Batch and online learning algorithms for Nonconvex Neyman-Pearson classification

GILLES GASSO[(a)]
LITIS, INSA Rouen, France.
ARISTIDIS PAPPAIOANNOU[(b)]
MARINA SPIVAK[(c)]
LÉON BOTTOU[(d)]
NEC Labs America, Princeton.

We describe and evaluate two algorithms for Neyman-Pearson (NP) classification problem which has been recently shown to be of a particular importance for bipartite ranking problems. NP classification is a nonconvex problem involving a constraint on false negatives rate. We investigated batch algorithm based on DC programming and stochastic gradient method well suited for large scale datasets. Empirical evidences illustrate the potential of the proposed methods.

Categories and Subject Descriptors: I.5.2 [**Pattern Recognition**]: Classifier Design and evaluation

General Terms: Algorithms

Additional Key Words and Phrases: Neyman-Pearson, Nonconvex SVM, DC algorithm, online learning

1. INTRODUCTION

Consider a binary classification problem with patterns $\mathbf{x} \in \mathcal{X}$ and classes $y = \pm 1$ obeying an unknown probability distribution $dP(\mathbf{x}, y)$. The probabilities of non detection $\mathbf{P}_{\text{nd}}$ and of false alarm $\mathbf{P}_{\text{fa}}$ measure the two kinds of errors made by a discriminant function f :

$$\mathbf{P}_{\text{nd}}(f) = \mathbb{P}\big(f(\mathbf{x}) \leq 0 \,|\, y=1\big), \quad \mathbf{P}_{\text{fa}}(f) = \mathbb{P}\big(f(\mathbf{x}) \geq 0 \,|\, y=-1\big)$$

The statistical decision theory recognizes the need to associate different costs to these two types of errors. This leads us to searching a classifier f that minimizes the Asymmetric Cost (AC) formulation :

$$\min_f \; C_+ \, \mathbf{P}_{\text{nd}}(f) + C_- \, \mathbf{P}_{\text{fa}}(f) \, . \tag{1}$$

Although C_+ and C_- have a meaningful interpretation, it is often very difficult to specify these costs in real situations such as medical diagnosis or fraud detection. There are also cases where such costs have no meaningful interpretation, for

[(a)] Gilles Gasso, `gilles.gasso@insa-rouen.fr`.
[(b)] Aristidis Papaioannou, `aristidis.papaioannou@gmail.com`, was also affiliated with Ecole Polytechnique Fédérale de Lausanne, Switzerland. He is now with Google, Mountain View, CA.
[(c)] Marina Spivak, `spivak.marina@gmail.com`, was also affiliated with New York University. She is now in University of Washingtom, WA.
[(d)] Léon Bottou, `leon@bottou.org`, is now with Microsoft, Bellevue, WA.

instance, as discussed in section 4, when one uses a classification framework to approach a false discovery problem.

In contrast, the Neyman-Pearson (NP) formulation,

$$\min_f \ \mathbf{P}_{\mathrm{nd}}(f) \quad \text{subject to} \quad \mathbf{P}_{\mathrm{fa}}(f) \leq \rho \tag{2}$$

requires only the specification of the maximal false alarm rate ρ and can be meaningfully applied to false discovery problems.

It is well known that the optimal decision function for both problems are obtained by thresholding the optimal ranking function

$$r^*(\mathbf{x}) \ = \ \mathbb{P}\big(y = +1 \,|\, \mathbf{x}\big)\,, \tag{3}$$

that is

$$\begin{aligned} f^*_{\mathrm{AC}} \ &= \ r^*(\mathbf{x}) - C_-/(C_+ + C_-)\,, \\ f^*_{\mathrm{NP}} \ &= \ r^*(\mathbf{x}) - \min\{r \text{ such that } \mathbf{P}_{\mathrm{fa}}(r^* - r) \leq \rho\}\,. \end{aligned}$$

Although this result suggests equivalent capabilities, it misses several important points. Firstly, when using a finite training set $\mathcal{D} = \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$ we must work with the empirical counterparts of $\mathbf{P}_{\mathrm{nd}}$ and $\mathbf{P}_{\mathrm{fa}}$:

$$\tilde{\mathbf{P}}_{\mathrm{nd}}(f) = \frac{1}{n_+} \sum_{i \in \mathcal{D}_+} \mathbb{I}_{f(\mathbf{x}_i) \leq 0}\ , \quad \tilde{\mathbf{P}}_{\mathrm{fa}}(f) = \frac{1}{n_-} \sum_{i \in \mathcal{D}_-} \mathbb{I}_{f(\mathbf{x}_i) \geq 0}$$

where $\mathcal{D}_+$ and $\mathcal{D}_-$ represent respectively the set of positives and negatives with cardinality n_+ and n_-. We must also choose the decision function f within a restricted class $\mathcal{H}$ that is unlikely to contain the optimal decision function. This approach is supported by standard results in statistical learning theory [e.g. Vapnik 1998] and their extension to the Neyman-Pearson formulation [Scott and Nowak 2005].

Secondly, the empirical counterparts of problems (1) and (2) involve the 0–1 loss function $\mathbb{I}_{y\, f(\mathbf{x}) \leq \mathbf{0}}$ that is neither continuous nor convex. Replacing this 0–1 loss with the SVM Hinge loss has been studied for both the Asymmetric Cost [Bach et al. 2006] and Neyman-Pearson [Davenport et al. 2010] formulations. This substitution introduces additional complexities. In particular, in order to hit the specified goals on $\mathbf{P}_{\mathrm{nd}}$ and $\mathbf{P}_{\mathrm{fa}}$, one must use asymmetric costs that are different from C_+ and C_-. Both works eventually rely on hyperparameter searches in the asymmetric cost space.

An alternative approach consists in first learning a scoring function that orders input patterns like the optimal ranking function (3). Both problems (1) and (2) are then reduced to the determination of a suitable threshold [e.g. Cortes and Mohri 2004]. However it is quite difficult to ensure that the ranking function is most accurate in the threshold area. Theoretical investigations of this problem conclude that Neyman-Pearson classification remains an important primitive for such focussed ranking algorithms [Clémençon and Vayatis 2007; 2009].

This contribution proposes two practical and efficient algorithms for NP classification using nonconvex but continuous and mostly differentiable loss functions. In particular, these algorithms are shown to work using asymmetric costs that maintain a clear relation with the specified goals. The first algorithm leverages modern

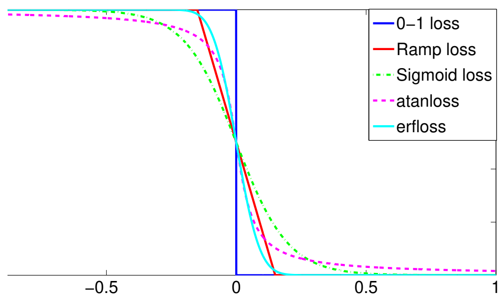

Fig. 1. Approximations of 0-1 loss. $\ell_{\text{atan}}(z) = 1/2 + \arctan(-\beta z)/\pi$ and $\ell_{\text{erf}}(z) = 1/2 + \text{erf}(-\beta z)/2$. Definitions of the other cost functions are detailed in the text.

nonconvex optimization techniques [Tao and An 1998]. The second algorithm is a stochastic gradient algorithm suitable for very large datasets. Various experimental results illustrate their properties.

2. EMPIRICAL RISK NP FORMULATION

Our approach consists in replacing the 0–1 loss in $\tilde{\mathbf{P}}_{\text{nd}}$ and $\tilde{\mathbf{P}}_{\text{fa}}$ by a continuous nonconvex approximation such as the sigmoid loss

$$\ell(z) = \frac{1}{1+e^{\eta z}} \tag{4}$$

or the ramp loss

$$\ell(z) = \max\left\{0,\ \frac{1}{2\eta}(\eta - z)\right\} - \max\left\{0,\ -\frac{1}{2\eta}(\eta + z)\right\}. \tag{5}$$

The positive parameter η determines how close the nonconvex costs are to the 0–1 loss and those approximations tend toward the 0–1 loss as η tends to 0. Figure 1 illustrates such approximations. The selection of an optimization algorithm usually dictates the choice of an approximation. The differentiable sigmoid loss lends itself to gradient descent, whereas the ramp loss is attractive with dual optimization algorithms.

Following common practice, we also add a regularization term $\Omega(f)$ to control the capacity of our classifiers. We therefore seek the solution of

$$\min_{f\in\mathcal{H}} \ \Omega(f) + C\,\hat{\mathbf{P}}_{\text{nd}}(f) \quad \text{subject to} \quad \hat{\mathbf{P}}_{\text{fa}}(f) \le \rho \tag{6}$$

where $C \in \mathbb{R}_+$ is the regularization parameter and

$$\hat{\mathbf{P}}_{\text{nd}}(f) = \frac{1}{n_+}\sum_{i\in\mathcal{D}_+} \ell\big(y_i f(\mathbf{x}_i)\big)\,, \quad \hat{\mathbf{P}}_{\text{fa}}(f) = \frac{1}{n_-}\sum_{i\in\mathcal{D}_-} \ell\big(y_i f(\mathbf{x}_i)\big)\,.$$

For instance, in the case of a Neyman-Pearson SVM (NP-SVM), the discriminant functions is $f(\mathbf{x}) = f_0(\mathbf{x}) + b$ with f_0 taken from a RKHS induced by a kernel $\mathsf{k}(\mathbf{x}, \mathbf{x}')$, and the regularizer is $\Omega(f) = \|f_0\|^2_{\mathcal{H}}$.

The nonconvex optimization problem (6) comes with the usual caveats and benefits. We can only obtain a local minimum of (6). On the other hand, we can obtain a local minimum that is better than the solution of any convex relaxation of (6), simply by initializing the nonconvex search using the solution of the convex relaxation.

2.1 Previous work

The NP classification problem has been extensively studied. Past methods can be roughly divided in two categories: generative and discriminative.

One of the earliest attempts [Streit 1990] uses multi-layered neural network to estimate class-conditional distributions as mixture of Gaussians. The discriminant function is then inferred with a likelihood ratio test. In the same vein, recent methods [Kim et al. 2006] assume the class-conditional distributions are Gaussian with means $\boldsymbol{\mu}_\pm$ and covariances $\boldsymbol{\Sigma}_\pm$, and consider a linear classifier $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. This amounts to solving (2) with the following definitions

$$\hat{\mathbf{P}}_{\text{nd}} = \Phi\left(-\frac{b + \mathbf{w}^\top \hat{\boldsymbol{\mu}}_+}{\sqrt{\mathbf{w}^\top \hat{\boldsymbol{\Sigma}}_+ \mathbf{w}}}\right), \; \hat{\mathbf{P}}_{\text{fa}} = \Phi\left(\frac{b + \mathbf{w}^\top \hat{\boldsymbol{\mu}}_-}{\sqrt{\mathbf{w}^\top \hat{\boldsymbol{\Sigma}}_- \mathbf{w}}}\right),$$

where Φ is the cumulative distribution function of standard normal distribution and $\hat{\boldsymbol{\mu}}_\pm$ and $\hat{\boldsymbol{\Sigma}}_\pm$ are empirical estimations. Since the Gaussian assumption proves too restrictive, some authors [Huang et al. 2006; Kim et al. 2006] replace the cumulative Φ by a Chebyshev bound $\Psi(u) = [u]_+^2/(1 + [u]_+^2)$, $[u]_+ = \max(0, u)$. The scheme is extended to nonlinear discrimination using the kernel trick. A third flavor of generative approach addresses the estimation of class-condition distributions by Parzen window [Bounsiar et al. 2008]. These generative methods share the same drawbacks: (1) the final classifiers are derived from estimated distributions whose accuracy is questionable when the datasets are small, and (2) the kernel version of these models lacks sparsity because all the examples are involved in the model.

On the discriminative side, the Asymmetric Cost SVM [Bach et al. 2006; Davenport et al. 2010]. introduces costs C_+ and C_- in the SVM formulation. As mentioned before, even if the true asymmetric costs were known, the benefits of the convex loss, such as the guaranteed convergence to global optimum, are balanced by the necessity of searching for different asymmetric costs to achieve the desired NP constraint [Bach et al. 2006]. SVMPerf [Joachims 2005] optimizes in polynomial time a convex upper bound of any performance measures computable from the confusion table. Since $\tilde{\mathbf{P}}_{\text{fa}}$ and $\tilde{\mathbf{P}}_{\text{nd}}$ can be computed from the confusion table, SVMPerf can address the Neyman-Pearson problem. Computing times grow very quickly with the number of examples n, typically with degree four for linear models, worse for nonlinear models. Finally, most similar to our approach, [Mozer et al. 2002] also consider sigmoid approximation of 0-1 loss. Compared to this work, our contributions are three-fold: we extend the nonconvex NP idea to SVM in order to benefit from off-the-shelf SVM solvers, we propose a stochastic approach to deal with large scale datasets, and we extend the potential of our approaches to related problems such as q-value optimization (section 4).

Algorithm 1 Uzawa Algorithm

Set initial value for $\lambda \geq 0$. Pick small gain $\nu > 0$.
repeat
  $f \leftarrow \operatorname{argmin}_{f\in\mathcal{H}} \mathcal{L}(f,\lambda)$
  $\lambda \leftarrow \max\{\, 0, \lambda + \nu\, \nabla_\lambda \mathcal{L}(f,\lambda)\,\}$
until convergence

3. SOLVING NON-CONVEX NP PROBLEMS

The algorithms discussed in this paper find a local minimum of (6) by searching a local saddle point $(f,\lambda) \in \mathcal{H}\times\mathbb{R}_+$ of the related Lagrangian

$$\mathcal{L}(f,\lambda) = \Omega(f) + C\,\hat{\mathbf{P}}_{\mathrm{nd}}(f) + \lambda\,(\hat{\mathbf{P}}_{\mathrm{fa}}(f) - \rho)\,. \tag{7}$$

The appendix summarizes several results that apply to nonconvex optimization. Local saddle points of the Lagrangian (7) are always feasible local minima of (6). Conversely, assuming differentiability, the local minima of (6) are always critical points of the Lagrangian.

3.1 Uzawa algorithm

The Uzawa algorithm [Arrow et al. 1958] is a simple iterative procedure for finding a saddle point of the Lagrangian (7). Each iteration of the algorithm first computes a minimum f^*_λ of the Lagrangian for the current value of λ, and then performs a small gradient ascent step in λ (algorithm 1) with $\nabla_\lambda\mathcal{L} = \hat{\mathbf{P}}_{\mathrm{fa}} - \rho$. The convergence of the Uzawa algorithm is not obvious because the function $\lambda \mapsto \mathcal{L}(f^*_\lambda, \lambda)$ can easily contain discontinuities. However a simple argument (see theorem 3 in the appendix) shows that $\hat{\mathbf{P}}_{\mathrm{fa}}(f^*_\lambda)$ is a nonincreasing function of λ. Therefore the sign of the gradient $\nabla_\lambda\mathcal{L} = \hat{\mathbf{P}}_{\mathrm{fa}} - \rho$ correctly indicates whether λ is above or below its target value. In general we prefer using a multiplicative update $\lambda \leftarrow \lambda(1 + \nu\,\nabla_\lambda\mathcal{L})$ because it keeps the λ positive. This makes very little difference in practice: the key is to adjust λ in very small increments, for instance using a very small gain ν.

The two algorithms discussed in this paper are essentially derived from the Uzawa algorithm. They differ in the minimization step. The first algorithm uses the DC[1] approach [Tao and An 1998] and is suitable for kernel machines. The second algorithm relies on a stochastic gradient approach [Tsypkin 1971; Andrieu et al. 2007] suitable for processing large datasets.

3.2 Batch learning of NP-SVM

The most difficult step in the Uzawa algorithm is minimization of $\mathcal{L}$ over f for λ fixed. In the case of the SVM classifier, the Lagrangian (7) reads

$$\mathcal{L} = \frac{1}{2}\|f\|^2_{\mathcal{H}} + C_+ \sum_{i\in\mathcal{D}_+} \ell(y_i f(\mathbf{x}_i)) + C_- \sum_{i\in\mathcal{D}_-} \ell(y_i f(\mathbf{x}_i)) - \lambda\rho$$

[1] The acronym DC stands for "Difference of Convex functions".

where $C_+ = C/n_+$, $C_- = \lambda/n_-$ and $\ell(z)$ stands for the ramp loss (5). This amounts to solving a nonconvex asymmetric cost SVM [Collobert et al. 2006b]. Since the analytical expression of the ramp loss (5) is a difference $\ell_1(z) - \ell_2(z)$ of two convex functions, the full Lagrangian can also be expressed as the difference of two convex functions amenable to DC programming [Tao and An 1998]. Consider the nonconvex optimization problem $\min_\theta J_1(\theta) - J_2(\theta)$ where both J_1 and J_2 are convex functions. Each DC iteration solves the convex problem obtained by linearizing J_2, that is,

$$\theta_{t+1} = \arg\min_{\theta} \{\, J_1(\theta) - \langle \nabla_\theta J_2(\theta_t), \theta \rangle \,\} \ , \tag{8}$$

where $\nabla_\theta J_2(\theta_t)$ denotes a subgradient of J_2. One can easily see that the cost $J_1 - J_2$ decreases after each iteration by summing the following two inequalities resulting from (8) and from the convexity of J_2.

$$\begin{aligned} J_1(\theta_{t+1}) + \langle \nabla_\theta J_2(\theta_t), \theta_{t+1} \rangle \ &\leq \ J_1(\theta_t) + \langle \nabla_\theta J_2(\theta_t), \theta_t \rangle \\ J_2(\theta_{t+1}) \ &\leq \ J_2(\theta_t) + \langle \nabla_\theta J_2(\theta_t), \theta_{t+1} - \theta_t \rangle \end{aligned}$$

Following [Collobert et al. 2006b], we write $\mathcal{L}(f, \lambda) = J_1(f) - J_2(f)$ with

$$\begin{aligned} J_1(f) \ &= \ \frac{1}{2}\|f\|^2_{\mathcal{H}} + \sum_i C_{y_i}\, \ell_1(y_i f(\mathbf{x}_i))\,, \\ J_2(f) \ &= \ \sum_i C_{y_i}\, \ell_2(y_i f(\mathbf{x}_i))\,, \end{aligned}$$

where notation C_{y_i} denotes either C_+ or C_-. Observe that $J_1(f)$ is the primal objective function of a standard convex SVM. Each DC iteration then reduces to solving the following convex problem

$$f_{t+1} = \underset{f}{\operatorname{argmin}} \left\{ J_1(f) + \frac{1}{2\eta} \sum_i C_{y_i} \beta_i f(\mathbf{x}_i) \right\} \tag{9}$$

where $\beta_i = \mathbb{I}\big(y_i f_t(\mathbf{x}_i) < -\eta\big)$ indicates whether $\ell_2(y_i f_t(\mathbf{x}_i))$ has a nonzero gradient. Standard SVM techniques can be used to obtain the dual formulation of this problem. This dual formulation turns out to be similar to that of a standard SVM problem with shifted box constraints [Collobert et al. 2006a, appendix]. This derivation leads to algorithm 2. To summarize, applying the Uzawa algorithm (algorithm 1) to the NP-SVM results in repeatedly solving a nonconvex asymmetric cost SVM via algorithm 2 and updating λ by gradient ascent.

This slow procedure requires solving many times a nonconvex SVM problem. Much faster convergence is observed by moving the gradient ascent step inside the DC iterations, leading to the *Annealed NonConvex NP-SVM algorithm* (algorithm 3). This algorithm departs from the Uzawa template because the Lagrange coefficient λ is updated after each iteration of the DC algorithm (9) instead of performing it in an additional loop surrounding the optimization of the nonconvex asymmetric cost SVM. Although we have no formal convergence guarantee for this algorithm, empirical evidence shows that it works as reliably and much faster than the plain Uzawa algorithm (see section 5.2.)

We pick an initial value for λ such that $C/n_+ = \lambda/n_-$ in order to balance the initial regularization parameters for positives and negatives. Since the DC

Algorithm 2 Asymmetric Nonconvex SVM with DC

Set $\boldsymbol{\beta} \leftarrow \mathbf{0}$.

repeat

- Compute $\boldsymbol{\alpha} \leftarrow \underset{\boldsymbol{\alpha}}{\operatorname{argmax}}\ -\frac{1}{2}\boldsymbol{\alpha}^\top \tilde{\mathbf{K}}\,\boldsymbol{\alpha} + \eta\boldsymbol{\alpha}^\top \mathbf{1}$
  where $\tilde{K}_{ij} = y_i y_j \mathsf{k}\big(\mathbf{x}_i, \mathbf{x}_j\big)$, subject to $\begin{cases} \mathbf{y}^\top \boldsymbol{\alpha} = 0 \\ -\beta_i C_+ \le 2\eta\alpha_i \le (1-\beta_i)C_+ & \forall i \in \mathcal{D}_+ \\ -\beta_i C_- \le 2\eta\alpha_i \le (1-\beta_i)C_- & \forall i \in \mathcal{D}_- \end{cases}.$
- Form $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathsf{k}\big(\mathbf{x}, \mathbf{x}_i\big)\ +\ b$ with b such that $y_i f(\mathbf{x}_i) = 1$ for all i such that the inequality constraints above were not reached.
- Compute $\beta_i \leftarrow \begin{cases} 1 & \text{if } y_i f(\mathbf{x}_i) < -\eta \\ 0 & \text{otherwise} \end{cases}$

until convergence of $\boldsymbol{\beta}$.

Algorithm 3 Annealed Nonconvex NP-SVM

Set $\boldsymbol{\beta} \leftarrow \mathbf{0}$ and $\lambda \leftarrow Cn_-/n_+$

repeat

- Set $C_+ = C/n_+$ and $C_- = \lambda/n_-$.
- Compute $\boldsymbol{\alpha} \leftarrow \underset{\boldsymbol{\alpha}}{\operatorname{argmax}}\ -\frac{1}{2}\boldsymbol{\alpha}^\top \tilde{\mathbf{K}}\,\boldsymbol{\alpha} + \eta\boldsymbol{\alpha}^\top \mathbf{1}$
  where $\tilde{K}_{ij} = y_i y_j \mathsf{k}\big(\mathbf{x}_i, \mathbf{x}_j\big)$, subject to $\begin{cases} \mathbf{y}^\top \boldsymbol{\alpha} = 0 \\ -\beta_i C_+ \le 2\eta\alpha_i \le (1-\beta_i)C_+ & \forall i \in \mathcal{D}_+ \\ -\beta_i C_- \le 2\eta\alpha_i \le (1-\beta_i)C_- & \forall i \in \mathcal{D}_- \end{cases}.$
- Form $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathsf{k}\big(\mathbf{x}, \mathbf{x}_i\big)\ +\ b$ with b such that $y_i f(\mathbf{x}_i) = 1$ for all i such that the inequality constraints above were not reached.
- Compute $\beta_i \leftarrow \begin{cases} 1 & \text{if } y_i f(\mathbf{x}_i) < -\eta \\ 0 & \text{otherwise} \end{cases}$
- Update $\lambda \leftarrow \lambda(1 + \nu(\hat{\mathbf{P}}_{\text{fa}} - \rho))$

until convergence.

iterations start with $\boldsymbol{\beta} = \mathbf{0}$, the first iteration solves the classical asymmetric cost SVM problem whose solution is progressively improved by taking the nonconvexity into account and updating λ in order to achieve the target $\mathbf{P}_{\text{fa}}$. The algorithm is stopped when the NP constraint is satisfied with a predefined precision ε.

3.3 Stochastic learning for the NP problem

The computational cost of the batch learning algorithm discussed above increases quickly when the dataset size becomes large. Faster algorithms are therefore desirable.

Algorithm 4 Stochastic NP algorithm

Initialize λ, $\mathbf{w}$, b.
repeat
Pick a random training example $(\mathbf{x}_t, y_t)$
Update $\mathbf{w}$ and b in the following ways

$$\mathbf{w} \leftarrow (1-\gamma_t\lambda_c)\mathbf{w} - \gamma_t a_t \nabla_{\mathbf{w}}\ell(y_t f(\mathbf{x}_t)) \tag{11a}$$

$$b \leftarrow b - \gamma_t a_t \nabla_b \ell(y_t f(\mathbf{x}_t)) \tag{11b}$$

If $y_t = -1$, set

$$\lambda \leftarrow \lambda\left(1 + \nu_t\left(\ell(-f(\mathbf{x}_t)) - \rho\right)\right) \tag{12}$$

until convergence

Let vector $\mathbf{w}$ and the scalar bias b be the parameters of the discriminant function, $\lambda_c = 1/C$, and reformulate problem (6) as

$$\min_f \frac{\lambda_c}{2}\|\mathbf{w}\|^2 + \hat{\mathbf{P}}_{\mathrm{nd}}(f) \quad \text{subject to} \quad \hat{\mathbf{P}}_{\mathrm{fa}}(f) \leq \rho\,.$$

Expanding $\hat{\mathbf{P}}_{\mathrm{nd}}$ and $\hat{\mathbf{P}}_{\mathrm{fa}}$ leads to the Lagrangian

$$\mathcal{L}(f,\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda_c}{2}\|\mathbf{w}\|^2 + a_i\,\ell(y_i f(\mathbf{x}_i)) - \lambda\rho\right)$$

with the coefficients

$$a_i = \begin{cases} n/n_+ & \forall i \in \mathcal{D}_+ \\ \lambda n/n_- & \forall i \in \mathcal{D}_- \end{cases} \tag{10}$$

Algorithm 4 is a stochastic variant of the Uzawa algorithm. Each iteration performs a stochastic gradient descent step for $(\mathbf{w}, b)$ and a stochastic ascent step for the Lagrange coefficient λ. Since the false alarm rate depends only on the negative examples, the update of λ only happens when sampling a negative example.

The convergence analysis of this stochastic descent/ascent procedure establishes how closely the updates (11-12) follow the trajectory described by the corresponding ordinary differential equation. Almost sure convergence to a local saddle point is guaranteed under mild conditions [Tsypkin 1971; Andrieu et al. 2007]. Such direct results are in fact stronger than those available for the generic Uzawa approach.

Linear NP-SVM. Following [Bottou 2007], when training a linear SVM (that is $f(\mathbf{x}) = \langle\mathbf{w},\mathbf{x}\rangle + b$), we choose a learning rate $\gamma_t = \gamma_0(1+\lambda_c\,t)^{-1}$ with an initial learning rate γ_0 chosen to ensure that the initial updates of $\mathbf{w}$ are compatible with its expected size. Choosing $\gamma_0 = 0.1$ works well when $\|\mathbf{x}_i\| \approx 1$.

Empirical evidence, on the other hand, suggests that ν should be smaller than γ_t in order to ensure that λ remain consistent. Otherwise the algorithm strongly enforces the false alarm rate constraint at the expense of the non detection rate. In our experimentations we always choose ν proportional to $1/n_-$.

Using the ramp loss makes the Lagrangian nondifferentiable when $y_t f(\mathbf{x}_t) = \pm\eta$. In such cases, we simply consider that the gradient is zero and skip the stochastic update. On the other hand, using the sigmoid loss simply avoids the problem. Both approaches work well in practice.

Nonlinear extensions. Algorithm 4 extends naturally to nonlinear discriminant functions such as multilayer neural network. Gradients are easily computed using the standard back-propagation algorithm. The learning rate γ_t must then be chosen using a trial and error approach.

Extending algorithm 4 to kernel representations is more complicated. The parameter $\mathbf{w}$ can be expressed with a kernel expansion $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x})$ where $\phi(\mathbf{x})$ is the mapping function associated with the kernel. Since $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$ we can plug the expression of the gradient $\nabla_{\mathbf{w}} \ell(y_t f(\mathbf{x}_t)) = -y_t \phi(\mathbf{x}_t) \ell'(y_t f(\mathbf{x}_t))$ into (11a), leading to the update

$$\alpha_i = (1 - \gamma_t \lambda_c)\, \alpha_i \; + \begin{cases} \gamma_t y_t \ell'(y_t f(\mathbf{x}_t)) & \text{if} \quad t = i \\ 0 & \text{otherwise} \end{cases}$$

However the noise introduced by the stochastic procedure compromises the sparsity of $\boldsymbol{\alpha}$. The computation of the gradient then requires numerous kernel evaluations in order to compute $f(\mathbf{x}_t)$. Although kernel values can be cached, these problems reduce the appeal of the stochastic approach. We plan to investigate this issues more closely in a forthcoming work.

Imbalanced Dataset. When the dataset is highly imbalanced, for instance, when there are many negatives and just a few positives, considerable speedups are achieved by picking random training examples from each class with equal probability. Since this procedure oversamples the examples from the minority class, we must adjust the coefficients a_i in algorithm 4 in proportion. The coefficients then become

$$a_i = \begin{cases} 1 & \forall i \in \mathcal{D}_+ \\ \lambda & \forall i \in \mathcal{D}_- \end{cases} .$$

4. Q-VALUE OPTIMIZATION

So far we have considered problems in fully supervised setting, with positive and negative labels assigned with equal confidence. The situation explored in this section is slightly different in the sense that we assume the labels of the negative class are confidently assigned and the goal of the classifier is to assign a confidence measure to labels of the first class.

Such problems are often encountered in large-scale biological screening, where multiple independent events are tested against a null hypothesis to discriminate real biological effects from noisy data. In microarray experiments, the expression level of genes is tested against the background signal [Storey and Tibshirani 2003]. In mass spectrometry protein screening, potentially positive matches between spectra and peptide sequences are tested for correctness against peptide-spectrum matches that are known to be negative because they were generated randomly [Elias and Gygi 2007].

Given a scoring function for these events, a widely used approach, both in microarray and proteomic applications, is to assign to each event a statistical confidence measure, the *q-value*, defined as the proportion of higher scoring events that turn out to be known negatives. Events below a certain user-specified q-value threshold are considered statistically significant and are selected for further biological identification. In practical terms, a q-value threshold can be interpreted as the proportion of significant events that will turn out to be false leads [Storey and Tibshirani 2003].

When such problems are addressed in classification framework, the goal is to discriminate the significant events from the nonsignificant ones. In this setup, the labels of the negatives are confidently assigned, while the labels of the positives are to be verified. In order to hit the user specified q-value threshold q, the classifier must satisfy the constraint

$$q(1 - \mathbf{P}_{\rm nd}) = \mathbf{P}_{\rm fa}.$$

Earlier works simply train a classifier using the available labels, without taking into account the uncertain nature of the positive labels. Then, they use the ordering induced by the classifier to assign q-values, and, finally, select a subset of examples with q-values below a desired threshold [Käll et al. 2007]. This approach is equivalent to constructing an ROC curve of the classifier and taking a single point on the ROC curve.

Alternatively, Spivak et al. [2009] propose an algorithm that directly optimizes the classifier for a single specified q-value threshold or a set of such thresholds. Their approach is equivalent to optimizing the ROC curve in the vicinity of the point corresponding to the q-value threshold q. A precise formulation of this objective leads to a formulation that is strikingly similar to NP classification. Indeed, maximizing true positives rate while keeping the user-specified q-value amounts to solving

$$\min_{f\in\mathcal{H}} \Omega(f) + C\,\hat{\mathbf{P}}_{\rm nd} \;\text{ subject to }\; \hat{\mathbf{P}}_{\rm fa} \leq q(1 - \hat{\mathbf{P}}_{\rm nd}).$$

The machinery of the section 3 is easily adaptable with the new Lagrangian

$$\mathcal{L}(f,\lambda) = \Omega(f) + (C + \lambda q)\hat{\mathbf{P}}_{\rm nd} + \lambda(\hat{\mathbf{P}}_{\rm fa} - q)\,. \tag{13}$$

This Lagrangian is obviously related to (7) and therefore is amenable to similar algorithms. Adapting the stochastic algorithm requires some ingenuity because the λ update depends on both the positive and negative examples. We have modified algorithm 4 to pick at each iteration both a random positive example $\mathbf{x}_t^+$ and a random negative example $\mathbf{x}_t^-$. We update the model parameters $\mathbf{w}$ using the gradient of the Lagrangian (13) restricted to these two randomly picked examples. Finally we update the Lagrange parameter λ with

$$\lambda \leftarrow \lambda\left(1 + \nu_t\ \left(\ \ell(-f(\mathbf{x}_t^-)) - q\left[\,1 - \ell(f(+\mathbf{x}_t^+))\,\right]\,\right)\,\right)\,.$$

5. EXPERIMENTS

This section reports experimental results obtained with the various algorithms discussed in this paper. Subsection 5.1 presents the various datasets used in our experiments. Subsections 5.2 and 5.3 present exploratory results characterizing the performance of the Annealed NonConvex NP-SVM algorithm. Subsections 5.4 and 5.5 present Neyman-Pearson classification results obtained using Gaussian Kernel SVMs on small-scale and medium-scale datasets and Linear SVMs on medium-scale and large-scale datasets. Finally, subsection 5.6 reports on a real life q-value optimization problem using both a Gaussian Kernel SVM and a nonlinear multilayer network as classifiers (subsection 5.6.)

5.1 Datasets

Table I summarizes the main characteristics of our datasets.

Some exploratory results reported in sections 5.2 were obtained using synthetic data inspired from [Bach et al. 2006]. The dataset consists of 2500 two-dimensional points. An equal number of of positives and negatives points are drawn from two gaussian distributions shown in figure 2.

Table I. Dataset summary

Dataset	Sections	Features	n_+	n_-
Synthetic	5.2	2	1250	1250
Breast	5.4	30	357	212
Pima	5.4	8	500	268
Spambase	5.2,5.3,5.4,5.5	57	2788	1813
Pageblocks	5.4, 5.5	10	4913	560
GammaTelescope	5.4, 5.5	10	12332	6688
Covertype	5.5	54	211840	20510
RCV1-V2	5.5	47152	684494	119920
Proteomics	5.6	17	69705	69705

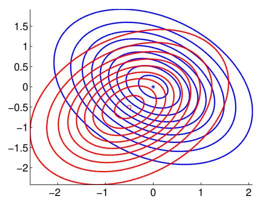

Fig. 2. Synthetic data for section 5.2.

The Neyman-Pearson classification experiments were carried out using two small scale datatsets (Breast and Pima), three medium scale datasets (Spambase, Pageblocks, and GammaTelescope) and two large scale datasets (Covertype and RCV1-V2). The first six datasets are well known classification datasets available from the UCI machine learning repository[2]. The small-scale and medium scale datasets originally specify two classes. We always use the majority class as positive examples and the minority class as negative examples. For the datasets Breast and Pima we set aside one quarter of the examples as a testing set and we use 4-fold cross-validation on the remaining examples for model selection. For the datasets

[2] `http://archive.ics.uci.edu/ml/`

Spambase, Pageblocks, GammaTelescope, and Covertype, we set aside one quarter of the examples as a testing set and one quarter of the examples as a validation set for model selection. The remaining half of the examples is used as the training set. In the case of the Covertype dataset, a binary classification problem was formed by picking species Spruce-Fir and Krummholz as positive and negative classes respectively. Finally, the dataset RCV1-V2 is available from the RCV1-V2 website[3]. In order to make the problem large scale, we use the official testing set (781,265 documents) as the training set and the official training set (23,149 documents) as the testing set. The RCV1-V2 TF/IDF features were recomputed to account for this new split. The documents associated with the category ECAT are used as negative examples. All the remaining documents are used as positive examples.

The q-value optimization experiments were carried out using a proteomics dataset consisting of 139410 samples with positive and negative samples equally represented [Spivak et al. 2009].

5.2 Speedups achieved by the Annealed NonConvex NP-SVM algorithm

This section compares the performance of the Annealed NonConvex NP-SVM (algorithm 3) with the performance of a more direct implementation of the Uzawa algorithm consisting of using algorithm 2 as the minimization step of the Uzawa algorithm 1.

Our first experiments trains a Gaussian Kernel SVM on the Synthetic dataset. We formed a set of 20 pairs $(C_+, \sigma) \in [10^{-2}, 10^{-2}] \times [10^{-2}, 10^{-2}]$ where σ is the bandwidth of the kernel and C_+ the cost assigned to positive samples. Both algorithms were tested with these hyperparameters for values of ρ in $\{0.01, 0.05, 0.1, 0.2\}$. The algorithms were implemented in C++ using a SMO optimizer [Platt 1999] and a kernel cache. The cache vastly speeds up the algorithms because kernel matrix coefficients computed during the earlier iterations can be used during the later iterations.

Table II reports results averaged on all values of the hyperparameters (C_+, σ). The Annealed NonConvex NP-SVM algorithm considerably accelerates the convergence. The table also shows the $\mathbf{P}_{\text{fa}}$ and $\mathbf{P}_{\text{nd}}$ differences evaluated by directly sampling test examples from the Gaussian distributions. These differences are statistically insignificant according to a Wilcoxson signed rank test with p-value threshold 10^{-3}.

Table III reports the speedup factors achieved by repeating this experiment on the medium size dataset Spambase. We do not report on the final $\mathbf{P}_{\text{fa}}$ and $\mathbf{P}_{\text{nd}}$ because the differences were similarly insignificant.

Both tables reveal that the Annealed NonConvex N-SVM speedup is higher for small values ρ. This probably happens because such values of ρ lead to extreme ratios between the quantities C_+ and $C_- = \lambda/n_-$ and penalize the computing times of the Asymmetric NonConvex SVM solver called during the minimization step of the Uzawa algorithm.

[3] `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

Table II. Comparison of the Annealed NonConvex NP-SVM solver (algorithm 3) with the pure Uzawa approach (algorithms 1 and 2) on the SYNTHETIC dataset. The second column reports the speedup factor. The remaining columns report the differences in final $\mathbf{P}_{\text{fa}}$ and $\mathbf{P}_{\text{nd}}$ test errors. These differences are not statistically significant.

ρ	SPEEDUP FACTOR	$\mathbf{P}_{\text{FA}}^{\text{ANNEALEDDC}} - \mathbf{P}_{\text{FA}}^{\text{FULLUZAWA}}$	$\mathbf{P}_{\text{ND}}^{\text{ANNEALEDDC}} - \mathbf{P}_{\text{ND}}^{\text{FULLUZAWA}}$
0.01	3.96 ± 2.74	$(20 \pm 71) \times 10^{-5}$	$(-3.6 \pm 31.5) \times 10^{-4}$
0.05	3.86 ± 3.30	$(37 \pm 88) \times 10^{-5}$	$(-7.3 \pm 18.2) \times 10^{-4}$
0.1	2.32 ± 1.51	$(24 \pm 90) \times 10^{-5}$	$(-7.4 \pm 24.7) \times 10^{-4}$
0.2	2.71 ± 1.78	$(36 \pm 72) \times 10^{-5}$	$(-6.5 \pm 14.8) \times 10^{-4}$

Table III. Comparison of the Annealed NonConvex NP-SVM solver (algorithm 3) with the pure Uzawa approach (algorithms 1 and 2) on the SPAMBASE dataset.

ρ	0.01	0.05	0.1	0.2
SPEEDUP FACTOR	6.45 ± 7.81	4.31 ± 5.12	3.96 ± 4.81	3.19 ± 3.58

5.3 Initialization of the Annealed NonConvex NP-SVM algorithm

This section investigates the influence of the initialization on the solution computed by the Annealed NonConvex NP-SVM solver (algorithm 3). For the sake of simplicity, we only consider a linear SVM trained on the SPAMBASE dataset. Besides the initialization $\boldsymbol{\beta} = \mathbf{0}$, which corresponds to training a plain SVM during the first iteration, we run the algorithm using 24 randomly generated initial vectors $\boldsymbol{\beta}$ and measure the error probabilities $\tilde{\mathbf{P}}_{\text{nd}}$ and $\tilde{\mathbf{P}}_{\text{fa}}$ achieved at convergence on the training set. The experiment was repeated for different values of C.

Figure 3 reports the results. Unsurprisingly, since the NP problem is intrinsically nonconvex, the initializations impact the solutions. The simplest initialization $\boldsymbol{\beta} = \mathbf{0}$ does not always lead to the best solution. However all the results are empirically very close to those achieved by using $\boldsymbol{\beta} = \mathbf{0}$.

5.4 Neyman-Pearson classification using Gaussian Kernel SVMs

This section compares Gaussian Kernel SVMs trained on the small-scale and medium-scale datasets using the Annealed NonConvex NP-SVM algorithm (NP-SVM, algorithm 3), the convex cost-sensitive SVM (AC-SVM) [Davenport et al. 2010], and the generative approach (GEN) [Huang et al. 2006]. We only report the best GEN results achieved using Chebychev bound $\Psi(u)$ to model probabilities of error (see section 2.1). We do not report results using the Stochastic NP optimizer (algorithm 4) because it is not suited to kernel machines. We do not report results obtained with SVMPerf [Joachims 2005] because, running the algorithm for plain kernel machines is "painfully slow", according to the SVMPerf web site.

The hyperparameters of the different methods were determined by cross-validation. Different algorithms have different model selection requirements: GEN only requires choosing a kernel bandwidth σ, whereas NP-SVM requires finding a pair (C, σ), and AC-SVM requires finding the best triplet (C_+, C_-, σ). The best classifier for each method was selected using the validation criterion [Davenport et al. 2010]:

$$J_{val} = \mathbf{P}_{\text{nd}} + \max(0, \mathbf{P}_{\text{fa}} - \rho)/\rho\,.$$

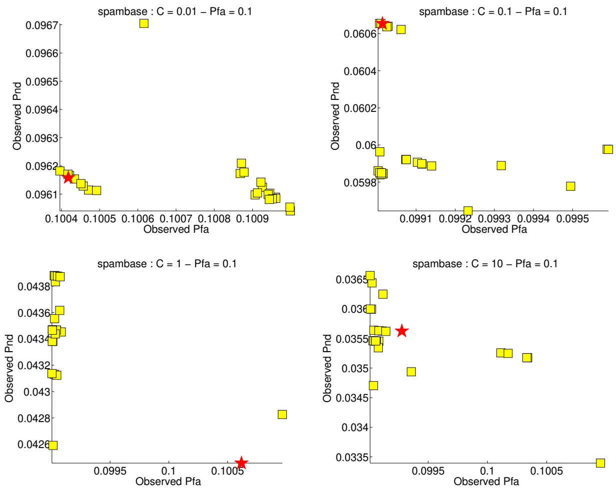

Fig. 3. Influence of the initialization on the Annealed NonConvex NP-SVM solver. The red star represents the initialization $\boldsymbol{\beta} = \mathbf{0}$. The yellow squares correspond to 24 random initializations.

This criterion has the property to enforce strongly the false alarm rate constraint for small values of ρ. All reported results were of course measured on the testing set after completing all the model selection procedure using a validation set or using 4-fold cross-validation, as explained in subsection 5.1. Following [Davenport et al. 2010], for each parameter, 10 logarithmically spaced values in the range $\left[10^{-2}, 10^{2}\right]$ were explored. This means that 10^3 triplets (C_+, C_-, σ) were tested for AC-SVM while only 100 pairs (C, σ) and 10 parameters σ were respectively tested for NP-SVM and GEN.

We used a C++ implementation of the SVM solvers with Torch[4] bindings, and a Matlab implementation of the generative approach[5].

Figure 4 reports the final classification errors $\mathbf{P}_{\mathrm{nd}}$ and $\mathbf{P}_{\mathrm{fa}}$, and the corresponding computation time achieved for several values of ρ.

As expected, NP-SVM tightly satisfies the false alarm constraint. NP-SVM achieves $\mathbf{P}_{\mathrm{nd}}$ misclassification rates similar or slightly better than AC-SVM on the Spambase, Breast, and Pima datasets. AC-SVM seems to perform better on PageBlocks. The reason could be related to the small number of negative samples available for the latter dataset. Indeed, according to the experimental protocol and Table I, when running NP-SVM for $\rho = 0.01$, one seeks a classifier with solely two misclassified negatives. NP-SVM strives to fullfill the constraint and overfits. On the other hand, on GammaTelescope, the number of examples is significantly higher, and NP-SVM reaches a better $\mathbf{P}_{\mathrm{nd}}$ than the competing techniques.

Except for Spambase and Breast, GEN was not able to attain $\mathbf{P}_{\mathrm{nd}}$ results competitive with NP-SVM or AC-SVM. GEN achieves a better $\mathbf{P}_{\mathrm{nd}}$ on PageBlock at the price of violating the $\mathbf{P}_{\mathrm{fa}}$ constraint. Despite its Matlab implementation, GEN

[4]See `http://torch5.sourceforge.net/`
[5]See `http://appsrv.cse.cuhk.edu.hk/~miplab/mempm_toolbox/index.htm`

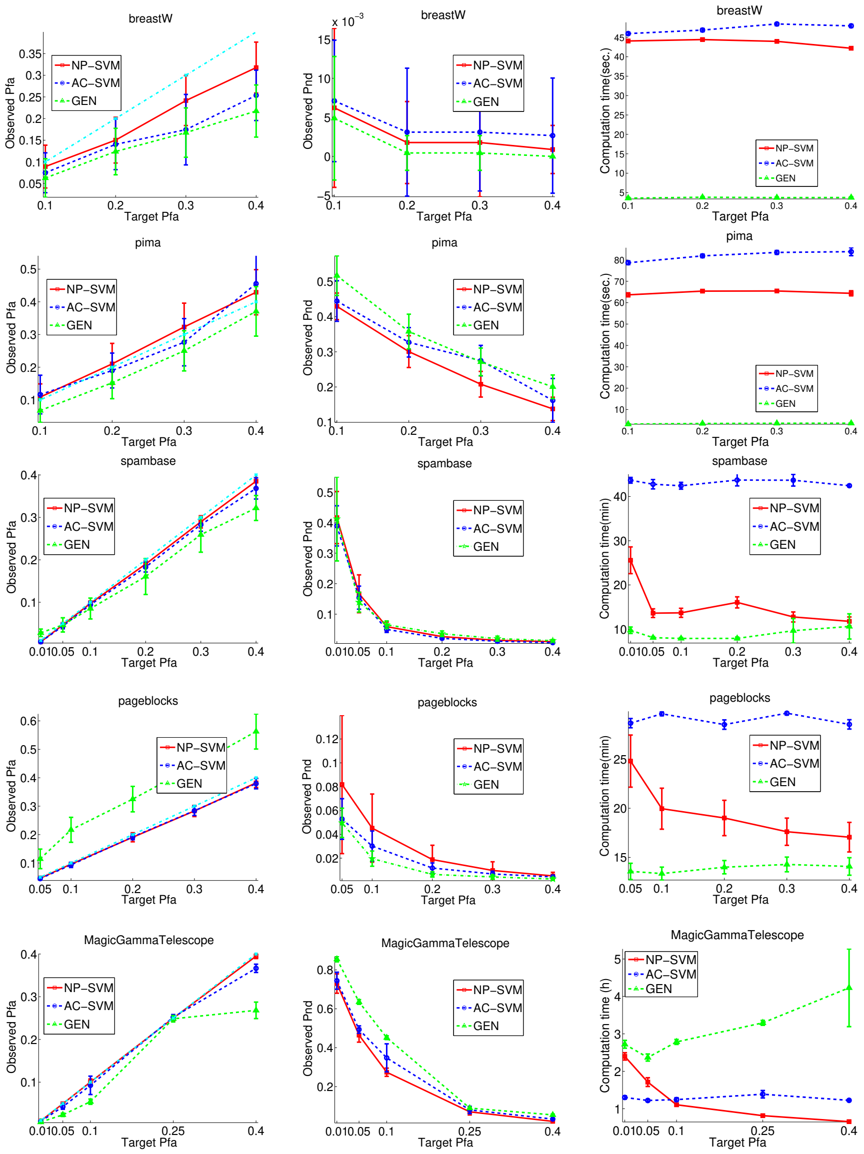

Fig. 4. Performance comparison for a Gaussian Kernel SVMs on different datasets (row-wise plots). Left: $\mathbf{P}_{\text{fa}}$. Middle: $\mathbf{P}_{\text{nd}}$. Right: total training time, including model selection. Results are averaged over 10 runs for `GammaTelescope` and 25 runs for the other datasets. In the left column, the cyan dash-dotted line represents the curve $\mathbf{P}_{\text{fa}} = \rho$.

appears to be the fastest method on the small size datasets. However, GEN slows down as soon as the size of the data grows, because its lack of sparsity results in expensive manipulations of the full kernel matrix. NP-SVM is very competitive with AC-SVM in terms of computation time, and even runs twice faster on Spambase[6]. The computational load of NP-SVM is high for small ρs and decreases when ρ increases. This phenomenon could be explained as follows: for small ρs, NP-SVM sets λ at high values during training, leading to highly skewed asymmetry. The SVM solver is then driven into extreme situations and runs slowly.

5.5 Neyman-Pearson classification using Linear SVMs

This section compares Linear SVMs trained on the medium-scale and large-scale datasets using the batch Annealed NonConvex NP-SVM solver (BNP-SVM, algorithm 3), the online Stochastic NP optimizer (ONP-SVM, algorithm 4), the convex cost-sensitive SVM (AC-SVM) [Davenport et al. 2010], the generative approach (GEN) [Huang et al. 2006], and the SVMPerf algorithm (SVMPerf) [Joachims 2005].

In the case of ONP-SVM, the hyperparameter selection searches the regularization parameter λ_c in the set $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$. The learning rate γ was sougth as τ/n_- with $\tau \in \{0.1, 1, 5, 10, 50, 100\}$. The algorithm was run for 100 epochs on the medium-scale datasets and 50 epochs on the large-scale datasets.

In the case of AC-SVM, 15×15 values of (C_+, C_-) were searched in the hyperbox $[0.001, 1000]^2$ for Pageblocks and $[0.01, 100]^2$ for the other datasets. The search for BNP-SVM only explores 25 values of C for Pageblocks and 15 values of C for the other datasets. The generative approach GEN is parameter free.

The SVMPerf results were obtained by adapting the SVMPerf program[7] to solve the Neyman-Pearson problem by exploiting its Precision/Recall@k mode. In this mode, SVMPerf computes a classifier yielding exactly k positive instances on the training set. Therefore the algorithm requires the specification of C and k. The hyperparameter C was searched as for the other SVM algorithm. Meanwhile, we explore ten values of k linearly spaced in the interval $(1/n_+, 1)$ and select value that most closely achieved the target $\mathbf{P}_{\text{fa}}$ on the validation set. However the reported SVMPerf training times exclude the time required to search parameter k. We simply report the running time associated with the final value of k because it would theoretically have been possible, but far from easy, to modify the intricate SVMPerf loops to do this automatically.

Experiments on the large scale datasets Covertype and RCV1-V2 were carried out using a number of speed optimizations. The SVM solver for AC-SVM was replaced by LIBLINEAR which is one of the most efficient solvers for linear SVM [Hsieh et al. 2008]. For the ONP-SVM experiments on RCV1-V2, we modified the stochastic gradient code of [Bottou 2007] because it handles sparse vectors more efficiently than our baseline code.

[6]During experiments carried out on other UCI small-scale datasets (Liver and Ionosphere), AC-SVM runs up to 1.5 times faster than NP-SVM. However, the computation time in question is about 25 seconds.

[7]`http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html`

Figure 5 reports the results obtained on all datasets but RCV1-V2. For readability, the SVMPerf results on COVERTYPE are shown separately in figure 6. The RCV1-V2 results are shown separately in table IV.

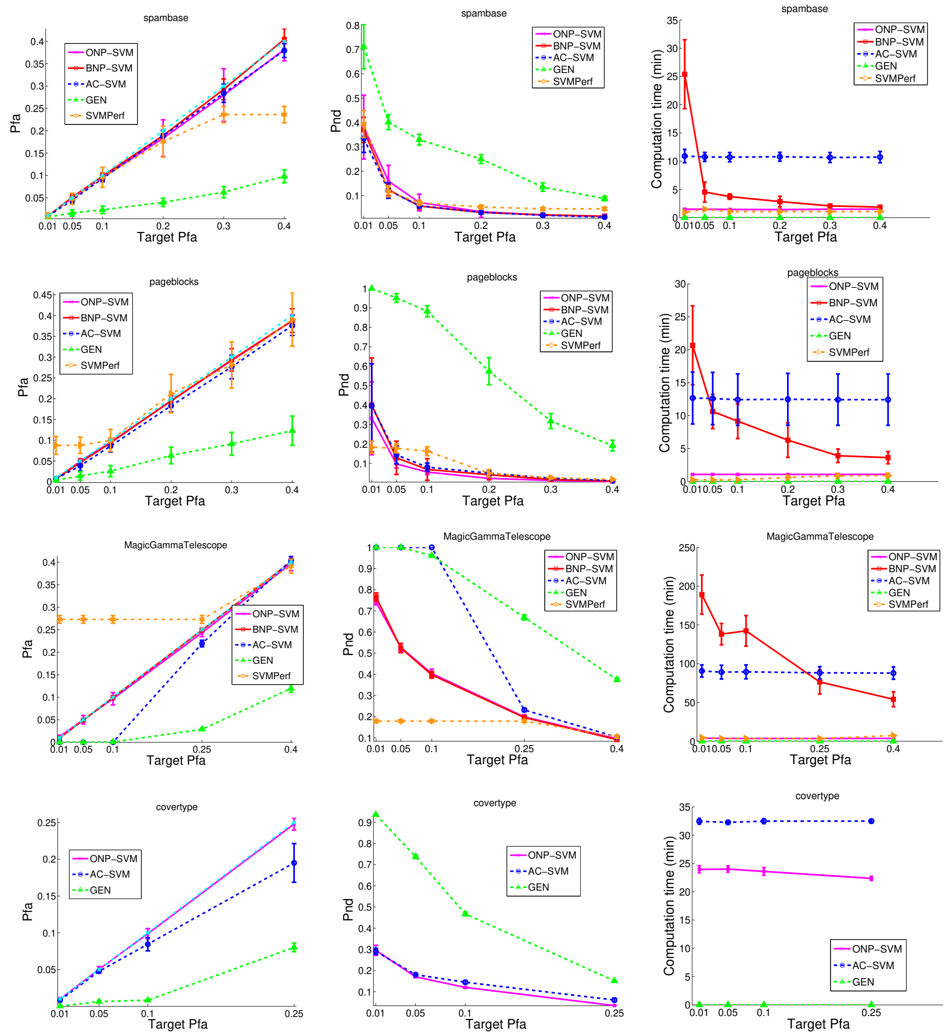

Fig. 5. Performance comparison for a Linear SVMs on different datasets (row-wise plots). Left: $\mathbf{P}_{\text{fa}}$. Middle: $\mathbf{P}_{\text{nd}}$. Right: training time, including model selection. The SPAMBASE and PAGEBLOCKS results are averaged over 25 runs. The other results are averaged on 10 runs. In the left column, the cyan dash-dotted line represents the curve $\mathbf{P}_{\text{fa}} = \rho$.

Several conclusions could be drawn: GEN is fast but performs poorly in terms of $\mathbf{P}_{\text{nd}}$. The Stochastic NP Optimizer has a constant numerical cost regardless

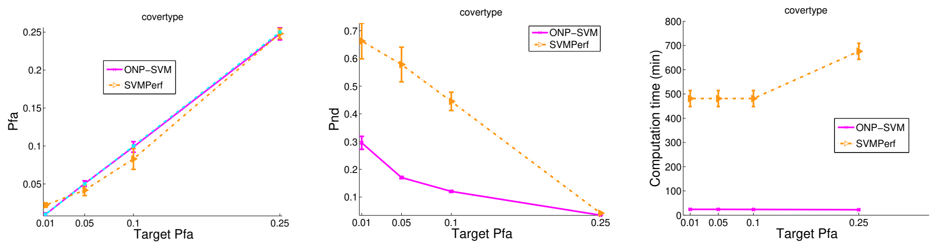

Fig. 6. Performance comparison for Linear SVMs on the COVERTYPE dataset, This plot shows the SVMPerf results that were not shown in figure 5 for the sake of readability.

Table IV. Performance comparison for a Linear SVM on the RCV1-V2 dataset. Top row: $\rho = 0.1\%$ (left) and $\rho = 0.5\%$ (right). Bottom Row: $\rho = 5\%$ (left) and $\rho = 10\%$ (right).

	ONP-SVM	AC-SVM
$\mathbf{P}_{\text{FA}}$	0.029%	0%
$\mathbf{P}_{\text{ND}}$	**76.8%**	93.26%

	ONP-SVM	AC-SVM
$\mathbf{P}_{\text{FA}}$	0.31%	0.145%
$\mathbf{P}_{\text{ND}}$	60%	**59.35%**

	ONP-SVM	AC-SVM
$\mathbf{P}_{\text{FA}}$	4.69%	5.01%
$\mathbf{P}_{\text{ND}}$	11.84%	**9.53%**

	ONP-SVM	AC-SVM
$\mathbf{P}_{\text{FA}}$	10%	8.3%
$\mathbf{P}_{\text{ND}}$	**4.63%**	7.9%

of ρ. It is really fast, especially for `Gammatelescope` where the computation gains compared to BNP-SVM and AC-SVM are respectively around 12 and 20. Moreover, the performances of ONP-SVM in terms of $\mathbf{P}_{\text{fa}}$ and $\mathbf{P}_{\text{nd}}$ are similar to those of batch methods. Interestingly, $\mathbf{P}_{\text{nd}}$ for ONP-SVM on PAGEBLOCKS is slightly better than its counterpart for the other algorithms, suggesting that ONP-SVM overcomes the lack of robustness of BNP-SVM in this regime. For GAMMATELESCOPE, our proposed approaches clearly outperform AC-SVM and GEN. The performances for AC-SVM could be improved using a finer grid search for C_+ and C_- or using a regularization path procedure [Yu et al. 2009]. However, this would increase the computing times. In particular the regularization path approach, in worst case scenario, has to visit an exponential number of kinks [Gärtner et al. 2009].

SVMPerf matches the speed and the accuracy of ONP-SVM on the SPAMBASE and PAGEBLOCKS datasets. It remains competitive in speed on the GAMMATELESCOPE dataset, but shows poor accuracies for small values of ρ. Enlarging the search interval for C or sampling k more finely did not provide any improvement. A possible explanation may lie with the fact that SVMPerf optimizes a convex upper bound of the target criterion, instead of the intrinsically nonconvex Neyman-Pearson objective. The scalability issues of SVMPerf become apparent on the COVERTYPE dataset (figure 6). Although the performances of SVMPerf could be improved by exploring more hyper-parameters, its very long training time discourages such a search.

The RCV1-V2 results shown in table IV again comfirm the good performance

Table V. Performance comparisons for q-value optimization measured as the number of true positives correctly identified on the testing set.

q	qRanker	qSVMOpt	qNNOpt
0.0025	4449	4947	**5005**
0.01	5462	5666	**5707**
0.1	7473	**7954**	7491

of the Stochastic NP Optimizer (ONP-SVM). Each run of ONP-SVM takes 30.44 seconds on average, whereas each call to the LIBLINEAR solver in AC-SVM takes 79.50 sec. As we tested 100 pairs of parameters (C_+, C_-) for AC-SVM and only 36 pairs (γ, ν) for ONP-SVM, the computing time gain is highly appreciable.

5.6 Illustration of q-value optimization on proteomics dataset

This subsection experimentally evaluates the q-value algorithms described in section 4 using the proteomics dataset described in [Spivak et al. 2009]. Following [Spivak et al. 2009], the performances was measured in terms of number of true positives correctly ranked over false positives. Table V summarizes the outcome of training a Gaussian Kernel SVM using of batch annealed nonconvex approach (qSVMOpt), and training a neural network using a stochastic gradient approach (qNNOpt). The neural network replicates the structure of the state-of-the-art model (qRanker) [Spivak et al. 2009] with a single hidden layer with 5 units. We use a sigmoid loss for qNNOpt and a ramp loss for qSVMOpt. Clearly our methods achieve the best generalization results, mainly for small q values which are of great importance in practice. The performance gain is greater than 10%. Let also mention that for qSVMOpt the computation time to solve for only one pair (C, σ) is about 41h while the neural network takes only 1h for 100 epochs. This illustrates the efficiency of the stochastic algorithm.

6. CONCLUSION

We have proposed a batch approach suited for kernel machines and online learning strategy for large datasets to tackle non-convex Neyman-Pearson classification problem. Experimental evaluations clearly illustrate the improvments achieved when dealing with non-convex NP problem. The stochastic gradient method was shown to be fast and interestingly efficient in generalisation. The approach was extended succesfully to solve q-value optimization problem in mass spectrometry protein screening applications. Interesting perspectives concern the extension of the online approach to kernel case or application of our NP solver to address bipartite ranking problem [Clémençon and Vayatis 2007].

REFERENCES

Andrieu, L., Cohen, G., and Vázquez Abad, F. 2007. Stochastic Programming with Probability Constraints. Under review.

Arrow, K., Hurwicz, L., and Uzawa, H. 1958. *Studies in Nonlinear Programming.* Stanford Univ. Press.

Bach, F. R., Heckerman, D., and Horvitz, E. 2006. Considering cost asymmetry in learning classifiers. *The Journal of Machine Learning Research 7*, 1741.

Bottou, L. 2007. Learning with large datasets. Tutorial of NIPS.

Bounsiar, A., Beauseroy, P., and Grall-Maës, E. 2008. General solution and learning method for binary classification with performance constraints. *Pattern Recognition Letters 29,* 10, 1455–1465.

Ciarlet, P. G. 1989. *Introduction to Numerical Linear Algebra and Optimisation*. Cambridge Univ. Press.

Clémençon, S. and Vayatis, N. 2007. Ranking the best instances. *Journal of Machine Learning Research 8*, 2671–2699.

Clémençon, S. and Vayatis, N. 2009. Overlaying classifiers: a practical approach for optimal ranking. In *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Vol. 21. MIT Press, Cambridge, MA, 313–320.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. 2006a. Large scale transductive svms. *Journal of Machine Learning Research 7*, 1687–1712.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. 2006b. Trading convexity for scalability. In *Proc. of International Conference on Machine Learning*. ACM, New York, NY, USA, 201–208.

Cortes, C. and Mohri, M. 2004. Auc optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. MIT Press, Cambridge, MA.

Davenport, M., Baraniuk, R., and Scott, C. 2010. Tuning support vector machines for minimax and neyman-pearson classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence 99,* PrePrints.

Elias, J. E. and Gygi, S. P. 2007. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods 4,* 3, 207–214.

Gärtner, B., Giesen, J., and Jaggi, M. 2009. An exponential lower bound on the complexity of regularization paths. *CoRR abs/0903.4817*.

Hsieh, C., Chang, K., Lin, C., Keerthi, S., and Sundararajan, S. 2008. A dual coordinate descent method for large-scale linear SVM. In *Proc. 25th Intl. Conf. on Machine Learning (ICML'08)*. Omnipress, 408–415.

Huang, K., Yang, H., King, I., and Lyu, M. 2006. Imbalanced learning with a biased minimax probability machine. *IEEE Transactions on Syst. Man and Cyb. Part B 36,* 4, 913.

Joachims, T. 2005. A support vector method for multivariate performance measures. In *Proc. of International Conference on Machine Learning*.

Käll, L., Canterbury, J., Weston, J., Noble, W., and MacCoss, M. 2007. A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. *Nature Methods 4*, 923–25.

Kim, S., Magnani, A., Samar, S., Boyd, S., and Lim, J. 2006. Pareto optimal linear classification. In *Proceedings of the 23rd international conference on Machine learning*. ACM, New York, NY, USA, 473–480.

Mozer, M., Dodier, R., Colagrosso, M., Guerra-Salcedo, C., and Wolniewicz, R. 2002. Prodding the ROC curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems*. Vol. 2. MIT Press, Cambridge, MA, 1409–1416.

Platt, J. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. MIT Press, Cambridge, MA, 185–208.

Scott, C. and Nowak, R. 2005. A Neyman-Pearson approach to statistical learning. *IEEE Transactions on Information Theory 51,* 11, 3806–3819.

Spivak, M., Weston, J., Bottou, L., Käll, L., and Noble, W. S. 2009. Improvments to the percolator algorithm for peptide identification form shotgun proteomics data sets. *Journal of Proteome Research 8*, 3737–3737.

Storey, J. D. and Tibshirani, R. 2003. Statistical significance for genome-wide studies. *PNAS 100*, 9440–9445.

Streit, R. 1990. A neural network for optimum neyman-pearson classification. In *Proc. of International Joint Conference on Neural Networks*. 685–690.

TAO, P. D. AND AN, L. T. H. 1998. Dc optimization algorithms for solving the trust region subproblem. *SIAM Journal of Optimization 8,* 2, 476–505.

TSYPKIN, Y. Z. 1971. *Adaptation and learning in automatic systems.* Academic Press, NY and London.

VAPNIK, V. N. 1998. *Statistical Learning Theory.* John Wiley & Sons.

YU, J., VISHWANATHAN, S., AND ZHANG, J. 2009. The entire quantile path of a risk-agnostic svm classifier. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence.*

A. NONCONVEX CONSTRAINED OPTIMIZATION

This appendix summarizes known results about the constrained optimization of nonconvex functions. We consider the problem

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad g_i(x) \leq 0,\ \forall i = 1\dots m \tag{14}$$

and its Lagrangian $\mathcal{L}(x,\lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x)$. We say that $(x,\lambda) \in \mathbb{R}^n \times \mathbb{R}^m_+$ is a *local saddle point* when there exists an open set $x \in \mathcal{V}(x) \subset \mathbb{R}^n$ such that

$$\begin{array}{ll} \text{(i)} & \forall \mu \in \mathbb{R}^m_+, \quad \mathcal{L}(x,\mu) \leq \mathcal{L}(x,\lambda)\,, \\ \text{(ii)} & \forall y \in \mathcal{V}(x), \quad \mathcal{L}(x,\lambda) \leq \mathcal{L}(y,\lambda)\,. \end{array} \tag{15}$$

The sufficiency result needs no additional assumption:

THEOREM 1. *If (x,λ) is a local saddle point* (15) *then x is a feasible local minimum of* (14)*.*

PROOF. From inequality (i) in (15), $\sum(\mu_i - \lambda_i)g_i(x) \leq 0$. We see that x is feasible, that is, $g_i(x) \leq 0$, by letting each μ_i go to $+\infty$. Therefore $\sum \lambda_i g_i(x) \leq 0$. On the other hand, setting $\mu = 0$ gives $\sum \lambda_i g_i(x) \geq 0$. Therefore $\sum \lambda_i g_i(x) = 0$. Combining with inequality (ii), for all $y \in \mathcal{V}(x)$ such that $g_i(y) \leq 0$ for $i = 1\dots m$, we have $g(x) = \mathcal{L}(x,\lambda) \leq \mathcal{L}(y,\lambda) \leq g(y)$. Therefore x is a local minimum of problem (14). □

Necessity results are more complicated because they involve delicate aspects of the Karush-Kuhn-Tucker (KKT) theory. But it is important to note that convexity is not required to establish that the KKT conditions are necessary:

THEOREM 2 [CIARLET 1989]. *Let x be a local minimum of problem* (14) *and let $\mathcal{I} = \{\, i;\ g_i(x) = 0 \,\}$. If the function f is differentiable in x, the functions g_i are differentiable in x for all $i \in \mathcal{I}$, the functions g_i are continuous in x for all $i \notin \mathcal{I}$, and the constraints are KKT-qualified in x, then there exist $\lambda \in \mathbb{R}^m_+$ such that $\lambda_i\, g_i(x) = 0$ and $\nabla_x \mathcal{L}(x,\lambda) = 0$.*

The Neyman-Pearson problem involves a single inequality constraint that is trivially KKT-qualified via the linear independence constraint qualification criterion. On the other hand, this theorem does not apply when we use the ramp loss because the minimum is likely to be achieved on a non differentiable point. But even in that case, theorem 1 ensures that any saddle point discovered by our algorithms is a feasible local minimum.

We now consider problem 14 with a single inequality constraint,

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad g(x)\leq 0\,,$$

and its Lagrangian $\mathcal{L}(x,\lambda)=f(x)+\lambda g(x)$. Let x^*_λ and $x^*_{\lambda'}$ be local minima of the Lagrangian for the fixed values λ and λ'. We can assume that our local minima satisfy the following inequalities

$$\mathcal{L}(x^*_\lambda,\lambda)\leq\mathcal{L}(x^*_{\lambda'},\lambda) \quad \text{and} \quad \mathcal{L}(x^*_\lambda,\lambda')\geq\mathcal{L}(x^*_{\lambda'},\lambda')\,. \tag{16}$$

Assume for instance that the first inequality above is not satisfied. This would mean that $x^*_{\lambda'}$ is a better local minimum of $\mathcal{L}(\cdot,\lambda)$ than x^*_λ. We could simply pick the best minimum instead of the inferior one.

THEOREM 3. *Consider a family of local minima* $\{\, x^*_\lambda : \lambda\in\Lambda\subset\mathbb{R}^*_+ \,\}$ *satisfying the inequalities* (16) *for any pair of Lagrange coefficients* $\lambda,\lambda'\in\Lambda$. *Then* $f(x^*_\lambda)$ *and* $g(x^*_\lambda)$ *are respectively nondecreasing and nonincreasing functions of* λ.

PROOF. The inequalities (16) can be rewritten as:

$$\lambda'(g(x^*_{\lambda'})-g(x^*_\lambda))\leq f(x^*_\lambda)-f(x^*_{\lambda'})\leq\lambda(g(x^*_{\lambda'})-g(x^*_\lambda))\,.$$

If $\lambda'>\lambda>0$, we must have $g(x^*_{\lambda'})-g(x^*_\lambda)\leq 0$ to ensure that the lower bound is smaller than the upper bound. Therefore $g(x^*_{\lambda'})\leq g(x^*_\lambda)$ and $f(x^*_{\lambda'})\geq f(x^*_\lambda)$. □

Therefore, under the assumption (16), the Uzawa algorithm (algorithm 1) amounts to searching the correct Lagrange coefficient in the nonincreasing sequence $\hat{\mathbf{P}}_{\text{fa}}(f^*_\lambda)$. The sign of the derivative $\nabla_\lambda\mathcal{L}(f^*_\lambda,\lambda)=\hat{\mathbf{P}}_{\text{fa}}(f^*_\lambda)-\rho$ truthfully indicates whether λ is above or below the correct value. Furthermore observe that assumption (16) can be realized by initializing each minimization with the local optimum obtained for the previous value of the Lagrange coefficient λ.