Lecture 5-6

Object Detection
– Boosting

Tae-Kyun Kim
Face Detection Demo
Multiclass object detection
[Torralba et al PAMI 07]

A boosting algorithm, originally for binary class problems, has been extended to multi-class problems.
Object Detection

**Input:** A single image is given as input, without any prior knowledge.
Object Detection

Output is a set of tight bounding boxes (positions and scales) of instances of a target object class (e.g. pedestrian).
Object Detection

**Scanning windows:** We scan every scale and every pixel location in an image.
Number of Hypotheses

It ends up with a huge number of candidate windows.

\[ \text{Number of Windows: e.g. 747,666} \]
Time per window

What amount of time is given to process a single scanning window?

or raw pixels

Num of feature vectors: 747,666

$f : x \rightarrow t$

Classification: $t \in \{1, 2, ..., n\}$
Time per window

In order to finish the task say in 1 sec

**Time per window (or vector):**
0.00000134 sec

Num of feature vectors:
747,666

Neural Network?  Nonlinear SVM?
Examples of face detection

From Viola, Jones, 2001
More traditionally… The **search space** is narrowed down.

By *Integrating Visual Cues* [Darrell et al IJCV 00].

- Face pattern detection output (left).
- Connected components recovered from **stereo** range data (mid).
- Regions from **skin colour (hue)** classification (right).
Since about 2001 (Viola & Jones 01)…

“Boosting Simple Features” has been a dominating art.

Adaboost classification

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

Strong classifier

Weak classifier

Weak classifiers: Haar-basis features/functions

The feature pool size is e.g. 45,396 (\( \gg T \))

\[ h_t(x) = \begin{cases} +1 & \text{if } f_t(x) \geq \theta \\ -1 & \text{otherwise} \end{cases} \]
Introduction to Boosting Classifiers

- **AdaBoost** (Adaptive Boosting)
Boosting

• Boosting gives good results even if the base classifiers have a performance slightly better than random guessing.

• Hence, the base classifiers are called weak classifiers or weak learners.
For a two (binary)-class classification problem, we train with

- training data $x_1, \ldots, x_N$
- target variables $t_1, \ldots, t_N$, where $t_N \in \{-1, 1\}$,
- data weight $w_1, \ldots, w_N$
- weak (base) classifier candidates $y(x) \in \{-1, 1\}$.

$$Y_M(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m y_m(x) \right)$$
Boosting does

Iteratively,

1) reweighting training samples,
   by assigning higher weights to previously misclassified samples,
2) finding the best weak classifier for the weighted samples.
In the previous example, the weak learner was defined by a horizontal or vertical line, and its direction.
AdaBoost (adaptive boosting)

1. Initialise the data weights \( \{w_n\} \) by \( w_n^{(1)} = 1/N \) for \( n = 1, \ldots , N \).

2. For \( m = 1, \ldots , M \): the number of weak classifiers to choose

   (a) Learn a classifier \( y_m(x) \) that minimises the weighted error, among all weak classifier candidates

   \[
   J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)
   \]

   where \( I \) is the impulse function.

   (b) Evaluate

   \[
   \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}
   \]
and set
\[
\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}
\]

(c) Update the data weights
\[
w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(y_m(x) \neq t_n) \}
\]

3. Make predictions using the final model by
\[
Y_M(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m y_m(x) \right)
\]
Boosting

$m = 150$
Boosting as an optimisation framework
Minimising Exponential Error

- AdaBoost is the sequential minimisation of the exponential error function

\[
E = \sum_{n=1}^{N} \exp\{-t_n f_m(x_n)\}
\]

where \( t_n \in \{-1, 1\} \) and \( f_m(x) \) is a classifier as a linear combination of base classifiers \( y_l(x) \)

\[
f_m(x) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(x)
\]

- We minimise \( E \) with respect to the weight \( \alpha_l \) and the parameters of the base classifiers \( y_l(x) \).
• **Sequential Minimisation:** suppose that the base classifiers $y_1(x), \ldots, y_{m-1}(x)$ and their coefficients $\alpha_1, \ldots, \alpha_{m-1}$ are fixed, and we minimise only w.r.t. $a_m$ and $y_m(x)$.

• The error function is rewritten by

\[
E = \sum_{n=1}^{N} \exp \left\{ -t_n f_m(x_n) \right\}
\]

\[
= \sum_{n=1}^{N} \exp \left\{ -t_n f_{m-1}(x_n) - \frac{1}{2} t_n \alpha_m y_m(x_n) \right\}
\]

\[
= \sum_{n=1}^{N} w^{(m)}_n \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(x_n) \right\}
\]

where $w^{(m)}_n = \exp\{-t_n f_{m-1}(x_n)\}$ are constants.
• Denote the set of data points correctly classified by $y_m(x_n)$ by $T_m$, and those misclassified $M_m$, then

$$E = e^{-\alpha_m/2} \sum_{n \in T_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in M_m} w_n^{(m)}$$

$$= \left( e^{\alpha_m/2} - e^{-\alpha_m/2} \right) \sum_{n=1}^N w_n^{(m)} I(y_m(x) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^N w_n^{(m)}$$

• When we minimise w.r.t. $y_m(x_n)$, the second term is constant and minimising $E$ is equivalent to

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(x) \neq t_n)$$
By setting the derivative w.r.t. $\alpha_m$ to 0, we obtain

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$$

where

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}.$$ 

From

$$E = \sum_{n=1}^{N} w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(x_n) \right\}$$

we have

$$w_n^{(m+1)} = w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(x_n) \right\}$$

As

$$t_n y_m(x_n) = 1 - 2I(y_m(x) \neq t_n),$$

we get

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m / 2) \exp\{\alpha_m I(y_m(x) \neq t_n)\}$$

The term $\exp(-\alpha_m / 2)$ is independent of $n$, thus we obtain

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(x) \neq t_n)\}$$
Pros: it leads to simple derivations of Adaboost algorithms.
Cons: it penalises large negative values. It is prone to outliers.

The exponential (green) rescaled cross-entropy (red) hinge (blue), and misclassification (black) error ftns.
Existence of weaklearners

Definition of a baseline learner

- **Data weights:** $d = (d_1, \ldots, d_N)$
- **Set** $D_+ = \sum_{n:y_n=+1} d_n, \quad D_- = \sum_{n:y_n=-1} d_n$
- **Baseline classifier:** $f_{BL}(x) = \text{sign}(D_+ - D_-)$ for all $x$
- **Error is at most $\frac{1}{2}$.

Each weaklearner in Boosting is required s.t.

$$
\varepsilon(h, d) = \sum_{n=1}^{N} d_n I(y_n \neq h_t(x_n)) \leq 1/2 - 1/2\gamma, \quad (\gamma > 0)
$$

Error of the composite hypothesis goes to zero as boosting rounds increase [Duffy et al 00].
Robust real-time object detector
Viola and Jones, CVPR 01

http://www.iis.ee.ic.ac.uk/icvl/mlcv/viola_cvpr01.pdf
Boosting Simple Features

[Viola and Jones CVPR 01]

Adaboost classification

\[
f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)
\]

Weak classifiers: Haar-basis like functions (45,396 in total feature pool)

\[
h_t(x) = \begin{cases} 
+1 & \text{if } f_t(x) \geq \theta \\
-1 & \text{otherwise}
\end{cases}
\]
Learning (concept illustration)

Face images

Resized to 24x24

Non-face images

Resized to 24x24

Output:

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]
The learnt boosting classifier i.e. \( f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \) is applied to every scan-window.

The **response map** is obtained, then non-local maxima suppression is performed.
Receiver Operating Characteristic (ROC)

Boosting classifier score (prior to the binary classification) is compared with a threshold.

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

- **Threshold**
  - Class 1 (face)
  - Class -1 (no face)

The ROC curve is drawn by the false negative rate against the false positive rate at various threshold values:

- False positive rate (FPR) = FP/N
- False negative rate (FNR) = FN/P

where P positive instances,
N negative instances,
FP false positive cases, and
FN false negative cases.
How to accelerate the boosting training and evaluation
Integral Image

- A value at \((x,y)\) is the sum of the pixel values above and to the left of \((x,y)\).

\[
ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),
\]

\[
s(x, y) = s(x, y - 1) + i(x, y)
\]

\[
ii(x, y) = ii(x - 1, y) + s(x, y)
\]

(where \(s(x, y)\) is the cumulative row sum, \(s(x, -1) = 0\), and \(ii(-1, y) = 0\))

- The integral image can be computed in one pass over the original image.
Boosting Simple Features
[Viola and Jones CVPR 01]

Integral image
- The sum of original image values within the rectangle can be computed: $\text{Sum} = A - B - C + D$
- This provides the fast evaluation of Haar-basis like features

$\text{(6-4-5+3)-(4-2-3+1)}$
In the coursework2, you can first crop image windows, then compute the integral images of the windows, than of the entire image.
Boosting as a Tree-structured Classifier
Boosting (very shallow network)

The strong classifier $H$ as boosted decision stumps has a flat structure

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

- Cf. Decision “ferns” has been shown to outperform “trees” [Zisserman et al, 07] [Fua et al, 07]
Good generalisation is achieved by a flat structure. It provides fast evaluation. It does sequential optimisation.

**Boosting Cascade** [viola & Jones 04], Boosting chain [Xiao et al]
- It is very imbalanced tree structured.
- It speeds up evaluation by rejecting easy negative samples at early stages.
- It is hard to design
A cascade of classifiers

- The detection system requires good detection rate and extremely low false positive rates.
- False positive rate and detection rate are

\[ F = \prod_{i=1}^{K} f_i, \quad D = \prod_{i=1}^{K} d_i, \]

\( f_i \) is the false positive rate of i-th classifier on the examples that get through to it.
- The expected number of features evaluated is

\[ N = n_0 + \sum_{i=1}^{K} \left( n_i \prod_{j<i} p_j \right) \]

\( p_j \) is the proportion of windows input to i-th classifier.
Demo video: Fast evaluation

Super tree on 2D toy data with 3D visualisation
Object Detection by a Cascade of Classifiers

It speeds up object detection by coarse-to-fine search.

Romdhani et al. ICCV01