ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

Rosen's Projection Method for SVM Training

Jorge López, José R. Dorronsoro *

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

Abstract. In this work we will give explicit formulae for the application of Rosen's gradient projection method to SVM training that leads to a very simple implementation. We shall experimentally show that the method provides good descent directions that result in less training iterations, particularly when large precision is wanted. However, a naive kernelization may end up in a procedure requiring more KOs than SMO and further work is needed to arrive at an efficient implementation.

1 Introduction

Given a sample $\mathcal{S} = \{(X_i, y_i) : i = 1, \ldots, N\}$ with $y_i = \pm 1$, linear penalty SVM training seeks [1] to maximize the margin of a separating hyperplane by solving

$$\min \frac{1}{2}\|W\|^2 + C\sum \xi_i \;\text{ s. t. }\; y_i(W \cdot X_i + b) \geq 1 - \xi_i, i = 1, \ldots, N. \qquad (1)$$

However, a simpler dual problem is solved in practice, that of minimizing

$$W(\alpha) \;=\; \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_i \alpha_i \;\text{ s.t. }\; 0 \leq \alpha_i \leq C,\;\; \sum_i \alpha_i y_i = 0. \qquad (2)$$

The optimal weight W^o can be then written as $W^o = \sum \alpha_i^o y_i X_i$ and patterns for which $0 < \alpha_i^o < C$ are called non–bounded support vectors (SVs) while those for which $\alpha_i^o = C$ are called bounded SVs. The more widely used approach to solve (2) is decomposition methods, where one iteratively selects working sets made up of some α components while the others are kept fixed. The extreme case of this approach corresponds to Platt's SMO [2], where the working set has size 2 and the corresponding minimization problem can be solved analytically. This may lead to a large number of iterations being needed but, on the other hand, since the corresponding weight updates are very simple, the total number of kernel operations (KOs) may be reasonably small. Joachims' SVM–Light method [3] considers working sets of size $q = 2l$. When $q = 2$ it reduces to SMO, but for larger q an analytic approach may not be possible, and some numerical quadratic programming (QP) solver has to be used. When $q > 2$ SVMLight usually requires less iterations than SMO but, as they are costlier, one may end up with a larger number of KOs.

*With partial support of Spain's TIN 2007–66862 project and Cátedra UAM–IIC en Modelado y Predicción. The first author is kindly supported by FPU-MICINN grant reference AP2007–00142.

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

A simpler alternative with potentially larger gains could be gradient descent; however, the linear constraint $\sum_i \alpha_i y_i = 0$ precludes this approach as the gradient may not define a feasible descent direction. The linear constraint disappears if we consider an homogeneous classifier dropping the b term, but with the risk of the final classifiers being less accurate. To avoid this, we may try to modify the gradient to arrive at a feasible descent direction. A number of such gradient projection methods have been proposed in the QP literature; examples of this are the Spectral Projected Gradient or the Variable Projection methods; some of them have been applied to SVM training (see [4] and the references there). They are quite efficient procedures suitable in some cases for parallel implementations that can be exploited to train large scale SVMs. Still, they have a markedly numerical nature and, in particular, they seem to exploit only partially the very simple nature of the SVM constraints.

In this work we shall consider Rosen's gradient projection method, one of the earlier projection proposals for QP that we describe in section 2, and show in section 3 that its projection map can be expressed in closed form; while not too deep, the required computations are somewhat involved and cannot be given in a short paper. Thus, the overall procedure can be directly applied without recourse to numerical procedures. and results in an easy to implement, kernelizable method that provides better descent directions that, for instance, SMO; in particular, and as we shall illustrate in section 4, it leads to models with the same accuracy but requiring many less iterations to converge, particularly when large precision is wanted. On the other hand, a naive implementation might result in a larger number of KOs and, thus, in a slower method overall. Therefore, a careful analysis is needed for a successful implementation, an issue that requires further work and that we will not address here. The paper will end with a short discussion.

2 Rosen's Gradient Projection Method

We shall review here the main points of Rosen's method [5] for gradient projection (see [6], chapter 10 for more details), in which a symmetric projection matrix P is used to define a feasible projection $P\nabla f$ of the gradient ∇f. Notice first that since $P^2 = P$, it is easy to check that if $d = -P\nabla f$, then $\nabla f \cdot d = -\|d\|^2$. Thus, in order for d to be descent direction, it is enough that it be feasible. To construct P we will consider a slightly more general problem than SVM training, in which we want to solve

$$\min_{\alpha} \; f(\alpha) \;\; s.t. \;\; \Theta\alpha \leq \theta, \;\; \Phi\alpha = \phi, \tag{3}$$

where we assume α to be N–dimensional and that we have L and K inequality and equality constraints respectively, with corresponding matrices Θ and Φ and right-hand side vectors θ and ϕ. Let us assume that the current α is a feasible point for (3). We can separate the inequality constraints for α into L_1 binding ones and L_2 non-binding ones by partitioning Θ and θ as $\Theta^T = \left(\Theta_1^T, \Theta_2^T\right)$, $\theta^T = \left(\theta_1^T, \theta_2^T\right)$, with $\Theta_1\alpha = \theta_1$ and $\Theta_2\alpha < \theta_2$. Let us now

define the $N \times (L_1 + K)$ matrix $M^T = \left(\Theta_1^T, \Phi^T\right)$ that corresponds to the binding constraints at α. If M has full rank, MM^T is invertible and we shall use $P = I - M^T \left(MM^T\right)^{-1} M$ as the projection matrix. It is easy to see that d is also feasible, i.e., that $M\mathbf{d} = 0$, i.e., $\Theta_1\mathbf{d} = \mathbf{0}$ and $\Phi\mathbf{d} = \mathbf{0}$, for we have $M\mathbf{d} = -MP\nabla f(\alpha) = -M\left(I - M^T \left(MM^T\right)^{-1} M\right)\nabla f(\alpha) = \mathbf{0}$. Thus, $\mathbf{d}$ will define a descent direction provided it is not $\mathbf{0}$. However, this may very well be the case if we have already optimized on the binding components of α. If so, we have $0 = P\nabla f(\alpha) = \left(I - M^T \left(MM^T\right)^{-1} M\right)\nabla f(\alpha) = \nabla f(\alpha) + M^T\beta$, with $\beta = -\left(MM^T\right)^{-1} M\nabla f(\alpha)$. Now β is an $(L_1 + K)$–dimensional vector and consider its decomposition $\beta^T = \left(\mathbf{u}^T, \mathbf{w}^T\right)$ into an L_1–dimensional vector $\mathbf{u}$ and a K–dimensional one $\mathbf{w}$. Adding in the L_2–dimensional vector $\mathbf{v} = \mathbf{0}$, we can write the preceding as $0 = \nabla f(\alpha) + \Theta_1^T\mathbf{u} + \Theta_2^T\mathbf{v} + \Phi^T\mathbf{w}$, which is just the gradient of the Lagrangian of (3) associated to the multipliers $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$. Moreover, we obviously have $\mathbf{v}^t(\Theta_2\alpha - \theta_2) = 0$ and since the Θ_1 constraints are binding, $\mathbf{u}^t(\Theta_1\alpha - \theta_1) = 0$ also holds. Hence, if $\mathbf{u} \geq \mathbf{0}$, the current α, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ would satisfy the KKT conditions and we would have solved (3) .

On the other hand, if α does not solve (3), there will be at least one negative multiplier u_j which we can use to construct a slightly modified projection matrix $\widehat{P}$ such that $\widehat{\mathbf{d}} = -\widehat{P}\nabla f(\alpha)$ is improving and feasible. More precisely, we define $\widehat{\Theta}_1$ as the matrix obtained deleting in Θ_1 the row corresponding to u_j and set now $\widehat{M^T} = \left(\widehat{\Theta}_1^T, \Phi^T\right)$. Just as before we can construct now a new projection matrix $\widehat{P} = I - \widehat{M^T}\left(\widehat{M}\widehat{M^T}\right)^{-1}\widehat{M}$, and it turns out that $\widehat{\mathbf{d}} = -\widehat{P}\nabla f(\alpha)$ also defines an improving and feasible direction. The slightly long proof can be found in chapter 10 of [6]; a simpler one can be given in the concrete case of SVM training as a consequence of the explicit formulae for the P and $\widehat{P}$ projections that we present next.

3 Rosen's method for SVM Training

It is clear that the SVM minimization problem is a particular case of (3) and, in the notation of the previous section, the SVM box constraints can be written in terms of $\Theta^T = \left(-I_N^T, I_N^T\right)$ and $\theta^T = \left(\mathbf{0_N}^T, C\mathbf{1_N}^T\right)$ while for the equality constraint we use $\Phi^T = \mathbf{y}^T$ and $\phi = 0$. Here $I_N, \mathbf{0_N}$ and $\mathbf{1_N}$ stand for the N-dimensional identity matrix and the zero and one vectors. At a given feasible α we will have two kinds of binding constraints: a certain number N_0 of the form $\alpha_i = 0$ and a number N_C of the form $\alpha_i = C$. The corresponding Θ_1 and θ_1 matrix and vector become then $\Theta_1^T = \left(-\mathcal{I}_{N_0}^T, \mathcal{I}_{N_C}^T\right)$ and $\theta_1^{\mathbf{T}} = \left(\mathbf{0}_{N_0}^T, C\mathbf{1}_{N_C}^T\right)$; the notation $\mathcal{I}$ reflects the fact that $\mathcal{I}_{N_0}$ and $\mathcal{I}_{N_C}$ may no longer be identity matrices as their rows correspond to the canonical vectors associated to the non-support vectors and the bounded SVs respectively. In particular, $M^T = \left(\Theta_1^T, \Phi^T\right)$ has a dimension $N \times (N_0 + N_C + 1)$. Now it is easy to see that

$$MM^T = \begin{pmatrix} -\mathcal{I}_{N_0} \\ \mathcal{I}_{N_C} \\ \mathbf{y^T} \end{pmatrix} \begin{pmatrix} -\mathcal{I}_{N_0}^T & \mathcal{I}_{N_C}^T & \mathbf{y} \end{pmatrix} = \begin{pmatrix} I_{N_0} & 0_{N_0 N_C} & -\mathbf{y}_{N_0} \\ 0_{N_C N_0} & I_{N_C} & \mathbf{y}_{N_C} \\ -\mathbf{y}_{N_0}^T & \mathbf{y}_{N_C}^T & N \end{pmatrix},$$

where 0_{ab} denotes a zero $a \times b$ matrix. $(MM^T)^{-1}$ can be explicitly computed by block matrix inversion, after which we can arrive to $P' = M^T \left(MM^T\right)^{-1} M$ and then to the projection matrix $P = I_N - P'$. While the intermediate computations are somewhat involved, the final form of the P_{ij} components of P is very simple, namely, $P_{ij} = \delta_{ij} - y_i y_j / \mathcal{N}$ if $i, j \in \mathcal{J}$ and 0 otherwise, where $\mathcal{J}$ is the index set of the non–bounded support vectors, $\mathcal{N} = |\mathcal{J}|$ is the number of such SVs (i.e., those for which $0 < \alpha_i < C$) and δ_{ij} denotes Kronecker's delta. From this, the feasible descent direction $\mathbf{d}$ is easily obtained as $d_i = -\nabla f(\alpha)_i + \frac{y_i}{\mathcal{N}} \sum_{j \in \mathcal{J}} y_j \nabla f(\alpha)_j$ if $i \in \mathcal{J}$ and 0 otherwise.

When this procedure leads to a $\mathbf{d} = 0$, we have to remove from M a constraint associated to a negative multiplier u_i. If we let now $\mathcal{J}_0$ and $\mathcal{J}_C$ be the index sets of the non–support vectors and of bounded SVs respectively, the multiplier values can be shown to be

$$u_i = \left\{ \begin{array}{ll} \nabla f(\alpha)_i - \frac{y_i}{\mathcal{N}} \sum_{k \in \mathcal{J}} y_k \nabla f(\alpha)_k & \forall i \in \mathcal{J}_0, \\ -\nabla f(\alpha)_i + \frac{y_i}{\mathcal{N}} \sum_{k \in \mathcal{J}} y_k \nabla f(\alpha)_k & \forall i \in \mathcal{J}_C \end{array} \right\}. \tag{4}$$

We have to decide which negative mutiplier to choose; it can be shown (we omit the details) that the most negative one is also the one that violates most the aproximate KKT conditions at the current α. Assuming the chosen index to be $\hat{p}$, computations similar to the previous ones give now $\widehat{P}_{ik} = \delta_{ik} - y_i y_k / (1 + \mathcal{N})$ if $i \in \{\hat{p}\} \cup \mathcal{J}$ and 0 otherwise. Moreover, if $i \in \{\hat{p}\} \cup \mathcal{J}$,

$$\widehat{d_i} = -\nabla f(\alpha)_i + \frac{y_i}{1 + \mathcal{N}} \sum_{k \in \mathcal{J} \cup \{\hat{p}\}} y_k \nabla f(\alpha)_k \tag{5}$$

while it is 0 otherwise. Finally, once we have obtained the new descent direction $\mathbf{d}$, it is straightforward to compute the optimal dilation λ^* that minimizes $f(\alpha + \lambda \mathbf{d})$. Taking into account that $f(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i$, where $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i}$, we can write $f(\alpha + \lambda \mathbf{d})$ as a function Φ of λ, namely $\Phi(\lambda) = f(\alpha) + \lambda \left(\mathbf{w^T z} - s\right) + \frac{1}{2}\lambda^2 \|z\|^2$, where $z = \sum_i y_i d_i \mathbf{x_i}$ and $s = \sum_i d_i$. It is easy to see that $\Phi(\lambda)$ has a minimum at $\lambda' = -(\mathbf{w^T z} - s)/\|z\|^2$. Note that $\lambda' > 0$ is guaranteed, since $-\left(\mathbf{w^T z} - s\right) = -\nabla f \cdot d > 0$. However, this λ' must be clipped so that the box constraints $0 \leq \alpha_i + \lambda d_i \leq C$ are not violated. This is done by choosing a new λ^* so that $\lambda^* = \min\{\lambda', \lambda_{max}\}$, where $\lambda_{max} = \min\left\{\min_{i \in \mathcal{Q}^-} \left\{\frac{-\alpha_i}{d_i}\right\}, \min_{i \in \mathcal{Q}^+} \left\{\frac{C - \alpha_i}{d_i}\right\}\right\}$, and $\mathcal{Q}^{\pm}$ denote the index sets of the non-bounded SVs whose $\mathbf{d}$ component is either greater or less than zero [6]. If we are using $\widehat{\mathbf{d}}$, additional terms $C/\widehat{d_{\hat{p}}}$ if $\alpha_{\hat{p}} = 0$ or $-C/\widehat{d_{\hat{p}}}$ if $\alpha_{\hat{p}} = C$ are required in the outer minimum.

Dataset	# SVs.		# Iters.		
	SMO	Rosen	SMO	Rosen	Ratio
Titanic	69.3±9.5	92.0±11.3	143.5±20.9	141.4±23.9	1.01
Heart	82.5±5.4	83.4±5.1	149.8±31.5	65.6±6.3	2.28
Cancer	114.1±6.1	116.4±6.2	1431.5±407.3	592.1±93.0	2.42
Thyroid	25.3±5.7	24.8±5.5	213.5±73.9	70.5±23.2	3.03
Diabetes	264.9±7.3	266.5±7.4	360.1±55.0	181.3±9.8	1.99
Flare	480.7±11.5	513.1±17.2	608.7±141.7	339.8±50.8	1.79

Table 1: Average number of support vectors, number of iterations and ratio of iterations needed by Rosen and SMO algorithms, with $\epsilon = 10^{-3}$.

4 Experimental results

We shall compare the performance of Keerthi's Modification 2 for SMO [7] and Rosen's method over 6 of the datasets provided in G. Rätsch's Benchmark Repository [8]. We employed the same experimental setup described in the data site; in particular we used the 100 partitions provided (with about 40% training and 60% test patterns) to compute the final number of support vectors and the number of training iterations. We shall also give the ratio of iterations carried out by SMO against the ones by Rosen. To do this, we train 1-SVMs with standard Gaussian kernels, making use of the C and σ parameters also given in the data site. The stopping criterion of both algorithms is based on an absolute ϵ-violation of the KKT conditions, which is the criterion used in state-of-the-art SVM software like SVM-Light [3]. We give in tables 1 and 2 the results for $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$, respectively.

As it can be seen, the number of support vectors in the final models is very similar for both methods, except for datasets *titanic* and *flare*. Even for those 2 datasets we have checked that the test accuracies of both methods are undistinguishable under a Wilcoxon rank-sum test at the 0.1 level, so Rosen's method is effectively converging to a point with at least the same accuracy as the one given by SMO. Furthermore, the last column in table 1 shows that for $\epsilon = 10^{-3}$, Rosen generally requires around 2 or 3 times less iterations than SMO.

If we look at the results for $\epsilon = 10^{-6}$ in table 2, now SMO needs about 2 or 3 times more iterations than in the previous case, whereas Rosen remarkably needs very few more; thus, Rosen is now about 4 to 8 times faster than SMO in terms of iterations. Even if for these 6 datasets an accuracy of $\epsilon = 10^{-3}$ is adequate enough (note that the number of support vectors has changed very little between both tables), this fact seems to imply that Rosen's solution is somewhat more robust and thus more suitable for problems where a strict accuracy is needed or when we want to finish off the convergence of a first approximate solution.

5 Conclusions and further work

In this work we have shown that, when applied to SVM training, Rosen's gradient projection method can be written under closed formulae that lead to a simple and efficient implementation. We have also shown that the resulting models

Dataset	# SVs.		# Iters.		
	SMO	Rosen	SMO	Rosen	Ratio
Titanic	69.3±9.5	92.4±11.4	214.2±37.4	145.2±25.4	1.48
Heart	82.5±5.4	83.5±5.2	331.9±119.4	65.7±6.5	5.05
Cancer	113.9±6.1	116.4±6.3	4711.5±2704.7	596.0±95.2	7.91
Thyroid	25.3±5.7	24.8±5.5	519.2±222.3	70.6±23.4	7.35
Diabetes	264.8±7.3	266.5±7.4	730.1±193.1	181.7±9.7	4.02
Flare	481.2±11.3	513.2±18.2	1837.3±1677.0	394.6±80.4	4.66

Table 2: Average number of support vectors, number of iterations and ratio of iterations needed by Rosen and SMO algorithms, with $\epsilon = 10^{-6}$.

have accuracies similar to those obtained by SMO while requiring less iterations (particularly when a strict convergence is desired), as its working sets are bigger and their descent directions seem to be better. Moreover, the method ends up using the non–bounded support vectors as the working set, self adjusting thus to a shrinking strategy, and our experiments point out that it can speed up the final phases of other, simpler algorithms such as SMO.

However, and as it is often the case with other decomposition methods with large working sets, its implementation must be quite careful if a large number of kernel operations is to be avoided. This needs further study and a first possibility is to apply an adequate caching scheme for kernel operations. For instance, the method naturally lends itself to a training setup similar to Platt's type I and II iterations, as it automatically detects whether the objective function has been minimized over a given working set. Finally, there have been many proposals for efficient gradient projection methods in the QP literature. Thus, our work suggests that it may be worthwhile to explore whether simplifications similar to the one proposed here are possible for some of them. We are currently pursuing these and other related research topics.

References

[1] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[2] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Machines*, pages 185–208, 1999.

[3] T. Joachims. Making large-scale support vector machine learning practical. *Advances in Kernel Methods - Support Vector Machines*, pages 169–184, 1999.

[4] Y.H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Math. Program.*, 106:403–421, 2006.

[5] J. B. Rosen. The gradient projection method for nonlinear programming, i: Linear constraints. *SIAM Journal on Applied Mathematics*, 8:181–217, 1960.

[6] M. Bazaraa, D. Sherali, and C. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 1992.

[7] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murphy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[8] G. Rätsch. Benchmark repository. http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.