CS 559 Machine Learning

Boosting

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Plan for today

Boosting

Reduce Bias and Decrease Variance

- Bagging reduces variance by averaging
- Bagging has little effect on bias
- Can we average and reduce bias?
- Yes: Boosting

Example: Email Spam

How would you classify an email as SPAM or not? using following criteria. If:

1. Email has only one image file (promotional image), SPAM
2. Email has only link(s), SPAM
3. Email body consist of sentence like "You won a prize money of $ xxxxxx", SPAM
4. Email from our official domain "stevens.edu" , Not a SPAM
5. Email from known source, Not a SPAM

The Boosting Approach

- devise computer program for deriving rough rules
- apply procedure to subset of examples
- obtain a simple rule
- apply to 2nd subset of examples
- obtain a 2nd rule
- repeat T times

Key details

- How to choose examples on each round?
  - concentrate on "hardest" examples (those most often misclassified by previous rule)
- How to combine the rules into single prediction rule?
  - take (weighted) majority vote of rules

Boosting

- boosting = general method of converting rough rules into highly accurate prediction rule
- technically
  - assume given "weak" learning algorithm that can consistently find classifiers at least slightly better than random, say, accuracy $\geq 55\%$
  - given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy say, 99%

A formal description of boosting

- Given training set $(x_1, y_1), ..., (x_N, y_N)$
- $y_i \in \{-1, +1\}$ correct labels of instance $x_i \in X$
- for $t = 1, ..., T$:
  - construct weight distribution $u^{(t)}$ on $\{1, ..., N\}$
  - find weak classifier:

$$f_t : X \to \{-1, +1\}$$

    with error ϵ_t on $u^{(t)}$:

$$\epsilon = P_{i \sim u^{(t)}}[f_t(x_i) \neq y_i]$$

- Output final/combined classifier H_{final}

The Idea of AdaBoost

- Training $f_2(x)$ on the new training set that fails $f_1(x)$
- How to find a new training set that fails $f_1(x)$? ϵ_1: the error rate of $f_1(x)$ on its training data

$$\epsilon_1 = \frac{\sum_n u_n^{(1)} \delta(f_1(x_n) != y_n)}{Z_1}$$

  where $Z_1 = \sum_n u_n^{(1)}$, $\epsilon_1 < 0.5$

- Changing the example weights from $u_n^{(1)}$ to $u_n^{(2)}$ such that: $\frac{\sum_n u_n^{(2)} \delta(f_1(x_n) != y_n)}{Z_2} = 0.5$ The performance of f_1 for new weights would be random
- Training $f_2(x)$ based on the new weights $u_n^{(2)}$

AdaBoost

- constructing $u^{(t)}$
  - $u_i^{(1)} = 1/n$
  - given $u^{(t)}$ and f_t:

$$u_i^{(t+1)} = \frac{u_i^{(t)}}{Z^t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = f_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq f_t(x_i) \end{cases}$$
$$= \frac{u_i^{(t)}}{Z^t} \times exp(-\alpha_t y_i f_t(x_i))$$

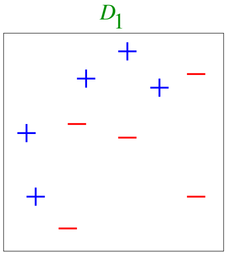
  where

$$Z^t = \text{ the normalization factor}$$
$$\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t}) > 0$$

- final classifier

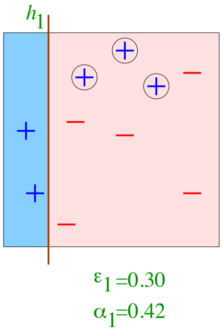
$$H_{\text{final}}(x) = \text{sign}(\sum_t \alpha_t f_t(x))$$

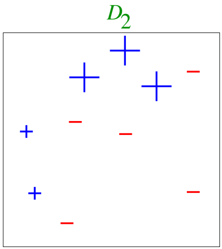
Toy example



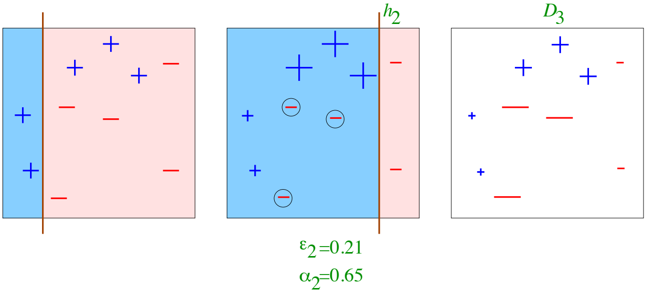
weak classifiers = vertical or horizontal half-planes

Round 1

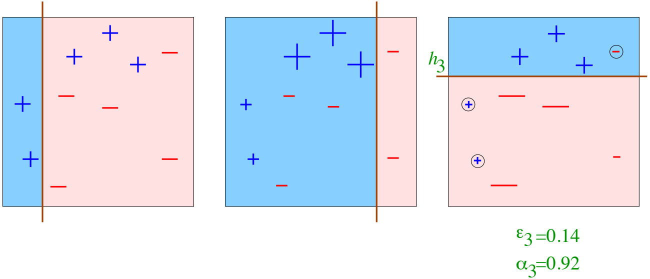




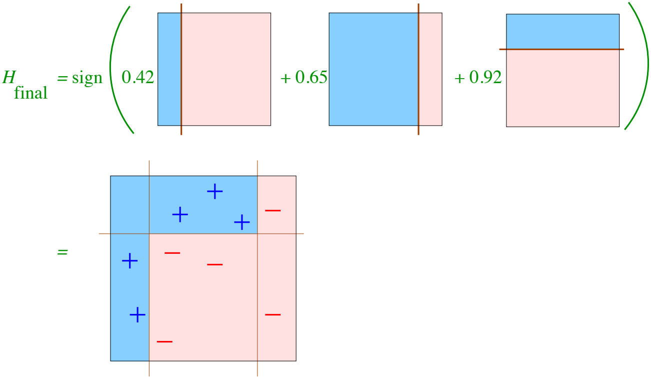
Round 2



Round 3



Final Classifier



Voted combination of classifiers

- The general problem here is to try to combine many simple "weak" classifiers into a single "strong" classifier
- We consider voted combinations of simple binary component classifiers

$$f_T(x) = \alpha_1 f(\mathbf{x}; \theta_1) + ... + \alpha_T f(\mathbf{x}; \theta_T)$$

where θ is the model parameter and the (non-negative) votes α_i can be used to emphasize component classifiers that are more reliable than others.

The AdaBoost algorithm

1. Set $u_i^{(0)} = 1/n$ for $i = 1, \ldots, n$
2. At the m^{th} iteration we find (any) classifier $f(\mathbf{x}; \hat{\theta}_m)$ for which the weighted classification error ϵ_m

$$\epsilon_m = 0.5 - \frac{1}{2}\Big(\sum_{i=1}^{n} u_i^{(m-1)} y_i f(\mathbf{x}_i; \hat{\theta}_m)\Big)$$

   is better than chance
3. The new component is assigned votes based on its error (smaller error rate, larger weight for voting)

$$\hat{\alpha}_m = 0.5 \log \frac{1 - \epsilon_m}{\epsilon_m}$$

4. The weights are updated according to (Z_m is chosen so that the new weights $u_i^{(m)}$ sum to one):

$$u_i^{(m)} = \frac{1}{Z_m} \cdot u_i^{(m-1)} \exp(-y_i \hat{\alpha}_m f(\mathbf{x}_i; \hat{\theta}_m))$$

Acknowledgements

Slides adapted from Dr. David Sontag's *Machine Learning* at MIT.