Machine Learning (CS 567)

Fall 2008

Time: T-Th 5:00pm - 6:20pm
Location: GFS 118

Instructor: Sofus A. Macskassy (macskass@usc.edu)
Office: SAL 216
Office hours: by appointment

Teaching assistant: **Cheol Han (cheolhan@usc.edu)**
Office: SAL 229
Office hours: M 2-3pm, W 11-12

Class web page:
http://www-scf.usc.edu/~csci567/index.html

Bias-Variance Outline

- Analysis of Ensemble Learning Algorithms
- Effect of Bagging on Bias and Variance
- Effect of Boosting on Bias and Variance

Bias-Variance Theory

- Decompose Error Rate into components, some of which can be measured on unlabeled data

- Bias-Variance Decomposition for Regression
- Bias-Variance Decomposition for Classification

Bias-Variance Analysis in Regression

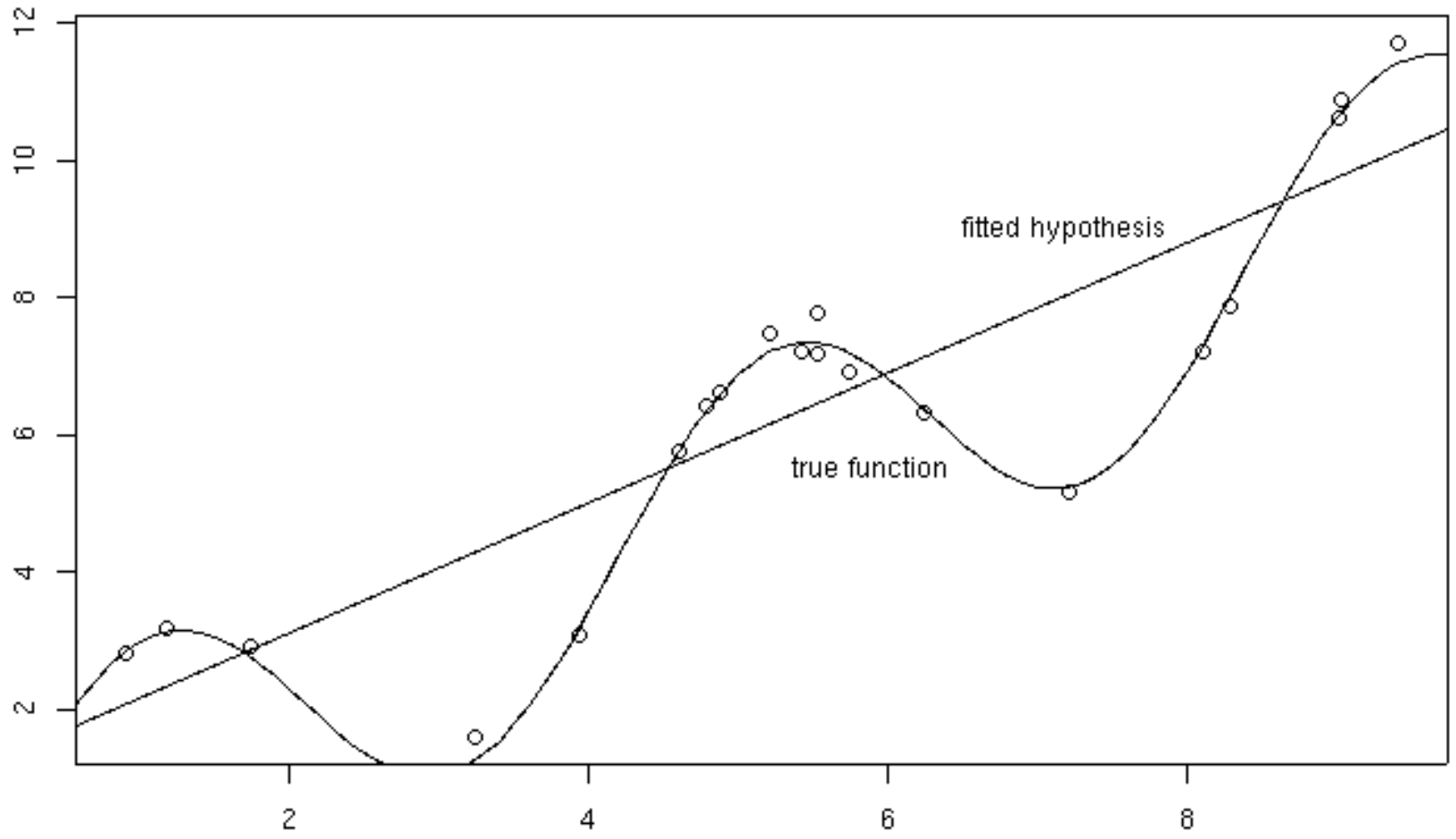
- Suppose we have examples <**x**,y> where the true function is $y = f(\mathbf{x}) + \varepsilon$ and where ε is Gaussian noise with zero mean and standard deviation σ.
- In linear regression, given a set of examples $\langle \mathbf{x}_i, y_i \rangle_{i=1\ldots m}$, we fit a linear hypothesis $h(\mathbf{x}) = \mathbf{w}\mathbf{x} + w_0$, such as to minimize sum-squared error over the training data:

$$\sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2$$

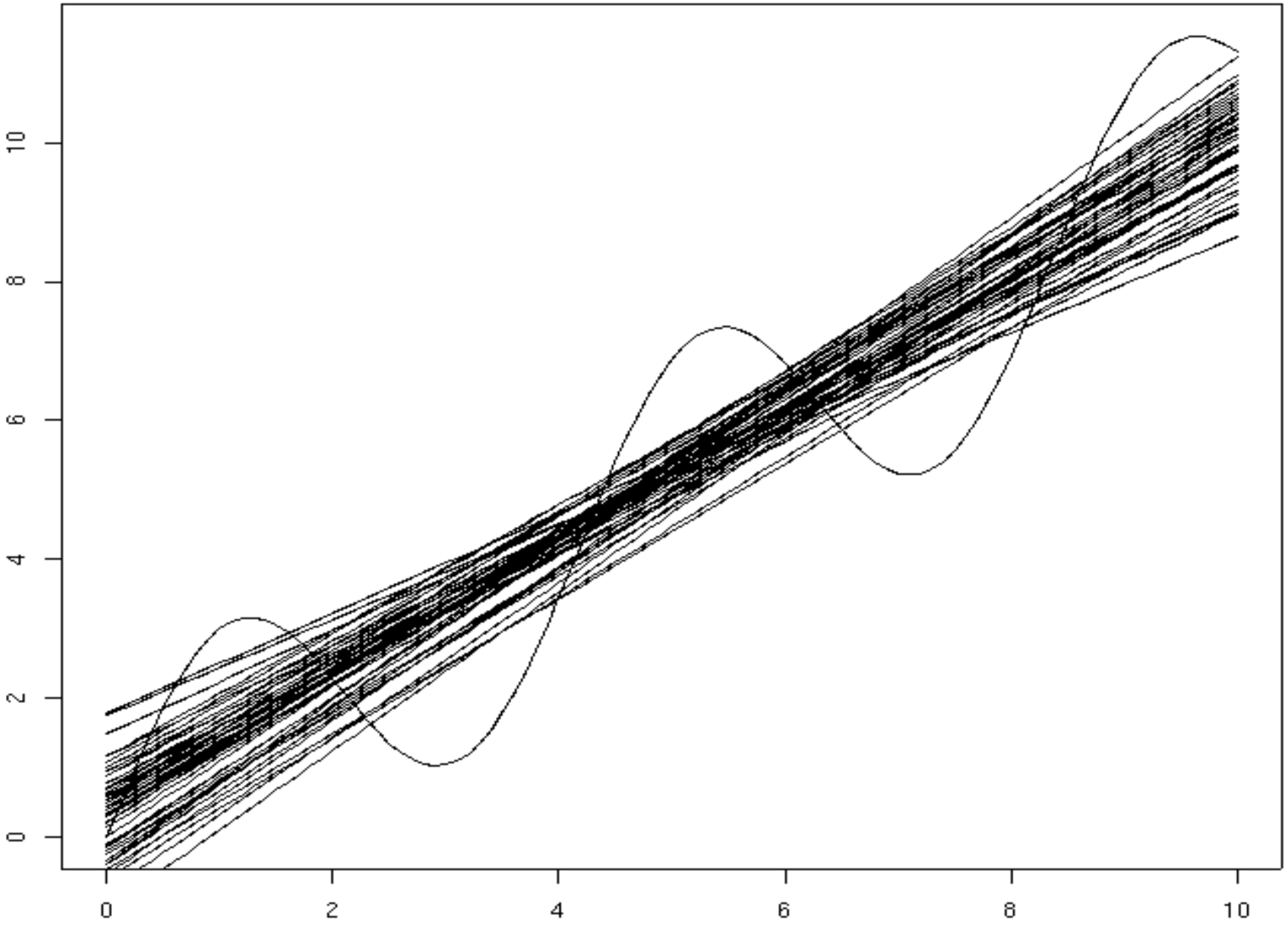
- Because of the hypothesis class that we chose (linear hypotheses) for some function f, we will have a systematic prediction error
- Depending on the data set we have, the parameters **w** that we find will be different.

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**, what is the **expected prediction error?**
- Assume that the data points are drawn i.i.d. from *a unique underlying probability distribution P*
- The goal of the analysis is to compute, for an arbitrary new point **x**,

$$E_P\left[(y - h(\mathbf{x}))^2\right]$$

  where y is the value of **x** that could be present in a data set, and the expectation is over *all training sets* drawn according to P.
- We will decompose this expectation into three components:

  bias, variance and noise

Recall: Statistics 101

- Let Z be a random variable with possible values $z_i, i=1...n$ and with probability distribution $P(Z)$
- The **expected value** or **mean** of Z is:

$$\underline{Z} = E_P[Z] = \sum_{i=1}^{n} z_i P(z_i)$$

- If Z is continuous the sum is replaced by an integral, and the distribution by a density function.
- The **variance** of Z is:

$$\begin{aligned} Var[Z] &= E\left[(Z - \underline{Z})^2\right] \\ &= E[Z^2] - \underline{Z}^2 \end{aligned}$$

(this last line is proven on the next slide)

The variance lemma

$$
\begin{aligned}
Var[Z] &= E\left[(Z-\underline{Z})^2\right] \\
&= \sum_{i=1}^{n}(z_i-\underline{Z})^2 P(z_i) \\
&= \sum_{i=1}^{n}\left(z_i^2-2z_i\underline{Z}+\underline{Z}^2\right)P(z_i) \\
&= \sum_{i=1}^{n} z_i^2 P(z_i) - 2\underline{Z}\sum_{i=1}^{n} z_i P(z_i) + \underline{Z}^2\sum_{i=1}^{n} P(z_i) \\
&= E[Z^2] - 2\underline{Z}\,\underline{Z} + \underline{Z}^2\cdot 1 \\
&= E[Z^2] - \underline{Z}^2
\end{aligned}
$$

We will use the form:

$$E\left[Z^2\right] = \underline{Z}^2 + Var\left[Z\right]$$

Bias-variance decomposition

$$\begin{aligned} E_P\left[(y - h(\mathbf{x}))^2\right] &= E_P\left[h(\mathbf{x})^2 + 2yh(\mathbf{x}) + y^2\right] \\ &= E_P\left[h(\mathbf{x})^2\right] + E_P\left[y^2\right] - 2E_P[y]E_P[h(\mathbf{x})] \end{aligned}$$

- Let $\overline{h}(\mathbf{x}) = E_P[h(\mathbf{x})]$ denote the ***mean prediction*** of the hypothesis at **x**, when h is trained with data drawn from P
- For the first term, using the variance lemma, we have:

$$E_P\left[h(\mathbf{x})^2\right] = E_P\left[(h(\mathbf{x}) - \overline{h}(\mathbf{x}))^2\right] + \overline{h}(\mathbf{x})^2$$

- Note that $E_P[y] = E_P[f(\mathbf{x}) + \varepsilon] = f(\mathbf{x})$
- For the second term, using the variance lemma, we have:

$$E_P\left[y^2\right] = E_P\left[(y - f(\mathbf{x}))^2\right] + f(\mathbf{x})^2$$

Bias-variance decomposition (2)

- Putting everything together, we have:

$$
\begin{aligned}
E_P\left[(y-h(\mathbf{x}))^2\right] &= E_P\left[(h(\mathbf{x})-\overline{h}(\mathbf{x}))^2\right] + \\
&\quad \overline{h}(\mathbf{x})^2 - 2f(\mathbf{x})\overline{h}(\mathbf{x}) + f(\mathbf{x})^2 + \\
&\quad E_P\left[(y-f(\mathbf{x}))^2\right] \\
&= E_P\left[(h(\mathbf{x})-\overline{h}(\mathbf{x}))^2\right] + \qquad \text{(variance)} \\
&\quad (h(\mathbf{x})-f(\mathbf{x}))^2 + \qquad \text{(bias)}^2 \\
&\quad E_P\left[(y-f(\mathbf{x}))^2\right] \qquad \text{(noise)} \\
&= \text{Var}[h(\mathbf{x})] + \text{Bias}[h(\mathbf{x})]^2 + E_P[\varepsilon^2] \\
&= \text{Var}[h(\mathbf{x})] + \text{Bias}[h(\mathbf{x})]^2 + \sigma^2
\end{aligned}
$$

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

Bias, Variance and Noise

- **Variance**: $E_P\left[(h(\mathbf{x}) - \overline{h}(\mathbf{x}))^2\right]$

  Is the hypothesis h when trained with finite data sets sampled randomly from P. It describes how much h(**x**) varies from one training set S to another.

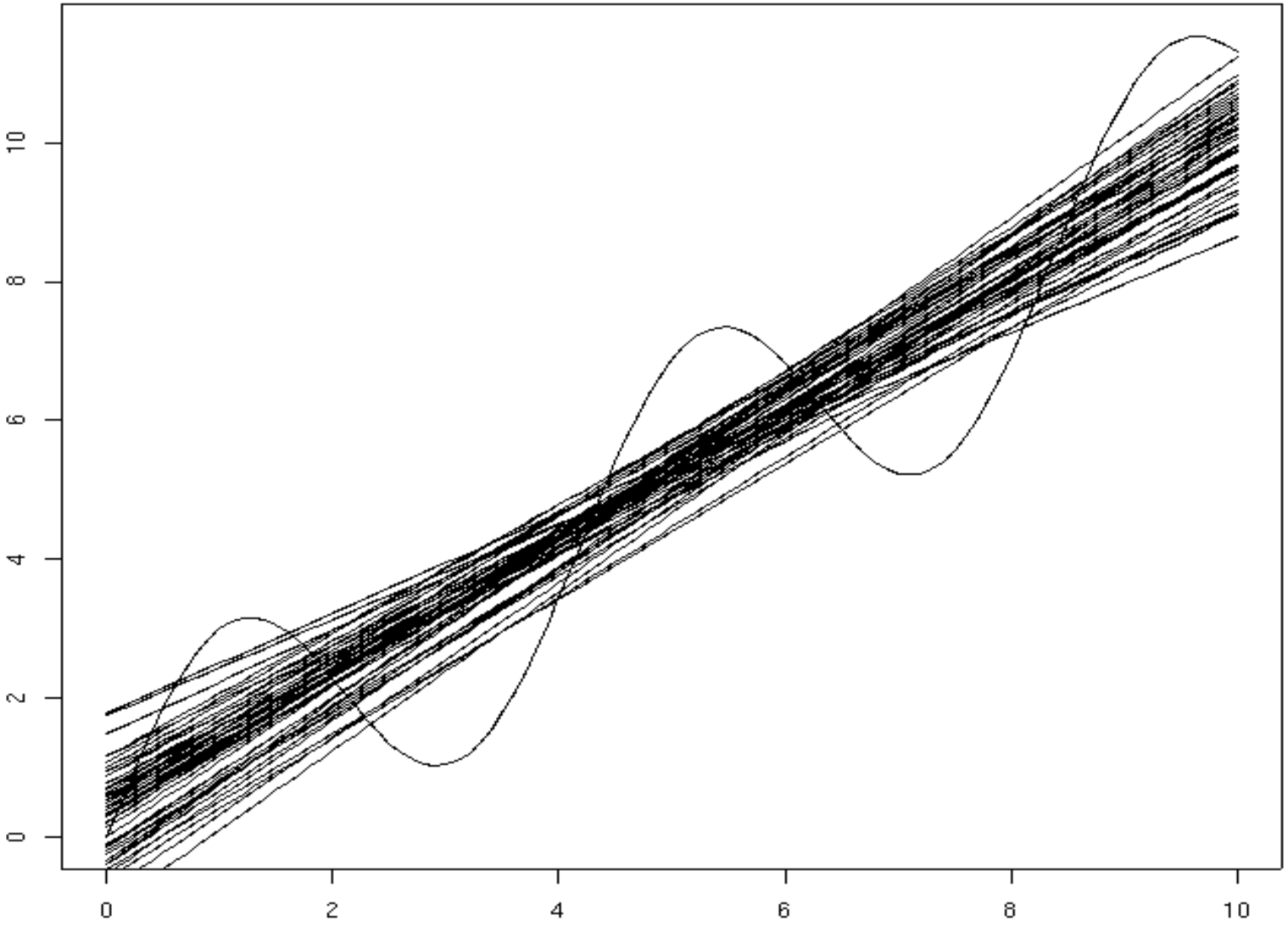
- **Bias** (or systemic error): $[h(\mathbf{x}) - f(\mathbf{x})]$

  Is associated with the class of hypotheses we are considering. It describes the *average* error of h(**x**).

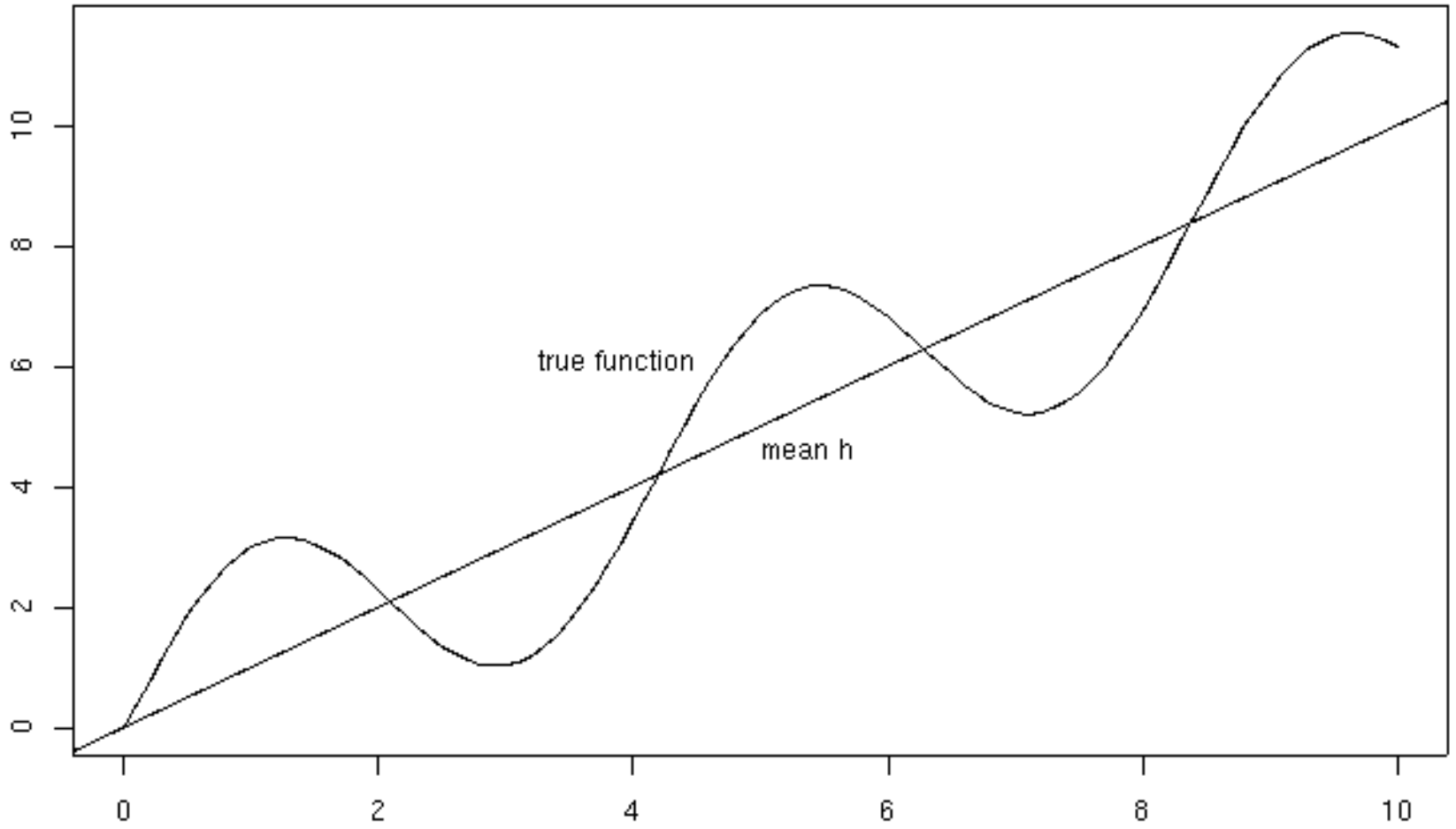
- **Noise**: $E_P\left[(y - f(\mathbf{x}))^2\right] = E[\varepsilon^2] = \sigma^2$

  Is due to the problem at hand and cannot be avoided. It describes how much y varies from f(**x**)

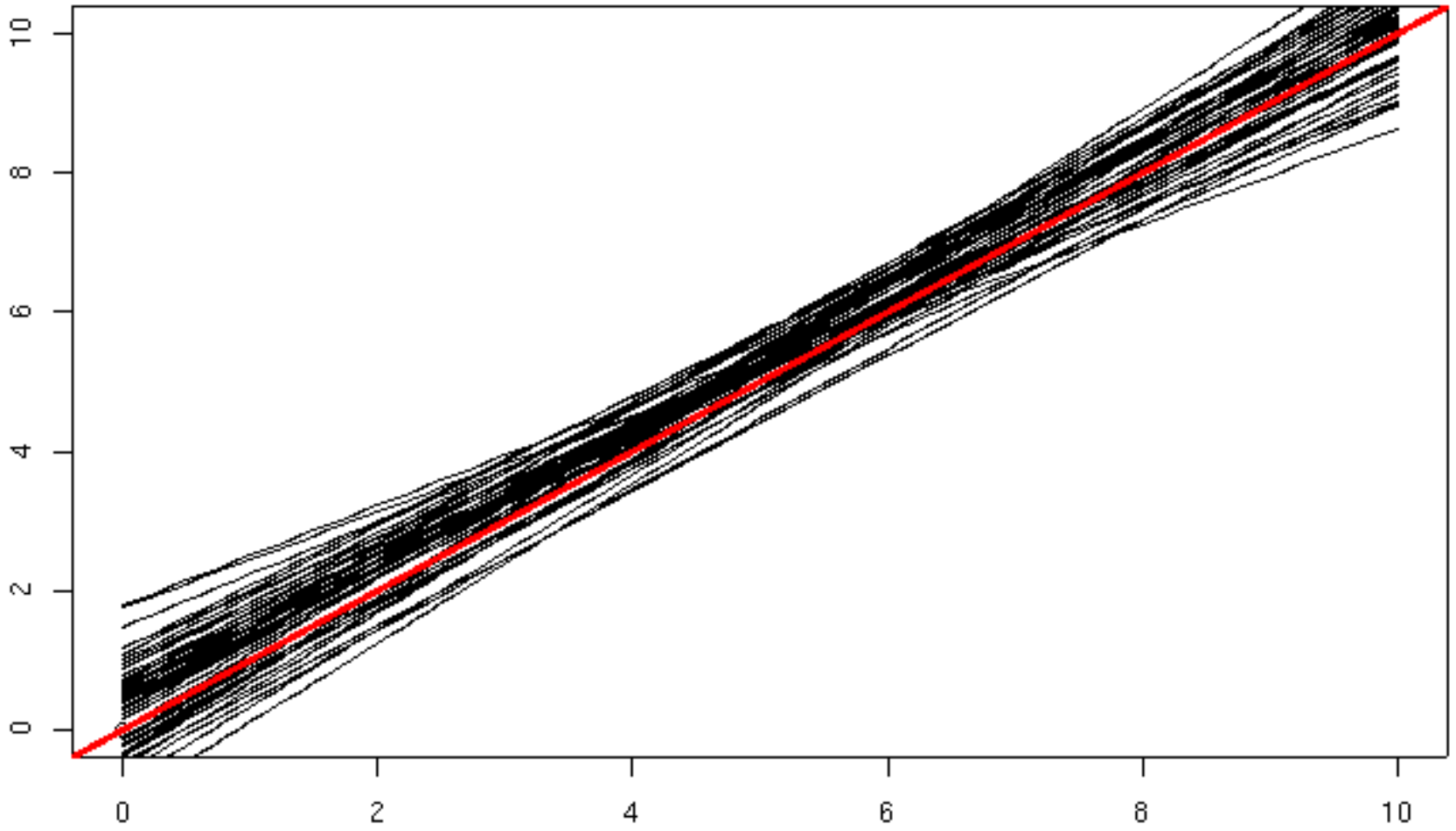
50 fits (20 examples each)



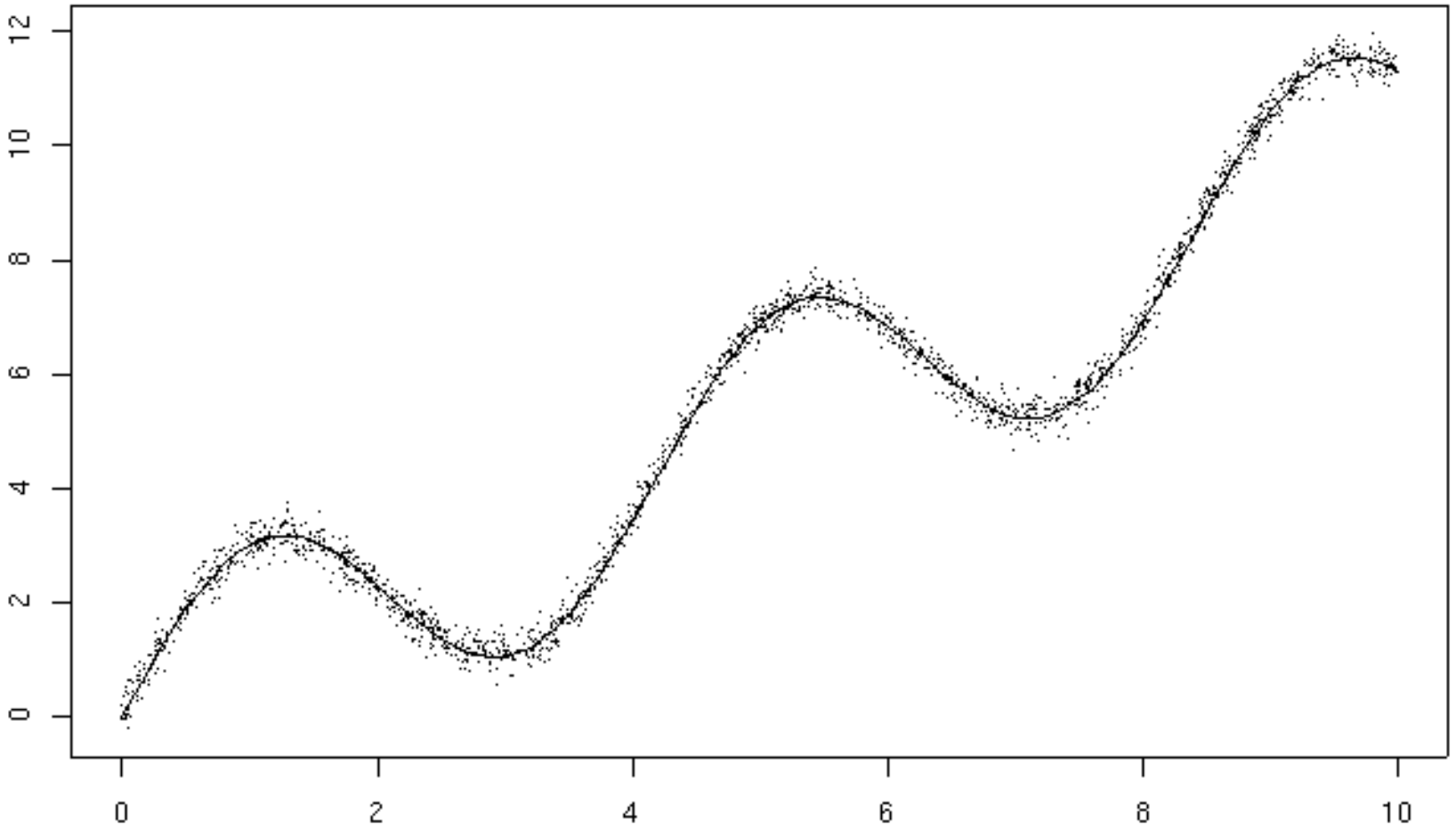
Bias



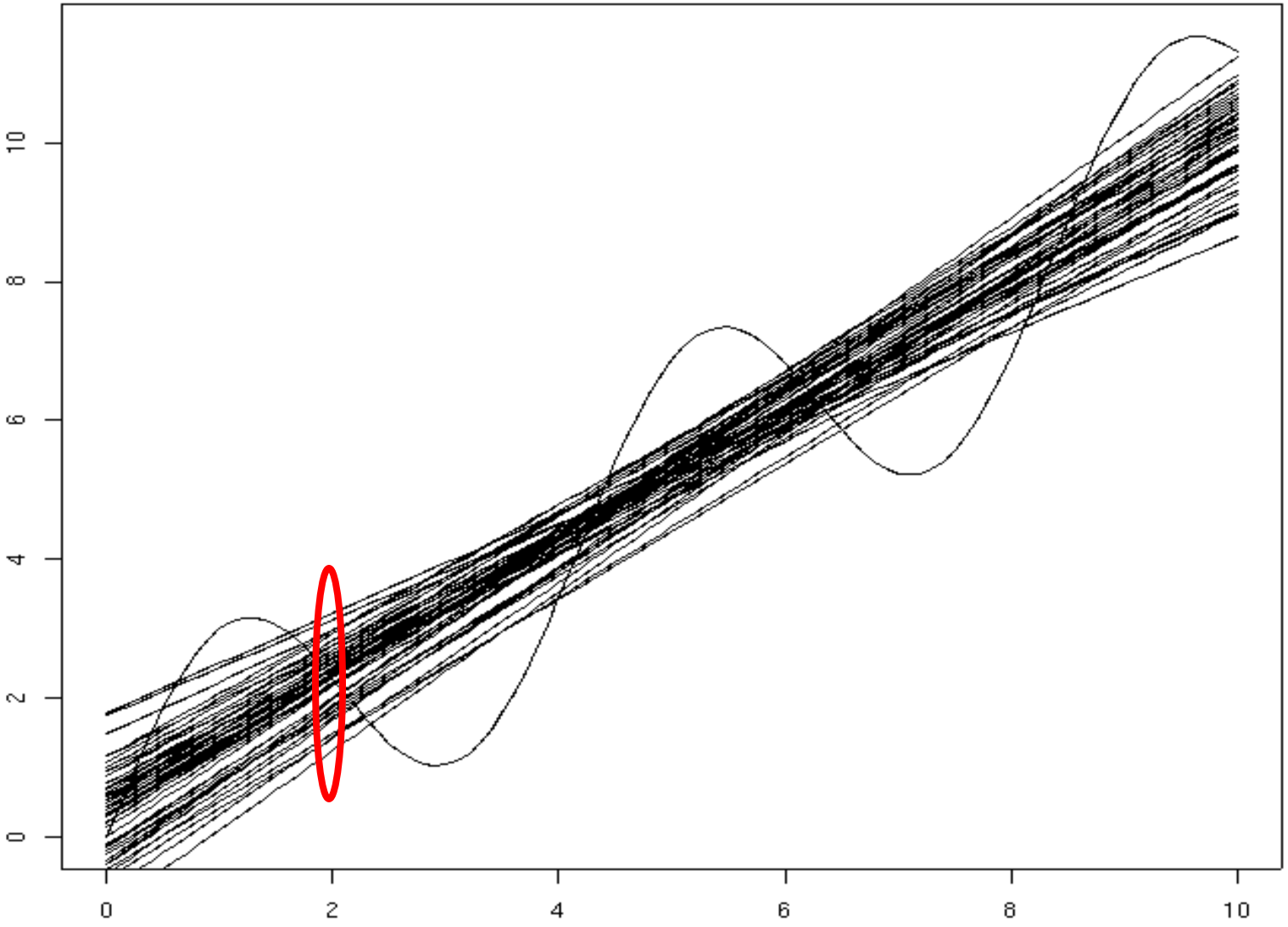
Variance



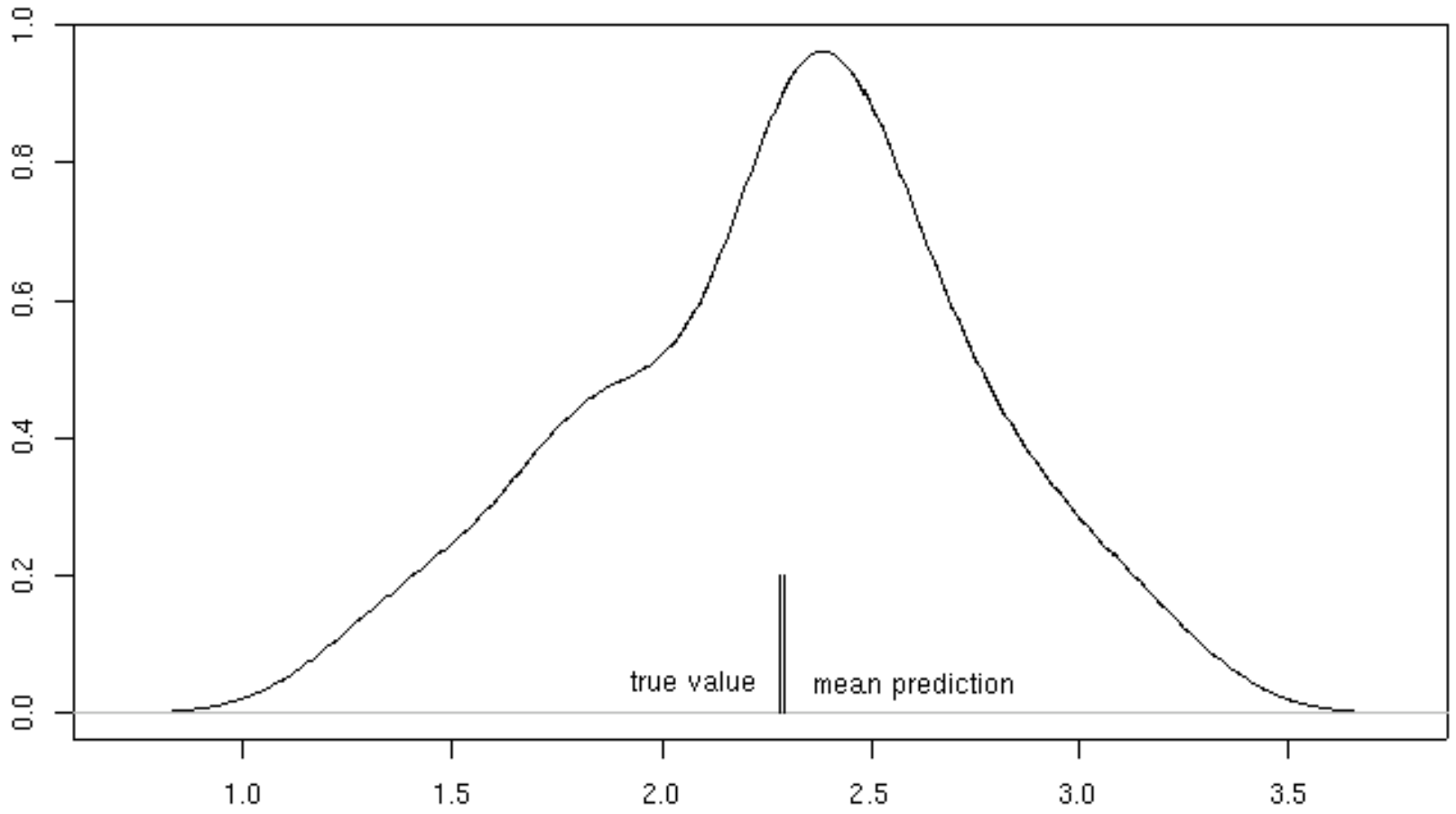
Noise



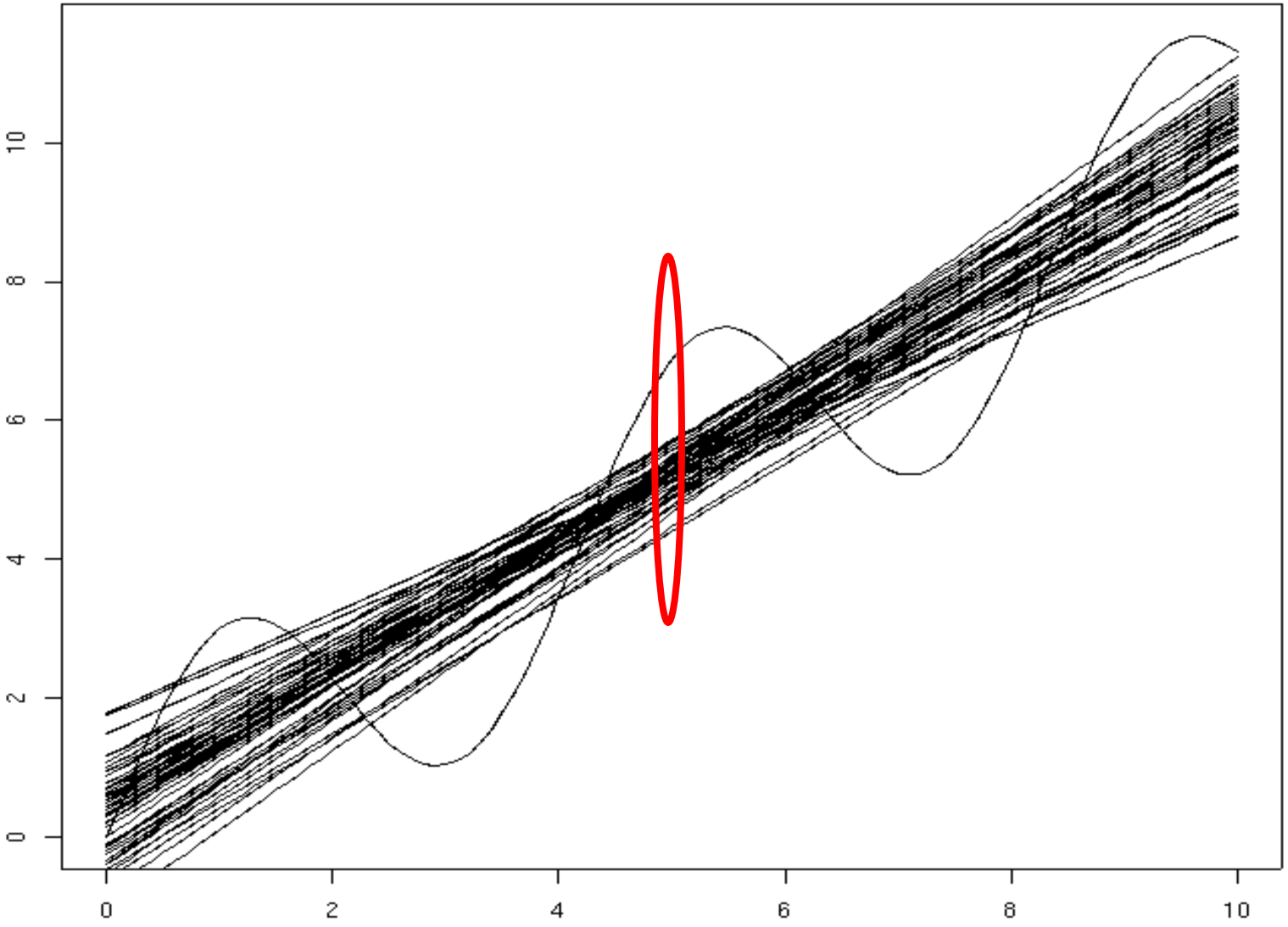
50 fits (20 examples each)



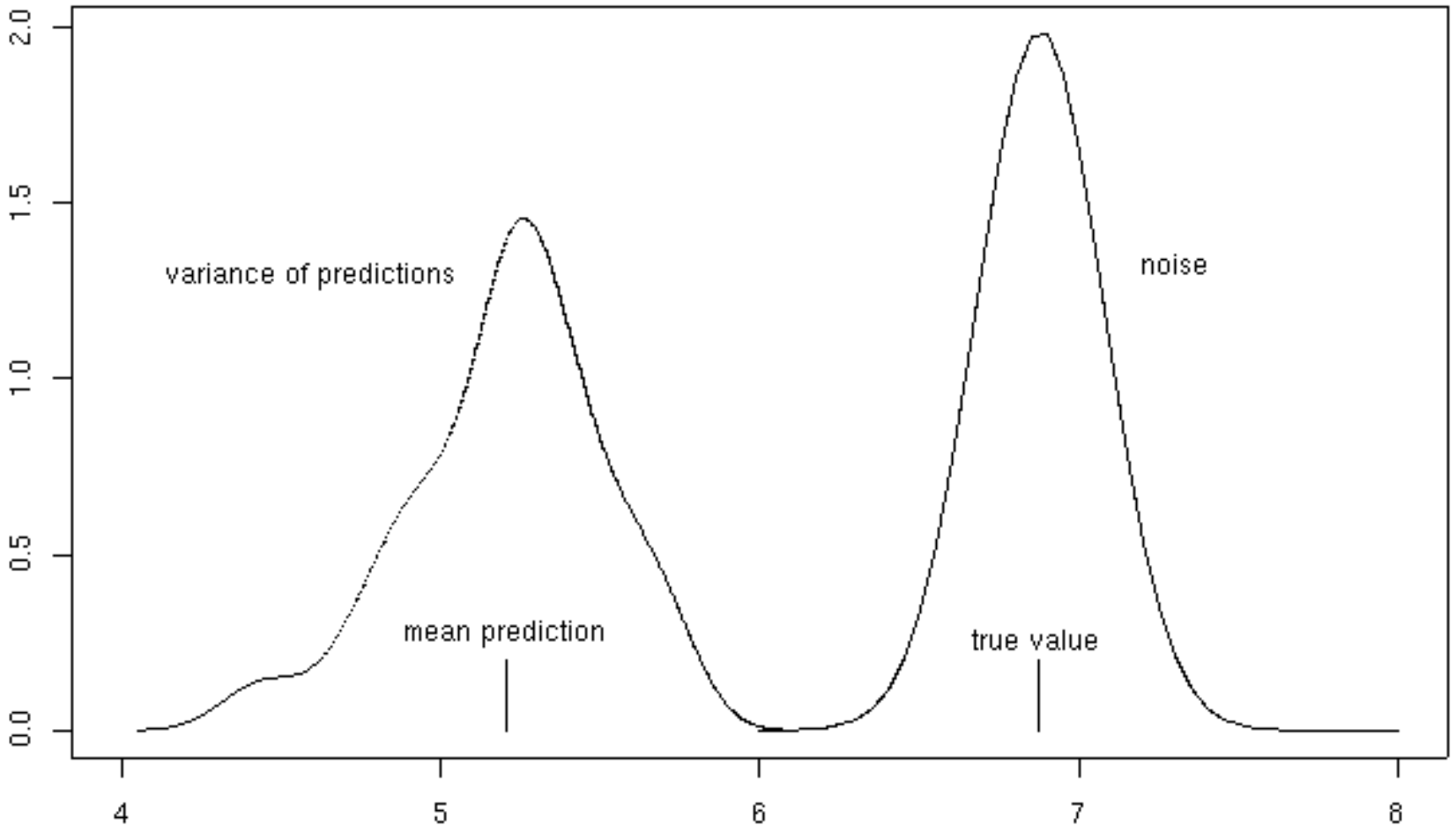
Distribution of predictions at x=2.0



50 fits (20 examples each)



Distribution of predictions at x=5.0



Bias-variance trade-off

- Consider fitting a logistic regression LTU to a data set vs. fitting a large neural net.
- Which one do you expect to have higher bias? Higher variance?
- Typically, *bias* comes from not having good hypotheses in the considered class
- *Variance* results from the hypothesis class containing too many hypotheses
- Hence, we are faced with a *trade-off:* choose a more expressive class of hypotheses, which will generate higher variance, or a less expressive class, which will generate higher bias.

Source of bias

- Inability to represent certain decision boundaries
  - E.g., linear threshold units, naïve Bayes, decision trees
- Incorrect assumptions
  - E.g, failure of independence assumption in naïve Bayes
- Classifiers that are "too global" (or, sometimes, too smooth)
  - E.g., a single linear separator, a small decision tree.

If the bias is high, the model is *underfitting* the data.

Source of variance

- Statistical sources
  - Classifiers that are "too local" and can easily fit the data
    - E.g., nearest neighbor, large decision trees
- Computational sources
  - Making decision based on small subsets of the data
    - E.g., decision tree splits near the leaves
  - Randomization in the learning algorithm
    - E.g., neural nets with random initial weights
  - Learning algorithms that make sharp decisions can be unstable (e.g. the decision boundary can change if one training example changes)

If the variance is high, the model is *overfitting* the data

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
  - $S' = \{\mathbf{x} \mid \mathbf{x}$ 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 (out of bag points)
- Compute predicted value $h_b(\mathbf{x})$ for $\mathbf{x} \in T_b$

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

Approximations in this Procedure

- Bootstrap replicates are not real data
- We ignore the noise
  - If we have multiple data points with the same **x** value, then we can estimate the noise
  - We can also estimate noise by pooling y values from nearby **x** values

Ensemble Learning Methods

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

$$\sum_l w_l h_l > \theta$$

Bagging: Bootstrap Aggregating

- For $b = 1, \ldots, B$ do
  - S_b = bootstrap replicate of S
  - Apply learning algorithm to S_b to learn h_b
- Classify new points by unweighted vote:

$$\frac{\sum_b w_b h_b}{B} > 0$$

Bagging

- Bagging makes predictions according to

$$y = \frac{1}{B}\sum_{b} h_b(\mathbf{x})$$

- Hence, bagging's predictions are $h(\mathbf{x})$

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
- This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

Bias-Variance Decomposition for Classification

- Can we extend the bias-variance decomposition to classification problems?
- Several extensions have been proposed; we will study the extension due to Pedro Domingos (2000a; 2000b)
- Domingos developed a unified decomposition that covers both regression and classification

Classification Problems: Noisy Channel Model

- Data points are generated by $y_i = n(f(\mathbf{x}_i))$, where
  - $f(\mathbf{x}_i)$ is the true class label of $\mathbf{x}_i$
  - $n(.)$ is a noise process that may change the true label $f(\mathbf{x}_i)$.
- Given a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, our learning algorithm produces an hypothesis h.
- Let $y^* = n(f(\mathbf{x}^*))$ be the observed label of a new data point $\mathbf{x}^*$.
  - $h(\mathbf{x}^*)$ is the predicted label.
  - The error ("loss") is defined as $L(h(\mathbf{x}^*), y^*)$

Loss Functions for Classification

- The usual loss function is 0/1 loss:

  $L(y',y)$ is 0 if $y' = y$ and 1 otherwise.

- Our goal is to decompose $E_p[L(h(\mathbf{x}^*), y^*)]$ into bias, variance, and noise terms

Discrete Equivalent of the Mean: The Main Prediction

- As before, we imagine that our observed training set S was drawn from some population according to $P(S)$
- Define the *main prediction* to be

$$y^m(\mathbf{x}^*) = \text{argmin}_{y'} E_P[\ L(y', h(\mathbf{x}^*))\ ]$$

- For 0/1 loss, the main prediction is the most common vote of $h(\mathbf{x}^*)$ (taken over all training sets S weighted according to $P(S)$)
- For squared error, the main prediction is $\underline{h(\mathbf{x}^*)}$

Bias, Variance, Noise

- Bias $B(\mathbf{x}^*) = L(y^m, f(\mathbf{x}^*))$
  - This is the loss of the main prediction with respect to the true label of $\mathbf{x}^*$
- Variance $V(\mathbf{x}^*) = E[\ L(h(\mathbf{x}^*), y^m)\ ]$
  - This is the expected loss of $h(\mathbf{x}^*)$ relative to the main prediction
- Noise $N(\mathbf{x}^*) = E[\ L(y^*, f(\mathbf{x}^*))\ ]$
  - This is the expected loss of the noisy observed value y^* relative to the true label of $\mathbf{x}^*$

Squared Error Loss

- These definitions give us the results we have already derived for squared error loss $L(y',y) = (y' - y)^2$
  - Main prediction $y^m = \underline{h(\mathbf{x}^*)}$
  - Bias^2: $L(\underline{h(\mathbf{x}^*)}, f(\mathbf{x}^*)) = (\underline{h(\mathbf{x}^*)} - f(\mathbf{x}^*))^2$
  - Variance:

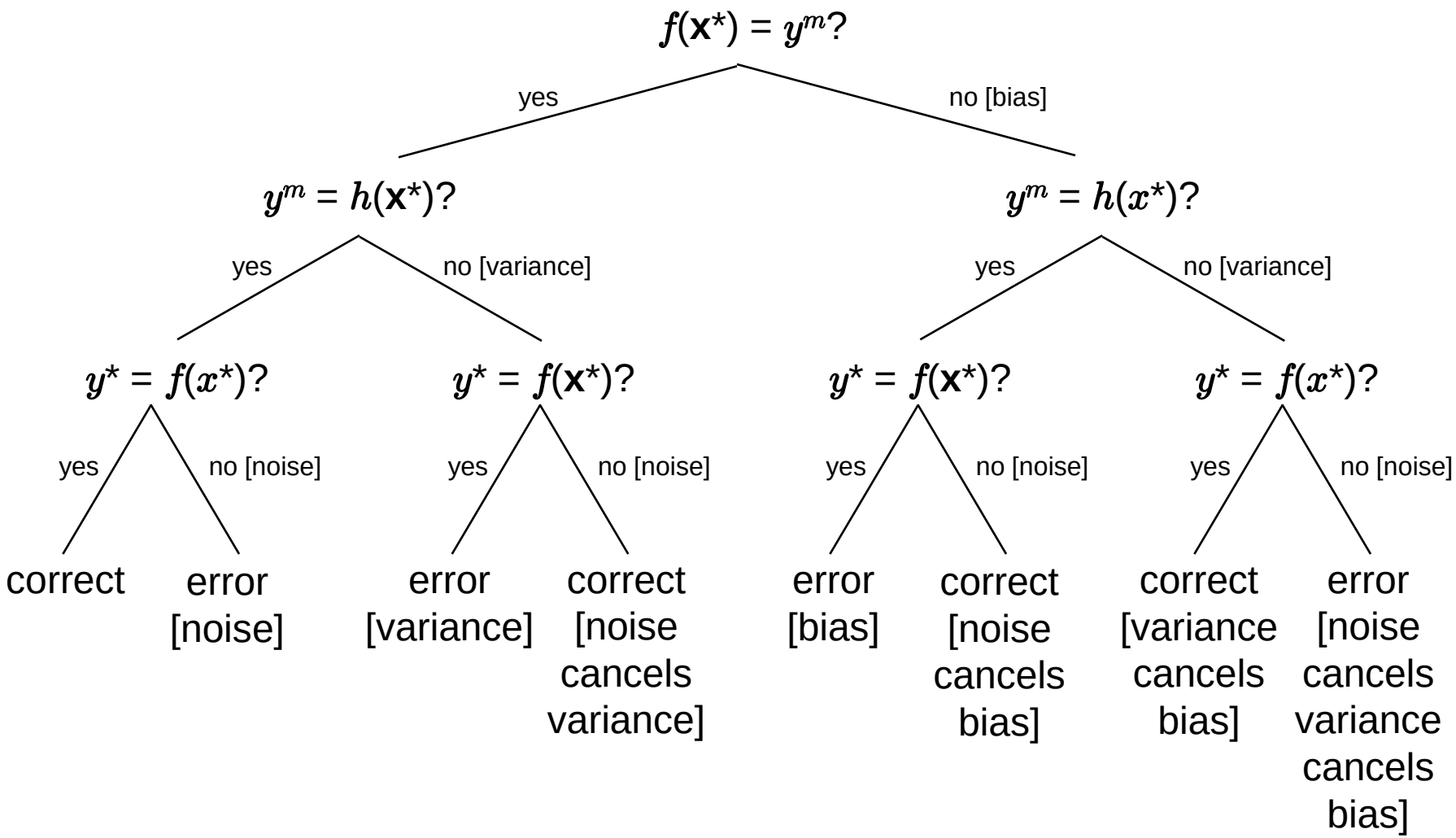
    $$E[\ L(h(\mathbf{x}^*), \underline{h(\mathbf{x}^*)})\ ] = E[\ (h(\mathbf{x}^*) - \underline{h(\mathbf{x}^*)})^2\ ]$$
  - Noise:

    $$E[\ L(y^*, f(\mathbf{x}^*))\ ] = E[\ (y^* - f(\mathbf{x}^*))^2\ ]$$

0/1 Loss for 2 classes

- There are three components that determine whether $y^* = h(\mathbf{x}^*)$
  - Noise: $y^* = f(\mathbf{x}^*)$?
  - Bias: $f(\mathbf{x}^*) = y^m$?
  - Variance: $y^m = h(\mathbf{x}^*)$?
- Bias is either 0 or 1, because neither $f(\mathbf{x}^*)$ nor y^m are random variables

Case Analysis of Error



Unbiased case

- Let $P(y^* \neq f(\mathbf{x}^*)) = N(\mathbf{x}^*) = \tau$
- Let $P(y^m \neq h(\mathbf{x}^*)) = V(\mathbf{x}^*) = \sigma$
- If ($f(\mathbf{x}^*) = y^m$), then we suffer a loss if exactly one of these events occurs:

$$L(h(\mathbf{x}^*), y^*) = \tau(1-\sigma) + \sigma(1-\tau)$$
$$= \tau + \sigma - 2\tau\sigma$$
$$= N(\mathbf{x}^*) + V(\mathbf{x}^*) - 2\, N(\mathbf{x}^*)\, V(\mathbf{x}^*)$$

Biased Case

- Let $P(y^* \neq f(\mathbf{x}^*)) = N(\mathbf{x}^*) = \tau$
- Let $P(y^m \neq h(\mathbf{x}^*)) = V(\mathbf{x}^*) = \sigma$
- If ($f(\mathbf{x}^*) \neq y^m$), then we suffer a loss if either both or neither of these events occurs:

$$L(h(\mathbf{x}^*), y^*) = \tau\sigma + (1-\sigma)(1-\tau)$$
$$= 1 - (\tau + \sigma - 2\tau\sigma)$$
$$= B(\mathbf{x}^*) - [N(\mathbf{x}^*)+V(\mathbf{x}^*)-2N(\mathbf{x}^*)V(\mathbf{x}^*)]$$

Decomposition for 0/1 Loss (2 classes)

- We do not get a simple additive decomposition in the 0/1 loss case:

  $E[\ L(h(\mathbf{x}^*), y^*)\ ] =$

  if $B(\mathbf{x}^*) = 1$: $B(\mathbf{x}^*) - [N(\mathbf{x}^*) + V(\mathbf{x}^*) - 2\ N(\mathbf{x}^*)\ V(\mathbf{x}^*)]$

  if $B(\mathbf{x}^*) = 0$: $B(\mathbf{x}^*) + [N(\mathbf{x}^*) + V(\mathbf{x}^*) - 2\ N(\mathbf{x}^*)\ V(\mathbf{x}^*)]$

- In biased case, noise and variance reduce error; in unbiased case, noise and variance increase error

Summary of 0/1 Loss

- A good classifier will have low bias, in which case the expected loss will approximately equal the variance
- The interaction terms will usually be small, because both noise and variance will usually be < 0.2, so the interaction term 2 $V(\mathbf{x}^*)$ $N(\mathbf{x}^*)$ will be < 0.08

0/1 Decomposition in Practice

- In the noise-free case:

  $E[\ L(h(\mathbf{x}^*), y^*)\ ] =$

  if $B(\mathbf{x}^*) = 1$: $B(\mathbf{x}^*) - V(\mathbf{x}^*)$

  if $B(\mathbf{x}^*) = 0$: $B(\mathbf{x}^*) + V(\mathbf{x}^*)$

- It is usually hard to estimate $N(\mathbf{x}^*)$, so we will use this formula

Decomposition over an entire data set

- Given a set of test points

$$T = \{(\mathbf{x}_1^*, y_1^*), \cdots, (\mathbf{x}_n^*, y_n^*)\}$$

we want to decompose the average loss:

$$\underline{L} = \frac{1}{n} \sum_i E\left[L(h(\mathbf{x}_i^*), y_i^*)\right]$$

- We will write it as

$$\underline{L} = \underline{B} + \underline{Vu} - \underline{Vb}$$

where $\underline{B}$ is the average bias, $\underline{Vu}$ is the average unbiased variance, and $\underline{Vb}$ is the average biased variance (We ignore the noise.)

- $\underline{Vu} - \underline{Vb}$ will be called *net variance*

Classification Problems: Overlapping Distributions Model

- Suppose at each point **x**, the label is generated according to a probability distribution $y \sim P(y|\mathbf{x})$
- The goal of learning is to discover this probability distribution
- The loss function $L(p,h) = KL(p,h)$ is the *Kullback-Liebler* divergence between the true distribution p and our hypothesis h.

Kullback-Leibler Divergence

- For simplicity, assume only two classes: $y \in \{0,1\}$
- Let p be the true probability $P(y=1|\mathbf{x})$ and h be our hypothesis for $P(y=1|\mathbf{x})$.
- The KL divergence is

$$KL(p,h) = p \log \frac{p}{h} + (1-p) \log \frac{1-p}{1-h}$$

Bias-Variance-Noise Decomposition for KL

- Goal: Decompose $E_S[\ KL(y, h)\ ]$ into noise, bias, and variance terms
- Compute the main prediction:

$$\underline{h} = \text{argmin}_u\ E_S[\ KL(u, h)\ ]$$

- This turns out to be the geometric mean:

$$\log \frac{\underline{h}}{1-\underline{h}} = E_S\left[\log \frac{h}{1-h}\right]$$

$$\underline{h} = \frac{1}{Z} \cdot \exp\left(E_S\left[\log h\right]\right)$$

Computing the Noise

- Obviously the best estimator h would be p. What loss would it receive?

$$
\begin{aligned}
E\left[KL(y,p)\right] &= E\left[y\log\frac{y}{p}+(1-y)\log\frac{1-y}{1-p}\right] \\
&= E\left[y\log y- \right. \\
&\qquad y\log p+ \\
&\qquad (1-y)\log(1-y)- \\
&\qquad \left.(1-y)\log(1-p)\right] \\
&= -p\log p-(1-p)\log(1-p) \\
&= H(p)
\end{aligned}
$$

Bias, Variance, Noise

- Variance: $E_S[\ KL(\underline{h}, h)\ ]$
- Bias: $KL(p, \underline{h})$
- Noise: $H(p)$
- Expected loss = Noise + Bias + Variance

$$E[\ KL(y, h)\ ] = H(p) + KL(p, \underline{h}) + E_S[\ KL(\underline{h}, h)\ ]$$

Consequences of this Definition

- If our goal is probability estimation and we want to do bagging, then we should combine the individual probability estimates using the geometric mean

$$\log \frac{\underline{h}}{1-\underline{h}} = E_S\left[\log \frac{h}{1-h}\right]$$

- In this case, bagging will produce pure variance reduction (as in regression)!

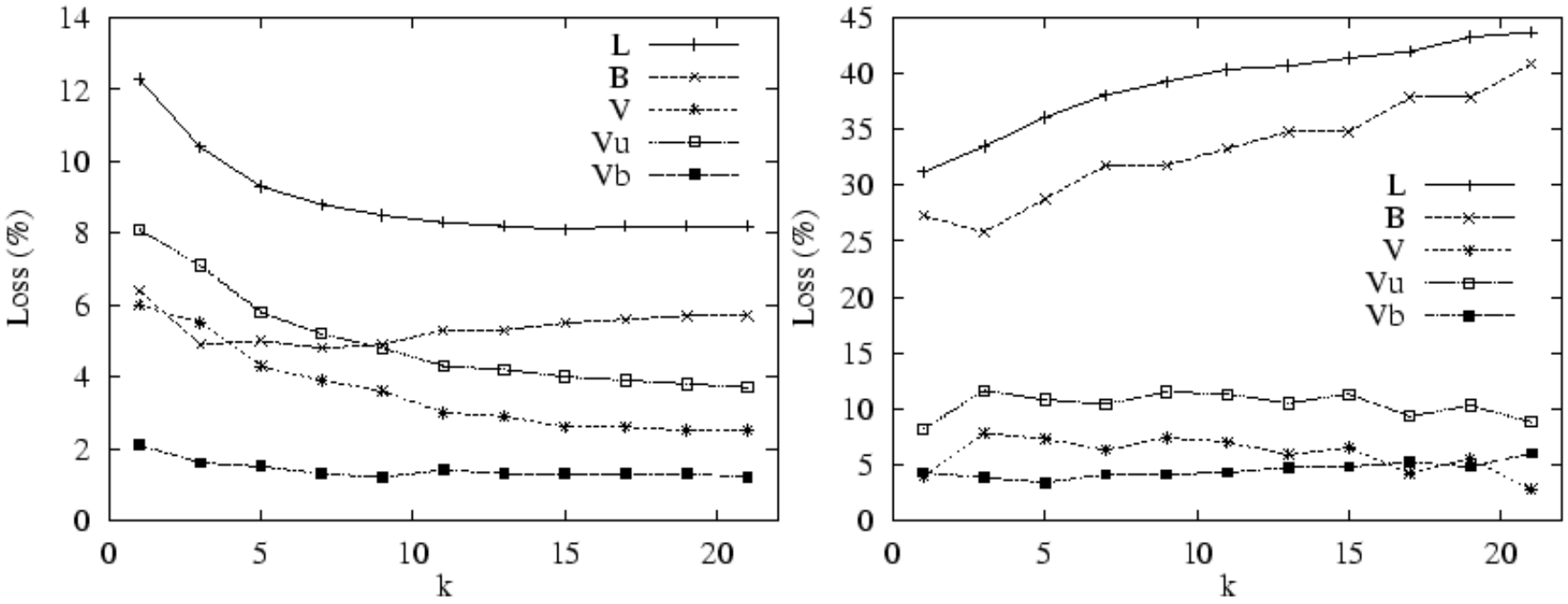
Experimental Studies of Bias and Variance

- Artificial data: Can generate multiple training sets S and measure bias and variance directly
- Benchmark data sets: Generate bootstrap replicates and measure bias and variance on separate test set

Algorithms to Study

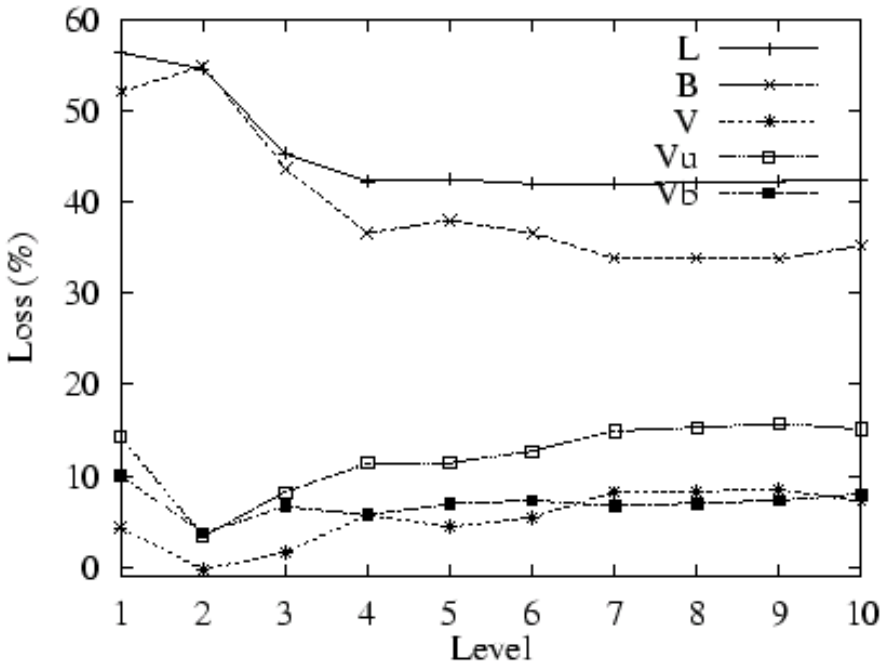
- K-nearest neighbors: What is the effect of K?
- Decision trees: What is the effect of pruning?

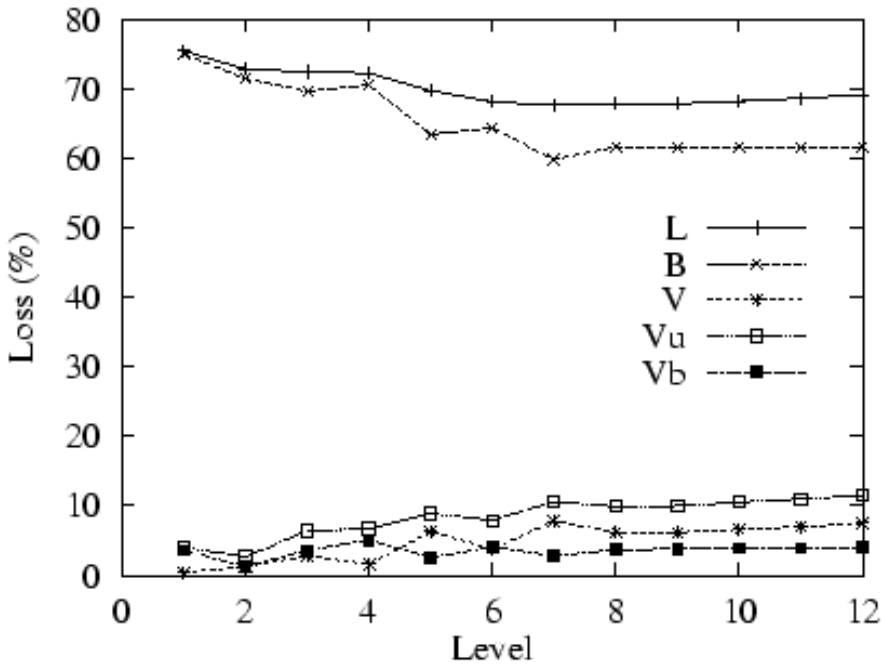
K-nearest neighbor (Domingos, 2000)



- Chess (left): Increasing K primarily reduces Vu
- Audiology (right): Increasing K primarily increases B.

Size of Decision Trees





- Glass (left), Primary tumor (right): deeper trees have lower B, higher Vu

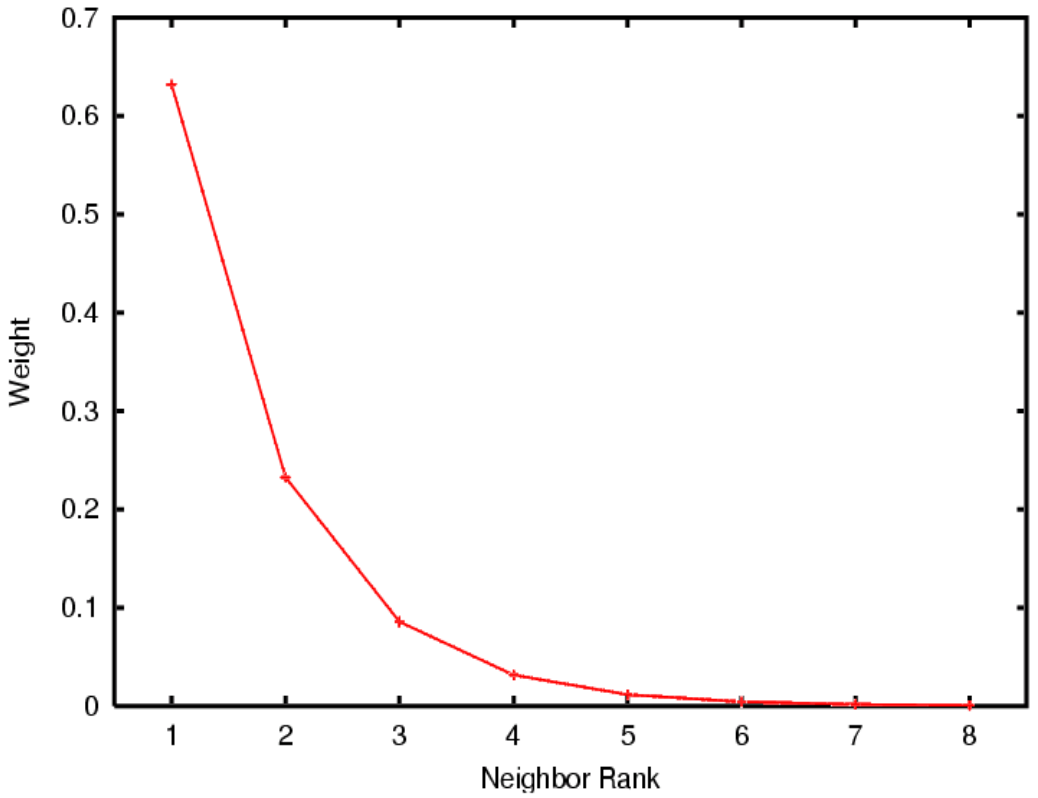
B/V Analysis of Bagging

- Under the bootstrap assumption, bagging reduces only variance
  - Removing Vu reduces the error rate
  - Removing Vb increases the error rate
- Therefore, bagging should be applied to low-bias classifiers, because then Vb will be small
- Reality is more complex!

Bagging Nearest Neighbor

Bagging first-nearest neighbor is equivalent (in the limit) to a weighted majority vote in which the k-th neighbor receives a weight of

exp(-(k-1)) – exp(-k)

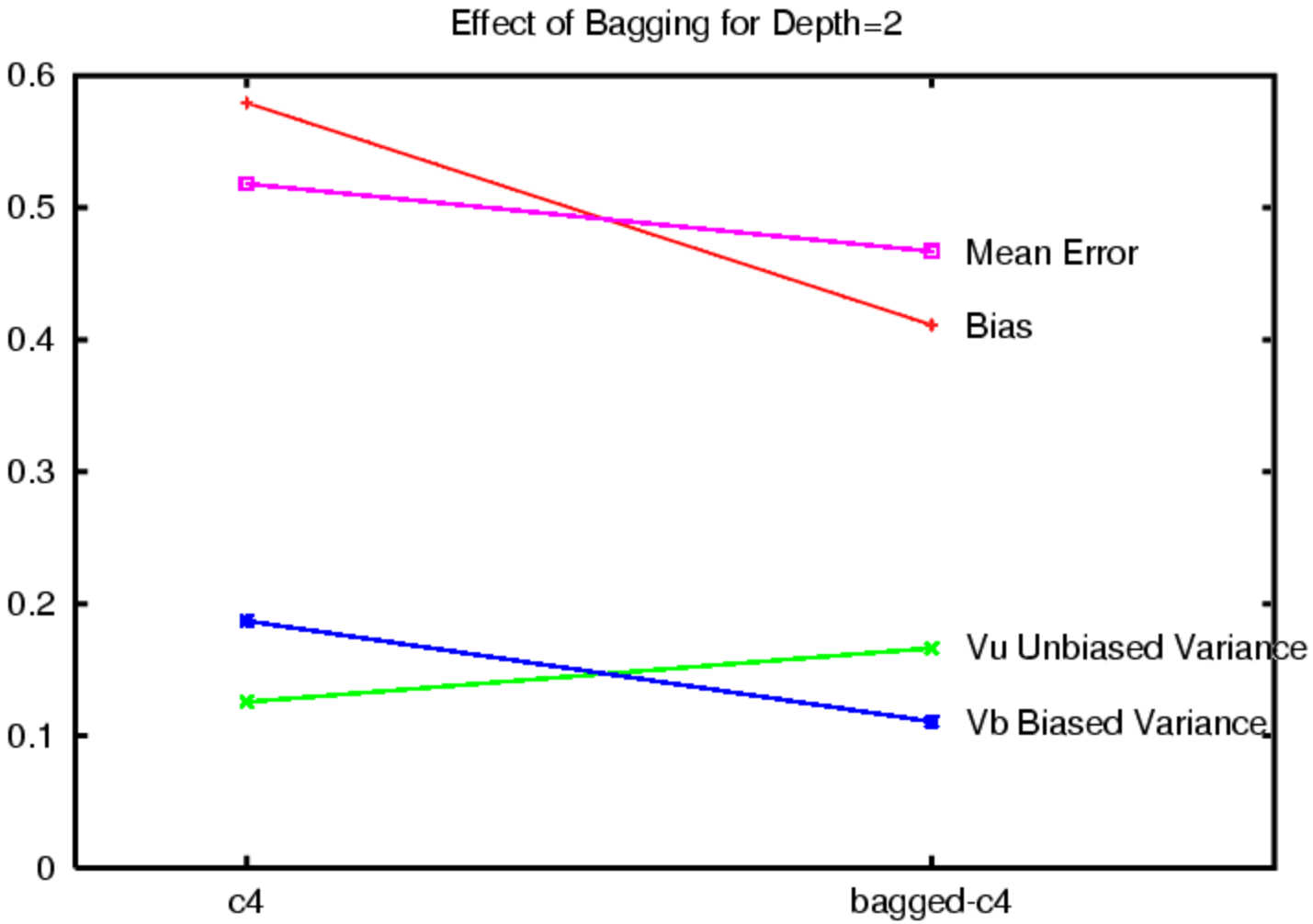


Since the first nearest neighbor gets more than half of the vote, it will always win this vote. Therefore, Bagging 1-NN is equivalent to 1-NN.

Bagging Decision Trees

- Consider unpruned trees of depth 2 on the Glass data set. In this case, the error is almost entirely due to bias
- Perform 30-fold bagging (replicated 50 times; 10-fold cross-validation)
- What will happen?

Bagging Primarily Reduces Bias!



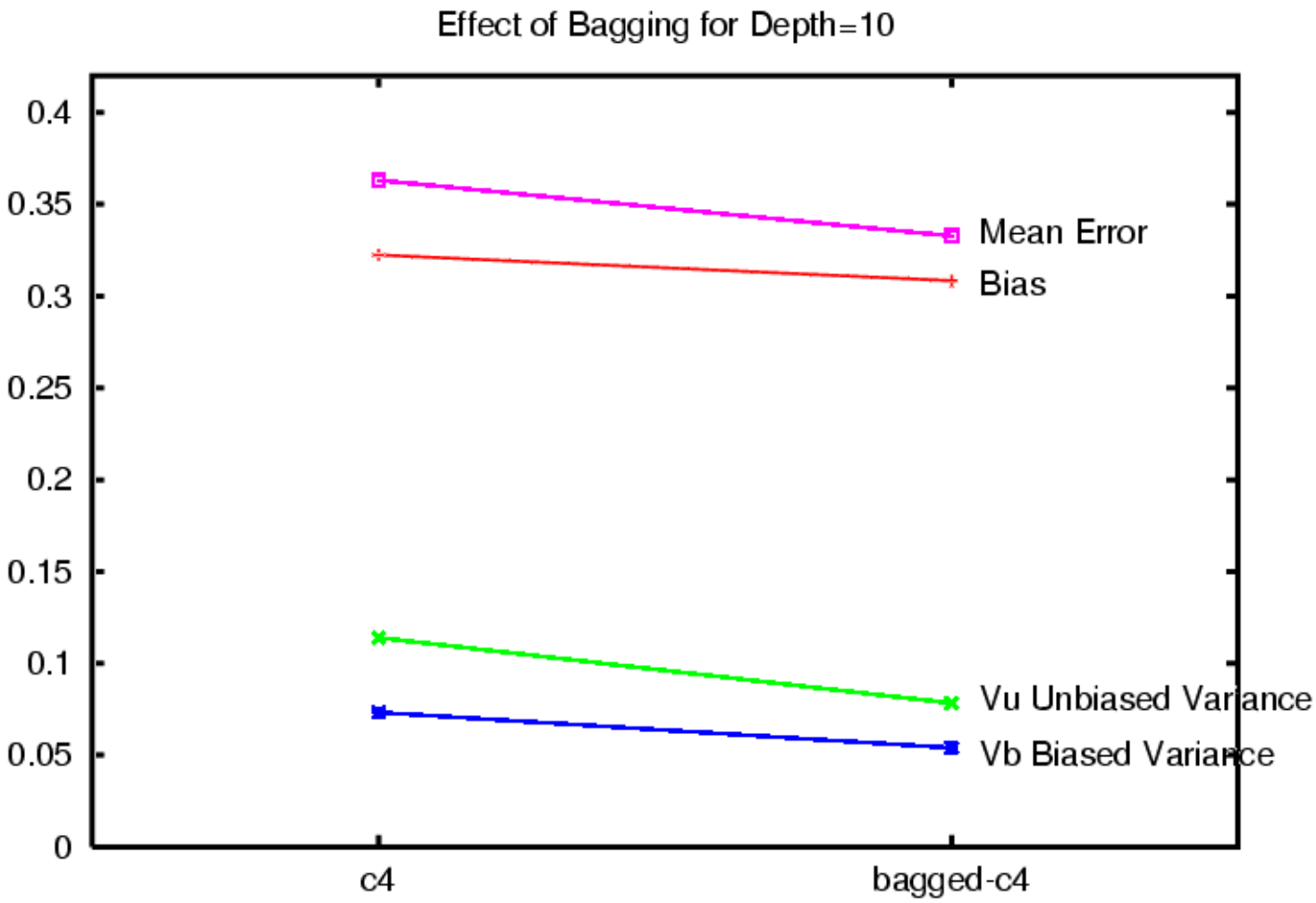
Questions

- Is this due to the failure of the bootstrap assumption in bagging?
- Is this due to the failure of the bootstrap assumption in estimating bias and variance?
- Should we also think of Bagging as a simple additive model that expands the range of representable classifiers?

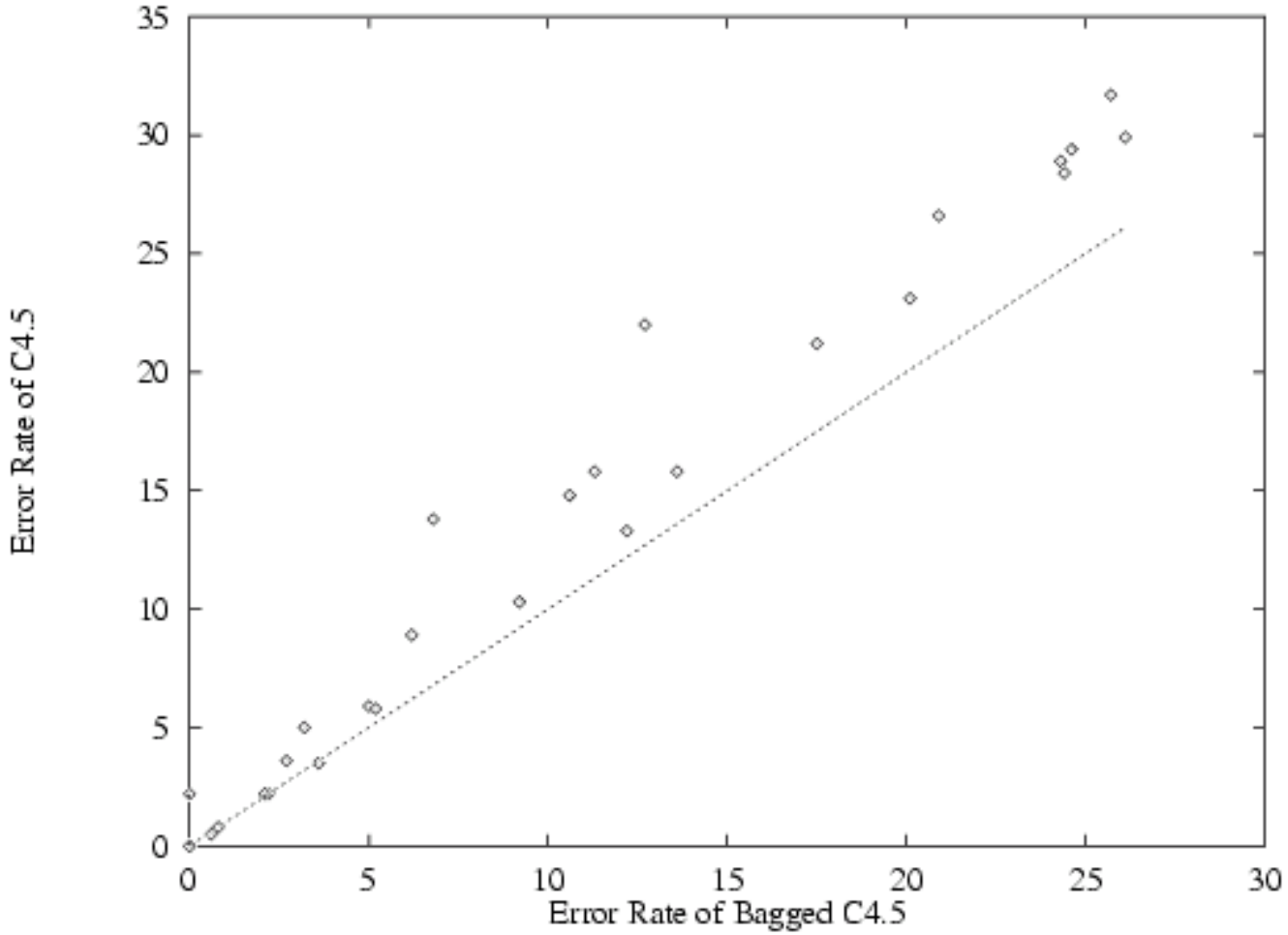
Bagging Large Trees?

- Now consider unpruned trees of depth 10 on the Glass dataset. In this case, the trees have much lower bias.
- What will happen?

Answer: Bagging Primarily Reduces Variance



Bagging Decision Trees (Freund & Schapire)



Boosting

Notes adapted from Rob Schapire

http://www.cs.princeton.edu/~schapire

Example: Spam Filtering

- problem: filter out spam (junk email)
- gather large collection of examples of spam and non-spam:

From: yoav@att.com Rob, can you review a paper... non-spam

From: xa412@hotmail.com Earn money without working!!! spam

…

- main observation:
  - easy to find "rules of thumb" that are "often" correct (If "buy now" appears, predict spam)
  - hard to find a single rule that is very highly accurate

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of emails
- obtain rule of thumb
- apply to 2nd subset of emails
- obtain 2nd rule of thumb
- repeat T times

Details

- how to *choose examples* on each round?
  - concentrate on "hardest" examples (those most often misclassified by previous rules of thumb)
- how to *combine* rules of thumb into single prediction rule?
  - take (weighted) majority vote of rules of thumb

Boosting

- boosting = general method of converting rough rules of thumb into highly accurate prediction rule
- more technically:
  - given "weak" learning algorithm that can consistently find classifier with error no more than 1/2-γ
  - a boosting algorithm can *provably* construct single classifier with no more than ε error (ε, γ small)

Boosting

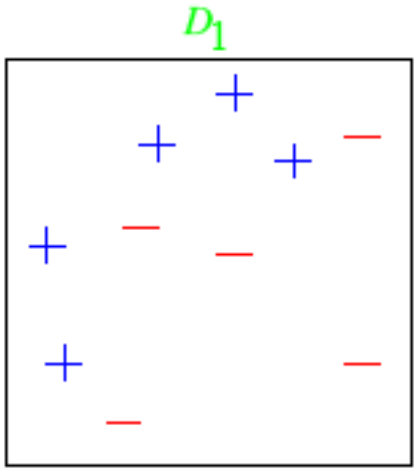
Input: a set S, of m labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$,
labels $y_i \in Y = \{1, \ldots, K\}$
Learn (a learning algorithm)
a constant L.

[1] **initialize for all** i: $w_1(i) := 1/m$ — *initialize the weights*
[2] **for** $\ell = 1$ **to** L **do**
[3] **for all** i: $p_\ell(i) := w_\ell(i)/(\Sigma_i w_\ell(i))$ — *compute normalized weights*
[4] $h_\ell := \text{Learn}(p_\ell)$ — *call* Learn *with normalized weights.*
[5] $\epsilon_\ell := \Sigma_i p_\ell(i)[h_\ell(x_i) \neq y_i]$ — *calculate the error of* h_ℓ
[7] **if** $\epsilon_\ell > 1/2$ **then**
[8] $L := \ell - 1$
[9] **exit**
[10] $\beta_\ell := \epsilon_\ell/(1 - \epsilon_\ell)$
[11] **for all** i: $w_{\ell+1}(i) := w_\ell(i)\beta_\ell^{1-[h_\ell(x_i)\neq y_i]}$ — *compute new weights*

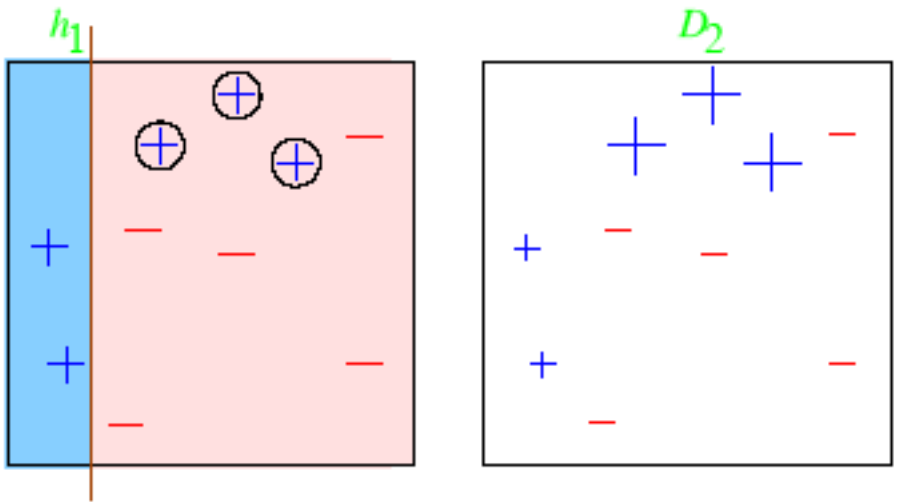
Output: $h_f(x) = \underset{y \in Y}{\text{argmax}} \sum_{\ell=1}^{L} \left(\log \frac{1}{\beta_\ell}\right) [\![h_\ell(x) = y]\!]$

Toy Example

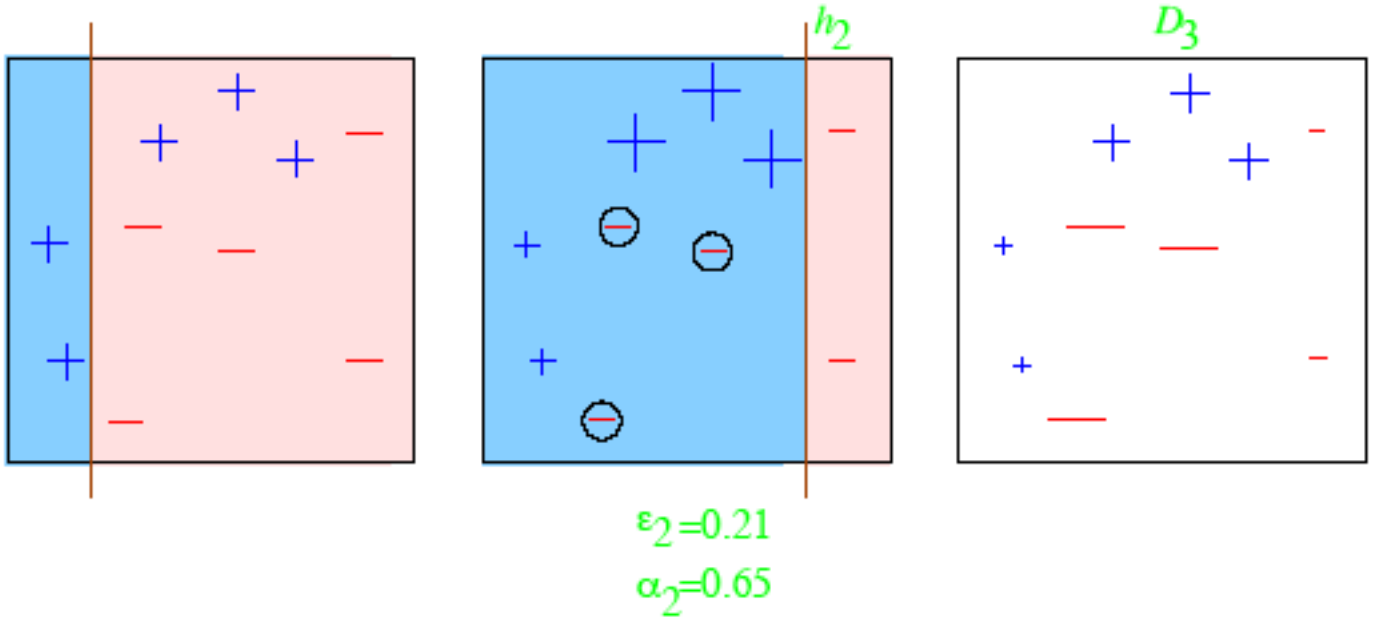
- weak classifiers = vertical or horizontal half-planes



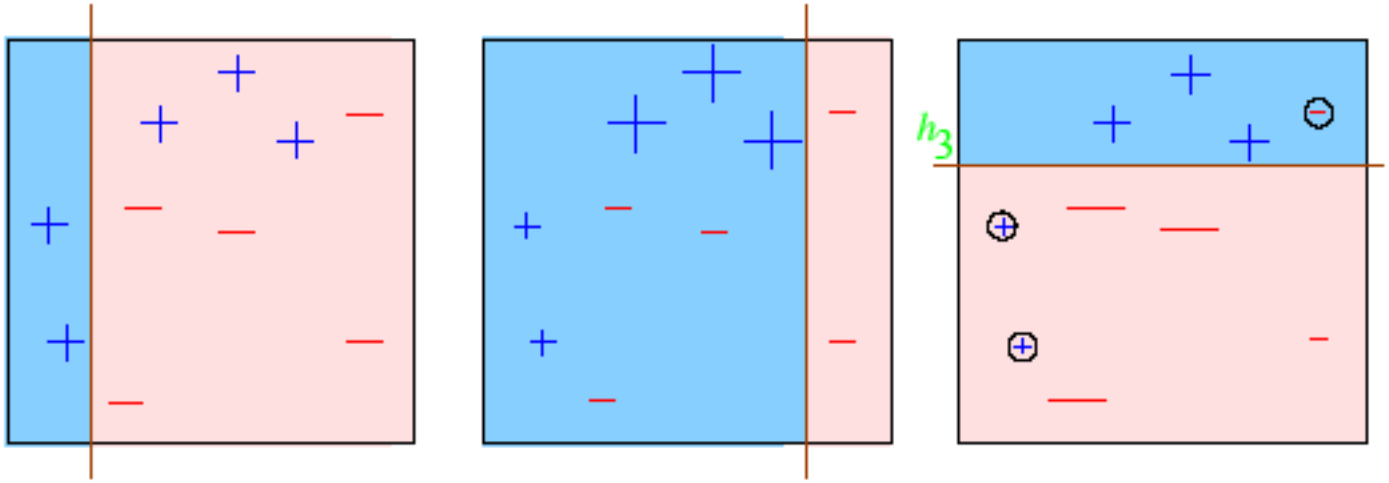
Round 1



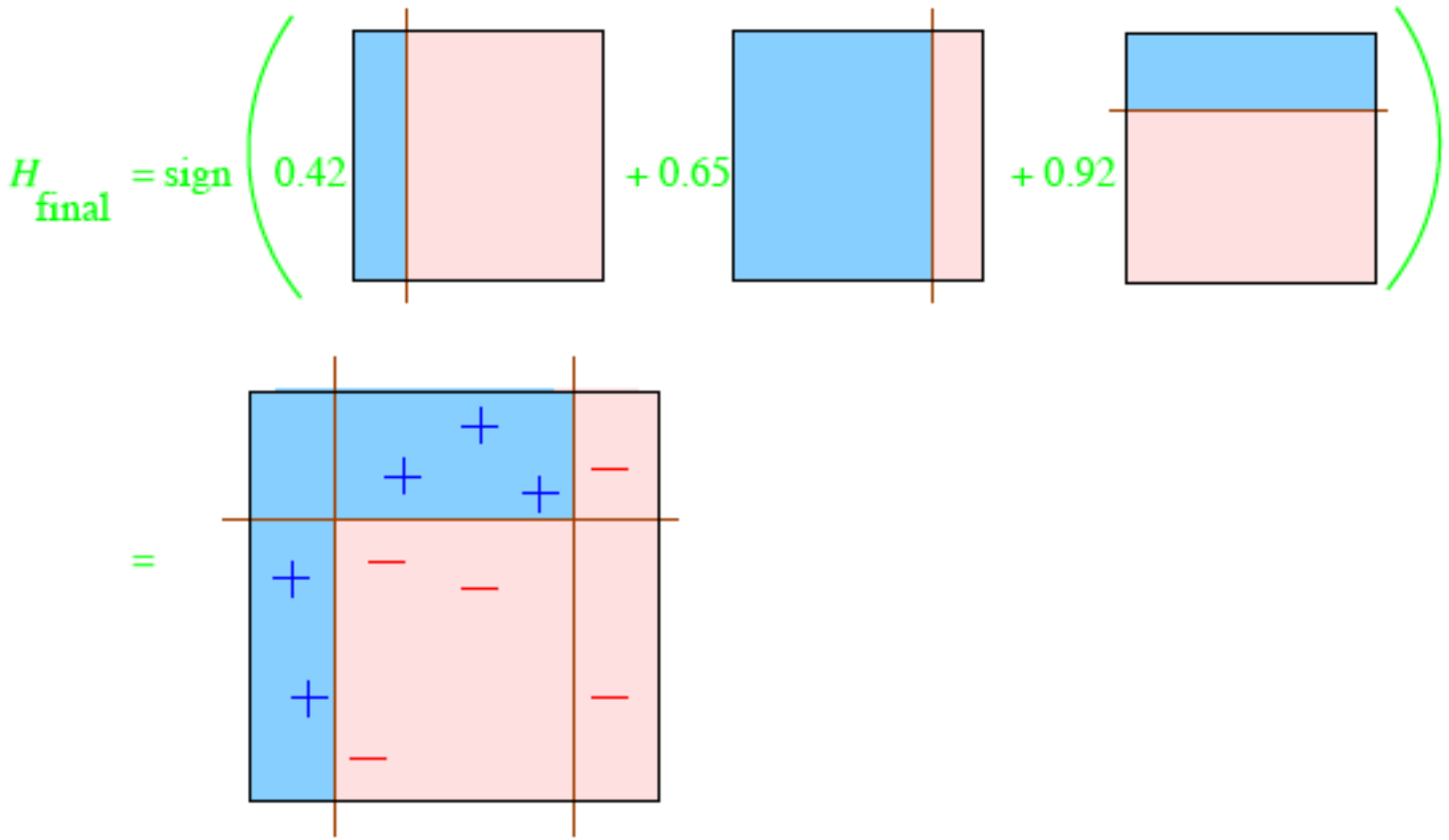
Round 2



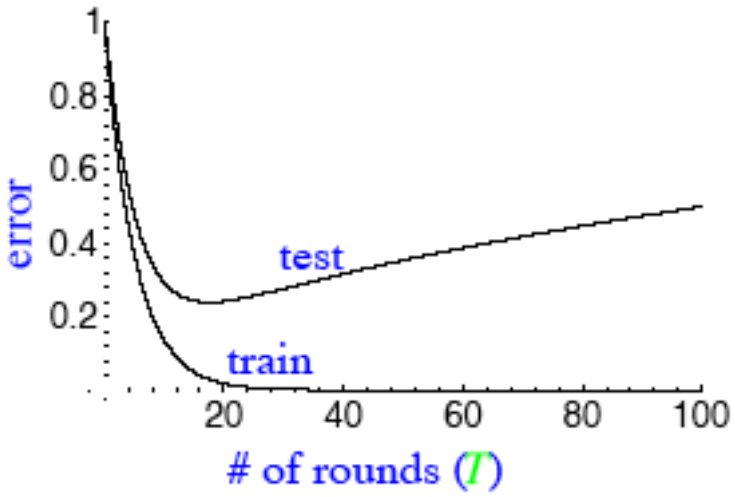
Round 3



Final Classifier

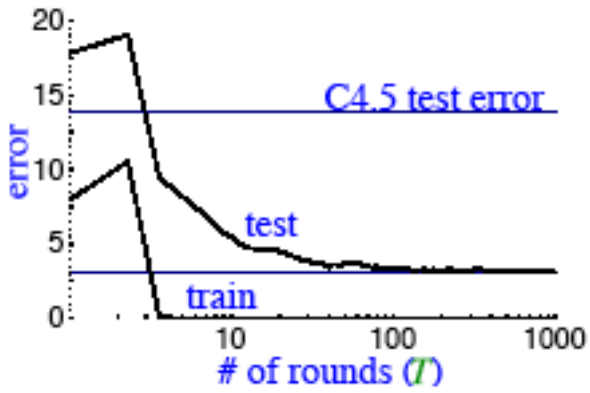


Generalization Error, Part 1



- expect:
  - training error to continue to drop (or reach zero)
  - test error to *increase* when H_{final} becomes "too complex"
    - "Occam's razor"
    - overfitting
      - hard to know when to stop training

Actual Typical Run



(boosting C4.5 on "letter" dataset)

- test error does *not* increase, even after 1000 rounds
  - (total size > 2M nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

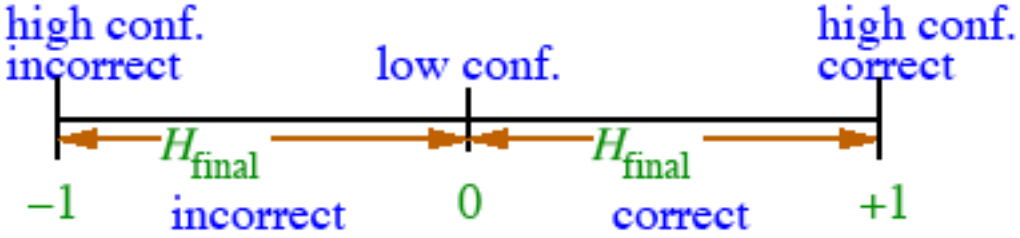
- Occam's razor *wrongly* predicts "simpler" rule is better

Better Story: Margins

- key idea (Schapire, Freund, Bartlett, Lee)
  - training error only measures whether classifications are right or wrong
  - should also consider *confidence* of classifications
- can write $H_{final}(x) = \text{sign}(f(x))$

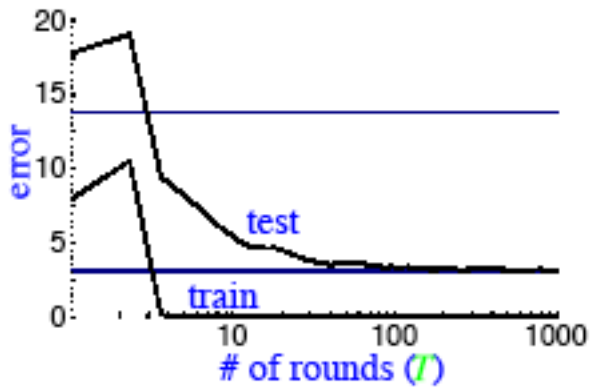
  where $f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, +1]$

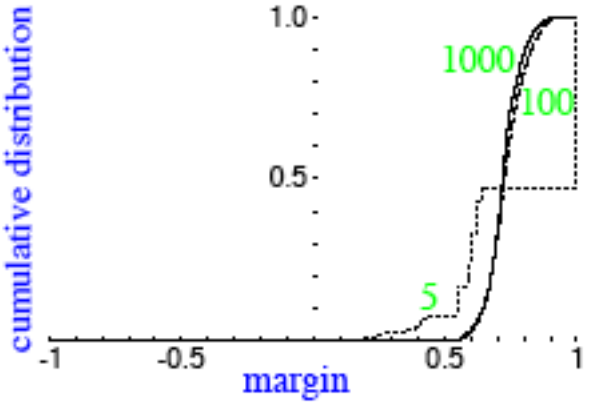
- define *margin* of example (x,y) to be y f(x) = measure of confidence of classifications



Empirical Evidence

- *margin distribution* = cumulative distribution of margins of training examples





	# rounds 5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Theoretical Evidence

- if all training examples have *large margins*, then can *approximate* final classifier by a much *small* classifier
  - (similar to how polls can predict outcome of a not-too-close election)
- can use this fact to prove that large margins imply better test error, *regardless* of number of weak classifiers
- can also prove that *boosting tends to increase margins* of training examples by concentrating on those with smallest margin
- so: although final classifier is getting *larger*, *margins* are likely to be *increasing*, so final classifier is actually getting close to a *simpler* classifier, driving *down* the test error

Practical Advantages

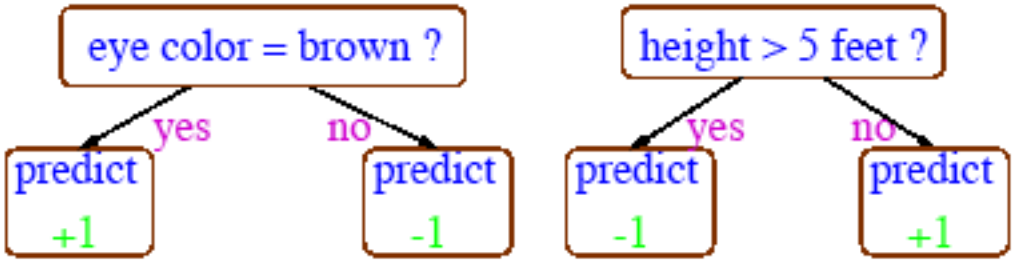
- *fast*
- *simple* and easy to program
- *no parameters* to tune (except T)
- *flexible*--- can combine with *any* learning algorithm
- *no prior knowledge* needed about weak learner
- *provably effective*, provided can consistently find rough rules of thumb
  - shift in mindset: goal now is merely to find classifiers barely better than random guessing
- *versatile*
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

Caveats

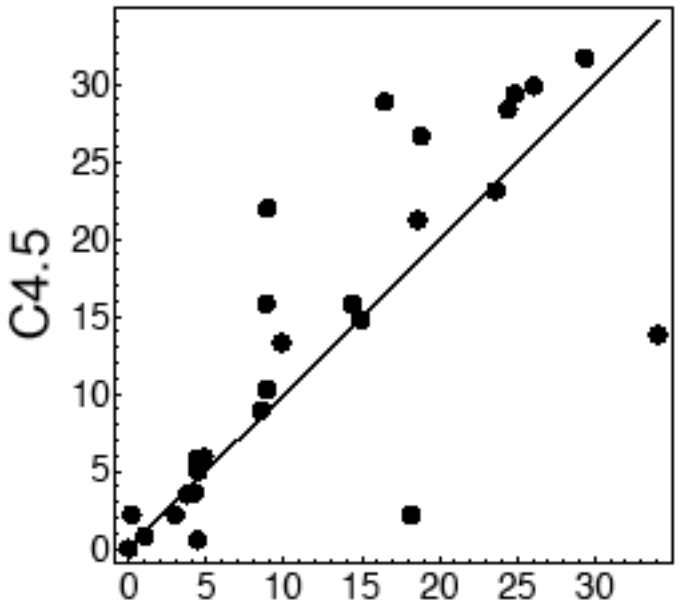
- performance of AdaBoost depends on *data* and *weak learner*
- consistent with theory, AdaBoost can *fail* if
  - weak classifier too complex
    - overfitting
  - weak classifiers too weak ($\gamma_t \rightarrow 0$ too quickly)
    - underfitting
    - low margins $\rightarrow$ overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

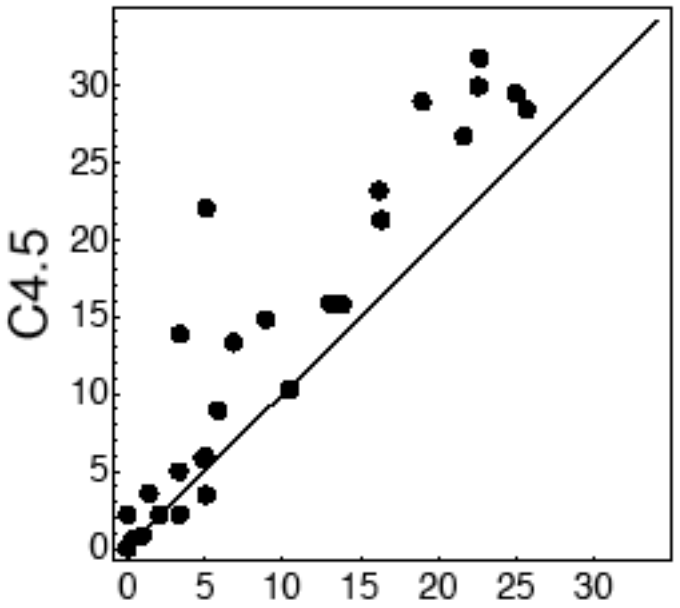
UCI Experiments

- tested AdaBoost on UCI benchmarks
- used: (Schapire and Freund)
  - C4.5 (Quinlan's decision tree algorithm)
  - "decision stumps": very simple rules of thumb that test on single attributes



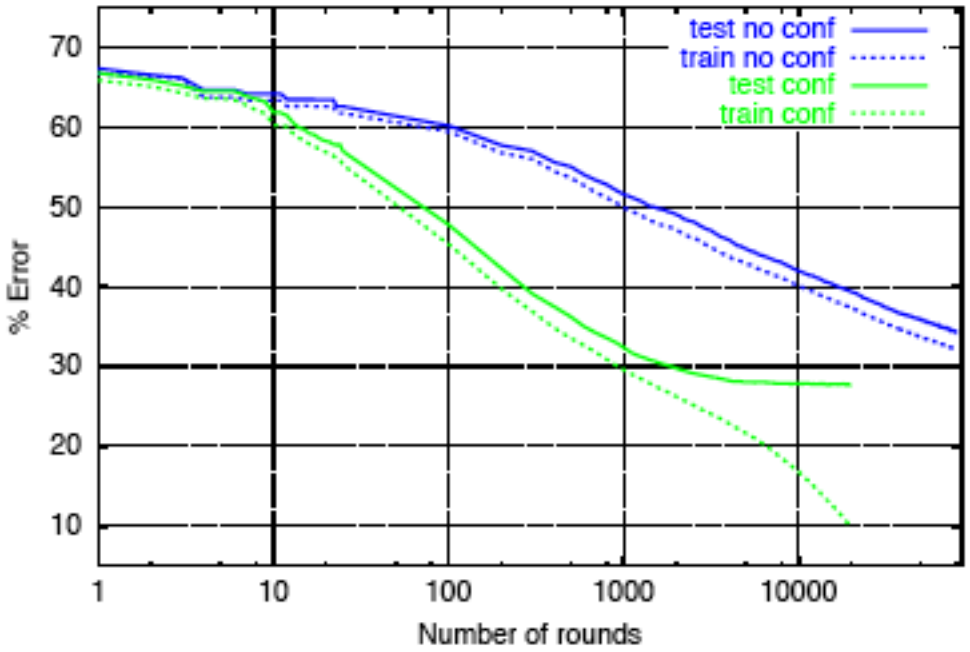
UCI Results





- error, error plots

Can Help A Lot

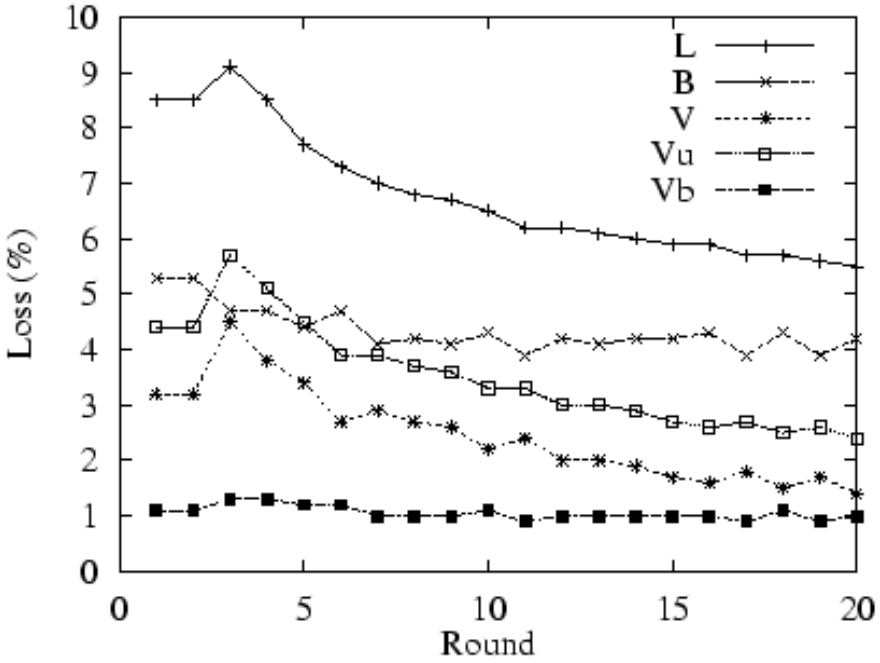


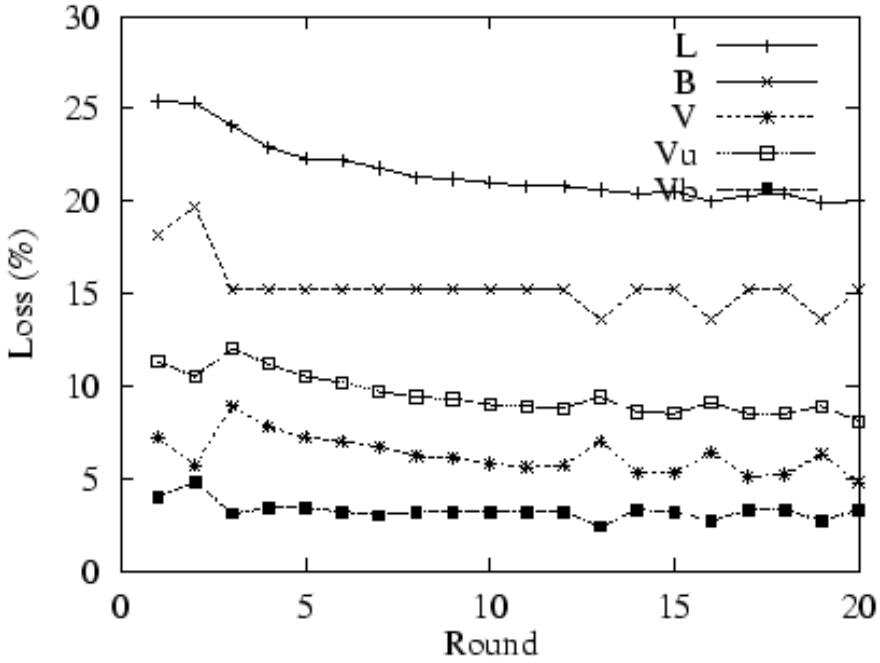
	round first reached		
% error	conf.	no conf.	speedup
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	–

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

Boosting DNA splice (left) and Audiology (right)





Early iterations reduce bias. Later iterations also reduce variance

Review and Conclusions

- For regression problems (squared error loss), the expected error rate can be decomposed into
  - $\text{Bias}(x^*)^2 + \text{Variance}(x^*) + \text{Noise}(x^*)$
- For classification problems (0/1 loss), the expected error rate depends on whether bias is present:
  - if $B(x^*) = 1$: $B(x^*) - [V(x^*) + N(x^*) - 2\, V(x^*)\, N(x^*)]$
  - if $B(x^*) = 0$: $B(x^*) + [V(x^*) + N(x^*) - 2\, V(x^*)\, N(x^*)]$
  - or $B(x^*) + Vu(x^*) - Vb(x^*)$ [ignoring noise]

Review and Conclusions (2)

- For classification problems with log loss, the expected loss can be decomposed into noise + bias + variance

  $E[\ KL(y, h)\ ] = H(p) + KL(p, \underline{h}) + E_S[\ KL(\underline{h}, h)\ ]$

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 (but may increase Vu)
  - For high-variance classifiers, it can reduce variance

Effect of Boosting

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