Data Mining
Classification: Alternative Techniques

Lecture Notes for Chapter 4

Instance-Based Learning

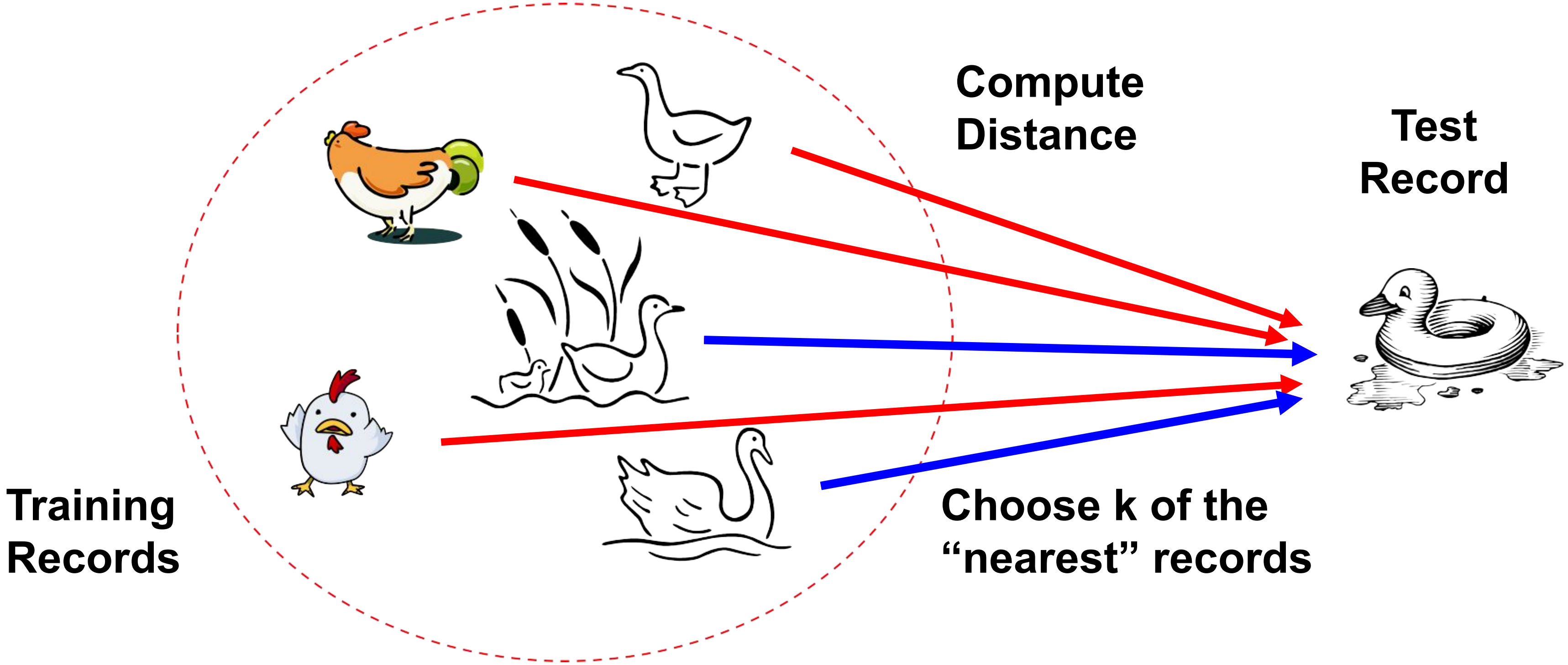
Introduction to Data Mining , 2nd Edition

by

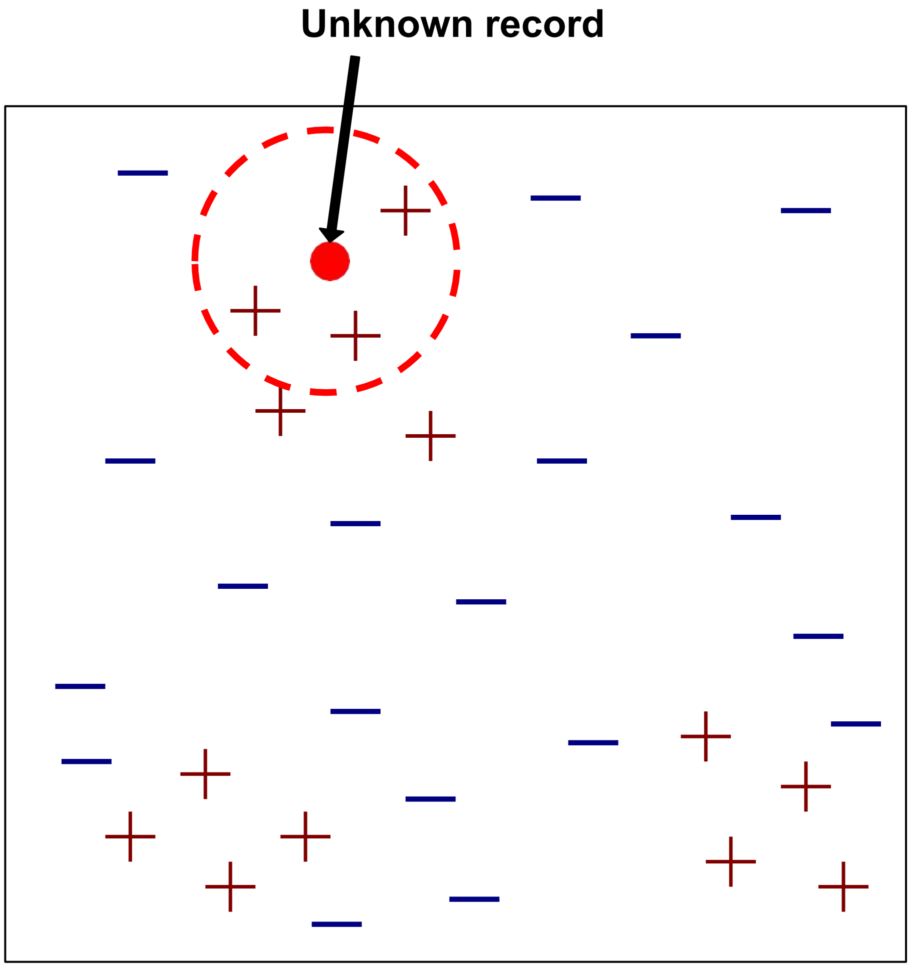
Tan, Steinbach, Karpatne, Kumar

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 the following:
  - A set of labeled records
  - Proximity metric to compute distance/similarity between a pair of records
    - e.g., Euclidean distance
  - The value of *k*, the number of nearest neighbors to retrieve
  - A method for using class labels of K nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

How to Determine the class label of a Test Sample?

- Take the majority vote of class labels among the k-nearest neighbors
- Weight the vote according to distance
  - weight factor, $w = 1/d^2$

Choice of proximity measure matters

- For documents, cosine is better than correlation or Euclidean

1 1 1 1 1 1 1 1 1 1 1 0		0 0 0 0 0 0 0 0 0 0 0 1
	vs	
0 1 1 1 1 1 1 1 1 1 1 1		1 0 0 0 0 0 0 0 0 0 0 0

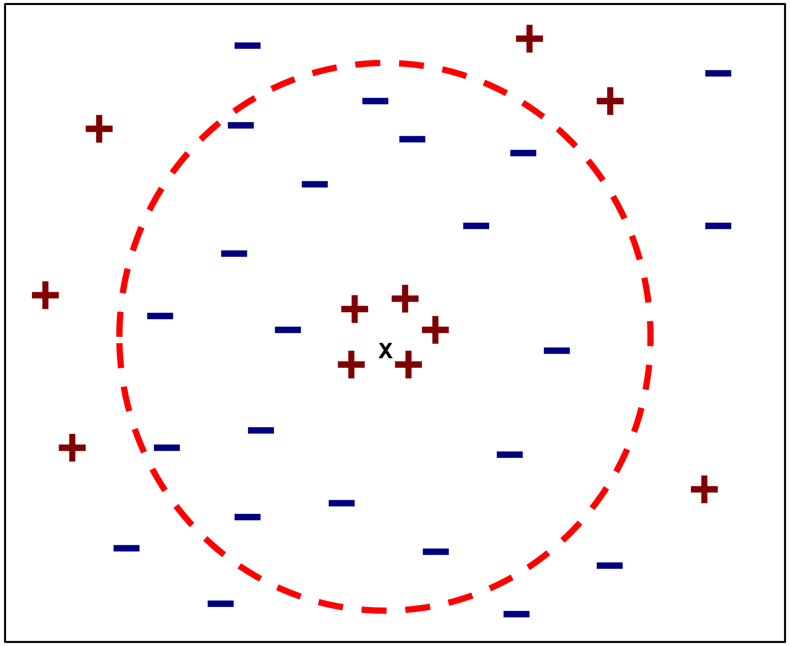
Euclidean distance = 1.4142 for both pairs, but the cosine similarity measure has different values for these pairs.

Nearest Neighbor Classification…

- **Data preprocessing is often required**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
    - Example:
      - height of a person may vary from 1.5m to 1.8m
      - weight of a person may vary from 90lb to 300lb
      - income of a person may vary from $10K to $1M

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



Data Mining
Classification: Alternative Techniques

Bayesian Classifiers

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)}$$

$$P(X \mid Y) = \frac{P(X,Y)}{P(Y)}$$

- Bayes theorem:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes (X_1, X_2,…, X_d), the goal is to predict class Y
  - Specifically, we want to find the value of Y that maximizes P(Y| X_1, X_2,…, X_d)
- Can we estimate P(Y| X_1, X_2,…, X_d) directly 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**

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No, Divorced, Income} = 120\text{K})$$

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**

- We need to estimate

  P(Evade = Yes | X) and P(Evade = No | X)

  In the following we will replace

  Evade = Yes by Yes, and

  Evade = No by No

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

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**

Using Bayes Theorem:

- $P(\text{Yes} \mid X) = \dfrac{P(X \mid \text{Yes})P(\text{Yes})}{P(X)}$
- $P(\text{No} \mid X) = \dfrac{P(X \mid \text{No})P(\text{No})}{P(X)}$
- How to estimate $P(X \mid \text{Yes})$ and $P(X \mid \text{No})$?

Naïve Bayes Classifier

- Assume independence among attributes X_i when class is given:
  - $P(X_1, X_2, \ldots, X_d | Y_j) = P(X_1| Y_j)\, P(X_2| Y_j) \ldots P(X_d| Y_j)$
  - Now we can estimate $P(X_i| Y_j)$ for all X_i and Y_j combinations from the training data
  - New point is classified to Y_j if $P(Y_j)\, \Pi\, P(X_i| Y_j)$ is maximal.

Naïve Bayes on Example Data

$$P(Yes \mid X) = \frac{P(X \mid Yes)P(Yes)}{P(X)}$$

$$P(No \mid X) = \frac{P(X \mid No)P(No)}{P(X)}$$

$$P(X_1, X_2, \ldots, X_d \mid Y_j) = P(X_1 \mid Y_j)\, P(X_2 \mid Y_j) \ldots P(X_d \mid Y_j)$$

Given a Test Record:

$$X = (\text{Refund} = \text{No, Divorced, Income} = 120\text{K})$$

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**

P(X | Yes) =
P(Refund = No | Yes) x
P(Divorced | Yes) x
P(Income = 120K | Yes)

P(X | No) =
P(Refund = No | No) x
P(Divorced | No) x
P(Income = 120K | No)

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**

- P(y) = fraction of instances of class y
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10
- For categorical attributes:

  $P(X_i = c | y) = n_c / n$

  - where $|X_i = c|$ is number of instances having attribute value $X_i = c$ and belonging to class y
  - Examples:

    P(Status=Married|No) = 4/7

    P(Refund=Yes|Yes)=0

Estimate Probabilities from Data

- For continuous attributes:
  - Discretization: Partition the range into bins:
    - Replace continuous value with bin value
      - Attribute changed from continuous to ordinal
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

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**

- Normal distribution:

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

  – One for each (X_i, Y_i) pair

- For (Income, Class=No):
  – If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No, Divorced, Income} = 120\text{K})$$

Naïve 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) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
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 | No) = P(Refund=No | No) $\times$ P(Divorced | No) $\times$ P(Income=120K | No) $= 4/7 \times 1/7 \times 0.0072 = 0.0006$

- P(X | Yes) = P(Refund=No | Yes) $\times$ P(Divorced | Yes) $\times$ P(Income=120K | Yes) $= 1 \times 1/3 \times 1.2 \times 10^{-9} = 4 \times 10^{-10}$

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

=> Class = No

Naïve Bayes (Summary)

- Robust to isolated noise points

- Handle missing values by ignoring the instance during probability estimate calculations

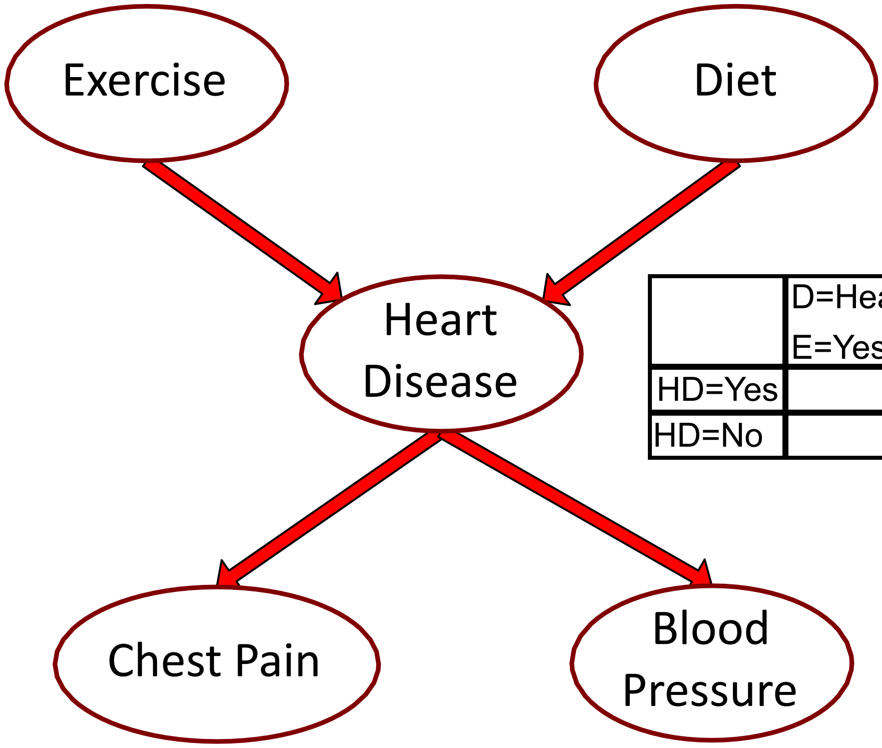
- Robust to irrelevant attributes

- Redundant and correlated attributes will violate class conditional assumption
  - Use other techniques such as Bayesian Belief Networks (BBN)

Example of Bayesian Belief Network

Exercise=Yes	0.7
Exercise=No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	D=Healthy E=Yes	D=Healthy E=No	D=Unhealthy E=Yes	D=Unhealthy E=No
HD=Yes	0.25	0.45	0.55	0.75
HD=No	0.75	0.55	0.45	0.25

	HD=Yes	HD=No
CP=Yes	0.8	0.01
CP=No	0.2	0.99

	HD=Yes	HD=No
BP=High	0.85	0.2
BP=Low	0.15	0.8

Data Mining

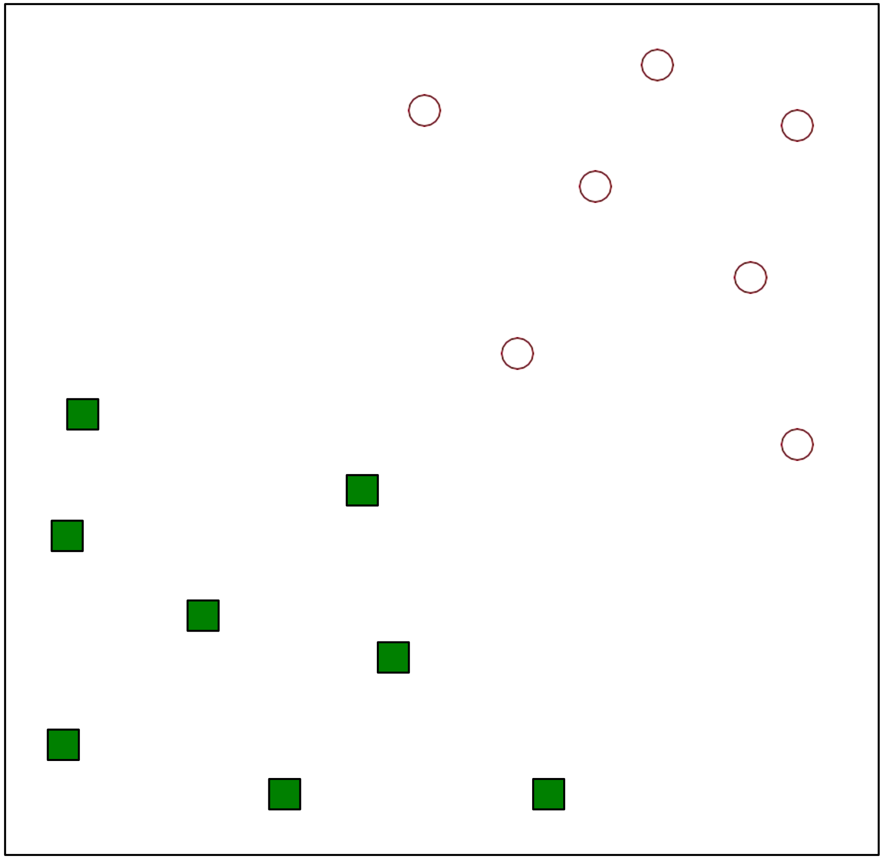
Support Vector Machines

Introduction to Data Mining, 2nd Edition

by

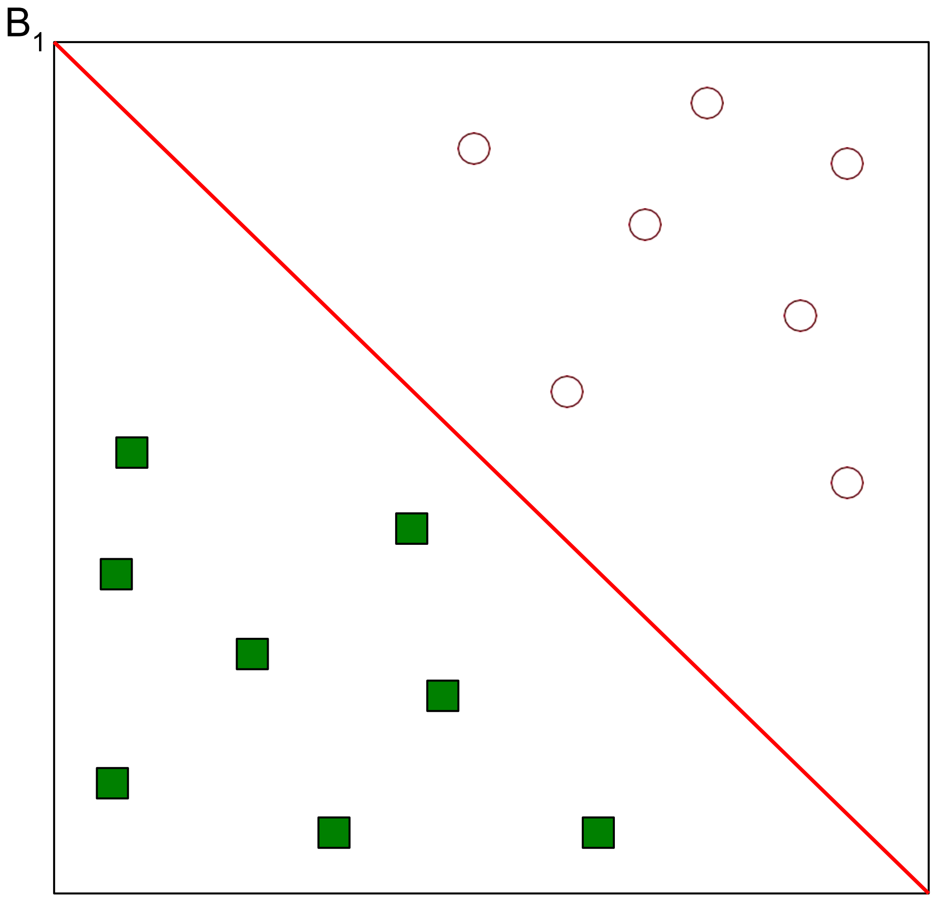
Tan, Steinbach, Karpatne, Kumar

Support Vector Machines



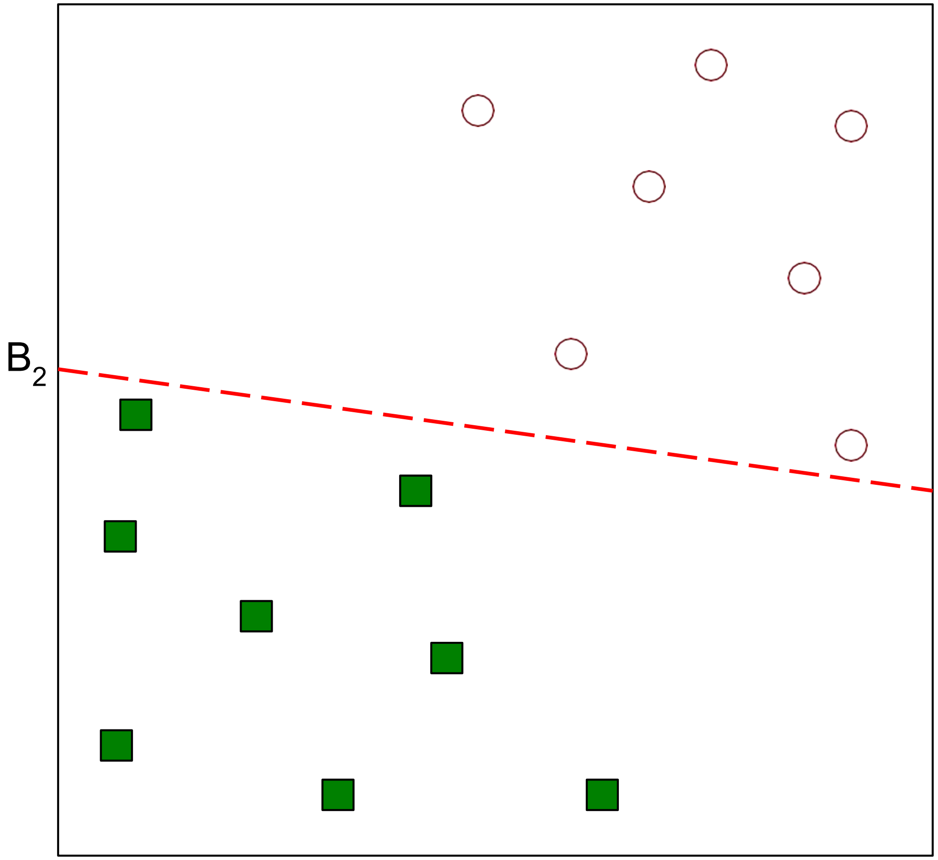
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



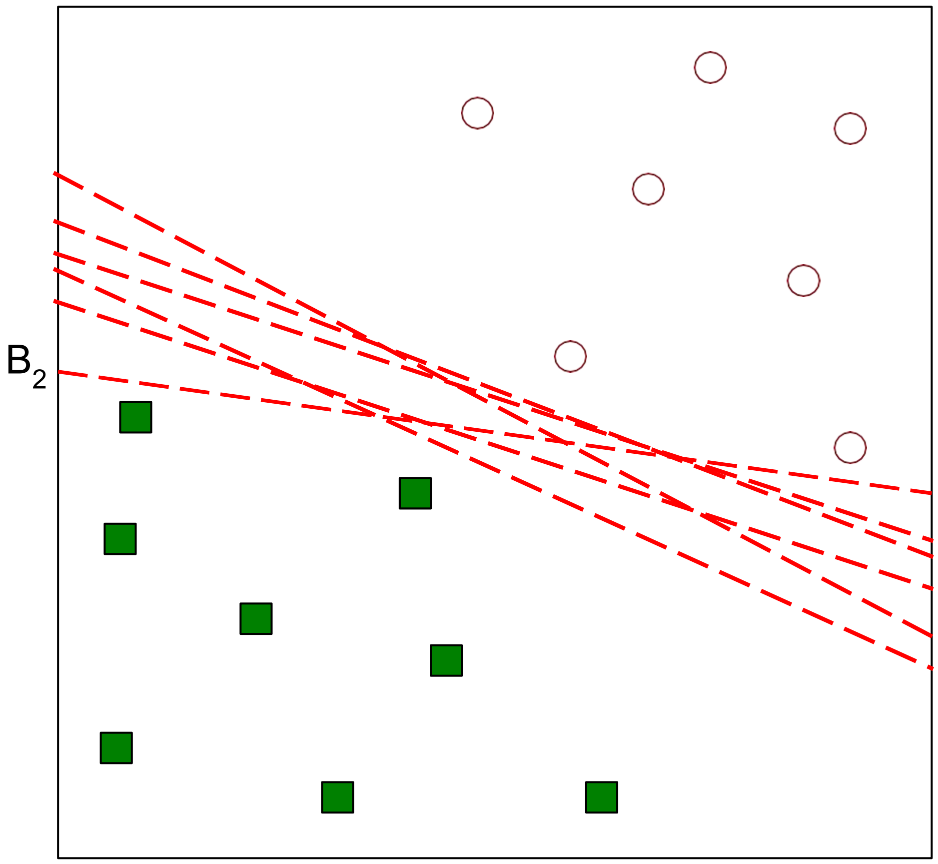
- One Possible Solution

Support Vector Machines



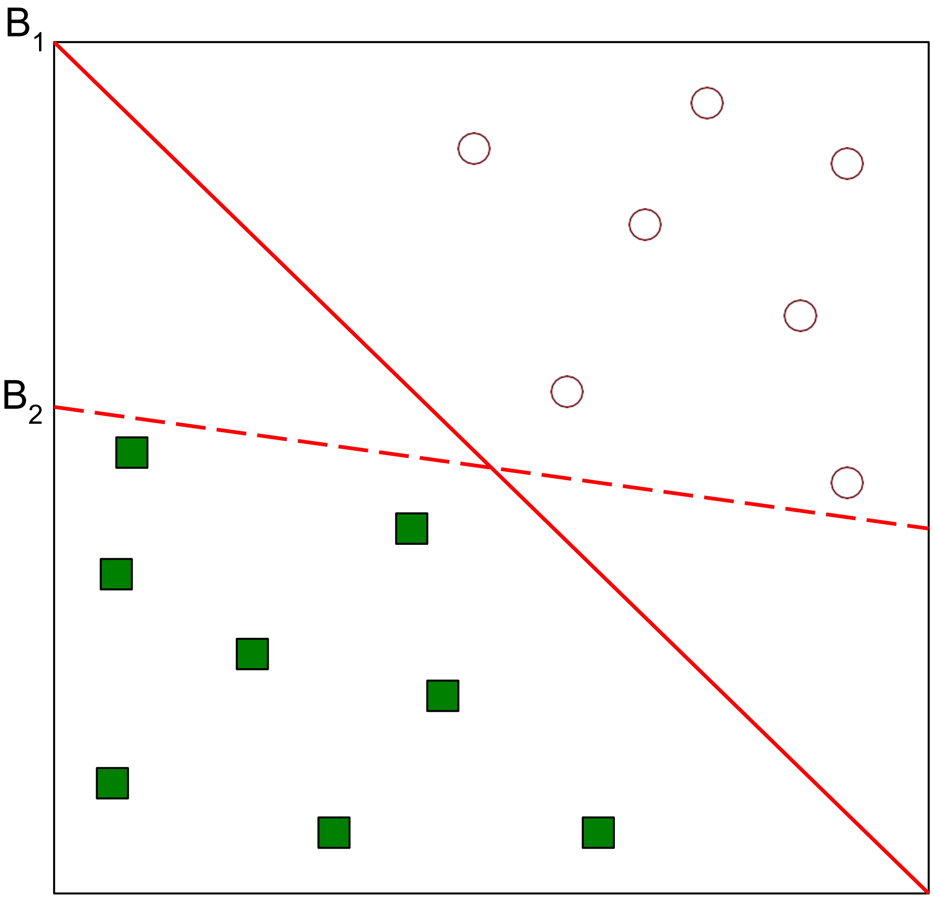
- Another possible solution

Support Vector Machines



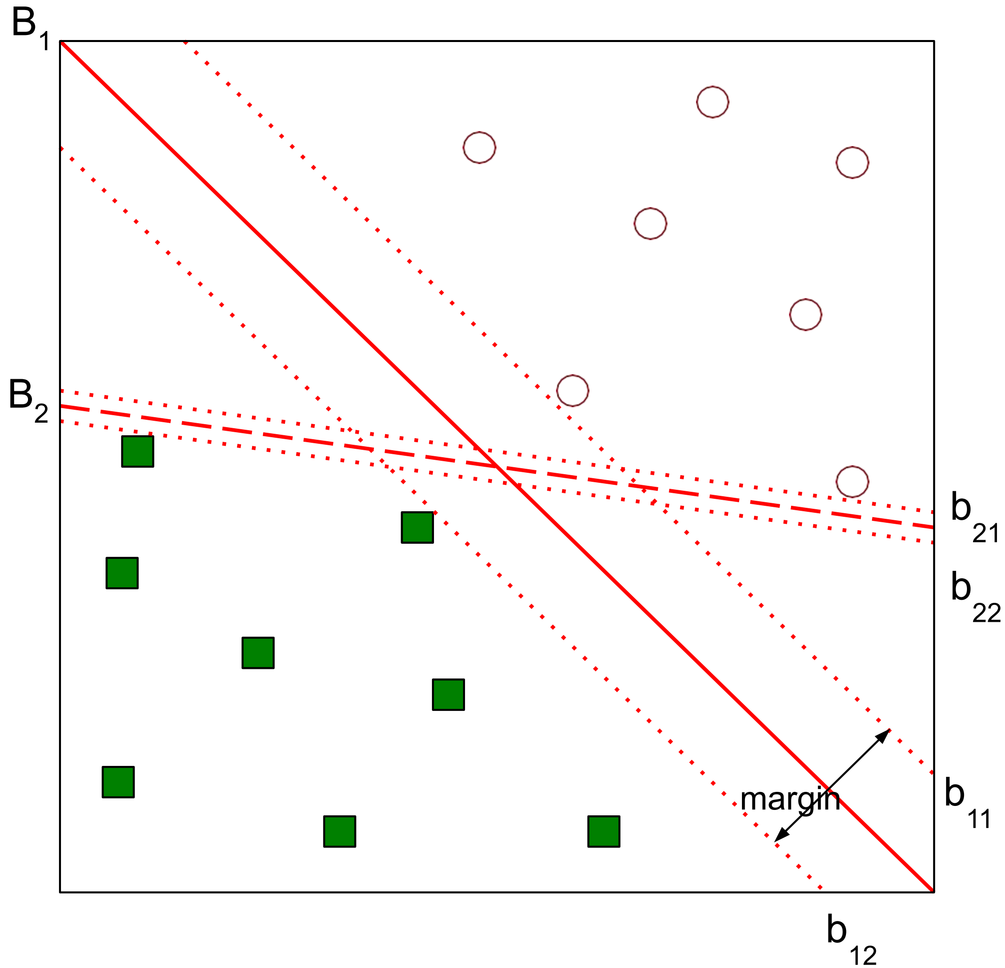
- Other possible solutions

Support Vector Machines



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

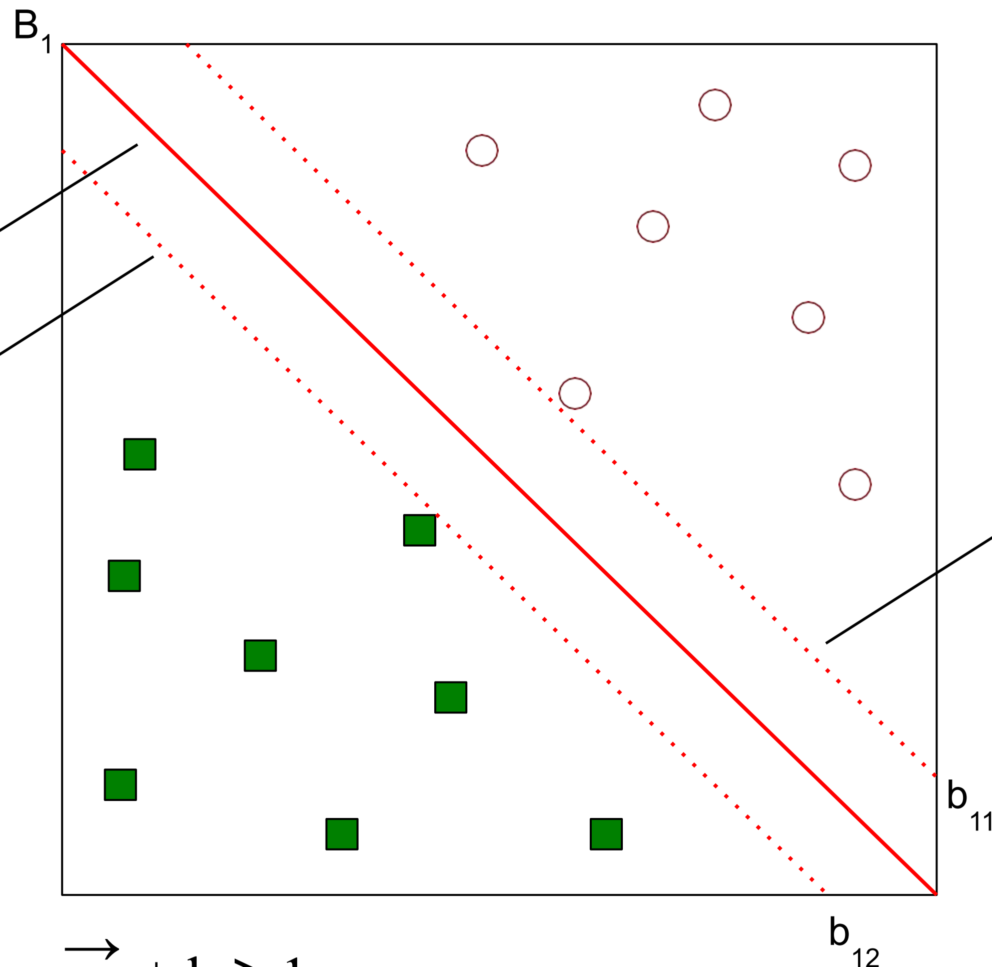
Support Vector Machines



b11 and b12, b21 and b22:
A pair of parallel hyperplanes such that they touch the closest instances of both classes.

- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



$w \bullet x + b = 0$

$w \bullet x + b = -1$

$w \bullet x + b = +1$

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

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

Learning Linear SVM

- Objective is to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|}$
  - Which is equivalent to minimizing: $L(\vec{w}) = \frac{\|\vec{w}\|^2}{2}$
  - Subject to the following constraints:

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

or

$$y_i(\mathrm{w} \bullet \mathrm{x}_i + b) \geq 1,\ i = 1,2,\ldots,N$$

  - This is a constrained optimization problem
    - Solve it using Lagrange multiplier method

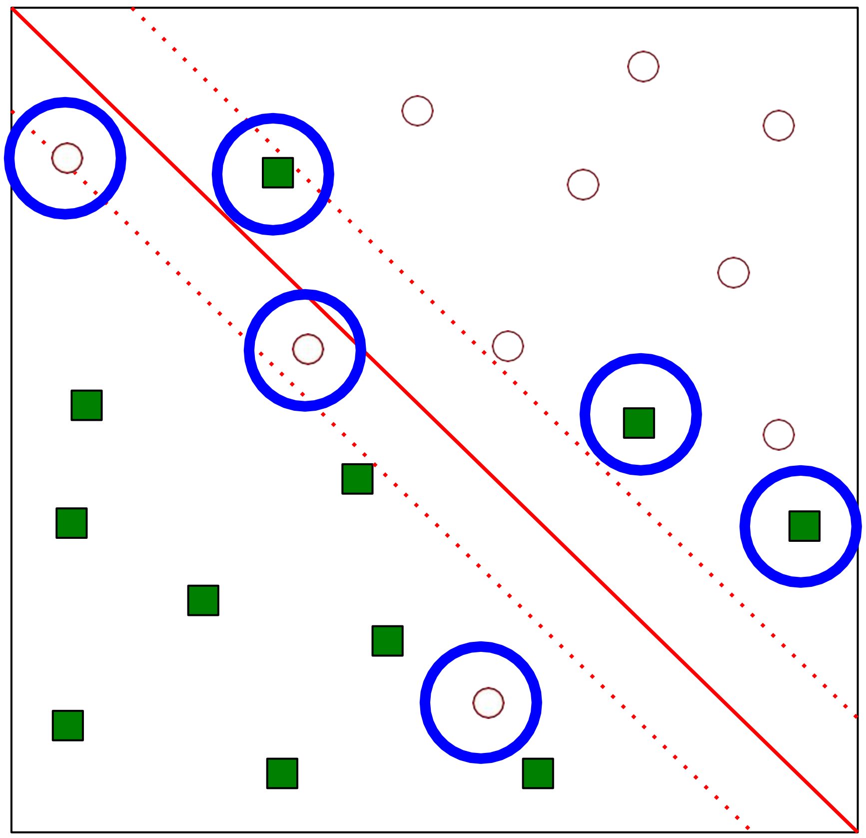
Learning Linear SVM

- Decision boundary depends only on support vectors
  - If you have data set with same support vectors, decision boundary will not change
  - How to classify using SVM once **w** and *b* are found? Given a test record, x_i

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

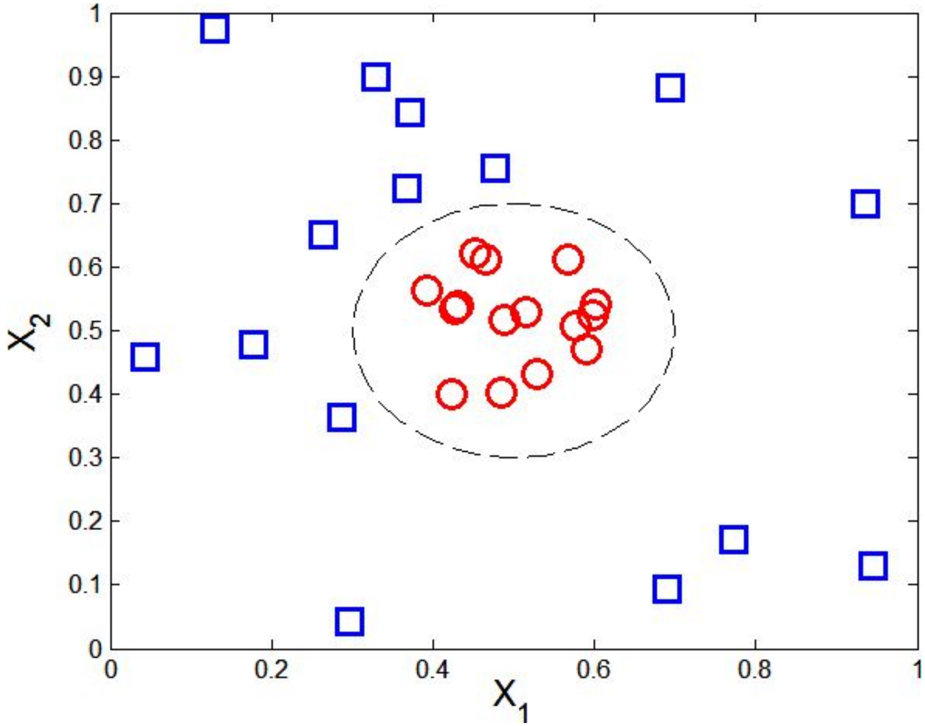
Support Vector Machines

- What if the problem is not linearly separable?



Nonlinear Support Vector Machines

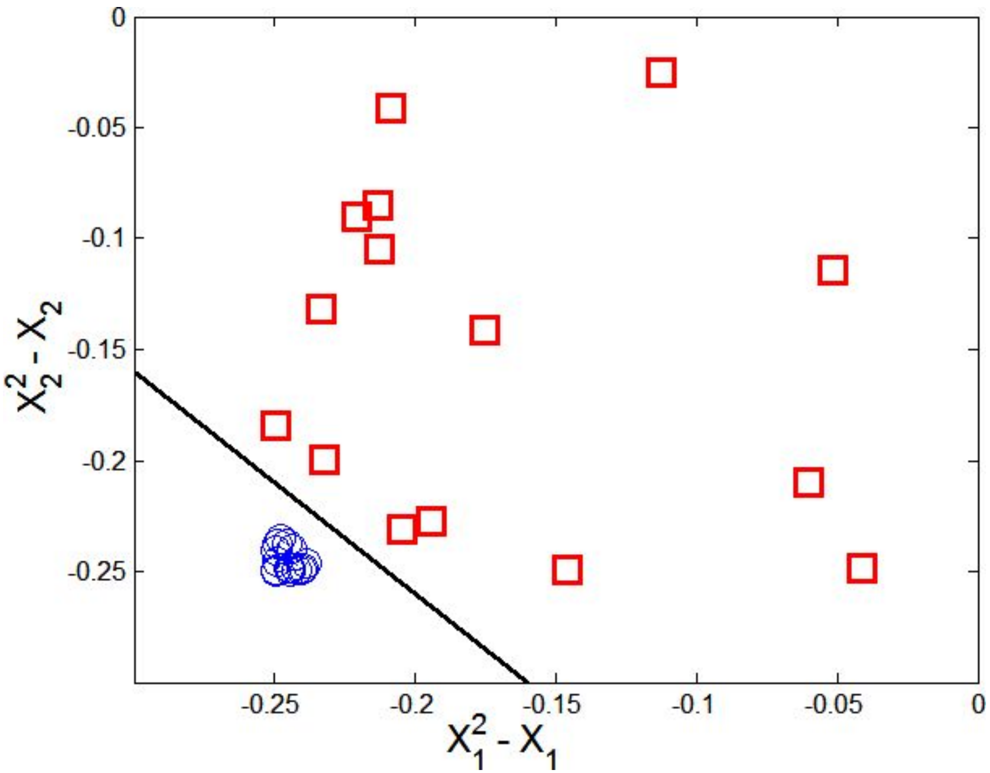
- What if decision boundary is not linear?



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

Nonlinear Support Vector Machines

- Transform data into higher dimensional space



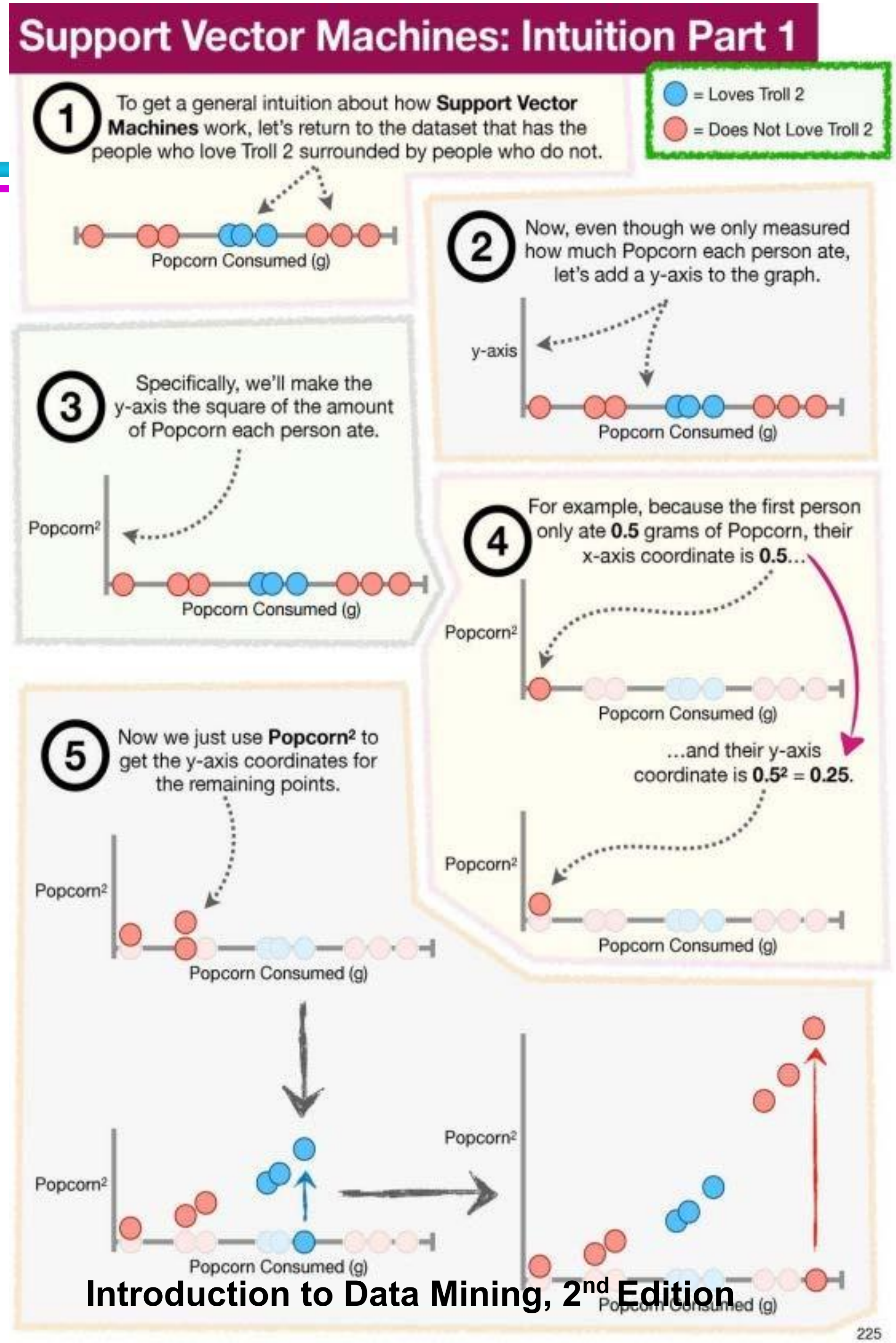
$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

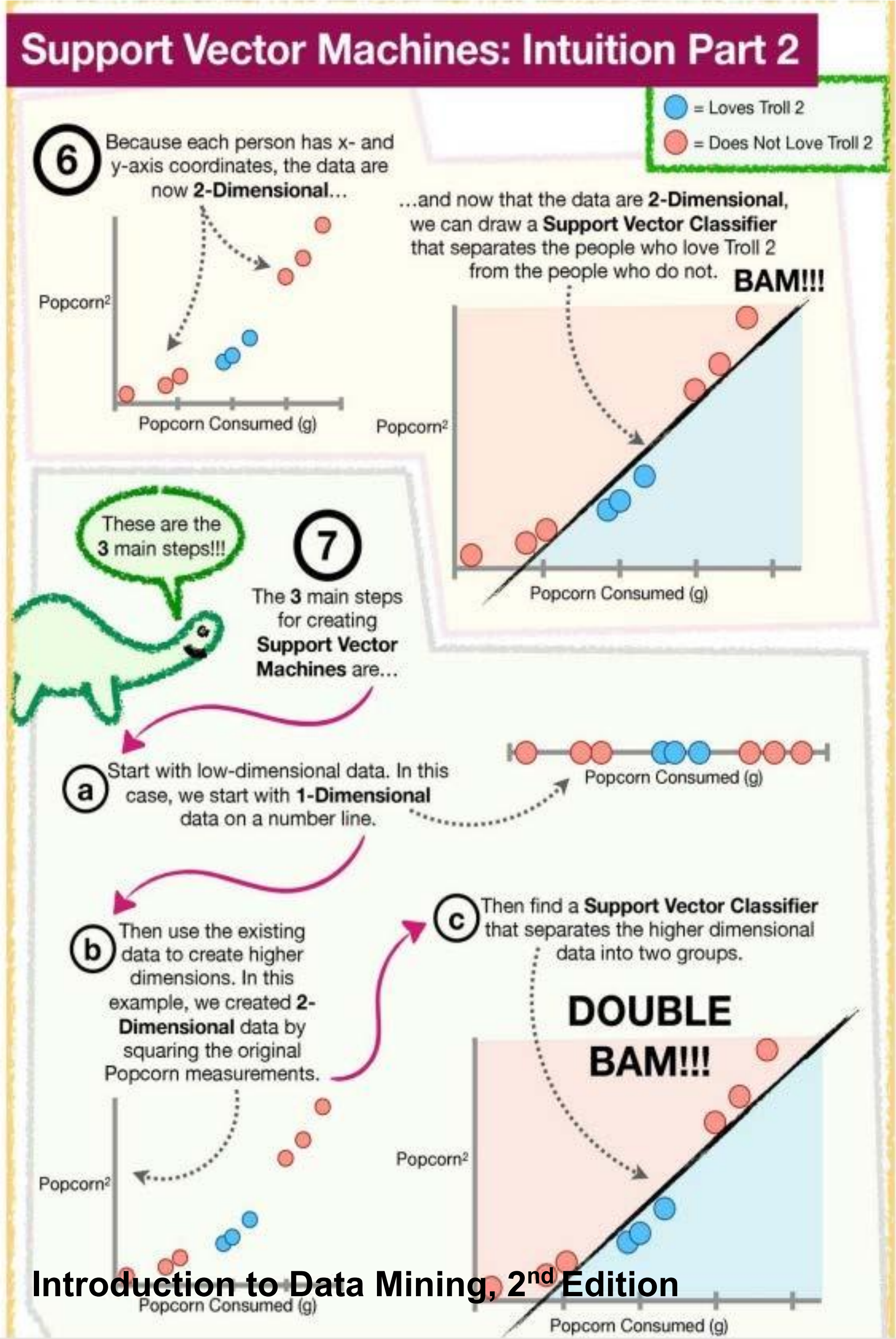
$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

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

Decision boundary:

$$w \bullet \Phi(x) + b = 0$$


Support Vector Machines: Intuition Part 1
= Loves Troll 2
= Does Not Love Troll 2
1
To get a general intuition about how **Support Vector Machines** work, let's return to the dataset that has the people who love Troll 2 surrounded by people who do not.
Popcorn Consumed (g)
2
Now, even though we only measured how much Popcorn each person ate, let's add a y-axis to the graph.
y-axis
Popcorn Consumed (g)
3
Specifically, we'll make the y-axis the square of the amount of Popcorn each person ate.
Popcorn²
Popcorn Consumed (g)
4
For example, because the first person only ate **0.5** grams of Popcorn, their x-axis coordinate is **0.5**…
Popcorn²
Popcorn Consumed (g)
…and their y-axis coordinate is **0.5² = 0.25**.
Popcorn²
Popcorn Consumed (g)
5
Now we just use **Popcorn²** to get the y-axis coordinates for the remaining points.
Popcorn²
Popcorn Consumed (g)
Popcorn²
Popcorn Consumed (g)
Popcorn²
225


Support Vector Machines: Intuition Part 2
= Loves Troll 2
= Does Not Love Troll 2
6
Because each person has x- and y-axis coordinates, the data are now **2-Dimensional**...
Popcorn²
Popcorn Consumed (g)
...and now that the data are **2-Dimensional**, we can draw a **Support Vector Classifier** that separates the people who love Troll 2 from the people who do not.
BAM!!!
Popcorn²
Popcorn Consumed (g)
These are the **3** main steps!!!
7
The **3** main steps for creating **Support Vector Machines** are...
a
Start with low-dimensional data. In this case, we start with **1-Dimensional** data on a number line.
Popcorn Consumed (g)
b
Then use the existing data to create higher dimensions. In this example, we created **2-Dimensional** data by squaring the original Popcorn measurements.
Popcorn²
Popcorn Consumed (g)
c
Then find a **Support Vector Classifier** that separates the higher dimensional data into two groups.
DOUBLE BAM!!!
Popcorn²
Popcorn Consumed (g)

Characteristics of SVM

- A convex optimization problem in which efficient algorithms are available to find the global minimum of the objective function
- Overfitting is handled regularizing the model parameters and maximizing the margin of the decision boundary
- SVM can handle irrelevant and redundant attributes better than many other techniques
- The user needs to provide the type of kernel function and cost function
- Difficult to handle missing values

Data Mining

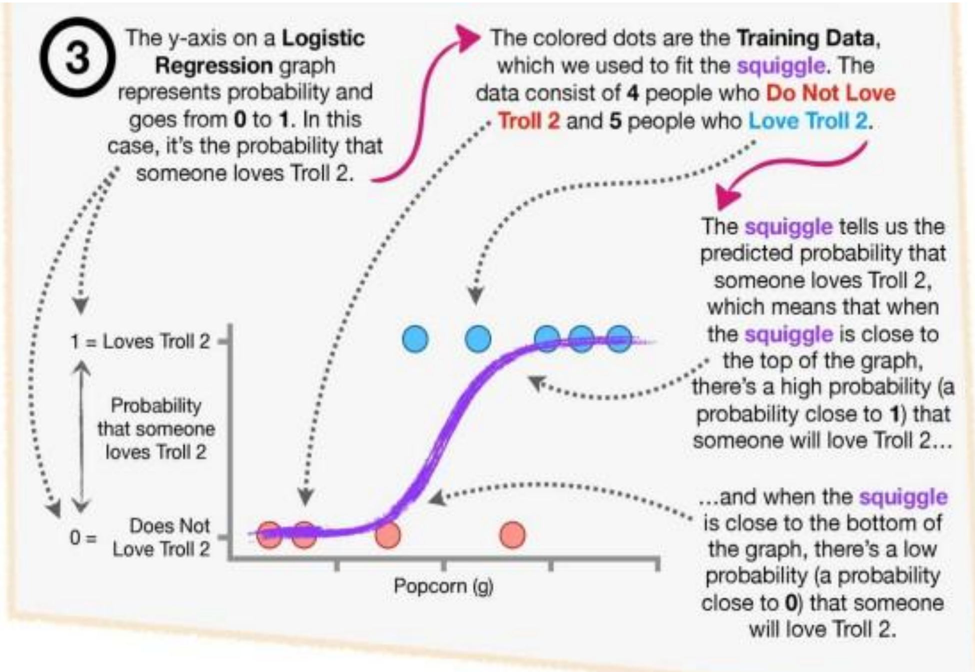
Lecture Notes for Chapter 4
Artificial Neural Networks

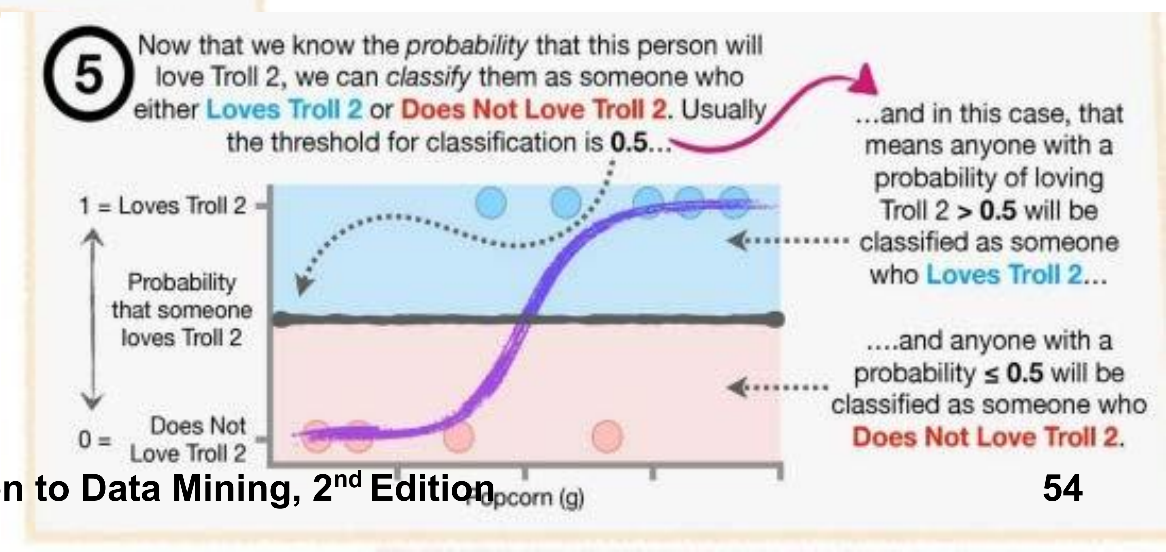
Introduction to Data Mining , 2nd Edition

by

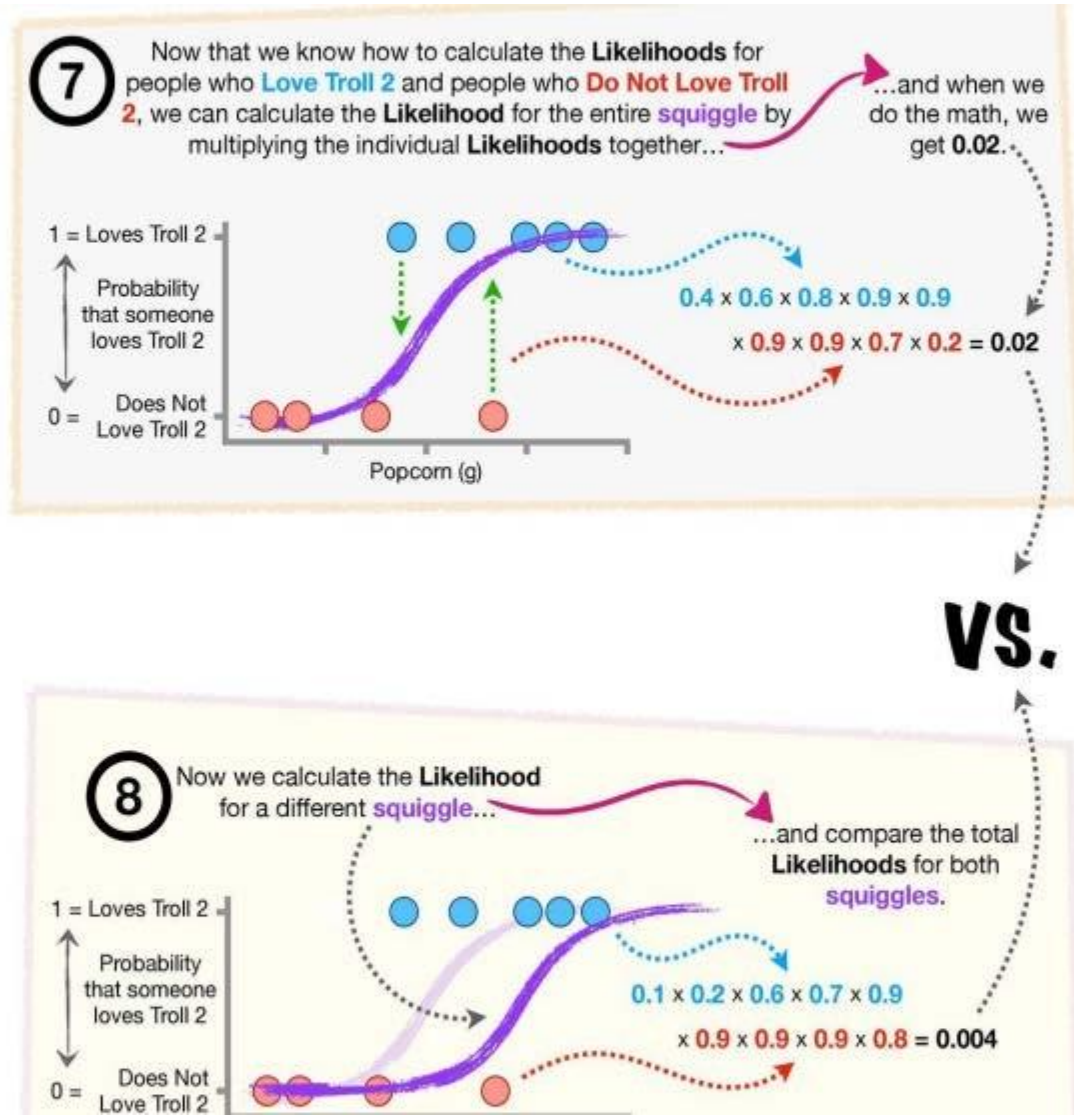
Tan, Steinbach, Karpatne, Kumar

Logistic Regression





Logistic Regression/2



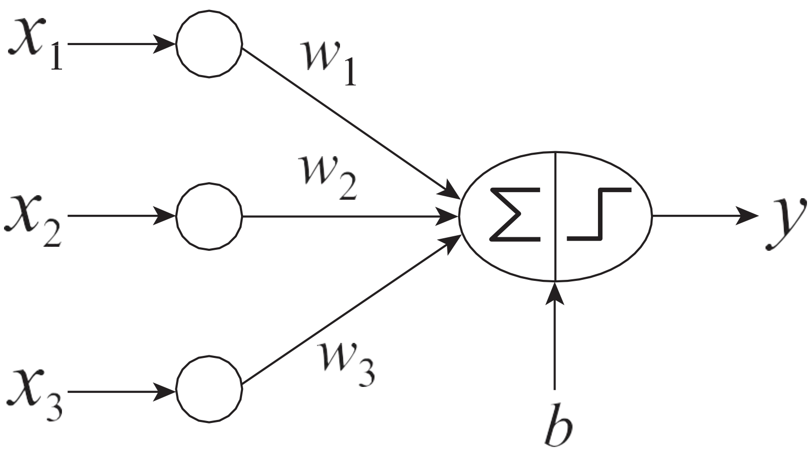
Artificial Neural Networks (ANN)

- **Basic Idea:** A complex non-linear function can be learned as a composition of simple processing units

- ANN is a collection of simple processing units (nodes) that are connected by directed links (edges)
  - Every node receives signals from incoming edges, performs computations, and transmits signals to outgoing edges
  - Analogous to *human brain* where nodes are neurons and signals are electrical impulses
  - Weight of an edge determines the strength of connection between the nodes

- Simplest ANN: **Perceptron** (single neuron)

Basic Architecture of Perceptron



$$y = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x} + b > 0. \\ -1, & \text{otherwise.} \end{cases}$$

$$\tilde{\mathbf{w}} = (\mathbf{w}^T \; b)^T \qquad \tilde{\mathbf{x}} = (\mathbf{x}^T \; 1)^T$$

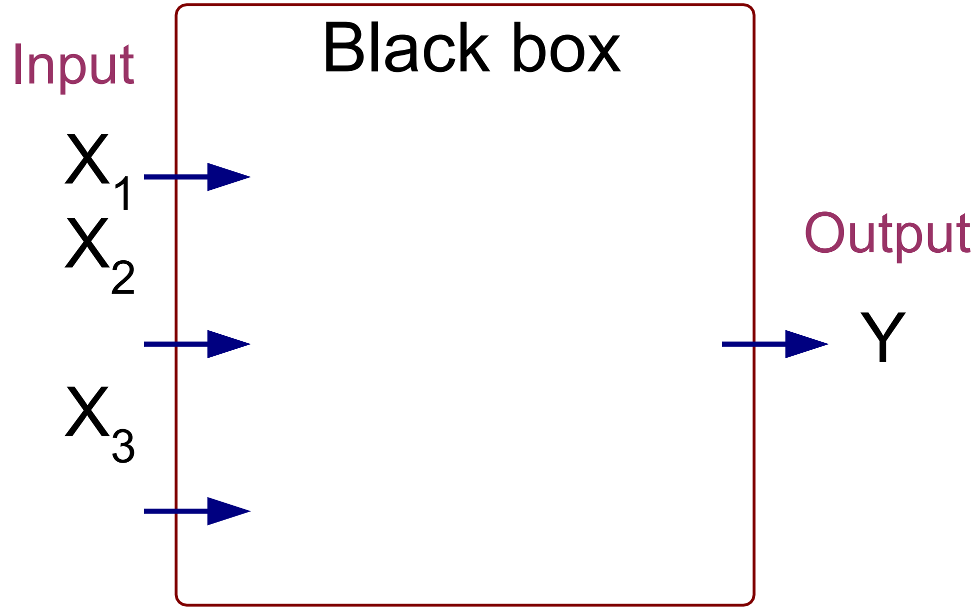
$$\hat{y} = sign(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

Activation Function

- Learns linear decision boundaries
- Related to logistic regression (activation function is sign instead of sigmoid)

Perceptron Example

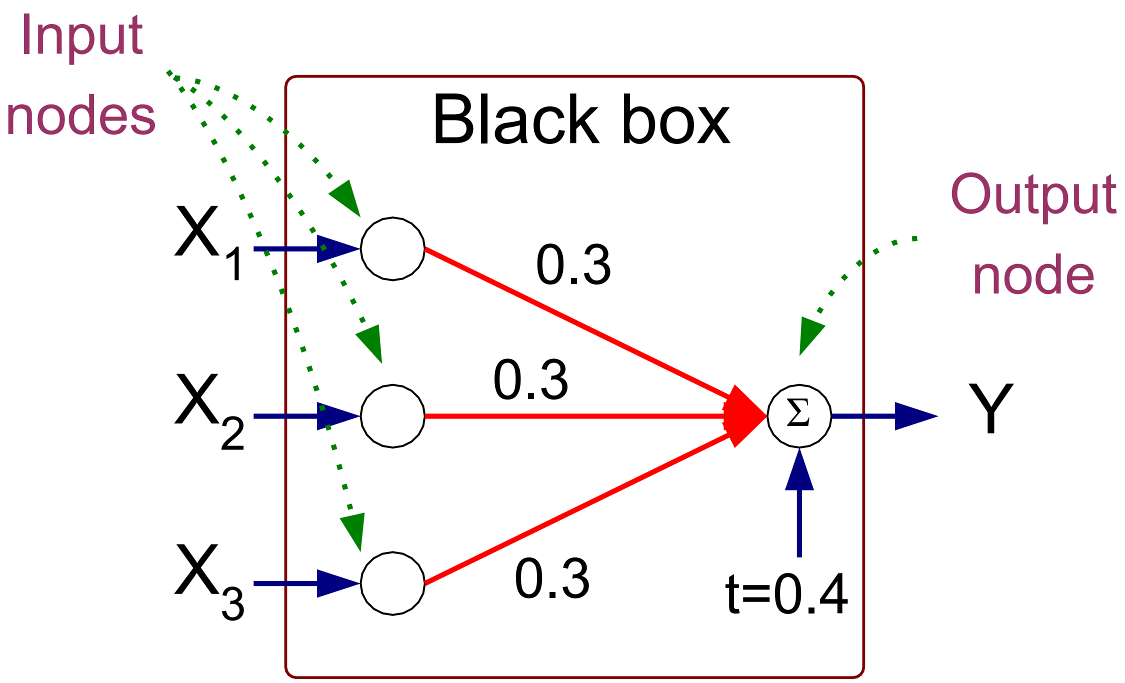
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Perceptron Example

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Perceptron Learning Rule

- Initialize the weights ($w_0, w_1, \ldots, w_d$)
- Repeat
  - For each training example (x_i, y_i)
    - Compute $\hat{y}_i$
    - Update the weights:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda\big(y_i - \hat{y}_i^{(k)}\big)x_{ij}$$

- Until stopping condition is met

- k: iteration number; λ: learning rate

Example of Perceptron Learning

$$\lambda = 0.1$$

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

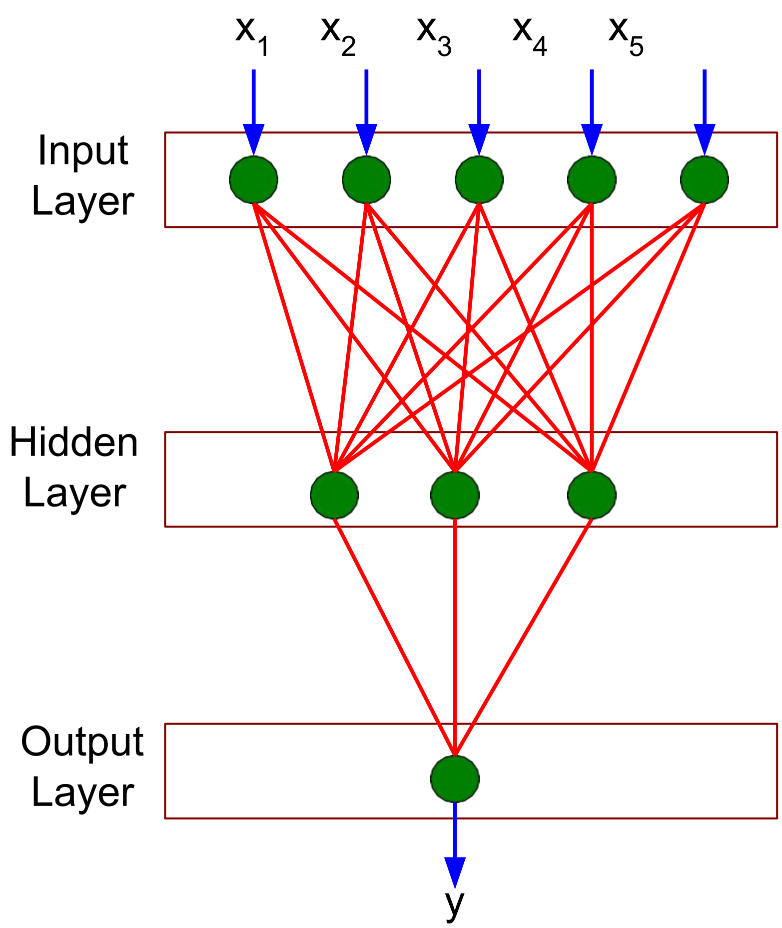
	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Weight updates over first epoch

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

Weight updates over all epochs

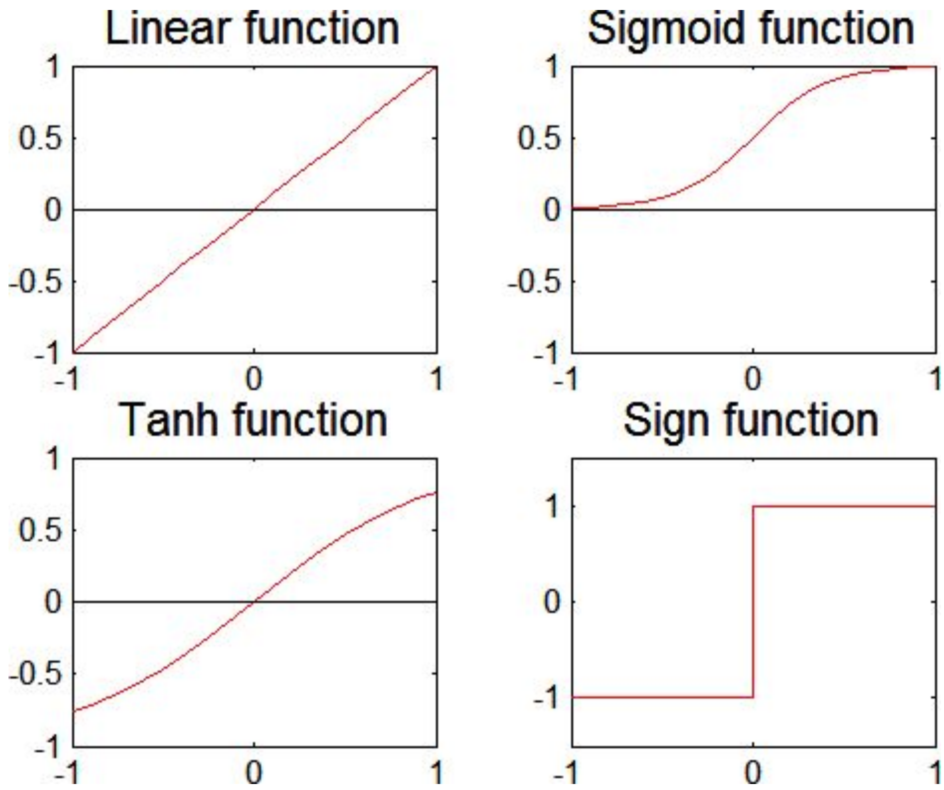
Multi-layer Neural Network



- More than one *hidden layer* of computing nodes
- Every node in a hidden layer operates on activations from preceding layer and transmits activations forward to nodes of next layer
- Also referred to as "feedforward neural networks"

Activation Functions

$$a_i^l = f(z_i^l) = f\big(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\big)$$



$$a_i^l = \sigma(z_i^l) = \frac{1}{1+e^{-z_i^l}}.$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial\ \sigma(z_i^l)}{\partial z_i^l} = a_i^l(1-a_i^l)$$

Deep Learning Trends

- Training **deep** neural networks (more than 5-10 layers) could only be possible in recent times with:
  - Faster computing resources (GPU)
  - Larger labeled training sets
- Algorithmic Improvements in Deep Learning
  - Responsive activation functions (e.g., RELU)
  - Regularization (e.g., Dropout)
  - Supervised pre-training
  - Unsupervised pre-training (auto-encoders)
- Specialized ANN Architectures:
  - Convolutional Neural Networks (for image data)
  - Recurrent Neural Networks (for sequence data)
  - Residual Networks (with skip connections)
- Generative Models: Generative Adversarial Networks