DATA MINING
SUPERVISED LEARNING

Regression

Classification

- Decision Trees
  - Classifier Expressiveness
- Nearest Neighbor Classifier
- Support Vector Machines (SVM)
- Logistic Regression
- Naïve Bayes

Supervised learning

- In supervised learning, except for the feature variables that describe the data, we also have a target variable
- The goal is to learn a function (model) that can estimate/predict the value of the target variable given the features
  - We learn the function using a labeled training set.
- Regression: The target variable (but also the features) is numerical and continuous
  - The price of a stock, the GDP of a country, the grade in a class, the height of a child, the life expectancy etc
- Classification: The target variable is discrete
  - Does a taxpayer cheat or not? Will the stock go up or down? Will the student pass or fail? Is a transaction fraudulent or not? What is the topic of an article?

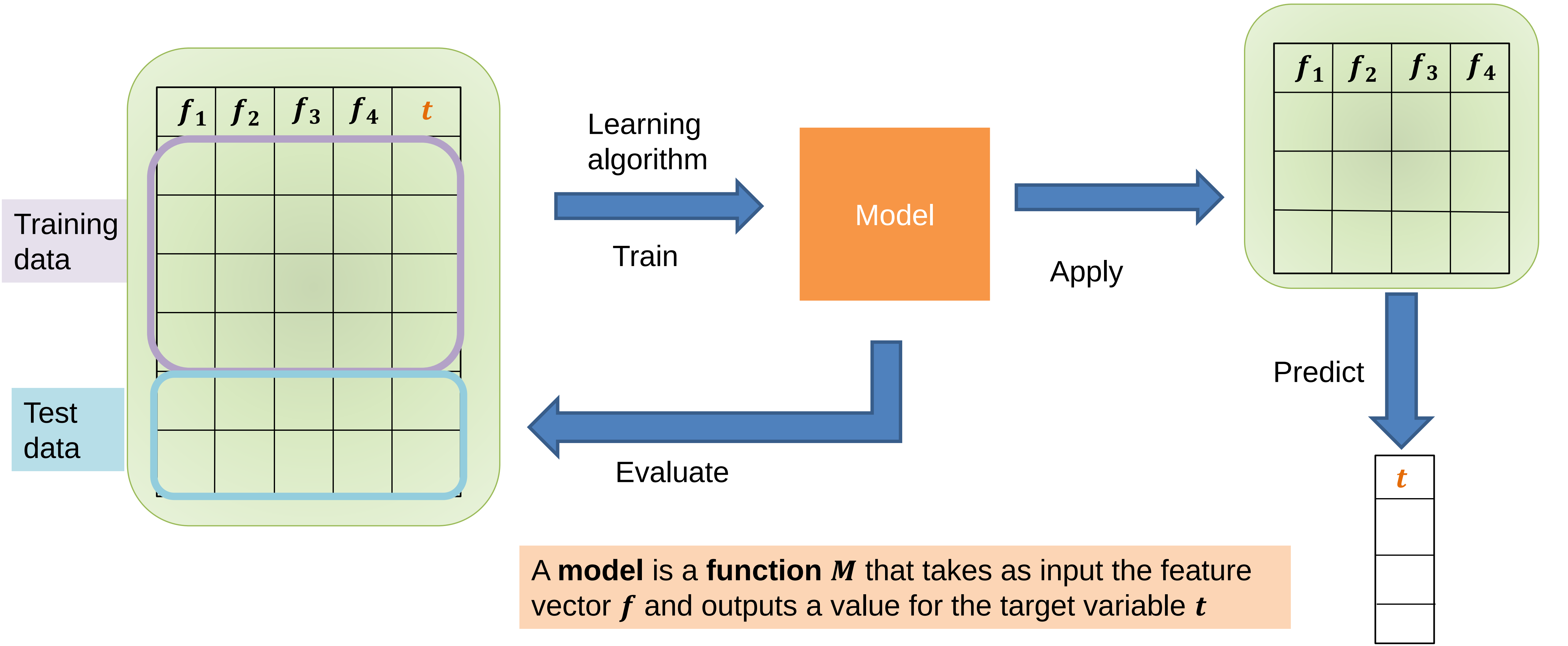
Applications

- Descriptive modeling: Explanatory tool to understand the data:
  - Regression: How does the change in the value of different factors affect our target variable?
    - What factors contribute to the price of a stock?
    - What factors contribute to the GDP of a country?
  - Classification: Understand what attributes distinguish between objects of different classes
    - Why people cheat on their taxes?
    - What words make an post offensive?

- Predictive modeling: Predict a class of a previously unseen record
  - Regression: What will the life-expectancy of a patient be?
  - Classification: Is this a cheater or not? Will the stock go up or not. Is this an offensive post?

- Predictive modeling is in the heart of the data science revolution.

Supervised Learning Overview



A **model** is a **function** $\boldsymbol{M}$ that takes as input the feature vector $\boldsymbol{f}$ and outputs a value for the target variable $\boldsymbol{t}$

LINEAR REGRESSION

Regression

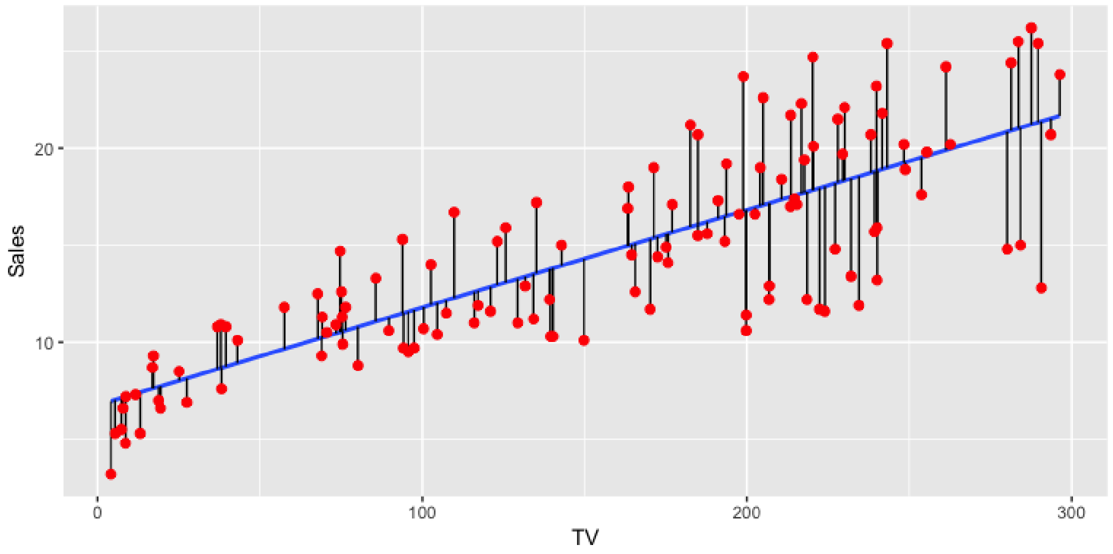
- We assume that we have k feature variables (numeric):
  - Also known as covariates, or independent variables
- The target variable is also known as dependent variable.
- We are given a dataset of the form $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ where, $\boldsymbol{x}_i$ is a k-dimensional feature vector, and y_i a real value
- We want to learn a function f which given a feature vector $\boldsymbol{x}_i$ predicts a value $y_i' = f(\boldsymbol{x}_i)$ that is as close as possible to the value y_i
- Minimize sum of squares:

$$\sum_i \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

Linear regression

- The simplest form of f is a linear function
- In linear regression the function f is typically of the form:

$$f(\boldsymbol{x_i}) = w_0 + \sum_{j=1}^{k} w_j x_{ij}$$



One-dimensional linear regression

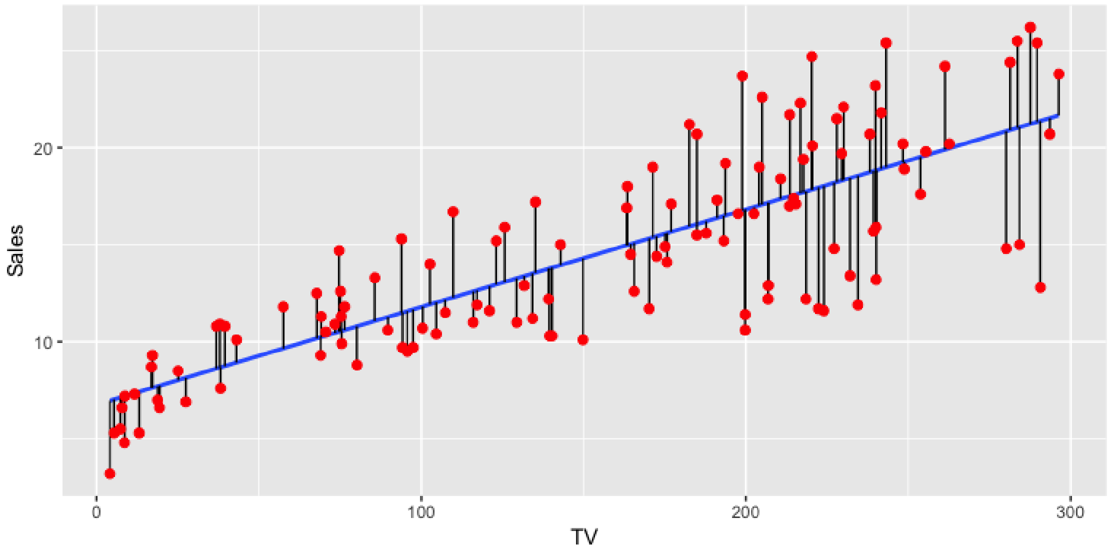
In the simplest case we have a single variable and the function is of the form:

$$f(x_i) = w_0 + w_1 x_i$$

Minimizing the error gives:

$$w_0 = \bar{y} - w_1 \bar{x}$$

$$w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = r_{xy} \frac{\sigma_y}{\sigma_x}$$



$\bar{x}$: mean value of x_i's
$\bar{y}$: mean value of y_i's
r_{xy}: correlation coefficient between $\boldsymbol{x}, \boldsymbol{y}$

Multiple linear regression

- In the general case we have k features, and $\boldsymbol{x_i}, \boldsymbol{w}$ are vectors.
- We simplify the notation:

$$\boldsymbol{x_i} = (1, x_{i1}, \dots, x_{ik})$$
$$\boldsymbol{w} = (w_0, w_1, \dots, w_k)$$
$$f(\boldsymbol{x_i}, \boldsymbol{w}) = \boldsymbol{x_i^T w}$$

- Let X be the $n \times (k+1)$ matrix with vectors $\boldsymbol{x_i}$ as rows.
- Let $\boldsymbol{y} = (y_1, \dots, y_n)$
- We can write the SSE function as:

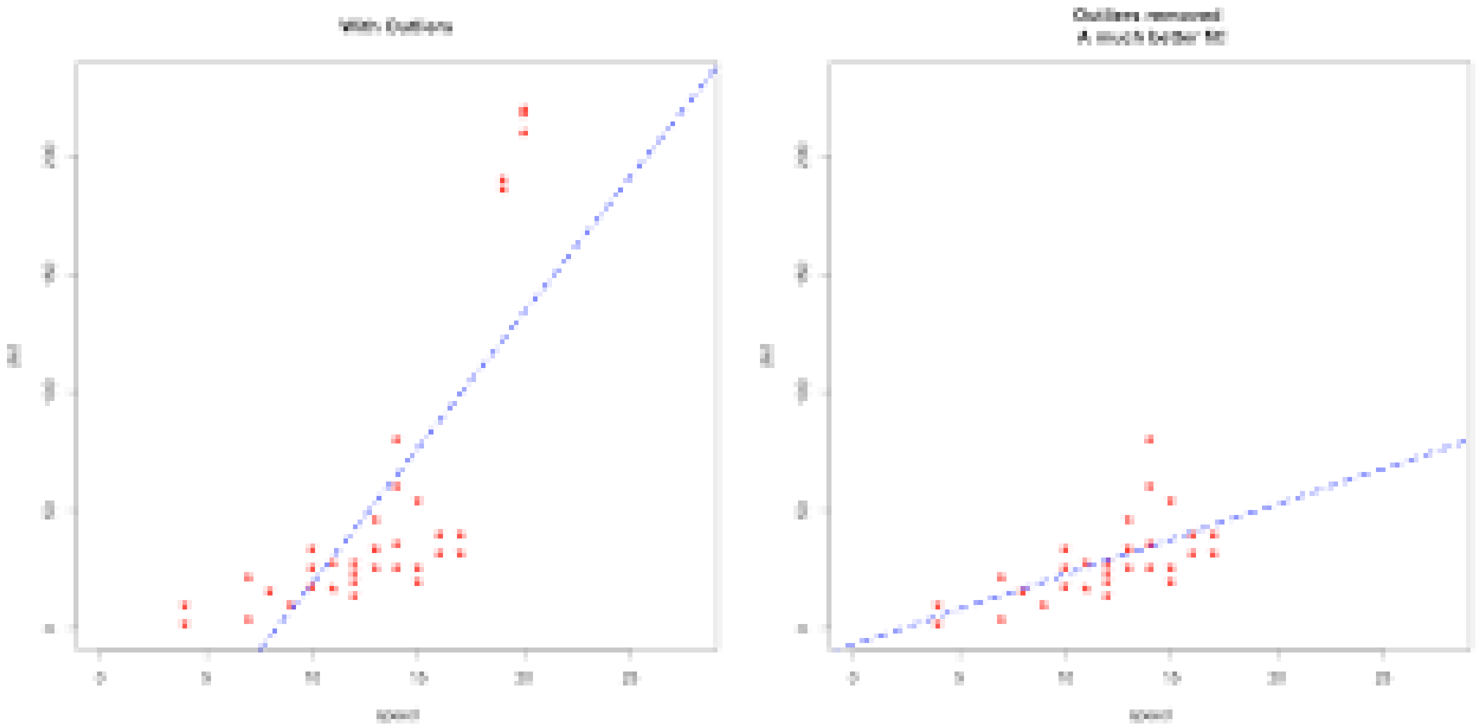
$$SSE = \|X\boldsymbol{w} - \boldsymbol{y}\|^2$$

- There is a closed-form solution for $\boldsymbol{w}$:

$$\boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{y}$$

- Matrix inversion may be too expensive. Other optimization techniques are often used to find the optimal vector (e.g., Gradient Descent)

Outliers



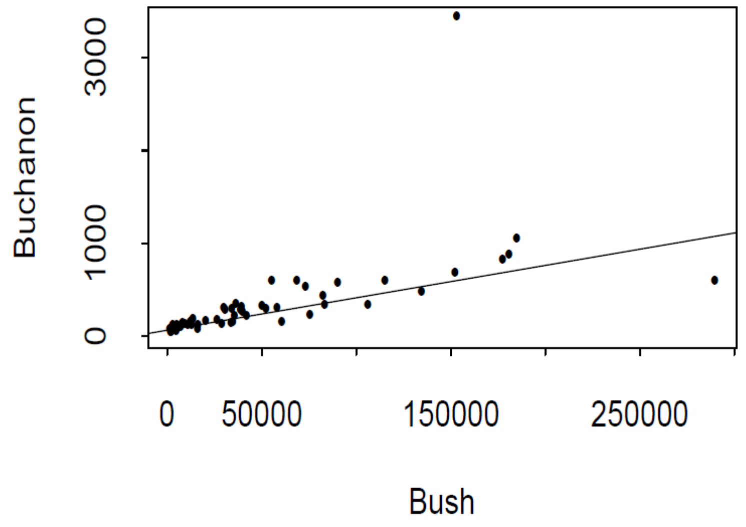
- Regression is sensitive to outliers:
  - The line will “tilt” to accommodate very extreme values
- Solution: remove the outliers
  - But make sure that they do not capture useful information

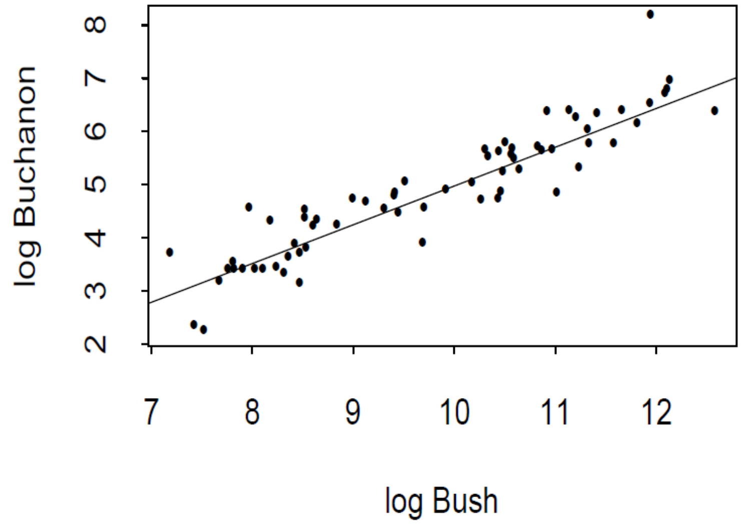
Normalization

- In the regression problem some times our features may have very different scales:
  - For example: predict the GDP of a country using as features the percentage of home owners and the income
  - The weights in this case will not be interpretable
- Solution: Normalize the features by replacing the values with the z-scores
  - Remove the mean and divide by the standard deviation

More complex models

- The model we have is linear with respect to the parameters $\boldsymbol{w}$ but the features we consider may be non-linear functions of the $\boldsymbol{x_i}$ values.
- To capture more complex relationships, we can take a transformation of the input (e.g., logarithm $\log x_{ij}$), or add polynomial terms (e.g., x_{ij}^2).
  - For example, we can learn a function of the form $f(x) = w_0 + w_1 x + w_2 x^2$
  - However this may increase a lot the number of features





Interpretation and significance

- A regression model is useful for making predictions for new data.
- The coefficients for the linear regression model are also useful for understanding the effect of the independent variables to the value of the dependent variable
  - The w_j value is the effect of the increase of x_{ij} by one to the value y_i
- We can also compute the significance of the value of w_j by testing the null hypothesis that $w_j = 0$

Covariate	Least Squares Estimate	Estimated Standard Error	t value	p-value	
(Intercept)	*-589.39*	*167.59*	*-3.51*	*0.001*	**
Age	*1.04*	*0.45*	*2.33*	*0.025*	*
Southern State	*11.29*	*13.24*	*0.85*	*0.399*	
Education	*1.18*	*0.68*	*1.7*	*0.093*	
Expenditures	*0.96*	*0.25*	*3.86*	*0.000*	***
Labor	*0.11*	*0.15*	*0.69*	*0.493*	
Number of Males	*0.30*	*0.22*	*1.36*	*0.181*	
Population	*0.09*	*0.14*	*0.65*	*0.518*	
Unemployment (14-24)	*-0.68*	*0.48*	*-1.4*	*0.165*	
Unemployment (25-39)	*2.15*	*0.95*	*2.26*	*0.030*	*
Wealth	*-0.08*	*0.09*	*-0.91*	*0.367*	

This table is typical of the output of a multiple regression program. The "t-value" is the Wald test statistic for testing $H_0 : \beta_j = 0$ *versus* $H_1 : \beta_j \neq 0$. *The asterisks denote "degree of significance" with more asterisks being significant at a smaller level. The example raises several important questions. In particular: (1) should we eliminate some variables from this model? (2) should we interpret this relationships as causal? For example, should we conclude that low crime prevention expenditures cause high crime rates? We will address question (1) in the next section. We will not address question (2) until a later Chapter.*

Predicting Crime rate

CLASSIFICATION

Classification

- Similar to the regression problem we have features and a target variable that we want to model/predict
- The target variable is now discrete. It is often called the class label
  - In the simplest case, it is a binary variable.
- In classification the features may also be categorical.

Example: Catching tax-evasion

Tid	Refund	Marital Status	Taxable Income	Cheat
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
10	No	Single	90K	Yes

Tax-return data for year 2011

A new tax return for 2012
Is this a cheating tax return?

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

An instance of the classification problem: learn a method for discriminating between records of different classes (cheaters vs non-cheaters)

Classification

- **Classification** is the task of ***learning a target function* f** that maps attribute set **x** to one of the predefined class labels **y**
- The function may be defined as an algorithm (e.g., if Single and Income < 125K then No)

categorical categorical continuous class

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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**
10	No	Single	90K	**Yes**

One of the attributes is the class attribute

In this case: Cheat

Two class labels (or classes): Yes (1), No (0)

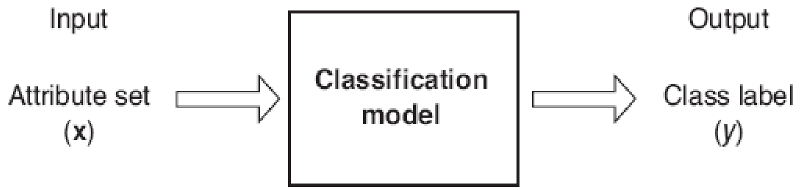


Figure 4.2. Classification as the task of mapping an input attribute set $\mathbf{x}$ into its class label y.

Examples of Classification Tasks

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports
- Identifying spam email, spam web pages, adult content
- Understanding if a web query has commercial intent or not

Classification is everywhere in data science
Big data has the answers to all questions.

General approach to classification

- Obtain a training set consisting of records with known class labels
- Training set is used to build a classification model
- A labeled test set of previously unseen data records is used to evaluate the quality of the model.
- The classification model is applied to new records with unknown class labels
- Important intermediate step: Decide on what features to use

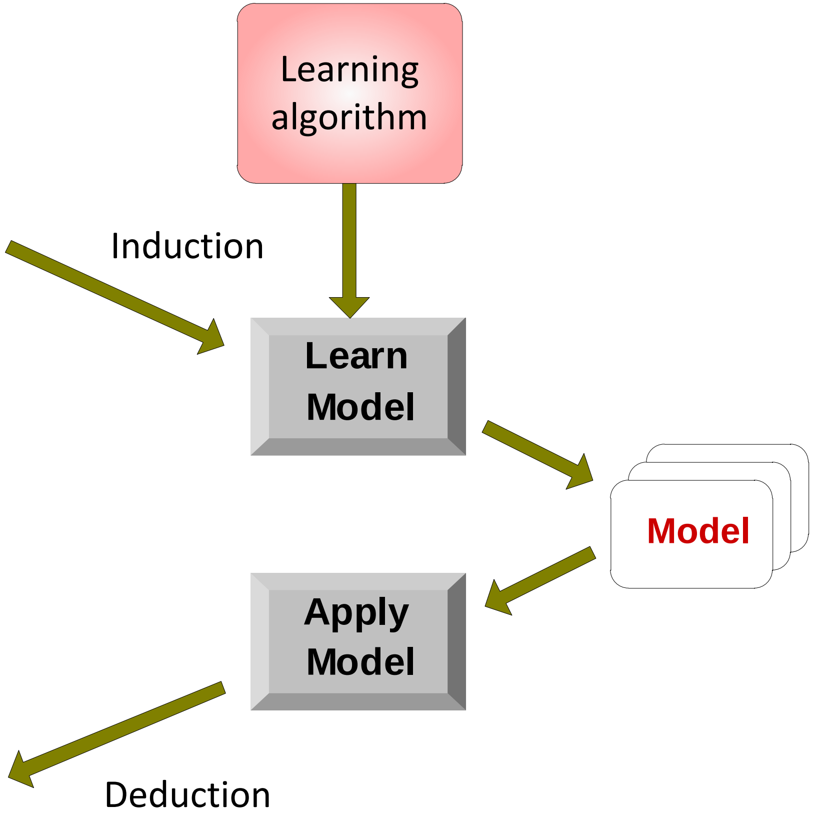
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Evaluation of classification models

- Counts of test records that are correctly (or incorrectly) predicted by the classification model
- **Confusion matrix**

Actual Class	Predicted Class: Class = 1	Predicted Class: Class = 0
Class = 1	f_{11}	f_{10}
Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{\#\,\text{correct predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\#\,\text{wrong predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Logistic Regression

DECISION TREES

Decision Trees

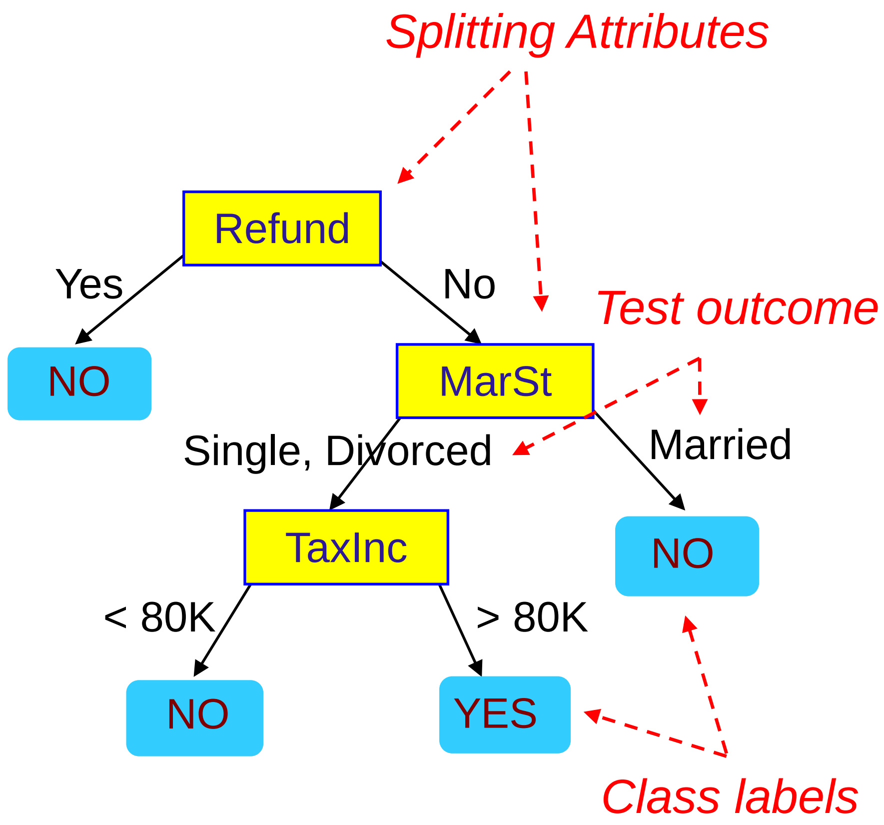
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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
10	No	Single	90K	Yes

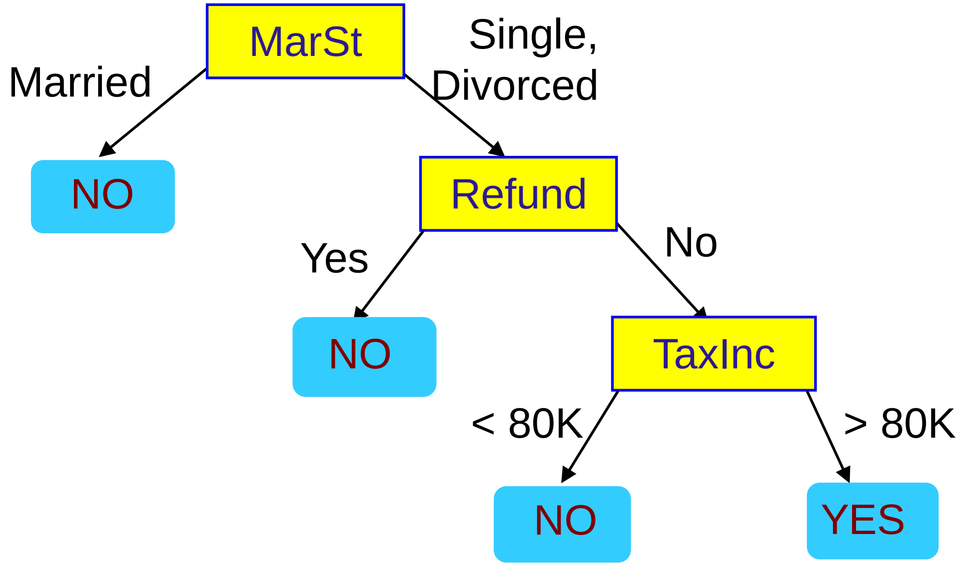
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (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
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

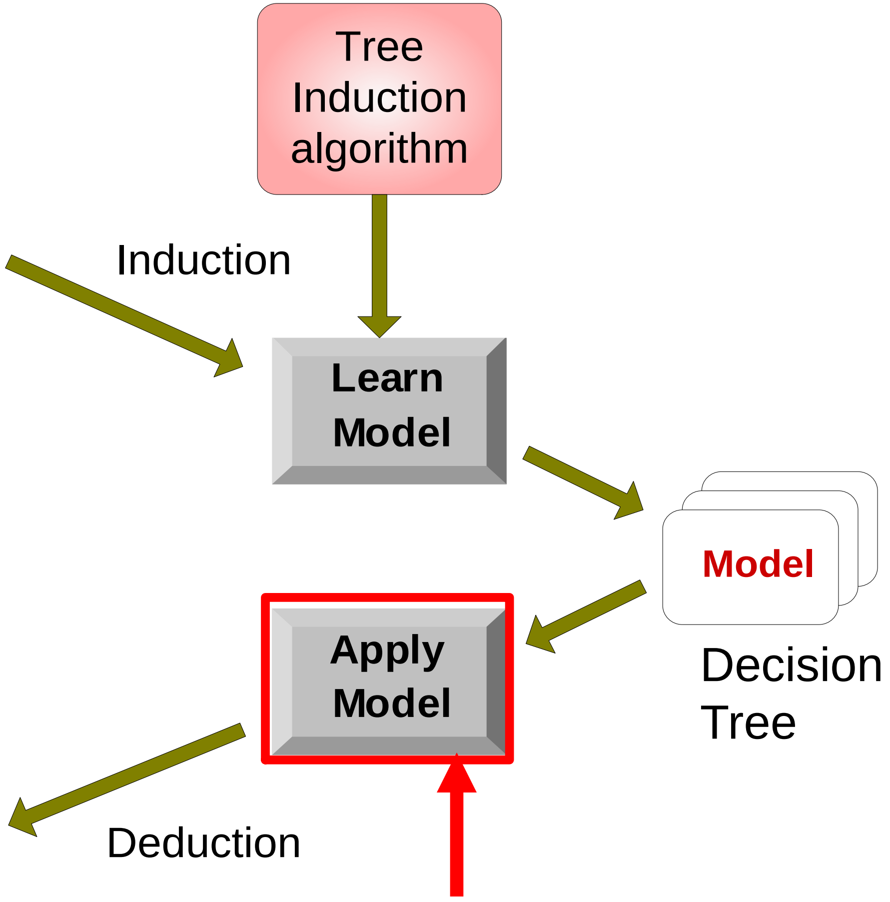
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

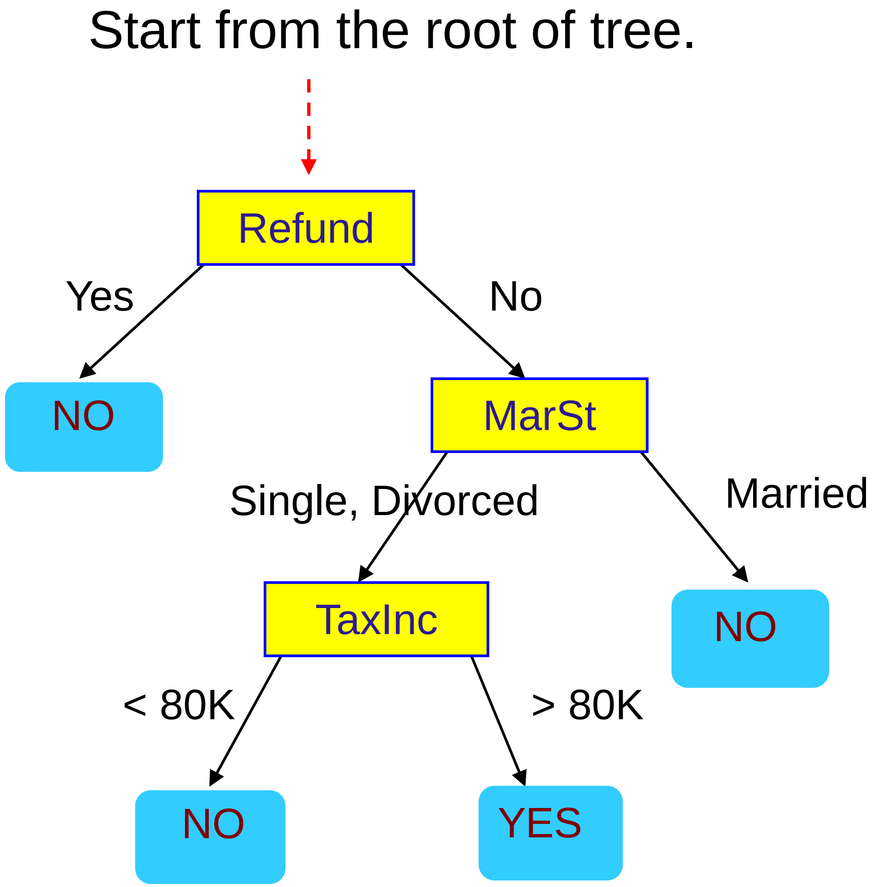
Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



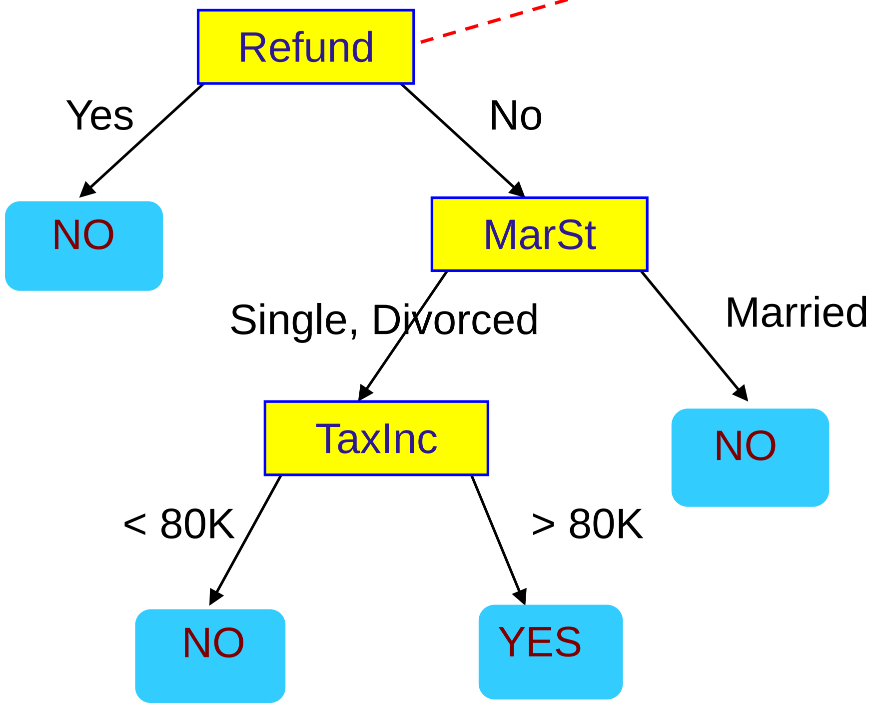
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



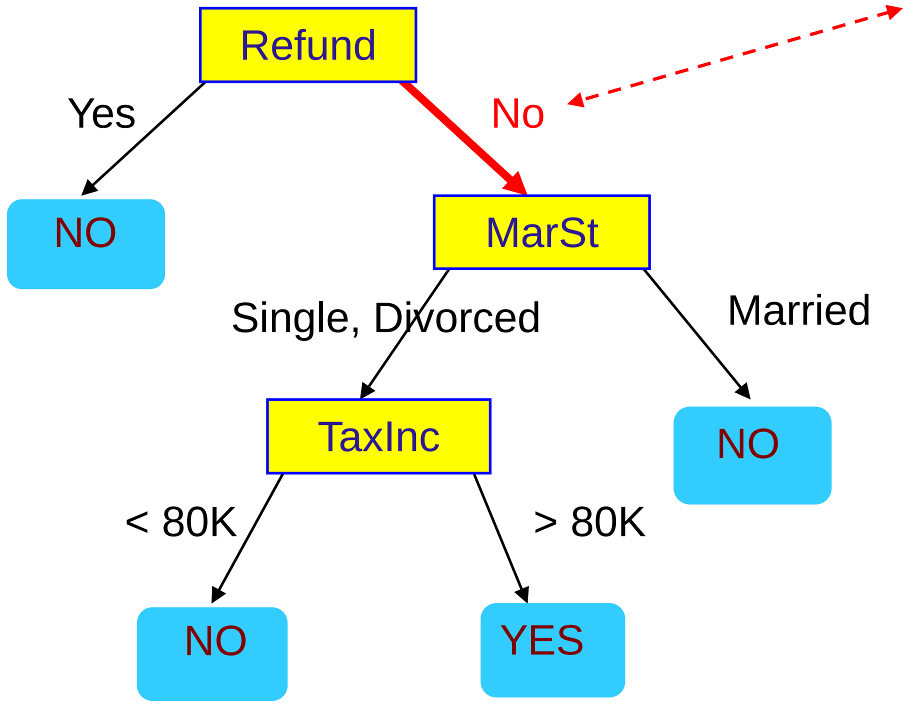
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

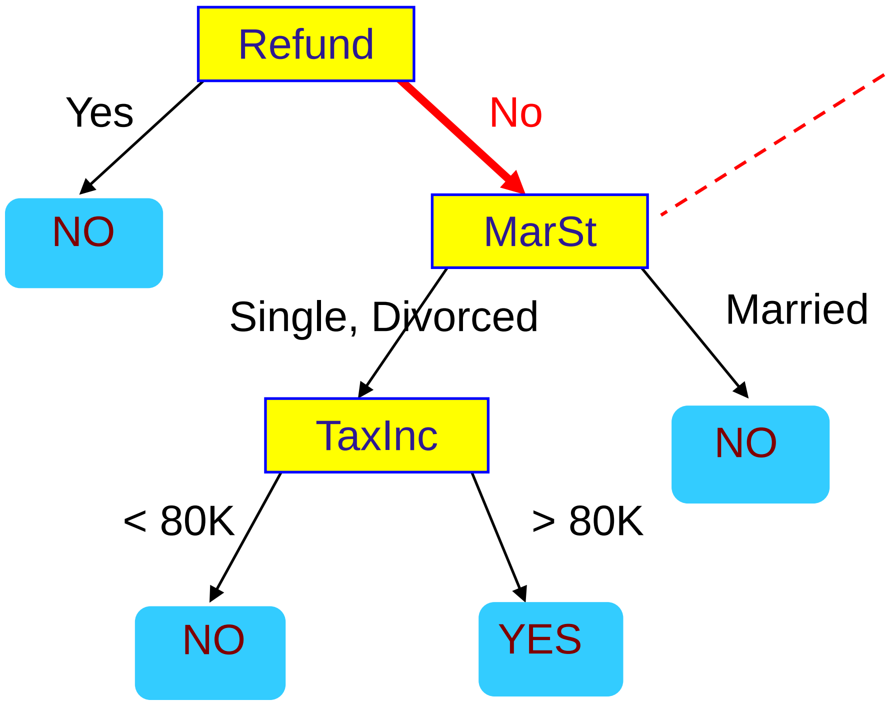
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



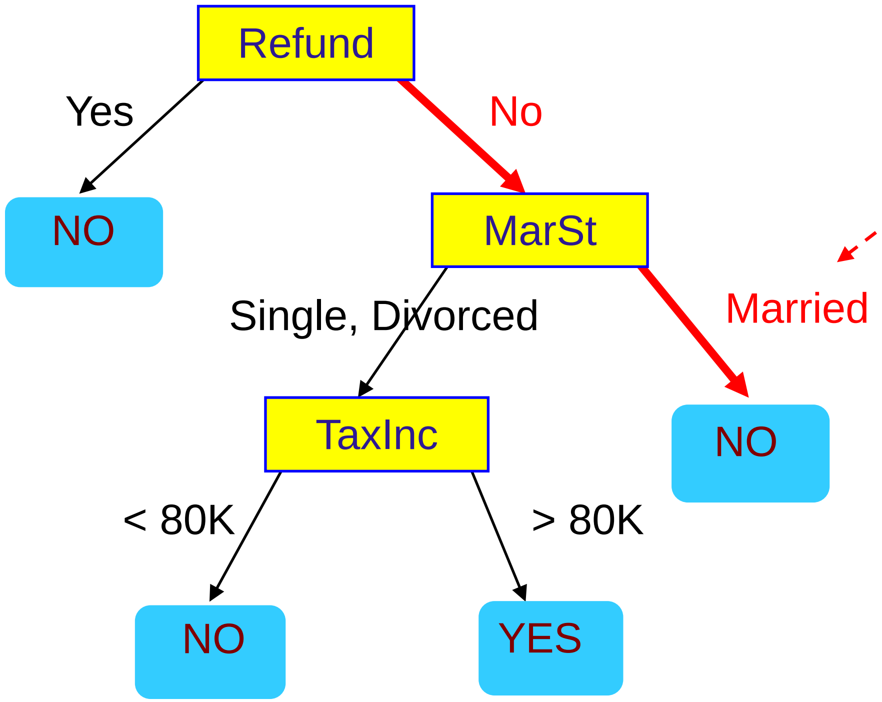
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

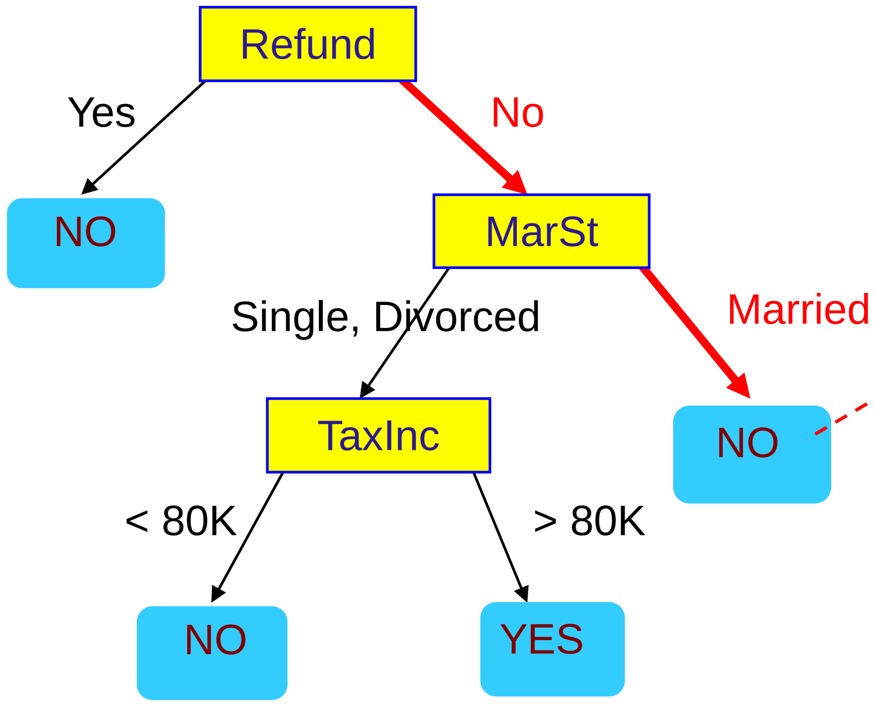
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

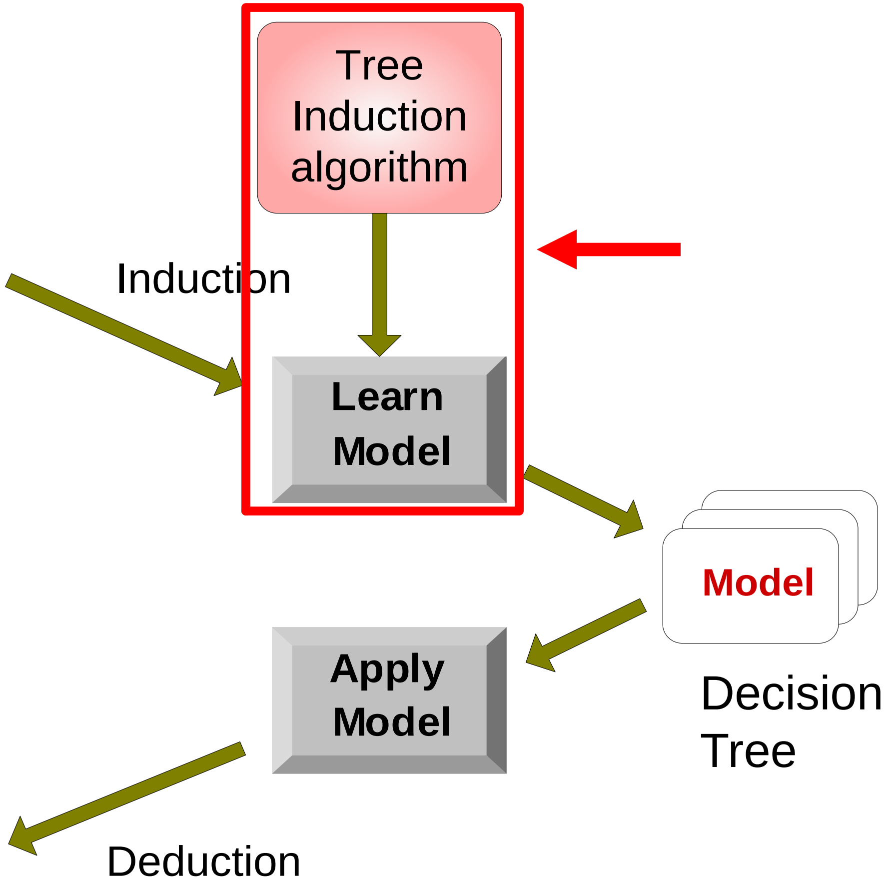
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

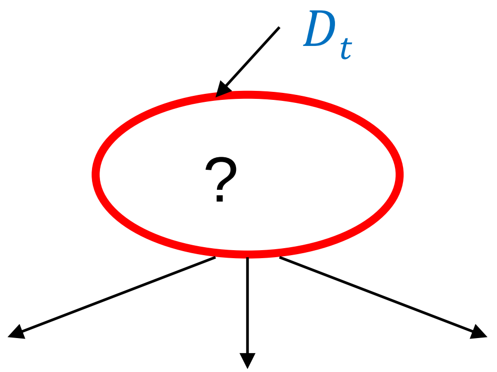
Tree Induction

- Goal: Find the tree that has low classification error in the training data (training error)
- Finding the best decision tree (lowest training error) is NP-hard
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

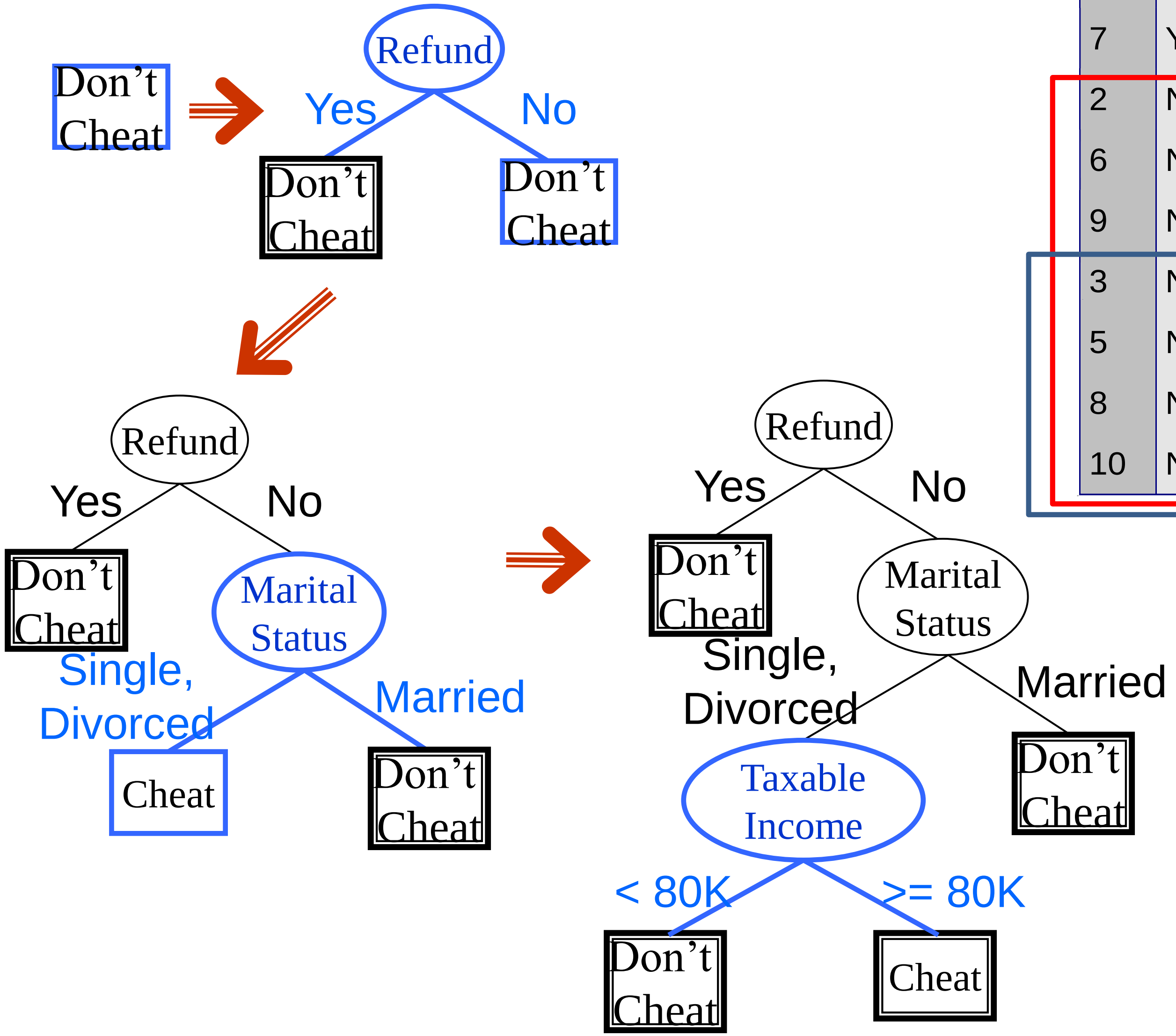
General Structure of Hunt's Algorithm

- D_t : the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records with the same attribute values, then t is a leaf node labeled with the majority class y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.
- Recursively apply the procedure to each subset.

Tid	Refund	Marital Status	Taxable Income	Cheat
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
10	No	Single	90K	Yes



Hunt's Algorithm



Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No
2	No	Married	100K	No
6	No	Married	60K	No
9	No	Married	75K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

Constructing decision-trees (pseudocode)

GenDecTree(Sample **S**, Features **F**)

1. **If stopping_condition(S,F)** = true **then**
   a. **leaf = createNode()**
   b. **leaf.label= Classify(S)**
   c. **return leaf**
2. **root = createNode()**
3. **root.test_condition = findBestSplit(S,F)**
4. **V = {v| v** a possible outcome of **root.test_condition}**
5. **for *each*** value **vϵV**:
   a. **S_v: = {s | root.test_condition(s)** = **v** and **s ϵ S}**;
   b. **child = GenDecTree(S_v ,F**) ;
   c. Add **child** as a descent of **root** and label the edge **(root→child) as v**
6. **return root**

Tree Induction

- Issues
  - How to Classify a leaf node
    - Assign the majority class
    - If leaf is empty, assign the default class – the class that has the highest popularity (overall or in the parent node).
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

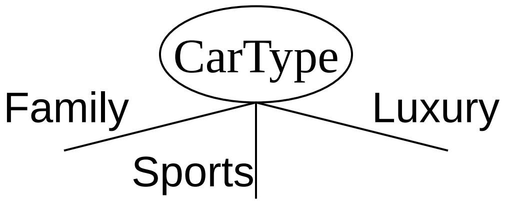
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

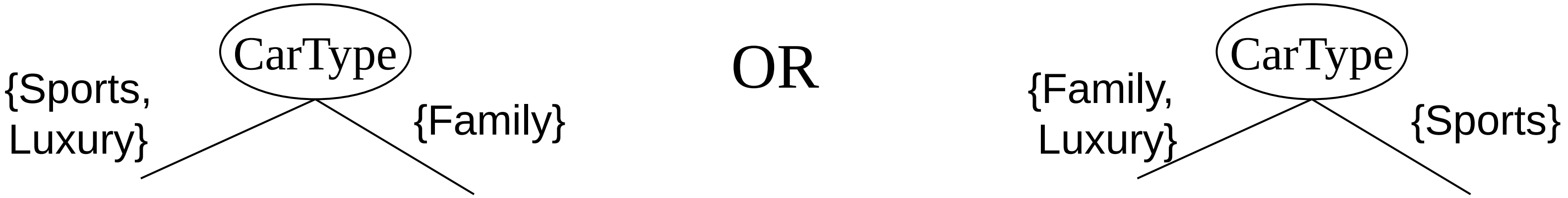
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

- Multi-way split: Use as many partitions as distinct values.

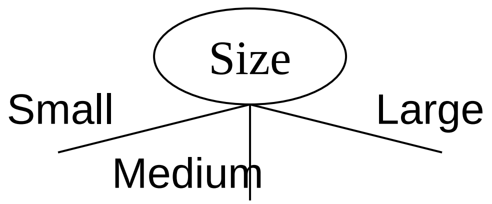


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

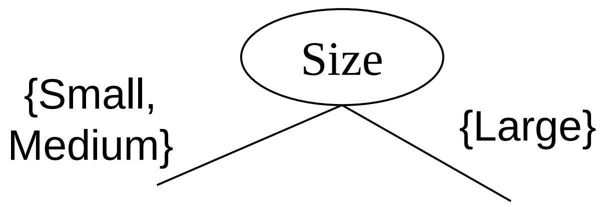


Splitting Based on Ordinal Attributes

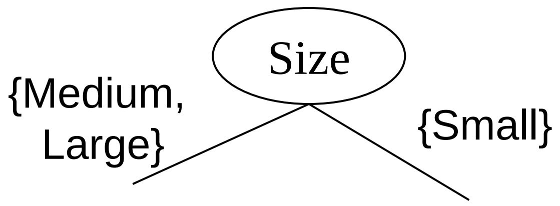
- Multi-way split: Use as many partitions as distinct values.



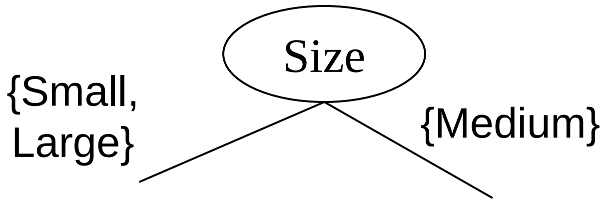
- Binary split: Divides values into two subsets – respects the order. Need to find optimal partitioning.



OR



- What about this split?

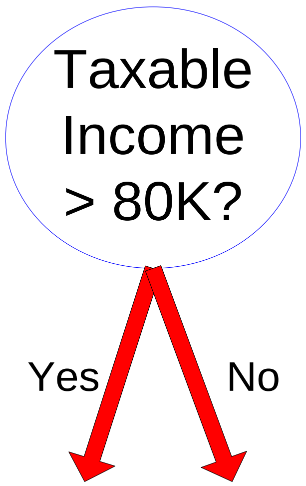


Splitting Based on Continuous Attributes

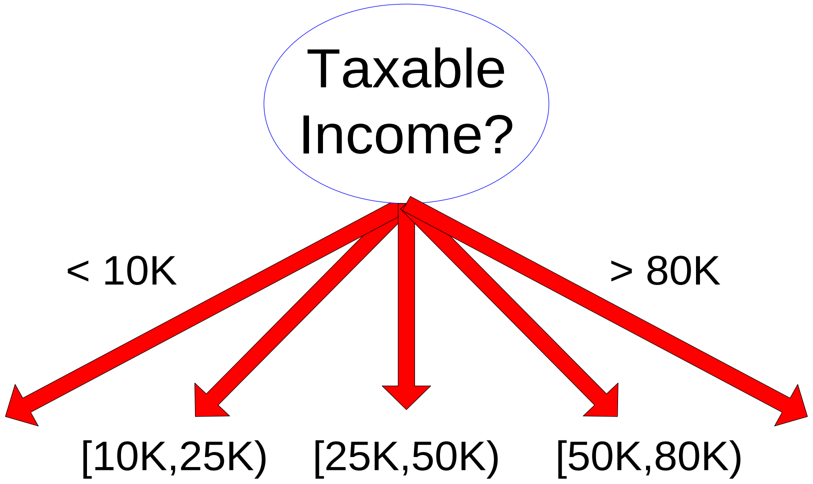
- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more computationally intensive

Splitting Based on Continuous Attributes



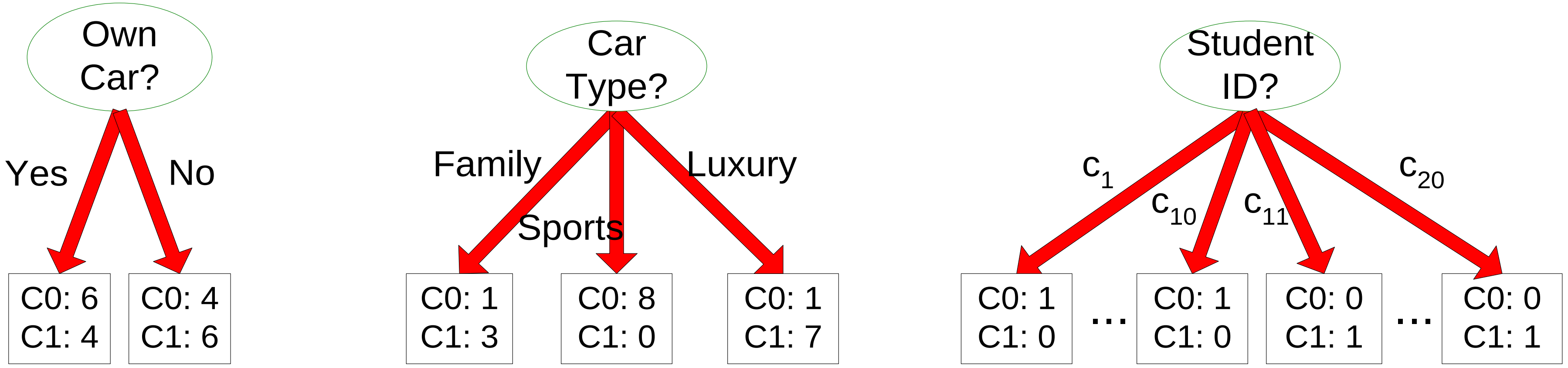
(i) Binary split



(ii) Multi-way split

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Creation of nodes with homogeneous class distribution is preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

- Ideas?

Measuring Node Impurity

- We are at a node D_t and the samples belong to classes $\{1, \dots, c\}$
  - $p(i|t)$: fraction of records associated with node D_t belonging to class $\boldsymbol{i}$
- Impurity measures:

$$Entropy(D_t) = -\sum_{i=1}^{c} p(i|t) \log p(i|t)$$

  - Used in ID3 and C4.5

$$Gini(D_t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

$$Classification\ Error(D_t) = 1 - \max p(i|t)$$

  - Used in CART, SLIQ, SPRINT.

Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

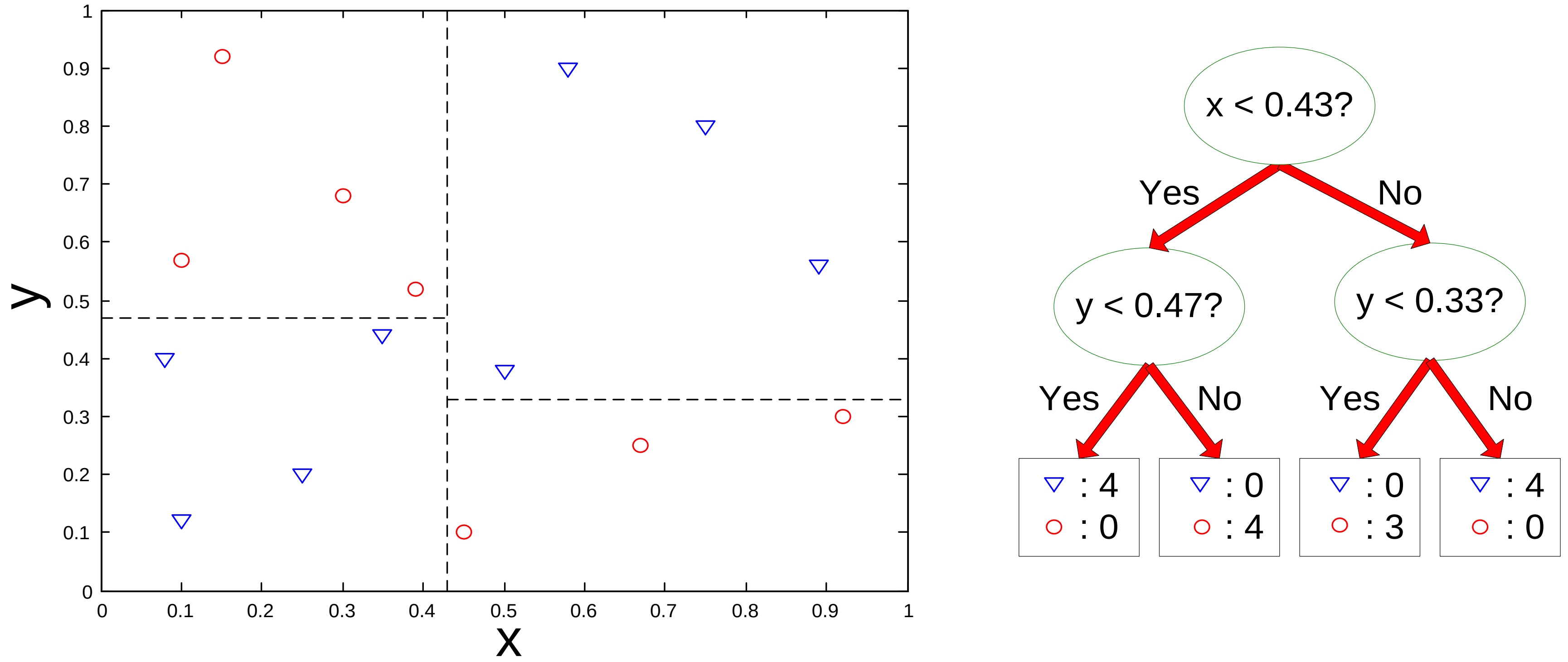
- You can download the software from:
  http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

EXPRESSIVENESS

Expressiveness

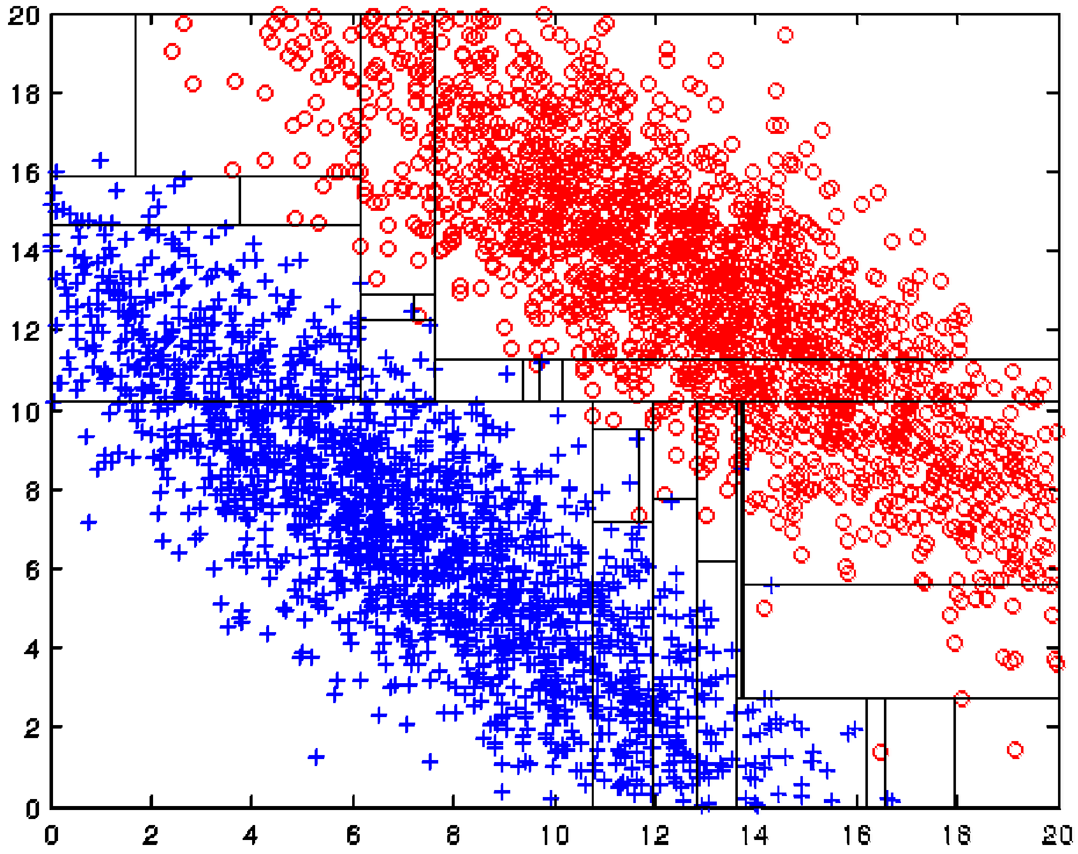
- A classifier defines a function that discriminates between two (or more) classes.
- The expressiveness of a classifier is the class of functions that it can model, and the kind of data that it can separate
- When we have discrete (or binary) values, we are interested in the class of boolean functions that can be modeled
- When the data-points are real vectors we talk about the decision boundary that the classifier can model
  - The decision boundary is the (multi-dimensional) surface defined by the function of the classifier that separates the YES and NO decisions

Decision Boundary for Decision Trees



- Consider a decision tree on real data where the test conditions involve a single attribute at a time, and a Yes/No question
- Each test defines a line parallel to an axis (the one corresponding to the test attribute)
- The decision boundary is a collection of lines parallel to the axes

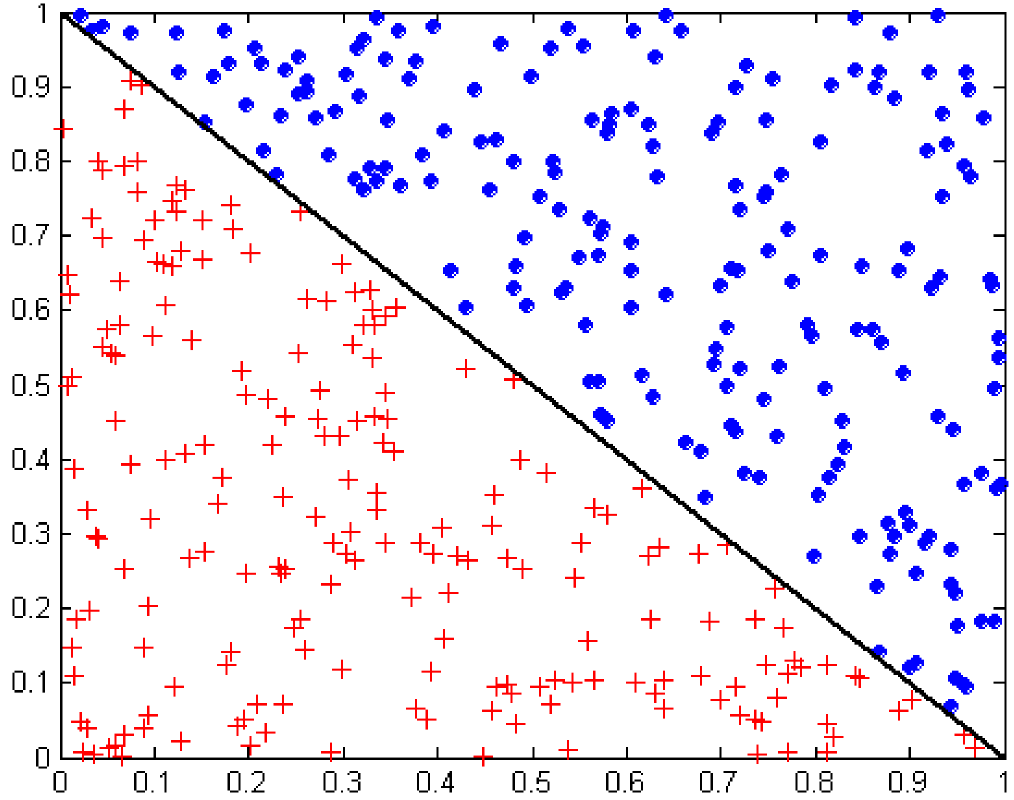
Limitations of single attribute-based decision boundaries

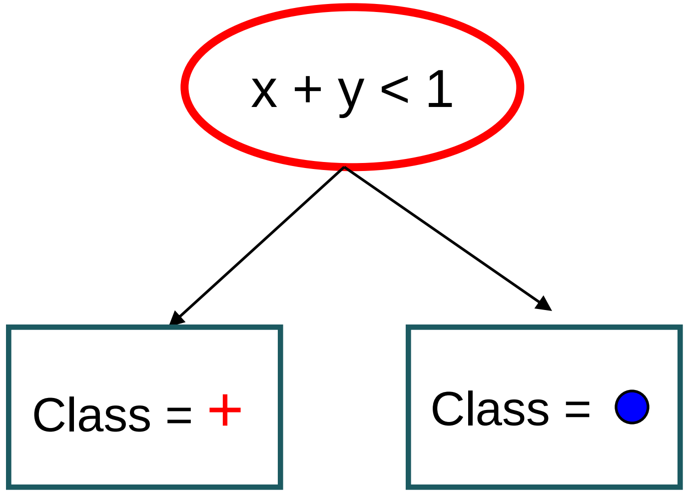


Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

The resulting boundary is very complex.

Oblique Decision Trees





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

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

- Less expressive for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

NEAREST NEIGHBOR CLASSIFICATION

Instance-Based Classifiers

Set of Stored Cases

Atr1	………	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

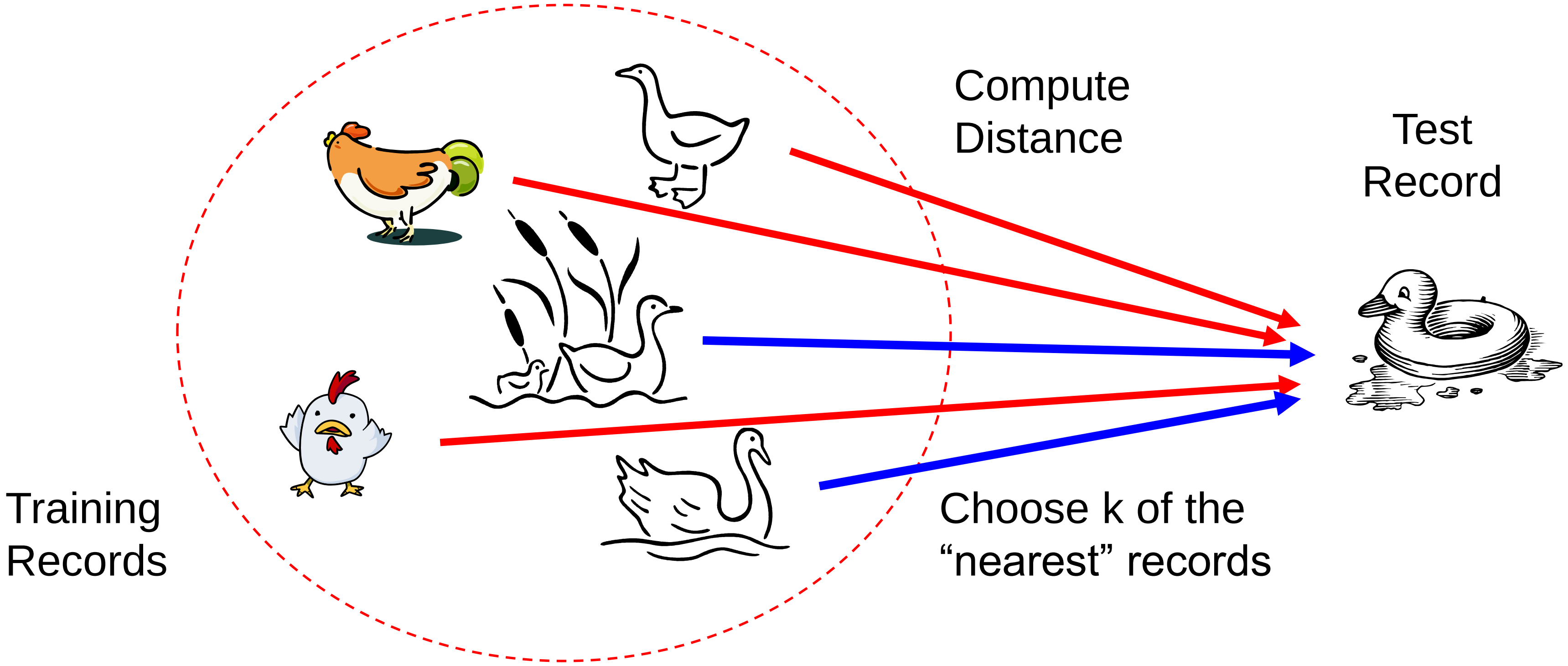
Atr1	………	AtrN

Instance Based Classifiers

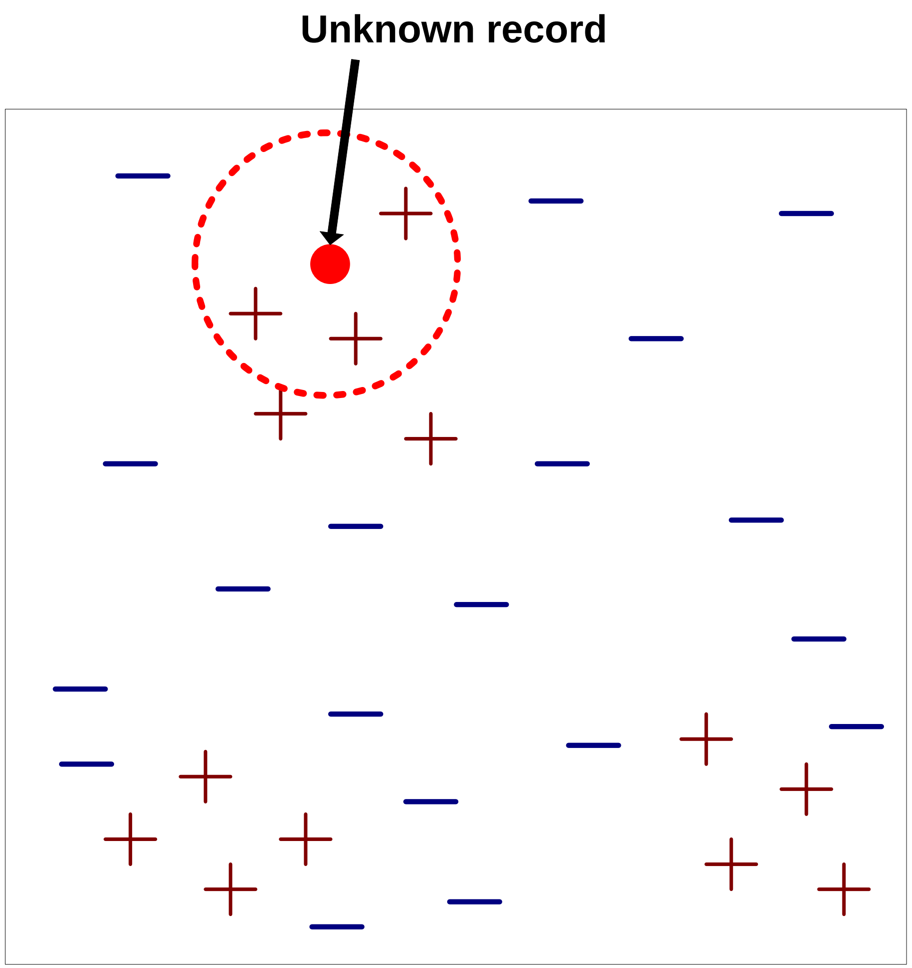
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor classifier
    - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

- Basic idea:
  - *“If it walks like a duck, quacks like a duck, then it’s probably a duck”*



Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  1. Compute distance to other training records
  2. Identify *k* nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

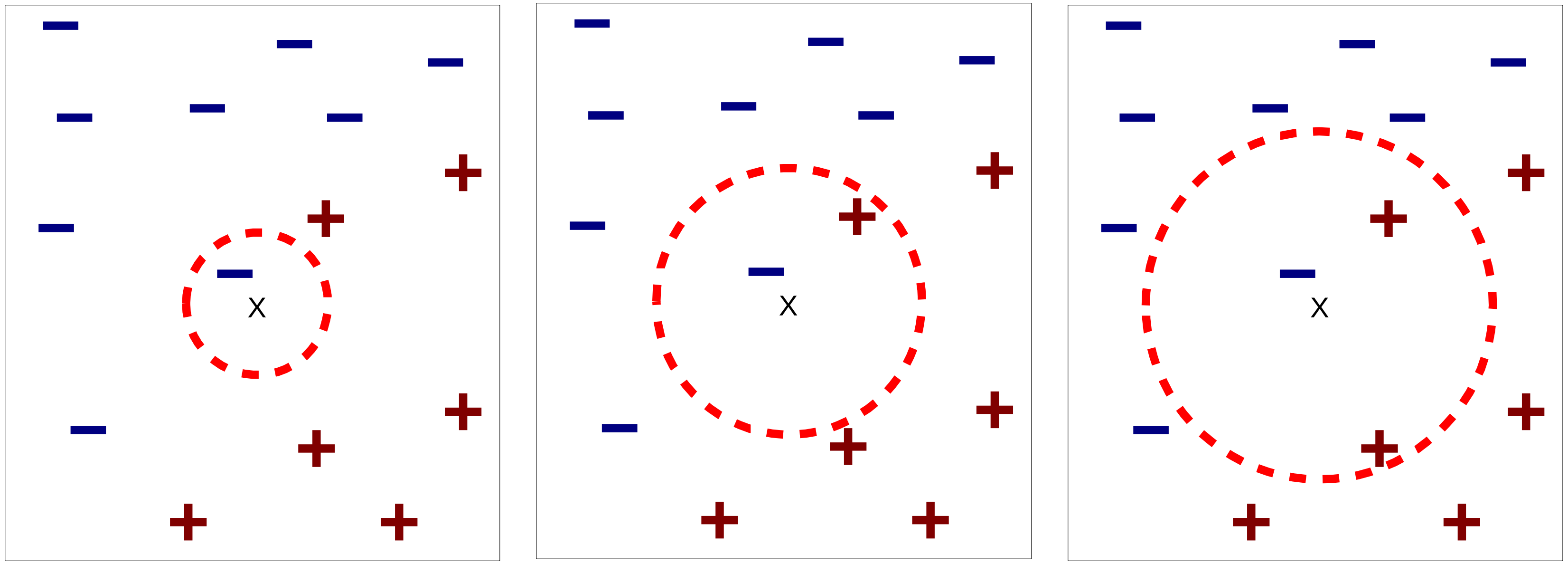
Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Definition of Nearest Neighbor

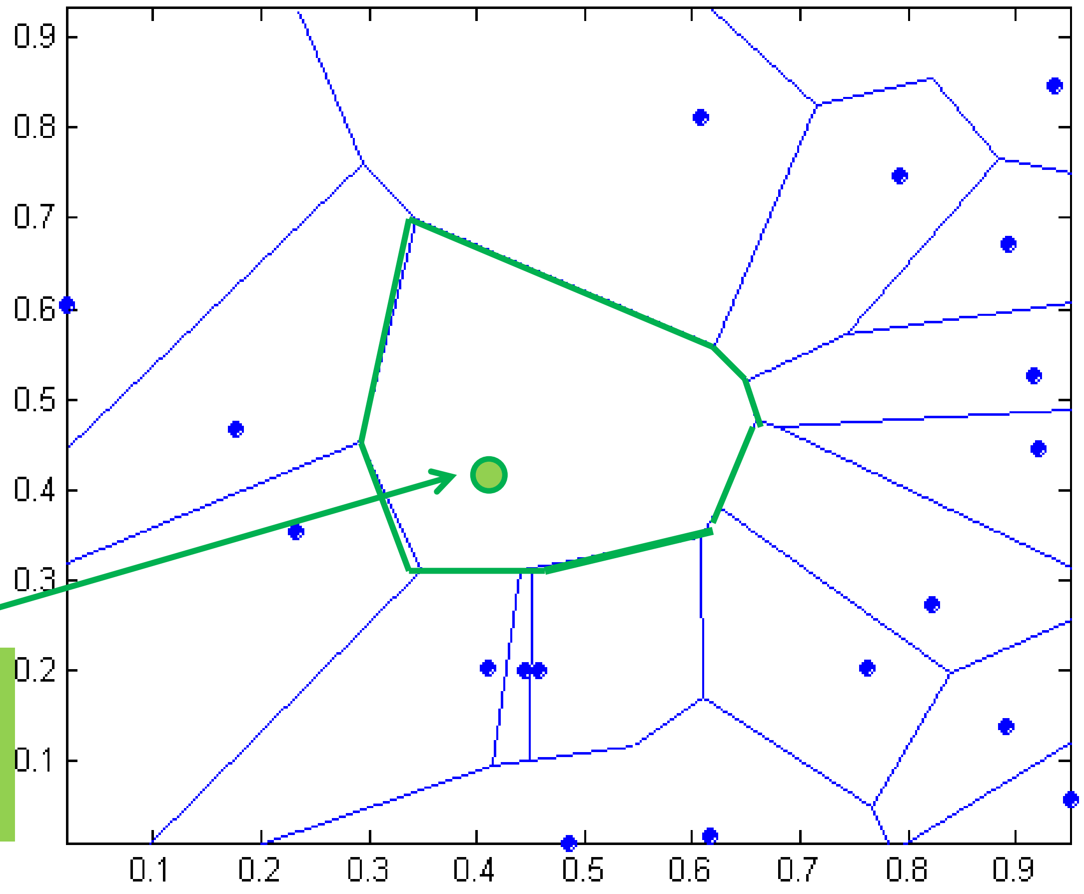


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1 nearest-neighbor

Voronoi Diagram defines the classification boundary



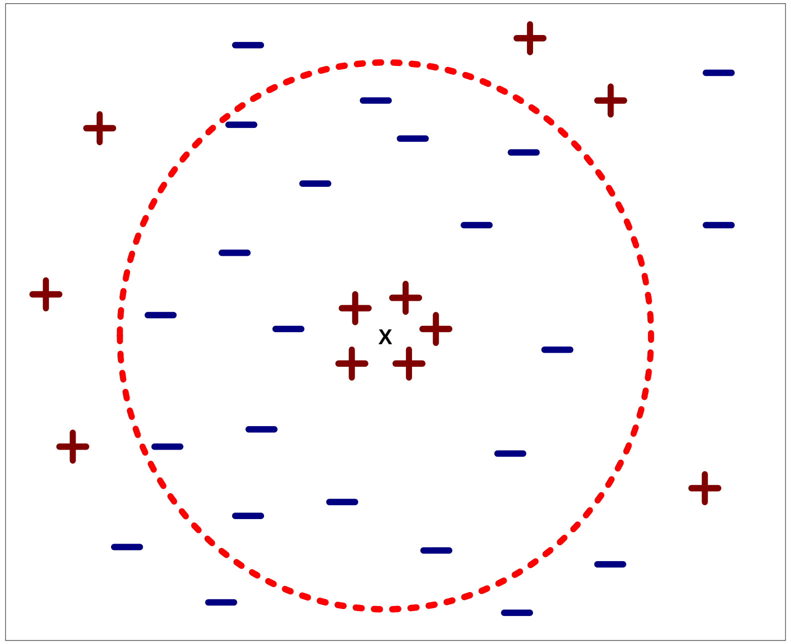
The area takes the class of the green point

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

The value of k determines the complexity of the model

Lower k produces more complex models



Example

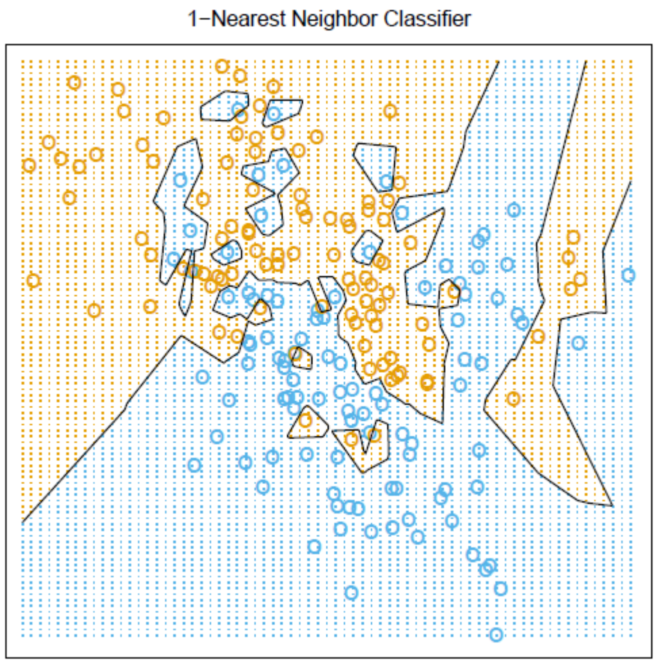


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

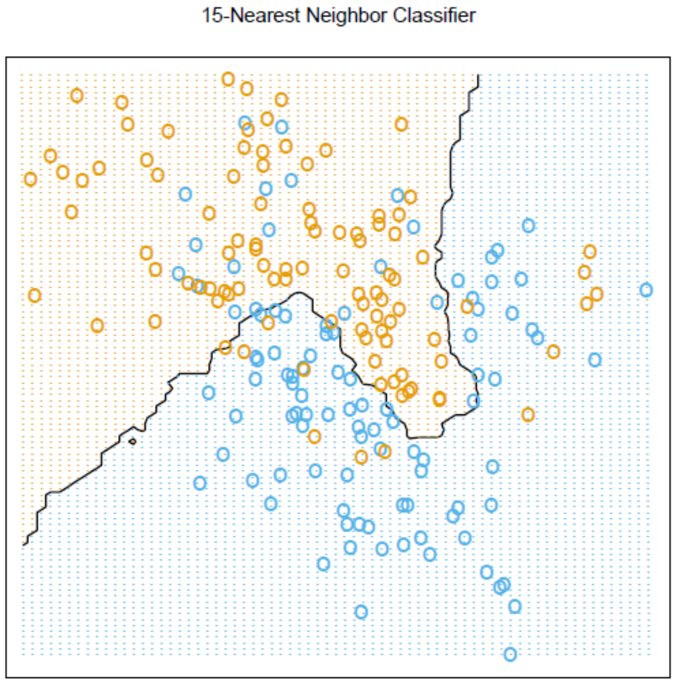


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		d = 1.4142

◆ Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision trees
- Classifying unknown records is relatively expensive
  - Naïve algorithm: O(n)
  - Need for structures to retrieve nearest neighbors fast.
    - The Nearest Neighbor Search problem.
    - Also, Approximate Nearest Neighbor Search

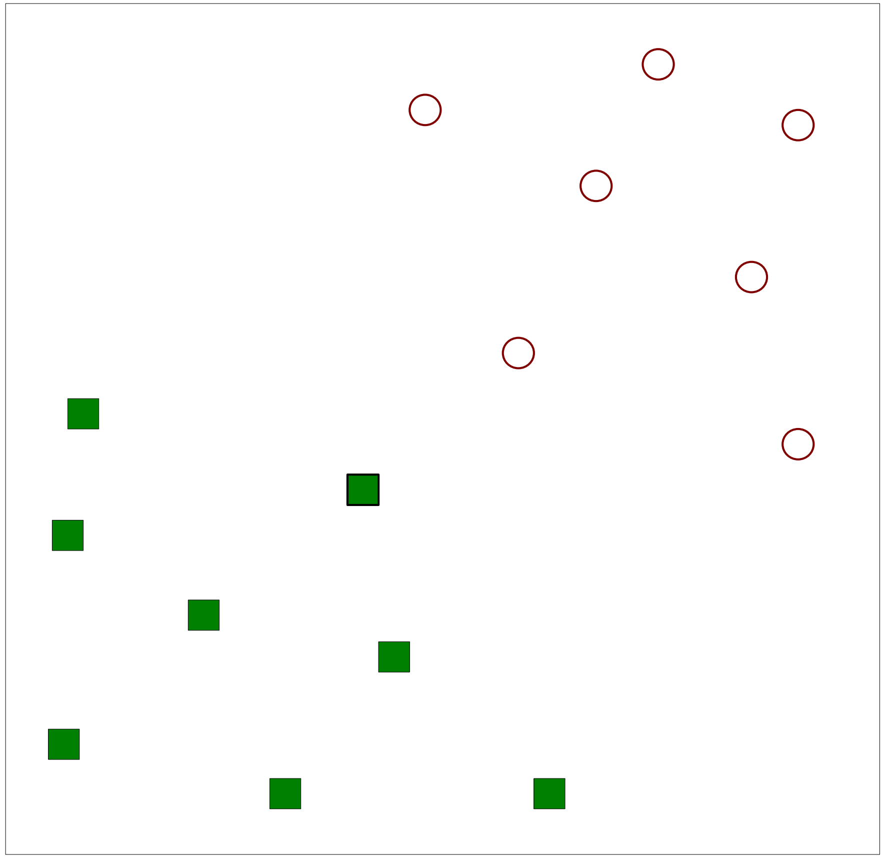
- Issues with distance in very high-dimensional spaces

SUPPORT VECTOR MACHINES

Linear classifiers

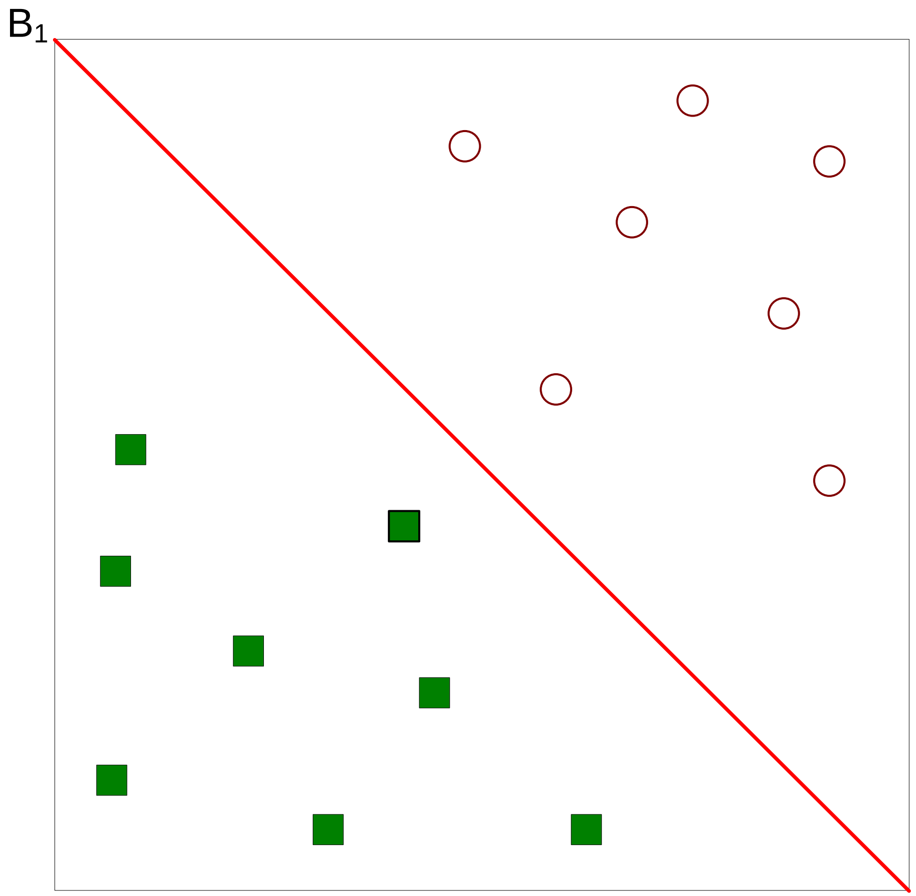
- SVMs are part of a family of classifiers that assumes that the classes are linearly separable
- That is, there is a hyperplane that separates (approximately, or exactly) the instances of the two classes.
- The goal is to find this hyperplane

Support Vector Machines



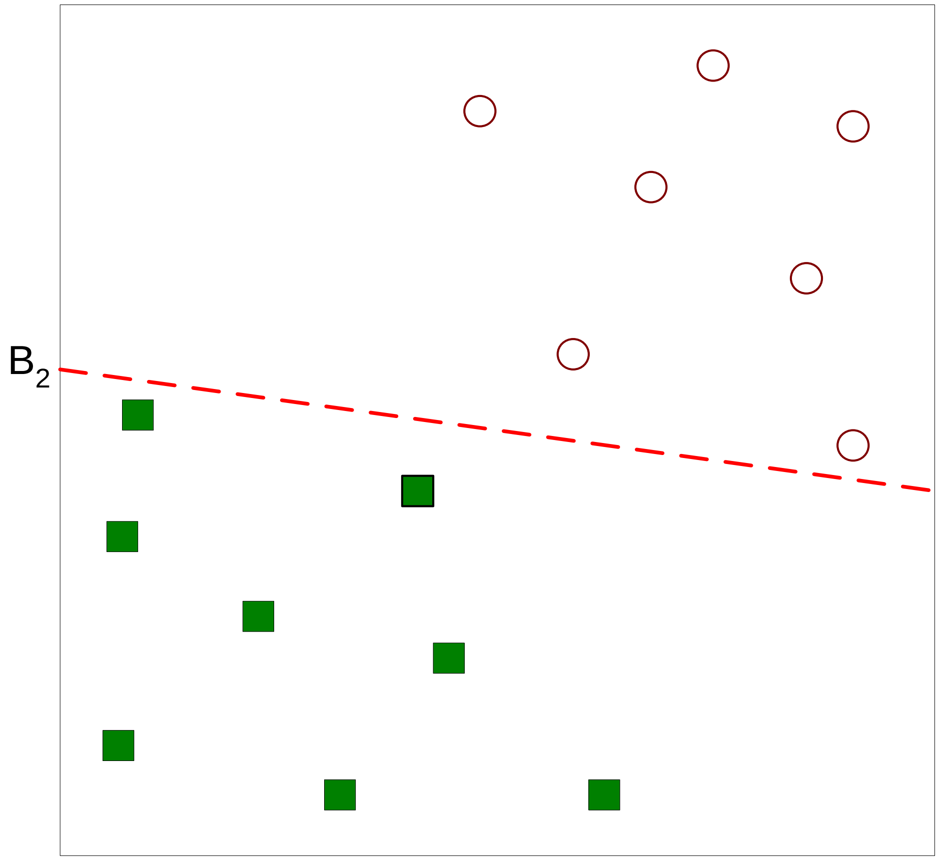
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



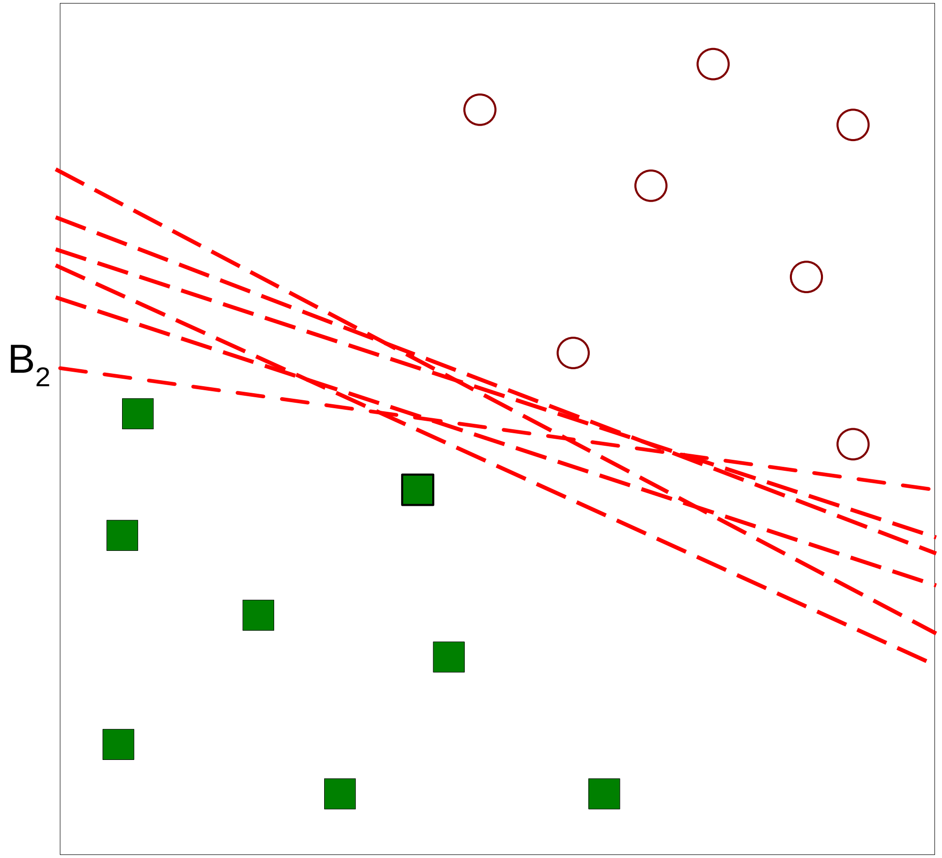
- One Possible Solution

Support Vector Machines



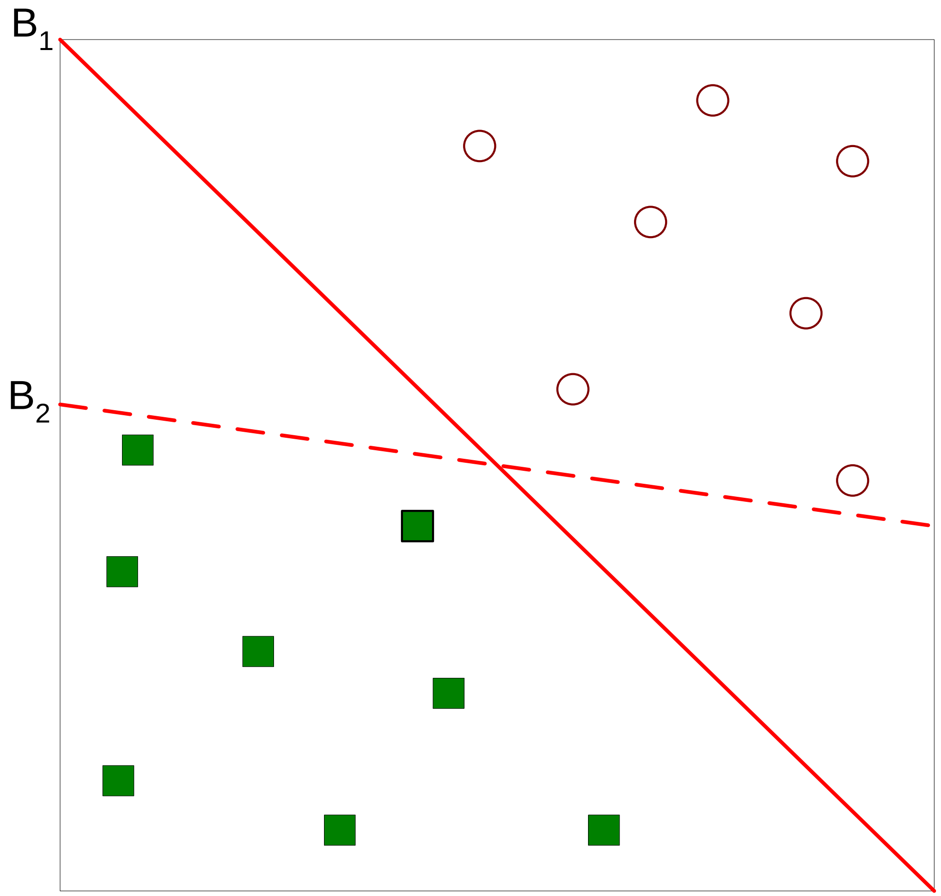
- Another possible solution

Support Vector Machines



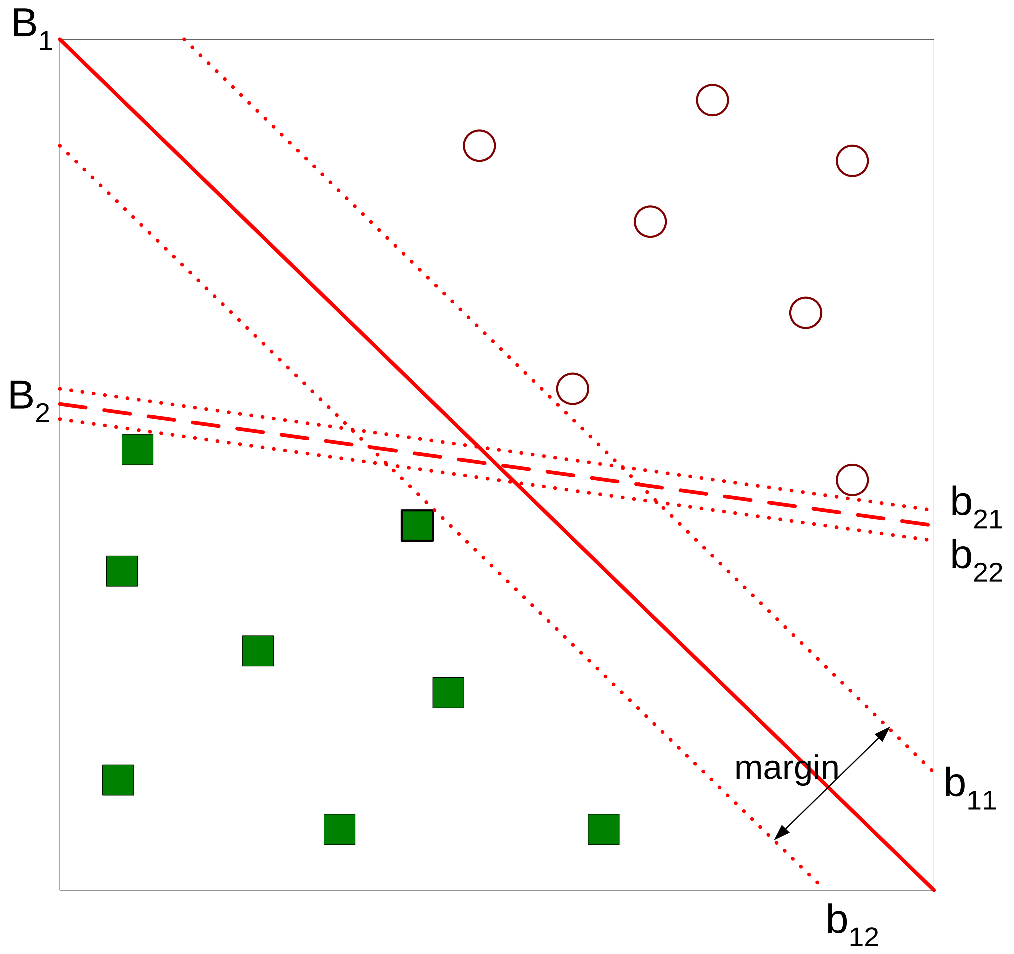
- Other possible solutions

Support Vector Machines



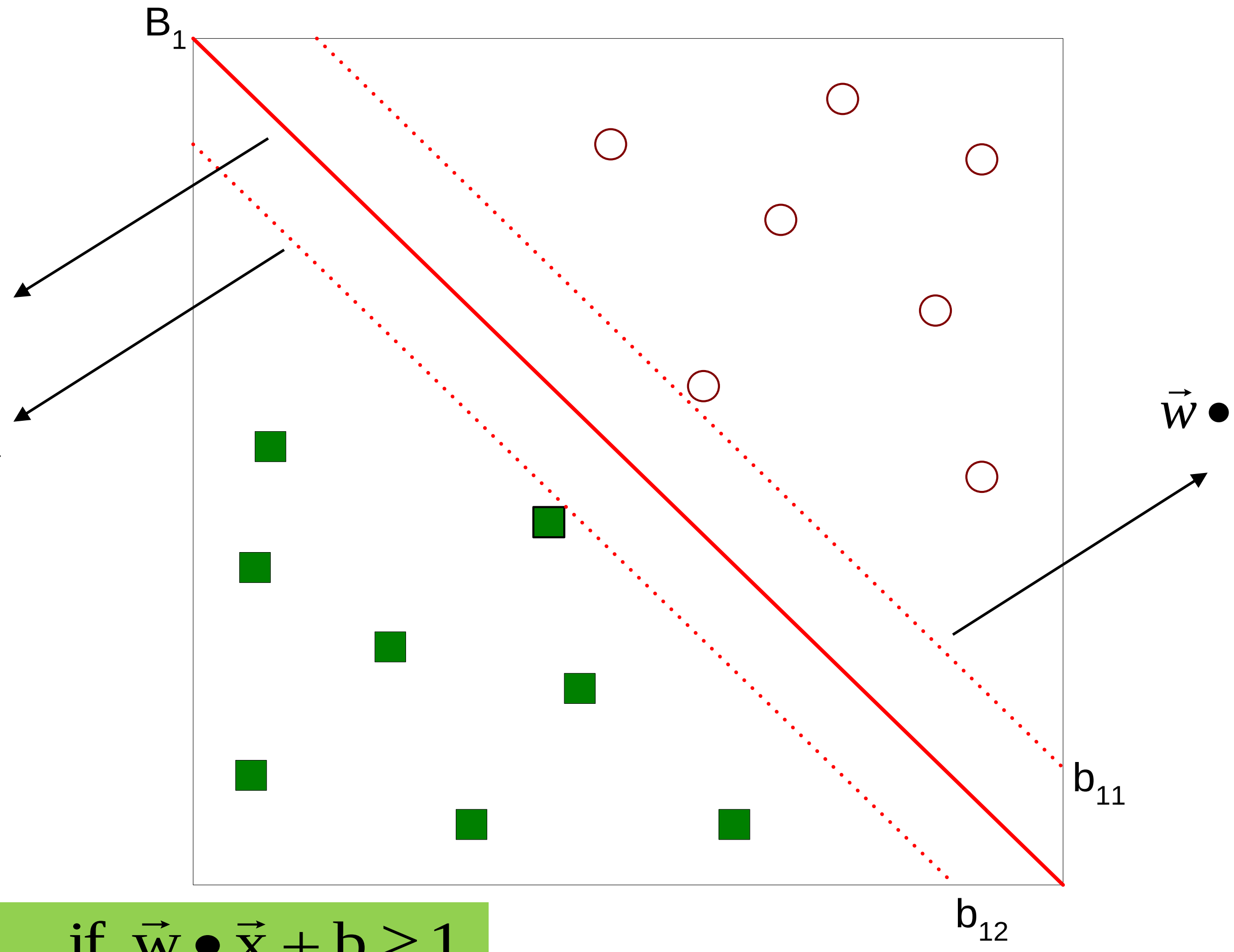
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin : B1 is better than B2

Support Vector Machines



$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$

$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

Support Vector Machines

- We want to maximize: $Margin = \frac{2}{\|\vec{w}\|}$

- Which is equivalent to minimizing: $L(\vec{w}) = \frac{\|\vec{w}\|}{2}$

- But subjected to the following constraints:

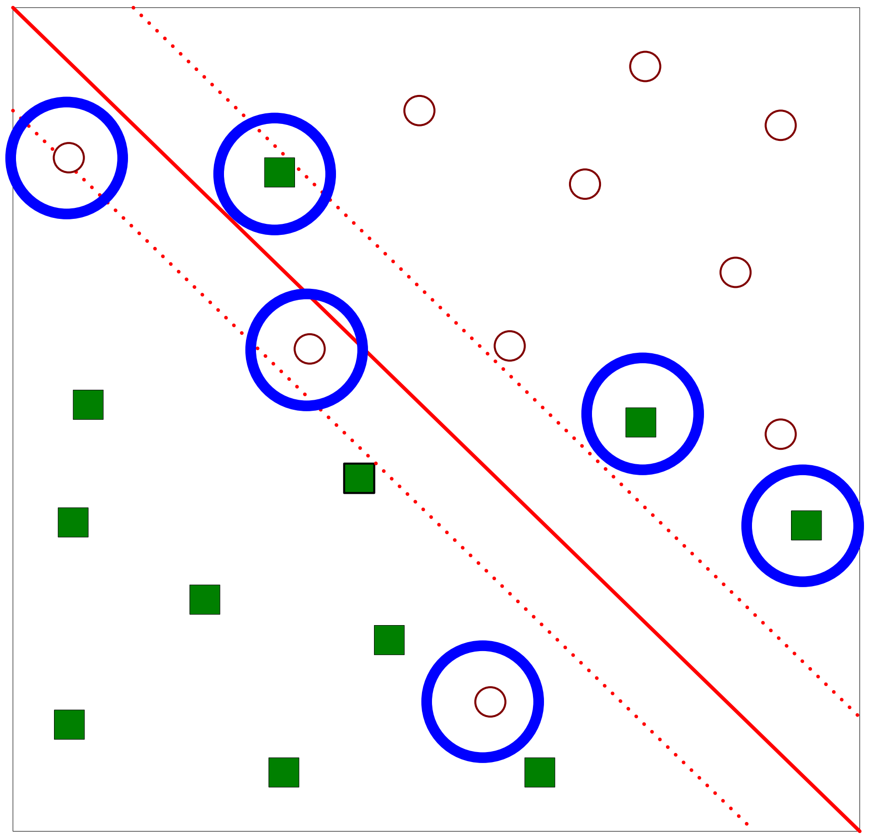
$$\vec{w} \cdot \vec{x_i} + b \geq 1 \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 \text{ if } y_i = -1$$

Concisely:
$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1$

- This is a constrained optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)

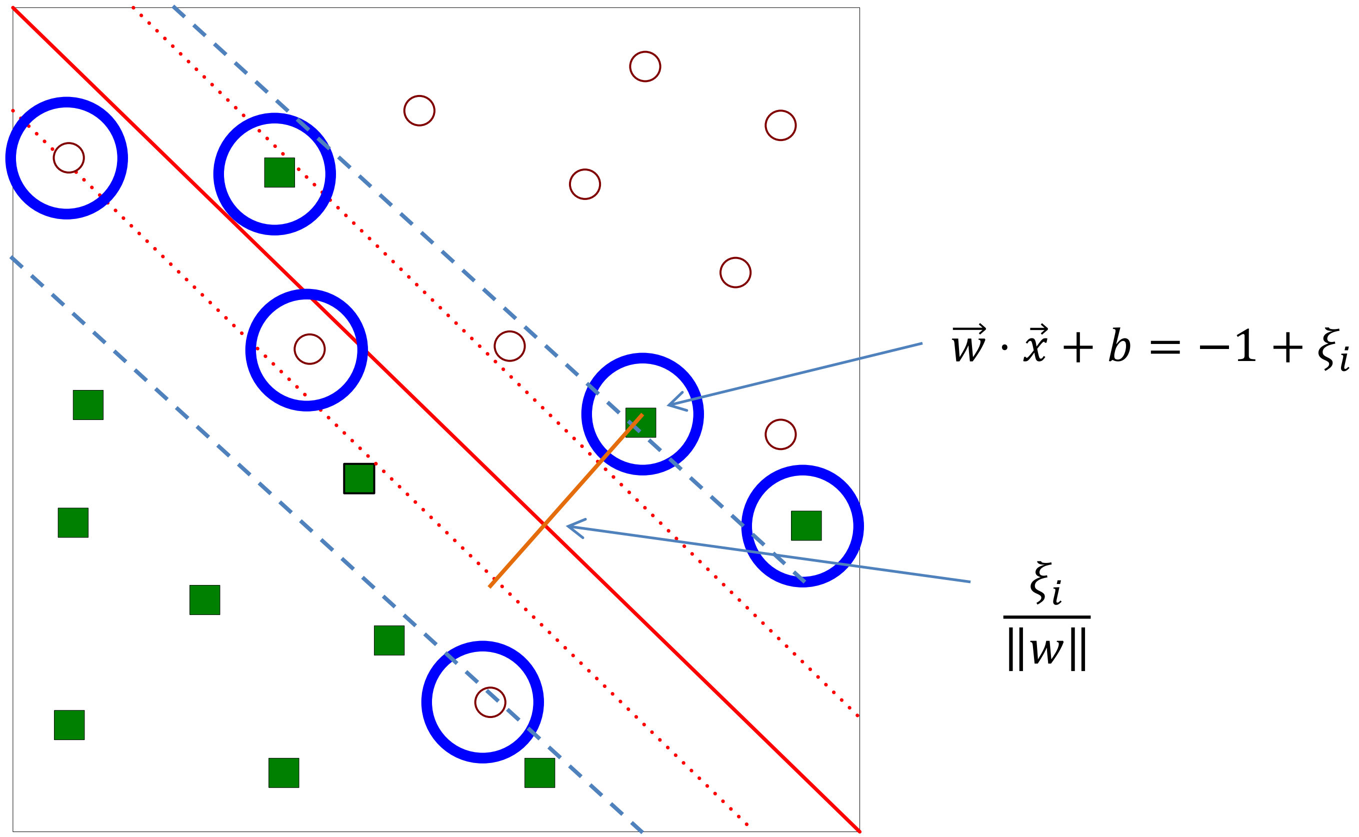
Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
- Introduce slack variables
  - Minimize:

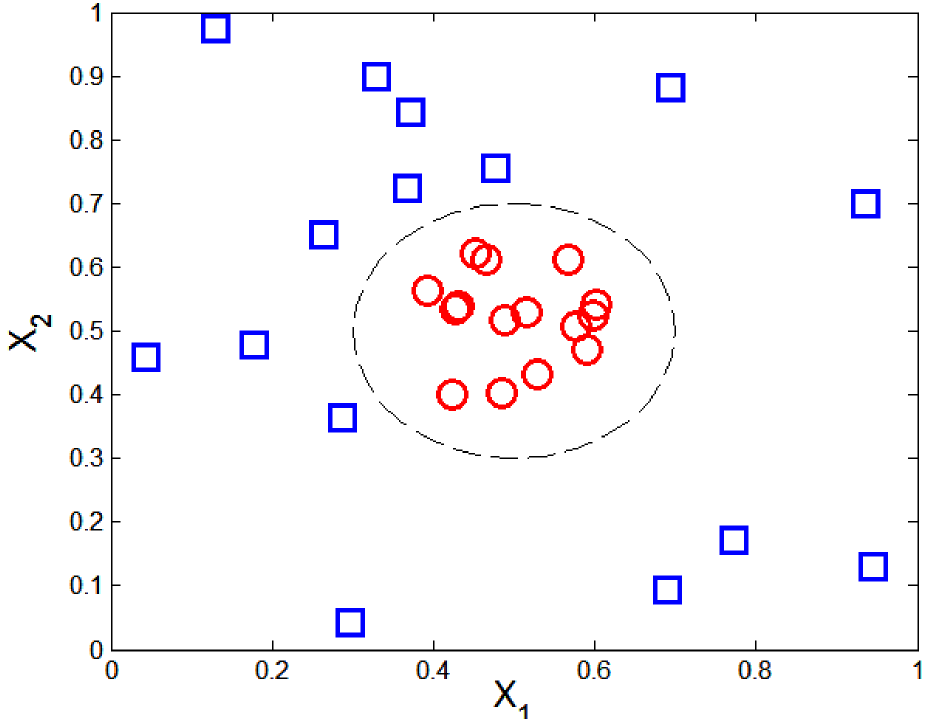
$$L(w) = \frac{\|\vec{w}\|}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

  - Subject to:

$$\vec{w} \cdot \vec{x_i} + b \geq 1 - \xi_i \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 + \xi_i \text{ if } y_i = -1$$

Nonlinear Support Vector Machines

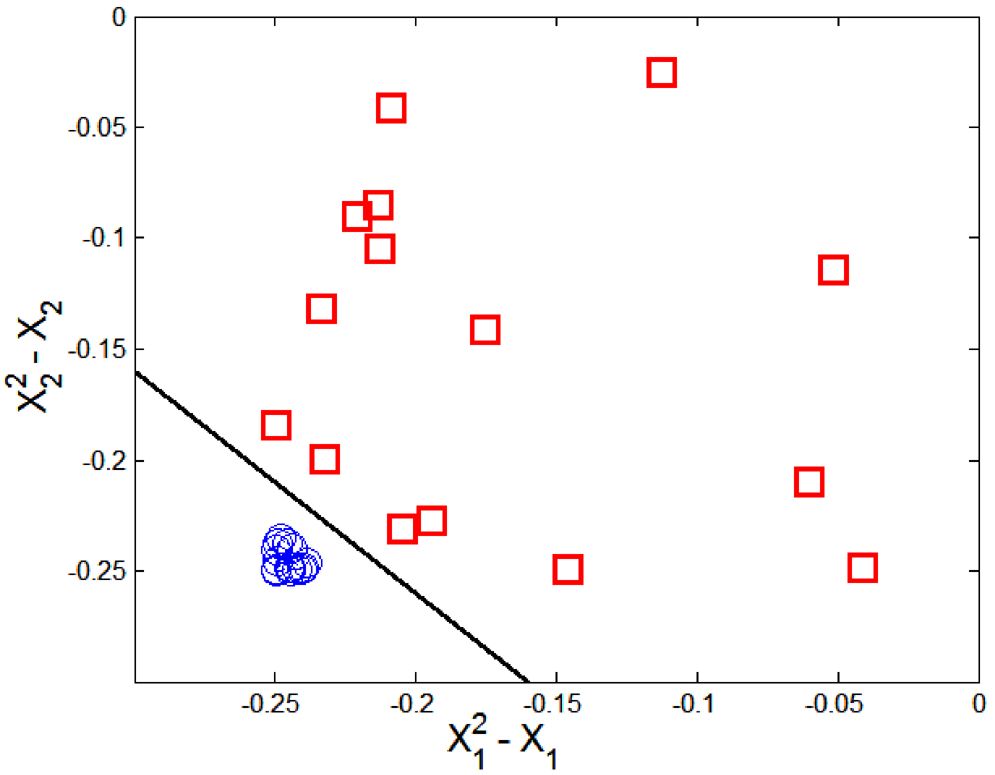
- What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

Nonlinear Support Vector Machines

- Trick: Transform data into higher dimensional space



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \cdot \Phi(\vec{x}) + b = 0$$

Learning Nonlinear SVM

- Optimization problem:

$$\min_{w} \frac{\|\mathbf{w}\|^2}{2}$$

$$\textit{subject to} \quad y_i(w \cdot \Phi(x_i) + b) \geq 1, \ \forall\{(x_i, y_i)\}$$

- Which leads to the same set of equations (but involve $\Phi(x)$ instead of x)

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$

$$\lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} = 0,$$

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = sign(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b).$$

Learning NonLinear SVM

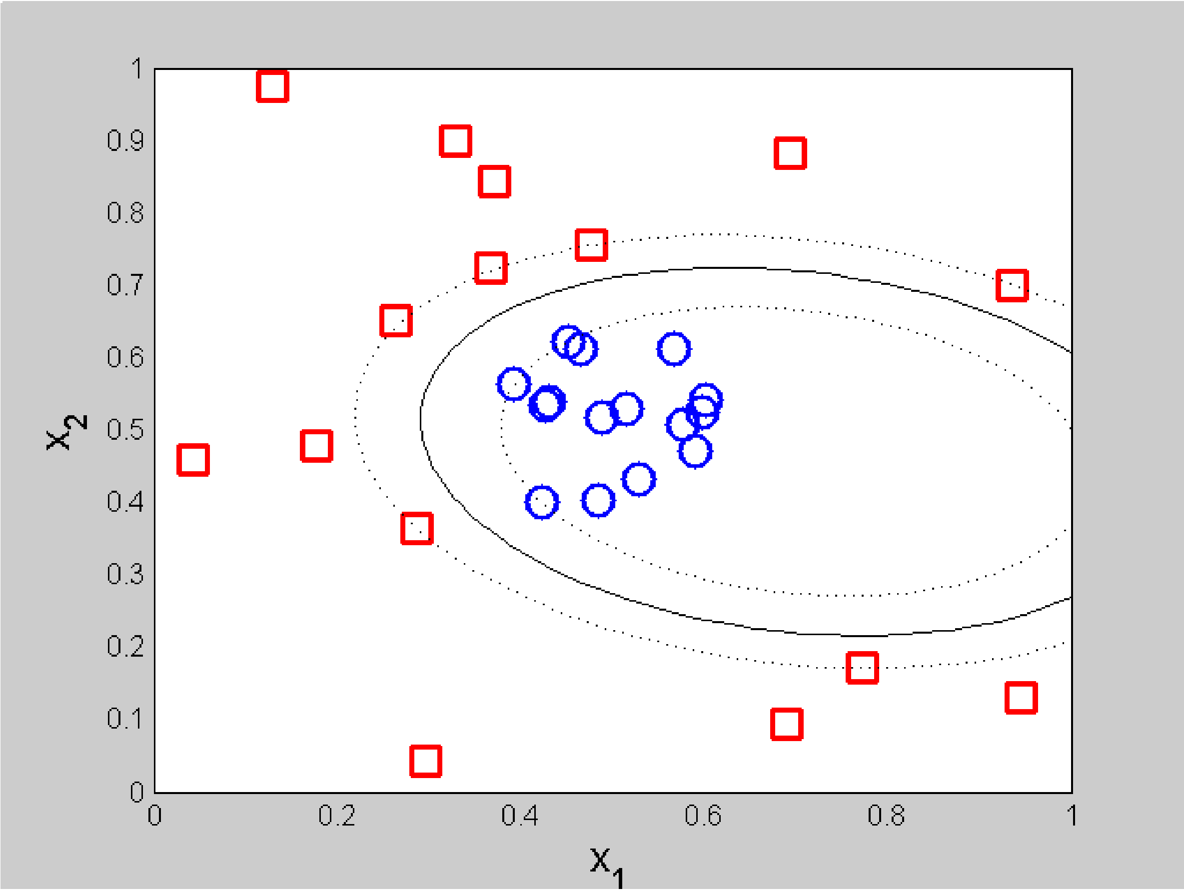
- Issues:
  - What type of mapping function Φ should be used?
  - How to do the computation in high dimensional space?
    - Most computations involve dot product $\Phi(x_i) \cdot \Phi(x_j)$
    - Curse of dimensionality?

Learning Nonlinear SVM

- Kernel Trick:
  - $\Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j)$
  - $K(x_i, x_j)$ is a kernel function (expressed in terms of the coordinates in the original space)
    - Examples:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$
$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)}$$
$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$$

Example of Nonlinear SVM



SVM with polynomial degree 2 kernel

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^2$$

Learning Nonlinear SVM

- Advantages of using kernel:
  - Don't have to know the mapping function Φ
  - Computing dot product $\Phi(x_i) \cdot \Phi(x_j)$ in the original space avoids curse of dimensionality

- Not all functions can be kernels
  - Must make sure there is a corresponding Φ in some high-dimensional space
  - Mercer's theorem (see textbook)

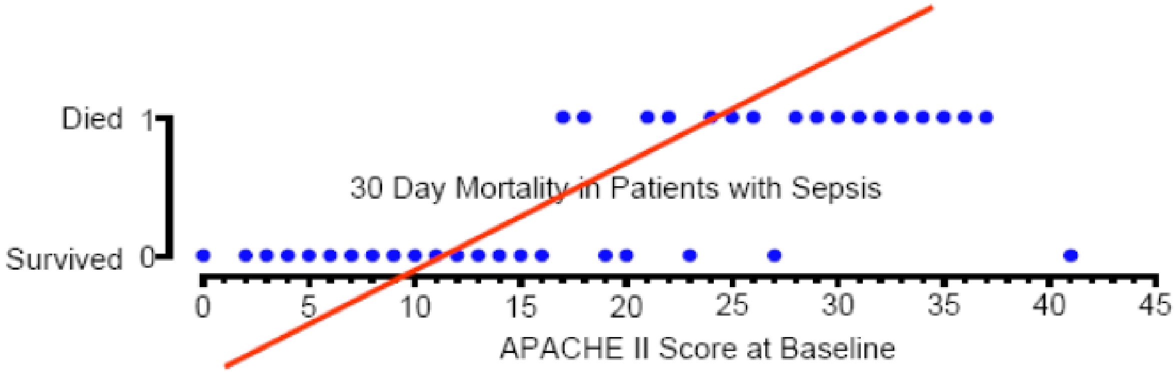
LOGISTIC REGRESSION

Classification via regression

- Instead of predicting the class of a record we want to predict the probability of the class given the record
- Transform the classification problem into a regression problem.
- But how do you define the probability that you want to predict?

Linear regression

- A simple approach: use linear regression to learn a linear function that predicts 0/1 values
  - Not good: It may produce negative probabilities, or probabilities that are greater than 1.
  - Also the probabilities it produces are not what we want. We want probability close to zero for small values, and close to 1 for large, and a transition from 0 to 1 around the value 20

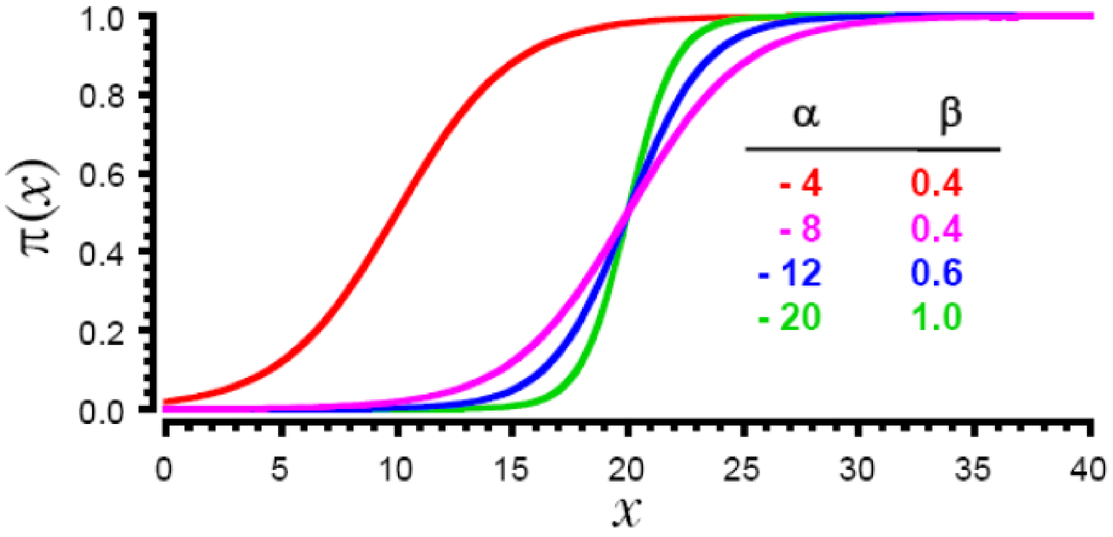


The logistic function

$$f(x) = \frac{1}{1 + e^{-\alpha - \beta x}}$$

β controls the slope
α controls the position of the turning point

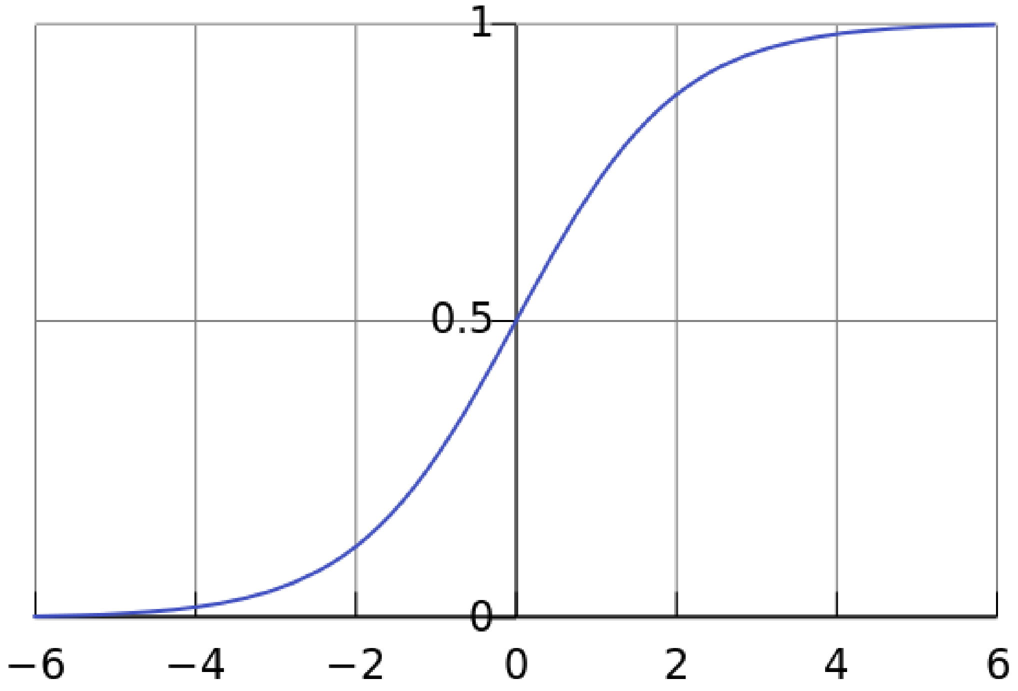
$$\pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x))$$



When $x = -\alpha / \beta$, $\alpha + \beta x = 0$ and hence $\pi(x) = 1/(1+1) = 0.5$

Logistic Regression

$$f(x) = \frac{1}{1+e^{-x}}$$



Class Probabilities

$$P(C_+|x) = \frac{1}{1+e^{-\beta x - a}}$$

$$P(C_-|x) = \frac{e^{-\beta x - a}}{1+e^{-\beta x - a}}$$

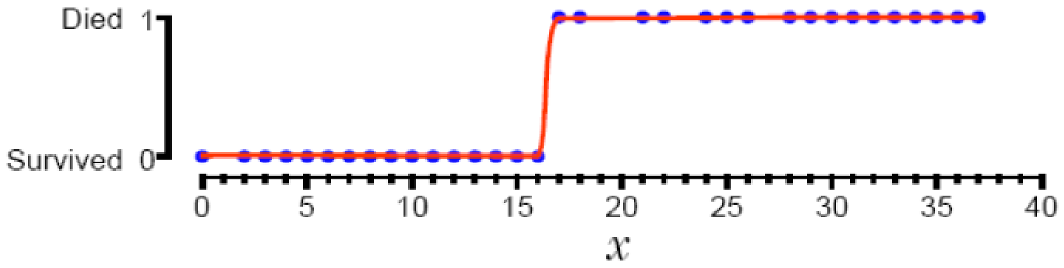
Logistic Regression: Find the values β, α that maximize the probability of the observed data

$$\log \frac{P(C_+|x)}{P(C_-|x)} = \beta x + a$$

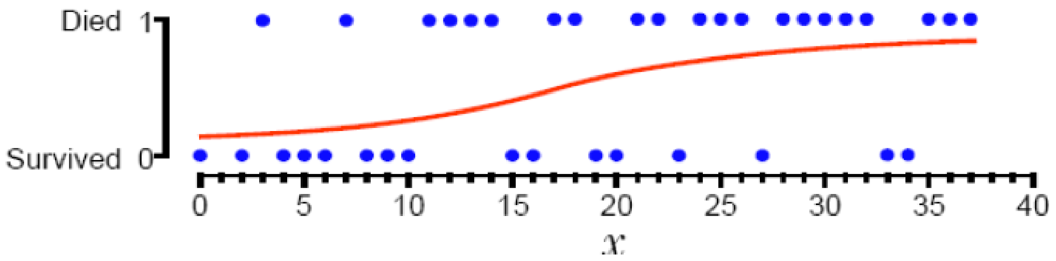
Linear regression on the log-odds ratio

Logistic Regression in one dimension

Data that has a sharp survival cut off point between patients who live or die should have a large value of β.



Data with a lengthy transition from survival to death should have a low value of β.



Logistic Regression in one dimension

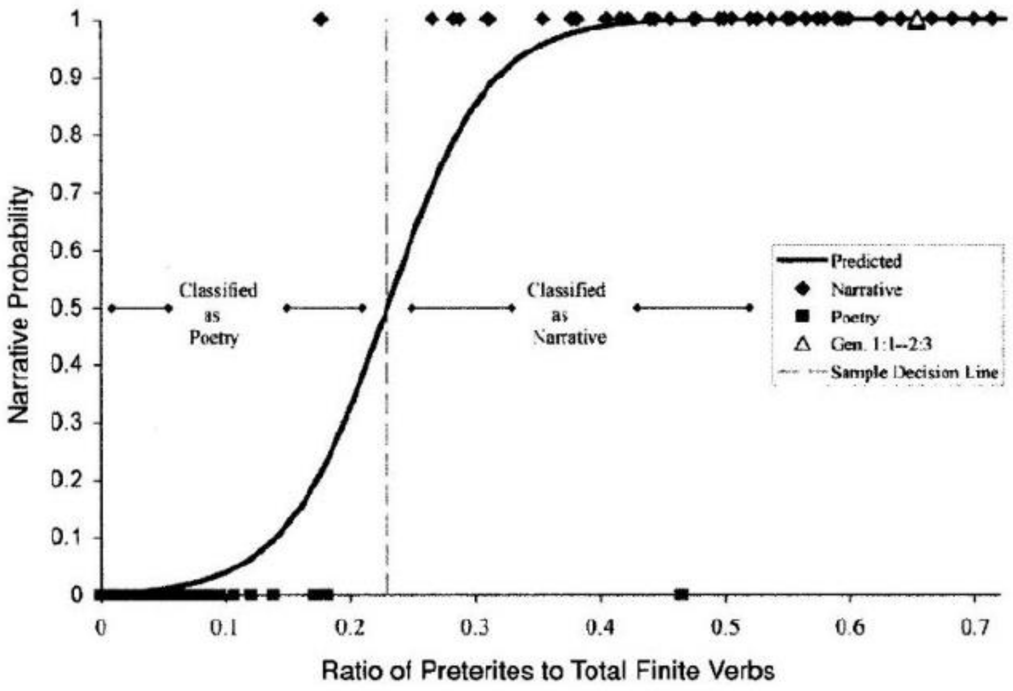


Figure 10-3. The solid curved line is called a logistic regression curve. The vertical axis measures the probability that an Old Testament passage is narrative, based on the use of preterite verbs. The probability is zero for poetry and unity or one for narrative. Passages with high preterite verb counts, falling to the right of the vertical dotted line, are likely narrative. The triangle on the upper right represents Genesis 1:1–2:3, which is clearly literal, narrative history.

Class probabilities for multiple dimensions

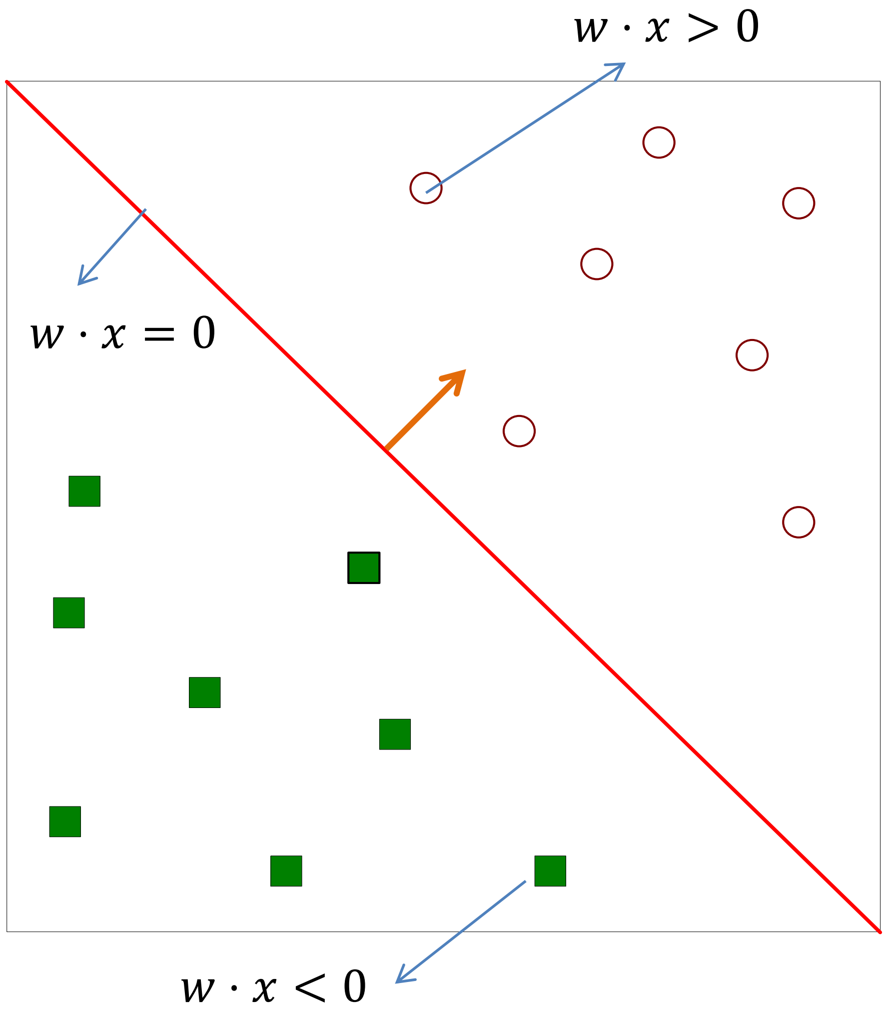
- Assume a linear classification boundary

For the positive class the bigger the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the positive class

- Define $P(C_+|x)$ as an increasing function of $w \cdot x$

For the negative class the smaller the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the negative class

- Define $P(C_-|x)$ as a decreasing function of $w \cdot x$



Logistic Regression

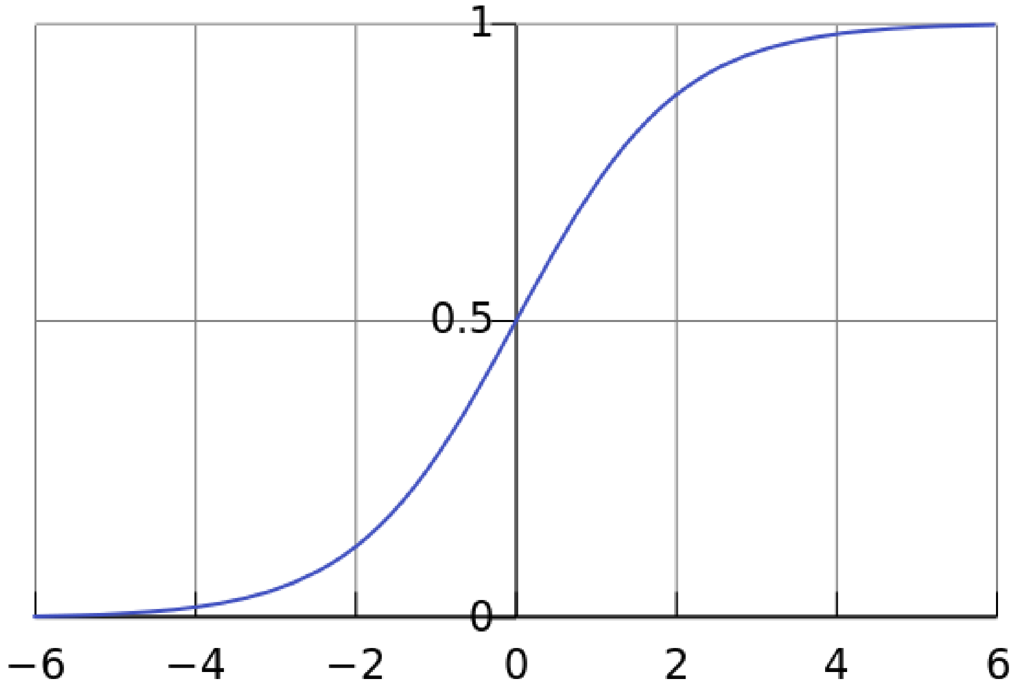
$$f(t) = \frac{1}{1+e^{-t}}$$

Class probabilities

$$P(C_+|x) = \frac{1}{1+e^{-w\cdot x-a}}$$

$$P(C_-|x) = \frac{e^{-w\cdot x-a}}{1+e^{-w\cdot x-a}}$$

Logistic Regression: Find the vector w, a that maximizes the probability of the observed data



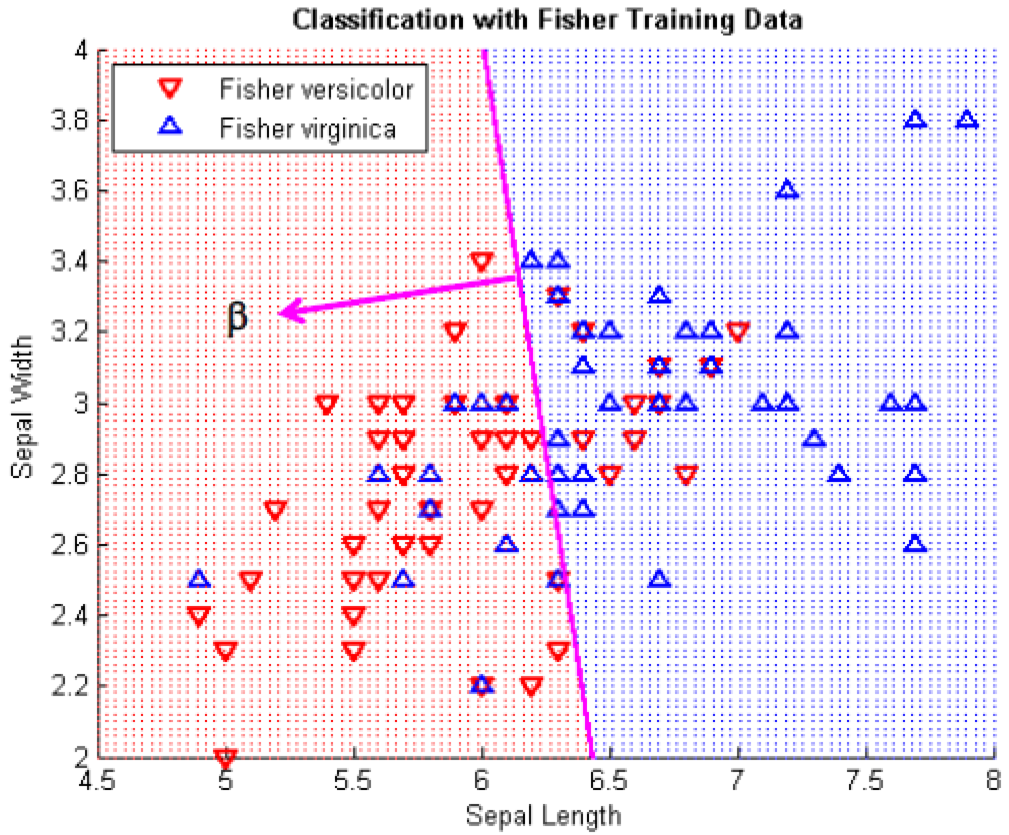
$$\log\frac{P(C_+|x)}{P(C_-|x)} = w\cdot x + a$$

Linear regression on the log-odds ratio

Logistic regression in 2-d

Coefficients

$$\beta_1 = -1.9$$
$$\beta_2 = -0.4$$
$$\alpha = 13.04$$



Estimating the coefficients

- Maximum Likelihood Estimation:
  - We have pairs of the form (x_i, y_i)
- Log Likelihood function

$$L(w) = \sum_i [y_i \log P(y_i|x_i, w) + (1 - y_i) \log(1 - P(y_i|x_i, w))]$$

- Unfortunately, it does not have a closed form solution
  - Use gradient descend to find local minimum

Logistic Regression

- Produces a probability estimate for the class membership which is often very useful.
- The weights can be useful for understanding the feature importance.
- Works for relatively large datasets
- Fast to apply.

NAÏVE BAYES CLASSIFIER

Bayes Classifier

- A probabilistic framework for solving classification problems
- A, C random variables
- Joint probability: Pr(A=a,C=c)
- Conditional probability: Pr(C=c | A=a)
- Relationship between joint and conditional probability distributions

$$\Pr(C, A) = \Pr(C|A)\, P(A) = P(A|C)P(C)$$

- Bayes Theorem:

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

Bayesian Classifiers

- How to classify the new record X = (‘Yes’, ‘Single’, 80K)

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Find the class with the highest probability given the vector values.

Maximum Aposteriori Probability estimate:

- Find the value c for class C that maximizes P(C=c| X)

- How do we estimate P(C|X) for the different values of C?
- We want to estimate
  - P(C=Yes| X)
  - P(C=No| X)

Bayesian Classifiers

- In order for probabilities to be well defined:
  - Consider each attribute and the class label as random variables
  - Probabilities are determined from the data

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Evade C
Event space: {Yes, No}
$$P(C) = (0.3, 0.7)$$

Refund A_1
Event space: {Yes, No}
$$P(A_1) = (0.3,0.7)$$

Martial Status A_2
Event space: {Single, Married, Divorced}
$$P(A_2) = (0.4,0.4,0.2)$$

Taxable Income A_3
Event space: R
$$P(A_3) \sim Normal(\mu, \sigma^2)$$
$\mu = 104$:sample mean, $\sigma^2 = 1874$:sample variance

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, An)$ using the Bayes theorem

$$P(C|A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n|C)P(C)}{P(A_1, A_2, \ldots, A_n)}$$

  - Maximizing

$$P(C \mid A_1, A_2, \ldots, An)$$

is equivalent to maximizing

$$P(A_1, A_2, \ldots, A_n|C)\ P(C)$$

    - The value $P(A_1, \ldots, A_n)$ is the same for all values of C.

- How do we estimate $P(A_1, A_2, \ldots, A_n|C\ )$?

Naïve Bayes Classifier

- Assume conditional independence among attributes A_i when class C is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1|C)\, P(A_2|C) \cdots P(A_n|C)$
  - We can estimate $P(Ai|\, C)$ from the data.
  - New point $X = (A_1 = \alpha_1, \ldots, A_n = \alpha_n)$ is classified to class c if

$$P(C = c|X) = P(C = c) \prod_i P(A_i = \alpha_i|c)$$

is maximum over all possible values of C.

Example

- Record

  X = (Refund = Yes, Status = Single, Income =80K)
- For the class C :'Evade', we want to compute:

  P(C = Yes|X) and P(C = No| X)
- We compute:
  - P(C = Yes|X) = P(C = Yes)*P(Refund = Yes |C = Yes)
    *P(Status = Single |C = Yes)
    *P(Income =80K |C= Yes)
  - P(C = No|X) = P(C = No)*P(Refund = Yes |C = No)
    *P(Status = Single |C = No)
    *P(Income =80K |C= No)

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Class Prior Probability:

$$P(C = c) = \frac{N_c}{N}$$

N_c: Number of records with class c

N = Number of records

P(C = No) = 7/10

P(C = Yes) = 3/10

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a|C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class c

N_c: number of instances of class c

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a|C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class c

N_c: number of instances of class c

P(Refund = Yes|No) = 3/7

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class c

N_c: number of instances of class c

P(Refund = Yes|Yes) = 0

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a|C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class c

N_c: number of instances of class c

P(Status=Single|No) = 2/7

How to Estimate Probabilities from Data?

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
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**
10	No	Single	90K	**Yes**

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class c

N_c: number of instances of class c

P(Status=Single|Yes) = 2/3

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Numerical Attributes:

- Assume a normal distribution for each (A_i, c_j) pair

$$P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

- For Class=Yes and attribute Income
  - sample mean $\mu = 90$
  - sample variance $\sigma^2 = 25$
- For Income = 80

$$P(Income = 80 \mid Yes) = \frac{1}{\sqrt{2\pi}(5)} e^{-\frac{(80-90)^2}{2(25)}} = 0.01$$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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
10	No	Single	90K	Yes

Numerical Attributes:

- Assume a normal distribution for each (A_i, c_j) pair

$$P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

- For Class=No and attribute Income
  - sample mean $\mu = 110$
  - sample variance $\sigma^2 = 2975$
- For Income = 80

$$P(Income = 80 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(80-110)^2}{2(2975)}} = 0.0062$$

Example

- Record

  X = (Refund = Yes, Status = Single, Income =80K)

- We compute:
  - P(C = Yes|X) = P(C = Yes)*P(Refund = Yes |C = Yes)

    *P(Status = Single |C = Yes)

    *P(Income =80K |C= Yes)

    = 3/10* 0 * 2/3 * 0.01 = 0
  - P(C = No|X) = P(C = No)*P(Refund = Yes |C = No)

    *P(Status = Single |C = No)

    *P(Income =80K |C= No)

    = 7/10 * 3/7 * 2/7 * 0.0062 = 0.0005

Example of Naïve Bayes Classifier

- Creating a Naïve Bayes Classifier, essentially means to compute counts:

Total number of records: N = 10

Class No:
Number of records: 7
Attribute Refund:
Yes: 3
No: 4
Attribute Marital Status:
Single: 2
Divorced: 1
Married: 4
Attribute Income:
mean: 110
variance: 2975

Class Yes:
Number of records: 3
Attribute Refund:
Yes: 0
No: 3
Attribute Marital Status:
Single: 2
Divorced: 1
Married: 0
Attribute Income:
mean: 90
variance: 25

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1

P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

Example of Naïve Bayes Classifier

Given a Test Record:

X = (Refund = Yes, Status = Single, Income =80K)

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1

P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:

If class=No: sample mean=110
sample variance=2975

If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=Yes|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 3/7 * 2/7 * 0.0062 = 0.00075

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 0 * 2/3 * 0.01 = 0

- P(No) = 0.3, P(Yes) = 0.7

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero

- Laplace Smoothing:

$$P(A_i = a|C = c) = \frac{N_{ac} + 1}{N_c + N_i}$$

  - N_i: number of attribute values for attribute A_i

Example of Naïve Bayes Classifier

- Creating a Naïve Bayes Classifier, essentially means to compute counts:

With Laplace Smoothing

Total number of records: N = 10

Class No:
Number of records: 7
Attribute Refund:
Yes: 3
No: 4
Attribute Marital Status:
Single: 2
Divorced: 1
Married: 4
Attribute Income:
mean: 110
variance: 2975

Class Yes:
Number of records: 3
Attribute Refund:
Yes: 0
No: 3
Attribute Marital Status:
Single: 2
Divorced: 1
Married: 0
Attribute Income:
mean: 90
variance: 25

naive Bayes Classifier:

P(Refund=Yes|No) = 4/9
P(Refund=No|No) = 5/9
P(Refund=Yes|Yes) = 1/5
P(Refund=No|Yes) = 4/5

P(Marital Status=Single|No) = 3/10
P(Marital Status=Divorced|No)=2/10
P(Marital Status=Married|No) = 5/10
P(Marital Status=Single|Yes) = 3/6
P(Marital Status=Divorced|Yes)=2/6
P(Marital Status=Married|Yes) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

Example of Naïve Bayes Classifier

With Laplace Smoothing

Given a Test Record:

X = (Refund = Yes, Status = Single, Income =80K)

naive Bayes Classifier:

P(Refund=Yes|No) = 4/9
P(Refund=No|No) = 5/9
P(Refund=Yes|Yes) = 1/5
P(Refund=No|Yes) = 4/5

P(Marital Status=Single|No) = 3/10
P(Marital Status=Divorced|No)=2/10
P(Marital Status=Married|No) = 5/10
P(Marital Status=Single|Yes) = 3/6
P(Marital Status=Divorced|Yes)=2/6
P(Marital Status=Married|Yes) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) $\times$ P(Married| Class=No) $\times$ P(Income=120K| Class=No) = 4/9 $\times$ 3/10 $\times$ 0.0062 = 0.00082
- P(X|Class=Yes) = P(Refund=No| Class=Yes) $\times$ P(Married| Class=Yes) $\times$ P(Income=120K| Class=Yes) = 1/5 $\times$ 3/6 $\times$ 0.01 = 0.001

- P(No) = 0.7, P(Yes) = 0.3
- P(X|No)P(No) = 0.0005
- P(X|Yes)P(Yes) = 0.0003

=> Class = No

Implementation details

- Computing the conditional probabilities involves multiplication of many very small numbers
  - Numbers get very close to zero, and there is a danger of numeric instability
- We can deal with this by computing the logarithm of the conditional probability

$$\log P(C|A) \sim \log P(A|C) + \log P(C)$$
$$= \sum_i \log P(A_i|C) + \log P(C)$$

Naïve Bayes for Text Classification

- Naïve Bayes is commonly used for text classification
- For a document with k terms $d = (t_1, \dots, t_k)$

$$P(c|d) = P(c)P(d|c) = P(c)\prod_{t_i \in d} P(t_i|c)$$

Fraction of documents in c

- $P(t_i|c)$= Fraction of terms from all documents in c that are t_i.

$$P(t_i|c) = \frac{N_{ic} + 1}{N_c + T}$$

Number of times t_i appears in all documents in c

Laplace Smoothing

Number of unique words (vocabulary size)

Total number of terms in all documents in c

- Easy to implement and works relatively well
- Limitation: Hard to incorporate additional features (beyond words).
  - E.g., number of adjectives used.

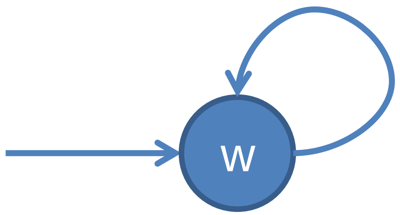
Multinomial document model

- Probability of document $d = (t_1, \dots, t_k)$ in class c:

$$P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)$$

- This formula assumes a multinomial distribution for the document generation:
  - If we have probabilities $p_1, \dots, p_T$ for events $t_1, \dots, t_T$ the probability of a subset of these is

$$P(d) = \frac{N}{N_{t_1}!\, N_{t_2}! \cdots N_{t_T}!} p_1^{N_{t_1}} p_2^{N_{t_2}} \cdots p_T^{N_{t_T}}$$



- Equivalently: There is an automaton spitting words from the above distribution

```
TrainMultinomialNB(ℂ, 𝔻)
V ← ExtractVocabulary(𝔻)
N ← CountDocs(𝔻)
for each c ∈ ℂ
do N_c ← CountDocsInClass(𝔻, c)
    prior[c] ← N_c / N
    text_c ← ConcatenateTextOfAllDocsInClass(𝔻, c)
    for each t ∈ V
    do T_ct ← CountTokensOfTerm(text_c, t)
    for each t ∈ V
    do condprob[t][c] ← (T_ct + 1) / Σ_t′ (T_ct′ + 1)
return V, prior, condprob

ApplyMultinomialNB(ℂ, V, prior, condprob, d)
W ← ExtractTokensFromDoc(V, d)
for each c ∈ ℂ
do score[c] ← log prior[c]
    for each t ∈ W
    do score[c] += log condprob[t][c]
return arg max_{c∈ℂ} score[c]
```

► **Figure 13.2** Naive Bayes algorithm (multinomial model): Training and testing.

Example

News titles for Politics and Sports

	Politics	Sports
documents	"Obama meets Merkel" "Obama elected again" "Merkel visits Greece again"	"OSFP European basketball champion" "Miami NBA basketball champion" "Greece basketball coach?"
	P(p) = 0.5	P(s) = 0.5
terms Vocabulary size: 14	obama:2, meets:1, merkel:2, elected:1, again:2, visits:1, greece:1	OSFP:1, european:1, basketball:3, champion:2, miami:1, nba:1, greece:1, coach:1
	Total terms: 10	Total terms: 11

New title: X = "Obama likes basketball"

P(Politics|X) ~ P(p)*P(obama|p)*P(likes|p)*P(basketball|p)
= 0.5 * 3/(10+14) *1/(10+14) * 1/(10+14) = 0.000108

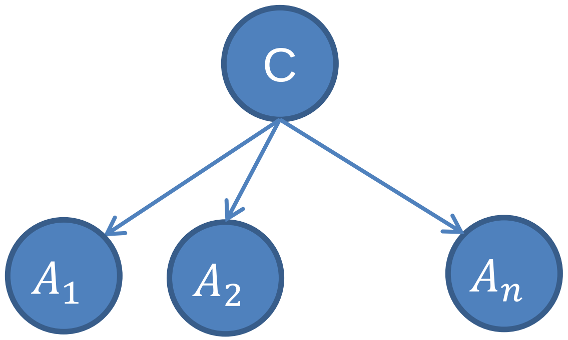
P(Sports|X) ~ P(s)*P(obama|s)*P(likes|s)*P(basketball|s)
= 0.5 * 1/(11+14) *1/(11+14) * 4/(11+14) = 0.000128

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)
- Naïve Bayes can produce a probability estimate, but it is usually a very biased one
  - Logistic Regression is better for obtaining probabilities.

Generative vs Discriminative models

- Naïve Bayes is a type of a generative model
  - Generative process:
    - First pick the category of the record
    - Then given the category, generate the attribute values from the distribution of the category
    - Conditional independence given C



- We use the training data to learn the distributions most likely to have generated the data

Generative vs Discriminative models

- Logistic Regression and SVM are discriminative models
  - The goal is to find the boundary that discriminates between the two classes from the training data

- In order to classify the language of a document, you can
  - Either learn the two languages and find which is more likely to have generated the words you see
  - Or learn what differentiates the two languages.