Boosting Methods

Lecture notes for Cmput466/551 30/Mar/05

S Wang

Boosting: A powerful procedure that combines the outputs of many "weak" classifiers to produce a "strong" classifier.

Consider binary classification problem:

- $Y \in \{-1, 1\}$
- X input variables
- $h(x)$: a classifier produces a prediction taking one of the values $\{-1, 1\}$
- Training data: $\Big\{(x_1, y_1), \cdots, (x_N, y_N)\Big\}$
- Error rate on training samples

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq h(x_i)$$

- Expected error rate: $E_{X,Y} I(Y \neq h(X))$

"Weak" classifier: a classifier whose error rate is only slightly better than random guessing.

The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of data, thereby producing a sequence of wek classifiers $h_m(x), m = 1, \cdots, M$. The predictions from all of them are then combined through a weighted mojority vote to produce the final prediction:

$$h(x) = \text{sign}\left(\sum_{m=1}^{M} \beta_m h_m(x)\right)$$

here $\beta_1, \cdots, \beta_M$ are computed by the boosting algorithm.

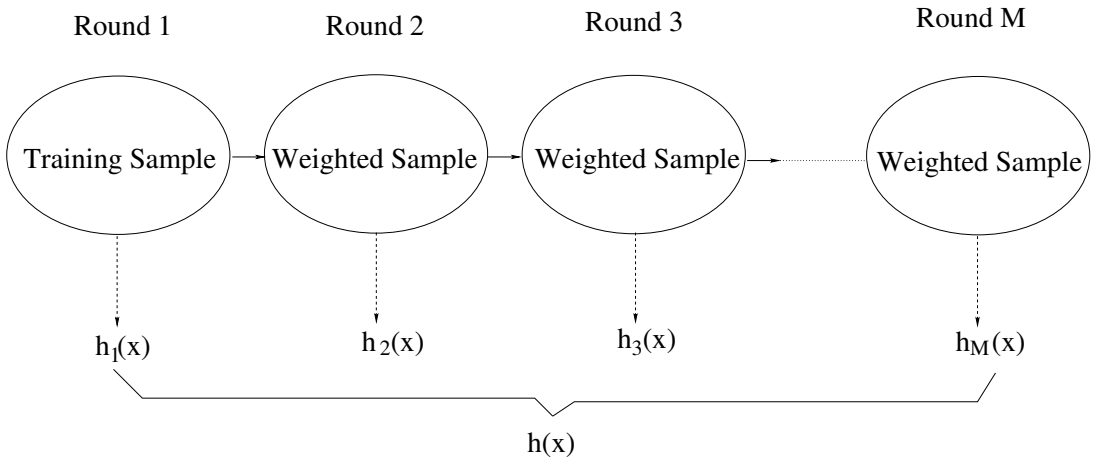

Figure 1: Classifiers are trained on weighted versions ofthe dataset, and then combined tp produce a final prediction.

AdaBoost (Freund and Schapire 1997)

1. Initialize the observation weights $w_i = \frac{1}{N}, i = 1, \cdots, N$
2. For $m = 1, \cdots, M$
   (a) Fit a classifier $h_m(x)$ to the training data using weights w_i
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h(x_i)}{\sum_{i=1}^{N} w_i}$$
   (c) Compute $\beta_m = \frac{1}{2} \log \left(\frac{(1-\text{err}_m)}{\text{err}_m} \right)$
   (d) Set
   $$w_i \leftarrow w_i \exp \Big[-\beta_m y_i h_m(x_i) \Big] = \begin{cases} w_i e^{-\beta_m} & \text{if } h_m(x_i) = y_i \\ w_i e^{\beta_m} & \text{if } h_m(x_i) \neq y_i \end{cases} \quad i = 1, \cdots, N$$
   , and renormalize so that $\sum_{i=1}^{N} w_i = 1$.
3. Output $h(x) = \text{sign} \left(\sum_{m=1}^{M} \beta_m h_m(x) \right)$

Note: Line 2b, "Weak" learner: $\text{err}_m < 0.5$

$$\text{err}_m < 0.5 \Longrightarrow 1 - \text{err}_m > 0.5 \Longrightarrow \frac{(1 - \text{err}_m)}{\text{err}_m} > 1 \Longrightarrow \beta_m > 0$$

This means in Line 2d, at step m, those observations that were misclassified by the classifier $h_{m-1}(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that were classified correctly.

Toy example: decision stumps (http://www.cs.princeton.edu/courses/archive/spring04/cos511/boost-slides.pdf)

There are many ways to derive and explain AdaBoost algorithm, such as gradient descend, game theory, linear programming, maximum generalized entropy, dynamical systems et al. Here we show that AdaBoost fits an additive model in a base learner, optimizing an exponential loss function.

Exponetial Loss and AdaBoost

Define:

- Loss function $L(y, f(x)) = \exp(-yf(x))$
- A set of basis functions $h(x; \theta_m), m = 1, \cdots, L$
- $f(x) = \sum_{m=1}^{L} \beta_m h(x; \theta_m)$

Optimization problem:

$$\min_{\beta_m, \theta_m} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{L} \beta_m h(x_i; \theta_m)) \tag{1}$$

Computationally "hard", try to approximately solve it.

Define $f_m(x) = f_{m-1}(x) + \beta_m h_m(x)$. Using the exponential loss function try to solve

$$(\beta_m, h_m) = \arg\min_{\beta, h} \sum_{i=1}^{N} \exp \Big[-y_i (f_{m-1}(x) + \beta h(x)) \Big] \tag{2}$$

for the classifier h_m and corresponding coefficient β_m to be added at each step. This can be expressed as:

$$(\beta_m, h_m) = \arg\min_{\beta,h} \sum_{i=1}^{N} w_i^{(m)} \exp\Big[-\beta y_i h(x))\Big] \tag{3}$$

with $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$. Since each $w_i^{(m)}$ depends neither on β nor $h(x)$, it can be regarded as a weight that is applied to each observation. This weight depends on $f_{m-1}(x_i)$, and so the individual weight values change with each iteration m.

The solution to (3) can be obtained in two steps. First, for any values of $\beta > 0$, solving (3) for $h_m(x)$ is equivalent to solving

$$h_m = \arg\min_{h} \left[e^{-\beta} \sum_{y_i = h(x_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq h(x_i)} w_i^{(m)} \right] \tag{4}$$

The criterion in (4) in turn can be written as

$$(e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq h(x_i)) + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)} \tag{5}$$

Thus we have

$$h_m = \arg\min_{h} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq h(x_i)) \tag{6}$$

which is the classifier that minimizes the weighted error rate in predicting y.

Plugging this h_m into (3) and solving for β one obtains

$$\beta_m = \frac{1}{2} \log \left(\frac{(1 - \mathrm{err}_m)}{\mathrm{err}_m} \right) \tag{7}$$

which err_m is the minimized weighted error rate

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq h_m(x_i)}{\sum_{i=1}^{N} w_i^{(m)}} \tag{8}$$

The approximation is then updated

$$f_m(x) = f_{m-1}(x) + \beta_m h_m(x)$$

which causes the weights for the next iteration to be

$$w_i^{(m+1)} \leftarrow w_i^{(m)} \exp\Big[-\beta_m y_i h_m(x_i)\Big] \tag{9}$$

References

[1] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119139, August 1997.

[2] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Spring-Verlag. 2001.