Image Analysis

Window-based face detection:
The Viola-Jones algorithm

Christophoros Nikou
cnikou@cs.uoi.gr

Images taken from:
Computer Vision course by Kristen Grauman, University of Texas at Austin.
Computer Vision course by Svetlana Lazebnik, University of North Carolina at Chapel Hill.
Face detection and recognition
Face detection and recognition

Detection

Recognition

“Sally”
Consumer application: iPhoto 2009

http://www.apple.com/ilife/iphoto/

C. Nikou – Image Analysis (T-14)
Consumer application: iPhoto 2009

It can be trained to recognize pets!

iPhoto decides that this is a face
• Face recognition
  – Eigenfaces
• Face detection
  – The Viola and Jones algorithm.


Recognition by finding patterns

- We have seen very simple template matching methods (using filters).
- Some objects behave like quite simple templates
  - Frontal face images.
- Strategy:
  - Build/train model (supervised classifier).
  - Find image windows.
  - Correct lighting.
  - Pass them to a statistical test (a classifier) that accepts faces and rejects non-faces with respect to the model.
Basic ideas in classifiers

- Loss: some errors may be more expensive than others
  - e.g. a fatal disease that is easily cured by a cheap medicine with no side-effects
    - false positives in diagnosis are better than false negatives
  - We discuss two class classification: $L(1 \rightarrow 2)$ is the loss caused by calling 1 a 2.

- Total risk of strategy $s$ is the total expected loss:
  \[
  R(s) = \Pr\{1 \rightarrow 2 \mid s\}L(1 \rightarrow 2) + \Pr\{2 \rightarrow 1 \mid s\}L(2 \rightarrow 1)
  \]

- The optimal classifier minimizes the total risk.
Basic ideas in classifiers (cont.)

• Generally, we should classify as 1 if the expected loss of classifying as 2 is larger than calling a 2 a 1:

\[ p(1/x)L(1 \rightarrow 2) > p(2/x)L(2 \rightarrow 1) \]

• Equivalently, we should classify as 2 if

\[ p(1/x)L(1 \rightarrow 2) < p(2/x)L(2 \rightarrow 1) \]

• Using Bayes rule:

\[ p(x/1)p(1)L(1 \rightarrow 2) > p(x/2)p(2)L(2 \rightarrow 1) \]

\[ p(x/1)p(1)L(1 \rightarrow 2) < p(x/2)p(2)L(2 \rightarrow 1) \]
Basic ideas in classifiers (cont.)

- Crucial notion: Decision boundary
  - points where the average loss is the same for either case.

\[ L(1 \rightarrow 2) = L(2 \rightarrow 1) = 1, \]
\[ L(1 \rightarrow 1) = L(2 \rightarrow 2) = 0 \]
• Some loss may be inevitable:
  - the total risk (shaded area) is called the Bayes risk.

\[
L(1 \rightarrow 2) = L(2 \rightarrow 1) = 1, \\
L(1 \rightarrow 1) = L(2 \rightarrow 2) = 0
\]
• **Example**: Gaussian class densities, $p$-dimensional measurements with common covariance $\Sigma$ and different means $\mu_1, \mu_2$ and class priors $\pi_k$:

$$p(x/k) = \left(\frac{1}{2\pi}\right)^{-p/2} |\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right]$$
Basic ideas in classifiers (cont.)

- Ignoring a common factor in the posteriors:

\[
p(k \mid x) \propto \pi_k \left(\frac{1}{2\pi}\right)^{-p/2} |\Sigma|^{-\frac{1}{2}} \exp\left[-\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right]
\]

- The classifier boils down to choose the class minimizing:

\[
(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) - 2 \log \pi_k
\]

- The common covariance simplifies the resulting expression.
• Cross validation to estimate the total risk.
  – Split the data set and average over all possible splits.
  – Leave-one-out, 10-fold,…

• Bootstrapping to estimate better the decision boundary.
  – Retrain the classifier with the false positives (FP) and false negatives (FN).
  – They are the most significant in the configuration of the decision boundary.
Histogram based classifiers

- Use a histogram to represent the class-conditional densities $p(x|1), p(x|2), \ldots$
- Advantage: estimates become quite good with enough data.
- Disadvantage: Histogram becomes big with high dimension.
  - we may assume feature independence.
Example: finding skin pixels

- Skin has a very small range of (intensity independent) colours and little texture.
- Get class conditional densities $p(x|\text{skin})$ and $p(x|\text{not skin})$ from examples.
- Get priors $p(\text{skin})$ and $p(\text{not skin})$ from examples
  - count the number of face/non face pixels.
- The classifier becomes:
  - if $p(\text{skin}|x) > \theta$, classify as skin
  - if $p(\text{skin}|x) < \theta$, classify as not skin
  - if $p(\text{skin}|x) = \theta$, choose classes uniformly and at random
We can represent a class-conditional density using a histogram (a “non-parametric” distribution).
• We can represent a class-conditional density using a histogram (a “non-parametric” distribution)

Now we get a new image, and want to label each pixel as skin or non-skin.
Example: finding skin pixels (cont.)

\[ P(skin \mid x) = \frac{P(x \mid skin)P(skin)}{P(x)} \]

\[ P(skin \mid x) \propto P(x \mid skin)P(skin) \]

Where does the prior come from?

Why use a prior?
For every pixel in a new image, we can estimate the probability that it is generated by skin.

Classify pixels based on these probabilities:

- if $p(\text{skin}|x) > \theta$, classify as skin
- if $p(\text{skin}|x) < \theta$, classify as not skin
- if $p(\text{skin}|x) = \theta$, choose classes uniformly and at random
Example: finding skin pixels (cont.)

Example: finding skin pixels (cont.)

- Parameter $\theta$ encapsulates the relative loss. There is a family of classifiers for each value of $\theta$.
  - Each one has different FP and FN rates described by the Receiver Operating Characteristic (ROC) curve.
Window-based generic object detection 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: building a model

Simple holistic descriptions of image content

- grayscale / color histogram
- vector of pixel intensities
Window-based models: building a model (cont.)

- Pixel-based representations sensitive to small shifts.

- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation.
• Consider edges, contours, and (oriented) intensity gradients.
• 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
• Given the representation, train a binary classifier.
### Discriminative classifiers

**Nearest neighbor**

Shakhnarovich, 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…
Window-based generic object detection 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

Car/non-car Classifier
Window-based models: Summary

Training:
1. Obtain training data
2. Define features
3. Define classifier

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

Nearest neighbor
Shakhnarovich, 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 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.

Boosting intuition

Weak Classifier 1
Boosting illustration

Weights Increased
Boosting illustration

Weak Classifier 2
Boosting illustration

Weights Increased
Boosting illustration

Weak Classifier 3
Final classifier is a combination of weak classifiers
- 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 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).
Challenges of face detection

• Sliding window detector must evaluate tens of thousands of location/scale combinations.
  – PCA is slow for real time face detection.

• Faces in images 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 $\sim10^6$ pixels and a comparable number of candidate face locations.
  – To avoid having a false positive in every image, the 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
  – Represent local texture with efficiently computed rectangular features (integral images) within the window of interest.
  – Select discriminative features as weak classifiers.
  – *Boosting* for classification.
  – *Attentional cascade* for fast rejection of non-face windows.

“Rectangular filters”

Output = \sum (\text{pixels in white area}) - \sum (\text{pixels in black area})
For real problems, the result depends strongly on the features used.

Do not employ pixel values.

Use a very large set of simple functions
  - sensitive to edges and other critical features
  - multiple scales.
Example
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 be quickly computed in one pass.
Computing the integral image
Computing the integral image

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

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

MATLAB: \( \text{ii} = \text{cumsum}(\text{cumsum}(	ext{double}(i)), 2); \)
Computing sum within a rectangle

• Let $A, B, C, D$ be the values of the integral image at the corners of a rectangle.

• 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

- Feature output is different between adjacent regions.
Feature selection

• The integral image significantly reduces the number of computations.

• However, for a base resolution of 24x24 detection region, the number of possible rectangle features is \(~160,000\)! (scale, size, type, position,…)

• It is impractical to evaluate the entire feature set for training.

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

• How to select such a subset?

Use AdaBoost both to select the informative features and to form the classifier.
We seek 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:

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

For next round, reweight the examples according to errors, choose another filter/threshold combo.
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 \(w_{t,i} = \frac{1}{m}, \frac{1}{l}\) 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 \(w_t\), \(e_j = \sum_i w_i |h_j(x_i) - y_i|\).
  3. Choose the classifier, \(h_t\), with the lowest error \(e_t\).
  4. Update the weights:
     \[
     w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}
     \]
     where \(e_i = 0\) if example \(x_i\) is classified correctly, \(e_i = 1\) otherwise, and \(\beta_t = \frac{e_t}{1-e_t}\).

- The final strong classifier is:
  \[
  h(x) = \left\{ \begin{array}{ll} 
  1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
  0 & \text{otherwise} \end{array} \right.
  \]
  where \(\alpha_t = \log \frac{1}{\beta_t}\).

AdaBoost Algorithm

Start with uniform weights on training examples

\[\{x_1, \ldots x_n\}\]

For T rounds

Evaluate weighted error for each feature, pick best.

Re-weight the examples:
- Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995
• The first two features selected by boosting.

• This feature combination can yield 100% detection rate and 50% false positive rate.
• A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 over 14084.

Not good enough!
Attentional cascade

- 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?
- There is no need to compute the linear combination of all of the features in a window that is clear to be negative from the first features.
- Define a computational risk hierarchy.
- The goal of the training process is different
  - instead of minimizing classification errors minimize the false positive rate.
• 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

- Train with 5K positives, 350M negatives (10k non face images).
- Real-time detector using 38 layer cascade.
- Implemented in OpenCV.
Output of face detector on test images
Other detection tasks

Facial Feature Localization

Profile Detection

Male vs. female
Profile Detection
Profile Features
M. Everingham, J. Sivic, and A. Zisserman.
Application of sliding windows: pedestrian detection

SVM on Haar wavelets.


C. Nikou – Image Analysis (T-14)
Application of sliding windows: pedestrian detection (cont.)


Space-time rectangle features.
Application of sliding windows: pedestrian detection (cont.)


SVM on histogram of oriented gradients.
Summary: Viola-Jones detector

- Rectangle features.
- Integral images for fast computation.
- Boosting for feature selection.
- Attentional cascade for fast rejection of negative windows.