LETTER **Communicated by John Platt**

A Parallel Mixture of SVMs for Very Large Scale Problems

Ronan Collobert
collober@idiap.ch and collober@iro.umontreal.ca
Dalle Molle Institute for Perceptual Artificial Intelligence, 1920 Martigny, Switzerland, and Université de Montréal, DIRO, Montréal, Québec, Canada

Samy Bengio
bengio@idiap.ch
Dalle Molle Institute for Perceptual Artificial Intelligence, 1920 Martigny, Switzerland

Yoshua Bengio
bengioy@iro.umontreal.ca
Université de Montréal, DIRO, Montréal, Québec, Canada

Support vector machines (SVMs) are the state-of-the-art models for many classification problems, but they suffer from the complexity of their training algorithm, which is at least quadratic with respect to the number of examples. Hence, it is hopeless to try to solve real-life problems having more than a few hundred thousand examples with SVMs. This article proposes a new mixture of SVMs that can be easily implemented in parallel and where each SVM is trained on a small subset of the whole data set. Experiments on a large benchmark data set (Forest) yielded significant time improvement (time complexity appears empirically to locally grow linearly with the number of examples). In addition, and surprisingly, a significant improvement in generalization was observed.

1 Introduction

Recently a lot of work has been done around support vector machines (Vapnik, 1995), mainly due to their impressive generalization performances on classification problems when compared to other algorithms such as artificial neural networks (Cortes & Vapnik, 1995; Osuna, Freund, & Girosi, 1997). However, SVMs need resources that are at least quadratic on the number of training examples in order to solve a quadratic optimization problem, and it is thus hopeless to try to solve problems having millions of examples using classical SVMs.

In order to overcome this drawback, we propose in this article to use a mixture of several SVMs, each of them trained on only a part of the data set. The idea of an SVM mixture is not new, although previous attempts such as Kwok's article (1998) on support vector mixtures trained

the SVMs not on part of the data set but on the whole data set and hence could not overcome the time complexity problem for large data sets. We propose here a simple method to train such a mixture and will show that in practice, this method is much faster than training only one SVM and leads to results that are at least as good as one SVM. We conjecture that the training time complexity of the proposed approach with respect to the number of examples is subquadratic for large data sets. Moreover this mixture can be easily parallelized, which could improve the training time significantly.

In the next section, we briefly introduce the SVM model for classification. In section 3, we present our mixture of SVMs, followed in section 4 by some comparisons to related models. In section 5, we show some experimental results, first on a toy data set and then on a large real-life data set. A brief conclusion follows.

2 Introduction to Support Vector Machines

SVMs (Vapnik, 1995) have been applied to many classification problems, generally yielding good performance compared to other algorithms. The decision function is of the form

$$y = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right), \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the d-dimensional input vector of a test example, $y \in \{-1, 1\}$ is a class label, $\boldsymbol{x}_i \in \mathbb{R}^d$ is the input vector for the ith training example, N is the number of training examples, $K(\boldsymbol{x}, \boldsymbol{x}_i)$ is a positive definite kernel function, and $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}$ and b are the parameters of the model. Training an SVM consists of finding $\boldsymbol{\alpha}$ that minimizes the objective function

$$Q(\boldsymbol{\alpha}) = -\sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2.2}$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.3}$$

and

$$0 \leq \alpha_i \leq C \; \forall i. \tag{2.4}$$

The kernel $K(\boldsymbol{x}, \boldsymbol{x}_i)$ can have different forms, such as the radial basis function (RBF),

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right) \tag{2.5}$$

with parameter σ.

Therefore, to train an SVM, we need to solve a quadratic optimization problem where the number of parameters is N. This makes the use of SVMs for large data sets difficult: computing $K(x_i, x_j)$ for every training pair would require $O(N^2)$ computation, and solving may take up to $O(N^3)$. Note, however, that current state-of-the-art algorithms appear to have a training time complexity scaling much closer to $O(N^2)$ than $O(N^3)$ (Collobert & Bengio, 2001; Joachims, 1999; Platt, 1999).

3 Mixtures of SVMs

In this section, we introduce a new type of mixture of SVMs. The proposed model should minimize the following cost function,

$$C = \sum_{i=1}^{N}\left[h\left(\sum_{m=1}^{M} w_m(\boldsymbol{x}_i)s_m(\boldsymbol{x}_i)\right) - y_i\right]^2, \tag{3.1}$$

where M is the number of experts in the mixture, $s_m(\boldsymbol{x}_i)$ is the output of the mth expert given input $\boldsymbol{x}_i$, $w_m(\boldsymbol{x}_i)$ is the weight for the mth expert given by a "gater" module also taking $\boldsymbol{x}_i$ in input, and h is a transfer function that could be, for example, the hyperbolic tangent for classification tasks. Here, each expert is an SVM, and we took a neural network for the gater in our experiments.

To train this model, we propose a very simple algorithm:

1. Divide the training set into M random subsets of size near N/M.
2. Train each expert separately over one of these subsets.
3. Keeping the experts fixed, train the gater to minimize the cost, equation 3.1, on the whole training set.
4. Reconstruct M subsets. For each example $(\boldsymbol{x}_i, y_i)$,
   - Sort the experts in descending order according to the values $w_m(\boldsymbol{x}_i)$.
   - Assign the example to the first expert in the list that has fewer than $(N/M+c)$ examples,[1] in order to ensure a balance between the experts.

[1] Where c is a small, positive constant.

5. If a termination criterion is not fulfilled (such as a given number of iterations or a validation error going up), go to step 2.

Note that step 2 of this algorithm can be easily implemented in parallel as each expert can be trained separately on a different computer. Note also that step 3 can be an approximate minimization (as usually done when training neural networks).

4 Related Models

The idea of mixture models is quite old and has given rise to very popular algorithms, such as the well-known mixture of experts (Jacobs, Jordan, Nowlan, & Hinton, 1991), where the cost function is similar to equation 3.1 but where the gater and the experts are trained, using gradient descent or expectation maximization, on the whole data set (and not subsets) and their parameters are trained simultaneously. Hence, such an algorithm is quite demanding in terms of resources when the data set is large if training time scales like $O(N^p)$ with $p > 1$.

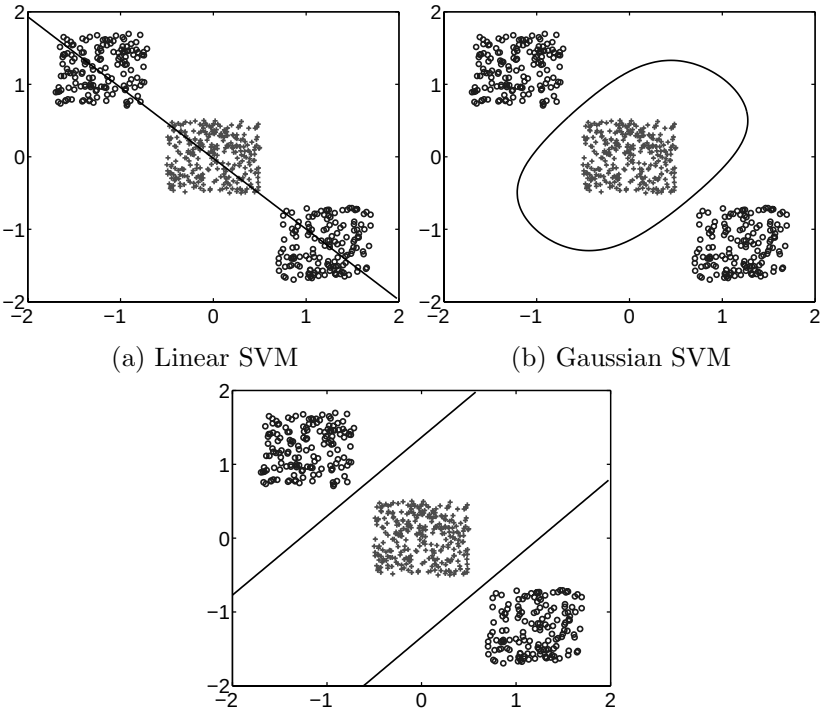
In the more recent SVM model (Kwok, 1998), the author shows how to replace the experts (typically neural networks) by SVMs and gives a learning algorithm for this model. Once again, the resulting mixture is trained jointly on the whole data set, and hence does not solve the quadratic barrier when the data set is large.

In another divide-and-conquer approach (Rida, Labbi, & Pellegrini, 1999), the authors propose to divide the training set using an unsupervised algorithm to cluster the data (typically a mixture of gaussians), train an expert (such as an SVM) on each subset of the data corresponding to a cluster, and finally recombine the outputs of the experts. Here, the algorithm does indeed separately train the experts on small data sets, like the present algorithm, but there is no notion of a loop reassigning the examples to experts according to the prediction made by the gater of how well each expert performs on each example. Our experiments suggest that this element is essential to the success of the algorithm.

5 Experiments

In this section, we present two sets of experiments comparing the new mixture of SVMs to other machine learning algorithms.

5.1 A Toy Problem. In the first series of experiments, we tested the mixture on an artificial toy problem where we generated 1000 training examples and 10,000 test examples. The problem had two nonlinearly separable classes and two input dimensions. Figure 1 shows the decision surfaces obtained first by a linear SVM, then by a gaussian SVM, and finally by the



(a) Linear SVM (b) Gaussian SVM

(c) Mixture of two linear SVMs

Figure 1: Comparison of the decision surfaces obtained by (a) a linear SVM, (b) a gaussian SVM, and (c) a linear mixture of two linear SVMs, on a two-dimensional classification toy problem.

proposed mixture of SVMs. Moreover, in the proposed mixture, the gater was a simple linear function and there were two linear SVMs in the mixture. This artificial problem thus shows clearly that the algorithm seems to work and is able to combine, even linearly, very simple models in order to produce a nonlinear decision surface.

5.2 A Large-Scale Realistic Problem: Forest. For a more realistic problem, we did a series of experiments on part of the UCI Forest data set.[2] Since this is a classification problem with seven classes, we modified it in a binary classification problem where the goal was to separate class 2 from the other

[2] The Forest data set is available on the UCI web site at the following address: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/covtype/covtype.info.

Table 1: Comparison of Performances.

	Train	Valid	Test	Time (minutes)		Total Number of
	Error (%)			1 cpu	50 cpu	Support Vectors
One MLP[a]	17.56	18.12	18.15	12		100
One SVM	16.03	16.68	16.76	3231		42,451
Uniform SVM mixture[b]	19.69	19.90	20.31	85	2	52,846
Gated SVM mixture	5.91	8.90	9.28	237	73	31,703

Notes: [a]100 Hidden Units. [b]The gater always output the same value for each expert.

six classes. The data set had more than 500,000 examples, and this enabled us to prepare a series of experiments as follows:

- We kept a test set of 50,000 examples to compare the best mixture of SVMs to other learning algorithms.
- We used a validation set of 10,000 examples to select the best mixture of SVMs, varying the number of experts and the number of hidden units in the gater.
- We trained our models on different training sets, using from 100,000 to 400,000 examples.
- The mixtures had from 10 to 50 expert SVMs with gaussian kernel, and the gater was a multilayer perceptron (MLP) with between 25 and 500 hidden units.

Because the number of examples was quite large, we selected the internal training parameters such as the σ of the gaussian kernel of the SVMs or the learning rate of the gater using a held-out portion of the training set. We compared our models to:

- A single MLP, where the number of hidden units was selected by cross-validation between 25 and 250 units.
- A single SVM, where the parameter of the kernel was also selected by cross validation.
- A mixture of SVMs where the gater was replaced by a constant vector, assigning the same weight value to every expert.

Table 1 gives the results of a first series of experiments with a fixed training set of 100,000 examples. We selected among the variants of the gated SVM mixture not only using performance on the validation set but also taking into account the time to train the model. The selected model had 50 experts and a gater with 150 hidden units. A model with 500 hidden units would have given a performance of 8.1% over the test set but would have taken 621 minutes on one machine (and 388 minutes on 50 machines). As

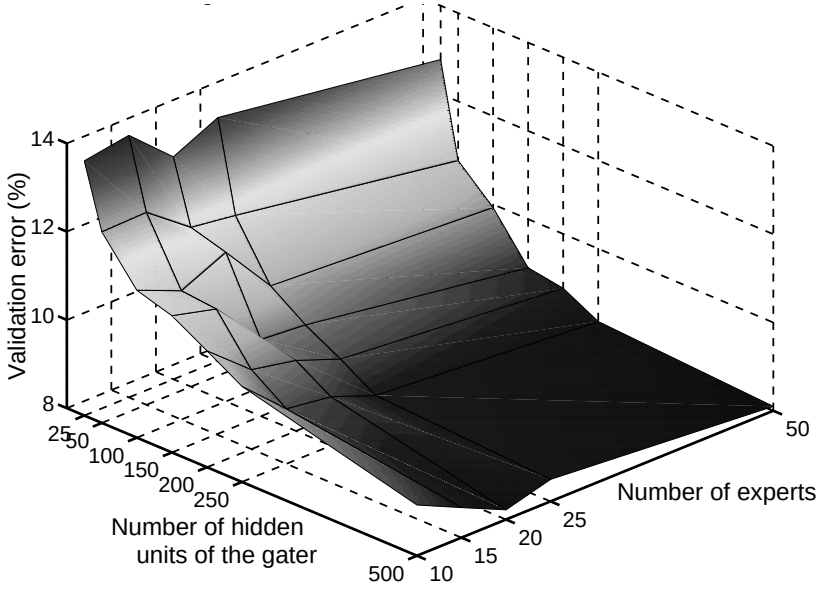


Figure 2: Comparison of the validation error of different mixtures of SVMs with various numbers of hidden units and experts.

can be seen, the gated SVM outperformed all models in terms of training, validation, and test error. It was also much faster, even on one machine, than the SVM, and since the mixture could easily be parallelized (each expert can be trained separately), we also give the time it took to train on 50 machines. It is interesting to note that the total number of support vectors of the gated SVM was fewer than the number of support vectors of the SVM. In a first attempt to understand these results, we can at least say that the power of the model lies not only on the gater, since a single MLP had quite bad performance on the test set, not only because we used SVMs, since a single SVM was not as good as the gated mixture, and not only because we divided the problem into many subproblems since the uniform mixture also performed badly. It seems to be a combination of all these elements.

We also did a series of experiments in order to see the influence of the number of hidden units of the gater as well as the number of experts in the mixture. Figure 2 shows the validation error of different mixtures of SVMs, where the number of hidden units varied from 25 to 500 and the number of experts varied from 10 to 50. There is a clear performance improvement when the number of hidden units is increased, while the improvement with

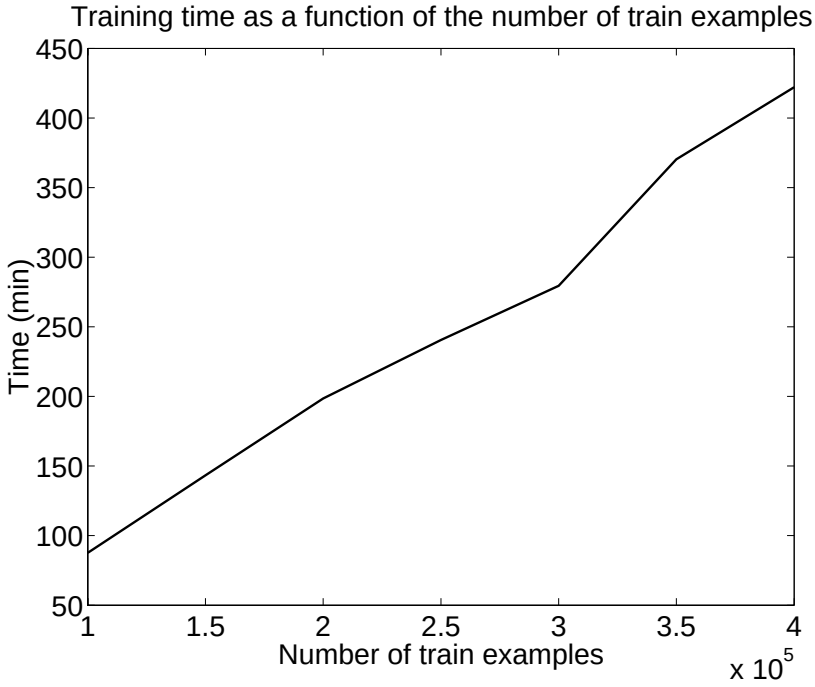


Figure 3: Comparison of the training time of the same mixture of SVMs (50 experts, 150 hidden units in the gater) trained on different training set sizes, from 100,000 to 400,000.

additional experts exists but is less clear. Note, however, that the training time also increases rapidly with the number of hidden units and slightly decreases with the number of experts if one uses one computer per expert.

In order to determine how the algorithm scaled with respect to the number of examples, we then compared the same mixture of experts (50 experts, 150 hidden units in the gater) on different training set sizes. Figure 3 shows the training time of the mixture of SVMs trained on training sets of sizes from 100,000 to 400,000. It seems that at least in this range and for this particular data set, the mixture of SVMs scales linearly with respect to the number of examples and not quadratically, like a classical SVM. It is interesting to see, for instance, that the mixture of SVMs was able to solve a problem of 400,000 examples in less than 7 hours (on 50 computers), while it would have taken more than one month to solve the same problem with a single SVM.

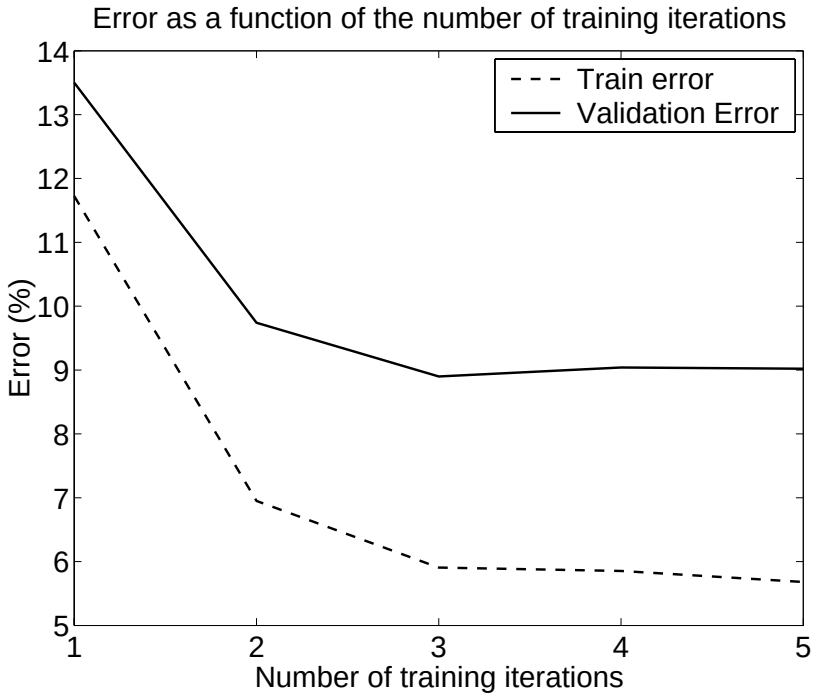


Figure 4: Comparison of the training and validation errors of the mixture of SVMs as a function of the number of training iterations.

Finally, Figure 4 shows the evolution of the training and validation errors of a mixture of 50 SVMs gated by an MLP with 150 hidden units, during five iterations of the algorithm. This should be convincing that the loop of the algorithm is essential to obtain good performance.

6 Conclusion

In this article, we have presented a new algorithm to train a mixture of SVMs that gave very good results compared to classical SVMs in terms of training time, generalization performance, or sparseness. Moreover, the algorithm appears to scale linearly with the number of examples, at least between 100,000 and 400,000 examples. Furthermore, the proposed algorithm has also been tested on another task with a similar number of examples and also yielded performance improvements.

These results are extremely encouraging and suggest that the proposed method could allow training SVM-like models for very large multimillion data sets in a reasonable time. If training of the neural network gater with

stochastic gradient takes time that grows much less than quadratically, as we believe to be the case for very large data sets (to reach a "good enough" solution), then the whole method is clearly subquadratic in training time with respect to the number of training examples.

Acknowledgments

R. C. thanks the Swiss National Science Foundation for financial support (project FN2100-061234.00). Y. B. thanks the NSERC funding agency and NCM^2 network for support.

References

Collobert, R., & Bengio, S. (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research, 1*, 143–160.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods*. Cambridge, MA: MIT Press.

Kwok, J. T. (1998). Support vector mixture for classification and regression problems. In *Proceedings of the International Conference on Pattern Recognition (ICPR)* (pp. 255–258). Brisbane, Queensland, Australia.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 130–136). San Juan, Puerto Rico.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods*. Cambridge, MA: MIT Press.

Rida, A., Labbi, A., & Pellegrini, C. (1999). Local experts combination trough density decomposition. In *International Workshop on AI and Statistics (Uncertainty '99)*. San Mateo, CA: Morgan Kaufmann.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. (2nd ed). New York: Springer-Verlag.

Received May 15, 2001; accepted August 13, 2001.