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

Multiclass Latent Locally Linear Support Vector Machines

Marco Fornoni MARCO.FORNONI@IDIAP.CH
Idiap Research Institute, Martigny, Switzerland
Ecole Polytechnique Fédérale (EPFL), Lausanne, Switzerland

Barbara Caputo BARBARA.CAPUTO@IDIAP.CH
Idiap Research Institute, Martigny, Switzerland
Sapienza - Università di Roma, Italy

Francesco Orabona FRANCESCO.ORABONA@TTIC.EDU
Toyota Technological Institute at Chicago, USA

Editor: Cheng Soon Ong and Tu Bao Ho

Abstract

Kernelized Support Vector Machines (SVM) have gained the status of off-the-shelf classifiers, able to deliver state of the art performance on almost any problem. Still, their practical use is constrained by their computational and memory complexity, which grows super-linearly with the number of training samples. In order to retain the low training and testing complexity of linear classifiers and the flexibility of non linear ones, a growing, promising alternative is represented by methods that learn non-linear classifiers through local combinations of linear ones.

In this paper we propose a new multi class local classifier, based on a latent SVM formulation. The proposed classifier makes use of a set of linear models that are linearly combined using sample and class specific weights. Thanks to the latent formulation, the combination coefficients are modeled as latent variables. We allow soft combinations and we provide a closed-form solution for their estimation, resulting in an efficient prediction rule. This novel formulation allows to learn in a principled way the sample specific weights and the linear classifiers, in a unique optimization problem, using a CCCP optimization procedure. Extensive experiments on ten standard UCI machine learning datasets, one large binary dataset, three 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 locally linear decision functions. We propose a new multi class learning algorithm based on a latent SVM formulation. For each sample and class, during training as well as during testing, our algorithm selects a different weighted combination of linear models (as in Yu et al. (2009) and Ladicky and Torr (2011)). The sample and class specific weights are treated as latent variables of the scoring function (Felzenszwalb et al., 2010) and are obtained by locally maximizing the confidence of the class model on the sample. 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 an optimal combination of models, competing with the others in the training process. Moreover, compared to standard latent SVM implementations, our formulation allows to use soft combinations of models, where the sparsity of the combinations, and thus the smoothness of the solution, is tuned using a p-norm constraint. The solution of the p-norm constrained score maximization problem is shown to be efficiently computable in closed-form, and using this analytic solution we also obtain a prediction rule in which the local weights do not need to be explicitly computed. We call our method Multiclass 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 character recognition databases (MNIST (LeCun et al., 1998), USPS (Hull, 1994) and LETTER (Frank and Asuncion, 2010)), a large dataset with more than 500,000 samples (Joachims, 2006) 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, and its optimization procedure. 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 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 locally 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 on 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 sample will significantly contribute to the prediction. Unfortunately, the testing time of such methods grows linearly with the number of support vectors, while their training complexity grows cubically with the training set size.

When using linear kernels, non-linear functions can be directly modeled as local linear combinations of linear ones, with a gating function assigning a weight to each model, for each sample (Gönen and Alpaydin, 2008). Still, the optimization problem is very complex and finding a solution is very slow. 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 was shown to outperform 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 the methods in this last group use sample-to-model assignments that are learned with a separate unsupervised learning phase, unaware of the supervised classification task. Hence, the local capacity of the classifiers is adjusted taking into account only the distribution of the instances, without considering the labels.

Differently from the manifold approaches, 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. In this way the sample-to-model coefficients take also into account the supervised information given by the labels of the problem, rather than using only the distribution of the sample points. Still, the objective function is simpler than some previous ones, e.g. (Gönen and Alpaydin, 2008), resulting in a much shorter training time. Moreover, as opposed to the latent SVM implementations in Wang and Mori (2009) and Felzenszwalb et al. (2010), ML3 uses soft model combinations that result in smooth decision boundaries. The sample specific weights are efficiently computed using a closed-form expression, while during testing no coefficients at all need to be explicitly computed. Hence, as we will show in Section 5, ML3 achieves state-of-art performances, with efficient training and testing procedures.

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 is trained using a CCCP procedure (Smola et al., 2005), where each of the CCCP subproblems is solved using stochastic gradient descent (s.g.d.), with an adapted version of Shalev-Shwartz et al. (2007).

Latent SVM. Latent SVMs were initially motivated and introduced in the field of computer vision to solve object detection (Felzenszwalb et al., 2010) and action recognition (Wang and Mori, 2009) tasks. For a given instance $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and a given model $\boldsymbol{v} \in \mathbb{R}^b$, a latent SVM uses a scoring function of the form:

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

where $\phi(\boldsymbol{x}_i, \boldsymbol{\beta}) \in \mathbb{R}^b$ defines a feature mapping that depends on a latent variable $\boldsymbol{\beta}$, while $\mathcal{B}(x_i)$ defines the set of feasible solutions for the sample $\boldsymbol{x}_i$. The main idea of latent SVM is to find, for each instance $\boldsymbol{x}_i$, a feature mapping ϕ maximizing the confidence of the model $\boldsymbol{v}$ on the sample. For example, in object detection the model $\boldsymbol{v}$ would likely include a set of sub-models corresponding to different poses (frontal, lateral, etc.), with the scoring function (1) being responsible for selecting the best fitting one. In Latent SVMs the decision space $\mathcal{Y}$ is usually defined as $\{-1, 1\}$ and the prediction rule as $\hat{y}_i \triangleq \operatorname{sign}(s_{\boldsymbol{v}}(\boldsymbol{x}_i))$. Given a set of n training examples $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, the objective function takes the form:

$$\min_{\boldsymbol{v}, \boldsymbol{\xi}} \ \frac{\lambda}{2}\|\boldsymbol{v}\|^2 + \sum_{i=1}^{n} \xi_i \tag{2a}$$

$$\text{s.t. } 1 - y_i s_{\boldsymbol{v}}(\boldsymbol{x}_i) \leq \xi_i \quad \forall i = 1, \ldots, n \tag{2b}$$

$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{2c}$$

Due to the maximization, $s_{\boldsymbol{v}}$ becomes strictly convex w.r.t. $\boldsymbol{v}$ so that the constraints become strictly concave for all i such that $y_i = 1$. A general way to optimize such problems is to use the Concave Convex Procedure (CCCP) (Smola et al., 2005; Yu and Joachims, 2009).

Algorithm 1 Constrained Concave Convex Procedure

1: Initialize $t = 0$ and $\boldsymbol{v}^t$ with a random value
2: **repeat**
3: $\quad \boldsymbol{v}^{t+1} = \arg\min_{\boldsymbol{v}} f_0(\boldsymbol{v}) - t_{\{g_0, \boldsymbol{v}^t\}}(\boldsymbol{v})$
4: $\qquad$ s.t. $f_i(\boldsymbol{v}) - t_{\{g_i, \boldsymbol{v}^t\}}(\boldsymbol{v}) \leq c_i \quad \forall i = 1, \ldots, n$
5: **until** a stopping criterion is satisfied

The CCCP procedure. Suppose we need to optimize a problem of the form

$$\min f_0(\boldsymbol{v}) - g_0(\boldsymbol{v}) \tag{3a}$$
$$\text{s.t. } f_i(\boldsymbol{v}) - g_i(\boldsymbol{v}) \leq c_i \quad \forall i = 1, \ldots, n, \tag{3b}$$

where f_i and g_i are real valued convex and differentiable functions on a vector space $\mathcal{X}$, for all $i = 0, \ldots, n$. A way to find a local minimizer of (3) is to apply the CCCP optimization procedure reported in Algorithm 1, where $t_{\{g_i, \boldsymbol{v}^t\}}$ is the first-order Taylor approximation of g_i around $\boldsymbol{v}^t$. The main idea of the algorithm is that, since all the g_i are convex, $g_i(\boldsymbol{v}) \geq t_{\{g_i, \boldsymbol{v}^t\}}(\boldsymbol{v})$ and consequently $f_i(\boldsymbol{v}) - t_{\{g_i, \boldsymbol{v}^t\}}(\boldsymbol{v}) \geq f_i(\boldsymbol{v}) - g_i(\boldsymbol{v})$. Any feasible point in Algorithm 1 is thus also a feasible point of problem (3) and the algorithm can be shown to converge to a local minima of (3) (Theorem 1 of Smola et al. (2005)).

Multiclass Latent Locally Linear SVM. Based on the Latent SVM formulation, we now derive our algorithm in a multi class setting. With $\mathcal{Y} \triangleq \{1, \ldots, c\}$, we define the mappings ψ, ϕ and the block matrix $\boldsymbol{W}$ as

$$\psi(\boldsymbol{x}_i, \boldsymbol{\beta}) \triangleq \boldsymbol{\beta}\boldsymbol{x}_i^\top \in \mathbb{R}^{m \times d}, \tag{4}$$

$$\phi(\boldsymbol{x}_i, y, \boldsymbol{\beta}) \triangleq \begin{bmatrix} \mathbf{0} & \ldots & \mathbf{0} & \underbrace{\psi(\boldsymbol{x}_i, \boldsymbol{\beta})^\top}_{y} & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix}^\top \in \mathbb{R}^{mc \times d}, \tag{5}$$

$$\boldsymbol{W} \triangleq \begin{bmatrix} \boldsymbol{W}_1^\top & \boldsymbol{W}_2^\top & \ldots & \boldsymbol{W}_c^\top \end{bmatrix}^\top \in \mathbb{R}^{mc \times d}, \tag{6}$$

where each $\boldsymbol{W}_y \in \mathbb{R}^{m \times d}$ is a matrix containing m d-dimensional models and $\boldsymbol{\beta}$ is our m-dimensional latent variable. Using this notation, the prediction of the algorithm is given by $\hat{y}_i \triangleq \arg\max_{y \in \mathcal{Y}} s_{\boldsymbol{W}}(\boldsymbol{x}_i, y)$, where

$$s_{\boldsymbol{W}}(\boldsymbol{x}_i, y) \triangleq \max_{\boldsymbol{\beta} \in \Omega_p} f_{\boldsymbol{W}}(\boldsymbol{x}_i, y, \boldsymbol{\beta}), \tag{7}$$

$$f_{\boldsymbol{W}}(\boldsymbol{x}_i, y, \boldsymbol{\beta}) \triangleq \boldsymbol{W} \cdot \phi(\boldsymbol{x}_i, y, \boldsymbol{\beta}) = \operatorname{Tr}\left(\boldsymbol{W}^\top \phi(\boldsymbol{x}_i, y, \boldsymbol{\beta})\right) = \operatorname{Tr}\left(\boldsymbol{W}_y^\top \boldsymbol{\beta}\boldsymbol{x}_i^\top\right) = \boldsymbol{\beta}^\top \boldsymbol{W}_y \boldsymbol{x}_i, \tag{8}$$

Ω_p is defined as the positive p-norm unit ball, $\Omega_p \triangleq \{\boldsymbol{\beta} \in \mathbb{R}^m : \|\boldsymbol{\beta}\|_p \leq 1, \beta_i \geq 0 \ \forall i = 1, \ldots, m\}$ and $\boldsymbol{\beta}$ can be understood as a model-weighting vector. A scoring function similar to (8) was also used in Ladicky and Torr (2011) for binary problems, with $\boldsymbol{\beta}$ fixed in advance via manifold learning. The proposed scoring function is termed *locally linear* in the sense that for each sample and class it makes use of an ad hoc linear combination of linear models (as can be seen from (7) and (8)), where the sample and class specific model weighting vector $\boldsymbol{\beta}$ is directly chosen to maximize the score. An important consequence of this choice

is that the model-weighting vectors $\boldsymbol{\beta}$ are local in a non-Euclidean sense. In other words, a model can be used in multiple areas of the input space, that are far apart in Euclidean distance, but similar from the point of view of the classification function. This will be empirically shown in Section 4.

Note that the non-negativity constraints on β_i are needed to avoid that the local weights invert the predictions of the linear models. Moreover, as it will be shown later, varying p in $[1, \infty)$ 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 extends the latent SVM implementations in Wang and Mori (2009); Felzenszwalb et al. (2010), that were limited to use only the single most confident sub-model for a given sample.

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

The ML3 objective function is finally defined as:

$$\min_{\boldsymbol{W}, \boldsymbol{\xi}} \ \frac{\lambda}{2} \|\boldsymbol{W}\|_F^2 + \sum_{i=1}^{n} \xi_i \tag{9a}$$

$$\text{s.t. } 1 - (s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i) - n_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)) \leq \xi_i, \ i = 1, \dots, n \tag{9b}$$

$$\xi_i \geq 0, \ i = 1, \dots, n \tag{9c}$$

where $\| \cdot \|_F$ is the Frobenius norm and $n_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i) \triangleq \max_{y \in Y \setminus \{y_i\}, \boldsymbol{\beta} \in \Omega_p} f_{\boldsymbol{W}}(\boldsymbol{x}_i, y, \boldsymbol{\beta})$. The objective function (9) is not convex w.r.t. $(\boldsymbol{W}, \boldsymbol{\xi})$ because, as noted above, $-s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)$ is a concave function of $\boldsymbol{W}_{y_i}$, and not just a linear one. However, each of the constraints in (9b) can be decomposed into the difference of two convex functions: $f_i(\boldsymbol{W}, y_i) = 1 - \xi_i + n_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)$ and $g_i(\boldsymbol{W}, y_i) = s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)$. To solve our problem we can thus make use of the CCCP optimization algorithm discussed before.

CCCP Iteration for ML3. The first step to solve the ML3 learning problem using CCCP is to compute the first order Taylor expansion of $s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)$ (as a function of $\boldsymbol{W}$) around an arbitrary point $\boldsymbol{C} \in \mathbb{R}^{mc \times d}$

$$t_{\{s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i), \boldsymbol{C}\}} = s_{\boldsymbol{C}}(\boldsymbol{x}_i, y_i) + \nabla s_{\boldsymbol{C}}(\boldsymbol{x}_i, y_i) \cdot (\boldsymbol{W} - \boldsymbol{C}) \ . \tag{10}$$

Using the fact that Ω_p is closed, $s_{\boldsymbol{W}}(\boldsymbol{x}_i, y)$ is convex w.r.t. $\boldsymbol{W}$ and applying Danskin's theorem (Bertsekas, 1999), one can see that $\nabla s_{\boldsymbol{C}}(\boldsymbol{x}_i, y)^\top = \begin{bmatrix} \boldsymbol{0} & \cdots & \boldsymbol{x}_i \boldsymbol{\beta}_{\boldsymbol{C}, i, y}^{*\top} & \cdots & \boldsymbol{0} \end{bmatrix}$, where

$$\boldsymbol{\beta}_{\boldsymbol{W}, i, y}^{*} \triangleq \underset{\beta_i \geq 0, \|\boldsymbol{\beta}\|_p \leq 1}{\arg\max} \ \boldsymbol{\beta}^\top \boldsymbol{W}_y \boldsymbol{x}_i \tag{11}$$

is the optimal local combination of submodels for a given model $\boldsymbol{W}$, a given sample $\boldsymbol{x}_i$ and a given class y. We can thus write

$$t_{\{s_{\boldsymbol{W}}(\boldsymbol{x}_i, y), \boldsymbol{C}\}} = s_{\boldsymbol{C}}(\boldsymbol{x}_i, y) + \operatorname{Tr}\left(\left(\boldsymbol{\beta}_{\boldsymbol{C}, i, y}^{*} \boldsymbol{x}_i^\top \right)^\top (\boldsymbol{W}_y - \boldsymbol{C}_y) \right) \tag{12a}$$

$$= s_{\boldsymbol{C}}(\boldsymbol{x}_i, y) + \boldsymbol{\beta}_{\boldsymbol{C}, i, y}^{*\top} \boldsymbol{W}_y \boldsymbol{x}_i - \boldsymbol{\beta}_{\boldsymbol{C}, i, y}^{*\top} \boldsymbol{C}_y \boldsymbol{x}_i = \boldsymbol{\beta}_{\boldsymbol{C}, i, y}^{*\top} \boldsymbol{W}_y \boldsymbol{x}_i \ . \tag{12b}$$

By replacing the $s_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i)$ with $t_{\{s_{\boldsymbol{W}}(\boldsymbol{x}_i,y_i),\boldsymbol{W}^t\}}$, we can now write the CCCP iteration for ML3 as

$$\boldsymbol{W}^{t+1} = \arg\min_{\boldsymbol{W}} \frac{\lambda}{2}\|\boldsymbol{W}\|_F^2 + \sum_{i=1}^{n} \ell\left(\boldsymbol{W}, \boldsymbol{W}^t, \boldsymbol{x}_i, y_i\right) \tag{13a}$$

$$\ell\left(\boldsymbol{W}, \boldsymbol{W}^t, \boldsymbol{x}_i, y_i\right) = \left|1 + n_{\boldsymbol{W}}(\boldsymbol{x}_i, y_i) - \boldsymbol{\beta}^{*\top}_{\boldsymbol{W}^t,i,y_i} \boldsymbol{W}_{y_i}\boldsymbol{x}_i\right|_+, \tag{13b}$$

where we also have switched to the unconstrained formulation of the optimization problem.

Computing the optimal $\boldsymbol{\beta}^*_{\boldsymbol{W},i,y}$. We now show how to find the optimal local combination of models for a given sample $\boldsymbol{x}_i$ and a given class y. The problem in equation (11) resembles a linear program, with additional p-norm ball constraints and its solution can be computed in closed form. Specifically, let $\boldsymbol{c} \triangleq \boldsymbol{W}_y\boldsymbol{x}_i$ and $\boldsymbol{c}^+$ be defined as the element wise maximum between $\boldsymbol{c}$ and $\boldsymbol{0}$, it is then possible to prove[1] that, for any $1 < p < \infty$

$$\boldsymbol{\beta}^*_{\boldsymbol{W},i,y} = \left[\left(\frac{c_1^+}{\|\boldsymbol{c}^+\|_q}\right)^{q-1} \quad \left(\frac{c_2^+}{\|\boldsymbol{c}^+\|_q}\right)^{q-1} \quad \cdots \quad \left(\frac{c_m^+}{\|\boldsymbol{c}^+\|_q}\right)^{q-1}\right]^\top, \tag{14}$$

where $q = p/(p-1)$. When $p = 1$, an optimal solution is simply given by

$$\boldsymbol{\beta}^*_{\boldsymbol{W},i,y} = \begin{bmatrix}0 & \cdots & 0 & 1 & 0 & \cdots & 0\end{bmatrix}^\top, \tag{15}$$

where the only 1 is in the position m, corresponding to any $c_m^+ > 0$ such that $c_m^+ \geq c_i^+$, $\forall i$. Moreover, if $c_m^+ = 0$, then $\boldsymbol{\beta}^* = \boldsymbol{0}$. Whenever the maximum c_m^+ is achieved by more than one element of the vector, any convex combination of those elements would provide the same maximal value of the objective function. In this case, amongst the possible solutions we again chose one with the form of (15).

Implementation. In order to efficiently optimize (13) we opt to make use of just one epoch of s.g.d., using an algorithm similar to Shalev-Shwartz et al. (2007) and some known strategies to accelerate convergence. Although one epoch of s.g.d. is not guaranteed to minimize the objective function (which is needed for convergence), we observed that in practice it is enough and its efficiency is especially compelling for large-scale problems.

Let $\tilde{\boldsymbol{W}}^{t+1}$ be the current estimate of $\boldsymbol{W}^{t+1}$ during the stochastic gradient descent, $\boldsymbol{1}(r)$ the indicator function of the predicate r, and $\check{y}_i \triangleq \arg\max_{y \in Y \setminus \{y_i\}} s_{\tilde{\boldsymbol{W}}^{t+1}}(\boldsymbol{x}_i, y)$. Applying again Danskin's theorem (Bertsekas, 1999), the stochastic approximation of the subgradient of (13) w.r.t. $\boldsymbol{W}_y$ using the sample $\boldsymbol{x}_i$ can be written as

$$\tilde{\nabla}_{i,\boldsymbol{W}_y} = \lambda\tilde{\boldsymbol{W}}_y^{t+1} + \boldsymbol{1}\left(\ell\left(\tilde{\boldsymbol{W}}^{t+1}, \boldsymbol{W}^t, \boldsymbol{x}_i, y_i\right) > 0\right)\left(\boldsymbol{1}(y = \check{y}_i)\boldsymbol{\beta}^*_{\tilde{\boldsymbol{W}}^{t+1},i,\check{y}_i} - \boldsymbol{1}(y = y_i)\boldsymbol{\beta}^*_{\boldsymbol{W}^t,i,y_i}\right)\boldsymbol{x}_i^\top.$$

As it can be seen, only the submodels of $\boldsymbol{W}_y$ for which the associated component of $\boldsymbol{\beta}^*_{\boldsymbol{W},i,y}$ is different from zero are updated using $\boldsymbol{x}_i$. Each submodel tends thus to be trained only with samples for which its confidence is higher than the other submodels for the same class.

A strategy to accelerate the convergence of s.g.d. is to bound the norm of the optimal classifier and use it to normalize the solution during training (Shalev-Shwartz et al., 2007).

1. A proof is provided in the Appendix.

Algorithm 2 Stochastic Gradient Descent for the $t+1$ CCCP iteration of ML3

Input: $\boldsymbol{X}$, $\boldsymbol{y}$, $\boldsymbol{W}^t$, λ, s_0, *lastIteration*
Output: $\boldsymbol{W}$, $\bar{\boldsymbol{W}}$

$\boldsymbol{W} \leftarrow \boldsymbol{W}^t$
$\bar{\boldsymbol{W}} \leftarrow \boldsymbol{0}$
for $s = 1 \dots, n$ **do**
  $\eta \leftarrow \frac{1}{\lambda(s+s_0)}$
  $\boldsymbol{W}_y \leftarrow \boldsymbol{W}_y - \eta \tilde{\nabla}_{s,\boldsymbol{W}_y},\ \forall y = 1, \dots, c$
  $\boldsymbol{W} \leftarrow \boldsymbol{W} \min\left\{1, \frac{\sqrt{2n}/\sqrt{\lambda}}{\|\boldsymbol{W}\|_F}\right\}$
  if *lastIteration* **then**
    $\bar{\boldsymbol{W}} \leftarrow \frac{(s-1)\bar{\boldsymbol{W}}+\boldsymbol{W}}{s}$
  end if
end for

Specifically, let O^* be value of the objective function in (13) obtained with the optimal classifier $\boldsymbol{W}^*$. Then $O^* \geq \frac{\lambda}{2}\|\boldsymbol{W}^*\|_F^2$; moreover $O^* \leq n$ (the value of the objective function evaluated in $\boldsymbol{W} = \boldsymbol{0}$), so that we have: $\|\boldsymbol{W}^*\|_F \leq \sqrt{\frac{2n}{\lambda}}$. The norm of the optimal classifier is thus bounded, and a projection rule of the form: $\boldsymbol{W} \leftarrow \boldsymbol{W} \min\left\{1, \sqrt{\frac{2\,n}{\lambda\,\|\boldsymbol{W}\|_F^2}}\right\}$ would enforce this condition. Secondly, Bordes et al. (2009) proposed to use an additional constant term, s_0, in the learning rate, to prevent the first updates from producing matrices $\boldsymbol{W}$ with an implausibly large norm. As a side effect, this also allows to use $\boldsymbol{W}^t$ to initialize the algorithm, when computing $\boldsymbol{W}^{t+1}$. Also, as underlined in Felzenszwalb et al. (2010), a careful initialization of $\boldsymbol{W}^0$ might be necessary to avoid selecting unreasonable values for $\boldsymbol{\beta}^*_{\boldsymbol{W}^0,i,y_i}$ in the first iteration. Finally, in the last CCCP iteration we take the average of all the generated solutions and use it as the final solution. The complete training algorithm for one CCCP iteration is summarized in Algorithm 2 and its complexity is $O(ndmc)$.

Prediction. When predicting the score $s_{\boldsymbol{W}}(\boldsymbol{x}_j, y)$ for a test sample j and candidate class y, $\boldsymbol{\beta}^*_{\boldsymbol{W},j,y}$ is computed according to (14), or (15). Differently from the manifold learning approaches, the only parameter of the model is thus the matrix $\boldsymbol{W}$. Moreover, for prediction purposes the explicit computation of $\boldsymbol{\beta}^*_{\boldsymbol{W},j,y}$ is unnecessary. We can indeed plug the solution for the optimal $\boldsymbol{\beta}^*_{\boldsymbol{W},j,y}$ (provided by (14) and (15)) into (8), to obtain:

$$s_{\boldsymbol{W}}(\boldsymbol{x}_j, y) = \||\boldsymbol{W}_y \boldsymbol{x}_j|_+\|_q, \tag{16}$$

where, again, $q = p/(p-1)$ and $|\boldsymbol{v}|_+$ is the element-wise maximum between $\boldsymbol{v}$ and $\boldsymbol{0}$. This provides us with a very efficient prediction rule, whose complexity/sample is $O(dmc)$.

4. Hyper-parameters setting

As anticipated in Section 3, a proper initialization of the algorithm avoids selecting unreasonable values for the latent variables during the very first CCCP iteration. To this end, we propose the following procedure: 1) randomly initialize $\boldsymbol{\beta}_{\boldsymbol{W}^0,i,y} \in \Omega_p$ for all training

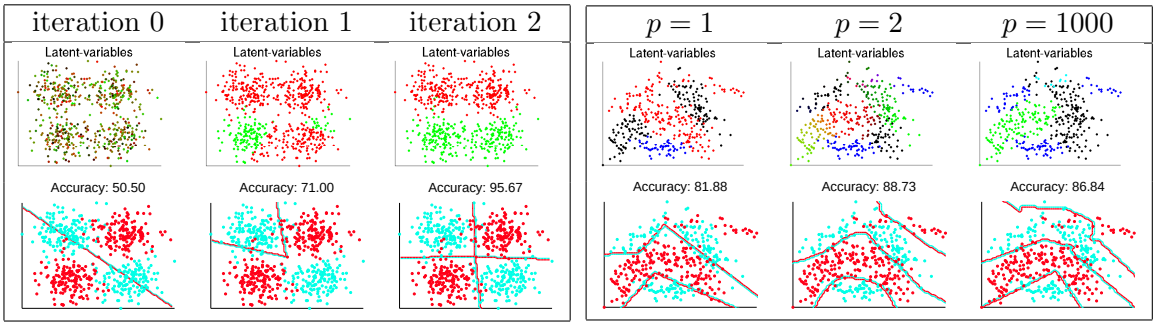

Figure 1: Left: training sequence on a synthetic XOR dataset, using two models and $p = 1$. Right: effect of varying the parameter p in the set $\{1, 2, 1000\}$, using five models on the Banana dataset. In the first row we color encode the sample-to-model assignment, with the RGB values set according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

samples and classes; 2) initialize $\boldsymbol{W}^0$ with one epoch of stochastic gradient descent (keeping all the latent variables fixed); 3) fix $s_0 = 2n$. Although still random, this procedure forces the CCCP procedure to start from a relatively good solution, increasing the chances to converge to a good local optima. To speed up convergence, at the end of each CCCP iteration we also increment s_0 by $2n$.

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 $\boldsymbol{\beta}_{\boldsymbol{W}^0,i,y_i}$ and the resulting $\boldsymbol{W}^0$. As it can be seen, as the ML3 training progresses, the models tend to specialize to separate parts of the space and the sample-to-model assignments cluster accordingly. At the second CCCP iteration, ML3 has learned two models, each one covering a well defined region of the input space. With this 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 models m and the model competitiveness parameter p. With respect to other approaches to local learning, p is the only additional parameter. However, as we will see, in practice it can be kept constant for a large set of problems, essentially removing it from the hyper-parameters. 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 $m = 5$. When $p = 1$, the optimal $\boldsymbol{\beta}^*_{\boldsymbol{W},i,y_i}$ assigns all the available weight to the single most confident positively scoring model (see (15)). This enforces a hard-clustering of the input space into well separated regions covered by a single model. The boundary between clusters is sharp and the classification boundary non-smooth. Similarly, when $p \to \infty$ all the weights tend to 1, except for those predicting negatively, which receive a sharp 0 (see (14)). 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 its confidence, resulting in smooth transitions between local models, and smooth decision boundaries. As anticipated in Section 3, in Figure 1 it is possible to note also that each model can cover

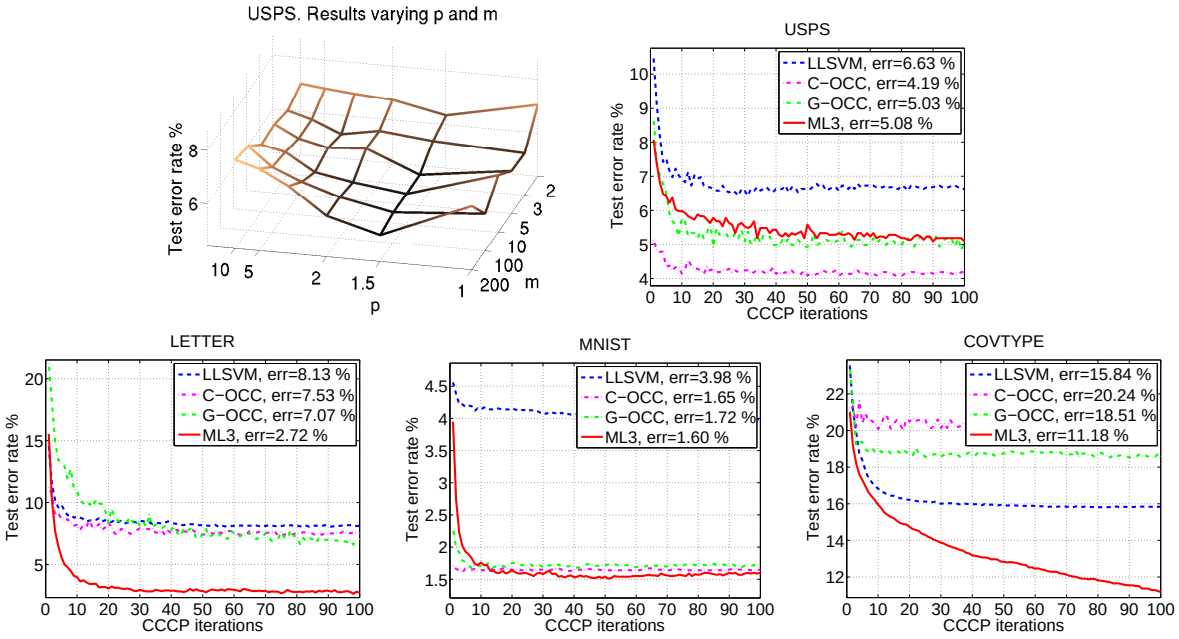

Figure 2: Top-left: performance on USPS, when varying p and the number of models m. Others: error rates on USPS, LETTER, MNIST and COVTYPE using $m = 100$.

regions that are far apart in space, while the boundary between different regions is free to vary according to the discriminative clustering properties of the data. The models are thus local in a classification-oriented fashion, rather than in an Euclidean neighborhood sense.

Although $p = 2$ is a reasonable candidate when the number of models is very low, whenever the number of models starts 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 models. In Figure 2 (top-left) we plot the testing performances of the ML3 algorithm when p varies between 1 and 10, 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 performance. 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 models is increased. For these reasons, in the following p will be set to 1.5 for all the experiments.

5. Experiments

We assess our algorithm by running experiments on three character recognition datasets, one large binary dataset, 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 Alpaydin, 2008; Yu and Zhang, 2010; Ladicky and Torr, 2011; Yu et al., 2009; Zhang et al., 2011), with a similar 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 Locally Linear SVM (LLSVM) (Ladicky and Torr, 2011), General OCC (G-OCC) and Class-specific OCC (C-OCC) (Zhang et al., 2011). For LLSVM we used

Table 1: Error rate and associated training and testing time (in seconds) of different algorithms. Results taken from other papers are reported with the citation.

	USPS			LETTER			MNIST			COVTYPE		
Gaussian-Libsvm	4.88%	2.55s	1.88s	**3.15%**	3.20s	2.81s	1.78%	327s	284s	**9.30%**	$> 10^5$s	1576s
Linear SVM (Bordes et al., 2009)	9.57%	0.30s	-	41.77%	0.20s	-	12.00%	1.50s	-	-	-	-
Liblinear	8.37%	0.89s	0.02s	45.85%	0.55s	0.01s	16.24%	13.0s	0.33s	23.13%	41.0s	0.04s
MC-Liblinear	7.67%	0.52s	0.02s	24.4%	35.5s	0.01s	7.37%	8.31s	0.33s	22.63%	313s	0.04s
LCC (Yu et al., 2009)	-	-	-	-	-	-	1.90%	-	-	-	-	-
Improved LCC (Yu and Zhang, 2010)	4.38%	-	-	4.12%	-	-	2.28%	-	-	-	-	-
LLSVM	6.78%	5.21s	0.11s	17.25%	1.99s	0.02s	3.81%	120s	1.65s	17.36%	9.34s	0.32s
G-OCC	5.03%	48.1s	1.00s	7.90%	3.99s	0.08s	1.65%	1365s	16.9s	18.58%	178s	0.89s
C-OCC	**4.19%**	93.6s	1.03s	7.93%	8.19s	0.09s	1.72%	2641s	17.4s	20.14%	349s	0.94s
ML3	5.43%	29.5s	0.20s	3.43%	13.2s	0.11s	**1.59%**	745s	2.78s	14.84%	225s	0.44s

the implementation privately provided by the authors, while for G-OCC and C-OCC we used 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 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 learning algorithms such as linear SVM (with a one vs all multiclass extension), multiclass linear SVM (Crammer and Singer, 2001), SVM with Gaussian kernel and 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 5-fold cross-validation on each training split and for ML3 we fix $p = 1.5$ (as discussed in Section 4). All the local learning algorithms are compared using the same number of local models, except where explicitly mentioned. For LLSVM we used the same manifold learning settings as in Ladicky and Torr (2011) (encoding based on inverse Euclidean distances to the 8 nearest cluster centers).

Character recognition. 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. LETTER (Frank and Asuncion, 2010) is a dataset 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. 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 these databases we normalize and center the data. Finally, COVTYPE (Joachims, 2006) is class 1 in the Covertype dataset 2 of Blackard, Jock & Dean and it consists of 522,911 54-dimensional training samples and 58,101 test samples. Although it is not a character recognition dataset, we included it here as an example of a large (in terms of number of samples) problem.

Following Zhang et al. (2011) we set $m = 90$ for MNIST, $m = 80$ for USPS and $m = 16$ for LETTER. For COVTYPE we set $m = 54$, which is the maximum allowed by OCC. The number of CCCP iterations (or s.g.d. epochs, for LLSVM/OCC) is set to 30 and in order to learn a bias for each local model, for ML3 and OCC 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 iterations, fixing m to 100. Note also that, since the LETTER and COVTYPE dataset consists of 16 and 54-dimensional instances, the maximum number of orthogonal coordinates are, respectively, 16 and 54. The OCC plots for these datasets are thus obtained with $m = 16$ and $m = 54$.

We see that for LETTER, MNIST and COVTYPE, ML3 obtains the state of the art for the class of locally linear SVMs algorithms, with performances close to the ones achieved by the Gaussian-Libsvm. Moreover, on COVTYPE the training and testing times of ML3 result to be several order of magnitude lower than the ones obtained by Gaussian-Libsvm. For USPS, the results are on par with the majority of the manifold learning algorithms and with the Gaussian-Libsvm, with Improved LCC and C-OCC obtaining better results.

The timings reported in Table 1 were measured by averaging the measurements of five different runs, on a single thread of an Intel(R) Core(TM) i7-2600K, with 16GB of RAM. As for LLSVM, OCC, Liblinear and Libsvm, ML3 was developed using a mixed Maltab/C++ implementation, with the main algorithm being implemented in a mex file[2]. Amongst the locally linear SVMs algorithms, the one with the lowest training and testing times results to be the original LLSVM. This is likely due to the fact that the encoding used by this algorithm makes use of only the eight closest anchor points, forcing all the other coefficients to be zero and thus saving many computations. On the other hand, even though ML3 computes non-sparse and class-specific weights for each sample, while solving the original multiclass problem, its training times are often on par or lower then those of G-OCC. The only major exception happens on the LETTER dataset, which is the one with the highest number of classes. Still, on this dataset the training times of ML3 are lower than those obtained by MC-Liblinear, with an error-rate that is also almost one order of magnitude lower. The training times of ML3 result to be 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 $O\left(\min\{nd^2, dn^2\}\right)$.

Finally, we also note that in all the experiments the testing times of ML3 result to be from one to four orders of magnitude lower than those of Gaussian-Libsvm.

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 between 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 his-

2. The software is freely available at http://www.idiap.ch/~mfornoni/code.html

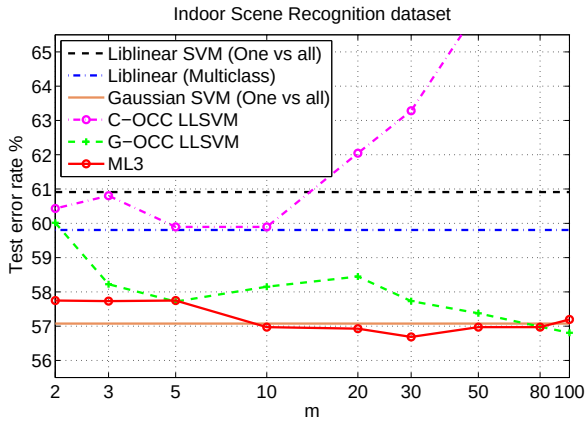

Figure 3: Average test error rate on the ISR dataset, varying the number of models m.

togram, 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 compact, but highly discriminative 4096-dimensional descriptor, specifically designed to work with linear classifiers.

The standard benchmarking procedure for this 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 test error rate 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 test error achieved by multiclass linear SVM, and by linear and Gaussian SVM.

On this dataset both ML3 and G-OCC are able to achieve and slightly outperform the Gaussian 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 ten models it is already able to match the performances of the Gaussian SVM. 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 ($d = 4096$).

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 (L-MKL) (Gönen and Alpaydin, 2008), using the MATLAB implementation available on the website of the authors and m linear kernels. For the Banana dataset we use 400 samples for training and the remaining for testing. For all the other 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 five folds that are used to select the regularization parameter. Each experiment was repeated ten times on ten different training / testing splits[3] and the average test error rate is reported in Table 2. For these experiments the number of local models was fixed to $m = 10$

3. For L-MKL we had to reduce the number of splilts and CV-folds, as it resulted to be unable to complete the benchmark in a reasonable time. For similar reasons the algorithm was not tested on other datasets.

Table 2: Average test error rate and ranking on the UCI benchmark datasets.

	Gaussian	**Liblinear**	**L-MKL**	**LL-SVM**	**G-OCC**	**C-OCC**	**ML3**
Banana	11.06	45.13	11.69	12.58	36.99	35.24	**11.03**
German	**23.59**	25.15	27.19	27.75	24.13	25.57	23.65
Heart	16.89	17.11	18.67	**15.78**	26.78	24.56	18.00
Ionosphere	**10.09**	21.54	14.70	14.02	10.85	11.79	11.88
Liver	33.30	32.87	33.04	35.13	**29.57**	31.91	32.17
PIMA	23.28	24.80	25.16	23.01	24.80	26.88	**22.50**
Ringnorm	**1.56**	22.98	13.03	3.02	19.07	18.72	7.59
Sonar	30.57	33.43	32.86	31.29	29.86	**28.29**	29.86
Spambase	**8.42**	11.57	9.39	23.21	18.95	15.21	11.23
WDBC	**9.87**	14.21	13.56	12.58	11.46	11.20	10.86
Avg. Rank	**2.2**	5.4	4.9	4.5	4.2	4.2	2.6

and, as before, for ML3 and OCC we concatenated 1 to each instance vector. 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 LL-SVM, G-OCC and C-OCC, with $p = 0.0419$, $p = 0.0295$ and $p = 0.0166$, 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, for example 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 implementation, by 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 will test ML3 on large scale classification datasets and perform experiments using manifold learning, as an initialization step for the local weights. From a theoretical side, we will explore different routes to optimize the objective function of ML3.

Acknowledgements This work was partially supported (M.F.) by the SNF grant ICS - Interactive Cognitive Systems.

References

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999. ISBN 1886529000.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 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, 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. *Journal of Machine Learning Research*, 7:1–30, 2006.

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.

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, 2006.

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, 2011.

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

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.

A. Smola, S. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

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

Proof [Closed-form solution for $\boldsymbol{\beta}^*_{\boldsymbol{W},i,y}$] Define $\boldsymbol{c} = \boldsymbol{W}_y \boldsymbol{x}_i$ and decompose the objective function (11) into two sums: one corresponding to the positive c_i, and the other corresponding to the negative c_i. As the β_i are constrained to be non-negative, it follows that all the β_i associated with negative c_i need to be zero. For the remaining β_i, the problem is equivalent to find an $\boldsymbol{\alpha}$ on the p-unit ball such that $\boldsymbol{\alpha}^\top \boldsymbol{d}$ is maximized for a vector $\boldsymbol{d}$, whose elements are the positive entries of $\boldsymbol{c}$. The solution of this problem is the point on the surface of the p-unit ball such that its tangent plane has a normal vector parallel to $\boldsymbol{d}$. For the case $p > 1$, define

$$F(\boldsymbol{\alpha}) = \left(\sum_i \alpha_i^p \right)^{\frac{1}{p}} - 1 = 0, \tag{17}$$

as the equation of the points on the surface of the p-unit ball. The j-th coordinate of the normal to the tangent plane of this surface is defined as $\nabla_{\alpha_j} F(\boldsymbol{\alpha}) = \left(\sum_i \alpha_i^p\right)^{\frac{1-p}{p}} \alpha_j^{p-1}$. In order for $\nabla_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha})$ to be parallel to $\boldsymbol{d}$ we have to enforce $\nabla_{\alpha_j} F(\boldsymbol{x}) = \theta d_j$ with $\theta > 0$, which gives:

$$\alpha_j = \left(\frac{\theta d_j}{\left(\sum_i \alpha_i^p\right)^{\frac{1-p}{p}}} \right)^{\frac{1}{p-1}}. \tag{18}$$

By plugging (18) into (17) we obtain $\theta = \frac{\left(\sum_i \alpha_i^p\right)^{\frac{1-p}{p}}}{\|\boldsymbol{d}\|_q}$, which in turn provides the solution $\alpha_j = \left(\frac{d_j}{\|\boldsymbol{d}\|_q}\right)^{q-1}$. Since $\|\boldsymbol{d}\|_q = \|\boldsymbol{c}^+\|_q$, the final solution can thus be written as $\beta_j^* = \left(\frac{c_j^+}{\|\boldsymbol{c}^+\|_q}\right)^{q-1}$. For the case $p = 1$, the tangent plane is fixed, hence the solution reduces to extreme values of the p-unit ball, and can also be obtained by taking the limit for $p \to 1$ of the previous solution. ■