Hash-Based SVM Approximation for Large Scale Prediction

Saloua Litayem, Alexis Joly, Nozha Boujemaa (INRIA)

Presented by Karen Simonyan (VGG)
Motivation

Large-scale multi-class classification (ImageNet)
• large number of classes (1K) and test samples (100K)
• slow evaluation and high storage requirements for one-vs-one classifiers (10M SVMs using 20 bootstraps)

Contribution – linear SVM hashing for
• fast evaluation
• low storage requirements
Note on State of the Art

• Winning ILSVRC 2011 submission [1] used one-vs-rest SVM
• Evaluation [2] shows that one-vs-rest is competitive:

<table>
<thead>
<tr>
<th></th>
<th>w-OVR</th>
<th>MUL</th>
<th>RNK</th>
<th>WAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td>BOV</td>
<td>26.4</td>
<td>22.7</td>
<td>20.8</td>
</tr>
<tr>
<td></td>
<td>FV</td>
<td>45.7</td>
<td>46.2</td>
<td>46.1</td>
</tr>
<tr>
<td>Top-5</td>
<td>BOV</td>
<td>46.4</td>
<td>38.4</td>
<td>41.2</td>
</tr>
<tr>
<td></td>
<td>FV</td>
<td>65.9</td>
<td>64.8</td>
<td>65.8</td>
</tr>
</tbody>
</table>

Accuracy on ILSVRC 2010


Locality Sensitive Hashing (LSH)

• Speed-up similarity search by projecting into low-dim Hamming space
  – search across the elements with the same hash key $F=[h_1, .., h_n]$
  – sub-linear complexity instead of linear

• The choice of hash functions $h_i$ depends on similarity
  – Locality-sensitive property: $Pr[h(x_i) = h(x_j)] = \text{sim}(x_i, x_j)$
  – Sign of a dot product with a random hyperplane is an LSH for

$$Pr[\text{sign}(x_i^T r) = \text{sign}(x_j^T r)] = 1 - \frac{1}{\pi} \cos^{-1}\left(\frac{x_i^T x_j}{\|x_i\|\|x_j\|}\right)$$

$r \sim \mathcal{N}(0, I)$

Hash function
Hash-Based SVM

Use LSH to speed-up the inner product in the linear SVM decision function:

\[ h(x) = sgn(\omega \cdot x + b) \]

\[ F_D(x) \] – hash of \( x \), \( \|x\| = 1 \)

\[ F_D(\omega) \] – hash of \( \omega \)

\[
\begin{cases}
\hat{h}(x) = sgn\left(r_{\omega,b} - d_H(F_D(x), F_D(\omega))\right) \\
r_{\omega,b} = \frac{D}{\pi} \cos^{-1}\left(\frac{-b}{\|\omega\|}\right)
\end{cases}
\]

\[
Pr[f(x) = f(\omega)] = 1 - \frac{1}{\pi} \cos^{-1}\left(\frac{x \cdot \omega}{\|x\| \|\omega\|}\right)
\]

Hamming distance is fast on modern CPUs

For 1000-D features and 256-bit LSH:
200 times less memory, 221 times faster
Convergence

Theorem - Convergence of a Hash based SVM classifier

The classification results of a hash based SVM classifier \( \hat{h}(x) \) converge to the results of the exact SVM classifier \( h(x) \) as the number \( D \) of binary hash functions tends to infinity.

ILSVRC 2010, 300 random classes, 1000-D BOW
Filter-and-Refine Strategy

• Hash-based multi-class SVM (HBMS):

\[
\hat{H}(x) = \arg \max_{c_k \in C} \# \left\{ \hat{h}_{k,j} \mid \hat{h}_{k,j}(x) = 1 \right\}
\]

• Filter-and-refine:
  – Obtain top-k candidate classes \( C_k(x) \) with HBMS \((k << K)\)
  – Exact one-vs-one for k classes:

\[
\hat{H}_k(x) = \arg \max_{c_m \in C_k} \# \left\{ h_{m,j} \mid \hat{h}_{m,j}(x) = 1 \right\}
\]

• Extension to bagging:
  – each binary classifier is an ensemble of SVMs
  – 20 bootstraps, 50 images/class
  – majority voting aggregation:

\[
p_{k,j}^* = \arg \max_{c_p \in \{c_k, c_j\}} \# \left\{ i \mid \hat{H}_i(x) = c_p \right\}
\]
Results: Filter-and-Refine vs Exact

Filter-and-Refine matches exact SVM with
- top-3 & 2048-bit LSH
- top-30 & 128-bit LSH

ILSVRC 2010, 300 random classes, 1000-D BOW
More Classes: Filter-and-Refine vs Exact

![Graph showing prediction time vs number of categories for different SVM methods.]

<table>
<thead>
<tr>
<th>Nb. categories</th>
<th>Exact SVM</th>
<th>Filter-and-Refine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>D = 256 bits</td>
</tr>
<tr>
<td>50</td>
<td>0.3208</td>
<td>0.3312</td>
</tr>
<tr>
<td>100</td>
<td>0.2314</td>
<td>0.2426</td>
</tr>
<tr>
<td>200</td>
<td>0.1597</td>
<td>0.1722</td>
</tr>
<tr>
<td>300</td>
<td>0.1341</td>
<td>0.1464</td>
</tr>
<tr>
<td>500</td>
<td>0.1069</td>
<td>0.1149</td>
</tr>
<tr>
<td>600</td>
<td>0.0922</td>
<td>0.1078</td>
</tr>
<tr>
<td>1000</td>
<td>0.0693</td>
<td>0.0724</td>
</tr>
</tbody>
</table>
Summary

• Scalable method for reducing memory footprint and computation time of linear SVMs

• Filter-and-refine HBMS with top-10% & 256-bit LSH is a good trade-off between speed and accuracy

• Practical applicability remains unclear – should be tested in state-of-the-art setup