Bias-Variance Theory

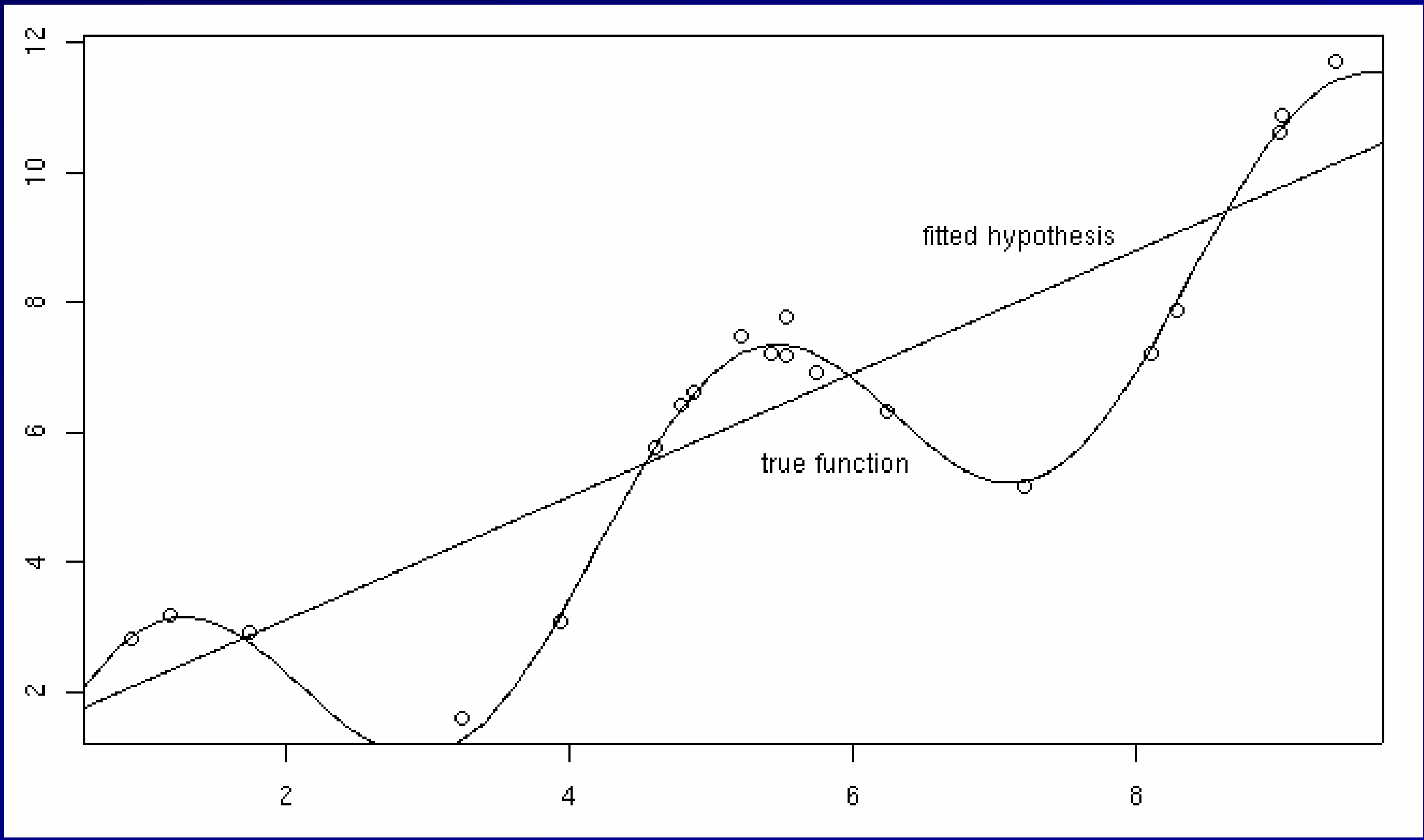
- Decompose Error Rate into components, some of which can be measured on unlabeled data
- Bias-Variance Decomposition for Regression
- Bias-Variance Decomposition for Classification
- Bias-Variance Analysis of Learning Algorithms
- Effect of Bagging on Bias and Variance
- Effect of Boosting on Bias and Variance
- Summary and Conclusion

Bias-Variance Analysis in Regression

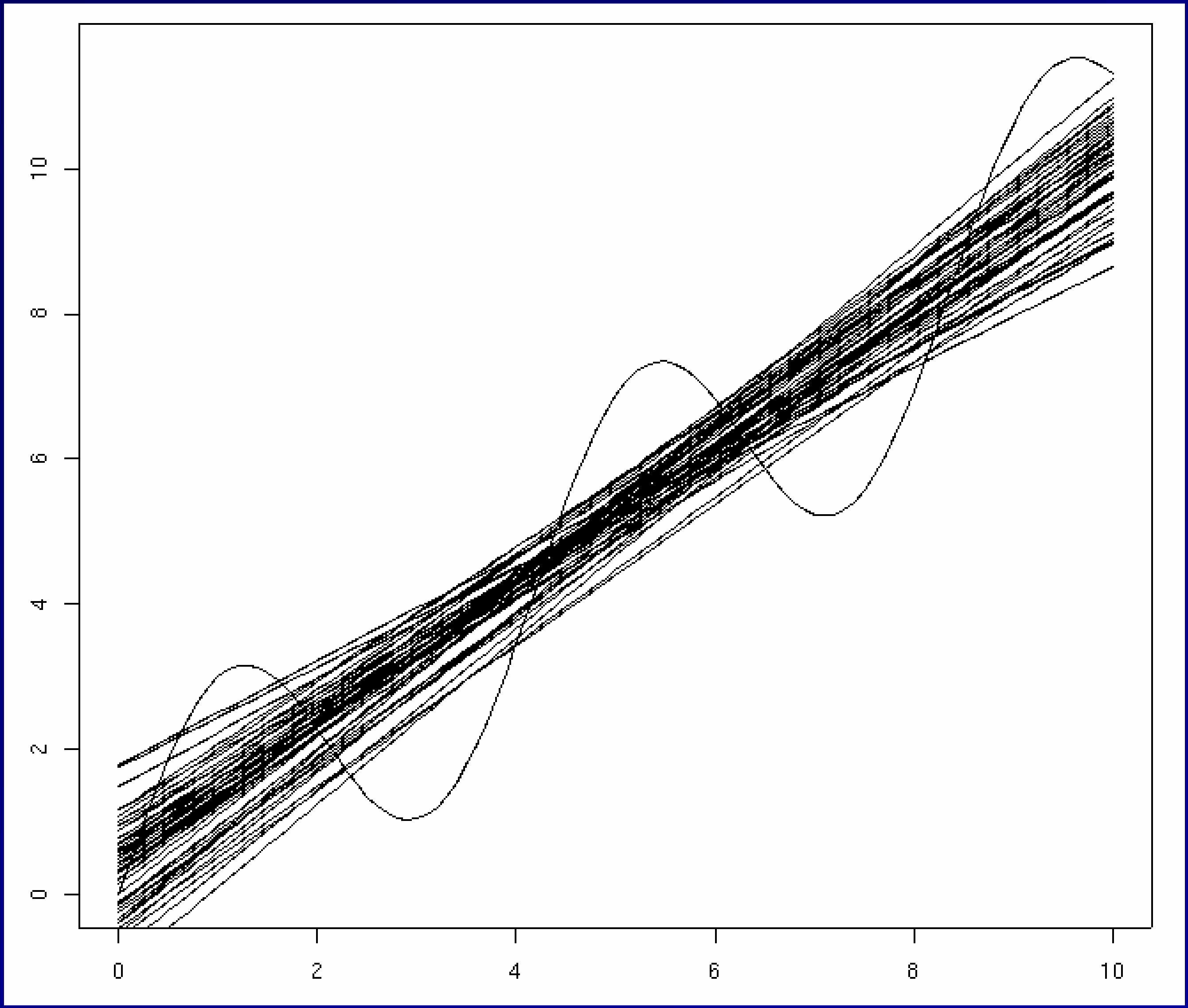
- True function is $y = f(x) + \varepsilon$
  - where ε is normally distributed with zero mean and standard deviation σ.
- Given a set of training examples, $\{(x_i, y_i)\}$, we fit an hypothesis $h(x) = w \cdot x + b$ to the data to minimize the squared error

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

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

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

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

Bias-Variance-Noise Decomposition

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

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

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

$\qquad - 2\,\underline{h(x^*)}\,f(x^*)$

$\qquad + E[\,(y^* - f(x^*))^2\,] + f(x^*)^2$ (lemma)

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

$\qquad (\underline{h(x^*)} - f(x^*))^2 +$ [bias^2]

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

Derivation (continued)

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

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

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

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

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

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

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

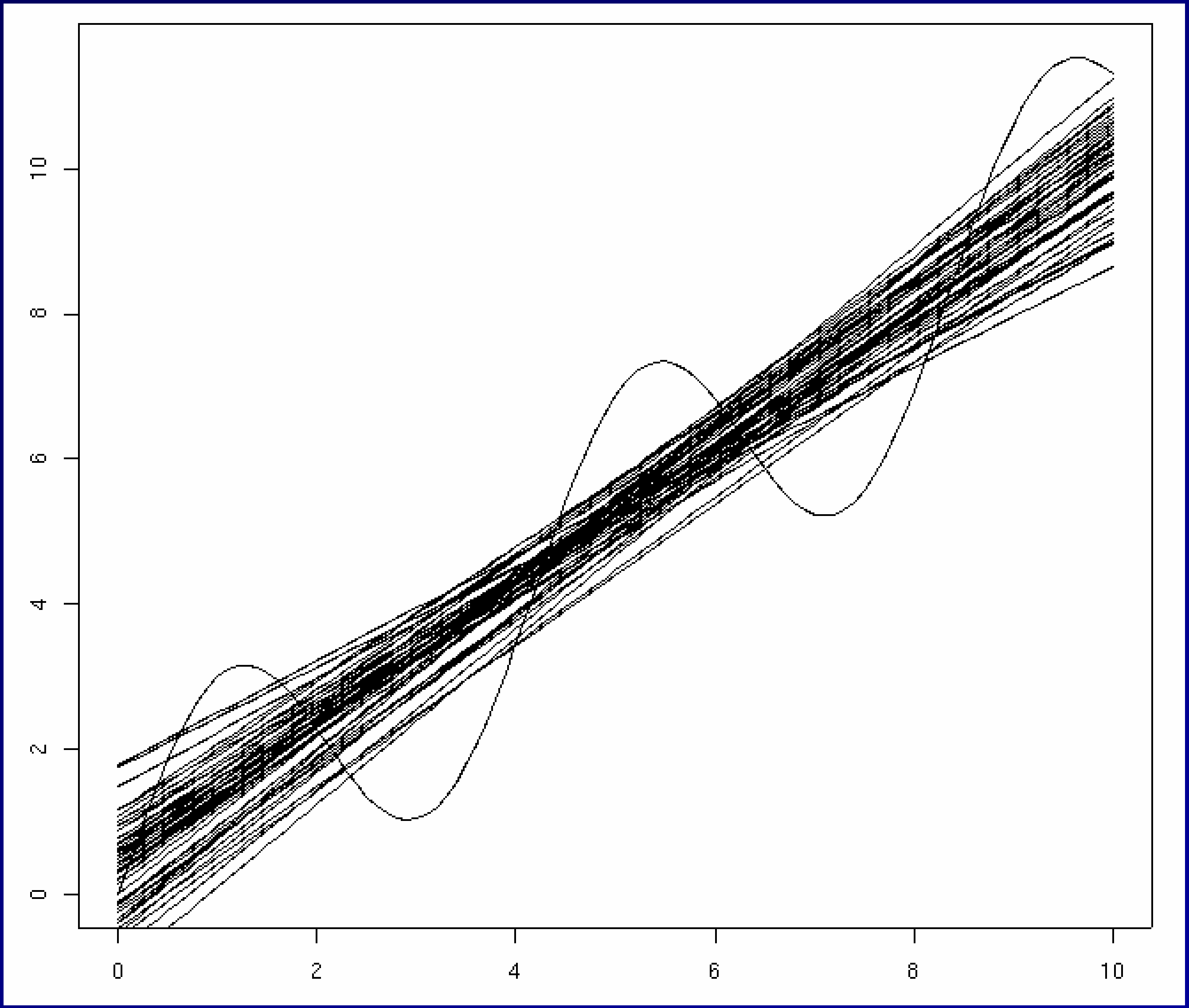
- Bias: $[\underline{h(x^*)} - f(x^*)]$

  Describes the average error of h(x*).

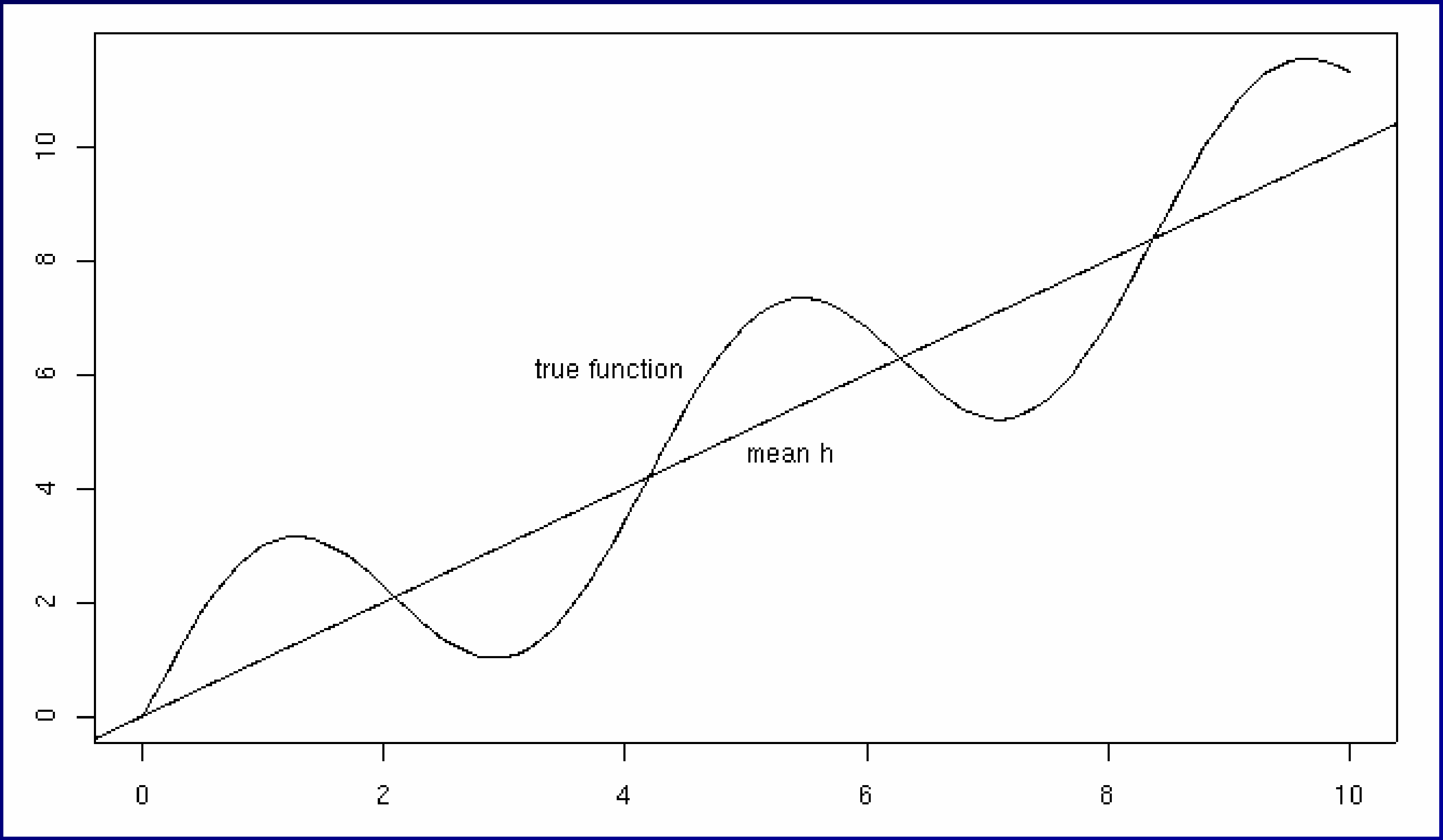
- Noise: $E[\, (y^* - f(x^*))^2 \,] = E[\varepsilon^2] = \sigma^2$

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

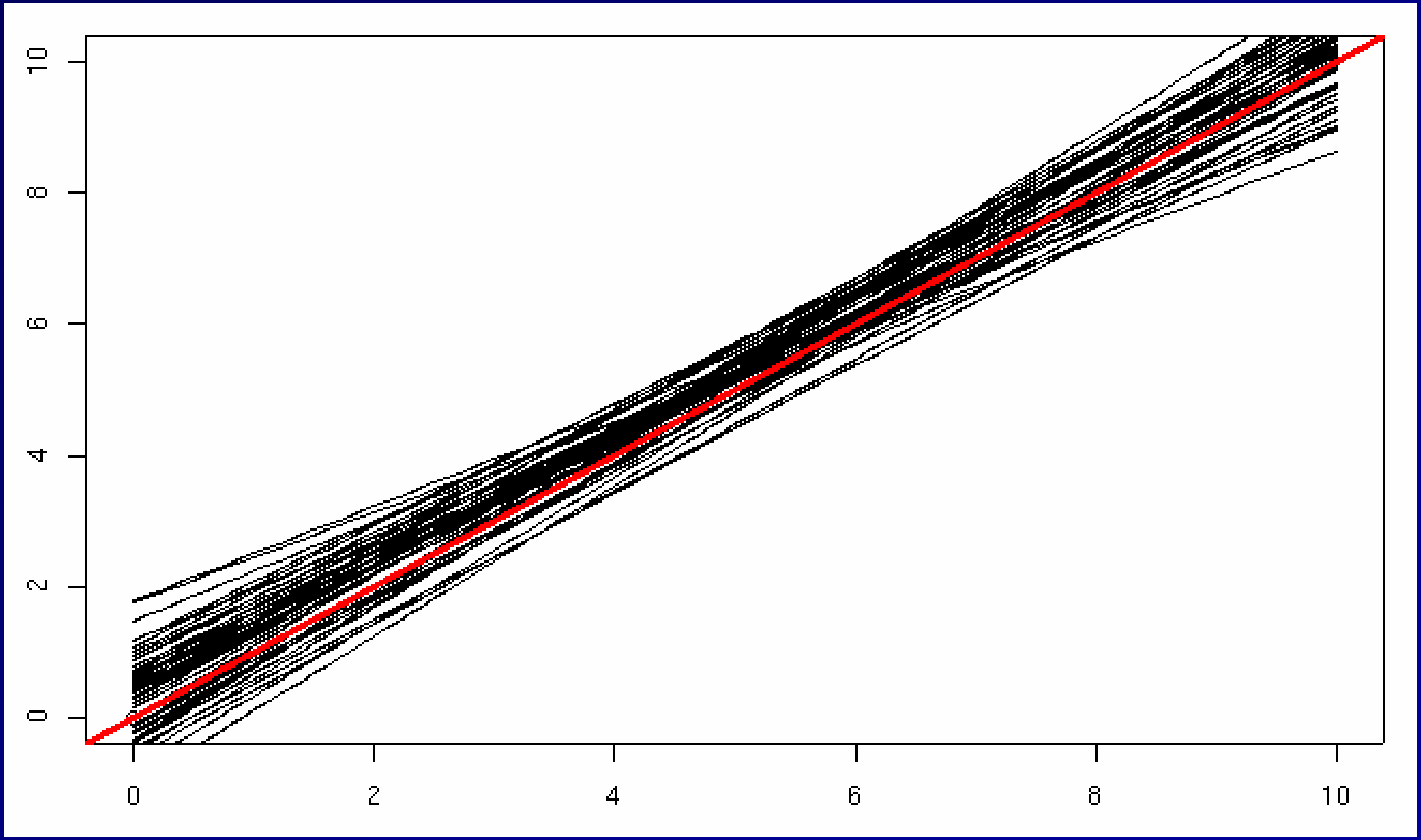
50 fits (20 examples each)



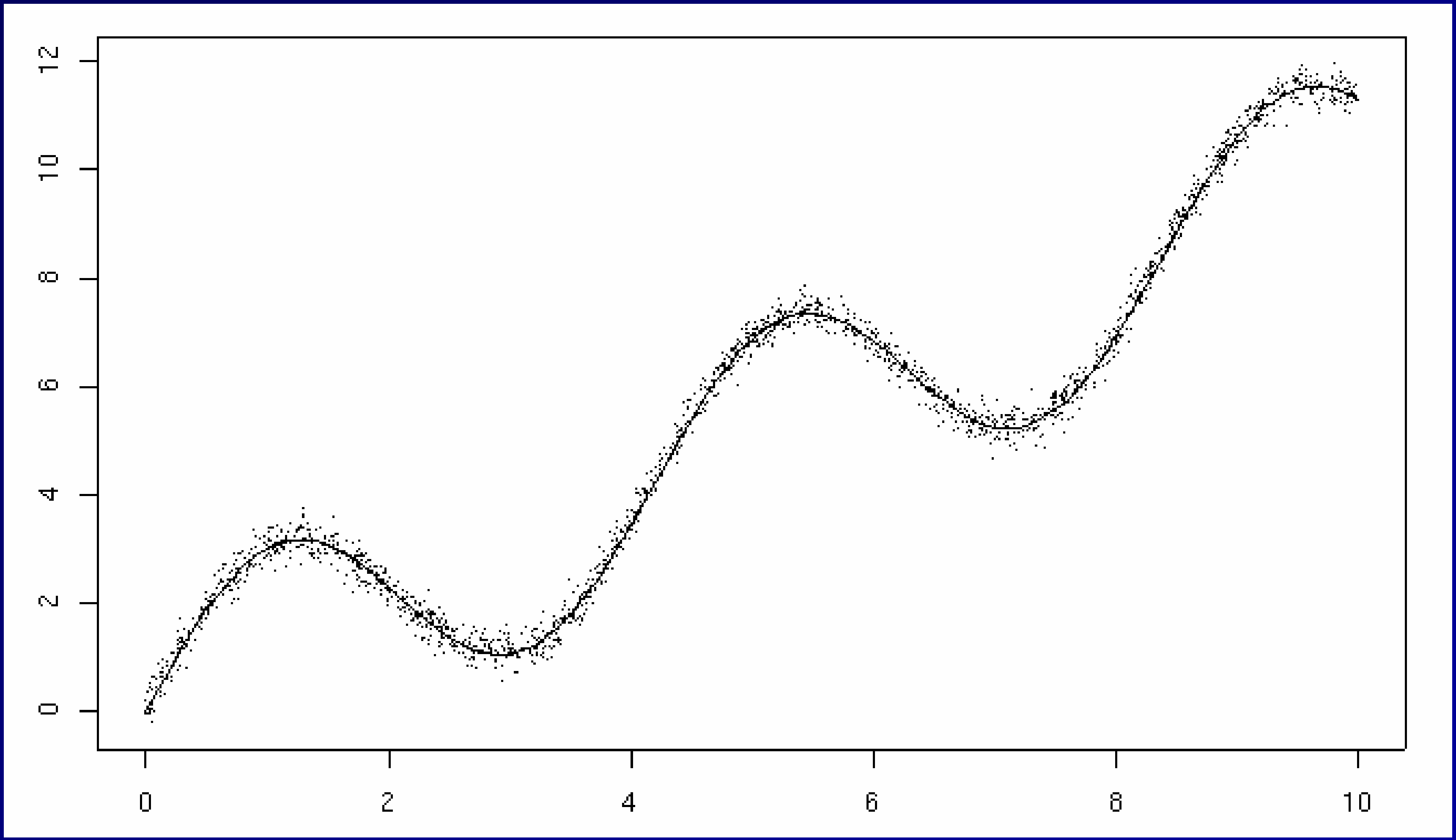
Bias



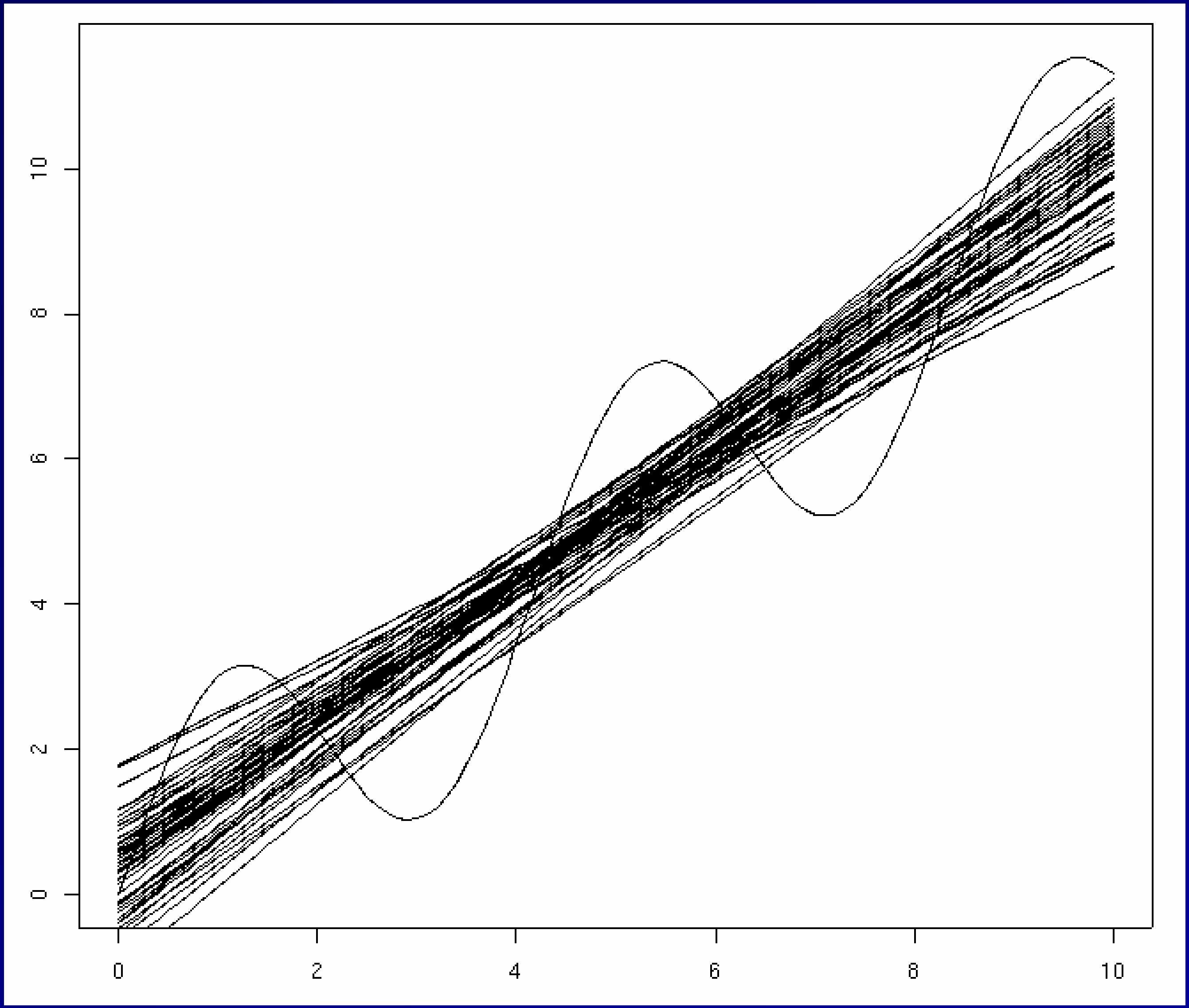
Variance



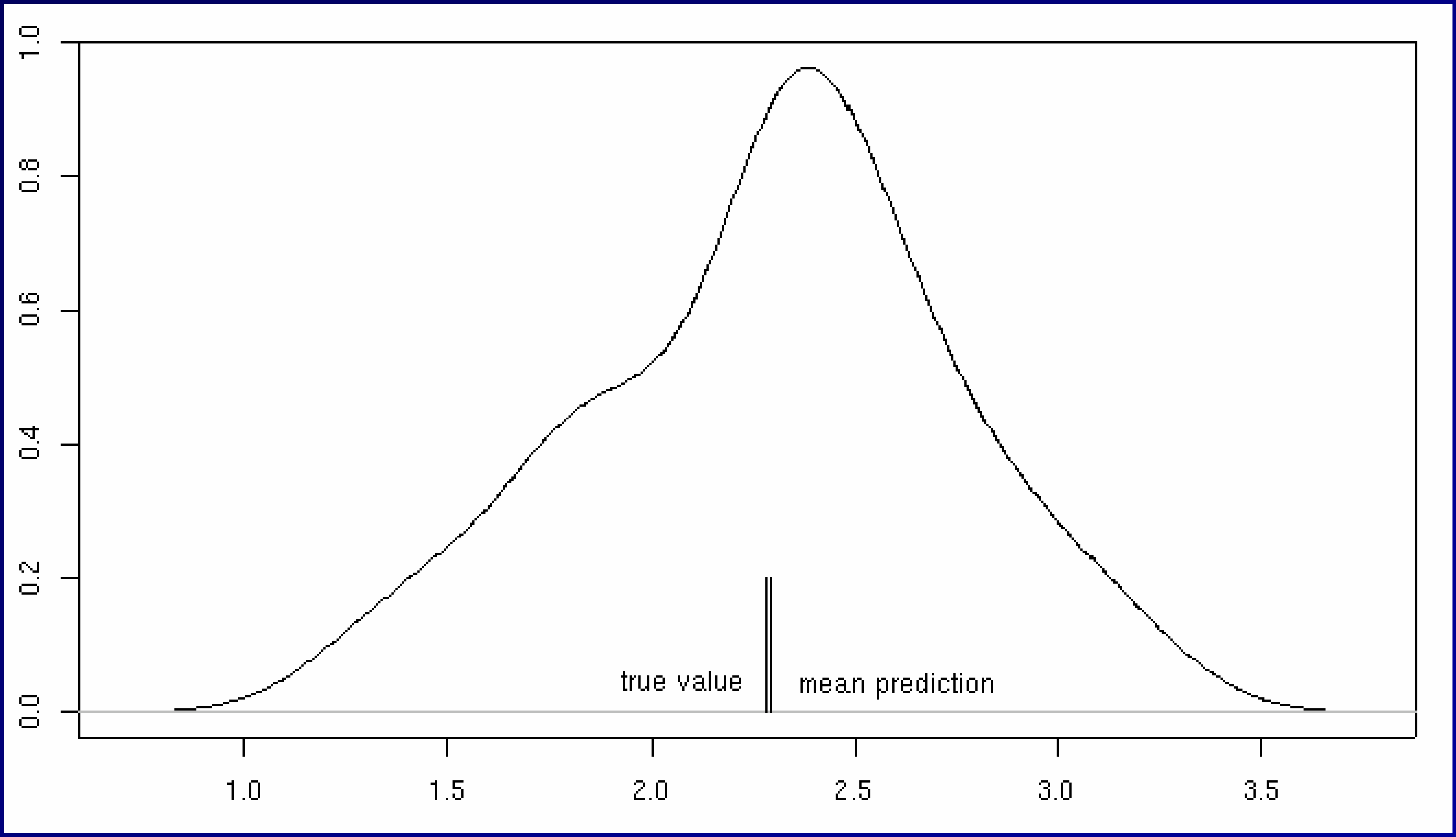
Noise



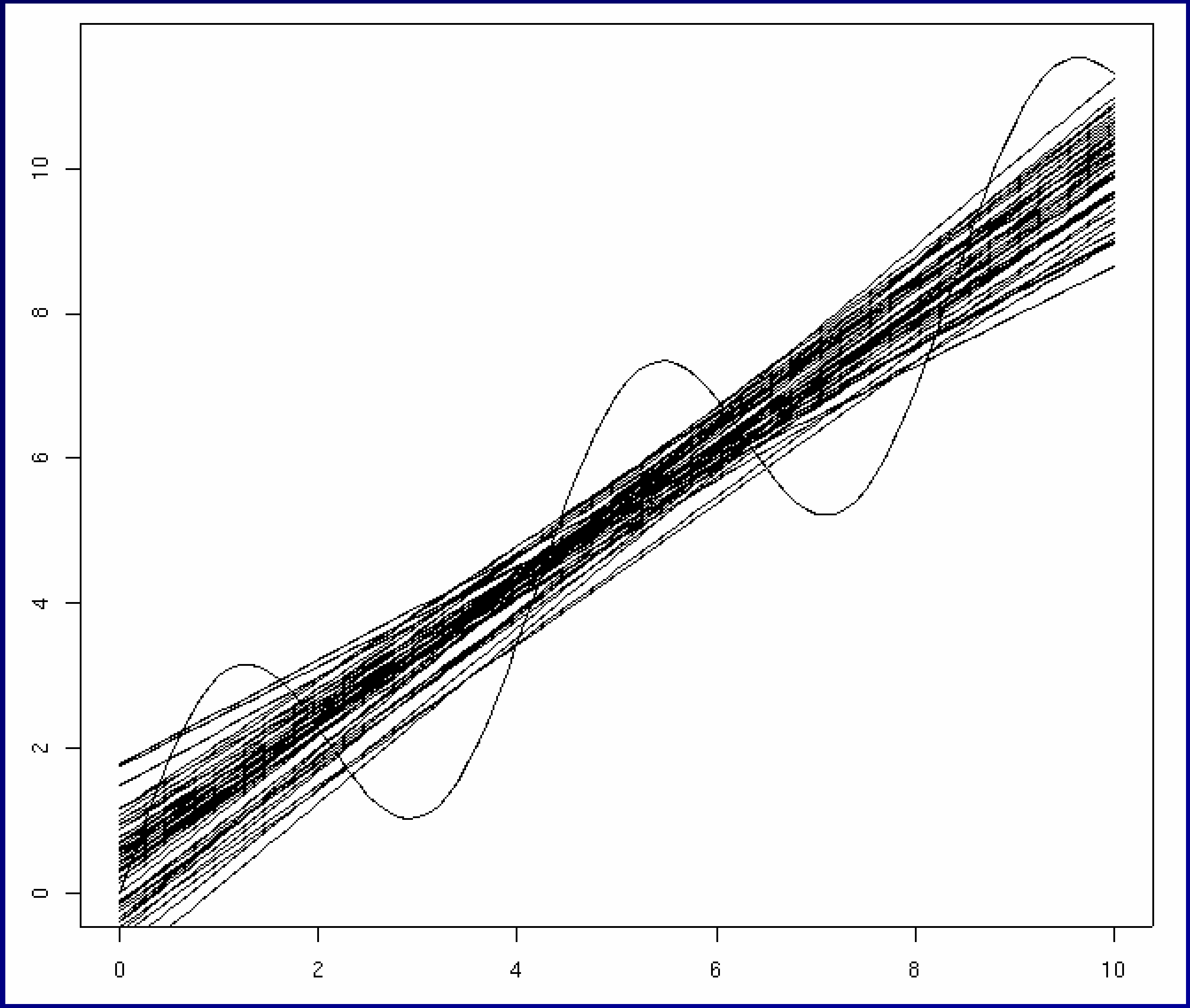
50 fits (20 examples each)



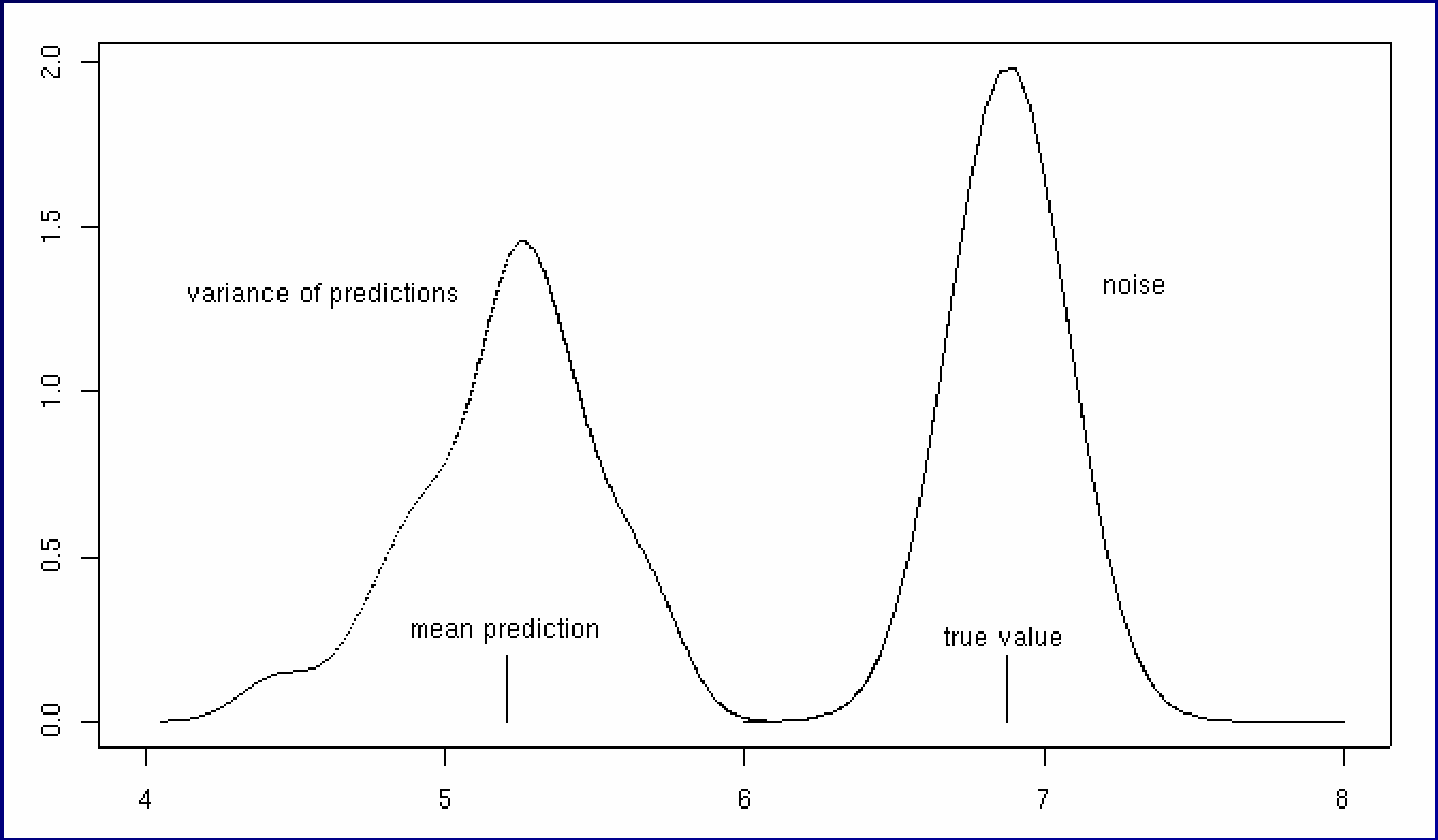
Distribution of predictions at x=2.0



50 fits (20 examples each)



Distribution of predictions at x=5.0



Measuring Bias and Variance

- In practice (unlike in theory), we have only ONE training set S.
- We can simulate multiple training sets by <u>bootstrap replicates</u>
  - 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): $S_1, \ldots, S_B$
- 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 $h_b(x)$ 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

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:
  - $[\sum_b h_b(x)]/B > 0$

Bagging

- Bagging makes predictions according to

  $y = \Sigma_b h_b(x) / B$

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

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 $y_i = n(f(x_i))$, where
  - $f(x_i)$ is the true class label of x_i
  - $n(\cdot)$ is a noise process that may change the true label $f(x_i)$.
- Given a training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, our learning algorithm produces an hypothesis h.
- Let $y^* = n(f(x^*))$ be the observed label of a new data point x^*. $h(x^*)$ is the predicted label. The error ("loss") is defined as $L(h(x^*), y^*)$

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 $E_p[L(h(x^*), y^*)]$ 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

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

- For 0/1 loss, the main prediction is the most common vote of h(x*) (taken over all training sets S weighted according to P(S))
- For squared error, the main prediction is <u>h(x*)</u>

Bias, Variance, Noise

- Bias $B(x^*) = L(y^m, f(x^*))$
  - This is the loss of the main prediction with respect to the true label of x*
- Variance $V(x^*) = E[\ L(h(x^*), y^m)\ ]$
  - This is the expected loss of h(x*) relative to the main prediction
- Noise $N(x^*) = E[\ L(y^*, f(x^*))\ ]$
  - This is the expected loss of the noisy observed value y* relative to the true label of 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 $y^m = \underline{h(x^*)}$
  - Bias^2: $L(\underline{h(x^*)}, f(x^*)) = (\underline{h(x^*)} - f(x^*))^2$
  - Variance:

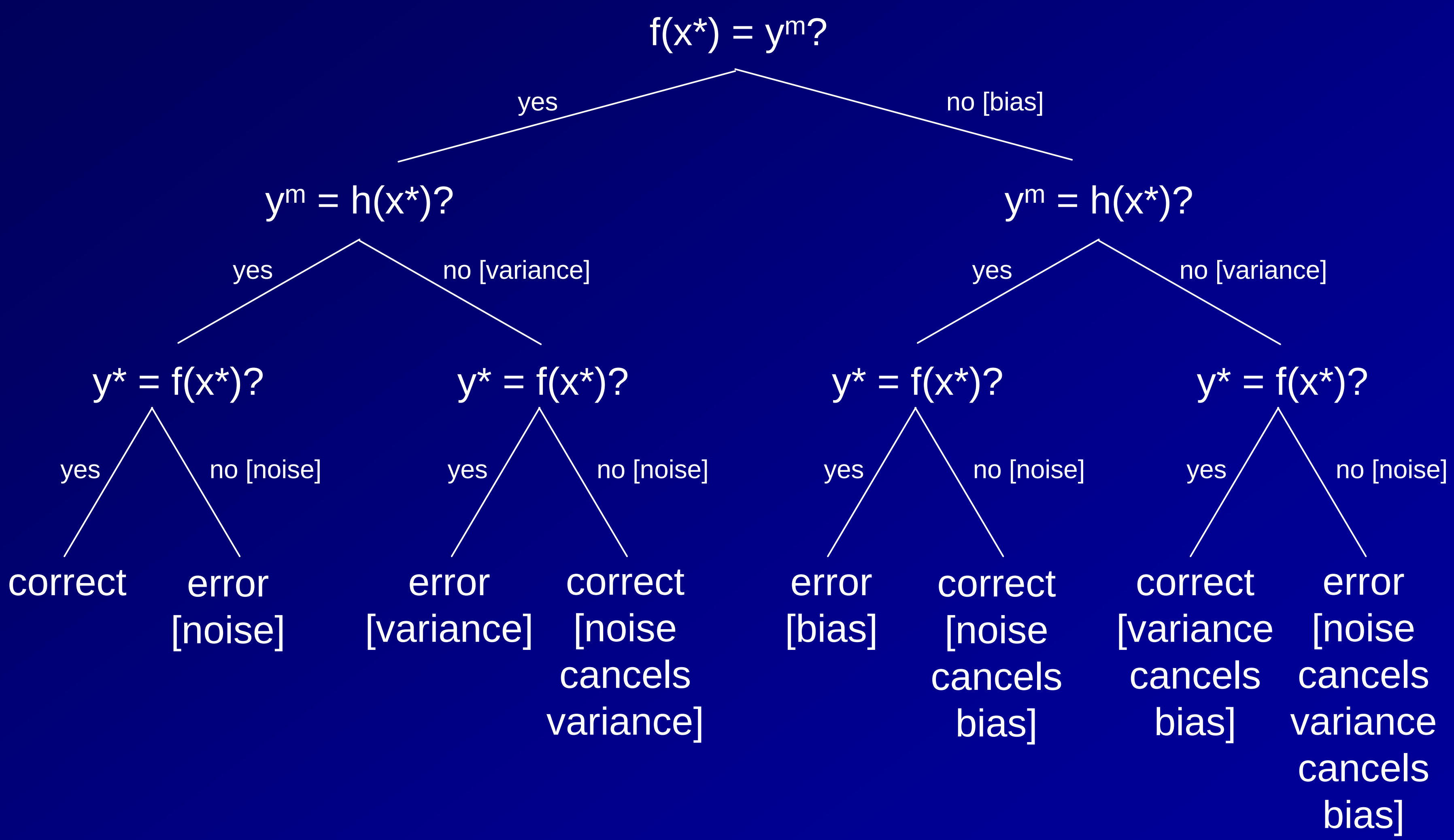
$$E[\ L(h(x^*), \underline{h(x^*)})\ ] = E[\ (h(x^*) - \underline{h(x^*)})^2\ ]$$

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

0/1 Loss for 2 classes

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

Case Analysis of Error



Unbiased case

- Let $P(y^* \neq f(x^*)) = N(x^*) = \tau$
- Let $P(y^m \neq h(x^*)) = V(x^*) = \sigma$
- If $(f(x^*) = y^m)$, then we suffer a loss if exactly one of these events occurs:
  $L(h(x^*), y^*) = \tau(1-\sigma) + \sigma(1-\tau)$
  $= \tau + \sigma - 2\tau\sigma$
  $= N(x^*) + V(x^*) - 2\, N(x^*)\, V(x^*)$

Biased Case

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

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

Decomposition for 0/1 Loss (2 classes)

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

  E[L(h(x*), y*)] =

  if B(x*) = 1: B(x*) – [N(x*) + V(x*) – 2 N(x*) V(x*)]

  if B(x*) = 0: B(x*) + [N(x*) + V(x*) – 2 N(x*) V(x*)]

- In biased case, noise and variance <u>reduce</u> error; in unbiased case, noise and variance <u>increase</u> 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 $2\ V(x^*)\ N(x^*)$ will be < 0.08

0/1 Decomposition in Practice

- In the noise-free case:

  E[L(h(x*), y*)] =

  if B(x*) = 1: B(x*) – V(x*)

  if B(x*) = 0: B(x*) + V(x*)

- It is usually hard to estimate N(x*), so we will use this formula

Decomposition over an entire data set

- Given a set of test points

  $T = \{(x^*_1,y^*_1),\ldots, (x^*_n,y^*_n)\}$,

  we want to decompose the average loss:

  $\underline{L} = \Sigma_i E[L(h(x^*_i), y^*_i)] / n$
- 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 ~ P(y|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|x) and h be our hypothesis for P(y=1|x).
- The KL divergence is

$$KL(p,h) = p \log p/h + (1-p) \log (1-p)/(1-h)$$

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:

  $\log(\underline{h}/(1-\underline{h})) = E_S[\, \log(h/(1-h)) \,]$

  $\underline{h} = 1/Z * \exp(\, E_S[\, \log h \,] \,)$

Computing the Noise

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

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

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

  $$\log(\underline{h}/(1-\underline{h})) = E_S[\ \log(h/(1-h))\ ]$$

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

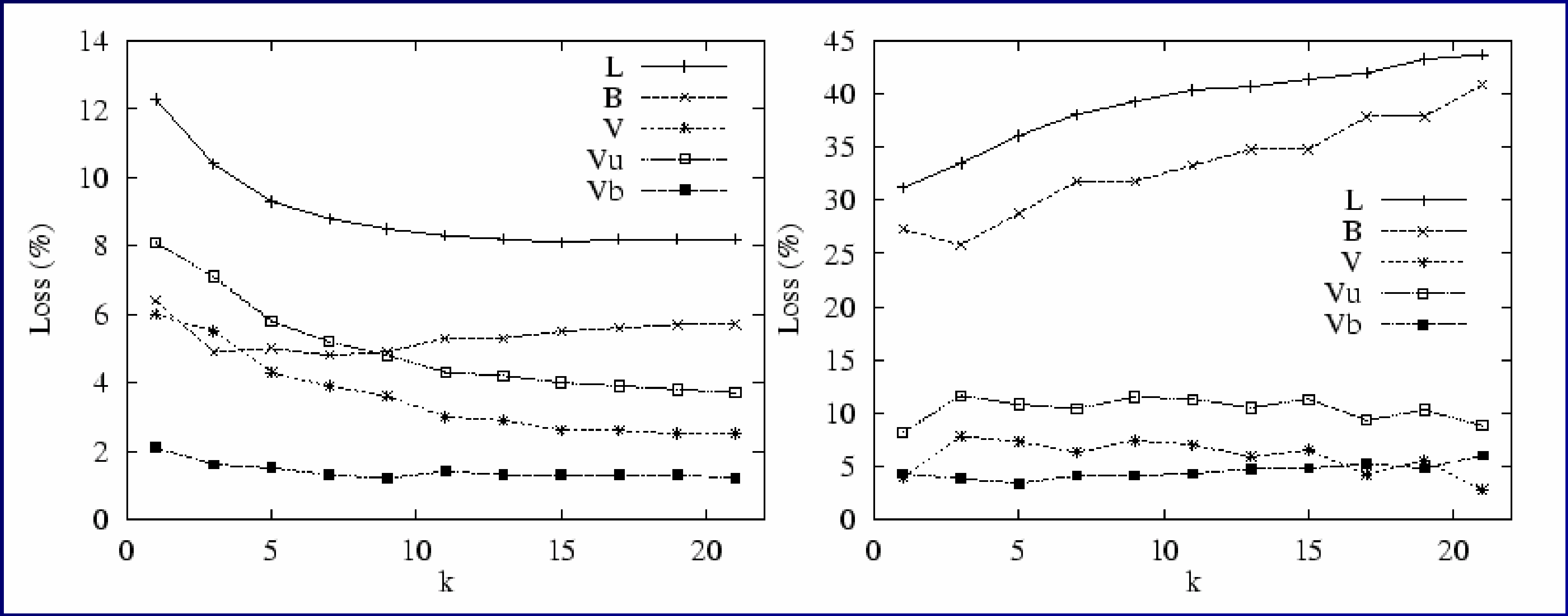
Experimental Studies of Bias and Variance

- 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

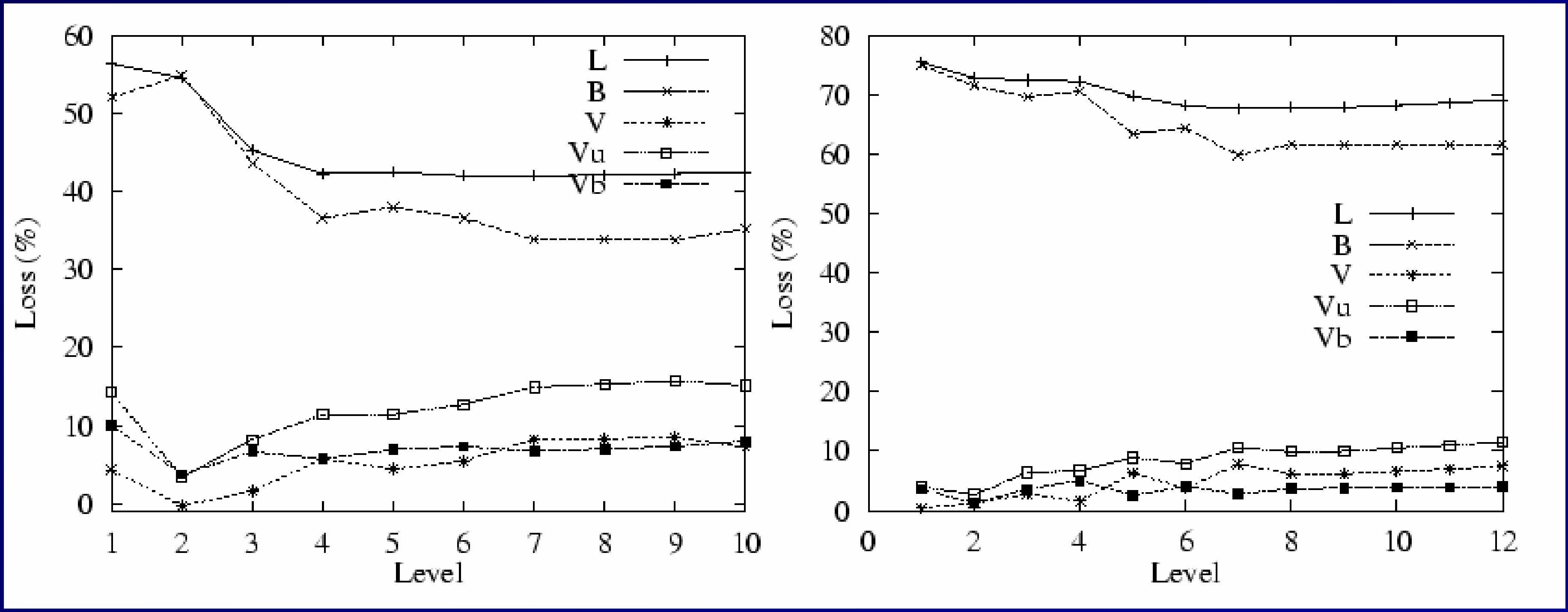
- K-nearest neighbors: What is the effect of K?
- Decision trees: What is the effect of pruning?
- Support Vector Machines: What is the effect of kernel width σ?

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

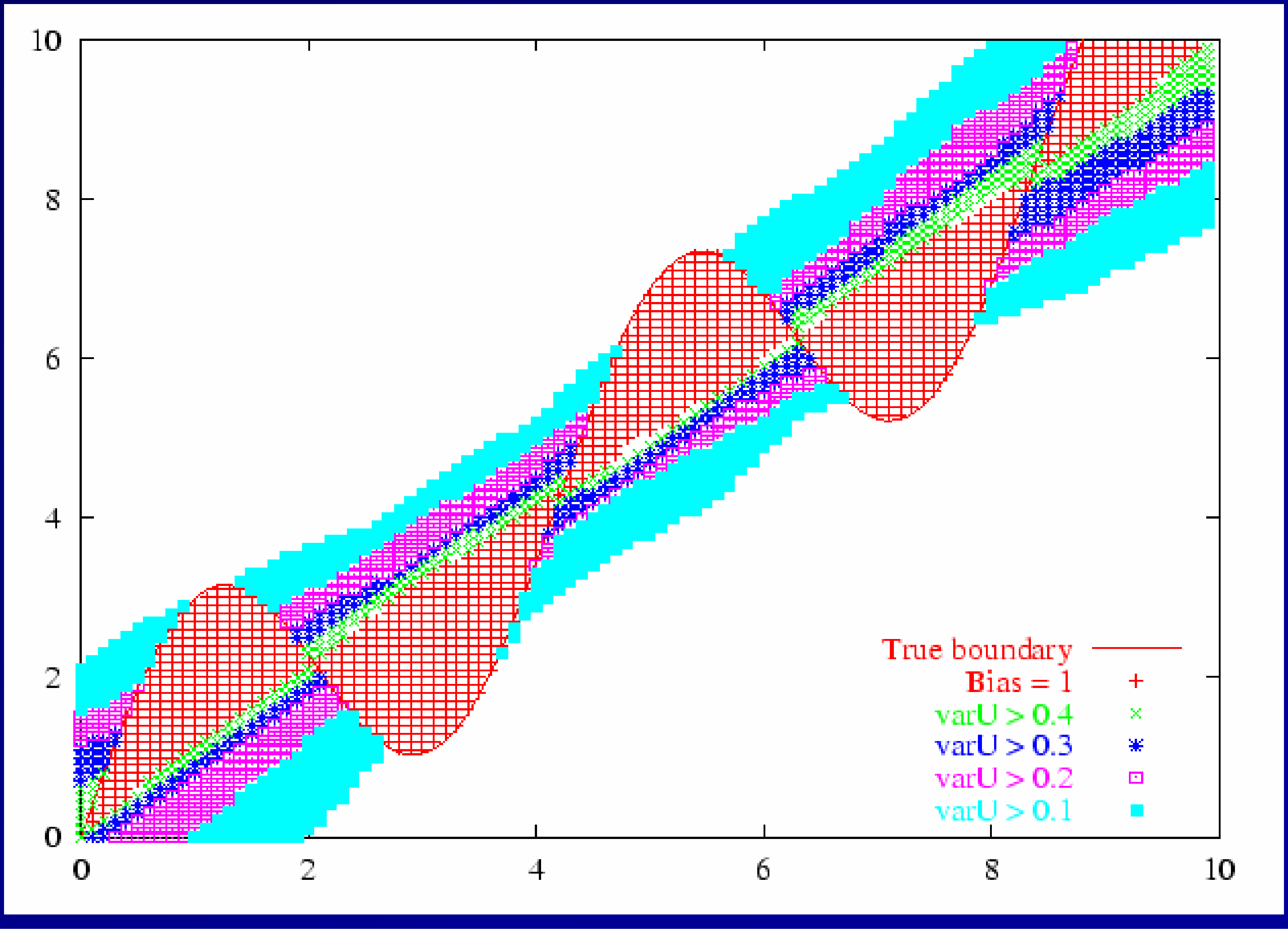
Example: 200 linear SVMs (training sets of size 20)

Error: 13.7%

Bias: 11.7%

Vu: 5.2%

Vb: 3.2%



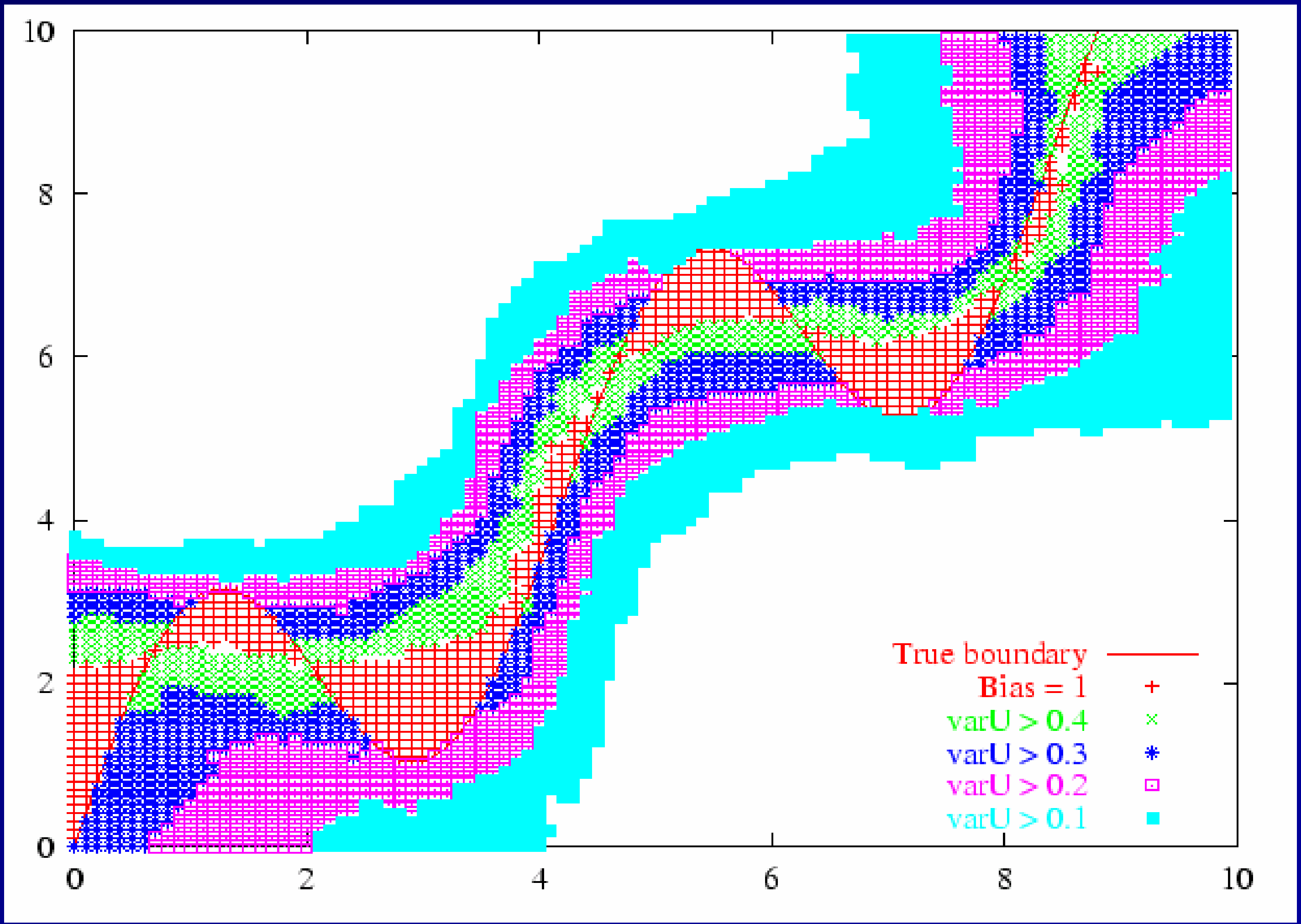
Example: 200 RBF SVMs
$\sigma = 5$

Error: 15.0%

Bias: 5.8%

Vu: 11.5%

Vb: 2.3%



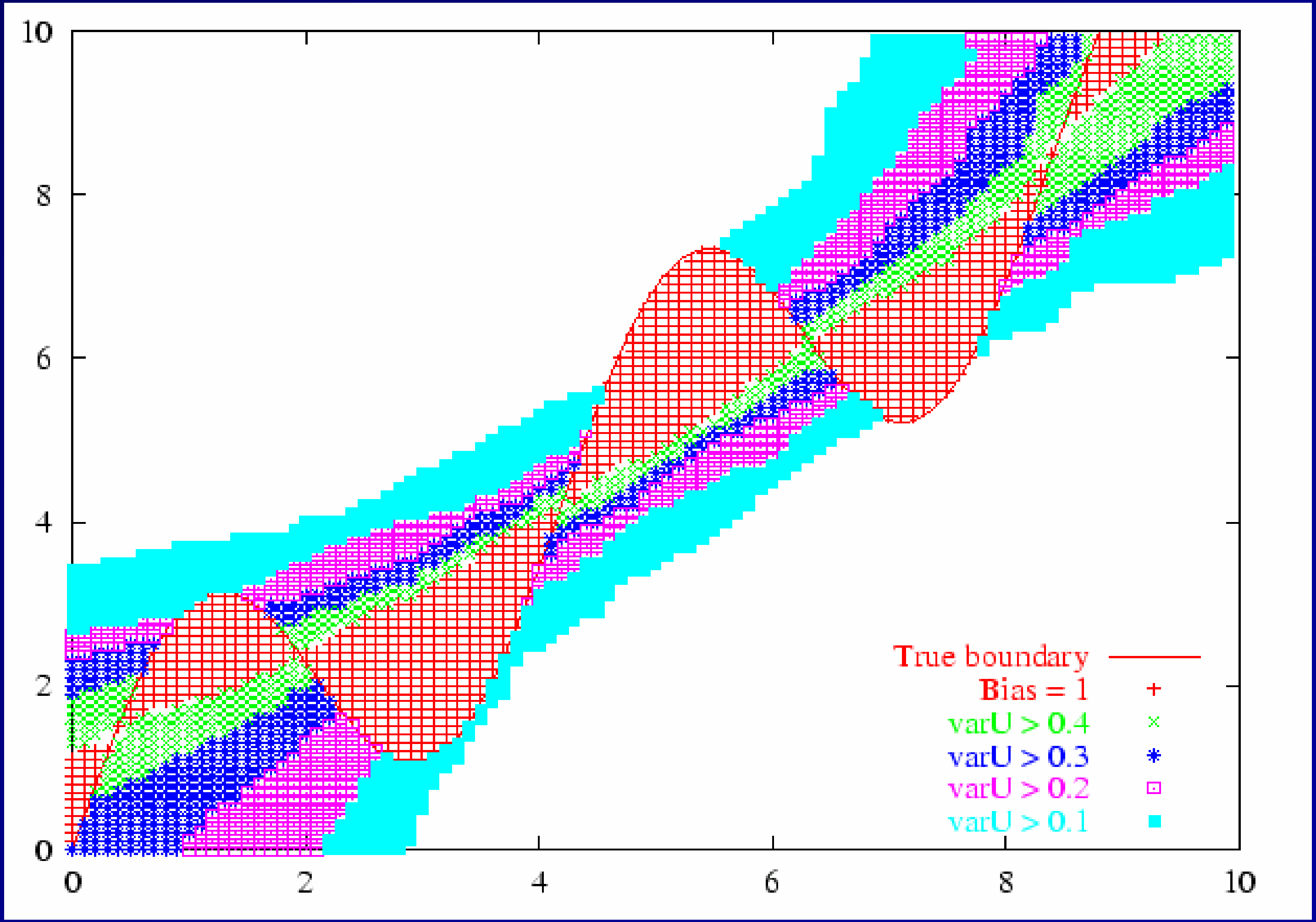
Example: 200 RBF SVMs
σ = 50

Error: 14.9%

Bias: 10.1%

Vu: 7.8%

Vb: 3.0%



SVM Bias and Variance

	Error	Bias	Var_U	Var_B	Net var	Tot var
linear	0.137	0.117	0.052	0.032	0.020	0.084
rbf $\sigma = 5$	0.150	0.058	0.115	0.023	0.092	0.137
rbf $\sigma = 50$	0.149	0.101	0.078	0.030	0.048	0.109

- Bias-Variance tradeoff controlled by σ
- Biased classifier (linear SVM) gives better results than a classifier that can represent the true decision boundary!

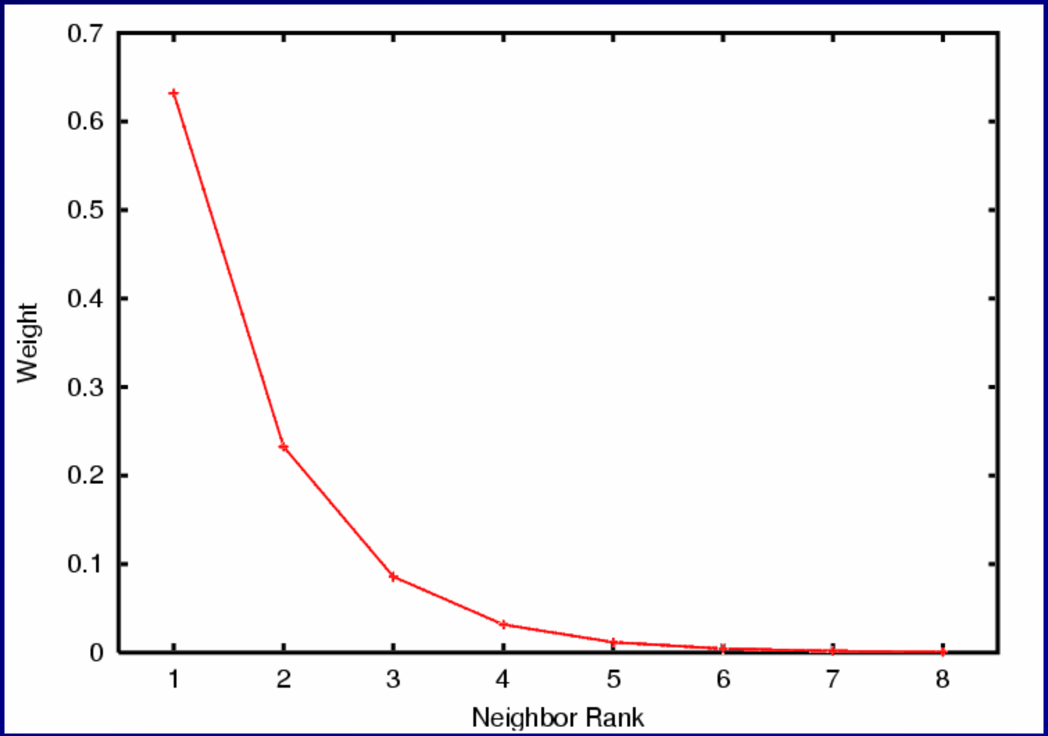
B/V Analysis of Bagging

- 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)

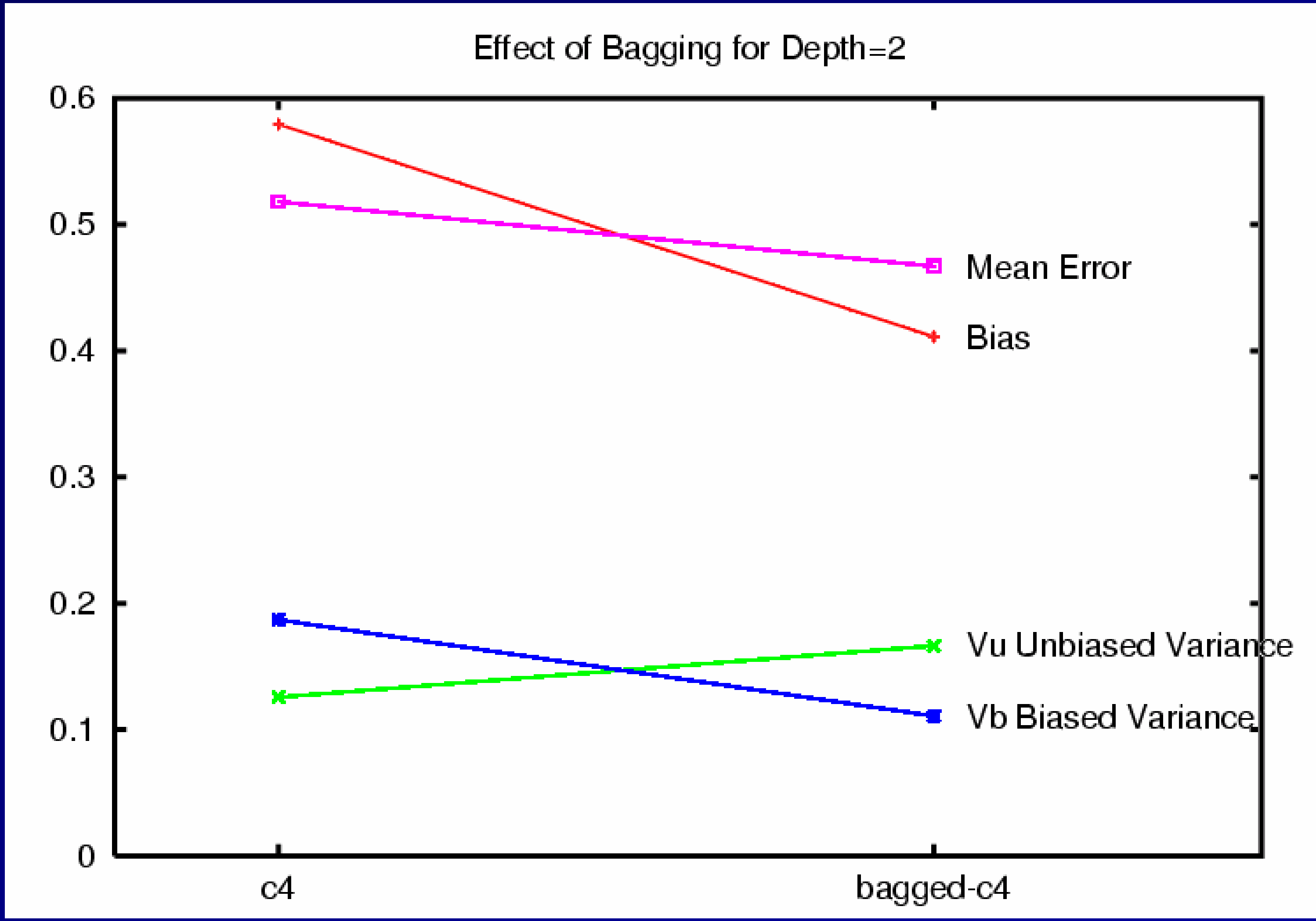


Since the first nearest neighbor gets more than half of the vote, it will always win this vote. Therefore, Bagging 1-NN is equivalent to 1-NN.

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!



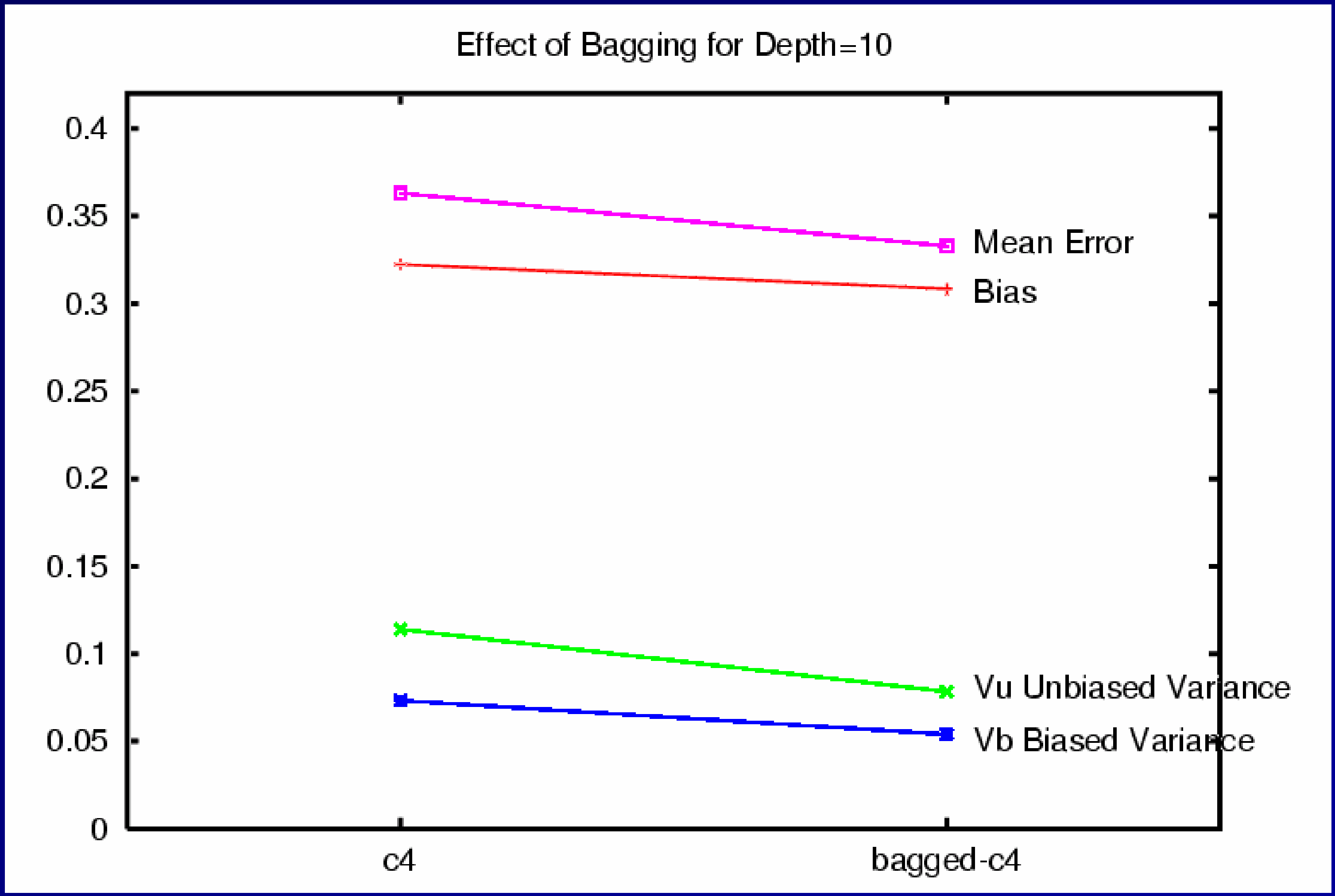
Questions

- 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?

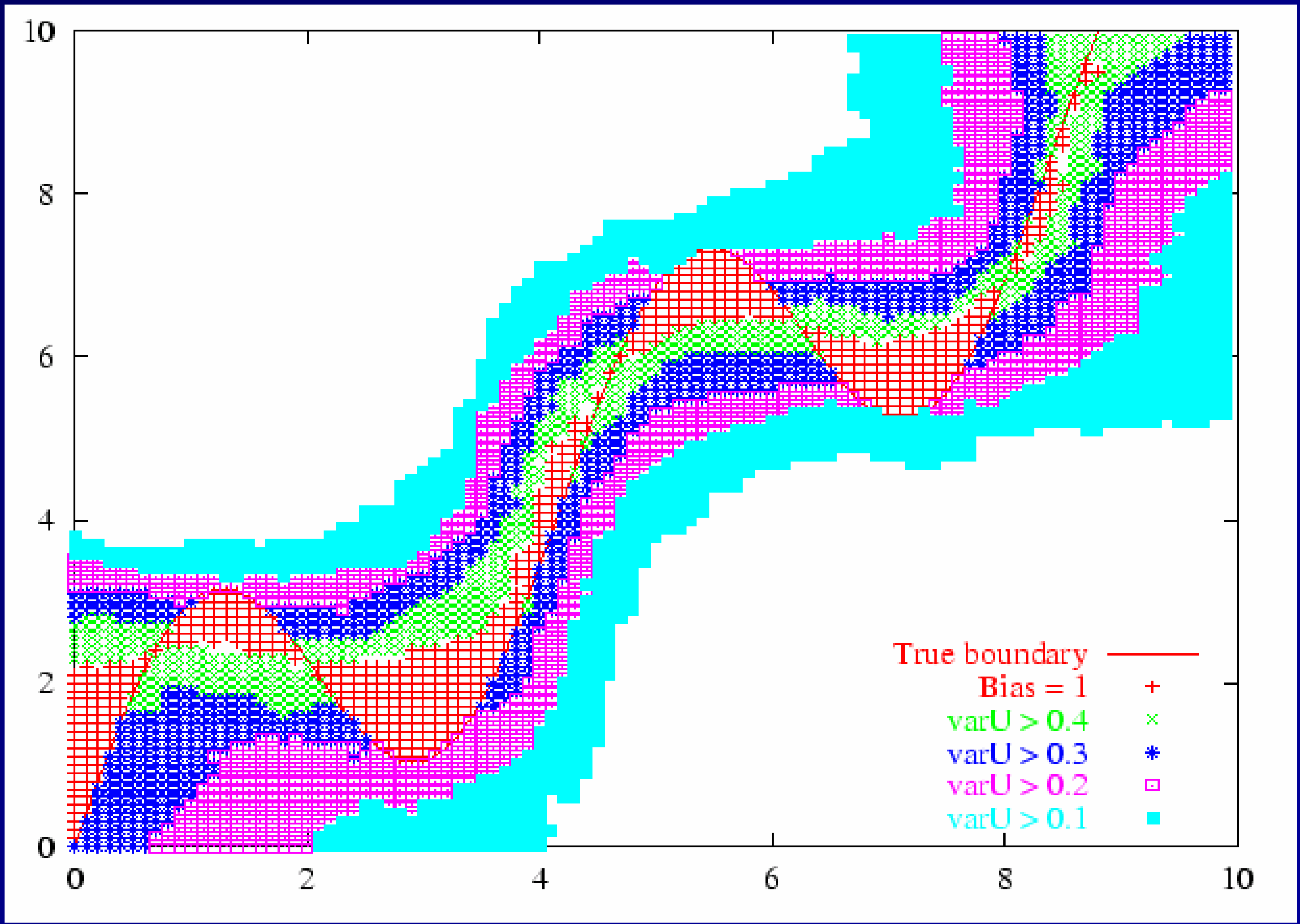
Answer: Bagging Primarily Reduces Variance



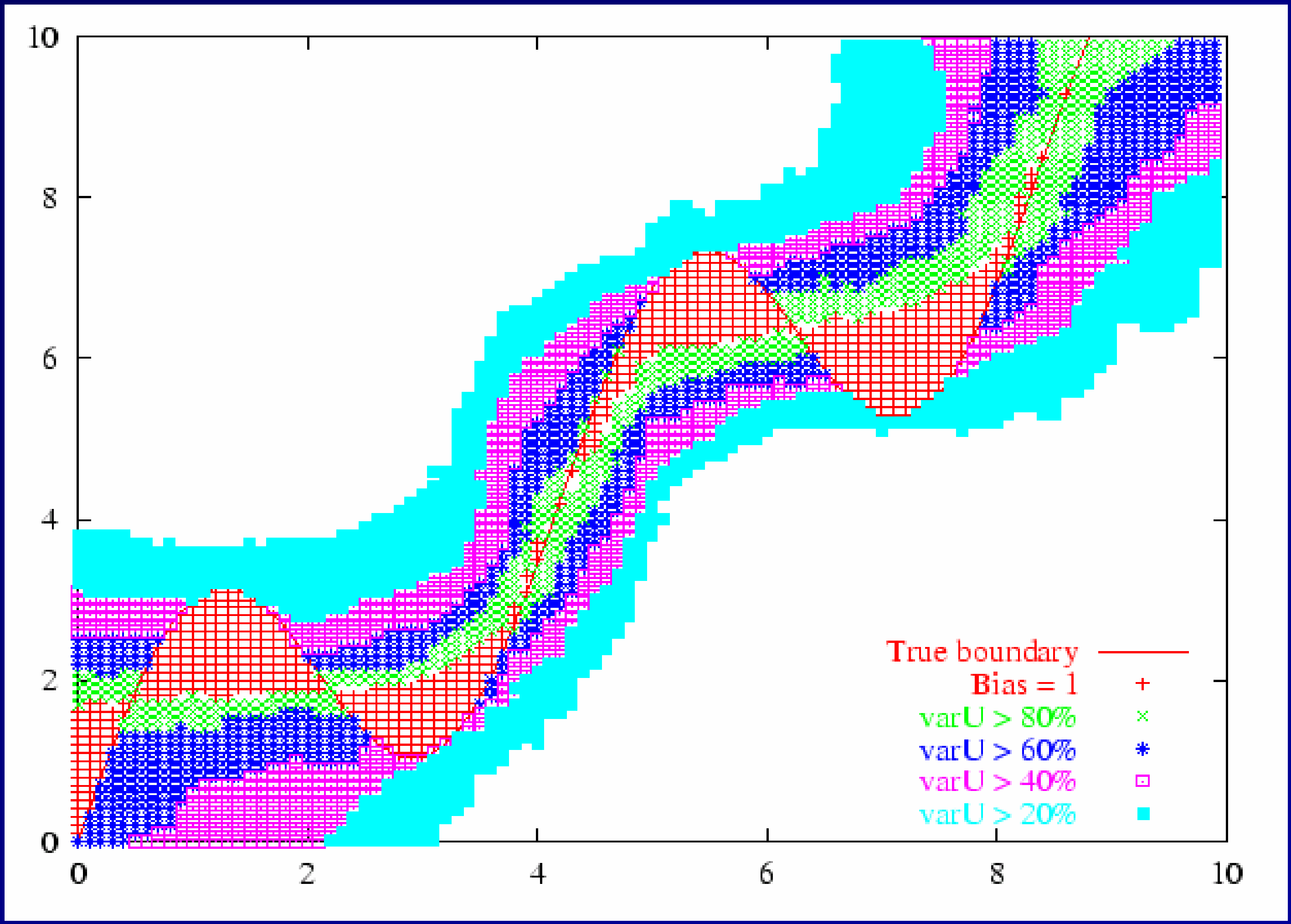
Bagging of SVMs

- We will choose a low-bias, high-variance SVM to bag: RBF SVM with σ=5

RBF SVMs again: $\sigma = 5$



Effect of 30-fold Bagging: Variance is Reduced

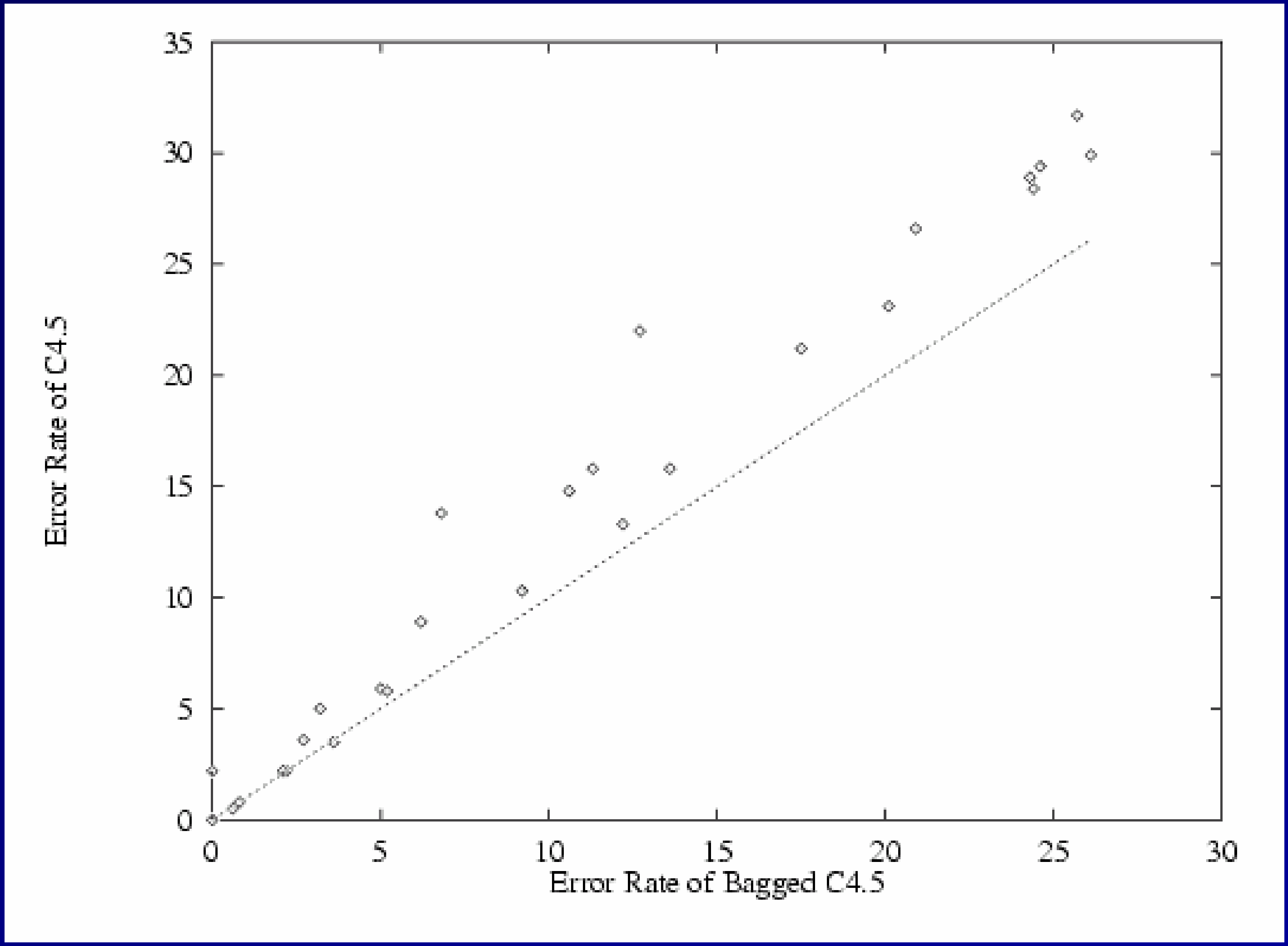


Effects of 30-fold Bagging

	Error	Bias	Var_U	Var_B	Net var	Tot var
rbf $\sigma = 5$	0.150	0.058	0.115	0.023	0.092	0.137
bagged rbf $\sigma = 5$	0.145	0.063	0.105	0.023	0.082	0.128

- Vu is decreased by 0.010; Vb is unchanged
- Bias is increased by 0.005
- Error is reduced by 0.005

Bagging Decision Trees (Freund & Schapire)



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.

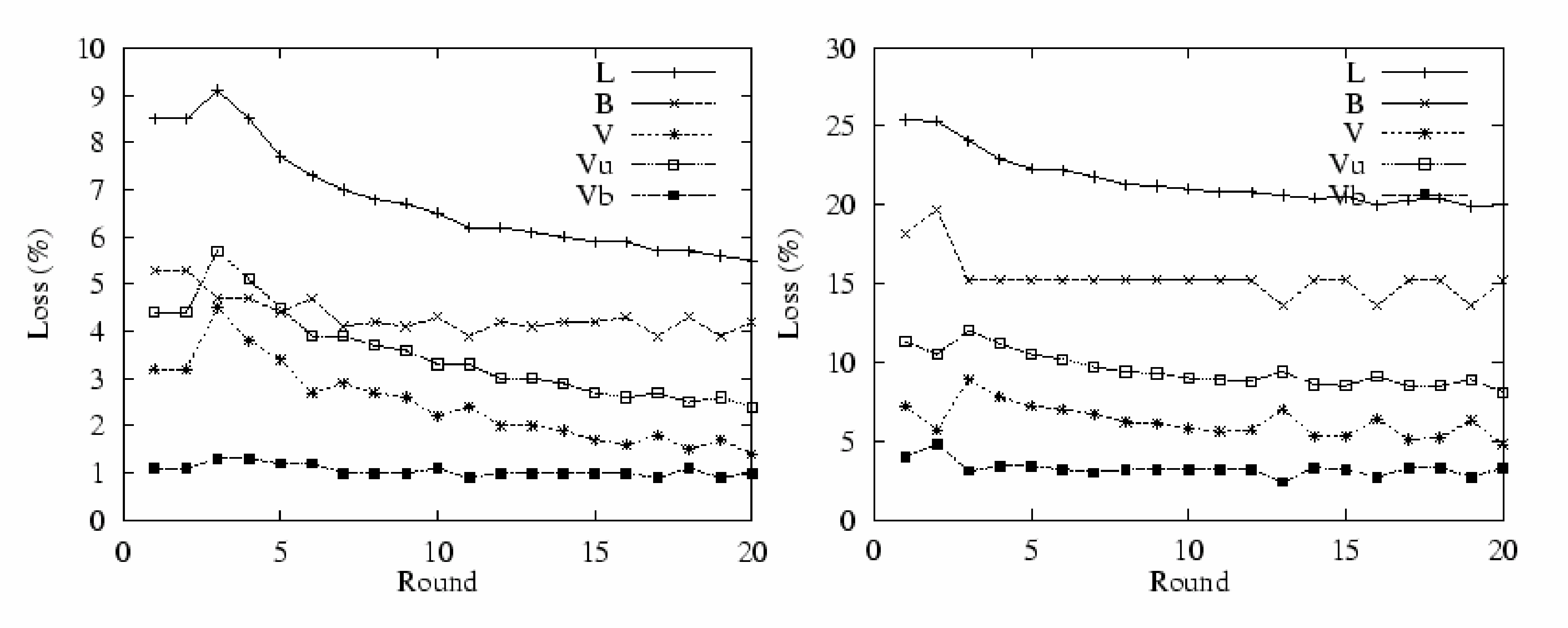
[1] **initialize for all** i**:** $w_1(i) := 1/m$	*initialize the weights*
[2] **for** $\ell = 1$ **to** L **do**	
[3] **for all** i: $p_\ell(i) := w_\ell(i)/(\Sigma_i w_\ell(i))$	*compute normalized weights*
[4] $h_\ell :=$ Learn(p_ℓ)	*call* Learn *with normalized weights.*
[5] $\epsilon_\ell := \Sigma_i p_\ell(i)[h_\ell(x_i) \neq y_i]$	*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: $h_f(x) = \operatorname*{argmax}_{y \in Y} \sum_{\ell=1}^{L} \left(\log \frac{1}{\beta_\ell}\right) [\![h_\ell(x) = y]\!]$

Bias-Variance Analysis of Boosting

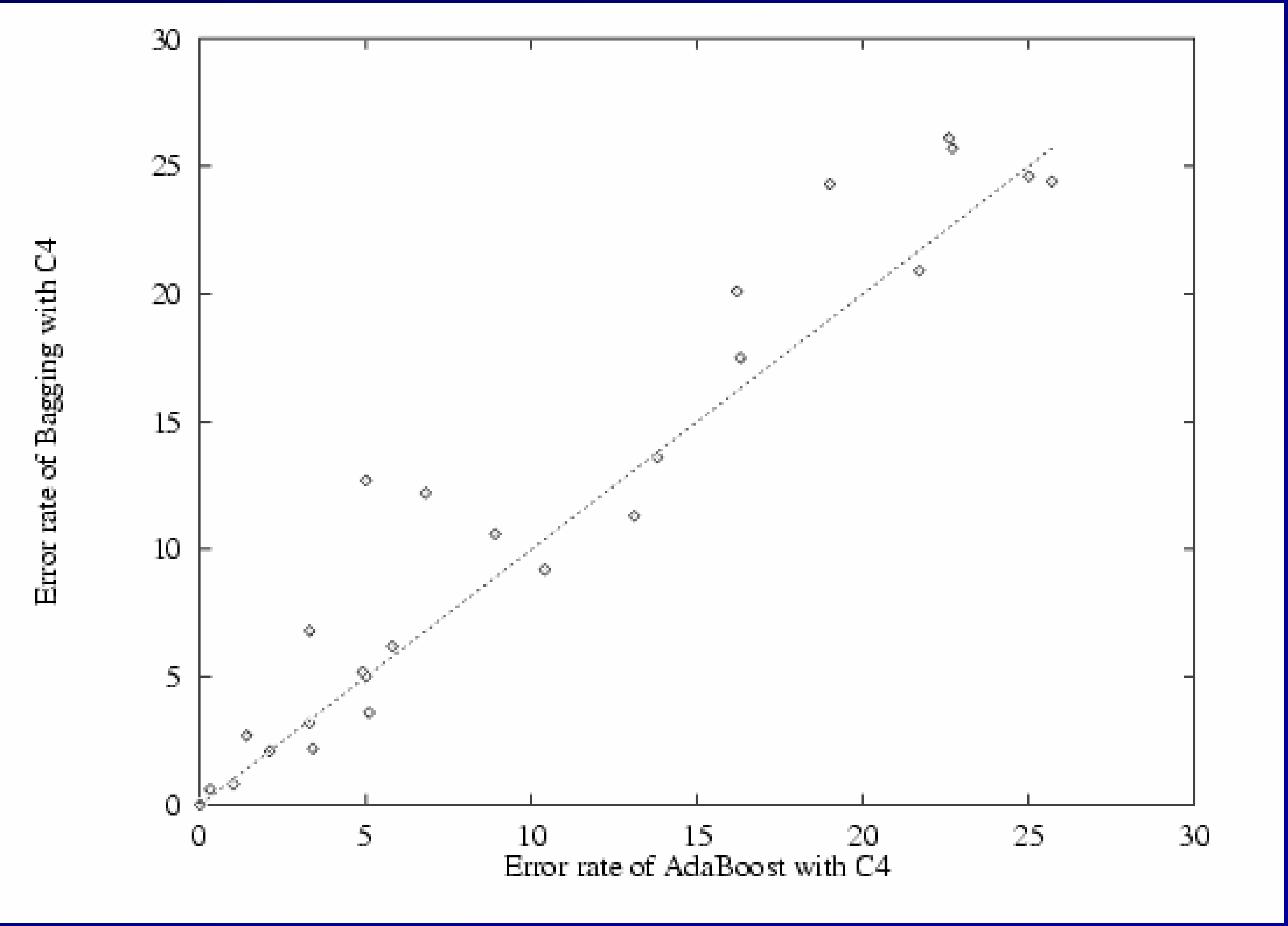
- Boosting seeks to find a weighted combination of classifiers that fits the data well
- Prediction: Boosting will primarily act to reduce bias

Boosting DNA splice (left) and Audiology (right)



Early iterations reduce bias. Later iterations also reduce variance

Boosting vs Bagging (Freund & Schapire)



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 Algorithm Parameters on Bias and Variance

- k-nearest neighbor: increasing k typically increases bias and reduces variance
- decision trees of depth D: increasing D typically increases variance and reduces bias
- RBF SVM with parameter σ: increasing σ increases bias and reduces 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