Evaluating Classifiers

Reading for this topic:

T. Fawcett, An introduction to ROC analysis, Sections 1-4, 7

(linked from class website)

Evaluating Classifiers

- What we want: Classifier that best predicts unseen ("test") data

- Common assumption: Data is "iid" (independently and identically distributed)

Topics

- Cross Validation
- Precision and Recall
- ROC Analysis
- Bias, Variance, and Noise

Cross-Validation

Accuracy and Error Rate

- Accuracy = fraction of correct classifications on unseen data (test set)

- Error rate = 1 − Accuracy

How to use available data to best measure accuracy?

Split data into training and test sets.

But how to split?

Too little training data: Cannot learn a good model

Too little test data: Cannot evaluate learned model

Also, how to learn hyper-parameters of the model?

One solution: "*k*-fold cross validation"

- Used to better estimate generalization accuracy of model
- Used to learn hyper-parameters of model ("model selection")

Using *k*-fold cross validation to estimate accuracy

- Each example is used both as a training instance and as a test instance.

- Instead of splitting data into "training set" and "test set", split data into k disjoint parts: S_1, S_2, ..., S_k.

- For $i = 1$ to k

  Select S_i to be the "test set". Train on the remaining data, test on S_i, to obtain accuracy A_i.

- Report $\frac{1}{k}\sum_{i=1}^{k} A_i$ as the final accuracy.

Using k-fold cross validation to learn hyper-parameters

(e.g., learning rate, number of hidden units, SVM kernel, etc.)

- Split data into training and test sets. Put test set aside.
- Split training data into k disjoint parts: $S_1, S_2, \ldots, S_k$.
- Assume you are learning one hyper-parameter. Choose R possible values for this hyper-parameter.
- For $j = 1$ to R

  For $i = 1$ to k

  Select S_i to be the "validation set"

  Train the classifier on the remaining data using the jth value of the hyperparameter

  Test the classifier on S_i , to obtain accuracy $A_{i,j}$.

  Compute the average of the accuracies: $\bar{A}_j = \frac{1}{k}\sum_{i=1}^{k} A_{i,j}$

Choose the value j of the hyper-parameter with highest $\bar{A}_j$.

Retrain the model with all the training data, using this value of the hyper-parameter.

Test resulting model on the test set.

Precision and Recall

Evaluating classification algorithms

"Confusion matrix" for a given class *c*

Actual	**Predicted (or "classified")**	
	Positive (in class *c*)	Negative (not in class *c*)
Positive (in class *c*)	TruePositive	FalseNegative
Negative (not in class *c*)	FalsePositive	TrueNegative

Example: “A” vs. “B”

Assume “A” is positive class

Results from Perceptron:

Instance	Class	Perception Output
1	“A”	−1
2	“A”	+1
3	“A”	+1
4	“A”	−1
5	“B”	+1
6	“B”	−1
7	“B”	−1
1	“B”	−1

Confusion Matrix

Actual	Predicted	
	Positive	Negative
Positive	2	2
Negative	1	3

Accuracy:

Evaluating classification algorithms

“Confusion matrix” for a given class *c*

Actual	**Predicted (or “classified”)**	
	Positive (in class *c*)	Negative (not in class *c*)
Positive (in class *c*)	TruePositive	FalseNegative
Negative (not in class *c*)	FalsePositive	TrueNegative

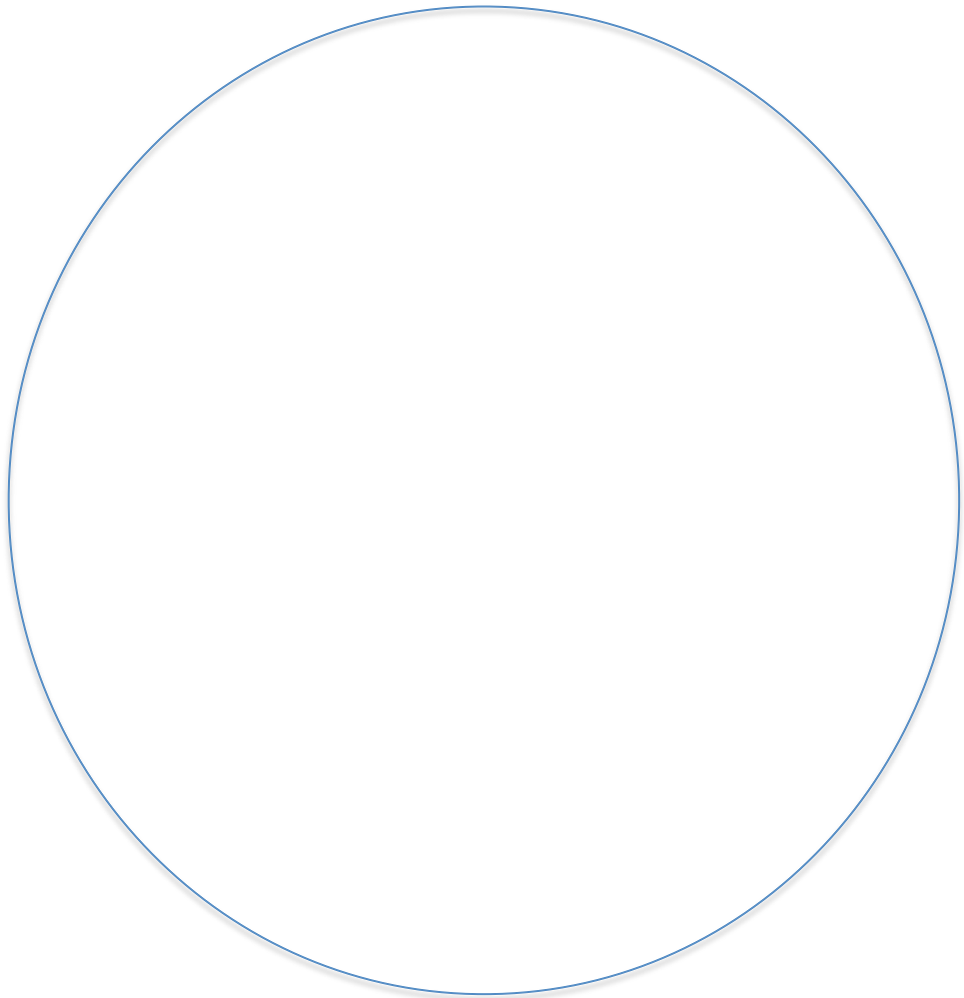
Type 2 error

Type 1 error

Exercise 1

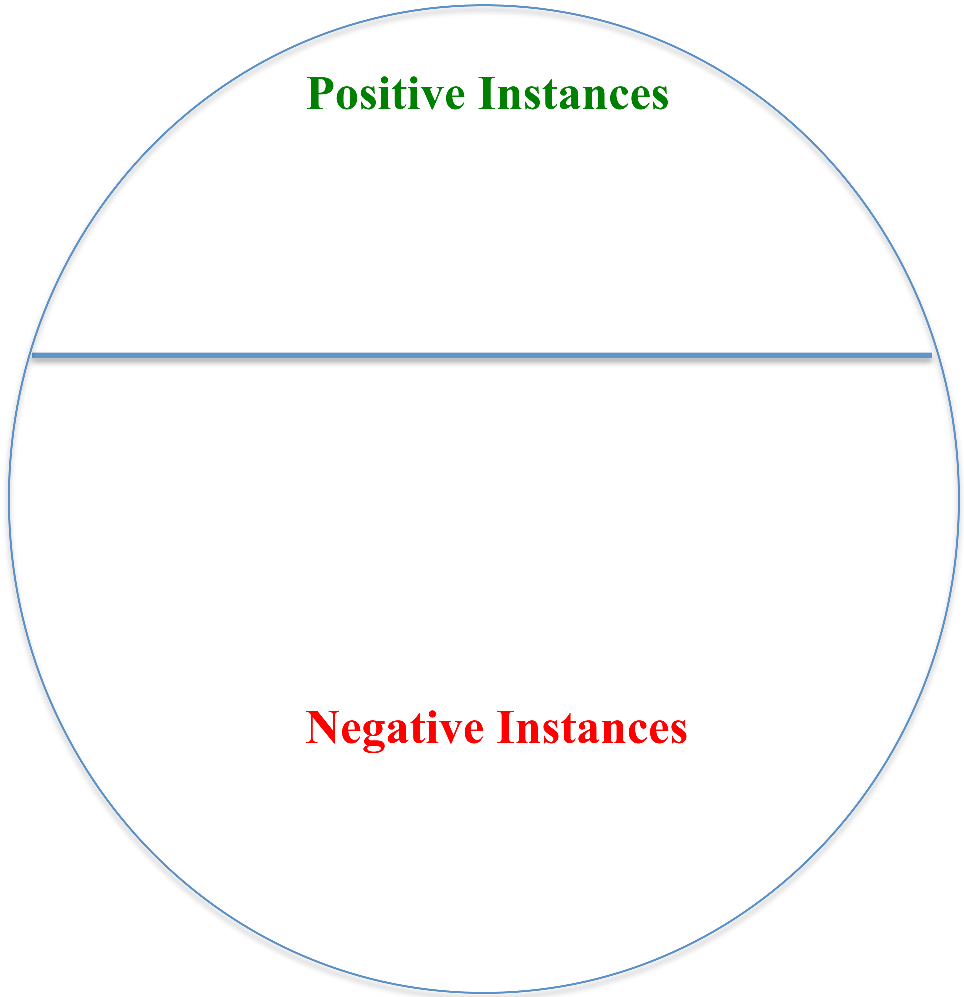
Precision and Recall

All instances in test set

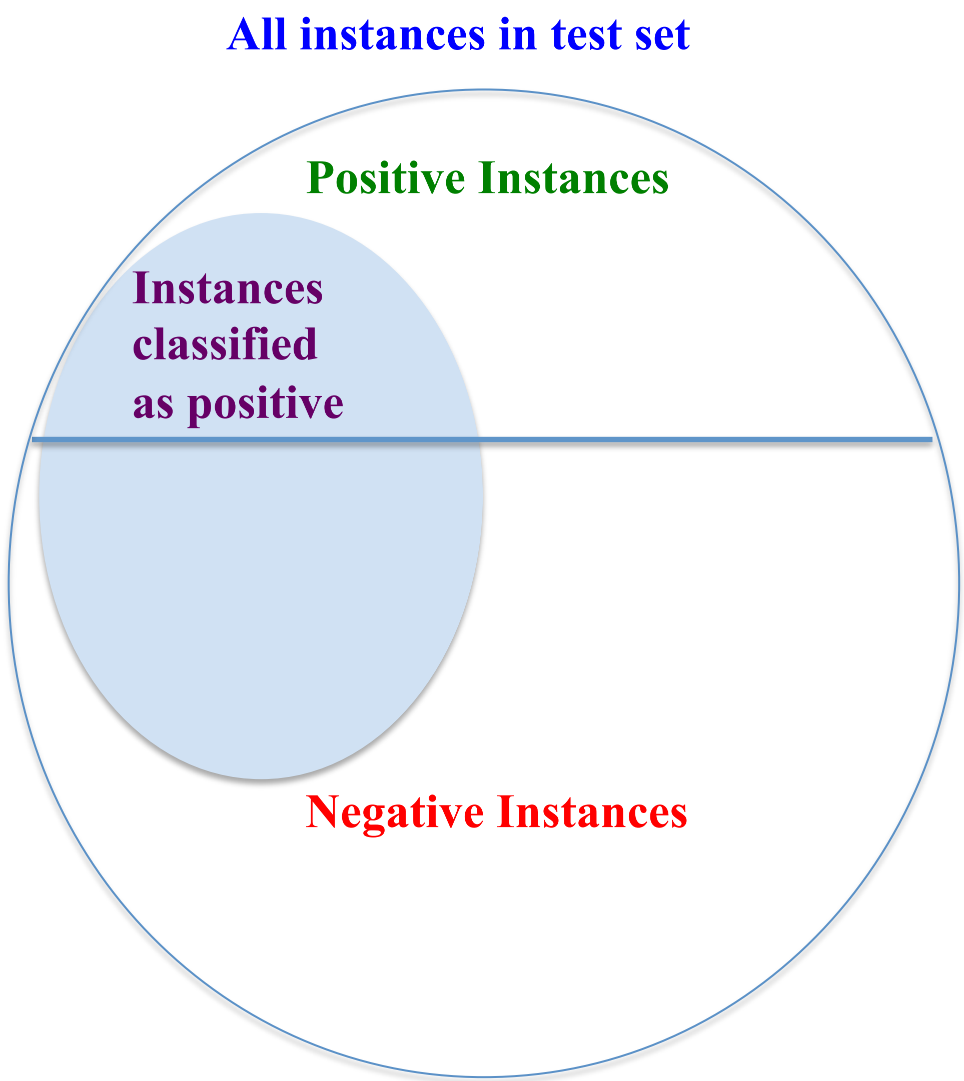


Precision and Recall

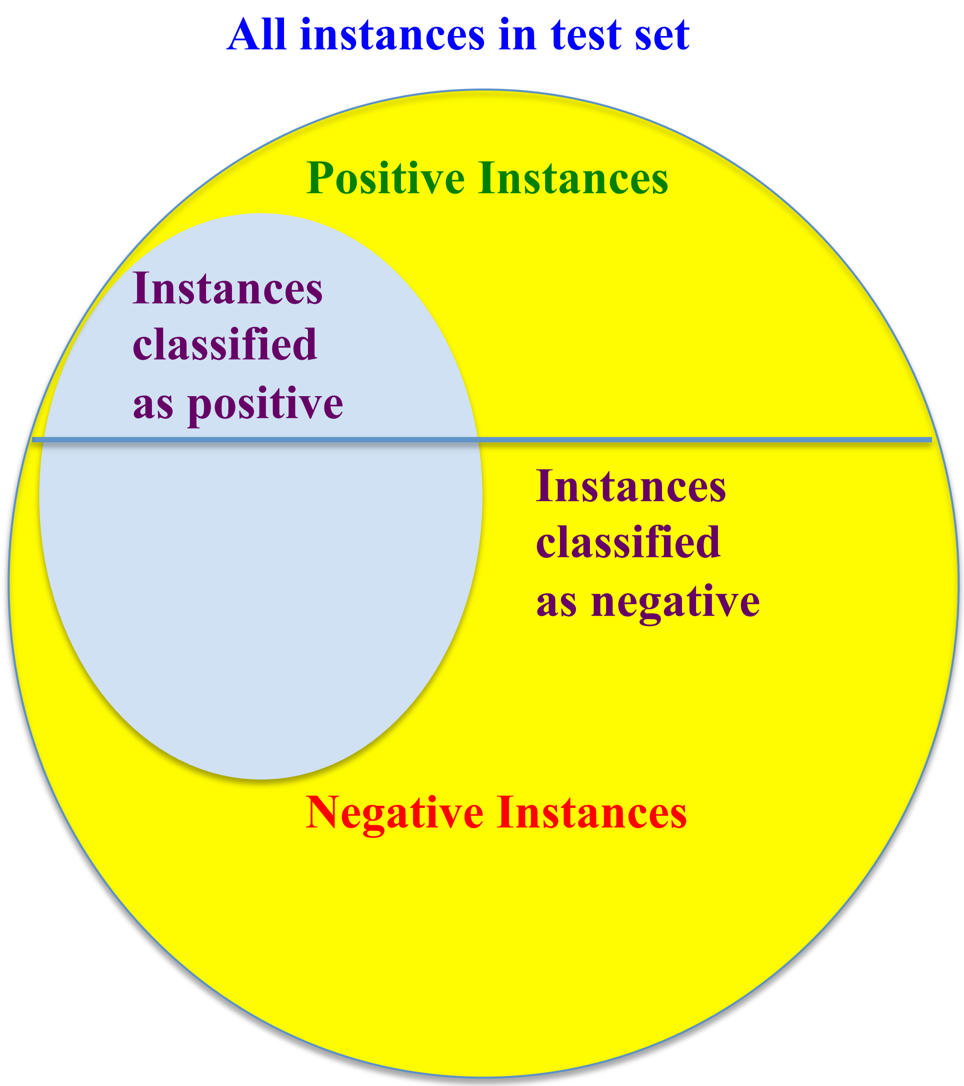
All instances in test set



Precision and Recall

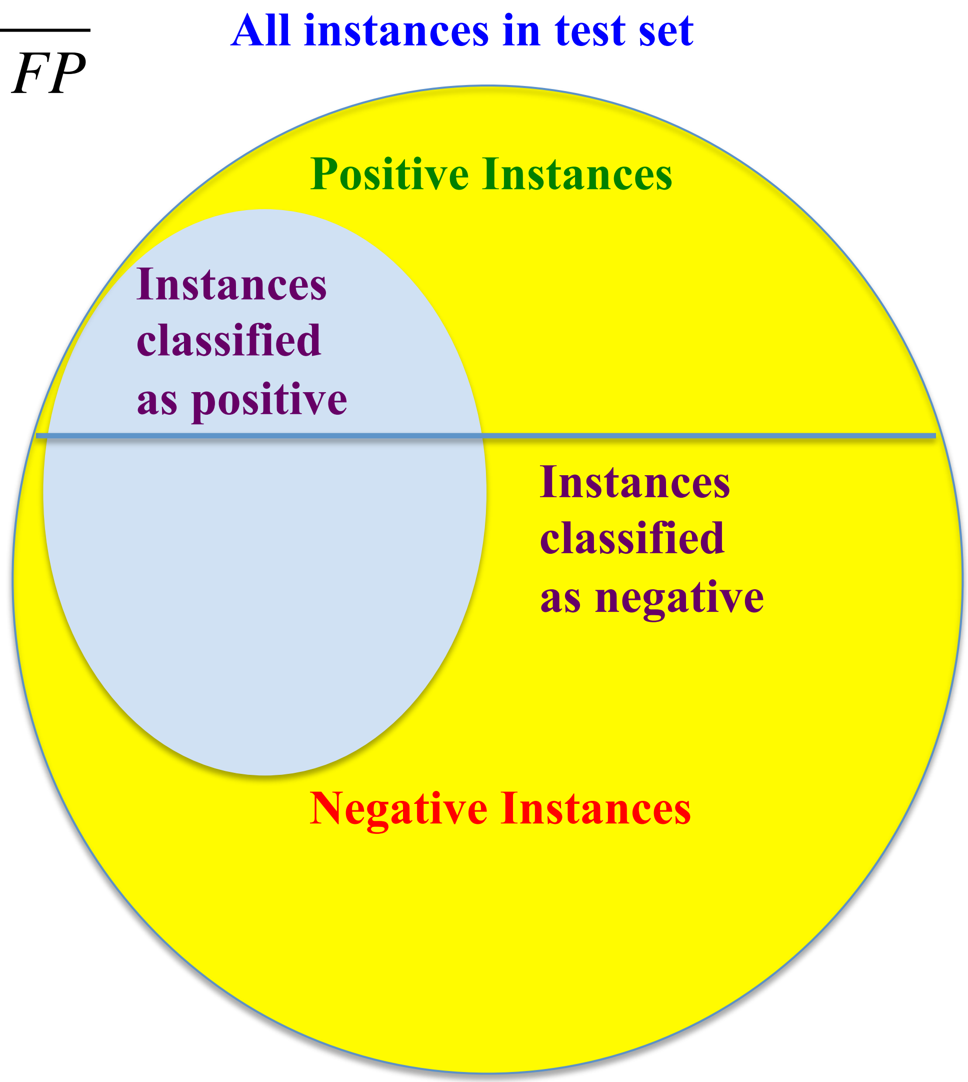


Precision and Recall



Precision and Recall

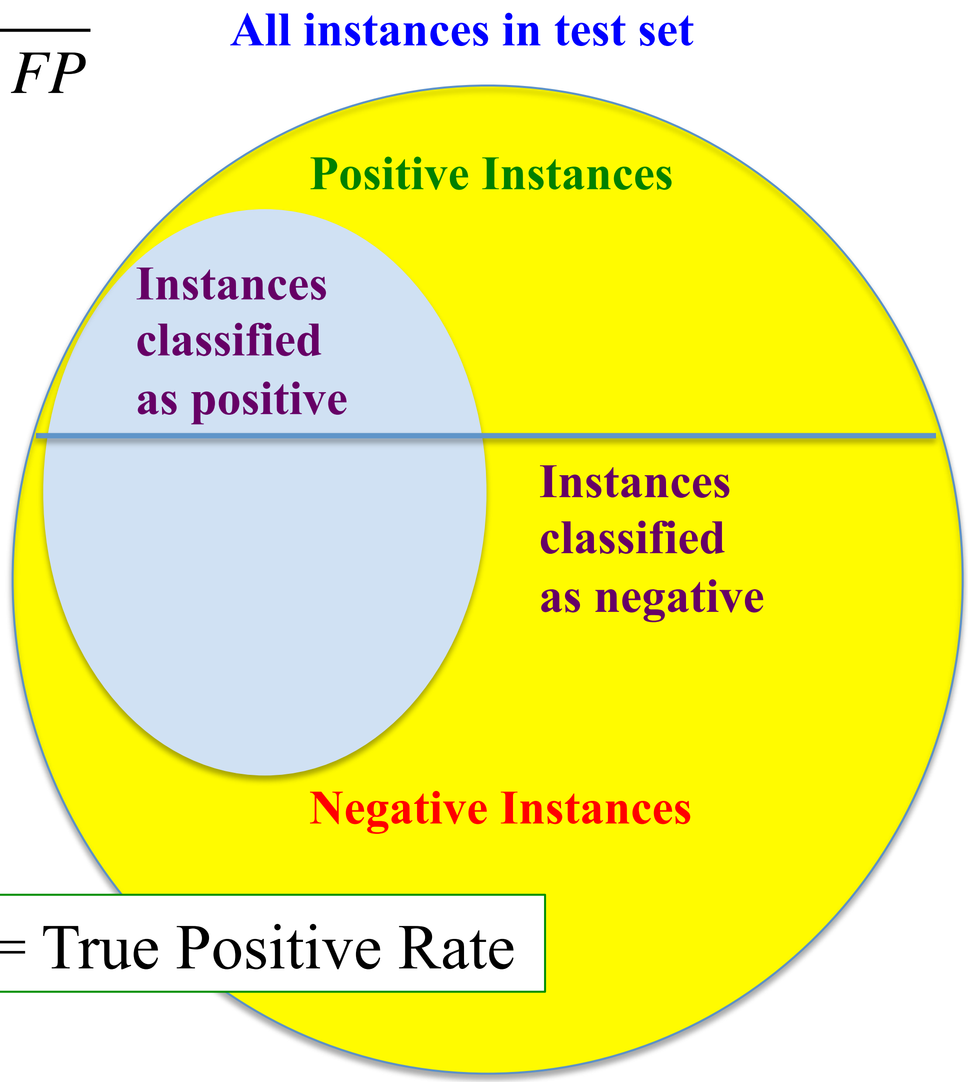
$$\text{Precision} = \frac{TP}{TP + FP}$$



Precision and Recall

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$



Recall = Sensitivity = True Positive Rate

Example: “A” vs. “B”

Assume “A” is positive class

Results from Perceptron:

Instance	Class	Perception Output
1	“A”	−1
2	“A”	+1
3	“A”	+1
4	“A”	−1
5	“B”	+1
6	“B”	−1
7	“B”	−1
1	“B”	−1

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F-measure} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Creating a Precision vs. Recall Curve

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Results of classifier

Threshold	Accuracy	Precision	Recall
.9	11/20	1	1/10
.8	12/20	1	2/10
.7			
.6			
.5			
.4			
.3			
.2			
.1	10/20	10/20	1
-∞			

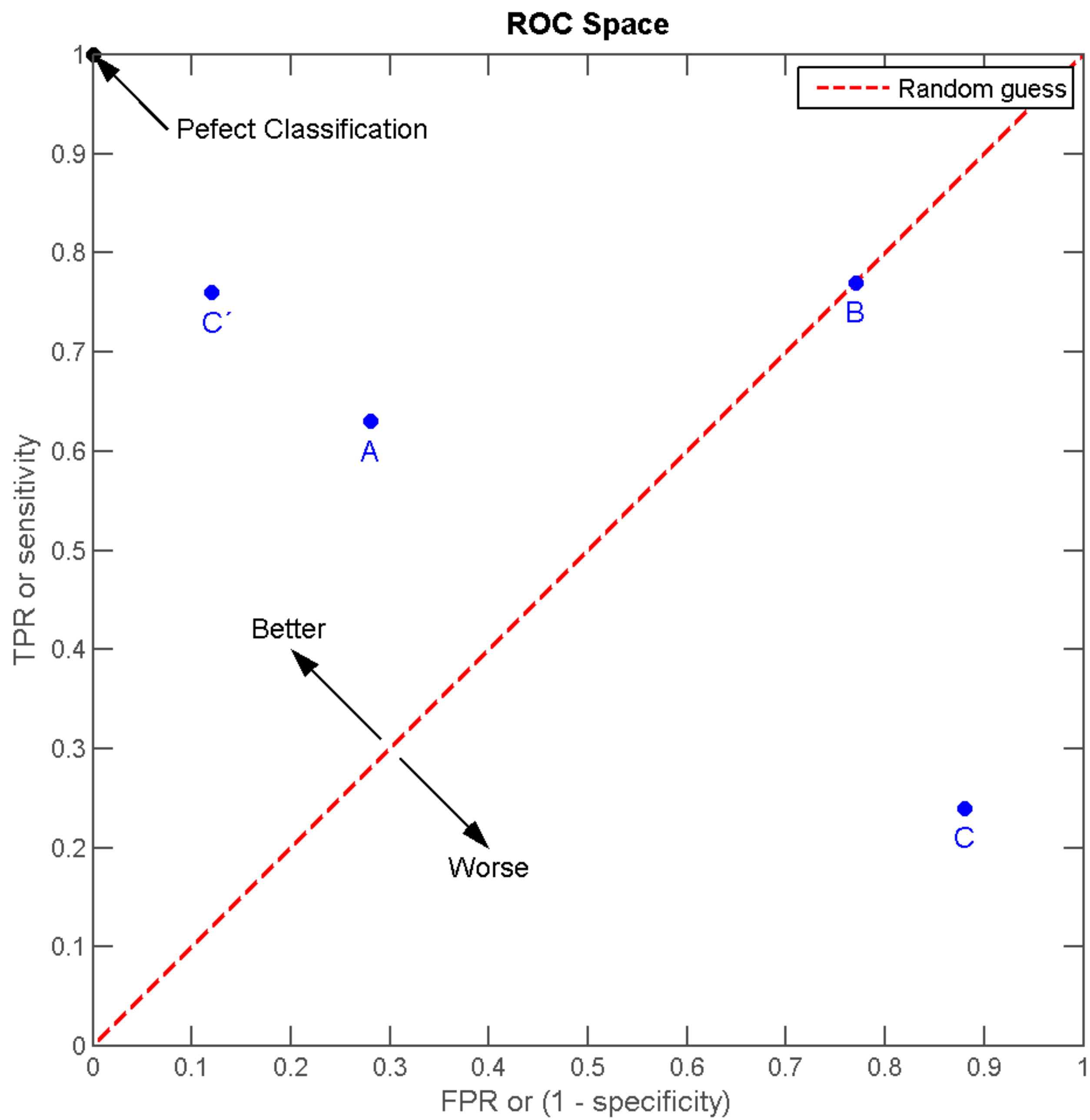
ROC Analysis

Receiver Operating Characteristic (ROC) Curves

- Alternative to precision/recall curves
- Shows tradeoff between true positive rate and false positive rate.

True positive rate = TP/(TP + FN) (“sensitivity”)

False positive rate = FP/(TN + FP) (1 - “specificity”)


ROC Space
Random guess
Pefect Classification
C´
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Creating a ROC Curve

$$\text{True Positive Rate (= Recall)} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate} = \frac{FP}{TN + FP}$$

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Results of classifier

Threshold	Accuracy	TPR	FPR
.9		1/10	0
.8		2/10	0
.7		2/10	1/10
.6			
.5			
.4			
.3			
.2			
.1		1	1
-∞			

Area under ROC curve (AUC)

- Summary statistic: Area under ROC curve (AUC) = probability that classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

- AUC is always between 0 and 1.

How to create a ROC curve for a perceptron

- Run your classifier on each instance in the test data, without doing the ***sgn*** step:

$$Score(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$$

- Get the range [*min*, *max*] of your scores

- Divide the range into about 200 thresholds, including $-\infty$ and *max*

- For each threshold, calculate TPR and FPR

- Plot TPR (y-axis) vs. FPR (x-axis)

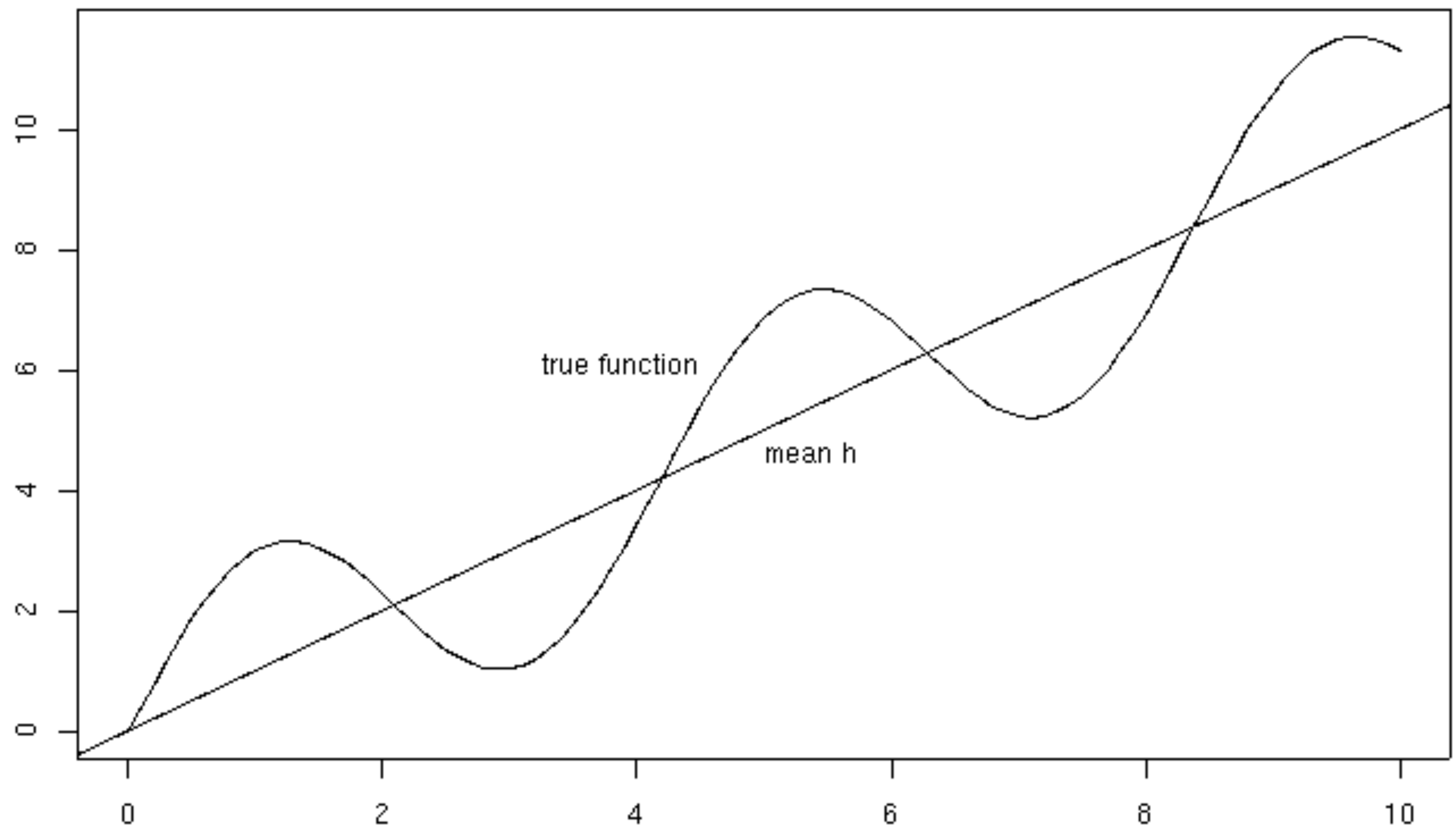
In-Class Exercise 1

Bias, Variance, and Noise

Bias:

Classifier is not powerful enough to represent the true function; that is, it *underfits* the function

.

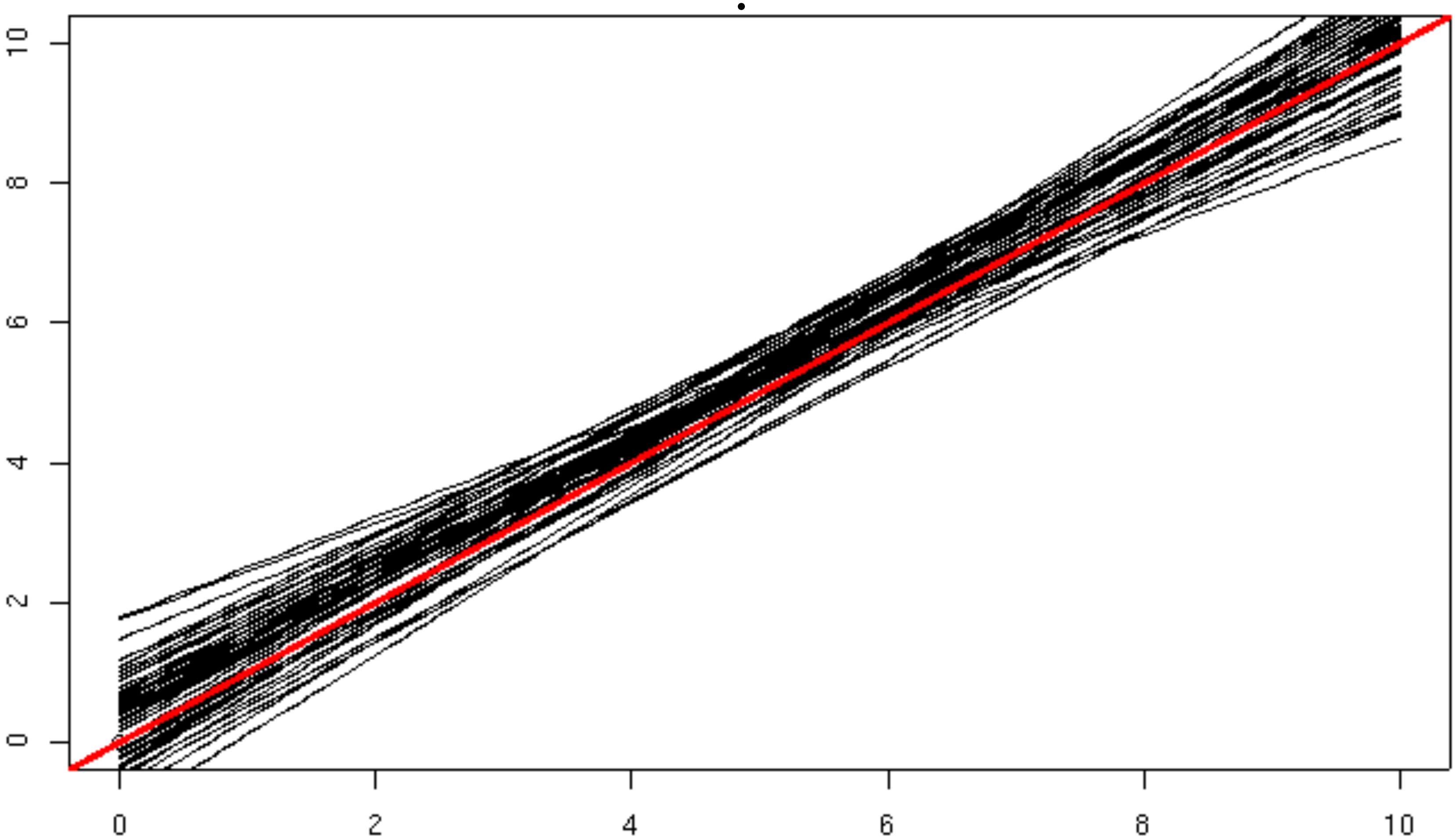


From http:// eecs.oregonstate.edu/~tgd/talks/BV.ppt

Variance:

Classifier's hypothesis depends on specific training set; that is, it *overfits* the function

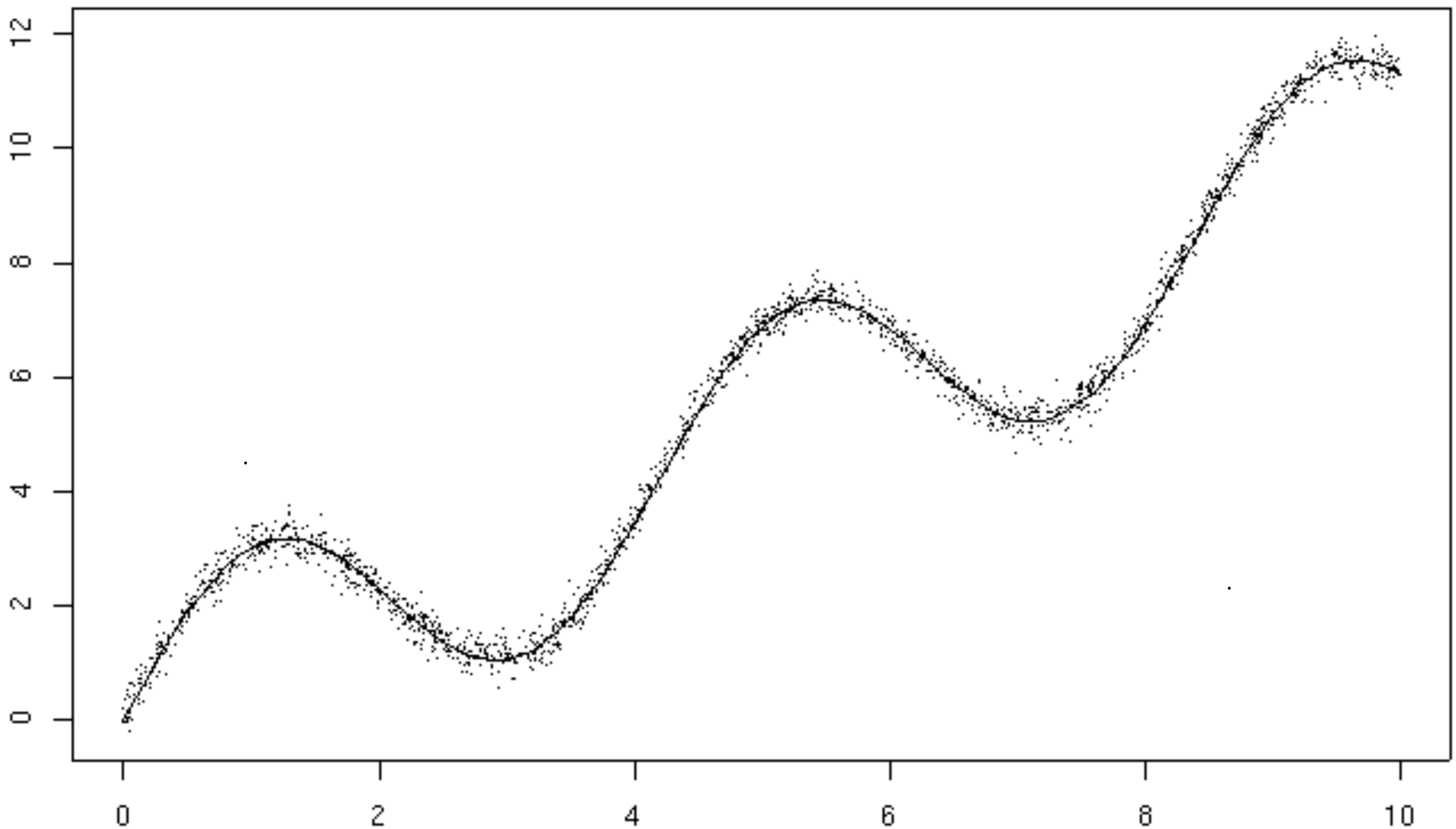
.



From http:// eecs.oregonstate.edu/~tgd/talks/BV.ppt

Noise:

Underlying process generating data is stochastic, or data has errors or outliers



From http:// eecs.oregonstate.edu/~tgd/talks/BV.ppt

- Examples of bias?
- Examples of variance?
- Examples of noise?

From http://lear.inrialpes.fr/~verbeek/MLCR.10.11.php

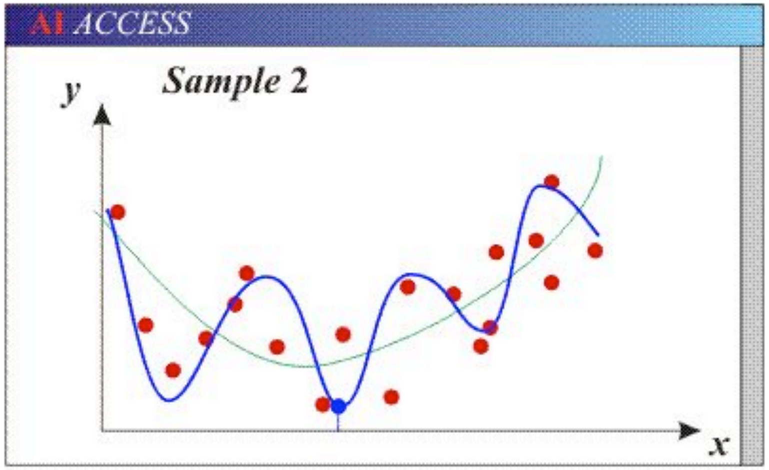
Bias/variance tradeoff

- Models with too many parameters may fit the training data well (low bias), but are sensitive to choice of training set (high variance)

- Generalization error is due to overfitting

- Models with too few parameters may not fit the data well (high bias) but are consistent across different training sets (low variance)

- Generalization error is due to underfitting



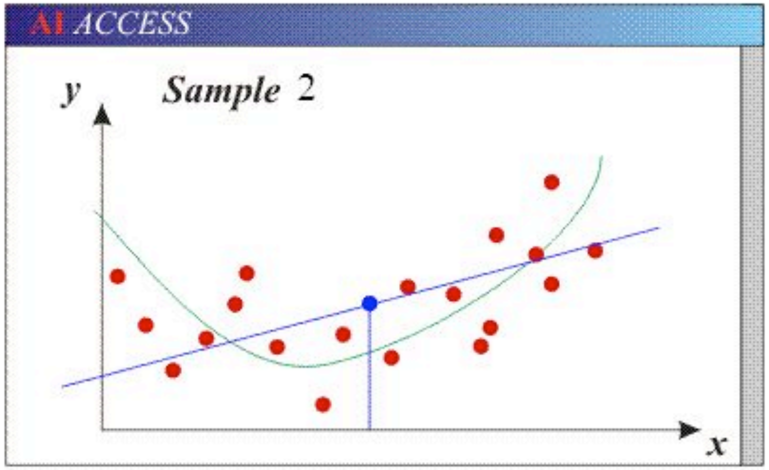
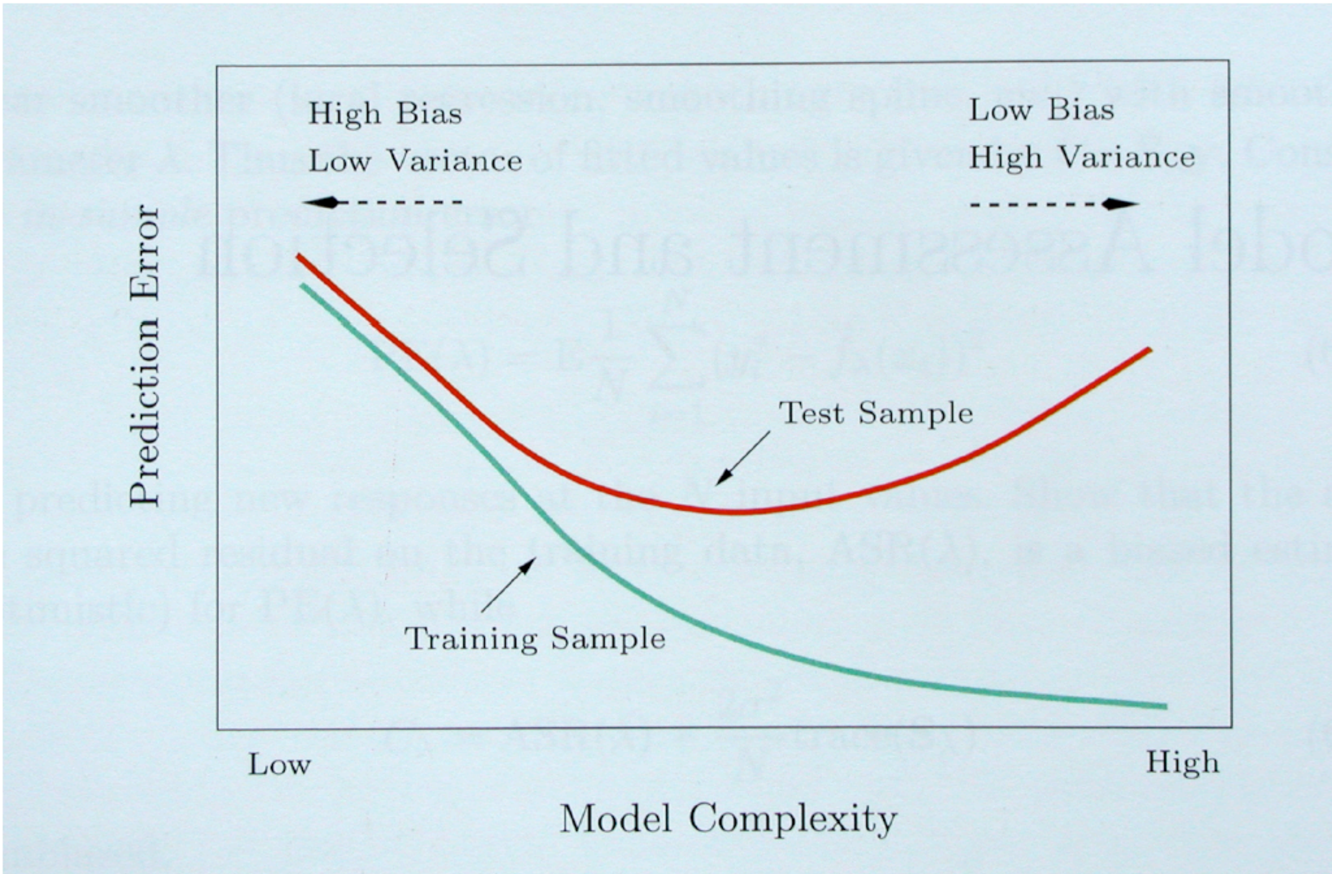


Illustration of Bias / Variance Tradeoff



Hastie, Tibshirani, Friedman “Elements of Statistical Learning” 2001

Model Error Decomposition: The Math

Let $h(x)$ be our learned model, which estimates true function $f(x)$.

$$Error(x) = E\left[\left(f(x) - h(x)\right)^2\right]$$

$$= \left(E[h(x)] - f(x)\right)^2 + E\left[h(x) - E[h(x)]\right]^2 + \sigma_e^2$$

$$= \text{bias}^2 + \text{variance}^2 + \text{irreducible error}$$

In-Class Exercise 2