A Divide-and-Conquer Solver for Kernel Support Vector Machines

Cho-Jui Hsieh
Dept of Computer Science
UT Austin

ICML
Beijing, China
June 23, 2014

Joint work with S. Si and I. S. Dhillon
Support Vector Machines (SVM)

- SVM is a widely used classifier.

**Given:**
- Training data points $x_1, \ldots, x_n$.
- Each $x_i \in \mathbb{R}^d$ is a feature vector:
- Consider a simple case with two classes: $y_i \in \{+1, -1\}$.

**Goal:** Find a hyperplane to separate these two classes of data:
if $y_i = 1$, $w^T x_i \geq 1 - \xi_i$; $y_i = -1$, $w^T x_i \leq -1 + \xi_i$. 

![Diagram of SVM classification with hyperplanes separating two classes](image)
Support Vector Machines (SVM)

- What if the data is not linearly separable?

  \[ x \rightarrow \varphi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \]

  **Solution:** map data \( x_i \) to higher dimensional (maybe infinite) feature space \( \varphi(x_i) \), where they are linearly separable.

- **Kernel trick:** \( K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) \).

- Various types of kernels:
  - Gaussian kernel: \( K(x, y) = e^{-\gamma \|x - y\|^2} \);
  - Polynomial kernel: \( K(x, y) = (\gamma x^T y + c)^d \).
Support Vector Machines (SVM)

- The dual problem for SVM:

  \[
  \min_\alpha \frac{1}{2} \alpha^T Q \alpha - e^T \alpha, \\
  \text{s.t. } 0 \leq \alpha_i \leq C, \text{ for } i = 1, \ldots, n,
  \]

  where \( Q_{ij} = y_i y_j K(x_i, x_j) \) and \( e = [1, \ldots, 1]^T \).

- At optimum:

  \( w = \sum_i \alpha_i^* y_i \varphi(x_i) \),

  Prediction: \( w^T \varphi(\hat{x}) = \sum_i \alpha_i^* y_i K(x_i, \hat{x}) \).
Support Vector Machines (SVM)

- The dual problem for SVM:

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha,
\]

s.t. \( 0 \leq \alpha_i \leq C \), for \( i = 1, \ldots, n \),

where \( Q_{ij} = y_i y_j K(x_i, x_j) \) and \( e = [1, \ldots, 1]^T \).

- Challenge for solving kernel SVMs:
  - Space: \( O(n^2) \);
  - Time: \( O(n^3) \) (assume \( O(n) \) support vectors).

- \( n = \) Number of variables = number of samples.
Scalability

- **LIBSVM** takes more than 8 hours to train on a CoverType dataset with 0.5 million samples (with prediction accuracy 96%).
- Many **inexact** solvers have been developed: AESVM (Nadan et al., 2014), Budgeted SVM (Wang et al., 2012), Fastfood (Le et al., 2013), Cascade SVM (Graf et al., 2005), ... 1-3 hours, with prediction accuracy 85 – 90%.
- Divide the problem into smaller subproblems – **DC-SVM** 11 minutes, with prediction accuracy 96%.

![Scalability Chart]
DC-SVM with a single level – data division

- Partition $\alpha$ into $k$ subsets $\{V_1, \ldots, V_k\}$.
- Solve each subproblem independently:

$$
\min_{\alpha(i)} \frac{1}{2} (\alpha(i))^T Q(i,i) \alpha(i) - e^T \alpha(i),
$$

s.t. $0 \leq \alpha(i) \leq C$,

- Approximate solution for the whole problem:

$$
\tilde{\alpha} = [\tilde{\alpha}(1), \ldots, \tilde{\alpha}(k)].
$$

- Space complexity: $O(n^2) \rightarrow O(n^2/k^2)$.
- Time complexity: $O(n^3) \rightarrow O(n^3/k^2)$. 

Cho-Jui Hsieh Dept of Computer Science UT Austin 

Divide & Conquer SVM
DC-SVM with a single level – conquer step

- Use $\tilde{\alpha}$ to initialize a global coordinate descent solver.
- Converges quickly if $\|\tilde{\alpha} - \alpha^*\|$ is small.
- What clustering algorithm should we use to minimize $\|\tilde{\alpha} - \alpha^*\|$?
Quality of $\tilde{\alpha}$ (solution from subproblems)

- $\alpha^*$: solution of SVM with kernel $K$.
- $\tilde{\alpha}$: solution of SVM with

$$\tilde{K}(x, y) = I(\pi(x) = \pi(y))K(x, y),$$

where $\pi(\cdot)$ is the cluster indicator.

- The error comes from the between-cluster kernels:

$$D(\pi) = \sum_{i, j: \pi(x_i) \neq \pi(x_j)} |K(x_i, x_j)|.$$
Kernel kmeans clustering

- **Theorem 1**: For a given partition \( \pi \), the corresponding \( \bar{\alpha} \) satisfies

\[
0 \leq f(\bar{\alpha}) - f(\alpha^*) \leq (1/2)C^2D(\pi),
\]

and furthermore,

\[
\|\alpha^* - \bar{\alpha}\|_2^2 \leq C^2D(\pi)/\sigma_n,
\]

where \( \sigma_n \) is the smallest eigenvalue of the kernel matrix.

- Want a partition which
  1. Minimizes \( D(\pi) = \sum_{i,j: \pi(x_i) \neq \pi(x_j)} K(x_i, x_j) \).
  2. Have balanced cluster sizes (for efficient training).

- Use kernel kmeans (but slow).

- Two step kernel kmeans:
  - Run kernel kmeans on a subset of samples with size \( m \ll n \) to find cluster centers.
  - Identify the clusters for the rest of data.
Demonstration of the bound

- **Theorem 1:** For a given partition $\pi$, the corresponding $\bar{\alpha}$ satisfies
  
  \[ 0 \leq f(\bar{\alpha}) - f(\alpha^*) \leq \frac{1}{2}C^2D(\pi). \]

- Covertype dataset with 10000 samples and $\gamma = 32$ (best in cross validation).

- Our data partition scheme leads to a good approximation to the global solution $\alpha^*$.
DC-SVM with multiple levels

- Run DC-SVM with multiple levels.

Data Division

\{1, \ldots, n\}
Run DC-SVM with multiple levels.

Solve the leaf level problems.
Run DC-SVM with multiple levels.

\{1, \ldots, n\}

Solve the intermediate level problems.
Run DC-SVM with multiple levels.

{1,...,n}

Solve the original problem.
Early Prediction

- An **anytime algorithm** – stop at any level and give the prediction.
- Prediction using the $l$-th level solution
  faster training time; the prediction accuracy is close to or even better than the global SVM solution.

Naive way to predict $\hat{x}$: \[ \text{sign}(\sum_{i=1}^{n} y_i \bar{\alpha}_i K(x_i, \hat{x})) . \]
Prediction by $\tilde{K}$:
\[ \text{sign}(\sum_{i=1}^{n} y_i \bar{\alpha}_i \tilde{K}(x_i, \hat{x})) = \text{sign}(\sum_{i \in V_{\pi(\hat{x})}} y_i \bar{\alpha}_i K(x_i, \hat{x})) \]
Use nearest model to predict; better performance.

Prediction time reduced from $O(d(\#SV))$ to $O(d(\#SV)/k)$

|          | webspam $k = 50$ | webspam $k = 100$ | covtype $k = 50$ | covtype $k = 100$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction by $K$</td>
<td>92.6% / 1.3ms</td>
<td>89.5% / 1.3ms</td>
<td>94.6% / 2.6ms</td>
<td>92.7% / 2.6ms</td>
</tr>
<tr>
<td>Prediction by $\tilde{K}$</td>
<td><strong>99.1% / .17ms</strong></td>
<td><strong>99.0% / .16ms</strong></td>
<td><strong>96.1% / .4ms</strong></td>
<td><strong>96.0% / .2ms</strong></td>
</tr>
</tbody>
</table>

Cho-Jui Hsieh  Dept of Computer Science UT Austin  Divide & Conquer SVM
Toy Example

Two Circle Data: each circle is a class; not separable by kernel kmeans.

1st cluster

2nd cluster

3rd cluster

4th cluster

DC-SVM (early)

RBF SVM
Methods included in comparisons

- **DC-SVM**: proposed method for solving exact global SVM problem.
- **DC-SVM (EARLY)**: proposed method with early stopping (at 64 clusters).
- **LIBSVM** (Chang and Lin, 2011)
- **CASCADE SVM** (Graf et al., 2005)
- **FASTFOOD** (Le et al., 2013)
- **LaSVM** (Bordes et al., 2005)
- **LLSVM** (Zhang et al., 2012)
- **SpSVM** (Keerthi et al., 2006)
- **LTPU** (Moody and Darken., 1989)
- **BUDGETED SVM** (Wang et al., 2012; Djuric et al., 2013)
- **AESVM** (Nandan et al., 2014)
Results with Gaussian kernel.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>webspam</th>
<th>covtype</th>
<th>mnist8m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 2.8 \times 10^5$, $d = 254$</td>
<td>$n = 4.65 \times 10^5$, $d = 54$</td>
<td>$n = 8 \times 10^6$, $d = 784$</td>
</tr>
<tr>
<td></td>
<td>$C = 8, \gamma = 32$</td>
<td>$C = 32, \gamma = 32$</td>
<td>$C = 1, \gamma = 2^{-21}$</td>
</tr>
<tr>
<td><strong>DC-SVM (early)</strong></td>
<td><strong>670</strong></td>
<td><strong>672</strong></td>
<td><strong>10287</strong></td>
</tr>
<tr>
<td><strong>DC-SVM</strong></td>
<td><strong>10485</strong></td>
<td><strong>11414</strong></td>
<td><strong>71823</strong></td>
</tr>
<tr>
<td><strong>LIBSVM</strong></td>
<td><strong>29472</strong></td>
<td><strong>83631</strong></td>
<td><strong>298900</strong></td>
</tr>
<tr>
<td><strong>LIBSVM (subsample)</strong></td>
<td><strong>1267</strong></td>
<td><strong>5330</strong></td>
<td><strong>31526</strong></td>
</tr>
<tr>
<td><strong>LaSVM</strong></td>
<td><strong>20342</strong></td>
<td><strong>102603</strong></td>
<td><strong>171400</strong></td>
</tr>
<tr>
<td><strong>CascadeSVM</strong></td>
<td><strong>3515</strong></td>
<td><strong>5600</strong></td>
<td><strong>64151</strong></td>
</tr>
<tr>
<td><strong>LLSVM</strong></td>
<td><strong>2853</strong></td>
<td><strong>4451</strong></td>
<td><strong>65121</strong></td>
</tr>
<tr>
<td><strong>FastFood</strong></td>
<td><strong>5563</strong></td>
<td><strong>8550</strong></td>
<td><strong>14917</strong></td>
</tr>
<tr>
<td><strong>SpSVM</strong></td>
<td><strong>6235</strong></td>
<td><strong>15113</strong></td>
<td><strong>121563</strong></td>
</tr>
<tr>
<td><strong>LTPU</strong></td>
<td><strong>4005</strong></td>
<td><strong>11532</strong></td>
<td><strong>105210</strong></td>
</tr>
<tr>
<td><strong>Budgeted SVM</strong></td>
<td><strong>2194</strong></td>
<td><strong>3839</strong></td>
<td><strong>29266</strong></td>
</tr>
<tr>
<td><strong>AESVM</strong></td>
<td><strong>3027</strong></td>
<td><strong>3821</strong></td>
<td><strong>16239</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>time(s)</th>
<th>acc(%)</th>
<th>time(s)</th>
<th>acc(%)</th>
<th>time(s)</th>
<th>acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC-SVM (early)</td>
<td>670</td>
<td>99.13</td>
<td>672</td>
<td>96.12</td>
<td>10287</td>
<td>99.85</td>
</tr>
<tr>
<td>DC-SVM</td>
<td>10485</td>
<td>99.28</td>
<td>11414</td>
<td>96.15</td>
<td>71823</td>
<td>99.93</td>
</tr>
<tr>
<td>LIBSVM</td>
<td>29472</td>
<td>99.28</td>
<td>83631</td>
<td>96.15</td>
<td>298900</td>
<td>99.91</td>
</tr>
<tr>
<td>LIBSVM (subsample)</td>
<td>1267</td>
<td>98.52</td>
<td>5330</td>
<td>92.46</td>
<td>31526</td>
<td>98.95</td>
</tr>
<tr>
<td>LaSVM</td>
<td>20342</td>
<td>99.25</td>
<td>102603</td>
<td>94.39</td>
<td>171400</td>
<td>98.95</td>
</tr>
<tr>
<td>CascadeSVM</td>
<td>3515</td>
<td>98.1</td>
<td>5600</td>
<td>89.51</td>
<td>64151</td>
<td>98.3</td>
</tr>
<tr>
<td>LLSVM</td>
<td>2853</td>
<td>97.74</td>
<td>4451</td>
<td>84.21</td>
<td>65121</td>
<td>97.64</td>
</tr>
<tr>
<td>FastFood</td>
<td>5563</td>
<td>96.47</td>
<td>8550</td>
<td>80.1</td>
<td>14917</td>
<td>96.5</td>
</tr>
<tr>
<td>SpSVM</td>
<td>6235</td>
<td>95.3</td>
<td>15113</td>
<td>83.37</td>
<td>121563</td>
<td>96.3</td>
</tr>
<tr>
<td>LTPU</td>
<td>4005</td>
<td>96.12</td>
<td>11532</td>
<td>83.25</td>
<td>105210</td>
<td>97.82</td>
</tr>
<tr>
<td>Budgeted SVM</td>
<td>2194</td>
<td>98.94</td>
<td>3839</td>
<td>87.83</td>
<td>29266</td>
<td>98.8</td>
</tr>
<tr>
<td>AESVM</td>
<td>3027</td>
<td>98.90</td>
<td>3821</td>
<td>87.03</td>
<td>16239</td>
<td>96.6</td>
</tr>
</tbody>
</table>
Results with Gaussian kernel

 covtype objective function

 MNIST8m objective function

 covtype prediction accuracy

 MNIST8m prediction accuracy
The results for DC-SVM and LIBSVM coincide with each other because they solve the exact SVM problem.
We have proposed a novel divide-and-conquer algorithm for solving kernel SVM.

- Divide the problem into smaller subproblems.
- Solutions from subproblems are close to the original problem (when using kernel kmeans).
- Run DC-SVM with multiple levels to solve the original problem.
- Run DC-SVM with early prediction: yields competitive prediction accuracy 100 times faster than exact SVM solvers.

Software can be downloaded at
http://www.cs.utexas.edu/~cjhsieh/dcsvm
References


### Results with grid of $C, \gamma$

<table>
<thead>
<tr>
<th>dataset</th>
<th>$C$</th>
<th>$\gamma$</th>
<th>DC-SVM (early)</th>
<th>DC-SVM</th>
<th>LIBSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>acc(%)</td>
<td>time(s)</td>
<td>acc(%)</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-10}$</td>
<td>$2^{-10}$</td>
<td>86</td>
<td>806</td>
<td>61</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-10}$</td>
<td>$2^{-6}$</td>
<td>83</td>
<td>935</td>
<td>61</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-10}$</td>
<td>2</td>
<td>87.1</td>
<td>886</td>
<td>91.1</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-10}$</td>
<td>$2^6$</td>
<td>93.7</td>
<td>1060</td>
<td>92.6</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-10}$</td>
<td>$2^{10}$</td>
<td>98.3</td>
<td>1898</td>
<td>98.5</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-6}$</td>
<td>$2^{-10}$</td>
<td>83</td>
<td>793</td>
<td>68</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-6}$</td>
<td>$2^{-6}$</td>
<td>84</td>
<td>762</td>
<td>69</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-6}$</td>
<td>2</td>
<td>93.3</td>
<td>599</td>
<td>93.5</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-6}$</td>
<td>$2^6$</td>
<td>96.4</td>
<td>704</td>
<td>96.4</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^{-6}$</td>
<td>$2^{10}$</td>
<td>98.3</td>
<td>1277</td>
<td>98.6</td>
</tr>
<tr>
<td>webspam</td>
<td>2</td>
<td>$2^{-10}$</td>
<td>87</td>
<td>688</td>
<td>78</td>
</tr>
<tr>
<td>webspam</td>
<td>2</td>
<td>$2^{-6}$</td>
<td>93</td>
<td>645</td>
<td>81</td>
</tr>
<tr>
<td>webspam</td>
<td>2</td>
<td>2</td>
<td>98.4</td>
<td>420</td>
<td>99.0</td>
</tr>
<tr>
<td>webspam</td>
<td>2</td>
<td>$2^6$</td>
<td>98.9</td>
<td>466</td>
<td>98.9</td>
</tr>
<tr>
<td>webspam</td>
<td>2</td>
<td>$2^{10}$</td>
<td>98.3</td>
<td>853</td>
<td>98.7</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^6$</td>
<td>$2^{-10}$</td>
<td>93</td>
<td>759</td>
<td>80</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^6$</td>
<td>$2^{-6}$</td>
<td>97</td>
<td>602</td>
<td>83</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^6$</td>
<td>2</td>
<td>98.8</td>
<td>406</td>
<td>99.1</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^6$</td>
<td>$2^6$</td>
<td>99.0</td>
<td>465</td>
<td>98.9</td>
</tr>
<tr>
<td>webspam</td>
<td>$2^6$</td>
<td>$2^{10}$</td>
<td>98.3</td>
<td>917</td>
<td>98.7</td>
</tr>
</tbody>
</table>
Results for polynomial kernel $K(x_i, x_j) = (\eta + \gamma x_i^T x_j)^3$

webspam objective function  webspam prediction accuracy

covtype objective function  covtype prediction accuracy
Toy Example

Two Circle Data: each circle is a class; separable by kernel kmeans.

1st cluster

2nd cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: each circle is a class; separable by kernel kmeans.

1st cluster

DC-SVM (early)

2nd cluster

RBF SVM
Toy Example

Two Circle Data: not separable by kernel kmeans

1st cluster

2nd cluster

3rd cluster

4th cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: not separable by kernel kmeans

1st cluster

2nd cluster

3rd cluster

4th cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: separable by kernel kmeans; 10% noise.

1st cluster

2nd cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: separable by kernel K-means; 10% noise.

1st cluster

2nd cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: not separable by kernel kmeans; 10% noise.

1st cluster

2nd cluster

3rd cluster

4th cluster

DC-SVM (early)

RBF SVM
Toy Example

Two Circle Data: not separable by kernel kmeans; 10% noise.