Local Deep Kernel Learning for Efficient Non-linear SVM Prediction

Cijo Jose, Prasoon Goyal, Parv Aggrwal {JOSEVANCIJO,PRASOONGOYAL13,PARV92}@GMAIL.COM
Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India 110 016

Manik Varma MANIK@MICROSOFT.COM
Microsoft Research India, Bangalore, India 560 001

Abstract

Our objective is to speed up non-linear SVM prediction while maintaining classification accuracy above an acceptable limit. We generalize Localized Multiple Kernel Learning so as to learn a tree-based primal feature embedding which is high dimensional and sparse. Primal based classification decouples prediction costs from the number of support vectors and our tree-structured features efficiently encode non-linearities while speeding up prediction exponentially over the state-of-the-art. We develop routines for optimizing over the space of tree-structured features and efficiently scale to problems with more than half a million training points. Experiments on benchmark data sets reveal that our formulation can reduce prediction costs by more than three orders of magnitude in some cases with a moderate sacrifice in classification accuracy as compared to RBF-SVMs. Furthermore, our formulation leads to better classification accuracies over leading methods.

1. Introduction

Real world applications often require efficient prediction. Non-linear SVMs have prediction costs that are proportional to the number of support vectors and these can grow linearly with the size of the training set. Thus, while non-linear SVMs have defined the state-of-the-art in terms of prediction accuracy on multiple benchmark tasks, their use in computationally intensive real world applications remains limited. The problem is getting exacerbated as large quantities of training data is becoming readily available in many domains.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

Our objective is to ameliorate the situation by reducing the cost of non-linear SVM prediction while maintaining classification accuracy above an acceptable threshold. Speeding up SVM prediction is an important research problem which has been approached from multiple perspectives including approximating the kernel function or matrix (Băzăvan et al., 2012; Kar & Karnick, 2012; Maji et al., 2013; Rahimi & Recht, 2007; 2008; Vedaldi & Zisserman, 2011; 2012; Williams & Seeger, 2001; Yang et al., 2012), reducing the number of support vectors (Cossalter et al., 2011; Joachims & Yu, 2009; Keerthi et al., 2006) and directly formulating SVM variants with low prediction costs (Ladicky & Torr, 2011).

We take a kernel learning based approach in this paper. The objective in kernel learning is to jointly learn both kernel and SVM parameters. In particular, Localized Multiple Kernel Learning (LMKL) (Gonen & Alpaydin, 2008) aims to learn a different kernel, and hence classifier, for each point in feature space (Ong et al., 2005; Tsang & Kwok, 2006). It has never been considered from the perspective of speeding up SVM prediction since it achieves only a modest reduction in the number of support vectors on average. In this paper, we generalize LMKL to learn arbitrary local feature embeddings that go beyond LMKL's non-negative gating functions. We learn local embeddings that are high dimensional, sparse and computationally deep [1]. This enables efficient prediction using primal variables and thereby decouples prediction costs from the num-

[1]The term deep typically refers to architectures where the cost of representation is logarithmic (Bengio et al., 2010). However, in this paper, our focus is primarily on computational cost and we seek computationally deep models where the cost of prediction is logarithmic. While bounding the generalization error of the proposed model is beyond the scope of this paper, note that we can perfectly represent highly varying patterns, such as the parity and the checker-board functions by models with logarithmic depth. Furthermore, unlike shallow axis-aligned decision trees, our proposed model can be appropriately regularized so as to generalize to regions of the feature space with no training points.

ber of support vectors and the size of the training set.

In our proposed Local Deep Kernel Learning (LDKL) formulation, a composite non-linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = K_L(\mathbf{x}_i, \mathbf{x}_j)K_G(\mathbf{x}_i, \mathbf{x}_j)$ is learnt as the product of a local kernel K_L and a global kernel K_G (note that the product of two Mercer kernels is also a Mercer kernel). This induces a high dimensional feature space embedding of the form $\boldsymbol{\phi}(\mathbf{x}) = \boldsymbol{\phi}_L(\mathbf{x}) \otimes \boldsymbol{\phi}_G(\mathbf{x})$ where $\boldsymbol{\phi}_L$ and $\boldsymbol{\phi}_G$ are the local and global feature embeddings induced by K_L and K_G respectively and $\otimes$ is the Kronecker product. We choose $\boldsymbol{\phi}_G$ to be low dimensional and $\boldsymbol{\phi}_L$ to be high dimensional and sparse so that classification can be carried out using primal variables as $y(\mathbf{x}) = sign(\mathbf{w}^t\boldsymbol{\phi}(\mathbf{x})) = sign(\boldsymbol{\phi}_L^t(\mathbf{x})W^t\boldsymbol{\phi}_G(\mathbf{x}))$ where the vector $\mathbf{w}$ has been reshaped to the matrix W and t denotes the transpose operator. To make prediction efficient, $\boldsymbol{\phi}_L$ is chosen to be tree-structured. Each dimension of $\boldsymbol{\phi}_L$ corresponds to a node in a tree and only those dimensions of $\boldsymbol{\phi}_L(\mathbf{x})$ are non-zero which correspond to the path traversed by $\mathbf{x}$ from the root to one of the leaves. Thus, for any $\mathbf{x}$, $\boldsymbol{\phi}_L(\mathbf{x})$ has only $\log M$ non-zero dimensions, which can be identified and computed in $O(\log M)$ time, where M is the dimensionality of $\boldsymbol{\phi}_L$. This results in an exponential speed up in prediction time with only a marginal reduction in classification accuracy in most cases. The learnt LDKL kernel also has a simple intuitive interpretation. It modulates an overall global similarity between pairs of points by a local similarity which is proportional to the length of the path shared by the two points in the learnt LDKL tree and which is maximal when the two points are in the same local region of space as defined by their common leaf node.

Most kernel learning formulations, including Localized Multiple Kernel Learning, are optimized using SVM dual parameters with the notable exception of (Orabona et al., 2010; Orabona & Jie, 2011). However, it is more attractive to optimize our LDKL formulation using primal stochastic sub-gradient descent since primal predictions are efficient in our case and we do not have to worry about maintaining dual variables or dual sparsity.

Note that optimizing over the space of trees is a hard non-convex problem. The loss incurred by a training point depends on its embedding which, in turn, depends on the path traversed by the point in $\boldsymbol{\phi}_L$. For instance, parameterizing node k by $\boldsymbol{\theta}_k$, $\boldsymbol{\phi}_{L_k}(\mathbf{x}_i)$ is proportional to $\prod_{l \in Ancestors(k)}(1 \pm sign(\boldsymbol{\theta}_l^t\mathbf{x}_i))/2$ which evaluates to unity only if node k lies on the path traversed by $\mathbf{x}_i$ and is zero otherwise. Modifying $\boldsymbol{\theta}_k$ changes the path traversed by each training point and this makes the objective function sharply discontinuous. In order to make tree learning amenable to sub-gradient descent, $\boldsymbol{\phi}_L$ is relaxed by replacing the signum function with the continuous tanh. However, this has the potential drawback that many entries in $\boldsymbol{\phi}_L$ might become non-zero and sparsity might be lost. We tackled this problem by introducing, and adaptively tuning, a scale parameter within the tanh functions during the optimization so as to make them tend to signum functions as convergence was approached. This ensured that, by and large, only a single path was dominant within the tree.

Experiments on benchmark data sets revealed that LDKL could significantly bring down prediction costs as compared to RBF-SVMs while maintaining classification accuracy above an acceptable threshold. The experiments also revealed that LDKL could yield better classification accuracies as compared to state-of-the-art methods for speeding up SVM prediction. For instance, on the challenging CoverType data set, an RBF-SVM took days to train and yielded a classification accuracy of 91.21% with a prediction cost that was 1.37×10^5 times that of a linear SVM. If prediction costs were restricted to 33 times that of a linear SVM, popular methods such as the Random Fourier Features (RFF) (Rahimi & Recht, 2007) yielded a classification accuracy of 58%. LDKL's accuracy was 88.21% for the same prediction cost. Thus LDKL could improve classification accuracy by 30% over RFF and could speed up RBF-SVMs by more than 4000 times with a tolerable loss in accuracy. Furthermore, LDKL took only hours to train and required less than 1 Mb of RAM to store its model parameters. This opens up the possibility of learning accurate non-linear SVMs on large data sets beyond the scope of RBF kernels.

Our main contribution in this paper is to formulate the problem from a local kernel learning perspective where we learn tree-structured features. Competing approaches to speeding up SVM prediction mainly follow the kernel approximation paradigm. These approaches suffer from the fact that the kernel is not approximated keeping the task and training set in mind. Thus, modelling power is wasted in learning good kernel approximations even far away from the decision boundary. On the other hand, learning a kernel which varies in feature space helps LDKL efficiently encode non-linearities into the classification model which is explicitly trained for the given task and training set. The focus is very much on learning the decision boundary as points that do not violate the margin play no role in the optimization. Furthermore, as the complexity of the learning task grows, LDKL can keep increasing the size of the embedding feature space while incurring only logarithmic prediction costs. This enables LDKL

to achieve an exponential speed up as compared to other leading methods. Our optimization ensures that training is efficient and that LDKL can scale to large data sets where the most benefits are to be gained. The LDKL source code can be downloaded from (Jose et al., 2013).

2. Related work

Low rank kernel approximation techniques have been specially developed for translation invariant kernels (Rahimi & Recht, 2007; 2008), dot product kernels (Kar & Karnick, 2012) and homogenous and additive kernels (Maji et al., 2013; Vedaldi & Zisserman, 2011). These techniques lead to a compact linear representation of the kernel taking $O(DM)$ time to embed D dimensional input features into an M dimensional space. The main limitation of these techniques is that they do not leverage knowledge about the task or training data set. Thus, modelling power is wasted in designing good kernel approximations even far away from the decision boundary. This is somewhat ameliorated by Nyström approximations (Vedaldi & Zisserman, 2012; Williams & Seeger, 2001; Yang et al., 2012) which are cognizant of the feature distribution and come at a higher prediction cost. However, since Nyström methods do not factor training labels into their approximation, they are still oblivious to the decision boundary. Finally, (Băzăvan et al., 2012) have proposed learning some of the parameters in translation invariant kernel approximations such as the RBF bandwidths in Random Fourier Features. While the type of embedding is fixed *a priori*, the parameters are learnt from training data and therefore improve performance over the base Random Fourier Features. In contrast, our proposed LDKL approach computes the embedding in $O(D \log M)$ time thereby gaining an exponential speed up over the kernel approximation techniques. Furthermore, the embedding is learnt from training data by directly minimizing the chosen loss and this can lead to a significant improvement in classification accuracy as shown in our experiments.

Methods designed to decrease the number of support vectors can potentially focus on the decision boundary. Post-processing based reduced set methods (Burges & Schölkopf, 1997; Cossalter et al., 2011) suffer from the limitation that the original set of support vectors needs to be determined before it can be reduced. This can become costly, and even infeasible for modern data sets, as non-linear SVM training typically scales quadratically in the number of training instances. This problem was addressed in (Keerthi et al., 2006; Tsang et al., 2005; 2007) where the number of support vectors was decreased during the training stage itself. This approach was extended in the Cutting-Plane Subspace Pursuit (CPSP) algorithm (Joachims & Yu, 2009) where support vectors were generated from outside the training set. Finally, LLSVM (Ladicky & Torr, 2011) directly formulates an SVM variant with low prediction cost. It can be interpreted as a special case of LDKL with a hand crafted, rather than learnt, local feature embedding without the tree structure. We empirically compare LDKL to LMKL, LLSVM and CPSP and demonstrate that LDKL can learn a significantly more accurate classifier for a given prediction cost.

Multiple kernel learning aims to learn the kernel from training data and many formulations, kernel parameterizations and regularizers have been proposed (Aflalo et al., 2011; Bach, 2008; Chapelle et al., 2002; Chen et al., 2008; Cortes et al., 2009a;b; Jain et al., 2012; Kloft et al., 2009; Lanckriet et al., 2004; Rakotomamonjy et al., 2008; Sindhwani & Lozano, 2011; Sonnenburg et al., 2006; Vishwanathan et al., 2010; Ye et al., 2008). Recent techniques, such as (Băzăvan et al., 2012), have applied standard convex MKL formulations to learn the RBF bandwidths in approximate Random Fourier Features (Rahimi & Recht, 2007). In contrast, LDKL does not aim to approximate any given kernel but focusses on learning the best decision boundary in a sparse, high dimensional representation.

The literature on learning trees is vast and includes methods for learning trees in feature space (decision trees and random forests), trees in label space (hierarchies, ontologies and structured output prediction) and trees for efficient retrieval (KD Trees, Ball Trees, *etc.*). Our approach should not be confused with these methods, nor with tree kernels which determine the similarity of two given trees rather than learn a tree structured feature embedding. Our problem setting and approach should also not be confused with kernel methods for deep learning as these do not focus on efficient SVM prediction. For instance, (Cho & Saul, 2009) design kernels which mimic the computation in multi-layer networks while (Vinyals et al., 2012) learn a deep classifier with layers of linear SVMs.

3. Localized Multiple Kernel Learning

The LMKL formulation (Gonen & Alpaydin, 2008) learns a prediction function of the form

$$y(\mathbf{x}) = sign(\sum_k p(\mathbf{w}_k|\mathbf{x})\mathbf{w}_k^t \boldsymbol{\phi}_k(\mathbf{x}) + b) \qquad (1)$$

where $p(\mathbf{w}_k|\mathbf{x})$ is a non-negative gating function which can be interpreted as the probability of picking classifier $\mathbf{w}_k$ for a given point $\mathbf{x}$. Thus, a different com-

bination of classifiers $\mathbf{w}_k$ and features $\boldsymbol{\phi}_k$ is selected for each $\mathbf{x}$. The gating functions are parameterized by $\Theta = \{(\boldsymbol{\theta}_k, \theta_{0k})\}$ as $p(\mathbf{w}_k|\mathbf{x}) = \frac{e^{\boldsymbol{\theta}_k^t\mathbf{x}+\theta_{0k}}}{\sum_m e^{\boldsymbol{\theta}_m^t\mathbf{x}+\theta_{0m}}}$. Kernel and SVM parameters are jointly optimized using the following non-convex formulation

$$\min_{\Theta}\max_{\boldsymbol{\alpha}} \quad \mathbf{1}^t\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{Y}K_\Theta\mathbf{Y}\boldsymbol{\alpha} \tag{2}$$

$$\text{subject to} \quad \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} = 0, \quad 0 \leq \boldsymbol{\alpha} \leq C \tag{3}$$

where $K_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \sum_k p(\mathbf{w}_k|\mathbf{x}_i)K_k(\mathbf{x}_i, \mathbf{x}_j)p(\mathbf{w}_k|\mathbf{x}_j)$. The optimization is carried out as a nested two stage procedure (Chapelle et al., 2002; Rakotomamonjy et al., 2008; Jain et al., 2012). In the outer loop, the kernel parameters are optimized using gradient descent while, in the inner loop, the kernel parameters are held fixed and the SVM parameters are learnt using a single kernel SVM training algorithm. The experiments in (Gonen & Alpaydin, 2008) demonstrated that for different K_k including linear, polynomial and RBF kernels, LMKL could learn a classifier that was competitive with a single kernel RBF-SVM but with slightly fewer support vectors resulting in a minor speed up over the RBF-SVM. It was also demonstrated that combining multiple linear kernels could improve classification accuracy over that of a linear SVM.

4. Local Deep Kernel Learning

The Local Deep Kernel Learning formulation learns a non-linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = K_L(\mathbf{x}_i, \mathbf{x}_j)K_G(\mathbf{x}_i, \mathbf{x}_j)$ as the product of a local kernel $K_L = \boldsymbol{\phi}_L^t\boldsymbol{\phi}_L$ and a global kernel $K_G = \boldsymbol{\phi}_G^t\boldsymbol{\phi}_G$ leading to the prediction function

$$\begin{aligned} y(\mathbf{x}) &= sign(\sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)) && (4)\\ &= sign(\sum_{ijk}(\alpha_i y_i \phi_{G_j}(\mathbf{x}_i)\phi_{G_j}(\mathbf{x})\phi_{L_k}(\mathbf{x}_i)\phi_{L_k}(\mathbf{x})) \\ &= sign(\mathbf{w}^t(\boldsymbol{\phi}_G(\mathbf{x}) \otimes \boldsymbol{\phi}_L(\mathbf{x}))) && (5)\\ &= sign(\boldsymbol{\phi}_L^t(\mathbf{x})W^t\boldsymbol{\phi}_G(\mathbf{x})) && (6)\\ &= sign(W^t(\mathbf{x})\boldsymbol{\phi}_G(\mathbf{x})) && (7) \end{aligned}$$

where $\mathbf{w}_k = \sum_i \alpha_i y_i \phi_{L_k}(\mathbf{x}_i)\boldsymbol{\phi}_G(\mathbf{x}_i)$, ϕ_{L_k} denotes dimension k of $\boldsymbol{\phi}_L \in \mathcal{R}^M$, $W = [\mathbf{w}_1 \dots \mathbf{w}_M]$, $W(\mathbf{x}) = W\boldsymbol{\phi}_L(\mathbf{x})$ and $\otimes$ is the Kronecker product. Thus, LDKL can be thought of as either learning a single fixed linear classifier in $\boldsymbol{\phi}_G \otimes \boldsymbol{\phi}_L$ space or a different classifier for each point in the given global feature space $\boldsymbol{\phi}_G$. Our prediction function in (6) is similar to LMKL's prediction function in (1) and generalizes it since our local features $\boldsymbol{\phi}_L$ can be arbitrary and are not restricted to being non-negative. Note that, if multiple heterogeneous global features needed to be combined as in LMKL, LDKL could easily be extended to learn $K = \sum_m K_{L_m}K_{G_m}$ and this could even give us the added power of learning a forest of local embeddings. However, in this paper, we focus on the problem of speeding up SVMs given a single global feature.

Primal based classification using W decouples prediction costs from the number of support vectors and the size of the training set. The global feature embedding should be chosen to be low dimensional, such as linear $\boldsymbol{\phi}_G(\mathbf{x}) = \mathbf{x} \in \mathcal{R}^D$ or quadratic $\boldsymbol{\phi}_G(\mathbf{x}) = \mathbf{x} \otimes \mathbf{x}$, in order to make prediction efficient. While the LDKL formulation is general, we report experimental results only for $\boldsymbol{\phi}_G(\mathbf{x}) = \mathbf{x}$ in this paper.

Having fixed the global features to be linear, non-linearities in LDKL are introduced through the local features. LDKL would reduce to LMKL if we set $\boldsymbol{\phi}_{L_k}(\mathbf{x}) = p(\mathbf{w}_k|\mathbf{x})$ and would further reduce to LLSVM (Ladicky & Torr, 2011) if the embedding was fixed and not learnt. Instead, $\boldsymbol{\phi}_L$ is chosen to be high dimensional, sparse and tree-structured. Each dimension k of $\boldsymbol{\phi}_L$ can be thought of as a node in a tree. For a given $\mathbf{x}$, only those dimensions of $\boldsymbol{\phi}_L(\mathbf{x})$ are non-zero for which the corresponding node lies on the path traversed by $\mathbf{x}$ from the root to the leaf. Thus, $\phi_{L_k}(\mathbf{x}) \propto I_k(\mathbf{x})$ where $I_k(\mathbf{x})$ is the indicator function which is unity only if node k lies on the path traversed by $\mathbf{x}$ and zero otherwise

$$I_k(\mathbf{x}) = \prod_{l \in Ancestors(k)} \tfrac{1}{2}(sign(\boldsymbol{\theta}_l^t\mathbf{x}) + (-1)^{C(l)}) \tag{8}$$

with $C(l)$ being zero if node l is its parent's left child and 1 if it is its parent's right child. Thus, if $\boldsymbol{\phi}_L$ has dimensionality M, then only $\log M$ dimensions will be non-zero and these can be identified in $\log M$ time.

Deep representations are typically obtained by function composition. A possible choice of ϕ_{L_k} is therefore,

$$\boldsymbol{\phi}_{L_k}(\mathbf{x}) = I_k(\mathbf{x})f_{k_0}(\mathbf{x}, f_{k_1}(\mathbf{x}, f_{k_2}(\cdots(f_{k_R}(\mathbf{x}, 1))))) \tag{9}$$

where k_r indexes the r^{th} ancestor of k and f is any smooth non-linear function. LDKL would generate a continuous, piecewise smooth decision boundary if f_{k_r} were a smooth function of $\boldsymbol{\theta}_{k_r}^t\mathbf{x}$ with $f_{k_r}(\mathbf{x}, z) = 0$ whenever $\boldsymbol{\theta}_{k_r}^t\mathbf{x} = 0$ – for instance, when $f_{k_r}(\mathbf{x}, z) = \tanh(z\boldsymbol{\theta}_{k_r}^t\mathbf{x})$ or for polynomials such as $f_{k_r}(\mathbf{x}, z) = (z\boldsymbol{\theta}_{k_r}^t\mathbf{x})^P$. This ensures that if $\boldsymbol{\phi}_{L_k}$ tends to zero then all its descendants would also tend to zero. Thus as the decision function transitions from one leaf node to another by switching paths at node k and crossing the leaf node cell boundary defined by $\boldsymbol{\theta}_k^t\mathbf{x} = 0$, the contributions of all the descendants of node k would smoothly diminish and the decision function would be determined by node k and its ancestors. LDKL could

be further generalized if f was no longer restricted to functions of $\boldsymbol{\theta}^t\mathbf{x}$ but could be any arbitrary function of x. While such a representation yielded much better classification accuracies than the state of the art, the best results were obtained by a non-compositional formulation with

$$\boldsymbol{\phi}_{L_k}(\mathbf{x}) = \tanh(\sigma\boldsymbol{\theta}'^t_k\mathbf{x})I_k(\mathbf{x}) \tag{10}$$

where the parameter set has been augmented with $\Theta' = [\boldsymbol{\theta}'_1, \ldots, \boldsymbol{\theta}'_M]$ and σ is a scaling parameter that could be set via validation.

LDKL's overall prediction time using (6) is therefore $O(D \log M)$ where D is the dimensionality of $\mathbf{x}$. This compares favourably, and leads to an exponential speed up, over all competing methods whose costs are at least $O(DM)$. The LDKL kernel $K = K_L K_G$ also has an intuitive interpretation. Two points are most similar if they belong to the same local region in space as defined by a given leaf node of $\boldsymbol{\phi}_L$. As one of the points moves away from the region to another leaf node, the similarity decreases depending on the number of common ancestors in $\boldsymbol{\phi}_L$. This local similarity is modulated by an overall global linear similarity.

The LDKL primal for jointly learning Θ and W from training data $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$ can be formulated as

$$\min_{W,\Theta.\Theta'} P(W, \Theta, \Theta') = \frac{\lambda_W}{2} Tr(W^tW) + \frac{\lambda_\Theta}{2} Tr(\Theta^t\Theta) + \frac{\lambda_{\Theta'}}{2} Tr(\Theta'^t\Theta') + \sum_{i=1}^{N} L(y_i, \boldsymbol{\phi}_L^t(\mathbf{x}_i)W^t\mathbf{x}_i) \tag{11}$$

Our focus in this paper is on the hinge loss for binary classification where $L = max(0, 1 - y_i\boldsymbol{\phi}_L^t(\mathbf{x}_i)W^t\mathbf{x}_i)$ though other appropriate loss functions for multi-class and multi-label classification, regression and novelty detection could also be plugged in. Note that LDKL does not include an explicit bias term b but instead adds a constant dimension to $\mathbf{x}$ whose value is determined through validation.

5. Optimizing LDKL

LDKL is optimized via primal stochastic sub-gradient descent. At iteration j, a training point $\mathbf{x}_i$ is picked at random and W and Θ are updated as

$$W^{j+1} = W^j - \eta_j \nabla_W P(W^j, \Theta^j, \Theta'^j, \mathbf{x}_i) \tag{12}$$

$$\Theta^{j+1} = \Theta^j - \eta_j \nabla_\Theta P(W^j, \Theta^j, \Theta'^j, \mathbf{x}_i) \tag{13}$$

$$\Theta'^{j+1} = \Theta'^j - \eta_j \nabla_{\Theta'} P(W^j, \Theta^j, \Theta'^j, \mathbf{x}_i) \tag{14}$$

where η_j is the step size at iteration j and

$$\nabla_{\mathbf{w}_k} P(\mathbf{x}_i) = \lambda_W \mathbf{w}_k - \delta_i y_i \phi_{L_k}(\mathbf{x}_i)\mathbf{x}_i \tag{15}$$

$$\nabla_{\boldsymbol{\theta}_k} P(\mathbf{x}_i) = \lambda_\Theta \boldsymbol{\theta}_k - \delta_i y_i \sum_l \tanh(\sigma\boldsymbol{\theta}'^t_l\mathbf{x}_i) \nabla_{\boldsymbol{\theta}_k} I_l(\mathbf{x}_i)\mathbf{w}_l^t\mathbf{x}_i \tag{16}$$

$$\nabla_{\boldsymbol{\theta}'_k} P(\mathbf{x}_i) = \lambda_\Theta \boldsymbol{\theta}'_k - \delta_i y_i \sigma(1 - \tanh^2(\sigma\boldsymbol{\theta}'^t_k\mathbf{x}_i)) I_k(\mathbf{x}_i)\mathbf{w}_k^t\mathbf{x}_i\mathbf{x}_i \tag{17}$$

where the gradient expressions have been derived for the hinge loss for binary classification, assuming ϕ_{L_k} is non-compositional as in (10) and δ_i is unity if $\mathbf{x}_i$ is a margin violator and zero otherwise.

Optimization over the space of trees is a hard non-convex problem. In this paper, we choose the tree structure to be fully balanced (though the tree structure could also have been learnt by introducing l_1 regularized node selection weights which ensure that all child node weights go to zero if a parent's weight goes to zero). The task then boils down to learning the indicator function I_k for each node k in the tree. Note that I_k is sharply discontinuous with respect to the tree parameters Θ. Thus, to make optimization via sub-gradient descent tractable, I_k is relaxed to

$$I_k(\mathbf{x}) = \prod_{l \in Ancestors(k)} \tfrac{1}{2}(\tanh(s_I\boldsymbol{\theta}_l^t\mathbf{x}) + (-1)^{C(l)})$$

$$\Rightarrow \nabla_{\boldsymbol{\theta}_l} I_k(\mathbf{x}) = I_k(\mathbf{x})(\tanh(s_I\boldsymbol{\theta}_l^t\mathbf{x}) + (-1)^{C(l)}) \delta_{l \in Ancestors(k)} s_I\mathbf{x} \tag{18}$$

However, this also implies that, for a given $\mathbf{x}$, $I_k(\mathbf{x})$ might be non-zero for many k. This destroys the sparsity in $\boldsymbol{\phi}_L$ and the cost of prediction no longer remains logarithmic. Furthermore, optimization remains hard, since the gradient needs to be propagated back from the root to multiple leaf nodes. We tackled this problem by introducing, and adaptively tuning, a scale parameter s_I within the tanh functions. Initially, s_I was set to a small value so as to ensure that most of the tanh relaxations were not saturated. As optimization progressed, s_I was adaptively increased so that $\tanh(s_i\boldsymbol{\theta}_k^t\mathbf{x})$ had tended to $sign(\boldsymbol{\theta}_k^t\mathbf{x})$ by the time convergence was approached. This was found to significantly speed up convergence and allowed us to efficiently scale to training sets with more than half a million data points.

Note that most MKL formulations, including LMKL, are optimized using SVM dual parameters. However, it is advantageous to optimize LDKL in the primal using stochastic sub-gradient descent for three reasons. First, primal based prediction using W is significantly more efficient in LDKL than dual based prediction using $\boldsymbol{\alpha}$. One could potentially consider dual co-ordinate ascent techniques which also maintain $\mathbf{W}$ such as (Hsieh et al., 2008), but then one runs into

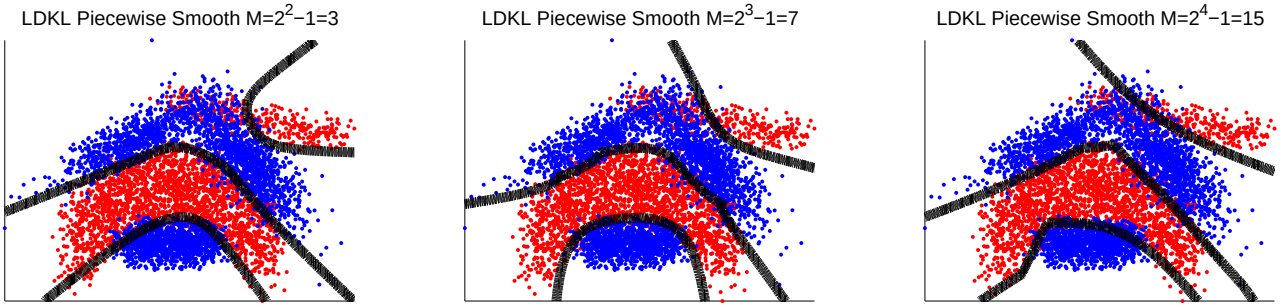

Figure 1. Piecewise smooth decision boundaries learnt by LDKL for $M = 2^2 - 1 = 3, 2^3 - 1 = 7$ and $2^4 - 1 = 15$ on the Banana data set. The first two decision boundaries are reasonable while $M = 15$ has started showing signs of over fitting.

Data Set	Num Train	Num Dims	Linear SVM	RBF-SVM		LDKL		
				Accuracy	Norm. Time	Accuracy	Norm. Time	Speed Up
CoverType	522,910	54	76.32	91.21	137,185.76	88.21 ± 0.15	32.54 ± 0.09	4216x
Letter	12,000	16	73.08	97.20	1,548.71	96.30 ± 0.42	33.29 ± 0.78	46x
MNIST	60,000	784	86.75	97.88	5,925.36	97.15 ± 0.06	17.55 ± 0.03	337x
CIFAR	50,000	400	68.45	81.58	28,529.87	76.54 ± 0.55	12.30 ± 0.02	2319x
Banana	1,000	2	55.49	90.21	322.12	88.67 ± 0.43	4.52 ± 0.05	71x
IJCNN	49,990	22	92.13	98.66	2,100.21	96.86 ± 0.07	13.24 ± 1.04	159x
USPS	7,291	256	83.35	96.86	1,415.72	95.18 ± 0.27	9.41 ± 0.03	150x
Magic04	14,226	10	79.08	86.88	4,493.89	85.74 ± 0.19	20.82 ± 0.49	215x

Table 1. LDKL can significantly speed up prediction time over an RBF-SVM (normalized time = prediction time / linear SVM prediction time). LDKL's average prediction cost, on 6 out of the 8 data sets, was 163 times lower than that of the RBF-SVM with a 1.3% loss in accuracy. On CoverType and CIFAR, the speed up was more than 2000 times but the loss was also higher at 3% and 5% respectively. Note that these data sets were found to be challenging for all methods.

the second problem. A co-ordinate or gradient step in Θ would change K_Θ and dual based techniques would no longer be able to take co-ordinate steps but would need to update all dual variables $\boldsymbol{\alpha}$ by solving the entire single kernel SVM for the new K_Θ. Even with warm restarts, this was found to be significantly more expensive than the primal update strategy of taking a sub-gradient step in$\mathbf{W}$ with respect to a single training point. Third, for multi-class problems, it is desirable to learn a single feature embedding for all classes using the multi-class hinge loss rather than a 1-vs-All formulation. Training time would rise to be quadratic in the number of categories for dual MKL methods. All in all, LDKL's primal based optimizer was found to be orders of magnitude faster than the LMKL optimizer and even state-of-the-art dual based optimizers employing spectral projected gradient descent (Jain et al., 2012). Finally, it should be noted that the vanilla convex MKL problem subject to $l_{p>1}$ regularization has been optimized in the primal (Orabona et al., 2010; Orabona & Jie, 2011) using stochastic sub-gradient descent. However, since LDKL learns an explicit feature map, it does not have to worry about maintaining dual variables or dual sparsity, and can therefore easily scale to large problems.

6. Experiments

The performance of LDKL was assessed on multiple benchmark data sets ranging from the small (Banana), where not much improvement is expected, to the large (CoverType) where the most gains are to be had. Table 1 lists the statistics of all the data sets used in the evaluation. Each data set comes with a predefined training and test set. The training set was further randomly partitioned into 80% for training and 20% for validation. Parameters for all algorithms were chosen so as to maximize classification accuracy on the validation set.

Two types of experiments were carried out. First, LDKL's performance was compared to that of an RBF-SVM in terms of classification accuracy and prediction cost. Second, LDKL's performance was compared to LMKL (Gonen & Alpaydin, 2008) and state-of-the-art methods such as CPSP (Joachims & Yu, 2009), LLSVM (Ladicky & Torr, 2011), Random Fourier Features (Rahimi & Recht, 2007) and the Nyström kernel

Data Set	LDKL	LMKL	RFF	Nyström	CPSP	LLSVM
CoverType	**88.21 ± 0.15**	-	58.94 ± 1.94	55.21 ± 1.49	71.06	78.77 ± 0.56
Letter	**96.30 ± 0.42**	-	70.90 ± 1.58	62.31 ± 1.72	82.80	87.43 ± 0.37
MNIST	**97.15 ± 0.06**	-	69.48 ± 1.66	56.08 ± 1.48	89.22	94.91 ± 0.14
CIFAR	**76.54 ± 0.55**	-	58.26 ± 1.52	58.52 ± 1.44	66.91	72.06 ± 0.20
Banana	**88.67 ± 0.43**	86.09	66.81 ± 2.43	56.13 ± 2.33	79.60	87.21 ± 0.13
IJCNN	**96.86 ± 0.07**	-	90.49 ± 2.59	90.49 ± 2.59	91.42	93.19 ± 0.72
USPS	**95.18 ± 0.27**	-	68.22 ± 2.31	66.21 ± 1.91	88.29	93.97 ± 0.23
Magic04	**85.74 ± 0.19**	-	72.13 ± 2.05	71.38 ± 2.05	83.73	85.20 ± 0.09

Table 2. For a fixed prediction cost, LDKL's classification accuracy can be significantly better than the state-of-the-art. LDKL can be 30% better than methods such as Random Fourier Features (RFF) and Nyström which do not train on the given labels and more than 10% better than methods such as CPSP and LLSVM that do.

approximation (Williams & Seeger, 2001; Yang et al., 2012). These methods are briefly reviewed in Section 2. For a fair comparison, the implementation as provided by the authors on their websites was used for all algorithms except for Nyström where the SciKit-Learn library was used (Pedregosa et al., 2011).

Both the compositional (9) and non-compositional (10) decision functions were tried out in LDKL and the latter was found to lead to better results. We therefore present results only for ϕ_{L_k} set as in (10). Figure 1 plots the decision boundaries for piecewise smooth LDKL for $M = 2^2 - 1, 2^3 - 1$ and $2^4 - 1$ on the Banana data set. The LDKL parameters λ_W, λ_Θ, $\lambda_{\Theta'}$ and σ were set by validation. LDKL's prediction time was controlled by the dimensionality of the local embedding space M and Figure 2 presents results for varying values of M for each data set. Note that the LDKL formulation is non-convex and mean and standard deviation values are reported for five random initializations.

Table 1 compares the performance of LDKL with linear and RBF-SVMs. Prediction costs are reported in terms of normalized time which is an algorithm's prediction time divided by a linear SVM's prediction time. LDKL's classification accuracy was found to be better than that of the linear SVM and tended to the RBF-SVM's accuracy in most cases. However, LDKL's prediction time could be orders of magnitude lower than that of the RBF-SVM. For instance, for 6 out of the 8 data sets, LDKL reduced the RBF-SVM's prediction cost by 163 times on average while suffering only a 1.3% loss in classification accuracy. On CoverType, the speed up was more than 4000 times with a 3% loss in accuracy. In contrast, popular methods such as Random Fourier Features have demonstrated only a thirty times speed up over the RBF-SVM using the given CoverType feature set (Rahimi & Recht, 2007) while achieving the same classification accuracy as LDKL. The CIFAR data set was a challenge for all methods. LDKL achieved a more than 2000 times speed up over the RBF-SVM but also resulted in a 5% loss in accuracy (though the loss incurred by other methods was even greater).

Table 2 compares LDKL's performance to LMKL and state-of-the-art methods for speeding up non-linear SVM prediction. Having fixed LDKL's prediction time as in Table 1, it tabulates the classification accuracies of all methods for the same prediction time. LDKL's accuracy was found to be as much as 30% higher than kernel approximation methods such as Random Fourier Features (RFF) and Nyström which do not learn from the given training labels. LDKL's accuracy could also be as much as 15% and 10% higher than CPSP's and LLSVM's accuracies respectively which do learn from the given training labels. Finally, LDKL was better than LMKL by 2.5% on even the small Banana data set. Unfortunately, the SVM dual based LMKL implementation provided by the authors was unable to scale to any other data set and hence no further comparisons are presented to LMKL. Figure 2 plots the classification accuracies of all methods for a range of prediction times. The scaling on the y-axis starts from the accuracy obtained by a linear SVM and hence if a method performed worse than a linear SVM then its curve would not be visible. As can be seen, LDKL's curve was consistently higher than all the other curves indicating that LDKL's classification accuracy was significantly better than the accuracies of other methods for a fixed prediction time.

7. Conclusions

We developed the Local Deep Kernel Learning formulation in this paper to speed up non-linear SVM prediction. We generalized the Localized Multiple Kernel Learning formulation (Gonen & Alpaydin, 2008) so as to learn arbitrary local feature embeddings. In particular, we learnt high dimensional, sparse and com-

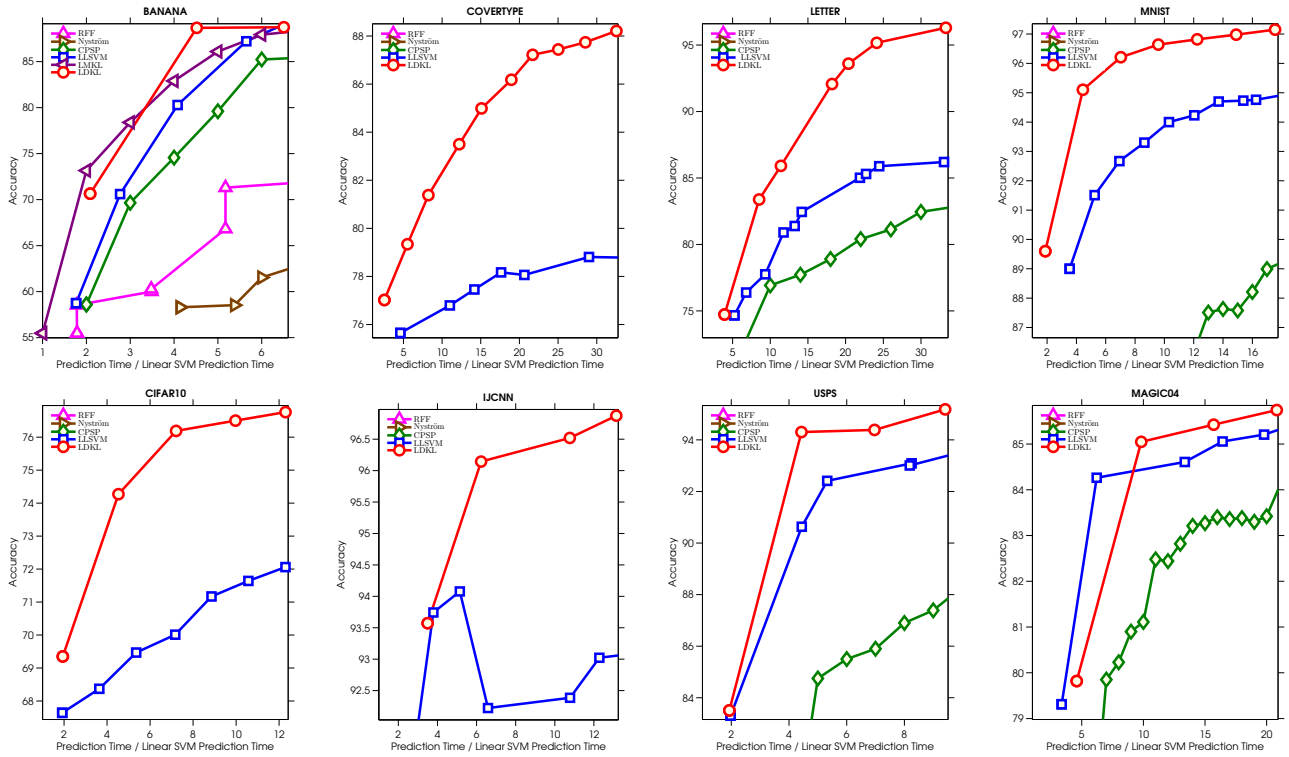

Figure 2. LDKL can have significantly higher classification accuracies as compared to state-of-the-art methods for a given prediction cost. A method's curve not appearing in the plot for a given data set indicates that, for the given prediction cost range, the methods performance was worse than that of a linear SVM. Figure best viewed magnified.

putationally deep local features. These introduced non-linearities into the LDKL model while ensuring that predictions could be made using primal variables in time that is logarithmic in the dimensionality of the local embedding space. We developed efficient primal based routines to optimize over the space of tree-structured local feature embeddings that scale to large training sets with more than half a million training points. Experimental results demonstrated that LDKL could achieve an exponential speed up over the RBF-SVM (more than four thousand times on CoverType) while maintaining classification accuracy over an acceptable threshold. Finally, we also demonstrated that LDKL could achieve significantly better classification accuracies as compared to LMKL and state-of-the-art methods. For instance, for a given prediction cost on CoverType, LDKL's classification accuracy was 30% higher than that of Random Fourier Features (Rahimi & Recht, 2007) and the Nyström approximation (Williams & Seeger, 2001; Yang et al., 2012), 15% higher than that of CPSP (Joachims & Yu, 2009) and almost 10% higher than that of LLSVM (Ladicky & Torr, 2011). LDKL took only hours to train on CoverType whereas an RBF-SVM took days. Furthermore, LDKL's parameters could be stored in less than 1 Mb of RAM. This opens up the possibility of training non-linear SVMs on large data sets beyond the scope of RBF and other traditional kernels and making accurate predictions efficiently using these learnt models.

Acknowledgments

We are grateful to Samy Bengio, Prateek Jain, Purushottam Kar, Yann Lecun, Vinod Nair, Yashoteja Prabhu and Nishal Shah for helpful discussions and feedback. Financial support was provided by the DST and the IRD,IIT-Delhi.

References

Aflalo, J., Ben-Tal, A., Bhattacharyya, C., Nath, J. Saketha, and Raman, S. Variable sparsity kernel learning. *JMLR*, 12:565–592, 2011.

Bach, F. R. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, pp. 105–112, 2008.

Bengio, Y., Delalleau, O., and Simard, C. Decision trees do not generalize to new variations. *Compt. Intl.*, 26, 2010.

Băzăvan, E. G., Li, F., and Sminchisescu, C. Fourier kernel learning. In *Proc. ECCV*, 2012.

Burges, C. J. C. and Schölkopf, B. Improving the accuracy and speed of support vector machines. In *NIPS*, 1997.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. Choosing multiple parameters for Support Vector Machines. *Machine Learning*, 46:131–159, 2002.

Chen, J., Ji, S., Ceran, B., Li, Q., Wu, M., and Ye, J. Learning subspace kernels for classification. In *KDD*, 2008.

Cho, Y. and Saul, L. Kernel methods for deep learning. In *NIPS*, 2009.

Cortes, C., Mohri, M., and Rostamizadeh, A. L2 regularization for learning kernels. In *UAI*, 2009a.

Cortes, C., Mohri, M., and Rostamizadeh, A. Learning non-linear combinations of kernels. In *NIPS*, 2009b.

Cossalter, M., Yan, R., and Zheng, L. Adaptive kernel approximation for large-scale non-linear svm prediction. In *ICML*, 2011.

Gonen, M. and Alpaydin, E. Localized multiple kernel learning. In *ICML*, 2008.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundarajan, S. A dual coordinate descent method for large-scale linear svm. In *ICML*, 2008.

Jain, A., Vishwanathan, S. V. N., and Varma, M. Spg-gmkl: Generalized multiple kernel learning with a million kernels. In *KDD*, 2012.

Joachims, T. and Yu, C.-N. J. Sparse kernel SVMs via cutting-plane training. *Machine Learning*, 76, 2009.

Jose, C., Goyal, P., Aggrwal, P., and Varma, M. The LDKL code. http://research.microsoft.com/en-us/um/people/manik/code/LDKL/download.html, 2013.

Kar, P. and Karnick, H. Random feature maps for dot product kernels. In *AISTATS*, 2012.

Keerthi, S. S., Chapelle, O., and DeCoste, D. Building support vector machines with reduced classifier complexity. *JMLR*, 7, 2006.

Kloft, M., Brefeld, U., Sonnenburg, S., Laskov, P., Muller, K.-R., and Zien, A. Efficient and accurate l_p-norm Multiple Kernel Learning. In *NIPS*, 2009.

Ladicky, L. and Torr, P. H. S. Locally linear support vector machines. In *ICML*, 2011.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M. I. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

Maji, S., Berg, A. C., and Malik, J. Efficient classification for additive kernel SVMs. *IEEE PAMI*, 35(1), 2013.

Ong, C. S., Smola, A. J., and Williamson, R. C. Learning the kernel with hyperkernels. *JMLR*, 6:1043–1071, 2005.

Orabona, F. and Jie, L. Ultra-fast optimization algorithm for sparse multi kernel learning. In *ICML*, June 2011.

Orabona, F., Jie, L., and Caputo, B. Online-batch strongly convex multi kernel learning. In *CVPR*, pp. 787–794, San Francisco, California, June 2010.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *JMLR*, 12, 2011.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *NIPS*, 2007.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, 2008.

Rakotomamonjy, A., Bach, F., Grandvalet, Y., and Canu, S. SimpleMKL. *JMLR*, 9:2491–2521, 2008.

Sindhwani, V. and Lozano, A. C. Non-parametric group orthogonal matching pursuit for sparse learning with multiple kernels. In *NIPS*, 2011.

Sonnenburg, S., Raetsch, G., Schaefer, C., and Schoelkopf, B. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

Tsang, I., Kwok, J. T., and Cheung, P. M. Core vector machines: Fast svm training on very large data sets. *JMLR*, 6, 2005.

Tsang, I., Kocsor, A., and Kwok, J. T. Simpler core vector machines with enclosing balls. In *ICML*, 2007.

Tsang, I. W. and Kwok, J. T. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

Vedaldi, A. and Zisserman, A. Efficient additive kernels via explicit feature maps. *IEEE PAMI*, 34(3), 2011.

Vedaldi, A. and Zisserman, A. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

Vinyals, O., Jia, Y., Deng, L., and Darrell, T. Learning with recursive perceptual representations. In *NIPS*, 2012.

Vishwanathan, S. V. N., Sun, Z., Theera-Ampornpunt, N., and Varma, M. Multiple kernel learning and the smo algorithm. In *NIPS*, 2010.

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *NIPS*, 2001.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. Nyström method vs random fourier features: A theoretical and empirical comparison. In *NIPS*, 2012.

Ye, J., , Ji, S., and Chen, J. Multi-class discriminant kernel learning via convex programming. *JMLR*, 9:719–758, 2008.