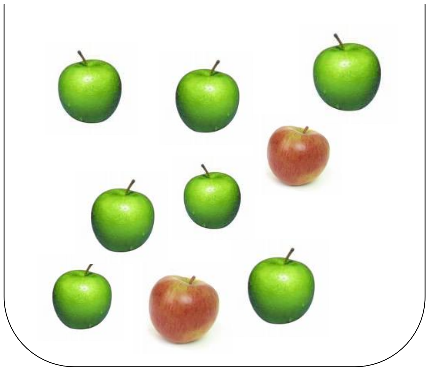
CS 484
Data Mining

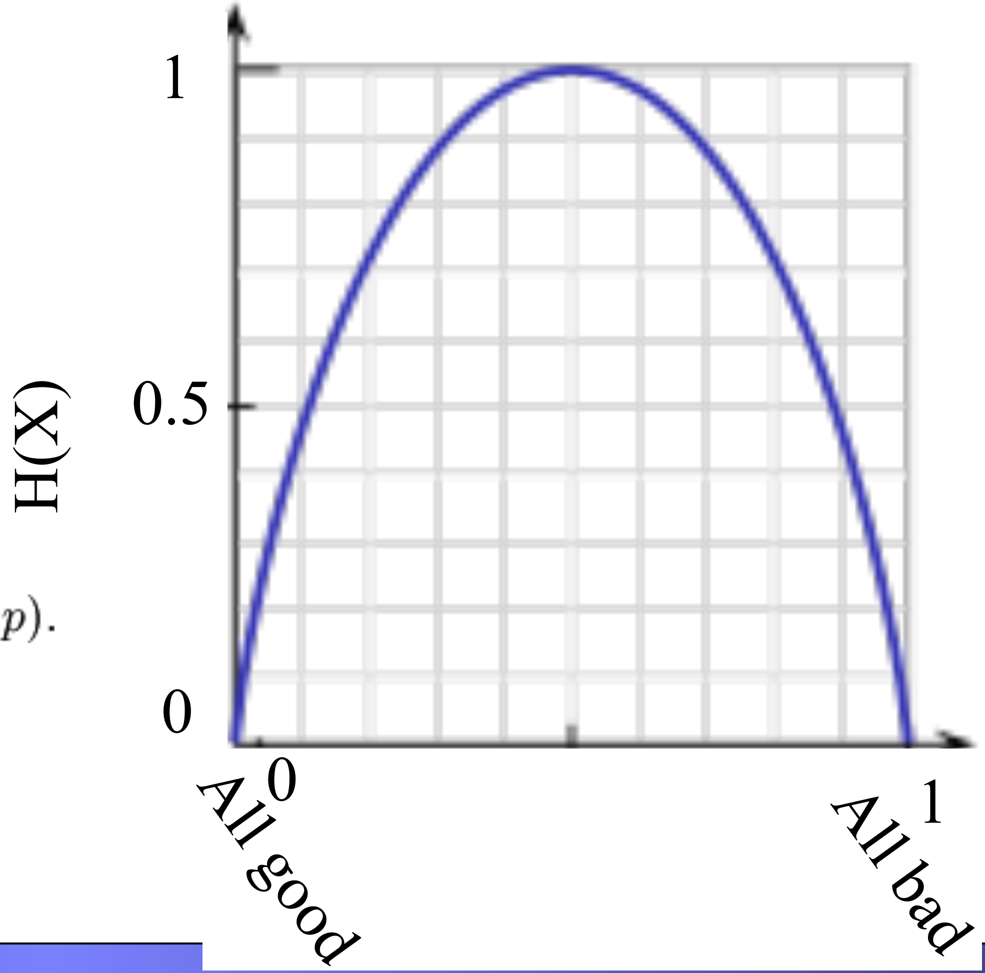
Classification 4

Some slides are from Professor Eamonn Keogh at UC Riverside

Entropy



I have a box of apples…



Pr(X = good) = p
then Pr(X = bad) = 1 − p
the entropy of X is given by

$$H(X) = H_{\mathrm{b}}(p) = -p \log p - (1-p) \log(1-p).$$

binary entropy function attains its maximum value when p = 0.5

Practical Issues of Classification

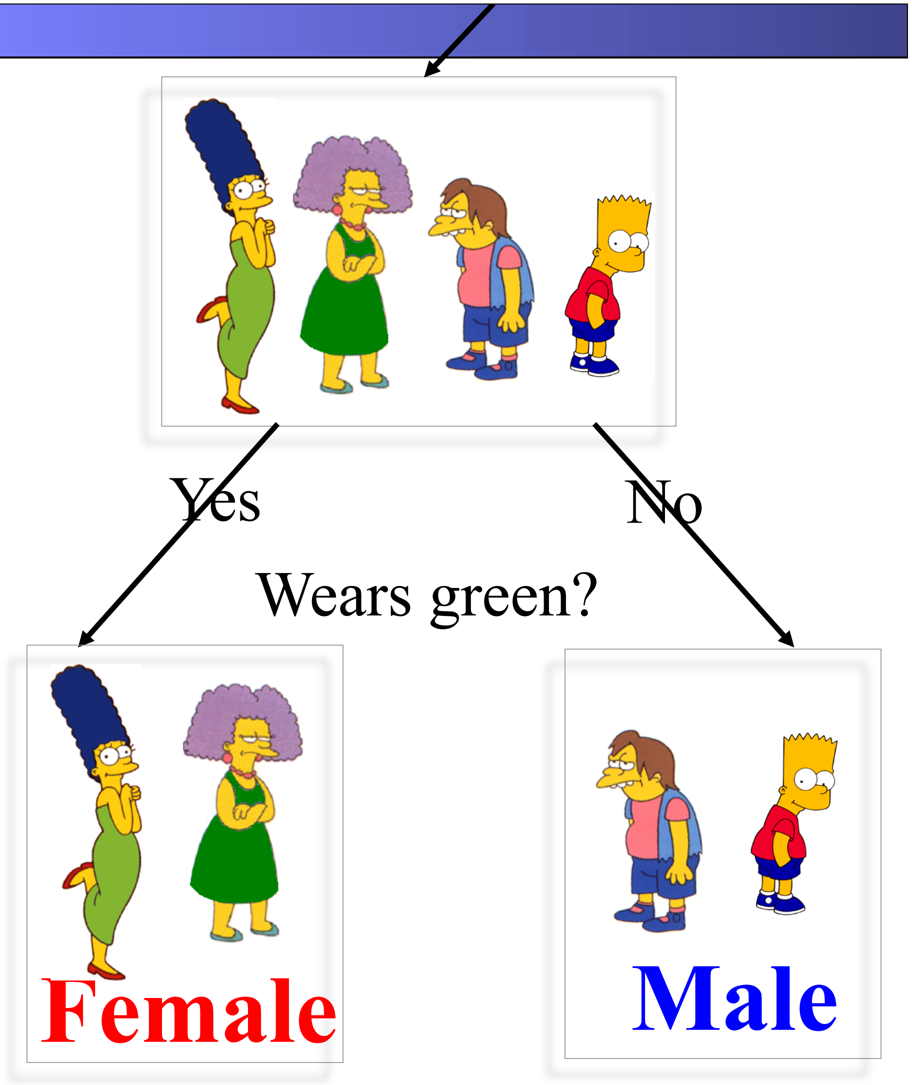
- Underfitting and Overfitting

- Missing Values

- Costs of Classification

The previous examples we have seen were performed on small datasets. However with small datasets there is a great danger of overfitting the data…

When you have few data points, there are many possible splitting rules that perfectly classify the data, but will not generalize to future datasets.



Yes

No

Wears green?

Female

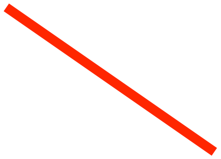
Male

For example, the rule “Wears green?” perfectly classifies the data, so does “Mother’s name is Jacqueline?”, so does “Has blue shoes”…

Suppose we need to solve a classification problem

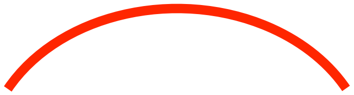
We are not sure if we should use the..

- Simple linear classifier



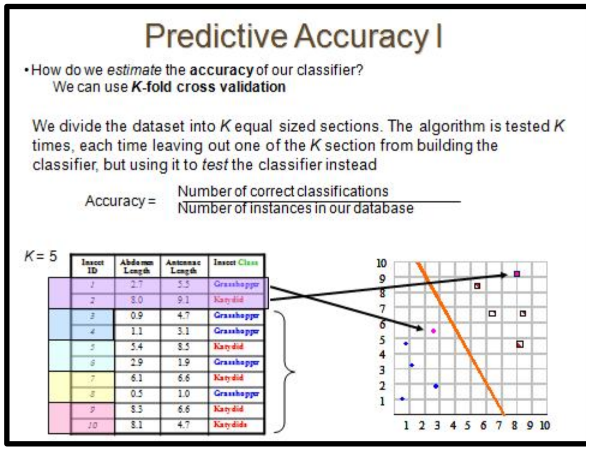
or the

- Simple quadratic classifier



How do we decide which to use?

We do cross validation (discussed later) and choose the best one.



Predictive Accuracy I

- How do we *estimate* the **accuracy** of our classifier?
  We can use ***K*-fold cross validation**

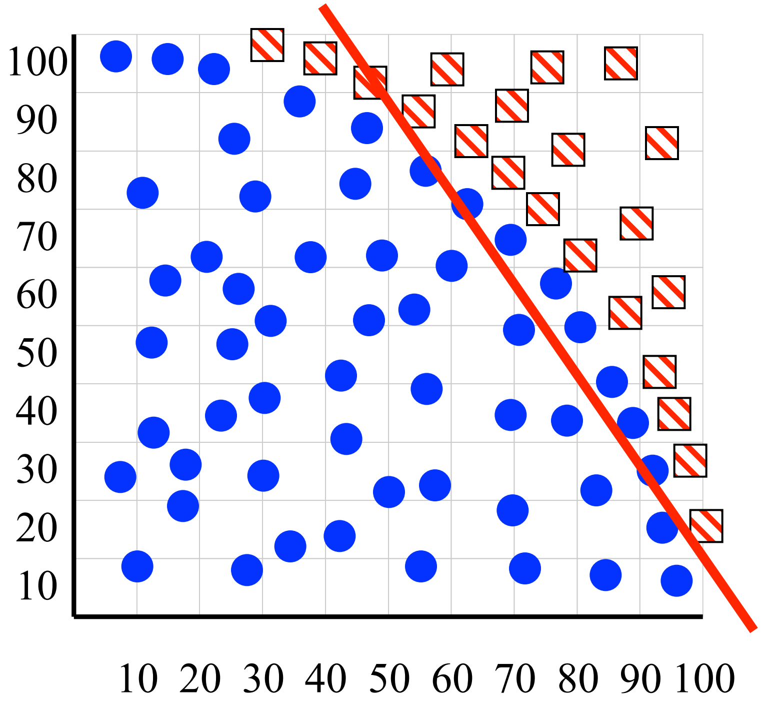
We divide the dataset into *K* equal sized sections. The algorithm is tested *K* times, each time leaving out one of the *K* section from building the classifier, but using it to *test* the classifier instead

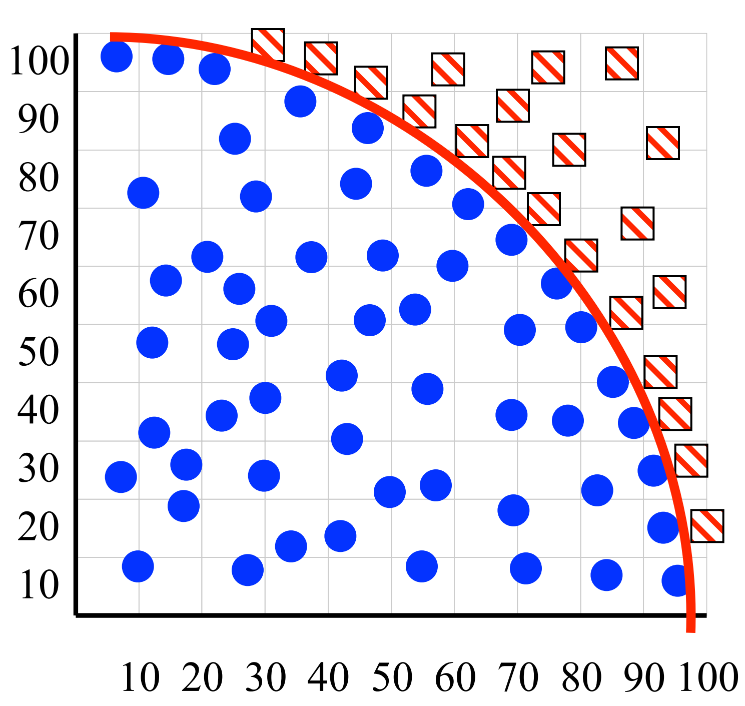
$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our database}}$$

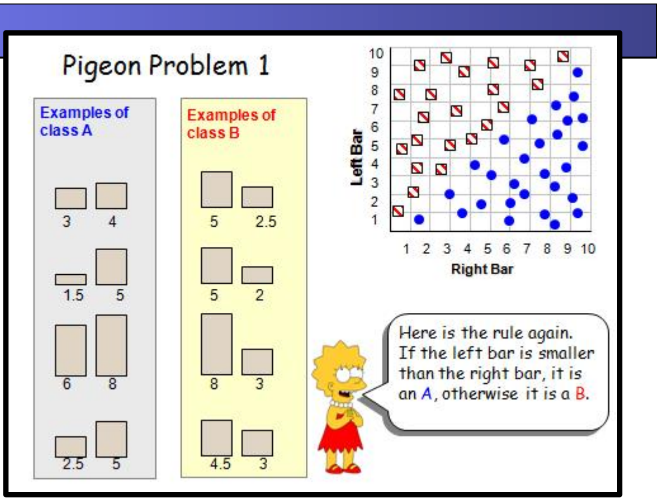
K = 5

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	Grasshopper
2	8.0	9.1	Katydid
3	0.9	4.7	Grasshopper
4	1.1	3.1	Grasshopper
5	5.4	8.5	Katydid
6	2.9	1.9	Grasshopper
7	6.1	6.6	Katydid
8	0.5	1.0	Grasshopper
9	8.3	6.6	Katydid
10	8.1	4.7	Katydids

- Simple linear classifier gets 81% accuracy
- Simple quadratic classifier gets 99% accuracy



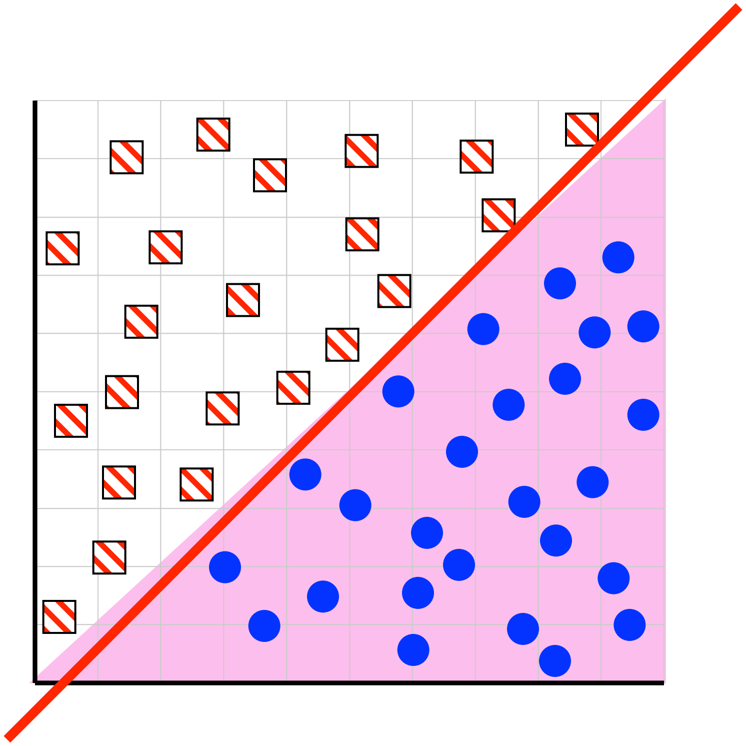


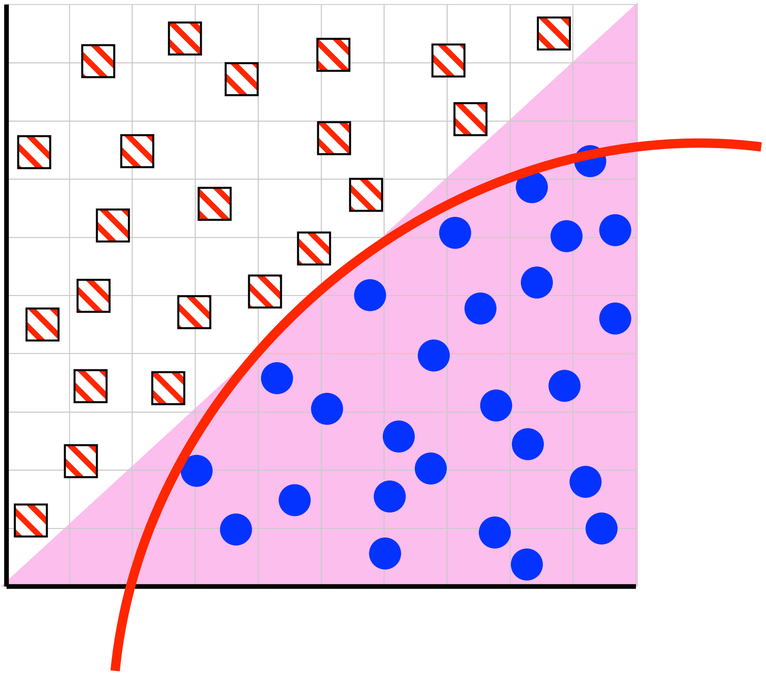


Pigeon Problem 1

Here is the rule again. If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

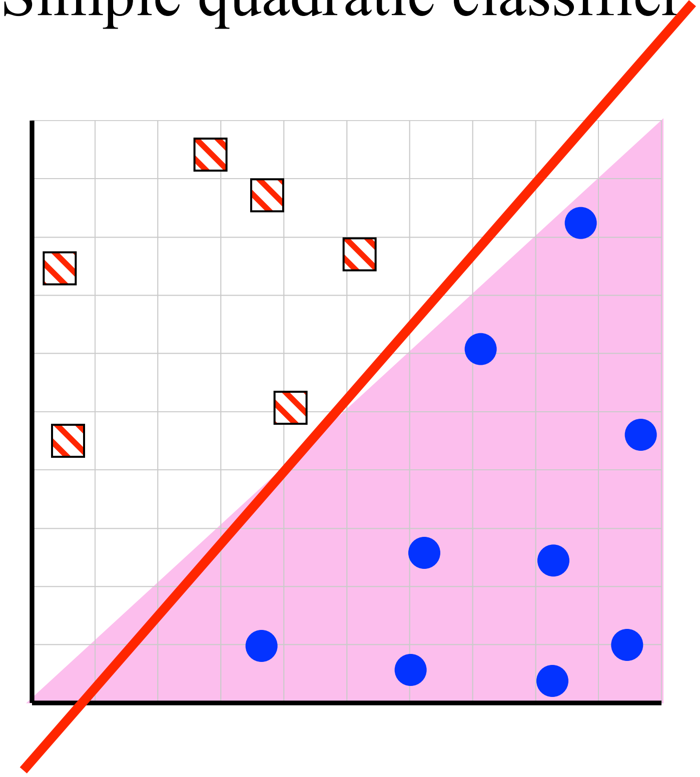
- Simple linear classifier gets 96% accuracy
- Simple quadratic classifier 97% accuracy

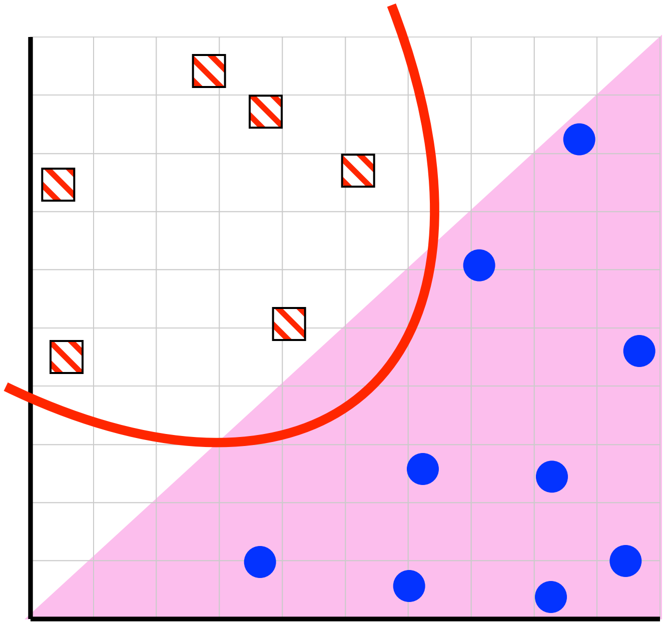




This problem is greatly exacerbated by having too little data

- Simple linear classifier gets 90% accuracy
- Simple quadratic classifier 95% accuracy





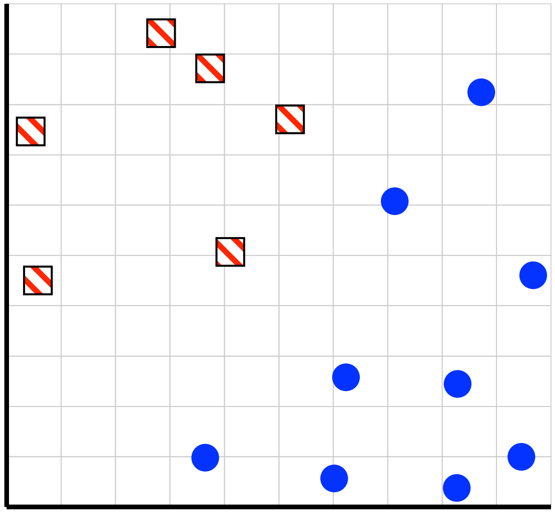
What happens as we have more and more training examples?

The accuracy for all models goes up!

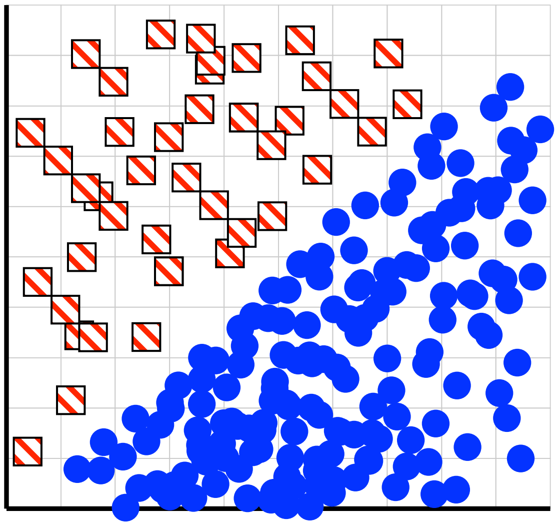
The chance of making a mistake goes down

The cost of the mistake (if made) goes down

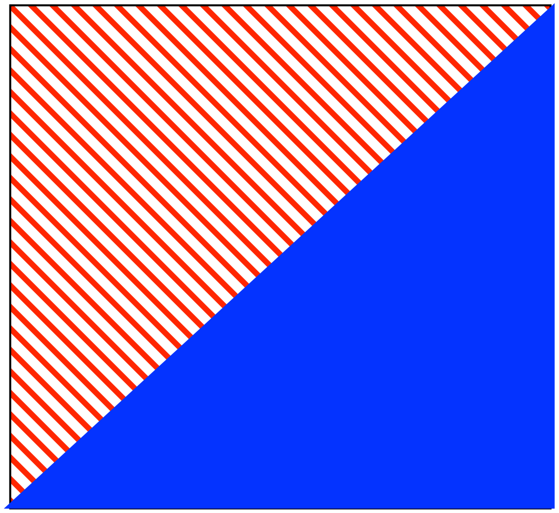
- Simple linear 70% accuracy
- Simple quadratic 90% accuracy



- Simple linear 90% accuracy
- Simple quadratic 95% accuracy



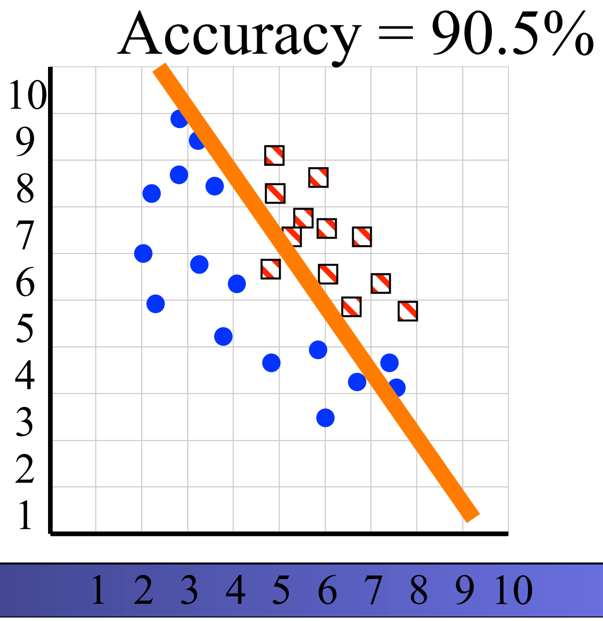
- Simple linear 99% accuracy
- Simple quadratic 99% accuracy

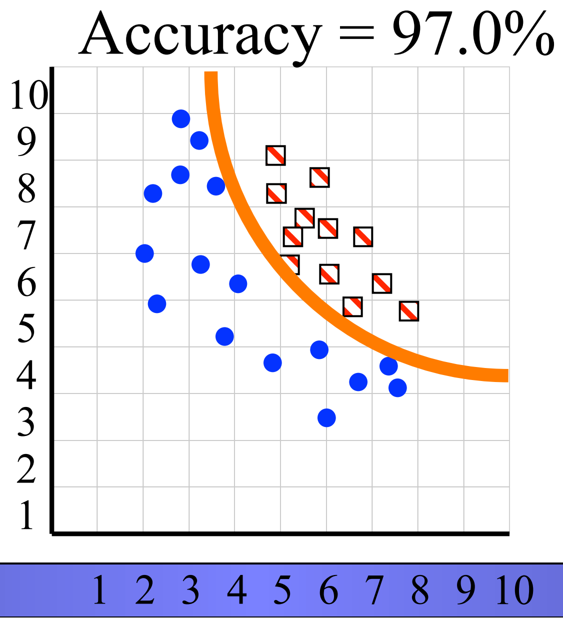


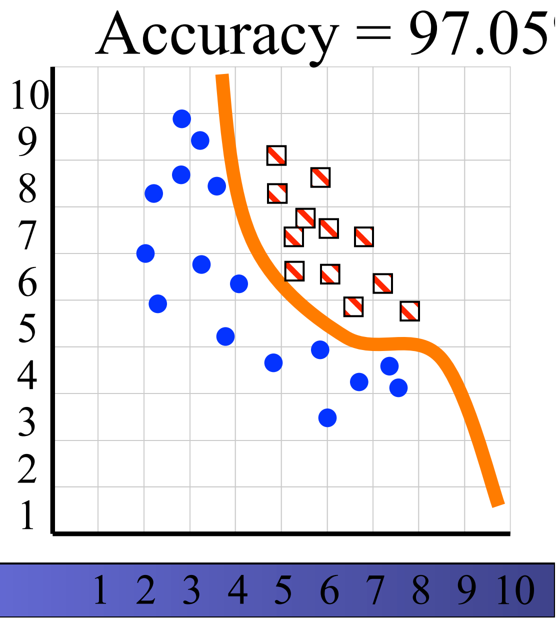
One Solution: Charge Penalty for complex models

- For example, for the simple {polynomial} classifier, we could charge 1% for every increase in the degree of the polynomial

  - Simple linear classifier gets 90.5% accuracy, minus 0, equals 90.5%
  - Simple quadratic classifier 97.0% accuracy, minus 1, equals 96.0%
  - Simple cubic classifier 97.05% accuracy, minus 2, equals 95.05%


Accuracy = 90.5%


Accuracy = 97.0%

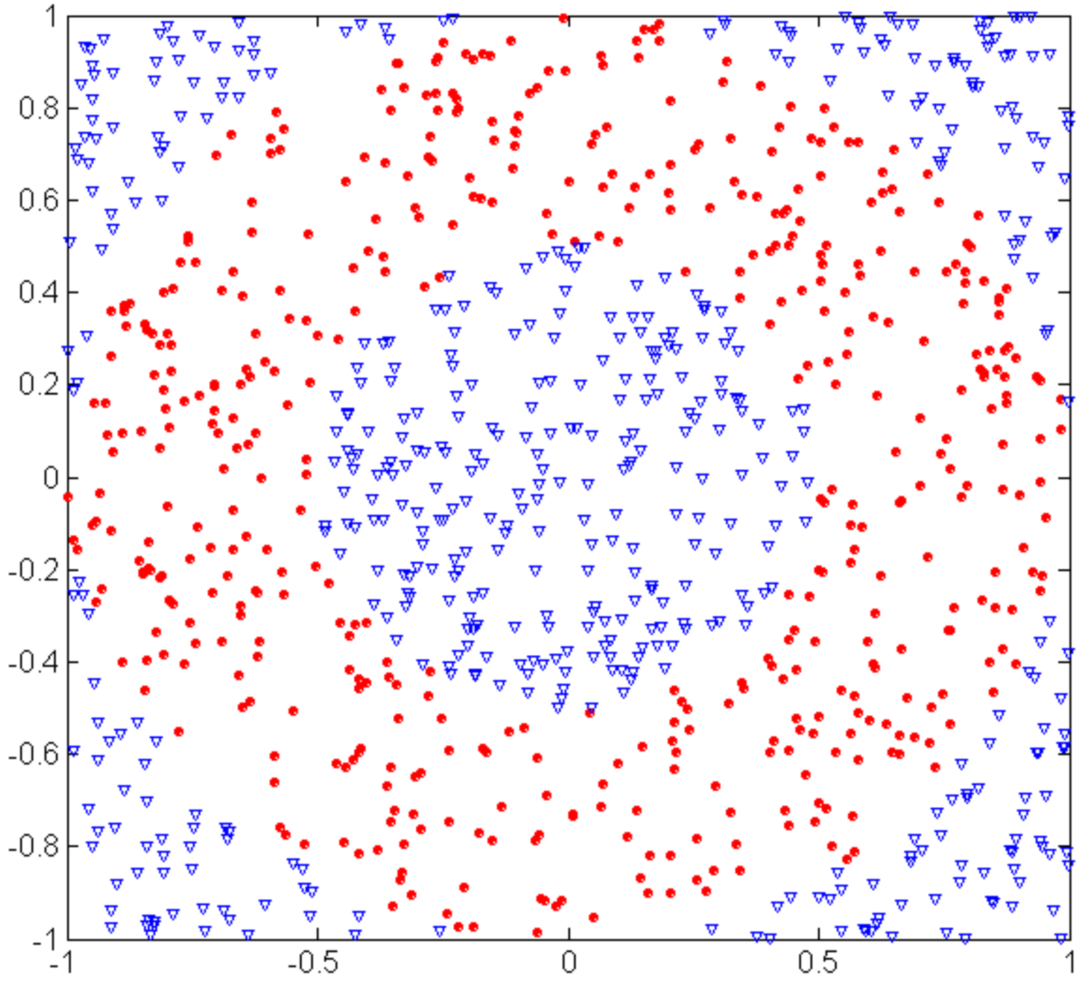

Accuracy = 97.05

One Solution: Charge Penalty for complex models

- For example, for the simple {polynomial} classifier, we could charge 1% for every increase in the degree of the polynomial.

- There are more principled ways to charge penalties
- In particular, there is a technique called **Minimum Description Length** (MDL)

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

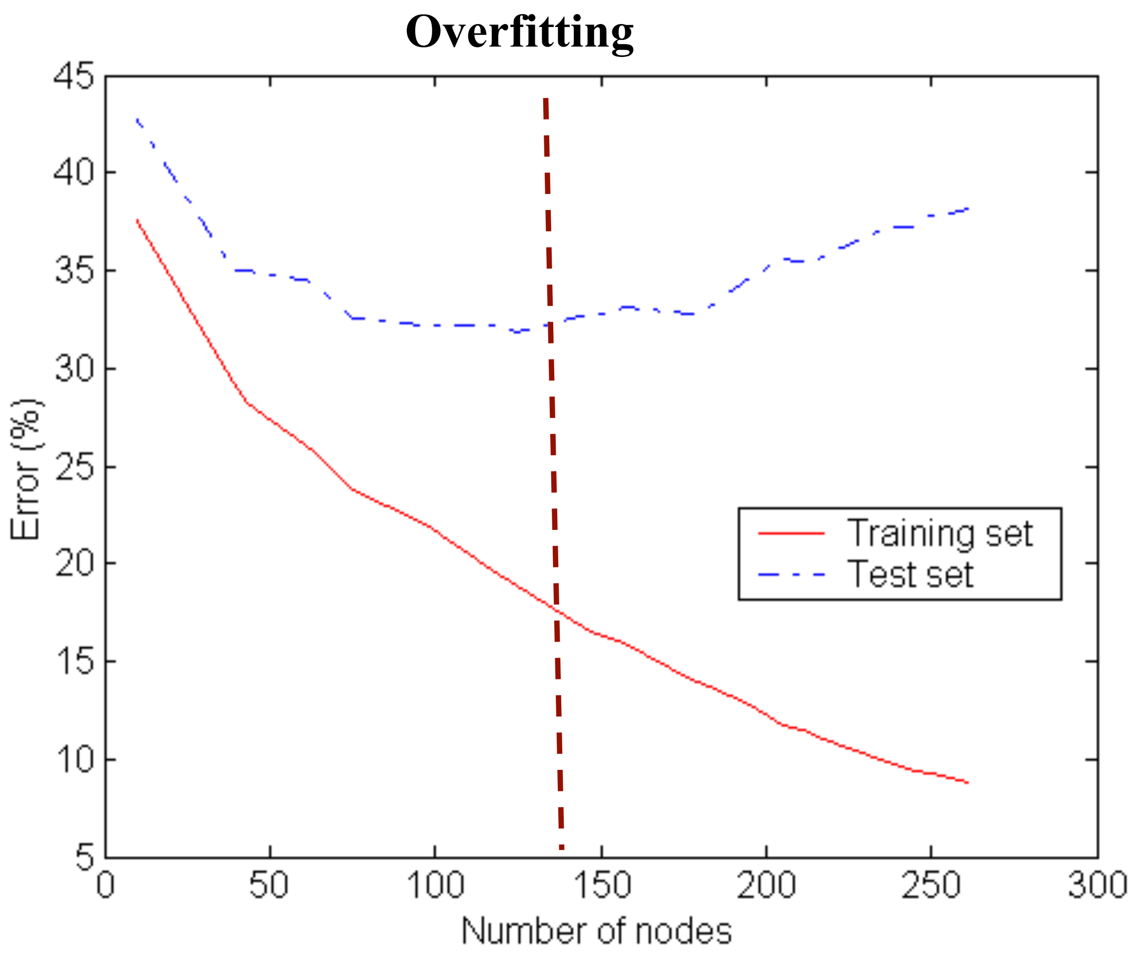
$0.5 \leq \mathrm{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\mathrm{sqrt}(x_1^2+x_2^2) > 0.5$ or

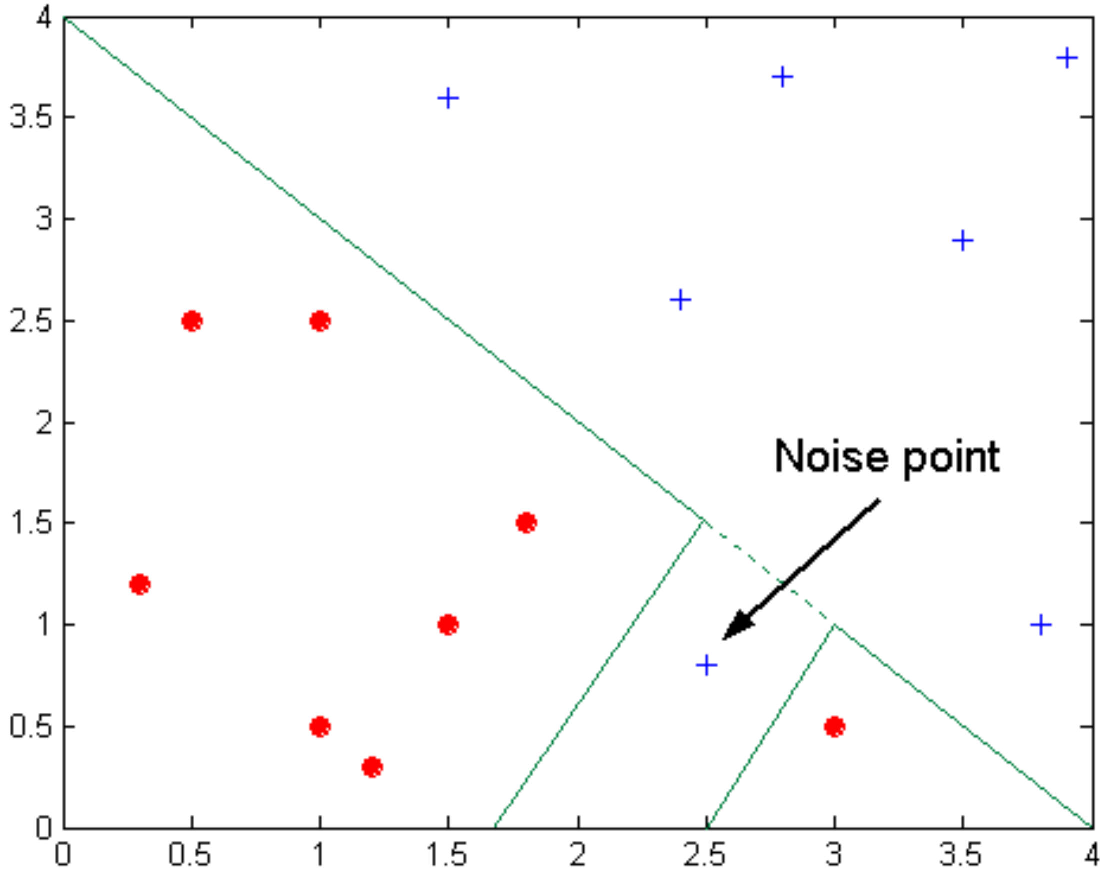
$\mathrm{sqrt}(x_1^2+x_2^2) < 1$

The Fitting Curve: Overfitting vs. Underfitting



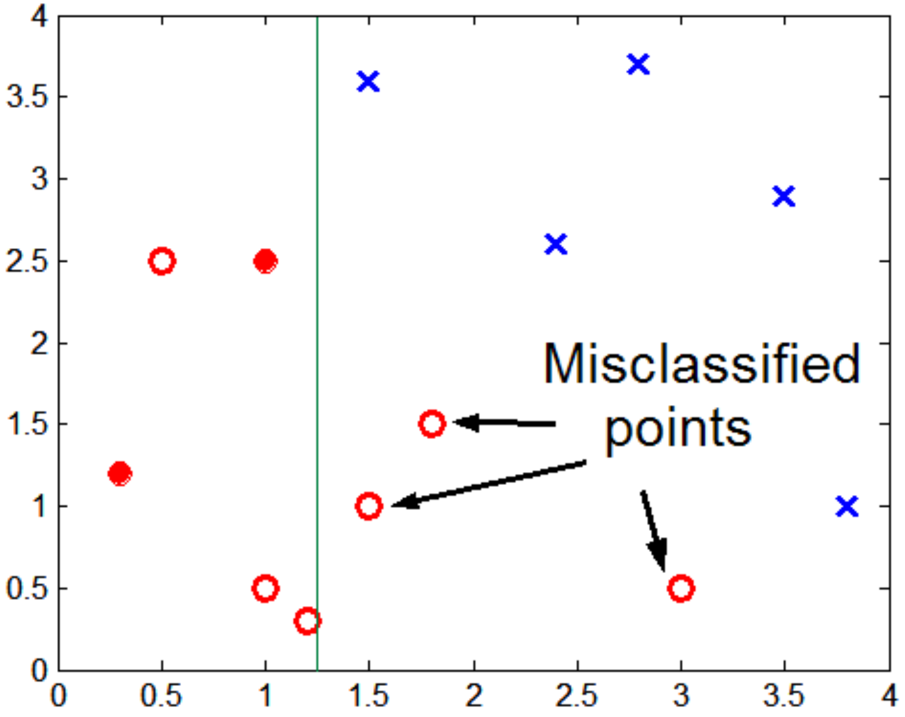
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Misclassified points

Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors

Estimating Generalization Errors

- Re-substitution errors: error on training (Σ e(t))
- Generalization errors: error on testing (Σ e' (t))

Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting…

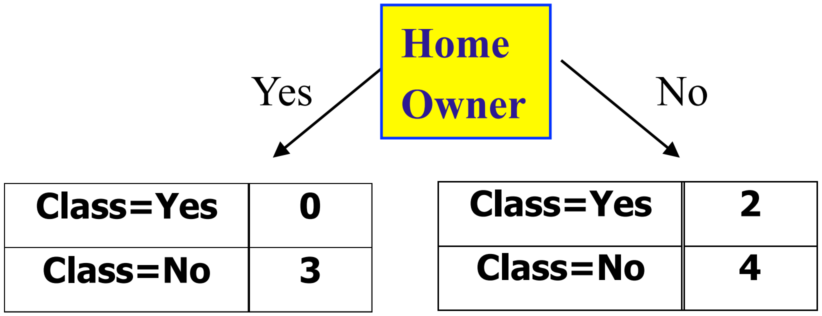
- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

Distribute Instances

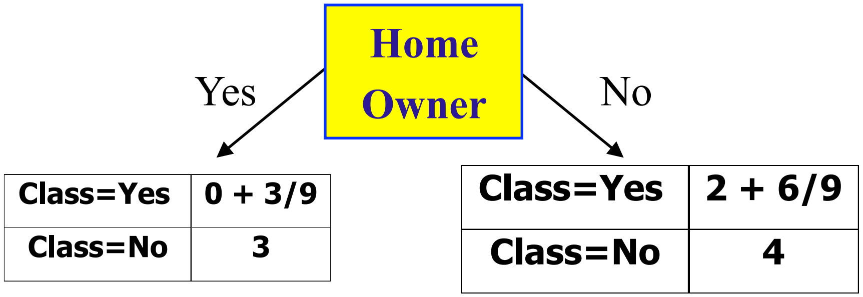
Tid	Home Owner	Marital Status	Annual Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No



Class=Yes	0
Class=No	3

Class=Yes	2
Class=No	4

Tid	Home Owner	Marital Status	Annual Income	Class
10	?	Single	90K	Yes



Class=Yes	0 + 3/9
Class=No	3

Class=Yes	2 + 6/9
Class=No	4

Probability that Home_Owner=Yes is 3/9

Probability that Home_Owner=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree

- Number of instances at the leaf nodes could be too small to make any statistically significant decision

Search Strategy

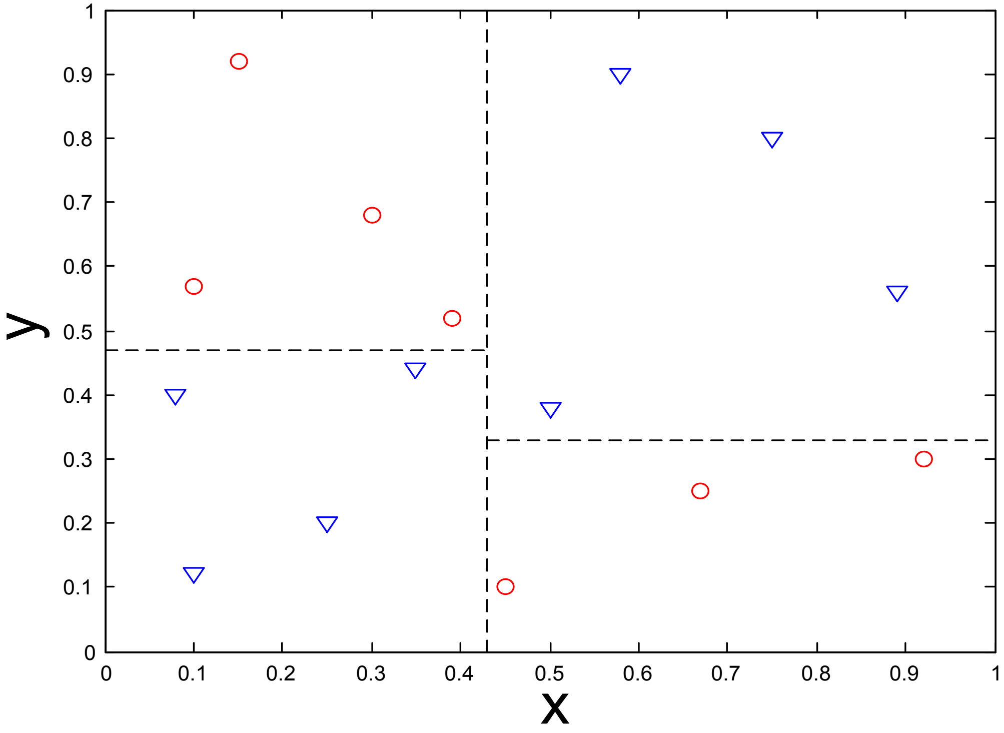
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

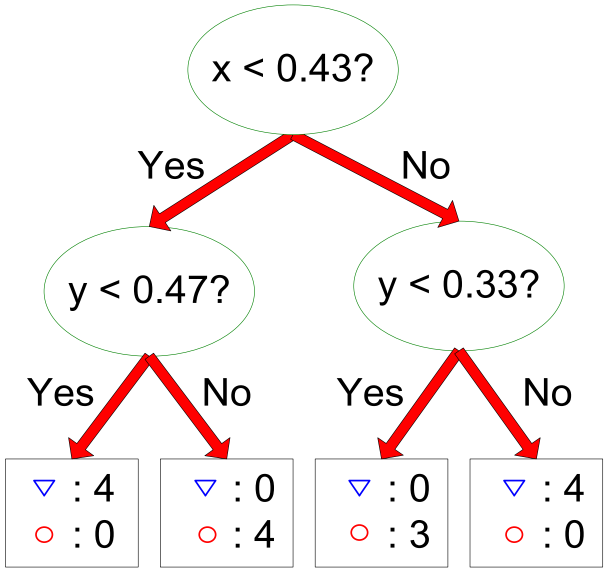
Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree

- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

Decision Boundary





x < 0.43?

Yes

No

y < 0.47?

y < 0.33?

Yes

No

Yes

No

▽ : 4
○ : 0

▽ : 0
○ : 4

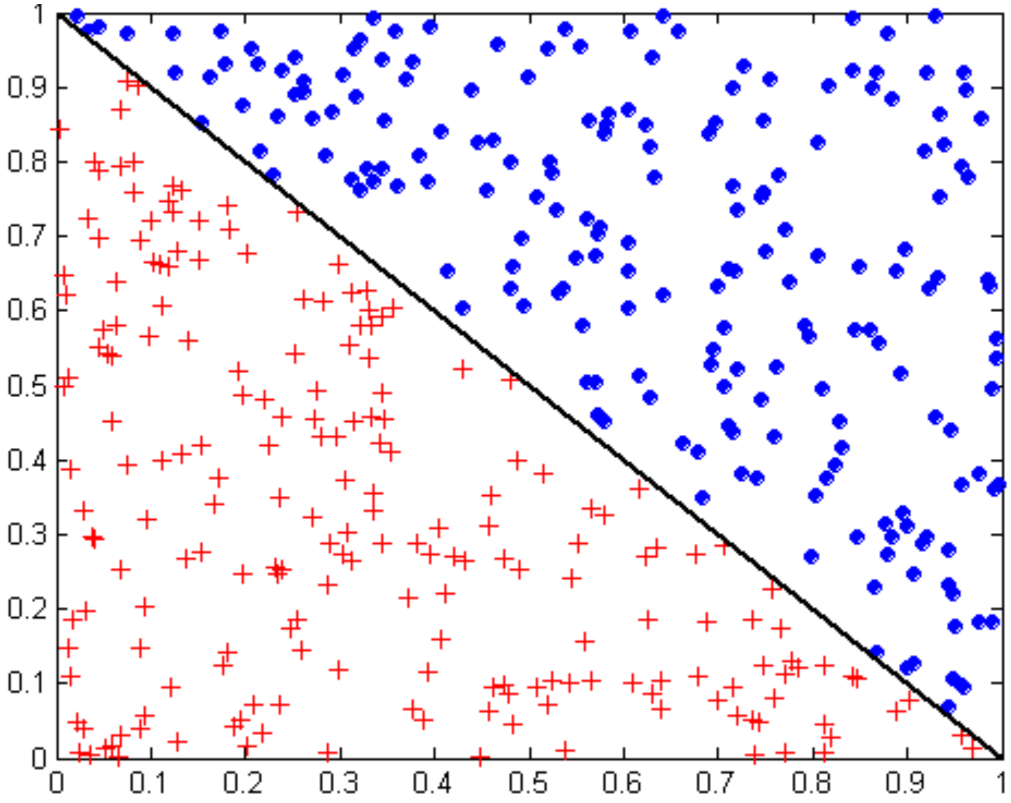
▽ : 0
○ : 3

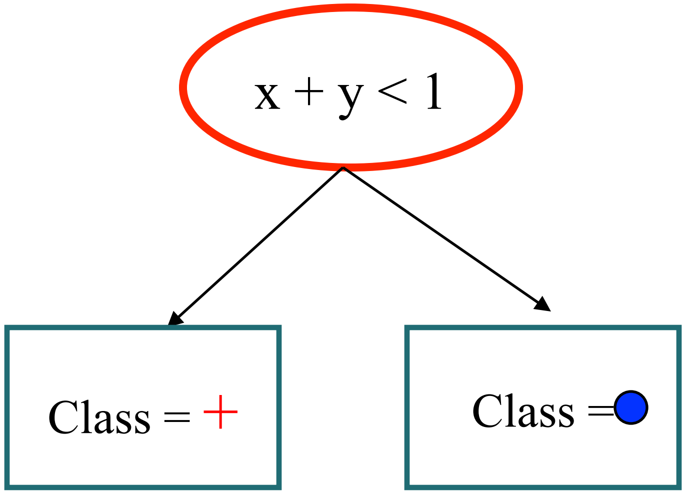
▽ : 4
○ : 0

- Border line between two neighboring regions of different classes is known as decision boundary

- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique Decision Trees





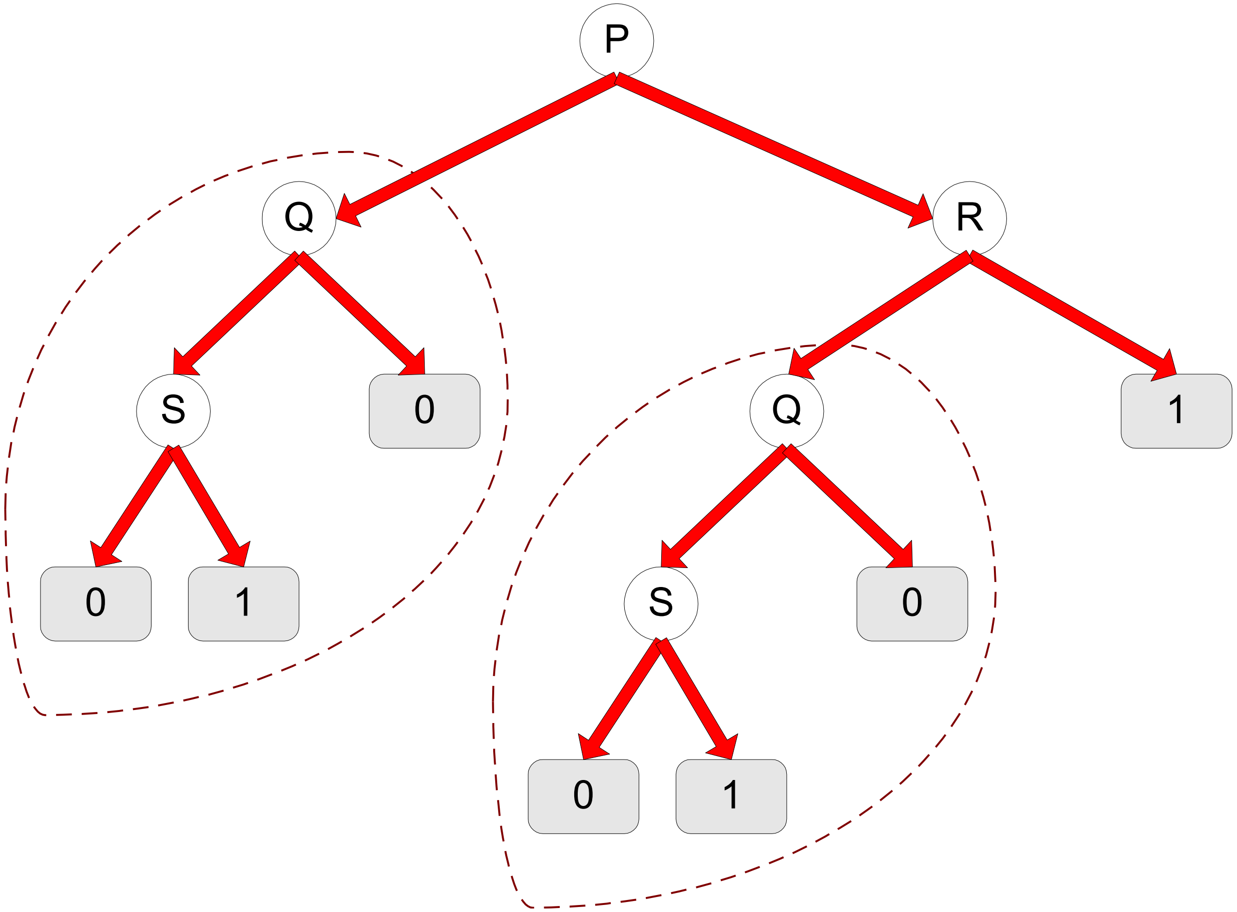
x + y < 1

Class = +

Class = ●

- **Test condition may involve multiple attributes**
- **More expressive representation**
- **Finding optimal test condition is computationally expensive**

Tree Replication



- **Same subtree appears in multiple branches**

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

Focus on the predictive capability of a model, rather than how fast it takes to classify or build models, scalability, etc.

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our database}}$$

Accuracy is a single number, we may be better off looking at a **confusion matrix**. This gives us additional useful information…

True label is...

Classified as a…

	Cat	Dog	Pig
Cat	100	0	0
Dog	9	90	1
Pig	45	45	10

Metrics for Performance Evaluation

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

(same as the previous equation, just rewriting it)

Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

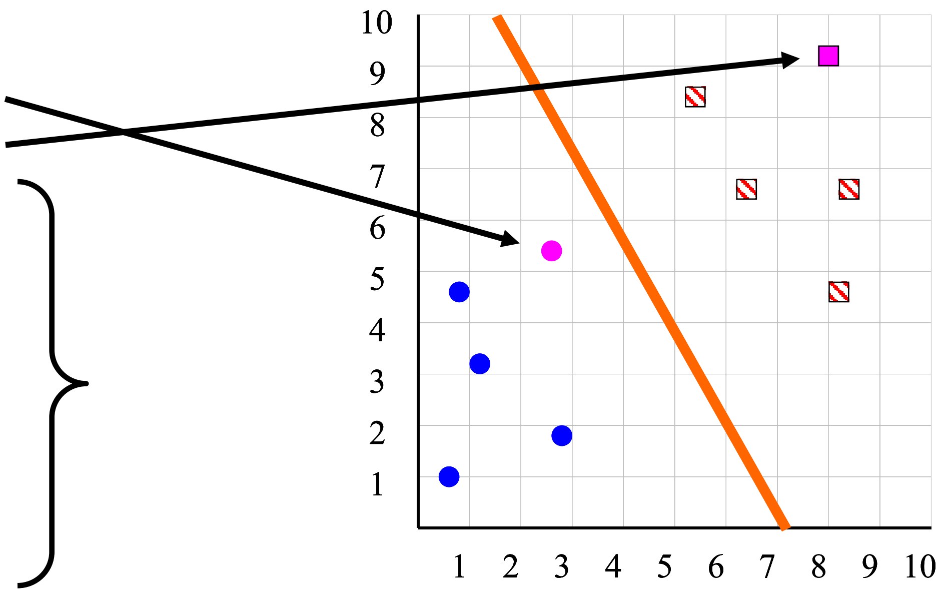
K-Fold Cross Validation

We divide the dataset into *K* equal sized sections. The algorithm is tested *K* times, each time leaving out one of the *K* section from building the classifier, but using it to *test* the classifier instead

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our database}}$$

K = 5

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	Grasshopper
2	8.0	9.1	Katydid
3	0.9	4.7	Grasshopper
4	1.1	3.1	Grasshopper
5	5.4	8.5	Katydid
6	2.9	1.9	Grasshopper
7	6.1	6.6	Katydid
8	0.5	1.0	Grasshopper
9	8.3	6.6	Katydid
10	8.1	4.7	Katydids



Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

<table>
<tr><td></td><td colspan="3">PREDICTED CLASS</td></tr>
<tr><td rowspan="3">ACTUAL CLASS</td><td>C(i|j)</td><td><b>Class=Yes</b></td><td><b>Class=No</b></td></tr>
<tr><td><b>Class=Yes</b></td><td>C(Yes|Yes)</td><td>C(No|Yes)</td></tr>
<tr><td><b>Class=No</b></td><td>C(Yes|No)</td><td>C(No|No)</td></tr>
</table>

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if

1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned} \text{Cost} &= p\,(a + d) + q\,(b + c) \\ &= p\,(a + d) + q\,(N - a - d) \\ &= q\,N - (q - p)(a + d) \\ &= N\,[q - (q - p) \times \text{Accuracy}] \end{aligned}$$

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$