Feature Selection, Advanced Classification, Metrics

CS 584 Data Mining (Fall 2016)

Huzefa Rangwala

Associate Professor,

Computer Science,

George Mason University

Email: rangwala@cs.gmu.edu

Website: www.cs.gmu.edu/~hrangwal

Slides are adapted from the available book slides developed by Tan, Steinbach and Kumar

Lesson Plan

- Curse of Dimensionality
- Metrics
- Naïve Bayes Classifier
- Perceptron
- Support Vector Machines

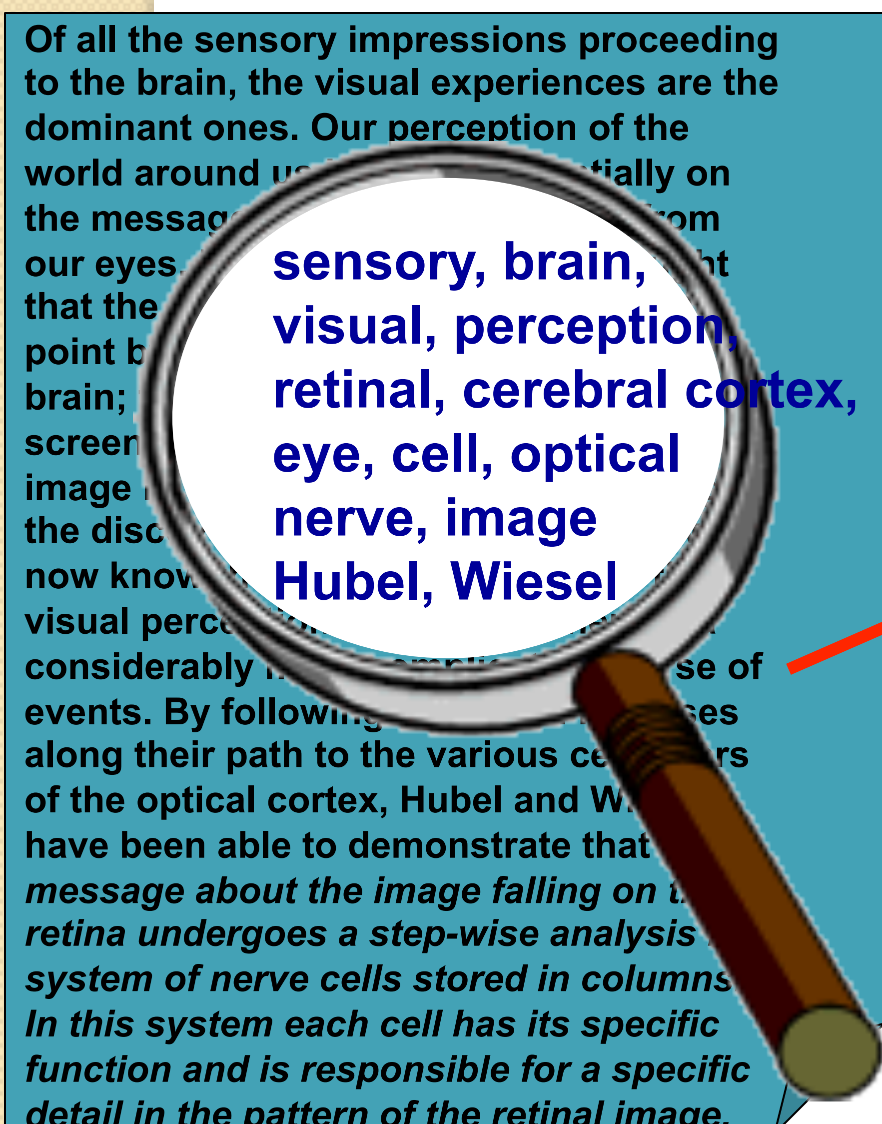
Next Time:

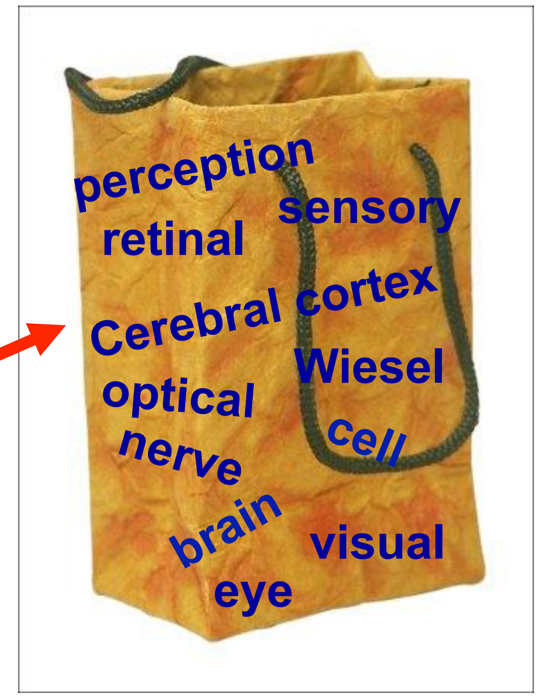
- Decision Trees
- Bayesian Networks
- PCA and Bias-Variance Tradeoff

Curse of Dimensionality

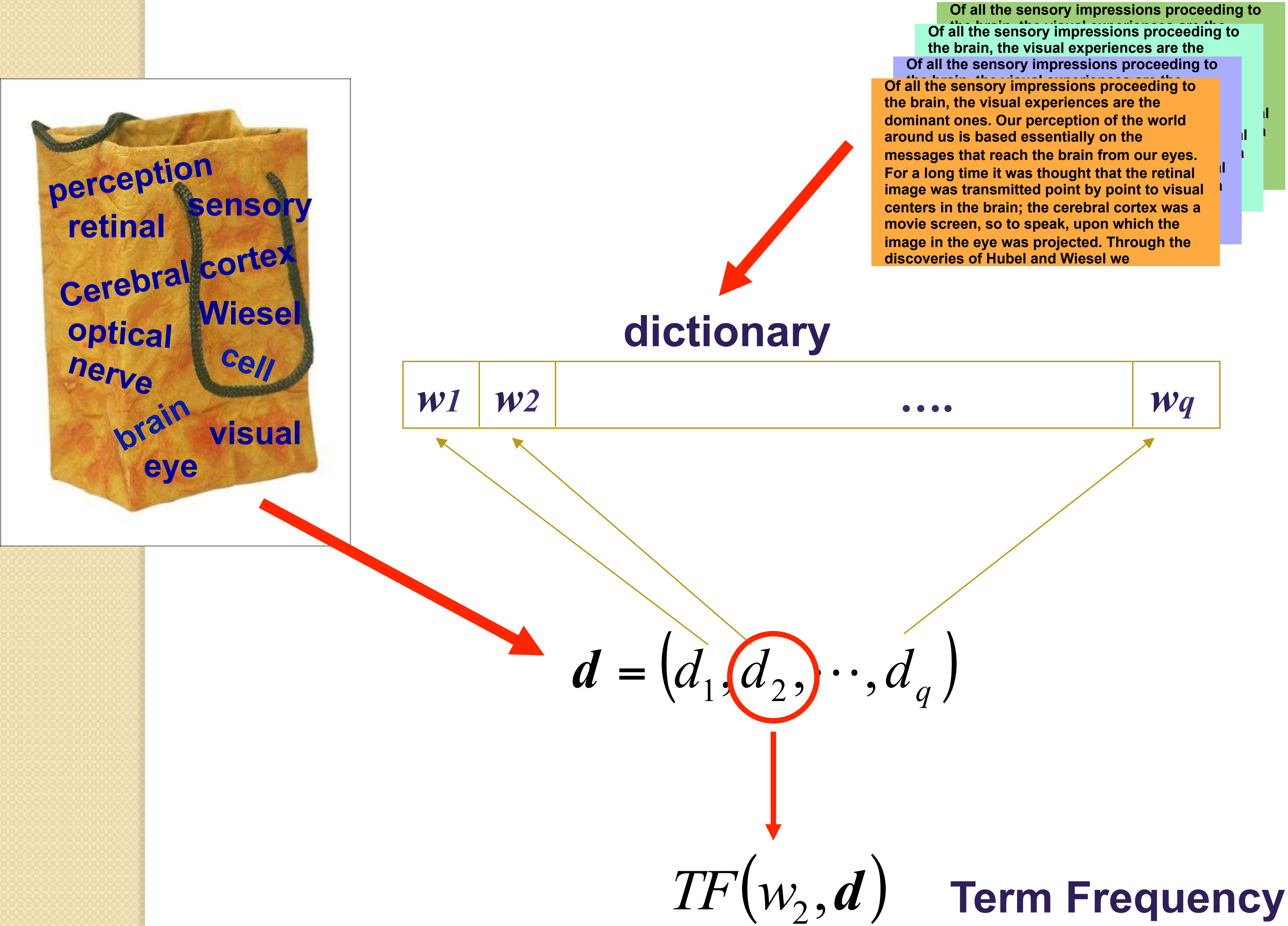
- Many problems of interest have objects with a large number of dimensions.
- An example…

Document Classification





Bag-of-words representation of a document


perception
sensory
retinal
Cerebral cortex
Wiesel
optical
nerve
cell
brain
visual
eye
Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that reach the brain from our eyes. For a long time it was thought that the retinal image was transmitted point by point to visual centers in the brain; the cerebral cortex was a movie screen, so to speak, upon which the image in the eye was projected. Through the discoveries of Hubel and Wiesel we
dictionary
w1
w2
....
wq
$\boldsymbol{d} = (d_1, d_2, \cdots, d_q)$
$TF(w_2, \boldsymbol{d})$
Term Frequency

What’s the size of the dictionary?

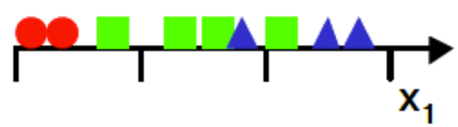
Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies

- Also distances between objects gets skewed
  - More dimensions that contribute to the notion of distance or proximity which makes it uniform. This leads to trouble in clustering and classification settings.

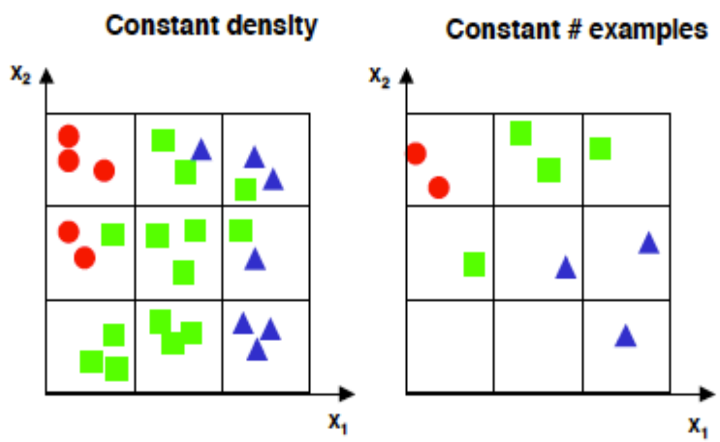
Driving the point ..

- **Consider a 3-class pattern recognition problem**
  - A simple approach would be to
    - Divide the feature space into uniform bins
    - Compute the ratio of examples for each class at each bin and,
    - For a new example, find its bin and choose the predominant class in that bin
  - In our toy problem we decide to start with one single feature and divide the real line into 3 segments

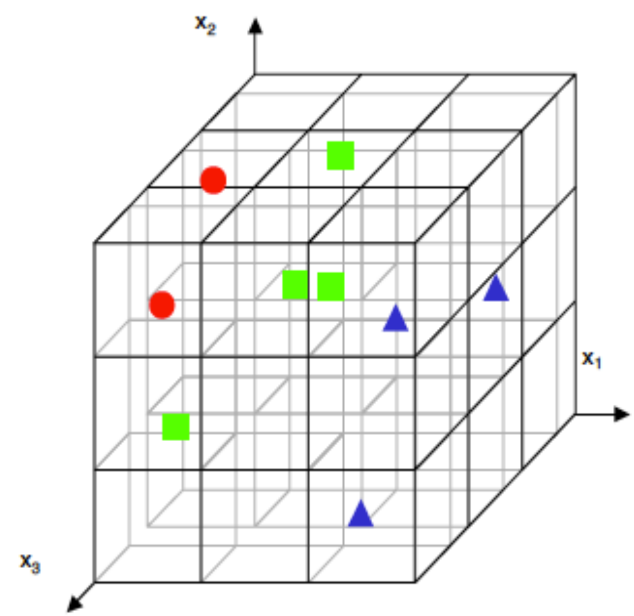


  - After we have done this, we notice that there exists too much overlap for the classes, so we decide to incorporate a second feature to try and improve the classification rate

- **We decide to preserve the granularity of each axis, which raises the number of bins from 3 (in 1D) to 3^2=9 (in 2D)**
  - At this point we are faced with a decision: do we maintain the density of examples per bin or do we keep the number of examples we used for the one-dimensional case?
    - Choosing to maintain the density increases the number of examples from 9 (in 1D) to 27 (in 2D)
    - Choosing to maintain the number of examples results in a 2D scatter plot that is very sparse

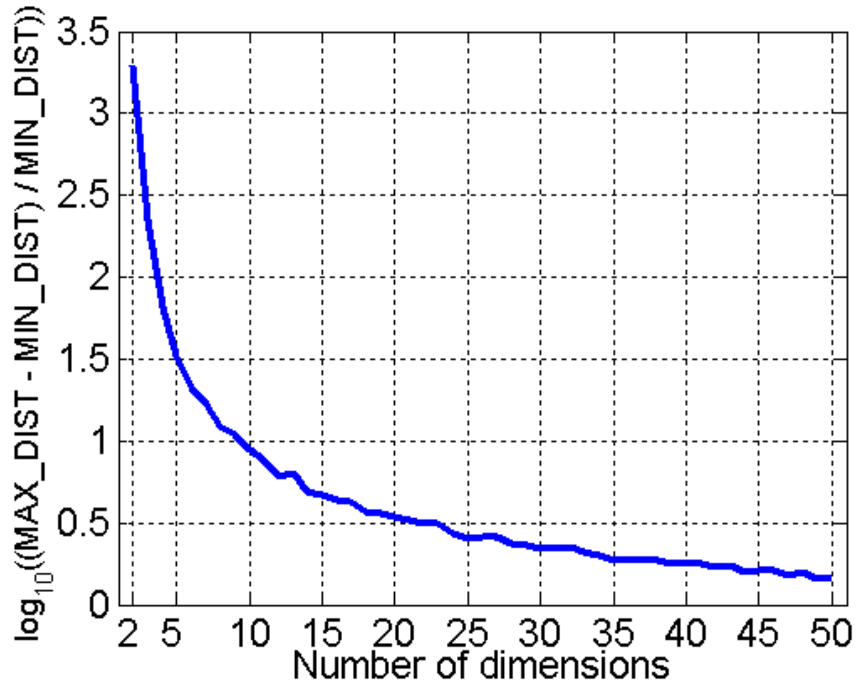


- **Moving to three features makes the problem worse:**
  - The number of bins grows to 3^3=27
  - For the same density of examples the number of needed examples becomes 81
  - For the same number of examples, well, the 3D scatter plot is almost empty



Curse of Dimensionality

- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful



- **Randomly generate 500 points**
- **Compute difference between max and min distance between any pair of points**

Dimensionality Reduction

- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise
- Techniques
  - Principle Component Analysis
  - Singular Value Decomposition
  - Others: supervised and non-linear techniques

Nearest Neighbor Classification…

- Scaling issues
  - 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…

- 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		1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1	vs	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

Model Selection

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

- Most widely-used metric:

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

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

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

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

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

Computing Cost of Classification

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

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

Accuracy = (a + d)/N

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

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

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)

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

Model Evaluation

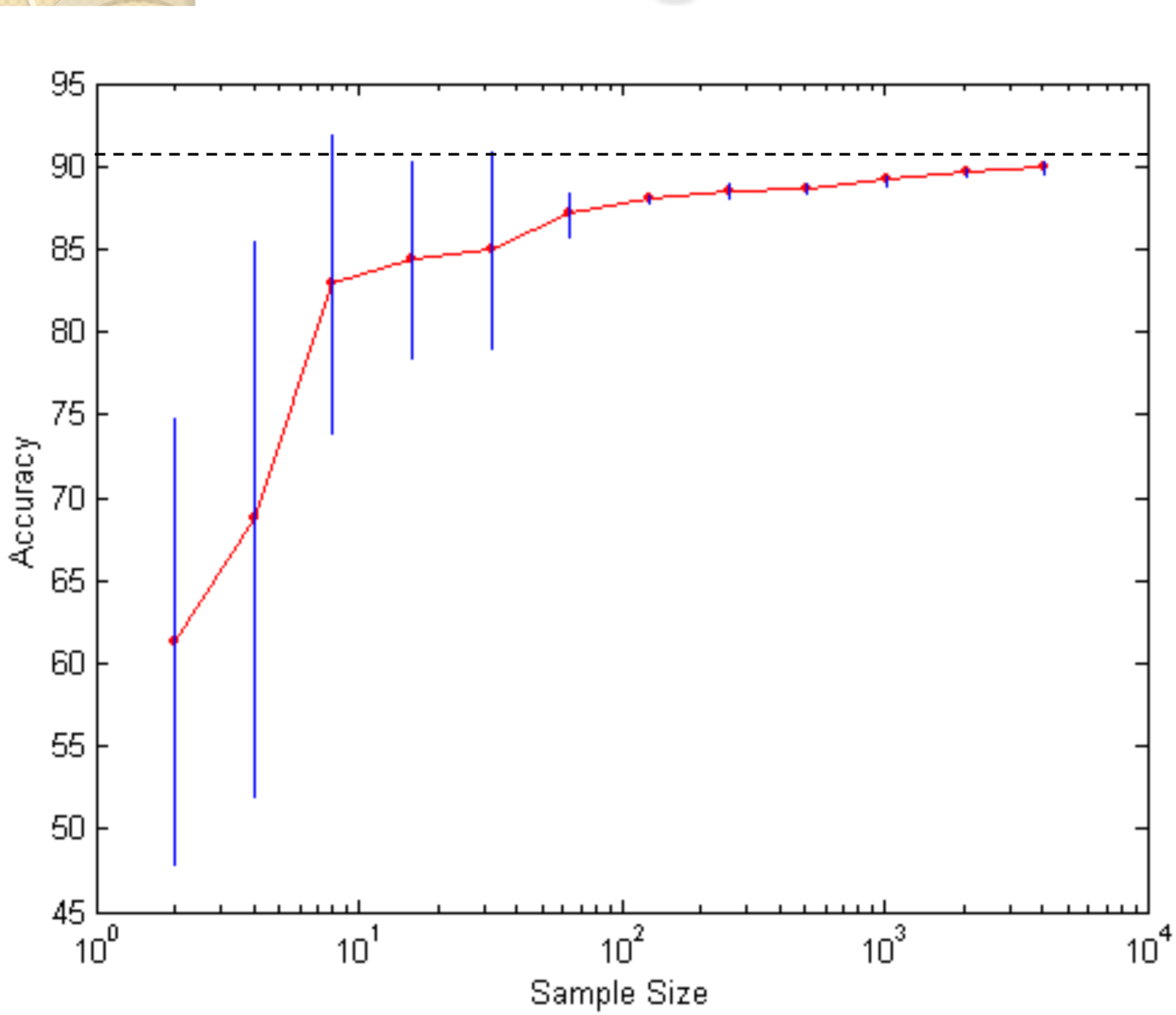
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

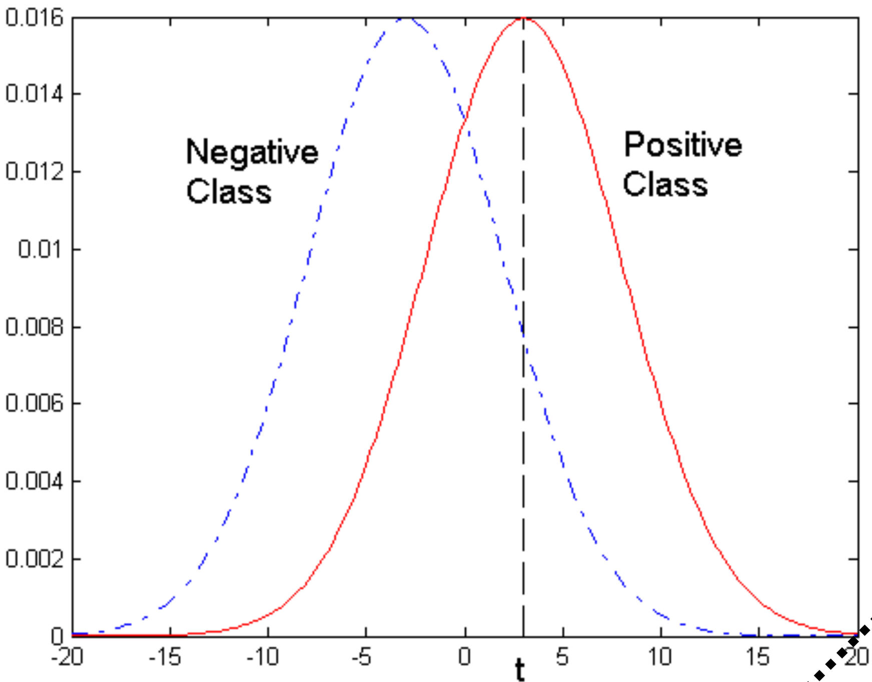
- Bias in the estimate
- Variance of estimate

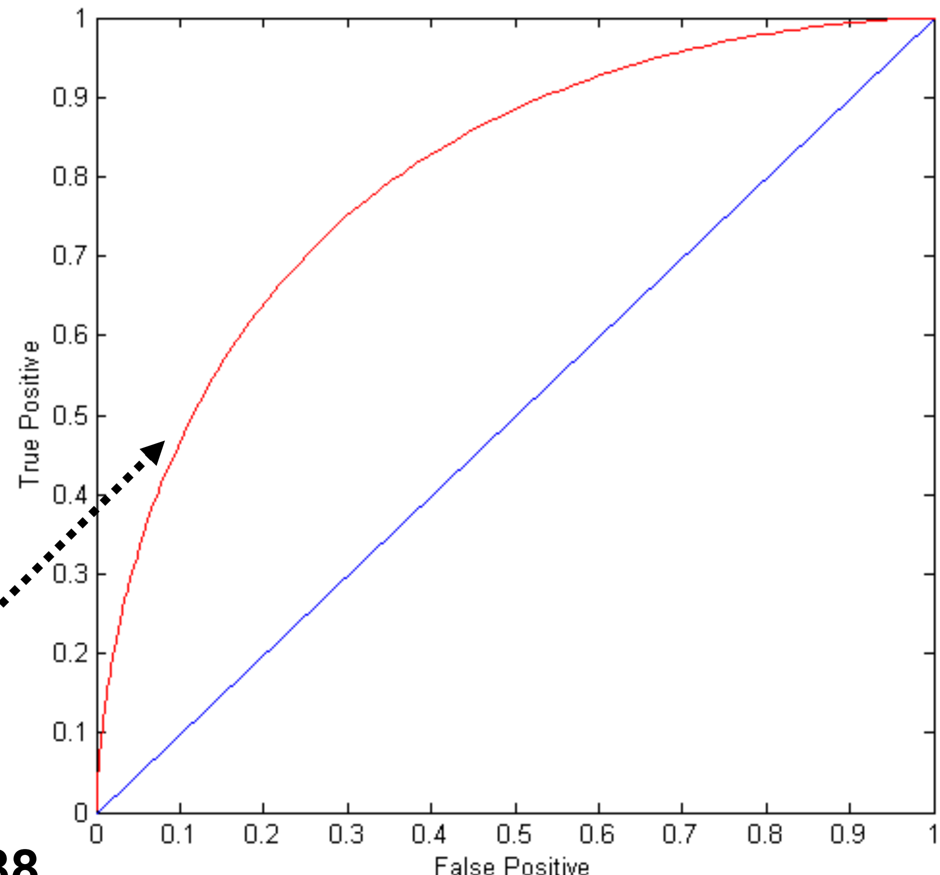
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at x > t is classified as positive





At threshold t:

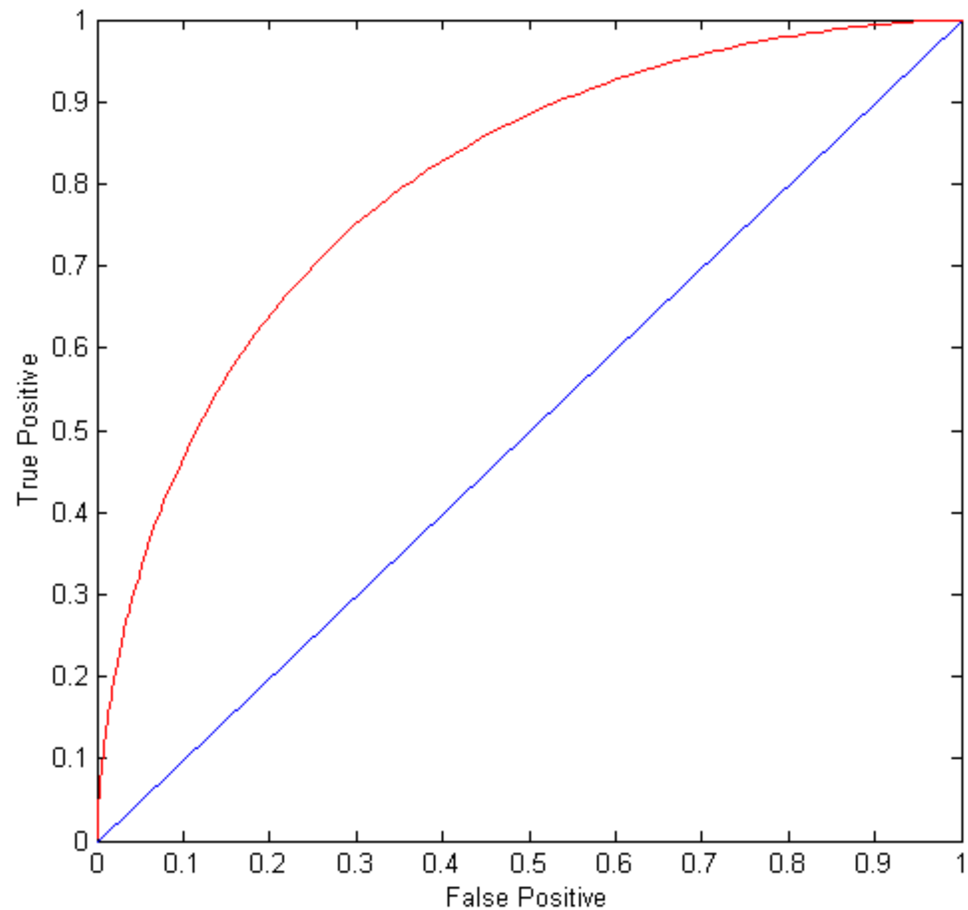
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

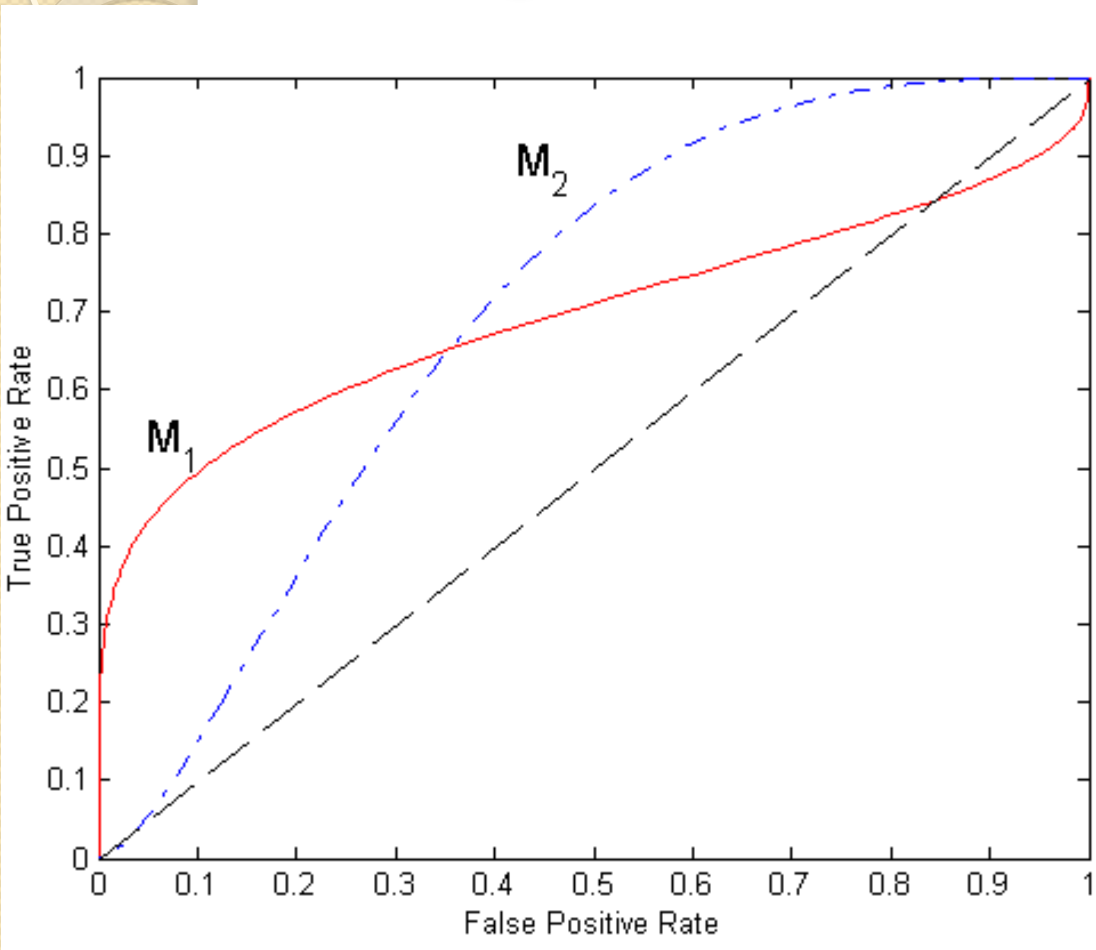
(TP,FP):

- (0,0): declare everything
  to be negative class
- (1,1): declare everything
  to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

How to Construct an ROC curve

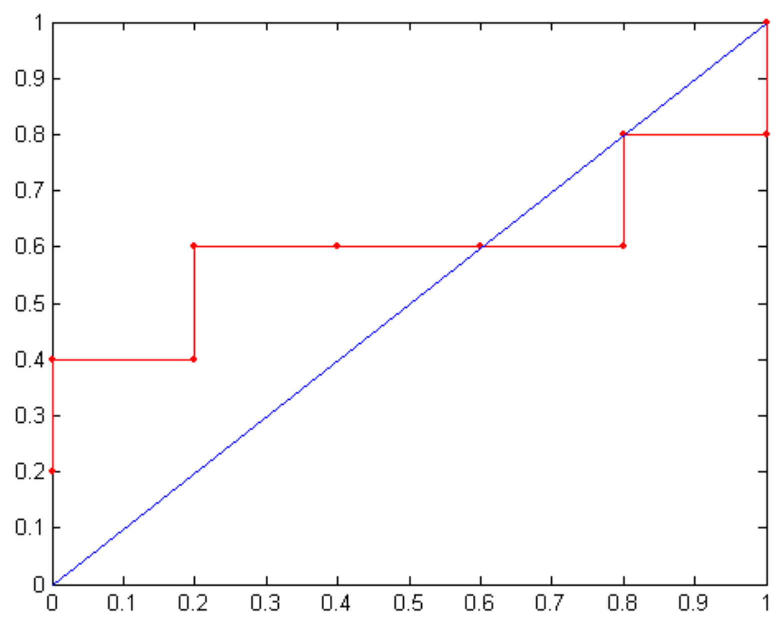
Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



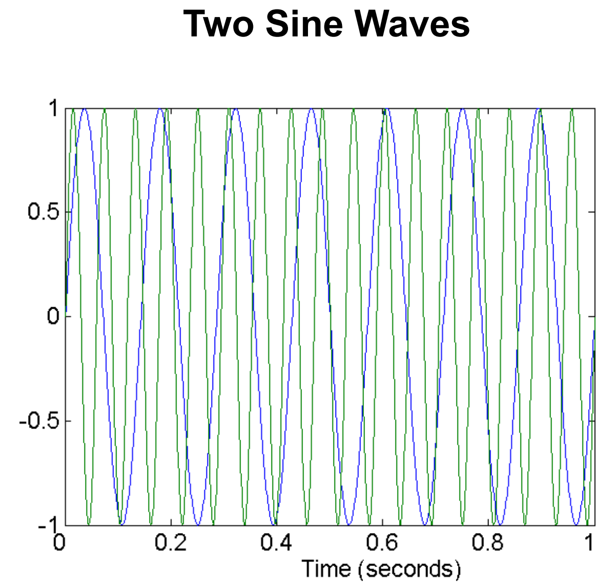
Data Quality

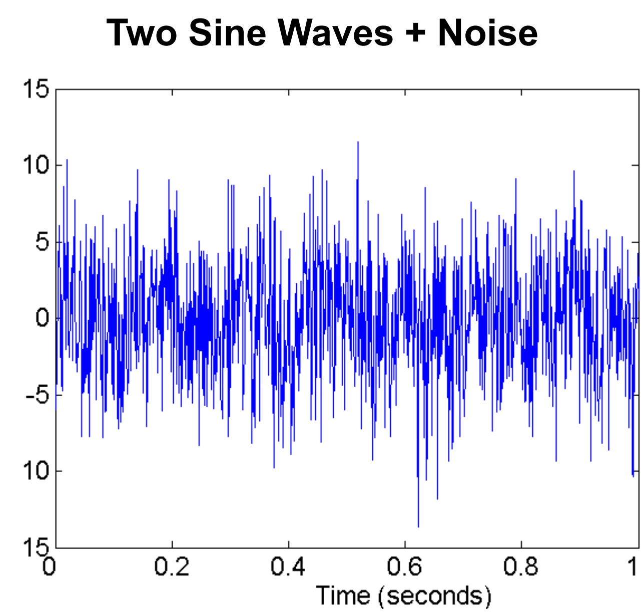
- What kinds of data quality problems?
- How can we detect problems with the data?
- What can we do about these problems?

- Examples of data quality problems:
  - Noise and outliers
  - missing values
  - duplicate data

Noise

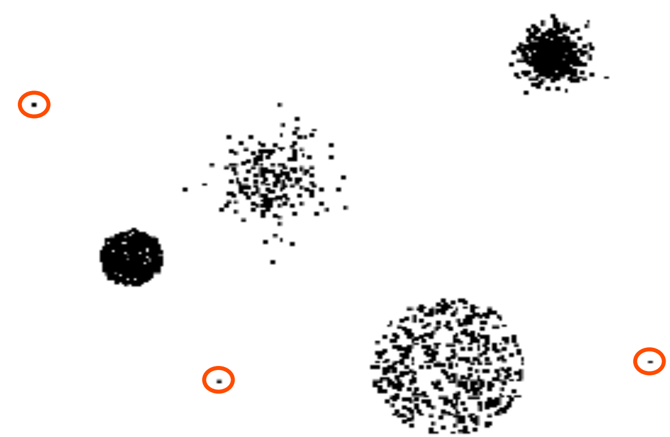
- Noise refers to modification of original values
  - Random collection of error.
  - Examples: distortion of a person' s voice when talking on a poor phone and "snow" on television screen





Outliers

- Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set



Missing Values (Think)

- Reasons for missing values
  - Information is not collected
    (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases
    (e.g., annual income is not applicable to children)

- Handling missing values (How? Think)
  - Eliminate Data Objects
  - Estimate Missing Values
  - Ignore the Missing Value During Analysis
  - Replace with all possible values (weighted by their probabilities)

Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
  - Major issue when merging data from heterogeous sources

- Examples:
  - Same person with multiple email addresses

- Data cleaning
  - Process of dealing with duplicate data issues

Data Preprocessing

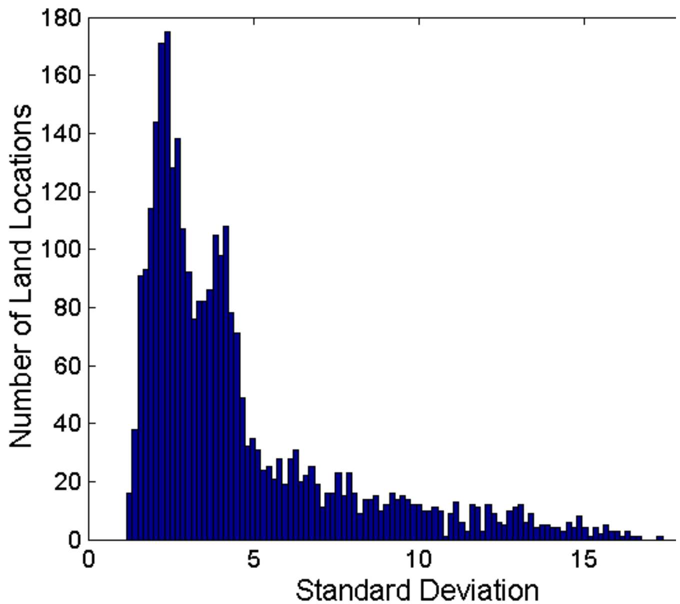
- Aggregation
- Sampling
- Dimensionality Reduction/Feature selection
- Feature creation
- Attribute Transformation
  - Discretization and Binarization

Aggregation (LESS IS MORE)

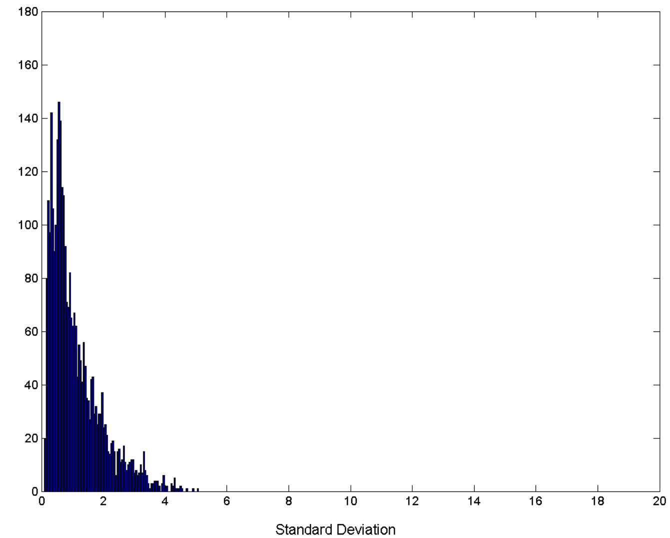
- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
  - Data reduction
    - Reduce the number of attributes or objects
  - Change of scale
    - Cities aggregated into regions, states, countries, etc
  - More "stable" data
    - Aggregated data tends to have less variability

Aggregation

Variation of Precipitation in Australia



Standard Deviation of Average Monthly Precipitation



Standard Deviation of Average Yearly Precipitation

Sampling

- Sampling is the main technique employed for data selection.
  - It is often used for both the preliminary investigation of the data and the final data analysis.

- Sampling is used in data mining because processing the entire set of data of interest is too expensive or time consuming.

Sampling …

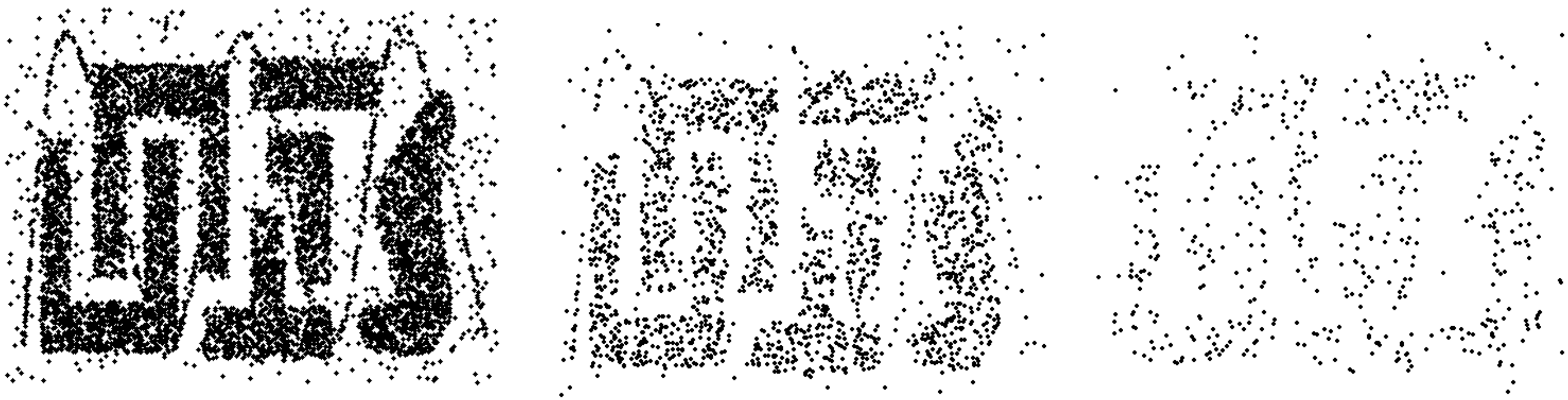
- The key principle for effective sampling is the following:
  - using a sample will work almost as well as using the entire data sets, if the sample is representative

  - A sample is representative if it has approximately the same property (of interest) as the original set of data

Types of Sampling

- Simple Random Sampling
  - There is an equal probability of selecting any particular item
- Sampling without replacement
  - As each item is selected, it is removed from the population
- Sampling with replacement
  - Objects are not removed from the population as they are selected for the sample.
    - In sampling with replacement, the same object can be picked up more than once
- Stratified sampling
  - Split the data into several partitions; then draw random samples from each partition

Sample Size



8000 points 2000 Points 500 Points

Classification Algorithms

Classification Methods

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

Bayes Classifier

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

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

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

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A1,A2,…,An)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A1,A2,…,An)
- Can we estimate P(C| A1,A2,…,An) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$\boxed{P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}}$$

  - Choose value of C that maximizes

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

  - Equivalent to choosing value of C that maximizes

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

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1 | C_j)\, P(A_2 | C_j) \ldots P(A_n | C_j)$
  - Can estimate $P(A_i | C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j) \prod P(A_i | C_j)$ is maximal.

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: $P(C) = N_c/N$
  - e.g., $P(No) = 7/10$,
    $P(Yes) = 3/10$
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}|/ N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: (A < v) or (A > v)
    - choose only one of the two splits as new attribute
  - 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, can use it to estimate the conditional probability $P(A_i|c)$

k

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**

- Normal distribution:

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

  - One for each (A_i, c_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:

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

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=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

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 probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

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)

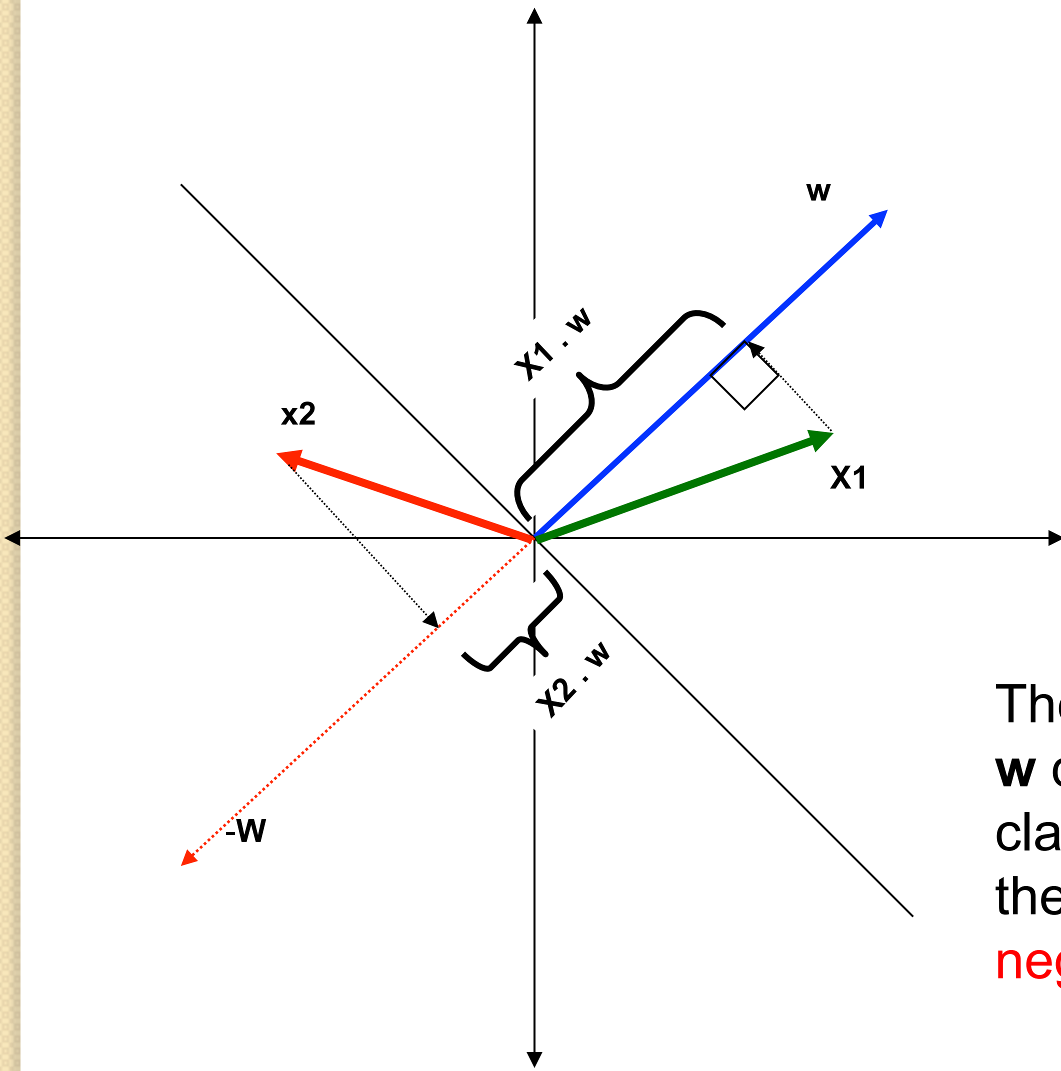
Perceptrons and Neural Networks

Linear Classifiers

- Let's simplify life by assuming:
  - Every instance is a vector of real numbers, $\mathbf{x}=(x_1,\ldots,x_n)$. (Notation: boldface $\mathbf{x}$ is a vector.)
  - There are only two classes, $y=(+1)$ and $y=(-1)$
- A <u>linear classifier</u> is vector **w** of the same dimension as **x** that is used to make this prediction:

$$\hat{y} = \mathrm{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$$

$$\mathrm{sign}(w.x) = \begin{cases} +1 & \text{if } \; \mathrm{w}.x \geq 0 \\ -1 & \text{if } \; \mathrm{w.x} < 0 \end{cases}$$

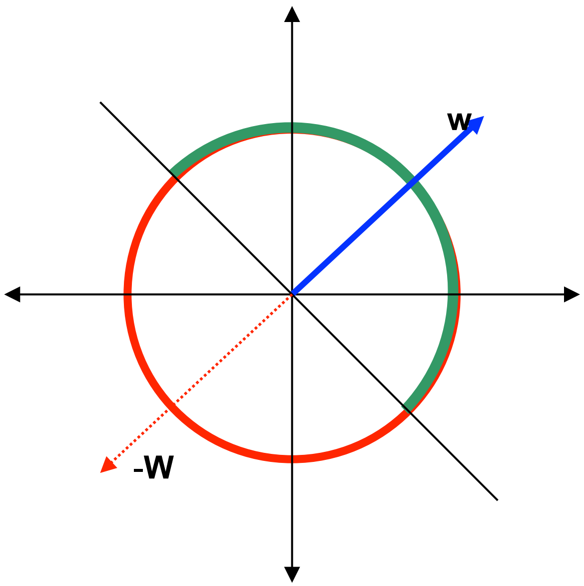


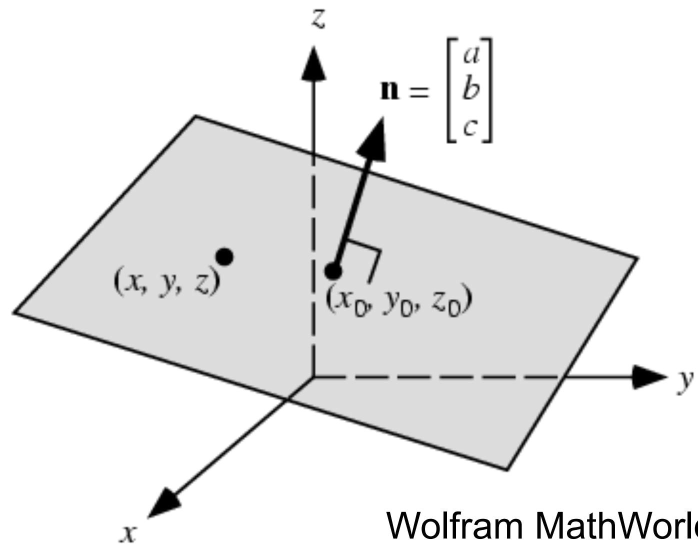
Visually, **x** · **w** is the distance you get if you "project **x** onto **w**"

In 3d: line→plane
In 4d: plane→hyperplane
…

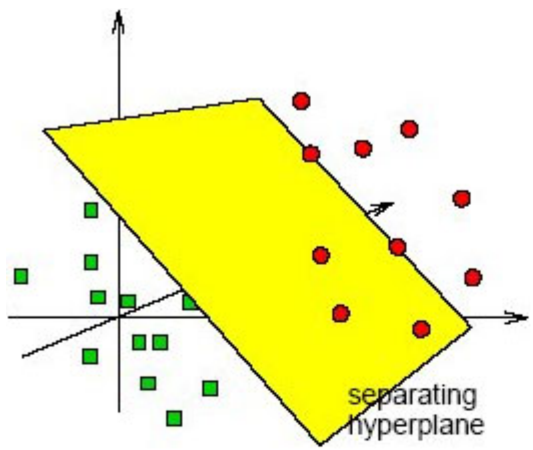
The <u>line</u> perpendicular to **w** divides the vectors classified as positive from the vectors classified as negative.

$$\hat{y} = \text{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

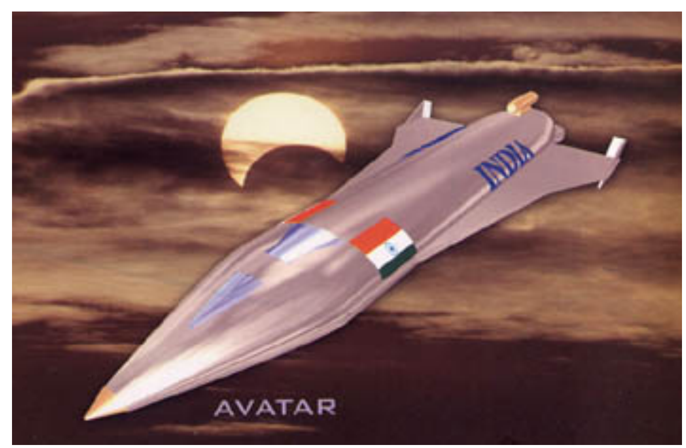




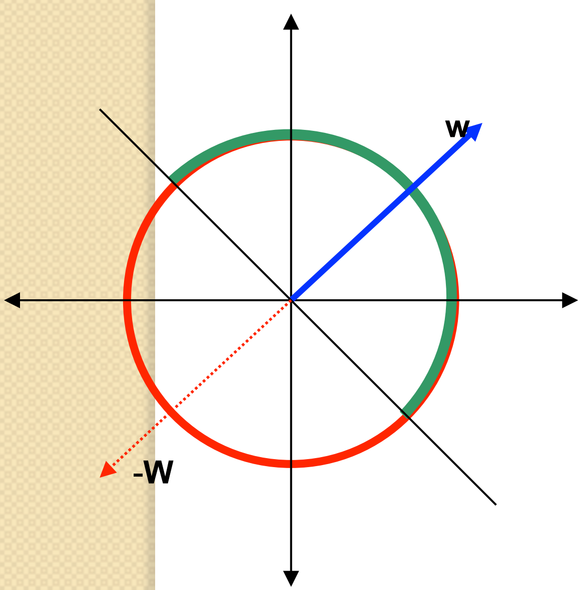
Wolfram MathWorld



Mediaboost.com



Geocities.com/bharatvarsha1947



Notice that the <u>separating hyperplane</u> goes through the origin…if we don't want this we can preprocess our examples:

$$\cancel{\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle}$$

$$\mathbf{x} = \langle 1, x_1, x_2, \ldots, x_n \rangle$$

$$\cancel{\hat{y} = \text{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \text{sign}(\mathbf{w} \cdot \mathbf{x})}$$

$$\hat{y} = \text{sign}(w_0 1 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

V

$$\langle x_1, \ldots, x_n \rangle \rightarrow$$

$$\langle x_{outlook,sunny}, x_{outlook,overcast}, x_{outlook,rain}, x_{temp,hot}, x_{temp,mild}, x_{temp,cool}, \ldots, \rangle$$

Day	Outlook	Temperature	Humidity	Wind	PlayTen	
D1	Sunny	Hot	High	Weak	No	
D2	Sunny	Hot	High	Strong	No	
D3	Overcast	Hot	High	Weak	Yes	
D4	Rain	Mild	High	Weak	Yes	→ +1
D5	Rain	Cool	Normal	Weak	Yes	
D6	Rain	Cool	Normal	Strong	No	→ -1
D7	Overcast	Cool	Normal	Strong	Yes	
D1		Mild		ng	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$$D7 = \langle x_{outlook,sunny} = 0, x_{outlook,overcast} = 1, x_{outlook,rain} = 0, \ldots, \rangle$$

$$= \langle 0,1,0, \; 0,0,1, \; 0,1, \; 1,0 \rangle$$

Outlook overcast

Humidity normal

What have we given up?

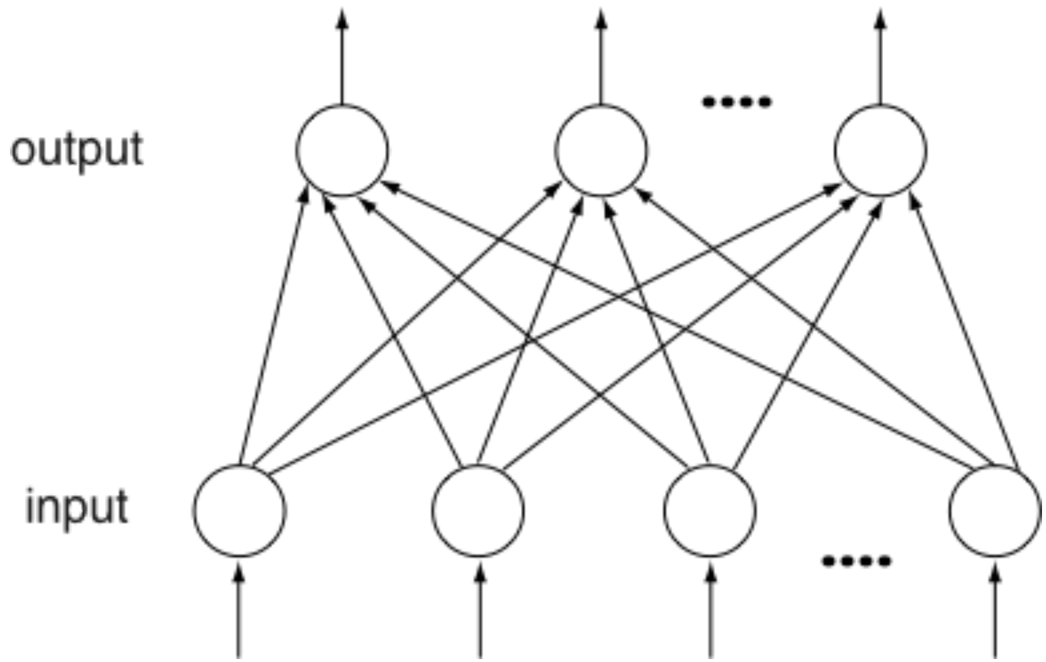
- Not much!
  - Practically, it’s a little harder to understand a particular example (or classifier)
  - Practically, it’s a little harder to debug
- You can still express the same information
- You can analyze things mathematically *much more easily*

Perceptron (Frank Rosenblatt, 1957)

- First learning algorithm for neural networks;

- Originally introduced for character classification, where each character is represented as an image;

Perceptron (contd.)



Total input to output node: $\sum_{j=1}^{n} w_j x_j$

Output unit performs the function: (activation function):

$$H(x) = \begin{cases} 1 & \text{if } \mathrm{H}(x) \geq 0 \\ 0 & \text{if } \mathrm{H}(x) < 0 \end{cases}$$

Perceptron: Learning Algorithm

- **Goal**: we want to define a learning algorithm for the weights in order to compute a mapping from the inputs to the outputs;
- **Example**: two class character recognition problem.
  - **Training set**: set of images representing either the character 'a' or the character 'b' (supervised learning);
  - **Learning Task**: Learn the weights so that when a new unlabelled image comes in, the network can predict its label.
  - Settings:

    Class 'a' → 1 (class C1)

    Class 'b' → 0 (class C2)

    n input units (intensity level of a pixel)

    1 output unit

The perceptron needs to learn

$$f : \Re^n \rightarrow \{0,1\}$$

Perceptron: Learning Algorithm

The algorithm proceeds as follows:

- Initial random setting of weights;
- The input is a random sequence $\{\boldsymbol{x}_k\}_{k \in \aleph}$
- For each element of class C1, if output = 1 (correct) do nothing, otherwise update weights;
- For each element of class C2, if output = 0 (correct) do nothing, otherwise update weights.

Perceptron: Learning Algorithm

A bit more formally:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \qquad \boldsymbol{w} = (w_1, w_2, \ldots, w_n)$$

w_0 : Threshold of the output unit

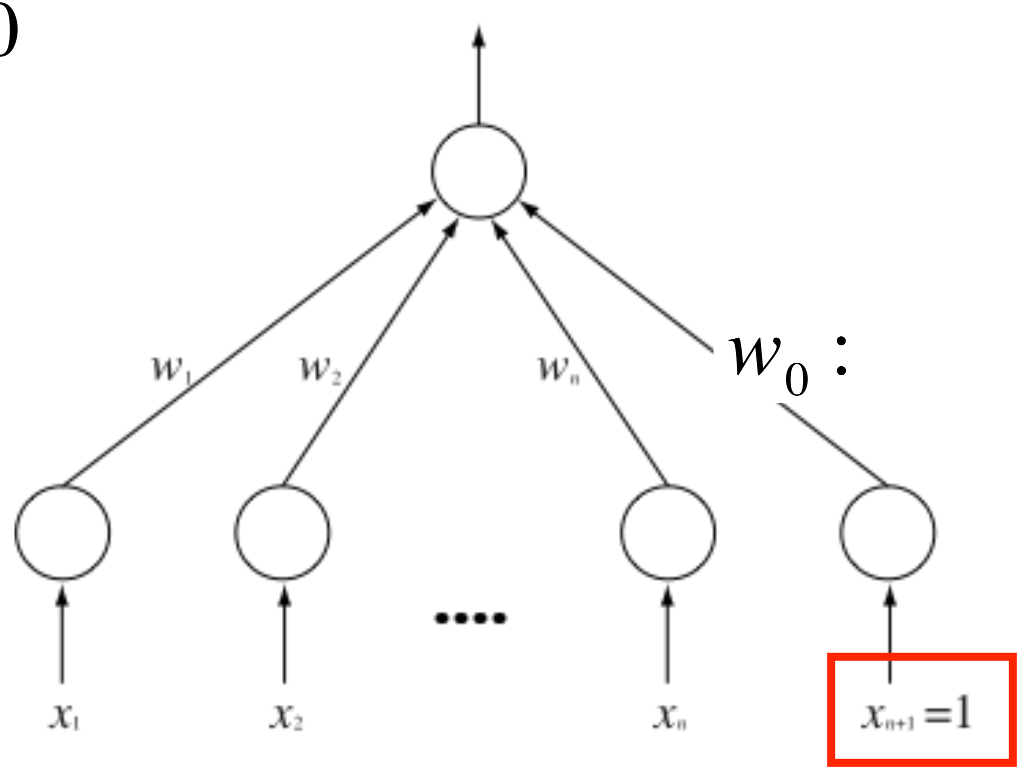
$$\boldsymbol{w}\boldsymbol{x}^T = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Output is 1 if $\boldsymbol{w}\boldsymbol{x}^T - w_0 \geq 0$

To eliminate the explicit dependence on w_0 :

Output is 1 if:

$$\hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T = \sum_{i=1}^{n+1} w_i x_i \geq 0$$



Perceptron: Learning Algorithm

- We want to learn values of the weights so that the perceptron correctly discriminate elements of C1 from elements of C2:

- Given x in input, if x is classified correctly, weights are unchanged, otherwise:

$$\boldsymbol{w}' = \begin{cases} \boldsymbol{w} + \boldsymbol{x} & \text{if element of class } C_1 \text{ (1) was classified as in } C_2 \\ \boldsymbol{w} - \boldsymbol{x} & \text{if element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

Perceptron: Learning Algorithm

$$w' = \begin{cases} w + x & \text{if element of class } C_1 \text{ (1) was classified as in } C_2 \\ w - x & \text{if element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

- **1st case**: $x \in C_1$ and was classified in C_2

The correct answer is 1, which corresponds to: $\hat{w}\hat{x}^T \geq 0$

We have instead: $\hat{w}\hat{x}^T < 0$

We want to get closer to the correct answer: $wx^T < w'x^T$

$$wx^T < w'x^T \quad \text{iff} \quad wx^T < (w + x)x^T$$

$$(w + x)x^T = wx^T + xx^T = wx^T + \|x\|^2$$

$$\text{because } \|x\|^2 \geq 0, \text{ the condition is verified}$$

Perceptron: Learning Algorithm

$$w' = \begin{cases} w + x & \text{if element of class } C_1 \ (1) \ \text{was classified as in } C_2 \\ w - x & \text{if element of class } C_2 \ (0) \ \text{was classified as in } C_1 \end{cases}$$

- **2nd case**: $x \in C_2$ *and was classified in* C_1

The correct answer is 0, which corresponds to: $\hat{w}\hat{x}^T < 0$

We have instead: $\hat{w}\hat{x}^T \geq 0$

We want to get closer to the correct answer: $wx^T > w'x^T$

$$wx^T > w'x^T \quad \text{iff} \quad wx^T > (w - x)x^T$$

$$(w - x)x^T = wx^T - xx^T = wx^T - \|x\|^2$$

$$\text{because } \|x\|^2 \geq 0, \text{ the condition is verified}$$

The previous rule allows the network to get closer to the correct answer when it performs an error.

Perceptron: Learning Algorithm

- <u>In summary</u>:

1. A random sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k, \cdots$ is generated such that $\boldsymbol{x}_i \in C_1 \cup C_2$

2. If $\boldsymbol{x}_k$ is correctly classified, then $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k$ otherwise

$$\boldsymbol{w}_{k+1} = \begin{cases} \boldsymbol{w}_k + \boldsymbol{x}_k & if\ \boldsymbol{x}_k \in C_1 \\ \boldsymbol{w}_k - \boldsymbol{x}_k & if\ \boldsymbol{x}_k \in C_2 \end{cases}$$

Perceptron: Learning Algorithm

Does the learning algorithm converge?

Convergence theorem: Regardless of the initial choice of weights, if the two classes are linearly separable, i.e. there exist $\boldsymbol{w}$ s.t.

$$\begin{cases} \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T \geq 0 \ \ \boldsymbol{if} \ \ \boldsymbol{x} \in C_1 \\ \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T < 0 \ \ \boldsymbol{if} \ \ \boldsymbol{x} \in C_2 \end{cases}$$

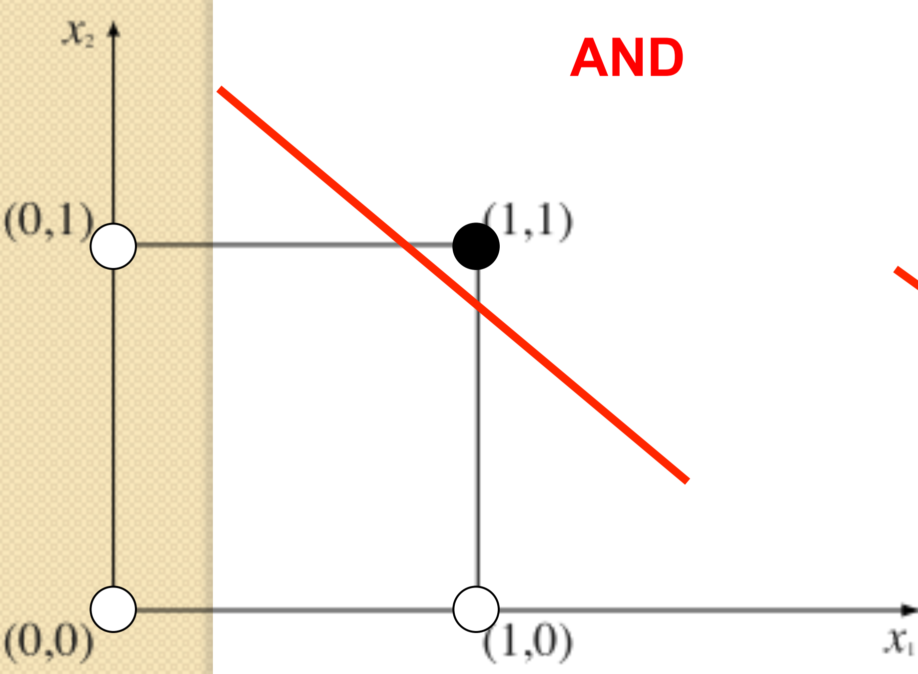
then the learning rule will find such solution after a finite number of steps.

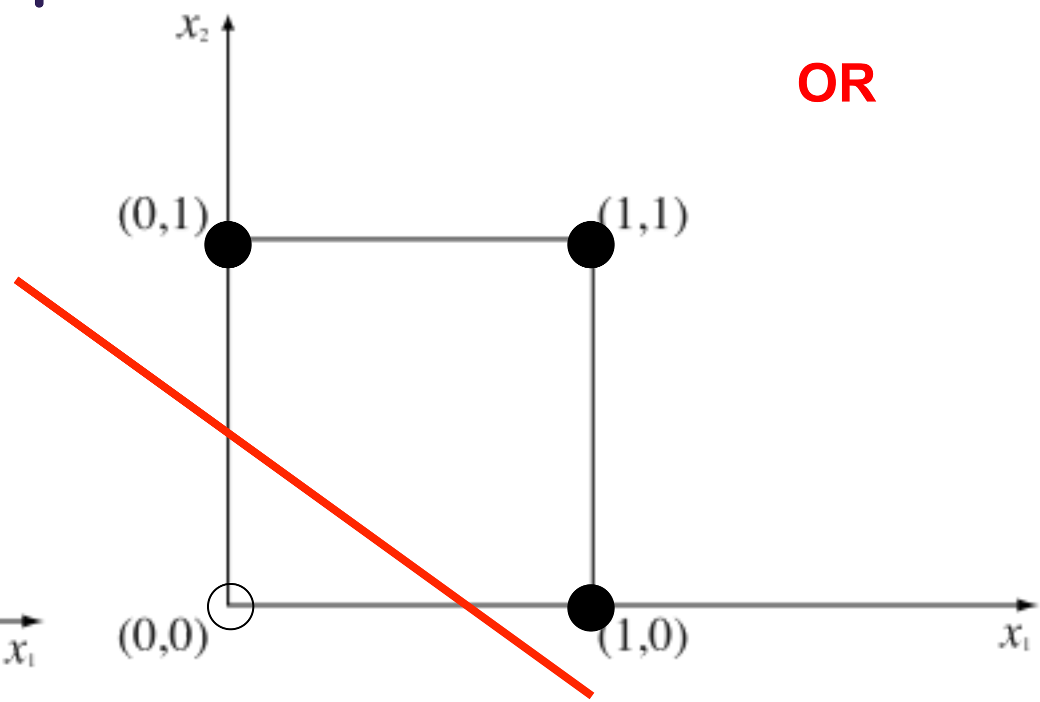
Representational Power of Perceptrons

- Marvin Minsky and Seymour Papert, "Perceptrons" 1969:

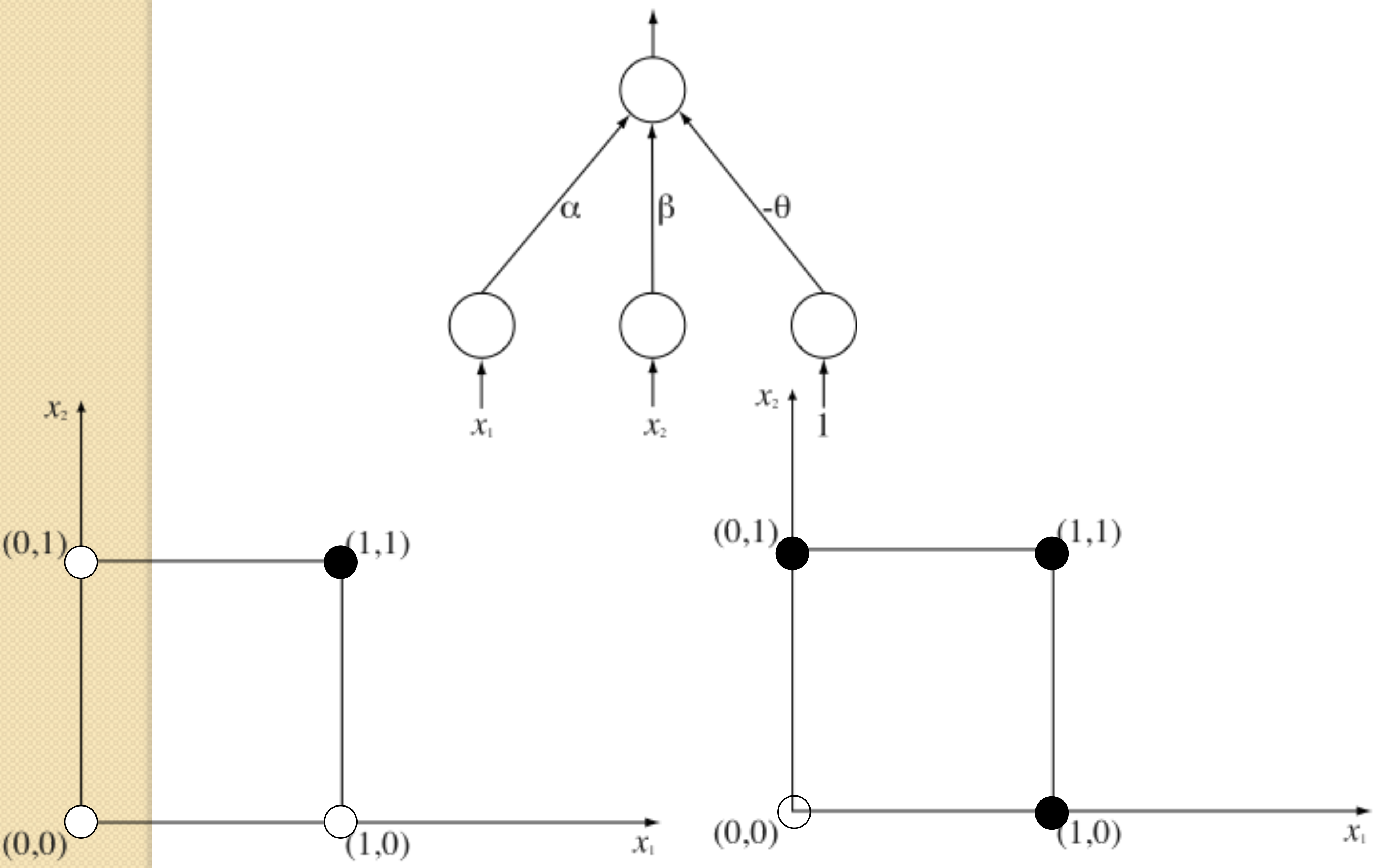
"The perceptron can solve only problems with linearly separable classes."

- Examples of linearly separable Boolean functions:

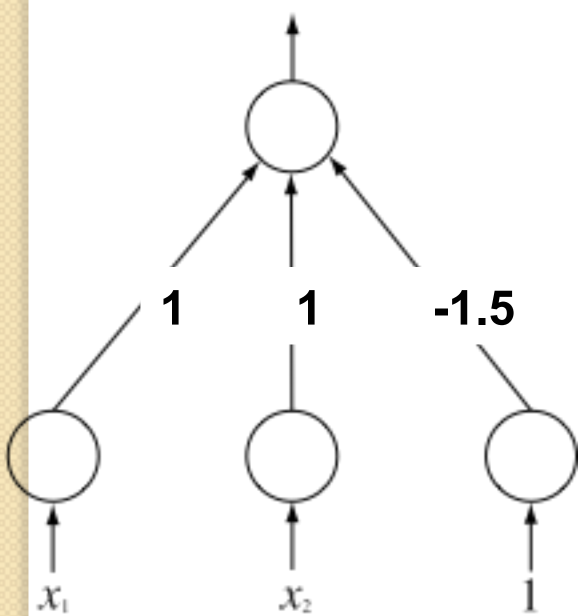




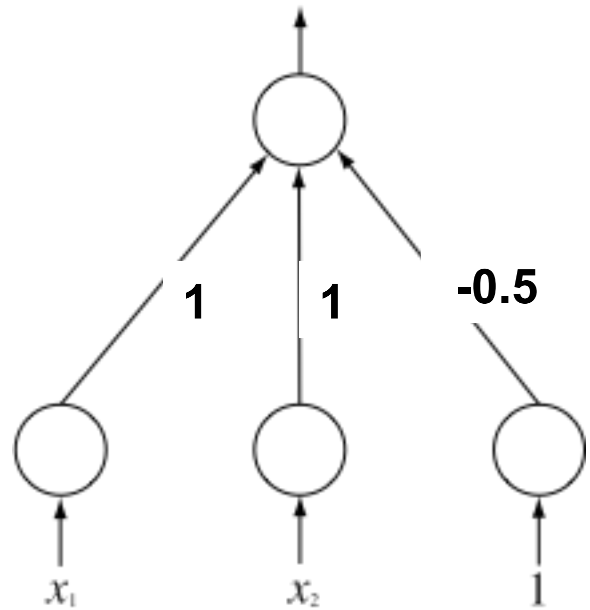
What should be the learned weights ?



Representational Power of Perceptrons



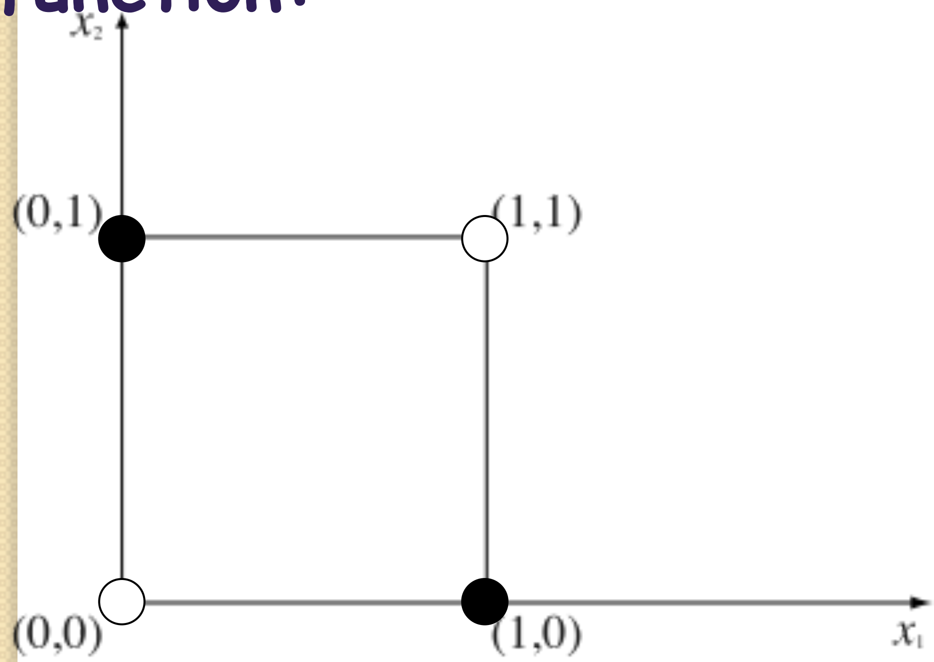
Perceptron that computes the AND function



Perceptron that computes the OR function

Representational Power of Perceptrons

- Example of a non linearly separable Boolean function:

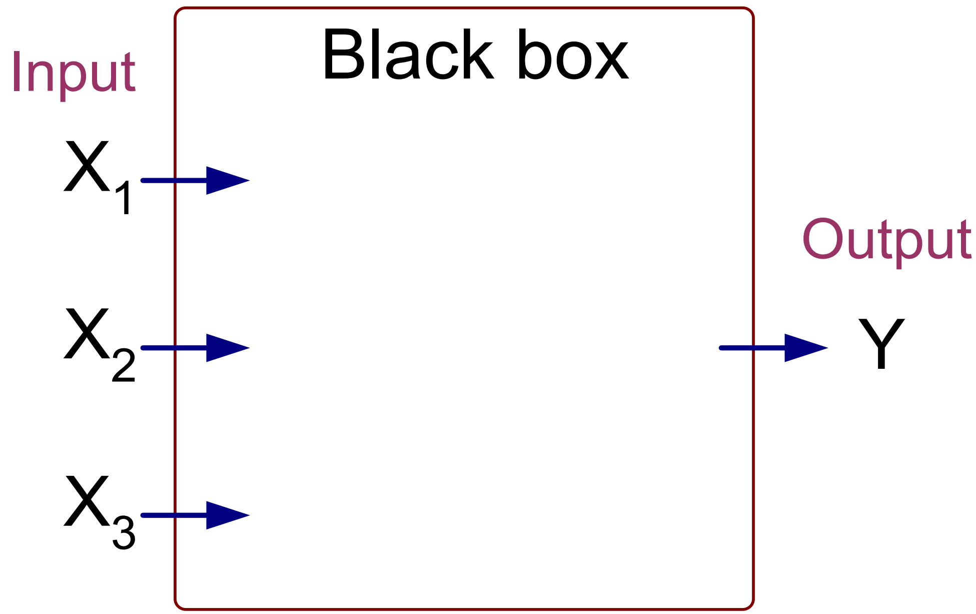


EX-OR

The EX-OR function cannot be computed by a perceptron

Artificial Neural Networks (ANN)

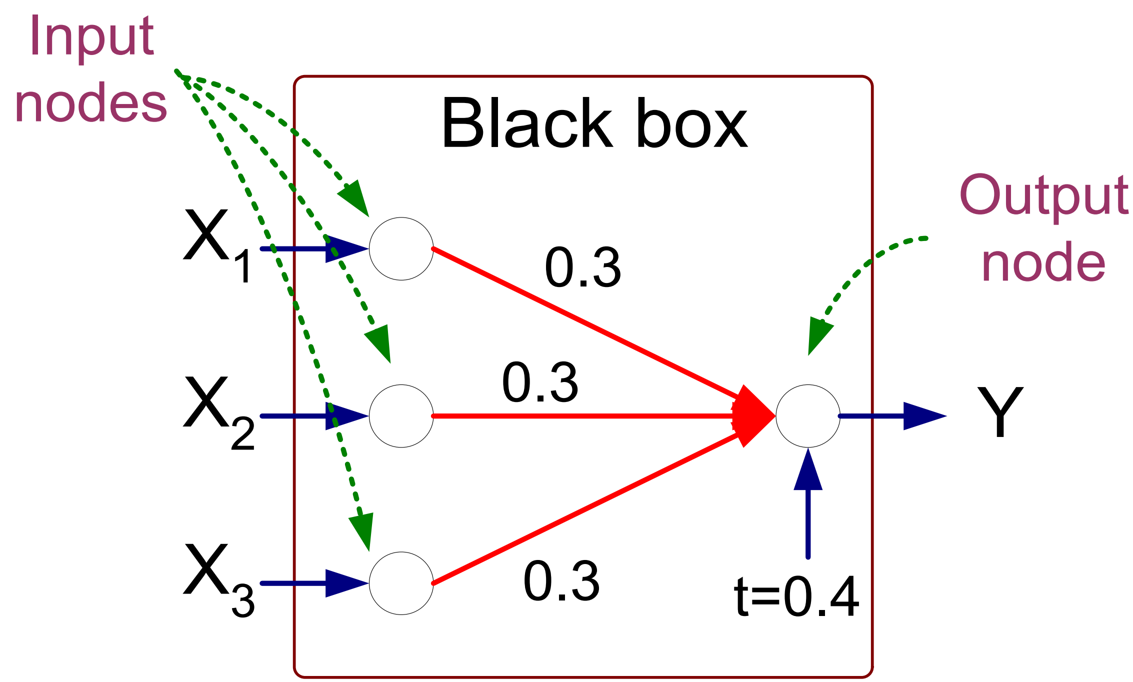
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



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

Artificial Neural Networks (ANN)

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

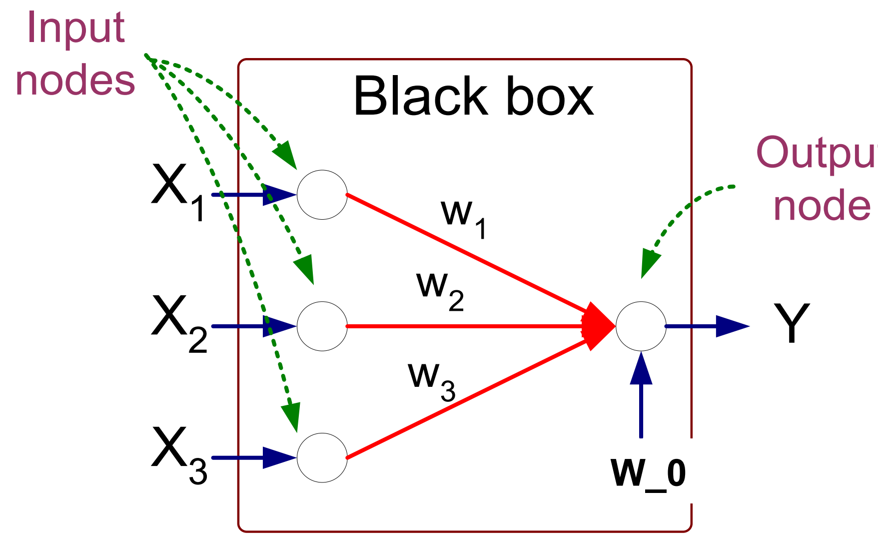


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold w_0

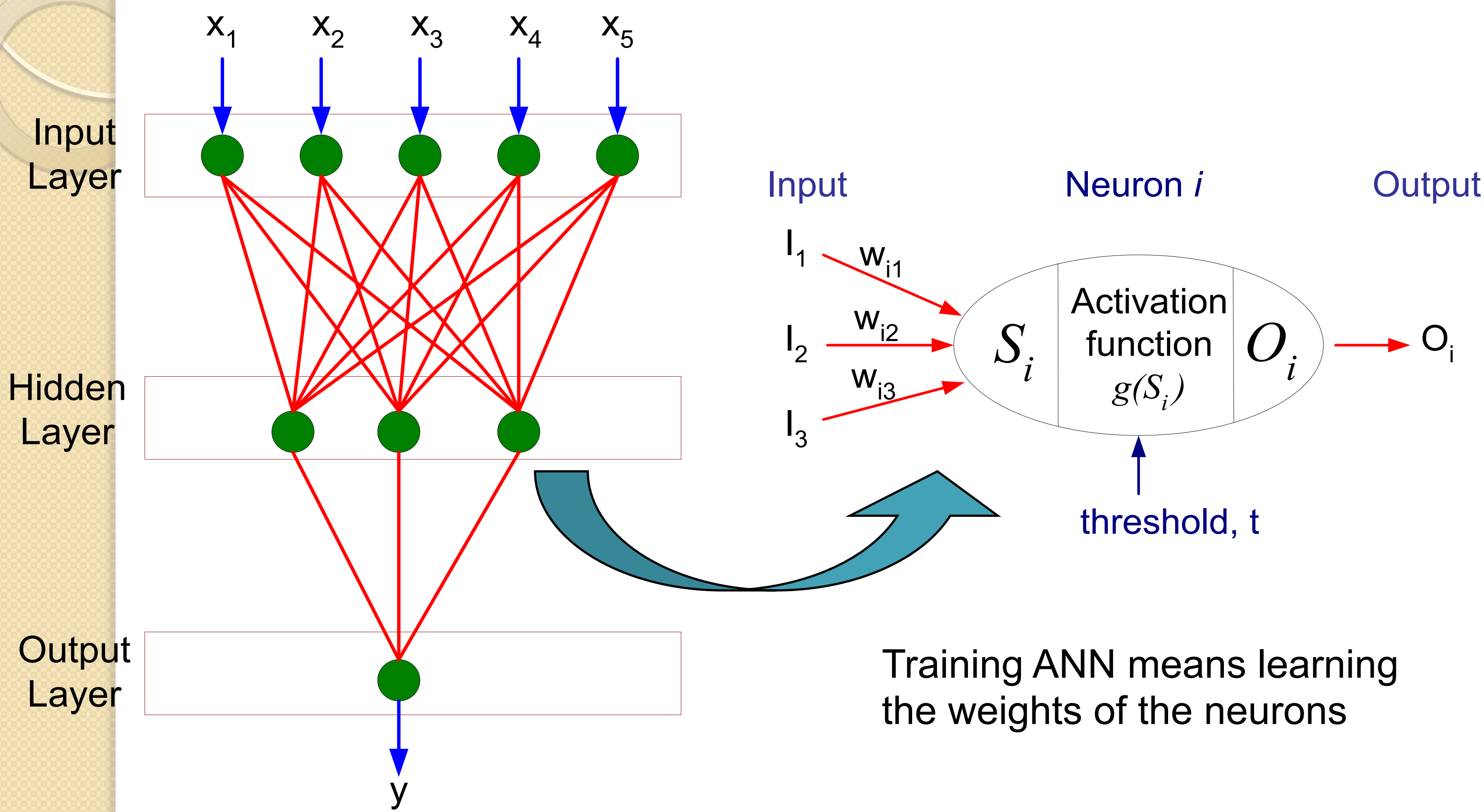


Perceptron Model

$$Y = I(\sum_i w_i X_i - w_0) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - w_0)$$

General Structure of ANN

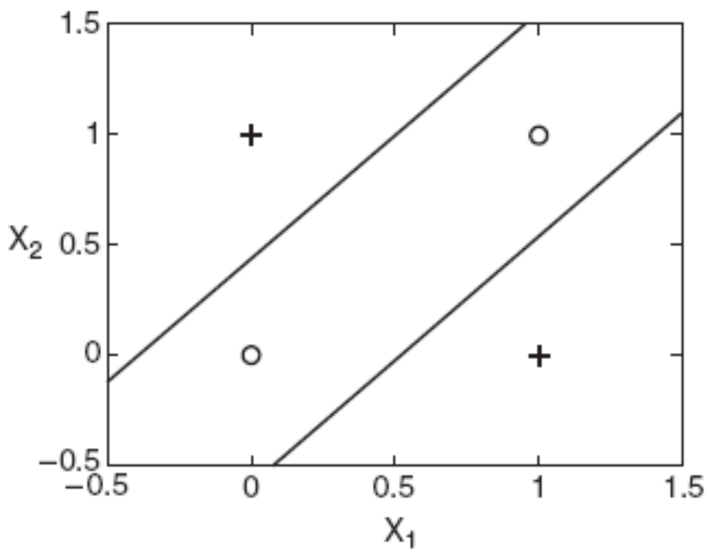


Training ANN means learning the weights of the neurons

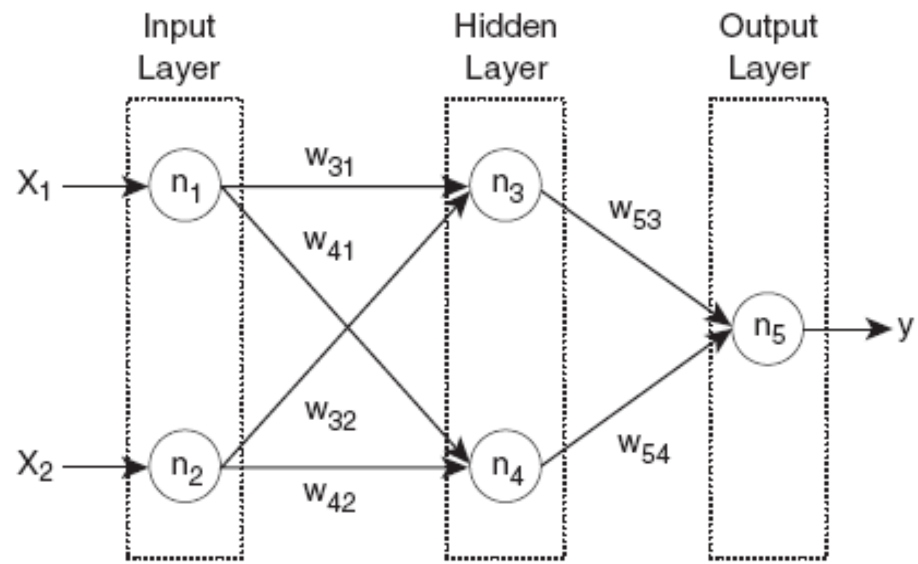
Algorithm for learning ANN

- Initialize the weights ($w_0, w_1, \ldots, w_k$)

- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Objective function: $$E = \sum_i \left[Y_i - f(w_i, X_i) \right]^2$$
  - Find the weights w_i' s that minimize the above objective function
    - e.g., backpropagation algorithm



(a) Decision boundary.



(b) Neural network topology.

Figure 5.19. A two-layer, feed-forward neural network for the XOR problem.

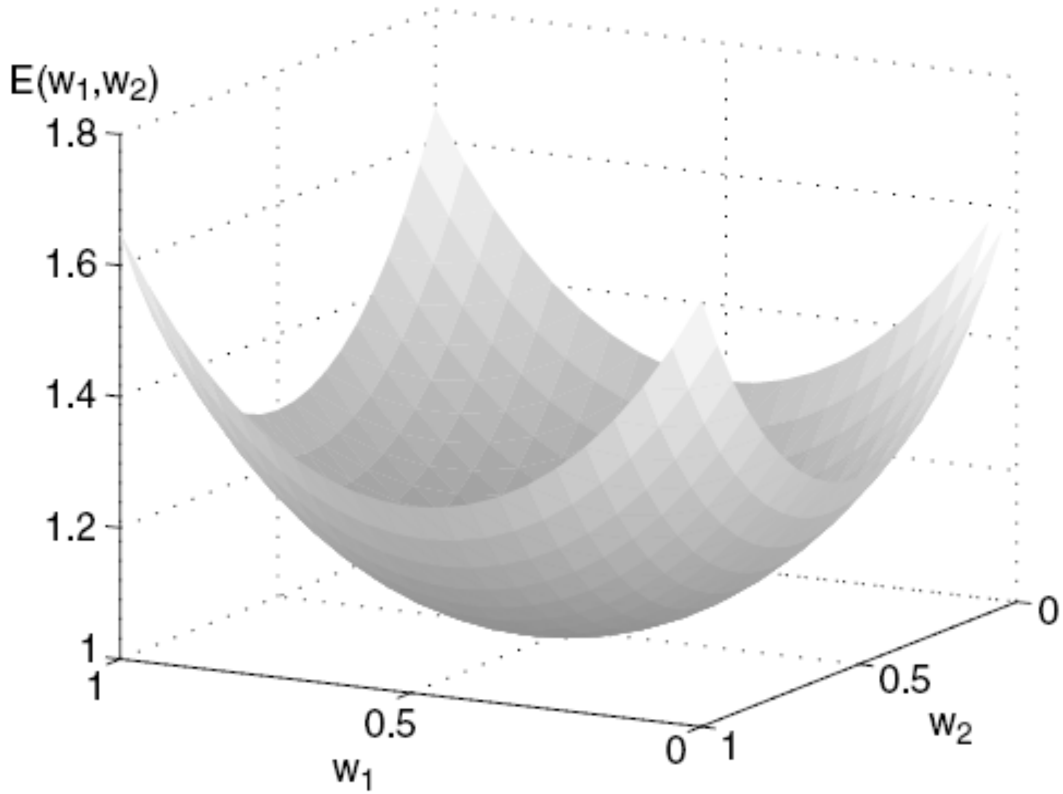


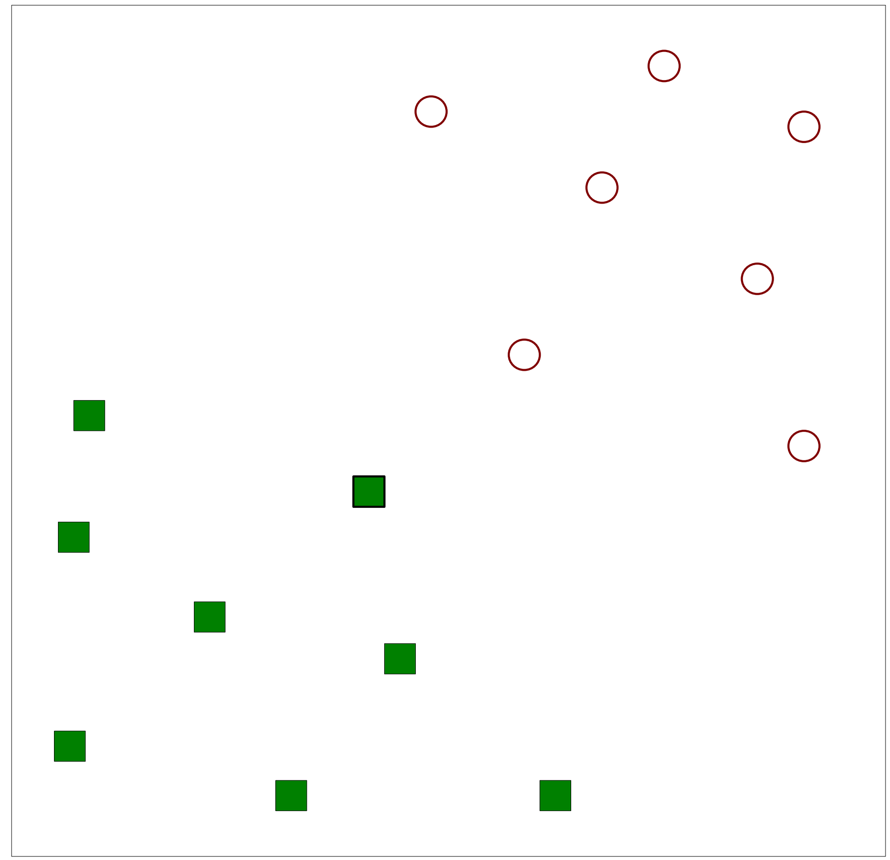
Figure 5.20. Error surface $E(w_1, w_2)$ for a two-parameter model.

ANN Characteristics.

- Universal approximators.
- Can handle redundant features
- Sensitive to noise
- Gradient descent -> can lead to local minima
- Can be time consuming training when nodes are high.
- Onus is on the user to model the network topology.

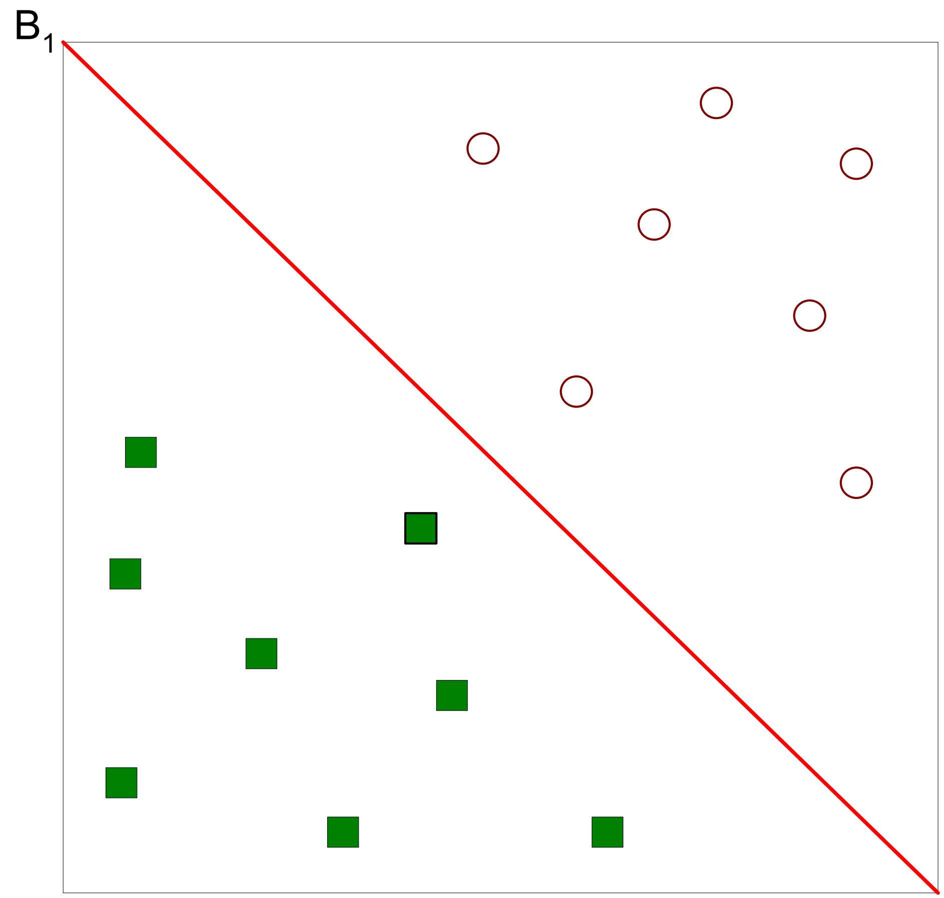
Support Vector Machines

Support Vector Machines



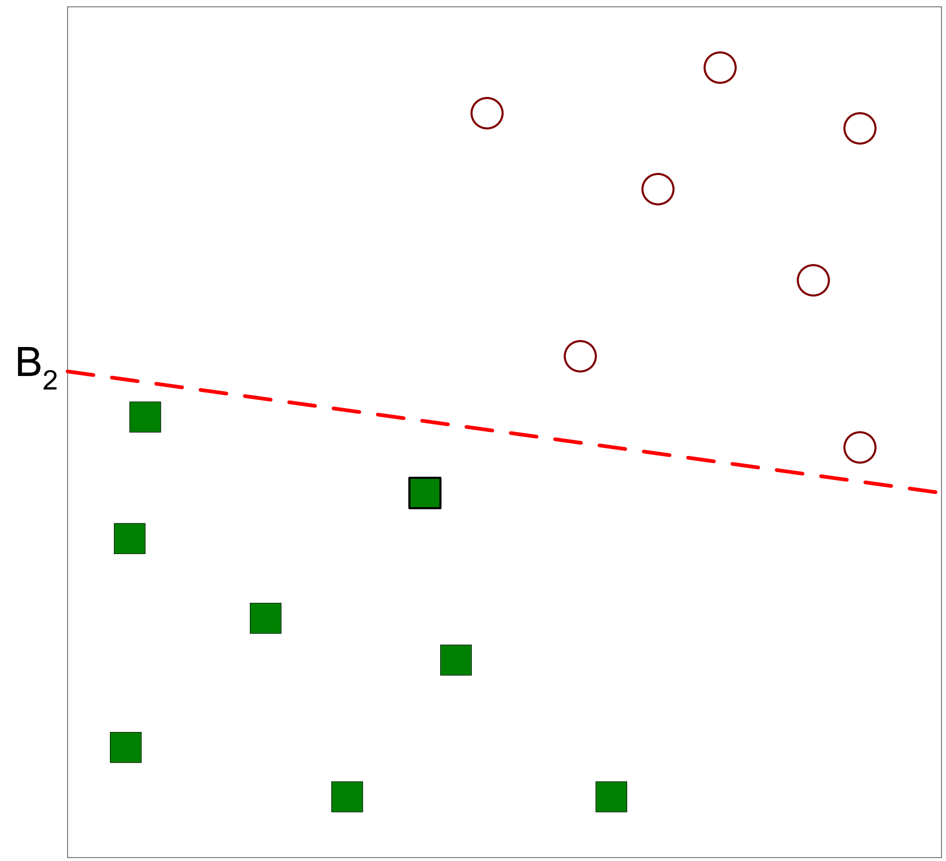
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



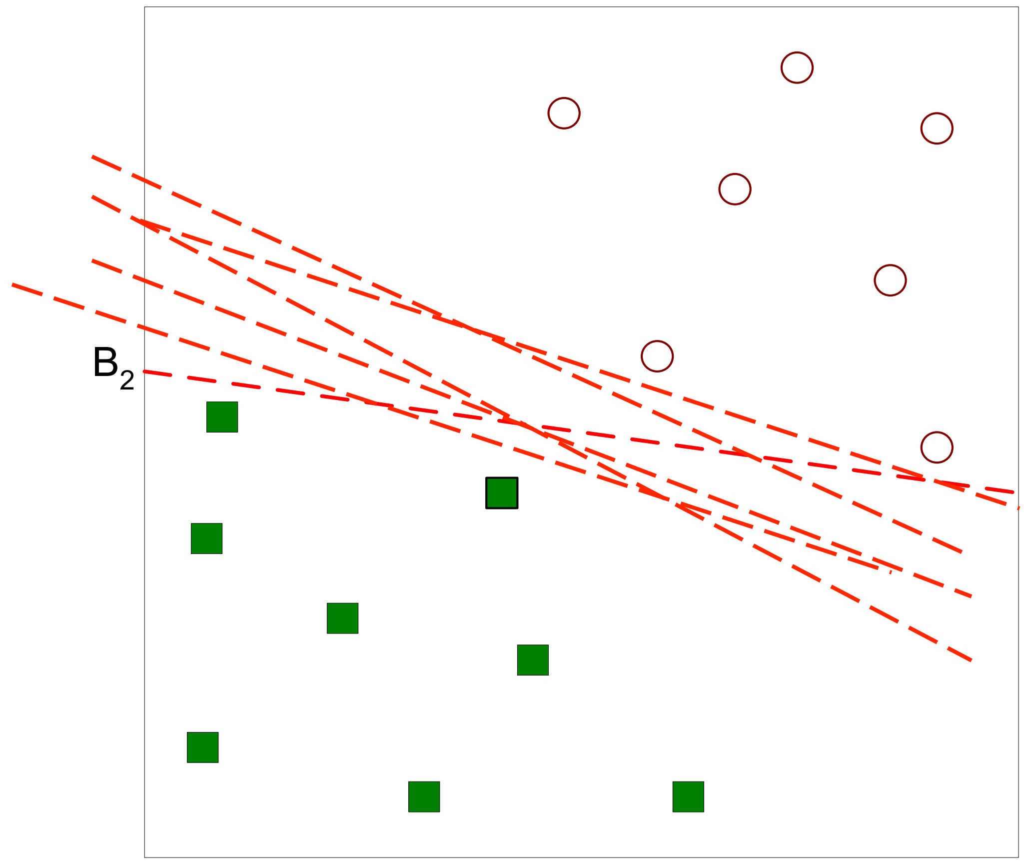
- One Possible Solution

Support Vector Machines



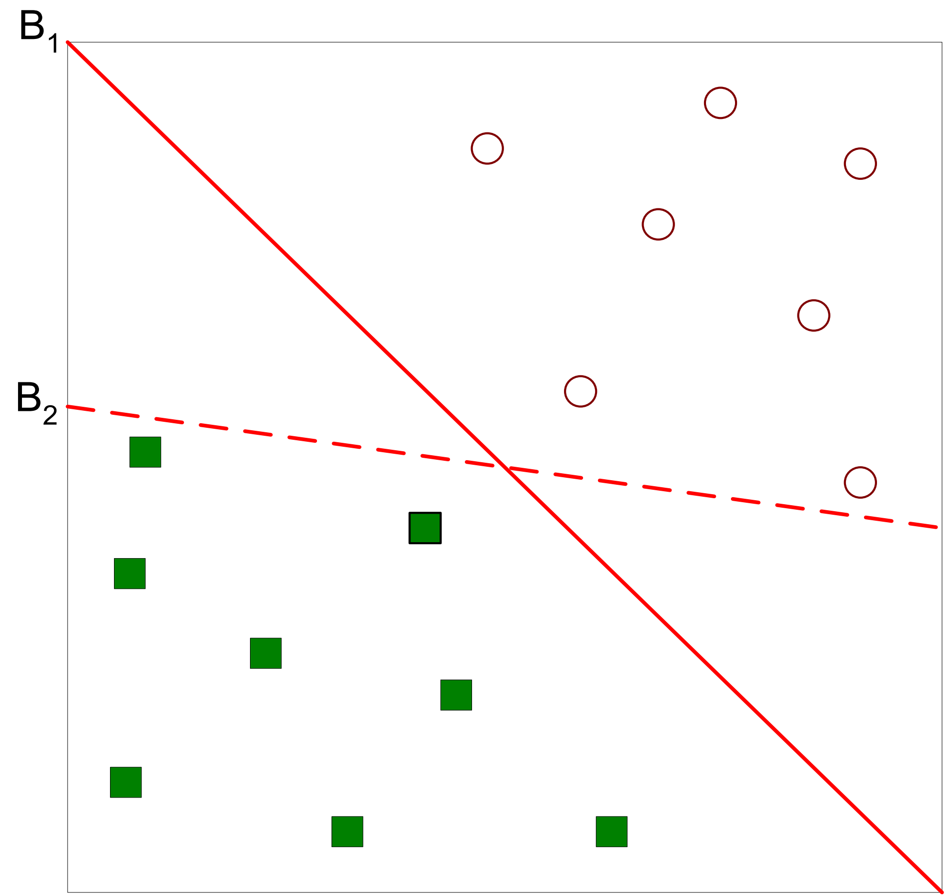
- Another possible solution

Support Vector Machines



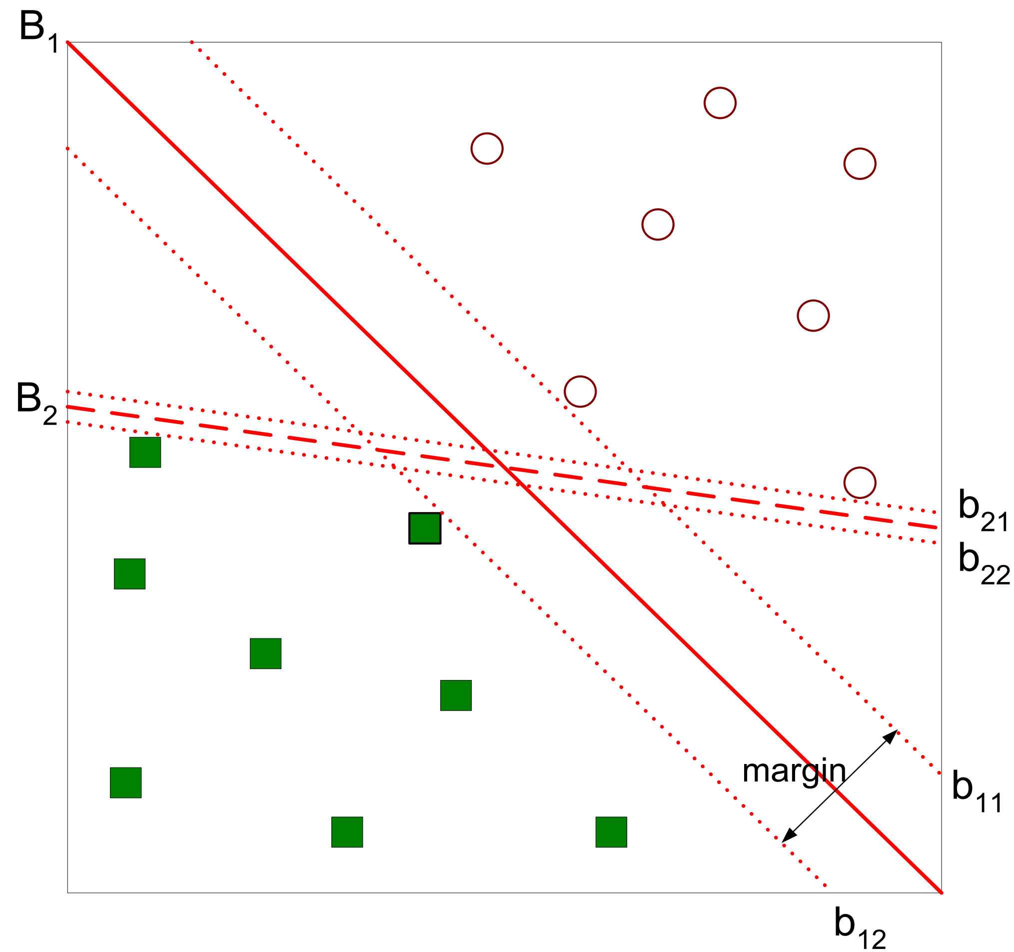
- Other possible solutions

Support Vector Machines



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

Support Vector Machines

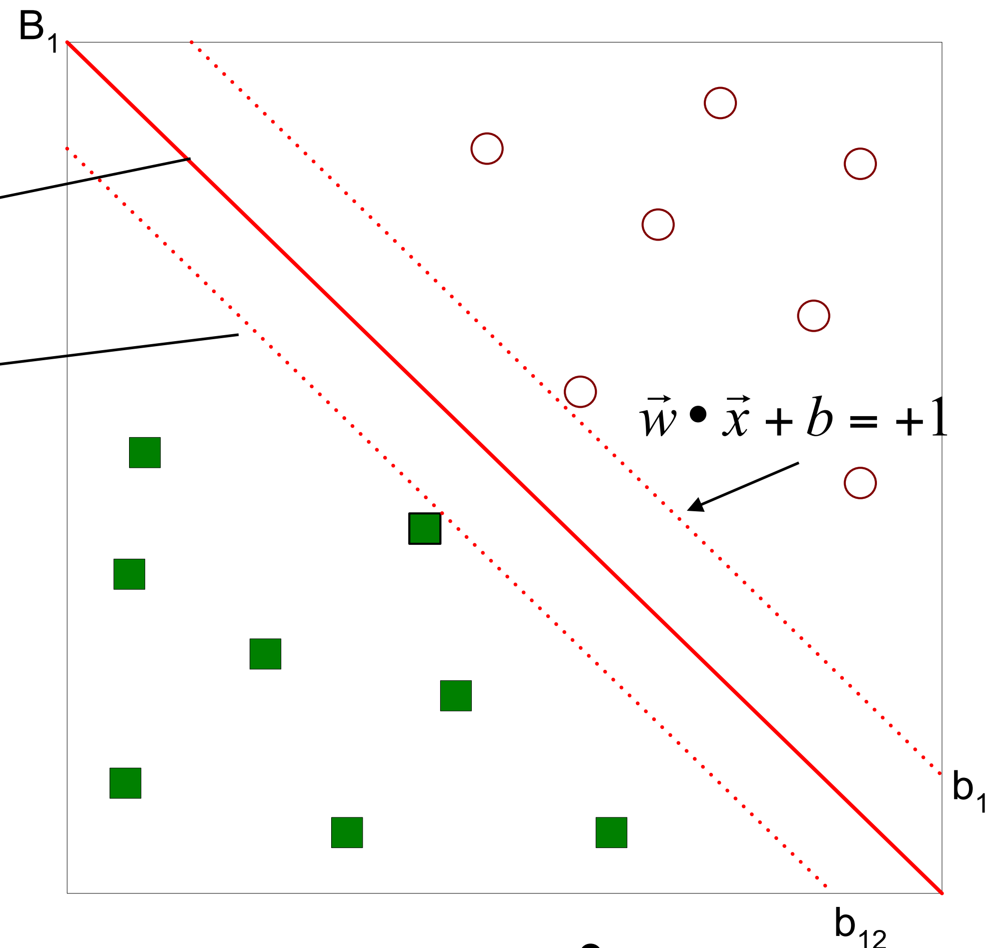


- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

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

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



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

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

Support Vector Machines

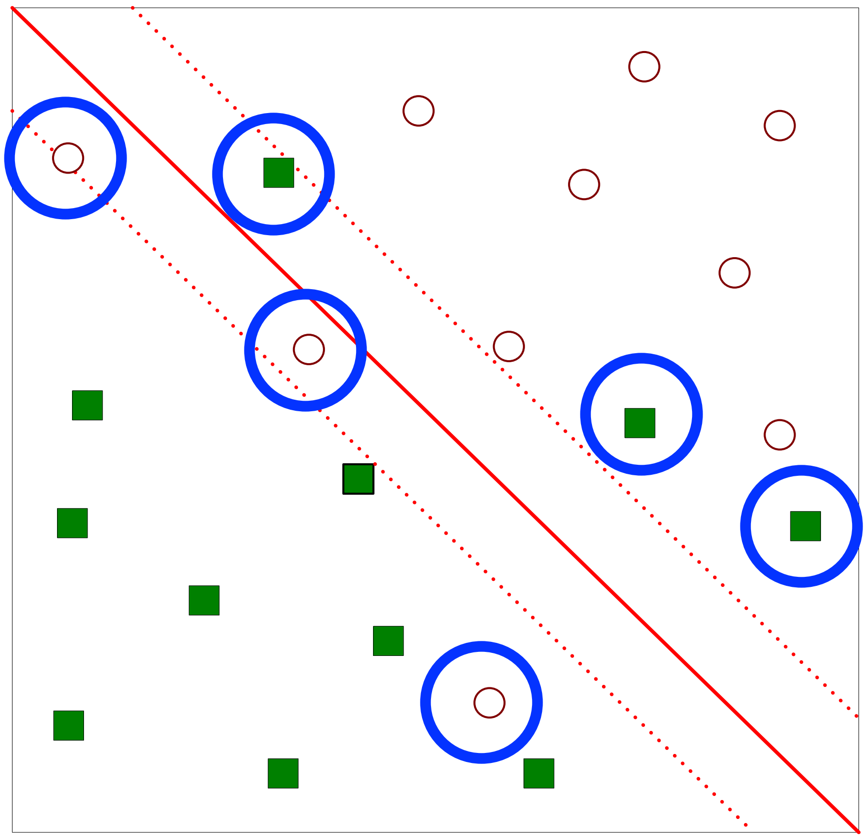
- We want to maximize: $\text{Margin} = \dfrac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \dfrac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

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

    - 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?
  - Introduce slack variables
    - Need to minimize:

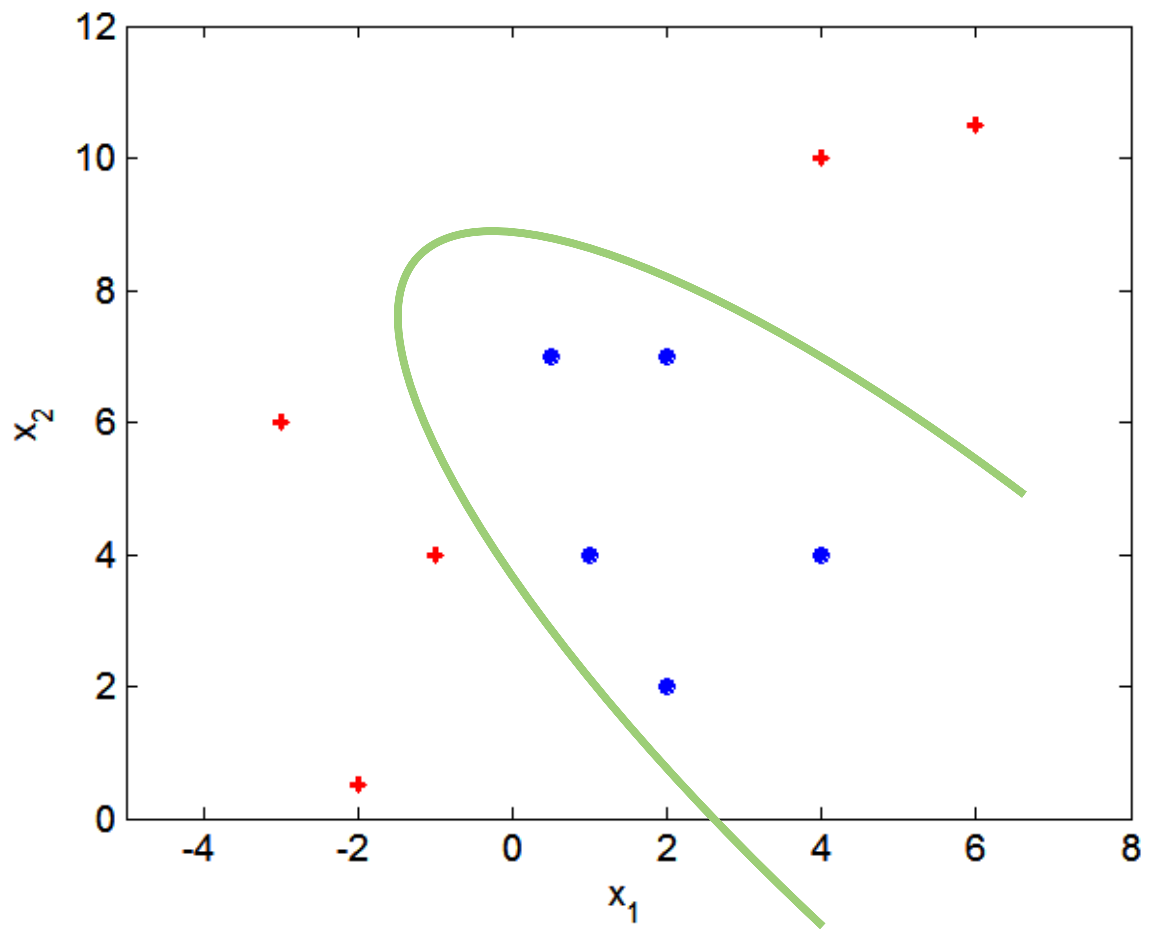
$$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

    - Subject to:

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

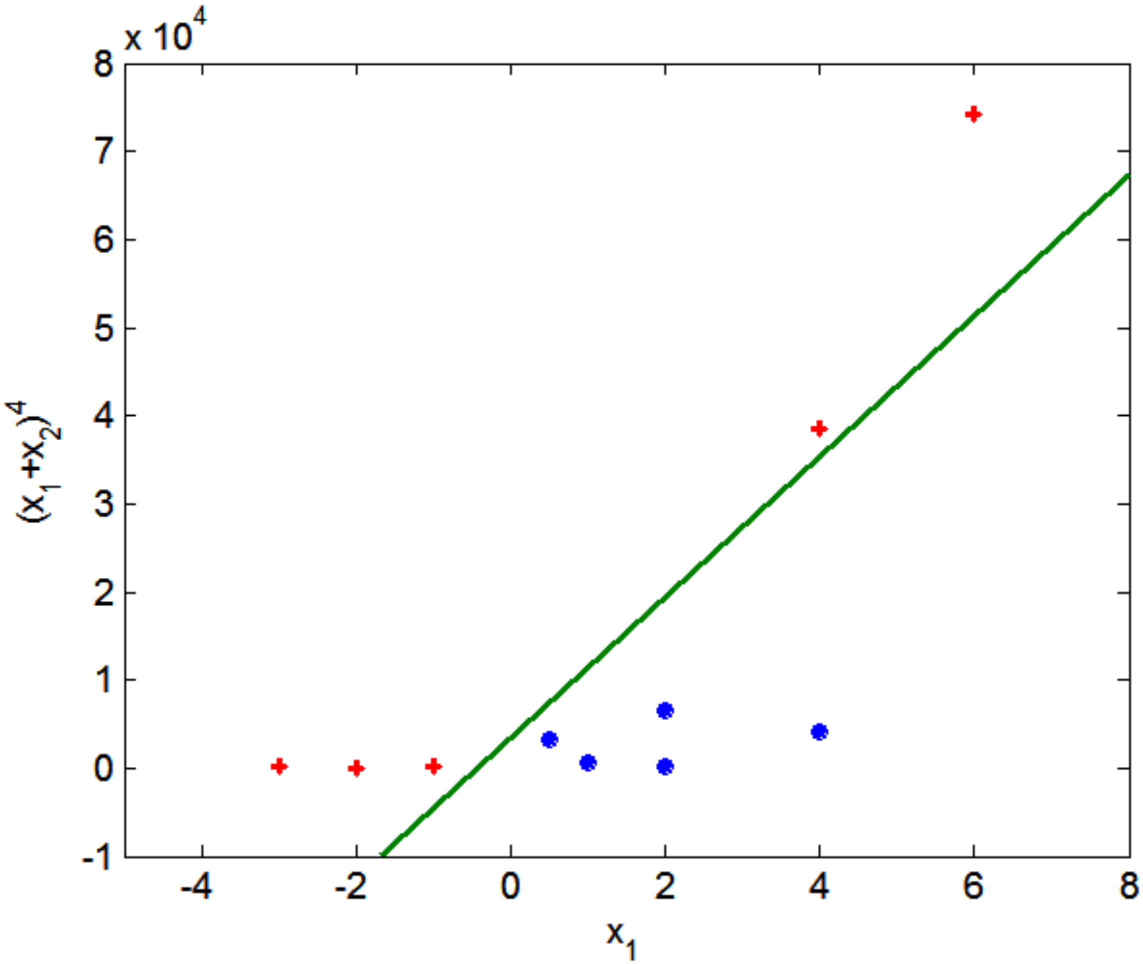
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Why SVMs?

- Convex Convex Convex
  - No trapping in local minimas like Neural Nets.
- SVMs work for categorical and continuous data.
- Can control the model complexity by providing the control on cost function, margin parameters to use.
- Kernel Trick (Not discussed) extends it to non-linear spaces.