Support Vector Committee machines

Dominique Martinez, Gilles Millerioux

Support Vector Committee Machines

Dominique Martinez[(1)] and Gilles Millerioux[(2)]

(1) LORIA-CNRS, BP 239, 54506 Vandoeuvre, France
(2) CRAN-ESSTIN, Rue Jean Lamour, 54500 Vandoeuvre, France

Abstract. This paper proposes a mathematical programming framework for combining SVMs with possibly different kernels. Compared to single SVMs, the advantage of this approach is twofold: it creates SVMs with local domains of expertise leading to local enlargements of the margin, and it allows the use of simple linear kernels combined with a fixed boolean operation that is particularly well suited for building dedicated hardware.

1. Introduction

A challenging problem in machine learning is how to use a restricted amount of training data for constructing classifiers that generalize well in high-dimensional spaces [15]. Such a classifier in the linearly separable case is the so-called *optimal hyperplane* that provides the largest distance or *margin* from the separating hyperplane to the closest training vector. The basic idea behind the Support Vector Machines (SVMs) is first to transform the original data onto a high-dimensional space by using some fixed a priori mapping, and then find the optimal hyperplane in this feature space [4, 6]. The mapping onto the feature space is obtained using a given kernel representation of the inner product [1]. SVMs have shown remarkable generalization performance. However, one of their biggest limitations lies in the choice of the kernel. Possible ways to overcome this drawback would be to combine several SVMs with different kernels by using averaging techniques such as bagging [5] or boosting [8]. However, existing averaging methods are based on random modifications of the training set and, thus, are inefficient for stable classifiers such as SVMs for which a small change in the training set does not necessarily lead to a large change in the classifier. In this paper, we propose a mathematical programming framework for combining SVMs with possibly different kernels. The SVMs compete during training so that each SVM focuses on a particular subset of the data.

2. Classifier architecture

Let us consider a training set of n labelled observations $(\mathbf{x}_i, y_i)$, $i = 1 \cdots n$, where $\mathbf{x}_i \in R^d$ and $y_i \in \{-1, +1\}$. The basic idea behind the Support Vector Machines is to use some non-linear function Φ for mapping the input space onto a feature space so that the training set $(\Phi(\mathbf{x}_i), y_i)$, $i = 1 \cdots n$, becomes linearly separable. In this feature space, there exists a weight vector $\mathbf{w}$ and a bias b such that $y_i = \text{sgn}(\mathbf{w}.\Phi(\mathbf{x}_i) + b)$ for all i, where $\text{sgn}(a) = +1$ if $a \geq 0$ and -1 otherwise. The optimal separating hyperplane is the one that maximizes the margin $2/||\mathbf{w}||$ or, equivalently, minimizes the norm of $\mathbf{w}$ subject to the constraint that $y_i(\mathbf{w}.\Phi(\mathbf{x}_i) + b) \geq 1$ for all i (see e.g. [15]). Instead of having a unique fixed a priori function Φ for mapping the entire data set, let us consider several non-linear functions Φ_k, with $k = 1 \cdots p$. The outputs $y_{ki} = \text{sgn}(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k)$ of the p 'units' represent the internal representation of the input pattern $\mathbf{x}_i$. The class label $y_i = \pm 1$ is obtained by decoding the binary string $(y_{1i} \cdots y_{pi})$ with some fixed boolean operation. A case of interest is the parity function that outputs +1 if the number of positive bits is odd and -1 otherwise. The resulting classifier, called the parity machine, can exactly separate any arbitrary dichotomy with a bounded number of units [10]. However, it is very noise sensitive since a change of any bit in the internal representation will switch the output. A more robust decoding rule is the majority vote that gives +1 if the number of positive bits exceeds the number of negative bits and -1 otherwise. The classifier known in the literature as a committee machine [12] can be written as $f(\mathbf{x}_i) = \mathrm{sgn}(\sum_{k=1}^{p} y_{ki})$.

3. Training – primal formulation

Training the committee machine consists of maximizing the margin for each unit, leading to the following optimization problem for all patterns $i = 1 \cdots n$ and units $k = 1 \cdots p$

$$\min_{\mathbf{w}_k, b_k, y_{ki}} \quad \frac{1}{2} \sum_{k=1}^{p} ||\mathbf{w}_k||^2 \quad \text{s.t.} \quad y_{ki}(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k) \geq 1 \quad \text{and} \quad y_i = f(\mathbf{x}_i) \qquad (1)$$

where $y_{ki} = \pm 1$ is the unknown desired output (internal representation) for the ith input pattern and the kth unit. Finding the optimal y_{ki}s is a difficult combinatorial credit assignment problem. A heuristic procedure, named *least action*, has been previously proposed for solving the credit assignment problem arising both in a committee machine [12] and in a parity machine [11]. Instead of optimizing the y_{ki}s directly, the least action algorithm considers that $y_{ki} = y_i$ for each unit k but determines which units are used for which subset of the training data. It provides a way of determining which of the units k should be regarded as responsable for generating the error when the response of the classifier is incorrect. Those units constitute a pool of candidate units in the sense that changing their response from -1 to $+1$ or vice versa tends to correct the overall output. For the parity machine, a change in any unit will switch the output $f(\mathbf{x}_i)$ so that all the units are candidate units. On the contrary, for the committee machine, only the units that give a response different to y_i are candidate units. The least action algorithm calls for the adjustment of the minimum number of candidate units to ensure a correct overall response $f(\mathbf{x}_i)$ and changes those for which the smallest perturbation of $y_{ki}(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k)$ is required to alter their output. Let $\mu_{ki} \geq 0$ be the perturbation required for switching the output of unit k in the presence of input pattern $\mathbf{x}_i$. These perturbations can be introduced into the optimization problem as slack variables indicating how much the constraints are violated. Then, (1) becomes

$$\min \quad \frac{1}{2} \sum_{k=1}^{p} ||\mathbf{w}_k||^2 + C \sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad y_i(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k) \geq 1 - \mu_{ki} \quad , \quad \mu_{ki} \geq 0 \qquad (2)$$

in which C is a parameter to be chosen by the user, a larger value corresponding to assigning a higher penalty to the errors ξ_i. For an input pattern $\mathbf{x}_i$, the error ξ_i is defined as the smallest sum of perturbations required to ensure a correct overall output. For instance, a pattern $\mathbf{x}_i$ is well classified by a committee machine composed of $p = 3$ units if at least two units have responses equal to y_i so that the error ξ_i is

$$\xi_i = \min \; \{ \; (\mu_{1i} + \mu_{2i}), (\mu_{1i} + \mu_{3i}), (\mu_{2i} + \mu_{3i}) \; \} \qquad (3)$$

4. Training – dual formulation

In order to use kernels, as it is the case for SVMs, we now consider the dual formulation of (2). For the sake of clarity, we just consider a particular case of a committee machine composed of $p = 3$ units. Note however that the results obtained easily extend to committee machines with an arbitrary number of units. In order to obtain a form amenable to a simple dual formulation, we rewrite the minimum error function as in [2, 3] and (3) now becomes

$$\xi_i = \lambda_{1i}(\mu_{1i} + \mu_{2i}) + \lambda_{2i}(\mu_{1i} + \mu_{3i}) + \lambda_{3i}(\mu_{2i} + \mu_{3i}) \tag{4}$$

$$\text{with } (\lambda_{1i}, \lambda_{2i}, \lambda_{3i}) = \begin{cases} (1,0,0) \text{ when } (\mu_{1i} + \mu_{2i}) \text{ is minimum} \\ (0,1,0) \text{ when } (\mu_{1i} + \mu_{3i}) \text{ is minimum} \\ (0,0,1) \text{ when } (\mu_{2i} + \mu_{3i}) \text{ is minimum} \end{cases} \tag{5}$$

For fixed $(\lambda_{1i}, \lambda_{2i}, \lambda_{3i})$, the solution of the optimization problem (2)-(4) is given by the saddle point of the Lagrangian

$$\begin{aligned} \mathcal{L}_P &= \frac{1}{2}\sum_{k=1}^{p=3} ||\mathbf{w}_k||^2 + C\sum_{i=1}^{n}\{\lambda_{1i}(\mu_{1i} + \mu_{2i}) + \lambda_{2i}(\mu_{1i} + \mu_{3i}) + \lambda_{3i}(\mu_{2i} + \mu_{3i})\} \\ &- \sum_{k=1}^{p=3}\sum_{i=1}^{n} \alpha_{ki}\{y_i(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k) - 1 + \mu_{ki}\} - \sum_{k=1}^{p=3}\sum_{i=1}^{n} r_{ki}\mu_{ki} \end{aligned} \tag{6}$$

where the α_{ki} are the Lagrange multipliers taking into account the inequality constraints in (2) and the $r_{ki} \geq 0$ are the Lagrange multipliers introduced to enforce positivity of the μ_{ki}. At the saddle point, the solution should satisfy the Karush-Kuhn-Tucker (KKT) conditions

$$\partial\mathcal{L}_P/\partial\mathbf{w}_k = 0 \Rightarrow \mathbf{w}_k = \sum_i \alpha_{ki} y_i \Phi_k(\mathbf{x}_i) \tag{7}$$

$$\partial\mathcal{L}_P/\partial b_k = 0 \Rightarrow \sum_i \alpha_{ki} y_i = 0 \tag{8}$$

$$\begin{aligned} \partial\mathcal{L}_P/\partial\mu_{1i} &= C(\lambda_{1i} + \lambda_{2i}) - \alpha_{1i} - r_{1i} = 0 \\ \partial\mathcal{L}_P/\partial\mu_{2i} &= C(\lambda_{1i} + \lambda_{3i}) - \alpha_{2i} - r_{2i} = 0 \\ \partial\mathcal{L}_P/\partial\mu_{3i} &= C(\lambda_{2i} + \lambda_{3i}) - \alpha_{3i} - r_{3i} = 0 \end{aligned} \tag{9}$$

As for SVMs, the optimal hyperplanes given by (7) are linear combinations of the vectors of the training set transformed by the mapping Φ_k. Those vectors $\mathbf{x}_i$, for which α_{ki} are nonzero, are called support vectors for $\mathbf{w}_k$. Replacing expressions of $\mathbf{w}_k$ in (6), and taking into account the other KKT conditions (8)-(9), one obtains

$$\mathcal{L}_D = \sum_{k=1}^{p}\sum_{i=1}^{n} \alpha_{ki} - \frac{1}{2}\sum_{k=1}^{p}\sum_{i,j=1}^{n} \alpha_{ki}\ \alpha_{kj}\ y_i\ y_j\ (\Phi_k(\mathbf{x}_i).\Phi_k(\mathbf{x}_j))$$

Note that $\mathcal{L}_D$ only requires the evaluation of dot products which can be replaced by simple kernels $K_k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi_k(\mathbf{x}_i).\Phi_k(\mathbf{x}_j))$. The optimization problem then becomes

$$\max_{\alpha_{ki}} \sum_{k=1}^{p}\sum_{i=1}^{n} \alpha_{ki} - \frac{1}{2}\sum_{k=1}^{p}\sum_{i,j=1}^{n} \alpha_{ki}\ \alpha_{kj}\ y_i\ y_j\ K_k(\mathbf{x}_i, \mathbf{x}_j) \tag{10}$$

subject to (8) and the following constraints

$$0 \leq \alpha_{1i} \leq C(\lambda_{1i} + \lambda_{2i}), 0 \leq \alpha_{2i} \leq C(\lambda_{1i} + \lambda_{3i}), 0 \leq \alpha_{3i} \leq C(\lambda_{2i} + \lambda_{3i}) \quad (11)$$

resulting from (9) and the fact that $r_{ki} \geq 0$. Assume that for a given input $\mathbf{x}_i$, $(\lambda_{1i}, \lambda_{2i}, \lambda_{3i}) = (1, 0, 0)$ from (5). The constraints (11) then become $0 \leq \alpha_{1i}, \alpha_{2i} \leq C$ and $\alpha_{3i} = 0$ meaning that the training data pair $(\mathbf{x}_i, y_i)$ is assigned to units 1 and 2 but not to unit 3. For the configurations $(\lambda_{1i}, \lambda_{2i}, \lambda_{3i}) = (0, 1, 0)$ and $(0, 0, 1)$, the training data pair is assigned to units $(1, 3)$ and $(2, 3)$, respectively. Hence, during training, the different SVMs compete for data so that each SVM focuses on a particular subset of the training data.

5. Practical algorithm and illustrative examples

Similarly to the support vector decision tree algorithm proposed in [3], we propose a simple descent algorithm which alternates between the primal and dual formulations until a local minimum is reached. The support vector committee machine algorithm iterates between the following two steps

1) solve the dual quadratic problem (10) subject to (8) and (11) for fixed λ_{ki}.

2) compute the μ_{ki} from the constraints $y_i(\mathbf{w}_k.\Phi_k(\mathbf{x}_i) + b_k) \geq 1 - \mu_{ki}$ of the primal problem (2) and the λ_{ki} from (5).

until there is no change in the λ_{ki}s (these were initialized randomly in the simulations below). Any kernel that a single SVM could have can be used, e.g. linear, polynomial $(\mathbf{x}_i.\mathbf{x}_j + 1)^{d_k}$ or RBF $\exp(-\gamma_k||\mathbf{x}_i - \mathbf{x}_j||^2)$. Polynomial kernels allow us to incorporate the Structural Risk Minimization principle into the support vector committee machine algorithm by increasing the order d_k of unit k that both makes large training errors and has a low VC dimension $h_k \approx R_k^2||\mathbf{w}_k||^2$ (with R_k being the radius of the minimum enclosing sphere [14] estimated on the subset of training data that has been assigned to unit k). Our program was written in C and uses a dual-active-set quadratic programming method originating from [9], implemented in [13], and available at http://www.isr.umd.edu/Labs/CACSE/FSQP/qld.c. We experimentally found this implementation both more efficient and numerically stable than the interior point quadratic solver available at http://svm.first.gmd.de.

Figure 1 shows the discriminant surface obtained with a support vector committee machine on the 2-bit parity discrimination problem. One of the advantages of the committee machine is the possibility of using simple linear kernels (see Fig. 1 left). Another solution, albeit equally effective, is obtained on this problem if the parity decoding rule is used instead of the majority (see Fig. 1 right). Changing the decoding rule only modifies the error term ξ_i of the primal problem and, consequently, the constraints (11) of the dual problem.

Figure 2 compares the discriminant surface obtained with a single SVM to the one obtained with a support vector committee machine on the Fisher Iris data discrimination problem [7]. The problem is linearly and non-linearly separable into the left and right region of the input space, respectively. The margin obtained with a unique kernel is infinitely small in the zero error case ($C = \infty$), see Fig. 2A. The only way to enlarge the margin in the linearly separable region is to tolerate errors in the non-linearly separable region by decreasing C, as indicated in Fig. 2B. On the contrary, the support vector committee machine allows us to enlarge the margin locally in the linearly separable region and still having all the training patterns correctly classified (see Fig. 2C). This is due to the fact that the different units of the committee competed for data during training so that each unit focused on a particular subset of the training data corresponding to the linearly and non-linearly regions of the input space. The VC argument described above was used to select the appropriate degrees for the polynomial kernels. The committee machine, depicted in Fig. 2D, has two linear kernels and yields an error probability, estimated at 0.087 by leaving one out, which is lower than the 0.15 and 0.10 error probability obtained with a single RBF kernel SVM and $C = \infty$ and 100, respectively. The solution obtained depends upon the initialization since the support vector committee machine algorithm stops at the first local minimum encountered. Although it is not garanteed to find the optimum solution in every case, the configuration depicted in Fig. 2D was obtained in more than 80% of the 500 simulations performed with random starting point. The number of iterations was 3.66 on average.

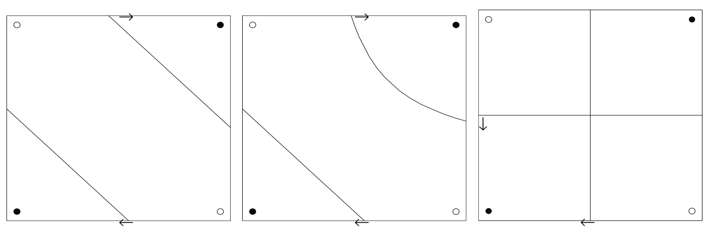

Figure 1: 2-bit parity discrimination problem. White data points have targets -1 and black data points have targets $+1$. The discriminant surfaces on the left and on the right have been obtained using support vector committee and parity machines with linear kernels, respectively. The discriminant surface in the middle is for a support vector committee machine with a linear kernel coupled with a polynomial kernel. Arrows indicate the positive sides of the elementary discriminants.

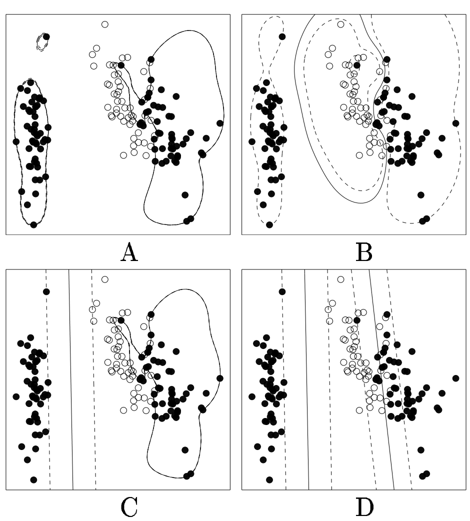

Figure 2: The Fisher Iris data discrimination problem represented on the first two principal component axes. White data points (target -1) are from the classes setosa and virginica. Black data points (target $+1$) are from the class versicolor. The discriminant surface is indicated with a full curve while the classification margin is delimited by the dashed curves. Note that full and dashed curves may be superimposed for margins of very small sizes. Figs A and B are for a single SVM having a RBF kernel ($\gamma = 1$) and optimized with $C = \infty$ and 100, respectively. Figs C and D are for a support vector committee machine optimized with two kernels and $C = \infty$. The kernels are different (linear and RBF) and identical (linear) in figure C and D, respectively.

6. Conclusion

In this paper, we presented a mathematical programming framework for combining SVMs with possibly different kernels. Compared to single SVMs, the advantage of this approach is twofold: it creates SVMs with local domains of expertise leading to local enlargements of the margin, and it allows the use of simple linear kernels combined with a fixed boolean operation (e.g. majority) that is particularly well suited for building dedicated VLSI circuits. Our approach can be related to the least action algorithm [11, 12] and to the support vector decision tree algorithm proposed in [2, 3]. During training, each SVM competes for data so as to focus on a particular subset of the training data. However, the solution obtained depends upon the initialization since the proposed algorithm stops at the first local minimum encountered. For initializing the algorithm better than randomly, we are investigating several lines of research including the use of unsupervised clustering techniques or the possibility of exploiting prior knowledge about the classification task to be learned.

References

[1] Aizerman, M.A., Braverman E.M. & Rozonoer, L.I. (1964) Theoretical foundations of the potential function method in pattern recognition learning, *Automation and remote control*, **25**, pp. 821-837.

[2] Bennett, K.P. & Blue, J.A. (1998) A Support Vector Machine Approach to Decision Trees, *In Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 2396-2401, Anchorage, Alaska.

[3] Blue, J.A. (1998) A Hybrid of Tabu Search and Local Descent Algorithms with Applications in Artificial Intelligence, *PhD Thesis*, Rensselaer Polytechnic Institute.

[4] Boser, B., Guyon, I. & Vapnik, V. (1992) A training algorithm for optimal margin classifiers, *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh ACM, pp. 144-152.

[5] Breiman, L. (1996) Bagging predictors, *Machine Learning*, **24**, pp. 123-140.

[6] Cortes, C. & Vapnik, V. (1995) Support vector networks, *Machine Learning*, **20**, pp. 1-25.

[7] Fisher,R.A. (1936) The use of multiple measurements in taxonomic problems, *Annual Eugenics*, **7**, Part II, pp. 179-188. (The Iris data are ftp available at ics.uci.edu:pub/machine-learning-databases).

[8] Freund, Y. (1995) Boosting a weak learning algorithm by majority, *Information and Computation*, **121**, pp. 256-285.

[9] Goldfarb, D. & Idnani, A. (1983) A numerically stable dual method for solving strictly convex quadratic programs, *Mathematical Programming*, **27**, pp. 1-33.

[10] Gordon, M. B. (1996) A convergence theorem for incremental learning with real-valued inputs, *Proc. of the IEEE Conference on Neural Networks*, pp. 381-386.

[11] Mitchison, G.J. & Durbin, R.M. (1989) Bounds on the learning capacity of some multi-layer networks, *Biological Cybernetics*, **60**, pp. 345-356.

[12] Nilsson N.J. (1969) Learning Machines, *McGraw-Hill.*

[13] Powell, M.J.D. (1983) ZQPCVX A fortran subroutine for convex quadratic programming, Technical report DAMTP/83/NA17, University of Cambridge (UK).

[14] B. Schölkopf and C. Burges and V. Vapnik (1995) Extracting support data for a given task, *Proc. First International Conference on Knowledge Discovery & Data Mining*, AAAI Press, Menlo Park, CA.

[15] Vapnik V. (1998) Statistical learning theory, *John Wiley & Sons, Inc.*