Active learning for visual object recognition

Written by
Yotam Abramson and Yoav Freund
Presented by
Ben Laxton

Outline

• Motivation and procedure
• How this works: adaboost and feature details
• Why this works: boosting the margin for discriminative classifiers
• How well this works: results
Motivation

Observations:
- Machine learning methods are preferred for visual object detection.
- Most, if not all, require large amounts of hand labeled training data
- It is hard for people to identify ‘hard’ instances for training

Motivation

How much data is required?
- Most require on the order of thousands of labeled positive examples
- Viola-Jones used 4916 labeled faces for training

4916
Motivation

How much data is required?
- Most require on the order of thousands of labeled positive examples
- Viola-Jones used 4916 labeled faces for training

4916 * 10(sec) = 14 hours
Motivation

- ‘Hard examples provide more information than ‘easy’ ones
- How do we identify ‘hard’ examples?
  - Interaction between classifier and human operator.
Motivation

- ‘Hard examples provide more information than ‘easy’ ones
- How do we identify ‘hard’ examples?
  - Interaction between classifier and human operator.
- We will see that the notion of margin provides a nice framework for selecting these ‘hard’ examples.

Motivation

- ‘Hard examples provide more information than ‘easy’ ones
- How do we identify ‘hard’ examples?
  - Interaction between classifier and human operator.
- ‘Easy’ values to classify
- ‘Hard’ examples provide the most new information
Procedure

• SEVILLE (Semi automatic Visual Learning) provides an intuitive framework to achieve these goals
• Combines adaboost detectors (similar to Viola-Jones) with human interaction in an iterative fashion
• Can be used as a good labeling framework in general

Procedure

• First, play input images and manually collect a small number of positive and negative examples
• Train a classifier on these examples
Procedure

• Upon running the classifier, SEVILLE displays examples within the margins
• Label all of these examples and re-train a classifier

Procedure

• Continue this process on new frames.
• The resulting classifier should converge to optimal
Outline

• Motivation and procedure
• How this works: adaboost and feature details
• Why this works: boosting the margin for discriminative classifiers
• How well this works: results

SEVILLE Implementation

• Structured after the detector developed by Viola and Jones
• Uses Adaboost to boost weak classifiers
• Additional idea of intensity-invariant, fast, single pixel-based detectors
Adaboost

• Boosting methods produce accurate or ‘strong’ classifiers by combining many ‘weak’ classifiers in a principled framework.

• Adaboost is one algorithm for doing this:
  – Training error approaches 0 exponentially in the number of rounds.
  – Probabilistic upper bound for generalization error
  – Tends not to overfit

Adaboost

• The weak requirement: \[ \frac{\sum_{i: y_i \neq y} w_i}{\sum_{i=1}^{n} w_i} < \frac{1}{2} - \gamma \]

Slide Figure Borrowed from Yoav Freund’s Into to Boosting Presentation
Adaboost

- Intuitively Adaboost hones in on the ‘best’ simple features.
- This is important since there is a huge number of features:
  - For haar-like features used by Viola and Jones, with a window size of 24x24, there are 45,396 possibilities.

**Adaboost**

- Given example images \((x_1, y_1), \ldots, (x_n, y_n)\) where \(y_j = 0, 1\) for negative and positive examples respectively.
- Initialize weights \(w_{1,i} = \frac{1}{2^m}, \frac{1}{2^l}\) for training example \(i\), where \(m\) and \(l\) are the number of negatives and positives respectively.

For \(t = 1 \ldots T\)
- 1) Normalize weights so that \(w_t\) is a distribution
- 2) For each feature \(j\) train a classifier \(h_j\) and evaluate its error \(\varepsilon_j\) with respect to \(w_t\)
- 3) Chose the classifier \(h_j\) with lowest error.
- 4) Update weights according to:
  \[ w_{t+1,i} = w_t \beta_t^{1 - \varepsilon_i} \]
  where \(\varepsilon_i = 0\) is \(x_i\) is classified correctly, 1 otherwise, and
  \[ \beta_t = \frac{1}{1 - \varepsilon_t} \]
- The final strong classifier is:
  \[ h(x) = \begin{cases} 1 & \sum_{j=1}^{t} \alpha_j h_j(x) \geq \frac{1}{2} \sum_{j=1}^{t} \alpha_j, \\ 0 & \text{otherwise} \end{cases} \]
  \[ \alpha_j = \log \left( \frac{1}{\beta_t} \right) \]
Viola-Jones

- The work by Viola and Jones extends the basic adaboost classifier in the following ways:
  - Rediscovery of the integral image representation for fast feature calculations
  - Use of haar-like feature basis as weak classifiers
  - Introduction of a cascaded classifier

Viola-Jones: Integral image

Def: The integral image at location \((x,y)\), is the sum of the pixel values above and to the left of \((x,y)\), inclusive.

Using the following two recurrences, where \(i(x,y)\) is the pixel value of original image at the given location and \(s(x,y)\) is the cumulative column sum, we can calculate the integral image representation of the image in a single pass.

\[
s(x,y) = s(x,y-1) + i(x,y)
\]
\[
i(x,y) = ii(x-1,y) + s(x,y)
\]

Taken from Gyozo Gidofalvi’s presentation on Viola-Jones
Viola-Jones: Simple features

Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example the integral sum inside rectangle D we can compute as:

\[ \text{ii}(4) + \text{ii}(1) - \text{ii}(2) - \text{ii}(3) \]

As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

Taken from Gyozo Gidofalvi’s presentation on Viola-Jones

---

Viola-Jones: Simple Features

- Resulting simple features offer some insight into what the classifier is looking at

First discriminative feature

Second discriminative feature
Viola-Jones: Cascaded Classifier

- Using a cascaded classifier can improve the running time substantially
  - Discard ‘easier’ examples early on and focus attention on ‘harder’ ones.
  - Takes the form of a degenerate decision tree:

Adaboost: Cascaded Classifier

- Detection and false-positive rates can now be broken down on a per-level basis:
  - e.g. To get a 10-stage cascaded classifier with a detection rate of 90% and false positive rate of 1/1000000
Adaboost: Cascaded Classifier

• Detection and false-positive rates can now be broken down on a per-level basis:
  – e.g. To get a 10-stage cascaded classifier with a detection rate of 90% and false positive rate of 1/1000000
    • Each stage must have a detection rate of .99 since .9~.99^{10}
  • But notice that the false positive rate for each level can be as high as 30% since .3^{10} ~ 6*10^{-6}. 
SEVILLE: YEF features

- SEVILLE substitutes the integral image and haar-like features with YEF (yet even faster) features.
- Observe that haar-like features require:
  - image intensity normalization
  - ~8 array accesses per feature on integral image

- Rather than comparing sums of pixels, YEF features are based on the statistical examination of single pixels.
- Each feature consists of two subsets of points, if all points in one subset have lower intensity than every point in the other, the feature outputs 1, otherwise 0.
- Results in ~4 image accesses per feature
SEVILLE: YEF Features

- Note there are \( \sim 10^{32} \) possible features
- A genetic algorithm is used to evolve good features:
  - Start with a first generation of 100 features
  - Discard the 90 features with the highest error
  - Create 50 features by slightly mutating the remaining 10
  - Add an additional 40 random features
  - Repeat this process until no performance improvement is seen for 40 generations

SEVILLE: YEF Features

- Notice that performance depends on \( K \), the number of points in the feature

<table>
<thead>
<tr>
<th>value of ( K )</th>
<th>Average # of operations</th>
<th>False positive rate at 90% detection rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.266</td>
<td>1.465,000</td>
</tr>
<tr>
<td>3</td>
<td>3.624</td>
<td>1.95,000</td>
</tr>
<tr>
<td>4</td>
<td>3.781</td>
<td>1.98,000</td>
</tr>
<tr>
<td>5</td>
<td>4.044</td>
<td>1.105,000</td>
</tr>
<tr>
<td>6</td>
<td>4.061</td>
<td>1.132,000</td>
</tr>
<tr>
<td>7</td>
<td>4.112</td>
<td>1.110,000</td>
</tr>
<tr>
<td>8</td>
<td>4.149</td>
<td>1.115,000</td>
</tr>
</tbody>
</table>
Outline

- Motivation and procedure
- How this works: adaboost and feature details
- Why this works: boosting the margin for discriminative classifiers
- How well this works: results

Theory: Adaboost

- Bounding the training error:
  - Consider binary decision problems (chance is 1/2)
  - Given a weak classifier $h_t$, $\epsilon_t = 1/2 - \gamma_t$
  - Freund and Schapire prove that the training error of the final strong classifier is:

  $$\text{Error}(H) \leq \exp\left(-2\sum \gamma_i^2\right)$$
Theory: Adaboost

- Bounding the generalization error:
  - First shot:
    \[ \text{Error}_{\text{generalization}} \leq \text{Pr}[H(x) \neq y] + O\left(\frac{\sqrt{\text{Td}}}{m}\right) \]

- Note that this suggests that the method will overfit as the number of trials, T, becomes large

---

Theory: Adaboost

- However, empirical finding show that over-fitting does not often happen in practice:

- This suggests an improved upper bound based on the margins…
Theory: Adaboost

- The margin of an example \((x,y)\) is defined as:
  \[
  \text{margin} = \frac{y \sum_i a_i h_i(x)}{\sum_i a_i}
  \]

- Schapire et al. proved that larger margins on the training set yield better generalization:
  \[
  \text{error}_{\text{generalization}} \leq \Pr[\text{margin}(x,y) \leq \Theta] + O\left(\frac{d}{m\Theta^2}\right)
  \]

Theory: SEVILLE

- SEVILLE makes the leap of mainly choosing training examples from the margin
- Note that in the generalization error bound, this amounts to simultaneously increasing \(m\) and \(\Pr[\text{margin}(x,y) < \Theta]\)
  - Note that Abramson makes the argument that the ‘theoretical \(m\)’ is much larger than the actual value since all values far from the margin are implicitly added to the set.
Outline

- Motivation and procedure
- How this works: adaboost and feature details
- Why this works: boosting the margin for discriminative classifiers
- How well this works: results

SEVILLE: Results

- The SEVILLE system was tested on a pedestrian detection task.
- 215 pedestrian and 37,064,791 non-pedestrian images were used for validation
- Videos used for training were segments clipped from 6 hours of footage taken from a car driving around Paris
SEVILLE: Results

<table>
<thead>
<tr>
<th>Step</th>
<th>positive examples</th>
<th>negative examples</th>
<th>Feature type</th>
<th>Weak rules</th>
<th>$\mu^{-}$</th>
<th>$\mu^{+}$</th>
<th>data length</th>
<th>human labor</th>
<th>training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>10</td>
<td>CP</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0.96</td>
<td>3m</td>
<td>2s</td>
</tr>
<tr>
<td>2</td>
<td>36</td>
<td>134</td>
<td>CP</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td>0.98</td>
<td>3m</td>
<td>8s</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
<td>526</td>
<td>CP</td>
<td>7</td>
<td>0.6</td>
<td>1</td>
<td>0.16</td>
<td>10m</td>
<td>36s</td>
</tr>
<tr>
<td>4</td>
<td>86</td>
<td>1312</td>
<td>CP</td>
<td>30</td>
<td>0.4</td>
<td>1</td>
<td>0.41</td>
<td>20m</td>
<td>2s</td>
</tr>
<tr>
<td>5</td>
<td>182</td>
<td>2675</td>
<td>CP</td>
<td>59</td>
<td>0.1</td>
<td>0.8</td>
<td>0.31</td>
<td>30m</td>
<td>10m</td>
</tr>
<tr>
<td>6</td>
<td>417</td>
<td>7864</td>
<td>CP</td>
<td>270</td>
<td>0</td>
<td>0.6</td>
<td>1.27</td>
<td>2h</td>
<td>1h20m</td>
</tr>
<tr>
<td>7</td>
<td>848</td>
<td>8236</td>
<td>CP</td>
<td>893</td>
<td>-0.02</td>
<td>0.5</td>
<td>0.23</td>
<td>5h</td>
<td>9h</td>
</tr>
<tr>
<td>8</td>
<td>1178</td>
<td>16613</td>
<td>CP</td>
<td>1500</td>
<td>-0.02</td>
<td>0.5</td>
<td>1.42</td>
<td>8h</td>
<td>28h</td>
</tr>
<tr>
<td>9</td>
<td>1486</td>
<td>19238</td>
<td>CP</td>
<td>2014</td>
<td>-0.02</td>
<td>0.5</td>
<td>9.46</td>
<td>15h</td>
<td>58h</td>
</tr>
<tr>
<td>10</td>
<td>2046</td>
<td>22533</td>
<td>CP</td>
<td>3150</td>
<td>-0.02</td>
<td>0.5</td>
<td>12.29</td>
<td>22h</td>
<td>88h</td>
</tr>
</tbody>
</table>

SEVILLE: Results
SEVILLE: Results

• Some observations:
  – Doesn’t achieve target False-Positive rate
    • May be because features are too simple
  – Doesn’t use a cascaded classifier in practice
  – Could be extended to allow for selection of arbitrary image regions
  – Note: This is like co-training with a human as the second classifier

References