



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 1(22<sup>nd</sup> July 2023)**

# Syllabus - Handout

No	Title of the Module	References
M1	Introduction to Machine Learning 1. What and Why 2. Applications of Machine Learning 3. Types of Machine Learning 4. Challenges in Machine Learning	T1: Chapter 1 R2
M2	Big Picture: End-to-end Machine Learning 2.1. Framing the ML Problem 2.2. Data Types, Pre-processing, Visualization and Analysis	T1: Chapter 2 T2: Chapter 2-3
M3	Big Picture: End-to-end Machine Learning 3.1 Model Selection and Training 3.1.1. Prediction Problem 3.1.2. Classification Problem 3.2 Evaluation 3.2.1. Prediction Problem 3.2.2. Classification Problem 3.3 Machine Learning Pipeline	T1: Chapter 2-3 T2: Chapter 4
M4	Linear Prediction Models 1. Linear Regression 2. Gradient Descent and Variants 3. Regularization 4. Bias Vs. Variance	T1: Chapter 4 T2: Appendix D

# Syllabus - Handout

M5	Classification Models I 1. Logistic Regression 2. Support Vector Machines 3. Naïve Bayes 4. Comparative Analysis	T1: Chapter 4, 5 T2: Chapter 5
M6	Classification Models II 1. Decision Tree 2. Challenges with Decision tree 3. Ensembles 6.2.1. Bagging 6.2.2. Boosting 1. Random Forest	T1: Chapter 6-7 T2: Chapter 4, 5
M7	Unsupervised Learning 7.1. Dimensionality reduction and feature extraction 7.2 K-means Clustering 7.4 EM Algorithm 7.5 Applications	T1: Chapter 8-9 T2: Chapter 8-9
M8	Neural Networks 8.1. Perceptrons and Delta Rule 8.2. Multi-Layer Perceptrons and Backpropagation 8.3. Deep Learning	T1: Chapter 10-11 T2: Chapter 5
M9	Deep Networks 9.1. Convolutional Neural Network 9.2. Recurrent Neural Network 9.3. Applications in Image and Text Processing	T1: Chapter 14, 15
M10	FAccT Machine Learning 10.1. Bias and Fairness 10.2. Interpretability and Transparency 10.3. Robustness and Adversarial Attacks	R1

# Treasure Hunt

## Text Book(s)

T1	Aurelien Geron, "Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow", O'Reilly, 2020
T2	P-N Tan, M. Steinbach, Vipin Kumar, "Introduction to Data Mining", 2016, Pearson

## Reference Book(s) & other resources



R1	Christoph Molner, "Interpretable Machine Learning", 2020, <a href="https://christophm.github.io/interpretable-ml-book/">https://christophm.github.io/interpretable-ml-book/</a>
R2	Pedro Domingos, "A Few Useful Things to Know About Machine Learning", pp.78-87, Communications of the ACM, vol. 55 no. 10, October 2012.

# Session 1 – 22<sup>nd</sup> July 2023

1

Introduction to Machine Learning: What and Why,  
Applications of Machine Learning, Types of Machine  
Learning, Challenges in Machine Learning

T1: Chapter 1

# In this segment

- What is Machine Learning
- Why Machine Learning
- Connection with Other technical areas



# What is Machine Learning?

Gmail  X ≡ ?

Compose

Inbox 8,707 Starred Snoozed Important Sent Drafts 63 Categories

← Delete forever Not spam Reply Compose Forward ⋮ 22 of 66

Fantastic offers to bring in 2023 Spam x

Imagine by Ample <support@imagineonline.store>  
to me Thu, Dec 29, 2022, 6:15 PM (9 days ago) ☆

Why is this message in spam? It is similar to messages that were identified as spam in the past.

Report not spam



# What is Machine Learning?

Machine learning “ Field of Study that gives the computers the ability to learn without being explicitly programmed”

---- Arthur Samuel (1959)

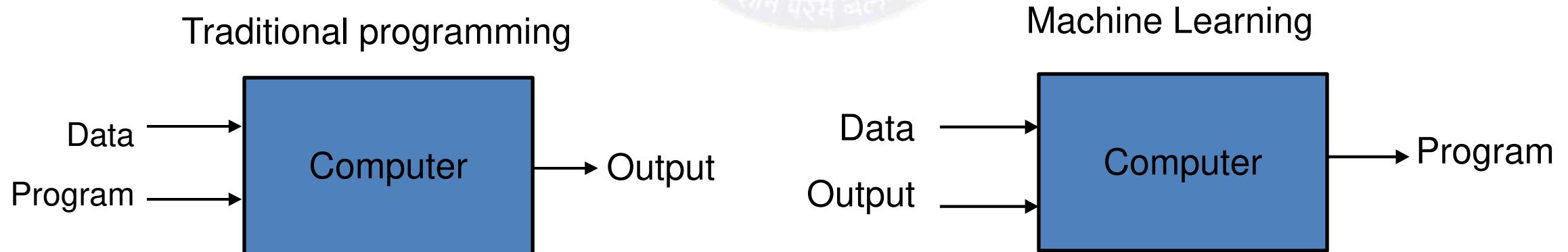
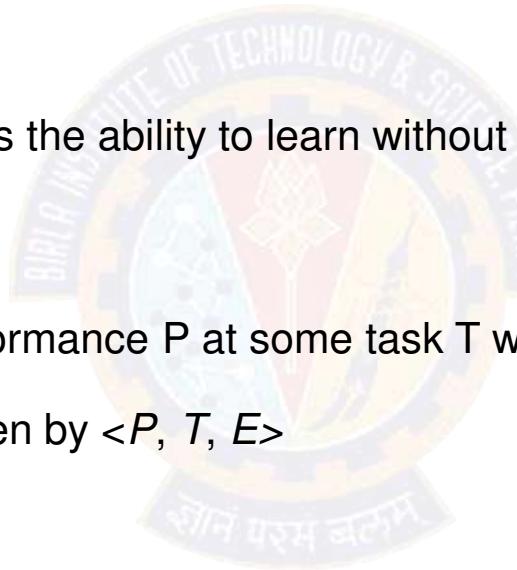
# What is Machine Learning?

<b>Input (X)</b>	<b>Output(Y)</b>	<b>Application</b>
email	Spam(0/1)	Spam filtering
English	Hindi	Machine translation
audio	Text transcripts	Speech recognition
Medical Images	Tumour(0/1)	Visual inspection
Image, radar info	Position of cars	Self-driving car

# What is Machine Learning (ML)?

## What is Machine Learning

- 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  $\langle P, T, E \rangle$



# Defining the Learning Tasks

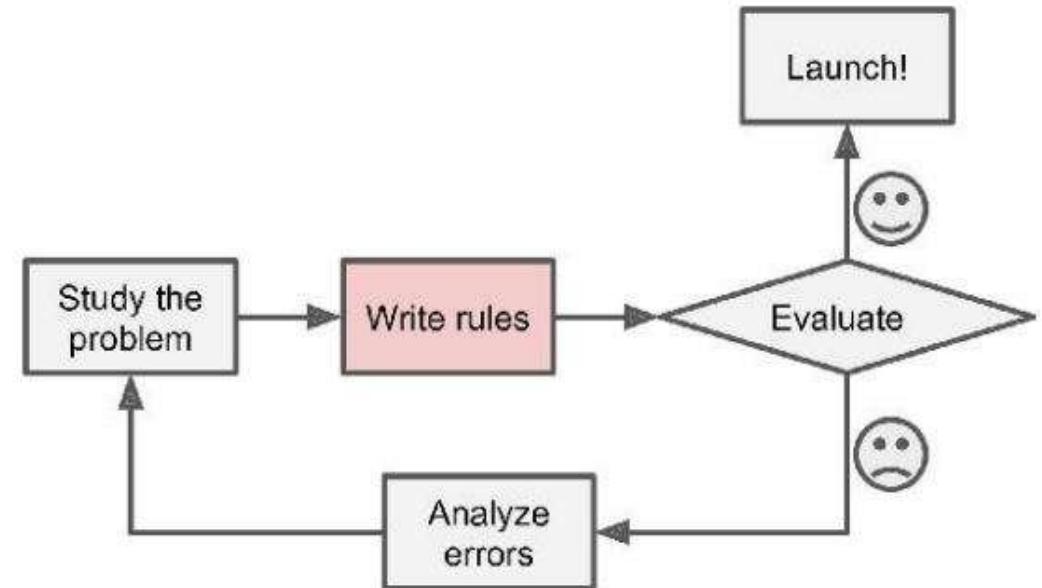
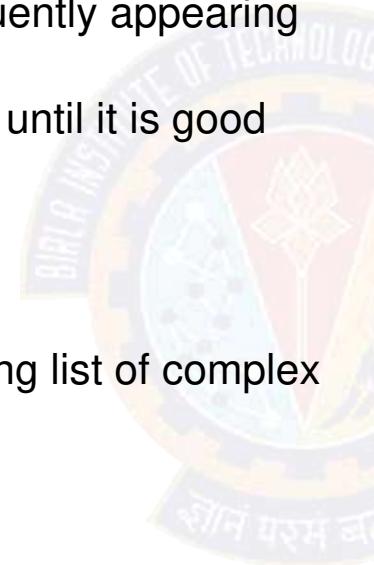
**Improve on task T, with respect to performance metric P, based on experience E**

- Example 1
  - T: Playing checkers
  - P: Percentage of games won against an arbitrary opponent
  - E: Playing practice games against itself
- Example 2
  - T: Recognizing hand-written words
  - P: Percentage of words correctly classified
  - E: Database of human-labeled images of handwritten words
- Example 3
  - T: Driving on four-lane highways using vision sensors
  - P: Average distance traveled before a human-judged error
  - E: A sequence of images and steering commands recorded while observing a human driver.
- Example 4
  - T: Categorize email messages as spam or legitimate.
  - P: Percentage of email messages correctly classified.
  - E: Database of emails, some with human-given labels

# Traditional Approach to Spam Filtering

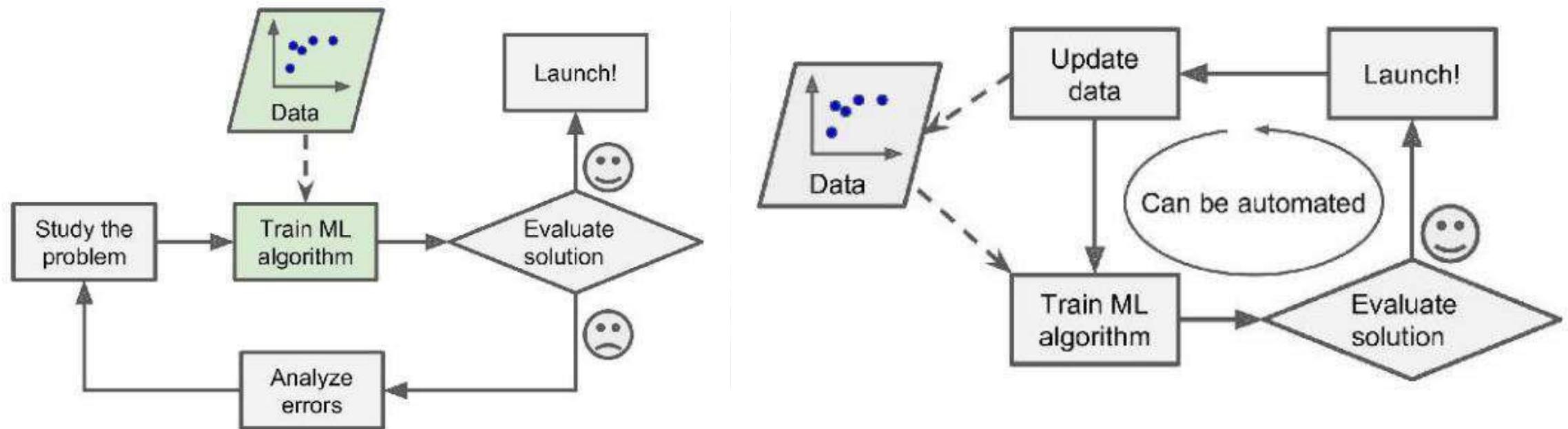
Spam typically uses words or phrases such as “4U,” “credit card,” “free,” and “amazing”

- 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.



# Machine Learning Approach

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.

# A Classic example of ML Task

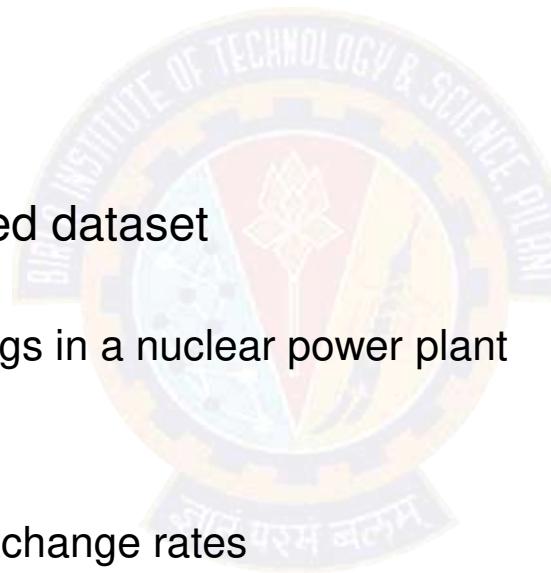
It is very hard to say what makes a “2”



# More ML Usage Scenarios

## Tasks that are best solved by using a learning algorithm

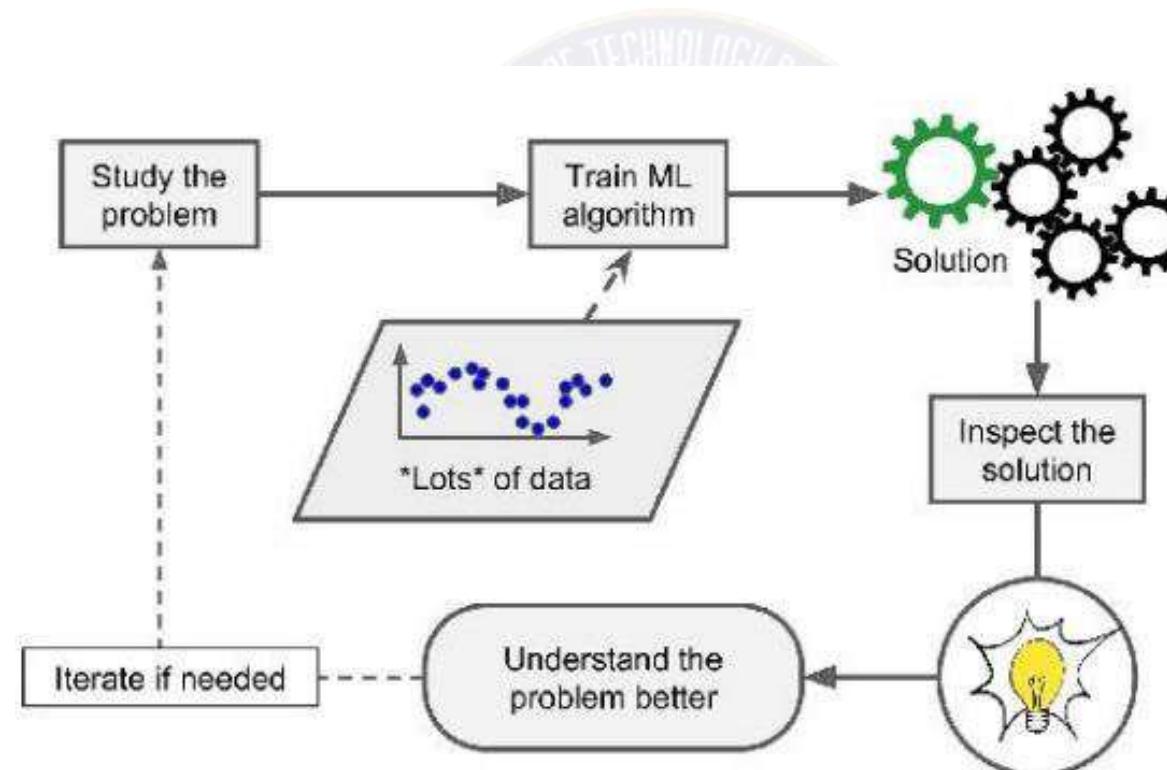
- Recognizing Patterns in images, text
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Recognizing Anomalies in structured dataset
  - Unusual credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant
- Prediction from time series data
  - Future stock prices or currency exchange rates
- Generating new Patterns
  - Generating images or motion sequences



# Machine Learning Helps Human Learning

Sometimes reveals unsuspected correlations / new trends leading to better understanding

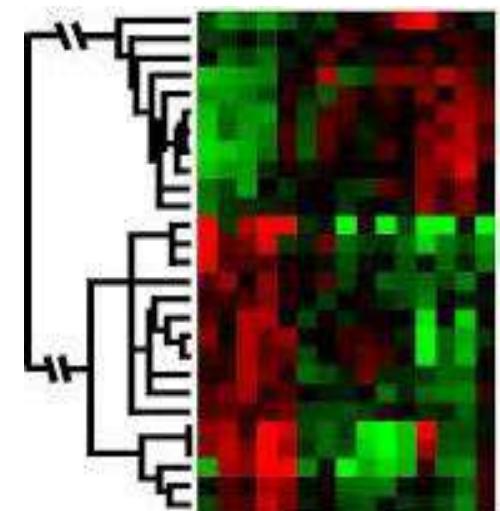
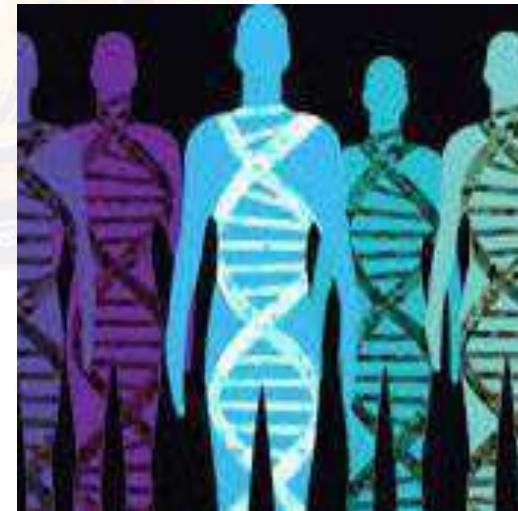
- Data Mining
  - Applying ML techniques to dig into large amounts of data can help discover patterns



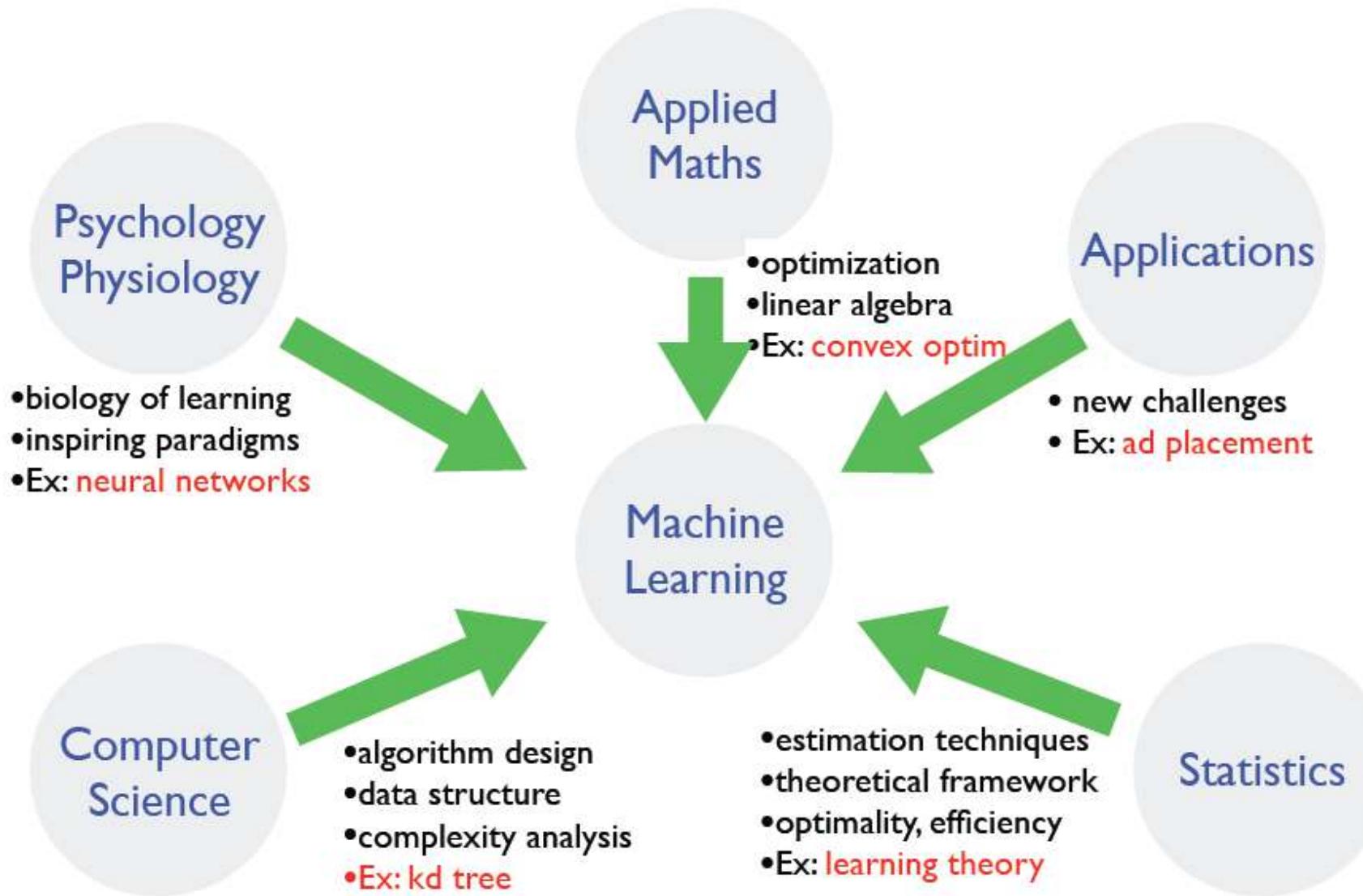
# When to Use Machine Learning

## ML Usage Scenarios

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules
- Complex problems for which there is no good solution at all using a traditional approach
- Changing environments
- Get insights about complex problems and large amounts of data



# Where does ML fit in?



# Application Domains

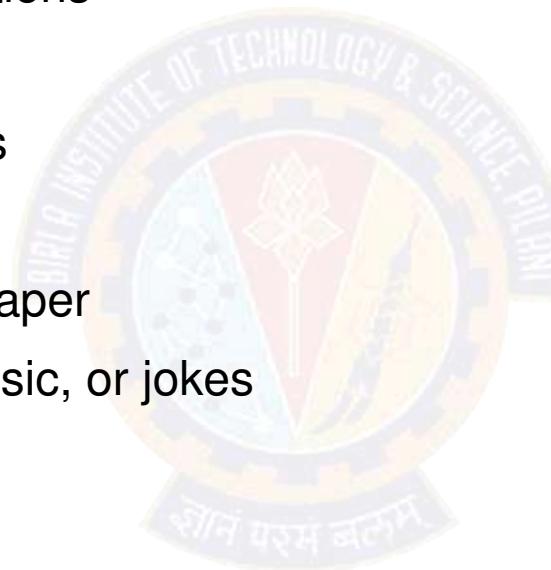
- Internet
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Software engineering
- System management
- Creative Arts



# Example: Classification

**Assign object/event to one of a given finite set of categories**

- Medical Diagnosis
- Credit card applications or transactions
- Fraud detection in e-commerce
- Worm detection in network packets
- Spam filtering in email
- Recommended articles in a newspaper
- Recommended books, movies, music, or jokes
- Financial investments
- DNA sequences
- Spoken words
- Handwritten letters
- Astronomical images



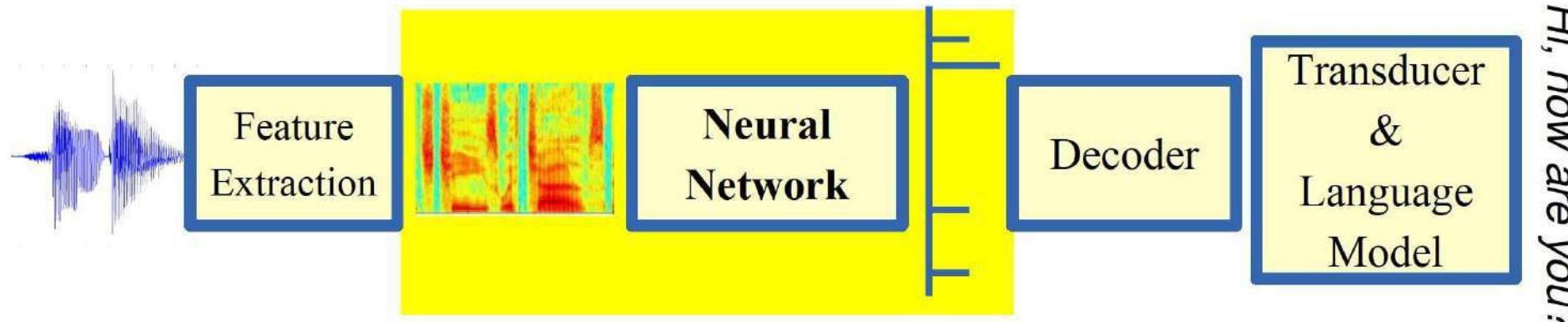
# Example: Planning, Control, Problem Solving

## Performing actions in an environment in order to achieve a goal

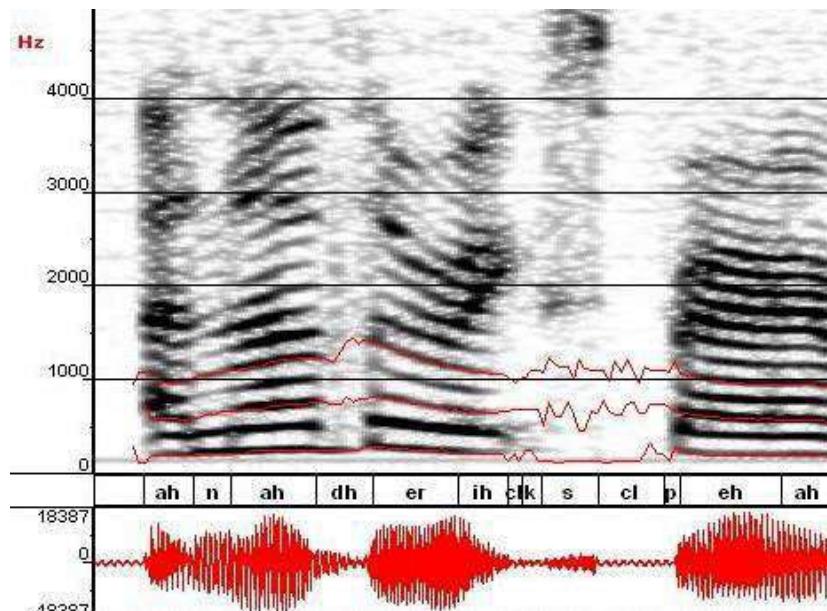
- Playing checkers, chess, or backgammon
- Balancing a pole
- Driving a car or a jeep
- Flying a plane, helicopter, or rocket
- Controlling an elevator
- Controlling a character in a video game
- Controlling a mobile robot



# Breakthrough in Automatic Speech Recognition



ML used to predict of phone states from the sound spectrogram



Deep learning has state-of-the-art results

# Hidden Layers	1	2	4	8	10	12
Word Error Rate %	16.0	12.8	11.4	10.9	11.0	11.1

Baseline (Gaussian Mixture Model) performance = 15.4%

[Reference: Zeiler *et al.*, “On rectified linear units for speech recognition” ICASSP 2013]

# Impact of Deep Learning in Speech Technology



# Visual Question Answering

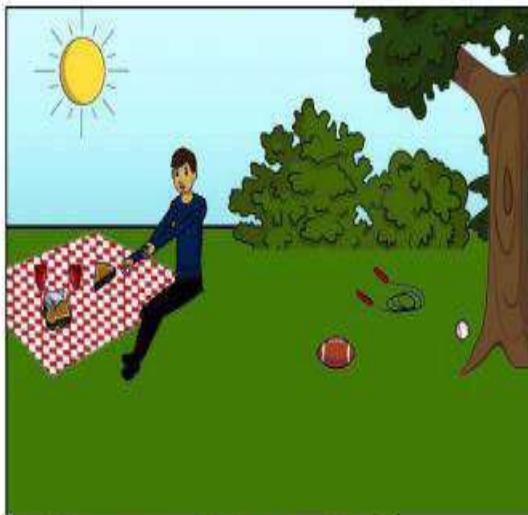
## Answering questions about images



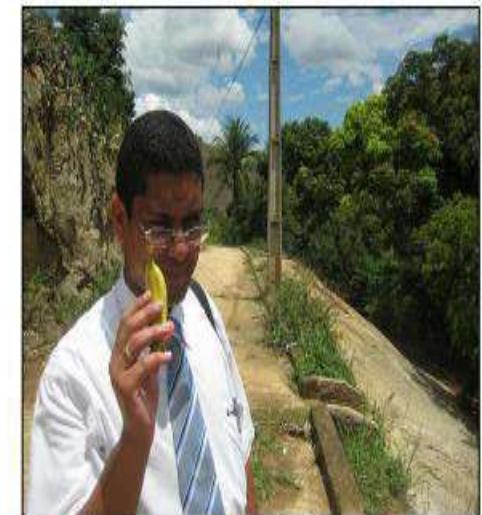
What color are her eyes?  
What is the mustache made of?



How many slices of pizza are there?  
Is this a vegetarian pizza?

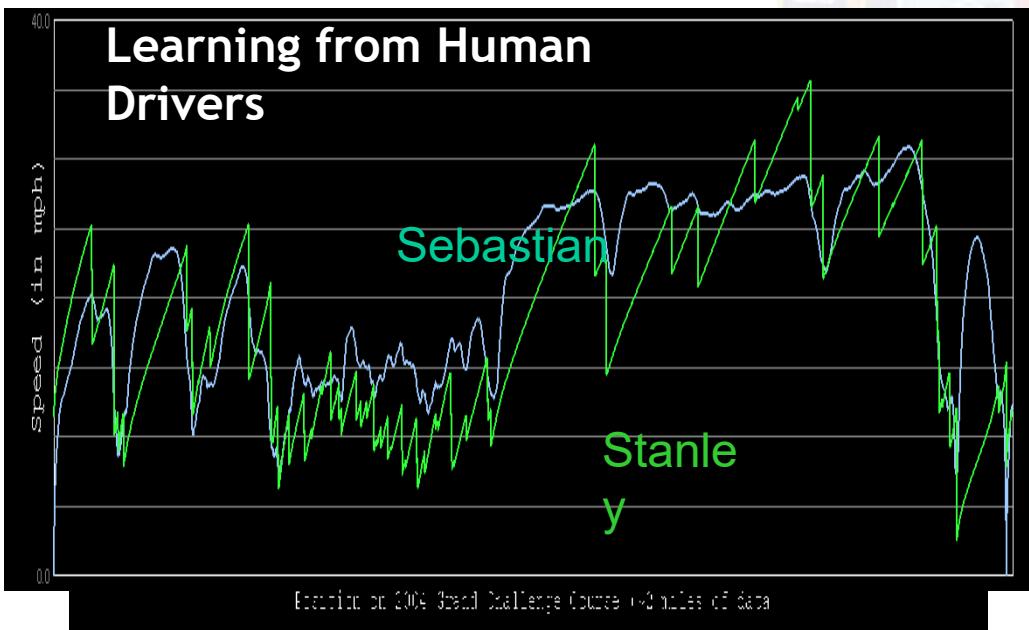
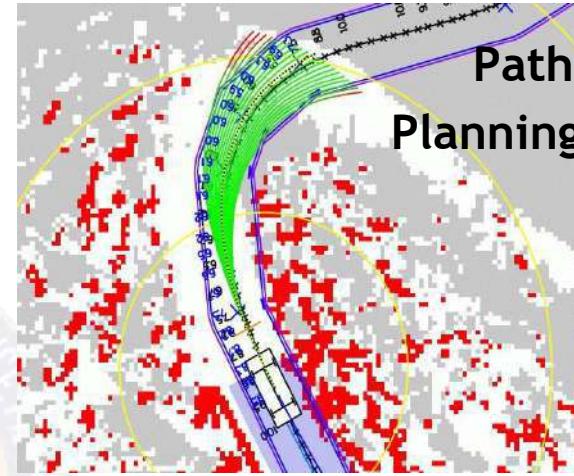


Is this person expecting company?  
What is just under the tree?



Does it appear to be rainy?  
Does this person have 20/20 vision?

# Autonomous Car Technology



# Contemporary ML Based Solutions

- Optical Character Recognition
- Product Image Classification on a production line
- Detecting tumors in brain scans
- Information extraction from document images
- Categorization of news articles
- Flagging offensive comments in discussion forums
- Document summarization
- Forecasting company revenue
- Chatbots / Voice activated App
- Detecting Credit Card Fraud
- Customer Segmentation
- Recommendation System for products, movies, news



# Types of Machine Learning Systems

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories based on:

1. Based on Training :Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)
2. Based on stream of incoming data: Whether or not they can learn incrementally on the fly (online versus batch learning)
3. How they generalize: Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do (instance-based versus model-based learning)

# ML Terminologies

**1. Labels:** A **label** is the thing we're predicting—the **y** variable in simple linear regression. The label could be the future price of gold, the kind of animal shown in a picture, the meaning of an audio clip.

**2. Features/attribute:** A **feature** is an input variable—the **x** variable in simple linear regression. A simple machine learning project might use a single feature, while a more sophisticated machine learning project could use millions of features, specified as:

$x_1, x_2, \dots, x_n$

In the spam detector example, the features could include the following:

- words in the email text
- sender's address
- time of day the email was sent

Another example: To predict the price (label) of used car , using the features,

1. mileage
2. age
3. brand

# ML Terminologies

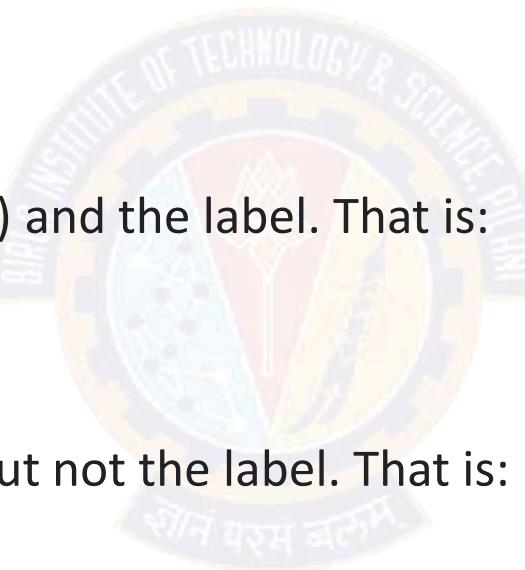
## 3.Examples/Instances

An **example** is a particular instance of data,  $\mathbf{x}$ . (We put  $\mathbf{x}$  in boldface to indicate that it is a vector.) We break examples into two categories:

- labeled examples
- unlabeled examples

A **labeled example** includes both feature(s) and the label. That is:

**labeled examples: {features, label}:  $(\mathbf{x}, y)$**



An **unlabeled example** contains features but not the label. That is:

**unlabeled examples: {features, ?}:  $(\mathbf{x}, ?)$**

# ML Terminologies

## 4. Models

A model defines the relationship between features and label. For example, a spam detection model might associate certain features strongly with "spam". Let's highlight two phases of a model's life:

- 1. Training** means creating or **learning** the model. That is, you show the model labeled examples and enable the model to gradually learn the relationships between features and label.
- 2. Inference** means applying the trained model to unlabeled examples. That is, you use the trained model to make useful predictions ( $y'$ ). For example, during inference, you can predict HouseValue for new unlabeled examples.

# Types of Learning

## Based on level of supervision

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

# Types of Learning : Supervised Learning

## Based on level of supervision

### Regression vs. classification

❖ A **regression** model predicts continuous values. For example, regression models make predictions that answer questions like the following:

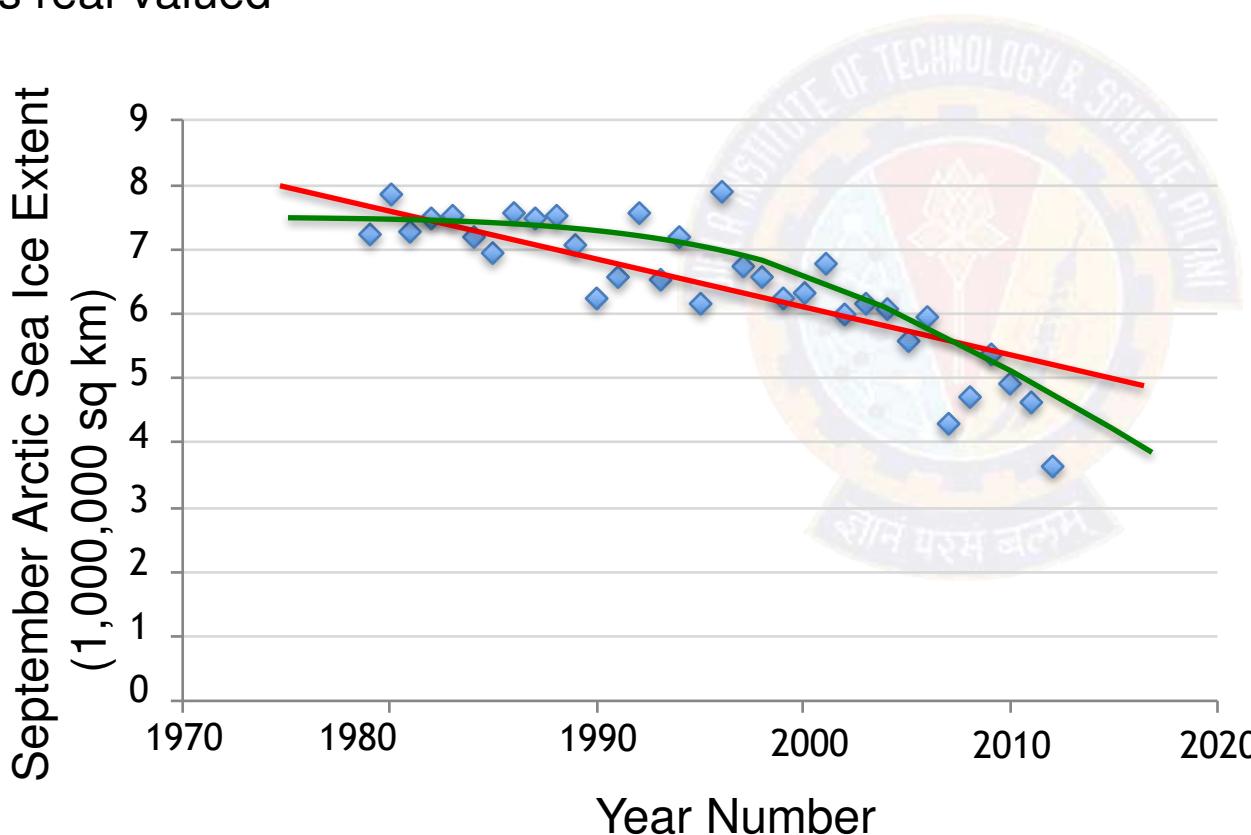
1. What is the value of a house in Bangalore?
2. What is the probability that a user will click on this ad?

❖ A **classification** model predicts discrete values. For example, classification models make predictions that answer questions like the following:

1. Is a given email message spam or not spam?
2. Is this an image of a dog, a cat, or a hamster?

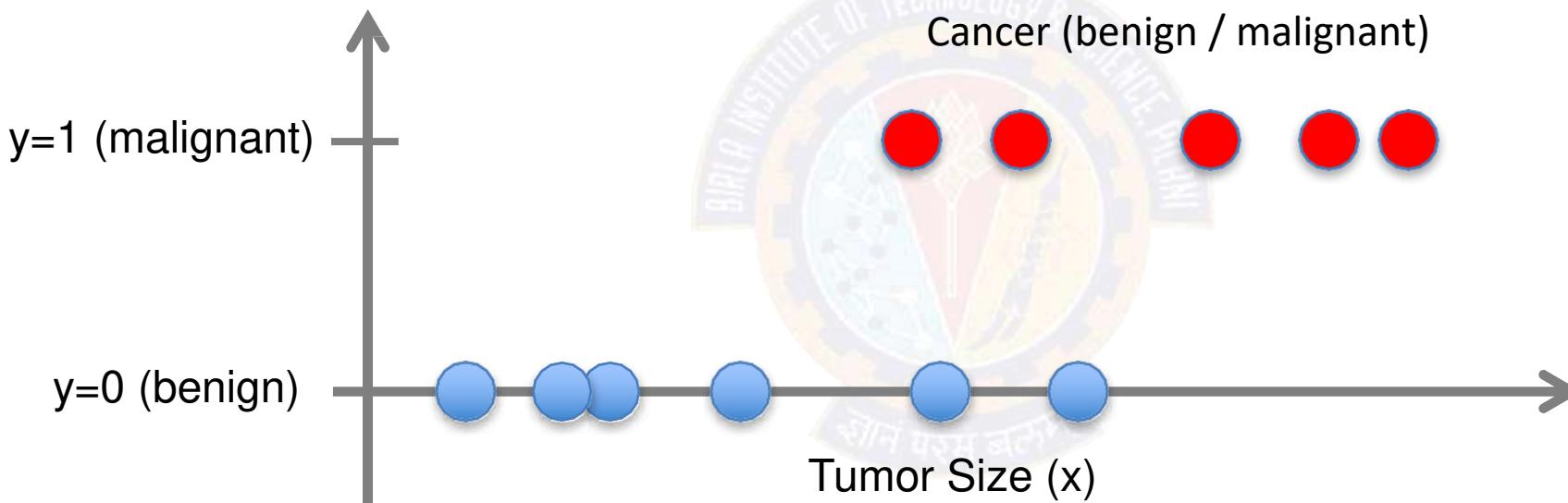
# Supervised Learning: Regression

- 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



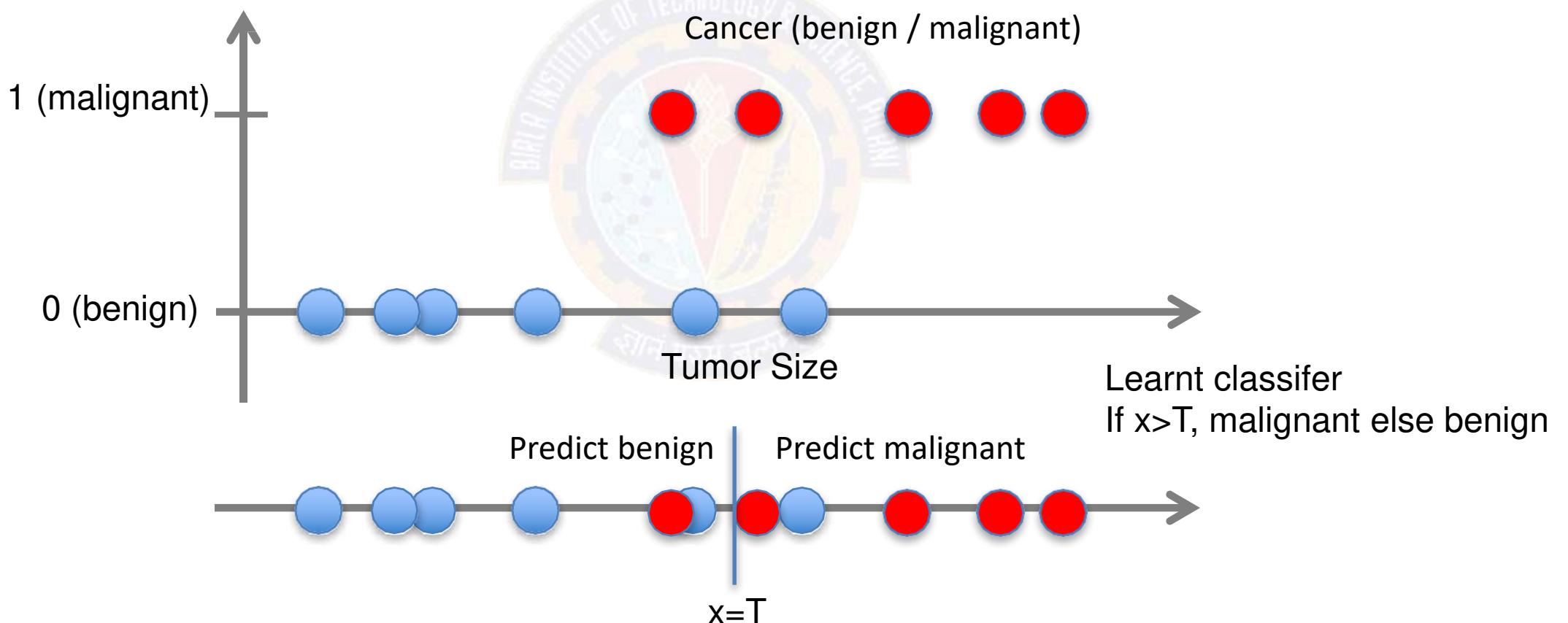
# Supervised Learning: Classification

- 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



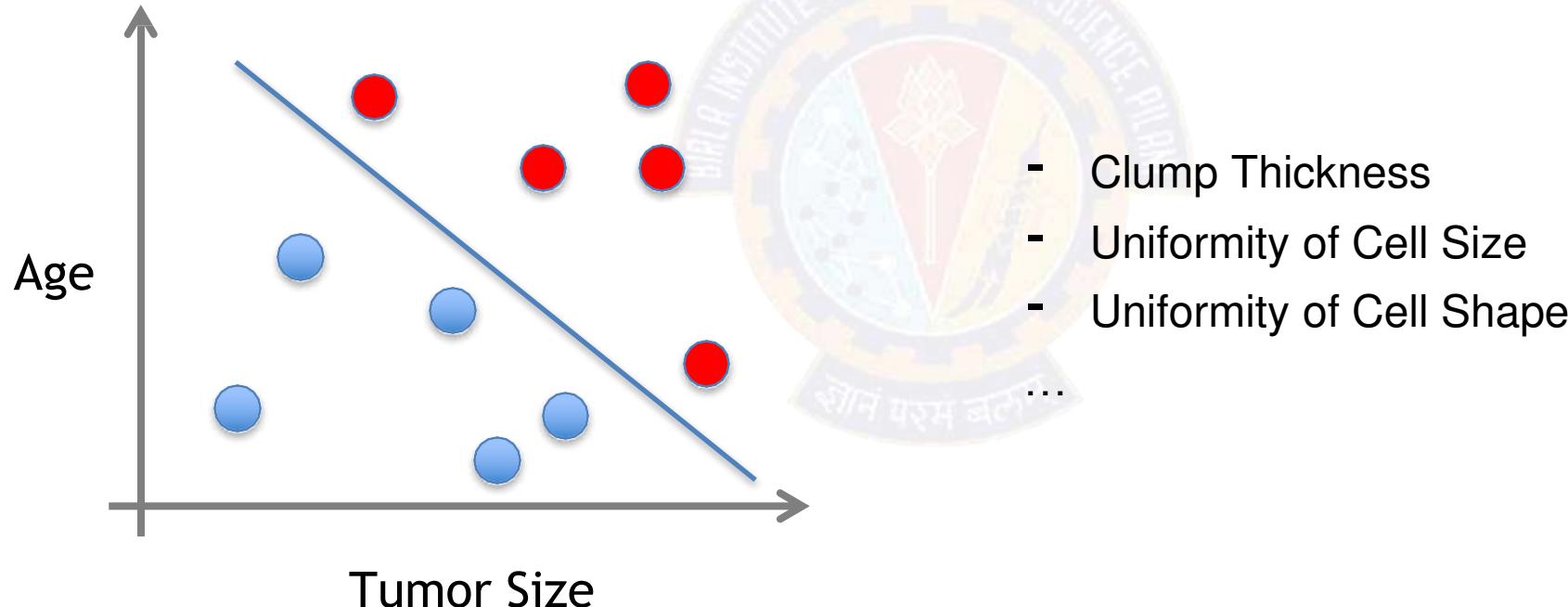
# Supervised Learning: Classification

- 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



# Increasing Feature Dimension

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



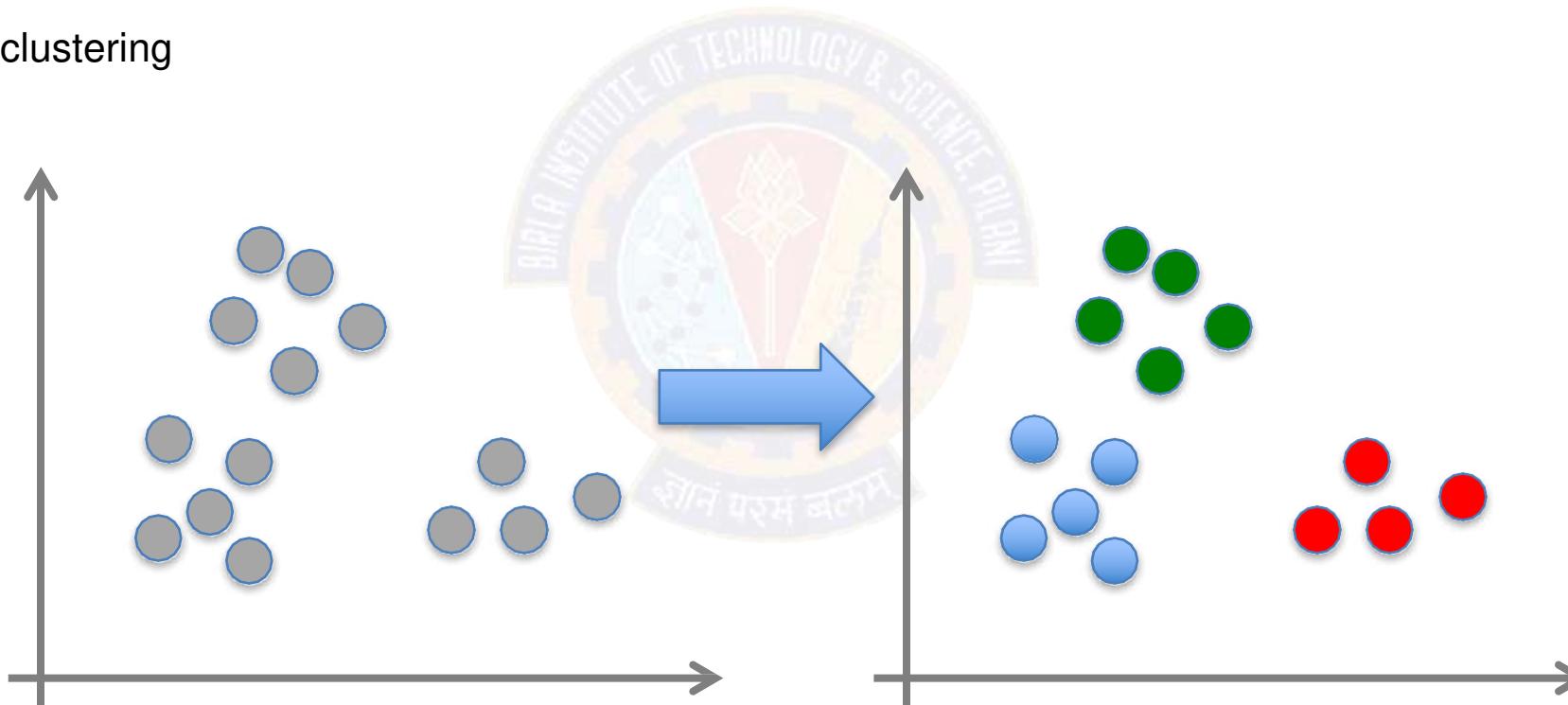
# Example: Supervised Learning Techniques

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



# Unsupervised Learning

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



# Example: Unsupervised Learning Techniques

## ❖ Clustering

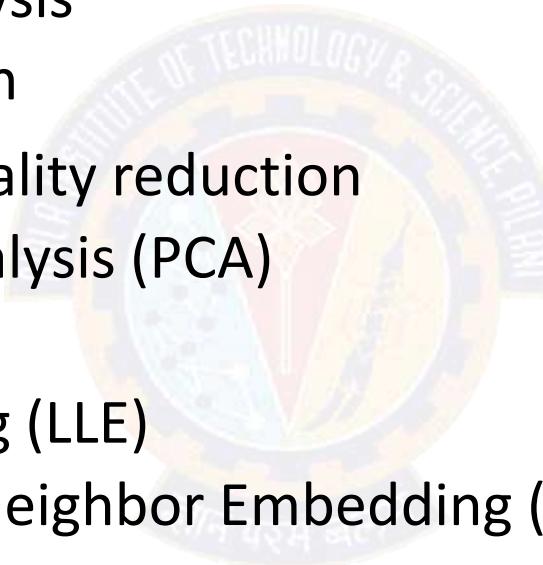
- k-Means
- Hierarchical Cluster Analysis
- Expectation Maximization

## ❖ Visualization and dimensionality reduction

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

## ❖ Association rule learning

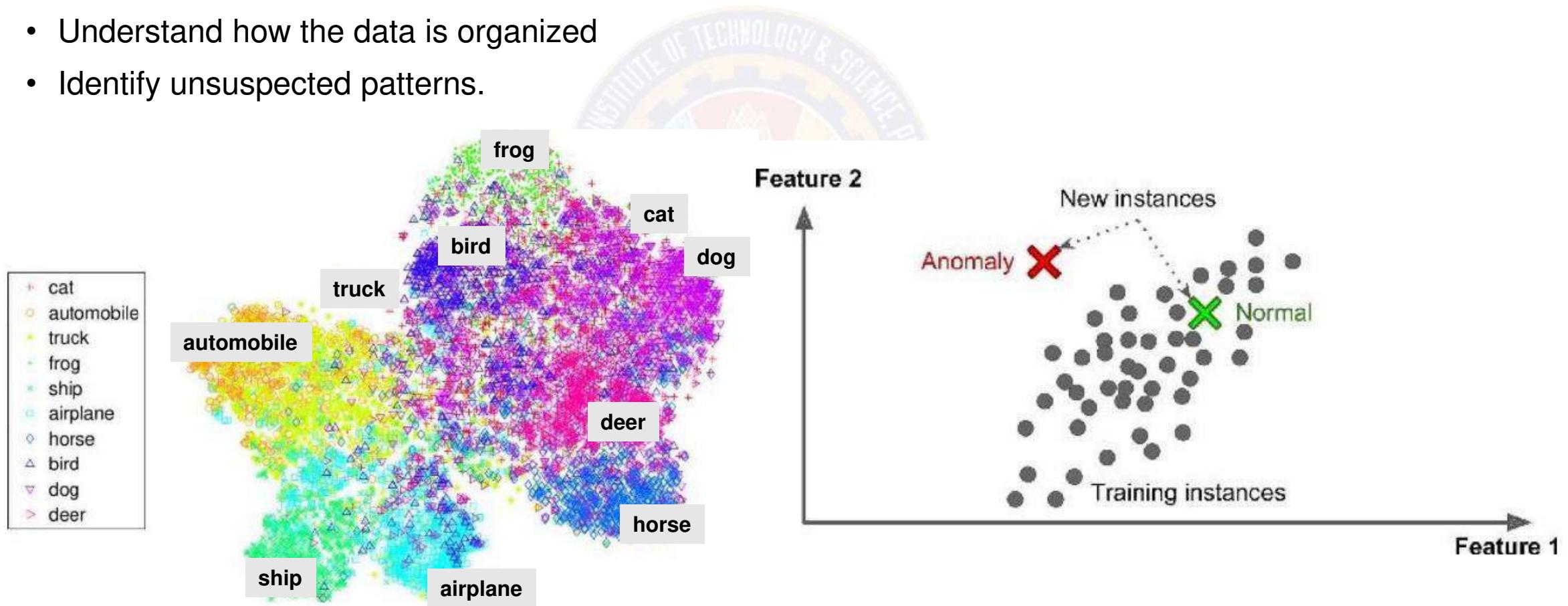
- Apriori
- Eclat



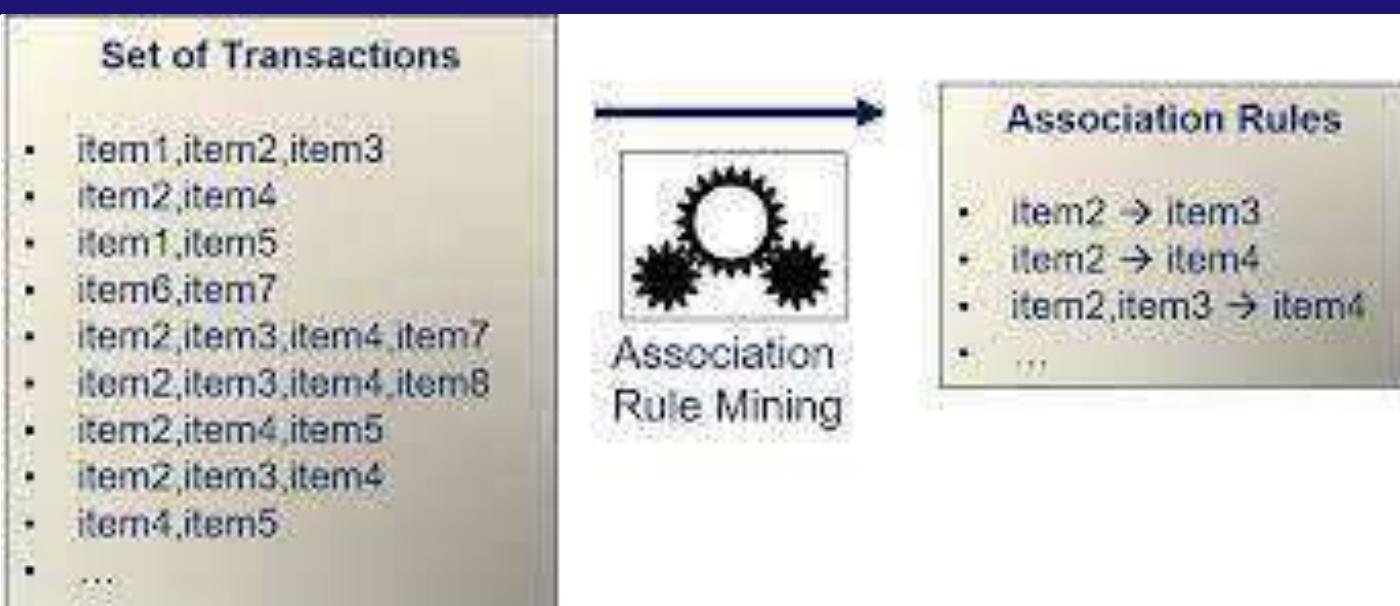
# Data Visualization

## Visualize 2/3D representation of complex unlabelled training data

- Preserve as much structure as possible
  - e.g., trying to keep separate clusters in the input space from overlapping in the visualization
- Understand how the data is organized
- Identify unsuspected patterns.



# Association Rule Mining



ID	Items
1	{Bread, Milk}
2	{Bread, <b>Diapers, Beer, Eggs</b> }
3	{Milk, <b>Diapers, Beer, Cola</b> }
4	{Bread, Milk, <b>Diapers, Beer</b> }
5	{Bread, Milk, Diapers, Cola}
...	...

market basket transactions

{Diapers, Beer}

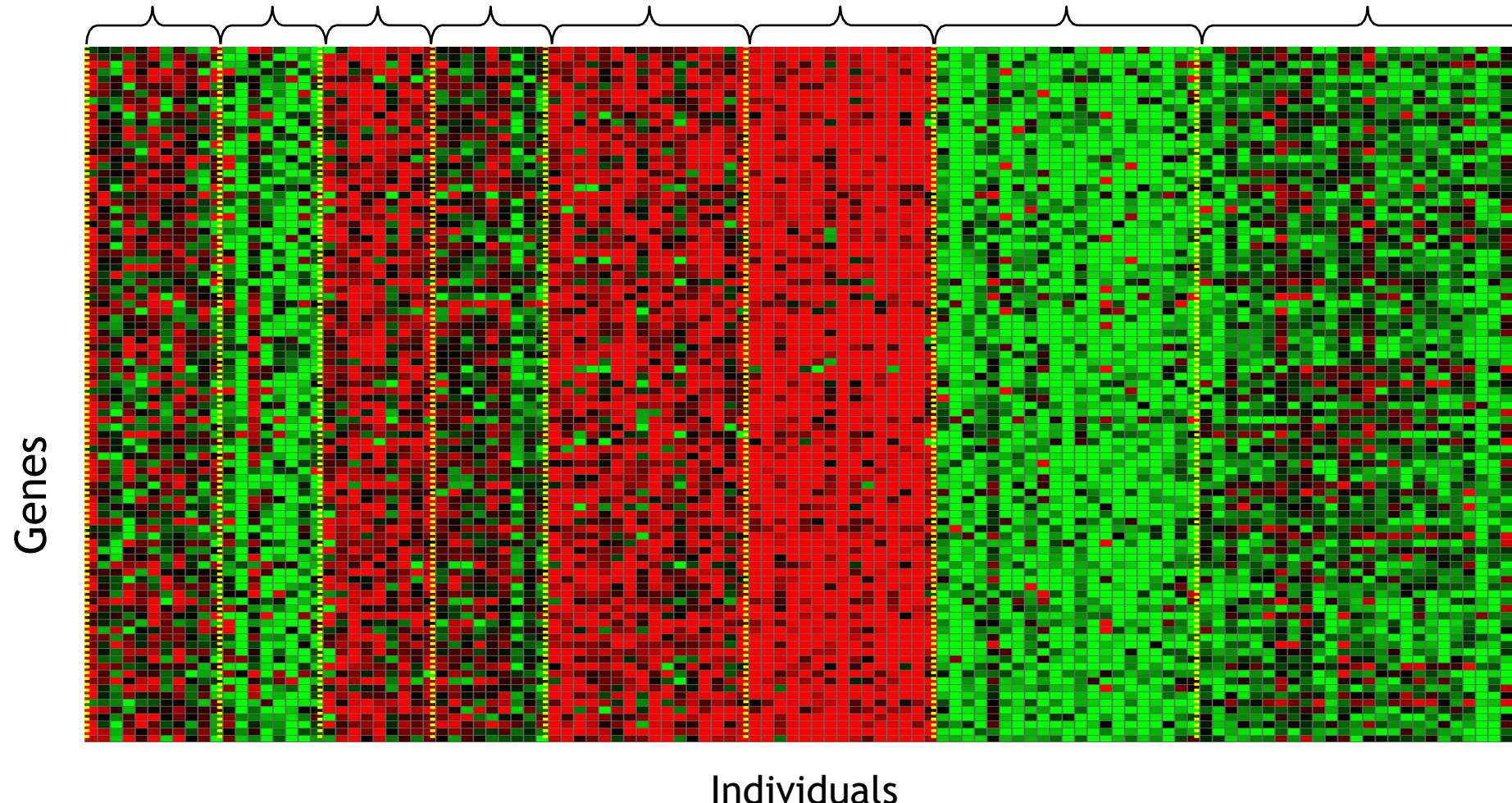
Example of a frequent itemset

{Diapers} → {Beer}

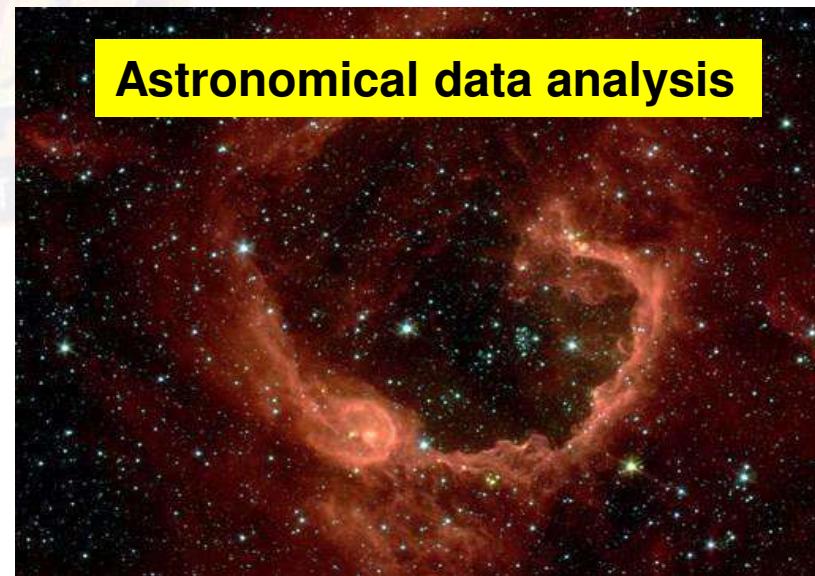
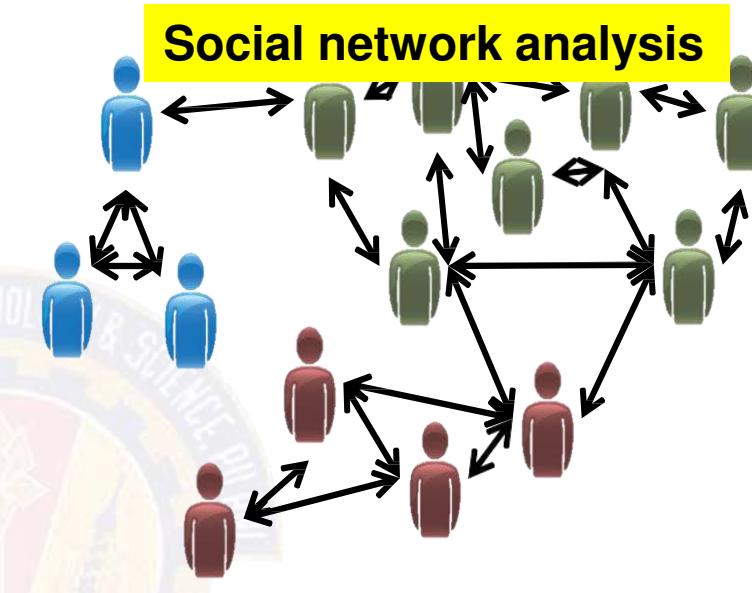
Example of an association rule

# Applications: Unsupervised Learning

Genomics application: Group individuals by genetic similarity



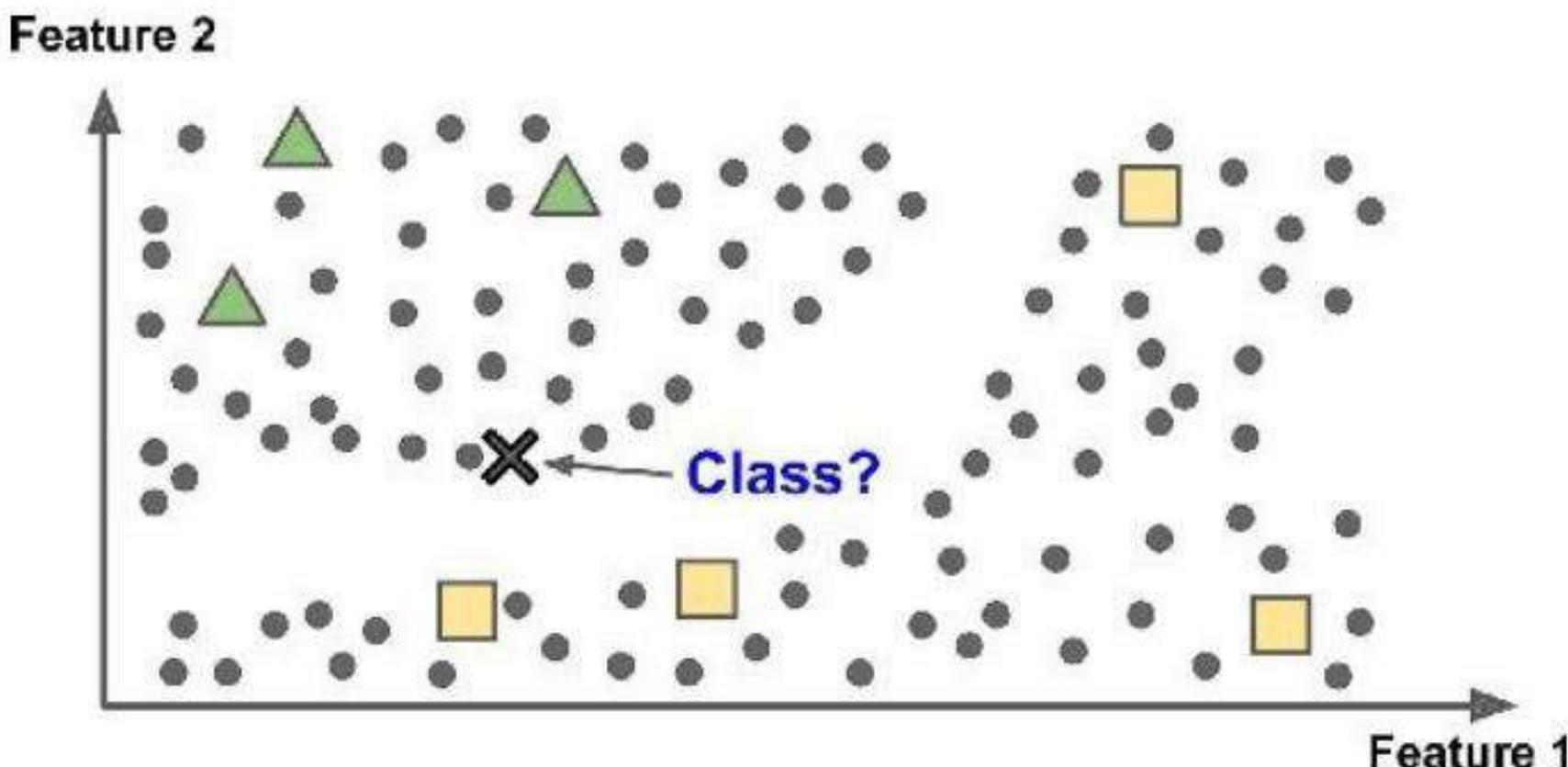
# Applications: Unsupervised Learning



# Semisupervised Learning

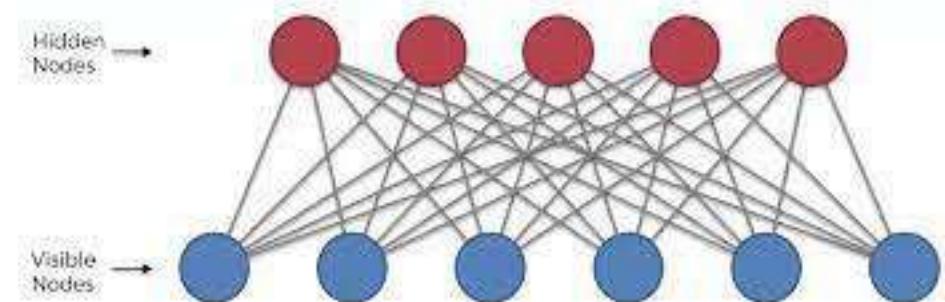
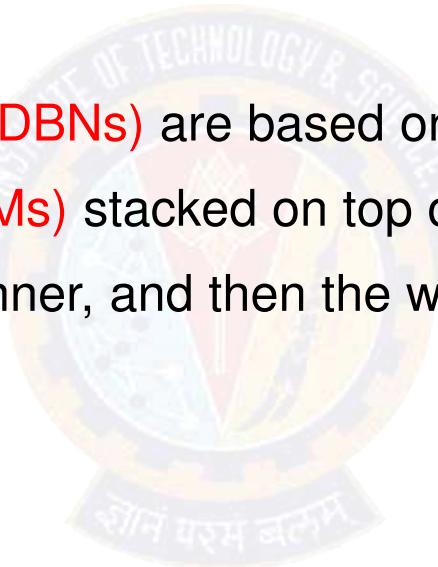
Partially labelled data – **few** labelled data and a **lot** of unlabelled data

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



# Semisupervised Learning Techniques

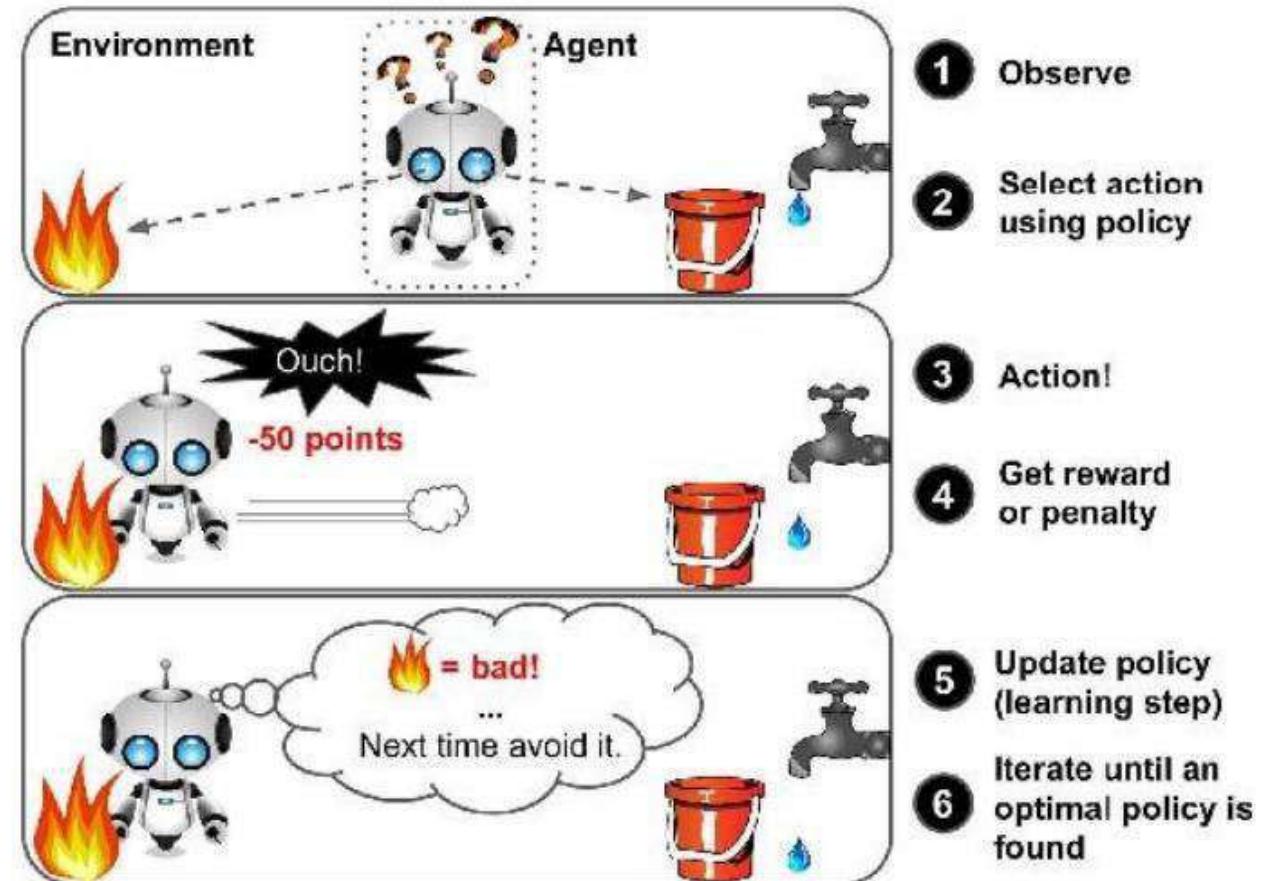
- Most semisupervised learning algorithms are combinations of unsupervised and supervised algorithms.
- For example, **deep belief networks (DBNs)** are based on unsupervised components called **restricted Boltzmann machines (RBMs)** stacked on top of one another. RBMs are trained sequentially in an unsupervised manner, and then the whole system is fine-tuned using supervised learning techniques.



# Reinforcement Learning

## A learning agent

- ❖ observes the *state* of the environment, select and perform actions
- ❖ gets +ve or -ve rewards in return
- ❖ learns the best strategy, *aka* a policy, to get the most reward over time.
  - policy is a mapping from states → actions
- Examples:
  - Game playing, e.g., *AlphaGo*
  - Robot in a maze
  - Balance a pole on your hand



# Types of Learning

## Based on how training data is used

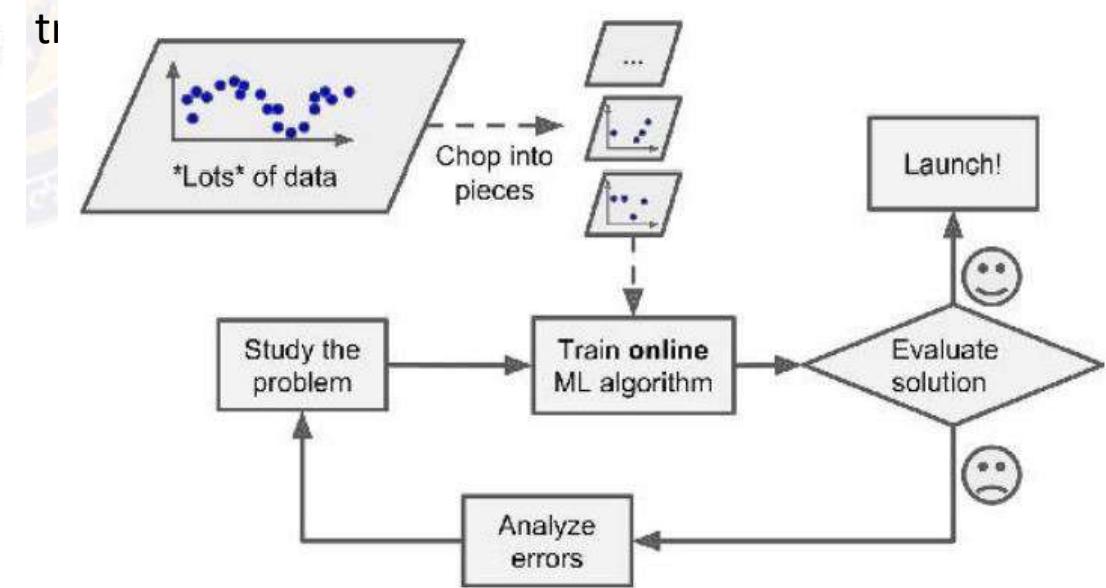
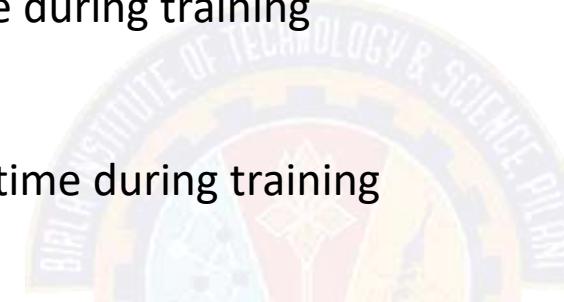
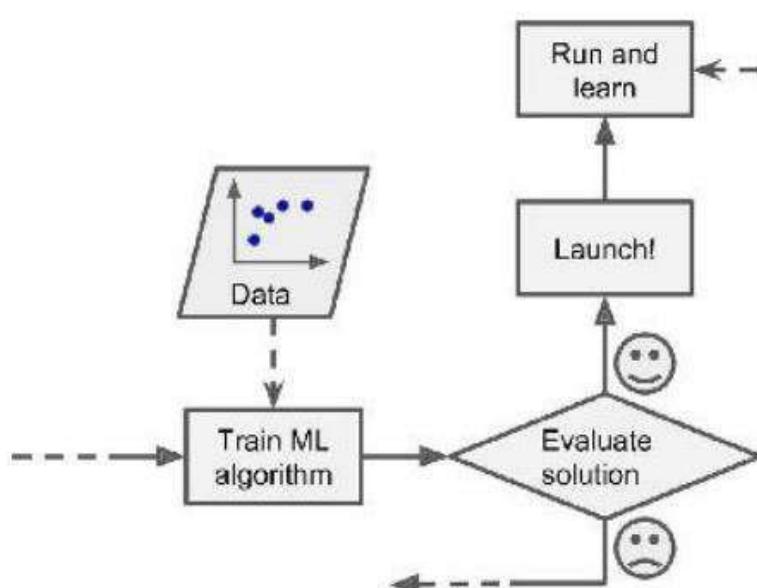
### ❖ Batch learning

- 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

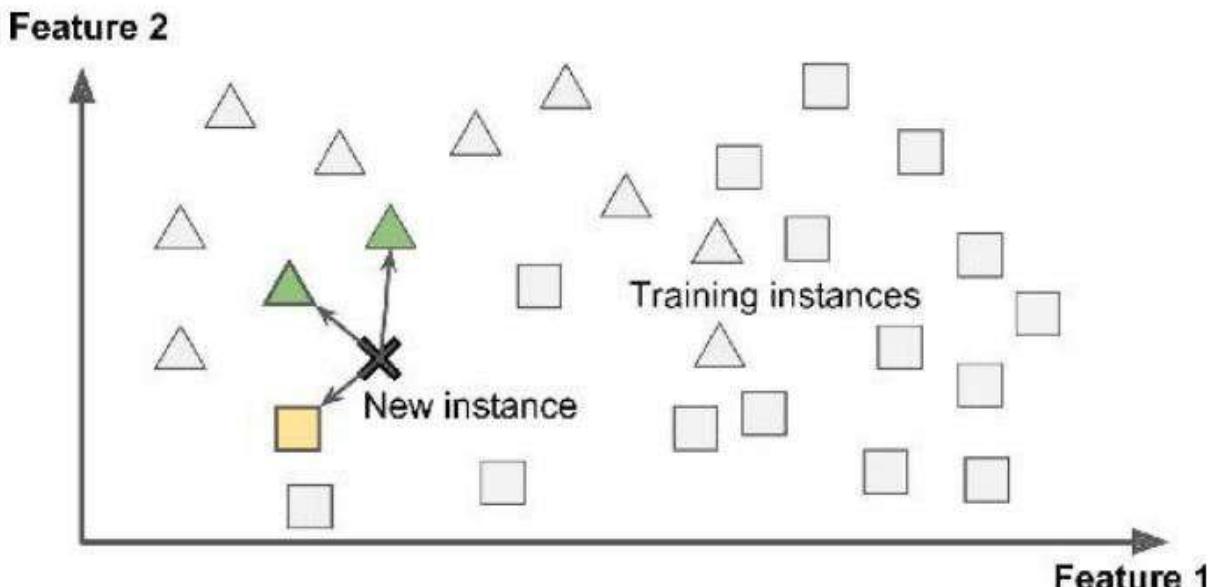


# Types of Learning

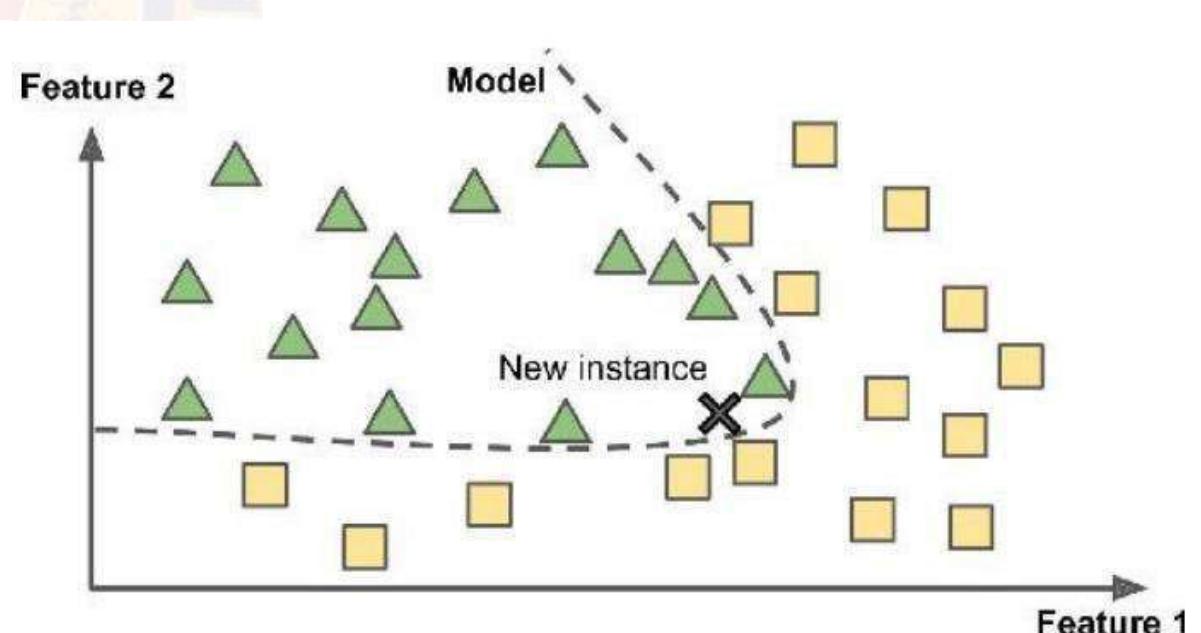
## Based on how training data is used

- 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

Instance Based



Model Based



# Challenges of ML

## ❖ Training Data

- Insufficient
- Non representative
- Poor Quality
- Irrelevant attributes



## ❖ Model Selection

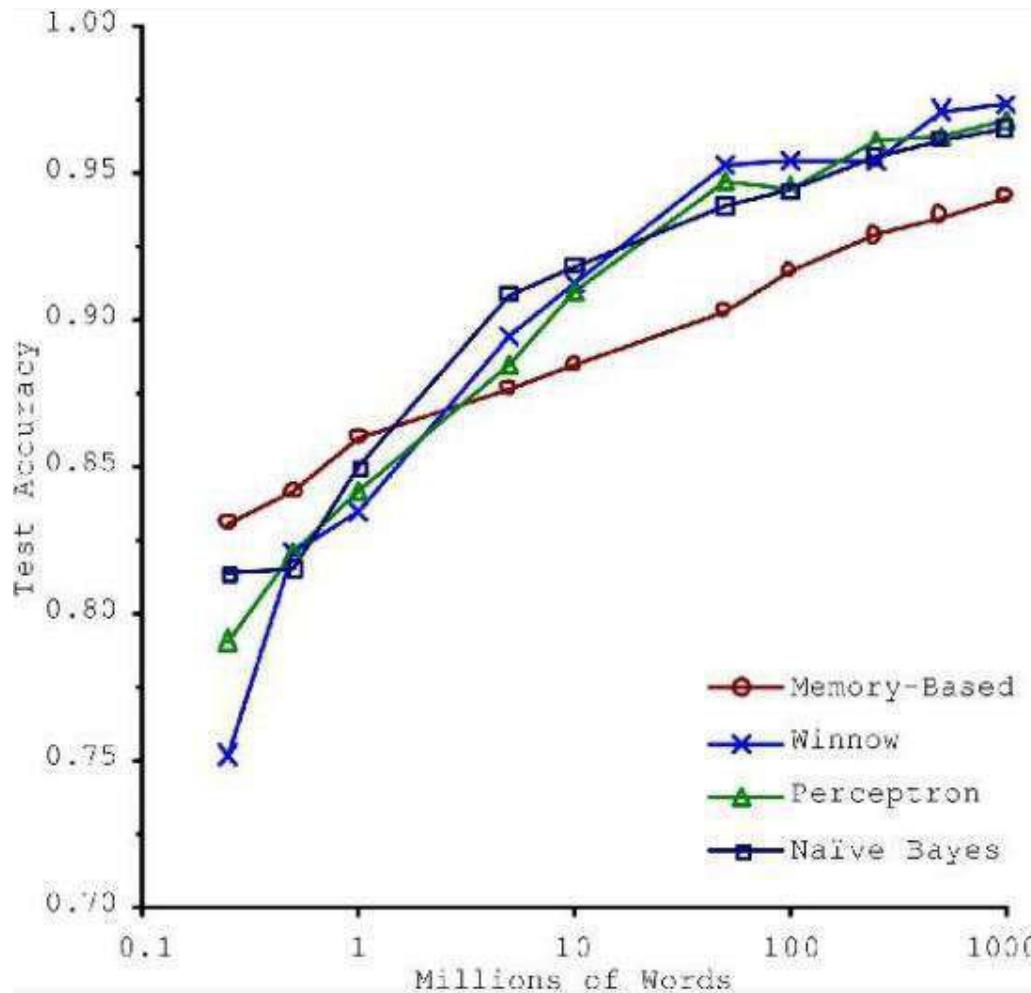
- Overfitting
- Underfitting

## ❖ Testing and Validation

- Hyperparameters

# Insufficient Training Data

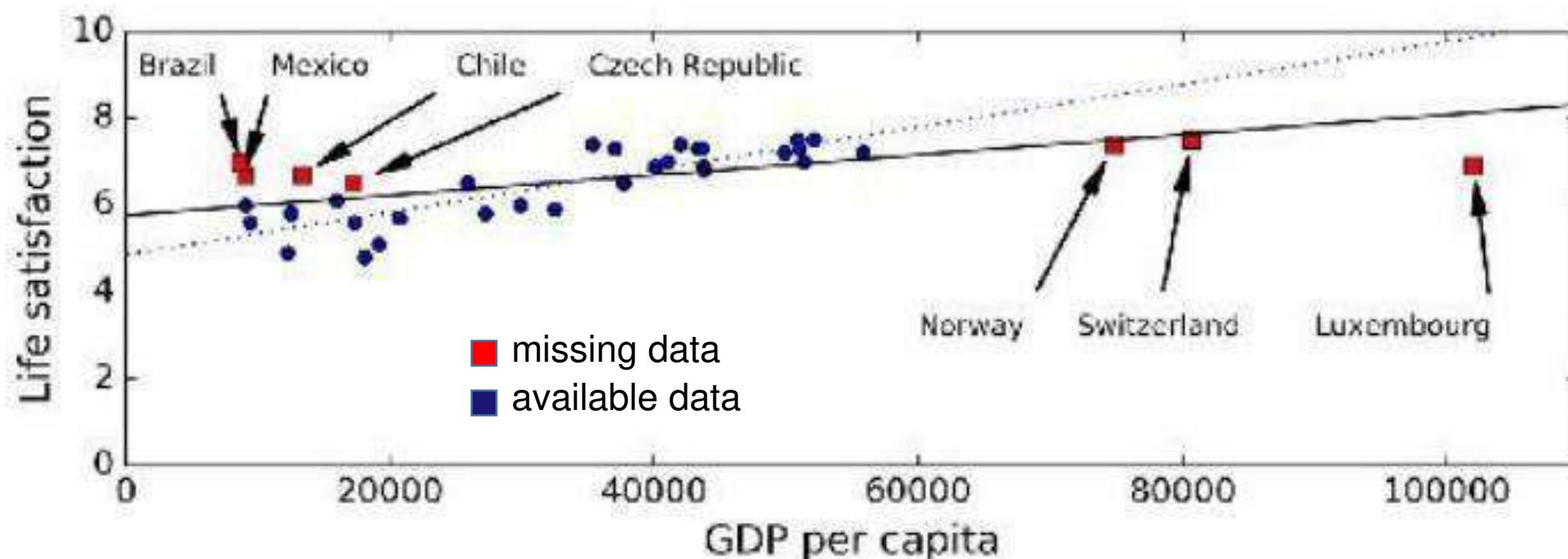
Consider trade-off Between Algorithm development & training data capture



# Non-representative Training Data

## Training Data be representative of the new cases we want to generalize

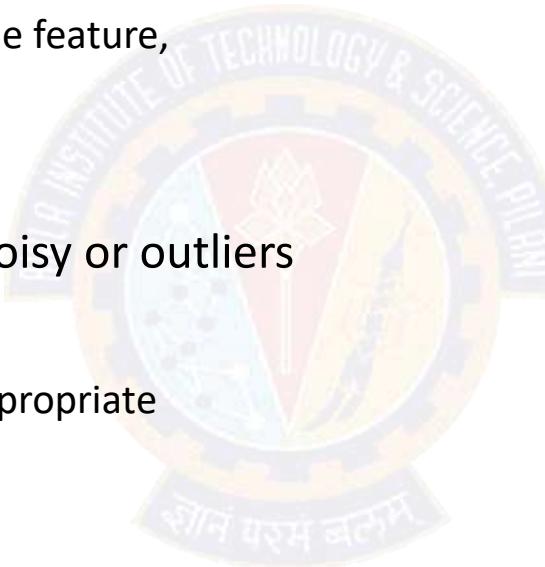
- Small sample size leads to sampling noise
  - Missing data over emphasizes the role of wealth on happiness
- If sampling process is flawed, even large sample size can lead to sampling bias



# Data Quality

## Cleaning often needed for improving data quality

- ❖ Some instances have missing features
  - e.g., 5% of customers did not specify their age
  - Ignore the instances all together or the feature,
  - fill in the missing values
  - Train multiple ML models
- ❖ Some instances can be erroneous, noisy or outliers
  - Human or machine generated
  - Identify, discard or fix manually, as appropriate



# Irrelevant Features

**Feature engineering needed for coming up with a good set of features**

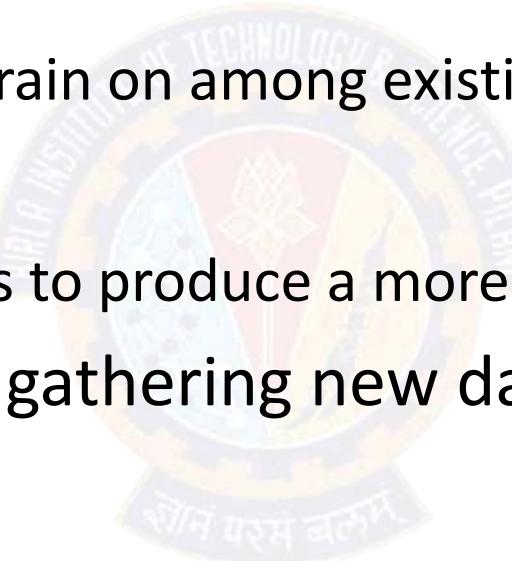
❖ Feature selection

- more useful features to train on among existing features.

❖ Feature extraction

- combine existing features to produce a more useful one.

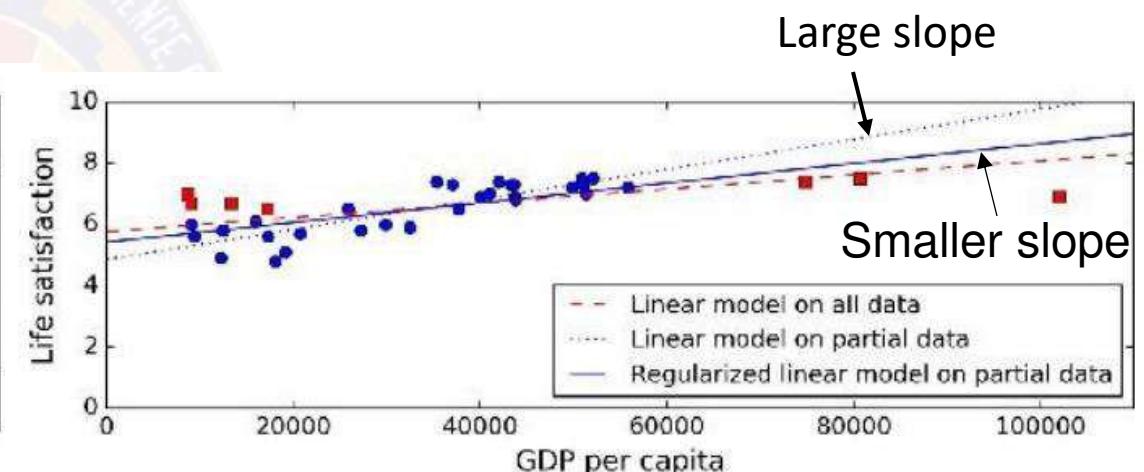
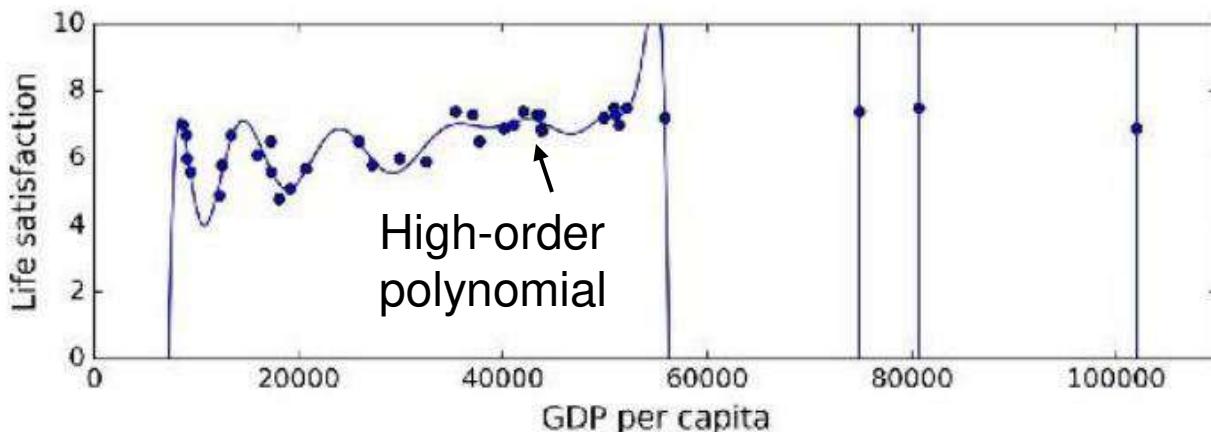
❖ Create new features by gathering new data



# Model Selection

## Overfitting or Underfitting

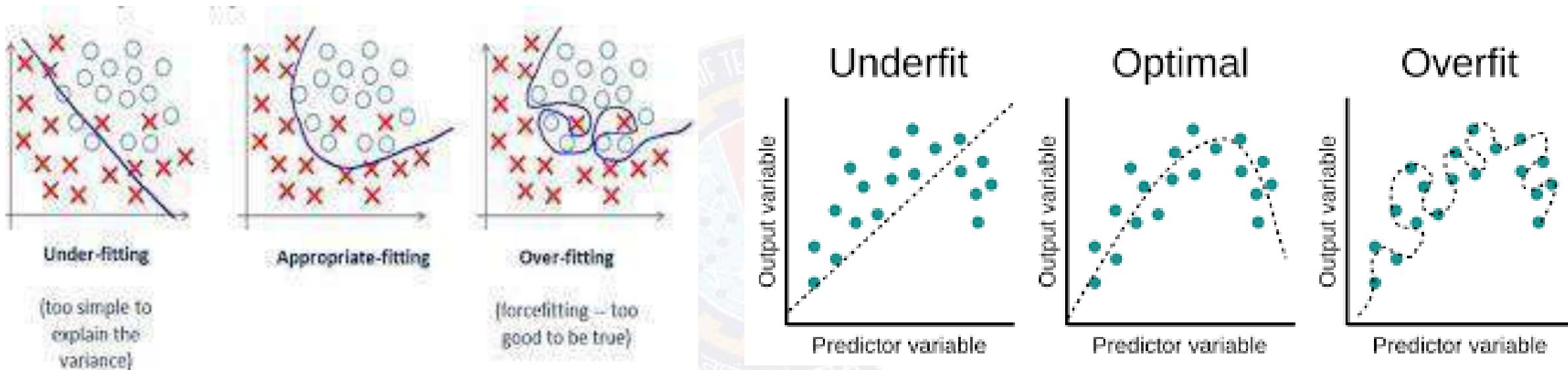
- Overfitting leads to high performance in training set but performs poorly on new data
  - e.g., a high-degree polynomial life satisfaction model that strongly overfits the training data
  - Small training set or sampling noise can lead to model following the noise than the underlying pattern in the dataset
  - Solution: Regularization



- Underfitting when the model is too simple to learn the underlying structure in the data
  - Select a more powerful model, with more parameters
  - Feed better features to the learning algorithm
  - Reduce regularization

# Model Selection

## Overfitting or Underfitting



# Testing and Validation

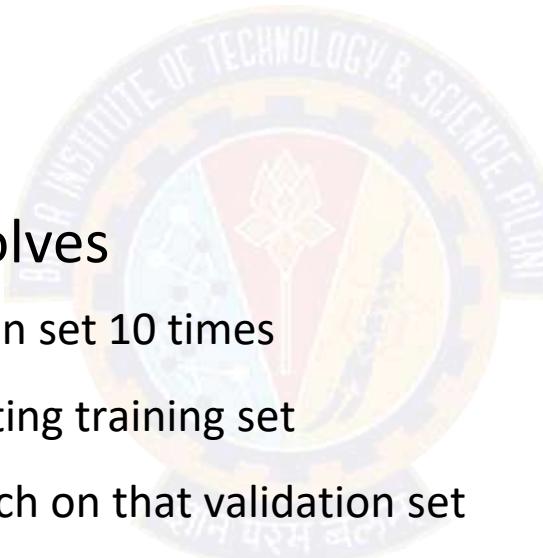
## Performance of ML algorithms is statistical / predictive

- ❖ Good ML algorithms need to work well on test data
  - But test data is often not accessible to the provider of the algorithm
- ❖ Common assumption is training data is representative of test data
- ❖ Randomly chosen subset of the training data is held out as validation set
  - *aka* dev set
- ❖ Once ML model is trained, its performance is evaluated on validation data
  - Expectation is ML model working well on validation set will work well on unknown test data
- ❖ Typically 20-30% of the data is randomly held out as validation data

# Cross Validation

## K-fold validation is often performed

- ❖ To reduce the bias of validation set selection process
- ❖ Often K is chosen as 10
  - *aka* 10 fold cross validation
- ❖ 10 fold cross validation involves
  - randomly selecting the validation set 10 times
  - model generation with 10 resulting training set
  - Evaluate the performance of each on that validation set
  - averaging the performance over the validation sets



# Choice of Hyper parameters

**Modern ML models often use a lot of model parameters**

- ❖ Known as *hyper parameters*
- ❖ Model performance depends on choice of parameters
- ❖ Each parameter can assume a number of values
  - Real numbers or *categories*
- ❖ Exponential number of hyper parameter combinations possible
- ❖ Best model correspond to best cross validation performance over the set of hyper parameter combinations
- ❖ Expensive to perform
- ❖ Some empirical frameworks available for hyper parameter optimization



**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 2(29<sup>th</sup> July,2023)**

# Session 2 – 29<sup>th</sup> July 2023

2

End-to-end Machine Learning: Framing the ML Problem. Data Types, Pre-processing, Visualization and Analysis

T1: Chapter 2  
T2: Chapter 2-3

## Framing the ML Problem

# Key Elements of a Machine Learning Project

- Framing a machine learning problem
- Data types and representation
- Data Pre processing
- Data Visualization and Analysis
- Feature Engineering
- Model Building and testing



# An Example Problem Statement

## Start with the Big Picture

1. Build a model of housing prices using the census data.

- Data attributes <population, median income, median housing price, ....> and so on for each district
- Districts are the smallest geographical unit (population ~600 – 3000)

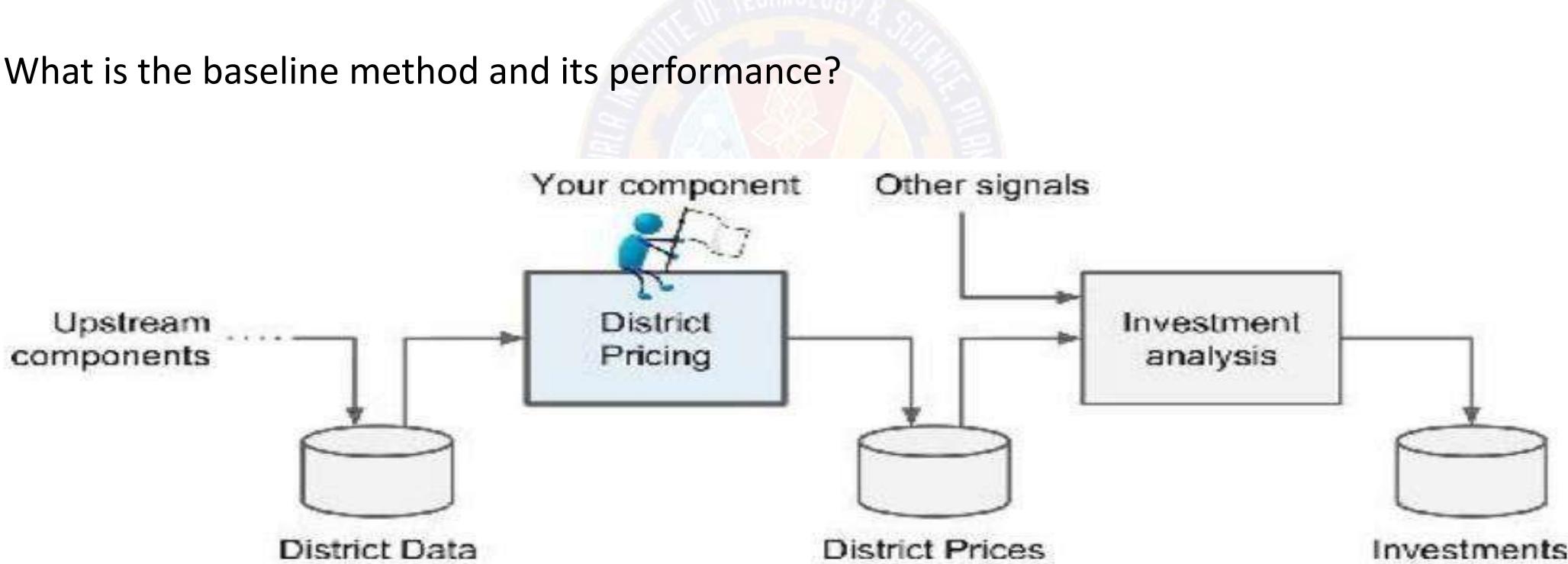
2. The model to predict the median housing price in any district, given all the other metrics.

3. Goodness of the model is determined by how close the model output is w.r.t. actual price for unseen district data

# Framing the Problem

## Understand the Business Objective and Context

- What is the expected usage and benefit?
  - ✓ impacts the choice of algorithms, goodness measure, and effort in lifecycle management of the model
- What is the baseline method and its performance?



# Choice of Model

## Choose between Different techniques

- Supervised or unsupervised? Prediction or classification? Online Vs. Batch? Instance-based or Model-based?
- Analyze the dataset
  - Each instance comes with the expected output, i.e., the district's median housing price.
    -  supervised
  - Goal is to predict a real valued price based on multiple variables like population, income etc.
    -  regression
  - Output is based on input data at rest, not rapidly changing data rapidly.
  - Dataset small enough to fit in memory
    -  batch
- So, it's a **supervised multivariate batch regression** problem

# Choice of Performance Metrics

## A typical Choice for Regression: Root Mean Square Error (RMSE)

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

- $h$  is the model,  $\mathbf{X}$  is the training dataset,  $m$  is number of instances,  $\mathbf{x}^{(i)}$  is  $i$ -th instance,  $y^{(i)}$  is the actual price for the  $i$ -th instance.

## Mean Absolute Error (MAE)

$$\text{MAE}(\mathbf{X}, h) = \frac{1}{m} \sum_{i=1}^m |h(\mathbf{x}^{(i)}) - y^{(i)}|$$

- MAE is preferred for a large number of outliers. aka  $L_1$  norm or Manhattan distance/norm.

## General Form

$$\| \mathbf{v} \|_k = (\| v_0 \|^k + \| v_1 \|^k + \dots + \| v_n \|^k)^{\frac{1}{k}}$$

# Stepping Back

## Check the Assumptions

- Verify that the downstream module actually uses real-valued prices rather than post processing them into say, categories, e.g “cheap,” “medium,” or “expensive”)
- If not, the problem should have been framed as a classification task, not a regression task.
- Such potential hazards need to be checked early in the design rather than finding out from deployed systems

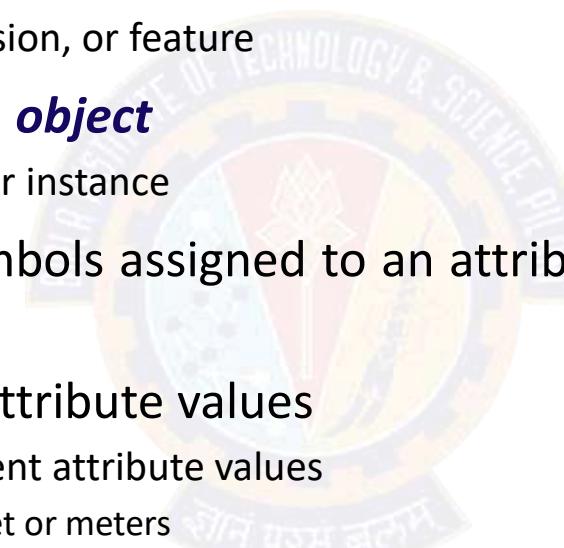
# Topic 2

## Data Types

# What is Data?

## Collection of data objects and attributes

- An **attribute** is a property or characteristic of an object
  - ❖ Examples: eye color of a person, temperature, etc.
  - ❖ aka variable, field, characteristic, dimension, or feature
- A collection of attributes describe an **object**
  - ❖ aka record, point, case, sample, entity, or instance
- **Attribute values** are numbers or symbols assigned to an attribute for a particular object
- Distinction between attributes and attribute values
  - ❖ Same attribute can be mapped to different attribute values
    - ❖ Example: height can be measured in feet or meters
    - ❖ Different attributes can be mapped to the same set of values
      - ❖ Example: Attribute values for ID and age are integers
- But properties of attribute can be different than the properties of the values used to represent the attribute



Attributes

Objects

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

# Discrete and Continuous Attributes

## ■ Discrete Attribute

- Has only a finite or countably infinite set of values
  - Examples: zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables.
- Note: *binary attributes* are a special case of discrete attributes

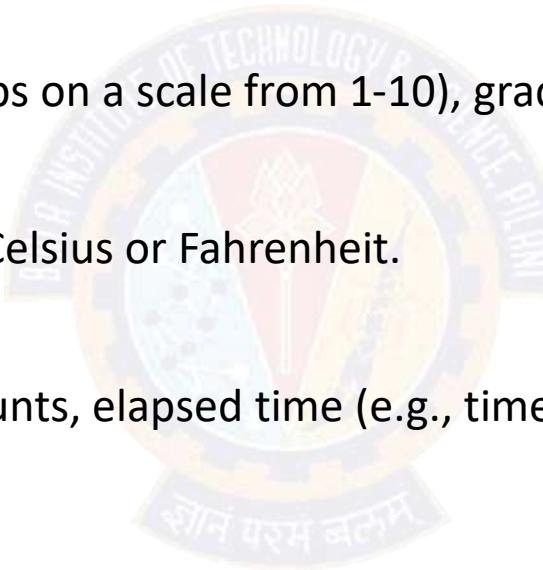
## ■ Continuous Attribute

- Has real numbers as attribute values
  - Examples: temperature, height, or weight.
- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.

# Types and properties of Attributes

## Types

- Nominal
  - ID numbers, eye color, zip codes
- Ordinal
  - rankings (e.g., taste of potato chips on a scale from 1-10), grades, height {tall, medium, short}
- Interval
  - calendar dates, temperatures in Celsius or Fahrenheit.
- Ratio
  - temperature in Kelvin, length, counts, elapsed time (e.g., time to run a race)



## Properties

- Distinctness:  $= \neq$
- Order:  $< >$
- Differences are meaningful :  $+ -$
- Ratios are meaningful :  $* /$

# Important Characteristics of Data

- Dimensionality (number of attributes)

- ❖ High dimensional data brings a number of challenges

- Sparsity

- ❖ Only presence counts

- Resolution

- ❖ Patterns depend on the scale

- Size

- Type of analysis may depend on size of data



# Types of data sets

## 1. Record

- Data Matrix
- Document Data
- Transaction Data

## 2. Graph

- World Wide Web
- Molecular Structures

## 3. Ordered

- Spatial Data
- Temporal Data
- Sequential Data
- Genetic Sequence Data



# Types of data sets

## ■ Record

- Data Matrix
- Document Data
- Transaction Data

Tid	Refund	Marital Status	Taxable 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

(a) Record data.

TID	ITEMS
1	Bread, Soda, Milk
2	Beer, Bread
3	Beer, Soda, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Soda, Diaper, Milk

(b) Transaction data.

Projection of x Load	Projection of y Load	Distance	Load	Thickness
10.23	5.27	15.22	27	1.2
12.65	6.25	16.22	22	1.1
13.54	7.23	17.34	23	1.2
14.27	8.43	18.45	25	0.9

(c) Data matrix.

season	timeout	lost	win	game	score	ball	play	coach	team
Document 1	3	0	5	0	2	6	0	2	0
Document 2	0	7	0	2	1	0	0	3	0
Document 3	0	1	0	0	1	2	2	0	3

(d) Document-term matrix.

# Types of data sets

- **Ordered**

- Spatial Data
- Temporal Data
- Sequential Data
- Genetic Sequence Data

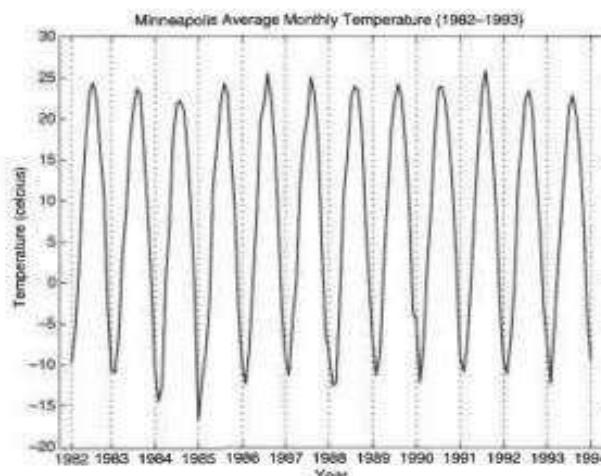
Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A, B) (t2: C, D) (t5: A, E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)

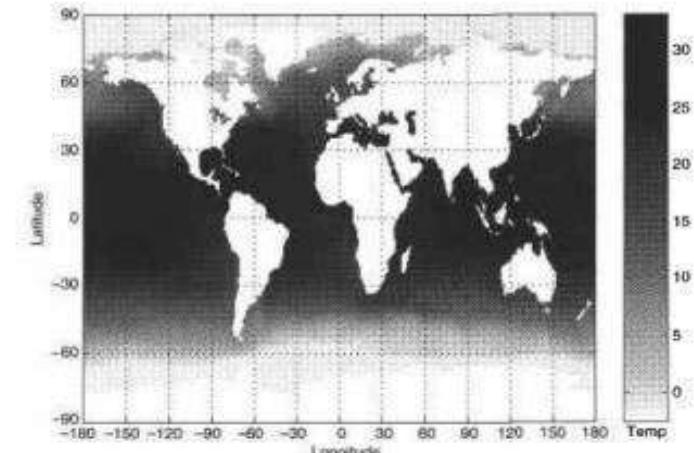
(a) Sequential transaction data.

GGTTCCGCCCTTCAGCCCCGGCG  
CGCAGGGCCCGCCCCGGCGCCGTC  
GAGAAGGGCCCGCCTGGCGGGCG  
GGGGGAGGCGGGGCCGCCCAGC  
CCAACCGAGTCCGACCAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGCAGCAGCGACAG  
GCCAAGTAGAACACCGCAAGCGC  
TGGGCTGCCTGCTGCGACCAGGG

(b) Genomic sequence data.

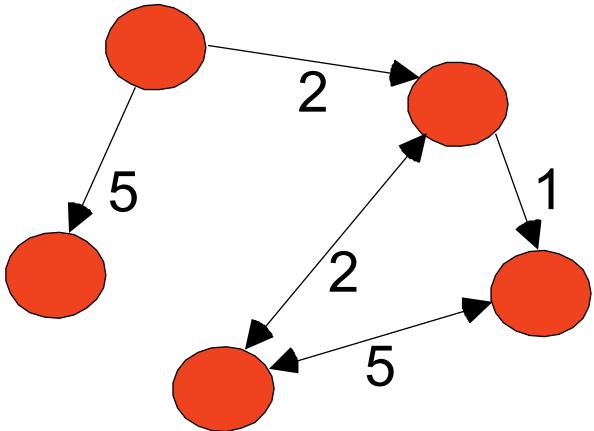
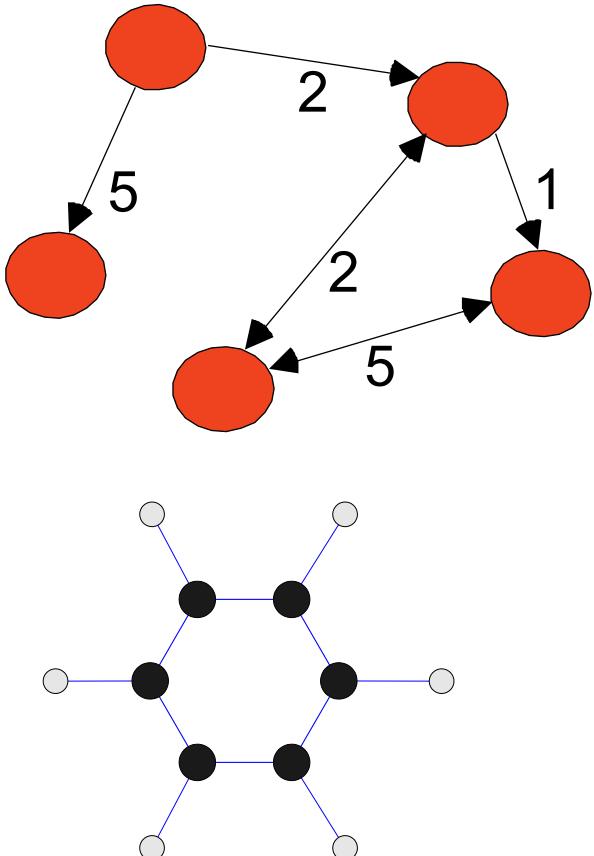


(c) Temperature time series.



(d) Spatial temperature data.

# Graph Data



## Useful Links:

- [Bibliography](#)
- Other Useful Web sites
  - [ACM SIGKDD](#)
  - [KDnuggets](#)
  - [The Data Mine](#)

## Knowledge Discovery and Data Mining Bibliography

(Gets updated frequently, so visit often!)

- [Books](#)
- [General Data Mining](#)

## Book References in Data Mining and Knowledge Discovery

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Ithurusamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/the MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993. Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support)", John Wiley & Sons, 1997.

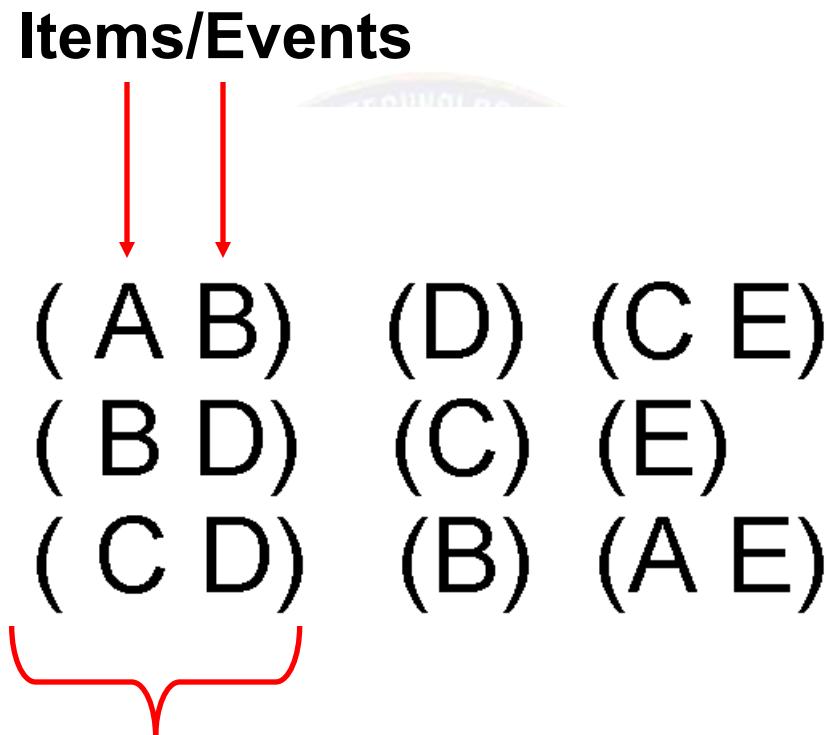
## General Data Mining

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for Knowledge Discovery in Databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

# Ordered Data

## Sequence of transactions



**An element of  
the sequence**

# Ordered Data

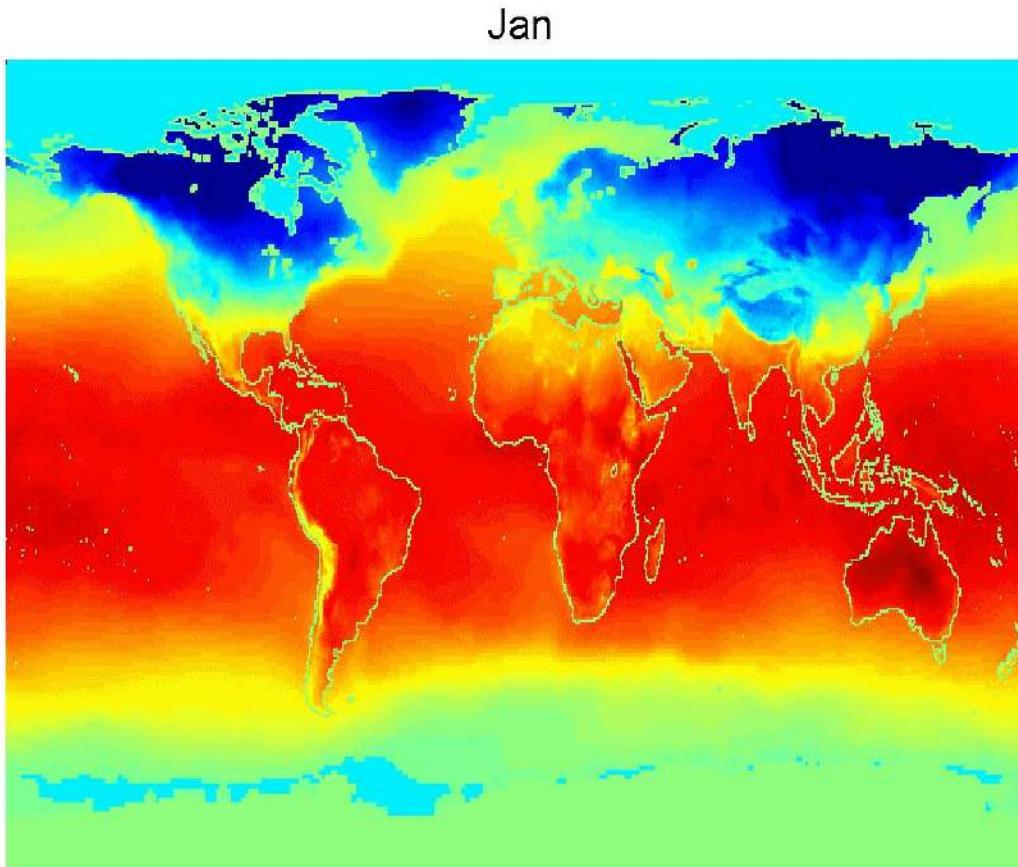
## Genomic sequence data

GGTTCCGCCTTCAGCCCCGCGCC  
CGCAGGGCCCGCCCCGCGGCCGTC  
GAGAAGGGCCCGCCTGGCGGGCG  
GGGGGAGGCAGGGCCGCCGAGC  
CCAACCGAGTCCGACCAAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGCAGCAGCGGACAG  
GCCAAGTAGAACACGCGAAGCGC  
TGGGCTGCCTGCTGCGACCAAGGG

# Ordered Data

## SpatioTemporal Data

Average Monthly Temperature of land and ocean



## Data Pre-processing

# In this segment

## Data Preprocessing

### 1. Data Quality

- Noise, Outlier, Missing attribute, Duplicate record

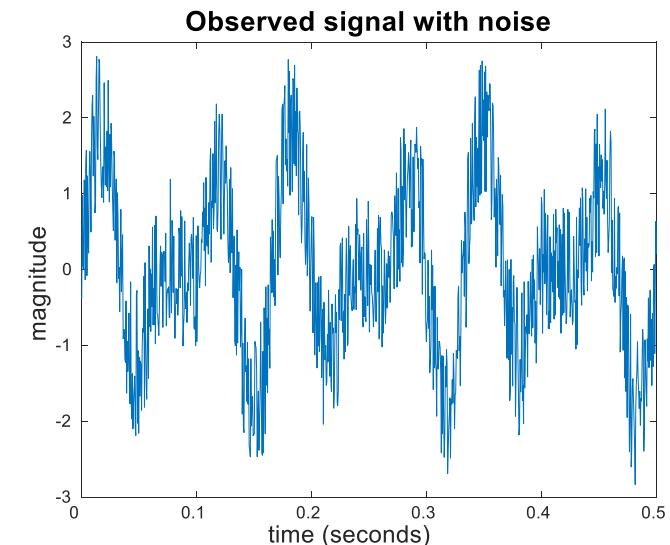
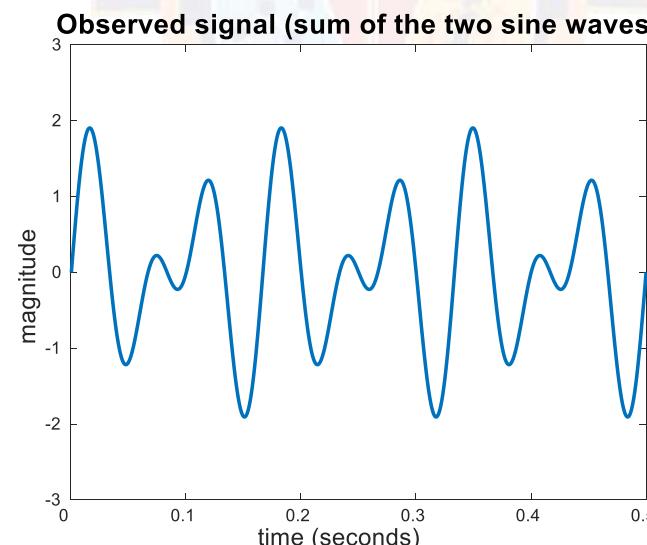
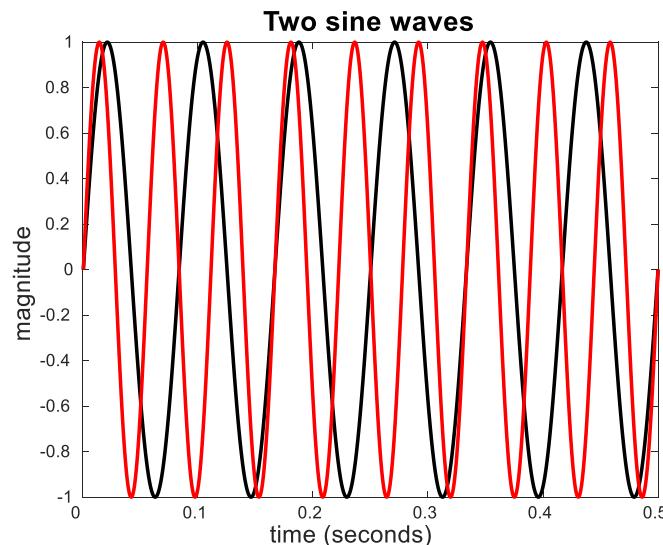
### 2. Data Transformation

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



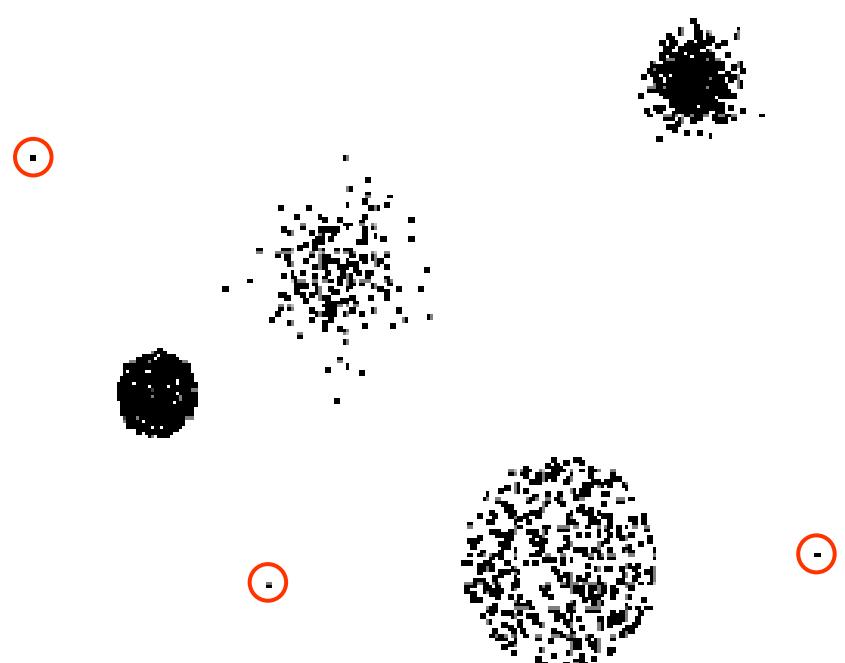
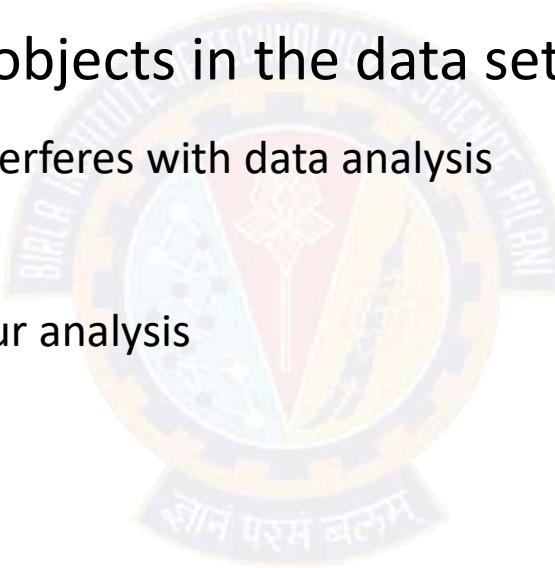
# Noise

- For objects, noise is an extraneous object
- For attributes, noise refers to modification of original values
  - Examples: distortion of a person's voice when talking on a poor phone and "snow" on television screen
  - The figures below show two sine waves of the same magnitude and different frequencies, the waves combined, and the two sine waves with random noise
    - The magnitude and shape of the original signal is distorted



# Outliers

- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
  - **Case 1:** Outliers are noise that interferes with data analysis
  - **Case 2:** Outliers are the goal of our analysis
    - Credit card fraud
    - Intrusion detection



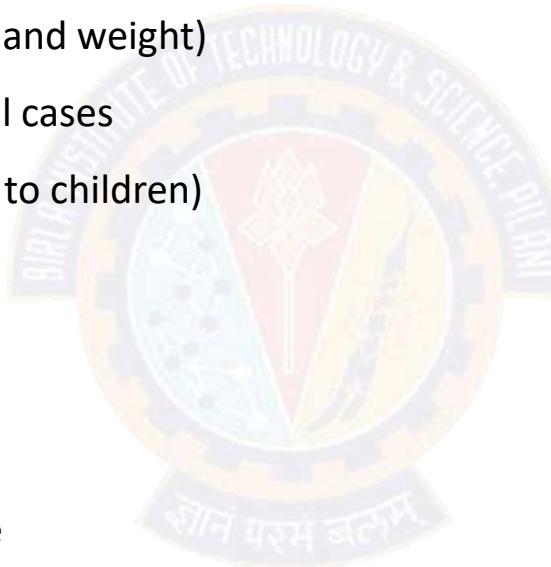
# Missing Values

## ■ Reasons for missing values

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

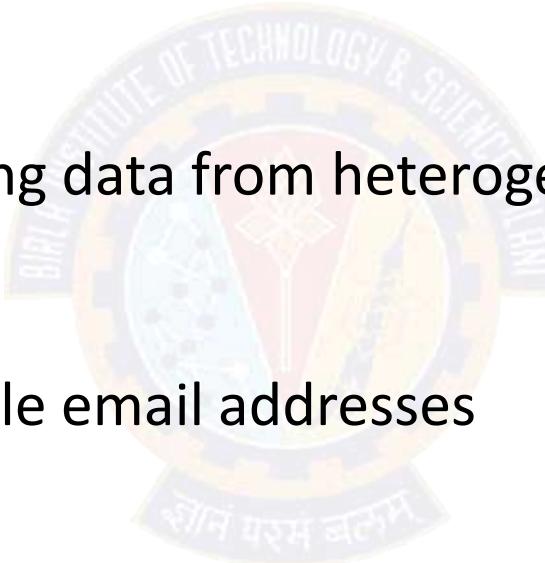
## ■ Handling missing values

- Eliminate data objects or variables
- Estimate missing values
  - Example: time series of temperature
  - Example: census results
- Ignore the missing value during analysis



# Duplicate Data

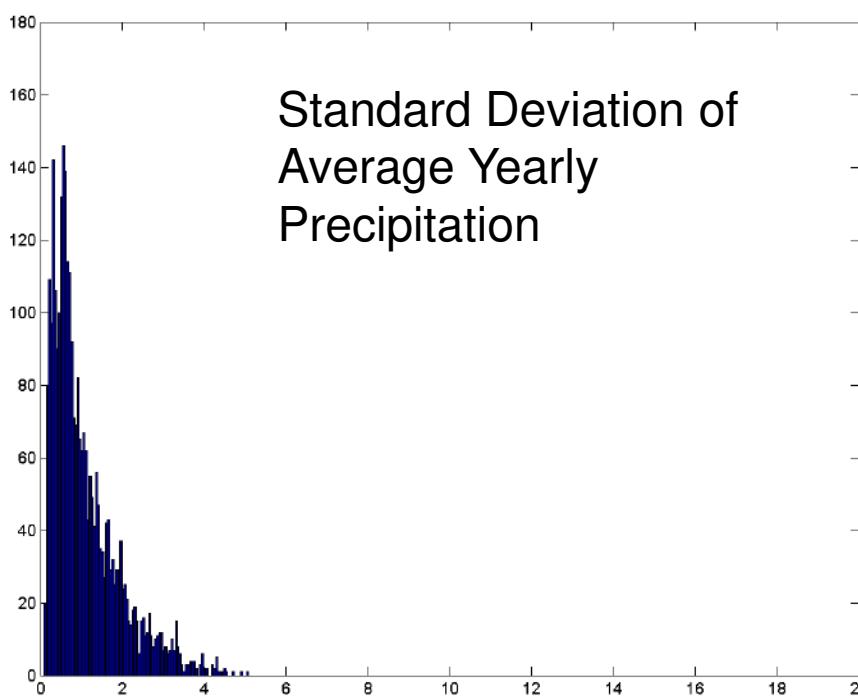
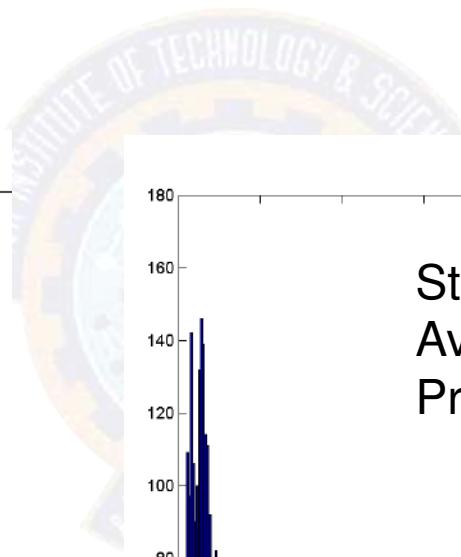
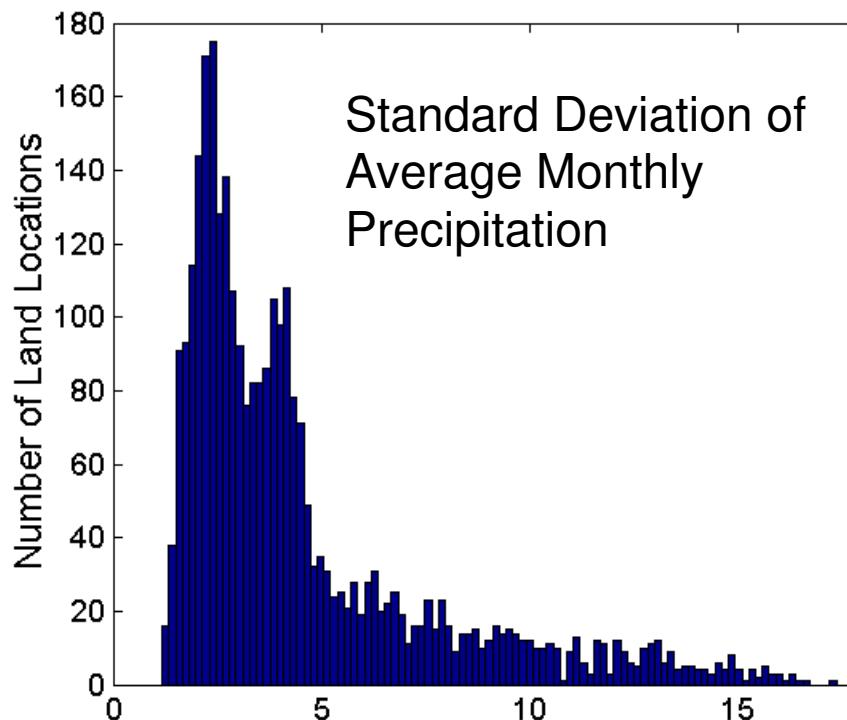
- Data set may include data objects that are duplicates, or almost duplicates of one another
  - ❖ Major issue when merging data from heterogeneous sources
- Examples:
  - Same person with multiple email addresses
- Data cleaning
  - Process of dealing with duplicate data issues



# Aggregation

Combining two or more attributes (or objects) into a single attribute (or object)

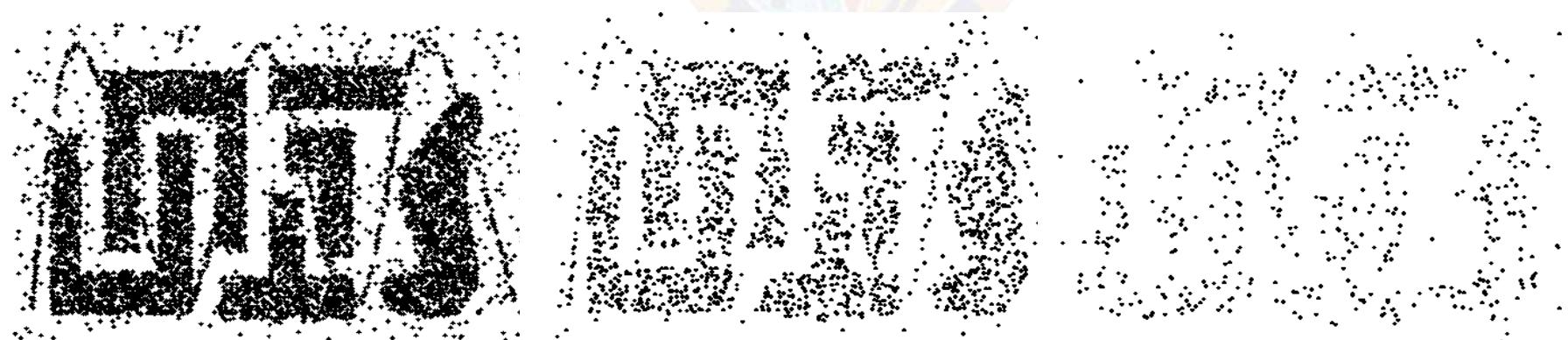
- Purpose
  - Data reduction
  - Change of scale
  - More “stable” data



# Sampling

## Primary technique used for data reduction

- It is often used for both the preliminary investigation of the data and the final data analysis.
- Statisticians often sample because *obtaining* the entire set of data of interest is too expensive or time consuming.
- Sampling is typically used in machine learning because *processing* the entire set of data of interest may be too expensive or time consuming.
- Key Principle
  - Using a sample will work almost as well as using the entire data set, if the sample is *representative*
  - A sample is *representative* if it has approximately the same properties as the original dataset



8000 points

2000 Points

500 Points

# Sampling ...

- **The key principle for effective sampling is the following:**

- Using a sample will work almost as well as using the entire data set, if the sample is *representative*
  - A sample is *representative* if it has approximately the same properties (of interest) as the original set of data

- **Simple Random Sampling**

- There is an equal probability of selecting any particular item
  - Sampling without replacement
    - As each item is selected, it is removed from the population
  - Sampling with replacement
    - Objects are not removed from the population as they are selected for the sample.
    - In sampling with replacement, the same object can be picked up more than once

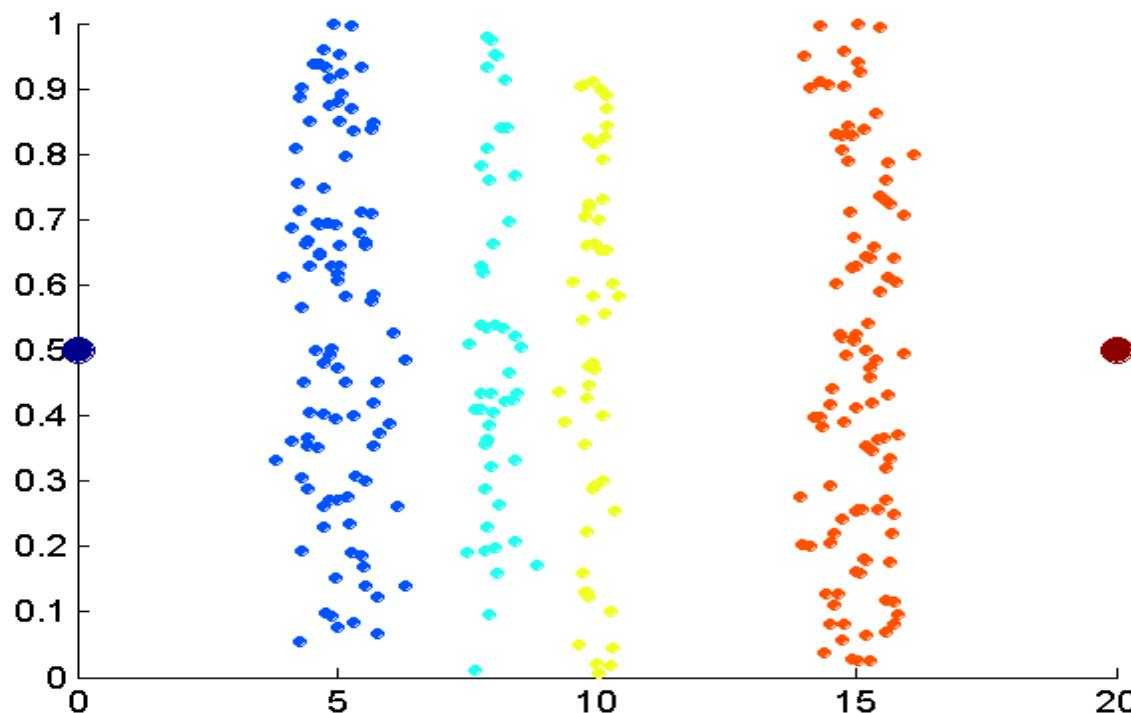
- **Stratified sampling**

- Split the data into several partitions; then draw random samples from each partition

# Discretization

**Process of converting a continuous attribute into an ordinal attribute**

- A potentially infinite number of values are mapped into a small number of categories
- Discretization is used in both unsupervised and supervised settings

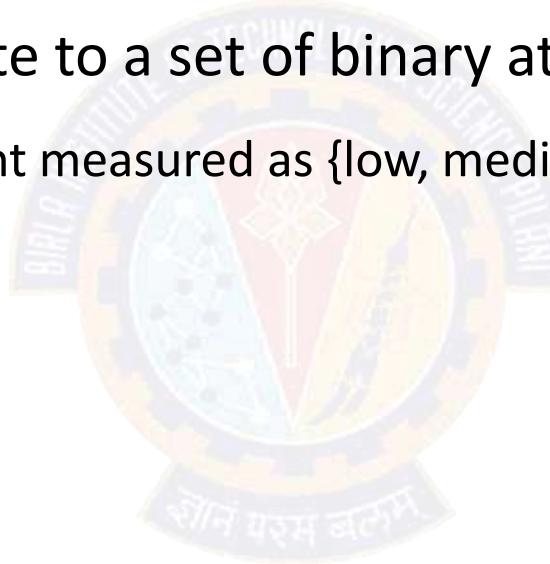


Data consists of four groups of points and two outliers. Data is one-dimensional, but a random y component is added to reduce overlap.

# Binarization

**Maps a continuous or categorical attribute into one or more binary variables**

- Often convert a continuous attribute to a categorical attribute and then convert a categorical attribute to a set of binary attributes
  - Examples: eye color and height measured as {low, medium, high}



# Attribute Transformation

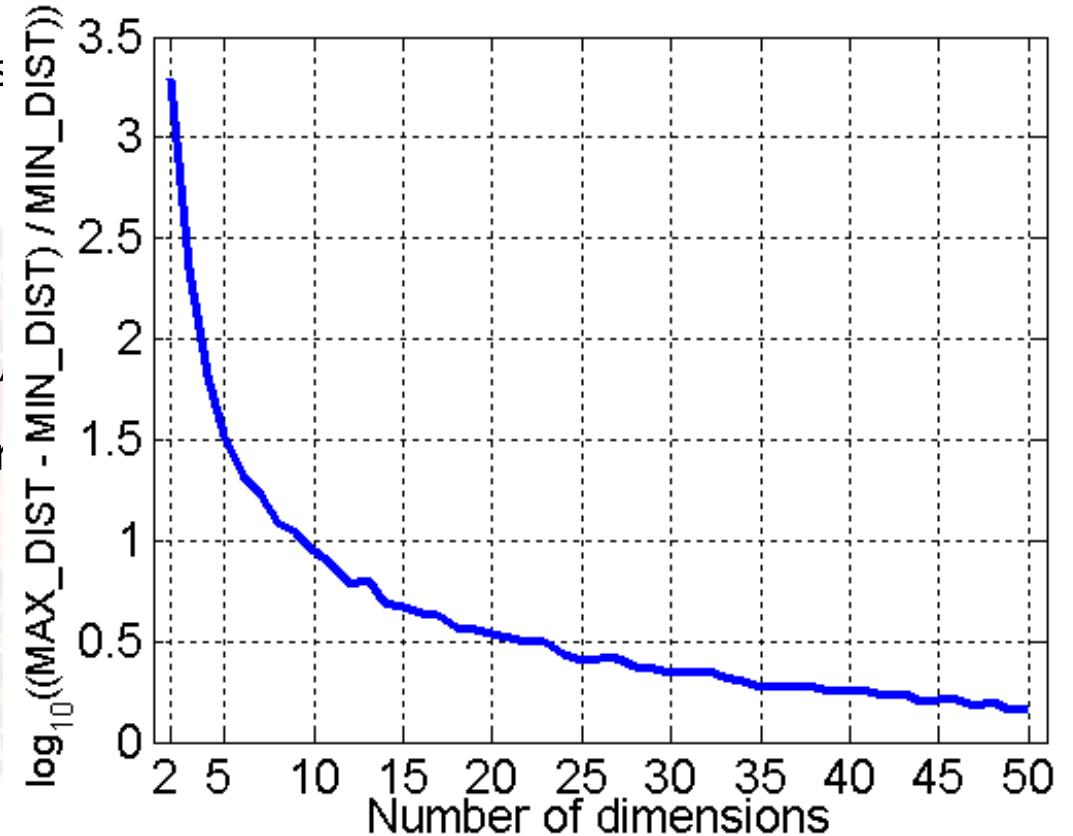
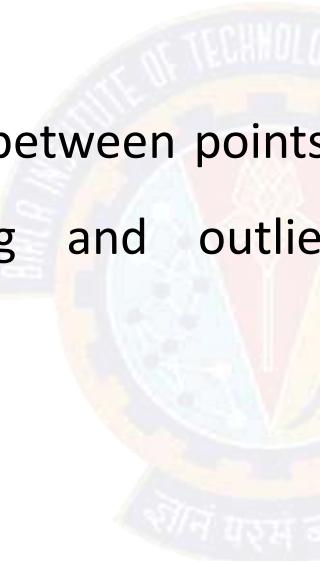
**Maps the entire set of values of a given attribute to a new set of replacement values**

- **Each original value can be identified with one of the new values**

- Simple functions:  $x^k$ ,  $\log(x)$ ,  $e^x$ ,  $|x|$
- Normalization
  - Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
  - Take out unwanted, common signal, e.g., seasonality
  - In statistics, standardization refers to subtracting off the means and dividing by the standard deviation

# Curse of Dimensionality

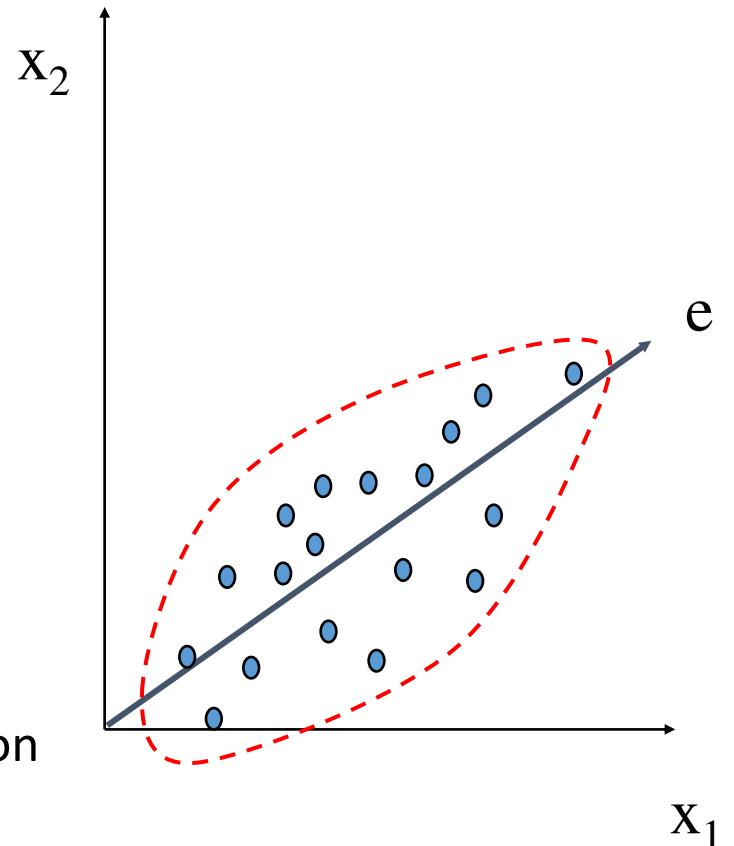
- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points, which are critical for clustering and outlier detection, become less meaningful
- Randomly generate 500 points
- Compute difference between max and min distance between any pair of points



# Dimensionality Reduction

## Purpose

- Avoid curse of dimensionality
- Reduce amount of time and memory required by ML algorithms
- Allow data to be more easily visualized
- May help to eliminate irrelevant features or reduce noise
- Popular Techniques
  - Principal Components Analysis (PCA)
  - Singular Value Decomposition
  - Find a projection that captures the largest amount of variation in data



# Feature Subset Selection

## Another way to reduce dimensionality of data

### ■ Redundant features

- Duplicate much or all of the information contained in one or more other attributes
- Example: purchase price of a product and the amount of sales tax paid

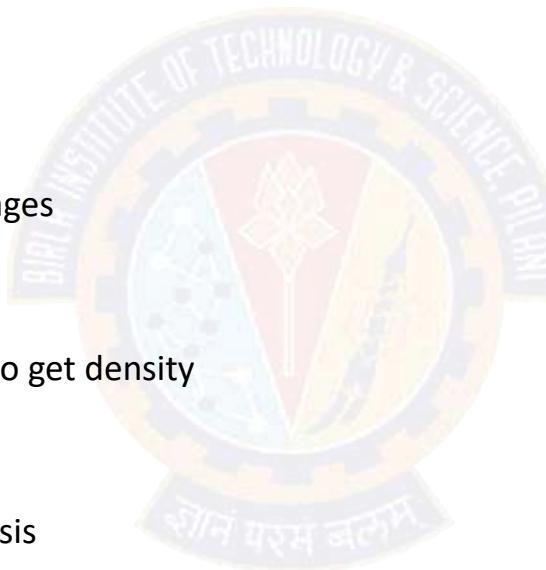
### ■ Irrelevant features

- Contain no information that is useful for the ML task at hand
- Example: students' ID is often irrelevant to the task of predicting students' GPA
- Many techniques developed, especially for classification

# Feature Creation

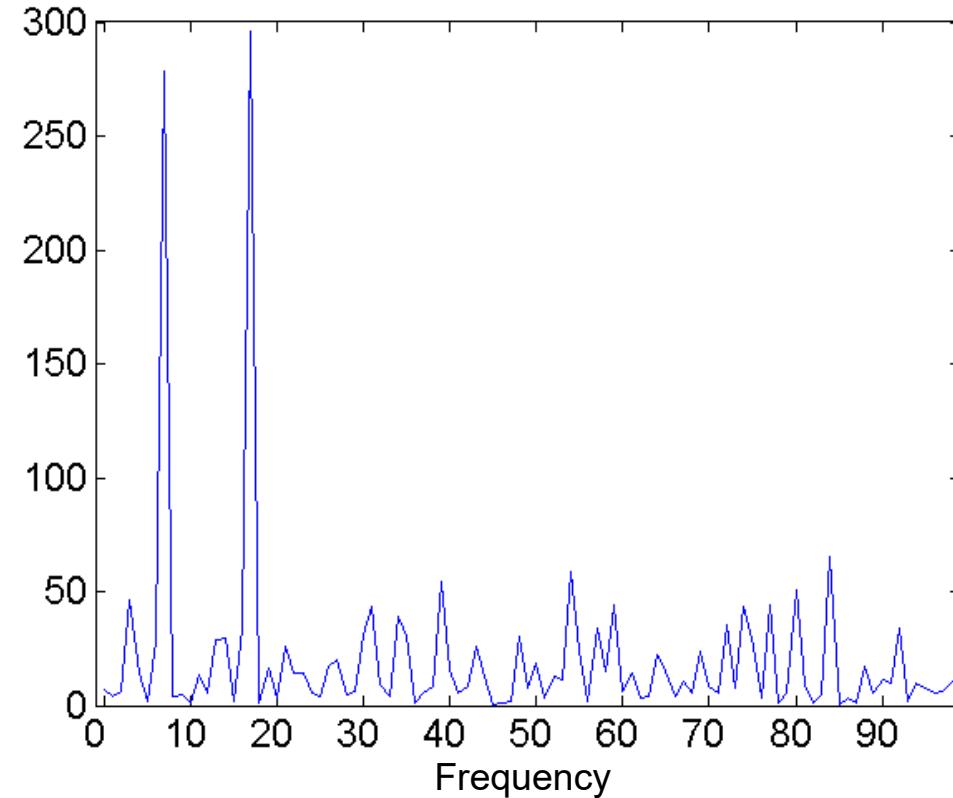
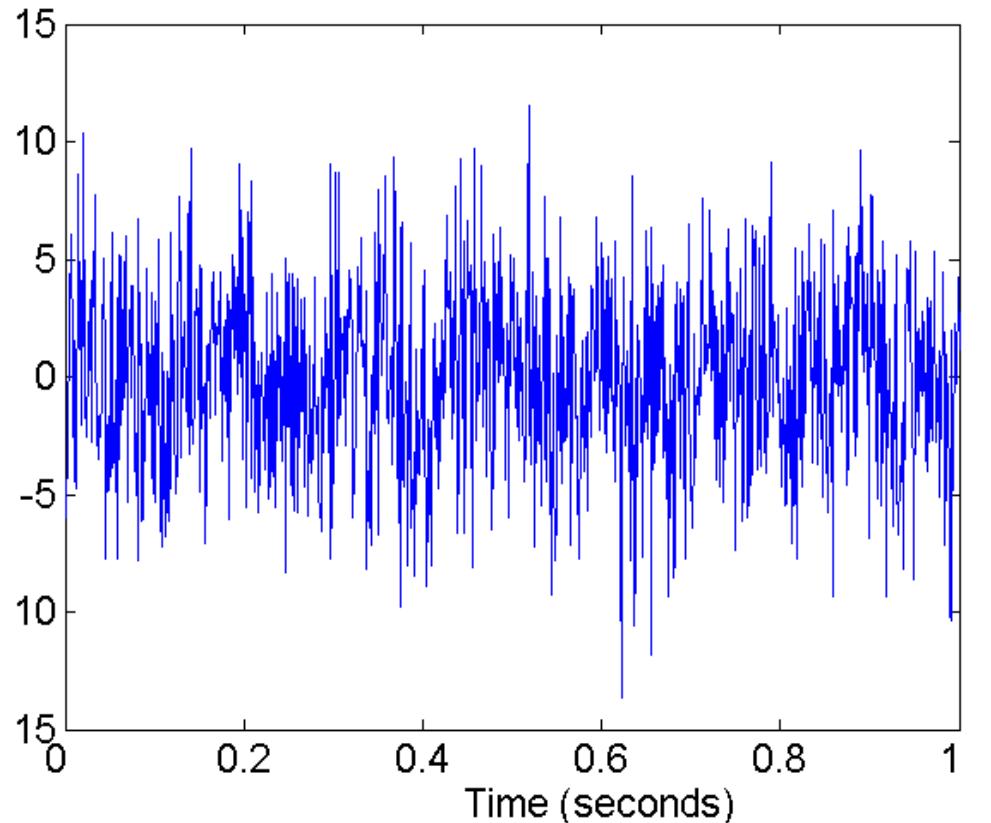
**Create new attributes that can capture the important information in a data set**

- More efficiently than the original attributes
- Three general methodologies
  - Feature extraction
    - Example: extracting edges from images
  - Feature construction
    - Example: dividing mass by volume to get density
  - Mapping data to new space
    - Example: Fourier and wavelet analysis



# Mapping Data to a New Space

Fourier and wavelet transform - Two Sine Waves + Noise



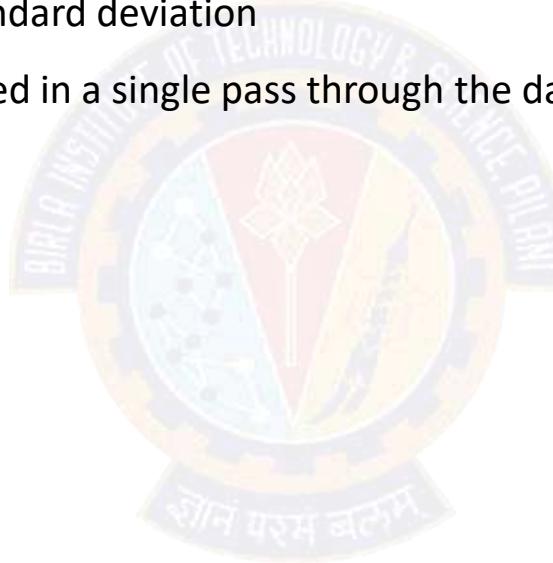
## Topic - 4

# Data Analysis

# Summary Statistics of Data

## Numbers that summarize properties of the data

- **Summarized properties include frequency, location and spread**
  - Examples: location – mean, spread - standard deviation
  - Most summary statistics can be calculated in a single pass through the data



# Measures of Location: Mean and Median

- The mean is the most common measure of the location of a set of points.
- However, the mean is very sensitive to outliers.
- Thus, the median or a trimmed mean is also commonly used.



$$\text{mean}(x) = \bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$$

- Assuming sorted  $\{x_i\}$

$$\text{median}(x) = \begin{cases} x_{(r+1)} & \text{if } m \text{ is odd, i.e., } m = 2r + 1 \\ \frac{1}{2}(x_{(r)} + x_{(r+1)}) & \text{if } m \text{ is even, i.e., } m = 2r \end{cases}$$

# Measures of Spread: Range and Variance

- Range is the difference between the max and min
- The variance or standard deviation  $s_x$  is the most common measure of the spread of a set of points.

$$\text{variance}(x) = s_x^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$$

- Because of outliers, other measures are often used. Average Absolute Distance is given by

$$\text{AAD}(x) = \frac{1}{m} \sum_{i=1}^m |x_i - \bar{x}|$$

$$\text{MAD}(x) = \text{median} \left( \{|x_1 - \bar{x}|, \dots, |x_m - \bar{x}|\} \right)$$

$$\text{interquartile range}(x) = x_{75\%} - x_{25\%}$$

# Similarity and Dissimilarity Measures

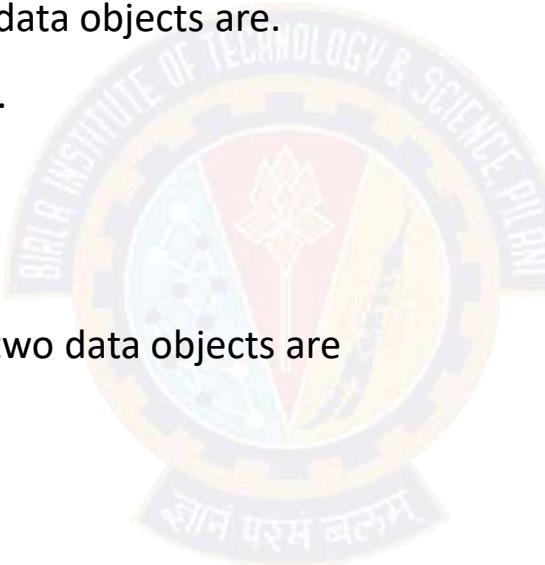
## ■ Similarity measure

- Numerical measure of how alike two data objects are.
- Is higher when objects are more alike.
- Often falls in the range [0,1]

## ■ Dissimilarity measure

- Numerical measure of how different two data objects are
- Lower when objects are more alike
- Minimum dissimilarity is often 0
- Upper limit varies

## ■ Proximity refers to a similarity or dissimilarity



# Similarity/Dissimilarity for Simple Attributes

The following table shows the similarity and dissimilarity between two objects,  $x$  and  $y$ , with respect to a single, simple attribute.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d =  x - y /(n - 1)$ (values mapped to integers 0 to $n-1$ , where $n$ is the number of values)	$s = 1 - d$
Interval or Ratio	$d =  x - y $	$s = -d, s = \frac{1}{1+d}, s = e^{-d},$ $s = 1 - \frac{d - \min_d}{\max_d - \min_d}$

# Euclidean Distance

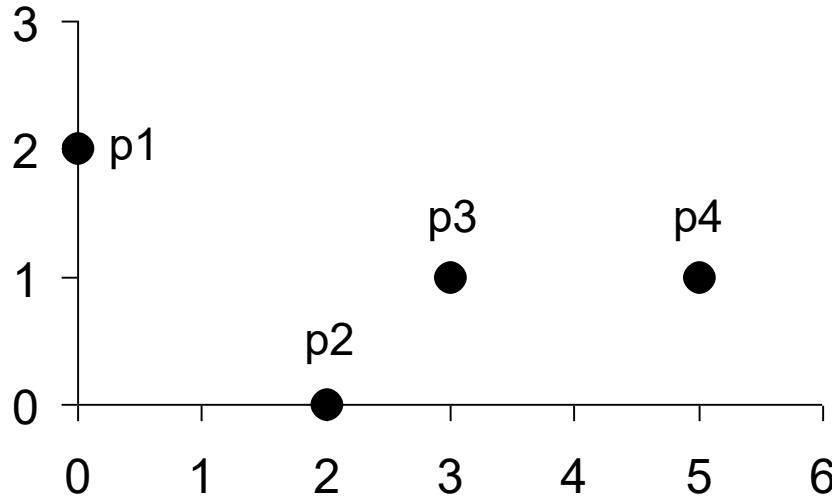
- Euclidean Distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

where  $n$  is the number of dimensions (attributes) and  $x_k$  and  $y_k$  are, respectively, the  $k^{th}$  attributes (components) or data objects  $\mathbf{x}$  and  $\mathbf{y}$ .

- Standardization is necessary, if scales differ.

# Euclidean Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

# Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$d(\mathbf{x}, \mathbf{y}) = \left( \sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}$$

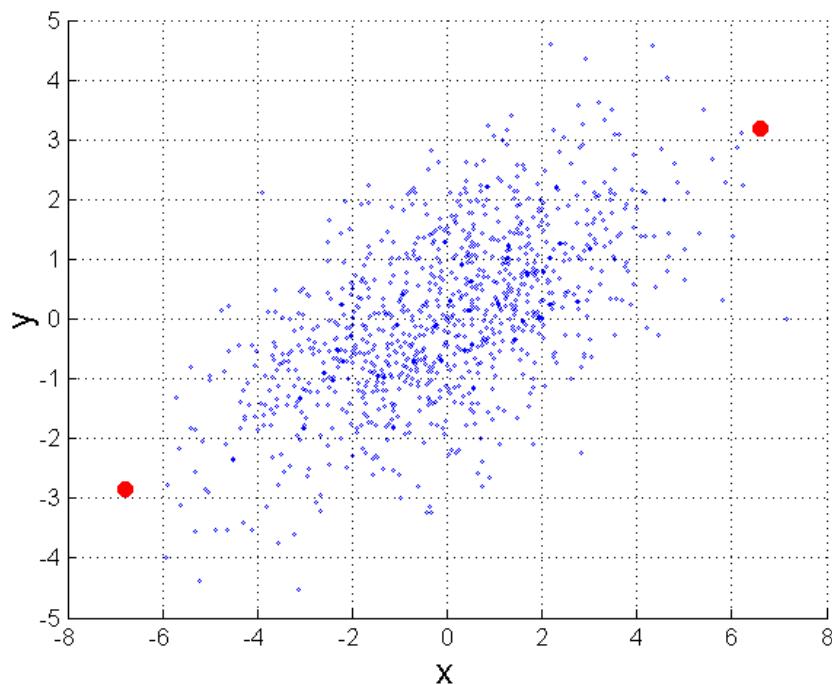
where  $r$  is a parameter,  $n$  is the number of dimensions (attributes) and  $x_k$  and  $y_k$  are, respectively, the  $k^{\text{th}}$  attributes (components) or data objects  $\mathbf{x}$  and  $\mathbf{y}$ .

# Minkowski Distance: Examples

- $r = 1$ . City block (Manhattan, taxicab,  $L_1$  norm) distance.
  - A common example of this for binary vectors is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$ . Euclidean distance
- $r \rightarrow \infty$ . “supremum” ( $L_{\max}$  norm,  $L_{\infty}$  norm) distance.
  - This is the maximum difference between any component of the vectors
- Do not confuse  $r$  with  $n$ , i.e., all these distances are defined for all numbers of dimensions.

# Mahalanobis Distance

$$\text{mahalanobis}(\mathbf{x}, \mathbf{y}) = ((\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y}))^{-0.5}$$



$\Sigma$  is the covariance matrix

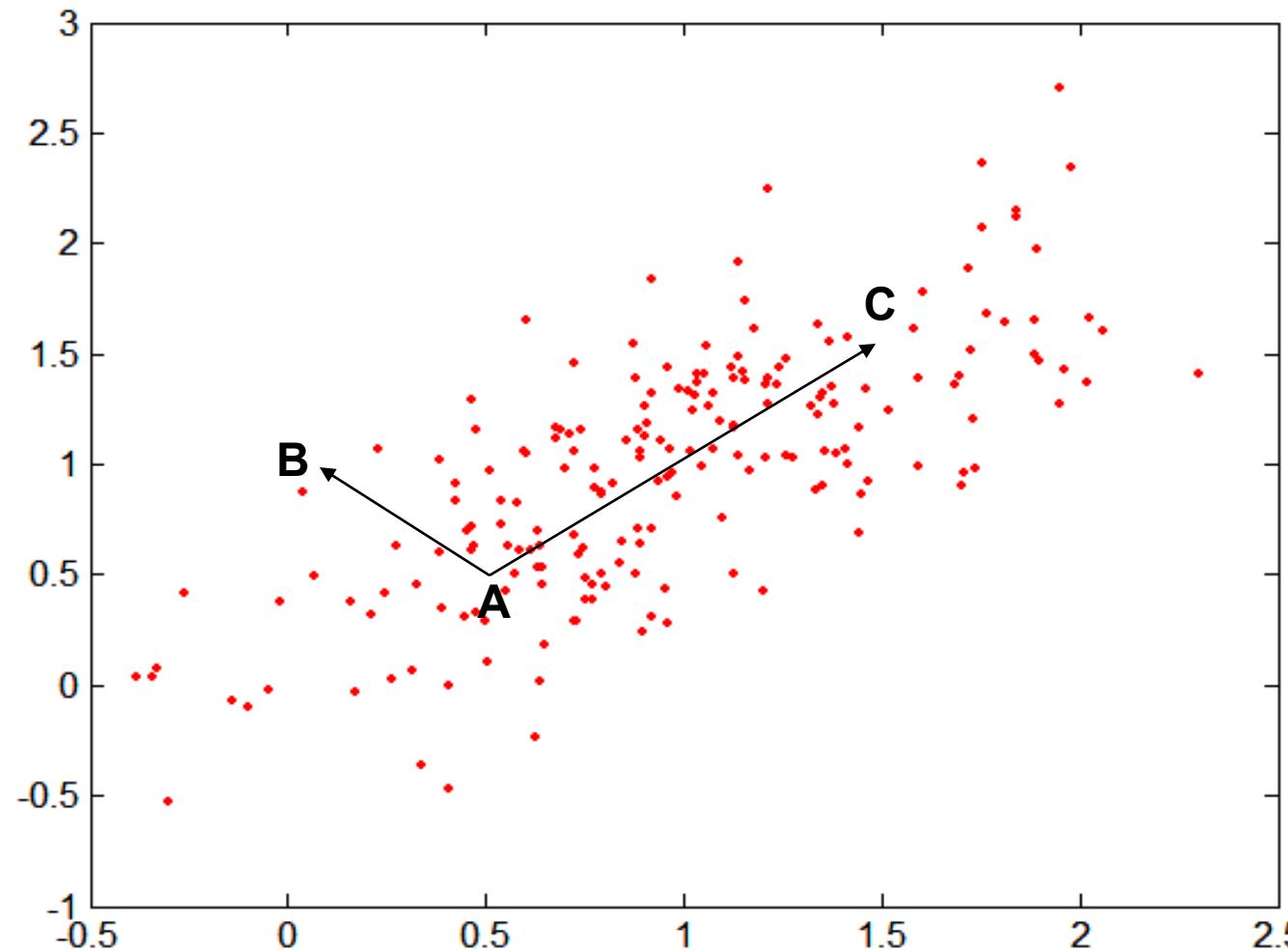
$$\frac{1}{N} \begin{bmatrix} \sum_i (x_i - m_x)^2 & \sum_i (x_i - m_x)(y_i - m_y) \\ \sum_i (x_i - m_x)(y_i - m_y) & \sum_i (y_i - m_y)^2 \end{bmatrix}$$

$$m_x = \frac{1}{N} \sum_i x_i$$

$$m_y = \frac{1}{N} \sum_i y_i$$

- For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.

# Mahalanobis Distance



Covariance Matrix:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

**Mahal(A,B) = 5**

**Mahal(A,C) = 4**

# Cosine Similarity

- If  $\mathbf{d}_1$  and  $\mathbf{d}_2$  are two document vectors, then

$$\text{Cos}(\mathbf{d}_1, \mathbf{d}_2) = \langle \mathbf{d}_1, \mathbf{d}_2 \rangle / \|\mathbf{d}_1\| \|\mathbf{d}_2\|,$$

where  $\langle \mathbf{d}_1, \mathbf{d}_2 \rangle$  indicates inner product or vector dot product of vectors,  $\mathbf{d}_1$  and  $\mathbf{d}_2$ , and  $\|\mathbf{d}\|$  is the length of vector  $\mathbf{d}$ .

- Example:

$$\mathbf{d}_1 = 3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0$$

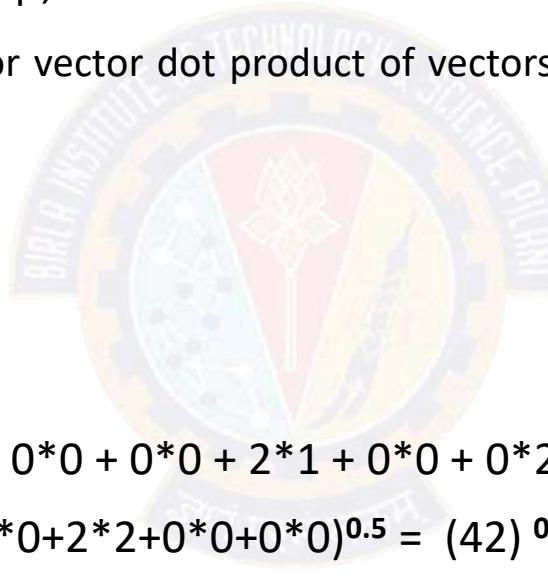
$$\mathbf{d}_2 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2$$

$$\langle \mathbf{d}_1, \mathbf{d}_2 \rangle = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$$

$$\|\mathbf{d}_1\| = (3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)^{0.5} = (42)^{0.5} = 6.481$$

$$\|\mathbf{d}_2\| = (1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2)^{0.5} = (6)^{0.5} = 2.449$$

$$\text{Cos}(\mathbf{d}_1, \mathbf{d}_2) = 0.3150$$



## Correlation measures the linear relationship between objects

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard\_deviation}(\mathbf{x}) * \text{standard\_deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x s_y}, \quad (2.11)$$

where we are using the following standard statistical notation and definitions

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y}) \quad (2.12)$$

$$\text{standard\_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}$$

$$\text{standard\_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (y_k - \bar{y})^2}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k \text{ is the mean of } \mathbf{x}$$

$$\bar{y} = \frac{1}{n} \sum_{k=1}^n y_k \text{ is the mean of } \mathbf{y}$$

# Correlation vs Cosine vs Euclidean Distance

- Compare the three proximity measures according to their behavior under variable transformation
  - scaling: multiplication by a value
  - translation: adding a constant

Property	Cosine	Correlation	Euclidean Distance
Invariant to scaling (multiplication)	Yes	Yes	No
Invariant to translation (addition)	No	Yes	No

- Consider the example
  - $\mathbf{x} = (1, 2, 4, 3, 0, 0, 0)$ ,  $\mathbf{y} = (1, 2, 3, 4, 0, 0, 0)$
  - $\mathbf{y}_s = \mathbf{y} * 2$  (scaled version of  $\mathbf{y}$ ),  $\mathbf{y}_t = \mathbf{y} + 5$  (translated version)

Measure	$(\mathbf{x}, \mathbf{y})$	$(\mathbf{x}, \mathbf{y}_s)$	$(\mathbf{x}, \mathbf{y}_t)$
Cosine	0.9667	0.9667	0.7940
Correlation	0.9429	0.9429	0.9429
Euclidean Distance	1.4142	5.8310	14.2127

# Comparison of Proximity Measures

- **Domain of application**

- Similarity measures tend to be specific to the type of attribute and data
- Record data, images, graphs, sequences, 3D-protein structure, etc. tend to have different measures

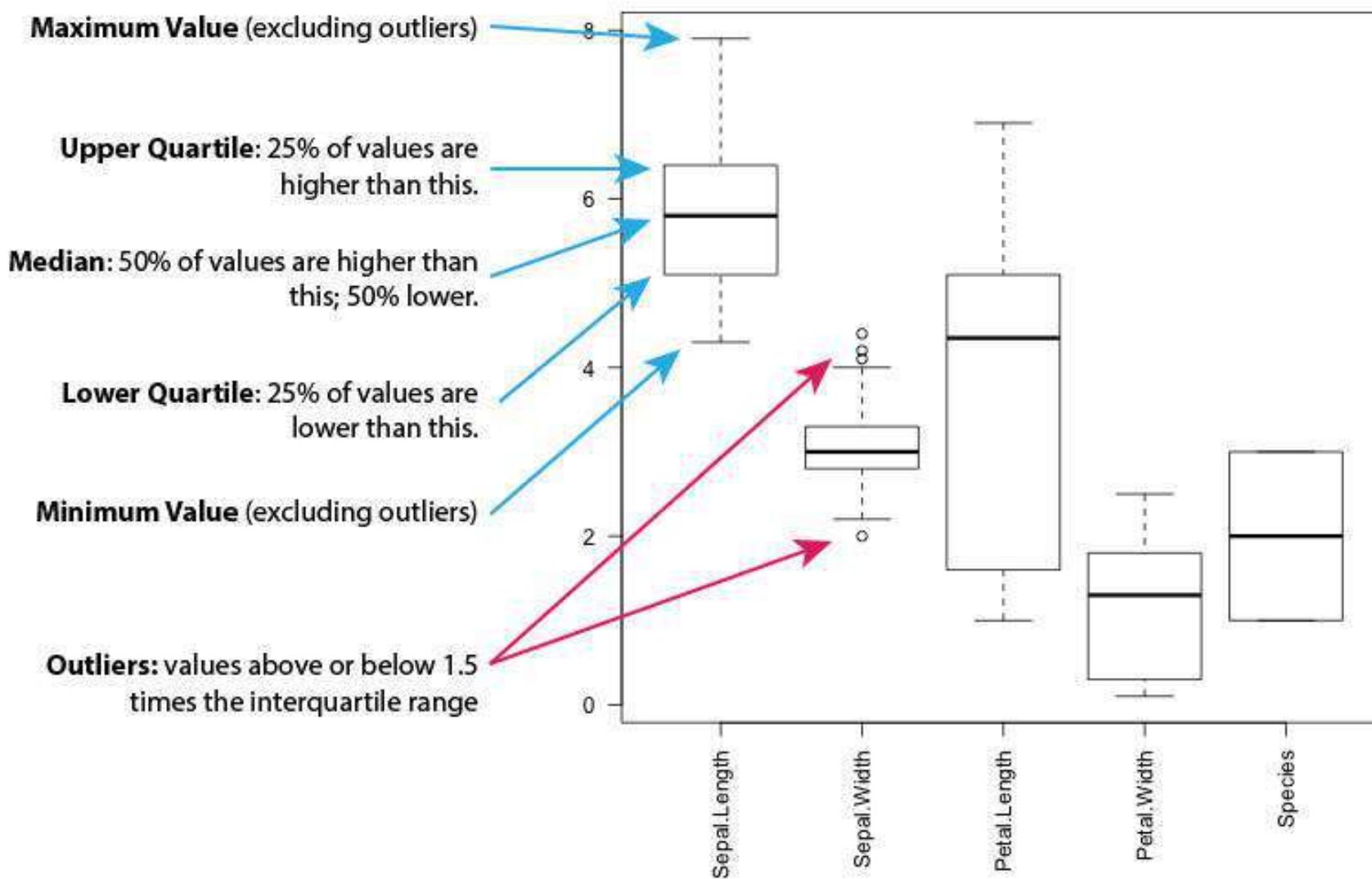
- **However, one can talk about various properties that you would like a proximity measure to have**

- Symmetry is a common one
- Tolerance to noise and outliers is another
- Ability to find more types of patterns?
- Many others possible

- **The measure must be applicable to the data and produce results that agree with domain knowledge**

## Data Visualization

# Visualization Techniques: Box Plots



# Topic 5

## Software Tools for ML

# Prerequisites

## ▪ Jupyter Notebook

- Open-source web application to create and share live code, equations, visualizations and narrative text.
- Used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- Easy to tinker with code and execute it in steps
- Run in local browser: <https://jupyter.org/try>
- Local installation: <https://jupyter.org/install.html>
- **Repository** of Jupyter Notebooks: <https://github.com/ageron/handson-ml>

## ▪ Colab

- Google's flavor of Jupyter notebooks tailored for machine learning and data analysis
- Runs entirely in the *cloud*.
- free access to hardware accelerators like GPUs and TPUs (with some restrictions).
- <http://colab.research.google.com>

## ▪ Popular Datasets

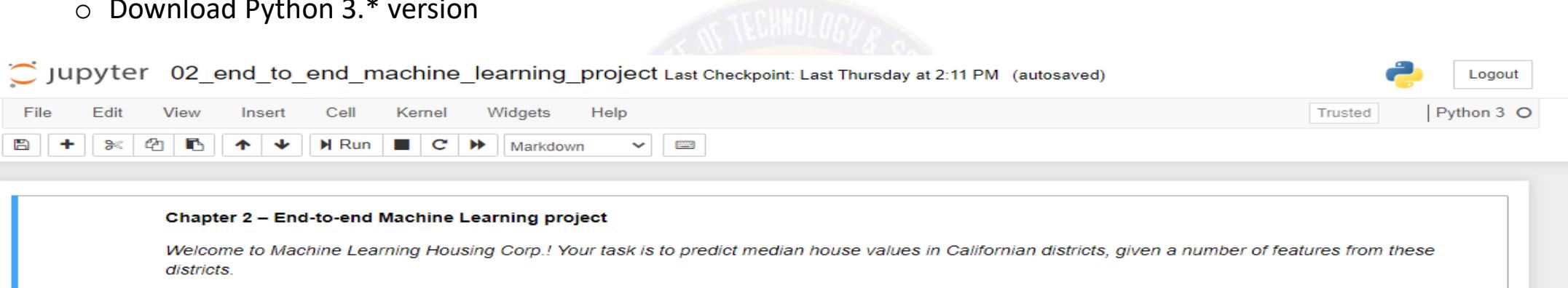
- UC Irvine Machine Learning Repository: <http://archive.ics.uci.edu/ml/>
- Kaggle datasets: <https://www.kaggle.com/datasets>
- Amazon's AWS datasets: <http://aws.amazon.com/fr/datasets/>

# Local Installation

## Preferred Mode

- Install a virtual environment via Anaconda
    - Free Anaconda Python distribution: <https://www.anaconda.com/products/individual>
    - Download Python 3.\* version

- Download Python 3.\* version



jupyter 02\_end\_to\_end\_machine\_learning\_project Last Checkpoint: Last Thursday at 2:11 PM (autosaved) [Logout](#)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Chapter 2 – End-to-end Machine Learning project

Welcome to Machine Learning Housing Corp.! Your task is to predict median house values in Californian districts, given a number of features from these districts.

This notebook contains all the sample code and solutions to the exercises in chapter 2.

**Note:** You may find little differences between the code outputs in the book and in these Jupyter notebooks: these slight differences are mostly due to the random nature of many training algorithms: although I have tried to make these notebooks' outputs as constant as possible, it is impossible to guarantee that they will produce the exact same output on every platform. Also, some data structures (such as dictionaries) do not preserve the item order. Finally, I fixed a few minor bugs (I added notes next to the concerned cells) which lead to slightly different results, without changing the ideas presented in the book.

## Setup

First, let's make sure this notebook works well in both python 2 and 3, import a few common modules, ensure Matplotlib plots figures inline and prepare a function to save the figures:

```
In [27]: # To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals
```

# Prerequisites

## Python and several scientific libraries

### ▪ Python

- high-level, dynamically typed multiparadigm programming language.
- almost like pseudocode, very readable
- Platform: Linux/Windows/MacOS
- Documentation: <https://www.python.org/doc/>
- Interactive Python tutorial: <http://learnpython.org/>
- Stand alone installation: (preferably 3.7 or higher) <https://www.python.org/downloads/>



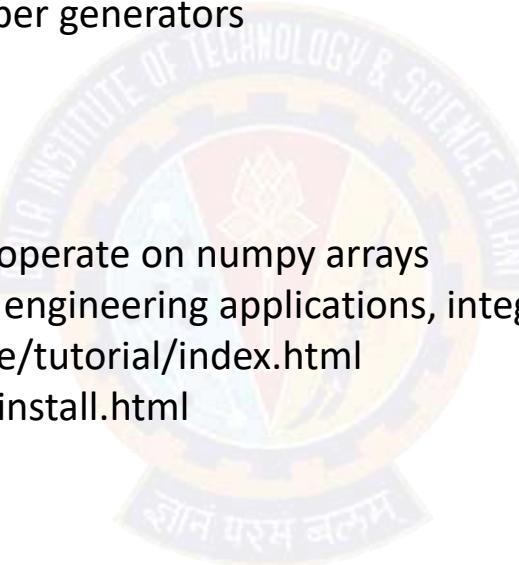
### ▪ scikit-learn

- Open source library that supports supervised and unsupervised learning.
- Various model fitting, data preprocessing, model selection and evaluation tools, and other utilities.
- Platform: Linux/Windows/MacOS
- User guide: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
- Stand alone installation: <https://scikit-learn.org/stable/install.html>

# Prerequisites

## Important libraries

- **Numpy**
  - Scientific computing library with Python
  - Provides high-performance multidimensional array (e.g., vector, matrix) and basic tools to compute with and manipulate these arrays, linear algebra, random number generators
  - <https://numpy.org/doc/stable/reference/>
  - <https://numpy.org/install/>
- **SciPy**
  - Builds on numpy, provides functions that operate on numpy arrays
  - Useful for different types of scientific and engineering applications, integration, image processing, ...
  - <https://docs.scipy.org/doc/scipy/reference/tutorial/index.html>
  - Standalone installation: <https://scipy.org/install.html>
- **Matplotlib**
  - Plotting library
  - <https://matplotlib.org/>
  - <https://matplotlib.org/tutorials/index.html>
  - <https://matplotlib.org/users/installing.html>



# Important ML Steps and Functions

[https://github.com/ageron/handson-ml/blob/master/02\\_end\\_to\\_end\\_machine\\_learning\\_project.ipynb](https://github.com/ageron/handson-ml/blob/master/02_end_to_end_machine_learning_project.ipynb)

```
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

In [6]: housing.info()

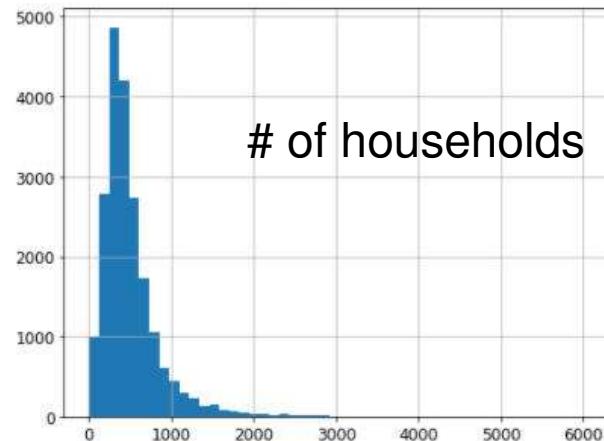
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude          20640 non-null float64
latitude           20640 non-null float64
housing_median_age 20640 non-null float64
total_rooms         20640 non-null float64
total_bedrooms      20433 non-null float64
population          20640 non-null float64
households          20640 non-null float64
median_income        20640 non-null float64
median_house_value   20640 non-null float64
ocean_proximity     20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```
>>> housing["ocean_proximity"].value_counts()
<1H OCEAN      9136
INLAND          6551
NEAR OCEAN      2658
NEAR BAY         2290
ISLAND            5
Name: ocean_proximity, dtype: int64
```

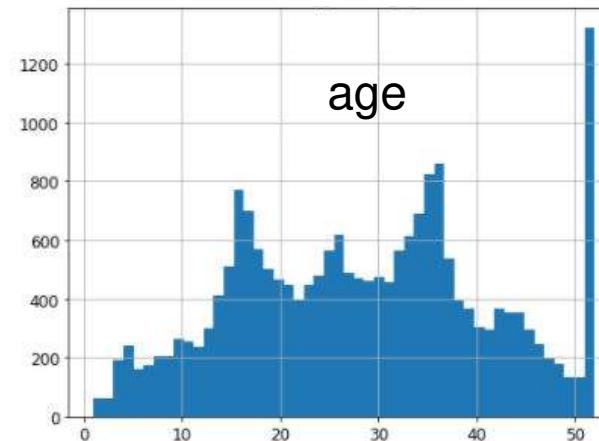
- 20,640 data instances in the dataset – fairly small
- 207 districts are missing total\_bedrooms attribute
- except ocean\_proximity, rest of the attributes numerical
- type of ocean\_proximity is object, so it must be a text attribute - a repeated categorical attribute!

# Attribute Visualization

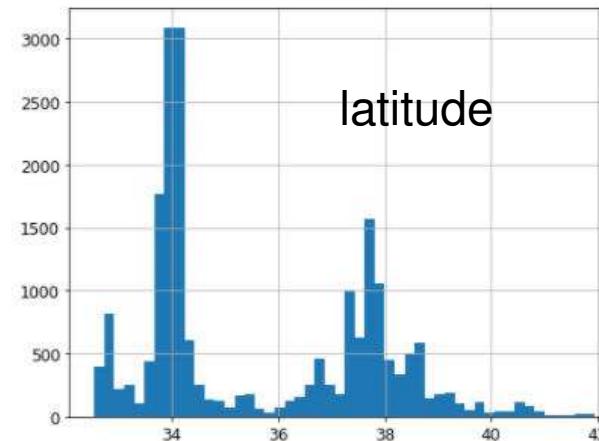
```
%matplotlib inline # only in a Jupyter notebook
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```



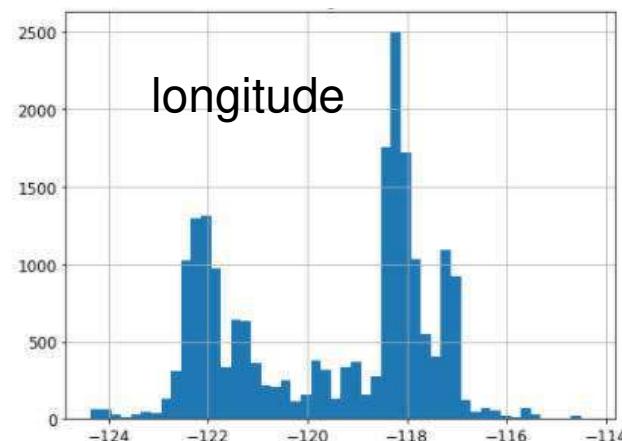
# of households



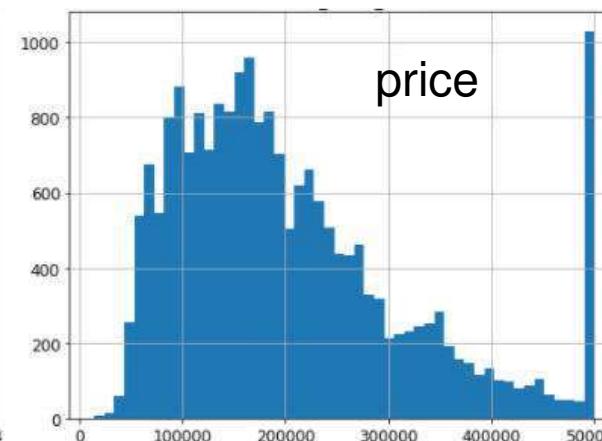
age



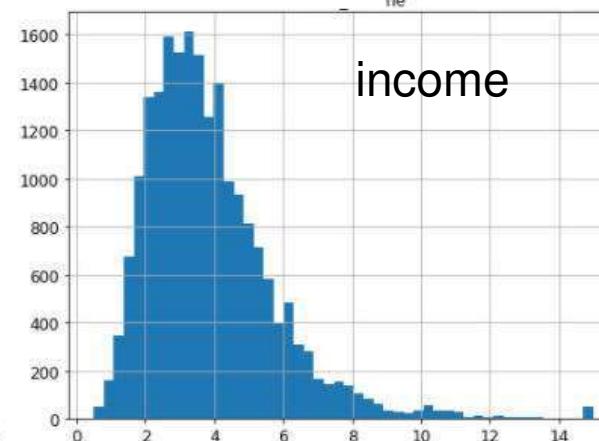
latitude



longitude



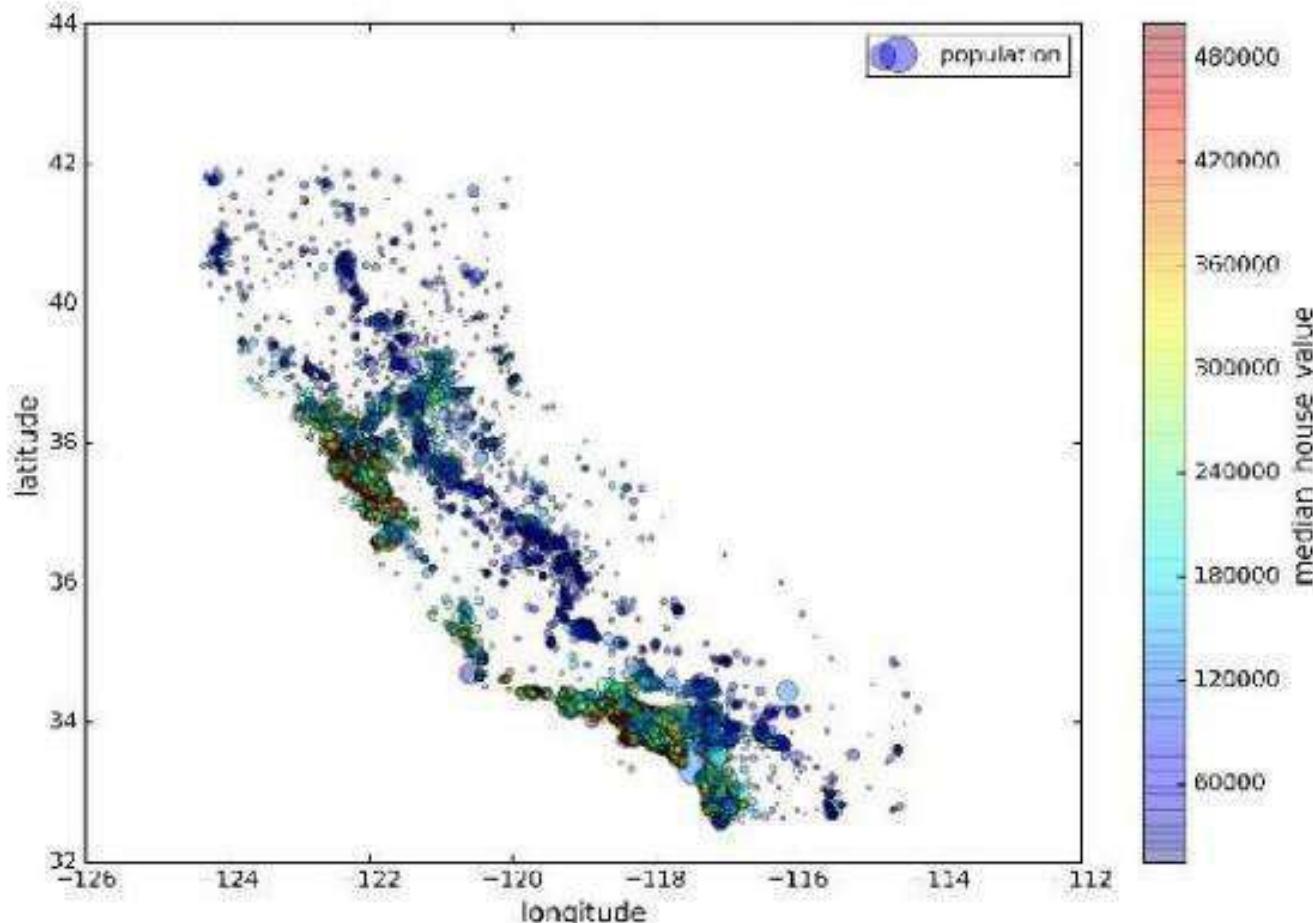
price



income

# Visualizing Training Data for Insights

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
    s=housing["population"]/100, label="population", figsize=(10,7),
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
)
plt.legend()
```



# Correlations in Training Data

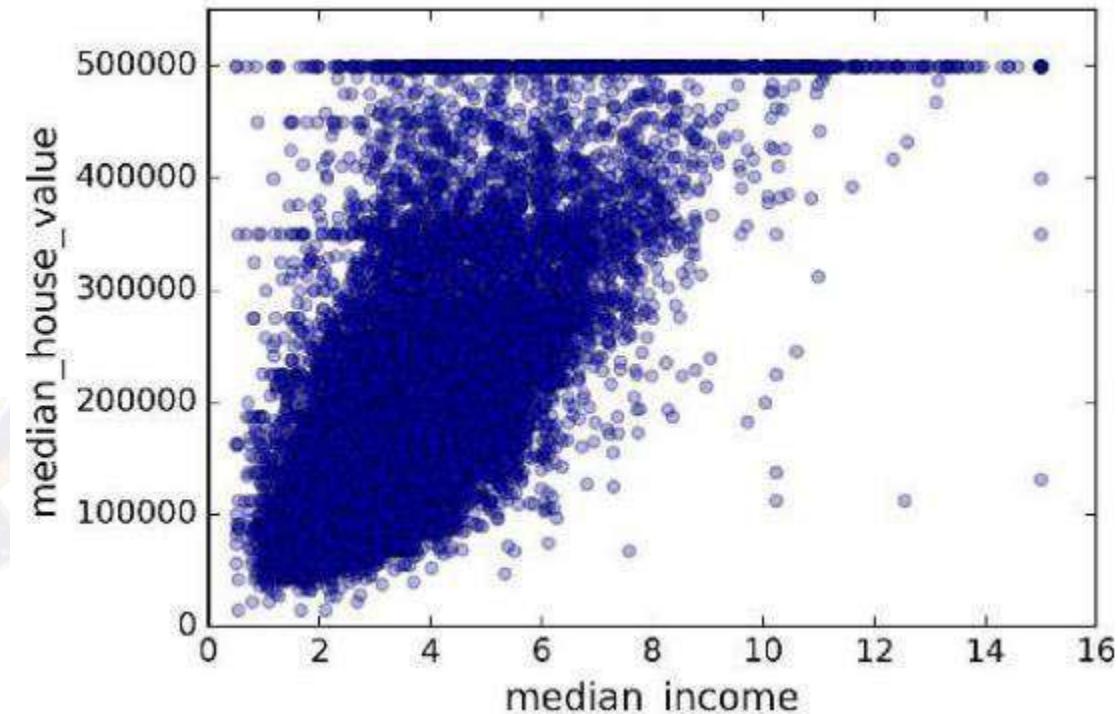
```
corr_matrix = housing.corr()
```

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value    1.000000
median_income         0.687170
total_rooms           0.135231
housing_median_age   0.114220
households            0.064702
total_bedrooms        0.047865
population            -0.026699
longitude             -0.047279
latitude              -0.142826
Name: median_house_value, dtype: float64
```

- House prices are strongly correlated with income
- small negative correlation between the latitude and the median house value
  - i.e., prices have a slight tendency to go down when you go north)
- price cap clearly visible as a horizontal line at \$500K.
- Additional horizontal lines around \$450k, \$350K, perhaps one around \$280K, ...

```
from pandas.tools.plotting import scatter_matrix
```

```
housing.plot(kind="scatter", x="median_income", y="median_house_value",
alpha=0.1)
```



# Experimenting with Attribute Combinations

- Total number of rooms in a district not very useful
- What may be important is the number of rooms per household.
- Similarly, the total number of bedrooms compared to the number of rooms may be more meaningful

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"] = housing["population"]/housing["households"]

>>> corr_matrix = housing.corr()
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value      1.000000
median_income           0.687160
rooms_per_household    0.146285
total_rooms             0.135097
housing_median_age      0.114110
households              0.064506
total_bedrooms          0.047689
population_per_household -0.021985
population              -0.026920
longitude               -0.047432
latitude                -0.142724
bedrooms_per_room       -0.259984
Name: median_house_value, dtype: float64
```

- `rooms_per_household` is more correlated with price than total number of bedrooms or rooms.

# Data Cleaning

- Some of the instances don't have total\_bedroom values

```
housing.dropna(subset=["total_bedrooms"])      # option 1
housing.drop("total_bedrooms", axis=1)          # option 2
median = housing["total_bedrooms"].median()     # option 3
housing["total_bedrooms"].fillna(median, inplace=True)
```

- Imputer utility in scikit-learn

```
from sklearn.preprocessing import Imputer
imputer = Imputer(strategy="median")
```

- Since median applies only to numerical data, create a copy of the data without ocean-proximity

```
housing_num = housing.drop("ocean_proximity", axis=1)
```

```
imputer.fit(housing_num)
```

- The imputer stores the median of each attribute the result in its statistics\_ instance variable.

```
>>> imputer.statistics_
array([-118.51, 34.26, 29., 2119.5, 433., 1164., 408., 3.5409])
```

- Use this “trained” imputer to replace any missing values by the learned medians

```
x = imputer.transform(housing_num)
```

# Handling Textual and Categorical Data

## Convert the text attributes to numbers

- Scikit-learn provides a transformer

```
>>> from sklearn.preprocessing import LabelEncoder
>>> encoder = LabelEncoder()
>>> housing_cat = housing["ocean_proximity"]
>>> housing_cat_encoded = encoder.fit_transform(housing_cat)
>>> housing_cat_encoded
array([0, 0, 4, ..., 1, 0, 3])

>>> print(encoder.classes_)
['<1H OCEAN' 'INLAND' 'ISLAND' 'NEAR BAY' 'NEAR OCEAN']
```

- ML algorithms assume closer values are more similar. But '<1H OCEAN' and 'NEAR OCEAN' are far away in values!
- Solution! Use 1-hot encoding

```
>>> from sklearn.preprocessing import LabelBinarizer
>>> encoder = LabelBinarizer()
>>> housing_cat_1hot = encoder.fit_transform(housing_cat)
>>> housing_cat_1hot
array([[1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1],
       ...,
       [0, 1, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0]])
```

# Test Set Creation

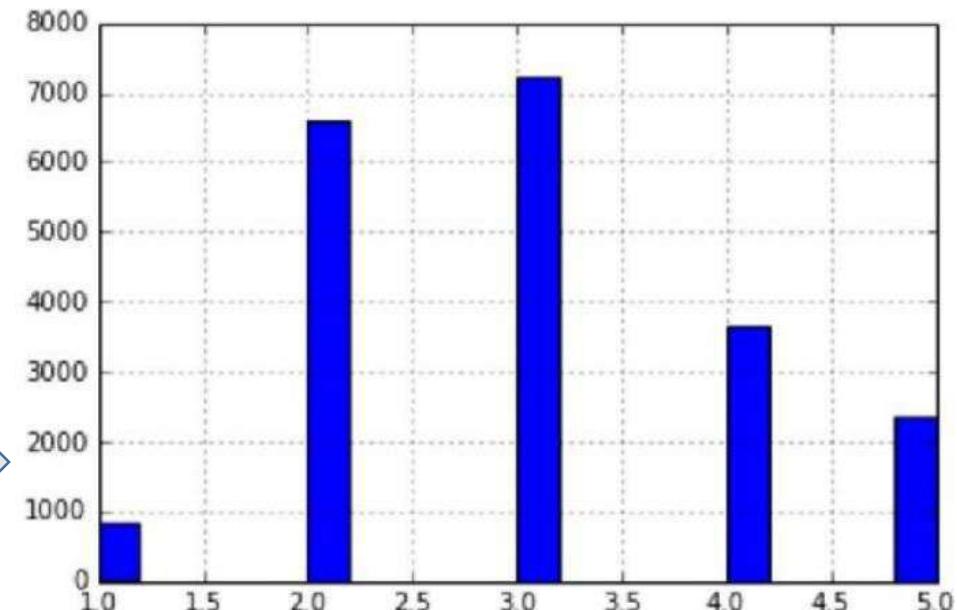
```
from sklearn.model_selection import train_test_split  
  
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

- Test set size is 20% of training set
- Uses uniform sampling of data.
- Not appropriate for heavy tailed distribution
- income very important to predict house prices.
- Important that test set is representative of the various categories of incomes in the dataset.
- Most income values are around \$20–\$50K, but some >> \$60K.

```
housing["income_cat"] = np.ceil(housing["median_income"] / 1.5)  
housing["income_cat"].where(housing["income_cat"] < 5, 5.0, inplace=True)
```

```
from sklearn.model_selection import StratifiedShuffleSplit  
  
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)  
for train_index, test_index in split.split(housing, housing["income_cat"]):  
    strat_train_set = housing.loc[train_index]  
    strat_test_set = housing.loc[test_index]
```

Ensures reproducibility of results



Histogram of income categories



**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 3(12<sup>th</sup> August,2023)**

# Session 3 – 12<sup>th</sup> August, 2023

## Big Picture: End-to-end Machine Learning

### 3.1 Model Selection and Training

3.1.1. Prediction Problem

3.1.2. Classification Problem

### 3.2 Evaluation

3.2.1. Prediction Problem

3.2.2. Classification Problem

### 3.3 Machine Learning Pipeline



# Model Selection and Training

## Types of Learning

### ■ Supervised Learning

- Each training tuple is *supplied* with the class label

- *Supervision*:

- ❖ The training data (observations, measurements, etc.) are accompanied by *labels* indicating the class of the observations

- New data is classified based on the classifier developed using the training set

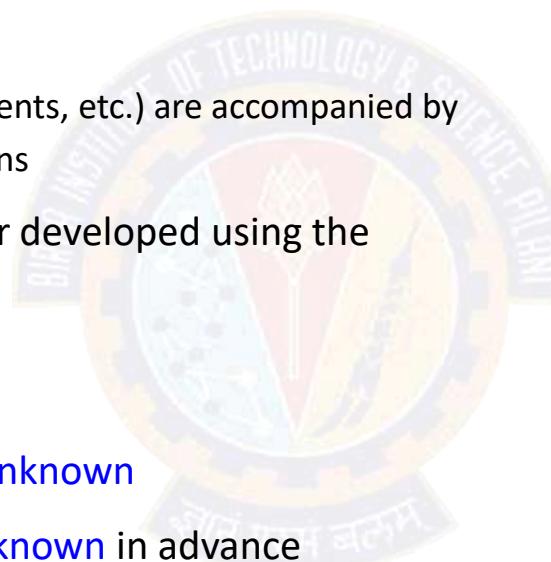
### ■ Unsupervised Learning

- The *class labels* for the training tuples are *unknown*

- The *number of classes* to be learned is *not known* in advance

- For example, if we did not have the *loan decision* data available for the training set, we could use clustering to try to determine "*groups of related tuples*"

- ❖ These groups may correspond to *risk groups* within the loan application data

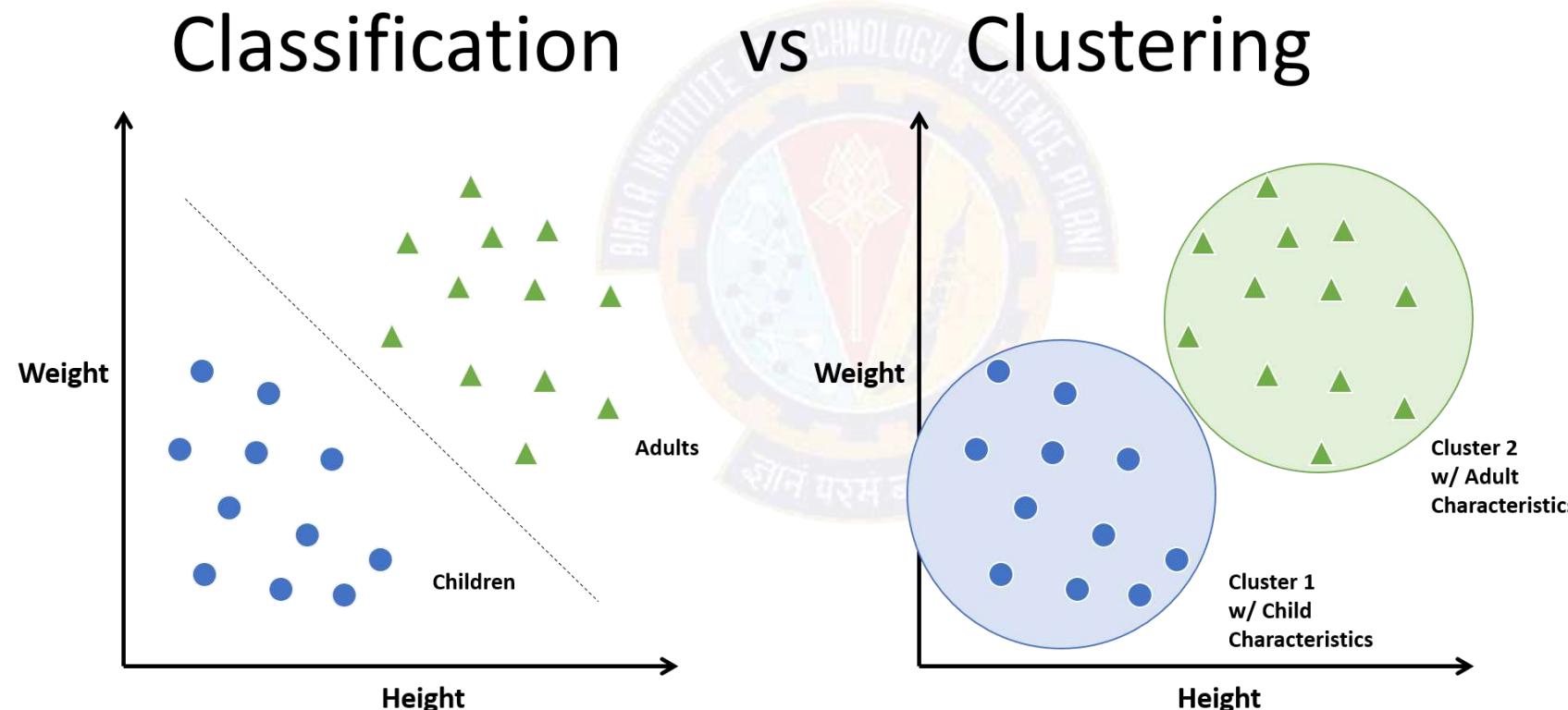


categorical  
categorical  
continuous  
class

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

# Model Selection & Training

## Supervised Vs. Unsupervised Learning

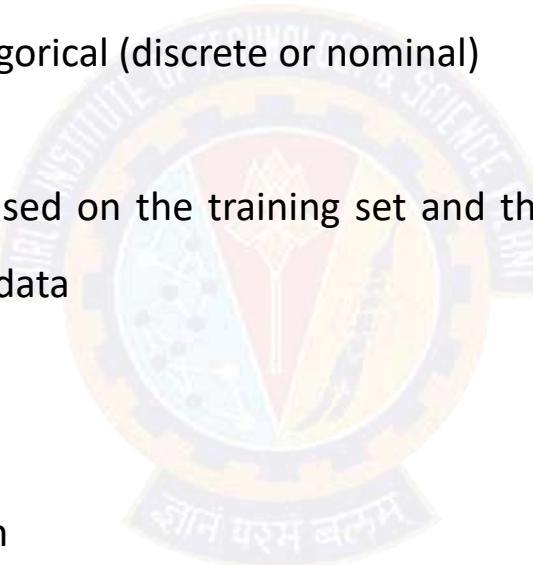


# Model Selection & Training

## Classification Vs. Prediction

### ■ Classification

- Models data where outcome variable is categorical (discrete or nominal)
- Example: Classifying emails as Spam Vs. Ham
  - ❖ classifies data (constructs a model) based on the training set and the values (class labels: spam/ham) in a classifying attribute and uses it in classifying new data



### ■ Prediction

- Models continuous-valued functions
- Example: the house price prediction problem
  - ❖ Here, prediction involves housing prices, which are continuous real-valued numbers.
  - ❖ Use regression model since median house prices are available along with training data (predictors)

# Model Selection and Training

## What is Classification?

- Consider the following examples:
  - A bank loans officer needs to analyze data to learn which loan applicants are "safe" and which are "risky" for the bank
  - A marketing manager at an electronics store wants to know whether a customer with a given profile will **buy a new computer (Yes/No)**
  - A medical researcher wants to analyze breast cancer data to predict which one of **three specific treatments** a patient should receive

# Model Selection and Training

## What is Classification?

- In each of these examples, the data analysis task is **classification**
- Here, a model (or **classifier**) is constructed to predict *class (categorical) labels*, such as:
  - "safe" or "risky" for the loan application data
  - "yes" or "no" for the marketing data
  - "treatment A", "treatment B" or "treatment C" for the medical data
- These categories can be represented by discrete values, where the ordering among values has no meaning

# Model Selection and Training

## What is Prediction?

- The data analysis may involve predicting a **numeric value**
  - E.g., the marketing manager wants to predict **how much** a given customer will spend during a sale at the Electronics Store
  - E.g., Organizations try to predict sales in the near and long term so that they can keep their inventory stocked appropriately.
- Here, the model predicts a ***continuous-valued function***, or ***ordered value***, as opposed to a **class label**
- This model is called a **predictor**
- The statistical technique most often used for numeric prediction is **Regression analysis**

# Model Selection and Training

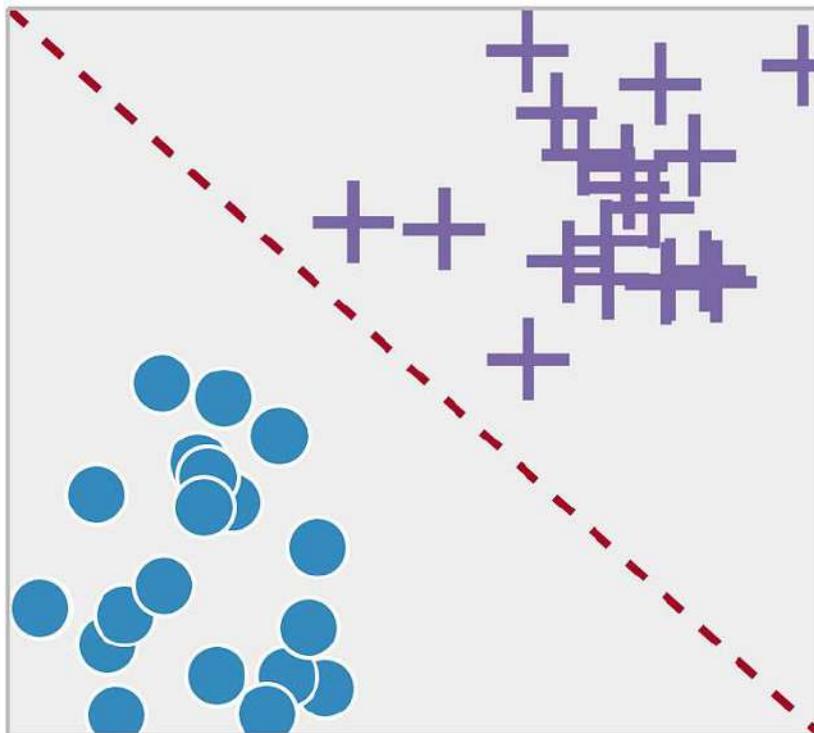
## What is Prediction?

- For prediction, we lose the terminology of "class label attribute" because the attribute for which values are being predicted is continuous-valued (ordered) rather than categorical (discrete-valued and unordered).
- The attribute can be referred to simply as the predicted attribute.
- Suppose that, in our example of "safe" or "risky" for the loan application data, we instead wanted to predict the amount (in dollars) that would be "safe" for the bank to loan an applicant.
- The task becomes prediction, rather than classification.
- We would replace the categorical attribute, loan decision, with the continuous-valued loan amount as the predicted attribute, and build a predictor for our task.

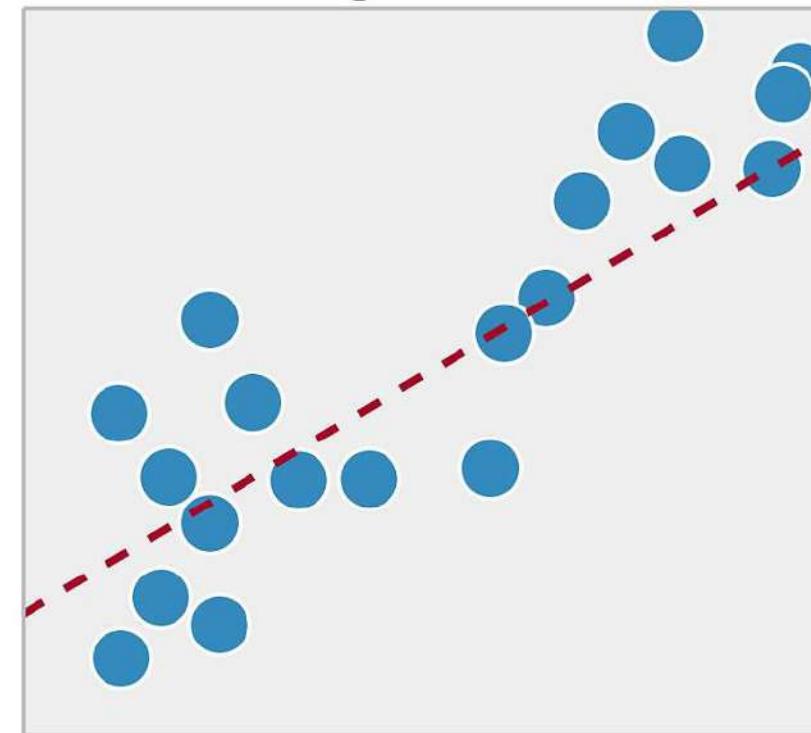
# Model Selection & Training

## Classification Vs. Prediction

Classification



Regression



# Classification

## A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - ❖ The known label of test sample is compared with the classified result from the model
    - ❖ Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - ❖ Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

# Classification

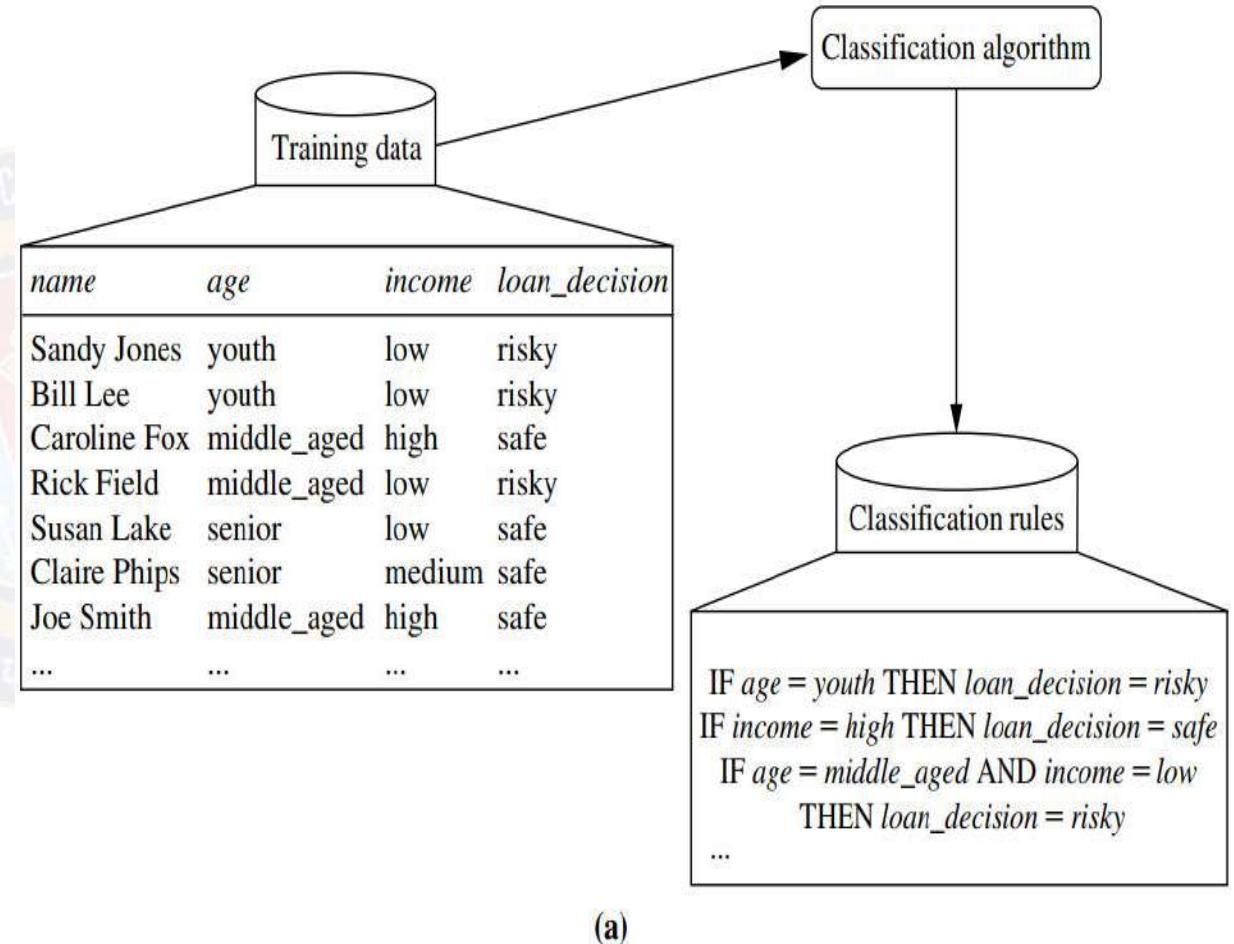
## Learning Step

- The classification algorithm builds the classifier by analyzing ("learning from") a **training set** made up of objects and their associated class labels
- An object,  $X$ , is represented by a d-dimensional **attribute vector**,  $X = (x_1, x_2, \dots x_d)$ ,
  - $x_1, x_2, \dots x_d$  represent measurements from d attributes,  $A_1, A_2, \dots A_d$
- Each object,  $X$ , belongs to a predefined class indicated by an attribute called the **class label attribute**
- The class label attribute is *categorical* (or *nominal*) and *unordered* where each value serves as a *category* or *class*
- The individual objects making up the training set are referred to as **training objects**
  - These are randomly sampled from the database under analysis

# Classification

## Learning Step

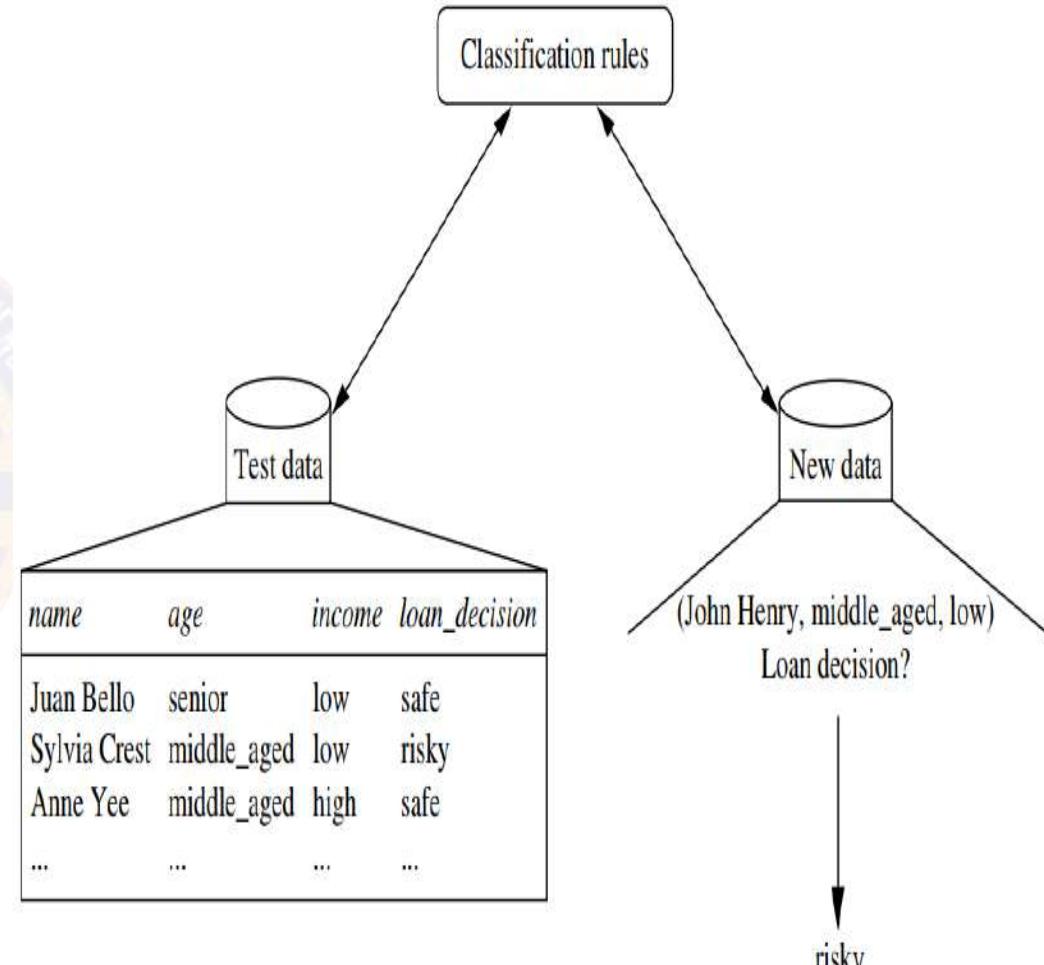
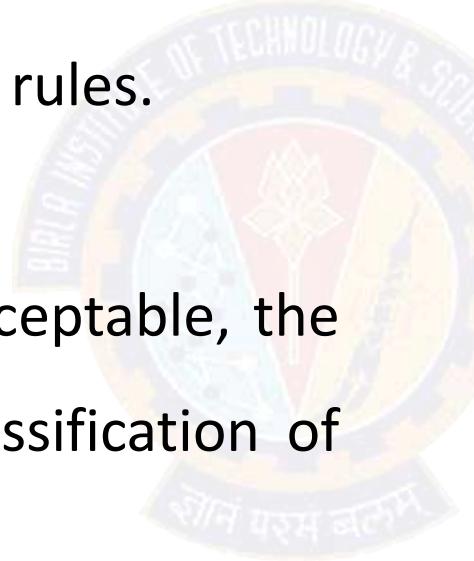
- Training data are analyzed by a classification algorithm
- Here,
  - ✓ the *class label* attribute is *loan decision*, and
  - ✓ the learned model or classifier is represented in the form of *classification rules*



# Classification

## Classification Step

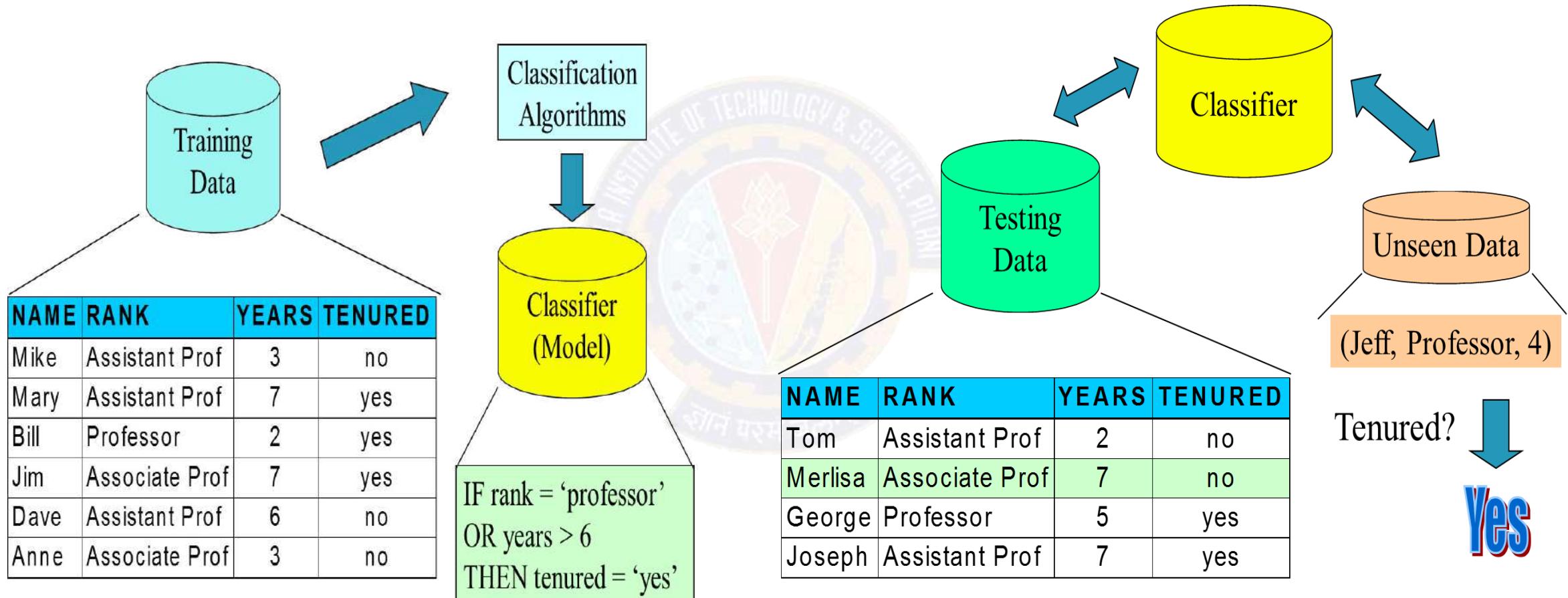
- Test data are used to estimate the *accuracy* of the classification rules.
- If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples



(b)

# Classification

## Illustrating Classification Task



# Classification

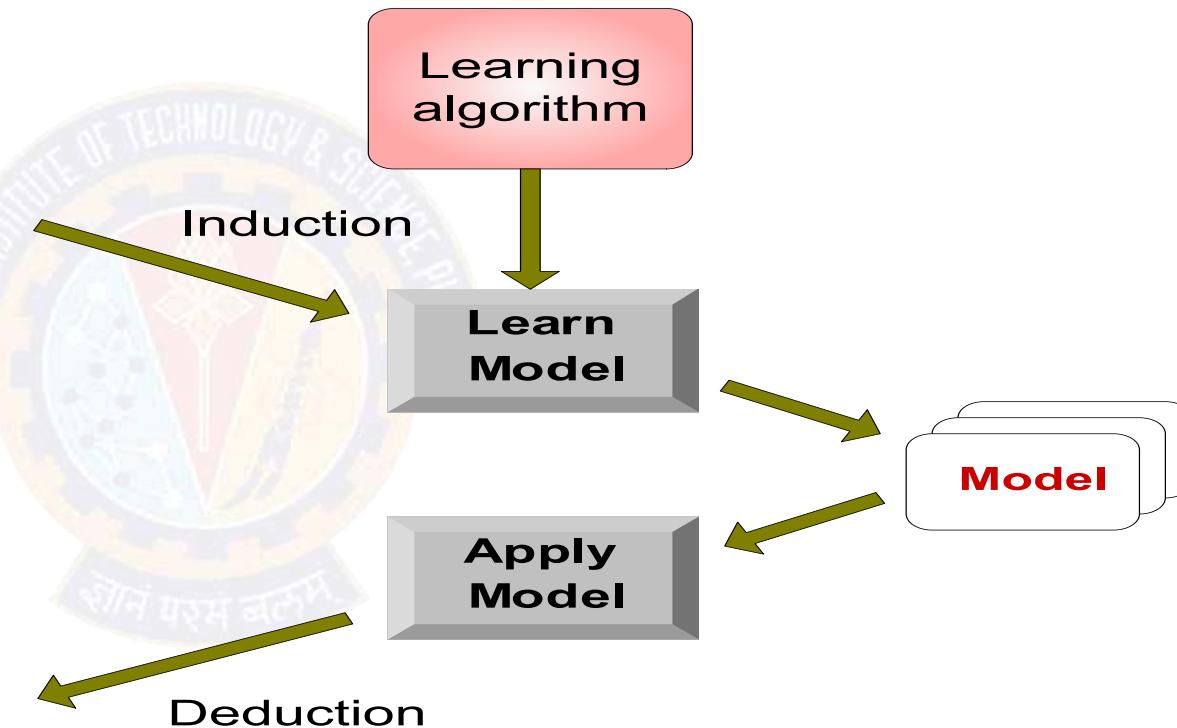
## Illustrating Classification Task

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

Training Set

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

Test Set



# Classification

## Techniques used in Classification

- **Logistic Regression**
  - Allows to model the probability of a particular event or class. It uses a logistic to model a binary dependent variable.
- **Decision Tree based Methods**
  - It is a flowchart like structure. Here, every internal node refers to a test on a condition, and each branch stands for an outcome of the test (whether it's true or false). Every leaf node in a decision tree holds a class label.
- **Neural Networks**
  - computational networks that simulate the decision process in neurons (networks of nerve cell)
- **Naïve Bayes and Bayesian Belief Networks**
  - uses the probability theory to find the most likely of the possible classifications
  - the algorithm assumes that every feature is independent of each other and that all the features contribute equally to the outcome.
- **Support Vector Machines**
  - fits a boundary to a region of points that are all alike; uses the boundary to classify a new point

# Classification

## Lazy Vs. Eager Learning

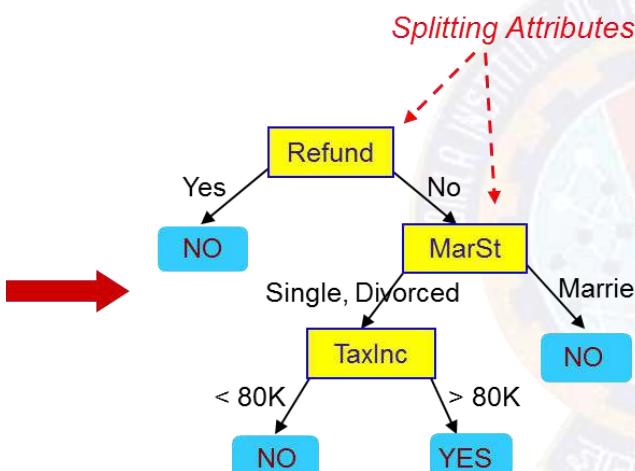
- **Lazy learning (e.g., instance-based learning):**
  - Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Less time in training but more time in predicting
- **Eager learning (the above discussed methods):**
  - Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- **Accuracy**
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

# Classification

## Decision Tree – Example

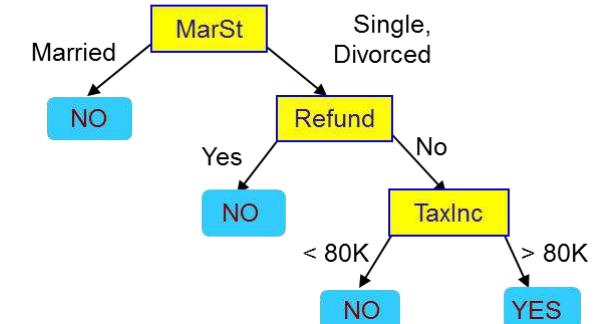
		categorical	categorical	continuous	class
<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat	
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single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

Training Data



Model: Decision Tree

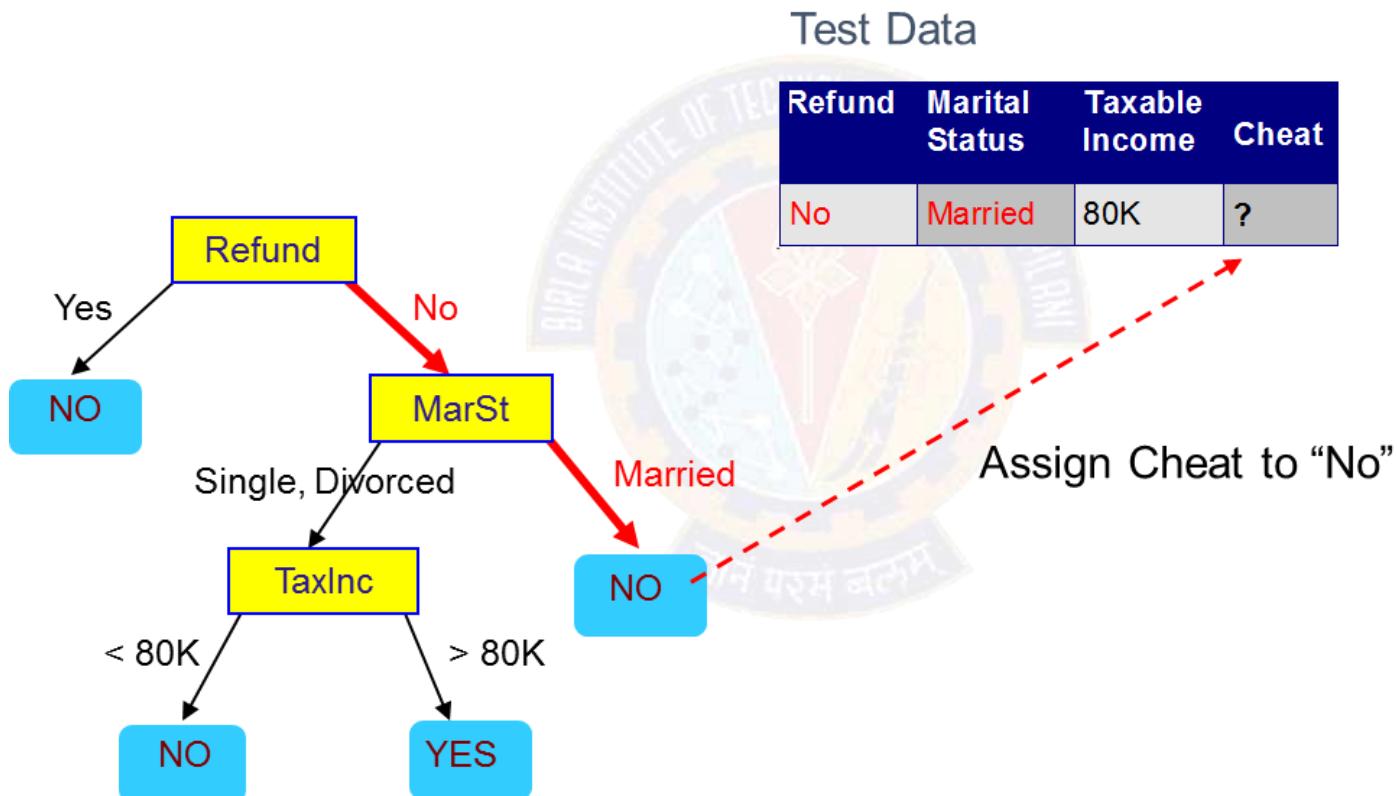
	<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	1	Yes	Single	125K	No
2	2	No	Married	100K	No
3	3	No	Single	70K	No
4	4	Yes	Married	120K	No
5	5	No	Divorced	95K	Yes
6	6	No	Married	60K	No
7	7	Yes	Divorced	220K	No
8	8	No	Single	85K	Yes
9	9	No	Married	75K	No
10	10	No	Single	90K	Yes



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

# Classification

## Apply Decision Tree Model to Test Data



# Classification

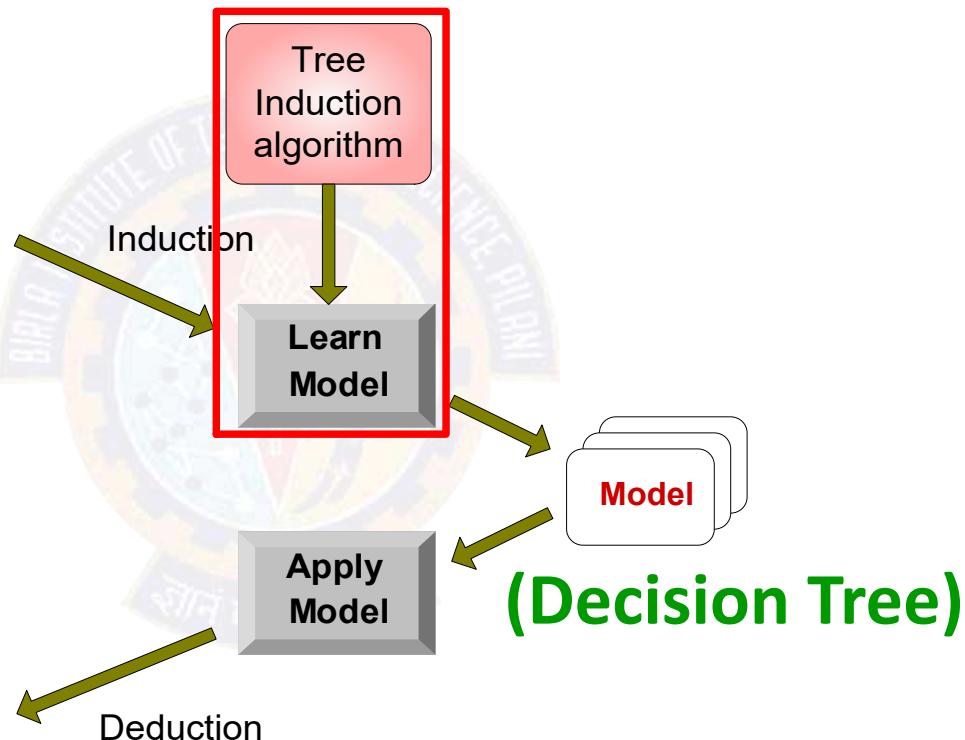
## Decision Tree Classification Task

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

Training Set

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

Test Set



# Classification

## Issues: Evaluating Classification Methods

- Classification methods can be compared and evaluated according to the following criteria:
  - Accuracy
  - Speed
  - Robustness
  - Scalability
  - Interpretability
- Note: This applies to prediction techniques as well.



# Classification

## Issues: Evaluating Classification Methods

- Accuracy
  - The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data (i.e., tuples without class label information).
  - Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.
  - We will discuss accuracy measures later in this module.
  - Accuracy can be estimated using one or more test sets that are independent of the training set.
  - Estimation techniques include techniques such as cross-validation and bootstrapping.
  - Because the accuracy computed is only an estimate of how well the classifier or predictor will do on new data tuples, confidence limits can be computed to help gauge this estimate.

# Classification

## Issues: Evaluating Classification Methods

- **Speed:**

- This refers to the computational costs involved in generating and using the given classifier or predictor.

- **Robustness:**

- This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.

- **Scalability:**

- This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

- **Interpretability:**

- This refers to the level of understanding and insight that is provided by the classifier or predictor.
  - Interpretability is subjective and therefore more difficult to assess.

# In this segment

## Model Selection and Training

- Select based on training data
  - If prediction label/output is available, use regression or classification model
    - Regression if real valued output
    - Classification if output is discrete (binary/integer)
  - else, unsupervised model is used.
- For the house price prediction problem, use regression model since median house prices are available along with training data (predictors)
- Example: **Linear Regression**

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)

>>> from sklearn.metrics import mean_squared_error
>>> housing_predictions = lin_reg.predict(housing_prepared)
>>> lin_mse = mean_squared_error(housing_labels, housing_predictions)
>>> lin_rmse = np.sqrt(lin_mse)
>>> lin_rmse
68628.198198489219
```

- Better model is necessary for improving the prediction accuracy

# Regression Model Selection and Training

- **Decision Tree Based Regression** produces low error on training data

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(housing_prepared, housing_labels)

from sklearn.model_selection import cross_val_score
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
```

```
>>> housing_predictions = tree_reg.predict(housing_prepared)
>>> tree_mse = mean_squared_error(housing_labels, housing_predictions)
>>> tree_rmse = np.sqrt(tree_mse)
>>> tree_rmse
0.0
>>> def display_scores(scores):
...     print("Scores:", scores)
...     print("Mean:", scores.mean())
...     print("Standard deviation:", scores.std())
...
>>> display_scores(tree_rmse_scores)
Scores: [ 70232.0136482  66828.46839892  72444.08721003  70761.50186201
         71125.52697653  75581.29319857  70169.59286164  70055.37863456
         75370.49116773  71222.39081244]
Mean: 71379.0744771
Standard deviation: 2458.31882043
```

- Cross-validation error is not satisfactory
- Better accuracy can be obtained using **Random Forest based regressor**

```
>>> from sklearn.ensemble import RandomForestRegressor
>>> forest_reg = RandomForestRegressor()
>>> forest_reg.fit(housing_prepared, housing_labels)
>>> [...]
>>> forest_rmse
21941.911027380233
>>> display_scores(forest_rmse_scores)
Scores: [ 51650.94405471  48920.80645498  52979.16096752  54412.74042021
         50861.29381163  56488.55699727  51866.90120786  49752.24599537
         55399.50713191  53309.74548294]
Mean: 52564.1902524
Standard deviation: 2301.87380392
```

# Classification Model

## MNIST Dataset Classification

- A set of 70,000 small images of handwritten digits
  - Each image 28x28 pixel with intensity 0 (black) – 255 (white)
  - Input data represented as 70000 x 784 matrix
  - Each image is labeled with the digit it represents.

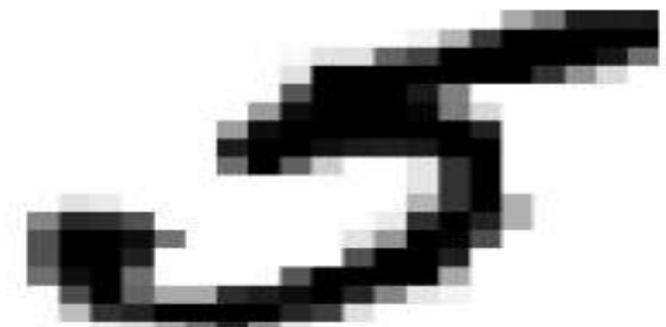
```
>>> from sklearn.datasets import fetch_mldata
>>> mnist = fetch_mldata('MNIST original')
>>> mnist
{'COL_NAMES': ['label', 'data'],
 'DESCR': 'mldata.org dataset: mnist-original',
 'data': array([[0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 ...,
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),
 'target': array([ 0.,  0.,  0., ..., 9., 9., 9.])}

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

some_digit = x[36000]
some_digit_image = some_digit.reshape(28, 28)

plt.imshow(some_digit_image, cmap = matplotlib.cm.binary,
           interpolation="nearest")

plt.axis("off")
plt.show()
```



# Classification Model Training

## Detect a '5'

- Segment the dataset into 60,000 training images and 10,000 test images

```
x_train, x_test, y_train, y_test = x[:60000], x[60000:], y[:60000], y[60000:]
```

- Shuffle the training dataset

```
import numpy as np

shuffle_index = np.random.permutation(60000)
x_train, y_train = x_train[shuffle_index], y_train[shuffle_index]
```

- Train a classification model for detecting '5'. Target output for training data instance corresponding an image of '5' is +1, else target output is '0'

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)

from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(x_train, y_train_5)
```

- Perform cross validation like the regression problem and try out multiple classification model for achieving acceptable performance.

# Multiclass Classification

- Multiclass classifiers (*aka* multinomial classifiers) can distinguish between more than two classes.
- Some algorithms (such as Random Forest classifiers or naive Bayes classifiers) are capable of handling multiple classes directly.
- Many (such as Support Vector Machine classifiers or Linear classifiers) are strictly binary
- **One-versus-all (OvA) or One-versus-rest strategy using multiple binary classifiers.**
  - e.g., for MNIST classification, train 10 binary classifiers, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).
  - get the decision score from each classifier for that image and select the class whose classifier outputs the highest score.
- **One-versus-one (OvO) strategy**
  - train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on.
  - If there are  $N$  classes, you need to train  $N \times (N - 1) / 2$  classifiers.
  - Run an image through all 45 classifiers and see which class wins the most duels.
- Main advantage of OvO is each classifier only needs to be trained on the part of the training set for the two classes that it must distinguish

# Model Evaluation



# Model Evaluation – Confusion Matrix

- The confusion matrix is a useful tool for analyzing how well the classifier can recognize tuples of different classes
- For two classes, a **confusion matrix** is a 2 by 2 table
- $TP$  and  $TN$  - Tell us when the classifier is getting things right
- $FP$  and  $FN$  - Tell us when the classifier is getting things wrong (i.e., mislabeling)
- $P$  is the number of tuples that are actually positive
- $N$  is the number of tuples that are actually negative
- $P'$  is the number of tuples that are predicted as positive ( $TP + FP$ )
- $N'$  is the number of tuples that are predicted as negative ( $TN + FN$ )
- The total number of tuples
  - $TP + TN + FP + FN = P + N = P' + N'$

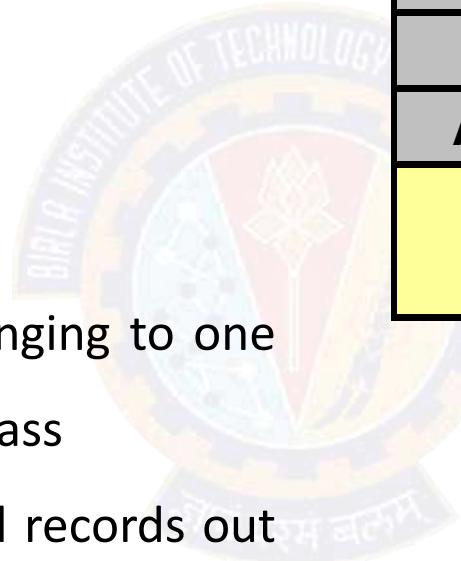
		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

# Model Evaluation – Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

# Model Evaluation – Misclassification Error

- 1's correctly classified as "1" = 201
- 1's incorrectly classified as "0" = 85
- 0's incorrectly classified as "1" = 25
- 0's correctly classified as "0" = 2689
- Misclassification
  - **Error:** classifying a record as belonging to one class when it belongs to another class
  - **Error rate:** percent of misclassified records out of the total records in the validation data



		Classification Confusion Matrix	
		Predicted Class	
		1	0
Actual Class	1	201	85
0	1	25	2689

# Model Evaluation – Cost Matrix

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

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

# Model Evaluation – Computing cost of Classification

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

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

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

- Accuracy = 80%
- Cost = 3910

- Accuracy = 90%
- Cost = 4255

# Model Evaluation – Class Imbalance Problem

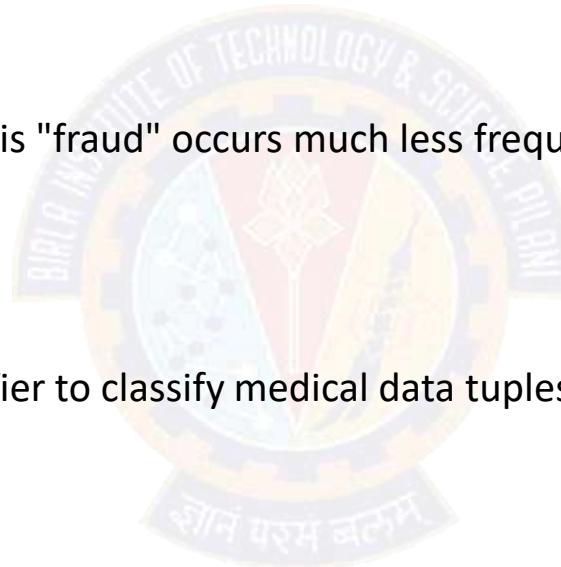
- The class imbalance problem occurs when the main class of interest is rare
- Examples:
  - Fraud detection applications
    - ✓ the class of interest (or positive class) is "fraud" occurs much less frequently than the negative "non-fraudulent" class
  - Medical data
    - ✓ rare class, such as "cancer"
    - ✓ suppose that we have trained a classifier to classify medical data tuples
      - ❖ Cancer = "Yes" or Cancer = "No"

# Model Evaluation – Class Imbalance Problem

- Accuracy rate of, say, 97%
  - may make the classifier seem quite accurate
  - what if only, say, 3% of the training tuples are actually cancer?
- Here, an accuracy rate of 97% may not be acceptable, because
  - the classifier could be correctly labeling only the non-cancer tuples and misclassifying all the cancer tuples
- In this case we need other measures, which assess how well the classifier can recognize the positive tuples (cancer = yes) and negative tuples (cancer = no)

# Model Evaluation – Class Imbalance Problem

- The class imbalance problem occurs when the main class of interest is rare
- Examples:
  - Fraud detection applications
    - ✓ the class of interest (or positive class) is "fraud" occurs much less frequently than the negative "non-fraudulant" class
  - Medical data
    - ✓ rare class, such as "cancer"
    - ✓ suppose that we have trained a classifier to classify medical data tuples
      - ✓ Cancer = "Yes" or Cancer = "No"
- Accuracy rate of, say, 97%
  - may make the classifier seem quite accurate
  - what if only, say, 3% of the training tuples are actually cancer?



# Model Evaluation – Class Imbalance Problem

- **Sensitivity/Recall** (true positive (recognition) rate)
- When it's actually YES, how often does the model predicts YES?

$$\text{Sensitivity or Recall} = \frac{TP}{TP + FN}$$

- A perfect **recall** score of 1.0 for C means that every item from class C was labeled as C

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

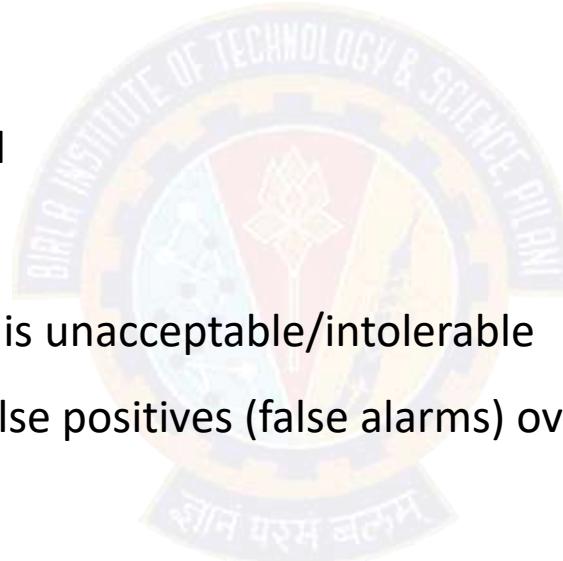
- **Sensitivity/Recall** (true positive (recognition) rate)

- *Important when:*

- Identifying true positives is crucial

- *Used when:*

- The occurrence of false negatives is unacceptable/intolerable
    - We can rather have some extra false positives (false alarms) over saving some false negatives
    - E.g., when predicting deadly disease (cancer) or financial default



		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Actual Class

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

## ■ Specificity (true negative rate)

- When it's actually NO, how often does it predict NO?

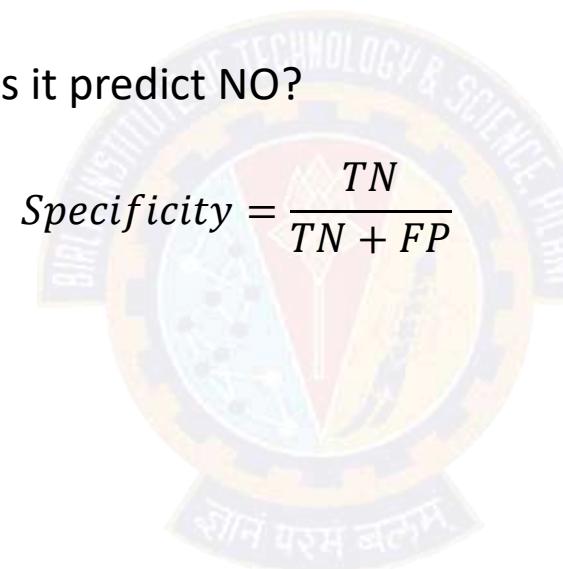
$$\text{Specificity} = \frac{TN}{TN + FP}$$

## ■ *Important when:*

- You want to cover all true negatives

## ■ *Used when:*

- You don't want to raise false alarms. Occurrence of false positives is not acceptable.
- E.g., you're running a drug test in which all people who test positive will immediately go to jail



Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

- Accuracy can be expressed a function of sensitivity & specificity

$$\text{Accuracy} = \text{Sensitivity} \left( \frac{P}{P + N} \right) + \text{Specificity} \left( \frac{N}{P + N} \right)$$

		Predicted Class		Sensitivity $\frac{TP}{(TP + FN)}$	Specificity $\frac{TN}{(TN + FP)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$
		Positive	Negative			
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>			
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)			
	Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$				

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

# Model Evaluation – Class Imbalance Problem

## ■ Precision (measure of *exactness*)

- When it predicts YES, how often is it correct?

$$Precision = \frac{TP}{TP + FP}$$

- A perfect **precision** score of 1.0 for a class C means that every tuple that the classifier labeled as belonging to class C does indeed belong to class C

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Actual Class

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

- **Precision** (measure of *exactness*)

- *Important when:*

- ✓ you want to be more confident of your predicted positives.

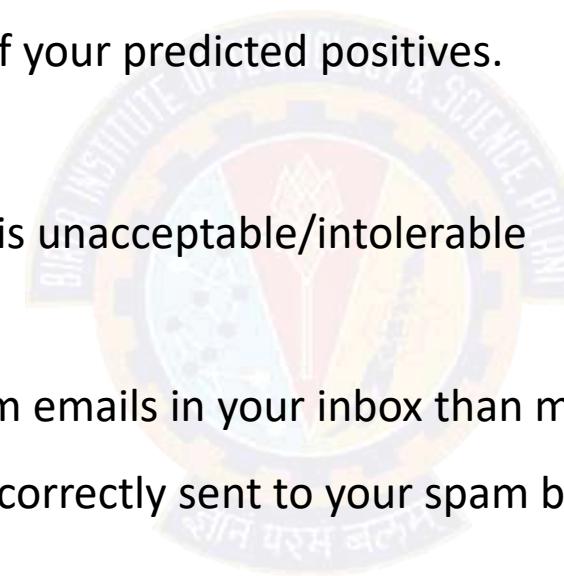
- *Used when:*

- ✓ the occurrence of false positives is unacceptable/intolerable

- ✓ For example, Spam emails

- ✓ We would rather have some spam emails in your inbox than miss out

- some regular emails that were incorrectly sent to your spam box.



A confusion matrix table illustrating the performance of a classification model for the 'Spam' class. The table is a 2x2 grid with 'Actual Class' on the rows and 'Predicted Class' on the columns. The columns are labeled 'Spam' and 'Non-Spam'. The rows are labeled 'Spam' and 'Non-Spam'. The matrix values are: TP=45 (Actual Spam, Predicted Spam), FN=20 (Actual Spam, Predicted Non-Spam), FP=5 (Actual Non-Spam, Predicted Spam), and TN=30 (Actual Non-Spam, Predicted Non-Spam). A pink curly brace on the left side groups the 'Actual Class' rows, and a pink curly brace at the top groups the 'Predicted Class' columns.

		Predicted Class	
		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

Classes	Predicted class		Total	Recognition (%)
Actual class ↓	Yes	No		
Yes	90	210	300	30.00 (Sensitivity)
No	140	9560	9700	98.56 (Specificity)
Total	230	9770	10000	96.50 (Accuracy)

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Confusion matrix for the classes cancer = yes and cancer = no

$$Sensitivity = \frac{TP}{P} = \frac{90}{300} = 30.00\%$$

$$Specificity = \frac{TN}{N} = \frac{9560}{9700} = 98.56\%$$

$$Overall\ Accuracy = 30.00 \times \left( \frac{300}{10000} \right) + 98.56 \times \left( \frac{9700}{10000} \right) = 0.9 + 95.60 = 96.50$$

# Model Evaluation – F Measure

## F measure and $F_\beta$ measure

- An alternative way to use precision and recall is to combine them into a single measure
- This is the approach of the  $F$  measure (also known as the  $F_1$  score or  $F$ -score) and  $F_\beta$  measure
- The  $F$  measure is the harmonic mean of precision and recall. It gives equal weight to precision and recall

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$

- Where
  - $\beta$  is a non-negative real number
  - The  $F_\beta$  measure is a weighted measure of precision and recall. It assigns  $\beta$  times as much weight to recall as to precision
  - Commonly used  $F_\beta$  measures are:
    - ✓  $F_2$  (weights recall twice as much as precision) (False negative has more impact than false positive, choose a beta value greater than 1)
    - ✓  $F_{0.5}$  (weights precision twice as much as recall) (False positive has more impact than false negative, you reduce beta value between 1 & 0)

# Model Evaluation

## F measure and $F_\beta$ measure

- *Important when:*
  - you have an uneven class distribution.
- *Used when:*
  - the cost of false positives and false negatives are different
- **F1 score**
  - Is higher if there is a balance between Precision and Recall
  - F1 Score isn't so high if one of these measures, Precision or Recall, is improved at the expense of the other.
- Each one of these is defined in such a way that they capture different aspects of a model's performance
- When we are choosing one of these metrics to improve our model on, we need to keep in mind:
  - The problem that we are trying to solve
  - The dataset that we have with us (prevalence of each class in the data)
  - The cost that we have to pay for either types of misclassifications (false positives and false negatives)

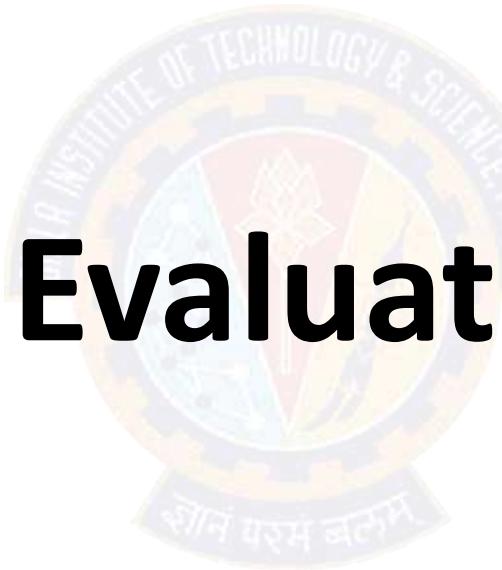
# Model Evaluation - Summary

- Two cases
  - The class tuples are more or less evenly distributed
  - Classes are unbalanced (e.g., where an important class of interest is rare such as in medical tests)
- The classifier evaluation measures:
  - Accuracy (also known as recognition rate)
  - Sensitivity (or recall)
  - Specificity
  - Precision
  - $F_1$
  - $F_\beta$



Measure	Formula
accuracy, recognition rate	$\frac{TP + TN}{P + N}$
error rate, misclassification rate	$\frac{FP + FN}{P + N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP + FP}$
$F$ , $F_1$ , $F$ -score, harmonic mean of precision and recall	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
$F_\beta$ , where $\beta$ is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

# Methods of Evaluation



# Methods of Evaluation – Data Selection Tech.

- Holdout Method
  - In this method, given data are randomly partitioned into two independent sets:
    - ✓ Training set (2/3 of data) for model construction
    - ✓ Test set (1/3 of data) for accuracy estimation
  - The estimate is pessimistic because only a portion of the initial data is used to derive the model
- Random subsampling
  - A variation of the holdout method
  - Here, the holdout method is repeated  $k$  times
  - The overall accuracy = average of accuracies obtained from each iteration

# Methods of Evaluation - Data Selection Tech.

## Data Selection Techniques

- K-fold Cross Validation
  - $k$ -fold cross validation, where  $k=10$  is most popular
  - The initial data are randomly partitioned into  $k$  mutually exclusive subsets or "folds", each of approximately equal size
    - $D_1, D_2, \dots, D_k$
  - Training and testing is performed  $k$  times. For example:
    - In iteration  $i$ , partition  $D_i$  is reserved as the test set, and the remaining partitions are collectively used to train the model
  - Unlike the holdout and random subsampling methods, here each sample is used the same number of times for training and once for testing
  - The accuracy estimate is the overall number of correct classifications from the  $k$  iterations, divided by the total number of tuples in the initial data

# Methods of Evaluation - Data Selection Tech.

## Data Selection Techniques

### ■ Cross Validation

- Leave-one-out

- ✓ A special case of  $k$ -fold cross-validation where  $k$  is set to the number of initial tuples
  - ✓ That is, only one sample is "left out" at a time for the test set

- Stratified cross-validation

- ✓ The folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data
  - ✓ In general, stratified 10-fold cross-validation is recommended for estimating accuracy
    - ❖ due to its relatively low bias and variance

# Methods of Evaluation - Data Selection Tech.

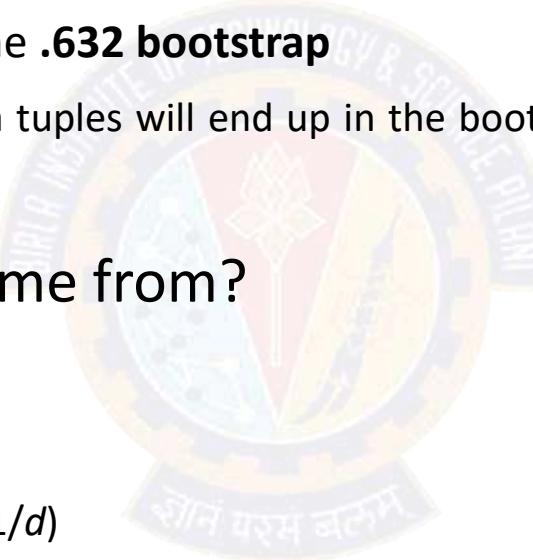
## Data Selection Techniques

### ■ Bootstrap

- Samples the given training tuples uniformly *with replacement*
- Commonly used bootstrap method is the **.632 bootstrap**
  - ✓ On average, 63.2% of the original data tuples will end up in the bootstrap sample, and the remaining 36.8% will form the test set.

### ■ Where does the figure, 63.2%, come from?

- For each tuple:
  - ❖ probability getting selected =  $1/d$
  - ❖ probability of not being chosen is  $(1 - 1/d)$
- if you select  $d$  times probability that a tuple will not be chosen during this whole time is  $(1 - 1/d)^d$
- for a large  $d$ , the probability approaches  $e^{-1} = 0.368$
- Thus, 36.8% of tuples will not be selected (remaining 63.2% will form the training set)



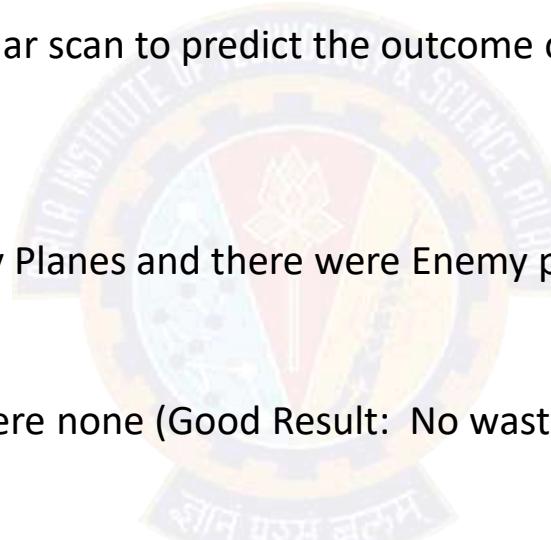
# ROC Curves



# ROC Curves

## History

- Originated from signal detection theory
- Applied in WW2 Battle of Britain to address:
  - Accurately identifying the signals on the radar scan to predict the outcome of interest – Enemy planes – when there were many extraneous signals (e.g. Geese)?
- True Positives
  - Radar Operator interpreted signal as Enemy Planes and there were Enemy planes (Good Result: No wasted Resources)
- True Negatives
  - Radar Operator said no planes and there were none (Good Result: No wasted resources)
- False Positives
  - Radar Operator said planes, but there were none (Geese: wasted resources)
- False Negatives
  - Radar Operator said no plane, but there were planes (Bombs dropped: very bad outcome)

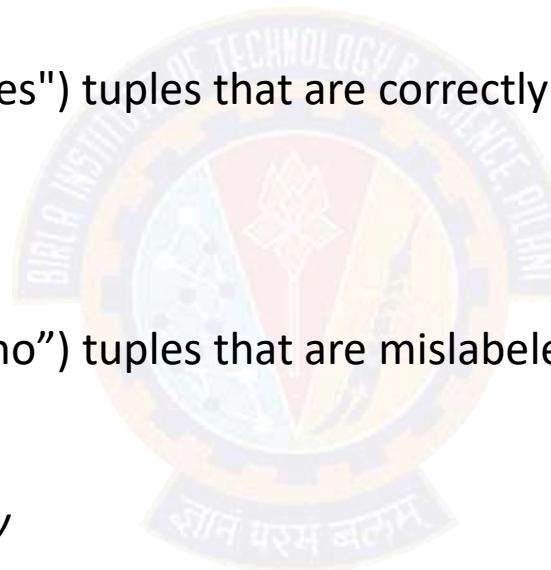


Truth	Predictions	
	POSITIVE CLASS	NEGATIVE CLASS
POSITIVE CLASS	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

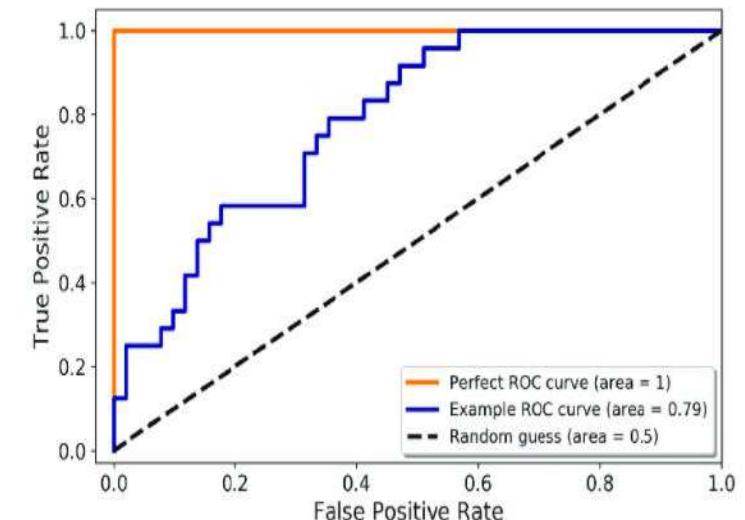
# ROC Curves

## Overview

- ROC curves are a useful visual tool for comparing two binary classification models
- Given TP, TN, FP, & FN
  - TPR is the proportion of positive (or "yes") tuples that are correctly labeled by the model  
 $= \text{TP}/(\text{TP}+\text{FN})$ , which is *sensitivity*
  - FPR is the proportion of negative (or "no") tuples that are mislabeled as positive  
 $= \text{FP}/(\text{FP}+\text{TN})$ , which is  $1 - \text{specificity}$
- An ROC curve for a given model shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR)



		Predictions	
		POSITIVE CLASS	NEGATIVE CLASS
Truth	POSITIVE CLASS	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
	NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)



# ROC Curves

## Optimal Probability Threshold

- Consider an email classification model that detects suspicious communication between terrorists over regular email
- Here, terrorist email is Class YES and a non-terrorist email is Class NO
- It is absolutely necessary for the model to identify the terrorists' emails correctly (True Positive Rate)
- That is, we choose Sensitivity as a metric to improve this model because
  - the occurrence of false negatives is unacceptable
- In this pursuit, we might end up having a few false positives/false alarms but that might be a compromise that we'll have to make.
- Let us say we develop 2 good models which have the same Sensitivity score
- Does that mean the two models have equal predictive power? NO

# ROC Curves

## Optimal Probability Threshold

- Most classification algorithms predict the probability that an observation belongs to class YES
- We need to decide a threshold for these probabilities, to classify the observations into one of the two classes
  - The observation having probability higher than the threshold are classified as class YES.
- For example:
  - the model predicts the probability of an email being a terrorist email as 0.75
  - If we set the threshold as 0.8, then we will classify this email as non-terrorist email
  - If we set the threshold as 0.7, then we will classify the email as a terrorist email
  - So, the performance of our system would vary as we change this threshold
  - This threshold can be adjusted to tune the behavior of the model for a specific problem
  - That is, we can reduce more of one or another type of error (FP/FN).

# ROC Curves

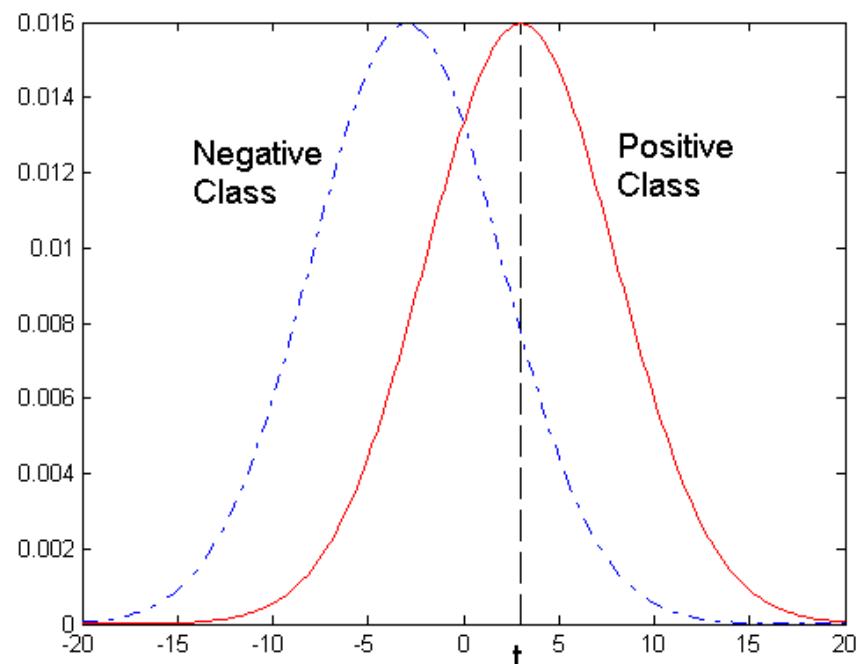
## Optimal Probability Threshold

- Two models with the **same sensitivity (TPR)** are **not equivalent**
- Among these two, the model with a **lower FPR** is obviously a better, more reliable model
- We do not want to waste any of our investigative resources on non-terrorist emails that were misclassified as terrorist emails
- The threshold that we set, can help us increase or decrease the TPR
- If we choose a low threshold, more emails will be classified as terrorist emails, we will be able to catch more true positives but then even the false positive rate would increase
- The choice of threshold entails a trade-off between false positives and false negatives.
- An ROC curve is a useful resource in this regard.

# ROC Curves

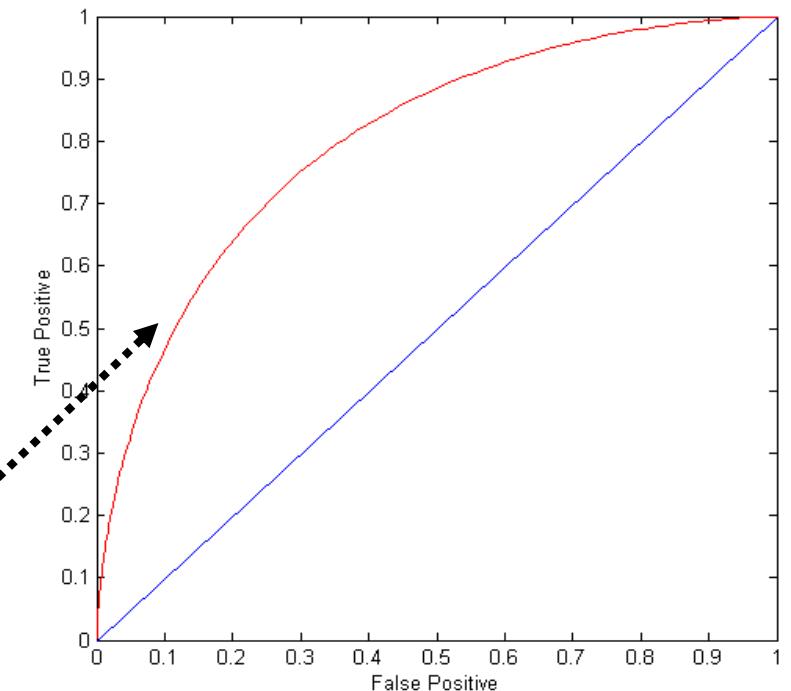
## Optimal Probability Threshold

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



At threshold  $t$ :

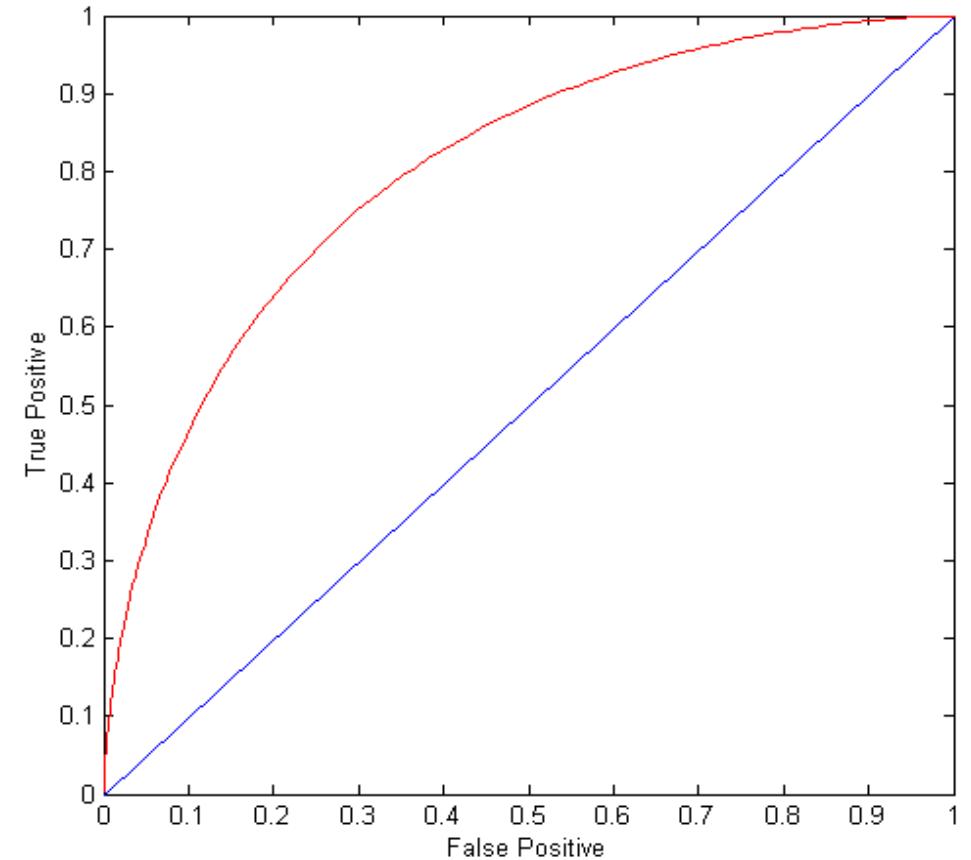
$TP=0.5$ ,  $FN=0.5$ ,  $FP=0.12$ ,  $TN=0.88$



# ROC Curves

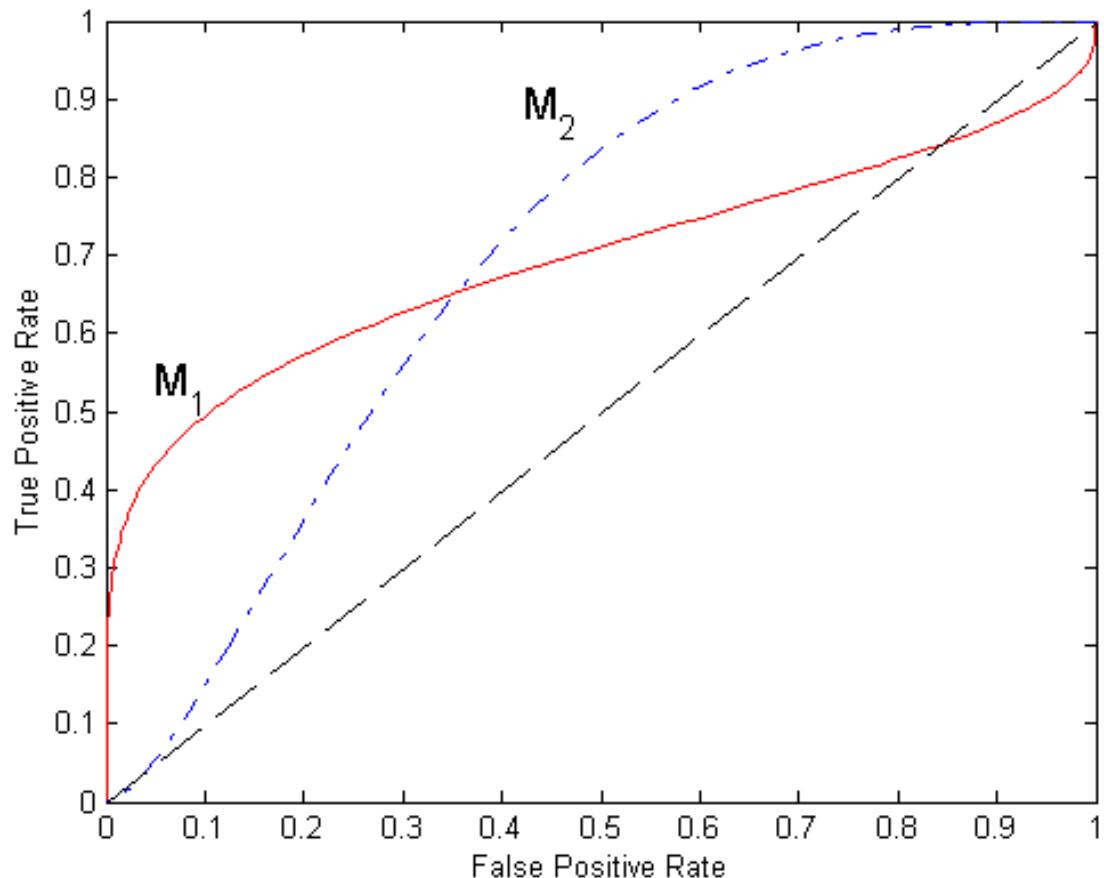
## Optimal Probability Threshold

- (TP, FP):
  - (0,0): declare everything to be negative class
  - (1,1): declare everything to be positive class
  - (1,0): ideal
- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



# ROC Curves

## Optimal Probability Threshold



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

# ROC Curves

## Example

0.90		Predicted	
		Yes	No
Actual	Yes	1	4
	No	0	5

0.80		Predicted	
		Yes	No
Actual	Yes	2	3
	No	0	5

0.70		Predicted	
		Yes	No
Actual	Yes	2	3
	No	1	4

0.60		Predicted	
		Yes	No
Actual	Yes	3	2
	No	1	4

0.55		Predicted	
		Yes	No
Actual	Yes	4	1
	No	1	4

0.54		Predicted	
		Yes	No
Actual	Yes	4	1
	No	2	3

0.53		Predicted	
		Yes	No
Actual	Yes	4	1
	No	3	2

0.51		Predicted	
		Yes	No
Actual	Yes	4	1
	No	4	1

0.50		Predicted	
		Yes	No
Actual	Yes	5	0
	No	4	1

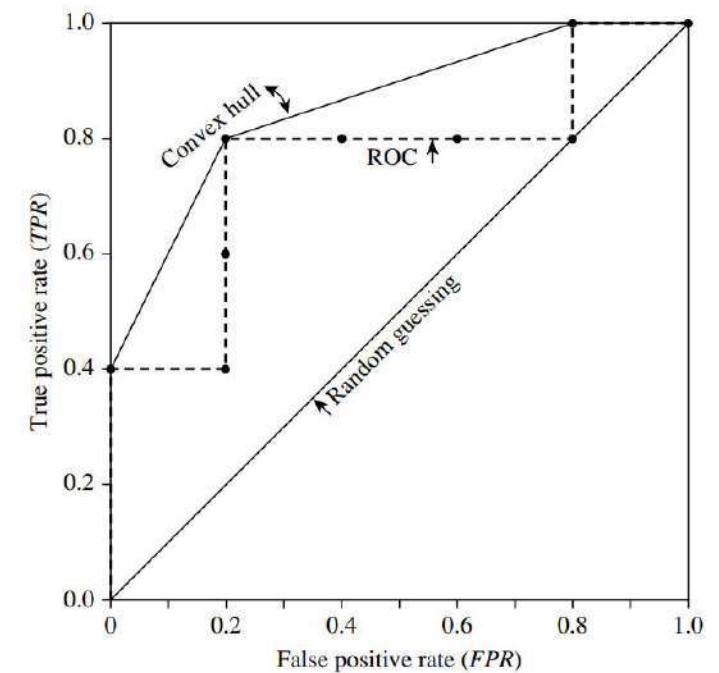
Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	0	1	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0

		Predictions	
		POSITIVE CLASS	NEGATIVE CLASS
Truth	POSITIVE CLASS	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
	NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

# ROC Curves

## Example

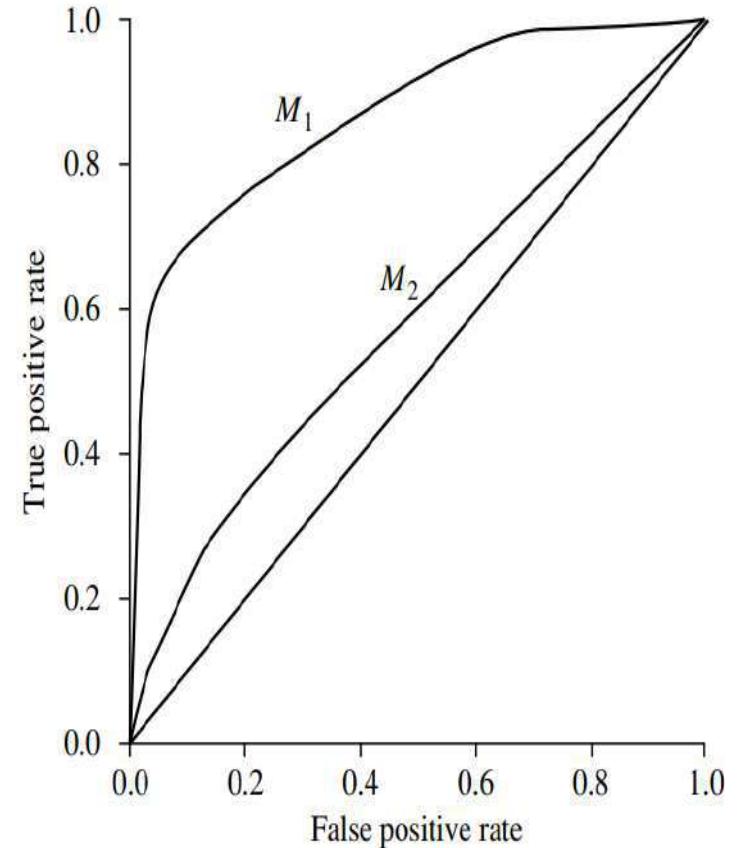
<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	0	1	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0



# ROC Curves

## Comparison between two models

- For a two-class problem, an ROC curve allows us to visualize the trade-off between the rate at which the model can accurately recognize positive cases versus the rate at which it mistakenly identifies negative cases as positive for different portions of the test set
- Any increase in  $TPR$  occurs at the cost of an increase in  $FPR$
- The area under the ROC curve is a measure of the accuracy of the model
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



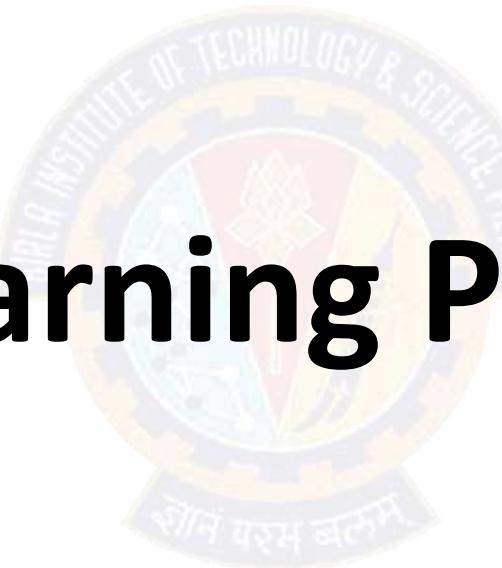
# Hyperparameters

Machine Learning systems use many parameters internally

- Gradient Descent
  - e.g., Learning rate, how long to run
- Mini-batch
  - Batch size
- Regularization constant



# Machine Learning Pipeline



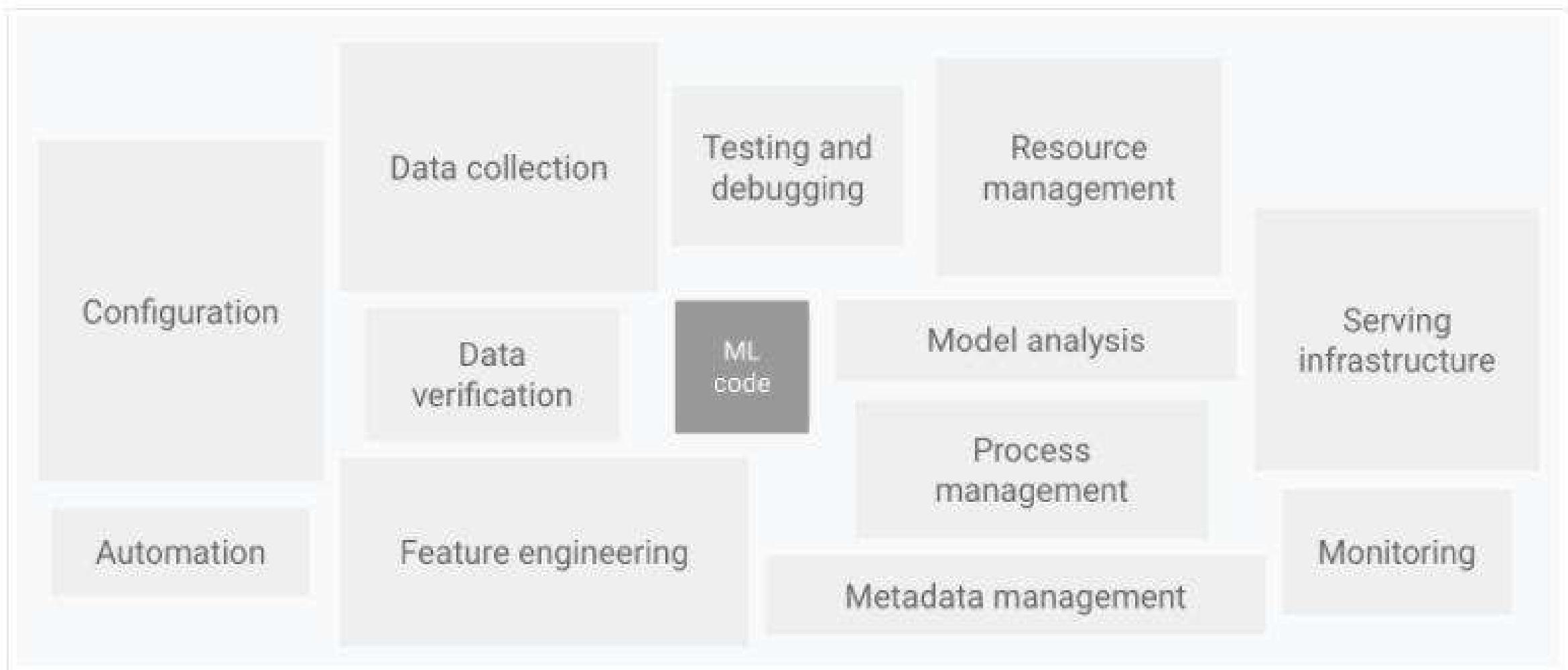
# What is MLOps?

## Apply DevOps principles to ML systems

- An engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops).
- Automation and monitoring at all steps of ML system construction, including integration, testing, releasing, deployment and infrastructure management.
- Data scientists can implement and train an ML model with predictive performance on an offline validation (holdout) dataset, given relevant training data for their use case.
- However, the real challenge is building an integrated ML system and to continuously operate it in production.

# Ecosystem of ML System Components

A small fraction of a real-world ML system is composed of the ML code



# DevOps Vs. MLOps

- DevOps for developing and operating large-scale software systems provides benefits such as
  - shortening the development cycles
  - increasing deployment velocity, and
  - dependable releases.
- Two key concepts
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
- An ML system is a software system, so similar practices apply to reliably build and operate at scale.
- However, ML systems differ from other software systems
  - **Team skills:** focus on exploratory data analysis, model development, and experimentation.
  - **Development:** ML is experimental in nature.
    - The challenge is tracking what worked and what did not, maintaining reproducibility, and maximizing code reusability.
  - **Testing:** Additional testing needed for data validation, trained model quality evaluation, and model validation.

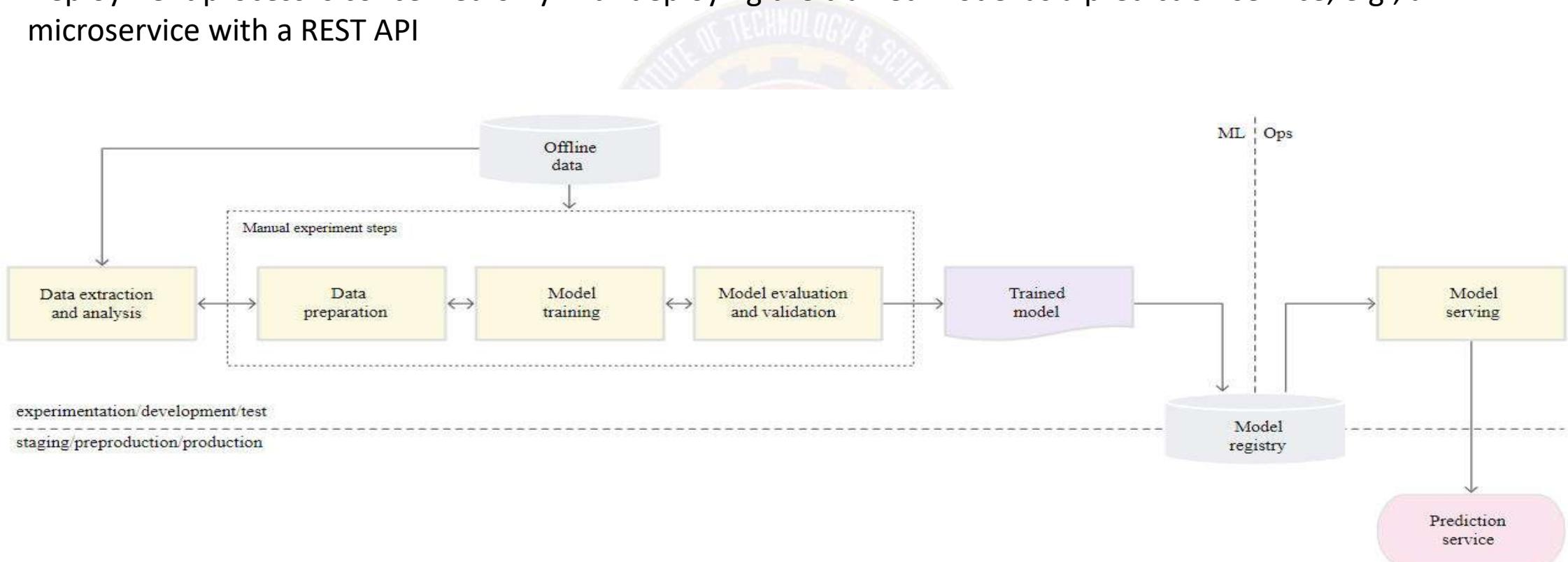


# DevOps Vs. MLOps

- **Deployment:** a multi-step pipeline to automatically retrain and deploy model.
  - adds complexity
  - Automation needed before deployment by data scientists to train and validate new models.
- **Production:** ML models can have reduced performance due to constantly evolving data profiles.
  - Need to track summary statistics of data and
  - monitor the online performance of model to send notifications or roll back for suboptimal values
- ML and other software systems are similar in CI of source control, unit / integration testing, and CD of the software module / package.
- However, in ML,
  - CI is also about testing and validating data, data schemas, and models.
  - CD is a system (an ML training pipeline) that automatically deploys another service (model prediction service).
- Continuous training (CT) is a new property, unique to ML systems, that is concerned with automatically retraining the model in production and serving the models.

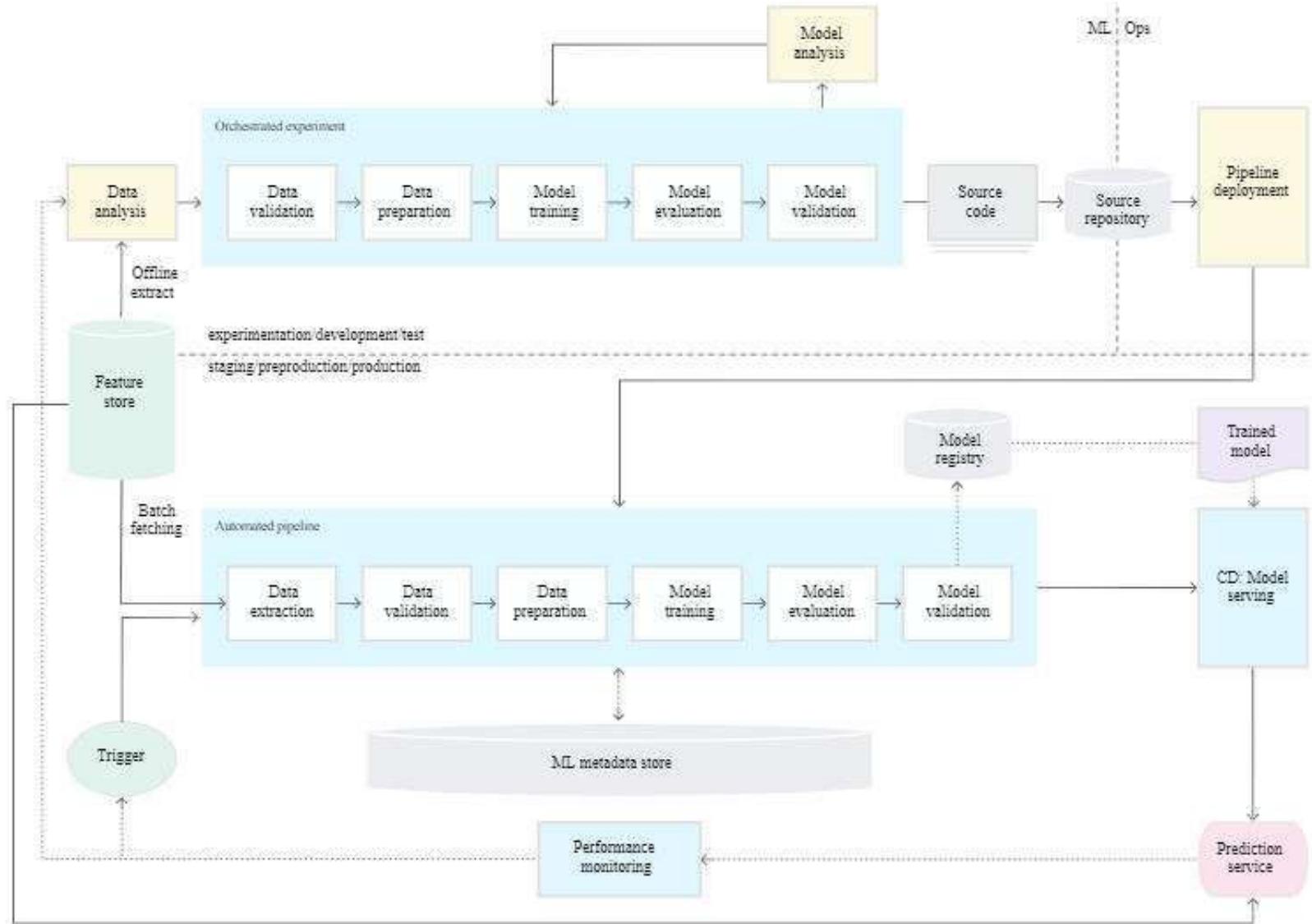
# Manual ML Steps

- Manual, script-driven, and interactive process.
- Disconnection between ML and operations, possibly leading to *training-serving skew*
- Infrequent release iterations. No CI, CD, active performance monitoring
- Deploy trained Model as a prediction service
- Deployment process is concerned only with deploying the trained model as a prediction service, e.g., a microservice with a REST API



# MLOps Level 1

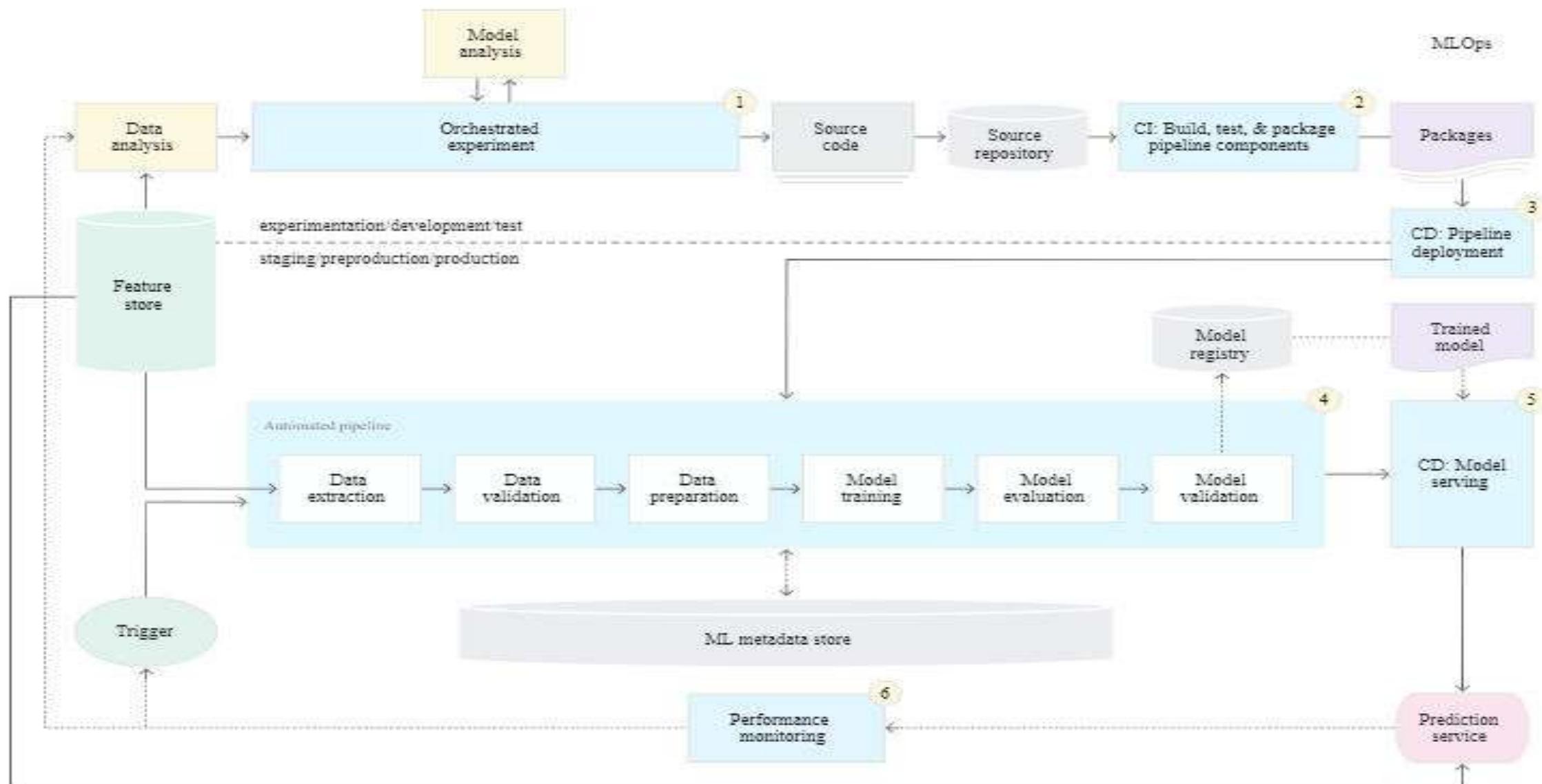
- Perform continuous training (CT) by automating the ML pipeline
- Achieves continuous delivery of model prediction service.
- Automated data and model validation steps to the pipeline
- Needs pipeline triggers and metadata management.



# Data and Model Validation

- **Data validation:** Required prior to model training to decide whether to retrain the model or stop the execution of the pipeline based on following
  - Data values skews: significant changes in the statistical properties of data, triggering retraining
  - Data schema skews: downstream pipeline steps, including data processing and model training, receives data that doesn't comply with the expected schema.
    - stop the pipeline to release a fix or an update to the pipeline to handle these changes in the schema.
    - Schema skews include receiving unexpected features or with unexpected values, not receiving all the expected features
- **Model validation:** Required after retraining the model with the new data. Evaluate and validate the model before promoting to production. This *offline model validation* step consists of
  - Producing evaluation metric using the trained model on test data to assess the model quality.
  - Comparing the evaluation metrics of production model, baseline model, or other business-requirement models.
  - Ensuring the consistency of model performance on various data segments
  - Test model for deployment, including infrastructure compatibility and API consistency
  - Undergo *online model validation*—in a canary deployment or an A/B testing setup

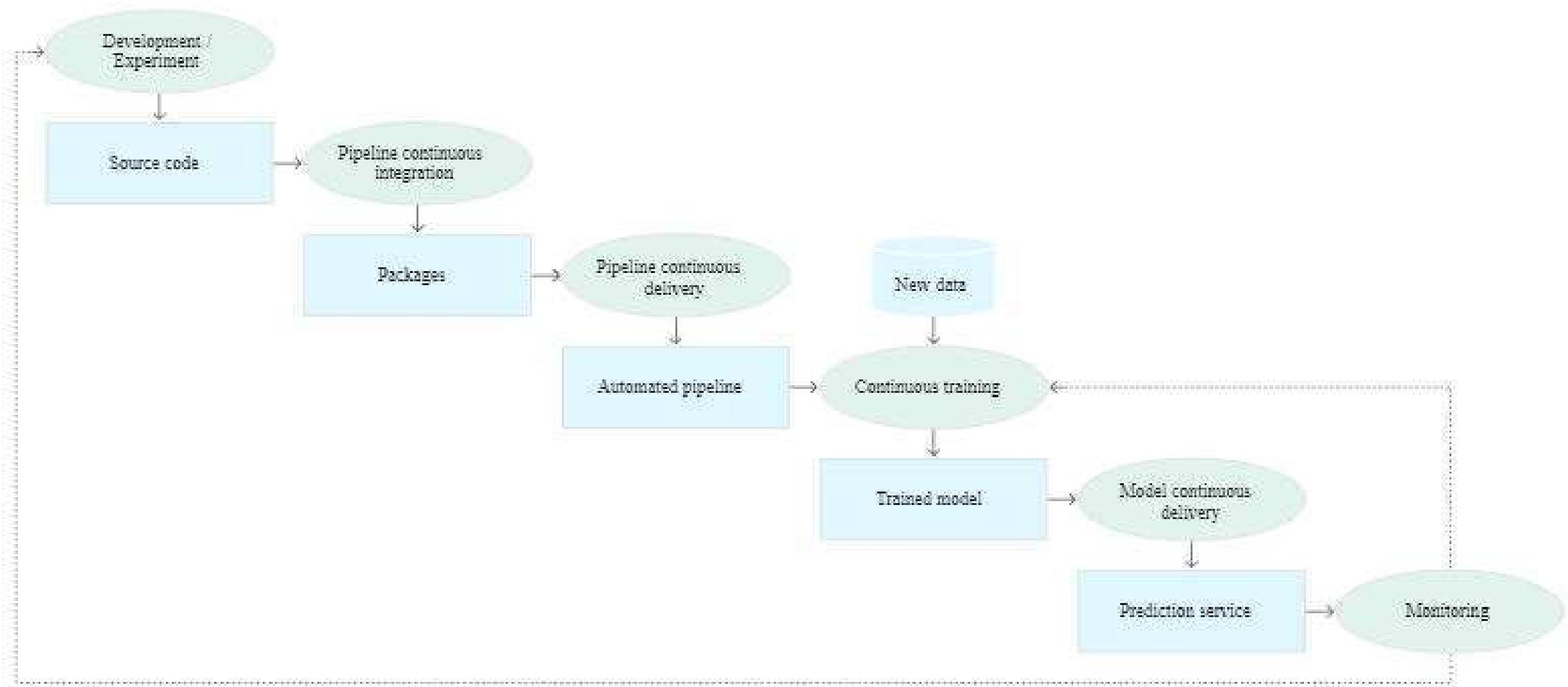
# Level 2: CI/CD and automated pipeline automation



# Stages of CI/CD Automation Pipeline

- **Development and experimentation:** iteratively try new ML algorithms and modeling. The output is the source code of the ML pipeline steps that are then pushed to a source repository.
- **Pipeline continuous integration:** build source code and run various tests. The outputs of this stage are pipeline components (packages, executables, and artifacts).
- **Pipeline continuous delivery:** deploy artifacts produced by the CI stage to the target environment.
- **Automated training:** automatically executed in production based on a schedule or trigger. The output is a trained model pushed to the model registry.
- **Model continuous delivery:** serve the trained model as a prediction service for the predictions.
- **Monitoring:** collect statistics on the model performance based on live data. The output is a trigger to execute the pipeline or to execute a new experiment cycle.

# Stages of the CI/CD automated ML pipeline



# Continuous Integration

- Pipeline and its components are built, tested, and packaged when
  - new code is committed or
  - pushed to the source code repository.

Besides building packages, container images, and executables, CI process can include

- Unit testing feature engineering logic.
- Unit testing the different methods implemented in your model.
  - ❖ For example, you have a function that accepts a categorical data column and you encode the function as a one-hot feature.
- Testing for training convergence
- Testing for NaN values due to dividing by zero or manipulating small or large values.
- Testing that each component in the pipeline produces the expected artifacts.
- Testing integration between pipeline components.

# Continuous Delivery

- Continuously delivers new pipeline implementations to the target environment
  - prediction services of the newly trained model.
- For rapid and reliable continuous delivery of pipelines and models, consider
  - Verifying the compatibility of the model with the target infrastructure
    - e.g., required packages are installed in the serving environment
    - Availability of memory, compute, and accelerator resources.
  - Testing the prediction service by calling the service API for the updated model
  - Testing prediction service performance, such as throughput, latency.
  - Validating the data either for retraining or batch prediction.
  - Verifying that models meet the predictive performance targets prior to deployment.
  - Automated deployment to a test environment, triggered by new code to the development branch.
  - Semi-automated deployment to a pre-production environment, triggered by code merging
  - Manual deployment to a production from pre-production.

# Frameworks

## Cloud Vendors are providing MLOps framework

- <https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning>
  - Kubeflow and Cloud Build
- Amazon AWS MLOps
- Microsoft Azure MLOps





**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 4(19<sup>th</sup> August,2023)**

# Session 4 – 19<sup>th</sup> August, 2023

## 3 Topics from previous session - continuation



# Model Evaluation – Confusion Matrix

- The confusion matrix is a useful tool for analyzing how well the classifier can recognize tuples of different classes
- For two classes, a **confusion matrix** is a 2 by 2 table
- $TP$  and  $TN$  - Tell us when the classifier is getting things right
- $FP$  and  $FN$  - Tell us when the classifier is getting things wrong (i.e., mislabeling)
- $P$  is the number of tuples that are actually positive
- $N$  is the number of tuples that are actually negative
- $P'$  is the number of tuples that are predicted as positive ( $TP + FP$ )
- $N'$  is the number of tuples that are predicted as negative ( $TN + FN$ )
- The total number of tuples
  - $TP + TN + FP + FN = P + N = P' + N'$

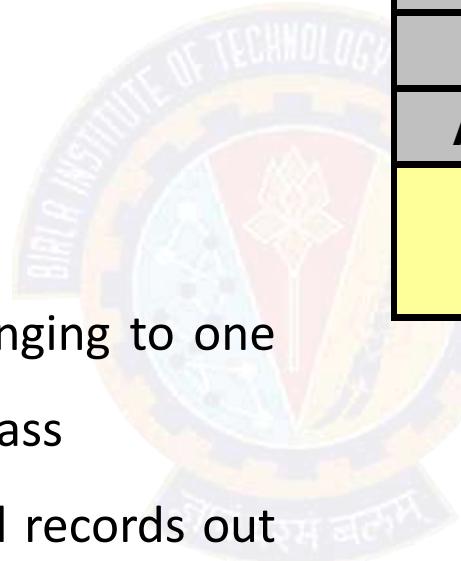
		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

# Model Evaluation – Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

# Model Evaluation – Misclassification Error

- 1's correctly classified as "1" = 201
- 1's incorrectly classified as "0" = 85
- 0's incorrectly classified as "1" = 25
- 0's correctly classified as "0" = 2689
- Misclassification
  - **Error:** classifying a record as belonging to one class when it belongs to another class
  - **Error rate:** percent of misclassified records out of the total records in the validation data



		Classification Confusion Matrix	
		Predicted Class	
		1	0
Actual Class	1	201	85
0	25		2689

# Model Evaluation – Cost Matrix

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

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

# Model Evaluation – Computing cost of Classification

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

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

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

- Accuracy = 80%
- Cost = 3910

- Accuracy = 90%
- Cost = 4255

# Model Evaluation – Class Imbalance Problem

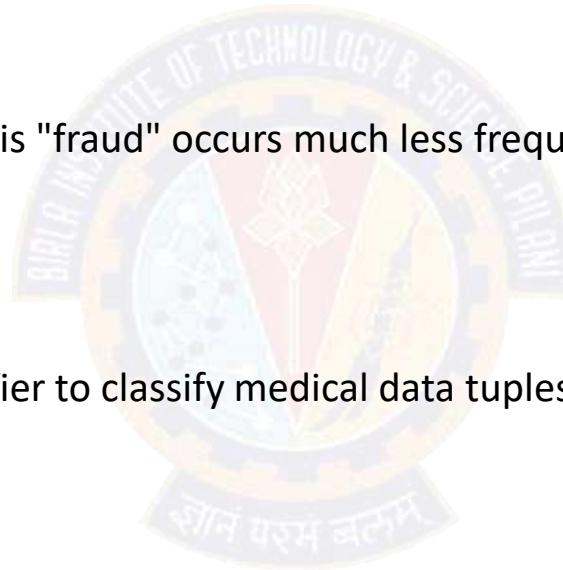
- The class imbalance problem occurs when the main class of interest is rare
- Examples:
  - Fraud detection applications
    - ✓ the class of interest (or positive class) is "fraud" occurs much less frequently than the negative "non-fraudulent" class
  - Medical data
    - ✓ rare class, such as "cancer"
    - ✓ suppose that we have trained a classifier to classify medical data tuples
      - ❖ Cancer = "Yes" or Cancer = "No"

# Model Evaluation – Class Imbalance Problem

- Accuracy rate of, say, 97%
  - may make the classifier seem quite accurate
  - what if only, say, 3% of the training tuples are actually cancer?
- Here, an accuracy rate of 97% may not be acceptable, because
  - the classifier could be correctly labeling only the non-cancer tuples and misclassifying all the cancer tuples
- In this case we need other measures, which assess how well the classifier can recognize the positive tuples (cancer = yes) and negative tuples (cancer = no)

# Model Evaluation – Class Imbalance Problem

- The class imbalance problem occurs when the main class of interest is rare
- Examples:
  - Fraud detection applications
    - ✓ the class of interest (or positive class) is "fraud" occurs much less frequently than the negative "non-fraudulant" class
  - Medical data
    - ✓ rare class, such as "cancer"
    - ✓ suppose that we have trained a classifier to classify medical data tuples
      - ✓ Cancer = "Yes" or Cancer = "No"
- Accuracy rate of, say, 97%
  - may make the classifier seem quite accurate
  - what if only, say, 3% of the training tuples are actually cancer?



# Model Evaluation – Class Imbalance Problem

- **Sensitivity/Recall** (true positive (recognition) rate)
- When it's actually YES, how often does the model predicts YES?

$$\text{Sensitivity or Recall} = \frac{TP}{TP + FN}$$

- A perfect **recall** score of 1.0 for C means that every item from class C was labeled as C

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

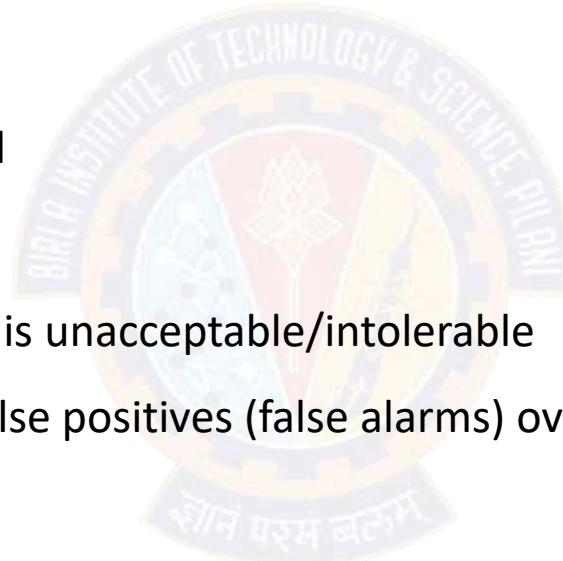
- **Sensitivity/Recall** (true positive (recognition) rate)

- *Important when:*

- Identifying true positives is crucial

- *Used when:*

- The occurrence of false negatives is unacceptable/intolerable
    - We can rather have some extra false positives (false alarms) over saving some false negatives
    - E.g., when predicting deadly disease (cancer) or financial default



		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Actual Class

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

## ■ Specificity (true negative rate)

- When it's actually NO, how often does it predict NO?

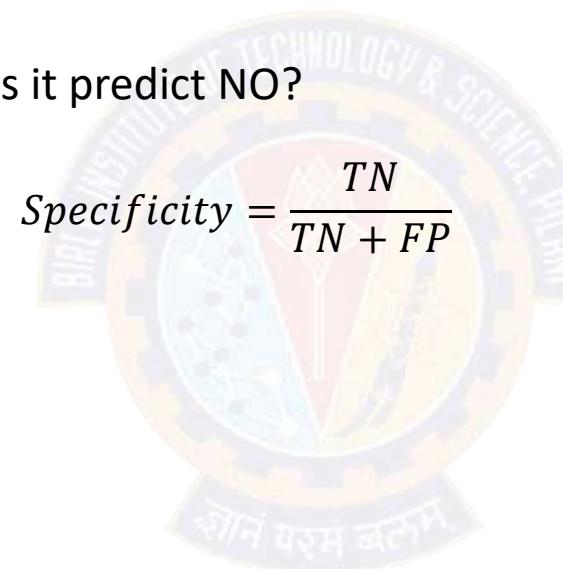
$$\text{Specificity} = \frac{TN}{TN + FP}$$

## ■ *Important when:*

- You want to cover all true negatives

## ■ *Used when:*

- You don't want to raise false alarms. Occurrence of false positives is not acceptable.
- E.g., you're running a drug test in which all people who test positive will immediately go to jail



Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

- Accuracy can be expressed a function of sensitivity & specificity

$$\text{Accuracy} = \text{Sensitivity} \left( \frac{P}{P + N} \right) + \text{Specificity} \left( \frac{N}{P + N} \right)$$

		Predicted Class		Sensitivity $\frac{TP}{(TP + FN)}$	Specificity $\frac{TN}{(TN + FP)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$
		Positive	Negative			
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>			
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)			
	Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$				

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

# Model Evaluation – Class Imbalance Problem

## ■ Precision (measure of *exactness*)

- When it predicts YES, how often is it correct?

$$Precision = \frac{TP}{TP + FP}$$

- A perfect **precision** score of 1.0 for a class C means that every tuple that the classifier labeled as belonging to class C does indeed belong to class C

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Actual Class

Predicted Class

		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

- **Precision** (measure of *exactness*)

- *Important when:*

- ✓ you want to be more confident of your predicted positives.

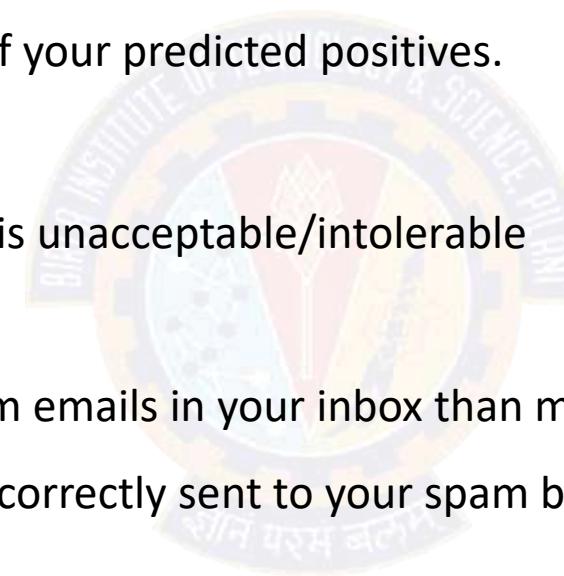
- *Used when:*

- ✓ the occurrence of false positives is unacceptable/intolerable

- ✓ For example, Spam emails

- ✓ We would rather have some spam emails in your inbox than miss out

- some regular emails that were incorrectly sent to your spam box.



A confusion matrix table illustrating the performance of a classification model for the 'Spam' class. The table is a 2x2 grid with 'Actual Class' on the rows and 'Predicted Class' on the columns. The columns are labeled 'Spam' and 'Non-Spam'. The rows are labeled 'Spam' and 'Non-Spam'. The matrix values are: TP=45 (Actual Spam, Predicted Spam), FN=20 (Actual Spam, Predicted Non-Spam), FP=5 (Actual Non-Spam, Predicted Spam), and TN=30 (Actual Non-Spam, Predicted Non-Spam). A pink curly brace on the left side groups the 'Actual Class' rows, and a pink curly brace at the top groups the 'Predicted Class' columns.

		Predicted Class	
		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# Model Evaluation – Class Imbalance Problem

Classes	Predicted class		Total	Recognition (%)
Actual class ↓	Yes	No		
Yes	90	210	300	30.00 (Sensitivity)
No	140	9560	9700	98.56 (Specificity)
Total	230	9770	10000	96.50 (Accuracy)

		Predicted class		Total
		Yes	No	
Actual class	Yes	TP	FN	P
	No	FP	TN	N
		P'	N'	

Confusion matrix for the classes cancer = yes and cancer = no

$$Sensitivity = \frac{TP}{P} = \frac{90}{300} = 30.00\%$$

$$Specificity = \frac{TN}{N} = \frac{9560}{9700} = 98.56\%$$

$$Overall\ Accuracy = 30.00 \times \left( \frac{300}{10000} \right) + 98.56 \times \left( \frac{9700}{10000} \right) = 0.9 + 95.60 = 96.50$$

# Model Evaluation – F Measure

## F measure and $F_\beta$ measure

- An alternative way to use precision and recall is to combine them into a single measure
- This is the approach of the  $F$  measure (also known as the  $F_1$  score or  $F$ -score) and  $F_\beta$  measure
- The  $F$  measure is the harmonic mean of precision and recall. It gives equal weight to precision and recall

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$

- Where
  - $\beta$  is a non-negative real number
  - The  $F_\beta$  measure is a weighted measure of precision and recall. It assigns  $\beta$  times as much weight to recall as to precision
  - Commonly used  $F_\beta$  measures are:
    - ✓  $F_2$  (weights recall twice as much as precision) (False negative has more impact than false positive, choose a beta value greater than 1)
    - ✓  $F_{0.5}$  (weights precision twice as much as recall) (False positive has more impact than false negative, you reduce beta value between 1 & 0)

# Model Evaluation

## F measure and $F_\beta$ measure

- *Important when:*
  - you have an uneven class distribution.
- *Used when:*
  - the cost of false positives and false negatives are different
- **F1 score**
  - Is higher if there is a balance between Precision and Recall
  - F1 Score isn't so high if one of these measures, Precision or Recall, is improved at the expense of the other.
- Each one of these is defined in such a way that they capture different aspects of a model's performance
- When we are choosing one of these metrics to improve our model on, we need to keep in mind:
  - The problem that we are trying to solve
  - The dataset that we have with us (prevalence of each class in the data)
  - The cost that we have to pay for either types of misclassifications (false positives and false negatives)

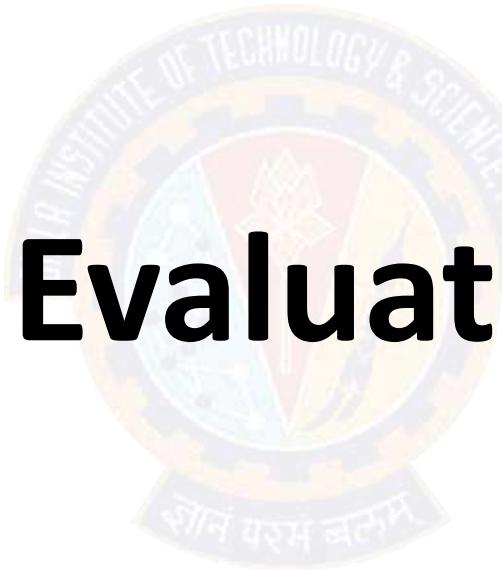
# Model Evaluation - Summary

- Two cases
  - The class tuples are more or less evenly distributed
  - Classes are unbalanced (e.g., where an important class of interest is rare such as in medical tests)
- The classifier evaluation measures:
  - Accuracy (also known as recognition rate)
  - Sensitivity (or recall)
  - Specificity
  - Precision
  - $F_1$
  - $F_\beta$



Measure	Formula
accuracy, recognition rate	$\frac{TP + TN}{P + N}$
error rate, misclassification rate	$\frac{FP + FN}{P + N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP + FP}$
$F$ , $F_1$ , $F$ -score, harmonic mean of precision and recall	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
$F_\beta$ , where $\beta$ is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

# Methods of Evaluation



# Methods of Evaluation – Data Selection Tech.

- Holdout Method
  - In this method, given data are randomly partitioned into two independent sets:
    - ✓ Training set (2/3 of data) for model construction
    - ✓ Test set (1/3 of data) for accuracy estimation
  - The estimate is pessimistic because only a portion of the initial data is used to derive the model
- Random subsampling
  - A variation of the holdout method
  - Here, the holdout method is repeated  $k$  times
  - The overall accuracy = average of accuracies obtained from each iteration

# Methods of Evaluation - Data Selection Tech.

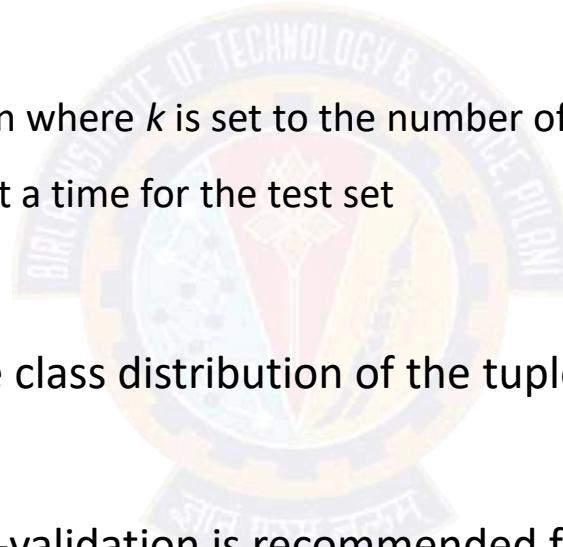
## Data Selection Techniques

- K-fold Cross Validation
  - $k$ -fold cross validation, where  $k=10$  is most popular
  - The initial data are randomly partitioned into  $k$  mutually exclusive subsets or "folds", each of approximately equal size
    - $D_1, D_2, \dots, D_k$
  - Training and testing is performed  $k$  times. For example:
    - In iteration  $i$ , partition  $D_i$  is reserved as the test set, and the remaining partitions are collectively used to train the model
  - Unlike the holdout and random subsampling methods, here each sample is used the same number of times for training and once for testing
  - The accuracy estimate is the overall number of correct classifications from the  $k$  iterations, divided by the total number of tuples in the initial data

# Methods of Evaluation - Data Selection Tech.

## Data Selection Techniques

- Cross Validation
  - Leave-one-out
    - ✓ A special case of  $k$ -fold cross-validation where  $k$  is set to the number of initial tuples
    - ✓ That is, only one sample is "left out" at a time for the test set
  - Stratified cross-validation
    - ✓ The folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data
    - ✓ In general, stratified 10-fold cross-validation is recommended for estimating accuracy
      - ❖ due to its relatively low bias and variance



# Methods of Evaluation - Data Selection Tech.

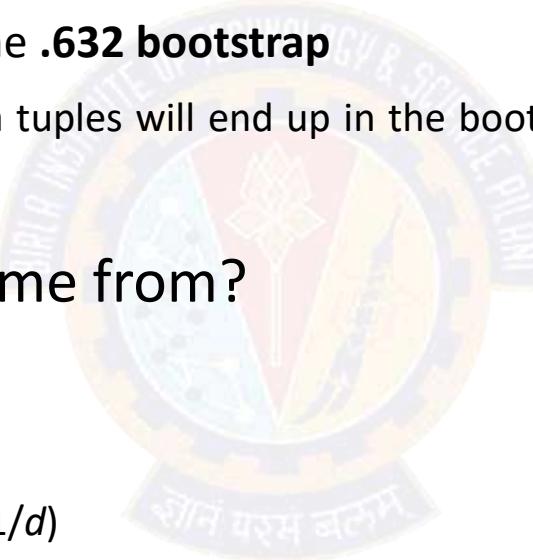
## Data Selection Techniques

### ■ Bootstrap

- Samples the given training tuples uniformly *with replacement*
- Commonly used bootstrap method is the **.632 bootstrap**
  - ✓ On average, 63.2% of the original data tuples will end up in the bootstrap sample, and the remaining 36.8% will form the test set.

### ■ Where does the figure, 63.2%, come from?

- For each tuple:
  - ❖ probability getting selected =  $1/d$
  - ❖ probability of not being chosen is  $(1 - 1/d)$
- if you select  $d$  times probability that a tuple will not be chosen during this whole time is  $(1 - 1/d)^d$
- for a large  $d$ , the probability approaches  $e^{-1} = 0.368$
- Thus, 36.8% of tuples will not be selected (remaining 63.2% will form the training set)



# Selecting Training and Test Data

## Data Selection Techniques

### ■ Bootstrap – Overall Accuracy of the Model

- Repeat the sampling procedure  $k$  times
- In each iteration, use the current test set to obtain an accuracy estimate of the model obtained from the current bootstrap sample
- The overall accuracy of the model,  $M$ , is then estimated as

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{testset} + 0.368 \times Acc(M_i)_{complete\ set})$$

❖ where

- ❑  $Acc(M_i)_{test\_set}$  is the accuracy of the model obtained with bootstrap sample  $i$  when it is applied to test set  $i$
- ❑  $Acc(M_i)_{train\_set}$  is the accuracy of the model obtained with bootstrap sample  $i$  when it is applied to the original full dataset
- ❑ Bootstrapping tends to be overly optimistic. It works best with **small data sets**

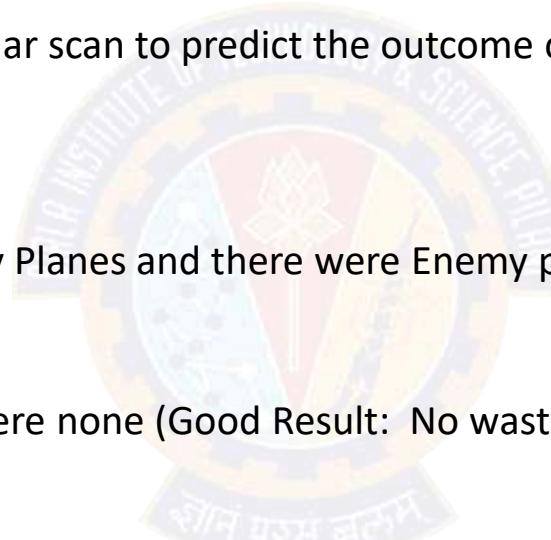
# ROC Curves



# ROC Curves

## History

- Originated from signal detection theory
- Applied in WW2 Battle of Britain to address:
  - Accurately identifying the signals on the radar scan to predict the outcome of interest – Enemy planes – when there were many extraneous signals (e.g. Geese)?
- True Positives
  - Radar Operator interpreted signal as Enemy Planes and there were Enemy planes (Good Result: No wasted Resources)
- True Negatives
  - Radar Operator said no planes and there were none (Good Result: No wasted resources)
- False Positives
  - Radar Operator said planes, but there were none (Geese: wasted resources)
- False Negatives
  - Radar Operator said no plane, but there were planes (Bombs dropped: very bad outcome)

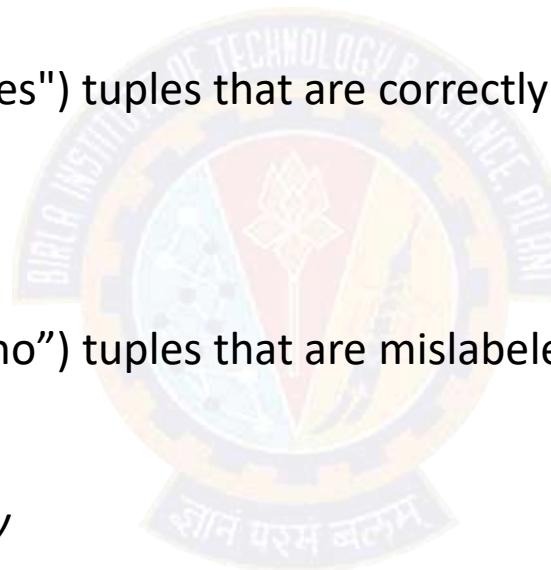


Truth	Predictions	
	POSITIVE CLASS	NEGATIVE CLASS
POSITIVE CLASS	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

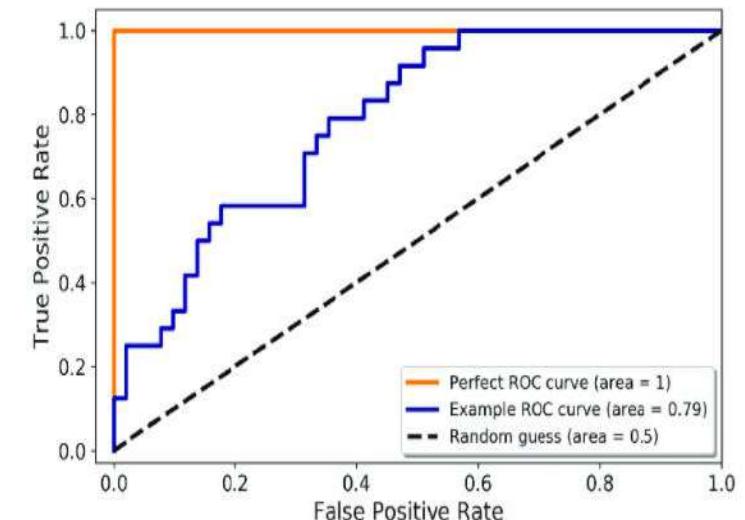
# ROC Curves

## Overview

- ROC curves are a useful visual tool for comparing two binary classification models
- Given TP, TN, FP, & FN
  - TPR is the proportion of positive (or "yes") tuples that are correctly labeled by the model  
 $= \frac{TP}{TP+FN}$ , which is *sensitivity*
  - FPR is the proportion of negative (or "no") tuples that are mislabeled as positive  
 $= \frac{FP}{FP+TN}$ , which is  $1 - \text{specificity}$
- An ROC curve for a given model shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR)



		Predictions	
		POSITIVE CLASS	NEGATIVE CLASS
Truth	POSITIVE CLASS	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
	NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)



# ROC Curves

## Optimal Probability Threshold

- Consider an email classification model that detects suspicious communication between terrorists over regular email
- Here, terrorist email is Class YES and a non-terrorist email is Class NO
- It is absolutely necessary for the model to identify the terrorists' emails correctly (True Positive Rate)
- That is, we choose Sensitivity as a metric to improve this model because
  - the occurrence of false negatives is unacceptable
- In this pursuit, we might end up having a few false positives/false alarms but that might be a compromise that we'll have to make.
- Let us say we develop 2 good models which have the same Sensitivity score
- Does that mean the two models have equal predictive power? NO

# ROC Curves

## Optimal Probability Threshold

- Most classification algorithms predict the probability that an observation belongs to class YES
- We need to decide a threshold for these probabilities, to classify the observations into one of the two classes
  - The observation having probability higher than the threshold are classified as class YES.
- For example:
  - the model predicts the probability of an email being a terrorist email as 0.75
  - If we set the threshold as 0.8, then we will classify this email as non-terrorist email
  - If we set the threshold as 0.7, then we will classify the email as a terrorist email
  - So, the performance of our system would vary as we change this threshold
  - This threshold can be adjusted to tune the behavior of the model for a specific problem
  - That is, we can reduce more of one or another type of error (FP/FN).

# ROC Curves

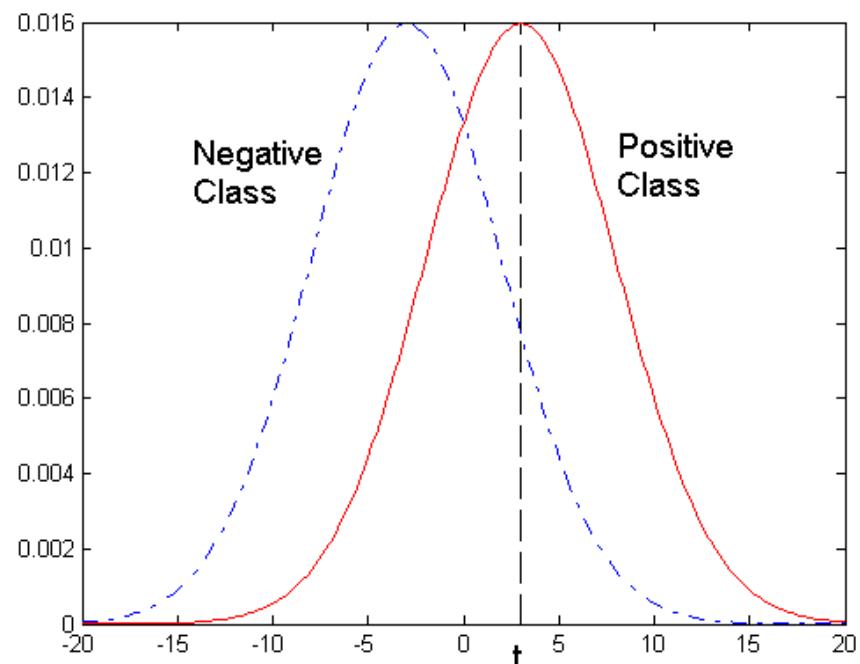
## Optimal Probability Threshold

- Two models with the **same sensitivity (TPR)** are **not equivalent**
- Among these two, the model with a **lower FPR** is obviously a better, more reliable model
- We do not want to waste any of our investigative resources on non-terrorist emails that were misclassified as terrorist emails
- The threshold that we set, can help us increase or decrease the TPR
- If we choose a low threshold, more emails will be classified as terrorist emails, we will be able to catch more true positives but then even the false positive rate would increase
- The choice of threshold entails a trade-off between false positives and false negatives.
- An ROC curve is a useful resource in this regard.

# ROC Curves

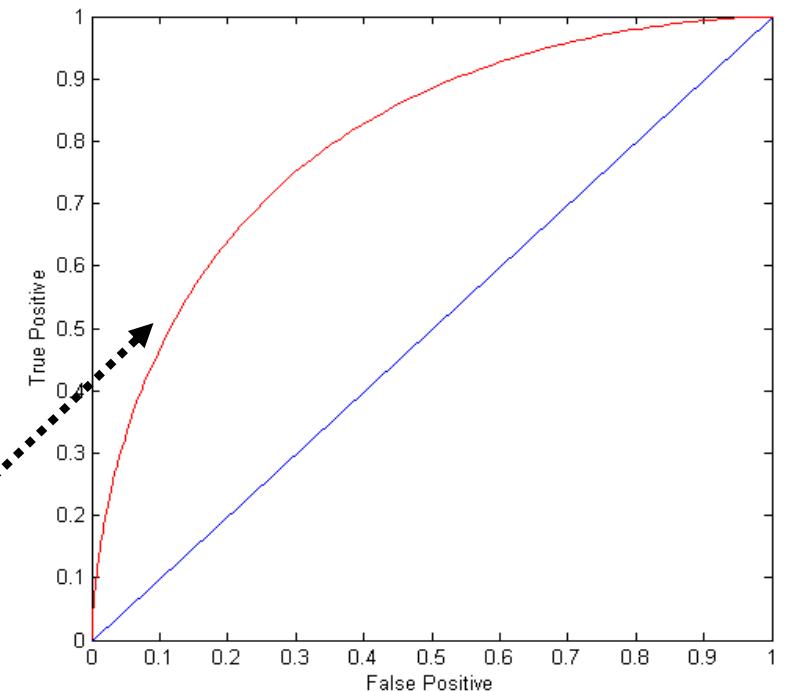
## Optimal Probability Threshold

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



At threshold  $t$ :

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



- |   |   |               |
|---|---|---------------|
| 4 | Linear Prediction Models: Linear Regression, Gradient Descent and Variants, Regularization, Bias Vs. Variance | T1: Chapter 4 |
|---|---|---------------|



# Regression

- Wish to learn a function  $f : X \rightarrow Y$ , where predicted output  $Y$  is real, given the  $n$  training instances  $\{<x^1, y^1> \dots <x^n, y^n>\}$ .
- For applications, where output will be a real value eg: Predicting housing price, predicting price of a stock market.
- Examples include
  - predict weight from gender, height, age, ...
  - Predict house price from locality, area, income, ...
  - predict Google stock price today from Google, Yahoo, MSFT prices yesterday
  - predict each pixel intensity in robot's current camera image, from previous image and previous action

## Correlation measures the linear relationship between objects

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard\_deviation}(\mathbf{x}) * \text{standard\_deviation}(\mathbf{y})} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}, \quad (2.11)$$

where we are using the following standard statistical notation and definitions

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y}) \quad (2.12)$$

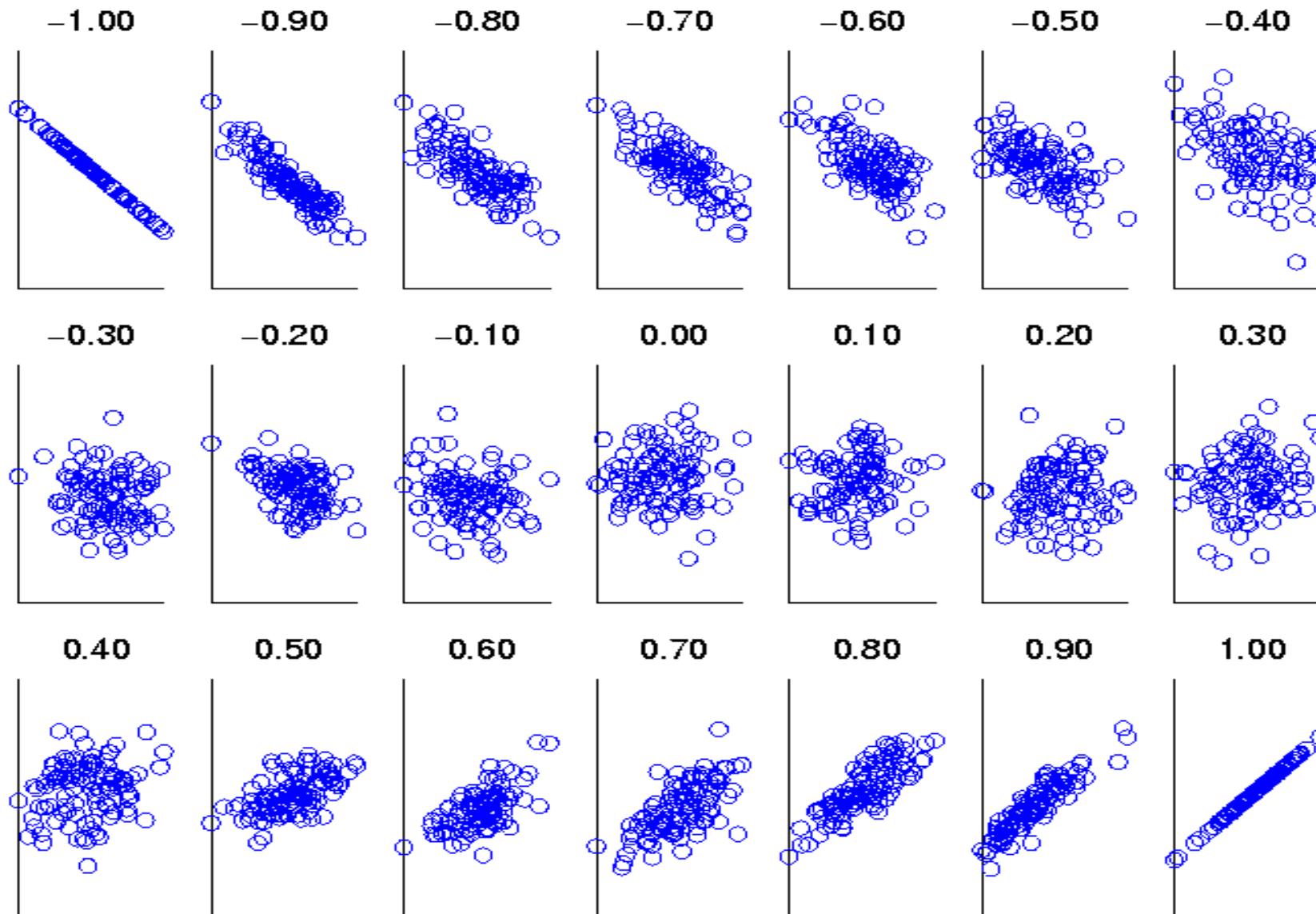
$$\text{standard\_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}$$

$$\text{standard\_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (y_k - \bar{y})^2}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k \text{ is the mean of } \mathbf{x}$$

$$\bar{y} = \frac{1}{n} \sum_{k=1}^n y_k \text{ is the mean of } \mathbf{y}$$

# Visually Evaluating Correlation



Scatter plots showing the similarity from  $-1$  to  $1$ .

# Two Approaches

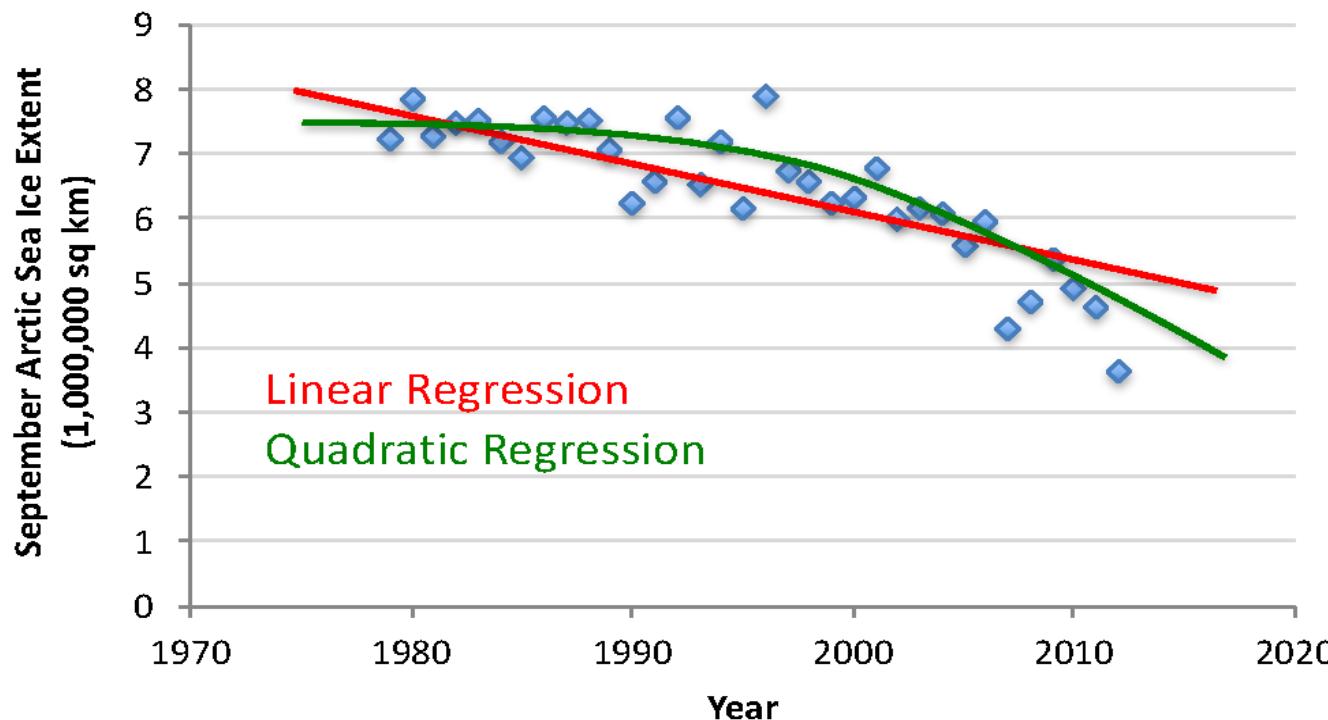
Two very different ways to train it:

- Using a direct “closed-form” equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set).
- Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method.

# Least Squares Approach

Given:

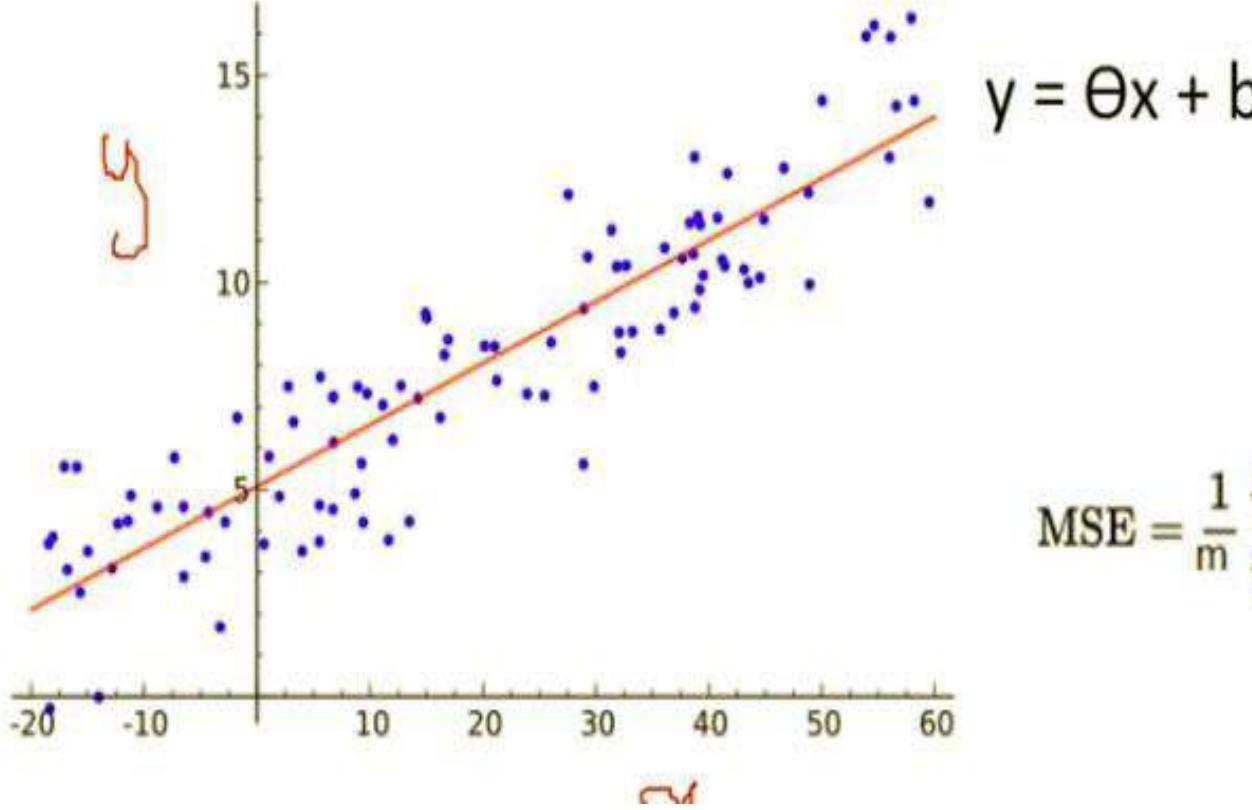
- Data  $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$  where  $\mathbf{x}^{(i)} \in \mathbb{R}^d$
- Corresponding labels  $\mathbf{y} = \{y^{(1)}, \dots, y^{(n)}\}$  where  $y^{(i)} \in \mathbb{R}$



N- no of samples  
D: dimension  
i- ith sample

# Least Squares Approach

## Linear Regression



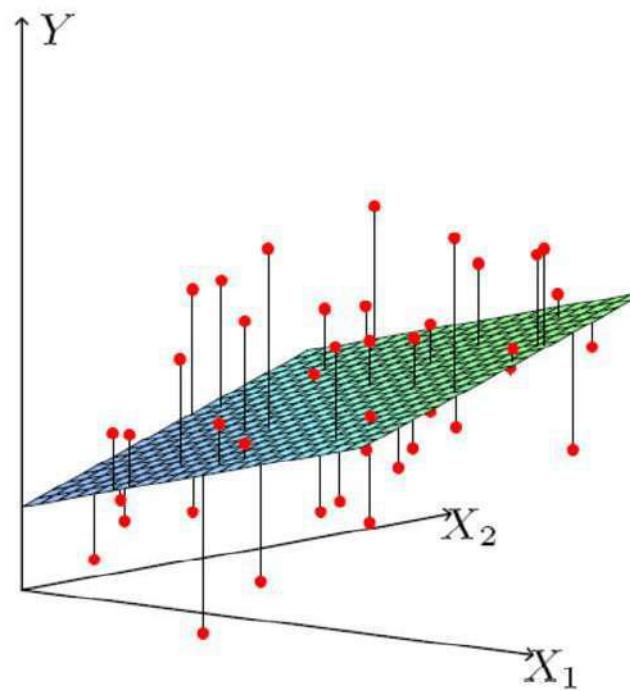
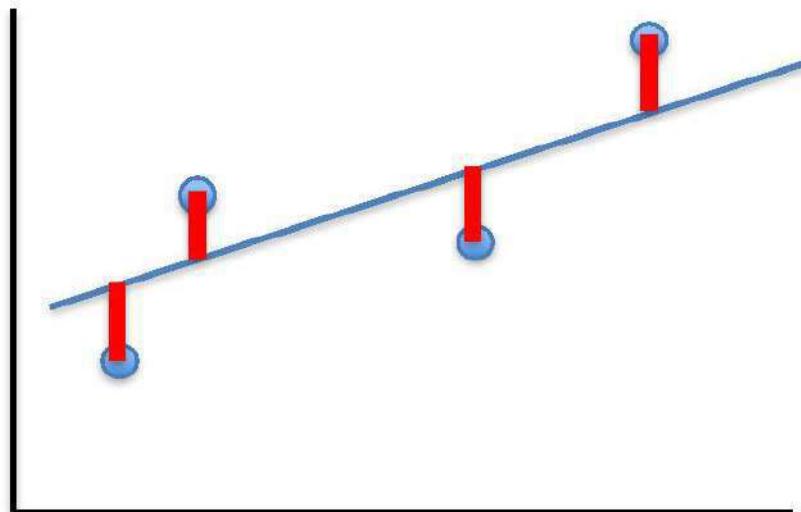
$$MSE = \frac{1}{m} \sum_{i=1}^m (\hat{Y}_i - Y_i)^2$$

# Least Squares Linear Regression

- Cost Function

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

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



# Least Squares Linear Regression

How to learn proper  $\theta$  using  $J(\theta)$ ?

- 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  $x_0 = 1$

Cost Function

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

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

How to minimize  $J()$

# Least Squares Based Solution

- 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  $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

# Least Squares Based Solution

- Consider our model for  $n$  instances:

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

- Let

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

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

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

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

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

# Least Squares Based Solution

- For the linear regression cost function:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \sum_{i=1}^n \left( \boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top}_{\mathbb{R}^{1 \times n}} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\mathbb{R}^{n \times 1}} \end{aligned}$$

Let:

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

# Least Squares Based Solution

- Solve for optimal  $\theta$  analytically
  - Notice that the solution is when  $\frac{\partial}{\partial \theta} J(\theta) = 0$
- Derivation:

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

1 x 1

Take derivative and set equal to 0, then solve for  $\theta$ :

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

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

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

Closed Form Solution:

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

# Intuition Behind Cost Function

## Training set

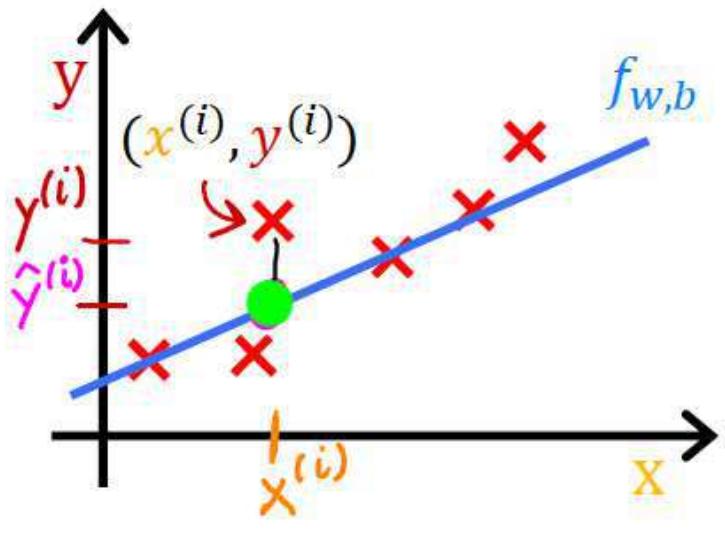
features	targets
size in feet <sup>2</sup> (x)	price \$1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

$$\text{Model: } f_{w,b}(x) = wx + b$$

$w, b$ : parameters  
coefficients  
weights

What do  $w, b$  do?

# Intuition Behind Cost Function



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = w x^{(i)} + b$$

Cost function: Squared error cost function

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

$m$  = number of training examples

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

intuition (next!)

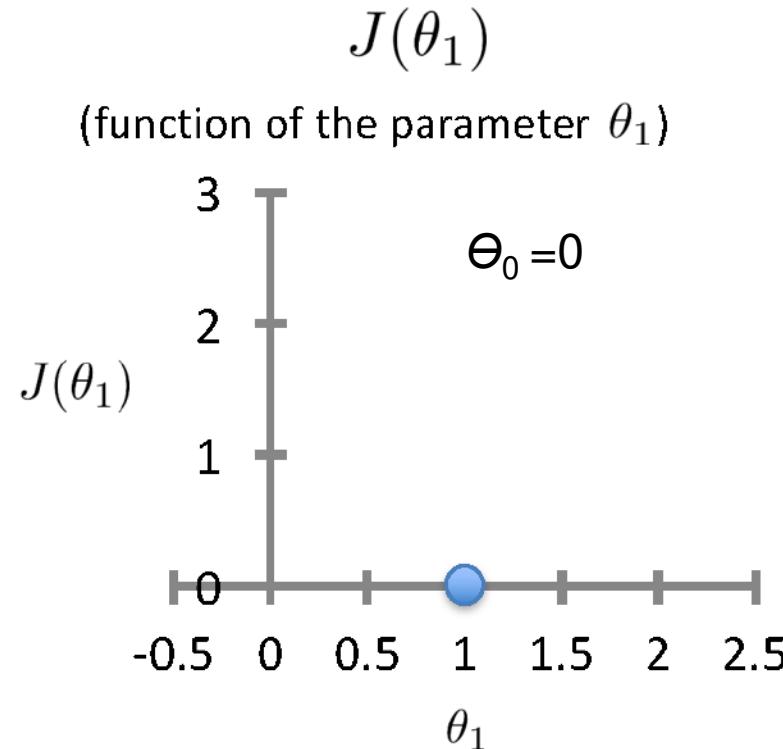
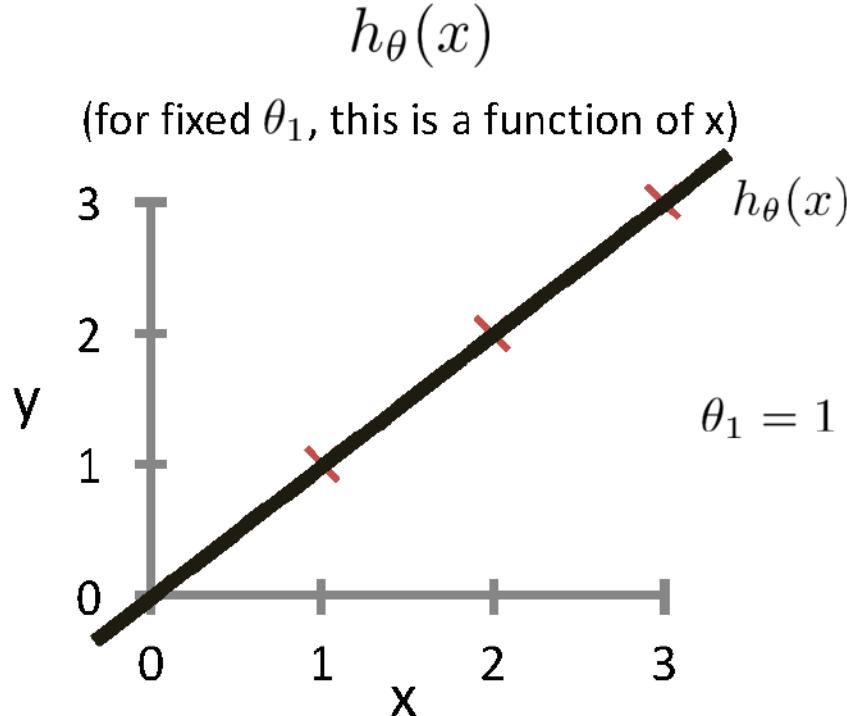
Find  $w, b$ :

$\hat{y}^{(i)}$  is close to  $y^{(i)}$  for all  $(x^{(i)}, y^{(i)})$ .

# Intuition Behind Cost Function

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

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



# Intuition Behind Cost Function

model:

$$\underline{f_{w,b}(x) = wx + b}$$

parameters:

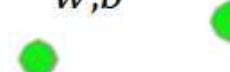
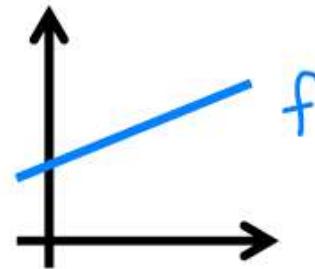
$$\underline{w, b}$$

cost function:

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

goal:

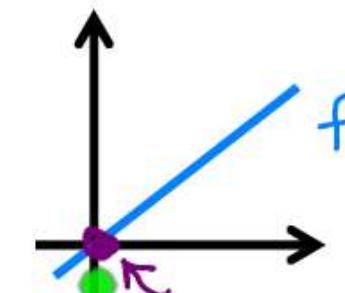
$$\underset{w, b}{\text{minimize}} J(w, b)$$



simplified

$$f_w(x) = \underline{wx} \quad b = \emptyset$$

$$w$$



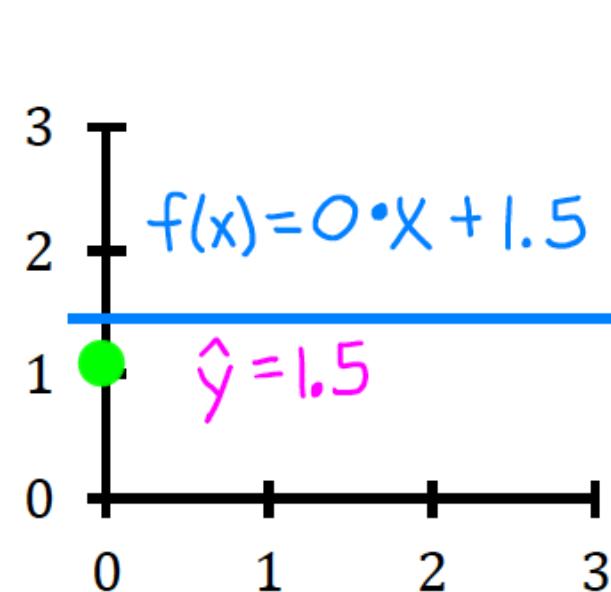
$$\underline{J(w)} = \frac{1}{2m} \sum_{i=1}^m (f_w(x^{(i)}) - y^{(i)})^2$$

$$\underset{w}{\text{minimize}} \underline{J(w)}$$

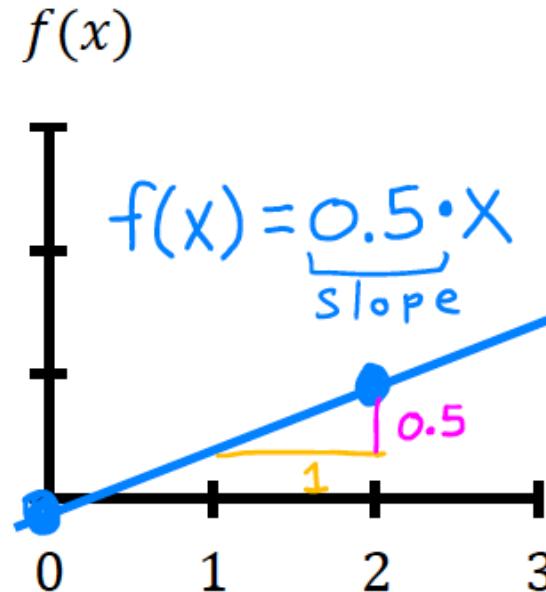
$$\nwarrow \underline{wx^{(i)}}$$

# Intuition Behind Cost Function

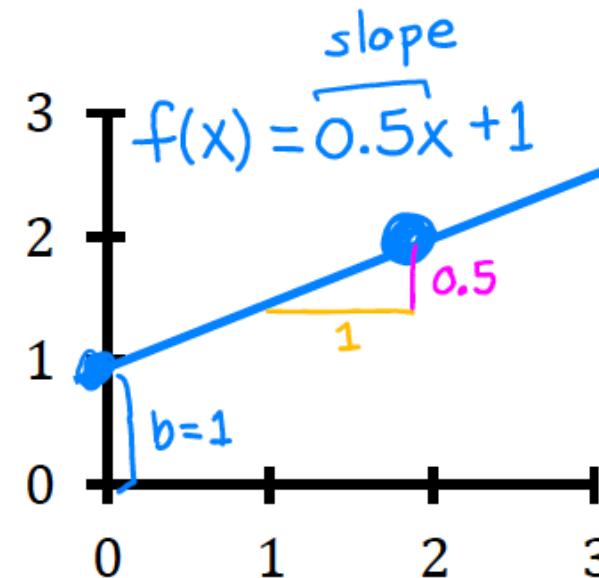
$$f_{w,b}(x) = wx + b$$



$$\begin{aligned}\rightarrow w &= 0 \\ \rightarrow b &= 1.5 \\ &\text{(y-intercept)}\end{aligned}$$

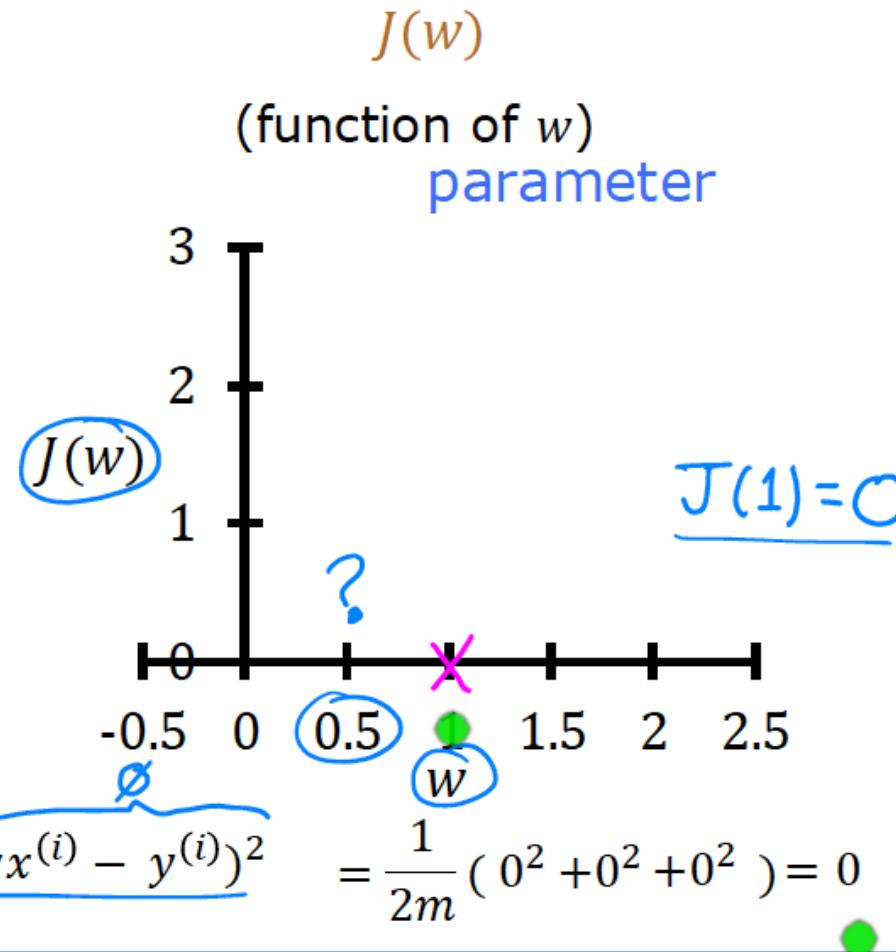
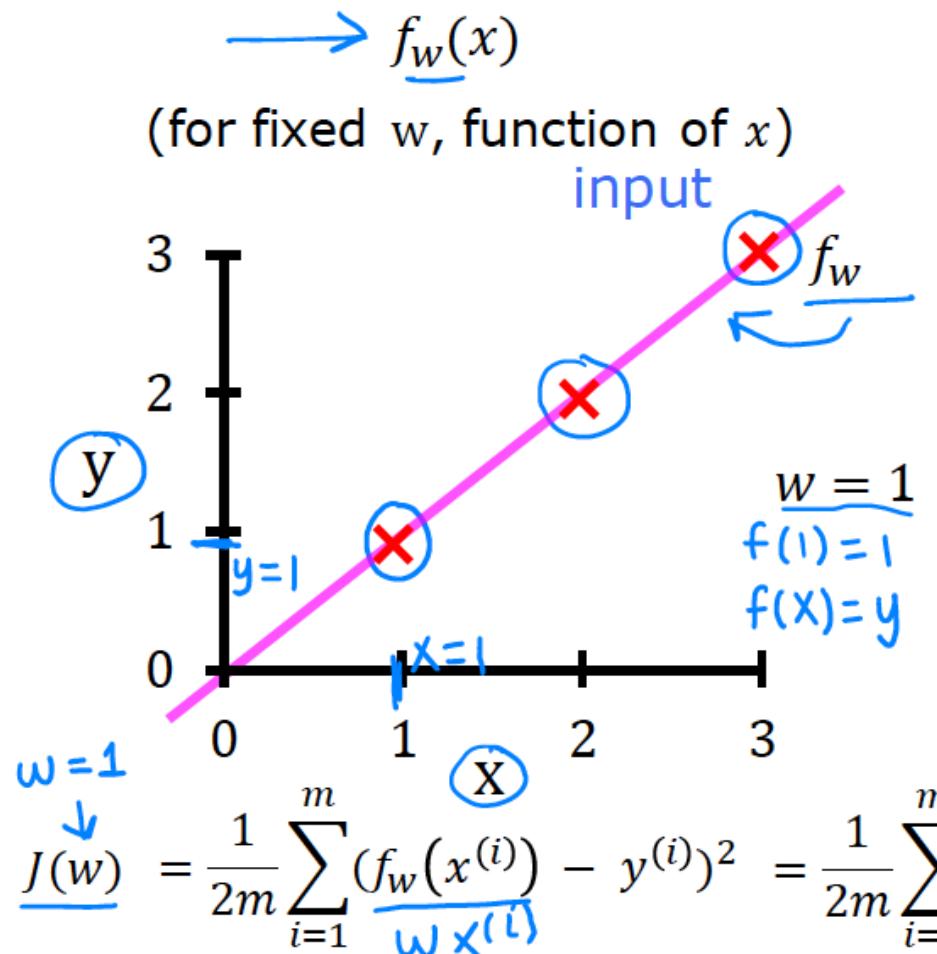


$$\begin{aligned}\rightarrow w &= 0.5 \\ \rightarrow b &= 0\end{aligned}$$

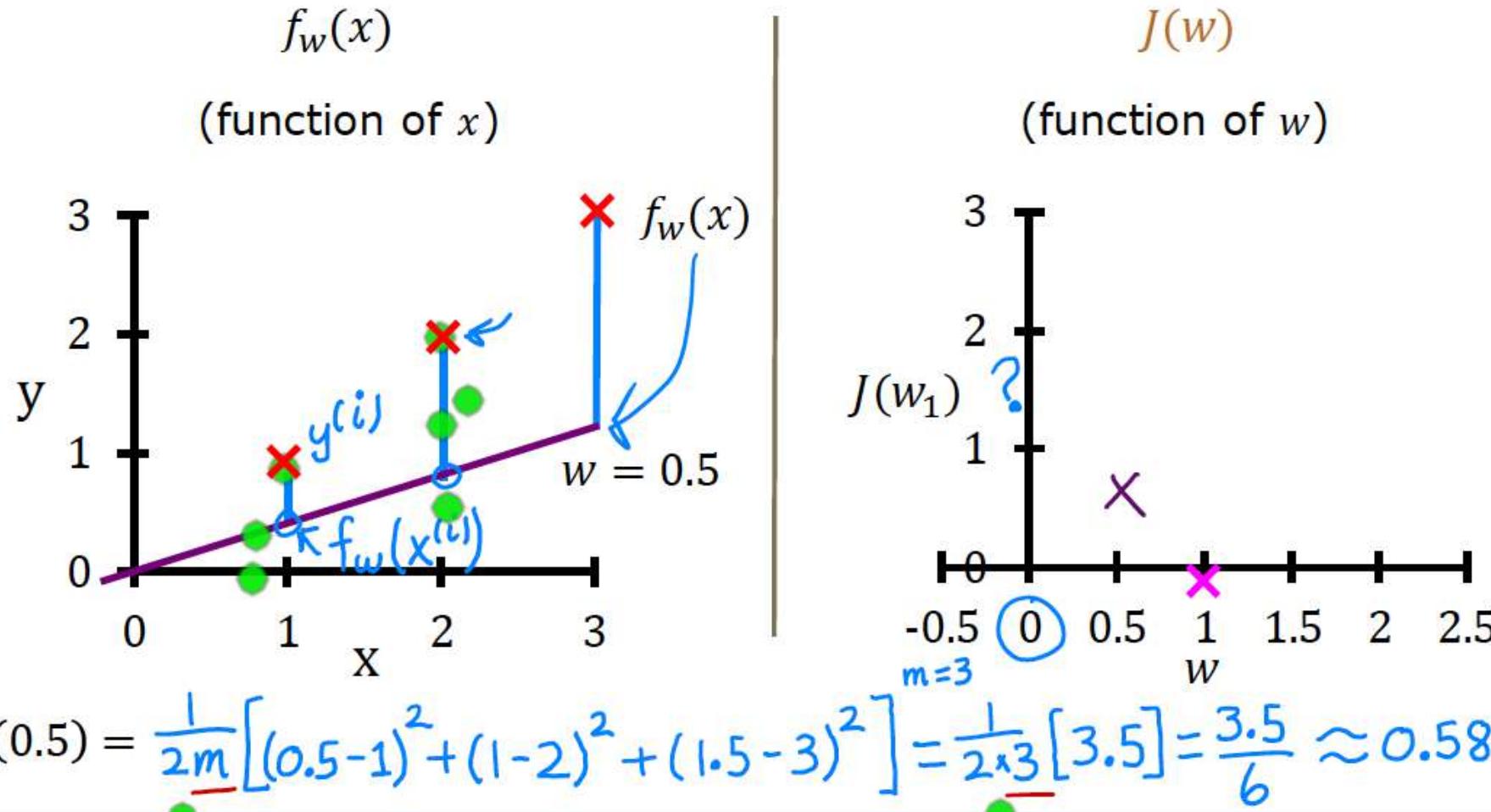


$$\begin{aligned}\rightarrow w &= 0.5 \\ \rightarrow b &= 1\end{aligned}$$

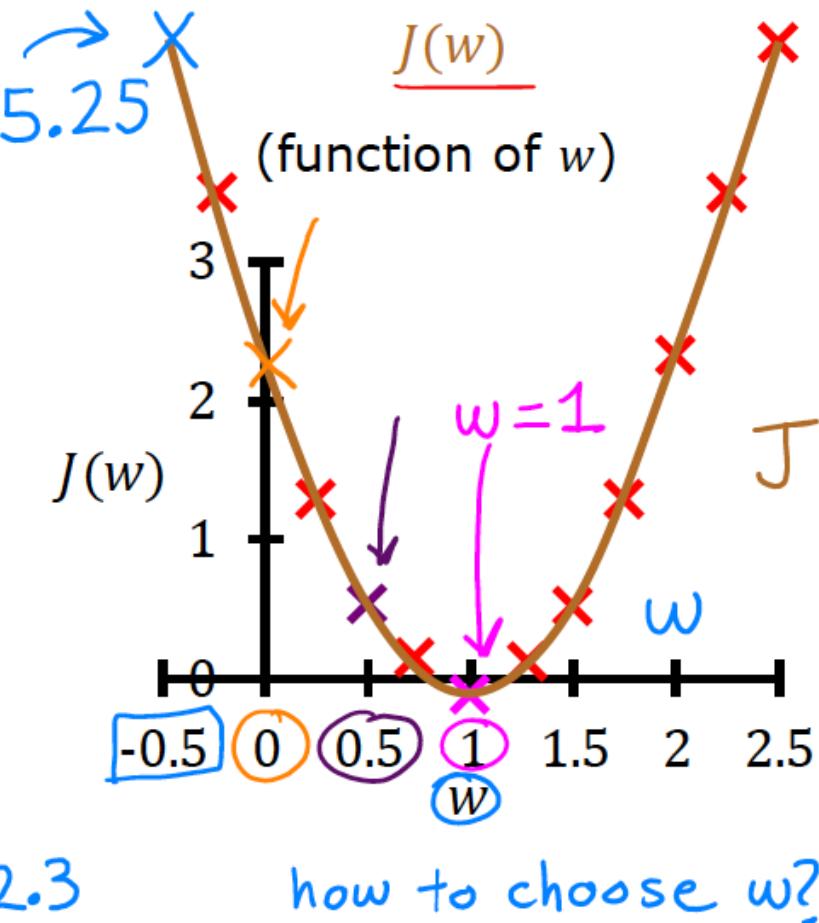
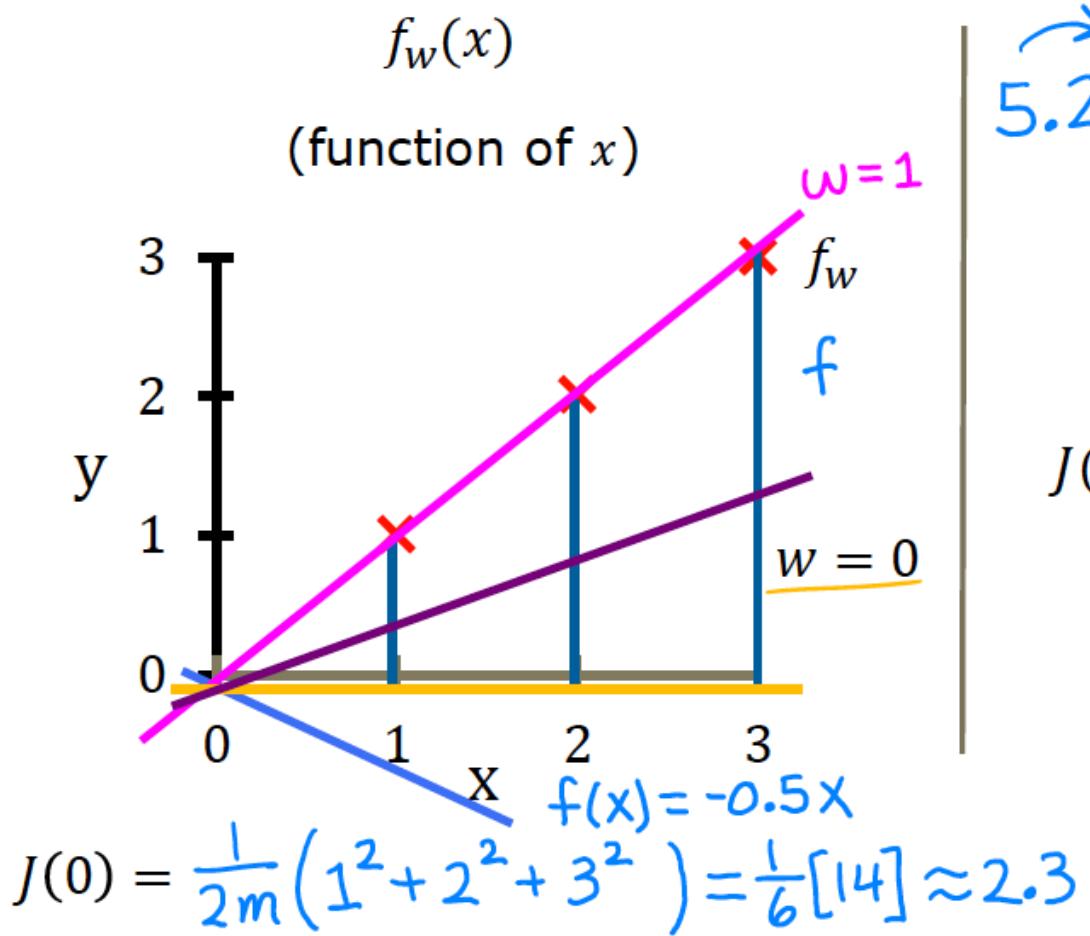
# Intuition Behind Cost Function



# Intuition Behind Cost Function



# Intuition Behind Cost Function



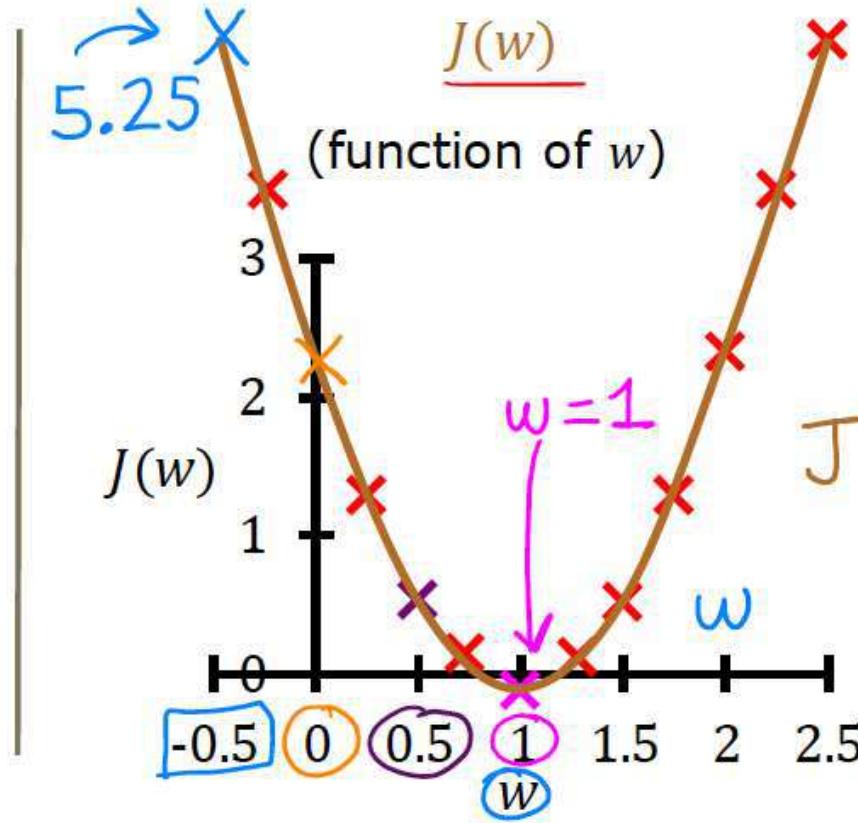
# Intuition Behind Cost Function

goal of linear regression:

$$\underset{w}{\text{minimize}} J(w)$$

general case:

$$\underset{w,b}{\text{minimize}} J(w, b)$$



choose  $w$  to minimize  $J(w)$

# Intuition Behind Cost Function

Model

$$f_{w,b}(x) = wx + b$$

Parameters

$w, b$

*before:  $b=0$*

Cost Function

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

Objective

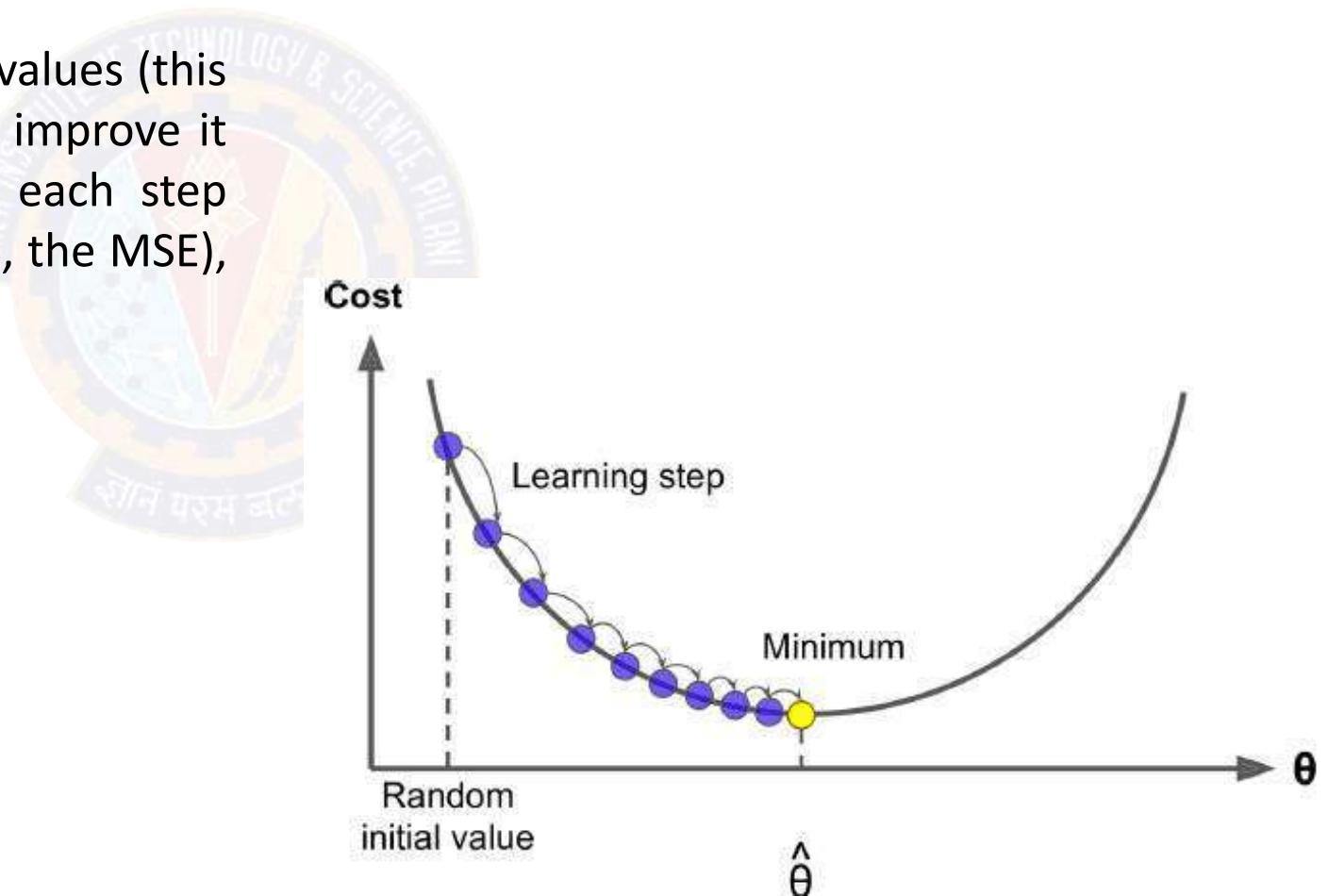
$$\underset{w,b}{\text{minimize}} J(w, b)$$

# Gradient Descent

Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.

The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function.

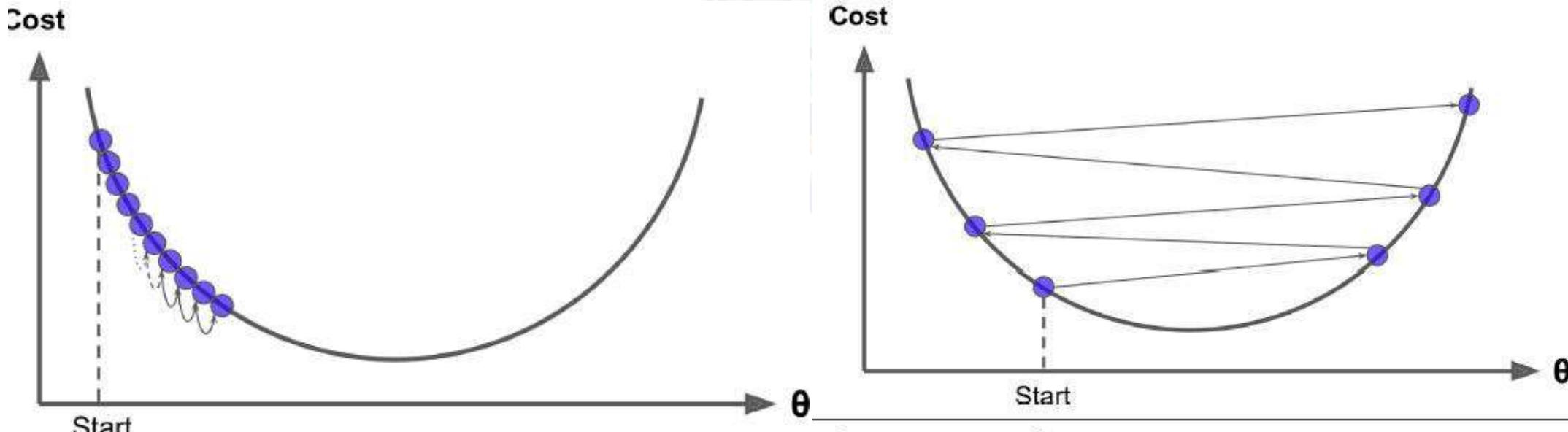
Concretely, you start by filling  $\theta$  with random values (this is called random initialization), and then you improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum



# Gradient Descent- Hyper parameter

If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time

On the other hand, if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before. This might make the algorithm diverge, with larger and larger values, failing to find a good solution

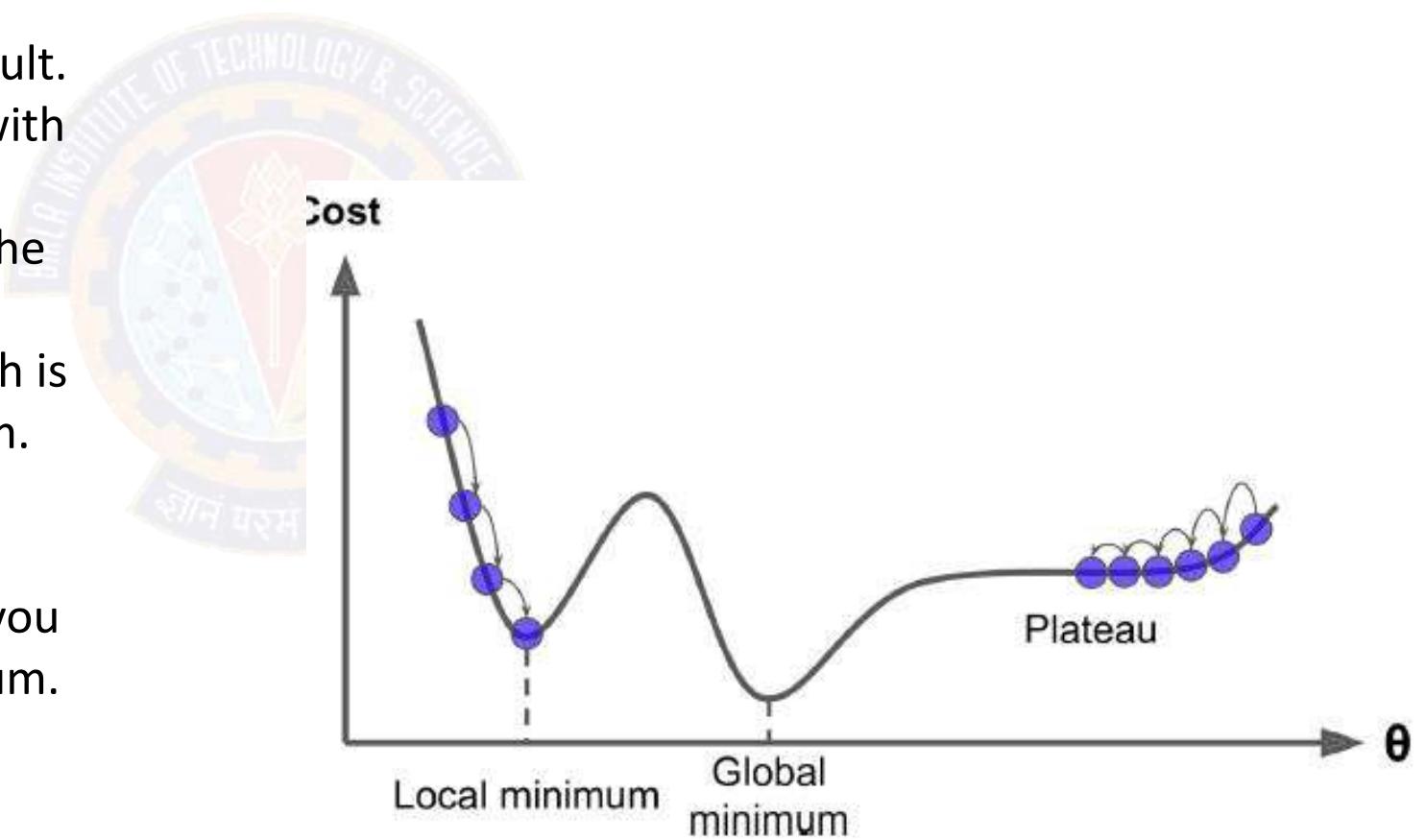


*Learning rate too small*

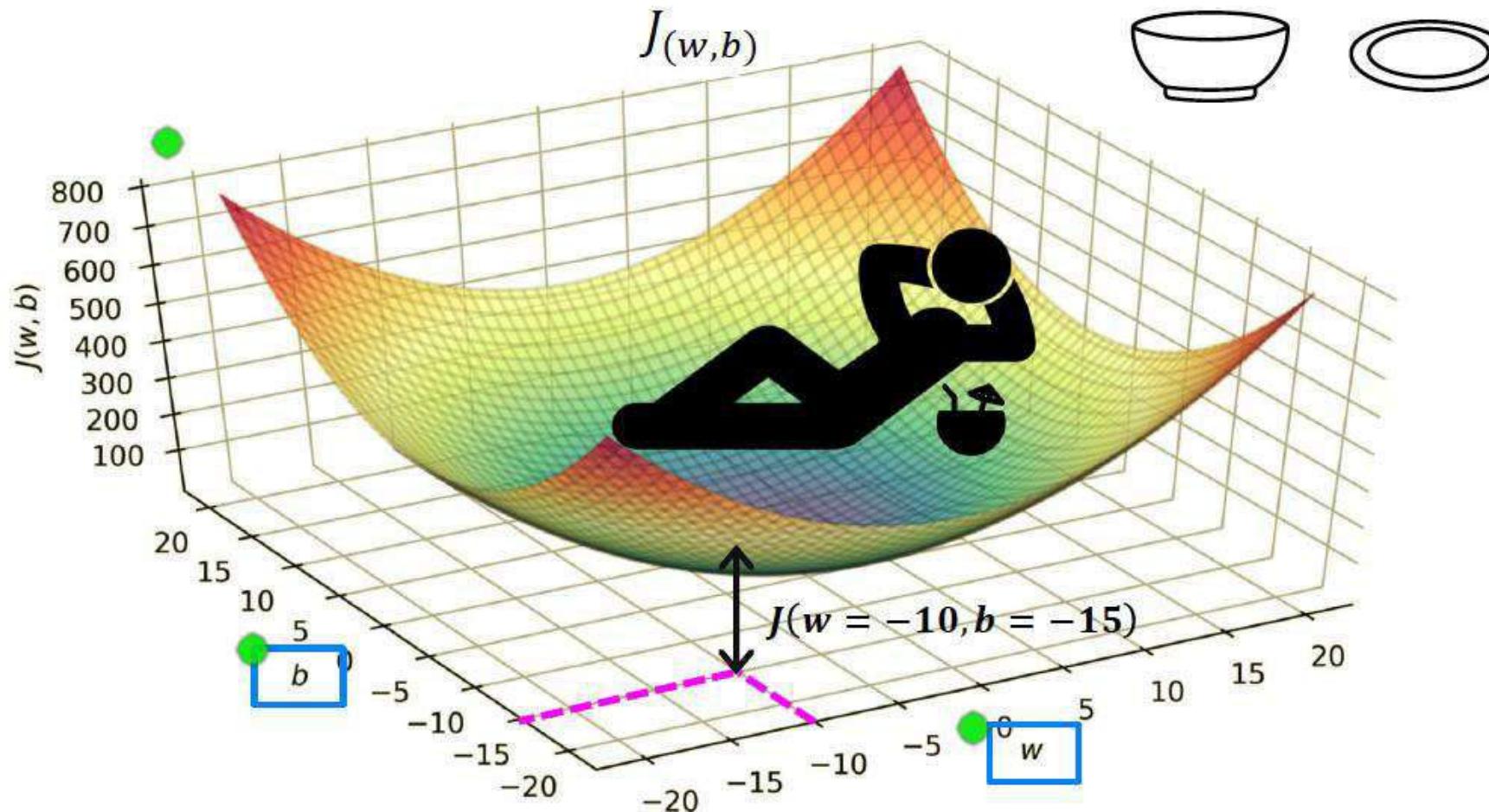
*5. Learning rate too large*

# Gradient Descent- Pitfalls

- Finally, not all cost functions look like nice regular bowls.
- There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult.
- **Figure** shows the two main challenges with Gradient Descent:
  1. if the random initialization starts the algorithm on the left, then it will converge to a local minimum, which is not as good as the global minimum.
  2. If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.



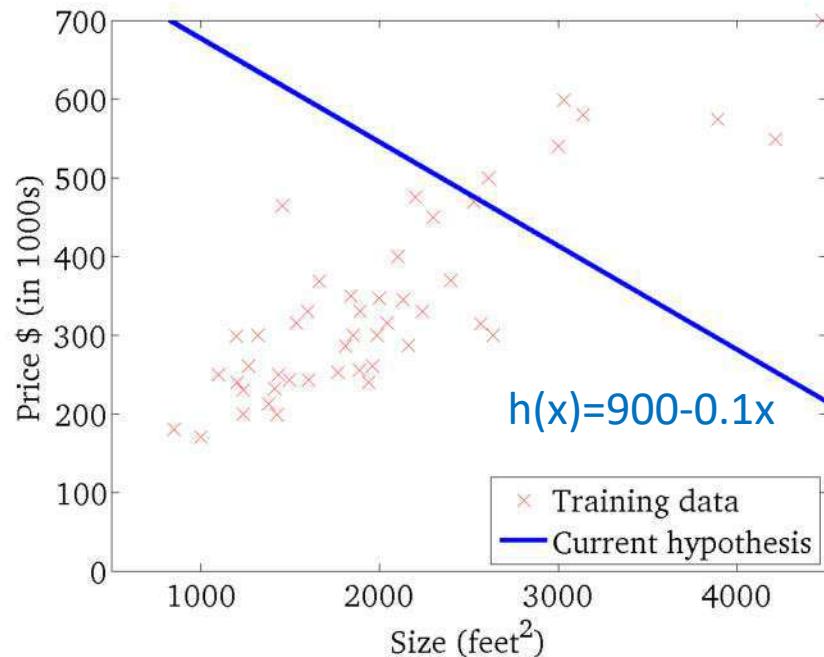
# Intuition Behind Cost Function



# Intuition Behind Cost Function

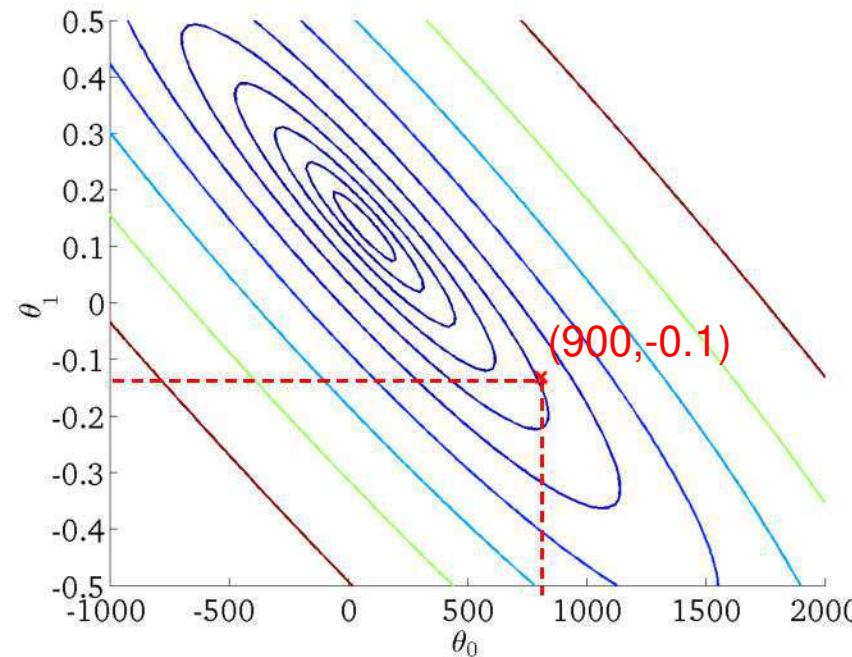
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

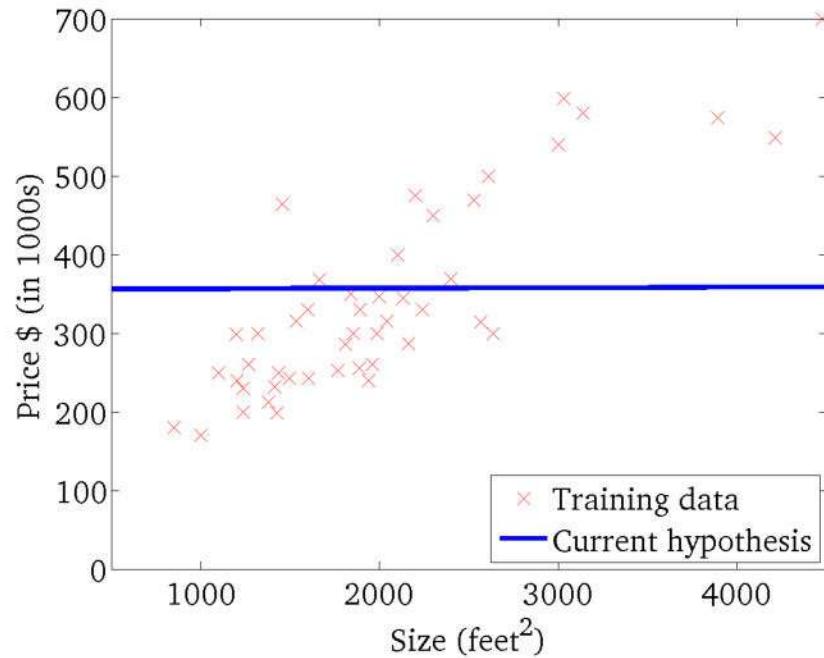
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind Cost Function

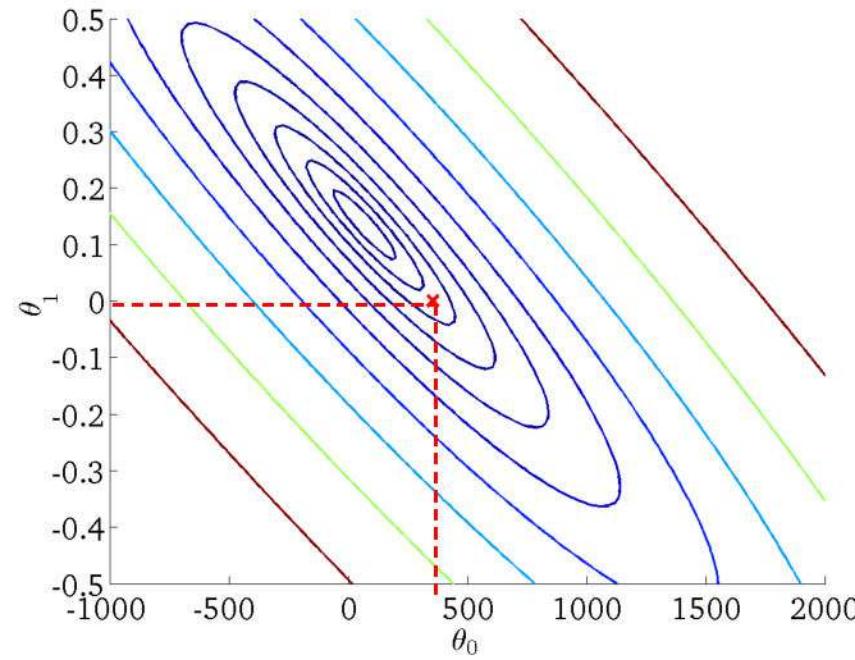
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

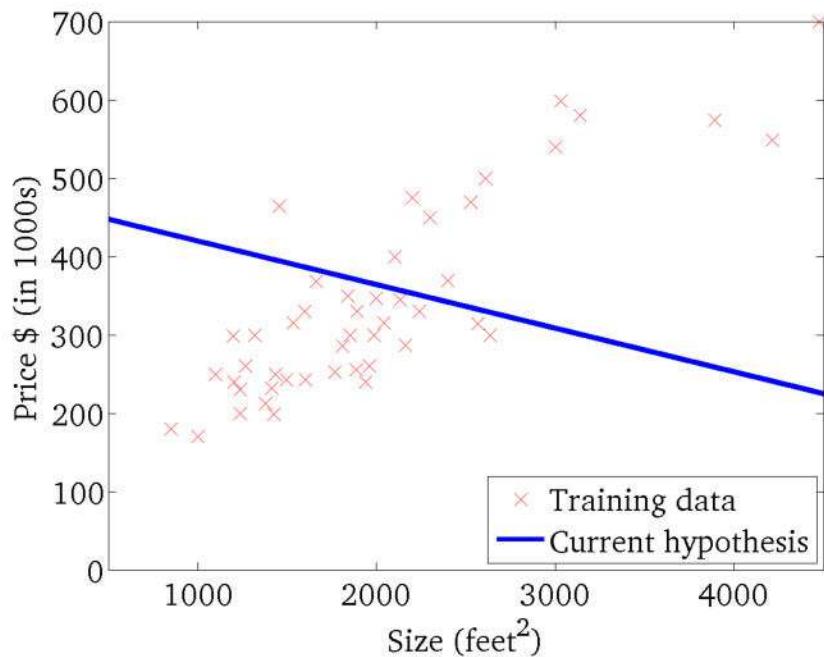
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind Cost Function

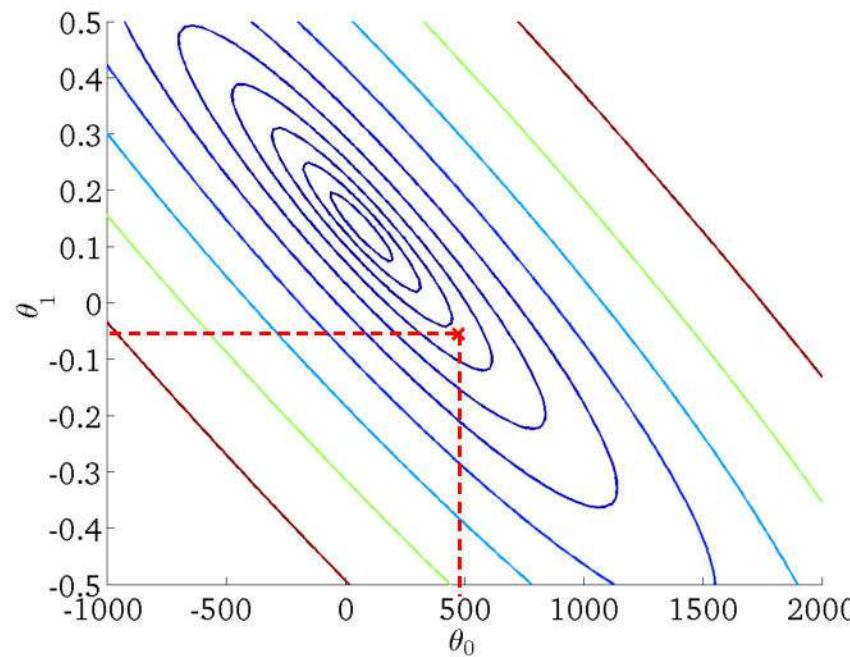
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

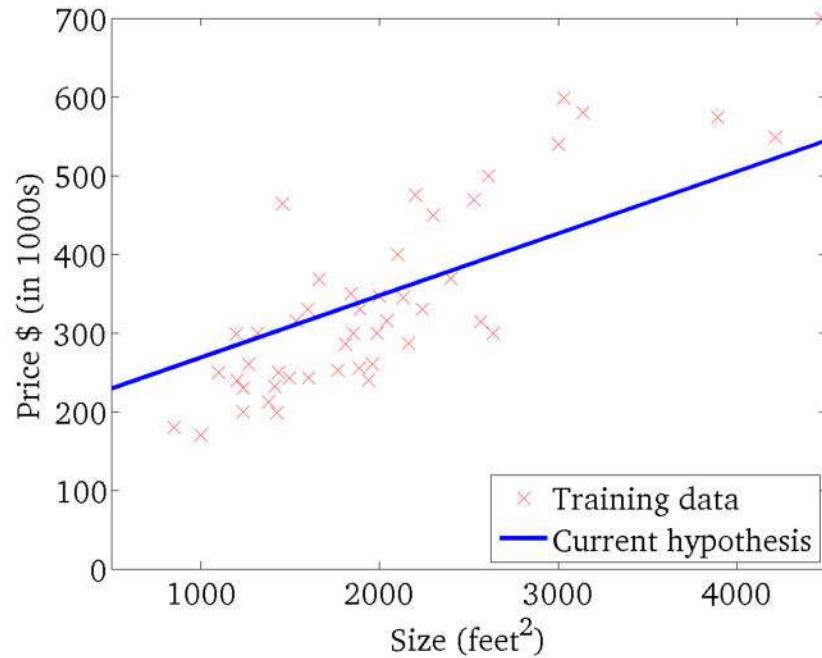
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind Cost Function

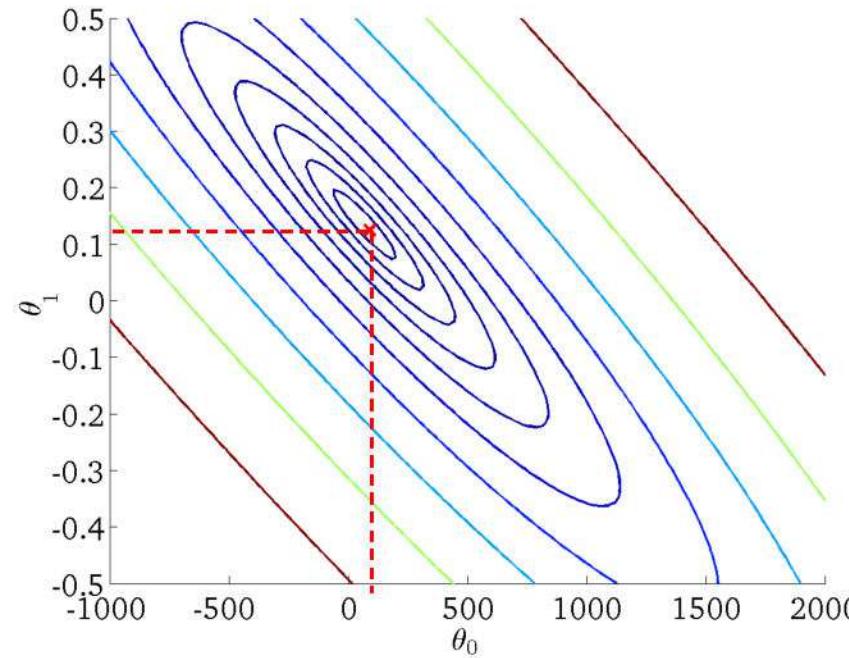
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



# Basic Search Procedure

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

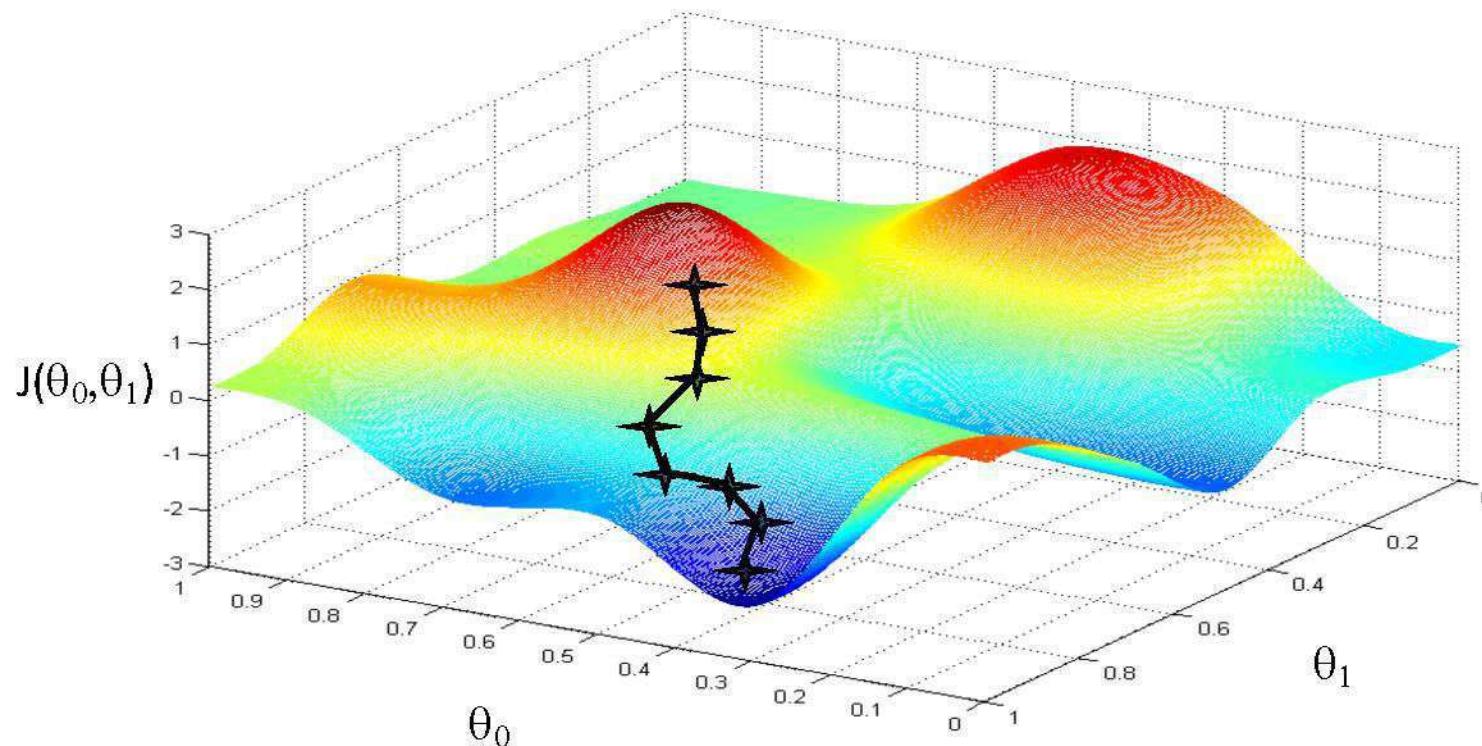


Figure by Andrew Ng

# Basic Search Procedure

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

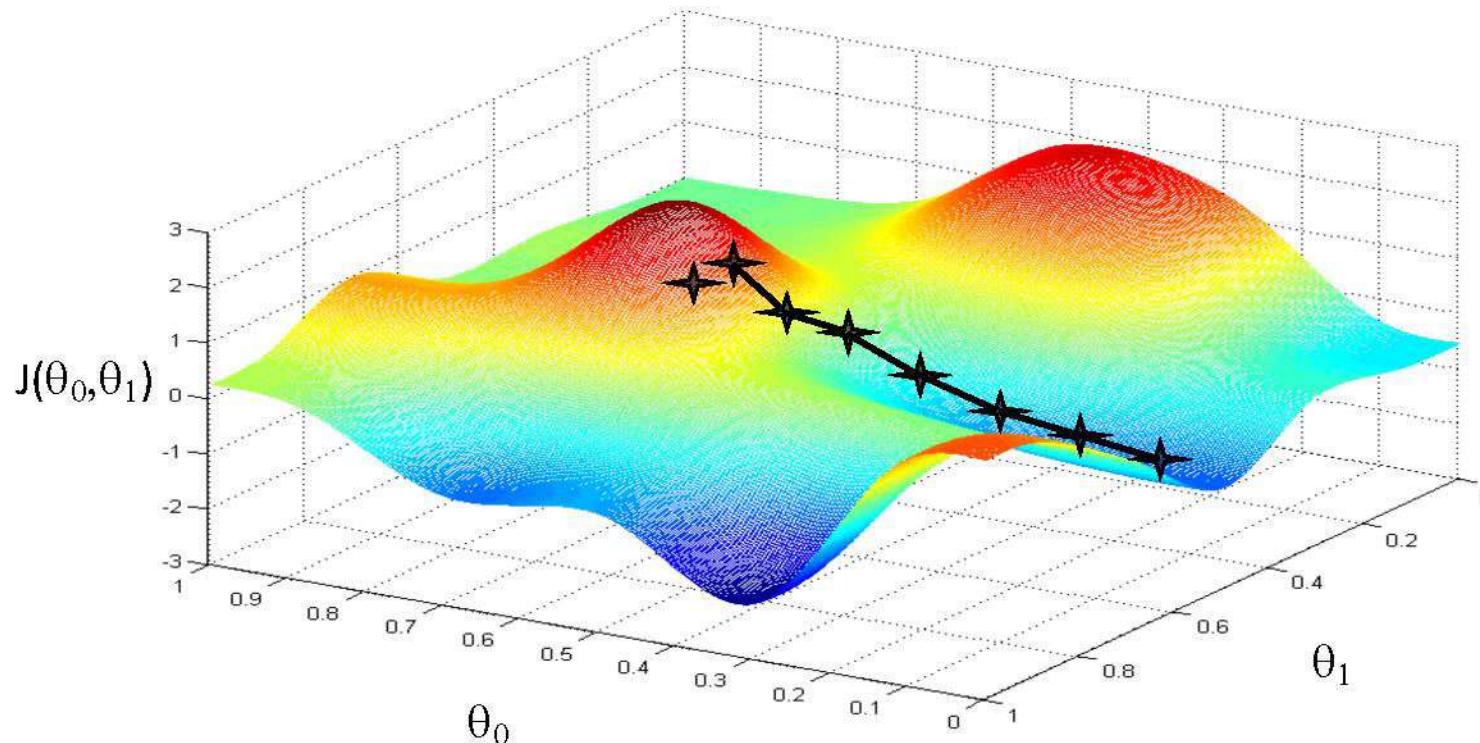
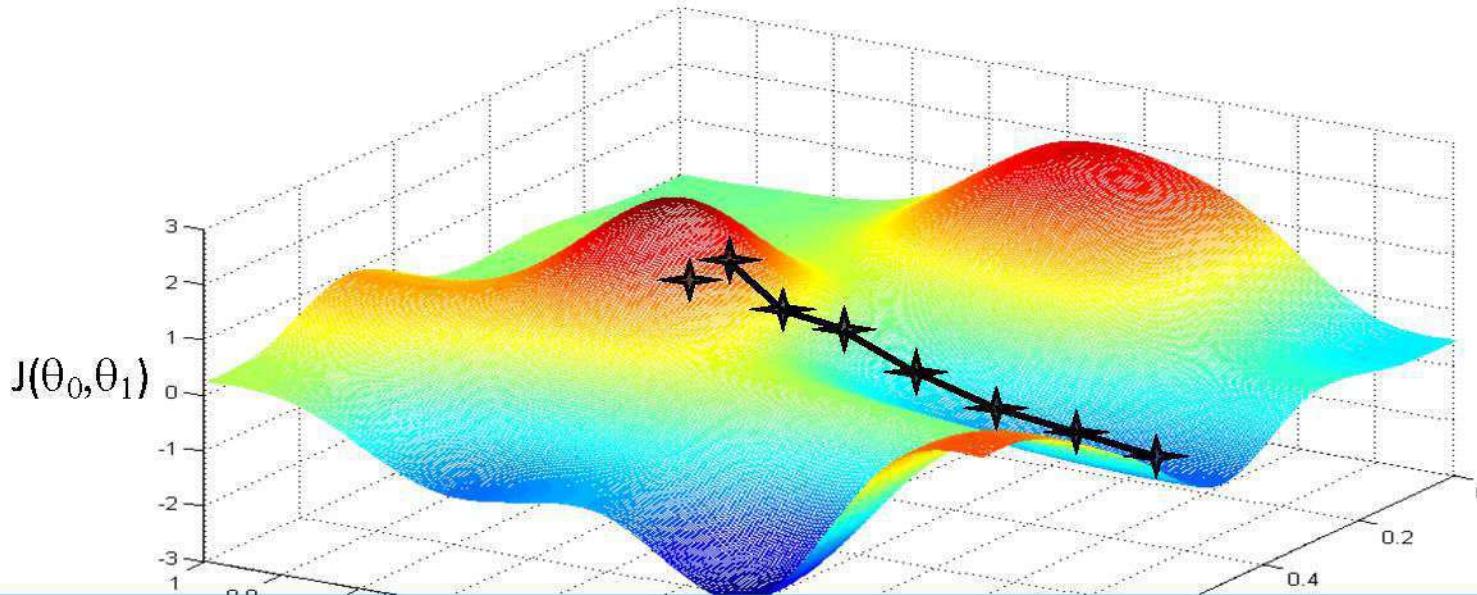


Figure by Andrew Ng

# Basic Search Procedure

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



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

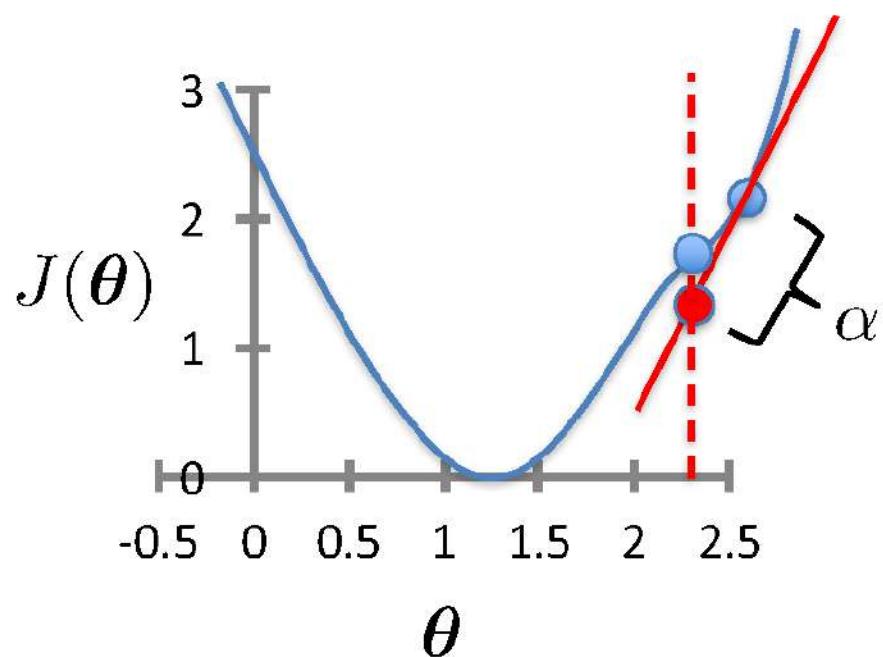
# Gradient Descent

- Initialize  $\theta$
- 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  $\theta$
- Repeat until convergence

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

For Linear Regression:

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

..

# Gradient Descent for Linear Regression

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous  
update  
for  $j = 0 \dots d$

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

$L_2$  norm:  $\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$

# Gradient Descent-Numerical Example

# Gradient Descent Numerical Example

Iteration	$\theta_0$	$\theta_1$
1	0.01	0.01
2	0.0397	0.0694
3	0.066527	0.176708
4	0.08056	0.218808
5	0.118814	0.410078
6	0.123526	0.414789
7	0.143994	0.455727
8	0.154325	0.497051
9	0.157871	0.507687
10	0.180908	0.622872
11	0.18287	0.624834
12	0.198544	0.656183
13	0.200312	0.663252
14	0.198411	0.65755
15	0.213549	0.733242
16	0.214081	0.733774
17	0.227265	0.760141
18	0.224587	0.749428
19	0.219858	0.735242
20	0.230897	0.790439

## Further Iterations



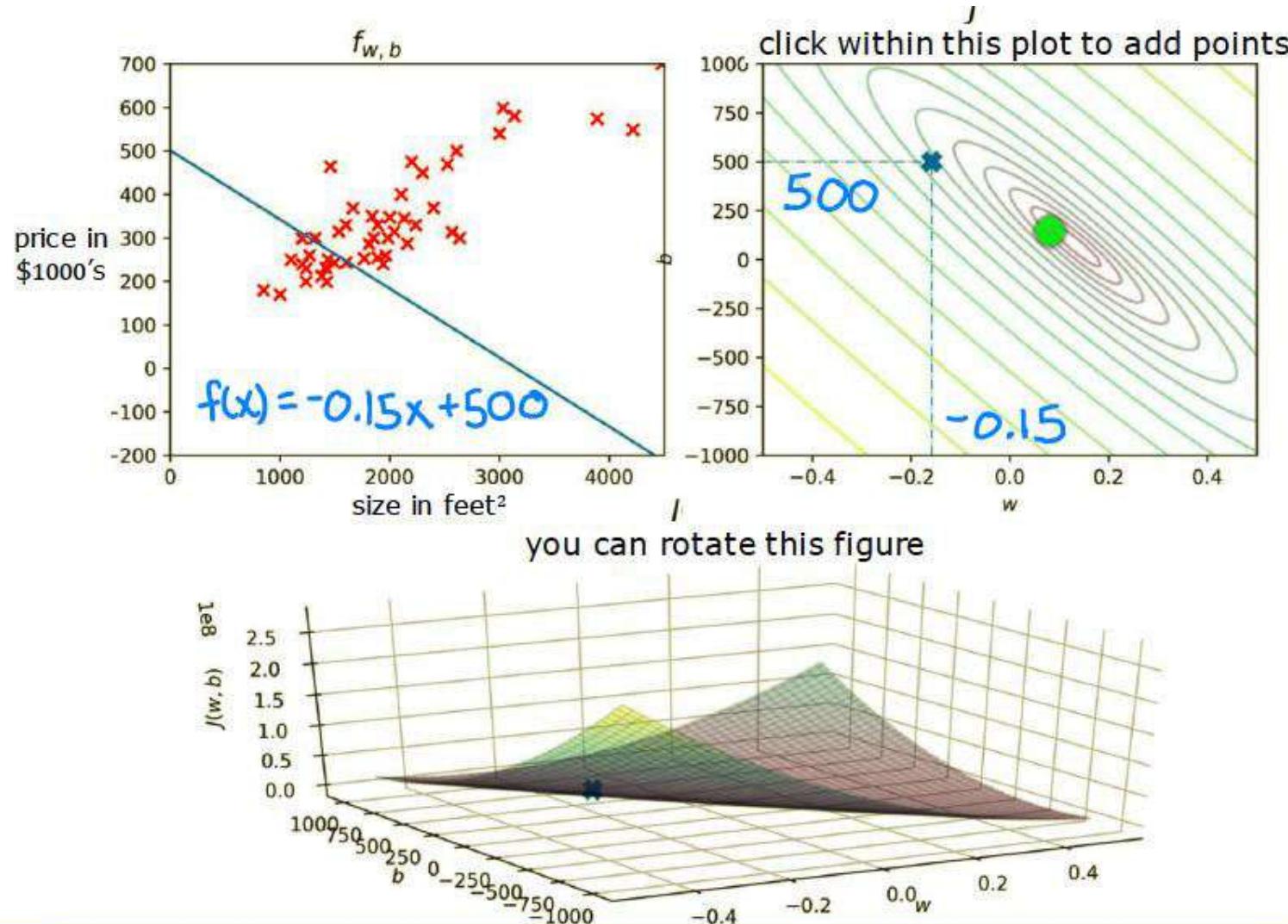
# Gradient Descent-Numerical Example



# Gradient Descent-Numerical Example



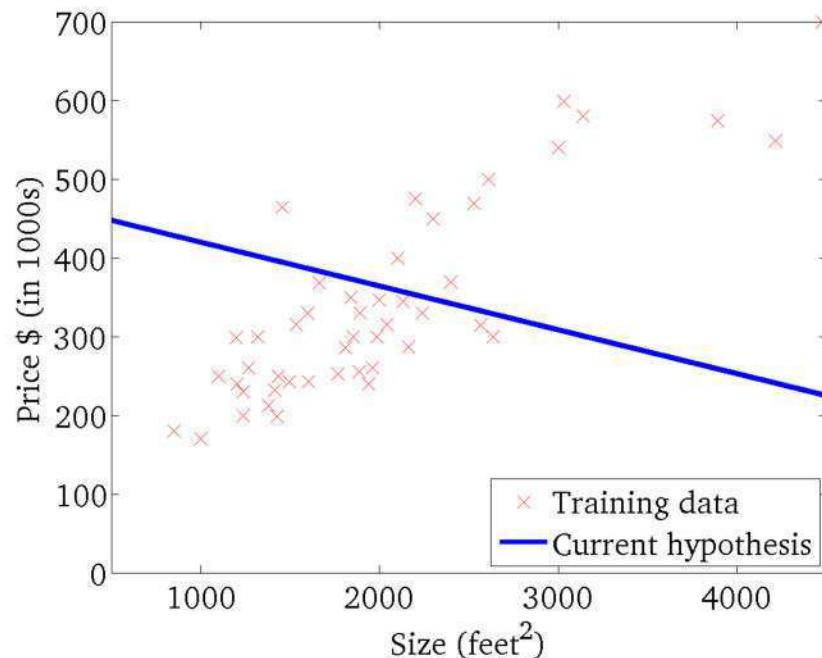
# Gradient Descent



# Gradient Descent

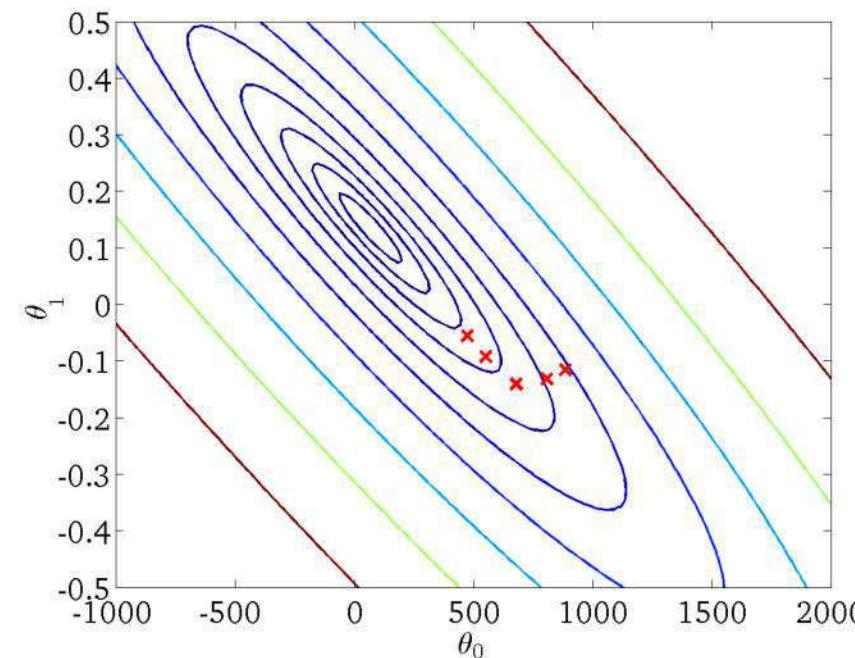
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )

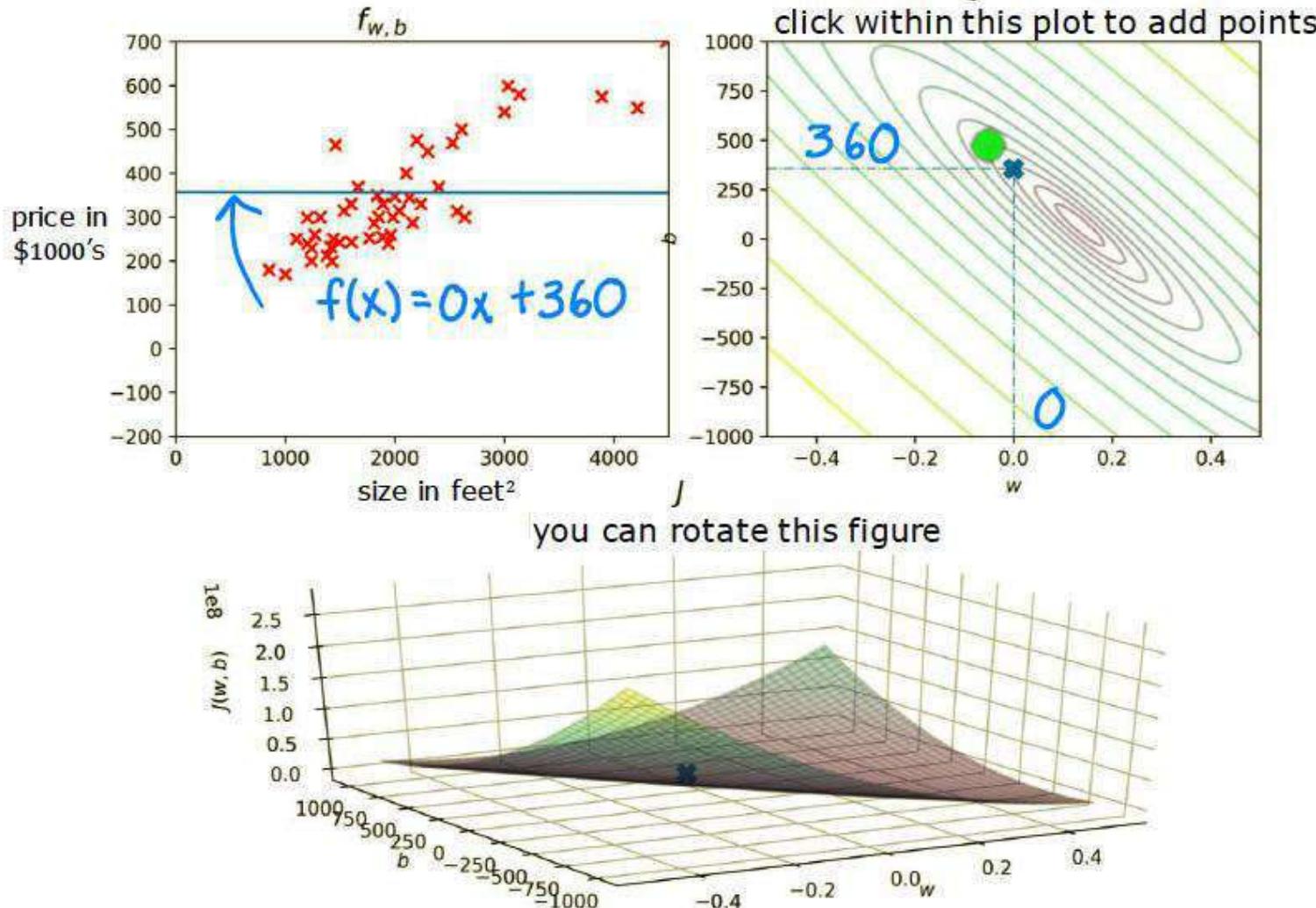


$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



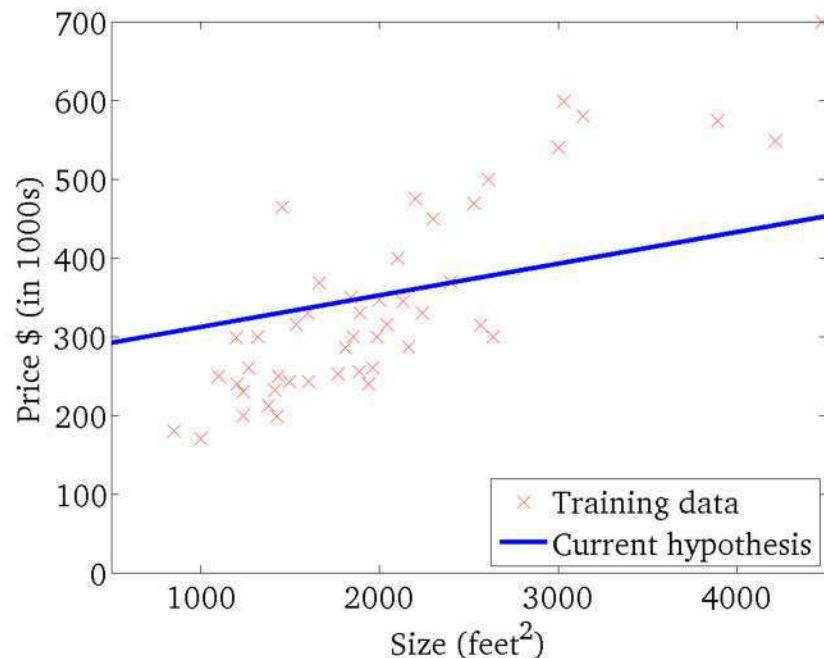
# Gradient Descent



# Gradient Descent

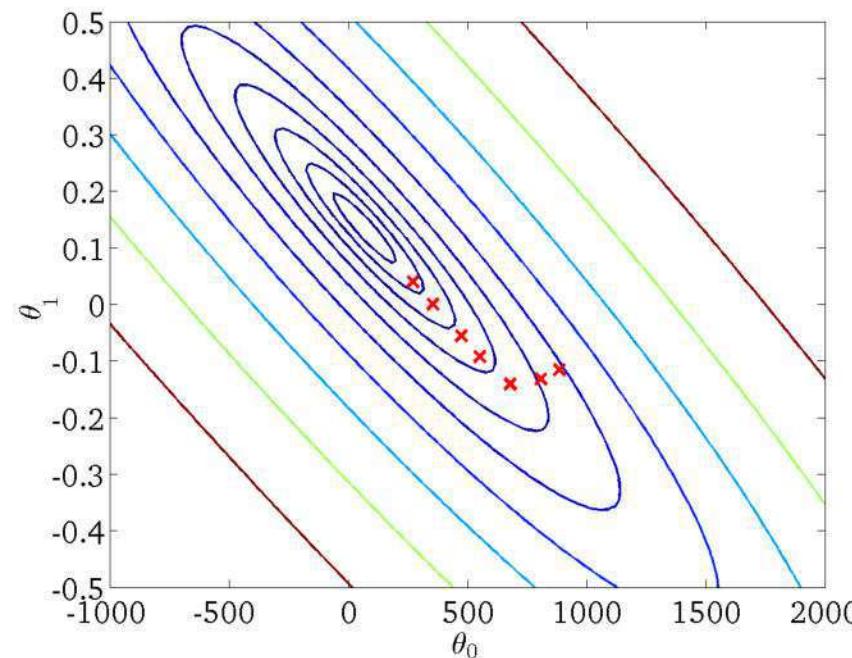
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

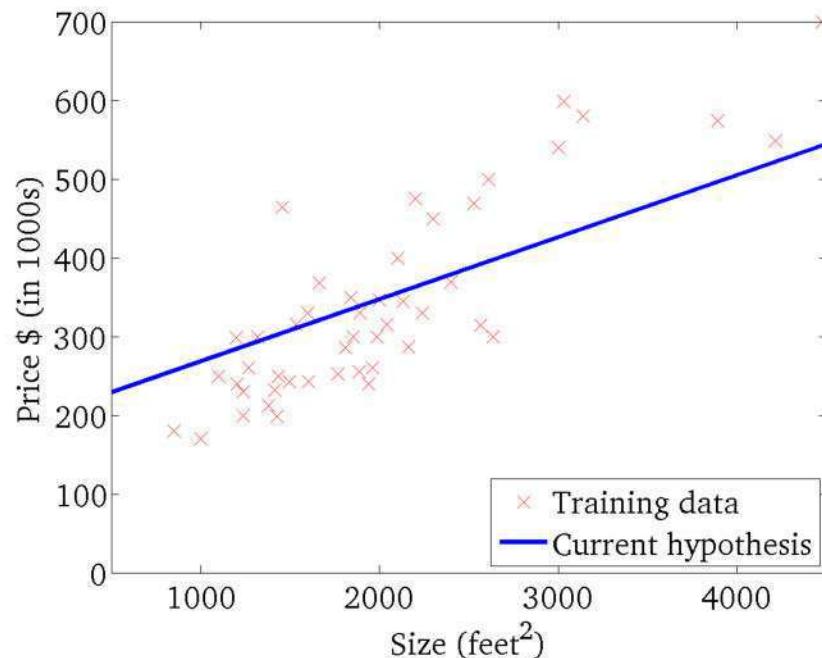
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

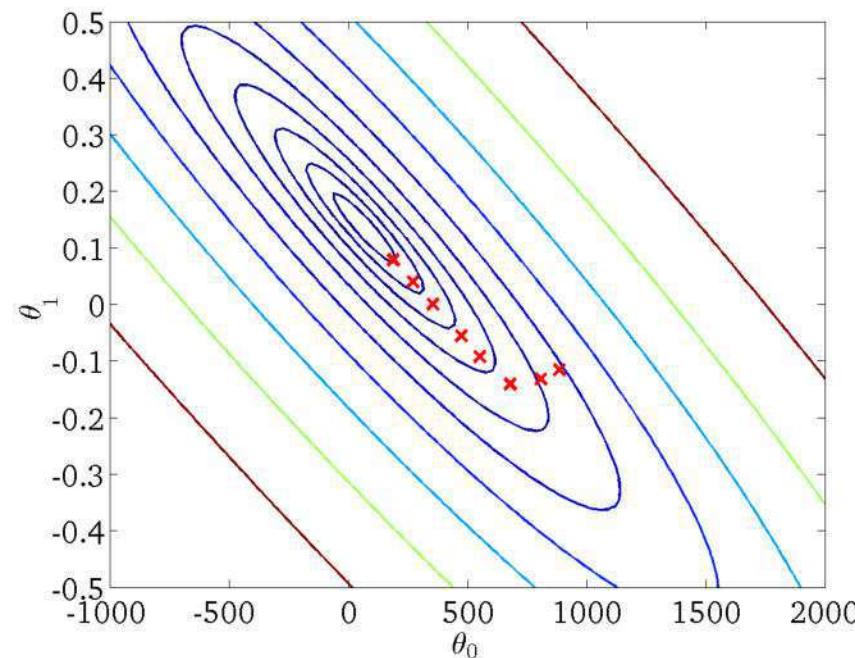
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )

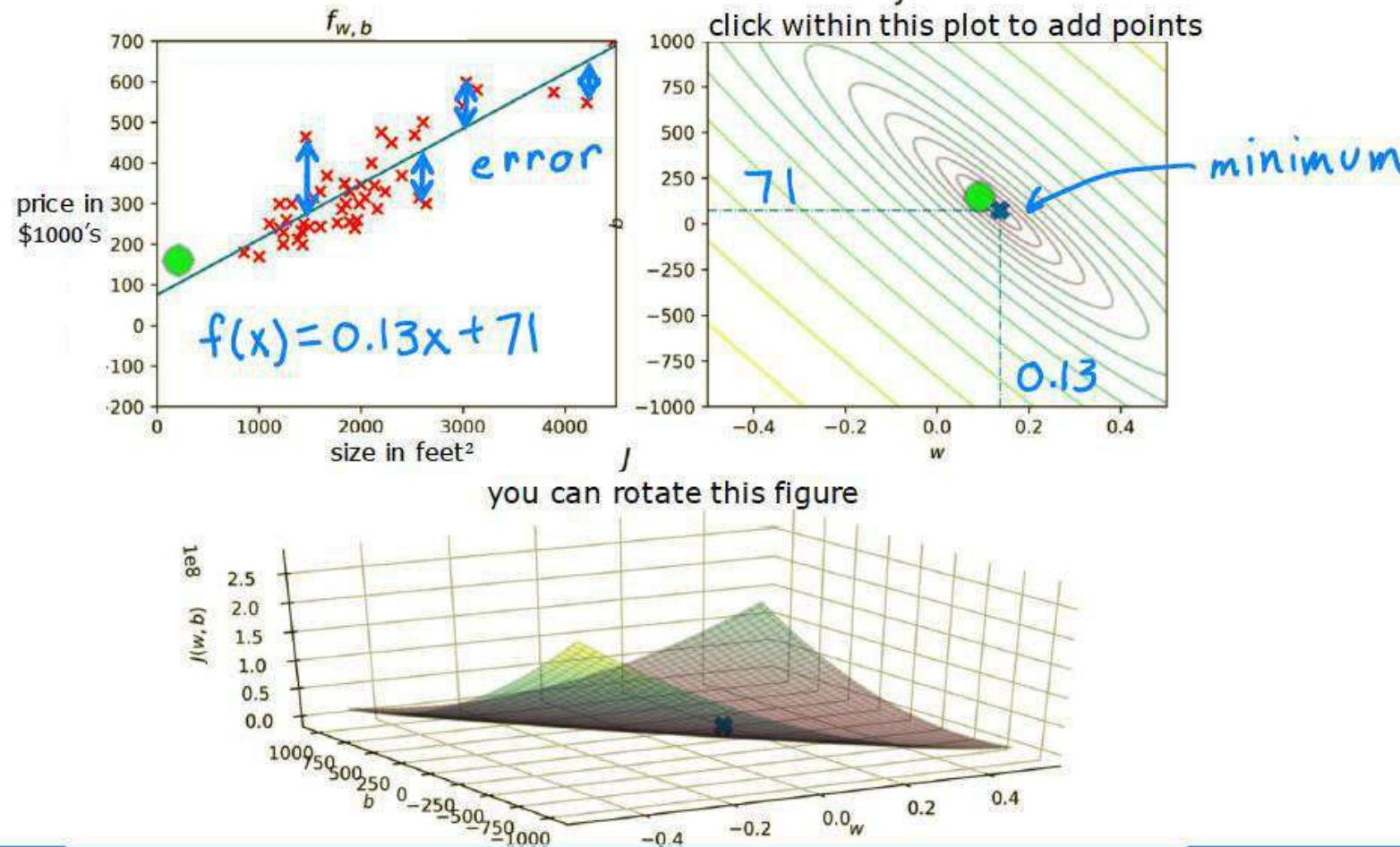


$$J(\theta_0, \theta_1)$$

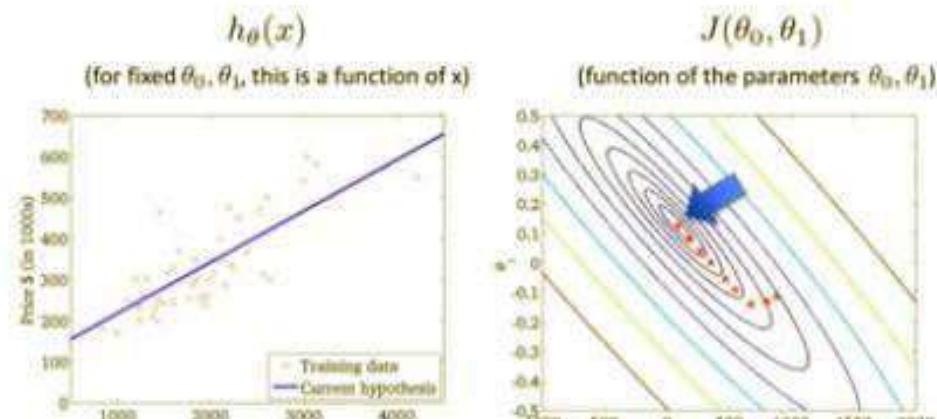
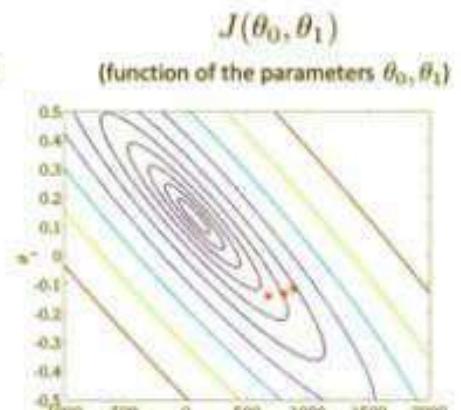
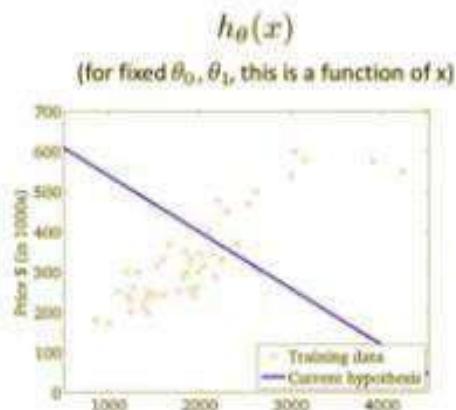
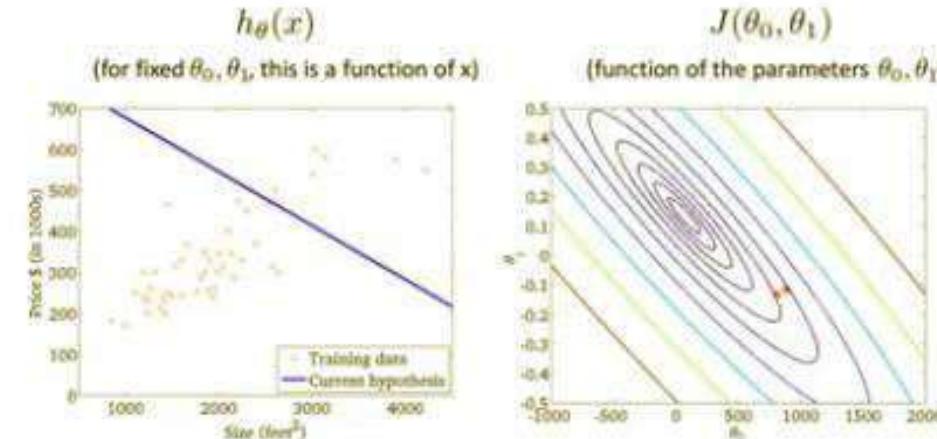
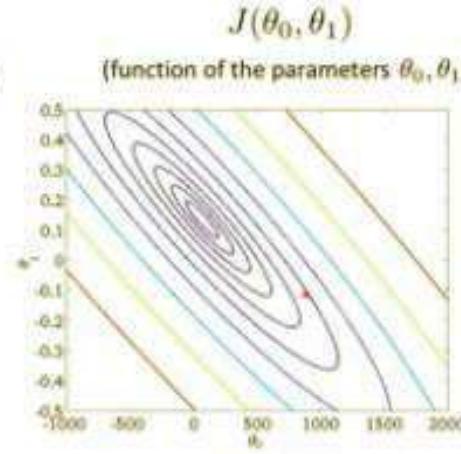
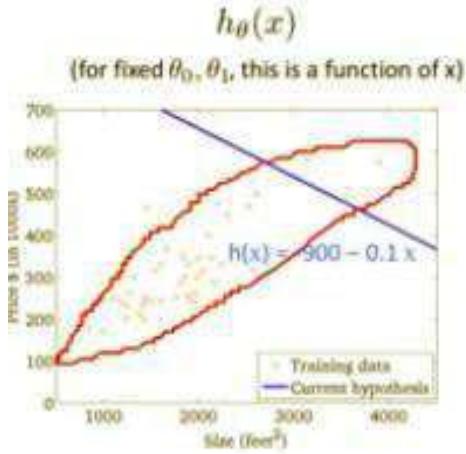
(function of the parameters  $\theta_0, \theta_1$ )



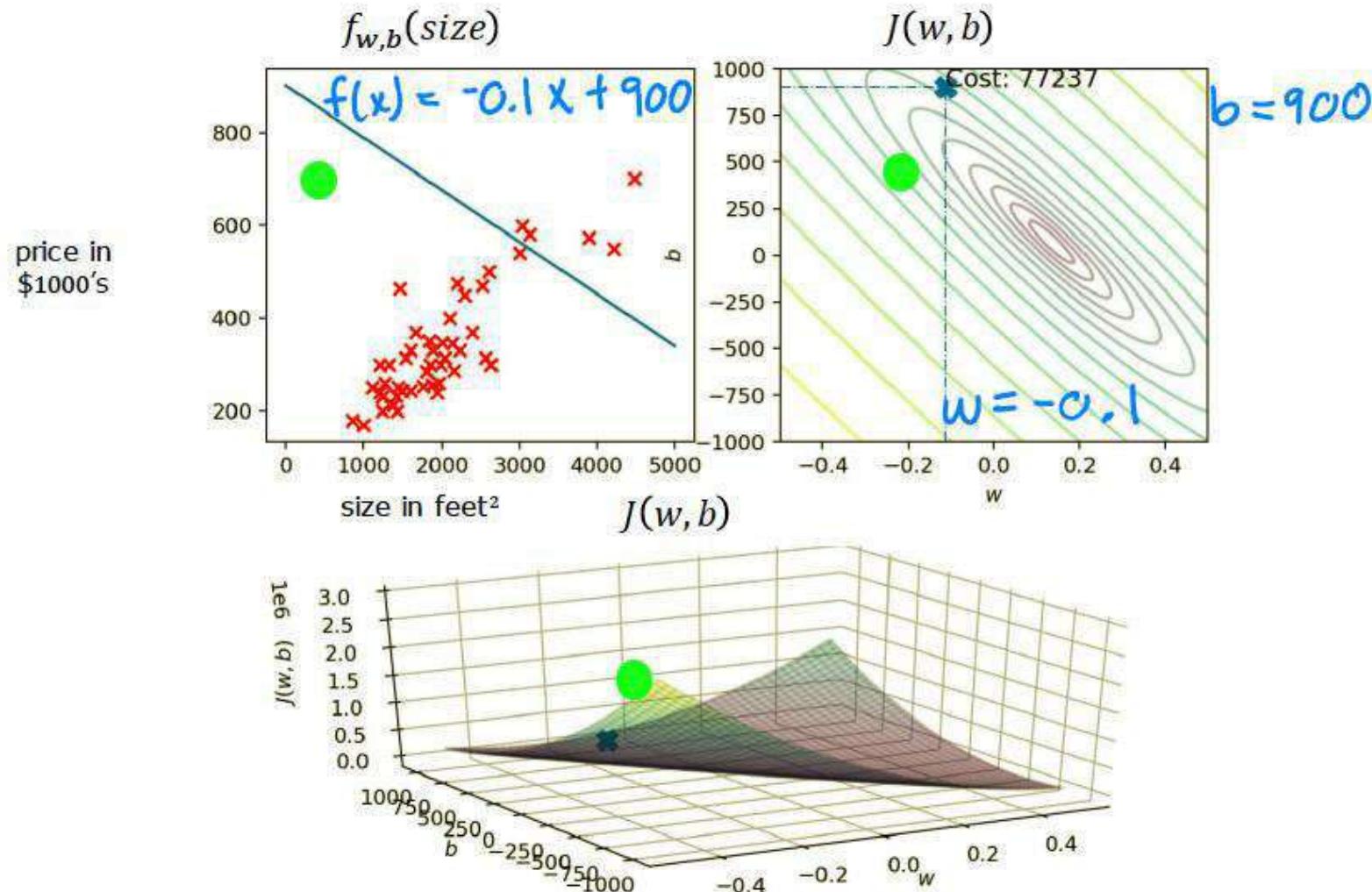
# Gradient Descent



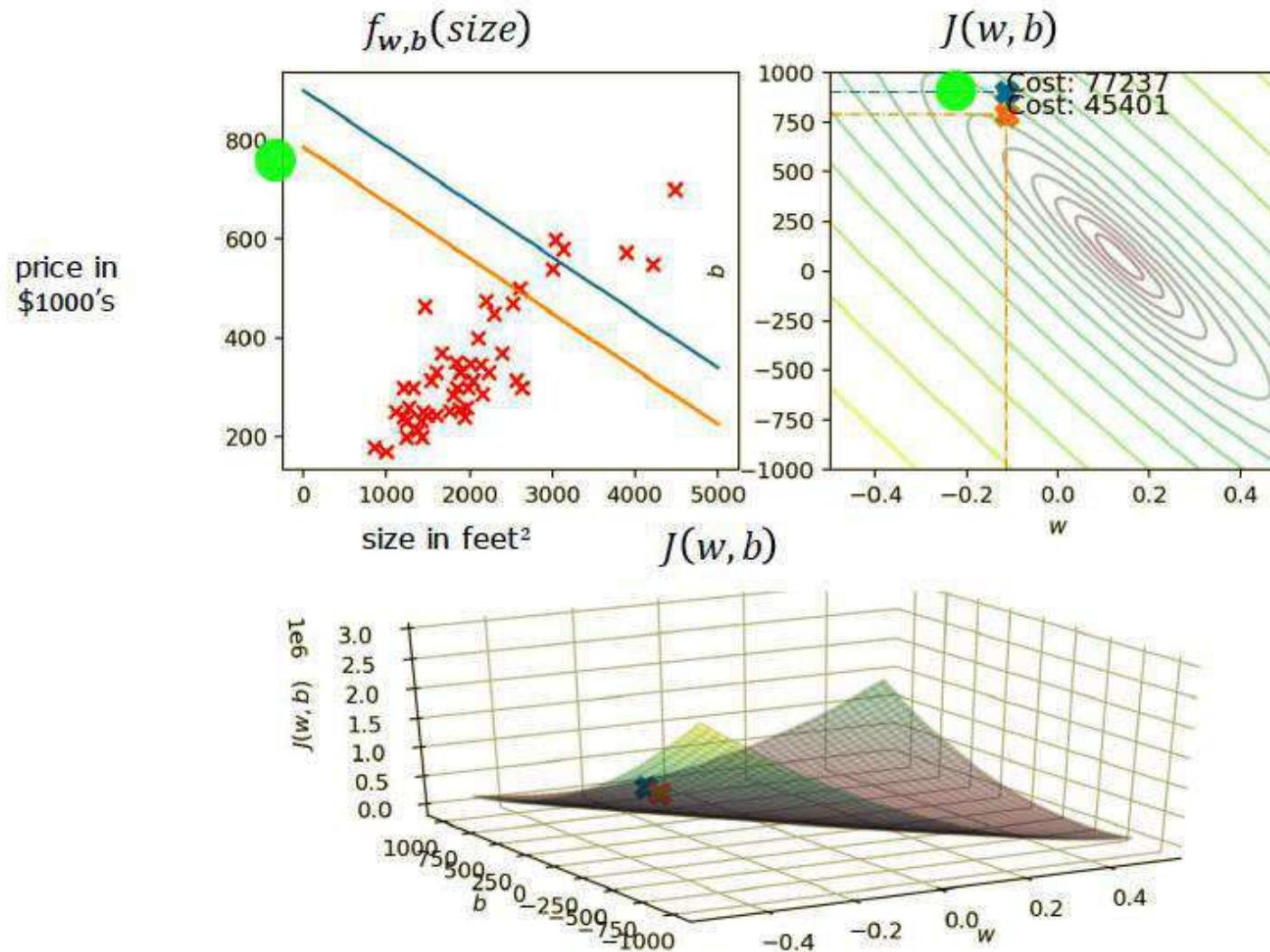
# Gradient Descent



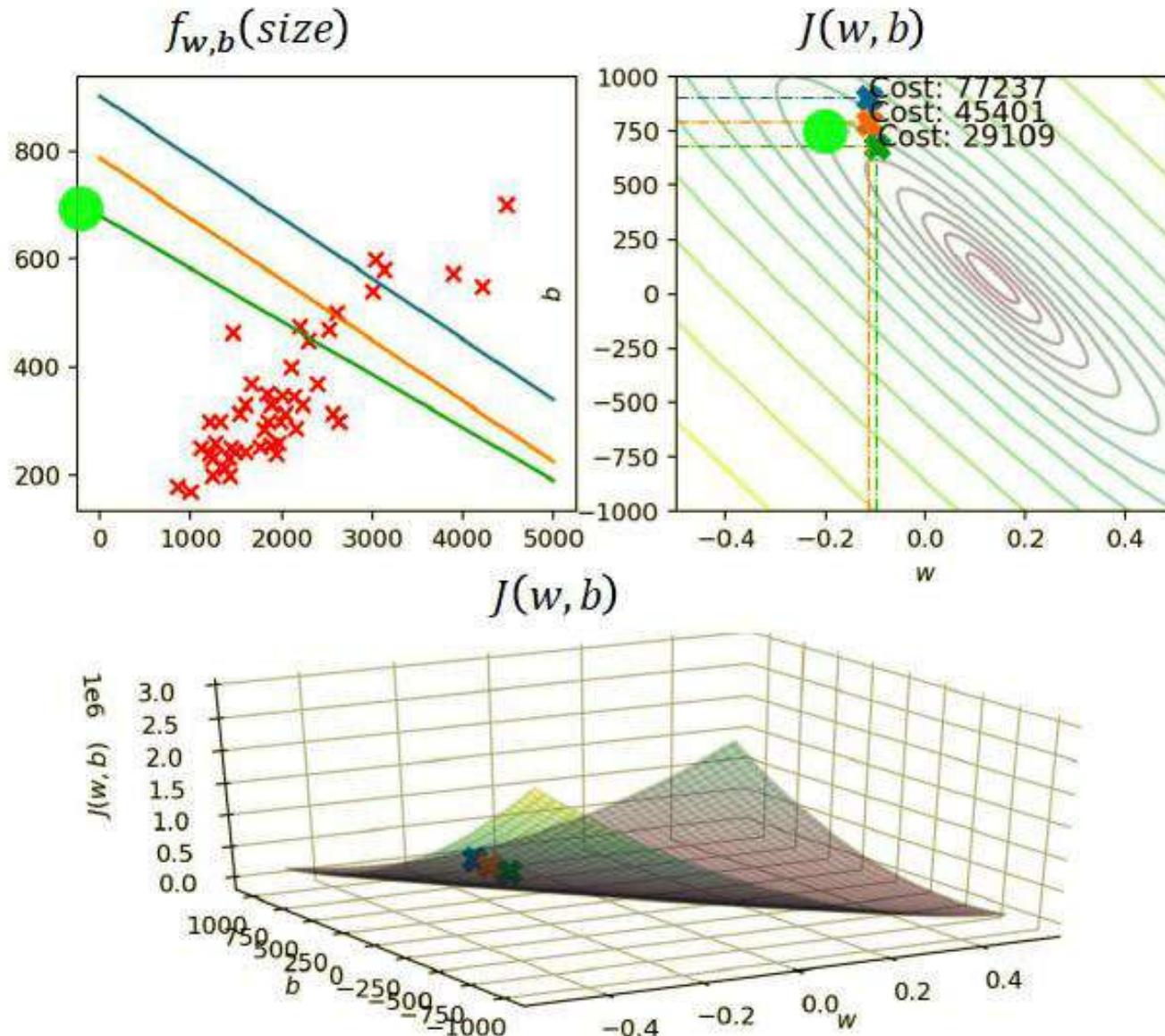
# Running Gradient Descent



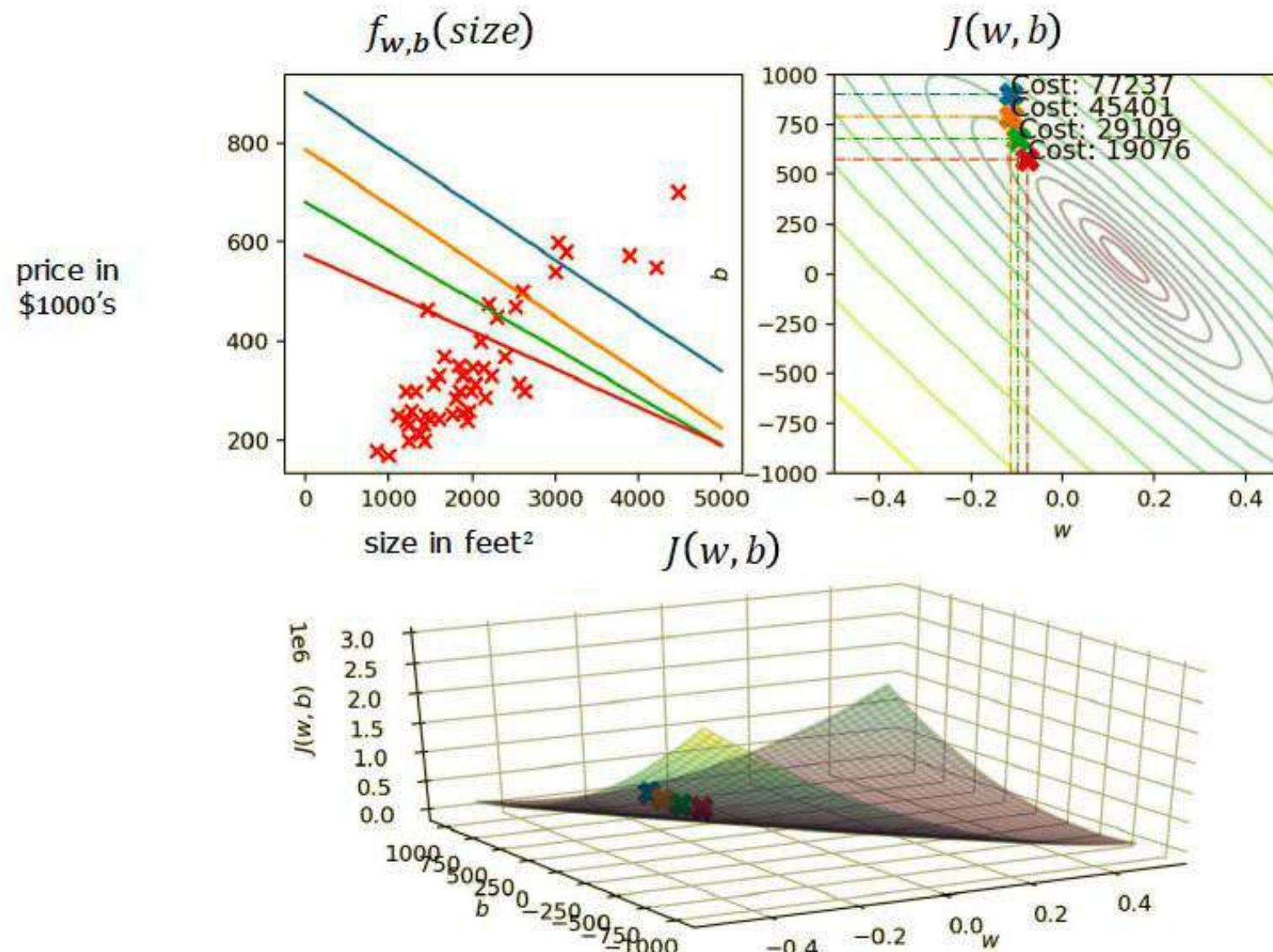
# Running Gradient Descent



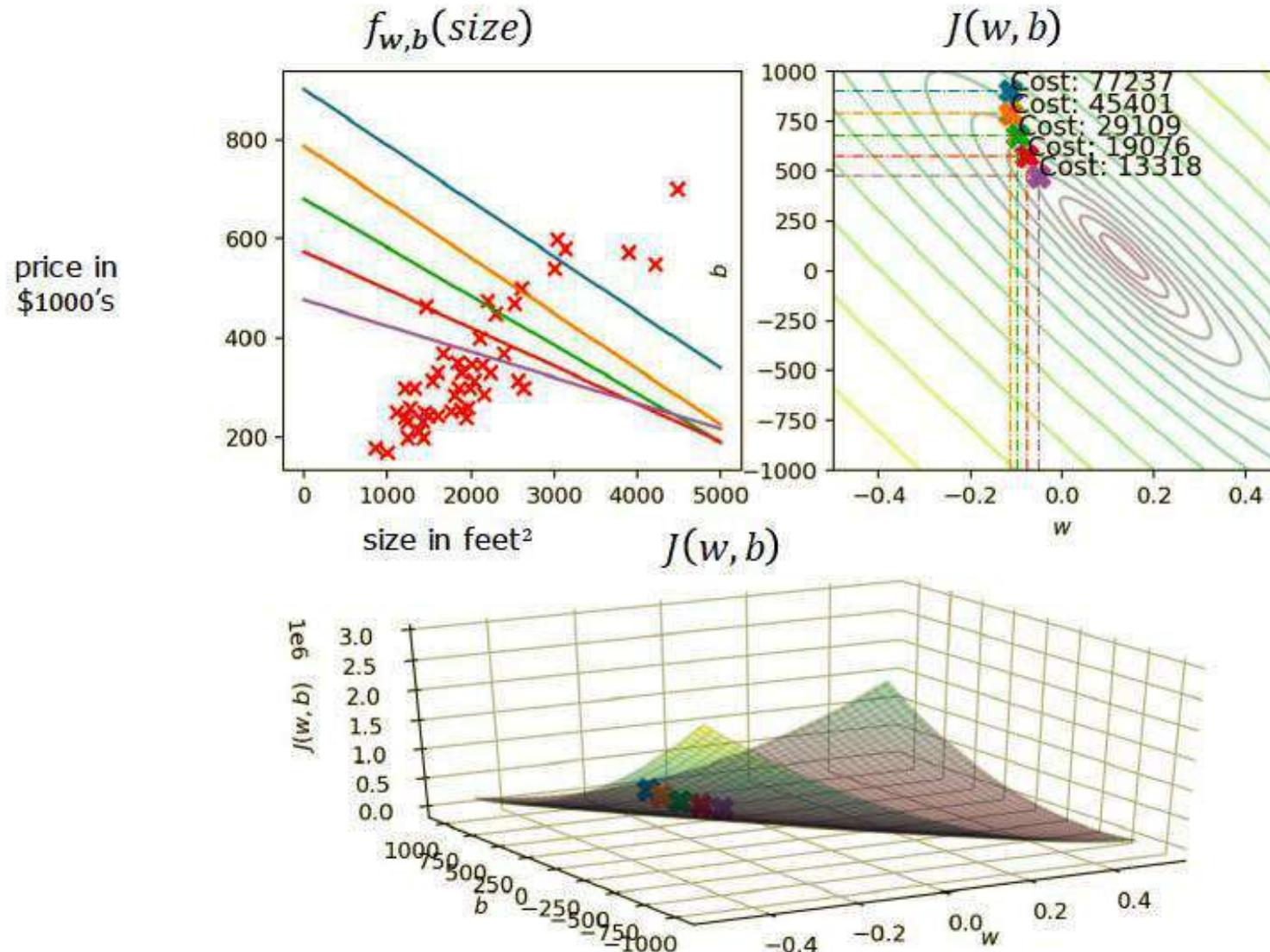
# Running Gradient Descent



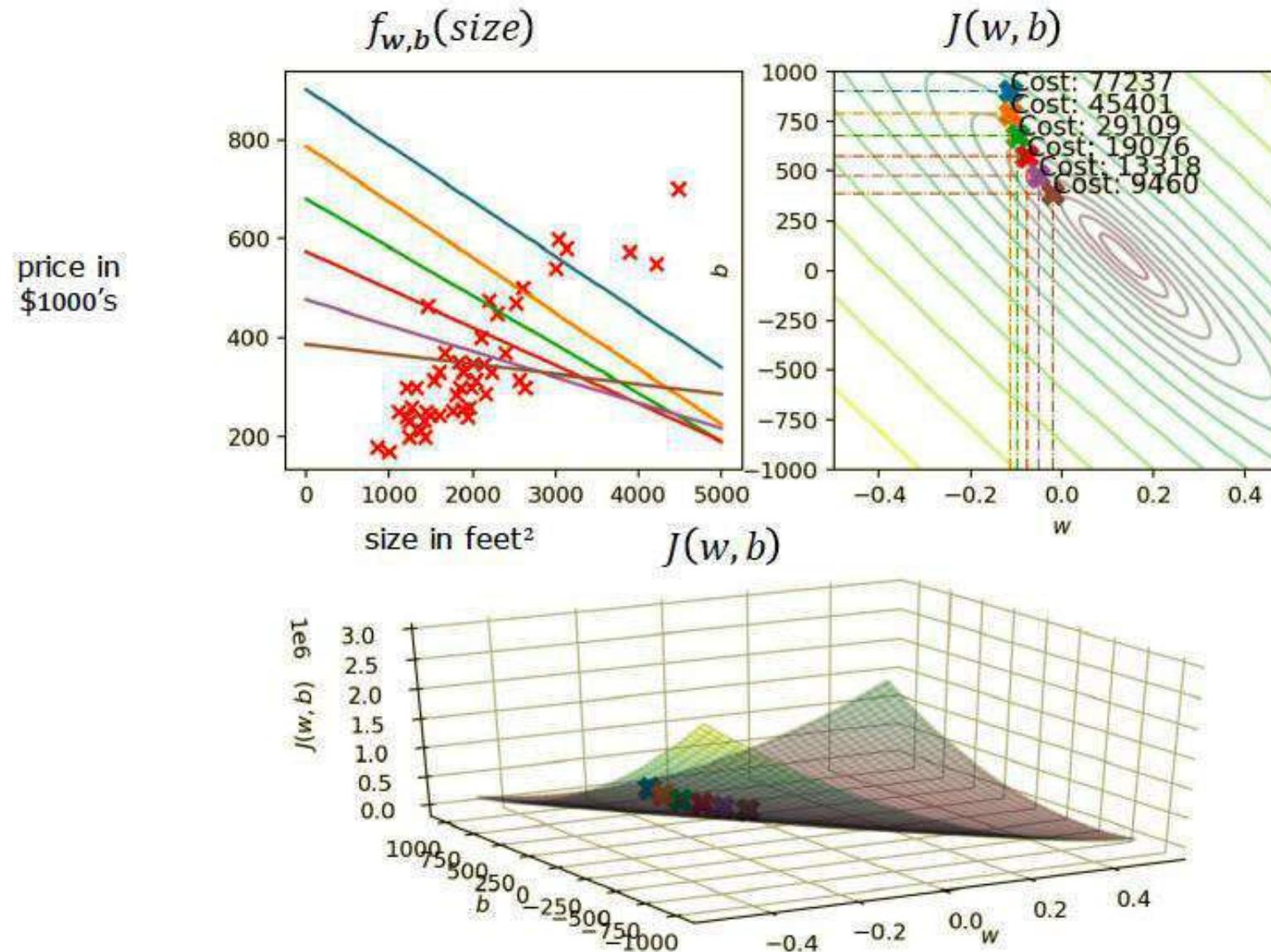
# Running Gradient Descent



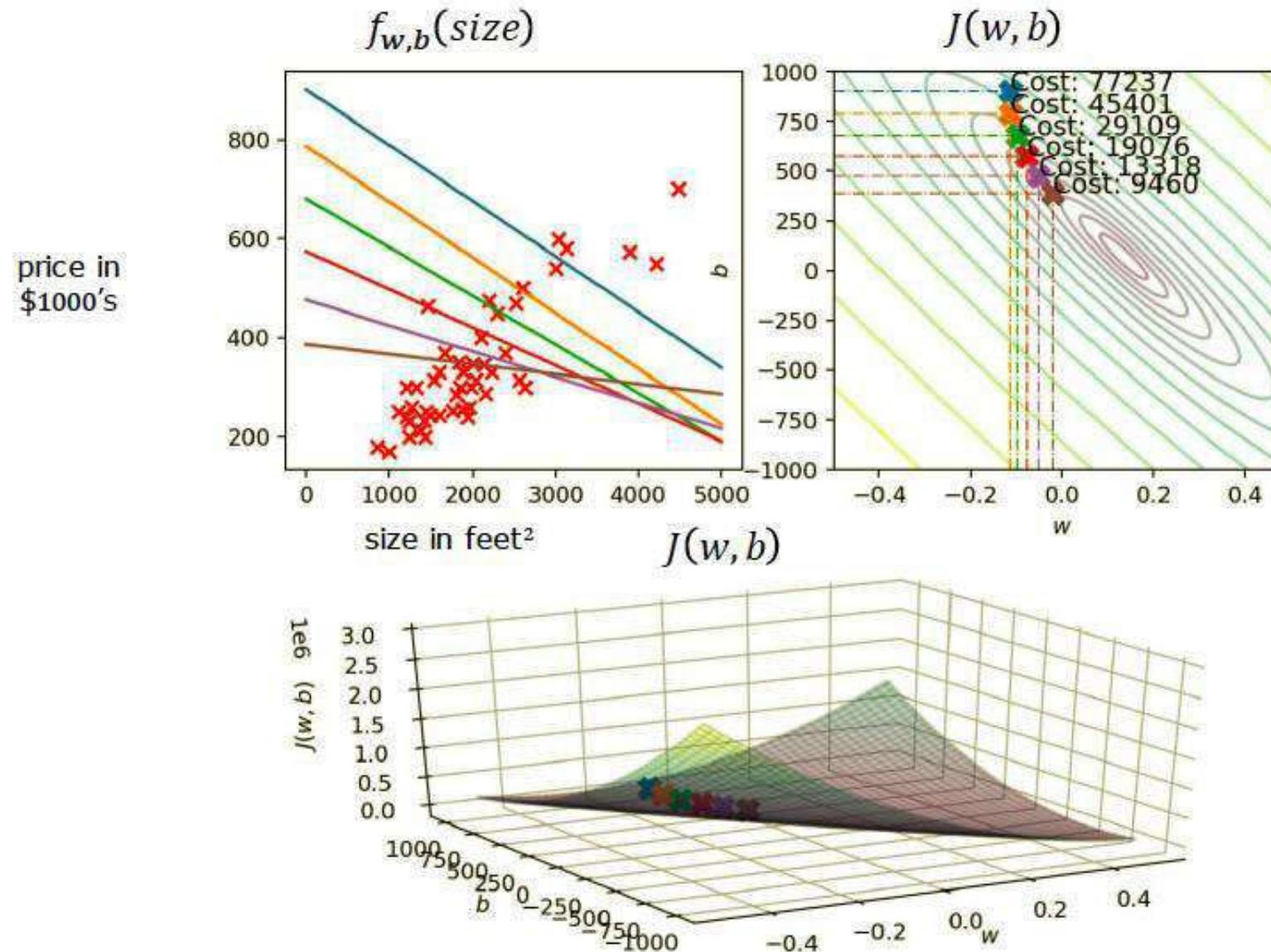
# Running Gradient Descent



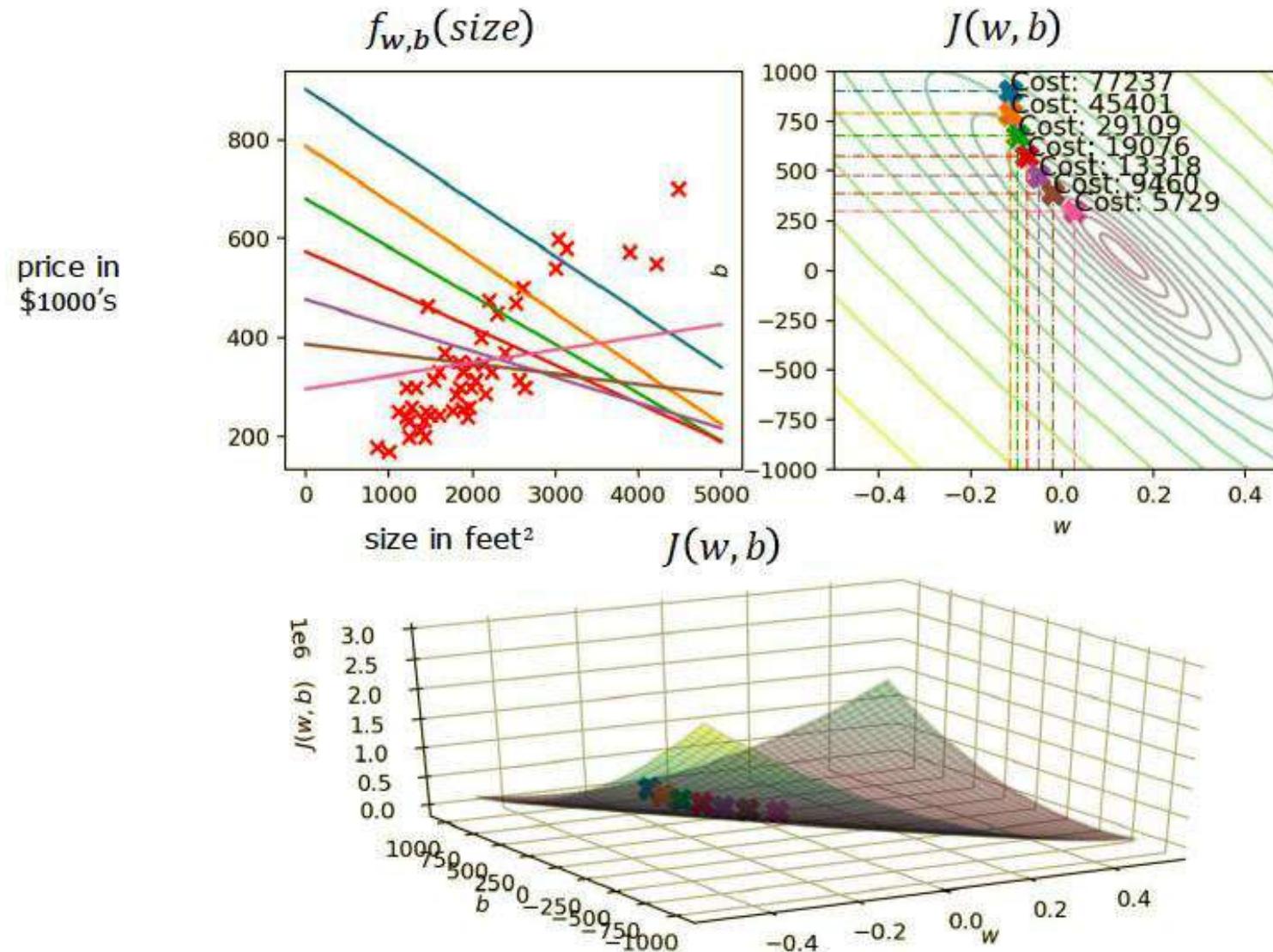
# Running Gradient Descent



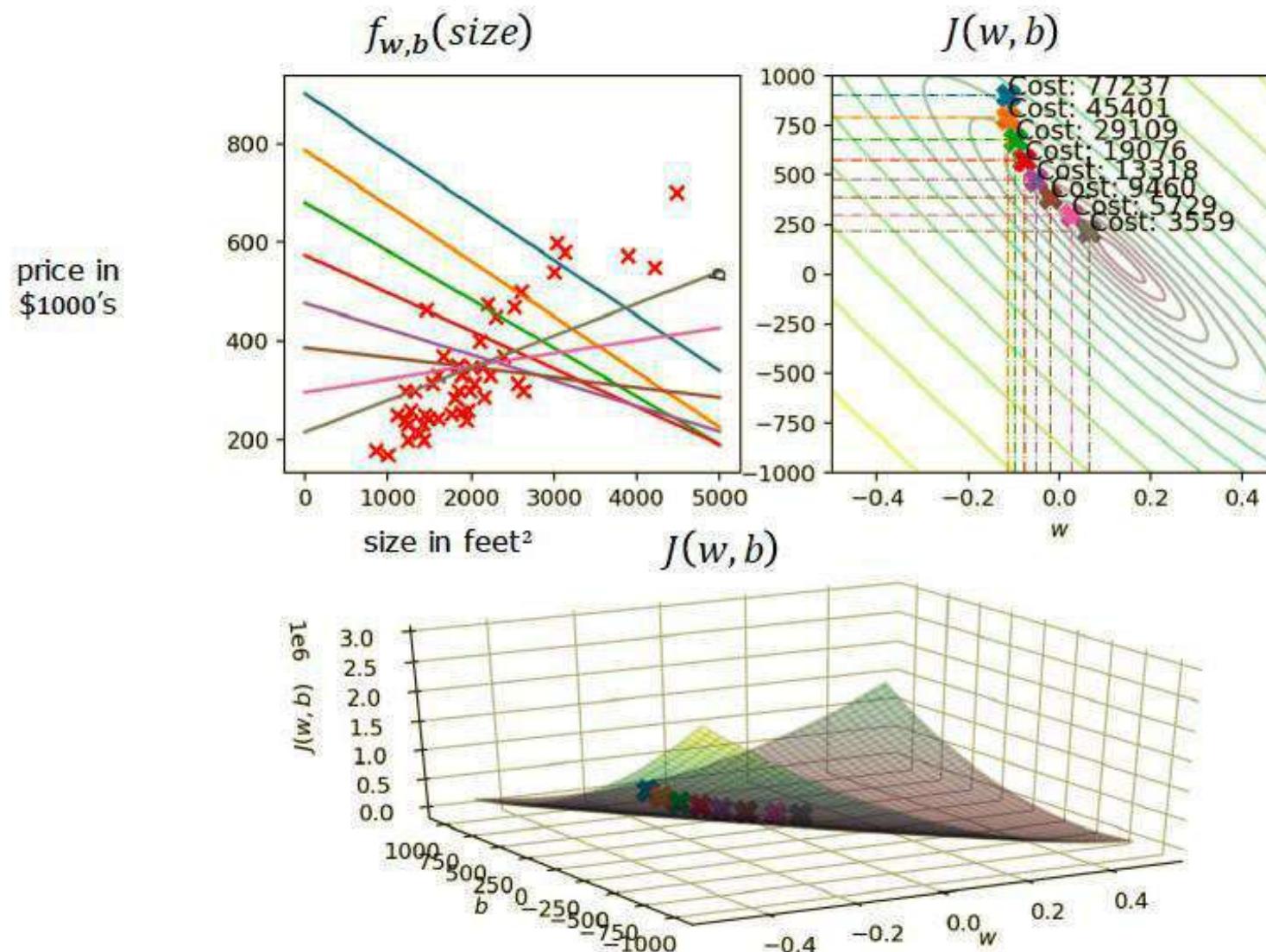
# Running Gradient Descent



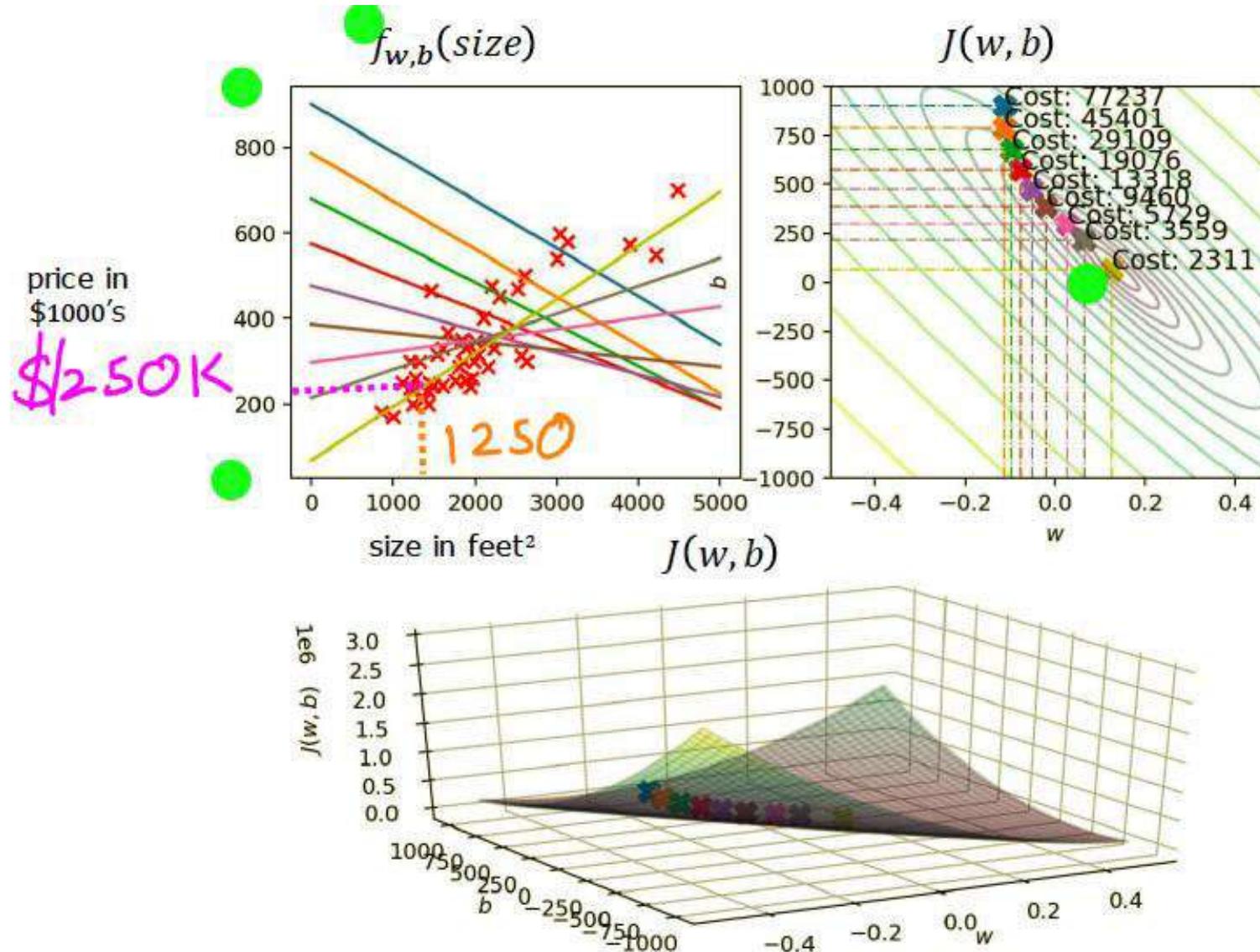
# Running Gradient Descent



# Running Gradient Descent

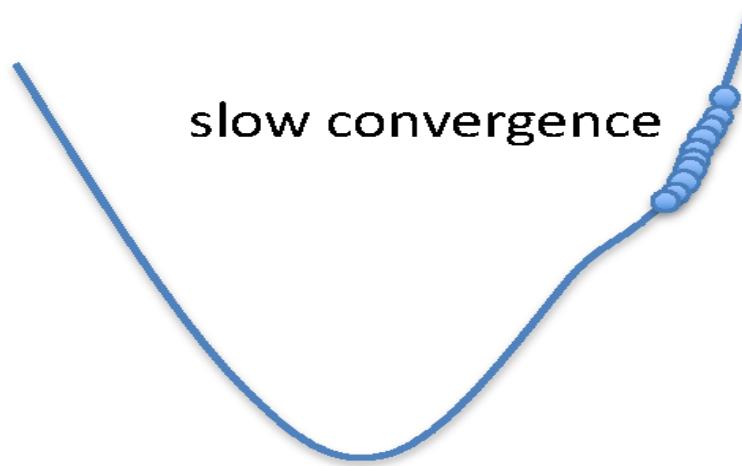


# Running Gradient Descent

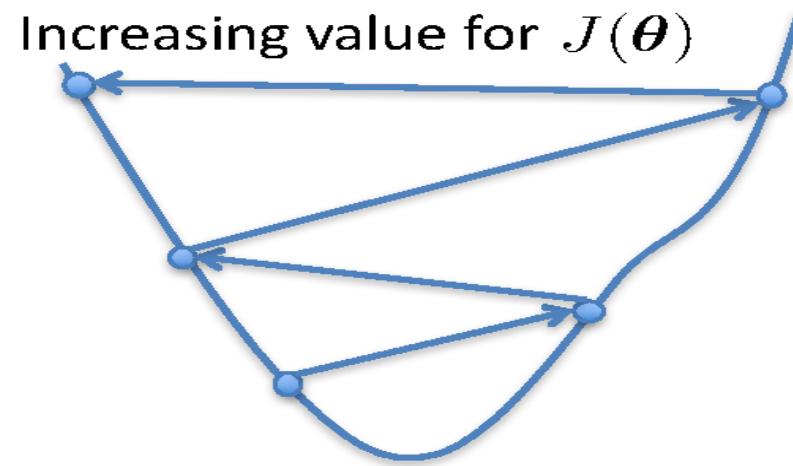


# Choosing Step Size

$\alpha$  too small



$\alpha$  too large

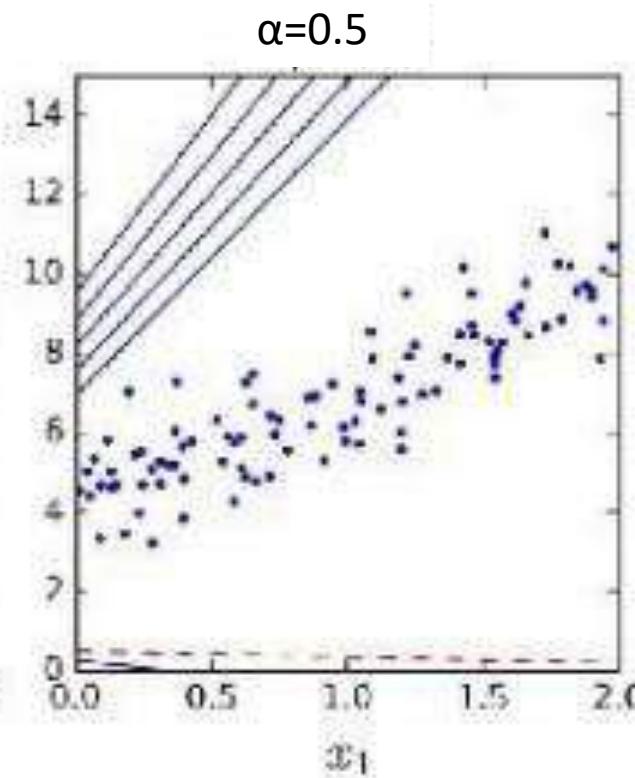
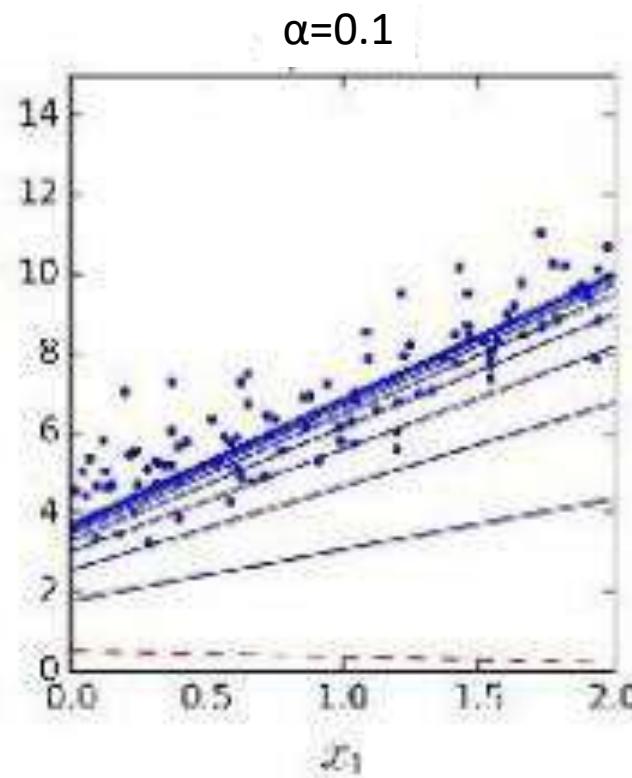
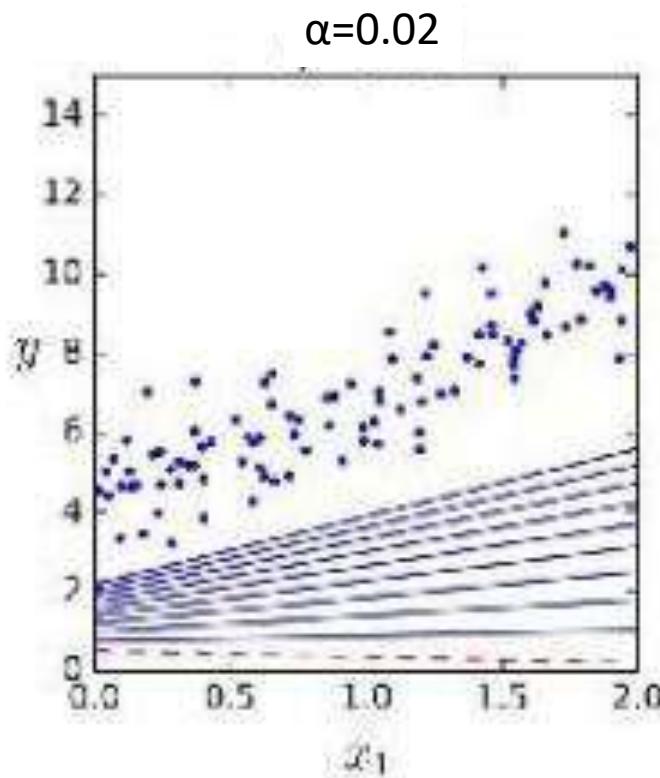


- May overshoot the minimum
- May fail to converge
- May even diverge

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  $\alpha$

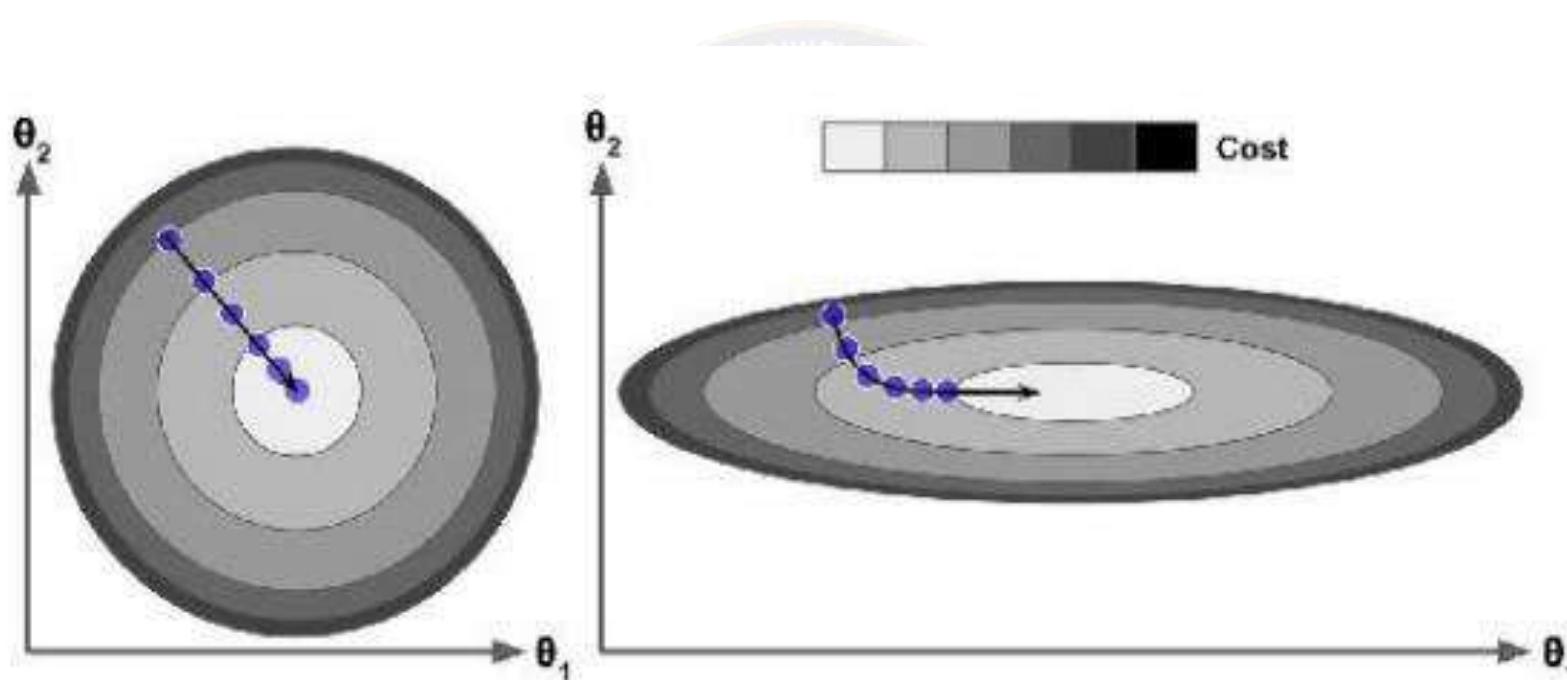
# Impact of Learning Rate



# Feature Normalization

**Gradient Descent can be slow when feature scales are widely different**

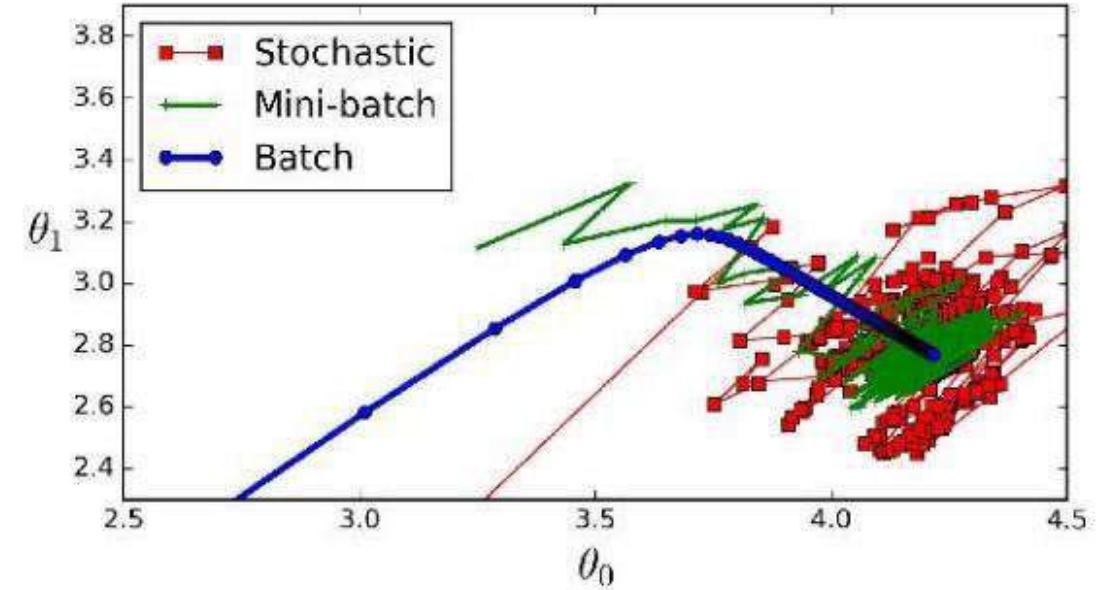
- Normalization of feature values for same variance needed for faster convergence



# Impact Due To Batch Size

## Choice of batch size impacts the rate of convergence of gradient descent (GD)

- In Batch GD, entire training set is used to calculate the training error in each iteration/epoch and gradient is calculated and used for weight updates
  - Converges in least number of iterations, i.e., rate of convergence is highest [ $O(1/\text{iterations})$ ]
  - Computation requirement per iteration is highest
  - Memory requirement is also highest
- In Stochastic gradient descent, a **randomly** selected training instance is used
  - Converges in highest number of iterations, i.e., rate of convergence is slowest [ $\sim O(1/\sqrt{\text{iterations}})$ ]
  - Computation requirement per iteration is lowest
  - Memory requirement is also lowest
- In mini batch GD, a subset of training data of size, say 64,128, 256 is used
  - very efficient implementation possible leveraging vector processing using GPUs

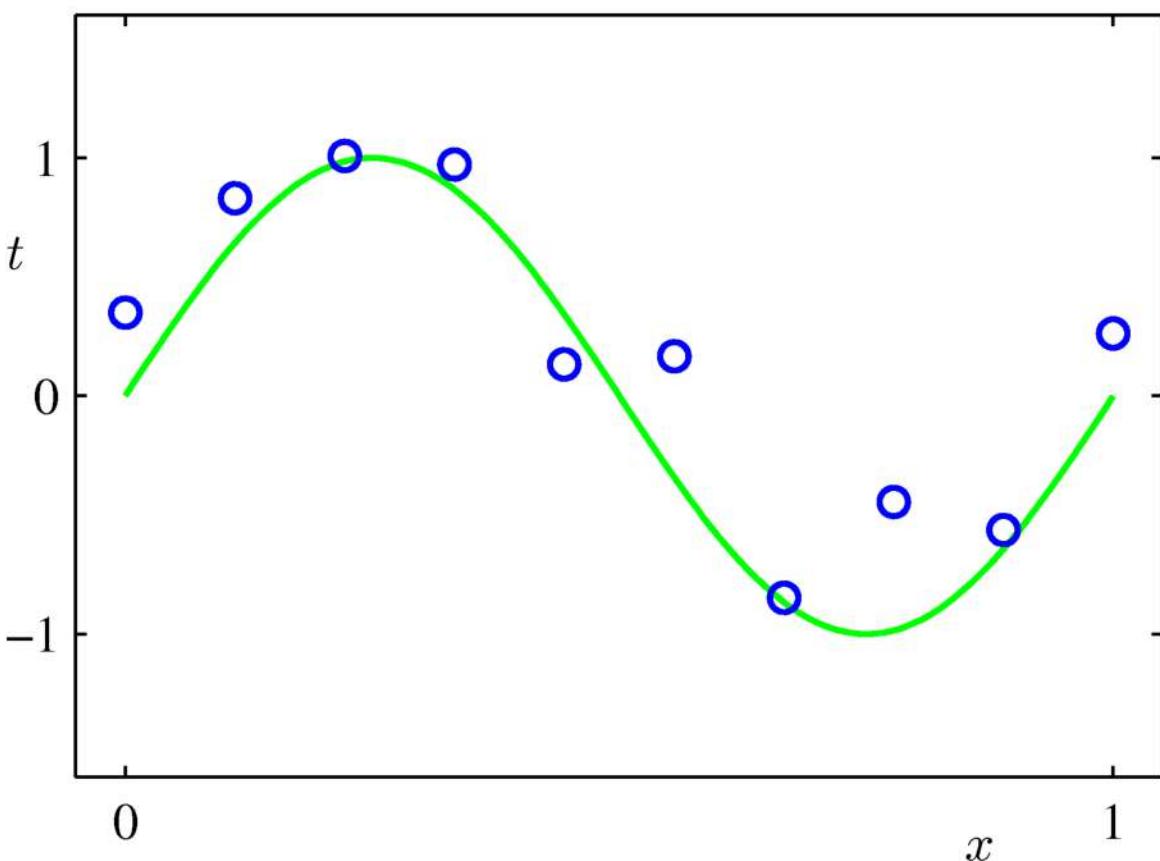


# Extending to More Complex Model

- 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:  $x_3 = x_1 \cdot x_2$

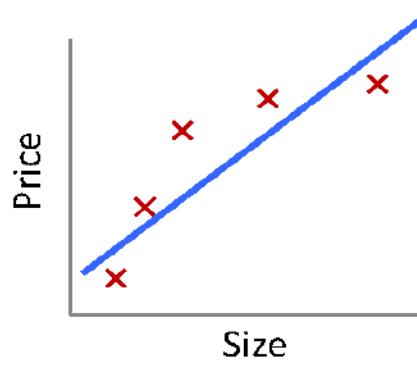
This allows use of linear regression techniques  
to fit non-linear datasets.

# Fitting a Polynomial Curve

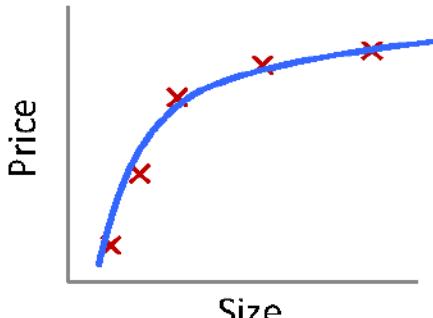


$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j = \sum_{j=0}^p \theta_j z_j$$
$$z_j = x^j$$

# Quality of Fit

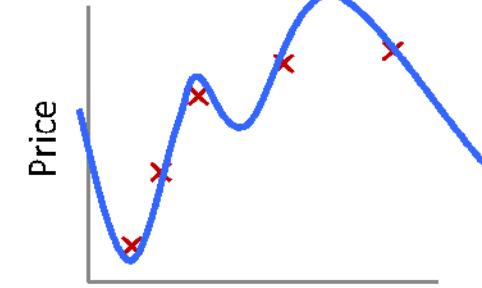


Underfitting  
(high bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$

Correct fit



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

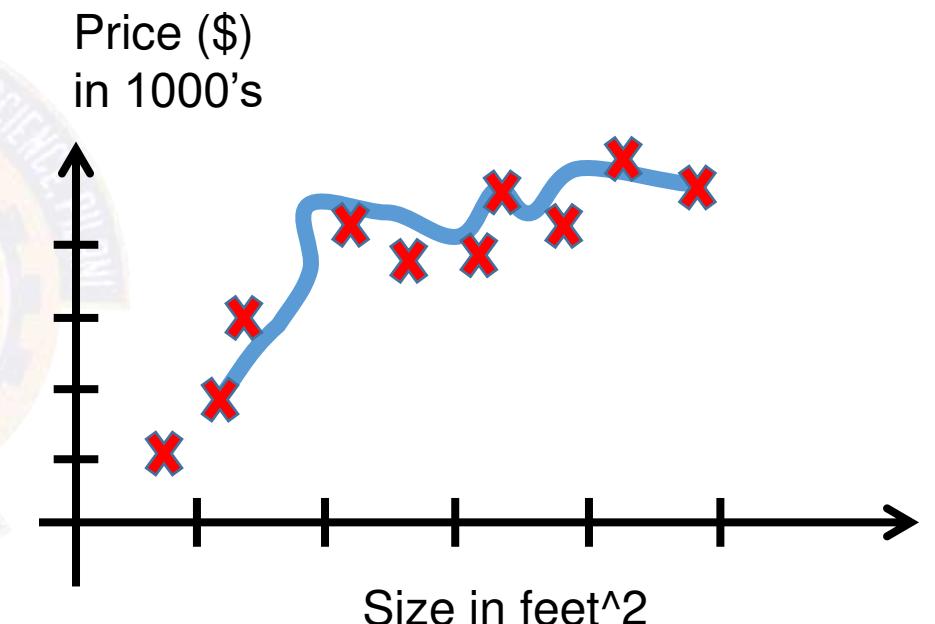
Overfitting  
(high variance)

## Overfitting:

- The learned hypothesis may fit the training set very well ( $J(\theta) \approx 0$ )
- ...but fails to generalize to new examples

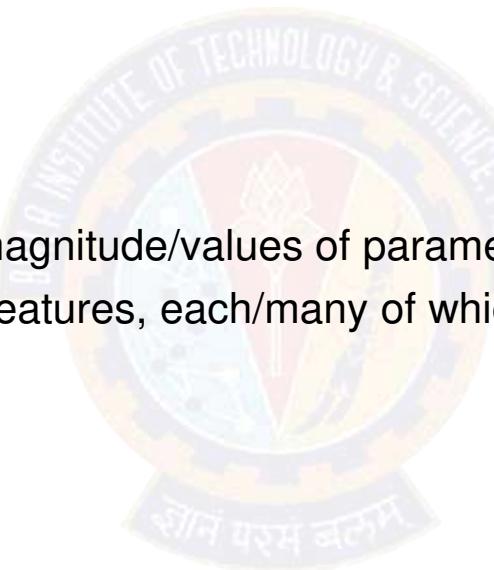
# Addressing overfitting

- $x_1$  = 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
- :
- $x_{100}$



# Addressing overfitting

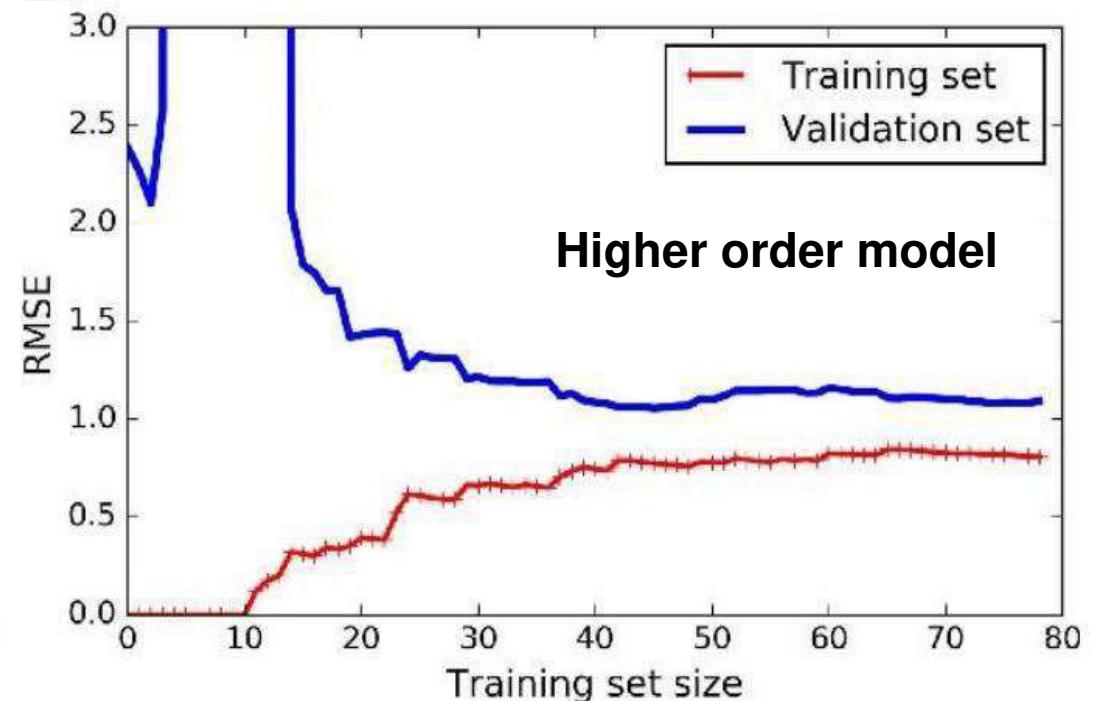
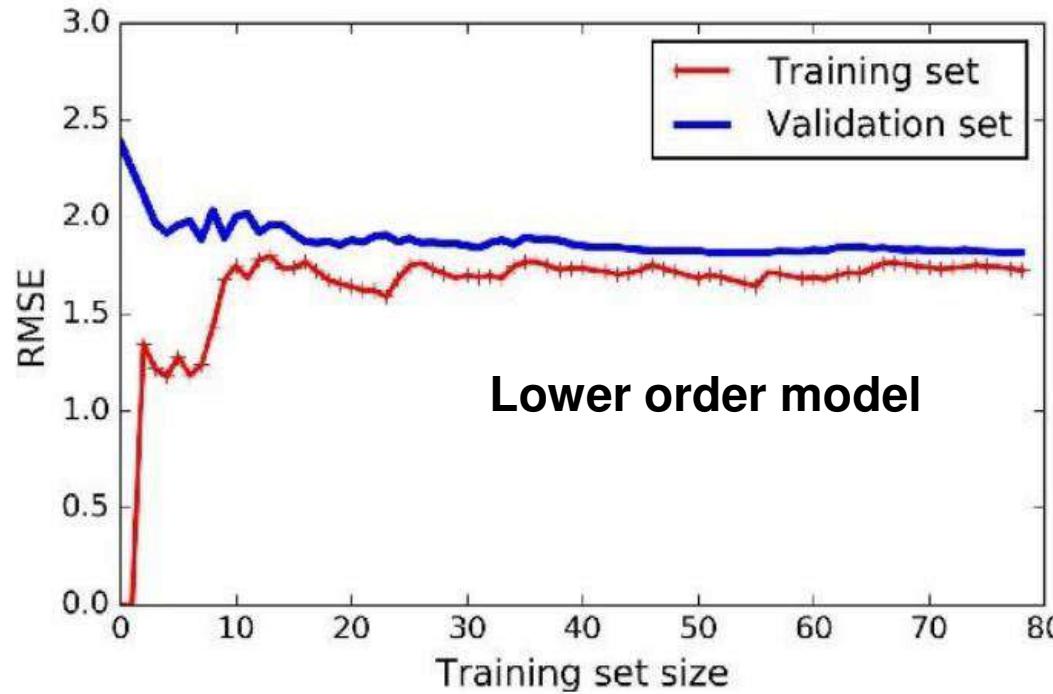
- Reduce number of features.
  - Manually select which features to keep.
  - Model selection algorithm
- Regularization.
  - Keep all the features, but reduce magnitude/values of parameters  $\theta_j$ .
  - Works well when we have a lot of features, each/many of which contributes a bit to predicting  $y$ .



# Effect of Training Size on Overfitting

Size of training dataset needs to be large to prevent overfitting

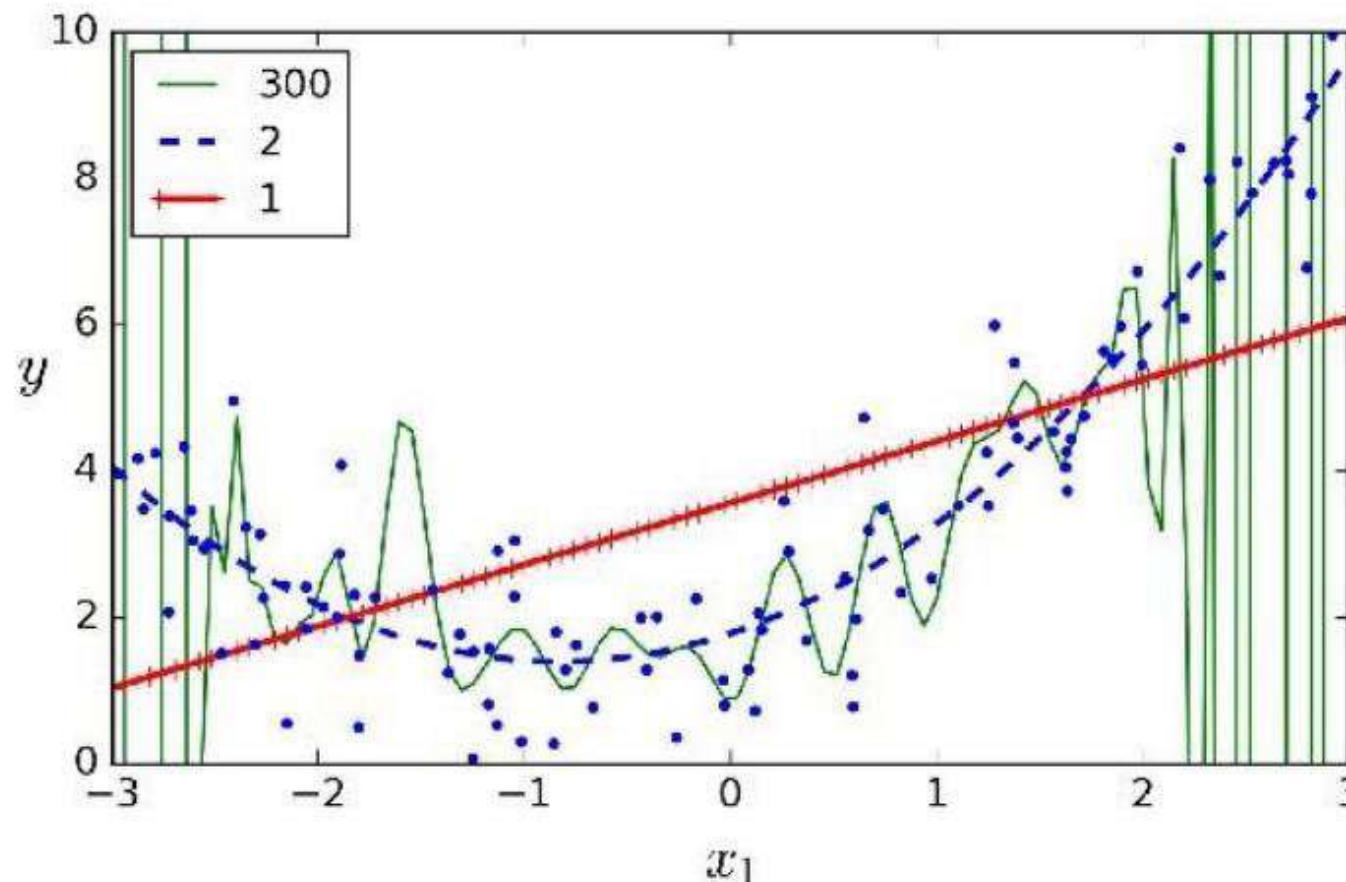
- when higher order model is used.



# Polynomial Fitting can lead to Overfitting

Underlying target function is quadratic

- Linear model results in underfitting with large bias
- Polynomial of order 300 results in a large variance

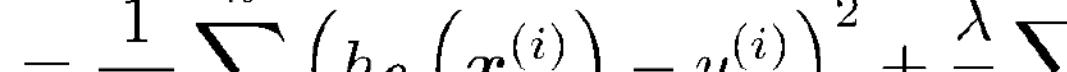


# Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of  $\theta_j$ 
  - 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)

# Ridge Regularization

- Linear regression objective function

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


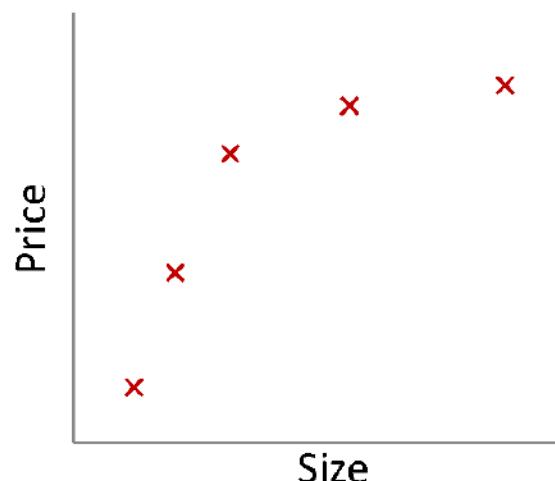
The diagram illustrates the cost function  $J(\boldsymbol{\theta})$  as a sum of two terms. The first term,  $\frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$ , is represented by a blue bracket below the equation and labeled 'model fit to data'. The second term,  $\frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$ , is represented by another blue bracket to the right of the first and labeled 'regularization'.

- $\lambda$  is the regularization parameter ( $\lambda \geq 0$ )
  - No regularization on  $\theta_0$ !

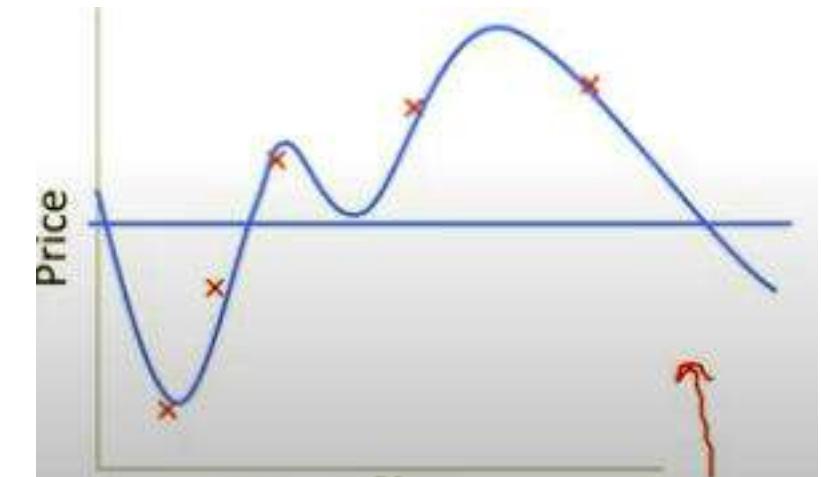
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



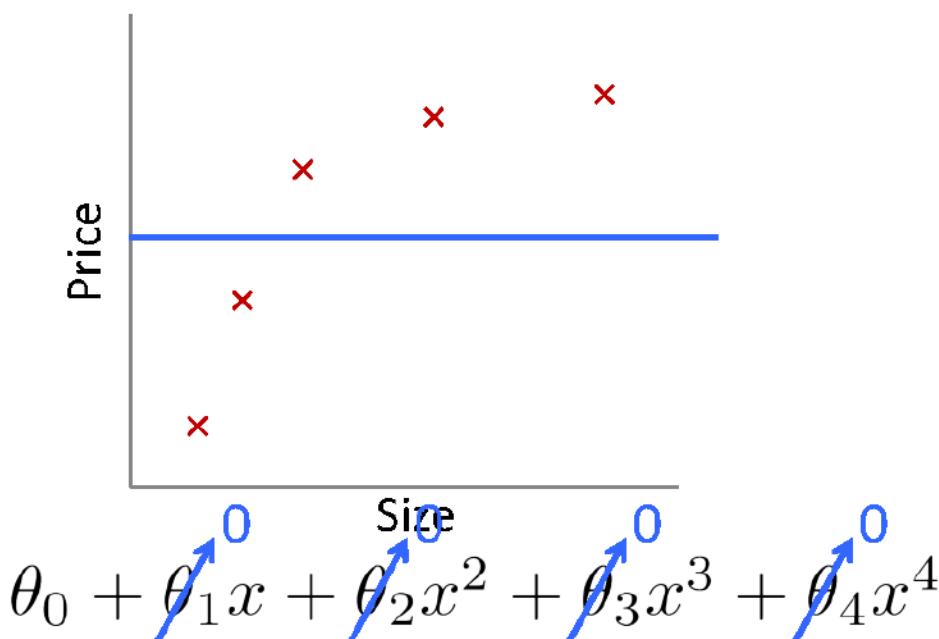
Overfitting

Based on example by Andrew Ng

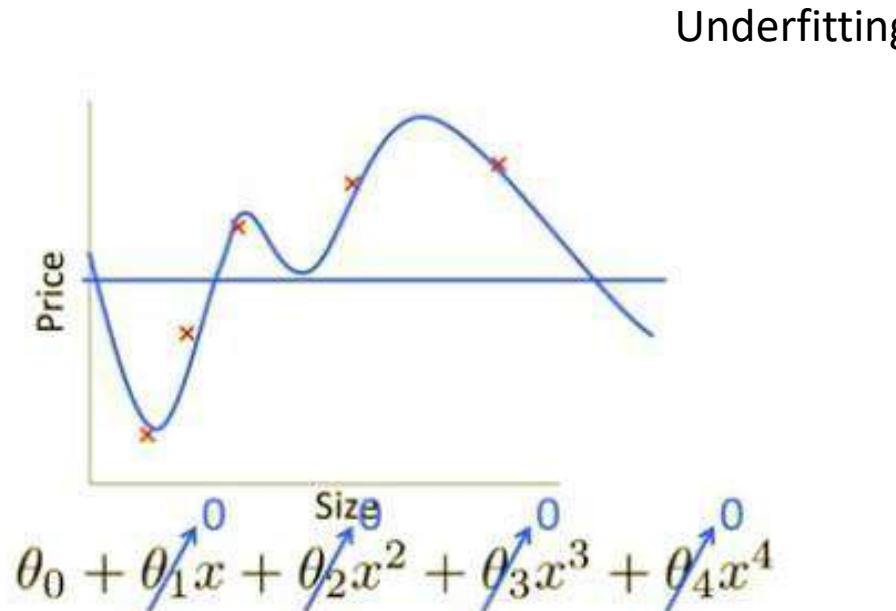
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



Based on example by Andrew Ng



# Ridge regression

## 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:

$$\frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta})$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

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

regularization 57

# Ridge Regression

Further simplified

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

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

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

- We can rewrite the gradient step as:

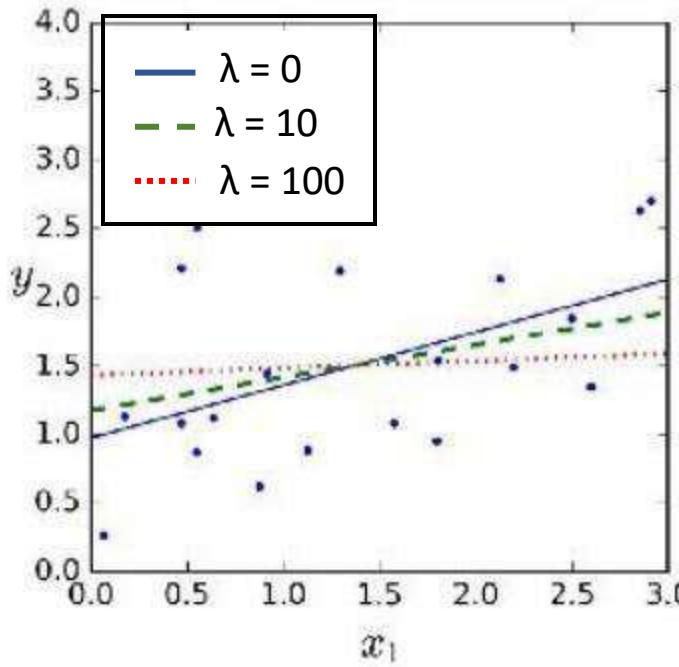
$$\theta_j \leftarrow \theta_j (1 - \alpha \lambda) - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

# Lasso Regularization

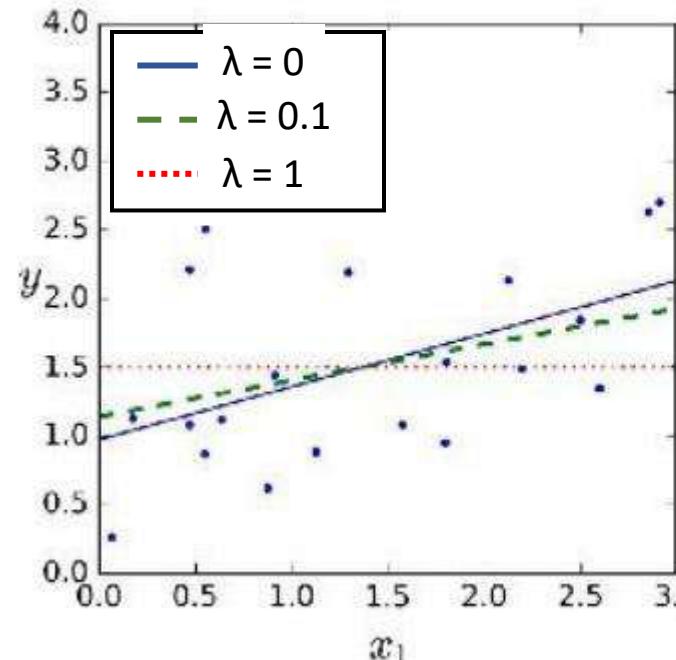
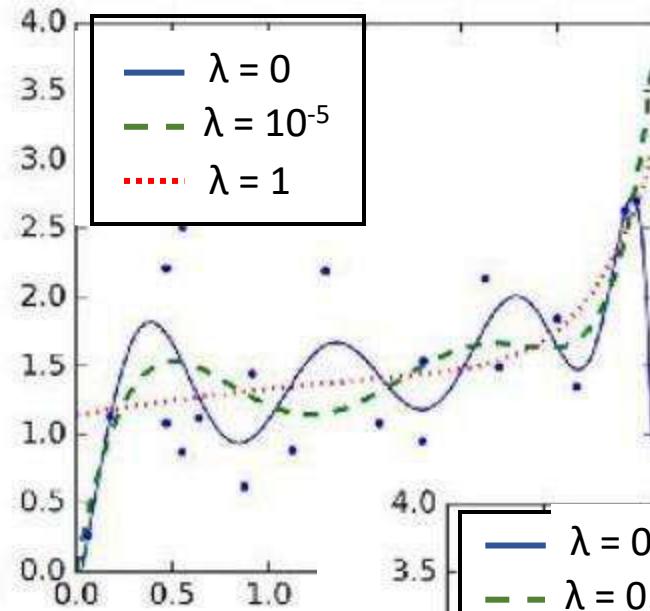
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d |\theta_j|$$

$$\theta_j = \theta_j - \frac{\alpha}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) \mathbf{x}_j^{(i)} - \alpha \lambda \text{sign}(\theta_j)$$

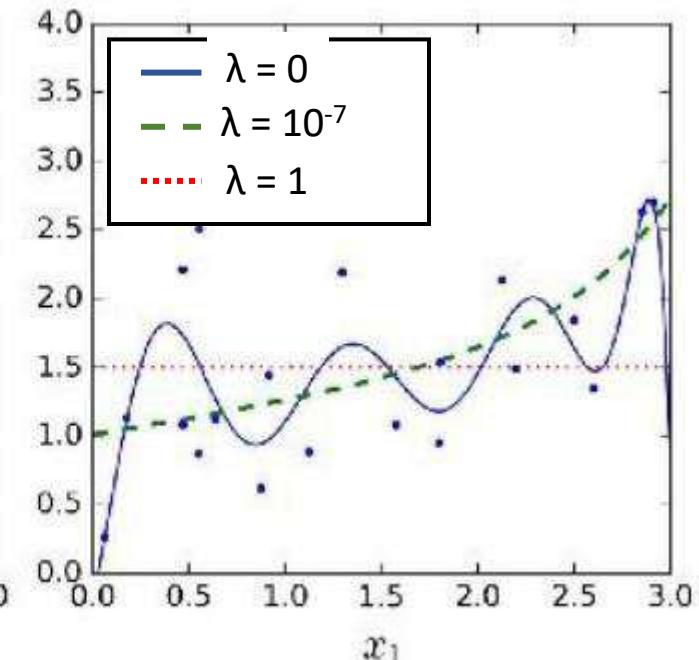
# Ridge Vs Lasso Regularization



Ridge



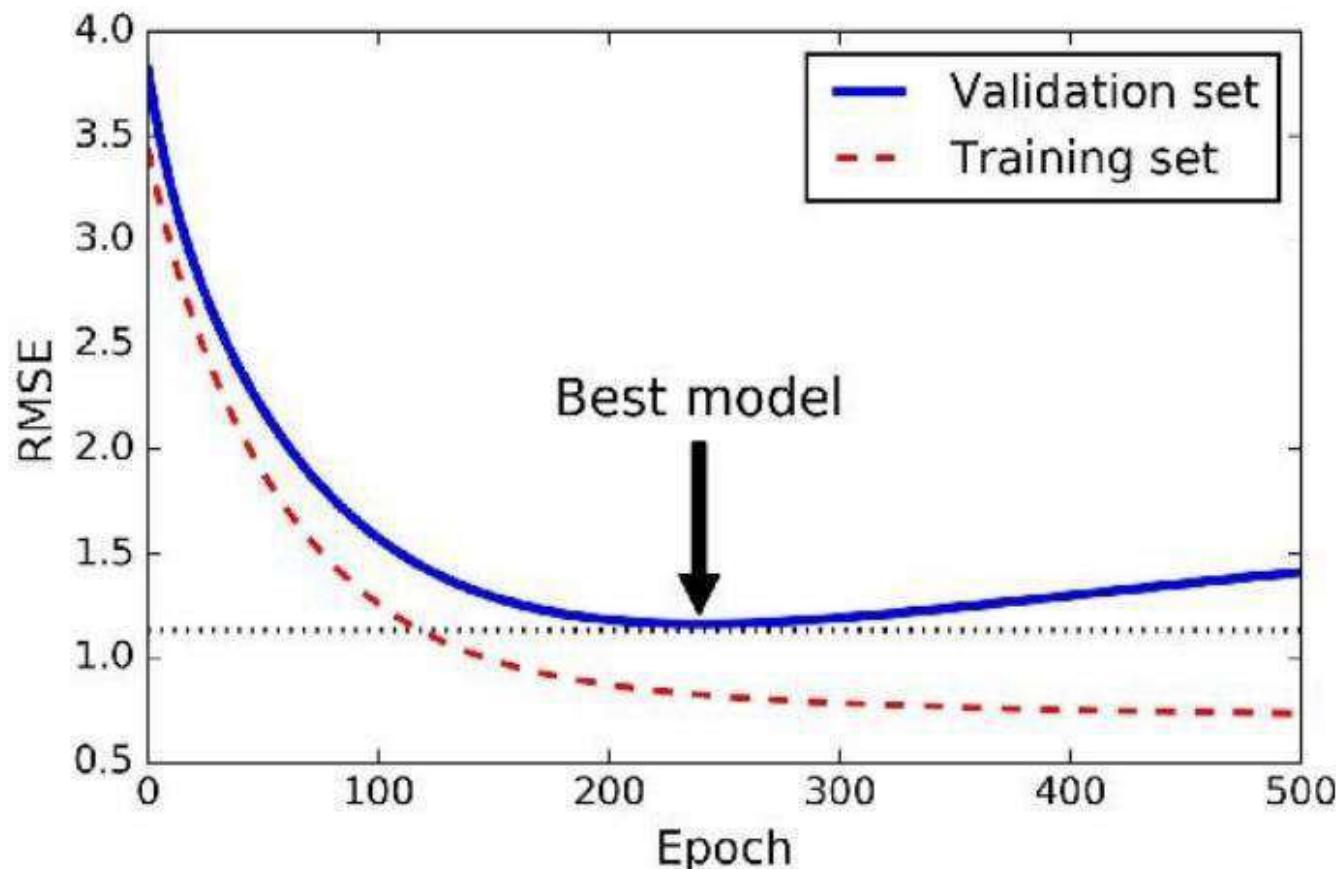
Lasso



# Early Stopping

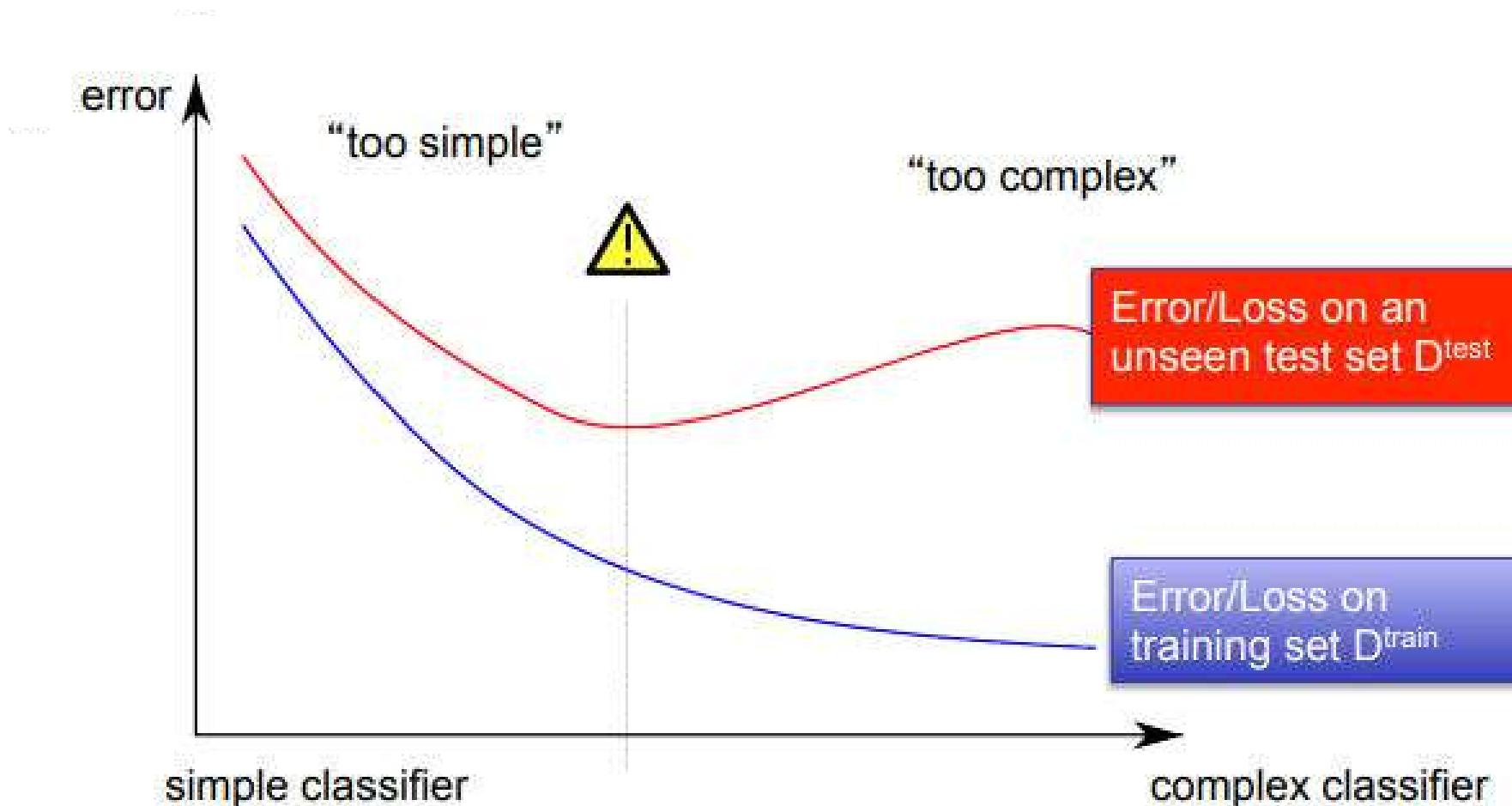
## Do Not Over train to prevent overfitting

- Stop training once error on the validation set starts showing an upward trend, even if the error on the training set keeps decreasing



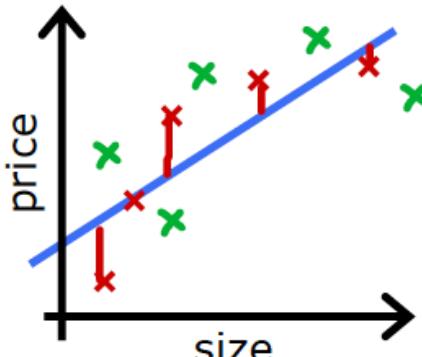
# Bias Variance Tradeoff

Bias/Variance is a Way to Understand Overfitting and Underfitting



# Bias and Variance

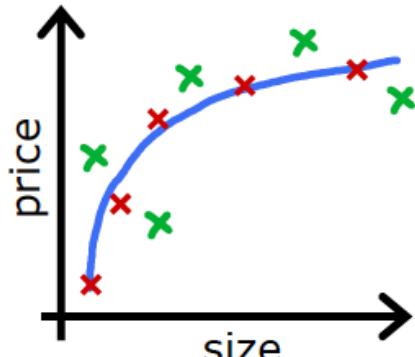
## Bias/variance



$$f_{\vec{w},b}(x) = w_1x + b$$

→ High bias  
(underfit)

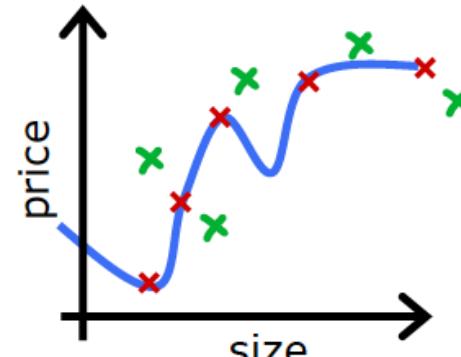
$d = 1$   $J_{train}$  is high  
 $J_{cv}$  is high



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$$

"Just right"

$d = 2$   $J_{train}$  is low  
 $J_{cv}$  is low



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$$

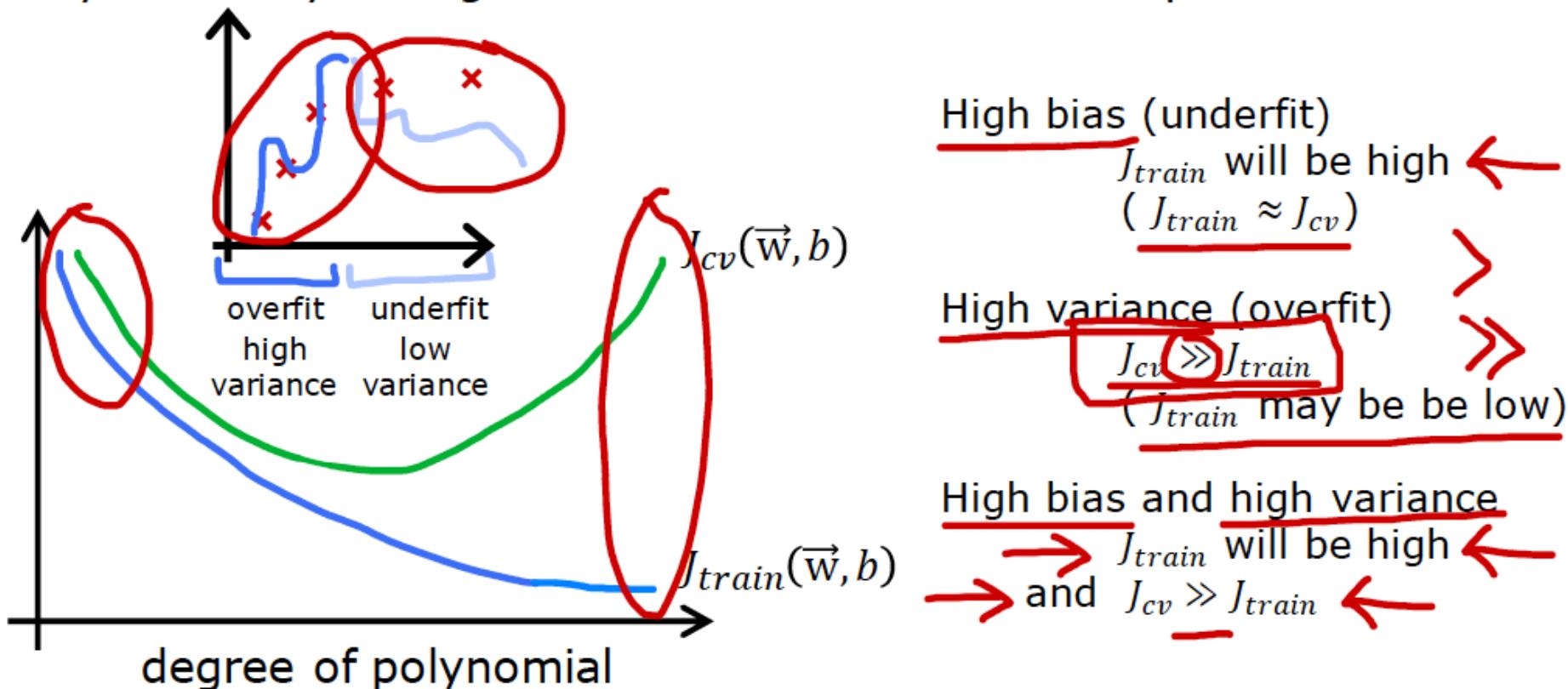
High variance  
(overfit)

$d = 4$   $J_{train}$  is low  
 $J_{cv}$  is high

# Bias and Variance

## Diagnosing bias and variance

How do you tell if your algorithm has a bias or variance problem?

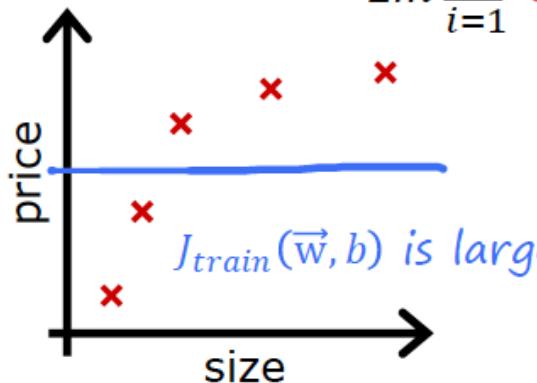


# Bias and Variance

## Linear regression with regularization

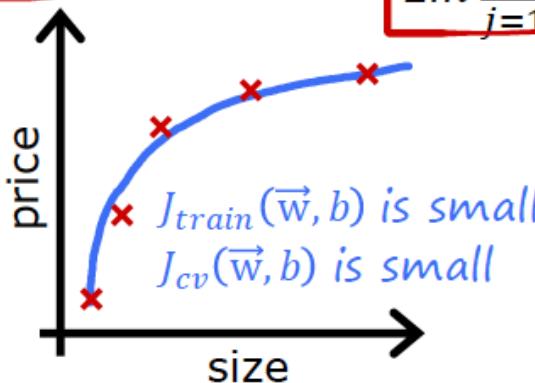
Model:  $f_{\vec{w}, b}(x) = \underbrace{w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b}_m$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



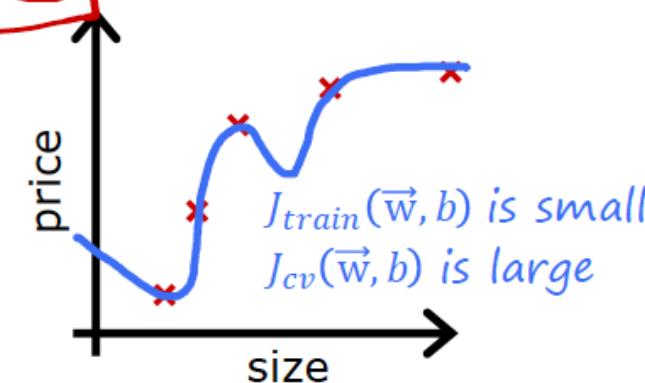
Large  $\lambda$   
High bias (underfit)

$\lambda = 10,000$   $w_1 \approx 0, w_2 \approx 0$   
 $f_{\vec{w}, b}(\vec{x}) \approx b$



Intermediate  $\lambda$

$\lambda$



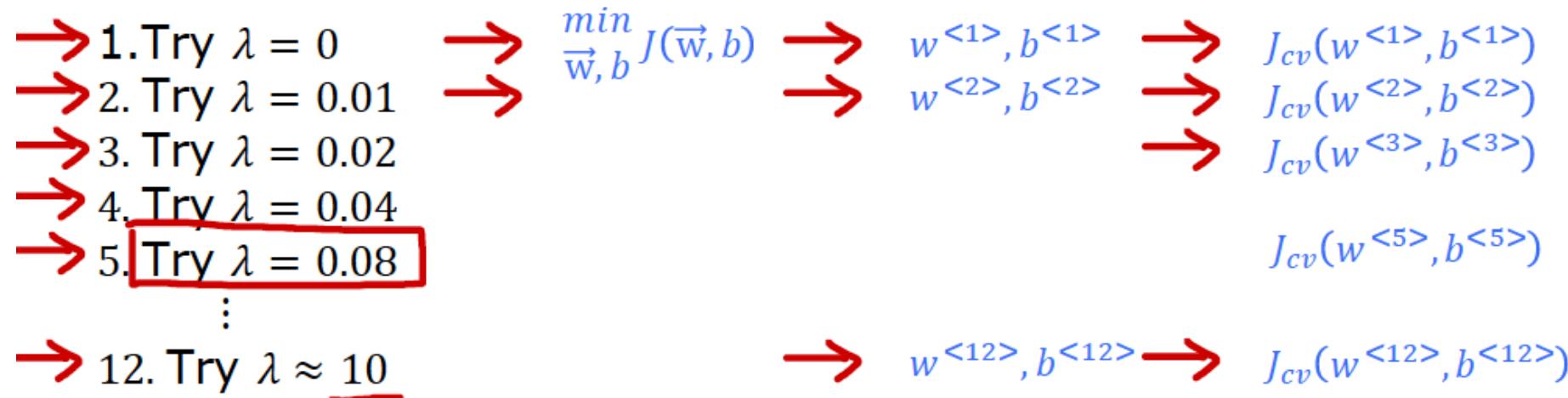
Small  $\lambda$   
High variance (overfit)

$\lambda = 0$

# Bias and Variance

## Choosing the regularization parameter $\lambda$

Model:  $f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$



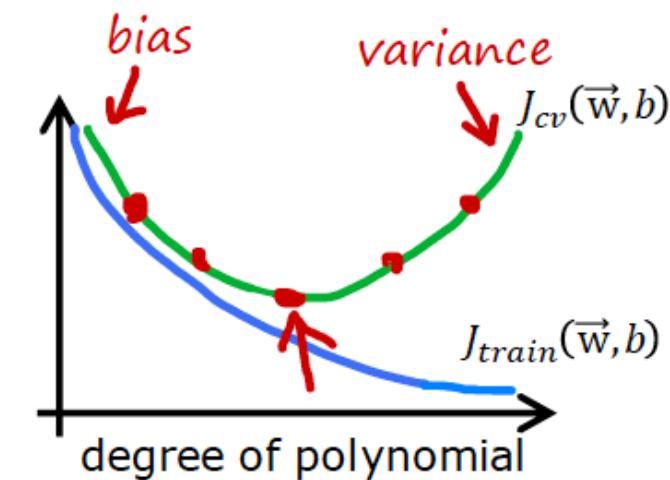
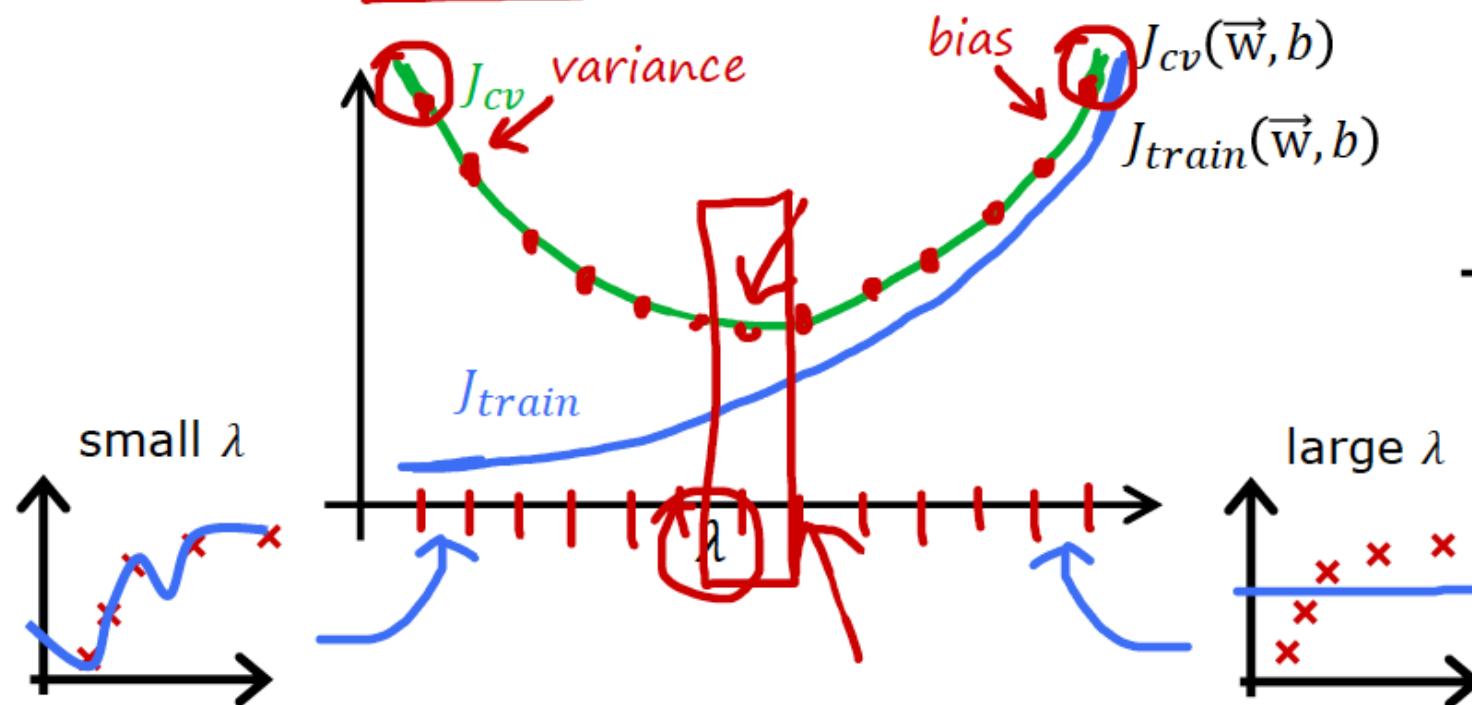
Pick  $w^{<5>}, b^{<5>}$

Report test error:  $J_{test}(w^{<5>}, b^{<5>})$

# Bias and Variance

Bias and variance as a function of regularization parameter  $\lambda$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



## Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

$$J(\vec{w}, b) = \underbrace{\frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2}_{\text{large errors}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n w_j^2}_{\text{large errors}}$$

But it makes unacceptably large errors in predictions. What do you try next?

- Get more training examples      fixes high variance
- Try smaller sets of features  $x, x^2, \cancel{x^3}, \cancel{x^4}, \cancel{x^5} \dots$       fixes high variance
- Try getting additional features      fixes high bias
- Try adding polynomial features  $(x_1^2, x_2^2, x_1 x_2, \text{etc})$       fixes high bias
- Try decreasing  $\lambda$       fixes high bias
- Try increasing  $\lambda$       fixes high variance



# Thank You!

In our next session: Classification Models



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



**Session 5(25<sup>th</sup> August,2023)  
(Friday)**

- |   |   |               |
|---|---|---------------|
| 4 | Linear Prediction Models: Linear Regression, Gradient Descent and Variants, Regularization, Bias Vs. Variance | T1: Chapter 4 |
|---|---|---------------|



# Regression



# Least Squares Based Solution

- 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  $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

# Least Squares Based Solution

- Consider our model for  $n$  instances:

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

- Let

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

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

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

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

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

# Least Squares Based Solution

- For the linear regression cost function:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \sum_{i=1}^n \left( \boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top}_{\mathbb{R}^{1 \times n}} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\mathbb{R}^{n \times 1}} \end{aligned}$$

Let:

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

# Least Squares Based Solution

- Solve for optimal  $\theta$  analytically
  - Notice that the solution is when  $\frac{\partial}{\partial \theta} J(\theta) = 0$
- Derivation:

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

1 x 1

Take derivative and set equal to 0, then solve for  $\theta$ :

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

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

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

Closed Form Solution:

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

# Linear regression



# Linear regression



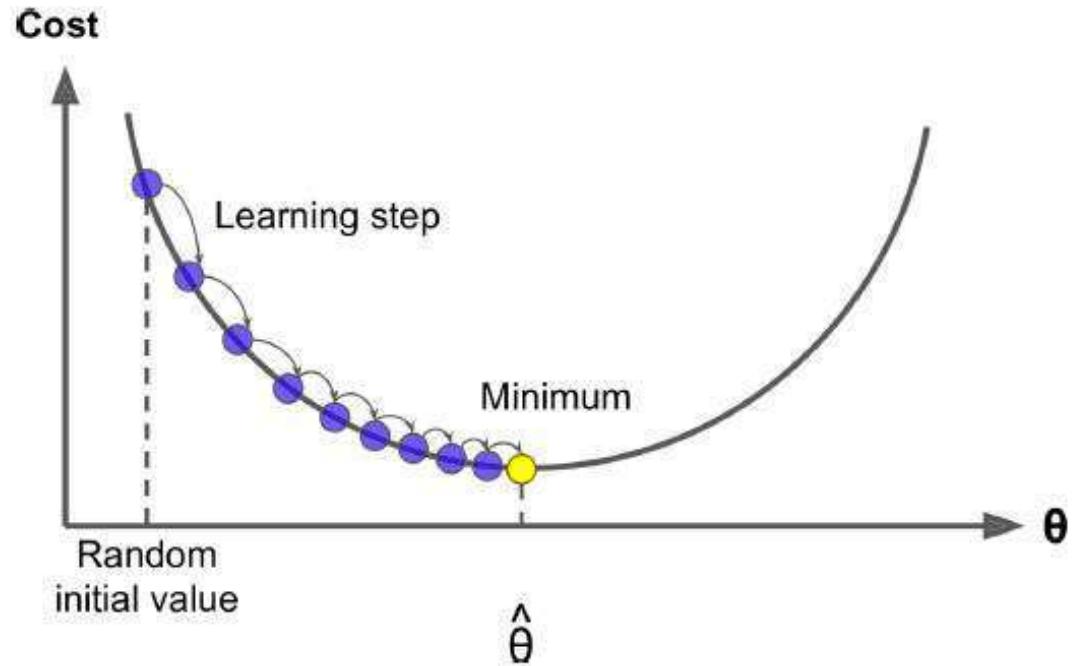
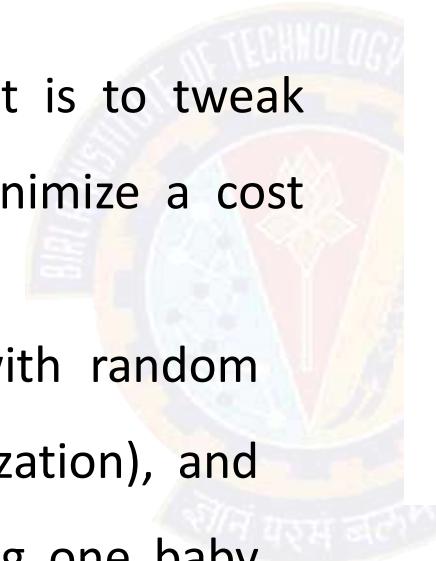
# Linear regression





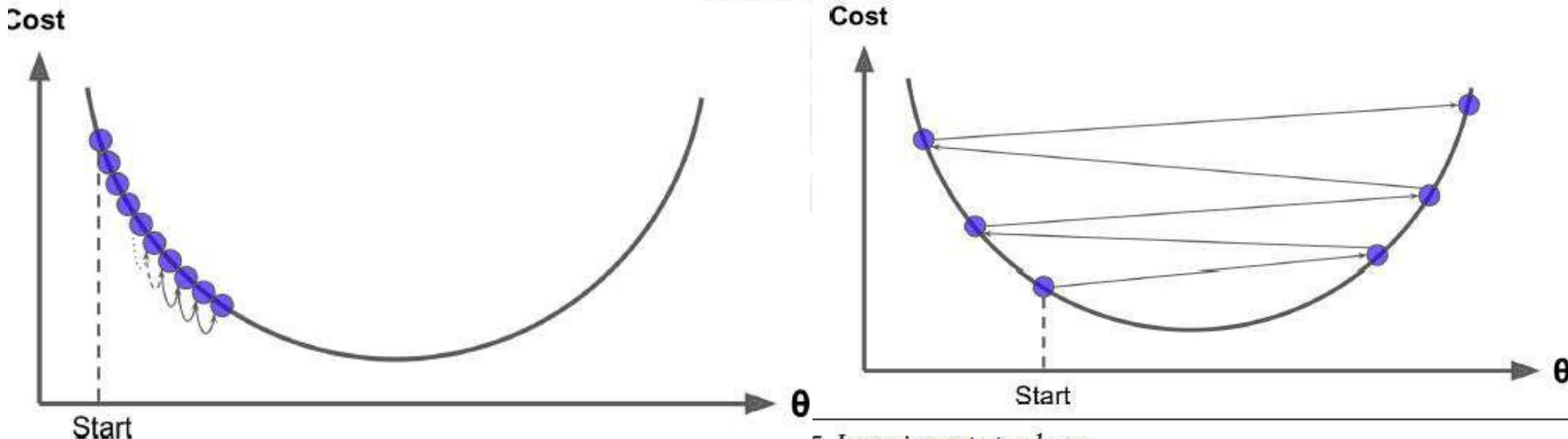
# Gradient Descent

- Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function.
- Concretely, you start by filling  $\theta$  with random values (this is called random initialization), and then you improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum



# Gradient Descent- Hyper parameter

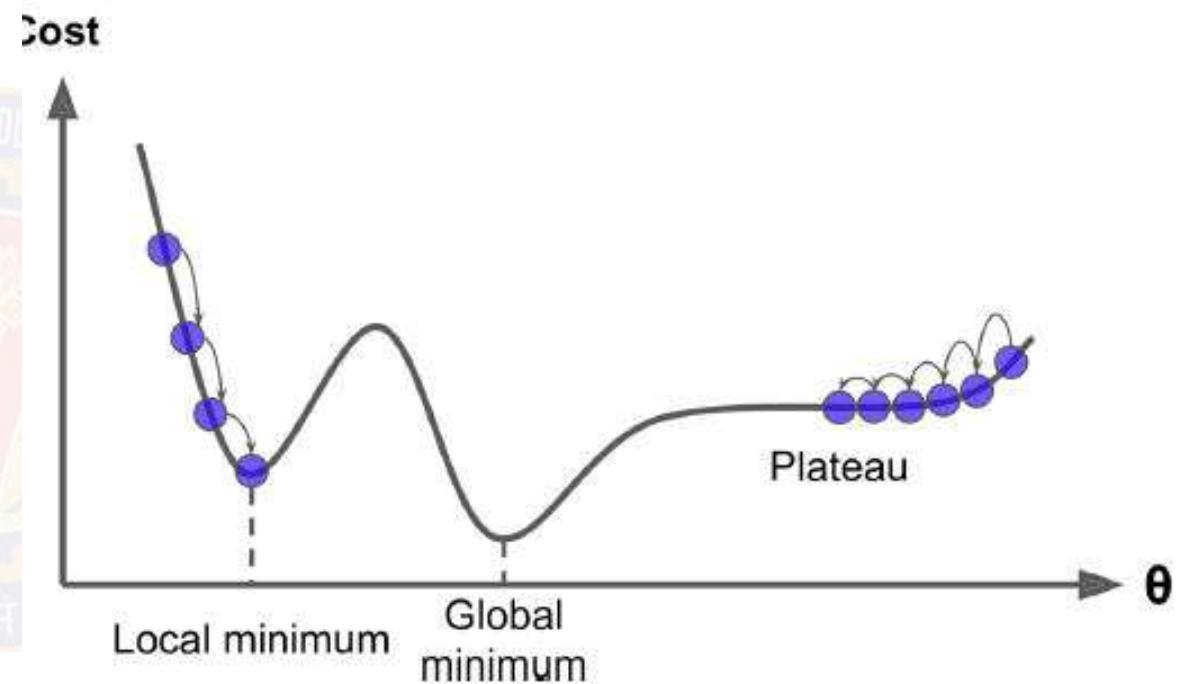
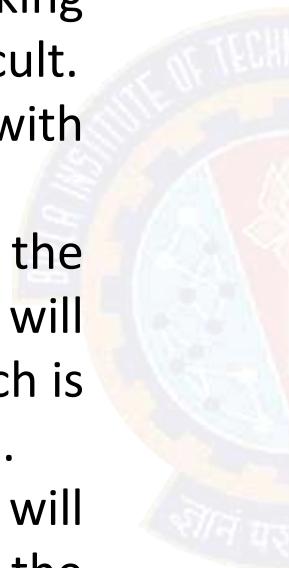
If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time



On the other hand, if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before. This might make the algorithm diverge, with larger and larger values, failing to find a good solution

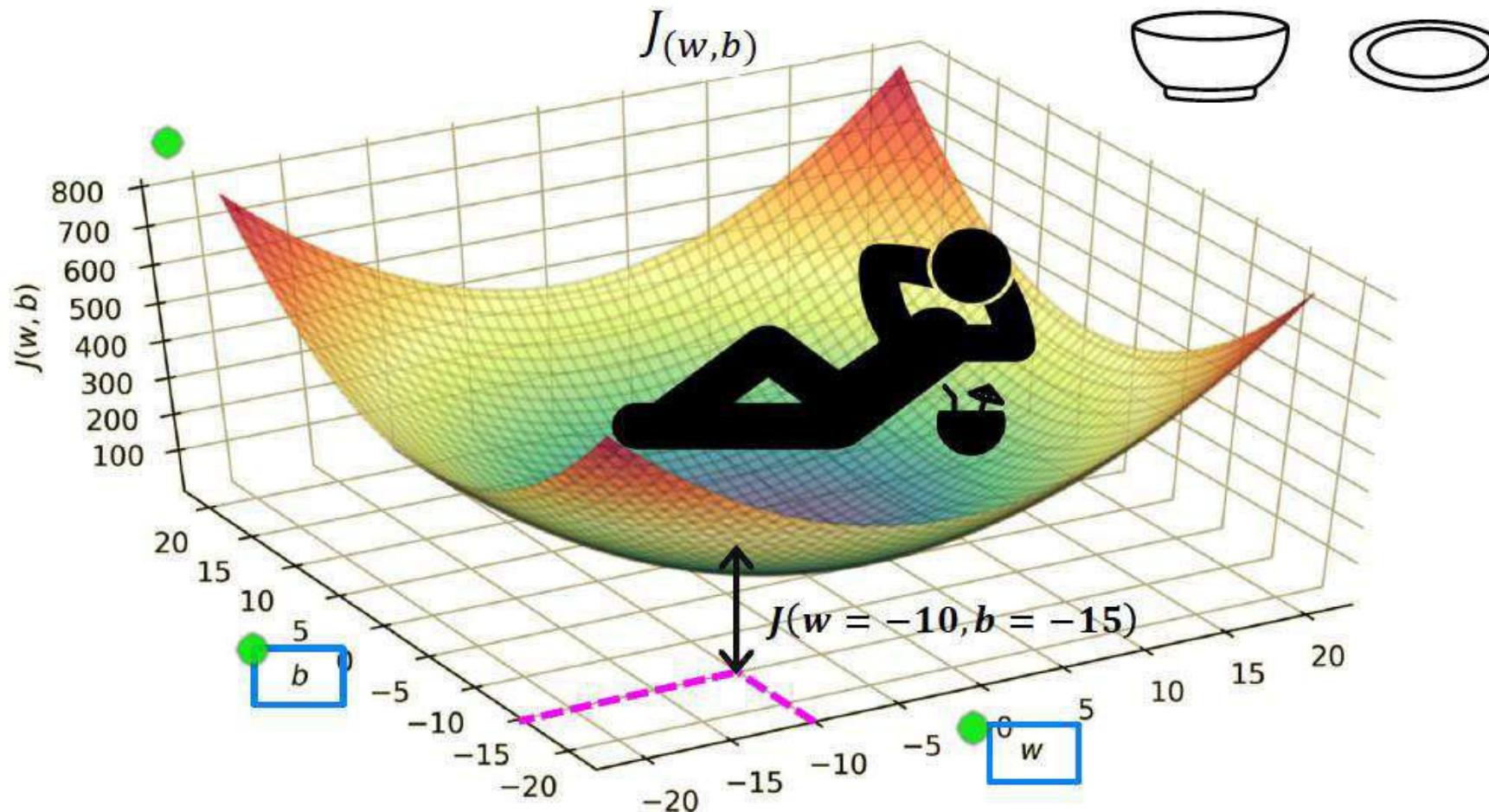
# Gradient Descent- Pitfalls

- Finally, not all cost functions look like nice regular bowls.
- There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult.
- **Figure** shows the two main challenges with Gradient Descent:
  - if the random initialization starts the algorithm on the left, then it will converge to a local minimum , which is not as good as the global minimum.
  - If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.



*Gradient Descent pitfalls*

# Intuition Behind Cost Function



# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

For Linear Regression:

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

..

# Gradient Descent for Linear Regression

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous  
update  
for  $j = 0 \dots d$

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

$L_2$  norm:  $\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$

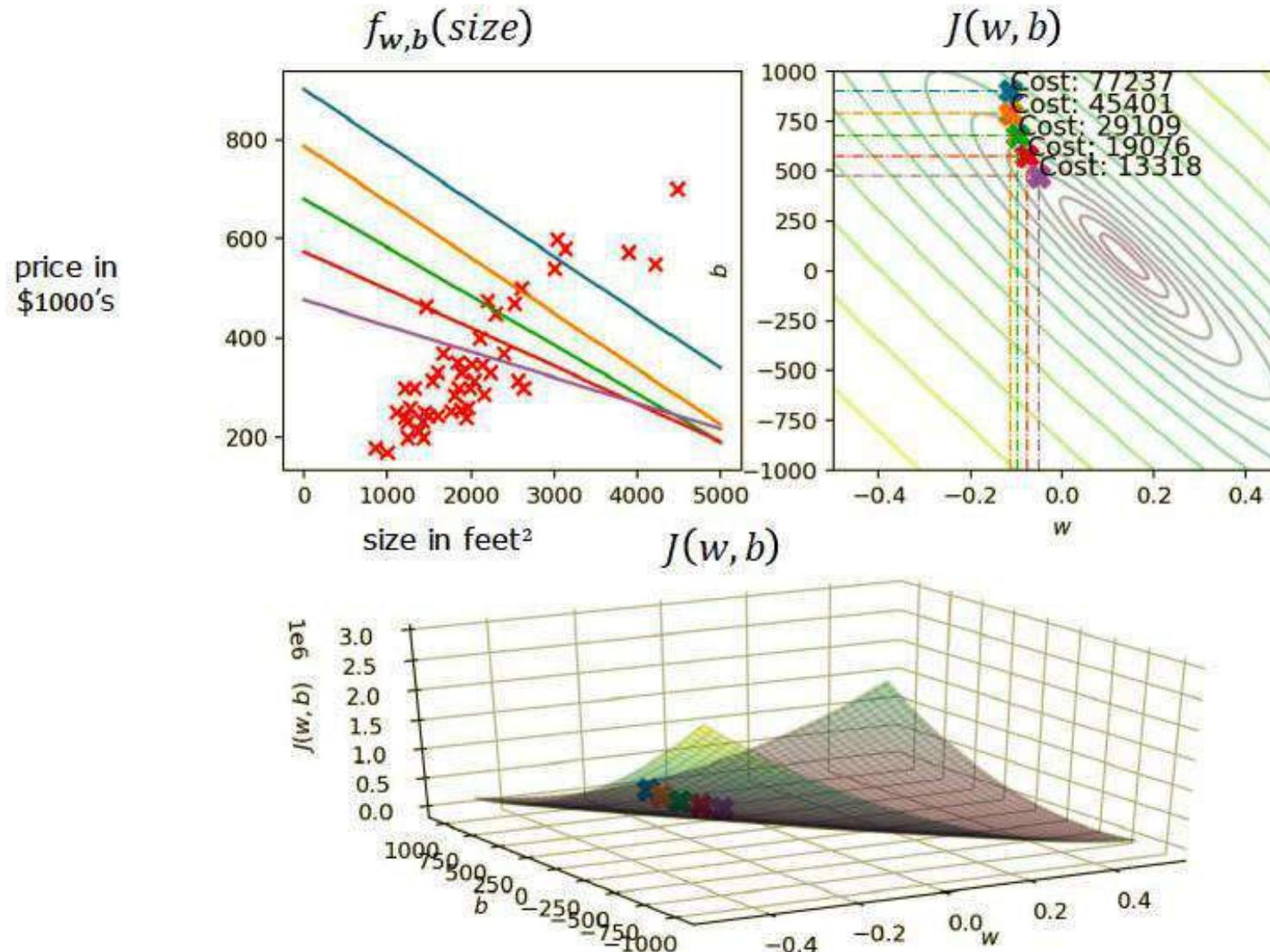






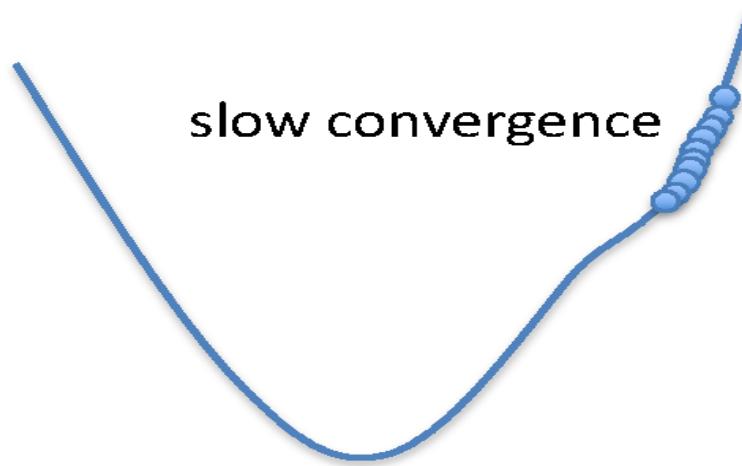


# Running Gradient Descent

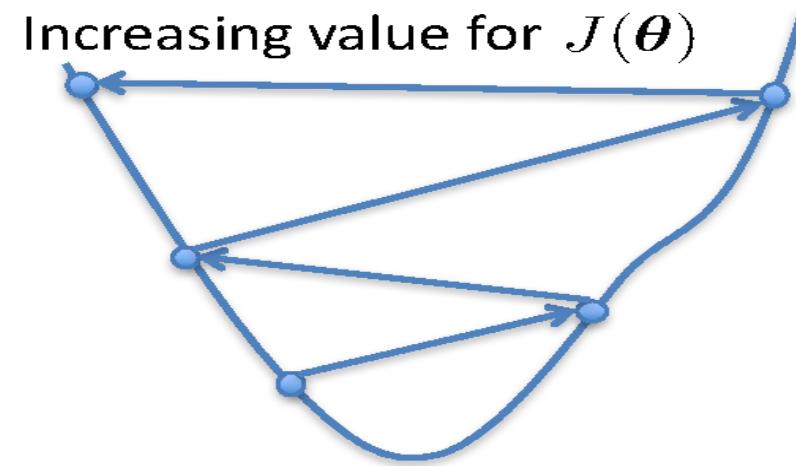


# Choosing Step Size

$\alpha$  too small



$\alpha$  too large

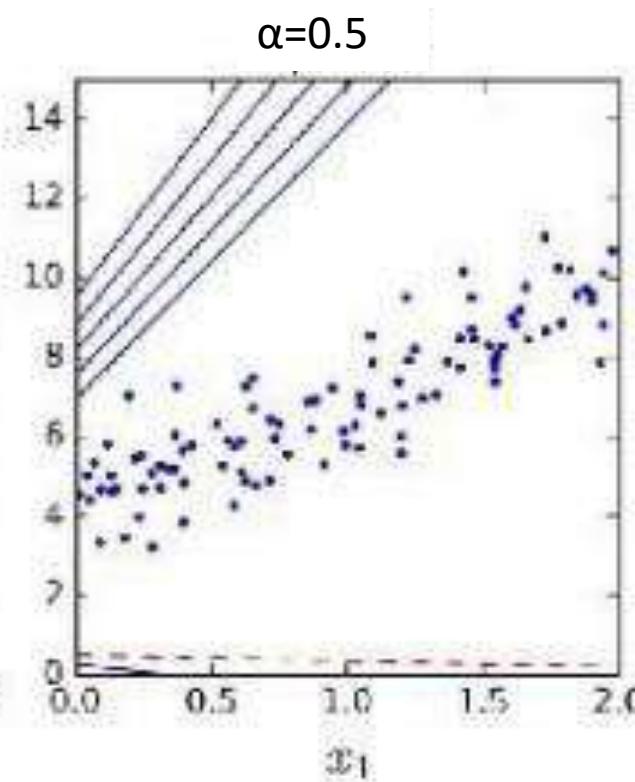
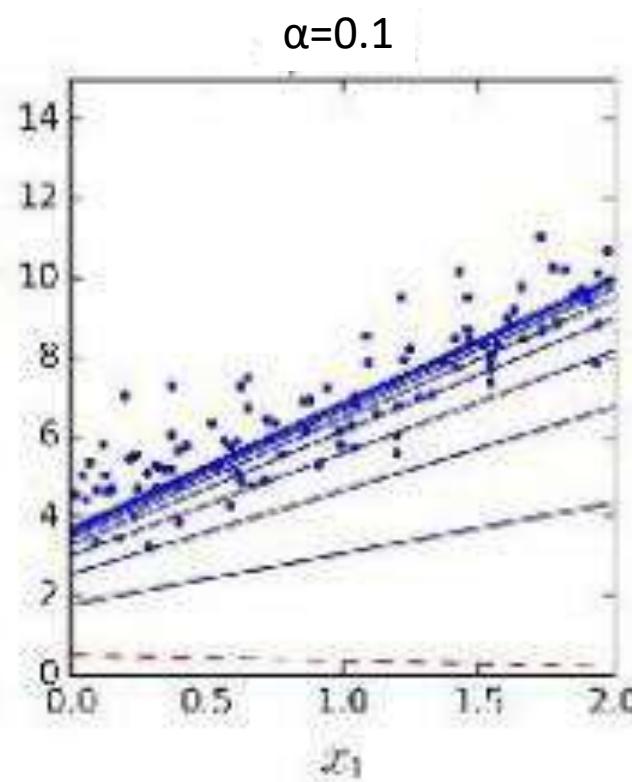
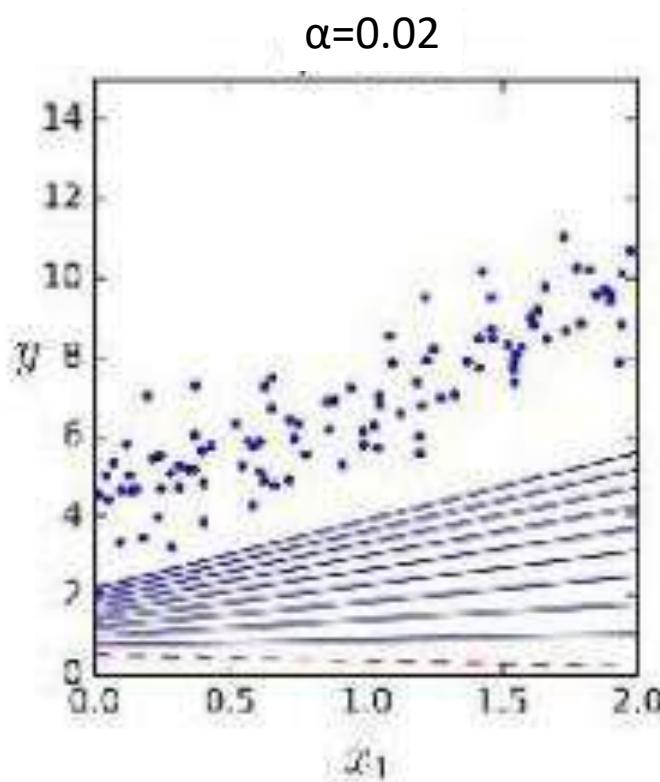


- May overshoot the minimum
- May fail to converge
- May even diverge

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  $\alpha$

# Impact of Learning Rate



# Mini-batch Gradient Descent Algorithm



- ❖ GD can be very slow on large datasets
- ❖ So in such case SGD can be used
- ❖ In SGD, the procedure is same as GD but the difference is updatation of coefficients is done for each training example rather than at the end of training
- ❖ i. e. cost is not summed over all training examples , but instead for one training example

# Stochastic Gradient Descent Algorithm



# Mini Batch Stochastic Gradient Descent Algorithm(MBSGD)









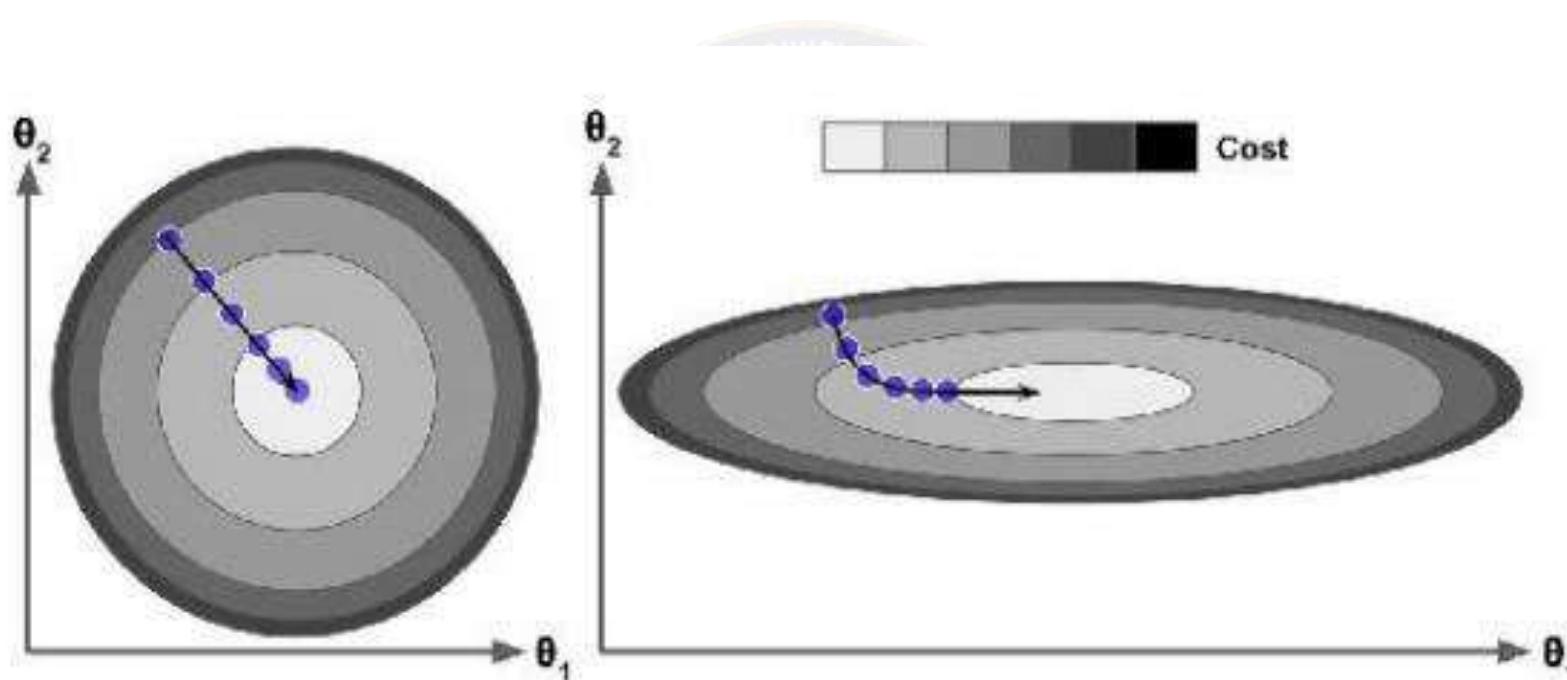
# Summary

- ❖ Optimization is a big part of machine learning.
- ❖ Gradient descent is a simple optimization procedure that you can use with many machine learning algorithms.
- ❖ 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

# Feature Normalization

**Gradient Descent can be slow when feature scales are widely different**

- Normalization of feature values for same variance needed for faster convergence

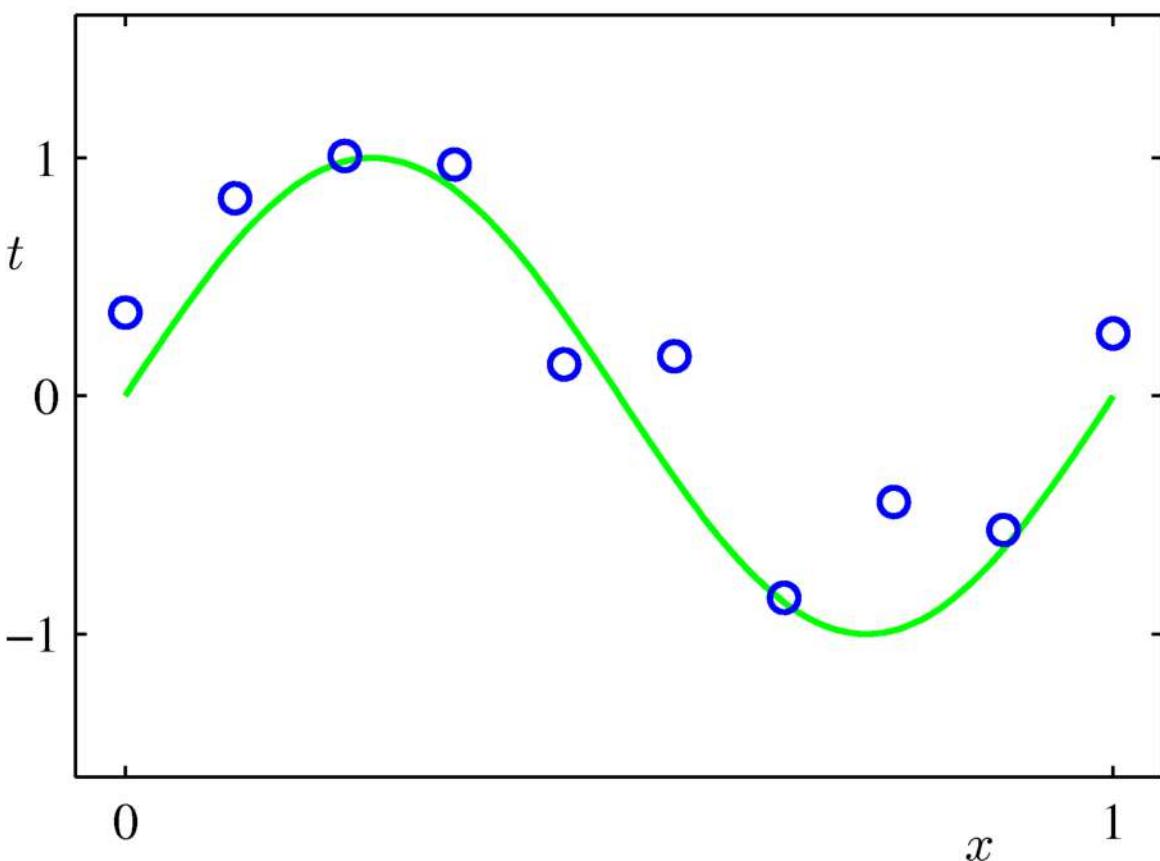


# Extending to More Complex Model

- 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:  $x_3 = x_1 \cdot x_2$

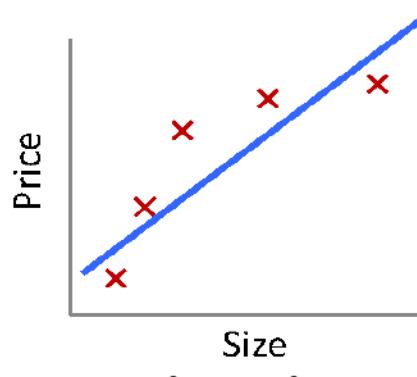
This allows use of linear regression techniques  
to fit non-linear datasets.

# Fitting a Polynomial Curve

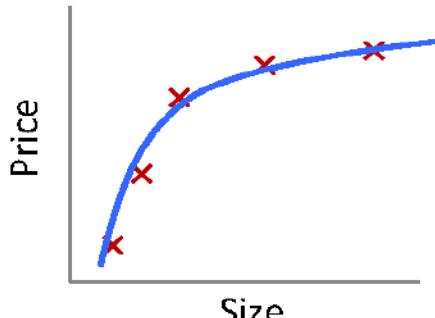


$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j = \sum_{j=0}^p \theta_j z_j$$
$$z_j = x^j$$

# Quality of Fit

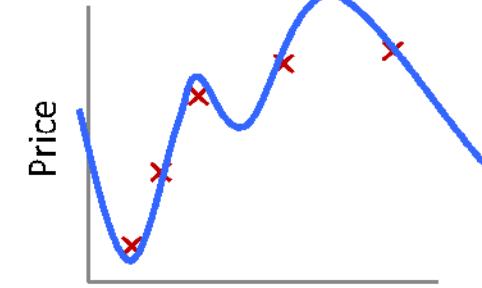


Underfitting  
(high bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$

Correct fit



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

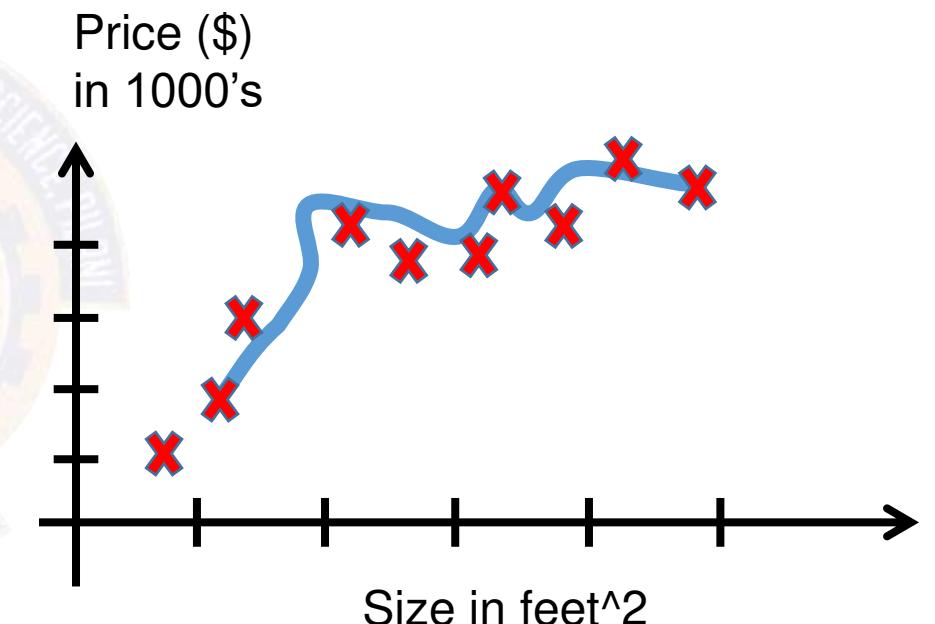
Overfitting  
(high variance)

## Overfitting:

- The learned hypothesis may fit the training set very well ( $J(\theta) \approx 0$ )
- ...but fails to generalize to new examples

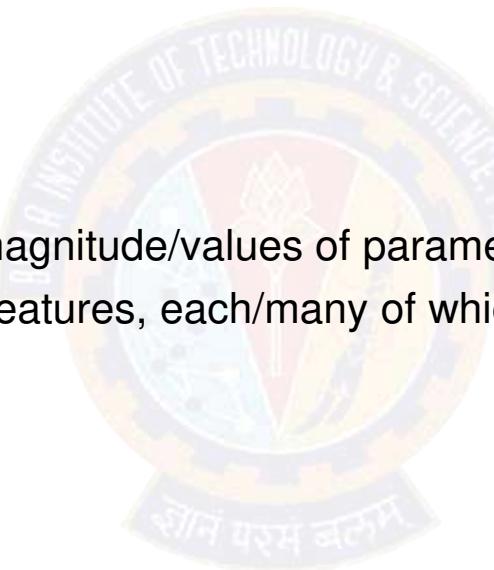
# Addressing overfitting

- $x_1$  = 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
- :
- $x_{100}$



# Addressing overfitting

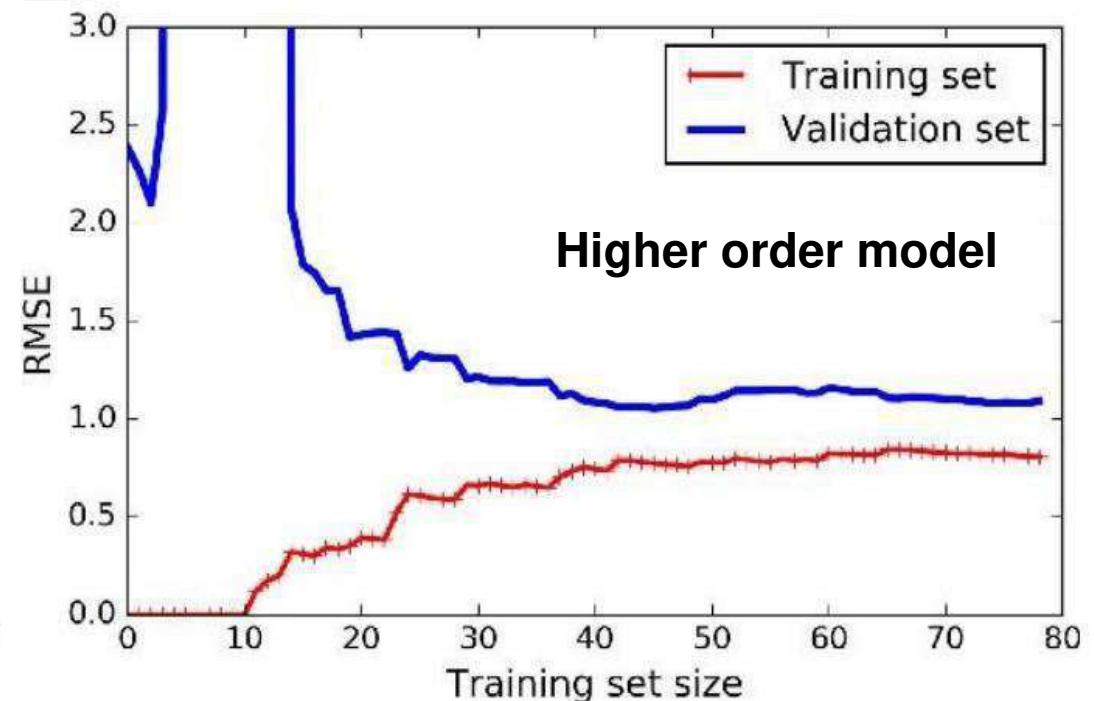
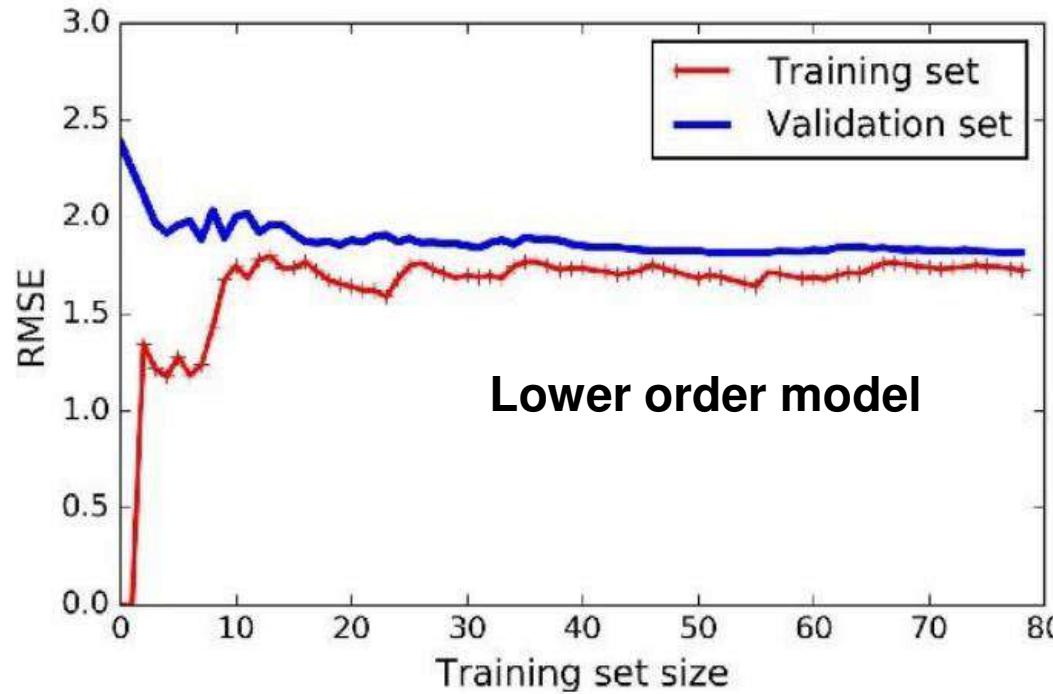
- Reduce number of features.
  - Manually select which features to keep.
  - Model selection algorithm
- Regularization.
  - Keep all the features, but reduce magnitude/values of parameters  $\theta_j$ .
  - Works well when we have a lot of features, each/many of which contributes a bit to predicting  $y$ .



# Effect of Training Size on Overfitting

Size of training dataset needs to be large to prevent overfitting

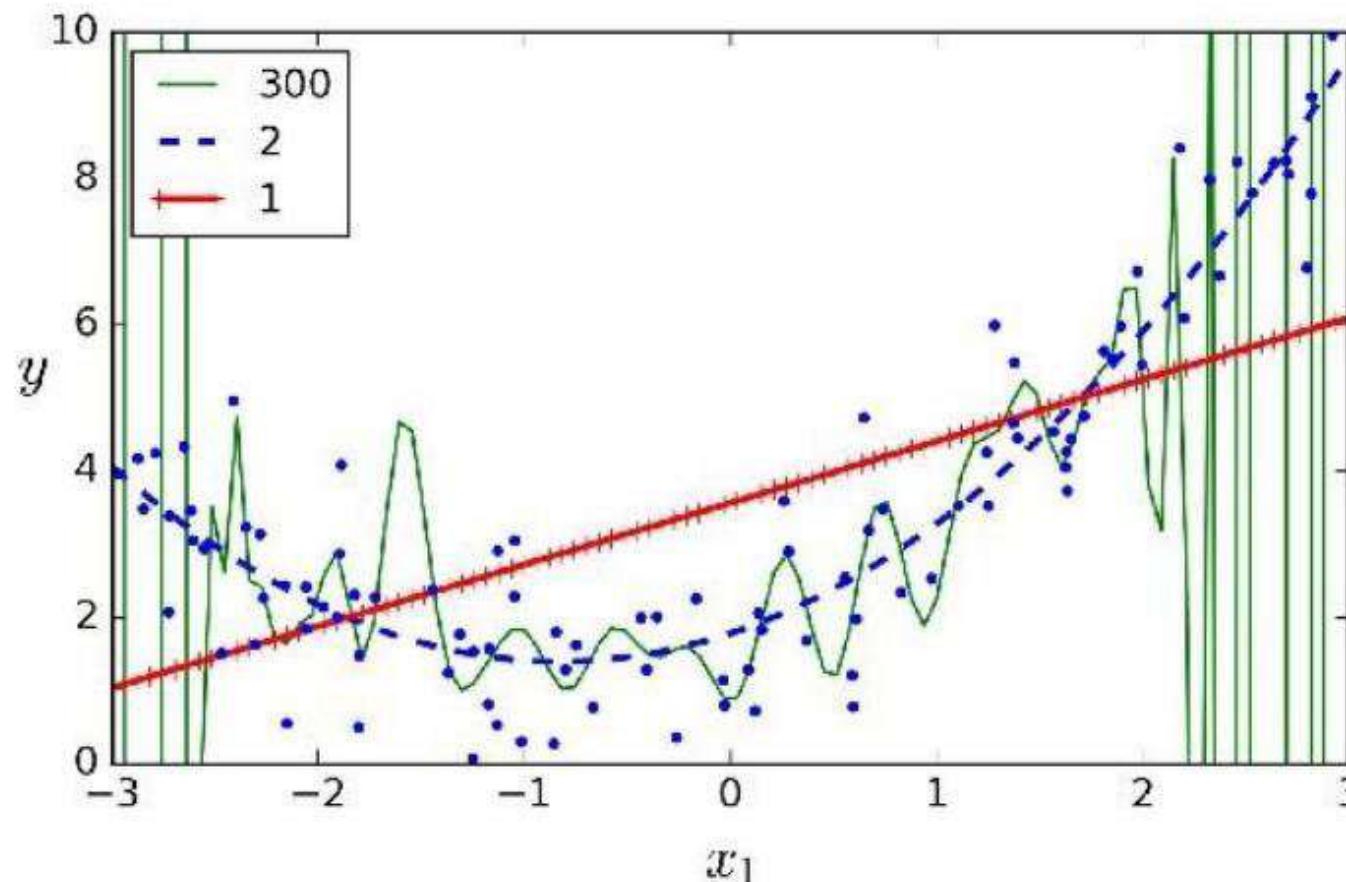
- when higher order model is used.



# Polynomial Fitting can lead to Overfitting

Underlying target function is quadratic

- Linear model results in under fitting with large bias
- Polynomial of order 300 results in a large variance

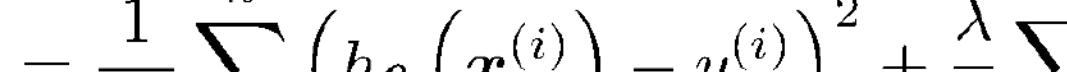


# Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of  $\theta_j$ 
  - 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)

# Ridge Regularization

- Linear regression objective function

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


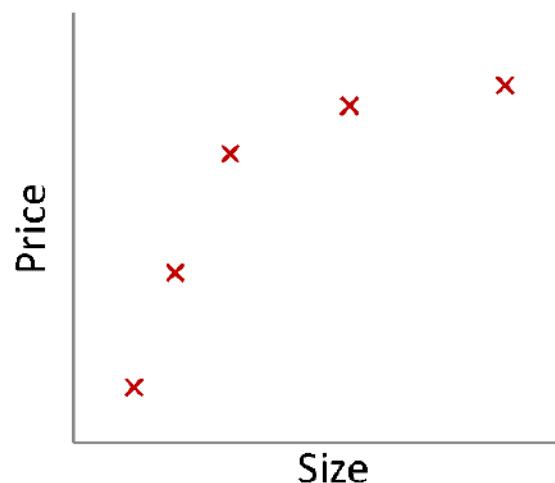
The diagram illustrates the cost function  $J(\boldsymbol{\theta})$  as a sum of two terms. The first term,  $\frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$ , is represented by a blue bracket below the equation and labeled 'model fit to data'. The second term,  $\frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$ , is represented by another blue bracket to the right of the first and labeled 'regularization'.

- $\lambda$  is the regularization parameter ( $\lambda \geq 0$ )
  - No regularization on  $\theta_0$ !

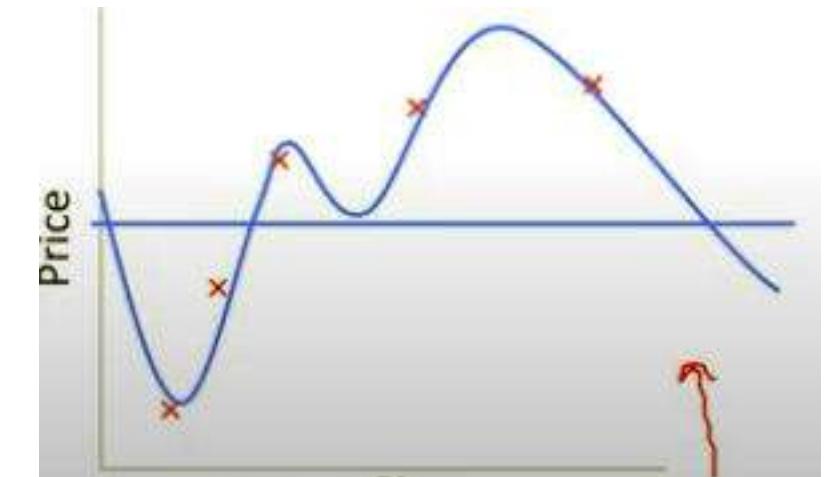
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



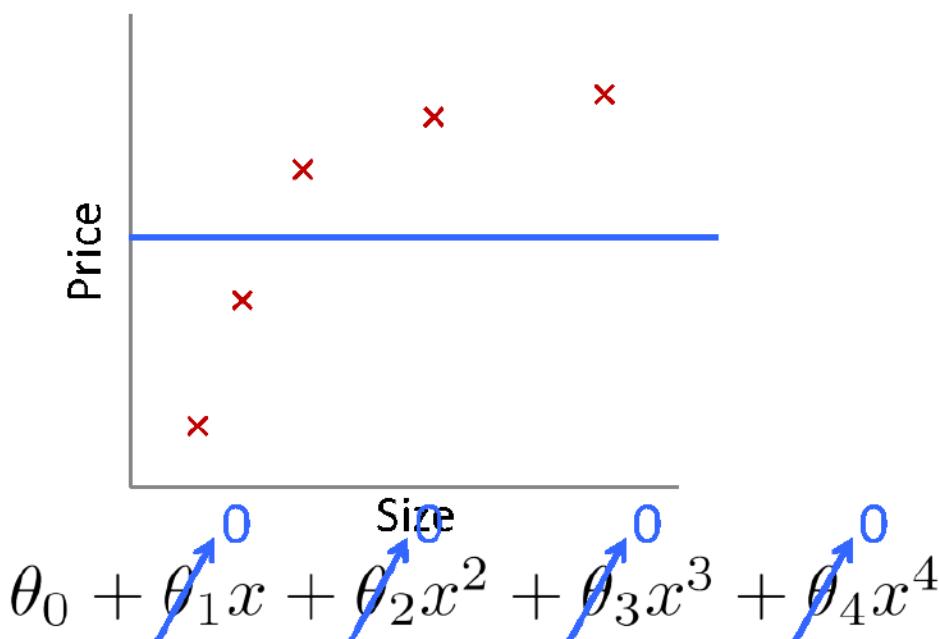
Overfitting

Based on example by Andrew Ng

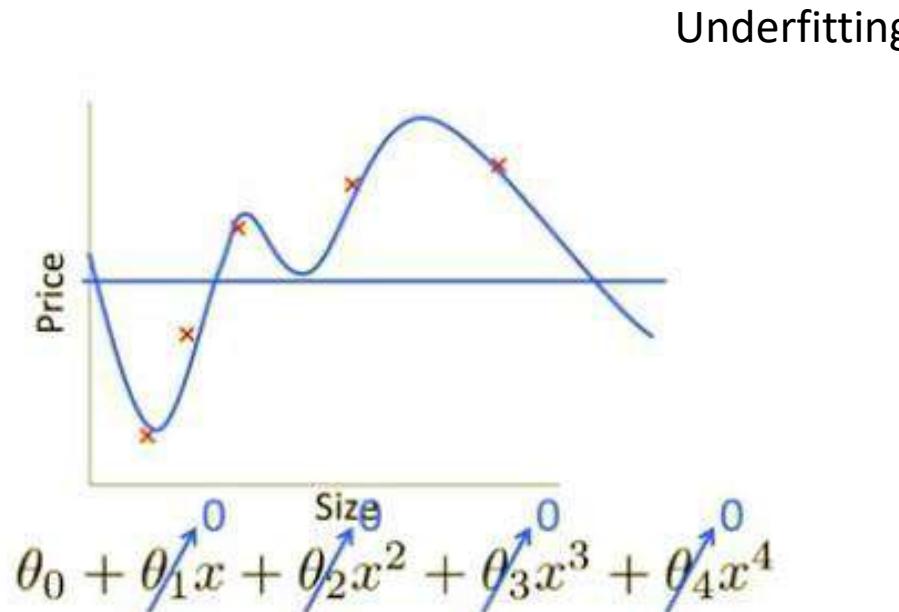
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



Based on example by Andrew Ng



# Ridge regression

## 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:

$$\frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta})$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

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

regularization 57

# Ridge Regression

Further simplified

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

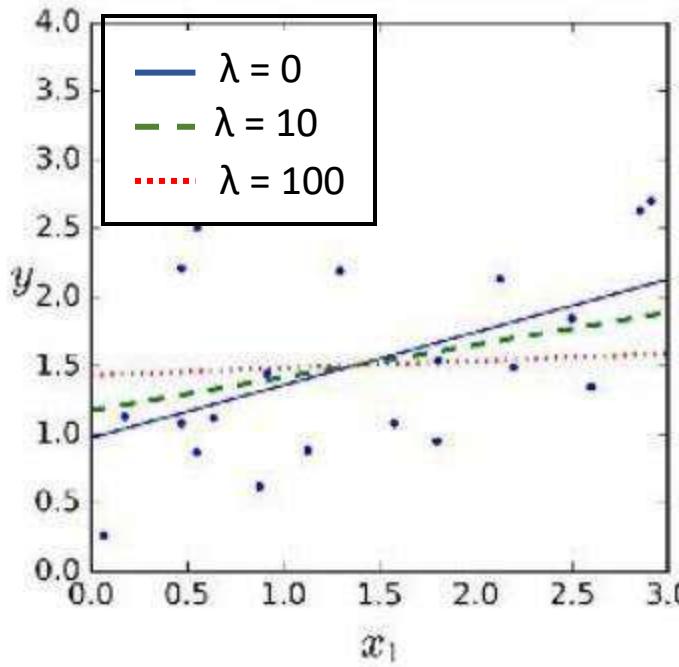
$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

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

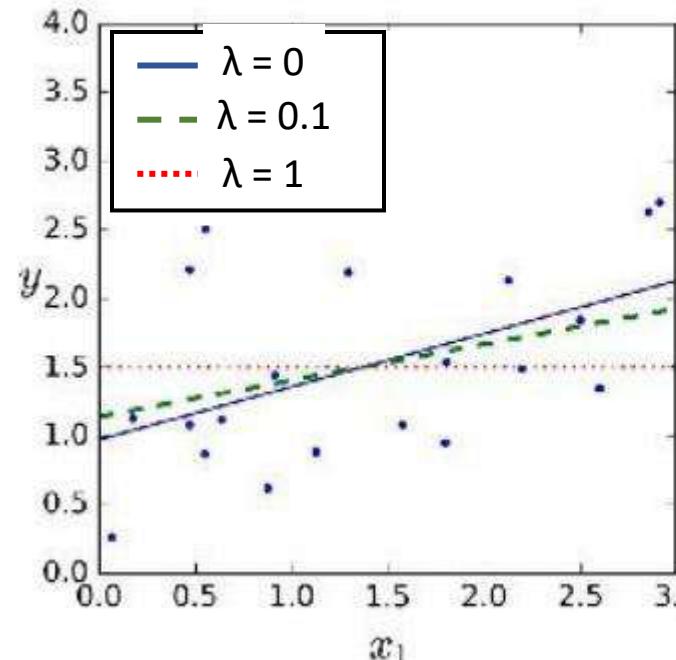
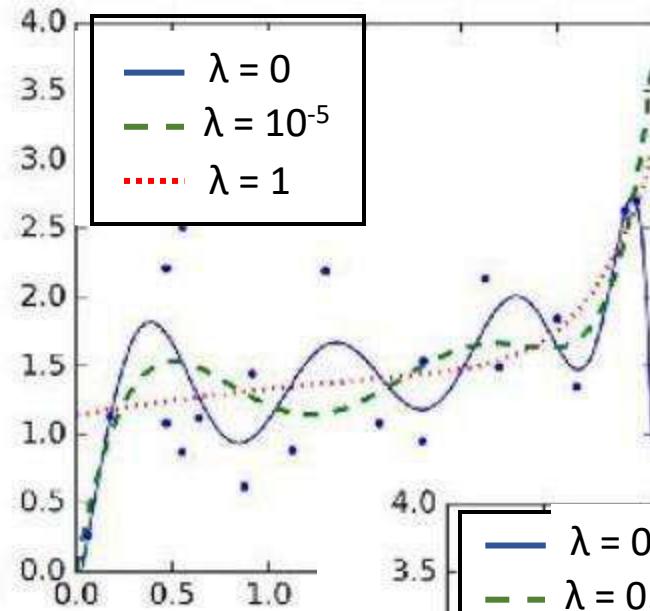
- We can rewrite the gradient step as:

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

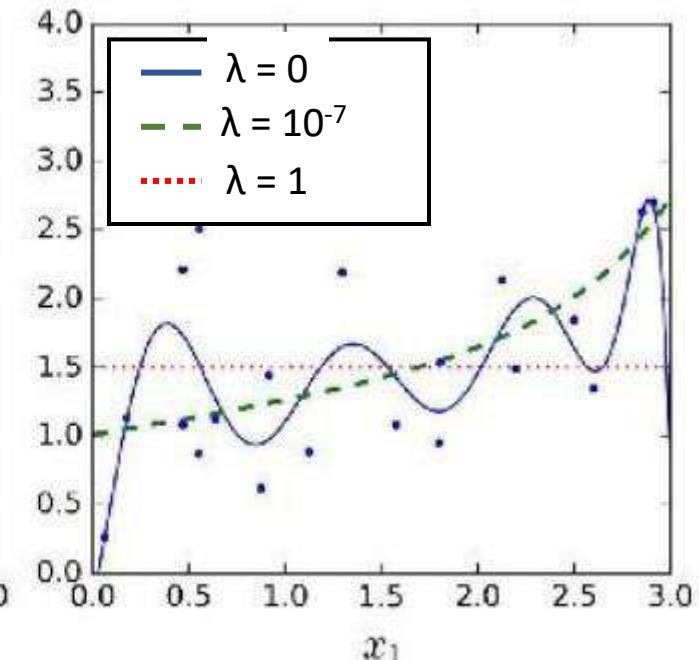
# Ridge Vs Lasso Regularization



Ridge



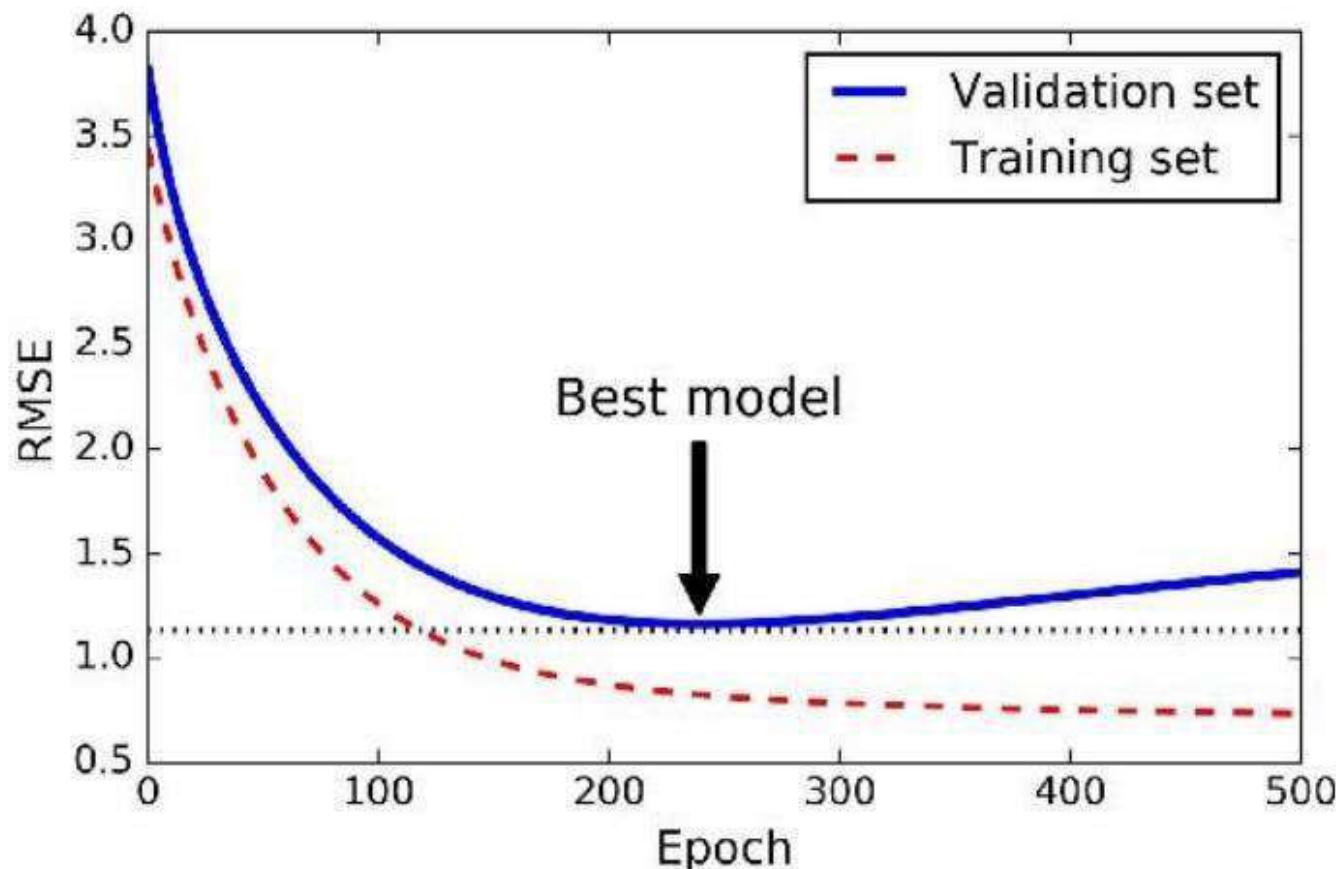
Lasso



# Early Stopping

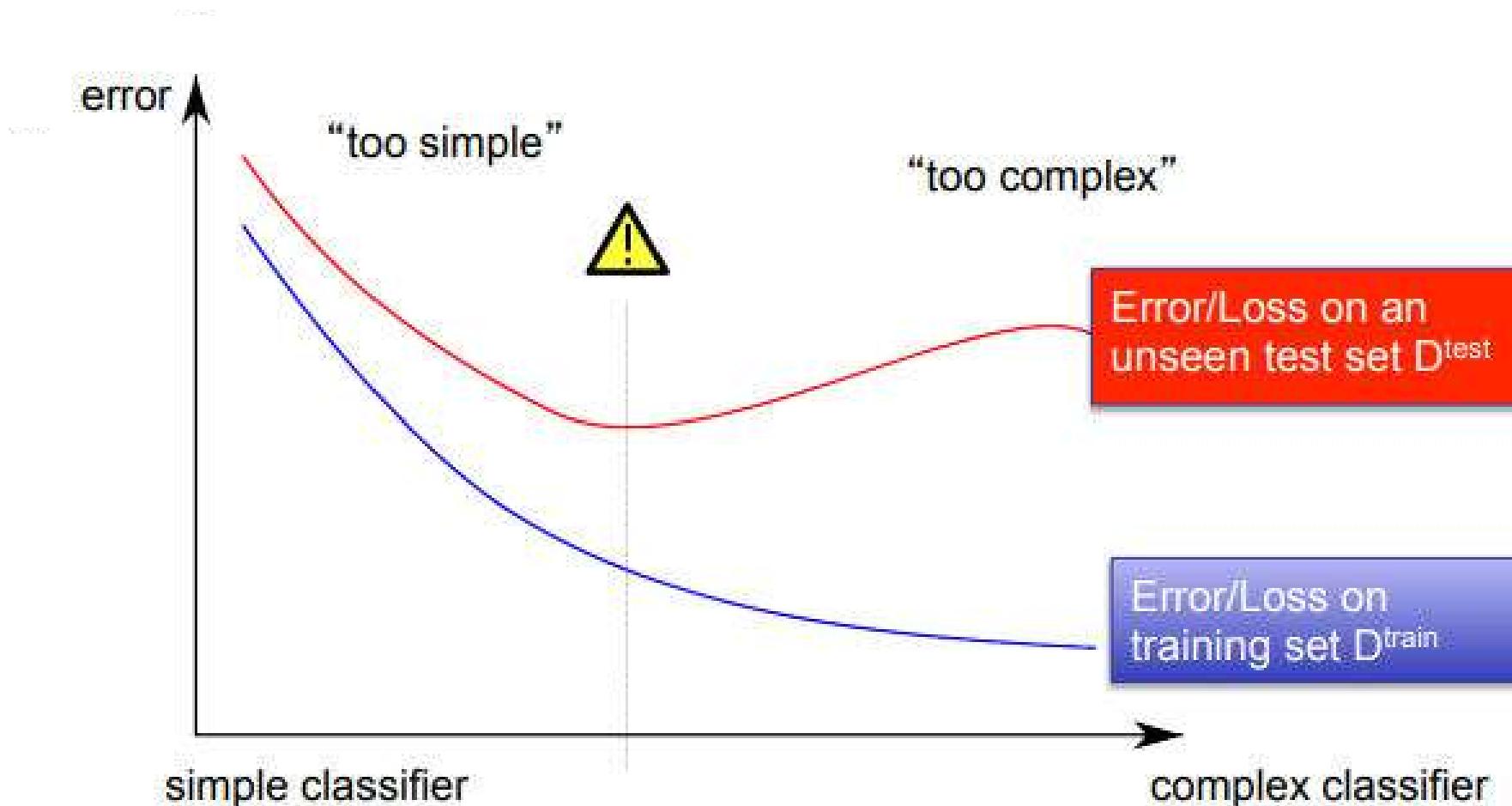
## Do Not Over train to prevent overfitting

- Stop training once error on the validation set starts showing an upward trend, even if the error on the training set keeps decreasing



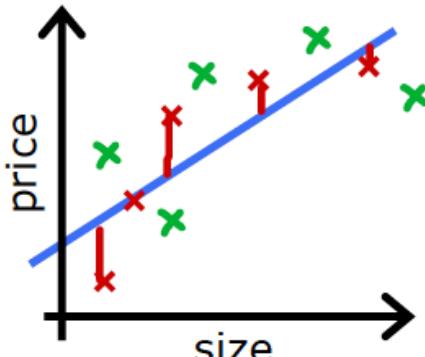
# Bias Variance Tradeoff

Bias/Variance is a Way to Understand Overfitting and Underfitting



# Bias and Variance

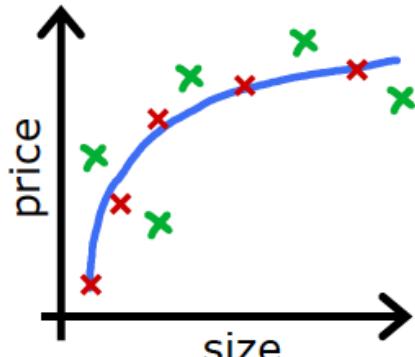
## Bias/variance



$$f_{\vec{w},b}(x) = w_1x + b$$

→ High bias  
(underfit)

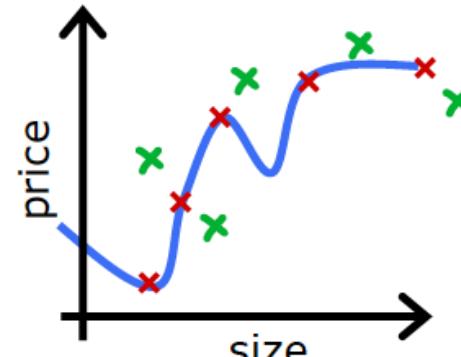
$d = 1$   $J_{train}$  is high  
 $J_{cv}$  is high



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$$

"Just right"

$d = 2$   $J_{train}$  is low  
 $J_{cv}$  is low



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$$

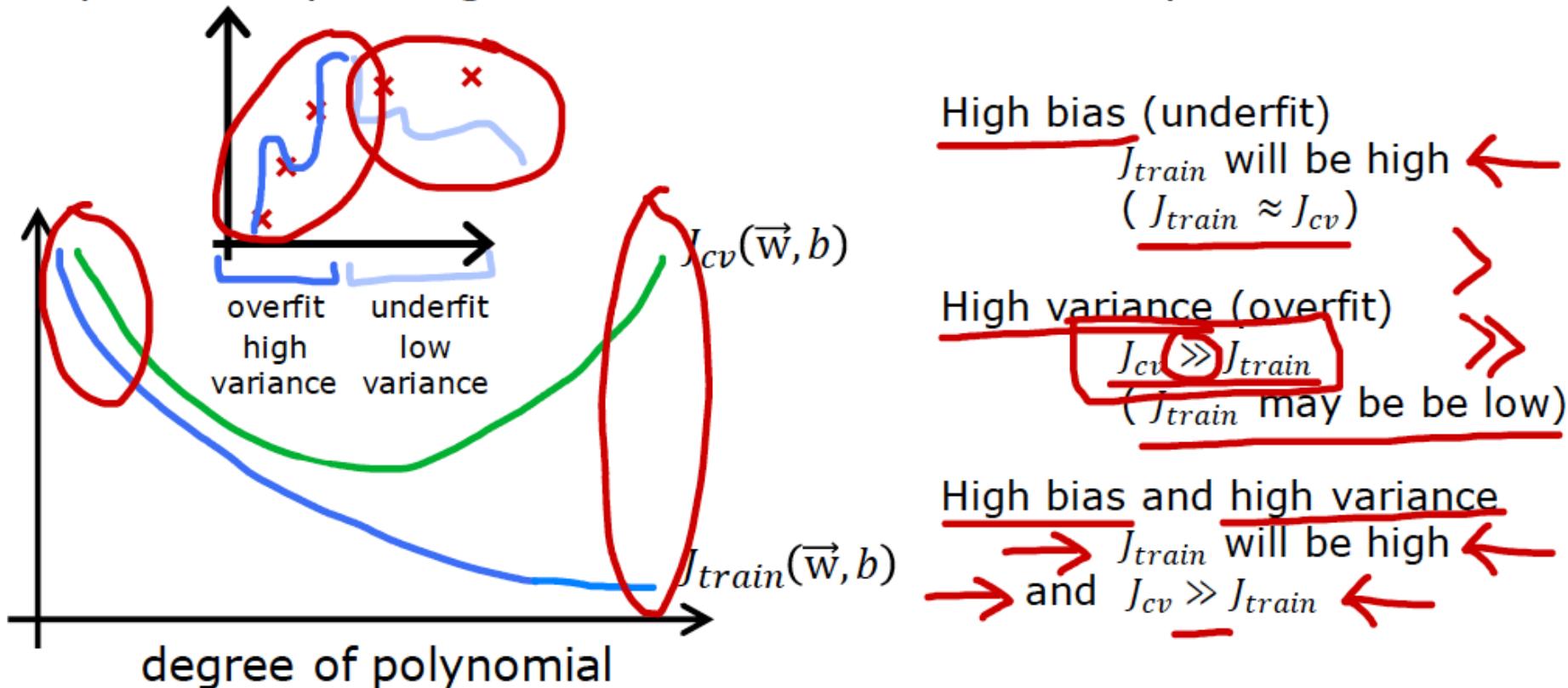
High variance  
(overfit)

$d = 4$   $J_{train}$  is low  
 $J_{cv}$  is high

# Bias and Variance

## Diagnosing bias and variance

How do you tell if your algorithm has a bias or variance problem?

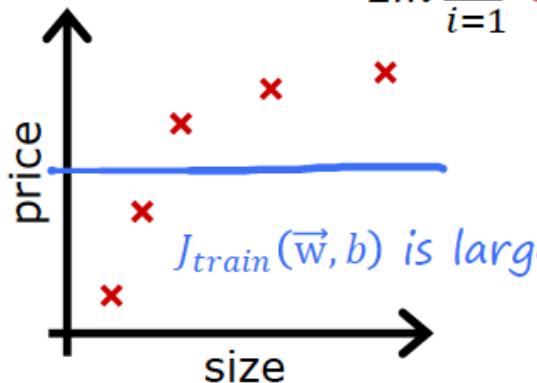


# Bias and Variance

## Linear regression with regularization

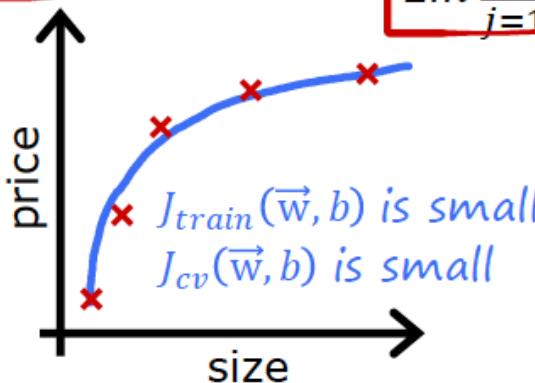
Model:  $f_{\vec{w}, b}(x) = \underbrace{w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b}_m$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



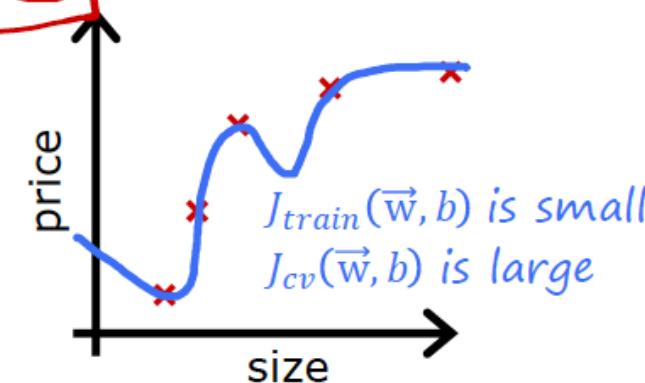
Large  $\lambda$   
High bias (underfit)

$\lambda = 10,000$   $w_1 \approx 0, w_2 \approx 0$   
 $f_{\vec{w}, b}(\vec{x}) \approx b$



Intermediate  $\lambda$

$\lambda$



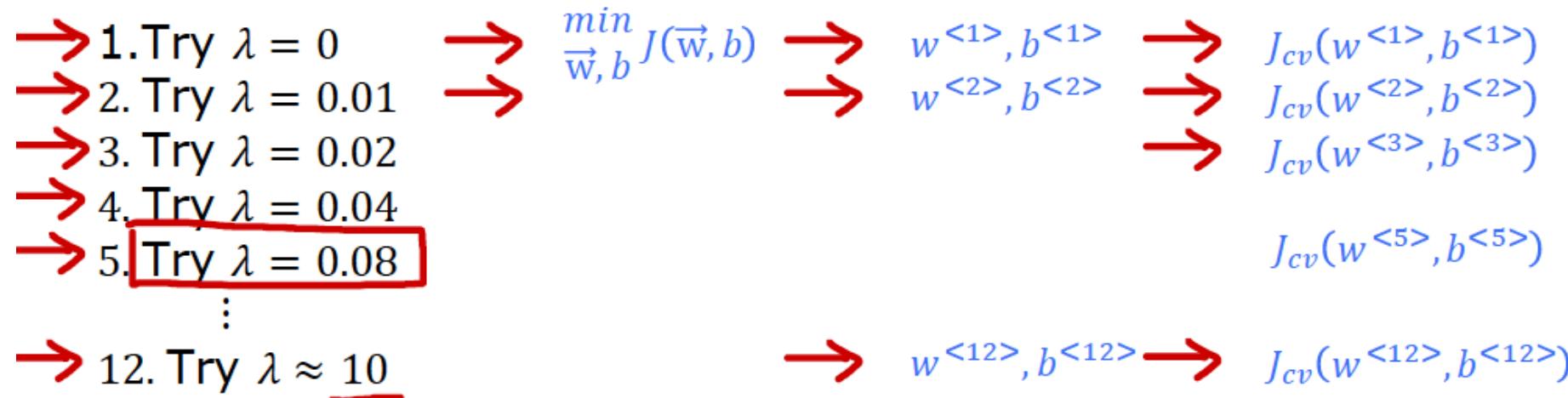
Small  $\lambda$   
High variance (overfit)

$\lambda = 0$

# Bias and Variance

## Choosing the regularization parameter $\lambda$

Model:  $f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$



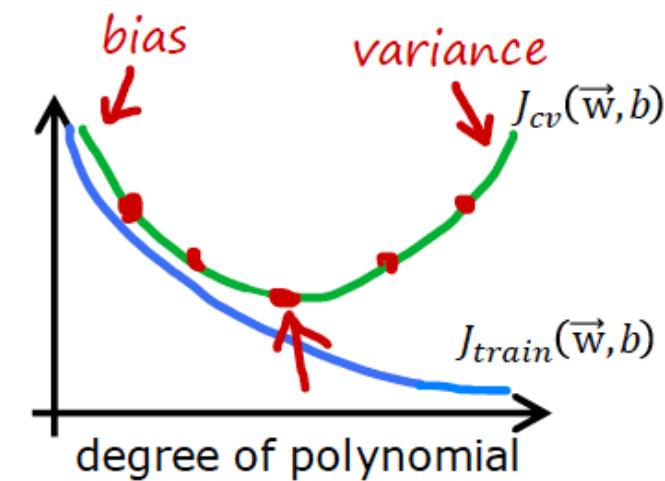
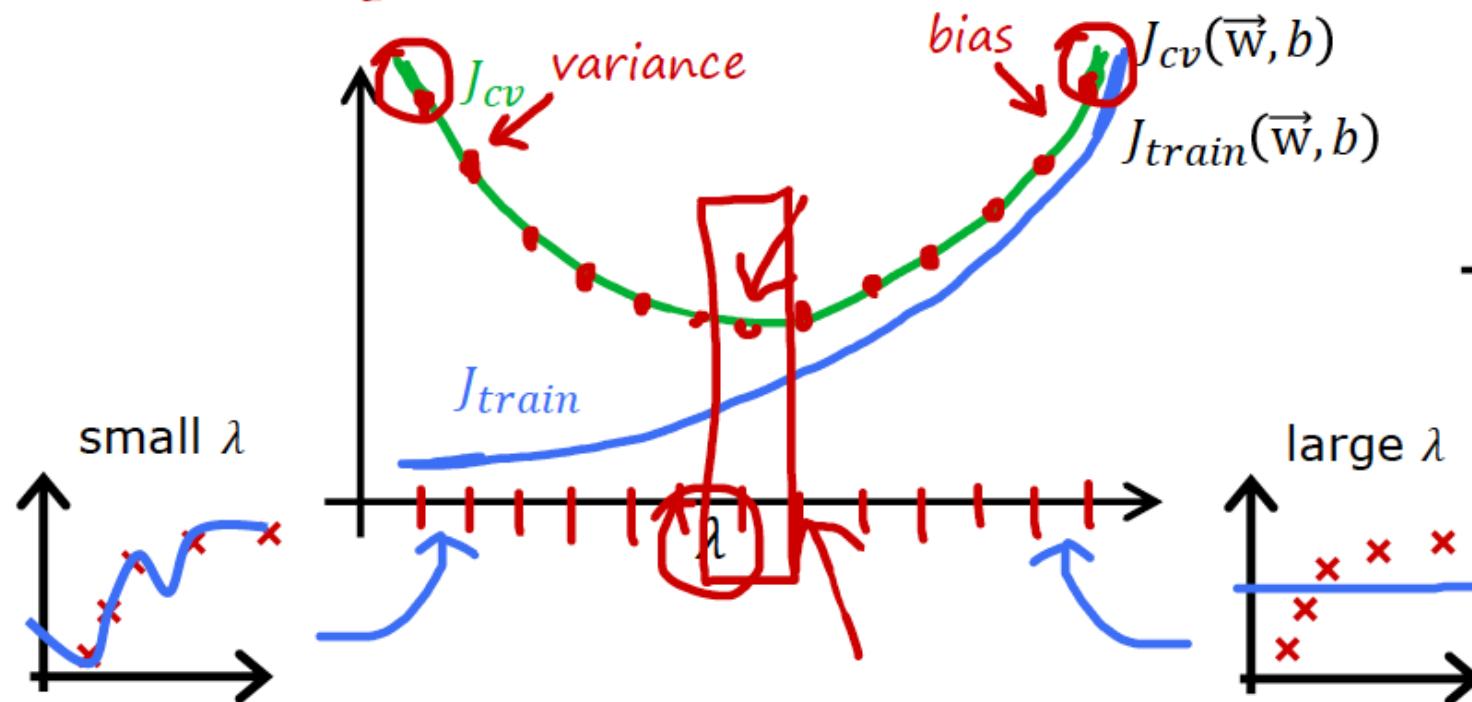
Pick  $w^{<5>}, b^{<5>}$

Report test error:  $J_{test}(w^{<5>}, b^{<5>})$

# Bias and Variance

## Bias and variance as a function of regularization parameter $\lambda$

$$J(\vec{w}, b) = \boxed{\frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2} + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



## Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

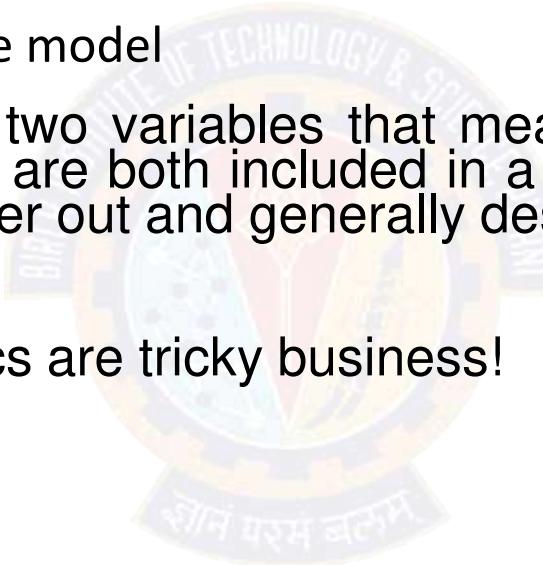
$$J(\vec{w}, b) = \underbrace{\frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2}_{\text{large errors}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n w_j^2}_{\text{large errors}}$$

But it makes unacceptably large errors in predictions. What do you try next?

- Get more training examples      fixes high variance
- Try smaller sets of features  $x, x^2, \cancel{x^3}, \cancel{x^4}, \cancel{x^5} \dots$       fixes high variance
- Try getting additional features      fixes high bias
- Try adding polynomial features  $(x_1^2, x_2^2, x_1 x_2, \text{etc})$       fixes high bias
- Try decreasing  $\lambda$       fixes high bias
- Try increasing  $\lambda$       fixes high variance

# Multi Collinearity

- When a dataset is having more features , it is possible that few of these features may be highly correlated
- This leads to multi collinearity
- Presence of this can destabilize the model
  - ❖ Multicollinearity arises when two variables that measure the same thing or similar things (e.g., weight and BMI) are both included in a multiple regression model; they will, in effect, cancel each other out and generally destroy your model.
  - ❖ Model building and diagnostics are tricky business!





## Assumptions

- Errors / residuals follow normal distribution
- Homoscedasticity\_ variance of error is constant
- The error and independent variable are uncorrelated
- Functional relationship between the outcome variable and feature is incorrectly defined

# R Squared Error

Advertising (in lakhs of rupees)	Sales (in lakhs of rupees)
10	520
20	625
35	700
50	780
20	605



# Thank You!

In our next session: Classification Models



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



**Session 5(25<sup>th</sup> August,2023)  
(Friday)**

4	Linear Prediction Models: Linear Regression, Gradient Descent and Variants, Regularization, Bias Vs. Variance	T1: Chapter 4
---	---	---------------



# Regression



# Least Squares Based Solution

- 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  $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

# Least Squares Based Solution

- Consider our model for  $n$  instances:

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

- Let

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

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

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

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

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

# Least Squares Based Solution

- For the linear regression cost function:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \sum_{i=1}^n \left( \boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top}_{\mathbb{R}^{1 \times n}} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\mathbb{R}^{n \times 1}} \end{aligned}$$

Let:

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

# Least Squares Based Solution

- Solve for optimal  $\theta$  analytically
  - Notice that the solution is when  $\frac{\partial}{\partial \theta} J(\theta) = 0$
- Derivation:

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

1 x 1

Take derivative and set equal to 0, then solve for  $\theta$ :

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

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

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

Closed Form Solution:

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

# Linear regression



# Linear regression



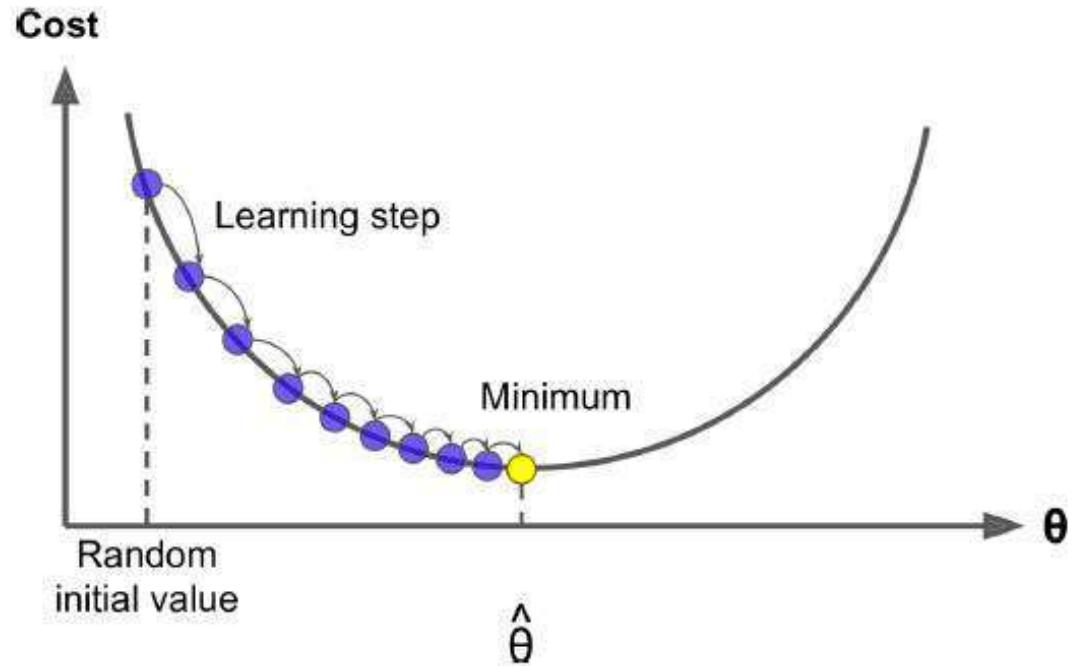
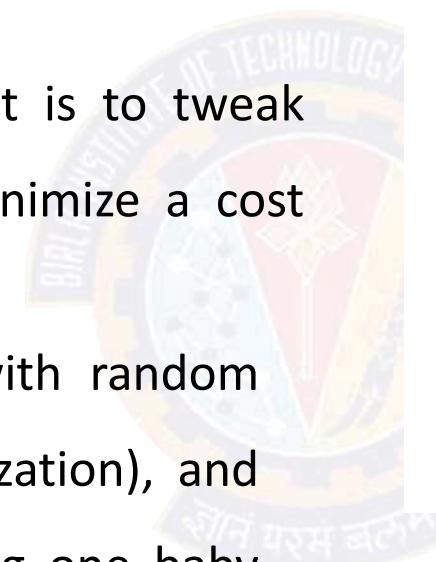
# Linear regression





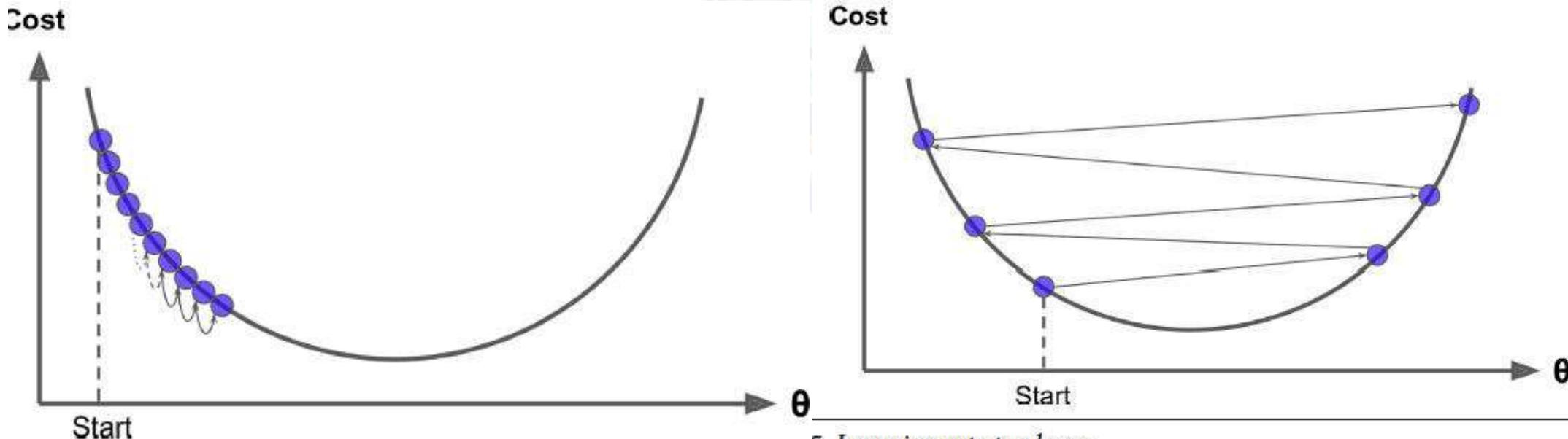
# Gradient Descent

- Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function.
- Concretely, you start by filling  $\theta$  with random values (this is called random initialization), and then you improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum



# Gradient Descent- Hyper parameter

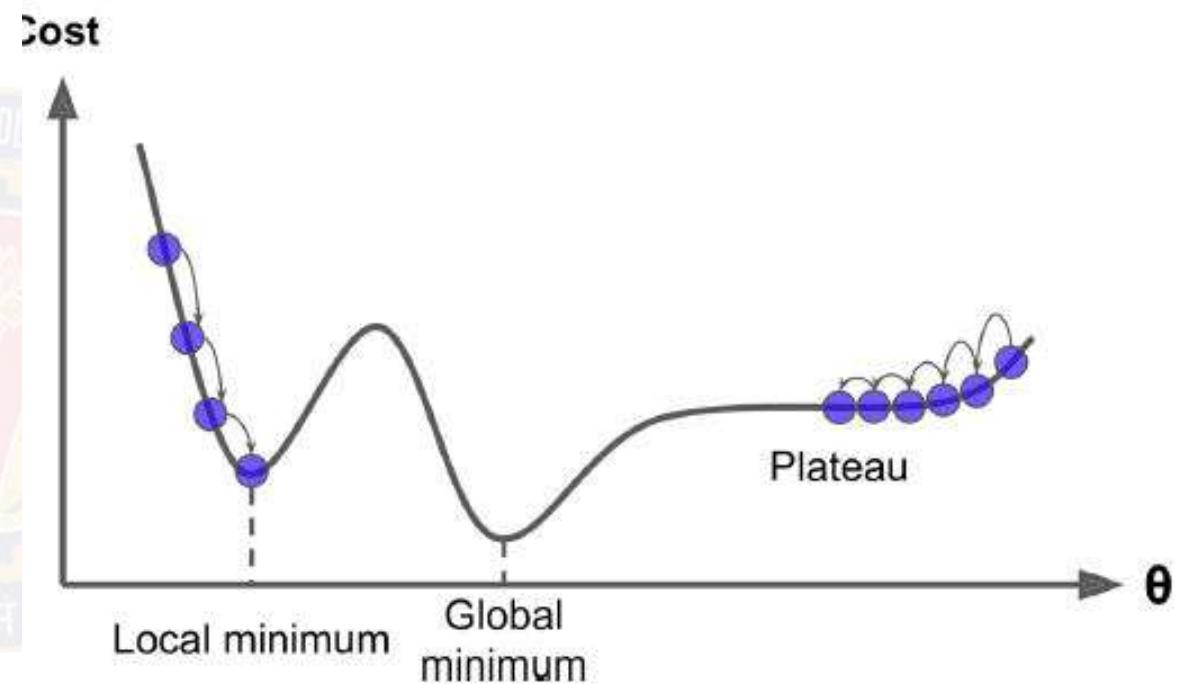
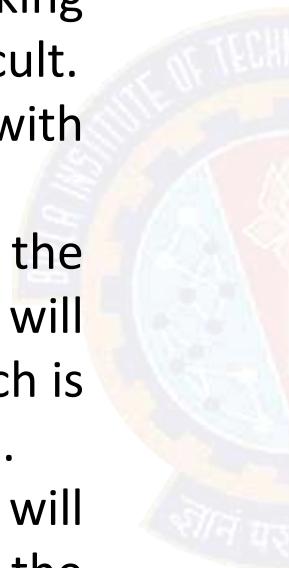
If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time



On the other hand, if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before. This might make the algorithm diverge, with larger and larger values, failing to find a good solution

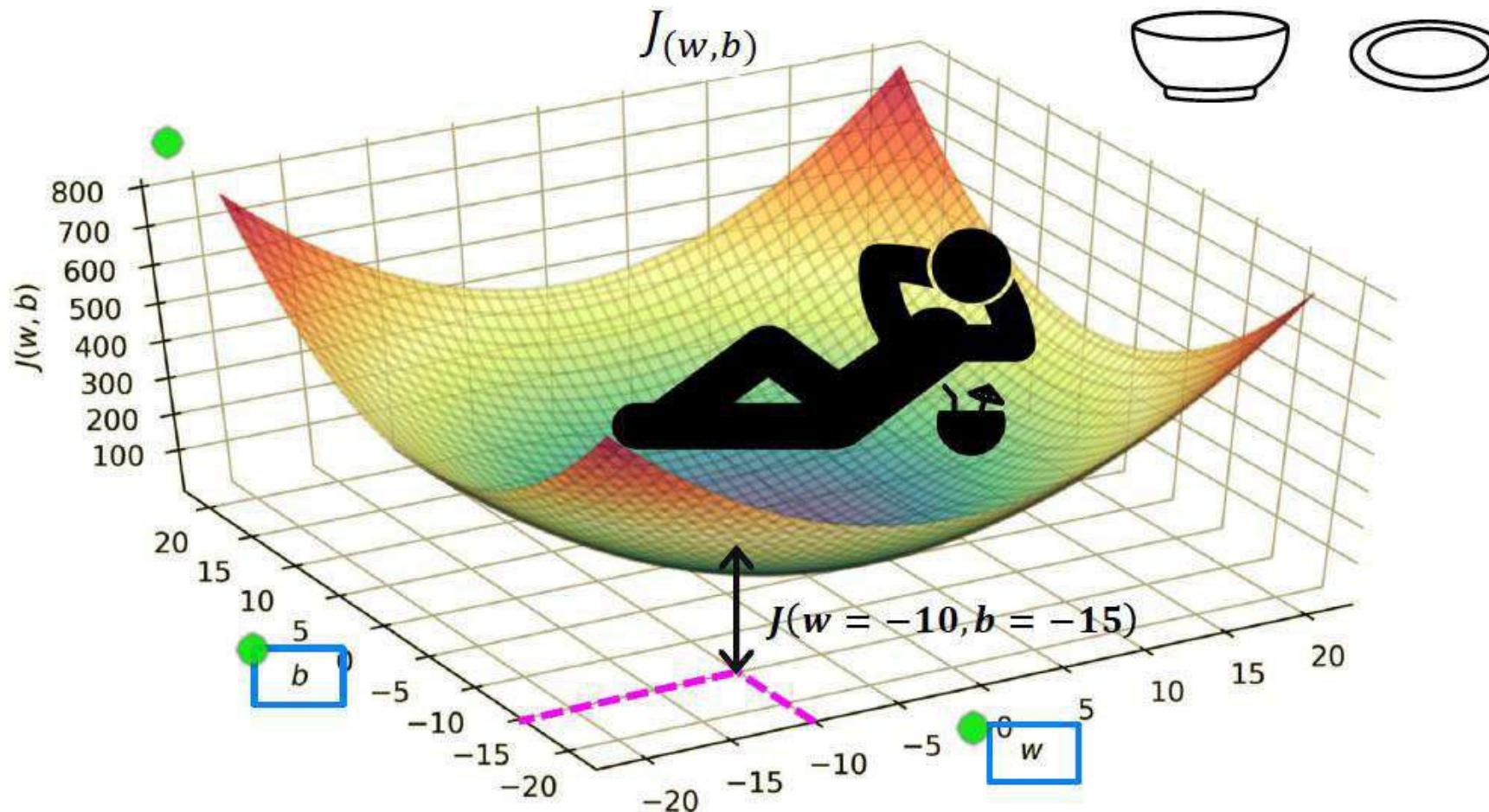
# Gradient Descent- Pitfalls

- Finally, not all cost functions look like nice regular bowls.
- There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult.
- **Figure** shows the two main challenges with Gradient Descent:
  - if the random initialization starts the algorithm on the left, then it will converge to a local minimum , which is not as good as the global minimum.
  - If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.



*Gradient Descent pitfalls*

# Intuition Behind Cost Function



# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

For Linear Regression:

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

..

# Gradient Descent for Linear Regression

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous  
update  
for  $j = 0 \dots d$

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

$L_2$  norm:  $\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$

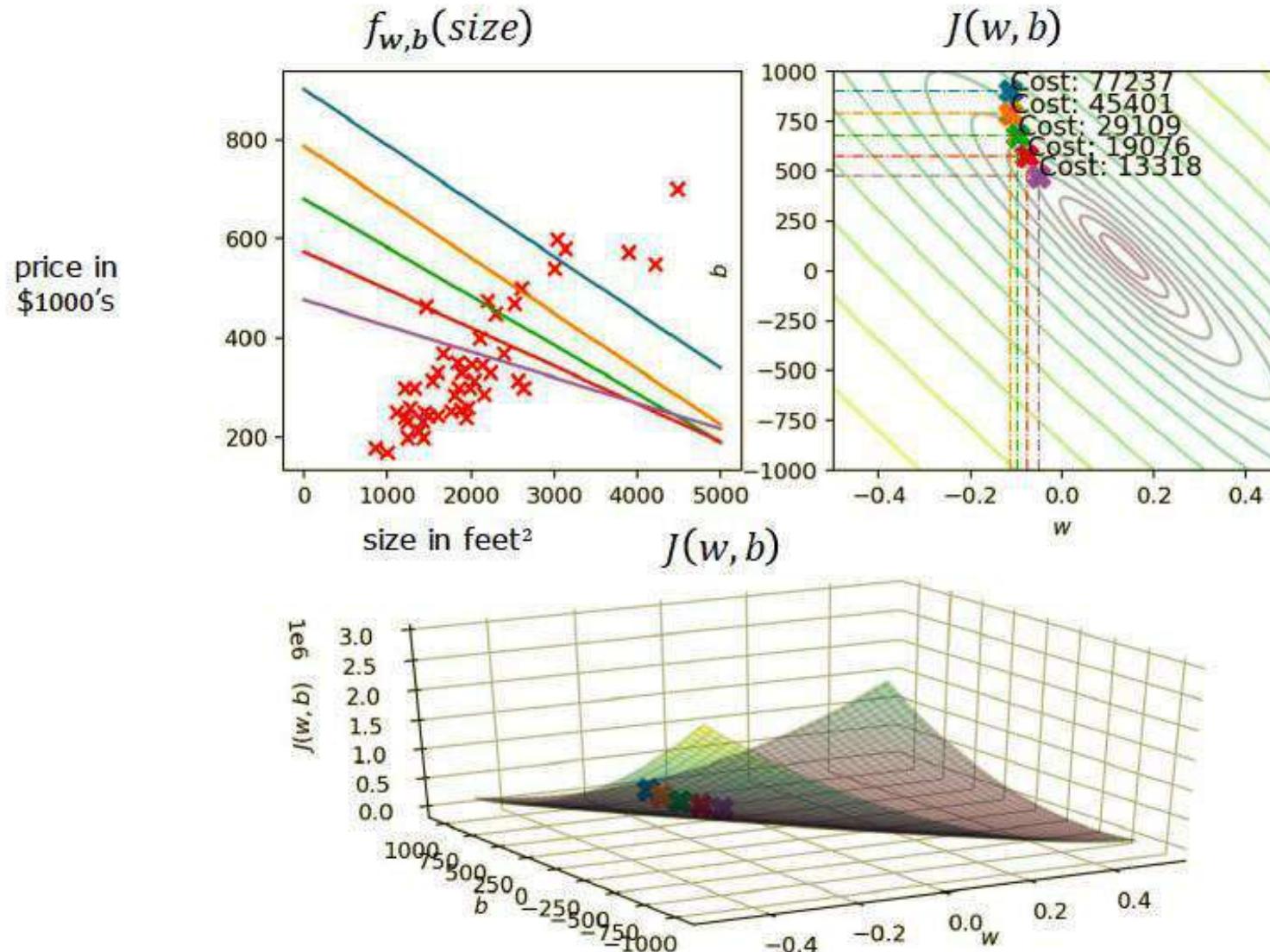






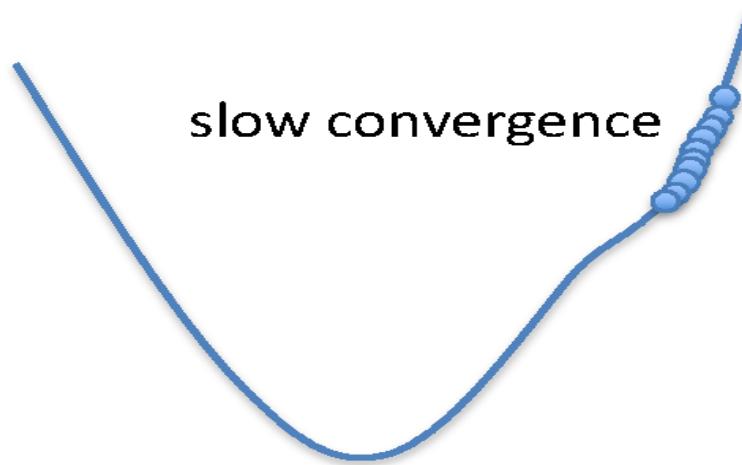


# Running Gradient Descent

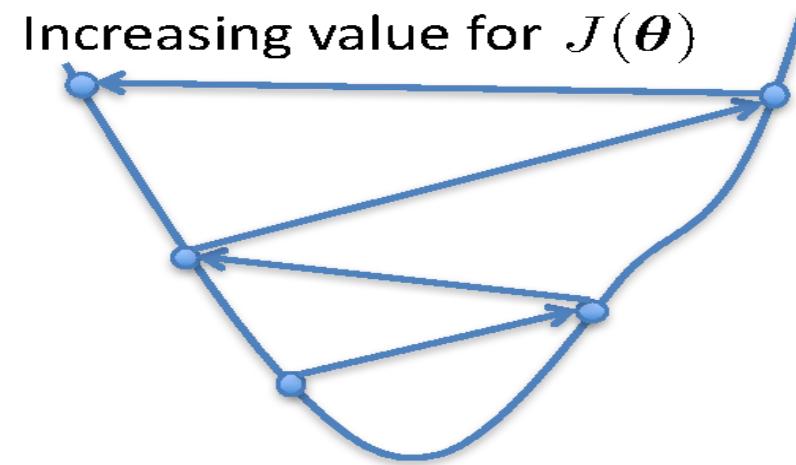


# Choosing Step Size

$\alpha$  too small



$\alpha$  too large

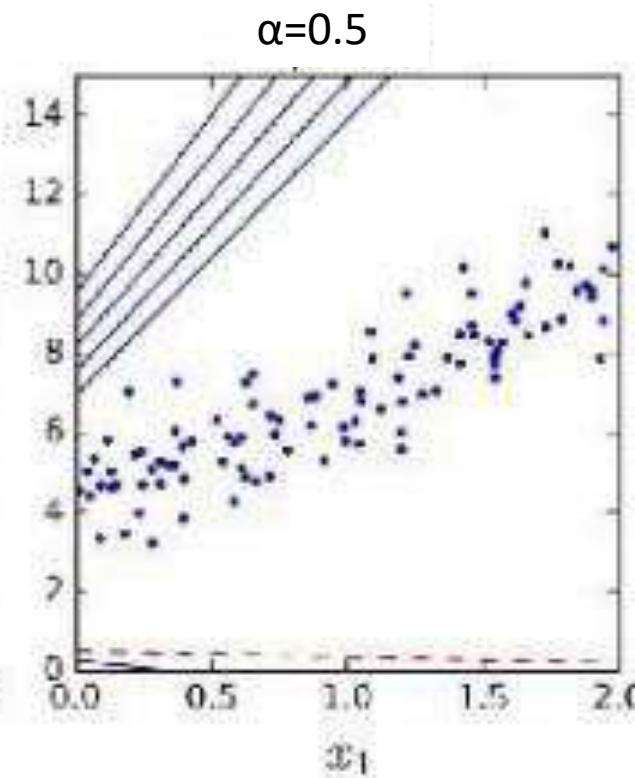
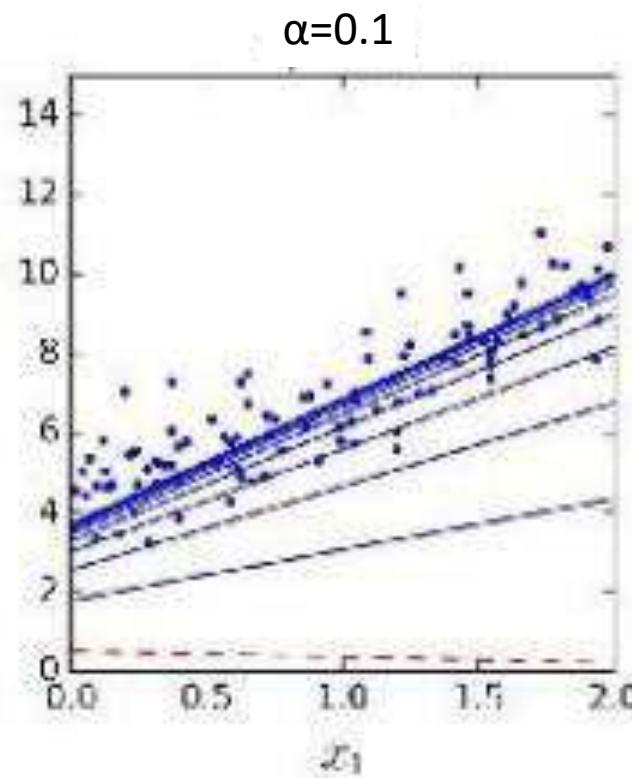
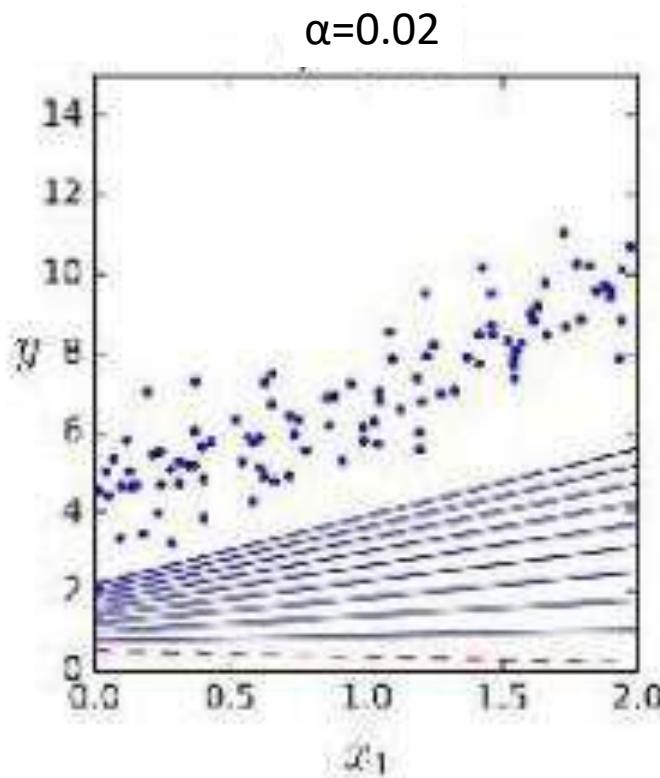


- May overshoot the minimum
- May fail to converge
- May even diverge

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  $\alpha$

# Impact of Learning Rate



# Mini-batch Gradient Descent Algorithm



- ❖ GD can be very slow on large datasets
- ❖ So in such case SGD can be used
- ❖ In SGD, the procedure is same as GD but the difference is updatation of coefficients is done for each training example rather than at the end of training
- ❖ i. e. cost is not summed over all training examples , but instead for one training example

# Stochastic Gradient Descent Algorithm



# Mini Batch Stochastic Gradient Descent Algorithm(MBSGD)









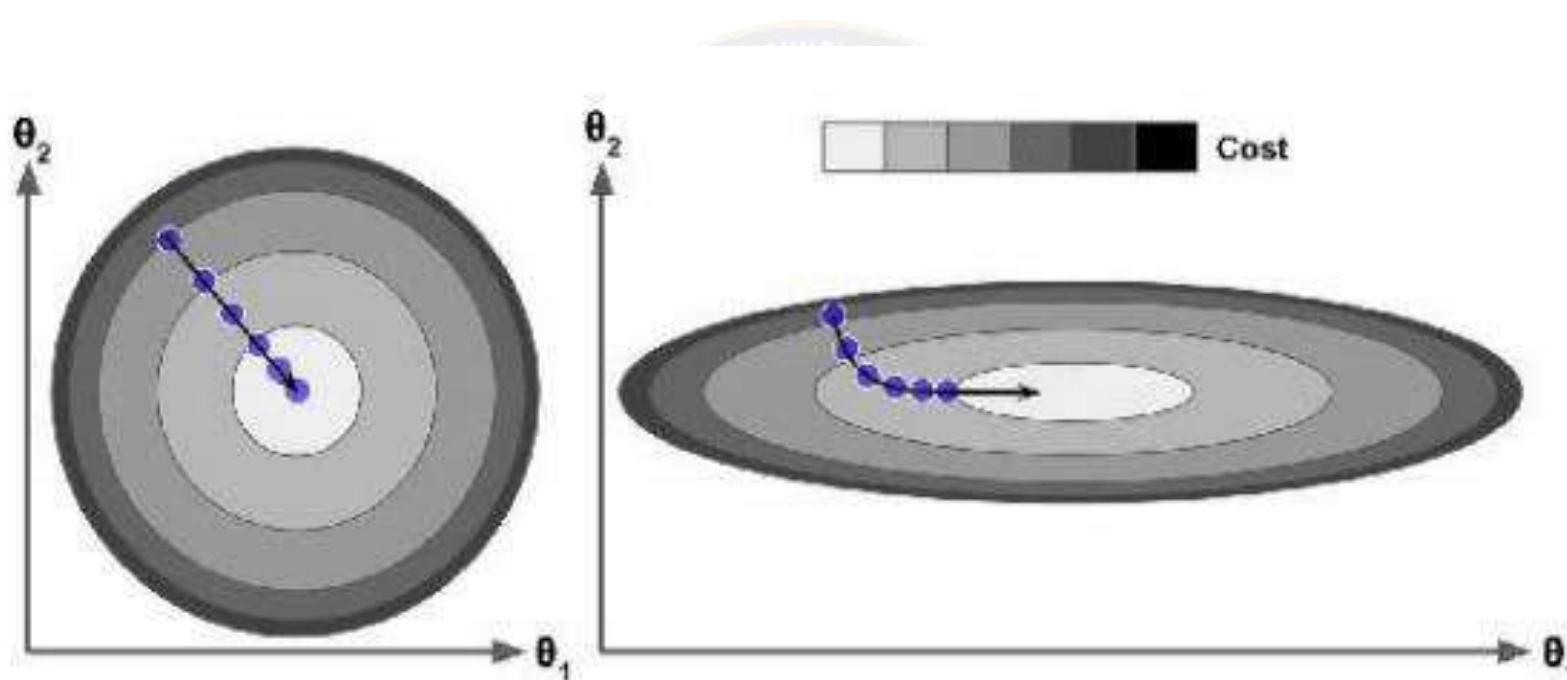
# Summary

- ❖ Optimization is a big part of machine learning.
- ❖ Gradient descent is a simple optimization procedure that you can use with many machine learning algorithms.
- ❖ 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

# Feature Normalization

**Gradient Descent can be slow when feature scales are widely different**

- Normalization of feature values for same variance needed for faster convergence

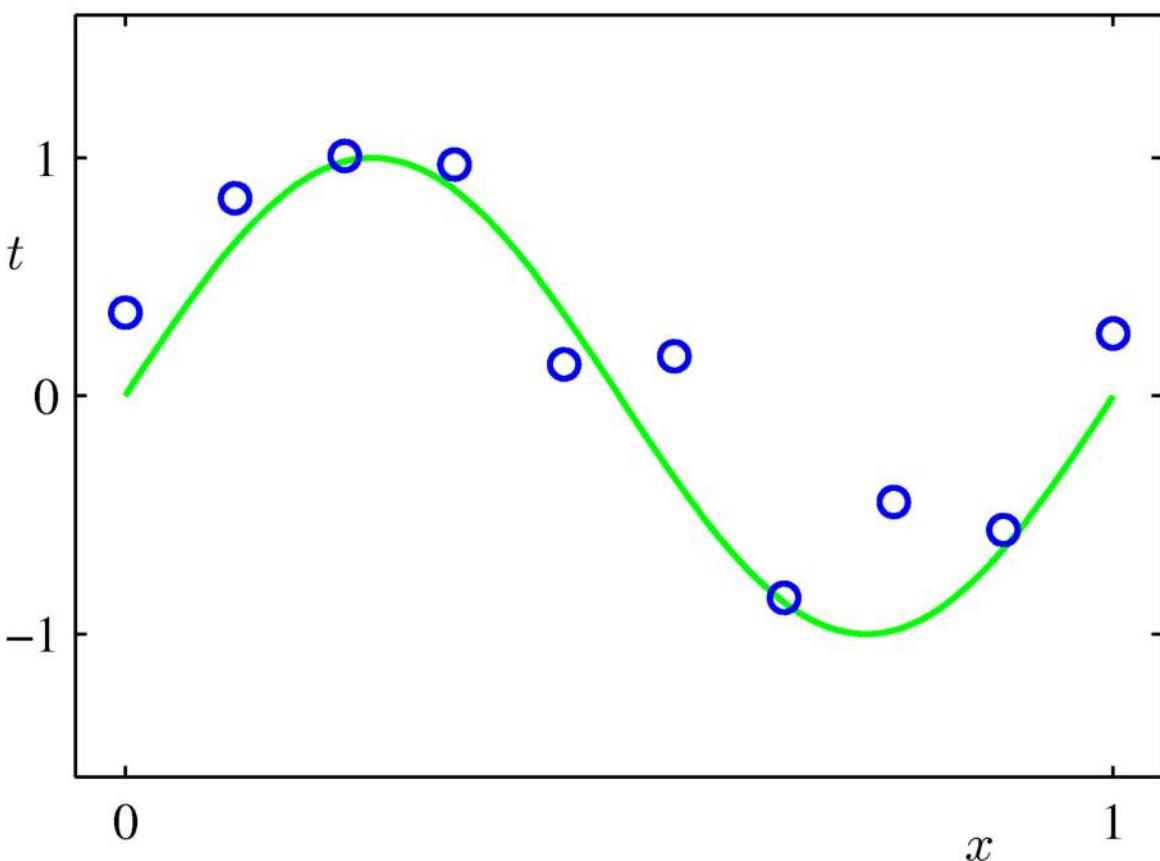


# Extending to More Complex Model

- 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:  $x_3 = x_1 \cdot x_2$

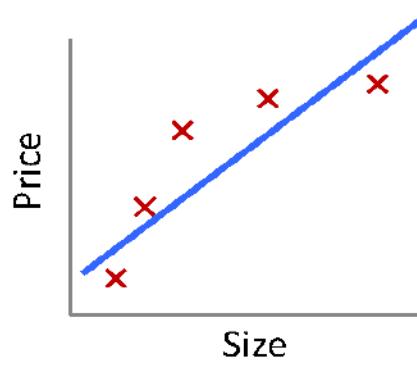
This allows use of linear regression techniques  
to fit non-linear datasets.

# Fitting a Polynomial Curve

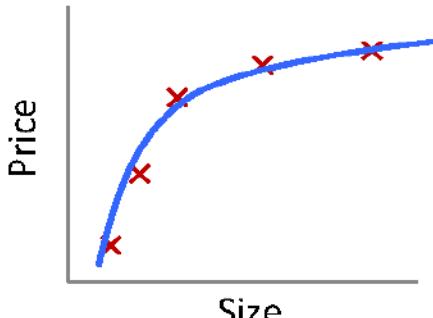


$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j = \sum_{j=0}^p \theta_j z_j$$
$$z_j = x^j$$

# Quality of Fit

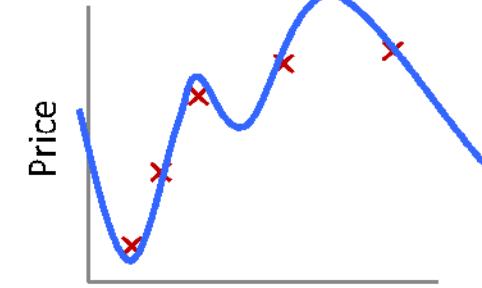


Underfitting  
(high bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$

Correct fit



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

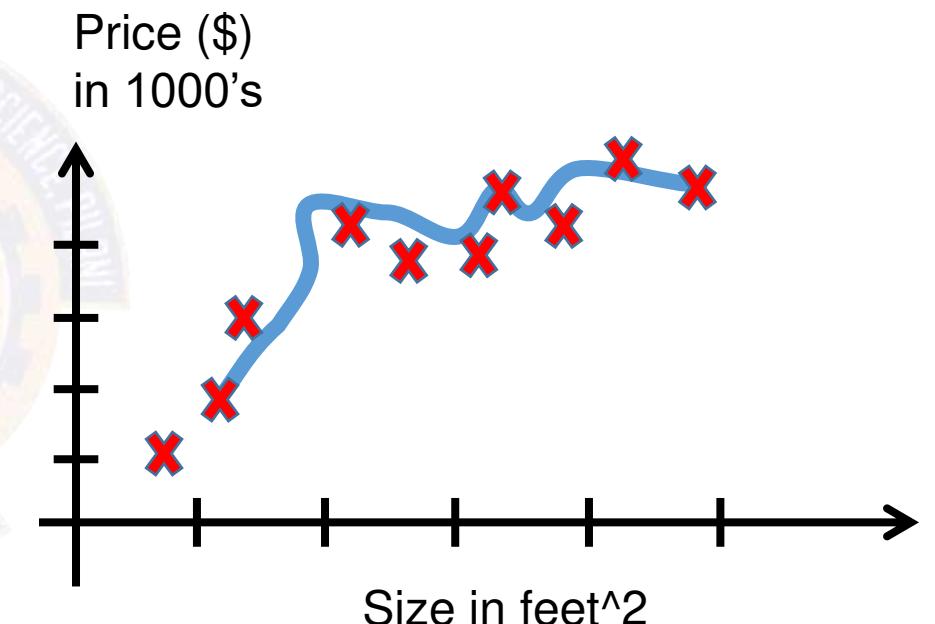
Overfitting  
(high variance)

## Overfitting:

- The learned hypothesis may fit the training set very well ( $J(\theta) \approx 0$ )
- ...but fails to generalize to new examples

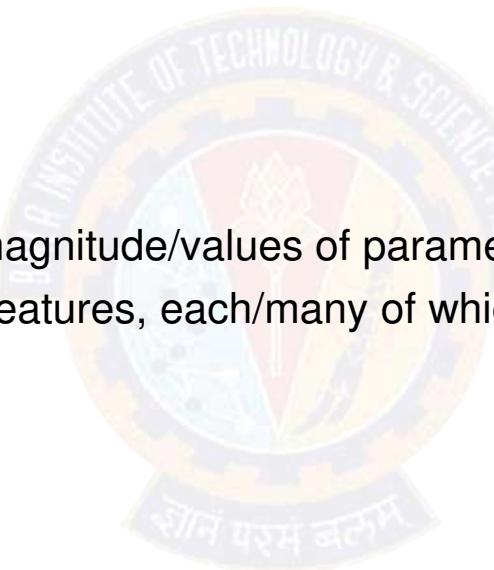
# Addressing overfitting

- $x_1$  = 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
- :
- $x_{100}$



# Addressing overfitting

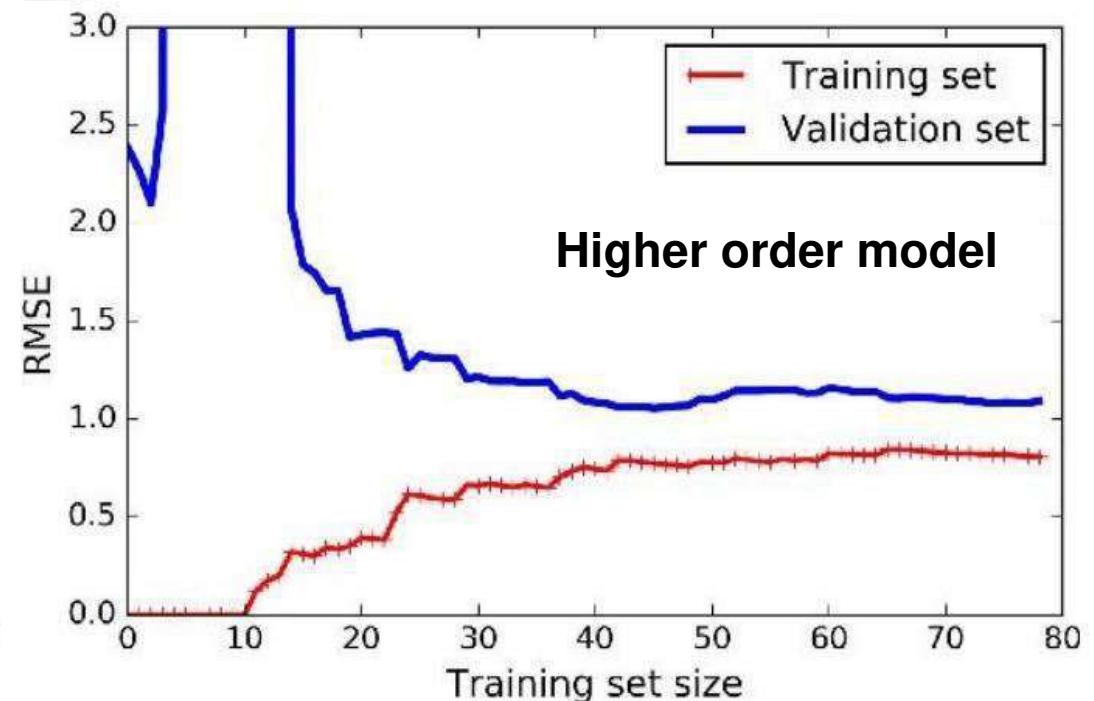
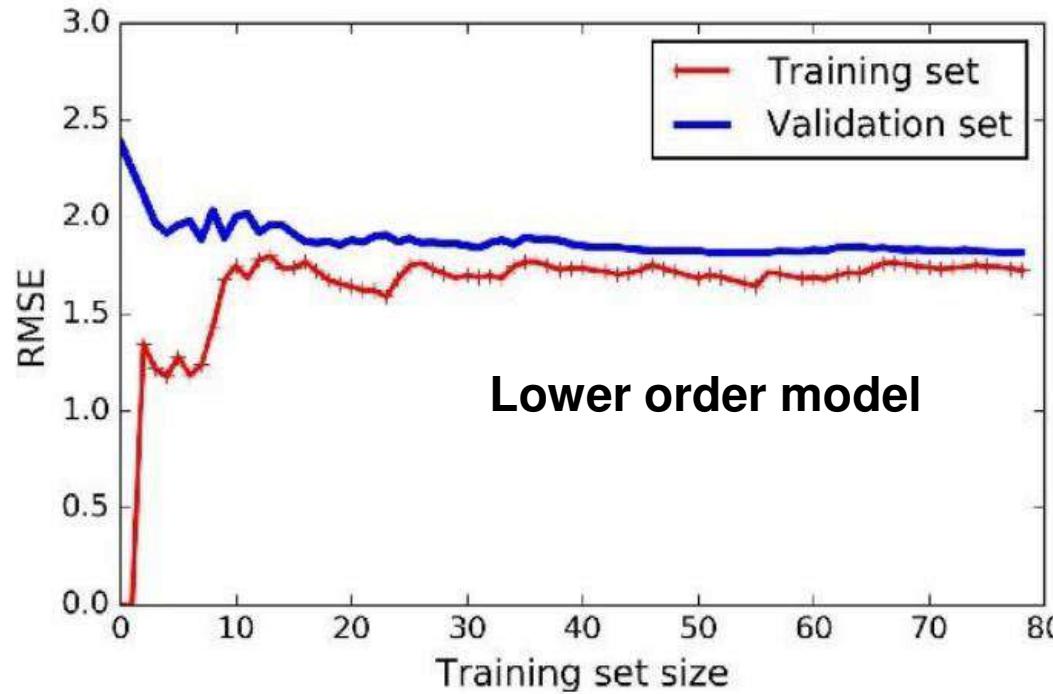
- Reduce number of features.
  - Manually select which features to keep.
  - Model selection algorithm
- Regularization.
  - Keep all the features, but reduce magnitude/values of parameters  $\theta_j$ .
  - Works well when we have a lot of features, each/many of which contributes a bit to predicting  $y$ .



# Effect of Training Size on Overfitting

Size of training dataset needs to be large to prevent overfitting

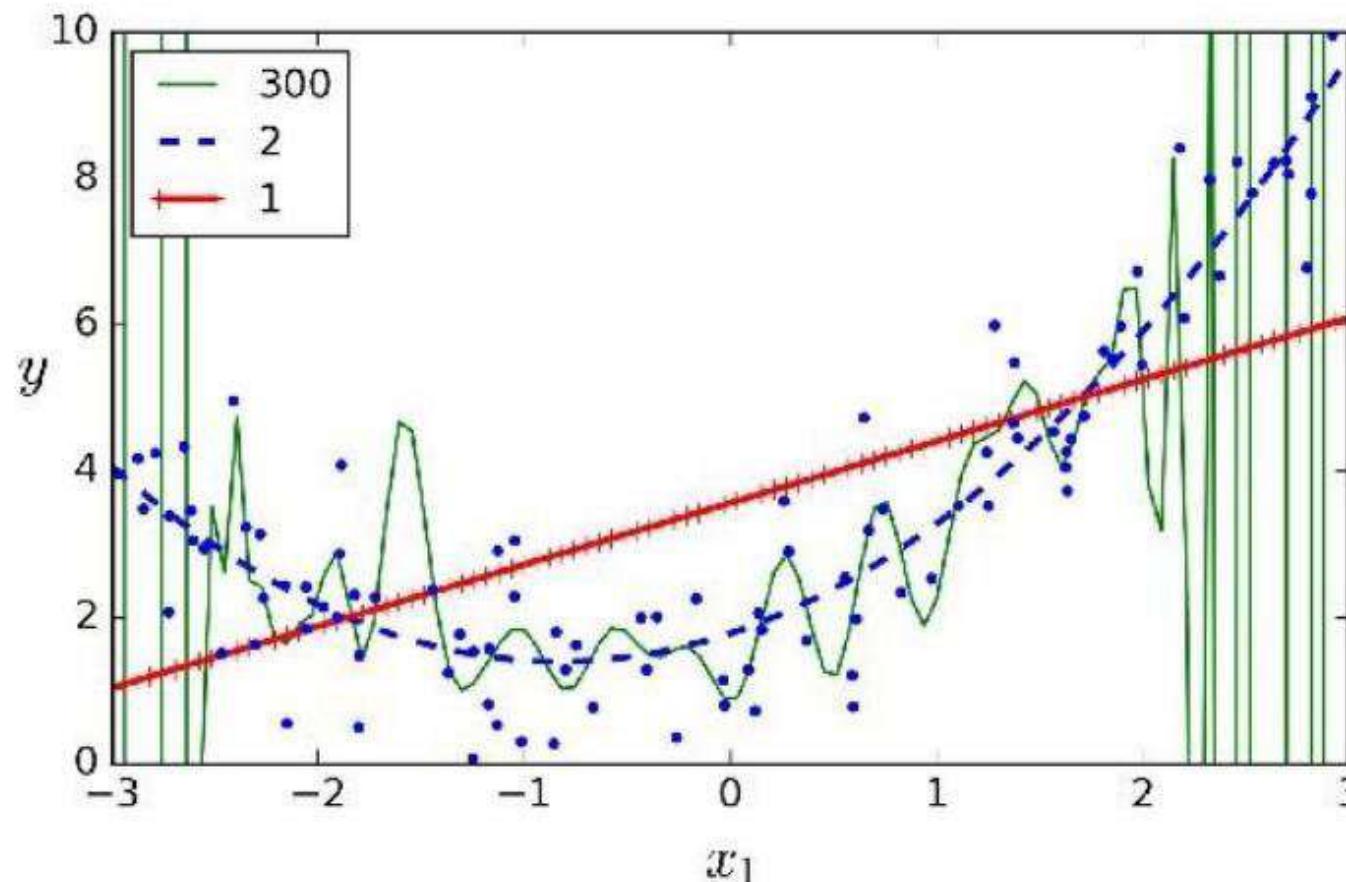
- when higher order model is used.



# Polynomial Fitting can lead to Overfitting

Underlying target function is quadratic

- Linear model results in under fitting with large bias
- Polynomial of order 300 results in a large variance

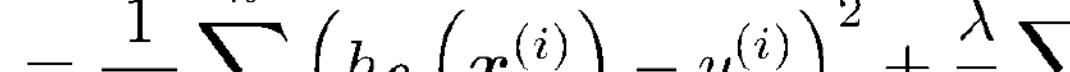


# Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of  $\theta_j$ 
  - 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)

# Ridge Regularization

- Linear regression objective function

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


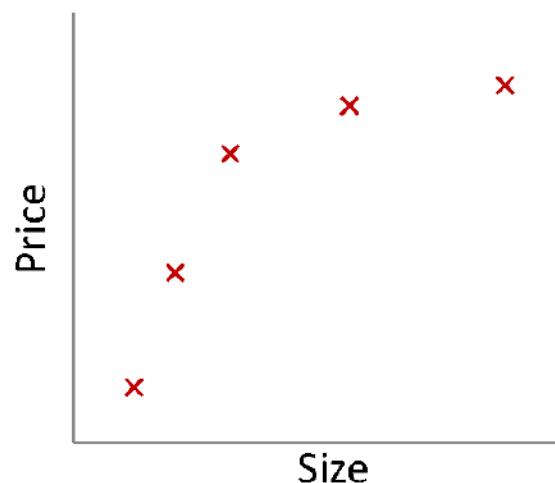
The diagram illustrates the cost function  $J(\theta)$  as a sum of two terms. The first term,  $\frac{1}{2n} \sum_{i=1}^n \left( h_{\theta} (x^{(i)}) - y^{(i)} \right)^2$ , is represented by a blue bracket below the equation and labeled 'model fit to data' in blue text. The second term,  $\frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$ , is represented by a blue bracket to the right of the first term and labeled 'regularization' in blue text.

- $\lambda$  is the regularization parameter ( $\lambda \geq 0$ )
  - No regularization on  $\theta_0$ !

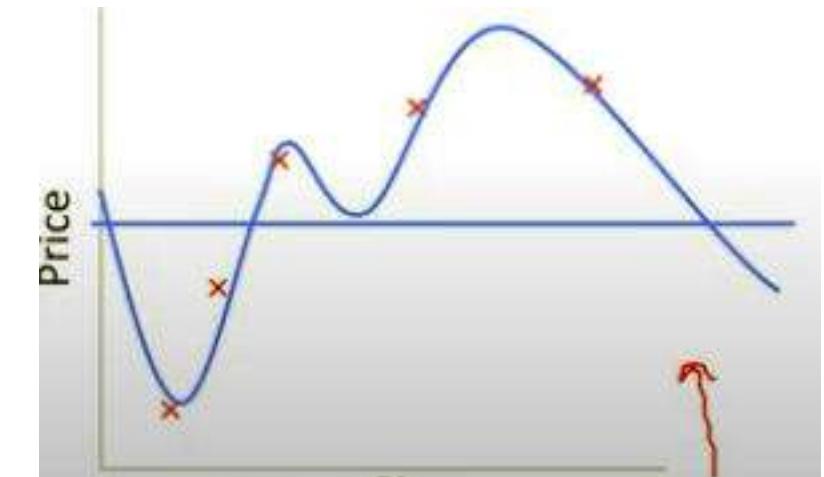
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



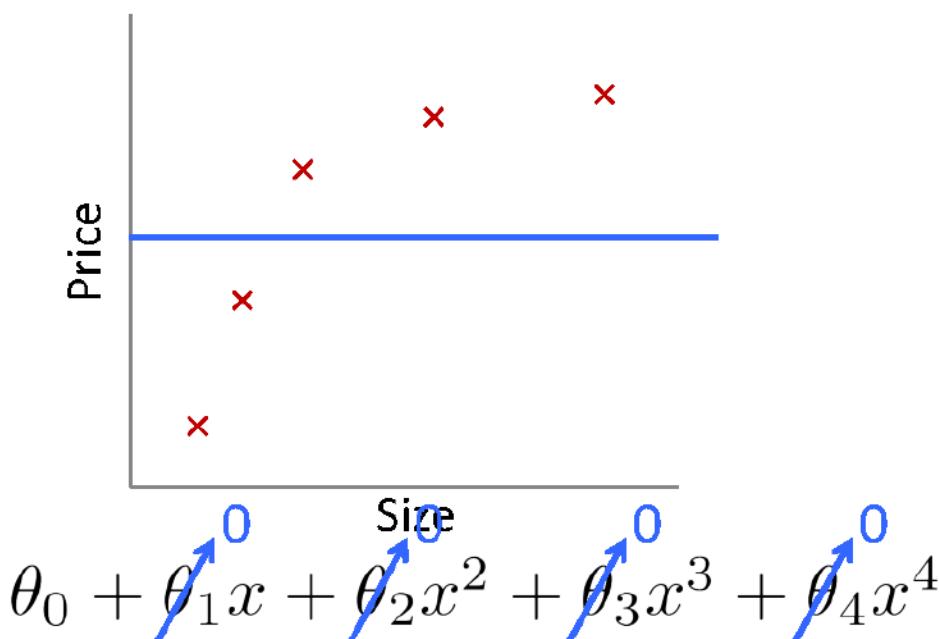
Overfitting

Based on example by Andrew Ng

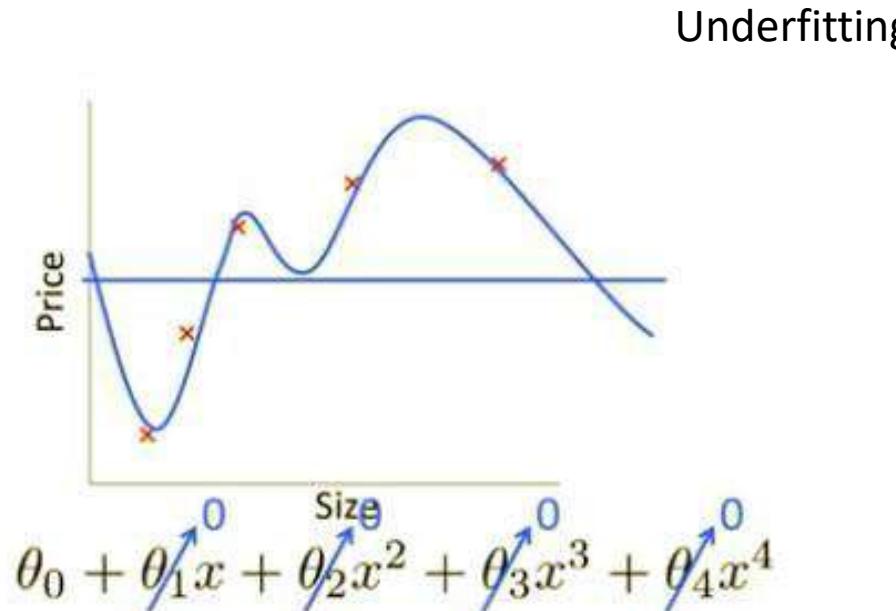
# Understanding Regularization

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

- What happens if we set  $\lambda$  to be huge (e.g.,  $10^{10}$ )?



Based on example by Andrew Ng



# Ridge regression

## 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:

$$\frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta})$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

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

regularization 57

# Ridge Regression

Further simplified

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

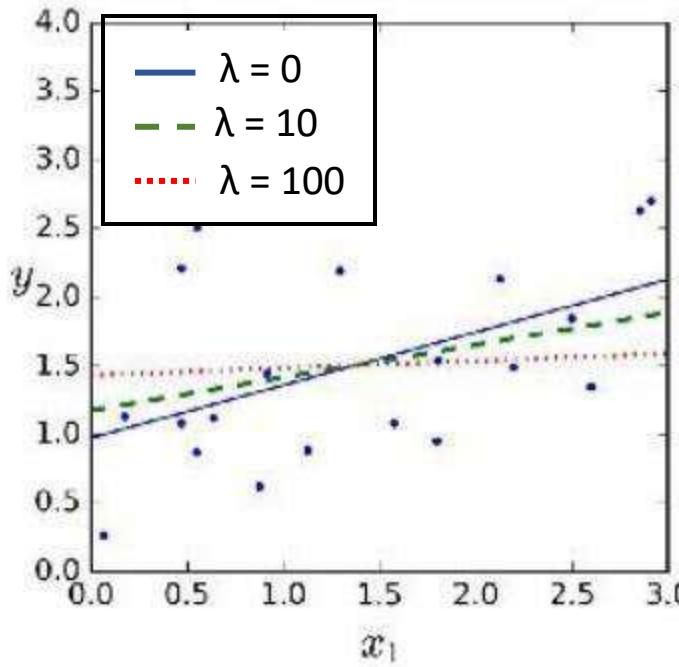
$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

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

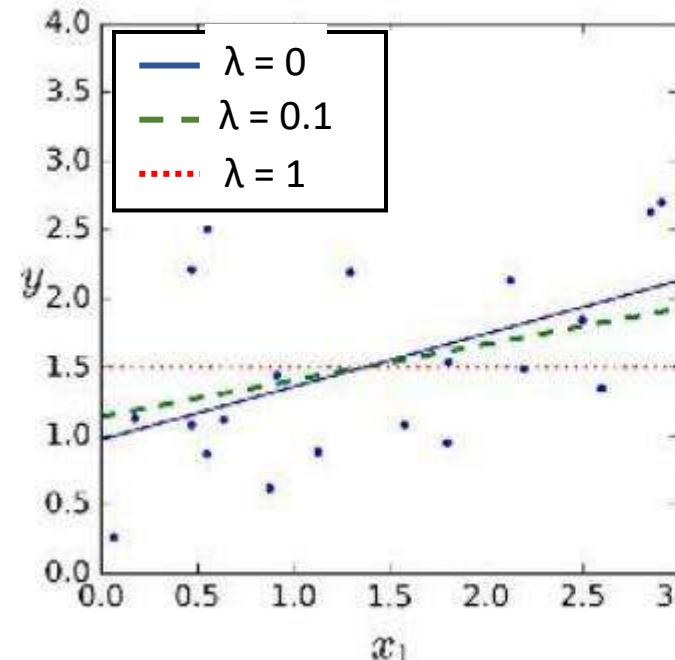
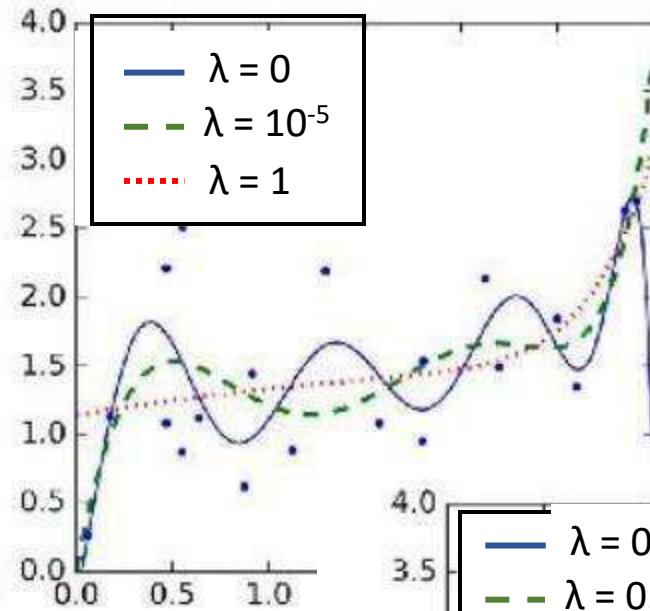
- We can rewrite the gradient step as:

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

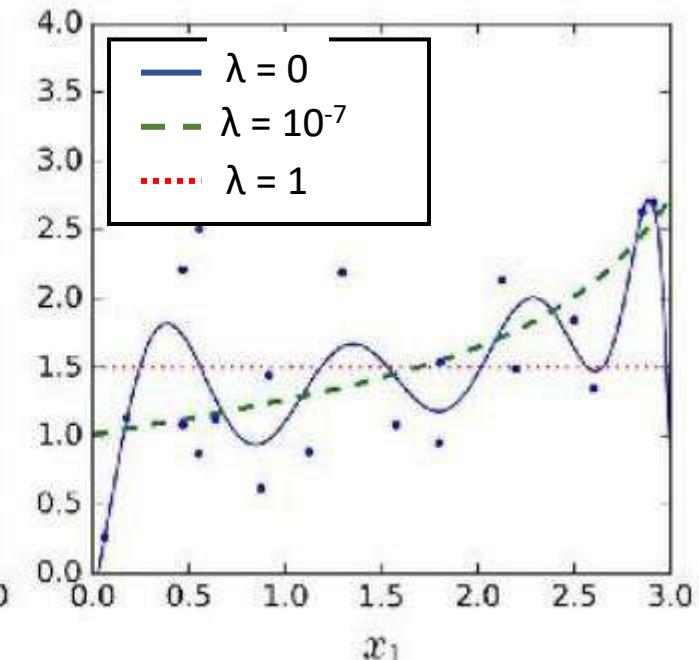
# Ridge Vs Lasso Regularization



Ridge



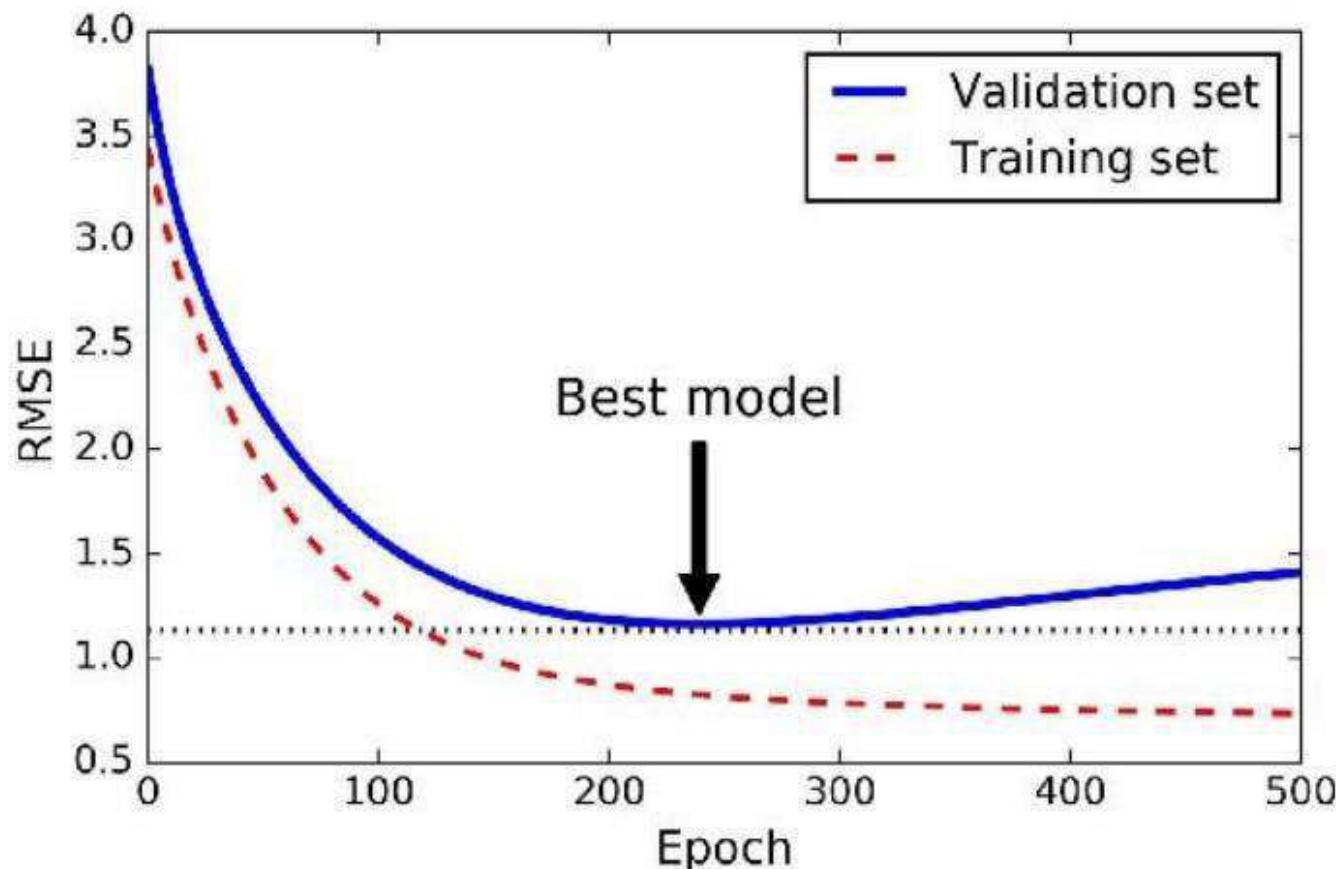
Lasso



# Early Stopping

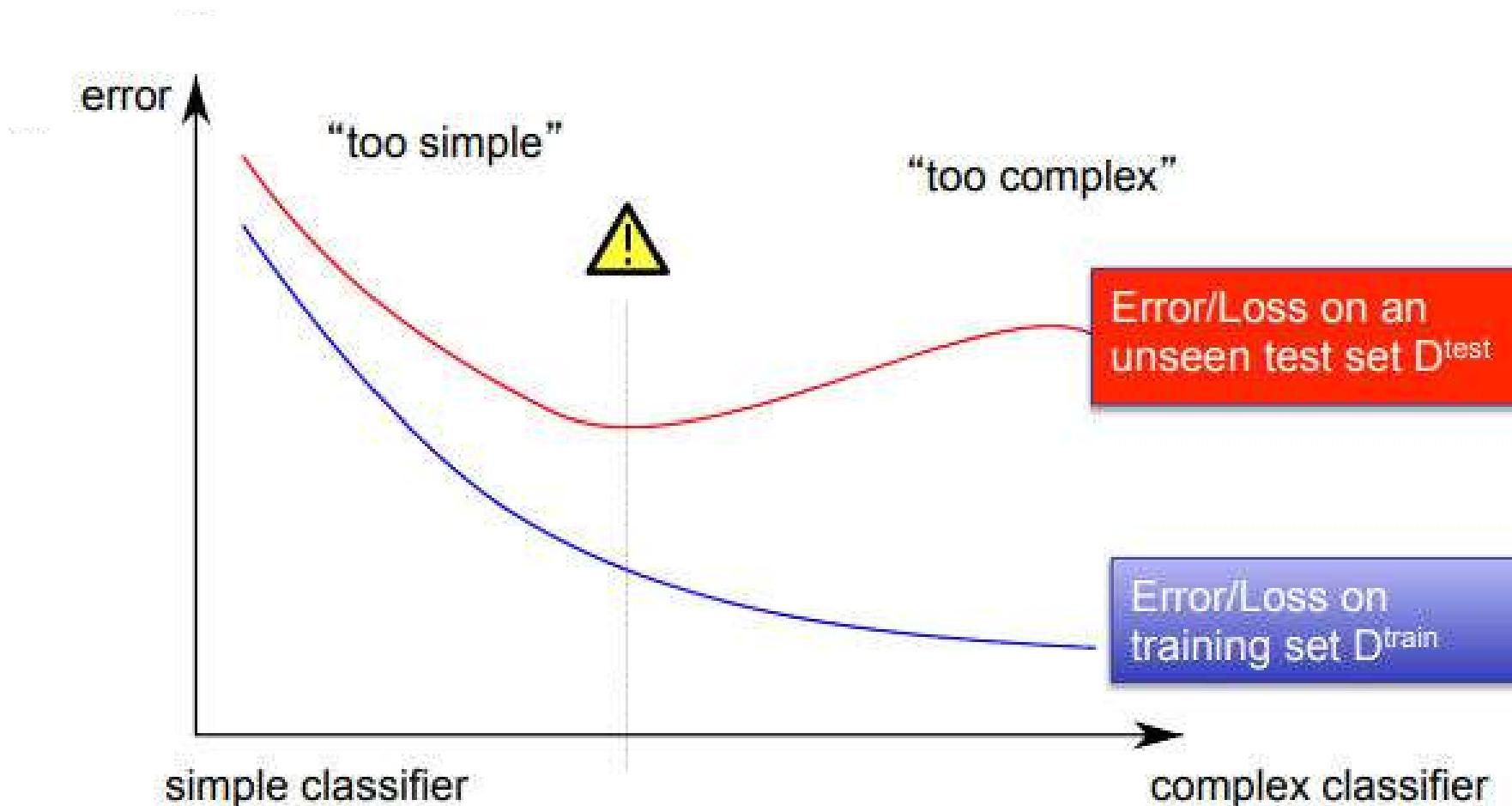
## Do Not Over train to prevent overfitting

- Stop training once error on the validation set starts showing an upward trend, even if the error on the training set keeps decreasing



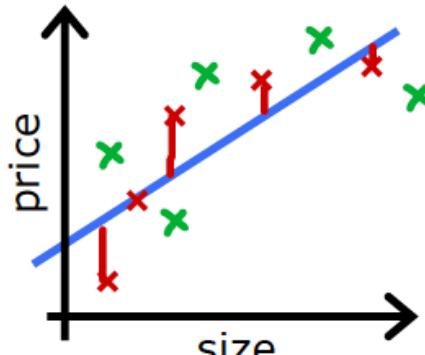
# Bias Variance Tradeoff

Bias/Variance is a Way to Understand Overfitting and Underfitting



# Bias and Variance

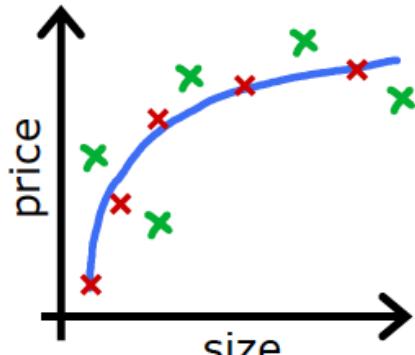
## Bias/variance



$$f_{\vec{w},b}(x) = w_1x + b$$

→ High bias  
(underfit)

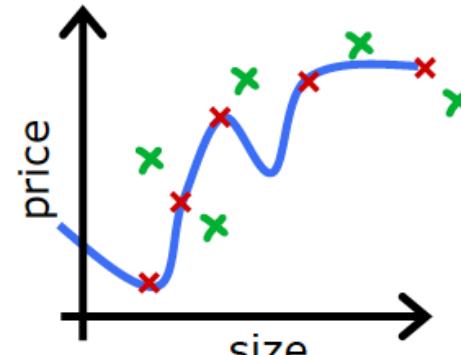
$d = 1$   $J_{train}$  is high  
 $J_{cv}$  is high



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$$

"Just right"

$d = 2$   $J_{train}$  is low  
 $J_{cv}$  is low



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$$

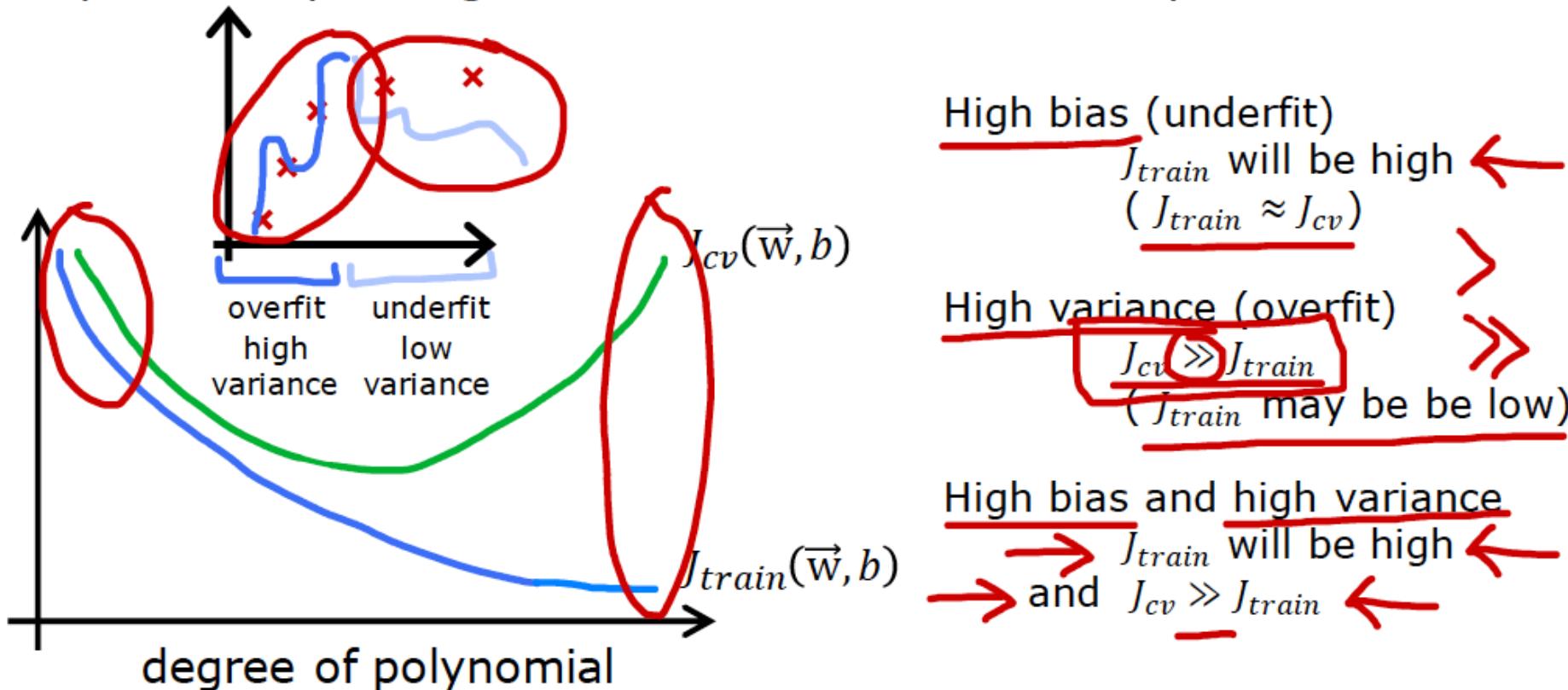
High variance  
(overfit)

$d = 4$   $J_{train}$  is low  
 $J_{cv}$  is high

# Bias and Variance

## Diagnosing bias and variance

How do you tell if your algorithm has a bias or variance problem?

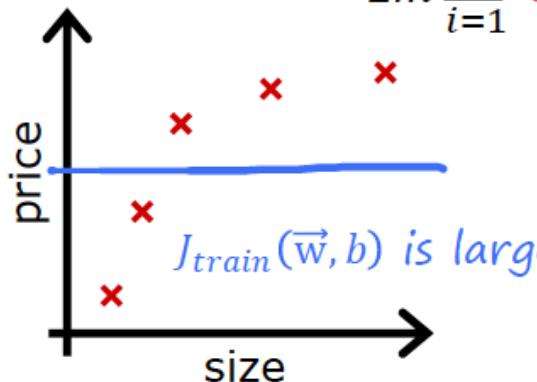


# Bias and Variance

## Linear regression with regularization

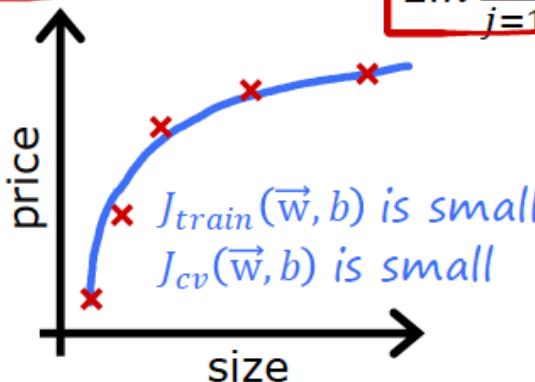
Model:  $f_{\vec{w}, b}(x) = \underbrace{w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b}_m$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



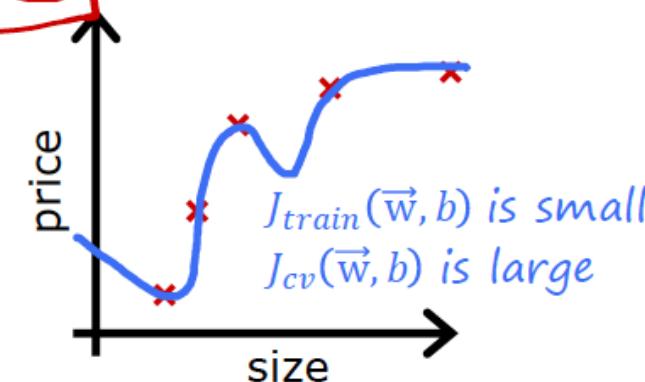
Large  $\lambda$   
High bias (underfit)

$\lambda = 10,000$   $w_1 \approx 0, w_2 \approx 0$   
 $f_{\vec{w}, b}(\vec{x}) \approx b$



Intermediate  $\lambda$

$\lambda$



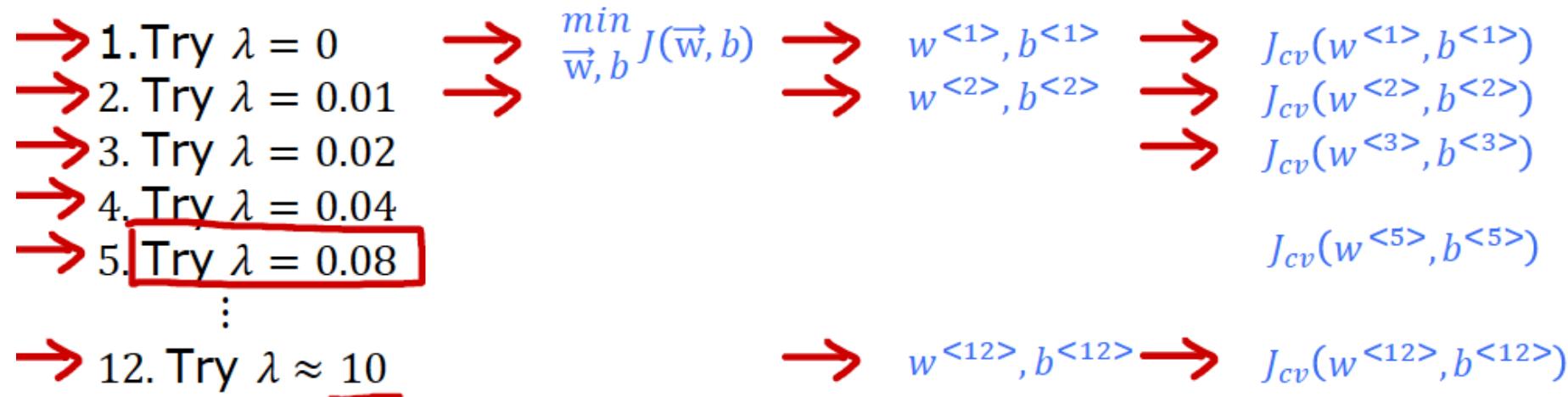
Small  $\lambda$   
High variance (overfit)

$\lambda = 0$

# Bias and Variance

## Choosing the regularization parameter $\lambda$

Model:  $f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$



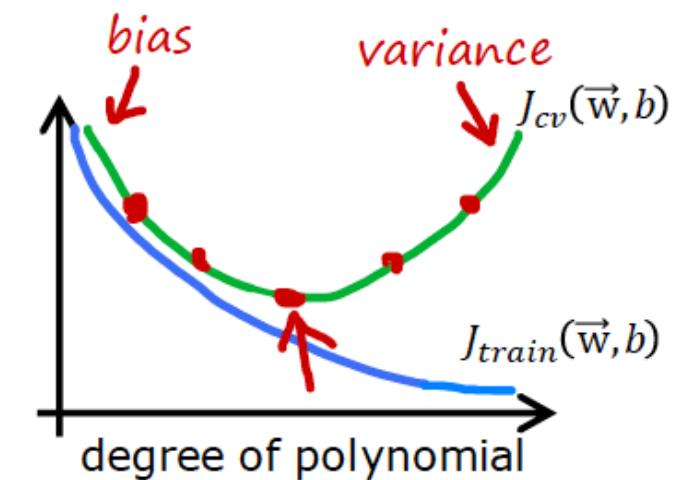
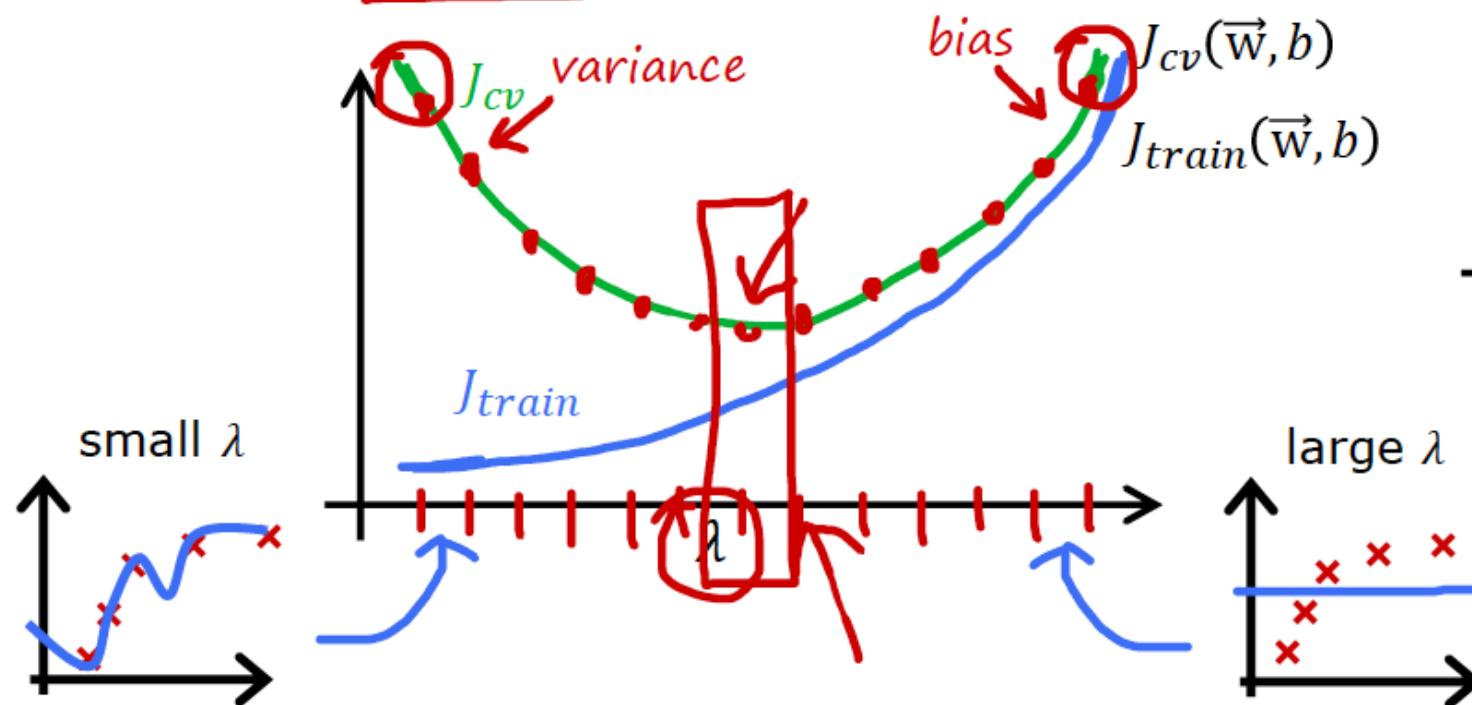
Pick  $w^{<5>}, b^{<5>}$

Report test error:  $J_{test}(w^{<5>}, b^{<5>})$

# Bias and Variance

Bias and variance as a function of regularization parameter  $\lambda$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$



## Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

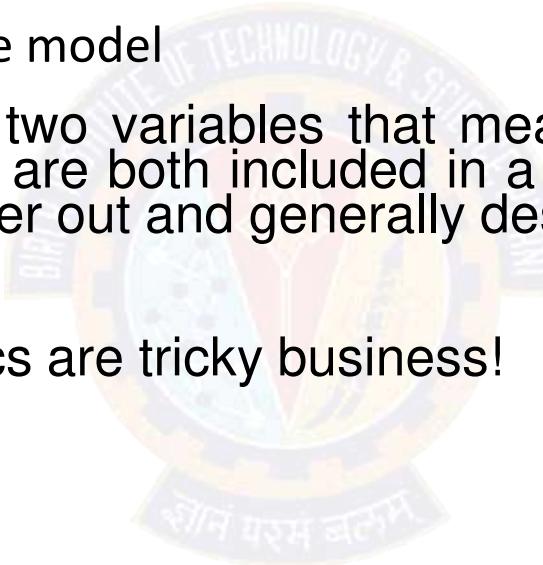
$$J(\vec{w}, b) = \underbrace{\frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2}_{\text{large errors}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n w_j^2}_{\text{large errors}}$$

But it makes unacceptably large errors in predictions. What do you try next?

- Get more training examples      fixes high variance
- Try smaller sets of features  $x, x^2, \cancel{x^3}, \cancel{x^4}, \cancel{x^5} \dots$       fixes high variance
- Try getting additional features      fixes high bias
- Try adding polynomial features  $(x_1^2, x_2^2, x_1 x_2, \text{etc})$       fixes high bias
- Try decreasing  $\lambda$       fixes high bias
- Try increasing  $\lambda$       fixes high variance

# Multi Collinearity

- When a dataset is having more features , it is possible that few of these features may be highly correlated
- This leads to multi collinearity
- Presence of this can destabilize the model
  - ❖ Multicollinearity arises when two variables that measure the same thing or similar things (e.g., weight and BMI) are both included in a multiple regression model; they will, in effect, cancel each other out and generally destroy your model.
  - ❖ Model building and diagnostics are tricky business!





## Assumptions

- Errors / residuals follow normal distribution
- Homoscedasticity\_ variance of error is constant
- The error and independent variable are uncorrelated
- Functional relationship between the outcome variable and feature is incorrectly defined

# R Squared Error

Advertising (in lakhs of rupees)	Sales (in lakhs of rupees)
10	520
20	625
35	700
50	780
20	605



# Thank You!

In our next session: Classification Models



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 6(26<sup>th</sup> August,2023)**

# Session 6 – 26<sup>th</sup> August, 2023

- |   |   |  |
|---|---|--|
| 5 | Classification Models I: Naïve Bayes classification, T2: Chapter 5<br>Applications in text and image classification |  |
|---|---|--|









# Classification

## Definition

- Given a collection of records (training set )
  - Each record is characterized by a tuple  $(x, y)$ , where  $x$  is the attribute (feature) set and  $y$  is the class label
    - $x$  aka attribute, predictor, independent variable, input
    - $Y$  aka class, response, dependent variable, output
- Task
  - Learn a model or function that maps each attribute set  $x$  into one of the predefined class labels  $y$

Task	Attribute set, $x$	Class label, $y$
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

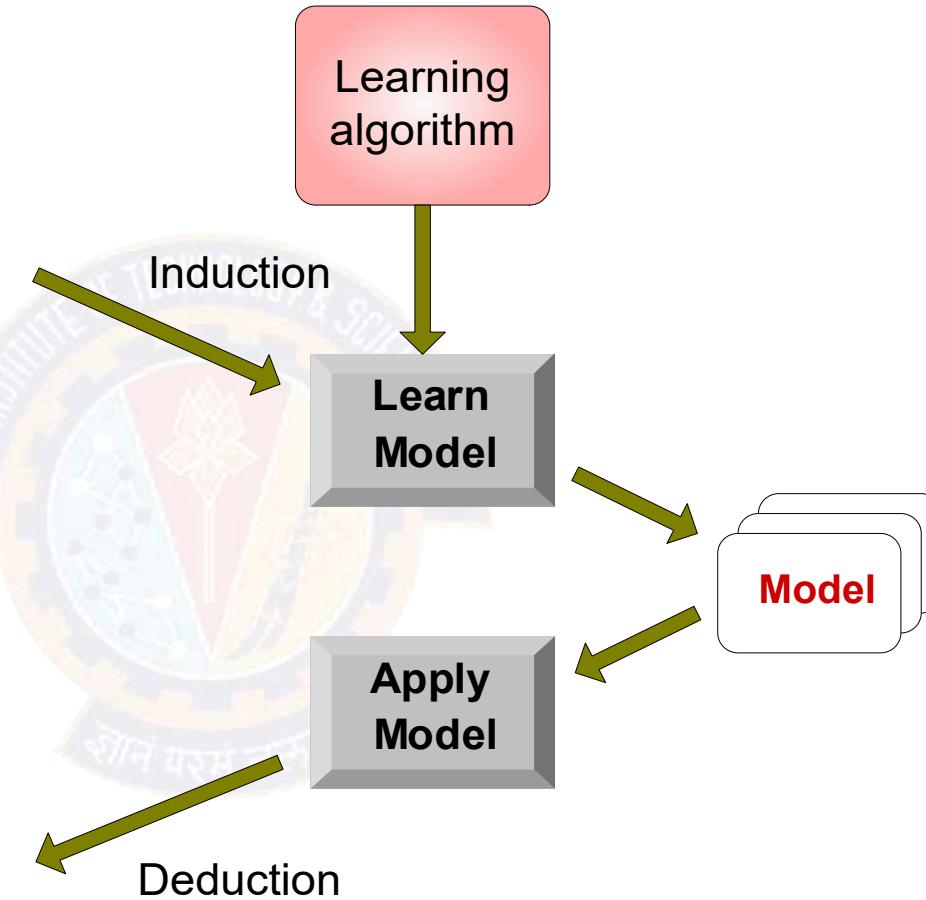
# General Approach for Building Classification Model

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

Training Set

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

Test Set



# Types of Classifiers

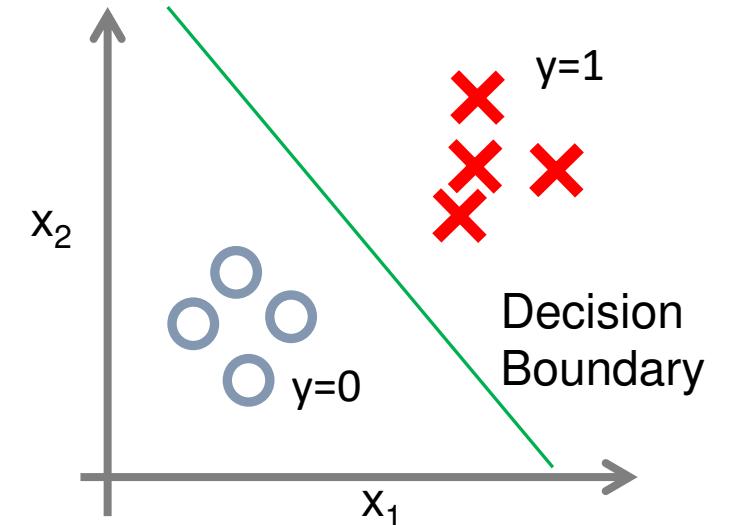
## Linear Classifier

- Classes are separated by a linear decision surface (e.g., straight line in 2-dimensional feature/attribute space)
  - If for a given record, linear combination of features  $x_i$  is  $\geq 0$ , i.e.,

$$w_0 + \sum_i w_i x_i \geq 0$$

it belongs to one class (say,  $y = 1$ ), else it belongs to the other class (say,  $y=0$  or -1)

- $w_i$ s are learned during the training (induction) phase of the classifier.
- Learnt  $w_i$ s are applied to a test record during the deduction / inferencing phase.

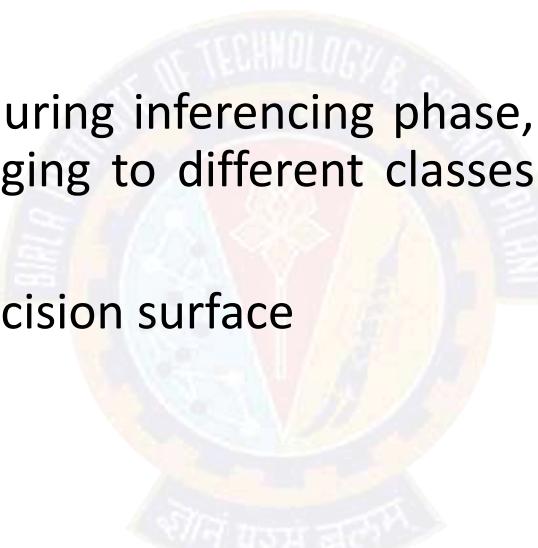


- In nonlinear classification, classes are separated by a non-linear surface

# Generative Vs. Discriminative Models

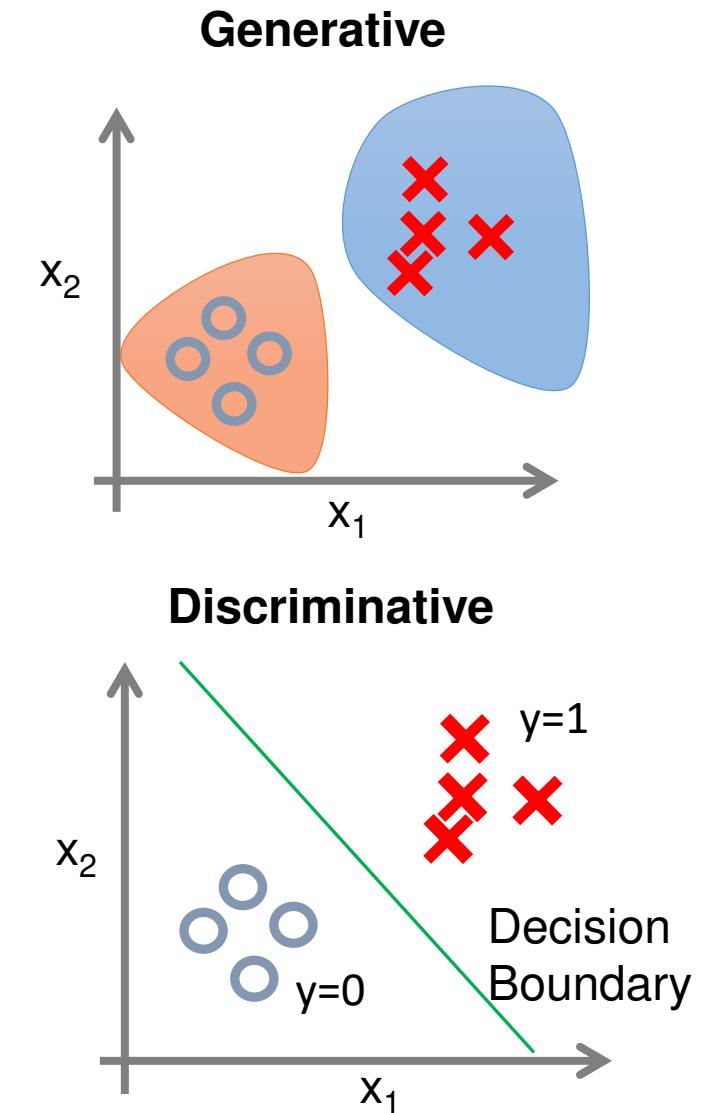
## Generative Model

- Class-conditional probability distribution of attribute/feature set and prior probability of classes are learnt during the training phase
- Given these learnt probabilities, during inferencing phase, probability of a test record belonging to different classes are calculated and compared.
- Can result in linear or nonlinear decision surface



## Discriminative Model

- Given a training set, a function  $f$  is learnt that directly maps an attribute/feature vector  $x$  to the output class ( $y=1$  or  $0/-1$ )
- A linear function  $f$  results in linear decision surface



# Bayes Rule

from the definition of conditional distributions:

$$P(A|B)P(B) = P(A, B) = P(B|A)P(A)$$

Hence:

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

is known as **Bayes rule**.

# Bayes Theorem

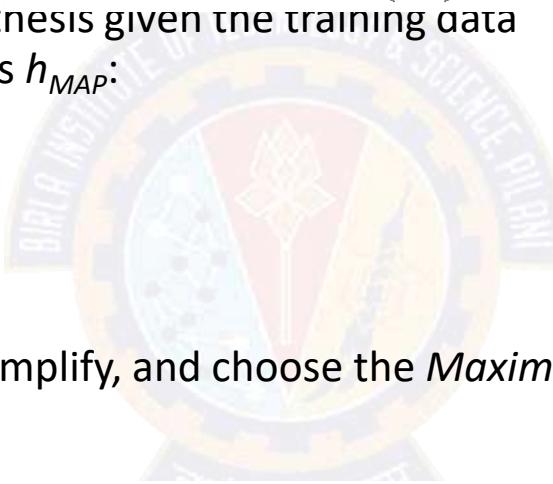
- $P(h)$  = prior probability of hypothesis  $h$
- $P(D)$  = prior probability of training data  $D$
- $P(h|D)$  = probability of  $h$  given  $D$
- $P(D|h)$  = probability of  $D$  given  $h$

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

# Choosing Hypotheses

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

- Generally want the most probable hypothesis given the training data
- *Maximum a posteriori* hypothesis  $h_{MAP}$ :



$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

- If assume  $P(h_i) = P(h_j)$  then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

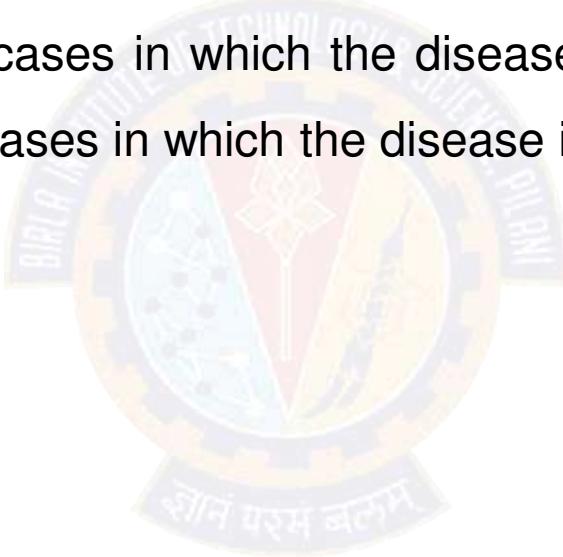
$$h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$$

# Example

- Consider a medical diagnosis problem in which there are two alternative hypotheses:
  - H1:That a patient has a particular form of cancer
  - H2:That the patient does not
- The available data is from a particular laboratory test with two possible outcomes
  - + Positive
  - Negative



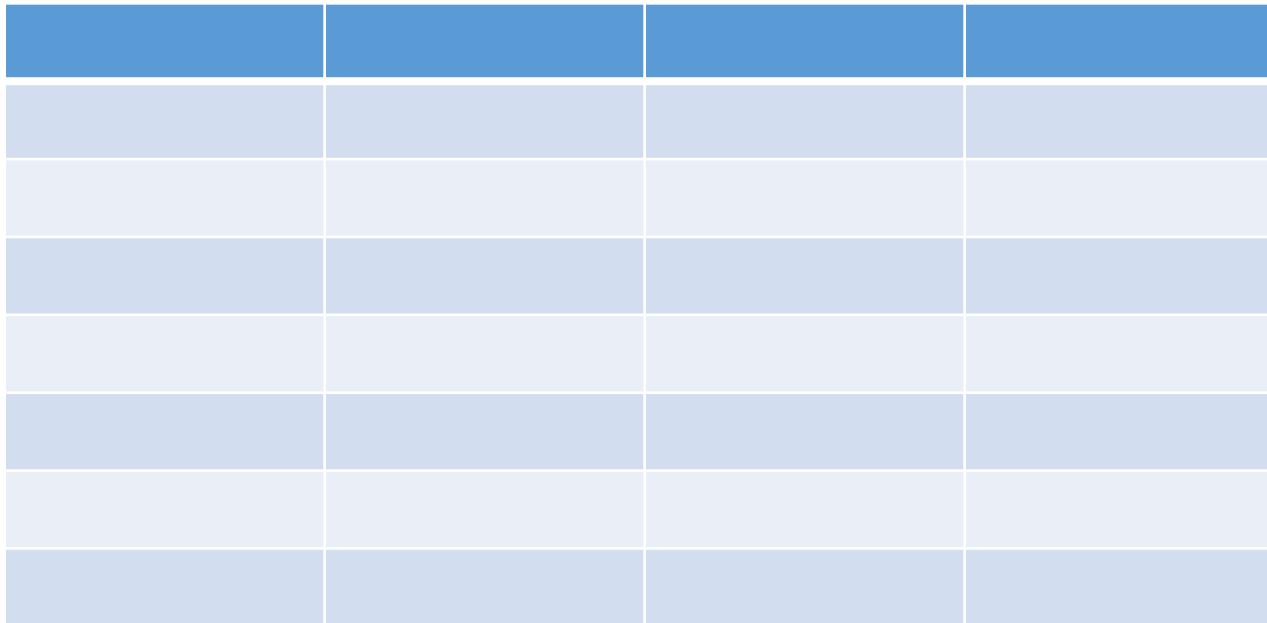
- Over the entire population of people only 0.008 have this disease. The test returns a corrective positive result in only 98% of the cases in which the disease is actually present and a correct negative result in only 97% of the cases in which the disease is not present.





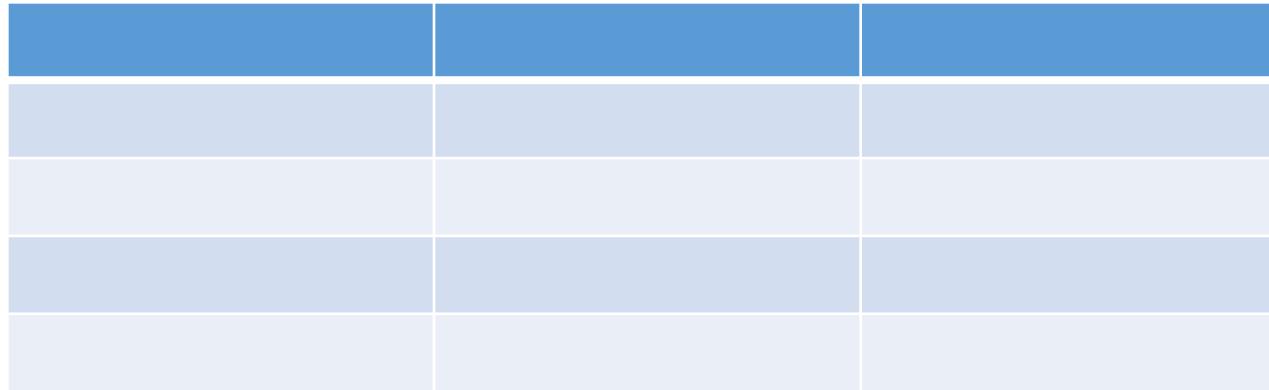












# Using Bayes Theorem for Classification

Consider each attribute and class label as random variables

- Given a record with attributes  $(X_1, X_2, \dots, X_d)$ 
  - Goal is to predict class  $Y$  ("Evade")
  - given  $(X_1, X_2, X_3) = (\text{Refund}, \text{Marital Status}, \text{Taxable Income})$
  - Specifically, we want to find the value of  $Y$  that maximizes  $P(Y | X_1, X_2, \dots, X_d)$
- Can we estimate  $P(Y | X_1, X_2, \dots, X_d)$  directly from data?

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

# Using Bayes Theorem for Classification

## Approach

- Compute posterior probability  $P(Y | X_1, X_2, \dots, X_d)$  using the Bayes theorem

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

- *Maximum a-posteriori*: Choose  $Y$  that maximizes  $P(Y | X_1, X_2, \dots, X_d)$
- Equivalent to choosing value of  $Y$  that maximizes  $P(X_1, X_2, \dots, X_d | Y) P(Y)$
- How to estimate  $P(X_1, X_2, \dots, X_d | Y)$ ?

# Example Data

## Given a Test Record

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

### Using Bayes Theorem:

$$P(\text{Yes} | X) = \frac{P(X | \text{Yes})P(\text{Yes})}{P(X)}$$

$$P(\text{No} | X) = \frac{P(X | \text{No})P(\text{No})}{P(X)}$$

How to estimate  $P(X | \text{Yes})$  and  $P(X | \text{No})$ ?

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

# Naïve Bayes on Example Data

Given a Test Record

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

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

$$P(\text{Refund} = \text{No} | \text{Yes}) \times$$

$$P(\text{Divorced} | \text{Yes}) \times$$

$$P(\text{Income} = 120\text{K} | \text{Yes})$$

$$P(X | \text{No}) =$$

$$P(\text{Refund} = \text{No} | \text{No}) \times$$

$$P(\text{Divorced} | \text{No}) \times$$

$$P(\text{Income} = 120\text{K} | \text{No})$$

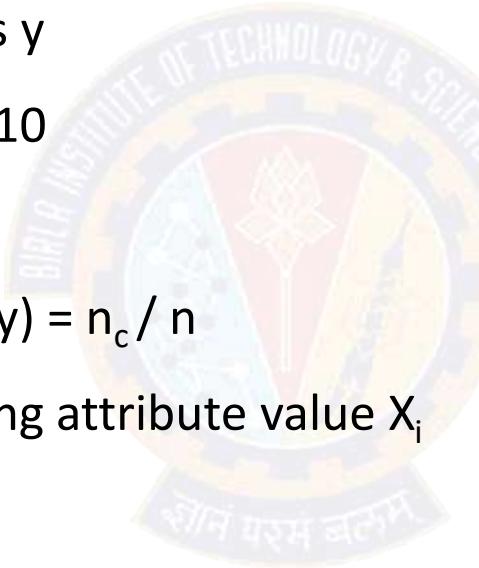


<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Estimate Probabilities from Data

## Discrete Attributes

- $P(y)$  = fraction of instances of class  $y$ 
  - e.g.,  $P(\text{No}) = 7/10$ ,  $P(\text{Yes}) = 3/10$
- For categorical attributes  $P(X_i = c \mid y) = n_c / n$ 
  - $n_c$  is number of instances having attribute value  $X_i = c$  and belonging to class  $y$
  - e.g.,  $P(\text{Status}=\text{Married} \mid y=\text{No}) = 4/7$ ,
  - $P(\text{Refund}=\text{Yes} \mid y=\text{Yes})=0$



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

# Example of Naïve Bayes Classifier

## Given a Test Record

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

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2975

If class = Yes: sample mean = 90

sample variance = 25

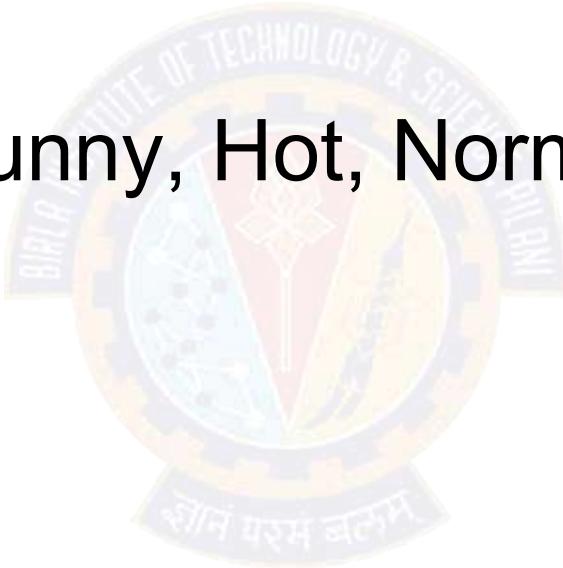
- $P(X \mid \text{No}) = P(\text{Refund}=\text{No} \mid \text{No}) \times P(\text{Divorced} \mid \text{No}) \times P(\text{Income}=120\text{K} \mid \text{No}) = 4/7 \times 1/7 \times 0.0072 = 0.0006$

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

- $P(X \mid \text{No})P(\text{No}) > P(X \mid \text{Yes})P(\text{Yes})$
- Therefore,  $P(\text{No} \mid X) > P(\text{Yes} \mid X) \Rightarrow \text{Class} = \text{No}$

S. No	Outlook	Temp	Humidity	Windy	Play Tennis
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

- today = (Sunny, Hot, Normal, False)



### Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

### Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

### Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

### Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
<b>Total</b>	<b>14</b>	<b>100%</b>

Magazine Promotion	Watch Promotion	Life Insurance Promotion	Credit Card Insurance	Sex
Yes	No	No	No	Male
Yes	Yes	Yes	Yes	Female
No	No	No	No	Male
Yes	Yes	Yes	Yes	Male
Yes	No	Yes	No	Female
No	No	No	No	Female
Yes	Yes	Yes	Yes	Male
No	No	No	No	Male
Yes	No	No	No	Male
Yes	Yes	Yes	No	Female

New Instance: Magazine Promotion = Yes, Watch Promotion = Yes,  
 Life Insurance Promotion = No, Credit Card Insurance = No then Sex = ?









# Example 4

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

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

**A: attributes**

**M: mammals**

**N: non-mammals**

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

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

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

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

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

**=> Mammals**

# Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

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

## Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/6$$

$$\rightarrow P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$$

$$\rightarrow P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

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!**

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

Fraction of documents in  $c$

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

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

Number of times  $t_i$  appears in some document in  $c$

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

Laplace Smoothing

Total number of terms in all documents in  $c$

Number of unique words (vocabulary size)

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

# A Simple Example

Text	Tag	Which tag does the sentence <i>A very close game</i> belong to? i.e. $P(\text{sports}   A \text{ very close game})$
“A great game”	Sports	Feature Engineering: Bag of words i.e use word frequencies without considering order
“The election was over”	Not sports	
“Very clean match”	Sports	Using Bayes Theorem:
“A clean but forgettable game”	Sports	$P(\text{sports}   A \text{ very close game})$ $= \frac{P(A \text{ very close game}   \text{sports}) P(\text{sports})}{P(A \text{ very close game})}$
“It was a close election”	Not sports	

We assume that every word in a sentence is **independent** of the other ones

“close” doesn’t appear in sentences of sports tag, So  $P(\text{close} | \text{sports}) = 0$ , which makes product 0

# 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'].

# 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

Doc No	Text
1	I LOVED THE MOVIE
2	I HATED THE MOVIE
3	A GREAT MOVIE ,GOOD MOVIE
4	POOR ACTING
5	GREAT ACTING , A GOOD MOVIE
NEW	I HATED THE POOR ACTING

# Example

Doc No	Text	
1	I LOVED THE MOVIE	<u>POSITIVE</u>
2	I HATED THE MOVIE	<u>NEGATIVE</u>
3	A GREAT MOVIE ,GOOD MOVIE	<u>POSITIVE</u>
4	POOR ACTING	<u>NEGATIVE</u>
5	GREAT ACTING , A GOOD MOVIE	<u>POSITIVE</u>
NEW	I HATED THE POOR ACTING	<u>????</u>











1	This is my book	Statement
2	They are novels	Statement
3	Have you read this book	Question
4	Who is the author	Question
5	What are the characters	Question
6	This is how I bought the book	Statement
7	I like fictions	Statement
8	What is your favourite book	Question























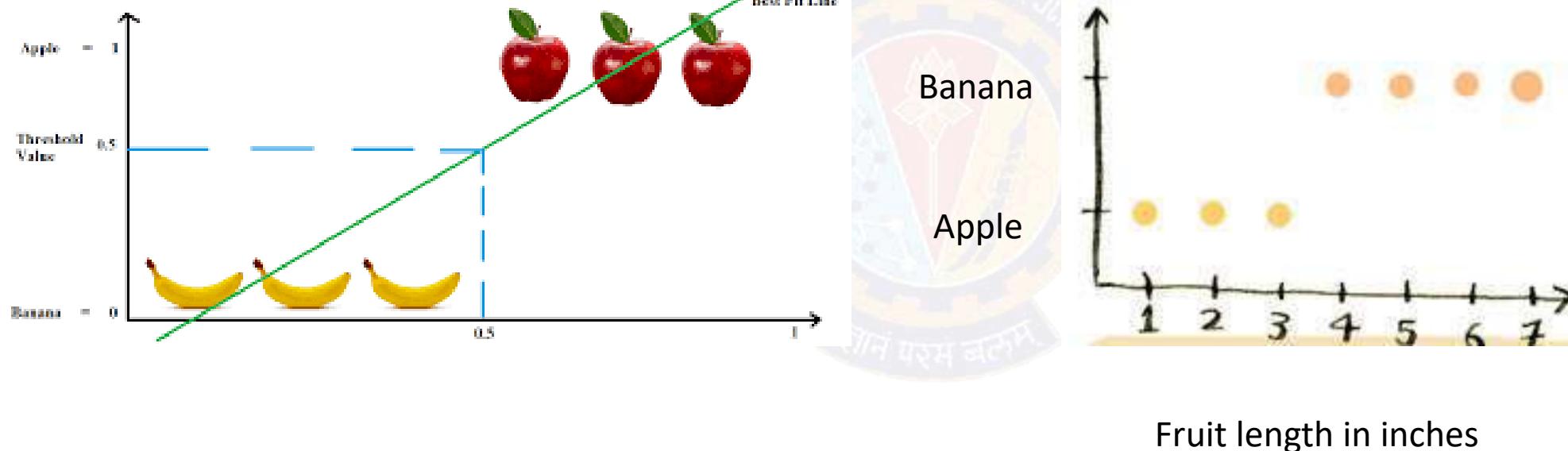


# Logistic regression

- **Logistic Regression** could help use predict, for example, whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model's classifications.
- In comparison, Linear Regression could help us predict the student's test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).
- Takes a probabilistic approach for learning discriminative functions
- **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

Similarly, when you are trying to classify something, first you plot your data on a graph. Suppose you are trying to classify a new piece of fruit as an apple or a banana. Here's your existing data:



i.e. anything that is less than 4 inches in length is an apple, anything greater than that is a banana.

# Logistic regression

- Remember, **the prediction function for logistic regression spits out a percentage.**
- As in "we are 95% sure this is a Banana".
- The  $g(\dots)$  is called the **sigmoid** function.
- This function is what makes the prediction function output a percentage.
- **The sigmoid function constrains the result to between 0 and 1.**
- So a result of .95 would mean we are 95% certain of something.
- That's all you really need to know about the sigmoid function.
- The sigmoid function is:

$$h_{\theta}(x) = g(\theta^T x)$$

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

where,  $x = y_{\text{hat}}$

# Sigmoid/Logistic Function

Hence, the equation becomes:

$$\begin{aligned} f(\hat{y}) &= \frac{1}{1 + e^{-\hat{y}}} \\ &= \frac{1}{1 + e^{-m^*X}} \end{aligned}$$



So, the final hypothesis for Logistic Regression becomes:

$$\hat{y} = \text{sigmoid}(m^*X)$$

# Sigmoid/Logistic Function

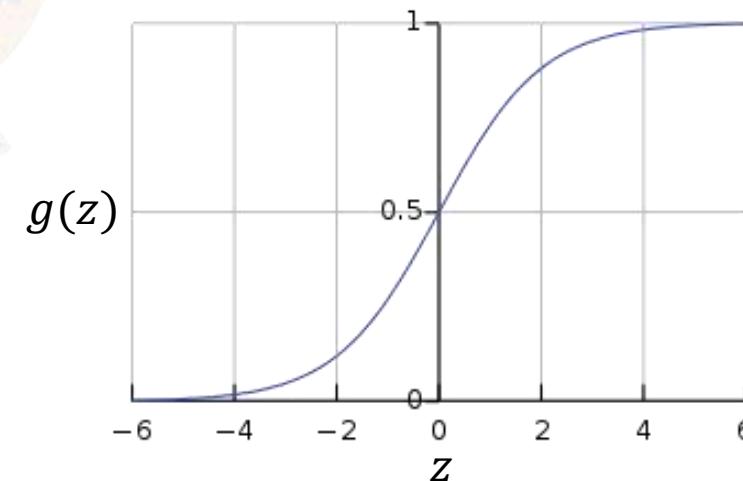
## Classification requires discrete output values

- For example, output  $y = 0$  or  $1$  for a two-category classification problem
- In logistic regression, sigmoid/logistic function  $h_\theta(x)$  takes a real vector  $x$  as input and outputs a value between  $0$  and  $1$

$$h_\theta(x) = g(\theta^T x)$$

- $\Theta$  is a vector of logistic regression parameters

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



# Cost function for Logistic Regression

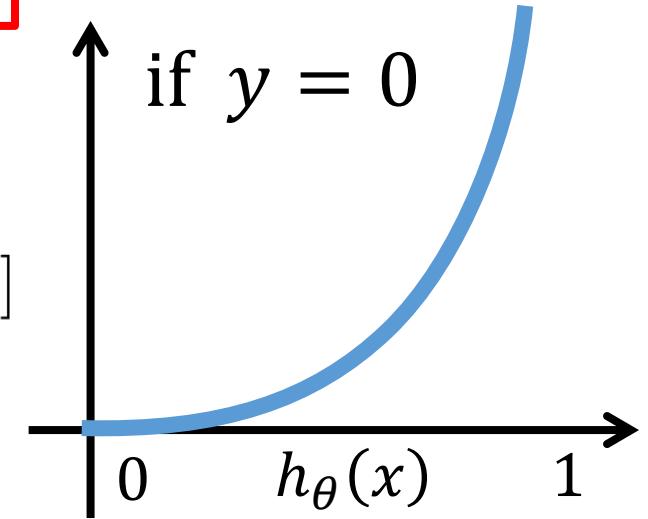
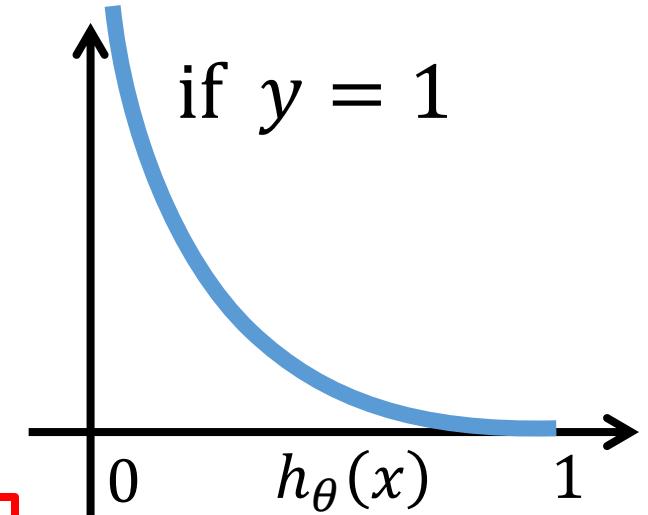
## Cross Entropy or Log Loss Function

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

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

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

- $J(\theta)$  is convex
- Apply gradient descent on  $J(\theta)$  w.r.t.  $\theta$  to find optimal parameters



# Gradient descent

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

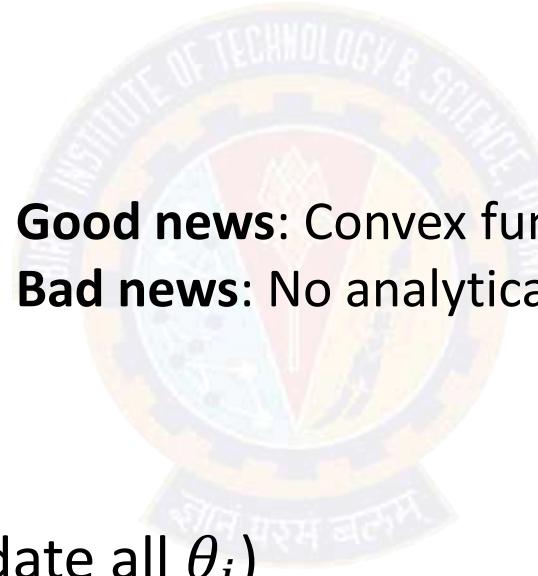
Goal:  $\min_{\theta} J(\theta)$

Repeat {

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

}

(Simultaneously update all  $\theta_j$ )



**Good news:** Convex function!

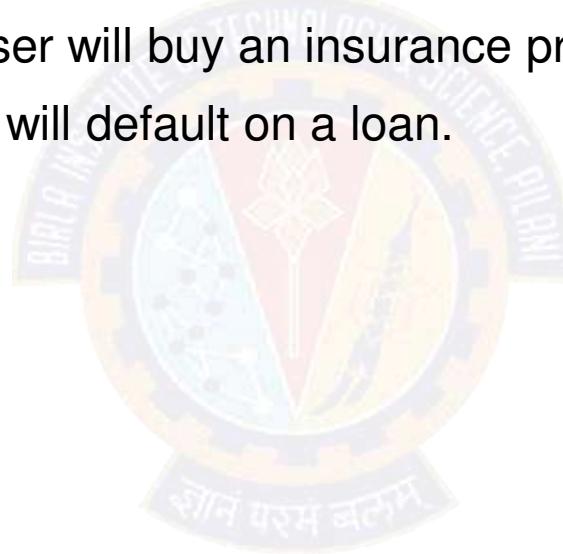
**Bad news:** No analytical solution

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

Slide credit: Andrew Ng

# Logistic Regression Applications

- **Credit Card Fraud** : Predicting if a given credit card transaction is fraud or not
- **Health** : Predicting if a given mass of tissue is benign or malignant
- **Marketing** : Predicting if a given user will buy an insurance product or not
- **Banking** : Predicting if a customer will default on a loan.





**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)

## Topics

- Naive Bayes Classifier
- Logistic Regression
- Linear Support Vector Machines
- Non Linear Support Vector Machines
- Comparative Analysis and Applicability



7	Classification Models I: Support Vector Machine. Margin maximization. Non-linear SVM. Kernel Function.	T1: Chapter 5 T2: Chapter 5
---	--	--------------------------------

# A Simple Example

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

Which tag does the sentence *A very close game* belong to? i.e.  $P(\text{sports} | \text{A very close game})$

Feature Engineering: Bag of words i.e use word frequencies without considering order

Using Bayes Theorem:

$$P(\text{sports} | \text{A very close game})$$

$$= \frac{P(\text{A very close game} | \text{sports}) P(\text{sports})}{P(\text{A very close game})}$$

We assume that every word in a sentence is **independent** of the other ones

“close” doesn’t appear in sentences of sports tag, So  $P(\text{close} | \text{sports}) = 0$ , which makes product 0

# 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'].

# 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

Doc No	Text
1	I LOVED THE MOVIE
2	I HATED THE MOVIE
3	A GREAT MOVIE ,GOOD MOVIE
4	POOR ACTING
5	GREAT ACTING , A GOOD MOVIE
NEW	I HATED THE POOR ACTING

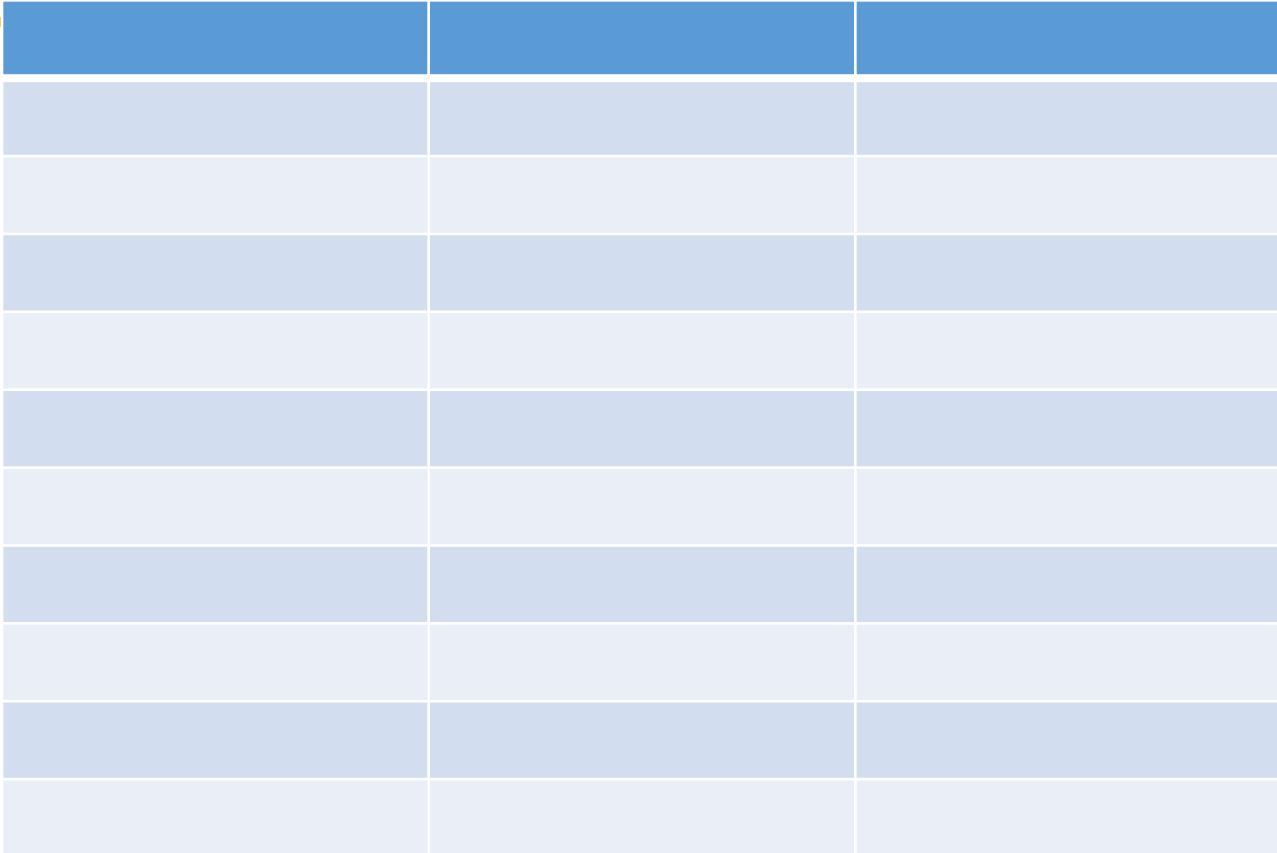
# Example



Doc No	Text	
1	I LOVED THE MOVIE	<u>POSITIVE</u>
2	I HATED THE MOVIE	<u>NEGATIVE</u>
3	A GREAT MOVIE ,GOOD MOVIE	<u>POSITIVE</u>
4	POOR ACTING	<u>NEGATIVE</u>
5	GREAT ACTING , A GOOD MOVIE	<u>POSITIVE</u>
NEW	I HATED THE POOR ACTING	<u>????</u>











---

# Logistic Regression

A circular watermark logo of the University of Delhi is centered on the slide. It features the university's name in English, "UNIVERSITY OF DELHI", and its name in Hindi, "दिल्ली विश्वविद्यालय", around the perimeter. The center of the logo contains a Sanskrit motto, "शान्तिः परमं वैत्यनः", which translates to "Shanti (Peace) is the highest goal".

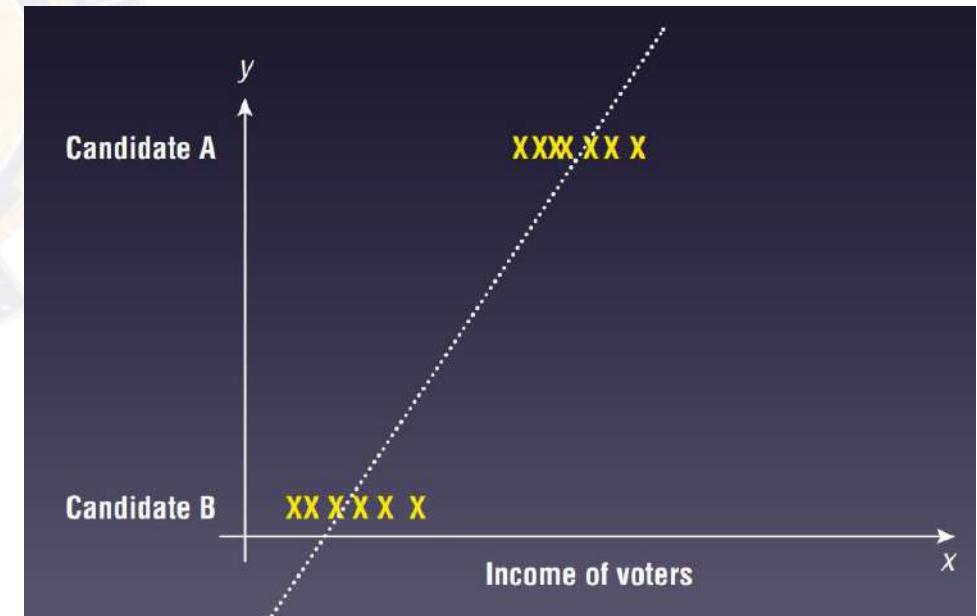
## Classification Example

- Consider the example shown in Fig.
- Suppose that you have a dataset containing information about voter income and voting preferences.
- For this dataset, you can see that low-income voters tend to vote for candidate B, while high-income voters tend to favor candidate A.
- We would be very interested in trying to predict which candidate future voters will vote for based on their income level.



## Classification Example

- At first glance, we might be tempted to solve this problem using linear regression.
- Fig shows what it looks like when you apply linear regression to this problem.
- With linear regression, the predicted value does not always fall within the expected range.
- Consider the case of a very low-income voter (near to 0)
  - We can see from the chart that the predicted result is a negative value.
- We actually want the prediction as a value from 0 to 1 (which is the probability of an event happening).



## Overview

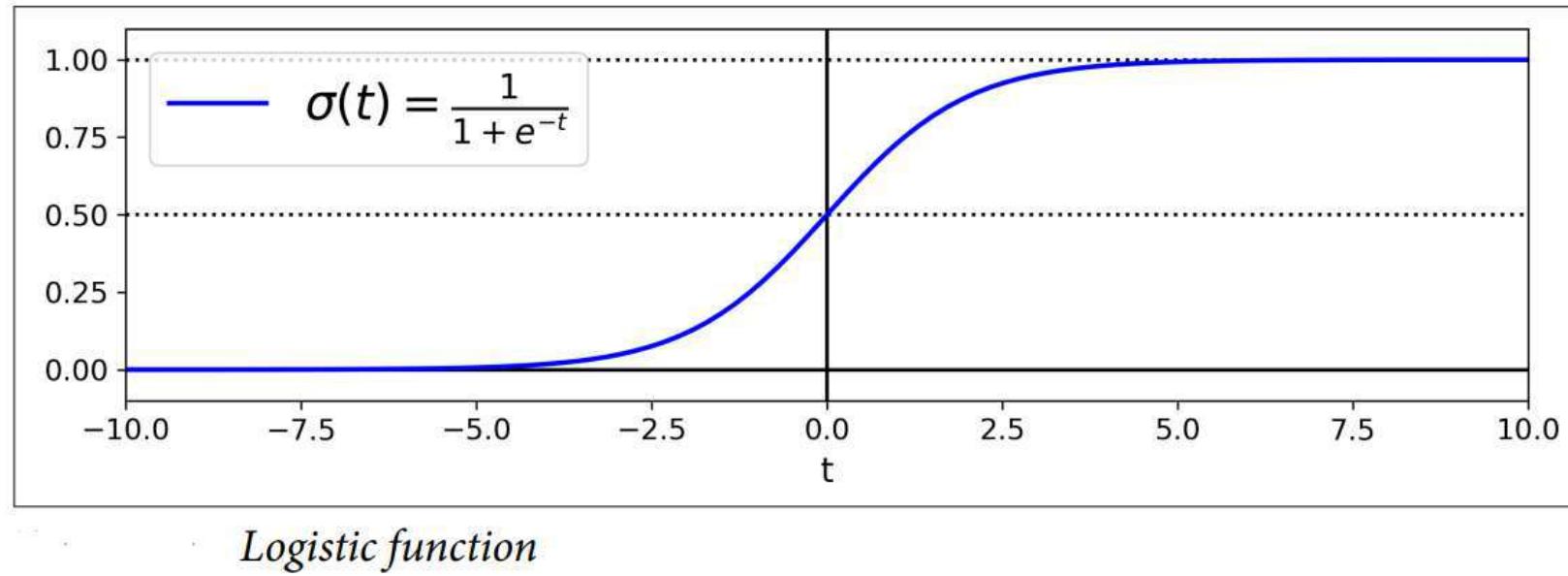
- Logistic Regression (also called Logit Regression)
- Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class
  - E.g., what is the probability that this email is spam?
- Estimate the probability
  - If greater than 50%,
    - then the model predicts that the instance belongs to that class (called the positive class, labeled “1”),
  - else
    - it predicts that it does not (i.e., it belongs to the negative class, labeled “0”).
- This makes logistic regression a binary classifier.

# Logistic Regression

## Sigmoid or Logistic Function

$$g(z) = \frac{1}{1 + e^{-z}} \quad 0 < g(z) < 1$$

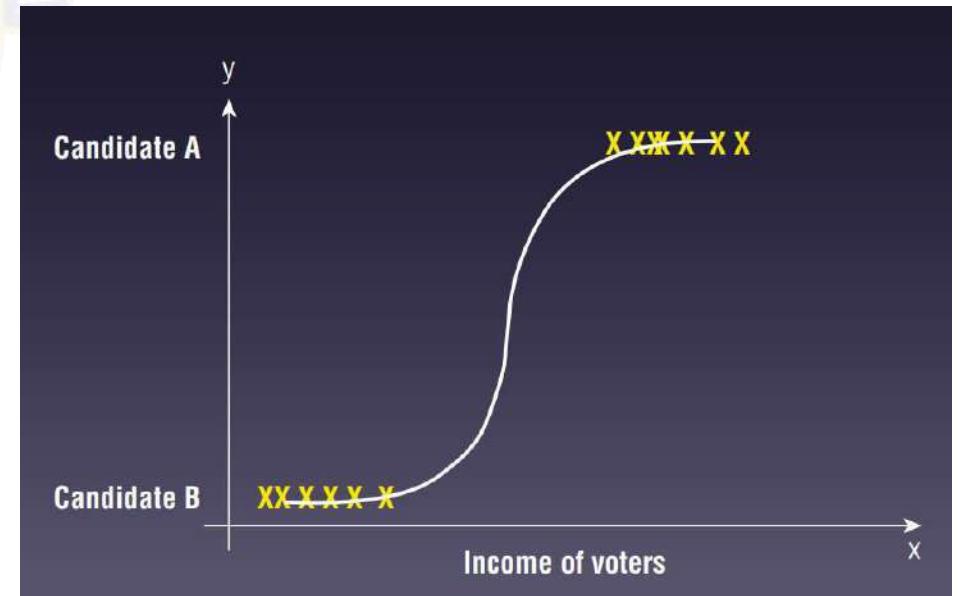
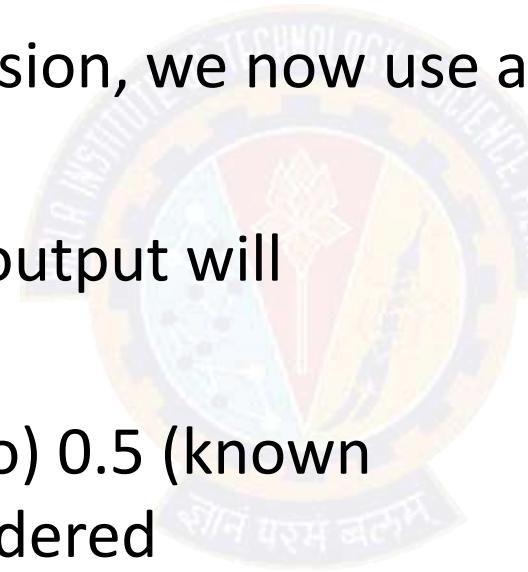
- Where  $e$  is the mathematical constant that takes on value 2.7



- Let's say  $z = 100$
- $e^{-z} = e^{-100}$  becomes a very tiny number, so  $g(z)$  gets very close to  $\frac{1}{1+tiny\ number} \approx 1$
- Let  $z = -100$
- $e^{-z} = e^{100}$  becomes a very large number, so  $g(z)$  gets very close to  $\frac{1}{1+big\ number} \approx 0$
- Let's say  $z = 0$
- $e^{-z} = e^{-0} = 1$
- So  $g(z) = \frac{1}{1+1} = 0.5$

## Applying Logistic Regression

- Logistic regression can solve this problem.
- Instead of using Linear Regression, we now use a curved line to try to fit all of the points on the chart.
- Using logistic regression, the output will be a value from 0 to 1
- Anything less than (or equal to) 0.5 (known as the *threshold*) will be considered as voting for candidate B, and anything greater than 0.5 will be considered as voting for candidate A.



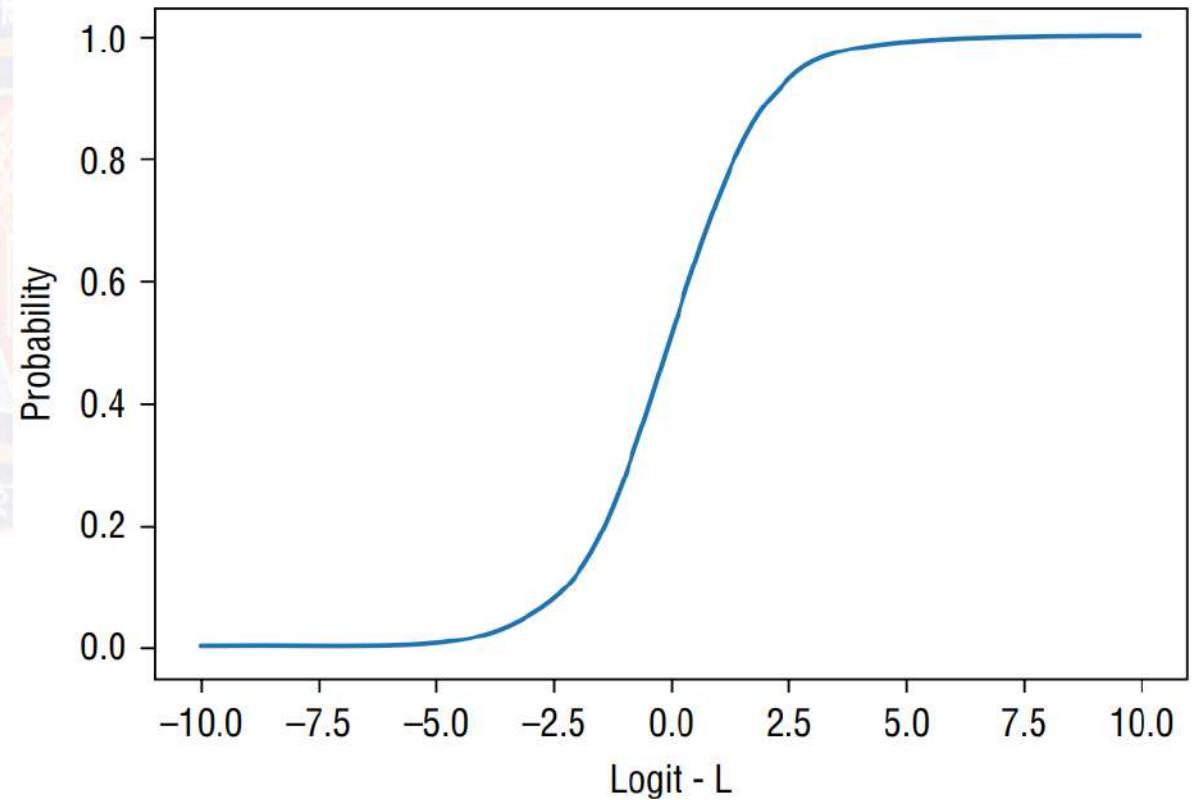
## Plotting Sigmoid Curve

- The following code snippet shows how the sigmoid curve is obtained:

```
def sigmoid(x):  
    return (1 / (1 + np.exp(-x)))  
  
x = np.arange(-10, 10, 0.0001)  
y = [sigmoid(n) for n in x]  
plt.plot(x,y)  
plt.xlabel("Logit - L")  
plt.ylabel("Probability")
```



- Fig shows the sigmoid curve.



## The Algorithm

- Now, let's use this background to build up the Logistic Regression algorithm. We will do this in two steps:
  - 1) We use the Linear Regression function  $f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ 
    - Let's call the value  $(\vec{w} \cdot \vec{x} + b) = z$
  - 2) Take the value of  $z$  and pass it to the sigmoid function
    - $g(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$
    - This function  $g(z)$  outputs the value between 0 and 1.
- The parameters  $\vec{w}$  and  $b$  are unknown, and they must be estimated based on the available training data using a technique known as Maximum Likelihood Estimation (MLE).
- In logistics regression,  $b$  is known as the intercept and  $\vec{w}$  is the weight vector corresponding to the feature vector  $\vec{x}$ .

## The Algorithm

- Combining these two steps, we get the logistic regression model
  - $$f_{\vec{w}, b}(\vec{x}) = g(z) = g(\vec{w} \cdot \vec{x} + b) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$
- Basically, the model takes input features  $X$  and outputs a number between 0 and 1

## Breast Cancer Dataset

- Scikit-learn includes the Breast Cancer Wisconsin (Diagnostic) Data Set.
- It is a popular dataset often used for illustrating binary classifications.
- The dataset contains 569 samples and 30 features
- These features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
- The label (outcome) of the dataset is a binary classification—M for malignant or B for benign.
- For more information:
  - [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

## Examining the Relationship Between Features

- First, load the Breast Cancer dataset by first importing the datasets module from sklearn.
- Then use the load \_ breast \_ cancer() function as follows:

- `from sklearn.datasets import load_breast_cancer`
- `cancer = load_breast_cancer()`

## Plotting the Features in 2D

- Let's plot the first two features of the dataset in 2D and examine their relationships.
- The following code snippet:
  - Loads the Breast Cancer dataset
  - Copies the first two features of the dataset into a two-dimensional list
  - Plots a scatter plot showing the distribution of points for the two features
  - Displays malignant growths in red and benign growths in blue

# Logistic Regression Application

innovate

achieve

lead

## Plotting the Features in 2D

```
1 %matplotlib inline
2
3 import matplotlib.pyplot as plt
4 from sklearn.datasets import load_breast_cancer
5
6 cancer = load_breast_cancer()
7
8 #---copy from dataset into a 2-d list---
9 X = []
10 for target in range(2):
11     X.append([[], []])
12     for i in range(len(cancer.data)):           # target is 0 or 1
13         if cancer.target[i] == target:
14             X[target][0].append(cancer.data[i][0]) # first feature - mean radius
15             X[target][1].append(cancer.data[i][1]) # second feature - mean texture
16
17 colours = ("r", "b")    # r: malignant, b: benign
18 fig = plt.figure(figsize=(10,8))
19 ax = fig.add_subplot(111)
20 for target in range(2):
21     ax.scatter(X[target][0],
22                 X[target][1],
23                 c=colours[target])
24
25 ax.set_xlabel("mean radius")
26 ax.set_ylabel("mean texture")
27 plt.show()
```

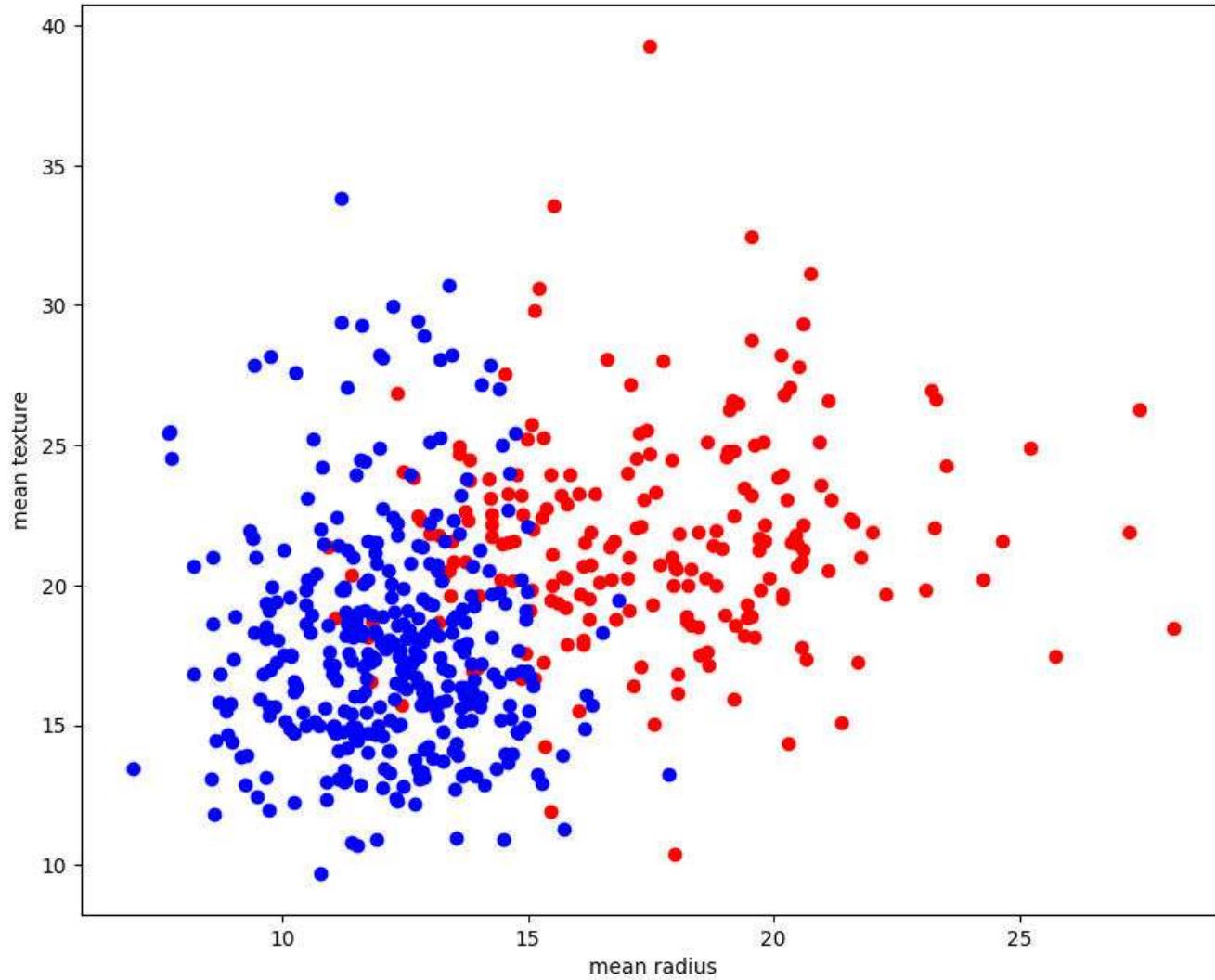
# Logistic Regression Application

innovate

achieve

lead

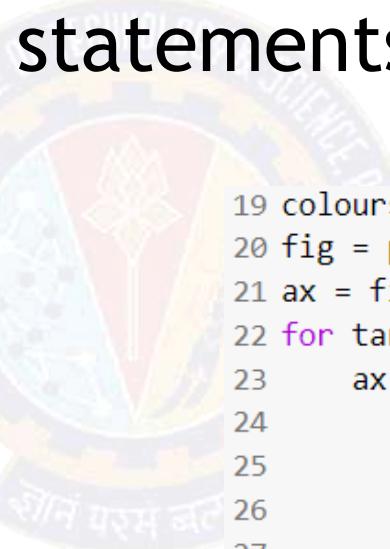
## Plotting the Features in 2D



## Plotting the Features in 3D

- The following code snippet is very similar to the 2D code snippet, with the additional statements in bold:

```
1 %matplotlib inline
2
3 import matplotlib.pyplot as plt
4 from mpl_toolkits.mplot3d import Axes3D
5 from sklearn.datasets import load_breast_cancer
6
7 cancer = load_breast_cancer()
8
9 #---copy from dataset into a 2-d array---
10 X = []
11 for target in range(2):
12     X.append([[], [], []])
13     for i in range(len(cancer.data)):    # target is 0,1
14         if cancer.target[i] == target:
15             X[target][0].append(cancer.data[i][0])
16             X[target][1].append(cancer.data[i][1])
17             X[target][2].append(cancer.data[i][2])
18
```

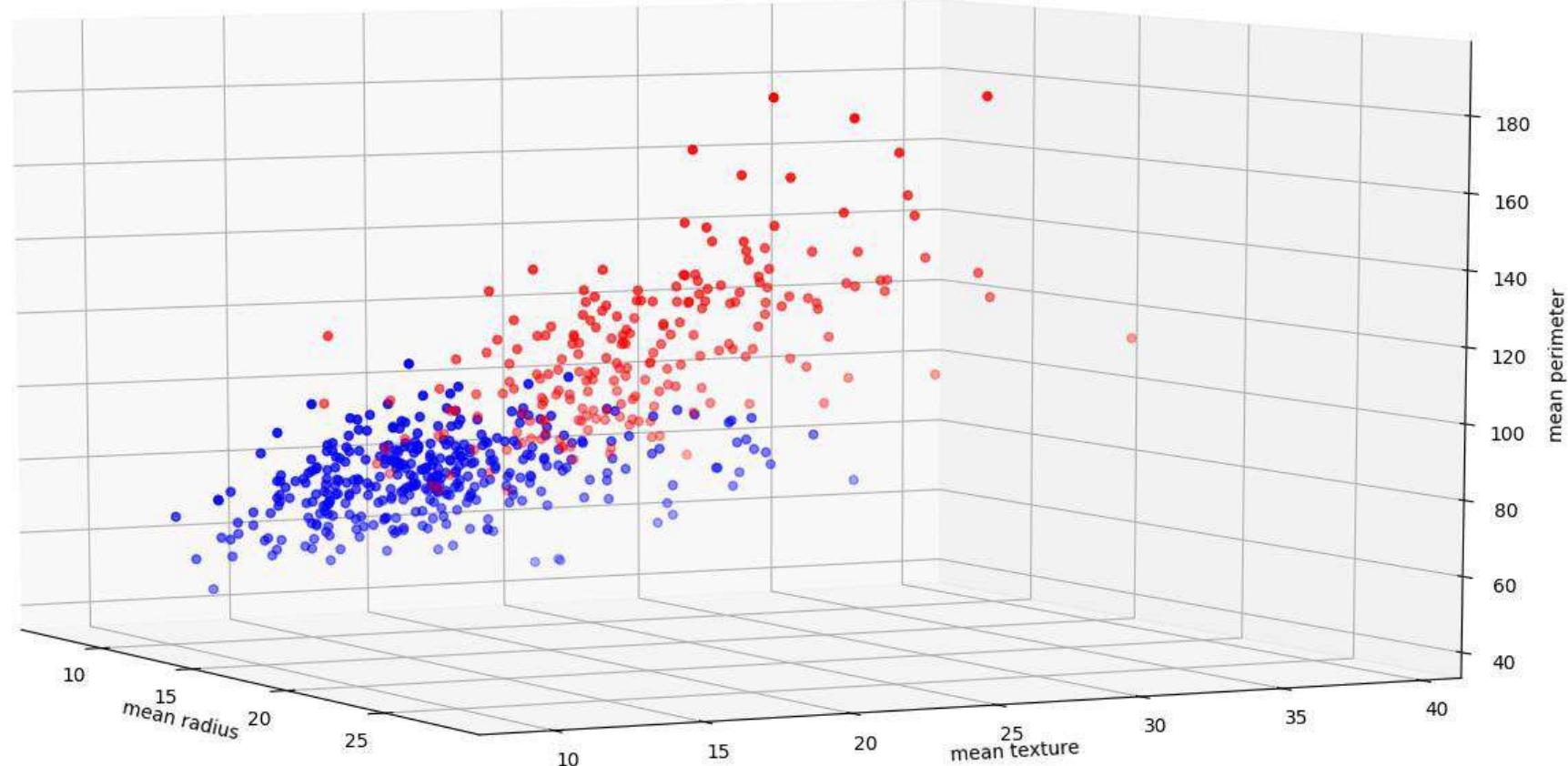


```
19 colours = ("r", "b")    # r: malignant, b: benign
20 fig = plt.figure(figsize=(18,15))
21 ax = fig.add_subplot(111, projection='3d')
22 for target in range(2):
23     ax.scatter(X[target][0],
24                 X[target][1],
25                 X[target][2],
26                 c=colours[target])
27
28 ax.set_xlabel("mean radius")
29 ax.set_ylabel("mean texture")
30 ax.set_zlabel("mean perimeter")
31 plt.show()
```

# Logistic Regression Application



## Plotting the Features in 3D



## Features of the Dataset

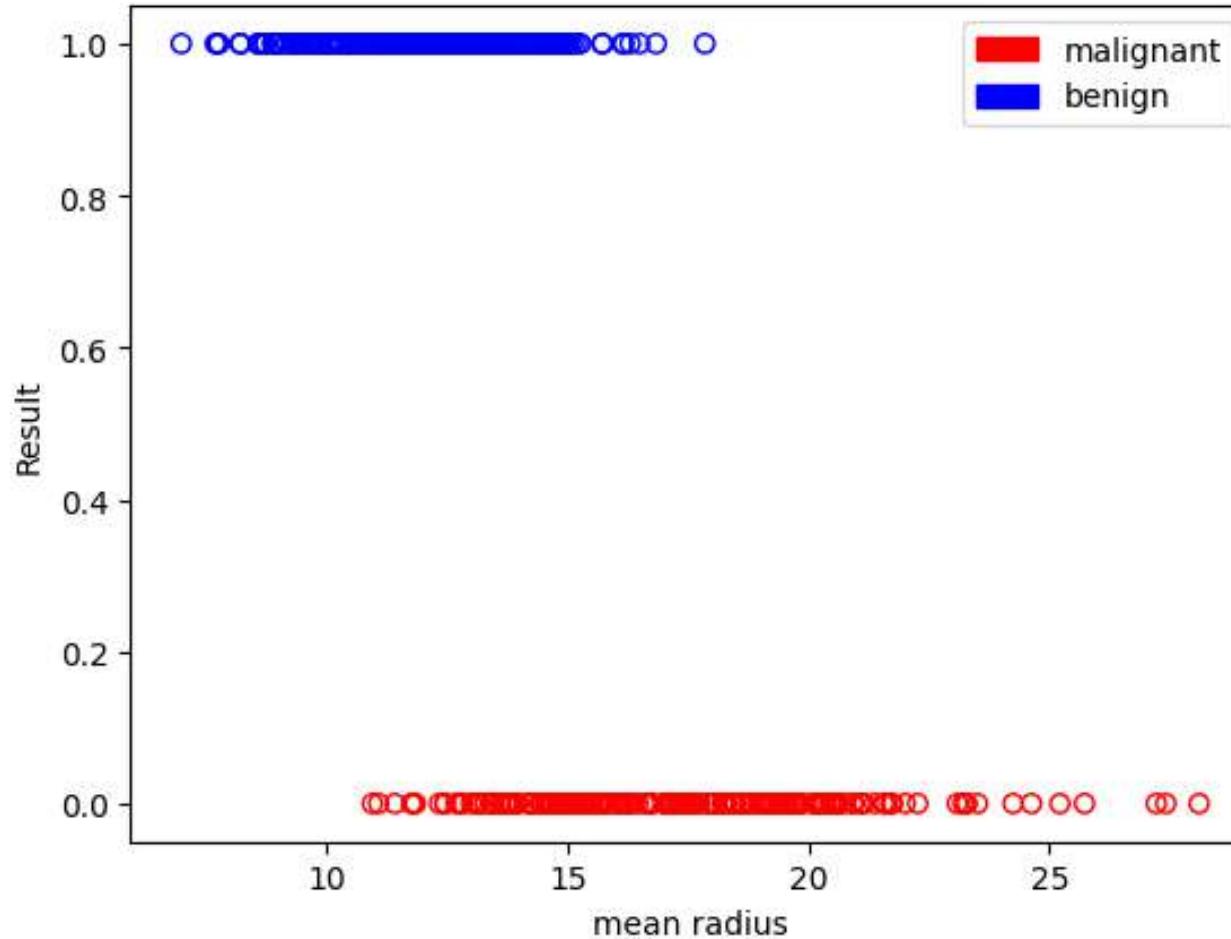
- Running the following command displays the list of features of this dataset.
- `>>>cancer.feature_names`
- `>>>array(['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness', 'mean compactness', 'mean concavity', 'mean concave points', 'mean symmetry', 'mean fractal dimension', 'radius error', 'texture error', 'perimeter error', 'area error', 'smoothness error', 'compactness error', 'concavity error', 'concave points error', 'symmetry error', 'fractal dimension error', 'worst radius', 'worst texture', 'worst perimeter', 'worst area', 'worst smoothness', 'worst compactness', 'worst concavity', 'worst concave points', 'worst symmetry', 'worst fractal dimension'], dtype='<U23')`

## Training Using One Feature

- Now, let's use logistic regression to predict if a tumor is cancerous.
- We will use only the first feature ('[mean radius](#)') of the dataset.
- The code snippet plots a scatter plot showing if a tumor is malignant or benign based on the mean radius:

```
1 %matplotlib inline
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import matplotlib.patches as mpatches
5
6 from sklearn.datasets import load_breast_cancer
7
8 cancer = load_breast_cancer()      # Load dataset
9 x = cancer.data[:,0]               # mean radius
10 y = cancer.target                # 0: malignant, 1: benign
11 colors = {0:'red', 1:'blue'}      # 0: malignant, 1: benign
12
13 plt.scatter(x,y,
14             facecolors='none',
15             edgecolors=pd.DataFrame(cancer.target)[0].apply(lambda x: colors[x]),
16             cmap=colors)
17
18 plt.xlabel("mean radius")
19 plt.ylabel("Result")
20
21 red  = mpatches.Patch(color='red',  label='malignant')
22 blue = mpatches.Patch(color='blue', label='benign')
23
24 plt.legend(handles=[red, blue], loc=1)
```

## Training Using One Feature



## Finding the Parameters ( $w$ and $b$ )

- Scikit-learn comes with the LogisticRegression class that allows you to apply logistic regression to train a model.
- In this example, we train a model using the first feature of the dataset.
- Once the model is trained, we get the parameters ( $w$  and  $b$ ).
- Knowing these two values allows us to plot the sigmoid curve that tries to fit the points on the chart.

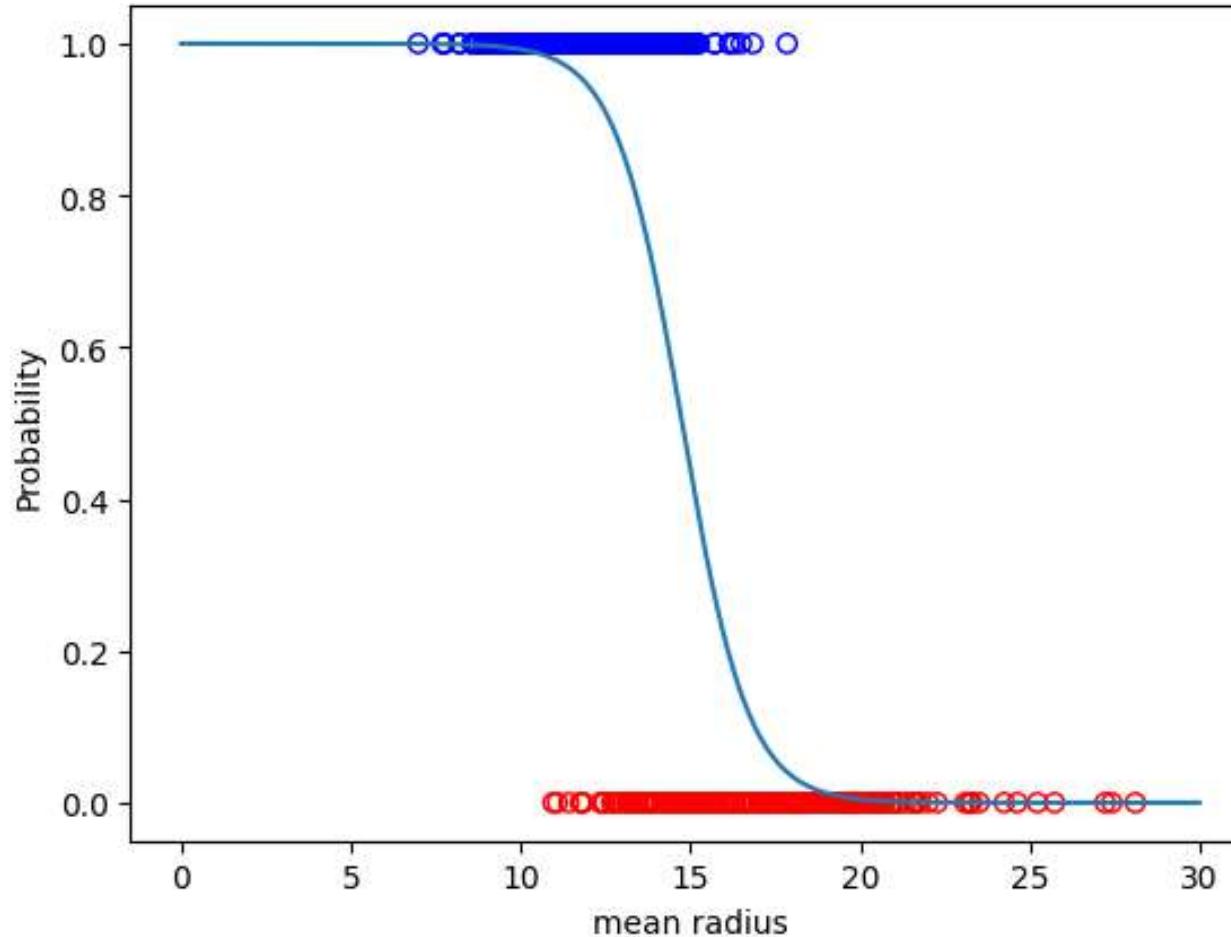
```
1 from sklearn import linear_model
2 import numpy as np
3
4 log_regress = linear_model.LogisticRegression()
5
6 #---train the model---
7 log_regress.fit(X = np.array(x).reshape(len(x),1),
8                  y = y)
9
10
11 #---print trained model intercept---
12 print(log_regress.intercept_)      # [ 8.19393897]
13
14 #---print trained model coefficients---
15 print(log_regress.coef_)          # [[-0.54291739]]
16
```

## Plotting the Sigmoid Curve

- With the values of  $w$  and  $b$  obtained, we can plot the sigmoid curve using the following code:

```
1 def sigmoid(x):  
2     return (1 / (1 +  
3         np.exp(-(log_regress.intercept_[0] +  
4             (log_regress.coef_[0][0] * x)))))  
5  
6 x1 = np.arange(0, 30, 0.01)  
7 y1 = [sigmoid(n) for n in x1]  
8  
9 plt.scatter(x,y,  
10     facecolors='none',  
11     edgecolors=pd.DataFrame(cancer.target)[0].apply(lambda x: colors[x]),  
12     cmap=colors)  
13  
14 plt.plot(x1,y1)  
15 plt.xlabel("mean radius")  
16 plt.ylabel("Probability")  
17
```

## Plotting the Sigmoid Curve



## Making Predictions

- Let's try to predict the result if the mean radius is 18:

```
test = np.array([18]).reshape(1,-1)
print(log_regress.predict_proba(test)) # [[0.96526677 0.03473323]]
print(log_regress.predict(test)[0])    # 0
```

- The `predict_proba()` function in the first statement returns a two-dimensional array.
- The result of `0.96526677` indicates the probability that the prediction is `0 (malignant)` while the result of `0.03473323` indicates the probability that the prediction is `1 (benign)`.
- Based on the default threshold of 0.5, the prediction is that the tumor is `malignant (value of 0)`, since its predicted probability (`0.96526677`) of 0 is more than 0.5.

## Making Predictions

- The `predict()` function in the second statement returns the class that the result lies in (which in this case can be a 0 or 1).
- The result of 0 indicates that the prediction is that the tumor is malignant.
- Try another example with the mean radius of 15, 8, and 25 and observe the result.

```
test = np.array([8]).reshape(1,-1)
print(log_regress.predict_proba(test)) # [[9.84046071e-04 9.99015954e-01]]
print(log_regress.predict(test)[0]) # 1
```

## Training the Model Using All Features

- Let's now try to train the model using all of the features and then see how well it can accurately perform the prediction.
- First, load the dataset:
  - `from sklearn.datasets import load_breast_cancer`
  - `cancer = load_breast_cancer() # Load dataset`
- Instead of training the model using all of the rows in the dataset, we are going to split it into two sets, one for training and one for testing, using `train_test_split()` function.
- This function splits the data into training and test subsets randomly.
- The following code snippet splits the dataset into a 75 percent training and 25 percent testing set:

## Training the Model Using All Features

```
1 from sklearn.model_selection import train_test_split
2 train_set, test_set, train_labels, test_labels = train_test_split(
3                                     cancer.data,                      # features
4                                     cancer.target,                     # labels
5                                     test_size = 0.25,                  # split ratio
6                                     random_state = 1,                # set random seed
7                                     stratify = cancer.target)       # randomize based on labels
8
```

- Figure shows how the dataset is split.



## Training the Model Using All Features

```
1 from sklearn.model_selection import train_test_split
2 train_set, test_set, train_labels, test_labels = train_test_split(
3             cancer.data,                      # features
4             cancer.target,                    # labels
5             test_size = 0.25,                 # split ratio
6             random_state = 1,                # set random seed
7             stratify = cancer.target)      # randomize based on labels
8
```

- The random \_ state parameter of the train \_ test \_ split() function specifies the seed used by the random number generator.
- If this is not specified, every time you run this function you will get a different training and testing set.
- The stratify parameter allows you to specify which column (feature/label) to use so that the split is proportionate.
  - For example, if the column specified is a categorical variable with 80 percent 0s and 20 percent 1s, then the training and test sets would each have 80 percent of 0s and 20 percent of 1s.

## Training the Model Using All Features

- Once the dataset is split, it is now time to train the model.
- The following code snippet trains the model using logistic regression:

```
1 from sklearn import linear_model
2 x = train_set[:,0:30]                      # mean radius
3 y = train_labels                             # 0: malignant, 1: benign
4 log_regress = linear_model.LogisticRegression()
5 log_regress.fit(x = x,
6                  y = y)
7
```

## Training the Model Using All Features

- In this example, we are training it with all of the 30 features in the dataset.
- When the training is done, let's print out the parameters  $b$  and  $w$ :

```
1 print(log_regress.intercept_)      #
2 print(log_regress.coef_)          #
3

[0.10277949]
[[ 0.5844612   0.52342016   0.51305013  -0.03121987  -0.01611099  -0.09080365
  -0.12870478  -0.05096804  -0.02957876  -0.00430479   0.02718268   0.13771429
   0.02726314  -0.11054188  -0.00142629  -0.02146372  -0.02938366  -0.00683036
  -0.00555795  -0.00205115   0.67665368  -0.57068549  -0.30919133  -0.00672371
  -0.03257438  -0.30727354  -0.38677455  -0.10422661  -0.08803816  -0.02870834]]
```

- Because we have trained the model using 30 features, there are 30 coefficients.

## Testing the Model

- The following code snippet uses the test set and feeds it into the model for making predictions.

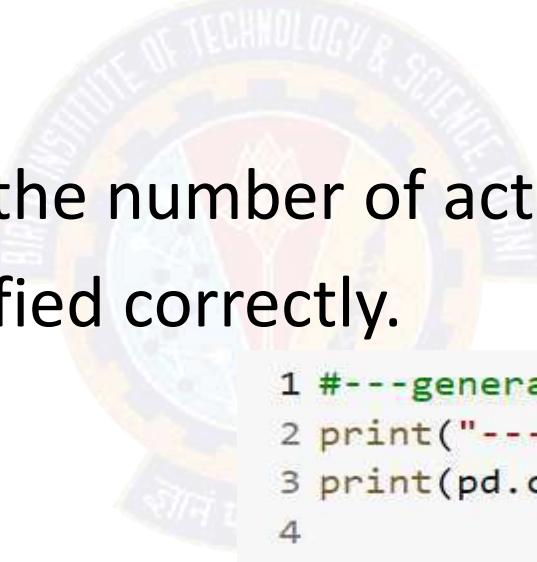
- The Output**

	Malignant	Benign	Prediction	Original	Result
0	0.999837	1.630691e-04	0	0	0
1	0.999748	2.518486e-04	0	0	0
2	0.173488	8.265124e-01	1	1	1
3	1.000000	5.658321e-09	0	0	0
4	0.089217	9.107834e-01	1	0	0

```
1 import pandas as pd
2
3 #---get the predicted probabilities and convert into a dataframe---
4 preds_prob = pd.DataFrame(log_regress.predict_proba(X=test_set))
5
6 #---assign column names to prediction---
7 preds_prob.columns = ["Malignant", "Benign"]
8
9 #---get the predicted class labels---
10 preds = log_regress.predict(X=test_set)
11 preds_class = pd.DataFrame(preds)
12 preds_class.columns = ["Prediction"]
13
14 #---actual diagnosis---
15 original_result = pd.DataFrame(test_labels)
16 original_result.columns = ["Original Result"]
17
18 #---merge the three dataframes into one---
19 result = pd.concat([preds_prob, preds_class, original_result], axis=1)
20 print(result.head())
21
```

## Getting the Confusion Matrix

- To see how good the model is in predicting if a tumor is cancerous, we use confusion matrix.
- The confusion matrix shows the number of actual and predicted labels and how many of them are classified correctly.
- Pandas's [crosstab\(\)](#) function computes a simple cross-tabulation of two factors.



```
1 #---generate table of predictions vs actual---
2 print("---Confusion Matrix---")
3 print(pd.crosstab(preds, test_labels))
4
```

---Confusion Matrix---

col_0	0	1
row_0	48	4
0	5	86

## Getting the Confusion Matrix

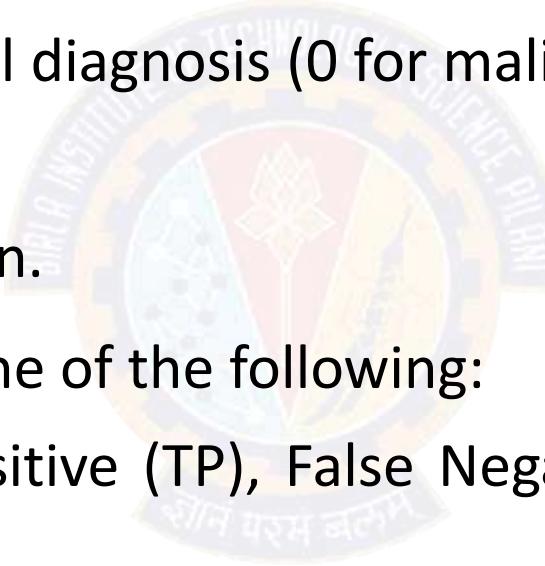
- In addition to using the crosstab() function, we can also use the confusion\_matrix() function to print out the confusion matrix:



```
1 from sklearn import metrics
2 #---view the confusion matrix---
3 print(metrics.confusion_matrix(y_true = test_labels,          # True labels
4                                     y_pred = preds))        # Predicted labels
5
[[48  5]
 [ 4 86]]
```

## Getting the Confusion Matrix

- The output is interpreted as shown below:
- The columns represent the actual diagnosis (0 for malignant and 1 for benign).
- The rows represent the prediction.
- Each individual box represents one of the following:
  - True Negative (TN), True Positive (TP), False Negative (FN), False Positive (FP)



		Actual	
		0	1
Prediction	0	TN (48)	FN (4)
	1	FP (5)	TP (86)
0 – Malignant		1 – Benign	

## Getting the Confusion Matrix

- True Positive (TP): The model correctly predicts the outcome as positive.
  - In this example, the number of TP (86) indicates the number of correct predictions that a tumor is benign.
- True Negative (TN): The model correctly predicts the outcome as negative.
  - In this example, 48 tumors were correctly predicted to be malignant.
- False Positive (FP): The model incorrectly predicted the outcome as positive, but the actual result is negative.
  - In this example, it means that the tumor is actually malignant, but the model predicted the tumor to be benign.
- False Negative (FN): The model incorrectly predicted the outcome as negative, but the actual result is positive.
  - In this example, it means that the tumor is actually benign, but the model predicted the tumor to be malignant.

		Actual	
		0	1
Prediction	0	TN (48)	FN (4)
	1	FP (5)	TP (86)

0 – Malignant  
1 – Benign

# Logistic Regression Application

innovate

achieve

lead

## Computing Accuracy, Recall, Precision, and Other Metrics

- Based on the confusion matrix, we can calculate the following metrics:

- Accuracy
- Precision
- Recall (True Positive Rate)
- F1 Score
- False Positive Rate

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

## Computing Accuracy, Recall, Precision, and Other Metrics

- To understand the concept of precision and recall, consider the following scenario
- If a malignant tumor is represented as negative and a benign tumor is represented as positive, then:
  - If the precision or recall is high, it means that more patients with benign tumors are diagnosed correctly, which indicates that the algorithm is good.
  - If the precision is low, it means that more patients with malignant tumors are diagnosed as benign.
  - If the recall is low, it means that more patients with benign tumors are diagnosed as malignant.

## Computing Accuracy, Recall, Precision, and Other Metrics

- Having a low precision is more serious than a low recall (although wrongfully diagnosed as having breast cancer when you do not have it will likely result in unnecessary treatment and mental anguish) because it causes the patient to miss treatment and potentially causes death.
- Hence, for cases like diagnosing breast cancer, it's important to consider both the precision and recall metrics when evaluating the effectiveness of an ML algorithm.

## Computing Accuracy, Recall, Precision, and Other Metrics

- To get the accuracy of the model, we can use the `score()` function of the model:

```
1 #---get the accuracy of the prediction---
2 print("---Accuracy---")
3 print(log_regress.score(X = test_set ,
4                           y = test_labels))
5
```

---Accuracy---  
0.9370629370629371

```
1 # View summary of common classification metrics
2 print("---Metrices---")
3 print(metrics.classification_report(
4     y_true = test_labels,
5     y_pred = preds))
6
```

---Metrices---

	precision	recall	f1-score	support
0	0.92	0.91	0.91	53
1	0.95	0.96	0.95	90
accuracy			0.94	143
macro avg	0.93	0.93	0.93	143
weighted avg	0.94	0.94	0.94	143

- To get the precision, recall, and F1-score of the model, use the `classification_report()` function of the `metrics` module:

## Receiver Operating Characteristic (ROC) Curve

- An easy way to examine the effectiveness of an algorithm is to plot a curve known as ROC curve.
- The ROC curve is created by plotting the TPR against the FPR at various threshold settings.
- Scikit-learn has the [roc\\_curve\(\)](#) function, which will calculate the FPR and TPR automatically based on the test labels and predicted probabilities:

## Receiver Operating Characteristic (ROC) Curve

```
1 from sklearn.metrics import roc_curve, auc
2
3 #---find the predicted probabilities using the test set
4 probs = log_regress.predict_proba(test_set)
5 preds = probs[:,1]
6
7 #---find the FPR, TPR, and threshold---
8 fpr, tpr, threshold = roc_curve(test_labels, preds)
9
10 print(fpr)
11 print(tpr)
12 print(threshold)
```

- As you can see from the output, the threshold starts at  $1.99994134e+00$  and goes down to  $1.71770685e-20$

### • The Output

```
[0.          0.          0.          0.01886792 0.01886792 0.03773585
 0.03773585 0.05660377 0.05660377 0.0754717 0.0754717 0.13207547
 0.13207547 0.1509434 0.1509434 0.22641509 0.22641509 1.        ]
[0.          0.01111111 0.66666667 0.66666667 0.87777778 0.87777778
 0.91111111 0.91111111 0.93333333 0.93333333 0.95555556 0.95555556
 0.97777778 0.97777778 0.98888889 0.98888889 1.        1.        ]
[1.99994134e+00 9.99941343e-01 9.80318580e-01 9.80023033e-01
 9.16500787e-01 9.10783370e-01 8.65783824e-01 8.32888841e-01
 8.26512365e-01 7.69015321e-01 7.00476088e-01 2.94982814e-01
 2.59110519e-01 2.35098546e-01 7.66225978e-02 3.09814091e-02
 1.09414306e-02 1.71770685e-20]
```

## Plotting the ROC and Finding the Area Under the Curve (AUC)

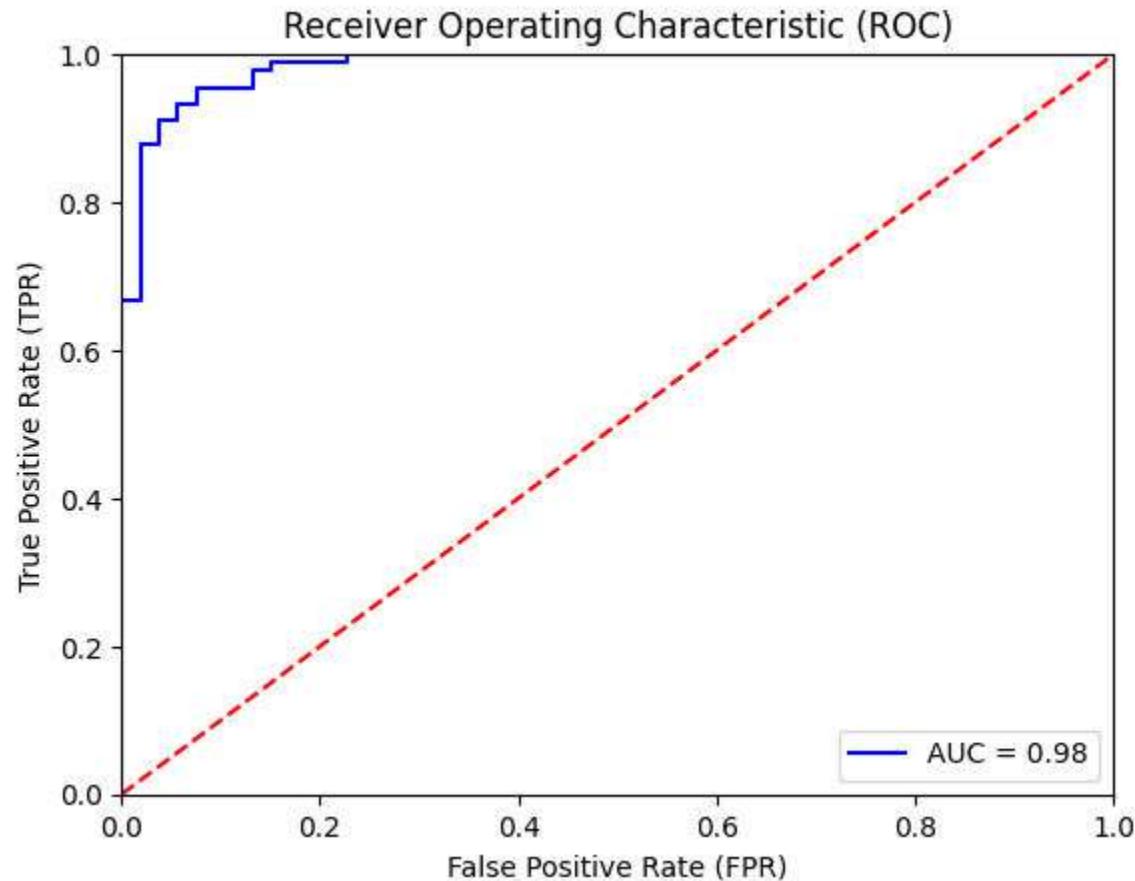
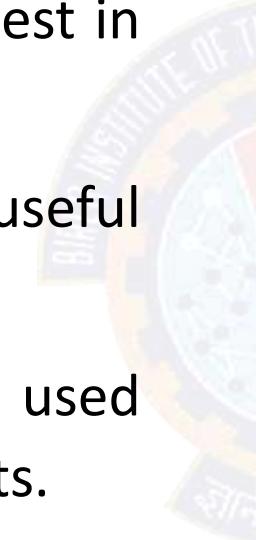
- To plot the ROC, we use matplotlib to plot a line chart using the values stored in the fpr and tpr variables.
- We can use the auc() function to find the area under the ROC:



```
1 #---find the area under the curve---
2 roc_auc = auc(fpr, tpr)
3
4 import matplotlib.pyplot as plt
5 plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
6 plt.plot([0, 1], [0, 1], 'r--')
7 plt.xlim([0, 1])
8 plt.ylim([0, 1])
9 plt.ylabel('True Positive Rate (TPR)')
10 plt.xlabel('False Positive Rate (FPR)')
11 plt.title('Receiver Operating Characteristic (ROC)')
12 plt.legend(loc = 'lower right')
13 plt.show()
14
```

## Plotting the ROC and Finding the Area Under the Curve (AUC)

- The area under an ROC curve is a measure of the usefulness of a test in general
- A greater area means a more useful test
- The areas under ROC curves are used to compare the usefulness of tests.
- Generally, aim for the algorithm with the highest AUC.



innovate

achieve

lead



# Support Vector Machine

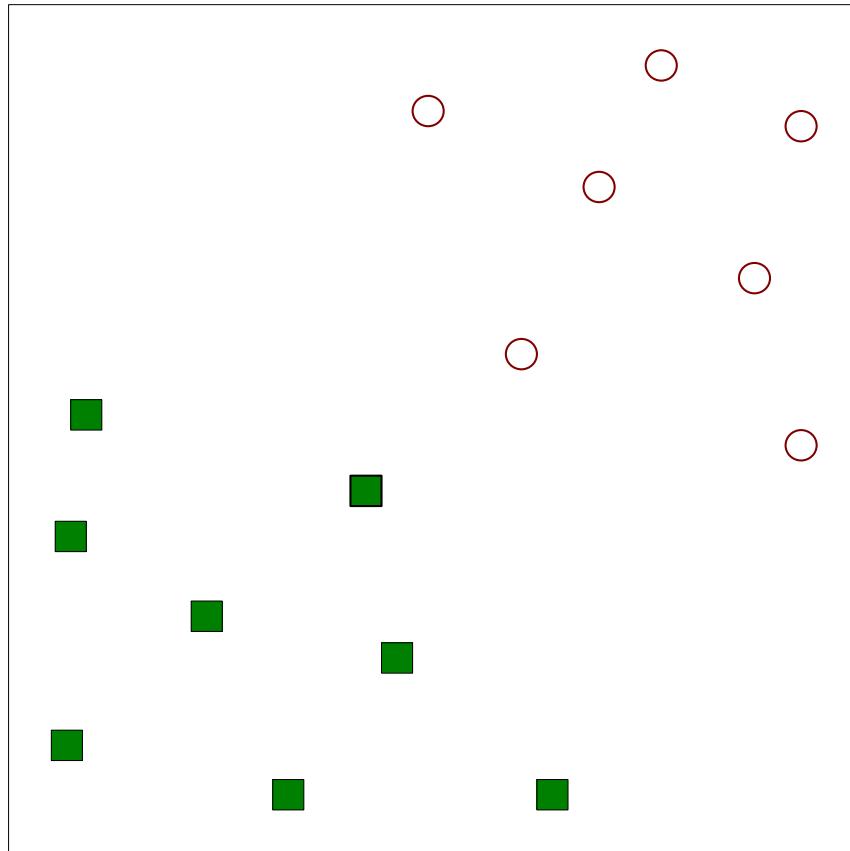
# Support Vector Machines

innovate

achieve

lead

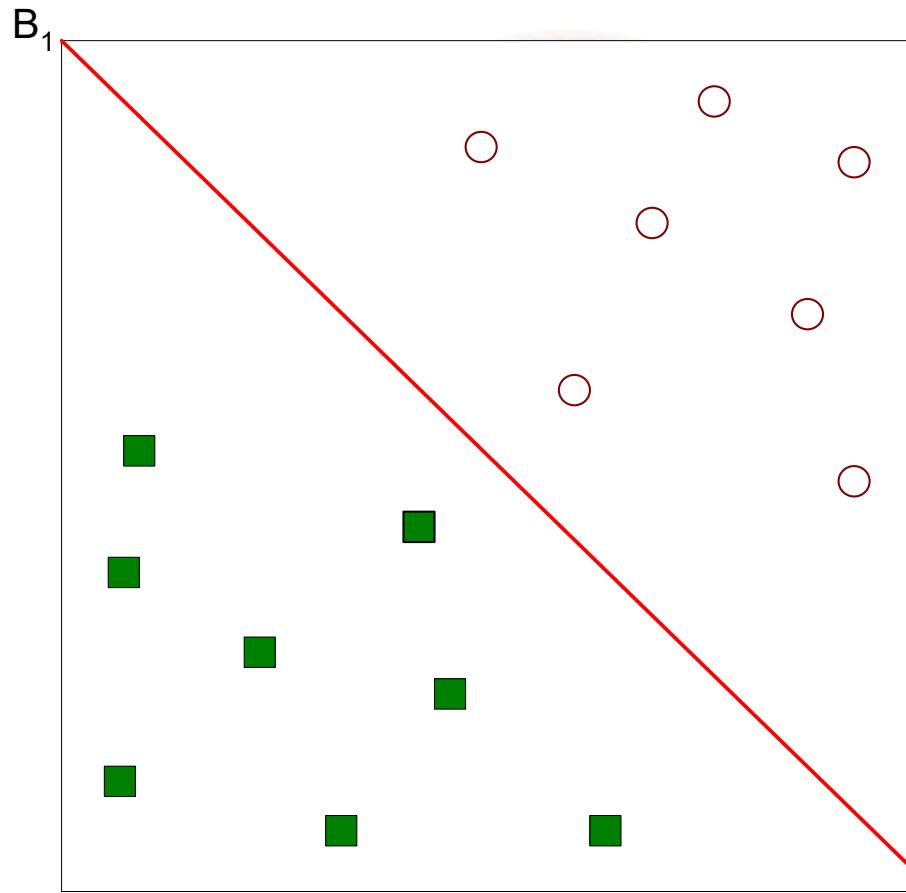
Find a linear hyperplane (decision boundary) that will separate the data



# Support Vector Machines



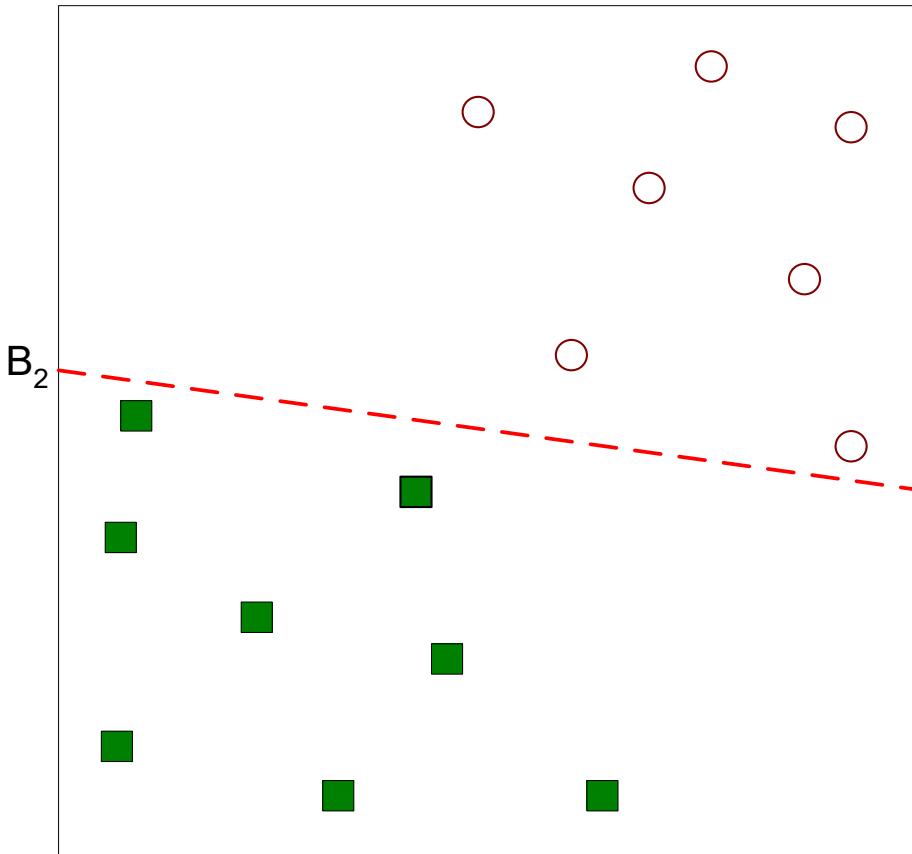
## One Possible Solution



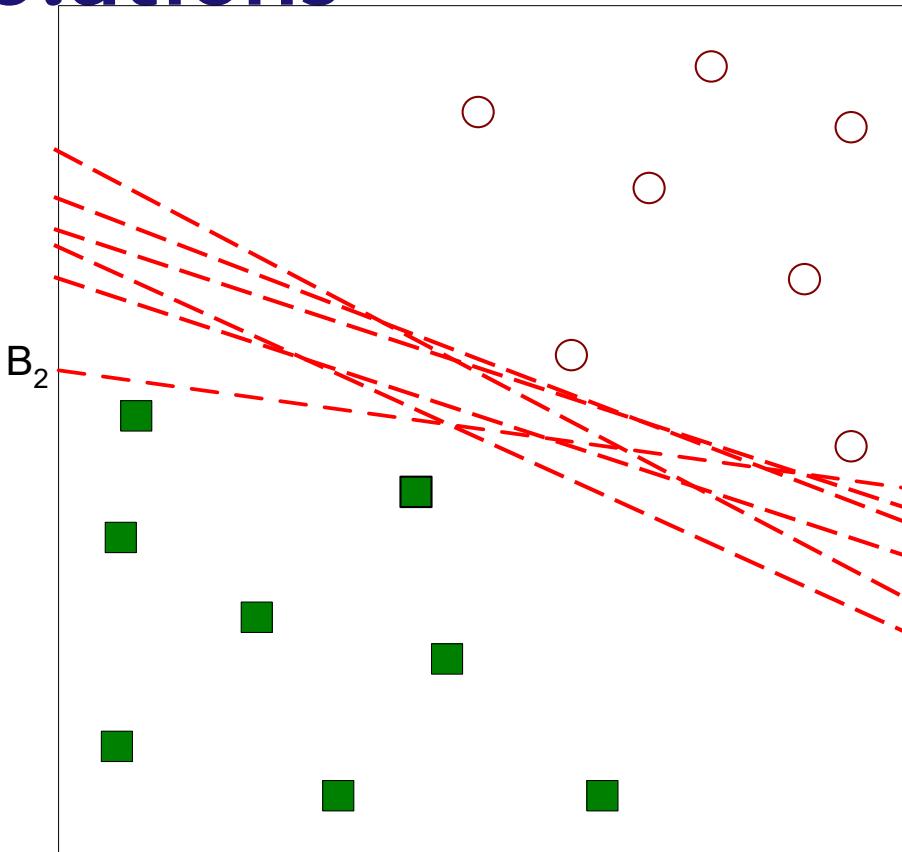
# Support Vector Machines



## Another possible solution



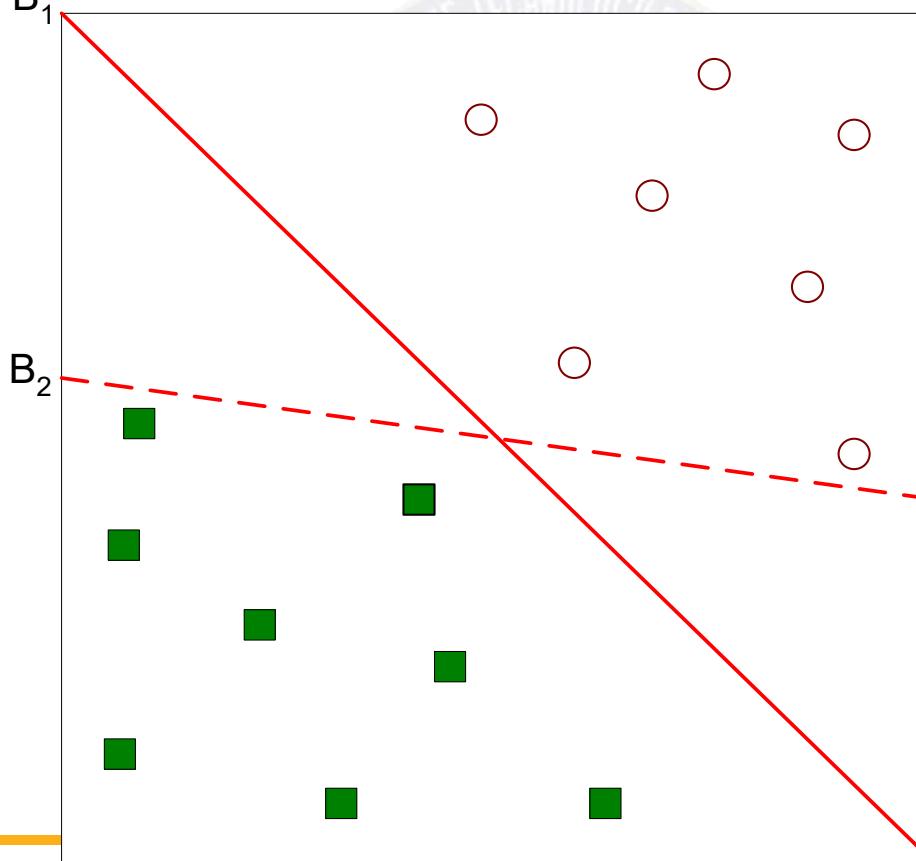
## Other possible solutions



# Support Vector Machines



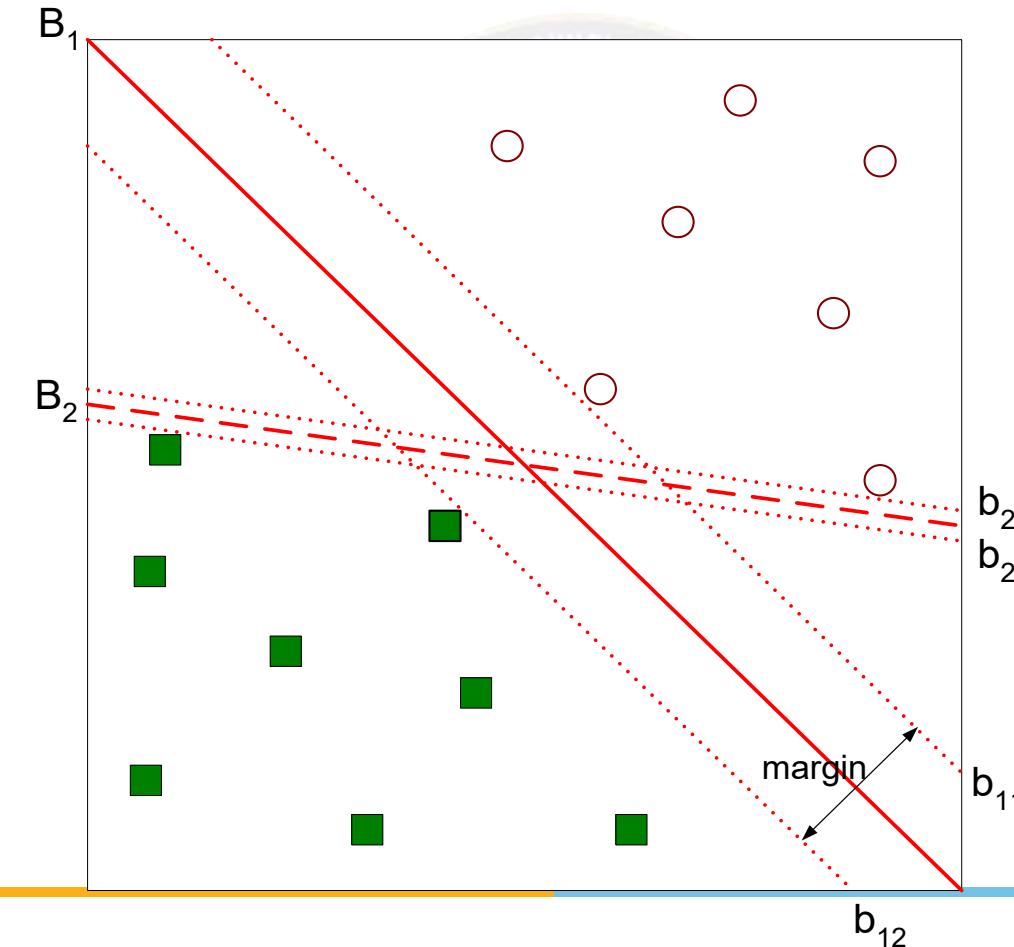
- Which one is better? B1 or B2?
- How do you define better?



# Support Vector Machines

## Find hyperplane **maximizes** the margin

- => B1 is better than B2

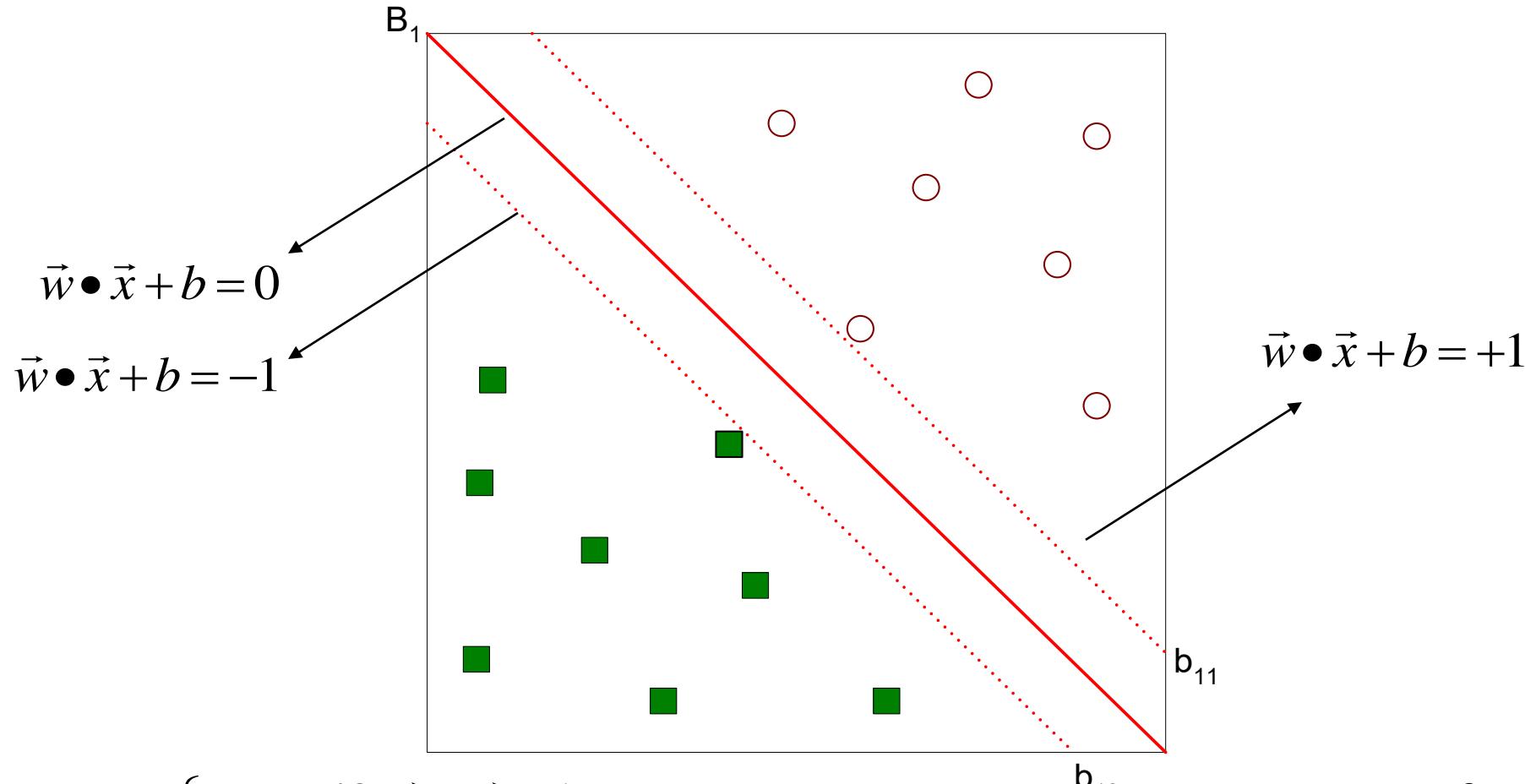


# Support Vector Machines

innovate

achieve

lead



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

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

## Linear Model

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

- Learning the model is equivalent to determining the values of  $\vec{w}$  and  $b$ 
  - How to find from training data?

# Learning Linear SVM

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

- Objective is to maximize:

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

- Which is equivalent to minimizing:

$$L(\vec{w}) = \frac{\|\vec{w}\|^2}{2}$$

- Subject to the following constraints:

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

or

- This is a constrained optimization problem
  - Solve it using Lagrange multiplier method
  - Lagrange multipliers  $\lambda_i$  are 0 or +ve

$$L(\mathbf{w}, b, \lambda_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum \lambda_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

# Learning Linear SVM



- Solution:  $\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i$     $\lambda_i$  is non-zero except for support vectors  
 $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$  (for any support vector)
- Classification function:

$$\begin{aligned}f(x) &= \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \\&= \text{sign}\left(\sum_i \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)\end{aligned}$$

If  $f(x) < 0$ , classify as negative, otherwise classify as positive.

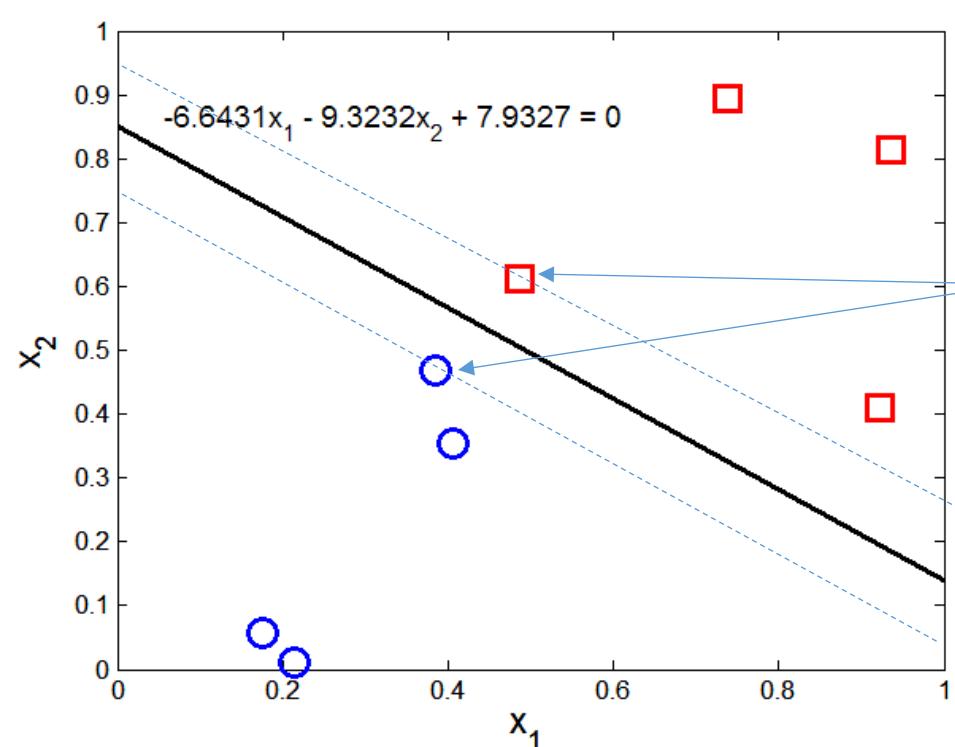
- Notice that it relies on an *inner product* between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$
- (Solving the optimization problem also involves computing the inner products  $\mathbf{x}_i \cdot \mathbf{x}_j$  between all pairs of training points)

# Example of Linear SVM

innovate

achieve

lead



<b>x1</b>	<b>x2</b>	<b>y</b>	<b><math>\lambda</math></b>
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

Support vectors

- Decision boundary depends only on support vectors
  - If you have data set with same support vectors, decision boundary will not change

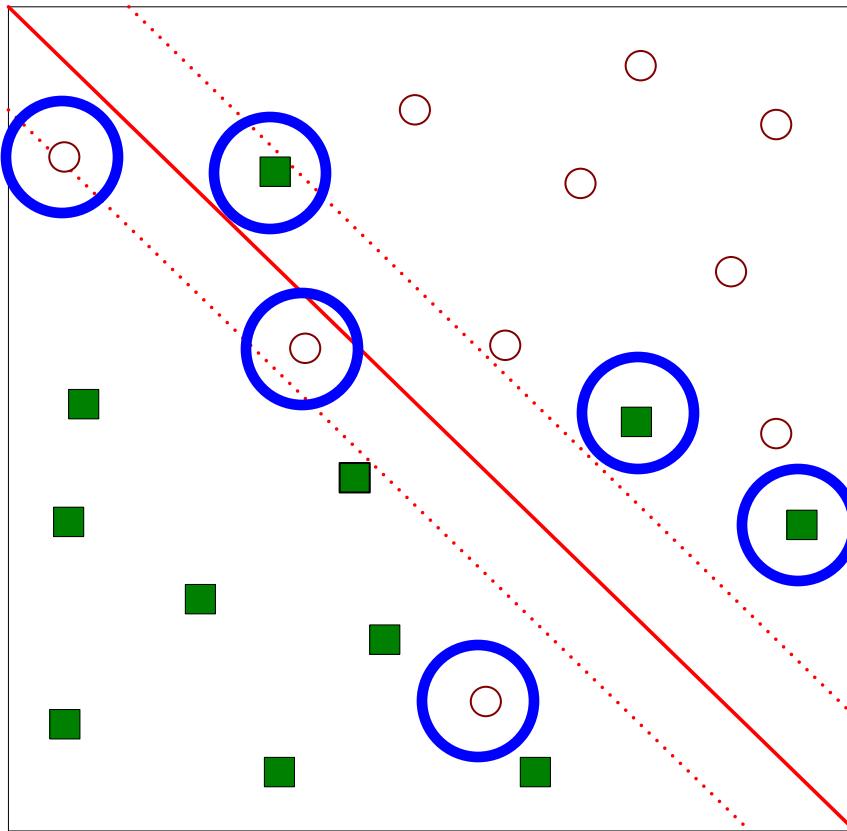
$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

- How to classify using SVM once  $w$  and  $b$  are found? Given a test record,  $x_i$

# Support Vector Machines



## What if the problem is not linearly separable?



## What if the problem is not linearly separable?

- Introduce slack variables

- Need to minimize

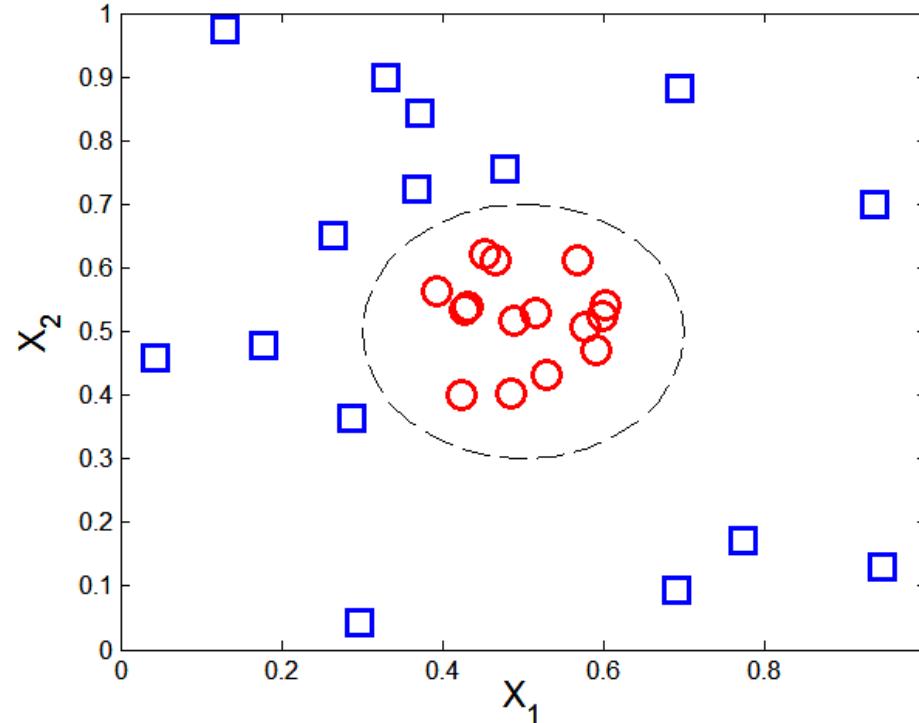
$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left( \sum_{i=1}^N \xi_i \right)$$

- subject to

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

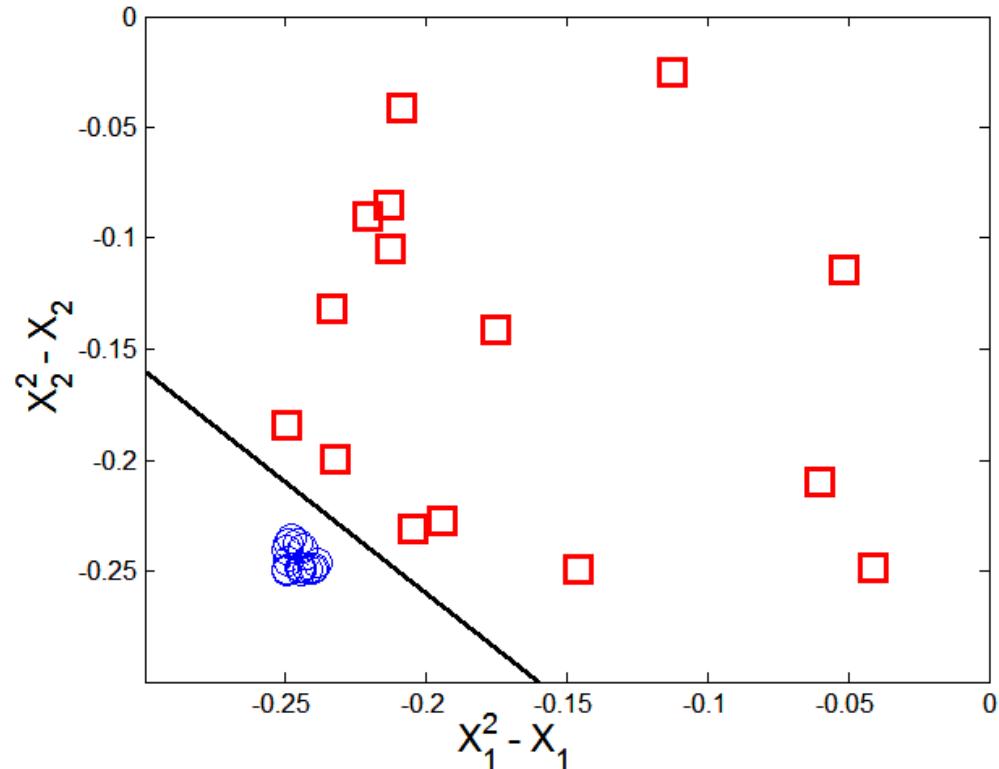
- If k is 1 or 2, this leads to similar objective function as linear SVM but with different constraints

## What if decision boundary is not linear?



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

## Transform data into higher dimensional space



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

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

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

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

## Optimization Problem

$$\begin{aligned} & \min_w \frac{\|w\|^2}{2} \\ & \text{subject to} \quad y_i(w \cdot \Phi(x_i) + b) \geq 1, \quad \forall \{(x_i, y_i)\} \end{aligned}$$



- **why**  $L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$  **etc**  $w = \sum_i \lambda_i y_i \Phi(x_i)$   
**inst**  $\lambda_i \{y_i (\sum_j \lambda_j y_j \Phi(x_j) \cdot \Phi(x_i) + b) - 1\} = 0,$

$$f(z) = \text{sign}(w \cdot \Phi(z) + b) = \text{sign}\left(\sum_{i=1}^n \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b\right).$$

## Issues

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



# Learning Nonlinear SVM



- Kernel Trick:

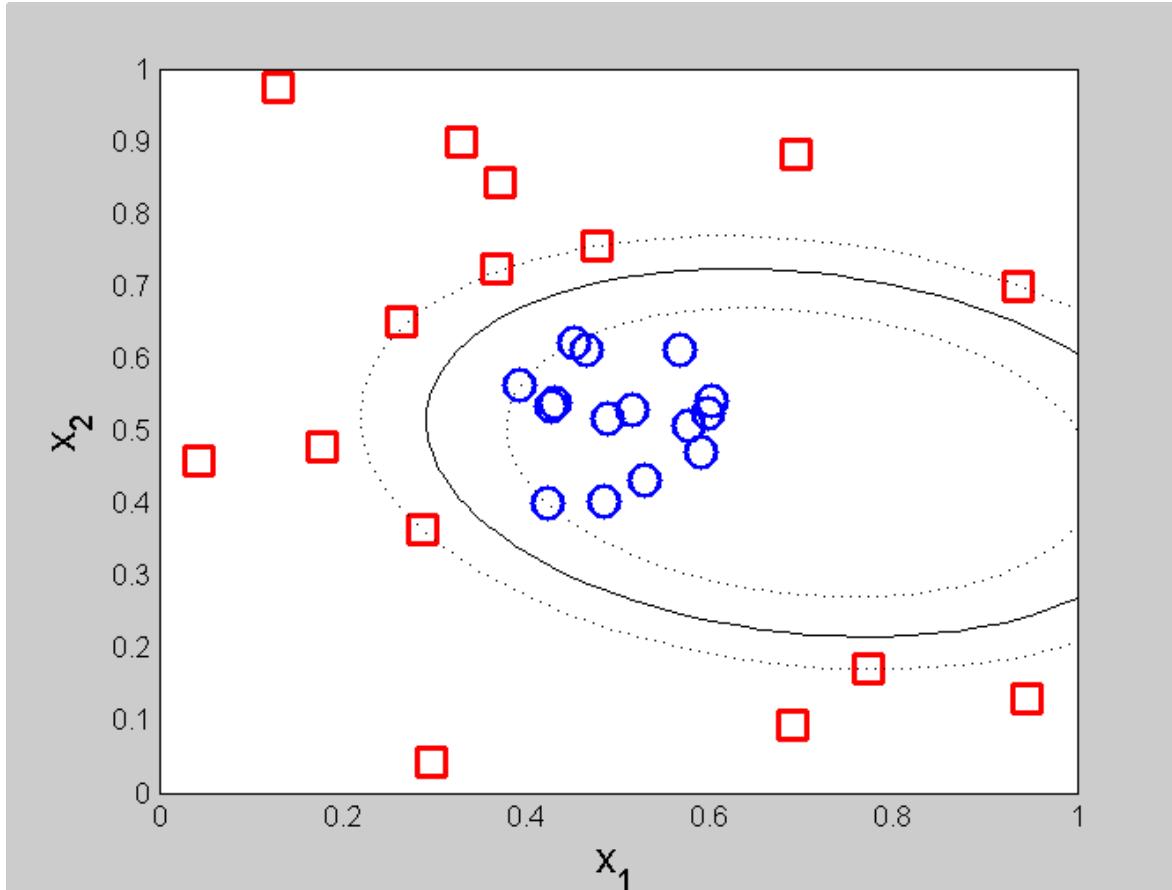
- $\Phi(x_i) \bullet \Phi(x_j) = K(x_i, x_j)$
- $K(x_i, x_j)$  is a kernel function (expressed in terms of the coordinates in the original space)
  - Examples:

$$K(x, y) = (x \cdot y + 1)^p$$

$$K(x, y) = e^{-\|x-y\|^2/(2\sigma^2)}$$

$$K(x, y) = \tanh(kx \cdot y - \delta)$$

# Example of Nonlinear SVM



**SVM with polynomial degree 2 kernel**

- Advantages of using kernel:
  - Don't have to know the mapping function  $\Phi$
  - Computing dot product  $\Phi(x_i) \bullet \Phi(x_j)$  in the original space avoids curse of dimensionality
- Not all functions can be kernels
  - Must make sure there is a corresponding  $\Phi$  in some high-dimensional space
  - Mercer's theorem (see textbook)



**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

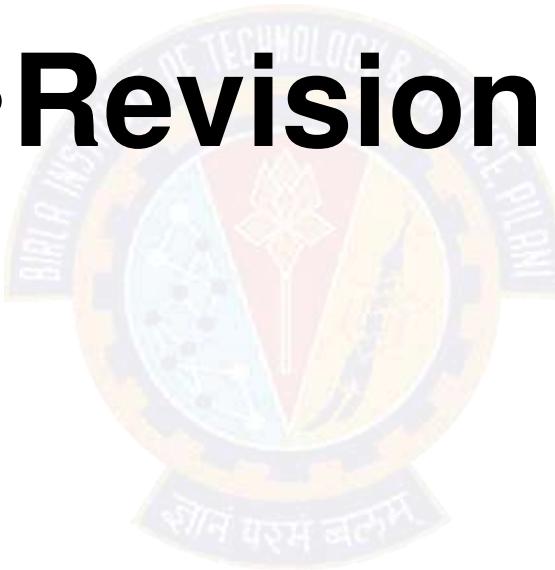
[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



# **Session 8(09<sup>th</sup> September,2023)**

# Session 8 – 09<sup>th</sup> September, 2023

- Revision





**Session – 13(17<sup>th</sup> June 2023)**  
**SUPPORT VECTOR MACHINES**

# Support Vector Machine

## What is Support Vector Machine?

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

# What is Support Vector Machine?

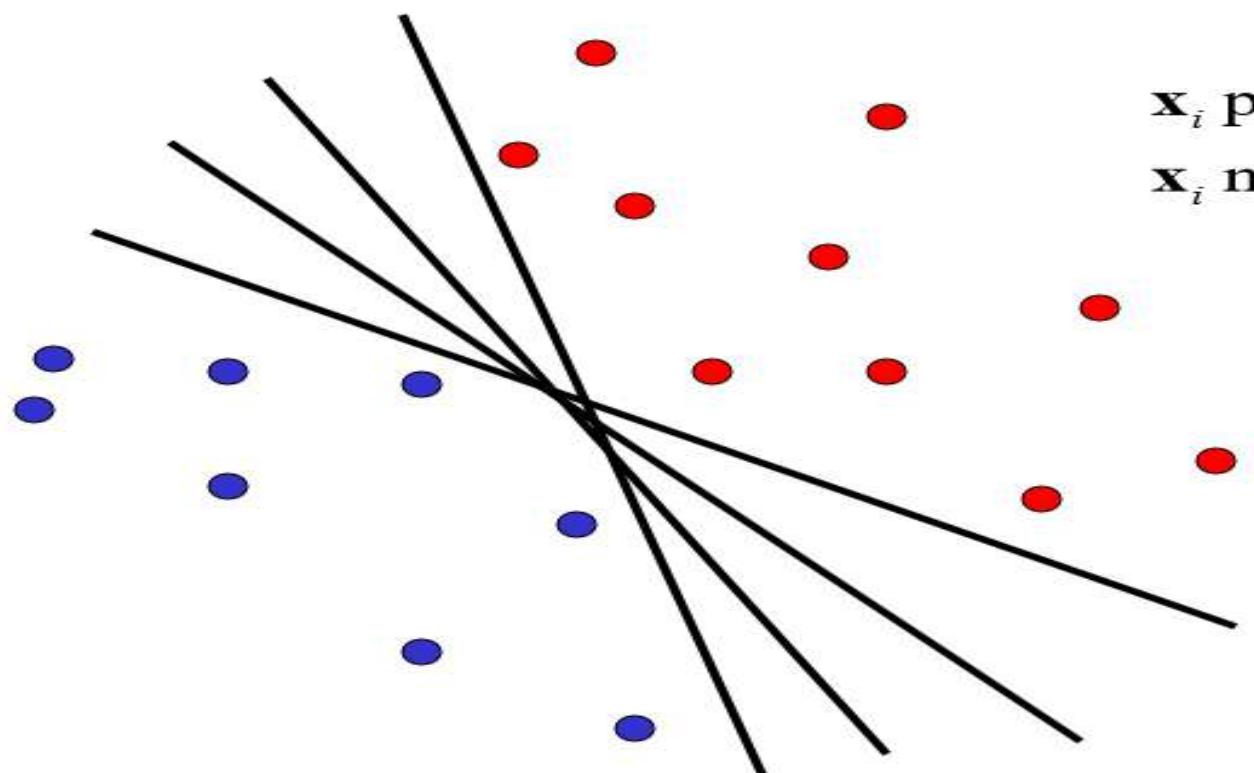
- The objective of the support vector machine algorithm is to find a hyperplane in an  $N$ -dimensional space ( $N$ —the number of features) that distinctly classifies the data points.
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen.
- Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

- Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

- **Support Vectors** – Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** – As we can see in the diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** – It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

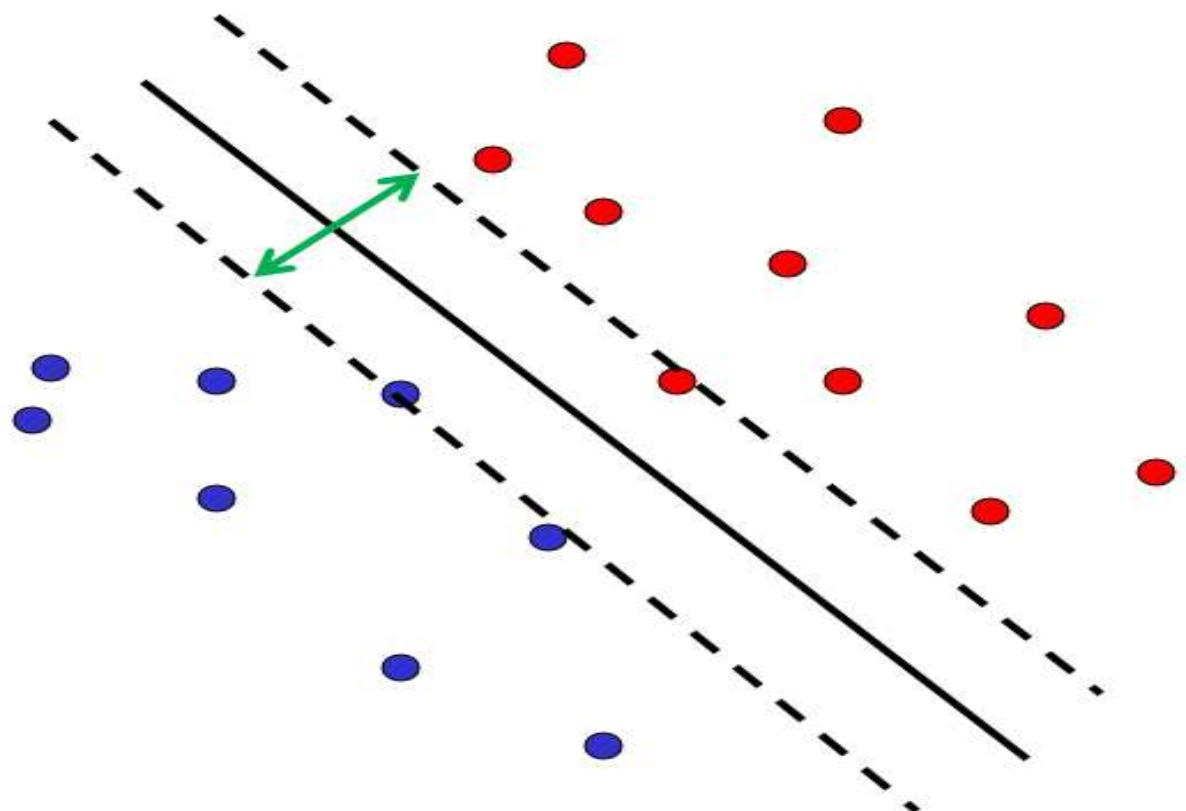
# Linear Classifier

- Find linear function to separate positive and negative examples

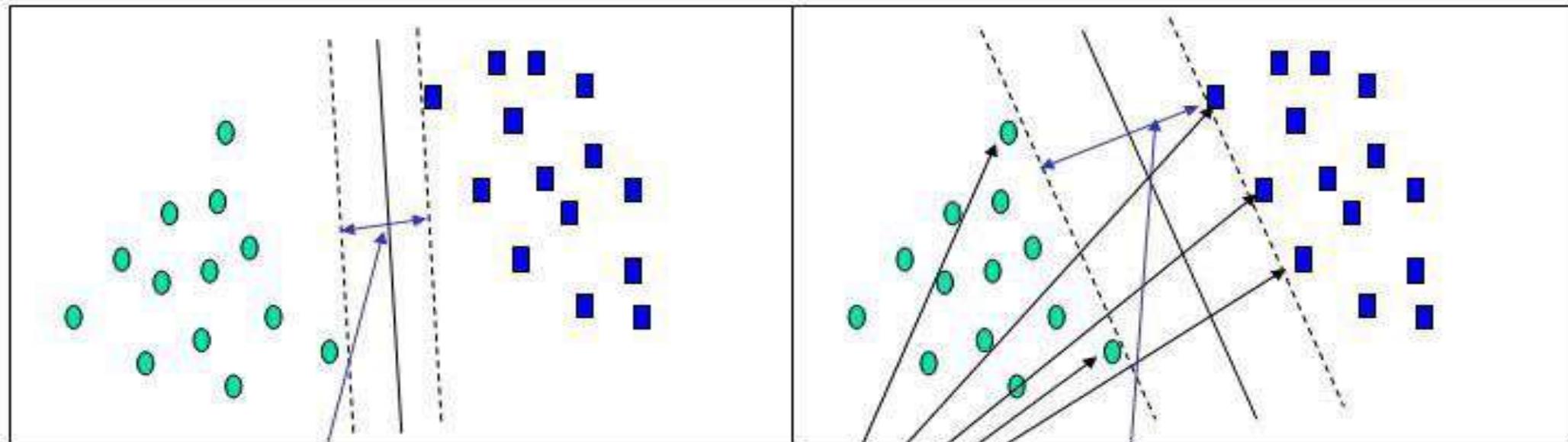


$\mathbf{x}_i$  positive :  $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$   
 $\mathbf{x}_i$  negative :  $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

Which line  
is best?



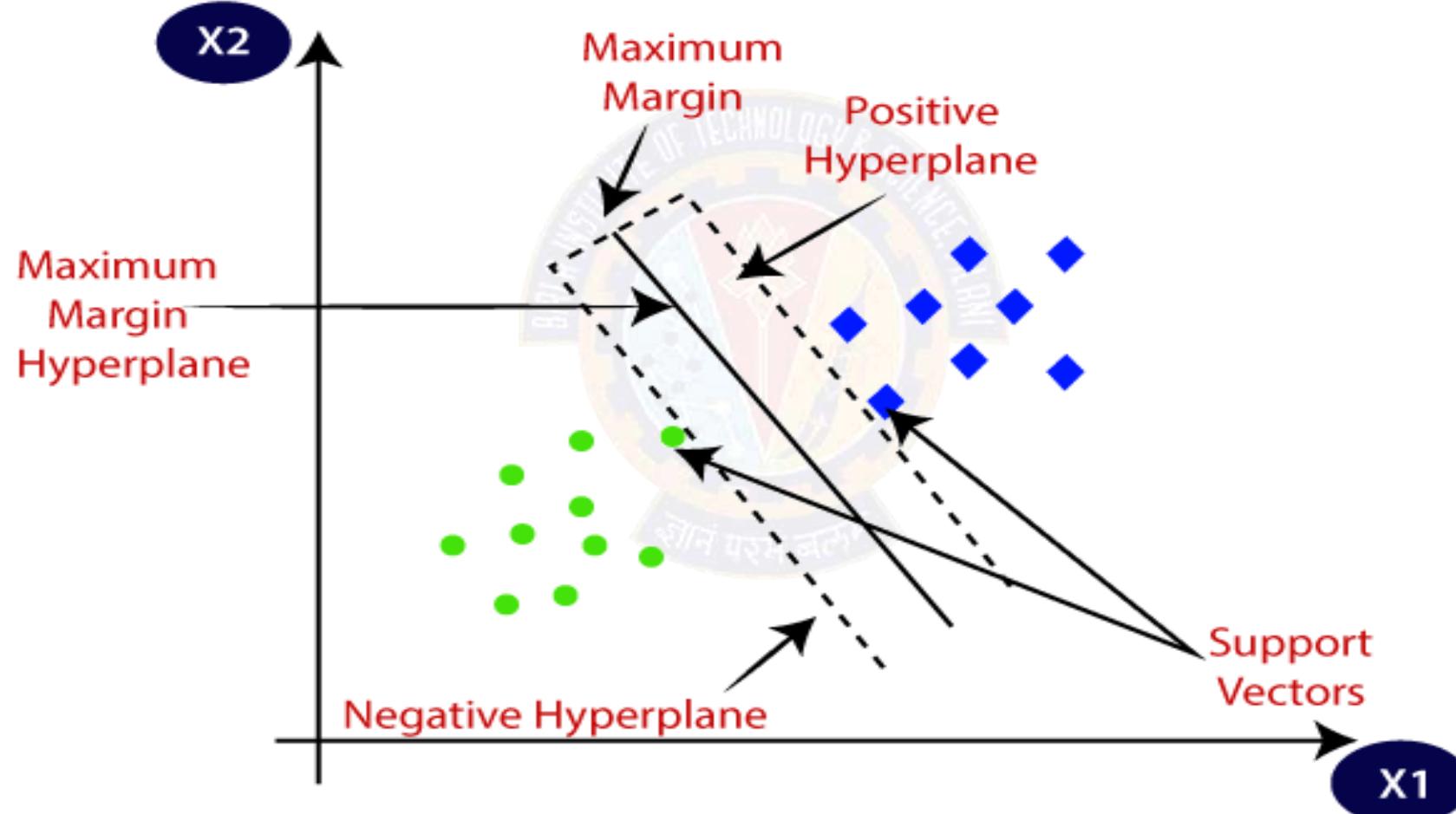
- Discriminative classifier based on *optimal separating line* (for 2d case)
- Maximize the *margin* between the positive and negative training examples



Small Margin

Large Margin

Support Vectors



# Types of SVM

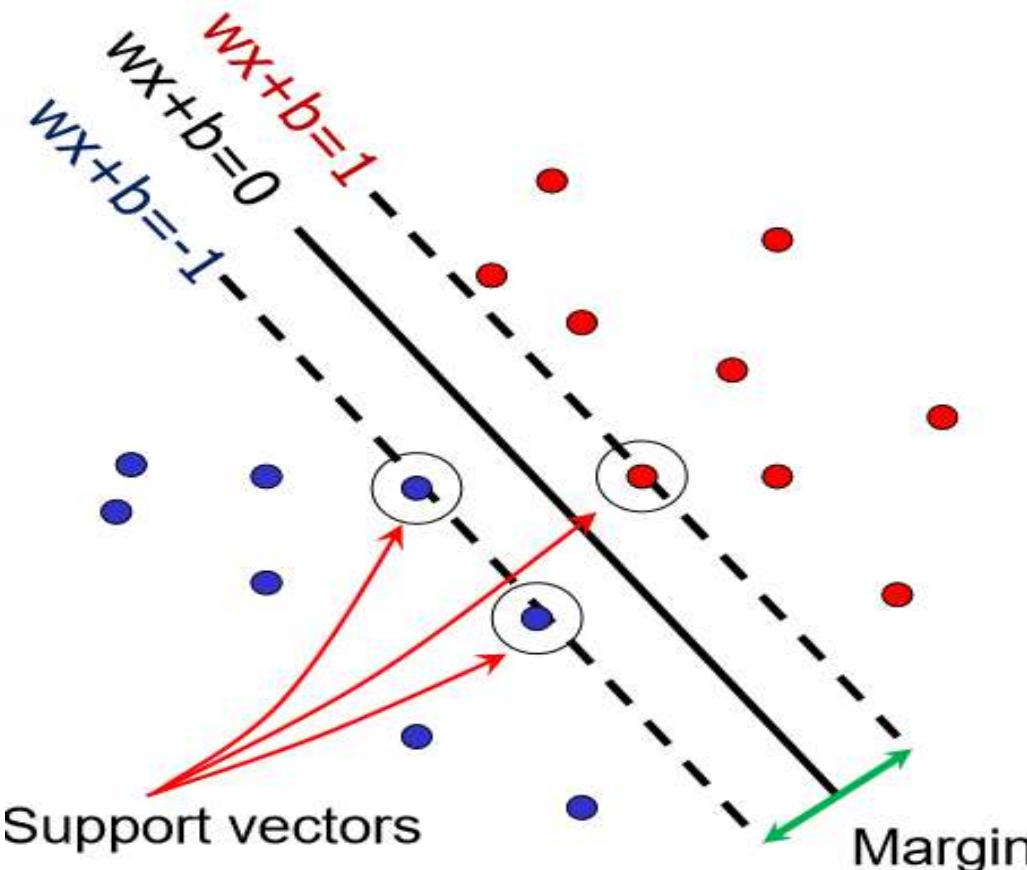
- **SVM can be of two types:**

➤ **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

➤ **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

# Support Vector Machines

- Want line that maximizes the margin.

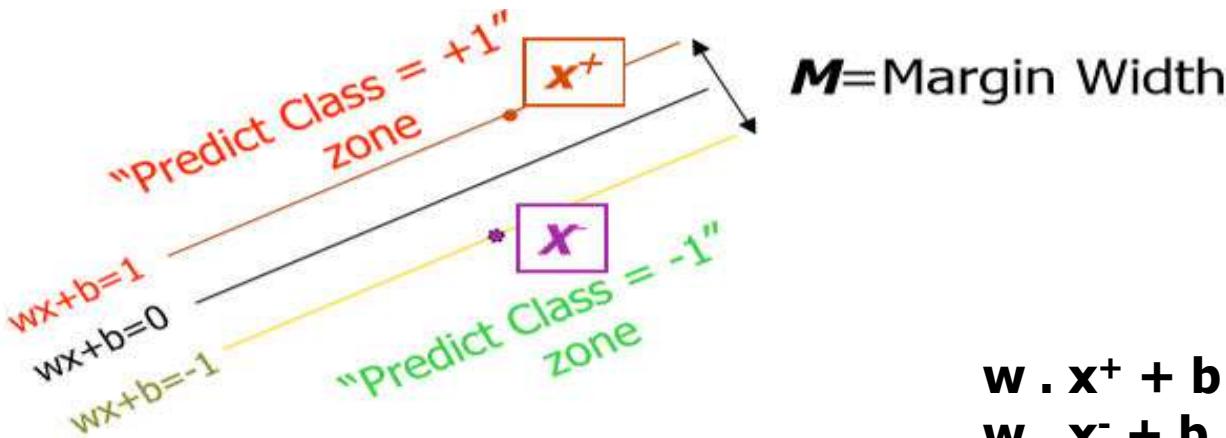


$\mathbf{x}_i$  positive ( $y_i = 1$ ):  $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$  negative ( $y_i = -1$ ):  $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors,  $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

# Linear SVM Mathematically



$$w \cdot x^+ + b = +1$$

$$w \cdot x^- + b = -1$$

**Margin width**

$$= x^+ - x^- \cdot \frac{w}{||w||}$$

$$= \frac{w \cdot x^+ - w \cdot x^-}{||w||}$$

$$= (1-b) - (-1-b) / ||w||$$

$$= \frac{2}{||w||}$$

# Solving the Optimization Problem

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:*

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

# Solving the Optimization Problem

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear inequality* constraints.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primal problem

# Solving the Optimization Problem

- 
- Solution:  $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned  
weight

Support  
vector

# Solving the Optimization Problem

- **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)

- **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

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$

- **Parameter  $C$  can be viewed as a way to control overfitting.**

# The “Kernel Trick”

- The linear classifier relies on dot product between vectors
  - $x_i^T \cdot x_j$
- If every data point is mapped into high-dimensional space via some transformation  $\Phi: x \rightarrow \phi(x)$ , the dot product becomes:
  - $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:
  - 2-dimensional vectors  $x = [x_1 \ x_2]$ ; let  $K(x_i, x_j) = (1 + x_i^T x_j)^2$ ,
  - Need to show that  $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ :



**Thank You!**



**BITS Pilani**  
Pilani | Dubai | Goa | Hyderabad

# Applied Machine Learning

## SEZG568/SSZG568

---

Dr Y V K RAVI KUMAR

[yvk.ravikumar@pilani.bits-pilani.ac.in](mailto:yvk.ravikumar@pilani.bits-pilani.ac.in)



**Session –8 (16<sup>th</sup> September,2023)**  
**SUPPORT VECTOR MACHINES & Revision**

# Support Vector Machine

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

# What is Support Vector Machine?

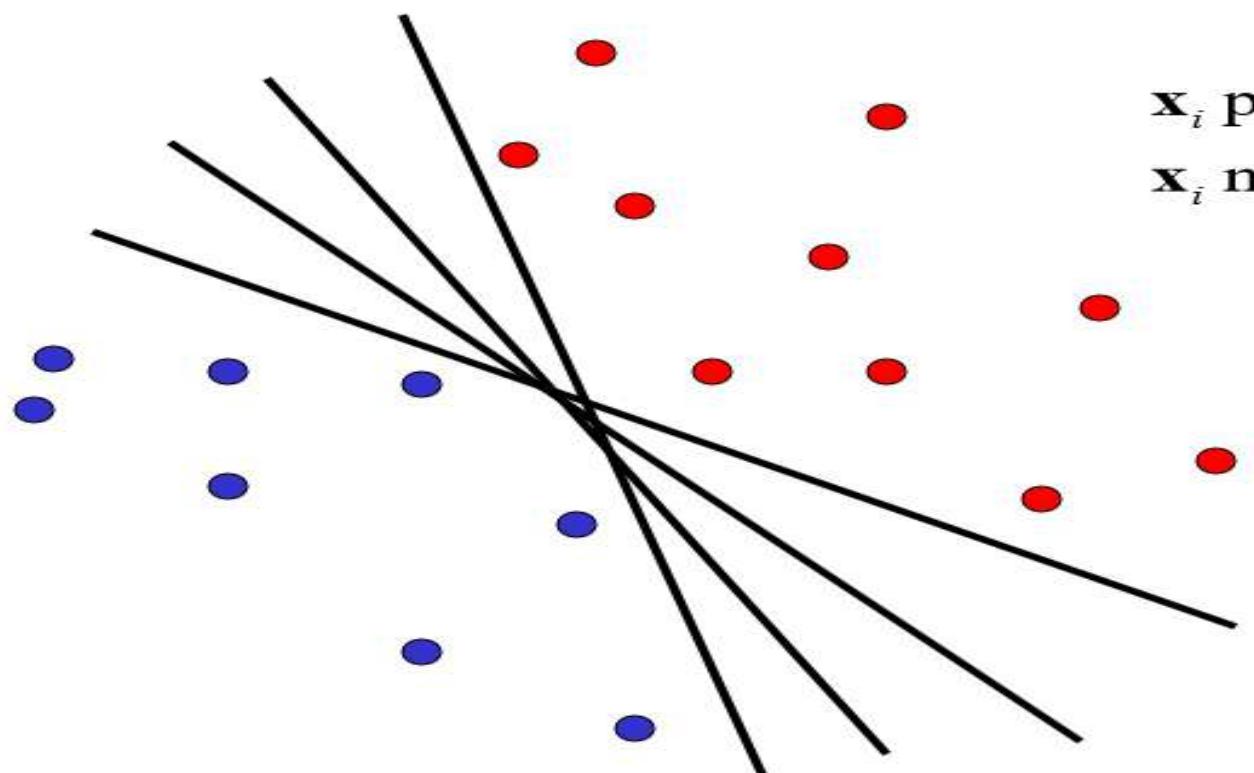
- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N—the number of features) that distinctly classifies the data points.
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen.
- Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

- Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

- **Support Vectors** – Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** – As we can see in the diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** – It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

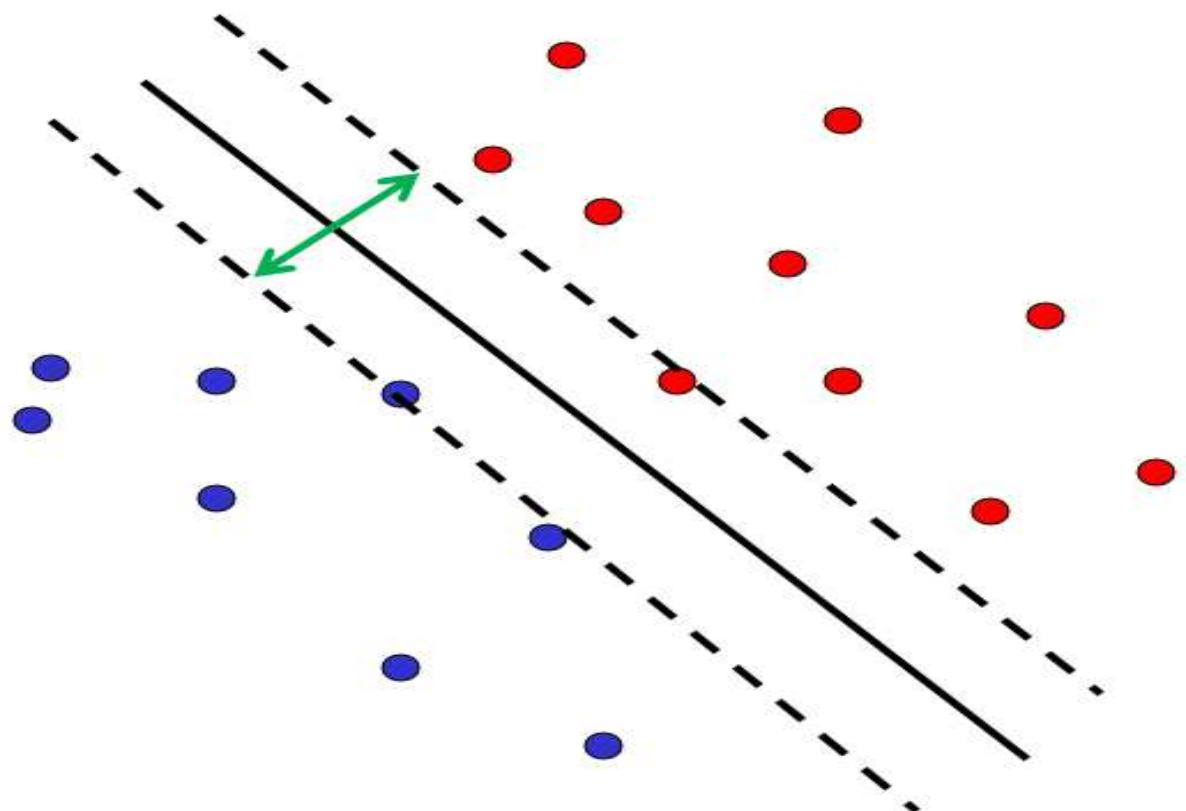
# Linear Classifier

- Find linear function to separate positive and negative examples

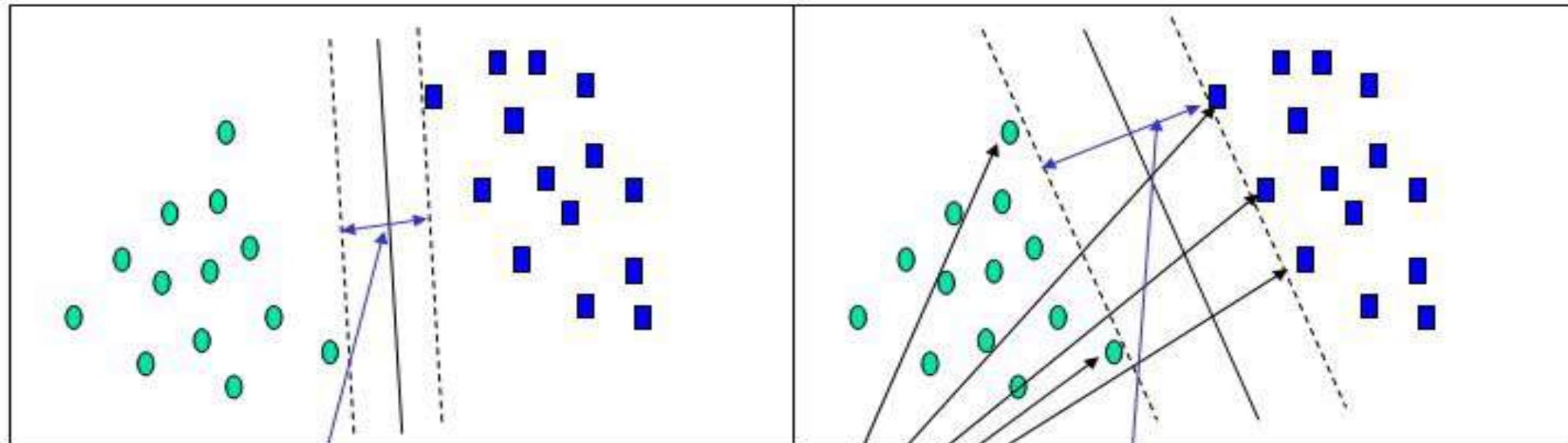


$$\mathbf{x}_i \text{ positive : } \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

Which line  
is best?



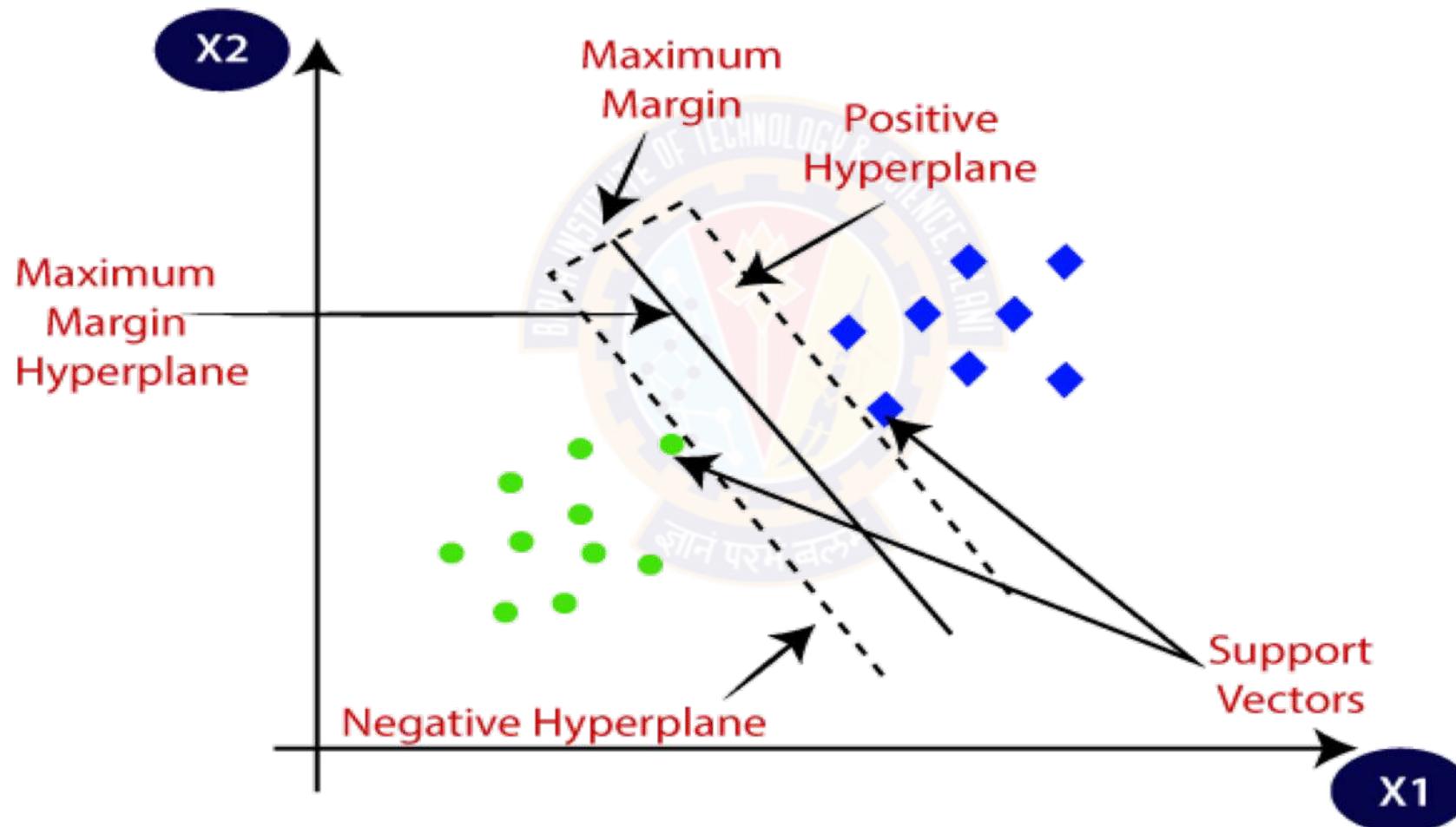
- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples



Small Margin

Large Margin

Support Vectors



# Types of SVM

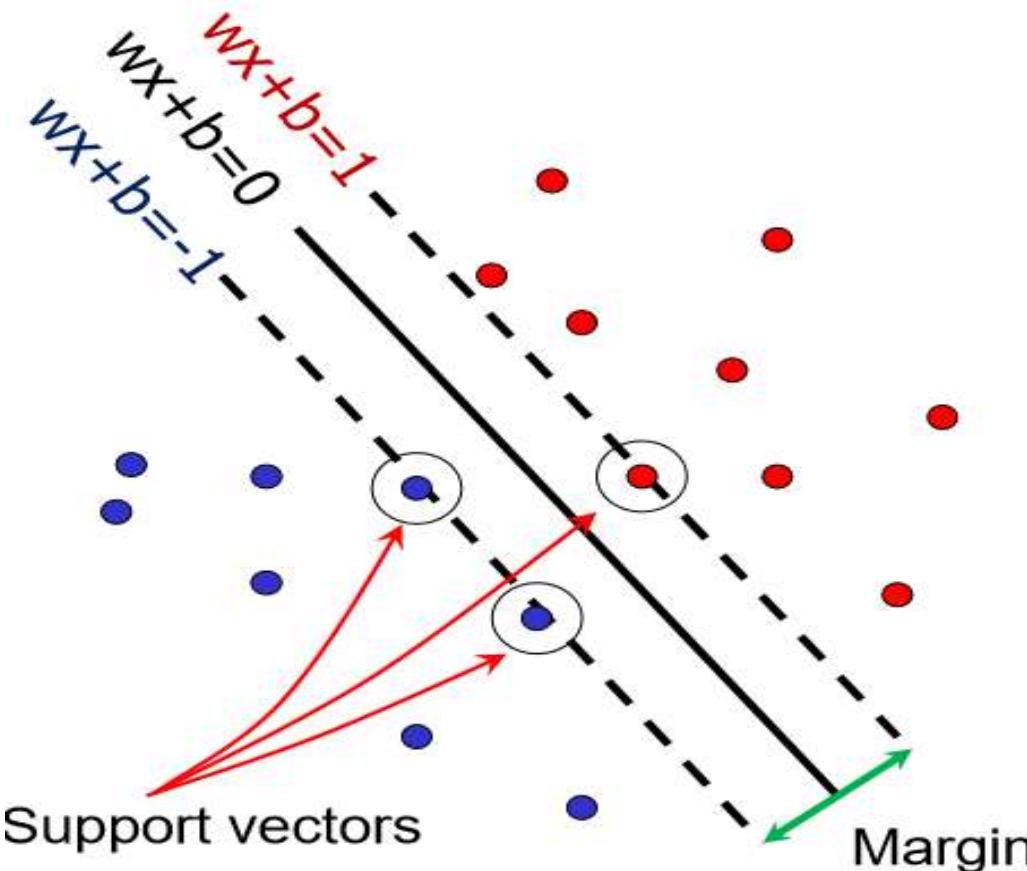
- **SVM can be of two types:**

➤ **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

➤ **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

# Support Vector Machines

- Want line that maximizes the margin.

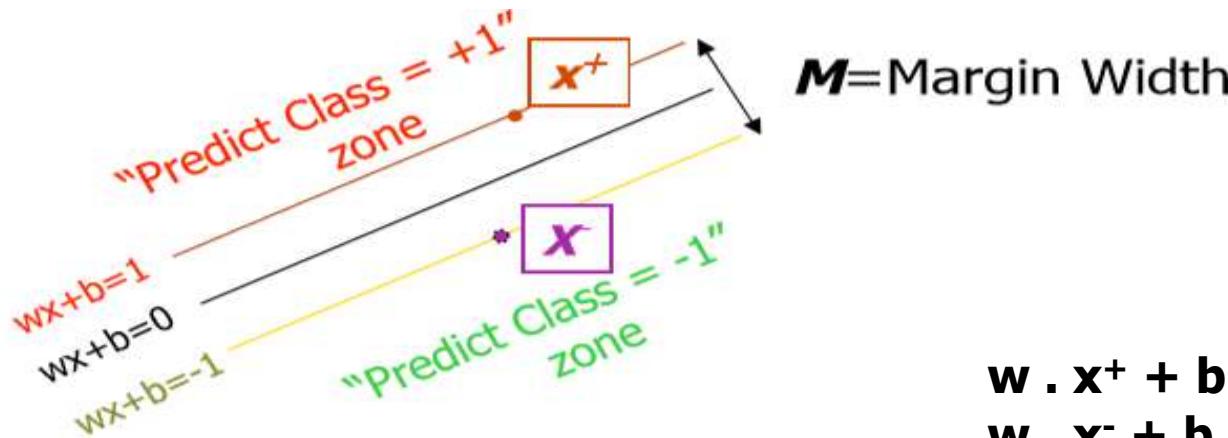


$\mathbf{x}_i$  positive ( $y_i = 1$ ):  $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$  negative ( $y_i = -1$ ):  $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors,  $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

# Linear SVM Mathematically



$$\mathbf{w} \cdot \mathbf{x}^+ + \mathbf{b} = +1$$

$$\mathbf{w} \cdot \mathbf{x}^- + \mathbf{b} = -1$$

**Margin width**

$$= \mathbf{x}^+ - \mathbf{x}^- \cdot \frac{\mathbf{w}}{||\mathbf{w}||}$$

$$= \frac{\mathbf{w} \cdot \mathbf{x}^+ - \mathbf{w} \cdot \mathbf{x}^-}{||\mathbf{w}||}$$

$$= (1-\mathbf{b}) - (-1-\mathbf{b}) / ||\mathbf{w}||$$

$$= \frac{2}{||\mathbf{w}||}$$

# Solving the Optimization Problem

1. Maximize margin  $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

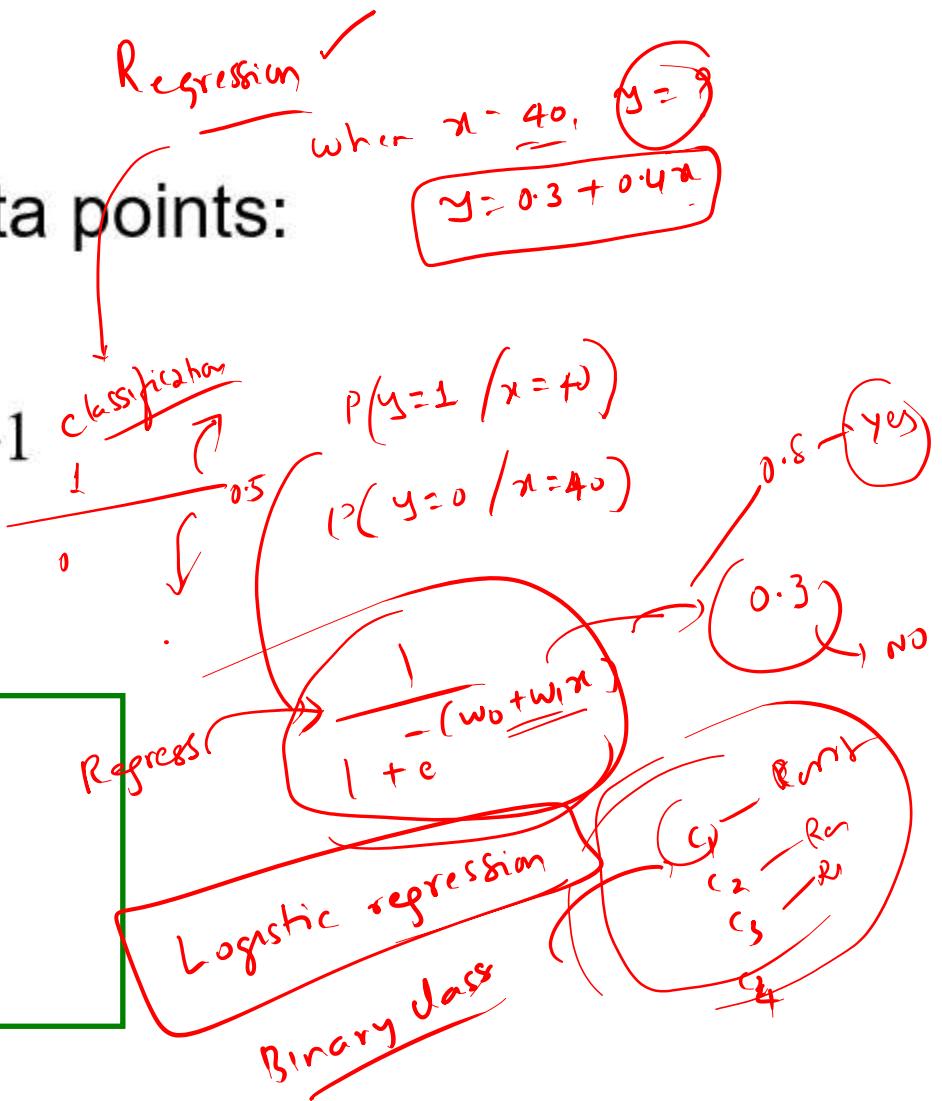
$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

*Quadratic optimization problem:*

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$



# Solving the Optimization Problem

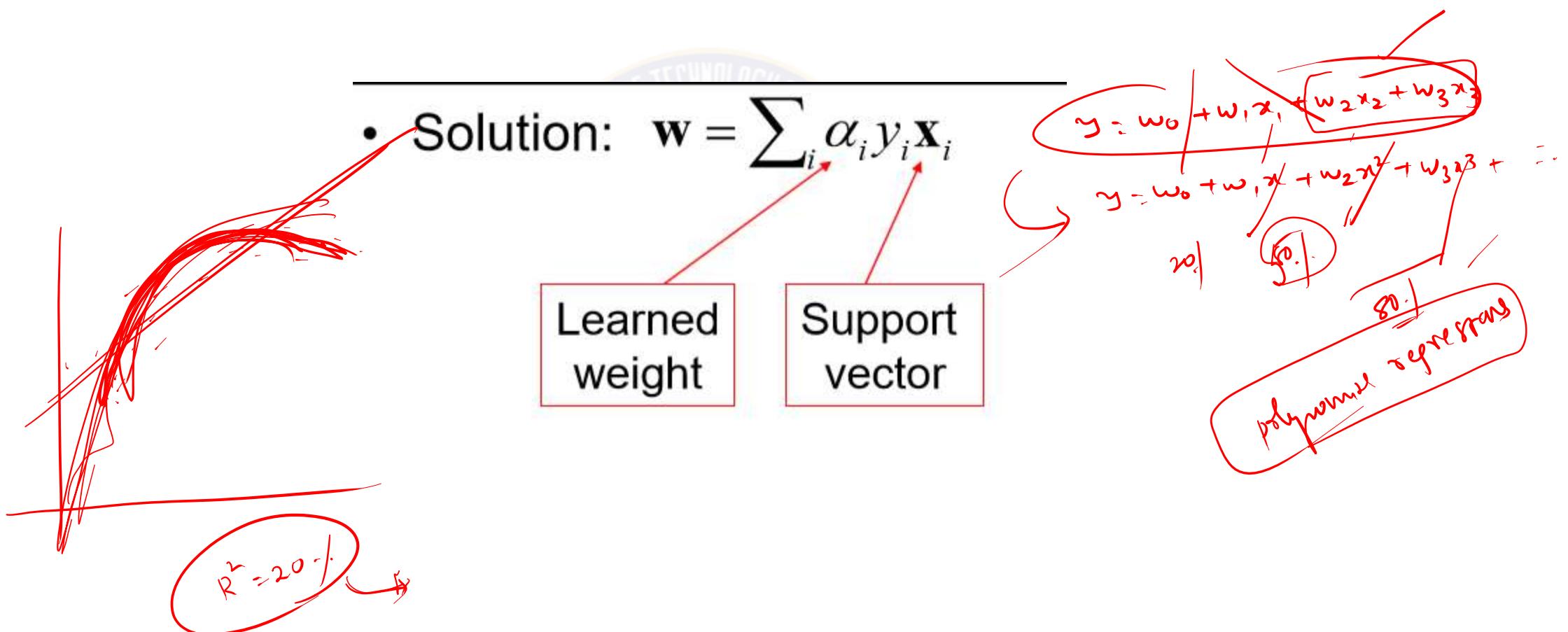
Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear inequality* constraints.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primal problem

# Solving the Optimization Problem



# Solving the Optimization Problem

- **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)

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

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

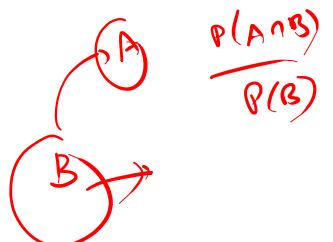
Bayes' theorem

- **Hard Margin:**

Find  $w$  and  $b$  such that

$$\Phi(w) = \frac{1}{2} w^T w \text{ is minimized and for all } \{(x_i, y_i)\}$$

$$y_i (w^T x_i + b) \geq 1$$



- **Soft Margin incorporating slack variables:**

Find  $w$  and  $b$  such that

$$\Phi(w) = \frac{1}{2} w^T w + C \sum \xi_i \text{ is minimized and for all } \{(x_i, y_i)\}$$

$$y_i (w^T x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$



- **Parameter  $C$  can be viewed as a way to control overfitting.**

# The “Kernel Trick”

- The linear classifier relies on dot product between vectors
    - $x_i^T \cdot x_j$
  - If every data point is mapped into high-dimensional space via some transformation  $\Phi: x \rightarrow \phi(x)$ , the dot product becomes:
    - $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
  - A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
  - Example:

Fig. 2.

T<sub>1</sub>

## Example:

2-dimensional vectors  $x = [x_1 \ x_2]$ ; let  $K(x_i, x_j) = (1 + x_i^T x_j)$

Need to show that  $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$ :

Adv. exp.

Schem

$$y = f(x) \quad \text{Expedit.}$$

$$(x_j): \quad (y - \bar{y}) = k \frac{\sigma_y}{\sigma_x} (x - \bar{x}) \quad \text{with } k = \frac{10}{20}$$

Laplace transform

$$= P(\text{How} \text{ and } \text{Stat}) \cdot P(\text{How} | \text{Stat}) \cdot P(\text{Year} | \text{Stat})$$

$$= (1 + x_1^T x_1)^2$$

1

Introduction to Machine Learning: What and Why, Applications of Machine Learning, Types of Machine Learning, Challenges in Machine Learning

core basics  
Interpretation

6

2

End-to-end Machine Learning: Framing the ML Problem. Data Types, Pre-processing, Visualization and Analysis

1. Learn  
2. Regress  
3. Classify

6.2

3

End-to-end Machine Learning: Model Selection and Training for Prediction and Classification, Evaluation, Machine Learning Pipeline.

Linear vs  
non-linear

9.3

4

Linear Prediction Models: Linear Regression, Gradient Descent and Variants, Regularization, Bias Vs. Variance

Logistic  
Regression  
LR  
SVM  
Decision  
Boundary  
Bias  
Variance  
Regularization  
Clustering

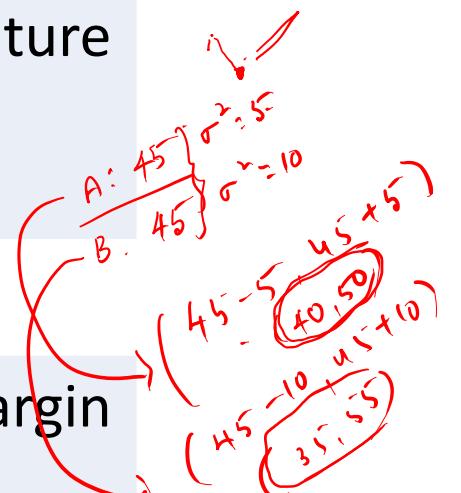
0.9  
0.4

5 Classification Models I: Naïve Bayes classification, Applications in text and image classification

$$\text{Variance: } \sigma^2 \approx \frac{(x - \bar{x})^2}{n}$$

6 Classification Models I: Logistic Regression, Log Loss error function, Optimization using gradient descent, Feature transformation for nonlinear classification

$$\text{Bias: } y = w_0 + w_1 x$$



7 Classification Models I: Support Vector Machine. Margin maximization. Non-linear SVM. Kernel Function.



Gradient Descent ✓  
Numerical

$w_0 = w_1 =$  } Initialization  
 $\text{error} = 0.2435$   
 $w_0 = w_0 - \alpha \frac{\partial L}{\partial w_0}$   
 $w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}$   
 $\text{error} = 0.2428$

convergence criteria :

1.  $| \frac{\text{Error}_{(n+1)} - \text{Error}_{(n)}}{\text{Error}_{(n)}} | < \epsilon$
2.  $\text{error} = 0.245$
3. Epochs : 1000

Alpha

Error n Min

$$y = w_0 + w_1 x$$

$w_0, w_1$   
turn  $w_0, w_1$  to minimum  
minimum - convex function  
 $y = 0.2 + 0.3x$   
 $y = w_0 + w_1 x$

$$\begin{aligned} &= 0.5 \\ &- 20 \end{aligned}$$

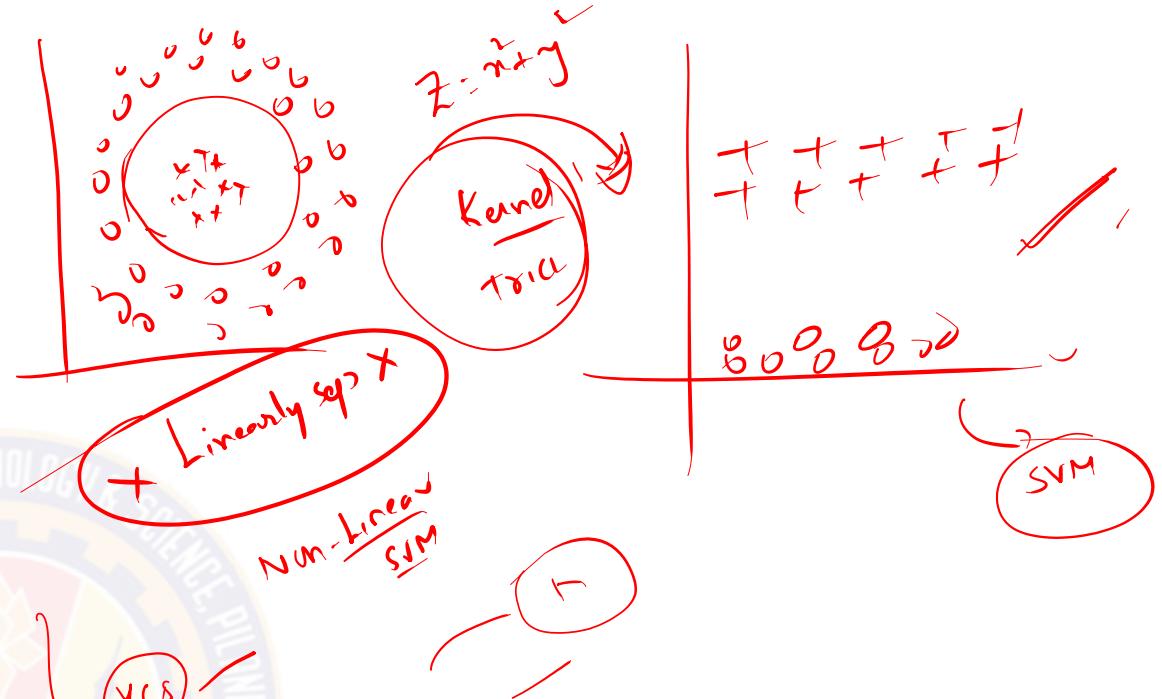
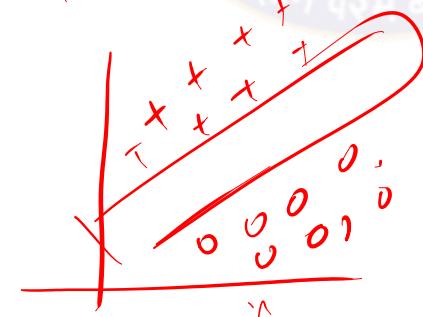


Not linearly separable  
Kernel

Linearly separable  
Ensemble Techniques

$\{$   $\text{GNN}$   
 $\text{DT}$   $\}$   
 $+ \mu_1 - y_{1s}$   
 $+ \mu_2 - y_{1s}$   
Cardiac

$- M_1 - LR - 1$   
 $- M_2 - NB - 1$   
 $- M_3 - SVM - 0$   
 $- M_4 - DT/RF - 1$



# Thank You!

$y = f(x_1, x_2, \dots, x_n)$