How to make it better?

- By extracting good features.
- By using more powerful classification algorithms.
- By transforming feature points to a new space.
- By using features and classifiers in different ways.
- By using prior and contextual information.
- By using more data.

Outline

- A conceptual understanding of boosting
- Mathematical Details
- Application in computer vision

Introduction

- It is hard to build a classifier which generalizes well and accurately.
- Meanwhile, it is easy to find weak classifiers that can perform better than chance e.g. Label an input image as “apple” if it has a big red blob.
- Can we combine several week classifiers to build a better one?
  Three humble shoemakers brainstorming will make a great statesman.
How to combine classifiers?

- A similar idea: combine classifiers \( h_1(x), \ldots, h_m(x) \)
  \[ H(x) = \alpha_1 h_1(x) + \ldots + \alpha_m h_m(x). \]

- \( \alpha_j \) is the vote assigned to classifier \( h_j \).
  - Votes should be higher for more reliable classifiers.

- Prediction:
  \[ \hat{y}(x) = \text{sign } H(x). \]

- Classifiers \( h_j \) can be simple (e.g., based on a single feature).

Basic Ideas

- Pick classifiers one at each round
- Maintain a distribution of weights over the training examples
  Examples that have not been classified correctly at previous iterations get larger weights.
- All weights are equal initially.
- The weight of misclassified examples is increased, forcing the successive weak classifier to focus on the hard examples in the training set that previously selected classifiers cannot handle.
Combine multiple classifiers

- Different learners may
  - be trained by different algorithms
  - Use different features
  - Focus on different subproblems.
- Key Problems:
  - How to train individual classifiers?
  - How to combine?
Ada Boosting (by Freund and Schapire, 1997)

**Initialization step:** for each example $x$, set $D(x) = \frac{1}{N}$, where $N$ is the number of examples

**Iteration step** (for $t = 1..T$):
1. Find best weak classifier $h_t(x)$ using weights $D_t(x)$
2. Compute the error rate $\varepsilon_t$ as $\varepsilon_t = \frac{N_{\text{error}}}{N}$, where $N_{\text{error}} = \sum_{x \in \mathcal{X}} D_t(x)\cdot \mathbf{1}[y \neq h_t(x)]$
3. Assign weight $\alpha_t$ to classifier $h_t$ in the final hypothesis $\alpha_t = \log \left( \frac{1}{\varepsilon_t} \right)$
4. For each $x$, $D_t(x) = D_t(x) \cdot \exp(-\alpha_t \cdot \mathbf{1}[y \neq h_t(x)])$
5. Normalize $D_t(x)$ so that $\sum_{x} D_t(x) = 1$

$f_{\text{final}}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

An example

- Examples of high weight are going to be more often in face/nonface classification example.

**Round 1**

<table>
<thead>
<tr>
<th>$\checkmark$</th>
<th>$\checkmark$</th>
<th>$\checkmark$</th>
<th>$\checkmark$</th>
<th>$\checkmark$</th>
<th>$\checkmark$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{16}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{16}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{16}$</td>
<td>$\frac{1}{4}$</td>
</tr>
</tbody>
</table>

Best classifier: $\checkmark$

Change Weights:

| $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{16}$ | $\frac{1}{4}$ | $\frac{1}{16}$ | $\frac{1}{4}$ |

* Give to the classifier the following re-sampled examples:

Assign weight $\alpha_t$ to classifier $h_t$ in the final hypothesis $\alpha_t = \log \left( \frac{1}{\varepsilon_t} \right)$

Example from previous slide:

$\varepsilon_t = \frac{5}{16}$ \implies \alpha_t = \log \left( \frac{16}{5} \right) = \log 3.2 \approx 0.8$

- Recall that $\varepsilon_t \leq \frac{1}{2}$
- Thus $(1 - \varepsilon_t) \geq 1 \implies \alpha_t > 0$
- The smaller is $\varepsilon_t$, the larger is $\alpha_t$ and thus the more important (weight) classifier $h_t(x)$ gets in the final classifier $f_{\text{final}}(x) = \text{sign} \left( \sum \alpha_t h_t(x) \right)$
AdaBoost Example (from Freund and Schapire)

For each $x_i$, $D(x_i) = D(x_i) \cdot \exp(-y_i f(x_i)/||f||)$

Example from previous slide: $x_i = 0.8$

- Weight of misclassified examples is increased and the new $D(x_i)$'s are normalized to be a distribution again

Normalize $D(x_i)$ so that $\sum D(x_i) = 1$

Example from previous slide:

- After normalization:

| 0.05 | 0.18 | 0.05 | 0.18 | 0.05 | 0.18 |

ROUND 1

$f_1(x) = \text{sign}(f_1 - 3)$
Generalization and overfitting

- We are interested in the generalization properties on the testing data, not the training error.
- AdaBoost has excellent generalization properties.
- It can be shown that boosting increases the margins of training examples as iterations proceed.
Overfitting

Occam’s Razor: The simple function explaining most of the data is preferable to a complex one fitting the data very well.

Pros

• Fast
• Simple
• Only one parameter T
• Flexible: can be combined with any classifier.
• The only requirement: weak learner, better than chance.

Cons

• AdaBoost may not work if
  – a weak learner is too complex (overfitting).
• It is sensitive to noise (e.g. wrong labels). It may over-emphasize noisy examples in later training.

Robust Real-Time Face Detection
(Viola & Jones, 2004)

• New representation
• Using AdaBoost for improved learning
• Cascaded detectors

• The combination of these ideas yields the fastest known face detector for gray scale images.
Classifier is Learned from Labeled Data

- Training Data
  - 5000 faces
    - All frontal
  - $10^6$ non faces
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose (rotation both in plane and out)

What are features?

- Three kinds of features are being used.

Huge library of filters

- Size 24x24, features 180,000
- Rectangle features are somewhat primitive.
- However, they do provide a rich presentation.

Efficient implementation

- Integral Image
  \[ i(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \]
  - Any rectangular sum can be computed in constant time.

\[ D = 1 + 1 - (2+3) = 1 + (1 + 2) - (1 + 0) = 2 \]
Feature Selection

- Features = Weak Classifiers
- Each round selects the optimal feature given:
  - Previous selected features
- In each round for each remaining feature sum of weighted errors is evaluated for all examples
  - The classifier with lowest error is selected
  - Reweight the examples misclassified by lowest error classifier
  - Finally build a weighted sum of classifiers

The first and second features

- A classifier with 200 rectangle features was learned using AdaBoost.
  - 95% correct detection on test set with 1 in 10034 false positives.

The Attentional Cascade

- Increases detection performance while reducing the computation time
  - A very simple classifier can be built that rejects many negative windows while keeping all positive ones
  - We could define a computational risk hierarchy a nested set of classifier classes
  - Initial simple classifiers minimize false positives for later complicated classifiers

Observation

- Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.
- The threshold of a boosted classifier can (and should, at least for the firsts in the cascade) be adjusted so that the false negative rate is close to zero! (not to throw too much)
The Cascade Classifier

- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative)

A Real-time Face Detection System

Training faces: 4916 face images (24 x 24 pixels) plus vertical flips for a total of 9832 faces.

Training non-faces: 250 million sub-windows from 5500 non-face images.

Final detector: 38 layer cascaded classifier
The number of features per layer was 1, 10, 25, 25, 50, 50, 50, 75, 100, ..., 200, ...

Final classifier contains 8061 features.

Examples
Summary

• Very fast: 15 fps with 384x288 pixel image
• Three contributions with broad application
  – Cascaded classifier can be used to speed up object detection.
  – AdaBoost as feature selector
  – Rectangle features are easy to compute and can yield rapid image analysis.
• Try it? Intel OpenCV