Introduction To Boosting

Presentation by Alex Kachurin

Feb. 13, 2006

Introduction To Boosting

Presentation by Alex Kachurin
Feb. 13, 2006

- 1.Boosting
  – Overview
  – Definition
  – Discrete AdaBoost algorithm
  – Boosting as greedy minimization technique
- 2. AdaBoost for face detection
  – Integral image and Haar features
  – AdaBoost applied to Face Detection
  – Cascaded AdaBoost
  – ROC curves
  – Comparison to other methods

Introduction to Boosting

- 3.Open problems
  - Basic vs. extended feature set
  - Statistical view of Boosting
  - Boosting variant: RealBoost
  - Boosting variant: LogitBoost
  - Other boosting variants
  - Hypothesis pruning

Boosting Overview

- Introduced by Freund and Shapire, 1996
- Classify objects (2 or more classes)
- Combine simple rules to form an ensemble
- The performance of an ensemble is better than that of each individual rule.
- Each rule should have an error rate slightly better than 50%

Y.Freund and R.E. Shapire. Experiments with a new boosting algorithm

Boosting, definitions

- X – a (high dimension) sample set
- Y – a set of labels
- P(X) = Pr(y(X)==1) probability distribution
- Training set Xt with the same probability distribution as X
- Weak classifiers h(X)
- Produce a strong classifier:

$$Y(x): x \in X \rightarrow \{0,1\}$$
$$y \in \{0,1\}$$

$$X_t \subset X$$

$$h(X): X \rightarrow \{0,1\}$$

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(X)$$

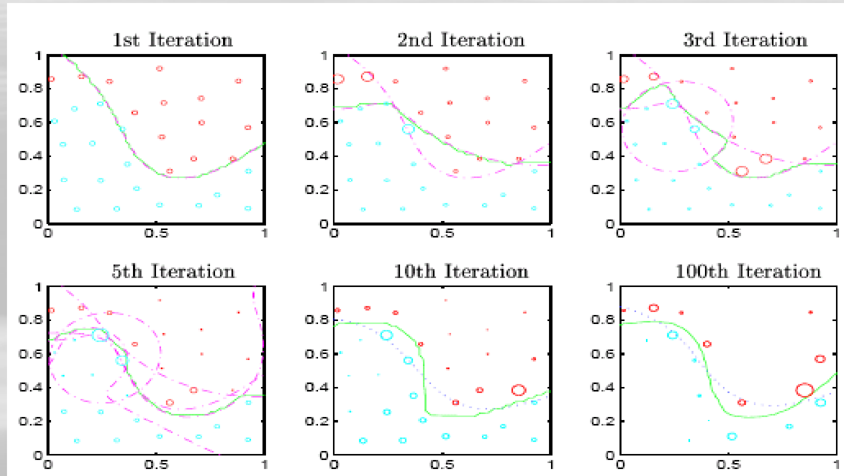
Discrete AdaBoost algorithm

Algorithm 2.1 The AdaBoost algorithm [70].

1. **Input:** $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of Iterations T
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$
   (b) Calculate the weighted training error ε_t of h_t:
   $$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ ,$$
   (c) Set:
   $$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
   (d) Update weights:
   $$d_n^{(t+1)} = d_n^{(t)} \exp\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\} / Z_t \ ,$$
   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\varepsilon_t = 0$ or $\varepsilon_t \geq \frac{1}{2}$ and set $T = t - 1$.
5. **Output:** $f_T(\boldsymbol{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(\boldsymbol{x})$

Ron Meir, Gunar Raetch. An introduction to Boosting and Leveraging

Boosting illustrated



Ron Meir, Gunar Raetch. An introduction to Boosting and Leveraging

Boosting as greedy minimization

- At each stage, the algorithm greedily minimizes the following function:

$$G^{AB} = \sum_{n=1}^{N} d_n \exp(-y_n(\alpha h_t(x_n) + f_{t-1}(x_n)))$$

$$f_{t-1}(x_n) = \sum_{r=1}^{n-1} \alpha_r h_r(x_n)$$

$$M(x_n) = -y_n(\alpha h_t(x_n) + f_{t-1}(x_n))$$

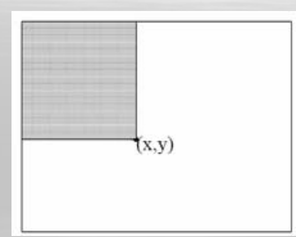
M(x) defines margin over training set, same as SVM

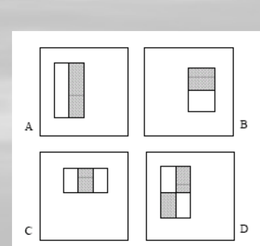
AdaBoost for face detection

- A database of frontal faces (with minor rotations and translations)
- Two classes - face (1) and nonface(0)
- Training set – about 5000 faces and 10000 nonfaces
- Apply AdaBoost to produce a strong classifier with detection rate of about 98%

Integral image and Haar-like features

- Integral image – sum of all pixels above and to the left of a point (x,y)
- Integral image may be computed recursively in O(n)
- Haar-like features use integral image values.
- Basic set includes 4 types of haar-like features
- To produce a weak classifier, subtract a threshold from feature value and use a sign() function !





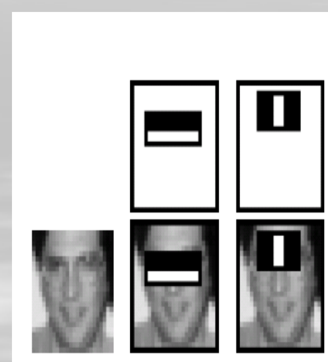
Paul Viola, Michael Jones. Robust Real Time Object Detection

AdaBoost for Face Detection

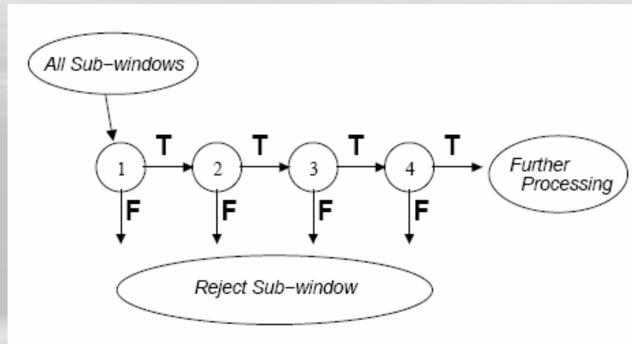
- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  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, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Interpreting features

- Eyes area is generally darker than nose area
- The forehead area should be lighter than eyes area



Boosting cascades

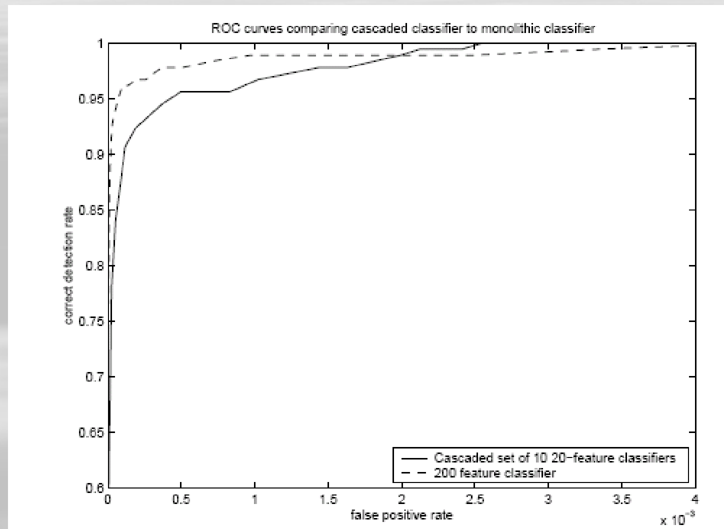


- Faster processing
- Hopefully, better performance

Cascaded AdaBoost

- User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, F_{target}.
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
  - $i \leftarrow i + 1$
  - $n_i = 0$; $F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use P and N to train a classifier with n_i features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine F_i and D_i.
    - Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)
  - $N \leftarrow \emptyset$
  - If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false dectections into the set N

Interpreting results



AdaBoost compared to other classification methods

False detections / Detector	10	31	50	65	78	95	110	167	422
Viola-Jones	78.3%	85.2%	88.8%	89.8%	90.1%	90.8%	91.1%	91.8%	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	-	90.1%	89.9%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-	-	-

Viola-Jones	AdaBoost
Rowley-Baluja-Kanade	Neural networks
Schneiderman-Kanade	Statistics, Bayesian
Roth-Yang-Ahuja	PAC learner, linear features

AdaBoost Performance Observations

- Increasing number of training images significantly improves detection.
- Increasing number of features somewhat improves detection (5% improvement after increasing number of features from 20 to 80)
- Time complexity:

T = O(n*N*t)

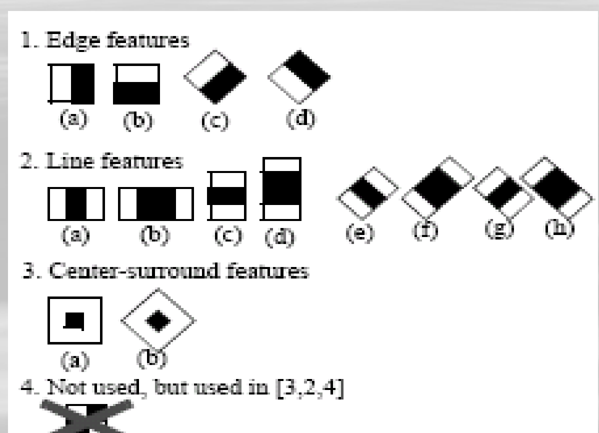
n- total number of training images

N – total number of features

t – number of features in the strong classifier

AdaBoost, open issues

- Basic vs. extended Haar feature set



OpenCV library
http://sourceforge.net/
/projects/opencvlibrary/

Rainer Lienhart, Vadim Pisarevsky, Alexander Kuranov. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection

Statistical view of boosting

- Consider the following criterion:

$$J(F) = E(e^{-yF(x)})$$

- Theorem: the Discrete AdaBoost method fits an additive logistic regression by using adaptive Newton updates to minimize E(J(F))

Additive Logistic Regression: a Statistical View of Boosting by Jerome Friedman, Trevor Hastie, Robert Tibshirani

AdaBoost variants: RealBoost

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
2. Repeat for $m = 1, 2, \dots, M$:
   (a) Fit the class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$ using weights w_i on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log \frac{p_m(x)}{1 - p_m(x)} \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \dots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$

- Better performance than discrete AdaBoost after 200 iterations
- Also minimizes E(J(F)) using Newton-like steps

AdaBoost variants: LogitBoost

$$\mathrm{G}^{LR}(\alpha) = \sum_{n=1}^{N} \log\left\{1 + \exp\left(-y_n(\alpha h_t(\boldsymbol{x}_n) + f_{t-1}(\boldsymbol{x}_n))\right)\right\} \ .$$

LogitBoost (2 classes)

1. Start with weights $w_i = 1/N \ i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights

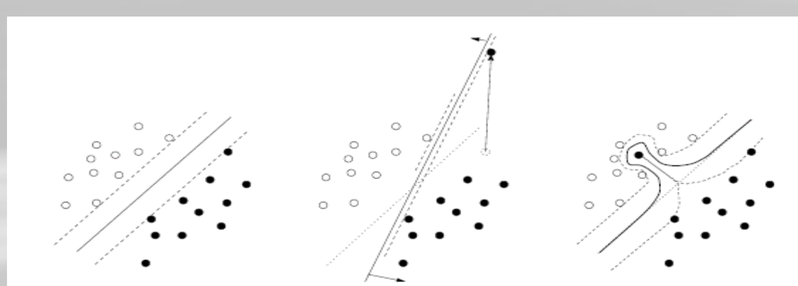
   $$\begin{aligned} z_i &= \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} \\ w_i &= p(x_i)(1 - p(x_i)) \end{aligned}$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.

   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x)$ via (28).
3. Output the classifier $\mathrm{sign}[F(x)] = \mathrm{sign}[\sum_{m=1}^{M} f_m(x)]$

AdaBoost- other variants

- FloatBoost- uses extended feature set, different target function and floating search method to locate a minimum. Very computationally expensive.
- AdaBoost.Reg – regularize training set by limiting weights of outliers. Improves performance in case of noisy data.



Conclusion

- AdaBoost allows to rapidly classify images
- Faster than any existing detector
- Comparable in accuracy to other methods
- Many improvement algorithms exist