Boosting: Combining Classifiers

Zhang Hongxin
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2008-02-28

The most material of this part come from:
http://sifaka.cs.uiuc.edu/taotao/stat/chap10.ppt
Boosting

- **INTUITION (三个臭皮匠，顶个诸葛亮)**
  - *Combining Predictions of an ensemble is more accurate than a single classifier*

- **Reasons**
  - Easy to find quite correct “rules of thumb” however hard to find single highly accurate prediction rule.
  - If the training examples are few and the hypothesis space is large then there are several equally accurate classifiers.
  - Hypothesis space does not contain the true function, but it has several good approximations.
  - Exhaustive global search in the hypothesis space is expensive so we can combine the predictions of several locally accurate classifiers.
Cross Validation (交叉检验)

- **K-fold Cross Validation**
  - Divide the data set into k sub samples
  - Use k-1 sub samples as the training data and one sub sample as the test data.
  - Repeat the second step by choosing different sub samples as the testing set.

- **Leave one out Cross validation**
  - Used when the training data set is small.
  - Learn several classifiers each one with one data sample left out
  - The final prediction is the aggregate of the predictions of the individual classifiers.
Bagging

- Generate a random sample from training set
- Repeat this sampling procedure, getting a sequence of $K$ independent training sets
- A corresponding sequence of classifiers $C_1, C_2, ..., C_k$ is constructed for each of these training sets, by using the same classification algorithm
- To classify an unknown sample $X$, let each classifier predict.
- The Bagged Classifier $C^*$ then combines the predictions of the individual classifiers to generate the final outcome. (sometimes combination is simple voting)
Boosting (Algorithm)

- \( W(x) \) is the distribution of weights over the \( N \) training points \( \sum W(x_i) = 1 \)
- Initially assign uniform weights \( W_0(x) = \frac{1}{N} \) for all \( x \), step \( k=0 \)

- At each iteration \( k \):
  - Find best weak classifier \( C_k(x) \) using weights \( W_k(x) \)
  - With error rate \( \epsilon_k \) and based on a loss function:
    - weight \( \alpha_k \) the classifier \( C_k \)'s weight in the final hypothesis
    - For each \( x_i \), update weights based on \( \epsilon_k \) to get \( W_{k+1}(x_i) \)

- \( C_{FINAL}(x) = \text{sign} \left[ \sum \alpha_i C_i(x) \right] \)
Boosting (Algorithm)

Final Classifier

\[ G(x) = \text{sign} \left[ \sum_{m=1}^{M} \alpha_m G_m(x) \right] \]
AdaBoost(Algorithm)

- $W(x)$ is the distribution of weights over the $N$ training points $\sum W(x_i) = 1$
- Initially assign uniform weights $W_0(x) = 1/N$ for all $x$.
- At each iteration $k$:
  - Find best weak classifier $C_k(x)$ using weights $W_k(x)$
  - Compute $\varepsilon_k$ the error rate as $\varepsilon_k = [\sum W(x_i) \cdot I(y_i \neq C_k(x_i))] / [\sum W(x_i)]$
  - Weight $\alpha_k$ the classifier $C_k$'s weight in the final hypothesis Set $\alpha_k = \log ((1 - \varepsilon_k)/\varepsilon_k)$
  - For each $x_i$, $W_{k+1}(x_i) = W_k(x_i) \cdot \exp[\alpha_k \cdot I(y_i \neq C_k(x_i))]$
  - $C_{FINAL}(x) = \text{sign} [\sum \alpha_i C_i(x)]$

$L(y, f(x)) = \exp(-y \cdot f(x))$ - the exponential loss function
AdaBoost (Example)

Original Training set: Equal Weights to all training samples
AdaBoost(Example)

ROUND 1

\[ \alpha_k = \log\left(\frac{1 - \varepsilon_k}{\varepsilon_k}\right) \]

\[ \varepsilon_1 = 0.30 \]

\[ \alpha_1 = 0.42 \]
AdaBoost (Example)

ROUND 2

$h_2$

$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

$D_3$
AdaBoost (Example)

ROUND 3

\[ h_3 \]

\[ \varepsilon_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
AdaBoost(Example)

\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]

Show demo ...
Boosting

- The final prediction is a combination of the prediction of several predictors.

**Differences** between Boosting and previous methods?
- It is **iterative**
- Boosting: Successive classifiers **depends upon its predecessors**.
  - Previous methods: Individual classifiers were independent.
- Training examples may have **unequal weights**.
- Look at errors from previous classifier step to decide how to focus on next iteration over data
- Set weights to focus more on ‘hard’ examples. (the ones on which we committed mistakes in the previous iterations)

\[
\hat{f}(x) = \sum_{i=1}^{k} w_i h_i \quad \text{and} \quad C(x) = \sum_{i=1}^{k} \alpha_i C_i(x; w_i)
\]
AdaBoost Case Study:

- Rapid Object Detection using a Boosted Cascade of Simple Features (IEEE CVPR2001)
Object detection using AdaBoost

- Object Detection
- Features
  - *two-rectangle*
  - *three-rectangle*
  - *four-rectangle*

Size: 24x24
Feature: 180,000
**Integral Image**

Definition: The integral image at location \((x,y)\) contains the sum of the pixels above and to the left of \((x,y)\), inclusive:

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

Using the following pair of recurrences:

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

Using the integral image any rectangular sum can be computed in four array references

\[ ii(4) + ii(1) - ii(2) - ii(3) \]
• 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 \(\epsilon_t\).

  4. Update the weights:

     \[
     w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_t}
     \]

     where \(\epsilon_t = 0\) if example \(x_i\) is classified correctly, \(\epsilon_t = 1\) otherwise, and \(\beta_t = \frac{\epsilon_t}{1-\epsilon_t}\).

• The final strong classifier is:

     \[
     h(x) = \begin{cases} 
     1 & \sum_{i=1}^{T} \alpha_i h_i(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
     0 & \text{otherwise}
     \end{cases}
     \]

     where \(\alpha_t = \log \frac{1}{\beta_t}\)
Homework

- Implement this CVPR paper.
  - Hint: You can use OpenCV.
Thank you