



# Variance Reduction and Ensemble Methods

Nicholas Ruozzi

University of Texas at Dallas

Based on the slides of Vibhav Gogate and David Sontag

- PAC learning
- Bias/variance tradeoff
  - small hypothesis spaces (not enough flexibility) can have high bias
  - rich hypothesis spaces (too much flexibility) can have high variance
- Today: more on this phenomenon and how to get around it

- Bias
  - Measures the accuracy or quality of the algorithm
  - High bias means a poor match
- Variance
  - Measures the precision or specificity of the match
  - 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

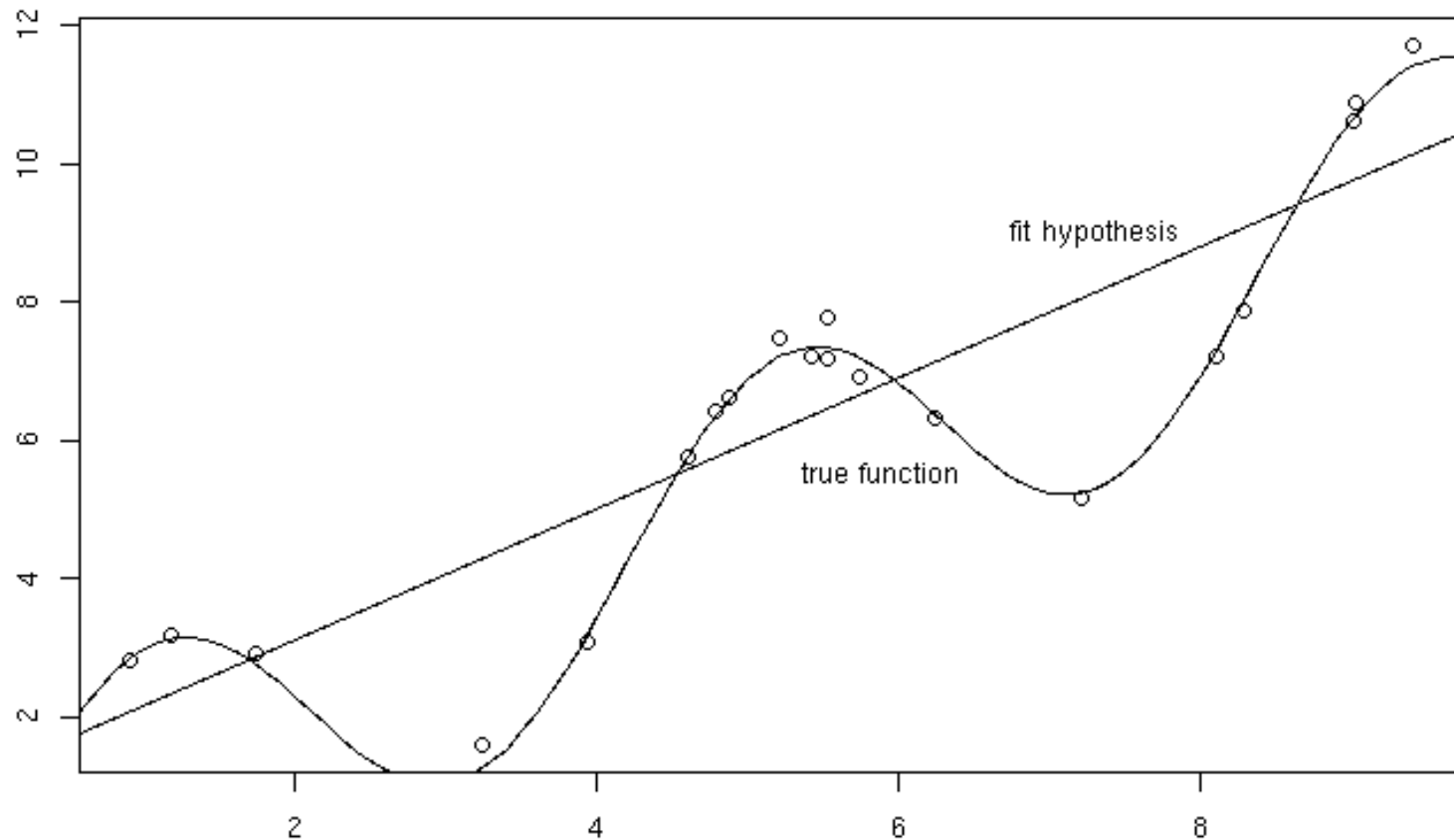
- True function is  $y = f(x) + \epsilon$ 
  - Where noise,  $\epsilon$ , is normally distributed with zero mean and standard deviation  $\sigma$
- Given a set of training examples,  $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$ , we fit a hypothesis  $g(x) = w^T x + b$  to the data to minimize the squared error

$$\sum_i [y^{(i)} - g(x^{(i)})]^2$$

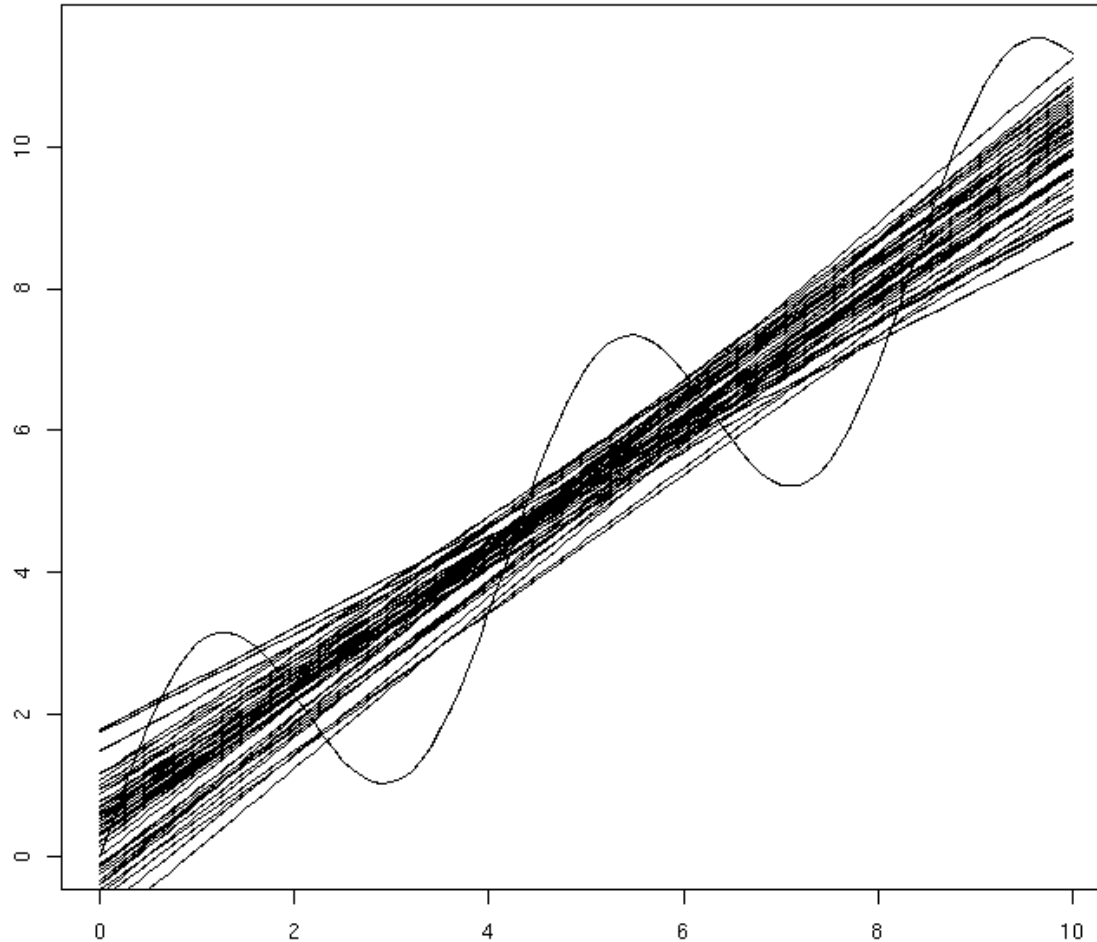
# 2-D Example



Sample 20 points from  $f(x) = x + 2 \sin(1.5x) + N(0,0.2)$



50 fits (20 examples each)



# Bias-Variance Analysis



- Given a new data point  $x'$  with observed value  $y' = f(x') + \epsilon$ , want to understand the expected prediction error
- Suppose that training samples are drawn independently from a distribution  $p(S)$ , want to compute the expected error of the estimator

$$E[ (y' - g_S(x'))^2 ]$$

- Variance of a random variable,  $Z$

$$\begin{aligned} \text{Var}(Z) &= E[(Z - E[Z])^2] \\ &= E[Z^2 - 2ZE[Z] + E[Z]^2] \\ &= E[Z^2] - E[Z]^2 \end{aligned}$$

- Properties of  $\text{Var}(Z)$

$$\text{Var}(aZ) = E[a^2Z^2] - E[aZ]^2 = a^2\text{Var}(Z)$$



$$\begin{aligned} E \left[ (y' - g_S(x'))^2 \right] &= E[g_S(x')^2 - 2g_S(x')y' + y'^2] \\ &= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \\ &= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\ &\quad + \text{Var}(y') + f(x')^2 \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2 \end{aligned}$$

$$\begin{aligned} E \left[ (y' - g_S(x'))^2 \right] &= E[g_S(x')^2 - 2g_S(x')y' + y'^2] \\ &= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \\ &= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\ &\quad + \text{Var}(y') + f(x')^2 \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2 \end{aligned}$$

The samples  $S$   
and the noise  
 $\epsilon$  are  
independent

$$\begin{aligned} E \left[ (y' - g_S(x'))^2 \right] &= E[g_S(x')^2 - 2g_S(x')y' + y'^2] \\ &= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \\ &= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\ &\quad + \text{Var}(y') + f(x')^2 \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2 \end{aligned}$$

Follows from  
definition of  
variance

$$\begin{aligned} E \left[ (y' - g_S(x'))^2 \right] &= E[g_S(x')^2 - 2g_S(x')y' + y'^2] \\ &= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \quad E[y'] = f(x') \\ &= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\ &\quad + \text{Var}(y') + f(x')^2 \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2 \end{aligned}$$

# Bias-Variance-Noise Decomposition



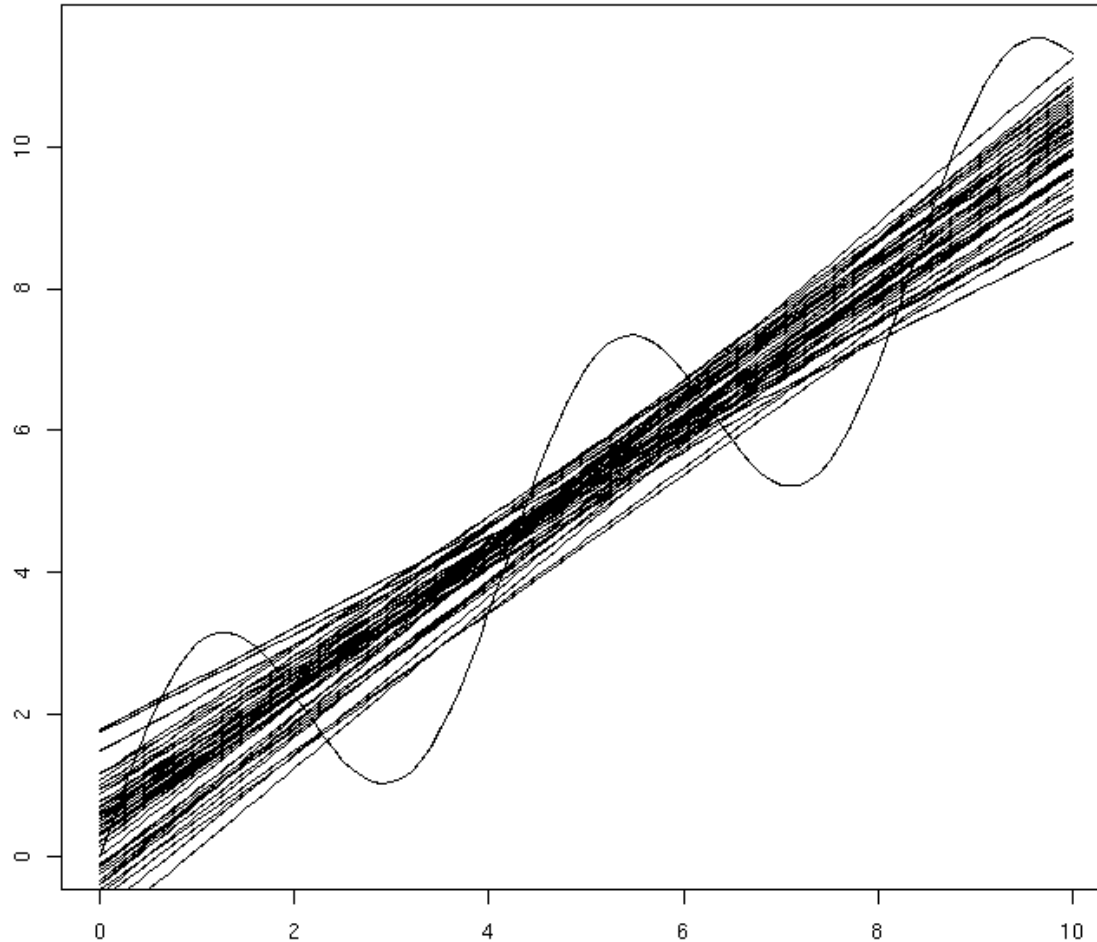
$$\begin{aligned} E \left[ (y' - g_S(x'))^2 \right] &= E[g_S(x')^2 - 2g_S(x')y' + y'^2] \\ &= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \\ &= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\ &\quad + \text{Var}(y') + f(x')^2 \\ &= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \\ &= \underbrace{\text{Var}(g_S(x'))}_{\text{Variance}} + \underbrace{(E[g_S(x')] - f(x'))^2}_{\text{Bias}} + \underbrace{\sigma^2}_{\text{Noise}} \end{aligned}$$

# Bias, Variance, and Noise

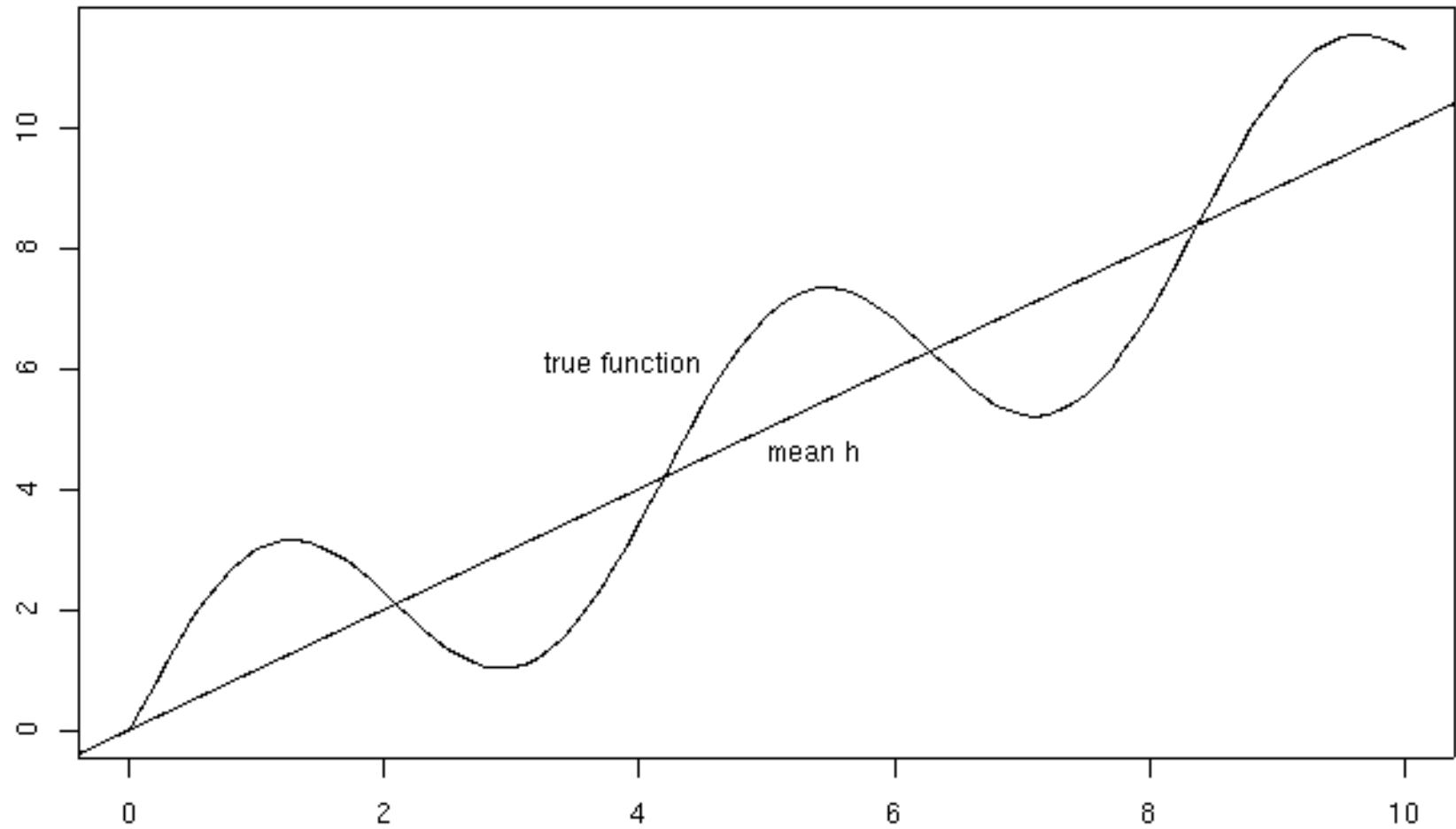


- Variance:  $E[ (g_S(x') - E[g_S(x')])^2 ]$ 
  - Describes how much  $g_S(x')$  varies from one training set  $S$  to another
- Bias:  $E[g_S(x')] - f(x')$ 
  - Describes the average error of  $g_S(x')$
- Noise:  $E[ (y' - f(x'))^2 ] = E[\epsilon^2] = \sigma^2$ 
  - Describes how much  $y'$  varies from  $f(x')$

50 fits (20 examples each)

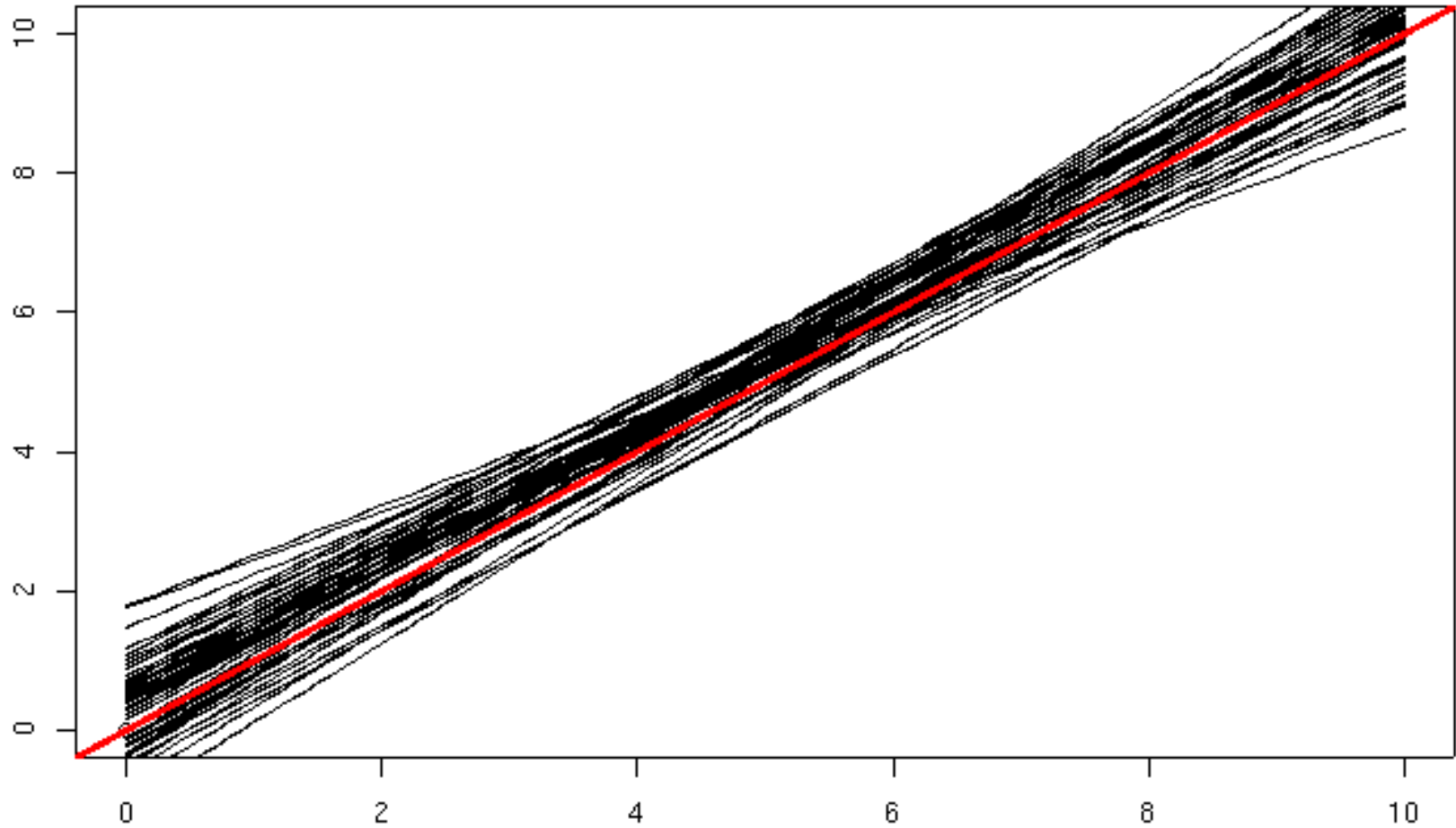


# Bias

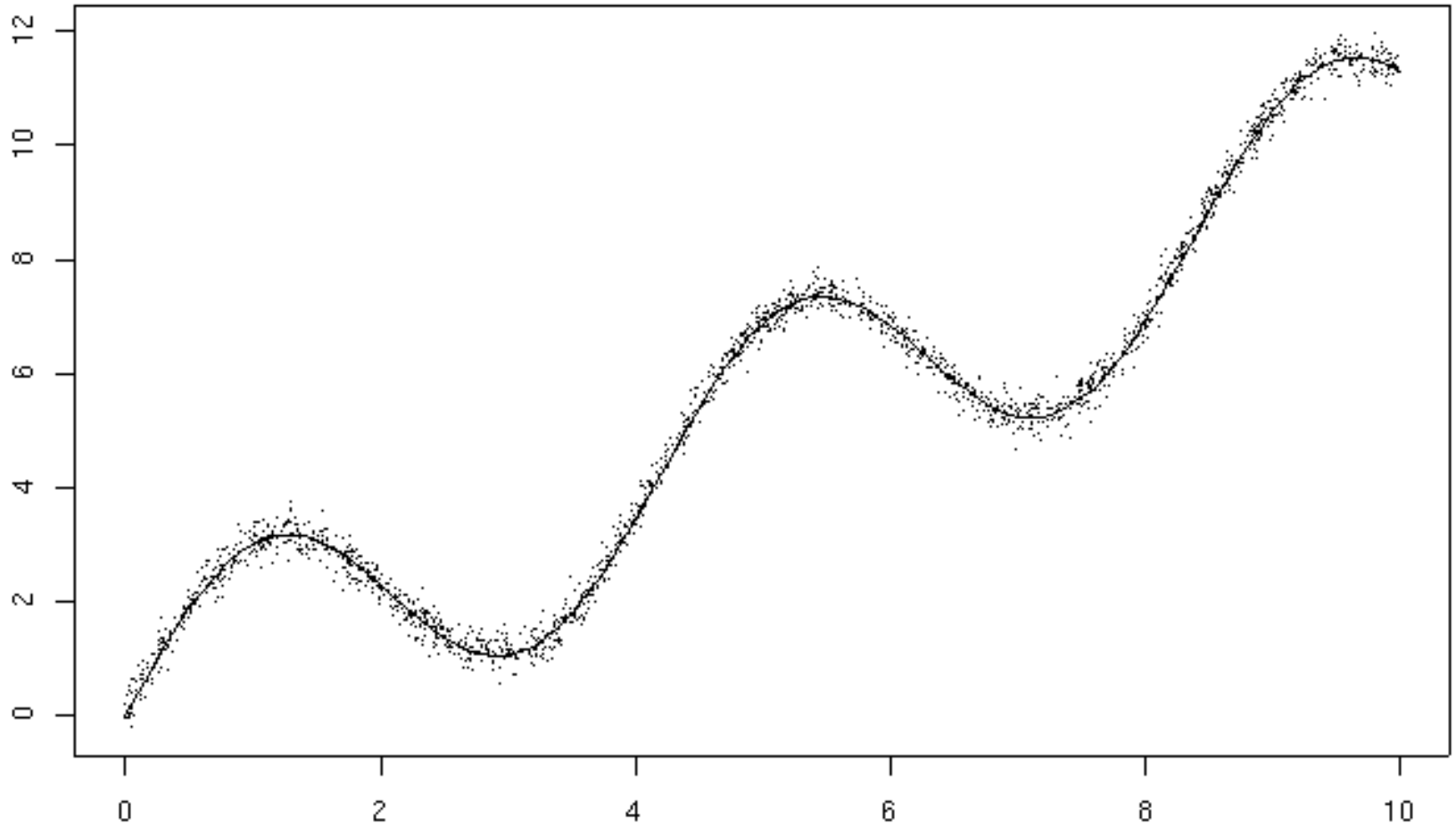




# Variance



# Noise



- Low bias
  - ?
- High bias
  - ?

- Low bias
  - Linear regression applied to linear data
  - 2nd degree polynomial applied to quadratic data
- High bias
  - Constant function applied to non-constant data
  - Linear regression applied to highly non-linear data

- Low variance
  - ?
- High variance
  - ?

- Low variance
  - Constant function
  - Model independent of training data
- High variance
  - High degree polynomial

# Bias/Variance Tradeoff



- $(\text{bias}^2 + \text{variance})$  is what counts for prediction
- As we saw in PAC learning, we often have
  - Low bias  $\Rightarrow$  high variance
  - Low variance  $\Rightarrow$  high bias
  - How can we deal with this in practice?

- **Averaging** reduces variance: let  $Z_1, \dots, Z_N$  be i.i.d random variables

$$\text{Var} \left( \frac{1}{N} \sum_i Z_i \right) = \frac{1}{N} \text{Var}(Z_i)$$

- Idea: average models to reduce model variance
- The problem
  - Only one training set
  - Where do multiple models come from?

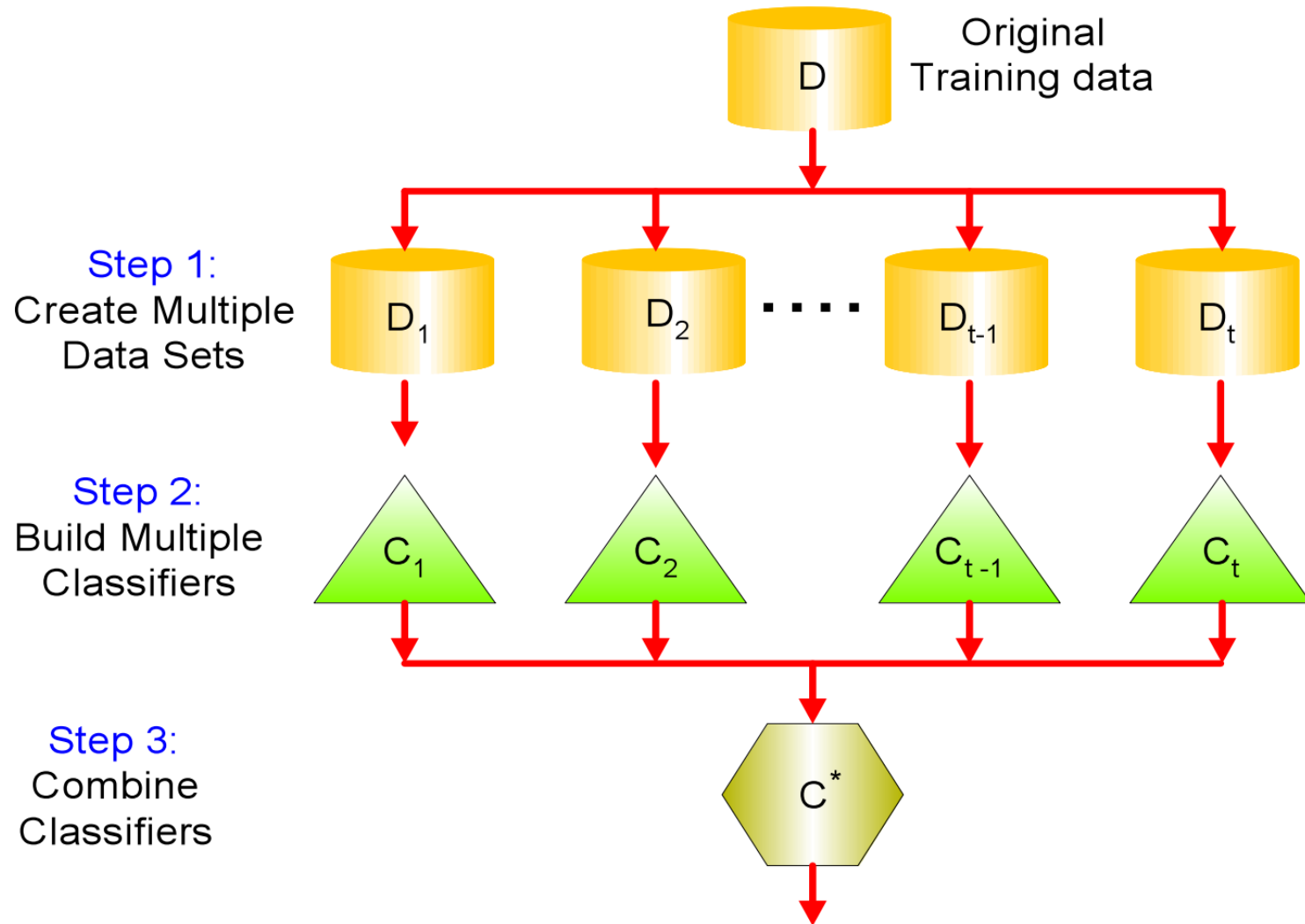


# Bagging: Bootstrap Aggregation



- Take repeated bootstrap samples from training set  $D$  (Breiman, 1994)
- **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  $D_1, \dots, D_k$
  - Train distinct classifier on each  $D_i$
  - Classify new instance by majority vote / average

# Bagging: Bootstrap Aggregation



# Bagging



Data	1	2	3	4	5	6	7	8	9	10
BS 1	7	1	9	10	7	8	8	4	7	2
BS 2	8	1	3	1	1	9	7	4	10	1
BS 3	5	4	8	8	2	5	5	7	8	8

- Build a classifier from each bootstrap sample
- In each bootstrap sample, each data point has probability  $\left(1 - \frac{1}{N}\right)^N$  of not being selected
- Expected number of distinct data points in each sample is then

$$N \cdot \left(1 - \left(1 - \frac{1}{N}\right)^N\right) \approx N \cdot (1 - \exp(-1)) = .632 \cdot N$$

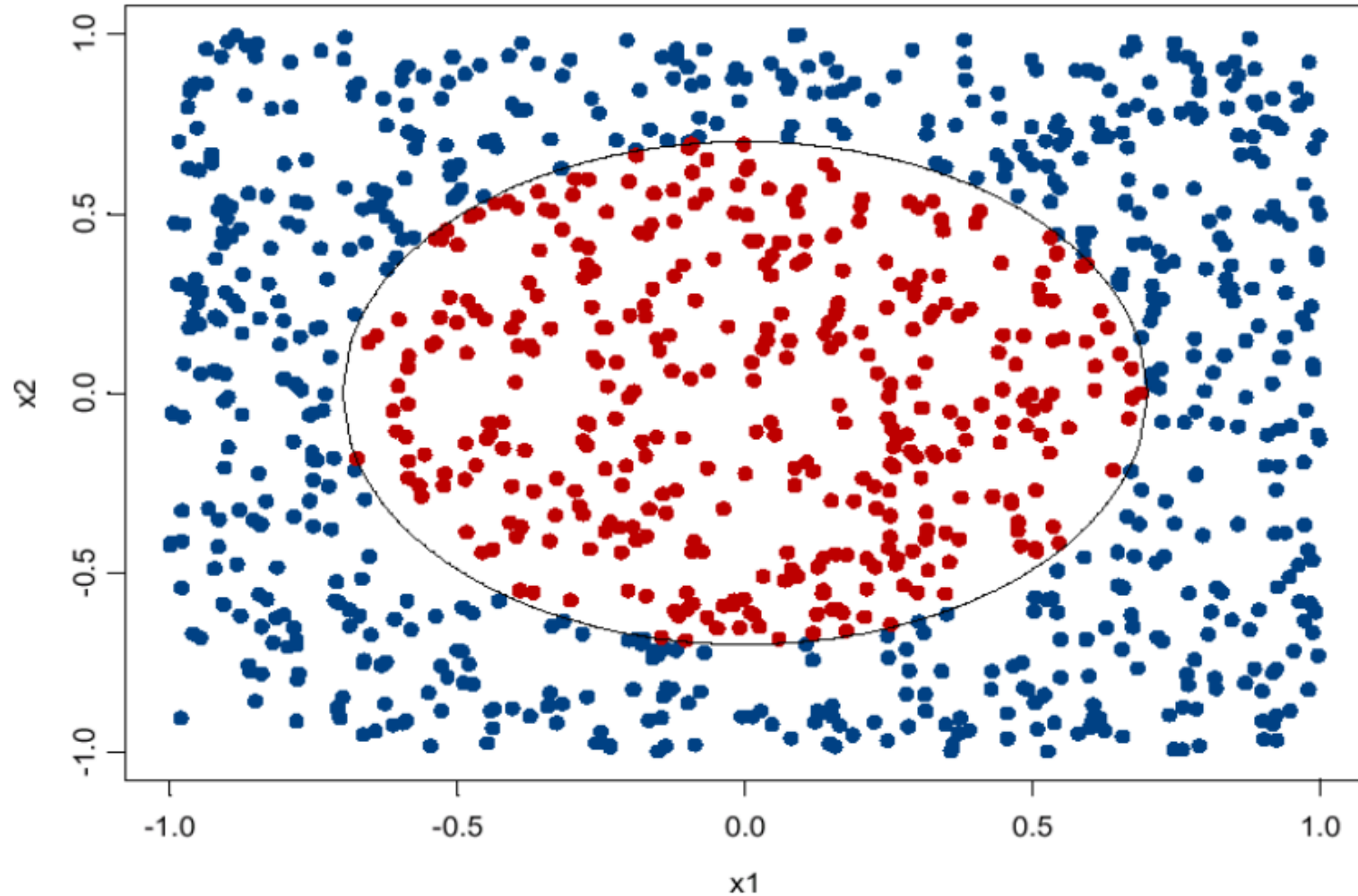
# Bagging



Data	1	2	3	4	5	6	7	8	9	10
BS 1	7	1	9	10	7	8	8	4	7	2
BS 2	8	1	3	1	1	9	7	4	10	1
BS 3	5	4	8	8	2	5	5	7	8	8

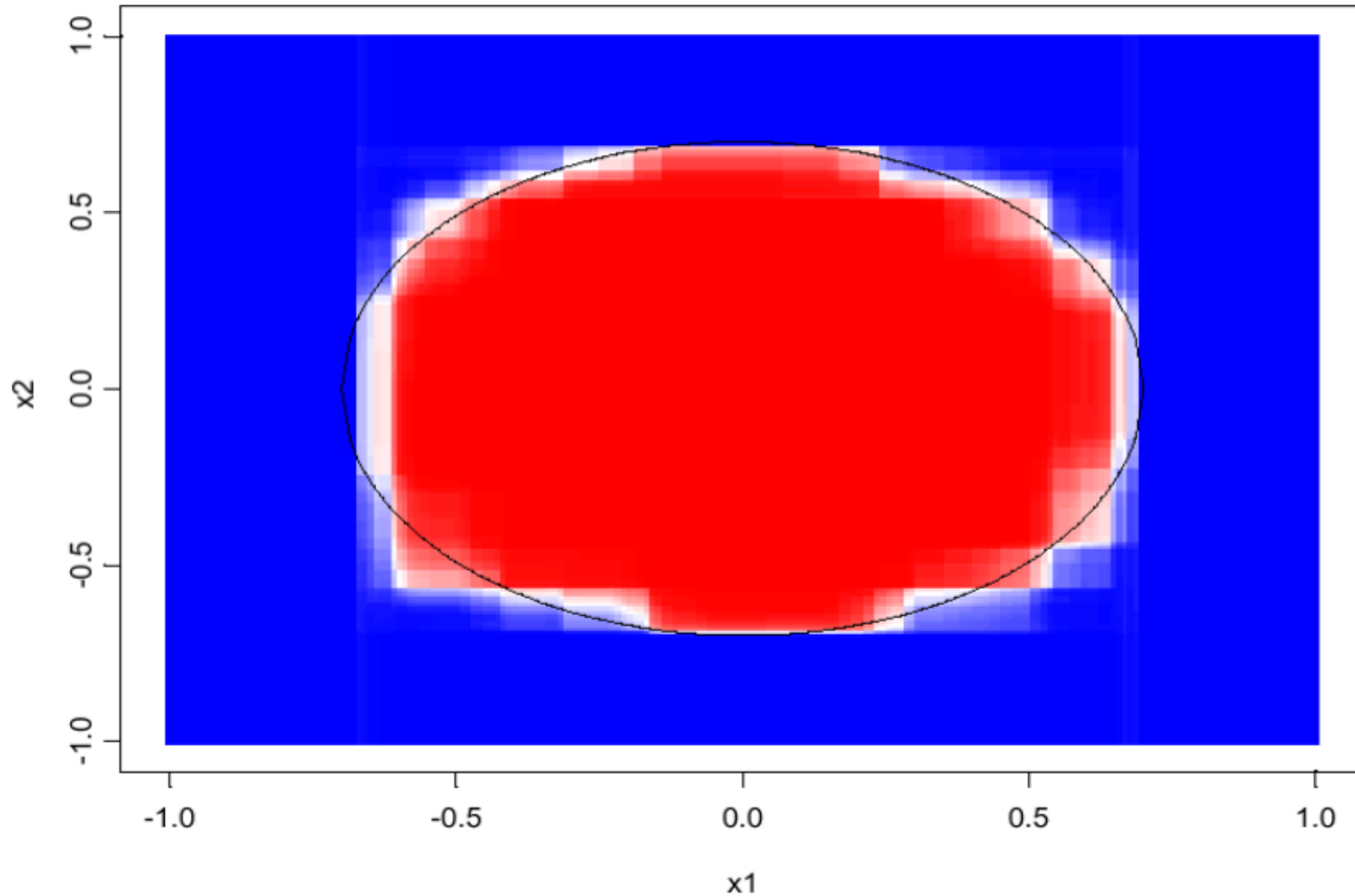
- Build a classifier from each bootstrap sample
- In each bootstrap sample, each data point has probability  $\left(1 - \frac{1}{N}\right)^N$  of not being selected
  - If we have 1 TB of data, each bootstrap sample will be ~ 632GB (this can present computational challenges, e.g., you shouldn't replicate the data)

# Decision Tree Bagging



[image from the slides of David Sontag]

# Decision Tree Bagging (100 Bagged Trees)



[image from the slides of David Sontag]

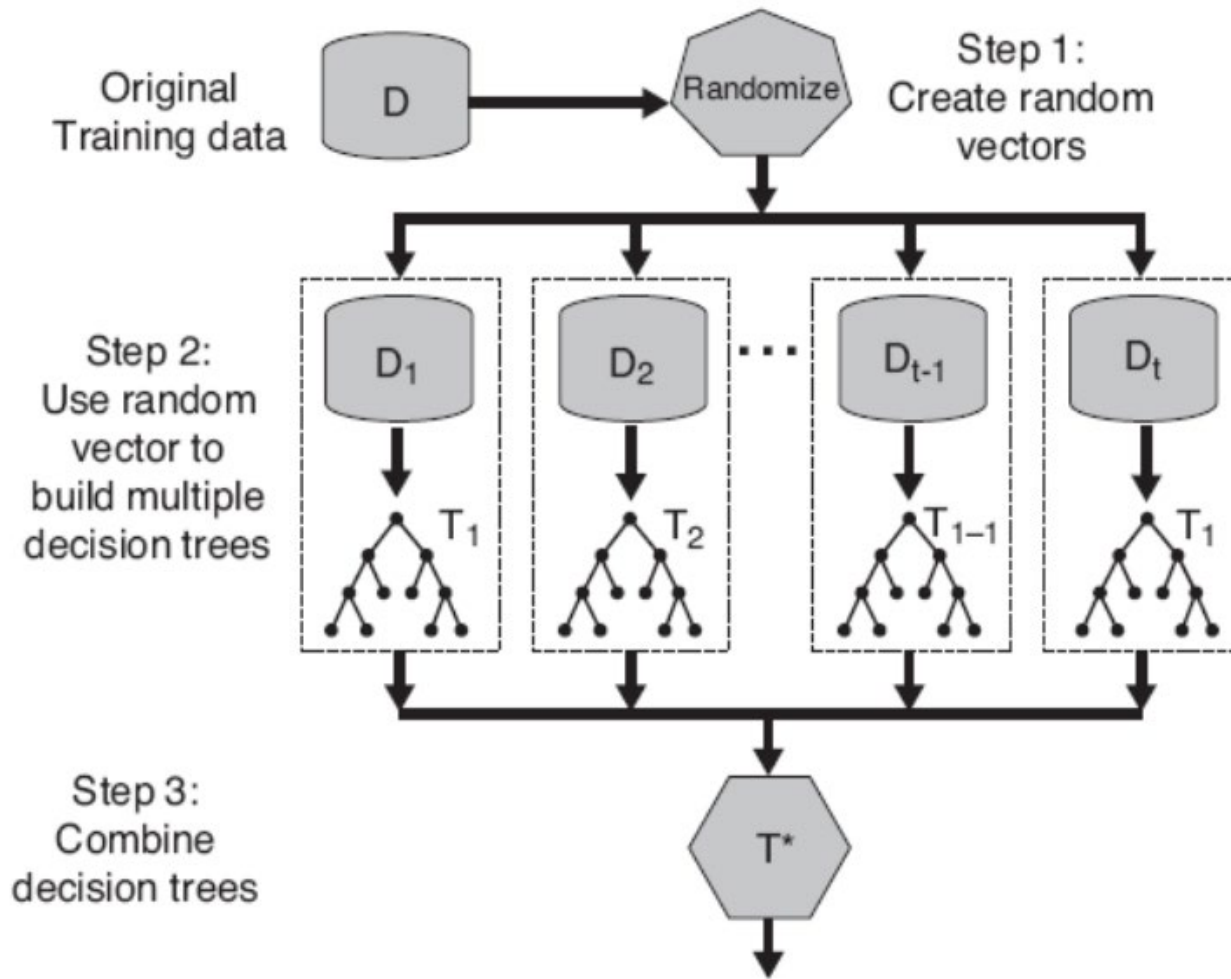
# Bagging Results



	Without Bagging	With Bagging	
<b>Data Set</b>	$\bar{e}_S$	$\bar{e}_B$	<b>Decrease</b>
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

# Random Forests





- Ensemble method specifically designed for decision tree classifiers
- Introduce two sources of randomness: “bagging” and “random input vectors”
  - Bagging method: each tree is grown using a bootstrap sample of training data
  - **Random vector method**: best split at each node is chosen from a random sample of  $m$  attributes instead of all attributes

# Random Forest Algorithm



- For  $b = 1$  to  $B$ 
  - Draw a bootstrap sample of size  $N$  from the data
  - Grow a tree  $T_b$  using the bootstrap sample as follows
    - Choose  $m$  attributes uniformly at random from the data
    - Choose the best attribute among the  $m$  to split on
    - Split on the best attribute and recurse (until partitions have fewer than  $s_{min}$  number of nodes)
- Prediction for a new data point  $x$ 
  - Regression:  $\frac{1}{B} \sum_b T_b(x)$
  - Classification: choose the majority class label among  $T_1(x), \dots, T_B(x)$

A [demo](#) of random forests implemented in JavaScript

# 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 causes a large change in the output hypothesis
  - If small changes in  $D$  cause large changes in the output, then there will likely be an improvement in performance with bagging
- Bagging can help unstable procedures, but could hurt the performance of stable procedures
  - Decision trees are unstable
  - $k$ -nearest neighbor is stable