Classifier Case Study: Viola-Jones Face Detector

P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004.

Most of these slides are from Svetlana Lazebnik
http://www.cs.unc.edu/~lazebnik/spring09/

Face detection

• Basic idea: slide a window across image and evaluate a face model at every location

Challenges of face detection

• Sliding window detector must evaluate tens of thousands of location/scale combinations
• Faces are rare: 0–10 per image
  • For computational efficiency, we should try to spend as little time as possible on the non-face windows
  • A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  • To avoid having a false positive in every image image, our false positive rate has to be less than 10^-6

The Viola/Jones Face Detector

• A seminal approach to real-time object detection
• Training is slow, but detection is very fast
• Key ideas:
  • Haar-like image features
  • Integral images for fast feature evaluation
  • Boosting for feature selection
  • Attentional cascade for fast rejection of non-face windows

Image Features

Haar-like filters
Rectangular regions consisting of +1, -1, coefficients
Similar to Haar wavelets

Value = \sum (pixels in white area) − \sum (pixels in black area)

Example

When does the filter have high response?

Source images

\[\times\]
Image Features

- Appropriate combinations of added and subtracted rectangles approximate various derivative filters, e.g., \(dx\), \(dy\), \(dx\ dy\), \(dx^2\), at different locations and scale.

Fast computation with integral images

- The integral image computes a value at each pixel \((x,y)\) that is the sum of the pixel values above and to the left of \((x,y)\), inclusive.
- This can quickly be computed in one pass through the image.

Origins of integral images

Note idea of cumulative distributions in probability theory

\[
D(x) = P(X \leq x) = \int_{-\infty}^{x} P(\xi) d\xi, \\
\text{Then } P(a \leq X \leq b) = D(b) - D(a) = \int_{a}^{b} P(\xi) d\xi,
\]

Similarly, a multivariate distribution function can be defined if outcomes depend on \(n\) parameters:

\[
D(x_1, \ldots, x_n) = P(X_1 = x_1, \ldots, X_n = x_n),
\]

The probability content of a closed region can be found much more efficiently than by direct integration

Probability density function \(P(\xi)\) by appropriate evaluation of the distribution function at all possible \(x\) defined on the region (Ross and Smith 1996, 2002, p. 193). For example, for a bivariate distribution for \(D(x, y)\) the probability content in the region \(a \leq x \leq b, y \leq y_1\) \(y \leq y_2\) is given by

\[
P(a \leq x \leq b, y_1 \leq y \leq y_2) = \int_{a}^{b} \int_{y_1}^{y_2} P(\xi, \psi) d\psi dx,
\]

but can be computed much more efficiently using

\[
P(a \leq x \leq b, y_1 \leq y \leq y_2) = D(b, y_2) - D(b, y_1) - D(a, y_2) + D(a, y_1)
\]

Computing the integral image

- Can be computed by a single pass through the image

Cumulative row sum: \(s(x, y) = s(x-1, y) + i(x, y)\)

Integral image: \(ii(x, y) = ii(x, y-1) + s(x, y)\)

MATLAB: \(ii = \text{cumsum}(\text{cumsum}(\text{double}(i)), 2);\)

Computing sum within a rectangle

- Let \(A, B, C, D\) be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

\[
\text{Sum} = A - B - C + D
\]

- Only 3 additions are required for any size of rectangle!

Example

- Integral Image
  - Sum within a rectangle
Feature selection

• For a 24x24 detection region, the number of Haar features considered is ~160,000!

At test time, it is impractical to evaluate the entire feature set

Can we create a good classifier using just a small subset of all possible features?

How to select such a subset?

Boosting

• Boosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier
  • A weak learner need only do better than chance
• Training consists of multiple boosting rounds
  • During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  • "Hardness" is captured by weights attached to training examples

Training procedure

• Initially, weight each training example equally
• In each boosting round:
  • Find the weak learner that achieves the lowest weighted training error
  • Raise the weights of training examples misclassified by current weak learner
• Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
• Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)


Boosting illustration

Increasing the weights forces subsequent classifiers to focus on residual errors
Boosting illustration

Weights Increased

Final classifier is a weighted combination of the weak classifiers

Final classifier is a weighted combination of the weak classifiers

Boosting

Boosting is a type of greedy method for minimizing average loss over a training set by adding new features/classifiers one at a time.

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement

- Disadvantages
  - Needs many training examples
  - Often doesn’t work as well as SVM (especially for many-class problems)

Boosting for face detection

• Weak learners here are defined based on thresholded Haar features

\[ h_t(x) = \begin{cases} 
+1 & \text{if } p_t f_t(x) > p_t \theta_t \\
-1 & \text{otherwise} 
\end{cases} \]

Note: the parity value just serves to change the direction of the threshold to be either less that or greater than, as appropriate.
For each round of boosting:
- Evaluate each rectangle filter on each example
- Select best threshold for each filter
- Select best filter/threshold combination as weak learner
- Reweight examples

Computational complexity of learning: \(O(MN^2K)\)
- \(M\) rounds,
- \(N\) examples,
- \(K\) features

Boosting for face detection
- First two features selected by boosting:

This feature combination can yield 100% detection rate and 50% false positive rate

Not good enough! (FP rate too high)

Recall that to avoid having a false positive in every image, our false positive rate has to be less than \(10^{-6}\)

Attentional cascade
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

Attentional cascade
- Cascading classifiers solves several problems:
  - Improves speed by early rejection of nonface regions by simple classifiers
  - Reduces false positive rates

Attentional cascade
- Chain classifiers that are progressively more complex and have lower false positive rates:
Attentional cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of 0.9 and a false positive rate on the order of $10^{-6}$ can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about $0.30$ ($0.3^{10} \approx 6 \times 10^{-6}$)

Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

The implemented system

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9000 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

System performance

- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Output of Face Detector on Test Images

Other detection tasks

Facial Feature Localization
Profile Detection
Male vs. female
Profile Detection

Profile Features

Summary: Viola/Jones detector
- Haar features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows