stair step plot where the first step gives a good approximation

of ϵ.
Following these two examples, the number of nearest noisy neighbors should not be too high in order to have a good estimation of the noise rate. The behaviour of the plots for high values of m depends not only on the distribution of the clean and noisy labels but also on the topology of the dataset.



(a) Estimation of ϵ on the generated sample



(b) Estimation of ϵ on the Banana Dataset

Figure 5: Estimation of the noise rate ϵ

6 Conclusions

We have introduced a modified Boosting algorithm in order to improve the performance of weak learners on datasets where a small amount of instances are, unlike the other examples of the set, not subject to flipping label noise at a rate $\epsilon < \frac{1}{2}$. When starting with the study of loss functions we introduced a first algorithm derived from a Function Gradient Descent. This new algorithm called Core AdaBoost has similarities with AdaBoost. In particular, the weights behave the same against misclassification. Besides, the optimal step sizes α, which are also the votes, may not be defined. From the failure of the Core AdaBoost was introduced a new method : the Stochastic Core AdaBoost. In this new algorithm, the training set is resampled at each iteration and the α are taken to verify a proposition related to stochastic approximation. The Stochastic Core AdaBoost may give the best performance when tested on large datasets. In addition to this new Boosting, we have defined a method to estimate the noise rate ϵ from a dataset that contains clean and noisy labels. With assumptions on the distribution, ϵ minimizes a certain polynomial function of any division of the dataset into subsets. By using the subsets as the nearest neighbors of each examples with clean labels, ϵ is well estimated provided each subset contains a rather low number of noisy neighbours.
This paper, through the new AdaBoost algorithm, tends to spread idea of classification problems with clean and noisy labels. This technique may look like the theory of semi supervised learning, even if the additional data have more information. One interesting future direction could be the comparison of this Boosting method with semi-supervised techniques. Also, in this paper we only present the boosting technique derived from the AdaBoost. Other Boosting algorithms such as the LogiBoost can be adapted version for this particular problem. The case of LogiBoost is furthermore interesting because it shows some robustness against noise labelling. But as said previously, other learning techniques different from Boosting can be developed regarding this new problem.

Proof of Core AdaBoost

This section is a proof of the new AdaBoost algorithm defined in 3.3.
Assume we have the data (X_i, Y_i) for $i = 1...N$, where $Y_i \in \{-1, 1\}$. $I = [\![1; k]\!]$ is the clean set, $\complement I = [\![k+1; N]\!]$ is the noisy set. We apply the function gradient descent algorithm on

$$R_L^N(f) = \sum_{i=1}^{N} L(\tilde{Y}_i, f(X_i))$$

with

$$\begin{cases} L_{|I}(\tilde{Y}, f)) = exp(-\tilde{Y}f) \\ L_{|\complement I}(\tilde{Y}, f)) = \dfrac{1-\epsilon}{1-2\epsilon} exp(-\tilde{Y}f) - \dfrac{\epsilon}{1-2\epsilon} exp(\tilde{Y}f) \end{cases}$$

The aggregated classifier is $H(.)$. At the next round we had a classifier $f(.)$ with a coefficient β so that

$$\beta, f(.) = \arg\min_{\alpha, h(.)} \sum_{i=1}^{N} L(\tilde{Y}_i, H(X_i) + \alpha h(X_i))$$

$$= \arg\min_{\alpha, h(.)} \sum_{i=1}^{k} exp(-\tilde{Y}_i H(X_i) - \alpha \tilde{Y}_i h(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(-\tilde{Y}_i H(X_i) - \alpha \tilde{Y}_i h(X_i)) - \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i) + \alpha \tilde{Y}_i h(X_i)) \quad (18)$$

$$= \arg\min_{\alpha, h(.)} \sum_{i=1}^{k} exp(-\tilde{Y}_i H(X_i)) exp(-\alpha \tilde{Y}_i h(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(-\tilde{Y}_i H(X_i)) exp(-\alpha \tilde{Y}_i h(X_i)) - \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i)) exp(\alpha \tilde{Y}_i h(X_i)) \quad (19)$$

The trick here is to distinguish the points where $h(X_i) = \tilde{Y}_i$ and $h(X_i) \neq \tilde{Y}_i$:

$$(19) = \arg\min_{\alpha, h(.)} \sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)(exp(-\tilde{Y}_i H(X_i)) exp(\alpha) + \sum_{i=1}^{k} (1 - \mathbb{1}(h(X_i) \neq \tilde{Y}_i))(exp(-\tilde{Y}_i H(X_i)) exp(-\alpha)$$
$$+ \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i) exp(-\tilde{Y}_i H(X_i)) exp(\alpha) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} (1 - \mathbb{1}(h(X_i) \neq \tilde{Y}_i)) exp(-\tilde{Y}_i H(X_i)) exp(-\alpha)$$
$$- \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i) exp(\tilde{Y}_i H(X_i)) exp(-\alpha) - \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} (1 - \mathbb{1}(h(X_i) \neq \tilde{Y}_i)) exp(\tilde{Y}_i H(X_i)) exp(\alpha)$$

(20)

After grouping terms with $\mathbb{1}(h(X_i) \neq \tilde{Y}_i))$ we have :

$$(20) = \arg\min_{\alpha,h(.)} 2sh(\alpha)[\sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i))$$
$$+ \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(\tilde{Y}_i H(X_i))] + exp(-\alpha) \sum_{i=1}^{k} exp(-\tilde{Y}_i H(X_i))$$
$$+ \frac{1-\epsilon}{1-2\epsilon} exp(-\alpha) \sum_{i=k+1}^{N} exp(-\tilde{Y}_i H(X_i)) - \frac{\epsilon}{1-2\epsilon} exp(\alpha) \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i)) \quad (21)$$

Notice that the only dependance on h is $2sh(\alpha)(\sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)(exp(-\tilde{Y}_i H(X_i))$.Thus, assuming that $\alpha \geq 0$:

$$f(.) = \arg\min_{h(.)} \sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)(exp(-\tilde{Y}_i H(X_i) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)(exp(-\tilde{Y}_i H(X_i))$$
$$+ \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(\tilde{Y}_i H(X_i))) \quad (22)$$

This equation defines the weights for the new AdaBoost algorithm :

$$\omega_i = \begin{cases} exp(-\tilde{Y}_i H(X_i) \text{ if } i \in I \\ \frac{1-\epsilon}{1-2\epsilon}(exp(-\tilde{Y}_i H(X_i)) + \frac{\epsilon}{1-2\epsilon} exp(\tilde{Y}_i H(X_i))) \text{ if } i \in \complement I \end{cases}$$

Now, to obtain β, we find α so that the derivative of (22) is equal to 0 :

$$(exp(\alpha)+exp(-\alpha))[\sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i))$$
$$+ \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(\tilde{Y}_i H(X_i))] - exp(-\alpha) \sum_{i=1}^{k} exp(-\tilde{Y}_i H(X_i))$$
$$- \frac{1-\epsilon}{1-2\epsilon} exp(-\alpha) \sum_{i=k+1}^{N} exp(-\tilde{Y}_i H(X_i)) - \frac{\epsilon}{1-2\epsilon} exp(\alpha) \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i)) = 0 \quad (23)$$

By grouping terms with $exp(\alpha)$ and terms with $exp(-\alpha)$ we obtain:

$$exp(\alpha)[\sum_{i=1}^{k} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i))$$
$$- \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) = \tilde{Y}_i)exp(\tilde{Y}_i H(X_i))]$$
$$= exp(-\alpha)[\sum_{i=1}^{k} \mathbb{1}(h(X_i) = \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i)) + \frac{1-\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) = \tilde{Y}_i)exp(-\tilde{Y}_i H(X_i))$$
$$- \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} \mathbb{1}(h(X_i) \neq \tilde{Y}_i)exp(\tilde{Y}_i H(X_i))] \quad (24)$$

By highlighting ω_i in this equation, we get :

$$\alpha = \frac{1}{2} \log(\frac{\sum_{i=1}^{N} \omega_i \mathbb{1}(h(X_i) = \tilde{Y}_i) - \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i))}{\sum_{i=1}^{N} \omega_i \mathbb{1}(h(X_i) \neq \tilde{Y}_i) - \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i))})$$

By having the same notation for the original AdaBoost $err = \sum_{i=1}^{N} \omega_i \mathbb{1}(h(X_i) \neq \tilde{Y}_i)$ we have :

$$\alpha = \frac{1}{2}\log(\frac{1 - err - a_\epsilon}{err - a_\epsilon})$$

with $a_\epsilon = \frac{\epsilon}{1-2\epsilon} \sum_{i=k+1}^{N} exp(\tilde{Y}_i H(X_i))$
The condition to have $\alpha > 0$ is the same condition for the AdaBoost :

$$\alpha > 0 \Leftrightarrow err < \frac{1}{2}$$

References

[1] P.L. BARTLETT, M.I. JORDAN, and J.D. McAULIFFE. "Convexity, classification and risk bounds". In: *Journal of the American Statistical Association* 101 (2006), pp. 138–156.

[2] J. BOOTKRAJANG and A. KABAN. "Boosting in the presence of the label noise". In: *UAI2013* (2013).

[3] L. BREIMAN. "Arcing classifiers (with discussion)". In: *The Annals of Statistics* 26 (1998), pp. 801–849.

[4] J. FRANK and A. ASUNCION. *UCI Machine Learning Repository*. `http://archive.ics.uci.edu/ml`. 2010.

[5] Y. FREUND. "An adaptive version of the boost by majority algorithm". In: *Machine Learning* 43.2 (2001), pp. 293–318.

[6] Y. FREUND and R. SCHAPIRE. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55 (1997), pp. 119–139.

[7] J. FRIEDMAN, T. HASTIE, and R. TIBSHIRANI. "Additive Logistic Regression: a Statistical View of Boosting". In: *Annals of Statistics* 28 (1998).

[8] M. LEDOUX and M. TALAGRAND. *Probability in Banach Spaces*. New York: Springer, 1991.

[9] T. LIU and D. TAO. "Classification with Noisy Labels by Importance Reweighting". In: *CoRR* (2014), pp. –1–1.

[10] H. ROBBINS and S. MONRO. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22 (1951), p. 400.

[11] A. VEZHNEVETS and V. VEZHNEVETS. "Modest AdaBoost. Teaching AdaBoost to Generalize Better". In: *GraphiCon* (2005).

[12] T. ZHANG. "Statistical behavior and consistency of classification methods based on convex risk minimization". In: *Annals of Statistics* 32 (2004), pp. 56–85.