Journal of Machine Learning Research 12 (2011) 1149-1184 Submitted 10/09; Revised 2/10; Published 3/11

Laplacian Support Vector Machines Trained in the Primal

Stefano Melacci MELA@DII.UNISI.IT
Department of Information Engineering
University of Siena
Siena, 53100, ITALY

Mikhail Belkin MBELKIN@CSE.OHIO-STATE.EDU
Department of Computer Science and Engineering
Ohio State University
Columbus, OH 43210, USA

Editor: Sathiya Keerthi

Abstract

In the last few years, due to the growing ubiquity of unlabeled data, much effort has been spent by the machine learning community to develop better understanding and improve the quality of classifiers exploiting unlabeled data. Following the manifold regularization approach, Laplacian Support Vector Machines (LapSVMs) have shown the state of the art performance in semi-supervised classification. In this paper we present two strategies to solve the *primal* LapSVM problem, in order to overcome some issues of the original *dual* formulation. In particular, training a LapSVM in the primal can be efficiently performed with preconditioned conjugate gradient. We speed up training by using an early stopping strategy based on the prediction on unlabeled data or, if available, on labeled validation examples. This allows the algorithm to quickly compute approximate solutions with roughly the same classification accuracy as the optimal ones, considerably reducing the training time. The computational complexity of the training algorithm is reduced from $O(n^3)$ to $O(kn^2)$, where n is the combined number of labeled and unlabeled examples and k is empirically evaluated to be significantly smaller than n. Due to its simplicity, training LapSVM in the primal can be the starting point for additional enhancements of the original LapSVM formulation, such as those for dealing with large data sets. We present an extensive experimental evaluation on real world data showing the benefits of the proposed approach.

Keywords: Laplacian support vector machines, manifold regularization, semi-supervised learning, classification, optimization

1. Introduction

In semi-supervised learning one estimates a target classification/regression function from a few labeled examples together with a large collection of unlabeled data. In the last few years there has been a growing interest in the semi-supervised learning in the scientific community. Many algorithms for exploiting unlabeled data in order to enhance the quality of classifiers have been recently proposed, see, for example, Chapelle et al. (2006) and Zhu and Goldberg (2009). The general principle underlying semi-supervised learning is that the marginal distribution, which can be estimated from unlabeled data alone, may suggest a suitable way to adjust the target function. The two commons assumption on such distribution that, explicitly or implicitly, are made by many of semi-supervised learning algorithms are the *cluster assumption* (Chapelle et al., 2003) and the

manifold assumption (Belkin et al., 2006). The cluster assumption states that two points are likely to have the same class label if they can be connected by a curve through a high density region. Consequently, the separation boundary between classes should lie in the lower density region of the space. For example, this intuition underlies the Transductive Support Vector Machines (Vapnik, 2000) and its different implementations, such as TSVM (Joachims, 1999) or S^3VM (Demiriz and Bennett, 2000; Chapelle et al., 2008). The manifold assumption states that the marginal probability distribution underlying the data is supported on or near a low-dimensional manifold, and that the target function should change smoothly along the tangent direction. Many graph based methods have been proposed in this direction, but the most of them only perform transductive inference (Joachims, 2003; Belkin and Niyogi, 2003; Zhu et al., 2003), that is classify the unlabeled data given in training. Laplacian Support Vector Machines (LapSVMs) (Belkin et al., 2006) provide a natural out-of-sample extension, so that they can classify data that becomes available after the training process, without having to retrain the classifier or resort to various heuristics.

In this paper, we focus on the LapSVM algorithm, that has been shown to achieve state of the art performance in semi-supervised classification. The original approach used to train LapSVM in Belkin et al. (2006) is based on the dual formulation of the problem, in a traditional SVM-like fashion. This dual problem is defined on a number of dual variables equal to l, the number of labeled points. If the total number of labeled and unlabeled points is n, the relationship between the l variables and the final n coefficients is given by a linear system of n equations and variables. The overall cost of the process is $O(n^3)$.

Motivated by the recent interest in solving the SVM problem in the primal (Keerthi and DeCoste, 2005; Joachims, 2006; Chapelle, 2007; Shalev-Shwartz et al., 2007), we present a solution to the primal LapSVM problem that can significantly reduce training times and overcome some issues of the original training algorithm. Specifically, the contributions of this paper are the following:

1. We propose two methods for solving the LapSVM problem in the primal form (not limited to the linear case), following the ideas presented by Chapelle (2007) for SVMs and pointing out some important differences resulting from an additional regularization term. Our Matlab library can be downloaded from:
   `http://sourceforge.net/projects/lapsvmp/`

   First, we show how to solve the problem using the Newton's method and compare the result with the supervised (SVM) case. Interestingly, it turns out that the advantages of the Newton's method for the SVM problem are lost in LapSVM due to the intrinsic norm regularizer, and the complexity of this solution is still $O(n^3)$, same as in the original dual formulation.

   The second method is preconditioned conjugate gradient, which seems better suited to the LapSVM optimization problem. We see that preconditioning by the kernel matrix comes at no additional cost, and each iteration has complexity $O(n^2)$. Empirically, we establish that only a small number of iterations is necessary for convergence. Complexity can be further reduced if the kernel matrix is sparse, increasing the scalability of the algorithm.

2. We note that the quality of an approximate solution of the traditional dual form and the resulting approximation of the target optimal function are hard to relate due to the change of variables when passing to the dual problem. Training LapSVMs in the primal overcomes this issue, and it allows us to directly compute approximate solutions by controlling the number of conjugate gradient iterations.

3. An approximation of the target function with roughly the same classification accuracy as the optimal one can be achieved with a small number of iterations due to the influence of the intrinsic norm regularizer of LapSVMs on the training process. We investigate those effects, showing that they make common stopping conditions for iterative gradient based algorithms hard to tune, often leading to either a premature stopping of the iteration or to a large amount of unnecessary iterations, which do not improve classification accuracy. Instead we suggest a criterion dependent on the *output* of the classifier on the training data for terminating the iteration of our algorithm. This criterion exploits the stability of the prediction on the unlabeled data, or the classification accuracy on validation data (if available). A number of experiments on several data sets support these types of criteria, showing that accuracy similar to that of the optimal solution can be obtained in significantly reduced training time.

4. The primal solution of the LapSVM problem is based on an L_2 hinge loss, that establishes a direct connection to the Laplacian Regularized Least Square Classifier (LapRLSC) (Belkin et al., 2006). We discuss the similarities between primal LapSVM and LapRLSC and we show that the proposed fast solution can be straightforwardly applied to LapRLSC.

The rest of the paper is organized as follows. In Section 2 we recall the basic approach of manifold regularization. Section 2.1 describes the LapSVM algorithm in its original formulation while in Section 3 we discuss in detail the proposed solutions in the primal form. The quality of an approximate solution and the data based early stopping criterion are discussed in Section 4. In Section 5 a parallel with the primal solution of LapSVM and the solution for LapRLSC (Regularized Least Squares) is drawn, describing some possible future work. An extensive experimental analysis is presented in Section 6, and, finally, Section 7 concludes the paper.

2. Manifold Regularization

First, we introduce some notation that will be used in this section and in the rest of the paper. We take $n = l+u$ to be the number of m dimensional training examples $x_i \in X \subset \mathbb{R}^m$, collected in $\mathcal{S} = \{x_i, i = 1,\ldots,n\}$. Examples are ordered so that the first l ones are labeled, with label $y_i \in \{-1,1\}$, and the remaining u points are unlabeled. We put $\mathcal{S} = \mathcal{L} \cup \mathcal{U}$, where $\mathcal{L} = \{(x_i,y_i), i = 1,\ldots,l\}$ is the labeled data set and $\mathcal{U} = \{x_i, i = l+1,\ldots,n\}$ is the unlabeled data set. Labeled examples are generated accordingly to the distribution P on $X \times \mathbb{R}$, whereas unlabeled examples are drawn according to the marginal distribution P_X of P. Labels are obtained from the conditional probability distribution $P(y|x)$. L is the graph Laplacian associated to $\mathcal{S}$, given by $L = D - W$, where W is the adjacency matrix of the data graph (the entry in position i, j is indicated with w_{ij}) and D is the diagonal matrix with the degree of each node (i.e., the element d_{ii} from D is $d_{ii} = \sum_{j=1}^{n} w_{ij}$). Laplacian can be expressed in the normalized form, $L = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, and iterated to a degree p greater that one. By $K \in \mathbb{R}^{n,n}$ we denote the Gram matrix associated to the n points of $\mathcal{S}$ and the i, j-th entry of such matrix is the evaluation of the kernel function $k(x_i,x_j)$, $k : X \times X \to \mathbb{R}$. The unknown target function that the learning algorithm must estimate is indicated with $f : X \to \mathbb{R}$, where f is the vector of the n values of f on training data, $f = [f(x_i), x_i \in \mathcal{S}]^T$. In a classification problem, the decision function that discriminates between classes is indicated with $y(x) = g(f(x))$, where we overloaded the use of y to denote such function.

Manifold regularization approach (Belkin et al., 2006) exploits the geometry of the marginal distribution P_X. The support of the probability distribution of data is assumed to have the geometric

structure of a Riemannian manifold $\mathcal{M}$. The labels of two points that are close in the intrinsic geometry of P_X (i.e., with respect to geodesic distances on $\mathcal{M}$) should be the same or similar in sense that the conditional probability distribution $P(y|x)$ should change little between two such points. This constraint is enforced in the learning process by an intrinsic regularizer $\|f\|_I^2$ that is empirically estimated from the point cloud of labeled and unlabeled data using the graph Laplacian associated to them, since $\mathcal{M}$ is truly unknown. In particular, choosing exponential weights for the adjacency matrix leads to convergence of the graph Laplacian to the Laplace-Beltrami operator on the manifold (Belkin and Niyogi, 2008). As a result, we have

$$\|f\|_I^2 = \sum_{i=1}^{n}\sum_{j=i}^{n} w_{ij}(f(x_i) - f(x_j))^2 = f^T L f. \tag{1}$$

Consider that, in general, several natural choices of $\|\|_I$ exist (Belkin et al., 2006).

In the established regularization framework for function learning, given a kernel function $k(\cdot,\cdot)$, its associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ of functions $X \to \mathbb{R}$ with corresponding norm $\|\|_A$, we estimate the target function by minimizing

$$f^* = \arg\min_{f\in\mathcal{H}_k} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2 \tag{2}$$

where V is some loss function and γ_A is the weight of the norm of the function in the RKHS (or *ambient* norm), that enforces a smoothness condition on the possible solutions, and γ_I is the weight of the norm of the function in the low dimensional manifold (or *intrinsic* norm), that enforces smoothness along the sampled $\mathcal{M}$. For simplicity, we removed every normalization factor of the weights of each term in the summation. The ambient regularizer makes the problem well-posed, and its presence can be really helpful from a practical point of view when the manifold assumption holds at a lesser degree.

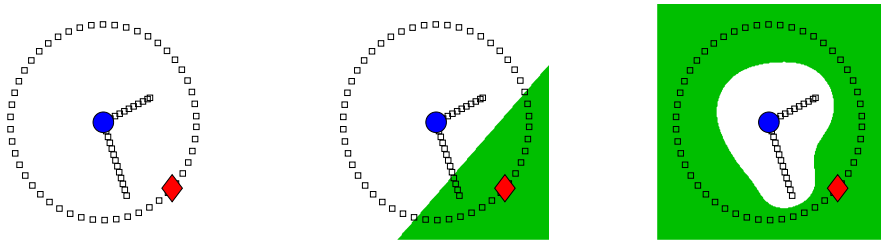
It has been shown in Belkin et al. (2006) that f^* admits an expansion in terms of the n points of $\mathcal{S}$,

$$f^*(x) = \sum_{i=1}^{n} \alpha_i^* k(x_i, x). \tag{3}$$

The decision function that discriminates between class $+1$ and -1 is $y(x) = sign(f^*(x))$. Figure 1 shows the effect of the intrinsic regularizer on the "clock" toy data set. The supervised approach defines the classification hyperplane just by considering the two labeled examples, and it does not benefit from unlabeled data (Figure 1(b)). With manifold regularization, the classification appears more natural with respect to the geometry of the marginal distribution (Figure 1(c)).

The intrinsic norm of Equation 1 actually performs a transduction along the manifold that enforces the values of f in nearby points with respect to geodesic distances on $\mathcal{M}$ to be the "same". From a merely practical point of view, the intrinsic regularizer can be excessively strict in some situations. Since the decision function $y(x)$ relies only on the sign of the target function $f(x)$, if f has the same sign on nearby points along $\mathcal{M}$ then the graph transduction is actually complete. Requiring that f assumes exactly the same value on a pair of nearby points could be considered as over constraining the problem. We will use this consideration in Section 4 to early stop the training algorithm.

This intuition is closely related to some recently proposed alternative formulations of the problem of Equation 2. In Tsang and Kwok (2006) the intrinsic regularizer is based on the ε-insensitive



(a) (b) (c)

Figure 1: (a) The two class "clock" data set. One class is the circular border of the clock, the other one is the hour/minute hands. A large set of unlabeled examples (black squares) and only one labeled example per class (red diamond, blue circle) are selected. - (b) The result of a maximum margin supervised classification - (c) The result of a semi-supervised classification with intrinsic norm from manifold regularization.

loss and the problem is mapped to a Minimal Enclosing Ball (MEB) formulation. Differently, the Manifold Co-Regularization (MCR) framework (Sindhwani and Rosenberg, 2008) has been introduced to overcome the degeneration of the intrinsic regularizer to the ambient one in some restricted function spaces where it is not able to model some underlying geometries of the given data. MCR is based on multi-view learning, and it has been shown that it corresponds to adding some extra slack variables in the objective function of Equation 2 to better fit the intrinsic regularizer. Similarly, Abernethy et al. (2008) use a slack based formulation to improve the flexibility of the graph regularizer of their spam detector.

2.1 Laplacian Support Vector Machines

LapSVMs follow the principles behind manifold regularization (Equation 2), where the loss function $V(x,y,f)$ is the linear hinge loss (Vapnik, 2000), or L_1 loss. The interesting property of such function is that well classified labeled examples are not penalized by $V(x,y,f)$, independently by the value of f.

In order to train a LapSVM classifier, the following problem must be solved

$$\min_{f\in\mathcal{H}_k} \sum_{i=1}^{l} \max(1-y_i f(x_i),0) + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2. \tag{4}$$

The function $f(x)$ admits the expansion of Equation 3, where an unregularized bias term b can be added as in many SVM formulations.

The solution of LapSVM problem proposed by Belkin et al. (2006) is based on the dual form. By introducing the slack variables ξ_i, the unconstrained primal problem can be written as a constrained one,

$$\min_{\alpha\in\mathbb{R}^n,\xi\in\mathbb{R}^l} \textstyle\sum_{i=1}^{l} \xi_i + \gamma_A \alpha^T K\alpha + \gamma_I \alpha^T KLK\alpha$$

$$\text{subject to:} \quad y_i\left(\textstyle\sum_{j=1}^{n} \alpha_i k(x_i,x_j) + b\right) \geq 1-\xi_i, \quad i=1,\ldots,l$$

$$\xi_i \geq 0, \quad i=1,\ldots,l.$$

After the introduction of two sets of n multipliers β, ς, the Lagrangian L_g associated to the problem is

$$\begin{aligned} L_g(\alpha,\xi,b,\beta,\varsigma) &= \sum_{i=1}^{l}\xi_i + \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \\ &\quad - \sum_{i=1}^{l}\beta_i(y_i(\sum_{j=1}^{n}\alpha_i k(x_i,x_j)+b)-1+\xi_i) - \sum_{i=1}^{l}\varsigma_i\xi_i. \end{aligned}$$

In order to recover the dual representation we need to set

$$\begin{aligned} &\frac{\partial L_g}{\partial b} = 0 \implies \sum_{i=1}^{l}\beta_i y_i = 0, \\ &\frac{\partial L_g}{\partial \xi_i} = 0 \implies 1-\beta_i-\varsigma_i = 0 \implies 0 \le \beta_i \le 1, \end{aligned}$$

where the bounds on β_i consider that $\beta_i, \varsigma_i \ge 0$, since they are Lagrange multipliers. Using the above identities, we can rewrite the Lagrangian as a function of α and β only. Assuming (as stated in Section 2) that the points in $\mathcal{S}$ are ordered such that the first l are labeled and the remaining u are unlabeled, we define with $J_{\mathcal{L}} \in \mathbb{R}^{l,n}$ the matrix $[I\ 0]$ where $I \in \mathbb{R}^{l,l}$ is the identity matrix and $0 \in \mathbb{R}^{l,u}$ is a rectangular matrix with all zeros. Moreover, $Y \in \mathbb{R}^{l,l}$ is a diagonal matrix composed by the labels $y_i, i = 1,\ldots,l$. The Lagrangian becomes

$$\begin{aligned} L_g(\alpha,\beta) &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \sum_{i=1}^{l}\beta_i(y_i(\sum_{j=1}^{n}\alpha_i k(x_i,x_j)+b)-1) = \\ &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \alpha^T K J_{\mathcal{L}}^T Y\beta + \sum_{i=1}^{l}\beta_i. \end{aligned}$$

Setting to zero the derivative with respect to α establishes a direct relationships between the β coefficients and the α ones,

$$\begin{aligned} &\frac{\partial L_g}{\partial \alpha} = 0 \implies (2\gamma_A K + 2\gamma_I KLK)\alpha - K J_{\mathcal{L}}^T Y\beta = 0 \\ &\qquad \implies \alpha = (2\gamma_A I + 2\gamma_I KL)^{-1} J_{\mathcal{L}}^T Y\beta. \end{aligned} \tag{5}$$

After substituting back in the Lagrangian expression, we get the dual problem whose solution leads to the optimal β^*, that is

$$\max_{\beta\in\mathbb{R}^l} \textstyle\sum_{i=1}^{l}\beta_i - \frac{1}{2}\beta^T Q\beta$$

$$\text{subject to:} \quad \textstyle\sum_{i=1}^{l}\beta_i y_i = 0$$

$$0 \le \beta_i \le 1, \quad i = 1,\ldots,l$$

where

$$Q = Y J_{\mathcal{L}} K(2\gamma_A I + 2\gamma_I KL)^{-1} J_{\mathcal{L}}^T Y. \tag{6}$$

Training the LapSVM classifier requires to optimize this l variable problem, for example using a standard quadratic SVM solver, and then to solve the linear system of n equations and n variables of Equation 5 in order to get the coefficients α^* that define the target function f^*.

The overall complexity of this solution is $O(n^3)$, due to the matrix inversion of Equation 5 (and 6). Even if the l coefficients β^* are sparse, since they come from a SVM-like dual problem, the expansion of f^* will generally involves all n coefficients α^*.

3. Training in the Primal

In this section we analyze the optimization of the primal form of the non linear LapSVM problem, following the growing interest in training SVMs in the primal of the last few years (Keerthi and DeCoste, 2005; Joachims, 2006; Chapelle, 2007; Shalev-Shwartz et al., 2007). Primal optimization of a SVM has strong similarities with the dual strategy (Chapelle, 2007), and its implementation does not require any particularly complex optimization libraries. The focus of researchers has been mainly on the solution of the linear SVM primal problem, showing how it can be solved fast and efficiently. In the Modified Finite Newton method of Keerthi and DeCoste (2005) the SVM problem is optimized in the primal by a numerically robust conjugate gradient technique that implements the Newton iterations. In the works of Joachims (2006) and Shalev-Shwartz et al. (2007) a cutting plane algorithm and a stochastic gradient descent are exploited, respectively. Most of the existing results can be directly extended to the non linear case by reparametrizing the linear output function $f(x) = \langle w, x \rangle + b$ with $w = \sum_{i=1}^{l} \alpha_i x_i$ and introducing the Gram matrix K. However this may result in a loss of efficiency. Other authors (Chapelle, 2007; Keerthi et al., 2006) investigated efficient solutions for the non linear SVM case.

Primal and dual optimization are two ways different of solving the same problem, neither of which can in general be considered a "better" approach. Therefore why should a solution of the primal problem be useful in the case of LapSVM? There are three primary reasons why such a solution may be preferable. First, it allows us to efficiently solve the original problem without the need of the computations related to the variable switching. Second, it allows us to very quickly compute good *approximate* solutions, while the exact relation between approximate solutions of the dual and original problems may be involved. Third, since it allows us to directly "manipulate" the α coefficients of f without passing through the β ones, greedy techniques for incremental building of the LapSVM classifier are easier to manage (Sindhwani, 2007). We believe that studying the primal LapSVM problem is the basis for future investigations and improvements of this classifier.

We rewrite the primal LapSVM problem of Equation 4 by considering the representation of f of Equation 3, the intrinsic regularizer of Equation 1, and by indicating with k_i the i-th column of the matrix K and with 1 the vector of n elements equal to 1:

$$\min_{\alpha \in \mathbb{R}^n, b \in \mathbb{R}} \sum_{i=1}^{l} V(x_i, y_i, k_i^T \alpha + b) + \gamma_A \alpha^T K \alpha + \gamma_I (\alpha^T K + 1^T b) L (K\alpha + 1b).$$

For completeness, we included the bias b in the expansion of f. Here and in all the following derivations, L can be interchangeably used in its normalized or unnormalized version.

We use the squared hinge loss, or L_2 loss, for the labeled examples. The differentiability of such function and its properties have been investigated in Mangasarian (2002) and applied to kernel classifiers. Afterwards, it was also exploited by Keerthi and DeCoste (2005) and Chapelle (2007). L_2 loss makes the LapSVM problem continuous and differentiable in f and so in α. The optimization

problem after adding the scaling constant $\frac{1}{2}$ becomes

$$\min_{\alpha\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{2}\left(\sum_{i=1}^{l} \max(1-y_i(k_i^T\alpha+b),0)^2+\gamma_A\alpha^T K\alpha+\gamma_I(\alpha^T K+1^T b)L(K\alpha+1b)\right). \quad (7)$$

We solved such convex problem by Newton's method and by preconditioned conjugate gradient, comparing their complexities and the complexity of the original LapSVM solution, and showing a parallel with the SVM case. The two solution strategies are analyzed in the following Subsections, while a large set of experimental results are collected in Section 6.

3.1 Newton's Method

The problem of Equation 7 is piecewise quadratic and the Newton's method appears a natural choice for an efficient minimization, since it builds a quadratic approximation of the function. After indicating with z the vector $z=[b,\alpha^T]^T$, each Newton's step consists of the following update

$$z^t = z^{t-1} - sH^{-1}\nabla \quad (8)$$

where t is the iteration number, s is the step size, and ∇ and H are the gradient and the Hessian of Equation 7 with respect to z. We will use the symbols ∇_α and ∇_b to indicate the gradient with respect to α and to b.

Before continuing, we introduce the further concept of *error vectors* (Chapelle, 2007). The set of error vectors $\mathcal{E}$ is the subset of $\mathcal{L}$ with the points that generate a L_2 hinge loss value greater than zero. The classifier does not penalize all the remaining labeled points, since the f function on that points produces outputs with the same sign of the corresponding label and with absolute value greater then or equal to it. In the classic SVM framework, error vectors correspond to support vectors at the optimal solution. In the case of LapSVM, all points are support vectors in the sense that they all generally contribute to the expansion of f.

We have

$$\nabla = \begin{bmatrix} \nabla_b \\ \nabla_\alpha \end{bmatrix} = \begin{pmatrix} 1^T I_{\mathcal{E}}(K\alpha+1b) - 1I_{\mathcal{E}}y + \gamma_I 1^T L(K\alpha+1b) \\ KI_{\mathcal{E}}(K\alpha+1b) - KI_{\mathcal{E}}y + \gamma_A K\alpha + \gamma_I KL(K\alpha+1b) \end{pmatrix} \quad (9)$$

where $y\in\{-1,0,1\}^n$ is the vector that collects the l labels y_i of the labeled training points and a set of u zeros. The matrix $I_{\mathcal{E}}\in\mathbb{R}^{n,n}$ is a diagonal matrix where the only elements different from 0 (and equal to 1) along the main diagonal are in positions corresponding to points of $\mathcal{S}$ that belong to $\mathcal{E}$ at the current iteration. Note that if the graph Laplacian is not normalized, we have $1^T L = 0^T$ and, equivalently, $L1=0$.

The Hessian H is

$$H = \begin{pmatrix} \nabla_b^2 & \nabla_b(\nabla_\alpha) \\ \nabla_\alpha(\nabla_b) & \nabla_\alpha^2 \end{pmatrix} = \begin{pmatrix} 1^T I_{\mathcal{E}}1+\gamma_I 1^T L1 & 1^T I_{\mathcal{E}}K+\gamma_I 1^T LK \\ KI_{\mathcal{E}}1+\gamma_I KL1 & KI_{\mathcal{E}}K+\gamma_A K+\gamma_I KLK \end{pmatrix} =$$

$$= \begin{pmatrix} -\gamma_A & 1^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}}1+\gamma_I L1 & I_{\mathcal{E}}K+\gamma_A I+\gamma_I LK \end{pmatrix}.$$

Note that the criterion function of Equation 7 is not twice differentiable everywhere, so that H is the generalized Hessian where the subdifferential in the breakpoint of the hinge function is set to

0. This leaves intact the least square nature of the problem, as in the Modified Newton's method proposed by Keerthi and DeCoste (2005) for linear SVMs. In other words, the contribute to the Hessian of the L_2 hinge loss is the same as the one of a squared loss $(y_i - f(x_i))^2$ applied to error vectors only.

Combining the last two expressions we can write $\nabla = Hz - \begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y$, and we can plug it into the Newton's update of Equation 8,

$$z^t = z^{t-1} - sH^{-1}\nabla = (1-s)z^{t-1} + sH^{-1}\begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y =$$

$$= (1-s)z^{t-1} + s\begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}}1 + \gamma_I L1 & I_{\mathcal{E}}K + \gamma_A I + \gamma_I LK \end{pmatrix}^{-1}\begin{pmatrix} -\gamma_A & 1^T \\ 0 & K \end{pmatrix}^{-1}\begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y = \qquad (10)$$

$$= (1-s)z^{t-1} + s\begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}}1 + \gamma_I L1 & I_{\mathcal{E}}K + \gamma_A I + \gamma_I LK \end{pmatrix}^{-1}\begin{pmatrix} 0 \\ I_{\mathcal{E}} y \end{pmatrix}.$$

The step size s must be computed by solving the one-dimensional minimization of Equation 7 restricted on the ray from z^{t-1} to z^t, with exact line search or backtracking (Boyd and Vandenberghe, 2004). Convergence is declared when the set of error vectors does not change between two consecutive iterations of the algorithm. Exactly like in the case of primal SVMs (Chapelle, 2007), in our experiments setting $s = 1$ did not result in any convergence problems.

3.1.1 COMPLEXITY ANALYSIS

Updating the α coefficients with the Newton's method costs $O(n^3)$, due to the matrix inversion in the update rule, the same complexity of the original LapSVM solution based on the dual problem discussed in Section 2.1. Convergence is usually achieved in a tiny number of iterations, no more than 5 in our experiments (see Section 6). In order to reduce the cost of each iteration, a Cholesky factorization of the Hessian can be computed before performing the first matrix inversion, and it can be updated using a rank-1 scheme during the following iterations, with cost $O(n^2)$ for each update (Seeger, 2008). On the other hand, this does not allow us to simplify K in Equation 10, otherwise the resulting matrix to be inverted will not be symmetric. Since a lot of time is wasted in the product by K (that is usually dense), using the update of Cholesky factorization may not necessarily lead to a reduction of the overall training time.

It is interesting to compare the training of SVMs in the primal with the one of LapSVMs for a better insight in the Newton's method based solution. Given the set $\mathcal{E}$ at a generic iteration, SVMs only require to compute the inverse of the block of the Hessian matrix that is related to the error vectors, and the complexity of the inversion is then $O(|\mathcal{E}|^3)$ (see Chapelle, 2007). Exploiting this useful aspect, the training algorithm can be run incrementally, reducing the complexity of the whole training process. In the case of LapSVM those benefits are lost due to the presence of the intrinsic norm $f^T L f$. The additional penalty $w_{ij}(f(x_i) - f(x_j))^2$ makes the Hessian a full matrix, making the block inversion impossible.

Finally, we are assuming that K and the matrix to invert on Equation 10 are non singular, otherwise the final expansion of f will not be unique, even if the optimal value of the criterion function of Equation 7 will be.

3.2 Preconditioned Conjugate Gradient

Instead of performing a costly Newton's step, the vector z for which $\nabla = 0$ can be computed by Conjugate Gradient (CG) descent. In particular if we look at Equation 9, we can write $\nabla = Hz - c$ and, consequently, we have to solve the system $Hz = c$,

$$Hz = c \Longrightarrow \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T L K \\ K I_{\mathcal{E}} 1 + \gamma_I K L 1 & K I_{\mathcal{E}} K + \gamma_A K + \gamma_I K L K \end{pmatrix} z = \begin{pmatrix} 1^T I_{\mathcal{E}} y \\ K I_{\mathcal{E}} y \end{pmatrix}. \tag{11}$$

The convergence rate of CG is related to the condition number of H (Shewchuk, 1994). In the most general case, the presence of the terms $KI_{\mathcal{E}}K$ and KLK leads to a not so well conditioned system and to a slow convergence rate.

In order to overcome this issue, Preconditioned Conjugate Gradient (PCG) can be exploited (Shewchuk, 1994). Given a preconditioner P, the algorithm indirectly solves the system of Equation 11 by solving $\hat{H}z = \hat{c}$, where $\hat{H} = P^{-1}H$ and $\hat{c} = P^{-1}c$. P is selected so that the condition number of $\hat{H}z = \hat{c}$ is improved with respect to the initial system, leading to a faster convergence rate of the iterative method. Moreover, P^{-1} must be easily computable for PCG to be efficient. In the specific case of LapSVM, we can follow a similar strategy to the one investigated by Chapelle (2007), due to the quadratic form of the intrinsic regularizer. In particular, we can factorize Equation 11 as

$$\begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T L K \\ I_{\mathcal{E}} 1 + \gamma_I L 1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I L K \end{pmatrix} z = \begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 1^T I_{\mathcal{E}} y \\ I_{\mathcal{E}} y \end{pmatrix}, \tag{12}$$

and select as a preconditioner the symmetric matrix $P = \begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix}$. We can see that P is a factor of H and c, hence the terms $\hat{H}$ and $\hat{c}$ (and, consequently, the preconditioned gradient $\hat{\nabla}$, given by $\hat{\nabla} = P^{-1}\nabla = \hat{H}z - \hat{c}$) can be trivially computed without explicitly performing any matrix inversions. The condition number of the preconditioned system is sensibly decreased with respect to the one of Equation 11, since $KI_{\mathcal{E}}K$ and KLK are reduced to $I_{\mathcal{E}}K$ and LK. Note that $\hat{H}$ is not symmetric, and it would not possible, for instance, to simply remove the factor P in both sides of Equation 12 and solve it by standard CG. For those reasons, PCG is appropriate for an efficient optimization of our problem. As in the Newton's method, we are assuming that K is non singular, otherwise a small ridge can be added to fix it.

The iterative solution of the LapSVM problem by means of PCG is reported in Algorithm 1. For an easier comparison with the standard formulation of PCG, consider that the vectors of residual of the original and preconditioned systems corresponds to $-\nabla$ and $-\hat{\nabla}$, respectively. Nevertheless, due to our choice of P, we do not need to compute ∇ first, and then $\hat{\nabla} = P^{-1}\nabla$. We can exchange the order of those operations to avoid the matrix inversion, that is, first compute $\hat{\nabla}$ and then $\nabla = P\hat{\nabla}$. Hence, P^{-1} never appears in Algorithm 1.

Classic rules for the update of the conjugate direction at each step are discussed by Shewchuk (1994). After several iterations the conjugacy of the descent directions tends to get lost due to round-off floating point error, so a restart of the preconditioned conjugate gradient algorithm is required. The Fletcher-Reeves (FR) update is commonly used in linear optimization. Due to the piecewise nature of the problem, defined by the $I_{\mathcal{E}}$ matrix, we exploited the Polak-Ribiere (PR) formula,[1]

1. Note that in the linear case FR and PR are equivalent.

where restart can be automatically performed when the update term becomes negative. In that case, the ρ coefficient in Algorithm 1 becomes zero, and the following iteration corresponds to a steepest descent one, as when PCG starts. We experimentally evaluated that for the LapSVM problem such formula is generally the best choice, both for convergence speed and numerical stability.

Convergence is usually declared when the norm of the preconditioned gradient falls below a given threshold (Chapelle, 2007), or when the current preconditioned gradient is roughly orthogonal with the real gradient (Shewchuk, 1994). We will investigate these conditions in Section 4.

Algorithm 1 Preconditioned Conjugate Gradient (PCG) for primal LapSVMs.

Let $t=0$, $z^t=0$, $\mathcal{E}=\mathcal{L}$, $\hat{\nabla}^t=[-1^T y,-y^T]^T$, $d^t=-\hat{\nabla}^t$
repeat
  $t=t+1$
  Find s^* by line search on the line $z^{t-1}+sd^{t-1}$
  $z^t=z^{t-1}+s^*d^{t-1}$
  $\mathcal{E}=\{x_i\in\mathcal{L}\ \ s.t.\ \ (k_i\alpha^t+b^t)y_i<1\}$
  $\hat{\nabla}^t=\hat{H}z-\hat{c}=\begin{pmatrix} 1^T I_{\mathcal{E}}1+\gamma_I 1^T L1 & 1^T I_{\mathcal{E}}K+\gamma_I 1^T LK \\ I_{\mathcal{E}}1+\gamma_I L1 & I_{\mathcal{E}}K+\gamma_A I+\gamma_I LK \end{pmatrix} z-\begin{pmatrix} 1^T I_{\mathcal{E}}y \\ I_{\mathcal{E}}y \end{pmatrix}$
  $\nabla^t=Hz-c=P\hat{H}z-P\hat{c}=P\hat{\nabla}^t$
  $\rho=\max(\frac{\nabla^{t^T}(\hat{\nabla}^t-\hat{\nabla}^{t-1})}{\nabla^{t-1^T}\hat{\nabla}^{t-1}},0)$
  $d^t=-\hat{\nabla}^t+\rho d^{t-1}$
until Goal condition

3.2.1 LINE SEARCH

The optimal step length s^* on the current direction of the PCG algorithm must be computed by backtracking or exact line search. At a generic iteration t we have to solve

$$s^*=\arg\min_{s\geq 0} obj(z^{t-1}+sd^{t-1}) \tag{13}$$

where obj is the objective function of Equation 7.

The accuracy of the line search is crucial for the performance of PCG. When minimizing a quadratic form that leads to a linear expression of the gradient, line search can be computed in closed form. In our case, we have to deal with the variations of the set $\mathcal{E}$ (and of $I_{\mathcal{E}}$) for different values of s, so that a closed form solution cannot be derived, and we have to compute the optimal s in an iterative way.

Due to the quadratic nature of Equation 13, the 1-dimensional Newton's method can be directly used, but the average number of line search iterations per PCG step can be very large, even if the cost of each of them is negligible with respect to the $O(n^2)$ of a PCG iteration. We can efficiently solve the line search problem analytically, as suggested by Keerthi and DeCoste (2005) for SVMs.

In order to simplify the notation, we discard the iteration index $t-1$ in the following description. Given the PCG direction d, we compute for each point $x_i\in\mathcal{L}$, being it an error vector or not, the step length s_i for which its state switches. The state of a given error vector switches when it leaves

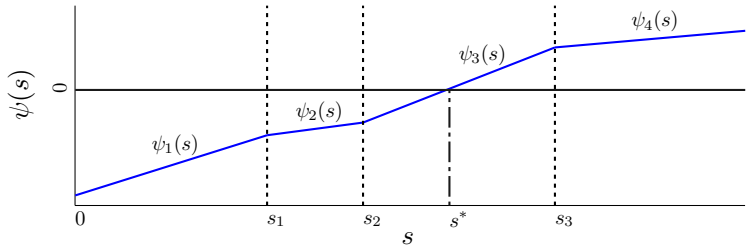


Figure 2: Example of the piecewise linear function $\psi(s)$ (blue plot). $\psi_1(s),\ldots,\psi_4(s)$ are the four linear portions of $\psi(s)$, and s_1, s_2, s_3 are the break points. The optimal step length, s^*, is the value for which $\psi(s)$ crosses zero.

the $\mathcal{E}$ set, whether the state of a point initially not in $\mathcal{E}$ switches when it becomes an error vector. We refer to the set of the former points with Q_1 while the latter is Q_2, with $\mathcal{L} = Q_1 \cup Q_2$. The derivative of Equation 13, $\psi(s) = \partial obj(z+sd)/\partial s$, is piecewise linear, and s_i are the break points of such function.

Let us consider, for simplicity, that s_i are in a non decreasing order, discarding the negative ones. Starting from $s = 0$, they define a set of intervals where $\psi(s)$ is linear and the $\mathcal{E}$ set does not change. We indicate with $\psi_j(s)$ the linear portion of $\psi(s)$ in the j-th interval. Starting with $j = 1$, if the value $s \geq 0$ where the line $\psi_j(s)$ crosses zero is within such interval, then it is the optimal step size s^*, otherwise the following interval must be checked. The convergence of the process is guaranteed by the convexity of the function obj. See Figure 2 for a basic example.

The zero crossing of $\psi_j(s)$ is given by $s = \frac{\psi_j(0)}{\psi_j(0)-\psi_j(1)}$, where the two points $(0,\psi_j(0))$ and $(1,\psi_j(1))$ determine the line $\psi_j(s)$. We indicate with $f_d(x)$ the function $f(x)$ whose coefficients are in $d = [d_b, d_\alpha^T]^T$, that is, $f_d(x_i) = k_i^T d_\alpha + d_b$, and $f_d = [f_d(x_i), x_i \in \mathcal{S}]^T$. We have

$$\psi_j(0) = \sum_{x_i \in \mathcal{E}_j} (f(x_i) - y_i) f_d(x_i) + \gamma_A \alpha^T K d_\alpha + \gamma_I f_d^T L f,$$
$$\psi_j(1) = \sum_{x_i \in \mathcal{E}_j} (f(x_i) + f_d(x_i) - y_i) f_d(x_i) + \gamma_A (\alpha + d_\alpha)^T K d_\alpha + \gamma_I f_d^T L (f + f_d)$$

where $\mathcal{E}_j$ is the set of error vectors for the j-th interval.

Given $\psi_1(0)$ and $\psi_1(1)$, their successive values for increasing j can be easily computed considering that only one point (that we indicate with x_j) switches status moving from an interval to the following one. From this consideration we derived the following update rules

$$\psi_{j+1}(0) = \psi_j(0) + \nu_j (f(x_j) - y_j) f_d(x_j),$$
$$\psi_{j+1}(1) = \psi_j(1) + \nu_j (f(x_j) + f_d(x_j) - y_i) f_d(x_j)$$

where ν_j is -1 if $x_j \in Q_1$ and it is $+1$ if $r \in Q_2$.

3.2.2 COMPLEXITY ANALYSIS

Each PCG iteration requires to compute the $K\alpha$ product, leading to a complexity of $O(n^2)$ to update the α coefficients. The term $LK\alpha$ can then be computed efficiently from $K\alpha$, since the matrix L is generally sparse. Note that, unlike the Newton's method and the original dual solution of the LapSVM problem, we never have to explicitly compute the LK product, always computing matrix by vector products instead. Even if L is sparse, when the number of training points is large or

L is iterated several times, a large amount of computation may be saved by avoiding such matrix by matrix product, as we will show in Section 6. Moreover, if the kernel matrix is sparse, the complexity drops to $O(n_{nz})$, where n_{nz} is the maximum number of non-zero elements between K and L. Note that the algorithm does not necessarily need to hold the whole matrix K (and L) in memory. The only requirement is a fast way to perform the product of K with the current α. On the other hand, computing each kernel function evaluation on the fly may require a large number of floating-point operations, so that some caching procedures must be devised.

Convergence of the conjugate gradient algorithm is theoretically declared in $O(n)$ steps, but a solution very close to the optimal one can be computed with far less iterations. The convergence speed is related to the condition number of the Hessian (Shewchuk, 1994), that it is composed by a sum of three contributes (Equation 11). As a consequence, their condition numbers and weighting coefficients (γ_A, γ_I) have a direct influence in the convergence speed, and in particular the condition number of the K matrix. For example, using a bandwidth of a Gaussian kernel that lead to a K matrix close to the identity allows the algorithm to converge very quickly, but the accuracy of the classifier may not be sufficient.

Finally, PCG can be efficiently seeded with an initial rough estimate of the solution ('warm" or "hot" start). For example, the solution computed for some given values of the γ_A and γ_I parameters can be a good starting point when training the classifier with some just slightly different parameter values (i.e., when cross-validating the model). Seeding is also crucial in schemes that allow the classifier to be incrementally built with reduced complexity. They have been deeply investigated by Keerthi et al. (2006) for the SVM classifier. Even if Keerthi et al. (2006) use the Newton optimization, a similar approach could be studied for LapSVMs exploiting the useful properties of the PCG algorithm.

4. Approximating the Optimal Solution

In order to reduce the training times, we want the PCG to converge as fast as possible to a good *approximation* of the optimal solution. By appropriately selecting the goal condition of Algorithm 1, we can discard iterations that may not lead to significant improvement in the classifier quality. This concept is widely used in optimization, where the early stop of the CG or PCG is exploited to approximately solve the Newton system in truncated Newton methods (see, for example, the trust region method for large-scale logistic regression of Lin et al., 2008).

The common goal conditions for the PCG algorithm and, more generally, for gradient based iterative algorithms, rely on the norm of the gradient $\|\nabla\|$ (Boyd and Vandenberghe, 2004), of the preconditioned gradient $\|\hat{\nabla}\|$ (Chapelle, 2007), on the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ (Shewchuk, 1994). These values are usually normalized by the first estimate of each of them. The value of the objective function obj or its relative decrement between two consecutive iterations can also be checked, requiring some additional computations since the PCG algorithm never explicitly computes it. When one of such "stopping" values falls below the chosen threshold τ associated to it, the algorithm terminates.[2] Moreover, a maximum number t_{max} of iterations is generally specified. Tuning these parameters is crucial both for the time spent running the algorithm and the quality of the resulting solution.

2. Thresholds associated to different conditions are obviously different, but, for simplicity in the description, we will refer to a generic threshold τ.

It is really hard to find a trade-off between good approximation and low number of iterations, since τ and t_{max} are strictly problem dependent. As an example, consider that the surface of obj, the objective function of Equation 7, varies among different choices of its parameters. Increasing or decreasing the values of γ_A and γ_I can lead to a less flat or a more flat region around the optimal point. Fixing in advance the values of τ and t_{max} may cause an early stop too far from the optimal solution, or it may result in the execution of a large number of iterations without a significant improvement on the classification accuracy.

The latter situation can be particularly frequent for LapSVMs. As described in Section 2 the choice of the intrinsic norm $f^T L f$ introduces the soft constraint $f(x_i) = f(x_j)$ for nearby points x_i, x_j along the underlying manifold. This allows the algorithm to perform a graph transduction and diffuse the labels from points in $\mathcal{L}$ to the unlabeled data $\mathcal{U}$.

When the diffusion is somewhat complete and the classification hyperplane has assumed a quite stable shape around the available training data, similar to the optimal one, the intrinsic norm will keep contributing to the gradient until a balance with respect to the ambient norm (and to the L_2 loss on error vectors) is found. Due to the strictness of this constraint, it will still require some iterations (sometimes many) to achieve the optimal solution with $\|\nabla\| = 0$, even if the decision function $y(x) = sign(f(x))$ will remain substantially the same. The described common goal conditions do not "directly" take into account the decision of the classifier, so that they do not appear appropriate to early stop the PCG algorithm for LapSVMs.

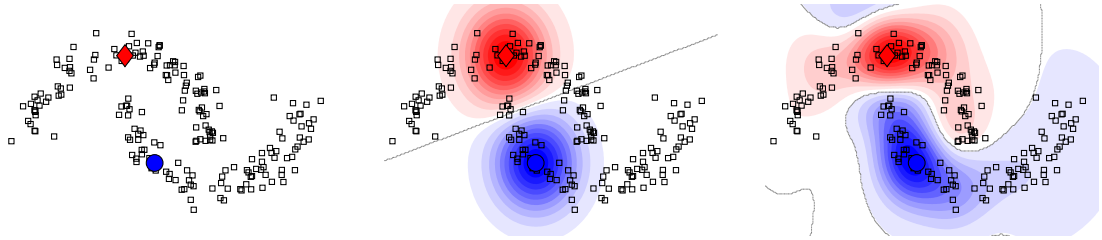
We investigate our intuition on the "two moons" data set of Figure 3(a), where we compare the decision boundary after each PCG iteration (Figure 3(b)-(e)) with the optimal solution (computed by Newton's method, Figure 3(f)). Starting with $\alpha = 0$, the first iteration exploits only the gradient of the L_2 loss on labeled points, since both the regularizing norms are zero. In the following iterations we can observe the label diffusion process along the manifold. After only 4 iterations we get a perfect classification of the data set and a separating boundary not far from the optimal one. All the remaining iterations until complete convergence are used to slightly asses the coherence along the manifold required by the intrinsic norm and the balancing with the smoothness of the function, as can be observed by looking at the function values after 25 iterations. The most of changes influences regions far from the support of P_X, and it is clear that an early stop after 4 PCG steps would be enough to roughly approximate the accuracy of optimal solution.

In Figure 4 we can observe the values of the previously described general stopping criterion for PCG. After 4 iterations they are still sensibly decreasing, without reflecting real improvements in the classifier quality. The value of the objective function obj starts to become more stable only after, say, 16 iterations, but it is still slightly decreasing even if it appears quite horizontal on the graph, due to its scale. It is clear that fixing in advance the parameters τ and t_{max} is random guessing and it will probably result in a bad trade-off between training time and accuracy.

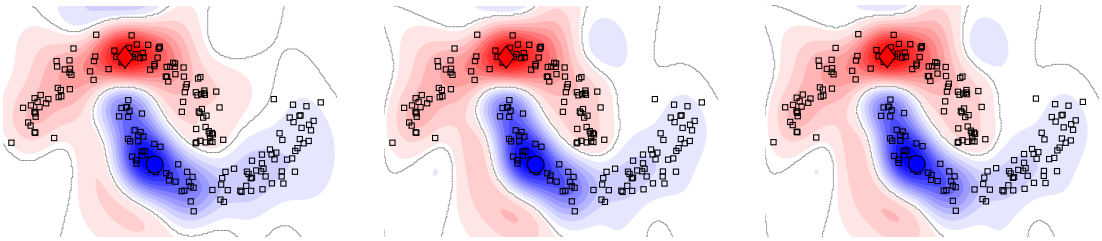
4.1 Early Stopping Conditions

Following these considerations, we propose to early stop the PCG algorithm exploiting the predictions of the classifier on the available data.

Due to the high amount of unlabeled training points in the semi-supervised learning framework, the stability of the decision $y(x) = sign(f(x))$, $x \in \mathcal{U}$, can be used as a reference to early stop the gradient descent (*stability check*). Moreover, if labeled validation data (set $\mathcal{V}$) is available for



(a) The "two moons" data set (b) 1 PCG iteration (c) 4 PCG iterations (**0% error**)



(d) 8 PCG iterations (e) 25 PCG iterations (f) Optimal solution

Figure 3: (a) The "two moons" data set (200 points, 2 classes, 2 labeled points indicated with a red diamond and a blue circle, whereas the remaining points are unlabeled) - (b-e) A LapSVM classifier trained with PCG, showing the result after a fixed number of iterations. The dark continuous line is the decision boundary ($f(x) = 0$) and the confidence of the classifier ranges from red ($f(x) \geq 1$) to blue ($f(x) \leq -1$) - (f) The optimal solution of the LapSVM problem computed by means of Newton's method

classifier parameters tuning, we can formulate a good stopping condition based on the classification accuracy on it (*validation check*), that can be eventually merged to the previous one (*mixed check*).

In detail, when $y(x)$ becomes quite stable between consecutive iterations or when $err(\mathcal{V})$, the error rate on $\mathcal{V}$, is not decreasing anymore, then the PCG algorithm should be stopped. Due to their heuristic nature, it is generally better to compare the predictions every θ iterations and within a certain tolerance η. As a matter of fact, $y(x)$ may slightly change also when we are very close to the optimal solution, and $err(\mathcal{V})$ is not necessarily an always decreasing function. Moreover, labeled validation data in the semi-supervised setting is usually small with respect to the whole training data, labeled and unlabeled, and it may not be enough to represent the structure of the data set.

We propose very simple implementations of such conditions, that we used to achieve the results of Section 6. Starting from these, many different and more efficient variants can be formulated, but it goes beyond the scope of this paper. They are sketched in Algorithms 2 and 3. We computed the classifier decision every $\sqrt{n}/2$ iterations and we required the classifier to improve $err(\mathcal{V})$ by one correctly classifier example at every check, due to the usually small size of $\mathcal{V}$. Sometimes this can also help to avoid a slight overfitting of the classifier.

Generating the decision $y(x)$ on unlabeled data does not require heavy additional machinery, since the $K\alpha$ product must be necessarily computed to perform every PCG iteration. Its overall cost is $O(u)$. Differently, computing the accuracy on validation data requires the evaluation of the kernel

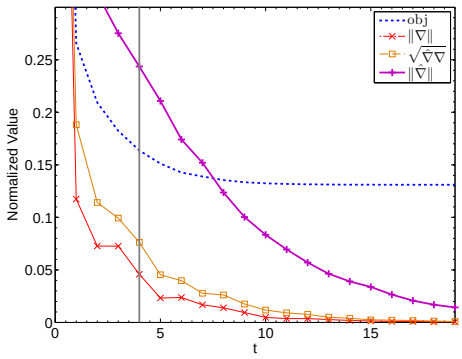


Figure 4: PCG example on the "two moons" data set. The norm of the gradient $\|\nabla\|$, of the preconditioned gradient $\|\hat{\nabla}\|$, the value of the objective function obj and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations. The vertical line represents the number of iterations after which the error rate is 0% and the decision boundary is quite stable.

Algorithm 2 The *stability check* for PCG stopping.

$d^{old} \leftarrow 0 \in \mathbb{R}^u$
$\eta \leftarrow 1.5\%$
$\theta \leftarrow \sqrt{n}/2$
Every θ *iterations do the followings:*
$d = [y(x_j), x_j \in \mathcal{U}, j = 1,\ldots,u]^T$
$\tau = (100 \cdot \|d - d^{old}\|_1/u)\%$
if $\tau < \eta$ **then**
  Stop PCG
else
  $d^{old} = d$
end if

Algorithm 3 The *validation check* for PCG stopping.

Require: $\mathcal{V}$
$err\mathcal{V}^{old} \leftarrow 100\%$
$\eta \leftarrow 100 \cdot |\mathcal{V}|^{-1}\%$
$\theta \leftarrow \sqrt{n}/2$
Every θ *iterations do the followings:*
if $err(\mathcal{V}) > (err\mathcal{V}^{old} - \eta)$ **then**
  Stop PCG
else
  $err\mathcal{V}^{old} = err(\mathcal{V})$
end if

function on validation points against the n training ones, and $O(|\mathcal{V}| \cdot n)$ products, that is negligible with respect to the cost of a PCG iteration.

Please note that even if these are generally early stopping conditions, sometimes they can help in the opposite situation. For instance they can also detect that the classifier needs to move some more steps toward the optimal solution than the ones limited by the selected t_{max}.

The proposed stopping criteria could be exploited in the optimization of alternative formulations of the LapSVM problem (following the improved models of Abernethy et al., 2008 and of Tsang and Kwok, 2006), with the aim of reducing training times and getting a classifier with a roughly comparable quality to the optimal one. Even with slightly different problem formulations, our criteria are reasonably more appropriate than classical goal conditions due to their direct relationship with the stability of the classifier prediction. In particular, some additional efficient solution strategies may be devised by directly working in the primal and exploiting the ε-insensitive loss based intrinsic regularizer of Tsang and Kwok (2006), where manifold regularization is applied to a large-scale setting in the Minimum Enclosing Ball (MEB) framework. We note these directions for future work.

5. Laplacian Regularized Least Squares

Laplacian Regularized Least Square Classifier (LapRLSC) has many analogies with the proposed L_2 hinge loss based LapSVMs. LapRLSC uses a squared loss function to penalize wrongly classified examples, leading to the following objective function

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{l} (y_i - f(x_i))^2 + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2.$$

The optimal α coefficients and the optimal bias b, collected in the vector z, can be obtained by solving the linear system

$$\begin{pmatrix} |\mathcal{L}| + \gamma_I 1^T L 1 & 1^T I_{\mathcal{L}} K + \gamma_I 1^T L K \\ K I_{\mathcal{L}} 1 + \gamma_I K L 1 & K I_{\mathcal{L}} K + \gamma_A K + \gamma_I K L K \end{pmatrix} z = \begin{pmatrix} 1^T y \\ K y \end{pmatrix} \tag{14}$$

where $I_{\mathcal{L}}$ is the diagonal matrix $\in \mathbb{R}^{n,n}$ with the first l elements equal to 1 and the remaining u elements equal to zero.

Following the notation used for LapSVMs, in LapRLSCs we have a set of error vectors $\mathcal{E}$ that is actually fixed and equal to $\mathcal{L}$. As a matter of fact a LapRLSC requires the estimated function to interpolate the given targets in order to not incur in a penalty. In a hypothetic situation where all the labeled examples always belong to $\mathcal{E}$ during the training of a LapSVM classifier in the primal, then the solution will be the same of LapRLSC.

Solving the least squares problem of LapRLSC can be performed by matrix inversion, after factoring and simplifying the previously defined matrix P in Equation 14. Otherwise the proposed PCG approach and the early stopping conditions can be directly used. In this case the classic instruments for linear optimization apply, and the required line search of Equation 13 can be computed in closed form without the need of an iterative process,

$$s^* = -\frac{\nabla^T d}{d^T H d}$$

where ∇ and H are no more functions of $\mathcal{E}$.

As shown by Belkin et al. (2006); Sindhwani and Rosenberg (2008) and in the experimental section of this paper, LapRLSC, LapSVM and primal LapSVM allow us to achieve similar classification performances. The interesting property of the LapSVM problem is that the effect of the regularization terms at a given iteration can be decoupled by the one of the loss function on labeled points, since the gradient of the loss function for correctly classified points is zero and do not disturb classifier design. This characteristic can be useful as a starting point for the study of some alternative formulations of the intrinsic norm regularizer.

6. Experimental Results

We ran a wide set of experiments to analyze the proposed solution strategies of the primal LapSVM problem. In this section we describe the selected data sets, our experimental protocol and the details on the parameter selection strategy. Then we show the main result of the proposed approach, very fast training of the LapSVM classifier with reduced complexity by means of early stopped PCG. We compare the quality of the L_2 hinge loss LapSVMs trained in the primal by Newton's method with respect to the L_1 hinge loss dual formulation and LapRLSCs. Finally, we describe the convergence speed and the impact on performances of our early stopping conditions.

As a baseline reference for the performances in the supervised setting, we selected two popular regularized classifiers, Support Vector Machines (SVMs) and Regularized Least Square Classifiers (RLSCs). We implemented and tested all the algorithms using Matlab 7.6 on a 2.33Ghz machine with 6GB of memory. The dual problem of LapSVM has been solved using the latest version of Libsvm (Fan et al., 2005). Multiclass classification has been performed using the one-against-all approach.

6.1 Data Sets

We selected eight popular data sets for our experiments. Most of them data sets has been already used in previous works to evaluate several semi-supervised classification algorithms (Sindhwani et al., 2005; Belkin et al., 2006; Sindhwani and Rosenberg, 2008), and all of them are available on the Web. G50C[3] is an artificial data set generated from two unit covariance normal distributions with equal probabilities. The class means are adjusted so that the Bayes error is 5%. The COIL20 data set is a collection of pictures of 20 different objects from the Columbia University. Each object has been placed on a turntable and at every 5 degrees of rotation a 32x32 gray scale image was acquired. The USPST data set is a collection of handwritten digits form the USPS postal system. Images are acquired at the resolution of 16x16 pixels. USPST refers to the test split of the original data set. We analyzed the COIL20 and USPST data set in their original 20 and 10-class versions and also in their 2-class versions, to discard the effects on performances of the selected multiclass strategy. COIL20(B) discriminates between the first 10 and the last 10 objects, whereas USPST(B) from the first 5 digits and the remaining ones. PCMAC is a two-class data set generated from the famous 20-Newsgroups collection, that collects posts on Windows and Macintosh systems. MNIST3VS8 is the binary version of the MNIST data set, a collection of 28x28 gray scale handwritten digit images from NIST. The goal is to separate digit 3 from digit 8. Finally, the FACEMIT data set of the Center for Biological and Computational Learning at MIT contains 19x19 gray scale, PGM format, images of faces and non-faces. The details of the described data sets are resumed in Table 1.

3. It can be downloaded from `http://people.cs.uchicago.edu/~vikass/manifoldregularization.html`.

Data Set	Classes	Size	Attributes
G50C	2	550	50
COIL20(B)	2	1440	1024
PCMAC	2	1946	7511
USPST(B)	2	2007	256
COIL20	20	1440	1024
USPST	10	2007	256
MNIST3VS8	2	13966	784
FACEMIT	2	31022	361

Table 1: Details of the data sets that have been used in the experiments.

6.2 Experimental Protocol

All presented results has been obtained by averaging them on different splits of the available data. In particular, a 4-fold cross-validation has been performed, randomizing the fold generation process for 3 times, for a total of 12 splits. Each fold contains the same number of per class examples as in the complete data set. For each split, we have 3 folds that are used for training the classifier and the remaining one that constitutes the test set ($\mathcal{T}$). Training data has been divided in labeled ($\mathcal{L}$), unlabeled ($\mathcal{U}$) and validation sets ($\mathcal{V}$), where the last one is only used to tune the classifier parameters. The labeled and validation sets have been randomly selected from the training data such that at least one example per class is assured to be present on each of them, without any additional balancing constraints. A small number of labeled points has been generally selected, in order to simulate a semi-supervised scenario where labeling data has a large cost. The MNIST3VS8 and FACEMIT data set are already divided in training and test data, so that the 4-fold generation process was not necessary, and just the random subdivision of training data has been performed (balancing the class labels on training and validation data). In particular, on the MNIST3VS8 collection we normalized the data vectors to unit norm, and on the FACEMIT data set we exchanged the original training and test sets, since, as a matter of fact, the latter is sensibly larger that the former. In this case our goal is just to show how we were able to handle a high amount of training data using the proposed primal solution with PCG, whereas it was not possible to do it with the original dual formulation of LapSVM. Due to the high unbalancing of such data set, we report the macro error rates for it ($1-TP/2+TN/2$, where TP and TN are the rates of true positives and true negatives). Details are collected in Table 2.

6.3 Parameters

We selected a Gaussian kernel function in the form $k(x_i,x_j) = \exp\left(-\frac{||x_i-x_j||}{2\sigma^2}\right)$ for each experiment, with the exception of the MNIST3VS8 where a polynomial kernel of degree 9 was used, as suggest by Decoste and Schölkopf (2002). The other parameters were selected by cross-validating them on the $\mathcal{V}$ set. In order to speedup this step, the values of the Gaussian kernel width and of the parameters required to build the graph Laplacian (the number of neighbors, *nn*, and the degree, *p*) for the first six data sets were fixed as specified by Sindhwani and Rosenberg (2008). For details on the selection of such parameters please refer to Sindhwani and Rosenberg (2008); Sindhwani et al. (2005). The graph Laplacian was computed by using its normalized expression. The optimal

Data Set	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$
G50C	50	314	50	136
COIL20(B)	40	1000	40	360
PCMAC	50	1358	50	488
USPST(B)	50	1409	50	498
COIL20	40	1000	40	360
USPST	50	1409	50	498
MNIST3VS8	80	11822	80	1984
FACEMIT	2	23973	50	6997

Table 2: The number of data points in each split of the selected data sets, where $\mathcal{L}$ and $\mathcal{U}$ are the sets of labeled and unlabeled training points, respectively, $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set.

weights of the ambient and intrinsic norms, γ_A, γ_I, were determined by varying them on the grid $\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ and chosen with respect to validation error. For the FACEMIT data set also the value 10^{-8} was considered, due to the high amount of training points. The selected parameter values are reported in Table 9 of Appendix A for reproducibility of the experiments.

6.4 Results

Before going into further detail, in Table 3 we report the training times of LapSVMs using the original dual formulation and the primal training approach.[4] The last column refers to LapSVMs trained using the best (in terms of accuracy) of the proposed stopping heuristics for each specific data set. As expected, training in the primal by the Newton's method requires training times similar to those for the dual formulation. On the other hand, training by PCG with the proposed early stopping conditions shows an appreciable reduction of training times for all data sets. As the size of labeled and unlabeled points increases, the improvement becomes very evident. On the MNIST3VS8 data set we go from roughly half an hour to two minutes. Both in the dual formulation of LapSVMs and in the primal one solved by means of Newton's method, a lot of time is spent in computing the LK matrix product. Even if L is sparse, the cost of this product could be quite high. Similar reductions are observed for the PCMAC data set, where the training time drops from 15 seconds to only 2 seconds when solving with PCG. Finally, the memory requirements are also reduced, since, when the PCG is used, there is no need to explicitly compute, store and invert the Hessian. To emphasize this point, we had no difficulty training the classifier on the FACEMIT data set using PCG. On the other hand, the high memory requirements of dual LapSVM and primal LapSVM solved with Newton's method, coupled with the high computational cost, made those methods impossible to runt on our machine.

We now investigate the details of the solution of the primal LapSVM problem. In order to compare the effects of the different loss functions of LapRLSCs, LapSVMs trained in the dual, and LapSVMs trained in the primal, in Table 4 the classification errors of the described techniques are reported. For this comparison, the solution of primal LapSVMs is computed by means of the Newton's method. The manifold regularization based techniques lead to comparable results, and,

4. For a fair comparison of the training algorithms, the Gram matrix and the Laplacian were precomputed.

Data Set	Laplacian SVMs		
	Dual [Original]	**Primal - Newton**	**Primal - PCG**
G50C	0.155 (0.004)	0.134 (0.006)	**0.043** (0.006)
COIL20(B)	0.311 (0.012)	0.367 (0.097)	**0.097** (0.026)
PCMAC	14.82 (0.104)	15.756 (0.285)	**1.967** (0.269)
USPST(B)	1.196 (0.015)	1.4727 (0.2033)	**0.300** (0.030)
COIL20	6.321 (0.441)	7.26 (1.921)	**3.487** (1.734)
USPST	12.25 (0.2)	17.74 (2.44)	**2.032** (0.434)
MNIST3VS8	2064.18 (3.1)	2824.174 (105.07)	**114.441** (0.235)
FACEMIT	-	-	**35.728** (0.868)

Table 3: Our main result. Training times (in seconds) of Laplacian SVMs using different algorithms (standard deviation in brackets). The time required to solve the original dual formulation and the primal solution with Newton's method are comparable, whereas solving the Laplacian SVMs problem in the primal with early stopped preconditioned conjugate gradient (PCG) offers a noticeable speedup.

as expected, all semi-supervised approaches show a sensible improvement over classical supervised classification algorithms. The error rates of primal LapSVMs and LapRLSCs are quite close, due to the described relationship of the L_2 hinge loss and the squared loss. We reported the average number of Newton's steps required to compute the solution in Table 5. In all our experiments we have observed convergence in less than 6 steps.

We compared the error rates of LapSVMs trained in the primal by Newton's method with ones of PCG training, in function of the number of gradient steps t. For this comparison, γ_A and γ_I were selected by cross-validating with the former (see Appendix A), and experiments were performed using all the described data sets. In Figure 5-7 we report the graphs in the case of the USPST, MNIST3VS8 and COIL20 data as a reference. The horizontal line on each graph represents the error rate of the non-approximated solution computed with the Newton's method. The number of iterations required to converge to a solution with the same accuracy of the non-approximated one is sensibly smaller than n. Convergence is achieved really fast, and only in the COIL20 data set we experienced a relatively slower rate with respect to the other data sets. The error surface of each binary classifier is quite flat around optimum with the selected γ_A and γ_I, leading to some round-off errors in gradient descent based techniques, stressed by the large number of classes and the one-against-all approach. Moreover labeled training examples are highly unbalanced. As a matter of fact, in the COIL20(B) data set we did not experience this behavior. Finally, in the FACEMIT data set the algorithm perfectly converges in a few iterations, showing that in this data set the most of information is contained in the labeled data (even if it is very small), and the intrinsic constraint is easily fulfilled.

In Figure 8-9 we collected the values of the gradient norm $\|\nabla\|$, of the preconditioned gradient norm $\|\hat{\nabla}\|$, of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$, and of the objective function obj for each data set, normalized by their respective values at $t = 0$. The vertical line is an indicative index of the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) becomes equal to the one at the stationary point (when the gradient of the objective function is zero). The curves generally keep sen-

Data Set	Classifier	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	SVM	9.33 (2)	9.83 (3.46)	10.06 (2.8)
	RLSC	10.43 (5.26)	10.17 (4.86)	11.21 (4.98)
	LapRLSC	6.03 (1.32)	6.17 (3.66)	6.54 (2.11)
	LapSVM Dual (Original)	5.52 (1.15)	5.67 (2.67)	5.51 (1.65)
	LapSVM Primal (Newton)	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	SVM	16.23 (2.63)	18.54 (6.2)	15.93 (3)
	RLSC	16.22 (2.64)	18.54 (6.17)	15.97 (3.02)
	LapRLSC	8.067 (2.05)	7.92 (3.96)	8.59 (1.9)
	LapSVM Dual (Original)	8.31 (2.19)	8.13 (4.01)	8.68 (2.04)
	LapSVM Primal (Newton)	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
PCMAC	SVM	19.65 (6.91)	20.83 (6.85)	20.09 (6.91)
	RLSC	19.63 (6.91)	20.67 (6.95)	20.04 (6.93)
	LapRLSC	9.67 (0.74)	7.67 (4.08)	9.34 (1.5)
	LapSVM Dual (Original)	10.78 (1.83)	9.17 (4.55)	11.05 (2.94)
	LapSVM Primal (Newton)	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
USPST(B)	SVM	17 (2.74)	18.17 (5.94)	17.1 (3.21)
	RLSC	17.21 (3.02)	17.5 (5.13)	17.27 (2.72)
	LapRLSC	8.87 (1.88)	10.17 (4.55)	9.42 (2.51)
	LapSVM Dual (Original)	8.84 (2.2)	8.67 (4.38)	9.68 (2.48)
	LapSVM Primal (Newton)	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
COIL20	SVM	29.49 (2.24)	31.46 (7.79)	28.98 (2.74)
	RLSC	29.51 (2.23)	31.46 (7.79)	28.96 (2.72)
	LapRLSC	10.35 (2.3)	9.79 (4.94)	11.3 (2.17)
	LapSVM Dual (Original)	10.51 (2.06)	9.79 (4.94)	11.44 (2.39)
	LapSVM Primal (Newton)	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
USPST	SVM	23.84 (3.26)	24.67 (4.54)	23.6 (2.32)
	RLSC	23.95 (3.53)	25.33 (4.03)	24.01 (3.43)
	LapRLSC	15.12 (2.9)	14.67 (3.94)	16.44 (3.53)
	LapSVM Dual (Original)	14.36 (2.55)	15.17 (4.04)	14.91 (2.83)
	LapSVM Primal (Newton)	14.98 (2.88)	15 (3.57)	15.38 (3.55)
MNIST3VS8	SVM	8.82 (1.11)	7.92 (4.73)	8.22 (1.36)
	RLSC	8.82 (1.11)	7.92 (4.73)	8.22 (1.36)
	LapRLSC	1.95 (0.05)	1.67 (1.44)	1.8 (0.3)
	LapSVM Dual (Original)	2.29 (0.17)	1.67 (1.44)	1.98 (0.15)
	LapSVM Primal (Newton)	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
FACEMIT	SVM	39.8 (2.34)	38 (1.15)	34.61 (3.96)
	RLSC	39.8 (2.34)	38 (1.15)	34.61 (3.96)
	LapSVM Primal (PCG)	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 4: Comparison of the accuracy of LapSVMs trained by solving the primal (Newton's method) or the dual problem. The average classification error (standard deviation is reported brackets) is reported. Fully supervised classifiers (SVMs, RLSCs) represent the baseline performances. $\mathcal{U}$ is the set of unlabeled examples used to train the semi-supervised classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all algorithms correctly classify such a few labeled training points.

Data Set	Newton's Steps
G50C	1 (0)
COIL20(B)	2.67 (0.78)
PCMAC	2.33 (0.49)
USPST(B)	4.17 (0.58)
COIL20	2.67 (0.75)
USPST	4.26 (0.76)
MNIST3VS8	5 (0)

Table 5: Newton's steps required to compute the solution of the primal Laplacian SVM problem.

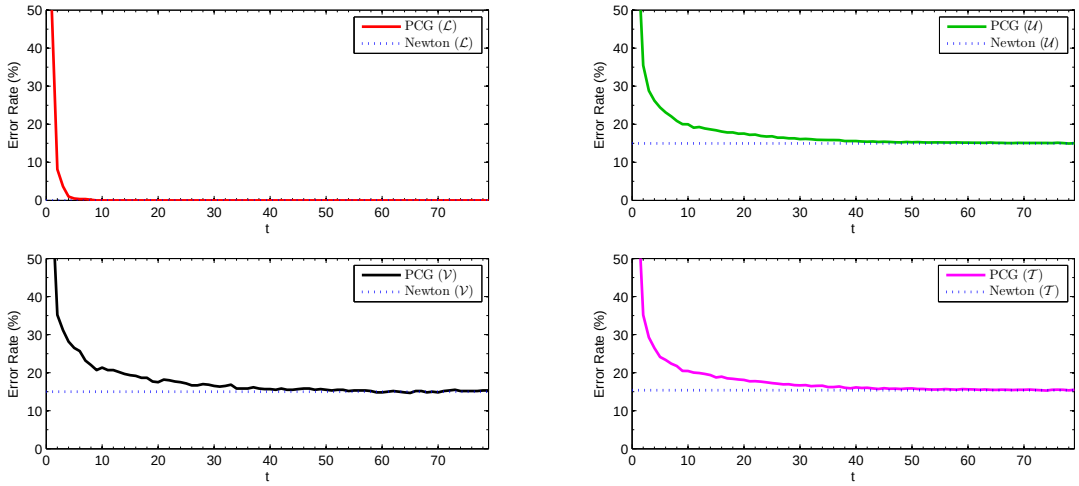


Figure 5: USPST data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

sibly decreasing even after such line, without reflecting real improvements in the classifier accuracy, and they differ by orders of magnitude among the considered data set, showing their strong problem dependency (differently from our proposed conditions). As described in Section 4, we can see how it is clearly impossible to define a generic threshold on them to appropriately stop the PCG descent (i.e., to find a good trade-off between number of iterations and accuracy). Moreover, altering the values of the classifier parameters can sensibly change the shape of the error function, requiring a different threshold every time. In those data sets where points keep entering and leaving the $\mathcal{E}$ set as t increases (mainly during the first steps) the norm of the gradient can show an instable behavior between consecutive iterations, due to the piecewise nature of the problem, making the threshold selection task ulteriorly complex. This is the case of the PCMAC and USPST(B) data set. In the MNIST data, the elements of kernel matrix non belonging to the main diagonal are very small due to the high degree of the polynomial kernel, so that the gradient and the preconditioned gradient are close.

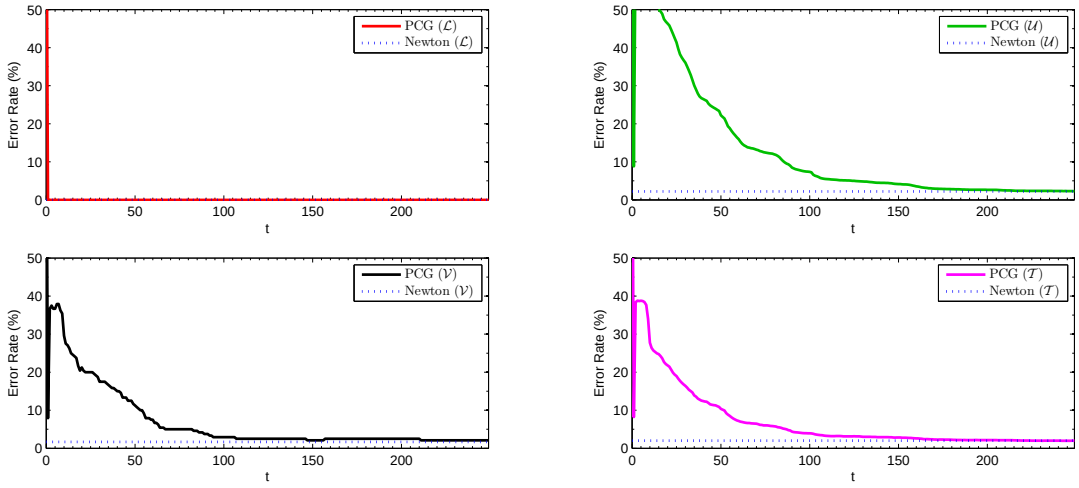


Figure 6: MNIST3VS8 data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

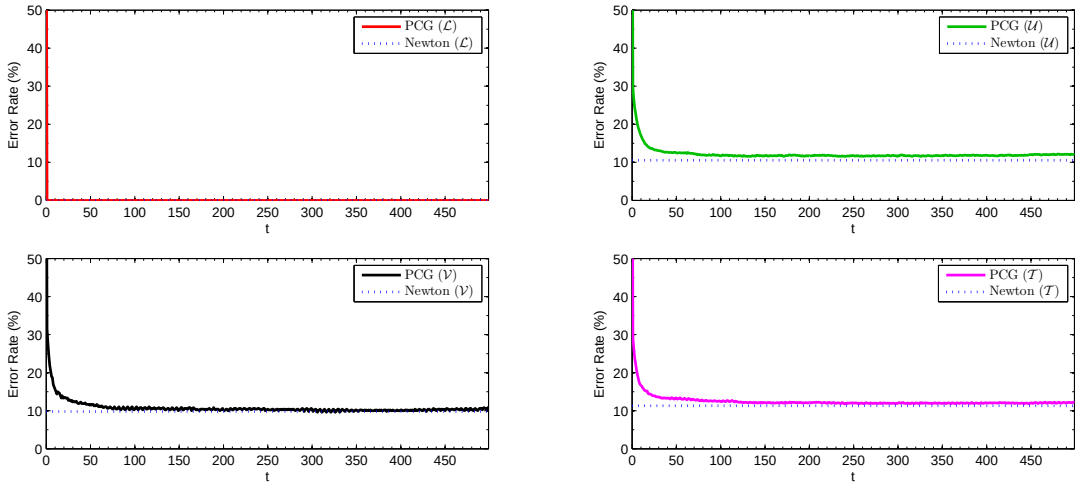


Figure 7: COIL20 data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

Using the proposed PCG goal conditions (Section 4), we cross-validated the primal LapSVM classifier trained by PCG, and the selected parameters are reported in Table 10 of Appendix A. In the USPST(B), COIL20(B), and MNIST3VS8 data sets, larger values for γ_A or γ_I are selected by the validation process, since the convergence speed of PCG is enhanced. In the other data sets, parameter values remain substantially the same of the ones selected by solving with the Newton's

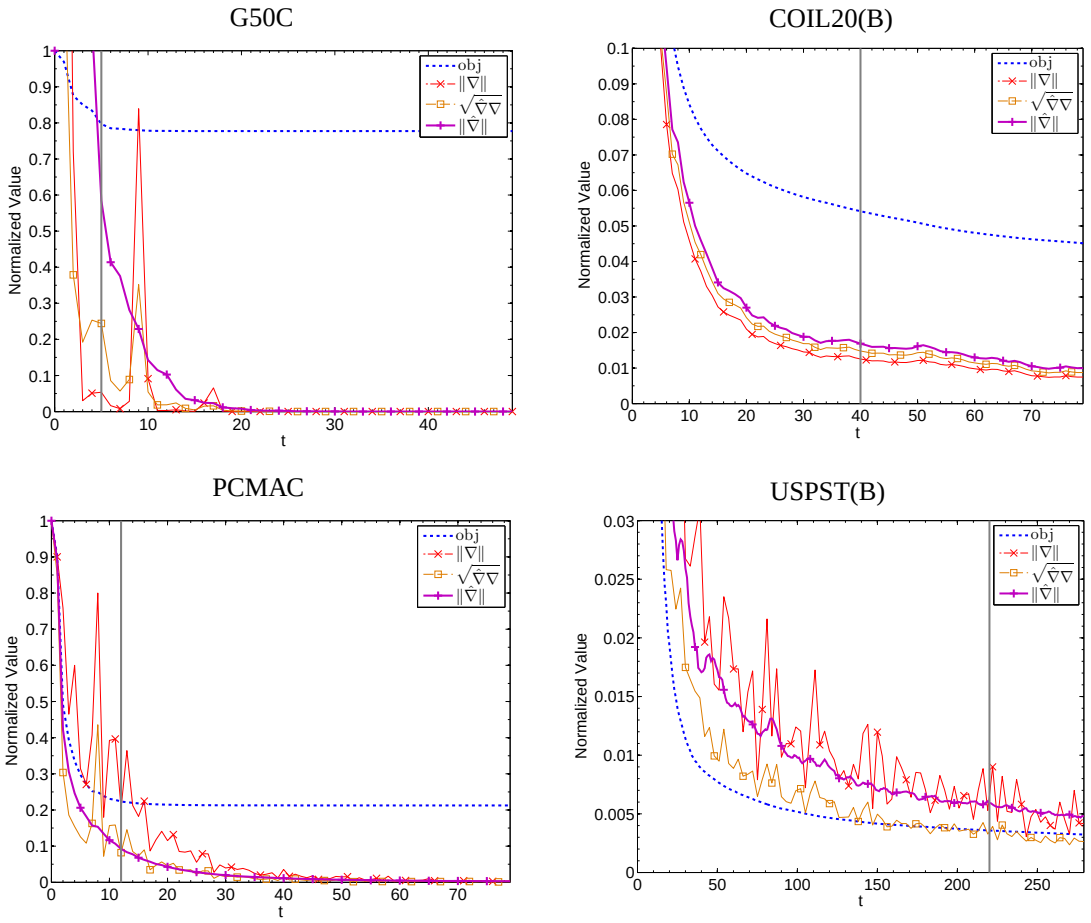


Figure 8: Details of each PCG iteration. The value of the objective function obj, of the gradient norm $\|\nabla\|$, of the preconditioned gradient norm $\|\hat{\nabla}\|$, and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations (t). The vertical line represents the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) is roughly the same to the one at the stationary point.

method, suggesting that a reliable and fast cross-validation can be performed with PCG and the proposed early stopping heuristics.

In Table 6 the training times, the number of PCG and line search iterations are collected, whereas in Table 7 the corresponding classification error rates are reported, for a comparison with the non-approximated solution computed using Newton's method. As already stressed, the training times appreciably drop down when training a LapSVM in the primal using PCG and our goal conditions, independently by the data set. Early stopping allows us to obtain results comparable to the Newton's method or to the original two step dual formulation, showing a direct correlation between the proposed goal conditions and the quality of the classifier. Moreover, our conditions are the same for each problem or data set, overcoming all the issues of the previously described ones. In the COIL20 data set we can observe performances less close to the one of the solution computed with Newton's method. This is due to the already addressed motivations, and it also suggests that the stopping

Data Set	Laplacian SVM	Training Time	PCG Iters	LS Iters
G50C	Dual	0.155 (0.004)	-	-
	Newton	0.134 (0.006)	-	-
	PCG [Stability Check]	**0.044** (0.006)	20 (0)	1 (0)
	PCG [Validation Check]	**0.043** (0.006)	20.83 (2.89)	1 (0)
	PCG [Mixed Check]	**0.044** (0.006)	20.83 (2.89)	1 (0)
COIL20(B)	Dual	0.311 (0.012)	-	-
	Newton	0.367 (0.097)	-	-
	PCG [Stability Check]	**0.198** (0.074)	74.67 (28.4)	2.41 (1.83)
	PCG [Validation Check]	**0.097** (0.026)	37.33 (10.42)	1 (0)
	PCG [Mixed Check]	**0.206** (0.089)	78.67 (34.42)	2.38 (1.79)
PCMAC	Dual	14.8203 (0.104)	-	-
	Newton	15.756 (0.285)	-	-
	PCG [Stability Check]	**1.897** (0.040)	38.00 (0)	1.16 (0.45)
	PCG [Validation Check]	**1.967** (0.269)	39.58 (5.48)	1.15 (0.44)
	PCG [Mixed Check]	**1.997** (0.258)	39.58 (5.48)	1.15 (0.44)
USPST(B)	Dual	1.196 (0.015)	-	-
	Newton	1.4727 (0.2033)	-	-
	PCG [Stability Check]	**0.300** (0.030)	58.58 (5.48)	1.74 (0.90)
	PCG [Validation Check]	**0.281** (0.086)	55.42 (17.11)	1.68 (0.90)
	PCG [Mixed Check]	**0.324** (0.059)	63.33 (12.38)	1.70 (0.89)
COIL20	Dual	6.321 (0.441)	-	-
	Newton	7.26 (1.921)	-	-
	PCG [Stability Check]	**3.297** (1.471)	65.47 (30.35)	2.53 (1.90)
	PCG [Validation Check]	**1.769** (0.299)	34.07 (6.12)	3.37 (2.22)
	PCG [Mixed Check]	**3.487** (1.734)	69.53 (35.86)	2.48 (1.87)
USPST	Dual	12.25 (0.2)	-	-
	Newton	17.74 (2.44)	-	-
	PCG [Stability Check]	**1.953** (0.403)	41.17 (8.65)	3.11 (1.73)
	PCG [Validation Check]	**2.032** (0.434)	42.91 (9.38)	3.13 (1.73)
	PCG [Mixed Check]	**2.158** (0.535)	45.60 (11.66)	3.12 (1.72)
MNIST3VS8	Dual	2064.18 (3.1)	-	-
	Newton	2824.174 (105.07)	-	-
	PCG [Stability Check]	**114.441** (0.235)	110 (0)	5.58 (2.79)
	PCG [Validation Check]	**124.69** (0.335)	110 (0)	5.58 (2.79)
	PCG [Mixed Check]	**124.974** (0.414)	110 (0)	5.58 (2.79)
FACEMIT	PCG [Stability Check]	**35.728** (0.868)	3 (0)	1 (0)
	PCG [Validation Check]	**35.728** (0.868)	3 (0)	1 (0)
	PCG [Mixed Check]	**35.728** (0.868)	3 (0)	1 (0)

Table 6: Training time comparison among the Laplacian SVMs trained in the dual (Dual), LapSVM trained in the primal by means of Newton's method (Newton) and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). Average training times (in seconds) and their standard deviations, the number of PCG iterations, and of Line Search (LS) iterations (per each PCG one) are reported.

Data Set	**Laplacian SVM**	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	Newton	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Stability Check]	6.13 (1.46)	6.17 (3.46)	7.27 (2.87)
	PCG [Validation Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Mixed Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	Newton	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
	PCG [Stability Check]	8.81 (2.23)	8.13 (3.71)	8.84 (1.93)
	PCG [Validation Check]	8.32 (2.28)	8.96 (4.05)	8.45 (1.58)
	PCG [Mixed Check]	8.84 (2.28)	8.13 (3.71)	8.84 (1.96)
PCMAC	Newton	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
	PCG [Stability Check]	9.65 (0.78)	7.83 (4.04)	9.42 (1.50)
	PCG [Validation Check]	9.67 (0.76)	7.83 (4.04)	9.40 (1.50)
	PCG [Mixed Check]	9.67 (0.76)	7.83 (4.04)	9.40 (1.50)
USPST(B)	Newton	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
	PCG [Stability Check]	9.11 (2.14)	10.50 (4.36)	9.70 (2.55)
	PCG [Validation Check]	9.10 (2.17)	10.50 (4.36)	9.75 (2.59)
	PCG [Mixed Check]	9.09 (2.17)	10.50 (4.36)	9.70 (2.55)
COIL20	Newton	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
	PCG [Stability Check]	12.42 (2.68)	10.63 (4.66)	12.92 (2.14)
	PCG [Validation Check]	13.07 (2.73)	12.08 (4.75)	13.52 (2.12)
	PCG [Mixed Check]	12.43 (2.69)	10.42 (4.63)	12.87 (2.20)
USPST	Newton	14.98 (2.88)	15 (3.57)	15.38 (3.55)
	PCG [Stability Check]	15.60 (3.45)	15.67 (3.60)	16.11 (3.95)
	PCG [Validation Check]	15.40 (3.38)	15.67 (3.98)	15.94 (4.04)
	PCG [Mixed Check]	15.45 (3.53)	15.50 (3.92)	15.94 (4.08)
MNIST3VS8	Newton	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
	PCG [Stability Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
	PCG [Validation Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
	PCG [Mixed Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
FACEMIT	PCG [Stability Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Validation Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Mixed Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 7: Average classification error (standard deviation is reported brackets) of Laplacian SVMs trained in the primal by means of Newton's method (Newton) and of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). $\mathcal{U}$ is the set of unlabeled examples used to train the classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all algorithms correctly classify such a few labeled training points.

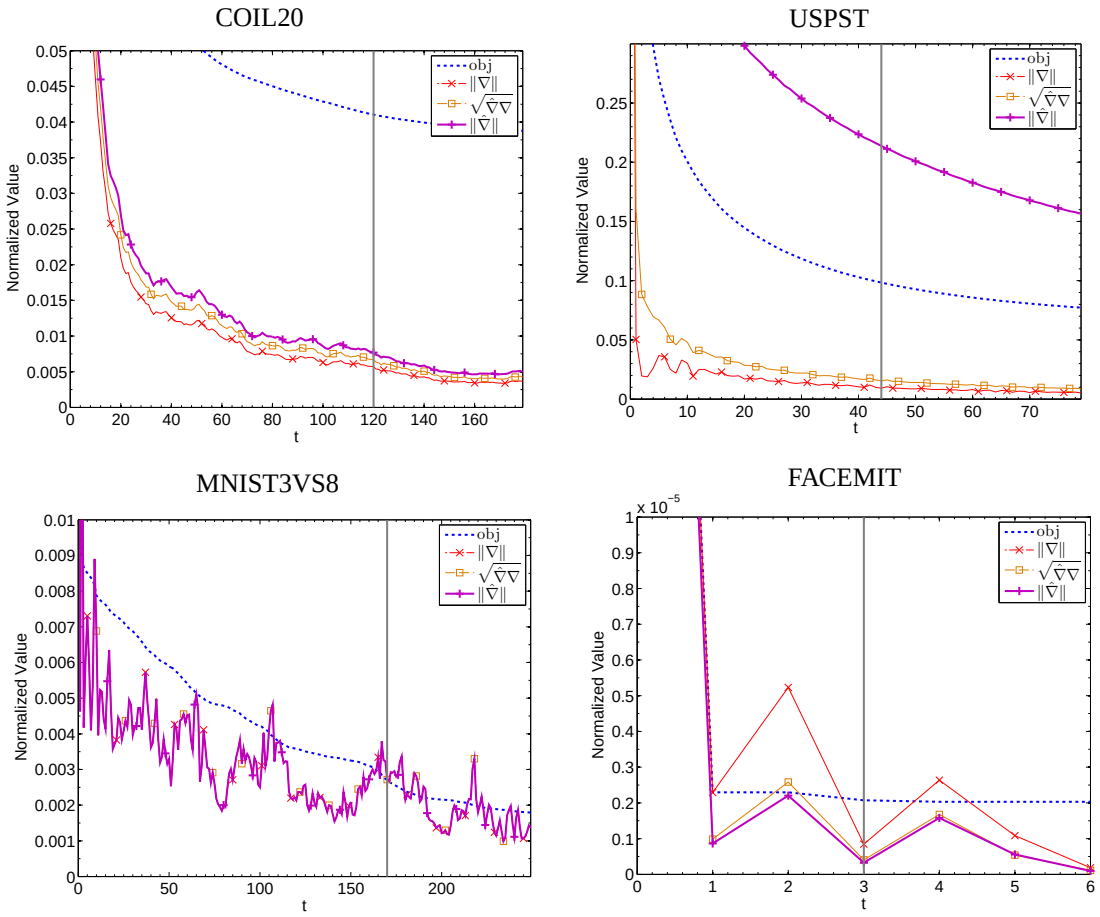


Figure 9: Details of each PCG iteration. The value of the objective function obj, of the gradient norm $\|\nabla\|$, of the preconditioned gradient norm $\|\hat{\nabla}\|$, and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations (t). The vertical line represents the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) is roughly the same to the one at the stationary point.

condition should probably be checked while training in parallel the 20 binary classifiers, instead of separately checking it on each of them. A better tuning of the goal conditions or a different formulation of them can move the accuracy closer to the one of primal LapSVM trained with Newton's method, but it goes beyond to the scope of this paper.

The number of PCG iterations is noticeably smaller than n. Obviously it is function of the gap between each checking of a stopping criterion, that we set to $\sqrt{n}/2$. The number of iterations from the stability check is sometimes larger that the one from the validation check (COIL20(B), USPST, COIL20). As a matter of fact, labeled validation data is more informative than a stable, but unknown, decision on the unlabeled one. On the other hand validation data could not represent test data enough accurately. Using a mixed strategy makes sense in those cases, as can be observed in the COIL20 data set. In our experiments the mixed criterion has generally the same behavior of the most strict of the two heuristics for each specific set of data. In the FACEMIT data set complete

Data Set	Laplacian RLSC	Training Time	PCG Iters	$\mathcal{T}$
PCMAC	Matrix Inversion	14.21 (0.067)	-	9.34 (1.5)
	PCG [Stability Check]	**1.818** (0.016)	38 (0)	9.34 (1.46)
	PCG [Validation Check]	**1.82** (0.05)	38 (0)	9.34 (1.46)
	PCG [Mixed Check]	**1.821** (0.047)	38 (0)	9.34 (1.46)

Table 8: Training time comparison among the Laplacian RLSCs trained by solving Equation 14 with matrix inversion and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). Average training times (in seconds), the number of PCG iterations, and the average classification error on test data $\mathcal{T}$ are shown. Standard deviations are reported brackets.

convergence is achieved in just a few iterations, independently by the heuristics. The number of line search iterations is usually very small and negligible with respect to the computational cost of the training algorithm.

For the sake of completeness, we show an example of the application of our early stopped PCG to LapRLSC, as described in Section 5. In Table 8 we report the training times, the PCG iterations, and the error rate (on test points) in the case of PCMAC data. The reduction of training times is significant, and positively influenced by the non iterative line search procedure.

7. Conclusions and Future Work

In this paper we described investigated in detail two strategies for solving the optimization problem of Laplacian Support Vector Machines (LapSVMs) in the primal. A very fast solution can be achieved using preconditioned conjugate gradient coupled with an early stopping criterion based on the stability of the classifier decision. Detailed experimental results on real world data show the validity of such strategy. The computational cost for solving the problem reduces from $O(n^3)$ to $O(kn^2)$, where n is the total number of training points, both labeled and unlabeled, and k is empirically evaluated to be significantly smaller than n, without the need of storing in memory the Hessian matrix and its inverse. Training times are significantly reduced on all selected benchmarks, in particular, as the amount of training data increases. This solution can be a useful starting point for applying greedy techniques for incremental classifier building or for studying the effects of a sparser kernel expansion of the classification function. Moreover, some recently proposed domain decomposition techniques for large scale RLSC (Li et al., 2007) could be investigated to solve the primal LapSVM problem, that we will address in future work.

Acknowledgments

We would like to acknowledge the NSF for supporting this research.

Appendix A.

This Appendix collects all the parameters selected using our experimental protocol, for reproducibility of the experiments (Table 9 and Table 10). Details of the cross-validation procedure are described in Section 6.

In the most of the data sets, parameter values selected using the PCG solution remain substantially the same of the ones selected by solving the primal problem with the Newton's method, suggesting that a reliable and fast cross-validation can be performed with PCG and the proposed early stopping heuristics. In the USPST(B), COIL20(B), and MNIST3VS8 data sets, larger values for γ_A or γ_I are selected when using PCG, since the convergence speed of gradient descent is enhanced.

To emphasize this behavior, the training times and the resulting error rates of the PCG solution computed using γ_A and γ_I tuned by means of the Newton's method (instead of the ones computed by PCG with each specific goal condition) are reported in Table 11 and in Table 12. Comparing these results with the ones presented in Section 6, it can be appreciated that both the convergence speed (Table 6) and the accuracy of the PCG solution (Table 7) benefit from an appropriate parameter selection. Note that the performance gaps between Newton's method and PCG of a given data set sometimes are slightly different among $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{T}$. As a matter of fact, the balancing of class labels may not be exactly the same among the three sets, due to the random sampling of $\mathcal{V}$ (and $\mathcal{L}$) from non-test data, as described in Section 6.

References

J. Abernethy, O. Chapelle, and C. Castillo. Witch: A new approach to web spam detection. *Technical Report 2008-001, Yahoo! Research*, 2008.

M. Belkin and P. Niyogi. Using manifold stucture for partially labeled classification. *Advances in Neural Information Processing Systems*, pages 953–960, 2003.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7: 2399–2434, 2006.

S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 585–592. Cambridge, MA, USA: MIT Press, 2003.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. MIT press, 2006.

O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

Data Set	Classifier	σ	nn	p	γ_A	γ_I
G50C	SVM	17.5	-	-	10^{-1}	-
	RLSC	17.5	-	-	1	-
	LapRLSC	17.5	50	5	10^{-6}	10^{-2}
	LapSVM Dual (Original)	17.5	50	5	1	10
	LapSVM Primal (Newton)	17.5	50	5	10^{-1}	10
COIL20(B)	SVM	0.6	-	-	10^{-6}	-
	RLSC	0.6	-	-	10^{-6}	-
	LapRLSC	0.6	2	1	10^{-6}	1
	LapSVM Dual (Original)	0.6	2	1	10^{-2}	100
	LapSVM Primal (Newton)	0.6	2	1	10^{-6}	1
PCMAC	SVM	2.7	-	-	10^{-6}	-
	RLSC	2.7	-	-	10^{-6}	-
	LapRLSC	2.7	50	5	10^{-6}	10^{-2}
	LapSVM Dual (Original)	2.7	50	5	10^{-6}	10^{-4}
	LapSVM Primal (Newton)	2.7	50	5	10^{-6}	1
USPST(B)	SVM	9.4	-	-	10^{-6}	-
	RLSC	9.4	-	-	10^{-1}	-
	LapRLSC	9.4	10	2	10^{-4}	10^{-1}
	LapSVM Dual (Original)	9.4	10	2	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9.4	10	2	10^{-6}	10^{-2}
COIL20	SVM	0.6	-	-	10^{-6}	-
	RLSC	0.6	-	-	10^{-6}	-
	LapRLSC	0.6	2	1	10^{-6}	1
	LapSVM Dual (Original)	0.6	2	1	10^{-6}	10
	LapSVM Primal (Newton)	0.6	2	1	10^{-6}	1
USPST	SVM	9.4	-	-	10^{-1}	-
	RLSC	9.4	-	-	10^{-6}	-
	LapRLSC	9.4	10	2	10^{-6}	10^{-1}
	LapSVM Dual (Original)	9.4	10	2	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9.4	10	2	10^{-4}	1
MNIST3VS8	SVM	9	-	-	10^{-6}	-
	RLSC	9	-	-	10^{-6}	-
	LapRLSC	9	20	3	10^{-6}	10^{-2}
	LapSVM Dual (Original)	9	20	3	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9	20	3	10^{-6}	10^{-2}
FACEMIT	SVM	4.3	-	-	10^{-6}	-
	RLSC	4.3	-	-	10^{-6}	-
	LapSVM Primal (PCG)	4.3	6	1	10^{-6}	10^{-8}

Table 9: Parameters selected by cross-validation for supervised algorithms (SVM, RLSC) and semi-supervised ones based on manifold regularization, using different loss functions (LapRLSC, LapSVM trained in the dual formulation and in the primal one by means of Newton's method). The parameter σ is the bandwidth of the Gaussian kernel or, in the MNIST3VS8, the degree of the polynomial one.

Data Set	**Laplacian SVM**	γ_A	γ_I
G50C	Newton	10^{-1}	10
	PCG [Stability Check]	10^{-1}	10
	PCG [Validation Check]	10^{-1}	10
	PCG [Mixed Check]	10^{-1}	10
COIL20(B)	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	1	100
	PCG [Mixed Check]	10^{-6}	1
PCMAC	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-4}	1
	PCG [Validation Check]	10^{-4}	1
	PCG [Mixed Check]	10^{-6}	10^{-1}
USPST(B)	Newton	10^{-6}	10^{-2}
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	10^{-6}	1
	PCG [Mixed Check]	10^{-6}	1
COIL20	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	10^{-6}	1
	PCG [Mixed Check]	10^{-6}	1
USPST	Newton	10^{-4}	1
	PCG [Stability Check]	10^{-4}	1
	PCG [Validation Check]	10^{-4}	1
	PCG [Mixed Check]	10^{-4}	1
MNIST3VS8	Newton	10^{-6}	10^{-2}
	PCG [Stability Check]	10^{-6}	10^{-1}
	PCG [Validation Check]	10^{-6}	10^{-1}
	PCG [Mixed Check]	10^{-6}	10^{-1}
FACEMIT	PCG [Stability Check]	10^{-6}	10^{-8}
	PCG [Validation Check]	10^{-6}	10^{-8}
	PCG [Mixed Check]	10^{-6}	10^{-8}

Table 10: A comparison of the parameters selected by cross-validation for Laplacian SVMs trained in the primal by means of Newton's method (Newton) and preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets).

Data Set	Laplacian SVM	Training Time	PCG Iters	LS Iters
G50C	Dual	0.155 (0.004)	-	-
	Newton	0.134 (0.006)	-	-
	PCG [Stability Check]	0.044 (0.006)	20 (0)	1 (0)
	PCG [Validation Check]	0.043 (0.006)	20.83 (2.89)	1 (0)
	PCG [Mixed Check]	0.044 (0.006)	20.83 (2.89)	1 (0)
COIL20(B)	Dual	0.311 (0.012)	-	-
	Newton	0.367 (0.097)	-	-
	PCG [Stability Check]	0.198 (0.074)	74.67 (28.4)	2.41 (1.83)
	PCG [Validation Check]	0.095 (0.018)	36 (7.24)	3.26 (2.21)
	PCG [Mixed Check]	0.206 (0.089)	78.67 (34.42)	2.38 (1.79)
PCMAC	Dual	14.8203 (0.104)	-	-
	Newton	15.756 (0.285)	-	-
	PCG [Stability Check]	1.901 (0.022)	38.00 (0)	1.18 (0.45)
	PCG [Validation Check]	1.970 (0.265)	39.58 (5.48)	1.18 (0.44)
	PCG [Mixed Check]	1.969 (0.268)	39.58 (5.48)	1.18 (0.44)
USPST(B)	Dual	1.196 (0.015)	-	-
	Newton	1.4727 (0.2033)	-	-
	PCG [Stability Check]	0.496 (0.172)	95.00 (33.40)	6.56 (3.18)
	PCG [Validation Check]	0.279 (0.096)	52.25 (18.34)	6.83 (3.44)
	PCG [Mixed Check]	0.567 (0.226)	107.67 (43.88)	6.49 (3.15)
COIL20	Dual	6.321 (0.441)	-	-
	Newton	7.26 (1.921)	-	-
	PCG [Stability Check]	3.297 (1.471)	65.47 (30.35)	2.53 (1.90)
	PCG [Validation Check]	1.769 (0.299)	34.07 (6.12)	3.37 (2.22)
	PCG [Mixed Check]	3.487 (1.734)	69.53 (35.86)	2.48 (1.87)
USPST	Dual	12.25 (0.2)	-	-
	Newton	17.74 (2.44)	-	-
	PCG [Stability Check]	1.953 (0.403)	41.17 (8.65)	3.11 (1.73)
	PCG [Validation Check]	2.032 (0.434)	42.91 (9.38)	3.13 (1.73)
	PCG [Mixed Check]	2.158 (0.535)	45.60 (11.66)	3.12 (1.72)
MNIST3VS8	Dual	2064.18 (3.1)	-	-
	Newton	2824.174 (105.07)	-	-
	PCG [Stability Check]	188.775 (0.237)	165 (0)	6.78 (3.65)
	PCG [Validation Check]	207.986 (35.330)	183.33 (31.75)	6.65 (3.57)
	PCG [Mixed Check]	207.915 (35.438)	183.33 (31.75)	6.65 (3.57)
FACEMIT	PCG [Stability Check]	35.728 (0.868)	3 (0)	1 (0)
	PCG [Validation Check]	35.728 (0.868)	3 (0)	1 (0)
	PCG [Mixed Check]	35.728 (0.868)	3 (0)	1 (0)

Table 11: Training time comparison among the Laplacian SVMs trained in the dual (Dual), LapSVM trained in the primal by means of Newton's method (Newton) and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). *Parameters of the classifiers were tuned using the Newton's method*. Average training times (in seconds) and their standard deviations, the number of PCG iterations, and of Line Search (LS) iterations (per each PCG one) are reported.

Data Set	**Laplacian SVM**	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	Newton	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Stability Check]	6.13 (1.46)	6.17 (3.46)	7.27 (2.87)
	PCG [Validation Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Mixed Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	Newton	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
	PCG [Stability Check]	8.81 (2.23)	8.13 (3.71)	8.84 (1.93)
	PCG [Validation Check]	8.97 (2.32)	9.17 (3.74)	8.96 (1.64)
	PCG [Mixed Check]	8.84 (2.28)	8.13 (3.71)	8.84 (1.96)
PCMAC	Newton	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
	PCG [Stability Check]	9.65 (0.76)	7.83 (4.04)	9.42 (1.43)
	PCG [Validation Check]	9.65 (0.76)	7.83 (4.04)	9.40 (1.43)
	PCG [Mixed Check]	9.65 (0.76)	7.83 (4.04)	9.40 (1.43)
USPST(B)	Newton	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
	PCG [Stability Check]	11.07 (2.27)	13.33 (4.21)	11.49 (2.55)
	PCG [Validation Check]	12.02 (2.22)	14.67 (2.99)	12.01 (2.14)
	PCG [Mixed Check]	10.81 (2.39)	12.83 (4.78)	11.31 (2.71)
COIL20	Newton	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
	PCG [Stability Check]	12.42 (2.68)	10.63 (4.66)	12.92 (2.14)
	PCG [Validation Check]	13.07 (2.73)	12.08 (4.75)	13.52 (2.12)
	PCG [Mixed Check]	12.43 (2.69)	10.42 (4.63)	12.87 (2.20)
USPST	Newton	14.98 (2.88)	15 (3.57)	15.38 (3.55)
	PCG [Stability Check]	15.60 (3.45)	15.67 (3.60)	16.11 (3.95)
	PCG [Validation Check]	15.40 (3.38)	15.67 (3.98)	15.94 (4.04)
	PCG [Mixed Check]	15.45 (3.53)	15.50 (3.92)	15.94 (4.08)
MNIST3VS8	Newton	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
	PCG [Stability Check]	3.16 (0.15)	2.5 (1.25)	2.4 (0.38)
	PCG [Validation Check]	2.89 (0.62)	2.50 (1.25)	2.37 (0.44)
	PCG [Mixed Check]	2.89 (0.62)	2.5 (1.25)	2.37 (0.44)
FACEMIT	PCG [Stability Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Validation Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Mixed Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 12: Average classification error (standard deviation is reported brackets) of Laplacian SVMs trained in the primal by means of Newton's method and of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). *Parameters of the classifiers were tuned using the Newton's method*. $\mathcal{U}$ is the set of unlabeled examples used to train the classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all classifiers perfectly fit such few labeled training points.

D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46 (1):161–190, 2002.

A. Demiriz and K. Bennett. Optimization approaches to semi-supervised learning. *Complementarity: Applications, Algorithms and Extensions*, 50:1–19, 2000.

R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209, 1999.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning*, volume 20, pages 290–297, 2003.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM New York, NY, USA, 2006.

S.S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *The Journal of Machine Learning Research*, 6(1):341–361, 2005.

S.S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *The Journal of Machine Learning Research*, 7:1493–1515, 2006.

W. Li, K.-H. Lee, and K.-S. Leung. Large-scale RLSC learning without agony. In *Proceedings of the 24th International Conference on Machine learning*, pages 529–536, New York, NY, USA, 2007. ACM.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton methods for large-scale logistic regression. *The Journal of Machine Learning Research*, 9:627–650, 2008.

O. L. Mangasarian. A finite newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

M. Seeger. Low rank updates for the Cholesky decomposition. *Department of EECS, University of California at Berkeley, Technical Report*, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the International Conference on Machine Learning*, pages 807–814, 2007.

J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science, Carnegie Mellon University, Techical Report*, 1994.

V. Sindhwani. *On Semi-supervised Kernel Methods*. PhD thesis, University of Chicago, 2007.

V. Sindhwani and D.S. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Proceedings of the International Conference on Machine Learning*, pages 976–983, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, volume 22, pages 825–832, 2005.

I.W. Tsang and J.T. Kwok. Large-scale sparsified manifold regularization. *Advances in Neural Information Processing Systems*, 19:1401–1408, 2006.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

X. Zhu and A.B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, 2009.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.