ImageNet classification: fast descriptor coding and large-scale SVM training

Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu

LiangLiang Cao, Zhen Li, Min-Hsuan Tsai, Xi Zhou, Thomas Huang

Tong Zhang
Where we are in imageNet challenge

Our classification cost: 0.282 (top 5 hit rate, 71.8%, classification rate 52.9%)

Best performance of other teams: 0.336
System overview

Dense grid descriptor: HOG, LBP

Coding: local coordinate, super-vector

Pooling, SPM

Linear SVM

Fairly standard

Make good use of low level descriptors

How to train SVM efficiently
Outline

❖ Fast descriptor coding
  ▪ Local coordinate coding (LCC)
    K. Yu et.al, NIPS2009; J. Wang et. al, CVPR 2010
  ▪ super-vector coding
    X. Zhou et.al, ECCV2010

❖ Large-scale SVM classification
  ▪ Averaged stochastic gradient descent
What is local coordinate coding (LCC)

\[
X(d \times N) \approx B(d \times D) \times Z(D \times N)
\]

Assume \( B \) is given.

**Sparse coding:**

\[
z^* = \arg \min_z \frac{1}{2} \| x - Bz \|^2 + \lambda \sum_{i=1}^{D} |z_i|
\]

**LCC:** K. Yu et. al, NIPS 2009

\[
z^* = \arg \min_z \frac{1}{2} \| x - Bz \|^2 + \lambda \sum_{i=1}^{D} \| x - b_i \|^2 |z_i|
\]

Explicitly enforcing locality constraint
Why LCC
-- from functional approximation point of view

\[ f(x) \approx \sum_{i=1}^{D} z_i(x)w_i \]

e.g. nonlinear separating hyperplane

\[ |f(x) - \sum_{i=1}^{D} z_i(x)f(b_i)| \leq \alpha \|x - Bz(x)\| + \beta \sum_{i=1}^{D} \|x - b_i\|^2 |z_i(x)| \]

Coding error

Locality term

\[ \text{Functional approximation error} \]

Good approximation: 1) local to the test point x 2) small reconstruction error
Local coordinate coding -- fast implementation

J. Wang et. al, CVPR 2009

Step 1: be local to the test point $\mathbf{x}$ -- given $\mathbf{x}$, find its KNNs.

Step 2: small reconstruction error -- solve LMS fitting using only the KNNs

Approximated solutions, but significant speedup

For a regular image (7k patches), with $D=8192$:

- sparse coding needs $\sim 10\text{mins}$,
- (approximate) LCC needs only $\sim 2\text{s}$
Parallel computing

😃 For LCC, $D = 8,192$, each image takes ~2 seconds

$$2s \times 1,200,000 \approx 28 \text{ days}$$

*Not counting file I/O, networking delay, etc*

😢😢 In our submission, $D = 16,384$

which would have taken **more than 56 days**

😊 With Hadoop map-reduce (about ~100 mappers),
this was finished **within one day**.
System overview

Dense grid descriptor: SIFT, LBP

Coding: local coordinate, SV

Pooling, SPM

Each image is represented by a long vector

Linear SVM
Our training sets

<table>
<thead>
<tr>
<th>Sets</th>
<th>Coding scheme</th>
<th>Descriptor</th>
<th>Coding dimension</th>
<th>SPM</th>
<th>Feature dimension</th>
<th>Data set Size(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Local coordinate coding</td>
<td>HOG+LBP</td>
<td>8,192</td>
<td>10</td>
<td>81,920</td>
<td>167*</td>
</tr>
<tr>
<td>2</td>
<td>HOG</td>
<td>HOG</td>
<td>16,384</td>
<td>10</td>
<td>163,840</td>
<td>187*</td>
</tr>
<tr>
<td>3</td>
<td>HOG+LBP</td>
<td>HOG+LBP</td>
<td>20,480</td>
<td>10</td>
<td>204,800</td>
<td>260*</td>
</tr>
<tr>
<td>4</td>
<td>Super-vector coding</td>
<td>HOG</td>
<td>32,768</td>
<td>8</td>
<td>262,144</td>
<td>1374</td>
</tr>
<tr>
<td>5</td>
<td>HOG+LBP</td>
<td>HOG+LBP</td>
<td>51,200</td>
<td>4</td>
<td>204,800</td>
<td>1073</td>
</tr>
<tr>
<td>6</td>
<td>HOG</td>
<td>HOG</td>
<td>65,536</td>
<td>4</td>
<td>262,144</td>
<td>1374</td>
</tr>
</tbody>
</table>

*In sparse format

* Very high dimensional features, huge data sets
* LCC features have smaller size -- they are sparse
How monster is the resulting feature sets

Compare to PASCAL classification task:

<table>
<thead>
<tr>
<th></th>
<th># of training data</th>
<th># of class</th>
<th>(assumed) training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASCAL</td>
<td>10,103</td>
<td>20</td>
<td>1 hour</td>
</tr>
<tr>
<td>ImageNet</td>
<td>1,200,000</td>
<td>1000</td>
<td>6000 hours = 250 days*</td>
</tr>
<tr>
<td>Ratio</td>
<td>120</td>
<td>50</td>
<td>6000</td>
</tr>
</tbody>
</table>

* Not including file I/O, networking delay, etc

Life is short -- we need efficient SVM training algorithms
SVM using averaged stochastic gradient descent (ASGD)

One-against-all SVM classifier:

\[
L = \sum_{t=1}^{T} L(w, x_t, y_t) = \sum_{t=1}^{T} \frac{\lambda}{2} ||w||^2 + \max \left[0, 1 - y_t(w^T x_t + b)\right]
\]

Stochastic update:

\[
w^t = w^{t-1} - \eta \nabla L(w, x_t, y_t)
\]

\[
\bar{w}^t = (1 - 1/t)\bar{w}^{t-1} + w^t/t
\]

B. Polyak and A. Juditsky, 1992

😊 Memory efficient: only need to load data one-by-one

😊 Easy to parallelize: distribute the training of 1000 binary classifiers to different machines

😊 Fast convergence: need only a small number of epochs...
☺ Significant speed-up by averaging:
   5 epochs already give fairly good results.

☺ ASGD: has similar convergence properties as Stochastic Newton methods when appropriate stepsize is chosen

☺ Training time: LCC features, ~ 2 days (using two 8-core machines)
   Super-vector features, ~ 7 days (using three 8-core machines)
Conclusion

What’s the key:
1) learning: local coordinate coding and supervector coding + linear SVM
2) being able to handle large-scale data
   Best single method: ~65%
   Combined the 6 sets of features: 71.8%

Long way to go:
   Our method performs poorly on some categories...
Long way to go ...

- Better features: Hierarchical coding, discriminative coding
- More data
Thank you