Bias/Variance Tradeoff and Ensemble Methods

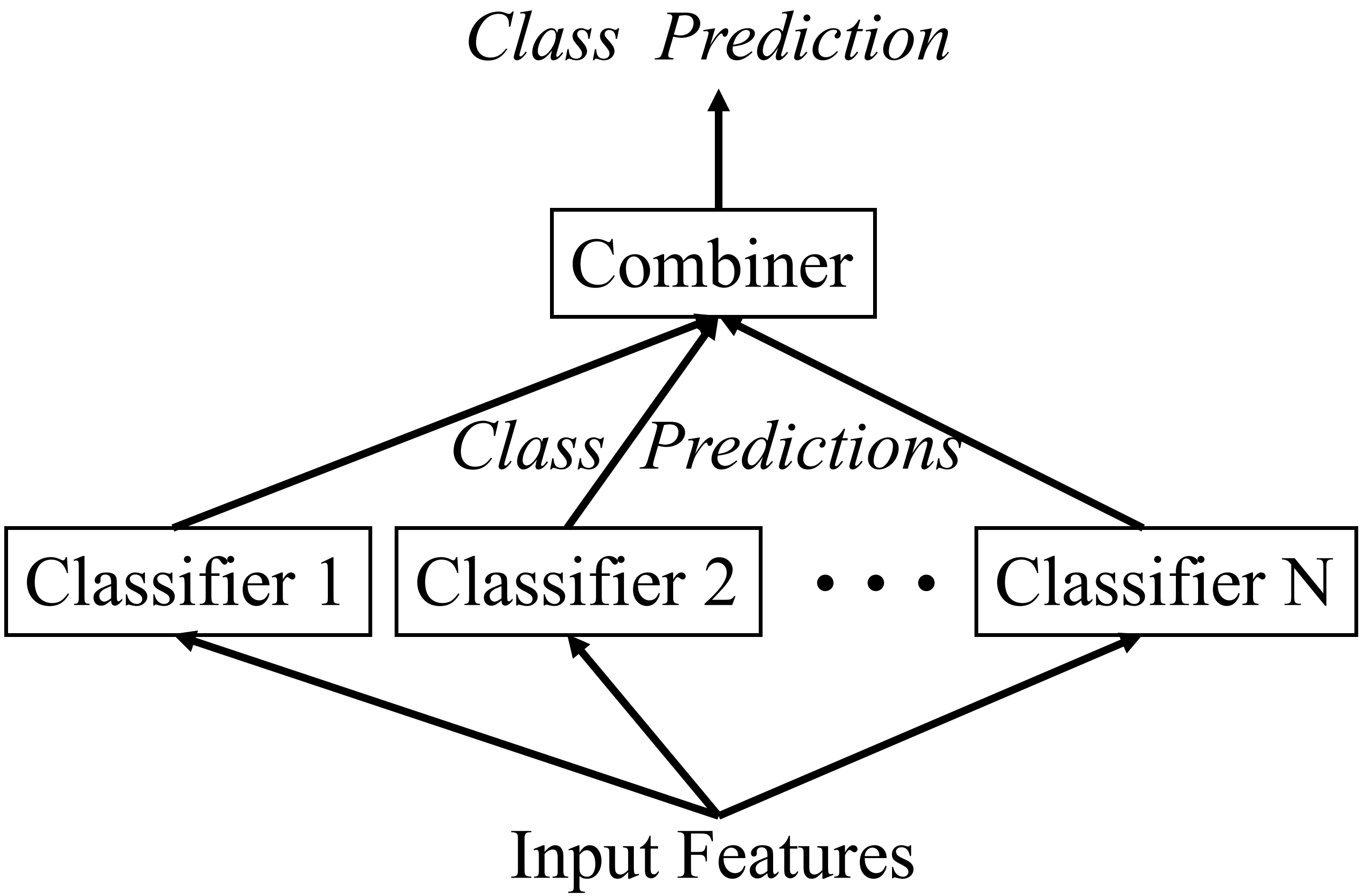
Vibhav Gogate
The University of Texas at Dallas
Machine learning
CS 6375

Slide courtesy of Tom Dietterich and Vincent Ng

Outline

- Bias-Variance Decomposition for Regression
- Ensemble Methods
  - Bagging
  - Boosting
- Summary and Conclusion

A Classifier Ensemble



Intuition 1

- The goal in learning is not to learn an exact representation of the training data itself, but to build a statistical model of the process which generates the data. This is important if the algorithm is to have good generalization performance
- We saw that
  - models with too few parameters can perform poorly
  - models with too many parameters can perform poorly
- Need to optimize the complexity of the model to achieve the best performance
- One way to get insight into this tradeoff is the decomposition of generalization error into bias^2 + variance
  - a model which is too simple, or too inflexible, will have a large bias
  - a model which has too much flexibility will have high variance

Intuition

- bias:
  - measures the accuracy or quality of the algorithm
  - high bias means a poor match

- variance:
  - measures the precision or specificity of the match
  - a high variance means a weak match

- We would like to minimize each of these

- Unfortunately, we can't do this independently, there is a trade-off

Bias-Variance Analysis in Regression

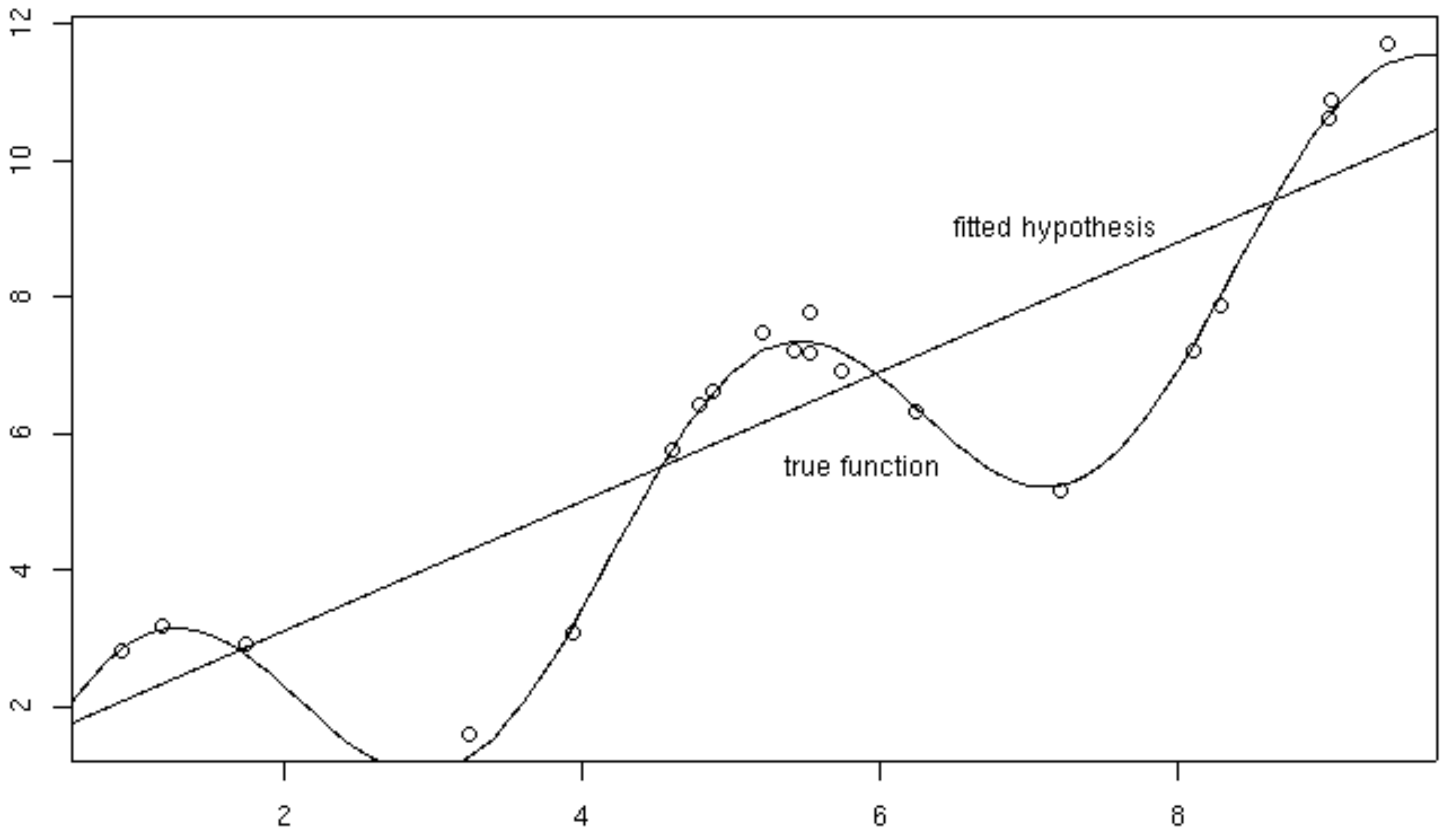
True function is $y = f(x) + \epsilon$

where ϵ is normally distributed with zero mean and standard deviation σ

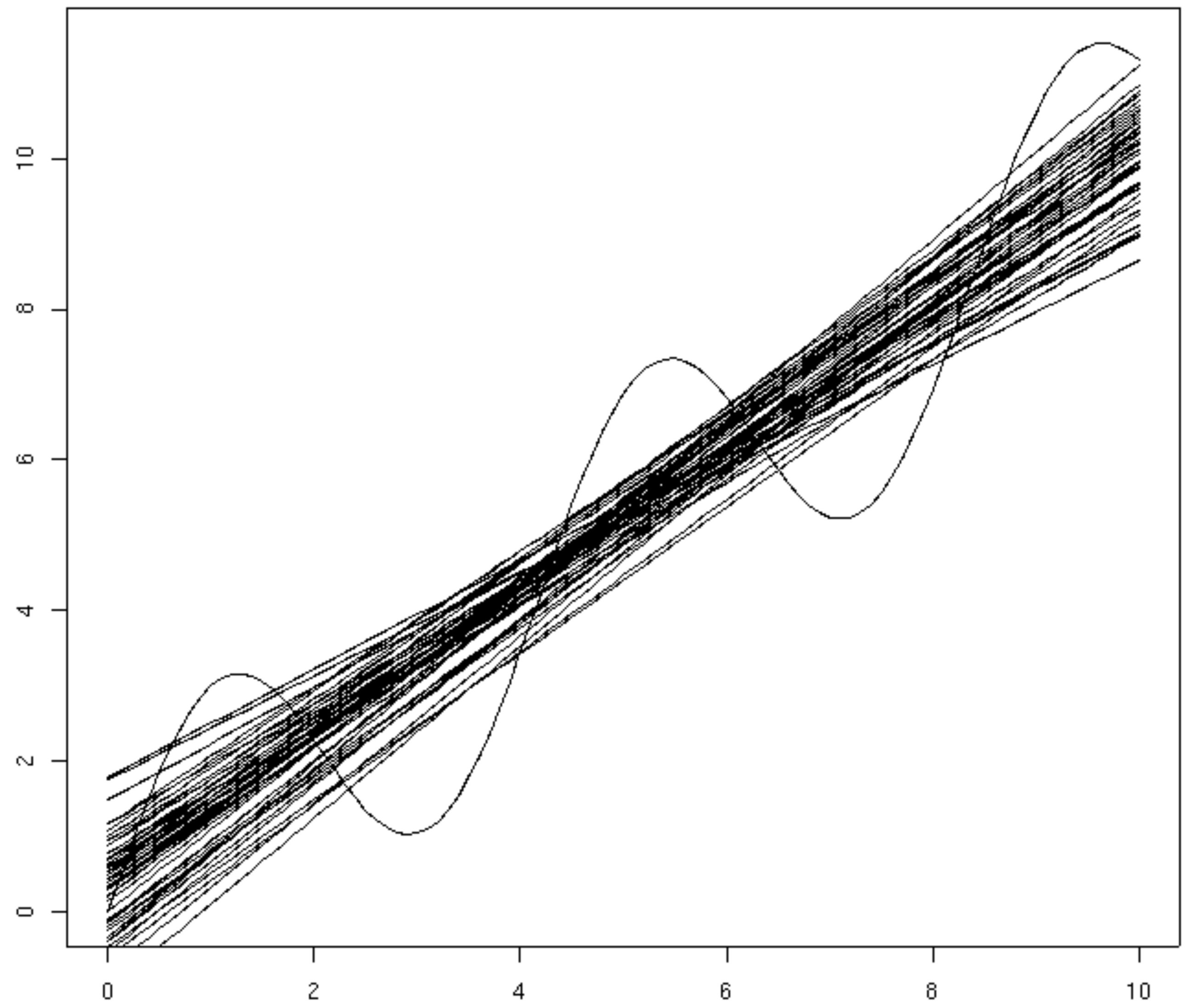
Given a set of training examples $\{x_i, y_i\}$ we fit an hypothesis $h(x) = w^T x + b$ to the data to minimize the squared error

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

Example: 20 points
y = x + 2 sin(1.5x) + N(0,0.2)



50 fits (20 examples each)



Bias-Variance Analysis

Given a new data point x^* (with observed value $y^* = f(x^*) + \epsilon$)
we would like to understand the expected prediction error

$$\mathbb{E}\left[\left(y^* - h(x^*)\right)^2\right]$$

Bias-Variance-Noise Decomposition

$$\begin{aligned}\mathbb{E}\left[(y^* - h(x^*))^2\right] &= \mathbb{E}\left[(y^*)^2 - 2h(x^*)y^* + h(x^*)^2\right] \\ &= \mathbb{E}[h(x^*)^2] - 2\mathbb{E}[h(x^*)]\mathbb{E}[y^*] + \mathbb{E}[(y^*)^2]\end{aligned}$$

We know that variance is given by

$$\mathbb{E}[(Z - \mathbb{E}[Z])^2] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2$$

Rewriting the equation above, we get

$$\mathbb{E}[Z^2] = \mathbb{E}[(Z - \mathbb{E}[Z])^2] + \mathbb{E}[Z]^2$$

Bias-Variance-Noise Decomposition

Note: $y^* - f(x^*) = \epsilon$
and $\mathbb{E}[y^*] = f(x^*)$

Using the following formula

$$\mathbb{E}[Z^2] = \mathbb{E}[(Z - \mathbb{E}[Z])^2] + \mathbb{E}[Z]^2$$

in the Equation for expected prediction error

$$\begin{aligned}
&\mathbb{E}[h(x^*)^2] - 2\mathbb{E}[h(x^*)]\mathbb{E}[y^*] + \mathbb{E}[(y^*)^2] \\
&= \mathbb{E}[(h(x^*) - \mathbb{E}[h(x^*)])^2] + \mathbb{E}[h(x^*)]^2 \\
&\quad -2\mathbb{E}[h(x^*)]f(x^*) \\
&\quad +\mathbb{E}[(y^* - f(x^*))^2] + f(x^*)^2 \\
&= \mathbb{E}[(h(x^*) - \mathbb{E}[h(x^*)])^2] \ldots \text{Variance} \\
&\quad +(\mathbb{E}[h(x^*)] - f(x^*))^2 \ldots \text{Bias} \\
&\quad +\mathbb{E}[\epsilon^2] \ldots \text{Noise}
\end{aligned}$$

Bias-Variance-Noise Decomposition

Note: $y^* - f(x^*) = \epsilon$
and $\mathbb{E}[y^*] = f(x^*)$

Using the following formula

$$\mathbb{E}[Z^2] = \mathbb{E}[(Z - \mathbb{E}[Z])^2] + \mathbb{E}[Z]^2$$

in the Equation for expected prediction error

$$\mathbb{E}[h(x^*)^2] - 2\mathbb{E}[h(x^*)]\mathbb{E}[y^*] + \mathbb{E}[(y^*)^2]$$

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

$$-2\mathbb{E}[h(x^*)]f(x^*)$$

$$+\mathbb{E}[(y^* - f(x^*))^2] + f(x^*)^2$$

$$= \mathbb{E}[(h(x^*) - \mathbb{E}[h(x^*)])^2] \ldots \text{Variance}$$

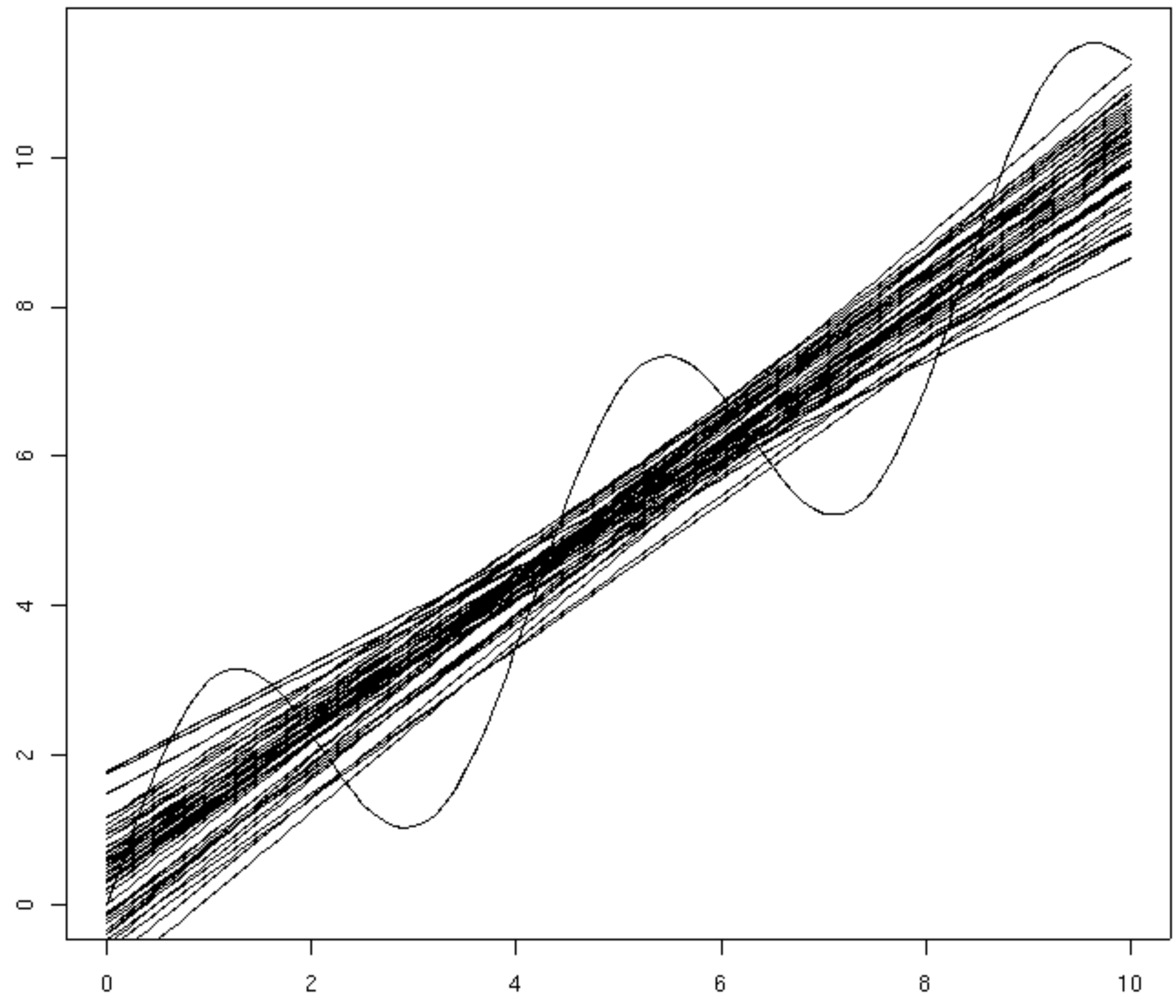
$$+(\mathbb{E}[h(x^*)] - f(x^*))^2 \ldots \text{Bias}$$

$$+\mathbb{E}[\epsilon^2] \ldots \text{Noise}$$

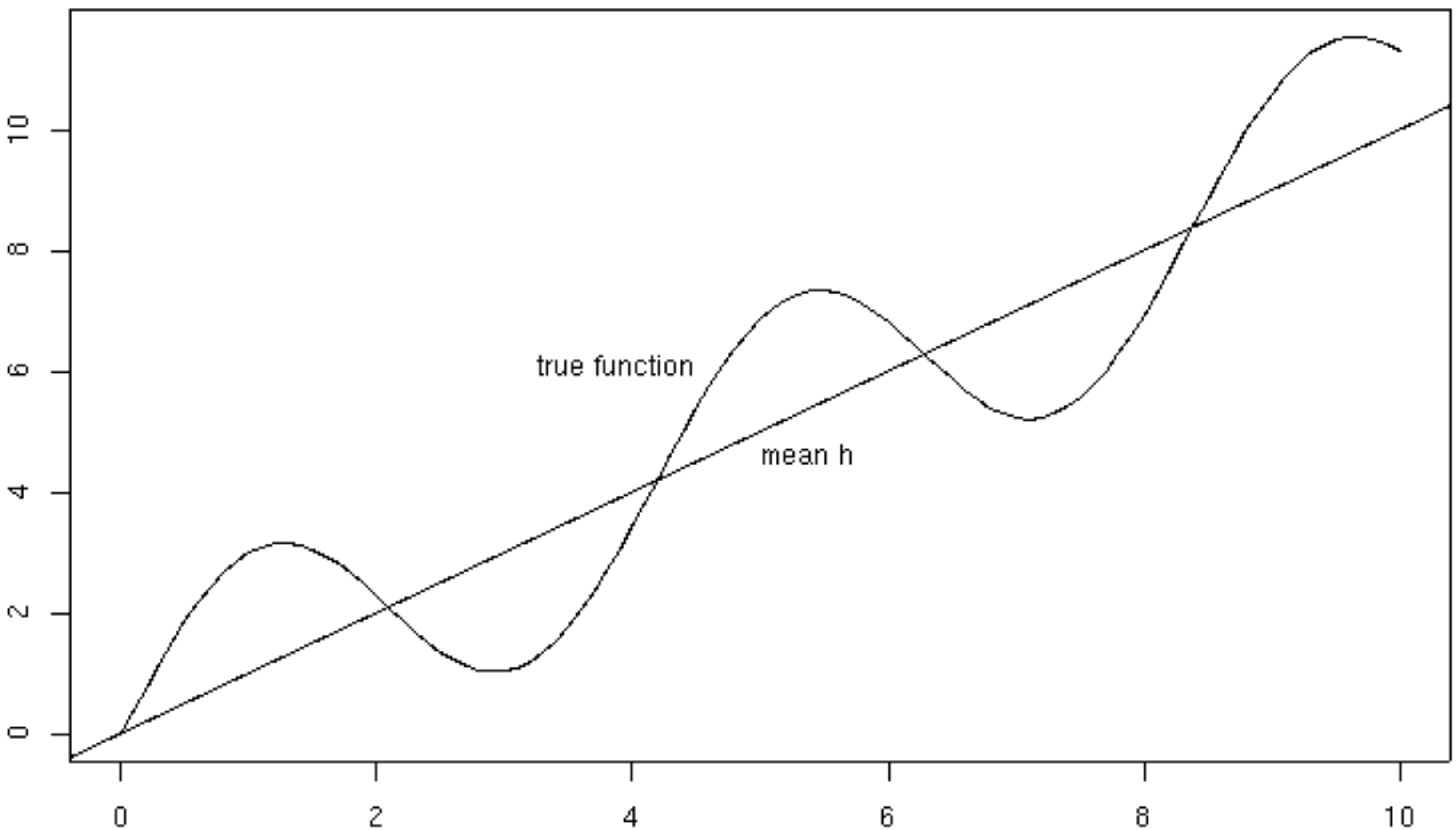
Bias, Variance, and Noise

- Prediction Error = Bias-squared + Variance + Noise.
- Variance: Describes how much the hypothesis "h" varies from one dataset to another
- Bias: Describes the average error of "h"
- Noise: Describes how much y varies from f(x)

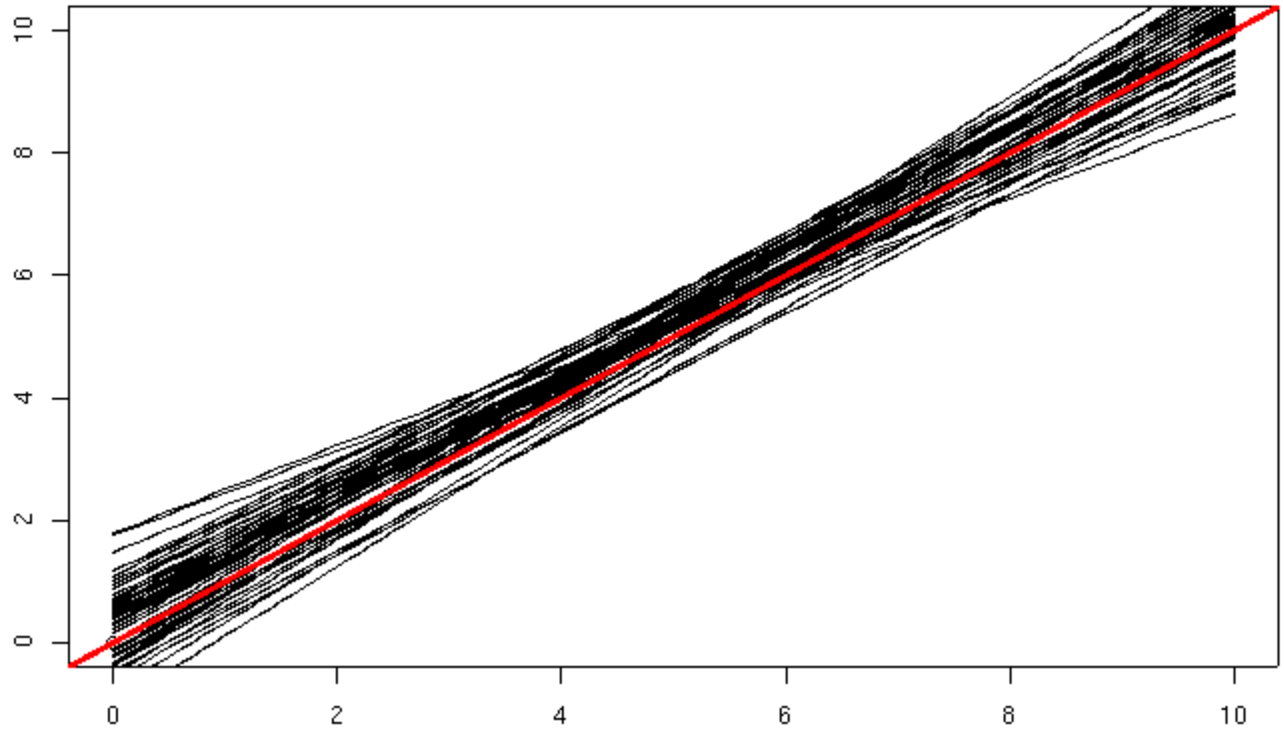
50 fits (20 examples each)



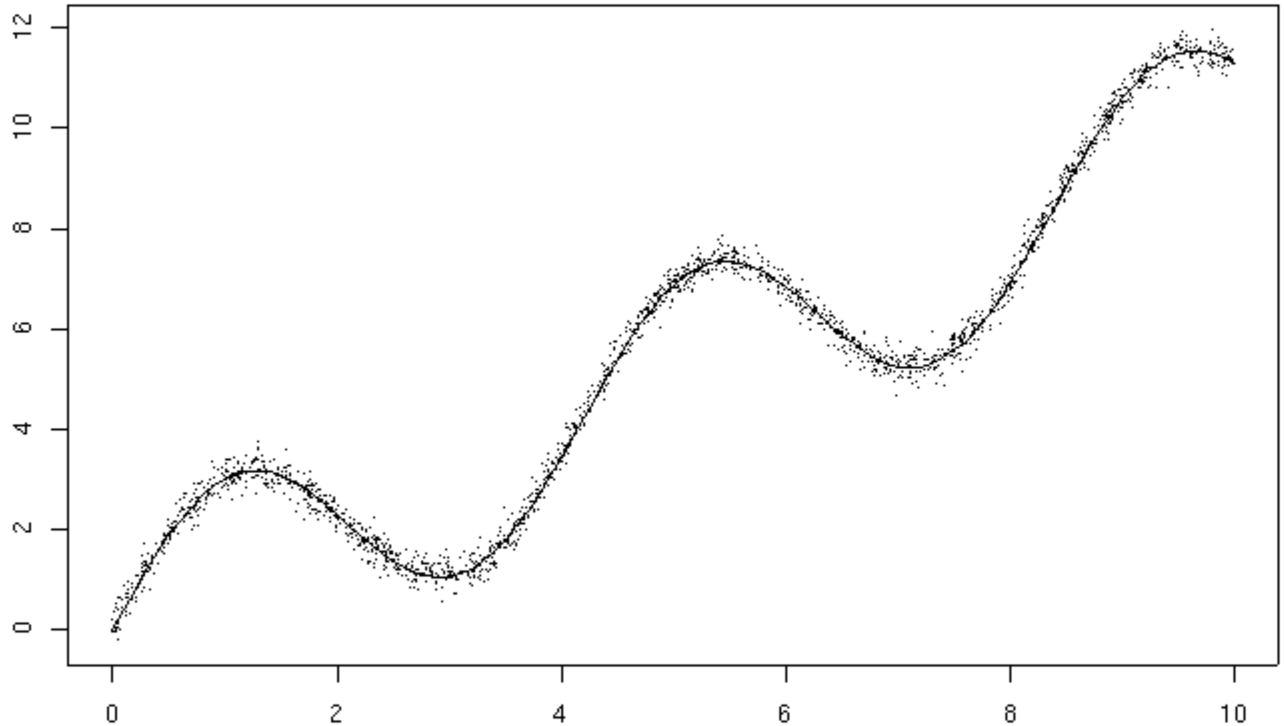
Bias



Variance



Noise



Bias^2

- Low bias
  - linear regression applied to linear data
  - 2nd degree polynomial applied to quadratic data
  - neural net with many hidden units trained to completion
- High bias
  - constant function
  - linear regression applied to non-linear data
  - neural net with few hidden units applied to non-linear data

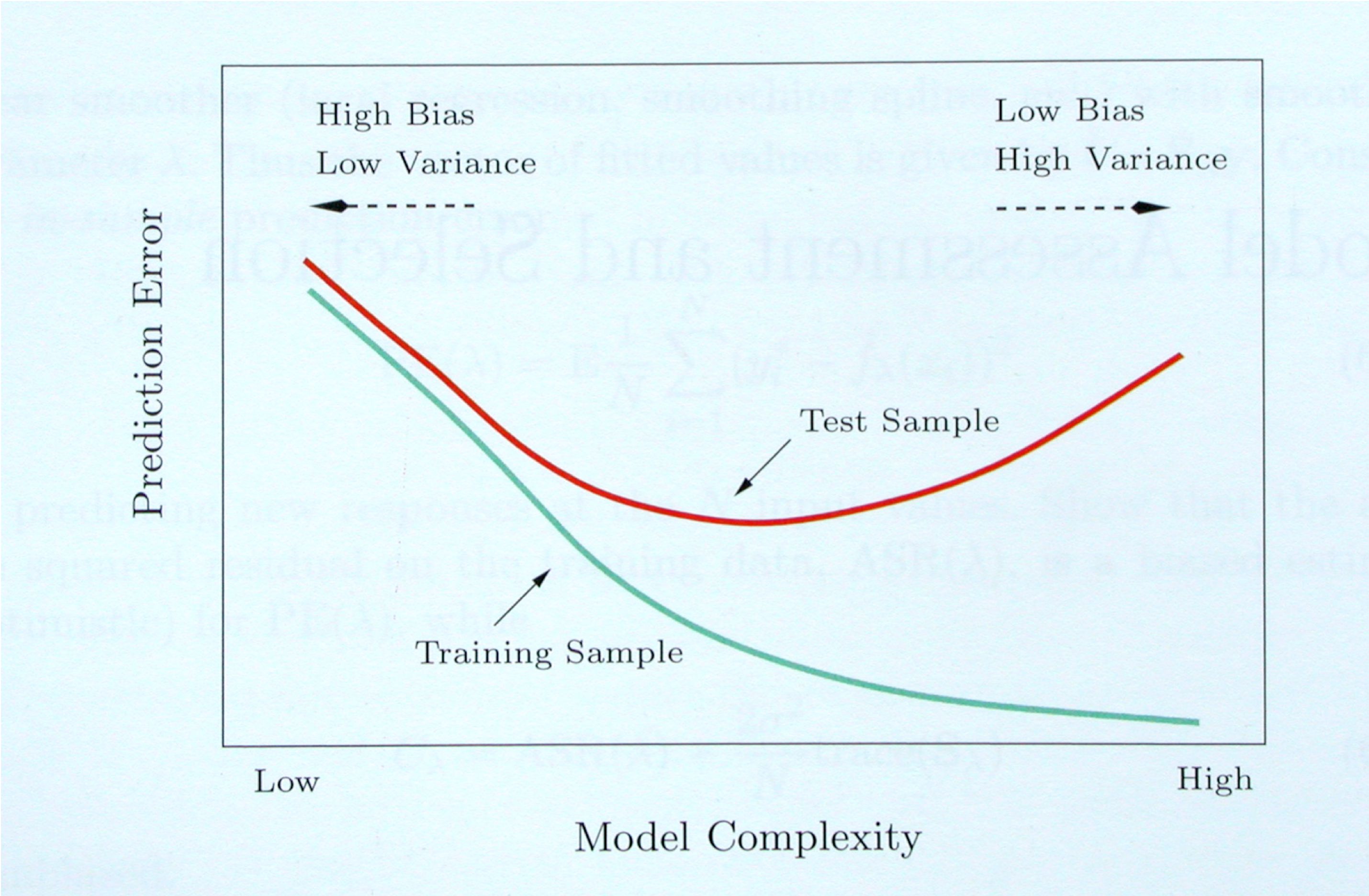
Variance

- Low variance
  - constant function
  - model independent of training data
- High variance
  - high degree polynomial
  - neural net with many hidden units trained to completion

Bias/Variance Tradeoff

- ($bias^2$+variance) is what counts for prediction
- Often:
  - low bias => high variance
  - low variance => high bias
- Tradeoff:
  - $bias^2$ vs. variance

Bias/Variance Tradeoff



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Reduce Variance Without Increasing Bias

- Averaging reduces variance:

$$Var(\bar{X}) = \frac{Var(X)}{N}$$

Average models to reduce model variance

One problem:

only one training set

where do multiple models come from?

Bagging: Bootstrap Aggregation

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set *D*.
- *Bootstrap sampling*: Given set *D* containing *N* training examples, create *D*' by drawing *N* examples at random with replacement from *D*.

- Bagging:
  - Create *k* bootstrap samples D_1 … D_k.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

Bagging

- Best case:

$$Var(Bagging(L(x,D))) = \frac{Variance(L(x,D))}{N}$$

In practice:

models are correlated, so reduction is smaller than 1/N

variance of models trained on fewer training cases usually somewhat larger

Bagging Experiments

i) The data set is randomly divided into a test set $\mathcal{T}$ and a learning set $\mathcal{L}$. In the real data sets $\mathcal{T}$ is 10% of the data. In the simulated waveform data, 1800 samples are generated. $\mathcal{L}$ consists of 300 of these, and $\mathcal{T}$ the remainder.

ii) A classification tree is constructed from $\mathcal{L}$ using 10-fold cross-validation. Running the test set $\mathcal{T}$ down this tree gives the misclassification rate $e_S(\mathcal{L}, \mathcal{T})$.

iii) A bootstrap sample $\mathcal{L}_B$ is selected from $\mathcal{L}$, and a tree grown using $\mathcal{L}_B$. The original learning set $\mathcal{L}$ is used as test set to select the best pruned subtree (see Section 4.3). This is repeated 50 times giving tree classifiers $\phi_1(\boldsymbol{x}), \ldots, \phi_{50}(\boldsymbol{x})$.

iv) If $(j_n, \boldsymbol{x}_n) \in \mathcal{T}$, then the estimated class of $\boldsymbol{x}_n$ is that class having the plurality in $\phi_1(\boldsymbol{x}_n), \ldots, \phi_{50}(\boldsymbol{x}_n)$. If there is a tie, the estimated class is the one with the lowest class label. The proportion of times the estimated class differs from the true class is the bagging misclassification rate $e_B(\mathcal{L}, \mathcal{T})$.

v) The random division of the data into $\mathcal{L}$ and $\mathcal{T}$ is repeated 100 times and the reported $\bar{e}_S, \bar{e}_B$ are the averages over the 100 iterations. For the waveform data, 1800 new cases are generated at each iteration. Standard errors of $\bar{e}_S$ and $\bar{e}_B$ over the 100 iterations are also computed.

Bagging Results

Data Set	$\bar{e}_S$	$\bar{e}_B$	**Decrease**
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Breiman “Bagging Predictors” Berkeley Statistics Department TR#421, 1994

When Will Bagging Improve Accuracy?

- Depends on the stability of the base-level classifiers.
- A learner is unstable if a small change to the training set *D* causes a large change in the output hypothesis φ.
  - If small changes in *D* causes large changes φ in then there will be an improvement in performance.
- Bagging helps unstable procedures, but could hurt the performance of stable procedures.
- Neural nets and decision trees are unstable.
- k-nn and naïve Bayes classifiers are stable.

More Randomness: Random Forests

- Build large collection of de-correlated trees and average them.

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Reduce Bias^2 and Decrease Variance?

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

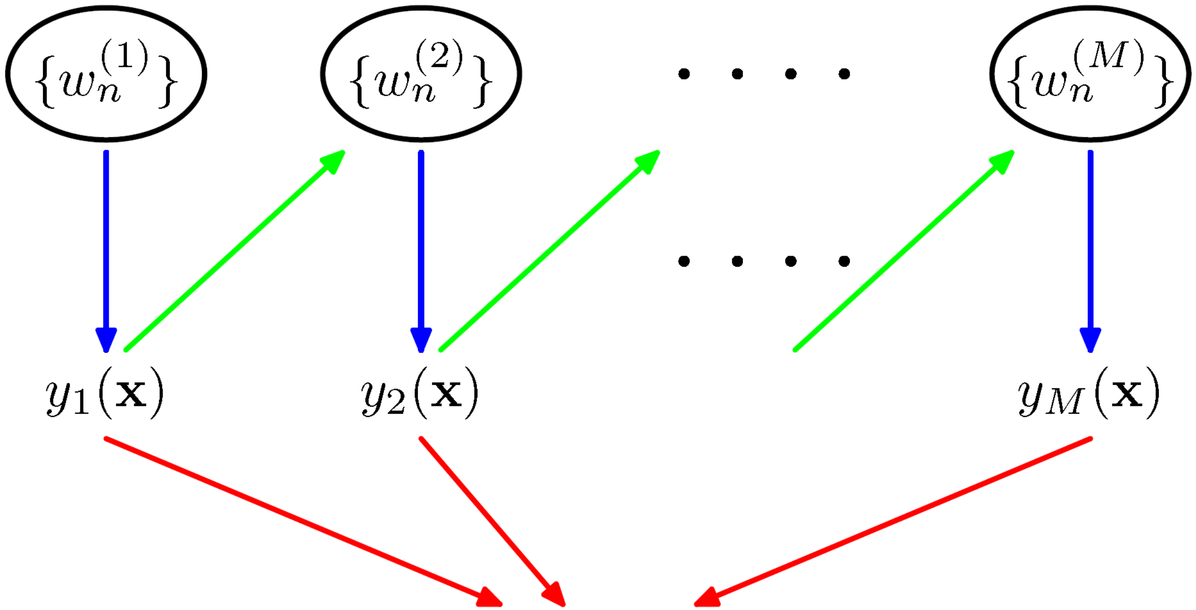
Boosting

- Freund & Schapire:
  - theory for “weak learners” in late 80’s
- Weak Learner: performance on ***any*** train set is slightly better than chance prediction
- intended to answer a theoretical question, not as a practical way to improve learning
- tested in mid 90’s using not-so-weak learners
- works anyway!

Boosting

- Weight all training samples equally
- Train model on training set
- Compute error of model on training set
- Increase weights on training cases model gets wrong
- Train new model on re-weighted training set
- Re-compute errors on weighted training set
- Increase weights again on cases model gets wrong
- Repeat until tired (100+ iterations)
- Final model: weighted prediction of each model

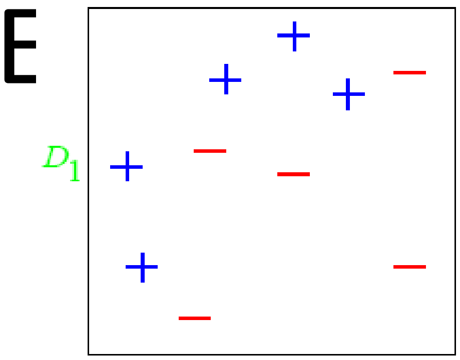
Boosting: Graphical Illustration

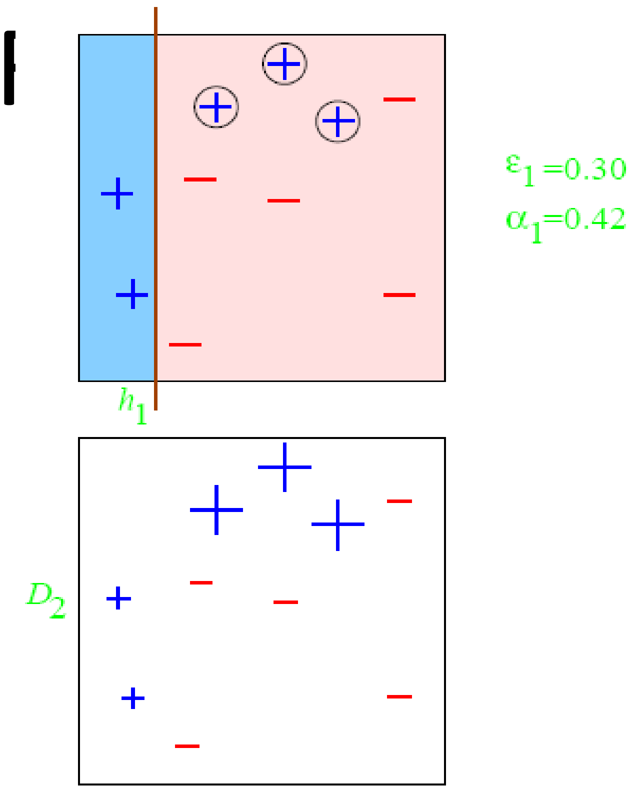


$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

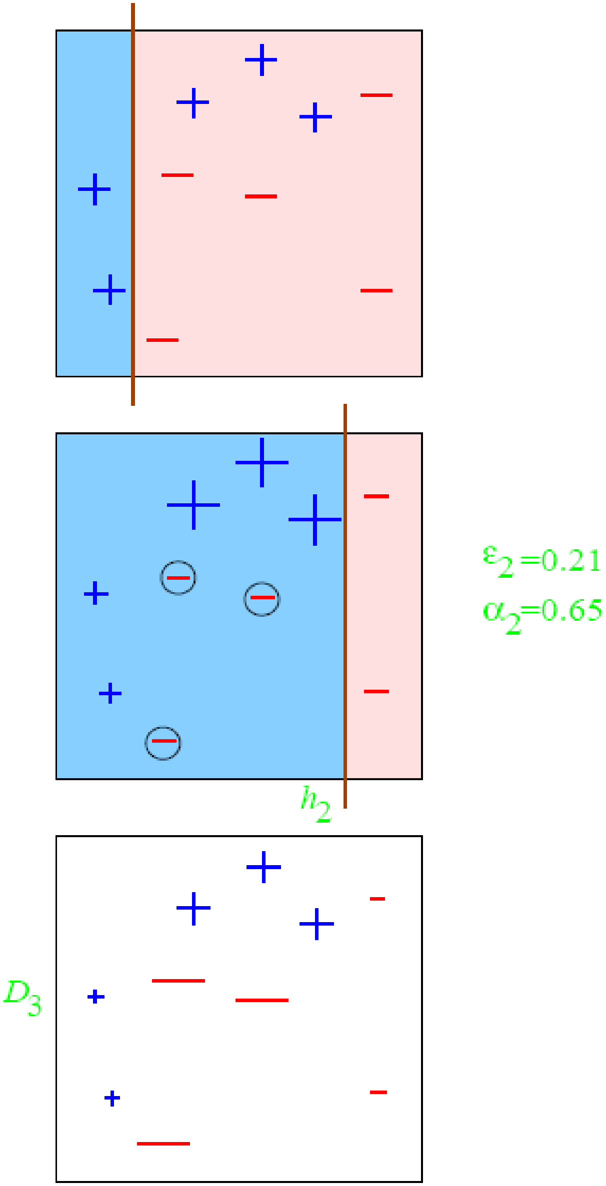
Algorithm 10.1 *AdaBoost.M1.*

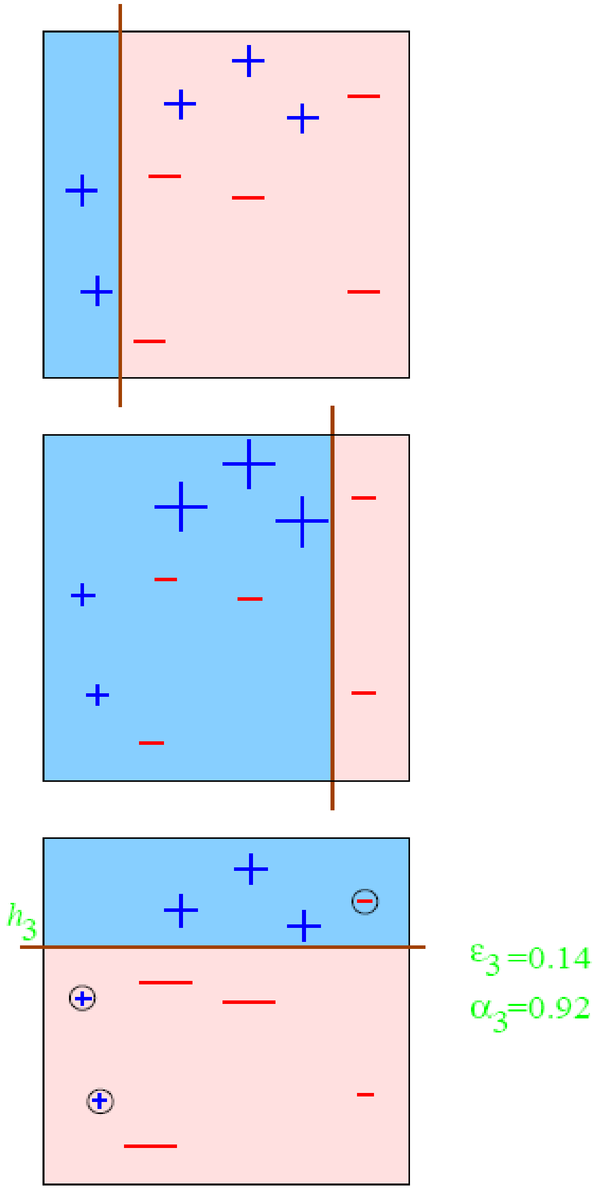
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.


E
D_1



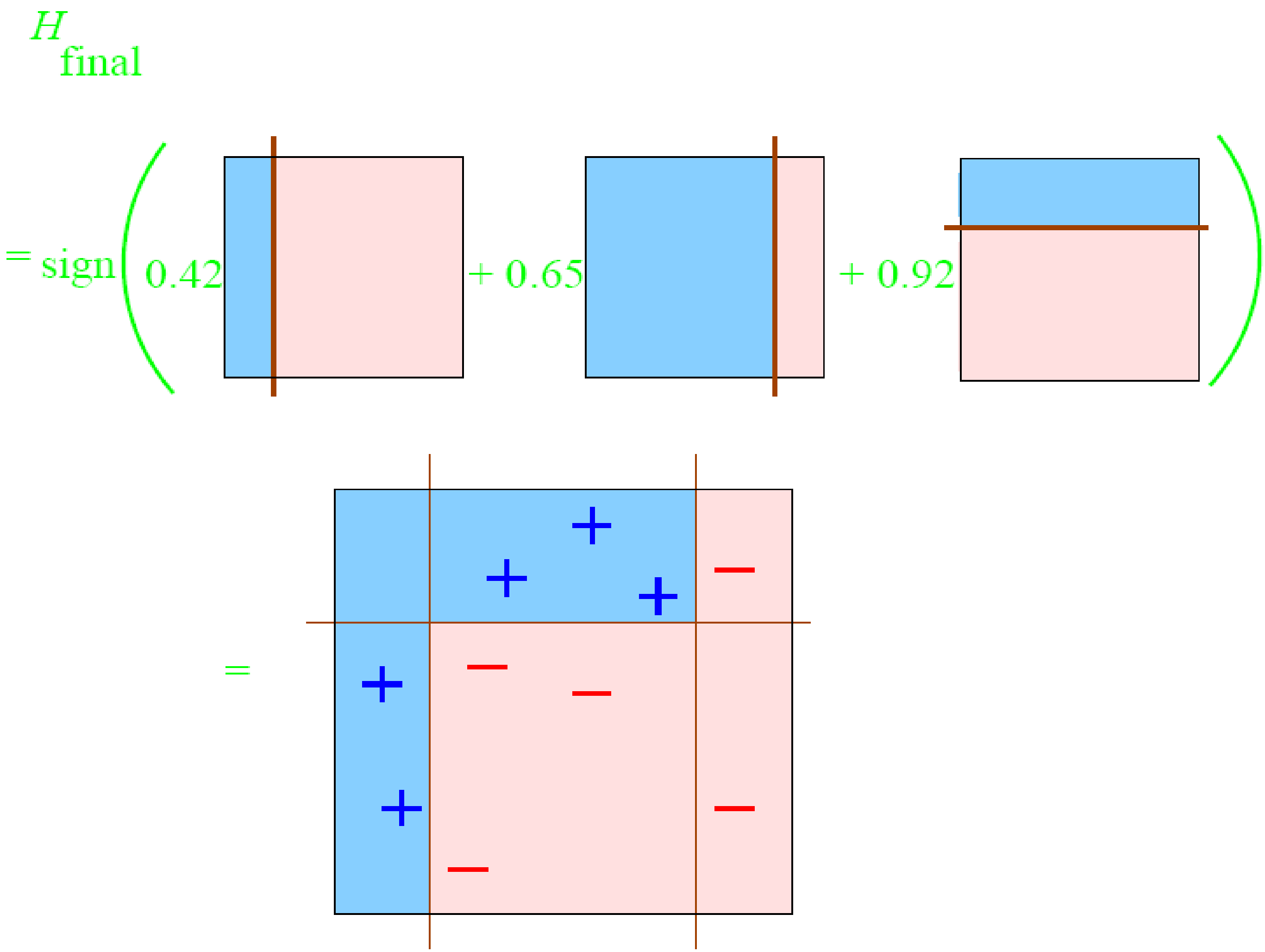
ε1 =0.30
α1=0.42
h1
D2


ε_2=0.21
α_2=0.65
h_2
D_3



h_3
$\varepsilon_3 = 0.14$
$\alpha_3 = 0.92$

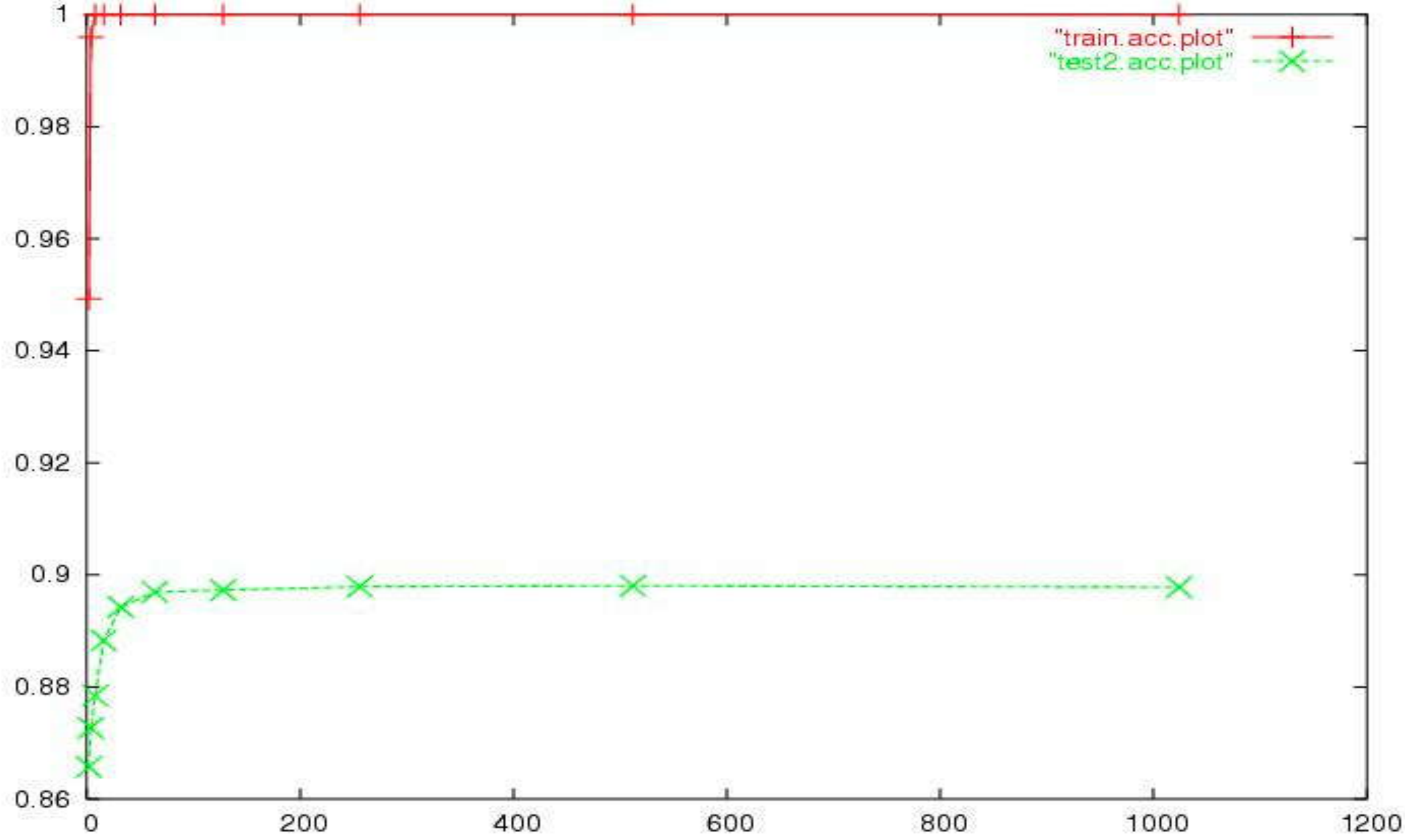
Final Hypothesis



Reweighting vs Resampling

- Example weights might be harder to deal with
  - Some learning methods can't use weights on examples
- We can resample instead:
  - Draw a bootstrap sample from the data with the probability of drawing each example proportional to its weight
- Reweighting usually works better but resampling is easier to implement

Boosting Performance



Summary: Boosting vs. Bagging

- Bagging doesn’t work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn’t have this problem.
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- Bagging is easier to parallelize.

Other Approaches

- Mixture of Experts (See Bishop, Chapter 14)
- Cascading Classifiers
- many others…