



Machine Learning

DSE CLZG565

M1 : Introduction

Raja vadhana P

Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Course Introduction

- **Objective of course**
 - Introduction to the basic concepts and techniques of Machine Learning
 - Gain experience of doing independent study and research in the field of Machine Learning
 - Develop skills of using recent machine learning software tools to evaluate learning algorithms and model selection for solving practical problems
- **Evaluation scheme**
 - **Quiz (10% - 2 quiz: best of 2)**
 - **1 Assignment (20%)**
 - **Mid-semester exam (30%)**
 - **Comprehensive exam (40%)**

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M5 Linear Models for Classification

M6 Linear Models for Regression

M7 Decision Tree

M3 & M4 Bayesian Learning & Bayesian Classifiers

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Pre-requisites

- Linear algebra: vector/matrix manipulations, properties
 - Calculus: partial derivatives
 - Probability: common distributions; Bayes Rule
 - Statistics: mean/median/mode; maximum likelihood
-

Agenda

-
- What is Machine Learning?
 - Why Machine Learning is important?
 - Types of Machine Learning
-

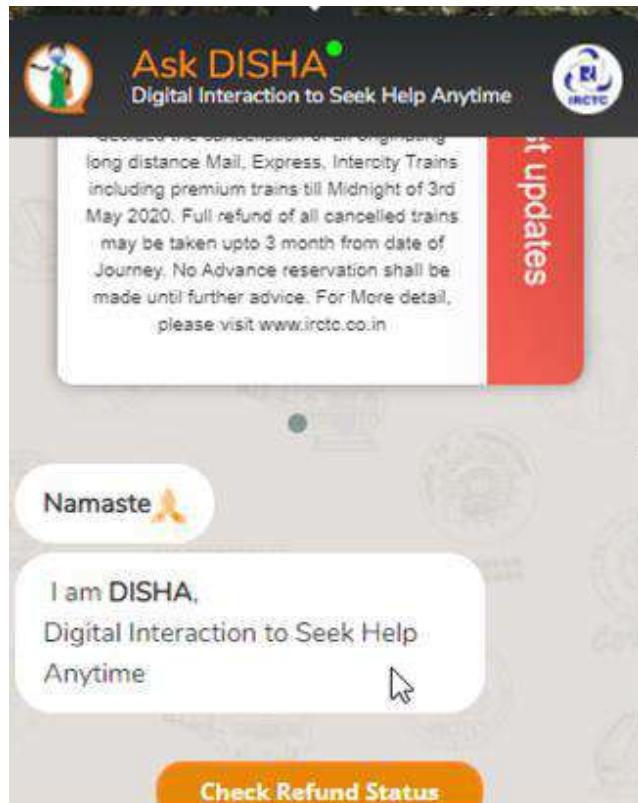


Common Applications

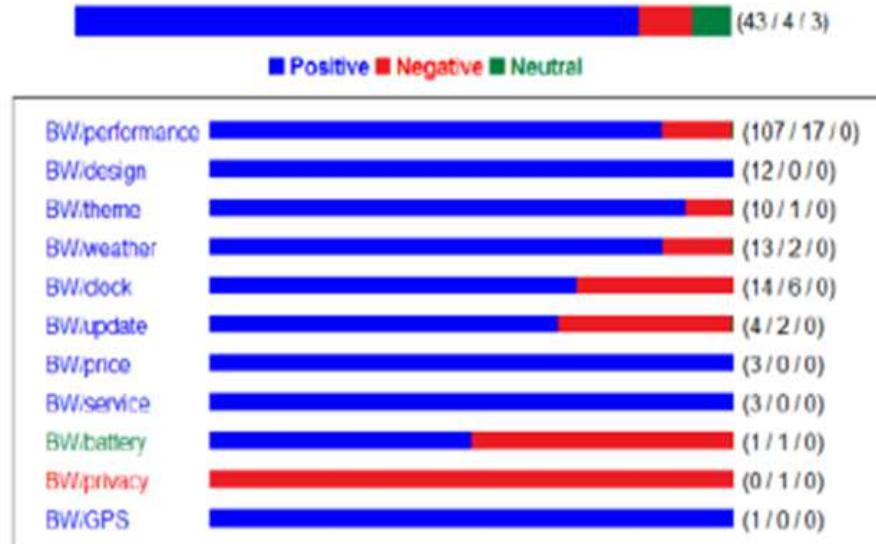
Introduction to Machine Learning



Common Applications



Chatbot



Sentiment analysis on Product review of Mobile phone

Introduction to Machine Learning



Common Applications

amazon.com

Recommended for You

Amazon.com has new recommendations for you based on items you purchased or told us you own.



[The Little Big Things: 163 Ways to Pursue EXCELLENCE](#)



[Fascinate: Your 7 Triggers to Persuasion and Captivation](#)



[Sherlock Holmes \[Blu-ray\]](#)



[Alice in Wonderland \[Blu-ray\]](#)

Recommended for you, Thomas

Exercise & Fitness Equipment	Health, Fitness & Dieting Books	Tableware
Management & Success	Technology & Engineering	Business & Finance

amazon.ca
Prime

Your Amazon.ca | Deals | See All Departments

Josh Reyes,

Are you looking for something in our Sandals department? If so, you might be interested in these items.

Sandals



[Birkenstock Women's Gizeh Cork Footbed Thong Sandal](#)
by Birkenstock

[Learn more](#)

[Add to Wish List](#)

Price: **CDN\$ 117.89 - CDN\$ 168.63**
Ships from and sold by [Byward Centre](#).



[Birkenstock Women's Gizeh Cork Footbed Thong Sandal](#)
by Birkenstock

[Learn more](#)

[Add to Wish List](#)

Price: **CDN\$ 104.60 - CDN\$ 146.98**



ML – What, When, Where ?

Introduction to Machine Learning



What is Machine Learning (ML)?

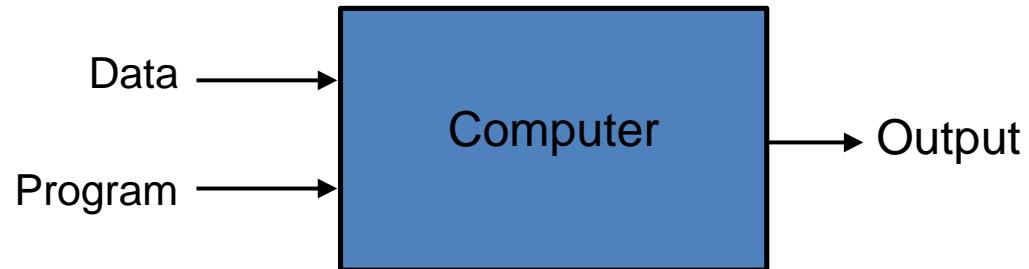
- The science (and art) of programming computers so they can *learn from data*

- More general definition
Field of study that gives computers the ability to learn without being explicitly programmed

- Engineering-oriented definition
Algorithms that improve their performance P at some task T with experience E

A well-defined learning task is given by $\langle P, T, E \rangle$

Traditional programming



Machine Learning



Slide credit: Pedro Domingos

Introduction to Machine Learning



Traditional Approach - Spam Filtering

Spam typically uses words or phrases such as “4U,” “credit card,” “free,” and “amazing”

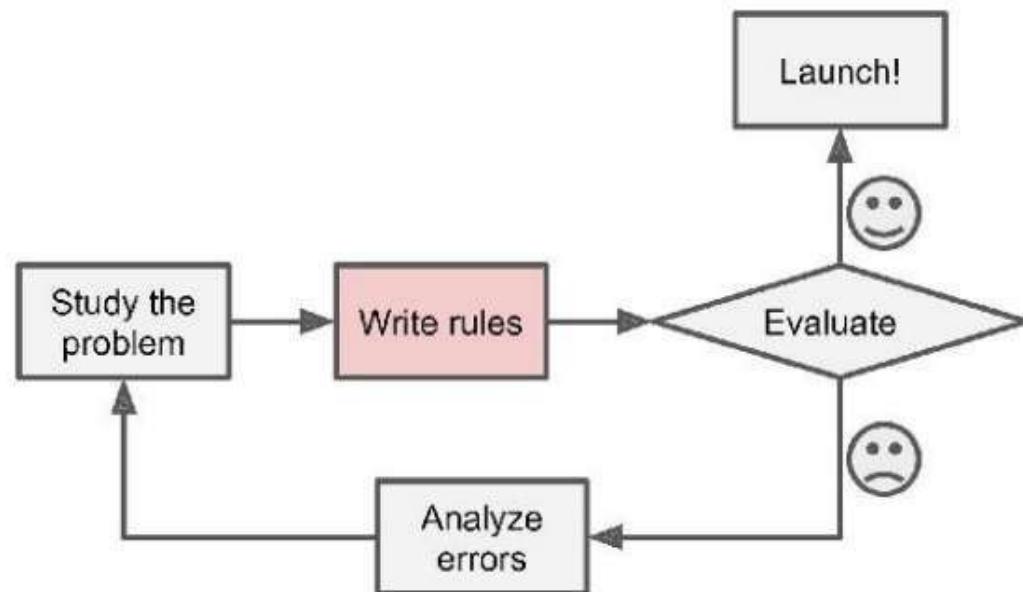
Solution

Write a detection algorithm for frequently appearing patterns in spams

Test and update the detection rules until it is good enough.

Challenge

Detection algorithm likely
to be a long list of complex rules
hard to maintain.

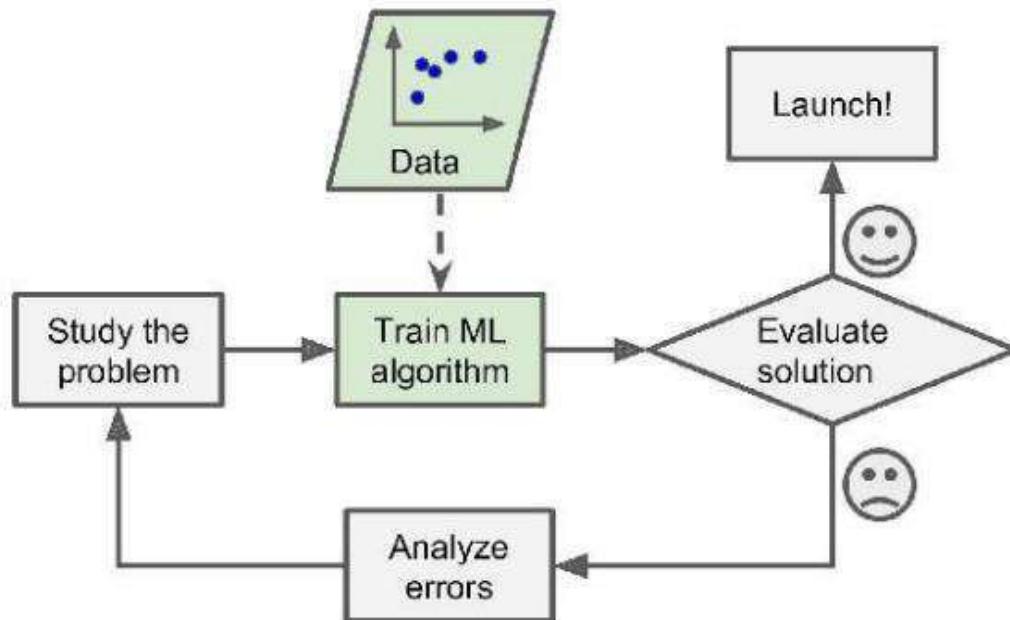


Introduction to Machine Learning



ML Approach - Spam Filtering

Automatically learns phrases that are good predictors of spam by detecting unusually frequent patterns of words in spams compared to “ham”s



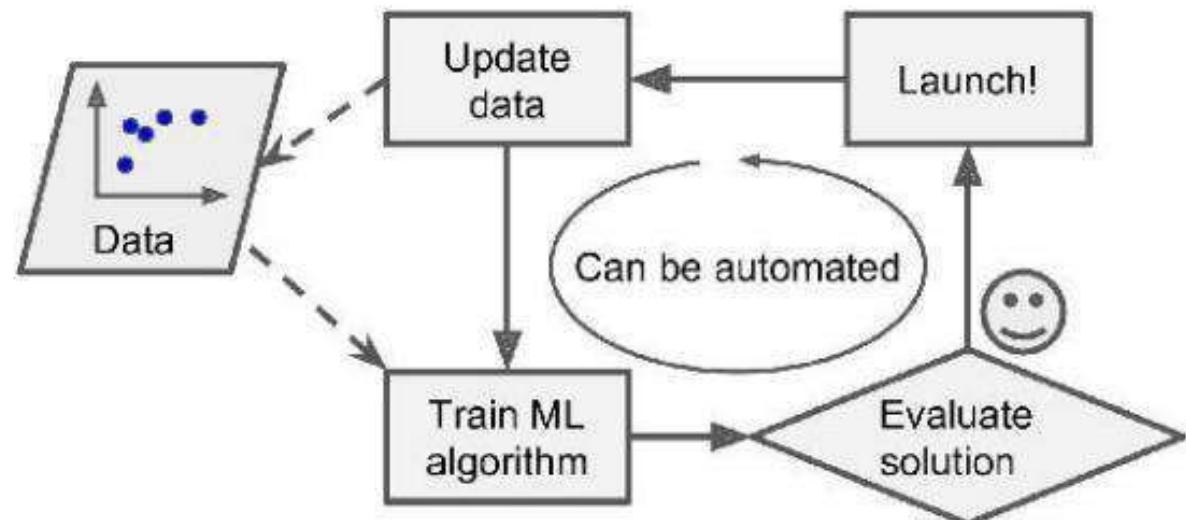
The program is much shorter, easier to maintain, and most likely more accurate.

Introduction to Machine Learning



ML Approach - Spam Filtering

Automatically learns phrases that are good predictors of spam by detecting unusually frequent patterns of words in spams compared to “ham”s



The program is much shorter, easier to maintain, and most likely more accurate.



Why ML

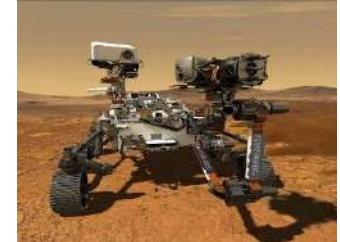
Why is Machine Learning Important ?

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.

When Do We Use Machine Learning?

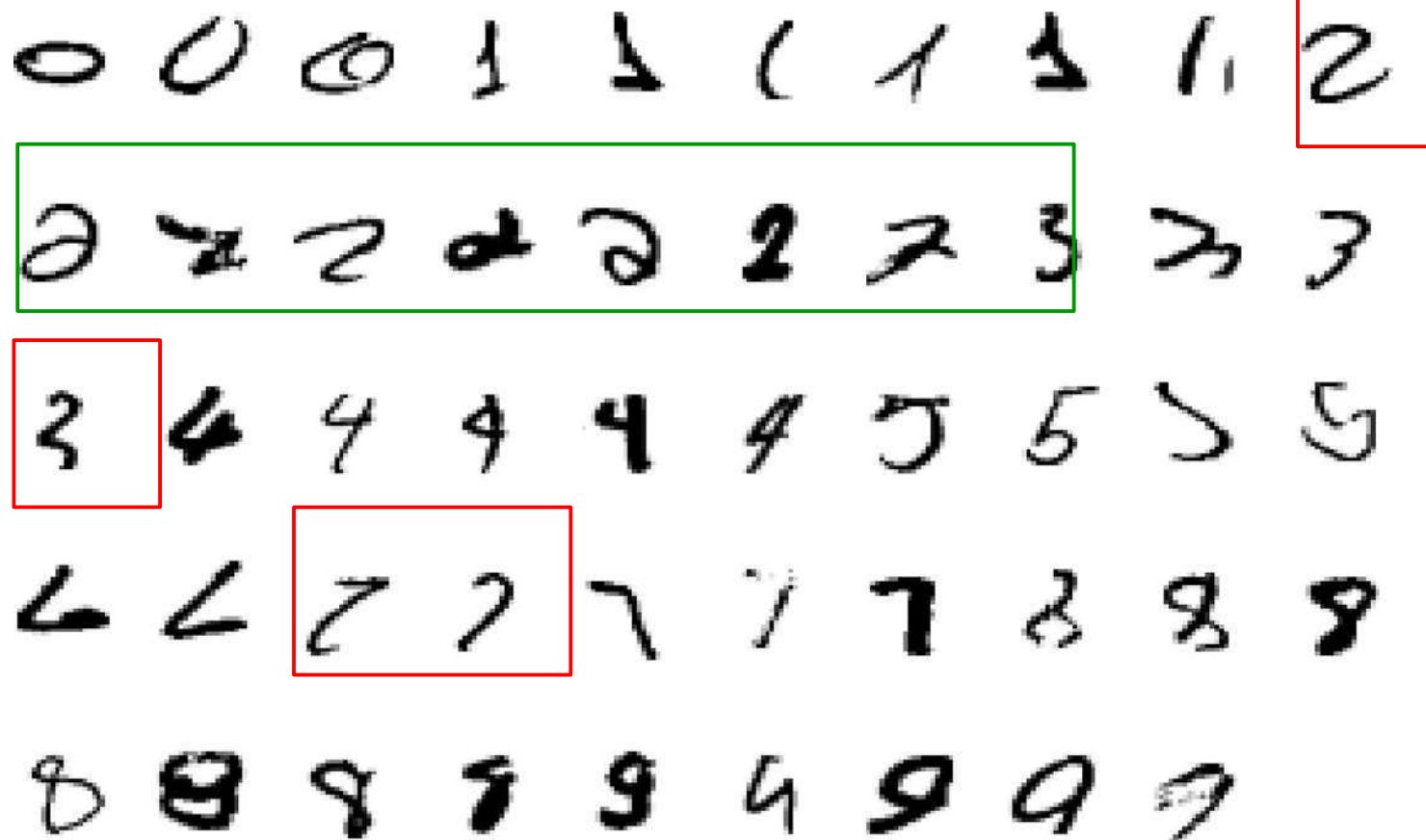
ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (Biometrics)
- Models must be customized (personalized medicine)
- **Learning isn't always useful:**
 - There is no need to “learn” to calculate payroll



Pattern recognition

It is very hard to say what makes a 2



Representation Example

- Apply a prediction function to a feature representation of the image to get the desired output:

$$f(\text{apple}) = \text{"apple"}$$
$$f(\text{tomato}) = \text{"tomato"}$$
$$f(\text{cow}) = \text{"cow"} \quad y = f(x)$$

output prediction function features



Types of ML

Types of Learning

- **Supervised (inductive) learning**
 - Given: training data, desired outputs (labels)
- **Unsupervised learning**
 - Given: training data (without desired outputs)
- **Semi-supervised learning**
 - Given: training data + a few desired outputs
- **Reinforcement learning**
 - Given: rewards from sequence of actions

Machine Learning - Examples

Objective: Employability Prediction

Features / Attributes / Predictors

- ✓ CGPA
- ✓ Communication Skills
- ✓ Aptitude
- ✓ Programming Skills

S.No.	CGPA	Communication Skills	Aptitude	Programming Skills	Job Offered?
1	9.1	Average	Good	Excellent	Yes
2	8.4	Good	Good	Good	Yes
3	8.3	Poor	Average	Average	No
4	7.1	Average	Good	Average	No
5	8.2	Good	Excellent	Excellent	No

Machine Learning - Examples

Objective : Predicting price of a used car

Features / Attributes / Predictors

- ✓ Brand
- ✓ Year (Mfg)
- ✓ Engine Capacity
- ✓ Mileage
- ✓ Distance travelled
- ✓ Cab?

S.No	Brand	Year (Mfg)	Engine Capacity	Mileage	Distance travelled	Cab?	Price (in Rs.)
1.	Honda City ZX	2008	1100	10.5	45000	N	3,50,000
2							
3							
4							

Machine Learning - Examples

Objective: Market Segmentation Study

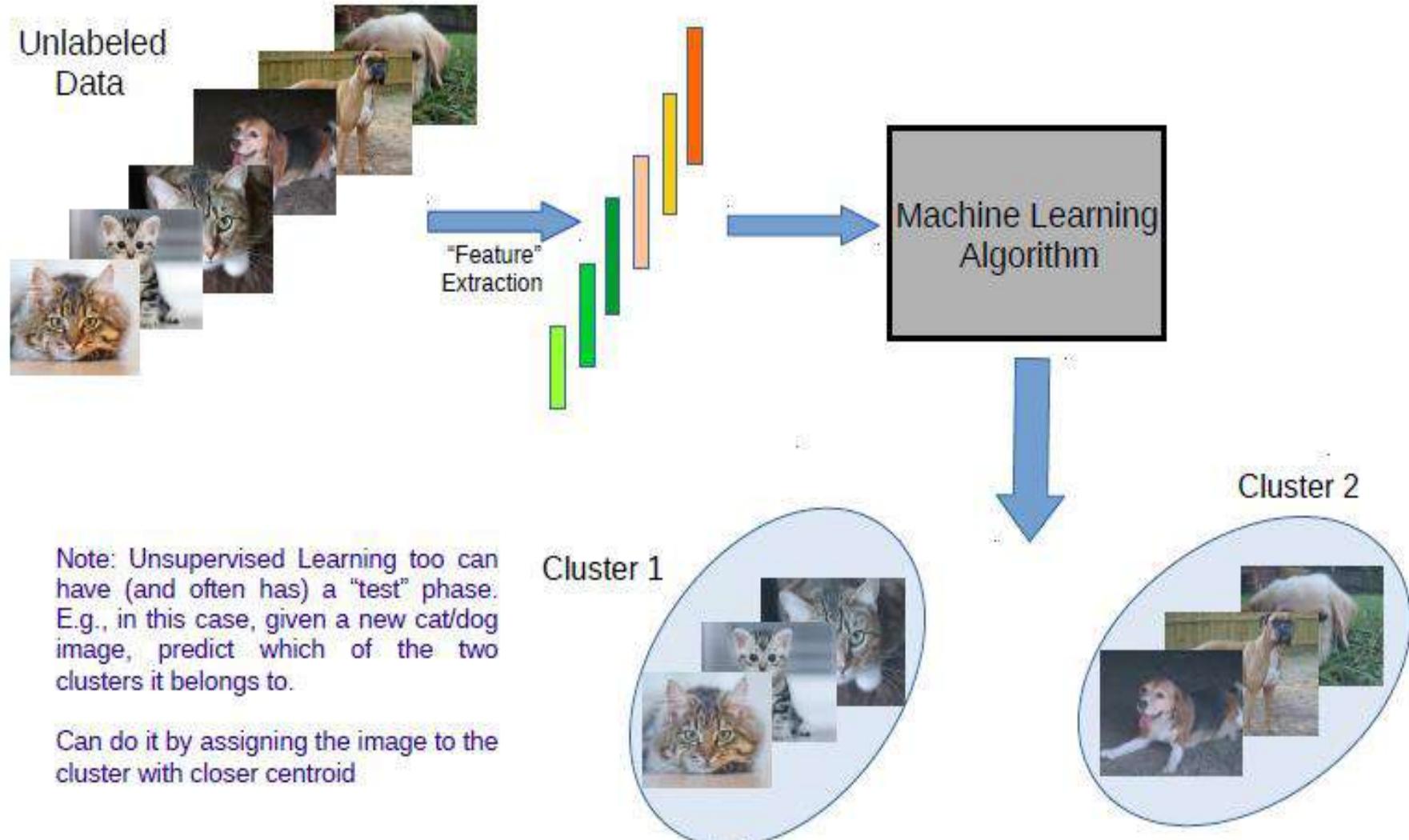
Features / Attributes / Predictors

- ✓ Family income
- ✓ # of visits in a month
- ✓ Average money spent in a month
- ✓ Zip code

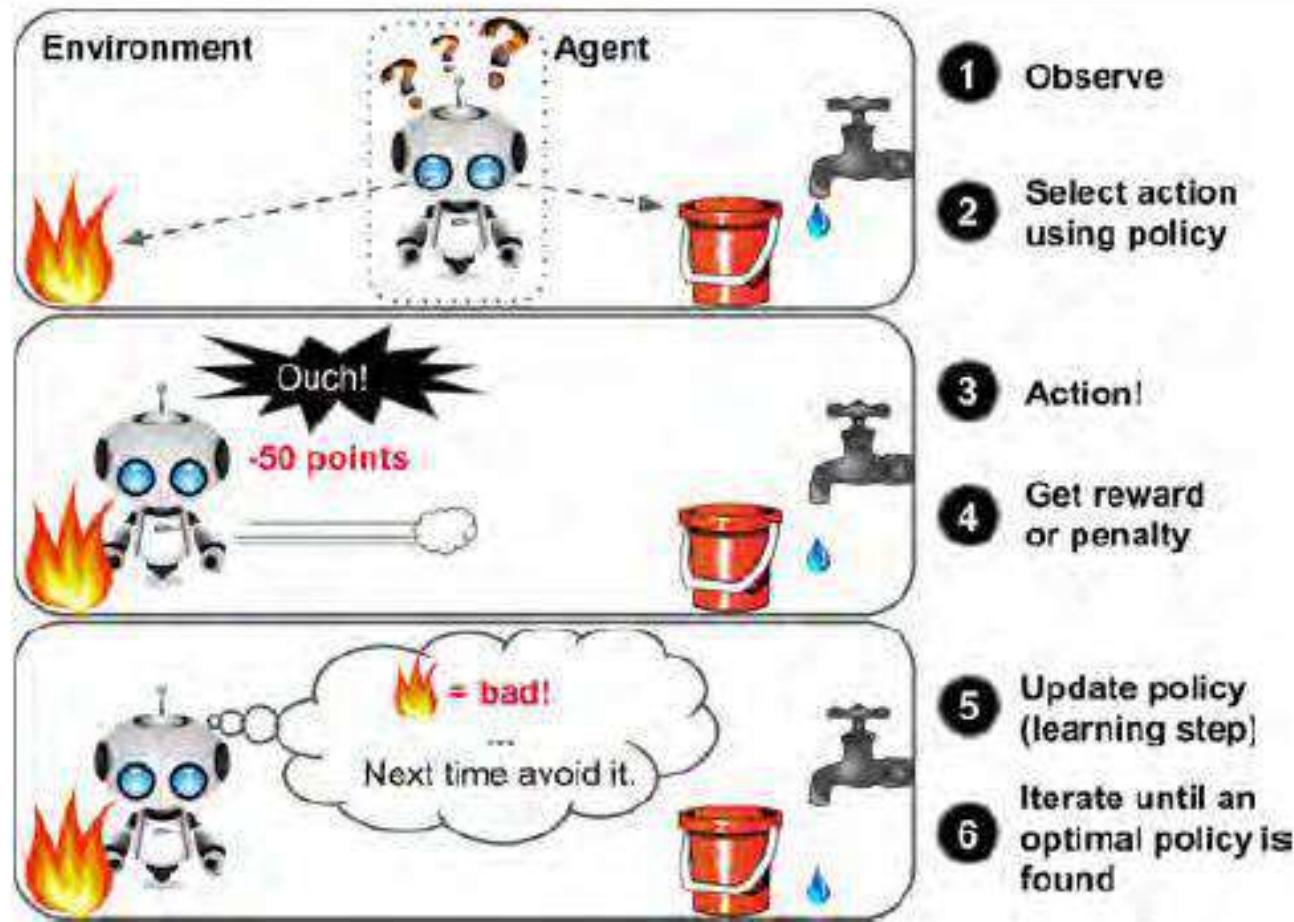
- Customers for a retailer may fall into
- ✓ two groups say big spenders and low spenders
 - ✓ three groups say big spenders, medium spenders and low spenders
 - ✓ Four groups,

S.No .	Zip Code	Family Income	# of visits in a month	Average Money Spent in a month
1	500078	11,50,000	4	8,000

A Typical Unsupervised Learning Workflow (for Clustering)



Reinforcement Learning



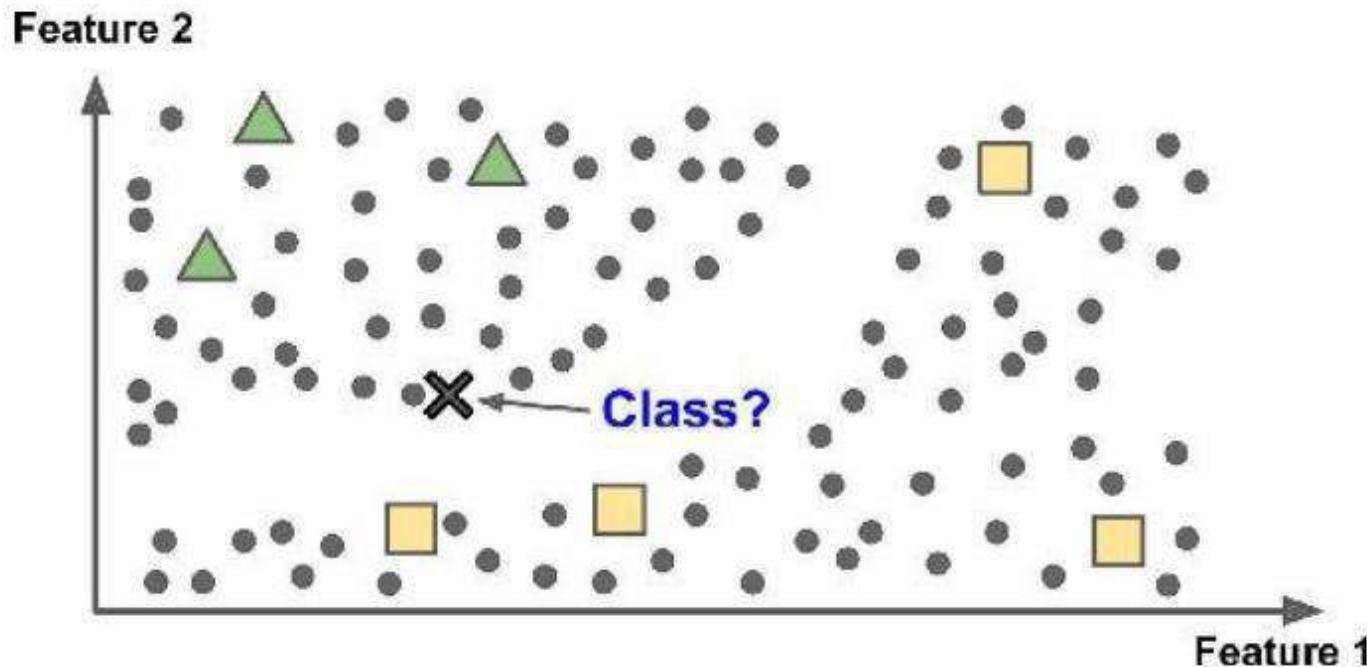
Introduction to Machine Learning



Semi supervised Learning

Partially labelled data – some labelled data and a lot of unlabelled data

- Combines unsupervised and supervised learning algorithms
- Photo hosting service, e.g., google photos

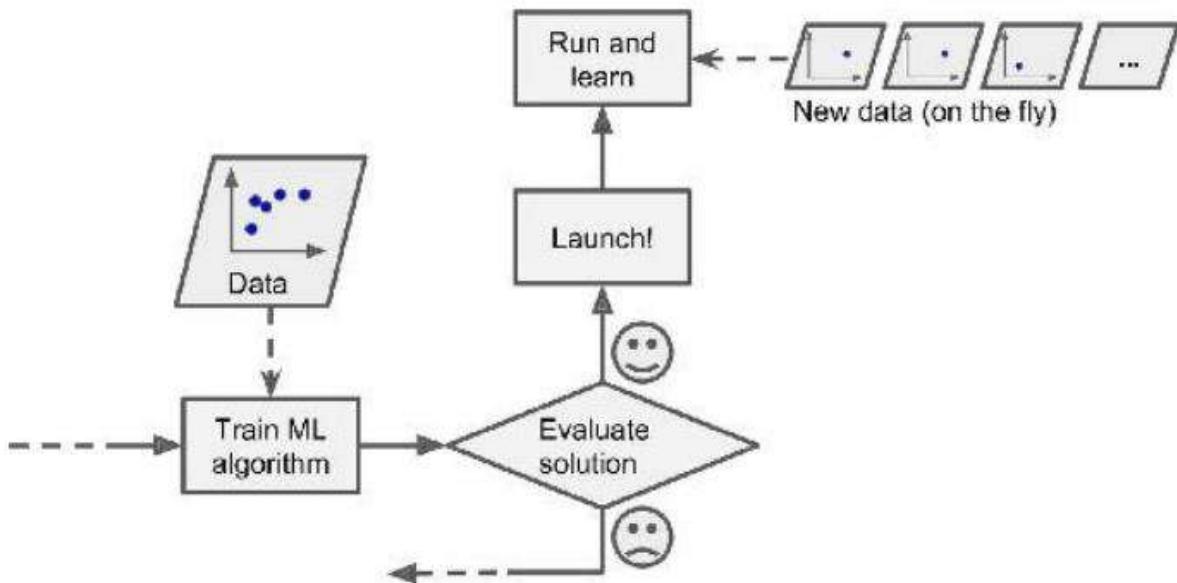


Introduction to Machine Learning



Types: Based on how training data is used

- Batch learning: Uses all available data at a time during training
- Mini Batch learning: Uses a subset of available at a time during training
- Online (incremental) learning: Uses single training data instance at a time during training

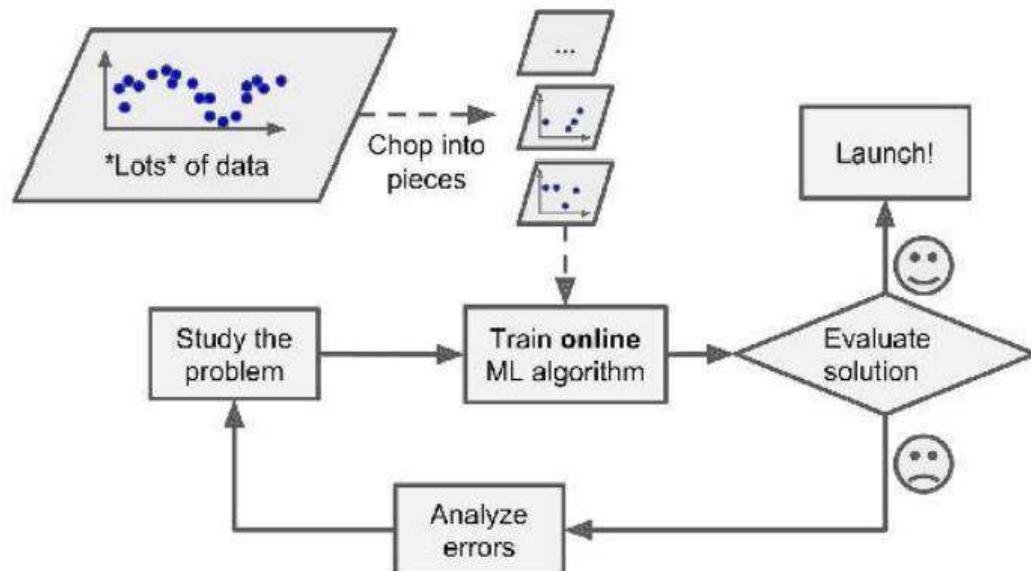


Introduction to Machine Learning



Types: Based on how training data is used

- Batch learning: Uses all available data at a time during training
- Mini Batch learning: Uses a subset of available at a time during training
- Online (incremental) learning: Uses single training data instance at a time during training

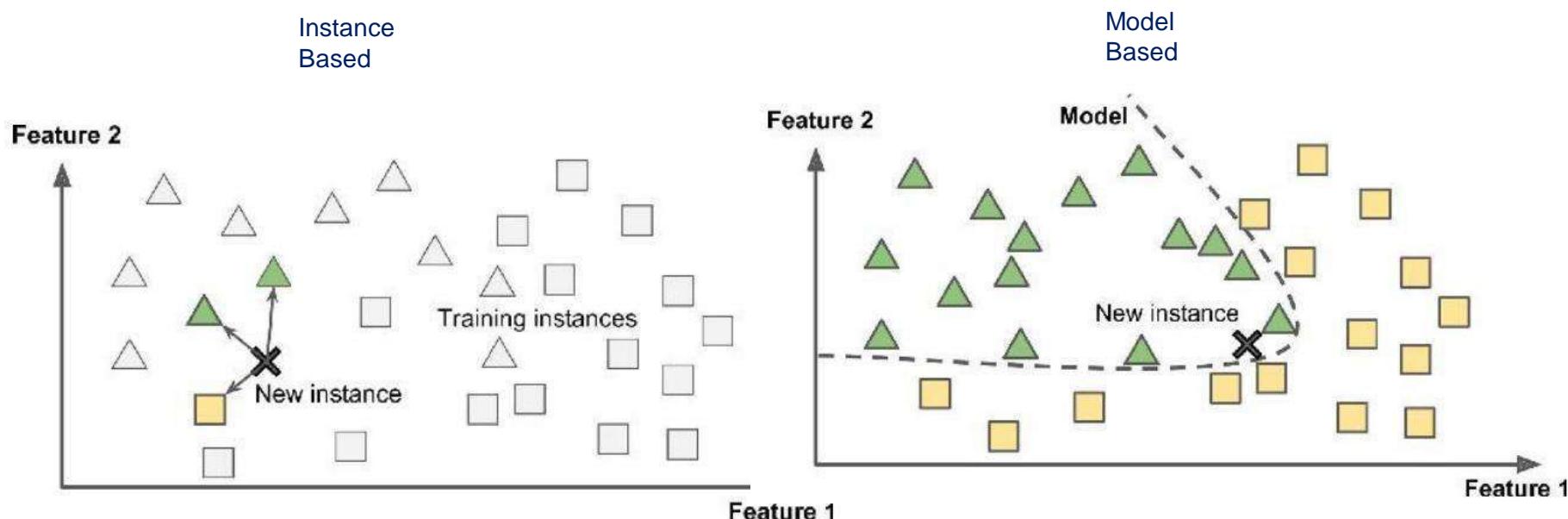


Introduction to Machine Learning

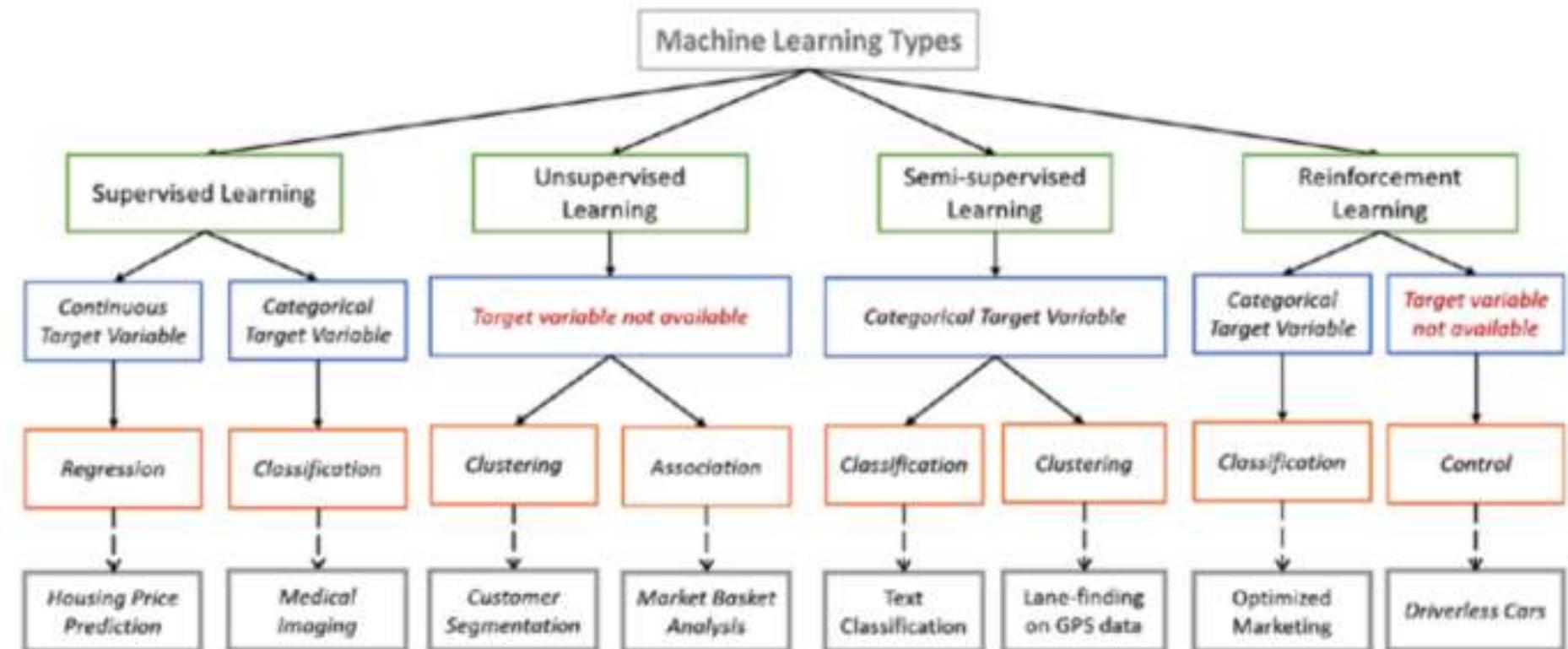


Types: Based on how training data is used

- Instance Based Learning: Compare new data points to known data points
- Model Based learning : Detect patterns in the training data and build a predictive model



Summary : Types of Learning



Thank you !

Required Reading

- Machine Learning, Tom Mitchell
- Read Chapter 1, 2 : Introduction to Machine Learning,
2nd edition, Ethem Alpaydin



Machine Learning

DSE CLZG565

M1 : Introduction

M2 : Mathematical Preliminaries

Raja vadhana P

Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M5 Linear Models for Classification

M6 Linear Models for Regression

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 1

- What is Machine Learning?
- Why Machine Learning is important?
- Types of Machine Learning
 - Supervised Machine Learning
 - Model selection-assessment
 - ML workflow

Module – 2

Prerequisites : Refresher only

- Linear Algebra
- Calculus
- Probability Theory

Learnt so far in previous session

- Notion of ML
- Need for ML
- Types of ML
 - Based on feedback
 - Based on data availability during learning
 - Based on data usage for learning
- Few Terminologies
 - Supervised/Unsupervised/Reinforcement
 - Features/Attributes/Predictors
 - Target/Output
 - Model/Hypothesis
 - Coefficients/Weights of the predictors

Agenda

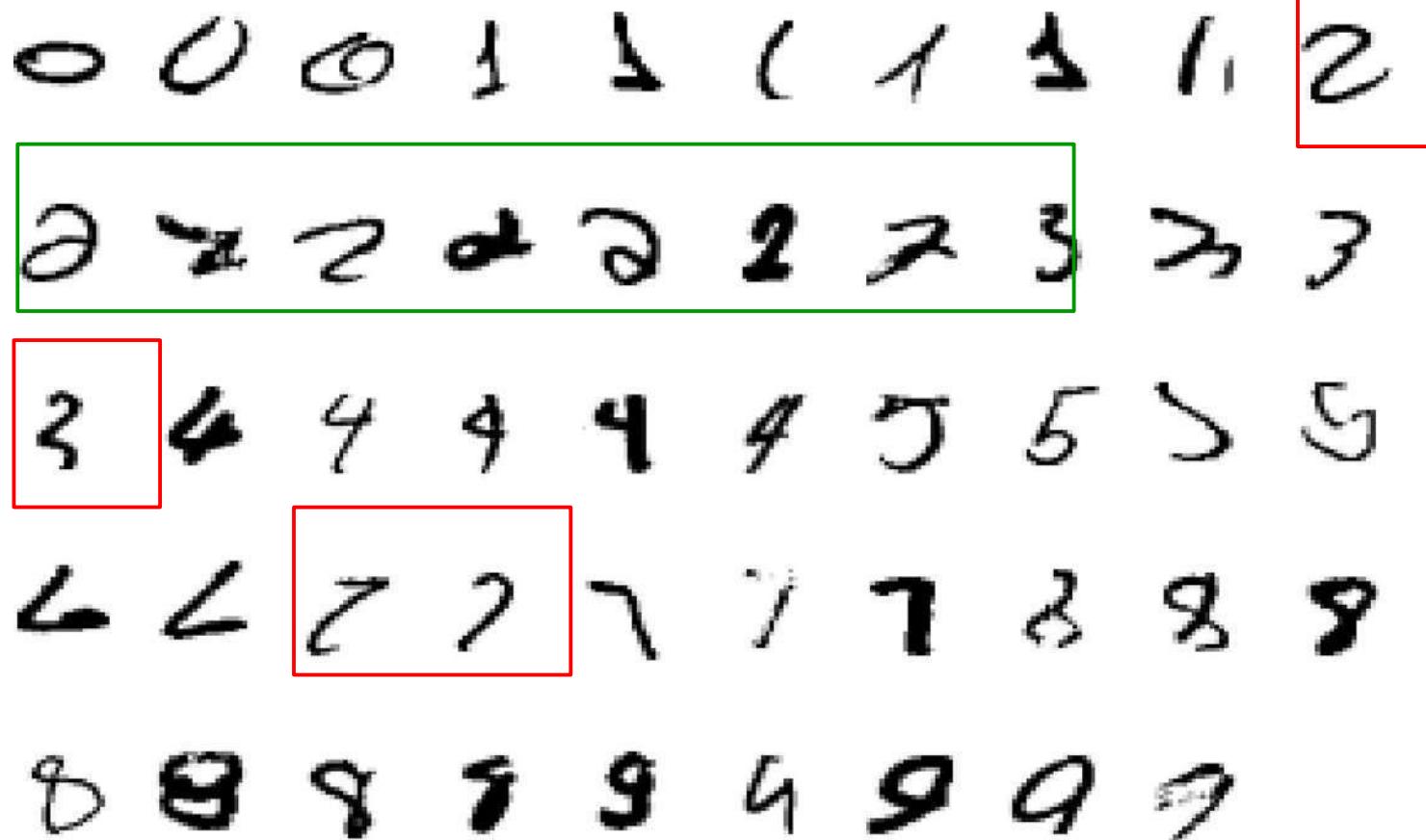
- Machine Workflow
- Formal Representations & few terminologies
- General Challenges in Learning
- Math Preliminaries - Refresher



Formal Representations

Pattern recognition

It is very hard to say what makes a 2



Is the training examples sufficient?

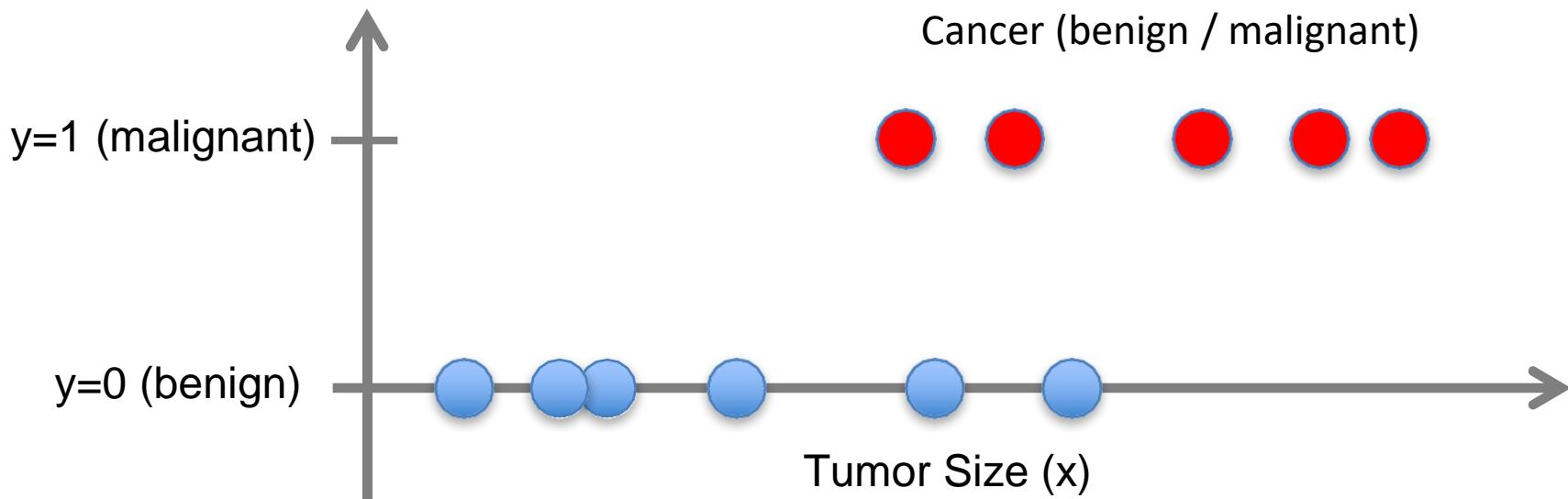
Introduction to Machine Learning



Supervised Learning: Classification

GOAL : Previously unseen records should be assigned a class as accurately as possible.

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is categorical



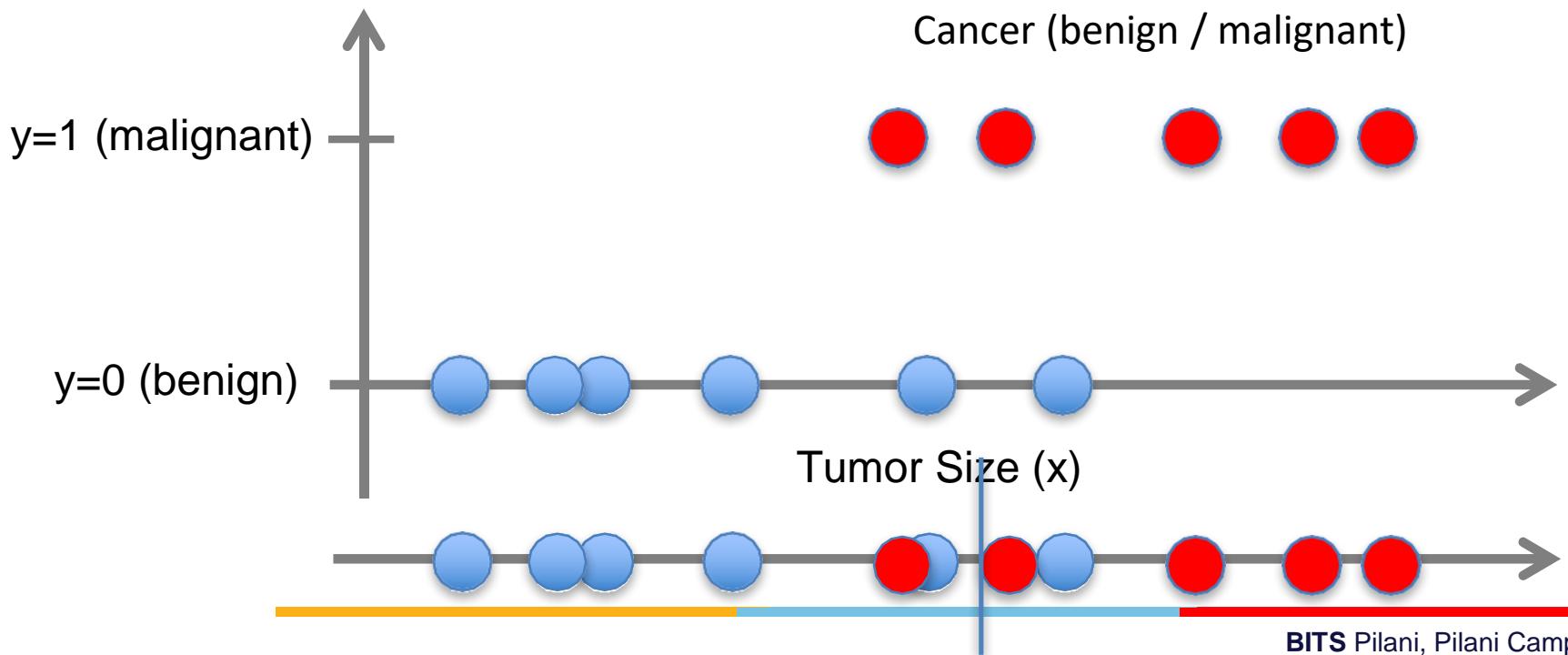
Introduction to Machine Learning



Supervised Learning: Classification

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is categorical

Learnt classifier
If $x > T$, malignant else benign



Introduction to Machine Learning

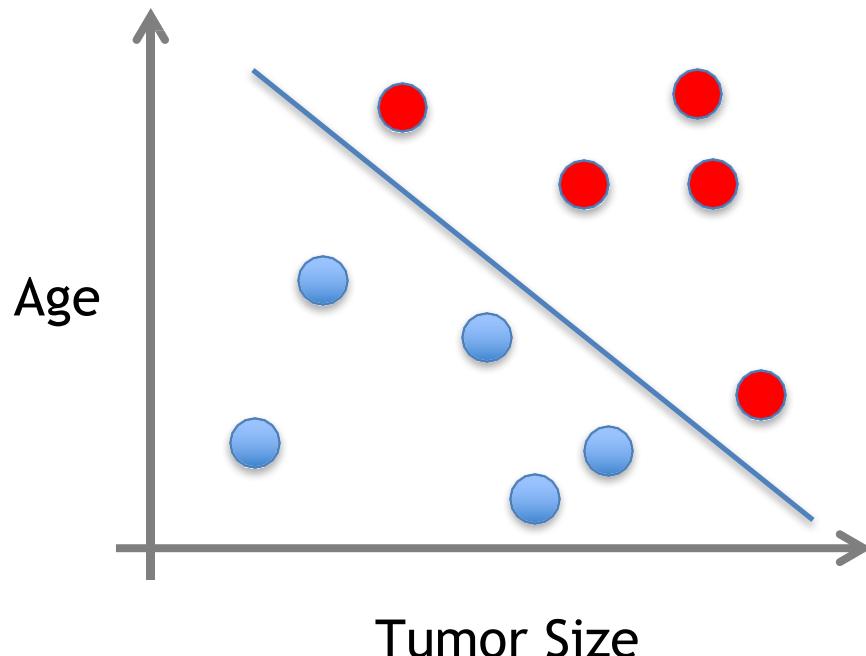


Supervised Learning: Classification

- x can be multi-dimensional
 - Each dimension corresponds to an attribute

Increasing Feature Dimension

- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
- ...



Supervised Learning Techniques / Algorithms

- Linear Regression
- Logistic Regression
- Naïve Bayes Classifiers
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Introduction to Machine Learning

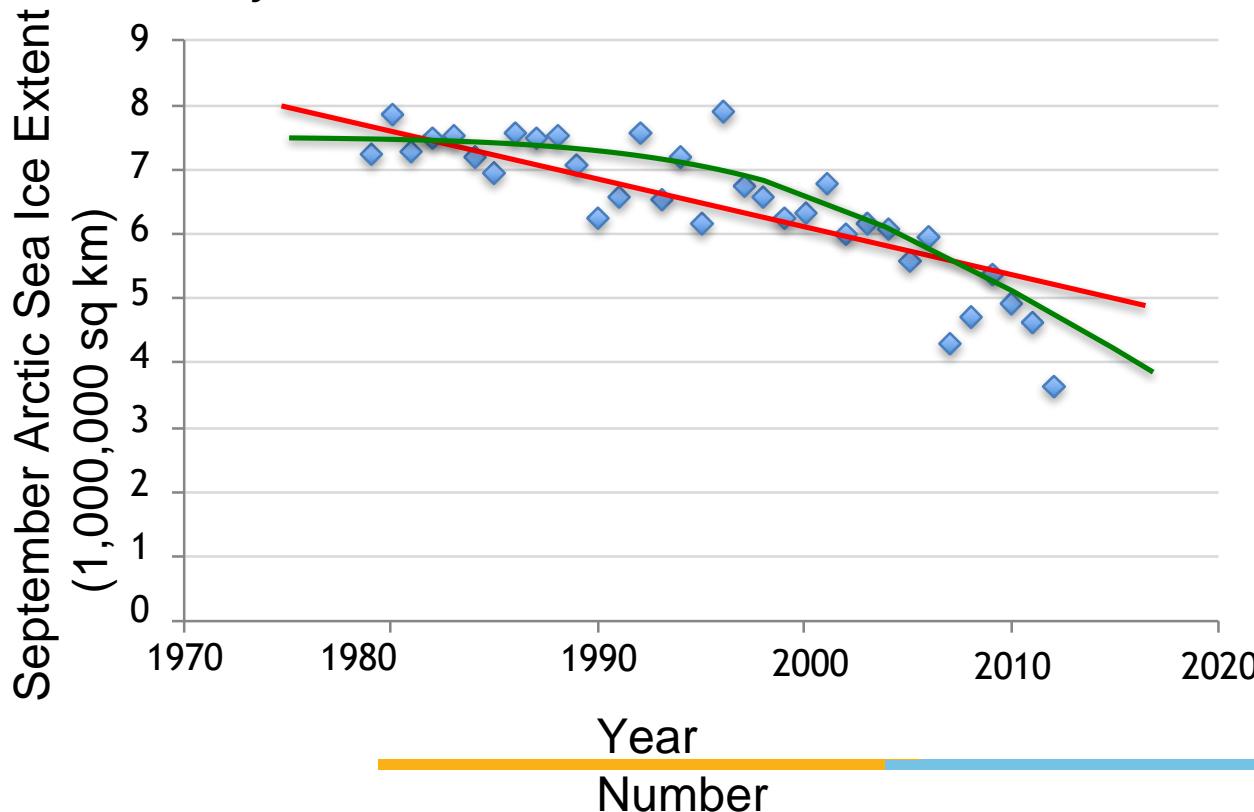


Supervised Learning: Regression

GOAL : Previously unseen records should be assigned a value as accurately as possible.

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is real-valued

Is all the training examples useful?



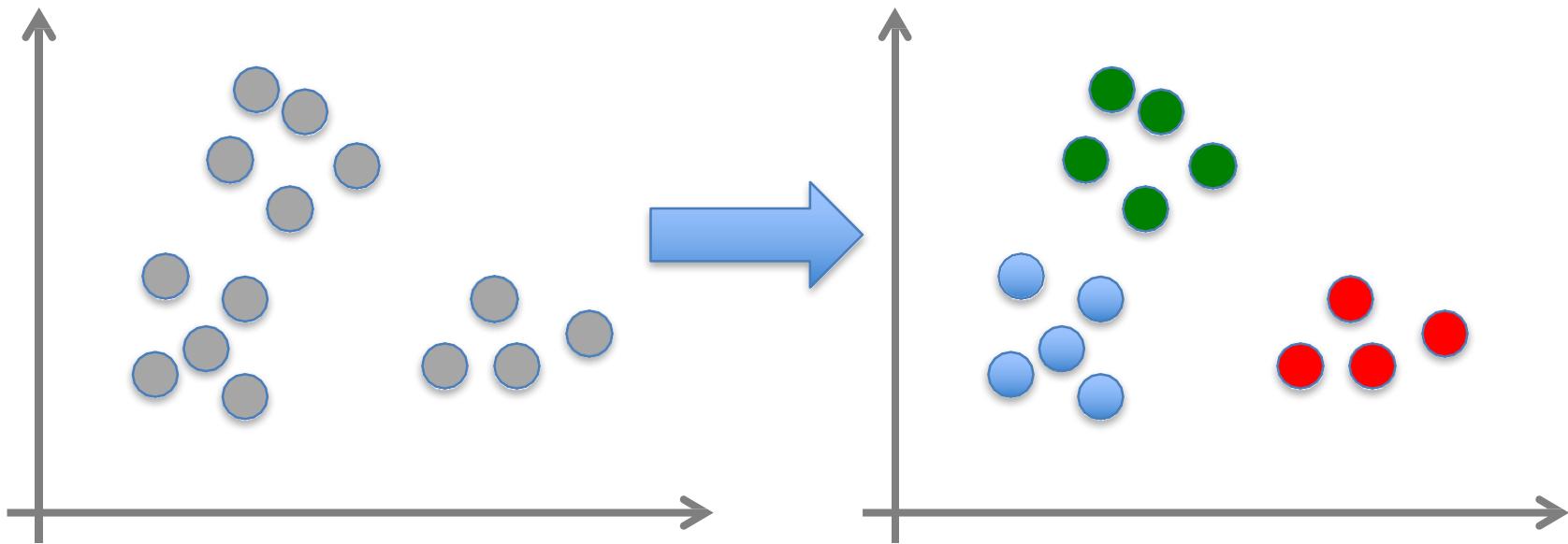
Introduction to Machine Learning



Unsupervised Learning

GOAL : Intra cluster distances are minimized and inter cluster distances are maximized

- Given x_1, x_2, \dots, x_n (without labels)
- Output hidden structure behind the x 's
 - e.g., clustering



Unsupervised Learning Techniques

Clustering

- k-Means
- Hierarchical Cluster Analysis
- Expectation Maximization

Visualization and dimensionality reduction

- Principal Component Analysis (PCA)
- Kernel PCA
- Locally-Linear Embedding (LLE)
- t-distributed Stochastic Neighbor Embedding (t-SNE)



Supervised (inductive) learning

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.
- **Target function (target concept).** The true function f .
- **Hypothesis.** A proposed function h believed to be similar to f .
- **Concept.** A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances**.
- **Classifier.** A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \dots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space.** The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space.** The space of all hypotheses in the hypothesis space that have not yet been ruled out by a training example.

Inductive Learning Hypothesis

- Input Representation : $\langle x, f(x) \rangle$
- Machine Learning Task : h that approximates f , such that $h(x) = f(x)$
 - $h_0: \langle \text{Sunny}, ?, ?, ?, ?, ? \rangle \rightarrow \text{Enjoy Sport}$
 - $h_1: \langle ?, ?, ?, \text{Strong}, \text{Warm}, ? \rangle \rightarrow \text{Enjoy Sport}$
 - $h_2: \langle \text{Sunny}, \text{Warm}, ?, ?, ?, ? \rangle \rightarrow \text{Enjoy Sport}$
 - $h_3: \langle \text{Rainy}, ?, \text{High}, ?, ?, ? \rangle \rightarrow \text{Don't Enjoy Sport}$
 - $h_4: \langle \text{Sunny}, \text{Warm}, ?, ?, ?, \text{Same} \rangle \rightarrow \text{Enjoy Sport}$
- Machine Learning Objective : $\text{Consistent}(h, D) \equiv h(x) = f(x) \quad \forall (x, f(x)) \in D$

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Inductive Learning Hypothesis

- Hypothesis Space : { h0, h1,h972}
- Version Space : { h0, h2, h3, h4,}
 - h0: <Sunny, ?, ?, ?, ?, ?> -> Enjoy Sport
 - h1: <?, ?, ?, Strong, Warm, ?> -> Enjoy Sport
 - h2: <Sunny, Warm, ?, ?, ?, ?> -> Enjoy Sport
 - h3: <Rainy, ?, High, ?, ?, ?> -> Don't Enjoy Sport
 - h4: <Sunny, Warm, ?, ?, ?, Same> -> Enjoy Sport
- $h_0 \geq_g h_2$
- $h_4 \geq_s h_2$

$$VS_{\mathcal{H},D} = [h \mid (h \in \mathcal{H}) \wedge (\text{Consistent}(h, D)) \wedge (h_{\text{most-general}} \geq_g h \geq_g h_{\text{most-specific}})]$$

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Inductive Learning Hypothesis

- Hypothesis Space : { h0, h1,h972}
 - Version Space : { h0, h2, h3, h4,}
 - h0: <Sunny, ?, ?, ?, ?, ?> -> Enjoy Sport
 - h1: <?, ?, ?, Strong, Warm, ?> -> Enjoy Sport
 - h2: <Sunny, Warm, ?, ?, ?, ?> -> Enjoy Sport
 - h3: <Rainy, ?, High, ?, ?, ?> -> Don't Enjoy Sport
 - h4: <Sunny, Warm, ?, ?, ?, Same> -> Enjoy Sport
 - $h_0 \geq_g h_2$
 - $h_4 \geq_s h_2$
- $VS_{\mathcal{H},D} = [h \mid (h \in \mathcal{H}) \wedge (\text{Consistent}(h, D)) \wedge (h_{\text{most-general}} \geq_g h \geq_g h_{\text{most-specific}})]$

Occam's razor: the simplest consistent hypothesis about the target function is actually the best

Minimum features: Unless there is good evidence that a feature is useful, it should be deleted. This is the assumption behind feature selection algorithms

Inductive Learning Hypothesis

- Target Concept
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Humidity	Wind	Water	Forecast	<i>EnjoySport?</i>
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Inductive Learning Hypothesis

- Target Concept
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

Inductive Learning Hypothesis

- Target Concept
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Humidity	Wind	Water	Forecast	$P(\text{EnjoySport} = \text{Yes})$
Sunny	Warm	Normal	Strong	Warm	Same	0.95
Sunny	Warm	High	Strong	Warm	Same	0.7
Rainy	Cold	High	Strong	Warm	Change	0.5
Sunny	Warm	High	Strong	Cool	Change	0.6

Another Example

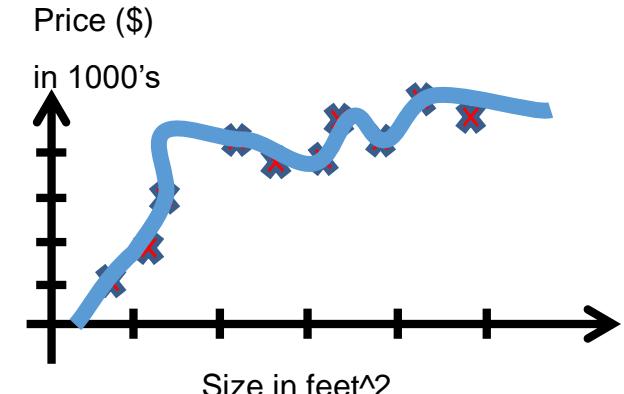
- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- :
- x_{100}

Three Components of ML Algorithms:

- Representations
- Optimizations
- Evaluations

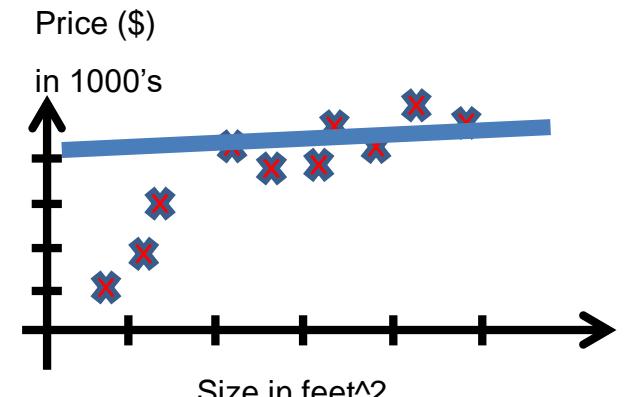
$$h_0: \text{Price} = 0.001 + 1.5 \text{ Size} + 0.12 \text{ Size}^2$$

$$h(x) = w_0 + w_1 x + w_2 x^2 + \dots$$



$$h_1: \text{Price} = 4020 + 0.15 \text{ Size}$$

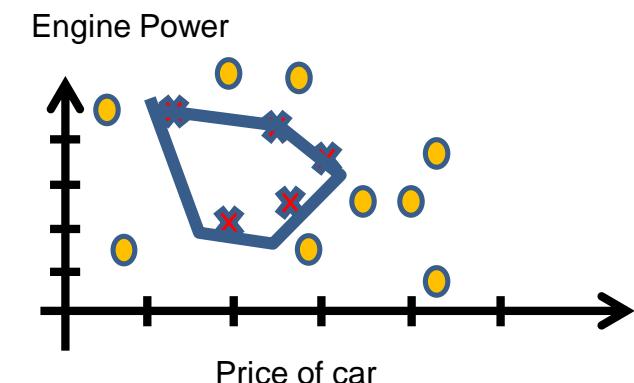
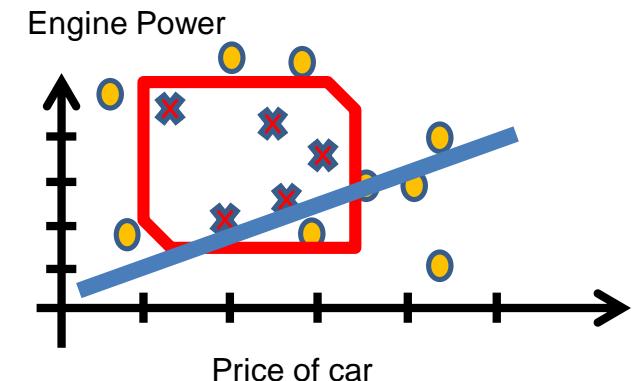
$$h(x) = w_0 + w_1 x$$



Source Credit : Slide by Andrew Ng

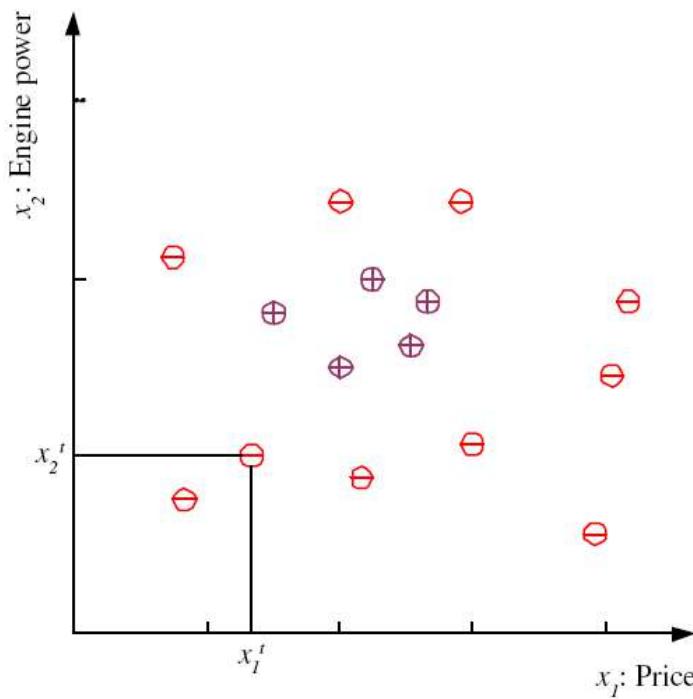
Another Example

- x_1 = price of the car
- x_2 = engine power
- :
- x_m



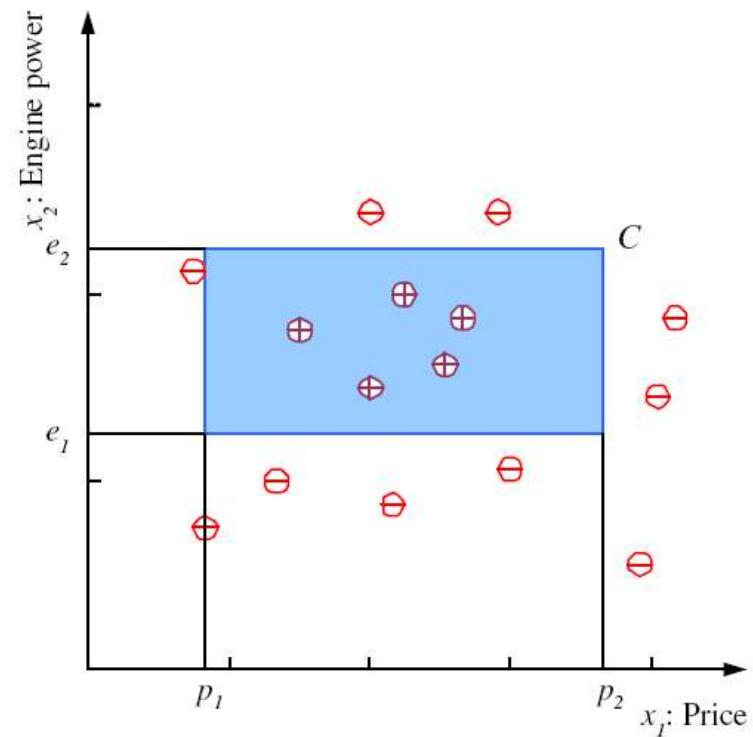
Classification

Training set X



Class C of a car

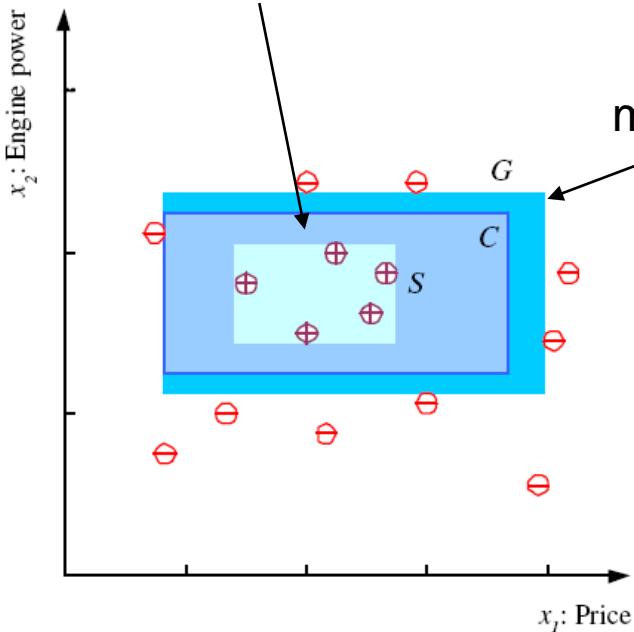
$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$



S, G, and the Version Space



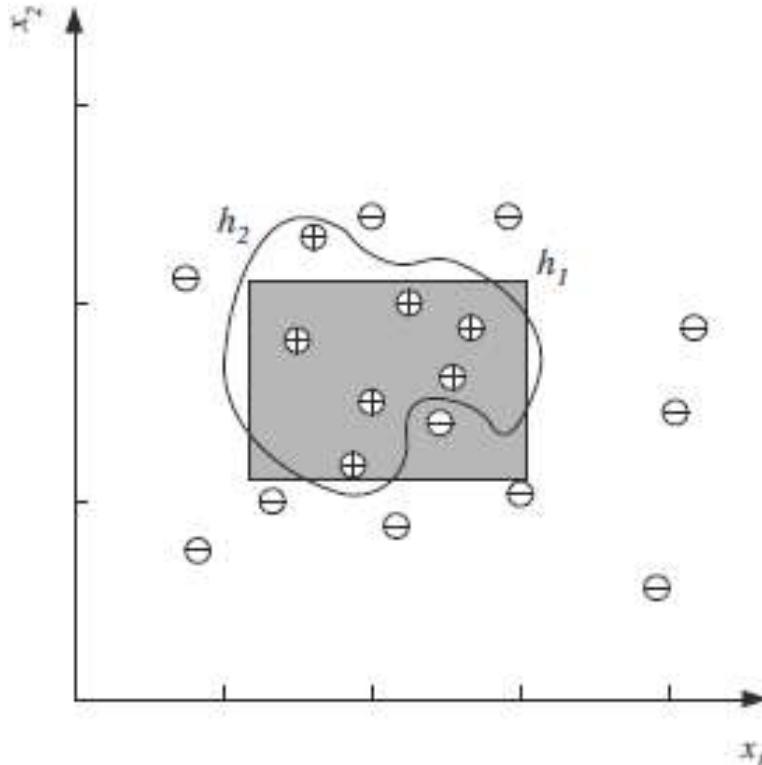
most specific hypothesis, S



most general hypothesis, G

- hypothesis class H is the set of all possible rectangles.
- $h \in H$, between S and G is **consistent**, valid hypothesis with no error and make up the **version space**
- Different training set, S , G , version space, the parameters different learned hypothesis, h

Noise



- Imprecision in recording the input attributes
- Errors in labeling the data points
- Additional hidden/latent attributes that affect the label of an instance



Model selection and assessment

Model selection and assessment



Model selection: estimating the performance of different models in order to choose the best one.

$h_0(x)$: If <Sunny, ?, High, ?, ?, ?> → Enjoy,
Else Not-Enjoy

Predicted → vs Actual		Enjoy	~Enjoy
Enjoy	2	1	
~Enjoy	0	1	

$h_1(x)$: If <?, ?, High, Strong, ?, ?> → Enjoy,
Else Not-Enjoy

Predicted → vs Actual		Enjoy	~Enjoy
Enjoy	2	1	
~Enjoy	1	0	

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Model selection and assessment

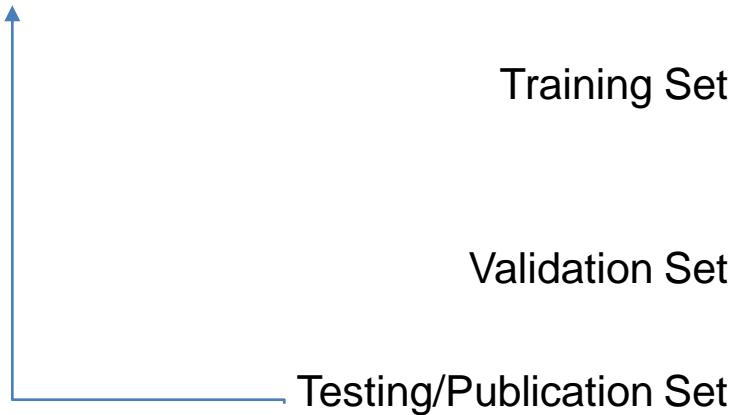


Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

$h_0(x)$: If <Sunny, ?, High, ?, ?, ?> \rightarrow Enjoy,
Else Not-Enjoy

Predicted \rightarrow vs Actual	Enjoy	\sim Enjoy
Enjoy	1	1
\sim Enjoy	0	0

Test / Generalization /Prediction Error



Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Training vs. Test Distribution

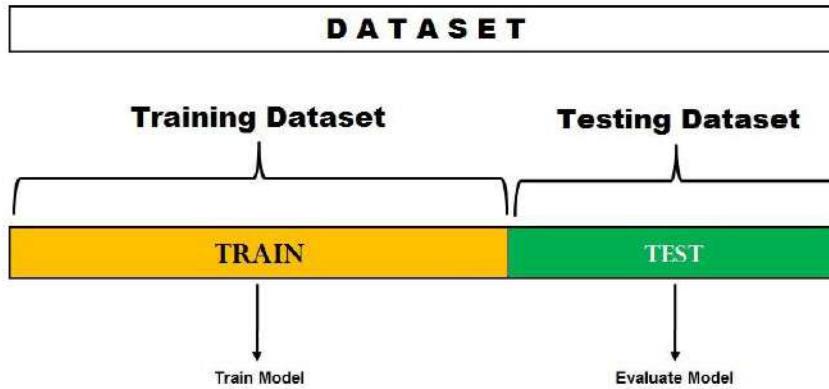


- Assumption : Training and Test instances are independently drawn from the same overall distribution of data
- Known as “i.i.d” (independent and identically distributed)

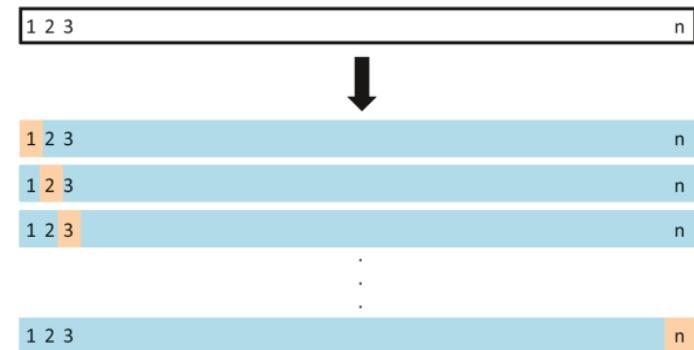
Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Cross-Validation: Resampling when there is few data

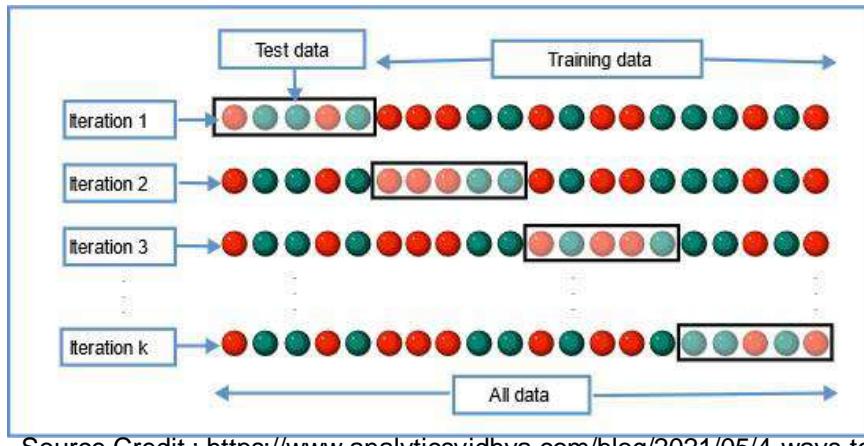
Hold Out method



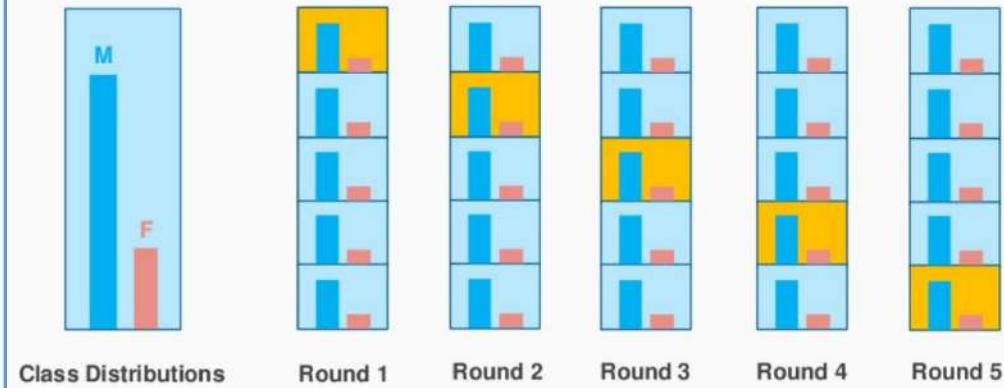
Leave One Out Cross-Validation



K-Fold Cross-Validation



Stratified K-Fold Cross-Validation



Source Credit : <https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/>

Loss vs Train error Vs Test error

- Target variable Y , a vector of inputs X, and prediction

model $f(x)$ -hat

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases}$$

Loss function:

Training error : Average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}(x_i))$$

X Mileage	Y Car Price	H0	Error L1
9.8	10.48	9	2
9.12	1.75	8	7
9.5	6.95	9	3
10	2.51	9	7
			Sum of Loss
			19
			Training Error
			4.75

Test error (generalization error) is the prediction error over an independent test sample, the training set T is fixed

$$\text{Err}_{\mathcal{T}} = \mathbb{E}[L(Y, \hat{f}(X))|\mathcal{T}]$$

Loss function Vs Evaluation metrics

- A loss function is used to **train** your model. A metric is used to **evaluate** your model
- A loss function is used **during** the learning process. A metric is used **after** the learning process



ML workflow

ML workflow

1. Should I use ML on this problem?
 - Is there a pattern to detect?
 - Can I solve it analytically?
 - Do I have data?
2. Gather and organize data.
3. Preprocessing, cleaning, visualizing.
4. Choosing a model, loss, regularization, ...
5. Optimization
6. Hyper parameter search.
7. Analyze performance and mistakes, and iterate back to step 5 (or 3)

Example: Car Price prediction based on Mileage

- Data: survey, Past Purchase data
- Process the data
 - **training set; test set**
 - representation of input features; output
- Choose form of model: **linear regression**
- System's performance evaluation: **objective function**
- optimize performance by setting appropriate parameters: **Optimization**
- Evaluate on test set: **generalization**

X Mileage	Y Car Price	H0	Error L1
9.8	10.48	9	2
9.12	1.75	8	7
9.5	6.95	9	3
10	2.51	9	7

References

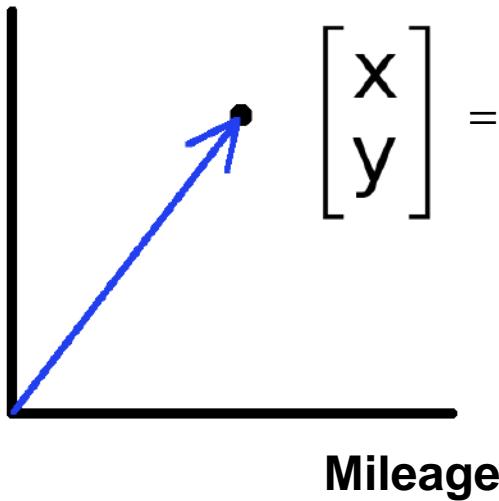
- Chapter 1 – Machine Learning, Tom Mitchell
- Chapter 1, 2 – Introduction to Machine Learning, 2nd edition, Ethem Alpaydin
- Chapter 1 - Pattern Recognition & Machine Learning Christopher M. Bishop

Linear algebra Review

Vectors

- Vectors can represent an offset in 2D or 3D space
- Points are just vectors from the origin

Car Price



- Data can also be treated as a vector

Mileage (in kmpl)	Size of engine cylinder (in cc)	Car Price (in cr)
9.8	6749	10.48
9.12	4395	1.75
9.5	6750	6.95
10	4691	2.51

Vector

- A column vector $v \in \mathbb{R}^{n \times 1}$ where

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Mileage= $\begin{bmatrix} 9.8 \\ 9.12 \\ 9.5 \\ 10 \end{bmatrix}$

- A row vector $v^T \in \mathbb{R}^{1 \times n}$ where

$$v^T = [v_1 \quad v_2 \quad \dots \quad v_n]$$

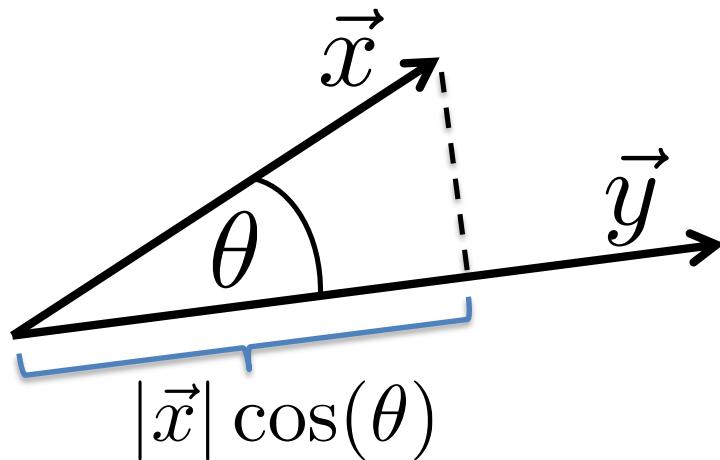
T denotes the transpose operation

$$\text{Mileage}^T = | 9.8 \quad 9.12 \quad 9.5 \quad 10 |$$

Mileage (in kmpl)	Size of engine cylinder (in cc)	Car Price (in cr)
9.8	6749	10.48
9.12	4395	1.75
9.5	6750	6.95
10	4691	2.51

Dot product geometric intuition:

“Overlap” of 2 vectors



$$\vec{x} \cdot \vec{y} = |\vec{x}| |\vec{y}| \cos(\theta)$$

Mileage (in kmpl)	Size of engine cylinder (in cc)	Car Price (in cr)
9.8	6749	10.48
9.12	4395	1.75
9.5	6750	6.95
10	4691	2.51

Slide credit: Mark Goldman and Emily Mackevicius

Multiplication: Dot product (inner product)



The dot product represents the similarity between vectors as a single number:

If two vectors are in the same direction the dot product is positive and if they are in the opposite direction the dot product is negative.

$$\vec{x} \cdot \vec{y} =$$

$$(x_1 \quad x_2 \quad \cdots \quad x_N) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = x_1 y_1 + x_2 y_2 + \cdots + x_N y_N$$

Outer dimensions give size of resulting matrix

1 X 1

Slide credit: Mark Goldman and Emily Mackevicius

Norm

- **Norm** is a function that assigns a strictly positive *length* or *size* to each vector in a vector space—except for the zero vector

- **L¹ norm** - *Manhattan/Taxicab Distance*, the *Mean Absolute Error (MAE)*, or the *Least Absolute Shrinkage and Selection Operator (LASSO)*

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|$$

- **L² norm** - *Euclidean Distance*, the *Mean Squared Error (MSE) / Least Squares Error*, or the *Ridge Operator*

$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

Norm

L^p norm - Let $p \geq 1$ be a real number. The p norm of vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

Matrix norm

$$\|A\| = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}.$$

Euclidean distance

u and v are n dim vectors

$$d(u, v) = \|u - v\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

$$d(\text{car1}, \text{car2}) = \sqrt{(9.8 - 9.12)^2 + (6.749 - 4.395)^2 + (10.48 - 1.75)^2} = 9.07$$

$$d(\text{car1}, \text{car3}) = \sqrt{(9.8 - 9.5)^2 + (6.749 - 6.750)^2 + (10.48 - 6.95)^2} = 3.54$$

Mileage (in kmpl)	Size of engine cylinder (in cc)	Car Price (in cr)
9.8	6749	10.48
9.12	4395	1.75
9.5	6750	6.95
10	4691	2.51

Required Reading for completed session :

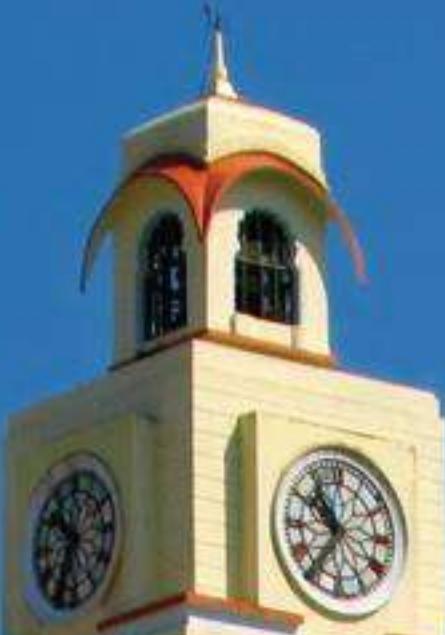
T1 - Chapter # 1 (Tom M. Mitchell, Machine Learning)

R1 – Chapter #2 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

& Refresh your MFDS previous semester course basics

Next Session Plan :

Refresher of Few more math preliminaries
Linear models for classification



Machine Learning

DSE CLZG565

M2 : Mathematical Preliminaries

M6 : Linear Models for Regression

Raja vadhana P

Assistant Professor,
BITS - CSIS

BITS Pilani
Pilani Campus



Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 6

- Linear Regression
- Geometric Approach
- Notion of Linear Basis Function Models
- Estimating the Parameters : Closed Form Solutions
- Estimating the Parameters : Gradient Descent Algorithm
- Regularization

Module – 2

Prerequisites : Refresher only

- Linear Algebra
- Calculus
- Probability Theory

Learnt so far in previous session

- ML Workflow
- Formal Representation of ML Models
- Linear Algebra
 - Notion of Vector Representations
 - Notion of Similarity vs Dissimilarity
- Few Terminologies /Concepts
 - Hypothesis Space , Version Space
 - Target Concept (Prediction Types : Regression vs Classification vs Probability Estimation)
 - Trade-off Specialization vs Generalization
 - Loss/Error
 - Training vs Testing Vs Validation
 - Evaluation Metrics
 - Overfitting vs Underfitting

Agenda

- Linear Models for Regression
- Geometric Interpretation
- Closed Form vs Gradient Descent
- Math Preliminaries – Refresher (Wherever applicable will be refreshed in sync with discussion)

Linear Regression

Inductive Learning Hypothesis : Interpretation

- Target Concept
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

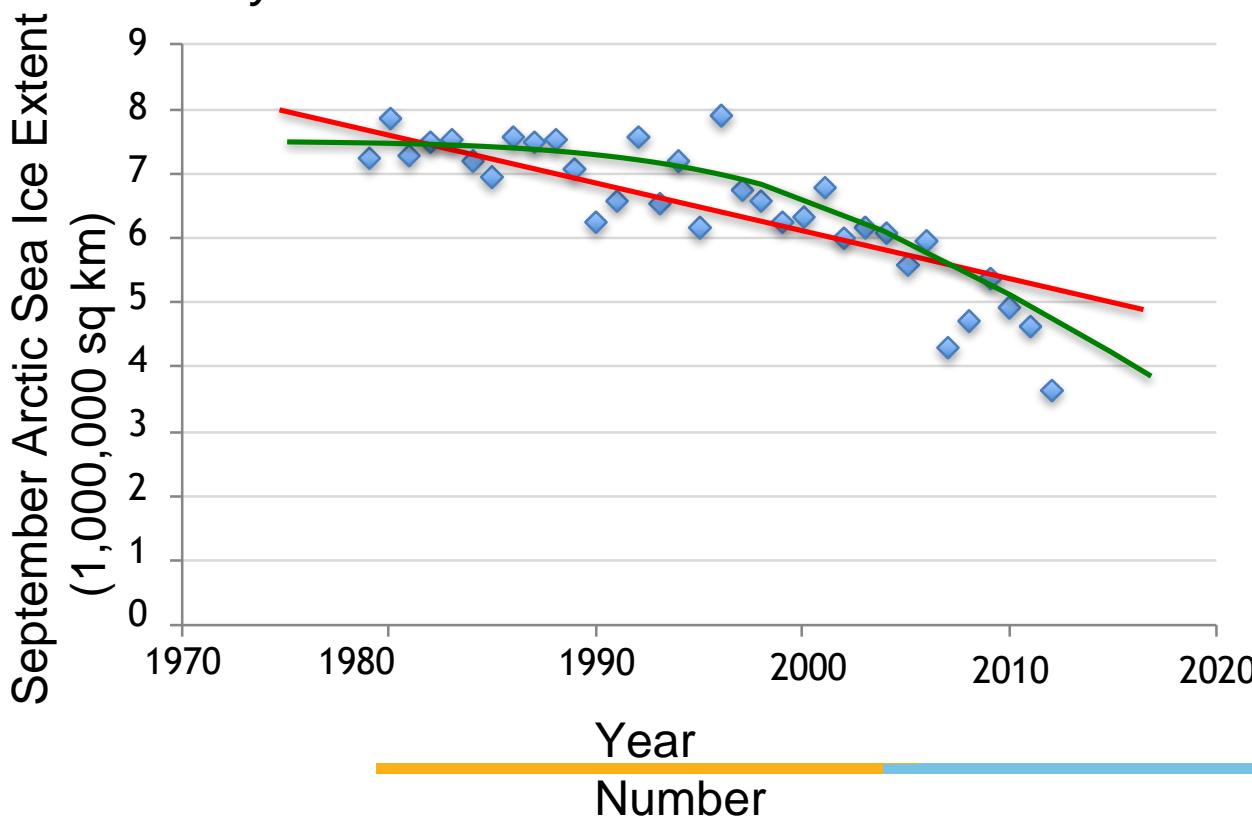
Formal Representation: Interpretation



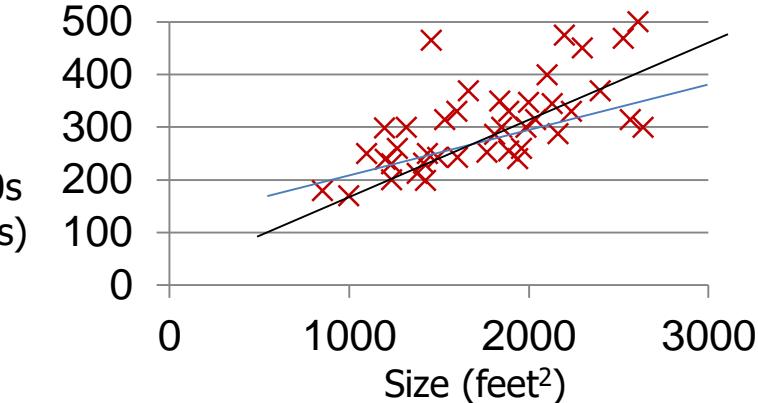
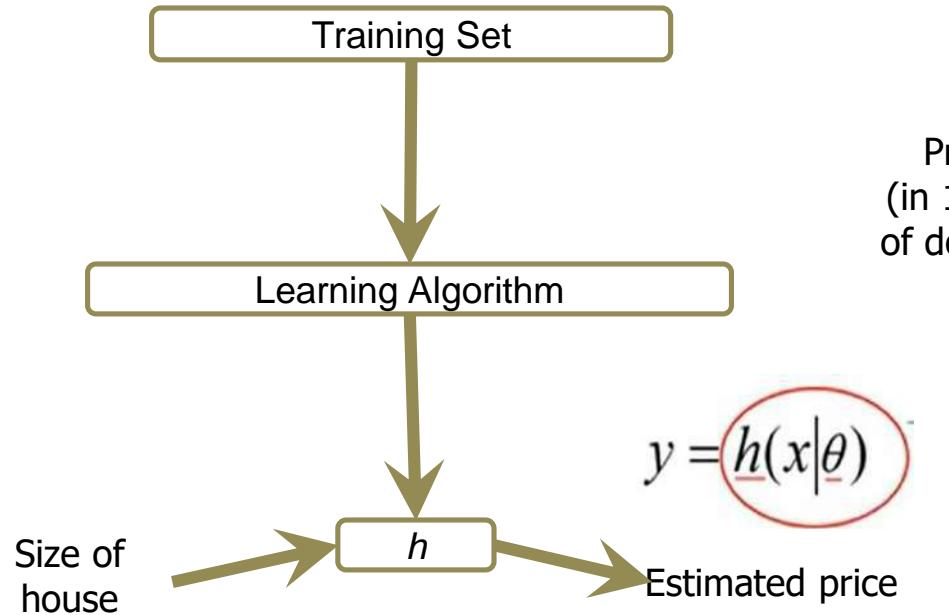
Supervised Learning: Regression

GOAL : Previously unseen records should be assigned a value as accurately as possible.

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is real-valued



Machine Learning Process : Interpretation



Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Hypothesis: $h_{\theta}(x) = \theta_0 + \theta_1 x$

θ_i 's : Parameters
 $h(\cdot)$: Model

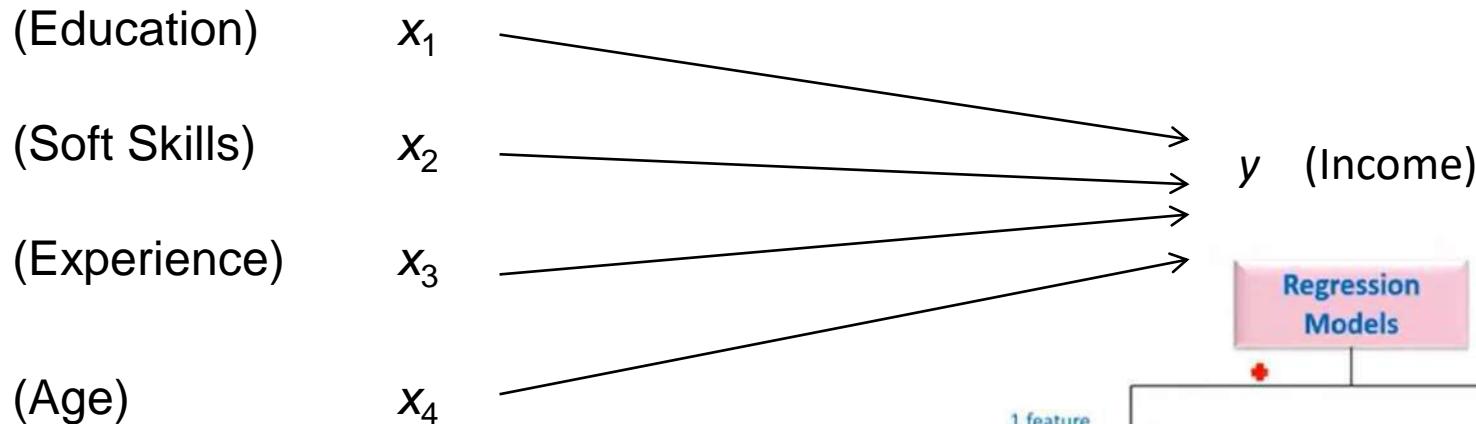
How to choose θ_i 's ?

Types of Regression Models

simple regression model

(Education) $x \longrightarrow y$ (Income)

multiple regression model

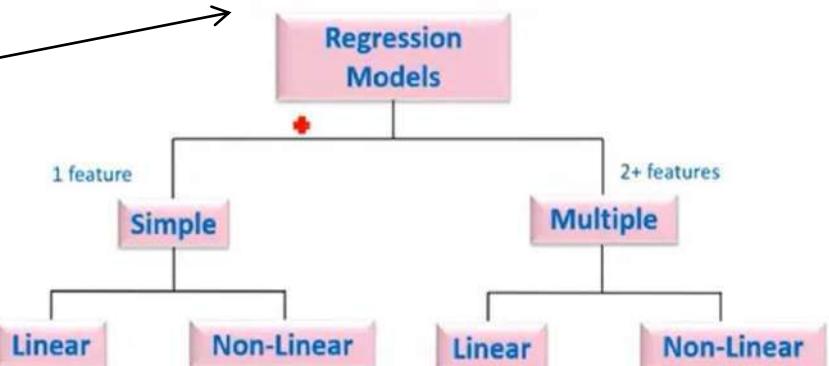


$$h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x,$$

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Assume $x_0 = 1$



Learning the Model Parameters

Closed Form Solution Approach

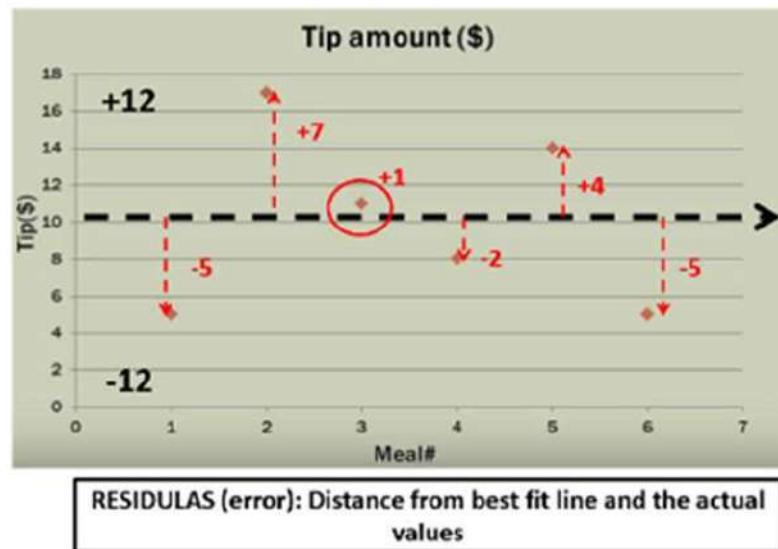
Gradient Descent Approach

How to choose θ_i 's ?

Notion of Cost Function

Meal #	Tip amount (\$)
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00

mean = \$10



Meal#	Residual	Residual ²
1	-5	25
2	+7	49
3	+1	1
4	-2	4
5	+4	16
6	-5	25

Sum of Squared Errors (SSE) = 120

$$y = \theta_0$$

$$J(\Theta) = \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\Theta) = \sum_{i=1}^n (h_{\Theta}(x^{(i)}) - y^{(i)})^2$$

Notion of Cost Function

Meal #	Tip amount
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00
mean = \$10	

Meal	Total bill (\$)	Tip amount (\$)	$\hat{y}_i = 0.1462x - 0.8188$	\hat{y}_i (predicted tip amount)
1	34	5	$\hat{y}_i = 0.1462(34) - 0.8188$	4.1505
2	108	17	$\hat{y}_i = 0.1462(108) - 0.8188$	14.9693
3	64	11	$\hat{y}_i = 0.1462(64) - 0.8188$	8.5365
4	88	8	$\hat{y}_i = 0.1462(88) - 0.8188$	12.0453
5	99	14	$\hat{y}_i = 0.1462(99) - 0.8188$	13.6535
6	51	5	$\hat{y}_i = 0.1462(51) - 0.8188$	6.6359
			Observed vs. Predicted	
			$\bar{x} = 74$	$\bar{y} = 10$

$$y = \theta_0 + \theta_1 x$$

$$J(\Theta) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})^2$$

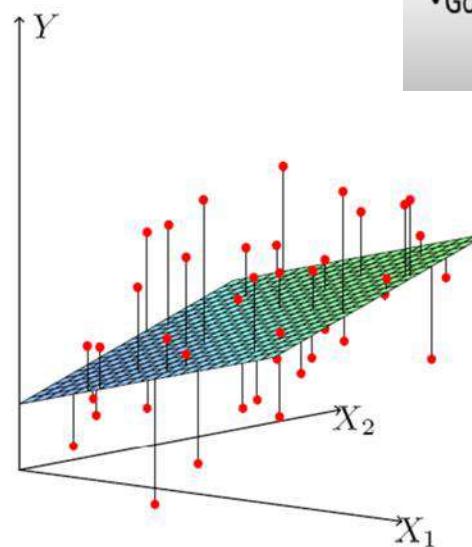
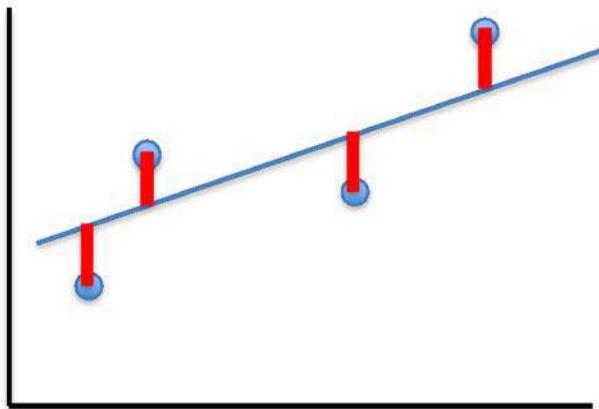
$$J(\Theta) = \frac{1}{2n} \sum_{i=1}^n (h_\Theta(x^{(i)}) - y^{(i)})^2$$

Linear Regression : Least Squares

- Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta} \left(x^{(i)} \right) - y^{(i)} \right)^2$$

- Fit by solving $\min_{\theta} J(\theta)$



- Hypothesis

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

- Parameters

$$\theta_0 \text{ and } \theta_1$$

- Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^i) - y^{(i)})^2$$

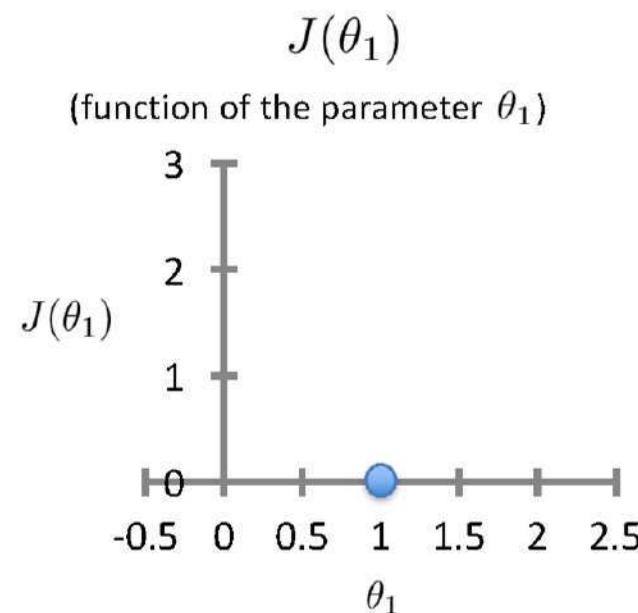
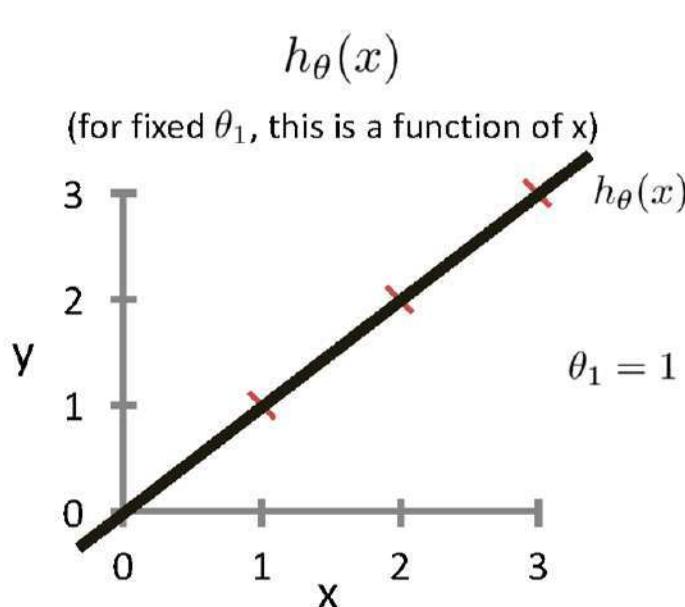
- Goal

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

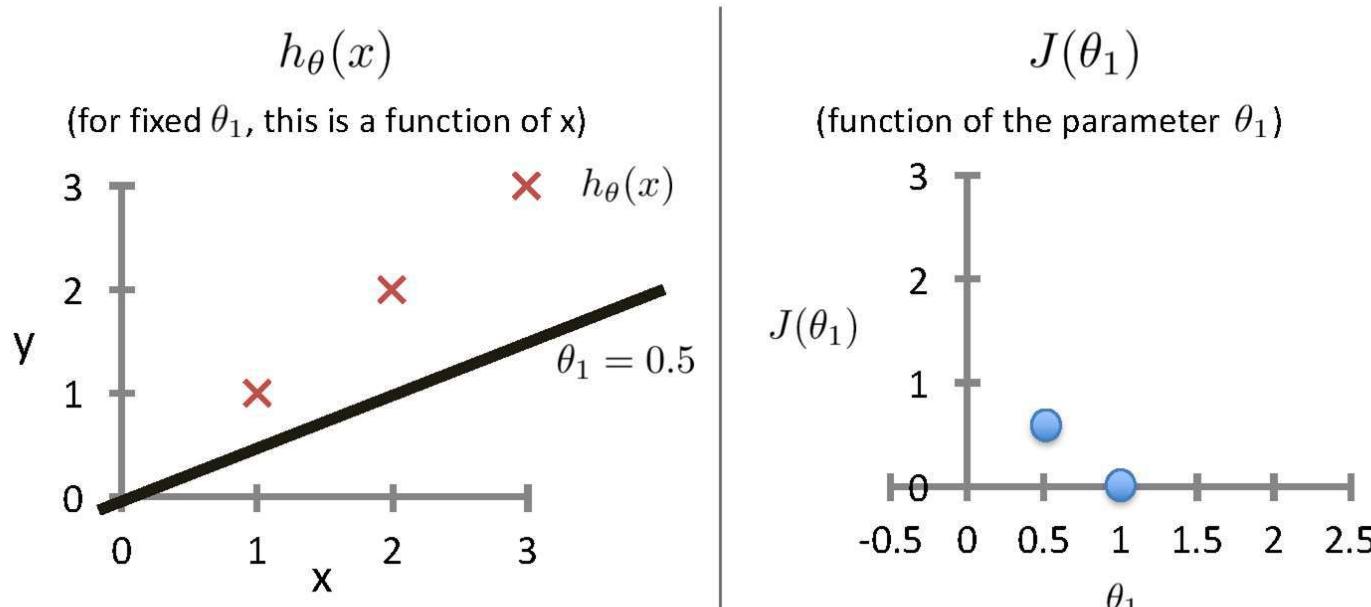


Source Credit : Based on example by Andrew Ng

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$



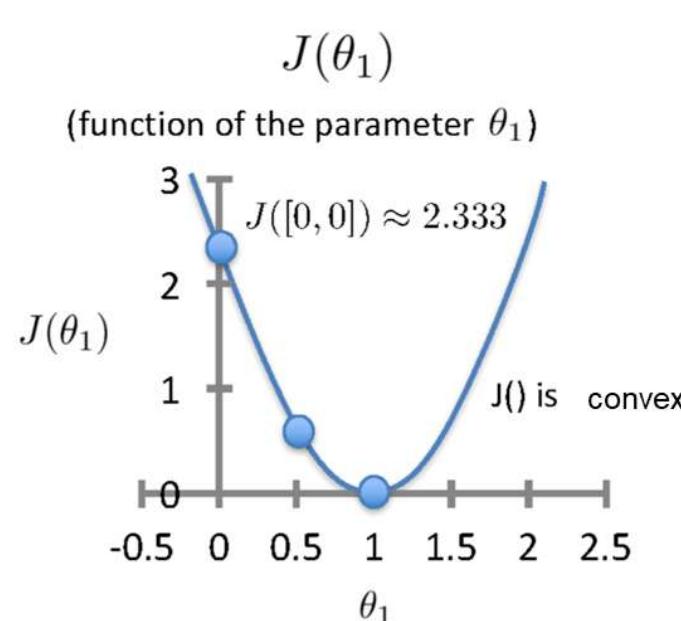
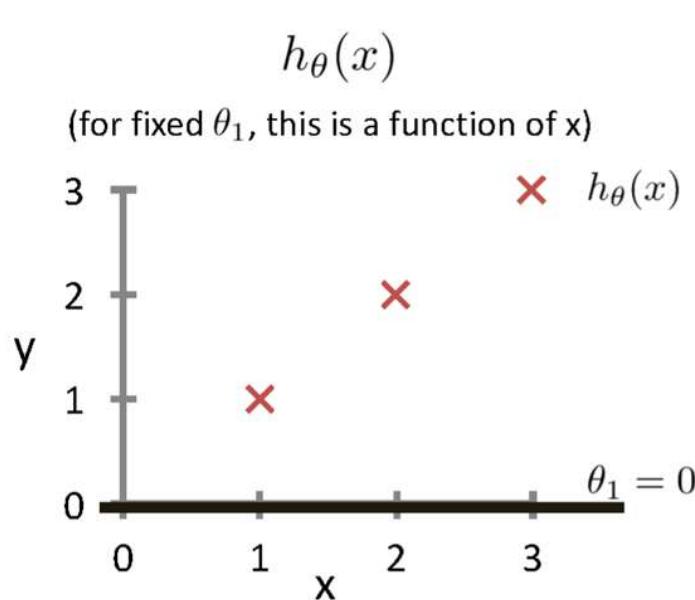
Based on example
by Andrew Ng

$$J([0, 0.5]) = \frac{1}{2 \times 3} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] \approx 0.58$$

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$



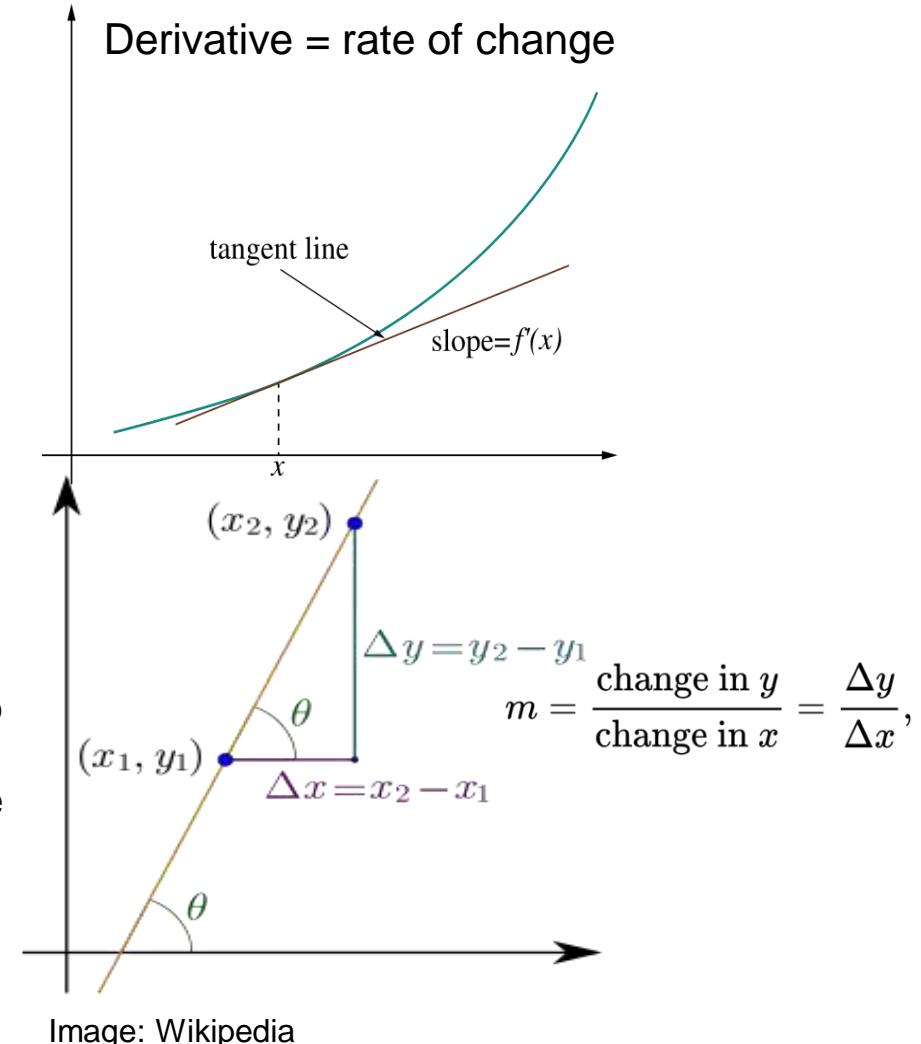
Source Credit : Based on example
by Andrew Ng



Calculus review

Differentiation

- The derivative provides us information about the rate of change of a function.
- The derivative of a function is also a function.
Example:
 - The derivative of the acceleration function is the velocity function.
 - Linear function $y = mx + b$
 - Slope



Source Credit : Texas A&M Dept of Statistics

Ways to Write the Derivative

Given the function $f(x)$, we can write its derivative in the following ways:

- $f'(x)$

The derivative of x is commonly written dx .

- $\frac{d}{dx} f(x)$

More Formulas

- The derivative of u to a constant power:

$$\frac{d}{du} u^n = n * u^{n-1}$$

- The derivative of e :

$$\frac{d}{du} e^u = e^u$$

- The derivative of \log :

$$\frac{d}{du} \log(u) = -\frac{1}{u}$$

Source Credit : Texas A&M Dept of Statistics

Chain Rule

The chain rule allows you to combine any of the differentiation rules we have already covered.

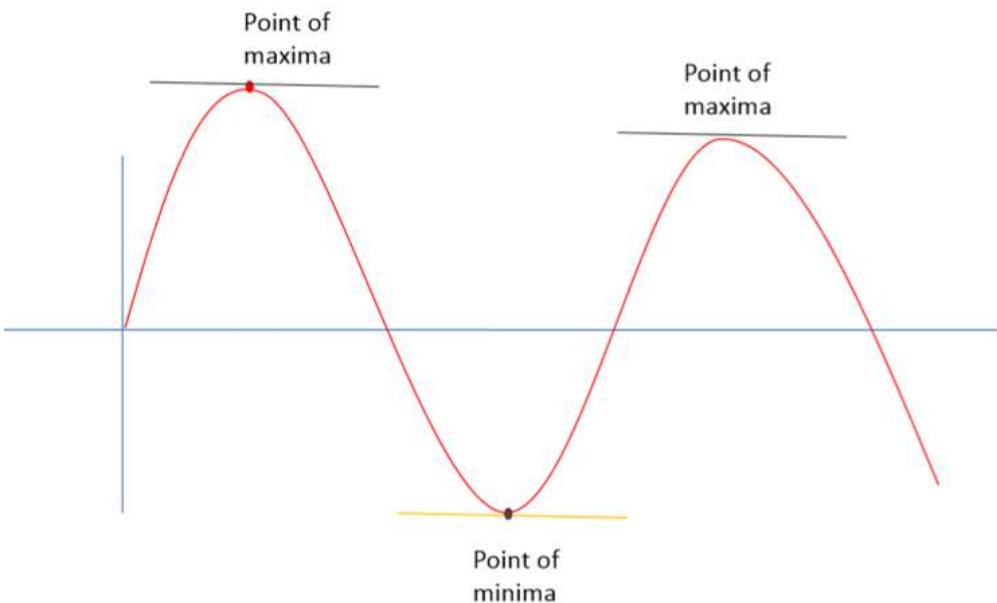
- First, do the derivative of the outside and then do the derivative of the inside.

$$\frac{d}{du} f(g(u)) = f'(g(u)) * g'(u) * du$$

$$\frac{d}{du} (2u+u^m)^n = n * (2u+u^m)^{n-1} * (2 + (m * u^{m-1}))$$

Maxima and Minima

- For maxima and minima $m = dy/dx = \tan \theta = 0$
- $dy/dx = 0$ means tangent is parallel to X –axis.



Notion of Maxima and Minima of a function

In Machine Learning

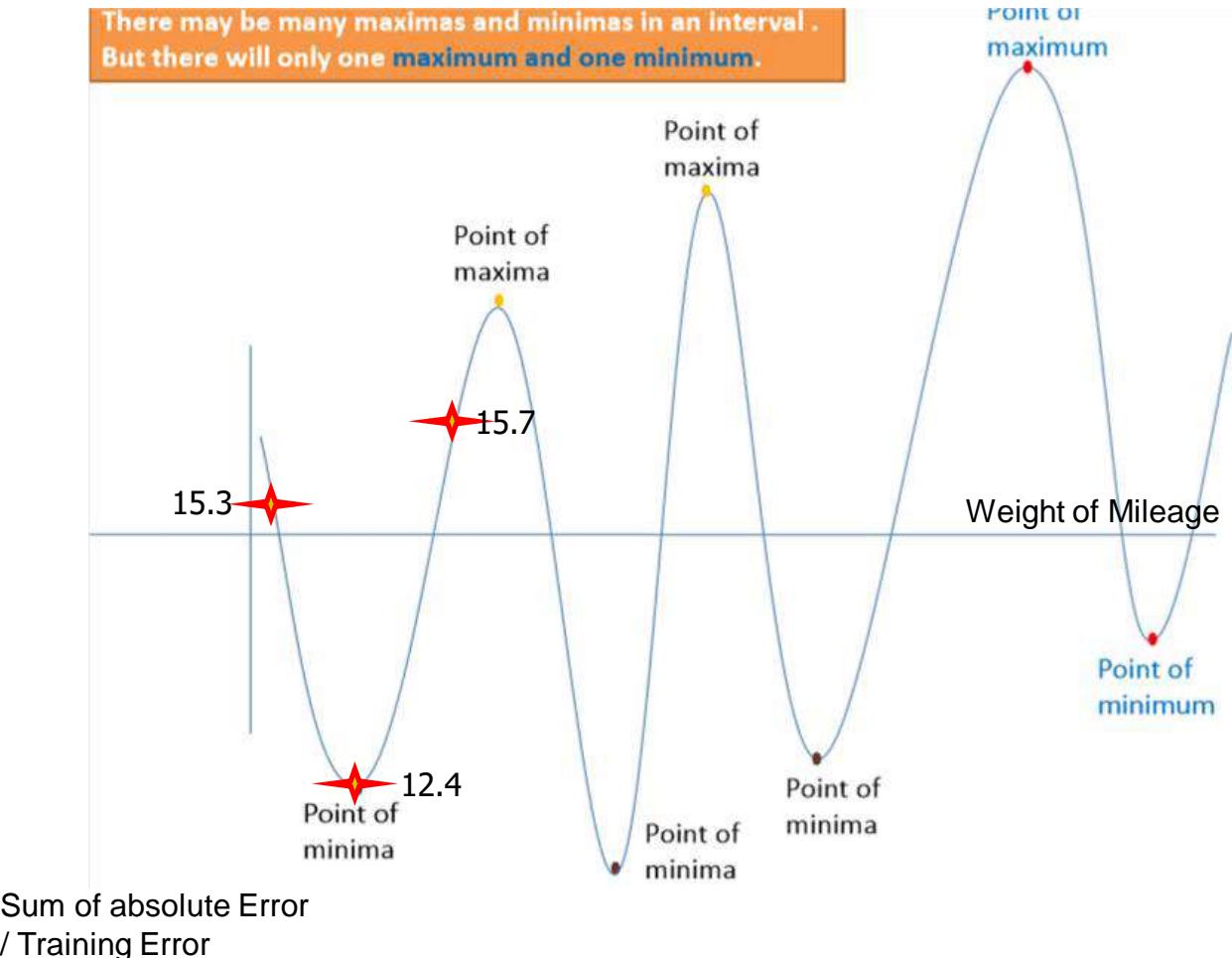
CarPrice = 1 + 0.05Mileage

CarPrice = 1 + 0.25Mileage

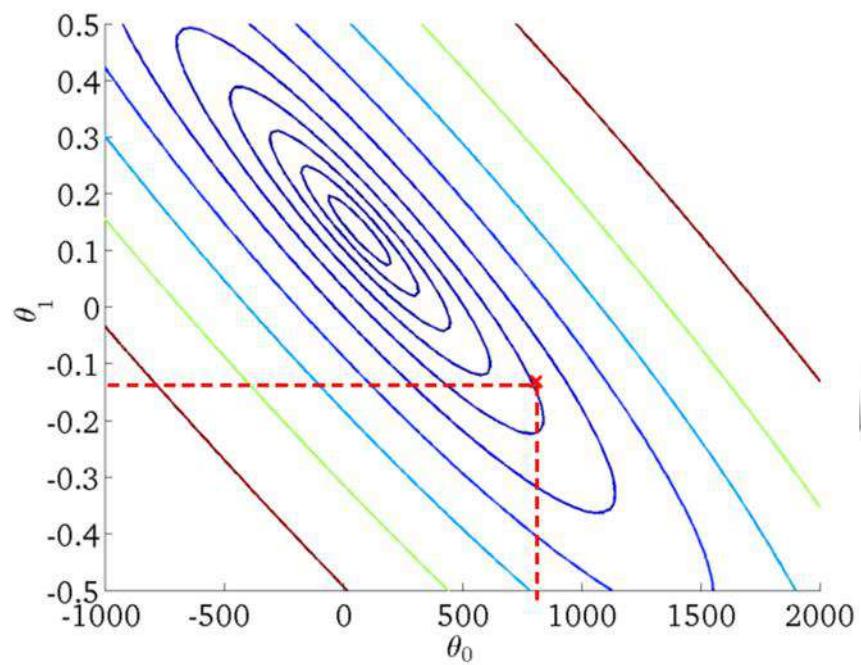
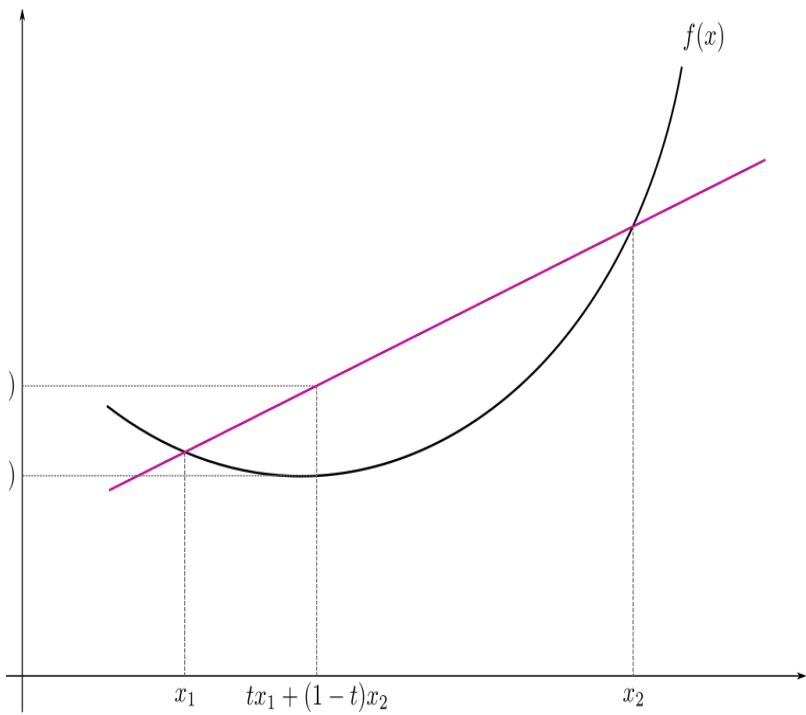
CarPrice = 1 + 0.75Mileage

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

There may be many maximas and minimas in an interval.
But there will only one maximum and one minimum.



Convex Function : Multivariate



Real-valued function defined on an n -dimensional interval is called **convex** if the line segment between any two points on the graph of the function lies above or on the graph



Closed Form Solution

Vectorization

- Benefits of vectorization

- More compact equations
- Faster code (using optimized matrix libraries)

- Consider our model:

$$h(\mathbf{x}) = \sum_{j=0}^d \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{x}^\top = [1 \quad x_1 \quad \dots \quad x_d]$$

- Can write the model in vectorized form as $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \mathbf{x}^\top = [1 \quad \text{No.of.Years Experience}]$$

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

Vectorization

- Consider our model for n instances:

$$h(\mathbf{x}^{(i)}) = \sum_{j=0}^d \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

$\mathbb{R}^{(d+1) \times 1}$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

$\mathbb{R}^{n \times (d+1)}$

Let:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{X}\boldsymbol{\theta}$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

Vectorization

- For the linear regression cost function:

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2n} \sum_{i=1}^n \left(\theta^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{2n} (\mathbf{X}\theta - \mathbf{y})^T (\mathbf{X}\theta - \mathbf{y})$$

$\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$
 $\mathbf{y} \in \mathbb{R}^{(d+1) \times 1}$
 $\mathbf{X} \in \mathbb{R}^{1 \times n}$
 $\mathbf{y} \in \mathbb{R}^{n \times 1}$

Let:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

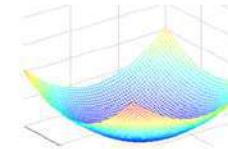
$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

Vectorization

- Instead of using GD, solve for optimal θ analytically

– Notice that the solution is when $\frac{\partial}{\partial \theta} J(\theta) = 0$



- Derivation:

$$\begin{aligned} J(\theta) &= \frac{1}{2n} (\mathbf{X}\theta - \mathbf{y})^\top (\mathbf{X}\theta - \mathbf{y}) \\ &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - \boxed{\mathbf{y}^\top \mathbf{X} \theta} - \boxed{\theta^\top \mathbf{X}^\top \mathbf{y}} + \mathbf{y}^\top \mathbf{y} \\ &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \end{aligned}$$

Take derivative and set equal to 0, then solve for θ :

$$\frac{\partial}{\partial \theta} (\theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \cancel{\mathbf{y}^\top \mathbf{y}}) = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta - \mathbf{X}^\top \mathbf{y} = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta = \mathbf{X}^\top \mathbf{y}$$

Closed Form Solution:

$$\theta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Vectorization

- Can obtain θ by simply plugging X and y into

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

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $X^T X$ is not invertible (i.e., singular), may need to:
 - Use pseudo-inverse instead of the inverse
 - In python, `numpy.linalg.pinv(a)`
 - Remove redundant (not linearly independent) features
 - Remove extra features to ensure that $d \leq n$

Fit the Linear Regression Model :

Using Closed Form

$$\left(X^T \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} * X \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \right) - 1 * X^T \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\left(\begin{bmatrix} 5 & 15 \\ 15 & 55 \end{bmatrix} \right) - 1 * X^T \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\left(\begin{bmatrix} 1.1 & -0.3 \\ -0.3 & 0.1 \end{bmatrix} \right) * X^T \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 0.8 & 0.5 & 0.2 & -0.1 & -0.4 \\ -0.2 & -0.1 & 0 & 0.1 & -0.2 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix}$$

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

Can obtain θ by simply plugging X and y into

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

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

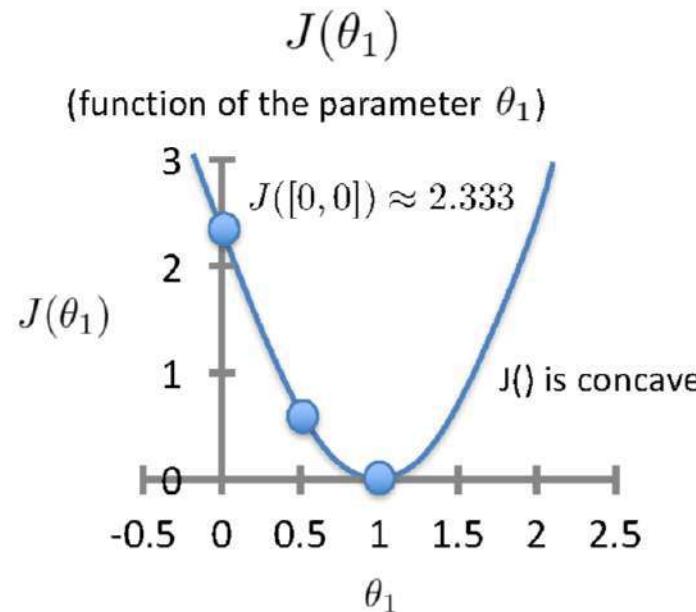
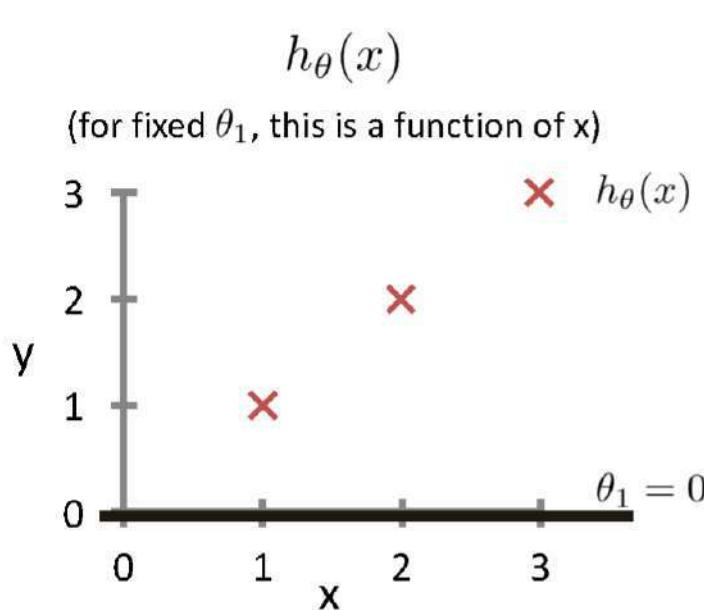


Gradient Descent Approach

Intuition Behind Cost Function

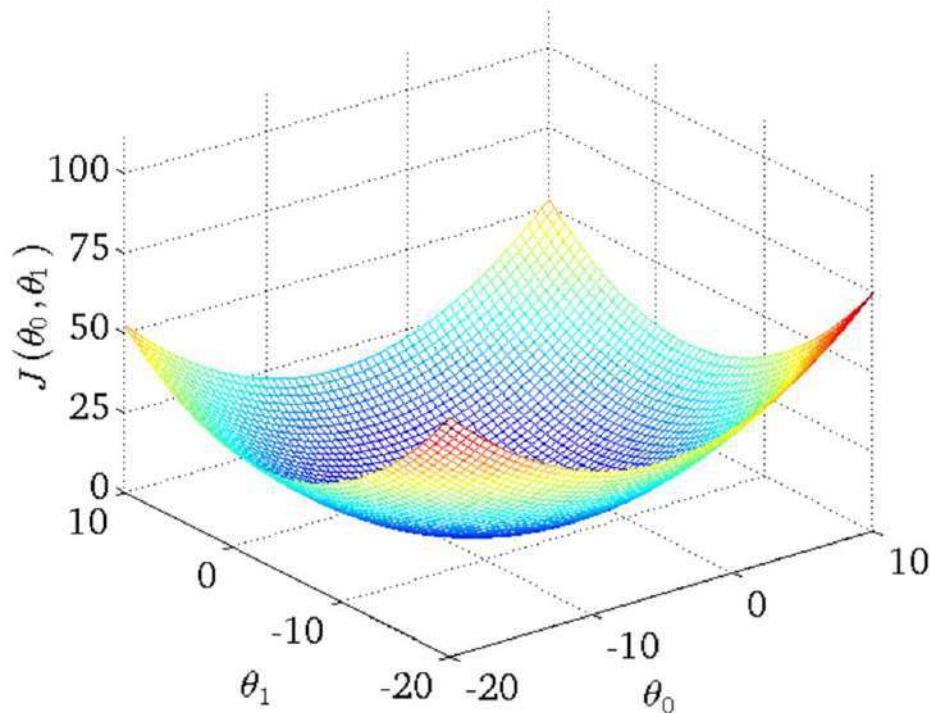
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$



Source Credit : Based on example by Andrew Ng

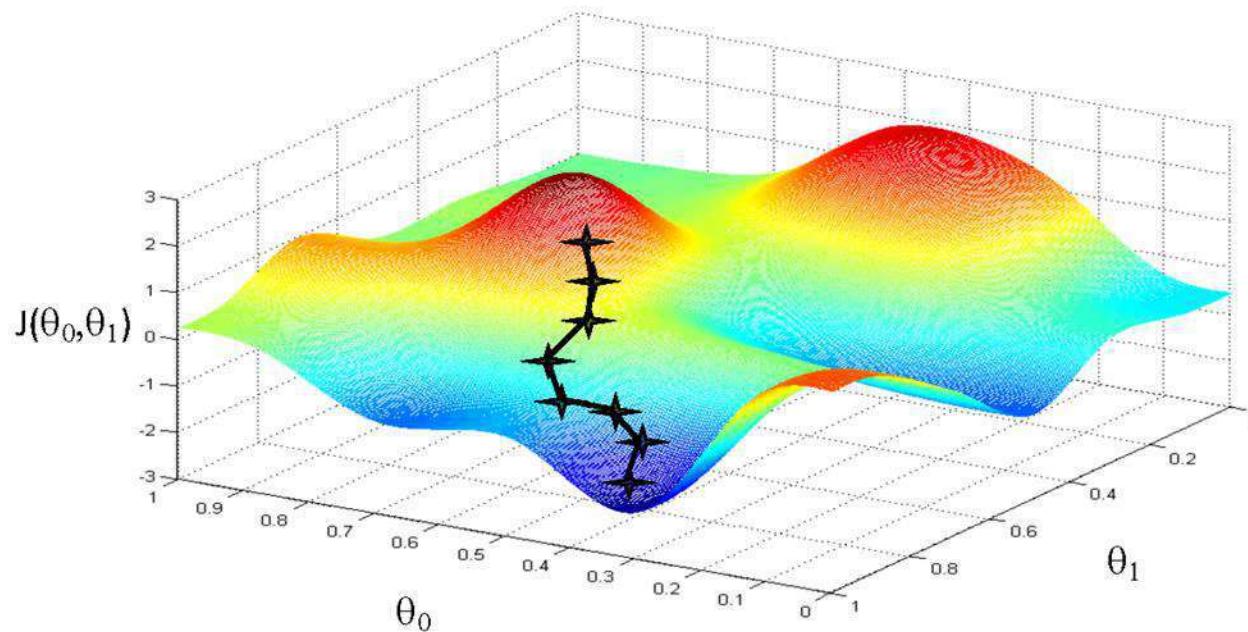
Intuition Behind Cost Function



Source Credit : Based on example
by Andrew Ng

Basic Search Procedure

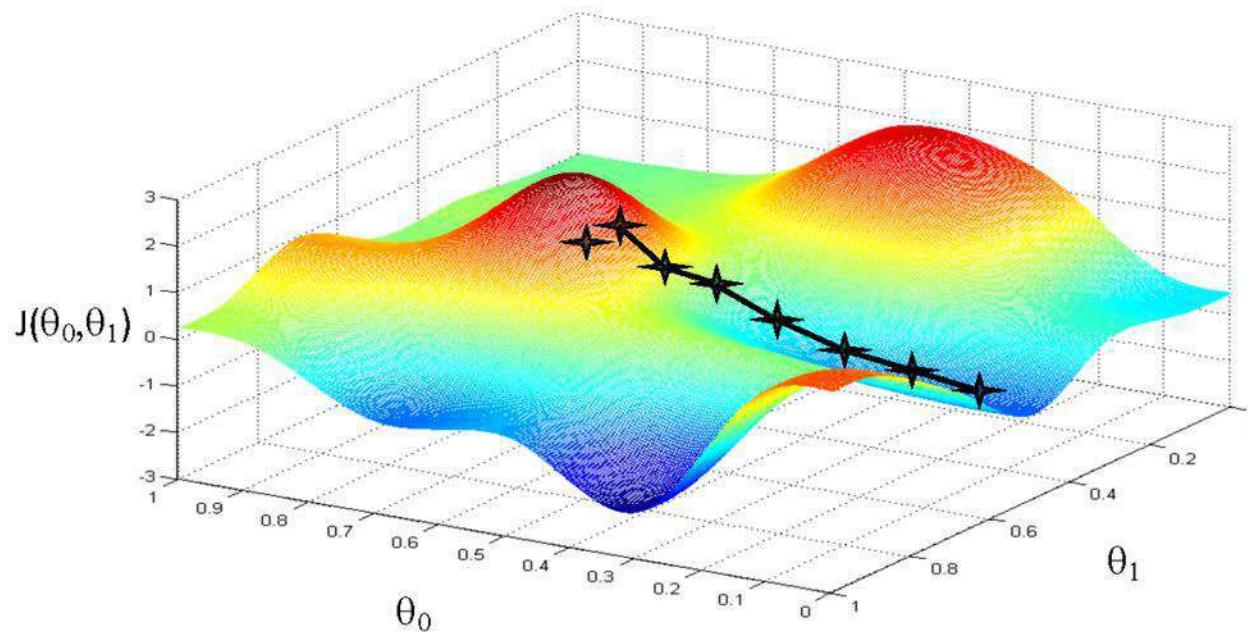
- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

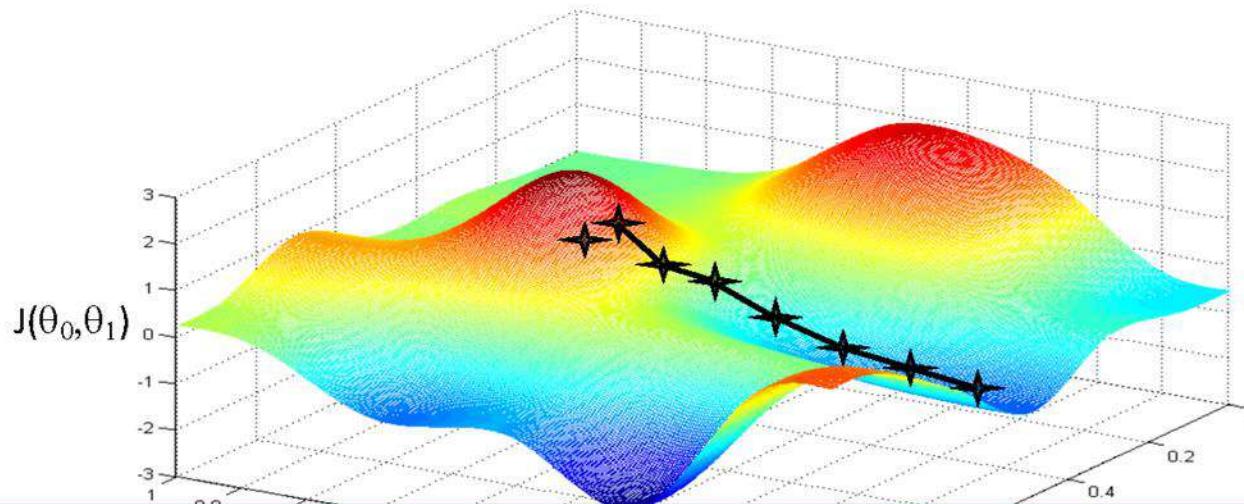
- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex (concave),
we don't need to worry about local minima

Source Credit : Figure by Andrew Ng

Gradient Descent

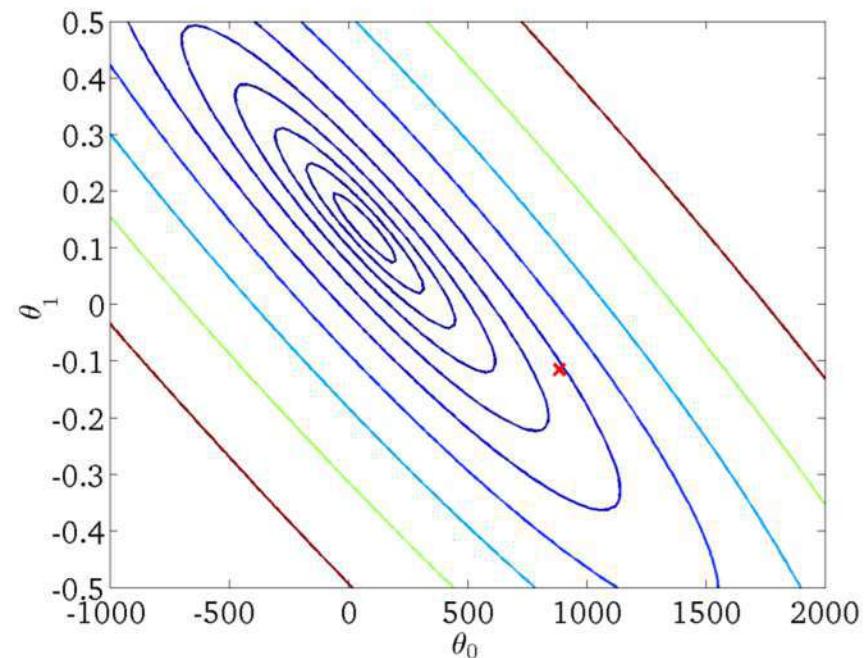
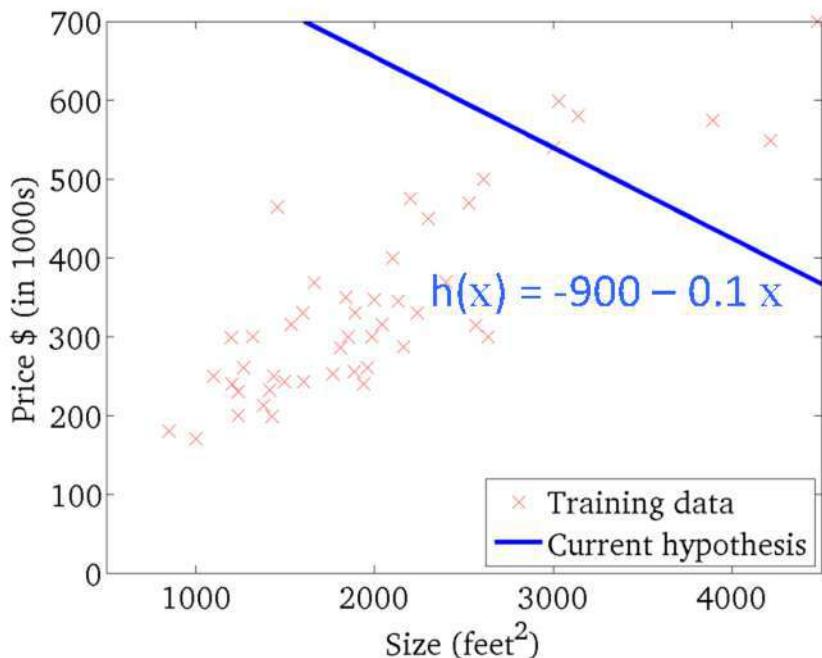
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)
...
852

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178

$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)
...

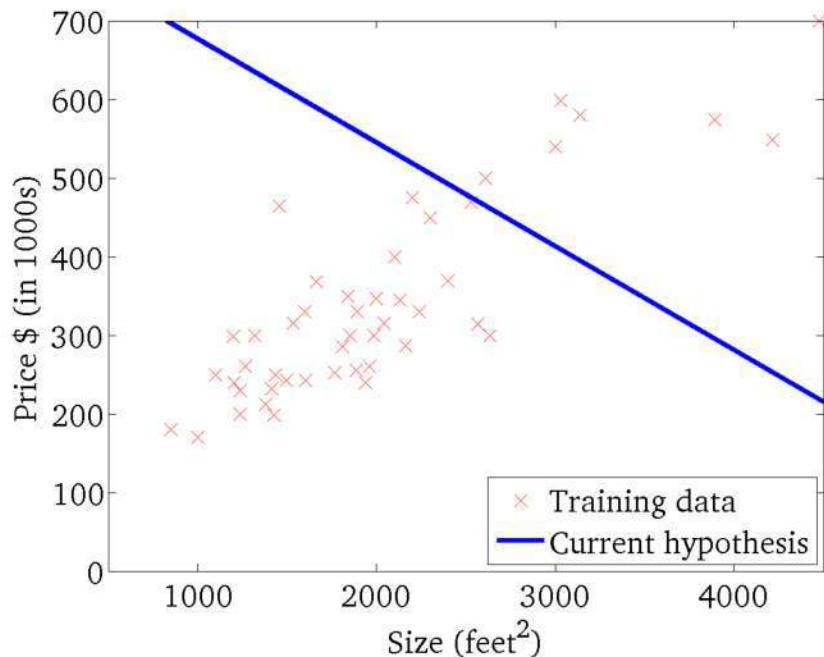


Source Credit : Slide by Andrew Ng

Gradient Descent

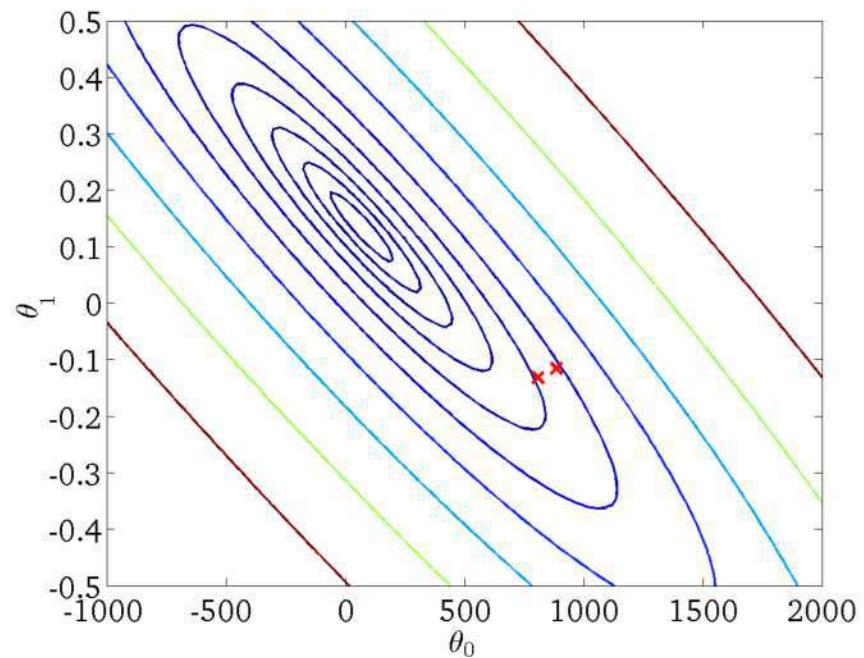
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

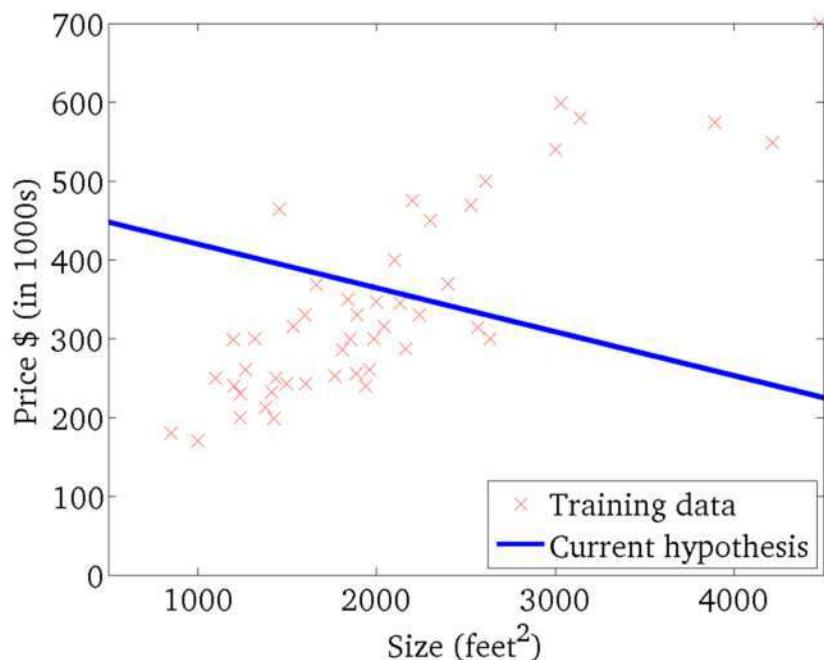


Source Credit : Slide by Andrew Ng

Gradient Descent

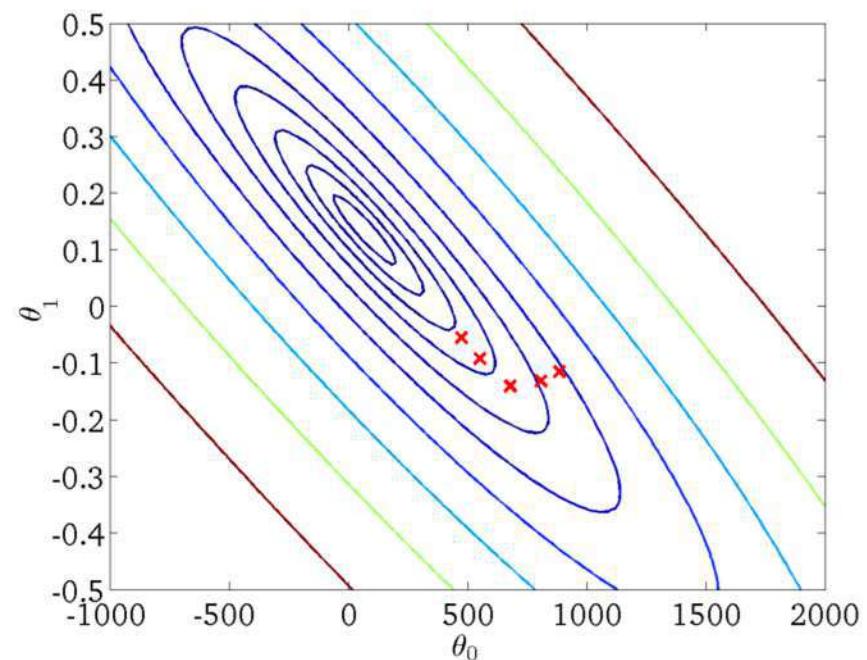
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

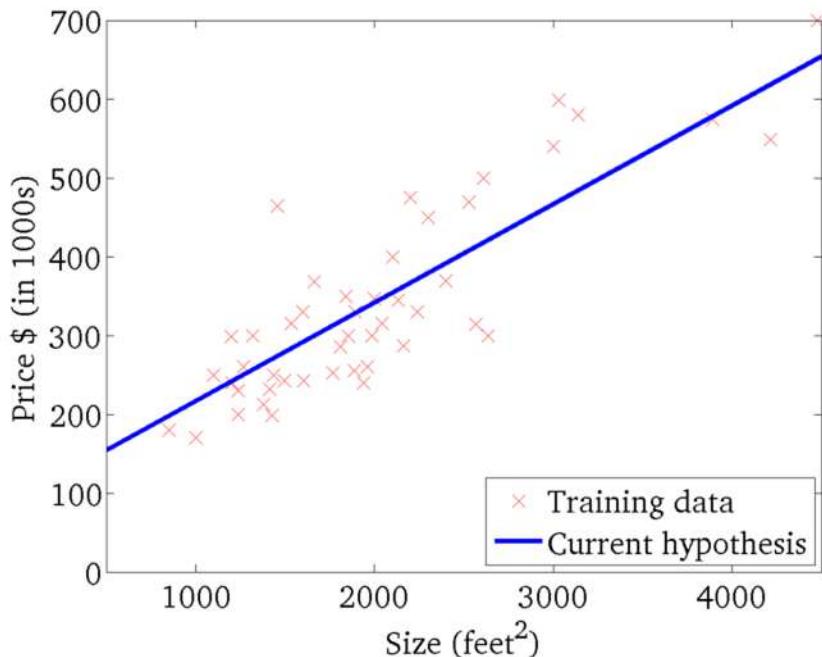


Source Credit : Slide by Andrew Ng

Gradient Descent

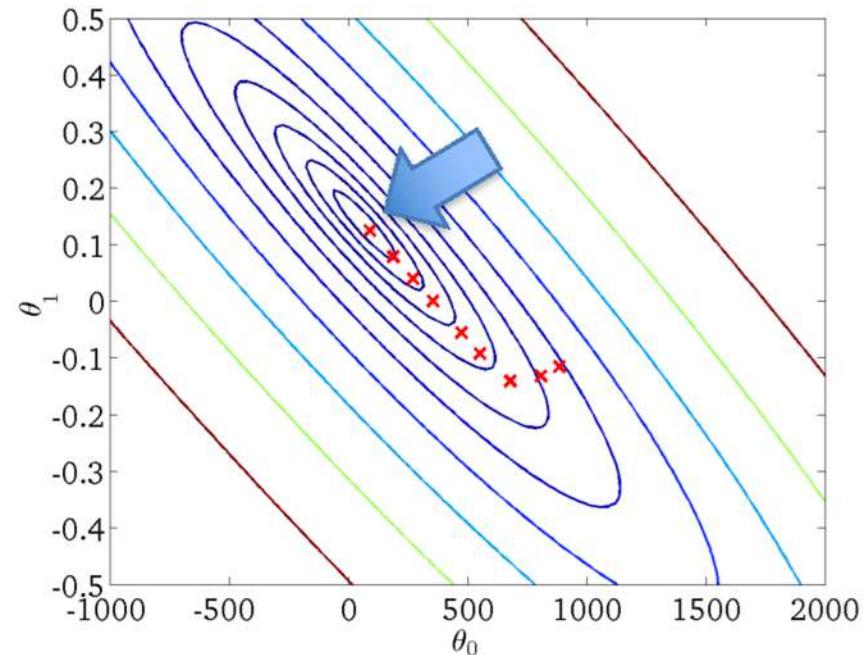
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



Source Credit : Slide by Andrew Ng

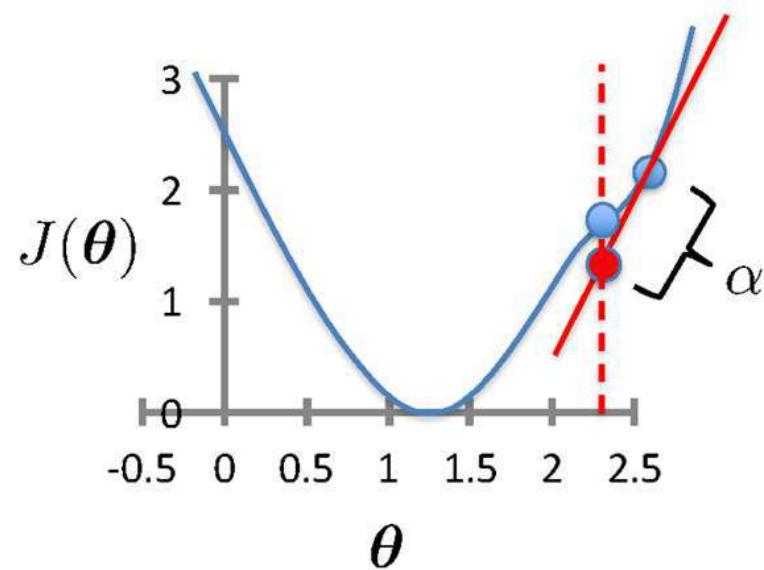
Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \text{simultaneous update for } j = 0 \dots d$$

For Linear Regression:

$$\begin{aligned}
 \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\
 &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\
 &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \\
 &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)} \\
 &\quad \ddots
 \end{aligned}$$

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

simultaneous update for $j = 0 \dots d$

- To achieve simultaneous update
 - At the start of each GD iteration, compute $h_{\theta}(\mathbf{x}^{(i)})$
 - Use this stored value in the update step loop
- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

$$\text{L}_2 \text{ norm: } \|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$$

Fit a linear Regression Line :

Gradient Descent

Steps :

(Assuming : 'n' no.of.instances and two predictors { x_1, x_2 } and linear regression)

1. Identification of the equations $y = w_0 + w_1X_1 + W_2X_2$
2. Cost function & derivative
 1. $W_0` = w_0 - 1/n * \text{learning rate} * (\sum (w_0 + w_1X_1 + W_2X_2 - y))$
 2. $W_1` = w_1 - 1/n * \text{learning rate} * (\sum (w_0 + w_1X_1 + W_2X_2 - y) * x_1)$
 3. $W_2` = w_2 - 1/n * \text{learning rate} * (\sum (w_0 + w_1X_1 + W_2X_2 - y) * x_2)$
3. Apply the equations

Fit a linear Regression Line :



Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

Steps :

1. Identification of the equations $y = w_0 + w_1X_1 + w_2X_2$
2. Cost function & derivative
 1. $w_0' = w_0 - \frac{1}{3} * \text{learning rate} * (\sum (w_0 + w_1X_1 + w_2X_2 - y))$
 2. $w_1' = w_1 - \frac{1}{3} * \text{learning rate} * (\sum (w_0 + w_1X_1 + w_2X_2 - y) * x_1)$
 3. $w_2' = w_2 - \frac{1}{3} * \text{learning rate} * (\sum (w_0 + w_1X_1 + w_2X_2 - y) * x_2)$
3. Apply the equations

Fit a linear Regression Line :

Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

Steps :

1. Identification of the equations: $RR-CHD = 5 - 0.03 * BMI - 0.03 * DiastolicPressure$
2. Cost function & derivative
 1. $W0` = w0 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD))$
 $= 5 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD))$
 2. $W1` = w1 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD) * BMI)$
 $= -0.03 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD) * BMI)$
 3. $W2` = w2 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD) * DiastolicPressure)$
 $= -0.03 - 1/3 * 0.02 * (\text{sum } (5-0.03BMI-0.03DiastolicPressure - RRCHD) * DiastolicPressure)$
3. Apply the equations : Answer at the end of first iteration:
 $W0 = 5.0016$, $W1 = 0.0476$, $w2= 0.179$

Exercise - 1 – Apply Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

A sample one iteration of the above problem was demo'd in class with steps and added in previous slide .

Now do another iteration using the results of the first iteration.

NOTE: All the updates are SIMULTANEOUS ie., all the three Wi's needs to be updated in the single go . i.e., Do not compute the equation of W0 and use it to compute W1 in the same iteration of the algorithm

Exercise - 2 – Find Closed Form Solution

For the same problem , if all the predictors are important features to predict the response variable RR-C HD, use the matrix based closed form solution and find the answer.
 Assume linear model of regression.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

Note:

Refresh the method from MFDS course and after the step $\mathbf{X}^T \mathbf{X}$ check if the resultant square matrix is invertible or not before proceeding with the calculations.

Exercise - 3 – Apply Gradient Descent

Fit a simple linear regression. Show only the first two iterations of Gradient descent algorithm using learning rate of **0.5** for the following. Assume the intercept of the regression model as **0.25** and the slope of independent variables as **-0.5 (negative)**.

x No.of.Years of Experience (in Years)	y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

This problem was discussed in the closed form solution. Now for the same problem and given above hyper-parameters perform 2 iterations of Gradient descent algorithm.

Also derive the equations by following and show the steps:

1. Represent the equation of the hypothesis
2. Formulate update equation for both intercept and the slope of X ie., two equations for Theta 0 , Theta 1
3. Differentiate the cost functions in both equations and show the steps explicitly
4. Plug in the result back to the update equation ‘

Exercise - 4 – Python Demo

1. Go to <https://bitscsis.vlabs.platifi.com/index1.html>
2. Download your virtual lab exercise under Machine Learning → LapCapsule2
Linear_Polynomial Regression → 4 Linear Regression using Gradient Descent
3. Modify the below hyper parameters and infer your observations
 - a) $\theta_1 = 0$ Change to { 5000 , -50 , 0.5 }
 - b) $\theta_0 = 0$ Change to { 5000 , 500 , -0.5 }
 - c) $\alpha = 0.0001$ Change to {1 , 0.2 , 5} for each of the combination
 - d) $\text{epochs} = 10000$ Change to {500 , 1000000} for each of the combination

Interpret on below in your own understanding:

- Influence of initial values of theta on the convergence
- Influence of larger alpha(learning rate) vs smaller alpha
- Influence of no.of.iterations in the convergence

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

& Refresh your MFDS previous semester course basics

Next Session Plan :

Remaining Topics in Linear Models for
Regression
Linear models for classification



BITS Pilani
Pilani Campus

Machine Learning

DSE CLZG565

Linear Models for Regression & Classification

Raja vadhana P
Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 6

- Linear Regression
- Geometric Approach
- Notion of Linear Basis Function Models
- Estimating the Parameters : Closed Form Solutions
- Estimating the Parameters : Gradient Descent Algorithm

Module – 2

Prerequisites : Refresher only

- Linear Algebra
- Calculus
- Probability Theory

Module – 5

- Discriminant Functions
- Probabilistic Generative Classifiers
- Probabilistic Discriminative Classifiers
- Applications : Text classification model – Sentiment analysis using logistic regression

Learnt so far in previous session

- Linear Model for Regression & Geometric Interpretation
- Gradient Descent & Closed form Solutions
- Intuition of Cost Function
- Calculus
 - Notion of differentiation/Slope
 - Significance of Convex function
- Few Terminologies /Concepts
 - Parameters/ Coefficients/Weights
 - Error/Loss/Cost Function
 - Explanatory vs Response Variables
 - Gradient Descent
 - Stationary Points/ Maxima / Minima / Local Optima vs Global Optima
 - Initialization bias
 - Grid Search
 - Learning Rate/Step Size
 - Convergence Criteria

Agenda

- Linear Basis Models (Remaining in earlier Module)
- Linear Models for Classification
- Probabilistic Generative Model Vs Probabilistic Discriminative Model
- Logistic Regression
- Gradient Descent Optimization
- Math Preliminaries – Refresher (Wherever applicable will be refreshed in sync with discussion)

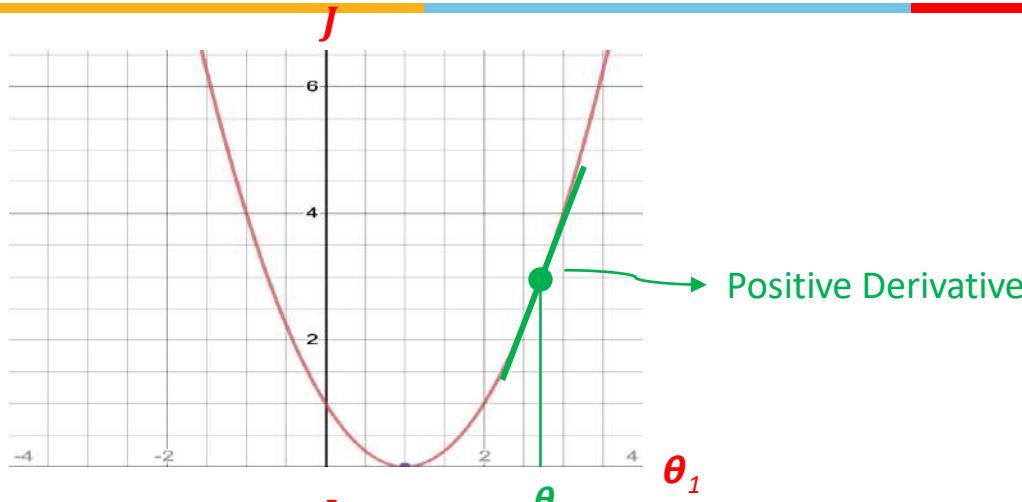
Student Queries

- How the GD is guaranteed to converge ie., move towards the minima?
- Do we need to scale all the features before modelling?
- How complex polynomial (non-linear) model can be implemented?



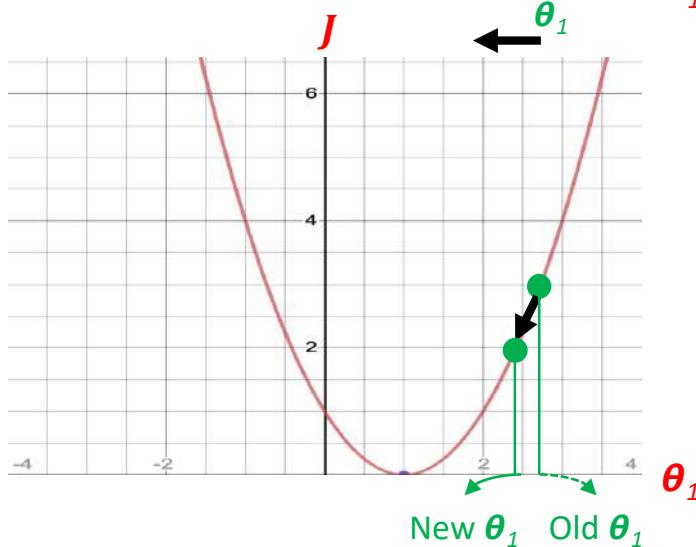
Convergence

Guarantee of Convergence



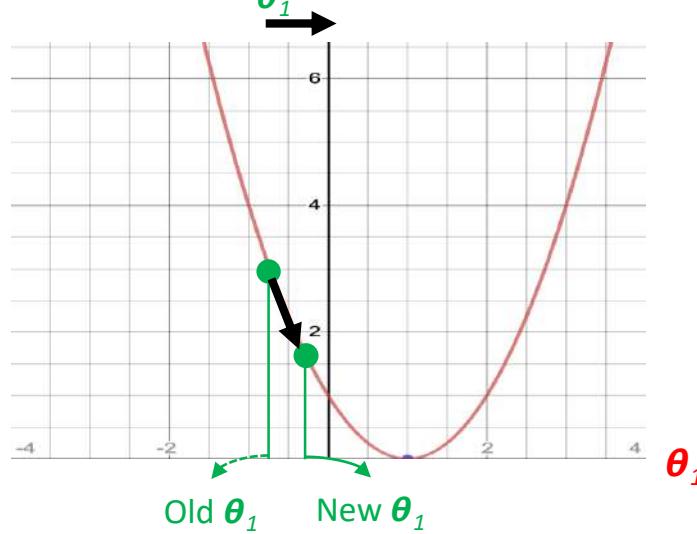
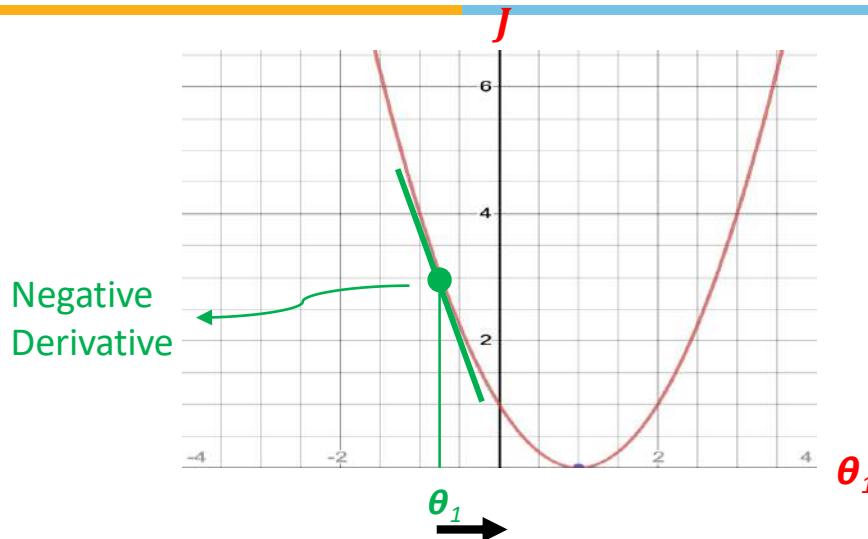
$$\begin{aligned}\theta_1 &= \theta_1 - \alpha \frac{d J(\theta_1)}{d \theta_j} \\ &= \theta_1 - \alpha (\text{Positive Number})\end{aligned}$$

Decrease θ_1 by a certain value



Source Credit: Prof. Mohammad Hammoud

Guarantee of Convergence



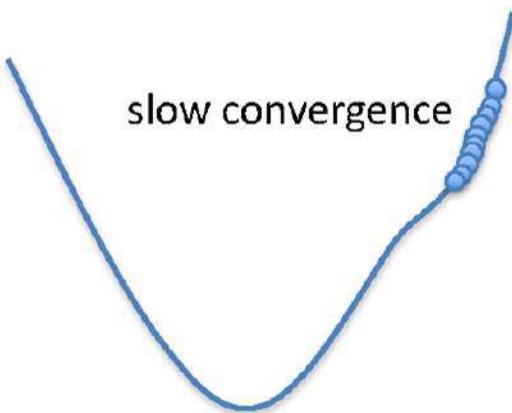
$$\begin{aligned}\theta_1 &= \theta_1 - \alpha \frac{d J(\theta_1)}{d \theta_j} \\ &= \theta_1 - \alpha (\text{Negative Number})\end{aligned}$$

Increase θ_1 by a certain value

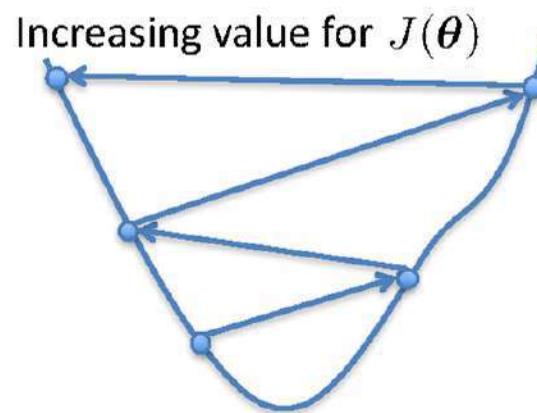
Source Credit: Prof. Mohammad Hammoud

Choosing Learning Rate

α too small



α too large



- Increasing value for $J(\theta)$
- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Gradient Descent algorithm : Effect of Feature Scaling

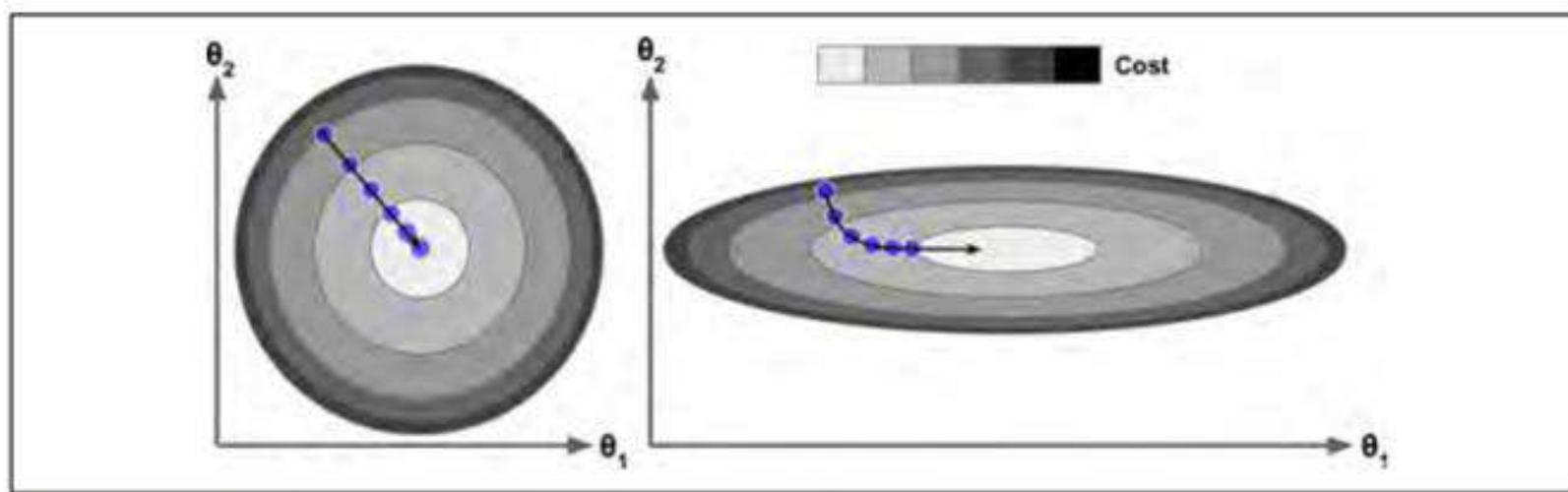


Figure 4-7. Gradient Descent with and without feature scaling

Gradient Descent algorithm : Effect of Learning Rate



Rate

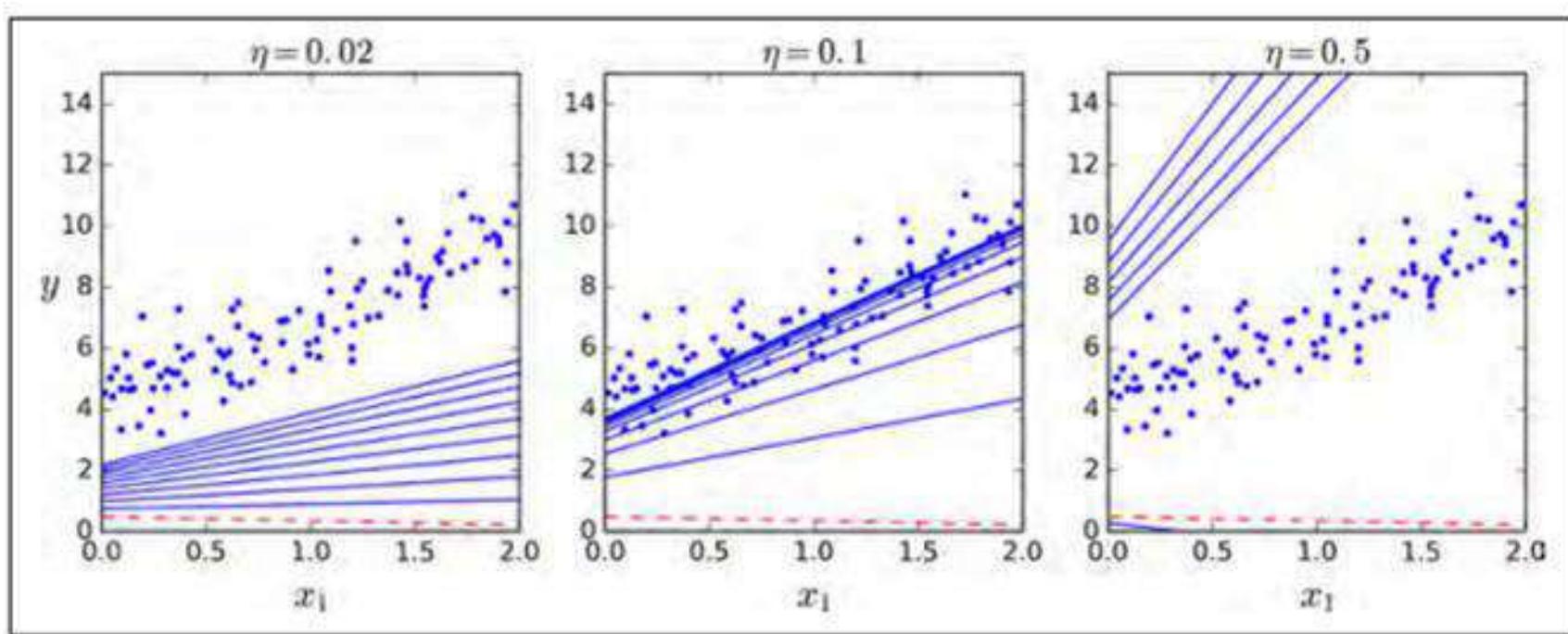
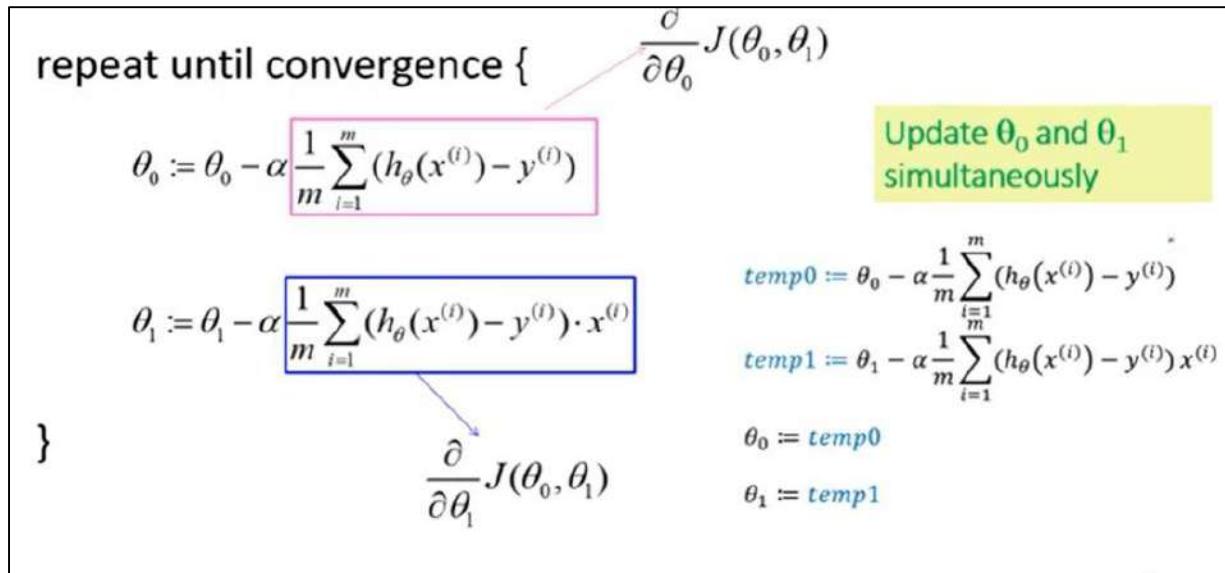


Figure 4-8. Gradient Descent with various learning rates

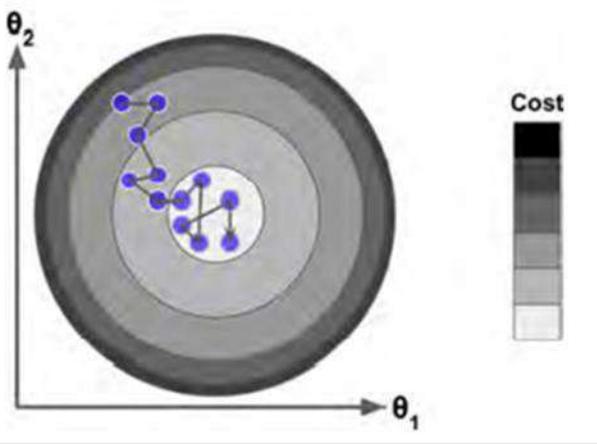
Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.
- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.
- **Stochastic** gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately

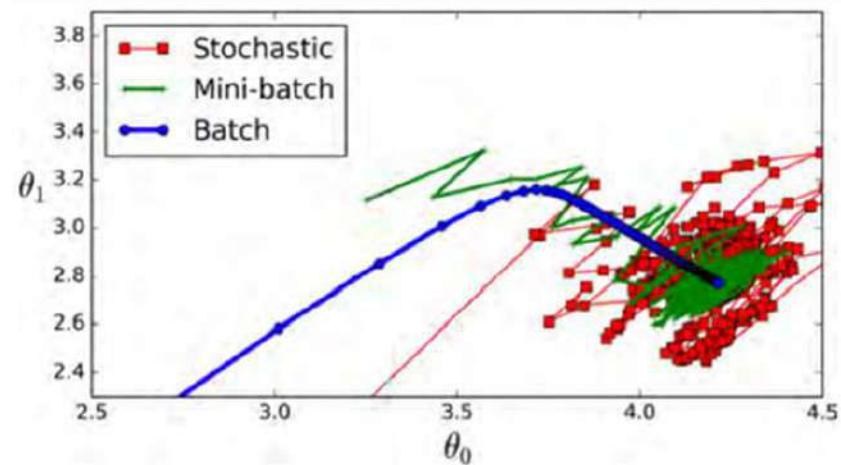


Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.
- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.
- **Stochastic** gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately



Stochastic Gradient Descent

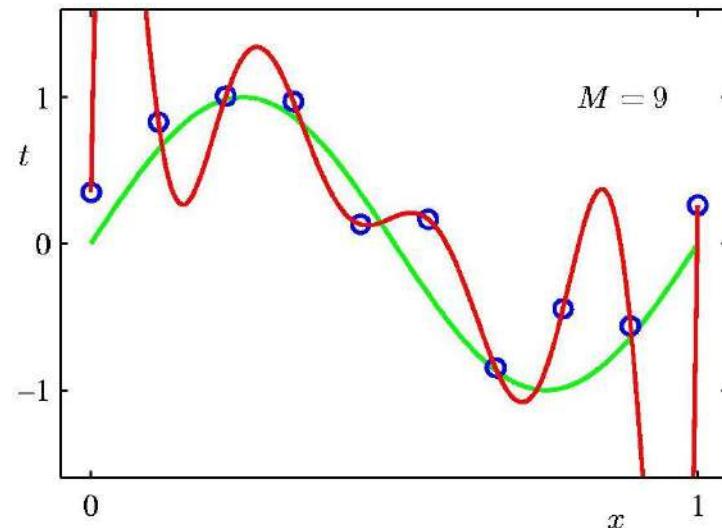
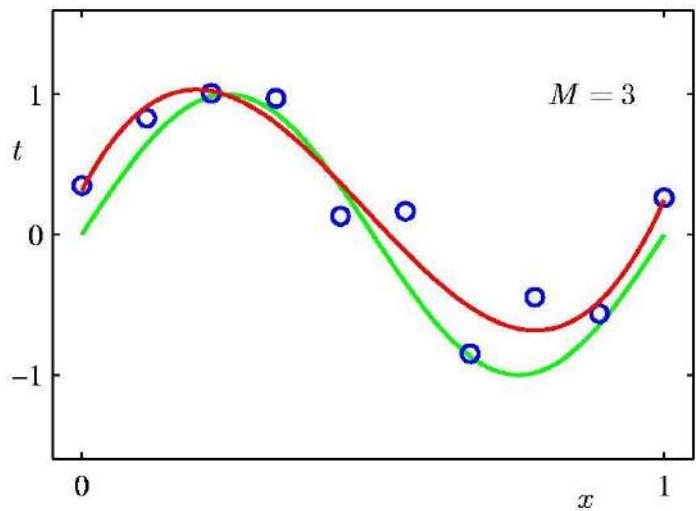
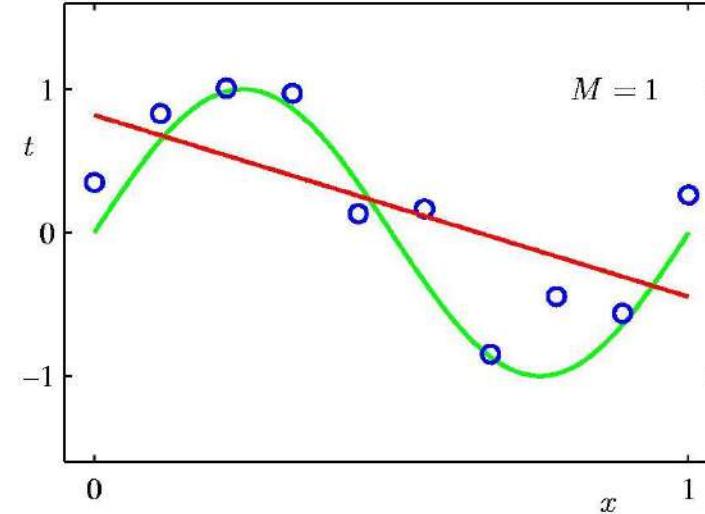
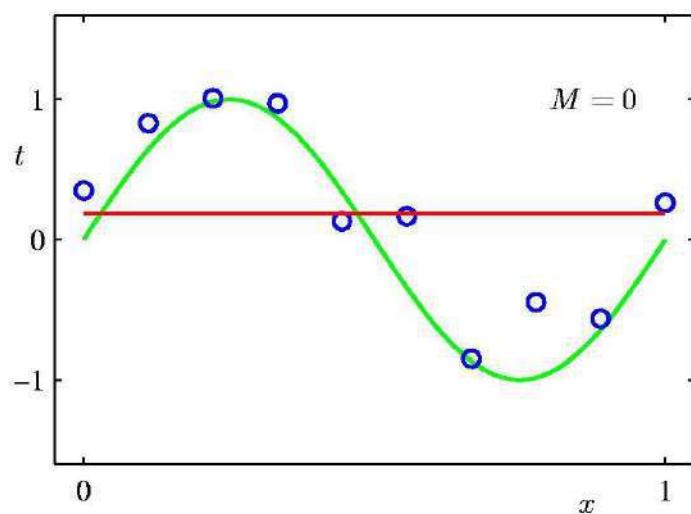


1. Gradient Descent paths in parameter space



Linear Basis Models

Polynomial Regression



Linear Basis Function Models

- The inputs \mathbf{X} for linear regression can be:
 - Original quantitative inputs
 - Transformation of quantitative inputs
 - e.g. log, exp, square root, square, etc.
 - Polynomial transformation
 - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
 - Basis expansions
 - Dummy coding of categorical inputs
 - Interactions between variables
 - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques
to fit non-linear datasets.

Linear Basis Function Models

Example: an M-th order polynomial function of one dimensional feature x:

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^M w_j x^j$$

where x^j = j-th power of x

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$

$\phi_j(x)$ are known as *basis functions*. Typically, $\phi_0(x) = 1$, so that w_0 acts as a bias.

In the simplest case, we use linear basis functions : $\phi_d(x) = x_d$.

They are called **linear models** because this function is
linear in \mathbf{w} .

X No.of.Years of Experience (in Years)	X^2	Y Salary Of the Employee (in Lakhs)
1	1	2
2	4	3
3	9	4
4	16	5
5	25	6

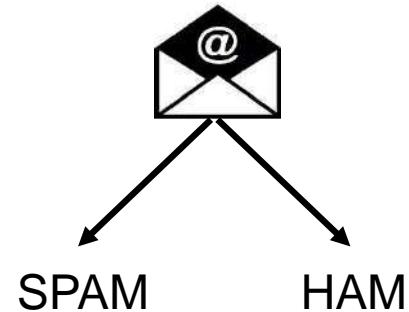
$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

Probability review

Two main "schools of thought"

- **Bayesian probability= degree of belief**
 - Probabilities are assigned to events based on evidence and personal belief and are centered around Bayes' theorem
 - $p(\text{email}=\text{SPAM})=0.4$ means you think the event that a particular email will be a spam is 40% likely.
- **Frequentist probability= long run frequencies**
 - Events are observed and counted, and their frequencies provide the basis for directly calculating a probability
 - $p(\text{email}=\text{SPAM})=0.4$ means that the empirical fraction of times this event will occur across infinitely repeated trials/observations in a mail box

Spam Classifier



Sample Space, S

- Email Type

$$S = \{\text{Spam}, \text{Ham}\}$$

Event, E

- An email is observed as Spam

$$E = \{\text{Spam}\}$$

$$P(E) = \lim_{n \rightarrow \infty} \frac{n(E)}{n}$$

n = # of total trials

$n(E)$ = # trials where E occurs

Conditional Probability and Independence

Let B be any event such that $P(B) \neq 0$

Conditional Probability :

$$P(A|B) := \frac{P(A \cap B)}{P(B)}$$

Joint Probability : $P(AB)$

Generally: $P(A)P(B|A)$

Independence:

$A \perp B$ if and only if $P(A \cap B) = P(A)P(B)$

$$\begin{aligned} A \perp B \text{ if and only if } P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} = P(A) \end{aligned}$$

B = Event of occurrence of word "Dear" in email

A = Event of occurrence of word "Lottery" in email

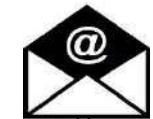
SPAM

HAM

$$\begin{aligned} &P(\text{"Lottery" | "Dear"}) \\ &= \frac{P(\text{Observing both words in same email})}{P(\text{Observing of word "Dear" in email})} \end{aligned}$$

$$\begin{aligned} &P(\text{Lottery, Dear}) \\ &= P(\text{Lottery}) * P(\text{Dear} | \text{Lottery}) \end{aligned}$$

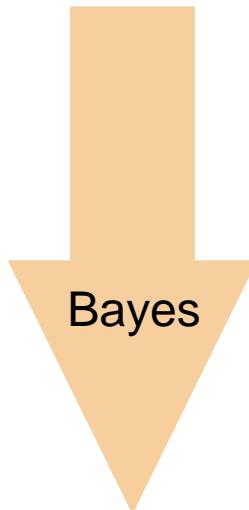
$$\begin{aligned} &\text{If } \text{Lottery} \perp \text{Dear} \text{ then} \\ &P(\text{Dear} | \text{Lottery}) = P(\text{Dear}) \\ &P(\text{Lottery, Dear}) = P(\text{Lottery}) * P(\text{Dear}) \end{aligned}$$



Spam Classifier

Bayes' Theorem

$P(E = \text{Evidence} | F = \text{Fact})$
(collected from data)



From the observations so far
(training data):
 $P(\text{"Lottery"} | \text{"SPAM"}) = 0.8$
 $P(\text{"SPAM"}) = 0.6$

posterior	likelihood	prior	
$P(F E) = \frac{P(E F)P(F)}{P(E)}$			
	normalization constant		

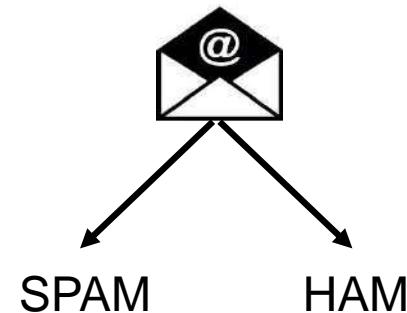
Given new evidence E , update belief of fact F

Prior belief \rightarrow Posterior belief
 $P(F) \rightarrow P(F|E)$

$P(F = \text{Fact} | E = \text{Evidence})$
(categorize a new data point)

For new data to predict:
 $P(\text{"SPAM"} | \text{"Lottery"}) = ?$

Spam Classifier



E = Event of occurrence of word "Lottery" in email

F = The email was categorized as SPAM

Bayes' Theorem

- Given the conditional probability of an event $P(x|y)$
- Want to find the "reverse" conditional probability, $P(y|x)$

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

where: $P(x) = \sum_{y' \in \text{value } y} P(x|y')P(y')$

X and Y are continuous

$$f(y|x) = \frac{f(x|y)f(y)}{f(x)}$$

where: $f(x) = \int_{y' \in \text{value } y} f(x|y')f(y')dy'$

Exercise

- 60% of all email received is spam.
- 20% of spam has the word “Dear”
- 1% of non-spam has the word “Dear”

You get an email with the word “Dear” in it. What is the probability that the email is spam?



Decision Theory & Classification Models

Inductive Learning Hypothesis : Interpretation

- Target Concept : t
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Humidity	Wind	Water	Forecast	<i>EnjoySport?</i>
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Decision Theory

- Target Concept : t
- Discrete : $f(x) \in \{\text{Yes, No}\}$ ie., $t \in \{0, 1\}$ Binary Classification
- Continuous : $f(x) \in [20-100]$
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Humidity	Wind	Water	Forecast	<i>EnjoySport?</i>
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Decision Theory :

The decision problem: given x , predict t according to a probabilistic model $p(x, t)$

- Target Concept : t
- Discrete : $f(x) \in \{\text{Yes, No}\}$ ie., $t \in \{0, 1\}$
- Continuous : $f(x) \in [20-100]$
- Probability Estimation : $f(x) \in [0-1]$

$p(x, C_k)$ is the (central!) inference problem

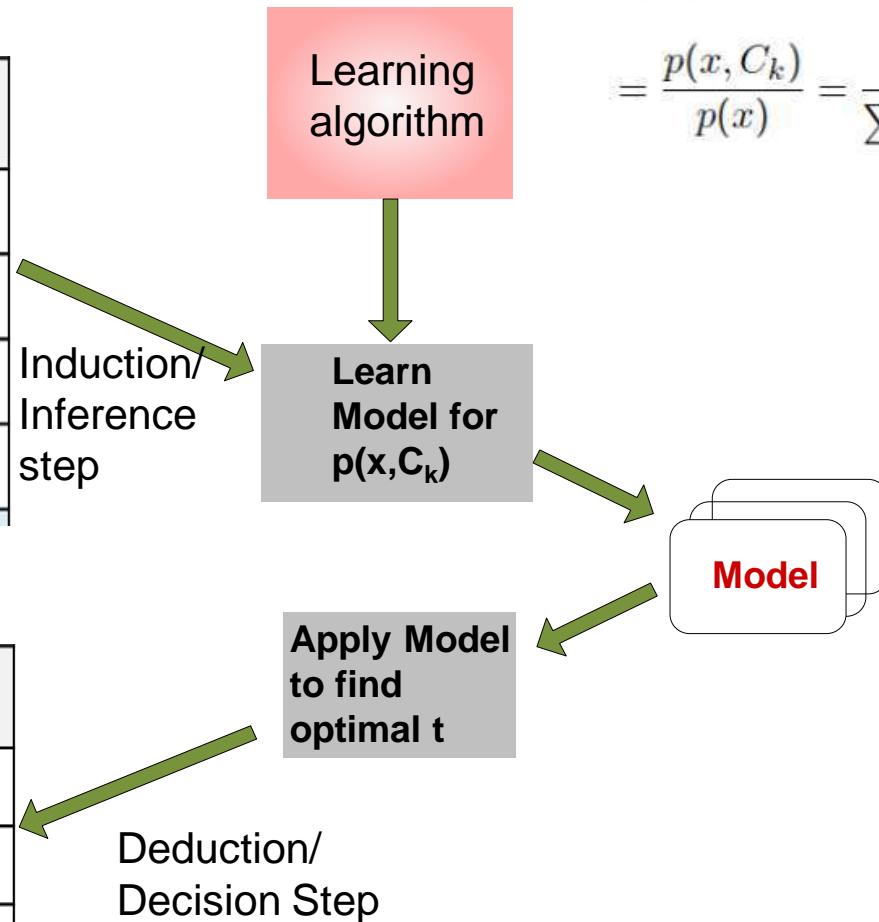
Sky	AirTemp	Humidity	Wind	Water	Forecast	$P(\text{EnjoySport} = \text{Yes})$
X = <Sunny , Warm , Normal , Strong , Warm , Same >						$\rightarrow 0.95 = P(C_k X)$
Sunny	Warm	High	Strong	Warm	Same	0.7
Rainy	Cold	High	Strong	Warm	Change	0.5
Sunny	Warm	High	Strong	Cool	Change	0.6

Classification Problem: Stages

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}.$$

$$= \frac{p(x, C_k)}{p(x)} = \frac{p(x, C_k)}{\sum_{k=1}^2 p(x, C_k)}$$

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

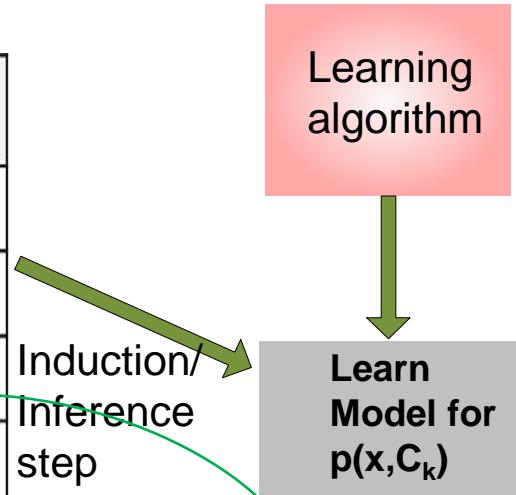


Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Rainy	Cold	High	Strong	Change	?
Sunny	Warm	High	Strong	Change	?
Rainy	Warm	Normal	Breeze	Same	?

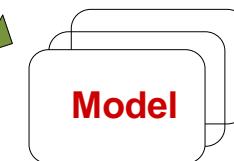
Test Set

Decision Region

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes



Model divides the input space into regions R_k called **decision regions**, one for each class, such that all points in R_k are assigned to class C_k . A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 .



Training Set

$$p(x, C_1)$$

$$p(x, C_2)$$

$$p(x, C_1)$$

$$p(x, C_2)$$

Misclassification Rate

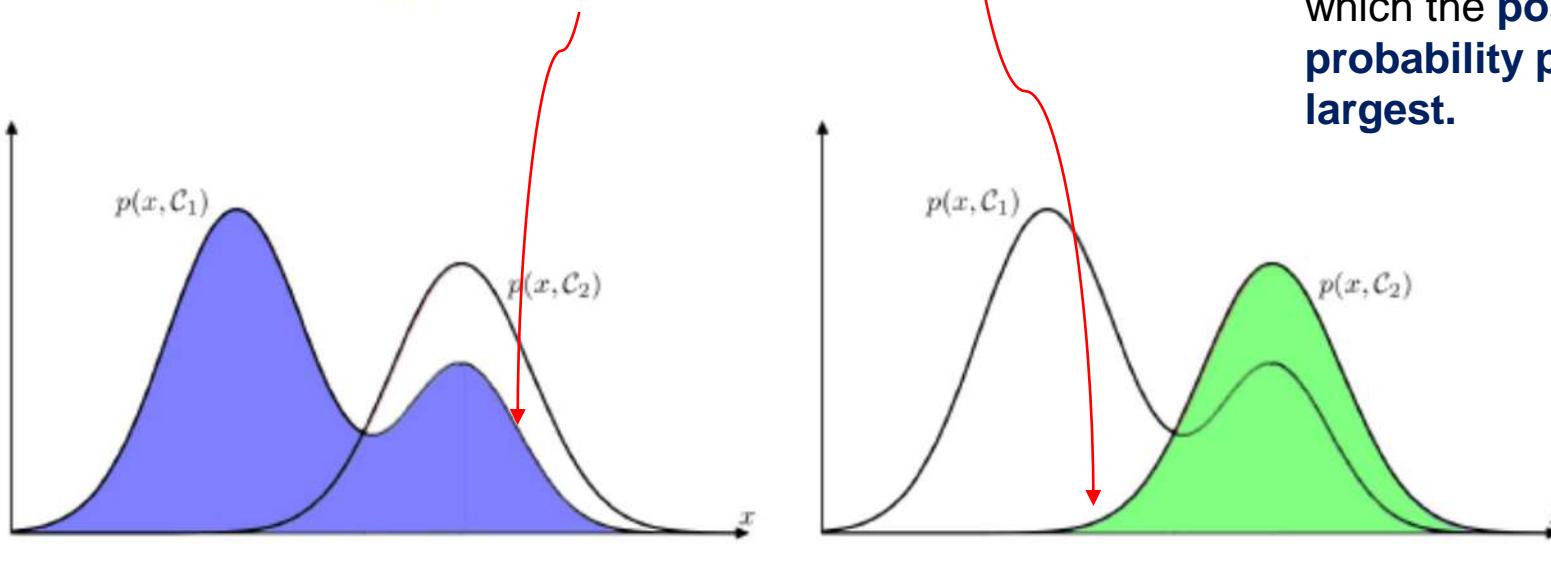
$$p(C_k|x) = \frac{p(x, C_k)}{p(x)}$$

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned}$$

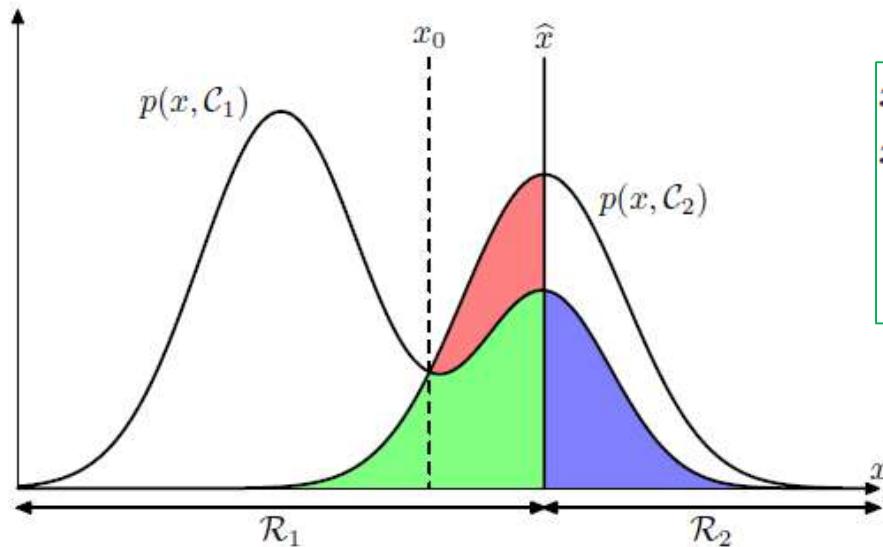
$$\begin{aligned} p(x, C_1) &> p(x, C_2) \\ \Leftrightarrow p(C_1|x)p(x) &> p(C_2|x)p(x) \\ \Leftrightarrow p(C_1|x) &> p(C_2|x) \end{aligned}$$

To minimize $p(\text{mistake})$, each \mathbf{x} is assigned to whichever class has the smaller value of the integrand

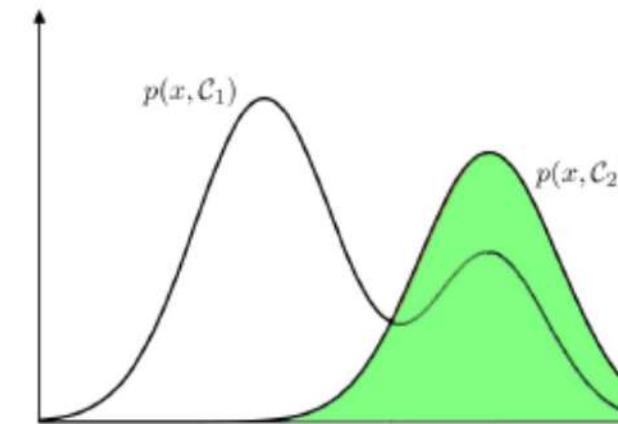
The minimum probability of making a mistake is obtained if each value of \mathbf{x} is assigned to the class for which the **posterior probability $p(C_k|x)$** is largest.



Decision Theory - Summary



\hat{x} : decision boundary.
 x_0 : optimal decision boundary
 $x_0 : \arg \min_{\mathcal{R}_1} \{p(\text{mistake})\}$





Linear Models for Classification

Types of Classification

Inductive Learning Hypothesis : Interpretation

- Target Concept
- Discrete : $f(x) \in \{\text{Yes, No, Maybe}\}$ Classification
- Continuous : $f(x) \in [20-100]$ Regression
- Probability Estimation : $f(x) \in [0-1]$

Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

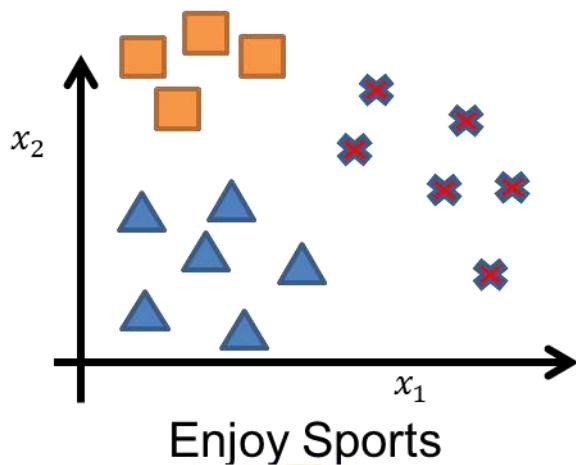
Types of Classification



Output Labels

- Target Concept

Multi Class

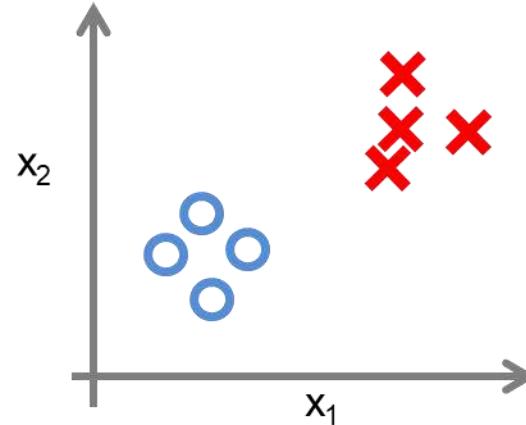


Enjoy Sports

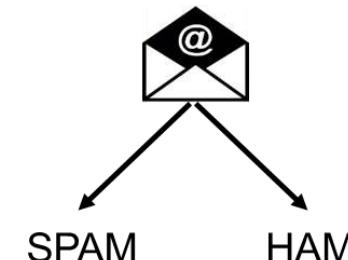


YES NO MAYBE

Binary

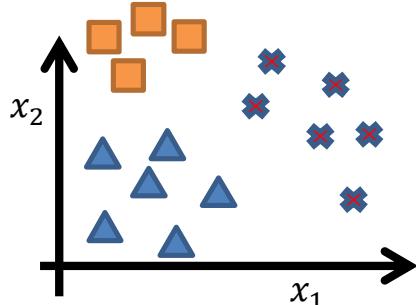


Spam Classifier



Prediction – Multi class Classification

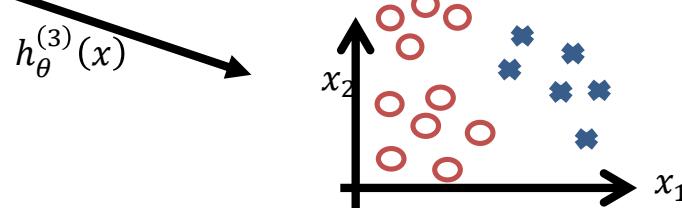
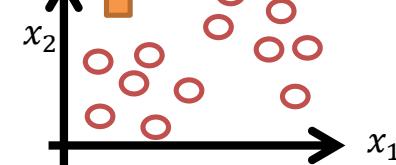
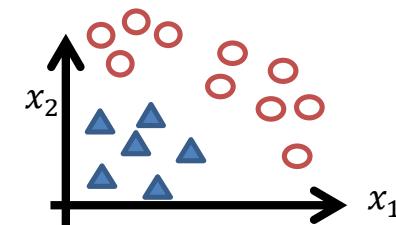
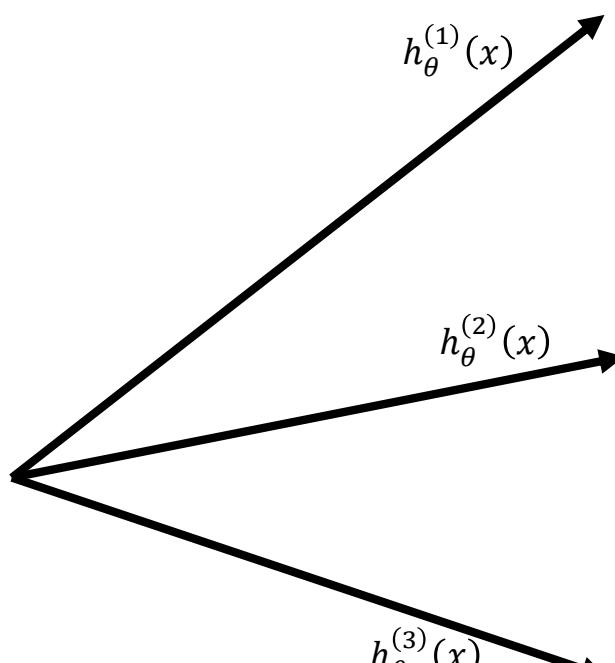
One Vs All Strategy(one-vs-rest)



Class 1:
 Class 2:
 Class 3:

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$

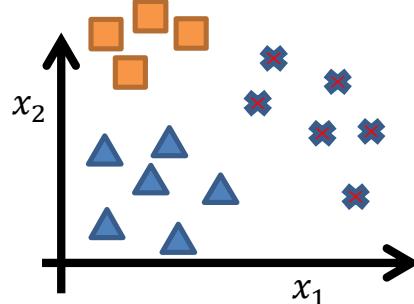
For input x Predict : $\max_i h_{\theta}^{(i)}(x)$



Note: Scikit-Learn detects when you try to use a binary classification algorithm for a multi-class classification task, and it automatically runs OvA (except for SVM classifiers for which it uses OvO)

Prediction – Multi class Classification

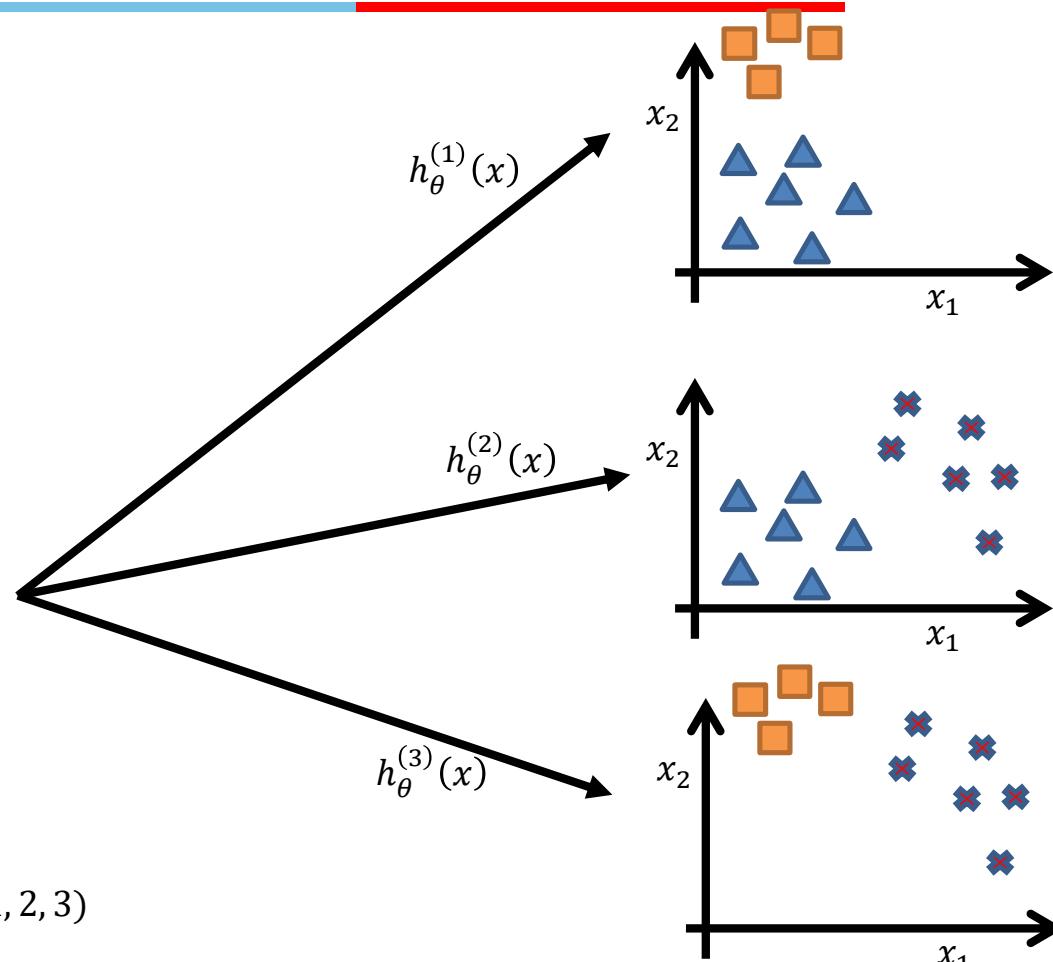
One Vs One Strategy



Class 1:
 Class 2:
 Class 3:

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$

For input x Predict : $\max_i h_{\theta}^{(i)}(x)$



$N \times (N - 1) / 2$ classifiers

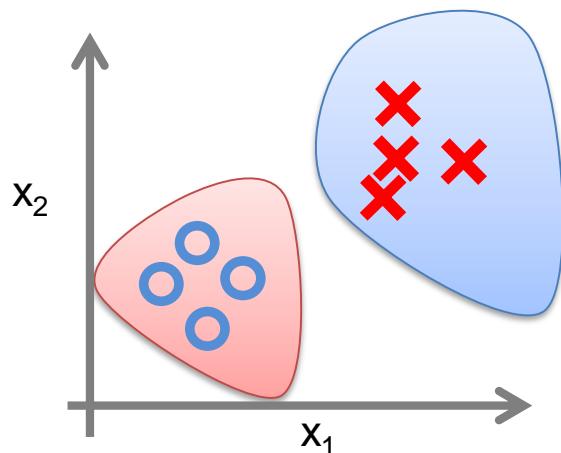
Types of Classification



Decision Theory: Interpretation

Model Building

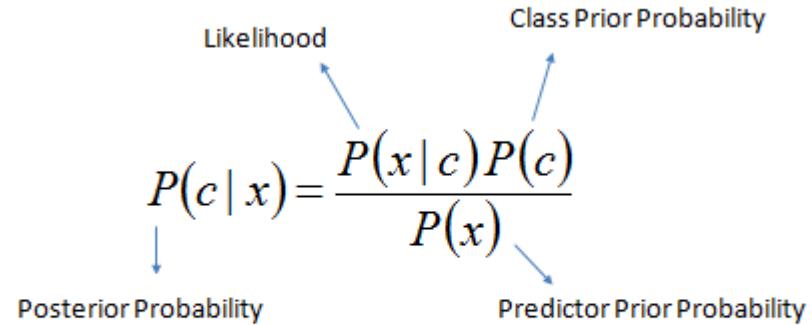
Generative



$$P(Y | X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d | Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

Known as generative models, because by sampling from them it is possible to generate synthetic data points in the input space.

Eg., Gaussians, **Naïve Bayes**, Mixtures of multinomials, **Mixtures of Gaussians**, Bayesian networks



$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

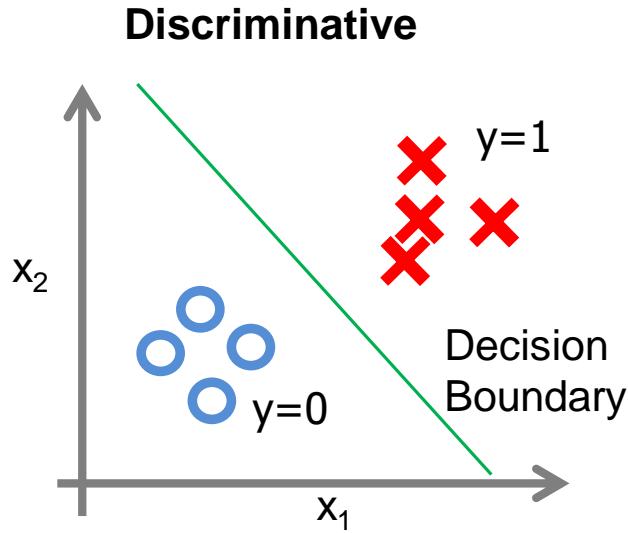
Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Types of Classification



Decision Theory: Interpretation

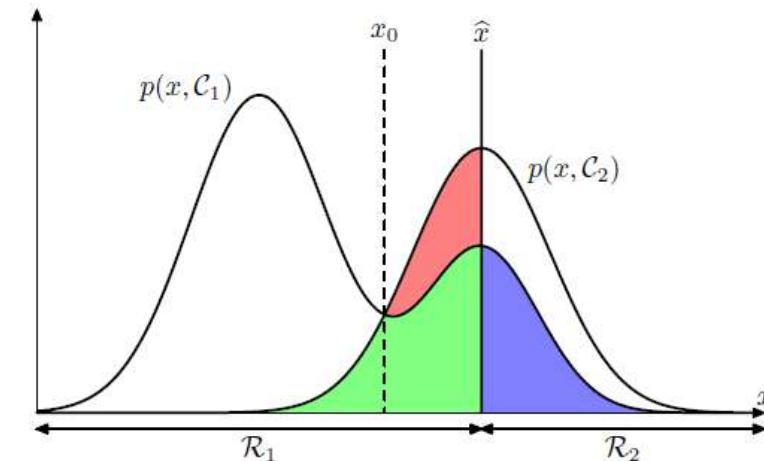
Model Building



$$\theta_0 + \sum_i \theta_i x_i \geq 0$$

$$\theta_0 + \sum_i \theta_i x_i < 0$$

Logistic regression, SVMs , tree based classifiers (e.g. decision tree) Traditional neural networks, Nearest neighbor

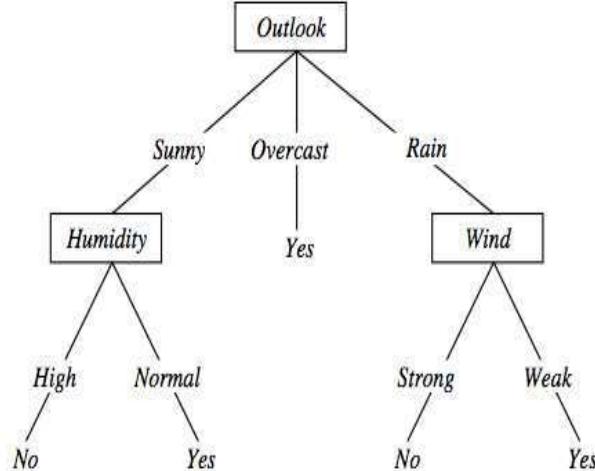


Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Types of Classification

Decision Theory: Interpretation

Model Building



$IF\ OUTLOOK = Overcast\ THEN\ PLAY = Yes$
 ELSE
 $IF\ OUTLOOK = Rain\ AND\ WIND = Strong\ THEN\ PLAY = No$

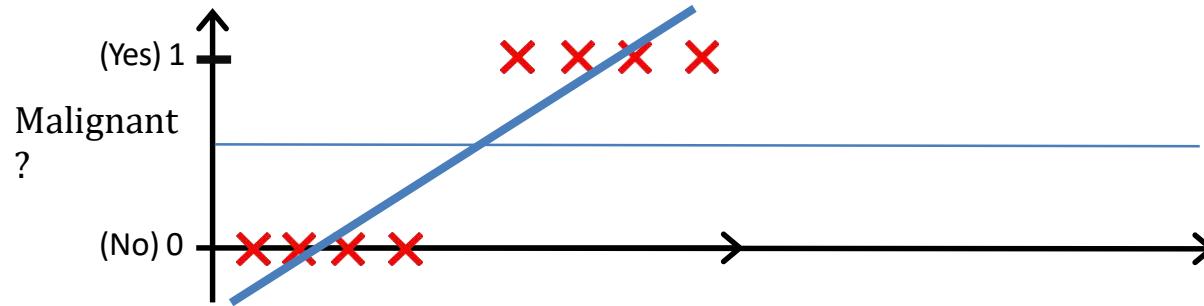
Logistic regression, SVMs , tree based classifiers (e.g. decision tree) Traditional neural networks, Nearest neighbor

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes



Logistic Regression

Logistic Regression vs Least Squares Regression



- Tumor Size
- Can we solve the problem using linear regression? E.g., fit a straight line and define a threshold at 0.5
- Threshold classifier output $h_{\theta}(x)$ at 0.5:

A Discriminant function $f(x)$ directly map input to class labels
In two-class problem, $f(.)$ is binary valued

If $h_{\theta}(x) \geq 0.5$, predict “ $y = 1$ ”
If $h_{\theta}(x) < 0.5$, predict “ $y = 0$ ”

Decision Rules

- Classifier:

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \mathbf{w}^T \mathbf{x}$$
 (linear discriminant function)

- Decision rule is

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}, \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

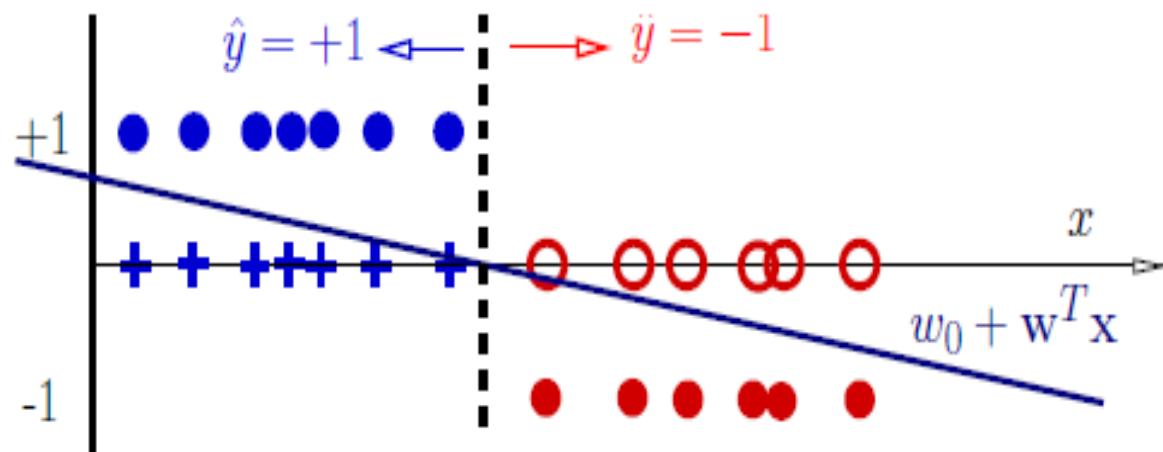
- Mathematically

$$y = \text{sign}(w_0 + \mathbf{w}^T \mathbf{x})$$

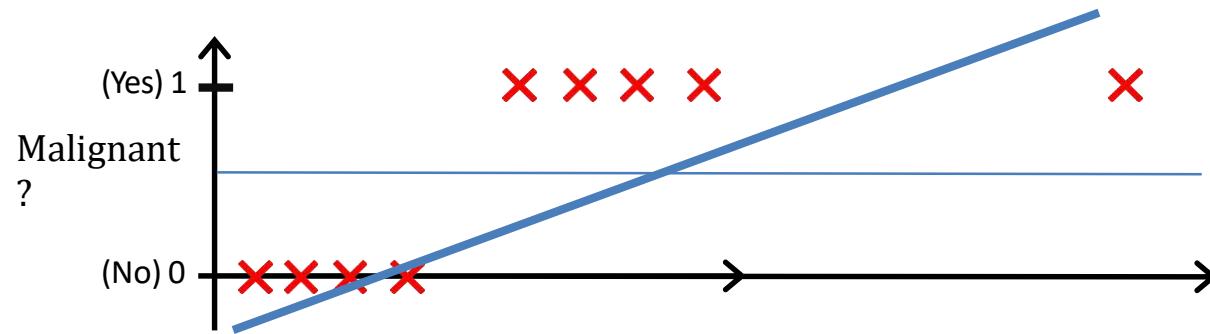
- This specifies a linear classifier: it has a linear boundary (hyperplane)

$$w_0 + \mathbf{w}^T \mathbf{x} = 0$$

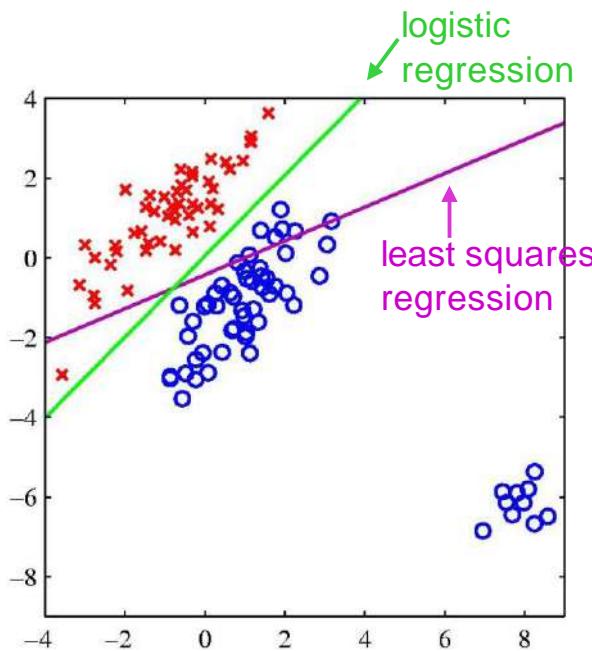
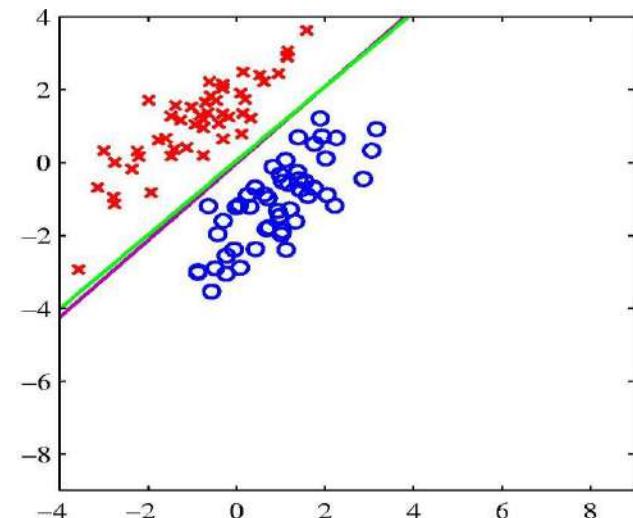
A discriminant is a function that takes an input vector \mathbf{x} and assigns it to one of K classes, denoted C_k .



Logistic Regression vs Least Squares Regression



Logistic Regression vs Least Squares Regression



The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Logistic Regression vs Least Squares Regression

Classification: $y = 0 \text{ or } 1$

$h_\theta(x)$ can be > 1 or < 0

What we need: $0 \leq h_\theta(x) \leq 1$

Logistic Regression:

$$Y = \theta_0 + \sum_i \theta_i x_i$$

$$p = \frac{1}{1 + e^{-y}} \quad h_\theta(x) = g(\theta^T x)$$

$$\ln \frac{p}{1-p} = \theta_0 + \sum_i \theta_i x_i$$

Learning model parameters

- Training set:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

- m examples

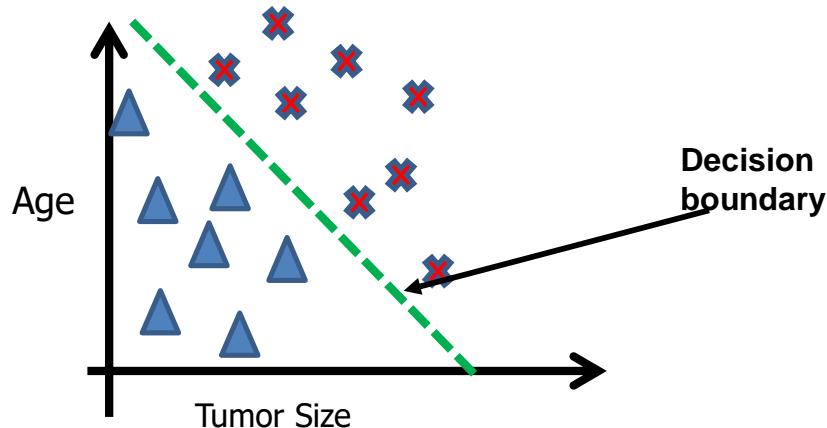
$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

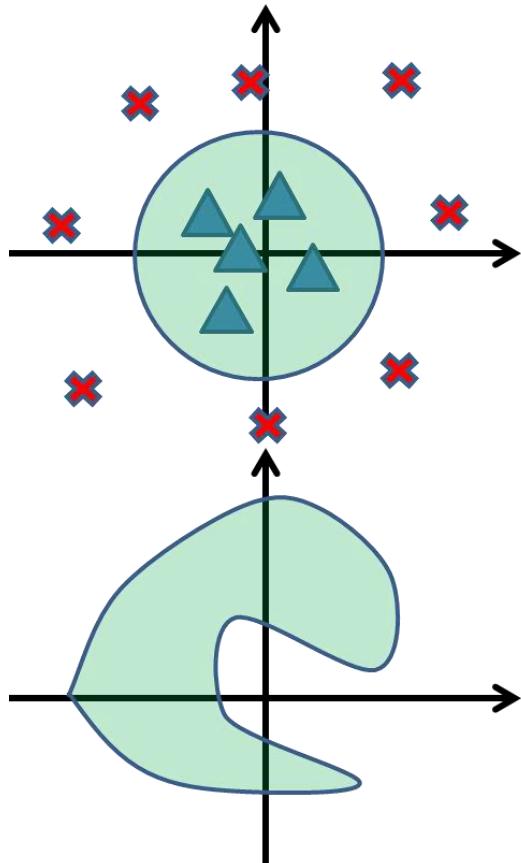
- How to choose parameters (feature weights) ?

Logistic Regression

- At decision boundary output of logistic regression is 0.5
- $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
 - e.g., $\theta_0 = -3, \theta_1 = 1, \theta_2 = 1$



Logistic Regression



- $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$
E.g., $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$
- Predict “ $y = 1$ ” if $-1 + x_1^2 + x_2^2 \geq 0$

- $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$

Logistic Regression

- Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

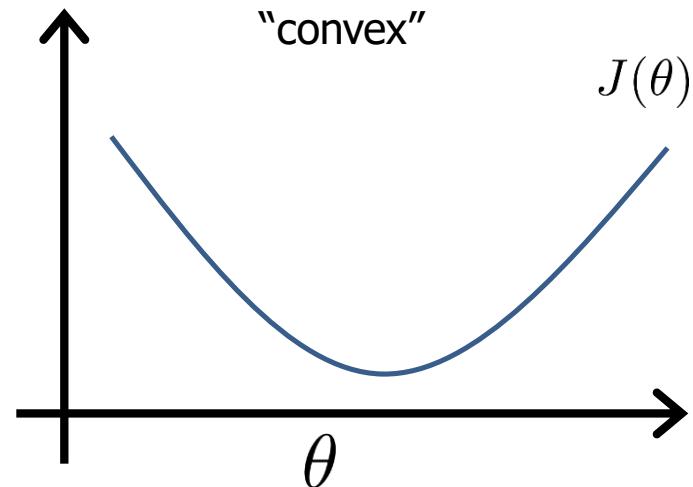
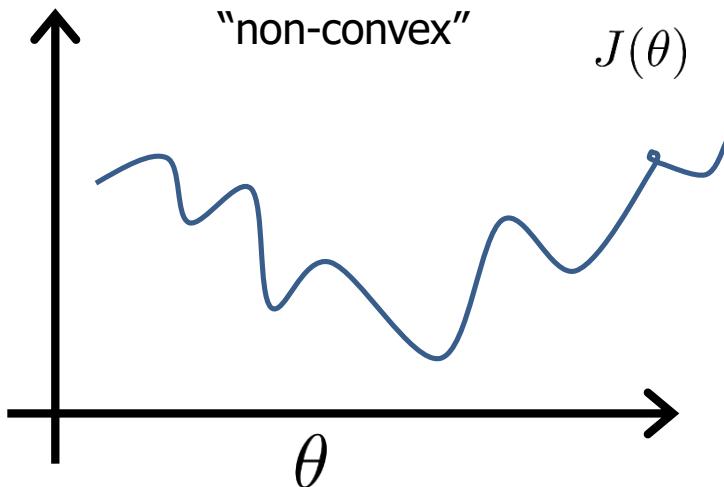
- m examples
 - n features
- $$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- How to choose parameters (feature weights)? θ

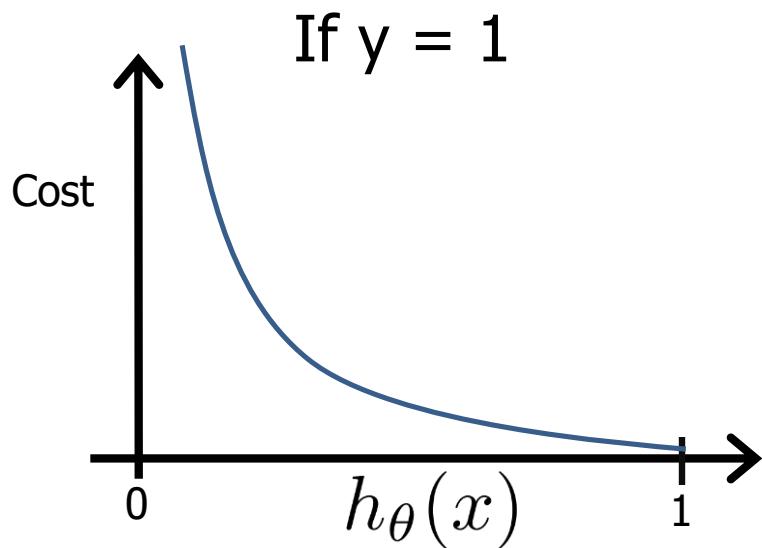
Logistic Regression

- Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- How to choose parameters (feature weights)? θ



Logistic regression cost function (cross entropy)

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

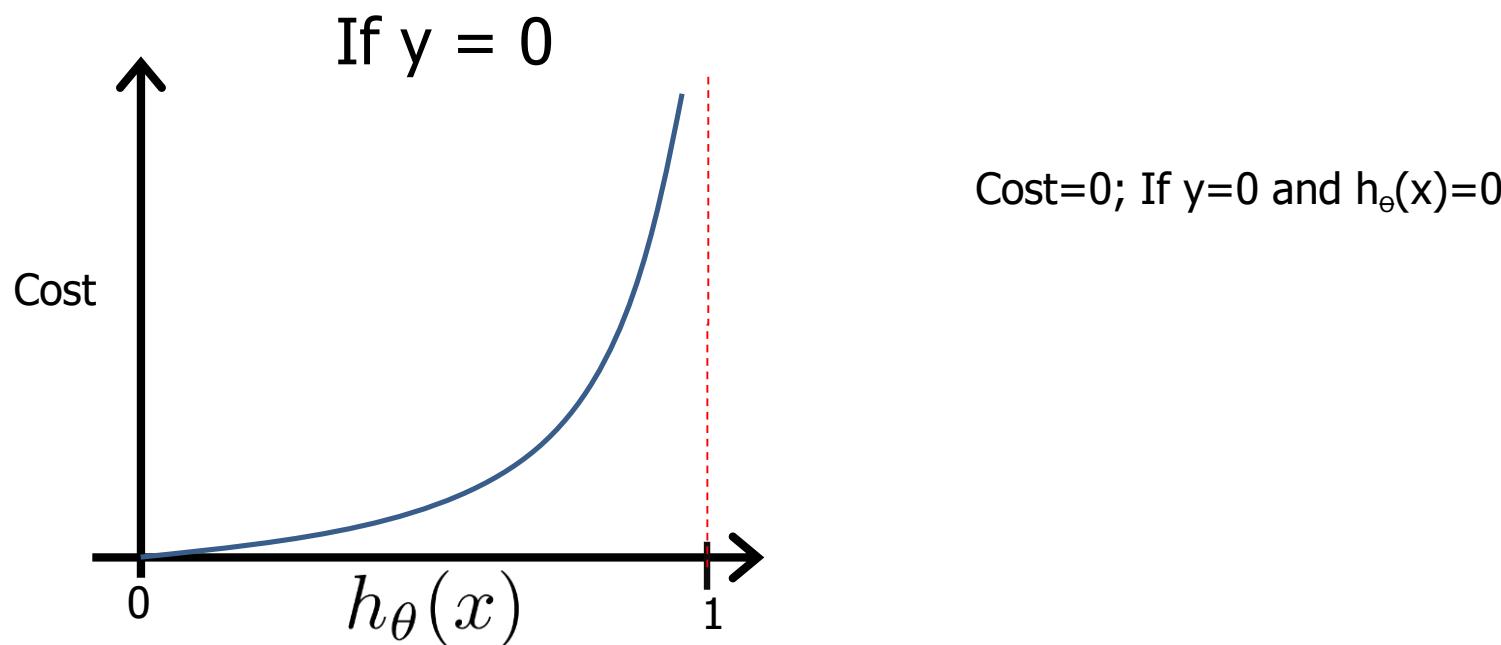


$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
 But as $h_\theta(x) \rightarrow 0$
 $\text{Cost} \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$,
 $(\text{predict } P(y = 1|x; \theta) = 0)$, but $y = 1$,
 we'll penalize learning algorithm by a very
 large cost.

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



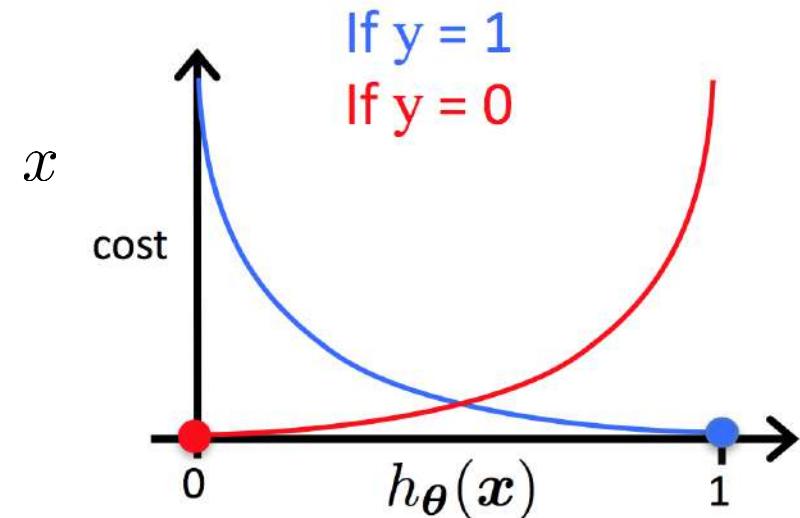
Cost function

$$\begin{aligned}
 J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\
 &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]
 \end{aligned}$$

To fit parameters θ : Apply Gradient Descent Algorithm $\min_{\theta} J(\theta)$

To make a prediction given new :

Output :
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



Gradient Descent Algorithm

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Goal: $\min_{\theta} J(\theta)$

Repeat

{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient Descent Algorithm

Linear Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

$$h_\theta(x) = \theta^\top x$$

Logistic Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

Slide credit: Andrew Ng

Example: Sentiment Analysis

It's **hokey**. There are virtually **no** surprises , and the writing is **second-rate**.

So why was it so **enjoyable** ? For one thing , the cast is

great . Another **nice** touch is the music **I** was overcome with the urge to get off the couch and start dancing . It sucked **me** in , and it'll do the same to **you** .

Var	Definition	$x_1=3$	$x_5=0$	$x_6=4.19$	Value in Fig. 5.2 Sentiment Features
x_1	count(positive lexicon) \in doc)			3	
x_2	count(negative lexicon) \in doc)			2	
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$			1	
x_4	count(1st and 2nd pronouns \in doc)			3	
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$			0	
x_6	log(word count of doc)			$\ln(66) = 4.19$	

Classifying sentiment using logistic regression

Suppose $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

$b = 0.1$

$$\begin{aligned} p(+|x) = P(Y = 1|x) &= \sigma(w \cdot x + b) \\ &= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= \sigma(.833) \\ &= 0.70 \end{aligned}$$

$$\begin{aligned} p(-|x) = P(Y = 0|x) &= 1 - \sigma(w \cdot x + b) \\ &= 0.30 \end{aligned}$$

Logistic Regression – Exercise for Practice

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

$$\theta_0 := \theta_0 - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) \quad (1)$$

$$\theta_{CGPA} := \theta_{CGPA} - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) x_{CGPA}^{(i)}$$

$$\theta_{IQ} := \theta_{IQ} - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) x_{IQ}^{(i)}$$

This Example -2 numerical problem we will discuss this in next class. You are advised to try working it for practice.

Hyper parameters:

Learning Rate = 0.3

Initial Weights = (0.5, 0.5, 0.5)

Regularization Constant = 0

$$\theta^\top X = 0.5 + 0.5 \text{ CGPA} + 0.5 \text{ IQ}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$h_\theta(x) = \frac{1}{1 + e^{-(0.5 + 0.5 \text{ CGPA} + 0.5 \text{ IQ})}}$$

Approx. : New weights

$$\theta_0 = 0.4$$

$$\theta_{1=CGPA} = -0.4$$

$$\theta_{2=IQ} = -0.6$$

Logistic Regression – Python for Practice

In your virtual lab file of this module perform below changes and interpret the results :

1. In your test result use the function `.predict_proba(X)` . This will return the class wise probability estimates before the actual discrete/binary class values are assigned
2. In the third python file LogisticRegression function call, add additional parameter like : `multi_class = 'ovr'` vs `multi_class = 'auto'`
3. In the second file where gradient descent is coded, similar to our last class PPT uploaded combination (refer there) , try to change the `n_iterations` and `learning_rate` and interpret the results

Summary :

Logistic regression (Classification)

- **Model**

$$h_{\theta}(x) = P(Y = 1|X_1, X_2, \dots, X_n) = \frac{1}{1+e^{-\theta^T x}}$$

- **Cost function**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \quad \text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- **Learning**

Gradient descent: Repeat $\{\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}\}$

- **Inference**

$$\hat{Y} = h_{\theta}(x^{\text{test}}) = \frac{1}{1 + e^{-\theta^T x^{\text{test}}}}$$

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 4 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

& Refresh your MFDS previous semester course basics

Next Session Plan :

Modules 3&4

Bayesian Learning

Bayesian Classifiers



BITS Pilani
Pilani Campus

Machine Learning

DSE CLZG565

Bayesian Learning & Bayesian Classifiers

Raja vadhana P

Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 3 & 4

- MLE Hypothesis
 - MAP Hypothesis
 - Bayes optimal classifier
 - Gibbs Algorithm
-
- Naïve Bayes Classifier

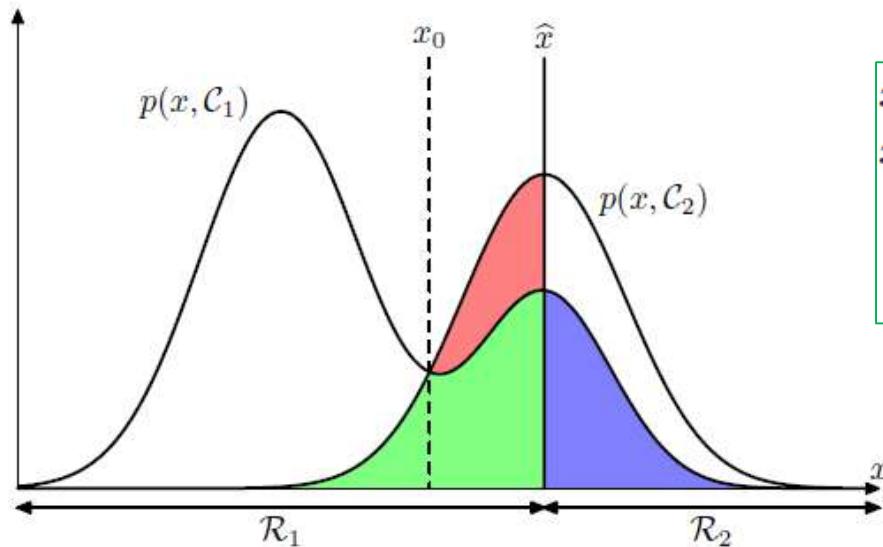
Learnt so far in previous session

- Effect of Learning Rate , Scaling in Gradient Descent Convergence
- Types of Gradient Descent
- Classification Model Interpretation
- Decision Theory – Probabilistic Classification Models
- Logistic Regression - Classifier
- Intuition of Cross Entropy Cost Function
- Probability
 - Probability concepts for ML
 - Bayes Theorem significance in ML
- Few Terminologies /Concepts
 - Joint Probability vs Conditional Probability, Chain Rule, Independence
 - Probabilities : Prior, Likelihood, Posterior
 - Decision Regions, Decision Boundary, Optimal Decision Boundary
 - Misclassification Rate
 - Logistic Regression
 - Sigmoid Function
 - Cross Entropy Cost function
 - Text Classification Features for Logistic regression

Agenda

- Logistic Regression (Refresher)
- Parameter Estimation
- Approaches
 - Maximum Likelihood Estimation (MLE)
 - Maximum A Posteriori (MAP)

Decision Theory – Last class Summary



\hat{x} : decision boundary.
 x_0 : optimal decision boundary
 $x_0 : \arg \min_{\mathcal{R}_1} \{p(\text{mistake})\}$



Logistic Regression – Exercise for Practice

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

$$\theta_0 := \theta_0 - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) \quad (1)$$

$$\theta_{CGPA} := \theta_{CGPA} - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) x_{CGPA}^{(i)}$$

$$\theta_{IQ} := \theta_{IQ} - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) x_{IQ}^{(i)}$$

Hyper parameters:

Learning Rate = 0.3

Initial Weights = (0.5, 0.5, 0.5)

Regularization Constant = 0

Use the scaled feature IQ in the second column

$$\theta^\top X = 0.5 + 0.5 \text{ CGPA} + 0.5 \text{ IQ}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$h_\theta(x) = \frac{1}{1 + e^{-(0.5 + 0.5 \text{ CGPA} + 0.5 \text{ IQ})}}$$

Logistic Regression – Exercise for Practice

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

$$\theta_0 := \theta_0 - 0.3 \frac{1}{6} \sum_{i=1}^6 (h_\theta(x^{(i)}) - y^{(i)}) \quad (1)$$

$$:= \theta_0 - 0.3 \frac{1}{6} \sum_{i=1}^6 \left(\frac{1}{1 + e^{-(0.5 + 0.5 \text{CGPA} + 0.5 \text{IQ})}} - y^{(i)} \right) \quad (1)$$

$$:= \theta_0 - 0.3 \frac{1}{6} \left[\left(\frac{1}{1 + e^{-(0.5 + 0.5(5.5) + 0.5(6.7))}} - 1 \right) * 1 + \left(\frac{1}{1 + e^{-(0.5 + 0.5(5) + 0.5(7))}} - 1 \right) * 1 + \dots \right]$$

$$:= \theta_0 - 0.3 \frac{1}{6} (-0.0014 + 0.9985 - 0.0006 - 0.0002 + 0.9994 + 0.9996)$$

$$:= 0.5 - 0.3(0.4992) = 0.3502$$

Hyper parameters:

Learning Rate = 0.3

Initial Weights = (0.5, 0.5, 0.5)

Regularization Constant = 0

Round of all the value to four decimal places

$$\theta^\top X = 0.5 + 0.5 \text{CGPA} + 0.5 \text{IQ}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$h_\theta(x) = \frac{1}{1 + e^{-(0.5 + 0.5 \text{CGPA} + 0.5 \text{IQ})}}$$

Approx. : New weights

$$\theta_0 = 0.35$$

$$\theta_{1=\text{CGPA}} = -0.42$$

$$\theta_{2=\text{IQ}} = -0.61$$

Repeat this for θ_1 and θ_2 as well

Logistic Regression – Inference & Interpretation

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

Assume if the converged parameters are:
 $(\theta_0, \theta_1, \theta_2) = 0.4, 0.3, -0.45$

Predict the Job offered for a candidate : (5, 6)

$$h_{\theta}(x) = \frac{1}{1 + e^{-(0.4 + 0.3 \text{ CGPA} - 0.45 \text{ IQ})}}$$

$$h(x) = 0.31$$

Y-Predicted = 0 / No

If $h_{\theta}(x) \geq 0.5$, predict "y = 1"

If $h_{\theta}(x) < 0.5$, predict "y = 0"

Bayesian Learning Parameter Estimation

- Where does the **cost** come from? - Logistic regression
- Why least-squares **cost** function, be a reasonable choice? – Linear regression

Distribution

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

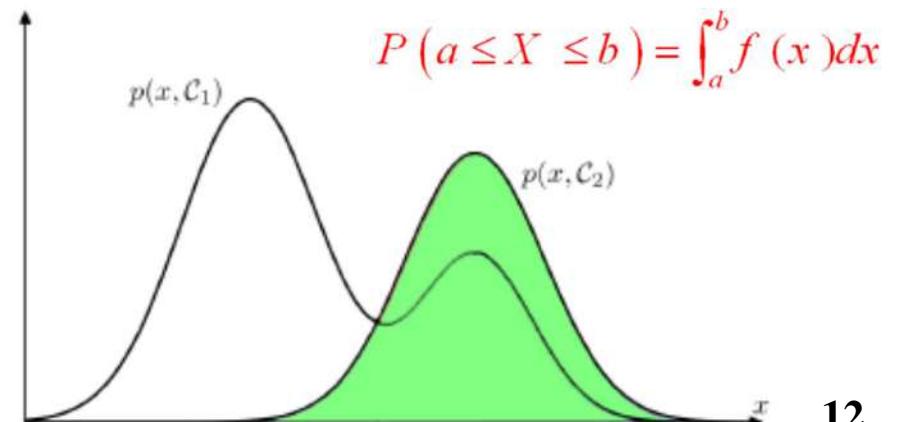
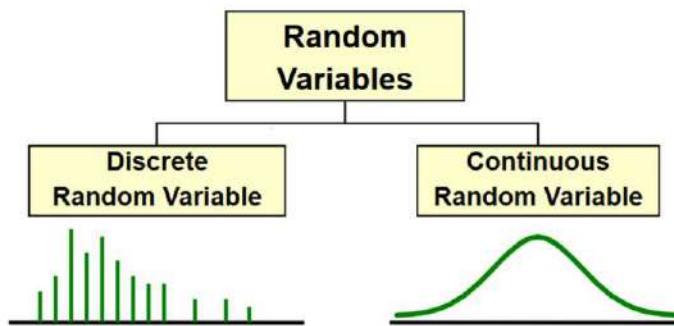
Mileage (in kmpl)	Car Price (in cr)
Neutral	High
Less	Low
Neutral	Medium
More	Low

Mileage (in kmpl)	Car Price (in cr)
9.8	High
9.12	Low
9.5	High
10	Low

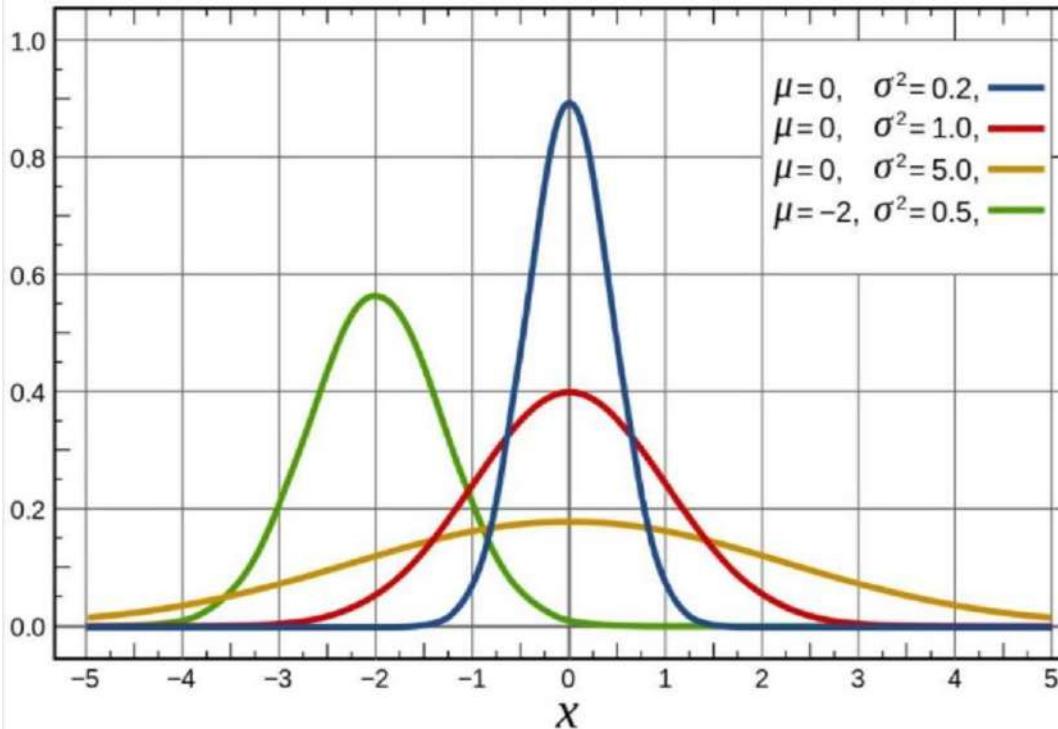
Represents a possible numerical value from a random event

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$



Parameter Estimation



Distribution	Parameters
Bernoulli(p)	$\theta = p$
Poisson(λ)	$\theta = \lambda$
Uniform(a, b)	$\theta = (a, b)$
Normal(μ, σ^2)	$\theta = (\mu, \sigma^2)$
$Y = mX + b$	$\theta = (m, b)$

Find $\theta = (\text{Mean}, \text{SD})$ from the data X_i
i.e., $(x_i, P(x_i))$

$$N(x | \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Parameter in ML

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

$$\text{CarPrice} = 8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2$$

Parameters : $(\theta_0, \theta_1, \theta_2)$

Mileage (in kmpl)	Car Price (in cr)
9.8	High
9.12	Low
9.5	High
10	Low

$$\text{CarPrice} = \frac{1}{1+e^{-8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2}}$$

Parameter Estimation in ML

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

$$\text{CarPrice} = 8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2$$

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

$$\epsilon^{(i)} \sim N(0, \sigma^2)$$

Parameters : $(\theta_0, \theta_1, \theta_2)$

Find $\theta = (\theta_0, \theta_1, \theta_2)$ from the data \mathbf{X}_i

i.e., $(\text{Mileage}_i, \text{CarPrice}_i)$

Assumption: Data are __

- IID samples: $X_1 \dots X_n$ where all X_i are independent and have the same distribution.
- Either same PMF (discrete) or same PDF (continuous)
- $f(X | \theta)$
Likelihood of different values of X depends on the values of our parameters θ

↓
 $f(\cdot)$ is either PDF or PMF

Maximum Likelihood Estimation (MLE)

select that parameters θ that make the observed data the most likely

$$f(X_1, X_2, \dots, X_n | \theta)$$

Maximum A Posteriori (MAP)

choose the parameters θ that is the most likely, given the data

$$f(\theta | X_1, X_2, \dots, X_n)$$

Intuition of Bayes Theorem

MAP : $f(\theta | X_1, X_2, \dots, X_n)$

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

MLE: $f(X_1, X_2, \dots, X_n | \theta)$

$P(h)$ = prior probability of hypothesis h

$P(D)$ = prior probability of training data D

$P(h|D)$ = probability of h given D

$P(D|h)$ = probability of D given h

After seeing
data, posterior
belief of θ

posterior

$$P(\theta | \text{data}) = \frac{P(\text{data}|\theta)P(\theta)}{P(\text{data})}$$

$L(\theta)$, probability of data
given parameter θ

likelihood prior

Before seeing data,
prior belief of θ
e.g. what is distribution over
parameters θ

MLE – Linear Regression Model

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

$$\text{CarPrice} = 8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2$$

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

$$\epsilon^{(i)} \sim N(0, \sigma^2)$$

Parameters : $(\theta_0, \theta_1, \theta_2)$

Find $\theta = (\theta_0, \theta_1, \theta_2)$ from the data \mathbf{X}_i

i.e., $(\text{Mileage}_i, \text{CarPrice}_i)$

Maximum Likelihood Estimation (MLE)

select that parameters θ that make the observed data the most likely

$$f(X_1, X_2, \dots, X_n | \theta)$$

Given the noise $\epsilon^{(i)}$ obeys a Normal distribution each $y^{(i)}$ must also obey a Normal distribution around the true target value

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right).$$

Parameters : $(\theta_0, \theta_1, \theta_2)$

Find $\theta = (\theta_0, \theta_1, \theta_2)$ from the data \mathbf{X}_i

i.e., $(\text{Mileage}_i, \text{CarPrice}_i)$

$$y^{(i)} | x^{(i)}; \theta \sim N(\theta^T x^{(i)}, \sigma^2)$$

$$p(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Maximum Likelihood Estimation (MLE)

1. Determine formula for $LL(\theta)$
2. Differentiate $LL(\theta)$ w.r.t. (each) θ
3. Solve

MLE answers the question: For which parameter value does the observed data have the biggest probability?

MLE – Linear Regression Model

$$p(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

1. Determine formula for $LL(\theta)$
2. Differentiate $LL(\theta)$ w.r.t. (each) θ

$$\begin{aligned} L(\theta) &= \prod_{i=1}^n p(y^{(i)} | x^{(i)}; \theta) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

$$\hat{\theta} = \operatorname{argmax} L(\theta) \quad LL(\theta) = \log L(\theta)$$

3. Solve

$$\begin{aligned} &= \log L(\theta) \\ &= \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2. \end{aligned}$$

MLE – Linear Regression Model

$$p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

1. Determine formula for $LL(\theta)$

$$\log L(\theta) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2$$

2. Differentiate $LL(\theta)$ w.r.t. (each) θ

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2$$

$$= -\frac{1}{2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2$$

3. Solve

$$\begin{aligned}\hat{\theta} &= \operatorname{argmax}_{\theta} L(\theta) \\ &= \operatorname{argmax} \left(-\frac{1}{2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2 \right) \\ &= \operatorname{argmin} \left(\frac{1}{2} \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2 \right)\end{aligned}$$

With probabilistic assumptions on the data, least-squares regression corresponds to finding the MLE of θ

MLE – Logistic Regression

$$y_i \mid x_i \sim \text{Bern}(\sigma(\mathbf{w}^\top \mathbf{x}_i))$$

1. Determine formula for $LL(\theta)$

$$LL(\theta) = \log \prod_{i=1}^n P_\theta(y^{(i)} | x^{(i)}) = \sum_{i=1}^n \log P_\theta(y^{(i)} | x^{(i)})$$

2. Differentiate $LL(\theta)$ w.r.t. (each) θ

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

3. Solve

Mileage (in kmpl)	Car Price (in cr)
9.8	High
9.12	Low
9.5	High
10	Low

Bernoulli MLE Estimation

X_1, X_2, \dots, X_n where $X_i \sim \text{Ber}(p)$. PMF of a Bernoulli

$$p^{X_i} (1-p)^{1-X_i}$$

$$P_\theta(Y = 1 | X = x) = h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$P_\theta(Y = 0 | X = x) = 1 - h_\theta(x) = \frac{e^{-\theta^\top x}}{1 + e^{-\theta^\top x}}$$

MLE answers the question: For which parameter value does the observed data have the biggest probability?

MLE – Logistic Regression

$$y_i \mid x_i \sim \text{Bern}(\sigma(\mathbf{w}^T \mathbf{x}_i))$$

1. Determine formula for $LL(\theta)$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^n P(Y = y^{(i)} | X = \mathbf{x}^{(i)}) \\ &= \prod_{i=1}^n \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} \cdot [1 - \sigma(\theta^T \mathbf{x}^{(i)})]^{(1-y^{(i)})} \end{aligned}$$

$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

2. Differentiate $LL(\theta)$ w.r.t. (each) θ

$$\begin{aligned} \frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})] \\ &= \left[\frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1 - y}{1 - \sigma(\theta^T \mathbf{x})} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T \mathbf{x}) \\ &= \left[\frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1 - y}{1 - \sigma(\theta^T \mathbf{x})} \right] \sigma(\theta^T \mathbf{x}) [1 - \sigma(\theta^T \mathbf{x})] \mathbf{x}_j \\ &= \left[\frac{y - \sigma(\theta^T \mathbf{x})}{\sigma(\theta^T \mathbf{x}) [1 - \sigma(\theta^T \mathbf{x})]} \right] \sigma(\theta^T \mathbf{x}) [1 - \sigma(\theta^T \mathbf{x})] \mathbf{x}_j \\ &= [y - \sigma(\theta^T \mathbf{x})] \mathbf{x}_j \end{aligned}$$

3. Solve

$$\theta := \theta + \alpha \nabla_{\theta} \ell(\theta)$$

MLE answers the question: For which parameter value does the observed data have the biggest probability?

MLE – Discrete - PMF

Example 1: Suppose that X is a discrete random variable with the following probability mass function: where $0 \leq \theta \leq 1$ is a parameter. The following 10 independent observations

X	0	1	2	3
$P(X)$	$2\theta/3$	$\theta/3$	$2(1-\theta)/3$	$(1-\theta)/3$

were taken from such a distribution: (3,0,2,1,3,2,1,0,2,1). What is the maximum likelihood estimate of θ .

1. Determine formula for $LL(\theta)$
2. Differentiate $LL(\theta)$ w.r.t. (each) θ
3. Solve

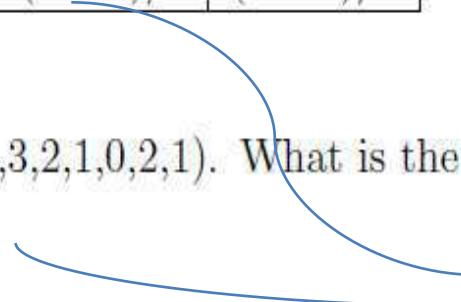
MLE – Discrete - PMF

Example 1: Suppose that X is a discrete random variable with the following probability mass function: where $0 \leq \theta \leq 1$ is a parameter. The following 10 independent observations

X	0	1	2	3
$P(X)$	$2\theta/3$	$\theta/3$	$2(1-\theta)/3$	$(1-\theta)/3$

were taken from such a distribution: $(3,0,2,1,3,2,1,0,2,1)$. What is the estimate of θ .

1. Determine formula for $LL(\theta)$
2. Differentiate $LL(\theta)$ w.r.t. (each) θ
3. Solve



X	$P(X)$
3	$(1-\Theta)/3$
0	$2\Theta/3$
2	$2(1-\Theta)/3$
1	$\Theta/3$
3	$(1-\Theta)/3$
2	$2(1-\Theta)/3$
1	$\Theta/3$
0	$2\Theta/3$
2	$2(1-\Theta)/3$
1	$\Theta/3$

MLE – Discrete - PMF

- Determine formula for $LL(\theta)$

$$\begin{aligned} L(\theta) &= P(X=3)*P(X=0)*P(X=2)*\dots*P(X=1) \\ &= ((1-\theta)/3)^2 * (2\theta/3)^2 * (2(1-\theta)/3)^3 * (\theta/3)^3 \end{aligned}$$

- Differentiate $LL(\theta)$ w.r.t. (each) θ

- Solve

X	P(X)
3	$(1-\theta)/3$
0	$2\theta/3$
2	$2(1-\theta)/3$
1	$\theta/3$
3	$(1-\theta)/3$
2	$2(1-\theta)/3$
1	$\theta/3$
0	$2\theta/3$
2	$2(1-\theta)/3$
1	$\theta/3$

MLE – Discrete - PMF

1. Determine formula for $LL(\theta)$

$$\begin{aligned} L(\theta) &= P(X=3)*P(X=0)*P(X=2)*\dots*P(X=1) \\ &= ((1-\theta)/3)^2 * (2\theta/3)^2 * (2(1-\theta)/3)^3 * (\theta/3)^3 \end{aligned}$$

2. Differentiate $LL(\theta)$ w.r.t. (each) θ

$$\begin{aligned} LL(\theta) &= \log[((1-\theta)/3)^2 * (2\theta/3)^2 * (2(1-\theta)/3)^3 * (\theta/3)^3] \\ &= \log((1-\theta)/3)^2 + \log(2\theta/3)^2 + \log(2(1-\theta)/3)^3 * \log(\theta/3)^3 \\ &= 2\log((1-\theta)/3) + 2\log(2\theta/3) + 3\log(2(1-\theta)/3) + 3\log(\theta/3) \\ &= 2(\log((1-\theta) - \log 3) + 2(\log(2\theta) - \log 3) + 3(\log(2(1-\theta)) - \log 3) + 3(\log(\theta) - \log 3) \end{aligned}$$

3. Solve

$$\begin{aligned} \text{Gradient (} LL(\theta) \text{)} &= -\frac{2}{(1-\theta)} + 2 - \frac{3}{(1-\theta)} + \frac{3}{\theta} \\ &= \frac{-5\theta + 2\theta - 2\theta^2 + 3 - 3\theta}{(1-\theta)\theta} = \frac{-6\theta - 2\theta^2 + 3}{(1-\theta)\theta} = 0 \end{aligned}$$

Solve & get the value of θ

MLE-Summary

- Consider a sample of n i.i.d. random variables X_1, X_2, \dots, X_n , drawn from a distribution $f(X_i|\theta)$
- θ_{MLE} maximizes the likelihood of data, $L(\theta)$ and $LL(\theta)$

$$L(\theta) = \prod_{i=1}^n f(X_i|\theta) \quad \hat{\theta} = \operatorname{argmax}_{\theta} L(\theta)$$

$$LL(\theta) = \log L(\theta) = \log \prod_{i=1}^n f(X_i|\theta) = \sum_{i=1}^n \log f(X_i|\theta)$$

$$\theta_{MLE} = \arg \max_{\theta} LL(\theta)$$

Maximum A Posteriori (MAP) Analysis

1. Determine prior probability

$$\theta_{MAP} = \arg \max f(\theta | X_1, X_2, \dots, X_n)$$

2. Find the posterior probability for every distinct prior

Brute Force MAP Hypothesis

3. Choose the posterior with highest h_{MAP} value

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D)$$

$$= \operatorname{argmax}_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \operatorname{argmax}_{h \in H} P(D|h)P(h)$$

Maximum a Posteriori (MAP) Estimator of θ is the value of θ that maximizes the posterior distribution of θ .

Best hypothesis \approx most probable hypothesis

Maximum A Posteriori Estimation (MAP)

-
1. Find the prior probability
 2. Derive the posterior probability
 3. Differentiate posterior w.r.t. (each) θ
 4. Solve

Maximum a Posteriori (MAP) Estimator of θ is the value of θ that maximizes the posterior distribution of θ .

Best hypothesis \approx most probable hypothesis

Example :

1. Example on MAP algorithm:

Let X be continuous random variable with probability density function $P(X)$ given by:

$$f(x) = \begin{cases} 2x, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

Given another distribution $p(Y|X = x) = x(1 - x)^{y-1}$ Find MAP estimate of X given $Y=3$

$$\begin{aligned} h_{\text{MAP}} &= P(X | Y=3) \\ &= P(Y=3 | X) * P(X) \\ &= x(1-x)^{y-1} * 2x \end{aligned}$$

To find the parameter X , differentiate the function & equate to zero.

$$\frac{d(P(X|Y=3))}{dx} = 0 \quad \frac{d(2x^2 - 4x^3 + 2x^4)}{dx} = 0 \quad x = \{0, 0.5, 1\}$$

$$\frac{d(x(1-x)2 * 2x)}{dx} = 0 \quad 4x - 12x^2 + 8x^3 = 0 \quad P(X|Y=3) = \{0, 0.125, 0\}$$

Most Probable Classification of New Instances

- So far we've sought the most probable hypothesis given the data D (i.e., h_{MAP})
- **Given new instance x, what is its most probable classification?**
 - $h_{MAP}(x)$ is not the most probable classification!
 - What's most probable classification of x?

Consider:

Three possible hypotheses:

$$P(h_1|D) = .4, P(h_2|D) = .3, P(h_3|D) = .3$$

Given new instance x, classification given by above 3 hypotheses is

$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$

$$P(\oplus|h_1) = 1 \quad \text{and} \quad P(\ominus|h_1) = 0$$

May be the classification for X is +

Bayes' Optimal Classifier

- The most probable classification of the new instance is obtained by combining **the predictions of all hypotheses, weighted by their posterior probabilities.**
- v_j from some set V , then the probability $P(v_j | D)$ that the correct classification for the new instance is v_j is:

$$P(v_j | D) = \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

- The optimal classification of the new instance is the value v_j for which $P(v_j | D)$ is maximum

Example 1 : Bayes Optimal Classifier

Bayes optimal classification:

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Example:

$$P(h_1|D) = .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1$$

$$P(h_2|D) = .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0$$

$$P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
- Gibbs algorithm:
 - Choose one hypothesis at random, according to posterior prob. Distribution over h , $P(h|D)$
 - Use this h to classify new instance
- Surprising fact: under certain conditions, the expected misclassification error for the Gibbs algorithm is at most twice the expected error of the Bayes optimal classifier

$$E[\text{error}_{\text{Gibbs}}] \leq 2E[\text{error}_{\text{BayesOptional}}]$$

- Suppose correct, uniform prior distribution over H , then
 - Pick any hypothesis from *Version space*, with uniform probability
 - Its expected error no worse than twice Bayes optimal

Some Additional References

- T1 book by Tom Mitchell – CH-6
- <https://web.stanford.edu/class/archive/cs/cs109>
- https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf
- <https://www.cs.cmu.edu/~ninanmf/courses/601sp15/lectures.shtml>

Self Learning : Logistic Regression Bayesian Analysis

Where does the **hypothesis function** come from?



- Logistic regression hypothesis representation

$$P(Y=1|X) = h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n)}}$$

- Model **likelihood** $P(X_i|Y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$ and assume variance is independent of class, i.e. $\sigma_{i0} = \sigma_{i1} = \sigma_i$

$$P(x|y_k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}$$

- Model **prior** $P(Y)$ as Bernoulli π : $P(Y=1) = \pi$ and $P(Y=0) = 1-\pi$

What is $P(Y|X_1, X_2, \dots, X_n)$?

Logistic Regression – Bayesian Analysis

$$P(Y = 1|X) = \frac{P(Y = 1)P(X|Y = 1)}{P(Y = 1)P(X|Y = 1) + P(Y = 0)P(X|Y = 0)}$$

Applying Bayes rule

$$P(Y = 1|X) = \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

Divide by $P(Y = 1)P(X|Y = 1)$

$$P(Y = 1|X) = \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

Apply $\exp(\ln(\cdot))$

$$P(Y = 1|X) = \frac{1}{1 + \exp (\ln \frac{P(Y = 0)}{P(Y = 1)} + \ln \frac{P(X|Y = 0)}{P(X|Y = 1)})}$$

Logistic Regression – Bayesian Analysis

By independence assumption:

$P(Y=1)=\pi$ and $P(Y=0)=1-\pi$
by modelling $P(Y)$ as Bernoulli

$$\frac{P(X|Y=0)}{P(X|Y=1)} = \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \ln \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

Logistic Regression - Bayesian Analysis



Plug in $P(X_i|Y)$

$$P(Y=1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right))}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

$$w_0 = \ln \frac{1-\pi}{\pi} + \sum_i \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}$$

$$w_i = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2}$$

$$\begin{aligned} \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i - \mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i - \mu_{i1})^2}{2\sigma_i^2}\right)} \\ &= \sum_i \ln \exp\left(\frac{(X_i - \mu_{i1})^2 - (X_i - \mu_{i0})^2}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{(X_i - \mu_{i1})^2 - (X_i - \mu_{i0})^2}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{(X_i^2 - 2X_i\mu_{i1} + \mu_{i1}^2) - (X_i^2 - 2X_i\mu_{i0} + \mu_{i0}^2)}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{2X_i(\mu_{i0} - \mu_{i1}) + \mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) \end{aligned}$$

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

& Refresh your MFDS previous semester course basics

Next Session Plan :

Remaining Topics – Naïve Bayes Classification



BITS Pilani
Pilani Campus

Machine Learning

DSE CLZG565

Bayesian Learning & Bayesian Classifiers

Raja vadhana P

Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 3 & 4

- MLE Hypothesis
- MAP Hypothesis
- Bayes optimal classifier
- Gibbs Algorithm
- Naïve Bayes Classifier

Learnt so far in previous session

- Few Terminologies /Concepts
 - Likelihood
 - Log Likelihood
 - Distribution
 - Prior / Posterior Probability
 - Parameter Estimation
 - Maximum Likelihood Estimation
 - Maximum A Posterior Estimation
 - Notion of Linear Regression & Logistic Regression Cost function Design
 - Optimal Classifiers
 - Bayesian Learning
 - Bayesian Optimal Classifier
 - Gibbs classifier

Agenda

- Naïve Bayes Classifier
- Gaussian Naïve Bayes Classifier
- Text classification model

Intuition of Bayes Theorem

MAP : $f(\theta | X_1, X_2, \dots, X_n)$

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

MLE: $f(X_1, X_2, \dots, X_n | \theta)$

$P(h)$ = prior probability of hypothesis h

$P(D)$ = prior probability of training data D

$P(h|D)$ = probability of h given D

$P(D|h)$ = probability of D given h

After seeing
data, posterior
belief of θ

posterior

$$P(\theta | \text{data}) = \frac{\text{likelihood} \quad \text{prior}}{P(\text{data}|\theta)P(\theta)} = \frac{L(\theta), \text{ probability of data given parameter } \theta}{P(\text{data})}$$

$L(\theta)$, probability of data
given parameter θ

Before seeing data,
prior belief of θ
e.g. what is distribution over
parameters θ

Example 1 : Bayes Optimal Classifier

Bayes optimal classification:

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Example:

$$P(h_1|D) = .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1$$

$$P(h_2|D) = .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0$$

$$P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

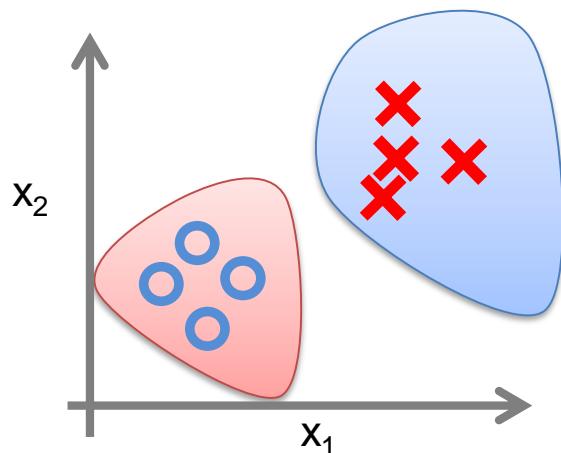
Types of Classification



Decision Theory: Interpretation

Model Building

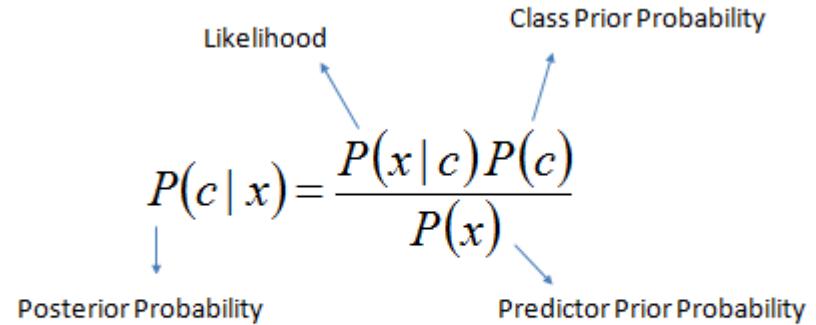
Generative



$$P(Y | X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d | Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

Known as generative models, because by sampling from them it is possible to generate synthetic data points in the input space.

Eg., Gaussians, **Naïve Bayes**, Mixtures of multinomials, **Mixtures of Gaussians**, Bayesian networks



$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Generative models for classification

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}.$$

- For binary classification the denominator is given by

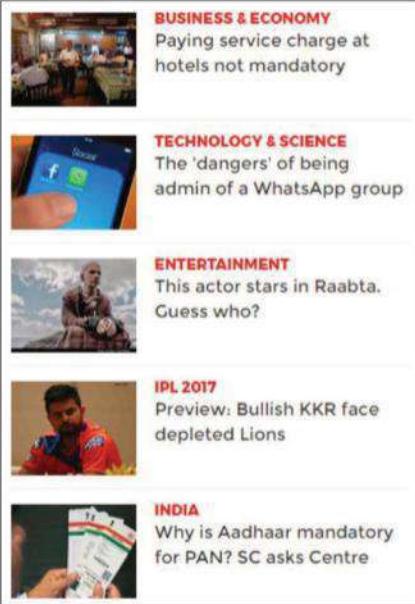
$$p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$$

- if we're calculating $p(y|x)$ in order to make a prediction, then we don't actually need to calculate the denominator, since

$$\begin{aligned}\arg \max_y p(y|x) &= \arg \max_y \frac{p(x|y)p(y)}{p(x)} \\ &= \arg \max_y p(x|y)p(y).\end{aligned}$$

Naïve Bayes Classifier - Applications

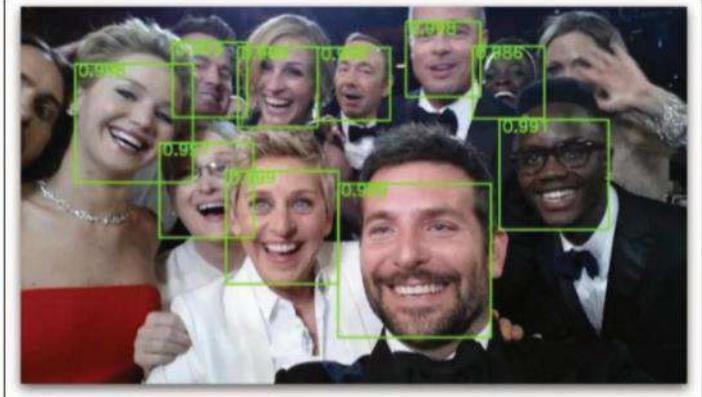
Categorizing News



Email Spam Detection



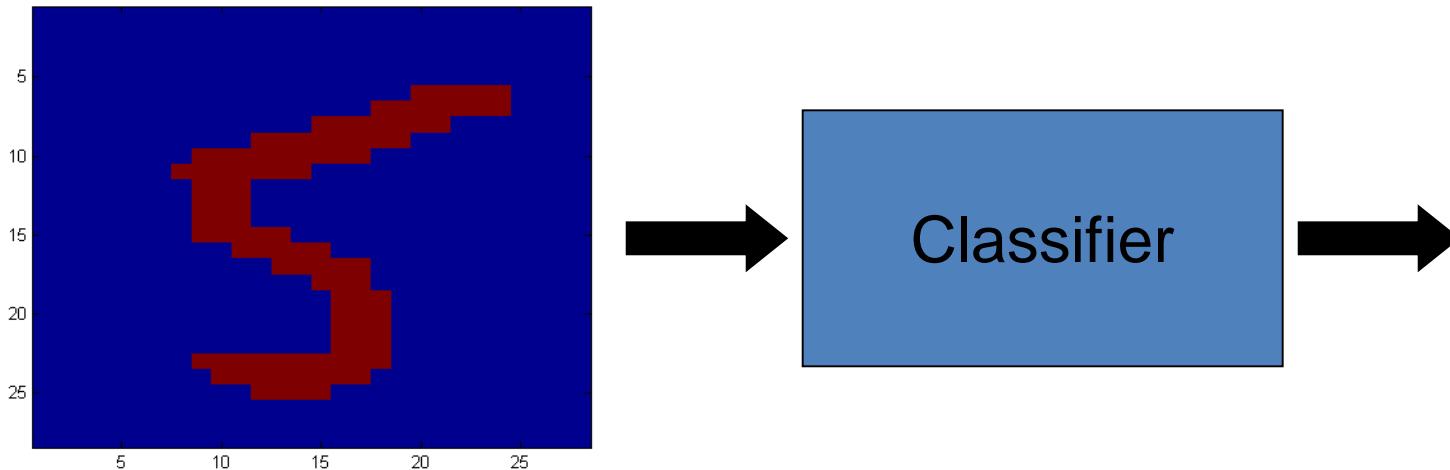
Face Recognition



Sentiment Analysis



Example: Digit Recognition



- $X_1, \dots, X_n \in \{0,1\}$ (Black vs. White pixels)
- $Y \in \{5,6\}$ (predict whether a digit is a 5 or a 6)

The Bayes Classifier

$$P(Y = 5|X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n|Y = 5)P(Y = 5)}{P(X_1, \dots, X_n|Y = 5)P(Y = 5) + P(X_1, \dots, X_n|Y = 6)P(Y = 6)}$$
$$P(Y = 6|X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n|Y = 6)P(Y = 6)}{P(X_1, \dots, X_n|Y = 5)P(Y = 5) + P(X_1, \dots, X_n|Y = 6)P(Y = 6)}$$

- To classify, we'll simply compute these two probabilities and predict based on which one is greater

Naïve Bayes conditional Independence assumption

- Naïve Bayes assumes X_i are conditionally independent given Y

$$P(X_1|X_2, Y) = P(X_1|Y)$$

- **Assumption:**

$$P(X_1, \dots, X_n|Y) = \prod_{j=1}^n P(X_j|Y)$$

i.e., X_i and X_j are conditionally independent given Y for $i \neq j$

Naïve Bayes classifier: Prediction

Goal of learning $P(Y|X)$ where $X = \langle X_1, \dots, X_n \rangle$

- Bayes rule:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)P(X_1, \dots, X_n | Y = y_k)}{\sum_j P(Y = y_j)P(X_1, \dots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i 's:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)\prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i | Y = y_j)}$$

- Classify New Instance(x) : Pick the most probable (MAP) Y for

$$\hat{Y} \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k)\prod_i P(X_i | Y = y_k)$$

$X_{new} = \langle X_1, \dots, X_n \rangle$

↑
Prior Likelihood

Example: Play Tennis

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood Class Prior Probability
 ↓ ↓
 Posterior Probability Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

$$\hat{Y} \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k) \prod_i P(X_i|Y = y_k)$$

$P(X|Y) \sim \text{Multinom}(\pi, n) \rightarrow \text{Multinomial NB } (X_i - \text{multinomial})$

$P(Y) \sim \text{Ber}(p)$

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Example: Play Tennis – Learning Phase

Look up tables

Sky	Play=Yes	Play=No
Sunny	3/3	2/4
Rainy	0/3	2/4

AirTemp	Play=Yes	Play=No
Hot	0/3	1/4
Warm	3/3	1/4
Cold	0/3	2/4

Humidity	Play=Yes	Play=No
High	1/3	3/4
Normal	2/3	1/4

Maximum likelihood estimates (MLE's):

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

Number of items in dataset D for which $Y=y_k$

Wind	Play=Yes	Play=No
Strong	2/3	3/4
Breeze	1/3	1/4

Forecast	Play=Yes	Play=No
Same	2/3	2/4
Change	1/3	2/4

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Example: Play Tennis - Testing Phase

$$\begin{aligned}
 P(\text{Enjoy}=\text{Yes} | X) &= P(X | \text{Enjoy}=\text{Yes}) \cdot P(\text{Enjoy}=\text{Yes}) / P(X) \\
 &= P(X | \text{Enjoy}=\text{Yes}) \cdot P(\text{Enjoy}=\text{Yes}) \\
 &= P(X | \text{Enjoy}=\text{Yes}) \cdot (3/7) \\
 &= P(\text{Sunny} | \text{Enjoy}=\text{Yes}) \cdot P(\text{Warm} | \text{Enjoy}=\text{Yes}) \cdot P(\text{Normal} | \text{Enjoy}=\text{Yes}) \cdot P(\text{Strong} | \text{Enjoy}=\text{Yes}) \\
 &\quad P(\text{Change} | \text{Enjoy}=\text{Yes}) \cdot (3/7) \\
 &= (3/3) \cdot (3/3) \cdot (2/3) \cdot (2/3) \cdot (1/3) \cdot (3/7) \\
 &= 0.0635
 \end{aligned}$$

$P(\text{Enjoy}=\text{Yes} | X) > P(\text{Enjoy}=\text{No} | X) \rightarrow \text{EnjoySport} = \text{Yes}$

$$\begin{aligned}
 P(\text{Enjoy}=\text{No} | X) &= P(X | \text{Enjoy}=\text{No}) \cdot P(\text{Enjoy}=\text{No}) / P(X) \\
 &= P(X | \text{Enjoy}=\text{No}) \cdot P(\text{Enjoy}=\text{No}) \\
 &= P(X | \text{Enjoy}=\text{No}) \cdot (4/7) \\
 &= P(\text{Sunny} | \text{Enjoy}=\text{No}) \cdot P(\text{Warm} | \text{Enjoy}=\text{No}) \cdot P(\text{Normal} | \text{Enjoy}=\text{No}) \cdot P(\text{Strong} | \text{Enjoy}=\text{No}) \cdot P(\text{Change} | \text{Enjoy}=\text{No}) \cdot (4/7) \\
 &= (2/4) \cdot (1/4) \cdot (1/4) \cdot (3/4) \cdot (2/4) \cdot (4/7) \\
 &= 0.006696
 \end{aligned}$$

MAP rule

$$\begin{aligned}
 Y^{new} &\leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k) \\
 Y^{new} &\leftarrow \arg \max_{y_k} \pi_k \prod_i \theta_{ijk}
 \end{aligned}$$

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Sunny	Warm	Normal	Strong	Change	????
Rainy	Warm	Normal	Breeze	Same	????

Naïve Bayes Algorithm – discrete X_i

- Train Naïve Bayes (examples) for each* value y_k
 estimate $\pi_k \equiv P(Y = y_k)$
 for each* value x_{ij} of each attribute X_i
 estimate $\theta_{ijk} \equiv P(X_i = x_{ij}|Y = y_k)$
- Classify (X^{new})

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new}|Y = y_k)$$

$$Y^{new} \leftarrow \arg \max_{y_k} \pi_k \prod_i \theta_{ijk}$$



Laplace Smoothing

Smoothing

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

- Technique for smoothing categorical data.
- A small-sample correction, or **pseudo-count**, will be incorporated in every probability estimate.
- No probability will be zero.

Smoothing



Probability estimation:

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

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

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



c: number of classes

N_c : number of instances in the class

N_{ic} : number of instances having attribute value A_i in class c

p: prior probability of the class

m: constant called the **equivalent sample size**, which determines how heavily to weight p relative to the observed data

Bayesian approach

Example: Play Tennis

$$\begin{aligned}
 P(Enjoy=Yes | X) &= P(X | Enjoy=Yes). P(Enjoy=Yes) / P(X) \\
 &= P(X | Enjoy=Yes). P(Enjoy=Yes) \\
 &= P(X | Enjoy=Yes). (3/7) \\
 &= P(Rainy | Enjoy=Yes). P(Warm | Enjoy=Yes). P(Normal | Enjoy=Yes). P(Breeze | Enjoy=Yes). \\
 P(Same | Enjoy=Yes).(3/7) \\
 &= (1/5) . (3/3) . (2/3) . (1/3) . (2/3) . (3/7) \\
 &= 0.0127
 \end{aligned}$$

$P(Enjoy=Yes | X) > P(Enjoy=No | X) \rightarrow EnjoySport = Yes$

$$\begin{aligned}
 P(Enjoy=No | X) &= P(X | Enjoy=No). P(Enjoy=No) / P(X) \\
 &= P(X | Enjoy=No). P(Enjoy=No) \\
 &= P(X | Enjoy=No). (4/7) \\
 &= P(Rainy | Enjoy=No). P(Warm | Enjoy=No). P(Normal | Enjoy=No). P(Breeze | Enjoy=No). P(Same | \\
 Enjoy=No). (4/7) \\
 &= (3/6) . (1/4) . (1/4) . (1/4) . (2/4) . (4/7) \\
 &= 0.0023
 \end{aligned}$$

Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Sunny	Warm	Normal	Strong	Change	????
Rainy	Warm	Normal	Breeze	Same	????

Naïve Bayes: Continuous Features

- X_i can be continuous

Naïve Bayes classifier:

$$Y = \arg \max_y P(Y = y) \prod_i P(X_i | Y = y)$$

Assumption: $P(X_i | Y)$ has a **Gaussian** distribution

The Gaussian Probability Distribution

- It is a continuous distribution with pdf:

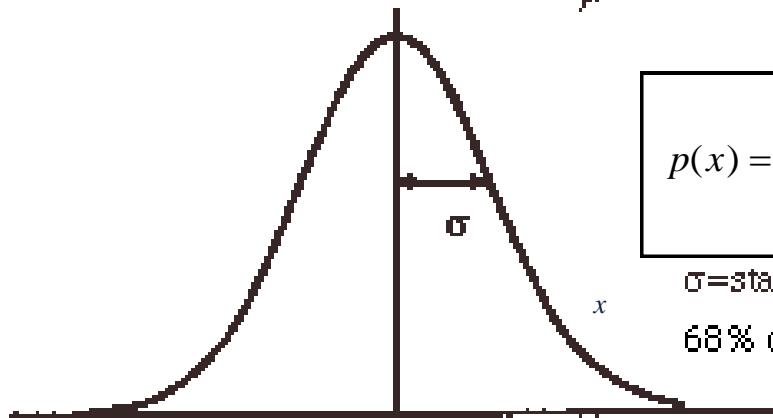
μ = mean of distribution

σ^2 = variance of distribution

x is a continuous variable ($-\infty \leq x \leq \infty$)

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

mode=median=mean = μ .



$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \text{ gaussian}$$

σ =standard deviation

68% of area within $\pm 1\sigma$

Continuous Features : learning and prediction

- For each target value Y_k (MLE estimate)

$P(Y = y_k) \leftarrow \text{No. of instances with } Y_k \text{ class}/\text{No. of Total instances}$

- For each attribute value X_i estimate $P(X_i|Y = y_k)$
 - class conditional mean , variance

- Classify New Instance(x)

Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \operatorname{argmax}_{y_k} P(Y = y_k) \prod_i P(X_i|Y = y_k)$$

Continuous Features : learning

- $P(X_i|Y)$ is Gaussian
- Training: estimate mean and standard deviation
 - $\mu_i = E[X_i|Y = y]$
 - $\sigma_i^2 = E[(X_i - \mu_i)^2|Y = y]$

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	0.4	1
1.2	0.3	0	0
2.2	1.1	0	1

Continuous Features : learning

- $\mu_i = E[X_i | Y = y]$
- $\sigma_i^2 = E[(X_i - \mu_i)^2 | Y = y]$

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	0.4	1
1.2	0.3	0	0
2.2	1.1	0	1

- $\mu_1 = E[X_1 | Y = 1] = \frac{2 + (-1.2) + 2.2}{3} = 1$
- $\sigma_1^2 = E[(X_1 - \mu_1)^2 | Y = 1] = \frac{(2 - 1)^2 + (-1.2 - 1)^2 + (2.2 - 1)^2}{3} = 2.43$

Example: Evade Tax

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
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$
 $P(\text{Refund} = \text{No} | \text{No}) = 4/6$
 $\rightarrow P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$
 $\rightarrow P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$
 For Taxable Income:
 If class = No: sample mean = 91
 sample variance = 685
 If class = Yes: sample mean = 90
 sample variance = 25

Given $X = (\text{Refund} = \text{Yes}, \text{Divorced}, 120\text{K})$

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

Naïve Bayes will not be able to classify X as Yes or No!

Example: Play Tennis

$P(X|Y) \sim N(\mu, \sigma^2) \rightarrow \text{GaussianNB } (X_i - \text{real valued})$

$$\begin{aligned}
 P(Enjoy=\text{Yes} | X) &= P(X | Enjoy=\text{Yes}). P(Enjoy=\text{Yes}) / P(X) \\
 &= P(X | Enjoy=\text{Yes}). P(Enjoy=\text{Yes}) \\
 &= P(X | Enjoy=\text{Yes}). (3/7) \\
 &= P(Rainy | Enjoy=\text{Yes}). P(Warm | Enjoy=\text{Yes}). P(60 | Enjoy=\text{Yes}). P(Breeze | Enjoy=\text{Yes}). P(Same | Enjoy=\text{Yes}). (3/7) \\
 &= (1/3) . (3/3) . 0.15 \cdot 10^{-95} . (1/3) . (2/3) . (3/7)
 \end{aligned}$$

$$\mu_i = E[X_i | Y = \text{yes}] = 84.33$$

$$\sigma_i^2 = E[(X_i - \mu_i)^2 | Y = \text{yes}] = 1.15$$

$$\mu_i = E[X_i | Y = \text{no}] = 72.5$$

$$\sigma_i^2 = E[(X_i - \mu_i)^2 | Y = \text{no}] = 17.08$$

$$\begin{aligned}
 P(Enjoy=\text{No} | X) &= P(X | Enjoy=\text{No}). P(Enjoy=\text{No}) / P(X) \\
 &= P(X | Enjoy=\text{No}). P(Enjoy=\text{No}) \\
 &= P(X | Enjoy=\text{No}). (4/7) \\
 &= P(Rainy | Enjoy=\text{No}). P(Warm | Enjoy=\text{No}). P(60 | Enjoy=\text{No}). P(Breeze | Enjoy=\text{No}). P(Same | Enjoy=\text{No}). (4/7) \\
 &= (2/4) . (1/4) . 0.02 . (1/4) . (2/4) . (4/7)
 \end{aligned}$$

Humidity	Enjoy Sport?	AirTemp	Sky	Wind	Forecast	Enjoy Sport?
85	Yes	Warm	Sunny	Strong	Same	Yes
80	No	Warm	Sunny	Strong	Same	No
70	No	Cold	Rainy	Strong	Change	No
83	Yes	Warm	Rainy	Breeze	Same	Yes
90	No	Hot	Sunny	Breeze	Same	No
50	No	Cold	Rainy	Strong	Change	No
85	Yes	Warm	Sunny	Strong	Change	Yes
60	????	Warm	Rainy	Breeze	Same	????



Text Classification using Naive Bayes Classifier

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsic	1
al	
times	1
sweet	1
satirical	1
adventur	1
e	
genre	1
fairy	1
humor	1
have	1
great	1

Example : Multinomial model :

Which Tag sentence “ A very close game” belong to?

$$P(\text{Sports}=\text{Yes}) = 3/5$$

$$P(\text{Sports}=\text{No}) = 2/5$$

$$P(A | \text{Sports}=\text{Yes}) = 2/11$$

$$P(A | \text{Sports}=\text{No}) = 1/9$$

$$\begin{aligned}
 P(\text{Sports}=\text{Yes} | X) &= P(X | \text{Sports}=\text{Yes}). P(\text{Sports}=\text{Yes}) / P(X) \\
 &= P(X | \text{Sports}=\text{Yes}). (3/5) \\
 &= P(A | \text{Sports}=\text{Yes}). P(\text{Very} | \text{Sports}=\text{Yes}). P(\text{Close} | \text{Sports}=\text{Yes}). P(\text{Game} | \text{Sports}=\text{Yes}). (3/5) \\
 &= (2/11) . (1/11) . (0/11) . (2/11) . (3/5)
 \end{aligned}$$

Text	Tag
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

A	Great	Game	The	Election	Was	Over	Very	Clean	Match	But	Forgettable	It	Close	Sports or Not Sports
1	1	1												1
			1	1	1	1								0
							1	1	1					1
1		1						1		1	1			1
1				1	1							1	1	0
1		1					1						1	????

Laplace Smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].
- In our example
- we add 1 to every probability, therefore the probability, such as **P(close | sports)**, will never be 0.

Example : Multinomial model :

Which Tag sentence “ A very close game” belong to?

$$P(\text{Sports}=\text{Yes}) = 3/5$$

$$P(\text{Sports}=\text{No}) = 2/5$$

$$P(A | \text{Sports}=\text{Yes}) = 2/11$$

$$P(A | \text{Sports}=\text{No}) = 1/9$$

$$P(\text{Sports}=\text{Yes} | X) = P(X | \text{Sports}=\text{Yes}). P(\text{Sports}=\text{Yes}) / P(X)$$

$$= P(X | \text{Sports}=\text{Yes}). (3/5)$$

$$= P(A | \text{Sports}=\text{Yes}). P(\text{Very} | \text{Sports}=\text{Yes}). P(\text{Close} | \text{Sports}=\text{Yes}). P(\text{Game} | \text{Sports}=\text{Yes}). (3/5)$$

$$= (2/11). (1/11). (0/11). (2/11). (3/5)$$

$$= (2+1/11+14). (1+1/11+14). (0+1/11+14). (2+1/11+14). (3/5)$$

$$= 0.00002765$$

Text	Tag
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

A	Great	Game	The	Election	Was	Over	Very	Clean	Match	But	Forgettable	It	Close	Sports or Not Sports
1	1	1												1
			1	1	1	1								0
							1	1	1					1
1		1						1		1	1			1
1				1	1							1	1	0
1		1					1						1	????

Apply Laplace Smoothing

Word	P(word Sports)	P(word Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$\begin{aligned}
 & P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times \\
 & P(Sports) \\
 & = 2.76 \times 10^{-5} \\
 & = 0.0000276
 \end{aligned}$$

$$\begin{aligned}
 & P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times \\
 & P(game|Not\ Sports) \times P(Not\ Sports) \\
 & = 0.572 \times 10^{-5} \\
 & = 0.00000572
 \end{aligned}$$

Example 2: Multinomial model

	docID	words in document	in $c = China?$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(w_i \mid C_k) = \frac{n_k(w_i)}{\sum_{x=1}^{|V|} n_k(w_x)},$$

N_{yes} (W=Chinese) = 5, N_{No} (W=Chinese) = 1,

$|V| = 6 = \{\text{Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}\}$

No of features (words) in Yes class = 8

No of features (words) in No class = 3

$P(X|C)$ = Fraction of tokens(positions) in documents of class c that contain term X

Example 2

	docID	words in document	in $c = \text{China?}$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\bar{c}) = 1/4$

$$\hat{P}(w_i | C_k) = \frac{n_k(w_i)}{\sum_{x=1}^{|V|} n_k(w_x)},$$

$$\hat{P}(\text{CHINESE}|c) = (5+1)/(8+6) = 6/14 = 3/7$$

$$\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) = (0+1)/(8+6) = 1/14$$

$$\hat{P}(\text{CHINESE}|\bar{c}) = (1+1)/(3+6) = 2/9$$

$$\hat{P}(\text{TOKYO}|\bar{c}) = \hat{P}(\text{JAPAN}|\bar{c}) = (1+1)/(3+6) = 2/9$$

Example 2

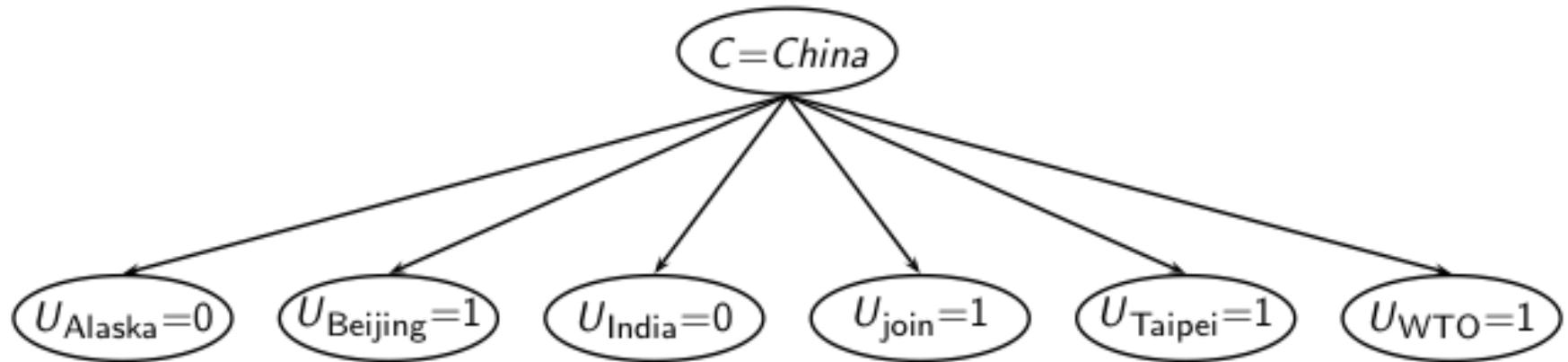
	docID	words in document	in $c = China?$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$P(C_k | \mathcal{D}) \propto P(C_k) \prod_{j=1}^{\text{len}(\mathcal{D})} P(u_j | C_k) \quad u\text{-each word in test document}$$

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$

$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Different Naive Bayes model: Bernoulli model



One feature X_w for each word in dictionary

X_w = true in document d if w appears in d

Example 3

	docID	words in document	in $c = \text{China?}$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(w_t | C_k) = \frac{n_k(w_t)}{N_k},$$

Let $n_k(w_t)$ be the number of documents of class k in which w_t is observed; and let N_k be the total number of documents of that class.

$$N_{\text{yes}} (\text{W=Chinese}) = 3, N_{\text{No}} (\text{W=Chinese}) = 1,$$

$$\text{No of features (documents) in Yes class} - (N_{\text{Yes}}) = 3$$

$$\text{No of features (documents) in No class} - (N_{\text{No}}) = 1$$

$$|v| = 6$$

$P(X|C)$ = Fraction of documents of class c that contain term X

Example 3

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(\text{Chinese}|c) = (3+1)/(3+2) = 4/5$$

$$\hat{P}(\text{Japan}|c) = \hat{P}(\text{Tokyo}|c) = (0+1)/(3+2) = 1/5$$

$$\hat{P}(\text{Beijing}|c) = \hat{P}(\text{Macao}|c) = \hat{P}(\text{Shanghai}|c) = (1+1)/(3+2) = 2/5$$

$$\hat{P}(\text{Chinese}|\bar{c}) = (1+1)/(1+2) = 2/3$$

$$\hat{P}(\text{Japan}|\bar{c}) = \hat{P}(\text{Tokyo}|\bar{c}) = (1+1)/(1+2) = 2/3$$

$$\hat{P}(\text{Beijing}|\bar{c}) = \hat{P}(\text{Macao}|\bar{c}) = \hat{P}(\text{Shanghai}|\bar{c}) = (0+1)/(1+2) = 1/3$$

Example 3

	docID	words in document	in $c = \text{China?}$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

b = feature vector for the document D

$b_t = \{0,1\} \Rightarrow$ absence or presence of word w_t in the document

$$P(C_k | b) \propto P(b | C_k) P(C_k)$$

$$\propto P(C_k) \prod_{t=1}^{|V|} [b_t P(w_t | C_k) + (1-b_t)(1 - P(w_t | C_k))].$$

$$\begin{aligned} P(c | d_5) &\propto \hat{P}(c) \cdot \hat{P}(\text{Chinese}|c) \cdot \hat{P}(\text{Japan}|c) \cdot \hat{P}(\text{Tokyo}|c) \\ &\quad \cdot (1 - \hat{P}(\text{Beijing}|c)) \cdot (1 - \hat{P}(\text{Shanghai}|c)) \cdot (1 - \hat{P}(\text{Macao}|c)) \\ &= 3/4 \cdot 4/5 \cdot 1/5 \cdot 1/5 \cdot (1 - 2/5) \cdot (1 - 2/5) \cdot (1 - 2/5) \\ &\approx 0.005 \end{aligned}$$

$$\begin{aligned} P(\bar{c} | d_5) &\propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1 - 1/3) \cdot (1 - 1/3) \cdot (1 - 1/3) \\ &\approx 0.022 \end{aligned}$$

Naïve Bayes classifier: Summary Model



Model: joint probability distribution given by

- $P(X, Y) = P(Y) P(X|Y)$
- $P(X = X_1, \dots, X_n, Y = y_k) = P(Y = y_k) P(X = X_1, \dots, X_n|Y = y_k)$

Learning/Training:

For output variable Y

- $P(Y) \sim \text{Ber}(p)$

For each attribute X

- $P(X|Y) \sim \text{Ber}(\pi) \rightarrow \text{Multivariate Bernoulli NB } (X_i - \text{binary})$
- $P(X|Y) \sim \text{Multinom}(\pi, n) \rightarrow \text{Multinomial NB } (X_i - \text{multinomial})$
- $P(X|Y) \sim N(\mu, \sigma^2) \rightarrow \text{GaussianNB } (X_i - \text{real valued})$

Logistic Regression vs Naïve Bayes

Idea:

- Naïve Bayes allows computing $P(Y|X)$ by learning $P(Y)$ and $P(X|Y)$
- Why not learn $P(Y|X)$ directly?

Logistic Regression and Gaussian Naïve Bayes Classifier

- Interestingly, the parametric form of $P(Y|X)$ used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier.
- Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases
- Derivation given in CS-5 slides in self learning section and also attached PDF of new chapter from Tom Mitchell book

Features of Bayesian learning

- Each observed training example can **incrementally decrease or increase the estimated probability** that a hypothesis is correct.
- Flexible approach to learning than algorithms that **completely eliminate a hypothesis** if it is found to be inconsistent with any single example.

Practical Issues of Bayesian learning

- Require initial knowledge of many probabilities
 - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

References

- <https://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn07-notes-nup.pdf>
- <https://cs229.stanford.edu/summer2019/cs229-notes2.pdf>
- Tom Mitchell – Chapter 6
- <https://nlp.stanford.edu/IR-book/pdf/13bayes.pdf>



Logistic Regression & Naïve Bayes

Where does the **form** come from?

- Logistic regression hypothesis representation

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n)}}$$

- Consider learning $f: X \rightarrow Y$, where
 - X is a vector of real-valued features $[X_1, \dots, X_n]^T$
 - Y is Boolean
 - Assume all X_i are conditionally independent given Y
 - Model $P(X_i|Y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$
 - Model $P(Y)$ as Bernoulli π

What is $P(Y|X_1, X_2, \dots, X_n)$?

Slide credit: Tom Mitchell

Where does the **form** come from?

- $$\begin{aligned} P(Y = 1|X) &= \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)} && \text{Applying Bayes rule} \\ &= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} && \text{Divide by } P(Y = 1)P(X|Y = 1) \\ &= \frac{1}{1 + \exp(\ln(\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}))} && \text{Apply } \exp(\ln(\cdot)) \\ &= \frac{1}{1 + \exp(\ln(\frac{1-\pi}{\pi}) - \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})} && \text{Plug in } P(X_i|Y) \end{aligned}$$

$$P(x|y_k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}$$

$$\sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$$

$$P(Y = 1|X_1, X_2, \dots, X_n) = \frac{1}{1 + \exp(\theta_0 + \sum_i \theta_i X_i)}$$

Slide credit: Tom Mitchell

Where does the **hypothesis function** come from?

- Logistic regression hypothesis representation

$$P(Y=1|X) = h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n)}}$$

- Model **likelihood** $P(X_i|Y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$ and assume variance is independent of class, i.e. $\sigma_{i0} = \sigma_{i1} = \sigma_i$

$$P(x|y_k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}$$

- Model **prior** $P(Y)$ as Bernoulli π : $P(Y=1) = \pi$ and $P(Y=0) = 1-\pi$

What is $P(Y|X_1, X_2, \dots, X_n)$?

Logistic Regression –Bayesian Analysis

$$P(Y = 1|X) = \frac{P(Y = 1)P(X|Y = 1)}{P(Y = 1)P(X|Y = 1) + P(Y = 0)P(X|Y = 0)}$$

Applying Bayes rule

$$P(Y = 1|X) = \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

Divide by $P(Y = 1)P(X|Y = 1)$

$$P(Y = 1|X) = \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

Apply $\exp(\ln(\cdot))$

$$P(Y = 1|X) = \frac{1}{1 + \exp (\ln \frac{P(Y = 0)}{P(Y = 1)} + \ln \frac{P(X|Y = 0)}{P(X|Y = 1)})}$$

Logistic Regression –Bayesian Analysis

By independence assumption:

$P(Y=1)=\pi$ and $P(Y=0)=1-\pi$
by modelling $P(Y)$ as Bernoulli

$$\frac{P(X|Y=0)}{P(X|Y=1)} = \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \ln \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

Logistic Regression –Bayesian Analysis



Plug in $P(X_i|Y)$

$$P(Y = 1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right))}$$

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

$$w_0 = \ln \frac{1-\pi}{\pi} + \sum_i \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}$$

$$w_i = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2}$$

$$\begin{aligned} \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i - \mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i - \mu_{i1})^2}{2\sigma_i^2}\right)} \\ &= \sum_i \ln \exp\left(\frac{(X_i - \mu_{i1})^2 - (X_i - \mu_{i0})^2}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{(X_i - \mu_{i1})^2 - (X_i - \mu_{i0})^2}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{(X_i^2 - 2X_i\mu_{i1} + \mu_{i1}^2) - (X_i^2 - 2X_i\mu_{i0} + \mu_{i0}^2)}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{2X_i(\mu_{i0} - \mu_{i1}) + \mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) \\ &= \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) \end{aligned}$$

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3,4 (Christopher M. Bishop, Pattern Recognition & Machine Learning) & Refresh your MFDS previous semester course basics

Next Session Plan :

Decision Trees
Handling Overfitting
Regularization
Evaluation Metrics



BITS Pilani
Pilani Campus

Machine Learning

DSE CLZG565

Model Evaluation & Metrics

Raja vadhana P
Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, Slides of Prof. Chetana, Prof. Sugata, Prof. Monali, Prof. Anita, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Agenda

- Evaluation of Supervised ML Model
 - Notion of Overfitting
 - Ways to Handle Overfitting
-
- ID3 - Decision Tree Classifier
 - Regularization in DT

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 2

Prerequisites : Refresher only

- Decision Theory
 - Minimum Misclassification Rate
 - Information Theory
 - Measure of Information, Entropy

Module – 7

- Decision Trees construction
- Entropy models : ID3 Algorithm
- Issues in Decision tree learning
- Random forest ← This subtopic will be discussed in Module 10 : Ensemble learners

Learnt so far in previous session

- Few Terminologies /Concepts
 - Naïve Bayes Classifier & MAP Estimation
 - Bayesian Learning
 - Text Classification using Naïve Bayes
 - Tokens, Vectorization
 - Text Categorization : Term Model vs Document Model
 - Laplace Smoothing
 - Bernoulli Naïve Bayes Model
 - Multinomial Naïve Bayes Model
 - Gaussian Naïve Bayes Model
 - Relation between Logistic Regression & Naïve Bayes Classifier
-



Evaluation Metrics

Evaluation of Linear Regression Model



Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51
....

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$$

Unseen Data	
Mileage (in kmpl)	Car Price (in cr)
7.5	9.25
10	6.5
....

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

Model 1

$$\text{CarPrice} = 8.5 + 0.5 \text{ Mileage} - 1.5 \text{ Mileage}^2$$

Model 2

$$\text{CarPrice} = -5.5 + 1.5 \text{ Mileage}$$

Evaluation of Linear Regression Model



R-squared/Adjusted R-squared

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51
....

variation in 'y' that is explained by a regression model

$$\text{explained variation} = \hat{y} - \bar{y}$$

variation in 'y' that is not captured/explained by a regression model

$$\text{unexplained variation} = y - \hat{y}$$

$$\text{total variation} = (y - \hat{y}) + (\hat{y} - \bar{y}) = (y - \bar{y})$$

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

Model 1

$$\text{CarPrice} = 8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2$$

Car Price (in cr)
10.48
1.75
6.95
2.51
Mean Y

$$SS_{explained} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

$$SS_{residual} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$SS_{Total} = \sum_{i=1}^n (y_i - \bar{y})^2$$

where :

$SS_{explained}$ = explained variation sum of squares

$SS_{residual}$ = unexplained variation sum of squares

SS_{Total} = total variation sum of squares

- **variation that is explained by a regression model**
- **measures the goodness of fit of a regression model**

Evaluation of Linear Regression Model



R-squared/Adjusted R-squared

Mileage (in kmpl)	Car Price (in cr)
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51
....

$$R_{adj}^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

where :

R^2 = R - squared

n = number of samples/rows in the data set

p = number of predictors/features

Car Price (in cr)
10.48
1.75
6.95
2.51
Mean Y

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

Model 1

$$\text{CarPrice} = 8.5 + 0.5 \text{Mileage} - 1.5 \text{Mileage}^2$$

- It only increases if the newly added predictor improves the model's predicting power
- Adjusted for the number of independent variables in the model,
- Always less than or equal to R^2

Evaluation of Classification Model



Confusion Matrix

		PREDICTED CLASS	
		Class= Low	Class= High
ACTUAL CLASS	Class= Low	a (TP)	b (FN)
	Class= High	c (FP)	d (TN)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Which Classifier is better?

Low Skew Case

T1	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	50	50
	Class>No	1	99

Precision (p) = 0.98

TPR = Recall (r) = 0.5

FPR = 0.01

TPR/FPR = 50

F – measure = 0.66

T2	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	10	90

Precision (p) = 0.9

TPR = Recall (r) = 0.99

FPR = 0.1

TPR/FPR = 9.9

F – measure = 0.94

T3	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	1	99

Precision (p) = 0.99

TPR = Recall (r) = 0.99

FPR = 0.01

TPR/FPR = 99

F – measure = 0.99

Which Classifier is better?

Medium Skew Case

T1	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	50	50
	Class>No	10	990

Precision (p) = 0.83

TPR = Recall (r) = 0.5

FPR = 0.01

TPR/FPR = 50

F – measure = 0.62

T2	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	100	900

Precision (p) = 0.5

TPR = Recall (r) = 0.99

FPR = 0.1

TPR/FPR = 9.9

F – measure = 0.66

T3	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	10	990

Precision (p) = 0.9

TPR = Recall (r) = 0.99

FPR = 0.01

TPR/FPR = 99

F – measure = 0.94

Which Classifier is better?

High Skew Case

T1	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	50	50
	Class>No	100	9900

Precision (p) = 0.3

TPR = Recall (r) = 0.5

FPR = 0.01

TPR/FPR = 50

F – measure = 0.375

T2	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	1000	9000

Precision (p) = 0.09

TPR = Recall (r) = 0.99

FPR = 0.1

TPR/FPR = 9.9

F – measure = 0.165

T3	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	99	1
	Class>No	100	9900

Precision (p) = 0.5

TPR = Recall (r) = 0.99

FPR = 0.01

TPR/FPR = 99

F – measure = 0.66

Which Model should you use?

	False Positive Rate	False Negative Rate
Model 1	41%	3%
Model 2	5%	25%

Mistakes have different costs:

- Disease Screening – LOW FN Rate
- Spam filtering – LOW FP Rate

Actually the same model
- different thresholds

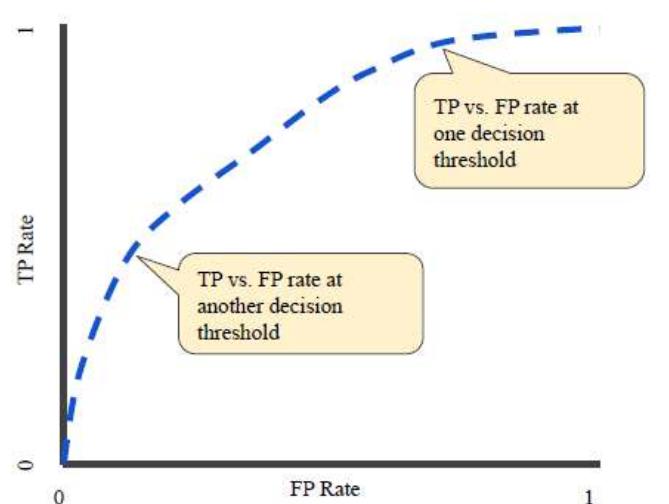
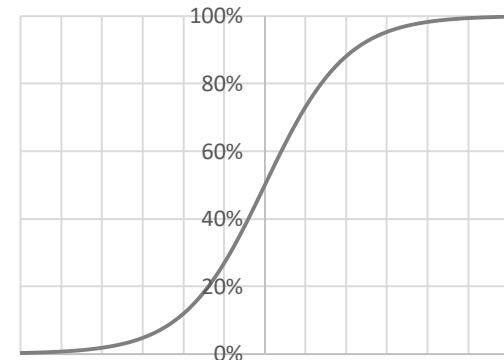
Conservative vs Aggressive settings:

- The same application might need multiple tradeoffs

Classifications and Probability Estimates



- Logistic regression produces a score between 0 – 1 (probability estimate)
- Use threshold to produce classification
- What happens if you vary the threshold?
- ROC Curve

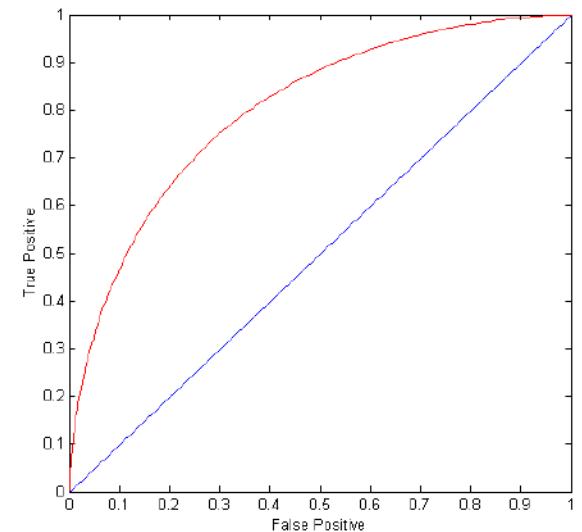


ROC Curve-(Receiver Operating Characteristic)

- ROC curve plots TPR against FPR
 - Performance of a model represented as a point in an ROC curve

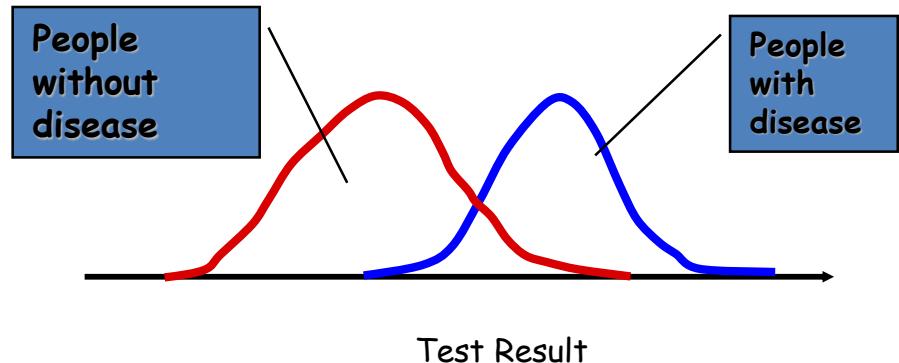
(TPR,FPR):

- (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

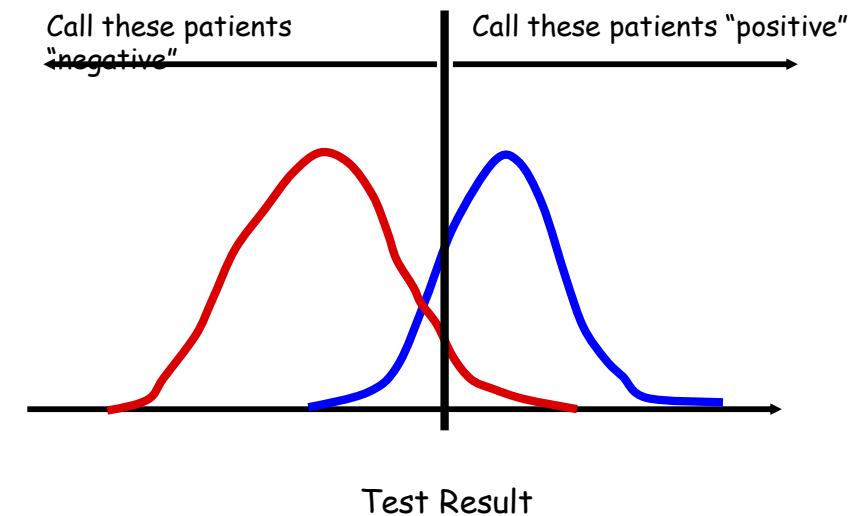
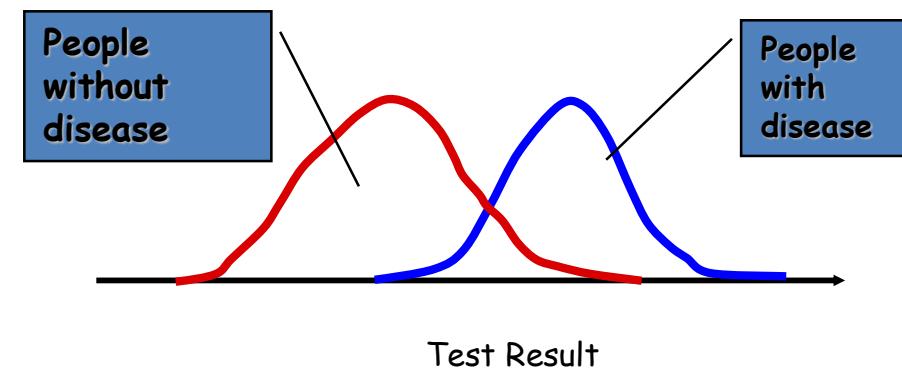


ROC Curve-(Receiver Operating Characteristic)

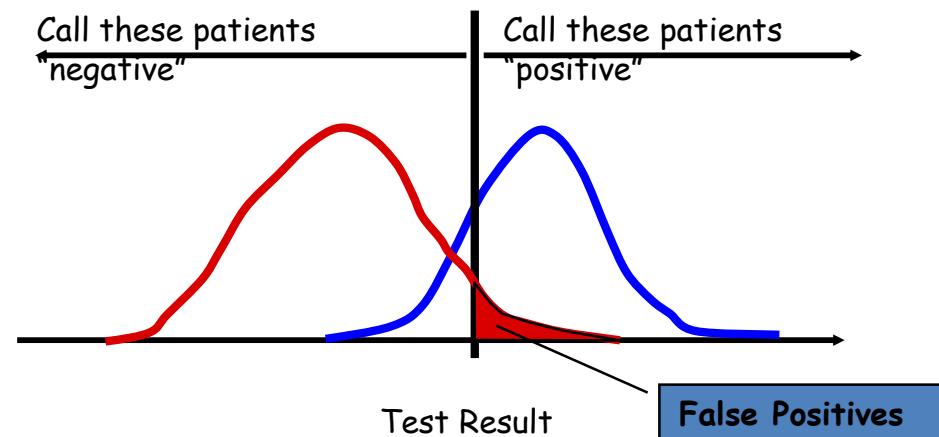
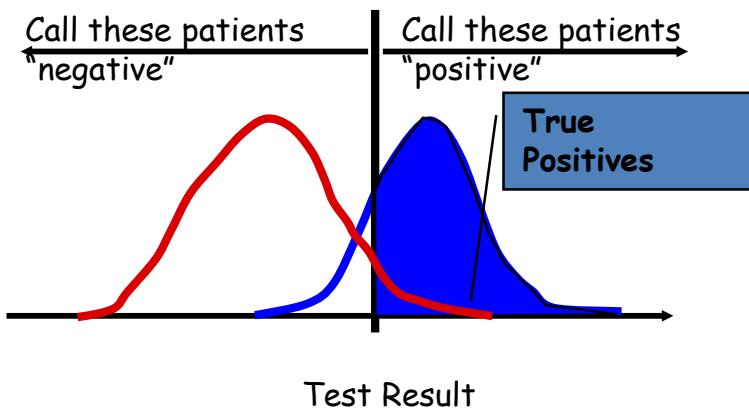
- ROC curve can be used to select a threshold for a classifier, which maximizes the true positives and in turn minimizes the false positives.
- ROC Curves help determine the exact trade-off between the true positive rate and false-positive rate for a model using different measures of probability thresholds.
- **ROC curves are more appropriate to be used when the observations present are balanced between each class.**



Example

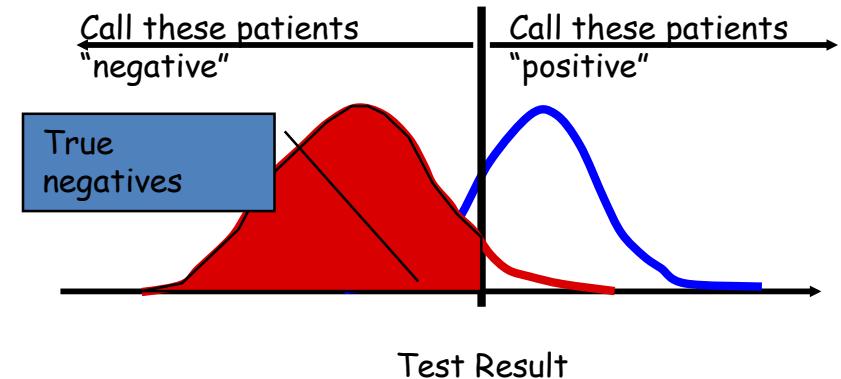
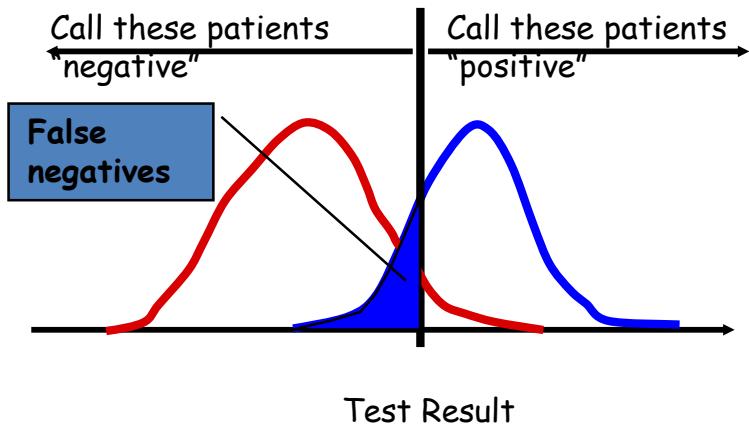


Example



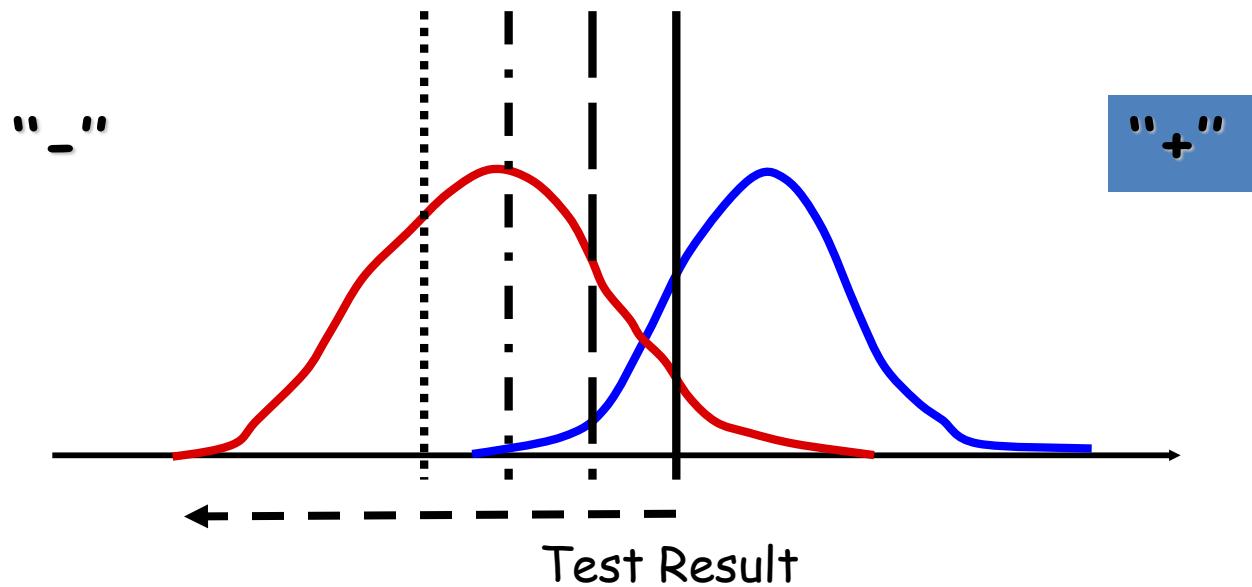
without the disease
with the disease

Example



without the disease
with the disease

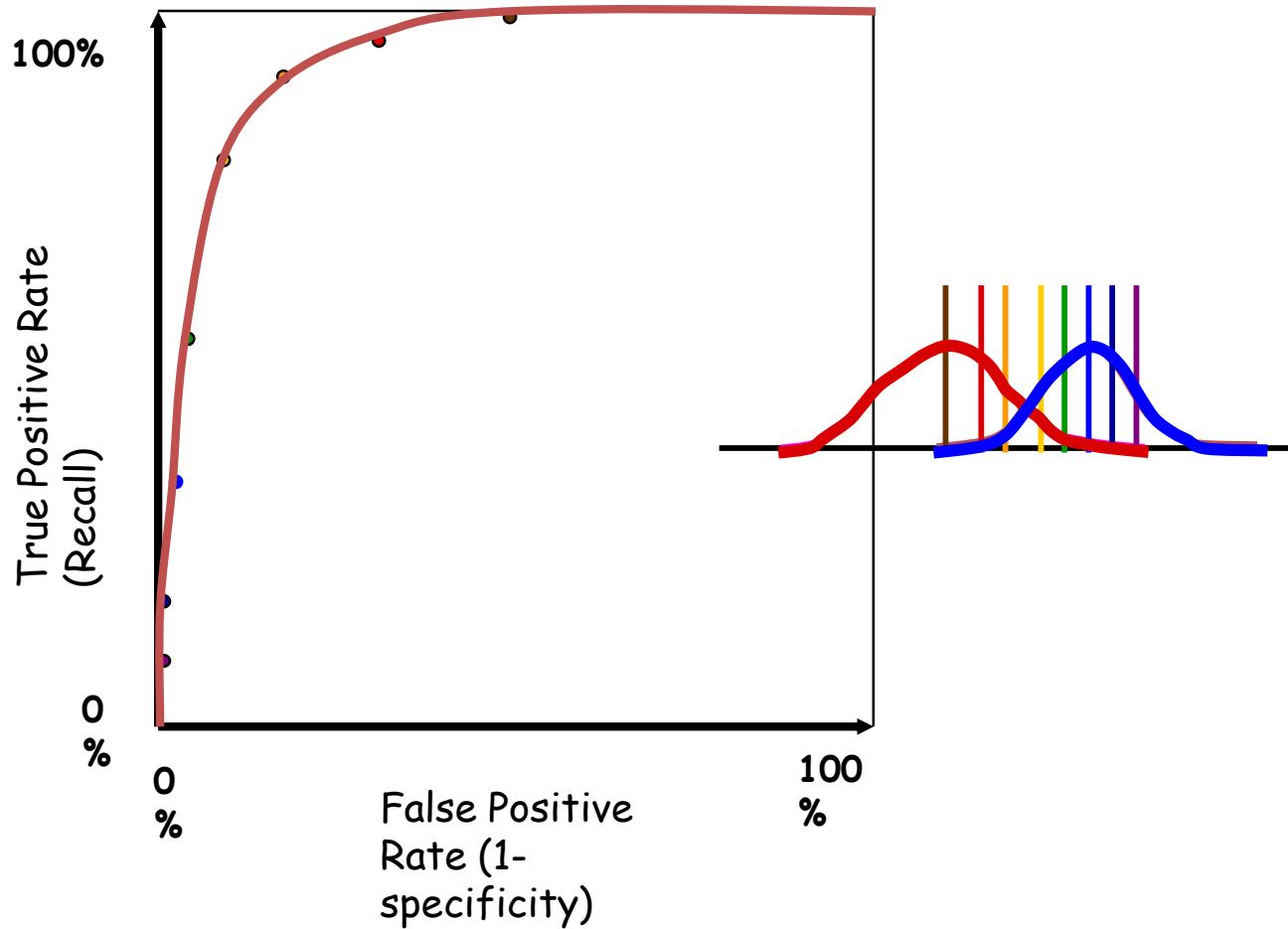
Moving the Threshold : Left



without the disease
with the disease

Which line has the higher recall of -?
Which line has the higher precision of -?

ROC Curve



How to Construct an ROC curve

Instance	Score	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	+

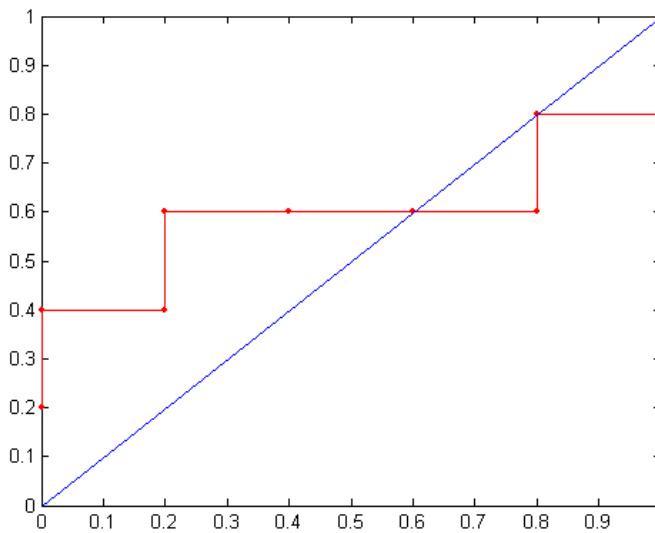
- Use a classifier that produces a continuous-valued score for each instance
 - The more likely it is for the instance to be in the + class, the higher the score
- Sort the instances in decreasing order according to the score
- Apply a threshold at each unique value of the score
- Count the number of TP, FP, TN, FN at each threshold
 - $TPR = TP/(TP+FN)$
 - $FPR = FP/(FP + TN)$

How to Construct an ROC curve

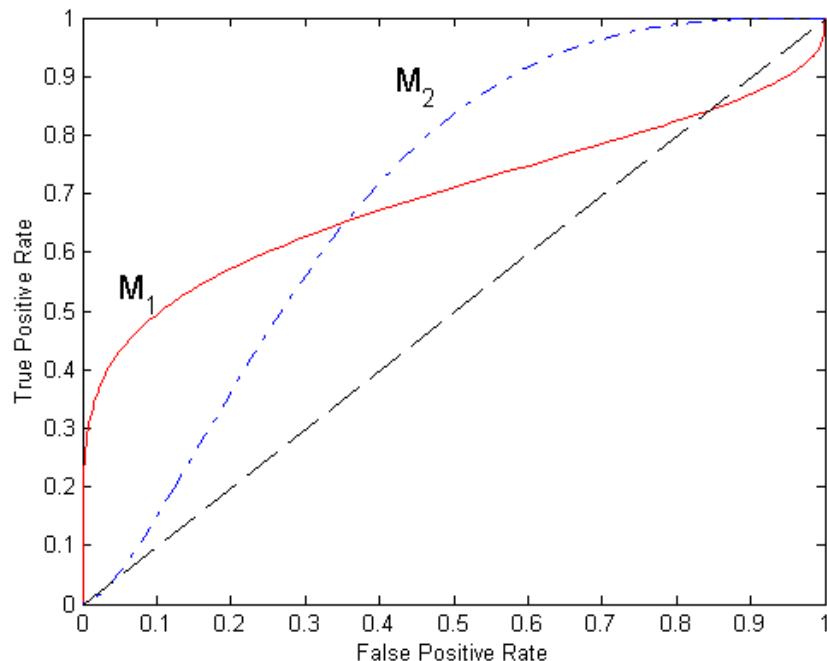
Threshold >=

Class	+	-	+	-	-	-	+	-	+	+	
	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:

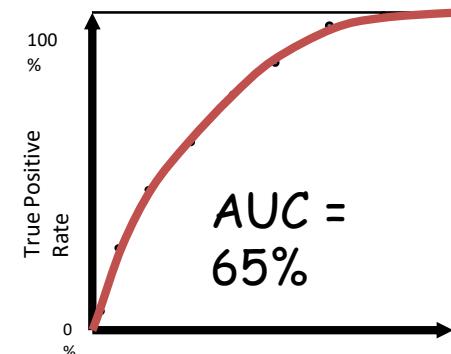
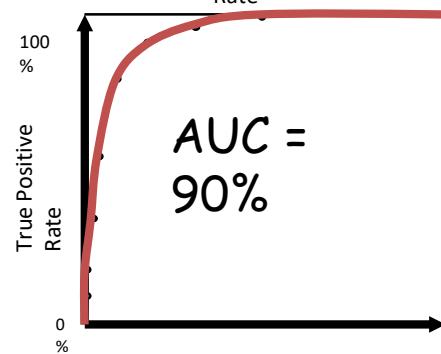
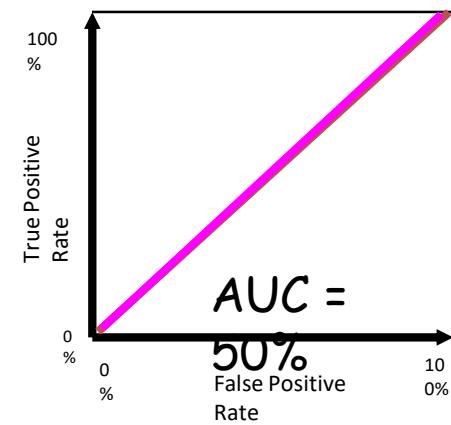
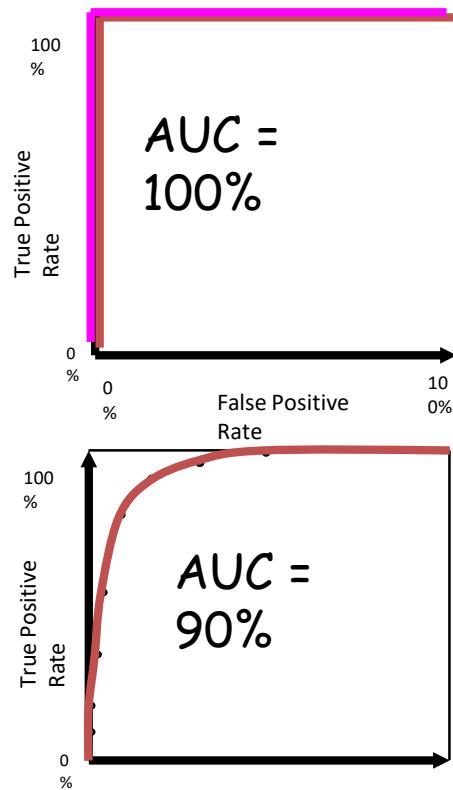
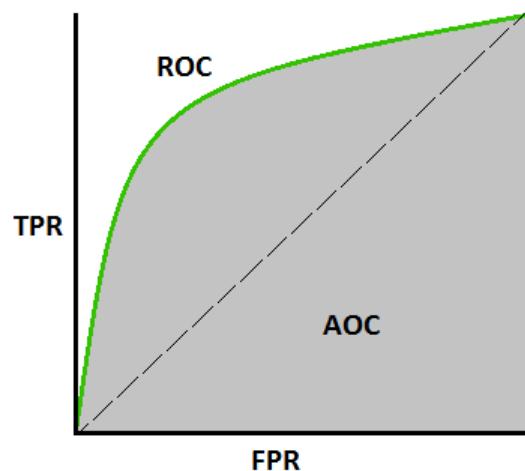


Interpretation of AUC ROC Curve

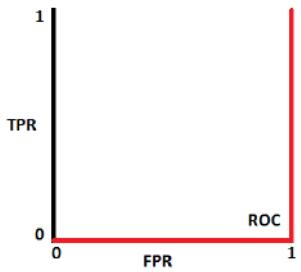
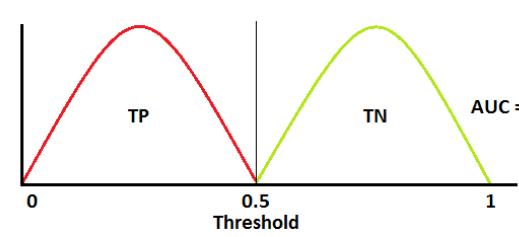
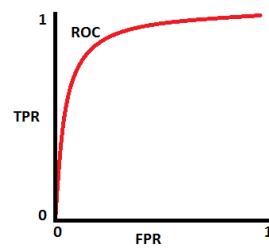
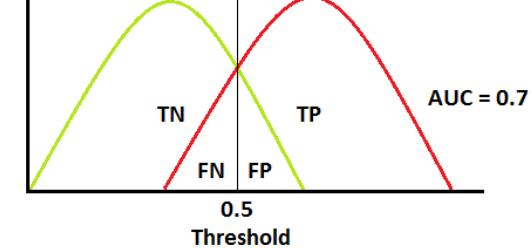
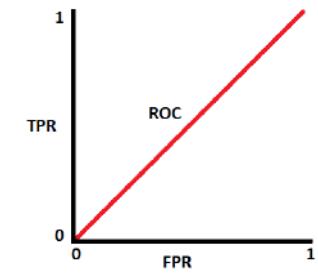
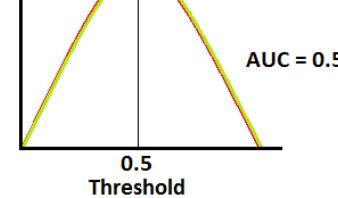
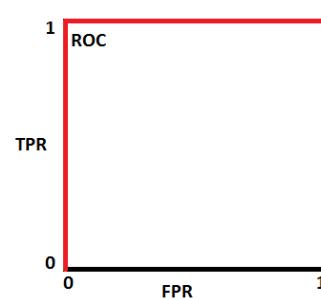
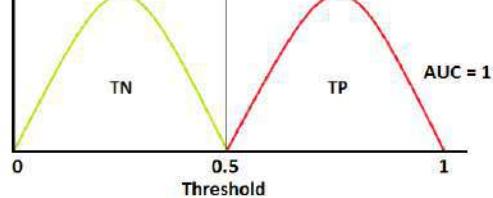


- No model consistently outperforms the other
 - M₁ is better for small FPR
 - M₂ is better for large FPR
- Area Under the ROC curve (AUC)
 - Ideal:
 - Area = 1
 - Random guess:
 - Area = 0.5

AUC - ROC Curve



AUC - ROC Curve

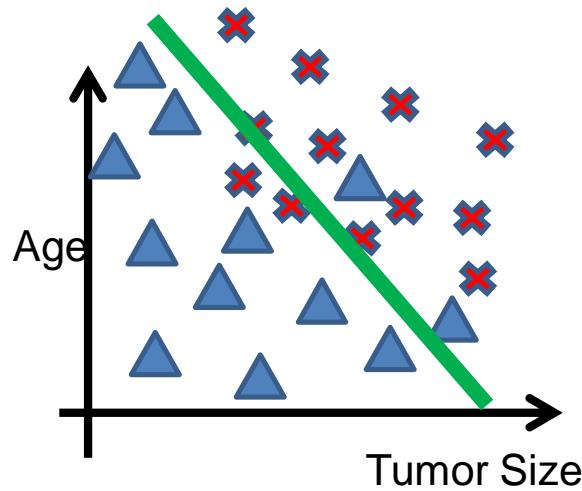


<https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>



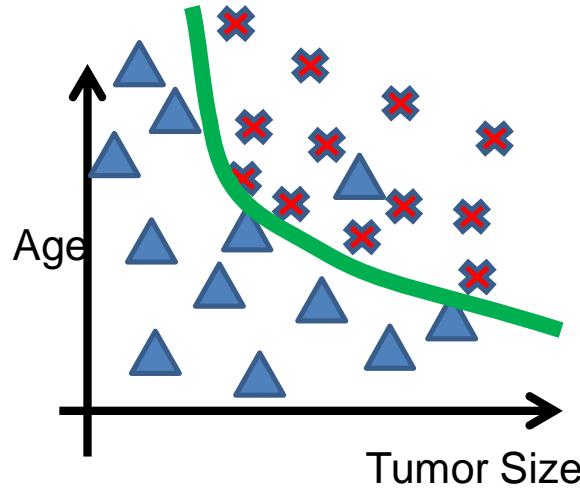
Evaluation of Model Complexity

Logistic regression

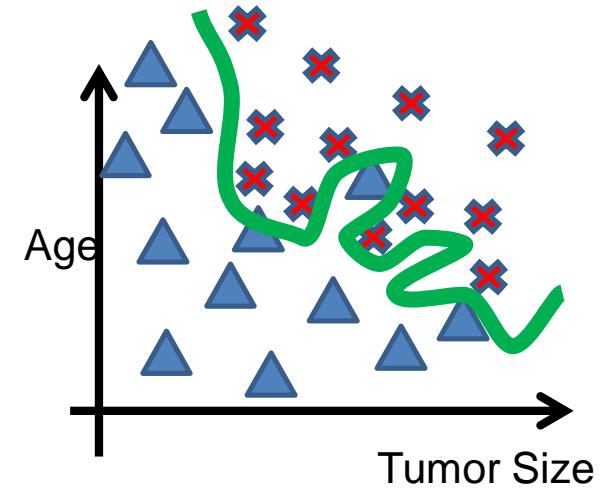


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x + \theta_2 x_2)$$

Underfitting



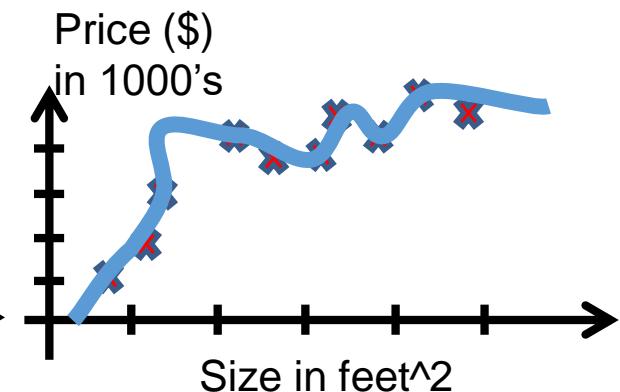
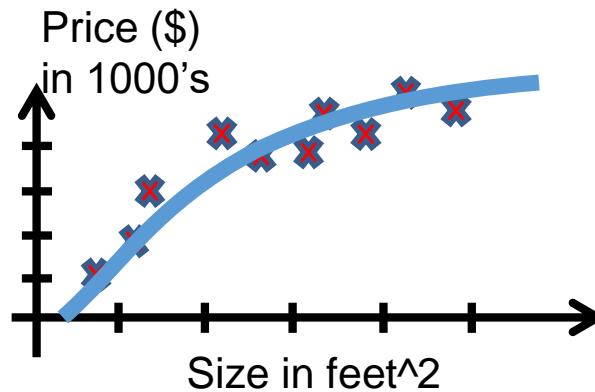
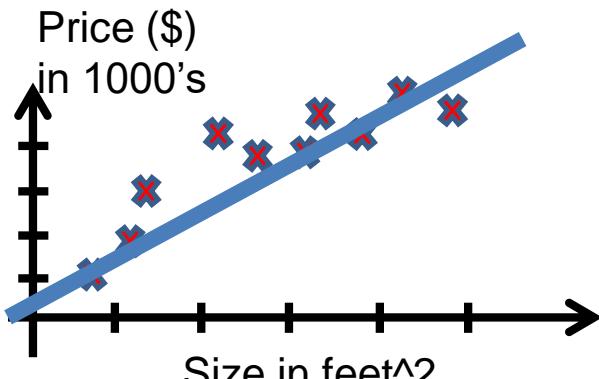
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \theta_6 x_1^3 x_2 + \theta_7 x_1 x_2^3 + \dots)$$

Overfitting

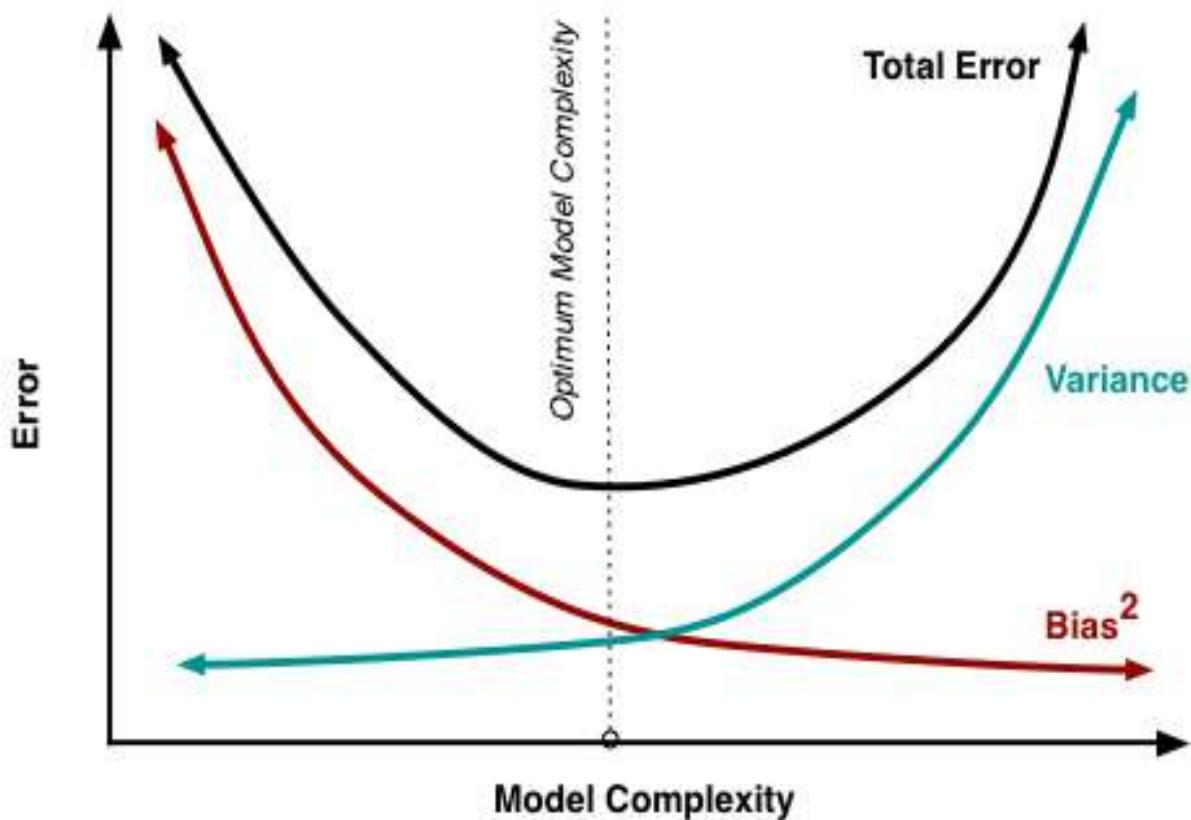
Linear regression



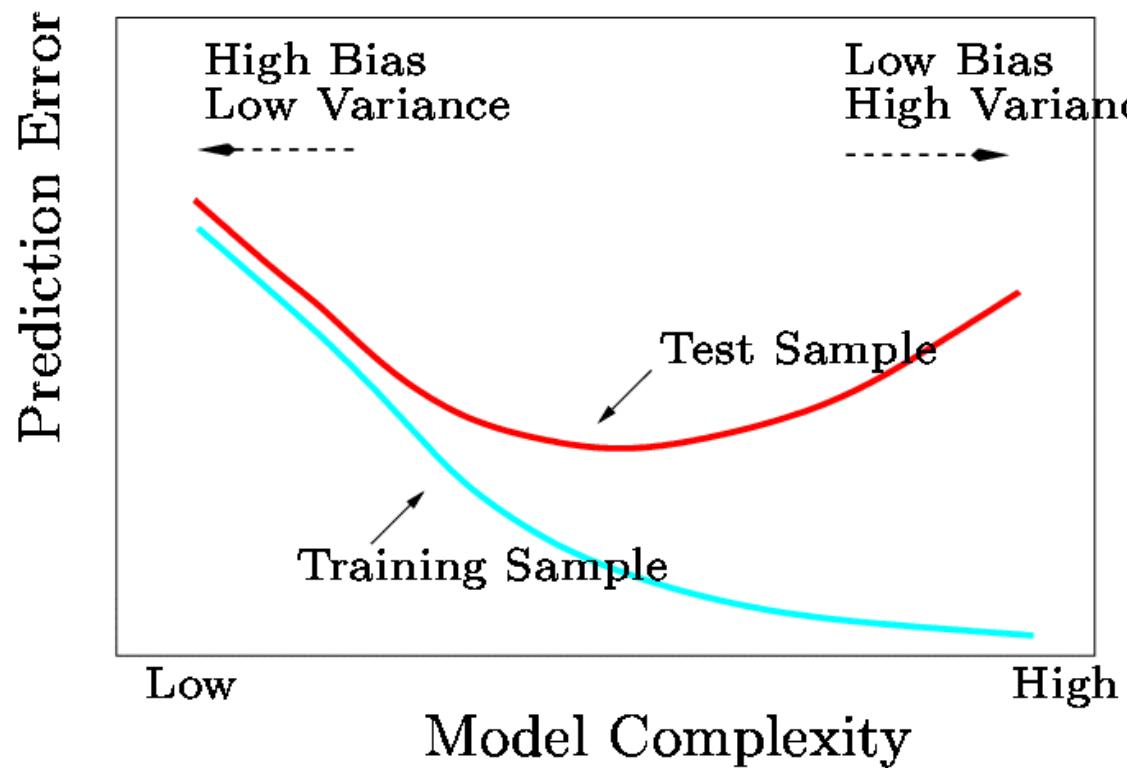
High bias

High variance

Model Complexity



Model Complexity



Bias–variance decomposition

- Training set $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- $y = f(x) + \varepsilon$
- We want $\hat{f}(x)$ that minimizes $E\left[\left(y - \hat{f}(x)\right)^2\right]$

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = (\text{Bias}[\hat{f}(x)])^2 + \text{Var}[\hat{f}(x)] + \sigma^2$$

$$\text{Bias}[\hat{f}(x)] = E[\hat{f}(x)] - f(x)$$

$$\text{Var}[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$$

Source Credit :

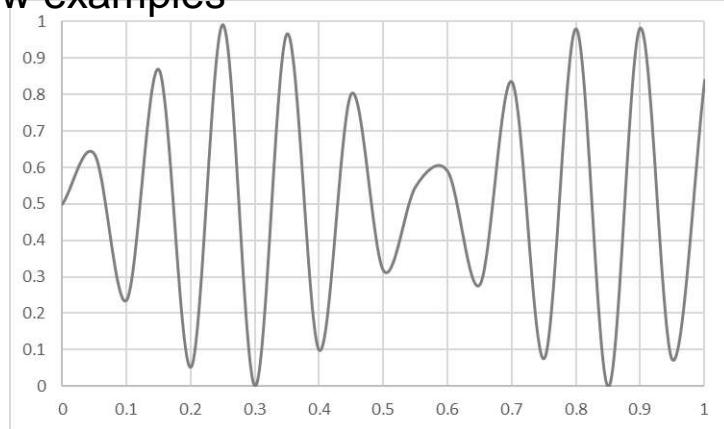
https://en.wikipedia.org/wiki/Bias-variance_tradeoff

Overfitting vs Underfitting

Overfitting

- Fitting the data too well
 - Features are noisy / uncorrelated to concept

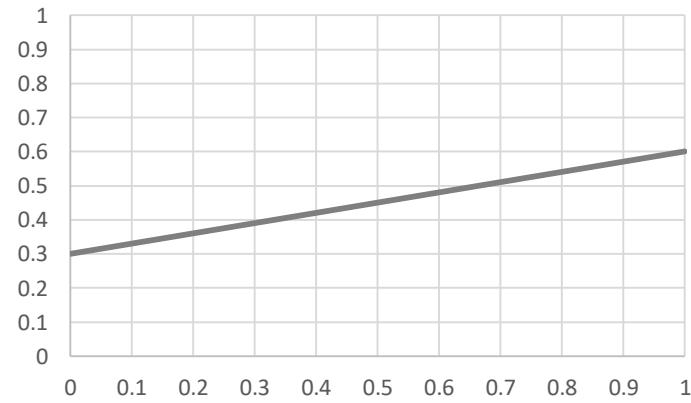
If we have too many features (i.e. complex model), the learned hypothesis may fit the training set very well but fail to generalize to new examples



$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$$

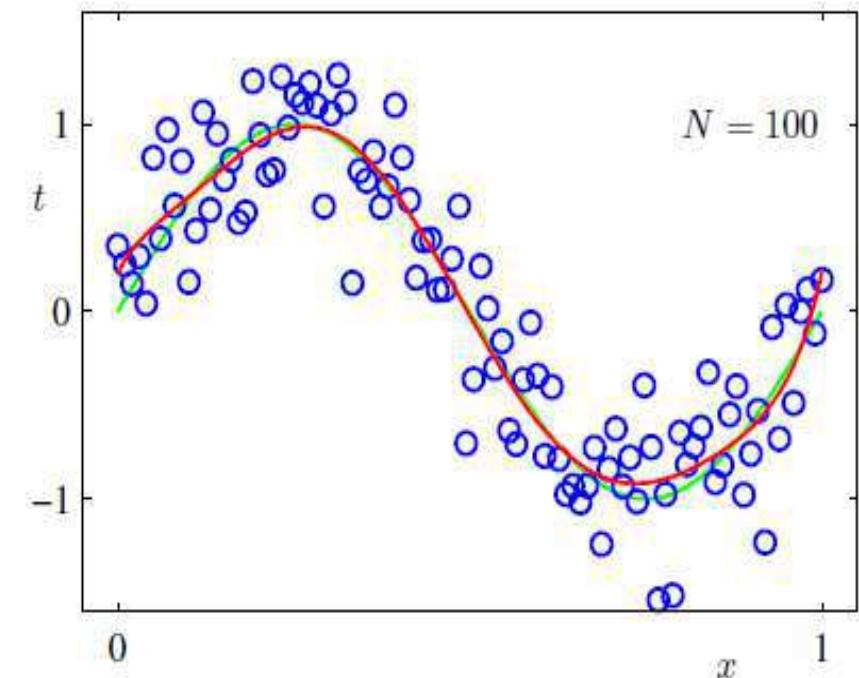
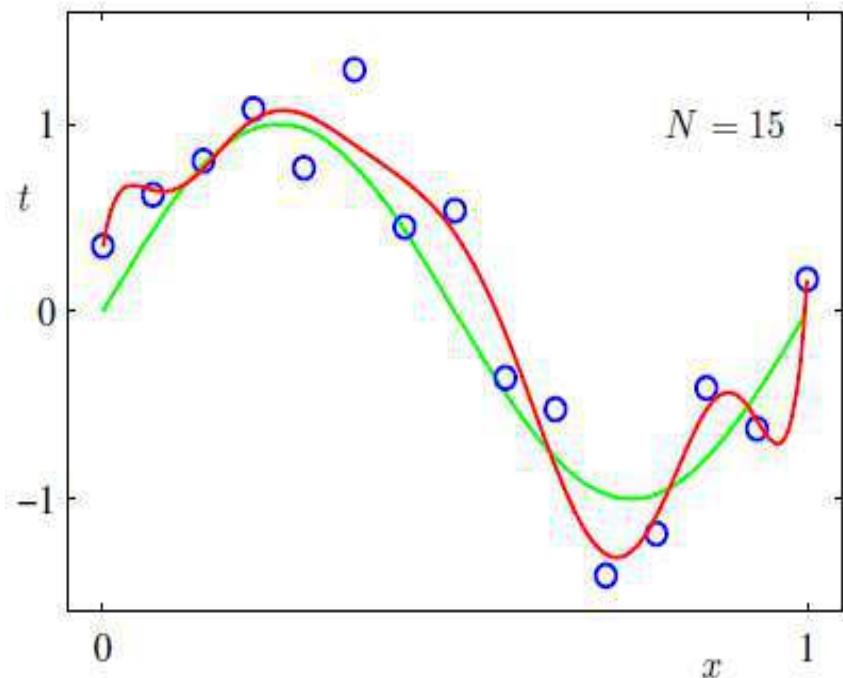
Underfitting

- Learning too little of the true concept
 - Features don't capture concept
 - Too much bias in model



Handling Overfitting

Size of the data set reduces the over-fitting problem





Regularization

Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of θ_j
 - Can incorporate into the cost function
 - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Gradient Descent Algorithm

Linear Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

$$h_\theta(x) = \theta^\top x$$

Logistic Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

Slide credit: Andrew Ng

Ways to Control Overfitting

- Regularization

$$Loss(S) = \sum_i^n Loss(\hat{y}_i, y_i) + \alpha \sum_j^{\# Weights} |\theta_j|$$

Adding a penalty term to the error function in order to discourage the coefficients from reaching large values.

E.g a sum of squares of all of the coefficients
Or Sum of absolute values of the coefficient

Note:

The hyperparameter controlling the regularization strength of a Scikit-Learn LogisticRegression model is not alpha (as in other linear models), but its inverse: C. The higher the value of C, the less the model is regularized.

Regularization

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$


- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0 !

Understanding Regularization

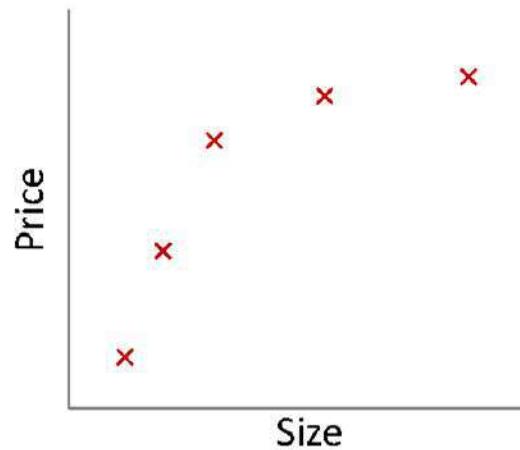
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- Note that $\sum_{j=1}^d \theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
 - This is the magnitude of the feature coefficient vector!
- We can also think of this as:
$$\sum_{j=1}^d (\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{0}\|_2^2$$
 - L₂ regularization pulls coefficients toward 0

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



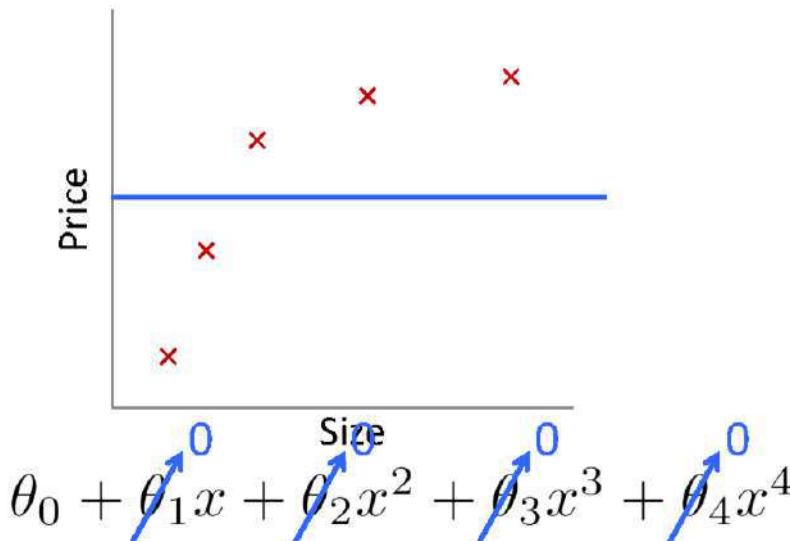
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Based on example by Andrew Ng

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



Based on example by Andrew Ng

Regularized Linear Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta}) \quad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \quad \theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j$$

- We can rewrite the gradient step as:

$$\theta_j \leftarrow \theta_j (1 - \alpha \lambda) - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Regularization function

Name

$$\|\theta\|_2^2 = \sum_{j=1}^n \theta_j^2$$

Tikhonov regularization
Ridge regression

$$\|\theta\|_1 = \sum_{j=1}^n |\theta_j|$$

LASSO regression

$$\alpha \|\theta\|_1 + (1 - \alpha) \|\theta\|_2^2$$

Elastic net regularization

α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

- L1 regularization has the ability to set some coefficients to 0 exactly leading to a *sparse* model
- L1 regularization helps in feature selection by eliminating the features that are not important
- L1 cannot be used in gradient-based approaches since it is not-differentiable unlike L2
- L2 will in general lead to small magnitudes of weights but not exactly 0.
- Highly correlated features - Elastic net

Lasso Regression (Least Absolute Shrinkage and Selection Operator Regression)

- Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d |\theta_j|$$

- Fit by solving $\min_{\theta} J(\theta)$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} J(\theta) \quad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \quad \theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) \mathbf{x}_j^{(i)} - \alpha \lambda \text{sign}(\theta_j)$$

regularization

where $\text{sign}(\theta_i) = \begin{cases} -1 & \text{if } \theta_i < 0 \\ 0 & \text{if } \theta_i = 0 \\ +1 & \text{if } \theta_i > 0 \end{cases}$

Typically L1 cannot be used in gradient-based approaches since it is not-differentiable unlike L2

References

- https://cs229.stanford.edu/notes2022fall/main_notes.pdf
- Ch 1 – Christopher Bishop
- Ch 5 : Introduction to Data Mining by Pang-Ning Tan Michael Steinbach Vipin Kumar
- <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

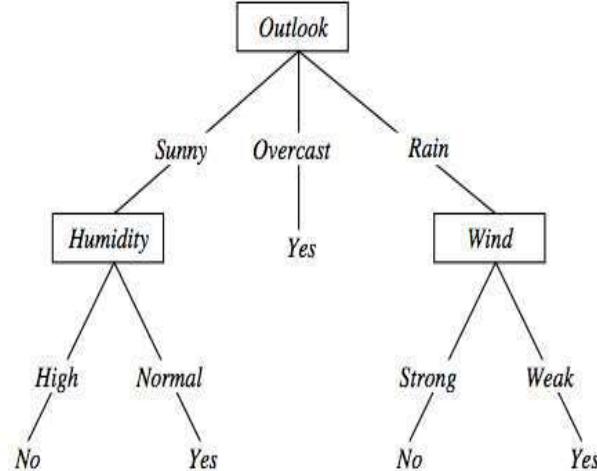


Decision Tree Classifier

Types of Classification

Decision Theory: Interpretation

Model Building



$IF\ OUTLOOK = Overcast\ THEN\ PLAY = Yes$
 ELSE
 $IF\ OUTLOOK = Rain\ AND\ WIND = Strong\ THEN\ PLAY = No$

Logistic regression, SVMs , tree based classifiers (e.g. decision tree) Traditional neural networks, Nearest neighbor

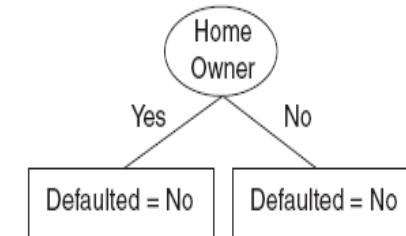
Sky	AirTemp	Humidity	Wind	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	Normal	Breeze	Same	Yes
Sunny	Hot	Normal	Breeze	Same	No
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Decision Tree Construction: Example

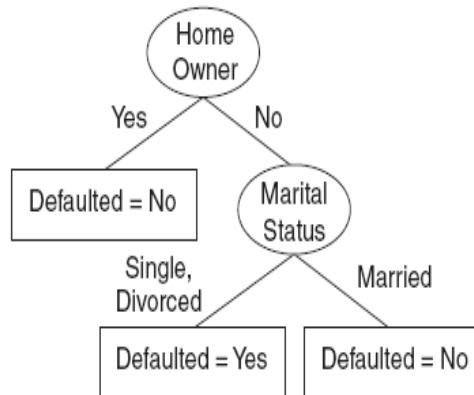
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Defaulted = No

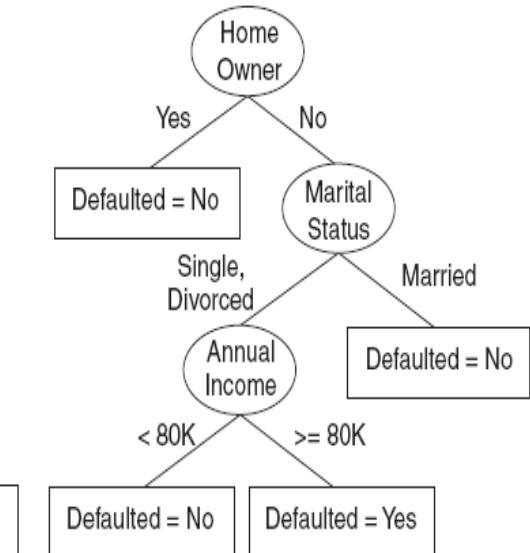
(a)



(b)



(c)



(d)

Decision Tree Construction: Hunt's Algorithm

- Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, y_3.. y_c\}$ be the class labels.

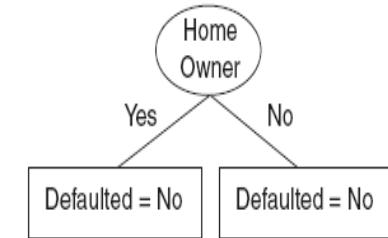
Step 1: If all the records in D_t belong to the same class y_t , then t is a leaf node labelled as y_t .

Step 2: If D_t contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes.

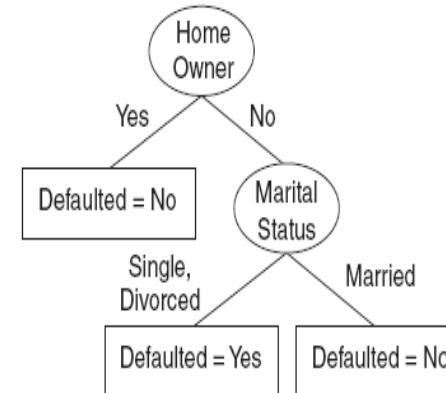
The algorithm is then recursively applied to each child node.



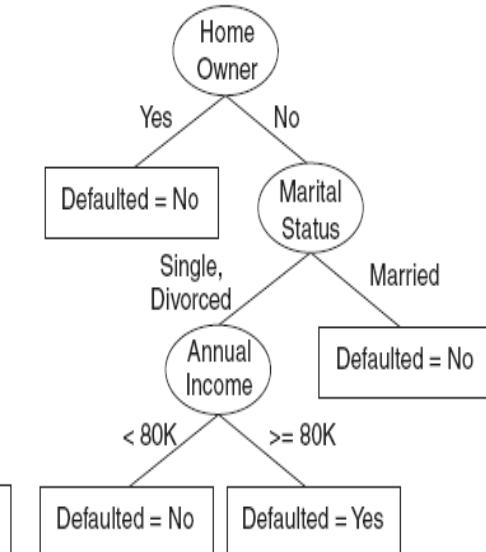
(a)



(b)



(c)

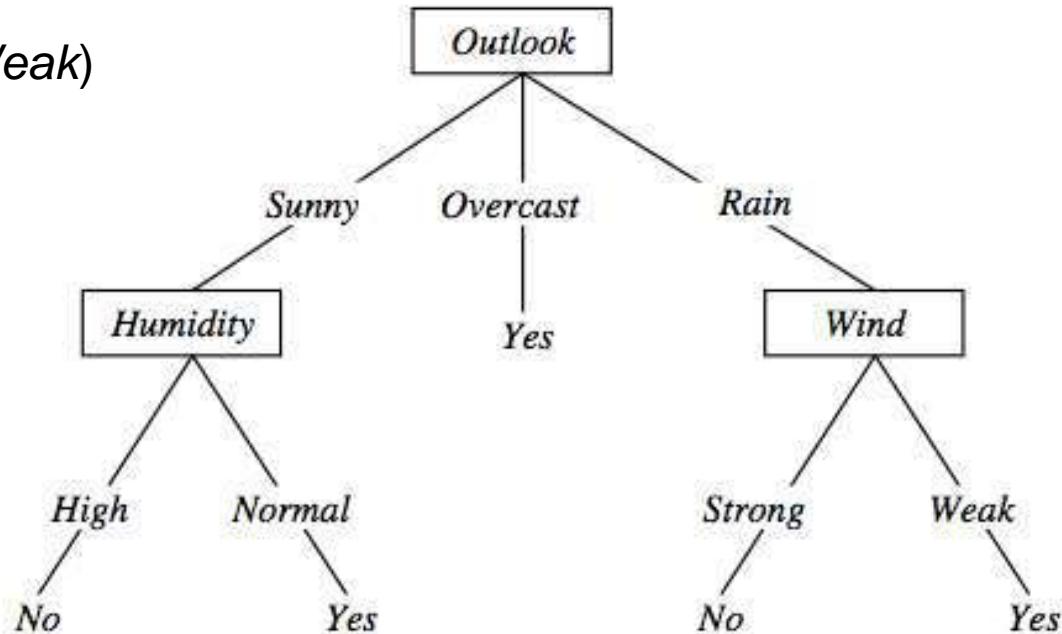


(d)

Decision Trees – Representation & Expressiveness

Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

$$\begin{aligned} & (\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal}) \vee \\ & (\text{Outlook} = \text{Overcast}) \vee \\ & (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak}) \end{aligned}$$



$\langle \text{Outlook}=\text{Sunny}, \text{Temp}=\text{Hot}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong} \rangle \quad \text{No}$

Measure of Information

- The amount of information (surprise element) conveyed by a message is inversely proportional to its probability of occurrence. That is $I_k \propto \frac{1}{p_k}$
- The mathematical operator satisfies above properties is the logarithmic operator. $I_k = \log_r \frac{1}{p_k} \text{ units}$

Entropy

- Entropy of discrete random variable $X=\{x_1, x_2 \dots x_n\}$

$$H(X) = E[I(X)] = E[-\log(P(X))].$$

since: $\log_2(1/P(\text{event})) = -\log_2 P(\text{event})$

- As uncertainty increases, entropy increases
- Entropy across all values

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i)$$

Entropy in general

- Entropy measures the amount of information in a random variable

$$H(X) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad X = \{+, -\}$$

for binary classification [two-valued random variable]

$$H(X) = -\sum_{i=1}^c p_i \log_2 p_i = \sum_{i=1}^c p_i \log_2 1/p_i \quad X = \{i, \dots, c\}$$

for classification in c classes

Entropy in binary classification

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable p .
 - S is a collection of training examples
 - p_+ the proportion of positive examples in S
 - p_- the proportion of negative examples in S

$$\text{Entropy}(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [0 \log_2 0 = 0]$$

$$\text{Entropy}([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$\text{Entropy}([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$\begin{aligned} \text{Entropy}([7+, 7-]) &= -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) = \\ &= 1/2 + 1/2 = 1 \end{aligned} \quad [\log_2 1/2 = -1]$$

Note: The log of a number < 1 is negative, $0 \leq p \leq 1$, $0 \leq \text{entropy} \leq 1$

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Values(A)$ possible values for A

S_v , subset of S for which A has value v

Decision Trees

Entropy of the Decision

Feature EnjoySport:

$$= \text{Entropy}_{\text{DecisionClass}}(S) = \sum p(c) \log_2 p(c)$$

$$= E(3,1) = - (3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) = 0.4421 + 0.5 = 0.94$$

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Note:

- Class balancing is recommended before building
- Value = 0 to log base 2 k, where k is the number of classes you have.

Decision Trees

Entropy of the Decision

Feature EnjoySport:

$$= \text{Entropy}_{\text{DecisionClass}}(S) = \sum p(c) \log_2 p(c)$$

$$= E(3,1) = -(3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) = 0.4421 + 0.5 = 0.94$$

Entropy of the attribute Sky:

$$= p(\text{Sunny}) \cdot E(3,0) + p(\text{Rainy}) \cdot E(0,1)$$

$$= (3/4) [-(3/3) \log_2 (3/3) - (1/1) \log_2 (1/1)] + (1/4) [-(0/1) \log_2 (0/1) - (1/1) \log_2 (1/1)]$$

$$= 0$$

Entropy of the attribute AirTemp:

$$= p(\text{Warm}) \cdot E(3,0) + p(\text{Cold}) \cdot E(0,1)$$

$$= (3/4) [-(3/3) \log_2 (3/3) - (1/1) \log_2 (1/1)] + (1/4) [-(0/1) \log_2 (0/1) - (1/1) \log_2 (1/1)]$$

$$= 0$$

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Sky	Enjoy	Not Enjoy
Sunny	3	0
Rainy	0	1

Air Temp	Enjoy	Not Enjoy
Warm	3	0
Cold	0	1

Decision Trees

Entropy of the Decision

Feature EnjoySport:

$$= \text{Entropy}_{\text{DecisionClass}}(S) = \sum p(c) \log_2 p(c)$$

$$= E(3,1) = -(3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) = 0.94$$

Entropy of the attribute Humidity :

$$= p(\text{Normal}) \cdot E(1,0) + p(\text{High}) \cdot E(2,1)$$

$$= (1/4) [-(1/1) \log_2 (1/1) - (0/1) \log_2 (0/1)] + (3/4) [-(2/3) \log_2 (2/3) - (1/3) \log_2 (1/3)]$$

$$= 0 + \frac{3}{4} [0.39 + 0.52] = 0.68$$

Entropy of the attribute Wind :

$$= p(\text{Strong}) \cdot E(3,1)$$

$$= (4/4) [-(3/4) \log_2 (3/4) - (1/4) \log_2 (1/4)]$$

$$= 1(0.94) = 0.94$$

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Humidity	Enjoy	Not Enjoy
Normal	1	0
High	2	1

Wind	Enjoy	Not Enjoy
Strong	3	1

Decision Trees

Entropy of the Decision

Feature EnjoySport:

$$= \text{Entropy}_{\text{DecisionClass}}(S) = \sum p(c) \log_2 p(c)$$

$$= E(3,1) = -(3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) = 0.94$$

Entropy of the attribute Forecast :

$$= p(\text{Same}) \cdot E(2,0) + p(\text{Change}) \cdot E(1,1)$$

$$= (2/4) [-(2/2) \log_2 (2/2) - (0/2) \log_2 (0/2)] + (2/4) [-(1/2) \log_2 (1/2) - (1/2) \log_2 (1/2)]$$

$$= 0 + 0.5 = 0.5$$

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Forecast	Enjoy	Not Enjoy
Same	2	0
Change	1	1

Decision Trees

Entropy of the Decision Feature EnjoySport: 0.94

Entropy of the attribute Sky: 0

Entropy of the attribute AirTemp: 0

Entropy of the attribute Humidity: 0.68

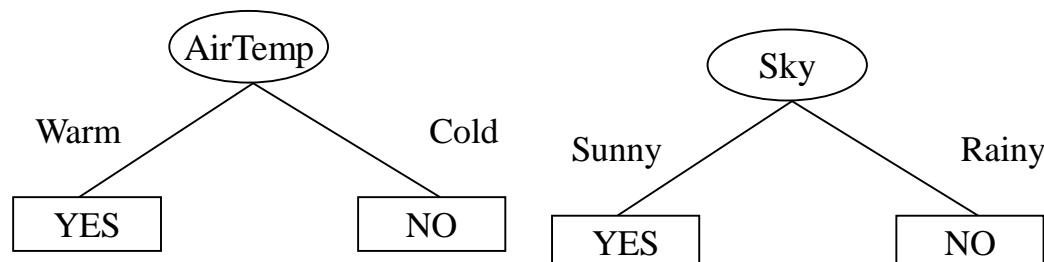
Entropy of the attribute Wind: 0.94

Entropy of the attribute Forecast: 0.5

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Attribute that generalizes the model faster = That which reduces the randomness if used to decide =
 That which have the highest info gain = SKY & AIRTEMP

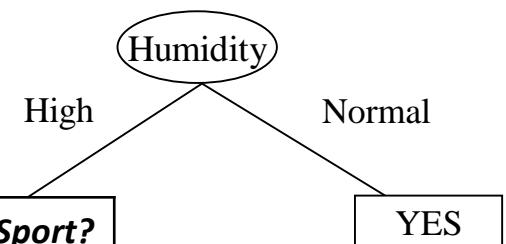
Decision Tree constructed :



Decision Trees

Sub Problem which has more scope to Model: None

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes



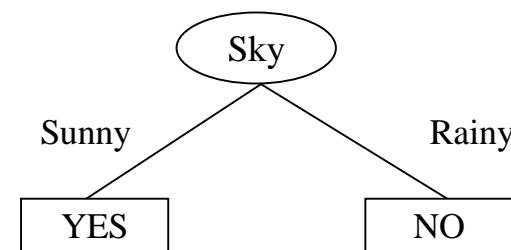
Sky	AirTemp	Wind	Forecast	EnjoySport?
Sunny	Warm	Strong	Same	Yes
Rainy	Cold	Strong	Change	No
Sunny	Warm	Strong	Change	Yes

Repeat the step again for the remaining features till:

- All training data are learnt
- No more learning is possible

Decision Trees

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	????



Evaluation

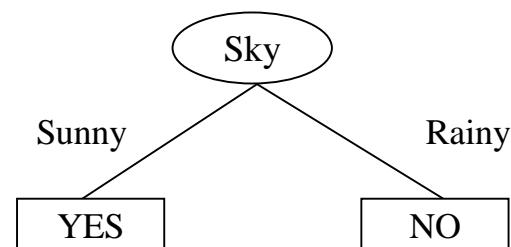
Confusion Matrix

- Parameters used in Confusion matrix are:
 - True positives (TP)**: These refer to the positive tuples that were correctly labelled by the classifier.
 - True negatives (TN)**: These are the negative tuples that were correctly labelled by the classifier.
 - False positives (FP)**: These are the negative tuples that were incorrectly labelled as positive .
 - False negatives (FN)**: These are the positive tuples that were incorrectly labelled as negative

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes

Actual class\Predicted class	C_1	$\neg C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

- $CM_{i,j}$ indicates the number of tuples of class i that were labelled by the classifier as class j.



Evaluation

Confusion Matrix

A\P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified by the classifier

- **Accuracy** = $(TP + TN)/(P+N)$

Error rate (or Misclassification rate): $1 - \text{accuracy}$, or

- **Error rate** = $(FP + FN)/(P+N)$

Sensitivity: True Positive (recognition) rate

- **Sensitivity** = TP/P

Specificity: True Negative recognition rate

- **Specificity** = TN/N

Actual class\Predicted class	Play = yes	Play = no	Total
Play = yes	6954	46	7000
Play = no	412	2588	3000
Total	7366	2634	10000

Evaluation

Other Metrics

- **Precision:** measure of exactness – what % of tuples that the classifier labeled as positive are actually positive.

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

A\P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All

- **Recall:** measure of completeness – what % of positive tuples did the classifier label as positive?

Perfect score is 1.0

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

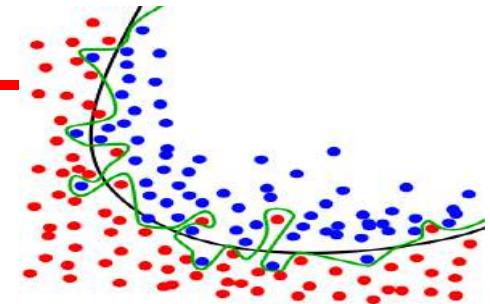
- **F measure (F_1 or F-score):** harmonic mean of precision and recall. F-score reaches its best value at 1 and worst at 0

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Actual class\Predicted class	Play = yes	Play = no	Total
Play = yes	6954	46	7000
Play = no	412	2588	3000
Total	7366	2634	10000

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
 - Too many branches, may reflect anomalies due to noise or outliers in the training data
 - Poor accuracy for unseen samples

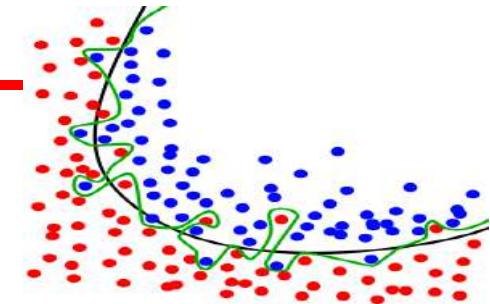


Approaches to tackle:

- Pruning
- Constraint on the depth of tree
- Constraint on the minimum nodes allowed at leaf

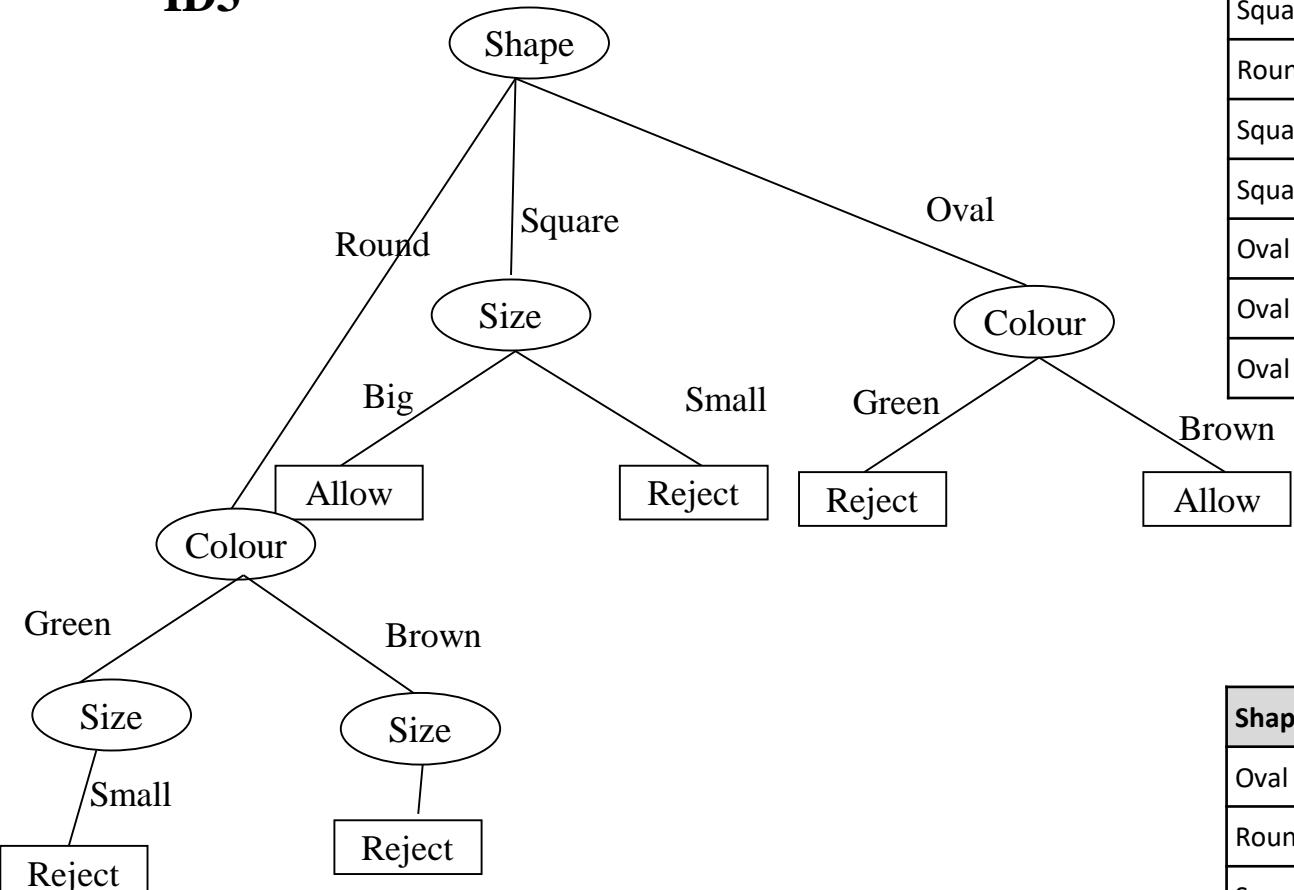
Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
 - Too many branches, may reflect anomalies due to noise or outliers in the training data
 - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
 - **Pre-pruning:** *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
 - Difficult to choose an appropriate threshold
 - **Post-pruning:** *Remove branches* from a “fully grown” tree—get a sequence of progressively pruned trees
 - Use a set of data different from the training data to decide which is the “best pruned tree”



Over fitting

- ID3

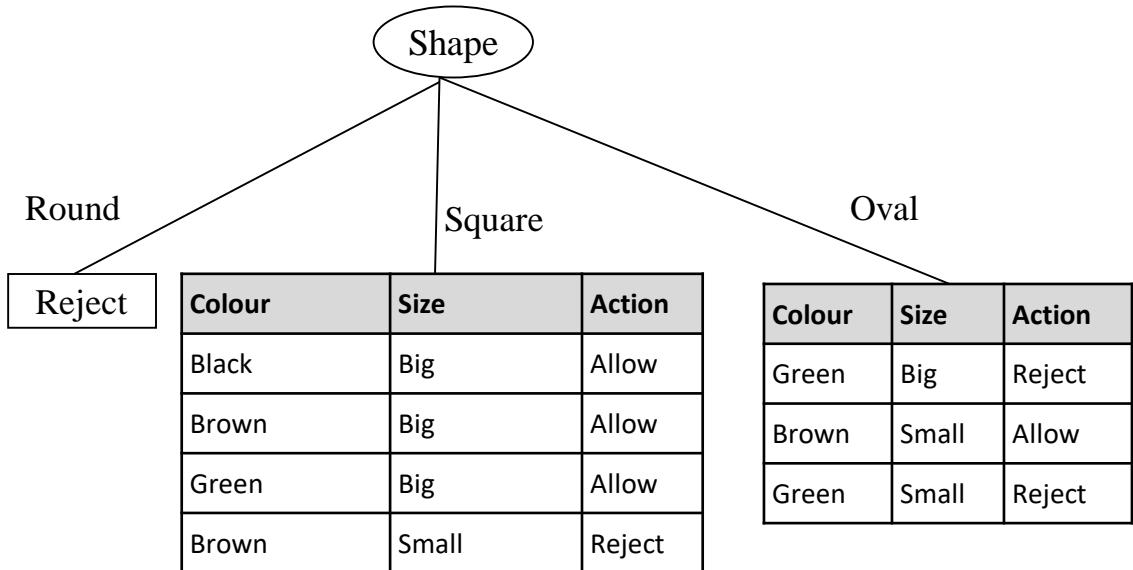


Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

- Threshold: Gain ≤ 0.85



Gain = 0.8113

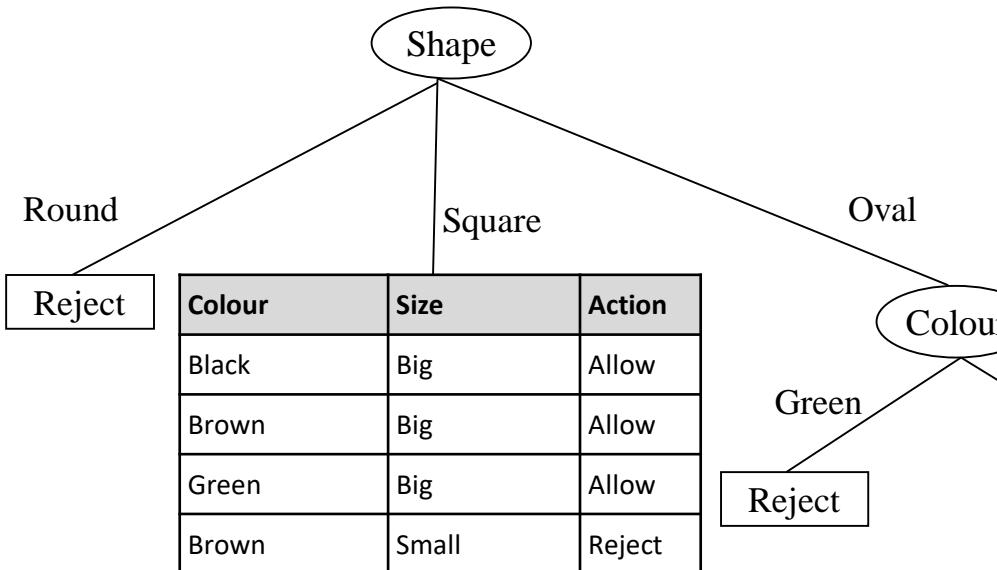
Gain = 0.9182

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

- Threshold: Gain ≤ 0.85

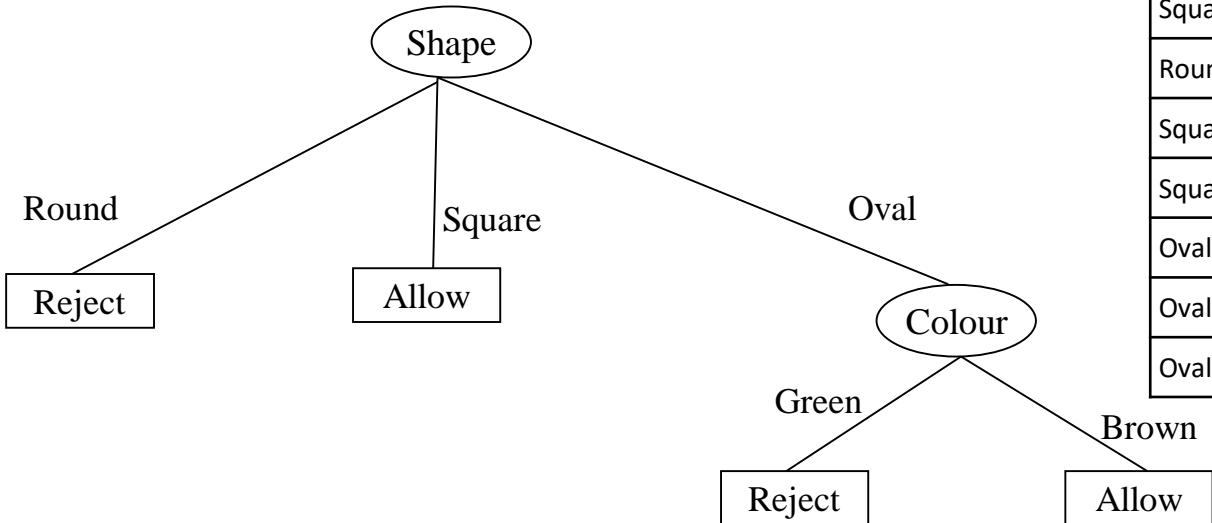


Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

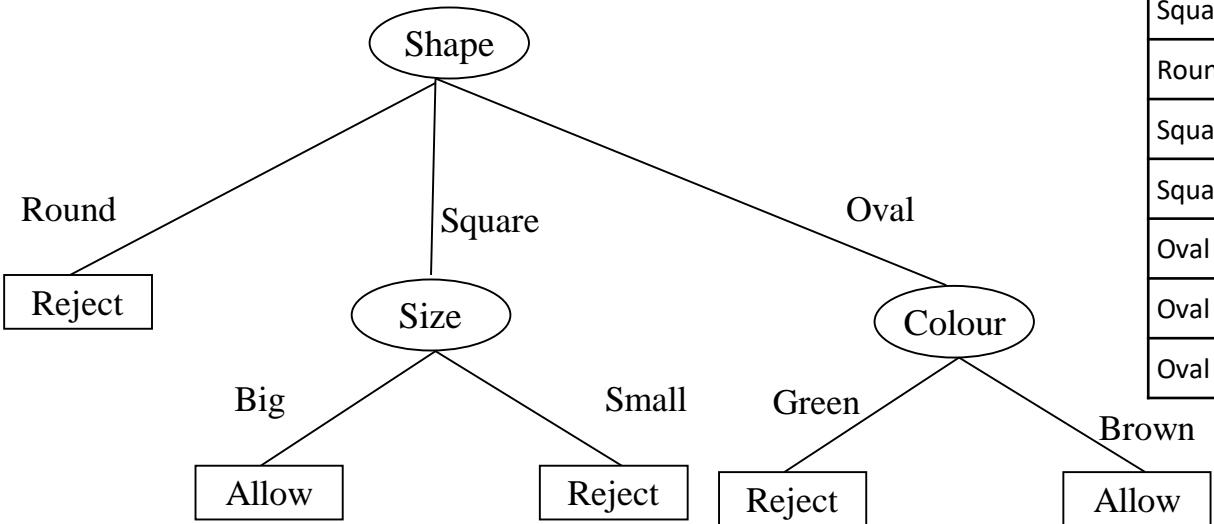
- Threshold: Gain ≤ 0.85



Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Post Pruning

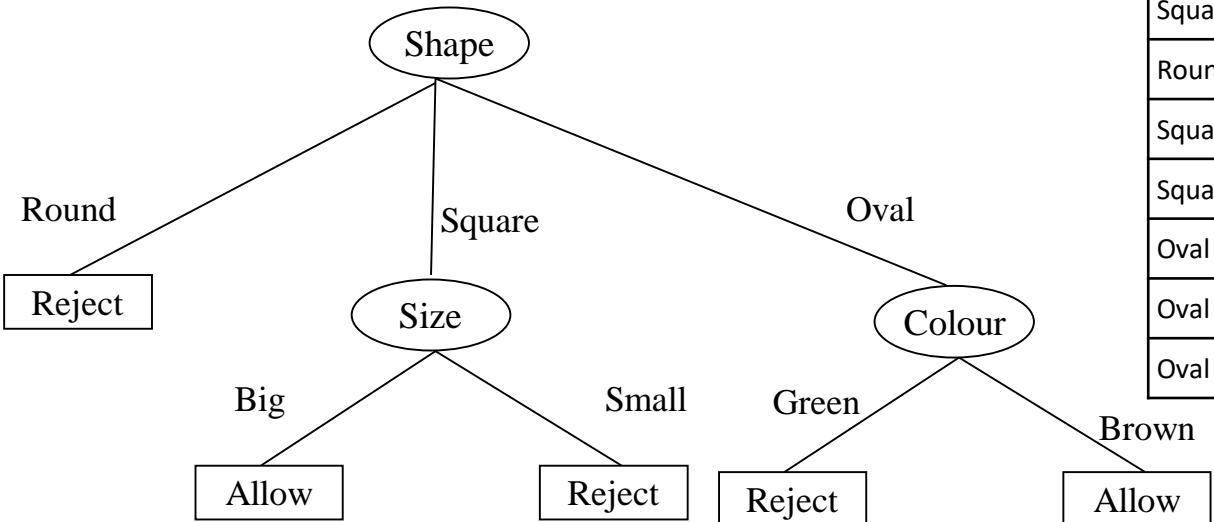


Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

- Error rate is the percentage of tuples misclassified
- Prune set is used to estimate the cost

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Post Pruning



Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Prune Size & Predict	Prune Colour & Predict	Above Tree's Prediction	Shape	Colour	Size	Action
Allow	Reject	Allow	Oval	Black	Small	Reject
Reject	Reject	Reject	Round	Brown	Big	Allow
Allow	Allow	Allow	Square	Brown	Big	Allow
Reject	Reject	Reject	Oval	Green	Small	Allow

ID3: algorithm

$\text{ID3}(X, T, \text{Attrs})$

X : training examples:
 T : target attribute (e.g. *PlayTennis*),
 Attrs : other attributes, initially all attributes

Create *Root* node

If all X 's are +, *return* *Root* with class +

If all X 's are -, *return* *Root* with class -

If Attrs is empty *return* *Root* with class most common value of T in X

else

$A \leftarrow$ best attribute; decision attribute for *Root* $\leftarrow A$

For each possible value v_i of A :

- add a new branch below *Root*, for test $A = v_i$

- $X_i \leftarrow$ subset of X with $A = v_i$

- *If* X_i is empty *then* add a new leaf with class the most common value of T in X

else add the subtree generated by $\text{ID3}(X_i, T, \text{Attrs} - \{A\})$

return *Root*

Decision Trees – Dealing with Continuous Values



- Given a continuous-valued attribute A , dynamically create a new attribute A_c

$$A_c = \text{True if } A < c, \text{ False otherwise}$$

- How to determine threshold value c ?
- Example. *Temperature* in the *PlayTennis* example
 - Sort the examples according to *Temperature*

<i>Temperature</i>	40	48		60	72	80		90
<i>PlayTennis</i>	No	No	54	Yes	Yes	Yes	85	No

- Determine candidate thresholds by averaging consecutive values where there is a change in classification: $(48+60)/2=54$ and $(80+90)/2=85$

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

& Refresh your MFDS previous semester course basics

Next Session Plan :

Remaining Topics

Refresher



Machine Learning

DSE CLZG565

Neural Network

Raja vadhana P

Assistant Professor,
BITS - CSIS



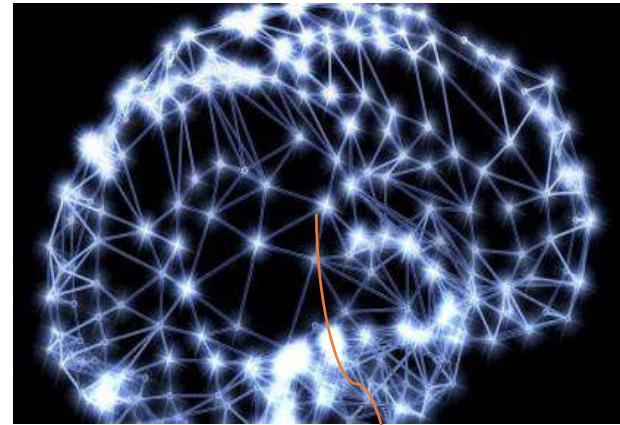
BITS Pilani
Pilani Campus



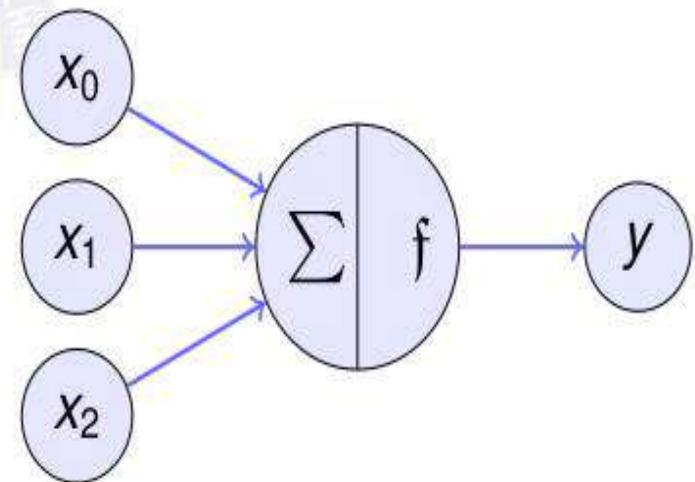
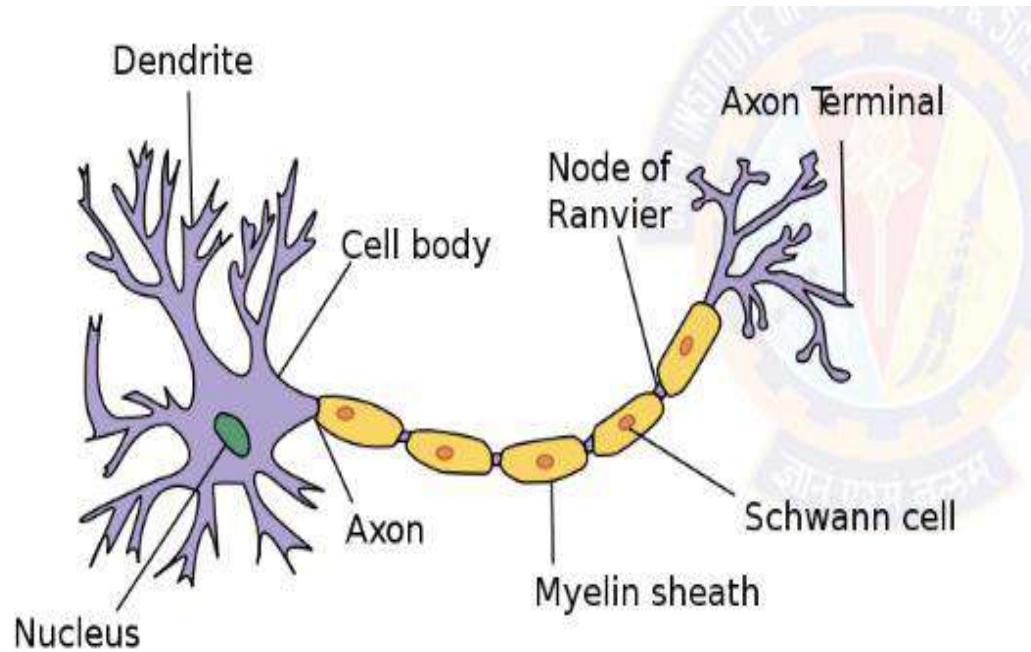
Neural Networks

Neural Networks

- **Origins:** Algorithms that try to mimic the brain.
- Very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure

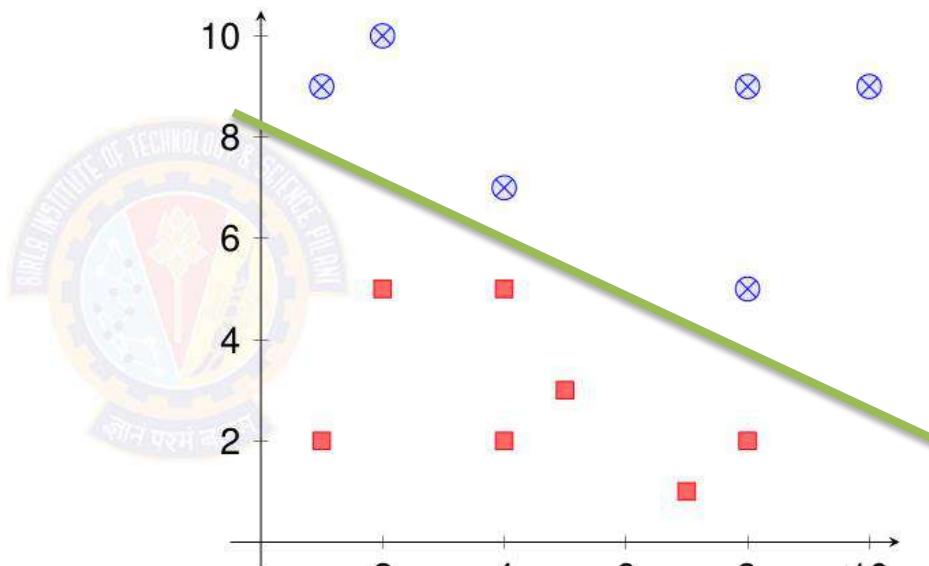


Neural Networks



Perceptron : Example

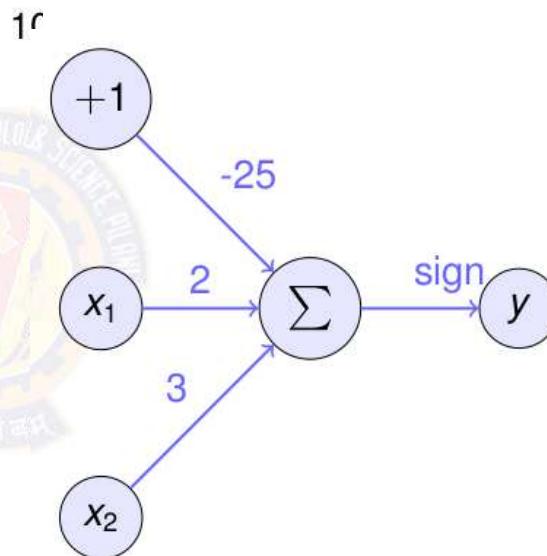
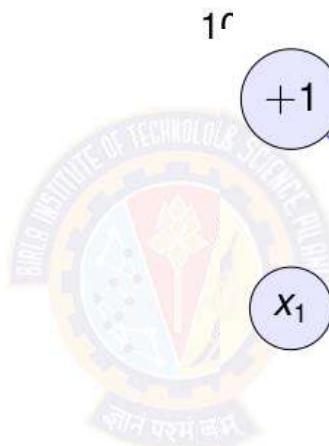
x_1	x_2	y
1	9	white
10	9	white
4	7	white
4	5	pink
5	3	pink
8	9	white
4	2	pink
2	5	pink
7	1	pink
2	10	white
8	5	white
1	2	pink
8	2	pink



Sample Decision boundary :
 $2x_1 + 3x_2 - 25 = 0$

Perceptron : Example

x_1	x_2	y
1	9	white
10	9	white
4	7	white
4	5	pink
5	3	pink
8	9	white
4	2	pink
2	5	pink
7	1	pink
2	10	white
8	5	white
1	2	pink
8	2	pink



Sample Decision boundary :
 $2x_1 + 3x_2 - 25 = 0$

Perceptron : Boolean Function

Input vector: $x = [x_0 \ x_1 \ x_2]$

Neuron Sums the weighted Inputs

Dot Product of Coefficient & Input: wx

$$= w_0x_0 + w_1x_1 + w_2x_2$$

$$= \sum_{i=0}^n w_i x_i = WX = a$$

Neuron Threshold Activation Function (T)

$$y = f(wx)$$

$$f(wx) = 0 \text{ if } wx < T$$

$$wx - T < 0$$

$$f(wx) = 1 \text{ if } wx \geq T$$

$$wx - T \geq 0$$

$$w_1(0) + w_2(0) + w_0(1) < 0$$

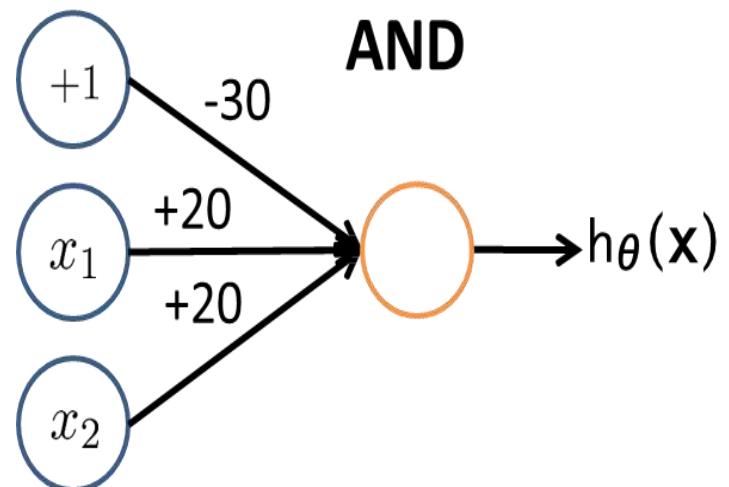
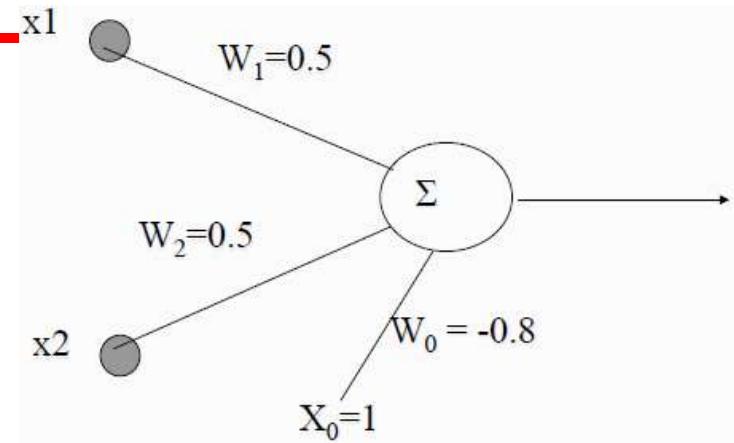
$$w_1(1) + w_2(0) + w_0(1) < 0$$

$$w_1(0) + w_2(1) + w_0(1) < 0$$

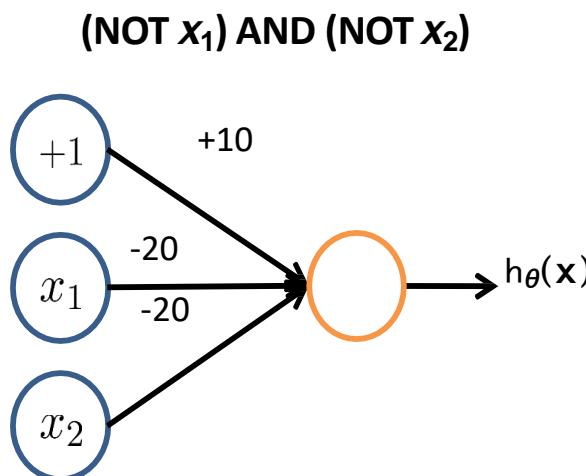
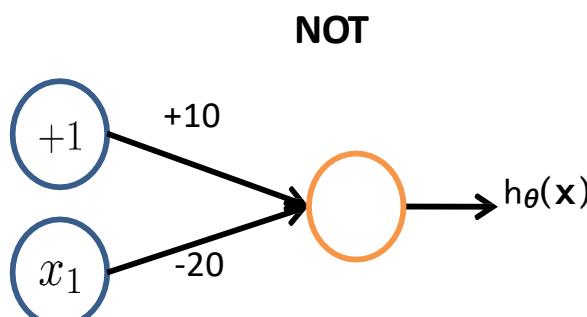
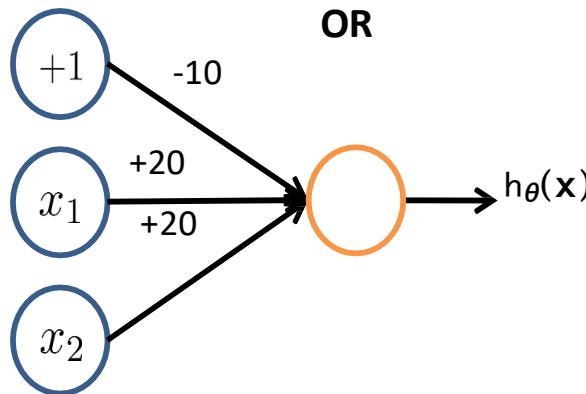
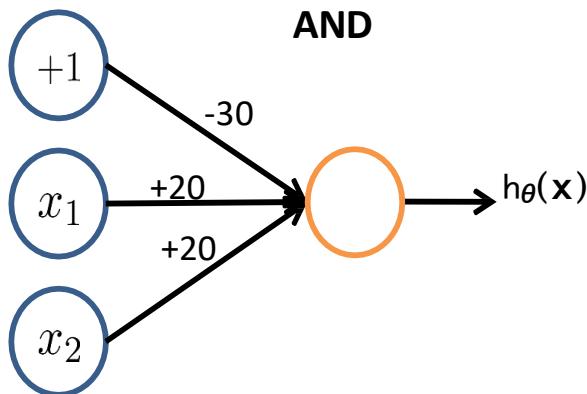
$$w_1(1) + w_2(1) + w_0(1) \geq 0$$

Find the value for weights

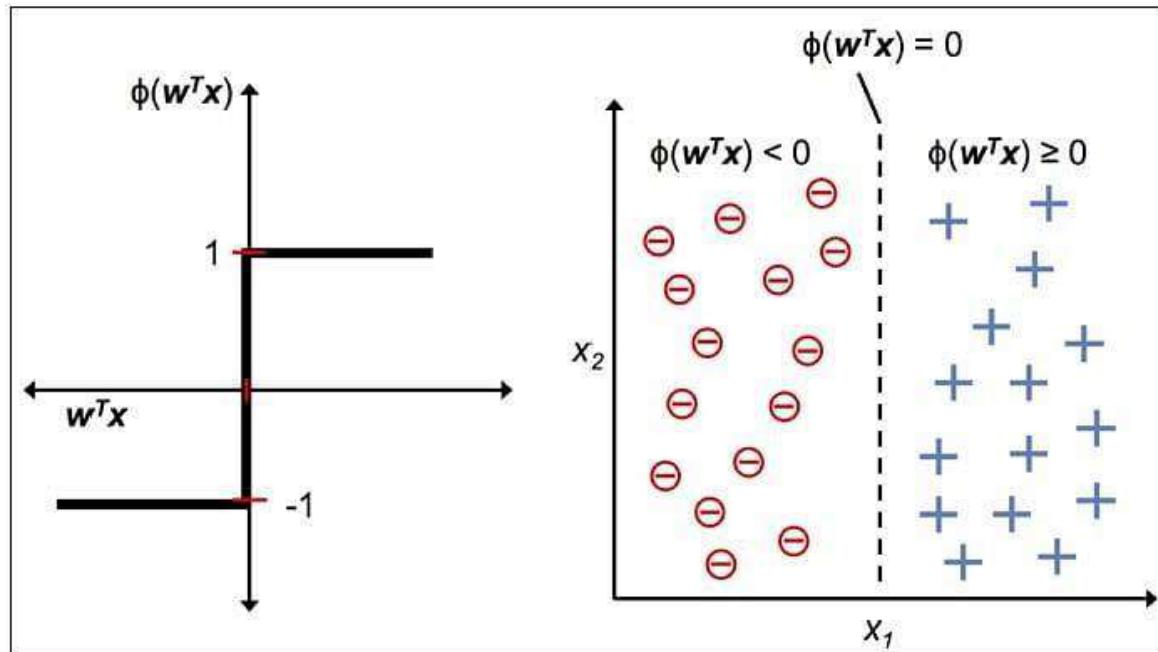
$$W = [w_0 \ w_1 \ w_2] = [1 \ 1 \ -1.5] \text{ OR } [0.5 \ 0.5 \ -1] \text{ OR } [20 \ 20 \ -30]$$



Perceptron : Boolean Function



Perceptron : Decision Surface

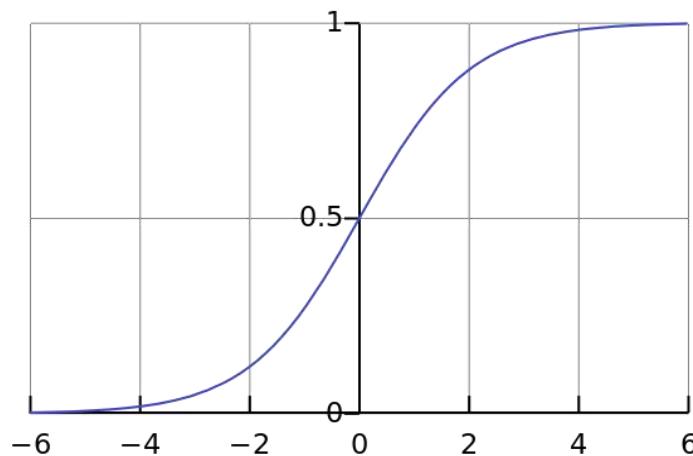
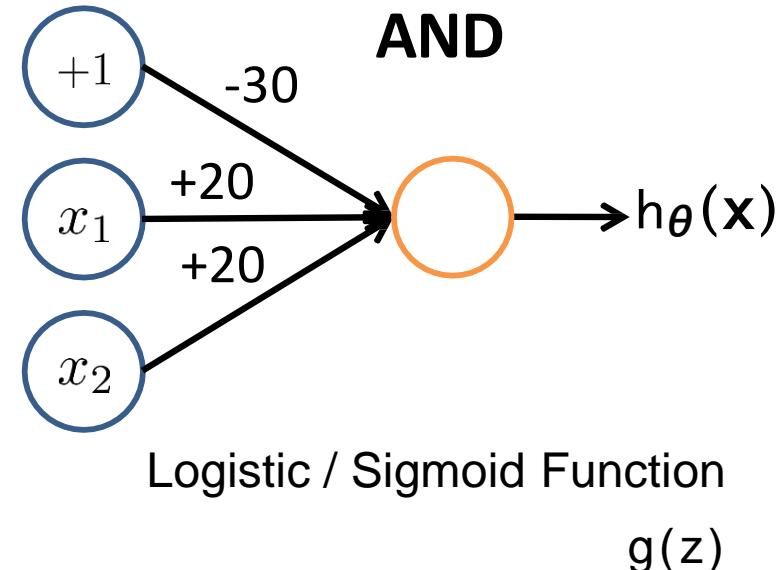


Based on slide and example by Andrew Ng

Perceptron : Decision Surface

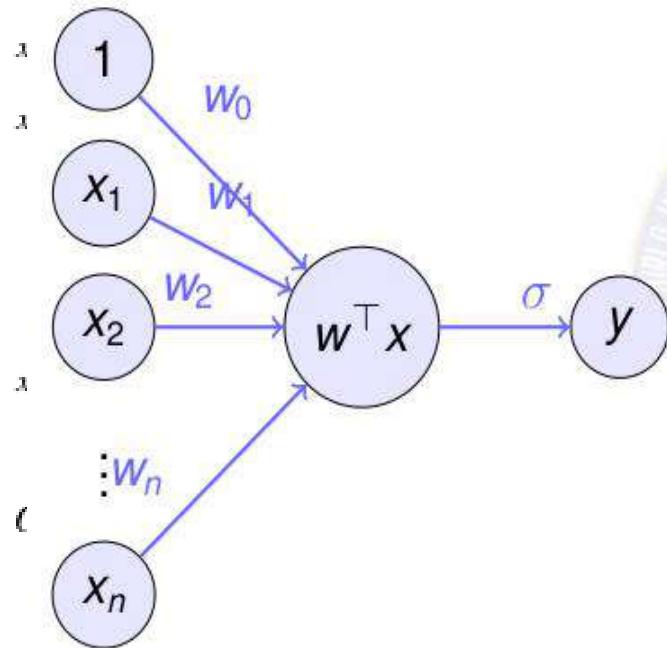
$$h_{\Theta}(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

x_1	x_2	$h_{\Theta}(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$



Based on slide and example by Andrew Ng

Perceptron : Summary



$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$\sum = w^\top x$$

$$y = \begin{cases} 1 & \text{if } \sigma(w^\top x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$1 + e^{-x}$$

...
er than

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

- The quantity $(-w_0)$ is a threshold that the weighted combination of inputs $w_1x_1 + \dots + w_nx_n$ must surpass in order for the perceptron to output a 1

Neural Net : Notations & Terminologies

Train Dataset has m training examples.

Single training example = $\{(x, y)\}$ where $x \in \mathbb{R}^n$ and $y \in \{0, 1\}$

training examples = m

m training examples = $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots (x^{(m)}, y^{(m)})\}$

$$X = \begin{bmatrix} | & | & \dots & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad \text{where } X \in \mathbb{R}^{n \times m}$$

$$Y = [y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)}] \quad \text{where } Y \in \mathbb{R}^{1 \times m}$$

Neural Net : Notations & Terminologies

Given X find $\hat{y} = P(y = 1 | X)$

Input $X \in \mathbb{R}^{n \times m}$

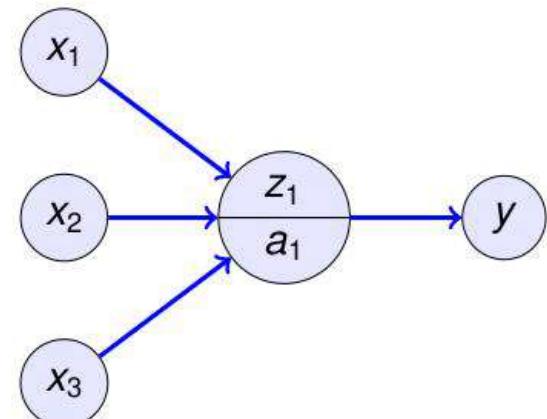
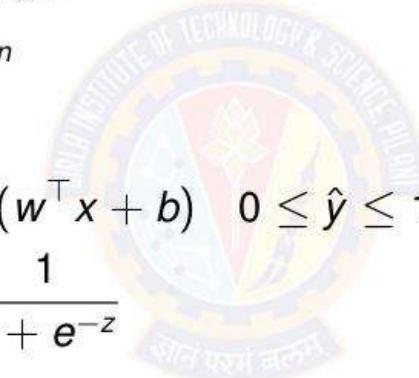
Parameters $w \in \mathbb{R}^n$

$b \in \mathbb{R}$

Output $\hat{y} = \sigma(w^\top x + b) \quad 0 \leq \hat{y} \leq 1$

Activation function $\sigma(z) = \frac{1}{1 + e^{-z}}$
 $= \begin{cases} 1 & \text{if } z \text{ is large positive} \\ 0 & \text{if } z \text{ is large negative} \end{cases}$

Activation $z = w^\top x + b$



Neural Net : Cost Function - Recap

(Same as Logistic Regression!!!!)

Loss function $\mathcal{L}(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$

If $y = 1$ $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$

$\mathcal{L} \approx 0 \implies \log \hat{y} \approx \text{large} \implies \hat{y} \approx \text{large}$

If $y = 0$ $\mathcal{L}(\hat{y}, y) = -\log(1 - \hat{y})$

$\mathcal{L} \approx 0 \implies \log(1 - \hat{y}) \approx \text{small} \implies \hat{y} \approx \text{small}$

m training examples = $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots (x^{(m)}, y^{(m)})\}$

$$\begin{aligned} \text{Cost function } J(w, b) &= \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^m -[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \end{aligned}$$

Neural Net : Learning using Gradient Descent- Recap (Same as Logistic Regression!!!!)

To Learn parameters use Gradient Descent Algorithm

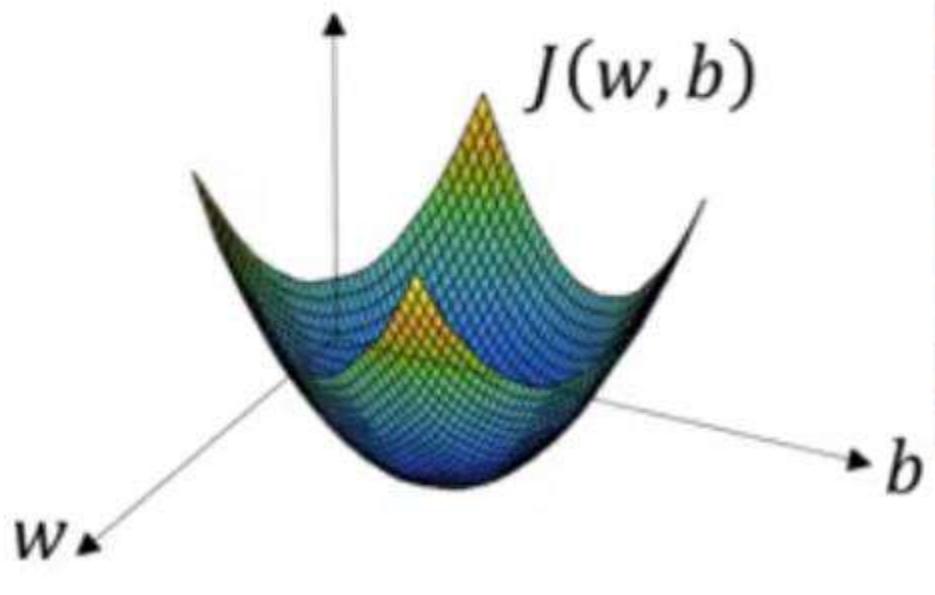
$$\hat{y} = \sigma(w^\top x + b) \quad \text{where } \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m - [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Find w, b that minimize $J(w, b)$.

Neural Net : Learning using Gradient Descent- Recap (Same as Linear & Logistic Regression!!!!)

Find w, b that minimize $J(w, b)$.



Repeat {
}

$$w := w - \alpha \frac{\partial J(w, b)}{\partial w}$$

$$b := b - \alpha \frac{\partial J(w, b)}{\partial b}$$

Neural Net : Learning using Gradient Descent- Recap (Same as Linear Regression!!!!)

To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1x_1 + \cdots + w_nx_n$$

Let's learn w_i 's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Where D is set of training examples

Neural Net : Learning using Gradient Descent- Recap (Same as Linear Regression!!!!)

Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\ \frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d) (-x_{i,d})\end{aligned}$$

- t : target value
- o : current prediction
- η : learning rate. small value.

GRADIENT-DESCENT(*training_examples*, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where \vec{x} is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
 - Initialize each Δw_i to zero.
 - For each $\langle \vec{x}, t \rangle$ in *training_examples*, Do
 - * Input the instance \vec{x} to the unit and compute the output o
 - * For each linear unit weight w_i , Do
$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

– For each linear unit weight w_i , Do

$$w_i \leftarrow w_i + \Delta w_i$$

Neural Net : Learning using Gradient Descent- Recap (Same as Linear Regression!!!!)

Batch mode Gradient Descent:

Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
 2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$
-

Incremental mode Gradient Descent:

Do until satisfied

- For each training example d in D
 1. Compute the gradient $\nabla E_d[\vec{w}]$
 2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate
Batch Gradient Descent arbitrarily closely if η
made small enough

NN – Learning of Weights & Bias:

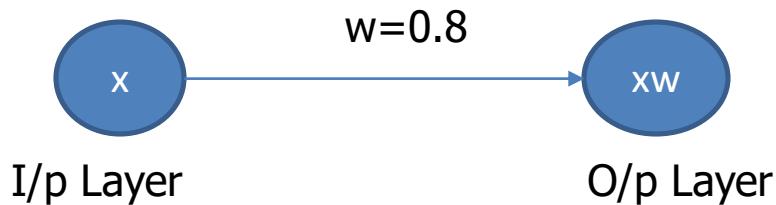
Sample Problem – Forward Backward Propagation

Intuition:

Without transformation function, bias or multiple neurons

Take only one training example

Randomly assign weights like in Gradient Descent



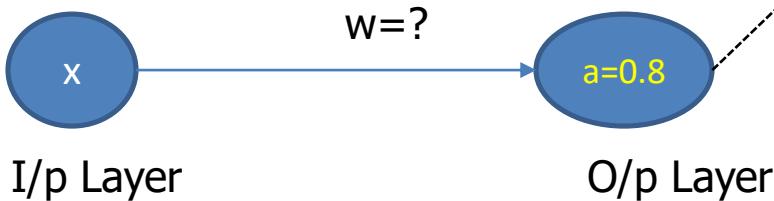
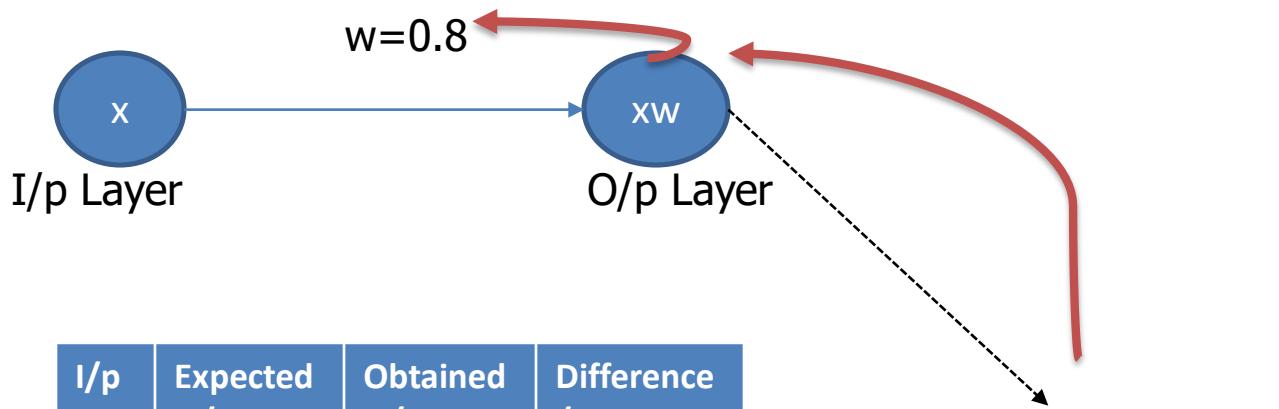
I/p x	Expected O/p y	Obtained O/p $a=xw$	Difference /Error (y-a)
2	0.8	1.6	-0.8

NN – Learning of Weights & Bias:

Sample Problem – Forward Backward Propagation

Intuition:

- ✓ How should "a" change for the Error to decrease?
- ✓ How should "w" change so that it influences a change in "a" to reduce Error?

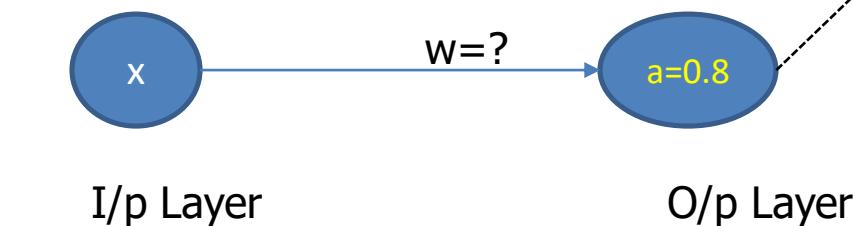
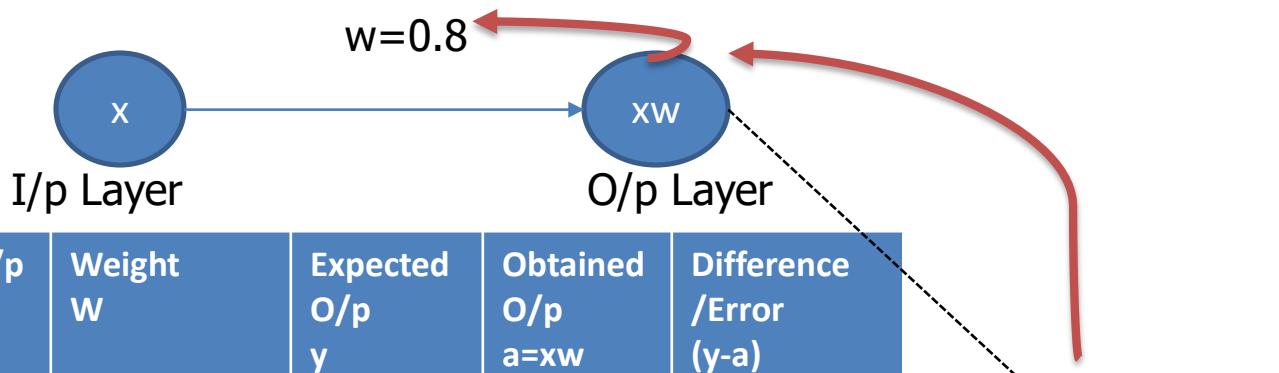


NN – Learning of Weights & Bias:

Sample Problem – Forward Backward Propagation

Intuition:

- ✓ Chain Rule of Differentiation: $\frac{\partial E}{\partial w} = \frac{\partial E}{\partial a} * \frac{\partial a}{\partial w} = (a - y) * x$
- ✓ Change the weights : $w_{\text{new}} = w_{\text{old}} - \text{LearningRate} * \frac{\partial E}{\partial w}$



Neural Net : Back Propagation Algorithm

Initialize all weights to small random numbers.

Until satisfied, Do

- For each training example, Do
 1. Input the training example to the network and compute the network outputs
 2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{h,k} \delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

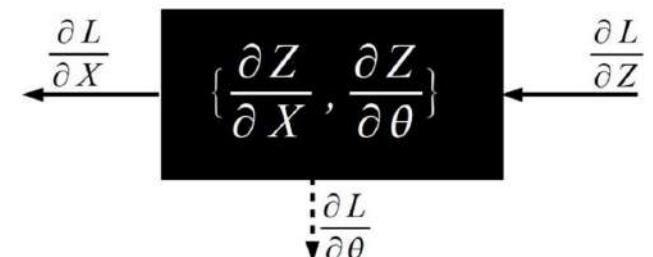
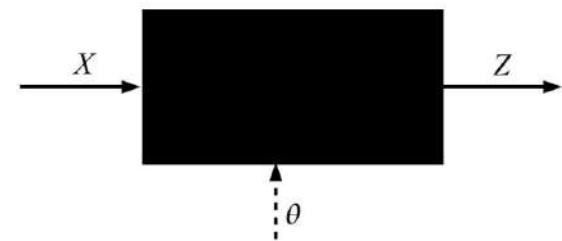
Neural Net : Back Propagation Algorithm

- Gradient descent over entire *network* weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum
 - In practice, often works well (can run multiple times)
- Often include weight *momentum* α
$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n - 1)$$
- Minimizes error over *training* examples
 - Will it generalize well to subsequent examples?
- Training can take thousands of iterations → slow!
- Using network after training is very fast

Neural Net : Classification - Summary

1. Represent the problem as a neural network.
 - One input neuron for each feature in the dataset.
2. Compute the values of Activation (z) and Hypothesis (a) using Forward propagation.
 - Apply sigmoid activation function for binary classification.
3. Compute the error for all training examples as Cost function.
4. Propagate the error back using back propagation algorithm.
5. Use gradient descent to learn change in the weights.
6. Update the weights simultaneously.

Key Computation: Forward-Prop

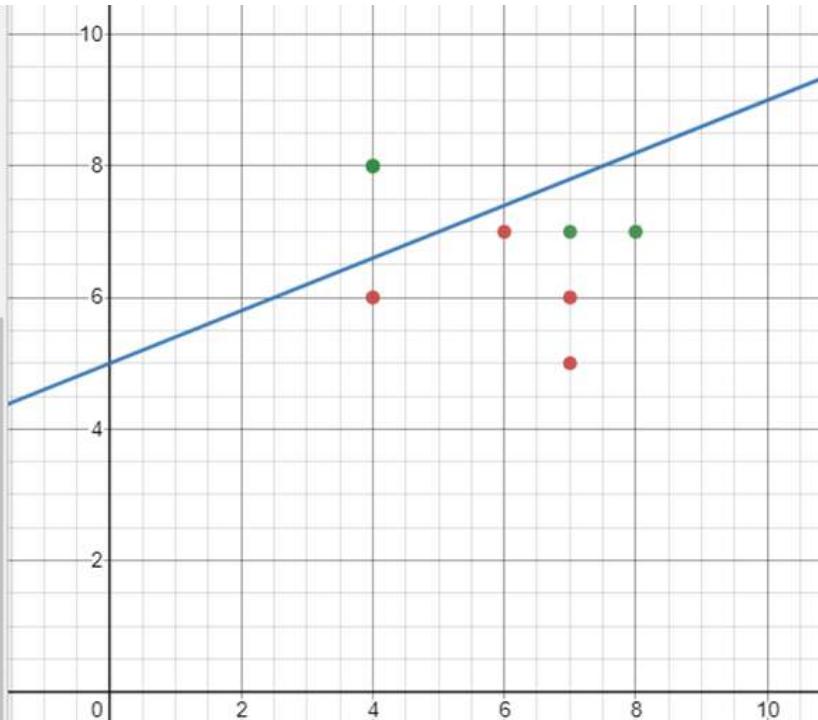




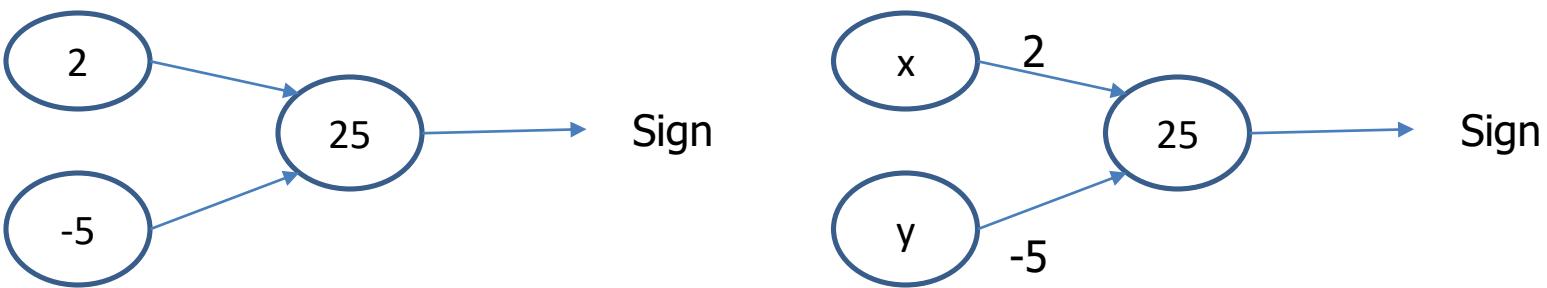
Deep Neural Networks

Multilayer Perceptron- Intuition

5		(4,6)	<input type="checkbox"/> Label
6		(4,8)	<input type="checkbox"/> Label
7		(6,7)	<input type="checkbox"/> Label
8		(7,6)	<input type="checkbox"/> Label
9		$2x - 5y + 25 = 0$	<input type="checkbox"/>
10		$-x - 2y + 20.5 = 0$	<input type="checkbox"/>



Technical Interview	HR Discussion	Selected in Interview?
8	7	Y
7	7	Y
7	5	N
4	8	Y
4	6	N
4	8	Y
6	7	N
7	6	N



Perceptron & Delta Rule

input vector: $x = [x_0 \ x_1 \ x_2 \ x_3 \dots \ x_n]$

Neuron Sums the weighted Inputs

Dot Product of Coefficient & Input:

wx

$$= w_0x_0 + w_1x_1 + \dots + w_nx_n$$

$$= \sum_{i=0}^n w_i x_i = WX = a$$

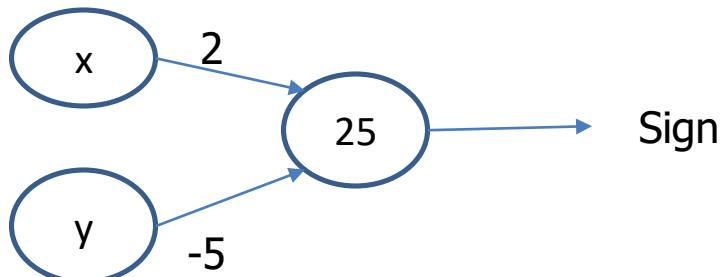
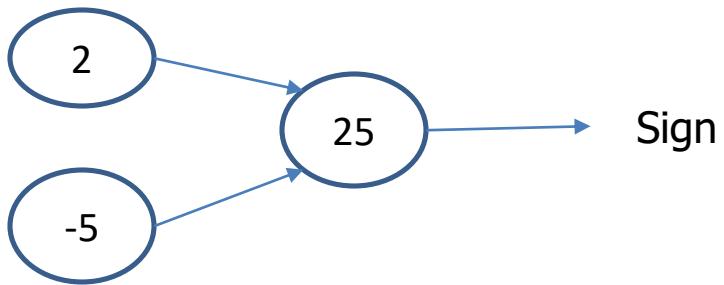
Neuron Threshold Activation Function

$$y = f(wx)$$

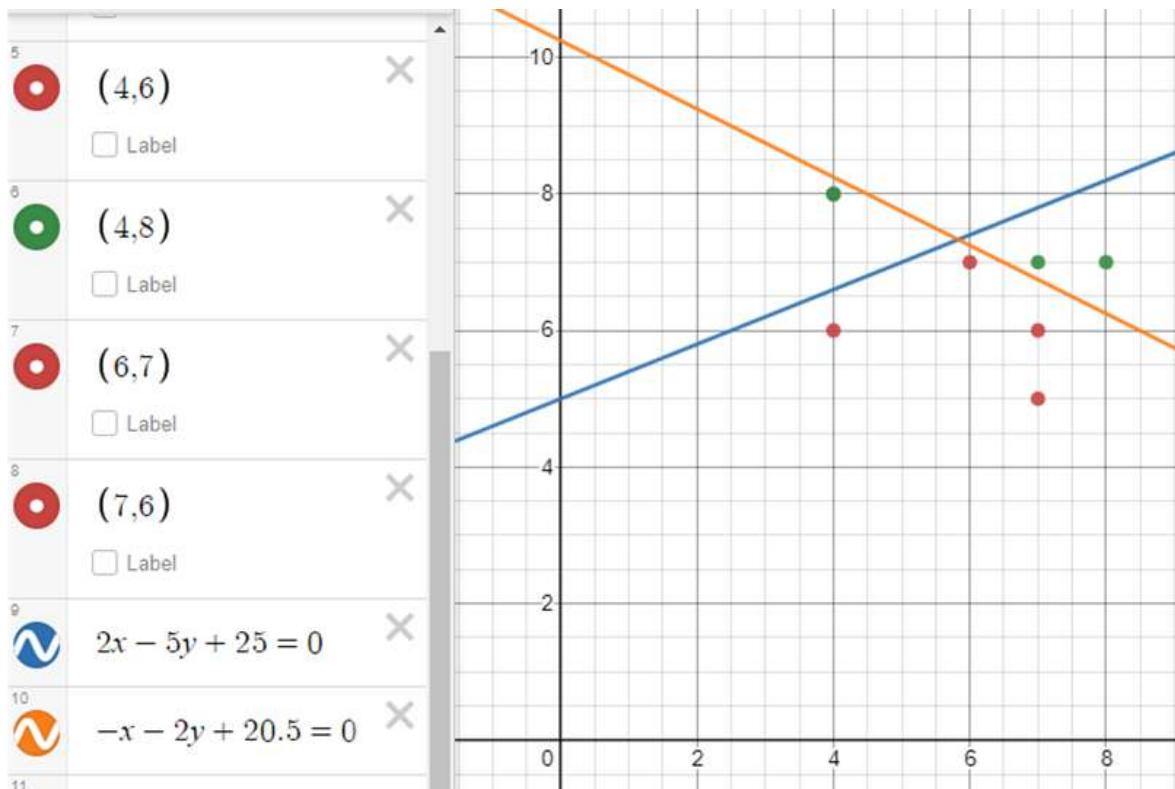
$$f(wx) = +1 \quad \text{if } wx > 0$$

$$f(wx) = -1 \quad \text{if } wx \leq 0$$

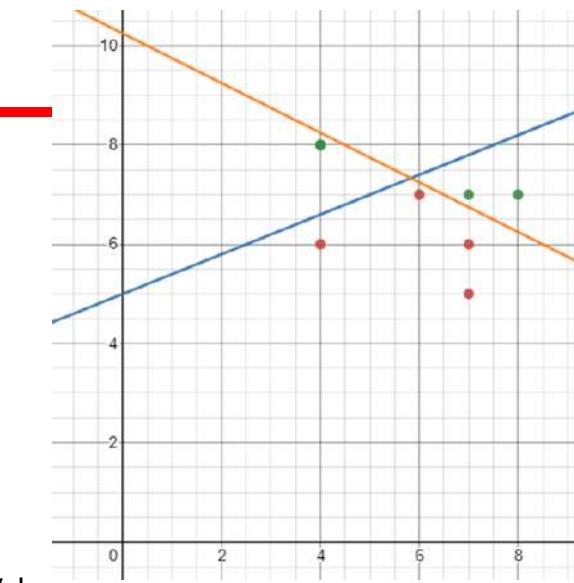
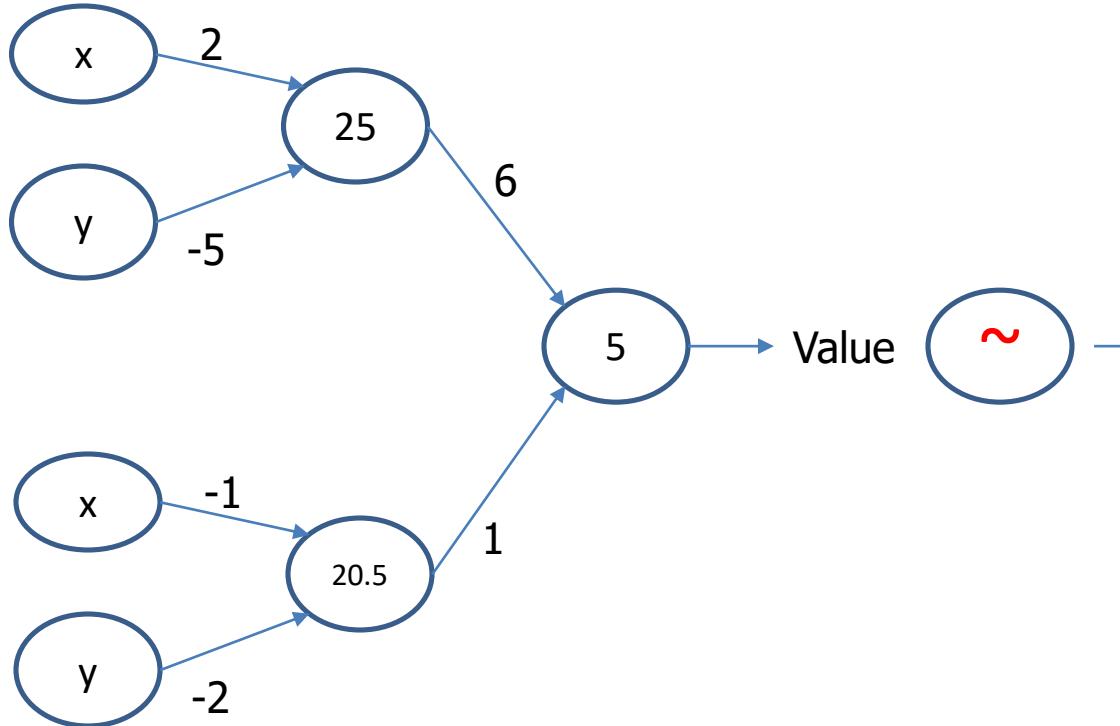
Technical Interview	HR Discussion	Selected in Interview?
8	7	Y
7	7	Y
7	5	N
4	8	Y
4	6	N
4	8	Y
6	7	N
7	6	N



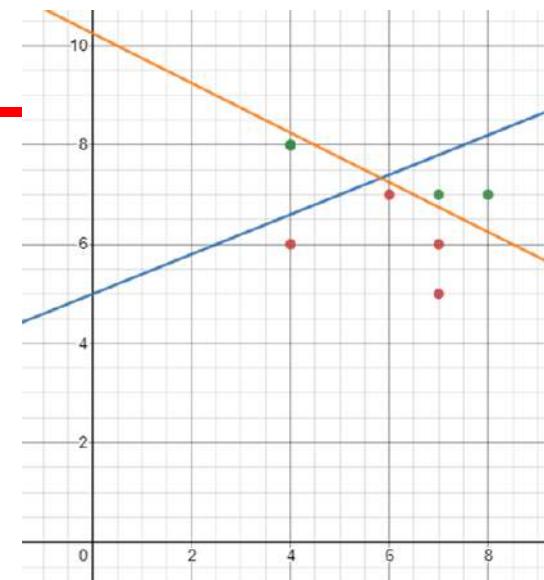
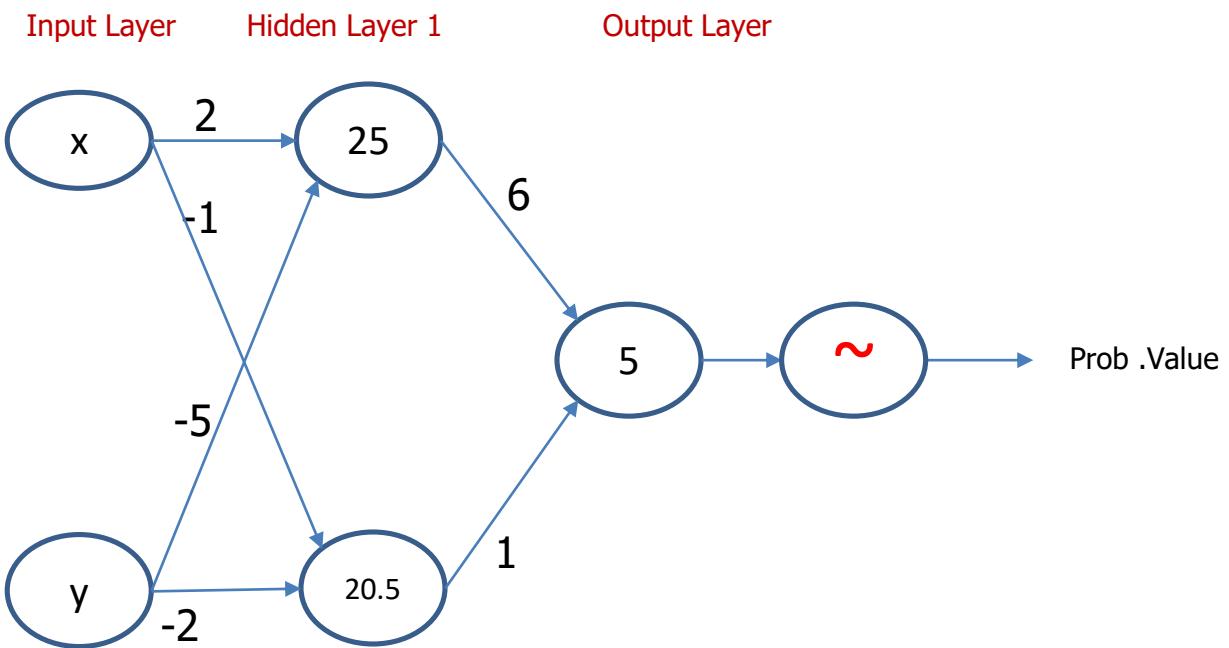
Multilayer Perceptron- Intuition



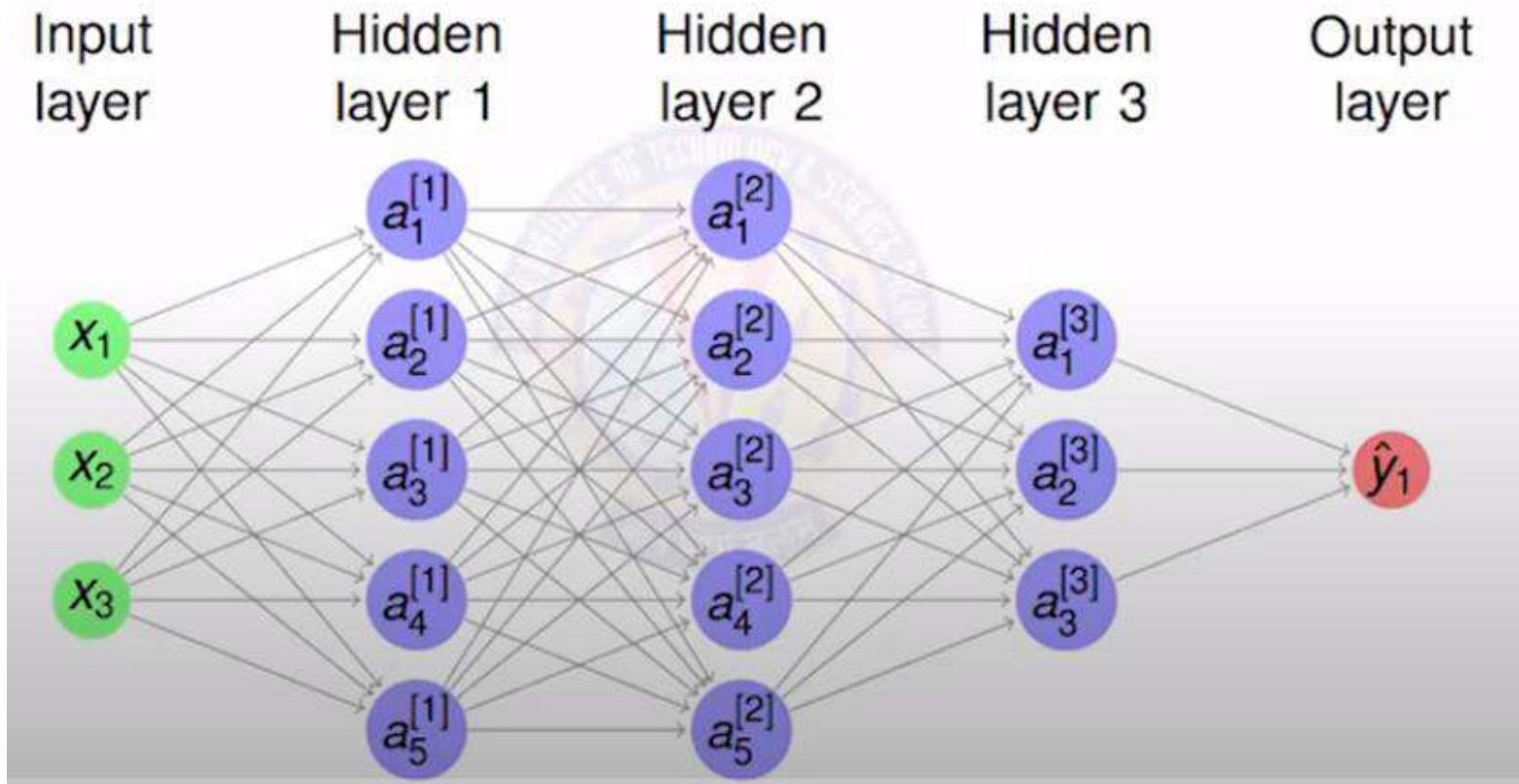
Multilayer Perceptron- Intuition



Multilayer Perceptron- Intuition

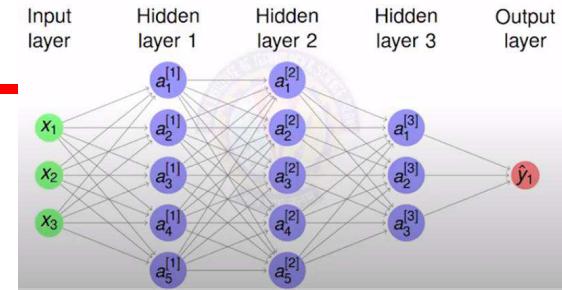


Multi-Layer Perceptron / Deep Learning Network



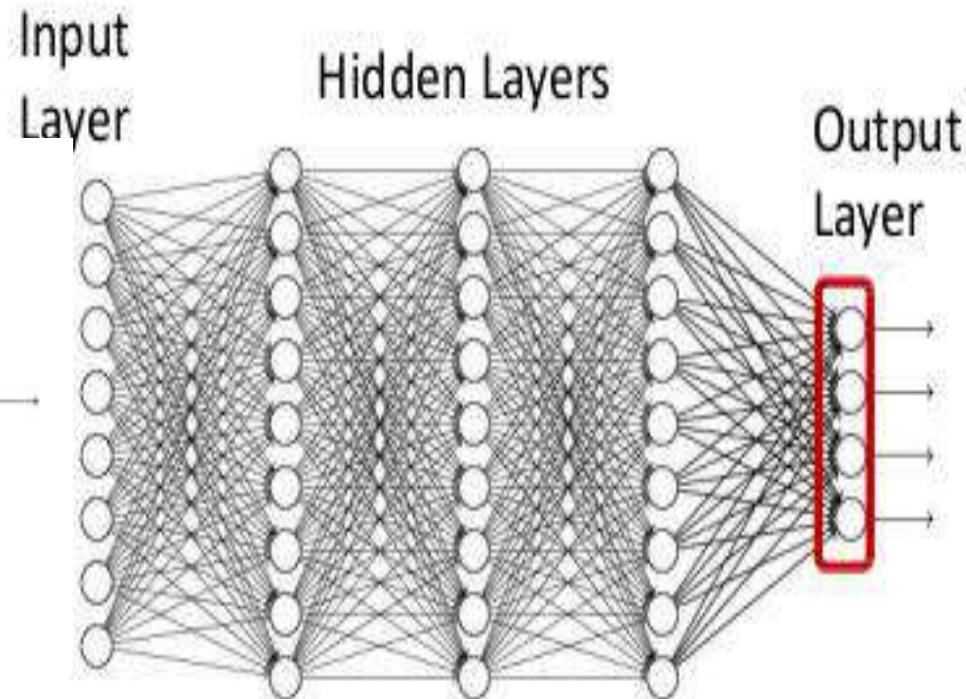
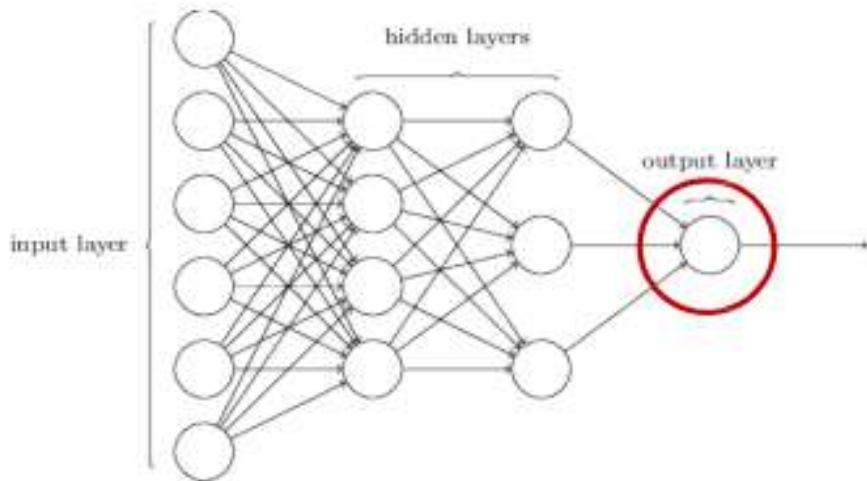
Multi-Layer Perceptron / Deep Learning Network

- Design the architecture of the network.
 - Input representation
 - Output representation
 - Decide the number of layers, number of neurons in each layer.
- Choose the activation functions that will be used on each hidden layer.
- Choose the loss function.
- Choose the optimizer algorithm.
- Train a feedforward network.
- Evaluate the performance of the network.



Multi-Layer Perceptron / Deep Learning Network

- Design the architecture of the network.
 - Input representation**
 - Output representation**
 - Decide the number of layers, number of neurons in each layer.

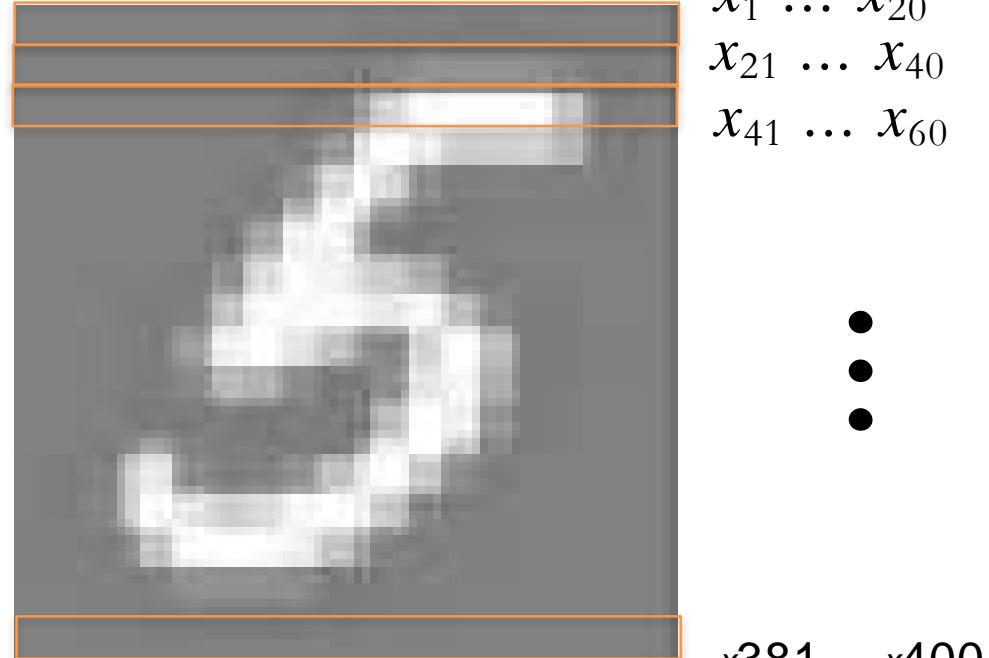


Layering Representations

7	9	6	5	8	7	4	4	1	8
0	7	3	3	2	4	8	4	5	1
6	6	3	2	9	1	3	3	2	6
1	3	7	1	5	6	5	2	4	4
7	0	9	8	7	5	8	9	5	4
4	6	6	5	0	2	1	3	6	9
8	5	1	8	9	3	8	7	3	6
1	0	2	8	2	3	0	5	1	5
6	7	8	2	5	3	9	7	0	0
7	9	3	9	8	5	7	2	9	8

20 × 20 pixel images

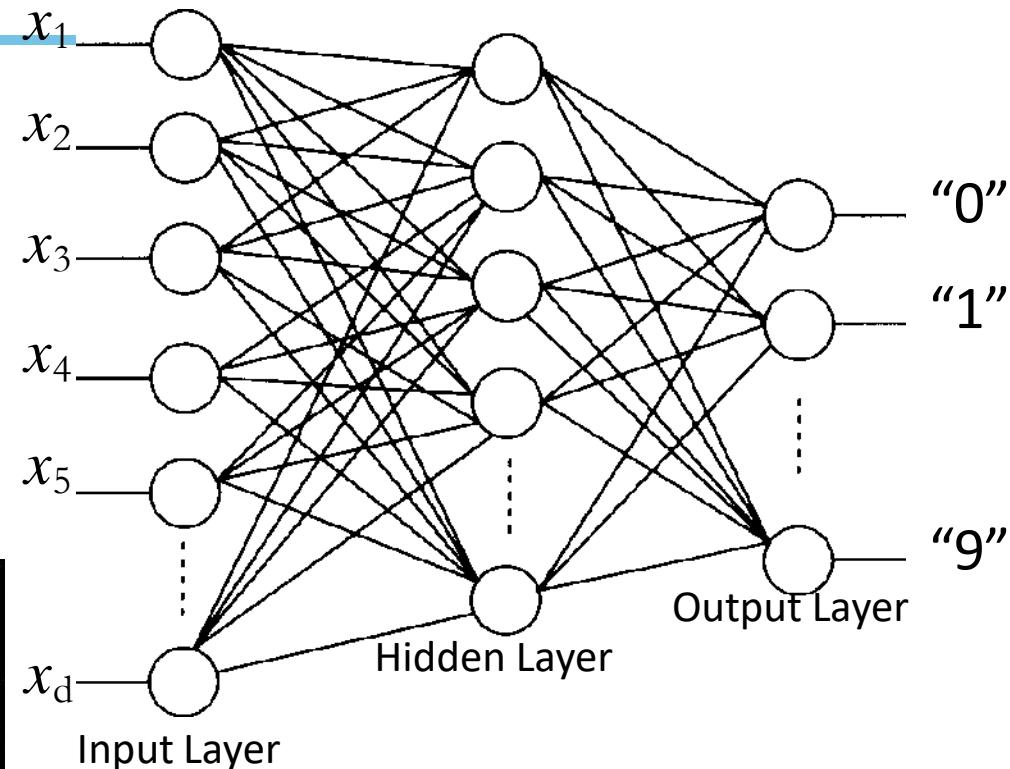
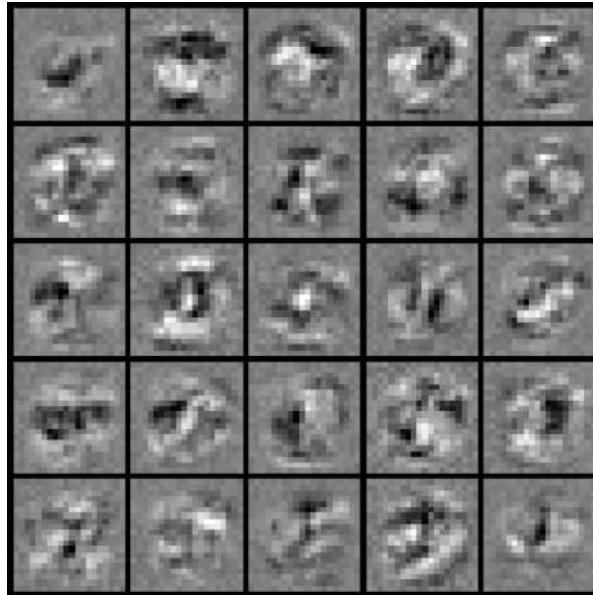
$d = 400$ 10 classes



Each image is “unrolled” into a vector \mathbf{x} of pixel intensities

Layering Representations

7	9	6	5	8	7	4	4	1	8
0	7	3	3	2	4	8	4	5	1
6	6	3	2	9	2	3	3	2	6
1	3	7	1	5	6	5	2	4	4
7	0	9	8	7	5	8	9	5	4
4	6	6	5	0	2	1	3	6	9
8	5	1	8	9	3	8	7	3	6
1	0	2	8	2	3	0	5	1	5
6	7	8	2	5	3	9	7	0	0
7	9	3	9	8	5	7	2	9	8



Visualization of
Hidden Layer

Multiple Output Units: One-vs-Rest



Pedestrian



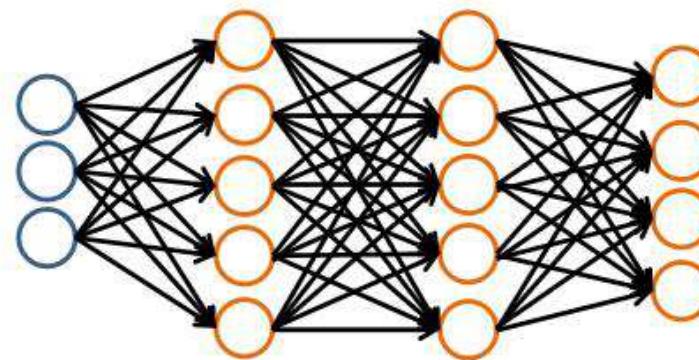
Car



Motorcycle



Truck



$$h_{\Theta}(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

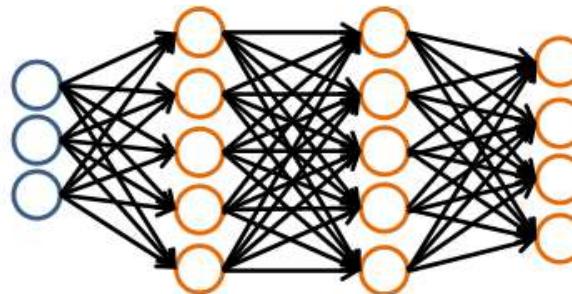
$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Multiple Output Units: One-vs-Rest



$$h_{\Theta}(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{when pedestrian}$$

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{when car}$$

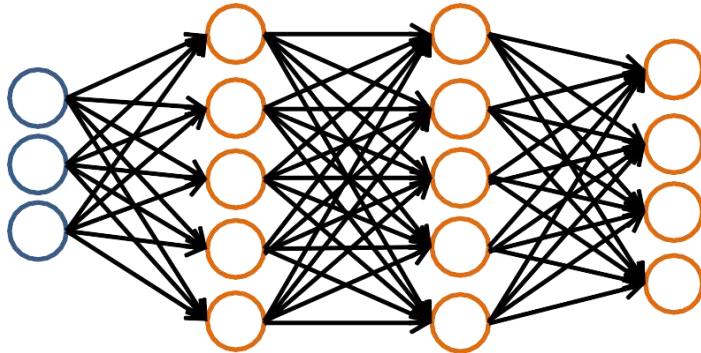
$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{when motorcycle}$$

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{when truck}$$

- Given $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$
- Must convert labels to 1-of- K representation

– e.g., $y_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle, $y_i = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car, etc.

Neural Network Classification



Given:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

$s \in \mathbb{N}^{+L}$ contains # nodes at each layer

- $s_0 = d$ (# features)

Binary classification

$$y = 0 \text{ or } 1$$

1 output unit ($s_{L-1} = 1$)

Multi-class classification (K classes)

$$y \in \mathbb{R}^K \quad \text{e.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

K output units ($s_{L-1} = K$)

pedestrian

car

motorcycle

truck

Activation functions



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Deep Neural Network - Training



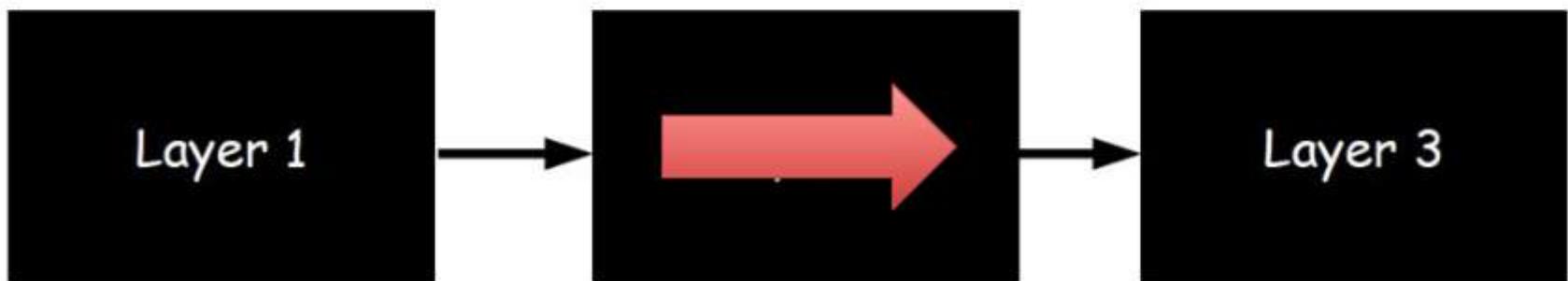
- Step 1: Compute Loss on mini-batch [F-Pass]



Deep Neural Network - Training



- Step 1: Compute Loss on mini-batch [F-Pass]



Deep Neural Network - Training

- Step 1: Compute Loss on mini-batch [F-Pass]



Deep Neural Network - Training



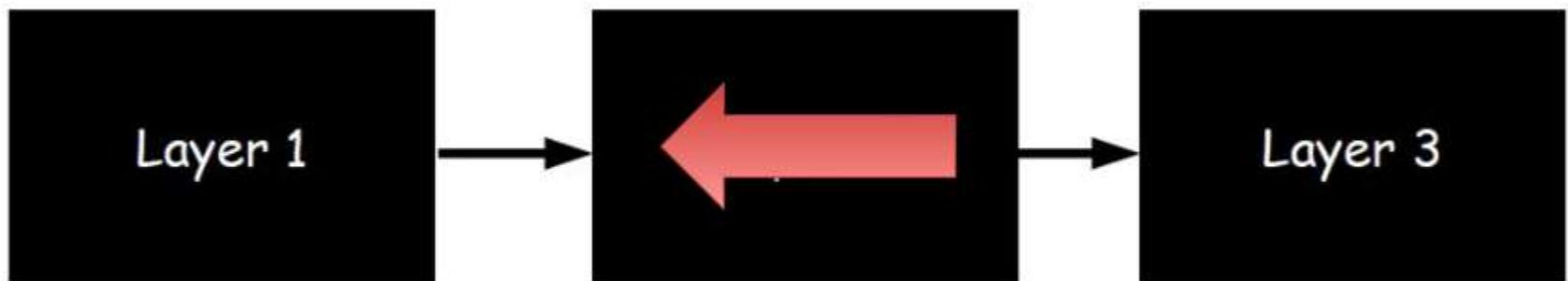
- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t. parameters [B-Pass]



Deep Neural Network - Training



- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t. parameters [B-Pass]



Deep Neural Network - Training



- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t. parameters [B-Pass]



Deep Neural Network - Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t. parameters [B-Pass]
- Step 3: Use gradient to update parameters



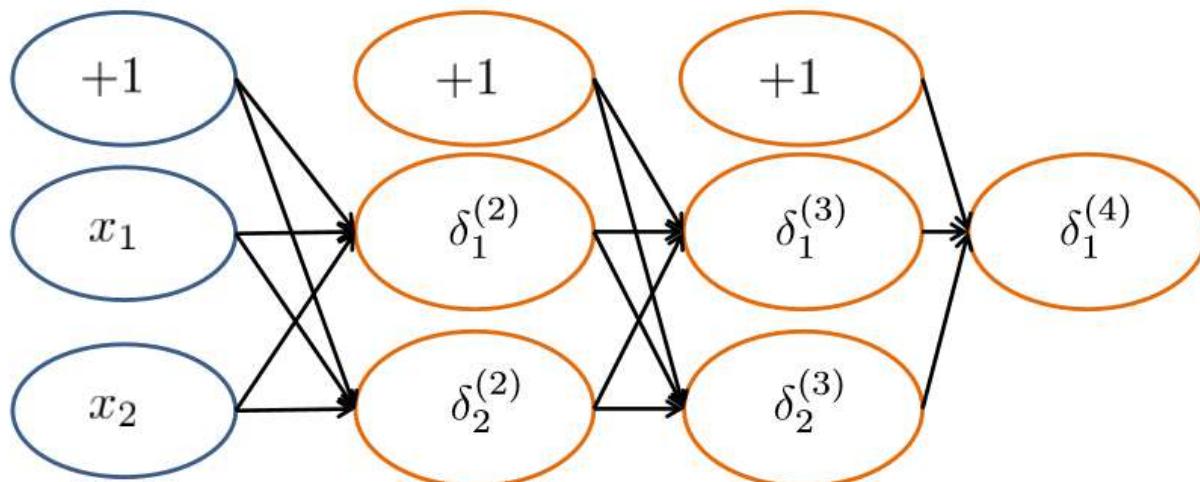
$$\theta \leftarrow \theta - n \frac{dL}{d\theta}$$



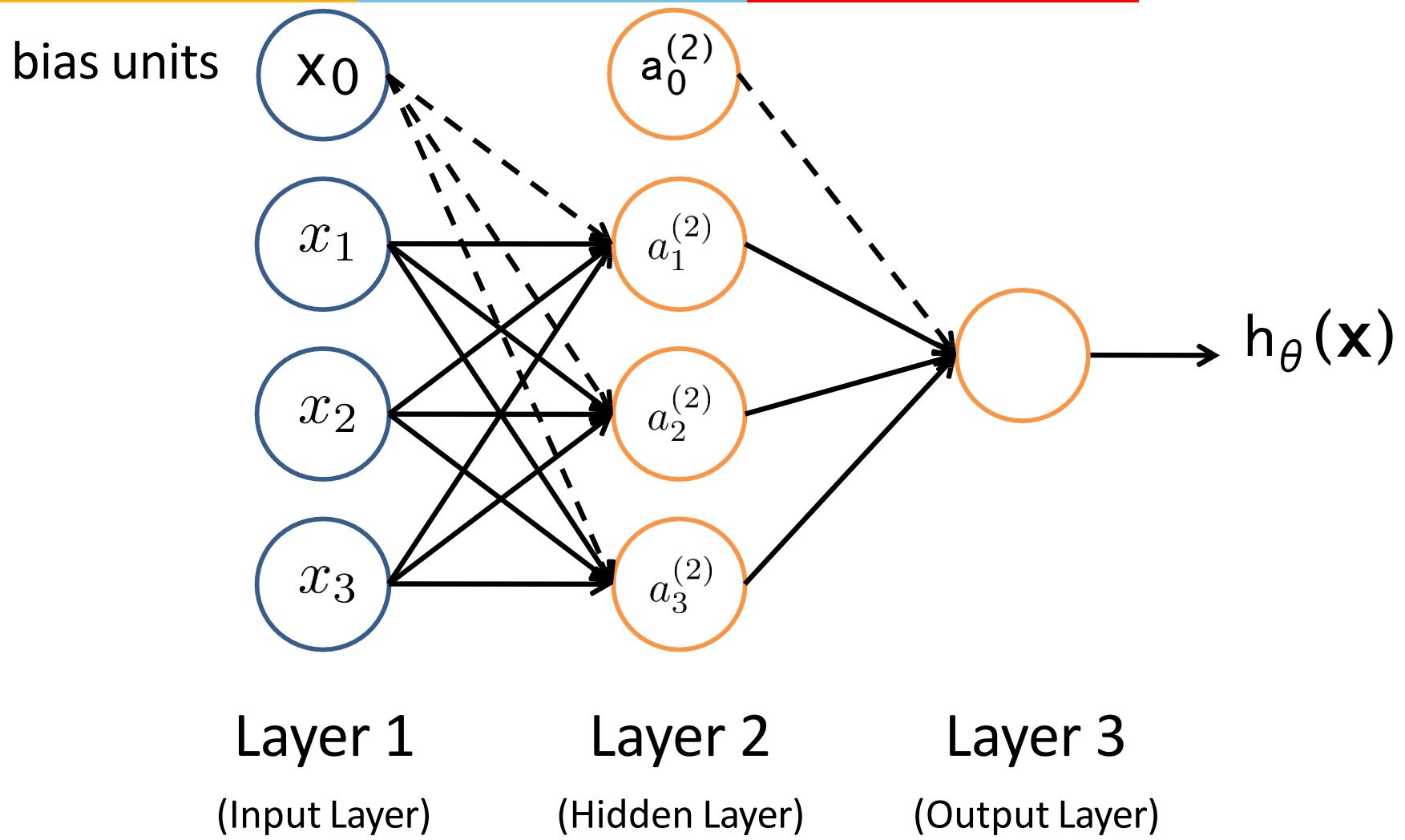
Training DNN & Notations

Random Initialization

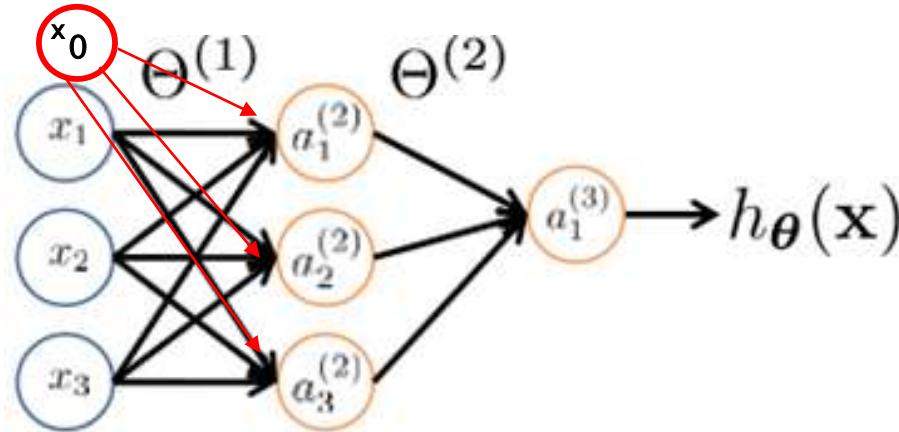
- Important to randomize initial weight matrices
- Can't have uniform initial weights, as in logistic regression
 - Otherwise, all updates will be identical & the net won't learn



Notations : Recap



Forward Propagation



$a_i^{(j)}$ = “activation” of unit i in layer j

$\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$,
then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

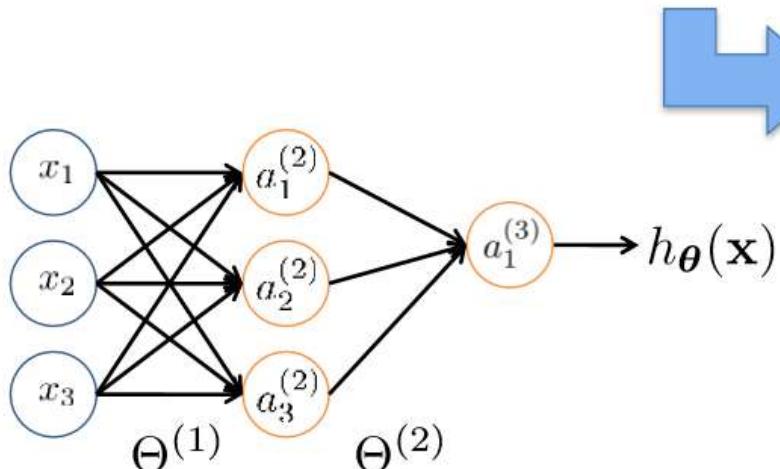
Vectorization

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left(z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left(z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left(z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left(z_1^{(3)} \right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

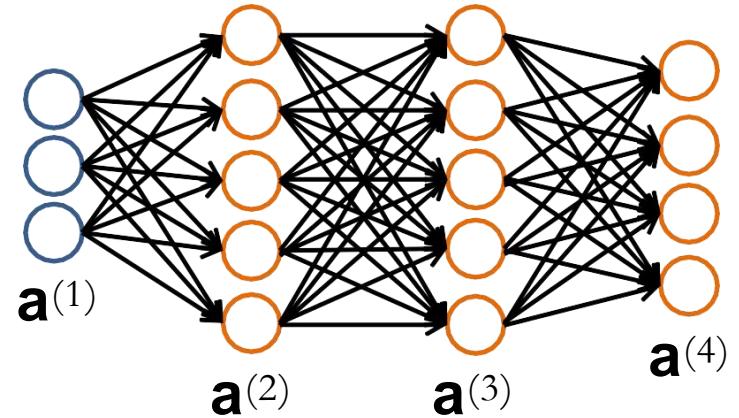
$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Forward Propagation : Notation

- Given one labeled training instance (\mathbf{x}, y) :

Forward Propagation

- $\mathbf{a}^{(1)} = \mathbf{x}$
- $\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{a}^{(1)}$
- $\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$ [add $a_0^{(2)}$]
- $\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$
- $\mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$ [add $a_0^{(3)}$]
- $\mathbf{z}^{(4)} = \Theta^{(3)}\mathbf{a}^{(3)}$
- $\mathbf{a}^{(4)} = h_{\Theta}(\mathbf{x}) = g(\mathbf{z}^{(4)})$



Based on slide by Andrew Ng

Thank you !

Required Reading for completed session :

T1 - Chapter # 4 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 5 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

Next Class Plan

- Back propagation in Multilayer Perceptrons
- Example Numerical Problem
- Overview of CNN & RNN



Machine Learning

DSE CLZG565

Neural Network

Raja vadhana P

Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: “Probabilistic Machine Learning, An Introduction”, Kevin P. Murphy, From BITS Pilani Slides : Prof. Chetana, Prof. Sugata, Prof. Monali, Prof. Anita, Prof. Raja vadhana , Prof.Seetha, CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.



Neural Networks

Neural Net : Notations & Terminologies

Given X find $\hat{y} = P(y = 1 | X)$

Input $X \in \mathbb{R}^{n \times m}$

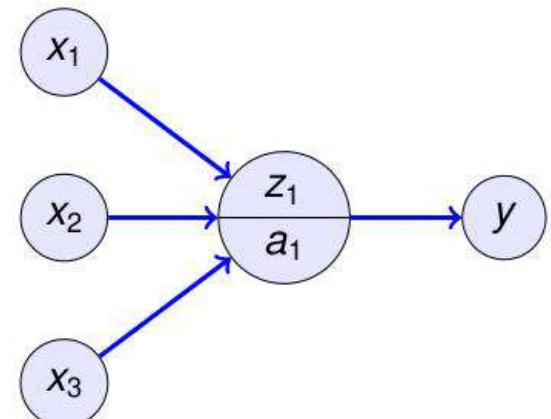
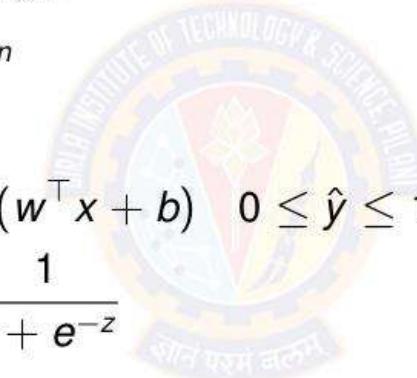
Parameters $w \in \mathbb{R}^n$

$b \in \mathbb{R}$

Output $\hat{y} = \sigma(w^\top x + b) \quad 0 \leq \hat{y} \leq 1$

Activation function $\sigma(z) = \frac{1}{1 + e^{-z}}$
 $= \begin{cases} 1 & \text{if } z \text{ is large positive} \\ 0 & \text{if } z \text{ is large negative} \end{cases}$

Activation $z = w^\top x + b$



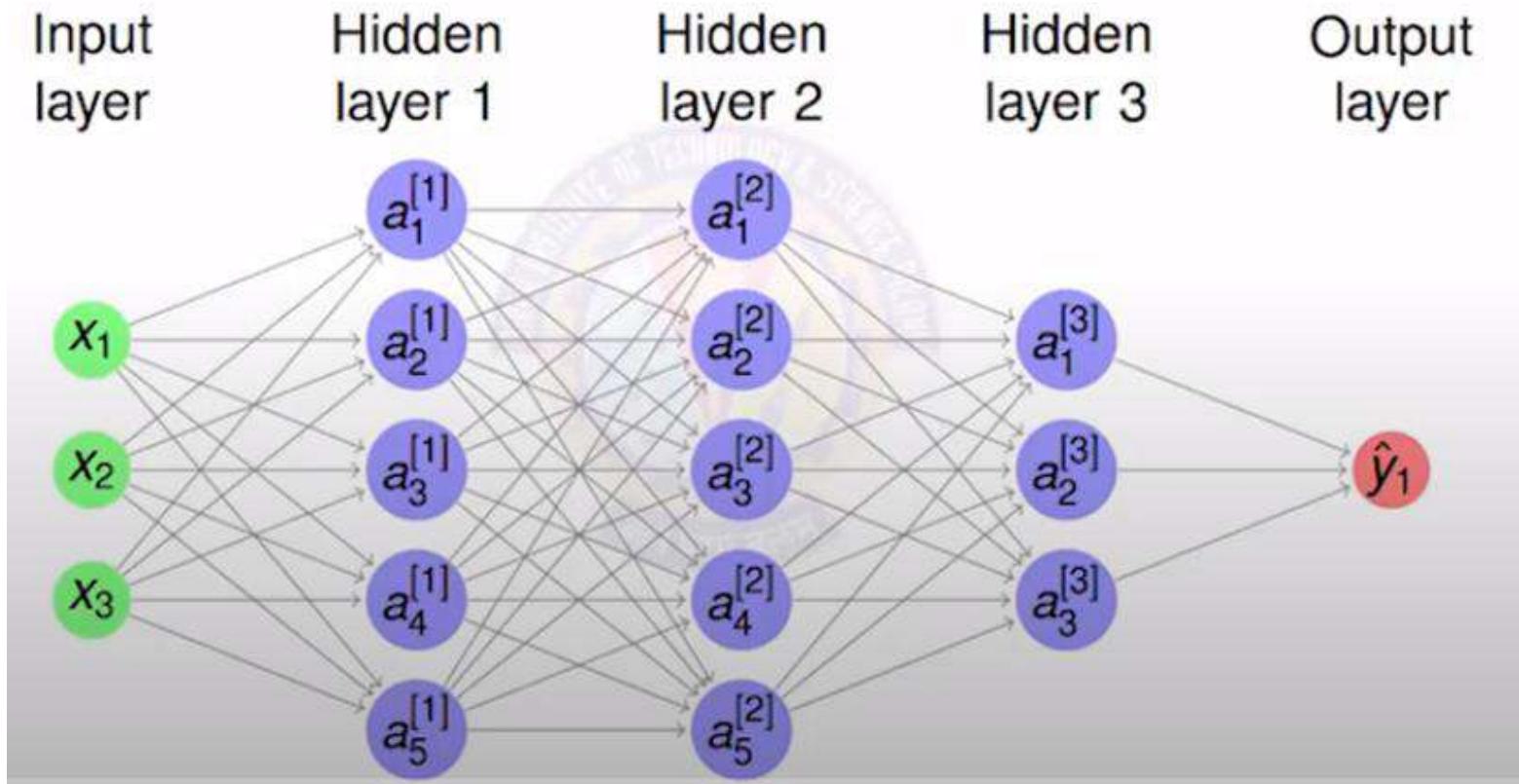
Neural Net : Classification - Summary

1. Represent the problem as a neural network.
 - One input neuron for each feature in the dataset.
2. Compute the values of Activation (z) and Hypothesis (a) using Forward propagation.
 - Apply sigmoid activation function for binary classification.
3. Compute the error for all training examples as Cost function.
4. Propagate the error back using back propagation algorithm.
5. Use gradient descent to learn change in the weights.
6. Update the weights simultaneously.



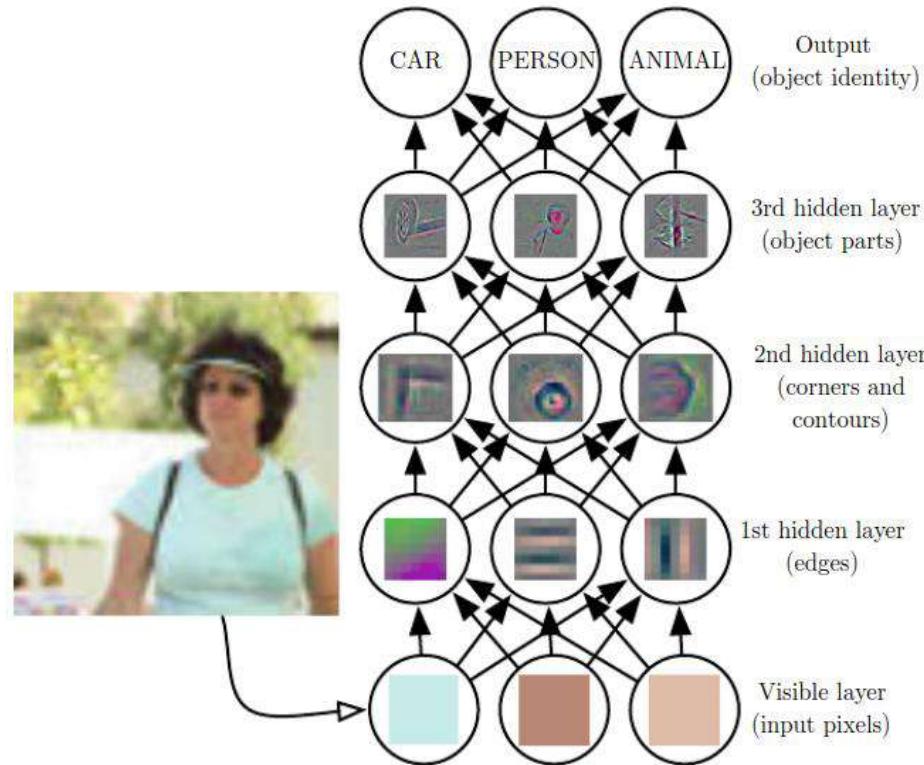
Deep Neural Networks

Multi-Layer Perceptron / Deep Learning Network



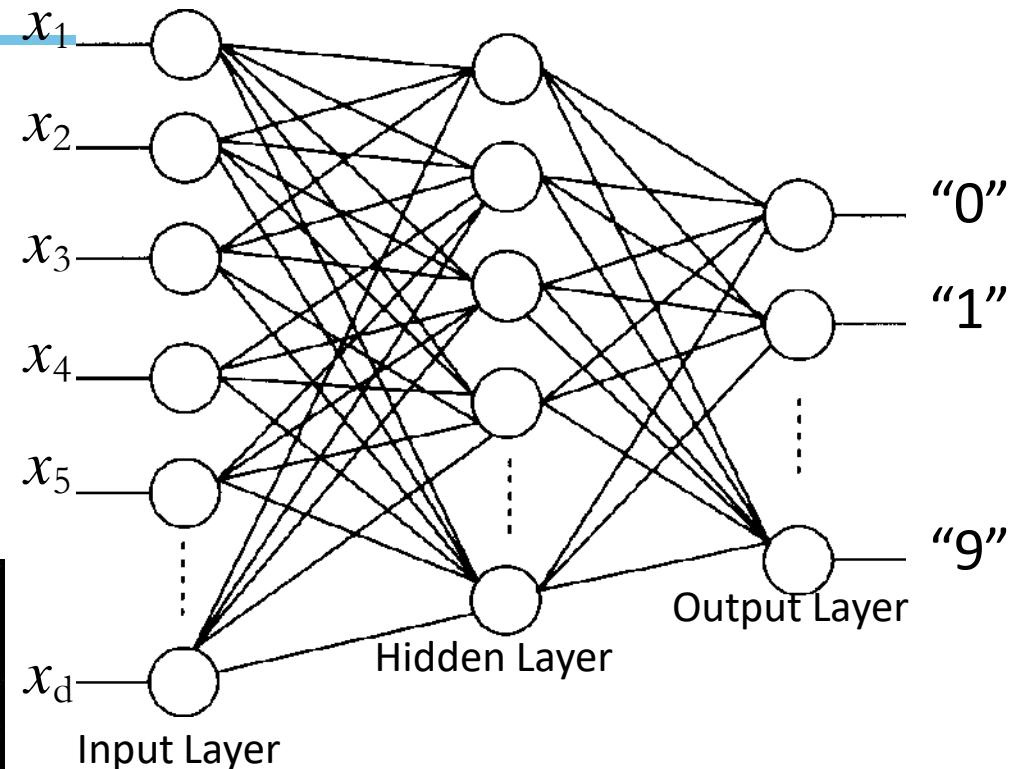
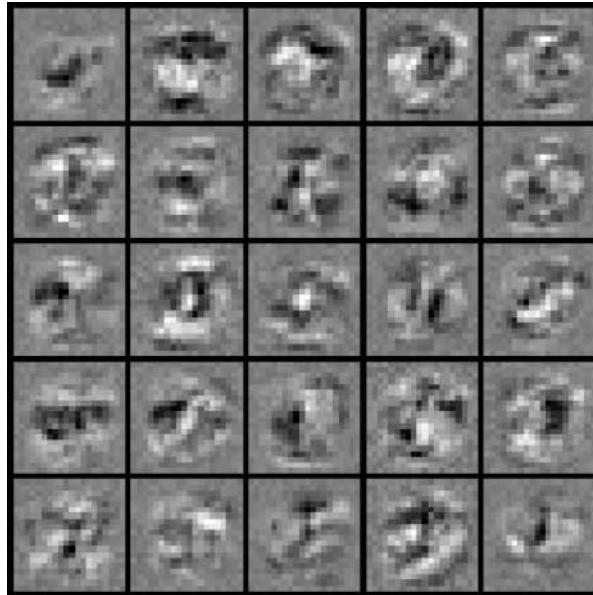
DL Builds Hierarchy of Features

Higher (or deeper) layers represents abstraction of the increasing complex features



Layering Representations

7	9	6	5	8	7	4	4	1	8
0	7	3	3	2	4	8	4	5	1
6	6	3	2	9	2	3	3	2	6
1	3	7	1	5	6	5	2	4	4
7	0	9	8	7	5	8	9	5	4
4	6	6	5	0	2	1	3	6	9
8	5	1	8	9	3	8	7	3	6
1	0	2	8	2	3	0	5	1	5
6	7	8	2	5	3	9	7	0	0
7	9	3	9	8	5	7	2	9	8



Visualization of
Hidden Layer

Multiple Output Units: One-vs-Rest



Pedestrian



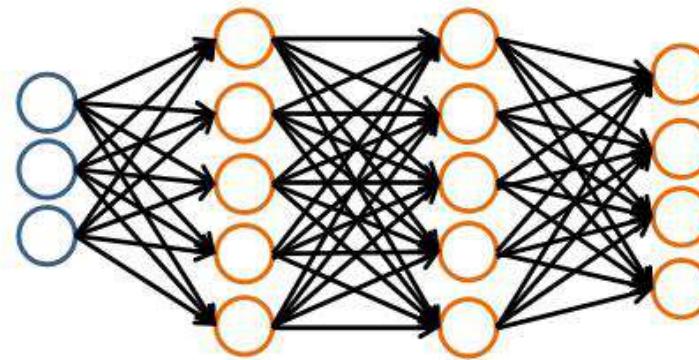
Car



Motorcycle



Truck



$$h_{\Theta}(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

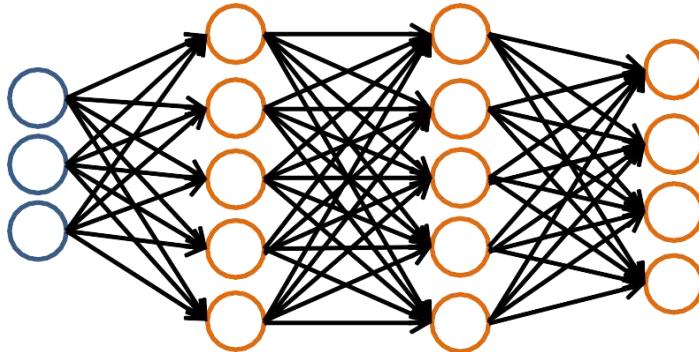
$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Neural Network Classification



Binary classification

$y = 0 \text{ or } 1$

1 output unit ($s_{L-1} = 1$)

Given:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

$s \in \mathbb{N}^{+L}$ contains # nodes at each layer

– $s_0 = d$ (# features)

Multi-class classification (K classes)

$$y \in \mathbb{R}^K \quad \text{e.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

K output units ($s_{L-1}=K$)
 pedestrian car
 motorcycle truck

Activation functions



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Deep Neural Network - Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t. parameters [B-Pass]
- Step 3: Use gradient to update parameters



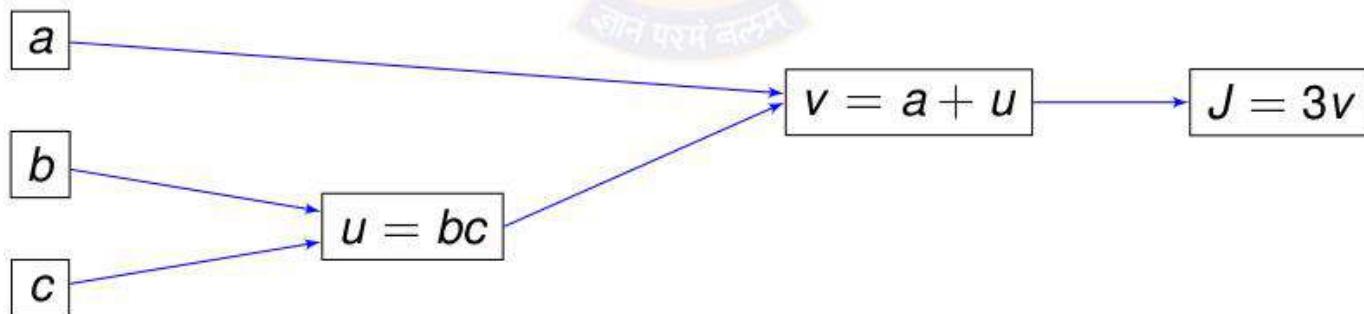
$$\theta \leftarrow \theta - n \frac{dL}{d\theta}$$

Let $J(a, b, c) = 3(a + bc)$

$$u = bc$$

$$v = a + u$$

Then $J = 3v$

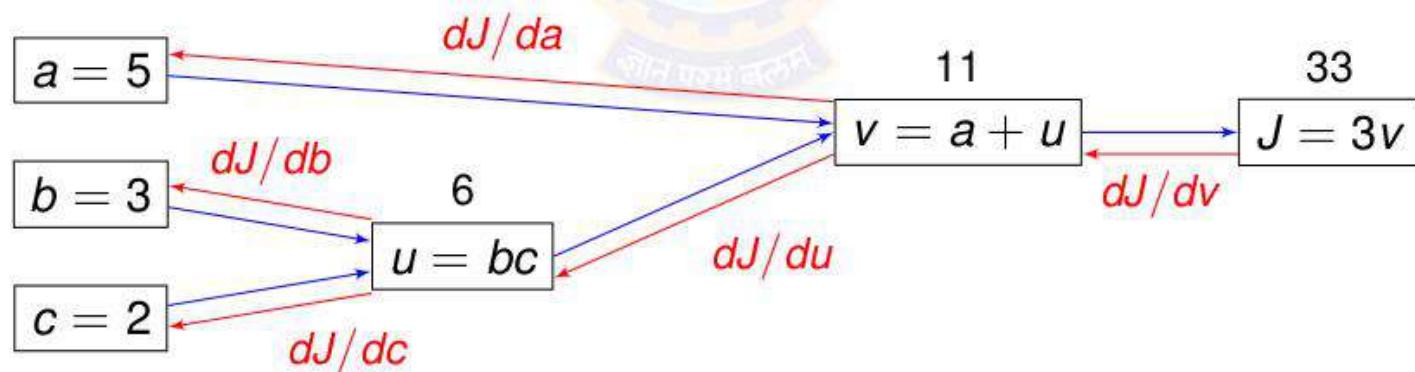


Let $J(a, b, c) = 3(a + bc)$

$$u = bc$$

$$v = a + u$$

Then $J = 3v$

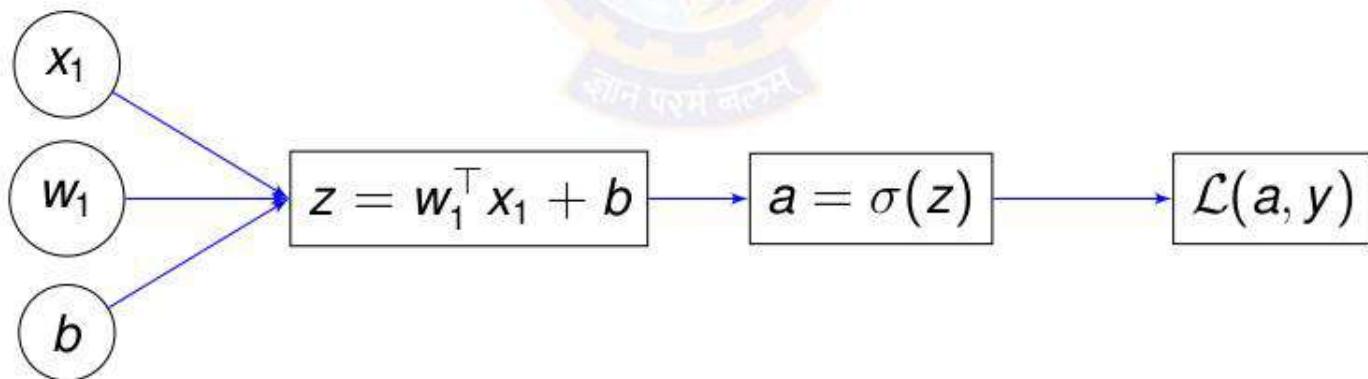


Binary Classification

$$z = w^\top X + b$$

$$a = \hat{y} = \sigma(z)$$

$$\mathcal{L}(a, y) = -[y \log a + (1 - y) \log(1 - a)]$$



Binary Classification

$$z = w^\top X + b$$

$$a = \hat{y} = \sigma(z)$$

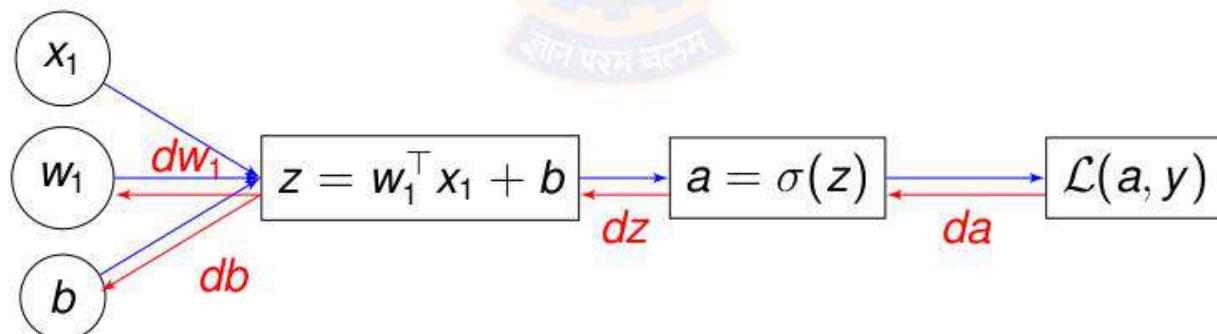
$$\mathcal{L}(a, y) = -[y \log a + (1 - y) \log(1 - a)]$$

$$da = \frac{d\mathcal{L}(a, y)}{da} = \frac{-y}{a} + \frac{1+y}{1+a}$$

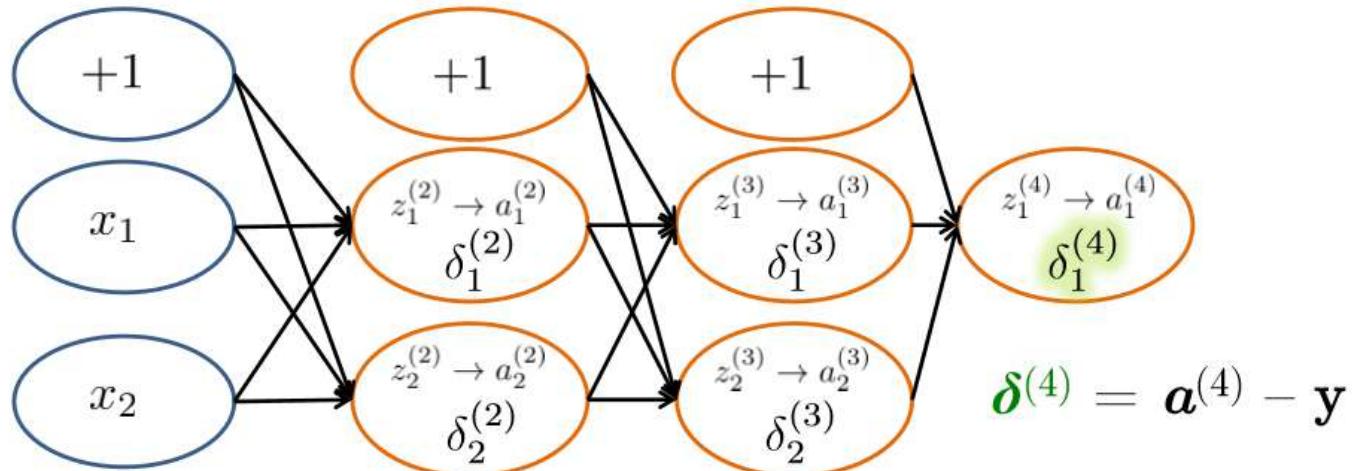
$$dz = \frac{d\mathcal{L}(a, y)}{dz} = a - y$$

$$dw_1 = \frac{d\mathcal{L}(a, y)}{dw_1} = x_1 dz$$

$$db = dz$$



Backpropagation Summary



$\delta_j^{(l)}$ = “error” of node j in layer l

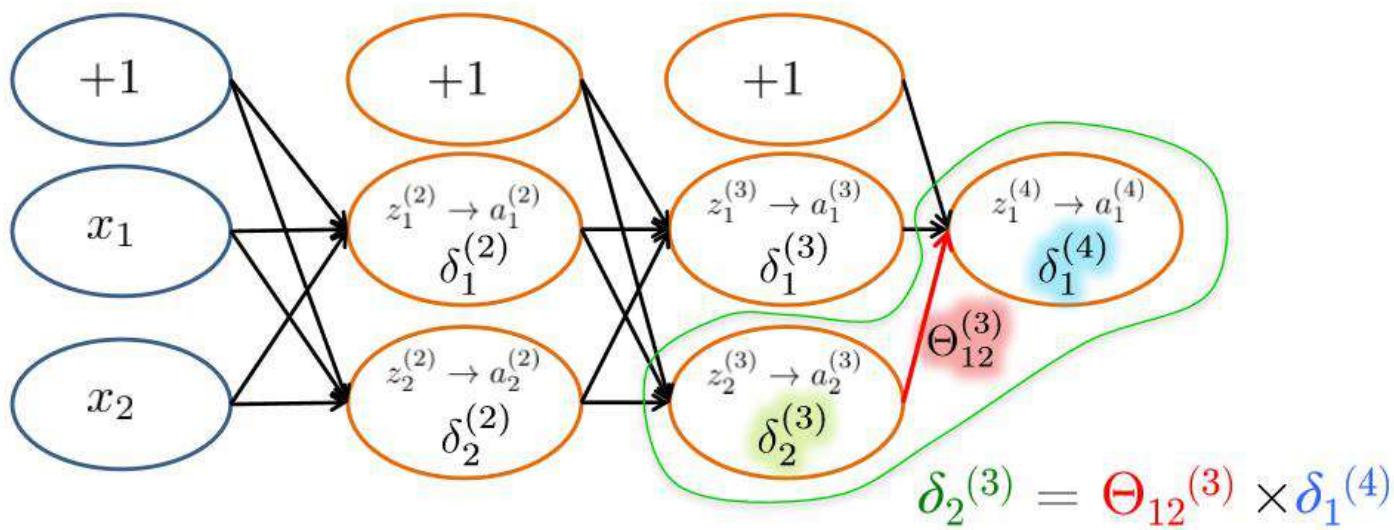
Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

Backpropagation Summary

innovate

achieve

lead

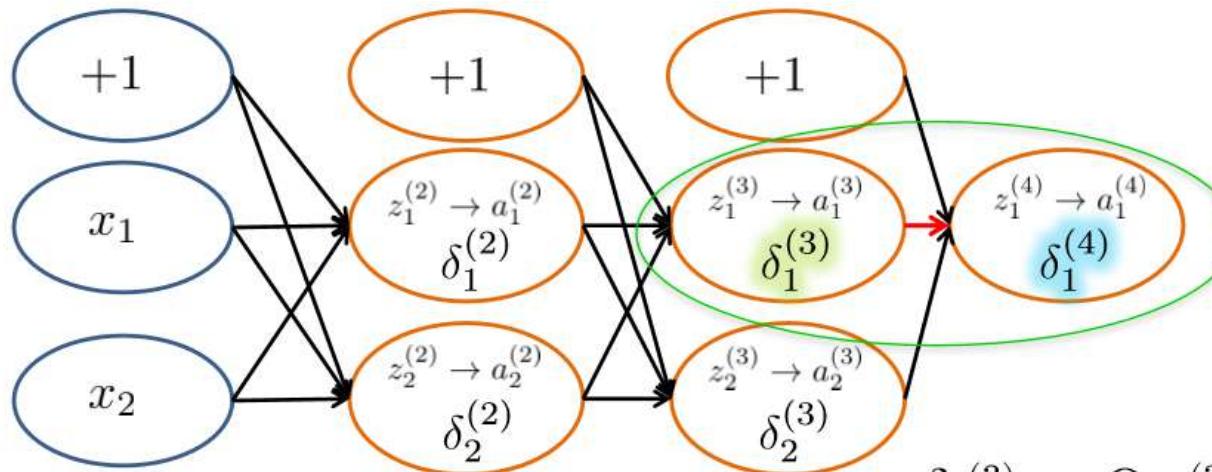


$\delta_j^{(l)}$ = “error” of node j in layer l

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

Backpropagation Summary



$$\delta_2^{(3)} = \Theta_{12}^{(3)} \times \delta_1^{(4)}$$

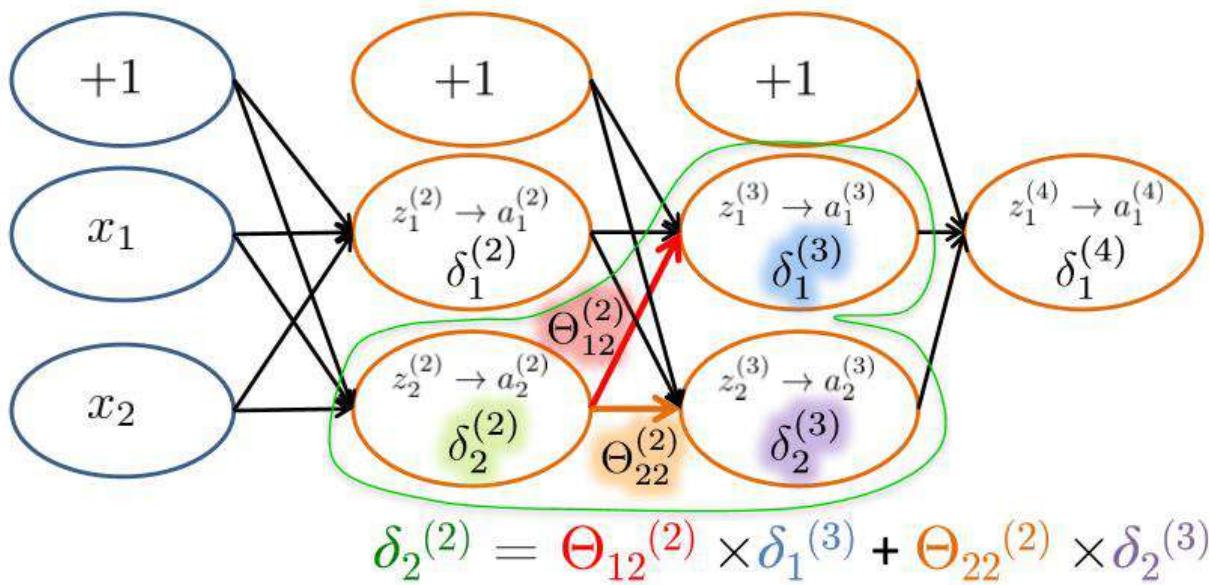
$$\delta_1^{(3)} = \Theta_{11}^{(3)} \times \delta_1^{(4)}$$

$\delta_j^{(l)}$ = “error” of node j in layer l

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

Backpropagation Summary



$\delta_j^{(l)}$ = “error” of node j in layer l

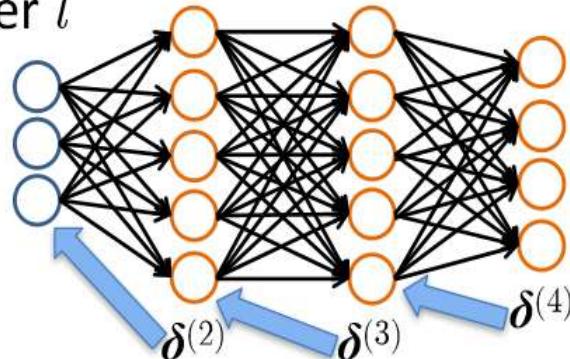
Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

Backpropagation: Gradient Computation

Let $\delta_j^{(l)}$ = “error” of node j in layer l

(#layers $L = 4$)

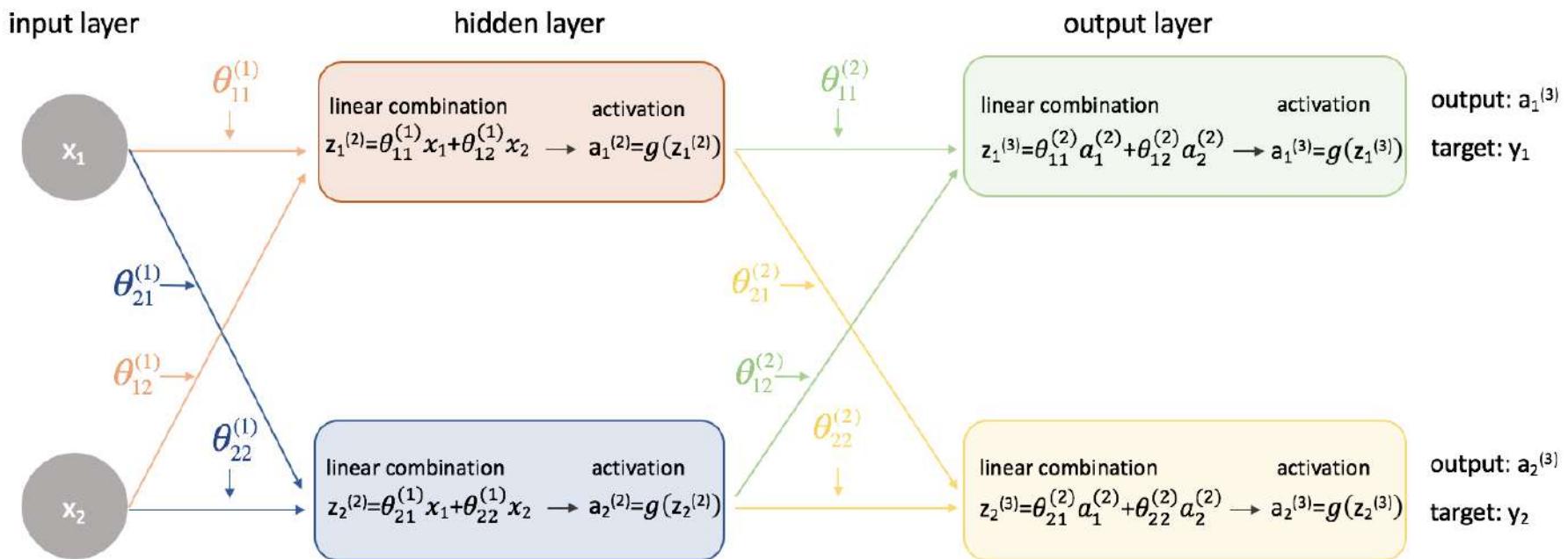


Backpropagation

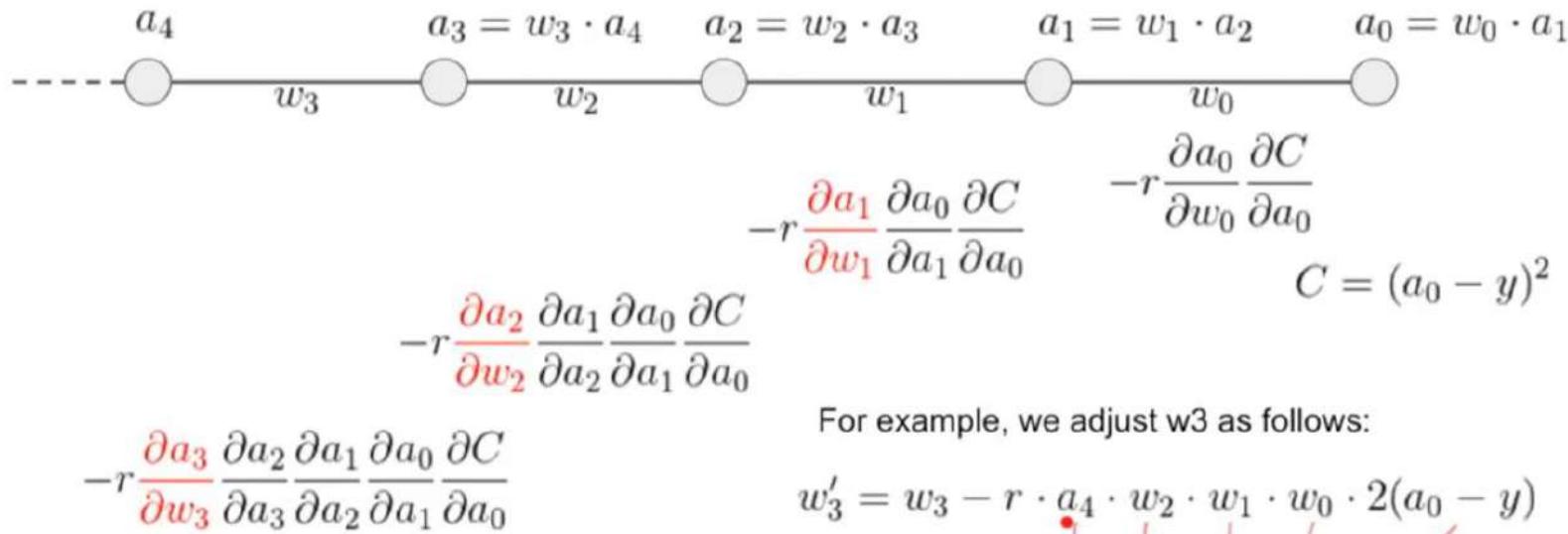
- $\delta^{(4)} = a^{(4)} - y$
- $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .*$
$$g'(\mathbf{z}^{(3)}) = a^{(3)} .* (1-a^{(3)})$$
- $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .*$
$$g'(\mathbf{z}^{(2)}) = a^{(2)} .* (1-a^{(2)})$$
- (No $\delta^{(1)}$)

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \quad (\text{ignoring } \lambda; \text{ if } \lambda = 0)$$

Example – back prop equations for 3 layer NN



Back propagation

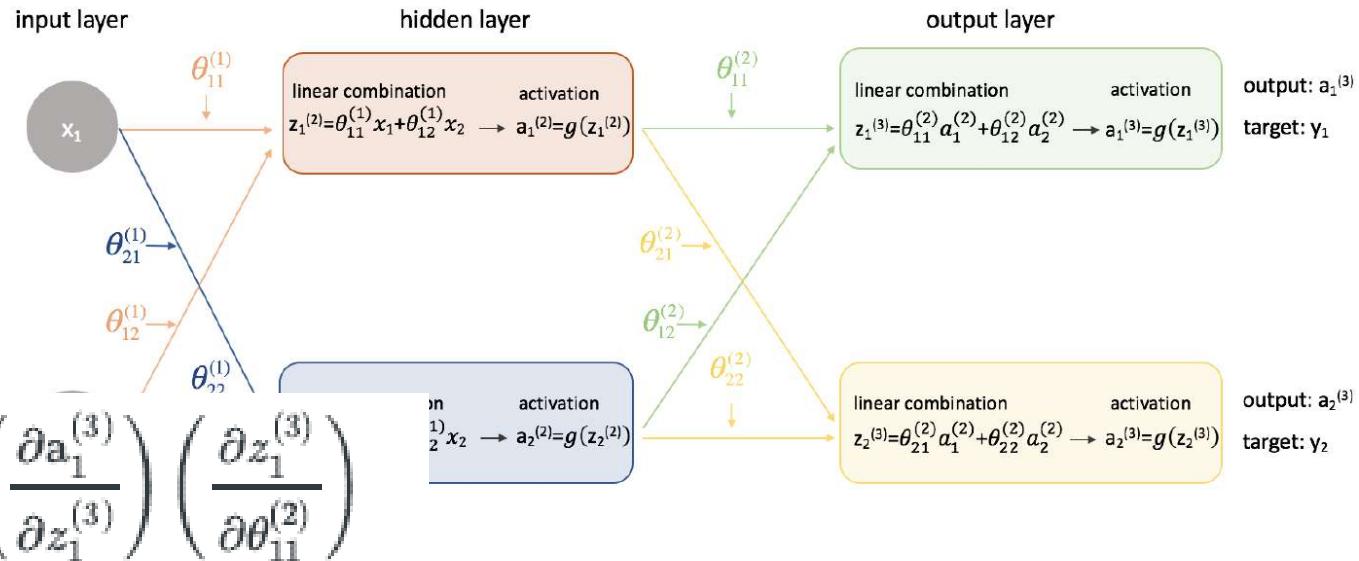


For example, we adjust w_3 as follows:

$$w'_3 = w_3 - r \cdot a_4 \cdot w_2 \cdot w_1 \cdot w_0 \cdot 2(a_0 - y)$$

$$-r \frac{\partial a_3}{\partial w_3} \frac{\partial a_2}{\partial a_3} \frac{\partial a_1}{\partial a_2} \frac{\partial a_0}{\partial a_1} \frac{\partial C}{\partial a_0}$$

Theta (2) parameters



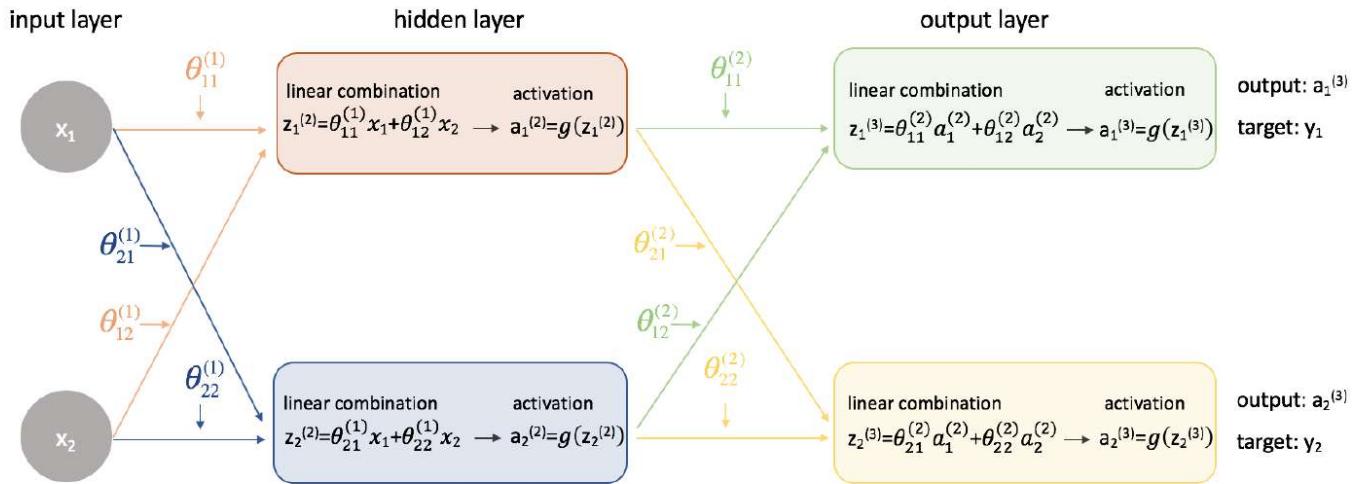
$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial \theta_{11}^{(2)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{12}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial \theta_{12}^{(2)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{21}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial \theta_{21}^{(2)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial \theta_{22}^{(2)}} \right)$$

Theta (1) parameters



$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \right) \left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}} \right) + \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} \right) \left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{12}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \right) \left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left(\frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}} \right) + \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} \right) \left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \right) \left(\frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{21}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial a_2^{(2)}} \right) \left(\frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left(\frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}} \right) + \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial a_2^{(2)}} \right) \left(\frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left(\frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}} \right) \left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \right) \left(\frac{\partial z_1^{(3)}}{\partial a_2^{(2)}} \right) \left(\frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left(\frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}} \right) + \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}} \right) \left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \right) \left(\frac{\partial z_2^{(3)}}{\partial a_2^{(2)}} \right) \left(\frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \right) \left(\frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}} \right)$$

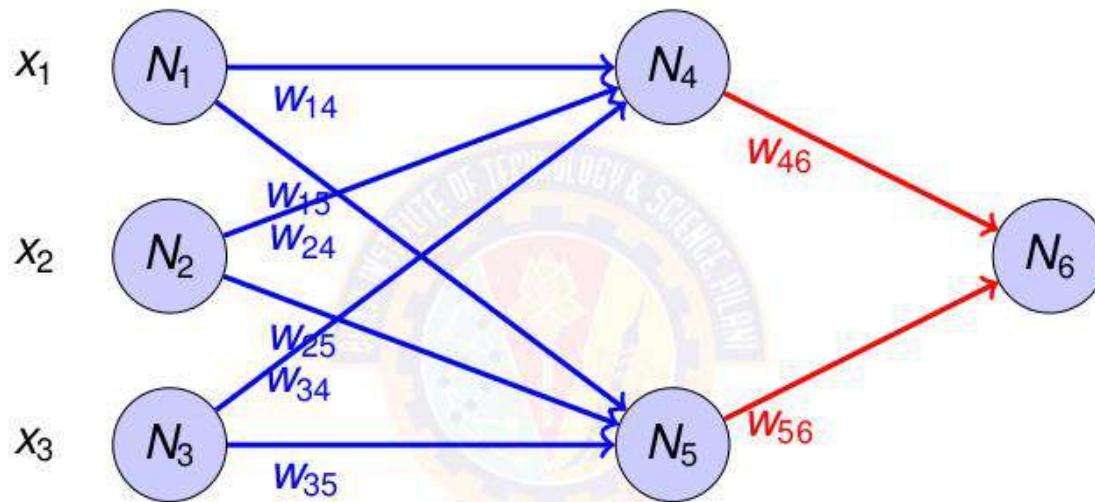
Neural Network - Practice Question 1

Consider the given feed forward neural network. Let learning rate be 0.9. The initial weights and the biases are given in the table. Consider the following sample as first training example $X = (1, 0, 1)$ with class label as 1. Show

1. computation of net output at each node.
2. calculation of error at each node.
3. calculation of updation of weights and biases after one iteration.

PS: Use Tom Mitchell book as reference to solve this problem.

Neural Network - Practice Question 1



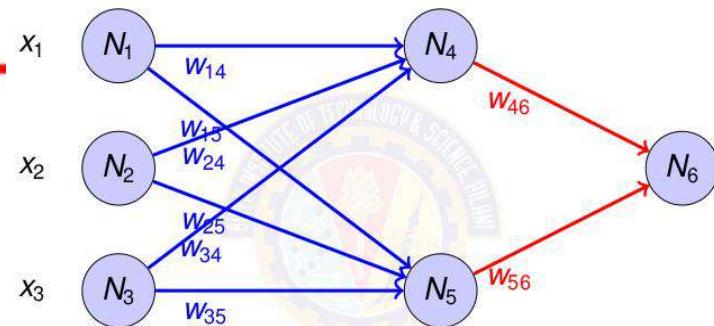
$$b_4 = (-0.4) \quad b_5 = 0.2 \quad b_6 = 0.1$$

$$w_{14} = 0.2 \quad w_{15} = (-0.3) \quad w_{45} = (-0.3)$$

$$w_{24} = 0.4 \quad w_{25} = 0.1 \quad w_{56} = (-0.2)$$

$$w_{34} = (-0.5) \quad w_{35} = 0.2$$

Neural Network - Practice Question 1



A] Computation of net output at each node.

$$\text{Equations } z_j = \sum_{ij} w_{ij}x_i + b_i$$

$$N_4 \quad z_4 = w_{14}x_1 + w_{24}x_2 + w_{34}x_3 + b_4$$

$$z_4 = 0.2 * 1 + 0 - 0.5 * 1 - 0.4 = (-0.7)$$

$$N_5 \quad z_5 = w_{15}x_1 + w_{25}x_2 + w_{35}x_3 + b_5$$

$$z_5 = (-0.3) * 1 + 0 + 0.2 * 1 + 0.2 = 0.1$$

$$N_6 \quad z_6 = w_{46}a_4 + w_{56}a_5 + b_6$$

$$z_6 = (-0.3) * 0.331 - 0.2 * 0.525 + 0.1$$

$$z_6 = (-0.104)$$

$$\begin{aligned} b_4 &= (-0.4) & b_5 &= 0.2 & b_6 &= 0.1 \\ w_{14} &= 0.2 & w_{15} &= (-0.3) & w_{45} &= (-0.3) \\ w_{24} &= 0.4 & w_{25} &= 0.1 & w_{56} &= (-0.2) \\ w_{34} &= (-0.5) & w_{35} &= 0.2 \end{aligned}$$

$$a_j = \frac{1}{1+e^{-z_j}}$$

$$a_4 = \frac{1}{1+e^{(-0.7)}} = 0.331$$

$$a_5 = \frac{1}{1+e^{-(0.1)}} = 0.525$$

$$a_6 = \frac{1}{1+e^{(-0.014)}} = 0.474$$

Neural Network - Practice Question 1

B] Calculation of error at each node

Equations :

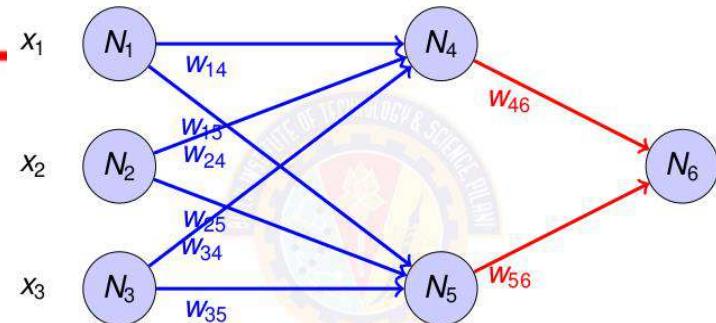
At the output layer $E_6 = a_6 - y$

$$\begin{aligned}\Delta W_{46} &= E_6 \cdot a_4 \\ &= (0.474 - 1) \cdot 0.331 = -0.1741\end{aligned}$$

At the hidden layer $E_j = a_j \cdot (1-a_j) \cdot \sum_k E_k W_{jk}$

$$\begin{aligned}E_4 &= a_4 \cdot (1-a_4) \cdot \sum_{k=6} E_6 W_{46} \\ &= g'(Z_{N4}) \cdot E_6 W_{46} \\ &= 0.331 (1-0.331) \cdot (-0.526) \cdot (-0.3) = \mathbf{0.0349}\end{aligned}$$

$$\begin{aligned}E_5 &= a_5 \cdot (1-a_5) \cdot \sum_{k=6} E_6 W_{56} \\ &= g'(Z_{N4}) \cdot E_6 W_{56} \\ &= 0.525 (1-0.525) \cdot (-0.526) \cdot (-0.2) = \mathbf{0.0262}\end{aligned}$$



$$\begin{array}{lll} b_4 = (-0.4) & b_5 = 0.2 & b_6 = 0.1 \\ w_{14} = 0.2 & w_{15} = (-0.3) & w_{46} = (-0.3) \\ w_{24} = 0.4 & w_{25} = 0.1 & w_{56} = (-0.2) \\ w_{34} = (-0.5) & w_{35} = 0.2 & \end{array}$$

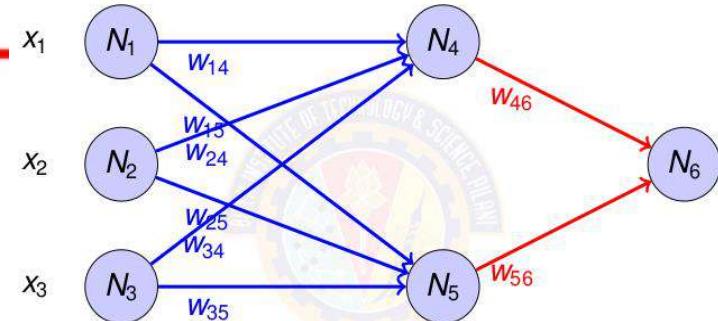
Neural Network - Practice Question 1

C] Update the gradient:

At the output layer $E_6 = a_6 - y$

$$\begin{aligned}\Delta W_{46} &= E_6 \cdot a_4 \\ &= (0.474 - 1) \cdot 0.331 = -0.1741\end{aligned}$$

$$\begin{aligned}W_{46} &= W_{46} - LR * \Delta W_{46} \\ &= -0.3 - (0.9) * (-0.1741)\end{aligned}$$



$$\begin{array}{lll} b_4 = (-0.4) & b_5 = 0.2 & b_6 = 0.1 \\ w_{14} = 0.2 & w_{15} = (-0.3) & w_{45} = (-0.3) \\ w_{24} = 0.4 & w_{25} = 0.1 & w_{56} = (-0.2) \\ w_{34} = (-0.5) & w_{35} = 0.2 & \end{array}$$

Given: training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$

Initialize all $\Theta^{(l)}$ randomly (NOT to 0!)

Loop // each iteration is called an epoch

Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$ (Used to accumulate gradient)

For each training instance (x_i, y_i) :

Set $a^{(1)} = x_i$

Compute $\{a^{(2)}, \dots, a^{(L)}\}$ via forward propagation

Compute $\delta^{(L)} = a^{(L)} - y_i$

Compute errors $\{\delta^{(L-1)}, \dots, \delta^{(2)}\}$

Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Compute avg regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{n} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n} \Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$

Update weights via gradient step $\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$

Until weights converge or max #epochs is reached

Backpropagation

DNN – Learning of Weights & Bias:

Sample Problem – Forward Backward Propagation

x1	x2	y	η
7	6	0	5

Forward Pass:

$$H_1 = w_1x_1 + w_2x_2 + b_1$$

$$H_{\text{out-1}} = \frac{1}{1+e^{-H_1}}$$

$$H_2 = w_3x_1 + w_4x_2 + b_2$$

$$H_{\text{out-2}} = \frac{1}{1+e^{-H_2}}$$

$$H_3 = w_5H_{\text{out-1}} + w_6H_{\text{out-2}} + b_3$$

$$H_{\text{out-3}} = \frac{1}{1+e^{-H_3}}$$

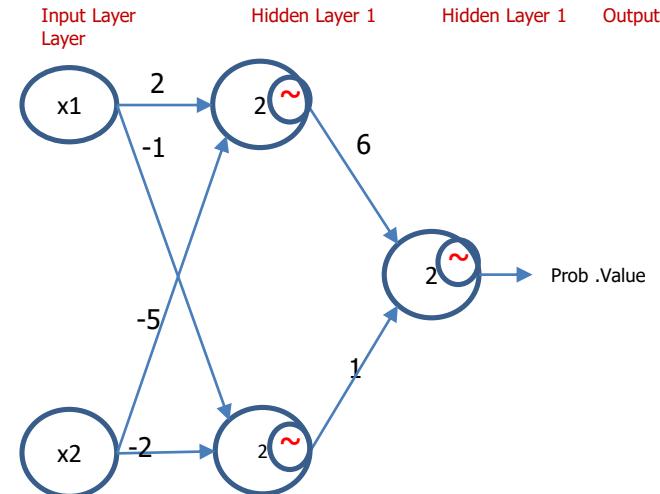
$$\text{Error} = \frac{1}{2} * (H_{\text{out-3}} - y)^2$$

Backward Pass

$$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial H_{\text{out-3}}} * \frac{\partial H_{\text{out-3}}}{\partial H_3} * \frac{\partial H_3}{\partial w_5}$$

$$\frac{\partial E}{\partial w_6} = \frac{\partial E}{\partial H_{\text{out-3}}} * \frac{\partial H_{\text{out-3}}}{\partial H_3} * \frac{\partial H_3}{\partial w_6}$$

$$\frac{\partial E}{\partial b_3} = \frac{\partial E}{\partial H_{\text{out-3}}} * \frac{\partial H_{\text{out-3}}}{\partial H_3} * \frac{\partial H_3}{\partial b_3}$$



DNN – Learning of Weights & Bias:

Sample Problem – Forward Backward Propagation

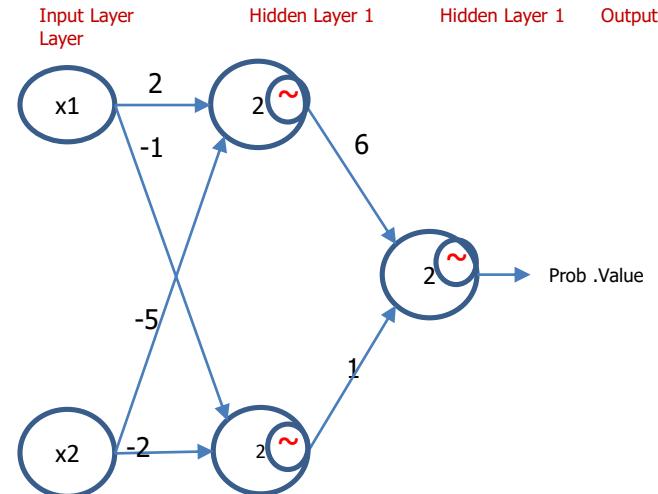
Backward Pass

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-1}} * \frac{\partial H_{out-1}}{\partial H_1} * \frac{\partial H_1}{\partial w_1}$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-1}} * \frac{\partial H_{out-1}}{\partial H_1} * \frac{\partial H_1}{\partial w_2}$$

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-1}} * \frac{\partial H_{out-1}}{\partial H_1} * \frac{\partial H_1}{\partial b_1}$$

x1	x2	y	η
7	6	0	5



$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-2}} * \frac{\partial H_{out-2}}{\partial H_2} * \frac{\partial H_2}{\partial w_3}$$

$$\frac{\partial E}{\partial w_4} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-2}} * \frac{\partial H_{out-2}}{\partial H_2} * \frac{\partial H_2}{\partial w_4}$$

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial H_{out-3}} * \frac{\partial H_{out-3}}{\partial H_3} * \frac{\partial H_3}{\partial H_{out-2}} * \frac{\partial H_{out-2}}{\partial H_2} * \frac{\partial H_2}{\partial b_2}$$



DNN – Learning of Weights & Bias:

Sample Problem – Forward Backward Propagation

x1	x2	y	η
7	6	0	5

Weight & bias Update:

$$w_5 - \eta * \frac{\partial E}{\partial w_5}$$

$$w_6 - \eta * \frac{\partial E}{\partial w_6}$$

$$b_3 - \eta * \frac{\partial E}{\partial b_3}$$

$$w_1 - \eta * \frac{\partial E}{\partial w_1}$$

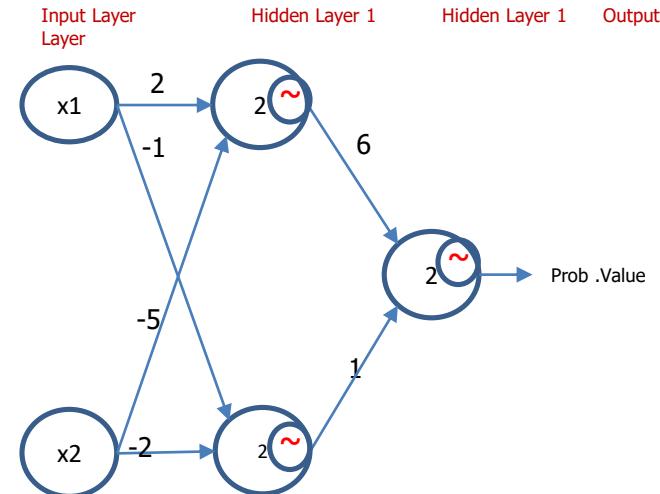
$$w_2 - \eta * \frac{\partial E}{\partial w_2}$$

$$b_1 - \eta * \frac{\partial E}{\partial b_1}$$

$$w_3 - \eta * \frac{\partial E}{\partial w_3}$$

$$w_4 - \eta * \frac{\partial E}{\partial w_4}$$

$$b_2 - \eta * \frac{\partial E}{\partial b_2}$$



Configure Training Parameters

I Design the architecture of the network

II Choose the activation function to compute the hidden layer values

III Choose the cost function

IV Choose the optimizer algorithm

V. Train the feedforward network

VI Evaluate the performance of the network

```
# Configure the model for training, by using appropriate optimizers and regularizations
# Available optimizer: adam, rmsprop, adagrad, sgd
# loss: objective that the model will try to minimize.
# Available loss: categorical_crossentropy, binary_crossentropy, mean_squared_error
# metrics: List of metrics to be evaluated by the model during training and testing.

dnnModel.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics=['accuracy'])
```

Train the Model

```
# train the model

h = dnnModel.fit(Xtrain, Ytrain, epochs=25, batch_size=64)
```

Early Stopping

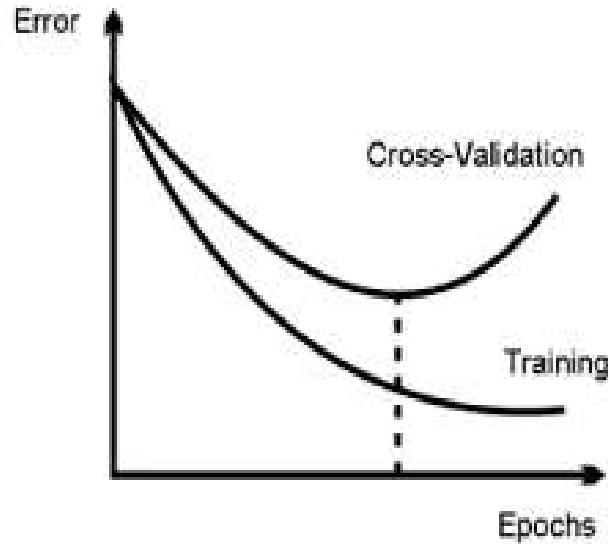
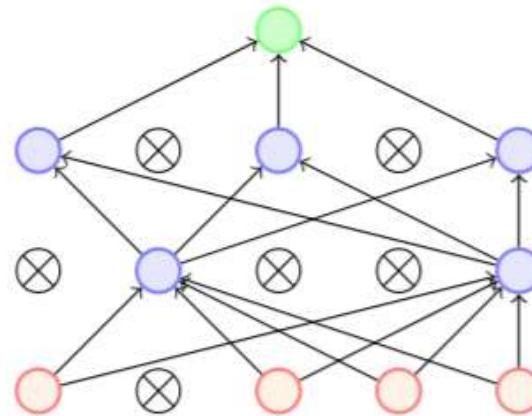
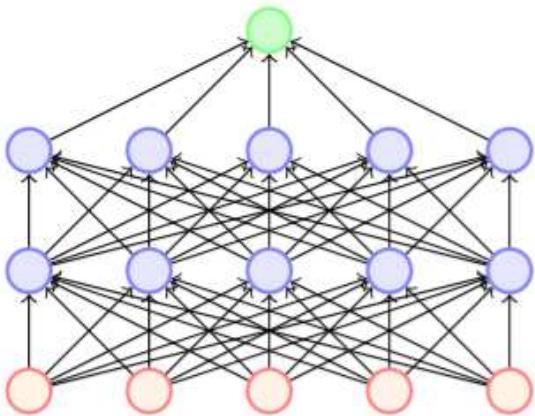


Figure 2: Profiles for training and cross-validation errors.

- If we let a complex model train long enough on a given data set it can eventually learn the data exactly.
- Given data that isn't represented in the training set, the model will perform poorly (overfitting).
- **How is the sweet spot for training located?**
- When the error on the training set begins to deviate from the error on the validation set, a threshold can be set to determine the early stopping condition and the ideal number of epochs to train.

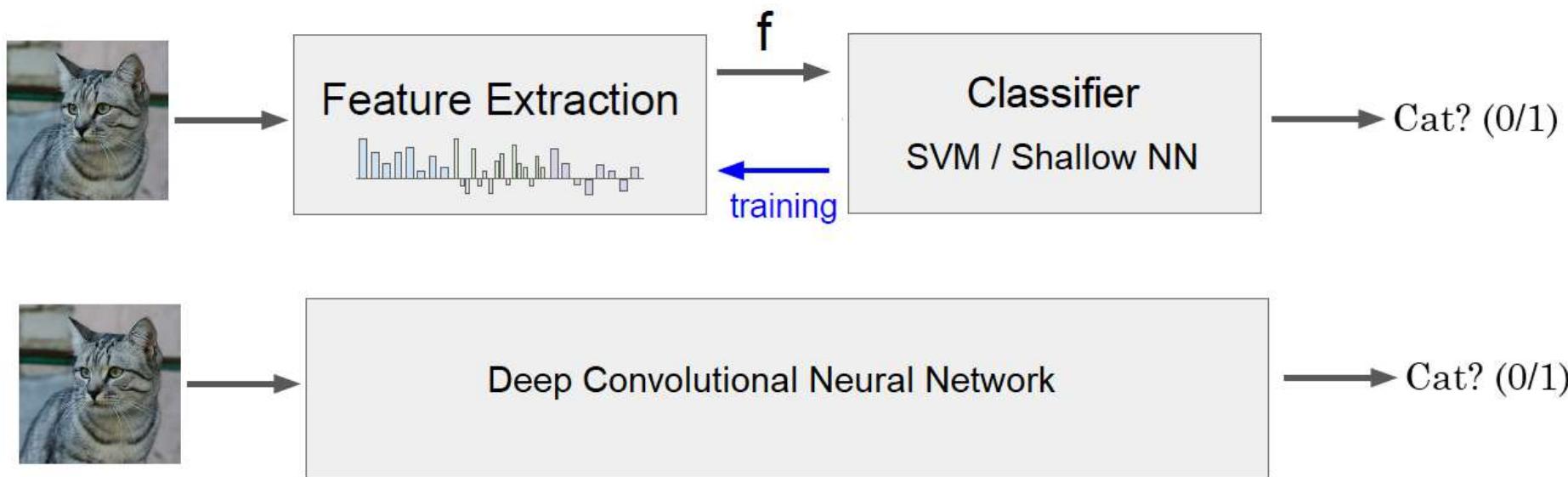
Dropout



- Dropout refers to dropping out units
- Temporarily remove a node and all its incoming/outgoing connections resulting in a thinned network
- Each node is retained with a fixed probability (typically $p = 0.5$) for hidden nodes and $p = 0.8$ for visible nodes

Convolutional Neural Network

Image Classification



Convolutional Neural Networks (CNN)



- Special kind of neural network for processing data that has a known, grid-like topology.
 - e.g., Time-series data – 1D grid taking samples at regular time intervals.
 - e.g., Image data – 2D grid of pixels.
 - Network employs a mathematical operation called **convolution**.
 - Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in some of their layers.
-

CNN – Feature Map Creation Sample Process

-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1
-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1
-1	-1	1	1	1	-1
-1	1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

CNN – Filters

1	1	1
1	-1	1
1	1	1

-1	-1	-1
1	1	1
-1	-1	-1

-1	1	-1
-1	1	-1
-1	1	-1

-1	-1	1
-1	1	-1
1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

Loops

Horizontals

Verticals

Diagonal1

Diagonal2

CNN – Convolution Process

-1	-1	-1	-1	-1	-1
-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1
1	1	1
-1	-1	-1

7					

CNN – Convolution Process

-1	-1	-1	-1	-1	-1
-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1
1	1	1
-1	-1	-1

7	9				

CNN – Convolution Process

-1	-1	-1	-1	-1	-1
-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1
1	1	1
-1	-1	-1

0.78	1	0.78	0.56		
-0.11	-0.56	-0.33	-0.11		
0.11	0.33	0.11	0.33		
0.56	0.33	0.33	0.11		

7	9	7	5		
-1	-5	-3	-1		
1	3	1	3		
5	3	3	1		

CNN – Transformation Process - ReLU

-1	-1	-1	-1	-1	-1
-1	1	1	1	1	-1
-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1

-1	-1	-1
1	1	1
-1	-1	-1

0.78	1	0.78	0.56		
0	0	0	0		
0.11	0.33	0.11	0.33		
0.56	0.33	0.33	0.11		

7	9	7	5		
0	0	0	0		
1	3	1	3		
5	3	3	1		

CNN – Padding

0	0	0	0	0	0	0	0	0
0	-1	1	1	1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	-1	-1	0	
0	-1	-1	-1	-1	-1	-1	0	
0	0	0	0	0	0	0	0	0

-1	-1	-1
1	1	1
-1	-1	-1

0.11	0.11	0.11	0.11	0.11	0.11	0.11
0.11	0.78	1	0.78	0.56	0.11	
0.11	0	0	0	0	0	0.11
0.56	0.11	0.33	0.11	0.33	0.11	
0.11	0.56	0.33	0.33	0.11	0.11	
0.11	0.11	0.56	0.33	0.33	0.11	
0.11	0.11	0.11	0.11	0.11	0.11	

Sample Values-approximated

CNN – Padding

0	0	0	0	0	0	0	0	0
0	-1	1	1	1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	1	-1	0	
0	-1	-1	-1	-1	-1	-1	0	
0	0	0	0	0	0	0	0	0

-1	1	-1
-1	1	-1
-1	1	-1

0.11	0.11	0.11	0.11	0.11	0.11	0.11
0.11	0.78	1	0.78	0.56	0.11	
0.11	0	0	0	1	0.11	
0.56	0.11	0.33	0.11	1	0.11	
0.11	0.56	0.33	0.33	0.11	0.11	
0.11	0.11	0.56	0.33	0.33	0.33	0.11

Sample Values-approximated

CNN – Pooling

0.11	0.11	0.11	0.11	0.11	0.11
0.11	0.78	1	0.78	0.56	0.11
0.11	0	0	0	1	0.11
0.56	0.11	0.33	0.11	1	0.11
0.11	0.56	0.33	0.33	0.11	0.11
0.11	0.11	0.56	0.33	0.33	0.11

0.78	1	0.56
0.56	0.33	1
0.56	0.56	0.33

Sample Values-approximated

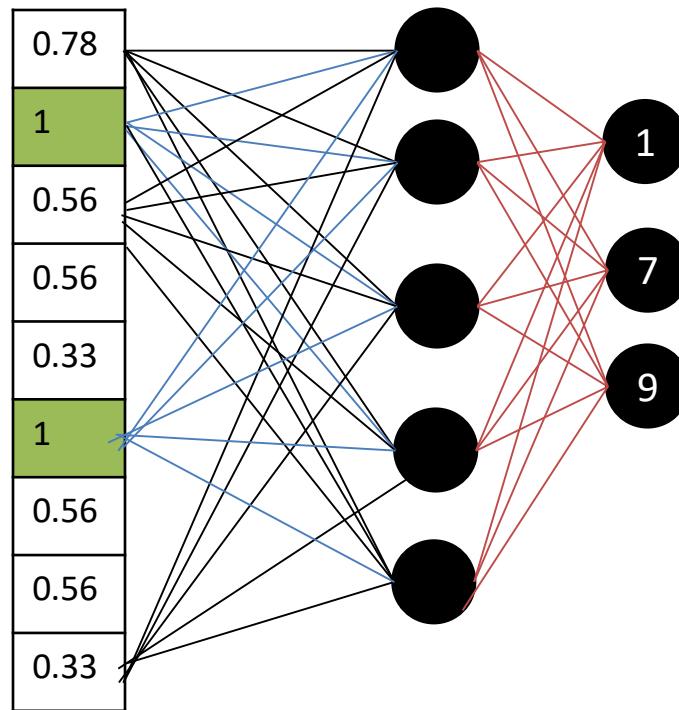
MAX Pooling Vs AVG Pooling

Eg. Stride =2

0.28	0.5	0.22
0.19	0.11	0.55
0.22	0.39	0.17

CNN – Classification – Fully Connected Network

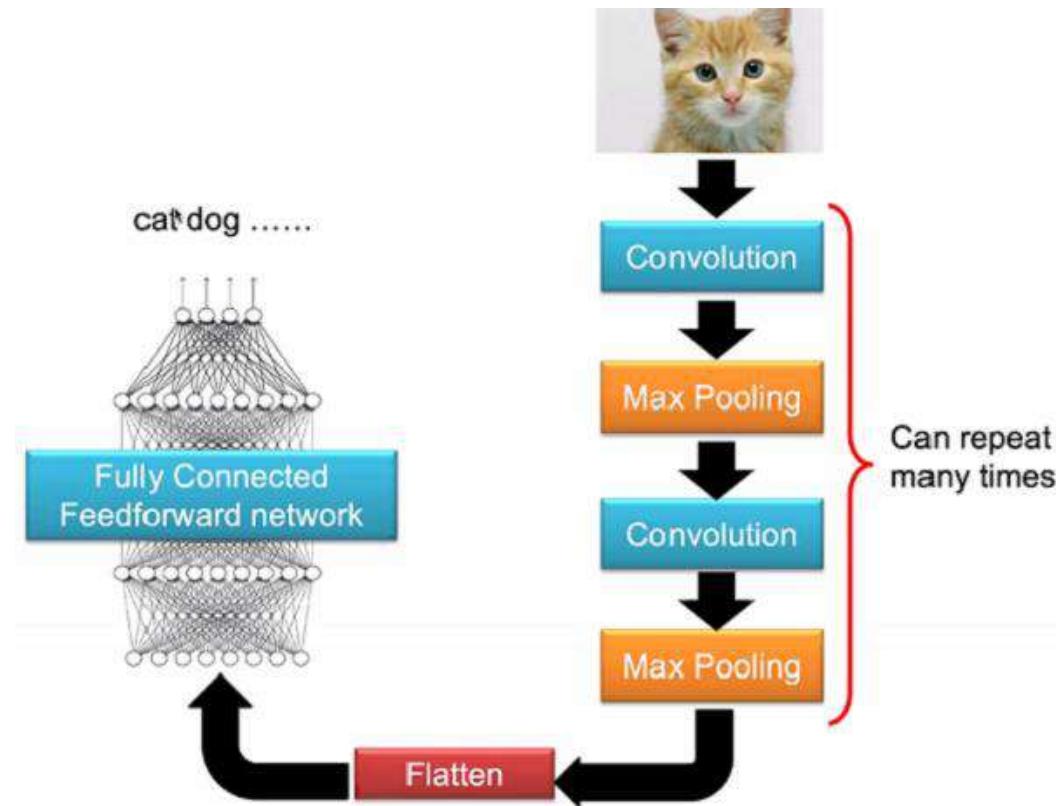
0.78	1	0.56
0.56	0.33	1
0.56	0.56	0.33



Types of Layers

- Convolution (Conv) Layer
- Pooling (Pool) Layer
- Fully Connected (FC) Layer

Convolutional Neural Networks



Handwritten Character recognition 10-categories classification problem

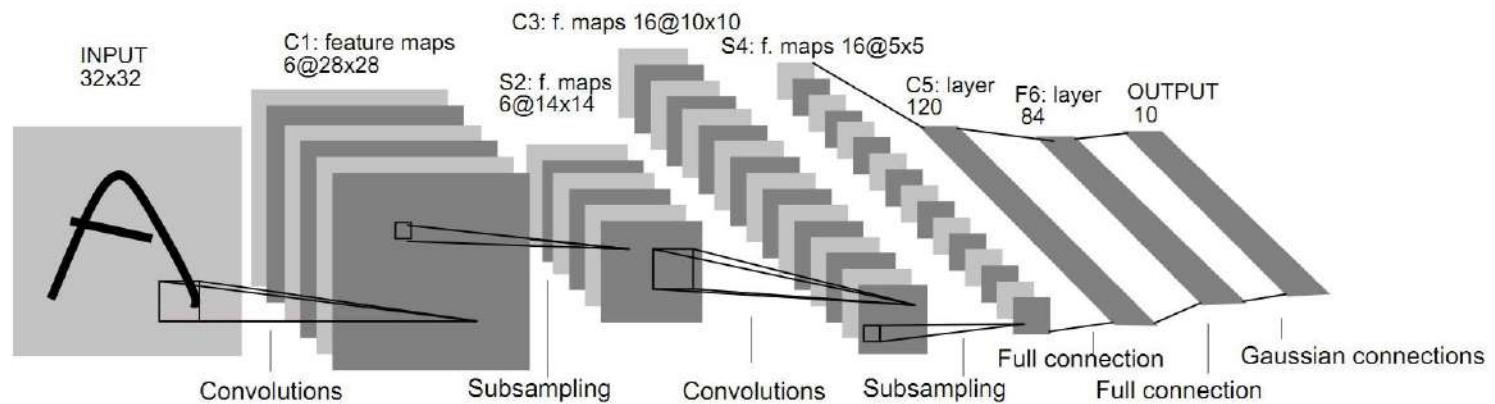


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Thank you !

Required Reading for completed session :

T1 - Chapter # 4 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 5 (Christopher M. Bishop, Pattern Recognition & Machine Learning)

Next Session Plan :

Instance-based Learning



BITS Pilani
Pilani Campus

Machine Learning

DSE CLZG565

Neural Network & Instance-based Learning

Raja vadhana P
Assistant Professor,
BITS - CSIS

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: Slides of Prof. Chetana, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 8 : Neural Network

- Perceptron
- Backpropagation Network
- Convolution Network
- Recurrent Network

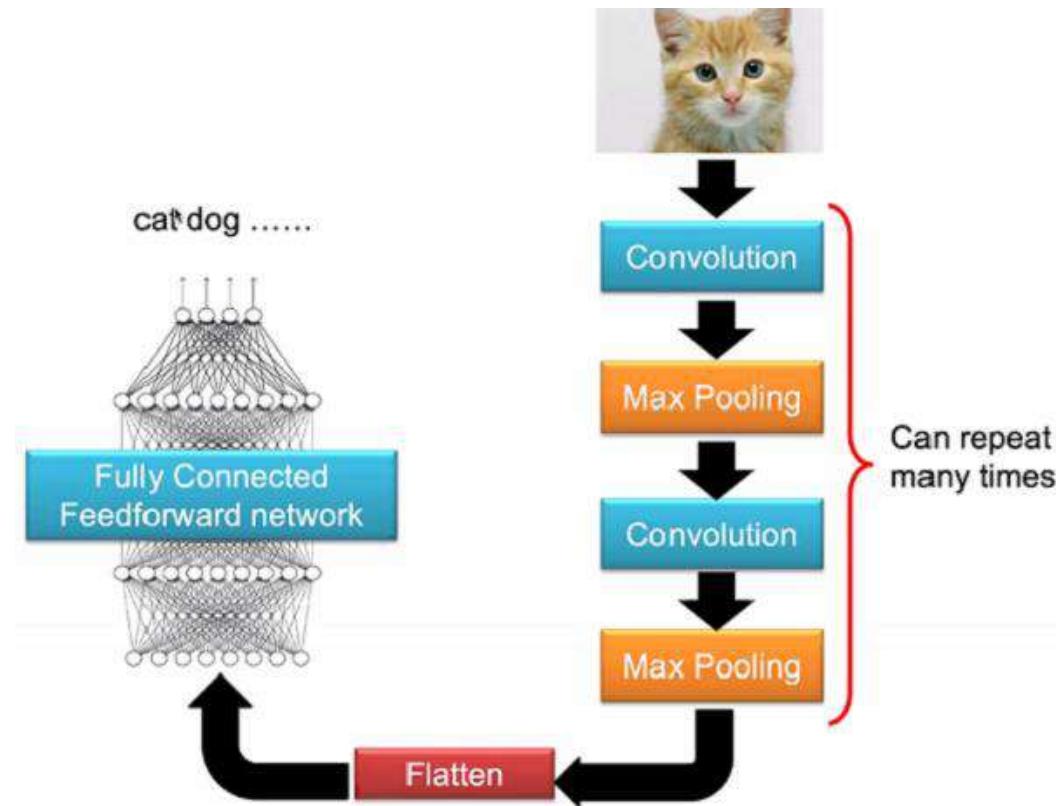
Module – 9 : Instance Based Learning

- K – Nearest Neighbor Learning
- Locally Weighted Regression Learning

Refer Tom Mitchell – Chapter 8

Convolutional Neural Network

Convolutional Neural Networks



Convolutional Neural Networks

The whole CNN

Property 1

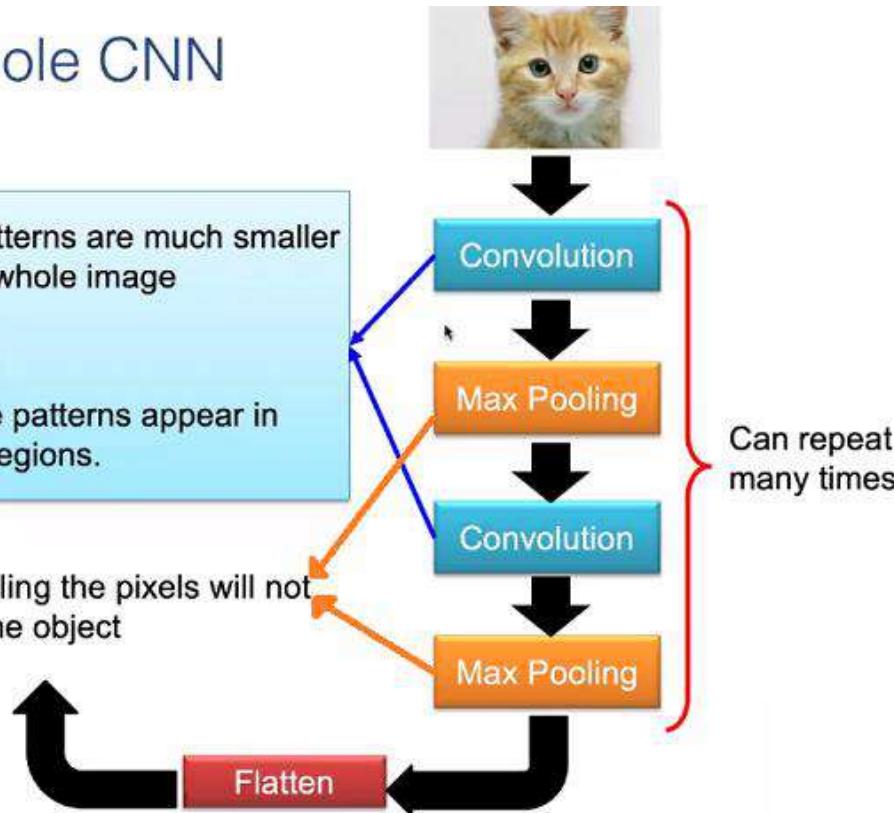
- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

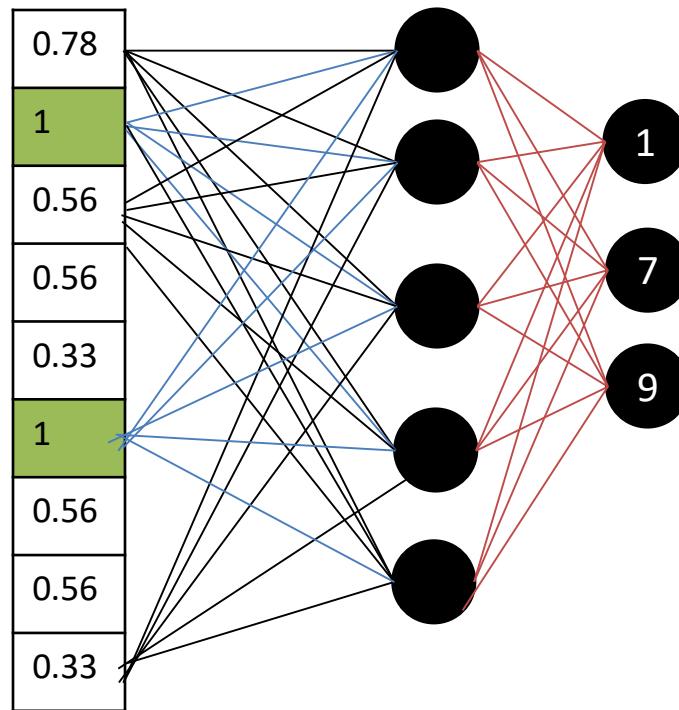
Property 3

- Subsampling the pixels will not change the object



CNN – Classification – Fully Connected Network

0.78	1	0.56
0.56	0.33	1
0.56	0.56	0.33



Handwritten Character recognition 10-categories classification problem

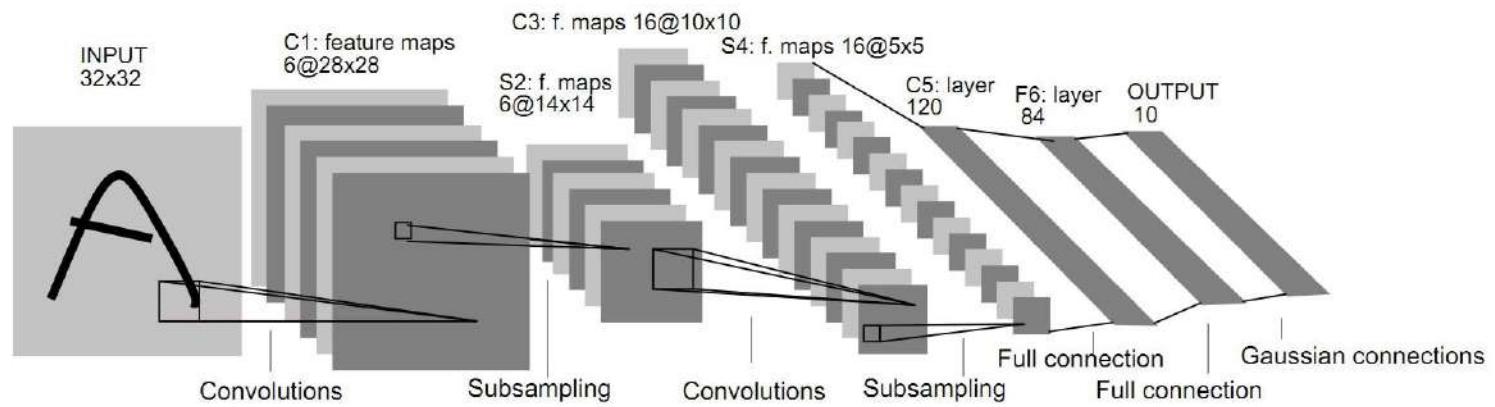


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Recurrent Neural Network Sequence Learning

Sequence Learning

Speech recognition



"The quick brown fox jumped over the lazy dog."

Music generation



Sentiment classification

"There is nothing to like
in this movie."



DNA sequence analysis

AGCCCCTGTGAGGAAC TAG



AGCCCCTGTGAGGAAC TAG

Machine translation

Voulez-vous chanter avec
moi?



Do you want to sing with
me?

Video activity recognition



Running

Name entity recognition

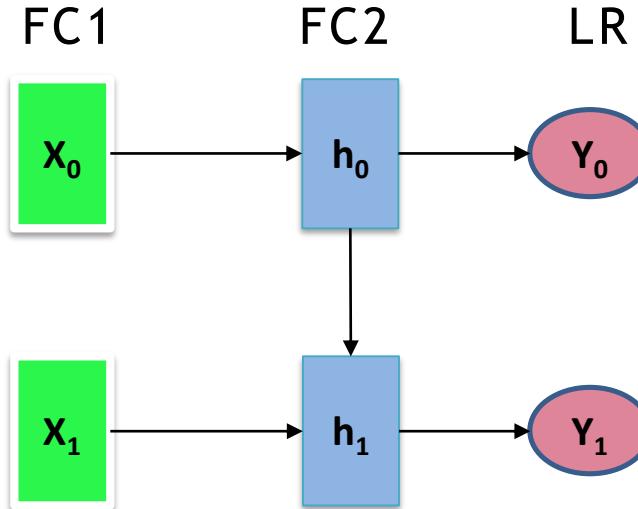
Yesterday, Harry Potter
met Hermione Granger.



Yesterday, Harry Potter
met Hermione Granger.

Recurrent Neural Network

- Stack of neurons in one layer is connected to every other neuron in another layer
- Stack of neurons in previous iteration influences the next iteration



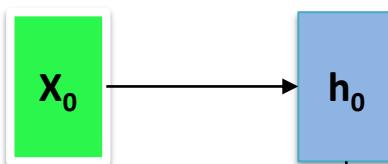
Earnings Per share	Cash Flow per share	Dividends per share	Risk	Discount rate	PRICE
0.3	0.7	0.1	0.3	0.5	100
0.3	0.5	0.5	0.4	0.5	150

Recurrent Neural Network - Models

Type 1

FC1 FC2 LR

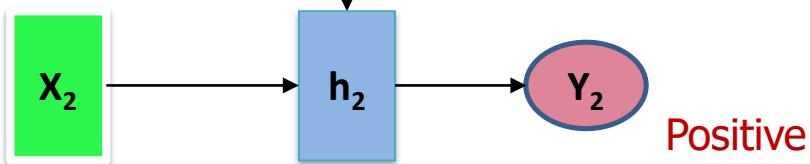
Good



Informative

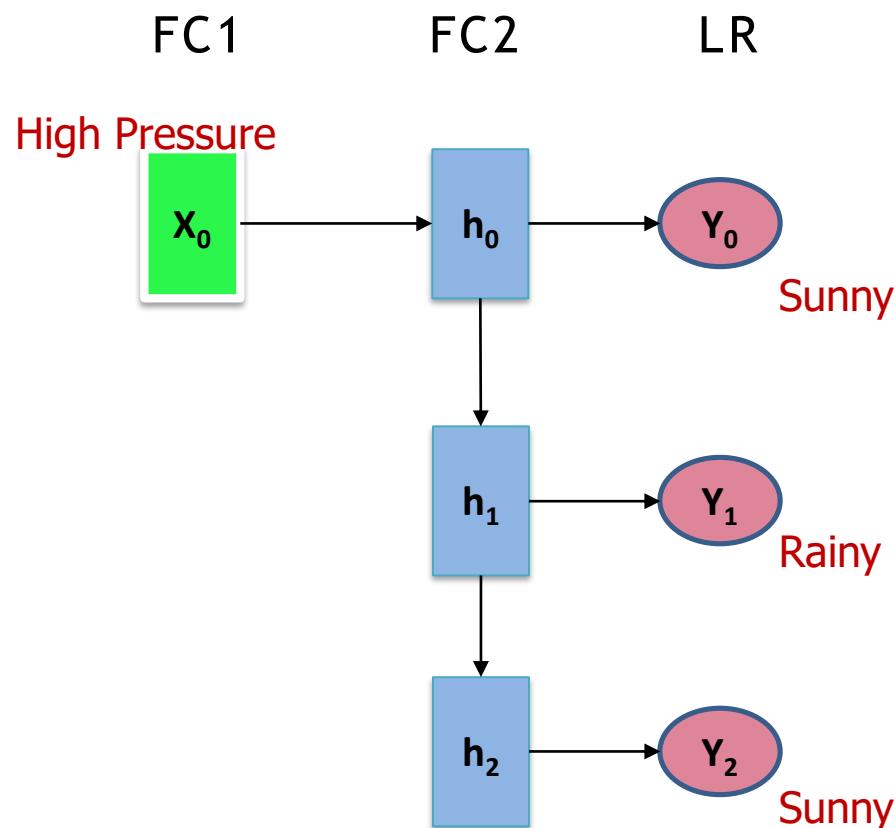


Session



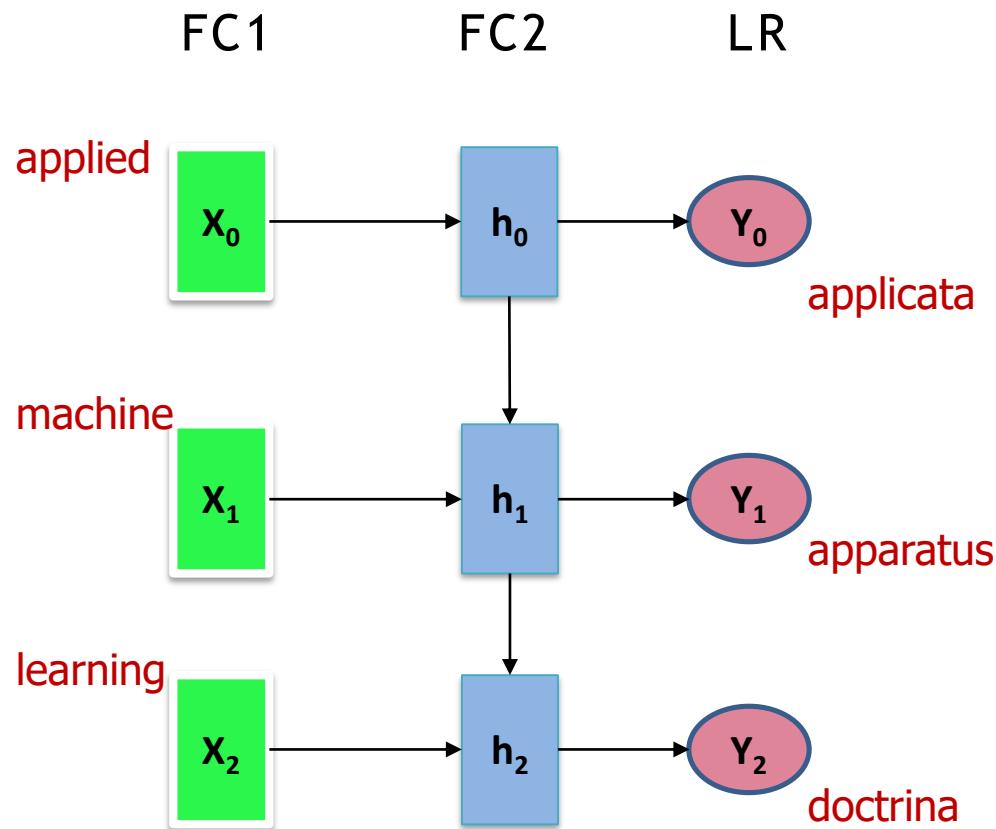
Recurrent Neural Network - Models

Type 2



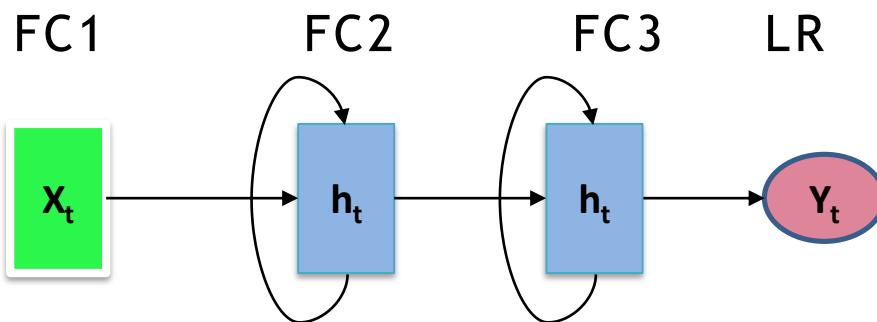
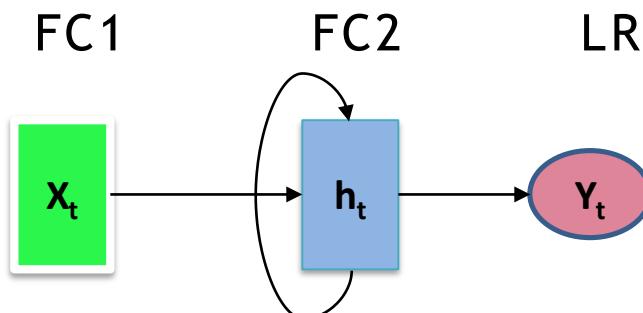
Recurrent Neural Network - Models

Type 3



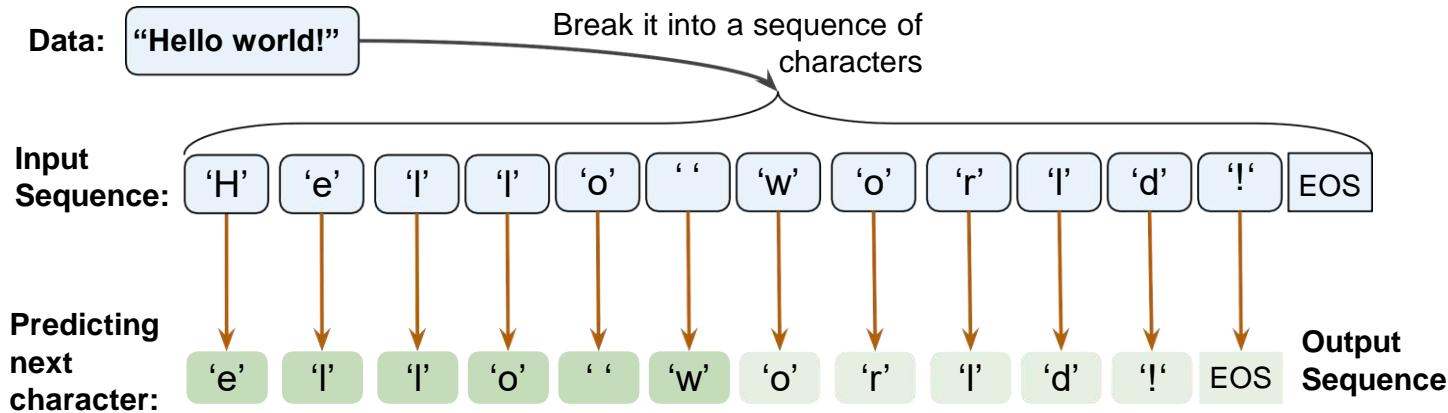
Recurrent Neural Network - Architecture

Single Vs Multilayer RNN



Data Processing for a Character RNN

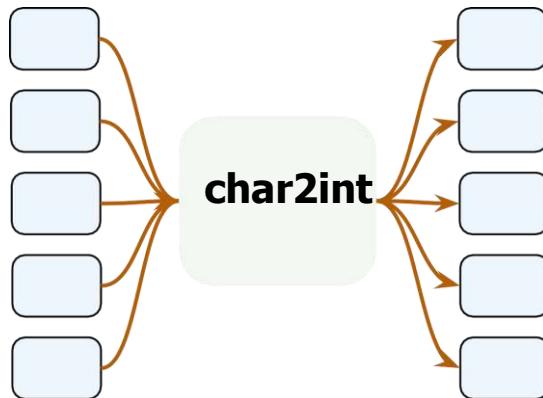
Step 1: Break up text into characters



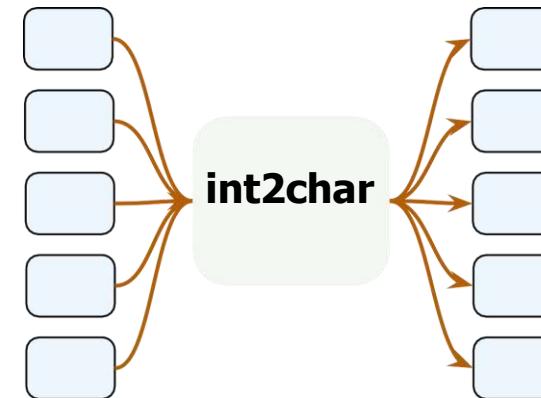
Data Processing for a Character RNN

Step 2: Define Mapping Dictionaries

Mapping characters to integers



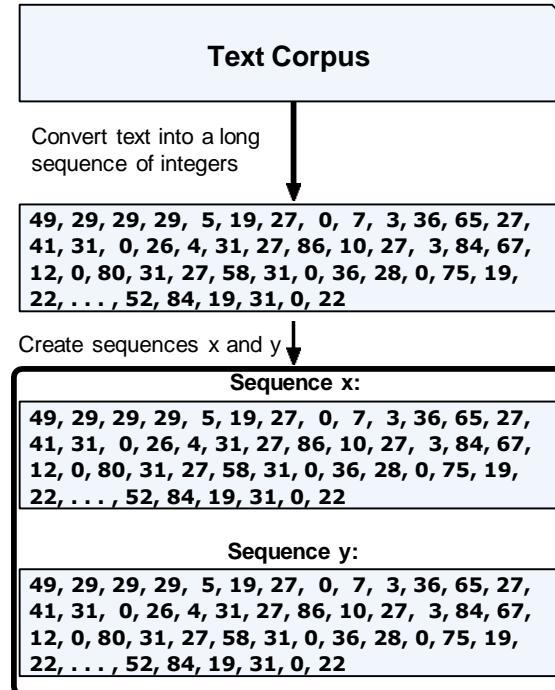
Mapping integers to characters



Data Processing for a Character RNN

Step 3: Define Inputs and Outputs

Outputs are the characters shifted by 1 position as we want to predict the next character



- Letters/words can be represented by one-hot encoding
- For words, semantic encodings available, e.g., word2vec, GLOVE etc.



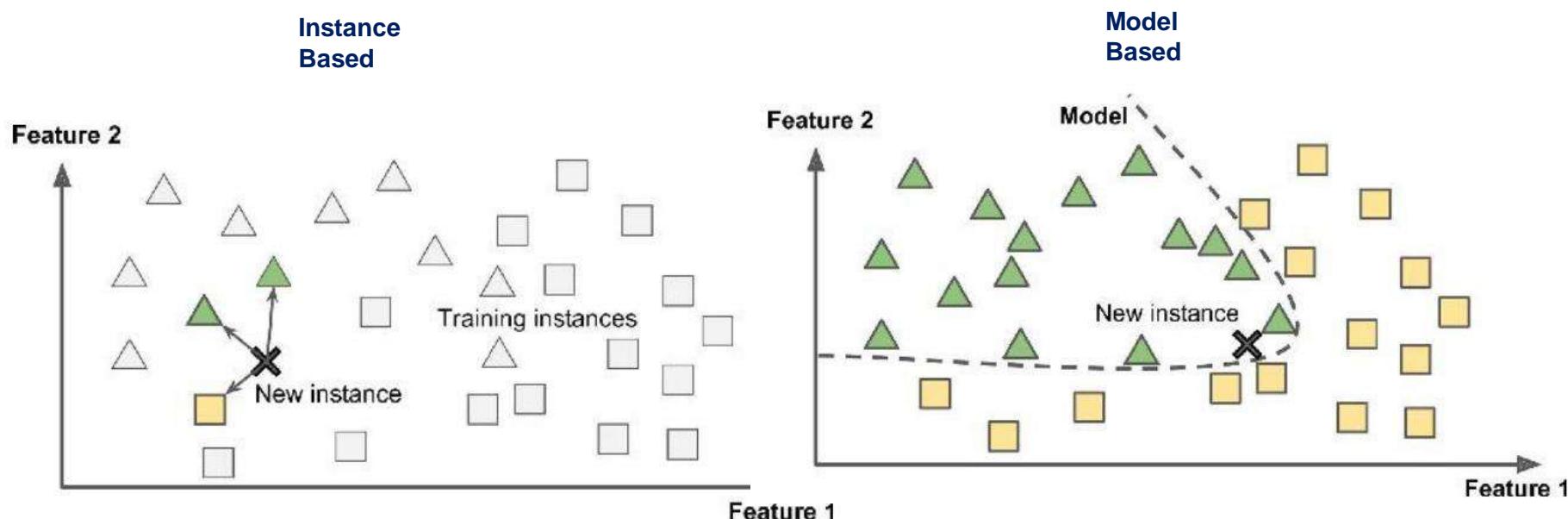
Instance Based Learning

Introduction to Machine Learning

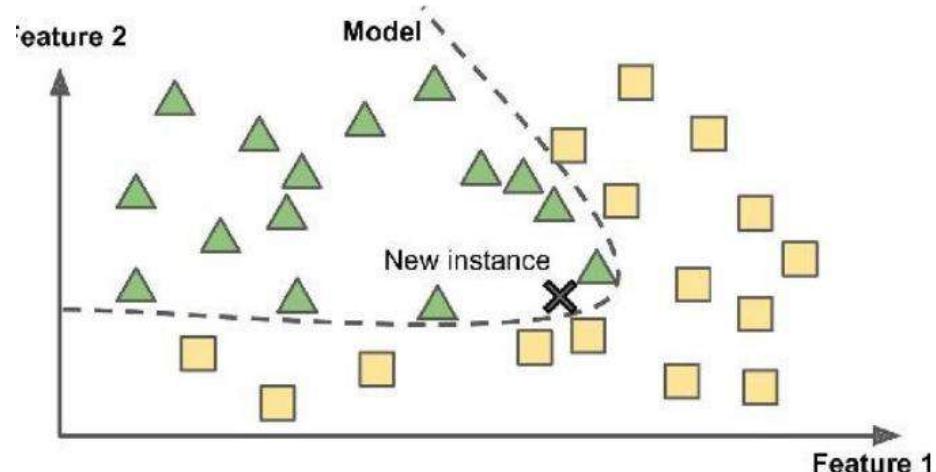
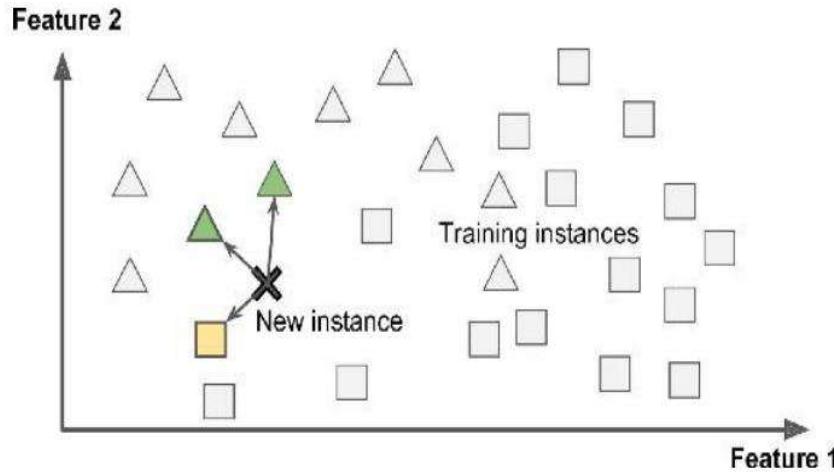


Types: Based on how & when the training data is used

- Instance Based Learning: Compare new data points to known data points
- Model Based learning : Detect patterns in the training data and build a predictive model



Instance Based Learning



Lazy Learners : Less time in training but more time in predicting

Eager Learners : More time in training but Less time in predicting

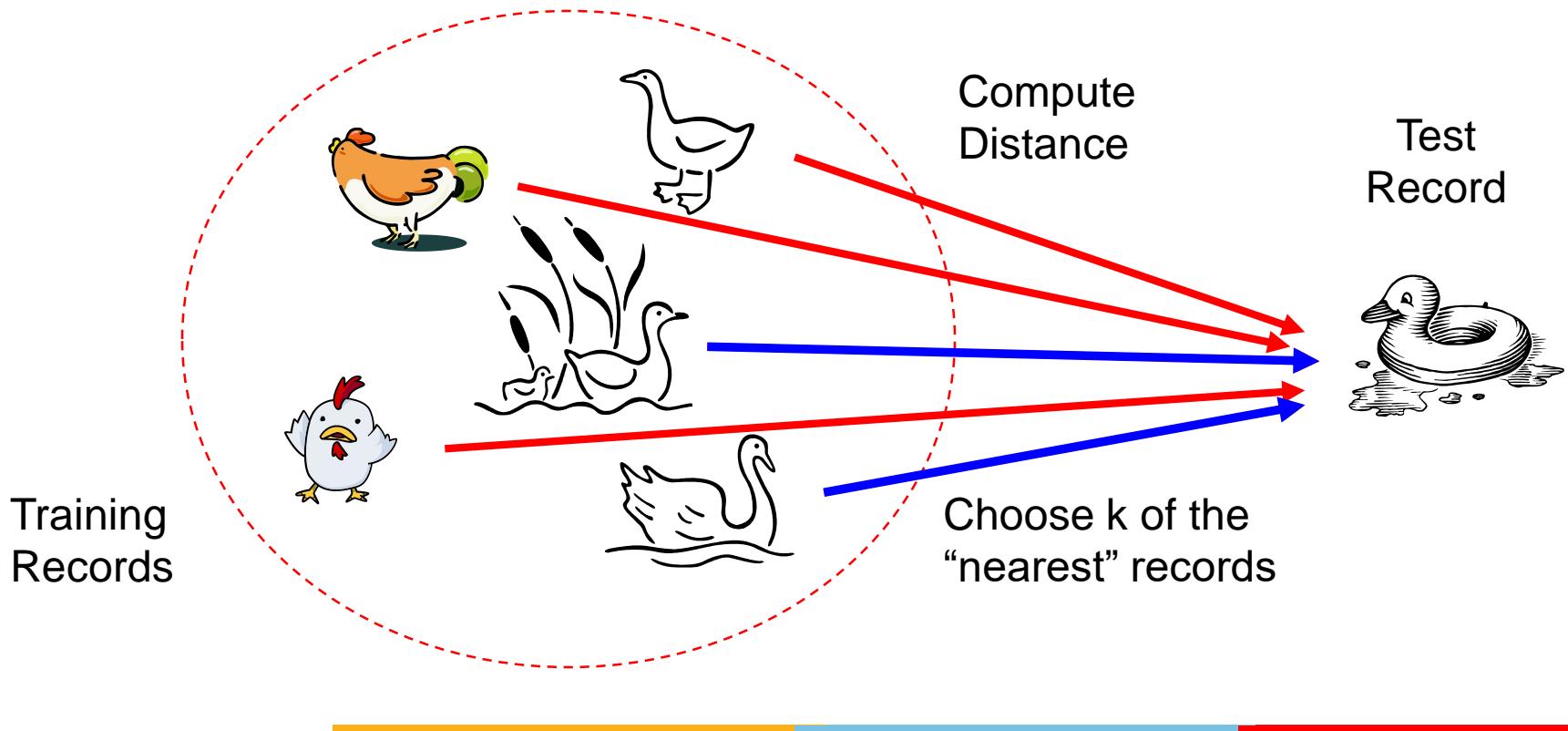
Instance Based Learning



Nearest Neighbor Approach

Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck

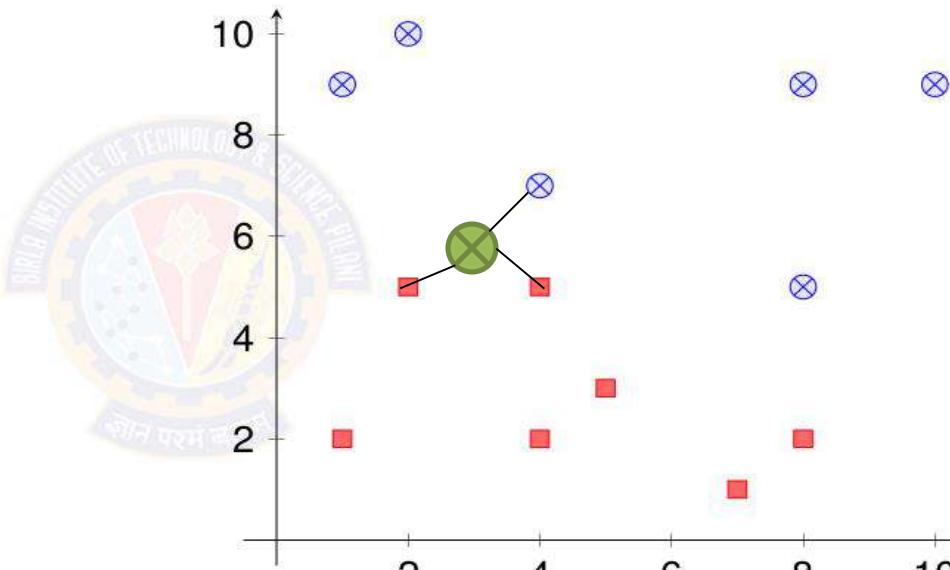


Nearest Neighbor Approach : Notations

K-nearest neighbor:

- Given x_q , take **vote** among its k nearest neighbors (if **discrete-valued target function**)
- Take **mean(or median)** of f values of k nearest neighbors (if **real-valued**) $f^*(x_q) = \sum_{i=1}^k f(x_i)/k$

x_1	x_2	y
1	9	white
10	9	white
4	7	white
4	5	pink
5	3	pink
8	9	white
4	2	pink
2	5	pink
7	1	pink
2	10	white
8	5	white
1	2	pink
8	2	pink



Interpretation : Assume that the Instances represented as points in a Euclidean space

Instance Based Learning



K-Nearest Neighbor Classifier: Another Example

Given a query item:



$k = 3$ votes for "cat"

Find k closest matches
in a labeled dataset ↓

Return the most
Frequent Label



Instance Based Learning



K-Nearest Neighbor Classifier: Another Example

Given a query item:



Find k closest matches
in a labeled dataset ↓



2 votes for cat,
1 each for Buffalo, Deer, Lion

Cat wins...

k = 5 Neighborhood Region

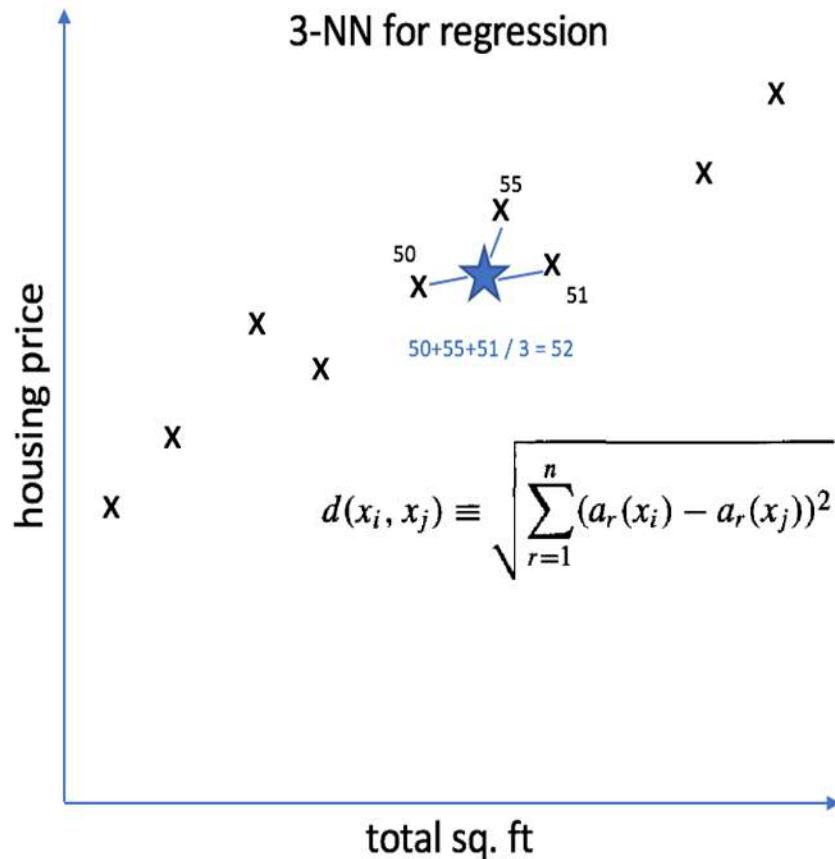
Return the most
Frequent Label

Instance Based Learning



K-Nearest Neighbor Regression: Example

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
1852	178
...	...
...	...



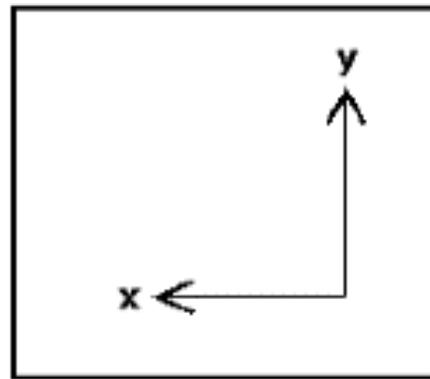
Source Credit : <https://www.jeremyjordan.me/k-nearest-neighbors/>

Measures of Dis/similarity

Distances

- Manhattan Distance

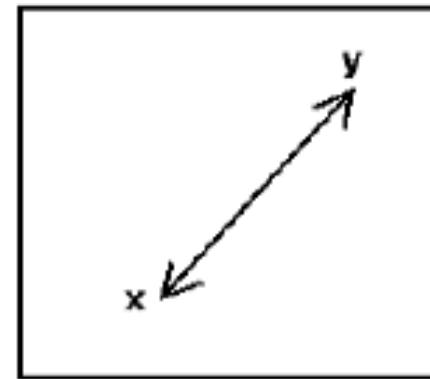
$$|X_1 - X_2| + |Y_1 - Y_2|$$



Manhattan

- Euclidean Distance

$$\sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2 + \dots}$$



Euclidean

- works if the points are arranged in the form of a grid
- if the input variables are not similar in type (such as age, gender, height, etc.)

- Euclidean is commonly used on dense, continuous variables.
- **There every dimension matters, and a 20 dimensional space can be challenging**
- if the input variables are of similar in type (such as width, height, depth etc.)

Measures of Dis/similarity

Distances : Special Cases of Minkowski

- $h = 1$: Manhattan (city block, L_1 norm) distance
 - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

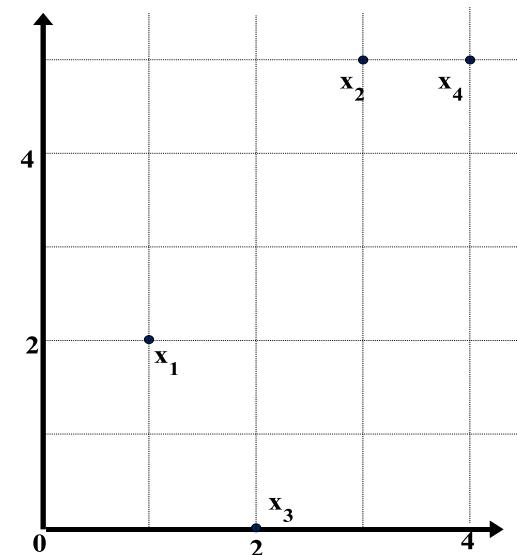
- $h = 2$: (L_2 norm) Euclidean distance

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

- $h \rightarrow \infty$. “supremum” (L_{\max} norm, L_∞ norm) distance.
 - This is the maximum difference between any component (attribute) of the vectors

$$d(i, j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f^p |x_{if} - x_{jf}|$$

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Measures of Dis/similarity

Distances : Special Cases of Minkowski

(Dissimilarity Matrices)

Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

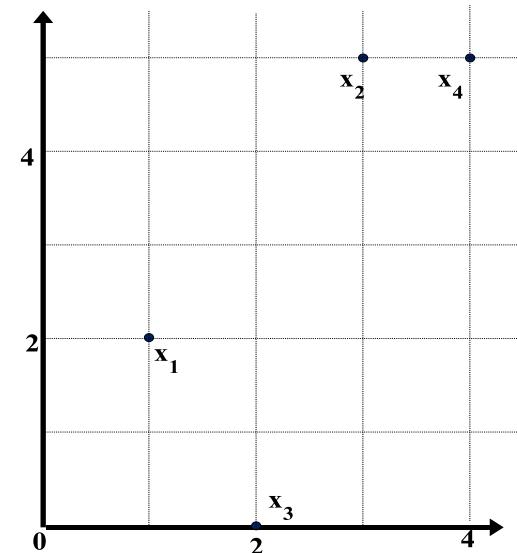
Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Measures of Dis/similarity

Distances : Attributes of Ordinal Type

- A database may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

An ordinal variable can be discrete or continuous

Order is important, e.g., rank

Can be treated like interval-scaled

replace x_{if} by their rank

map the range of each variable onto $[0, 1]$ by replacing i -th object
in the f -th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

compute the dissimilarity using methods for interval-scaled variables

$$d_{ij} = \frac{|r_i - r_j|}{n-1}$$

Measures of Dis/similarity

Distances : Attributes of Mixed Type – Gower's Distance

- A database may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

$$d(i, j) = \frac{\sum_{c=1}^n \omega_c \delta_{ij}^{(c)} d_{ij}^{(c)}}{\sum_{c=1}^n \omega_c \delta_{ij}^{(c)}}$$

$d(i, j)$ = dissimilarity between row i and row j

c = the c th column

n = number of columns in the dataset

ω_c = weight of c th column = $\frac{1}{\text{nrows in dataset}}$

$\delta_{ij}^{(c)}$ =
$$\begin{cases} 0 & \text{if column } c \text{ is missing in row } i \text{ or } j \\ 0 & \text{if column } c \text{ is asymmetric binary and both} \\ & \text{values in row } i \text{ and } j \text{ are 0} \\ 1 & \text{otherwise} \end{cases}$$

$d_{ij}^{(c)}$ (categorical) =
$$\begin{cases} 0 & \text{if } i \text{ and } j \text{ are equal in column } c \\ 1 & \text{otherwise} \end{cases}$$

$d_{ij}^{(c)}$ (continuous/ordinal) =
$$\frac{|\text{row } i \text{ in column } c - \text{row } j \text{ in column } c|}{\max(\text{column } c) - \min(\text{column } c)}$$

Measures of Dis/similarity

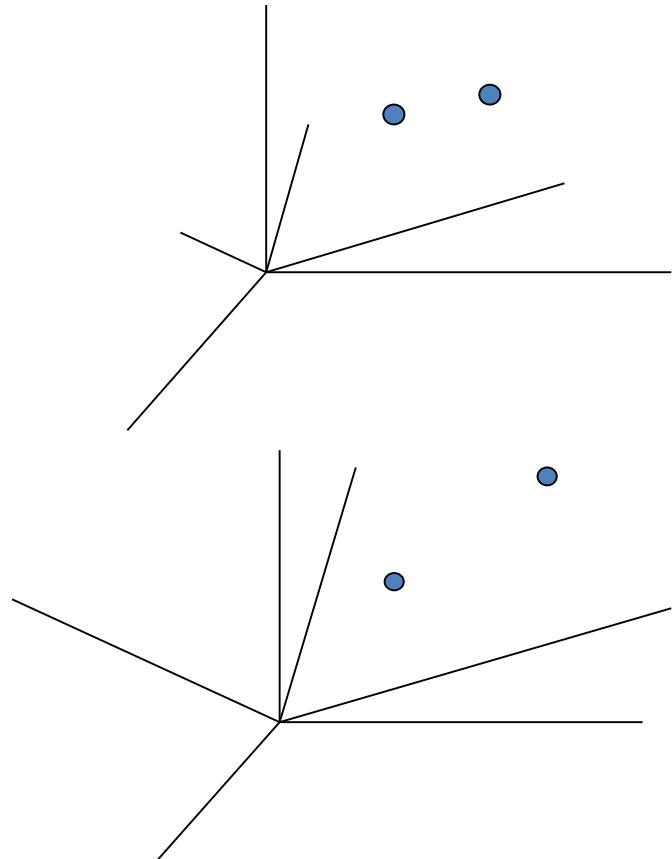
Curse of Dimensionality

- It uses all attributes to classify instances
- If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.
- ***Curse of dimensionality:*** nearest neighbor is easily misled when instance space is high-dimensional

Solution 1: weigh the attributes differently
 (use cross-validation to determine the weights)

Solution 2: eliminate the least relevant attributes (again, use cross-validation to determine which attributes to eliminate)

Stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes



Instance Based Learning

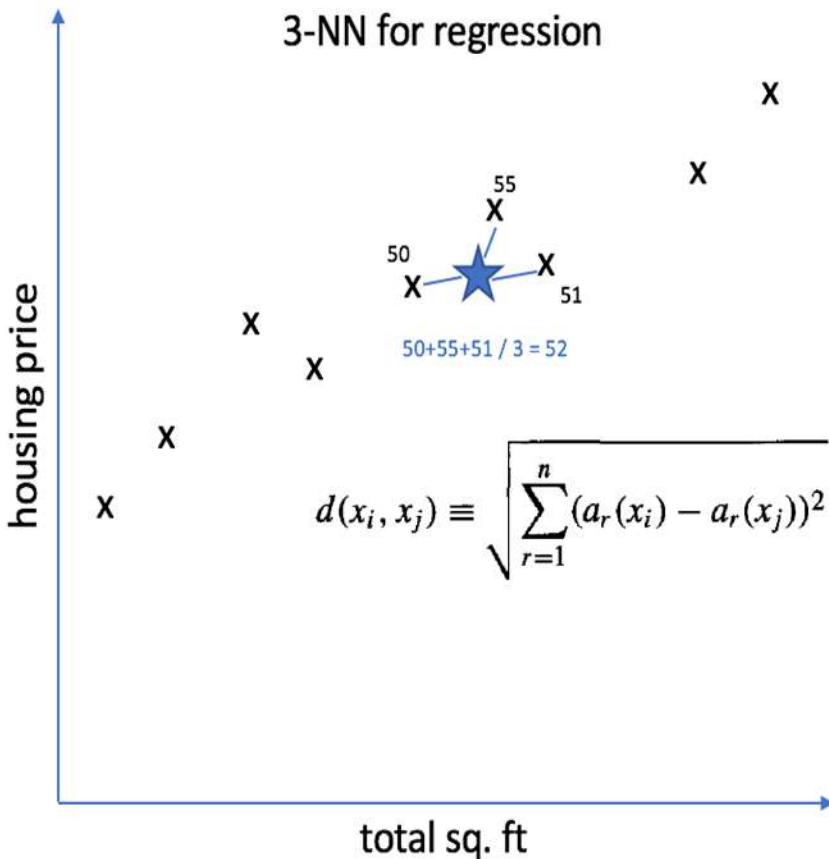


K-Nearest Neighbor Regression: Example

Size in feet ² (x_1)	No.of.Bed rooms (x_2)	Price (\$) in 1000's (y)
2104	4	460
1416	2	232
1534	3	315
1852	2	178
...

$$z = \frac{x - \mu}{\sigma}$$

The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around 0, with a standard deviation of 1



Source Credit : <https://www.jeremyjordan.me/k-nearest-neighbors/>

Instance Based Learning



K-Nearest Neighbor Regression: Example

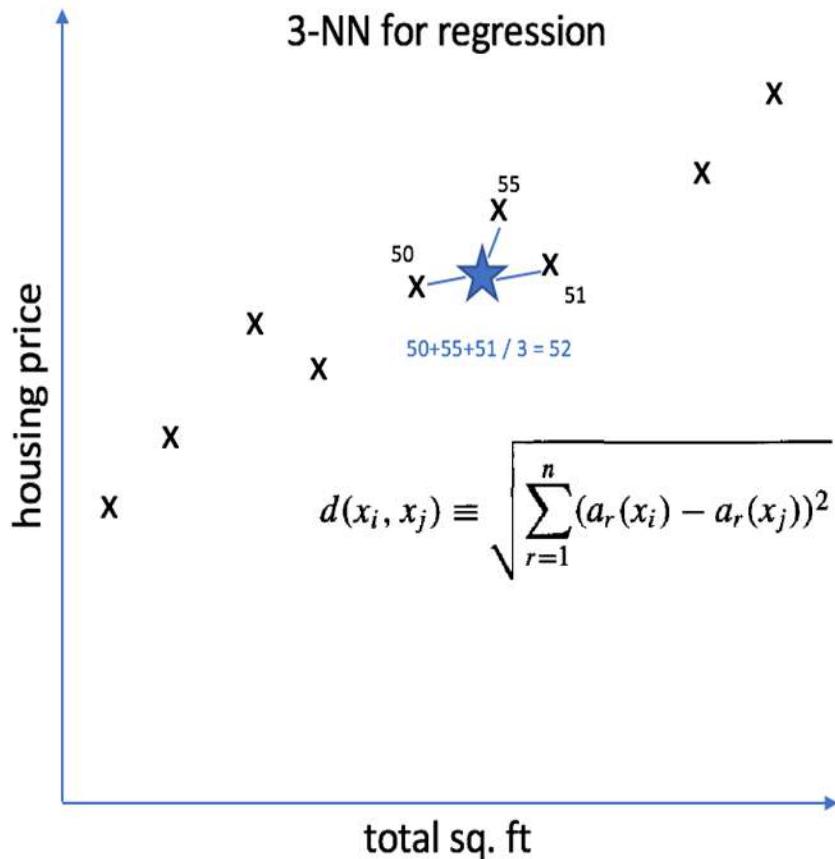
Size in feet ² (x_1)	No.of.Bed rooms (x_2)	Price (\$) in 1000's (y)
2104	4	460
1416	2	232
1534	3	315
1852	2	178
...

$$x_1 = \frac{\text{size} - 1000}{2000} \quad \begin{matrix} \text{Average value of } x_1 \\ \text{Maximum value of } x_1 \end{matrix}$$

$$x_2 = \frac{\#\text{bedrooms} - 2}{5} \quad \begin{matrix} \text{Maximum value of } x_1 \\ - \text{min value of } x_1 \end{matrix}$$

Min-Max scaler/Normalization

Feature scaling of features x_i consists of rescaling the range of features to scale the range in $[0, 1]$ or $[-1, 1]$ (Do not apply to x_0)



K-Nearest Neighbor : Algorithm

Approximating a discrete-valued function $f : \Re^n \rightarrow V$.

Training algorithm:

- For each training example $(x, f(x))$, add the example to the list *training-examples*

Classification algorithm:

- Given a query instance x_q to be classified,
 - Let $x_1 \dots x_k$ denote the k instances from *training-examples* that are nearest to x_q
 - Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Approximate a real-valued target function $f : \Re^n \rightarrow \Re$

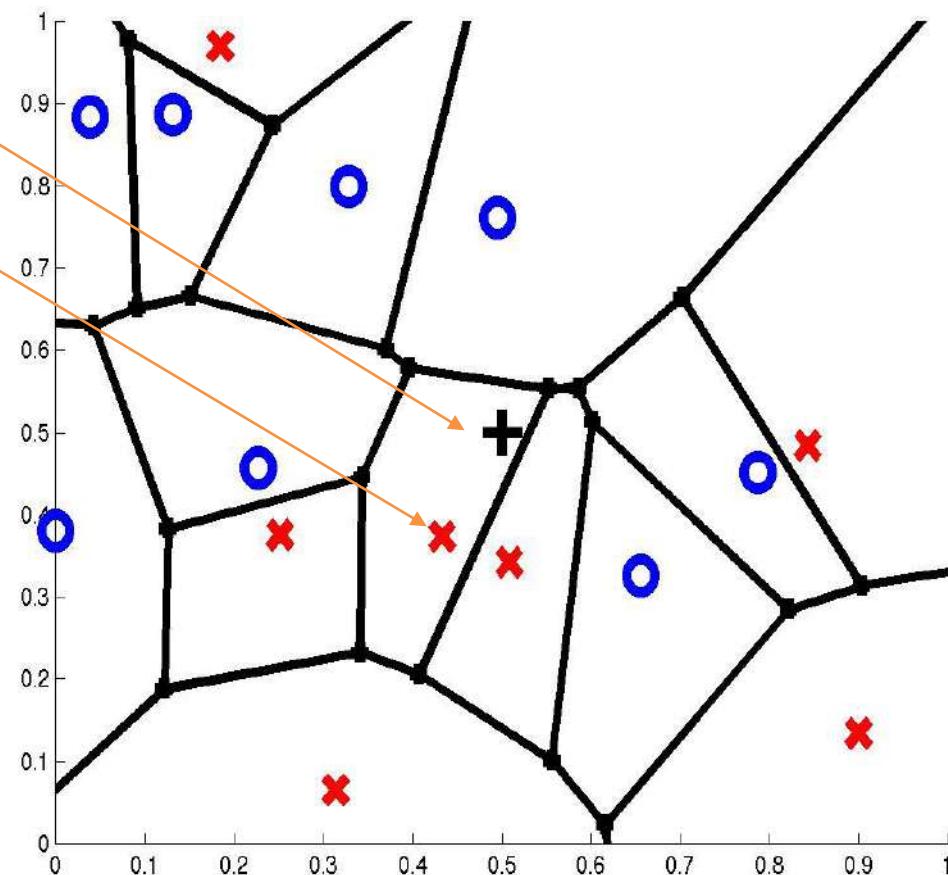
$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

Voronoi Diagram

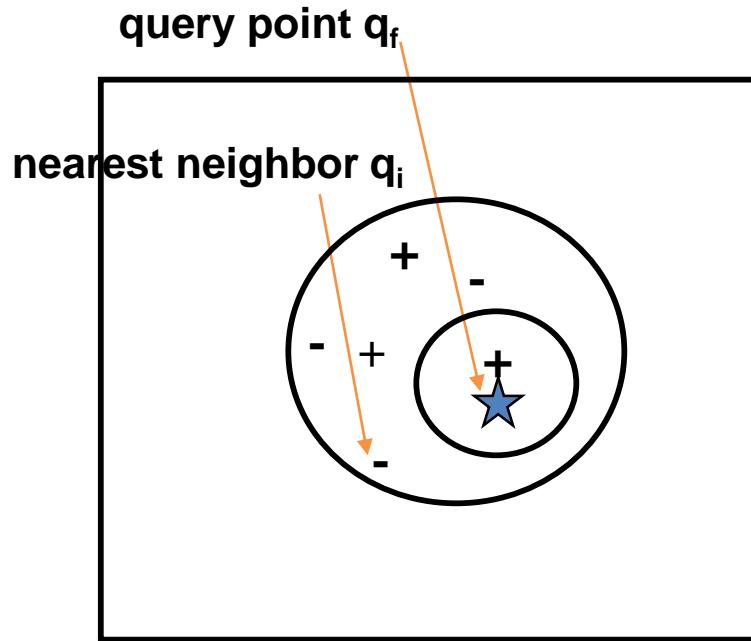
- Nearest neighbour approach induces a **Voronoi tessellation/partition** of the input space (all test points falling in a cell will get the label of the training input in that cell)
- For any sample, the nearest sample is determined by the closest Voronoi cell edge

query point q_f

nearest neighbor q_i



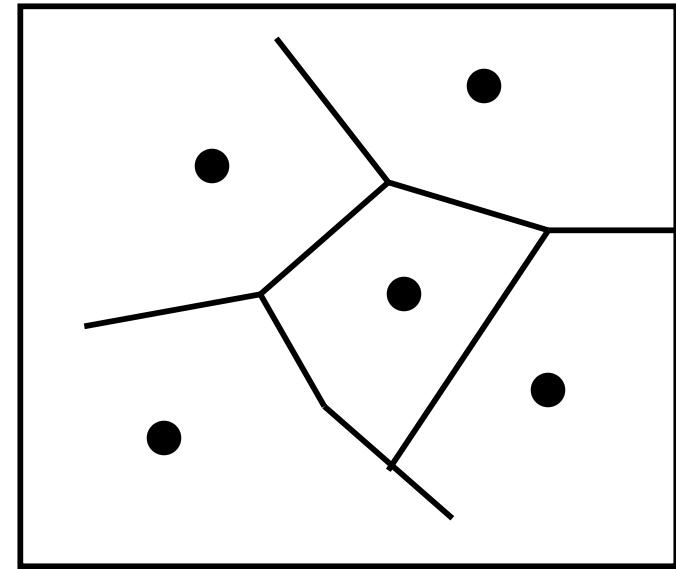
Voronoi Diagram



: query, x_q

1-NN: +

5-NN: -



Decision Surface
for 1-NN

K-Nearest Neighbor : Hyper parameter “K”

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

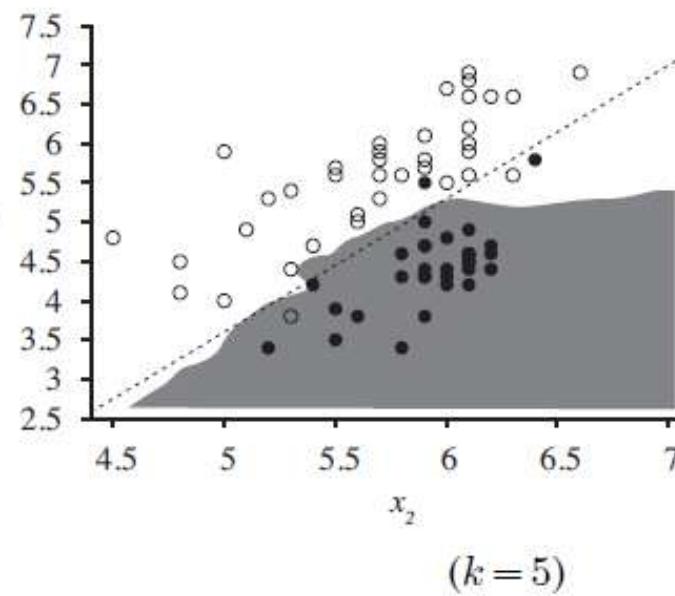
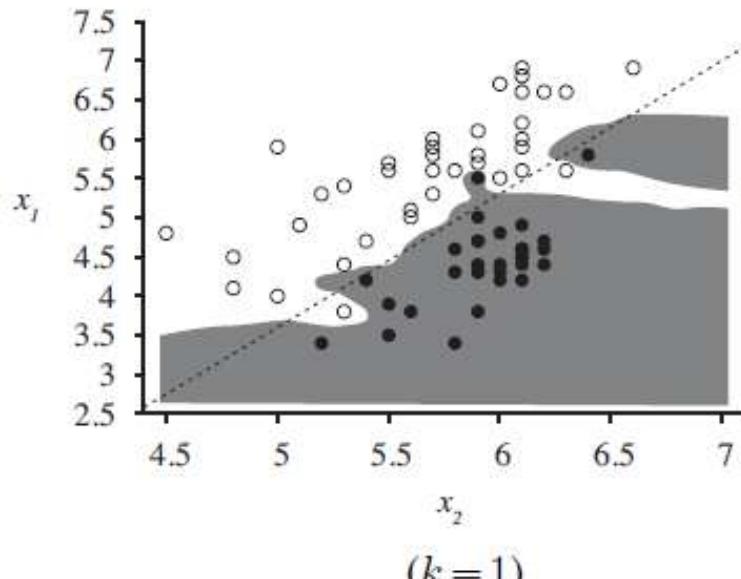


Figure 18.26 (a) A k -nearest-neighbor model showing the extent of the explosion class for the data in Figure 18.15, with $k = 1$. Overfitting is apparent. (b) With $k = 5$, the overfitting problem goes away for this data set.

K-Nearest Neighbor : Hyper parameter “K”

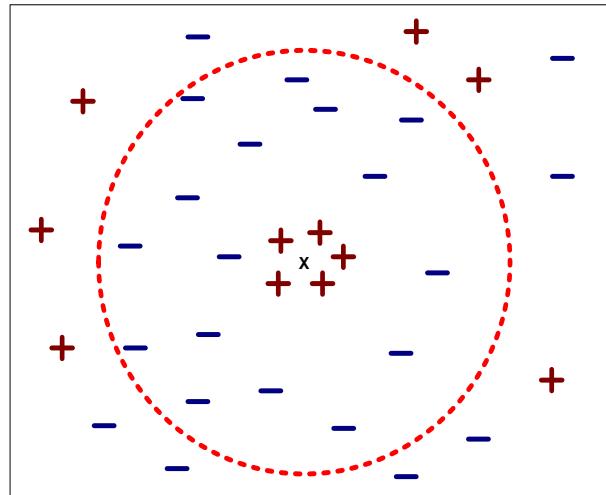
Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

Rule of thumb:

$$K = \sqrt{N}$$

N: number of training points

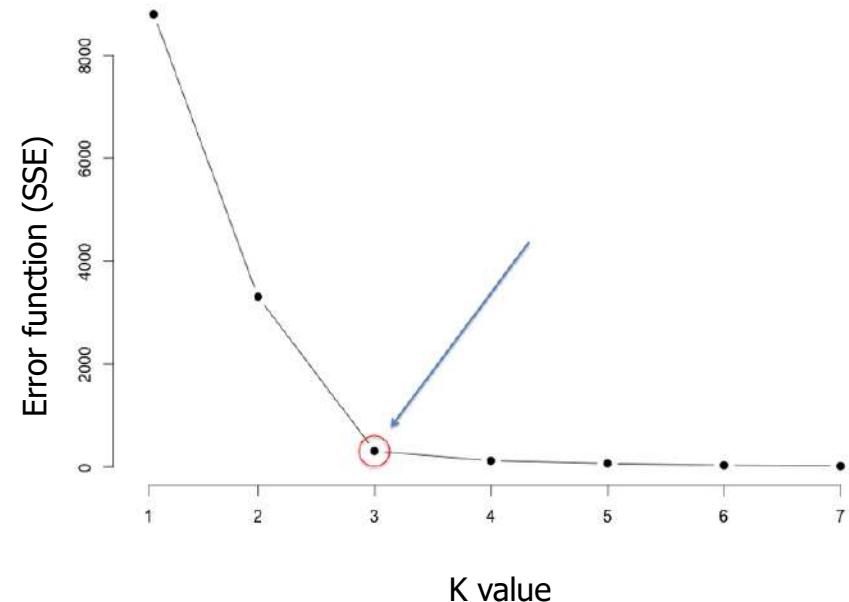


Instance Based Learning



K-Nearest Neighbor : Hyper parameter “K” : Elbow Method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to ‘0’



Instance Based Learning



K-Nearest Neighbor : Variation

- Locally Weighted K-NN algorithm or Distance Weighted K-NN algorithm

contribution of each of the k nearest neighbors is weighted accorded to their distance to x_q

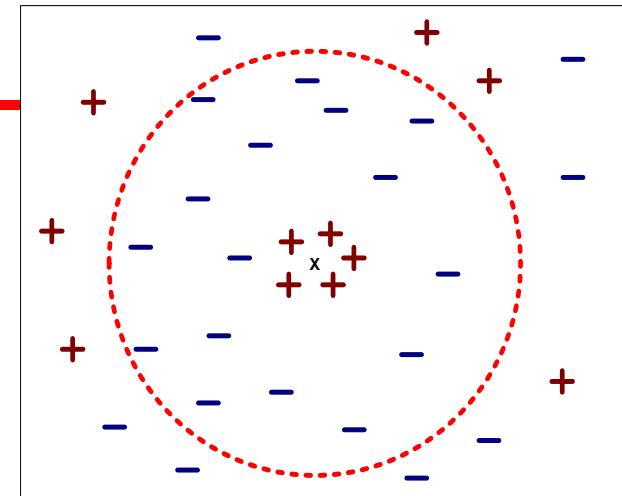
- discrete-valued target functions

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

- continuous-valued target function:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$



Kernel Functions

$$w_i = K(d(x_q, x_i))$$

$$K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$$

$$K(d(x_q, x_i)) = 1/(d_0 + d(x_q, x_i))^2$$

$$K(d(x_q, x_i)) = \exp(-(d(x_q, x_i)/\sigma_0)^2)$$

$d(x_q, x_i)$ is the distance between x_q and x_i

Exercise - 1

Consider the following training set in 2-dimensional Euclidean space. What is the class of the point (1, 1) if 7NN classifier is considered? If the value of K is reduced whether the class will change? (Consider K=3 and K=5). What should be the final class if the above 3 values of K are considered?

Point	Coordinate	Class
X1	(-1, 1)	Negative
X2	(0, 1)	Positive
X3	(0, 2)	Negative
X4	(1, -1)	Negative
X5	(1, 0)	Positive
X6	(1, 2)	Positive
X7	(2, 2)	Negative
X8	(2, 3)	Positive

Distance from (1,1)
2
1
1.414
2
1
1
1.41
2.236

Point	Distance	Class
X2	1	Positive
X5	1	Positive
X6	1	Positive
X7	1.41	Negative
X3	1.414	Negative
X1	2	Negative
X4	2	Negative

Exercise - 2

Based on the information given in the table below, find the customer type of $x_q = C4$ using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$.

Apply min-max normalization on income to obtain [0, 1] range. Consider profession and Region as nominal. Consider Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute .

Customer	Income	Profession	Region	Locality	Category
C0	60000	Doctor	Hindi	Village	L1
C1	70000	Doctor	Bengali	Village	L2
C2	60000	Carpenter	Hindi	Suburban	L2
C3	80000	Doctor	Bhojpuri	Metropolitan	L2
C5	80000	Data Scientist	Hindi	Small Town	L1
C4	50000	Data Scientist	Hindi	Small Town	

Exercise - 2

Based on the information given in the table below, find the customer type of $x_q = C4$ using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$.

Apply min-max normalization on income to obtain [0, 1] range. Consider profession and Region as nominal. Consider Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Locality
C0	0.33	Doctor	Hindi	1
C1	0.67	Doctor	Bengali	1
C2	0.33	Carpenter	Hindi	3
C3	1	Doctor	Bhojpuri	4
C5	1	Data Scientist	Hindi	2
C4	0	Data Scientist	Hindi	2

$d(C4, C2)$

= distance w.r.t to numerical attributes + distance w.r.t Nominal attributes + distance w.r.t ordinal attribute +

= $\text{dist}(\text{Income}_{c4}, \text{Income}_{c2}) + \text{dist}(\text{Profession}_{c4}, \text{Profession}_{c2}) + \text{dist}(\text{Region}_{c4}, \text{Region}_{c2}) + \text{dist}(\text{Locality}_{c4}, \text{Locality}_{c2})$

$$= 0.33 + 1 + 0 + (3-2)/(4-1) = 1.67$$

Similarly calculate for all other instances

Correction : You need not use Gower's distances unless specified in the question. In the live class this was used & discussed for ordinal and categorical attribute without any weights or normalization of the answers.

Students are advised to use the distance measure $(p-m)/p$ for categorical attributes' calculation and treat the encoded ordinal attribute as numerical attribute to apply the same Euclidean distance as instructed in your first semester course Data Mining. As per this approach formula in the next slides recalculated based on this.

Exercise - 2

Based on the information given in the table below, find the customer type of $x_q = C4$ using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$.

Apply min-max normalization on income to obtain [0, 1] range. Consider profession and Region as nominal. Consider Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 → 0	L1	0.97	1.06
C1	0.67	Doctor	Bengali	1 → 0	L2	1.75	0.33
C2	0.33	Carpenter	Hindi	3 → 0.67	L2	0.97	1.06
C3	1	Doctor	Bhojpuri	4 → 1	L2	2.2	0.21
C5	1	Data Scientist	Hindi	2 → 0.33	L1	1	1
C4	0	Data Scientist	Hindi	2 → 0.33		0 (NA)	

$d(C4, C2)$

= distance w.r.t to numerical attributes + distance w.r.t Nominal attributes

= Euclidean distance on (Income, Scaled Locality) + Categorical distance on attributes (Profession, Region)

= $\sqrt{(Income_{c4} - Income_{c2})^2 + (Locality_{c4} - Locality_{c2})^2}$ + (No. of Categorical features -

No. of Matches between C4 & C2)/(No. of Categorical Features)

= $\sqrt{(0 - 0.33)^2 + (0.33 - 0.67)^2} + \frac{2-1}{2}$

= $0.47 + 0.5 = 0.97$

Exercise - 2

Based on the information given in the table below, find the customer type of $x_q = C4$ using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$.

Apply min-max normalization on income to obtain [0, 1] range. Consider profession and Region as nominal. Consider Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 → 0	L1	0.97	1.06
C1	0.67	Doctor	Bengali	1 → 0	L2	1.75	0.33
C2	0.33	Carpenter	Hindi	3 → 0.67	L2	0.97	1.06
C3	1	Doctor	Bhojpuri	4 → 1	L2	2.2	0.21
C5	1	Data Scientist	Hindi	2 → 0.33	L1	1	1
C4	0	Data Scientist	Hindi	2 → 0.33		0 (NA)	

Point	Weighted distance	Class
C5	1	L1
C0	1.03	L1
C2	1.06	L2
C1	0.58	L2
C3	0.46	L2

Inference : Without weighted KNN, L2 is the majority voted class.

But with weighted distances, L1 class instances are more similar to C4

Nearest-Neighbor Classifiers: Issues

- The value of k , the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Expensive, Slow at query time
 - To determine the nearest neighbour of a query point q , must compute the distance to all N training examples
 - + Pre-sort training examples into fast data structures (kd-trees)
 - + Remove redundant data (condensing)
- Storage Requirements
 - Must store all training data \mathbf{P}
 - + Remove redundant data (condensing)

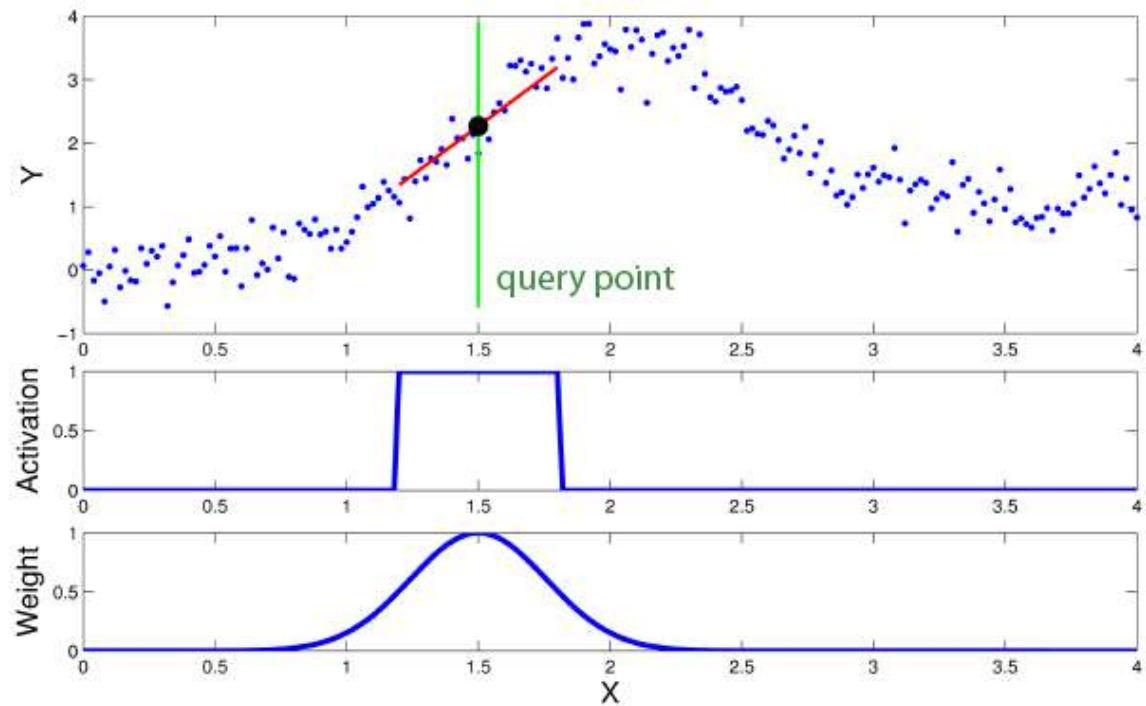
When to Consider Nearest Neighbors

- Suitable for Low dimensional datasets
- Local information can yield highly adaptive behavior
- Lots of training data (distance-weighted KNN)
 - Training is very fast
- Learn complex target functions
 - Do not lose information
- Noisy training data (distance-weighted KNN)
 - by taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples.

Locally Weighted Regression

Locally Weighted Regression

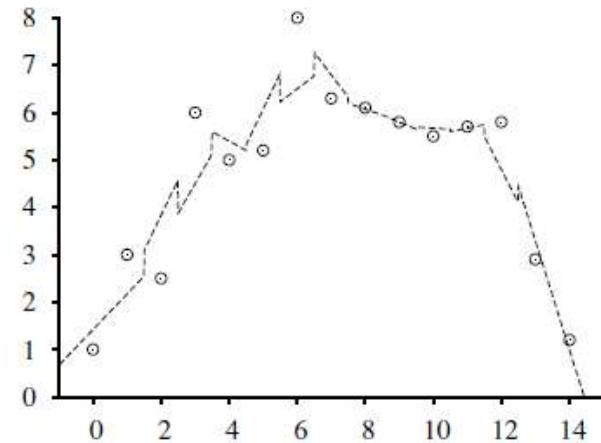
- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function
- **Residual** is the error in approximating the target function. $\hat{f}(x) - f(x)$
- **Kernel function** is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function K such that $w_i = K(d(x_i, x_q))$.



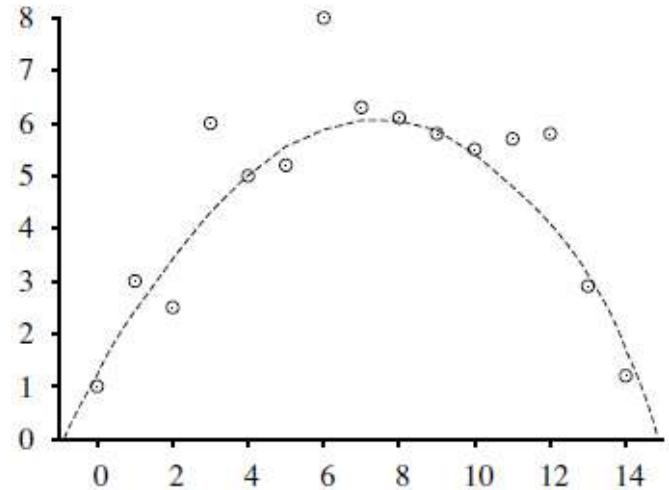
Locally Weighted Regression

- The nearest-neighbor approaches can be thought of as approximating the target function $f(\mathbf{x})$ at the single query point $\mathbf{x} = \mathbf{x}_q$
- Locally weighted regression is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding \mathbf{x}_q
- Approximate the target function in the neighborhood surrounding x , using a linear function, a quadratic function, a multilayer neural network, or some other functional form.

3-nearest-neighbors linear regression



Locally weighted regression with a quadratic kernel



Locally weighted linear regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \dots + w_n a_n(x)$$

- methods like **gradient descent** can be used to calculate the coefficients w_0, w_1, \dots, w_n to minimize the error in fitting such linear functions
 - ANNs require a global approximation to the target function
 - here, just a local approximation is needed
- ⇒ the error function has to be redefined

Locally weighted linear regression

- ➊ possibilities to redefine the error criterion E
 1. Minimize the squared error over just the k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2$$

2. Minimize the squared error over the entire set D , while weighting the error of each training example by some decreasing function K of its distance from x_q

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

Locally weighted linear regression

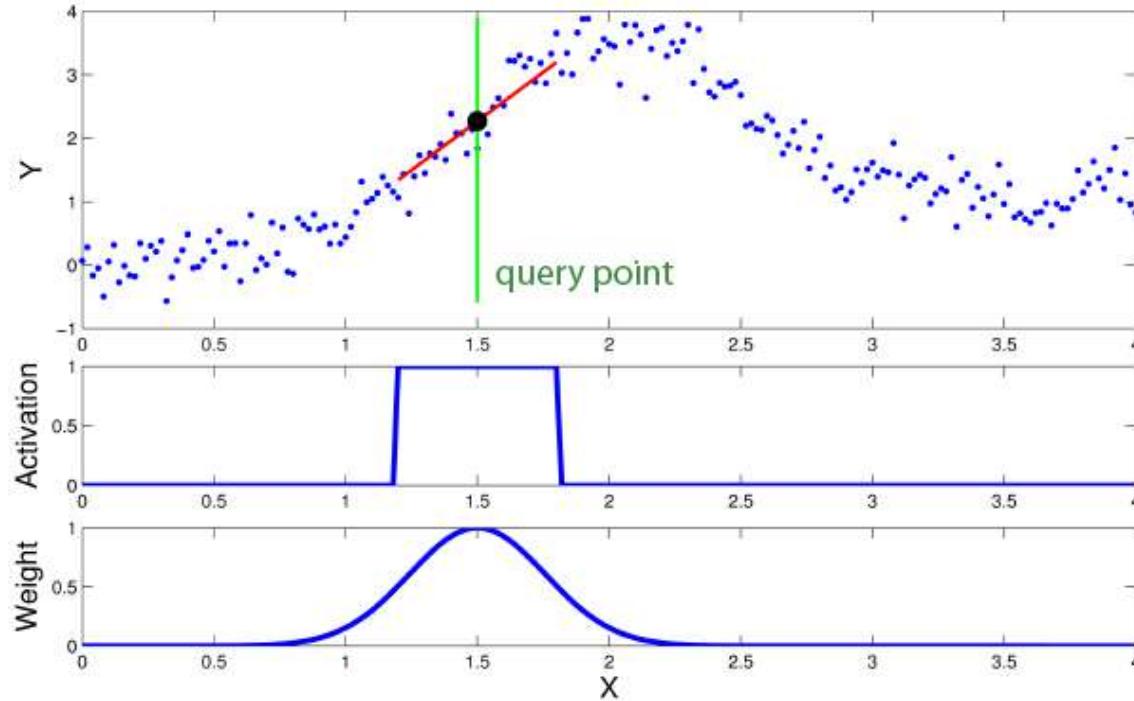
- For a given query point \mathbf{x}_q we solve the following weighted regression problem using gradient descent training rule:

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest nbrs of } x_q} K(d(x_q, x)) (f(x) - \hat{f}(x)) a_j(x)$$

This is used in GD formula for calculation of Delta Wi

- Note that we need to solve a new regression problem for *every* query point—that's what it means to be *local*.
- In ordinary linear regression, we solved the regression problem once, globally, and then used the same h_w for any query point.

Example



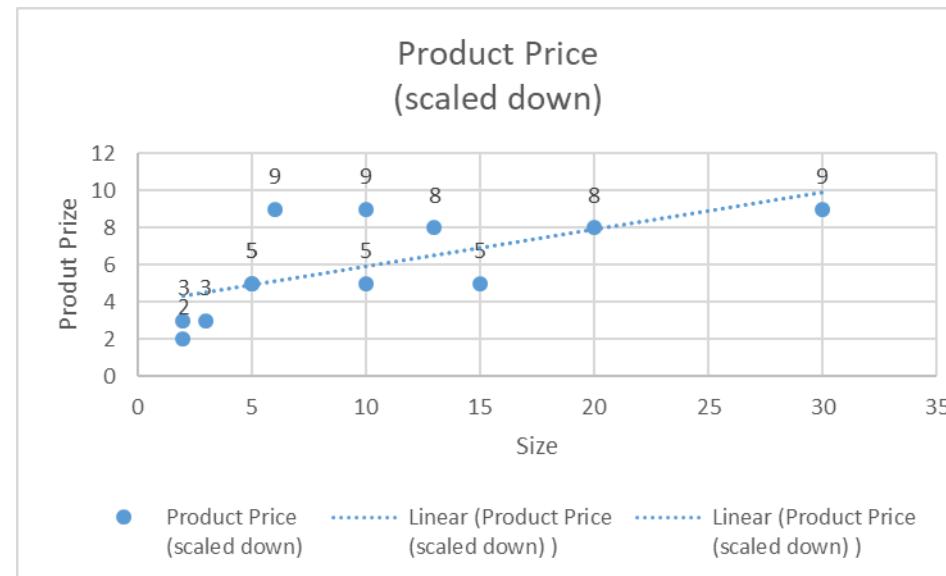
- The upper graphic shows the set of data points (x,y) (blue dots), query point (green line), local linear model (red line) and prediction (black dot).
- The graphic in the middle shows the activation area of the model.
- The corresponding weighting kernel (receptive field) is shown in the bottom graphic.

Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q, x_i)) = \exp(-d(x_q, x_i)^2 / 2b^2)$ where $b=1$. Use the linear regression of the form $y = 3.5 + 0.8 \text{ Size}$ and the learning rate of 0.5 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8

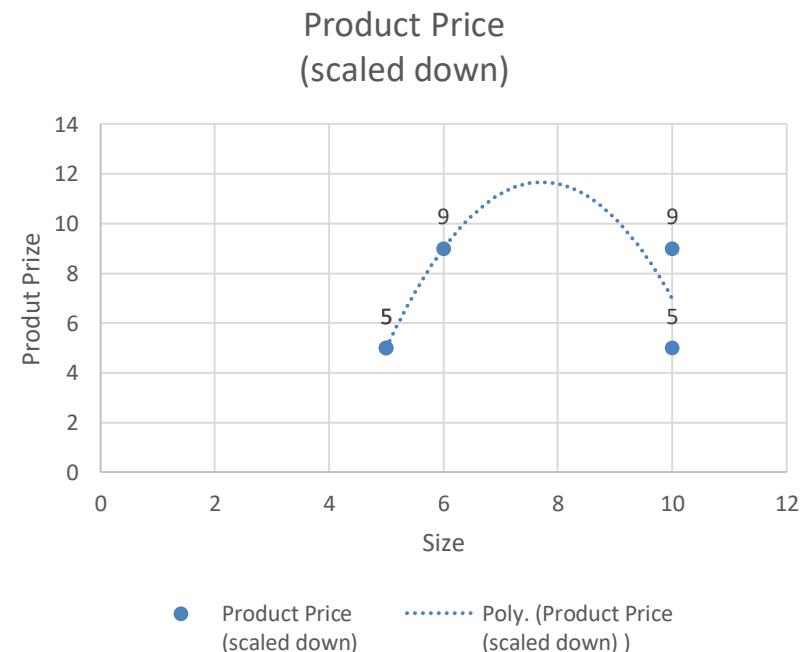


Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q, x_i)) = \exp(-d(x_q, x_i)^2 / 2b^2)$ where $b=1$. Use the linear regression of the form $y = 3.5 + 0.8 \text{ Size}$ and the learning rate of 0.5 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8

Considering the 5-NN of the query point
 $x = \text{size} = 7$



Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the price of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q, x_i)) = \exp(-d(x_q, x_i)^2 / 2b^2)$ where $b=1$. Use the linear regression of the form Price = $3.5 + 0.8 \text{ Size}$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

Product Price			
Size	(scaled down)	$d(x_q, x_i)$	$K(d(x_q, x_i))$
10	5	3	0.01
5	5	2	0.14
2	2	5	0
3	3	4	0
5	5	2	0.14
15	5	8	0
30	9	23	0
20	8	13	0
6	9	1	0.61
2	3	5	0
10	9	3	0.01
13	8	6	0

Apply gradient descent where the gradient is weighed by the kernel output.

Answer :
Updated weights after 1st iteration
 $w_0 = 3.57$, $w_1 = 1.17$.

Price Prediction	
11.74049	After 1st iteration of GD
9.1	Before GD



Machine Learning

DSE CLZG565

Ensemble Learning

Raja vadhana P

Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: Slides of Prof. Chetana, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 10 - Ensemble

- Bagging
- Boosting

Ensemble Learning

The Single Model Philosophy

- Motivation: Occam's Razor
 - “one should not increase, beyond what is necessary, the number of entities required to explain anything”
- Infinitely many models can explain any given dataset
 - Might as well pick the smallest one...

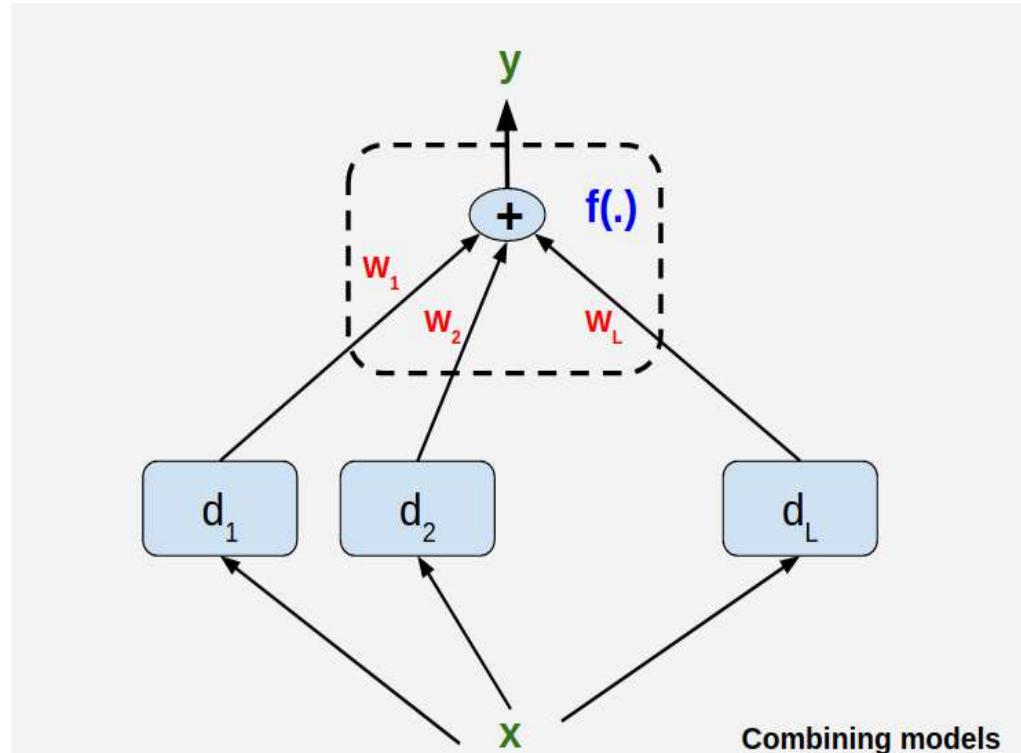
Ensemble Philosophy

- No Free Lunch Theorem: There is no algorithm that is always the most accurate
- Each learning algorithm dictates a certain model that comes with a set of assumptions
 - Each algorithm converges to a different solution and fails under different circumstances
 - The best tuned learners could miss some examples and there could be other learners which works better on (may be only) those !
 - In the absence of a single expert (a superior model) , a committee (combinations of models) can do better !
 - A committee can work in many ways ...

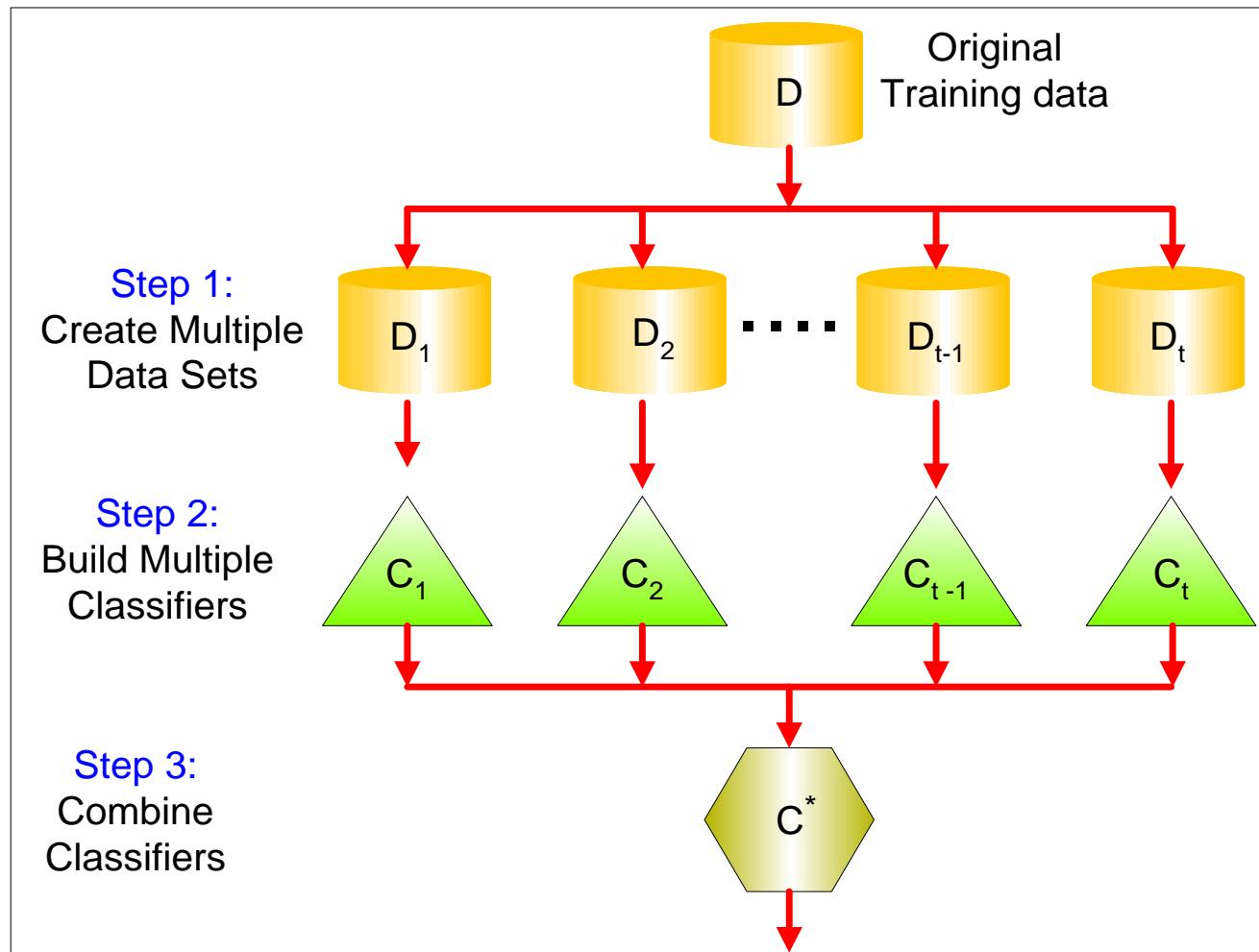
Weak learner does only slightly better than random guessing

Committee of Models

- Committee Members are base learners !
- Major challenges dealing with this committee
 - Expertise of each of the members (Does it help / not?)
 - Combining the results from the members for better performance

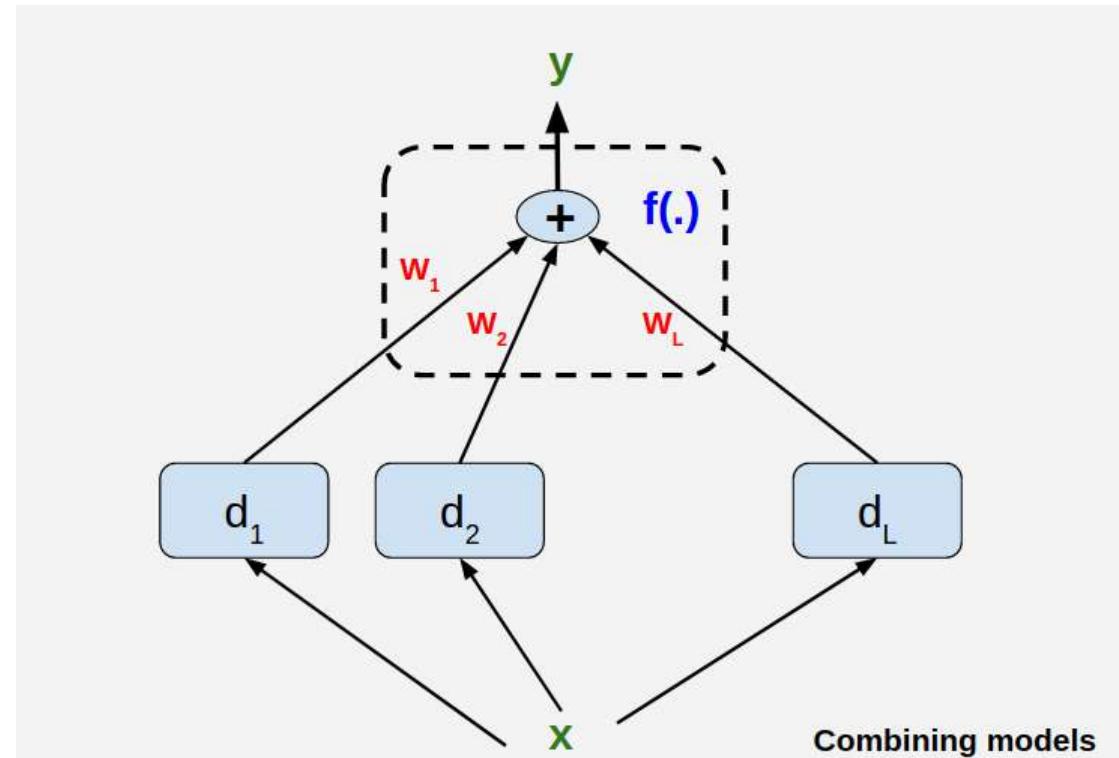


General Approach



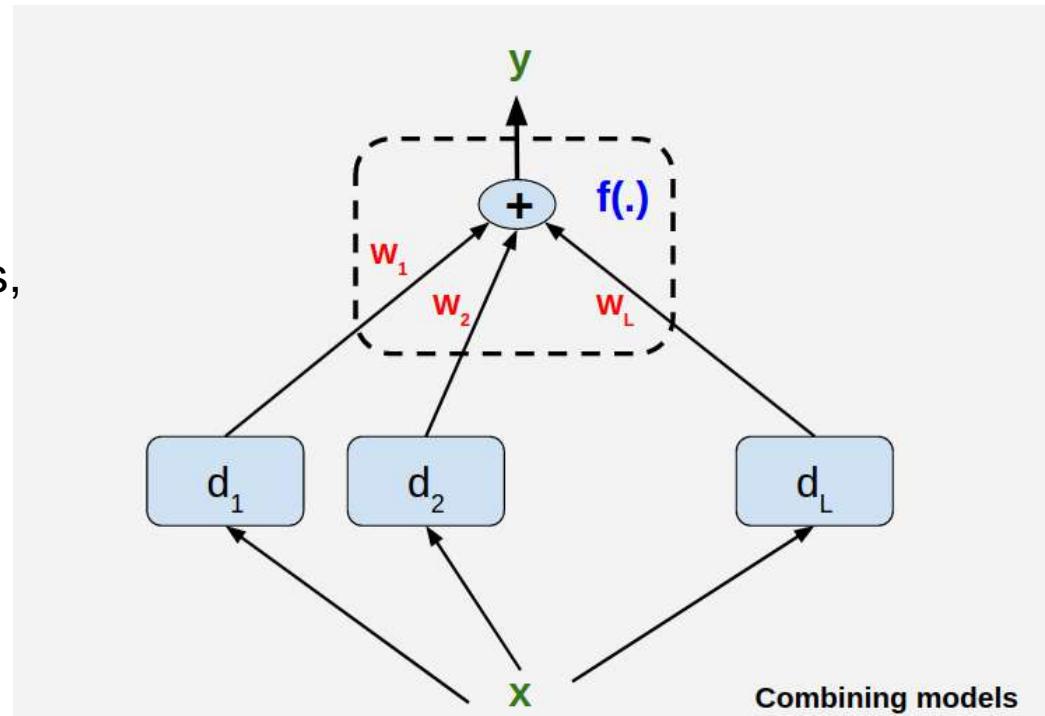
Issue 1 : On the members (Base Learners)

- It does not help if all learners are good/bad at roughly same thing
 - Need Diverse Learners



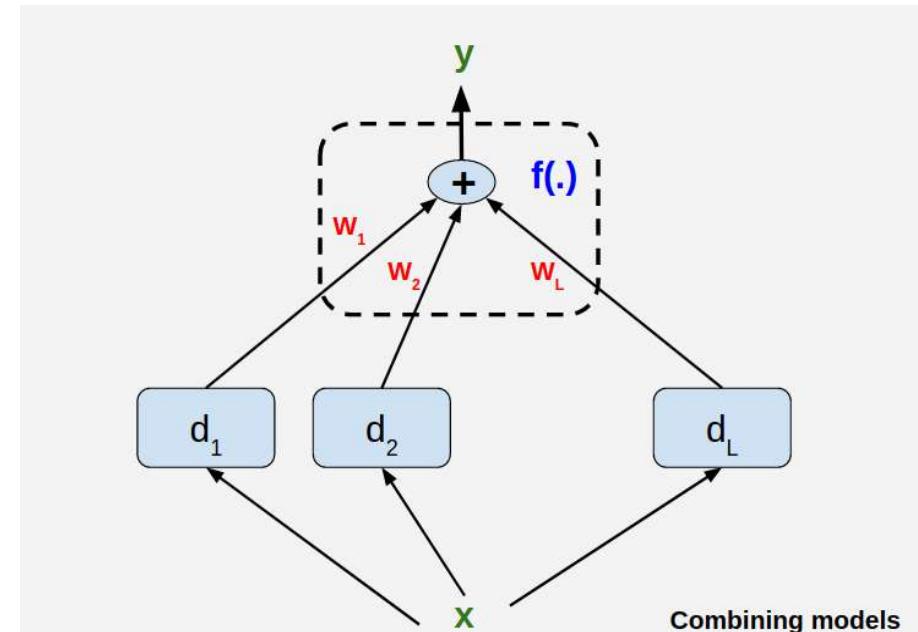
Issue 1 : On the members (Base Learners)

- Use Different Algorithms
 - Different algorithms make different assumptions
- Use Different Hyper parameters,
 - E.g. vary the structure of neural nets

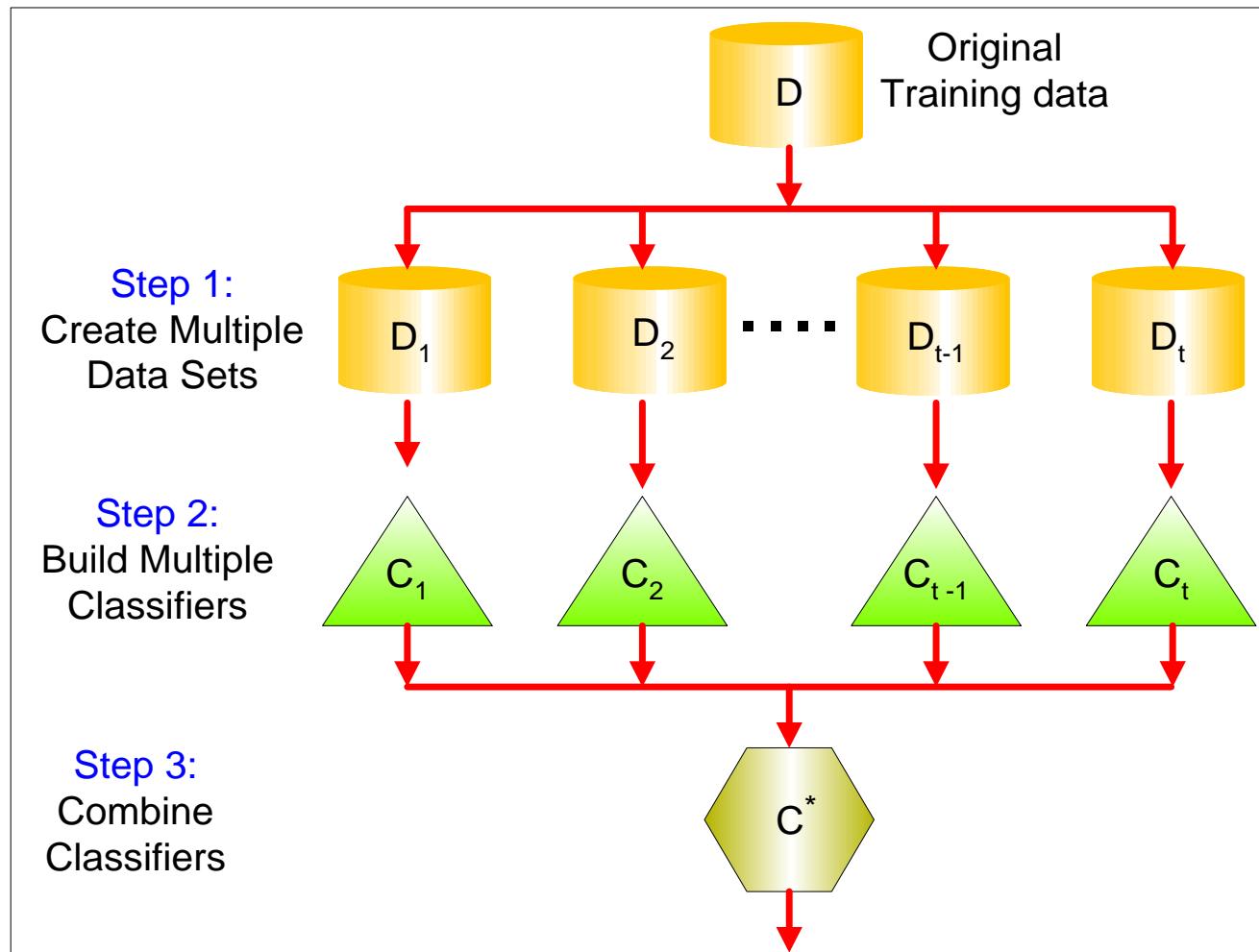


Issue 1 : On the members (Base Learners)

- Different input representations
 - Uttered words + video
information of speakers clips
 - image + text annotations
- Different training sets
 - Draw different random samples of data
 - Partition data in the input space and have learners specialized in those spaces (mixture of experts)



General Approach



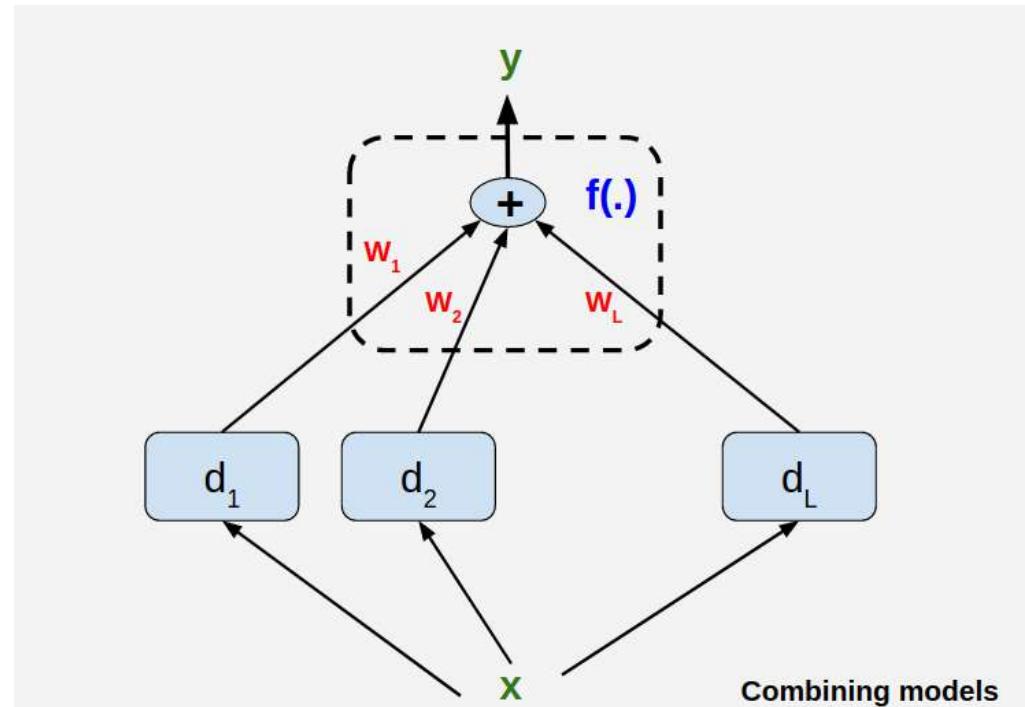
Issue -2 : Combining Results Base Learners

$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

A Simple Combination Scheme:

$$y = \sum_{j=1}^L w_j d_j$$

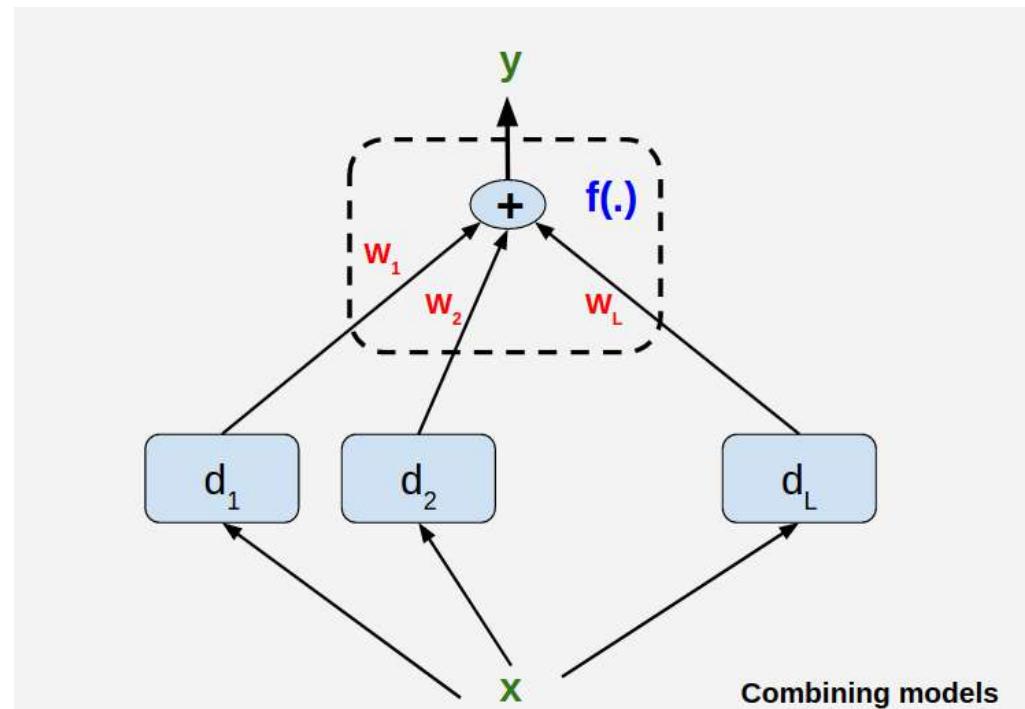
$$w_j \geq 0 \text{ and } \sum_{j=1}^L w_j = 1$$



Issue -2 : Combining Results Base Learners

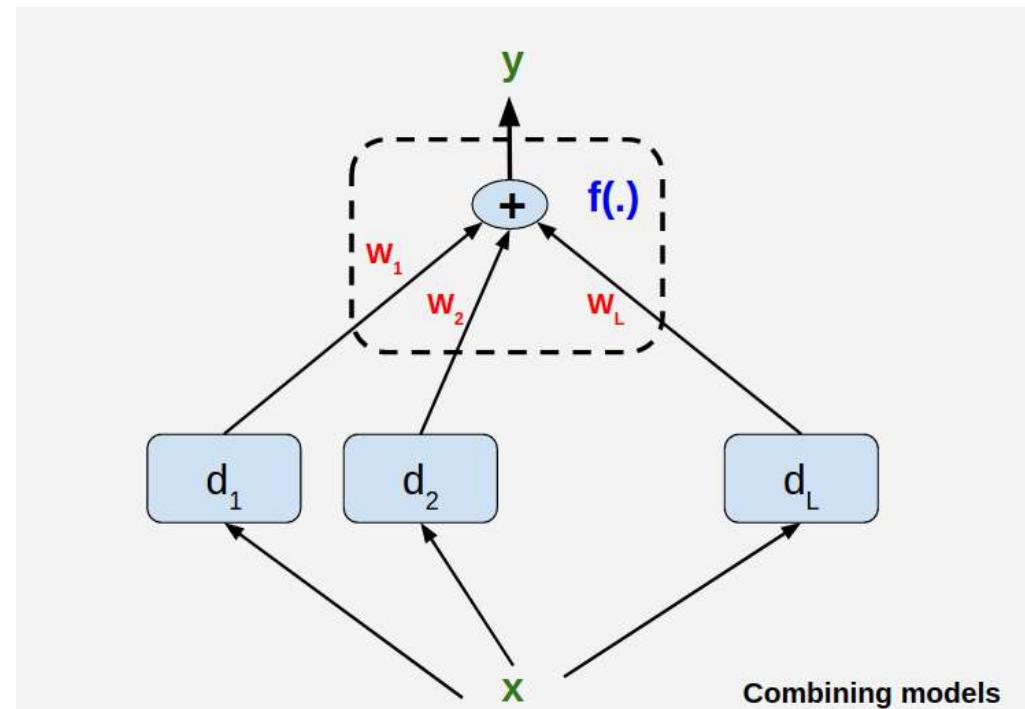
$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L} \sum_{j=1}^L d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$



Issue -2 : Combining Results Base Learners

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	0.5	0.3
Avg	0.2	0.5	0.4
Median	0.2	0.5	0.4
Minimum	0.0	0.4	0.2
Maximum	0.4	0.6	0.4
Product	0.0	0.12	0.032



When Ensemble Methods Work?

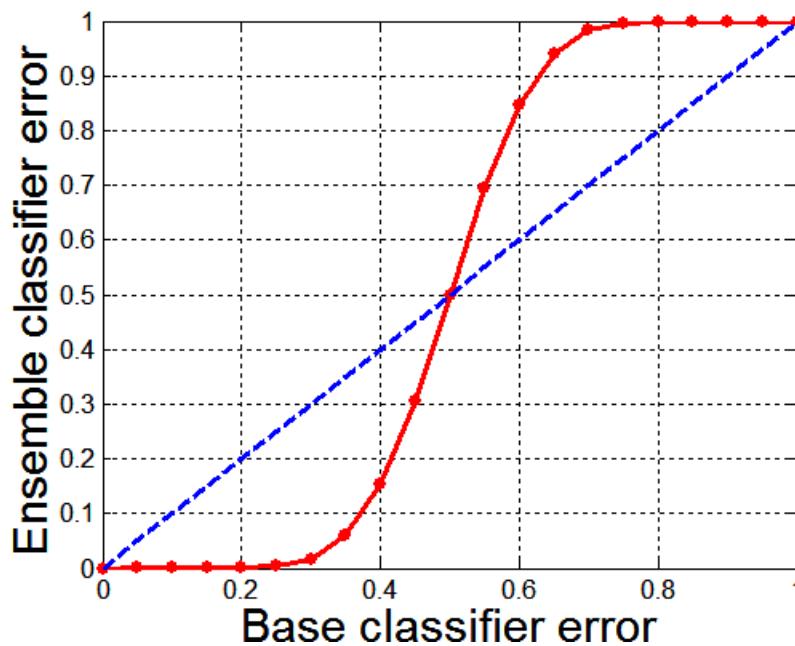
- Ensemble classifier performs better than the base classifiers when e is smaller than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
 - Base classifiers should be independent of each other
 - Base classifiers should do better than a classifier that performs random guessing

Example: Why Do Ensemble Methods Work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\epsilon = 0.35$
 - Majority vote of classifiers used for classification
 - If all classifiers are identical:
 - ◆ Error rate of ensemble = $\epsilon (0.35)$
 - If all classifiers are independent (errors are uncorrelated):
 - ◆ Error rate of ensemble = probability of having more than half of base classifiers being wrong

Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
 - All base classifiers are independent of each other
 - All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.

Rationale for Ensemble Learning

- Ensemble Methods work best with **unstable base classifiers**
 - Classifiers that are sensitive to minor perturbations in training set, due to *high model complexity*
 - Examples: Unpruned decision trees, ANNs, ...

Types of Ensemble Methods

- By manipulating training set
 - Example: bagging, boosting, random forests

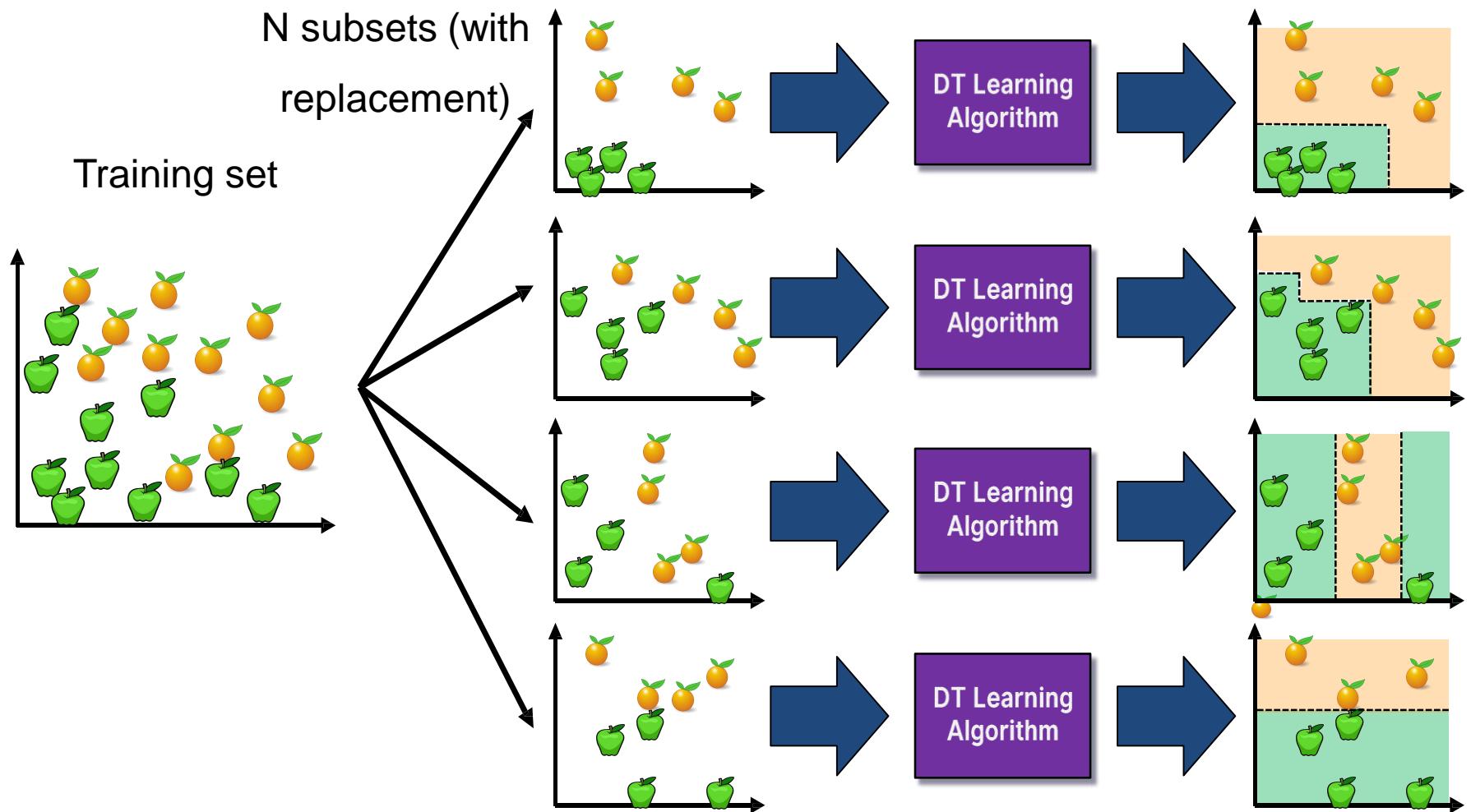
- By manipulating input features
 - Example: random forests

Bagging

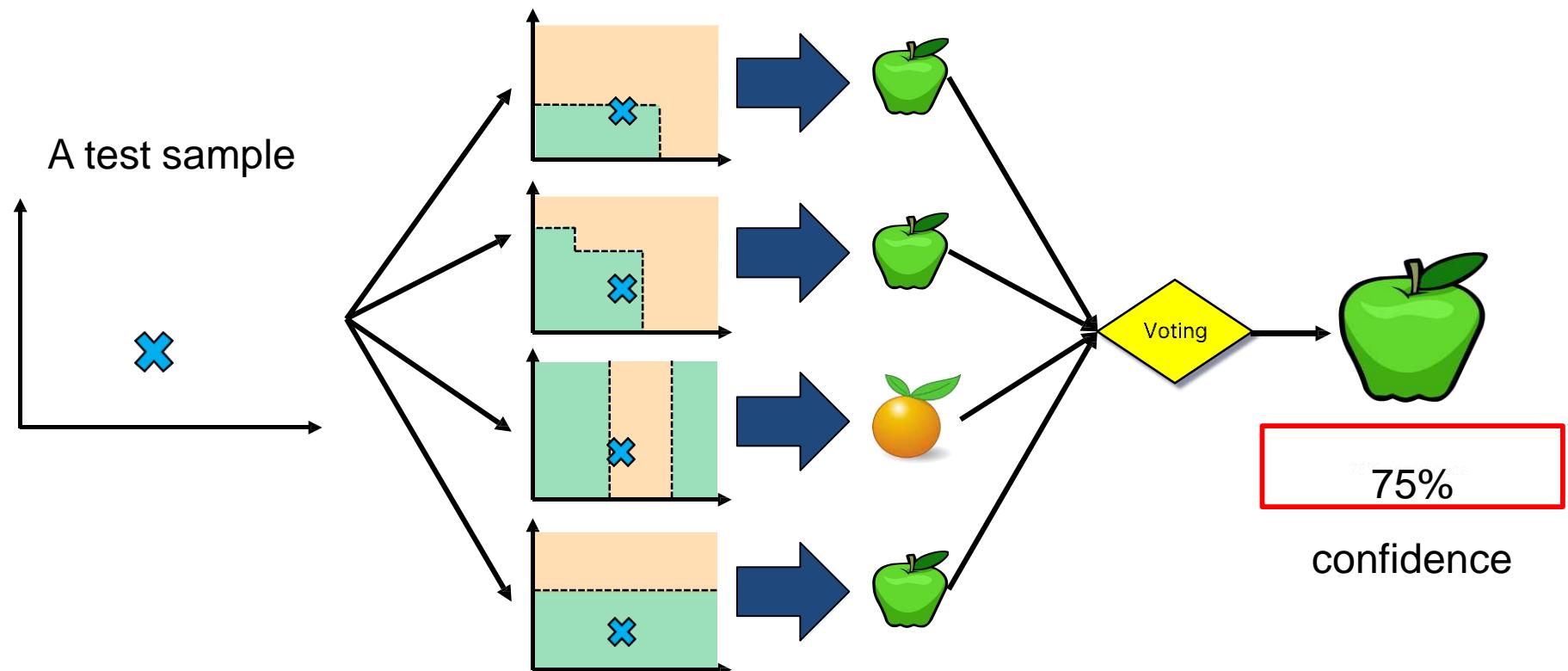
Bootstrap Sampling

- In Bootstrapping sampling technique, multiple subsets are created from the original dataset, selecting observations with replacement.
- Given data: $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
- For $m=1:M$ (use validation data to pick M)
- Obtain bootstrap sample D_m from the training data D
- Build a model $G_m(x)$ from bootstrap data D_m

Bagging at training time



Bagging at inference time



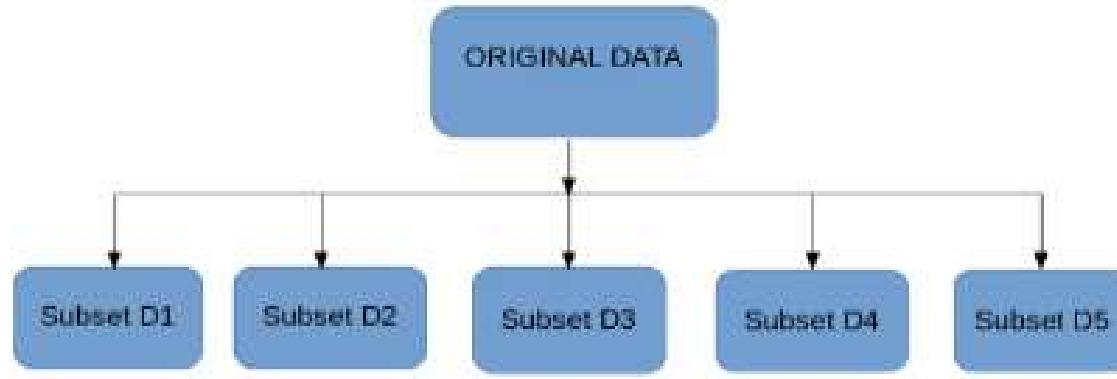
Bootstrap Sampling

- Bootstrap sampling: sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Probability of a training instance being selected in a bootstrap sample is:
 - $1 - (1 - 1/n)^n$ (n : number of training instances)
 - ~ 0.632 when n is large

Bagging (Bootstrap Aggregating)



- The size of subsets created for bagging may be less than the original set.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.

The Bagging Model

- Regression

$$\hat{y} = \frac{1}{M} \sum_{m=1}^M G_m(\mathbf{x})$$

- Classification:
 - Vote over classifier outputs

$$G_1(\mathbf{x}), \dots, G_M(\mathbf{x})$$

Bagging Algorithm

Algorithm 4.5 Bagging algorithm.

- 1: Let k be the number of bootstrap samples.
 - 2: **for** $i = 1$ to k **do**
 - 3: Create a bootstrap sample of size N , D_i .
 - 4: Train a base classifier C_i on the bootstrap sample D_i .
 - 5: **end for**
 - 6: $C^*(x) = \operatorname{argmax}_y \sum_i \delta(C_i(x) = y).$
 $\{\delta(\cdot) = 1 \text{ if its argument is true and } 0 \text{ otherwise.}\}$
-

Bagging Example

Consider 1-dimensional data set:

Original Data:

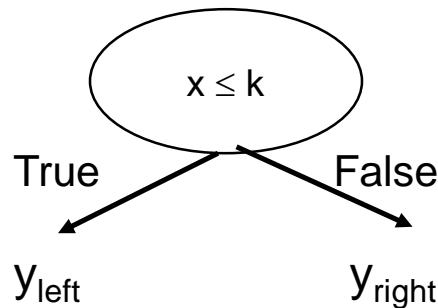
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Classifier is a decision stump

Decision rule: $x \leq k$ versus $x > k$

$x < 0.35$ or $X \geq 0.75$.

Split point k is chosen based on entropy



Decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). **Decision stump** makes a prediction based on the value of just a single input feature. Sometimes they are also called 1-rules

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$X \leq 0.7 \rightarrow y = 1$
 $X > 0.7 \rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$
 $x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$

$$x > 0.75 \rightarrow y = 1$$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$

$$x > 0.75 \rightarrow y = 1$$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$

$$x > 0.75 \rightarrow y = 1$$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$

$$x > 0.75 \rightarrow y = 1$$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$$x \leq 0.05 \rightarrow y = 1$$

$$x > 0.05 \rightarrow y = 1$$

Bagging Example

Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1
Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Introduction to Data Mining, 2nd Edition

Bagging – Effect on Variance

- Improves generalization error by reducing the variance of the base classifiers.
- If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data.
- Since every sample has an equal probability of being selected, bagging does not focus on any particular instance of the training data.
 - It is therefore less susceptible to model overfitting when applied to noisy data.

Bagging – Effect on Bias

- If a base classifier is stable, i.e., robust to minor perturbations in the training set, then the error of the ensemble is primarily caused by bias in the base classifier.
- In this situation, bagging may not be able to improve the performance of the base classifiers significantly.
- It may even degrade the classifier's performance because the effective size of each training set is about 37% smaller than the original data.

Random Forest

- Ensemble method specifically designed for decision tree classifiers
- Random Forests grows many trees
 - Ensemble of unpruned decision trees
 - Each base classifier classifies a “new” vector of attributes from the original data
 - Final result on classifying a new instance: voting.
 - Forest chooses the classification result having the most votes (over all the trees in the forest)

Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating training set as well as features
 - Use bootstrap sample to train every decision tree (similar to Bagging)
 - Use the following tree induction algorithm:
 - At every internal node of decision tree, randomly sample p attributes for selecting split criterion
 - Repeat this procedure until all leaves are pure (unpruned tree)

Feature Randomness

- Decision tree : to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs. those in the right node.
- Random forest : each tree can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation

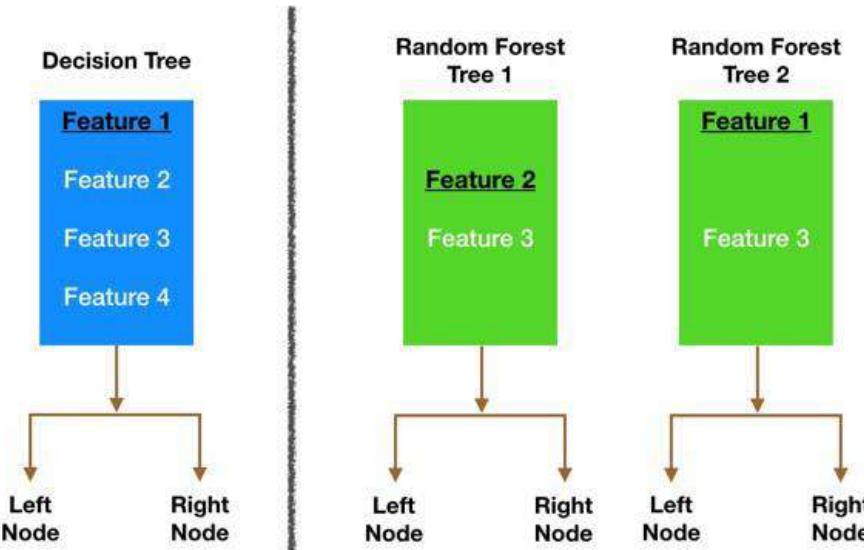
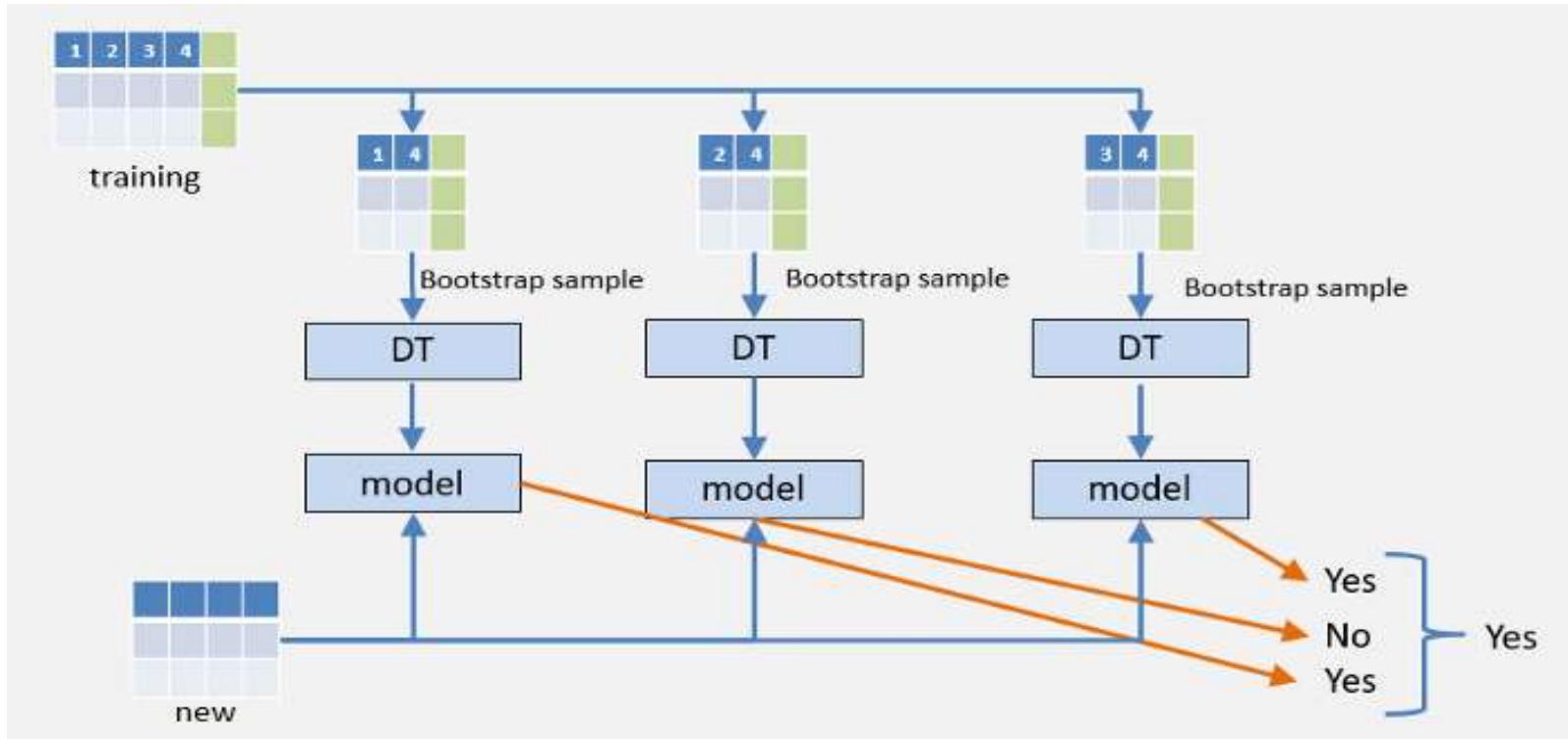


Image credit: <https://medium.com>

Random Forest



- Trees that are trained on different sets of data (bagging)
- but also use different features to make decisions.

Image credit: <https://medium.com>

Random Forest

- Random Forest need features that have at least some predictive power.
- The trees of the forest and more importantly their predictions need to be uncorrelated (or at least have low correlations with each other).
- Algorithm can solve both type of problems i.e. classification and regression
- Power to handle large data set with higher dimensionality.
- It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.



Boosting

Boosting

- What if a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.
- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights (for being selected for training)
 - Unlike bagging, weights may change at the end of each boosting round

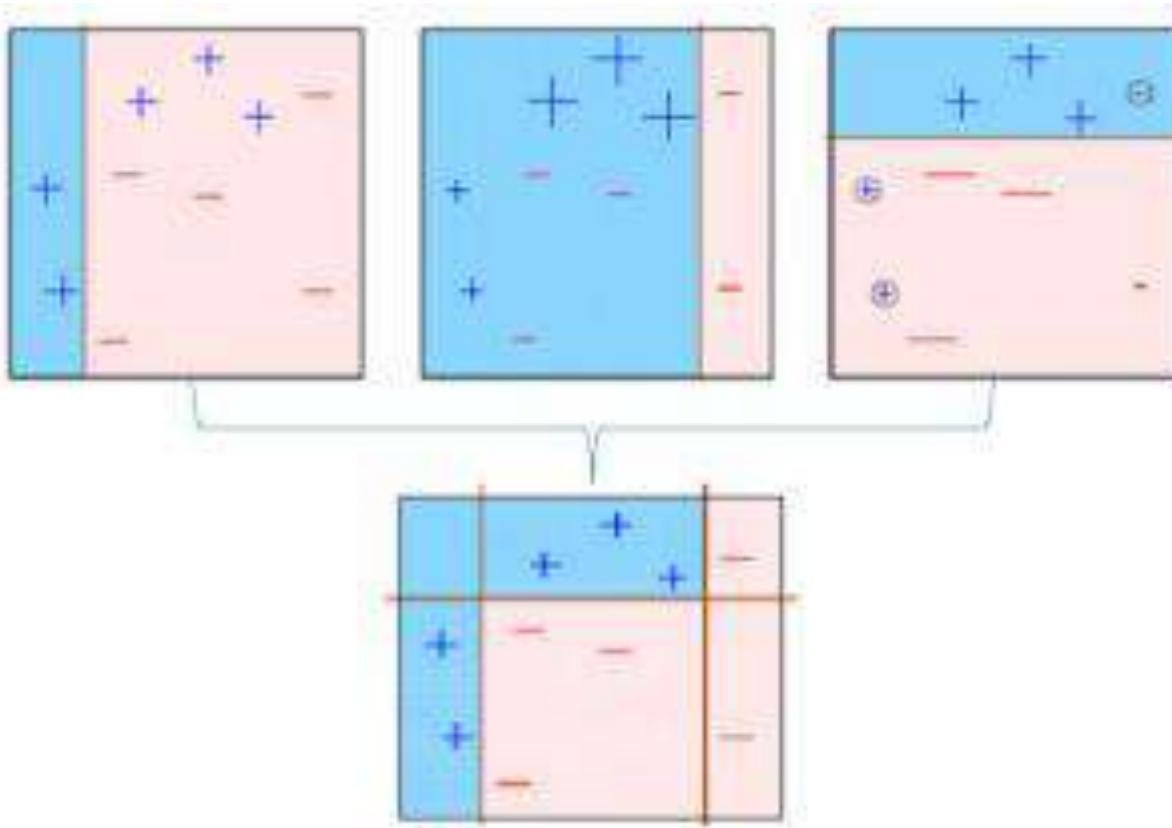
Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

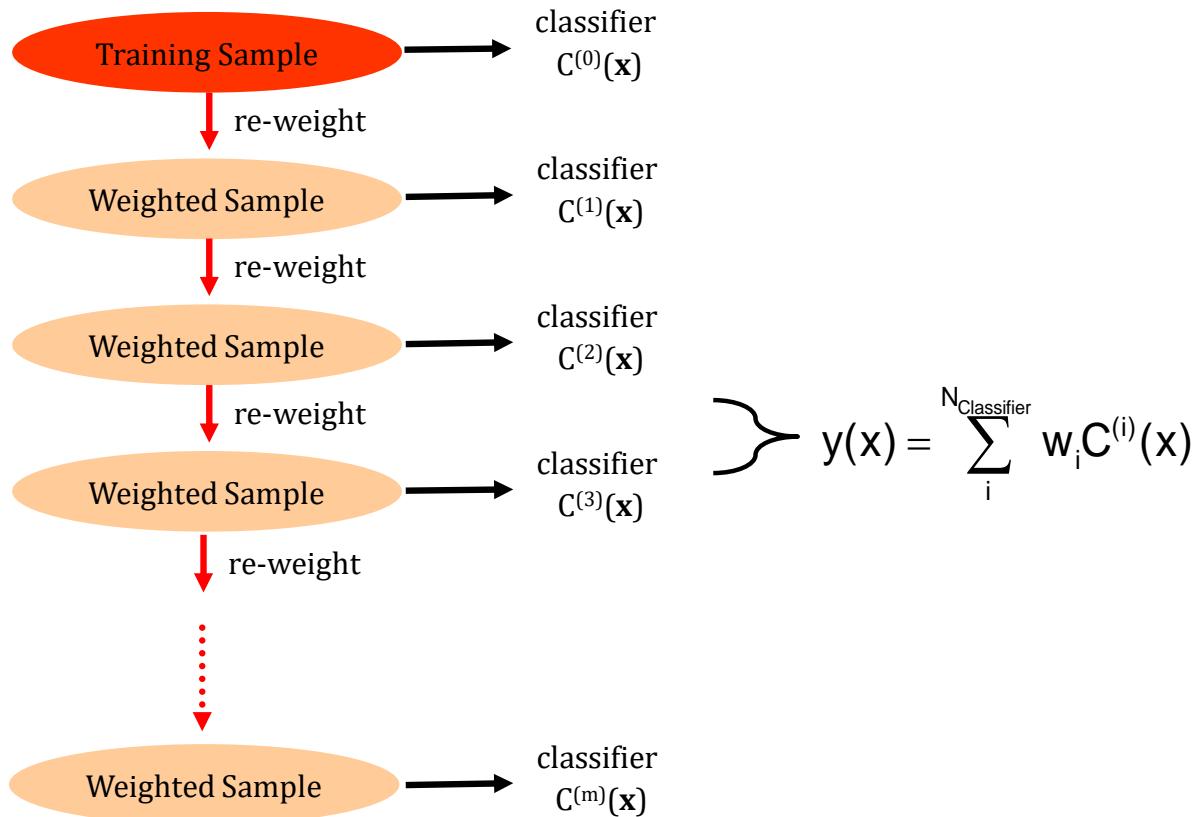
- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Boosting



- Multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).
- Individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Boosting



Ada Boost - Adaptive boosting Algorithm

- Initially, all observations (n) in the dataset are given equal weights ($1/n$).
 - A model is built on a subset of data.
 - Using this model, predictions are made on the whole dataset.
 - Errors are calculated by comparing the predictions and actual values.
 - Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
 - This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.
 - Usually, decision trees are used for modelling.
-

Ada Boost - Adaptive boosting Algorithm

- Base classifiers C_i : C_1, C_2, \dots, C_T
- Error rate: N input samples, (Averaging over weights of training examples of misclassified points)

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$
- Other two variants for calculating error rate
 - Taking sum of weights of training examples of misclassified points
 - Taking weighted average over weights of training examples of misclassified points
$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$
- **Notice that the error is measured with respect to the same distribution D_t on which the base classifier was trained.**
- Importance of a classifier:

Source Credit : https://en.wikipedia.org/wiki/AdaBoost#Choosing_at

Ada Boost – Weight Update

Weight Update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

<- Eqn:5.88

- Reduce weight if correctly classified else increase
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated

• Prediction:

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

Ada Boost Algorithm – Version 1

(with bootstrapping + average of error term)

Algorithm 5.7 AdaBoost Algorithm

- 1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$. {Initialize the weights for all n instances.}
 - 2: Let k be the number of boosting rounds.
 - 3: **for** $i = 1$ to k **do**
 - 4: Create training set D_i by sampling (with replacement) from D according to \mathbf{w} .
 - 5: Train a base classifier C_i on D_i .
 - 6: Apply C_i to all instances in the original training set, D .
 - 7: $\epsilon_i = \frac{1}{n} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$ {Calculate the weighted error}
 - 8: **if** $\epsilon_i > 0.5$ **then**
 - 9: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$. {Reset the weights for all n instances.}
 - 10: Go back to Step 4.
 - 11: **end if**
 - 12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
 - 13: Update the weight of each instance according to equation (5.88).
 - 14: **end for**
 - 15: $C^*(\mathbf{x}) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(\mathbf{x}) = y))$.
-

Ada Boost Algorithm – Version 2



(with bootstrapping + sum for error term)

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

Member classifier with less error are given more weight in final ensemble hypothesis.
Final prediction is a weighted combination of each members prediction

AdaBoost Example (version 1)

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Training sets for the first 3 boosting rounds:

AdaBoost Example (version 1)

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Boosting Round 1:



$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_i} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

Boosting Round 2:



x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Round - 2

norm w2	x	y	w3	norm w3
0.311	0.1		1	0.019
0.311	0.2		1	0.019
0.311	0.3		1	0.019
0.010	0.4		-1	0.155
0.010	0.5		-1	0.155
0.010	0.6		-1	0.155
0.010	0.7		-1	0.155
0.010	0.8		1	0.001
0.010	0.9		1	0.001
0.010	1		1	0.001
0.004	=ε2			
2.778	=α2			
		sum(w1-w10)	0.679	

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1



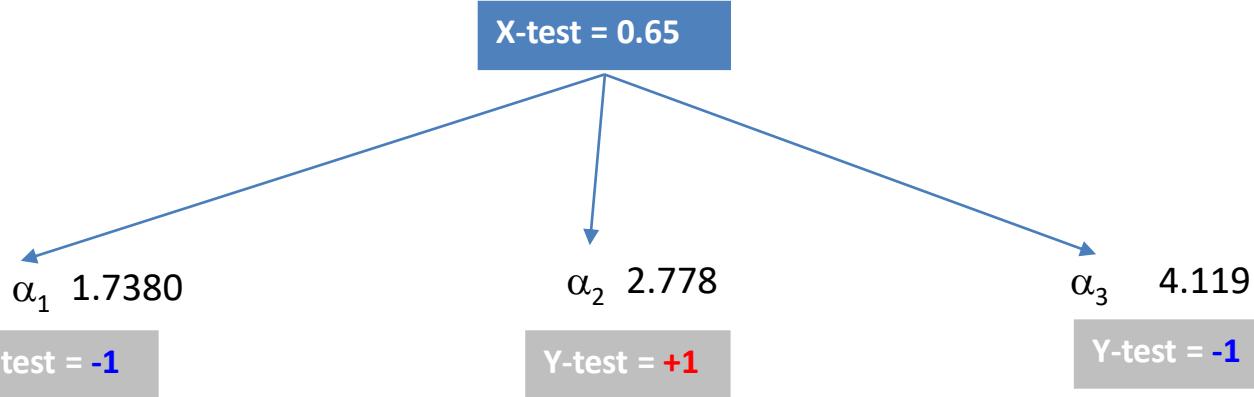
$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

Round 3					
norm w3	x	y	w4	norm w4	
0.028	0.1		1	0.000463	0.002588738
0.028	0.2		1	0.000463	0.002588738
0.028	0.3		1	0.000463	0.002588738
0.228	0.4		-1	0.003706	0.020736589
0.228	0.5		-1	0.003706	0.020736589
0.228	0.6		-1	0.003706	0.020736589
0.228	0.7		-1	0.003706	0.020736589
0.001	0.8		1	0.054162	0.30309581
0.001	0.9		1	0.054162	0.30309581
0.001	1		1	0.054162	0.30309581
0.000264	=ε3				
4.119	=α3				
		sum(w1-w10)	0.178695		



$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

$$C^*(x_test) = \arg \max_y \sum_{j=1}^3 \alpha_j \delta(C_j(x_test) = y)$$

$$y = +1 \longrightarrow \sum_{j=1}^3 \alpha_j \delta(C_j(x_test) = y) = 2.778$$

$$y = -1 \longrightarrow \sum_{j=1}^3 \alpha_j \delta(C_j(x_test) = y) = 1.738 + 4.119 = \sim 5.857$$

Answer: $C^*(x_test) = -1$

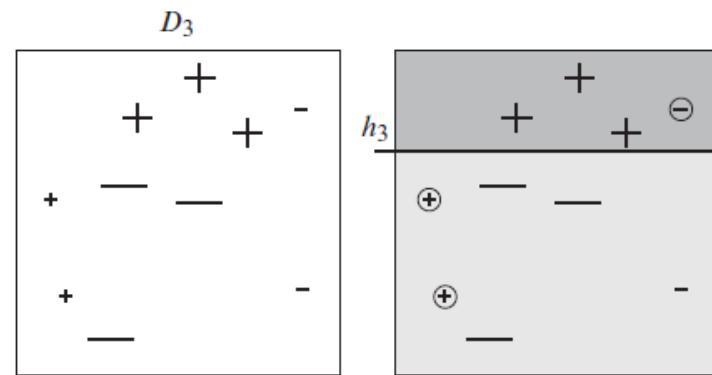
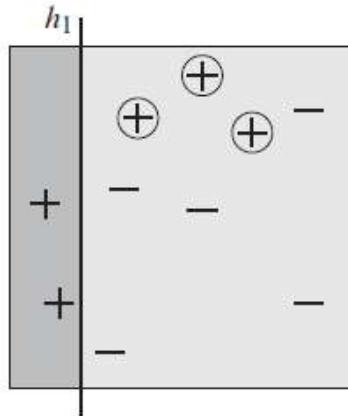
OR

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

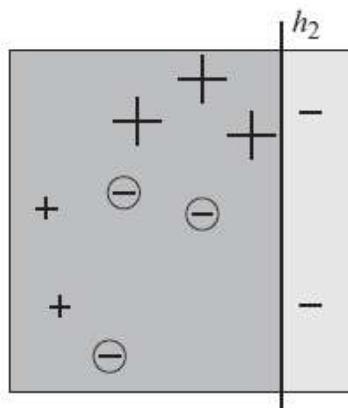
AdaBoost error function takes into account the fact that only the sign of the final result is used, thus sum can be far larger than 1 without increasing error

AdaBoost Example (version 2)

D_1
+ ¹ + ² + ³ - ⁴ + ⁵ - ⁶ - ⁷ + ⁸ - ⁹ - ¹⁰



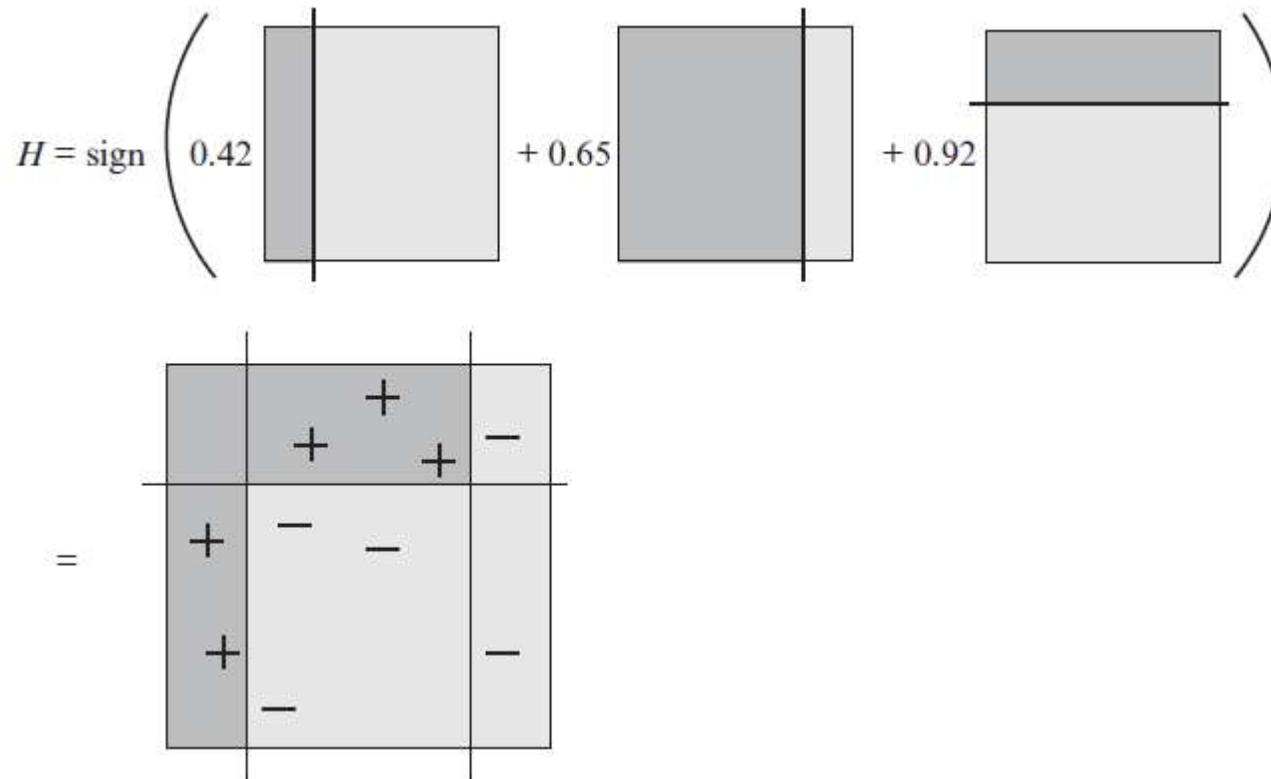
D_2
+ ¹ + ² + ³ - ⁴ + ⁵ - ⁶ - ⁷ + ⁸ - ⁹ - ¹⁰



Source Credit : Schapire and Freund, 2012

AdaBoost Example (version 2)

How do we combine the results now?



Source Credit : Schapire and Freund, 2012

AdaBoost Example (version 2)

Table 1.1

The numerical calculations corresponding to the toy example in figure 1.1

	1	2	3	4	5	6	7	8	9	10	
$D_1(i)$	<u>0.10</u>	<u>0.10</u>	<u>0.10</u>	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$\epsilon_1 = 0.30, \alpha_1 \approx 0.42$
$e^{-\alpha_1 y_i h_1(x_i)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
$D_1(i) e^{-\alpha_1 y_i h_1(x_i)}$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
$D_2(i)$	0.17	0.17	0.17	0.07	0.07	<u>0.07</u>	<u>0.07</u>	0.07	<u>0.07</u>	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i) e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
$D_3(i)$	0.11	0.11	0.11	<u>0.05</u>	<u>0.05</u>	0.17	0.17	<u>0.05</u>	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i) e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

Calculations are shown for the ten examples as numbered in the figure. Examples on which hypothesis h_t makes a mistake are indicated by underlined figures in the rows marked D_t .

Source Credit : Schapire and Freund, 2012

AdaBoost base learners

- AdaBoost works best with “weak” learners
 - Should not be complex
 - Typically high bias classifiers
 - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
 - Can prove training error goes to 0 in $O(\log n)$ iterations
- Examples:
 - Decision stumps (1 level decision trees)
 - Depth-limited decision trees
 - Linear classifiers

AdaBoost in practice

Strengths:

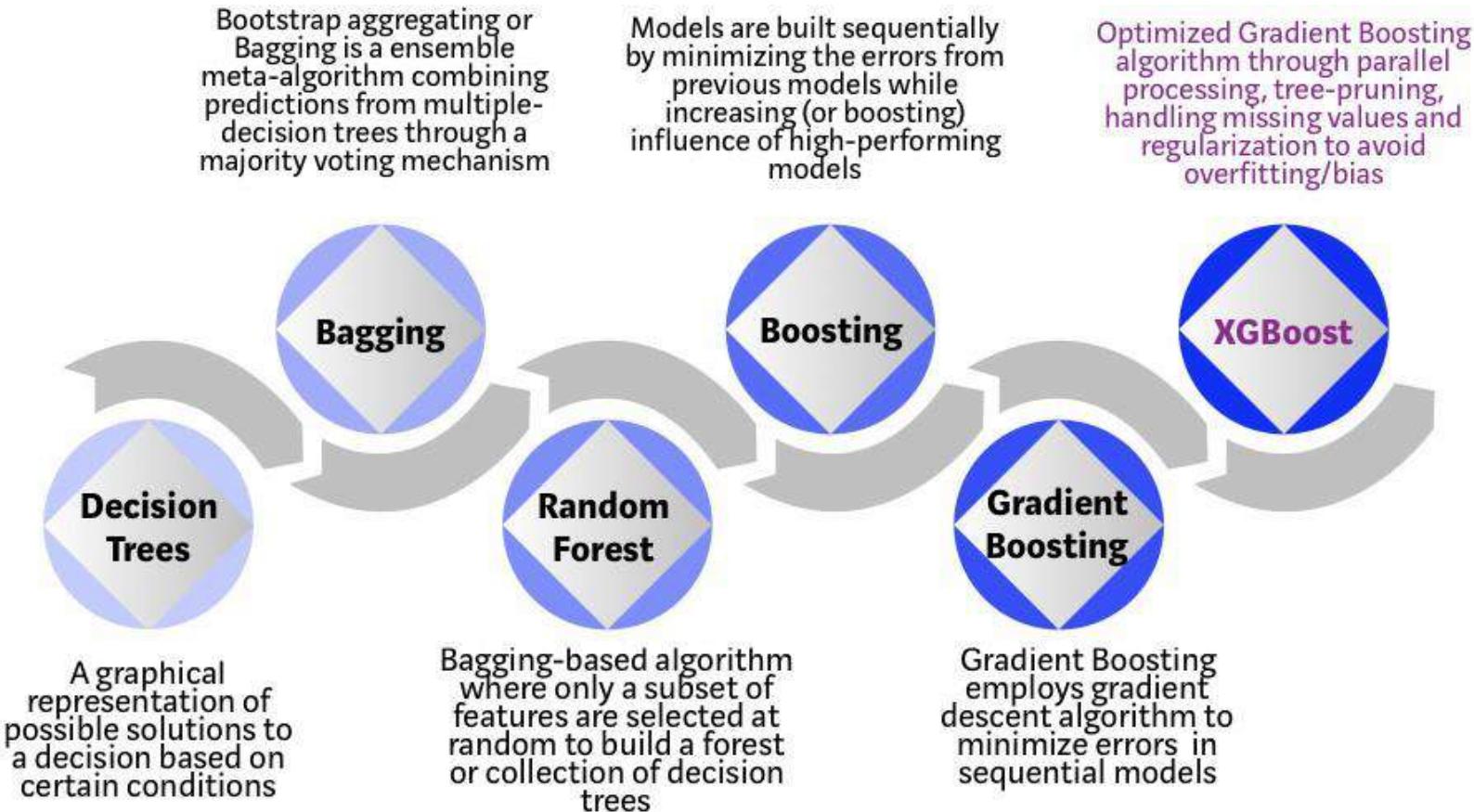
- Fast and simple to program
- No parameters to tune (besides T)
- No assumptions on weak learner

When boosting can fail:

- Given insufficient data
- Overly complex weak hypotheses
- Can be susceptible to noise
- When there are a large number of outliers
- The model cannot be parallelized since each predictor can only be trained after the previous one has been trained and evaluated.

Fine Tuning Ensembles

- Model combination does not always guaranteed to decrease error, unless
 - base-learners are diverse and accurate
- Ignore poor base learners
 - Use accuracy as a cut-off
 - Introduce some pruning with which at each iteration remove poor learners / learners whose absence lead to improvement (if any)
 - Modify iterations to allow both additions / deletions of learners
 - Discarding appropriately leads to better performance



Source Credit : <https://towardsdatascience.com/>

Gradient Boosting

- The idea of gradient boosting originated in the observation by [Leo Breiman](#) that boosting can be interpreted as an optimization algorithm on a suitable cost function
- optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction.
- predictor can be any machine learning algorithm like SVM, Logistic regression, KNN , Decision tree etc. But Decision tree version of gradient boosting is much popular
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, “shortcomings” are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

Gradient Boosting

- You are given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss
- There are some mistakes:

$F(x_1) = 0.8$, while $y_1 = 0.9$,

$F(x_2) = 1.4$ while $y_2 = 1.3\dots$

How can you improve this model?

- Rules:
 - You are not allowed to remove anything from F or change any parameter in F .
 - You can add an additional model (regression tree) h to F , so the new prediction will be $F(x) + h(x)$.

Gradient Boosting

You wish to improve the model such that

- $F(x_1) + h(x_1) = y_1$
- $F(x_2) + h(x_2) = y_2 \dots$
- $F(x_n) + h(x_n) = y_n$

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$

$$h(x_2) = y_2 - F(x_2) \dots$$

$$h(x_n) = y_n - F(x_n)$$

Fit a regression tree h to data

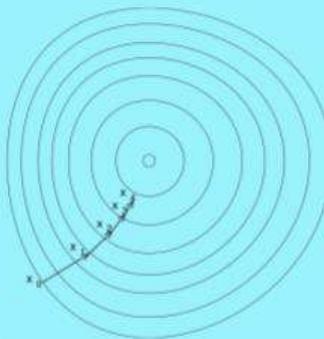
$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$$

- Simple solution: $y_i - F(x_i)$ are called residuals
- These are the parts that existing model F cannot do well.
- The role of h is to compensate the shortcoming of existing model F
- If the new model $F + h$ is still not satisfactory, we can add another regression tree...

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$



Gradient Boosting for regression

Loss function $L(y, F(x)) = (y - F(x))^2/2$

We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), \dots, F(x_n)$.

Notice that $F(x_1), F(x_2), \dots, F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Gradient Boosting for regression

$$F(x_i) := F(x_i) + h(x_i)$$

$$F(x_i) := F(x_i) + y_i - F(x_i)$$

$$F(x_i) := F(x_i) - 1 \frac{\partial J}{\partial F(x_i)}$$

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

For regression with **square loss**,

residual \Leftrightarrow negative gradient

fit h to residual \Leftrightarrow fit h to negative gradient

update F based on residual \Leftrightarrow update F based on negative gradient

So we are actually updating our model using **gradient descent!**

Gradient boosting algorithm

let F_0 be a “dummy” constant model

for $m = 1, \dots, M$

for each pair (\mathbf{x}_i, y_i) in the training set

 Compute the pseudo-residual $R(y_i, F_{m-1}(\mathbf{x}_i)) = \text{negative}$
 gradient of the loss

 Train a regression sub-model h_m on the pseudo-residuals

 Add h_m to the ensemble: $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho \cdot h_m(\mathbf{x})$

return the ensemble F_M

Gradient boosting: Example

Height	Age	Gender	Weight
5.4	28	Male	88
5.2	26	Female	76
5	28	Female	56
5.6	25	Male	73
6	25	Male	77
4	22	Female	57

F_0 be a “dummy” constant model

Average value is predicted.

Height	Age	Gender	Actual Weight	Predicted weight 1
5.4	28	M	88	71.2
5.2	26	F	76	71.2
5	28	F	56	71.2
5.6	25	M	73	71.2
6	25	M	77	71.2
4	22	F	57	71.2

Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

Iteration 1

compute the pseudo-residual $R(y_i, F_{m-1}(x_i)) =$
negative gradient of the loss

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1
5.4	28	Male	88	71.2	88-71.2 = 16.8
5.2	26	Female	76	71.2	76-71.2 = 4.8
5	28	Female	56	71.2	56-71.2 = -15.2
5.6	25	Male	73	71.2	73-71.2 = 1.8
6	25	Male	77	71.2	77-71.2 = 5.8
4	22	Female	57	71.2	57-71.2 = -14.2

Residual: $h_1(x) = y - F_0(x)$

A tree with maximum leaf nodes as 4 using Height, Age and Gender to predict the residuals(Error)



Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

Iteration 1

Combining the trees to make the new prediction: $F_m(x) = F_{m-1}(x) + \rho \cdot h_m(x)$

Learning rate : 0.1

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1	Predicted weight 2
5.4	28	Male	88	71.2	$88 - 71.2 = 16.8$	$71.2 + 0.1 * 16.8 = 72.9$
5.2	26	Female	76	71.2	$76 - 71.2 = 4.8$	$71.2 + 0.1 * (-5.2) = 70.7$
5	28	Female	56	71.2	$56 - 71.2 = -15.2$	$71.2 + 0.1 * (-5.2) = 70.7$
5.6	25	Male	73	71.2	$73 - 71.2 = 1.8$	$71.2 + 0.1 * 3.8 = 71.6$
6	25	Male	77	71.2	$77 - 71.2 = 5.8$	$71.2 + 0.1 * 3.8 = 71.6$
4	22	Female	57	71.2	$57 - 71.2 = -14.2$	$71.2 + 0.1 * (-14.2) = 69.8$

Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

Iteration 1

$$\text{Residual: } h_2(x) = y - F_1(x)$$

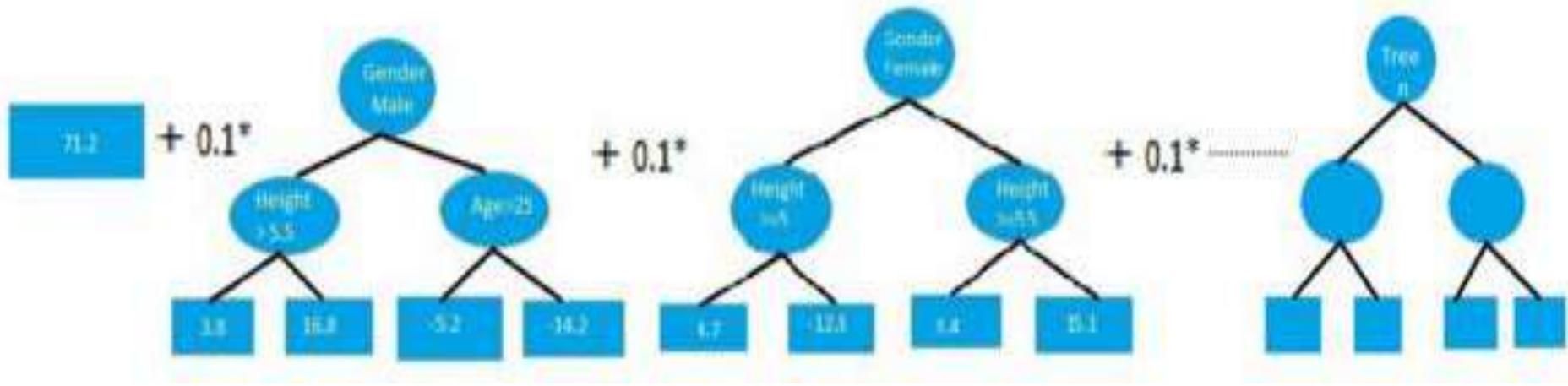
Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1	Predicted weight 2	Pseudo Residuals2
5.4	28	Male	88	71.2	88-71.2=16.8	71.2+0.1*16.8=72.9	88-72.9=15.1
5.2	26	Female	76	71.2	76-71.2=4.8	71.2+0.1*(-5.2)=70.7	76-70.7=5.3
5	28	Female	56	71.2	56-71.2=-15.2	71.2+0.1*(-5.2)=70.7	56-70.7=-14.7
5.6	25	Male	73	71.2	73-71.2=1.8	71.2+0.1*3.8=71.6	73-71.6=1.4
6	25	Male	77	71.2	77-71.2=5.8	71.2+0.1*3.8=71.6	77-71.6=5.4
4	22	Female	57	71.2	57-71.2=-14.2	71.2+0.1*(-14.2)=69.8	57-69.8=-12.8



Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

Iteration 2

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1	Predicted weight 2	Pseudo Residuals 2	Predicted Weight 3
5.4	28	Male	88	71.2	88-71.2 = 16.8	71.2 + 0.1 * 16.8 = 72.9	88 - 72.9 = 15.1	71.2 + 0.1 * 16.8 + 0.1 * 15.1 = 74.4
5.2	26	Female	76	71.2	76 - 71.2 = 4.8	71.2 + 0.1 * (-5.2) = 70.7	76 - 70.7 = 5.3	71.2 + 0.1 * (-5.2) + 0.1 * (-4.7) = 70.2
5	28	Female	56	71.2	56 - 71.2 = -15.2	71.2 + 0.1 * (-5.2) = 70.7	56 - 70.7 = -14.7	71.2 + 0.1 * (-5.2) + 0.1 * (-4.7) = 70.2
5.6	25	Male	73	71.2	73 - 71.2 = 1.8	71.2 + 0.1 * 3.8 = 71.6	73 - 71.6 = 1.4	71.2 + 0.1 * 3.8 + 0.1 * 3.4 = 71.9
6	25	Male	77	71.2	77 - 71.2 = 5.8	71.2 + 0.1 * 3.8 = 71.6	77 - 71.6 = 5.4	71.2 + 0.1 * 3.8 + 0.1 * 3.4 = 71.9
4	22	Female	57	71.2	57 - 71.2 = -14.2	71.2 + 0.1 * (-14.2) = 69.8	57 - 69.8 = -12.8	71.2 + 0.1 * (-14.2) + 0.1 * (-12.8) = 68.5



Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

Gradient boosting: Summary

- Gradient boosting involves three elements:
 - A loss function to be optimized
 - For example, regression may use a squared error and classification may use logarithmic loss.
 - A weak learner to make predictions E.g Decision tree/Decision stump
 - An additive model to add weak learners to minimize the loss function.
 - Trees are added one at a time, and existing trees in the model are not changed.
 - A gradient descent procedure is used to minimize the loss when adding trees.
 - Instead of parameters, we have weak learners
 - After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient).

References

- Introduction to Data Mining, by Pang-Ning Tan, Michael Steinbach , Vipin Kumar
- Bishop - Pattern Recognition And Machine Learning - Springer 2006
- https://en.wikipedia.org/wiki/Gradient_boosting#:~:text=Gradient%20boosting%20is%20a%20machine,which%20are%20typically%20decision%20trees
- <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>



Machine Learning

DSE CLZG565

Support Vector Machine

Raja vadhana P
Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
 - I here by acknowledge all the contributors for their material and inputs.
 - I have provided source information wherever necessary
 - I have added and modified the content to suit the requirements of the course
- Source:** Slides of Prof. Chetana, Prof.Vimal, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes, Tom Mitchell, Andrew Ng and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

Module – 11 & 12 – Support Vector Machine

- Introduction
- VC Dimension
- Linearly Separable Data
- Non-Linearly Separable Data – Kernel Trick – [Next Class](#)

Learning Objectives

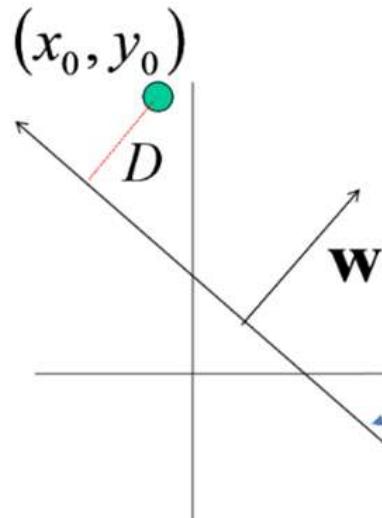
- Maximum Margin Classification
- SVM optimization problem
- Optimization using Langrangian
- Solving SVM using KKT conditions

Refresher from MFDS

(Self Study)

- Distance between a point and plane
- Vector Distance between two points
- Optimization Problem – Significance Primal Dual
- Non-Linear Optimization Problem

Math Basics



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

↔

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^\top \mathbf{x} + b|}{\|\mathbf{w}\|}$$

distance from
point to line

Optimization Problem

- Optimization problem is typically written:

Minimize $f(x)$

subject to

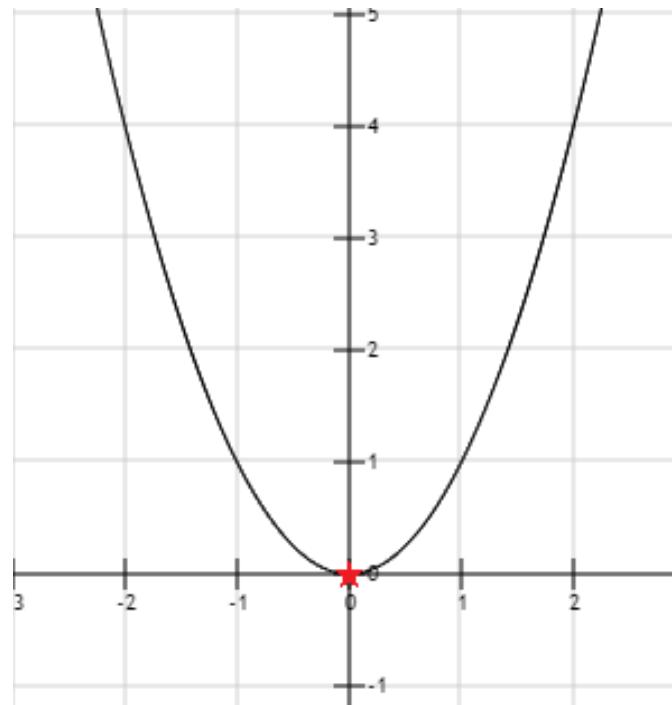
$$g_i(x) = 0, \quad i=1, \dots, p$$

$$h_i(x) \leq 0, \quad i=1, \dots, m$$

- $f(x)$ is called the objective function
- By changing x (the optimization variable) we wish to find a value x^* for which $f(x)$ is at its minimum.
- p functions of g_i define equality constraints and
- m functions h_i define inequality constraints.
- The value we find MUST respect these constraints!

Unconstrained Optimization

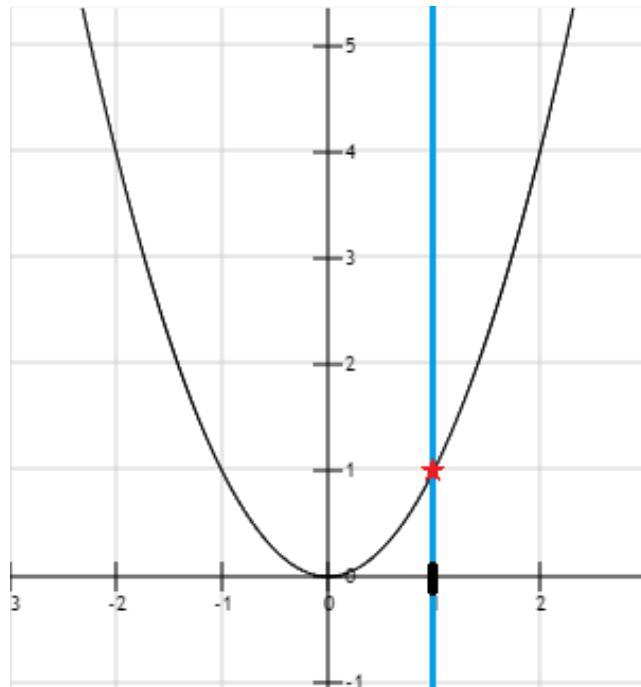
Minimize x^2



Constrained Optimization – Equality Constraint

Minimize x^2

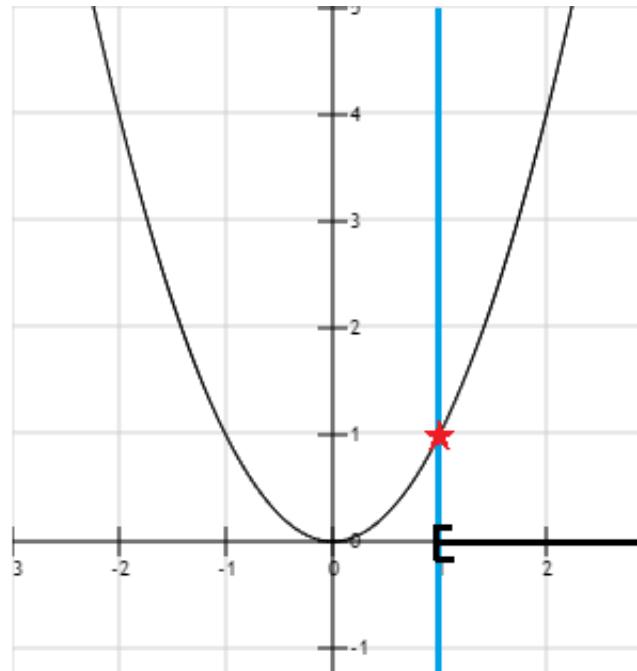
Subject to $x = 1$



Constrained Optimization – Inequality Constraint

Minimize x^2

Subject to $x \geq 1$



Constrained Optimization

- We can also have mix equality and inequality constraints together.
- Only restriction is that if we use contradictory constraints, we can end up with a problem which does not have a feasible set

Minimize x^2

Subject to

$$x = 1$$

$$x < 0$$

Impossible for x to be equal 1 and less than zero at the same

Lagrange Multiplier

- **How do we find the solution to an optimization problem with constraints?**
- Constrained maximization (minimization) problem is rewritten as a Lagrange function.
- Lagrange function is a strategy for finding the local maxima and minima of a function subject to constraints
- It uses Lagrange multipliers

Lagrange Multiplier - Example

Maximize

$$f(x,y) = x^2 y$$

Subject to

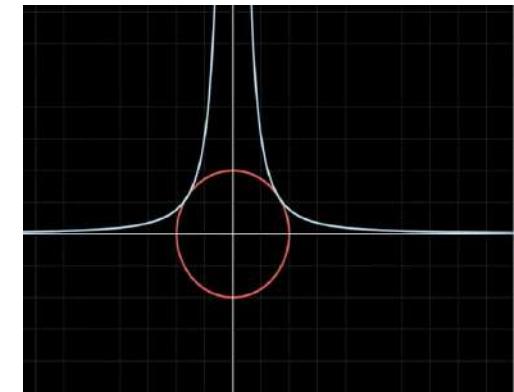
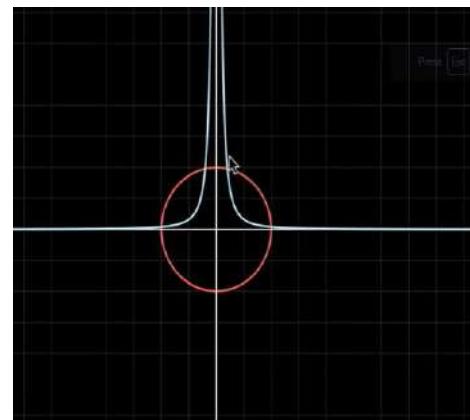
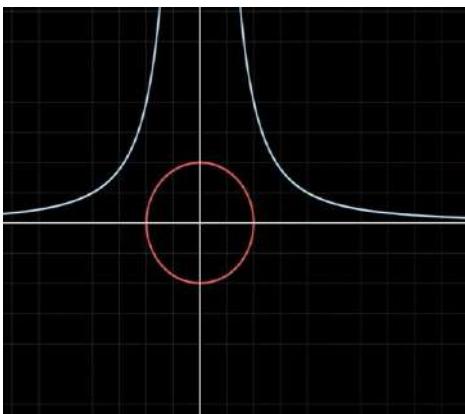
$$g(x, y) : x^2 + y^2$$

$$f(x,y) = K = 1$$

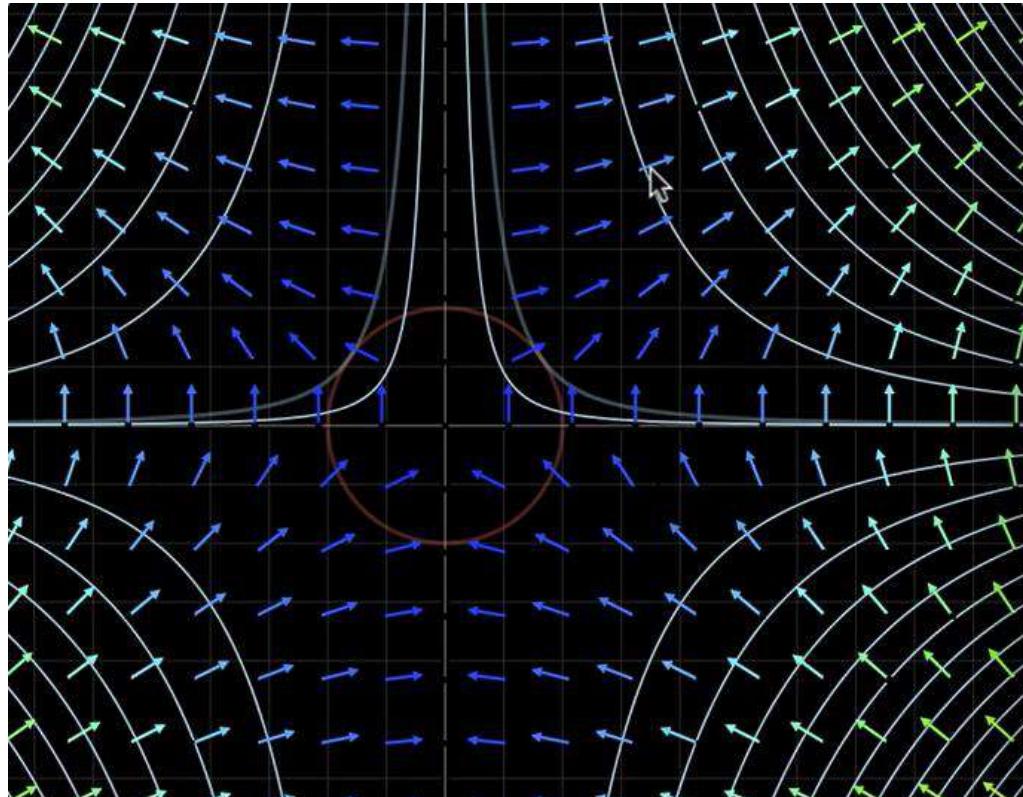
Larger constant K
constant K

Smaller

Maximize the
function for
some value of K



Lagrange Multiplier - Example



- Gradients are perpendicular to various contour lines(w.r.t to different values of function) of function f
- Function is not changing the value along the points on any the contour line

Constrained to Unconstrained Optimization

- Lagrange Multiplier



- Maximum of $f(x,y)$ under constraint $g(x, y)$ is obtained when their gradients point to same direction

(when they are tangent to each other).

- Introduce a Lagrange multiplier λ for the equality constraint
- Mathematically,

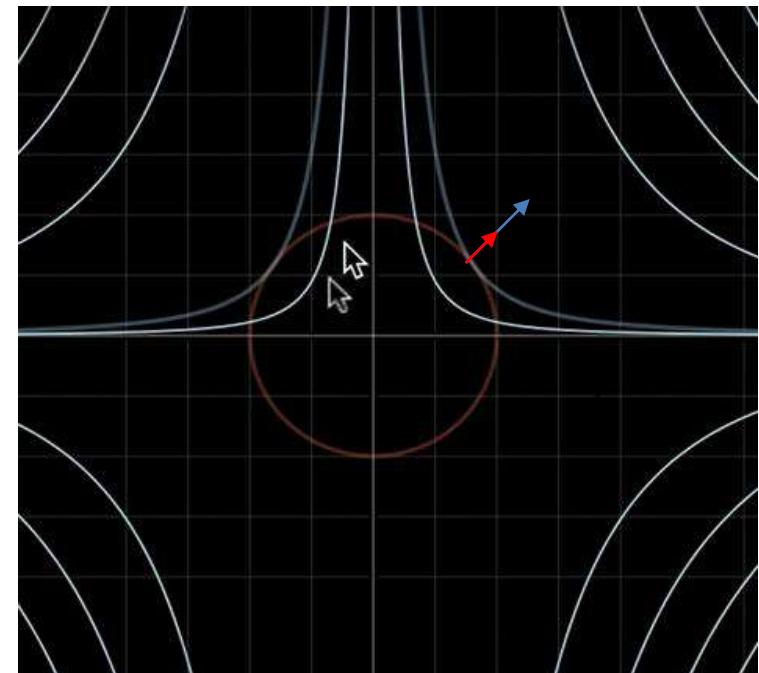
$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

Equivalently:

$$L(x,y, \lambda) = f(x,y) - \lambda \cdot g(x,y) - C$$

Optimize L using gradients

$$\nabla L(x,y, \lambda) = 0$$



Summary - Lagrange multiplier method

1. Construct the Lagrangian function by introducing one multiplier per constraint
2. Get the gradient ∇L of the of the Lagrangian
3. Solve for $\nabla L(x,y, \lambda)=0$

Example

$\max xy$ subject to $x + y = 6$

- Introduce a Lagrange multiplier λ for constraint
- Construct the Lagrangian

$$L(x, y) = xy + \lambda(x + y - 6)$$

- Stationary points

$$\frac{\partial L(x, y)}{\partial \lambda} = x + y - 6 = 0$$

$$\begin{cases} \frac{\partial L(x, y)}{\partial x} = y - \lambda = 0 \\ \frac{\partial L(x, y)}{\partial y} = x - \lambda = 0 \end{cases} \Rightarrow x = y = \lambda$$

$$\Rightarrow x = y = 3$$

x and y values remain same even if you take $+\lambda$ or $-\lambda$ for equality constraint

$$\begin{aligned} 2x &= 6 \\ x &= y = 3 \\ \lambda &= 3 \end{aligned}$$

Karush–Kuhn–Tucker (KKT) theorem

- KKT approach to nonlinear programming (quadratic) generalizes the method of [Lagrange multipliers](#), which allows only equality constraints.
- KKT allows inequality constraints

Karush–Kuhn–Tucker (KKT) theorem

- Start with

$\max f(x)$ subject to

$$g_i(x) = 0 \text{ and } h_j(x) \geq 0 \text{ for all } i, j$$

- f is the objective function, g_i ($i=1 \dots m$) are the equality constraint functions and h_j ($j=1 \dots l$) are the inequality constraint functions
- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Take gradient and set to 0 – but satisfy other conditions also.

KKT conditions

- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Necessary conditions to have a minimum are

Stationarity

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$$

primal feasibility condition

$$g_i(x^*) = 0 \text{ for all } i$$

primal feasibility condition

$$h_j(x^*) \geq 0 \text{ for all } j$$

dual feasibility condition

$$\mu_j \geq 0 \text{ for all } j$$

complementary slackness condition

$$\mu_j^* h_j(x^*) = 0 \text{ for all } j$$

Notion of SVM Classifier

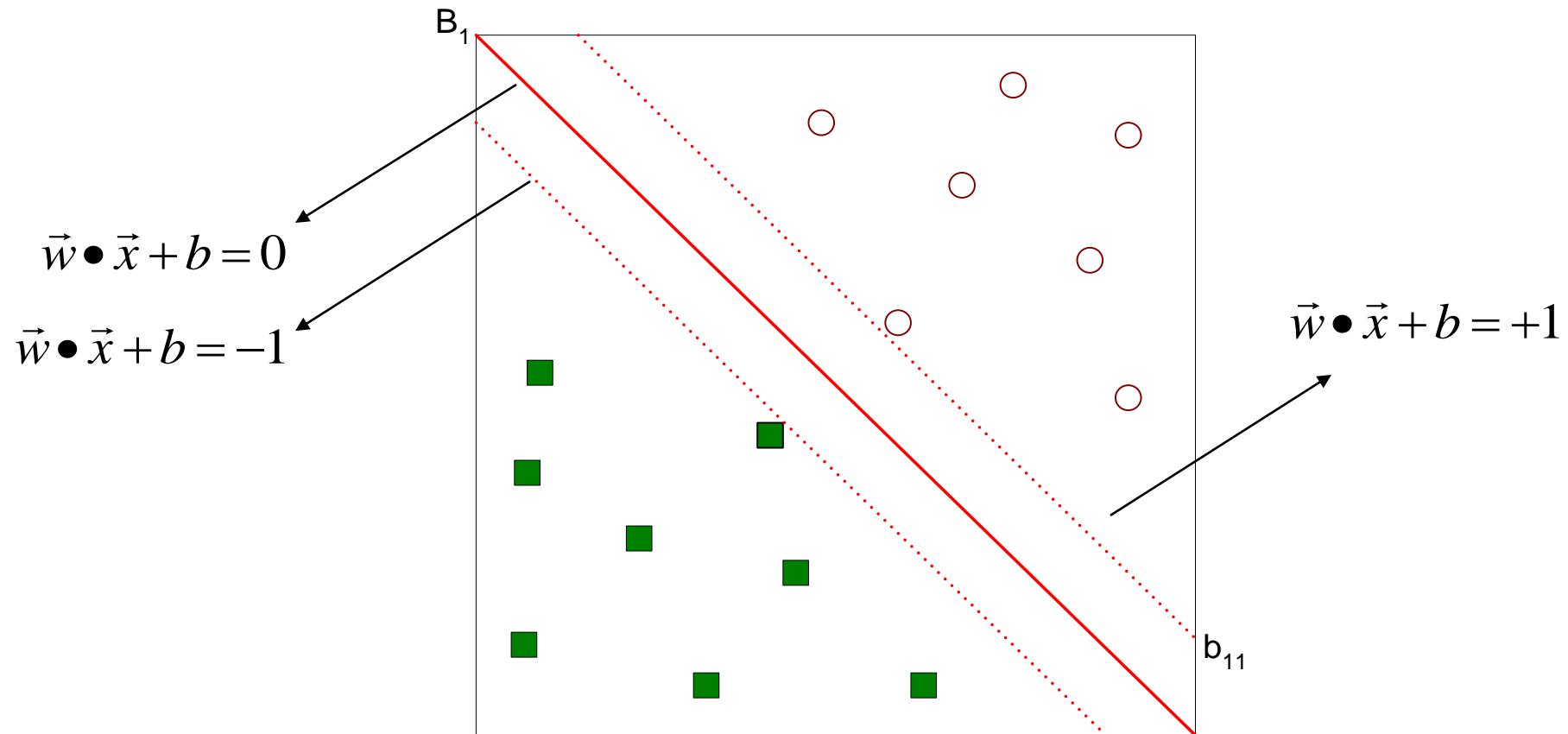
Support Vector Machines - SVM

- SVM technique has its roots in statistical learning theory (Vladimir Vapnik, 1992).
- As a task of classification, it searches for optimal hyperplane(i.e., decision boundary) separating the tuples of one class from another.
- SVM works well with higher dimensional data and thus avoids dimensionality problem.

Key Concepts

- Maximum margin hyperplane : a key concept in SVM.
- Linear SVM : a classification technique when training data are linearly separable.
 - **Optimization using Langrangian**
- Non-linear SVM : a classification technique when training data are linearly non-separable.
 - **Kernel Trick**
 - Linear SVM with soft margin
 - Non-linear SVM

Support Vector Machine



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

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

Support Vector Machine

$$\mathbf{w} \cdot \mathbf{x}^+ + b = +1$$

$$\mathbf{w} \cdot \mathbf{x}^- + b = -1$$

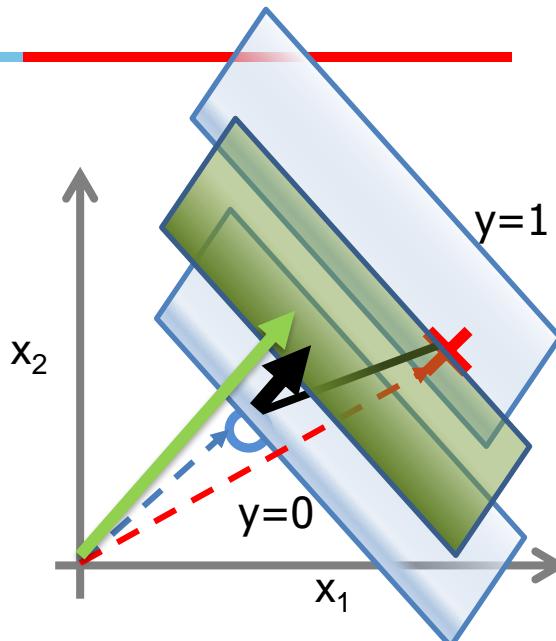
Margin width

$$= \mathbf{x}^+ - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$= \frac{\mathbf{w} \cdot \mathbf{x}^+ - \mathbf{w} \cdot \mathbf{x}^-}{\|\mathbf{w}\|}$$

$$= (1-b) - (-1-b) / \|\mathbf{w}\|$$

$$= \frac{2}{\|\mathbf{w}\|}$$



Support Vector Machine

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

The solution involves constructing a unconstrained problem where a Lagrange multiplier is associated with every constraint in the primary problem.

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \text{ is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Support Vector Machine

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$
 $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

- Classification function:

$$\begin{aligned}f(x) &= \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \\&= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)\end{aligned}$$

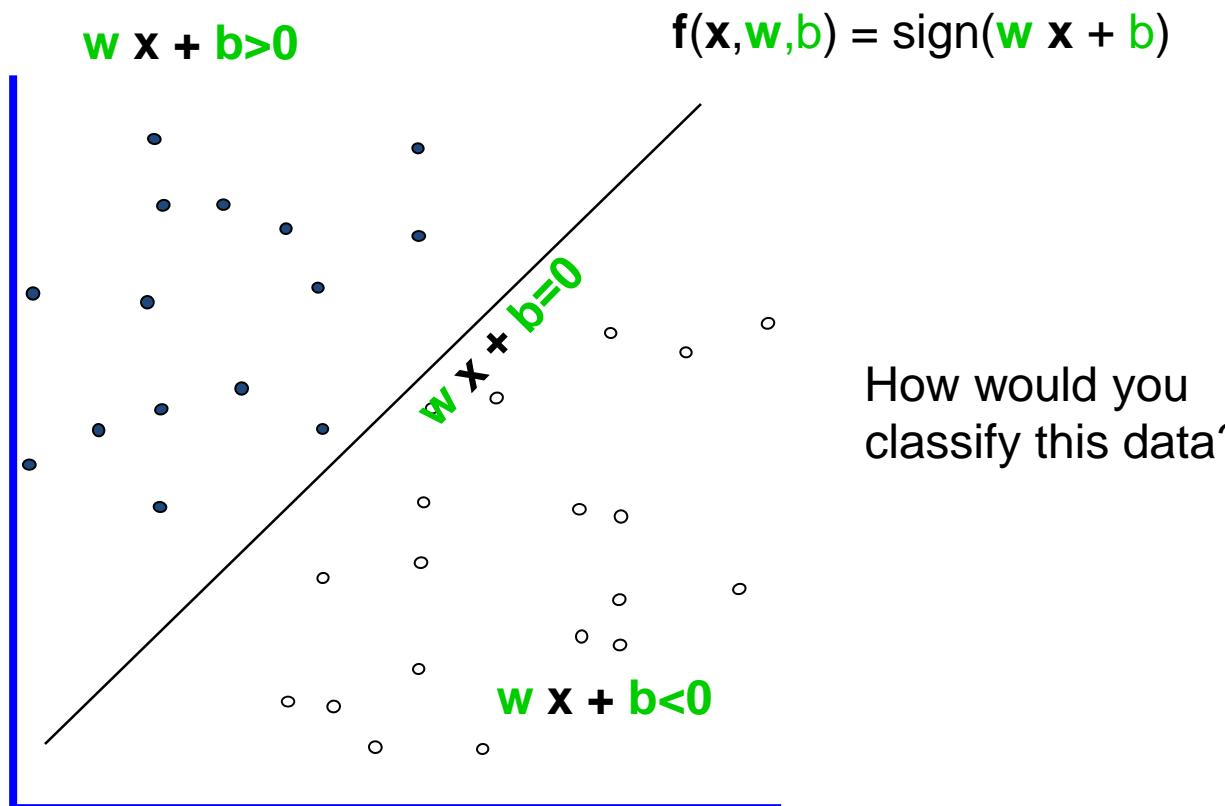
If $f(x) < 0$, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i
- (Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points)

Maximum Margin Hyperplane

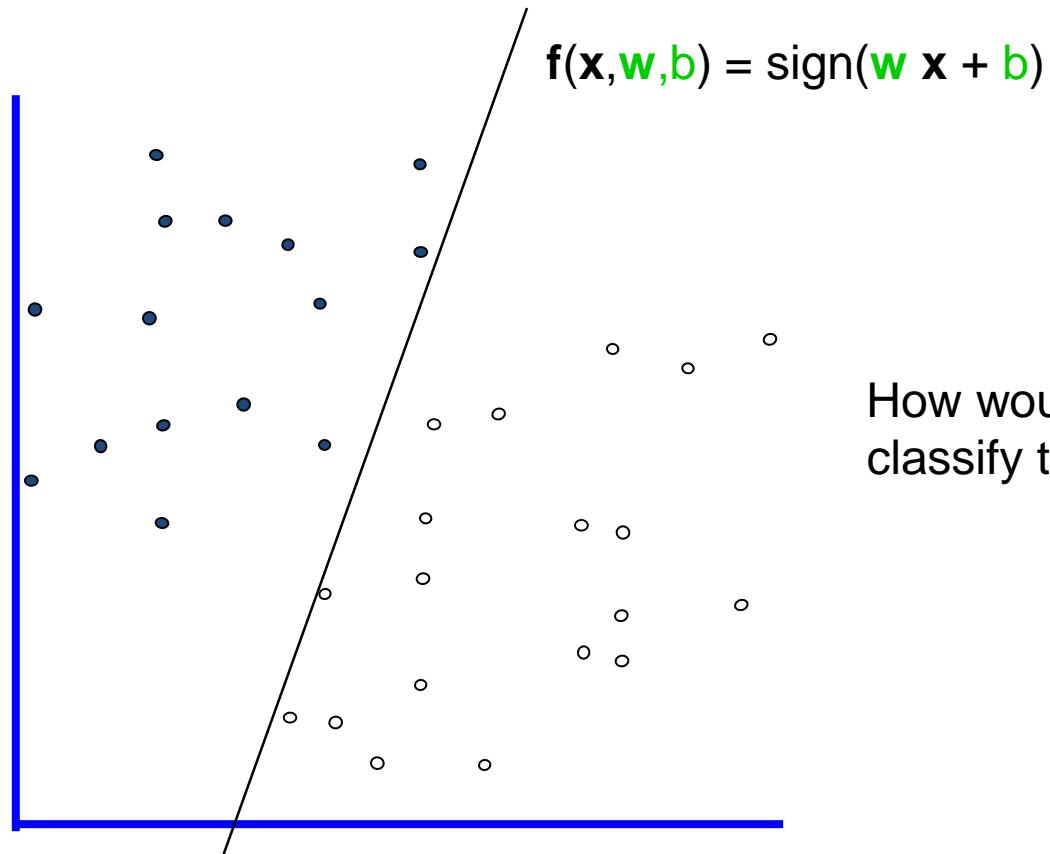
Linear Classifiers

- denotes +1
- denotes -1



Linear Classifiers

- denotes +1
- denotes -1



How would you
classify this data?

Linear Classifiers

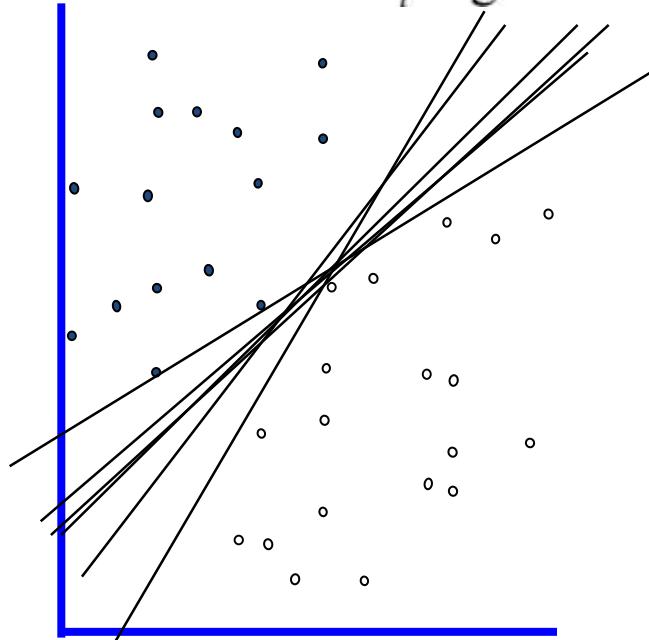
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$$

\mathbf{x}_i positive : $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

\mathbf{x}_i negative : $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

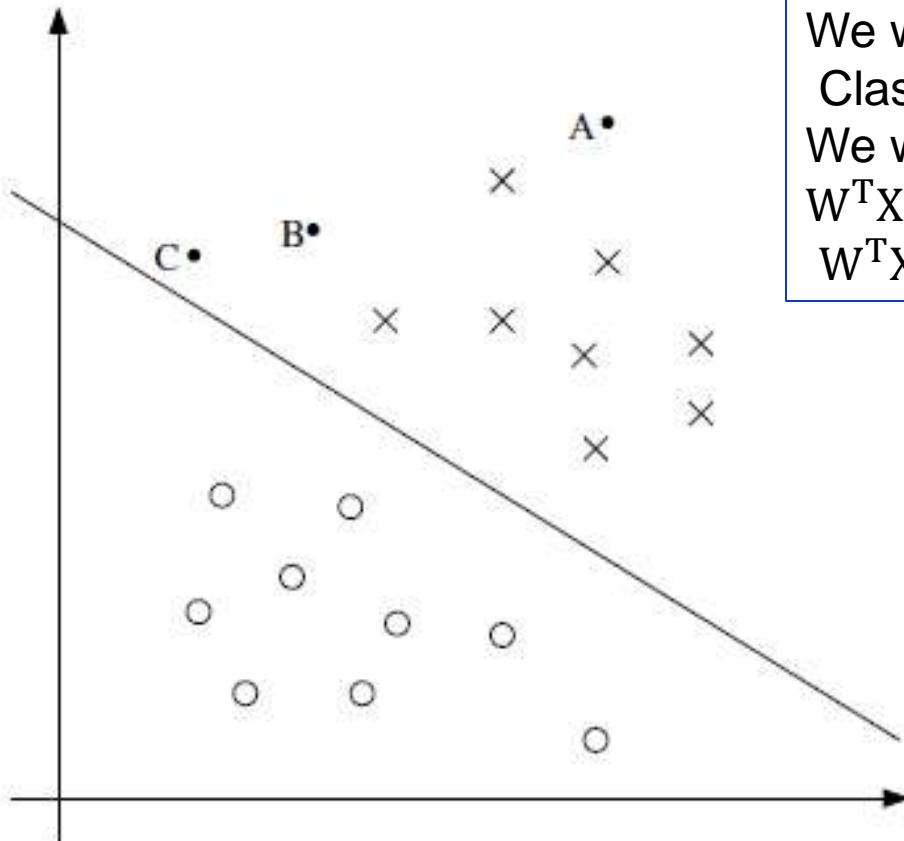
Any of these
would be fine..

..but which is
best?



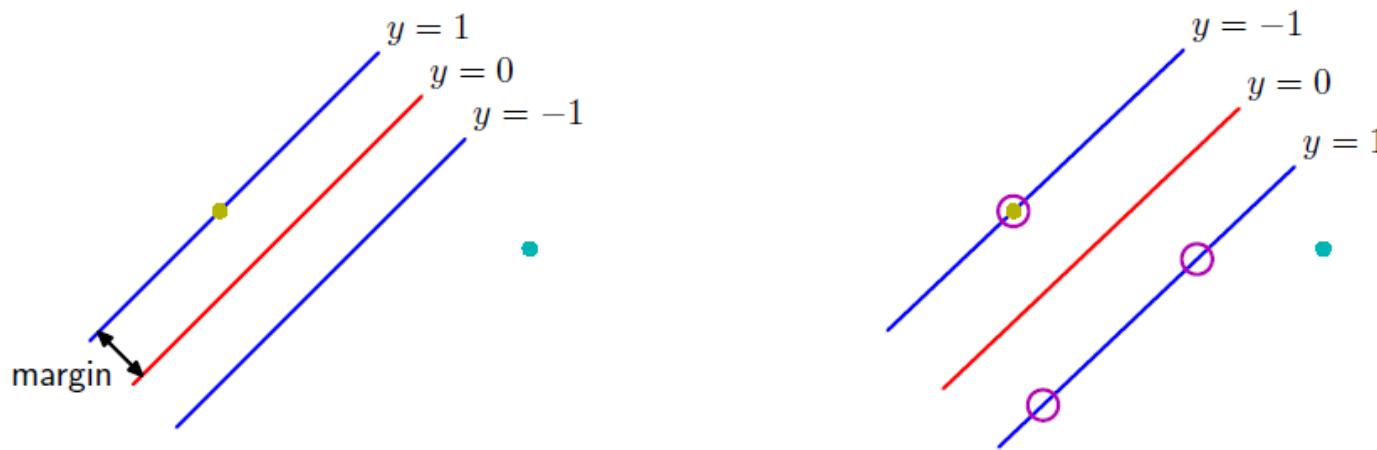
- We may note that so far the classification error is concerned (with training data), all of them are with zero error.
- However, there is no guarantee that all hyperplanes perform equally well on unseen (i.e., test) data.

Decision Boundary - SVM



We would be more confident for Classification of A then B and C i.e
We want
 $W^T X \gg 0$ for $y = 1$ and
 $W^T X \ll 0$ for $y = -1$

Decision Boundary - SVM



- The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure.
- Maximizing the margin leads to a particular choice of decision boundary, as shown on the right.
- The location of this boundary is determined by a subset of the data points, known as **support vectors**, which are indicated by the circles.

Decision Boundary - Margin

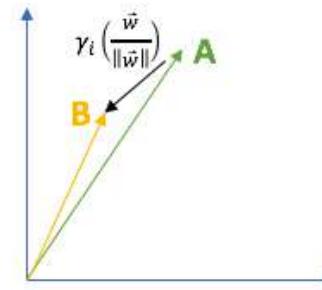
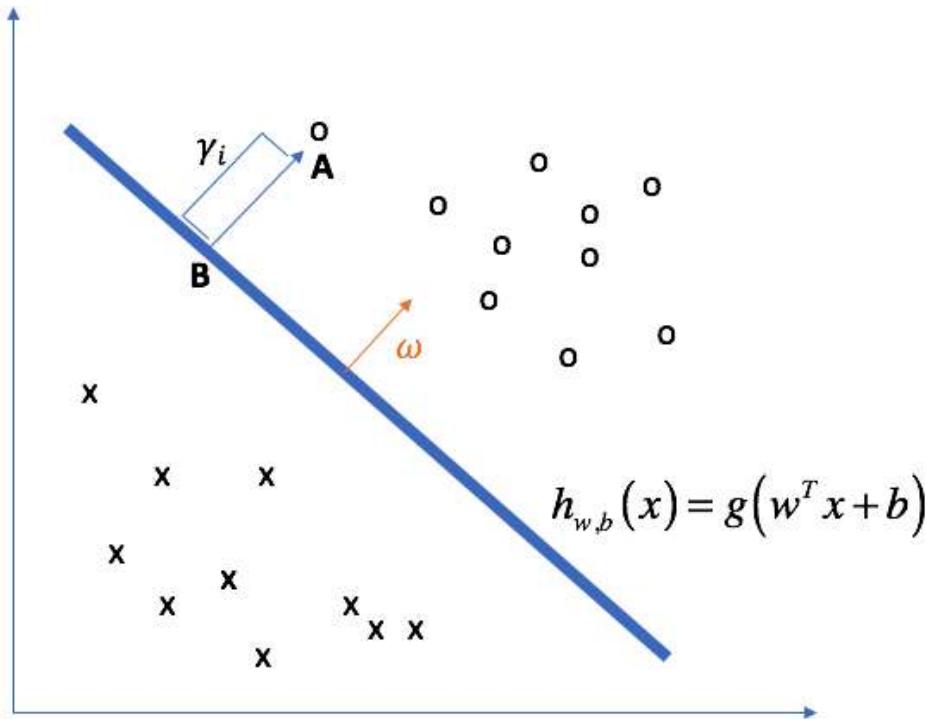
- Functional margin gives the position of the point with respect to the plane, which does not depend on the magnitude.
- It is a measure of correctness of a classification for a data unit
- The **geometric margin** is defined as the distance from a data point to the decision boundary. It is a functional margin scaled by $\|w\|$

$$\gamma^i = \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|}$$

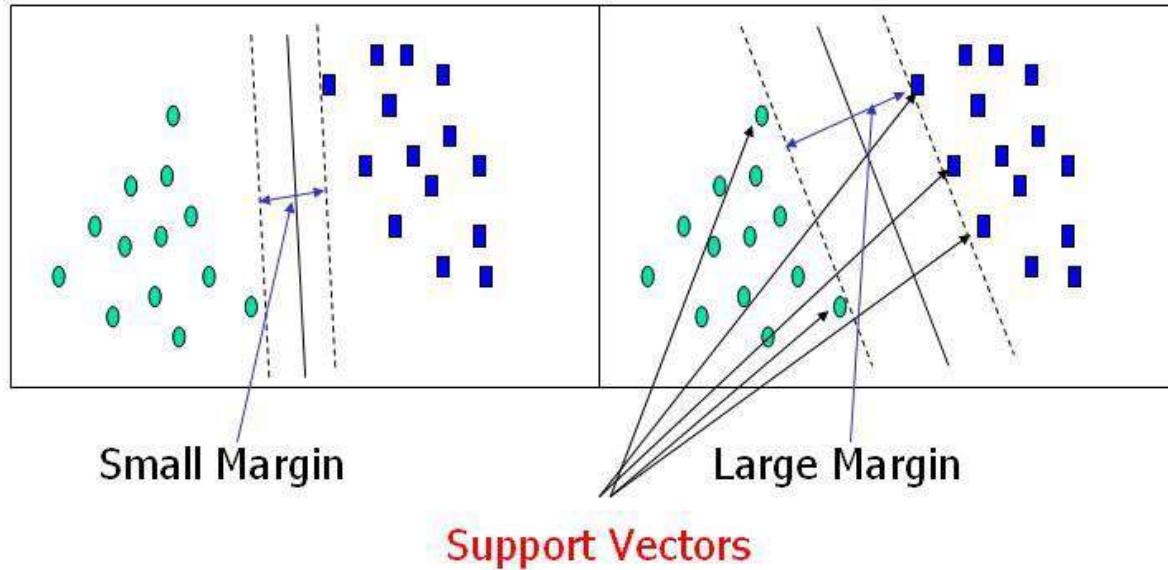
geometric margin
Functional margin

- Functional margin gives a number but without a reference we can't tell if the point is actually far away or close to the decision plane.
- The geometric margin gives not only if the point is properly classified or not, but the magnitude of that distance in term of units of $\|w\|$

Margin



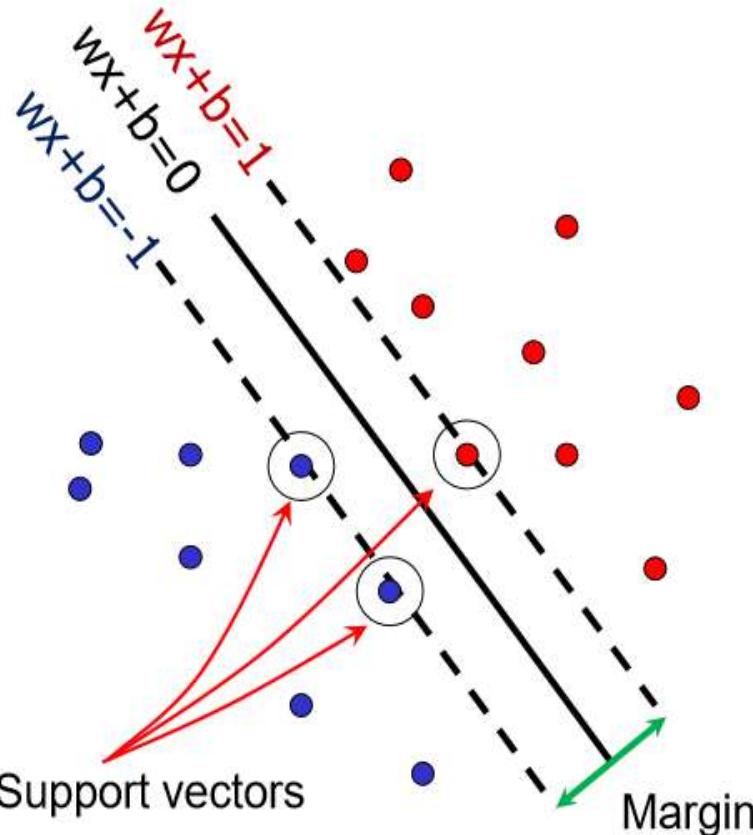
Large Margin & Support Vectors



1. Classifier that contains hyperplane with a small margin are more susceptible to model over fitting and tend to classify with weak confidence on unseen data.
2. Thus during the training or learning phase, the approach would be to search for the hyperplane with maximum margin. Such a hyperplane is called maximum margin hyperplane

Support Vector Machine

- Want line that maximizes the margin.



suppose that all the training data satisfy the following constraints:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

- Find the \mathbf{W} (that defines the decision boundary) that correctly classifies the data with biggest possible margin

- Essentially, the support vectors are the most difficult tuples to classify and give the most information regarding classification.

Support Vectors

- Geometric description of SVM is that the max-margin hyperplane is completely determined by those points that lie nearest to it.
- Points that lie on this margin are the support vectors.
- The points of our data set which if removed, would alter the position of the dividing hyperplane
- For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin.

Margin

Define the hyperplanes H such that:

$$w \cdot x_i + b \geq +1 \text{ when } y_i = +1$$

$$w \cdot x_i + b \leq -1 \text{ when } y_i = -1$$

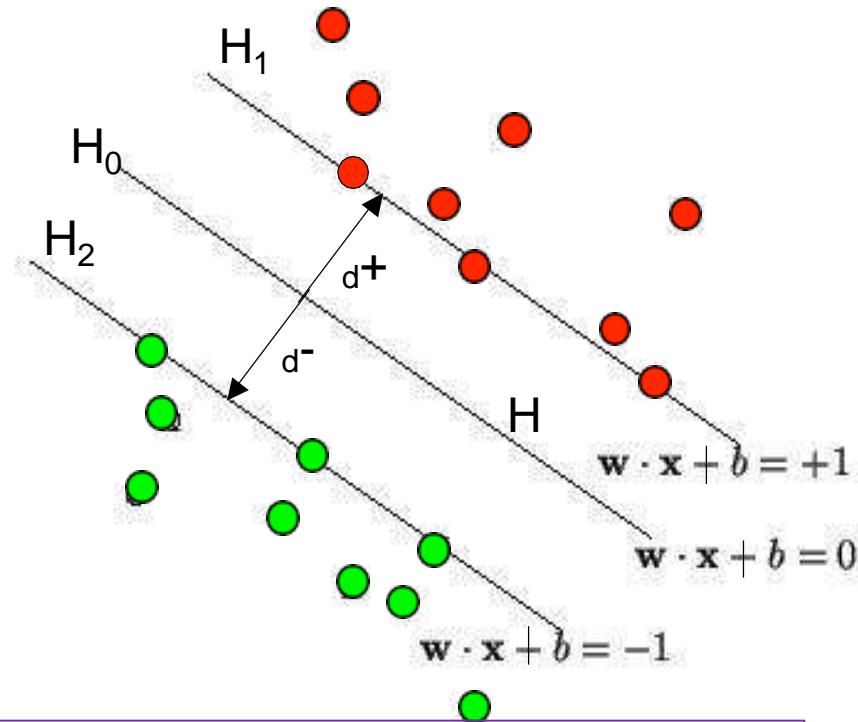
H_1 and H_2 are the planes:

$$H_1: w \cdot x_i + b = +1$$

$$H_2: w \cdot x_i + b = -1$$

The points on the planes H_1 and H_2 are the tips of the Support Vectors

The plane H_0 is the median in between, where $w \cdot x_i + b = 0$



$d+$ = the shortest distance to the closest positive point

$d-$ = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is $d+ + d-$.

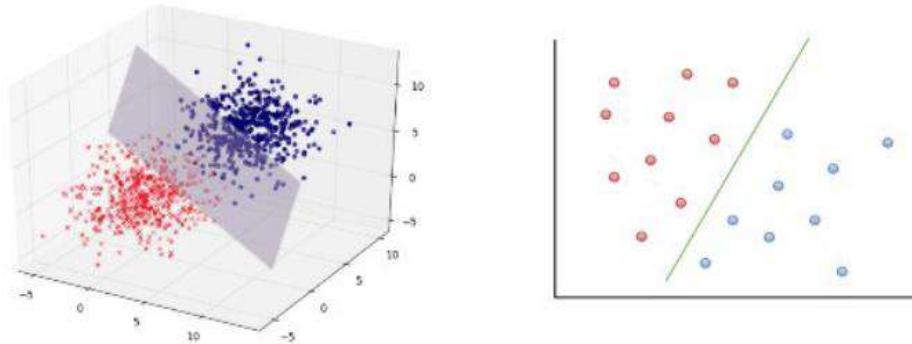
Decision Boundary – Margin Visualization

$$\mathbf{w}^T \mathbf{x} = 0$$

$$y = ax + b$$

Hyperplane

Line

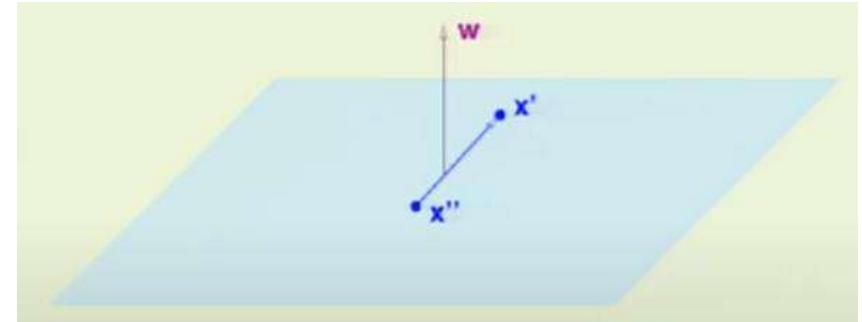


Decision Boundary – Margin – Weight Vector

- Consider points x_a and x_b , which lie on the decision boundary.
- This gives us two equations:

$$w^T x_a + b = 0$$

$$w^T x_b + b = 0$$



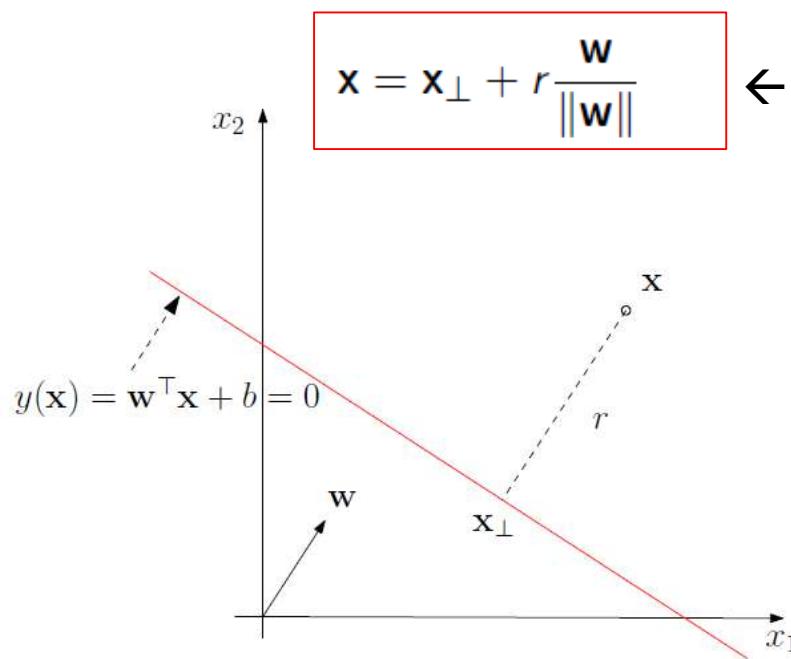
- Weight Vectors are perpendicular to the hyperplane
- Subtracting these two equations gives us

$$w^T(x_a - x_b) = 0$$

Note that the vector $x_a - x_b$ lies on the decision boundary, and it is directed from x_b to x_a .

- Since the dot product $w^T(x_a - x_b)$ is zero, w^T must be orthogonal to $x_a - x_b$ and in turn, to the decision boundary.

Distance of a point to a line - orthogonal projection



← by vector addition

where $\frac{w}{\|w\|}$ = unit vector in the direction of w

x_{\perp} = orthogonal projection of x onto line
 $y(x) = 0$

r is the geometric margin

Multiply left and right by w^T and add b :

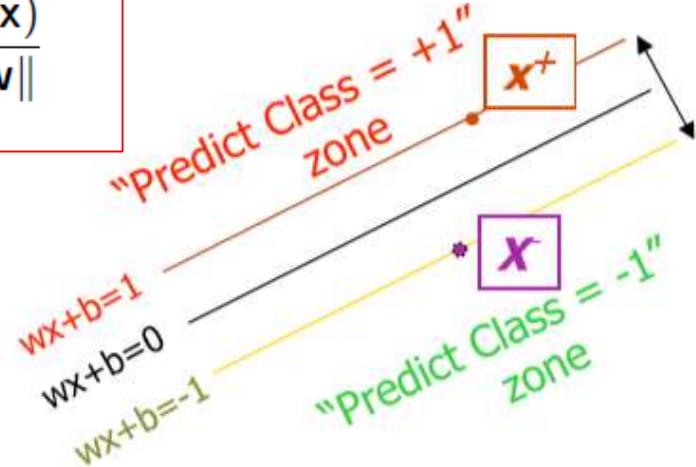
$$\underbrace{w^T x + b}_{y(x)} = \underbrace{w^T x_{\perp} + b}_{0} + r \frac{w^T w}{\|w\|}$$

$$r = y(x) \frac{\|w\|}{\|w\|^2} = \frac{y(x)}{\|w\|}$$

$$\|w\|^2 = w^T w = w_1^2 + \dots + w_{M-1}^2$$

Calculating Margin

$$r = y(\mathbf{x}) \frac{\|\mathbf{w}\|}{\|\mathbf{w}\|^2} = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$



M=Margin Width

$$\mathbf{w} \cdot \mathbf{x}^+ + b = +1$$

$$\mathbf{w} \cdot \mathbf{x}^- + b = -1$$

$$\text{Margin width} = D(\mathbf{x}^+) + D(\mathbf{x}^-)$$

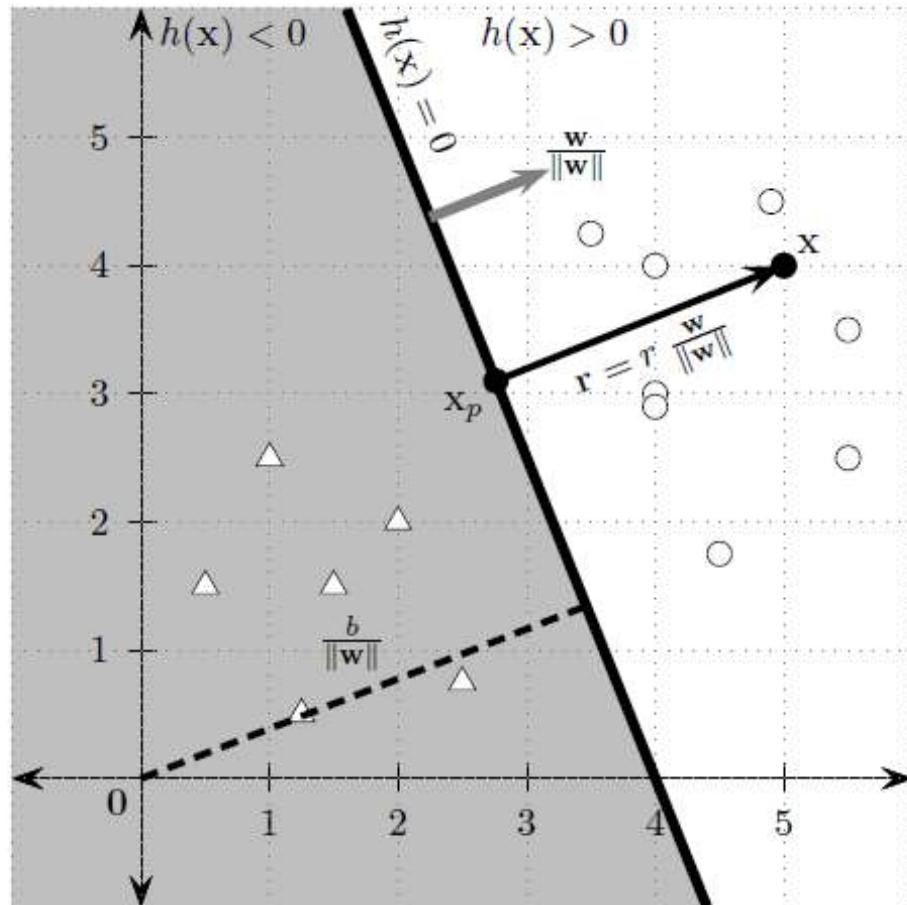
$$= \frac{|\mathbf{w} \cdot \mathbf{x}^+ + b|}{\|\mathbf{w}\|} + \frac{|\mathbf{w} \cdot \mathbf{x}^- + b|}{\|\mathbf{w}\|}$$

$$= \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Distance between lines given by solving linear equation:

$$\text{Maximize margin: } r = \frac{2}{\|\mathbf{w}\|} \rightarrow \text{Equivalent to minimize: } \frac{1}{2} \|\mathbf{w}\|^2$$

Maximum margin



- Displacement of a decision boundary from the origin is controlled by the bias parameter b
- r is positive if r is in the same direction as w , and r is negative if r is in a direction opposite to w .
- r thus gives the offset or directed distance in multiples of the unit weight vector.
- Maximize margin: $r = \frac{2}{\|w\|}$

SVM objective function

- In order to maximize the margin, we thus need to

$$\text{minimize} : \frac{1}{2} \|\mathbf{w}\|^2$$

- With the condition that there are no datapoints between H₁ and H₂:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ when } y_i = +1$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq -1 \text{ when } y_i = -1$$

Can be combined into: $y_i(\mathbf{x}_i \cdot \mathbf{w}) \geq 1$

$$+1(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$-1(\mathbf{w}^T \mathbf{x}_i + b) \leq 1$$

$$\text{same as } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Solving the Optimization Problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$ is minimized;

← Primal

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a quadratic function subject to linear inequality constraints.
- All constraints in SVM are linear
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many algorithms exist for solving them.
- The solution involves constructing a unconstrained problem where a Lagrange multiplier α_i is associated with every constraint in the primary problem:

Solving SVM using KKT conditions

SVM objective function

max f(x) subject to

$$g_i(x) = 0 \text{ and } h_j(x) \geq 0 \text{ for all } i, j$$

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \text{ is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^\top x_i + b) - 1]$$

Solving the Optimization Problem

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

KKT condition : Stationarity

- Taking partial derivative with respect to w , $\frac{\partial L}{\partial w} = 0$
 - $w - \sum \alpha_i y_i x_i = 0$
 - $w = \sum \alpha_i y_i x_i$
- Taking partial derivative with respect to b , $\frac{\partial L}{\partial b} = 0$
 - $- \sum \alpha_i y_i = 0$
 - $\sum \alpha_i y_i = 0$

Solving the Optimization Problem

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T \cdot x_i + b) - 1]$$

- Expanding above equation:

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i y_i w^T \cdot x_i - \sum \alpha_i y_i b + \sum \alpha_i$$

- Substituting $\mathbf{W} = \sum \alpha_i y_i \mathbf{x}_i$ and $\sum \alpha_i y_i = 0$ in above equation

$$L(w, b, \alpha_i) = \frac{1}{2} (\sum_i \alpha_i y_i \mathbf{x}_i) (\sum_j \alpha_j y_j \mathbf{x}_j) - (\sum_i \alpha_i y_i \mathbf{x}_i) (\sum_j \alpha_j y_j \mathbf{x}_j) + \sum \alpha_i$$

$$L(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \alpha_i y_i \mathbf{x}_i) (\sum_j \alpha_j y_j \mathbf{x}_j)$$

$$\mathbf{L}(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j) \quad \rightarrow \text{Wolfe dual Lagrangian function.}$$

The Dual Problem

- This is constrained optimization problem , popularly known as convex optimization problem, where objective function is quadratic and constraints are linear in the parameters W and b. The original problem is known as the **primal problem**
 - **The solution involves constructing a *dual problem* where a *Lagrange multiplier* is associated with every constraint in the primary problem:** The new objective function **is in terms of a_i only**
 - It is known as the dual problem: if we know w , we know all a_i ; if we know all a_i , we know w . The objective function of the dual problem needs to be maximized (comes out from the KKT theory)
 - **The duality principle** tells us that an optimization problem can be viewed from two perspectives. The first one is the primal problem, a minimization problem in our case, and the other one is the dual problem, which will be a maximization problem.
-

The Dual Problem

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to $\alpha_i \geq 0$, for any $i = 1, \dots, m$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

→ Properties of α_i when we introduce the Lagrange multipliers

→ The result when we differentiate the original Lagrangian w.r.t. b

When we solve the Wolfe dual problem, we get a vector α containing all Lagrange multipliers.

Lagrange multiplier for support vectors

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Using KKT *complementary slackness* condition
 $\alpha_i [y_i (w^T x_i + b) - 1] = 0$

For this condition to be satisfied either $\alpha_i = 0$ and $y_i (w^T x_i + b) - 1 > 0$ OR
 $y_i (w^T x_i + b) - 1 = 0$ and $\alpha_i > 0$

- Using dual feasibility condition* $\alpha_i \geq 0$

case 1 : $\alpha_i > 0$

- For support vectors: $\{w^T x_i + b = +1 \text{ when } Y=1 \text{ or } w^T x_i + b = -1 \text{ when } Y=-1\}$
 $y_i (w^T x_i + b) - 1 = 0$

case 2 : $\alpha_i = 0$

- For all points other than support vectors:
 $\alpha_i = 0$

Linear SVM summary – 1

SVM Optimization Problem – primal



Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

→ Objective function

$$L(w, b, \alpha_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

→ Langrangian function

$$\min_{\mathbf{w}, b} \max_{\alpha} \mathcal{L}(\mathbf{w}, b, \alpha)$$

$$\text{subject to } \alpha_i \geq 0, \quad i = 1, \dots, m$$

→ Lagrangian primal problem

Solving the minimization problem involves taking the partial derivatives of L with respect to \mathbf{W} and b .

SVM Optimization Problem - dual

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to $\alpha_i \geq 0$, for any $i = 1, \dots, m$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

→ Wolfe dual problem

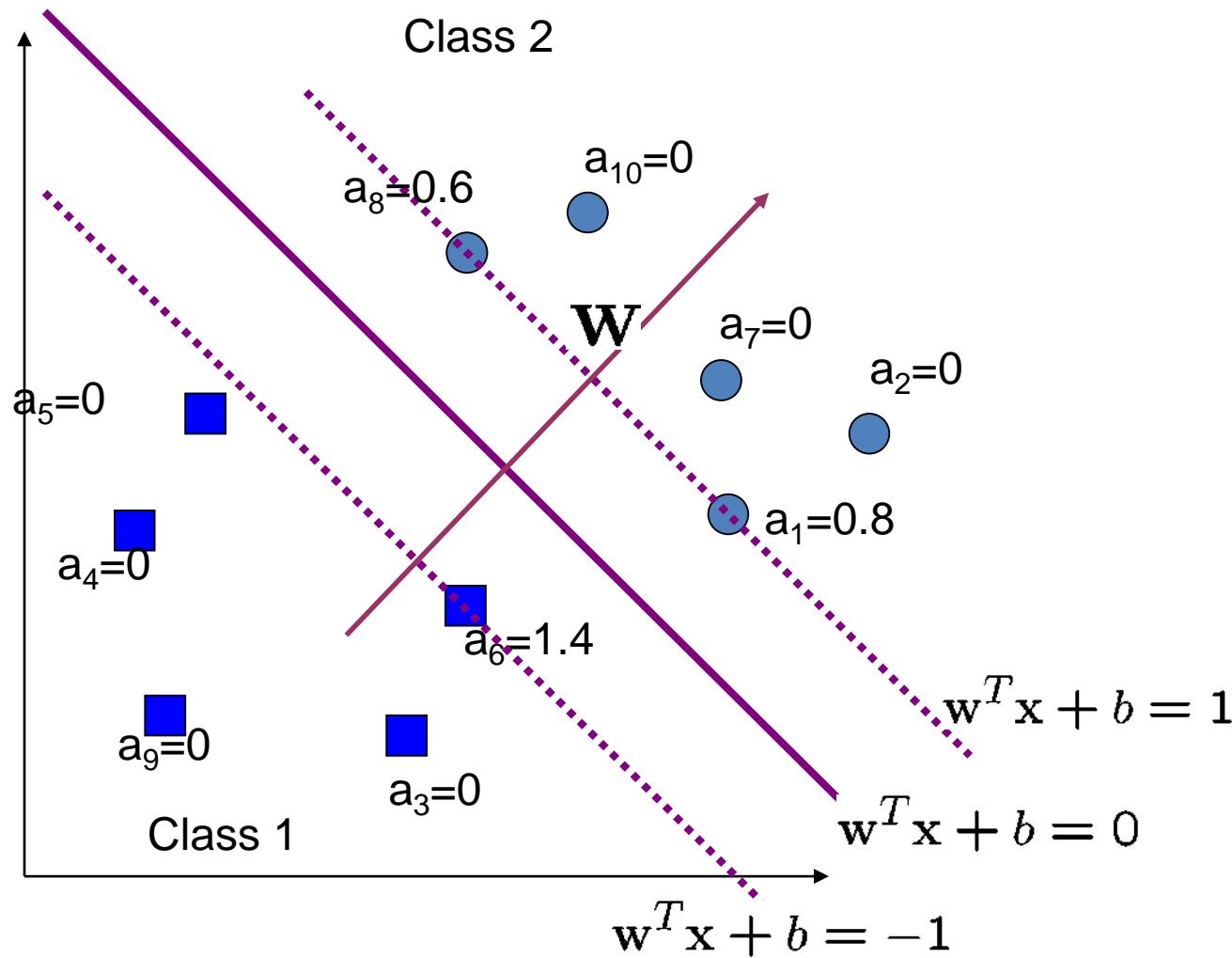
- solve the Wolfe dual problem, we get a vector α containing all Lagrange multipliers.
- Compute w : Taking partial derivative with respect to w
- $\frac{\partial L}{\partial w} = w = \sum \alpha_i y_i x_i$
- Compute b : For support vectors $y_i(w \cdot x_i + b) = 1$

Linear SVM summary – 3

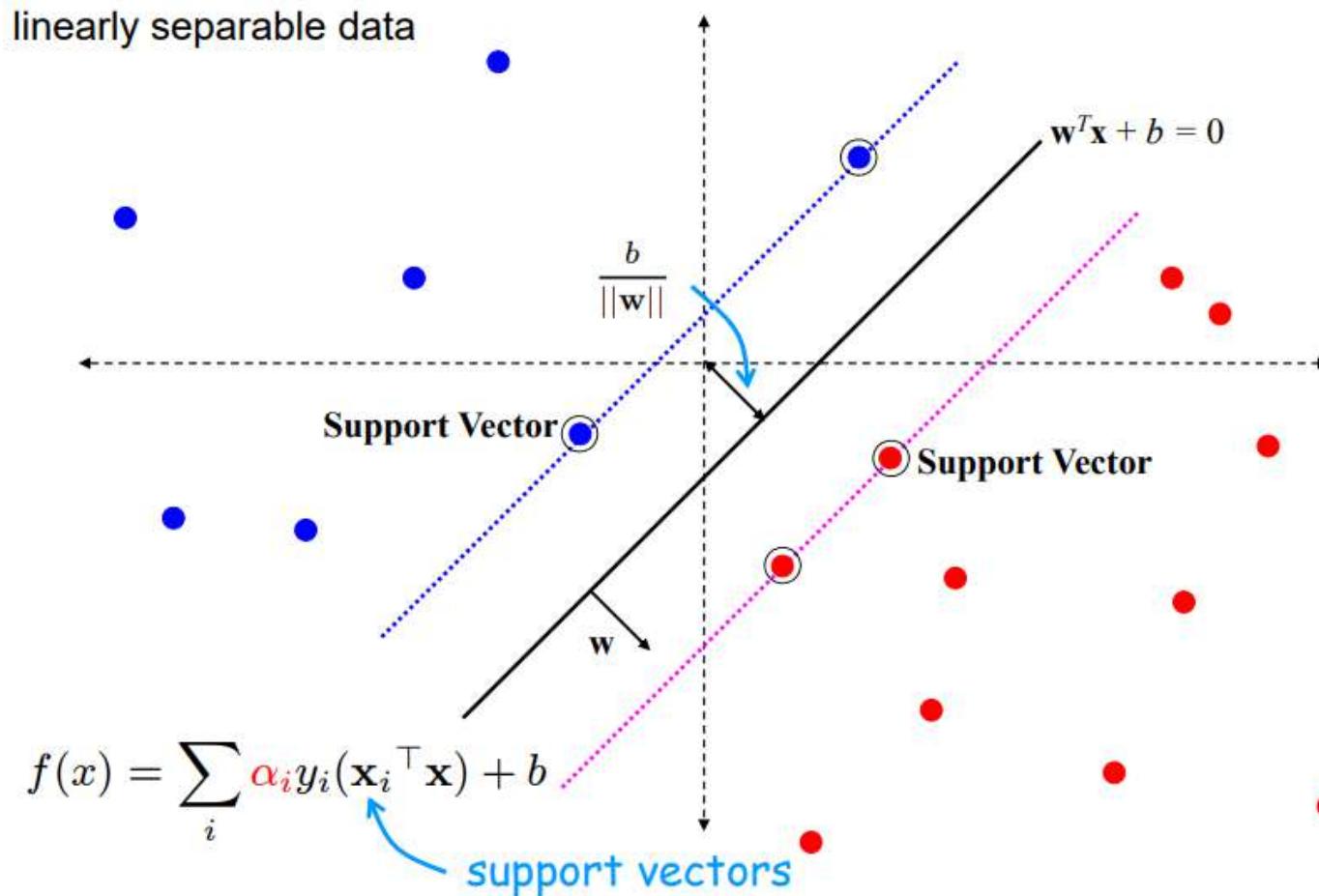
Hypothesis function

- The SVMs use the same hypothesis function as the Perceptron. The class of an example is given by: $h(\mathbf{x}_i) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b)$
- When using the dual formulation, it is computed using only the support vectors: $h(\mathbf{x}_i) = \text{sign}\left(\sum_{j=1}^S \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i) + b\right)$
- $\mathbf{x}_j \Rightarrow$ support vectors
- $\mathbf{x}_i \Rightarrow$ Test example

A Geometrical Interpretation



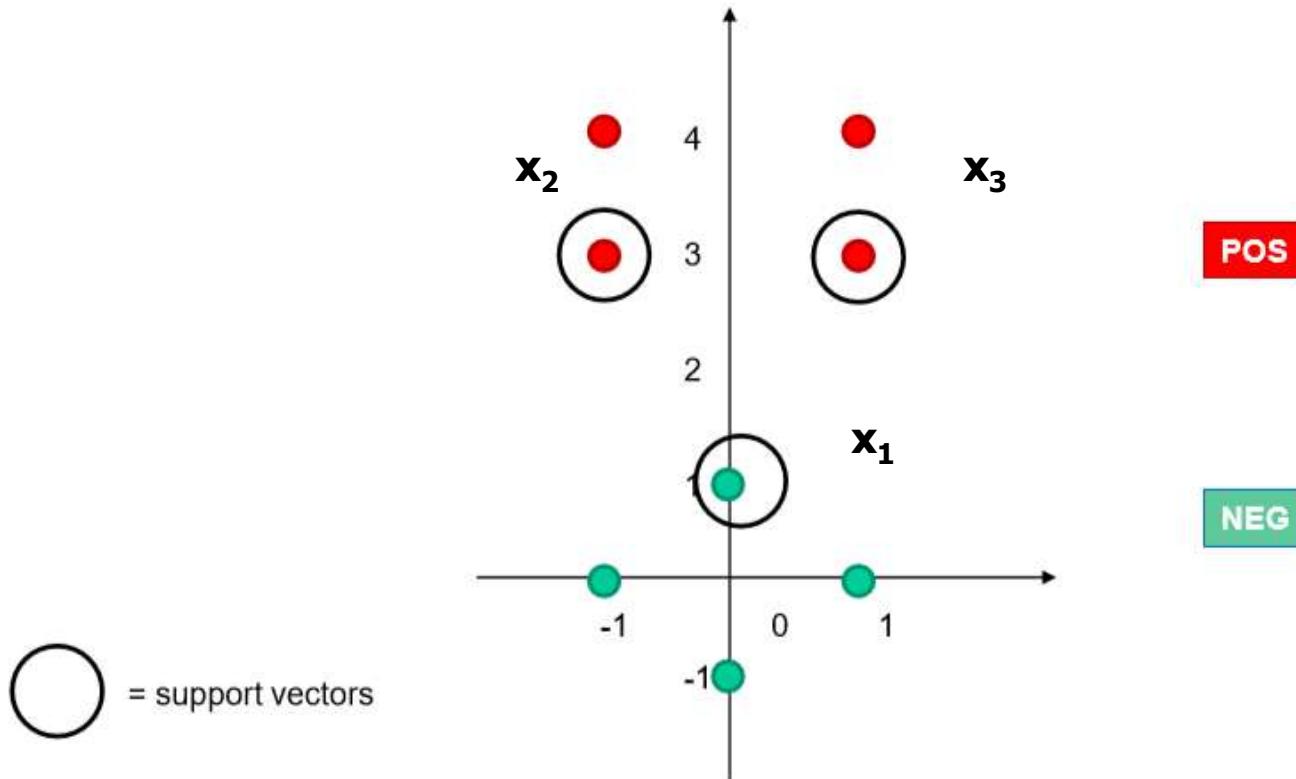
Substituting w in support vectors function



SVM

- Once the SVM is trained with training data, the complexity of the classifier is characterized by the number of support vectors
- Dimensionality of data is not an issue in SVM unlike in other classifier.

Example



Example adapted from Dan Ventura

Solving for α

- We know that for the support vectors, $f(x) = 1$ or -1 exactly
- Add a 1 in the feature representation for the bias
- The support vectors have coordinates and labels:
 - $x_1 = [0 \ 1 \ 1]$, $y_1 = -1$
 - $x_2 = [-1 \ 3 \ 1]$, $y_2 = +1$
 - $x_3 = [1 \ 3 \ 1]$, $y_3 = +1$
- Thus we can form the following system of linear equations:

Solving for α

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

- System of linear equations:

$$\alpha_1 y_1 \text{dot}(\mathbf{x}_1, \mathbf{x}_1) + \alpha_2 y_2 \text{dot}(\mathbf{x}_1, \mathbf{x}_2) + \alpha_3 y_3 \text{dot}(\mathbf{x}_1, \mathbf{x}_3) = y_1$$

$$\alpha_1 y_1 \text{dot}(\mathbf{x}_2, \mathbf{x}_1) + \alpha_2 y_2 \text{dot}(\mathbf{x}_2, \mathbf{x}_2) + \alpha_3 y_3 \text{dot}(\mathbf{x}_2, \mathbf{x}_3) = y_2$$

$$\alpha_1 y_1 \text{dot}(\mathbf{x}_3, \mathbf{x}_1) + \alpha_2 y_2 \text{dot}(\mathbf{x}_3, \mathbf{x}_2) + \alpha_3 y_3 \text{dot}(\mathbf{x}_3, \mathbf{x}_3) = y_3$$

$$-2 * \alpha_1 + 4 * \alpha_2 + 4 * \alpha_3 = -1$$

$$-4 * \alpha_1 + 11 * \alpha_2 + 9 * \alpha_3 = +1$$

$$-4 * \alpha_1 + 9 * \alpha_2 + 11 * \alpha_3 = +1$$

$$\begin{aligned}\alpha_i [-1 (\mathbf{w} \cdot \mathbf{x}_i + b)] &= -1 \\ \alpha_i [+1 (\mathbf{w} \cdot \mathbf{x}_i + b)] &= 1\end{aligned}$$

- Solution: $\alpha_1 = 3.5$, $\alpha_2 = 0.75$, $\alpha_3 = 0.75$

Solving for w , b ; plotting boundary

- We know $w = \sum \alpha_i y_i x_i$ i.e $w = \alpha_1 y_1 x_1 + \dots + \alpha_N y_N x_N$
where N = No of SVs
- Thus $w = -3.5 * [0 1 1] + 0.75 [-1 3 1] + 0.75 [1 3 1] = [0 1 -2]$
- Separating out weights and bias, we have: $w = [0 1]$ and $b = -2$
a=0, c=1

Boundary:

- For SVMs, we used this eq for a line: $ax + cy + b = 0$ where $w = [a c]$
- Thus $ax + b = -cy \rightarrow y = (-a/c)x + (-b/c)$
- Thus y-intercept is $(-b/c) = -(-2)/1 = 2$
- The decision boundary is perpendicular to w and it has slope
 $=(-a/c) = -0/1 = 0$

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

I Select the support vectors:

$$\mathbf{x}_1 = [0 \ 1 \ 1], y_1 = -1$$

$$\mathbf{x}_2 = [-1 \ 3 \ 1], y_2 = +1$$

$$\mathbf{x}_3 = [1 \ 3 \ 1], y_3 = +1$$

$$\alpha_i [-1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = -1$$

$$\alpha_i [+1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = 1$$

II Substitute in Lagrangian function: $L(\mathbf{w}, b, \alpha_i) = \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j)$

$$\begin{aligned} L(\mathbf{w}, b, \alpha_i) &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 y_1 y_1 \mathbf{x}_1 \cdot \mathbf{x}_1 + \alpha_2 \alpha_2 y_2 y_2 \mathbf{x}_2 \cdot \mathbf{x}_2 + \alpha_3 \alpha_3 y_3 y_3 \mathbf{x}_3 \cdot \mathbf{x}_3 \\ &\quad + 2 \alpha_1 \alpha_2 y_1 y_2 \mathbf{x}_1 \cdot \mathbf{x}_2 + 2 \alpha_1 \alpha_3 y_1 y_3 \mathbf{x}_1 \cdot \mathbf{x}_3 + 2 \alpha_2 \alpha_3 y_2 y_3 \mathbf{x}_2 \cdot \mathbf{x}_3) \\ &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 \mathbf{x}_1 \cdot \mathbf{x}_1 + \alpha_2 \alpha_2 \mathbf{x}_2 \cdot \mathbf{x}_2 + \alpha_3 \alpha_3 \mathbf{x}_3 \cdot \mathbf{x}_3 - 2 \alpha_1 \alpha_2 \mathbf{x}_1 \cdot \mathbf{x}_2 - 2 \alpha_1 \alpha_3 \mathbf{x}_1 \cdot \mathbf{x}_3 + 2 \alpha_2 \alpha_3 \mathbf{x}_2 \cdot \mathbf{x}_3) \end{aligned}$$

$$\begin{aligned} &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 [0 \ 1 \ 1] \cdot [0 \ 1 \ 1] + \alpha_2 \alpha_2 [-1 \ 3 \ 1] \cdot [-1 \ 3 \ 1] + \alpha_3 \alpha_3 [1 \ 3 \ 1] \cdot [1 \ 3 \ 1] \\ &\quad - 2 \alpha_1 \alpha_2 [0 \ 1 \ 1] \cdot [-1 \ 3 \ 1] - 2 \alpha_1 \alpha_3 [0 \ 1 \ 1] \cdot [1 \ 3 \ 1] + 2 \alpha_2 \alpha_3 [-1 \ 3 \ 1] \cdot [1 \ 3 \ 1]) \end{aligned}$$

III Find the Unconstrained Optimization Function:

$$L(\mathbf{w}, b, \alpha_i) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 + 11 \alpha_2 \alpha_2 + 11 \alpha_3 \alpha_3 - 8 \alpha_1 \alpha_2 - 8 \alpha_1 \alpha_3 + 18 \alpha_2 \alpha_3)$$

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

IV Gradient of the Lagrangian:

$$\alpha_i [-1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = -1$$

$$\alpha_i [+1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = 1$$

$$L(\mathbf{w}, b, \alpha_1, \alpha_2, \alpha_3) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (\alpha_1 \alpha_1 + 11 \alpha_2 \alpha_2 + 11 \alpha_3 \alpha_3 - 8 \alpha_1 \alpha_2 - 8 \alpha_1 \alpha_3 + 18 \alpha_2 \alpha_3)$$

$$\frac{\partial L}{\partial \alpha_1} = 0 \rightarrow -2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$$

$$\frac{\partial L}{\partial \alpha_2} = 0 \rightarrow -4\alpha_1 + 11\alpha_2 + 9\alpha_3 = +1$$

$$\frac{\partial L}{\partial \alpha_3} = 0 \rightarrow -2\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1$$

V Solve the simultaneous linear equation and find the lagrange multiplier:

$$(\alpha_1, \alpha_2, \alpha_3) = (3.5, 0.75, 0.75)$$

Linear SVM Problem

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

VI Substitute the Lagrange multiplier and obtain the weight's:

$$\alpha_i [-1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = -1$$

$$\alpha_i [+1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = 1$$

$$\begin{aligned} W &= -\alpha_1 [0 \ 1 \ 1] + \alpha_2 [-1 \ 3 \ 1] + \alpha_3 [1 \ 3 \ 1] \\ &= -3.5 [0 \ 1 \ 1] + 0.75 [-1 \ 3 \ 1] + 0.75 [1 \ 3 \ 1] \\ &= [0 \ 1 \ -2] \end{aligned}$$

VII Find the bias with help of any one of the support vectors:

Note the Bias is found above as a part of weight vector!!

VIII Construct the equation of the LSVM hyperplane:

$$W X + b = 0$$

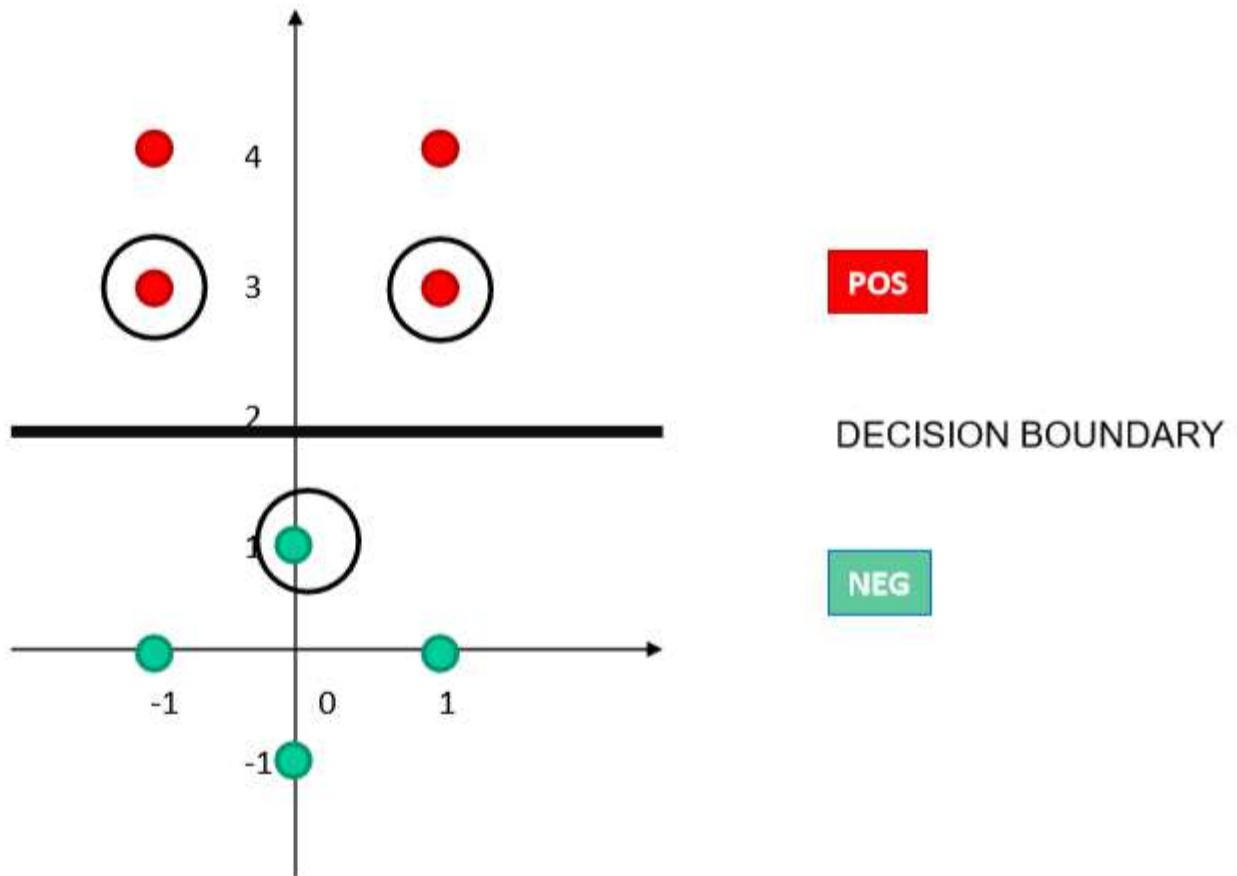
$$Y-2 = 0$$

$$Y=2$$

IX Optionally find the width of the margin:

$$\frac{2}{||w||} H.W$$

Decision boundary



References for SVM

- **Text categorization with Support Vector Machines: learning with many relevant features** - T. Joachims, ECML
 - **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges
 - <http://www.cs.utexas.edu/users/mooney/cs391L/>
 - <https://www.coursera.org/learn/machine-learning/home/week/7>
 - <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
 - <https://data-flair.training/blogs/svm-kernel-functions/>
 - [MIT 6.034 Artificial Intelligence, Fall 2010](https://mit-6.034-fall-2010.readthedocs.io/en/latest/lectures/lec10.html)
 - <https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior>
 - <https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796>
 - <https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be>
 - <https://www.svm-tutorial.com/>
-



Machine Learning

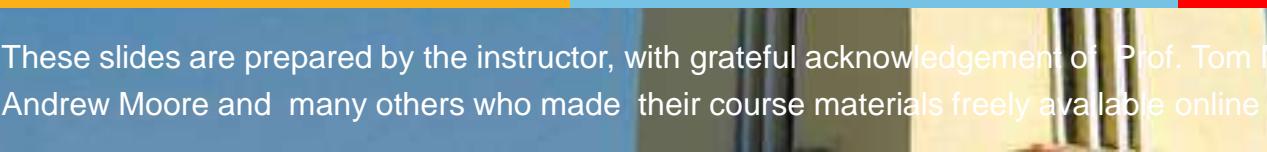
DSECLZG565

Support Vector Machines –II

• Dr. Monali Mavani

BITS Pilani
Pilani Campus

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore and many others who made their course materials freely available online



Topics to be covered

- Soft Margin SVM
- Nonlinear SVM
 - Kernel Trick
 - SVM Kernels
- SVM vs Logistic Regression

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the decision hyperplane (unlike other algorithms like linear regression, ANN etc. where all training points determine decision hyperplane)
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i .

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$ is maximized and

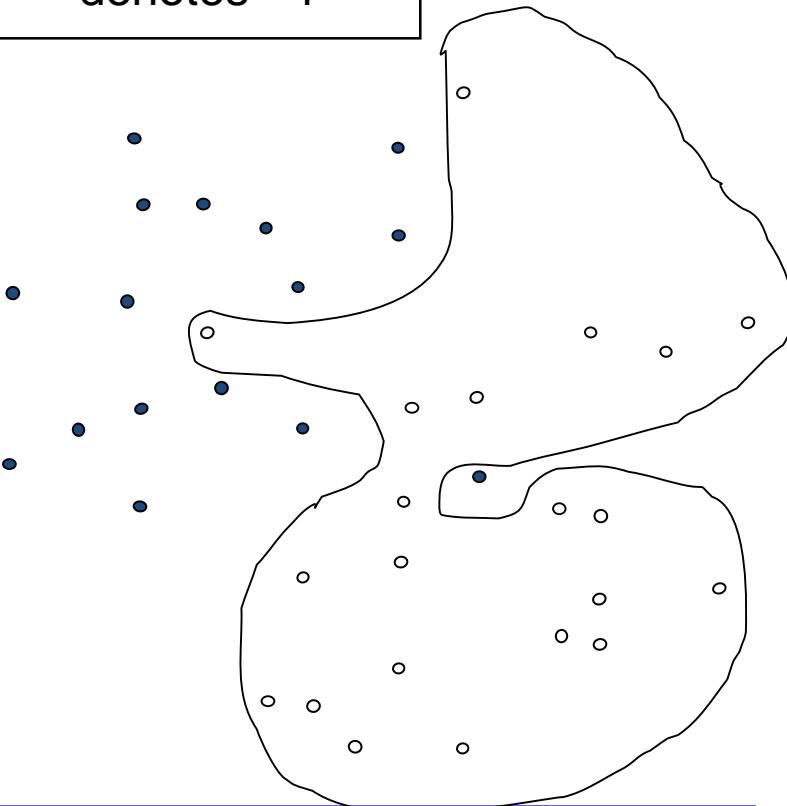
$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

Dataset with noise

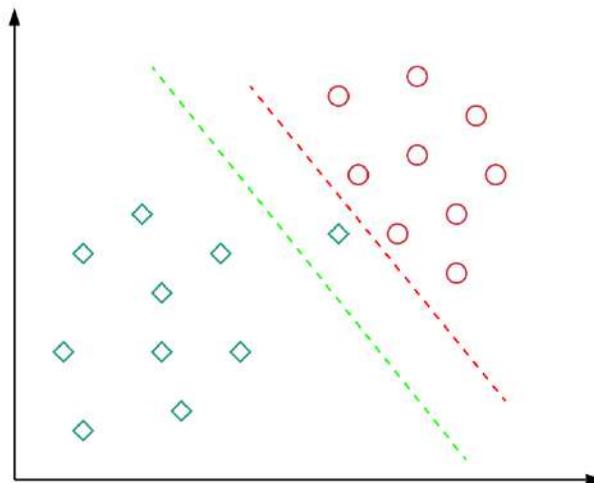
- denotes +1
- denotes -1



- Hard Margin: So far we require all data points be classified correctly
 - No training error
- What if the training set is noisy?

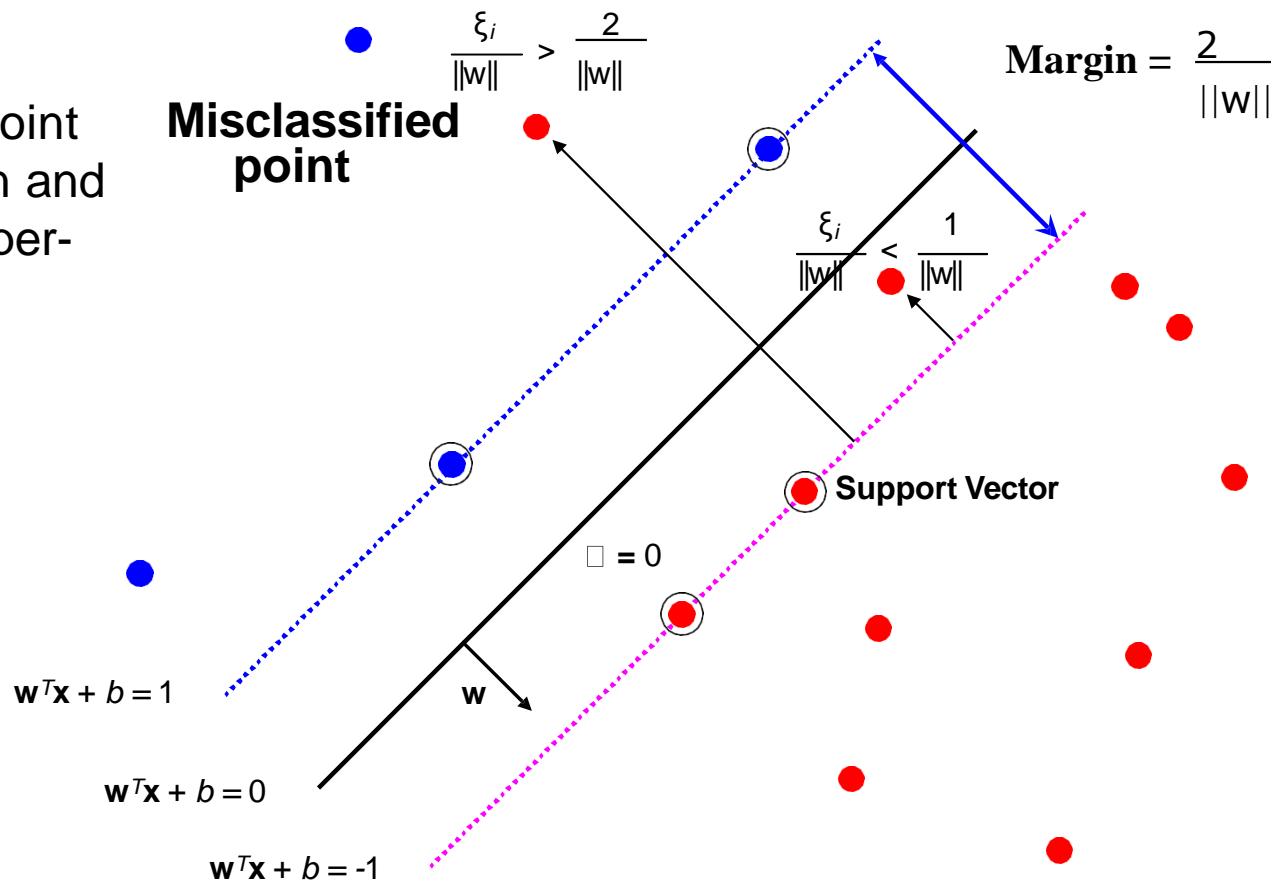
Motivation : Soft margin

- Almost all real-world applications have data that is linearly inseparable.
- When data *is* linearly separable, choosing perfect decision boundary leads to overfitting



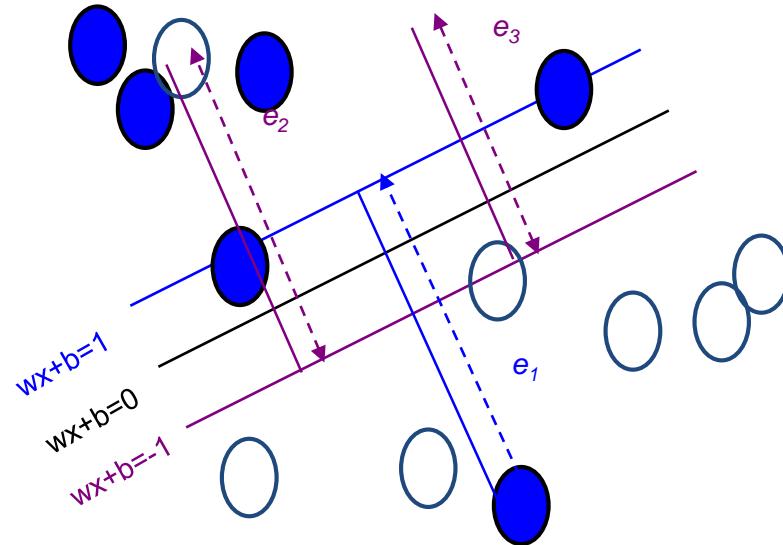
Introduce “slack” variables

- $\xi_i \geq 0$
- for $0 < \xi_i \leq 1$: point is between margin and correct side of hyperplane. This is a margin violation
- for $\xi_i > 1$ point is misclassified



Soft Margin Classification

- ***Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.**
- slack variable ξ_i for every data point x_i ,
- ξ_i is the distance of x_i from the corresponding class's margin if x_i is on the wrong side of the margin, otherwise zero. **It can be regarded as a measure of confidence**
- Points that are far away from the margin on the wrong side would get more penalty



Soft Margin SVMs

$$\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i$$

Misclassification cost # data samples

The w that minimizes... Maximize margin Minimize misclassification Slack variable

subject to $y_i w^T x_i \geq 1 - \xi_i,$
 $\xi_i \geq 0, \quad \forall i = 1, \dots, N$

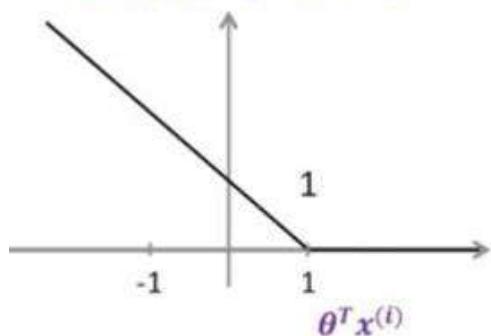
$$\min_{\theta} \frac{1}{2} \|\hat{\theta}\|^2 + C \sum_{i=1}^n \max\{0, 1 - y^{(i)} \theta^T x^{(i)}\}$$

Margin Regularization parameter Cost/Loss of classifying one data-point (hinge loss).

Slack variable-Hinge loss

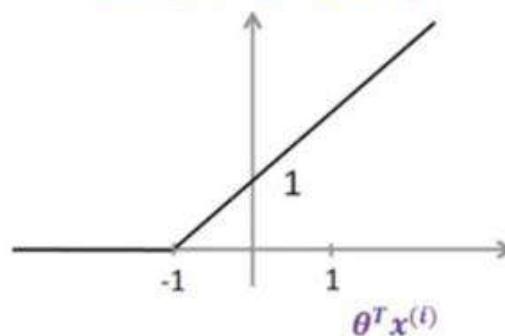
Case where $y^{(i)} = +1$

$$\max\{0, 1 - y^{(i)} \theta^T x^{(i)}\}$$



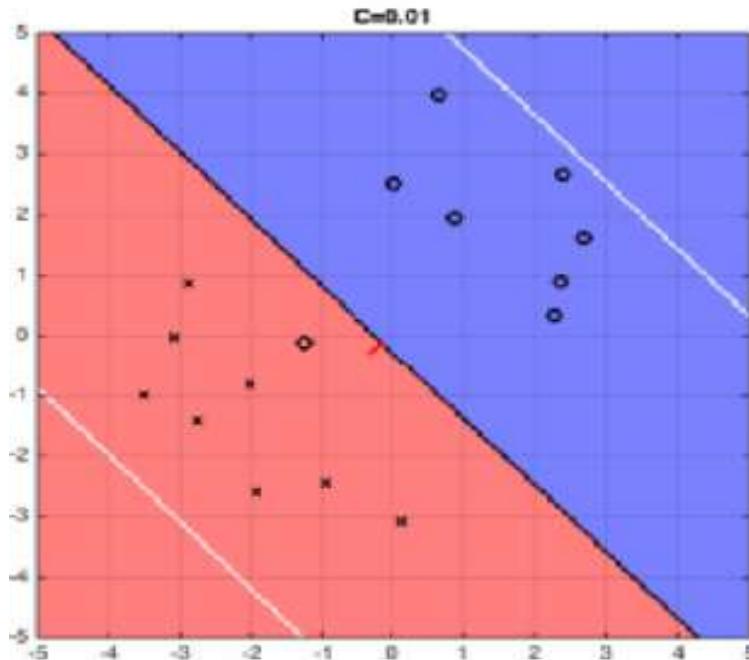
Case where $y^{(i)} = -1$

$$\max[0, 1 - y^{(i)} \theta^T x^{(i)}]$$

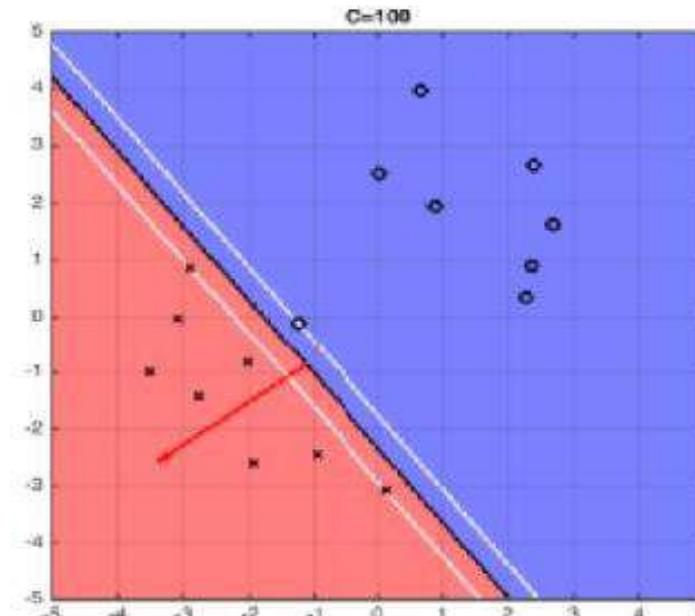


Effect of Margin size v/s misclassification cost

Effect of C



Small value of C will cause the optimizer to look for a larger-margin (small penalties) separating hyperplane, even if that hyperplane misclassifies more points.

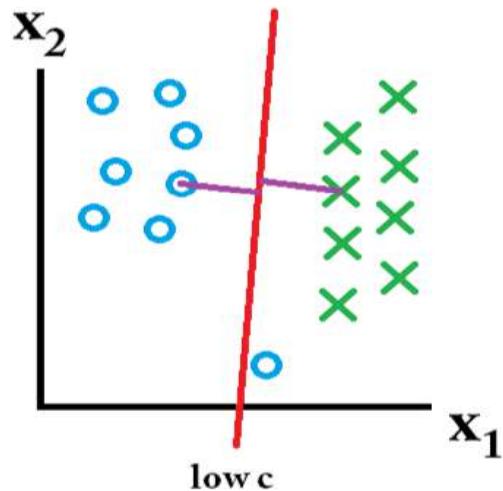


For large values of C, the optimization will choose a smaller-margin (large penalties) hyperplane if that hyperplane does a better job of getting all the training points classified correctly.

C= infinity \rightarrow hard margin SVM

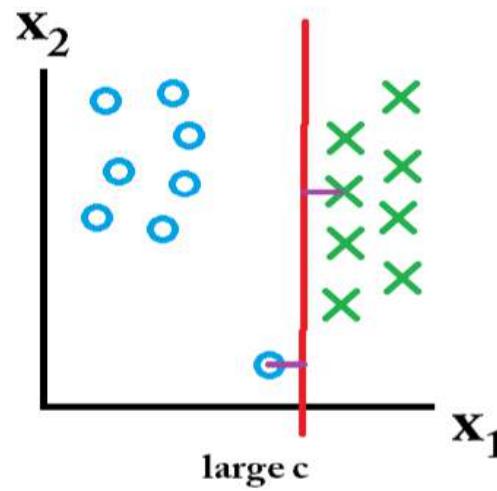
Effect of Margin size v/s misclassification cost

Training set



Misclassification ok, want large margin

Classification mistakes are given less importance



Misclassification not ok

The focus is more on avoiding Misclassification at the expense of keeping the margin small.

Multi Class Problem

Default OvR

One-vs-all (a.k.a. one-vs-others)

- Train C classifiers
- In each, pos = data from class i , neg = data from classes other than i
- The class with the most confident prediction wins
- Example:
 - You have 4 classes, train 4 classifiers
 - 1 vs others: score 3.5
 - 2 vs others: score 6.2
 - 3 vs others: score 1.4
 - 4 vs other: score 5.5
 - Final prediction: class 2

Multi Class Problem

Default OvR

One-vs-one (a.k.a. all-vs-all)

- Train $C(C-1)/2$ binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
 - You have 4 classes, then train 6 classifiers
 - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
 - Votes: 1, 1, 4, 2, 4, 4
 - Final prediction is class 4

SVM Problem

SVM: Optimization

$$\min \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i$$

$$\text{Subject to: } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

SVM: Training

Input: (X,y), C

Output: alpha for support vectors, b

Hyper parameter : C

C parameter tells the SVM optimization how much you want to avoid misclassifying each training example and C can be viewed as a way to control overfitting

SVM: Classification

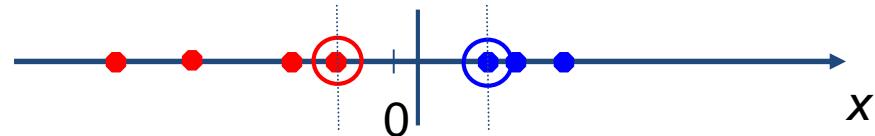
$$\begin{aligned} & sign(\mathbf{w}^T \mathbf{x} + b) \\ &= sign\left(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right) \end{aligned}$$



Nonlinear SVM - kernels

Non-linear SVMs

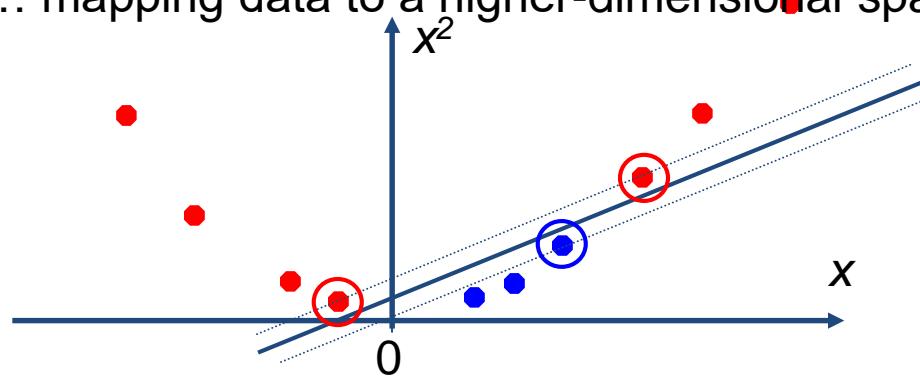
- What if data is not noisy or having outliers(soft margin can work) but inherently non linear
- Datasets that are linearly separable with some noise soft margin work out great:



- But what are we going to do if the dataset is just too hard?



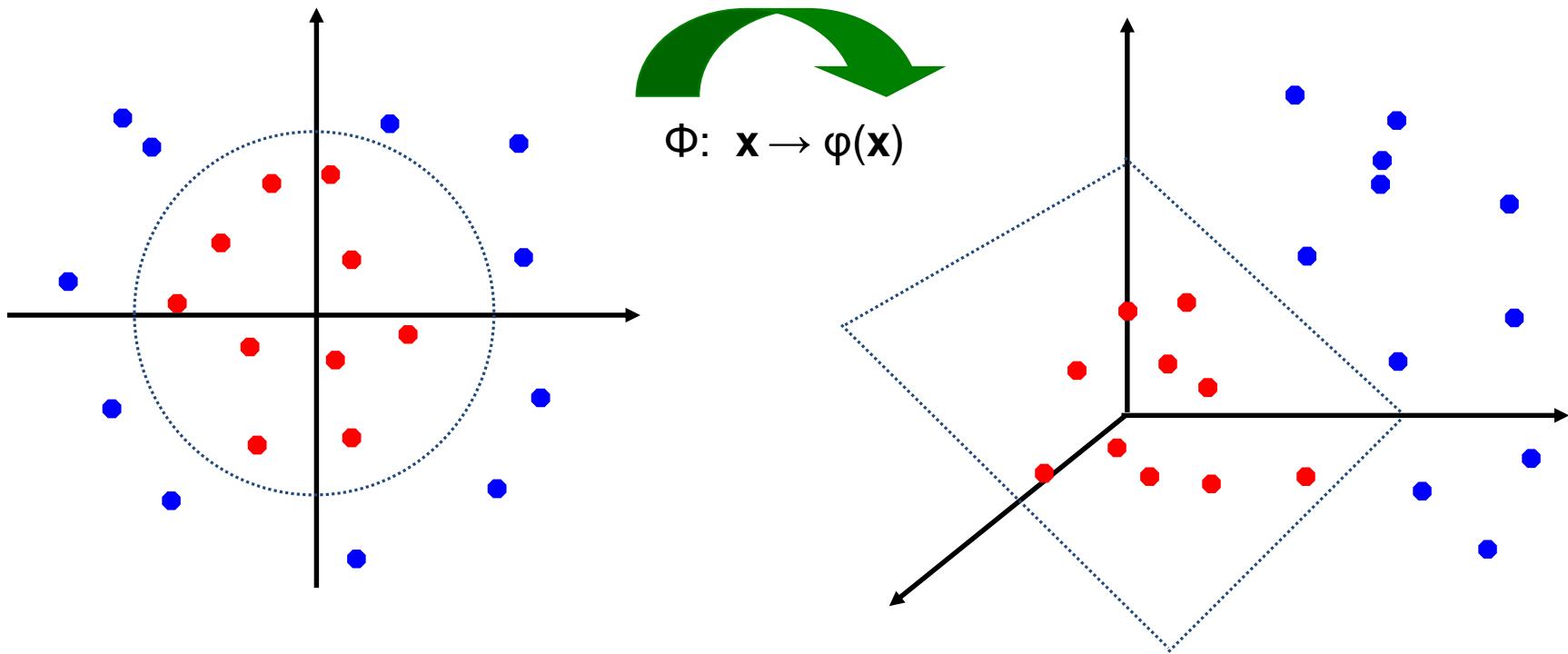
- How about... mapping data to a higher-dimensional space:



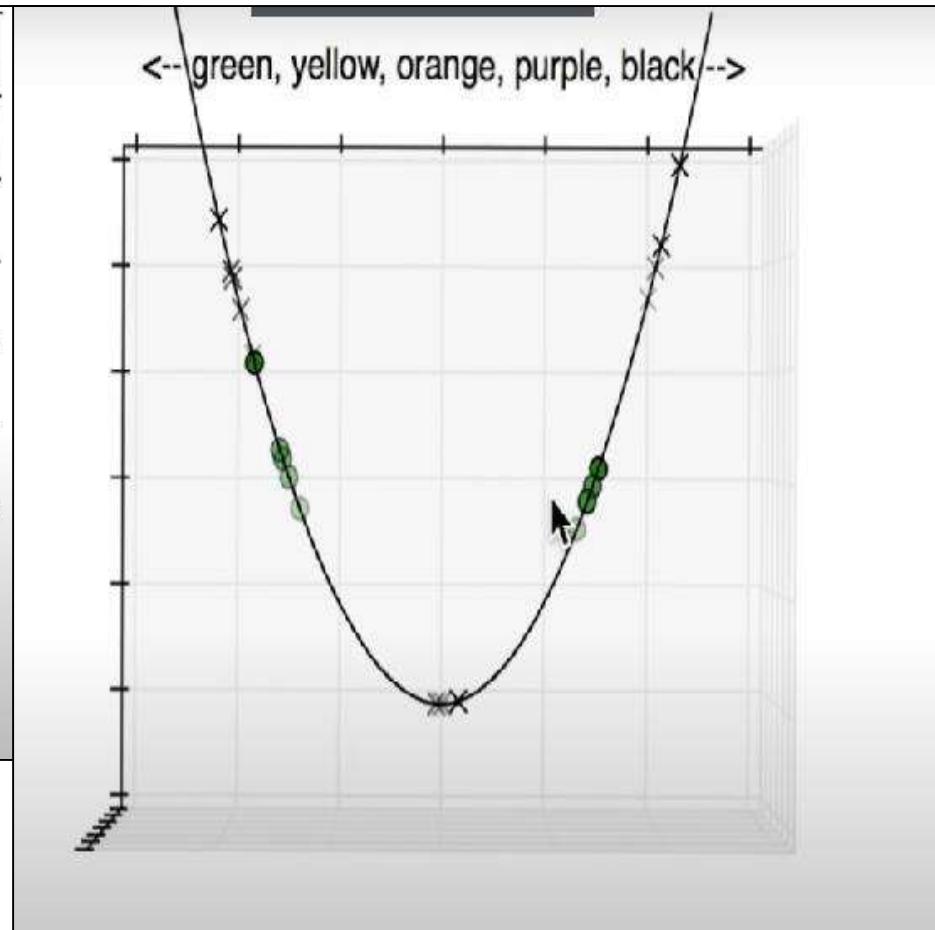
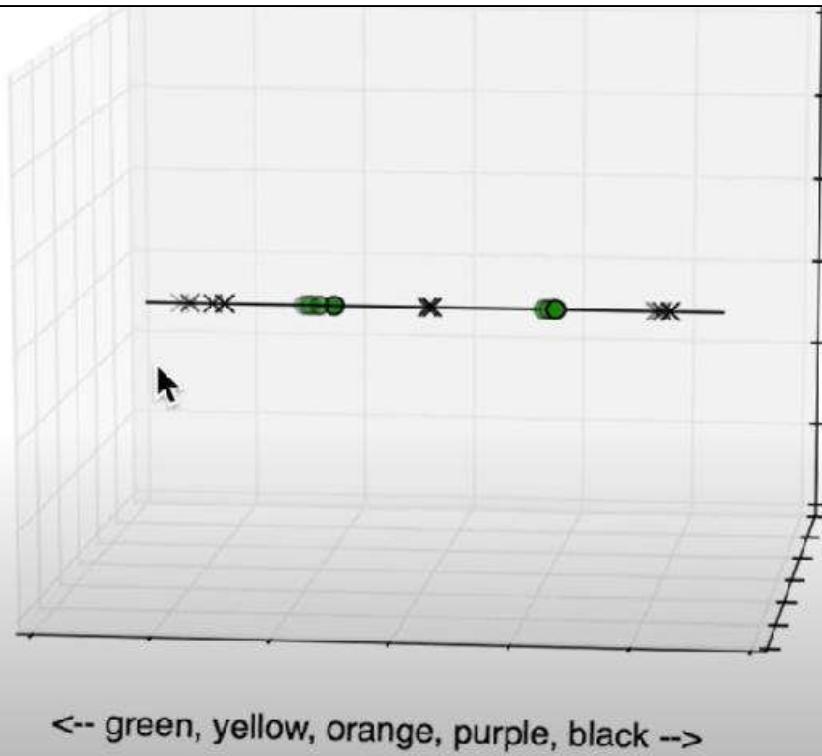
Non-linear SVMs: Feature spaces

General idea:

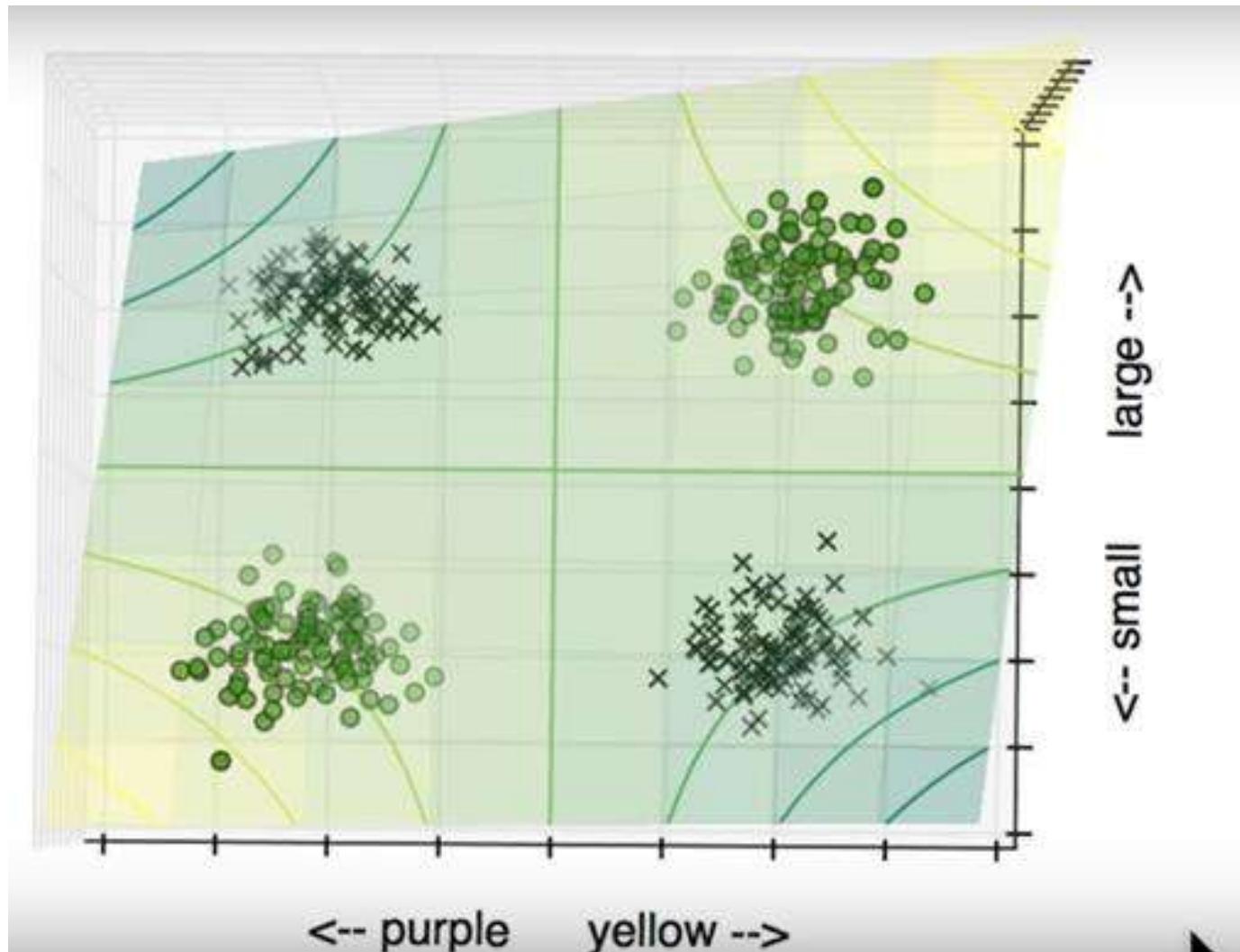
The original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



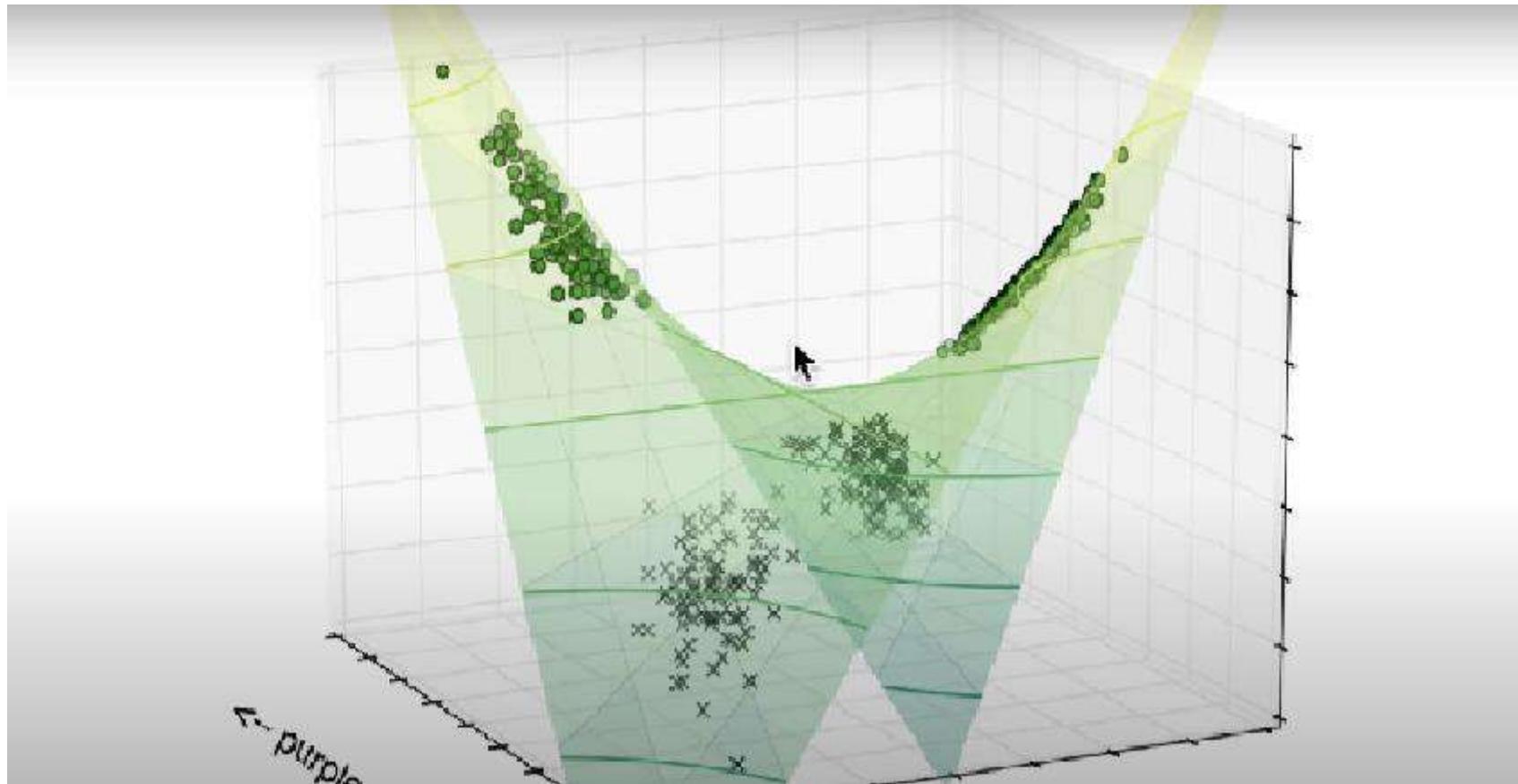
Few more examples : Linear Separation in high dimension



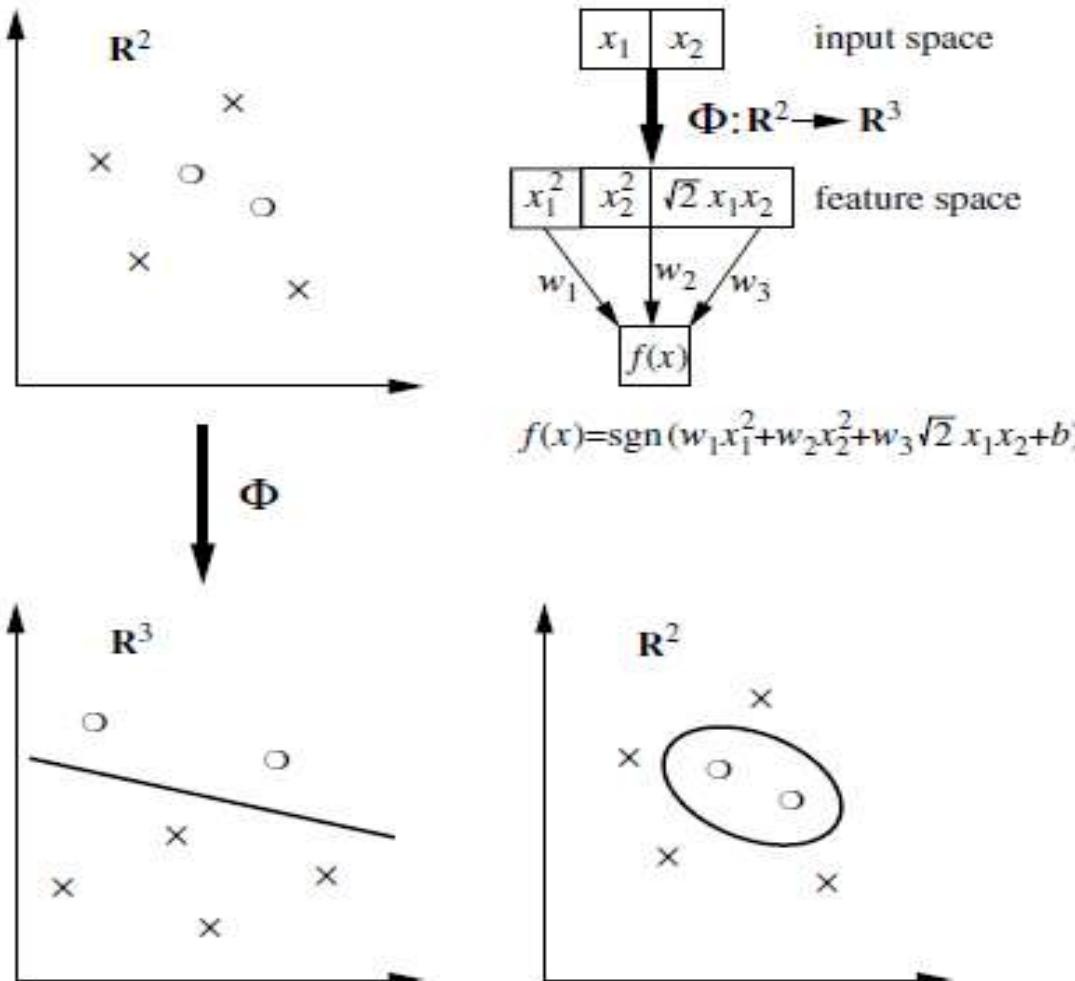
Few more examples : Linear Separation in high dimension



Few more examples : Linear Separation in high dimension

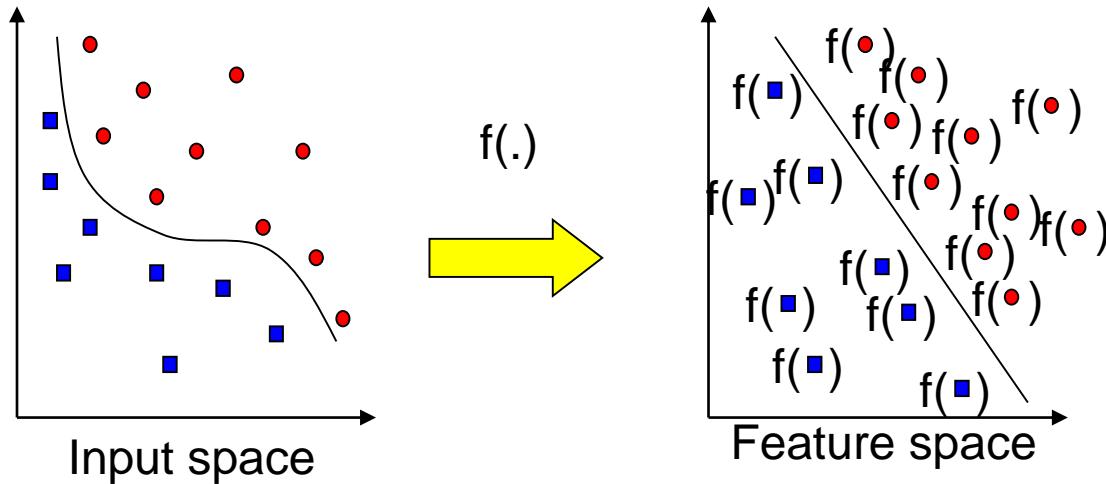


Mapping into a New Feature Space



- Rather than run SVM on x_i , run it on $\Phi(x_i)$
- Find non-linear separator in input space
- What if $\Phi(x_i)$ is really big?
- Use kernels to compute it implicitly!

Transforming the Data



Note: Feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
 - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

SVM Kernels

- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- Function of kernel is to take data as input and transform it into the required form.
- Different SVM algorithms use different types of kernel functions. Example *linear, nonlinear, polynomial, and sigmoid etc.*
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Kernel Trick (SVM)

- E.g. remember the hypothesis function of the original simplified SVM:

$$h_{\theta}(x) = \theta^T x = \theta_0 + \sum_{i=1}^n \alpha_i y^{(i)} \mathbf{x}^T \mathbf{x}^{(i)}$$

- It involves a dot product between the test data-point x and the support vectors $\mathbf{x}^{(i)}$
- Instead of explicitly mapping the data to a higher dimensional space, we can just use a kernel function, and the hypothesis function would have the same form:

$$h_{\theta}(x) = \theta^T x = \theta_0 + \sum_{i=1}^n \alpha_i y^{(i)} k(\mathbf{x}^T \mathbf{x}^{(i)})$$

$\underbrace{}_{z^T z^{(i)}}$

Because since k is a kernel function, we know that $k(x, x^{(i)}) = \underbrace{\Phi(x)^T}_{z^T} \underbrace{\Phi(x^{(i)})}_{z^{(i)}} *$

So we can use the dot product between the higher dimensional vectors, without explicitly knowing them (i.e. a trick).

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$
- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

Kernel: Example Polynomial Kernel

Let $\mathbf{p} = [p_1, p_2]^T$ and $\mathbf{q} = [q_1, q_2]^T$ be two samples in 2D.

$$\begin{aligned}\kappa(\mathbf{p}, \mathbf{q}) &= (\mathbf{p}^T \mathbf{q})^2 = ([p_1, p_2]^T [q_1, q_2])^2 \\ &= (p_1 q_1)^2 + (p_2 q_2)^2 + 2p_1 q_1 p_2 q_2\end{aligned}$$

This is equivalent to: $\phi(\mathbf{p}) = \phi\left(\begin{array}{c} p_1 \\ p_2 \end{array}\right) = \left[\begin{array}{c} p_1^2 \\ p_2^2 \\ \sqrt{2}p_1 p_2 \end{array}\right]$

Kernel: Example

Example:

Let $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$, $\textcolor{red}{x}^{(j)} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}$. $k(x^{(i)}, x^{(j)}) = (1 + x^{(i)T} x^{(j)})^2$

- Is this a kernel function ?
- We need to show that $k(x^{(i)}, x^{(j)}) = \Phi(x^{(i)})^T \Phi(x^{(j)})$

Kernel: Example

Example:

$$\text{Let } x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}, \quad x^{(j)} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}, \quad k(x^{(i)}, x^{(j)}) = (1 + x^{(i)T} x^{(j)})^2$$

$$k(x^{(i)}, x^{(j)}) = 1 + x_1^{(i)2} x_1^{(j)2} + 2x_1^{(i)} x_1^{(j)} x_2^{(i)} x_2^{(j)} + x_2^{(i)2} x_2^{(j)2} + 2x_1^{(i)} x_1^{(j)} + 2x_2^{(i)} x_2^{(j)}$$

$$= \begin{bmatrix} 1 & x_1^{(i)2} & \sqrt{2}x_1^{(i)}x_2^{(i)} & x_2^{(i)2} & \sqrt{2}x_1^{(i)} & \sqrt{2}x_2^{(i)} \end{bmatrix} \begin{bmatrix} 1 \\ x_1^{(j)2} \\ \sqrt{2}x_1^{(j)}x_2^{(j)} \\ x_2^{(j)2} \\ \sqrt{2}x_1^{(j)} \\ \sqrt{2}x_2^{(j)} \end{bmatrix}$$

{ $\Phi(x^{(i)})^T$ } { $\Phi(x^{(j)})$ }

So, yes, this is a kernel function.

Mercer kernels

- What functions are valid kernels that correspond to feature vectors $\phi(x)$?
- Mercer kernels K
 - K is continuous
 - K is symmetric
 - K is positive semi-definite, i.e. $x^T K x \geq 0$ for all x
 - Ensures optimization is concave maximization

What Functions are Kernels?

1) We can *construct kernels from scratch*:

- For any $\varphi : \mathcal{X} \rightarrow \mathbb{R}^m$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathbb{R}^m}$ is a kernel.
- If $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *distance function*, i.e.
 - $d(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$,
 - $d(x, x') = 0$ only for $x = x'$,
 - $d(x, x') = d(x', x)$ for all $x, x' \in \mathcal{X}$,
 - $d(x, x') \leq d(x, x'') + d(x'', x')$ for all $x, x', x'' \in \mathcal{X}$,

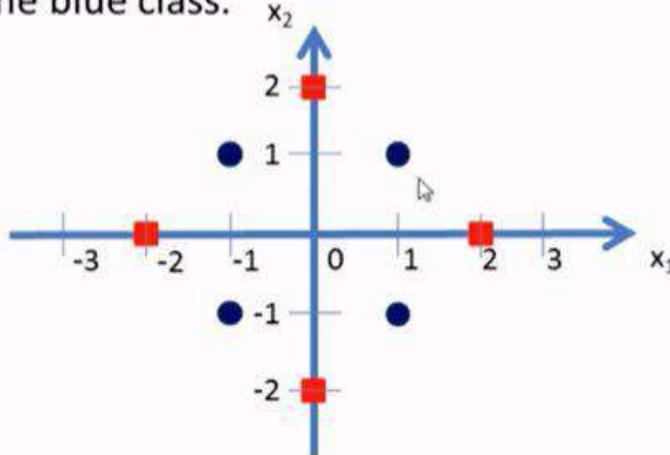
then $k(x, x') := \exp(-d(x, x'))$ is a kernel.

2) We can *construct kernels from other kernels*:

- if k is a kernel and $\alpha > 0$, then αk and $k + \alpha$ are kernels.
- if k_1, k_2 are kernels, then $k_1 + k_2$ and $k_1 \cdot k_2$ are kernels.

Example-1

- Obviously there is no clear separating hyperplane between the red class and the blue class.



- Blue class vectors are: $\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

- Red class vectors are: $\begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -2 \end{pmatrix}$

Kernel: Example

- Here we need to find a non-linear mapping function Φ which can transform these data into a new feature space where a separating hyperplane can be found.
- Let us consider the following mapping function.

$$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 6 - x_1 + (x_1 - x_2)^2 \\ 6 - x_2 + (x_1 - x_2)^2 \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} \geq 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$

Example-1

- Now let us transform the blue and red class vectors using the non-linear mapping function Φ .
- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 6 - x_1 + (x_1 - x_2)^2 \\ 6 - x_2 + (x_1 - x_2)^2 \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} \geq 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$
- Blue class vectors are: $\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ no change since $\sqrt{x_1^2 + x_2^2} < 2$ for all the vectors

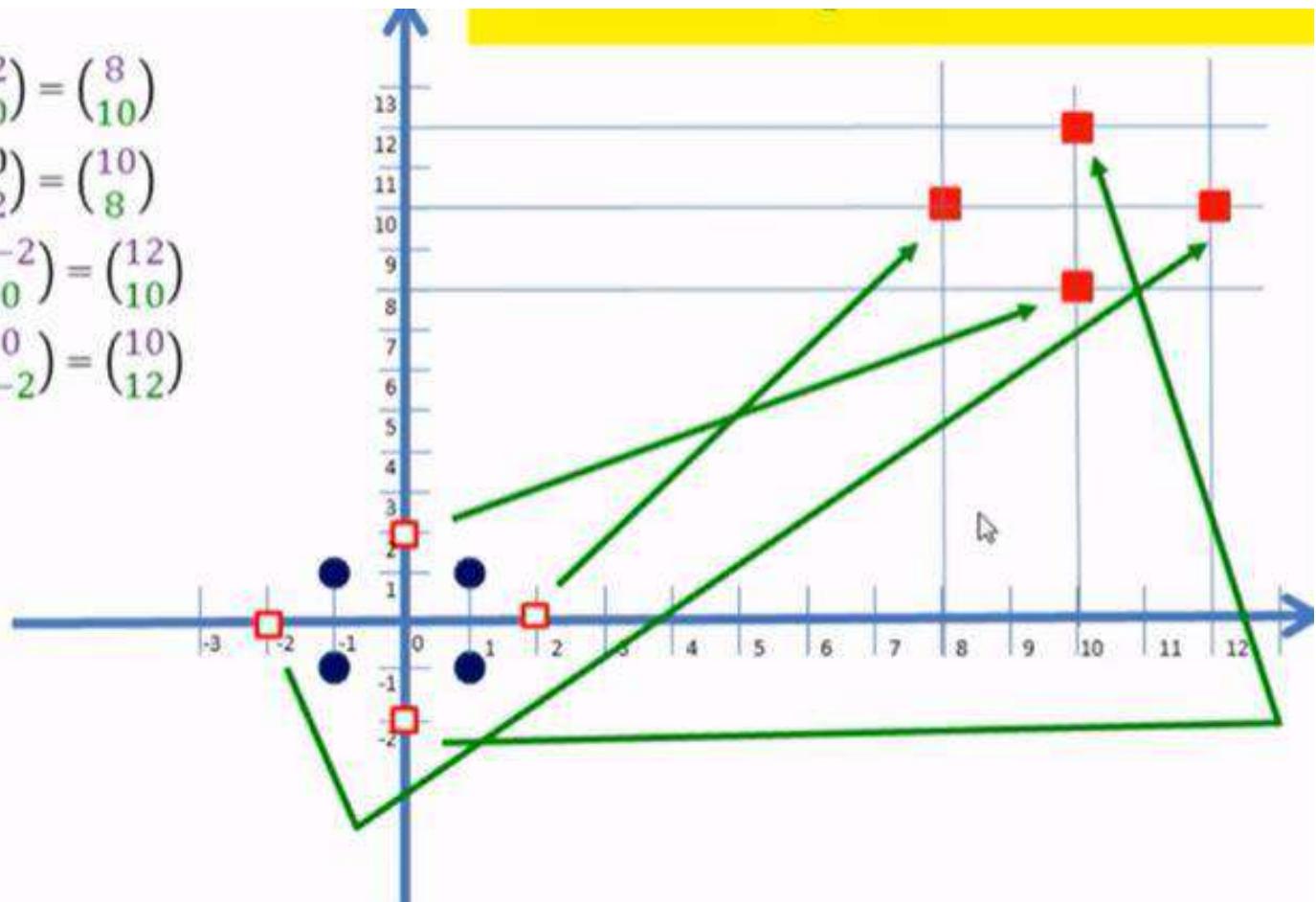
Example-1

- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 6 - x_1 + (x_1 - x_2)^2 \\ 6 - x_2 + (x_1 - x_2)^2 \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} \geq 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$
- Let us take Red class vectors : $\begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -2 \end{pmatrix}$

- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 6 - 2 + (2 - 0)^2 \\ 6 - 0 + (2 - 0)^2 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$
- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 - 0 + (0 - 2)^2 \\ 6 - 2 + (0 - 2)^2 \end{pmatrix} = \begin{pmatrix} 10 \\ 8 \end{pmatrix}$$
- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} -2 \\ 0 \end{pmatrix} = \begin{pmatrix} 6 + 2 + (-2 - 0)^2 \\ 6 - 0 + (-2 - 0)^2 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \end{pmatrix}$$
- $$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 6 - 0 + (0 + 2)^2 \\ 6 + 2 + (0 + 2)^2 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix}$$

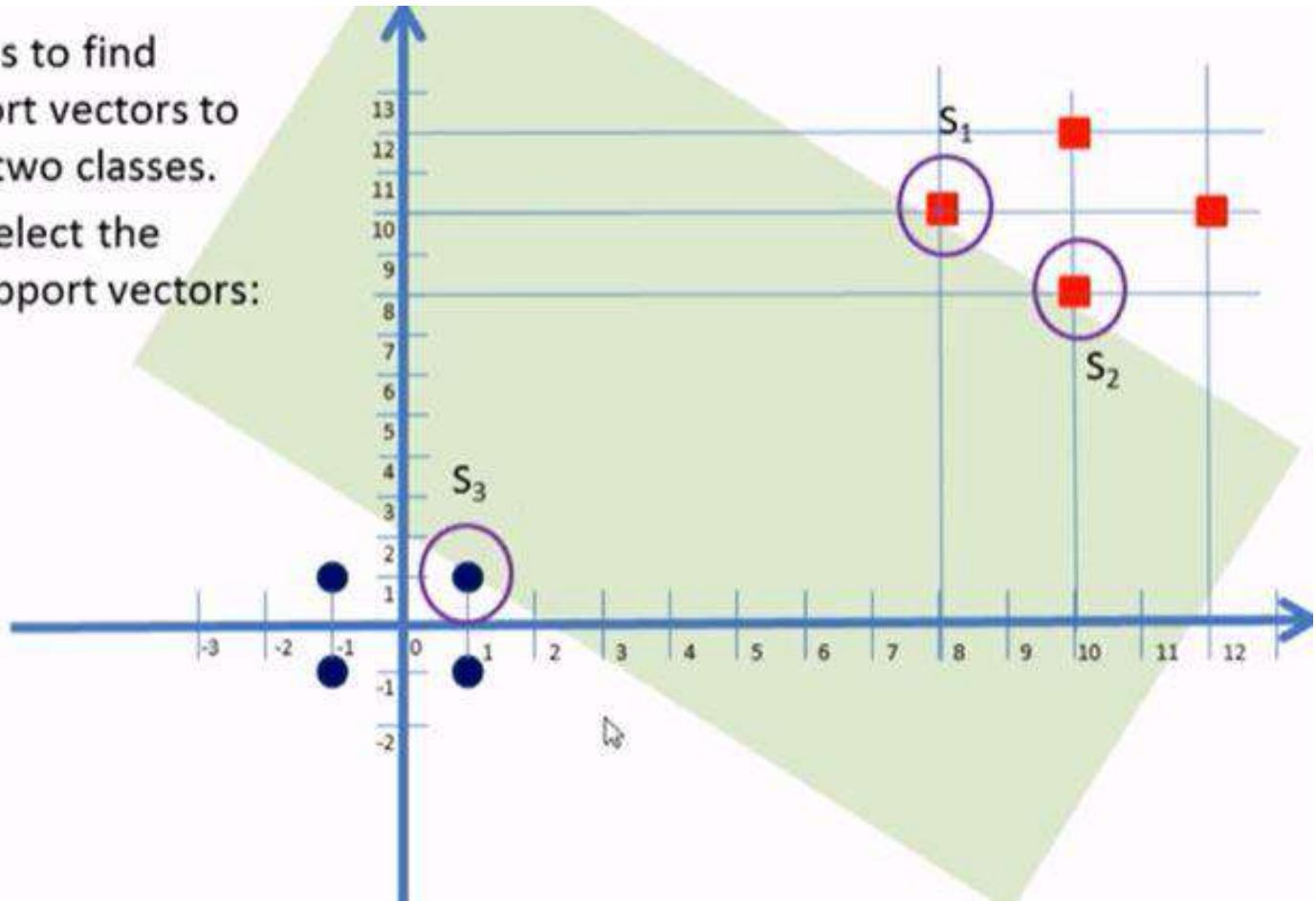
Example-1

- $\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$
- $\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 10 \\ 8 \end{pmatrix}$
- $\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} -2 \\ 0 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \end{pmatrix}$
- $\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Phi \begin{pmatrix} 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix}$



Example-1

- Now our task is to find suitable support vectors to classify these two classes.
- Here we will select the following 3 support vectors:
- $s_1 = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$,
- $s_2 = \begin{pmatrix} 10 \\ 8 \end{pmatrix}$,
- and $s_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$



Example-1

- Here we will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde. That is:

$$s_1 = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$

$$s_2 = \begin{pmatrix} 10 \\ 8 \end{pmatrix}$$

$$s_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\widetilde{s}_1 = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix}$$

$$\widetilde{s}_2 = \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix}$$

$$\widetilde{s}_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Example-1

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

- Now we need to find 3 parameters α_1, α_2 , and α_3 based on the following 3 linear equations:

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_1} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_1} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_1} = +1 \quad (+ve\ class)$$

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_2} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_2} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_2} = +1 \quad (+ve\ class)$$

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_3} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_3} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_3} = -1 \quad (-ve\ class)$$

- Let's substitute the values for S_1, S_2 and S_3 in the above equations.

$$\widetilde{S_1} = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} \quad \widetilde{S_2} = \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} \quad \widetilde{S_3} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} = +1$$

$$\alpha_1 \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} = +1$$

$$\alpha_1 \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1$$

Example-1

- After multiplication we get:

$$165 \alpha_1 + 161 \alpha_2 - 19 \alpha_3 = +1$$

$$161 \alpha_1 + 165 \alpha_2 - 19 \alpha_3 = +1$$

$$19 \alpha_1 + 19 \alpha_2 - 3 \alpha_3 = -1$$

- Simplifying the above 3 simultaneous equations we get: $\alpha_1 = \alpha_2 = 0.859$ and $\alpha_3 = +1.4219$.



Example-1

- The hyper plane that discriminates the positive class from the negative class is given by:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

- Substituting the values we get:

$$\begin{aligned}\tilde{\mathbf{w}} &= \alpha_1 \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} - \alpha_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ \tilde{\mathbf{w}} &= (0.0859) \cdot \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} + (0.0859) \cdot \begin{pmatrix} 10 \\ 8 \\ 1 \end{pmatrix} - (+1.4219) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1243 \\ 0.1243 \\ -1.2501 \end{pmatrix}\end{aligned}$$

Example-1

For SVMs, we used this eq for a line: $ax + cy + b = 0$ where $w = [a \ c]$

Thus $ax + b = -cy \rightarrow y = (-a/c)x + (-b/c)$

Thus y-intercept is $(-b/c)$

The decision boundary is perpendicular to w and it has slope $=(-a/c)$

- Our vectors are augmented with a bias.
- Hence we can equate the entry in \tilde{w} as the hyper plane with an offset b .
- Therefore the separating hyper plane equation

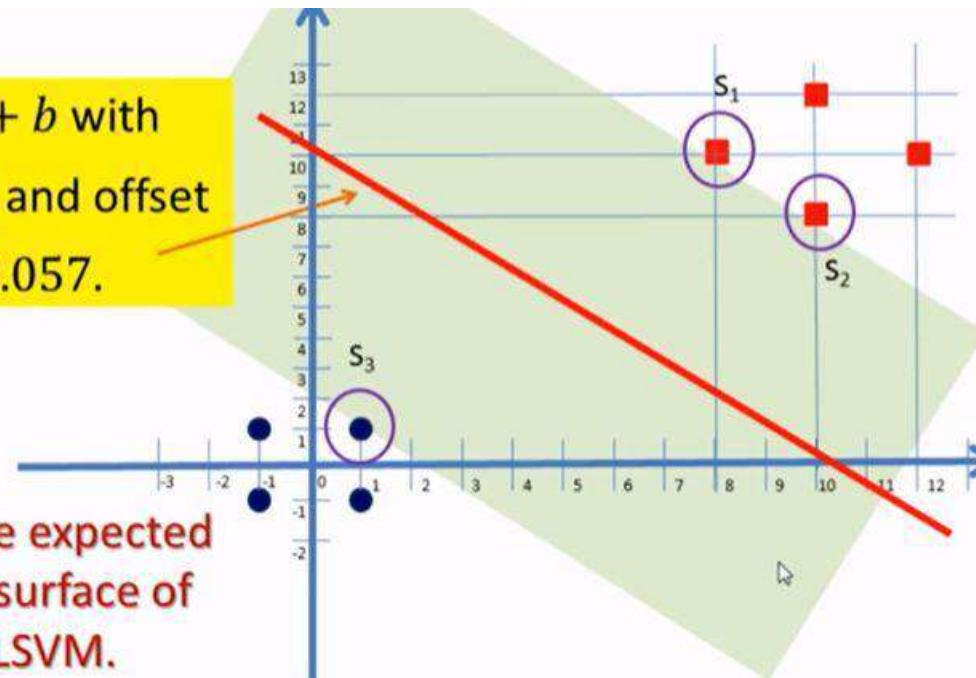
$$y = wx + b \text{ with } w = \begin{pmatrix} 0.1243/0.1243 \\ 0.1243/0.1243 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\text{and an offset } b = -\frac{1.2501}{0.1243} = -10.057. \quad \leftarrow \text{Y intercept}$$

Example-1

- $y = wx + b$ with
 $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and offset
 $b = -10.057.$

- This is the expected decision surface of the Non LSVM.



Example-1

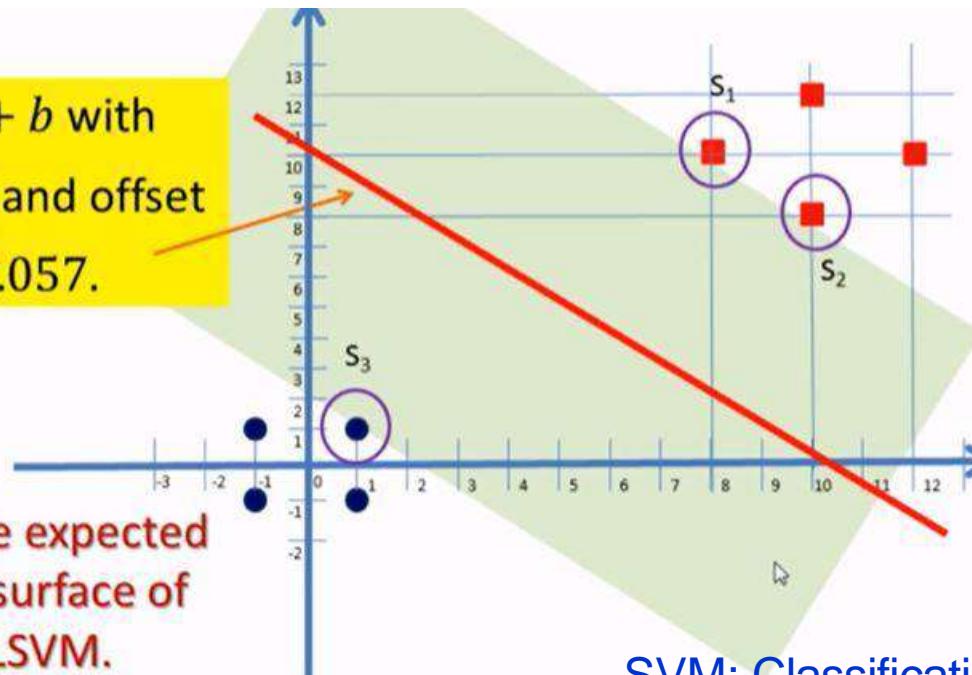
$$\text{New } X_t = [0 \ 1.5]^\top$$

$$\Phi(X_t) = [4.5 \ 4.5]$$

Prediction: $y=-1$
 $\text{sign}(-1.057)$

$$\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \left(6 - x_1 + (x_1 - x_2)^2\right) & \text{if } \sqrt{x_1^2 + x_2^2} \geq 2 \\ \left(\frac{x_1}{x_2}\right) & \text{otherwise} \end{cases}$$

- $y = wx + b$ with
 $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and offset
 $b = -10.057.$



- This is the expected decision surface of the Non LSVM.

SVM: Classification

$$f(z) = \text{sign}(w \cdot \Phi(z) + b) = \text{sign}\left(\sum_{i=1}^n \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b\right)$$

Non-linear SVM using kernel steps

1. Select a kernel function.
 2. Compute pairwise kernel values between labeled examples.
 3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
 4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.
-

SVM versus Logistic Regression

- When viewed from the point of view of regularized empirical loss minimization, SVM and logistic regression appear quite similar:

$$\text{SVM: } \sum_{i=1}^n \left(1 - y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1]\right)^+ + \|\mathbf{w}_1\|^2/2$$

$$\text{Logistic: } \sum_{i=1}^n \underbrace{-\log P(y_i|\mathbf{x}, \mathbf{w})}_{-\log \sigma(y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1])} + \|\mathbf{w}_1\|^2/2$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function.

(Note that we have transformed the problem maximizing the penalized log-likelihood into minimizing negative penalized log-likelihood.)

SVM versus Logistic Regression

- The difference comes from how we penalize “errors”:

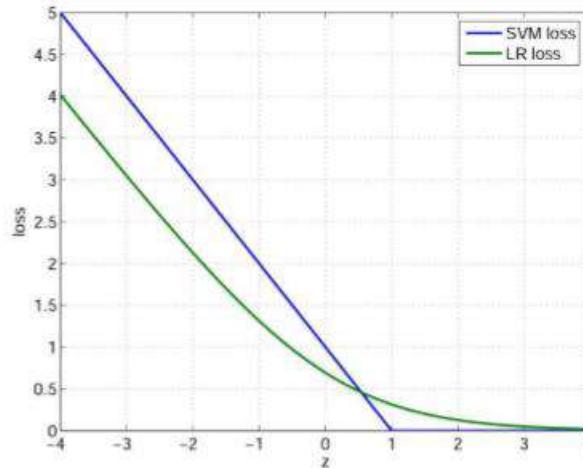
Both:
$$\sum_{i=1}^n \text{Loss}\left(\overbrace{y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1]}^z\right) + \|\mathbf{w}_1\|^2/2$$

- SVM:

$$\text{Loss}(z) = (1 - z)^+$$

- Regularized logistic reg:

$$\text{Loss}(z) = \log(1 + \exp(-z))$$



SVM versus Logistic Regression

- Logistic regression focuses on maximizing the probability of the data. The farther the data lies from the separating hyperplane (on the correct side), the happier LR is
- An SVM tries to find the separating hyperplane that maximizes the distance of the closest points to the margin (the support vectors). If a point is not a support vector, it doesn't really matter.

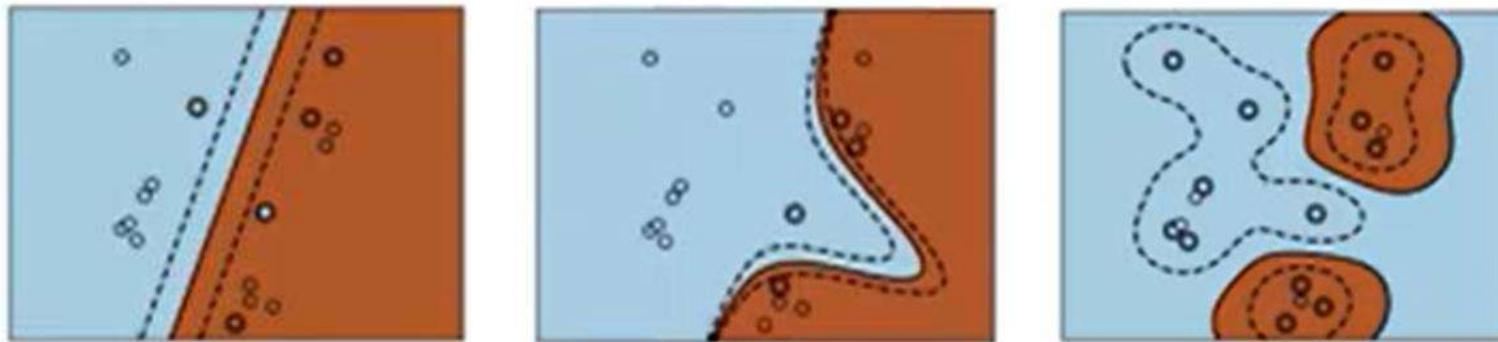
SVM versus Logistic Regression

- LR gives calibrated probabilities that can be interpreted as confidence in a decision.
- LR gives us an unconstrained, smooth objective
- LR can be (straightforwardly) used within Bayesian models.
- SVMs don't penalize examples for which the correct decision is made with sufficient confidence. This may be good for generalization.
- SVMs have a nice dual form, giving sparse solutions when using the kernel trick (better scalability).

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
 - Only support vectors are used to specify the separating hyperplane
 - Therefore SVM also called sparse kernel machine.
- Ability to handle large feature spaces
 - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution

SVM Kernels

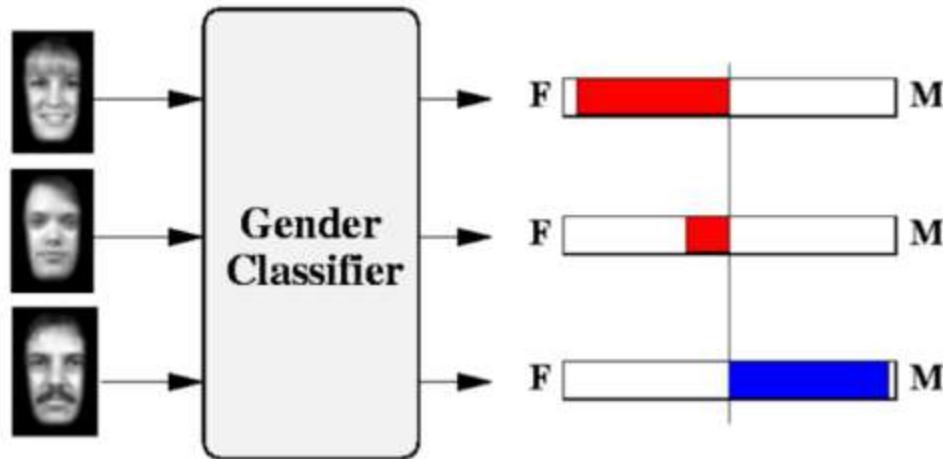


Name of Kernel Function	Definition
Linear	$K(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v}$
Polynomial of degree d	$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^T \mathbf{v} + 1)^d$
Gaussian Radial Basis Function (RBF)	$K(\mathbf{u}, \mathbf{v}) = e^{-\frac{1}{2}[(\mathbf{u}-\mathbf{v})^T \Sigma^{-1} (\mathbf{u}-\mathbf{v})]}$
Sigmoid	$K(\mathbf{u}, \mathbf{v}) = \tanh[\mathbf{u}^T \mathbf{v} + b]$

Linear Kernel	Large Data	Text
Polynomial Kernel	Normalized Data	Image Processing
Gaussian Kernel	EDA not clear	Computing Power

SVM Application – Observations

Learning Gender from Images

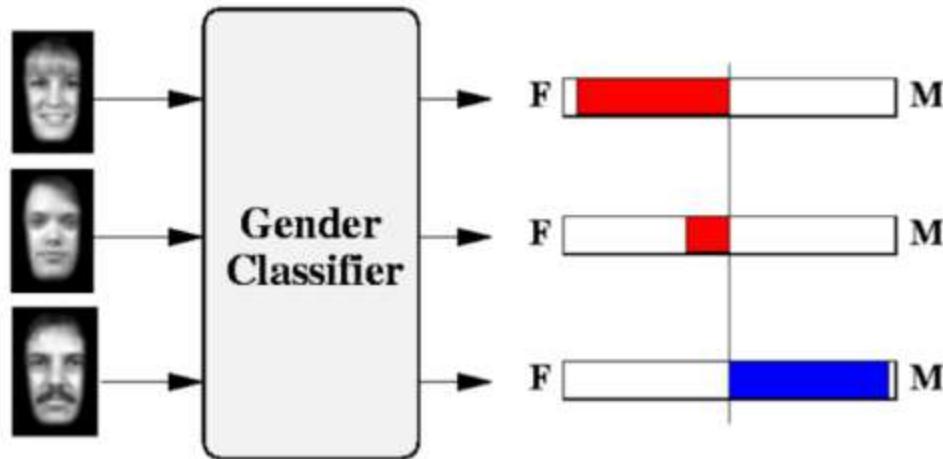


Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

Moghaddam and Yang, Face & Gesture 2000

SVM Application – Observations

Learning Gender from Images



Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

Moghaddam and Yang, Face & Gesture 2000

SVM Application – Observations

Image Analysis

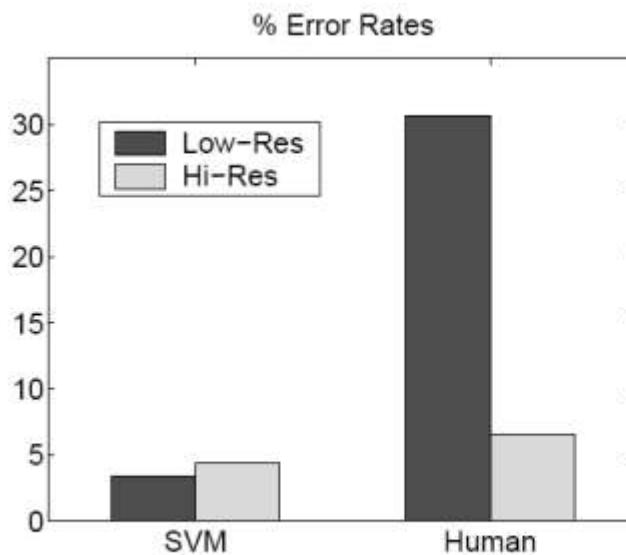


Figure 6. SVM vs. Human performance

- SVMs performed better than humans, at either resolution

SVMs: Pros and cons

- Pros
 - Kernel-based framework is very powerful, flexible
 - Often a sparse set of support vectors – compact at test time
 - Work very well in practice, even with very small training sample sizes
 - Solution can be formulated as a quadratic program
- Cons
 - Can be tricky to select best kernel function for a problem
 - Computation, memory
 - At training time, must compute kernel values for all example pairs
 - Learning can take a very long time for large-scale problems

Adapted from Lana Lazebnik



Summary

SVM Points to remember

- Primal and Dual optimization problems
 - Kernel functions
 - Support Vector Machines
 - Maximizing margin
 - Derivation of SVM formulation
 - Slack variables and hinge loss
 - Relationship between SVMs and logistic regression
 - Hinge loss
 - Log loss
-

Hard Margin versus Soft Margin

- **Hard Margin:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- **Soft Margin incorporating slack variables:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

- **Corresponding Lagrangian function**

$$L = \frac{1}{2} \|\vec{w}\|^2 + C \sum_i \xi_i + \sum_i \lambda_i (y_i (\vec{w} \cdot \vec{x}_i + b) - 1 + \xi_i)$$

Soft Margin Vs Hard margin Classification – Solution

Find $\alpha_1, \dots, \alpha_N$ such that

The dual problem

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

Solution to the dual problem

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$\mathbf{b} = y_k(1 - \xi_k) - \mathbf{w}^T \mathbf{x}_k \text{ where } k = \operatorname{argmax} \alpha_k$$

Find $\alpha_1, \dots, \alpha_N$ such that

The dual problem

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j)$ is maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) \alpha_i \geq 0 \text{ for all } \alpha_i$$

Solution to the dual problem

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$\mathbf{b} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b$$

Reference

- Support Vector Machine Classification of Microarray Gene Expression Data, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- Text categorization with Support Vector Machines:
learning with many relevant features T. Joachims, ECML - 98
- Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
- A Tutorial on Support Vector Machines for Pattern Recognition, Kluwer Academic Publishers - Christopher J.C. Burges

Extra References for SVM

- <http://www.cs.utexas.edu/users/mooney/cs391L/>
 - <https://www.coursera.org/learn/machine-learning/home/week/7>
 - <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
 - <https://data-flair.training/blogs/svm-kernel-functions/>
 - [MIT 6.034 Artificial Intelligence, Fall 2010](#)
 - <https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior>
 - <https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796>
 - <https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be>
 - [Radial basis kernel](#)
 - <http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf>
-



Machine Learning

DSE CLZG565

Unsupervised Learning

Raja vadhana P
Assistant Professor,
BITS - CSIS



BITS Pilani
Pilani Campus

Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
 - I here by acknowledge all the contributors for their material and inputs.
 - I have provided source information wherever necessary
 - I have added and modified the content to suit the requirements of the course
- Source:** Slides of Prof. Chetana, Prof.Vimal, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes, Tom Mitchell, Andrew Ng and many others who made their course materials freely available online.

Course Plan

M1 & M2 Introduction & Mathematical Preliminaries

M6 Linear Models for Regression

M5 Linear Models for Classification

M3 & M4 Bayesian Learning & Bayesian Classifiers

M7 Decision Tree

M8 Neural Networks

M9 Instance Based Learning

M10 Ensemble

M11 & M12 Support Vector Machine

M13 Unsupervised Learning

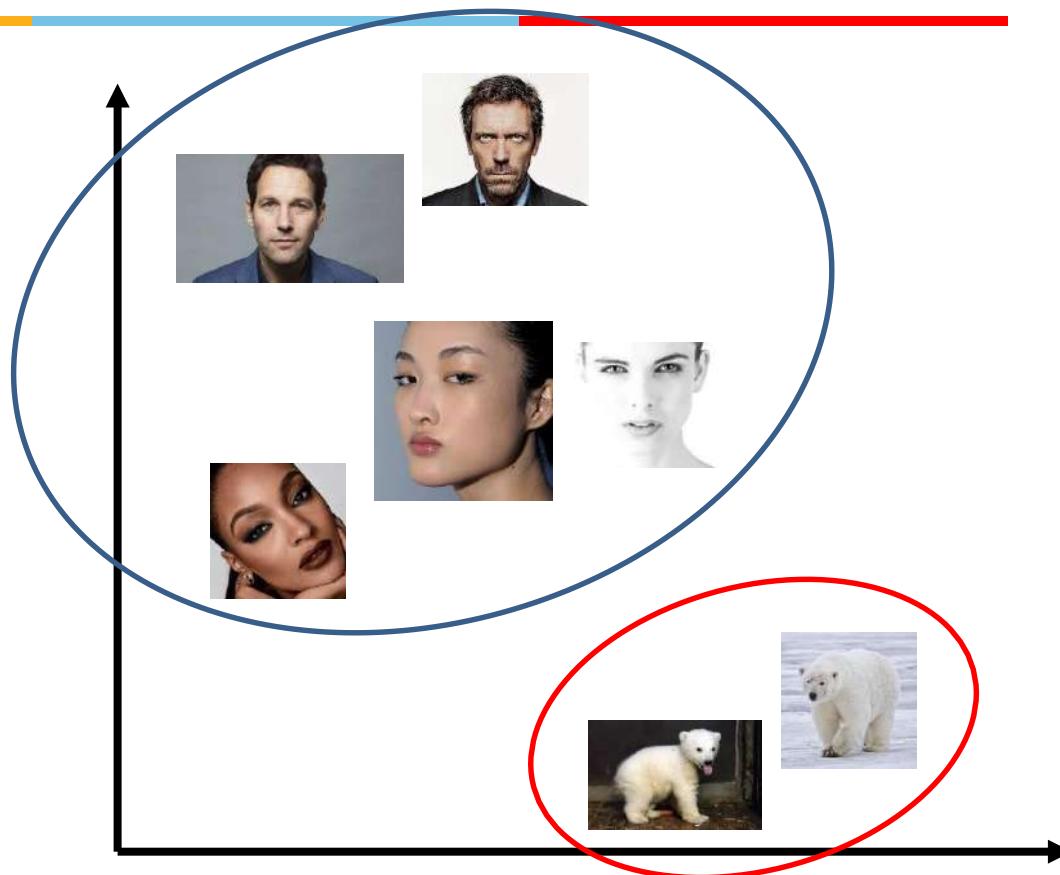
Module – 13 – Unsupervised Learning

- Mixture Models
- Expectation Maximization (EM) Algorithm
- K-means Clustering

Unsupervised Learning

- We only use the features X, not the labels Y
- This is useful because we may not have any labels but we can still detect patterns
- For example:
 - We can detect that news articles revolve around certain topics, and group them accordingly
 - Discover a distinct set of objects appear in a given environment, even if we don't know their names, then ask humans to label each group
 - Identify health factors that correlate with a disease
- Clustering: Grouping items that “belong together” (i.e. have similar features)

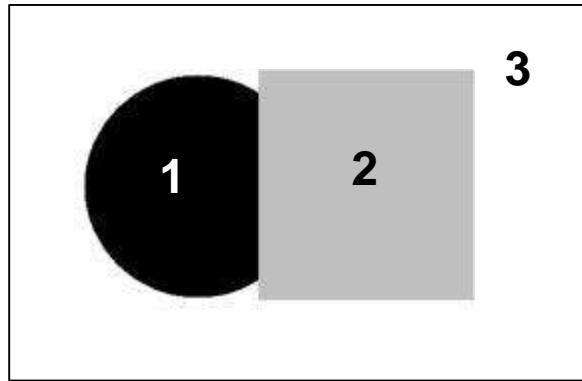
Unsupervised discovery



Clustering algorithms

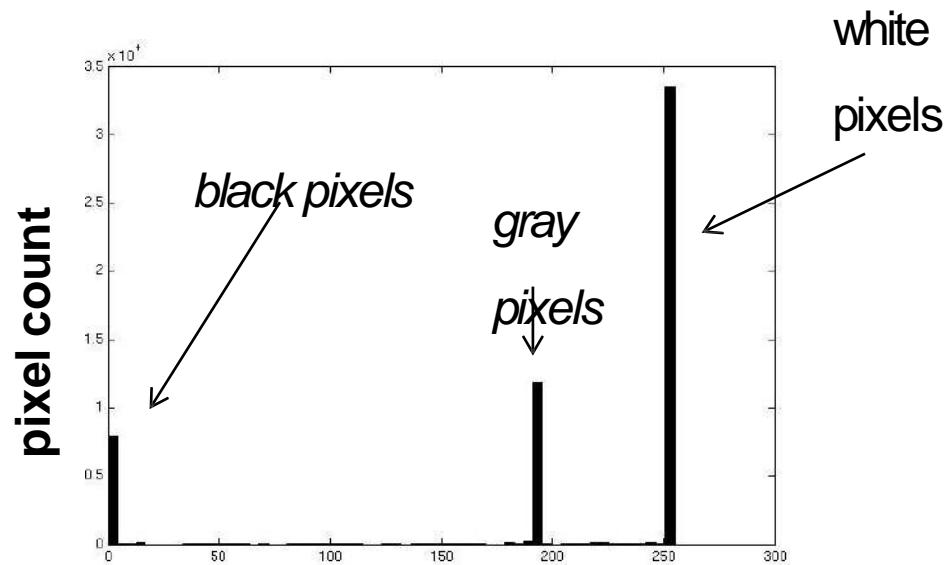
- In depth
 - K-means (iterate between finding centers and assigning points)
 - Gaussian Mixture Models (GMMs)

Image segmentation: toy example



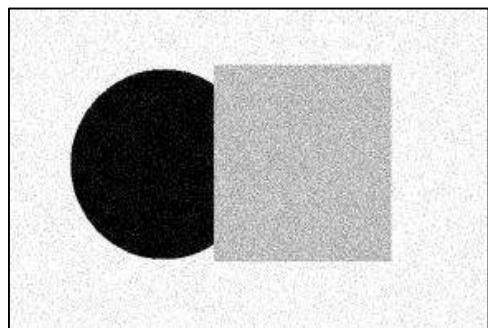
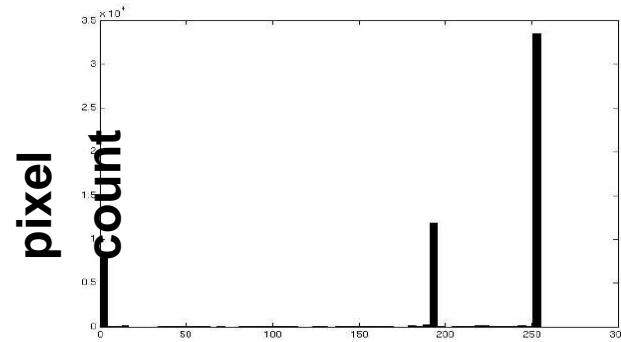
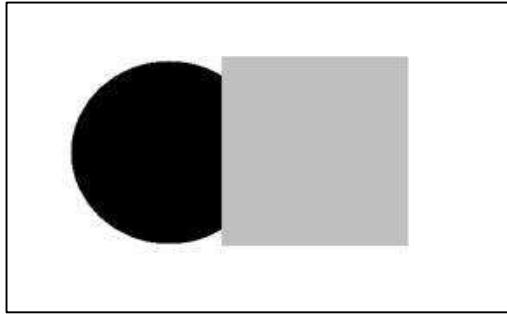
input image

- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
 - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

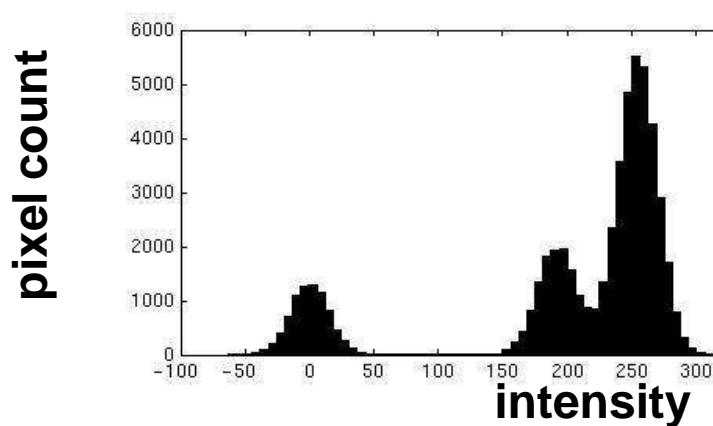


Source: K. Grauman

Image segmentation: toy example



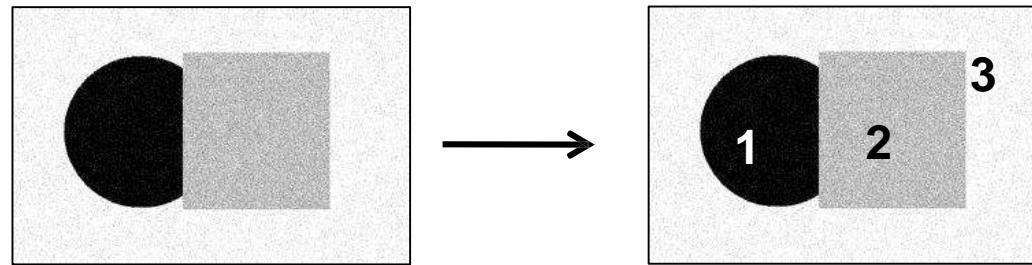
input image



Source: K. Grauman

- Now how to determine the three main intensities that define our groups?
- We need to **cluster**.

Image segmentation: toy example



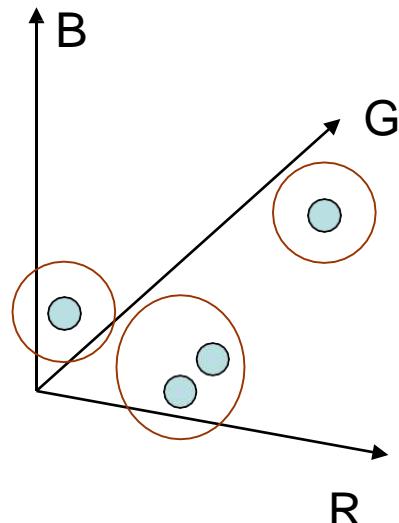
- **Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.**
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i : $SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$

Source: K. Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



Feature space: color value (3-d)

Source: K. Grauman

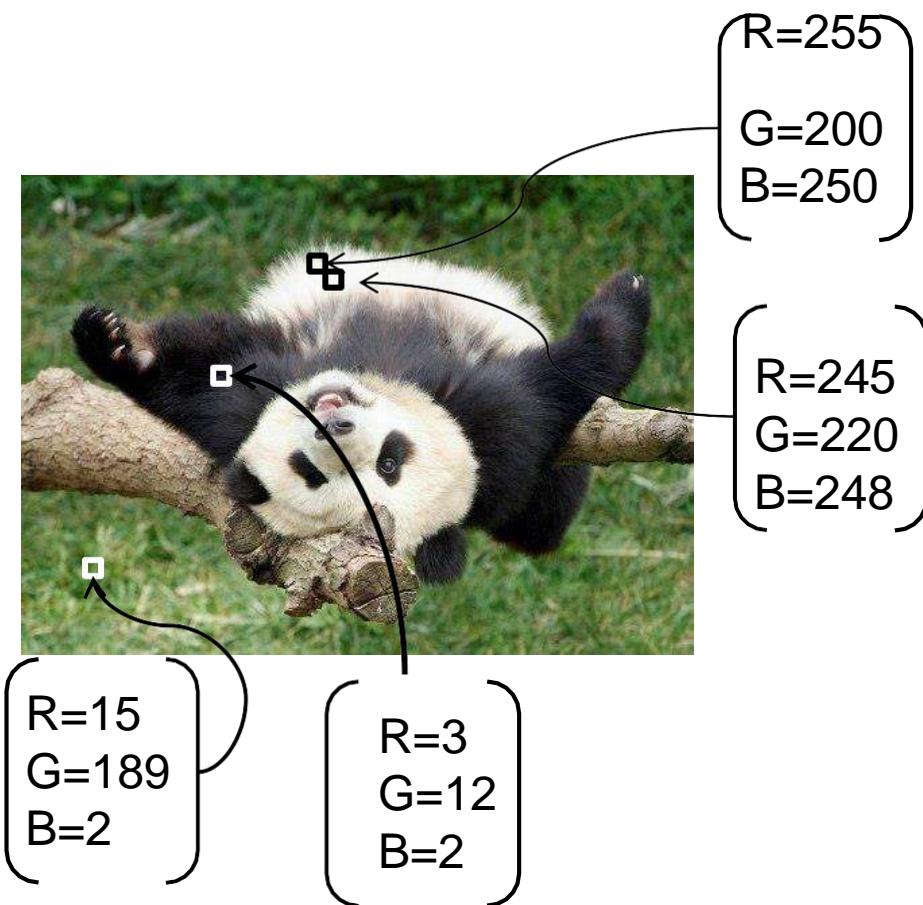
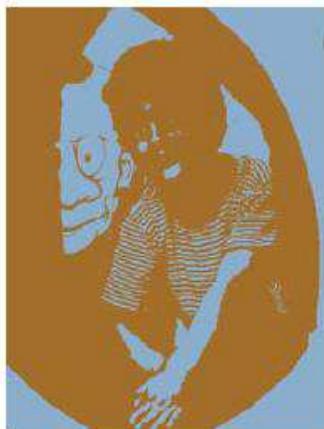


Image Compression using Segmentation

$K = 2$



$K = 3$



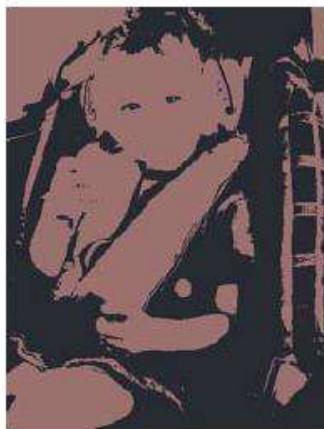
$K = 10$



Original image



Smaller values of K give higher compression at the expense of poorer image quality.

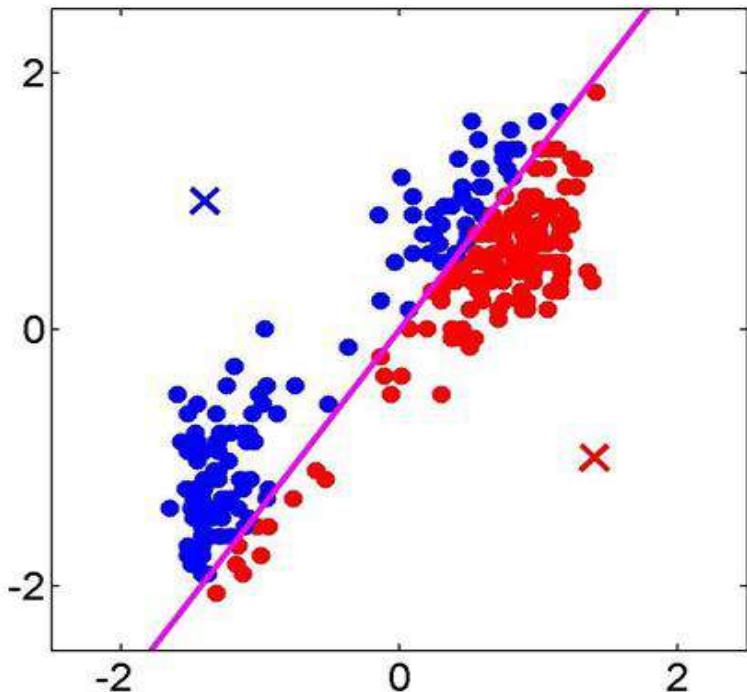




Expectation Maximization Algorithm

Iterative Step 1

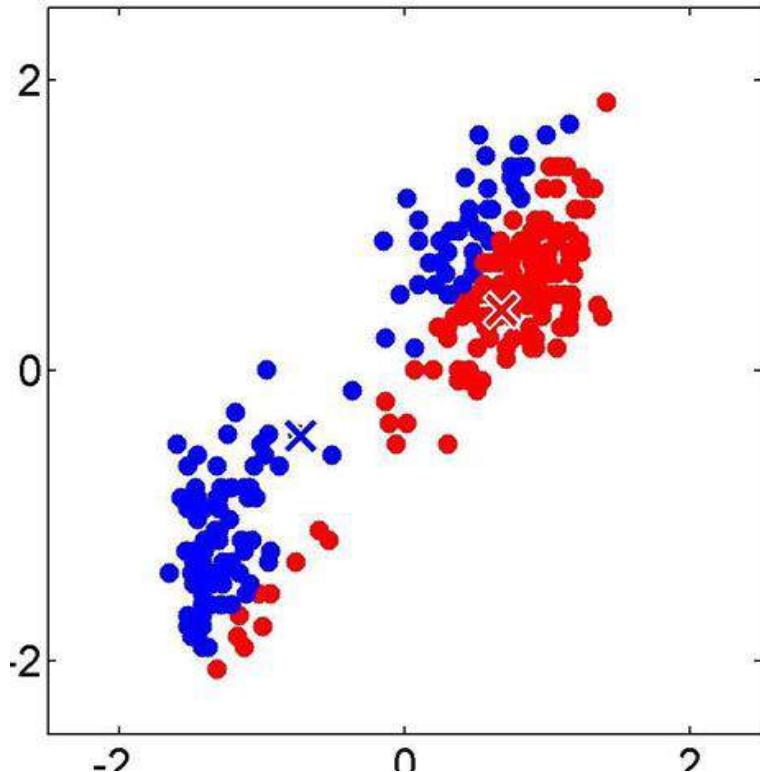
Assign data points to closest cluster center



- In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster center is nearer.
- This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centers, shown by the magenta line, they lie on.

Iterative Step 2

Change the cluster center to the average of the assigned points

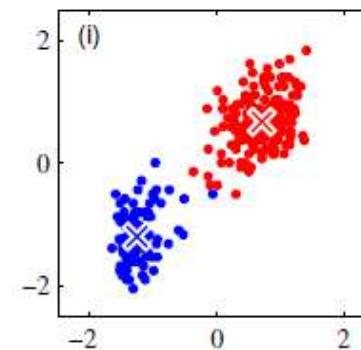
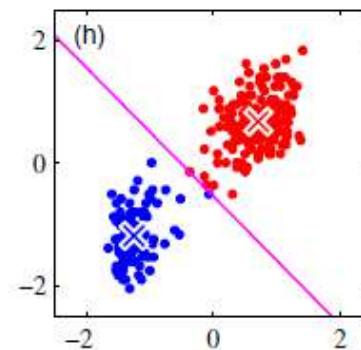
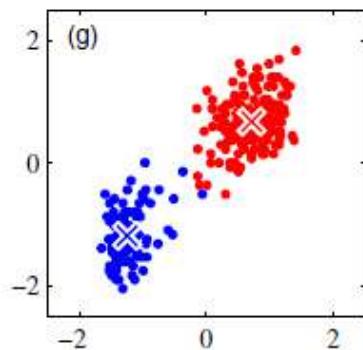
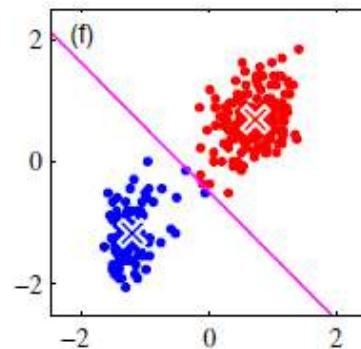
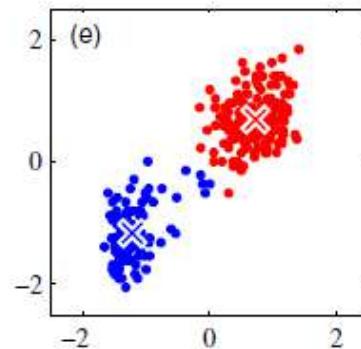
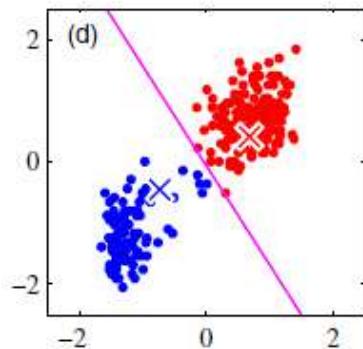


- In the M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster.

Repeat until convergence



Repeat to alternate E and M steps through to final convergence of the algorithm.





k-Means Clustering

Exercise

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ip} - x_{jp})^2}$$

Use the k-means algorithm and Euclidean distance to cluster the following 5 examples into 2 clusters:

A(0, 1) B(3, 0) C(2, 4) D(2, 1) E(3, 5)

Ex: Distance from
A to C
 $=\sqrt{(2-0)^2 + (4-1)^2} = 3.61$

Step 1 : Compute the Euclidean distance matrix

	A	B	C	D	E
A	0	3.16	3.61	2	5
B		0	4.12	1.414	5
C			0	3	1.414
D				0	4.12
E					0

Step 2 : Randomly choose two cluster centroids.

Let A and C be the cluster centroids.

Step 3 : Cluster 1: { A, B, D }

Cluster 2: { C, E }

k-Means Clustering Exercise

Step 4 : Compute the Centroids,

Cluster1 = mean of A,B,D

$$= \frac{0+3+2}{3}, \frac{1+0+1}{3} = (1.66, 0.66)$$

Cluster 2 = mean of C,E = $\frac{2+3}{2}, \frac{4+5}{2} = (2.5, 4.5)$

Step 5 : Repeat; Compute the Euclidean distance matrix to the cluster centers.

	A	B	C	D	E	C1	C2
A	0	3.16	3.61	2	5		
B		0	4.12	1.414	5		
C			0	3	1.414		
D				0	4.12		
E					0		
C1	1.69	1.49	3.35	0.48	4.54	0	
C2	4.30	4.52	0.70	3.53	0.70	3.93	0

k-Means Clustering Exercise

Step 6 : Cluster 1: { A, B, D }

Cluster 2: { C, E}

Step 7 : Recompute the Centroids,

Cluster1 = mean of A,B,D

$$= \frac{0+3+2}{3}, \frac{1+0+1}{3} = (1.66, 0.66)$$

$$\text{Cluster 2} = \text{mean of } C,E = \frac{2+3}{2}, \frac{4+5}{2} = (2.5, 4.5)$$

Step 8 : Stop as converged or no change is seen.

1 of K coding mechanism

- For each data point x_n , we introduce a set of binary indicator variables $r_{nk} \in \{0,1\}$ such that
- $\sum_k r_{nk} = 1$

where $k = 1, \dots, K$ describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then $r_{nk} = 1$, and $r_{nj} = 0$ for j not equal to k .

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

K-means Cost Function

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \| \mathbf{x}_n - \boldsymbol{\mu}_k \|^2$$

↑ data
 ↑ responsibilities ↑ prototypes

- Goal is to find values for the $\{r_{nk}\}$ and the $\{\boldsymbol{\mu}_k\}$ so as to minimize J (*Distortion measure*)
- relatively slow, because in each E step it is necessary to compute the Euclidean distance between every prototype vector and every data point

Minimizing the Cost Function

- E-step: minimize J w.r.t r_{nk}

choose some initial values for the μ_k . minimize J with respect to the r_{nk} , keeping the μ_k fixed. Assign the n th data point to the closest cluster centre

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

- M-step: minimize J w.r.t μ_k

minimize J with respect to the μ_k , keeping r_{nk} fixed. The objective function J is a quadratic function of μ_k , and it can be minimized by setting its derivative with respect to μ_k to zero giving

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}.$$

K-means Algorithm

- **Goal:** Represent a data set in terms of K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
 - **E-step:** assign each data point to nearest prototype
 - **M-step:** update prototypes to be the cluster means
- Simplest version is based on Euclidean distance

CGPA	Technical Interview	HR Discussion
5	10	9
6	10	9
6	8	8
7	8	8



K Means Clustering

I Standardize the data if required:

$$\sqrt{(5 - 6)^2 + (10 - 7)^2 + (9 - 7)^2}$$

II Fix the no.of.cluster expected

III Initialize the prototypes:

IV Expectation-Step: Fix prototype & find the membership for which the distortion is minimum

V Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototype

VI Repeat E & M Step till convergence is achieved:

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations are reached

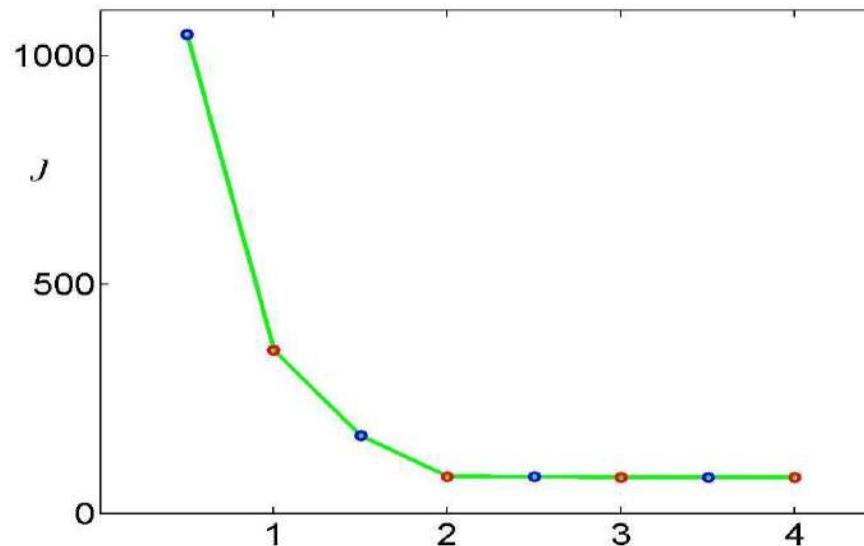
K=2	C1=(5,9,9)	C2=(6,7,7)
D(p1,Ci)	1	3.87
D(p2,Ci)	1.41	3.61
D(p3,Ci)	1.73	1.41
D(p4,Ci)	2.45	1.73
Membership	Cluster 1 = {p1, p2}	Cluster 2 = {p3,p4}

New Centroid
Take mean of cluster points.
This is used in next iteration

$$C1 = (5.5, 10, 9) \quad C2 = (6.5, 8, 8)$$

Convergence

- Each E step (blue points) and M step (red points). The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.
- Because each phase reduces the value of the objective function J , convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of J



Another way of writing objective function

K-means:

- The center of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).
- Uses squared Euclidean distance as the measure of dissimilarity between a data point and a prototype vector
- Not suitable where some or all of the variables represent categorical labels

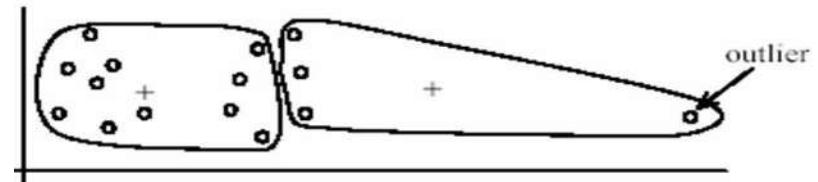
$$\tilde{J} = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

K-medoids (more general distances):

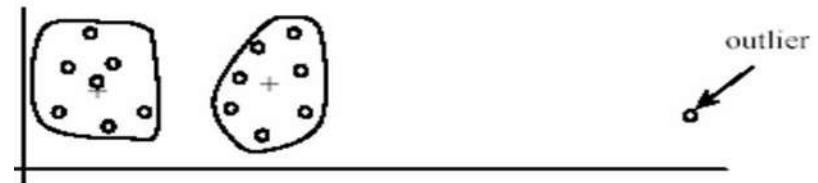
- It is common to restrict each cluster prototype to be equal to one of the data vectors assigned to that cluster and can be used with **arbitrary distances**
- k-medoids more robust to noise and outliers as compared to k-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

Limitations of K-means

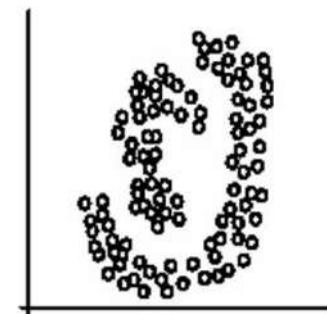
- Setting k?
- Sensitive to outliers



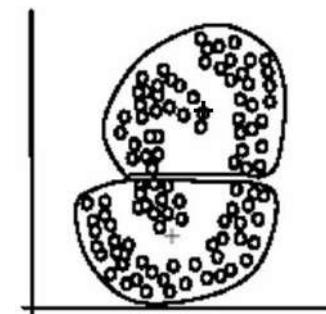
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters

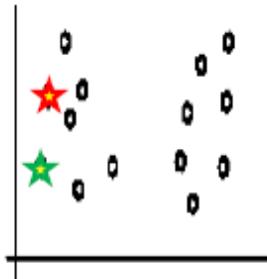


(B): k -means clusters

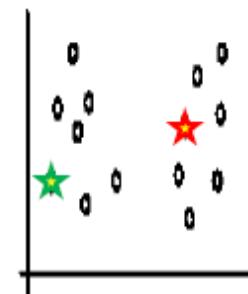
Source: K. Grauman

Limitations of K-means

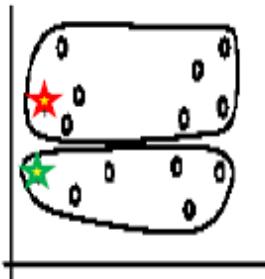
- Sensitive to initial centers
 - Use heuristics or output of another method



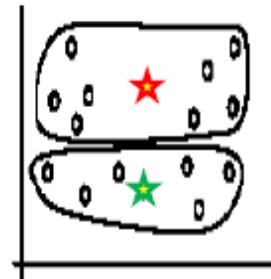
Random selection of seeds (centroids)



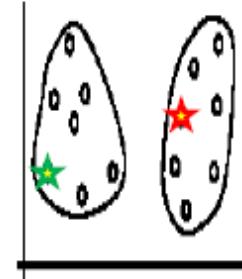
Random selection of seeds (centroids)



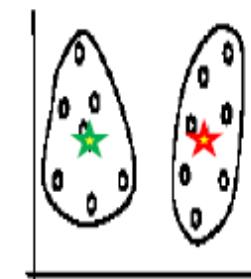
Iteration 1



Iteration 2



Iteration 1



Iteration 2

K Means – Notion of Outliers

Outliers are relatively away from the centroid

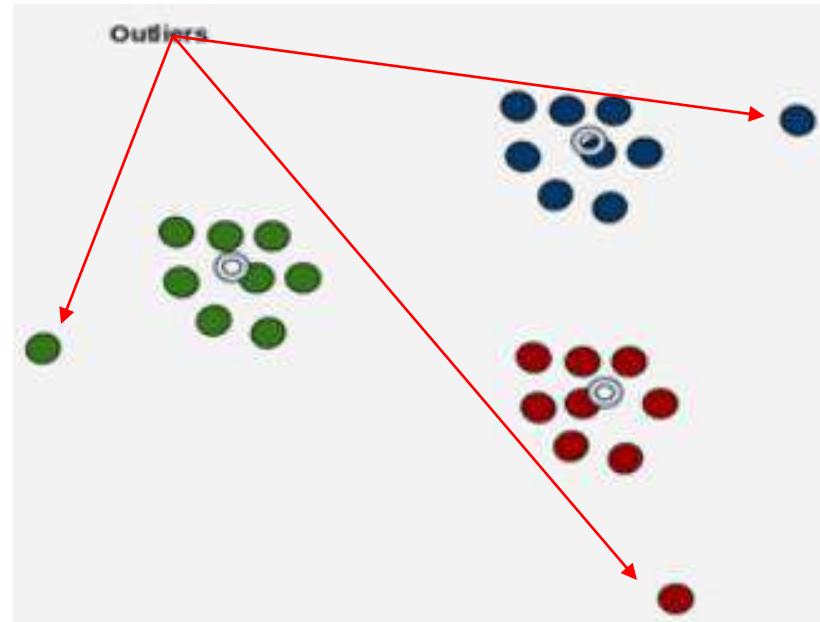
K-Means Clustered points:

$\text{dist}(x, \mu_k)$ be the distance of a point x , assigned to cluster k to its center μ_k .

L_{μ_k} be the average distance of all the points assigned to cluster k with its centre

The **ratio $\text{dist}(x, \mu_k) / L_{\mu_k}$** for each point is the outlier score for each point.

Higher the ratio for a point x , **more likely x is a outlier.**



K Means – Outlier Detection

Outliers are relatively away from the centroid

K-Means Clustered points:

$\text{dist}(x, \mu_k)$ be the distance of a point x , assigned to cluster k to its center μ_k .

$L_{\mu k}$ be the average distance of all the points assigned to cluster k with its centre

The **ratio $\text{dist}(x, \mu_k) / L_{\mu k}$** for each point is the outlier score for each point.

Higher the ratio for a point x , **more likely x is a outlier.**

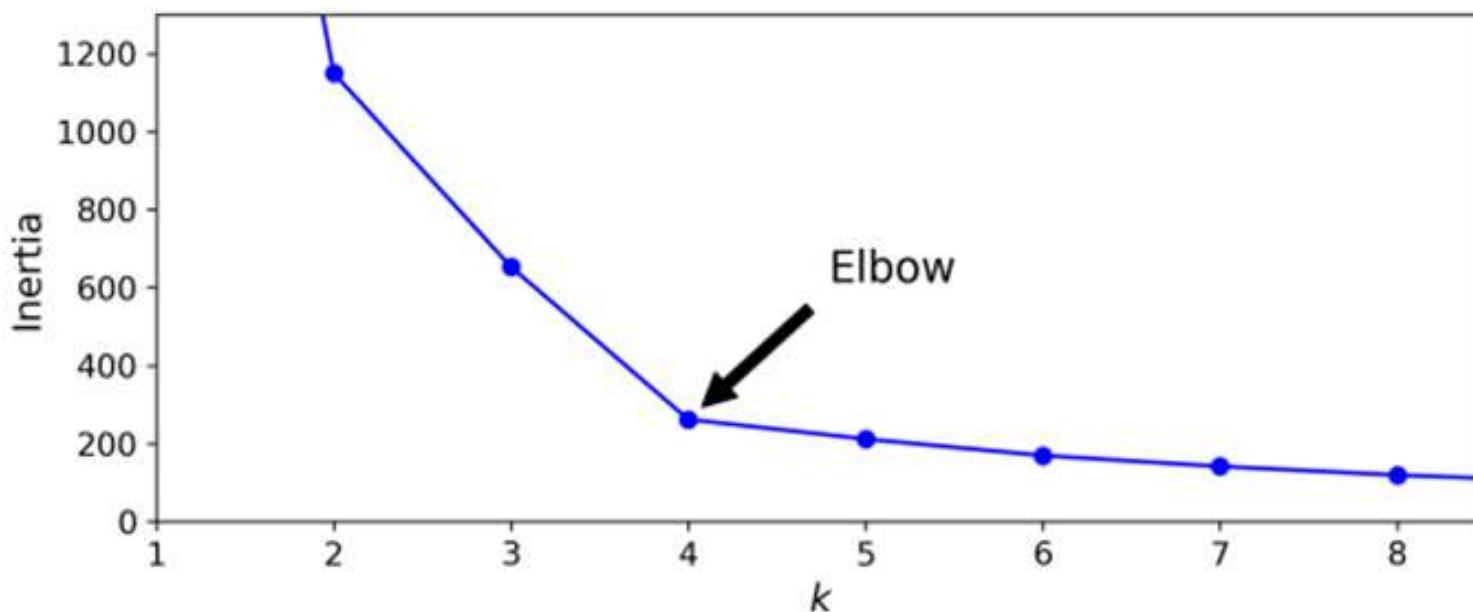
Candidate	Round 1 Rating	Round 2 Rating
A	0	1
B	3	0
C	2	4
D	2	1
E	3	5

Random centroid assigned at the first iteration of K-Means

K=2	$C1=(1.66, 0.66)$	$C2=(2.5, 4.5)$	$\text{dist}(x, \mu_k) / L_{\mu k}$
$D(A, Ci)$	1.7		1.39
$D(B, Ci)$	1.49		1.22
$D(C, Ci)$	0.47		0.38
$D(D, Ci)$		0.71	1
$D(E, Ci)$		0.71	1

Membership calculated for 1 st iteration are : →	Cluster 1 = {A,B,C,C1}	Cluster 2 = {D,E,C2}
$L_{\mu k}$	1.22	0.77

K Means –K Value

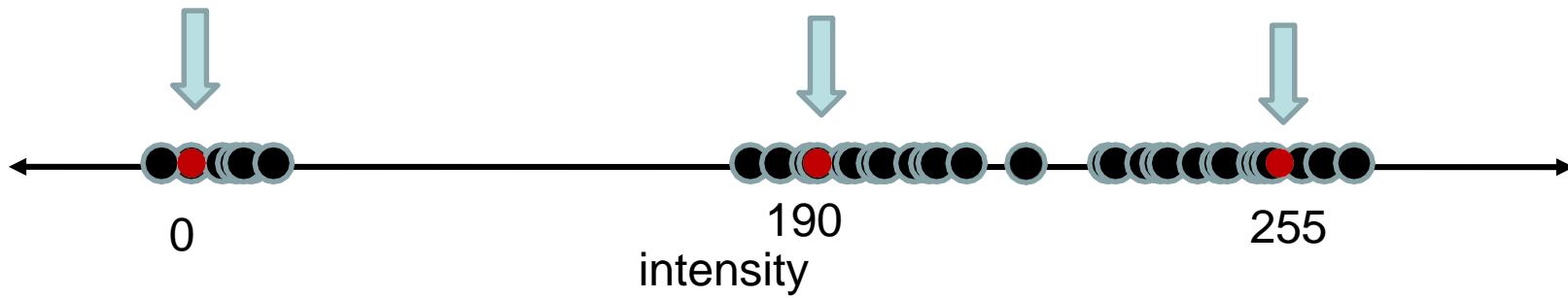




Notion of Mixture Models

K means – Hard clustering

- Assigned each example to exactly one cluster
- What if clusters are overlapping?
 - Hard to tell which cluster is right
 - Maybe we should try to remain uncertain
- What if cluster has a non-circular shape?
- Hard assignments of data points to clusters – small shift of a data point can flip it to a different cluster
- For example, if a point is near the 'border' between two clusters, it's often better to know that it has near equal membership probabilities for these clusters, rather than blindly assigning it to the nearest one.



1 of K coding mechanism

- For each data point x_n , we introduce a set of **binary indicator variables** $r_{nk} \in [0,1]$ such that
- $\sum_k r_{nk} = 1$

where $k = 1, \dots, K$ describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then $r_{nk} > r_{nj}$ for j not equal to k .

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

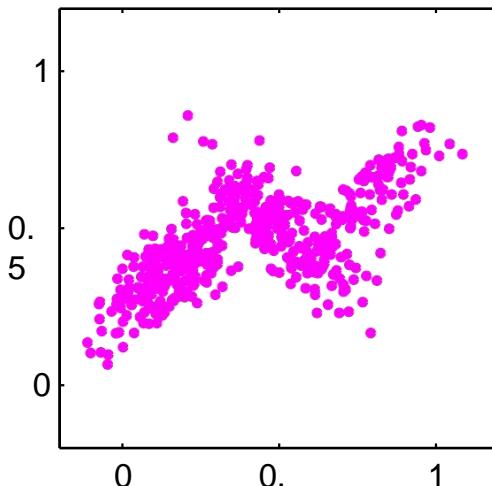
GMM - soft Clustering

- **Solution:** replace ‘hard’ clustering of K-means with ‘soft’ probabilistic assignments
- Probabilistic Clustering
 - Represents the probability distribution of the data as a **Gaussian mixture model (GMM)**
 - GMMs give a probabilistic assignment of points to clusters which quantify uncertainty.
 - Mixture models combines a set of distributions to create a convex space where we can search for the optimal parameters for such distributions using MLE.
- In GMM, Clusters modeled as Gaussians
- **EM algorithm:** assign data to cluster with some **probability**

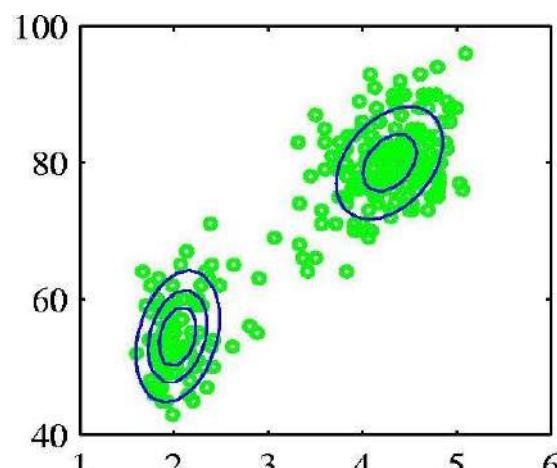
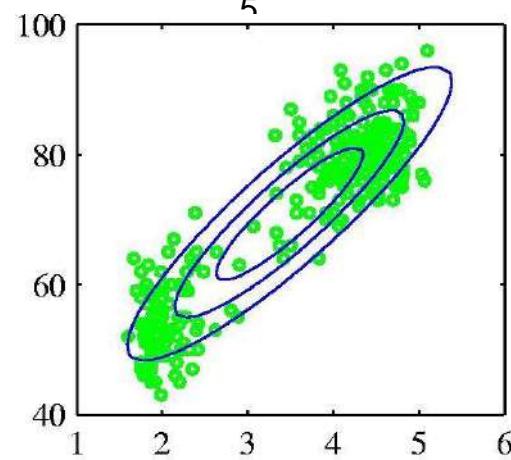
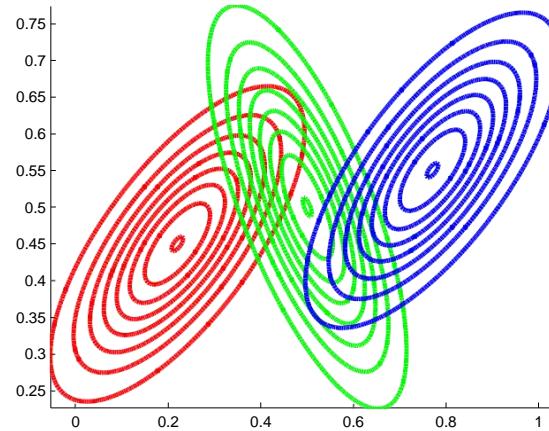
Source Credit: Christopher M. Bishop

Learning a Mixture of Gaussians

Our actual observations

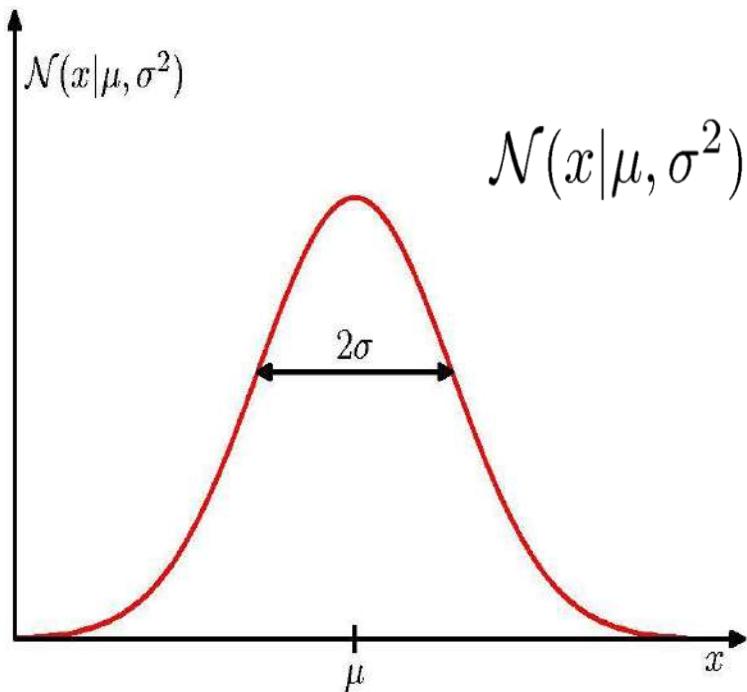


Mixture of 3 Gaussians



Source Credit :CS229: Machine Learning ©2021 Carlos Guestrin

Review: Gaussian Distribution

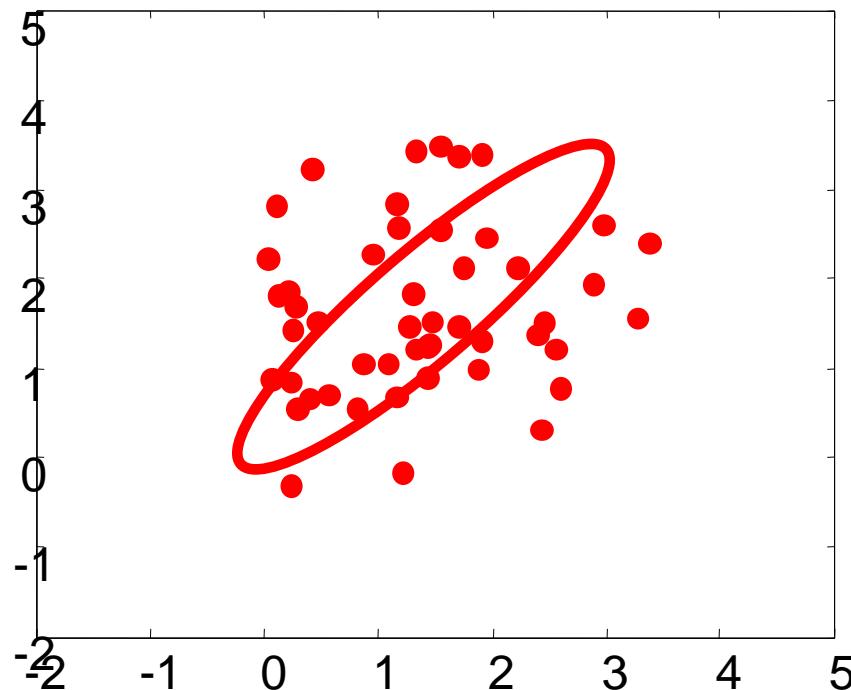


$$N(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\}$$

Source Credit : Chris Bishop

Multivariate Gaussian models

$$\mathcal{N}(\underline{x} ; \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) \right\}$$



Maximum Likelihood estimates

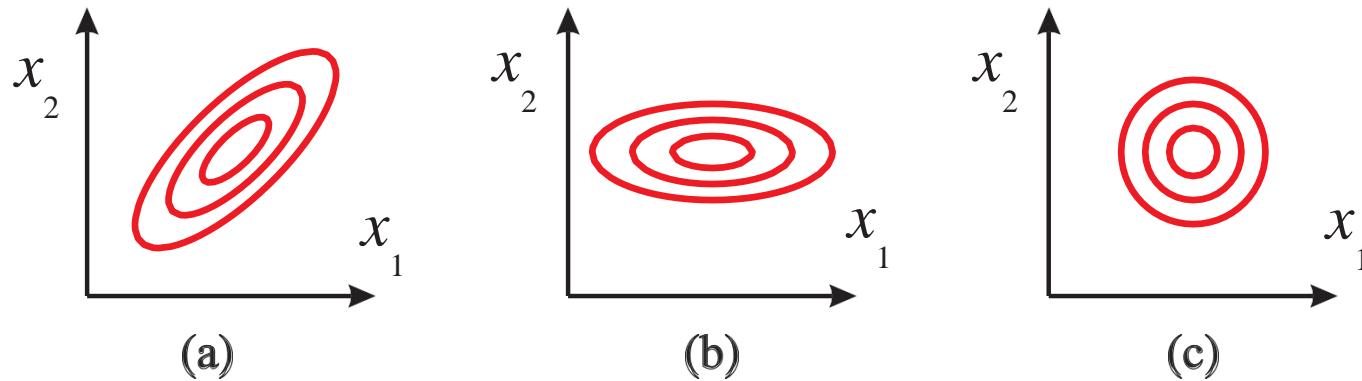
$$\hat{\mu} = \frac{1}{N} \sum_i x^{(i)}$$

$$\hat{\Sigma} = \frac{1}{N} \sum_i (x^{(i)} - \hat{\mu})^T (x^{(i)} - \hat{\mu})$$

We'll model each cluster using one of these Gaussian "bells"...

Example: Mixture of 3 Gaussians

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

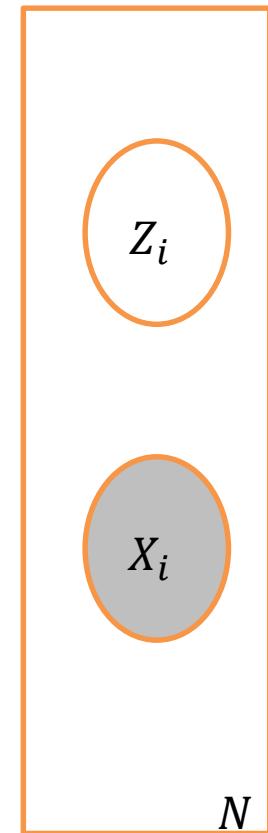


- The observed data come from a Gaussian mixture distribution which consists of K Gaussians with their own means and variances.
 - K classes are latent

GMM as Latent Variable model

- Each data point is associated with a latent variable(new binary random variable Z_i for each X_i) that indicates which cluster it belongs to.
- When fitting a GMM, we learn a distribution over these latent variables.
- This gives a probability that each data point is a member of each cluster. Given a data point x , what is the probability it came from Gaussian k
- Latent / hidden: not observed in the data

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$



GMM equations

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

Mixture models

K = number of mixture components (clusters),

π_j = mixture weights

$p_j(\mathbf{x})$ = family of distributions

(Gaussian, Poisson, Bernoulli, etc.).

$$p(\mathbf{x}) = \sum_{j=1}^K \pi_j p_j(\mathbf{x})$$

$$0 \leq \pi_j \leq 1, \quad \sum_{j=1}^K \pi_j = 1$$

Gaussian Mixture models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

$$0 \leq \pi_j \leq 1, \quad \sum_{j=1}^K \pi_j = 1$$

$\boldsymbol{\theta}$ = collection of all the parameters of the model
 (mixture weights, means, and covariance
 matrices):

$$\boldsymbol{\theta} := \{\pi_1, \dots, \pi_K, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\}$$

Parameters of GMM

X	Assuming Latent Variable z for 2 mixture components.	
	z(1)	z(2)
x(1)	0	1
x(2)	1	0
...		
x(N)	1	0

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

$$\pi : \{\pi_1, \dots, \pi_k\}$$

$$\mu : \{\mu_1, \dots, \mu_K\}$$

$$\Sigma : \{\Sigma_1, \dots, \Sigma_K\}$$

$$\boldsymbol{\theta} := \{\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K\}$$

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Mixture Density

GMM Model - π_k Mixing weights

- Marginal distribution over z is specified in terms of the mixing coefficients π_k , such that $p(z_k = 1) = \pi_k$
- $0 \leq \pi_k \leq 1$ and $\sum \pi_k = 1$
- K-dimensional binary random variable z having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0.

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x}|\mathbf{z})$$

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}$$

GMM Model- $p(\mathbf{x}|\mathbf{z})$ Component Density

- $p(\mathbf{x}|z)$: Conditional distribution of \mathbf{x} given a particular value for z is a Gaussian. Mixture densities

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

- z uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

GMM as Clustering algorithm

- For every observed data point x_i there is a corresponding latent variable z_i
- posterior probability of a given instance x_i belonging to component k , referred to as the 'responsibility' of component k for producing x_i , denoted as $\gamma(z_k) = p(z_k = 1 | x)$
- This gives us the probabilities of x_i belonging to the different components.
- That is precisely how a GMM can be used to cluster data.
- Use Bayes' theorem

GMM Model - the 'responsibility' of component k for producing \mathbf{x}_i

Using Bayes' theorem

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} | z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.\end{aligned}$$

π_k : prior probability of $z_k = 1$, i.e prior probability of picking the k th component

Maximum Likelihood

- Data set of observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and we wish to model this data using a mixture of Gaussians.
- Log of likelihood function:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Summation inside the logarithm →
 Complicated expressions for ML
 solution
 → Cannot get closed form solutions to
 ML parameters.

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Maximizing the log likelihood function

- Set $d/d\theta \{LL(\theta)\} = 0$ and solve for θ : **Non-linear, non-analytically solvable**
- Use gradient Descent: **Doable, but often slow**
- **Use EM**

Expectation maximization (EM)

- Expectation maximization (EM) is an iterative optimization approach that is used in many ways to find maximum likelihood estimates of parameters in probabilistic models in the presence of missing data.
- A generalization of MLE based on maximization of a posterior that data was generated by a model.
- EM is “simpler” than gradient methods
 - No need to choose step size.
- EM alternates between performing
 - an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed
 - a maximization (M) step, which computes the MLEs of the parameters by maximizing the expected likelihood found on the E step.
 - The parameters found on the M step are then used to begin another E step, and the process is repeated.

General EM algorithm

Observed data: $D = \{x_1, \dots, x_n\}$

Unknown variables: y

In clustering: $y=1 \dots K$ clusters

Parameters: θ

In GMM : $\theta = \{\pi_1, \dots, \pi_K\}$
 $\mu : \{\mu_1, \dots, \mu_K\}$
 $\Sigma : \{\Sigma_1, \dots, \Sigma_K\}$

Goal: $\hat{\theta}_n = \arg \max_{\theta} \log P(D|\theta)$

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

EM algorithm for GMM

- Start with parameters describing each cluster. Initialize the means μ_k , covariances Σ_k and mixing coefficients π_k , and evaluate the initial value of the log likelihood.
- E Step :
 - "partial membership" of each data point in each constituent distribution is computed
 - Compute "expected" classes (conditional probabilities of the latent variables) of all datapoints for each class
 - In K-means "E-step" we do hard assignment. EM does soft assignment

EM algorithm for GMM

E step:

Evaluate the responsibilities (posterior probabilities) using the current parameter

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} | z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.\end{aligned}$$

- For each example \mathbf{x}_n ,
- Compute $\gamma(z_{nk})$ the probability that \mathbf{x}_n is generated by component Z_k i.e it belongs to cluster k
- If \mathbf{x}_n is very likely under the k_{th} Gaussian, it gets high weight

EM algorithm for GMM

M-Step :

- maximize the expectation of the complete-data log-likelihood, computed with respect to the conditional probabilities found in the Expectation step. The result of the maximization is a new parameter vector μ_{new} , Σ_{new} and π_{new}
- Keep γ (z_{nk}) fixed, and apply MLE for maximizing of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ for μ_k , Σ_k and π_k , to get μ_{new} , Σ_{new} and π_{new}

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Repeat E & M until convergence

- In practice, the algorithm is deemed to have converged when the change in the log likelihood function, or alternatively in the parameters, falls below some threshold

EM algorithm for GMM

To Estimate:

M-Step

Initialize π, μ, Σ and
also evaluate the log likelihood

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$$N_k = \sum_{n=1}^N \gamma(z_{nk})$$

Perform E-Step Given $\gamma(z_k)$

E-Step

$$\begin{aligned} \gamma(z_k) &\equiv p(z_k = 1 | \mathbf{x}) = \\ &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \end{aligned}$$

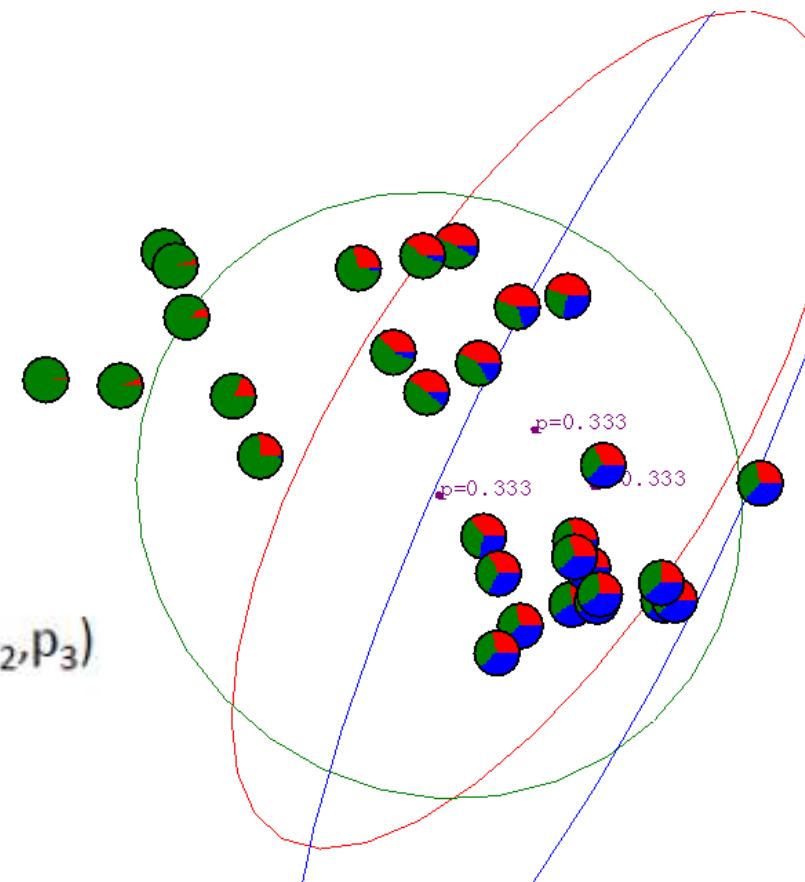
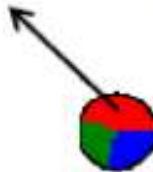
Perform M-Step Given π, μ, Σ

Repeat Until
Convergence

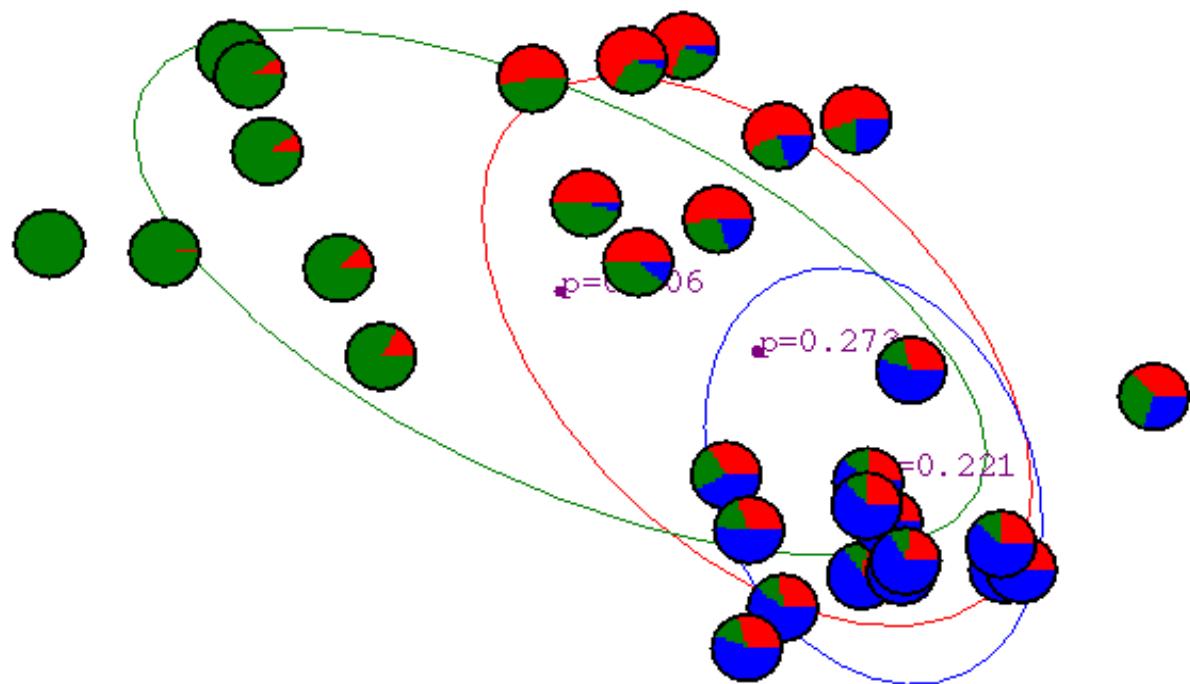
N_k = the effective number of points assigned to cluster k

Gaussian Mixture Example: Start

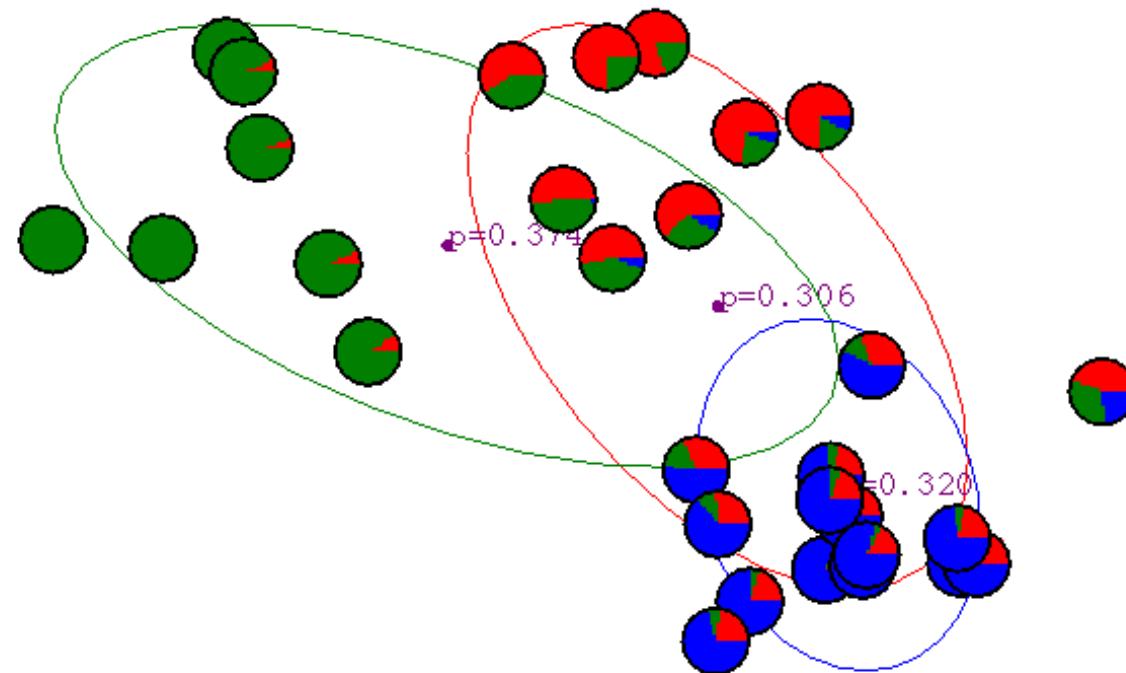
$$P(y = \bullet | x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$$



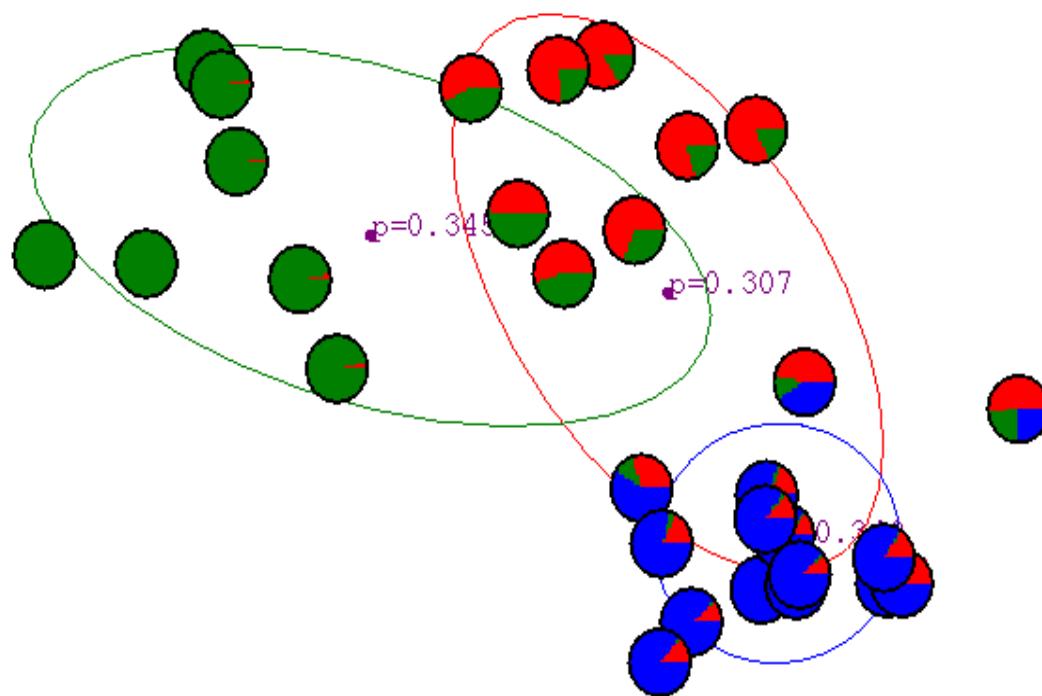
After first iteration



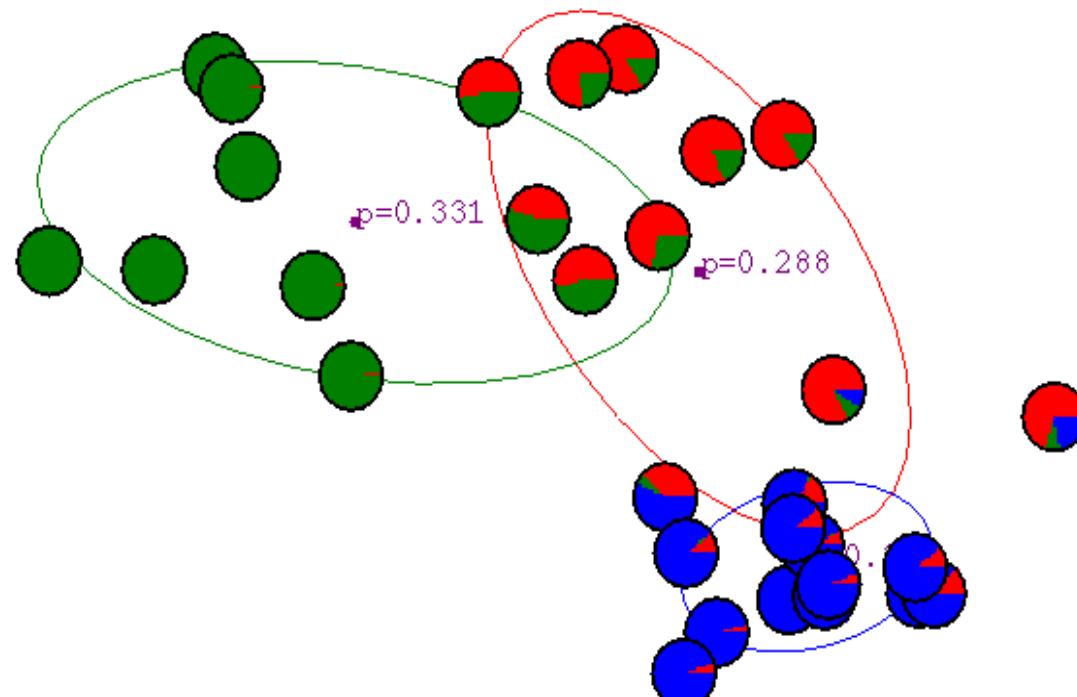
After 2nd iteration



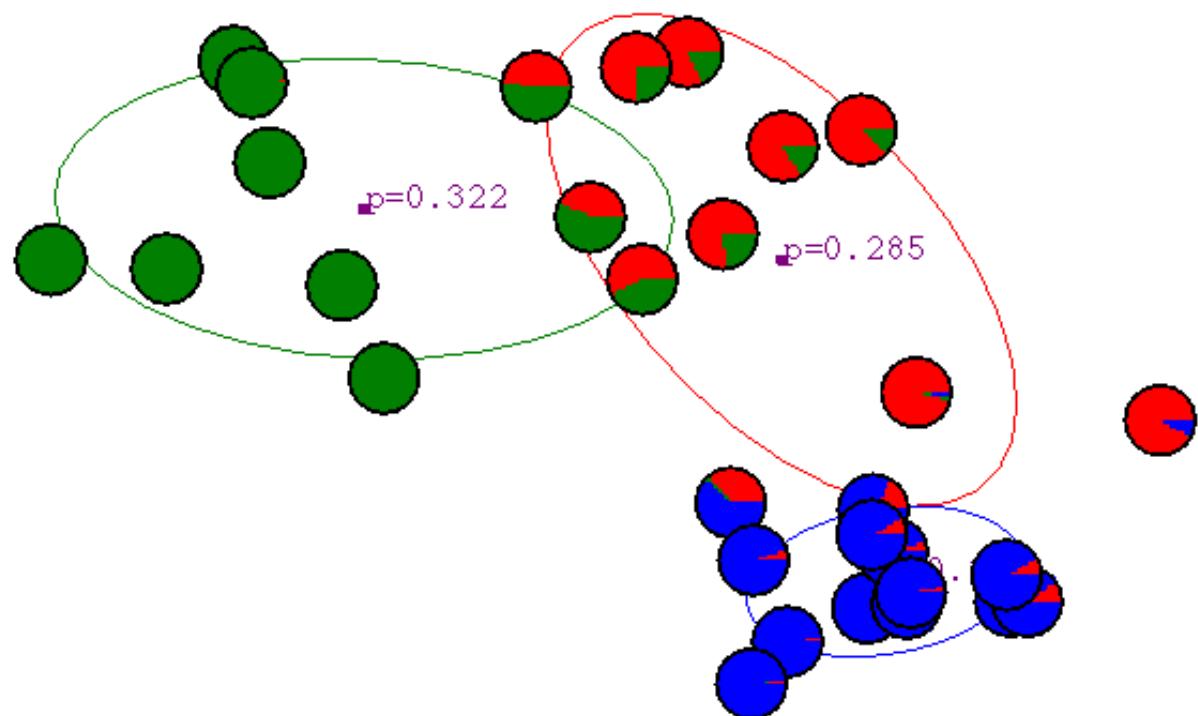
After 3rd iteration



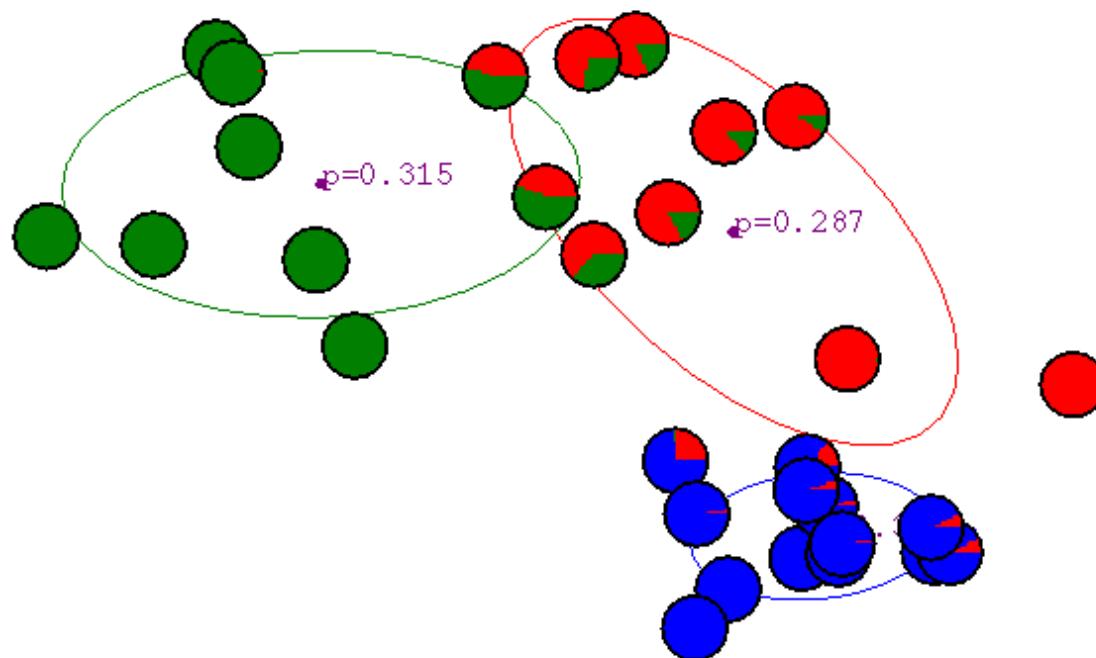
After 4th iteration



After 5th iteration

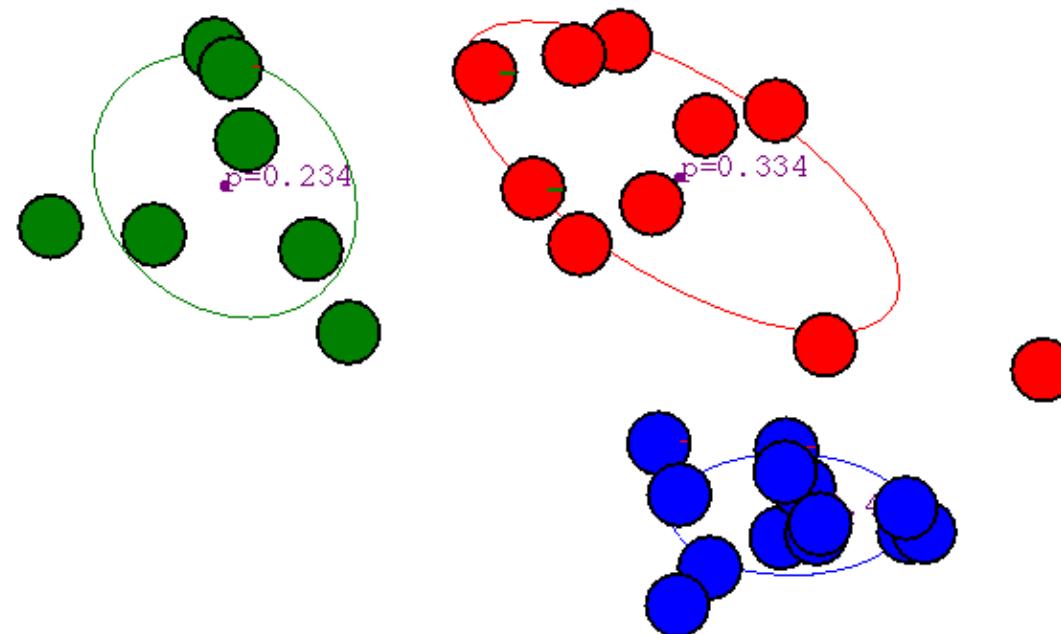


After 6th iteration

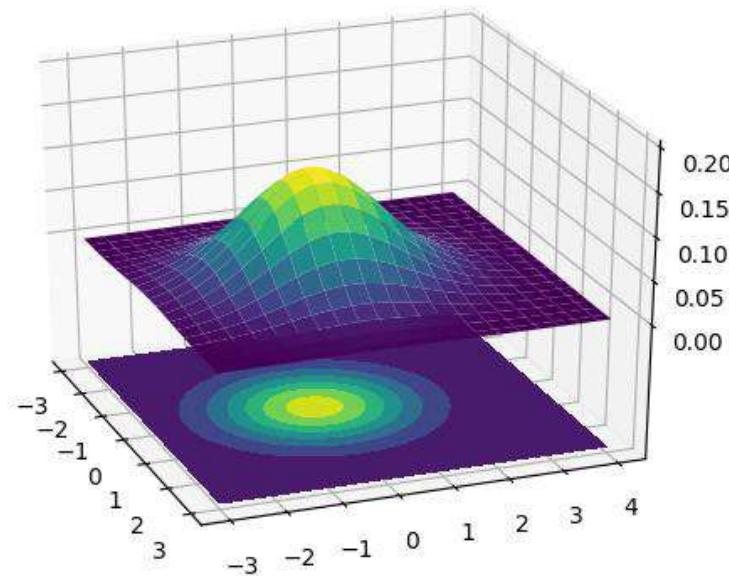
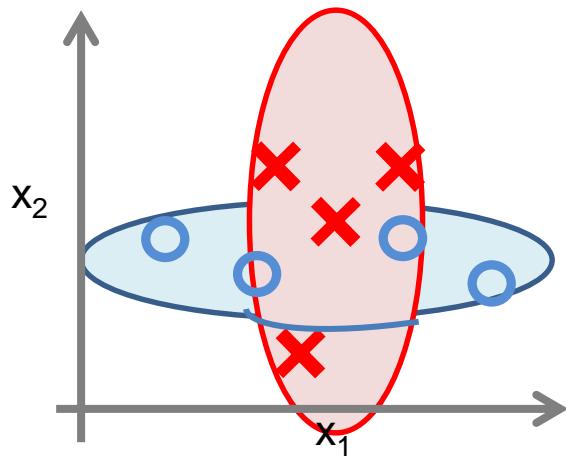


Slide Credit: Andrew W. Moore

After 20th iteration

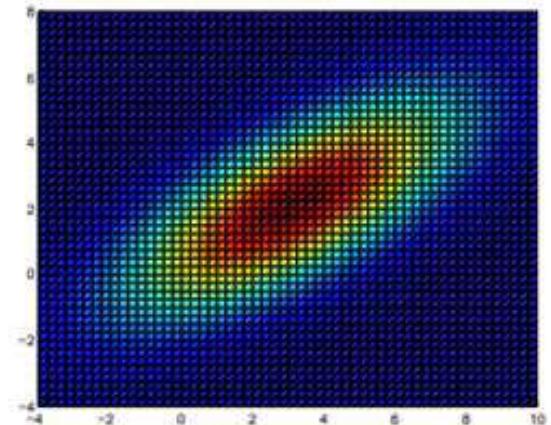
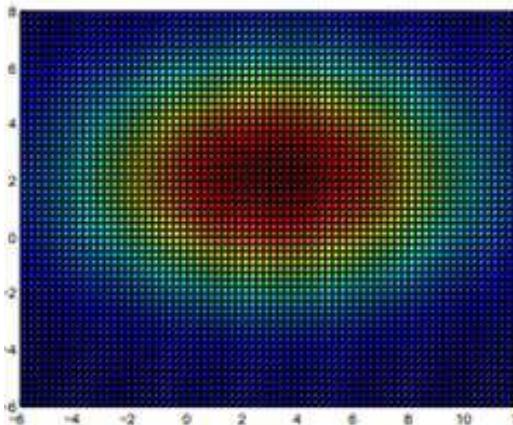
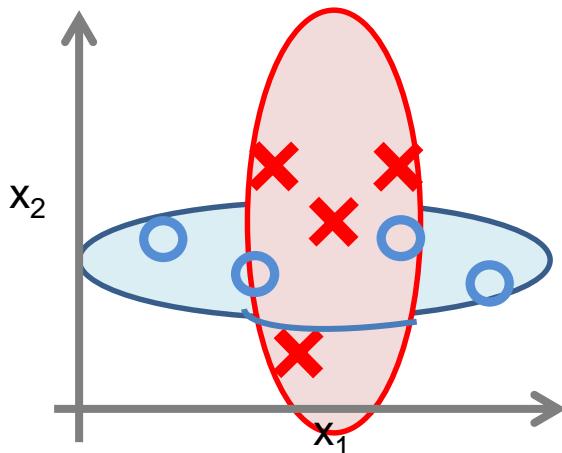


Gaussian Mixture Model



$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Gaussian Mixture Model



$$\frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2\right) \cdot \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2\right).$$

An n -dimensional Gaussian with mean $\mu \in \mathbb{R}^n$ and diagonal covariance matrix $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2)$ is the same as a collection of n independent Gaussian random variables with mean μ_i and variance σ_i^2 , respectively.

EM Algorithm

	Word1	Word2	innovate	achieve	lead
Doc1	2	2			
Doc2	4	5			
Doc3	7	2			
Cluster 1	(3,3)	(4,0.707)	0.5		
Cluster 2	(4,4)	(0.5,0.707)	0.5		

I Standardize the data if required:

II Fix the no.of.cluster expected

III Initialize the prototypes:
Mean, Covariance, Weights

IV Expectation-Step: Fix prototype & find
the membership of each point weighted
by the probability value

V. Calculate the log likelihood of the
points

VI Maximization Step: Fix the
membership(responsibility matrix) and
re-estimate the prototypes

VII Calculate the new log likelihood of
the points. Repeat E & M Step till
convergence is achieved:

GMM – Example

2 dimension

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value

Sample calculation for this step is shown below:

$$\begin{aligned}
 \text{Weight1} * N1 &= \\
 0.5 * P(\text{Doc1}, \text{Cluster1}) &= (\text{Doc1}, (3,3), (4,0.707)) \\
 &= 0.5 * \frac{e^{\frac{-(2-3)^2}{2*4^2}}}{4*\sqrt[2]{2*3.14}} * \frac{e^{\frac{-(2-3)^2}{2*0.707^2}}}{0.707*\sqrt[2]{2*3.14}} \\
 &= 0.01004
 \end{aligned}$$



	Word1	Word2	Means	SDs	Weight
Doc1	2	2	(3,3)	(4,0.707)	0.5
Doc2	4	5	(4,4)	(0.5,0.707)	0.5
Doc3	7	2			

$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

For Cluster 1 assumed parameters, here for Doc1 apply in the below formula and multiply with the weight = 0.5. For simplicity , assume off diagonal values in covariance matrix as 0 (This is usually done in text analysis use cases). similarly do it for all the Doc's

$$p(x; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

GMM – Example

2 dimension

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value



	Word1	Word2	innovate	achieve	lead
Doc1	2	2			
Doc2	4	5			
Doc3	7	2			

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Similarly:

For Cluster 2 assumed parameters, here for Doc1 apply in the below formula and multiply with the weight = 0.5. For simplicity , assume off diagonal values in covariance matrix as 0 (This is usually done in text analysis use cases). similarly do it for all the Doc's

$$p(x; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

GMM – Example

2 dimension

	Word1	Word2	innovate	achieve	lead
Doc1	2	2			
Doc2	4	5		Means	SDs
Doc3	7	2	Cluster 1	(3,3)	(4,0.707)
			Cluster 2	(4,4)	(0.5,0.707)
					0.5

$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

For Doc2 :

$$\text{Weight1} * \text{N1} = 0.0005$$

$$\text{Weight2} * \text{N2} = 0.0828$$

$$\text{Sum} = 0.08329 = \sim 0.0833$$

$$P(Z_{\text{Doc1-Cluster1}}) = 0.006$$

$$P(Z_{\text{Doc1-Cluster1}}) = 0.994$$

Per Docs sum the previous two found values.
This gives the denominator of this formula

Now find the membership of doc :

$P(Zi1) = \text{dividing Weight1} * \text{N1 by Sum}$

$P(Zi2) = \text{dividing Weight2} * \text{N1 by Sum}$

Similarly do it for all the Doc's

GMM – Example

2 dimension

V. Calculate the log likelihood of the points

	Word1	Word2	innovate	achieve	lead
Doc1	2	2			
Doc2	4	5			
Doc3	7	2	Cluster 1	(3,3)	(4,0.707)
			Cluster 2	(4,4)	(0.5,0.707)
					0.5

$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Log likelihood =
 $\ln(0.010) + \ln(0.083) + \ln(0.006)$
 $= -12.16$

Log likelihood = $\ln(\text{sum of Doc1}) + \ln(\text{sum of Doc2}) + \ln(\text{sum of Doc3})....$

This value needs to be found for every iteration and at one iteration it will approach a very large value . That will be one of the stopping convergence point of the algorithm.

GMM – Example

2 dimension

VI Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototypes

Sample calculations shown below

	Word1	Word2	innovate	achieve	lead	
Doc1	2	2				
Doc2	4	5				
Doc3	7	2				
			Cluster 1	Means	SDs	
			Cluster 1	(3,3)	(4,0.707)	Weight
			Cluster 2	(4,4)	(0.5,0.707)	0.5

	Doc1	Doc2	Doc3
P(Zi1)	0.9999	0.006	1
P(Zi2)	0.0001	0.994	0

	Means	SDs	Weight
Cluster 1			
Cluster 2			

$$N_{k=1} = \text{Cluster 1} = \\ = 0.9999 + 0.006 + 1 \\ = 2.0059$$

This is the value N1 represented here

$$\text{MEAN}_{k=1} = \\ (4.49, 2.00) = \sim(4.49, 2)$$

$$\text{Std.Dev}_{k=1} = \\ ([0.9999(2-4.49)^2 + 0.006(4-4.49)^2 + 1(7-4.49)^2]/N1, \\ [0.9999(2-2)^2 + 0.006(5-2)^2 + 1(2-2)^2]/N1) \\ = (6.23, 0.03)$$

$N_k = \sum_{n=1}^N \gamma(z_{nk})$ N1 for cluster k=1 is got by summing these.
Similarly do it for N2

For cluster k=1 is Means obtained by multiplying the data by the membership.

$$[0.9999(2) + 0.006(4) + 1(7)]/N1, [0.9999(2) + 0.006(5) + 1(2)]/N1$$

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

Similarly do it for cluster 2

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$$\text{Weight}_{k=\text{cluster1}} (\text{new}) \\ = 2.0059/3 = 0.67$$

GMM - EM Algorithm

2nd Iteration continues

I Standardize the data if required:

II Fix the no.of.cluster expected

III Initialize the prototypes:
Mean, Covariance, Weights

IV Expectation-Step: Fix prototype
& find the membership of each
point weighted by the probability
value

V. Calculate the log likelihood of the
points

VI Maximization Step: Fix the
membership(responsibility matrix)
and re-estimate the prototypes

VII Calculate the new log likelihood
of the points. Repeat E & M Step till
convergence is achieved:

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

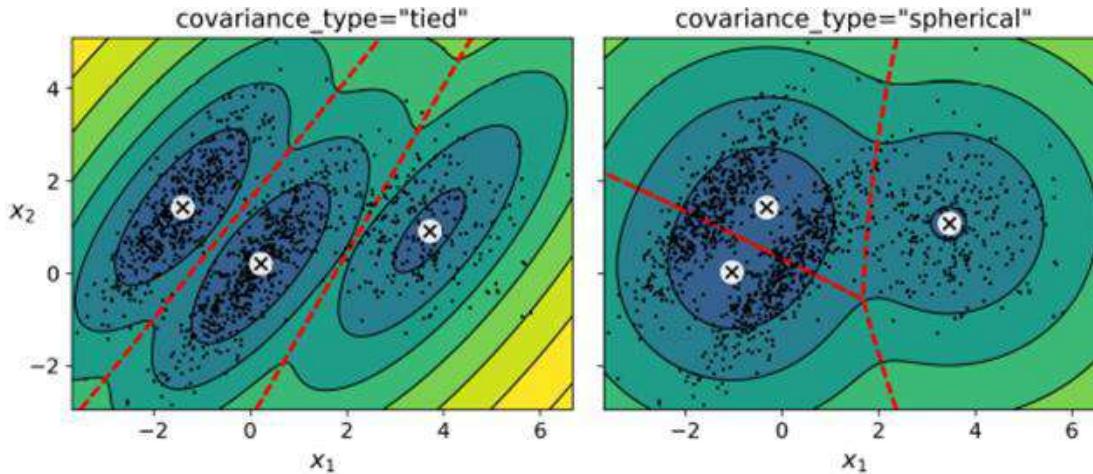


$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

	Means	SDs	Weight
Cluster 1	(4.49, 2)	(6.32, 0.03)	0.67
Cluster 2	(3.99, 4.99)	(0.001, 0.001)	0.33

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Gaussian Mixture Model



Note:

The GMM model can be relaxed by setting these parameters while calling the function

- "**spherical**": all clusters must be spherical, but they can have different diameters (i.e., different variances).
- "**diag
- "**tied**": all clusters must have the same ellipsoidal shape, size and orientation (i.e., all clusters share the same covariance matrix).**

Example – 2 for Students – One dimensional



Data

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.

The prior over the two Gaussians is also given: $\lambda_1 = \lambda_2 = 0.5$

Let j be an index over the Gaussians, i an index over the data points, and k an index over the E-M iterations.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3, \mu_2^{[0]} = 6$.

$$\begin{aligned}\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \\ \pi_k^{\text{new}} &= \frac{N_k}{N}\end{aligned}$$

Example – 2 for Students – One dimensional



Data

i	1	2	3
x_i	2.0	4.0	7.0
$N(x_i \mu_1)$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-16}$
$N(x_i \mu_2)$	$\frac{1}{\sqrt{\pi}}e^{-16}$	$\frac{1}{\sqrt{\pi}}e^{-4}$	$\frac{1}{\sqrt{\pi}}e^{-1}$
$N(x_i \mu_1) + N(x_i \mu_2)$	$\frac{1}{2\sqrt{\pi}}(e^{-1} + e^{-16})$	$\frac{1}{2\sqrt{\pi}}(e^{-1} + e^{-4})$	$\frac{1}{2\sqrt{\pi}}(e^{-16} + e^{-1})$
$\gamma(z_i, 1)$	$\frac{e^{-1}}{e^{-1} + e^{-16}} \approx 1$	$\frac{e^{-1}}{e^{-1} + e^{-4}} \approx 0.953$	$\frac{e^{-16}}{e^{-1} + e^{-16}} \approx 0$
$\gamma(z_i, 2)$	$\frac{e^{-16}}{e^{-1} + e^{-16}} \approx 0$	$\frac{e^{-4}}{e^{-1} + e^{-4}} \approx 0.047$	$\frac{e^{-1}}{e^{-1} + e^{-16}} \approx 1$

And therefore:

$$\mu_1^{[1]} \approx \frac{1 * 2.0 + 0.953 * 4.0 + 0 * 7.0}{1 + 0.953} \approx 2.976$$

and

$$\mu_2^{[1]} \approx \frac{0 * 2.0 + 0.047 * 4.0 + 1 * 7.0}{1 + 0.047} \approx 6.865$$

References

- Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
- <https://www.youtube.com/watch?v=TG6Bh-NFhA0>
- <https://www.youtube.com/watch?v=qMTuMa86NzU>