Ensemble algorithms: ADTrees, bagging, and random forests

Sources:
T. Dietterich, Bias-Variance Analysis of Ensemble Learning.
L. Breiman, Bagging predictors (1996) and Random Forests (2001)
(ftp://ftp.stat.berkeley.edu/pub/users/breiman/)
Y. Freund and Schapire, A decision theoretic generalization of online learning and an application to boosting
(http://www1.cs.columbia.edu/~freund/papers/adaboost.ps)
Some slides from Y. Freund.

Introduction Of Graphical Models Or Decision Trees…

…may help to understand how different variables interact and what is their final impact on performance.

Review and compare few graphical models such as:

- Bagging
- Random forests, and
- Adaboosting

First, let us introduce the bias-variance analysis.

Bias-Variance Analysis in Regression

- True function is $y = f(x) + \varepsilon$
  - where ε is normally distributed with zero mean and standard deviation σ.
- Given a set of training examples, $\{(x_i, y_i)\}$, we fit an hypothesis

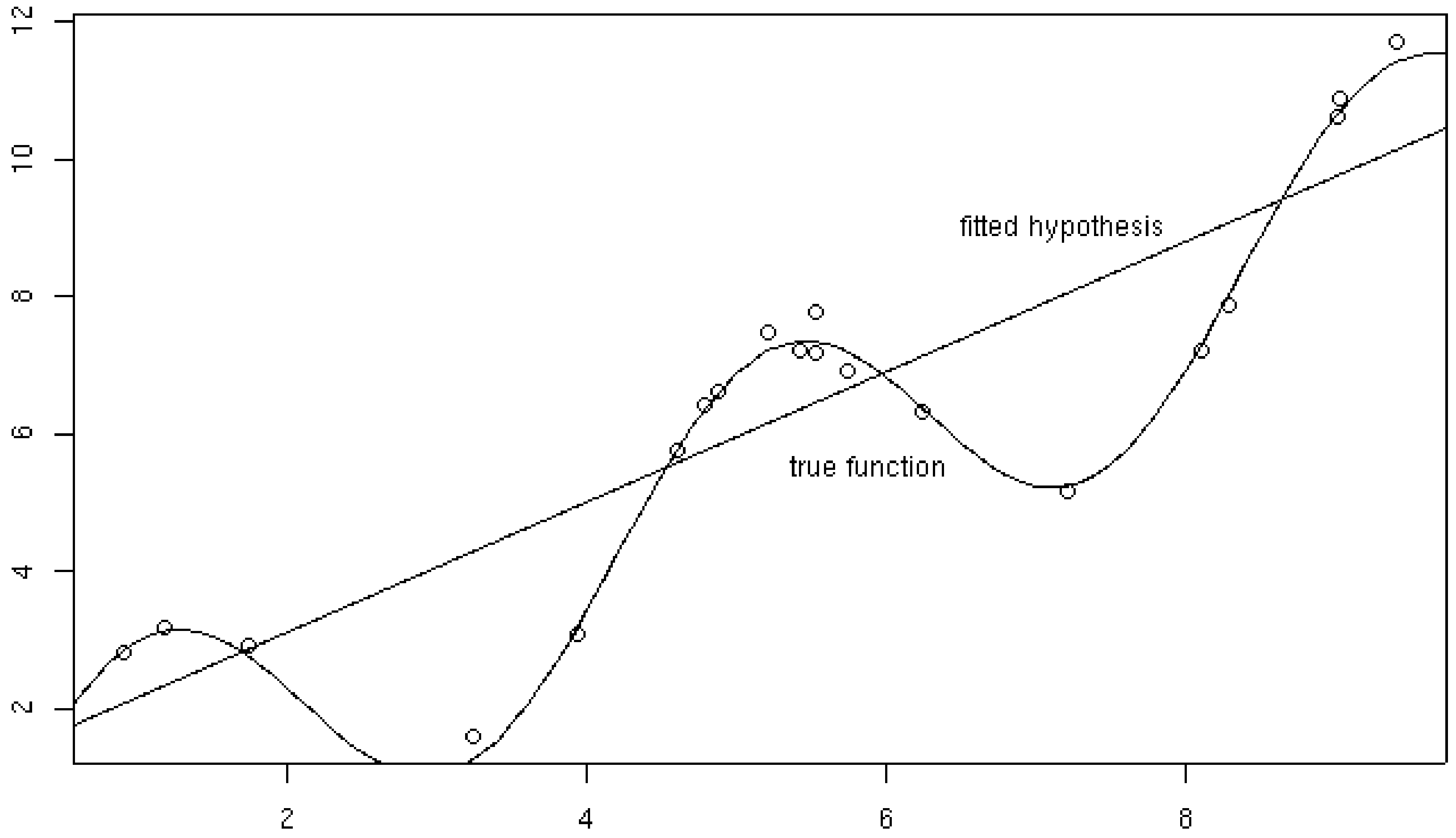
$h(x) = \alpha x + b$ to the data to minimize the squared error

$$\Sigma_i [y_i - h(x_i)]^2$$

Source: Dietterich <http://www.cs.orst.edu/~tgd>

Example: 20 points

$y = x + 2 \sin(1.5x) + N(0,0.2)$



Bias-Variance Analysis

- Now, given a new data point x* (with observed value $y^* = f(x^*) + \varepsilon$), we would like to understand the expected prediction error

$$E[\,(y^* - h(x^*))^2\,]$$

Classical Statistical Analysis

- Imagine that our particular training sample S is drawn from some population of possible training samples according to P(S).
- Compute $E_P [(y^* – h(x^*))^2]$
- Decompose this into “bias”, “variance”, and “noise”

Lemma

- Let Z be a random variable with probability distribution $P(Z)$
- Let $\underline{\underline{Z}} = E_P[\, Z \,]$ be the average value of Z.
- Lemma: $E[\, (Z - \underline{\underline{Z}})^2 \,] = E[Z^2] - \underline{\underline{Z}}^2$

  $$\begin{aligned} E[\, (Z - \underline{\underline{Z}})^2 \,] &= E[\, Z^2 - 2\, Z\, \underline{\underline{Z}} + \underline{\underline{Z}}^2 \,] \\ &= E[Z^2] - 2\, E[Z]\, \underline{\underline{Z}} + \underline{\underline{Z}}^2 \\ &= E[Z^2] - 2\, \underline{\underline{Z}}^2 + \underline{\underline{Z}}^2 \\ &= E[Z^2] - \underline{\underline{Z}}^2 \end{aligned}$$

- Corollary: $E[Z^2] = E[\, (Z - \underline{\underline{Z}})^2 \,] + \underline{\underline{Z}}^2$

Bias-Variance-Noise Decomposition

$$
\begin{aligned}
E[\,(h(x^*) - y^*)^2\,] &= E[\,h(x^*)^2 - 2\,h(x^*)\,y^* + y^{*2}\,] \\
&= E[\,h(x^*)^2\,] - 2\,E[\,h(x^*)\,]\,E[y^*] + E[y^{*2}] \\
&= E[\,(h(x^*) - \underline{h(x^*)})^2\,] + \underline{h(x^*)}^2 && \text{(lemma)} \\
&\quad - 2\,\underline{h(x^*)}\,f(x^*) \\
&\quad + E[\,(y^* - f(x^*))^2\,] + f(x^*)^2 && \text{(lemma)} \\
&= E[\,(h(x^*) - \underline{h(x^*)})^2\,] + && \text{[variance]} \\
&\quad (\underline{h(x^*)} - f(x^*))^2 + && \text{[bias}^2\text{]} \\
&\quad E[\,(y^* - f(x^*))^2\,] && \text{[noise]}
\end{aligned}
$$

Derivation (continued)

$E[\,(h(x^*) - y^*)^2\,] =$

$\quad = E[\,(h(x^*) - \underline{h(x^*)})^2\,] +$

$\quad\quad (\underline{h(x^*)} - f(x^*))^2 +$

$\quad\quad E[\,(y^* - f(x^*))^2\,]$

$\quad = \mathrm{Var}(h(x^*)) + \mathrm{Bias}(h(x^*))^2 + E[\,\varepsilon^2\,]$

$\quad = \mathrm{Var}(h(x^*)) + \mathrm{Bias}(h(x^*))^2 + \sigma^2$

Expected prediction error = Variance + Bias^2 + Noise^2

Bias, Variance, and Noise

- Variance: $E[\ (h(x^*) - \underline{h(x^*)})^2\ ]$

  Describes how much $h(x^*)$ varies from one training set S to another

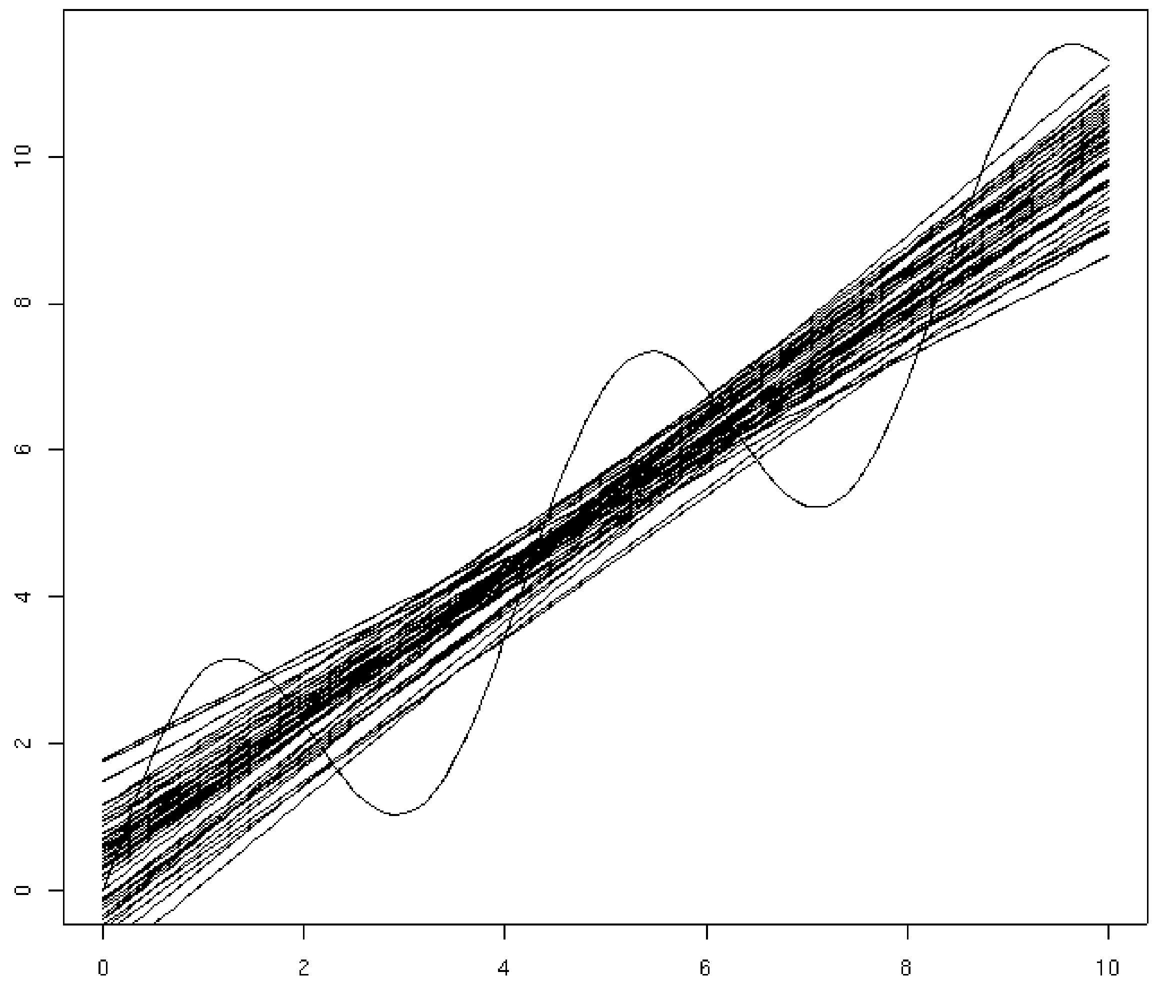
- Bias: $[\underline{h(x^*)} - f(x^*)]$

  Describes the average error of $h(x^*)$.

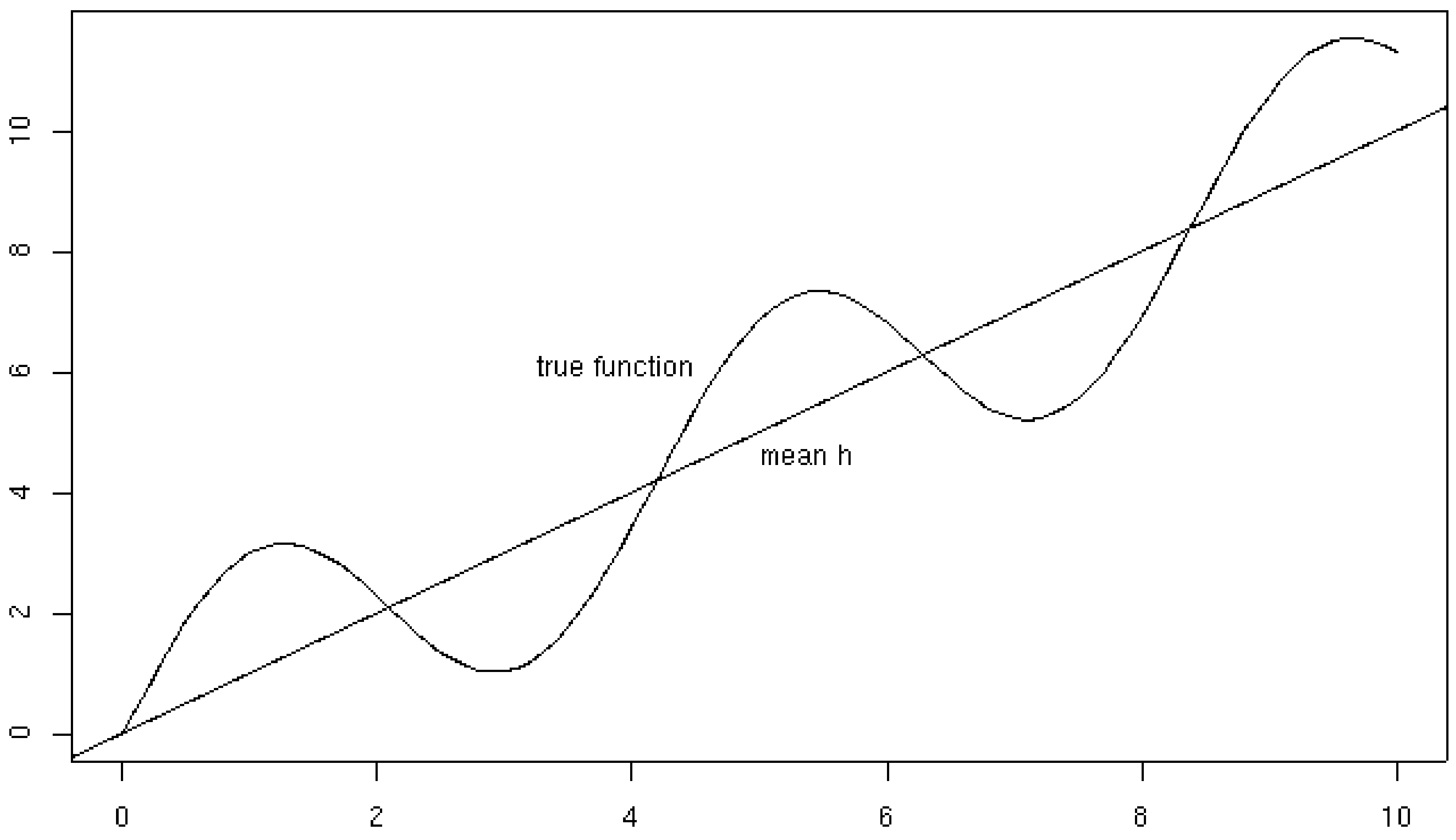
- Noise: $E[\ (y^* - f(x^*))^2\ ] = E[\varepsilon^2] = \sigma^2$

  Describes how much y^* varies from $f(x^*)$

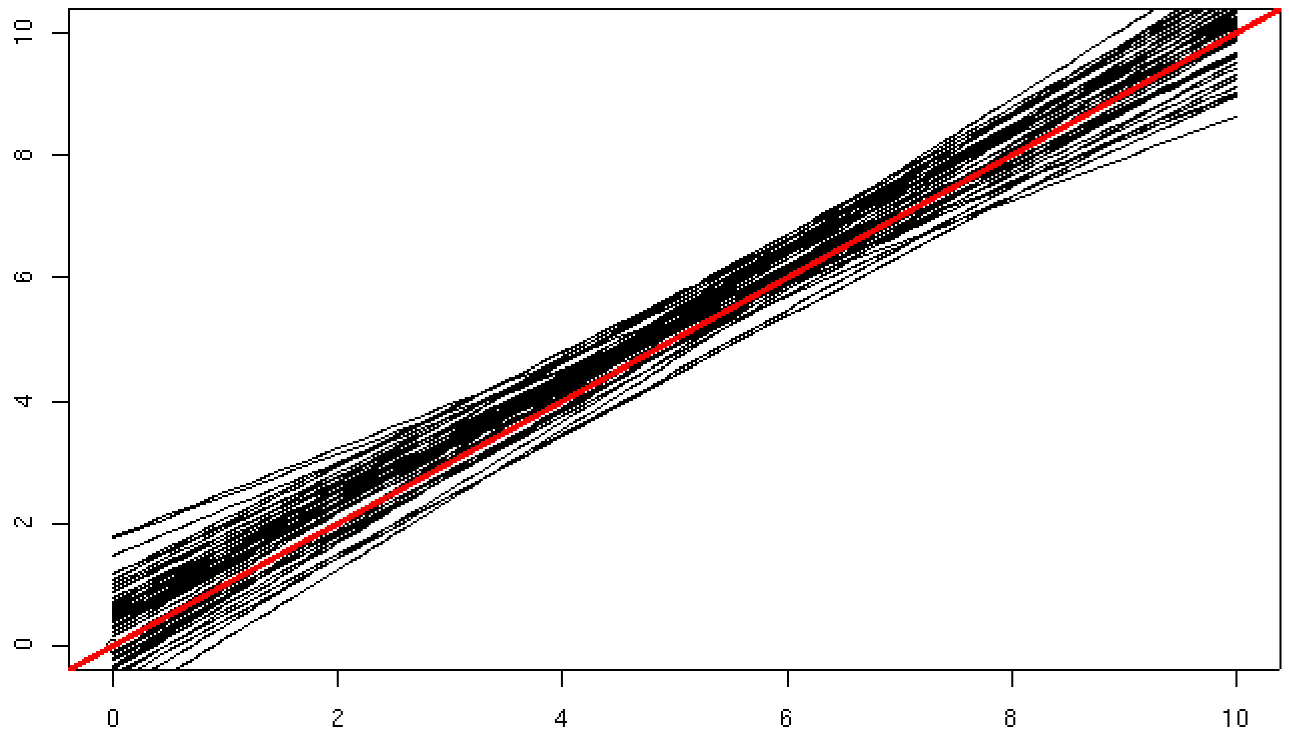
50 fits (20 examples each)



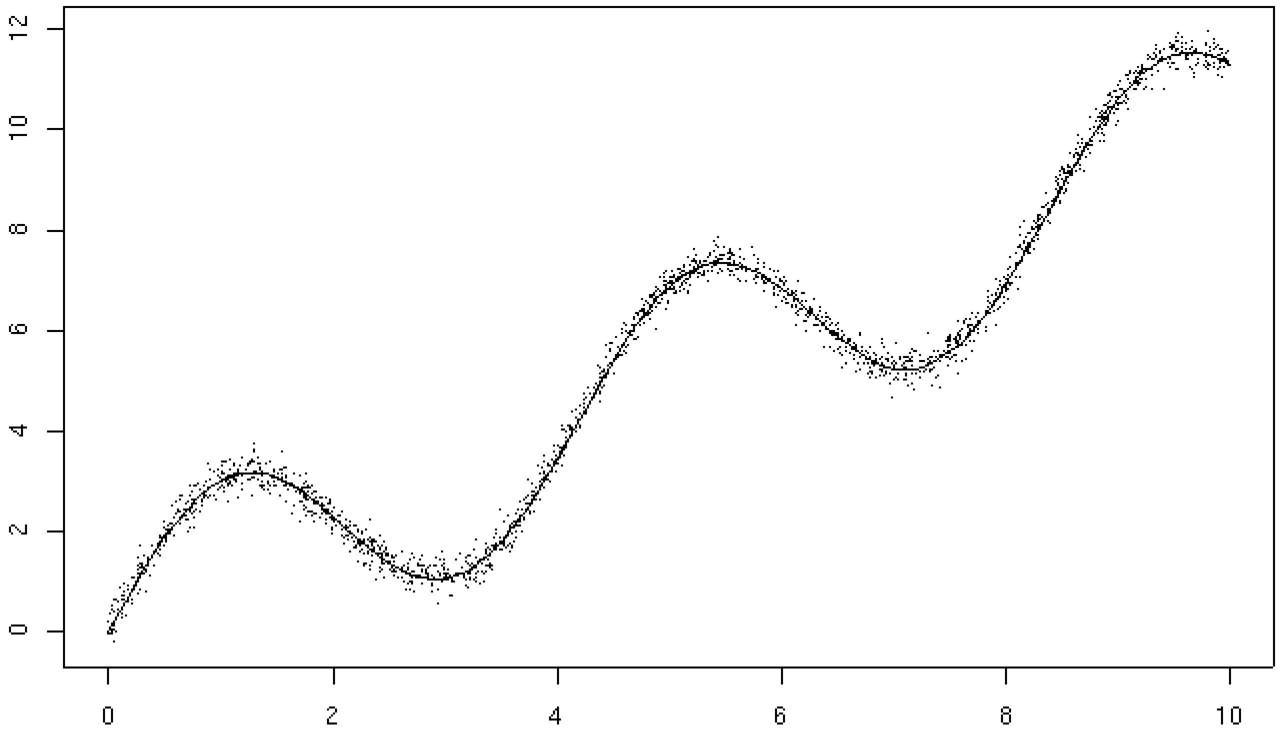
Bias



Variance



Noise



Measuring Bias and Variance

- In practice (unlike in theory), we have only ONE training set S.
- We can simulate multiple training sets by bootstrap replicates
  - S' = {x | x is drawn at random with replacement from S} and |S'| = |S|.

Procedure for Measuring Bias and Variance

- Construct B bootstrap replicates of S (e.g., B = 200): S_1, …, S_B
- Apply learning algorithm to each replicate S_b to obtain hypothesis h_b
- Let $T_b = S \setminus S_b$ be the data points that do not appear in S_b (<u>out of bag</u> points)
- Compute predicted value $h_b(x)$ for each x in T_b

Estimating Bias and Variance (continued)

- For each data point x, we will now have the observed corresponding value y and several predictions $y_1, \ldots, y_K$.
- Compute the average prediction $\underline{h}$.
- Estimate bias as $(\underline{h} - y)$
- Estimate variance as $\Sigma_k (y_k - \underline{h})^2/(K - 1)$
- Assume noise is 0

Ensemble Learning Methods

- Given training sample S
- Generate multiple hypotheses, $h_1, h_2, \ldots, h_L$.
- Optionally: determining corresponding weights $w_1, w_2, \ldots, w_L$
- Classify new points according to

$$\sum_l w_l h_l > \theta$$

Bagging: Bootstrap Aggregating (Breiman 1996)

- For b = 1, …, B do
  - S_b = bootstrap replicate of S
  - Apply learning algorithm to S_b to learn h_b
- Classify new points by unweighted vote:
  - $[\sum_b h_b(x)]/B > 0$

Bagging

- If y is numerical, bagging makes predictions according to
  $y = \Sigma_b h_b(x) / B$
- Hence, bagging's predictions are $\underline{h}(x)$

- If y is class label bagging makes predictions according to voting.

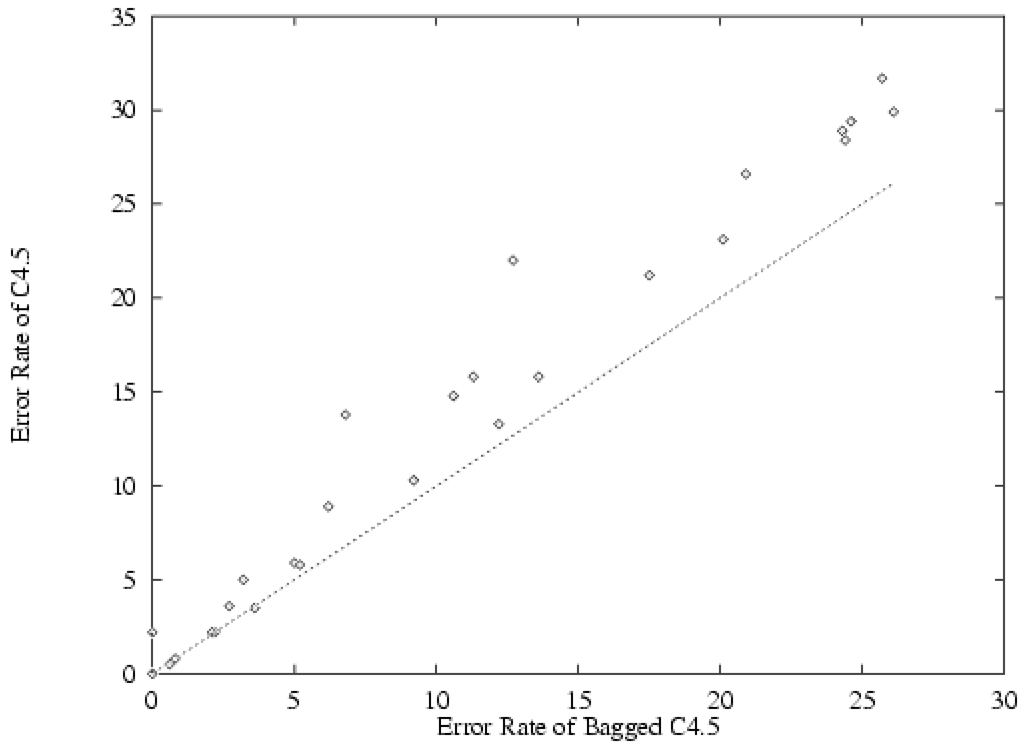
Estimated Bias and Variance of Bagging

- If we estimate bias and variance using the same B bootstrap samples, we will have:
  - Bias = $(\underline{h} - y)$ [same as before]
  - Variance = $\Sigma_k (\underline{h} - \underline{h})^2/(K - 1) = 0$
- Hence, according to this approximate way of estimating variance, bagging removes the variance while leaving bias unchanged.
- In reality, bagging only *reduces* variance and tends to slightly increase bias

Bias/Variance Heuristics

- Models that fit the data poorly have high bias: “inflexible models” such as linear regression, regression stumps
- Models that can fit the data very well have low bias but high variance: “flexible” models such as nearest neighbor regression, regression trees, and Adaboosting
- This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

Bagging Decision Trees (Freund & Schapire 1996)



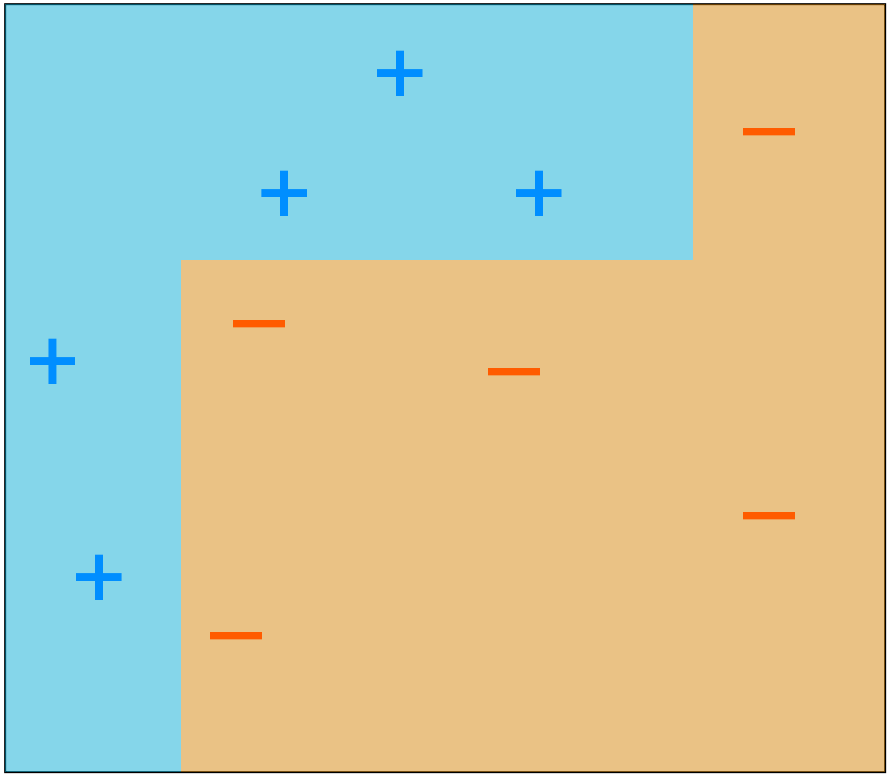
Boosting (Schapire 2002, Freund and Schapire 1997) is

- … a method for improving classifier accuracy
- Basic idea:
    - Perform iterative search to locate the regions/ examples that are more difficult to predict.
    - Through each iteration reward accurate predictions on those regions.
    - Combines the rules from each iteration.
- Only requires that the underlying learning algorithm be better than guessing.

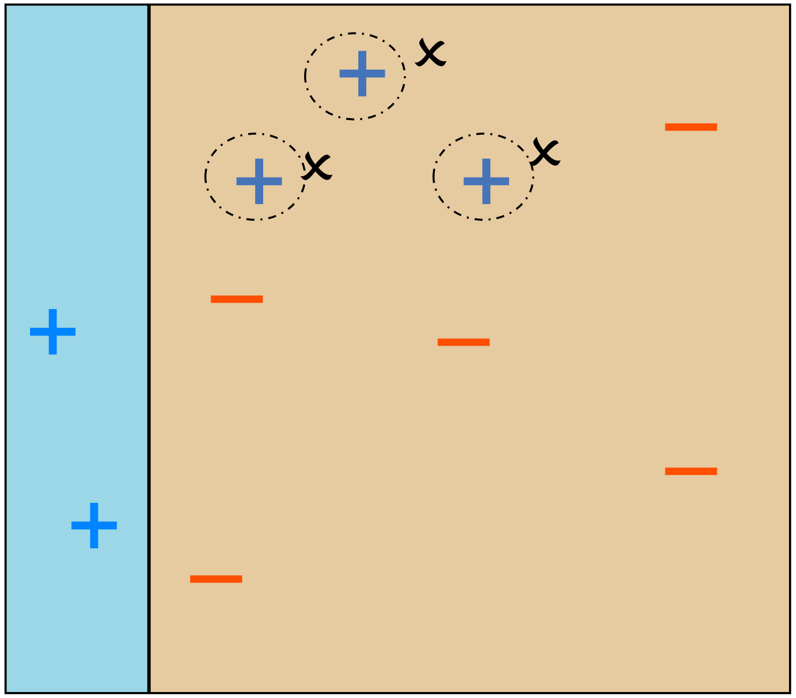
History of Boosting

- "Kearns & Valiant (1989) proved that learners performing only slightly better than random, can be combined to form an arbitrarily good ensemble hypothesis."
- Schapire (1990) provided the first polynomial time Boosting algorithm.
- Freund (1995) “Boosting a weak learning algorithm by majority”
- Freund & Schapire (1997) AdaBoost. Solved many practical problems of boosting algorithms. “Ada” stands for adaptive.

Example of a Good Classifier



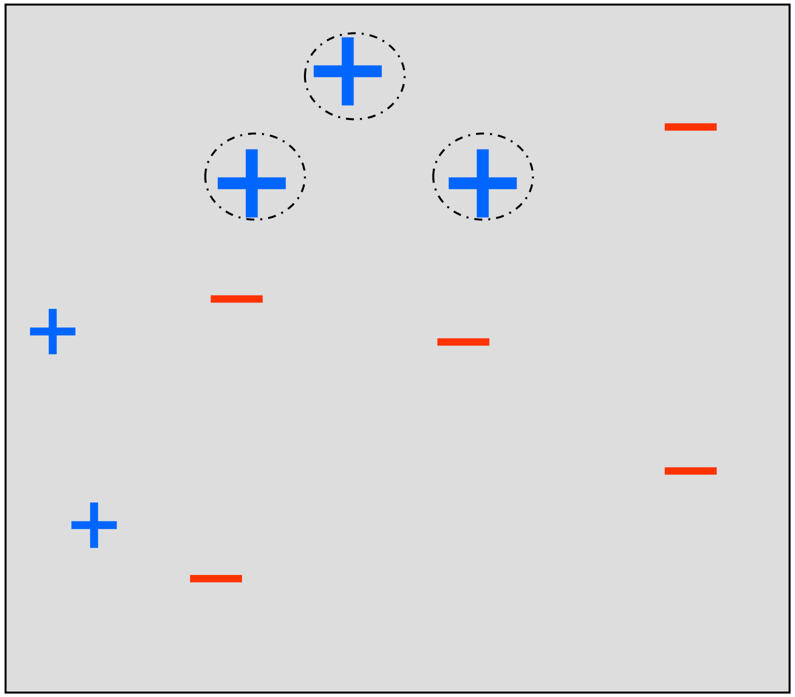
Round 1 of 3



h_1

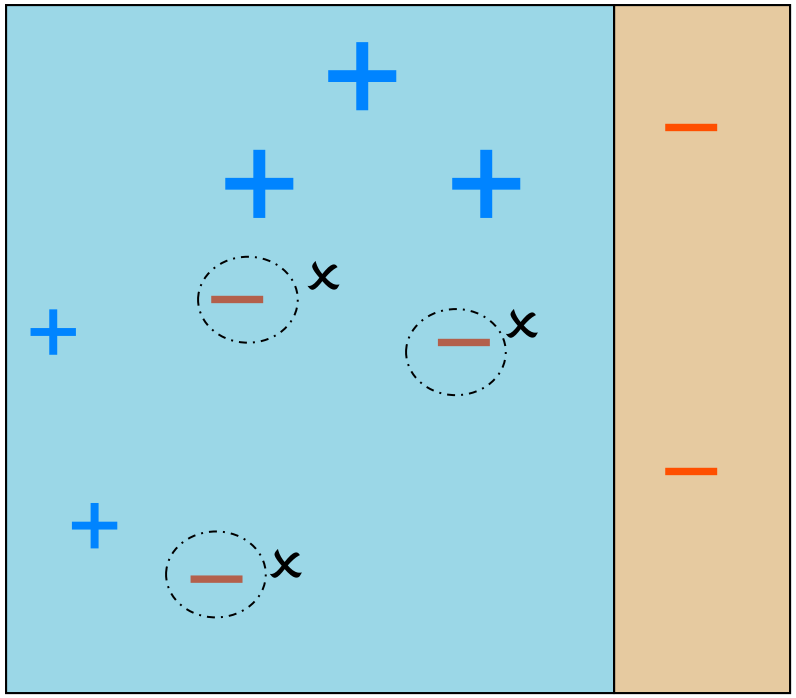
$\varepsilon_1 = 0.300$

$\alpha_1 = 0.424$



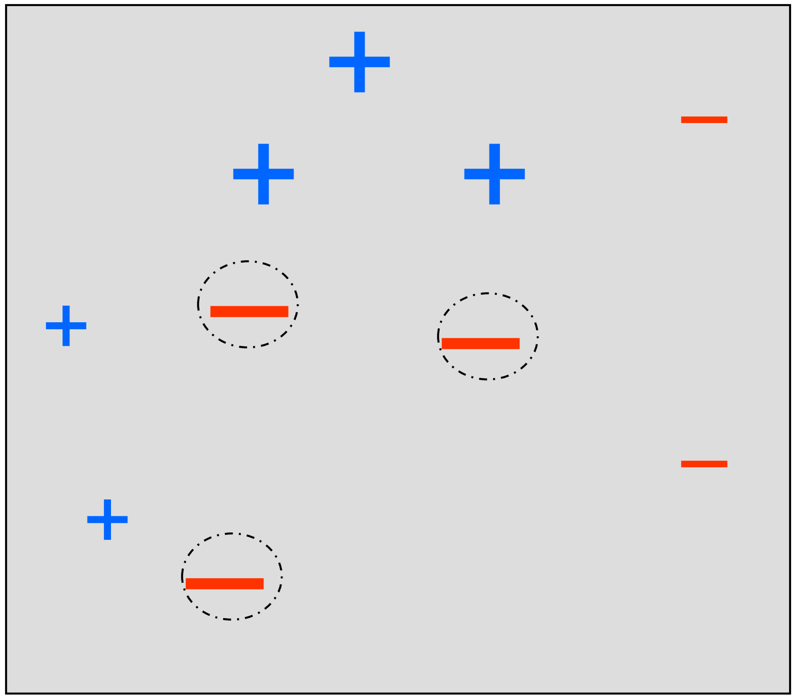
D_2

Round 2 of 3



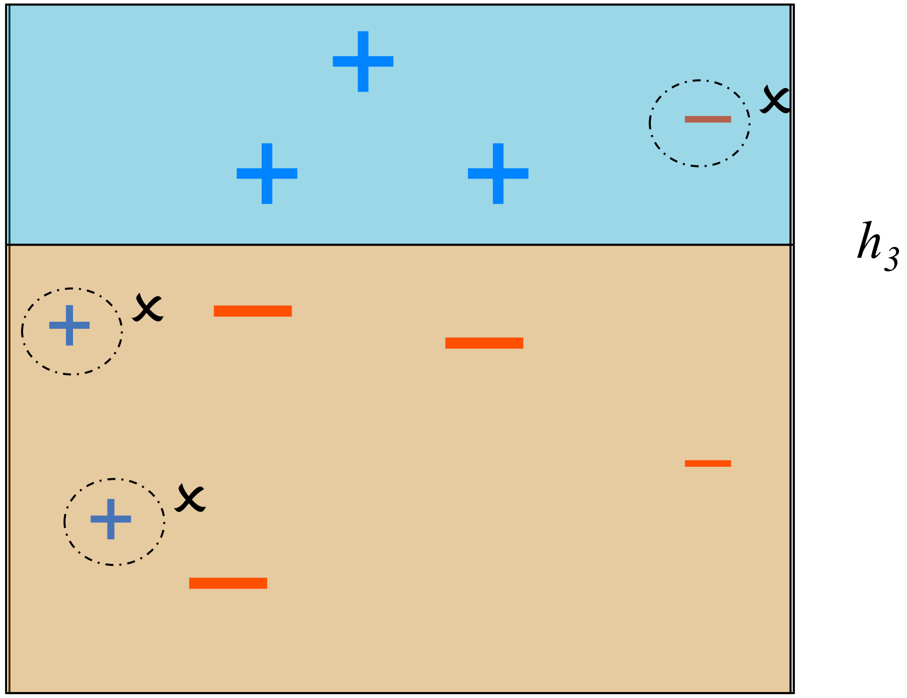
$\varepsilon_2 = 0.196$

$\alpha_2 = 0.704$



D_2

Round 3 of 3

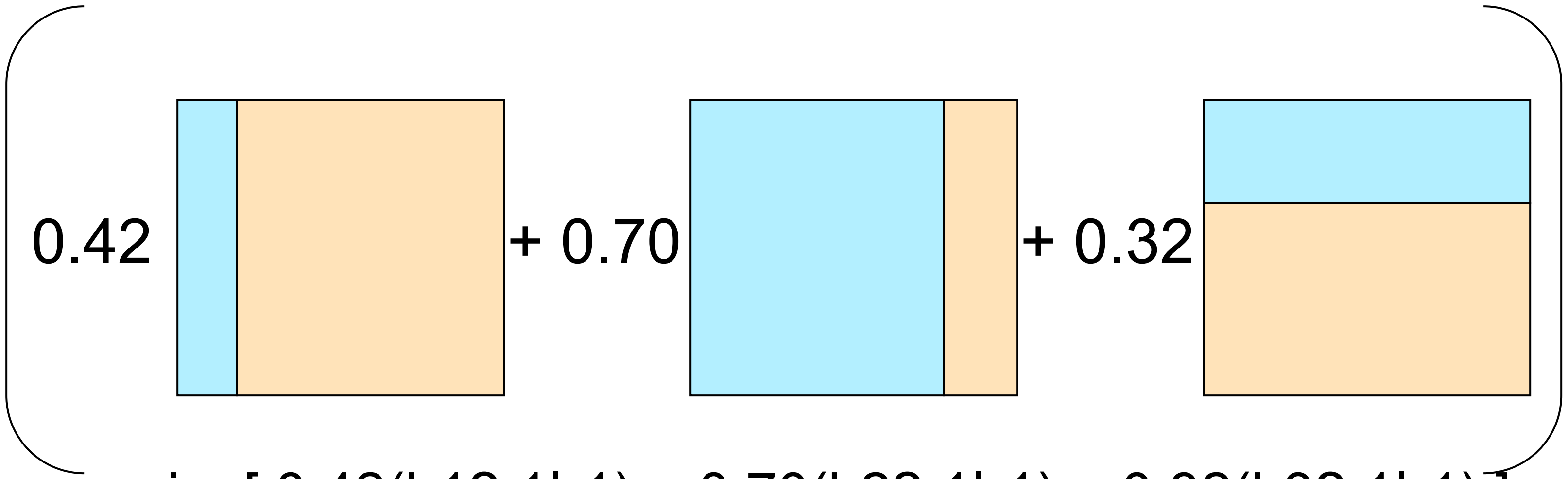


STOP

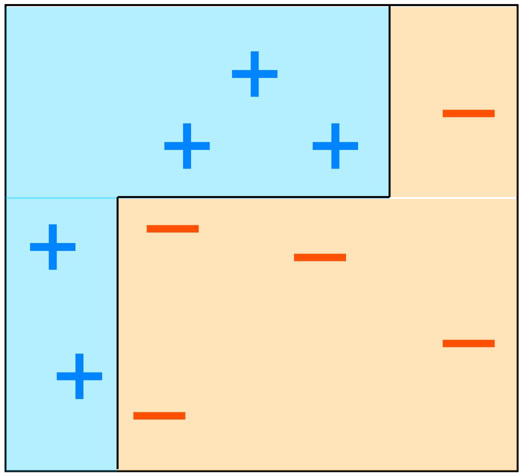
ε_3 = 0.344

α_2=0.323

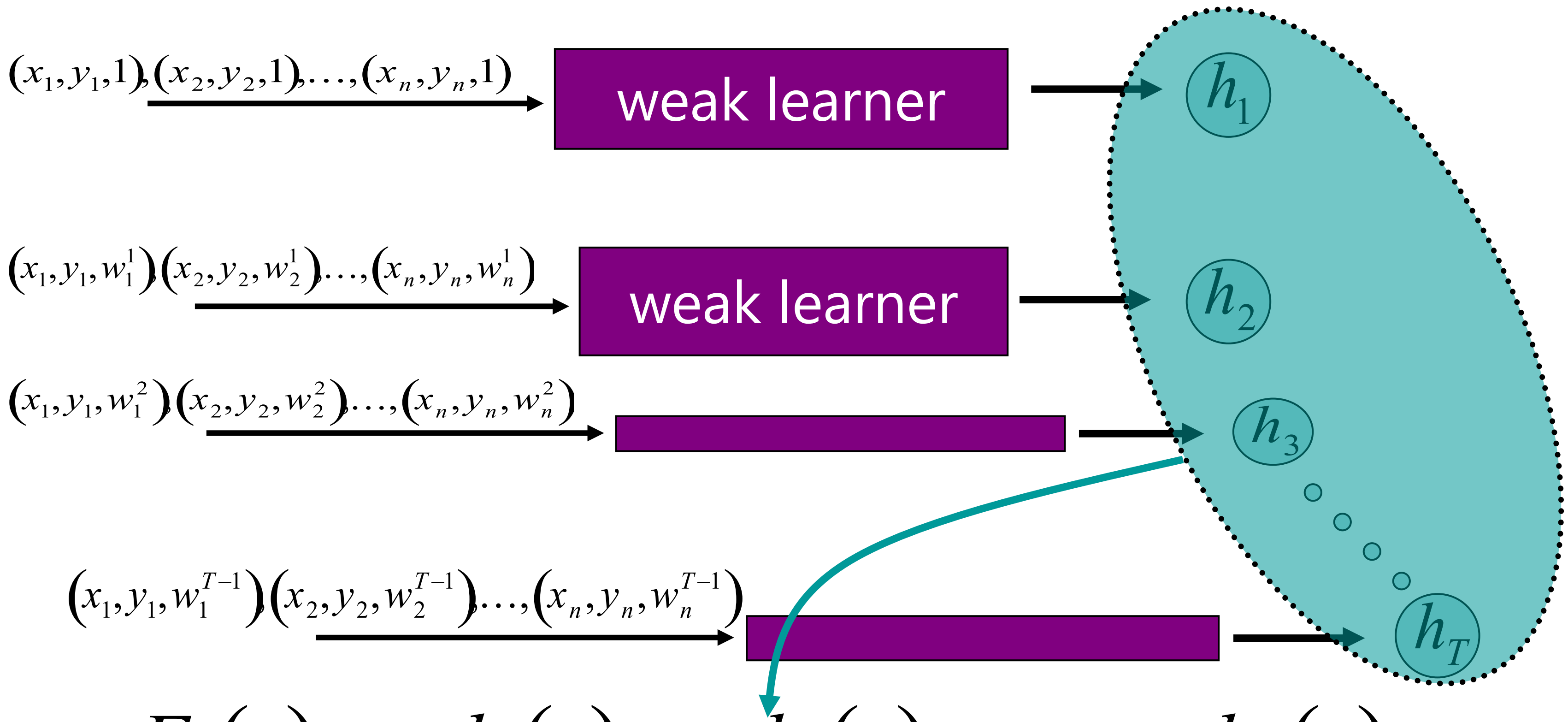
Final Hypothesis



$$H_{final} = \text{sign}[\ 0.42(h1?\ 1|-1) + 0.70(h2?\ 1|-1) + 0.32(h3?\ 1|-1)\ ]$$



The boosting process (Freund & Schapire 1997)



$(x_1,y_1,1),(x_2,y_2,1),\ldots,(x_n,y_n,1)$
weak learner
h_1
$(x_1,y_1,w_1^1),(x_2,y_2,w_2^1),\ldots,(x_n,y_n,w_n^1)$
weak learner
h_2
$(x_1,y_1,w_1^2),(x_2,y_2,w_2^2),\ldots,(x_n,y_n,w_n^2)$
h_3
$(x_1,y_1,w_1^{T-1}),(x_2,y_2,w_2^{T-1}),\ldots,(x_n,y_n,w_n^{T-1})$
h_T

$$F_T(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_T h_T(x)$$

Final rule: $f_T(x) = \text{sign}(F_T(x))$

The Adaboost algorithm

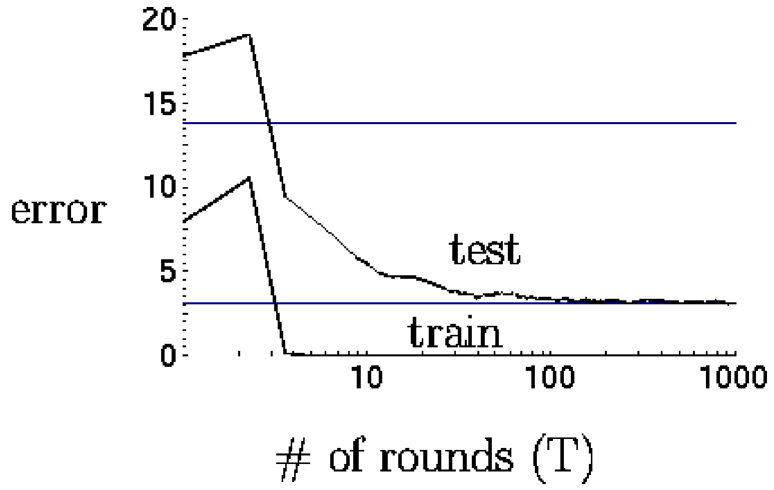
```
$F_0(x) \equiv 0$
for $t = 1 \ldots T$
        $w_i^t = e^{-y_i F_{t-1}(x_i)}$
        Get $h_t$ from *weak learner*
        $\alpha_t = \frac{1}{2} \ln \left( \frac{\sum_{i:h_t(x_i)=1, y_i=1} w_i^t}{\sum_{i:h_t(x_i)=1, y_i=-1} w_i^t} \right)$
        $F_{t+1} = F_t + \alpha_t h_t$
```

y_i is the binary label to be predicted; x_i corresponds to the features of an instance i, w_i^t is the weight of instance i at time t, h_t and $F_t(x)$ are the prediction rule and the prediction score at time t respectively

No Overfitting

- Curious phenomenon
  - For graph “Using <10,000 training examples we fit >2,000,000 parameters”
- Expected to overfit
  - First bound on generalization error rate implies that overfit may occur as T gets large
- Does not
  - Empirical results show the generalization error rate still decreasing after the training error has reached zero.
  - Resistance explained by “margin”

Accuracy Change per Round



Shortcomings

- Actual performance of boosting can be:
    - dependent on the data and the weak learner
- Boosting can fail to perform when:
    - Insufficient data
    - Overly complex weak hypotheses
    - Weak hypotheses which are too weak
- Susceptible to noise

ADTrees (Freund and Mason 99)

The alternating decision tree learning algorithm:

- Finds best constant prediction for the entire data set (score at root)
- Grows the tree iteratively, adding one base rule at a time using any boosting algorithm.
- Added base rule: subtree with decision node as its root and 2 prediction nodes as leaves.

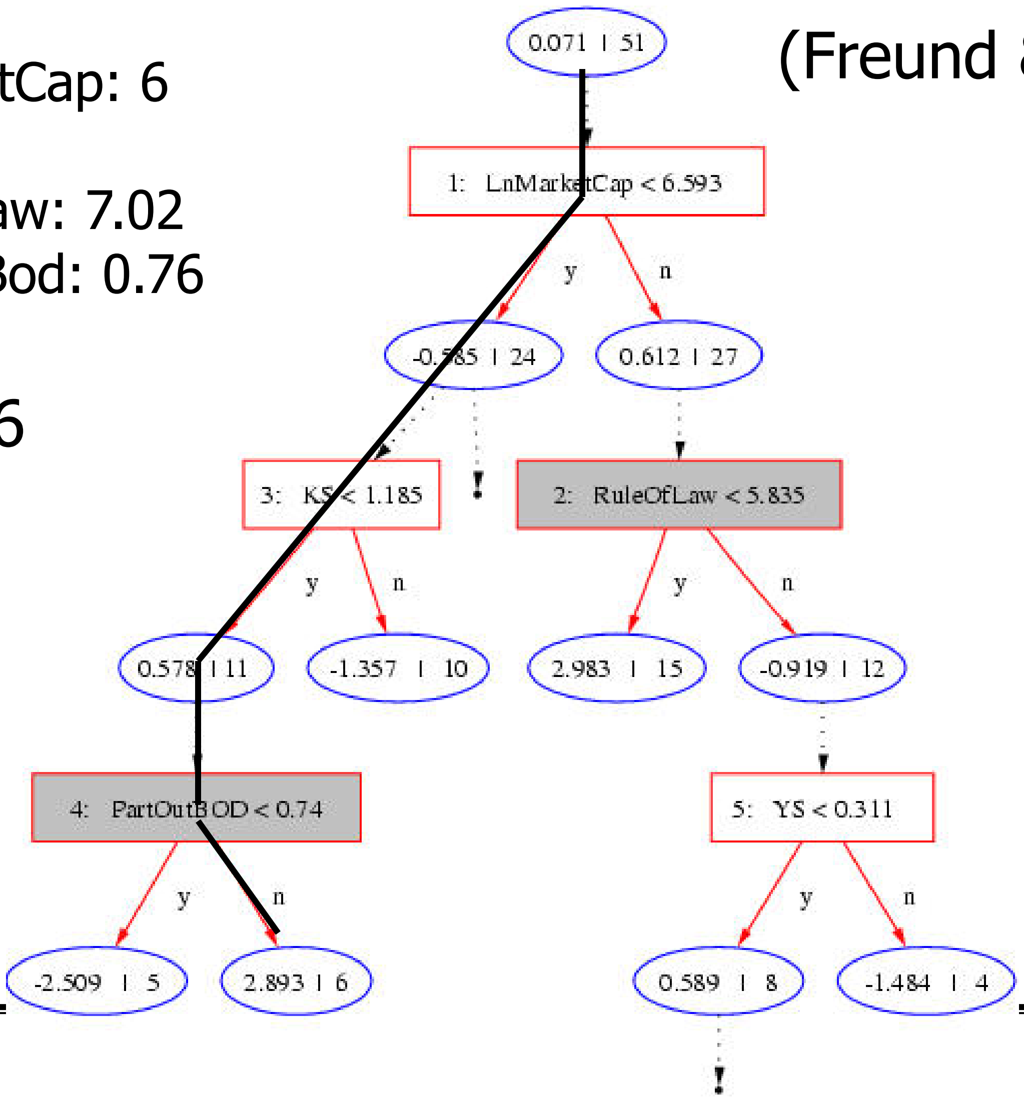
- The set of rules are equivalent to the “weak hypotheses”, and they increase as the tree grows.

An alternating decision tree (ADT) for corporate decisions

LnMarketCap: 6
KS: 0.86
RuleOfLaw: 7.02
PartOutBod: 0.76

F= 2.96

(Freund & Mason, 1999)



Bias-Variance Analysis of Boosting

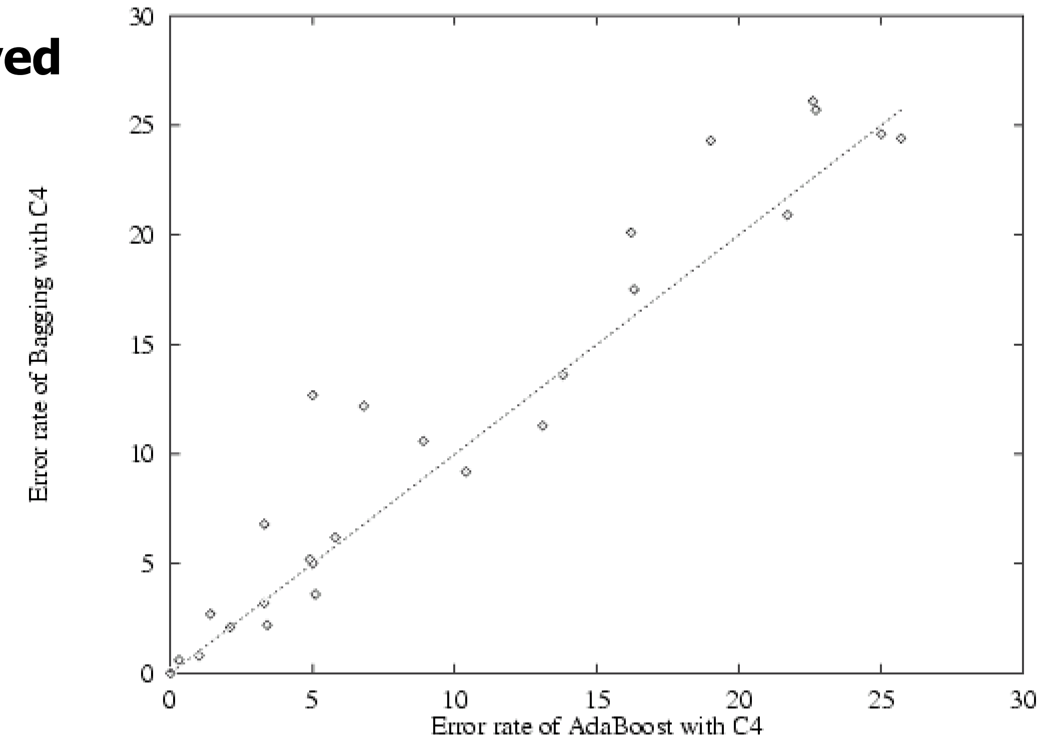
- Boosting seeks to find a weighted combination of classifiers that fits the data well
- Prediction: Boosting will primarily act to reduce bias

Difference between Bagging and Boosting (Freund and Schapire 1996)

- Bagging always uses resampling rather than reweighting;
- Bagging does not modify the distribution over examples or mislabels, but instead always uses the uniform distribution;
- In forming the final hypothesis, bagging gives equal weight to each of the weak hypotheses

Boosting vs Bagging (Freund & Schapire 1996)

Boosting improved error rate by 24.8%
Bagging by 20%



Sources of Bias and Variance

- Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data
- Variance arises when the classifier overfits the data
- There is often a tradeoff between bias and variance

Effect of Bagging

- If the bootstrap replicate approximation were correct, then bagging would reduce variance without changing bias
- In practice, bagging can reduce both bias and variance
    - For high-bias classifiers, it can reduce bias
    - For high-variance classifiers, it can reduce variance

Effect of Boosting

- In the early iterations, boosting is primarily a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method

Random Forests (Breiman 01)

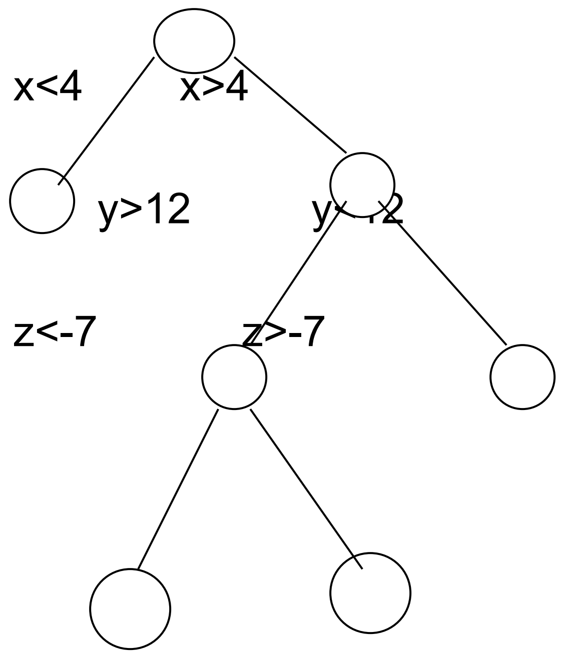
- Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest.
    - Each tree gives a classification
    - Forest chooses the classification having the most votes (over all the trees in the forest).

Random Forests - Algorithm

- Initially select the number of trees to be generated
- At step k
- Construct tree h(x, $@_k$) where:
  - Observations are selected based on a random vector $@_k$ that represents a “Bootstrap” sample (sample with replacement) of the original dataset
  - A given number of m variables are randomly selected using any decision tree algorithm
    - If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing
    - No Pruning
- Each tree casts 1 vote for the most popular class at X
- The class at X is predicted by selecting the class with maximum votes among all the trees

Example

Variables: {a,x,b,y,z,c,……..} –x,y,z



Example

- A 2 class problem {0,1}

  No. of trees = 1000

  Consider an Input X

  $P = \sum I(h(x, @_k) = 1) = 600$

  $Q = \sum I(h(x, @_k) = 0) = 400$

  P>Q => R. Forest classifies X as 1

Advantages – Random Forests

- Upper bound on error
- Do not overfit data
- Robust to noise
- Fastest tree-based technique
- Not very sensitive to node split variable

- Gives Internal estimates like :

  - Error
  - Strength
  - Correlation

Disadvantages - RF

- Mechanism hard to interpret i.e how variables interact, etc

  -Solution:- Rank variables by their capacity to reduce impurity weighted by the probability to reach that node.

Random Forests - Regression

- Each tree predicts Numerical Values
- Score Function – Mean Squared Error
- Instead of Voting at X, Average at X is taken

Conclusions – Random Forests

- Combination of Tree predictors
- Trees vote to elect a class at every X
- No over fitting of data
- Test data is generated from training data

Comparing bagging, boosting & others

- Dietterich (1998) compares 3 methods for constructing ensemble classifiers using C4.5:
  - Randomizing
  - Bagging, and
  - Boosting

- Use t-test 95% confidence interval for difference in error rates of algorithms

Comparing bagging, boosting & others

- Without noise:
  - Boosting best results in most cases
  - Randomizing & bagging similar

- With noise:
  - Bagging is best method
  - Boosting is affected by classification noise

Comparing bagging, boosting & others

Bauer and Kohavi (1999) compares methods using variance bias decomposition:

- Boosting: reduced both bias and variance of unstable methods but increased variance for Naïve-Bayes, which was stable
- Bagging: reduced variance of unstable methods

Again, this study shows that Adaboost does not deal well with noise.