Breaking SVM Complexity with Cross-Training

Gökhan H. Bakır
Max Planck Institute
for Biological Cybernetics,
Tübingen, Germany
gb@tuebingen.mpg.de

Léon Bottou
NEC Labs America
Princeton NJ, USA
leon@bottou.org

Jason Weston
NEC Labs America
Princeton NJ, USA
jasonw@nec-labs.com

Abstract

We propose an algorithm for selectively removing examples from the training set using probabilistic estimates related to editing algorithms (Devijver and Kittler, 1982). The procedure creates a separable distribution of training examples with minimal impact on the decision boundary position. It breaks the linear dependency between the number of SVs and the number of training examples, and sharply reduces the complexity of SVMs during both the training and prediction stages.

1 Introduction

The number of Support Vectors (SVs) has a dramatic impact on the efficiency of Support Vector Machines (SVM) (Vapnik, 1995) during both the learning and prediction stages. Recent results (Steinwart, 2004) indicate that the number k of SVs increases linearly with the number n of training examples. More specifically,

$$k/n \longrightarrow 2\mathcal{B}_K \tag{1}$$

where n is the number of training examples and $\mathcal{B}_K$ is the smallest classification error achievable with the SVM kernel K. When using a universal kernel such as the Radial Basis Function kernel, $\mathcal{B}_K$ is the Bayes risk $\mathcal{B}$, i.e. the smallest classification error achievable with any decision function.

The computational requirements of modern SVM training algorithms (Joachims, 1999; Chang and Lin, 2001) are very largely determined by the amount of memory required to store the active segment of the kernel matrix. When this amount exceeds the available memory, the training time increases quickly because some kernel matrix coefficients must be recomputed multiple times. During the final phase of the training process, the active segment always contains all the $k(k+1)/2$ dot products between SVs. Steinwart's result (1) then suggests that the critical amount of memory scales at least like $\mathcal{B}^2n^2$. This can be practically prohibitive for problems with either big training sets or large Bayes risk (noisy problems). Large numbers of SVs also penalize SVMs during the prediction stage as the computation of the decision function requires a time proportional to the number of SVs.

When the problem is separable, i.e. $\mathcal{B} = 0$, equation (1) suggests[1] that the number k of SVs increases less than linearly with the number n of examples. This improves the scaling laws for the SVM computational requirements.

[1] See also (Steinwart, 2004, remark 3.8)

In this paper, we propose to selectively remove examples from the training set using probabilistic estimates inspired by training set editing algorithms (Devijver and Kittler, 1982). The removal procedure aims at creating a separable distribution of training examples without modifying the location of the decision boundary. It effectively breaks the linear dependency between the number of SVs and the number of training examples.

2 Related work

2.1 Salient facts about SVMs

We focus now on the C-SVM applied to the two-class pattern recognition problem. See (Burges, 1998) for a concise reference. Given n training patterns x_i and their associated classes $y_i = \pm 1$, the SVM decision function is:

$$f(x) \;=\; \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + b^* \tag{2}$$

The coefficient α_i^* in (2) are obtained by solving a quadratic programing problem:

$$\begin{aligned} \alpha^* \quad &= \quad \arg\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &\text{subject to } \forall i,\ 0 \le \alpha_i \le C \text{ and } \sum_i \alpha_i y_i = 0 \end{aligned} \tag{3}$$

This optimization yields three categories of training examples depending on α_i^*. Within each category, the possible values of the margins $y_i f(x_i)$ are prescribed by the Karush-Kuhn-Tucker optimality conditions.

- Examples such that $\alpha_i^* = C$ are called *bouncing SVs* or *margin errors* and satisfy $y_i f(x_i) < 1$. The set of bouncing SVs includes all training examples misclassified by the SVM, i.e. those which have a negative margin $y_i f(x_i) < 0$.
- Examples such that $0 < \alpha_i^* < C$ are called *ordinary SVs* and satisfy $y_i f(x_i) = 1$.
- Examples such that $\alpha_i^* = 0$ satisfy relation $y_i f(x_i) > 1$. These examples play no role in the SVM decision function (2). Retraining after discarding these examples would still yield the same SVM decision function (2).

These facts provide some insight into Steinwart's result (1). The SVM decision function, like any other decision rule, must asymptotically misclassify at least $\mathcal{B}n$ examples, where $\mathcal{B}$ is the Bayes risk. All these examples must therefore become bouncing SVs.

To illustrate dependence on the Bayes risk, we perform a linear classification task in two dimensions under varying amount of class overlap. The class distributions were uniform on a unit square with centers c_1 and c_2. Varying the distance between c_1 and c_2 allows us to control the Bayes risk. The results are shown in figure 1.

2.2 A posteriori reduction of the number of SVs.

Several techniques aim to reduce the prediction complexity of SVMs by expressing the SVM solution (2) with a smaller kernel expansion. Since one must compute the SVM solution before applying these post-processing techniques, they are not suitable for reducing the complexity of the training stage.

Reduced Set Construction. Burges (Burges, 1996) proposes to construct new patterns z_j in order to define a compact approximation of the decision function (2). Reduced set construction usually involves solving a non convex optimization problem and is not applicable on arbitrary inputs such as graphs or strings.

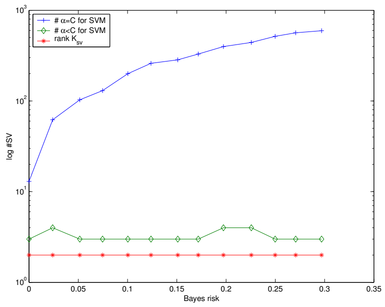

Figure 1: Effect of noise on the number of support vectors. The number of ordinary SVs stays almost constant where as the number of bouncing SVs grows. Additional support vectors do not gain information as indicated by the rank of the kernel matrix. See section 2.1.

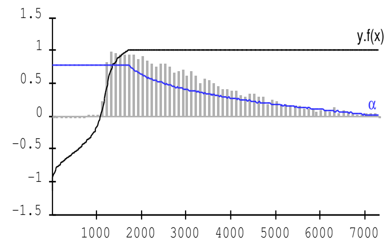

Figure 2: Histogram of SVs selected by the ℓ_1 penalization method on the MNIST 3-8 discrimination task. The initial SVs have been ordered on the x-axis by increasing margin $yf(x)$ and decreasing α. See last paragraph in section 2.2.

Reduced Set Selection. The set of basis[2] functions $K(x_i,\cdot)$ associated with the SVs x_i do not necessarily constitute a linearly independent family. The same decision function $f(\cdot)$ can then be expressed by multiple linear combination of the functions $K(x_i,\cdot)$. Reduced set selection methods attempt to select a subset of the SVs that is sufficient to express the SVM decision function. For instance, (Downs, Gates and Masters, 2001) propose to compute the row echelon form of the kernel matrix and discard SVs that lead to zero rows. This approach maintains the original SVM decision function.

In contrast, the ℓ_1 penalization method suggested in (Schölkopf and Smola, 2002, sect. 18.4.2) simply attempts to construct a sufficiently good approximation of the original SVM decision function by solving

$$\arg\min_{\beta} \left\| \sum_i \alpha_i^* y_i K(x_i,\cdot) - \sum_i \beta_i y_i K(x_i,\cdot) \right\|_{\mathcal{K}}^2 + \lambda \sum_i |\beta_i| \tag{4}$$

where parameter λ trades accuracy versus sparsity, and $\|\cdot\|_{\mathcal{K}}$ denotes the Reproducing Kernel Hilbert Space norm (Schölkopf and Smola, 2002, definition 2.9). Simplifying expression (4) yields a numerically tractable quadratic programming problem.

Which examples are selected? We have investigated the ℓ_1 penalization method (4) as follows. We train a first SVM to discriminate digits 3 and 8 on the MNIST dataset (see section 4.2) after randomly swapping 10% of the class labels in the training set. We then select a subset of the resulting support vectors using the ℓ_1 penalization method.

Choosing λ is quite difficult in practice. To evaluate the accuracy of the procedure, we train a second SVM on the selected vectors, compare its recognition accuracy with that of the first SVM. This was best achieved by enforcing the constraint $\beta_i \geq 0$ in (4) because the second SVM cannot return an expansion with negative coefficients.

Figure 2 shows the histogram of selected SVs. The initial support vectors have been ordered on the x-axis by increasing values of $y_i f(x_i)$, and, in the case of margin SVs, by decreasing values of α_i. The selected SVs includes *virtually no misclassified SVs*, but instead concentrates on *SVs with large* α_i.

This result suggests that simple pre-processing methods might indicate which training examples are really critical for SVM classification.

[2] We use the customary name *basis functions* despite linear dependence…

2.3 Training set editing techniques

We now consider techniques for reducing the set of training examples before running a training algorithm. Reducing the amount of training data is indeed an obvious way to reduce the complexity of training. Quantization and clustering methods might be used to achieve this goal. These methods however reduce the training data without considering the loss function of interest, and therefore sacrifice classification accuracy. We focus instead on *editing techniques*, i.e. techniques for discarding selected training examples with the aim of achieving similar or better classification accuracy.

Two prototypical editing techniques, MULTIEDIT and CONDENSE, have been thoroughly studied (Devijver and Kittler, 1982, chapter 3) in the context of the nearest neighbor (1-NN) classification rule.

Removing interior examples. The CONDENSE algorithm was first described by (Hart, 1968). This algorithm selects a subset of the training examples whose 1-NN decision boundary still classifies correctly all of the initial training examples:

Algorithm 1 (CONDENSE).

1. *Select a random training example and put it in set R.*
2. *For each training example $i = 1, \ldots, n$: classify example i using the 1-NN rule with set R as the training set, and insert it into R if it is misclassified.*
3. *Return to step 2 if R has been modified during the last pass.*
4. *The final contents of R constitute the condensed training set.*

This is best understood when both classes form homogeneous clusters in the feature space. Algorithm 1 discards training examples located in the interior of each cluster.

This strategy works poorly when there is a large overlap between the pattern distributions of both classes, that is to say when the Bayes risk $\mathcal{B}$ is large. Consider for instance a feature space region where $P(y = +1 \mid x) > P(y = -1 \mid x) > 0$. A small number of training examples of class $y = -1$ can still appear in such a region. We say that they are located on the wrong side of the Bayes decision boundary. Asymptotically, all such training examples belong to the condensed training set in order to ensure that they are properly recognized as members of class $y = -1$.

Removing noise examples. The *Edited Nearest Neighbor* rule (Wilson, 1972) suggests to first discard all training examples that are misclassified when applying the 1-NN rule using all $n - 1$ remaining examples as the training set. It was shown that removing these examples *improves* the asymptotic performance of the nearest neighbor rule. Whereas the 1-NN risk is asymptotically bounded by $2\mathcal{B}$, the Edited 1-NN risk is asymptotically bounded by $1.2\,\mathcal{B}$, where $\mathcal{B}$ is the Bayes risk.

The MULTIEDIT algorithm (Devijver and Kittler, 1982, section 3.11) asymptotically discards *all* the training examples located on the wrong side of the Bayes decision boundary. The asymptotic risk of the multi-edited nearest neighbor rule is the Bayes risk $\mathcal{B}$.

Algorithm 2 (MULTIEDIT).

1. *Divide randomly the training data into s splits $\mathcal{S}_1, \ldots, \mathcal{S}_s$. Let us call f_i the 1-NN classifier that uses $\mathcal{S}_i$ as the training set.*
2. *Classify all examples in $\mathcal{S}_i$ using the classifier $f_{(i+1) \bmod s}$ and discard all misclassified examples.*
3. *Gather all the remaining examples and return to step 1 if any example has been discarded during the last T iterations.*
4. *The remaining examples constitute the multiedited training set.*

By discarding examples located on the wrong side of the Bayes decision boundary, algorithm MULTIEDIT constructs a new training set whose apparent distribution has the same Bayes decision boundary as the original problem, but with Bayes risk equal to 0. Devijver and Kittler claim that MULTIEDIT produces an ideal training set for CONDENSE.

Algorithm MULTIEDIT also discards a some training examples located on the right side of Bayes decision boundary. Asymptotically this makes no difference. When the training set size is limited, this can have a negative impact on the error rates.

2.4 Editing algorithms and SVMs

Training examples recognized with high confidence usually do not appear in the SVM solution (2) because they do not become support vectors. Intuitively, SVMs display the properties of the CONDENSE algorithm. On the other hand, noise examples always become support vectors. In that respect, SVMs lack the properties of the MULTIEDIT algorithm.

> This contribution is an *empirical* attempt to endow SVMs with the properties of the MULTIEDIT algorithm. The resulting algorithm breaks the linear relation between number of SVs and number of examples.

Of course, the mathematical proofs for the properties of MULTIEDIT or CONDENSE depend of the specific nature of the 1-NN classification rule. Yet the algorithms themselves could be identically defined for any classification rule. This suggests (but does not prove) that the validity of these algorithms might extend to other classifiers. Further comfort comes from the knowledge that a SVM with the RBF kernel and without bias term[3] implements the 1-NN rule when the RBF radius tends to zero.

3 Cross-Training

In this section we introduce the Cross-Training SVM, loosely inspired by the training set editing algorithms of section 2.3. Cross-Training begins with partitioning the training set randomly into s subsets of size r and training independent SVM on each subset $\mathcal{S}_i$. The decision functions of these SVMs are then used to discard two types of training examples, namely those which are confidently recognized (as in CONDENSE), and those with are misclassified (as in MULTIEDIT). A final SVM is trained using the remaining examples.

Algorithm 3 (CROSSTRAINING).

1. *Split the data into s non-overlapping subsets of equal size, $\mathcal{S}_1, \dots, \mathcal{S}_s$.*
2. *Train s independent SVMs $f_1, \dots, f_s$ using each of the subsets as the training set.*
3. *For each training example (x_i, y_i) compute the average margin $m_i = \frac{1}{s}\sum_{r=1}^{s} y_i f_r(x_i)$.*
4. *Discard all training examples such that $m_i < 0$ or $m_i > 1$.*
5. *Train a final SVM on the remaining training examples.*

We call this method Cross-Training because each independent random split is used to make predictions on the remaining data, similar to Cross-Validation.

Selecting parameter C. Cross-Training also provides an elegant way to set the SVM parameter C for both the first stage SVMs and the second stage SVM.

1. We believe that the first stage SVMs should be made as accurate as possible. For each considered value of C, we train each first stage SVMs on its specific subset, and evaluate its error rate using the remaining $s-1$ subsets as testing examples. We choose the value of C that yields the smallest error rate.
2. The apparent distribution of the edited training set is meant to have Bayes risk 0. Therefore the second stage SVM is a Hard-Margin SVM ($C = \infty$).

---

[3] We mean a SVM whose solution (2) does not include a bias parameter b^*. This is equivalent to dropping the equality constraint in the quadratic problem (3). A more complete discussion on the impact of the bias parameter is clearly beyond the scope of this paper.

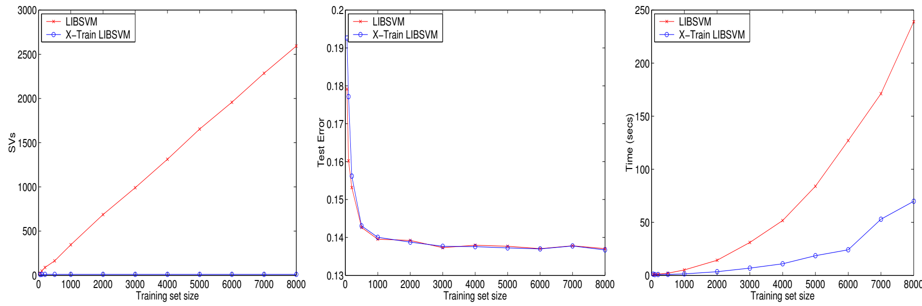

Figure 3: Comparing LIBSVM and Cross-Training on a toy problem of two Gaussian clouds for increasing number of training points. Cross-Training gives a constant number of support vectors (left figure) for increasing training set size, whereas in LIBSVM the number of support vectors increases linearly. The error rates behave similarly (middle figure), and Cross-Training gives an improved training time (right figure). See section 4.1.

Relation with robust statistics. This editing stage of Cross-Training implicitly modifies the SVM loss function in a way that relates to robust statistics. Editing alters the apparent distribution of training examples such that the class distributions $P(x \mid y = 1)$ and $P(x \mid y = -1)$ no longer overlap. If the class distributions were known, this could be done by trimming the tails of the class distributions. A similar effect could be obtained by altering the SVM loss function (the hinge loss) into a more robust loss function that gives less weight to examples with negative margin. This is however difficult to tune, and can lead to a non convex optimization problem.

4 Experiments

4.1 Toy Experiments

We first constructed artificial data, by generating two classes from two Gaussian clouds in 10 dimensions with means $(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$ and $(-1, -1, -1, -1, -1, 0, 0, 0, 0, 0)$ and standard deviation 4. We trained a linear SVM for differing amounts of training points, selecting C via cross validation. We compare the performance of LIBSVM[4] with Cross-Training using LIBSVM with $s = 5$, averaging over 10 splits. The results given in figure 3 show a reduction in SVs and computation time using Cross-Training, with no loss in accuracy.

4.2 MNIST experiments

Our second experiment involves the discrimination of digits 3 and 8 in the MNIST[5] database. Artificial noise was introduced by swapping the labels of 0%, 5%, 10% and 15% of the examples. There are 11982 training examples and 1984 testing examples. All experiments were carried out using LIBSVM's ν-SVM (Chang and Lin, 2001) with the RBF kernel ($\gamma = 0.005$). Cross-Training was carried out by splitting the 11982 training examples into 5 subsets. Figure 4 reports our results for various amounts of label noise. The number of SVs (left figure) increases linearly for the standard SVM and stays constant for the Cross-Training SVM. The test errors (middle figure) seem similar. Since our label noise is artificial, we can also measure the misclassification rate on the unmodified testing set (right figure). This measurement shows a slight loss of accuracy without statistical significance.

[4] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
[5] `http://yann.lecun.com/exdb/mnist`

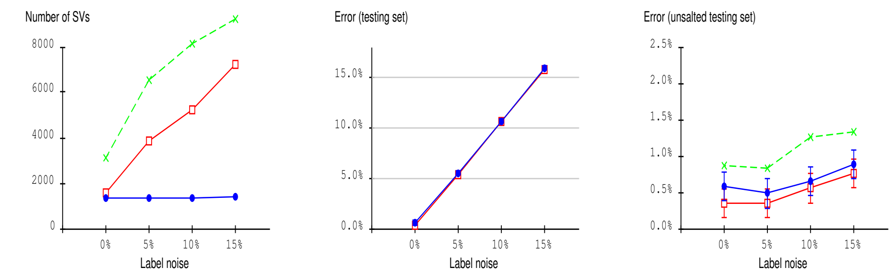

Figure 4: Number of SVs (left figure) and test error (middle figure) for varying amounts of label noise on the MNIST 3-8 discrimination task. In all graphs, the x-axis shows the amount of label noise, the white squares correspond to the standard SVM, the black circles correspond to the Cross-Training SVM, and the crosses correspond to bagging the 5 first stage Cross-Training SVMs. The last graph (right figure) shows the test error measured without label noise. See section 4.2.

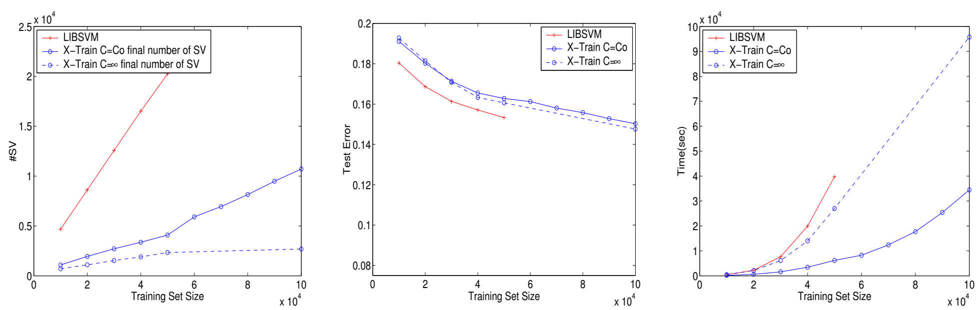

Figure 5: Cross-Training vs standard SVM training for varying training set sizes on the forest cover type database. See section 4.3.

4.3 Forest experiments

Finally, we apply our algorithm to a large scale problem. The UCI Forest Cover Type dataset[6] contains 581012 observations with 54 attributes. All attributes were rescaled in range $[-1, 1]$. We randomly selected 50000 testing examples. The remaining observations were used as training examples. Cross-Training was performed with $s = 5$ splits. SVM parameters were determined by 5-fold cross-validation on 10000 training examples ($C_0 = 150$, RBF kernel, $\gamma = 4$). The full SVM and Cross-Training first stage SVMs were trained with $C = C_0$. The Cross-Training second stage SVM was trained with both $C = \infty$ and $C = C_0$ because LIBSVM scales poorly when $C = \infty$ despite the small number of SVs[7].

Figure 5 compares the number of SVs, the error rates, and the training times for plain SVMs, Cross-Training with $C = C_0$ in the second stage, and Cross-Training with hard margin ($C = \infty$) in the second stage for training sets of sizes 10000 to 100000. Because of long computing times, we do not report plain SVM results beyond 50000 examples. With the hard-margin second stage, Cross-Training drastically reduces the number of SVs, while sacrificing less than 1% on accuracy.

---

[6] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/covtype`

[7] See also the anomaly in figure 3 (right).

5 Discussion

We have introduced a simple Cross-Training method for breaking the linear relationship between number of support vectors and number of examples. This method sharply reduces both the training and recognition time. Cross-training apparently causes a minor loss of accuracy, comparable to that of reduced set methods (Burges, 1996). On the other hand, Cross-training provides a practical means to use much larger training sets.

This work raises interesting theoretical questions regarding the significance of training set editing ideas in the context of SVMs. With better theoretical understanding should come better algorithms.

Finally this work accelerates SVMs with ideas that are orthogonal to those presented in (Graf et al., 2004). Combining the two approaches will provide the means to run SVMs on training sets of unprecedented scale.

6 Acknowledgement

We thank Hans Peter Graf, Eric Cosatto and Vladimir Vapnik for their advice and support. Part of this work was funded by NSF grant CCR-0325463.

References

Burges, C. J. C. (1996). Simplified Support Vector Decision Rules. In Saitta, L., editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, San Mateo, CA. Morgan Kaufmann.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Chang, C.-C. and Lin, C.-J. (2001). Training ν-Support Vector Classifiers: Theory and Algorithms. *Neural Computation*, 13(9):2119–2147.

Devijver, P. and Kittler, J. (1982). *Pattern Recogniton, A statistical approach*. Prentice Hall, Englewood Cliffs.

Downs, T., Gates, K. E., and Masters, A. (2001). Exact Simplification of Support Vector Solutions. *Journal of Machine Learning Research*, 2:293–297.

Graf, H., Cosatto, L., Bottou, L., and Durdanovic, I. (2004). Parallel Support Vector Machines: The Cascade SVM. submitted to NIPS*2004.

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transasctions on Information Theory*, (14):515–516.

Joachims, T. (1999). Making Large–Scale SVM Learning Practical. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA. MIT Press.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA.

Steinwart, I. (2004). Sparseness of Support Vector Machines—Some Asymptotically Sharp Bounds. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag, New York.

Wilson, D. L. (1972). Asymptotic properties of the nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2:408–420.