Introduction
- Learning algorithm operates on given set of instances to produce a classifier
- Goal is to find classifier with low generalization error
- Focus on algorithm which achieve high accuracy by voting
  - Base classifier – each classifier combined in vote
  - Combined classifier – final vote classifier
- Boosting and bagging two common method
- Analysis of prediction error

Background
- Valiant'84 introduced theoretical PAC model for studying machine learning
- Kearns&Valiant'88 Can weak learner be “boosted” into accurate algorithm?
- Schapire'89, Freund'90 first polynomial-time boosting algorithms
- Freund&Schapire '95 introduced AdaBoost algorithm
- Strong practical advantages over previous boosting algorithms
- Continuing development of theory & algorithms:
  - Schapire,Freund,Bartlett&Lee '97
  - Schapire&Singer '98
  - Breiman '97
  - Mason, Bartlett&Baxter '98
  - Grive and Schuurmans '98
  - Friedman, Hastie&Tibshirani '98

Bagging
- Combined the prediction of several classifiers
- Repeatedly
  - Samples data with replacement from the training set
  - Train a new classifier on the sample data
- The predictions of the classifier are combined by majority vote
- Bagging works by reducing the variance part

Boosting
- Popular method of producing ensemble
- General method of converting rule of thumbs into highly accurate prediction rule.
- "Weak" learning algorithm combines to consistently find hypothesis with lower error
- Final Hypothesis: f(h1, h2, h3, …ht)
Idea of Boosting

- Examine the training set $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $y_i \in \{-1, +1\}$ is the correct label of instance $x_i$.
- Derive some rough rule of thumb.
- Reweight the sample to concentrate on “hard” cases for the previous rule.
- Derive a second rule of thumb.
- Repeat $T$ times.
- Combine the rules of thumb into a single accurate rule.

Boosting works by reducing the bias part.

Boosting: Reweighing the sample

- For $t = 1, \ldots, T$:
  - Construct distribution $D_t$ on $\{1, \ldots, m\}$.
  - Find weak hypothesis (“rule of thumb”) $h_t : X \rightarrow \{-1, +1\}$ with small error $\epsilon_t$ on $D_t$:
    \[ \epsilon_t = \Pr_{(x,y) \sim D_t} [h_t(x) \neq y] = \sum_{i=1}^{m} D_t(i) \]
  - Output final hypothesis $H_{\text{final}}$.

Ada Boost

- Constructing $D_t$:
  - Given $D_t$ and $h_t$:
    \[ D_{t+1}(i) = \frac{D_t(i)}{Z_t} \]
    \[ D_{t+1}(i) = \begin{cases} \exp(-\epsilon_t) & \text{if } y_i = -h_t(x_i) \\ \exp(\epsilon_t) & \text{if } y_i \neq -h_t(x_i) \end{cases} \]
    where: $Z_t = \text{normalization constant}$
  - Final hypothesis:
    \[ H_{\text{final}}(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

Illustrative Example

Example cont...

- Input data points and coefficients:
  - $h_1$ and $D_1$
  - $h_2$ and $D_2$
  - $h_3$ and $D_3$

- Output final hypothesis $H_{\text{final}}$.
Overview

- Introduction
- Idea of Boosting
- Error Analysis of boosting
- Generalization error analysis based on margin
- Experiments
- Conclusions

Generalization Error

- Generalization error bound of the final hypothesis in terms
  - training error
  - the sample size
  - VC dimension of the hypothesis space
  - the number of boosting round
- As classifier becomes more complex, test error expected to increase – Occam’s razor

Generalization Error Analysis

- Theorem [Freund & Schapire 97]:
  write $\epsilon_t$ as $\frac{1}{2} \gamma_t$
  then $\text{training error}(H_{\text{final}}) \leq \exp \left( -2 \sum \gamma_t^2 \right)$
  so if $\forall \gamma_t \geq \gamma > 0$ then
  $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2}$
  So, training error continues to drop and reaches zero as boosting iteration (T) is increased

Final Hypothesis

$x_1 < 0.14$
$\epsilon_T = 0.92$

$-0.42 - 0.65 + 0.92 = -0.15 > 0$
A Typical Test Run

- Test error does not increase even after 1,000 rounds (~2,000,000 nodes)
- Test error continues to drop after training error is zero!
- Occam’s razor wrongly predicts “simpler” rule is better.

Another Argument

- Based on bias and variance – by Breiman and others
- Voting method works by reducing the variance of a learning algorithm
- Useful explanation for bagging but incomplete for boosting
- Large variance not a requirement for boosting

A reasonable argument

- “Voting the classifiers does not increase the complexity, but merely smooth the prediction”
  - The complexity of such combined classifier much greater than the base and may result in overfit

A better Explanation: Margin

- Consider more than just the training error
- Take into account the classifier confidence
- Margin – a measure of classification confidence
- Improvements on margin on the training set guarantees an improvement in the upper bound on the generalization error

Margin of binary class problem

- For binary class problem, the classification of an example is correct if \( \text{sgn}(f(x)) = y \)
- So in this case margin = \( y \cdot f(x) \)
  
  \[
  f(x) = \sum_{i=1}^{T} \alpha_i (x) \epsilon_{[-1,1]} \]

  - Higher margin => lower generalization error

Margin

- Boosting constructs hypothesis of the form \( \text{sgn}(f(x)) \)
- The prediction of the combined classifier is the result of the vote over a set of base classifiers. The weights assigned to the base classifiers sums to 1
- The classification margin is defined as the difference between the weight assigned to the correct label and the maximum weight assigned to the incorrect label.

Generalization error as a function of margin distribution

- Margin distribution graphs – plot of fraction of examples whose margin is at most \( x \) as a function of \( x \in [-1, 1] \)
- Bagging and Boosting
  - increase the margins associated with training examples
  - converge to a margin distribution with most examples having large margins
- Experiments shows correlation between a reduction between fraction of examples with small margin and improvements in the test error
Effect of boosting on the margin

Bounds on Generalization error
- upper bounds on generalization error of:
  - in terms of $\#$ training examples
  - complexity of base hypotheses =
    - but not on $\#$ of base classifiers
  - considers not only training error but $\#$ incorrect classifications, and confidence of classifications
- these bounds imply:
  - $\#$ of training examples with small margin drops exponentially fast with the number of base classifiers
  - Theorem proves achieving a large margin results in an improved bound

Theorem 1 – finite base-classifier space
For $\delta > 0$ then with probability at least $1 - \delta$ every weighted average function $f$ satisfies the following bound for all $\theta > 0$:

$$P_{\theta} [y(f(x) \leq 0) \leq P_B [y(f(x) \leq \theta] + \frac{1}{\sqrt{N}} \left( \log N \log |\mathcal{H}| + \log(1/\beta) \right)^{1/2}$$

Taking $N$ to infinity, and by substituting for $\#$ of hypotheses:

$$P_{\theta} [y(f(x) \leq 0] \leq P_B [y(f(x) \leq \theta] + \frac{\sqrt{2} N \log |\mathcal{H}|}{N^{1/2}} - \frac{1}{2} \left( \frac{\log N}{N} \right)^{1/2}$$

Theorem 2 – infinite base-classifier space
Assume $N$ (number of training examples) $\geq d \geq 1$
For $\delta > 0$ then with probability at least $1 - \delta$ every weighted average function $f$ satisfies the following bound for all $\theta > 0$:

$$P_{\theta} [y(f(x) \leq 0] \leq P_B [y(f(x) \leq \theta] + \frac{1}{\sqrt{N}} \left( \log^2 (1/\delta) + \log(1/\beta) \right)^{1/2}$$

Bound dependence on training set

Effect of boosting on margin distribution
Suppose the base learner when called by AdaBoost generates weighted training errors: $\epsilon_1, \ldots, \epsilon_M$ then for any theta we have:

$$P_{\epsilon_1, \ldots, \epsilon_M} [y(f(x) \leq \theta] \leq 2^M \prod_{i=1}^M 1 - \epsilon_i \log(1 - \epsilon_i)$$

If error $< \frac{1}{2}$ for all $D_i$ then the training error of the combined classifier decreases exponentially fast with $M$.

In effect: the larger our aggregation size $M$, the more we shift the distribution of margins towards the right.

Example: training error rate is $\frac{1}{4}$ for all rounds, then for theta = 0:
- right hand term
  - $M=10$ : 0.2373
  - $M=100$: 5.663 x 10^-7

Effect of boosting on the margin

<table>
<thead>
<tr>
<th>epoch</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>training error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>0.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
<tr>
<td>%margins/0.5</td>
<td>7.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Minimum margin</td>
<td>0.14</td>
<td>0.52</td>
<td>0.55</td>
</tr>
</tbody>
</table>
Experimental Results

Relation to Bias – Variance theory

- **Bias – Variance Decomposition**
  - Separate the expected error of a classifier into a bias term and a variance term.
  - Bias measures the persistent error of a learning algorithm.
  - Variance term measures the error due to the fluctuations for one single classifier.

Definition of Bias and Variance

- By Kong & Dietterich

  - \( C_b \) : classification rule from one base learner given training set \( S \).
  - \( C_m \) : classification rule from majority vote of base learners, each which are run on infinite # of training sets.
  - \( C^* \) : Bayes optimal prediction rule given distribution \( D \).

  \[
  \text{PE}(C) = P_{x,y \sim D}(C(x) \neq y) \\
  \text{Bias} = \text{PE}(C_b) - \text{PE}(C^*) \\
  \text{Variance} = \text{E}_D[\text{PE}(C_m) - \text{PE}(C_b)]
  \]

Bagging and variance reduction

- Under idealized condition Variance is decrease in error effected by bagging a large number of base classifier.
- Ideal situation – bootstrap samples used in bagging faithfully approximate truly independent samples.
- Poor performance in reality – ideal condition is not met in practical.
Boosting and variance reduction
- Boosting a variance reduction procedure – by Breiman
- Experiments shows boosting does more than reducing variance
- Theorem suggest different characterization
  - Poor performance of boosting
    - Insufficient training data
    - Training error become large too quickly

Averaging increases complexity
- Voting seen as smoothing or averaging a simple classification rule (the weak learner)
- Poor performance of boosting
  - Insufficient training data
  - Training error become large too quickly
- Margin based analysis gives a correct explanation – the margin is low.

Relation to SVMs
SVM: map $x$ into high-dim space, separate data linearly
- Aims to find a linear combination is high dimensional space which has large margin on the instances
- SVM- maximize the margin
- Boosting- minimize an exponential weighting of examples as function of their margin

Conclusion
- A new approach to analyze the generalization error of voted classifier
- Upper bound on prediction error
- Error is a function of empirical distribution of margin
- Boosting finds classifier with large margin
- Open problem: Does there exist a better bound
Handling Multi Class Problems

- Real World problems are generally multi-class
  - Eg. OCR problem

Some methods to deal them
- One Versus Rest
- Pair wise classification

Variant of Boosting

- Predict plausible classes
- Combined classifier chooses most frequent label from plausible sets
- Pseudoloss measure
- Overcomes the necessity of having $\frac{1}{2}$ accuracy for base classifiers
Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods

By,
Schapire, Freund, Bartlett and Lee

Comments by,
Srikanth Varanasi

AdaBoost and SVMs - differences

- Different norms can result in very different margins
  1. difference in norms may not be very significant in low dimensional spaces
  2. in high dimension spaces, difference in norms can result in very large margin difference
- When number of relevant weak hypotheses is a small fraction of total weak hypotheses – margin in AdaBoost is larger

Differences cont..

- Computation requirements are different
- Computation involved is maximizing the margin
- SVMs corresponds to quadratic programming and AdaBoost corresponds to linear programming
- Quadratic programming is more computationally demanding

Differences cont..

- A different approach is used to search efficiently in high dimensional space
- SVMs use kernels which allow to perform low dimensional calculations
- AdaBoost employs greedy search