Local Deep Kernel Learning for Efficient Non-linear SVM Prediction

**Prasoon Goyal**  
Dept. Of Computer Science & Technology  
IIT Delhi

**Vinayak Agarwal**  
Dept. Of Electrical Engineering  
IIT Delhi

**Dr. Manik Varma**  
Dept. Of Computer Science & Technology  
IIT Delhi

---

**OBJECTIVE**  
Support Vector Machines (SVMs) are state-of-the-art classifiers in machine learning. However, the prediction cost of non-linear SVMs is very high. The objective of this project is to improve the prediction time of non-linear SVMs.

**MACHINE LEARNING**  
- What is machine learning?
- Applications of machine learning problems
  - Optical character recognition
  - Face detection
  - Spam filtering
- How do machines "learn"?
  A large class of machine learning algorithms take a set of training data, and identify patterns in the data. A new data point is now given to the machine, which classifies it into one of the several classes based on the identified patterns.
  For example, in optical character recognition, a machine is given a set of handwritten characters, along with the letter it represents. The machine is then given a handwritten character, and is expected to interpret the letter it represents.

**SUPPORT VECTOR MACHINES**  
SVMs are mathematical models to classify data. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

For nearly linearly separable data, SVM learns a hyperplane that maximises the distance from the two classes, known as the “margin”.

For non-linearly separable data, an SVM maps the data points in the given space to a higher dimensional space where the data becomes linearly separable.

**MATHEMATICAL FORMULATION OF SVM**  
Given a data point \( x \), an SVM classifies the point into class +1 or -1 as:

\[
g = \text{sign}(w^T x)
\]

where \( w \) is the weight vector learnt using the training data.

A better classifier: Use linear combination of weight vectors

\[
g = \text{sign}\left( \sum_{i=1}^{M} \gamma_i \theta_i w_i^T x \right)
\]

where the coefficients \( \gamma_i \)'s are functions of \( x \), defined in terms of parameters \( \theta_i \), also learnt by the classifier along with the weight vectors.

**OUR APPROACH: LOCAL DEEP KERNELE LEARNING**  
Main idea: Use sparse coefficients. This makes the computation of the decision function efficient.

**COMPUTING COEFFICIENTS**  
1) For a given test point \( x \), start at the root.
2) Compute \( \theta^T x \) for current node. If positive, go to left child. If negative, go to right child.
3) Continue step 2 till a leaf node is reached.
4) All \( \gamma_i \)'s on the path travelled are set to 1, while the others are set to 0.

**SPEED-UP ANALYSIS**  
The \( \gamma_i \)'s are defined such that only log(M) of them are non-zero for any point \( x \). This enables computing them efficiently, and also ensures that the number of non-zero terms in the decision function are only log(M), thereby giving an exponential speed-up.

**DECISION BOUNDARY**  
The figures below show the decision boundary learnt by our classifier. The blue and red points correspond to points of different classes, and the curve shown is the decision boundary learnt. Given a new data point, it is classified into blue or red class depending where it lies with respect to this decision boundary.

**LDKL SPEED-UP**  
The following table compares our classification accuracy and the prediction cost with the linear and RBF-SVMs on various benchmark datasets.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Num Train</th>
<th>Num Test</th>
<th>Linear SVM</th>
<th>Accuracy</th>
<th>Norm. Time</th>
<th>RBF-SVM</th>
<th>Accuracy</th>
<th>Norm. Time</th>
<th>LDKL</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoverType</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>LETTE</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>CoverType</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>LETTE</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>CoverType</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>LETTE</td>
<td>500</td>
<td>500</td>
<td>99.3</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>99.3</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
<td>0.01</td>
<td>99.3</td>
<td>0.02</td>
</tr>
</tbody>
</table>

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 9 data sets, was 64 times lower than that of the RBF-SVM with a 3.5% loss in accuracy. On CoverType and CIFAR, the speed up was more than 2000 times but the loss was only higher at 5% and 4% respectively. Note that these data sets were found to be challenging for all methods.

**COMPARISON WITH OTHER POPULAR METHODS**  
- SVMs, neural networks, decision trees, random forests, gradient boosting, etc.
- LDKL is faster than SVMs and neural networks.
- LDKL is more accurate than decision trees and random forests.
- LDKL is more scalable than gradient boosting.

---

**Prasoon Goyal**  
Dept. Of Computer Science & Technology  
IIT Delhi

**Vinayak Agarwal**  
Dept. Of Electrical Engineering  
IIT Delhi

**Dr. Manik Varma**  
Dept. Of Computer Science & Technology  
IIT Delhi