Previously

- Instance recognition
  - Local features: detection and description
  - Local feature matching, scalable indexing
  - Spatial verification
- Intro to generic object recognition
- Supervised classification
  - Main idea
  - Skin color detection example

Last time: supervised classification

Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

So, best decision boundary is at point \( x \) where

\[
P(\text{class} \mid 9 \mid x)L(9 \rightarrow 4) = P(\text{class} \mid 4 \mid x)L(4 \rightarrow 9)
\]

To classify a new point, choose class with lowest expected loss; i.e., choose “four” if

\[
P(4 \mid x)L(4 \rightarrow 9) > P(9 \mid x)L(9 \rightarrow 4)
\]

Last time: Example: skin color classification

We can represent a class-conditional density using a histogram (a “non-parametric” distribution)

Now for every pixel in a new image, we can estimate probability that it is generated by skin.

Classify pixels based on these probabilities
- if \( p(\text{skin} \mid x) > \theta \), classify as skin
- if \( p(\text{skin} \mid x) < \theta \), classify as not skin
Today

- Window-based generic object detection
  - basic pipeline
  - boosting classifiers
  - face detection as case study

Generic category recognition: basic framework

- Build/train object model
  - Choose a representation
    - Learn or fit parameters of model / classifier
  - Generate candidates in new image
  - Score the candidates

Generic category recognition: representation choice

Window-based Part-based

Window-based models

Building an object model

- Pixel-based representations sensitive to small shifts

- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

Window-based models

Building an object model

- Consider edges, contours, and (oriented) intensity gradients

Simple holistic descriptions of image content
  - grayscale / color histogram
  - vector of pixel intensities
Window-based models
Building an object model
• Consider edges, contours, and (oriented) intensity gradients
• Summarize local distribution of gradients with histogram
  > Locally orderless: offers invariance to small shifts and rotations
  > Contrast-normalization: try to correct for variable illumination

Discriminative classifier construction
Nearest neighbor
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Support Vector Machines
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,...

Neural networks
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998

Boosting
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,...

Conditional Random Fields
McCallum, Freitag, Pereira 2000;
Kumar, Hebert 2003
...

Generic category recognition:
basic framework
• Build/train object model
  – Choose a representation
  – Learn or fit parameters of model / classifier
• Generate candidates in new image
• Score the candidates

Window-based models
Generating and scoring candidates

Window-based object detection: recap
Training:
1. Obtain training data
2. Define features
3. Define classifier

Given new image:
1. Slide window
2. Score by classifier

Car/non-car Classifier
Discriminative classifier construction

- Nearest neighbor
  - Shakmanovich, Viola, Darrell 2003
  - Berg, Berg, Malik 2005
- Neural networks
  - LeCun, Bottou, Bengio, Haffner 1998
  - Rowley, Baluja, Kanade 1998
- Support Vector Machines
  - Guyon, Vapnik
  - Heisele, Serre, Poggio, 2001
- Boosting
  - Viola, Jones 2001
  - Torralba et al. 2004
  - Opelt et al. 2006
- Conditional Random Fields
  - McCallum, Freitag, Pereira 2000
  - Kumar, Hebert 2003

Boosting intuition

- Weak Classifier 1
- Weak Classifier 2
- Weak Classifier 3

Boosting illustration

- Weights Increased

Boosting illustration

- Weights Increased
Boosting illustration

Final classifier is a combination of weak classifiers

Boosting: training

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest weighted training error
  - Raise 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: pros and cons

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear 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 found not to work as well as an alternative discriminative classifier, support vector machine (SVM)
    - especially for many-class problems

Viola-Jones face detector

Main idea:
- Represent local texture with efficiently computable “rectangular” features within window of interest
- Select discriminative features to be weak classifiers
- Use boosted combination of them as final classifier
- Form a cascade of such classifiers, rejecting clear negatives quickly

Viola-Jones detector: features

“Rectangular” filters
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time.
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!

Viola-Jones detector: features

- • Considering all possible filter parameters: position, scale, and type:
  - 180,000+ possible features associated with each 24 x 24 window
- Which subset of these features should we use to determine if a window has a face?
- Use AdaBoost both to select the informative features and to form the classifier

Viola-Jones detector: AdaBoost

- • Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.
- Outputs of a possible rectangle feature on faces and non-faces.
- Resulting weak classifier:
  \[ b_t(x) = \begin{cases} +1 & \text{if } f_t(x) > 0 \\ -1 & \text{otherwise} \end{cases} \]
- For next round, reweight the examples according to errors, choose another filter/threshold combo.

AdaBoost Algorithm

1. Start with uniform weights on training examples.
2. For T rounds:
   - Evaluate weighted error for each feature, pick best.
   - Re-weight the examples:
     - Incorrectly classified \( \to \) more weight
     - Correctly classified \( \to \) less weight
   - Final classifier is combination of the weak ones, weighted according to error they had.

Viola-Jones Face Detector: Results

- First two features selected
• Even if the filters are fast to compute, each new image has a lot of possible windows to search.

• How to make the detection more efficient?

Cascading classifiers for detection

- Form a cascade with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Viola-Jones detector: summary

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


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

Viola-Jones Face Detector: Results
Viola-Jones Face Detector: Results

Detecting profile faces?

Can we use the same detector?

Viola-Jones Face Detector: Results

Example using Viola-Jones detector

Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles.


Consumer application: iPhoto 2009

http://www.apple.com/ilife/iphoto/
Consumer application: iPhoto 2009

**Things iPhoto thinks are faces**

 Slide credit: Lana Lazebnik

**Can be trained to recognize pets!**

[Image of cookies and a cat]


**What other categories are amenable to window-based representation?**

**Pedestrian detection**

- Detecting upright, walking humans also possible using sliding window’s appearance/texture; e.g.,
  - SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]
  - Space-time rectangle features [Viola, Jones & Snow, ICCV 2001]
  - SVM with HoGs [Dalal & Triggs, CVPR 2005]

**Window-based detection: strengths**

- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

**Window-based detection: Limitations**

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
  - With so many windows, false positive rate better be low
Limitations (continued)

- Not all objects are “box” shaped

Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions

Limitations (continued)

- If considering windows in isolation, context is lost

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

Summary

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Boosting classifiers: general idea
- Viola-Jones face detector
  - Exemplar of basic paradigm
  - Plus key ideas: rectangular features, Adaboost for feature selection, cascade
- Pros and cons of window-based detection
• Given example images \((x_1, y_1), \ldots, (x_n, y_n)\) where 
  \(y_i = 0, 1\) for negative and positive examples respectively.
• Initialize weights \(w_{1,i} = \frac{1}{2m}, \frac{1}{2l}\) for \(y_i = 0, 1\) respectively, where \(m\) and \(l\) are the number of negatives and positives respectively.
• For \(t = 1, \ldots, T\):
  1. Normalize the weights,
     \[ w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \]
     so that \(w_t\) is a probability distribution.
  2. For each feature, \(j\), train a classifier \(h_j\) which is restricted to using a single feature. The error is evaluated with respect to \(w_t\),
     \[ e_j = \sum_i w_i \left| h_j(x_i) - y_i \right| \]
  3. Choose the classifier, \(h_t\), with the lowest error \(e_t\).
  4. Update the weights:
     \[ w_{t+1,i} = w_{t,i} \beta^{-e_t}_t \]
     where \(\epsilon_t = 0\) if example \(x_i\) is classified correctly, \(\epsilon_t = 1\) otherwise, and \(\beta_t = \frac{1}{1-\epsilon_t}\).
• The final strong classifier is:
   \[ h(x) = \begin{cases} 
   1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{T}{2} \sum_{t=1}^{T} \alpha_t \\
   0 & \text{otherwise}
   \end{cases} \]
   where \(\alpha_t = \log \frac{1}{\beta_t}\).

Table 1: The AdaBoost algorithm for classifier learning. Each round of boosting selects one feature from the 180,000 potential features.

number of features are retained (perhaps a few hundred or thousand).

3.2. Learning Results

While details on the training and performance of the final system are presented in Section 5, several simple results merit discussion. Initial experiments demonstrated that a frontal face classifier constructed from 200 features yields a detection rate of 95% with a false positive rate of 1 in 14084. These results are compelling, but not sufficient for many real-world tasks. In terms of computation, this classifier is probably faster than any other published system, requiring 0.7 seconds to scan an 384 by 288 pixel image. Unfortunately, the most straightforward technique for improving detection performance, adding features to the classifier, directly increases computation time.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks (see Figure 3). This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

Figure 3: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

4. The Attentional Cascade

This section describes an algorithm for constructing a cascade of classifiers which achieves increased detection performance while radically reducing computation time. The key insight is that smaller, and therefore more efficient, boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances (i.e. the threshold of a boosted classifier can be adjusted so that the false negative rate is close to zero). 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 overall form of the detection process is that of a degenerate decision tree, what we call a “cascade” (see Figure 4). A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window.

Stages in the cascade are constructed by training classifiers using AdaBoost and then adjusting the threshold to minimize false negatives. Note that the default AdaBoost threshold is designed to yield a low error rate on the training data. In general a lower threshold yields higher detec-