Local Deep Kernel Learning for Efficient Non-linear SVM Prediction

Suraj Jain
MSR India

Vinayak Agrawal
IIT Delhi

Cijo Jose
EPFL

Prasoon Goyal
IIT Delhi

Manik Varma
MSR India
Non-linear SVM Prediction

- Non-linear SVM prediction can be accurate
Non-linear SVM Prediction

- Cost of prediction = $O(DN)$
Linear SVM Prediction

- Cost of prediction = $O(D)$
Local Deep Kernel Learning Prediction

- Cost of prediction = $O(D \log N)$
Our Contributions

• Model
  • We learn a locally linear composite kernel
  • We learn a tree structured kernel for logarithmic prediction
  • Each node in the tree generates real-valued feature

• Optimization
  • We learn all the tree parameters jointly
  • We optimize efficiently using primal SGD
  • We scale to problems with millions of data points

• Prediction speedup and accuracy
  • 13,000 times faster than the RBF-SVM in some cases
  • Significantly higher accuracy over the state-of-the-art
Localized Multiple Kernel Learning

- Prediction function

\[ y(x) = \text{sign} \left( \sum_k p(w_k|x)w_k^T \phi_k(x) + b \right) \]

- Kernel function

\[ K_p(x_i, x_j) = \sum_k p(w_k|x_i)K_k(x_i, x_j)p(w_k|x_j) \]

- Dual optimization

\[ \min_p \max_\alpha \quad 1^T \alpha - \frac{1}{2} \alpha^T Y K_p Y \alpha \]

\[ \text{s. t. } 1^T Y \alpha = 0 \]

\[ 0 \leq \alpha \leq C \]
Non-linear SVM Prediction

• Non-linear SVM prediction

\[ y(x) = \text{sign} \left( \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) \right) \]

• RBF kernel: \( K(x, x_i) = e^{-\gamma ||x-x_i||_2^2} \)

• Cost of prediction = \( O(DN) \)
Composite Kernel Prediction

• Non-linear SVM prediction

\[ y(x) = \text{sign} \left( \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) \right) \]

• LDKL’s composite kernel

\[ K(x, x_i) = K_N(x, x_i) K_L(x, x_i) = \phi^t(x)\phi(x_i)x^tx_i \]

• LDKL’s prediction function

\[ y(x) = \text{sign} \left( \sum_{k=1}^{M} \phi_k(x)w_k^tx \right) \]
• LDKL’s prediction function: $y(x) = \text{sign}(\sum_{k=1}^{M} \phi_k(x) w_k^T x)$
Composite Kernel Prediction

• LDKL’s prediction function: \( y(x) = \text{sign}(\sum_{k=1}^{M} \phi_k(x)w_k^T x) \)
Composite Kernel Prediction

- LDKL’s prediction function: 
  \[ y(x) = \text{sign}(\sum_{k=1}^{M} \phi_k(x) w_k^T x) \]
Composite Kernel Prediction

- LDKL’s prediction function: $y(x) = \text{sign}(\sum_{k=1}^{M} \phi_k(x)w_k^t x)$
LDKL’s prediction function:

\[ y(x) = \text{sign}(\sum_{k=1}^{M} \phi_k(x)w_k^T x) \]

\[ \phi_k(x) = \tanh(\sigma \theta_k^T x) \]
A Shallow Architecture

\[ x \star w_1^t x \star w_2^t x \star \ldots \star w_M^t x \]
Learning Tree Structured Features

- LDKL’s prediction function

\[ y(x) = \text{sign} \left( \sum_{k=1}^{M} \phi_k(x)w_kx \right) \]
Learning Tree Structured Features

- LDKL’s prediction function
  
  \[ y(x) = \text{sign}(w_1^t x) \]
Learning Tree Structured Features

- LDKL’s prediction function

\[ y(x) = \text{sign}(w_1^T x + w_2^T x) \]
• LDKL’s prediction function

\[ y(x) = \text{sign}(w_1^T x + w_2^T x + w_4^T x) \]
Comparison to a Perceptron Tree
Comparison to a Decision Tree
Comparison to a Decision Tree
Learning Tree Structured Features

- LDKL’s prediction function

\[ y(x) = \text{sign} \left( \tanh(\sigma v_1^t x) w_1^t x \right) + \tanh(\sigma v_2^t x) w_2^t x + \tanh(\sigma v_4^t x) w_4^t x \]
LDKL’s Decision Boundaries
Training LDKL

- LDKL’s prediction function

\[ y(x) = \text{sign}\left( \tanh(\sigma v_1^T x) w_1^T x + \tanh(\sigma v_2^T x) w_2^T x + \tanh(\sigma v_4^T x) w_4^T x \right) \]
• Smoothing the tree
Training LDKL

- Primal optimization via stochastic sub-gradient descent

\[
\min_{V,W,\Theta} P = \frac{\lambda_w}{2} \text{Tr}(W^tW) + \frac{\lambda_v}{2} \text{Tr}(V^tV) + \frac{\lambda_\Theta}{2} \text{Tr}(\Theta^t\Theta) \\
+ \sum_i \max(0, 1 - y_i \Phi_L^t(x_i)W^tx_i)
\]

- Sub-gradients

\[
\nabla_{w_k} P(x_i) = \lambda_w w_k - \delta_i y_i \phi_{L_k}(x_i)x_i
\]
\[
\nabla_{\theta_k} P(x_i) = \lambda_{\Theta} \theta_k - \delta_i y_i \sum_l \tanh(\sigma v^t_l x_i) \nabla_{\theta_k} I_l(x_i)w^t_lx_i
\]
\[
\nabla_{v_k} P(x_i) = \lambda_v v_k - \delta_i y_i \sigma(1 - \tanh^2(\sigma v^t_k x_i))I_k(x_i)w^t_kx_ix_i
\]
Accuracy Comparison: RBF vs Linear
RBF-SVM: Cost of Prediction

Normalized Prediction Time = RBF Time / Linear Time
Training Time Comparison: RBF vs LDKL

- Training time on a single core of a 2.68 Ghz Xeon processor with 8 GB RAM.
Training Time Comparison: RBF vs LDKL

• Training time in minutes on a single core of a 2.68 Ghz Xeon processor with 8 GB RAM.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Linear SVM</th>
<th>RBF-SVM</th>
<th>LDKL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANANA</td>
<td>1.48E-05</td>
<td>0.006</td>
<td>0.01</td>
</tr>
<tr>
<td>CIFAR</td>
<td>6.76E-02</td>
<td>1283.68</td>
<td>0.278</td>
</tr>
<tr>
<td>CoverType</td>
<td>2.35</td>
<td>1369.96</td>
<td>7.990</td>
</tr>
<tr>
<td>IJCNN</td>
<td>3.93E-02</td>
<td>0.45</td>
<td>0.090</td>
</tr>
<tr>
<td>Letter</td>
<td>5.75E-04</td>
<td>0.43</td>
<td>2.200</td>
</tr>
<tr>
<td>Magic04</td>
<td>1.74E-03</td>
<td>0.17</td>
<td>0.047</td>
</tr>
<tr>
<td>MNIST</td>
<td>2.86E-01</td>
<td>39.12</td>
<td>1.376</td>
</tr>
<tr>
<td>USPS</td>
<td>5.97E-03</td>
<td>0.75</td>
<td>0.096</td>
</tr>
<tr>
<td>RCV1</td>
<td>0.13</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>MNIST8M</td>
<td>0.7</td>
<td>-</td>
<td>65.21</td>
</tr>
</tbody>
</table>
CoverType

COVERTYPE

Prediction Time / Linear SVM Prediction Time

Accuracy

LDKL
LLSVM
PPLLE
CPS
RFF
Nystrom
CIFAR10

![Graph showing the performance of different methods on CIFAR10 dataset.](image)

- **Prediction Time / Linear SVM Prediction Time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDKL</td>
<td>79</td>
</tr>
<tr>
<td>LLSVM</td>
<td>74</td>
</tr>
<tr>
<td>PPLE</td>
<td>59</td>
</tr>
<tr>
<td>CPSP</td>
<td>49</td>
</tr>
<tr>
<td>RFF</td>
<td>49</td>
</tr>
<tr>
<td>Nyström</td>
<td>49</td>
</tr>
</tbody>
</table>
MNIST

Accuracy vs Prediction Time / Linear SVM Prediction Time

- RFF
- Nyström
- LLSVM
- PPLE
- LDKL
- CPSP

Accuracy:
- LDKL
- LLSVM
- PPLE
- CPSP
- RFF
- Nyström

Prediction Time:
- 2
- 4
- 6
- 8
- 10
USPS

Prediction Time / Linear SVM Prediction Time

Accuracy

- RFF
- Nyström
- LLSVM
- PPLE
- CPSP
- LDKL
- CFSP

USPS

Accuracy

- LDKL
- LLSVM
- PPLE
- CPSP
- RFF
- Nyström

1 2 3 4 5 6 7 8

90 92 94 96 98 100

84 86 88 90 92 94
LDKL’s Decision Boundaries
The RBF-SVM’s Decision Boundaries
LDKL: Mimicking the RBF-SVM
## Pruning LDKL

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy (%)</th>
<th>Prediction Time (ms)</th>
<th># Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original Tree</td>
<td>Pruned Tree</td>
<td>Original Tree</td>
</tr>
<tr>
<td>Banana</td>
<td>89.53</td>
<td>89.50</td>
<td>1.08</td>
</tr>
<tr>
<td>CoverType</td>
<td>90.22</td>
<td>90.26</td>
<td>104.38</td>
</tr>
<tr>
<td>CIFAR</td>
<td>76.14</td>
<td>76.14</td>
<td>38.53</td>
</tr>
<tr>
<td>IJCNN</td>
<td>98.20</td>
<td>98.20</td>
<td>43.2</td>
</tr>
<tr>
<td>Letter</td>
<td>95.94</td>
<td>95.83</td>
<td>6.68</td>
</tr>
<tr>
<td>Magic</td>
<td>85.93</td>
<td>86.01</td>
<td>1.49</td>
</tr>
<tr>
<td>MNIST</td>
<td>97.27</td>
<td>97.26</td>
<td>128.43</td>
</tr>
<tr>
<td>USPS</td>
<td>95.51</td>
<td>95.46</td>
<td>6.95</td>
</tr>
</tbody>
</table>
Generating Training Data

- Sample points and have them labelled by the RBF-SVM
Training LDKL on the Extended Data Set

• Learn a balanced LDKL tree and then prune irrelevant nodes
Adding Two Nodes
Adding Three Nodes
Adding Four Nodes
Adding Five Nodes
Adding Six Nodes
Adding Seven Nodes
Adding Eight Nodes
Adding Nine Nodes
LDKL’s Final Decision Boundaries
Conclusions

• Publications and code
  • ICML 2013 paper
  • Code: http://research.microsoft.com/~manik/code/LDKL/download.html

• LDKL learns a local, deep composite kernel for efficient non-linear SVM prediction

• LDKL can be exponentially faster than the state-of-the-art

• Efficiency is important during both training and prediction
Acknowledgements

• Samy Bengio
• Purushottam Kar
• Prateek Jain
• Yann Lecun
• Vinod Nair
• Yashoteja Prabhu
• Nishal Shah