Bias/Variance Tradeoff and Ensemble Methods

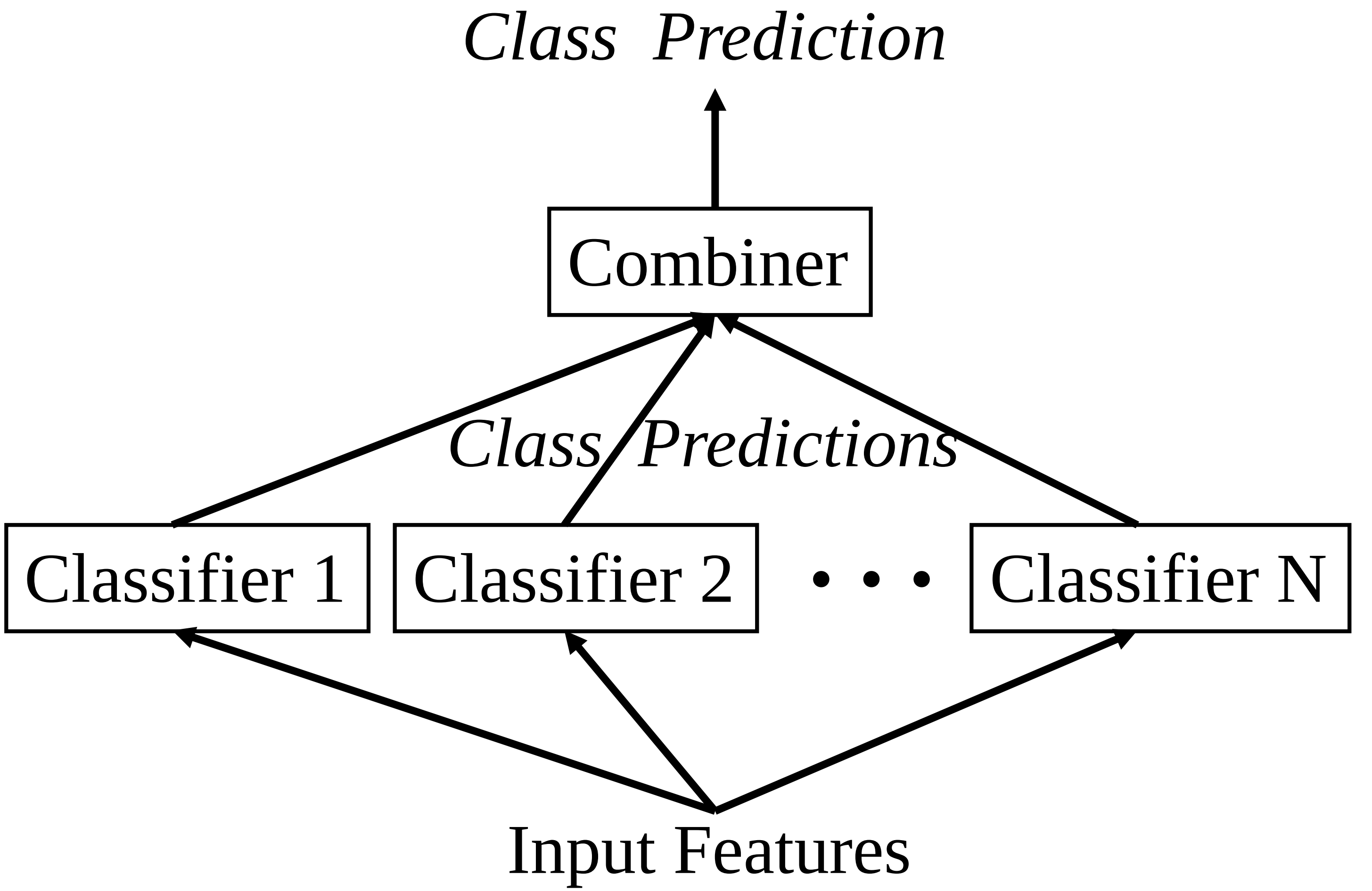
Vibhav Gogate
The University of Texas at Dallas
Machine learning
CS 6375

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

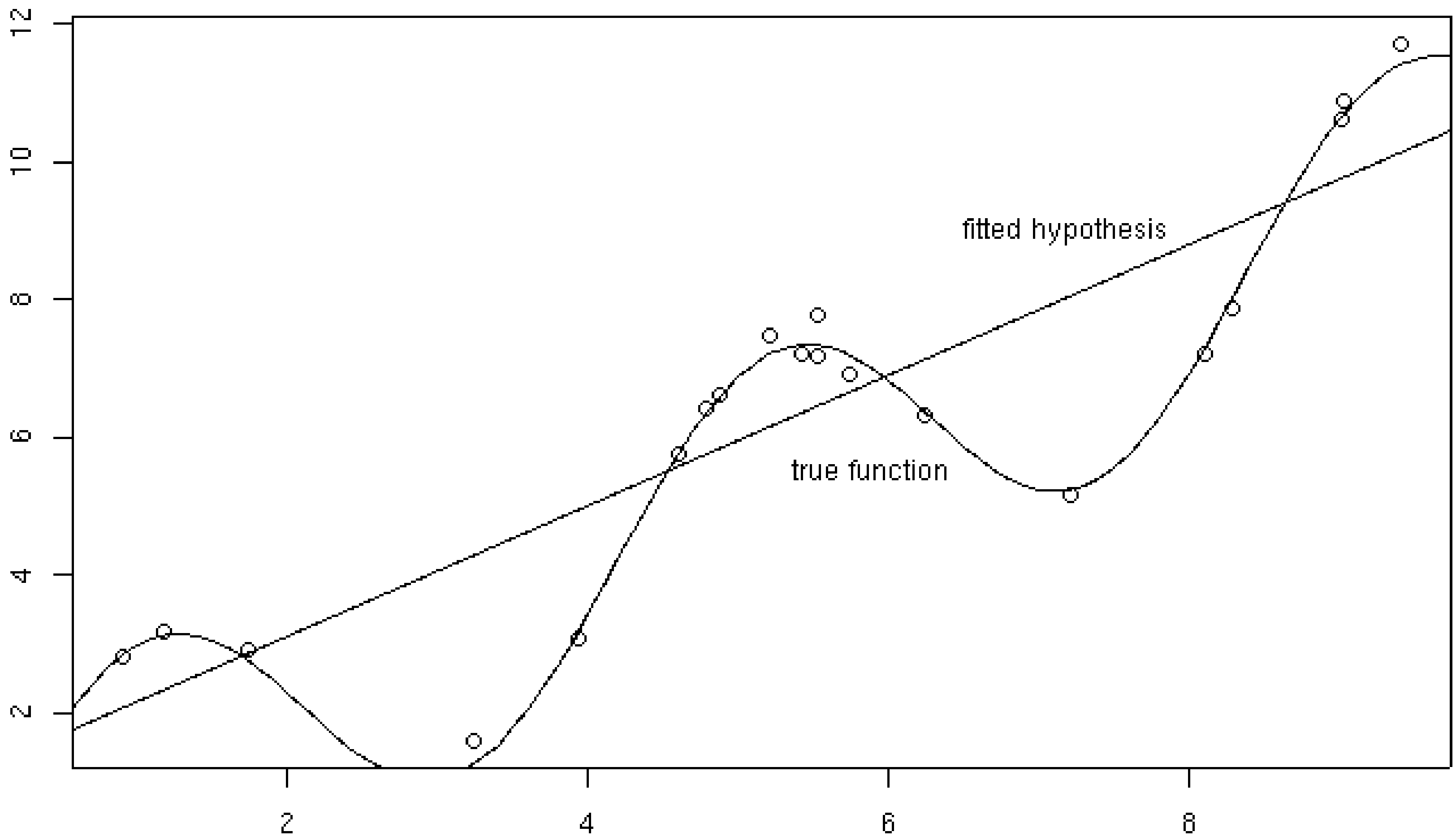
- bias:
  - measures the accuracy or quality of the algorithm
  - high bias means a poor match
- variance:
  - measures the precision or specificity of the match
  - a high variance means a weak match
- We would like to minimize each of these
- Unfortunately, we can’t do this independently, there is a trade-off

Bias-Variance Analysis in Regression

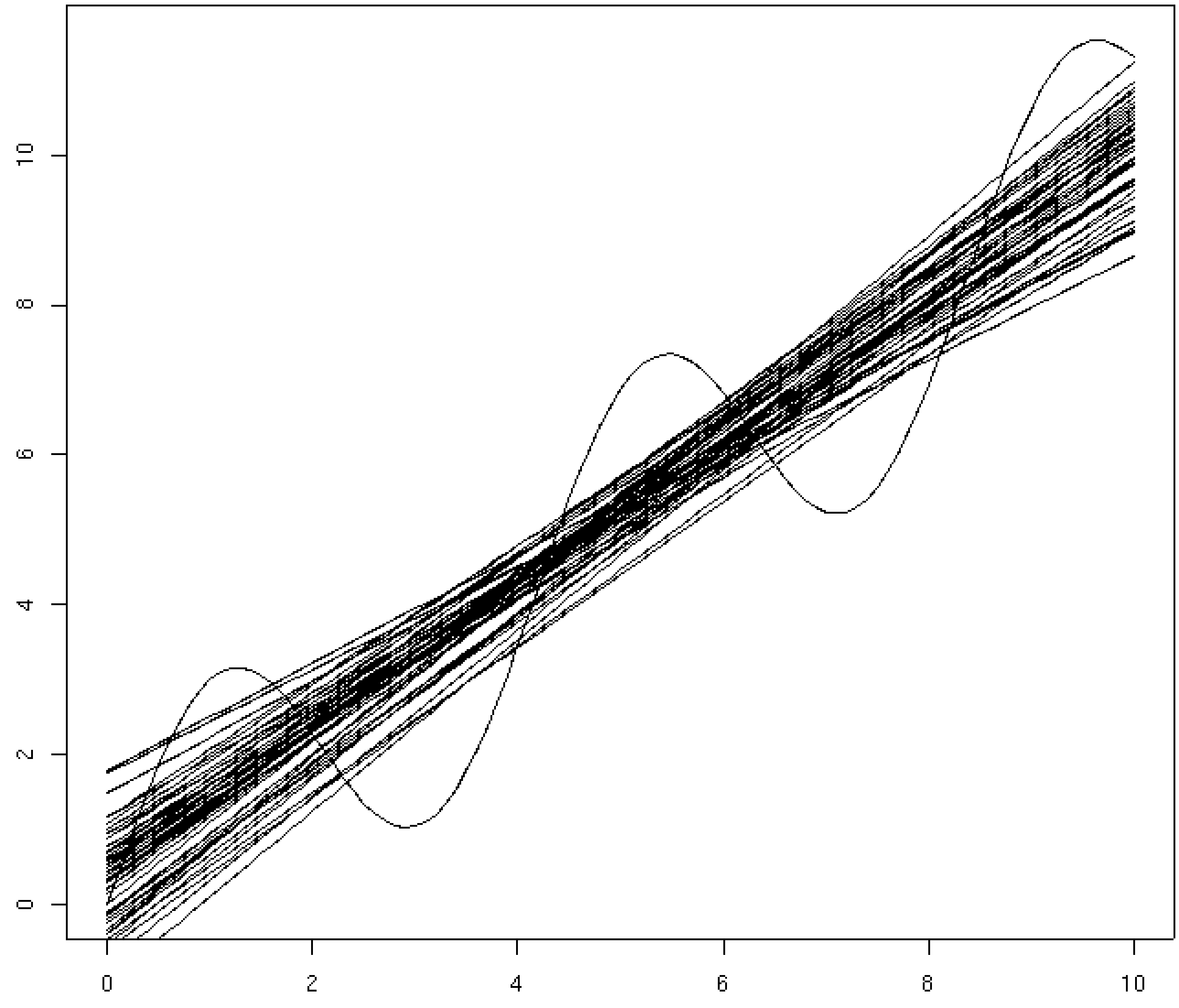
- True function is $y = f(x) + \varepsilon$
  - where ε is normally distributed with zero mean and standard deviation σ.
- Given a set of training examples, $\{(x_i, y_i)\}$, we fit an hypothesis $h(x) = w \cdot x + b$ to the data to minimize the squared error

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

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



50 fits (20 examples each)



Bias-Variance Analysis

- 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 $\bar{Z} = E_P[\,Z\,]$ be the average value of Z.
- Lemma: $E[\,(Z-\bar{Z})^2\,] = E[Z^2] - \bar{Z}^2$

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

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

Bias-Variance-Noise Decomposition

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

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

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

$$- 2\overline{h(x^*)}f(x^*)$$

$$+ E[(y^* - f(x^*))^2] + f(x^*)^2$$

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

$$\left(\overline{h(x^*)}^2 - f(x^*)\right)^2$$ BIAS

$$+ E[(y^* - f(x^*))^2]$$ NOISE

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

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

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

Bias, Variance, and Noise

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

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

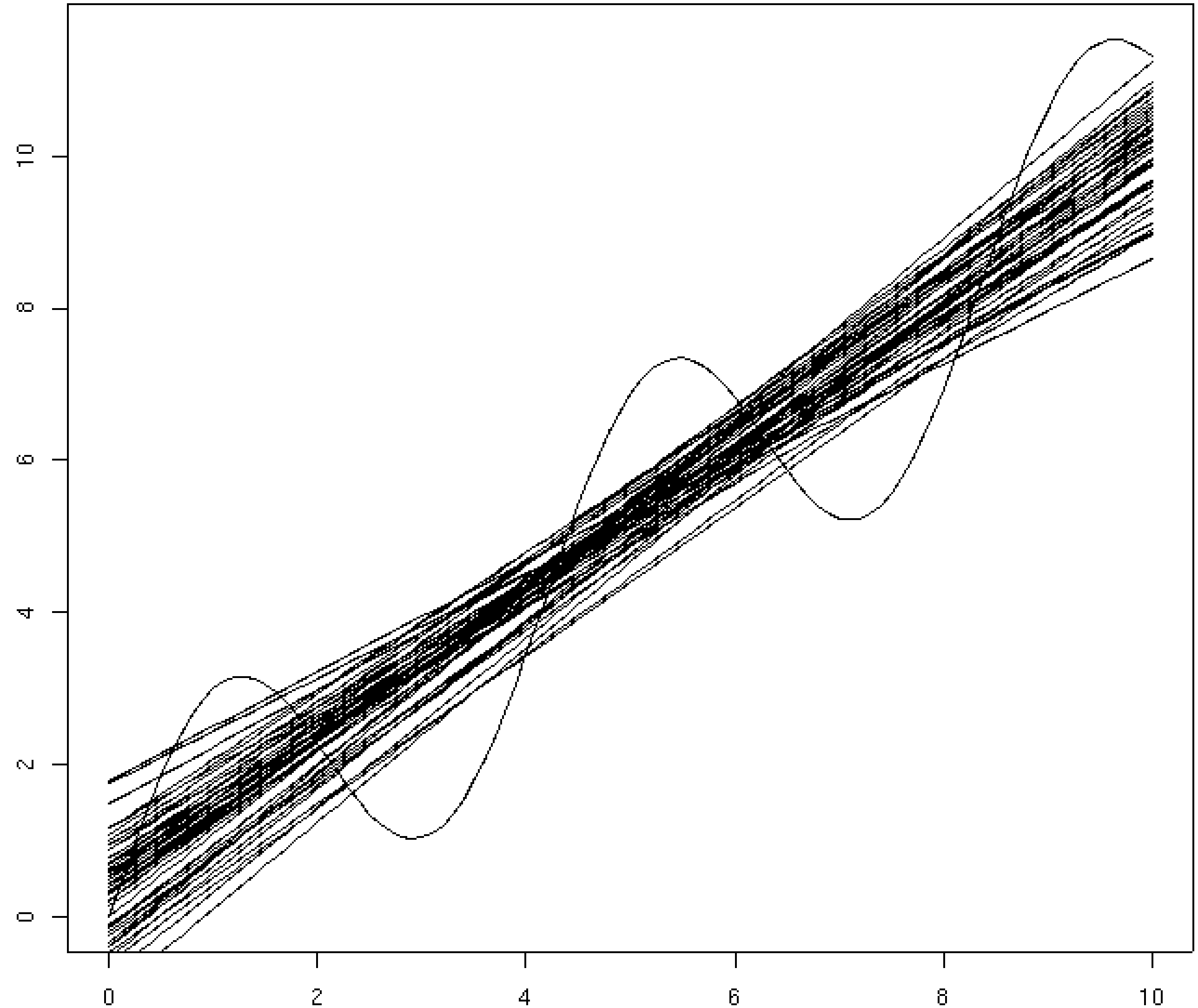
- Bias: $[\overline{h(x^*)} - f(x^*)]$

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

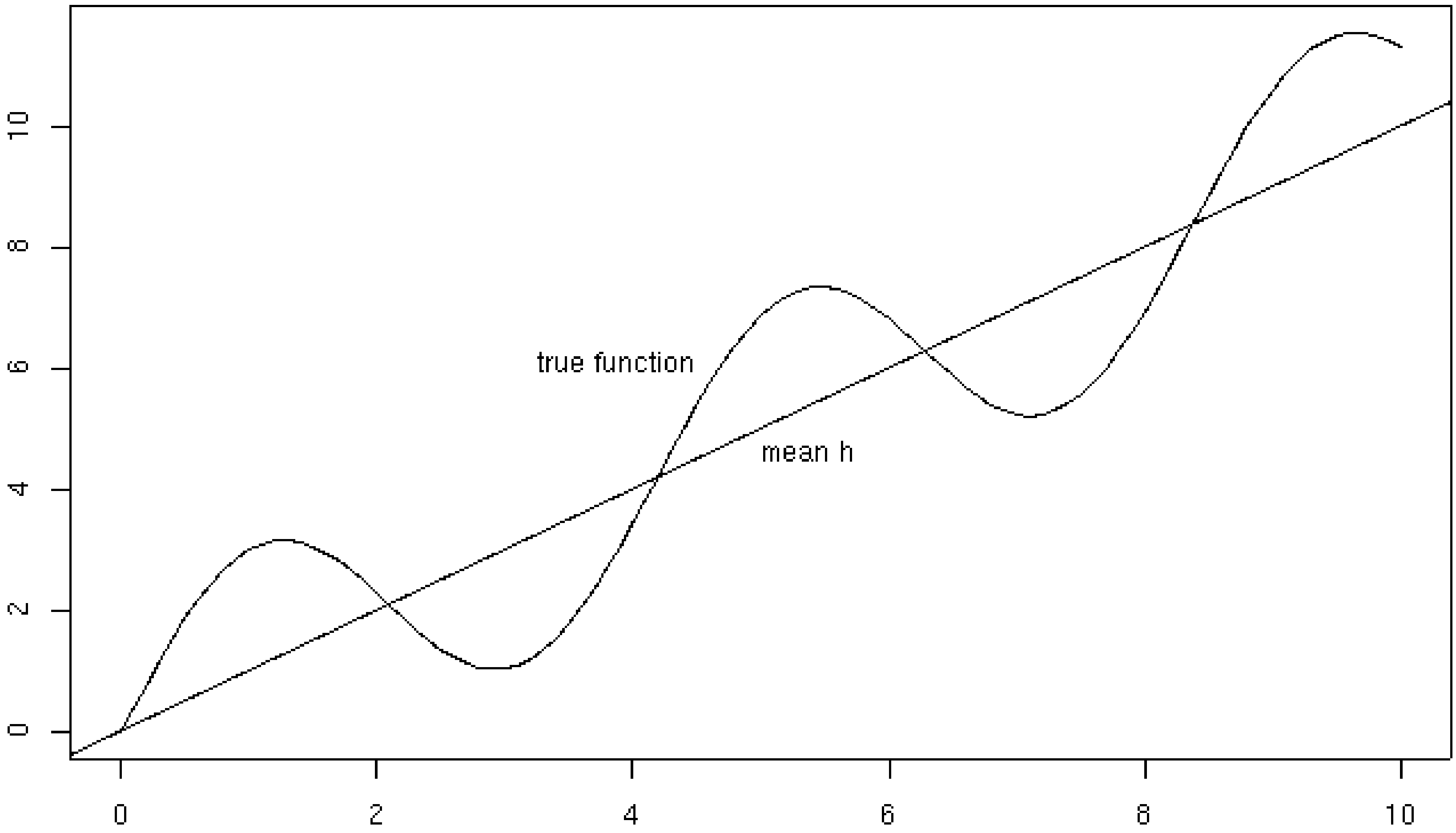
- Noise: $E[\,(y^* - f(x^*))^2\,] = E[\varepsilon^2] = \sigma^2$

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

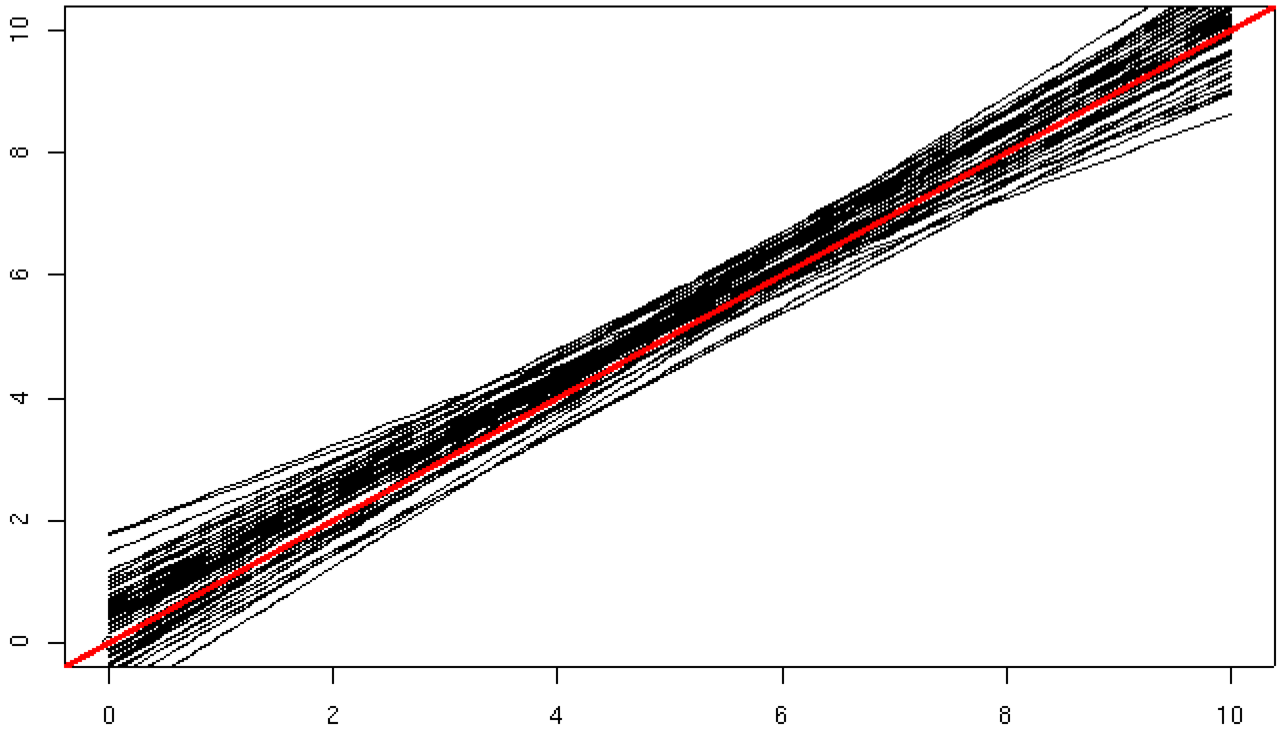
50 fits (20 examples each)



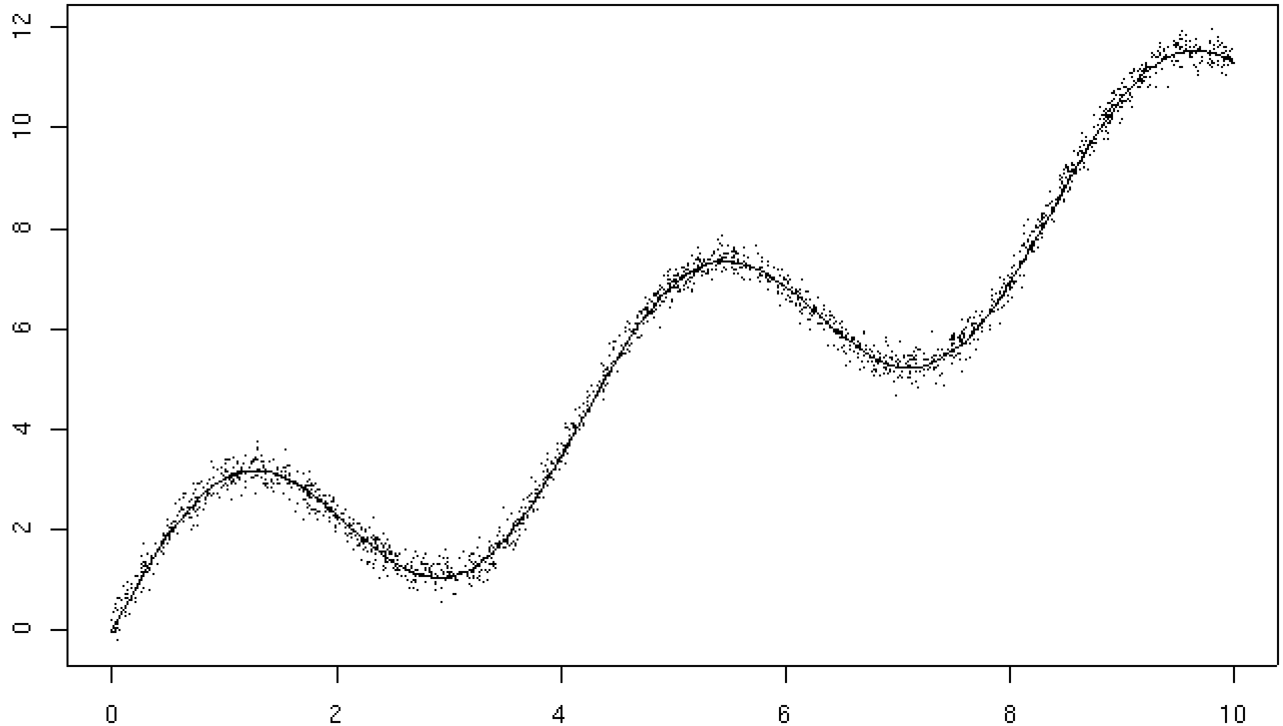
Bias



Variance



Noise



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

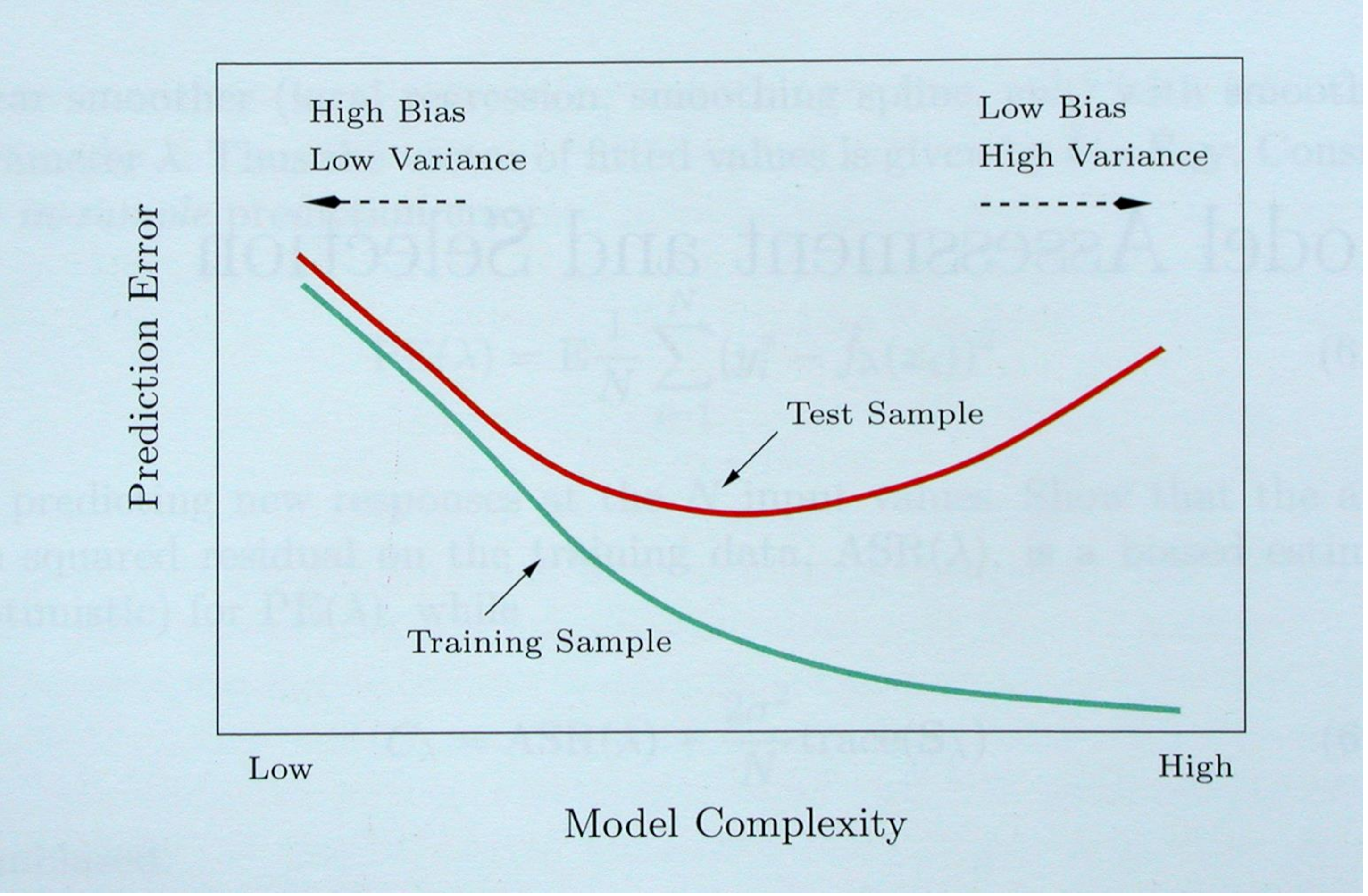
Variance

- 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 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.

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

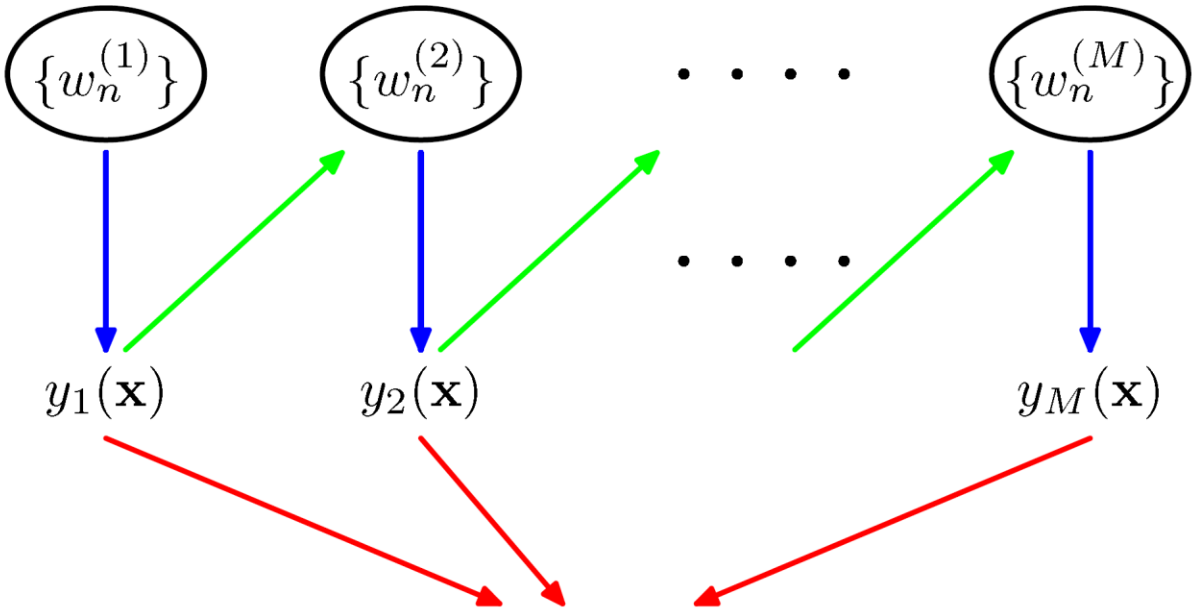
Boosting

- 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



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

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

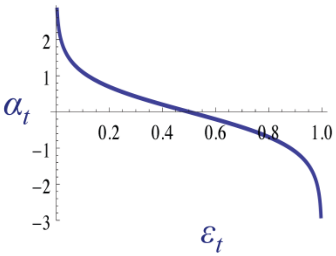
   and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}. \tag{14.17}$$

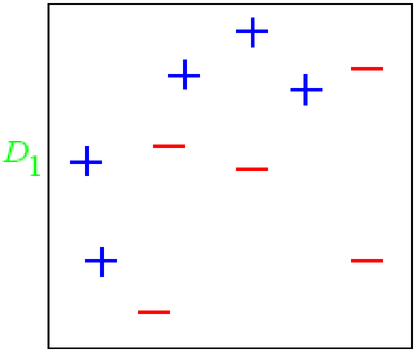
   (c) Update the data weighting coefficients

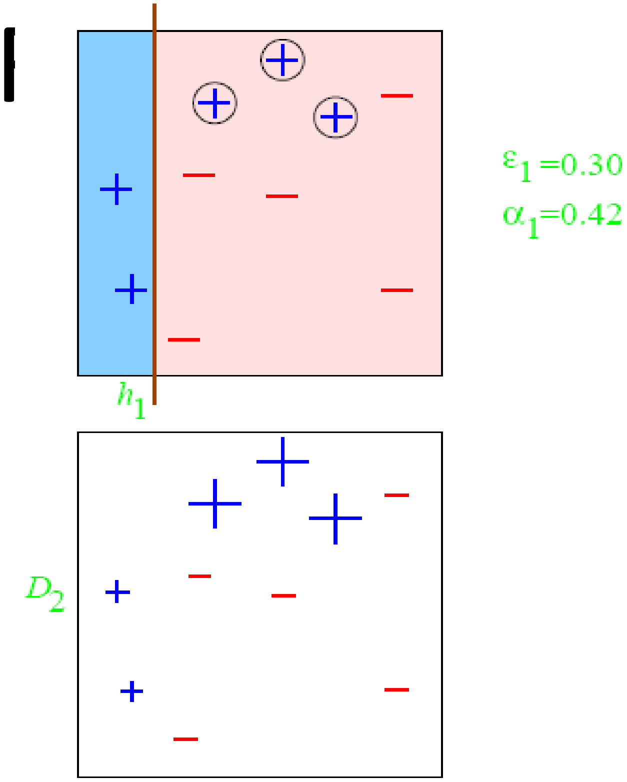
$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{14.18}$$

- α_t :
  - No errors: ε_t=0 → α_t=∞
  - All errors: ε_t=1 → α_t=−∞
  - Random: ε_t=0.5 → α_t=0

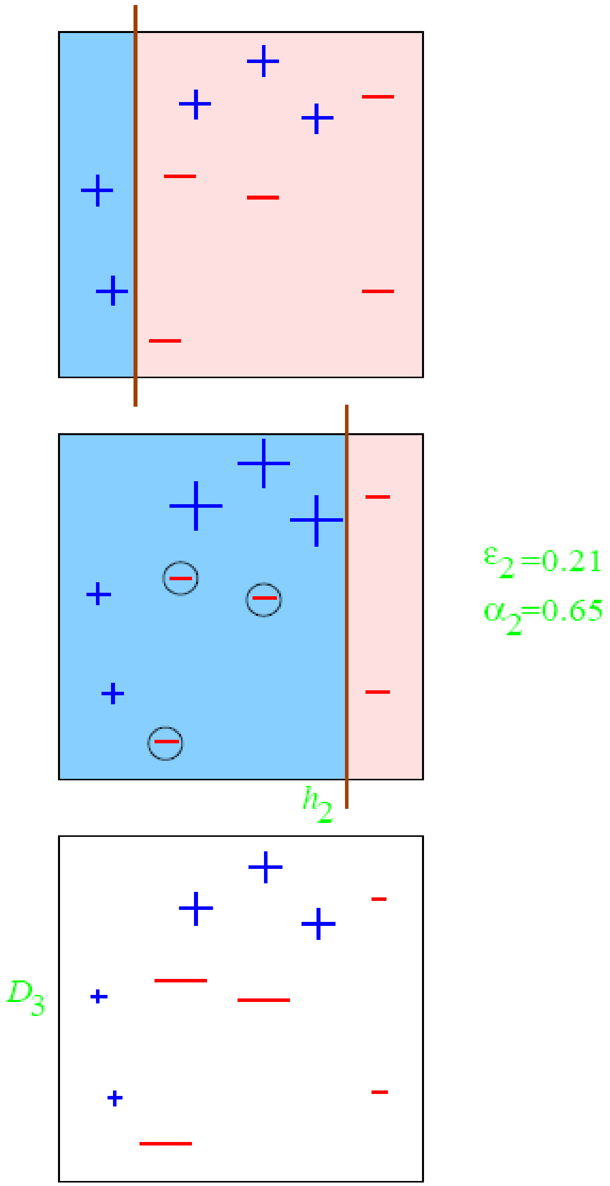


E

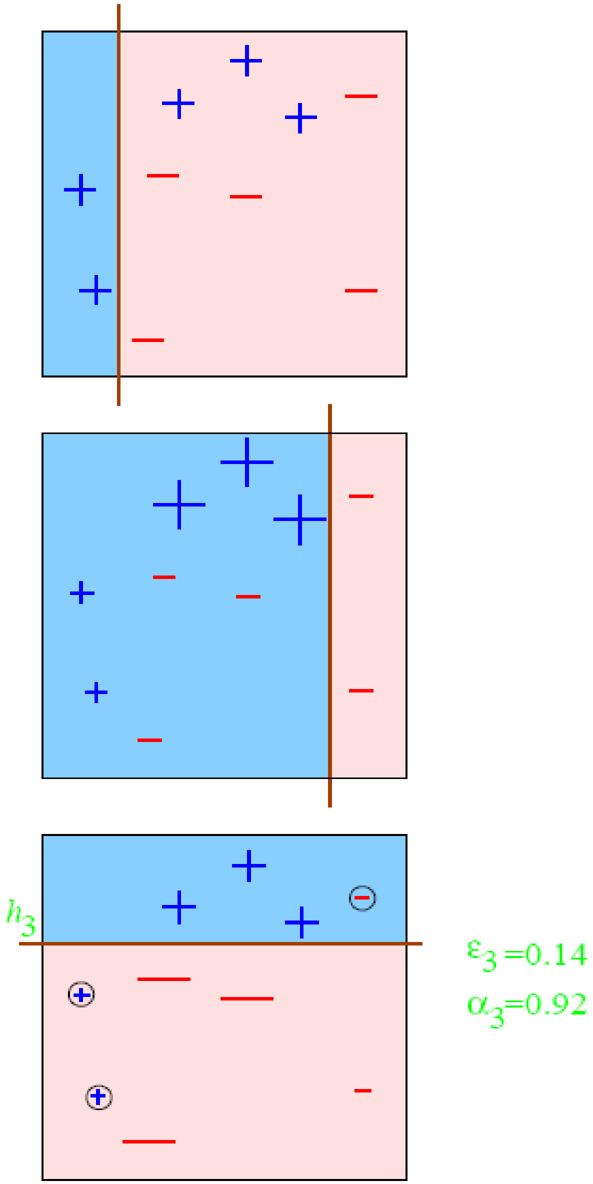




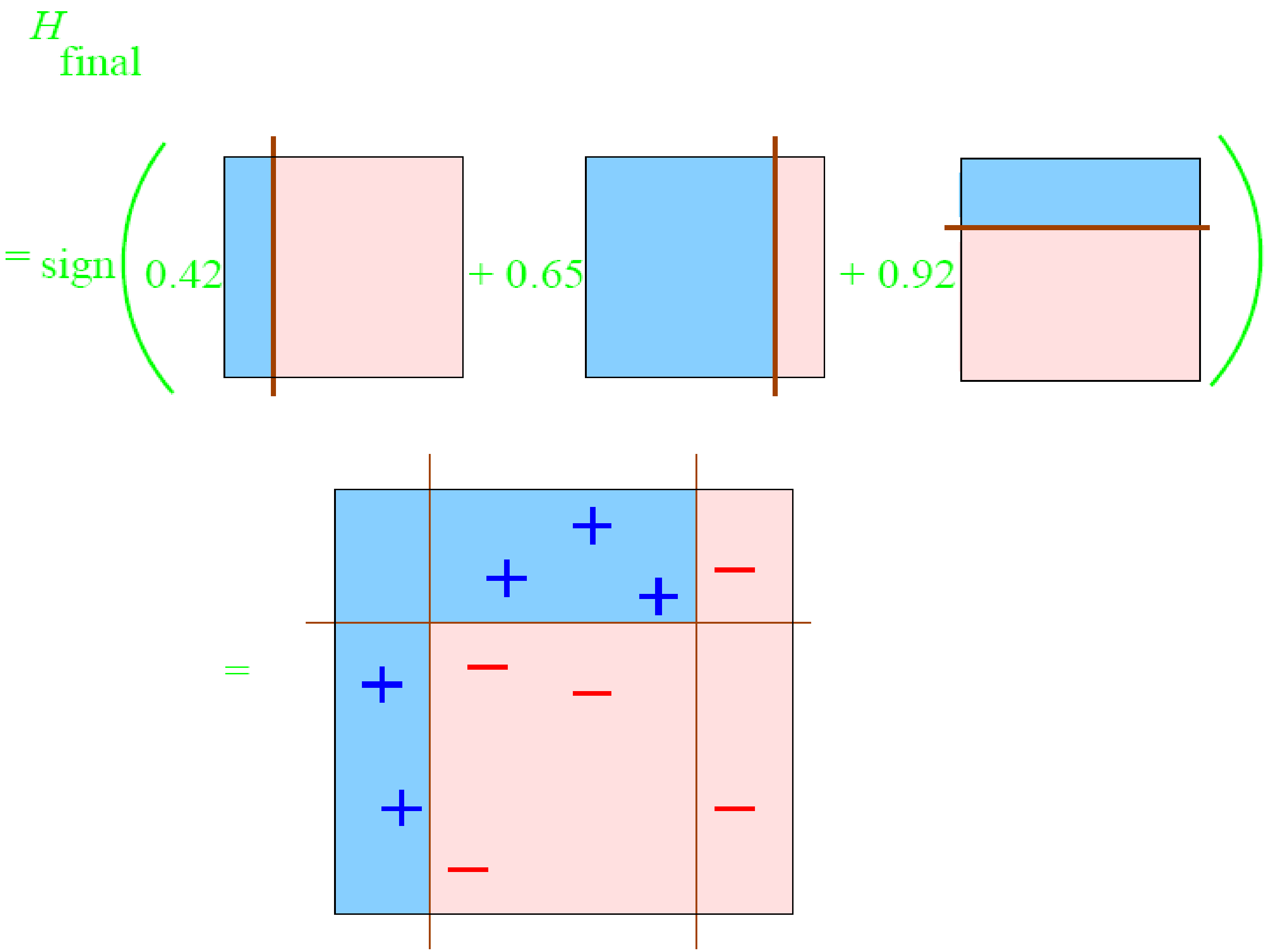
$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$
h_1
D_2



$\varepsilon_2=0.21$
$\alpha_2=0.65$
h_2
D_3


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

Final Hypothesis



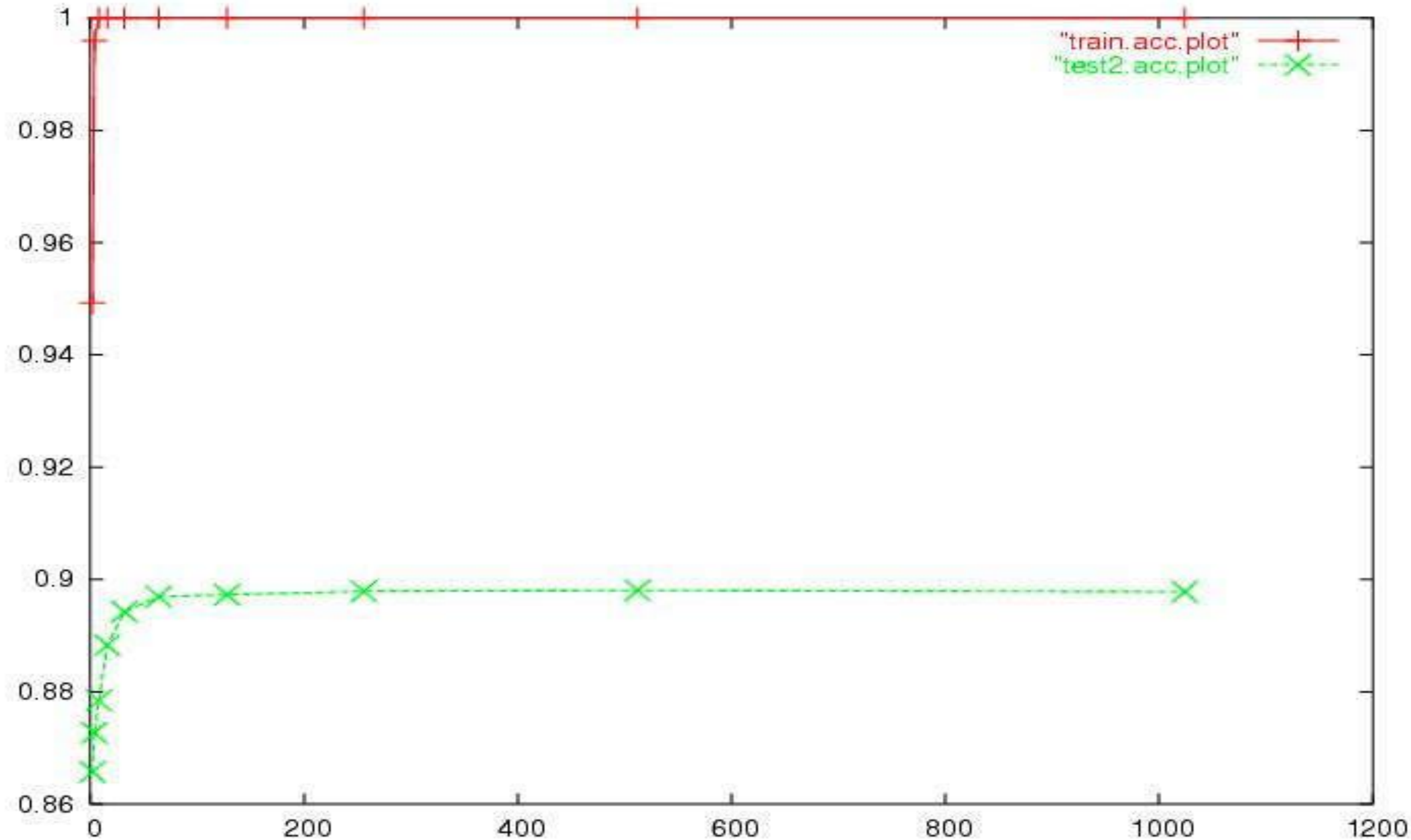
Reweighting vs Resampling

- 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…