The cross entropy method for classification

Conference Paper · January 2005

DOI: 10.1145/1102351.1102422 · Source: DBLP

3 authors, including:

Dori Peleg
Technion - Israel Institute of Technology
8 PUBLICATIONS **481** CITATIONS

SEE PROFILE

The cross entropy method for classification

Shie Mannor SHIE@ECE.MCGILL.CA
Dept. of Electrical and Computer Engineering, McGill University, Montreal, Canada

Dori Peleg DORIP@TECHUNIX.TECHNION.AC.IL
Dept. of Electrical Engineering, Technion Institute of Technology, Haifa 32000, Israel

Reuven Rubinstein IERRR01@IE.TECHNION.AC.IL
Faculty of Industrial Engineering and Management, Technion Institute of Technology, Haifa 32000, Israel

Abstract

We consider support vector machines for binary classification. As opposed to most approaches we use the number of support vectors (the "L_0 norm") as a regularizing term instead of the L_1 or L_2 norms. In order to solve the optimization problem we use the cross entropy method to search over the possible sets of support vectors. The algorithm consists of solving a sequence of efficient linear programs. We report experiments where our method produces generalization errors that are similar to support vector machines, while using a considerably smaller number of support vectors.

1. Introduction

Many classification algorithms are focused on finding a classifier which minimizes the sum of the training error (or some related cost functional) and a complexity penalty. The intuition is that a classifier which is too complex might be tailored to the data and result in overfitting. The exact choice of a complexity penalty is often motivated by generalization abilities of the resulting classifier. However, this choice is also restricted by guaranteeing that the optimization problem is tractable. In Support Vector Machines (SVMs) one typically considers an L_2 complexity term (e.g., Cristianini & Shawe-Taylor, 2000; Vapnik, 1998), which can be justified in terms of minimizing the margin. This yields a quadratic optimization problem which can be solved remarkably efficiently up to a few thousand samples (see Platt, 1998; Keerthi et al., 2001). The main problem with using the L_2 complexity term is that it does not encourage sparse solutions. That is, the resulting classifier involves a significant fraction of the training points. Moreover, it was observed (e.g., Downs et al., 2001) that the solution produced by SVMs can even be made more sparse without changing the resulting classifier. Other kernel-based methods (e.g., Schölkopf & Smola, 2002) attempt to guarantee that the obtained classifier is sparse using different ideas. Most notably, the ν-SVM formulation (Chang & Lin, 2002) attempts to control the fraction of Support Vectors (SVs) by introducing a parameter used to tradeoff the number of SVs with the size of errors. Still, regularization is done using L_2 norm. In the context of classification, using the L_1 norm for regularization was used in Bradley and Mangasarian (1998) and Zhu et al. (2003).

A classifier with a small number of SVs is termed "sparse". Such classifiers enjoy several merits. First, sparse classifiers typically have better generalization error (e.g., Floyd & Warmuth, 1995 and more recently within the luckiness framework of Herbrich and Williamson (2002) and via compression bounds of Herbrich, 2002). Second, such classifiers are considered more robust to noise in the data (in the context of L_1 classification see Zhu et al., 2003, for a discussion). Third, the resulting classifier can be computed more quickly on test data, as it involves less terms. Improving the complexity of classification (over the test data) was considered Burges (1996); Burges and Schölkopf (1997); Downs et al. (2001), who considered simplifying a classifier obtained from solving an SVM. Typically, some SVs are discarded with little (or no, as in Downs et al., 2001) effect on the resulting classifier. Our approach is different. Instead of trying to simplify a solution obtained from solving an SVM, we directly solve an optimization problem aimed at producing a sparse classifier.

In this paper we consider the classical soft SVM prob-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

lem, but regularize using the number of SVs (the so-called L_0 norm). As a result we obtain a discontinuous and non-convex optimization problem. We formulate the problem as a search problem where one looks for the set of SVs. We apply the Cross Entropy (CE) method for efficiently searching for the set of SVs. The CE method provides a sequence of candidate sets of SVs. For each such candidate set, finding the weights of the SVs amounts to solving a reasonably sized linear program provided that the size of the candidate set is not too large. The CE method is a state-of-the-art heuristic method for solving combinatorial optimization problems and continuous multi-extremal problems (Rubinstein & Kroese., 2004). The basic idea behind the CE method is to transform a combinatorial optimization problem into a stochastic optimization problem, and then solve the stochastic optimization method adaptively. In many problems of practical interest, such as the travelling salesperson problem and the Max-cut problem, the CE method was observed to quickly reach a solution close to optimal (for a discussion and examples, see Rubinstein & Kroese., 2004). We note that it is not claimed that the CE method is capable of solving every instance of some NP-hard problem efficiently, but rather that it typically performs well. For a convergence proof of the CE method see Rubinstein and Kroese. (2004). We will use the CE-method to search over possible subsets of SVs.

We are concerned with the classical classification setup, i.e., given a training set of pairs $\left\{\mathbf{x}^{(i)}, y^{(i)}\right\}_{i=1}^{n}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ are input patterns and $y^{(i)} \in \{-1, 1\}$ are the corresponding labels, find a classifier $f(\mathbf{x})$ which will classify new patterns with as few errors as possible. We are also given a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. For a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ let $\|\boldsymbol{\alpha}\|_0 = \sum_{i=1}^{n} 1_{\{\alpha_i \neq 0\}}$ denote the "L_0 norm" that counts how many elements are not zero. The objective of this paper is to solve the problem

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{\alpha}\|_0 + C \sum_{i=1}^{n} \phi(y_i f(\mathbf{x}_i)) \\ \text{subject to} & 0 \leq \boldsymbol{\alpha} \leq C, \end{array} \tag{1}$$

with variables $\alpha \in \mathbb{R}^n$ and $b \in \mathbb{R}$, and where the hinge loss $\phi : \mathbb{R} \to \mathbb{R}_+$ is defined as $\phi(z) = 1 - z$ if $z \leq 1$, and zero otherwise; C is a hyperparameter; and the classifier f is defined by:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

The reminder of the paper is organized as follows. In Section 2 we review the CE method and provide the essential details needed here. In Section 3 we describe the classification algorithm based on the CE method. In Section 4 we report experimental results with an artificial data set and with several common data sets. In Section 5 we discuss the results and offer some conclusions.

2. The Cross Entropy Method

In this section we recapitulate the principles of the CE method. The reader is referred to Rubinstein and Kroese. (2004); de-Boer et al. (2005) and references therein for context, extensions, and applications. As mentioned, the basic idea behind the CE method is to transform the original (combinatorial) optimization problem to an associated stochastic optimization problem, and then to tackle the stochastic problem efficiently by an adaptive sampling algorithm. By doing so one constructs a random sequence of solutions which converges (probabilistically) to the optimal or at least a reasonable solution. Once the associated stochastic optimization is defined, the CE method alternates the following two phases:

1. Generation of a sample of random data (trajectories, vectors, etc.) according to a specified random mechanism.

2. Update of the parameters of the random mechanism, on the basis of the data, in order to produce a "better" sample in the next iteration.

We now present a proto-typical version of the CE method. Suppose we wish to minimize some cost function $S(\mathbf{z})$ over all $\mathbf{z}$ in some set $\mathcal{Z}$. Let us denote the minimum by γ^*, thus

$$\gamma^* = \min_{\mathbf{z} \in \mathcal{Z}} S(\mathbf{z}) \,. \tag{2}$$

We randomize our deterministic problem by defining a family of auxiliary pdfs $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ on $\mathcal{Z}$ and we associate with Eq. (2) the following estimation problem for a given scalar γ:

$$\mathbb{P}_{\boldsymbol{u}}(S(\mathbf{Z}) \leq \gamma) = \mathbb{E}_{\boldsymbol{u}} \left[I_{\{S(\mathbf{Z}) \leq \gamma\}} \right],$$

where $\boldsymbol{u}$ is some known (initial) parameter. We consider the event "cost is low" to be the rare event $I_{\{S(\mathbf{Z}) \leq \gamma\}}$ of interest. To estimate this event, the CE method generates a sequence of tuples $\{(\hat{\gamma}_t, \hat{\mathbf{v}}_t)\}$, that converge (with high probability) to a small neighborhood of the optimal tuple $(\gamma^*, \mathbf{v}^*)$, where γ^* is the solution of the program (2) and $\mathbf{v}^*$ is a pdf that emphasizes values in $\mathcal{Z}$ with a low cost. We note that

typically the optimal $\mathbf{v}^*$ is degenerated as it concentrates on the optimal solution (or a small neighborhood thereof). Let ρ denote the fraction of the best samples used to find the threshold γ. The process that is based on sampled data is termed the *stochastic counterpart* since it is based on stochastic samples of data. The number of samples in each stage of the stochastic counterpart is denoted by N, which is a predefined parameter. The following is a standard CE procedure for minimization borrowed from de-Boer et al. (2005). We initialize by setting $\hat{\mathbf{v}}_0 = \mathbf{v}_0 = \boldsymbol{u}$ and choose a not very small ρ, say $10^{-2} \leq \rho$. We then proceed iteratively as follows:

1. **Adaptive updating of γ_t.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be a $\rho 100\%$-percentile of $S(\mathbf{Z})$ under $\mathbf{v}_{t-1}$. That is, γ_t satisfies $\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{Z}) \leq \gamma_t) \geq \rho$ and $\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{Z}) \geq \gamma_t) \geq 1-\rho$ where $\mathbf{Z} \sim f(\cdot; \mathbf{v}_{t-1})$. A simple estimator $\widehat{\gamma}_t$ of γ_t can be obtained by taking a random sample $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(N)}$ from the pdf $f(\cdot; \mathbf{v}_{t-1})$, calculating the performances $S(\mathbf{Z}^{(\ell)})$ for all ℓ, ordering them from smallest to biggest as $S_{(1)} \leq \ldots \leq S_{(N)}$ and finally evaluating the $\rho 100\%$ sample percentile as $\widehat{\gamma}_t = S_{(\lceil \rho N \rceil)}$.

2. **Adaptive updating of $\mathbf{v}_t$.** For a fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ from the solution of the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} I_{\{S(\mathbf{Z}) \leq \gamma_t\}} \log f(\mathbf{Z}; \mathbf{v}) \; . \tag{3}$$

The stochastic counterpart of (3) is as follows: for fixed $\widehat{\gamma}_t$ and $\widehat{\mathbf{v}}_{t-1}$, derive $\widehat{\mathbf{v}}_t$ from the following program:

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{\ell=1}^{N} I_{\{S(\mathbf{Z}^{(\ell)}) \leq \widehat{\gamma}_t\}} \log f(\mathbf{Z}^{(\ell)}; \mathbf{v}) \; . \tag{4}$$

We note that if f belongs to the Natural Exponential Family (e.g., Gaussian, Bernoulli), then Eq. (4) has a closed form solution (see de-Boer et al., 2005). The CE optimization algorithm is summarized in Algorithm 2.1. In this paper we will assume that f belongs to a Bernoulli family. In our case $\mathcal{Z} = \{0,1\}^n$ and $\mathbf{v}$ is an n dimensional vector of numbers between 0 and 1. The update formula of the kth element in $\mathbf{v}$ (Eq. (4)) in this case simply becomes:

$$\hat{\mathbf{v}}_t(k) = \frac{\sum_{\ell=1}^{N} I_{\{S(\mathbf{Z}^{(\ell)}) \leq \widehat{\gamma}_t\}} I_{\{Z_k^{(\ell)}=1\}}}{\sum_{\ell=1}^{N} I_{\{S(\mathbf{Z}^{(\ell)}) \leq \widehat{\gamma}_t\}}}.$$

This formula has the interpretation that it counts how many times a value of 1 (in $I_{\{Z_k^{(\ell)}=1\}}$) led to a significant result (matches with the indicator $I_{\{S(\mathbf{Z}^{(\ell)}) \leq \widehat{\gamma}_t\}}$), how many times a value of 0 led to a significant result, and normalize the value of the parameter accordingly. Instead of the updating the parameter vector $\mathbf{v}$ directly via the solution of Eq. (4) we use the following *smoothed* version

$$\widehat{\mathbf{v}}_t = \beta \tilde{\mathbf{v}}_t + (1-\beta)\widehat{\mathbf{v}}_{t-1}, \tag{5}$$

where $\tilde{\mathbf{v}}_t$ is the parameter vector obtained from the solution of (4), and β is a smoothing parameter, with $0.7 < \beta < 1$. The reason for using the smoothed (5) instead of the original updating rule is to smooth out the values of $\widehat{\mathbf{v}}_t$, and to reduce the probability that some component $\widehat{v}_{t,i}$ of $\widehat{\mathbf{v}}_t$ will be zeros or unities at an early stage, and the algorithm will get stuck in a local maxima. Note that for $0 < \beta < 1$ we always have that $\widehat{v}_{t,i} > 0$, while for $\beta = 1$ one might have (even at the first iterations) that either $\widehat{v}_{t,i} = 0$ or $\widehat{v}_{t,i} = 1$ for some indices i. As a result, the algorithm may converge to a wrong solution.

Algorithm 2.1 The CE Method for Stochastic Optimization

1. Choose some $\widehat{\mathbf{v}}_0$. Set $t = 1$ (level counter).
2. Generate a sample $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(N)}$ from the density $f(\cdot; \mathbf{v}_{t-1})$ and compute the sample $\rho 100\%$-percentile $\widehat{\gamma}_t$ of the sample scores.
3. Use the **same** sample $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(N)}$ and solve the stochastic program (4). Denote the solution by $\widetilde{\mathbf{v}}_t$.
4. Apply (5) to smooth out the vector $\widetilde{\mathbf{v}}_t$.
5. If for some $t \geq d$, say $d = 3$, $\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d}$ then **stop**; otherwise set $t = t + 1$ and reiterate from step 2.

Figure 1. A proto-typical CE algorithm.

It is found empirically (Rubinstein & Kroese., 2004) that the CE method is robust with respect to the choice of its parameters N, ρ and β, as long as ρ is not too small, $\beta < 1$, and N is large enough. Typically those parameters satisfy that $0.01 \leq \rho \leq 0.1$, $0.5 \leq \beta \leq 0.9$, and $N \geq 3n$, where n is the number of parameters. Improvements of Algorithm 2.1 include the Fully Adaptive CE (FACE) variant, where the parameters N and ρ are updated online. See de-Boer et al. (2005) for more details.

3. The CE classifier algorithm

Problem (1) is a difficult non-convex discontinuous optimization problem due to the 'zero norm' of $\boldsymbol{\alpha}$. In

order to circumvent the discontinuity we introduce a vector of n binary variables $\boldsymbol{\sigma} \in \{0,1\}^n$ and define $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$. We replace $\boldsymbol{\alpha}$ with $\Sigma\hat{\boldsymbol{\alpha}}$, i.e., the vector $\boldsymbol{\sigma}$ multiplies the n dimensional vector $\hat{\boldsymbol{\alpha}}$ componentwise. Thus (1) is transformed to the problem

$$\begin{array}{ll} \text{minimize} & \|\Sigma\hat{\boldsymbol{\alpha}}\|_0 + C\mathbf{1}^\top\boldsymbol{\xi} \\ \text{subject to} & Y\left(KY\Sigma\hat{\boldsymbol{\alpha}} + b\mathbf{1}\right) \geq \mathbf{1} - \boldsymbol{\xi} \\ & \boldsymbol{\xi} \geq 0 \\ & 0 \leq \Sigma\hat{\boldsymbol{\alpha}} \leq C \\ & \boldsymbol{\sigma} \in \{0,1\}^n, \end{array} \tag{6}$$

with variables $\hat{\boldsymbol{\alpha}}, \boldsymbol{\sigma}, b$ and $\boldsymbol{\xi} \in \mathbb{R}^n$, where $\mathbf{1}$ is a vector of ones, K is the kernel matrix satisfying $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, and $Y = \text{diag}(y_1, \ldots, y_n)$. Since any component of α can be written as $1 \cdot \hat{\alpha}$ if $\alpha > 0$ or $0 \cdot \hat{\alpha}$ if $\alpha = 0$ then problems (1) and (6) are equivalent.

The zero norm counts the number of non-zero elements in a vector. Therefore $\|\Sigma\hat{\boldsymbol{\alpha}}\|_0 = \sum_{i=1}^n \sigma_i$ equals the number of SVs (in practice $\hat{\boldsymbol{\alpha}}$ is never exactly zero). Denote by $\tilde{\boldsymbol{\alpha}}$ the vector of non-zero elements of vector $\Sigma\hat{\boldsymbol{\alpha}}$. Denote $\tilde{K}$ as the kernel matrix K without the columns indicated by the zeros of $\boldsymbol{\sigma}$ and $\tilde{Y}$ as the diagonal matrix of labels without the labels indicated by the zeros of $\boldsymbol{\sigma}$. For a given vector $\boldsymbol{\sigma}$, the problem (6) is reduced to:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^\top\boldsymbol{\xi} \\ \text{subject to} & Y\left(\tilde{K}\tilde{Y}\tilde{\boldsymbol{\alpha}} + b\mathbf{1}\right) \geq \mathbf{1} - \boldsymbol{\xi} \\ & \boldsymbol{\xi} \geq 0 \\ & 0 \leq \tilde{\boldsymbol{\alpha}} \leq C, \end{array} \tag{7}$$

with variables $\tilde{\boldsymbol{\alpha}}, \boldsymbol{\xi}, b$. Notice that the dimension of $\tilde{\boldsymbol{\alpha}}$ is equal to the number of SVs, i.e., $\sum_{i=1}^n \sigma_i$. This is typically a much smaller number than the number of training patterns n. Problem (7) is a Linear Programming (LP) problem which can be solved with highly efficient optimization algorithms. The number of variables is $(\sum_{i=1}^n \sigma_i)+n+1$ and the number of constraints is $2n+2\left(\sum_{i=1}^n \sigma_i\right)$. Therefore the algorithm is affected by the number of features only when the kernel matrix is initially calculated. We propose to solve problem (1) by using the CE algorithm for the combinatorial optimization problem of selecting the set of SVs. We use Bernoulli distribution for $f(\cdot; \mathbf{v})$, with the variable $\boldsymbol{\sigma}$ playing the role of $\mathbf{z}$ in the CE method. The objective function is

$$S(\boldsymbol{\sigma}) = \mathbf{1}^\top\boldsymbol{\sigma} + C\mathbf{1}^\top\boldsymbol{\xi},$$

where $\mathbf{1}^\top\xi$ was calculated by solving (7). We chose $\rho = 0.03, \beta = 0.7, d = 3, N = 5n$, where n is the number of training patterns. The values of parameters ρ, β, d are standard[1] and the value of N was determined via trial and error to be the lowest number which consistently produces the same results as with a larger N.

Consider the following optimization problem, known as the "one norm SVM":

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^\top\boldsymbol{\alpha} + C\mathbf{1}^\top\boldsymbol{\xi} \\ \text{subject to} & Y\left(KY\boldsymbol{\alpha} + b\mathbf{1}\right) \geq \mathbf{1} - \boldsymbol{\xi} \\ & \boldsymbol{\xi} \geq 0 \\ & 0 \leq \boldsymbol{\alpha} \leq C, \end{array} \tag{8}$$

Problem (8) is an upper bound on the "zero norm" problem (1). This approach was introduced by Smola et al., (2000, pp. 141-142) and produces less sparse solutions than those generated by the "zero norm" problem. From a computational point of view, this problem can be solved efficiently. In order to reduce the complexity of our algorithm, we first run a one norm SVM, and then only consider the SVs that were computed by the one norm as potential SVs. It is reasonable to assume that the majority of SVs of the zero norm problem are SVs of the one-norm problem because these SVs are representative in some sense. A similar approach was adopted in Burges and Schölkopf (1997). Thus we suggest to use the solution of problem (8) as an initial solution for problem (6).

Three modifications were made to the standard CE algorithm:

1. The solution $\tilde{\boldsymbol{\alpha}}$ of (7) may have nearly zero elements. To remove redundancy and to make the term of number of SVs in the target function $S(\boldsymbol{\sigma})$ slightly less rigid, we decided that if the value of $\tilde{\alpha}_j < 10^{-6}$ then the corresponding σ_j was set to zero.

2. In a CE iteration frequently identical $\boldsymbol{\sigma}$ vectors are generated (especially near convergence). Given the same $\boldsymbol{\sigma}$, the solution of the convex problem (7) is identical. Thus computing (7) again is superfluous. To avoid this redundancy we used a cache for the results of (7). We note that caching can be done in an extremely simple way. Each $\boldsymbol{\sigma}$ (a vector of ones and zeros) is assigned a binary number corresponding to its value. The $\boldsymbol{\sigma}$'s are sorted in ascending order. For each $\boldsymbol{\sigma}$ problem (7) is computed only if its value does not appear in the (sorted) list of already computed values[2].

3. Before the CE algorithm is executed the non SVs of the solution of Problem (8) are excluded from

[1] We found that the CE algorithm is not sensitive to these parameters as long as they are chosen from the ranges specified at the end of Section 2.

[2] After the target function is calculated without redundancy, the $\boldsymbol{\sigma}$ vectors are sorted according to the value of their target function as explained in (2.1).

consideration as potential SVs. All the training points (including the "removed" SVs) are still taken into account when solving (7). Note that this steps guarantees sparser solution than those generated by Smola et al. (2000).

In summary, the training phase of the CE classifier algorithm consists of solving typically several thousand LP problems. We note that the number of variables used by each LP is independent of the number of features that only affects the computation of the kernel matrix, which is performed once.

4. Experiments

The CE classifier algorithm was compared to the SVM and the one-norm SVM (Smola et al., 2000) algorithms on one synthetic dataset and five real world two class problem datasets taken from the UCI repository (see Table 1).

4.1. Synthetic data

The synthetic dataset was generated in the following manner. The probability of $y = 1$ or -1 was equal. Features x_1, x_2 were drawn as $x|y \sim \mathcal{N}(y\mathbf{1}, \Lambda)$, with the covariance matrix $\Lambda = VDV^T$ and $\mathbf{1}$ a vector of ones. The matrix D was diagonal with the values $1, 2$ and V a randomly generated unitary matrix. The number of training patterns was 300 and the test set consisted of 1000 patterns.

The linear kernel was used for all algorithms. The value of hyperparameter C for each algorithm was set as the minimizer of the errors on the test set. The SVM algorithm achieved 24.5% with 56.7% of the training patterns. The one-norm SVM algorithm achieved 24.5% with 0.667% of the training patterns. The CE algorithm achieved 24.3% with 0.333% of the training patterns. Figures 2-4 illustrate the sparsity of the one-norm SVM and CE algorithm in comparison to the SVM algorithm. The class of each training pattern is marked by a circle or a square. If the shape is full, this means that the training pattern acts as a support vector. Note that the CE algorithm utilizes only one training pattern out of 300 for the best results.

4.2. Real-world data

Each dataset was divided ten times into a training set of 150 patterns and a test set of the remaining patterns. Three kernels were used:

1. Linear: $K_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.
2. Polynomial of degree 5: $K_{i,j} = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^5$.
3. Gaussian, or radial basis functions (RBF): $K_{i,j} = \exp(-(\langle \mathbf{x}_i, \mathbf{x}_j \rangle^2)/(2\sigma^2_{RBF}))$.

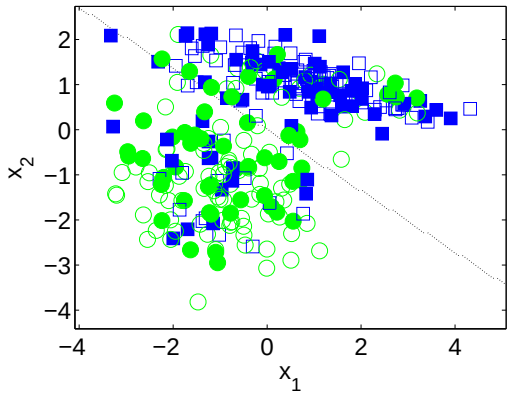

Figure 2. SVM classifier for synthetic problem (56.7% SVs).

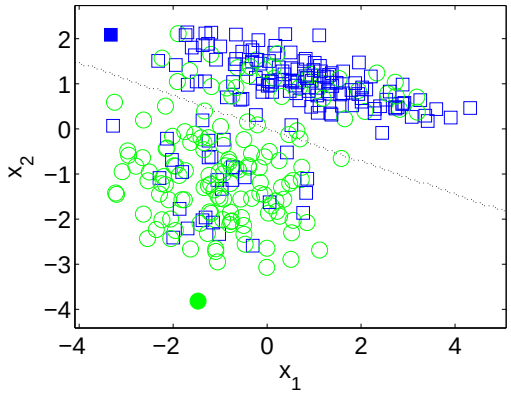

Figure 3. One-norm SVM classifier for synthetic problem (0.667% SVs).

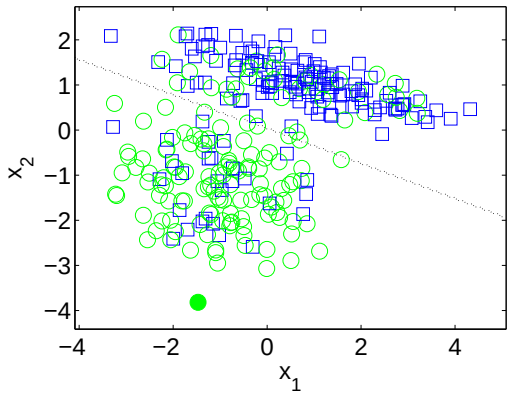

Figure 4. CE classifier for synthetic problem (0.333% SVs).

For the linear and polynomial kernels the 5-fold cross validation error was calculated for $C = \lambda/(1-\lambda)$, where λ had values between 0.1 and 0.9 in increments of 0.1. The RBF kernel is strongly dependant on its hyperparameter σ_{RBF}. Therefore we set $C = 1$ and the 5-fold cross validation error was calculated for 9 linearly spaced values of σ_{RBF} between 1 and 5. Nine values were used to match the number of hyperparameter values for the other kernels. This particular range of value of hyperparameter σ_{RBF} was used because we observed that typically for these datasets the kernel matrix shifted from an almost unit matrix ($\sigma_{RBF} = 1$) to a matrix of ones ($\sigma_{RBF} = 5$).

The training features of all the datasets, algorithms and kernels were preprocessed to have zero mean. In the case of the linear kernel only zero mean preprocessing was used. In the case of the polynomial kernel all datasets but the first dataset were preprocessed to have a unit variance (in addition to zero mean). In the case of the Gaussian kernel for all but the first dataset and only the SVM algorithm for the second and third datasets the data were preprocessed to have a unit variance (all datasets were preprocessed to have zero mean). The test examples were processed accordingly.

Table 1. The real-world datasets. The value of hyperparameter C was equal to λ for the Bupa dataset to allow convergence.

No.	Name	Features	Patterns
1	Ionosphere	34	351
2	Pima	8	768
3	Wdbc	30	569
4	Bupa	6	345
5	Sonar	60	208

The algorithms are compared by two criteria: the error rate and the number of Support Vectors (SVs). Results for linear kernel, polynomial kernel, and Gaussian kernel are presented in Tables 2, 3, and 4, respectively. The criteria was averaged over the partitions into training and test sets.

The CE classifier produced error rates comparable to the standard SVM algorithm. Yet it consistently did so by using a significantly smaller percentage of SVs. On average, across the real-world datasets, the CE classifier used $29\%, 70\%, 40\%$ less SVs than the one-norm algorithm with the linear, polynomial and Gaussian kernels, respectively. It is interesting to note that this behavior was observed for both a "local" kernel (such as the RBF kernel) and a "global" kernel (such as the linear kernel), indicating that this phenomena cannot be attributed to the locality of the kernel.

5. Discussion

In this paper we considered solving SVM type classification problems with zero norm regularization. Since the optimization problem is non-convex and even discontinuous we formulated it as a combinatorial optimization problem and applied the CE method to solve it. The suggested method was shown to be able to cope well with several rather difficult problems. It produced test errors that were statistically similar to SVMs with L_2 regularization. The main benefit of the method is the sparsity of the solution which was significant in all the instances we experienced with.

A natural question at this point would be why did we employ the CE method and not some other heuristic search method such as genetic algorithms or simulated annealing for searching over the set of SVs. There are two broad types of heuristic search methods. The first type is based on having a proximity relation that facilitates efficient local search. Examples of this type include simulated annealing (Aarts & Korst, 1989) and guided local search (Voudouris & Tsang, 1999). The second type includes global methods that are typically based on iteratively creating collections of solutions, improving the quality of the collection from iteration to iteration. The first type of search algorithms is not suitable for the zero-norm minimization problem because the problem has many discontinuities, and algorithms that are based on local search are likely to get stuck. Algorithms that employ some penalty function on traversed areas (such as the guided local search algorithm) are not likely to perform well since there are many discontinuities leading to an overly complicated penalty function. The CE method belongs to the second type of algorithms. As opposed to other algorithms of this type, such as genetic algorithms, the CE method naturally fits the classification problem and is easy to use. The CE method does not require tedious tweaking or definition of unnatural operators as genetic algorithms do. Finally, similarly to simulated annealing the CE method is guaranteed to converge to the optimal solution in the large sample limit. This provides some theoretical guarantees on the performance of the algorithm.

The computational cost of running the CE based classifier is determined by the size of the LP problems and by the available computational resources. Currently, there are excellent and highly optimized LP solvers that are readily available. In the problems we describe above, a typical run of the CE method required solving a few thousand small linear programs, which can be done rather efficiently. It is worth noting that the CE classification algorithm is "embarrassingly paral-

Table 2. The error and SV percentage (in parentheses) for the real-world datasets with a linear kernel. The number after the ± represents the standard deviation of either the error or the percentage of SVs. The lowest error and sparsest solution for every dataset are in boldface.

No.	SVM	One-norm SVM	CE
1	14.7±2.0 (36.4±8.9)	**13.0±1.8** (15.1±2.5)	14.6±1.8 (**7.7±2.6**)
2	**24.3±1.4** (51.2±6.2)	24.6±1.1 (4.9±0.5)	24.8±1.5 (**3.9±1.2**)
3	**5.7±1.2** (10.1±2.5)	5.9±1.4 (4.8±1.1)	5.9±0.8 (**3.7±1.4**)
4	32.6±2.1 (71.9±3.8)	**32.5±1.7** (4.0±0.0)	33.4±2.8 (**3.1±0.6**)
5	25.9±3.7 (53.7±7.9)	**25.5±4.7** (14.7±2.4)	**25.5±4.7** (**10.3±1.9**)

Table 3. The error and SV percentage for the real-world datasets with a polynomial kernel of degree 5.

No.	SVM	One-norm SVM	CE
1	15.3±2.7 (36.1±3.7)	13.7±2.6 (20.5±8.4)	**12.5±1.3** (**7.1±1.1**)
2	33.2±1.5 (48.8±5.2)	30.6±1.8 (29.5±4.6)	**30.2±2.4** (**11.2±6.6**)
3	6.0±2.1 (21.9±2.7)	8.5±2.8 (15.1±3.2)	**5.6±1.3** (**2.5±0.7**)
4	**33.7±5.2** (58.0±6.0)	36.3±2.2 (33.9±3.5)	37.9±4.4 (**14.4±9.9**)
5	**15.9±4.7** (70.3±1.7)	20.3±7.0 (51.1±6.8)	23.3±5.3 (**6.9±1.6**)

Table 4. The error and SV percentage (in parentheses) for the real-world datasets with a Gaussian kernel and $C = 1$.

No.	SVM	One-norm SVM	CE
1	9.8±2.3 (76.3±2.2)	**6.2±1.5** (19.3±3.1)	6.6±2.3 (**14.1±2.6**)
2	27.5±1.7 (67.9±5.1)	**25.2±3.0** (12.9±4.9)	25.4±3.3 (**8.5±1.7**)
3	7.5±0.8 (42.4±3.4)	**4.6±1.5** (14.4±1.5)	4.7±1.4 (**9.7±1.2**)
4	**34.4±3.0** (93.4±1.6)	36.9±3.9 (28.3±25.5)	36.9±4.6 (**10.4±4.3**)
5	46.7±6.3 (100.0±0.0)	**24.3±3.5** (41.7±6.2)	24.5±3.7 (**22.5±2.5**)

lel" since the sampling method and calculation of the score (solving the LP) can be efficiently divided across processors. As a result, a speedup which is linear in the number of available CPUs can be obtained.

We observed in experiments that using the CE method as a second phase after running one norm SVM results in a sparser solution with a similar error. The CE method can therefore be considered as a post-processing phase, where classifiers that are obtained using a "standard" method are being made sparser. We expect that even greater gain would be obtained for regression problems. For such problems, the fraction of SVs that is typically obtained by kernel methods is large. The issue of using the CE method to further sparsify regressors is left for future research.

The classification algorithm we suggested operates on a single data set. In order to speed-up the computation for large data sets, it may be of interest to use active set methods (see Mangasarian & Musicant, 2001 for consideration of active set methods in SVMs). The idea is to partition the classification problem to multiple smaller optimization problems. After each of these problems is solved separately a global solution is obtained by merging the solutions of the smaller problems. This may be done, perhaps, by interleaving activation of one norm SVMs with CE method type iterations on different parts of the data set. This issue is left for future research.

Acknowledgements

This research was partially supported by NSERC.

References

Aarts, E., & Korst, J. (1989). *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons.

Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. *Proceedings of the 15th International Conference on Machine Learning* (pp. 82–90).

Burges, C. J. C. (1996). Simplified support vector decision rules. *Proceedings of the 13th International Conference on Machine Learning* (pp. 71–77).

Burges, C. J. C., & Schölkopf, B. B. (1997). Improving speed and accuracy of support vector learning ma-

chines. *Advances in Neural Information Processing Systems, 9* (pp. 375 – 381).

Chang, C.-C., & Lin, C.-J. (2002). Training nu-support vector regression: theory and algorithms. *Neural Computation*, *14*, 1959–1977.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, England: Cambridge University Press.

de-Boer, P., Kroese, D., Mannor, S., & Rubinstein, R. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, *134*, 19–67.

Downs, T., Gates, K., & Masters, A. (2001). Exact simplification of support vector solutions. *Journal of Machine Learning Research*, *2*, 293–297.

Floyd, S., & Warmuth, M. (1995). Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, *21*, 269–304.

Herbrich, R. (2002). *Learning kernel classifiers: Theory and algorithms*. Boston: MIT Press.

Herbrich, R., & Williamson, R. C. (2002). Algorithmic luckiness. *Journal of Machine Learning Research*, *3*, 175–212.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to platts SMO algorithm for SVM classifier design. *Neural Computation*, *13*, 637 –649.

Mangasarian, O., & Musicant, D. (2001). Active set support vector machine classification. *Advances in Neural Information Processing Systems, 13* (pp. 577–583).

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods - support vector learning*. MIT press.

Rubinstein, R., & Kroese., D. (2004). *The cross-entropy method: A unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer-Verlag.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.

Smola, A. J., Bartlett, P., Schökopf, B., & Schuurmans, D. (2000). *Advances in large margin classifiers*. MIT Press.

Vapnik, V. N. (1998). *Statistical Learning Theory*. New York: Wiley Interscience.

Voudouris, C., & Tsang, E. (1999). Guided local search. *European Journal of Operational Research*, *113*, 469–499.

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2003). 1-norm support vector machines. *Advances in Neural Information Processing Systems, 16*.