What is Object Detection?

- Locate an object in an input image

Extensions
- None or multiple objects
- Object segmentation
- Object detection in videos

Detection as Classification

- Binary classification: $x \in \mathbb{R}^d \rightarrow f(x) \in \{+1, -1\}$
- Training/Learning: find the classifier function $H(x)$
- Example: face detection

Face Detection using AdaBoost

- Viola & Jones, IJCV 2004
- Key ingredients:
  - Feature selection - weak classifiers $\leftrightarrow$ features
  - Fast feature evaluation - integral image
  - Fast classifier evaluation - classifier cascade

Principle of Boosting (Schapire’90)

- Three cobblers with their wits combined, equal Zhuge Liang the master mind. (Or, two heads are better than one).
- In the language of learning: a strong classifier can be created by many weak classifiers.

Principle of Adaboost (Freund&Schipiere’96)

- Different weak classifier have different capability
- Failure is the mother of success
Toy Example – taken from Antonio Torralba @MIT

Each data point has a class label:
\[ y_i = \begin{cases} +1 & \text{if positive} \\ -1 & \text{if negative} \end{cases} \]
and a weight:
\[ w_i = t \]

This one seems to be the best
This is a ‘weak classifier’: It performs slightly better than chance.

We set a new problem for which the previous weak classifier performs at chance again
\[ w_i = w_i \exp(-y_i t) \]

We set a new problem for which the previous weak classifier performs at chance again
\[ w_i = w_i \exp(-y_i t) \]

We set a new problem for which the previous weak classifier performs at chance again
\[ w_i = w_i \exp(-y_i t) \]
The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.

**Toy example**

**Fast Feature Computation**

- **Integral Image**
  - Integration along rows:
    \[ s(x,y) = s(x,y-1) + f(x,y) \]
  - Integration along columns:
    \[ j(x,y) = j(x-1,y) + s(x,y) \]

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

The sum inside rectangle D we can compute as:

\[ j(4) + j(1) - j(2) - j(3) \]

**Haar Feature**

- **Rectangular features**
  - Two-rectangle features (A,B)
  - Three-rectangle feature (C)
  - Four-rectangle feature (D)

- **Properties**
  - Over complete
  - Computationally efficient
  - Feature computation → summing rectangle intensities

**What are the features used?**

- Haar features?
  - There are over 45k different Haar features.
  - Computationally prohibitive.
  - Feature selection needed.

- Joint feature selection and classifier learning
  - Combine a subset of discriminative features to create an effective classifier
  - An effective classifier → Adaboost
  - Selected Haar features → Weak classifiers

**Performance of 200 feature face detector**

A reasonable detection rate of 0.95 can be achieved while maintaining an extremely low false positive rate of approximately 10^-4.

- First features selected by AdaBoost are meaningful and have high discriminative power.
- By varying the threshold of the final classifier one can construct a two-feature classifier which has a detection rate of 1 and a false positive rate of 0.4.

**Speed-up through Cascading**

- Simple, boosted classifiers can reject many negative sub-windows while detecting all positive instances.
- Series of such simple classifiers can achieve good detection performance while eliminating the need for further processing of negative sub-windows

**Training:** subsequent classifiers are trained only on examples which pass through all the previous classifiers.
Experiments (dataset for training)

- 4916 positive training examples
  - Manually cropped
  - Aligned, normalized, and scaled
  - Resolution of 24x24
- 10,000 negative examples
  - Sub-windows from 9500 images which did not contain faces

Experimental Results

- Testing on the MIT+CMU frontal face test set
  - 130 images
  - 507 labelled frontal faces

<table>
<thead>
<tr>
<th>False detections</th>
<th>0</th>
<th>10</th>
<th>31</th>
<th>50</th>
<th>65</th>
<th>78</th>
<th>95</th>
<th>110</th>
<th>167</th>
<th>422</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viola-Jones</td>
<td>78.3%</td>
<td>85.2%</td>
<td>88.8%</td>
<td>89.8%</td>
<td>90.1%</td>
<td>90.8%</td>
<td>91.1%</td>
<td>91.8%</td>
<td>93.7%</td>
<td></td>
</tr>
<tr>
<td>Rowley-Baluja-Kanade</td>
<td>83.2%</td>
<td>86.0%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89.2%</td>
<td>-</td>
<td>90.1%</td>
<td>89.9%</td>
<td></td>
</tr>
<tr>
<td>Schneiderman-Kanade</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>94.4%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Roth-Yang-Ajuha</td>
<td>-</td>
<td>9.4%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

AdaBoost.M1

Discrete AdaBoost(Craven & Schapire 1996b)

1. Start with weights $w_1 = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in [-1,1]$ using weights $w_i$ on the training data.
   (b) Compute $err_m = E_i [I(y_i \neq f_m(x_i))]$, $\alpha_m = \log(1/err_m) / err_m$.
   (c) Set $w_i = w_i \exp(-\alpha_m I(y_i \neq f_m(x_i)))$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $sign(\sum_{m=1}^M \alpha_m f_m(x))$.

Additive Model

- Additive Model
  - $f(x) = \sum_{m=1}^M \beta_m h(x;\gamma_m)$ weak classifier
  - $\sum_{m=1}^M \beta_m h(x;\gamma_m)$ strong classifier
- Fitting an additive model by minimize a loss function $L(y, f(x))$
  \[ (\beta_m, \gamma_m)_{m=1}^M = \arg \min_{(\beta_m, \gamma_m)} L(y, f(x)) \]
  \[ = \arg \min_{(\beta_m, \gamma_m)} \sum_{i=1}^N \sum_{m=1}^M \beta_m h(x;\gamma_m) \]

Forward Stagewise Additive Fitting

1. Input: $X = \{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{-1,1\}$
2. Initialization
   $$ f^{0}(x) = 0 $$
3. For $m=1$ to $M$:
   (a) Compute
   \[ (\beta_m, \gamma_m) = \arg \min_{(\beta_m, \gamma_m)} \sum_{i=1}^N L(y_i, f^{m-1}(x_i) + \beta h(x_i;\gamma_m)) \]
   (b) Set
   \[ f^{m}(x) = f^{m-1}(x) + \beta h(x;\gamma_m) \]
4. Output
   \[ f(x) = f^M(x) \]
Fitting with Exponential Loss

- Exponential loss function
  \[ L(y, f(x)) = \exp(-y \cdot f(x)) \]
- For each component
  \[ (\beta, g) = \arg\min_{\beta, g} \sum_{i=1}^{N} \left[ y_i \left( f(x_i^m) + \beta h(x_i; \gamma) \right) \right] \]
  \[ = \arg\min_{\beta, g} \sum_{i=1}^{N} \exp \left[ -y_i \left( f(x_i^m) + \beta h(x_i; \gamma) \right) \right] \]
  \[ = \arg\min_{\beta, g} \sum_{i=1}^{N} w_i^m \exp \left[ -\beta f(x_i^m) \right] \]
  \[ w_i^m = \exp(-y_i f(x_i^m)) \]

This is AdaBoost!

- For classification, use weak classifiers \( g(x) \in \{-1, 1\} \)
  \[ (\beta, g) = \arg\min_{\beta, g} \sum_{i=1}^{N} w_i^m \exp \left[ -\beta y_i g(x_i) \right] \]
- Solve it, we have
  \[ g_m = \arg\min_{g} \sum_{i=1}^{N} w_i^m I(y_i \neq g(x_i)) \]
  \[ \beta_m = \frac{1}{2} \log \frac{1 - \text{err}_m}{\text{err}_m} \]
  \[ \text{err}_m = \sum_{i=1}^{N} w_i^m I(y_i \neq g(x_i)) \]
  \[ \text{Coefficient for weak classifier } g_m(x) \]
  \[ \text{Weighted training error} \]

Object Detection using
Histogram of Oriented Gradient (HOG)

Dalal & Triggs, CVPR 2005
Followingslides revised from Dalal & Triggs

- Focus on creating robust encoding of images
- Linear SVM as classifier on normalized image windows, is reliable & fast
- Moving window based detector with non-maximum suppression over scale space

Overall Architecture

Learning Phase
- Create normalized training data set
- Encode images into feature vectors
- Learn binary classifier

Detection Phase
- Scan image at all scales and locations
- Run classifier to obtain object/non-object decisions
- Fuse multiple detections in 3-D position & scale space
- Object detections with bounding boxes

Descriptor Processing Chain

Object/Non-object
Linear SVM
Collect HOGs over detection window
Contrast normalize over overlapping spatial cells
Weighted vote in spatial & orientation cells
Compute gradients
Gamma compression

Image Window

Multi-Scale Detection

After dense multi-scale scan of detection window
Map each detection to 3D \([x, y, \text{scale}] \)

\[ H_{ij} = \exp \left( \sum_{k} \exp \left( \gamma_k H \{ f \} \right) \right) \]

Apply robust mode detection, like mean shift

Final detections
HOG for Human Detection

Descriptor Cues: Motorbikes

Figure 6: HOG detection uses mainly on silhouette contours, especially the head, shoulders and feet. The most active blocks are related to the image background and outline the contours. (a) The average gradient image over the training examples. (b) Each pixel shows the maximum positive SVM weight on the block centered on the pixel. (c) Histogram for the negative SVM weights. (d) A raw image with a component HOG descriptors. (e-g) The HOG descriptors weighted by respectively the positive and negative SVM weights.