

# Ensemble Learning

**(With Emphasis on Boosting)**

# Ensemble Learning

- It is often found that the decision and opinion of a group is better than that of a single individual
- In everyday-life, it is common practice for all of us to consult multiple experts before making a crucial decision
- Similarly, consulting group of classifiers can enhance performance on classification task

# Bagging (In a nutshell)

- Training examples are sampled into many sets with replacement – Same example can be used multiple times
- For each set of training data, a classifier is obtained by training on that particular set
- Same type of classifier is used for every partition
- Many classifiers trained on slightly different data are obtained
- The result is usually obtained by voting (all classifiers have equal weight)

# Boosting

- Principle: Many weak learners, where each learner performs better than random guessing, can be combined to form a successful classifier
- Building a highly accurate prediction rule is very hard
- Combining many rough rules of thumb may be easier, hence many classifiers are generated
- Classifiers complement each other as a new classifier is learned in a way so as to complement the one before it

# Bagging vs Boosting

- In Bagging, classifiers might not be significantly different from each other
- Boosting can create classifiers that complement each other
- Like Bagging, same type of classifier is used
- Unlike Bagging, instead of random sampling of data, weight assigned to examples

# Boosting

- Two Questions:
- How should the distribution be chosen?
- How should the classifiers be combined?

# Boosting

- Distribution choice: Previously misclassified and hard examples are weighted more heavily than the correctly classified examples
- For combining the rules, weighted majority vote is used

# AdaBoost

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize  $D_1(i) = 1/m$ .

For  $t = 1, \dots, T$ :

- Train base learner using distribution  $D_t$ .
- Get base classifier  $h_t : X \rightarrow \mathbb{R}$ .
- Choose  $\alpha_t \in \mathbb{R}$ .
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where  $Z_t$  is a normalization factor (chosen so that  $D_{t+1}$  will be a distribution).

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right).$$

Figure 1: The boosting algorithm AdaBoost.

# AdaBoost

- AdaBoost calls a given base learning algorithm repeatedly in a series of rounds 1, 2, 3,.....T.
- Set of weights are maintained over the training set – Each example has a weight that is updated regularly
- Weight of training example  $i$  in round  $t$  is denoted by  $D_t(i)$
- Initially all weights are equal – hence  $D_1(i) = 1/m$  where  $m$  is the number of examples
- The weights are updated so as to assign higher weight to misclassified examples:-

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

# AdaBoost

- For each Distribution  $D_t$ , hypothesis (base learner)  $h_t$  is obtained
- The base learner minimizes the error:-

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

# AdaBoost

- The weight assigned to each hypothesis (in binary classifier) is given by:-

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$

For details of this derivation, refer to Freund and Schapire (1997)

# AdaBoost

- The final classifier is obtained by majority vote of  $T$  classifiers where  $\alpha_t$  is the weight assigned to each classifier  $h_t$ :

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right).$$

# Analyzing the training error

- The training error of the final classifier is bounded by:

$$\frac{1}{m} |\{i : H(x_i) \neq y_i\}| \leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$$

- The above equation suggests that training error can be reduced by choosing  $\alpha_t$  and  $h_t$  so as to minimize the

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)).$$

# Analyzing the training error

- For Binary classifiers:

$$\prod_t Z_t = \prod_t \left[ 2\sqrt{\epsilon_t(1-\epsilon_t)} \right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

where

$$\gamma_t = 1/2 - \epsilon_t.$$

- The above eq. shows that for each base classifier slightly better than random, the training error decreases exponentially

# Analyzing the training error

- AdaBoost is nothing but an algorithm to CONDUCT STEEPEST DECENT SEARCH TO MINIMIZE THIS EQUATION:

$$\sum_i \exp(-y_i f(x_i)) = \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right).$$

# Generalization Error

- Earlier, we calculated error only on training examples. However, it would be interesting to know algorithm's capacity on unseen examples too
- Probability of misclassifying new example is at most:

$$\hat{\Pr}[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

- This equation suggests that there will be overfitting if run continues for many rounds
- However, empirical evidence points to contrary. Boosting does not overfit, even if run for many rounds

# Multiclass Classification

- More sophisticated binary classification  
a vs not-a (all labels but a)  
b vs not-b (all labels but b) and so on
- Error correcting output codes

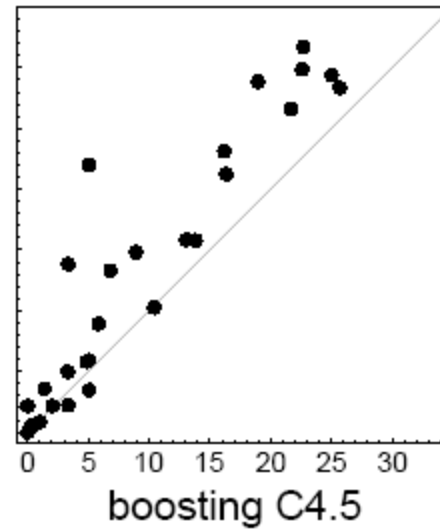
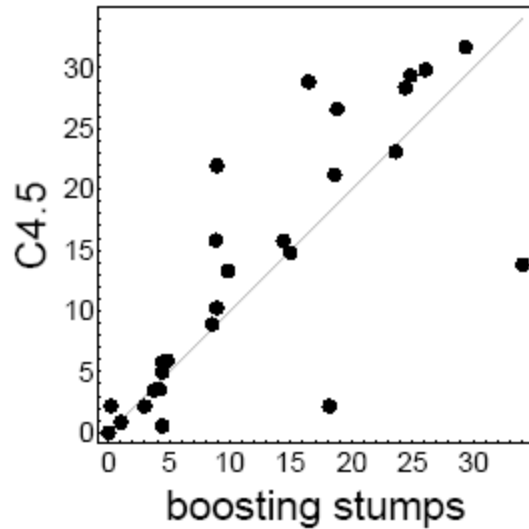
# Boosting (Problems)

- Actual Performance – dependent on data and the type of classifier
- Can fail to perform on insufficient data,
- Can fail to perform with overly complex classifier, and weak classifiers
- Susceptible to Noise

# Boosting (Applications)

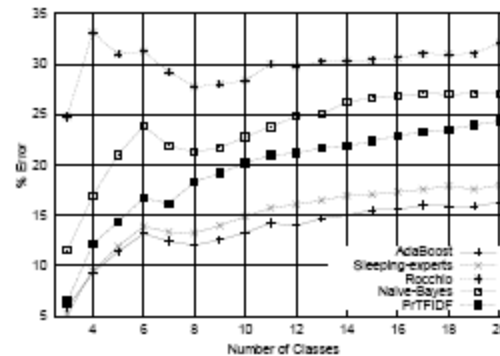
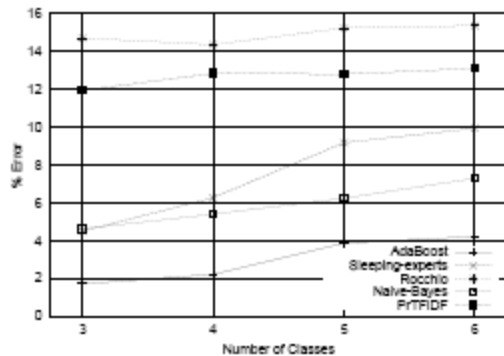
- Text Categorization
- Routing
- Medical diagnosis
- Natural language processing
- Image retrieval

# AdaBoost (Empirical Results)



# AdaBoost

## (Empirical Results)



# Stacking

- Stacked Generalization – Developed by Wolpert
- Unlike Bagging and Boosting, different kinds of classifiers can be used
- Since different algorithms have different strengths and weaknesses, search space exploration is more exhaustive
- Stage 1 – Different types of Classifiers are trained on the same training set
- Stage 2 – The Outputs of these Classifiers is fed to the meta-learner (itself a classifier), that makes a final decision on classification