JMLR: Workshop and Conference Proceedings 29:1–16, 2013 ACML 2013

Multiclass Latent Locally Linear Support Vector Machines

Editor: Cheng Soon Ong and Tu Bao Ho

Abstract

As the big data paradigm is gaining momentum, kernelized SVM has started losing its status of off-the-shelf classifier able to deliver state of the art performance, due to its computational and memory efficiency limitations. A growing, promising trend is that of developing linear algorithms able to learn non linear decision functions through the use of locality. In this paper we propose a new multi class local learning classifier, based on a latent SVM formulation. As opposed to previous works based on manifold embedding, we learn the non linear classification function in the original space, as a local linear combination of linear ones. To this end, we directly embed the local weights as latent variables of the scoring function and we call the resulting algorithm ML3, Multi class Latent Locally Linear SVM. Extensive experiments on ten standard UCI machine learning datasets, three hand written character and digit recognition databases, and a visual place categorization dataset, show the power of the proposed approach.

1. Introduction

Over the last 15 years, Support Vector Machines (SVMs) have become one of the most powerful tools for classification and the de facto standard in several fields. A large part of this success is due to the use of *kernel functions*. Still, kernelized SVMs (and learning with kernels in general) do not scale well with the number of samples. As the amount of data available for training is quickly moving to unprecedented scales in several domains, there is a growing need for efficient learning methods, whose training complexity with respect to the number of samples is not super-linear.

Linear SVMs, for example trained with fast stochastic gradient descent algorithms, would satisfy such complexity requirements, unfortunately to the expense of a disappointing performance. To try to address this issue, local learning SVM-based methods have received increasing attention, both in the kernel learning community (Gönen and Alpaydin, 2008) and in the linear learning community (Zhang et al., 2006; Yu et al., 2009; Ladicky and Torr, 2011). A key feature of such methods is the ability to exploit the structure of the data to learn specific models in different zones of the input space. Performance-wise, when coupling local methods with (infinite) kernels, the improvement over non-local versions of the algorithms is usually relatively small (Gönen and Alpaydin, 2008), because the boundary is already flexible enough to separate any training set. However, when combined with linear classifiers they can lead to large improvements, thanks to the increased flexibility of the separation surface between the classes.

This paper contributes to this research thread. Our focus is on enhancing linear algorithms to obtain the complex decision functions traditionally given by kernels, through the use of locality. We propose a multi class local learning algorithm based on a latent SVM formulation. For each sample, during training as well as during testing, we use a locally weighted combination of linear models (as in Yu et al. (2009) and Ladicky and Torr (2011)). Our idea is to locally find a good combination of models that maximizes the confidence of the full model on each sample. The local weights can thus be modeled as latent variables of the scoring function (Felzenszwalb et al., 2010), resulting in each linear model being trained only on a local subset of the input space. As opposed to previous methods, we do not require a 2-stage formulation, i.e. our approach does not require to first learn the manifold using a reconstruction (or soft-assignment) technique and then learn a linear SVM in the manifold, nor any nearest-neighbor search. Our algorithm is trained in a winner-take-all competitive multi class fashion, so that each class tries to maximize its score on each sample by using a locally optimal combination of models, competing with the others in the training process. Moreover, compared to standard latent SVM, our formulation allows to use soft local combinations of models, where the sparsity of the local combinations, and thus the smoothness of the solution, is tuned using a p-norm constraint. This is achieved by treating the score maximization as a convex optimization problem, whose solution is shown to be efficiently computable using a closed-form expression. Using this closed form solution, we also obtain a testing prediction rule in which the local weights do not need to be explicitly computed. We call our method Multi class Latent Locally Linear Support Vector Machine (ML3).

Experiments on real and synthetic data illustrate how ML3 behaves as the p-norm constraints are varied. We also compare its performance and speed to previously proposed approaches, on ten UCI machine learning datasets (Frank and Asuncion, 2010) (for the binary case), three hand-written character and digit recognition databases (MNIST (LeCun et al., 1998), USPS (Hull, 1994) and LETTER (Frank and Asuncion, 2010)), and an indoor visual scene categorization dataset (Quattoni and Torralba, 2009). Results consistently show the value of our method.

An outline of the paper is as follows. In Section 2 we review previous work. Section 3 defines the algorithm, discusses its properties, what contributions it brings to the field, and its optimization procedure (Section 3.1). In Section 4 we report experiments on synthetic data showing the behavior of ML3 varying its parameters, and in Section 5 we show the results of benchmarking ML3 against other approaches. We conclude in Section 6, pointing out some possible future avenues for research. In the Appendix we report the derivation of the closed-form solution for the computation of the local weights.

2. Related works

The appealing statistical properties of local classifiers have first been analyzed in Vapnik (1991). The idea is that the capacity of a classifier should match the density of the training samples in a specific area of the instance space: low-density areas of the input space would require a low-capacity classifier, while more populated zones would benefit from a high capacity one. Such localization could be achieved by either using a separate classifier with a specific capacity in each area, or by building a set of classifiers with the same capacity, but constrained to have access to different amounts of training samples originated in different parts of the space. Following the second approach, many successful algorithms have been proposed. Bottou and Vapnik (1992) proposed to train a linear classifier on the k-Nearest Neighbors of a testing sample and then use it to label the sample; Yang and Kecman (2008) introduced a properly weighted Euclidean distance for the k-NN computation, while Zhang et al. (2006) and Kecman and Brooks (2010) used a linear (and non-linear) SVM as the local classifier. These simple local models perform surprisingly well in practical applications. However, due to the k-NN search and the local training that has to be performed for each testing sample, such models are slow to test and quite inefficient at large scale problems.

Arguably, the most popular form of local classifiers is represented by kernel methods. For example, when classifying an instance with a Gaussian kernel SVM, only the support vectors located in a neighborhood of the test sample will significantly contribute to the prediction. Also, when using linear kernels, non-linear functions can be directly modeled as a local linear combination of linear ones, with a gating function assigning a weight to each model, for each sample (Gönen and Alpaydin, 2008). While the testing time of such methods is linear in the number of support vectors, their training complexity grows cubically with the training set size.

More recently, manifold learning methods have also been proposed to approximate non-linear functions, using a local combination of linear ones. For example, in Yu et al. (2009), the combination coefficients are given by the reconstruction coordinates obtained using Local Coordinate Coding (LCC). In Locally Linear SVM (LLSVM), Ladicky and Torr (2011) make use of inverse Euclidean distances as a form of manifold learning, while also learning all the local models in a single optimization problem. LLSVM outperforms LCC both in terms of number of anchor points needed (hundreds instead of thousands) and accuracy. This approach was further improved in Zhang et al. (2011), by combining it with a more sophisticated manifold learning scheme, named Orthogonal Coordinate Coding (OCC). In Qi et al. (2011) a hashing function is used to group samples with the same hash and to train a separate model for each hashing value. To smooth the resulting irregular piecewise-linear boundary, the authors also introduce a "global reference classifier", which is additionally used to classify test samples with unknown hashes. Although efficient, all methods in this last group require to learn the manifold, and some kind of encoding for all the samples. Moreover, they either require a very large number of models (Yu et al., 2009), or are prone to overfitting when the number of models increases (Ladicky and Torr, 2011). This is likely due to the fact that the manifold learning procedure, which determines the sample-to-model assignments, is performed as a separate step and is unaware of the classification task.

3. The ML3 algorithm

Our aim is to directly learn a smooth non-linear classification function in the original space, as a local linear combination of linear ones. Here, instead of having a manifold learning procedure separated from the classifier training, we directly embed the local weights as latent variables of the scoring function, in a latent SVM framework (Felzenszwalb et al., 2010; Yu and Joachims, 2009). This choice is motivated by the intuition that, if locally trained, the most confident sub-models are the most useful in predicting the label of a testing sample. Our model training is carried out as a coordinate-descent procedure, where in each step we minimize an objective function w.r.t. some of the variables, keeping the others fixed. We use stochastic gradient descent, with an adapted version of Shalev-Shwartz et al. (2007), and call the resulting algorithm ML3: “Multi class Latent Locally Linear” SVM.

Latent SVM. Latent SVMs were initially motivated and introduced in the field of computer vision to solve object detection (Felzenszwalb et al., 2008) and action recognition (Wang and Mori, 2009) tasks. A latent SVM makes use of scoring functions of the form:

$$s_{\boldsymbol{v}}(\boldsymbol{x}_i) = \max_{\boldsymbol{\beta} \in B(x_i)} \boldsymbol{v}^\top \phi(\boldsymbol{x}_i, \boldsymbol{\beta}), \tag{1}$$

where $B(x_i)$ defines the set of possible latent variables for sample $\boldsymbol{x}_i$. The scoring function (1) is then used in a classical primal SVM objective function:

$$L(\boldsymbol{v}, \boldsymbol{X}, \boldsymbol{y}) = \frac{\lambda}{2}\|\boldsymbol{v}\|^2 + \sum_{i=1}^{n} |1 - y_i s_{\boldsymbol{v}}(\boldsymbol{x}_i)|_+ \,, \tag{2}$$

where $|x|_+ = \max(x, 0)$. Due to the maximization in the scoring function, $L(\boldsymbol{v}, \boldsymbol{X}, \boldsymbol{y})$ is not convex anymore w.r.t. $\boldsymbol{v}$. However, if the latent variables for the positive samples $\boldsymbol{\beta}_p$ are fixed, the modified objective function $L(\boldsymbol{v}, \boldsymbol{\beta}_p, \boldsymbol{X}, \boldsymbol{y})$ becomes convex again. Hence, using an alternating optimization strategy that minimizes $L(\boldsymbol{v}, \boldsymbol{\beta}_p, \boldsymbol{X}, \boldsymbol{y})$ w.r.t. to $\boldsymbol{v}$ and then maximizes the latent variables for the positive samples, the procedure is guaranteed to converge to a local minima.

Multiclass Latent Locally Linear SVM. Based on the Latent SVM formulation, we now derive our algorithm. First, we define the mapping ϕ and weight vector $\boldsymbol{v}$ as

$$\phi(\boldsymbol{x}_i, \boldsymbol{\beta}) \triangleq \begin{bmatrix} \beta_1 \boldsymbol{x}_i \\ \beta_2 \boldsymbol{x}_i \\ \dots \\ \beta_m \boldsymbol{x}_i \end{bmatrix}, \tag{3}$$

$$\boldsymbol{v} \triangleq \left[\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \dots, \boldsymbol{w}_m^\top\right]^\top, \tag{4}$$

so that

$$\boldsymbol{v}^\top \phi\left(\boldsymbol{x}_i, \boldsymbol{\beta}\right) = \sum_{j=1}^{m} \beta_j \boldsymbol{w}_j^\top \boldsymbol{x}_i = \boldsymbol{\beta}^\top \boldsymbol{W} \boldsymbol{x}_i, \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ is a matrix containing m d-dimensional models and $\boldsymbol{\beta}$ represents the (local) model weighting vector. We then add an additional element: we constrain the vector $\boldsymbol{\beta}$ to lie within the positive p-norm unit ball $\Omega_p = \{\boldsymbol{\beta} \in \mathbb{R}^m : \|\boldsymbol{\beta}\|_p \leq 1, \beta_i \geq 0 \ \forall i = 1, \dots, m\}$. Varying p from 1 to ∞, allows to move from the case where only one model is contributing to the prediction of each sample, to the case where all the models tend to contribute in the same way. This generalizes the approach in Felzenszwalb et al. (2010), where only the most-confident model contributes to the prediction. Note that the non-negativity constraints on the elements of $\boldsymbol{\beta}$ are needed to avoid that the local weights invert the predictions of the linear models.

Finally, we extend (5) to the multiclass scenario, where we now have C classes and models $\boldsymbol{W}^y,\ y = 1, \ldots, C$. Hence our scoring function takes the form

$$s(\boldsymbol{x}_i, y) \triangleq \max_{\boldsymbol{\beta} \in \Omega_p} f_{\boldsymbol{W}^y}(\boldsymbol{x}_i, \boldsymbol{\beta}), \tag{6}$$

$$f_{\boldsymbol{W}^y}(\boldsymbol{x}_i, \boldsymbol{\beta}) \triangleq \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}_i. \tag{7}$$

A similar scoring function was proposed in Ladicky and Torr (2011) for binary problems, with $\boldsymbol{\beta}$ fixed in advance via manifold learning. Conversely, here we have introduced a specific constrained local maximization over $\boldsymbol{\beta}$, which has become a latent variable.

Note that $s(\boldsymbol{x}_i, y)$ is a convex function of $\boldsymbol{W}^y$. This comes from the facts that: 1) for every $\boldsymbol{\beta}$, $\boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}$ is a linear function of $\boldsymbol{W}^y$, so that $\sup_{\boldsymbol{\beta} \in \Omega_p} \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}$ is a convex function of $\boldsymbol{W}^y$ (Boyd and Vandenberghe, 2004); 2) since $\boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}$ is a continuous function of $\boldsymbol{\beta}$, and Ω_p is non-empty and closed: $\sup_{\boldsymbol{\beta} \in \Omega_p} \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x} = \max_{\boldsymbol{\beta} \in \Omega_p} \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}$.

Given the above scoring function (6), the ML3 objective function is defined as:

$$L_\lambda(\boldsymbol{W}^1, \ldots, \boldsymbol{W}^C, \boldsymbol{X}, \boldsymbol{y}) = \frac{\lambda}{2} \sum_{y=1}^{C} \|\boldsymbol{W}^y\|_F^2 + \sum_{i=1}^{n} \ell\left(s(\cdot), \boldsymbol{x}_i, y_i\right), \tag{8}$$

where

$$\ell\left(s(\cdot), \boldsymbol{x}_i, y_i\right) \triangleq \left|1 - \left(s(\boldsymbol{x}_i, y_i) - s(\boldsymbol{x}_i, \check{y}_i)\right)\right|_+,$$

$\check{y}_i \triangleq \arg\max_{y \neq y_i} s(\boldsymbol{x}_i, y)$ and $\|\cdot\|_F$ is the Frobenius norm.

Multiclass semi-convexity. The proposed objective function (8) is not convex w.r.t. the matrices $\boldsymbol{W}^y$, because, as noted above, $-s(\boldsymbol{x}_i, y_i)$ is a concave function of $\boldsymbol{W}^{y_i}$, and not just a linear one. However, using a similar approach as in (Felzenszwalb et al., 2010; Wang and Mori, 2009), we note that

$$\left|1 - \max_{\boldsymbol{\beta} \in \Omega_p} f_{\boldsymbol{W}^{y_i}}(\boldsymbol{x}_i, \boldsymbol{\beta}) + s(\boldsymbol{x}_i, \check{y}_i)\right|_+ = \min_{\boldsymbol{\beta} \in \Omega_p} \left|1 - f_{\boldsymbol{W}^{y_i}}(\boldsymbol{x}_i, \boldsymbol{\beta}) + s(\boldsymbol{x}_i, \check{y}_i)\right|_+ .$$

Moreover for any fixed $\boldsymbol{b}_i$, the loss function $\left|1 - f_{\boldsymbol{W}^{y_i}}(\boldsymbol{x}_i, \boldsymbol{b}_i) + s(\boldsymbol{x}_i, \check{y}_i)\right|_+$ is convex. Using this fact we can equivalently rewrite the objective function as:

$$\min_{\substack{\boldsymbol{W}^y \ y=1,\ldots,C \\ \boldsymbol{b}_i \in \Omega_p \ i=1,\ldots,n}} O_\lambda(\boldsymbol{W}^1, \ldots, \boldsymbol{W}^C, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{X}, \boldsymbol{y}), \tag{9}$$

where

$$O_\lambda(\boldsymbol{W}^1, \ldots, \boldsymbol{W}^C, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{X}, \boldsymbol{y}) \triangleq \frac{\lambda}{2} \sum_{y=1}^{C} \|\boldsymbol{W}^y\|^2 + \sum_{i=1}^{n} \ell\left(s(\cdot), f_{\boldsymbol{W}^{y_i}}, \boldsymbol{b}_i, \boldsymbol{x}_i, y_i\right), \tag{10}$$

$$\ell\left(s(\cdot), f_{\boldsymbol{W}^{y_i}}, \boldsymbol{b}_i, \boldsymbol{x}_i, y_i\right) \triangleq |1 - f_{\boldsymbol{W}^{y_i}}(\boldsymbol{x}_i, \boldsymbol{b}_i) + s(\boldsymbol{x}_i, \check{y}_i)|_+ . \tag{11}$$

This function is not jointly convex in $\{\boldsymbol{W}^1, \ldots, \boldsymbol{W}^C\}$ and $\boldsymbol{b}$, but it is convex in both of them, when considered separately. Using the terminology introduced in Felzenszwalb et al. (2010), we call this property *multiclass semi-convexity*.

3.1. Optimization

The semi-convexity of problem (9) suggests the following optimization strategy:

1. minimize (10) w.r.t. $\boldsymbol{b}_i$ $i = 1, \dots, n$, keeping the matrices $\boldsymbol{W}^y$ fixed
2. minimize (10) w.r.t. $\boldsymbol{W}^y$ $y = 1, \dots, C$, keeping the vectors $\boldsymbol{b}_i$ fixed

Each and every of the two steps reduces the objective function, so that the procedure is guaranteed to converge to a local optimum. As it will be explained afterwards, we optimize step 1 exactly, using a closed-form solution. On the other hand, step 2 is optimized by mean of one epoch of stochastic gradient descent (as in Felzenszwalb et al. (2010)), which produces an efficient, but noisy solution. Although one epoch of stochastic gradient descent is not guaranteed to minimize the objective function (which is needed for convergence), in practice we observed that it is usually good enough and its efficiency is especially compelling for large-scale problems.

Computing the optimal $\boldsymbol{\beta}$. The first problem to address when solving the ML3 learning task is how to find the optimal local combination of models for a given sample $\boldsymbol{x}_i$ and a given class y:

$$\begin{aligned} &\max_{\boldsymbol{\beta}} \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}_i \\ &\text{s.t. } \beta_i \geq 0 \ \forall i = 1, \dots, m \\ &\|\boldsymbol{\beta}\|_p \leq 1 \ . \end{aligned} \tag{12}$$

This problem resembles a linear program, with additional p-norm ball constraints and it needs to be solved for each training / testing sample and each class. Its solution can be computed in closed form and, specifically, if we call $\boldsymbol{c} = \boldsymbol{W}^y \boldsymbol{x}_i$ and $c_j^+ = |c_j|_+$, it is possible to prove[1] that, for any $1 < p < \infty$:

$$\boldsymbol{\beta}^* = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_p}, \tag{13}$$

where $v_j = c_j^+ \left(\frac{c_j^+}{\|\boldsymbol{c}^+\|_q} \right)^{q-2}$ and $q = p/(p-1)$. For the case $p = 1$ the solution is not unique. A feasible solution is given by

$$\boldsymbol{\beta}^* = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}, \tag{14}$$

where the only 1 is in the position m, corresponding to any c_m^+ s.t. $c_m^+ \geq c_i^+$, $\forall i$. If $c_m^+ = 0$, then $\boldsymbol{\beta}^* = \boldsymbol{0}$. This can be seen by noting that, for every $\boldsymbol{\beta}$ satisfying the constraints and $c_m \geq c_i \, \forall i$, we have:

$$\sum_i \beta_i c_i \leq c_m \sum_i \beta_i \leq c_m = \sum_i \underbrace{\delta_{i,m}}_{\beta_i^*} c_i \ . \tag{15}$$

Moreover, if $c_m \leq 0$, then $\sum_i \delta_{i,m} c_i \leq 0 = \sum_i \underbrace{0}_{\beta_i^*} c_i$.

1. A proof is provided in the appendix.

Stochastic optimization w.r.t $\boldsymbol{W}$. As anticipated above, to optimize (10) w.r.t. the matrices $\boldsymbol{W}^y$ we use a stochastic gradient descent algorithm similar to Pegasos (Shalev-Shwartz et al., 2007). Hence we need to calculate a stochastic approximation of the sub-gradient of (10) w.r.t. $\boldsymbol{W}^y$ using the sample $\boldsymbol{x}_i$, that we denote by $\tilde{\nabla}_{i,\boldsymbol{W}^y}$. Define

$$\boldsymbol{\beta}^*_{i,y} = \underset{\boldsymbol{\beta}\in\Omega_p}{\arg\max}\ \boldsymbol{\beta}^\top \boldsymbol{W}^y \boldsymbol{x}_i, \tag{16}$$

the optimal weight for a sample i and the class y; and $\mathbf{1}(r)$, the indicator function of the predicate r. Using the fact that Ω_p is closed, $s(\boldsymbol{x}_i, \check{y}_i)$ is convex w.r.t. $\boldsymbol{W}^{\check{y}_i}$, and applying Danskin's theorem (Bertsekas, 1999), one can see that

$$\tilde{\nabla}_{i,\boldsymbol{W}^y} = \lambda \boldsymbol{W}^y + \left(\left(\mathbf{1}(y=\check{y}_i)\boldsymbol{\beta}^*_{i,\check{y}_i} - \mathbf{1}(y=y_i)\boldsymbol{b}_i\right)\boldsymbol{x}_i^\top\right)\mathbf{1}\left(\ell\left(s(\cdot), f_{\boldsymbol{W}^{y_i}}, \boldsymbol{b}_i, \boldsymbol{x}_i, y_i\right) > 0\right). \tag{17}$$

We also use some known strategies to accelerate the convergence of the optimization algorithm. First, by lower and upper bounding the optimal value of objective function it is possible to derive a bound on the norm of the optimal classifier and use it to normalize the solution during training (Shalev-Shwartz et al., 2007). Specifically, call O^* the optimal value of (10), and $\rho = \sqrt{\sum_y ||W^{*y}||_F^2}$ the norm of the optimal ML3 classifier. Then $O^* \geq \frac{\lambda}{2}\rho^2$; moreover $O^* \leq n$ (the value of the objective function evaluated in $\boldsymbol{W}^y = 0, \forall y$), so that in total we have: $\rho \leq \sqrt{\frac{2n}{\lambda}}$. The optimal classifier thus needs to lie in a ball with the specified radius, and a projection rule of the form: $\boldsymbol{W}^y \leftarrow \boldsymbol{W}^y \min\left(1, \sqrt{\frac{2\,n}{\lambda\sum_{y=1}^C \|\boldsymbol{W}^y\|_F^2}}\right)$, $\forall y = 1, \ldots, C$ would enforce this condition. Secondly, as underlined in Felzenszwalb et al. (2010), a careful initialization of the matrices $\boldsymbol{W}^y$ might be necessary to avoid selecting unreasonable values for $\boldsymbol{\beta}^*_{i,y_i}$ in the first iteration of the algorithm. Bordes et al. (2009) also proposed to use an additional s_0 constant in the learning rate, to prevent the first updates from producing matrices $\boldsymbol{W}^y$ with an implausibly large norm. And finally, in the last epoch of the optimization we take the average of all the generated solutions and use it as the final solution, as suggested by the theory in Rakhlin et al. (2012).

The complete training algorithm is summarized in Algorithm 1 and its complexity/epoch is $O(ndmC)$.

Prediction. During the training phase, the latent variables $\boldsymbol{b}_i$ are treated as a model parameters to be learned. The $\boldsymbol{b}_i$ drive the learning procedure, by changing the value of the loss function and the computation of the sub-gradients (see (17)). However, when predicting the score $s(\boldsymbol{x}_i, y)$ for a test sample i and candidate class y, only $\boldsymbol{W}^y$ is used, since $\boldsymbol{\beta}^*_{i,y}$ have to be computed according to (13), or (14). Therefore, the only real parameters of the model are the matrices $\boldsymbol{W}^1, \ldots, \boldsymbol{W}^C$. Moreover, since during the testing phase there are no sub-gradients to be updated, the explicit computation of $\boldsymbol{\beta}^*_{i,y}$ is unnecessary. We can thus plug the solution for the optimal $\boldsymbol{\beta}^*_{i,y}$ (provided by (13) and (14)) into (7), to obtain:

$$s(\boldsymbol{x}_i, y) = \||\boldsymbol{W}^y\boldsymbol{x}_i|_+\|_q, \tag{18}$$

where, again, $q = p/(p-1)$. This provides us with a very efficient prediction rule, whose complexity/sample is $O(dmC)$, and which is likely faster than computing manifold coefficients and subsequently use them to locally weight the predictions.

Algorithm 1 Alternating optimization for ML3

```
Input: $s_0$, $W_0^1, \dots, W_0^C$, $T$, $\lambda$, $\boldsymbol{X}$, $\boldsymbol{y}$
Output: $\boldsymbol{W}$
 1: $s \leftarrow 0$
 2: $\boldsymbol{W}^y \leftarrow \boldsymbol{W}_0^y,\ \forall y = 1, \dots, C$
 3: $\bar{\boldsymbol{W}}^y \leftarrow \boldsymbol{0},\ \forall y = 1, \dots, C$
 4: for $t = 1, \dots, T$ do
 5:   $\boldsymbol{b}_i \leftarrow \arg\max_{\boldsymbol{\beta} \in \Omega_p} f_{\boldsymbol{W}^{y_i}}(\boldsymbol{x}_i, \boldsymbol{\beta}),\ \forall i = 1 \dots, n$
 6:   for $i = 1 \dots, n$ do
 7:     $\eta_i \leftarrow \frac{1}{\lambda(s+s_0)}$
 8:     $\boldsymbol{W}^y \leftarrow \boldsymbol{W}^y - \eta_i \tilde{\nabla}_{i,\boldsymbol{W}^y},\ \forall y = 1, \dots, C$
 9:     $\boldsymbol{W}^y \leftarrow \boldsymbol{W}^y \min\left(1, \sqrt{\frac{2\,n}{\lambda \sum_{y=1}^{C} \|\boldsymbol{W}^y\|_F^2}}\right),\ \forall y = 1, \dots, C$
10:     $s \leftarrow s + 1$
11:     if t==T then
12:       $\bar{\boldsymbol{W}}^y \leftarrow \frac{(i-1)\bar{\boldsymbol{W}}^y + \boldsymbol{W}^y}{i},\ \forall y = 1, \dots, C$
13:     end if
14:   end for
15: end for
16: $\boldsymbol{W}^y \leftarrow \bar{\boldsymbol{W}}^y,\ \forall y = 1, \dots, C$
```

Comparison with other approaches. Differently from the manifold embedding approaches (Yu et al., 2009; Ladicky and Torr, 2011; Zhang et al., 2011), ML3 does not approximate input samples and functions using a given set of anchor points, or planes. On the contrary, it makes use of a latent SVM approach to directly learn a non-linear decision function in the original input space. It is thus intrinsically different from the literature based on manifold learning. Indeed, by adopting this latent approach, the sample-to-model coefficient computation is parameter-free (e.g. no anchor points, anchor planes, or gating functions need to be part of the model) and it is explicitly performed only during training, to guide the algorithm and force the sub-models to specialize to different regions of the input space. During testing, no coefficients at all need to be computed.
As opposed to classical latent SVM (Wang and Mori, 2009; Felzenszwalb et al., 2010), ML3 is not limited to use only the single most confident sub-model for a given sample and class. On the contrary, the score maximization is treated as a class-dependent local convex optimization problem, with sparsity-inducing p-norm constraints, resulting in soft model combinations and smooth decision boundaries. Moreover, the optimal solution of this sample and class specific optimization is efficiently computed using a closed-form expression. This also results in a prediction rule that simply consists, for a given sample and class, in computing the q-norm of the positive part of the m-dimensional vector of predictions.

4. Hyper-parameters setting

As anticipated in Section 3.1, a proper initialization of $\boldsymbol{W}_0^1, \dots, \boldsymbol{W}_0^C$ avoids selecting unreasonable values for the latent variables during the very first iteration. To this end, we

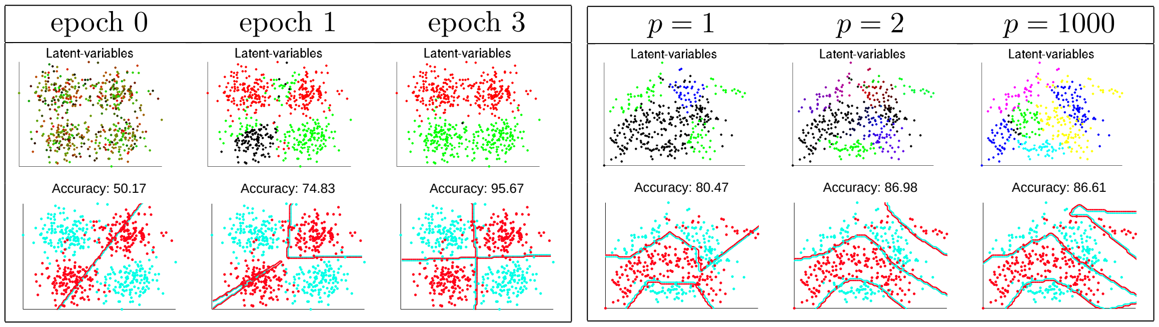

Figure 1: Left: training sequence on a synthetic XOR dataset, using two local models and $p = 1$. Right: effect of varying the parameter p in the set $\{1, 2, 1000\}$, using six local models on the Banana dataset. In the first row we color encode the sample-to-model assignment, with the RGB values set according to the first three components of $\boldsymbol{\beta}^*_{i,y_i}$. In the second row we plot the resulting classification boundary, with the ground-truth label color encoded in red and cyan.

propose the following procedure: 1) randomly initialize $\boldsymbol{\beta} \in \Omega_p$ for all training samples; 2) use the selected $\boldsymbol{\beta}$ to initialize $\boldsymbol{W}_0^1, \ldots, \boldsymbol{W}_0^C$ with one epoch of stochastic gradient descent; 3) fix $s_0 = n$. Although still random, this procedure forces the algorithm to start from a relatively stable point, so that in the subsequent updates the vectors $\boldsymbol{\beta}_i$ become meaningful. A visualization of a short learning sequence obtained with our random initialization is shown in Figure 1 left, where the leftmost plot shows the initial random $\boldsymbol{\beta}_{i,y_i}$ and the resulting $\boldsymbol{W}_0^1, \boldsymbol{W}_0^2$. As it can be seen, as the ML3 training progresses, the local models tend to specialize to separate parts of the space and the sample-to-model assignments cluster accordingly. At the third epoch, ML3 has learned two local models, each one covering a well defined region of the input space. With this local specialization of the linear models, the XOR problem becomes locally linearly separable and the global decision boundary almost perfect.

The ML3 algorithm has three hyper-parameters: the regularization trade-off λ, the number of local models m and the local model competitiveness parameter p. With respect to the manifold based local learning algorithms, the only additional parameter of ML3 is p. However, as we will see, setting it is not hard and in practice it can be kept constant for a large set of problems.
The role of this parameter can be understood by looking at Figure 1 right, where we plot the classification results on the synthetic dataset "Banana" (Frank and Asuncion, 2010), for different values of p and with $m = 6$. When $p = 1$, the optimal $\boldsymbol{\beta}^*_{i,y_i}$ assigns all the available weight to the single most confident positively scoring model (see (14)). This enforces a hard-clustering of the input space into well separated regions covered by a single local model. The boundary between clusters becomes sharp and the classification boundary non-smooth. Similarly, when $p \to \infty$ all the local models tend to be weighted with 1, except for those predicting negatively, which receive a sharp 0. This again results in hard boundaries between clusters, with sharply defined intersecting areas and a non-smooth decision boundary. Finally, when $p = 2$, each model is given a weight proportional to

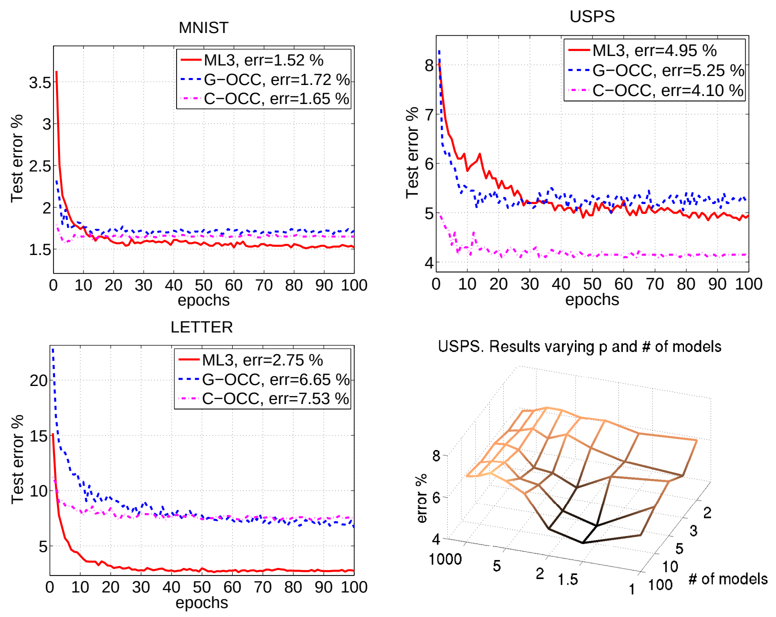

Figure 2: Test Error rate on MNIST, USPS and LETTER. The first three plots (from the top-left corner) are obtained with $m = 100$ and $p = 1.5$, for ML3. In the third plot (LETTER), the curves related to OCC are obtained with $m = 16$, due to the intrinsic limitations of the encoding. In the last figure (bottom-right), we plot the testing performances of ML3 , when varying both p and the number of models m. In all the experiments λ is set using 2-fold cross-validation.

its confidence, resulting in smooth transitions between local models, and smooth decision boundaries. In Figure 1 it is also possible to note that the boundary between regions covered by different local models is not forced to be linear (as in Gönen and Alpaydin (2008)), but is free to vary according to the discriminative clustering properties of the training set.

Although $p = 2$ is a reasonable candidate when the number of models is very low, whenever the number of models start to grow, a lower value of p will tend to retain only the top-scoring predictions. This in turn reduces the noise due to the "randomly" positive prediction of non-confident local models. In Figure 2 (bottom-right) we plot the testing performances of the ML3 algorithm when p varies between 1 and 1000, on a character recognition task (see section 5). As we can see, whenever m is large enough, setting p to 1.5 results in the best performances. Moreover, as it is possible to see, with values of p in [1, 2] the performance of ML3 does not degrade when the number of local models is increased.

5. Experiments

We assess our algorithm by running experiments on three character and digit recognition data collections, a challenging scene recognition dataset and on ten standard UCI machine learning datasets. This set of datasets largely overlaps with the ones used in (Gönen and

Table 1: Error rate (in percentage) and associated training time (in seconds) of different algorithms. When applicable, the number of local models / training epochs is also shown in parenthesis. The best regularization parameter was found using 2-fold cross-validation. Results taken from other papers are reported with the citation.

	MNIST		**USPS**		**LETTER**	
Linear SVM (Bordes et al., 2009)	12.00% (-/10)	1.5s	9.57% (-/10)	0.3s	41.77% (-/10)	0.2s
LCC (Yu et al., 2009)	1.90% (4096/-)		-		-	
Improved LCC (Yu and Zhang, 2010)	2.28% (4096/-)		4.38% (4096/-)		4.12% (4096/-)	
LLSVM (Ladicky and Torr, 2011)	1.85% (100/10)	81.7s	5.78% (100/10)	6.2s	5.32% (100/10)	4.2s
Liblinear (One vs all)	15.18%	9.4s	8.42%	1.3s	46%	13.6s
Liblinear (Multiclass)	7.3%	11.4s	7.92%	0.7s	24.38%	33.5s
G-OCC LLSVM	1.67% (90/30)	1366s	4.83% (80/30)	49.1s	9.23% (16/30)	4.6s
C-OCC LLSVM	1.72% (90/30)	2643s	**4.19**% (80/30)	95.5s	7.93% (16/30)	12.8s
ML3	**1.59**% (90/30)	978s	5.38% (80/30)	38.5s	**3.30**% (16/30)	32s

Alpaydin, 2008; Yu and Zhang, 2010; Ladicky and Torr, 2011; Yu et al., 2009; Zhang et al., 2011), sharing also the same scale in terms of number of samples and classes.

We compare our algorithm against state-of-the-art manifold learning techniques. Specifically, we compare against General OCC (G-OCC) and Class-specific OCC (C-OCC) (Zhang et al., 2011), using the implementations available on the website of the authors. As underlined in Zhang et al. (2011), the manifold learning step of OCC consists of learning a set of basis. This limits the maximum number of local models used by OCC LLSVM to be equal to the rank of the data matrix. We will specifically remark the cases in which this limitation results in a different number of local models with respect to ML3, or other baselines. We also report the results of standard linear learning algorithms, such as linear SVM (with a one vs all multiclass extension), multiclass linear SVM (Crammer and Singer, 2001) and, when possible, the results achieved by other authors using local learning algorithms.

In all our experiments (except for indoor scene recognition) the best regularization parameter for each algorithm is selected by performing 2-fold cross-validation on each training split and, for ML3, we fix $p = 1.5$. All the local learning algorithms are compared using the same number of local models, except where explicitly mentioned.

Hand written character recognition. MNIST (LeCun et al., 1998) is a dataset comprising 70,000 28×28 gray-scale images of hand-written digits, from 0 to 9. This dataset has one official training split with 60,000 samples and an associated test set with 10,000 samples. As a preprocessing step for this database, we normalize and center the images. The USPS (Hull, 1994) dataset consists of 7,291 training and 2,007 testing 16×16 gray-scale images of US postcodes, where each label corresponds to a digit between 0 and 9. Finally, LETTER (Frank and Asuncion, 2010) is composed of 16,000 training and 4,000 testing images of the 26 capital letters in the English alphabet; each image being compactly represented by a 16-dimensional vector.

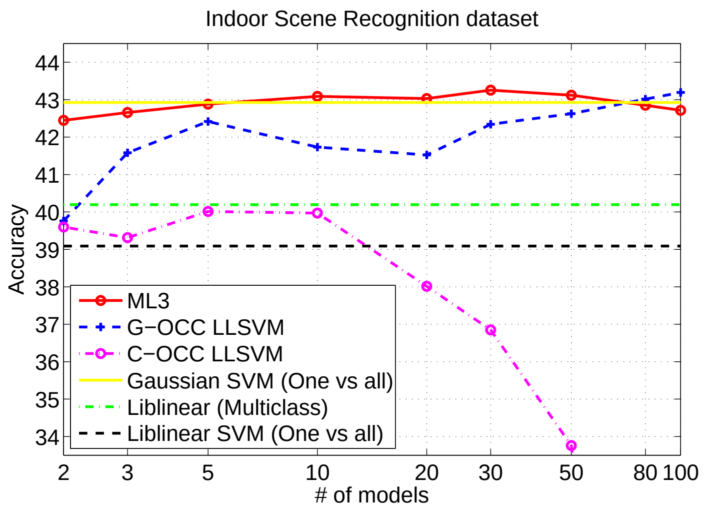

Figure 3: Average accuracy (five splits) on the ISR dataset. Best λ found on the test-set.

Following (Zhang et al., 2011) we set $m = 90$ for MNIST, $m = 80$ for USPS and $m = 16$ for LETTER. Moreover, for ML3 and LLSVM and in order to learn a bias for each local model, we concatenate a 1 to each instance vector. The experimental results obtained with this settings are summarized in Table 1. In Figure 2 we also plot the testing error as a function of the number of epochs, using $m = 100$. Note also that, since the LETTER dataset consists of 16-dimensional instances, the maximum number of orthogonal coordinates is 16. The OCC plots for the LETTER dataset are thus obtained with $m = 16$.

We see that, for MNIST and LETTER, ML3 obtains the state of the art for this class of algorithms (both with $m = 90/16$ and with $m = 100$). For USPS, the results are on par with the majority of the manifold learning algorithms, with Improved LCC and C-OCC obtaining better results.

The training times reported in Table 1 were obtained using a single thread on an Intel(R) Core(TM) i7-2600K, with 16GB of RAM. As for OCC and Liblinear, ML3 was developed using a mixed Maltab/C++ implementation, with the main algorithm being implemented in a .mex file[2]. As it can be seen, even though ML3 computes class-specific weights for each sample, its training times are on par or lower then those of G-OCC LLSVM, while being also comparable or lower than the ones measured using C-OCC. This could be due to the fact that in OCC the manifold is trained using SVD, whose complexity is $O\left(\min\{nd^2, dn^2\}\right)$. For the LETTER dataset, the training times of ML3 are on par with those obtained by Liblinear multiclass, but with an error-rate which is almost one order of magnitude lower.

Indoor Scene Recognition. As a second benchmark we use the Indoor Scene Recognition (ISR) dataset (Quattoni and Torralba, 2009), consisting of 15,620 images collected from the web and belonging to 67 different categories, with a minimum of 100 images per category. This is a difficult classification task, with a high degree of intra-class variability. In indoor environments, indeed, the location of meaningful regions and objects within a category changes drastically from sample to sample. Moreover, the close-up distance be-

2. The software will be freely available online, upon acceptance of the paper.

Table 2: Average accuracy (five splits) on the UCI benchmark datasets. For ML3 and LMKL we fix $p = 1.5$ and $m = 10$. For OCC $m = 10$, except for Banana, Liver, PIMA and WDBC, for which the maximal number of orthogonal coordinates is, respectively, 2, 6, 8 and 9. Best λ found by 2-fold cross-validation.

	ML3	**G-OCC**	**C-OCC**	**L-MKL**	**Liblinear**
Banana	87.71	63.55	65.72	**88.31**	54.08
German	**77.13**	76.47	73.41	72.81	74.97
Heart	**84.00**	74.67	76.44	81.33	83.78
Ionosphere	**89.06**	88.03	87.35	85.30	82.39
Liver	69.74	**70.26**	67.30	66.96	64.17
PIMA	**76.25**	74.92	72.42	74.84	73.83
Ringnorm	**91.68**	80.50	81.07	86.97	76.66
Sonar	**71.43**	70.57	68.86	67.14	66.86
Spambase	88.68	80.91	85.92	**90.61**	88.27
WDBC	**89.61**	88.58	88.76	86.44	86.61

tween the camera and the subject increases the severity of the view-point changes, making this dataset a perfect test-bed for local classification algorithms.

Following (Fornoni and Caputo, 2012), we extract SIFT descriptors on a regular grid, with 8 pixels spacing and 16 x 16 pixels patch size. We then compute a multiresolution histogram, downsampling the image and reducing the spacing and patch size to 6 and 12 x 12. For each resolution, a vocabulary with 1024 visual words is obtained by running k-means on a random subset of the training features. The local features are then encoded using approximated unconstrained LLC encoding (Wang et al., 2010) and pooled with max-pooling, using an horizontal partitioning scheme. This results in a relatively compact, but highly discriminative 4096-dimensional image descriptor, specifically designed to work with linear classifiers.

The standard benchmarking procedure for the ISR dataset consists of randomly selecting 100 images per category and split them into 80 images for training and 20 for testing. We repeated the experiment on five random training / testing splits. In Figure 3 we plot the accuracy of ML3 and OCC w.r.t. the number of models used. For each point of each curve, the best performing regularization coefficient was used. We also report the accuracy achieved by multiclass linear SVM, and by linear and Gaussian SVM with the One-VS-All multiclass extension.

On this dataset both ML3 and G-OCC are able to achieve and outperform a Gaussian kernel SVM, with the former generally outperforming the latter. It is also worth noting that already with two models ML3 performs largely better than a linear SVM, and that with as few as five models it is already able to match the performances of the SVM with the Gaussian kernel. On the other hand, the performances of C-OCC seem quite unsatisfactory. A reason for this could be found in the limited amount of samples (80) available to separately train each class-specific manifold, on the high-dimensional data (4096-D).

Benchmark datasets. Finally, we test our algorithm on ten two-class benchmark datasets from the UCI collection (Frank and Asuncion, 2010). For this benchmark we additionally compare against Localized Multiple Kernel Learning (Gönen and Alpaydin, 2008) using m linear kernels. For each of these datasets two thirds of the samples are used as a training set, while the remaining third is used as a test set. As explained before, each training set is divided in two folds that are used to select the regularization parameter. Each experiment was repeated five times on five different training / testing splits and the average accuracy is reported in Table 2. For these experiments, the number of local models was fixed to $m = 10$ and, as before, for ML3 and LLSVM we concatenated 1 to each instance vector. On the majority of the ten databases, once again ML3 achieves the best performances. Please note that the dimensionality of Banana, Liver, PIMA and WDBC is lower than 10, resulting in a reduced number of models (2, 6, 8 and 9, respectively) for the OCC encodings.

Using a Wilcoxon signed rank test on the accuracies reported in tables 1 and 2 (Demšar, 2006), ML3 results to perform significantly better than G-OCC and C-OCC, with $p = 0.0024$ and $p = 0.0012$, respectively. It is also worth noting that, although not being experimented in this paper, ML3 could be combined with manifold learning techniques such as LCC or OCC, simply by replacing the random initialization of the local weights, with a manifold learning step. This could further improve the convergence speed and the final performances of the algorithm, without affecting its testing efficiency.

6. Conclusions

We have proposed a new algorithm for multiclass classification based on Latent SVMs. It allows to have non-linear separation surfaces, through the use of local combinations of linear classifiers. Moreover, differently from previous works, our formulation has the advantage of not requiring a 2-stage training and testing (i.e. manifold and classifier). We also extend the standard Latent SVM formulation, adding a parameter that allows to modulate how the local models contribute to the prediction of a single sample. This allows to increase the smoothness of the decision function, while controlling the overfitting for large numbers of local models. During training, the sample-to-model soft assignments are computed using a closed form solution, while in testing the coefficients do not need to be explicitly computed. Experimental results show the advantage of the proposed method over similar algorithms.

In the future we plan to test ML3 on large scale classification datasets and to perform experiments using manifold learning, as an initialization step for the local weights of ML3. From a theoretical side, we plan to explore different routes to optimize the objective function of ML3. For example, the use of CCCP optimization, as in Yu and Joachims (2009), might give better theoretical guarantees on the convergence.

References

D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *J. Mach. Learn. Res.*, 10:1737–1754, December 2009.

L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

P. F. Felzenszwalb, D. A. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. of CVPR*, 2008.

P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9): 1627–1645, 2010.

M. Fornoni and B. Caputo. Indoor scene recognition using task and saliency-driven feature pooling. In *Proc. of BMVC*, 2012.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *Proc. of ICML*, pages 352–359. ACM, 2008.

J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5), 1994.

V. Kecman and J. P. Brooks. Locally linear support vector machines and other local models. In *Proc. of IJCNN*, pages 1–6, 2010.

L. Ladicky and P. H. S. Torr. Locally linear support vector machines. In *Proc. of ICML*, pages 985–992, 2011.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, November 1998.

G.-J. Qi, Q. Tian, and T. Huang. Locality-sensitive support vector machine by exploring local correlation and global regularization. In *Proc. of CVPR*, pages 841–848, Washington, DC, USA, 2011. IEEE Computer Society.

A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. of CVPR*, pages 413–420, 2009.

A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proc. of ICML*, 2012.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proc. of ICML*, pages 807–814. ACM, 2007.

V. Vapnik. Principles of risk minimization for learning theory. In *Proc. of NIPS*, pages 831–838, 1991.

J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. of CVPR*, 2010.

Y. Wang and G. Mori. Max-margin hidden conditional random fields for human action recognition. In *Proc. of CVPR*, pages 872–879, 2009.

T. Yang and V. Kecman. Adaptive local hyperplane classification. *Neurocomputing*, 71(1315): 3001–3004, 2008.

C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proc. of ICML*, pages 1169–1176, New York, NY, USA, 2009. ACM.

K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *Proc. of ICML*, pages 1215–1222, 2010.

K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Proc. of NIPS*, pages 2223–2231, 2009.

H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proc. of CVPR*, pages 2126–2136, 2006.

Z. Zhang, L. Ladicky, P. H. S. Torr, and A. Saffari. Learning anchor planes for classification. In *Proc. of NIPS*, pages 1611–1619, 2011.

Appendix

Closed-form solution for $\boldsymbol{\beta}^*$. Proof With $\boldsymbol{c} := \boldsymbol{W}^y \boldsymbol{x}_i$, the Lagrangian of the minimization problem corresponding to (12) is given by

$$L = -\boldsymbol{c}'\boldsymbol{\beta} + \alpha\left(\frac{1}{2}\|\boldsymbol{\beta}\|_p^2 - \frac{1}{2}\right) - \boldsymbol{\gamma}'\boldsymbol{\beta}, \tag{19}$$

with the corresponding KKT optimality conditions:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = 0 \rightarrow \nabla\left(\frac{1}{2}\|\boldsymbol{\beta}\|_p^2\right) = \frac{1}{\alpha}(\boldsymbol{c} + \boldsymbol{\gamma}) \tag{20}$$

$$\alpha\left(\frac{1}{2}\|\boldsymbol{\beta}\|_p^2 - \frac{1}{2}\right) = 0 \tag{21}$$

$$\beta_i \gamma_i = 0 \Rightarrow \begin{cases} \gamma_i \neq 0 \rightarrow \beta_i = 0 \\ \beta_i \neq 0 \rightarrow \gamma_i = 0 \end{cases} \tag{22}$$

$$\alpha \geq 0,\ \boldsymbol{\gamma} \geq 0,\ \boldsymbol{\beta} \geq 0 \tag{23}$$

$$\frac{1}{2}\|\boldsymbol{\beta}\|_p^2 - \frac{1}{2} \leq 0. \tag{24}$$

Making use of the fact that the inverse of $\frac{1}{2}\nabla\|\cdot\|_p^2$ is its convex conjugate $\frac{1}{2}\nabla\|\cdot\|_q^2$, from (20) we obtain

$$\beta_i = \nabla\left(\frac{1}{2}\left\|\frac{1}{\alpha}(\boldsymbol{c} + \boldsymbol{\gamma})\right\|_q^2\right)_i = \frac{1}{\alpha}(c_i + \gamma_i)\left(\frac{|c_i + \gamma_i|}{\|\boldsymbol{c} + \boldsymbol{\gamma}\|_q}\right)^{q-2}. \tag{25}$$

Equation (25) tells us that the optimal solution $\boldsymbol{\beta}^*$ is equal to a scaled version of the sum of the vector $\boldsymbol{c}$ plus a vector $\boldsymbol{\gamma}$. We know $\boldsymbol{c}$, but we don't know α and $\boldsymbol{\gamma}$. Since we must ensure $\beta_i \geq 0$, whenever $c_i < 0$ we must have $\gamma_i \geq -c_i > 0$. Moreover, using the condition given by (22), that $\gamma_i \neq 0 \rightarrow \beta_i = 0$, we obtain $\gamma_i = -c_i$, whenever c_i is negative. If we now set $\gamma_i = 0$ for $\{i : c_i \geq 0\}$ and $\alpha = \|\boldsymbol{v}\|_p$, we obtain: $\boldsymbol{\beta}^* = \frac{1}{\|\boldsymbol{v}\|_p}\boldsymbol{v}$, where $v_j = c_j^+\left(\frac{c_j^+}{\|\boldsymbol{c}^+\|_q}\right)^{q-2}$ and $c_j^+ = |c_j|_+$.

It is possible to verify that this solution satisfies all the KKT conditions and it is thus optimal. ■