Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Optimizing Locally Linear Classifiers with Supervised Anchor Point Learning

Xue Mao[1], **Zhouyu Fu**[2], **Ou Wu**[1], **Weiming Hu**[1]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences, China
[2]School of Computing, Engineering & Mathematics, University of Western Sydney, Australia
{xue.mao, wuou, wmhu}@nlpr.ia.ac.cn, Z.Fu@uws.edu.au

Abstract

Kernel SVM suffers from high computational complexity when dealing with large-scale nonlinear datasets. To address this issue, locally linear classifiers have been proposed for approximating nonlinear decision boundaries with locally linear functions using a local coding scheme. The effectiveness of such coding scheme depends heavily on the quality of anchor points chosen to produce the local codes. Existing methods usually involve a phase of unsupervised anchor point learning followed by supervised classifier learning. Thus, the anchor points and classifiers are obtained separately whereas the learned anchor points may not be optimal for the discriminative task. In this paper, we present a novel fully supervised approach for anchor point learning. A single optimization problem is formulated over both anchor point and classifier variables, optimizing the initial anchor points jointly with the classifiers to minimize the classification risk. Experimental results show that our method outperforms other competitive methods which employ unsupervised anchor point learning and achieves performance on par with the kernel SVM albeit with much improved efficiency.

1 Introduction

Kernel Support Vector Machine (SVM) [Schölkopf and Smola, 2002] is a widely used classification technique. Although effective in dealing with nonlinear data, kernel SVM is computationally expensive especially for large datasets. To address this issue, locally linear classifiers [Ladicky and Torr, 2011; Yu *et al.*, 2009] have been proposed to approximate nonlinear decision boundaries with locally linear classifiers using a local coding scheme. Since a nonlinear manifold behaves linearly in the local neighborhood, data on the manifold can be encoded locally in a local coordinate system established by a set of anchor points. Each data point can then be approximated with a linear combination of surrounding anchor points, and the linear coefficients are local coordinate values that can be used for subsequent classifier training.

Although effective, the classification performance of locally linear classifiers depends heavily on the quality of the local coding, which further depends on the anchor points being used. Current locally linear classification methods learn the anchor points and classifiers in two separate steps. In the first step, anchor points are learned and used to encode the training data with a local coding scheme. After that, supervised classifier training is performed based on the results of encoding. One major issue with this decoupled approach is that the purposes of each step are different. While local coding techniques mainly focus on minimizing the data reconstruction error and exploit unsupervised learning algorithms such as clustering [van Gemert *et al.*, 2008; Zhou *et al.*, 2009] for anchor point learning, the main purpose of classifier learning is to minimize classification error. Since the anchor points are obtained in an unsupervised fashion without making use of class label information, discriminative information might be lost in the encoding step and the local coding produced by these anchor points may not be optimal for the classification task. Therefore, an optimized classification model should exploit supervised information for anchor point learning and update both the anchor points and classifier simultaneously for further performance improvement.

In this paper, we propose an efficient nonlinear classification algorithm based on the Locally Linear Classifiers with Supervised Anchor Point Learning (LLC-SAPL). We take a fully supervised approach for learning both anchor points and classifiers jointly in a discriminative framework. Firstly, we use a localized soft-assignment coding scheme [Liu *et al.*, 2011] for encoding the training data. The use of soft-assignment coding enables us to express the local coordinates in analytic form. With these local coordinates, any nonlinear but smooth function defining the classifier can be approximated using multiple linear functions. We can then formulate a single optimization problem over both anchor point and classifier variables that can be efficiently solved using stochastic gradient descent. Instead of choosing anchor points independently from the classifiers, our LLC-SAPL model is capable of refining the initial anchor points while simultaneously updating the classifier to minimize the regularized classification risk. Moreover, the stochastic gradient descent is simple and fast so that it can be applied to large datasets in an online fashion. Experimental results on benchmark datasets show that LLC-SAPL outperforms state-of-the-art methods with unsupervised anchor point learning and achieves accuracy rates on par with the kernel SVM.

However, LLC-SAPL achieves much higher efficiency than kernel SVM in prediction.

2 Related Work

Local coding methods provide a useful tool for approximating data on the nonlinear manifold. Different encoding schemes have been proposed in the literature, including local soft-assignment coding [van Gemert *et al.*, 2008; Liu *et al.*, 2011], Local Coordinate Coding (LCC) [Yu *et al.*, 2009]. All these methods employ a set of anchor points to encode data as a linear combination of surrounding anchor points, so as to minimize the approximation error. The anchor points are learned in unsupervised fashion either by using a clustering algorithm or solving a locality or sparsity constrained least squares optimization problem for direct minimization of the reconstruction error.

A number of locally linear classifiers have been proposed based on local coding methods, most notably LCC+SVM [Yu *et al.*, 2009] and LLSVM [Ladicky and Torr, 2011]. LCC+SVM applies a linear SVM directly to the local coordinates obtained by the LCC [Yu *et al.*, 2009] scheme by treating them as feature values. Alternatively, LLSVM treats the local coordinates as weight values for assigning training data into different local regions. Separate linear classifiers are then trained for the weighted data points in each local region and combined to form a locally linear classifier.

Apart from local coding based methods, there exist other locally linear classifiers, such as SVM-KNN [Zhang *et al.*, 2006], CSVM [Gu and Han, 2013] and LDKL [Jose *et al.*, 2013]. SVM-KNN employs a lazy learning strategy by training a linear SVM in the subregion of a testing example and using the trained SVM classifier for prediction. CSVM adopts K-means to partition the data into clusters and then trains a linear SVM for each cluster. Meanwhile, CSVM requires the weight vector of each linear SVM to align with a global weight vector, which can be treated as a type of regularization. LDKL learns a tree-based primal feature embedding that encodes non-linearities, and then combines this feature embedding with the SVM classifier. Same as the proposed LLC-SAPL, LDKL is a fully supervised technique where the trees are built by utilizing the label information.

An alternative approach for large-scale kernel classification is the Nyström method [Li *et al.*, 2010; Williams and Seeger, 2001; Yang *et al.*, 2012], that provides a low rank approximation to the kernel matrix by sampling a subset of columns. As a generic technique for matrix factorization, the Nyström method is not specifically designed for the classification task and does not take into account label information in the factorization process. Hence discriminative information may be lost with the Nyström approximation, which leads to sub-optimal solution for the classifier model.

3 LLC-SAPL Model

3.1 Localized Soft-assignment Coding

We first introduce the local coding scheme employed by the LLC-SAPL model. Let $\{\mathbf{v}_j\}_{j=1}^m$ denote the set of m anchor points generated by K-means clustering. Each data point $\mathbf{x}$ is then approximated by a linear combination of surrounding anchor points:

$$\mathbf{x} \approx \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x})\mathbf{v}_j \tag{1}$$

The coefficient $\gamma_{\mathbf{v}_j}(\mathbf{x})$ specifies the degree of membership of $\mathbf{x}$ to the jth anchor point $\mathbf{v}_j$ and is defined by:

$$\gamma_{\mathbf{v}_j}(\mathbf{x}) = \begin{cases} \dfrac{\exp(-\beta d(\mathbf{x}, \mathbf{v}_j))}{\sum_{l \in N_k(\mathbf{x})} \exp(-\beta d(\mathbf{x}, \mathbf{v}_l))} & j \in N_k(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $d(\mathbf{x}, \mathbf{v}_j)$ is the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{v}_j$, and $N_k(\mathbf{x})$ denotes the set of indices of k-nearest anchor points to $\mathbf{x}$. Notice here only the k-nearest anchor points are considered for encoding each data point $\mathbf{x}$ with k nonzero coefficients, and the coefficients for the remaining anchor points are all set to zero. In contrast, a global coding scheme expands a data point to all the anchor points and ignores the underlying local manifold structure, which leads to unreliable estimation of the membership to distant anchor points. By assigning a data point only to its surrounding anchor points, the above "localized" soft-assignment coding scheme alleviates this problem and achieves computational efficiency at the same time due to the sparse representation.

3.2 A Unified Framework for Learning Anchor Points and Locally Linear Classifier

We next integrate the above local coding scheme into locally linear classifier, leading to a unified framework for learning both the anchor points and the locally linear classifier. In this paper, we focus on the linear SVM classifier, but the technique discussed below can be applied to other linear classifiers by modifying the loss function. Given the labeled binary dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^D$ and $y_n \in \{-1, 1\}$, one can solve the optimization problem below for linear SVM training:

$$\min_{\mathbf{w}, b} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\mathbf{x}_n)) \tag{3}$$

$$\ell(y_n, f(\mathbf{x}_n)) = \max(0, 1 - y_n f(\mathbf{x}_n)) \tag{4}$$

$$f(\mathbf{x}_n) = \mathbf{w}^T\mathbf{x}_n + b \tag{5}$$

where the first term in Equation (3) is the regularization term on the classifier weight vector $\mathbf{w}$ and the second term is the average hinge loss incurred. $f(\mathbf{x})$ is the decision function for linear SVM.

Although linear SVM is extremely efficient, it cannot handle nonlinear decision boundaries with acceptable accuracy due to the use of linear prediction function $f(\mathbf{x})$. In a sufficiently small region, a nonlinear decision boundary is approximately linear and data is locally linearly separable. To encode this local linearity, the weight vector $\mathbf{w}$ and bias b of the SVM classifier should vary according to the location of the point $\mathbf{x}$ in the feature space as:

$$f(\mathbf{x}) = \mathbf{w}(\mathbf{x})^T\mathbf{x} + b(\mathbf{x}) = \sum_{d=1}^{D} w_d(\mathbf{x})x_d + b(\mathbf{x}) \tag{6}$$

The dimensionality of feature vector $\mathbf{x}$ is D.

To deal with nonlinear data, a smooth decision boundary with constrained curvature is desired, since an unconstrained decision boundary is likely to overfit [Ladicky and Torr, 2011]. Smoothness and constrained curvature imply that the functions $\mathbf{w}(\mathbf{x})$ and $b(\mathbf{x})$ are Lipschitz smooth in the feature space $\mathbf{x}$. Thus according to [Yu *et al.*, 2009], for a local coding scheme defined by anchor points $\mathbf{v}$, any Lipschitz smooth function $h(\mathbf{x})$ defined on a lower dimensional manifold can be approximated by a linear combination of function values $h(\mathbf{v})$ over the set of anchor points as:

$$h(\mathbf{x}) = \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) h(\mathbf{v}_j) \tag{7}$$

Thus we can approximate the weight functions $w_d(\mathbf{x})$ and bias function $b(\mathbf{x})$ in Equation (6) using the localized soft-assignment coding as:

$$w_d(\mathbf{x}) = \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) w_d(\mathbf{v}_j) \tag{8}$$

$$b(\mathbf{x}) = \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) b(\mathbf{v}_j) \tag{9}$$

Substituting the above equations into Equation (6), we obtain:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{d=1}^{D}\sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) w_d(\mathbf{v}_j) x_d + \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) b(\mathbf{v}_j) \\ &= \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) \Big(\sum_{d=1}^{D} w_d(\mathbf{v}_j) x_d + b(\mathbf{v}_j) \Big) \\ &= \sum_{j=1}^{m} \gamma_{\mathbf{v}_j}(\mathbf{x}) \left(\mathbf{w}(\mathbf{v}_j)^T \mathbf{x} + b(\mathbf{v}_j) \right) \end{aligned} \tag{10}$$

This transformation can be seen as a finite kernel transforming a D-dimensional problem into a mD-dimensional one. It can also be interpreted as defining a locally linear classifier as the weighted sum of m separate linear classifiers for each anchor point, where the weights are determined by the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$. To evaluate $f(\mathbf{x})$ for each data point $\mathbf{x}$, one needs to calculate the corresponding local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$, which further depend on the anchor points $\mathbf{v}$ being used for local coding. This leads to a natural two-step approach taken by existing methods [Yu *et al.*, 2009; Ladicky and Torr, 2011], which involves learning the anchor points first and evaluating the local codes for each data point, followed by the subsequent step of supervised classifier training. The anchor points are learned separately from the classifier, and are directly obtained using unsupervised learning techniques. This may lead to sub-optimal solutions for the discriminative classification task.

Let $\mathbf{W} = [\mathbf{w}(\mathbf{v}_1), \dots, \mathbf{w}(\mathbf{v}_m)]^T$ be a $m \times D$ matrix composed by stacking the m classifier weight vectors in rows. Let $\mathbf{b} = [b(\mathbf{v}_1), \dots, b(\mathbf{v}_m)]^T$ and $\gamma_{\mathbf{v}}(\mathbf{x}) = [\gamma_{\mathbf{v}_1}(\mathbf{x}), \dots, \gamma_{\mathbf{v}_m}(\mathbf{x})]^T$ be m-dimensional vectors of bias terms and local coordinates respectively. The decision function of the locally linear classifier in Equation (10) can then be written as:

$$f_{\mathbf{W},\mathbf{b},\mathbf{v}}(\mathbf{x}) = \gamma_{\mathbf{v}}(\mathbf{x})^T \mathbf{W}\mathbf{x} + \gamma_{\mathbf{v}}(\mathbf{x})^T \mathbf{b} \tag{11}$$

Note that the decision function $f_{\mathbf{W},\mathbf{b},\mathbf{v}}(\mathbf{x})$ in Equation (11) depends on both the classifier variables $\mathbf{W}$, $\mathbf{b}$ and the anchor point variable $\mathbf{v}$, which determines the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$. This motivates an optimization-based approach for jointly learning both the classifier and the anchor points. Using a similar formulation to Equation (3), we define the LLC-SAPL optimization problem as follows:

$$\min_{\mathbf{W},\mathbf{b},\mathbf{v}} Q(\mathbf{W},\mathbf{b},\mathbf{v}) = \frac{\lambda}{2}\|\mathbf{W}\|_F^2 + \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f_{\mathbf{W},\mathbf{b},\mathbf{v}}(\mathbf{x}_n)) \tag{12}$$

where $\|\mathbf{W}\|_F^2 = \sum_{j=1}^{m}\sum_{d=1}^{D} W_{j,d}^2$. Note that here the anchor point variable $\mathbf{v}$ is treated as one of the target variables to be optimized in the objective function Q. This is made possible by the use of localized soft-assignment coding scheme in our approach, where the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$ are sub-differentiable to the anchor point variable $\mathbf{v}$. In contrast, when using the standard sparse coding schemes, there is no closed-form solution to local coordinate values. Hence, it is infeasible to incorporate them into the optimization problem formulated above.

The use of embedded optimization for both feature and classifier variables in Equation (12) is crucial to the success of our approach. A naïve approach for selecting anchor points with supervised information is not guaranteed to retain the discriminative information for classification. Consequently, the selected anchor points may not be optimal for the linear classifier being used. On the contrary, our approach provides a principled solution to joint anchor point optimization and classifier training instead of treating them separately.

The LLC-SAPL formulation in Equation (12) defines a generic framework that includes LLSVM [Ladicky and Torr, 2011] and LCC+SVM [Yu *et al.*, 2009] as special cases. Specifically, it reduces to LLSVM if the anchor points $\mathbf{v}$ are fixed and only the classifier variables $\mathbf{W}$ and $\mathbf{b}$ are optimized. LLSVM further reduces to the degenerate case LCC+SVM if $\mathbf{W} = \mathbf{0}$ and only $\mathbf{b}$ is optimized. In this case, the decision function reduces to a linear classifier where the local coordinate values $\gamma_{\mathbf{v}}(\mathbf{x})$ are used as input features and the bias vector $\mathbf{b}$ becomes the weight vector for the linear classifier. We can improve either LLSVM or LCC+SVM with the proposed LLC-SAPL framework by minimizing the objective function Q in Equation (12) with respect to both the anchor points and classifier variables. In the following discussion, we focus on the more generic scenario of LLC-SAPL, which can be treated as the fully supervised counterpart of LLSVM. However, the techniques presented below can be easily adapted to obtain optimized anchor points for the degenerate case LCC+SVM, which we refer to as LLC-SAPL$_{lcc}$ in subsequent text.

3.3 The Stochastic Gradient Descent Algorithm for LLC-SAPL

The Stochastic Gradient Descent (SGD) algorithm is simple and efficient, and can be applied to large datasets in an online fashion. Empirical results have shown that SGD significantly outperforms other complex optimization methods for training linear SVMs [Bordes *et al.*, 2009; Shalev-Shwartz *et al.*, 2007]. We now apply the SGD algorithm to the LLC-SAPL formulation in Equation (12). Each iteration of SGD involves drawing a random data point $\mathbf{x}$ and its corresponding label y and updating the current anchor points $\mathbf{v}$ and classifier variables $\mathbf{W}, \mathbf{b}$ if the hinge loss is positive. For updating anchor points $\mathbf{v}$, we take the partial derivative of Q in Equation (12) with respect to $\mathbf{v}$ while fixing $\mathbf{W}, \mathbf{b}$. Since the data point $\mathbf{x}$ is approximated as a linear combination of its k-nearest anchor points, only the k-nearest anchor points need to be optimized in each iteration. Assume that the jth anchor point $\mathbf{v}_j$ belongs to the k-nearest neighbors of $\mathbf{x}$. Firstly, we take the partial derivative of $\gamma_{\mathbf{v}}(\mathbf{x})^T$ with respect to $\mathbf{v}_j$. The obtained derivative $\frac{\partial \gamma_{\mathbf{v}}(\mathbf{x})^T}{\partial \mathbf{v}_j}$ is a $D \times m$ matrix, among which only k columns are nonzero. The jth column is computed as:

$$\frac{\mathbf{s}\big(\sum_{l \in N_k(\mathbf{x})} \exp(-\beta d(\mathbf{x}, \mathbf{v}_l)) - \exp(-\beta d(\mathbf{x}, \mathbf{v}_j))\big)}{\big(\sum_{l \in N_k(\mathbf{x})} \exp(-\beta d(\mathbf{x}, \mathbf{v}_l))\big)^2} \tag{13}$$

where

$$\mathbf{s} = \frac{\partial \exp(-\beta d(\mathbf{x}, \mathbf{v}_j))}{\partial \mathbf{v}_j} = 2\beta(\mathbf{x} - \mathbf{v}_j)\exp(-\beta d(\mathbf{x}, \mathbf{v}_j)) \tag{14}$$

The other nonzero columns except the jth column are computed as:

$$-\frac{\mathbf{s}\exp(-\beta d(\mathbf{x}, \mathbf{v}_h))}{\big(\sum_{l \in N_k(\mathbf{x})} \exp(-\beta d(\mathbf{x}, \mathbf{v}_l))\big)^2} \tag{15}$$

where $\mathbf{v}_h$ also belongs to the k-nearest neighbors of $\mathbf{x}$ and it is not equal to $\mathbf{v}_j$.

Then the jth anchor point $\mathbf{v}_j$ is updated as:

$$\mathbf{v}_j^{(t+1)} = \mathbf{v}_j^{(t)} + \frac{1}{\lambda(t+t_0)} y \frac{\partial \gamma_{\mathbf{v}}(\mathbf{x})^T}{\partial \mathbf{v}_j}(\mathbf{W}^{(t)}\mathbf{x} + \mathbf{b}^{(t)}) \tag{16}$$

where t denotes the current iteration number. The optimal learning rate is denoted as $\frac{1}{\lambda(t+t_0)}$ [Shalev-Shwartz *et al.*, 2007], where t_0 is a positive constant [Bordes *et al.*, 2009], that avoids producing too large steps in the first few iterations. The classifier variables $\mathbf{W}$ and $\mathbf{b}$ are updated as:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \frac{1}{\lambda(t+t_0)} y\big(\gamma_{\mathbf{v}}(\mathbf{x})\mathbf{x}^T\big) \tag{17}$$

$$\mathbf{b}^{(t+1)} = \mathbf{b}^{(t)} + \frac{1}{\lambda(t+t_0)} y \gamma_{\mathbf{v}}(\mathbf{x}) \tag{18}$$

To speed up the training process, we adopt a similar strategy to [Bordes *et al.*, 2009] and perform a regularization update to the weight matrix $\mathbf{W}$ every $skip$ iterations by:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t+1)} - \frac{skip}{t+t_0}\mathbf{W}^{(t+1)} \tag{19}$$

Our LLC-SAPL algorithm is presented in Algorithm 1. K-means is utilized to initialize the set of anchor points $\mathbf{v}$, which are then used to encode the training data with the localized soft-assignment coding scheme and obtain the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$. To initialize the classifier weights $\mathbf{W}$ and biases $\mathbf{b}$, the following transformation is applied to each data point in the training set

$$\mathbf{z} = \gamma_{\mathbf{v}}(\mathbf{x}) \otimes [\mathbf{x}; 1] \tag{20}$$

where the operator $\otimes$ is the Kronecker product. This is equivalent to appending 1 to the column vector $\mathbf{x}$ and multiplying each element of $\gamma_{\mathbf{v}}(\mathbf{x})$ with the augmented feature vector. Then we apply LibLinear [Fan *et al.*, 2008] to the new data $\{(\mathbf{z}_n, y_n)\}_{n=1}^N$ for obtaining the initial $\mathbf{W}$ and $\mathbf{b}$.

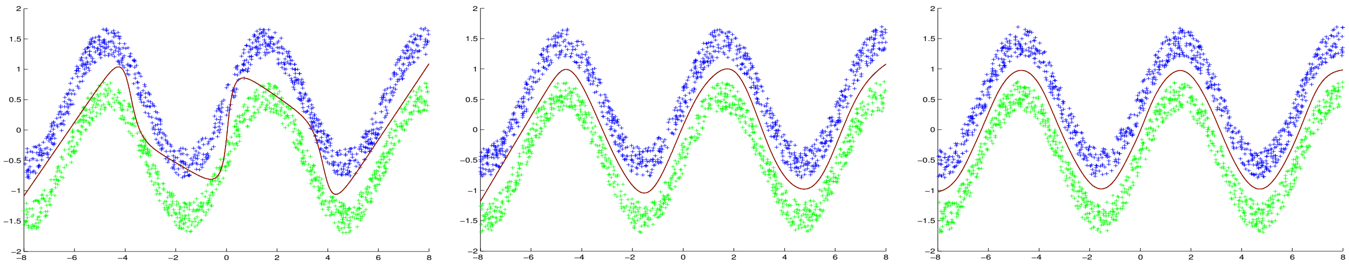
The SGD iterations start after the initialization steps. In each iteration, a subset of anchors corresponding to the k-nearest neighbors of the selected data point are updated along with the classifier variables if a positive hinge loss is incurred. In this way, all the anchor points are updated after one epoch (a full pass through the whole training set).

For the degenerate version LLC-SAPL$_{lcc}$, we need to make some minor modifications to the LLC-SAPL algorithm. Most importantly, we need to set all occurrences of $\mathbf{W}$ to $\mathbf{0}$ in the relevant algorithm steps and equations for LLC-SAPL$_{lcc}$ and ignore those steps that involve updating $\mathbf{W}$ in Algorithm 1. To obtain initial $\mathbf{b}$, we only need to train a linear SVM on the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$. Moreover, as the bias vector $\mathbf{b}$ serves as the weight vector of the globally linear SVM, we should also apply regularization updates via Equation (19) by substituting $\mathbf{W}$ with $\mathbf{b}$ in the equation, in addition to the regular updates of $\mathbf{b}$ in Equation (18).

For prediction, the test data are encoded using the learned anchor points and then classified by the locally linear classifier. The prediction complexity is linear in the number of an-

Algorithm 1 LLC-SAPL

Input: Training data $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{-1, 1\}$, the number of anchor points m and parameters λ, t_0, T, $skip$
Output: Classifier variables $\mathbf{W}$, $\mathbf{b}$ and anchor points $\mathbf{v}$
  Initialize anchor points $\mathbf{v}$ by K-means.
  Convert each $\mathbf{x}$ to $\mathbf{z}$ via Equation (20) and initialize classifier variables $\mathbf{W}$, $\mathbf{b}$ by training a linear SVM on the transformed dataset $\{(\mathbf{z}_n, y_n)\}_{n=1}^N$.
  $t = 0$
  while $t \leq T$ **do**
    Sample a data point $\mathbf{x}$ randomly.
    Compute the local coordinates $\gamma_{\mathbf{v}}(\mathbf{x})$ via Equation (2).
    Compute hinge loss $L = 1 - y\big(\gamma_{\mathbf{v}}(\mathbf{x})^T\mathbf{W}\mathbf{x} + \gamma_{\mathbf{v}}(\mathbf{x})^T\mathbf{b}\big)$.
    if $L > 0$ **then**
      for $j = 1 \to k$ **do**
        Update the jth nearest anchor point of $\mathbf{x}$ via Equation (16).
      end for
      Update $\mathbf{W}$ via Equation (17).
      Update $\mathbf{b}$ via Equation (18).
    end if
    if $t \bmod skip == 0$ **then**
      Update $\mathbf{W}$ via Equation (19).
    end if
    $t = t + 1$
  end while



(a) LLSVM (b) LLC-SAPL (c) Kernel SVM

Figure 1: Learned classifiers on the synthetic Sine dataset

chor points while the prediction complexity of kernel SVM scales with the number of support vectors. For large datasets, the number of support vectors is usually orders of magnitude larger than that of anchor points. Hence the proposed algorithm is much more efficient than kernel SVM in prediction, which is critical for large-scale or online applications.

3.4 Extension to Multi-class Classification

For multi-class classification problems, we can combine multiple binary classifiers discussed above following standard one-vs-all strategy. However, in this case, the multi-class algorithm learns class-specific anchor points, with different anchor points obtained for different binary classification problems. This leads to significant computational overhead for multi-class problems with many classes, as the number of anchor points produced scales by mI, where I denotes the number of classes. Hence it is desirable to share the same anchor points among different classes, which leads to the following formulation for multi-class classification problem:

$$\arg\min_{\mathbf{W}_{i=1}^{I}, \mathbf{b}_{i=1}^{I}, \mathbf{v}} \frac{\lambda}{2}\sum_{i=1}^{I}\|\mathbf{W}_i\|_F^2 + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{I}\ell(y_{n,i}, f_{\mathbf{W}_i,\mathbf{b}_i,\mathbf{v}}(\mathbf{x}_n)) \tag{21}$$

$$f_{\mathbf{W}_i,\mathbf{b}_i,\mathbf{v}}(\mathbf{x}_n) = \gamma_{\mathbf{v}}(\mathbf{x}_n)^T\mathbf{W}_i\mathbf{x}_n + \gamma_{\mathbf{v}}(\mathbf{x}_n)^T\mathbf{b}_i$$

The variables $\mathbf{W}_{i=1}^{I}$ and $\mathbf{b}_{i=1}^{I}$ represent the weight matrices and bias vectors of the locally linear classifiers for I classes. The label $y_{n,i}$ equals 1 if the nth example belongs to the ith class and -1 otherwise. Compared with the LLC-SAPL formulation for binary classification in Equation (12), the multi-class formulation in (21) learns different weight matrices and bias vectors for different classes while sharing the same anchor points among classes. This is much cheaper than inducing different anchor points for different classes with the naïve one-vs-all approach, and scales our algorithm up to large multi-class datasets nicely. This multi-class classification problem can be solved by the same SGD algorithm presented in Algorithm 1 subject to slight modifications.

4 Experiments

4.1 Synthetic Dataset

The synthetic dataset contains points randomly sampled from two Sine signals, as shown in Figure 1. Each signal contains 1000 points and forms one class. We then apply LLSVM, LLC-SAPL and Kernel SVM to this dataset. The number of anchor points is set to four for both LLSVM and LLC-SAPL. The decision boundaries produced by the above methods are displayed by red curves in Figures 1(a), (b) and (c) respectively. These results clearly show that LLSVM produces sub-optimal decision boundary compared to the other two methods. Since LLSVM employs unsupervised anchor point learning and fixes the anchor points for subsequent classifier learning, the anchor points thus obtained may not be optimized for the learned classifier and be likely to lead to degenerate performance. On the contrary, our LLC-SAPL model utilizes a fully supervised approach for learning both the anchor points and classifier jointly. The decision boundary produced by LLC-SAPL is quite similar to that of kernel SVM. However, LLC-SAPL only uses four anchor points, while kernel SVM learns 186 support vectors. The prediction cost scales linearly with the number of anchor points for LLC-SAPL and the number of support vectors for kernel SVM. Therefore, LLC-SAPL is much more efficient than kernel SVM in the prediction phase.

4.2 Real Datasets

We use ten benchmark datasets: Banana, IJCNN, SKIN, Magic04, CIFAR, RCV1, USPS, MNIST, LETTER and MNIST8m. The Banana, USPS and MNIST datasets are used in [Rätsch *et al.*, 2001] [Hull, 1994] [LeCun *et al.*, 1998]. The IJCNN, RCV1 and MNIST8m datasets are obtained from the LibSVM website [Chang and Lin, 2011]. The preprocessed binary CIFAR dataset is taken from [Jose *et al.*, 2013]. The others are available at the UCI repository [Bache and Lichman, 2013]. The first six datasets are used for binary classification tasks, and the other four are multi-class datasets. All the datasets have been divided into training and testing sets except the Banana, SKIN, Magic04 and MNIST8m datasets. For Banana and Magic04, we randomly selected two thirds of examples for training and the rest for testing. For SKIN, we used half for training and the rest for testing. The MNIST8m dataset contains 8.1 million examples and was generated by performing careful elastic deformation of the original MNIST training set [Loosli *et al.*, 2007]. We used the first 8 million examples as training data and tested on the 10,000 examples in the original MNIST testing set. All the datasets are normalized to have zero mean and unit variance in each dimension. Table 1 gives a brief summary of these datasets. It can be seen that the datasets used in our experiments have a wide coverage of different scenarios, including high-dimensional

(RCV1) and large-scale data (MNIST8m).

Table 1: Summary of the real datasets in our experiments

Datasets	# training	# test	# features	# classes
Banana	3,533	1,767	2	2
IJCNN	49,990	91,701	22	2
SKIN	122,529	122,528	3	2
Magic04	12,680	6,340	10	2
CIFAR	50,000	10,000	400	2
RCV1	20,242	677,399	47,236	2
USPS	7,291	2,007	256	10
MNIST	60,000	10,000	784	10
LETTER	16,000	4,000	16	26
MNIST8m	8,000,000	10,000	784	10

We compare our methods with seven previously mentioned methods: linear SVM, kernel SVM, LLSVM, LCC+SVM, SVM-KNN, CSVM and LDKL. We use LibLinear [Fan *et al.*, 2008] and LibSVM [Chang and Lin, 2011] for training linear SVM and Gaussian kernel SVM respectively. The regularization parameter C is tuned by 5-fold cross validation on the training set for both linear and kernel SVMs. Cross validation is also used to tune the kernel width of the Gaussian kernel for kernel SVM, and the number of nearest neighbors for SVM-KNN. For the two proposed methods (LLC-SAPL and LLC-SAPL$_{lcc}$), LLSVM and LCC+SVM, we adopted the same parameter values suggested in [Ladicky and Torr, 2011] by setting the number of anchor points m to 100 and number of nearest neighbours k to 8 for the local coding step. The parameters of all the other methods are set as in their original papers, with most parameters set by cross validation. For those methods involving K-means clustering or other random factors, we take the average testing results and the standard deviation over 10 random repetitions. The comparison results in terms of classification accuracy and testing time are presented in Tables 2 and 3. Note that the results for kernel SVM and SVM-KNN are left empty for the RCV1 and MNIST8m datasets, since the large size of the datasets makes the training or testing of kernel SVM and SVM-KNN infeasible.

Unsurprisingly, linear SVM does not perform well on all the datasets as it can not handle nonlinear data. Kernel SVM achieves the best performance overall. Nevertheless, kernel SVM is expensive in prediction especially for large datasets as shown in Table 3. The proposed LLC-SAPL achieves comparable performance to kernel SVM, but it is much more efficient in prediction. This is due to the fact that LLC-SAPL uses much less number of anchor points compared to the large number of support vectors used by the kernel SVM model, which hinders efficiency in prediction. As a degenerate case of LLSVM, LCC+SVM does not perform as well as LLSVM. This is also true for LLC-SAPL and LLC-SAPL$_{lcc}$. However and most notably, LLC-SAPL outperforms LLSVM in classification accuracy on all datasets, and so does LLC-SAPL$_{lcc}$ and LCC+SVM. Since LLC-SAPL and LLC-SAPL$_{lcc}$ are supervised counterparts of LLSVM and LCC+SVM, this clearly demonstrates the effectiveness of anchor point learning for classification. Moreover, LLC-SAPL is as efficient as LLSVM for prediction since both follow the same procedure for prediction and use same number of anchor points. Hence we have only included the testing time results for LLC-SAPL in Table 3. For the same reason, the testing time results for LLC-SAPL$_{lcc}$ are included while the results of LCC+SVM are omitted.

The training speeds of LLC-SAPL and LLC-SAPL$_{lcc}$ are slower than LLSVM and LCC+SVM due to the update of anchor points during training. However, training LLC-SAPL is much faster than kernel SVM for large datasets. For each epoch, classifier learning amounts to training multiple linear SVMs over small subsets of training data, which can be efficiently solved and effectively parallelized for further speedup. The extra operations for searching and updating k nearest anchor points scale linearly with training size and anchor point number in each epoch. Therefore the whole SGD algorithm simply repeats linear operations over different epochs for anchor point optimization and linear SVM training. In contrast, there are no linear-time algorithms known for kernel SVM training, which scales poorly to large-scale data.

For other locally linear classifiers, both SVM-KNN and CSVM perform well on some datasets but not all of them. Moreover, SVM-KNN is time-consuming due to the nature of lazy learning. CSVM enforces alignment between the weight vectors of individual linear SVMs and a global weight vector, which is undesirable in some cases. Additionally, the performance of CSVM is heavily influenced by the initial K-means clustering results. Hence it is likely to perform poorly with improper initialization. LDKL achieves quite good performance overall, but LLC-SAPL still gains a margin of advantage over LDKL. The reason lies in the fact that LDKL learns a tree-based primal feature embedding to partition the feature space, which may lead to non-smooth partition over the feature space and abrupt change across region boundaries.

Now we investigate how the performance of LLC-SAPL varies over different numbers of anchor points. Figure 2 shows the classification accuracies of LLC-SAPL in solid line with the number of anchor points m ranging from 2 to 180. For comparison, we also display the accuracy values of LLSVM in broken line on the same plots. It can be clearly seen that the performances of both LLC-SAPL and LLSVM improve with increasing number of anchor points m and stabilize as m exceeds a certain threshold. After that, the changes in classification accuracy become quite small. Hence, both methods are not overly sensitive to the exact choice of m, which makes it easier for model selection. Notably, LLC-SAPL outperforms LLSVM for all different numbers of anchor points on all these datasets, which demonstrates the effectiveness of supervised anchor point learning. To achieve a similar performance as LLC-SAPL with 40 anchor points, LLSVM needs to use more than 100 anchor points for local coding, leading to higher prediction cost.

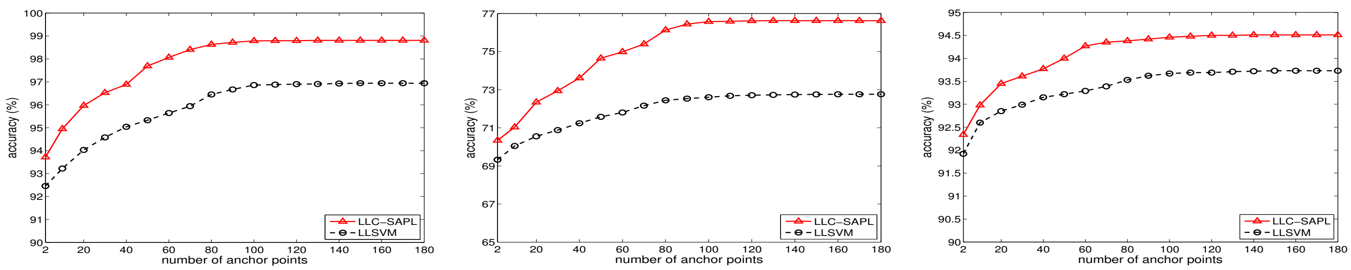
To further demonstrate the effectiveness of anchor point optimization, we record the objective function value as well as the classification accuracy on the testing set for each epoch and plot the values over different epochs. Figure 3 displays the epoch-wise results for optimization and classification, where red solid lines and black broken lines show the changes of accuracy values as well as objective function values respectively against epoch number. Due to space limitation, we only show the results of LLC-SAPL for two datasets, namely the

Table 2: Comparison of different classifiers in terms of classification accuracy (%)

Methods	Banana	IJCNN	SKIN	Magic04	CIFAR	RCV1	USPS	MNIST	LETTER	MNIST8m
Linear SVM	55.29	92.25	93.34	79.30	69.09	93.21	91.87	91.91	57.52	83.48
Kernel SVM	91.05	98.52	98.98	86.70	81.62	-	94.86	97.84	97.57	-
LLSVM	89.71±0.09	96.86±0.25	95.24±0.46	85.19±0.28	72.60±0.29	94.43±0.27	93.67±0.21	96.50±0.17	95.03±0.32	96.63±0.21
LLC-SAPL	90.82±0.13	98.79±0.32	98.57±0.14	86.53±0.23	76.57±0.23	95.48±0.29	94.46±0.26	97.67±0.22	97.27±0.18	97.96±0.24
LCC+SVM	89.47±0.17	91.58±0.37	94.04±0.08	80.24±0.19	65.48±0.25	87.62±0.13	89.43±0.33	80.35±0.15	77.54±0.20	82.79±0.35
LLC-SAPL$_{lcc}$	90.15±0.20	95.03±0.26	96.65±0.19	83.75±0.32	72.85±0.16	92.18±0.26	92.63±0.17	88.27±0.29	86.36±0.25	89.44±0.28
SVM-KNN	90.61	96.11	98.15	85.52	76.13	-	93.57	97.04	96.37	-
CSVM	89.64±0.17	97.25±0.48	97.41±0.29	85.01±0.51	73.37±0.21	94.77±0.25	92.85±0.18	95.75±0.24	94.58±0.13	96.02±0.37
LDKL	90.15±0.45	98.31±0.26	98.08±0.38	86.22±0.23	76.04±0.48	95.24±0.19	93.83±0.34	97.24±0.31	97.15±0.37	97.63±0.29

Table 3: Comparison of different classifiers in terms of testing time (in seconds)

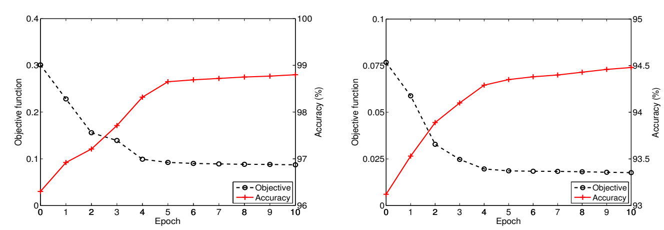
Methods	Banana	IJCNN	SKIN	Magic04	CIFAR	RCV1	USPS	MNIST	LETTER	MNIST8m
Linear SVM	0.0029	0.1475	0.0212	0.0043	0.1631	33.5286	0.0315	0.3339	0.0076	0.3427
Kernel SVM	0.74	357.06	68.58	10.82	594.08	-	43.17	867.59	13.49	-
LLC-SAPL	0.035±0.009	0.851±0.023	0.647±0.048	0.090±0.020	2.909±0.142	92.355±1.272	0.482±0.018	6.415±0.090	0.117±0.011	6.437±0.086
LLC-SAPL$_{lcc}$	0.009±0.002	0.187±0.004	0.075±0.020	0.014±0.005	0.076±0.011	57.728±1.063	0.012±0.006	0.067±0.008	0.027±0.006	0.063±0.009
SVM-KNN	9.57	5044.99	6857.23	94.05	3124.58	-	212.60	8758.74	34.54	-
CSVM	0.069±0.017	4.327±0.295	1.581±0.033	0.402±0.041	21.771±0.098	264.195±2.642	3.658±0.018	60.813±0.551	1.641±0.013	61.522±0.461
LDKL	0.010±0.004	0.391±0.032	0.275±0.027	0.026±0.004	0.900±0.029	68.097±0.925	0.098±0.008	1.731±0.080	0.082±0.004	1.958±0.074



(a) IJCNN (b) CIFAR (c) USPS

Figure 2: Comparison of classification accuracy for LLSVM and LLC-SAPL with respect to the number of anchor points

binary IJCNN dataset and the multi-class USPS dataset, in Figures 3(a) and (b). Similar observations can be made for both LLC-SAPL and LLC-SAPL$_{lcc}$ on other datasets. It can be seen clearly that the objective function values are monotonically decreasing over the epochs. Classification accuracies are closely correlated with objective function values, and lower objective function values usually lead to higher accuracies. The trend is especially obvious for the first 5 epochs. For the real datasets in our experiments, our algorithm generally converges within 10 epochs. Therefore, our model can be efficiently trained. These results clearly demonstrate the effectiveness of supervised anchor point learning.



(a) LLC-SAPL on IJCNN (b) LLC-SAPL on USPS

Figure 3: Epoch-wise demonstration of the proposed supervised anchor point learning approach

5 Conclusion

In this paper, we have proposed the LLC-SAPL model, a local coding based locally linear classifier model with supervised anchor point learning for nonlinear classification problems. Unlike previous methods that learn anchor points in an unsupervised fashion separately from supervised classifier learning, LLC-SAPL is a fully supervised method and learns the anchor points and classifier together by solving a single optimization problem with both variables. Consequently, LLC-SAPL is able to refine the initial anchor points to optimize classification performance.

In the future, we will look into the model selection problem and explore potential techniques [Zhu *et al.*, 2011; Fu *et al.*, 2010] for automatically selecting the optimal number of anchor points for the underlying problem.

Acknowledgments

This work is partly supported by NSFC (Grant No. 61379098, 61203239, 61472421) and Baidu research fund.

References

[Bache and Lichman, 2013] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[Bordes *et al.*, 2009] Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Fu *et al.*, 2010] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Mixing linear SVMs for nonlinear classification. *IEEE Transactions on Neural Networks*, 21(12):1963–1975, 2010.

[Gu and Han, 2013] Quanquan Gu and Jiawei Han. Clustered support vector machines. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, AISTATS '13, pages 307–315, 2013.

[Hull, 1994] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[Jose *et al.*, 2013] Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. Local deep kernel learning for efficient non-linear SVM prediction. In *Proceedings of the 30th International Conference on Machine Learning*, ICML '13, pages 486–494, 2013.

[Ladicky and Torr, 2011] Lubor Ladicky and Philip Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 985–992, 2011.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li *et al.*, 2010] Mu Li, James T Kwok, and B-L Lu. Making large-scale Nyström approximation possible. In *Proceedings of the 27th International Conference on Machine Learning*, ICML '10, pages 631–638, 2010.

[Liu *et al.*, 2011] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision*, ICCV '11, pages 2486–2493, 2011.

[Loosli *et al.*, 2007] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

[Rätsch *et al.*, 2001] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.

[Schölkopf and Smola, 2002] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[Shalev-Shwartz *et al.*, 2007] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 807–814, 2007.

[van Gemert *et al.*, 2008] Jan C van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders. Kernel codebooks for scene categorization. In *European Conference on Computer Vision*, ECCV '08, pages 696–709. Springer, 2008.

[Williams and Seeger, 2001] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, NIPS '01, pages 682–688, 2001.

[Yang *et al.*, 2012] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*, NIPS '12, pages 476–484, 2012.

[Yu *et al.*, 2009] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems*, NIPS '09, pages 2223–2231, 2009.

[Zhang *et al.*, 2006] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '06, pages 2126–2136, 2006.

[Zhou *et al.*, 2009] Xi Zhou, Na Cui, Zhen Li, Feng Liang, and Thomas S Huang. Hierarchical gaussianization for image classification. In *IEEE International Conference on Computer Vision*, ICCV '09, pages 1971–1977, 2009.

[Zhu *et al.*, 2011] Jun Zhu, Ning Chen, and Eric P. Xing. Infinite latent SVM for classification and multi-task learning. In *Advances in Neural Information Processing Systems*, NIPS '11, pages 1620–1628, 2011.