Journal of Machine Learning Research 13 (2012) 3539-3583 Submitted 4/10; Revised 4/12; Published 12/12

Regularized Bundle Methods for Convex and Non-Convex Risks

Trinh-Minh-Tri Do[*] TRI.DO@IDIAP.CH
Idiap Research Institute
Rue Marconi 19
1920 Martigny, Switzerland

Thierry Artières THIERRY.ARTIERES@LIP6.FR
LIP6 - Université Pierre et Marie Curie
104 avenue du président Kennedy
75016 Paris, France

Editor: Tony Jebara

Abstract

Machine learning is most often cast as an optimization problem. Ideally, one expects a convex objective function to rely on efficient convex optimizers with nice guarantees such as no local optima. Yet, non-convexity is very frequent in practice and it may sometimes be inappropriate to look for convexity at any price. Alternatively one can decide not to limit *a priori* the modeling expressivity to models whose learning may be solved by convex optimization and rely on non-convex optimization algorithms. The main motivation of this work is to provide efficient and scalable algorithms for non-convex optimization. We focus on regularized unconstrained optimization problems which cover a large number of modern machine learning problems such as logistic regression, conditional random fields, large margin estimation, etc. We propose a novel algorithm for minimizing a regularized objective that is able to handle convex and non-convex, smooth and non-smooth risks. The algorithm is based on the cutting plane technique and on the idea of exploiting the regularization term in the objective function. It may be thought as a limited memory extension of convex regularized bundle methods for dealing with convex and non convex risks. In case the risk is convex the algorithm is proved to converge to a stationary solution with accuracy ε with a rate $O(1/\lambda\varepsilon)$ where λ is the regularization parameter of the objective function under the assumption of a Lipschitz empirical risk. In case the risk is not convex getting such a proof is more difficult and requires a stronger and more disputable assumption. Yet we provide experimental results on artificial test problems, and on five standard and difficult machine learning problems that are cast as convex and non-convex optimization problems that show how our algorithm compares well in practice with state of the art optimization algorithms.

Keywords: optimization, non-convex, non-smooth, cutting plane, bundle method, regularized risk

1. Introduction

Machine learning is most often cast as an optimization problem where one looks for the best model among a parameterized family of models. The best model is defined as the one with the set of parameters that minimizes an objective function (i.e. criterion). For some years now machine learning community aimed at designing new models in such a way that the resulting objective function is convex. Doing so brings the fundamental advantage that one can rely on efficient convex optimiza-

[*] Part of this work was done when TMT Do was at LIP6.

tion algorithms, with nice guarantees such as no local optima and easier theoretical analysis (e.g. for the convergence rate). For instance logistic regression, support vector machine, maximum margin Markov network, and conditional random fields have found widespread use in basic machine learning applications.

However, such a “simple convex modeling” may actually be outperformed by non-convex modeling in some important applications. For example on MNIST database, convex Gaussian-SVM reaches 1.4% error rate vs. 0.53% for non-convex convolutional nets (Jarrett et al., 2009).[1] Also non-convexity is much more frequent than convexity “in real life”. A number of problems that machine learning researchers face today may not be easily cast as convex optimization problems without limiting *a priori* the expressivity of the models used and the potential of the models to learn (LeCun et al., 1998; Collobert et al., 2006; Bengio and Lecun, 2007). First, many real-world problems need complicated models whose learning requires solving non-convex optimization problems. For instance, models with non-convex discriminant function such as neural networks and hidden Markov models (HMMs) have become classical and reference models for many difficult tasks in vision and speech. Second, non-convexity of objective function naturally arises in learning paradigms such as unsupervised and semi-supervised learning as well as in transductive SVM, etc (Chapelle et al., 2006; Joachims, 1999).

Two strategies have been investigated to handle non-convexity in machine learning approaches. Few works attempted to use convex relaxation technique in order to transform an original non-convex problem into a convex one, this is a kind of “convexity at any price” strategy. Convex relaxation mechanics strongly depend on the application, there is no principled method for turning a non-convex problem to a convex one. It has been used in maximum margin clustering (Xu et al., 2004), transductive SVM (Xu et al., 2008), discriminative unsupervised structured predictors (Xu et al., 2006), large margin CDHMM (Sha and Saul, 2007). However, the robustness of this approach for complex problems is questionable since the use of strong assumptions may lead to poor approximation quality, thus provide poor performance in practice.

Since convex modeling does not cover all real-world problems and convex relaxation techniques are not always easy and robust, few researchers proposed to give up convexity and to focus on non-convex optimization techniques, for instance concave-convex procedure (CCCP) (Yuille and Rangarajan, 2003) and difference of convex (DC) programming (Horst and Thoai, 1999). These non-convex optimization techniques have been successfully applied for some tasks such as ramp loss SVM, non-convex TSVM (Collobert et al., 2006), kernel selection (Argyriou et al., 2006) or non-convex maximum margin clustering (Zhao et al., 2008). Note that these techniques cover only a limited class of problems and require an ad-hoc design for every machine learning problem. For instance, the CCCP can theoretically be applied to any continuous objective function since any such function can be decomposed into the difference of two convex functions, yet reformulating the original function to a concave-convex form may call for mathematical efforts. Furthermore not all decomposition are interesting.

We are concerned here with the development of generic optimization techniques able to deal with the general unconstrained optimization problem

$$\begin{array}{ll} \min_{\mathbf{w}} & f(\mathbf{w}) \\ \textit{with} & f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + R(\mathbf{w}) \end{array} \tag{1}$$

[1] A collection of evaluation results on MNIST data is available at: `http://yann.lecun.com/exdb/mnist`.

where $\mathbf{w} \in \mathbb{R}^D$ are the model parameters and $R(\mathbf{w})$ (the main objective) is a data-fitting measurement to be minimized which we consider to be not necessarily smooth everywhere nor convex. This unconstrained formulation covers many mentioned machine learning problems such as SVM, CRF, M3N, transductive SVM, ramp loss SVM, neural network (Do and Artières, 2010), Gaussian HMM (Do and Artières, 2009). Note that the formulation in Equation 1 does not apply easily to kernel methods which are based on an implicit data transformation (e.g. RBF kernel) and are preferably solved in the dual space. However, there are several methods that can enrich the model flexibility without considering an implicit data transformation. As an example, for low dimensional or sparse data, one could have an explicit and efficient transformation for polynomial kernel. Furthermore, instead of using a predefined implicit transformation one could also learn the explicit data transformation directly such as latent feature discovery based on Boltzmann machine (Hinton et al., 2006). At the end, while not covering kernel tricks, our general optimization problem can be used for learning many powerful non-linear models.

As the problem in Equation 1 is at the heart of many machine learning application, it is important to have an efficient non-convex optimization method for this class of minimization problem. Among candidate families of optimization algorithms, cutting plane methods and bundle based methods are very appealing for optimization problems such as the one in Equation 1 since, as opposed to many gradient descent based methods, it can naturally deal with its non-smooth everywhere feature (Kiwiel, 1985; Gaudioso and Monaco, 1992; Makela, 2002; Makela and Neittaanmaki, 1992; Schramm and Zowe, 1992). However the convergence of bundle methods for non-convex optimization is rather slow in practice. And theoretical results on convergence rate are indeed missing for non-convex objective functions. This explains in our opinion why the use of general non-convex bundle methods is still limited in machine learning. Another reason is the lack of easy-to-use implementation of non-convex bundle methods.

The recent success of convex regularized bundle methods (CRBMs) in machine learning (Smola et al., 2008; Weimer et al.; Joachims et al., 2009) motivated us to investigate extensions of bundle methods for proposing efficient algorithms able to deal with machine learning non-convex optimization problems, which is the core idea of this work. To design such an algorithm, we investigated new optimization algorithms that combines ideas from non-convex bundle methods (NBM) (Kiwiel, 1985) and from CRBMs (Smola et al., 2008). Our algorithm relies on two main contributions, a limited memory variant of bundle methods and the extension of CRBM to non-convex risks.

The limited memory variant may be used in CRBM as well as in our non-convex extension of CRBM. It allows limiting the algorithmic complexity of a single iteration in bundle methods while it is usually increasing (at least quadratically) with the number of iteration, which makes bundle methods not practical for difficult and large scale problems requiring thousands of iterations. We show that our limited memory variant, when included to CRBM, inherits its fast convergence rate in $O(1/\lambda\varepsilon)$ iteration to reach a gap below ε.

Our extension of CRBM to non-convex risks includes the limited memory variant and is designed to make bundle methods scalable for real life non-convex learning problems.[2] This is achieved by making the algorithm focus on the current best solution and by using a specific locality measure for regularized risks. Such a strategy allows fast convergence in practice on difficult and large scale machine learning problems that we investigated. Unfortunately this comes with only weak proof of convergence towards a stationary solution, relying on a moot assumption. In our

[2] The MATLAB implementation of the proposed method is available at `https://forge.lip6.fr/projects/nrbm`.

opinion it is a kind of trade-off, a price to pay to achieve algorithmic efficiency in practice. As a consequence though we provide main theoretical results we do not include our convergence proofs here since these are weak, but these are available in an internal report (Do and Artieres, 2012).

First, in Section 2 we provide background on the cutting plane technique and on bundle methods, and we describe two main existing extensions, the convex regularized bundle method (CRBM) and the non-convex bundle method (NBM). Then, we present in Section 3 our two contributions yielding our algorithm, NRBM, which is a regularized bundle method for non-convex optimization. We propose few variants of our method in Section 3.3 and we discuss in Section 4 the convergence behavior of our method both for convex risks and for non-convex risks. Finally we provide in Section 5 a number of experimental results. We investigate first artificial test problems that show that our algorithm compares well to standard non-convex bundle methods while converging much faster, suggesting our algorithm may make large scale problems practical. Second we compare our algorithms to dedicated state of the arts optimization algorithms for a number of machine learning problems, including standard problems such as learning of transductive support vector machines learning, learning of maximum margin Markov networks, learning conditional random fields, as well as less standard but difficult optimization problems related to discriminative training of complex graphical models for handwriting and speech recognition.

2. Background on Cutting Plane and Bundle Methods

We provide now some background on the cutting plane principle and on optimization methods that have been built on this idea for convex and non-convex objective functions.

2.1 Cutting Plane Principle

Surely the most powerful method for non-smooth optimization is based on *polyhedral approximations*, whose basic element is the *cutting plane* (CP). For a given function $f(\mathbf{w})$, a cutting plane $c_{\mathbf{w}'}(\mathbf{w})$ is a first-order Taylor approximation computed at a particular point $\mathbf{w}'$:

$$f(\mathbf{w}) \approx c_{\mathbf{w}'}(\mathbf{w}) = f(\mathbf{w}') + \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w} - \mathbf{w}' \rangle$$

where $\mathbf{a}_{\mathbf{w}'} \in \partial f(\mathbf{w}')$ is a subgradient of f at $\mathbf{w}'$. For convex function, the subdifferential $\partial f(\mathbf{w}')$ is the set of vectors $\mathbf{a}$ that satisfies: $f(\mathbf{w}) \geq f(\mathbf{w}') + \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w} - \mathbf{w}' \rangle$. The concept of subdifferential is also generalized for non-convex functions, which is defined as the set of vectors $\mathbf{a}$ that satisfies: $f^{\circ}(\mathbf{w}';\mathbf{h}) \geq \langle \mathbf{a}, \mathbf{h} \rangle\ \forall \mathbf{h}$, where $f^{\circ}(\mathbf{w}', \mathbf{h})$ denotes the generalized directional derivative of f at $\mathbf{w}'$ in the direction $\mathbf{h}$.

Go back to the definition of the cutting plane approximation based on Taylor approximation, it may be rewritten as:

$$\begin{array}{ll} c_{\mathbf{w}'}(\mathbf{w}) & = \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w} \rangle + b_{\mathbf{w}'} \\ with & \mathbf{a}_{\mathbf{w}'} \in \partial f(\mathbf{w}') \\ & b_{\mathbf{w}'} = f(\mathbf{w}') - \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w}' \rangle. \end{array} \tag{2}$$

A cutting plane $c_{\mathbf{w}'}$ is an approximation of f which is accurate for $\mathbf{w}$ lying in the vicinity of $\mathbf{w}'$ where the CP is defined, i.e. where the subgradient is computed. The quality of such an approximation and the area where it is accurate depend on higher order information on f such as the Hessian matrix. Figure 1 illustrates the linear approximation implemented by a cutting plane for a one-dimensional function. Importantly, a cutting plane of a convex function f is an underestimator of f.

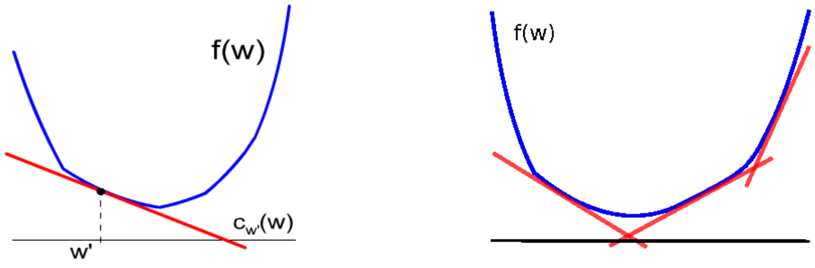


Figure 1: Basic approximation of a function f by a (underestimator) cutting plane at a point $\mathbf{w}'$ (left), and a more accurate approximation by taking the maximum over many cutting planes of f (right).

2.2 Cutting Plane Method for a Convex Objective

The *cutting plane method* has been proposed for the minimization of convex functions. In the case of a convex objective, any cutting plane of the objective f is an underestimator of f. The idea of the cutting plane method is that one can build an accurate approximation function (named g_t hereafter) of f, which is also an underestimator of f, as the maximum over many cutting plane approximation built at different points $\{\mathbf{w}_1, ..., \mathbf{w}_t\}$ as follows:

$$f(\mathbf{w}) \approx g_t(\mathbf{w}) = \max_{j=1..t} \langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w} \rangle + b_{\mathbf{w}_j}. \tag{3}$$

Of course $g_t(\mathbf{w})$ is an underestimator of $f(\mathbf{w})$. It is called the approximation function of f at iteration t.

The cutting plane method aims at iteratively building an increasingly accurate piecewise linear underestimator of the objective function by successively adding new cutting planes to the approximation g of f. If the approximation is good enough, one may hope that the minimum of f and of its approximation g will be very close or even equal. Every iteration, one adds a new cutting plane underestimator built at current solution, yielding a new piece-wise linear underestimator of f as in Equation 3. The minimization of this underestimator approximation is usually called the approximated problem (it is a linear program) and gives a new current solution, etc.

Note that the approximation function may not have a minimum, then artificial bounds may be placed on the points of $\mathbf{w}$, so that the minimization will be carried out over a compact set and consequently a exists.

The cutting plane method is described in Algorithm 1, it is proved to converge in a finite number of iterations to an ε-solution (Bertsekas et al., 2003).

2.3 Bundle Methods for a Convex Risk

Convex bundle method. One of the drawbacks of the cutting plane method is its instability. It may make large steps away from the optimum even when the current solution is close to it. Standard convex bundle method (CBM), also called *proximal cutting plane method* or *proximal bundle method*, tries to overcome this problem by adding to the polyhedral approximation function a regularization

Algorithm 1 Cutting Plane Method (for convex objective function)

```
Input: $\mathbf{w}_1$, $f$, $\varepsilon$
Output: $\mathbf{w}^*$
for $t = 1$ to $\infty$ do
    Compute $\mathbf{a}_{\mathbf{w}_t}$ and $b_{\mathbf{w}_t}$ according to Equation 2
    Minimize $g_t(\mathbf{w})$ (defined as in Equation 3) to get $\mathbf{w}_{t+1} \leftarrow \text{argmin}_{\mathbf{w}} g_t(\mathbf{w})$
    $gap = [\min_{j=1..t} f(\mathbf{w}_j)] - g_t(\mathbf{w}_{t+1})$
    if $gap < \varepsilon$ then return $\mathbf{w}_t$
end for
```

Algorithm 2 Convex Regularized Bundle Method (CRBM)

```
Input: $\mathbf{w}_1$, $R$, $\varepsilon$
Output: $\mathbf{w}^*$
for $t = 1$ to $\infty$ do
    Compute $\mathbf{a}_{\mathbf{w}_t}$ and $b_{\mathbf{w}_t}$ of $R$ at $\mathbf{w}_t$
    $\mathbf{w}_t^* = \text{argmin}_{\mathbf{w} \in \{\mathbf{w}_1,\ldots\mathbf{w}_t\}} f(\mathbf{w})$
    $\tilde{\mathbf{w}}_t \leftarrow \text{argmin}_{\mathbf{w}} g_t(\mathbf{w})$ where $g_t(\mathbf{w})$ is defined as in Equation 6
    $gap_t = f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t)$
    $\mathbf{w}_{t+1} = \tilde{\mathbf{w}}_t$
    if $gap_t < \varepsilon$ then return $\mathbf{w}_t^*$
end for
```

term. The approximation function becomes:

$$f(\mathbf{w}) \approx g_t(\mathbf{w}) = (\mathbf{w} - \mathbf{w}_t)^\top H_t(\mathbf{w} - \mathbf{w}_t) + \max_{j=1..t} \langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w} \rangle + b_{\mathbf{w}_j} \tag{4}$$

where H_t is a positive definite symmetric matrix. The regularization term forces the new solution not to be too far from the current solution. In addition it makes the approximation function have a unique minimum (as long as the Hessian matrix of the regularization term is positive-definite as in our example) without adding artificial constraints. While the approximation function in Equation 4 can be used to generate new points, the standard bundle method also includes a line-search procedure which returns either a serious step (the objective at current solution has significantly decreased) or a null step (the decrease of f is too low and the approximation function should be improved).

Convex regularized bundle method. The convex regularized bundle method (CRBM) (Smola et al., 2008) is an instance of CBM algorithms for dealing with regularized (and convex) risks as in Equation 1. It relies on cutting planes that are built on the risk $R(\mathbf{w})$ only and does not use a line search procedure. Such a linear approximation of the risk $R(\mathbf{w})$ yields a quadratic approximation of the objective $f(\mathbf{w})$:

$$f(\mathbf{w}) \approx \frac{\lambda}{2}\|\mathbf{w}\|^2 + \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w} \rangle + b_{\mathbf{w}'}. \tag{5}$$

These two approximation functions on $R(\mathbf{w})$ and on $f(\mathbf{w})$ are illustrated in Figure 2. Note that this quadratic approximation of $f(\mathbf{w})$ is more accurate than a cutting plane approximation on $f(\mathbf{w})$. Furthermore, this trick avoids adding an artificial regularization term into the approximation problem as in standard bundle methods.

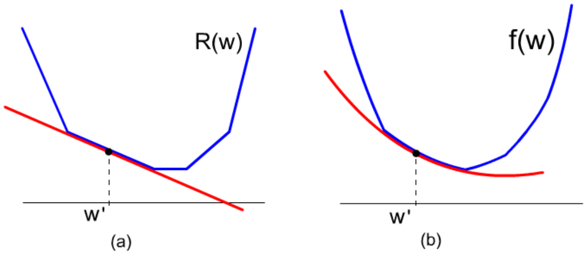


Figure 2: Cutting plane approximations in CRBM : A linear underestimator of $R(\mathbf{w})$ (a), and a quadratic underestimator of $f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + R(\mathbf{w})$ derived from this linear underestimator (b) (Cf. Equation 5)

CRBM is very similar to the cutting plane technique described before, where every iteration a new cutting plane approximation is built (at the current solution) and added to the current approximation function. The approximation of $f(\mathbf{w})$ at iteration t is then:

$$f(\mathbf{w}) \approx g_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max_{j=1..t}\langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j} \tag{6}$$

and the approximation problem is

$$\tilde{\mathbf{w}}_t = \operatorname*{argmin}_{\mathbf{w}} g_t(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max_{j=1..t}\langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j} \tag{7}$$

where $\langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j}$ is the approximation cutting plane of R built at $\mathbf{w}_j$, the solution at iteration j. Importantly, if $R(\mathbf{w})$ is convex then any cutting plane $\langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j}$ is an underestimator of $R(\mathbf{w})$, and its maximum, $\max_{j=1..t}\langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j}$, is also an underestimator approximation of R. Hence, $g_t(\mathbf{w})$ are monotonically increasing quadratic underestimators of $f(\mathbf{w})$ which converge towards $f(\mathbf{w})$ as cutting planes are added.

Minimizing the approximation problem in CRBM. The approximation problem (Equation 7) at iteration t is an SVM-like optimization problem:

$$\begin{aligned} \tilde{\mathbf{w}}_t = \quad & \operatorname{argmin}_{\mathbf{w}} \min_{\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi \\ & \qquad\qquad s.t. \quad \langle \mathbf{a}_j, \mathbf{w}\rangle + b_j \leq \xi \ \ j = 1..t \end{aligned}$$

with $c_j(\mathbf{w}) = \langle \mathbf{a}_j, \mathbf{w}\rangle + b_j$. We can get its dual form easily through Lagrangian mechanics. The Lagrangian of the above optimization problem is:

$$L(\mathbf{w}, \xi, \boldsymbol{\alpha}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi + \sum_{j=1..t} \alpha_j(\langle \mathbf{a}_j, \mathbf{w}\rangle + b_j - \xi)$$

where $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_t)$ are Lagrange multipliers. The solution is given by a saddle point of the Lagrangian, that must be minimized wrt. primal variables ($\mathbf{w}$,ξ) and maximized wrt. Lagrange multipliers. At a saddle point, the derivative of the Lagrangian wrt. ($\mathbf{w}$,ξ) must satisfy:

$$\begin{aligned} \frac{\partial L}{\partial \xi} = 0 \quad & \Longleftrightarrow \sum_{j=1..t} \alpha_j = 1, \\ \frac{\partial L}{\partial \mathbf{w}} = 0 \quad & \Longleftrightarrow \lambda \mathbf{w} = -(\textstyle\sum_{j=1..t} \alpha_j \mathbf{a}_j). \end{aligned}$$

By substituting these results back into the Lagrangian, primal variables $\mathbf{w}$ and ξ disappear and we get the dual problem:

$$\begin{array}{lll} \boldsymbol{\alpha}_t = & \operatorname{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}^t} & -\frac{1}{2\lambda}\|\boldsymbol{\alpha} A_t\|^2 + \boldsymbol{\alpha} B_t \\ & s.t & \alpha_j \geq 0 \; \forall j = 1..t \\ & & \sum_{j=1..t} \alpha_j = 1 \end{array} \quad (8)$$

where $A_t = [\mathbf{a}_1;...;\mathbf{a}_t]$ is a matrix (with $\mathbf{a}_j$ being row vectors), $B_t = [b_1;...;b_t]$ is the vector of scalars and $\boldsymbol{\alpha}$ stands for the (row) vector of Lagrange multipliers (of length t at iteration t). Let $\boldsymbol{\alpha}_t$ be the solution of the above dual problem at iteration t, the solution of the primal problem is given by:

$$\begin{array}{ll} \tilde{\mathbf{w}}_t & = -\frac{\alpha_t A_t}{\lambda}, \\ g_t(\tilde{\mathbf{w}}_t) & = -\frac{1}{2\lambda}\|\boldsymbol{\alpha}_t A_t\|^2 + \boldsymbol{\alpha}_t B_t. \end{array}$$

Convergence rate of CRBM. The convergence of CRBM is proved based on the fact that the gap between the best observed value $f(\mathbf{w}_t^*)$ and the minimum of the approximation function $g_t(\tilde{\mathbf{w}}_t)$ decreases every iteration. Since $g_t(\mathbf{w})$ is an underestimator of $f(\mathbf{w})$, the gap is greater than or equal to the difference between the best observed value $f(\mathbf{w}_t^*)$ and the minimum of $f(\mathbf{w})$. Therefore, if $gap_t \leq \varepsilon$ then $\mathbf{w}_t^*$ is an ε-solution of $f(\mathbf{w})$. By characterizing the decrease of the gap after each iteration, the authors of CRBM proved that the method require $O(1/\lambda\varepsilon)$ iterations to reach a gap below ε (Smola et al., 2008).

2.4 Non-Convex Bundle Methods (NBM)

Bundle methods have also been extended to deal with non-convex functions and have become a standard for minimizing non-smooth and non-convex function.

2.4.1 PRINCIPLE

There are many variants of non-convex bundle algorithm (NBM), with many parameters to tune. We present here a simple description of the method to better stress its main features. Basically NBM works similarly as standard bundle methods by building iteratively an approximation function via the cutting plane technique. However since the objective is no more convex, such an approximation function is not an underestimator of the objective anymore which makes things harder and requires a more complicated algorithm.

Every iteration the algorithm updates a number of quantities, whose set is usually called the *state* of the algorithm, based on the state in previous iteration. The state of the algorithm at iteration t, named $\mathbb{B}_t$, is a set of points, subgradients and locality measures to the current solution. At iteration t, the algorithm performs the following steps:

- ***Determine the search direction.*** This is done through minimizing the approximated problem defined by $\mathbb{B}_t$. The approximation problem is an instance of quadratic programming similar to the one in Equation 4, except that the raw cutting planes are adjusted to make sure that the approximation is a local underestimator of the objective function. The minimization of the approximation problem yields a new point $\tilde{\mathbf{w}}_t$.
- ***Perform a line search.*** The algorithm performs a special line search from the best current solution $\mathbf{w}_t^*$ to the minimum of the approximation problem $\tilde{\mathbf{w}}_t$.[3] The line search outputs a

[3] Under some semi-smoothness assumptions it is proved that this line search algorithm terminates in a finite number of iterations (Luksan and Vlcek, 2000).

new solution $\mathbf{w}_{t+1}$. Two cases may arise. In a first case, this new solution does not lead to a significant improvement (i.e. decrease) in the objective function, we say the current iteration is a null step. In such a case, the best solution does not change (i.e. $\mathbf{w}^*_{t+1} \equiv \mathbf{w}^*_t$). Alternatively the new solution may bring a significant improvement in the objective (iteration is called a serious step). Then one defines the new best solution as $\mathbf{w}^*_{t+1} \equiv \mathbf{w}_{t+1}$. Note that in both cases, the approximation function is improved by adding a new cutting plane at $\mathbf{w}_{t+1}$. We do not present in details the line search procedure since it is both rather complicated and standard. Interested readers may find detailed description in the literature, e.g., (Luksan and Vlcek, 2000).

- ***Update the bundle and build a new approximation function.*** The set of cutting planes is expanded with the new cutting plane built at $\mathbf{w}_{t+1}$. Due to the non-convex feature of the objective function, the definition of approximation is not trivial, involving additional concepts such as locality measure, the strategy of NBM to deal with non-convexity will be detailed in the next subsection. Importantly, note that one gets more cutting planes in the bundle as the algorithm iterates, and such a ever increasing number of cutting planes may represent a potential problem wrt. computational and memory cost if many iterations are required. Usually to overcome such a problem, one uses an aggregated cutting plane in order to accumulate information of all cutting planes in previous iterations (Kiwiel, 1985). It allows discarding older cutting planes and helps limiting the algorithmic complexity. For instance, one may keep a fixed number of cutting planes in the bundle $\mathbb{B}_t$ by removing the oldest cutting plane. Then, the aggregated cutting plane allows preserving part of the information brought by removed cutting planes.

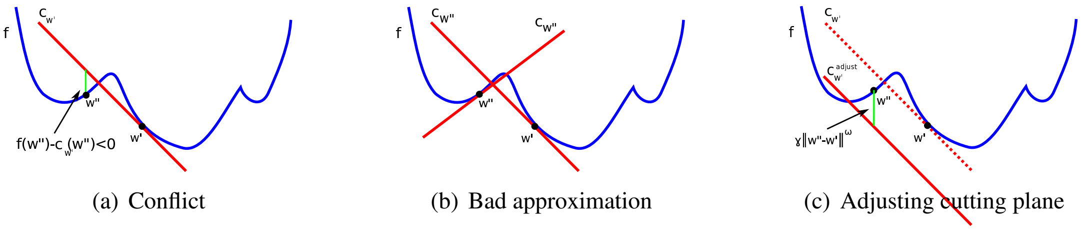
2.4.2 HANDLING NON-CONVEX OBJECTIVE FUNCTION

Bundle methods must be adapted to work for non-convex optimization since the core idea of using a first order Taylor approximation as an underestimator of the objective function does not hold anymore. Then, the standard approximation function, which is defined as the maximum over a set of cutting plane approximations, is not an underestimator of the non-convex objective function anymore. In addition although one may reasonably assume that a cutting plane built at a point $\mathbf{w}'$ is an accurate approximation of f in a small region around $\mathbf{w}'$, such an approximation may become very poor for $\mathbf{w}$ far from $\mathbf{w}'$. At the end, the maximum over cutting plane "approximations" may be a very poor approximation of the objective.

An example of poor approximation is shown in Figure 3(a). The linearization error $(f(\mathbf{w}) - c_{\mathbf{w}'}(\mathbf{w}))$ of a cutting plane $c_{\mathbf{w}'}$ at a point $\mathbf{w}''$ may be negative, meaning that the function is overestimated at that point. In the following we will say in such a case that there is a **conflict** between cutting plane $c_{\mathbf{w}'}$ and $\mathbf{w}''$. As can be seen, overestimation of a cutting plane at a local minimum will probably "remove" this minimum from the set of reachable solutions. Figure 3(b) shows that all three visible local minimums are "removed" by overestimation of the two cutting planes built at $\mathbf{w}'$ and $\mathbf{w}''$.

Non-convex bundle method strategy. In non-convex bundle methods (Kiwiel, 1985; Gaudioso and Monaco, 1992; Makela, 2002; Makela and Neittaanmaki, 1992; Schramm and Zowe, 1992) the solution to overcome conflicts between a cutting plane $c_{\mathbf{w}'}$ and a point $\mathbf{w}''$ is to lower the cutting plane $c_{\mathbf{w}'}$ by changing its offset while preserving the normal vector $\mathbf{a}_{\mathbf{w}'}$ (see Figure 3(c)). This leads to an adjusted cutting plane:

$$c^{adjust}_{\mathbf{w}'}(\mathbf{w}) = \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w} \rangle + b^{adjust}_{\mathbf{w}'}.$$



(a) Conflict (b) Bad approximation (c) Adjusting cutting plane

Figure 3: Cutting planes and linearization errors.

The offset $b_{\mathbf{w}'}$ is changed in $b_{\mathbf{w}'}^{adjust}$ so that that the linearization error of $c_{\mathbf{w}'}^{adjust}$ at $\mathbf{w}''$ is greater than or equal to both, the absolute value of the linearization error between $c_{\mathbf{w}'}$ and f at $\mathbf{w}''$, and a *locality measure* between $\mathbf{w}'$ and $\mathbf{w}''$:

$$f(\mathbf{w}'') - c_{\mathbf{w}'}^{adjust}(\mathbf{w}'') \geq |f(\mathbf{w}'') - c_{\mathbf{w}'}(\mathbf{w}'')|, \tag{9}$$

$$f(\mathbf{w}'') - c_{\mathbf{w}'}^{adjust}(\mathbf{w}'') \geq \gamma\|\mathbf{w}'' - \mathbf{w}'\|^{\omega} \tag{10}$$

where $\gamma \geq 0, \omega \geq 1$ are *locality measure parameters*. The condition (9) ensures that if the linearization error, $f(\mathbf{w}'') - c_{\mathbf{w}'}(\mathbf{w}'')$, is negative then the cutting plane has to be lowered at least twice the amount that is required to have linearization error zero. In other words, in the case of negative linearization error at $\mathbf{w}''$, the cutting plane is adjusted so that the new linearization error is positive, with at least the same magnitude as the “old” negative linearization error. The condition (10) defines another underestimator on the linearization error (of the adjusted cutting plane) which is based on the distance between two points $\mathbf{w}'$ and $\mathbf{w}''$. The further the two points are the greater the linearization error should be. The two conditions lead to the following offset change definition:

$$b_{\mathbf{w}'}^{adjust} = f(\mathbf{w}'') - \langle \mathbf{a}_{\mathbf{w}'}, \mathbf{w}'' \rangle - \max\left[|f(\mathbf{w}'') - c_{\mathbf{w}'}(\mathbf{w}'')|, \gamma\|\mathbf{w}'' - \mathbf{w}'\|^{\omega}\right].$$

This is the greatest offset (closest to $b_{\mathbf{w}'}$) that satisfies the two above conditions. Besides, one can easily check that if $c_{\mathbf{w}'}$ already satisfies both conditions (9) and (10) then $b_{\mathbf{w}'}^{adjust} = b_{\mathbf{w}'}$ and $c_{\mathbf{w}'}^{adjust}(\mathbf{w})$ and $c_{\mathbf{w}'}(\mathbf{w})$ coincide.

2.5 Conclusion

CRBM are a fast adaptation of bundle methods to convex and regularized risks. Every iteration a new cutting plane is added to the bundle so that the size of the bundle at iteration t is t. This makes tackling complex tasks, eventually requiring many iterations, difficult since the cost of solving the minimization of the approximated function is quadratic in the size of the bundle. To make CRBM more scalable we will provide a limited memory variant where the size of the bundle is limited to a given size (theoretically three CP are sufficient) whatever the iteration.

General non-convex bundle methods have been proved to have global convergence to cluster points which are stationary solutions. Note that a stationary solution is not necessarily a local minimum but may be a saddle point or even a local maximum. In practice, however, there are many hyper-parameters to tune (γ, ω, regularization term, and several hyper-parameters for the line search procedure) and convergence rate is not guaranteed, both drawbacks preventing using such algorithms for large scale applications. We will propose a variant of regularized bundle method that is adapted to non-convex risks and which is scalable in practice.

3. Non-Convex Regularized Bundle Method (NRBM)

The success of convex regularized bundle methods with improved convergence rate over bundle methods, both in theory and practice, motivated us to investigate their extension to non-convex optimization, leading to bundle methods for regularized non-convex risks (NRBM). To design such an algorithm, we propose two main contributions, the extension of CRBM to non-convex risks and a limited memory variant of bundle methods that allows limiting the algorithmic cost of a single iteration.

The extension of CRBM for non-convex function is not straightforward since, as we already observed when presenting NBM, the cutting plane approximation does not yield an underestimator of the objective function. Our proposal is to exploit some techniques of NBM for handling non-convex function while considering a special design of the algorithm in order to keep the fast convergence rate of CRBM. On one hand, we use standard techniques such as the introduction of locality measure and the adjustment of cutting planes in order to build local underestimator of the function at a given point. On the other hand, we propose novel techniques such as a particular definition of the locality measure for regularized risk and the introduction of constraints on CPs adjustment when dealing with conflicts, which guarantee a minimal improvement on the approximation gap within an iteration. At the end, we come up with a non-convex variant which inherits, in practice, the convergence rate of CRBM. Note however that we may only provide weak theoretical results on the convergence to a local minimum for the non-convex case. Convergence analysis is discussed in Section 4.

The ability of our method, NRBM, to deal with non-convex risk allows tackling a wide range of application and especially a number of everyday machine learning problems. Yet the algorithmic cost of a single iteration grows with the number of the iteration. Actually, the dual program of the approximation problem minimization in Equation 8 has a memory cost of $O(tD+t^2)$ for storing all the cutting planes and the dot product matrix between cutting planes' normal vectors (i.e. $\langle a_i, a_j \rangle$), where t is the number of cutting planes (it is equal to the iteration number in CRBM) and D is the dimensionality of $\mathbf{w}$. In addition, the computational cost for solving the dual program is usually quadratic or cubic in t. These costs may be prohibitive especially in situations where the objective is hard to optimize and the algorithm requires a large number of iterations to converge (e.g. weak regularization), where t may become very large. For instance, in experiments of training a linear SVM for adult data set (Teo et al., 2007), CRBM requires thousands of iterations for small values of λ. To overcome such an issue and to make our NRBM practical for large scale and difficult optimization problems we propose a limited memory mechanism. It is based on the use of a cutting plane aggregation method which allows drastically limiting the number of CPs in the working set at the price of a less accurate underestimator approximation. Note that such a limited memory variant may be used with convex and non-convex risks. Also, this limited memory variant applied to convex risks may be shown to inherit the convergence rate (w.r.t. the number of iterations) of CRBM, while the cost of every iteration does not depend on the iteration number anymore.

To ease the presentation, we will present in Section 3.1 the limited memory variant of bundle methods for the special case of convex risks. Then, we will consider in Section 3.2 our non-convex extension of CRBM for dealing with non-convex risks, named Non-convex Regularized Bundle Method, with includes as a particular feature the limited memory strategy.

Algorithm 3 Limited memory CRBM

```
Input: $\mathbf{w}_1$, $R$, $\lambda$, $\varepsilon$, $M$
Output: $\mathbf{w}^*$
Compute $\mathbf{a}_{\mathbf{w}_1}$ and $b_{\mathbf{w}_1}$ of $R$ at $\mathbf{w}_1$
$\tilde{\mathbf{w}}_1 = -\mathbf{a}_1/\lambda$, $\tilde{\mathbf{a}}_1 = \mathbf{a}_{\mathbf{w}_1}$; $\tilde{b}_1 = b_{\mathbf{w}_1}$; $J_1 = \{1\}$
for $t = 2$ to $\infty$ do
    Compute new CP ($\mathbf{a}_{\mathbf{w}_t}$,$b_{\mathbf{w}_t}$) of $R$ at $\mathbf{w}_t$
    $\mathbf{w}_t^* = \text{argmin}_{\mathbf{w} \in \{\mathbf{w}_1, \ldots \mathbf{w}_t\}} f(\mathbf{w})$
    $J_t \leftarrow \text{UpdateWorkingSet}(J_{t-1}, t, M)$
    $[\tilde{\mathbf{w}}_t, \tilde{c}_t] \leftarrow$ Minimize $g_t(\mathbf{w})$ in Equation 11
    $gap_t = f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t)$
    if $gap_t < \varepsilon$ then return $\mathbf{w}_t^*$
end for
```

3.1 Limited Memory for Convex Case

Our goal here is to limit the number of cutting planes used in the approximation function, which can be done by removing some of the previous cutting planes if the number of cutting planes reaches a given limit. However, the approximation gap is no more guaranteed to decrease after each iteration if one removes some of the CPs without care. The subgradient aggregation technique (Kiwiel, 1983) appears then to be an appealing solution since it can be used to accumulate information from multiple subgradients. Our proposal is to apply a similar technique to the set of cutting planes approximation of the risk function R, yielding an aggregated cutting plane.[4] Interestingly, we can show that if such an aggregated cutting plane is included in the approximation function, then one can remove any (or even all) previous cutting plane(s) while preserving the theoretical convergence rate $O(1/\lambda\varepsilon)$ iterations of CRBM.

Recall that the approximation function at iteration t is :

$$g_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max\left(\left(\max_{j \in J_t} c_j(\mathbf{w})\right), \tilde{c}_{t-1}(\mathbf{w})\right) \qquad (11)$$

where $J_t \subset \{1,..,t\}$ stands for a working set of active cutting plane indexes that we keep at iteration t and $\tilde{c}_{t-1}(\mathbf{w}) = \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w} \rangle + \tilde{b}_{t-1}$ is the aggregated cutting plane which accumulates information from previous cutting planes, $c_1, \ldots, c_{t-1}$.

The limited memory CRBM is described in Algorithm 3. It takes as input an initial solution $\mathbf{w}_1$, the convex risk function R, the regularization parameter λ, the tolerance ε, and the maximum number of active CPs $M \geq 1$. It produces as output a solution of the optimization problem, $\mathbf{w}^*$. The principle of the algorithm is similar to CRBM except that one has to decide how to define J_t via the function UpdateWorkingSet(J_{t-1}, t, M) and how to define the aggregated cutting plane.

UpdateWorkingSet. At iteration t, a new cutting plane is added to the current set of cutting planes J_{t-1}, but if J_{t-1} is full (i.e., $|J_{t-1}| = M$) then we need to select a cutting plane in J_{t-1} to remove. A simple strategy is to replace the oldest cutting plane in J_{t-1} by the new one: $J_t = J_{t-1} \cup \{t\} \setminus \{t - M - 1\}$. Alternately, one may rely on a more sophisticated way for selecting which cutting plane to

[4] We prefer this terminology to standard *aggregate subgradients* to stress that some cutting planes might be fully artificial and would not correspond to real subgradient of the risk in the non-convex case.

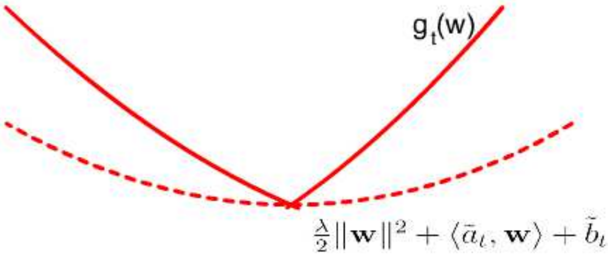

$g_t(w)$
$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \langle \tilde{a}_t, \mathbf{w} \rangle + \tilde{b}_t$

Figure 4: Quadratic underestimator of $g_t(\mathbf{w})$ (solid line) and corresponding aggregated cutting plane $\tilde{c}_t(\mathbf{w})$ (dash line)).

remove from J_{t-1}. In our implementation, we maintain a count for each CP which is the number of iterations in which the CP does not contribute to the aggregation CP (see below for details about the definition of the aggregation CP). Then the CP with highest count is selected to be removed.

Cutting plane aggregation. The use of an aggregated cutting plane is a key issue to limit storage requirements and computational effort per iteration. The technique is inspired by the subgradient aggregation idea of Kiwiel (1983), which can be viewed as building a low cost approximation of the piece-wise quadratic function in Equation 4. Basically, by considering a linear combination of subgradient of f computed in previous iterations, we can discard previous subgradients without losing all information. In our method, we also use aggregation technique for building a low cost approximation of the approximation function $g_t(\mathbf{w})$. Note that we use a slightly different terminology (CP aggregation instead of subgradient aggregation) since our goal is to build an approximation of f using cutting planes, rather than building an approximation of subdifferential as in standard bundle methods which aims at finding a solution with small sub-gradient. There are two key differences between our CP aggregation technique and the subgradient aggregation proposed originally by Kiwiel (1983). First, our method is specifically designed for quadratically regularized objective which makes possible to show that our limited memory variant using CP aggregation inherits the theoretical convergence rate of CRBM (as least for convex risks). Instead the standard subgradient aggregation technique can be applied to any objective function by using an additional regularization term in the search direction optimization problem. Second, while the original method focuses on aggregating subgradients, our algorithm applies the aggregation idea to both the direction, $\tilde{a}$, and to the offset, $\tilde{b}$ (and also to the locality measure in the non convex case, see later in Section 3.2.4).

At iteration t of Algorithm 3, the cutting plane aggregation $\tilde{c}_t(\mathbf{w})$ is derived from the minimization of $g_t(\mathbf{w})$. We use the cutting plane technique to build an underestimator of $g_t(\mathbf{w})$ at its minimum $\tilde{\mathbf{w}}_t = \operatorname{argmin}_{\mathbf{w}} g_t(\mathbf{w})$. Although any linear combination of previous cutting planes could yield an under estimator of $g_t(\mathbf{w})$, only one of them, that we note $\tilde{c}_t(\mathbf{w})$ hereafter, corresponds to a tight quadratic approximation $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ that reaches the same minimum as $g_t(\mathbf{w})$:

$$\tilde{\mathbf{w}}_t = \underset{\mathbf{w}}{\operatorname{argmin}}\, g_t(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w}).$$

The particular property of $\tilde{c}_t(\mathbf{w})$ is important since it allows to guarantee that for the limited memory version of the algorithm, the gap between the best observed objective value and the minimum of the approximated function is unchanged even if one discards all previous cutting planes.

Figure 4 illustrates the quadratic function (in red dash line) derived from the aggregated cutting plane at iteration $t = 2$. The cutting plane $\tilde{c}_t(\mathbf{w})$ can be defined based on the dual solution of the approximation problem which may be characterized in primal and dual forms as follows:

$$\begin{array}{llll} & \textit{Primal} & & \textit{Dual} \\ \min_{\mathbf{w}} & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi & \max_{\boldsymbol{\alpha}} & -\frac{1}{2\lambda}\|\boldsymbol{\alpha} A_t\|^2 + \boldsymbol{\alpha} B_t \\ s.t & \langle a_j, \mathbf{w}\rangle + b_j \leq \xi \; \forall j \in J_t & s.t & \alpha_j \geq 0 \; \forall j \in J_t ; \tilde{\alpha} \geq 0 \\ & \langle \tilde{a}_{t-1}, \mathbf{w}\rangle + \tilde{b}_{t-1} \leq \xi & & (\sum_{j \in J_t} \alpha_j) + \tilde{\alpha} = 1 \end{array}$$

where $A_t = [...;\mathbf{a}_j;...,\tilde{\mathbf{a}}_{t-1}]$ is a matrix (with $\mathbf{a}_j$ and $\tilde{\mathbf{a}}_{t-1}$ being row vectors), $B_t = [...;b_j;...;\tilde{b}_{t-1}]$ is the vector of scalars and $\boldsymbol{\alpha}$ stands for the (row) vector of Lagrange multipliers (of length $|J_t| + 1$ at iteration t). We denote α_j as the Lagrange multiplier associated with the CP c_j and we denote $\tilde{\alpha}$ as the Lagrange multiplier associated with the aggregated CP $\tilde{c}_{j-1}$. Let $\boldsymbol{\alpha}_t$ be the solution of the above dual program then the minimizer of the primal can be expressed as:

$$\tilde{\mathbf{w}}_t = -\frac{\boldsymbol{\alpha}_t A_t}{\lambda} = -\frac{\sum_{j \in J_t} \alpha_j \mathbf{a}_j + \tilde{\alpha}\tilde{\mathbf{a}}_{t-1}}{\lambda}.$$

The following proposition show how to use $\boldsymbol{\alpha}_t$ for defining a tight underestimator of $g_t(\mathbf{w})$.

Proposition 1 *Let $\tilde{c}_t(\mathbf{w}) = \langle \tilde{\mathbf{a}}_t, \mathbf{w}\rangle + \tilde{b}_t$ be the aggregated CP defined by:*

$$\begin{array}{ll} \tilde{\mathbf{a}}_t & = \boldsymbol{\alpha}_t A_t = \sum_{j \in J_t} \alpha_j \mathbf{a}_j + \tilde{\alpha}\tilde{\mathbf{a}}_{t-1}, \\ \tilde{b}_t & = \boldsymbol{\alpha}_t B_t = \sum_{j \in J_t} \alpha_j b_j + \tilde{\alpha}\tilde{b}_{t-1} \end{array}$$

then the quadratic function $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is an underestimator of $g_t(\mathbf{w})$, which reaches the same minimum value as $g_t(\mathbf{w})$ at the same point, $\tilde{\mathbf{w}}_t$.

Proof First, by construction we have $\tilde{\mathbf{w}}_t = -\frac{\tilde{\mathbf{a}}_t}{\lambda}$ which implies that the derivative of $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is null at $\tilde{\mathbf{w}}_t$. Second, we can show that $\frac{\lambda}{2}\|\tilde{\mathbf{w}}_t\|^2 + \tilde{c}_t(\tilde{\mathbf{w}}_t) = g_t(\tilde{\mathbf{w}}_t)$. Actually:

$$\begin{array}{llll} g_t(\tilde{\mathbf{w}}_t) & = -\frac{1}{2\lambda}\|\boldsymbol{\alpha}_t A_t\|^2 + \boldsymbol{\alpha}_t B_t & = -\frac{\lambda}{2}\|\frac{\tilde{\mathbf{a}}_t}{\lambda}\|^2 + \tilde{b}_t \\ & = \frac{\lambda}{2}\|\frac{\tilde{\mathbf{a}}_t}{\lambda}\|^2 - \lambda\|\frac{\tilde{\mathbf{a}}_t}{\lambda}\|^2 + \tilde{b}_t & = \frac{\lambda}{2}\|\tilde{\mathbf{w}}_t\|^2 - \langle \tilde{\mathbf{a}}_t, \frac{\tilde{\mathbf{a}}_t}{\lambda}\rangle + \tilde{b}_t \\ & = \frac{\lambda}{2}\|\tilde{\mathbf{w}}_t\|^2 + \langle \tilde{\mathbf{a}}_t, \tilde{\mathbf{w}}_t\rangle + \tilde{b}_t. & \end{array} \tag{12}$$

In other words, the quadratic function $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ and the approximation function $g_t(\mathbf{w})$ reach the same minimum value $g_t(\tilde{\mathbf{w}})$ at the same point $\tilde{\mathbf{w}}_t$.

Finally, we show that $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is an underestimator of $g_t(\mathbf{w})$. Let

$$h_t(\mathbf{w}) = \max\left[\max_{j = \in J_t} \langle \mathbf{a}_j, \mathbf{w}\rangle + b_j, \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}\rangle + \tilde{b}_{t-1}\right]$$

be the piecewise linear approximation of $R(\mathbf{w})$ at iteration t, we have:

$$0 \in \partial g_t(\tilde{\mathbf{w}}_t) \equiv \lambda\tilde{\mathbf{w}}_t + \partial h_t(\tilde{\mathbf{w}}_t)$$

since $\tilde{\mathbf{w}}_t$ is the optimum solution of minimizing $g_t(\mathbf{w})$. Note that $\tilde{\mathbf{a}}_t = -\lambda\tilde{\mathbf{w}}_t$, the above equation implies that $\tilde{\mathbf{a}}_t \in \partial h_t(\tilde{\mathbf{w}}_t)$. In other words, $\tilde{\mathbf{a}}_t$ is a subgradient of $h_t(\mathbf{w})$ at $\tilde{\mathbf{w}}_t$. Furthermore, since $g_t(\tilde{\mathbf{w}}_t) = \frac{\lambda}{2}\|\tilde{\mathbf{w}}_t\|^2 + h_t(\tilde{\mathbf{w}}_t)$, Equation 12 gives:

$$\langle \tilde{\mathbf{a}}_t, \tilde{\mathbf{w}}_t\rangle + \tilde{b}_t = h_t(\tilde{\mathbf{w}}_t).$$

The cutting plane $\tilde{c}_t(\mathbf{w})$ is then an underestimator of $h_t(\mathbf{w})$ built at $\tilde{\mathbf{w}}_t$ (recall that $h_t(\mathbf{w})$ is convex), and thus $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is a quadratic underestimator of $g_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + h_t(\mathbf{w})$. Note that since $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is an underestimator of $g_t(\mathbf{w})$ and $g_t(\mathbf{w})$ is an underestimator of $f(\mathbf{w})$ at $\mathbf{w}_t^*$, the quadratic function $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \tilde{c}_t(\mathbf{w})$ is also an underestimator of $f(\mathbf{w})$ at $\mathbf{w}_t^*$.

■

3.2 Regularized Bundle Method for Non-Convex Risks

To handle non-convex objective function, we introduce some new notations in addition to the notation used in Algorithm 3. In the following, we recall useful notations from previous section, and we introduce additional notations that will be useful hereafter.

Notations from limited memory CRBM. At iteration t, $\mathbf{w}_t$ is the current solution and $\mathbf{w}_t^*$ is the best observed solution. J_t corresponds to the working set of cutting plane, which is involved in the definition of the approximation $g_t(\mathbf{w})$. $\tilde{\mathbf{w}}_t$ is the solution of the minimization of $g_t(\mathbf{w})$, it is also considered as the solution in the next iteration.

Raw and modified cutting planes. We have to distinguish between a raw linear cutting plane of the risk $c_{\mathbf{w}_j}$ (with $c_{\mathbf{w}_j}(\mathbf{w}) = \langle \mathbf{a}_{\mathbf{w}_j}, \mathbf{w}\rangle + b_{\mathbf{w}_j}$) that is built at a particular iteration j of the algorithm and the eventually modified versions of this cutting plane that might be used in posterior iterations. Indeed a cutting plane may be modified multiple times for solving conflicts as in standard NBM method. At iteration t we note c_j^t (with $c_j^t(\mathbf{w}) = \langle \mathbf{a}_j, \mathbf{w}\rangle + b_j^t$) the cutting plane which is derived from $c_{\mathbf{w}_j}$, the raw CP originally built at iteration j. Unlike NBM, the normal vector $\mathbf{a}_j$ in our algorithm might be different than the subgradient $\mathbf{a}_{\mathbf{w}_j}$ computed at $\mathbf{w}_j$, due to our particular solving conflict method. However, once defined at iteration j, the normal vector $\mathbf{a}_j$ remains fixed over iterations. On the contrary, the offset might be modified multiple times for solving conflicts occurring after iteration j, and we use a superscript t indicating the iteration number for the cutting plane's offset b_j^t.

Bundle. The bundle $\mathbb{B}_t$ denotes the state of the algorithm at iteration t. It consists in a set of cutting planes which were built at previous solutions, c_j^t for $j \in J_t$. Similarly to non-convex bundle methods, we define a locality measure which is associated to any active cutting plane. It is related to the locality measure between the cutting plane (actually the point where the cutting plane was built) and the best current observed solution. We note s_j^t the locality measure between cutting plane c_j^t and the best observed solution up to iteration t, $\mathbf{w}_t^*$. The full bundle information is:

$$\mathbb{B}_t = \{c_j^t, s_j^t\}_{j \in J_t} \cup \{\tilde{c}_{t-1}^t, \tilde{s}_{t-1}^t\}$$

where $\tilde{c}_{t-1}^t$ is an aggregated cutting plane and $\tilde{s}_{t-1}^t$ is its locality measure to the best observed solution $\mathbf{w}_t^*$. Similar to the aggregation technique presented in Section 3.1, the aggregated CP $\tilde{c}_{t-1}^t$ can be viewed as a convex combination of CPs in previous iterations. For non-convex objective function, each CP in the bundle is associated with a locality measure, including the aggregated CPs whose locality measure is a convex combinations of locality measures of other CPs.

3.2.1 SKETCH OF ALGORITHM

The main algorithm is described in Algorithm 4, for which the input is similar to the case of Algorithm 3 except the fact that the risk R can be non-convex. To deal with non-convexity, the key idea to use CPs in the bundle to build a local underestimator of f around the best observed solution.

Algorithm 4 NRBM

```
Input: $\mathbf{w}_1$,$R$, $\lambda$, $\varepsilon$,$M$
Output: $\mathbf{w}^*$
Initialization:
  Compute cutting plane $c_{\mathbf{w}_1}$ of $R$
  $[c_1^1, s_1^1] = [\tilde{c}_1^1, \tilde{s}_1^1] = [c_{\mathbf{w}_1}, 0]$
  $\tilde{\mathbf{w}}_1 = -\mathbf{a}_1/\lambda$
  $\mathbb{B}_1 = \{c_1^1, s_1^1, \tilde{c}_1^1, \tilde{s}_1^1\}$
for $t = 2$ to $\infty$ do
  $\mathbf{w}_t \leftarrow \tilde{\mathbf{w}}_{t-1}$
  Compute cutting plane $c_{\mathbf{w}_t}$ of $R$
  $\mathbf{w}_t^* = \text{argmin}_{\mathbf{w} \in \{\mathbf{w}_1, \ldots \mathbf{w}_t\}} f(\mathbf{w})$
  $\mathbb{B}_t$ = UpdateBundle($\mathbb{B}_{t-1}, \mathbf{w}_{t-1}^*, \mathbf{w}_t^*, c_{\mathbf{w}_t}, \mathbf{w}_t$,M)
  $(\tilde{\mathbf{w}}_t, \tilde{c}_t^t, \tilde{s}_t^t)$ = MinimizeApproximationProblem($\mathbb{B}_t$,$\lambda$)
  $gap_t = f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t)$
  if $gap_t < \varepsilon$ then return $\mathbf{w}_t^*$
end for
```

Similar to CRBM and limited memory CRBM, the approximation problem is designed in such a way that one can use the minimum of the approximation problem as the new current solution. In other words, NRBM does not require a dedicated line search procedure to ensure convergence as in the standard NBM (Kiwiel, 1985). Such a line search is not required for convergence matters in our method but it may be still used for improving convergence rate in practice (see Section 3.3.2).

Initialization

Initialization consists in providing a first bundle $\mathbb{B}_1$. Starting with an initial solution $\mathbf{w}_1$, we build the first cutting plane $c_1^1 = c_{\mathbf{w}_1} = \langle \mathbf{a}_{\mathbf{w}_1}, \mathbf{w} \rangle + b_{\mathbf{w}_1}$. Note that at iteration $t = 1$, there is only one cutting plane c_1^1 and the aggregated cutting plane is also c_1^1: $[\tilde{c}_1^1, \tilde{s}_1^1] = [c_1^1, s_1^1]$. The *approximation function* is then:

$$g_1(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \langle \mathbf{a}_1, \mathbf{w} \rangle + b_1^1$$

which reaches its minimum at $\tilde{\mathbf{w}}_1 = -\mathbf{a}_1/\lambda$. The state of algorithm $\mathbb{B}_1$ is set to c_1^1 and $\tilde{c}_1^1$ (which coincide) with their corresponding locality mesures to the best solution $\mathbf{w}_1$ ($\tilde{s}_1^1 = s_1^1 = 0$).

Iteration **t**

Every iteration the algorithm determine a new bundle $\mathbb{B}_t$, the best observed solution up to iteration t, $\mathbf{w}_t^*$, and the new current (and temporary) solution $\mathbf{w}_t$. At iteration $t > 1$, few steps are successively performed:

- Build a new cutting plane at $\tilde{\mathbf{w}}_{t-1}$ the minimizer of approximation function in previous iteration ($g_{t-1}(\mathbf{w})$).
- Update the best observed solution $\mathbf{w}_t^*$.
- Solve any conflict between the best observed solution,$\mathbf{w}_t^*$, and all cutting planes in the bundle. This is done through a call to UpdateBundle function which we detail later. This yields a piece-wise quadratic function g_t which is a local underestimator approximation of f. As said before, in addition to cutting planes built at previous solutions (e.g. at $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$), we use a

special aggregated cutting plane, $\tilde{c}^t_{t-1}$ for gathering information of previous cutting planes up to iteration $t-1$. The approximation function at iteration t is then:

$$g_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max\left[\max_{j\in J_t} c^t_j(\mathbf{w}), \tilde{c}^t_{t-1}(\mathbf{w})\right] \tag{13}$$

where, as in Section 3.1, J_t stands for a subset of cutting planes defined in previous iterations if one wishes to use a limited memory variant.

- Minimize g_t. This gives a solution named $\tilde{\mathbf{w}}_t$ which will be used in next iteration. Note that a side effect of this minimization is the definition of a new aggregated cutting plane and its locality measure to the best observed solutions.

This procedure is repeated until the gap (i.e. the difference between the best observed value of objective function and the minimum of the approximation function) is less than a desired accuracy ε. We say that an ε-solution has been reached.

We detail in the following sections how the approximation is built and procedure for solving conflict in the update of the bundle. Then we provide details on our definition of the aggregated cutting plane.

3.2.2 LOCALITY MEASURE AND CONDITIONS ON CPS

Given a set of cutting plane approximation of R, one could build a local underestimator of f in the vicinity of $\mathbf{w}$ by descending CPs that yields non positive linearization error of f at $\mathbf{w}$. Our algorithm focus on solving conflicts between CPs in the bundle and the best observed solution $\mathbf{w}^*_t$. While sharing some concepts with NBM such as locality measure, null step and descent step our method is based on a new greedy strategy for solving conflicts which guarantee a minimum improvement of the approximation gap after each iteration which is similar to CRBM.[5]

Locality measure definition. We propose to define the locality measure between a cutting plane previously built at iteration j and the current best solution $\mathbf{w}^*_t$ based on the trajectory from $\mathbf{w}_j$ to $\mathbf{w}^*_t$. We exploit the same shape of our regularization term (L2 norm) to define our locality measure.[6] At iteration t, we define the locality measure between CP c^t_j built at $\mathbf{w}_j$ and $\mathbf{w}^*_t$ as:

$$s^t_j = s(\mathbf{w}_j, \mathbf{w}^*_t) = \frac{\lambda}{2}\left(\|\mathbf{w}_j - \mathbf{w}^*_j\|^2 + \sum_{k=j+1}^{t}\|\mathbf{w}^*_k - \mathbf{w}^*_{k-1}\|^2\right)$$

which yields a natural recursive formulate:

$$s^t_j = s^{t-1}_j + \frac{\lambda}{2}\|\mathbf{w}^*_t - \mathbf{w}^*_{t-1}\|^2, \forall j < t.$$

Lower bound and upper bound on offset adjustment. As in NBM, raw CP cannot always be used to build an underestimator of $f(\mathbf{w})$, which is non-convex so that CP need adjustments. We discuss two conditions that define an upper and an underestimator on a CP's offset modification when solving a conflict with respect to $\mathbf{w}^*_t$.

[5] Note that we use the terminology descent step instead of serious steps since descent step here is not fully similar to serious step in standard non convex bundle methods.

[6] Standard bundle methods use γd^{ω} where d is the Euclidean distance and $\gamma > 0$ and ω are hyper parameters (Cf. Equation 10).

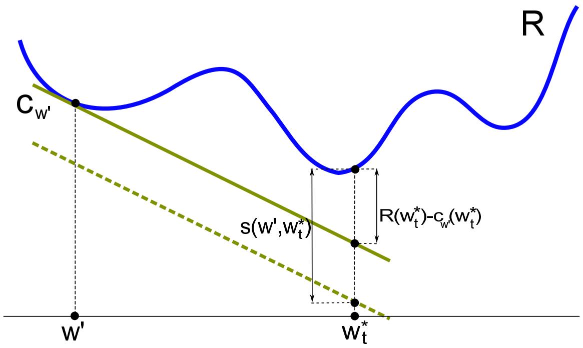


Figure 5: Conflict between $\mathbf{w}_t^*$ and a cutting plane $c_{\mathbf{w}'}$.

First, as in standard NBM (recall Equation 10), we consider the following first *condition* requiring that a CP built at $\mathbf{w}'$, $c_{\mathbf{w}'}$, gives a positive linearization error at $\mathbf{w}_t^*$, which must grow with the locality measure of the CP to $\mathbf{w}_t^*$:

$$R(\mathbf{w}_t^*) - c(\mathbf{w}_t^*) \geq s(\mathbf{w}', \mathbf{w}_t^*) \tag{14}$$

where $s(.,.)$ is our non-negative locality measure between the two points. The positive value of $s(\mathbf{w}', \mathbf{w}_t^*)$ ensures that the linear approximation $c_{\mathbf{w}'}(\mathbf{w})$ is an underestimator of $R(\mathbf{w})$ at least within a small region around $\mathbf{w}_t^*$. Figure 5 illustrates this case. The cutting plane $c_{\mathbf{w}'}$ which was built at $\mathbf{w}'$ does not satisfy condition 14. This conflict between cutting plane $c_{\mathbf{w}'}$ and $\mathbf{w}_t^*$ is solved in NBM by lowering $c_{\mathbf{w}'}$ (by tuning its offset b') so that the linearization error at $\mathbf{w}_t^*$, $R(\mathbf{w}_t^*) - c_{\mathbf{w}'}(\mathbf{w}_t^*)$, becomes at least $s(\mathbf{w}', \mathbf{w}_t^*)$. This yield an *upper bound* on the new offset b':

$$b' \leq R(\mathbf{w}_t^*) - \langle \mathbf{a}', \mathbf{w}_t^* \rangle - s(\mathbf{w}', \mathbf{w}_t^*). \tag{15}$$

Unfortunately if a cutting plane is lowered too much, the minimum of the approximation function is not guaranteed to improve every iteration anymore. For instance it may happen that the minimum of the approximated function is not changed once the new cutting plane has been lowered, yielding a infinite loop without any improvement on the solution. Standard non-convex bundle methods handle this problem with a special line search procedure (between the current best observed solution and the minimum of the approximation problem) with stopping conditions that ensure some minimal changes of the approximation problem.

We found instead that there is a simple sufficient condition that guarantees an improvement of the minimum of the approximation function every iteration (required by Lemma 4). It concerns the new added cutting plane only and writes: $\frac{\lambda}{2}\|\mathbf{w}_t\|^2 + \langle \mathbf{a}_t, \mathbf{w}_t \rangle + b_t^t \geq f(\mathbf{w}_t^*)$. In other words, we need to ensure that the approximation at $\mathbf{w}_t$ using the new added cutting plane is greater or equal to the best observed function value. Note that $\mathbf{w}_t$ is the minimizer of the approximation in the previous iteration, $g_{t-1}(\mathbf{w})$, this condition influences directly the gap between the best observed function value and the minimum of the approximation. The condition can be seen as a lower bound on the modified offset:

$$b_t^t \geq f(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \mathbf{a}_t, \mathbf{w}_t \rangle. \tag{16}$$

Algorithm 5 UpdateBundle

```
Input: $\mathbb{B}_{t-1} = \{c_j^{t-1}, s_j^{t-1}\}_{j\in J_{t-1}} \cup \{\tilde{c}_{t-1}^{t-1}, \tilde{s}_{t-1}^{t-1}\}, \mathbf{w}_{t-1}^*, \mathbf{w}_t^*, \mathbf{w}_t, c_{\mathbf{w}_t}, M$
Output: $\mathbb{B}_t = \{c_j^t, s_j^t\}_{j\in J_t} \cup \{\tilde{c}_{t-1}^t, \tilde{s}_{t-1}^t\}$
if $\mathbf{w}_t^* \neq \mathbf{w}_{t-1}^*$ then                                   Descent Step
  for $j \in J_{t-1}$
    $s_j^t = s_j^{t-1} + \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|^2$
    $b_j^t = \min[b_j^{t-1}, R(\mathbf{w}_t^*) - \langle \mathbf{a}_j, \mathbf{w}_t^* \rangle - s_j^t]$
  end
  $\tilde{s}_{t-1}^t = \tilde{s}_{t-1}^{t-1} + \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|^2$
  $\tilde{b}_{t-1}^t = \min[\tilde{b}_{t-1}^{t-1}, R(\mathbf{w}_t^*) - \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}_t^* \rangle - \tilde{s}_{t-1}^t]$
  $\tilde{c}_{t-1}^t(\mathbf{w}) := \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w} \rangle + \tilde{b}_{t-1}^t$
  $[c_t^t, s_t^t] = [c_{\mathbf{w}_t}, 0]$
else                                                                  Null Step
  for $j \in J_{t-1}$
    $c_j^t = c_j^{t-1}$ ; $s_j^t = s_j^{t-1}$ ;
  end
  $\tilde{c}_{t-1}^t = \tilde{c}_{t-1}^{t-1}$ ; $\tilde{s}_{t-1}^t = \tilde{s}_{t-1}^{t-1}$ ;
  if condition (15) is not satisfied for $c_{\mathbf{w}_t}$ then
    $[c_t^t, s_t^t]$ = SolveConflictNullStep($\mathbf{w}_t^*, \mathbf{w}_t, c_{\mathbf{w}_t}$)
  else $[c_t^t, s_t^t] = [c_{\mathbf{w}_t}, \frac{\lambda}{2}\|\mathbf{w}_t - \mathbf{w}_t^*\|^2]$
end
$J_t$ =UpdateWorkingSet($J_{t-1}, t, M$)
return $\mathbb{B}_t = \{c_j^t, s_j^t\}_{j\in J_t}\} \cup \{\tilde{c}_{t-1}^t, \tilde{s}_{t-1}^t\}$
```

3.2.3 BUNDLE UPDATE

The approximation function, g_t, is refined every iteration, Algorithm 5 describes the *UpdateBundle* process. It takes as input:

- The bundle at previous iteration
- The best observed solutions at previous iteration $\mathbf{w}_{t-1}^*$
- The best observed solutions at current iteration $\mathbf{w}_t^*$
- The current solution $\mathbf{w}_t$ and its corresponding raw cutting plane, $c_{\mathbf{w}_t}$.

The algorithm is designed so that at the end of iteration t, all $(|J_t|+1)$ cutting planes in the bundle (i.e. the $|J_t|$ "normal" cutting planes and the aggregated cutting plane) satisfy condition in Equation 15 while the new added cutting plane c_t^t also satisfies condition in Equation 16. Note that $c_{\mathbf{w}_t}$ always satisfies (16) by definition of $\mathbf{w}_t^*$, so that c_t^t also satisfies (16) in case there is no conflict ($c_t^t \equiv c_{\mathbf{w}_t}$).

As the two conditions (15) and (16) involve the best observed solution, we distinguish two cases when solving conflict. Either the current solution is the best solution up to now (hence $\mathbf{w}_t^* \neq \mathbf{w}_{t-1}^*$), in which case we call the iteration a descent step. Or the current solution is not the best solution (i.e. $\mathbf{w}_t^* \equiv \mathbf{w}_{t-1}^*$), then the iteration is said to be a null step. We detail these two cases now.

Descent Step. In the case of a descent step, condition (16) is trivially satisfied for the new added cutting plane since $c_t^t \equiv c_{\mathbf{w}_t}$. Hence solving an eventual conflict is rather simple in this case. It is done by setting:

$$b_j^t = \min[b_j^{t-1}, R(\mathbf{w}_t^*) - \langle \mathbf{a}_j, \mathbf{w}_t^* \rangle - s_j^t]$$

Algorithm 6 SolveConflictNullStep

```
1: **Input:** $\mathbf{w}_t^*, \mathbf{w}_t, c_{\mathbf{w}_t}$ with parameters $(\mathbf{a}_{\mathbf{w}_t}, b_{\mathbf{w}_t})$
2: **Output:** $c_t^t$ with parameters $(\mathbf{a}_t, b_t^t)$ and $s_t^t$
3: $s_t^t = \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_t\|^2$
4: Compute $L, U$ according to Equation 17
5: **if** $L \leq U$ **then** $[\mathbf{a}_t, b_t^t] = [\mathbf{a}_{\mathbf{w}_t}, L]$ **else**
6:    $\mathbf{a}_t = -\lambda \mathbf{w}_t^*$                                NullStep2 case
7:    $b_t^t = f(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \mathbf{a}_t, \mathbf{w}_t \rangle$
```

for all j in the working set. A similar modification may be applied to the aggregated cutting plane:

$$\tilde{b}_{t-1}^t = \min[\tilde{b}_{t-1}^{t-1}, R(\mathbf{w}_t^*) - \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}_t^* \rangle - \tilde{s}_{t-1}^t]$$

where $\tilde{s}_{t-1}^t = \tilde{s}_{t-1} + \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|^2$. At the end, the adjusted aggregated CP (in the working set of iteration t) is:

$$\tilde{c}_{t-1}^t(\mathbf{w}) = \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w} \rangle + \tilde{b}_{t-1}^t.$$

Null Step. In the case of a null step, the best observed solution did not change, so that $s_j^t = s_j^{t-1}, \forall j = 1, ..., (t-1)$ and $\tilde{s}_{t-1}^t = \tilde{s}_{t-1}^{t-1}$. Since all cutting planes in $\mathbb{B}_{t-1}$ were already adjusted to satisfy positive linearization error condition wrt. the best solution at previous iteration, a conflict (if any) may only arise between the new cutting plane $c_{\mathbf{w}_t}$ and the best observed solution $\mathbf{w}_t^*$. So that all CPs (including aggregated CP) remain unchanged (see Algorithm 5 line 13) except the new added CP which must be checked for conflict.

In the null step case, solving conflict is not as simple as in a descent step case since as we said before, for convergence proof matters, we need the new cutting plane to satisfy both conditions (15) and (16). Algorithm 6 modifies c_t^t in such a way that it guarantees that the new cutting plane c_t^t with parameters $\mathbf{a}_t$ and b_t^t satisfies conditions (15) and (16). In a first attempt it tries to solve the conflict by tuning b_t^t alone while fixing $\mathbf{a}_t = \mathbf{a}_{\mathbf{w}_t}$. Indeed conditions (15) and (16) may be rewritten as:

$$\begin{aligned} b_t^t &\leq R(\mathbf{w}_t^*) - \langle \mathbf{a}_{\mathbf{w}_t}, \mathbf{w}_t^* \rangle - s_t^t = U, \\ b_t^t &\geq f(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \mathbf{a}_{\mathbf{w}_t}, \mathbf{w}_t \rangle = L \end{aligned} \tag{17}$$

which define an upper bound U and a lower bound L for b_t^t. If $L \leq U$ any value in (L, U) works (in our implementation we set $b_t^t = L$).

However it may happen that $L > U$, then tuning b_t^t is not enough (this is what we call a NullStep2 case in Algorithm 6). Both b_t^t and the normal vector $\mathbf{a}_t$ need to be adjusted to make sure that the conflict is solved (see Line 6 in Algorithm 6).

Figure 6(top-left) illustrates an example of NullStep2 where the gradient information given at $\mathbf{w}_t$ is not helpful for building a local underestimator approximation at $\mathbf{w}_t^*$. The quadratic approximation corresponding to cutting plane $c_{\mathbf{w}_t}$ is plotted in orange, which is not a local underestimator of $f(\mathbf{w})$ at $\mathbf{w}_t^*$. The conflict is so severe that it cannot be solved by just lowering the cutting plane. It should be lowered too much with respect to condition in Equation 15 (Figure 6 (top-right)), meaning that the approximation function would be unchanged and the algorithm would loop without finding a good solution.

In a NullStep2 case, we propose to ignore the gradient information at $\mathbf{w}_t$ and to rather focus on the region around the best observed solution $\mathbf{w}_t^*$ by adding a particular CP (leading to a quadratic

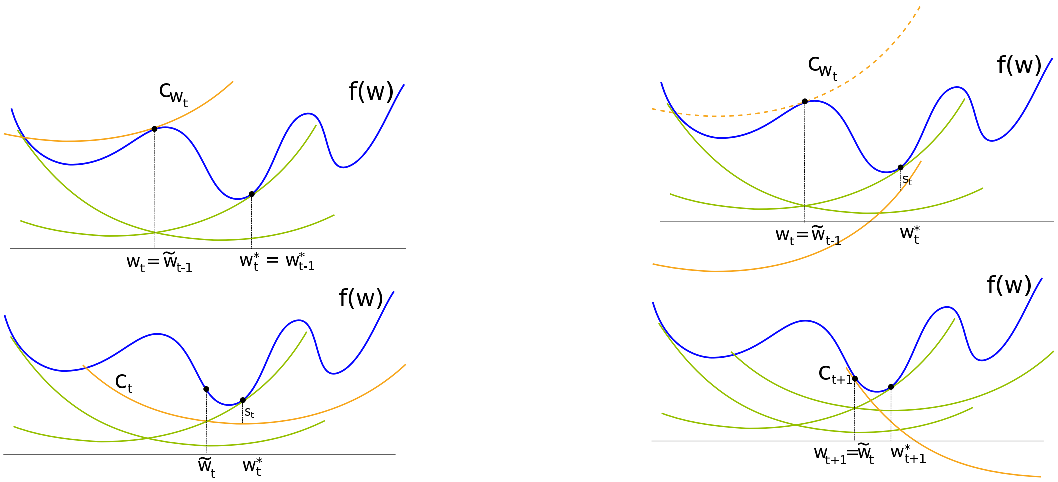


Figure 6: Illustration of NullStep2. Top-left: conflict arise at iteration t. Top-right: can not solve conflict by descend the cutting plane. Bottom-left: Nullstep2, modifying the cutting plane to solve the conflict at iteration t. Bottom-right: There is no conflict at iteration $t+1$.

local underestimator, $\frac{\lambda}{2}\|\mathbf{w}\|^2 + \langle \mathbf{a}_t, \mathbf{w}\rangle + b_t^t$) satisfying both conditions in Equation 15 and 16). This quadratic function is defined so that it reaches its minimum at $\mathbf{w}_t^*$ and the linearization error of the cutting plane $\langle \mathbf{a}_t, \mathbf{w}\rangle + b_t^t$ at $\mathbf{w}_t^*$ is $\frac{\lambda}{2}\|\mathbf{w}_t - \mathbf{w}_t^*\|^2$ (see the orange quadratic curve in Figure 6 (bottom-left)). The new cutting plane is defined as:

$$\begin{aligned}
c_t^t(\mathbf{w}) &= \langle \mathbf{a}_t, \mathbf{w}\rangle + b_t^t, \\
\mathbf{a}_t &= -\lambda \mathbf{w}_t^*, \\
b_t^t &= f(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \mathbf{a}_t, \mathbf{w}_t\rangle, \\
s_t^t &= \frac{\lambda}{2}\|\mathbf{w}_t - \mathbf{w}_t^*\|^2.
\end{aligned}$$

This CP satisfies condition (16) by construction. It also satisfies condition (15) as we show now:

$$\begin{aligned}
\langle \mathbf{a}_t, \mathbf{w}_t^*\rangle + b_t^t &= \langle \mathbf{a}_t, \mathbf{w}_t^*\rangle + f(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \mathbf{a}_t, \mathbf{w}_t\rangle \\
&= R(\mathbf{w}_t^*) + \langle \mathbf{a}_t, \mathbf{w}_t^* - \mathbf{w}_t\rangle + \frac{\lambda}{2}(\|\mathbf{w}_t^*\|^2 - \|\mathbf{w}_t\|^2) \\
&= R(\mathbf{w}_t^*) + \langle \mathbf{a}_t + \frac{\lambda}{2}(\mathbf{w}_t^* + \mathbf{w}_t), \mathbf{w}_t^* - \mathbf{w}_t\rangle
\end{aligned}$$

where we used the definition of the objective function $f(\mathbf{w}_t^*) = \frac{\lambda}{2}\|\mathbf{w}_t^*\|^2 + R(\mathbf{w}_t^*)$. Then, substituting $-\lambda \mathbf{w}_t^*$ for $\mathbf{a}_t$ (Cf. Line 6) we obtain:

$$\begin{aligned}
& \langle \mathbf{a}_t, \mathbf{w}_t^*\rangle + b_t^t &&= R(\mathbf{w}_t^*) - \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_t\|^2 \\
\Longleftrightarrow\quad & \langle \mathbf{a}_t, \mathbf{w}_t^*\rangle + b_t^t &&= R(\mathbf{w}_t^*) - s_t^t \\
\Longleftrightarrow\quad & b_t^t &&= R(\mathbf{w}_t^*) - \langle \mathbf{a}_t, \mathbf{w}_t^*\rangle - \frac{\lambda}{2}\|\mathbf{w}_t^* - \mathbf{w}_t\|^2
\end{aligned}$$

and condition in Equation 15 is satisfied.

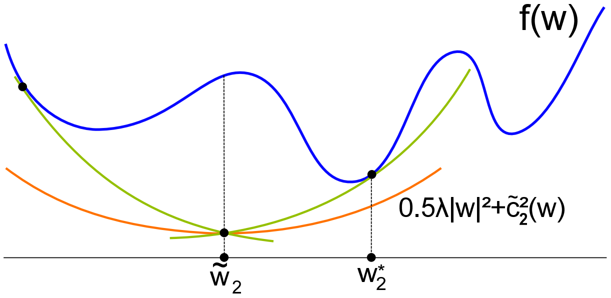


Figure 7: Quadratic underestimator of $g_t(\mathbf{w})$ derived from the aggregated cutting plane $\tilde{c}_t^t(\mathbf{w})$.

3.2.4 APPROXIMATED PROBLEM AND AGGREGATED CUTTING PLANE

In the non-convex case the aggregated CP is still an underestimator of approximation problem. Figure 7 illustrates the quadratic function (in orange) derived from the aggregated cutting plane at iteration $t = 2$.

Solving the approximated problem and definition of the aggregated cutting plane are completely similar to the case of limited memory CRBM, with the only difference that we use here at iteration t the bundle at iteration t that may include cutting planes that have been modified during previous iterations.The minimization of the approximation function ($g_t(\mathbf{w})$ in Equation 13) can be solved in the dual space as:

$$\begin{array}{llll} & \textit{Primal} & & \textit{Dual} \\ \min_{\mathbf{w}} & \frac{\lambda}{2}\|\mathbf{w}\|^2+\xi & \max_{\boldsymbol{\alpha}} & -\frac{1}{2\lambda}\|\boldsymbol{\alpha} A_t\|^2+\boldsymbol{\alpha} B_t \\ s.t & \langle a_j^t,\mathbf{w}\rangle + b_j^t \le \xi\ \forall j\in J_t & s.t & \alpha_j\ge 0\ \forall j \in J_t; \tilde{\alpha} \ge 0 \\ & \langle \tilde{a}_{t-1}^t, \mathbf{w}\rangle + \tilde{b}_{t-1}^t \le \xi & & (\sum_{j\in tJ_t}\alpha_j)+\tilde{\alpha} = 1\end{array}$$

where $A_t = [\ldots;\mathbf{a}_j^t;\ldots;\tilde{\mathbf{a}}_{t-1}^t]$ is a matrix (with $\mathbf{a}_j^t$ and $\tilde{\mathbf{a}}_{t-1}^t$ being row vectors), $B_t = [\ldots;b_j^t;\ldots;\tilde{b}_{t-1}^t]$ is the vector of scalars and $\boldsymbol{\alpha}$ stands for the (row) vector of Lagrange multipliers (of length $|J_t|+1$ at iteration t). We denote α_j as the Lagrange multiplier associated with the CP c_j^t and we denote $\tilde{\alpha}$ as the Lagrange multiplier associated with the aggregated CP $\tilde{c}_{j-1}^t$. Let $\boldsymbol{\alpha}_t$ be the solution of the above dual program then the minimizer of the primal can be expressed as:

$$\tilde{\mathbf{w}}_t = -\frac{\boldsymbol{\alpha}_t A_t}{\lambda}.$$

Hence the definition of the aggregated cutting plane follows:

$$\begin{array}{ll} \tilde{\mathbf{a}}_t & = \boldsymbol{\alpha}_t A_t, \\ \tilde{b}_t & = \boldsymbol{\alpha}_t B_t. \end{array}$$

*Locality measure associated to the aggregated cutting plane.*The aggregated CP $\tilde{c}_t^t$ accumulates information from many cutting planes built at different points so that one cannot immediately define a locality measure $\tilde{s}_t^t$ between $\tilde{c}_t^t$ and the current best observed solution $\mathbf{w}_t^*$. However, $\tilde{c}_t^t$ being a convex combination of cutting planes, we chose to define $\tilde{s}_t^t$ as the corresponding convex combination of locality measures associated to cutting planes:

$$\tilde{s}_t^t = \sum_{j\in J_t} \alpha_j s_j^t + \tilde{\alpha}\tilde{s}_{t-1}^t.$$

Interestingly using this aggregated locality measure, one can show that there is no conflict between $\tilde{c}^t_t$ and $\mathbf{w}^*_t$ since $R(\mathbf{w}^*_t) - \tilde{c}^t_t(\mathbf{w}^*_t) \geq \tilde{s}^t_t$. Indeed, we have:

$$\begin{aligned} R(\mathbf{w}^*_t) - c^t_j(\mathbf{w}^*_t) &\geq s^t_j \ \forall j \in J_t, \\ R(\mathbf{w}^*_t) - \tilde{c}^t_{t-1}(\mathbf{w}^*_t) &\geq \tilde{s}^t_{t-1}. \end{aligned}$$

Multiplying these equations by α_j's and $\tilde{\alpha}$ then taking the sum gives the result:

$$R(\mathbf{w}^*_t) - \tilde{c}^t_t(\mathbf{w}^*_t) \geq \tilde{s}^t_t.$$

3.3 Variants

In this section we discuss two variants (and their implementations issues) that allow speeding up convergence in practice.

3.3.1 REGULARIZATION

In previous section we presented our method with a standard L2 regularization term $\frac{\lambda}{2}\|\mathbf{w}\|^2$. Yet this choice is not always a good one for non-convex optimization problems where convergence to a poor local optima is a severe problem. Alternatively one may prefer to regularize around a first reasonable solution $\mathbf{w}^{reg}$ and use a regularization term such as $\|(\mathbf{w} - \mathbf{w}^{reg})\|^2$. For instance to learn Hidden Markov Models with a large margin criterion using a variant of NRBM, we used a model learned with Maximum Likelihood as $\mathbf{w}^{reg}$ (Do and Artières, 2009). Furthermore, if all parameters in $\mathbf{w}$ do not have the same nature (magnitude) then using only one weight-cost (λ) for all parameters is not wise. So one may prefer the following regularization term:

$$\frac{\lambda}{2}\|(\mathbf{w} - \mathbf{w}^{reg}) \otimes \theta\|^2$$

where θ is a positive vector of regularization weights and $\otimes$ stands for element-wise product. The use of different θ values depending on the parameters allows introducing some prior information. Again, taking our example of learning Hidden Markov Models, we used different θ values for regularizing transition probabilities and emission probabilities parameters.

3.3.2 FAST VARIANT WITH LINE SEARCH

In Algorithm 4, the minimum point of the approximation function is not guaranteed to be a better solution than the current best observed solution, which may result in null steps. Few works showed that one can speed up cutting plane based methods with a linesearch procedure (Franc and Sonnenburg, 2008; Do and Artières, 2008), which may be efficient to compute in some cases (e.g. primal objective of linear SVM).

The idea is that a line search ensures that we get a better solution every iteration, assuming that the search direction is a descent direction. If the search direction is not a descent direction then the line search returns the best solution along the search direction (should be close to the current solution), which will be used to build a new cutting plane in the next iteration. In our case, without specific knowledge of $f(\mathbf{w})$ we use a general line search technique.

Since the line search may require considerable more function/subgradient evaluations, one can initialize the step size based on the step size reached in previous iteration. In our implementation (a line search with Wolfe conditions), initial step size is computed so that the step length is the same as the final stepsize in previous iteration. This simple implementation works well and most of the time we need only one function/subgradient evaluation (when initial step size satisfies Wolfe conditions).

We investigated two strategies. In the *full line search strategy* , every iteration we add two cutting planes to the approximation problem, one at the minimum point of the current approximated problem and one at the solution of the line search. In this case, the role of the line search is to improve the quality of the approximated problem every iteration. In the *greedy line search strategy* we consider adding only one cutting plane at the solution of the line search in order to limit the number of function/subgradient evaluation at each iteration. This strategy also works well in practice as we will see in experiment section.

4. Convergence Analysis

In this section, we provide theoretical results for our algorithm. For a convex objective function, when disabling locality measure (putting these to 0), our algorithm can be viewed as a limited memory variant of CRBM, and we provide a proof on the convergence rate of the algorithm under a standard assumption. For non-convex objective function, the convergence analysis is much more complicated and requires a disputable assumption. For these reasons, we only present main results for the non-convex case in this paper, while the corresponding proofs can be found in an internal report (Do and Artieres, 2012).

4.1 Convergence Analysis for NRBM: Convex Case

We provide in this section theoretical results on the convergence behavior of our algorithm applied to convex risks. First we present a theorem in Section 4.1.2 which characterizes its convergence rate and shows that our algorithm inherits the fast convergence rate of CRBM from which it is inspired (note that we consider here the particular case of quadratic regularization with non-smooth objective function).

In the case of a convex risk one can either use the convex version of our algorithm which remains to using Algorithm 3 or the non-convex version (Algorithm 4) while disabling all locality measure (i.e. putting these to 0, Algorithm 4 will become Algorithm 3 since conflicts will not occur for convex risk). We prove in the following the main results for the convex version.

4.1.1 ASSUMPTIONS

The necessary assumption for proving our main results are the following:

- H1 : The empirical risk is Lipschitz continuous with a constant G.

H1 is a rather standard assumption, which was used for proving convergence results in previous works (Smola et al., 2008; Shalev-Shwartz et al., 2007; Joachims, 2006). It is in particular a reasonable assumption in case of smooth almost everywhere risks such as those one gets using hinge loss and maximum margin criterion (SVM, structured output prediction, etc).

4.1.2 MAIN RESULTS

We provide here an upper bound on the convergence rate of our variant of limited memory CRBM, by studying the decrease of the gap, defined as the difference between the minimum observed value of the objective and the minimum of the current approximated problem, with iteration number. Indeed, this gap can be used for bounding from above the accuracy of the current solution (in terms of the objective value).

We begin with some preliminary results. Lemmas 1 and 2 are general results that are needed for Lemma 3 which establishes a lower bound on the improvement of the approximation gap at each iteration.

Lemma 1 *Teo et al., 2007 The minimum of* $\frac{1}{2}qx^2 - lx$ *with* $l, q > 0$ *and* $x \in [0,1]$ *is bounded from above by* $-\frac{l}{2}\min(1, l/q)$

Lemma 2 *Function* $h(x) = x - \frac{x}{2}\min(1, x/q)$ *is monotonically increasing for all* $q > 0$.

Proof We have :

$$h(x) = \begin{cases} x - x^2/2q & if x < q \\ x/2 & if x \geq q \end{cases}$$

where $x/2$ is always monotonically increasing, then h is for monotonically increasing for $x \geq q$. For $x \in (-\infty, q)$, $h'(x) = 1 - x/q > 0$ because $x < q$ and $q > 0$. Moreover, h is continuous (at $x = q$), thus h is monotonically increasing whatever x. ■

Lemma 3 *The approximation gap decreases according to:*

$$gap_{t-1} - gap_t \geq \min(\frac{gap_{t-1}}{2}, \frac{(gap_{t-1})^2\lambda}{8G^2}) \tag{18}$$

where the approximation gap is defined as $gap_t = f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t)$.

Proof We focus on deriving an underestimator on the minimum value of $g_t(\mathbf{w})$ based solely on this aggregated cutting plane and on the new added cutting plane at iteration t. This is simpler than exploiting the complete approximation function. Note that this is possible since the aggregated cutting plane accumulates information about the approximation problem at previous iterations. We have:

$$g_t(\mathbf{w}) \geq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max\left[\langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}\rangle + \tilde{b}_{t-1}, \langle \mathbf{a}_t, \mathbf{w}\rangle + b_t\right]. \tag{19}$$

Let find the minimum of the right side. The dual program of this minimization problem is:

$$\begin{array}{ll} \max_{\tilde{\alpha}_{t-1},\alpha_t} & -\frac{\lambda}{2}\|\frac{\tilde{\alpha}_{t-1}\tilde{\mathbf{a}}_{t-1}+\alpha_t\mathbf{a}_t}{\lambda}\|^2 + \tilde{\alpha}_{t-1}\tilde{b}_{t-1} + \alpha_t b_t \\ s.t & 0 \leq \tilde{\alpha}_{t-1}, \alpha_t \leq 1 \\ & \tilde{\alpha}_{t-1} + \alpha_t = 1 \end{array}$$

where $\tilde{\alpha}_{t-1}, \alpha_t \in \mathbb{R}$ are Lagrange multipliers. This quadratic program has 2 variables and can be further simplified as:

$$\begin{array}{lll} & \max\limits_{\alpha_t\in[0,1]} & -\frac{1}{2\lambda}\|\tilde{\mathbf{a}}_{t-1} + \alpha_t(a_t - \tilde{\mathbf{a}}_{t-1})\|^2 + \alpha_t(b_t - \tilde{b}_{t-1}) + \tilde{b}_{t-1} \\ = & \max\limits_{\alpha_t\in[0,1]} & -\frac{1}{2\lambda}\|\mathbf{a}_t - \tilde{\mathbf{a}}_{t-1}\|^2(\alpha_t)^2 + (\frac{\|\tilde{\mathbf{a}}_{t-1}\|^2}{\lambda} - \frac{\langle \mathbf{a}_t, \tilde{\mathbf{a}}_{t-1}\rangle}{\lambda} + b_t - \tilde{b}_{t-1})\alpha_t - \frac{\|\tilde{\mathbf{a}}_{t-1}\|^2}{2\lambda} + \tilde{b}_{t-1} \\ = & -\min\limits_{\alpha_t\in[0,1]} & \frac{1}{2}q(\alpha_t)^2 - l\alpha_t - g_{t-1}(\mathbf{w}_t) \end{array} \tag{20}$$

where $q = \frac{\|\mathbf{a}_t - \tilde{\mathbf{a}}_{t-1}\|^2}{\lambda}$ and $l = \frac{\|\tilde{\mathbf{a}}_{t-1}\|^2}{\lambda} - \frac{\langle \mathbf{a}_t, \tilde{\mathbf{a}}_{t-1}\rangle}{\lambda} + b_t - \tilde{b}_{t-1}$.

Note that $\mathbf{w}_t = \tilde{\mathbf{w}}_{t-1} = -\frac{\tilde{\mathbf{a}}_{t-1}}{\lambda}$. Hence the linear factor may be rewritten as:

$$\begin{aligned} l &= \frac{\|\tilde{\mathbf{a}}_{t-1}\|^2}{\lambda} - \frac{\langle \mathbf{a}_t, \tilde{\mathbf{a}}_{t-1}\rangle}{\lambda} + b_t - \tilde{b}_{t-1} \\ &= \langle \mathbf{a}_t, \mathbf{w}_t \rangle + b_t - \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}_t \rangle - \tilde{b}_{t-1} \\ &= \frac{\lambda}{2}\|\mathbf{w}_t\|^2 + \langle \mathbf{a}_t, \mathbf{w}_t \rangle + b_t - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - \langle \tilde{\mathbf{a}}_{t-1}, \mathbf{w}_t \rangle - \tilde{b}_{t-1} \\ &= f(\mathbf{w}_t) - g_{t-1}(\mathbf{w}_t). \end{aligned}$$

Using Lemma 1 the maximum value in Equation 20 is greater or equal than $\frac{l}{2}\min(1, l/q) + g_{t-1}(\mathbf{w}_t) = \frac{f(\mathbf{w}_t) - g_{t-1}(\mathbf{w}_t)}{2} \min\left(1, \frac{f(\mathbf{w}_t) - g_{t-1}(\mathbf{w}_t)}{q}\right) + g_{t-1}(\mathbf{w}_t)$. This latter quantity is then a lower bound of the minimum of the right side in Equation 19, thus:

$$\begin{aligned} & g_t(\mathbf{w}_{t+1}) && \geq \min\left(\frac{f(\mathbf{w}_t) - g_{t-1}(\mathbf{w}_t)}{2}, \frac{(f(\mathbf{w}_t) - g_{t-1}(\mathbf{w}_t))^2}{2q}\right) + g_{t-1}(\mathbf{w}_t) \\ \Rightarrow\ & g_t(\mathbf{w}_{t+1}) && \geq \min\left(\frac{f(\mathbf{w}_t^*) - g_{t-1}(\mathbf{w}_t)}{2}, \frac{(f(\mathbf{w}_t^*) - g_{t-1}(\mathbf{w}_t))^2}{2q}\right) + g_{t-1}(\mathbf{w}_t) \\ \Rightarrow\ & f(\mathbf{w}_t^*) - g_t(\mathbf{w}_{t+1}) && \leq f(\mathbf{w}_t^*) - g_{t-1}(\mathbf{w}_t) - \min\left(\frac{f(\mathbf{w}_t^*) - g_{t-1}(\mathbf{w}_t)}{2}, \frac{(f(\mathbf{w}_t^*) - g_{t-1}(\mathbf{w}_t))^2}{2q}\right). \end{aligned}$$

Note that $f(\mathbf{w}_t^*) \leq f(\mathbf{w}_{t-1}^*)$. Replacing $f(\mathbf{w}_t^*)$ by $f(\mathbf{w}_{t-1}^*)$ in the right side of previous equation and using Lemma 2 one gets:

$$\begin{aligned} & gap_t && \leq f(\mathbf{w}_{t-1}^*) - g_{t-1}(\mathbf{w}_t) - \min\left(\frac{f(\mathbf{w}_{t-1}^*) - g_{t-1}(\mathbf{w}_t)}{2}, \frac{(f(\mathbf{w}_{t-1}^*) - g_{t-1}(\mathbf{w}_t))^2}{2q}\right) \\ \Leftrightarrow\ & gap_t && \leq gap_{t-1} - \min\left(\frac{gap_{t-1}}{2}, \frac{gap_{t-1}^2}{2q}\right). \end{aligned}$$

Finally since $q = \frac{1}{\lambda}\|\mathbf{a}_t - \tilde{\mathbf{a}}_{t-1}\|^2 \leq 4G^2/\lambda$, and substituting this back in previous formula gives the result. ■

Theorem 1 *Algorithm 3 produces an approximation gap below* ε *in* $O(1/\lambda\varepsilon)$ *iterations. More precisely it reaches a approximation gap below* ε *after* T *steps with:*

$$\begin{aligned} & T \leq T_0 + 8G^2/\lambda\varepsilon - 2 \\ & \textit{with } T_0 = 2log_2 \frac{\lambda\|\mathbf{w}_1 + \mathbf{a}_1/\lambda\|}{G} - 2. \end{aligned}$$

Proof Let consider the two quantities occurring in Equation 18, $gap_{t-1}/2$ and $\lambda gap_{t-1}^2/8G^2$.

We first show that the situation where $gap_{t-1}/2 > \lambda gap_{t-1}^2/8G^2$ (i.e. $gap_{t-1} > 4G^2/\lambda$) may only happen a finite number of iterations, T_0. Actually if $gap_{t-1} > 4G^2/\lambda$ Lemma 3 shows that $gap_t \leq gap_{t-1}/2$ and the gap is at least divided by two every iteration. Then $gap_{t-1} > 4G^2/\lambda$ may arise for at most $T_0 = log_2(\lambda gap_1/4G^2) + 1$. Since $gap_1 = \frac{\lambda}{2}\|\mathbf{w}_1 + \mathbf{a}_1/\lambda\|^2$ (it may be obtained analytically since the approximation function in the first iteration is quadratic), $T_0 = 2log_2 \frac{\lambda\|\mathbf{w}_1 + \mathbf{a}_1/\lambda\|}{G} - 2$.

Hence after at most T_0 iterations the decrease of the gap obeys $gap_t - gap_{t-1} \leq -gap_{t-1}^2/8G^2 \leq 0$. To estimate the number of iterations required to reach $gap_t \leq \varepsilon$ we introduce a function $u(t)$ which is an upper bound of gap_t (Teo et al., 2007). Solving differential equation $u'(t) = -\frac{\lambda}{8G^2}u^2(t)$ with boundary condition $u(T_0) = 4G^2/\lambda$ gives $u(t) = \frac{8G^2}{\lambda(t+2-T_0)} \geq gap_t/\forall t \geq T_0$. Solving $u(t) \leq \varepsilon \iff t \geq 8G^2/\lambda\varepsilon + T_0 - 2$, the solution is reached with accuracy ε within $\left[T_0 + 8G^2/\lambda\varepsilon - 2\right]$ iterations. ■

Next we show that if the algorithm reaches a null gap then it has found the global minimum.

Theorem 2 *If $gap_t = 0$ at iteration t of Algorithm 3, then $\mathbf{w}_t^* = \tilde{\mathbf{w}}_t$ and $\mathbf{w}_t^*$ is the global minimum of f.*

Proof

We have $g_t(\mathbf{w}_t^*) = f(\mathbf{w}_t^*)$ since the approximation errors are zero at points where cutting plane were built. Hence, the null gap between $f(\mathbf{w}_t^*)$ and the minimum of g_t, $g_t(\tilde{\mathbf{w}}_t)$, indicates that $g_t(\mathbf{w}_t^*) = g_t(\tilde{\mathbf{w}}_t)$, i.e., $\mathbf{w}_t^* \equiv \tilde{\mathbf{w}}_t$. Since $g_t(\mathbf{w})$ is an underestimator of $f(\mathbf{w})$, it's minimum value,$g_t(\tilde{\mathbf{w}}_t)$,is less than or equal to the minimum of $f(\mathbf{w})$. Therefore, $\mathbf{w}_t^*$ is the minimum of $f(\mathbf{w})$. ■

Note that in the case the algorithm can not reach null gap after a finite iterations, both $f(\mathbf{w}_t^*)$ and $g_t(\tilde{\mathbf{w}}_t)$ converge to the minimum of f, $f(\mathbf{w}^*)$, since $f(\mathbf{w}_t^*) \geq f(\mathbf{w}^*)$, $g_t(\tilde{\mathbf{w}}_t) \leq f(\mathbf{w}^*)$ and $f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t) \to 0$.

4.1.3 EXTENSION OF RESULTS TO VARIANTS WITH LINE SEARCH

As our proof is based on the cutting plane built at the minimum of approximated problem $\langle \mathbf{a}_t, \mathbf{w} \rangle + b_t$, and the aggregated cutting plane, $\langle \tilde{\mathbf{a}}_t, \mathbf{w} \rangle + \tilde{b_t}$, all theoretical results hold for the *full line search variant* for which the two CPs are present in the approximation problem.

However, the things are more complicated for the *greedy line search strategy* and the proofs do not hold anymore in their actual shape. Yet, such a strategy is less expensive than the *full* one and it is efficient in practice. All results of the line search variant in the experiment section have been gained using this implementation.

4.2 Convergence Analysis for NRBM : Non-Convex Case

We provide in this section theoretical results on the convergence behavior of our algorithm. First we present a theorem in Section 4.2.2 which characterizes its convergence rate and shows that our algorithm inherits fast convergence rate of CRBM from which it is inspired (note that we consider here the particular case of quadratic regularization with non-smooth objective function). Next we provide theorem that characterizes the solution the algorithm converges to. First of all we detail and discuss the necessary assumptions used for proving these results, then we present our main results. See Do and Artieres (2012) for detailed proofs.

4.2.1 ASSUMPTIONS

The necessary assumptions for proving our main results are the following:

- H1 : The empirical risk is Lipschitz continuous with a constant G.
- H2 : The number of iterations where a conflict is solved by modifying the normal vector $\mathbf{a}_t$ (NullStep2 case in Algorithm 4) is finite.

Under the H1 assumption, we could get the same theoretical results on convergence rate as previous section for non-convex objective function. In other words, we can prove (with a more complicated proofs) that the approximation gap_t decrease towards zeros with a rate $O(1/\lambda\varepsilon)$ and that the algorithm has found a stationary solution if a null gap is reached. However, these results do not imply the convergence to a global minimum for the non-convex case, and we need to provide additional results for proving that the algorithm generates stationary solutions.

Our proof on the convergence towards stationary solution require an additional assumption H2, which states that the number of NullStep2 in Algorithm 4 is finite. Recall that there is a NullStep2 at iteration t if and only if the raw cutting plane built at current solution $\mathbf{w}_t$ is not compatible with the best observed solution $\mathbf{w}_t^*$. Hence, since the current solution and the best observed solution get closer as the iteration number increases we may hope that NullStep2 do not arise after a finite number of iterations. Furthermore, it is very likely that if the algorithm gets close enough to a stationary solution $\mathbf{w}^*$ lying within a smooth area then it should converge towards this stationary solution without conflicts anymore, as it would do in case of a convex and smooth objective. This is particularly expected for our algorithm (compared to standard non-convex bundle methods) since it focuses on maintaining a good approximation function around the best current solution. Another important point is that we did not observe any case of infinite number of conflicts in our experiments (on both academic optimization problems and machine learning problems) where NullStep2 mainly occurred in a few early iterations.

At the end these claims are still not proved so that the convergence of NRBM to a stationary solution is not fully proved here, but we believe that our convergence analysis establishes some important elements towards a fast and fully proved bundle method for minimizing non-convex regularized function.

4.2.2 MAIN RESULTS

Similar to the case of convex risk, we can prove that the approximation decreases as the algorithm iterates, under the hypothesis H1 only.

Lemma 4 *The approximation gap of Algorithm 4 decreases according to:*

$$gap_{t-1} - gap_t \geq \min(\frac{gap_{t-1}}{2}, \frac{(gap_{t-1})^2\lambda}{8G^2})$$

where the approximation gap is defined as $gap_t = f(\mathbf{w}_t^*) - g_t(\tilde{\mathbf{w}}_t)$.

Proof The proof is provided in supplementary material. ■

This lemma implies a first theorem that provides a theoretical lower bound on convergence speed.

Theorem 3 *Algorithm 4 reaches a gap below* ε *with a number of iterations* $O(1/\lambda\varepsilon)$.

Next we show that if the algorithm reaches a null gap then it has found a stationary solution.

Theorem 4 *If* $gap_t = 0$ *at iteration* t *of Algorithm 4, then* $\mathbf{w}_t^* = \tilde{\mathbf{w}}_t$ *and* $\mathbf{w}_t^*$ *is a stationary point of objective function* f, *i.e.* $\mathbf{0} \in \partial f(\mathbf{w}_t^*)$.

Optimizer	Non-Convex	Non-Smooth	line-search
Non-convex Bunlde Method (NBM)	yes	yes	yes
LBFGS	no	no	yes
Stochastic Gradient Descend (SGD)	yes	yes	no
Subgradient Descend (SG)	yes	yes	no
Concave-Convex Procedure (CCCP)	dedicated solver for difference of convex functions		
SVMstruct	dedicated solver for a particular convex problem		
UniverSVM	dedicated solver for a particular non-convex problem		

Table 1: List of solvers that are considered in the experimental comparison.

The two following theorems say that, under Hypothesis H1 and H2, if the sequence $(\mathbf{w}_t)$ and $(\mathbf{w}_t^*)$ generated by NRBM Algorithm are infinite they have cluster points, and these cluster points are stationary solutions.[7]

Theorem 5 *If Algorithm 4 does not reach a stationary solution in a limited number of iterations, the two infinite sequences $(\mathbf{w}_t)$ and $(\mathbf{w}_t^*)$ generated by Algorithm 4 have cluster points.*

Theorem 6 *Let $\mathbf{w}^*$ be a cluster point of the sequence $(\mathbf{w}_t^*)$. Then under assumptions H1 and H2, $\mathbf{w}^*$ is a stationary solution of $f(\mathbf{w})$.*

4.2.3 EXTENSION OF RESULTS TO VARIANTS

Similarly to the convex case, our results on convergence rate and vonvergence to a stationary solution hold for the *full line search strategy* but not for the *greedy line search strategy*, the latter being less expensive and equally accurate in practice.

5. Experiments

In this section, we compare our optimization method NRBM to standard and non-standard (i.e., methods designed for solving a particular machine learning problem) optimizers listed in Table 1. We also implemented the sped-up version of NRBM with soft line search procedure, this latter is called NRBMLS. The implementation of our two algorithms NRBM and NRBMLS are in Matlab (available at `https://forge.lip6.fr/projects/nrbm`), and the implementation of NBM is in Fortran (available at http://napsu.karmitsa.fi/lmbm/).

First a series of experiments has been performed on artificial problems where we tested optimization algorithms for optimizing a manually defined non-convex and/or non smooth objective function. This allows deep understanding of the behavior of our approach. Then we consider machine learning problems of increasing optimization difficulty. We first consider a convex and smooth optimization problem (learning a CRF). Next we consider a convex and non smooth optimization problem (learning a M3N). Next we consider a non-convex and non smooth optimization problem, learning a transductive SVM. Finally we study two larger scale non-convex optimization problems for learning graphical models for speech and handwriting recognition: learning Hidden Markov Models with a large margin criterion (non smooth) (Do and Artières, 2009); and learning a model

[7] Let $\{x_n\}$ be a sequence of real vectors, then x is a cluster point of $\{x_n\}$ if for every $\varepsilon > 0$, there are infinitely many points x_n such that $\|x - x_n\| < \varepsilon$.

mixing a deep neural network feature extractor and conditional random fields (smooth) (Do and Artières, 2010). For each of these optimization problems we compare our methods with state of the art dedicated optimization methods. Note that although many optimizer are implemented in Matlab, dynamic programming for structured problems (CRF, M3N, CDHMM, NeuroCRF) are written in C mex-files.

5.1 Artificial Test Problems

Experiments were carried out on two academic non-convex test problem problem (Haarala et al., December 2004). We followed here experimental settings with a few modifications. Actually, we add a regularization term to the initial solution of each problem. We did not use the origin as regularization point, which may lead to trivial optimum solution **0** and optimum value 0, since this may cause numeric problems when using relative tolerance on objective value. In the following, we note w_i the i^{th} coordinate of vector $\mathbf{w} \in \mathbb{R}^D$ in the search space. We note $\mathbf{w}^{(0)}$ the initial solution.

The two objective functions that we seek to optimize are named and defined as follows. Note that both problems may be instantiated with a varying number of dimensions (i.e. parameters), D, this allows investigating small to larger scale problems.
Chained Mifflin 2

$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}-\mathbf{w}^{(0)}\|^2 + \sum_{i=1}^{D-1}\left(-w_i + 2(w_i^2 + w_{i+1}^2 - 1) + 1.75|w_i^2 + w_{i+1}^2 - 1|\right)$$

with $w_i^{(0)} = -1$ for all $i = 1, ..., D$.
Chained Cessent 2

$$f(\mathbf{w}) = \tfrac{\lambda}{2}\|\mathbf{w}-\mathbf{w}^{(0)}\|^2 + \textstyle\sum_{i=1}^{D-1} \max \quad \begin{bmatrix} w_i^2 + (w_{i+1}-1)^2 + w_{i+1} - 1\ , \\ -w_i^2 - (w_{i+1}-1)^2 + w_{i+1} + 1 \end{bmatrix}$$

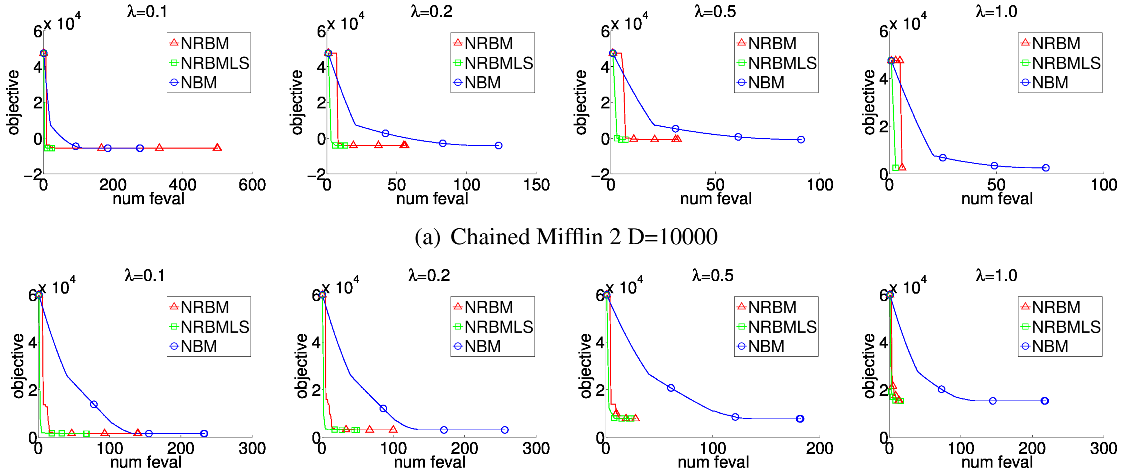
with

$$w_i^{(0)} = \begin{cases} -1.5 & when \quad mod(i,2) = 1 \\ \ \ 2.0 & when \quad mod(i,2) = 0 \end{cases} .$$

We compare our algorithms, NRBM and NRBMLS (NRBM with linesearch), and standard Non-Convex Bundle Method (NBM). In order to do so we conducted extensive experiments to investigate the respective convergence behaviour of the methods (convergence rate and quality of the solution found). Tables 2 and 3 report results gained for both data sets Chained Mifflin 2 and Chained 2 Cessent for NBM, NRBM and NRBMLS for various experimental settings: Data dimensionality D ranges from 10^2 up to 10^5, and regularization parameter λ ranges from 0.1 to 1.0. We compare the three algorithms with respect to:

- The value of the objective at the solution found.
- The number of objective evaluations needed.
- The cpu time of the optimization (indicative).

For NRBM and NRBMLS, optimization is performed until the approximation gap becomes less than 0.1% of the absolute value of objective function (relative tolerance 10^{-3}), and we also set the relative tolerance of NBM to 10^{-3}. Note however that the two stopping criteria do not coincide, which may lead to different final accuracy (we will come back on this point later in sections 5.5 and



(a) Chained Mifflin 2 D=10000

(b) Chained Cessent 2 D=10000

Figure 8: Comparison of convergence behavior for NRBM, NRBMLS, and NBM. Figures show the value of the objective function as a function of the number of objective function evaluations. There is one row of plots per data set, with a plot for every value of λ.

5.6). Note also that the CPU time of the optimization is not indicative and should be taken carefully since implementations are not equally optimized.

We may draw some comments from these tables (note that we observed similar results, not included here, on few other artificial data sets than the two studied here). First we focus on the linesearch efficiency by comparing NRBM and NRBMLS in terms of convergence rate (measured by the number of evaluations of the objective) and of accuracy of the solution reached. One can easily observe that in some cases NRBMLS performs similarly as NRBM but in most cases it significantly improves convergence rate and leads to a better solution, whatever the dimensionality D, and whatever the amount of regularization λ.

From the point of view of convergence accuracy NBM often outperforms both of our methods, NRBM and NRBMLS, and converges to a better solution with a very slightly lower objective value. This is reasonable since NBM uses an additional stopping condition which is similar to the one used in proximal bundle method. However, the results in Figure 8 suggest that NRBM and NRBMLS are faster than NBM to reach a reasonable solution.

From the convergence rate point of view, NBM is faster than NRBM for low dimensionality and low λ only, but NRBM is faster than NBM when λ increases for low dimensionality and whatever λ for high dimensional problems. NRBMLS is always faster than NRBM and NBM, whatever λ and whatever the problem dimensionality. Depending on the settings, NRBM and NRBMLS may be up to 50 times faster than NBM, and this is particularly true for high dimensional optimization problems. Finally, as λ gets bigger, both NRBM and NRBMLS converge faster, as expected by the theoretical proven convergence rate $O(\frac{1}{\lambda\varepsilon})$. On the contrary, NBM cannot always benefit from the large value of λ (see Table 3).

At the end, our algorithms are shown to converge towards solutions that compare well to the ones found by standard non-convex bundle methods but they do converge much faster and the benefit seems to increase with the dimensionality of the problem. This suggests that our methods are better candidates for large scale machine learning problems involving non-convex optimization.

$D = 10^2$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
	obj.	eval	time	obj.	eval	time	obj.	eval	time	obj.	eval	time
NBM	-55.68	254	0.03s	-41.32	29	0.00s	-8.167	12	0.00s	24.92	10	0.00s
NRBM	-54.19	501	1.52s	-41.31	280	0.44s	-8.163	20	0.02s	24.93	8	0.01s
NRBMLS	-55.66	45	0.03s	-41.32	18	0.01s	-8.165	7	0.01s	24.93	5	0.01s
$D = 10^3$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	-560.9	157	0.16s	-416.3	48	0.04s	-83.17	30	0.03s	249.9	23	0.02s
NRBM	-556.2	501	1.96s	-416.3	163	0.39s	-83.16	19	0.04s	250.0	7	0.02s
NRBMLS	-560.7	43	0.08s	-416.3	19	0.04s	-83.16	7	0.02s	250.0	4	0.01s
$D = 10^4$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	-5613	431	4.77s	-4166	123	2.20s	-833.2	91	1.44s	2500	73	0.99s
NRBM	-5609	501	13.77s	-4166	67	2.13s	-832.8	34	1.06s	2500	6	0.19s
NRBMLS	-5611	24	0.54s	-4166	13	0.36s	-833.2	7	0.18s	2500	3	0.09s
$D = 10^5$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	-56138	407	96.64s	-41666	363	94.75s	-8333	284	71.54s	25000	214	50.87s
NRBM	-56119	172	57.57s	-41661	33	12.84s	-8332	32	9.2s	25000	6	1.73s
NRBMLS	-56097	30	7.19s	-41664	17	6.14s	-8333	7	1.74s	25000	3	0.85s

Table 2: Chained Mifflin 2 data set. Comparative results of convergence quality (objective value, column *obj*) and convergence rate (number of evaluations of the objective function, column *eval* and cputime, column *time*) for the three optimization methods NBM, NRBM and NRBMLS, for various values of regularization level (λ) and for various data dimensionality (D).

5.2 CRF Training: Smooth and Convex Objective Function

To begin with, we perform experiments with a smooth and convex objective function for learning a conditional random field (CRF), and compare NRBMLS (convex setting) with NBM, Stochastic Gradient Descent (SGD), and LBFGS, a popular choice of optimizer for CRF.[8] Note that both NRBM and LBFGS use an approximation of the objective function (non-smooth piece-wise quadratic function in NRBM and a second order approximation in LBFGS). While the smooth approximation in LBFGS is suitable for smooth objective function, we would like to know how the non-smooth approximation works for smooth objective function. For LBFGS, the algorithm stop once the ratio between the norm of the gradient and the norm of the solution is smaller than a small tolerance. In our experiment, we find a tolerance parameter which yields comparative accuracies between LBFGS and NRBMLS for several values of λ, but we could not guarantee fair stopping criterion in all cases. The SGD setting is based on Leon Bottou implementation (Bottou, 2008).

We use a fixed tolerance on the norm of gradient for LBFGS which yields comparable accuracies to NRBMLS for several values of λ.

[8] We used a Matlab implementation of LBFGS provided by Fei Sha.

$D = 10^2$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
	obj.	eval	time	obj.	eval	time	obj.	eval	time	obj.	eval	time
NBM	15.75	58	0.00s	31.23	105	0.01s	77.84	124	0.02s	152.2	145	0.02s
NRBM	15.64	334	81.71s	31.23	132	1.91s	77.93	66	1.14s	152.3	21	0.02s
NRBMLS	15.63	111	8.12s	31.21	103	4.66s	79.15	49	0.38s	152.2	29	0.02s
$D = 10^3$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	156.3	180	0.21s	312.5	194	0.35s	780.9	146	0.17s	1531	163	0.18s
NRBM	157.1	501	11.44s	312.8	104	4.34s	781.6	41	0.32s	1532	15	0.03s
NRBMLS	156.3	105	11.75s	313.0	82	1.27s	781.5	36	0.18s	1531	17	0.03s
$D = 10^4$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	1563	276	5.76s	3125	311	6.49s	7812	194	3.06s	15316	289	4.02s
NRBM	1564	218	25.31s	3128	72	2.74s	7818	29	0.78s	15319	14	0.32s
NRBMLS	1564	75	6.9s	3127	42	2.64s	7829	22	0.40s	15327	16	0.21s
$D = 10^5$	$\lambda = 0.1$			$\lambda = 0.2$			$\lambda = 0.5$			$\lambda = 1$		
NBM	15625	628	154.32s	31251	660	178.43s	78126	636	143.97s	1.562e+5	628	151s
NRBM	15635	106	49.97s	31272	48	20.38s	78146	21	8.38s	1.532e+5	14	3.86s
NRBMLS	15638	62	30.12s	31269	48	16.29s	78247	21	6.24s	1.532e+5	16	2.51s

Table 3: Chained Cessent 2. Comparative results of convergence quality (objective value, column *eval*) and convergence rate (number of evaluations of the objective function, *eval* and cputime, column *time*) for the three optimization methods NBM, NRBM and NRBMLS, for various values of regularization level (λ) and for various data dimensionality (D).

When working with convex function, we can simply disable the use of the locality measure and the conflict handling procedure so that the method, NRBMLS-convex, becomes much similar to CRBM. Note that in NRBMLS-convex, we use the soft linesearch in order to speed up the convergence rate.

We conducted experiments on an OCR data set used by Taskar et al. (2004) for evaluating Maximum Margin Markov Networks. This OCR data set consists of 6877 words which correspond to roughly 52,000 characters (Kassel, 1995). OCR data are sequences of isolated characters (each represented as a binary vector of dimension 128) belonging to 26 classes. The data set was divided in 10 folds for cross validation. We use here the 8 first folds for training and the two last fold for testing.

Table 4 shows results of the four optimization algorithms for training CRFs with various regularization values (λ). As can be seen, the three batch optimization methods (NRBMLS, NBM, and LBFGS) significantly outperform SGD for this task, which is in contradiction to Bottou's result where SGD clearly outperformed LBFGS for the CONLL2000 data (Bottou, 2008). We believe that the difference between these two contradictory results may come from the fact that the OCR problem is maybe more complex, meaning that the OCR problem would require more training data (the predictive error rates on the two data sets are 14% for OCR and 6% for CONLL2000). This suggests that the OCR data is less redundant than the CONLL2000 data, which would explain why SGD, whose efficiency depends on data redundancy, is less effective for OCR data.

Now looking at batch optimization methods, NRBMLS requires less iterations than NBM to reach similar objective value, it also slightly outperforms LBFGS. While these results might be biased by unequal stopping criteria, they suggest that the non-smooth approximation technique also

	NRBMLS				NBM				LBFGS			
λ	time	eval	obj.	err.	time	eval	obj.	err.	time	eval	obj.	err.
1e-04	306s	192	2.71	0.145	370s	244	2.71	0.145	340s	225	2.71	0.146
5e-05	405s	258	2.53	0.145	520s	353	2.53	0.144	458s	309	2.53	0.144
2e-05	408s	266	2.37	0.143	671s	471	2.37	0.142	583s	406	2.37	0.144
1e-05	419s	279	2.29	0.143	881s	636	2.29	0.142	583s	414	2.29	0.142
5e-06	556s	376	2.24	0.143	1199s	881	2.24	0.144	815s	593	2.24	0.143
2e-06	700s	483	2.21	0.145	1136s	834	2.21	0.145	956s	710	2.21	0.145
1e-06	753s	515	2.20	0.145	1026s	757	2.20	0.145	741s	554	2.20	0.146

	SGD											
	100 epochs			500 epochs			1000 epochs			2000 epochs		
λ	time	obj.	err.	time	obj.	err.	time	obj.	err.	time	obj.	err.
1e-04	161s	2.83	0.156	805s	2.75	0.146	1610s	2.72	0.145	3221s	2.71	0.146
5e-05	157s	2.71	0.156	784s	2.59	0.148	1567s	2.56	0.147	3134s	2.54	0.144
2e-05	151s	2.63	0.161	755s	2.48	0.153	1509s	2.44	0.151	3019s	2.41	0.147
1e-05	147s	2.60	0.165	733s	2.47	0.158	1466s	2.41	0.154	2933s	2.36	0.150
5e-06	143s	2.59	0.166	714s	2.48	0.161	1429s	2.42	0.158	2857s	2.36	0.154
2e-06	139s	2.58	0.168	696s	2.51	0.163	1392s	2.47	0.163	2784s	2.41	0.159
1e-06	137s	2.58	0.167	687s	2.52	0.167	1374s	2.50	0.164	2747s	2.45	0.163

Table 4: Comparative results of learning a CRF on OCR data set (Kassel, 1995). The maximum number of gradient stored is set to 200 for NRBMLS, NBM and LBFGS. Each row corresponds to a particular value of λ and provides for both methods the total cputime (time), the number of objective evaluation (eval), the value of the objective at convergence (obj), the classification error rate on the test set (err).

works well for smooth functions. Moreover, NRBM can exploit efficiently the regularization term in order to outperform NBM.

5.3 M3N Training: Non-Smooth and Convex Optimization Problem

In a second series of experiments we compare the efficiency of different optimization methods on learning a M3N, which yields a convex and non-smooth optimization problem. In addition to NBM and SGD, we also report the results of SVMstruct, the state of the art solver for M3N. We used the latest version SVMstruct 3.0 with 1-slack formulation, which is several orders of magnitude faster than previous methods.[9]

Here again we use NRBM-convex and a soft linesearch for speeding up convergence rate (noted NRBMLS). Note that SVMstruct 1-slack can also be viewed as a special case of CRBM, and the implementation of SVMstruct includes also some speed up techniques in the dual space.

Table 5 reports experimental results on the OCR data set (with the same setting as before). Similar to CRF experiments, we observe that SGD is not very effective. Looking at the results

[9] The implementation of SVMstruct was downloaded from: `http://svmlight.joachims.org/svm_struct.html`.

	NRBMLS				NBM				SVMstruct			
λ	time	eval	obj.	err.	time	eval	obj.	err.	time	eval	obj.	err.
1e-03	53s	158	2.92	0.143	61s	196	2.92	0.143	104s	1359	2.93	0.152
5e-04	69s	208	2.73	0.142	83s	269	2.73	0.141	137s	1849	2.74	0.148
2e-04	105s	314	2.52	0.135	151s	490	2.52	0.140	204s	2633	2.54	0.140
1e-04	161s	476	2.41	0.139	285s	926	2.41	0.138	311s	3644	2.43	0.140
5e-05	225s	644	2.33	0.137	383s	1241	2.33	0.138	468s	4975	2.35	0.138
2e-05	484s	1224	2.26	0.137	605s	1963	2.26	0.137	1432s	9292	2.28	0.139
1e-05	763s	1636	2.23	0.136	749s	2435	2.23	0.139	4834s	18369	2.25	0.143

	SGD											
	100 epochs			500 epochs			1000 epochs			2000 epochs		
λ	time	obj.	err.	time	obj.	err.	time	obj.	err.	time	obj.	err.
1e-03	41s	3.05	0.150	206s	2.96	0.146	411s	2.93	0.145	823s	2.92	0.144
5e-04	41s	2.91	0.150	207s	2.79	0.145	414s	2.76	0.144	827s	2.74	0.142
2e-04	41s	2.78	0.151	207s	2.66	0.145	415s	2.60	0.142	829s	2.56	0.140
1e-04	41s	2.71	0.149	206s	2.58	0.146	413s	2.53	0.144	825s	2.49	0.140
5e-05	41s	2.65	0.148	206s	2.52	0.145	412s	2.48	0.144	824s	2.43	0.142
2e-05	41s	2.63	0.147	205s	2.51	0.146	410s	2.47	0.145	821s	2.42	0.144
1e-05	41s	2.65	0.152	205s	2.49	0.147	409s	2.46	0.144	819s	2.42	0.144

Table 5: Comparative results of NRBM, NBM, SVMstruct and SGD for learning a M3N on OCR data set (Kassel, 1995). Each row corresponds to a particular value of λ and provides for both methods the total cputime (time), the number of objective evaluation (eval), the value of the objective at convergence (obj), the classification error rate on the test set (err).

of NRBMLS and SGD-2000 epochs, we see that NRBMLS outputs better objective values and test error rates, while requiring significantly less number of iteration. Furthermore, NRBMLS is constantly faster than NBM of the same family, and significantly outperforms SVMstruct too.

5.4 Transductive SVM Training: Non-Smooth and Non-Convex Optimization Problem

Finally, we consider the problem of binary classification in a semi supervised setting with a training set of n labeled examples $\{(x_1,y_1),..,(x_n,y_n)\}$ and m unlabeled examples $\{x_{n+1},..,x_{n+m}\}$. The unconstrained primal formulation of TSVM is then:

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\langle x_i, \mathbf{w}\rangle) + \gamma\frac{1}{m}\sum_{i=n+1}^{n+m}\max(0, 1 - |\langle x_i, \mathbf{w}\rangle|)$$

where γ is a trade-off parameter between the labeled loss term and the unlabeled loss term. This objective function belongs to the regularized function family that can be solved by NRBM. Furthermore it is also an instance of difference of convex functions and can be solved by CCCP.

We trained TSVM on a subset of the MNIST data set consisting of samples of digit 3 and 8, the data was preprocessed with Principal Components Analysis (PCA) as is usually done. We split

	NRBMLS				NBM				CCCP Primal			
γ	time	eval	obj.	err.	time	eval	obj.	err.	time	eval	obj.	err.
1	0.2s	24	0.26	0.057	0.8s	113	0.26	0.058	1.4s	181	0.26	0.057
2	0.1s	21	0.35	0.055	0.4s	59	0.35	0.054	1.6s	206	0.36	0.054
5	0.2s	25	0.55	0.051	1.1s	197	0.55	0.051	2.7s	375	0.57	0.057
10	0.3s	39	0.81	0.049	0.3s	57	0.81	0.051	4.8s	680	0.86	0.056
20	0.7s	90	1.20	0.048	0.5s	104	1.20	0.051	7.5s	1083	1.72	0.122
50	0.3s	41	2.19	0.084	1.0s	193	2.00	0.054	1.1s	151	9.38	0.548

	UniverSVM			SG 100 epochs			SG 500 epochs			SG 5000 epochs		
γ	time	obj.	err.	time	obj.	err.	time	obj.	err.	time	obj.	err.
1	64s	n.a.	0.057	1s	0.28	0.058	3s	0.26	0.057	33s	0.26	0.058
2	65s	n.a.	0.057	1s	0.37	0.055	3s	0.35	0.054	32s	0.35	0.053
5	53s	n.a.	0.049	1s	1.18	0.329	3s	0.61	0.051	32s	0.55	0.048
10	59s	n.a.	0.075	1s	1.81	0.455	3s	1.62	0.401	30s	1.60	0.390
20	63s	n.a.	0.055	1s	2.49	0.489	3s	2.37	0.476	27s	2.35	0.475
50	69s	n.a.	0.143	1s	4.09	0.501	3s	3.60	0.505	26s	3.48	0.503

Table 6: Comparative results of transductive SVM training. Each row corresponds to a particular value of γ. For each optimization method, we report the total cputime (time), the number of objective evaluation (eval), the value of the objective (obj), the classification error rate on the test set (err). The error rate of the initial solution (a SVM trained on labeled samples only, with $\lambda = 0.01$) is 0.0776.

the training data into 200 labeled samples and 11782 unlabeled samples, the test set contains 1984 samples. We set $\lambda = 0.01$ since it gives best results on the test data.

We initialized the TSVM with the SVM solution on the labeled set. Then we optimized the non-convex objective with NRBMLS, CCCP in primal (Yuille and Rangarajan, 2003), CCCP in dual with UniverSVM, and Gradient Descent.[10] For CCCP-Primal, we used our NRBM-convex solver (disabling the use of locality measures) for every iteration, then we stopped CCCP once the improvement of the objective function was not better than 1% of its current value. Note that UniverSVM solve the TSVM problem in dual space and there is no equivalent concept of the number of primal objective evaluations. We also perform experiments with Gradient Descend method since the implementation of SGD for TSVM is not trivial.

The comparative results of the 5 optimization methods are shown in Table 6. We observe that all methods achieve similar best error rates ($0.048 - 0.049$) but primal optimization methods outperform the only one dual optimization method (UniverSVM) in term of speed. Looking at each row (for the same γ value), we see that NRBMLS always reach a good solution which make it the most robust optimizer in this experiment.

The behavior of Subgradient Method (SG) is quite complex. On one hand, one observe that it seems to converge to the same solution of NRBMLS for small values of γ. On other hand, for

[10] The implementation of UniverSVM was downloaded from: `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/universvm.html`.

large γ (i.e. the non-convex term is more important), the solution found by SG is not good in term of objective value and test error rate. Although SG and NRBMLS reach the same best error rate (0.048), SG requires a larger number of iterations to converge. This suggests the potential of using SG for the TSVM problem, but we need a very careful design (learning rate, update rule, etc.) to overcome the slow convergence rate.

Comparing NRBMLS and CCCP-Primal, we see that NRBMLS clearly outperforms CCCP, it is faster and converges to a better solution. This can be explained by the fact that CCCP spends too much effort to minimize the convex approximation function (whose evaluation requires the same computing cost as the original non-convex function) every CCCP iteration. While convex approximation of non-convex function is often accurate only on a neighborhood of the coordinate where its was constructed, CCCP may perform useless computations to reach the minimum of the convex approximation. Instead, NRBMLS optimizes more directly the non-convex objective by updating iteratively the approximation. At the end, it converges faster to a good solution. Furthermore, CCCP may fail to get a good solution when the value of γ is too big (i.e. when the non-convex term becomes more important) while NRBM seems to be more robust in this hard setting.

At last it must be noticed that using CCCP may require a lot of effort to mathematically reformulate the objective function into a concave and a convex term. The extension of TSVM for multiclass classification and for structured prediction is for instance not straightforward with CCCP while it is with NRBM.

5.5 Large Margin Training for Continuous Density Hidden Markov Models

Hidden Markov Models (HMMs) have been widely used for automatic speech recognition (Rabiner, 1990) and handwriting recognition (Hu et al., 2000). Continuous Density HMMs (CDHMMs) are particularly suited for dealing with sequences of real-valued feature vectors that one gets after typical front-end processing in signal processing tasks. CDHMMs usually exploit Gaussian mixture models to describe the variability of observations in a state. HMM parameters are learnt on a partially labeled data set (since state sequences of training sequences are unknown) with the Expectation-Maximization algorithm (EM) to maximize the joint likelihood of observation sequences and of hidden state sequences.

Recently, few approaches have been proposed for large margin learning of HMMs, especially in the speech recognition community (Sha and Saul, 2007; Jiang and Li, 2007) (see Yu and Deng, 2007 for a review). However none of these works actually handle the whole problem of max-margin learning for HMM parameters in the standard partially labeled setting. For instance Sha and Saul (2007) and Jiang and Li (2007) tackle a simplified convex optimization problem. Indeed, the main difficulty one encounters when formulating the maximum margin learning of CDHMM lies in the non-convexity of the optimization problem which comes from the presence of hidden state variables (the sequence of states and of Gaussian components) and from the discriminative function which is quadratic with respect to some parameter (e.g., covariance matrix). Instead of relying on a convex relaxation technique, we proposed to directly optimize the non-smooth non-convex objective function (Do and Artières, 2009):

$$\min_{\mathbf{w}} \quad \tfrac{\lambda}{2}\|\mathbf{w}-\mathbf{w}_0\|^2 + \textstyle\sum_i \max_{\mathbf{y}} \left(F(\mathbf{x}^i,\mathbf{y},\mathbf{w}) + \Delta(\mathbf{y}^i,\mathbf{y}) - F(\mathbf{x}^i,\mathbf{y}^i,\mathbf{w})\right) \tag{21}$$

where $(\mathbf{x}^i,\mathbf{y}^i)$ are input and output sequences, $\mathbf{w}$ are model parameters, and discriminative functions $F(\mathbf{x}^i,\mathbf{y},\mathbf{w}) = \max_{\mathbf{s}\in S(\mathbf{y}),\mathbf{m}} \log p(\mathbf{x}^i,\mathbf{y},\mathbf{s},\mathbf{m}|\mathbf{w})$ are Viterbi-approximation of log likelihood ($\mathbf{s}$ and $\mathbf{m}$

	NRBMLS			SGD$^+$			SGD		
λ	eval	obj.	err.	eval	obj.	err.	eval	obj.	err.
0.010	72	0.583	0.281	72	0.598	0.286	439	0.594	0.285
0.005	104	0.542	0.276	104	0.612	0.281	502	0.552	0.280
0.003	132	0.507	0.274	132	0.653	0.278	594	0.517	0.274
0.002	396	0.474	0.277	396	0.765	0.277	667	0.487	0.275
	NBM$^+$			NBM*			NBM		
0.010	72	0.584	0.282	83	0.583	0.282	416	0.580	0.283
0.005	104	0.545	0.278	139	0.542	0.279	629	0.539	0.277
0.003	132	0.512	0.276	242	0.507	0.275	916	0.504	0.275
0.002	396	0.475	0.278	541	0.474	0.277	1181	0.472	0.278

Table 7: Large margin training for Continuous Density Hidden Markov Model. SGD and NBM results were reported with different various criteria. The $^+$ superscript corresponds to setting the maximum number of function evaluation to the one of NRBM. The * superscript corresponds to running optimization until the same objective function as NRBM solution was found.

are hidden state segmentation and mixture component assignments respectively). Note that the objective function is regularized with an initial solution $\mathbf{w}_0$ (MLE solution in practice) which yields better results than considering the standard regularization term $\frac{\lambda}{2}\|\mathbf{w}\|^2$.

Experiments were conducted on the TIMIT data set, which is a well-known benchmark data set for speech recognition. The training set consists of 3696 utterances, corresponding to 1,100,000 frames. We consider here the best CDHMM topology with 3 states per phoneme and 4 Gaussians per state, it corresponds to about 925,000 model parameters (Do and Artières, 2009).

We compare the optimization results of NRBMLS with SGD and NBM. Note that CCCP is not applicable since there is no concave-convex reformulation of the objective function in Equation 21 in the literature. Table 7 reports the number of function evaluation, the objective function value, and the phone recognition error rate for the three optimization methods (execution time is not reported since the experiments were launch on different machines). Here again, NRBM is stopped once the gap is below 1% of the objective value. For comparison, we reported SGD and NBM results with various stoping criteria: same number as NRBMLS (SGD$^+$ and NBM$^+$), reach NRBMLS's objective value (NBM*) and the native stop criteria of the optimizers (SGD stops once the (online estimated) objective value is not improved after 5 iterations or the maximum number of function evaluations (2000) is reached, NBM stops once the norm of the search direction is smaller than 10^{-4}).

Looking at final results of the 3 optimizers with their native stoping criteria, we found that NRBMLS and NBM seems to converge to the same local minima in all cases while SGD converges to different solutions. For example, in the case of $\lambda = 0.003$ (the optimum value), NRBMLS and NBM reach objective values 0.507 and 0.504 respectively, i.e, the relative difference is 0.6% (which is less than the stoping condition of NRBMLS). Also, while NBM reaches a more accurate solution it does not outperform NRBM form the recognition rate point of view (0.275 vs. 0.274). Based on the final objective values of SGD results, one could say that SGD is not a good optimizer, yet it also allows achieving good recognition rate. This observation confirms again that stochastic learning

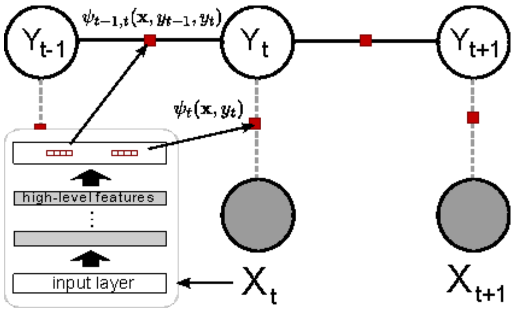


Figure 9: A chain-structured NeuroCRF.

may not always be good as an optimization tool but it may lead to solutions with good generalization properties.

At the end, NRBMLS converges significantly faster than the two other optimizers as it reaches a better solution (in term of objective value and error rate) when using a fixed number of iterations (columns SGD$^+$ and NBM$^+$), and it requires less iterations to reach a given value of the objective function (column NBM*, there is no column SGD* since SGD does not reach a similar objective value as NRBMLS).

5.6 Learning a Non Linear CRF

Finally, we consider the discriminative training of a NeuroCRF, this may be viewed as an extension of deep neural networks for structured output prediction (Do and Artières, 2010). Alternatively it may be thought as a hybrid model for labeling sequences that consists in a conditional random field exploiting features extracted with a deep neural network. Figure 9 illustrates such a linear chain NeuroCRF which combines a standard CRF with a deep neural network for feature extraction. Such a model implements a posterior distribution following

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_{c \in C} e^{\langle \mathbf{w}_c, \Phi(\mathbf{x}, \mathbf{y}_c, \mathbf{w}_{NN}) \rangle} \tag{22}$$

where C denote the set of cliques of the CRF, $\mathbf{w}_c$ the weights associated to clique c, $\mathbf{w}_{NN}$ the weights of the deep NN, and Φ a function that extract features from $\mathbf{x}$.

Learning involves optimizing jointly the NN weights $\mathbf{w}_{NN}$ and the CRF parameters $\mathbf{w}_c$. As it has been suggested in previous works (Hinton et al., 2006), the deep neural network needs to be pretrained through unsupervised learning (we used Restricted Boltzmann Machine (RBM) in our experiments). This initial solution is then fine-tuned by optimizing a regularized version of the conditional likelihood in Equation 22. Our experiments compare optimization algorithms for the fine tuning step. Indeed, the pretraining of RBMs cannot be done by greedy optimizer such as NRBM or LBFGS since the computation of the gradient is intractable. Hence in our experiments, we pretrained the deep NNs by cascading RBMs trained with the original Contrastive Divergence algorithm (Hinton et al., 2006). The pretrained deep neural network is used to build the initial NeuroCRF model, called $\mathbf{w}_0$ where the initial CRF weights are drawn at random. Then, the NeuroCRF is fine tuned on a labeled data set of n input-output sequence $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1..n}$ using the following criteria:

$$\min_{\mathbf{w}} \quad \tfrac{\lambda}{2}\|\mathbf{w} - \mathbf{w}_0\|^2 + \tfrac{1}{m}\textstyle\sum_{i=1..n} log(p(\mathbf{y}^i|\mathbf{x}^i;\mathbf{w})) \tag{23}$$

	NRBMLS			SGD$^+$			SGD		
λ	eval	obj.	err.	eval	obj.	err.	eval	obj.	err.
0.0010	64	1.847	0.050	64	1.604	0.059	2000	1.528	0.058
0.0005	81	1.203	0.047	81	1.190	0.056	2000	1.076	0.054
0.0003	116	0.842	0.046	116	0.941	0.053	2000	0.840	0.055
0.0002	116	0.616	0.045	116	0.775	0.054	2000	0.691	0.056
0.0001	130	0.365	0.050	130	0.557	0.055	2000	0.514	0.056
	NBM$^+$			NBM*			NBM		
0.0010	64	1.853	0.052	66	1.844	0.052	318	1.645	0.050
0.0005	81	1.233	0.046	94	1.202	0.048	305	1.075	0.048
0.0003	116	0.854	0.048	128	0.842	0.046	323	0.764	0.046
0.0002	116	0.660	0.047	175	0.616	0.046	238	0.594	0.045
0.0001	130	0.405	0.046	-	-	-	203	0.371	0.047
	LBFGS$^+$			LBFGS*			LBFGS		
0.0010	64	1.802	0.051	54	1.845	0.052	2095	1.591	0.048
0.0005	81	1.196	0.047	79	1.201	0.047	2104	1.010	0.047
0.0003	116	0.805	0.045	88	0.841	0.046	2119	0.696	0.049
0.0002	116	0.617	0.045	118	0.616	0.045	2131	0.511	0.052
0.0001	130	0.375	0.047	152	0.365	0.049	2135	0.300	0.053

Table 8: Optimization results for the discriminative training of NeuroCRFs. SGD, NBM and LBFGS results were reported with different various criteria. The $^+$ superscript corresponds to setting the maximum number of function evaluation to the one of NRBM. The * superscript corresponds to running optimization until the same objective function as NRBM solution was found.

where m is total the number of tokens in the data set and $p(\mathbf{y}^i|\mathbf{x}^i;\mathbf{w})$ is the conditional likelihood of the correct labeling, whose gradient can be computed by back propagation. Since we use sigmoid activation in hidden units of the deep neural network, the final objective function is non-convex with many local minima and plateaus. This is the reason why neural network optimization is difficult and the final solution is sensitive to initialization.

We evaluated the discriminative training of NeuroCRF on OCR data. Table 8 reports optimization results of NRBMLS and the other applicable state-of-the-art optimizers: SGD, NBM, and LBFGS. We use similar stopping condition setting as in previous sections. Looking at final error rates, we found that the final solutions of NRBMLS and NBM are better than the solutions obtained by other optimizers. The final objective values of these two algorithms are also quite close, suggesting that the two methods converge to similar local optimum. Note that NRBMLS is faster than NBM in all cases as NBM requires more iterations to reach the final objective value of NRBMLS (see column NBM*). Note also that the result of NBM* for $\lambda = 0.0001$ is not available, since NBM converged to a local minima with higher objective value than that of NRBMLS. Importantly, while NRBMLS can exploit the regularization term to converge quickly, a strong regularization term does not help NBM reducing the number of iterations. NRBMLS was also faster and reached better solution than SGD for small values of λ (including the best one $\lambda = 0.0002$). Unlike reported in previous section on training CDHMMs, SGD solutions have relative high error rates, indicating that the problem of local minima is maybe more severe in the case of deep neural networks.

Finally, the results of LBFGS$^+$ and LBFGS* show that NRBM and LBFGS have comparable convergence speed until the stoping criteria of NRBM were reached (at least for the studied range of λ). Actually, LBFGS is slightly faster than NRBMLS for large values of λ but it is slower than NRBMLS for small values of λ. While NRBM stopped with acceptable solutions (in term of recognition rate) after about one hundred iterations, LBFGS continued looking for better solution and did not converge after 2000 iterations (corresponding to more than 2000 function evaluation). The long run of LBFGS also leads to a significantly better objective function than that of NRBMLS. This reflects the fact that the final solutions of NRBMLS are local minima or belong to plateaus in which gradients is very close to zero. Interestingly as already reported in previous sections, a better objective value does not always mean a better recognition rate: the long run of LBFGS does not improve the recognition performance but leads to over fitting problems. At the end, LBFGS can be viewed as a good/greedy optimizer for NeuroCRF (in term of objective value) but it lacks of early stopping condition. While having comparable convergence speed as LBFGS, NRBMLS has an intuitive built-in stoping criteria (based on the gap) that appears to be relatively robust for many applications.

5.7 Summary of Results and Discussion

We conducted extensive objective evaluation of NRBM and NRBMLS on few artificial problems and on five standard machine learning problems. Our preliminary results on artificial optimization problems show the potential of our approach and highlight the benefit of the line-search procedure to the proposed method, which improves both the convergence speed and the quality of the solution. We then evaluated our method on machine learning problems with real data sets, showing that it can be applied to a large variety of machine learning frameworks including convex and non-convex problems, smooth and non-smooth objectives, to learn linear and non-linear model (e.g. artificial neural network). While being generic to regularized machine learning problem, NRBMLS also showed great performance compared to dedicated solvers such as SVMStruct (for M3N) or UniverSVM (for transductive SVM).

Comparison with competitive solvers. Since NRBMLS may be viewed as a variant of NBM (both use a linesearch), their differences in performance is much interesting. We found that NRBMLS and NBM often converge to the same solution (i.e., having similar objective value) but NBM generally requires more objective function evaluations than NRBMLS to reach a solution with similar objective value (e.g. NRBMLS 132 vs. NBM* 242 in Table 7). We also compared NRBMLS to LBFGS, a popular solver for smooth functions, and we showed that NRBMLS is slightly faster than LBFGS. While NRBMLS and LBFGS have competitive performances the main advantage of NRBMLS is its generic feature, it can be applied to non-smooth function as well. Finally, the experimental comparison with SGD shows that the implemented version (which has been used for training CRFs) is not efficient for the five considered problems. While SGD may be very fast for large data sets, its main weakness remains the need of carefully tuning of parameters (e.g., step size) for every problem. Note that all these generic optimizers have gradient-based stopping criterion which work well if one needs an accurate solution (when subgradient is close to null vector). However, in the case where one does not need a too accurate solution (for example, to avoid over fitting), the subgradient information is not a strong measure of the quality of the solution. Alternatively NRBM uses a gap-based stopping criterion which measures directly the quality of the solution in terms of the objective value. This criterion of NRBMs seems to be efficient for machine learning problems since NRBMLS systematically reaches the best error rates in all five machine learning experiments (using only one threshold of 0.1% of the objective value).

6. Conclusion

We proposed a new bundle optimization method called Non-convex Regularized Bundle Method (NRBM) able to deal with the minimization of regularized non-convex functions. We built on ideas from Convex Regularized Bundle Methods and on ideas from Non-Convex Bundle Methods, exploiting the regularization term of the objective and using a particular design of the aggregated cutting plane to build limited memory variants. We also discussed variants of the method and showed that integrating a line search may increase convergence rate in practice.

Experimental results on artificial problems show that our method is significantly faster than standard Non-convex Bundle Method, which is a state of the art method for non-smooth and non-convex optimization. We also presented experimental results on various convex and non-convex difficult machine learning problems, which demonstrate the potential and the wide application range of our algorithms. On one hand, our variant of bundle method got positive results on five different machine learning problems compared to state-of-the-art optimizers. On the other hand, our method is rather easy to use as the stopping condition is intuitive and efficient for various machine learning applications.

At the end, though the limited memory variant can be proved to inherit the fast convergence rate of CRBM in case of convex risks, we did not provide an analog satisfactory proof for the non-convex extension which is then more an algorithmic proposition that we validate experimentally on various machine learning optimization problems. Theoretical convergence analysis could be one direction for future work. We are also interested in considering more sophisticated approximation techniques such as second order approximation (similar to LBFGS) or non-convex approximation. While the approximation function could be more complex, one could expect that a more accurate approximation technique could improve the convergence speed.

References

A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil. A dc-programming algorithm for kernel selection. In *Proceedings of the Twenty-third International Conference on Machine Learning*, pages 41–48, 2006.

Y. Bengio and Y. Lecun. *Scaling Learning Algorithms Towards AI.* MIT Press, 2007.

D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, 2003.

L. Bottou. Stochastic gradient descent examples on toy problems, 2008. URL `http://leon.bottou.org/projects/sgd`.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the Twenty-third International Conference on Machine Learning*, pages 201–208. ACM Press, 2006.

T. M. T. Do and T. Artières. A fast method for training linear svm in the primal. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, pages 272–287, Berlin, Heidelberg, 2008. Springer-Verlag.

T. M. T. Do and T. Artières. Large margin training for hidden Markov models with partially observed states. In Léon Bottou and Michael Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning*, pages 265–272, Montreal, June 2009. Omnipress.

T. M. T. Do and T. Artières. Neural conditional random fields. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

T. M. T. Do and T. Artieres. Regularized bundle methods for convex and non convex risks: additional material to the jmlr paper. Technical report, LIP6, Pierre and Marie Curie University, Paris, 2012. URL `http://www.lip6.fr/lip6/reports/2012/lip6-2012-001.pdf`.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the Twenty-fifth International Conference on Machine learning*, pages 320–327, New York, NY, USA, 2008. ACM.

M. Gaudioso and M.F. Monaco. Variants to the cutting plane approach for convex nondifferentiable optimization. *Optimization 25*, pages 65–75, 1992.

M. Haarala, K. Miettinen, and M. M. Makela. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software*, 19:673–692(20), December 2004.

G. E. Hinton, S. Osindero, and Y-W Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

R. Horst and N. V. Thoai. Dc programming: overview. *Optimization Theory and Applications*, 103 (1):1–43, 1999. ISSN 0022-3239.

J. Hu, S. Gek Lim, and M.K. Brown. Writer independent on-line handwriting recognition using an hmm approach. *Pattern Recognition*, 33(1):133–147, 2000.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proceedings of the Twelfth International Conference on Computer Vision*. IEEE, 2009.

H. Jiang and X. Li. Incorporating training errors for large margin HMMs under semi-definite programming framework. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages IV–629. IEEE, 2007.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the Twelfth International Conference On Knowledge Discovery and Data Mining*, pages 217 – 226, 2006.

T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

R. H. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Cambridge, MA, USA, 1995.

K. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Springer-Verlag, 1985.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

L. Luksan and J. Vlcek. Introduction to nonsmooth analysis. theory and algorithms. Technical report, Universita degli Studi di Bergamo, 2000.

M. Makela. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17:1–29, 2002.

M. Makela and P. Neittaanmaki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., 1992.

L. R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2: 121–152, 1992.

F. Sha and L.K. Saul. Large margin hidden markov models for automatic speech recognition. *Advances in Neural Information Processing Systems*, 19:1249, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, pages 807–814, New York, USA, 2007. ACM Press.

A. Smola, S. V. N. Vishwanathan, and Q. V. Le. Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20:1377–1384, 2008.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. *Advances in Neural Information Processing Systems*, 16, 2004.

C. H. Teo, Q. V. Le, A. Smola, and S. V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the Thirteenth International Conference On Knowledge Discovery and Data Mining*, pages 727–736, New York, USA, 2007. ACM.

M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. *Advances in Neural Information Processing Systems*, 20.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. *Advances in Neural Information Processing Systems*, 17:1537–1544, 2004.

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *Proceedings of the Twenty-third international conference on Machine learning*, pages 1057–1064. ACM, 2006.

Z. Xu, R. Jin, J. Zhu, I. King, and M. Lyu. Efficient convex relaxation for transductive support vector machine. *Advances in Neural Information Processing Systems*, 20:1641–1648, 2008.

D. Yu and L. Deng. Large-margin discriminative training of hidden markov models for speech recognition. In *Proceedings of the International Conference on Semantic Computing*, pages 429–438, Washington, DC, USA, 2007. IEEE Computer Society.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *Proceeding of the Twenty-fifth International Conference on Machine Learning*, pages 1248–1255, Helsinki, Finland, 2008.