Chapter 1
Online learning of a weighted selective naive Bayes classifier with non-convex optimization

C. Hue, M. Boullé, V. Lemaire

Abstract We study supervised classification for data streams with a high number of input variables. The basic naïve Bayes classifier is attractive for its simplicity and performance when the strong assumption of conditional independence is valid. Variables selection and models averaging are two common ways to improve this model. This process leads to manipulate a weighted naïve Bayes classifier. We focus here on direct estimation of weighted naïve Bayes classifiers. We propose a sparse regularization of the model log-likelihood which takes into account the information contained in each input variable. The sparse regularized likelihood being non convex, we propose an online gradient algorithm using mini-batches and a post-optimization to avoid local minima. In our experiments we first study first the optimization quality, then the classifier performance according to its parameterization. These results confirm the effectiveness of our approach.

Key words: supervised classification, naïve Bayes classifier, non-convex optimization, stochastic optimization, variables selection, sparse regularisation

1.1 Introduction

Due to a continuous increase of storage capacities, data acquisition and processing have deeply evolved during these last decades. Henceforth, it is common to process data including a very large number of variables and data amount are such massive that it hardly seems possible to fully load them : online processing is then applied and data are only seen once. In this context, we consider the supervised classification problem where Y is a target cate-

Hue Carine
Orange Labs Lannion, 2 avenue Pierre Marzin, 22300 Lannion, emailFirstname.Name@orange.com

gorical variable with J modalities $C_1, \ldots, C_J$ and $X = (X_1, \ldots, X_K)$ is the set of K explicative variables, numerical or categorical. We focus on naïve Bayes classifiers family. The explicative variables are assumed to be independent conditionally to the target variable. This "naïve" assumption allows us to compute the model directly from the univariate conditional estimates. For an instance denoted n, the probability of the target modality C conditionally to the explicative variables values is computed according to the formulae :

$$P_w(Y = C|X = x^n) = \frac{P(Y = C)\prod_{k=1}^{K} p(x_k^n|C)^{w_k}}{\sum_{j=1}^{J} P(C_j)\prod_{k=1}^{K} p(x_k^n|C_j)^{w_k}} \tag{1.1}$$

We consider here that estimates of prior probabilities $P(Y = C_j)$ and of conditional probabilities $p(x_k|C_j)$ are available. In our experiments, these probabilities will be estimated using univariate discretization or grouping according to the MODL method (cf. [Boullé, 2007]). The univariate probabilities being given, a weighted naïve Bayes classifier is completely described by its variable weight vector $W = (w_1, w_2, \ldots, w_K)$. Within this classifiers family, we can distinguish :

- classifiers with boolean weights. By browsing through all the possible values for the weight vector, we can compute the MAP classifier, i.e. the classifier which maximizes the training data conditional likelihood. However, when the variables number is high, such a browsing is infeasible and only a sub-optimal browsing of the space $\{0, 1\}^K$ can be completed.
- classifiers with continuous weights in $[0, 1]^K$. Such classifiers can be obtained by averaging classifiers with boolean weights with a weighting proportionnal to the posterior probability of the model [Hoeting et al., 1999] or proportionally to their compression rate [Boullé, 2007]. However, for databases with a very high number of variables, we observe that the models issued from averaging keep a lot of variables that make the obtained classifiers both costly to deploy and difficult to interpret.

In this present work, we are interested in direct estimation of the weight vector by optimisation in $[0, 1]^K$ of the regularized log-likelihood. Our main expectation is to obtain by this way robust models with less variables and equivalent performance. Preliminary works [Guigourès and Boullé, 2011] have shown the interest of such a direct weights estimation. The remainder of this paper is organized as follows : in section 1.2 we present a sparse regularization and in section 1.3 the set up of an online algorithm, anytime and with limited budget dedicated to the optimization of the regularized criterion. Experiments are presented in section 1.4, before the conclusion and future works statement.

1.2 Construction of a regularized criterion

Given a dataset $D_N = (x_n, y_n)_{n=1}^N$, we are looking for the minimization of the negative log-likelihood which is given by :

$$ll(w, D_N) = -\sum_{n=1}^{N} \left(\log P(Y = y^n) + \sum_{k=1}^{K} \log p(x_k^n | y^n)^{w_k} - \log \left(\sum_{j=1}^{J} P(C_j) \prod_{k=1}^{K} p(x_k^n | C_j)^{w_k} \right) \right) \tag{1.2}$$

Considered as a classical optimization problem, the regularization of the log-likelihood is performed by the addition of a regularization term (or prior term) which express constraints we wish force on the weight vector W. The regularized criterion is of the form :

$$CR^{D_N}(w) = -\sum_{n=1}^{N} ll(w, z_n = (x_n, y_n)) + \lambda f(w, X_1, \ldots, X_K) \tag{1.3}$$

where ll refers to the log-likelihood, f is the regularization function, and λ the regularization weight. Several objectives have guided our choice for the regularization function :

1. Its sparsity, i.e. she favors the weight vectors composed of as much null components as possible. The L^p norm functions are usually employed with the addition of a regularization term of the form $\sum_{k=1}^{k} |w_k|^p$. All these functions are increasing and hence favour the weight vectors with low components. For $p > 1$, the norm function L^p is convex which makes the optimization easier and renders this function attractive. However, because of its convexity, the minimization of the regularization terme for $p > 1$ does not necessarily lead to variables elimination and the choice $p \leq 1$ favours sparse weight vectors.
2. its ability to take into account a C_k coefficient associated to each explicative variable so that, for equivalent likelihood, the "simple" variables are preferred to "complex" ones. By weighting the term with L^p norm by such a coefficient, we obtain a penalization terme of the form : $\sum_{k=1}^{K} C_k * |w_k|^p$. This coefficient is supposed to be known upstream the optimization. If any knowledge is available, this coefficient is fixed to 1. It can be used to include trade preference. In our case, this coefficient translates the preparation cost of the variable, i.e. the discretization cost for a numerical variable and the grouping cost for a categorical variable respectively described in equations (2.4), resp. (2.7) of [Boullé, 2007].
3. its consistency with the regularized criterion of the MOLD naïve Bayes classifier with binary selection of variables [Boullé, 2007]. In order that the two criterions coincide for $\lambda = 1$ and w_k with boolean values, we finally use the regularization term :
   $f(w, X_1, \ldots, X_K) = \sum_{k=1}^{K} (\log K - 1 + C_k) * w_k^p$

1.3 Optimization algorithm : gradient descent with mini-batches and variable neighborhood search

Let $p_n = P(Y = y^n)$, $p_j = P(C_j)$, $a_{k,n} = p(x_k^n|y^n)$, $a_{k,j} = p(x_k^n|C_j)$ be all constant quantities in this optimization problem.
The regularized criterion to minimize can be written :

$$CR^{D_N}(w) = -\sum_{n=1}^{N}\left\{\log p_n + \sum_{k=1}^{K}(w_k * \log a_{k,n}) - \log\left(\sum_{j=1}^{J} p_j \prod_{k=1}^{K}(a_{k,j})^{w_k}\right)\right\} + \lambda\sum_{k=1}^{K}(\log K - 1 + C_k) * {w_k}^p \tag{1.4}$$

We optimize this criterion subject to the constraint that w takes its values in $[0,1]^K$ in order to obtain interpretable models. This criterion is not convex but differentiable at each weight vector with partial derivative :

$$\frac{\partial CR^{D_N}(w)}{\partial w_\gamma} = -\sum_{n=1}^{N}\left\{\log a_{\gamma,n} - \frac{\sum_{j=1}^{J} p_j \log a_{\gamma,j} \prod_{k=1}^{K}(a_{k,j})^{w_k}}{\sum_{j=1}^{J} p_j \prod_{k=1}^{K}(a_{k,j})^{w_k}}\right\} + \lambda(\log K - 1 + C_k) * p * {w_\gamma}^{p-1} \tag{1.5}$$

The gradient $\nabla CR^{D_N}(w^t)$ is the vector of partial derivatives for $\gamma = 1, \ldots, K$. To respect the constraint that w takes its values in $[0,1]^K$, we have been interested in projected gradient descent algorithm type [Bertsekas, 1976] i.e. a gradient descent algorithm for which, at each iteration, the obtained w vector is projected on $[0,1]^K$.
Several objectives have guided our choice for the algorithmic structure :

1. online algorithm : the algorithm structure is adapted to data stream processing and it does not need the processing of the entire base;
2. anytime algorithm : that the algorithm is interruptible and is able to return the best optimisation given a budgeted computational time.

Within the batch gradient descent algorithm, the weight vector is updated at each iteration t according to the gradient computed on all the instances, and weighted by a step. If the weight vector obtained at iteration t is denoted by w^t, the update at $t+1$ iteration is performed according the equation : $w^{t+1} = P_{[0,1]^K}[w^t - \eta_t \nabla CR^{D_N}(w^t)]$ where the η step may, according to the variants, be a scalar constant or vary across the iterations and/or vary according to the weight vector components. The projection on $[0,1]^K$ just consists in bounding obtained values in interval $[0,1]$. This batch approach assumes that all the database is available to start optimization. In its stochastic version, the update is done by assimilation of the gradient computed on one

Inputs : D : data stream ;
N : historical depth to evaluate the criterion ;
L : batch size used for weights update ;
w_0 : initial weight vector;
η_0 : initial step vector;
Max : maximal number of iterations ;
Tol : Tolerated number of successive degradations;
Outputs: $w^* = argmin CR^D(w)$;
t_{total} = performed iterations number ;
while *(Criterion improvement* **or** *less than Tol successive degradations)* **, and** *iterations number* $<$ *Max* **do**
  t : current iteration index;
  $D_{t,L}$=t-th batch of size L;
  $D_{t,N}$=data historical of size N including at the end of the t-th data batch;
  $w^{t+1} = P_{[0,1]^K}\left(w^t - \eta_t \frac{1}{L} \nabla CR^{D_{t,L}}(w^t)\right)$;
  Computation of η_{t+1};
  Computation of the criterion value data historical of size N : $CR^{D_{t,N}}(w^{t+1})$;
  if *Criterion improvement* **then**
    Best value storage : $w^* = w_{t+1}$;
  else
    Successive degradations counter incrementation;
  end
end

Algorithm 1: Projected gradient descent with mini-batches (PGDMB)

single instance. The gradient descent may turn out to be chaotic if the gradient variance from one instance to another one is high. Aiming for an online approach, we have retained a variant mixing batch and stochastic, namely mini-batch approach [Dekel et al., 2012] which consists in directing the descent according to gradients computed on successives data batches denoted by L. In order that descent paths are comparable when mini-batches size vary, we used a gradient standardized to the mini-batches sizes. The projected gradient descent with mini-batches is summarized in Algorithm 1.

The optimal value for step η_t has been the subject of several researches leading to more or less costly algorithms. We have opted for the Rprop method [Riedmiller and Braun, 1993] : the step computation is specific for each vector component i.e. η is a step vector of dimension K, and each vector component is multiplied by a factor which is bigger, resp. smaller than 1, if the partial derivative sign change, resp. doesn't change from one iteration to another. As far as the computational complexity is concerned, each iteration needs a criterion evaluation on a sample of size N that is to say a $O(K * N)$ complexity. The classical batch algorithm is obtained for $L = N$ and the stochastic one for $L = 1$.

As the criterion to be optimized is non convex, it often shows many local minimums towards which such a gradient descent may converge. In that case, it is common to start several gradient descents with distinct random

Inputs : T : total maximal number of iterations;
NeighSize: initial neighborhood size;
Inputs : (PGDMB) : D : data stream ;
N : historical depth to evaluate the criterion ;
L : batch size used for weights update ;
w_0 : initial weight vector;
η_0 : initial step vector ;
Max : maximal number of iterations for one PGDMB optimisation;
Tol: tolerated number of successive degradations;
Outputs: $w^* = argminCR^D(w)$
Initialisation of $w_1^0 = 0.5^K$;
Initialisation $w^* = w_1^0$;
Initialisation $SumT = 0$;
while $SumT < T$ **do**
  Computation of $(w_m^*, t_{total}^m) = PGDMB(D, N, L, w_m^0, \eta_0, \text{Max}, \text{Tol})$;
  $SumT = SumT + t_{total}^m$;
  if *Improvement on* w^* **then**
    Storage of $w^* = w_m^*$
  else
    NeighSize $= min(2 * \text{NeighSize}, 1)$
  end
  $w_{m+1}^0 = P_{[0,1]^K}(w_m^* + Random([-\text{NeighSize}, \text{NeighSize}]))$;
end

Algorithm 2: Projected gradient descent with variable neighbor search (PGDMB-VNS)

initialisations (multi-start approach) in the hope that one of these descent paths converges to the global minimum of the criterion. In order to make the optimization efficient and to not waste computational time at the beginning of each descent, it is also possible to modify the solution obtained after a given number of iterations in order to get out of a potential valley containing a local minimum. The current solution is regularly randomized within a neighborhood of variable size. This randomization is inspired from the metaheuristic Variable Neighborhood Search [Hansen and Mladenovic, 2001]. Our approach denoted PGDMB-VNS is described in Algorithm 2. It can be noticed that, for a neighborhood that completely covers $[0, 1]^K$, the PGDMB-VNS algorithm is equivaleent to a multi-start algorithm with random initialisations. Besides, let precise that the random perturbation can lead to a non-null component for a weight set to zero after a precedent run. One variable can re-appear during the data stream reading. The PGDMB-VNS algorithm is anytime in the sense that an estimation of the criterion argmin is available at the end of the first gradient descent and that she is afterwards improved according to the available budget and interruptible at any time. Its entire complexity is in $O(T * K * N)$ where T is the total number of budgeted iterations.

1.4 Experiments

The first experiments goal is to evaluate the optimization quality obtained with PGDMB-VNS according to the mini-batches size L and to the iterations total number T. To study the intrinsic quality regardless of the associated classifier statistical performance, we have set the λ weight value to 0, that means we directly optimize the non regularized likelihood. The second part of the experiments deals with statistical performance of classifier obtained by optimization of regularized criterion ($\lambda \neq 0$).
For the whole experiments, the parameters for PGDMB algorithm are set to the following values :

- $w_0 = \{0.5\}^K$
- $\eta_0 = \{10^{-2}\}^K$ with a multiplication by 0.5, resp. 1.2, in case of sign change, resp. no sign change, between two succesive gradients
- Max = 100 the iteration maximal number (i.e. the treated mini-batches number). We have checked that this threshold had never been reached for the 36 tested bases.
- Tol = 5 the authorized successive degradations number

Improvement criterion is considered for a decreasing of at least $\epsilon = 10^{-4}$ with regard to the precedent criterion value. The weights smaller than 10^{-3} are set to 0.

The whole experiments have been done in 10-fold-cross-validation on the 36 UCI bases described in table 1.1. In the results presentation, 'SNB' designs performance of a Bayes classifier averaged with compression rate [Boullé, 2007].

1.4.1 Experiments on optimization quality

First of all, we have studied the PGDMB algorithm performance, that is to say the projected gradient descent algorithm without post-optimisation, according to the mini-batch size denoted L. We have chosen as optimization quality indicator the compression rate which measures the negative logarithm of the model likelihood, normalized by the Shannon entropy. The closer the rate is to 1, the higher is the model likelihood. For model less competitive than the random model, compression rate is negative. The compression rate value on train data is then a good indicator of the optimization quality as the non regularized criterion is reduced to the negative log-likelihood.
Figure 1.1 presents train and test compression rate averaged on 36 UCI bases for various mini-batches sizes $L = 100, 1000, N$. In the last case, the choice

Base	Ni	Nv	Nc	Base	Ni	Nv	Nc
Abalone	4177	8	28	Mushroom	8416	22	2
Adult	48842	15	2	PenDigits	10992	16	10
Australian	690	14	2	Phoneme	2254	256	5
Breast	699	10	2	Pima	768	8	2
Bupa	345	6	2	Satimage	768	8	6
Crx	690	15	2	Segmentation	2310	19	7
Flag	194	29	8	Shuttle	58000	9	7
German	1000	24	2	SickEuthyroid	3163	25	2
Glass	214	10	6	Sonar	208	60	2
Heart	270	13	2	Soybean	376	35	19
Hepatitis	155	19	2	Spam	4307	57	2
Horsecolic	368	27	2	Thyroid	7200	21	3
Hypothyroid	3163	25	2	Tictactoe	958	9	2
Ionospehre	351	34	2	Vehicle	846	18	4
Iris	150	4	3	Waveform	5000	21	3
LED	1000	7	10	WaveformNoise	5000	40	3
LED17	10000	24	10	Wine	178	13	3
Letter	20000	16	26	Yeast	1484	9	10

Table 1.1 Description of the 36 UCI bases : Ni=instances number, Nv=initial number of variables, Nc=class number.

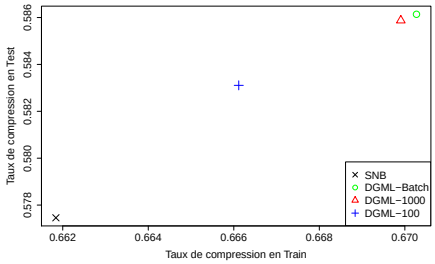

Fig. 1.1 Train and test mean compression rate for 36 UCI bases

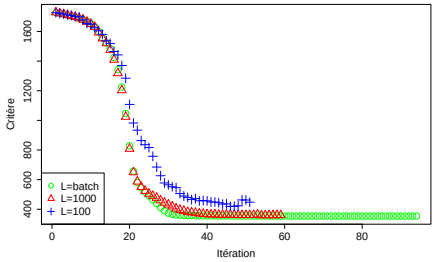

Fig. 1.2 Criterion convergence paths according to the mini-batches size (PGDMB) for the Phoneme base

$L = N$ amounts to a batch algorithm. Train and test compression rates obtained with SNB MODL classifier [Boullé, 2007] serve as a reference. Obtained results indicate that, the smaller the mini-batches size is, the more the optimization quality deteriorates. Morevover, the results obtained for $L = 1000$ and $L = N$ are very similar. The train compression rate is significantly better for batch mode than for $L = 1000$ for 8 of the 36 bases.
Figure 1.2 presents as an example the criterion values serie during optimization according to the mini-batches size $L = 100, 1000, N$ for Phoneme base. For the whole 36 bases, the convergence is faster but more chaotic when the mini-batches size decreases.

We have compared the optimization quality for a PGDMB algorithm without post-optimization on the one hand, and for a post-optimized PGDMB on the other hand. Several post-optimizations have been tested : with multi-start (PGDMB-MS) or with variable neighborhood search (PGDMB-VNS). To get a complexity with the same order of size as that of the univariate MODL pretreatement, that is to say $O(K * N * \log(K * N))$, we have fixed the total number of authorized iterations T proportional to $\log(K * N)$. More precisely, we have chosen $T = \log(K * N) * 2^{\text{PostOptiLevel}}$ where PostOptiLevel is an integer which enables to tune the desired post-optimization level.
For each of the two post-optimization ways, we have studied the influence of the post-optimization level OptiLevel $= 3, 4, 5$. Since the post-optimized algorithm stores as the best solution is encountered, the post-optimization can only improve the train compression rate. We have measured as a first step if the improvement was significant or not. For a MS post-optimization, the train compression rate is significantly improved for resp. $7, 16, 18$ of the 36 bases with a post-optimisation level equal resp. to $3, 4, 5$. For a VNS post-optimization, the train compression rate is significantly improved for resp. $18, 19, 23$ of the 36 bases with a post-optimization level equal resp. to $3, 4, 5$. The VNS post-optimization seems then better than the MS post-optimization : the guided exploration within a variable sized neighborhood from the best minimum encountered enables a more fruitful exploration than a purely random exploration.
Figure 1.3 illustrates this iterations "waste" phenomenon with MS post-optimization at the beginning of each start.

Experiments presented in this section have illustrated the effect of the mini-batches size on the optimization quality. They have also illustrated that the higher is this size, the better the optimization quality is and that the VNS post-optimization is superior over the MS post-optimization. We then retain for the rest of the experiments a PGDMB-VNS algorithm with mini-batchs size fixed to $L = 1000$ and an optimization level set to PostOptiLevel $= 5$.

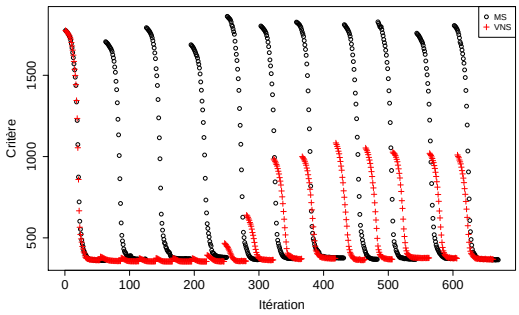

Fig. 1.3 Criterion convergence paths according to the post-optimization type for the Phoneme base and a post-optimization level equal to 5

1.4.2 Regularized classifier performance

We present the classifier performance according to the setting of the regularization weight λ and the p exponent of the function $|w_k|^p$. Three values have been tested for $\lambda = 0.01, 0.1, 0.5$ and for $p = 0.5, 1, 2$. The performance for AUC indicator for the nine regularized classifiers are presented in Figure 1.4 and, taking as a reference, the performance of the non-regularized classifier obtained with $\lambda = 0$ and of the SNB classifier.

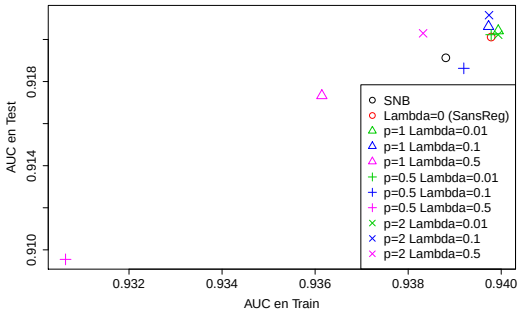

Fig. 1.4 Train and test AUC averaged for 36 UCI bases according to the weight and the type of regularization

For the highest regularization weight, that is to say $\lambda = 0.5$ (in purple in the Figure), the AUC performance are deteriorated with regards to the performance obtained without regularization (red circles in the Figure) whatever

the p value. On the other hand, for the other weight values $\lambda = 0.01$ and $\lambda = 0.1$, the performance are similar for all p values and slightly superior or equal on average to those of the non-regularized classifier. These two regularisation weights lead to statistical performance equivalent to those of non regularized classifier.

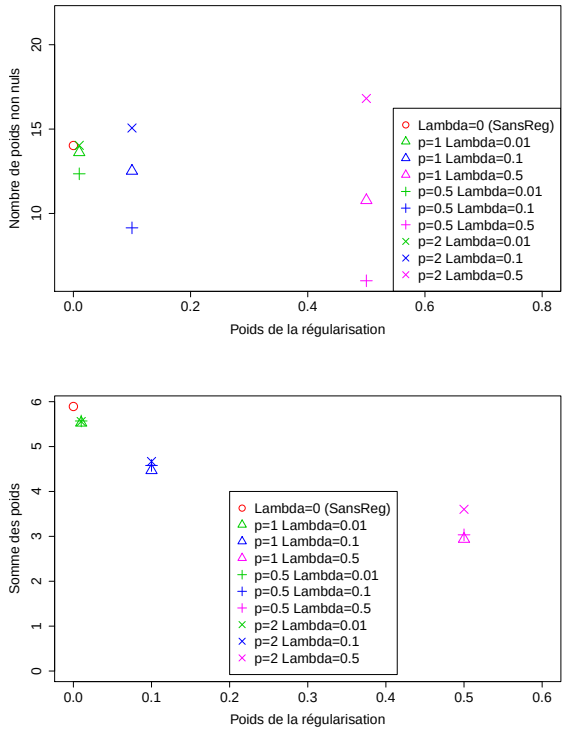

Fig. 1.5 Kept variable number and weight sum averaged for 36 UCI bases according to the weight and the type of regularization

Furthermore, to study the obtained classifiers sparsity, Figure 1.5 presents the kept variables number and their wiehgts sum. First, it shows that the smaller p, the smaller the non-null weights number. The quadratic regularization ($p = 2$) leads to sparser classifiers. Among the regularization with absolue value ($p = 1$) and the squared root one ($p = 0.5$), the second one enables the most important reduction of the kept variables number. As far as the weights sum is concerned, all the regularization exponent enable to reduce on average the weights sum. Moreover, given a λ weight, the quadratic regularization has a less important impact on the weights sum reduction than

the two others regularizations whose performance are very close for this indicator.
Considering both aspects of statistical performance and classifier sparsity, the compromise $p = 1$ and $\lambda = 0.1$ seems the most favorable. Without deteriorating the non regularized classifier performance, it enables a significant reduction of the selected variables number. This reduction makes the classifier more interpretable and less complex to deploy.

1.5 Conclusion

We proposed a sparse regularization of the log-likelihood for a weighted naïve Bayes classifier. We described and experimented a gradient descent algorithm which treats online mini-batches data and optimizes the weights classifier through a more or less extensive exploration of the current optimization depending according to the iterations budget. The experiments have shown the interest of using mini-batches and post-optimization. Moreover, a parametrization study of the regularization points out that the optimal choice was a regularization term with the $L1$ norm and a weight $\lambda = 0.1$. Experiments on substantially larger databases are necessary to evaluate the approach performance on real data streams and will be the subject of future works.

References

[Bertsekas, 1976] Bertsekas (1976). On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*.

[Boullé, 2007] Boullé (2007). *Recherche d'une représentation des données efficace pour la fouille des grandes bases de données*. PhD thesis, Ecole Nationale Supérieure des Télécommunications.

[Boullé, 2007] Boullé (2007). Compression-based averaging of selective naive bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685.

[Dekel et al., 2012] Dekel, Gilad-Bachrach, Shamir, and Xiao (2012). Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202.

[Guigourès and Boullé, 2011] Guigourès and Boullé (2011). Optimisation directe des poids de modèles dans un prédicteur bayésien naif moyenné. In *13èmes Journées Francophones "Extraction et Gestion de Connaissances" (EGC 2011)*, pages 77–82.

[Hansen and Mladenovic, 2001] Hansen and Mladenovic (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130.

[Hoeting et al., 1999] Hoeting, Madigan, Raftery, and Volinsky (1999). Bayesian model averaging : A tutorial. *Statistical Science*, 14(4):382–401.

[Riedmiller and Braun, 1993] Riedmiller and Braun (1993). A direct adaptive method for faster backpropagation learning: The Rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591.