

Mathematics required for the module

- Calculus (Differential equations)
- Statistics
- Linear Algebra
- Matrix operations
- Cartesian geometry

Programming languages : Octave (Matlab open source), Python

Tools: Jupyter Notebook (Anaconda Distribution), Octave, Azure ML Studio

Machine Learning – An Introduction



Agenda

- Why ML?
- Introduction to ML
- Types of Machine learning
- Common Issues and Challenges

About

- Subfield of Artificial Intelligence (AI)/Application of optimization
- Name is derived from the concept that it deals with “construction and study of systems that can learn from data”
- Can be seen as building blocks to make computers learn to behave more intelligently
- It is an applied field of study. There are various *techniques* with various *implementations*.
- Supports Other fields in CS (Vision, Data Mining, OCR, NLP, etc.)

Why now?

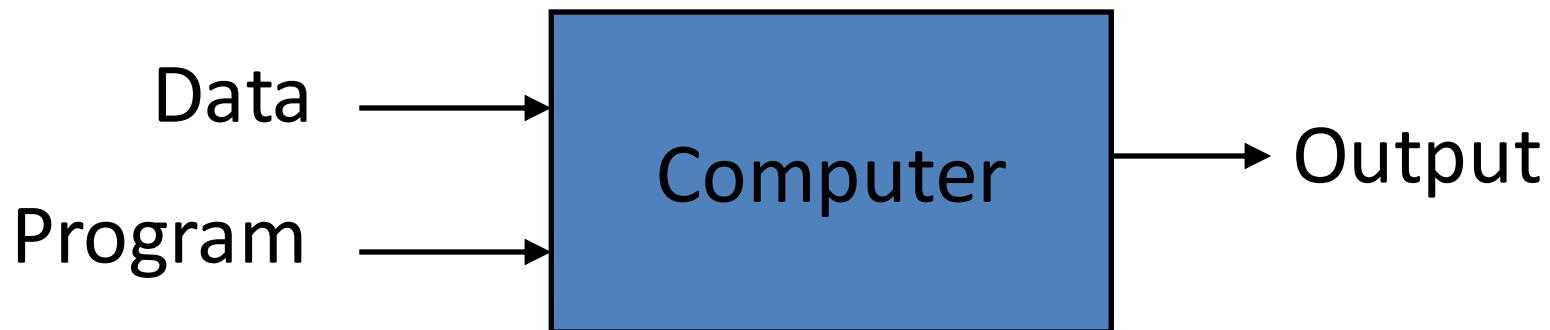
- Flood of available data (especially with the advent of the Internet)
- Increasing computational power (e.g. Multi-core)
- Growing progress in available algorithms and theory developed by researchers
- Increasing support from industries
- Cloud computing

In other words...

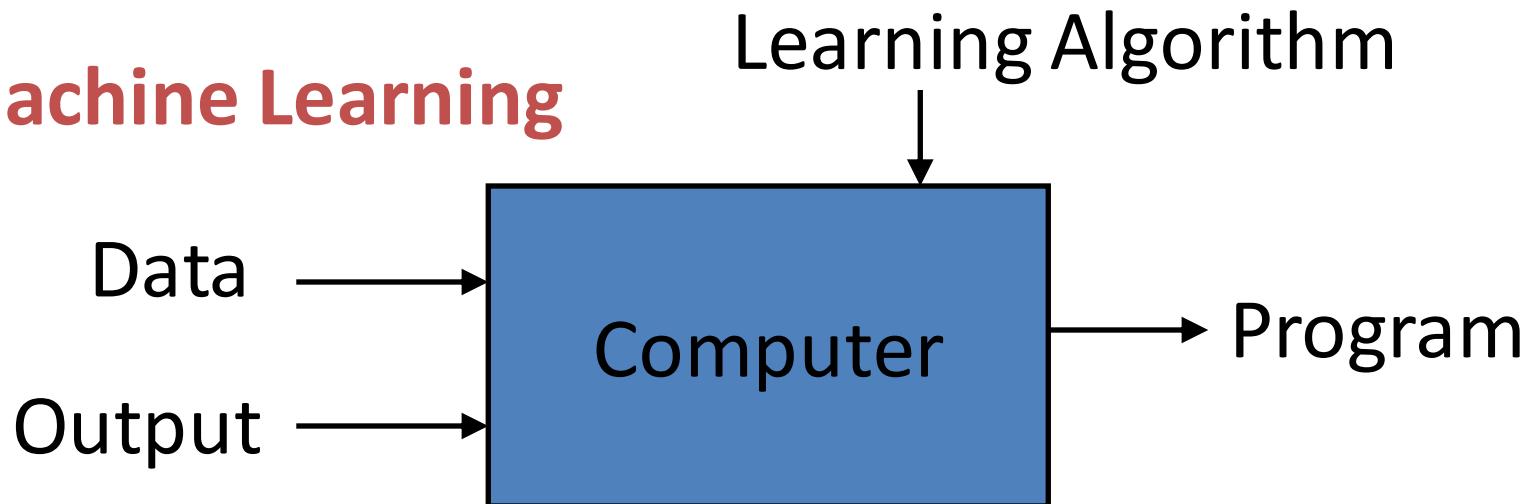
“A computer program is said to learn from experience (E) with some class of tasks (T) and a performance measure (P) if its performance at tasks in T as measured by P improves with E”

ML vs Traditional programming

Traditional Programming



Machine Learning



Motivating Example

Learning to Filter Spam

Example: Spam Filtering

Spam - is all email the user does not want to receive and has not asked to receive

T: Identify Spam Emails

P:

% of spam emails that were filtered

% of ham/ (non-spam) emails that were incorrectly filtered-out

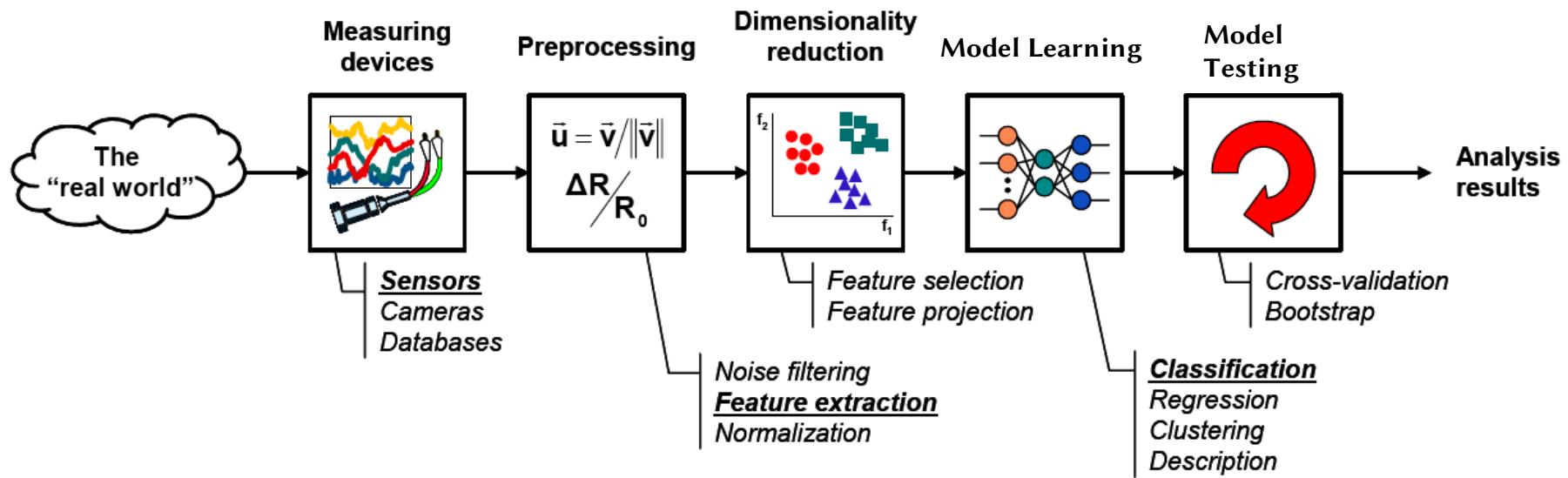
E: a database of emails that were labelled by users



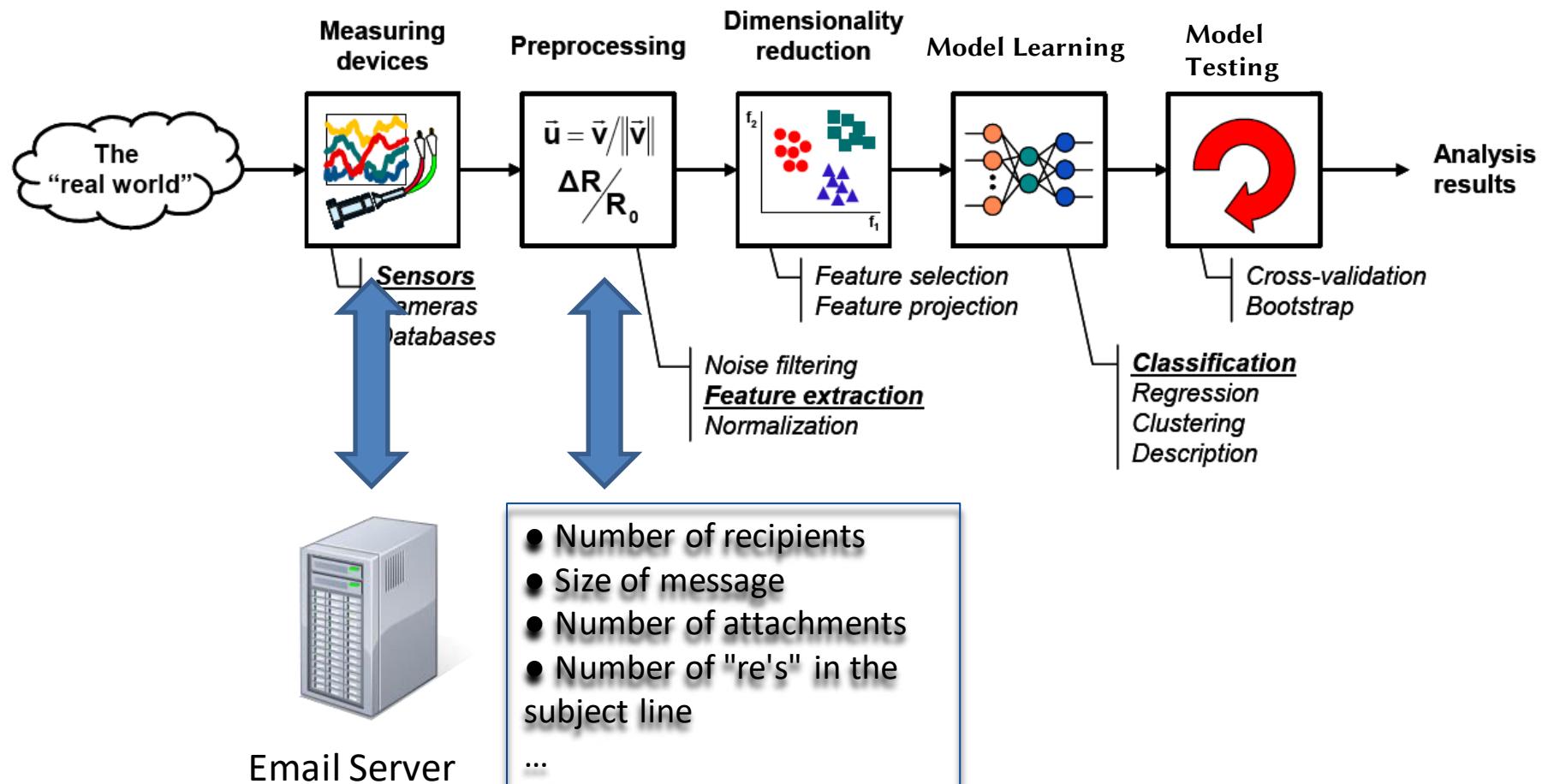
Terminology

- Features
 - The number of features or distinct traits that can be used to describe each item in a quantitative manner.
- Samples
 - A sample is an item to process (e.g. classify). It can be a document, a picture, a sound, a video, a row in database or CSV file, or whatever you can describe with a fixed set of quantitative traits.
- Feature vector
 - is an n-dimensional vector of numerical features that represent some object.
- Feature extraction
 - Preparation of feature vector
 - transforms the data in the high-dimensional space to a space of fewer dimensions.
- Training/Evolution set
 - Set of data to discover potentially predictive relationships.

The Learning Process



The Learning Process in our Example



Data Set

Input Attributes

Target Attribute

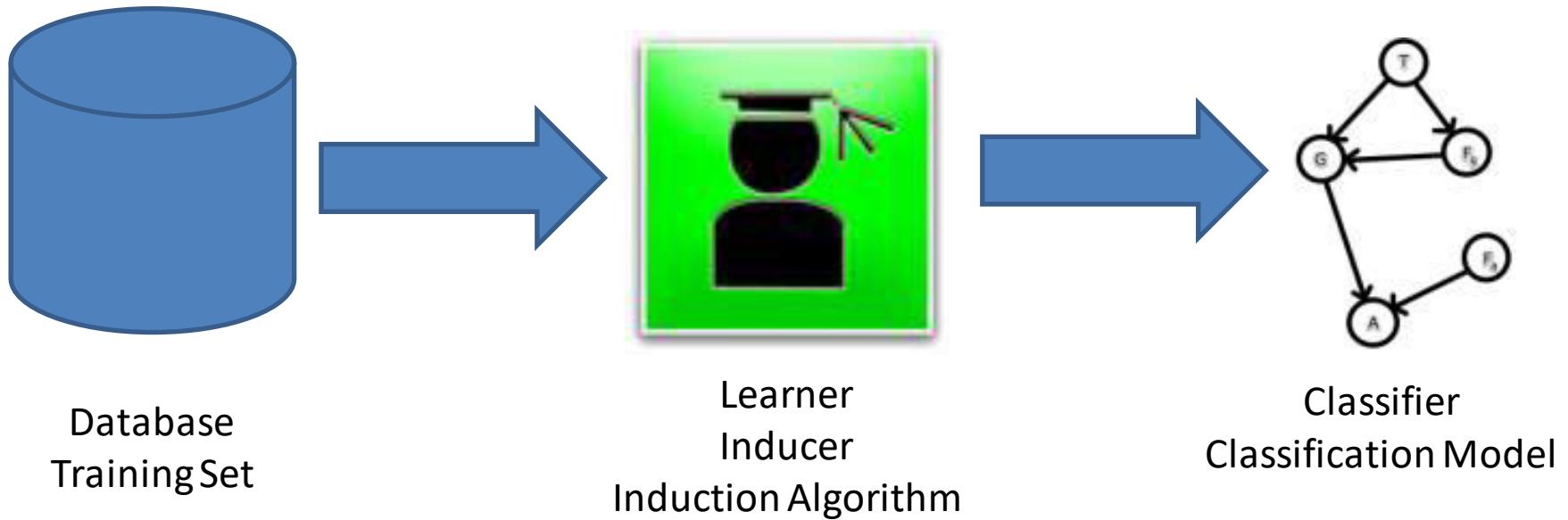
Instances

Number of new Recipients	Email Length (K)	Country (IP)	Customer Type	Email Type
0	2	Germany	Gold	Ham
1	4	Germany	Silver	Ham
5	2	Nigeria	Bronze	Spam
2	4	Russia	Bronze	Spam
3	4	Germany	Bronze	Ham
0	1	USA	Silver	Ham
4	2	USA	Silver	Spam

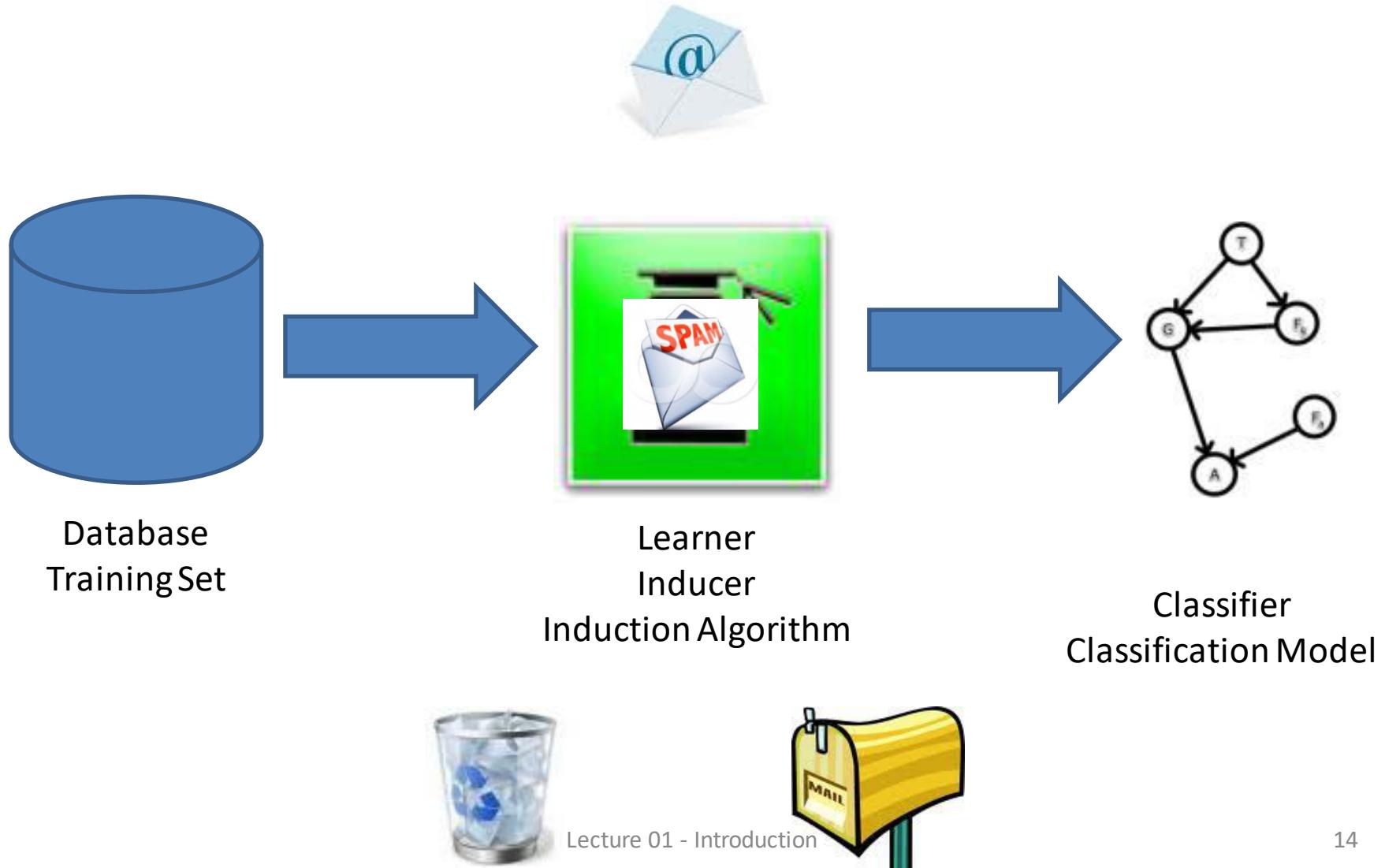
Numeric Nominal Ordinal

The diagram illustrates a data set structure. On the left, a vertical bracket labeled 'Instances' groups seven rows of data, each preceded by an envelope icon with an '@' symbol. Above the table, a bracket labeled 'Input Attributes' spans the first four columns, and another bracket labeled 'Target Attribute' spans the last column. Below the table, arrows point from the labels 'Numeric', 'Nominal', and 'Ordinal' to the second, third, and fourth columns respectively, indicating the data type for each attribute.

Model Learning

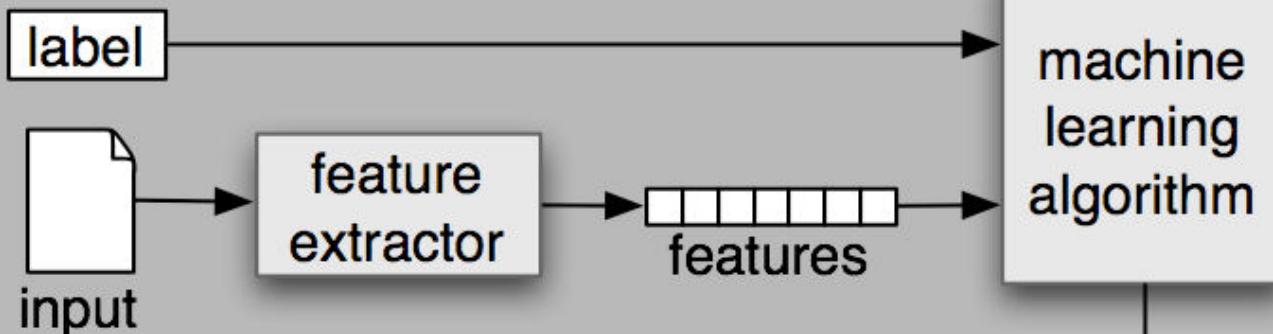


Model Testing

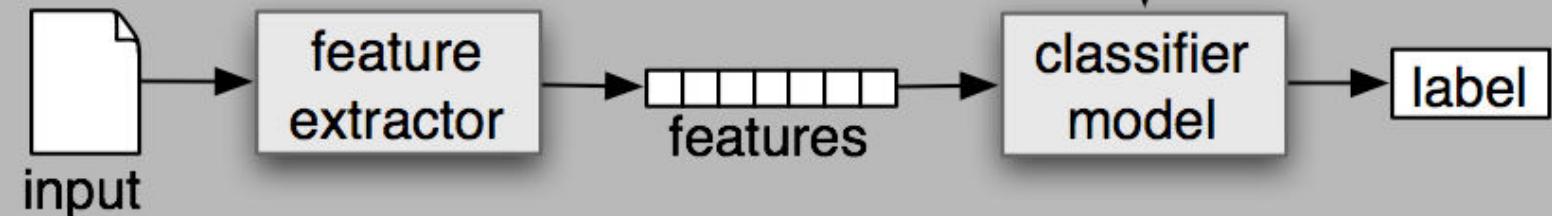


Workflow

(a) Training



(b) Prediction



Categories

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Semi-Supervised Learning
- Bayesian learning

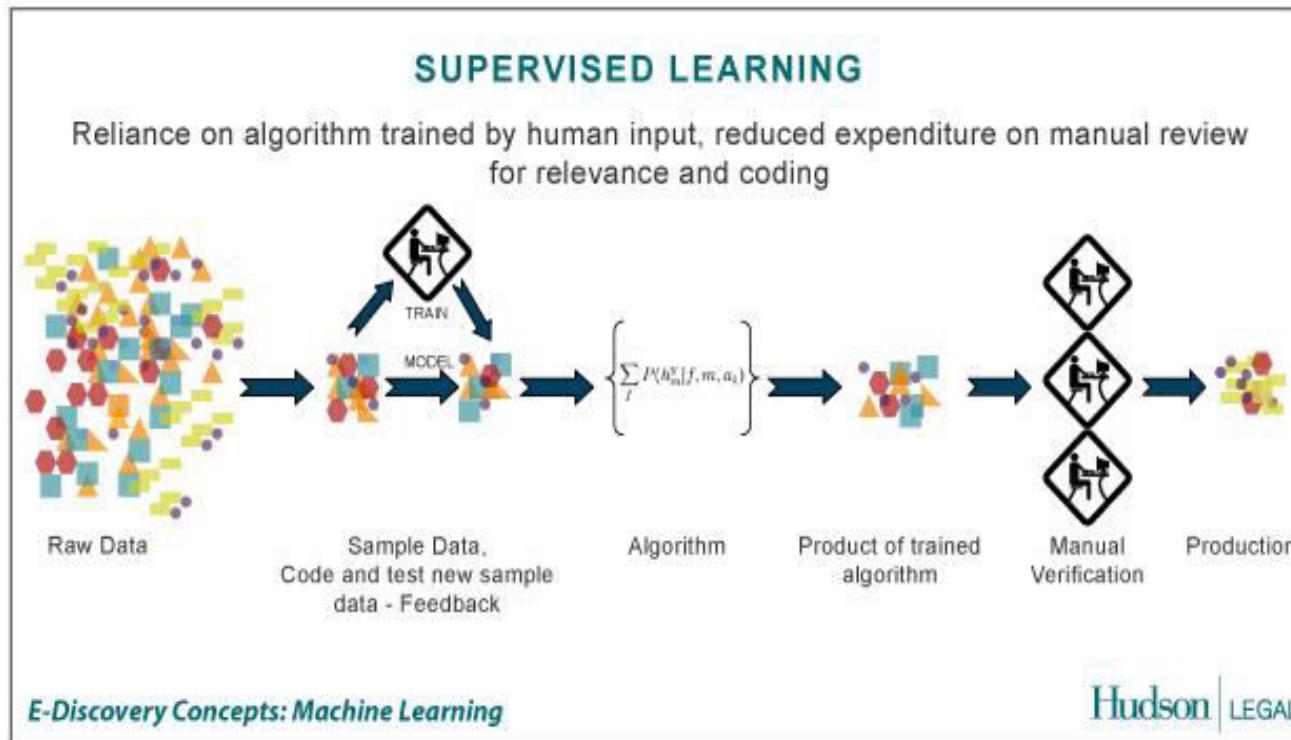
Use-Cases

- Spam Email Detection (Classification)
- Image Search (Similarity/Classification)
- Clustering (KMeans) : Amazon Recommendations
- Autonomous driving/flying : Reinforcement learning

continued...

Supervised Learning (Classification)

- the correct classes (labels) of the training data are known



Supervised learning examples

- A Bank may have borrower details (age, income, gender, etc.) of the past (**features**)
- Also it may have details of the borrowers who defaulted in the past (**labels**)
- Based on the above, can train a classifier to learn the patterns of borrowers who are likely to default on their payments

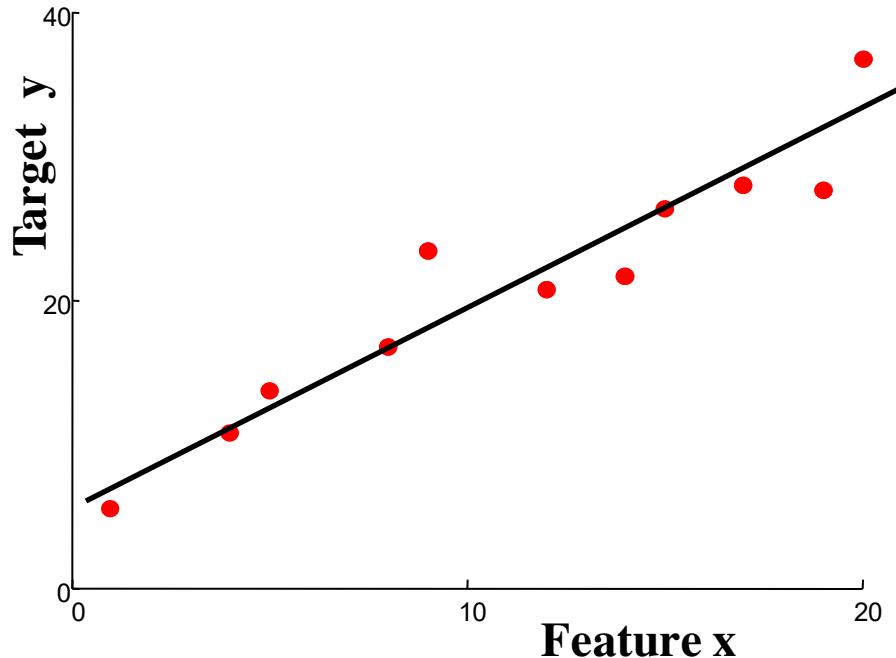
Supervised learning

- Used when the dataset has classes/labels
- Includes a ‘training’ phase with the dataset and a ‘testing’ phase to validate the accuracy of the classifier
- Algorithms – Regression, Support Vector Machines, Neural Networks, Convolutional Neural Networks, Decision Trees, Logistic Regression, Random Forest, Naïve Bayesian, etc.

Supervised learning

- **Regression** – Predict continuous variables (salary, rent)
- **Binary classification** (facial recognition, whether a tumor is benign or malignant)
- **Multi-class classification** (the type of a vehicle, the stage of progression of a cancer – level 1,2,3)

Linear regression



“Predictor”:

Evaluate line:

$$r = \theta_0 + \theta_1 x_1$$

return r

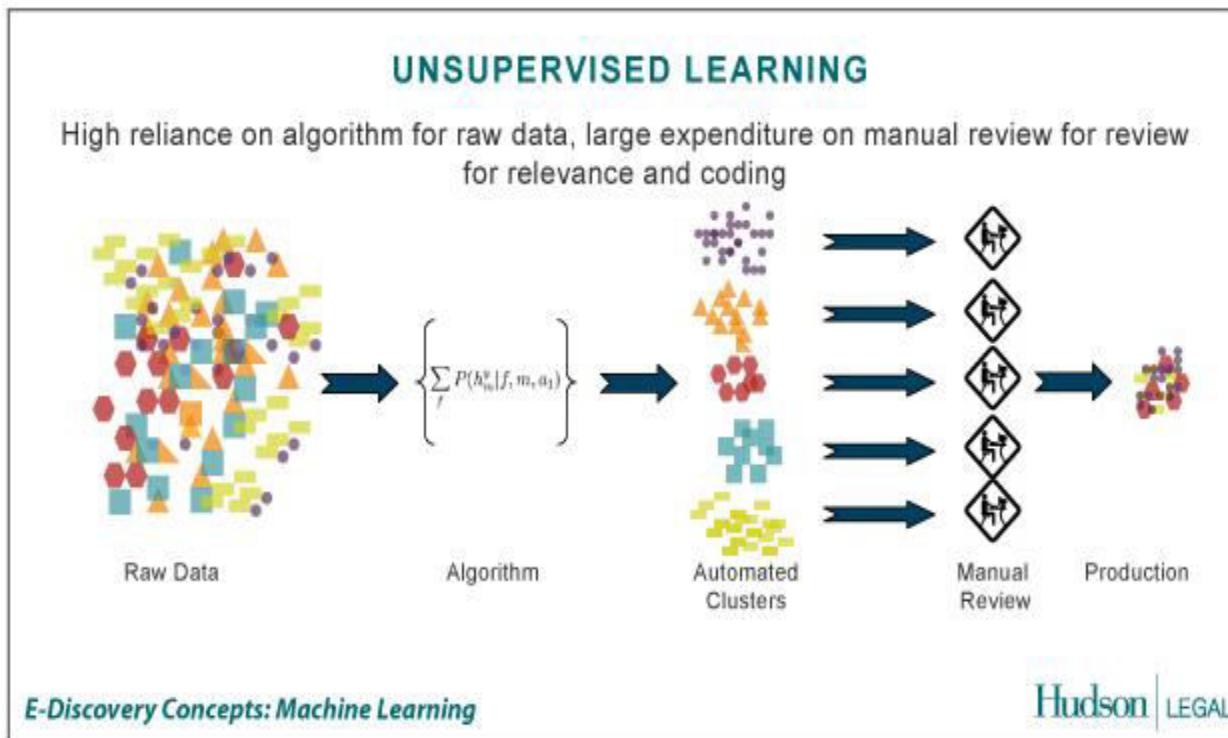
- Define form of function $f(x)$ explicitly
- Find a good $f(x)$ within that family

Unsupervised learning

- Used when the dataset does not have the labels (classes)
- Used to group/cluster the data into clusters, which may then be used for decision making, making recommendations, classification, etc.
- Algorithms – K-means, Self Organizing Maps, Deep belief Networks, etc.

Unsupervised Learning/Clustering

- The correct classes of the training data are not known



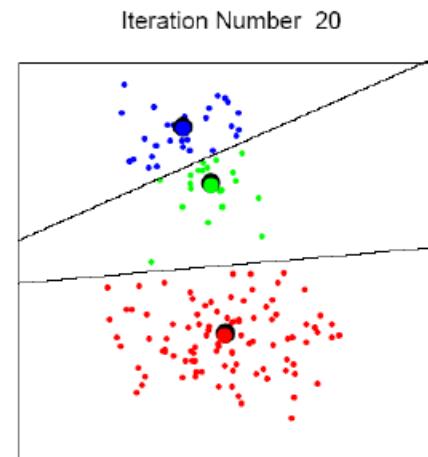
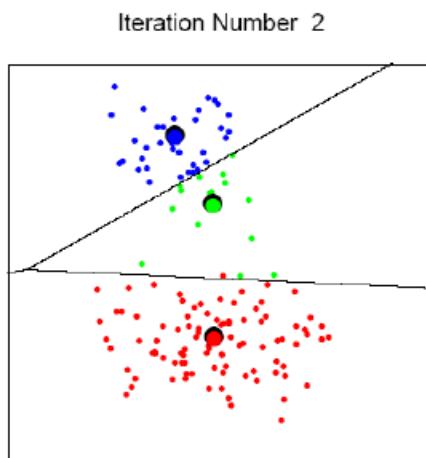
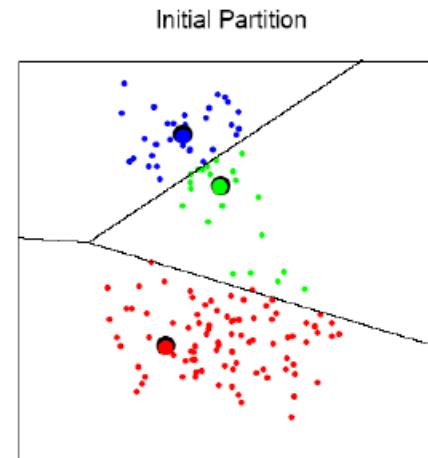
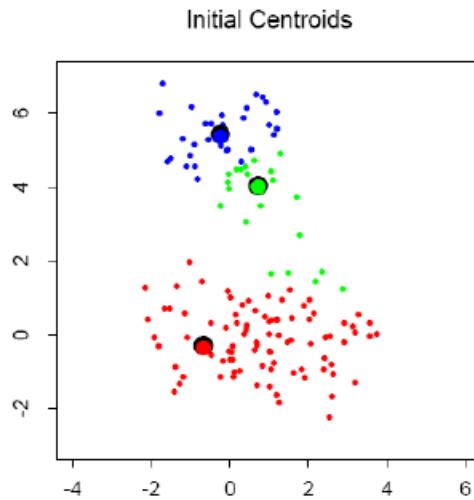
Unsupervised learning examples

- A Supermarket may store each buyer's basket content details (**features**)
- There are **NO** grouping (**labels**)
- Need to group the buyers based on their buying patterns in order to best use the shelf space (recommendation)

Unsupervised learning/Clustering

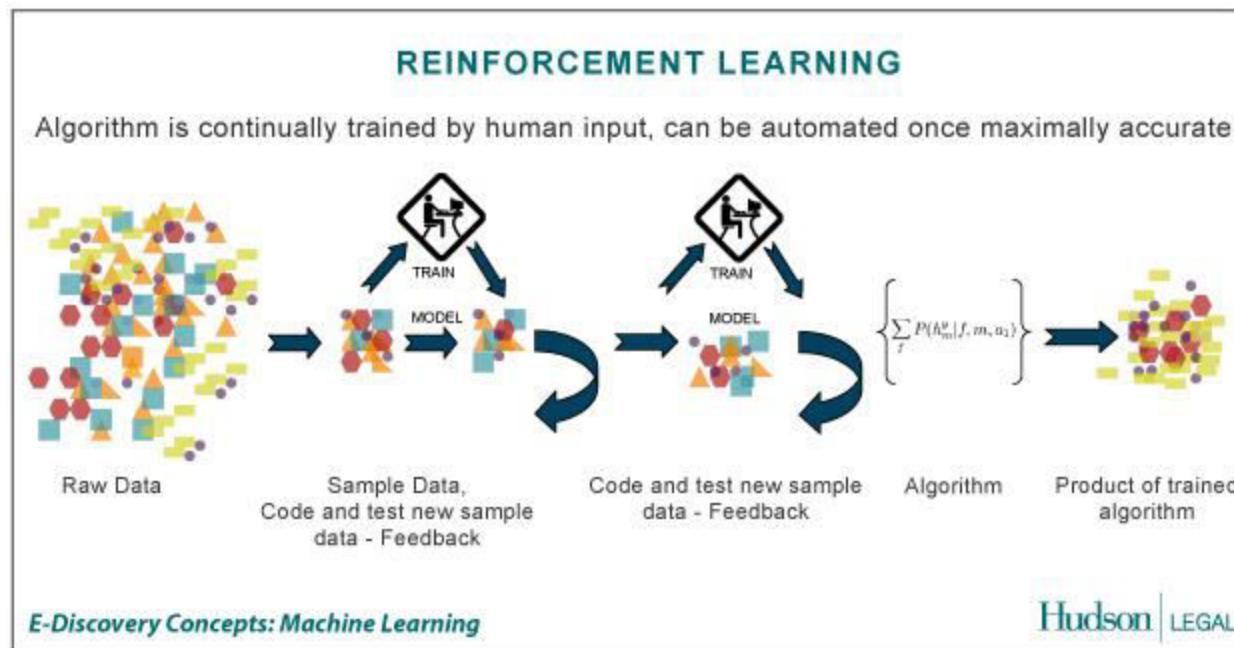
- K-means clustering
- Self organizing maps
- Deep Belief Networks

K-means clustering example



Reinforcement Learning

- Allows the machine or software agent to learn its behavior based on feedback from the environment.
- This behavior can be learnt once and for all, or keep on adapting as time goes by.



Reinforcement learning

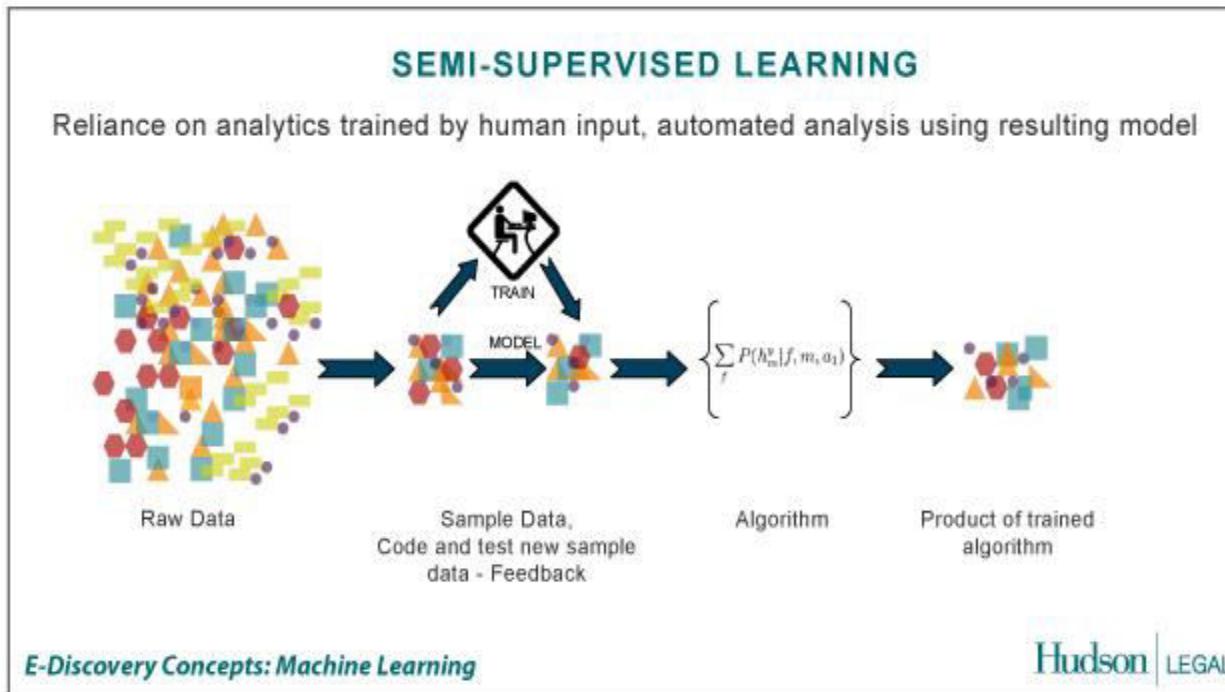
- Can be used when there's no data available
- A reward function is used to measure the reward for a given action
- Based on the reward values, a probability distribution can be obtained for a given set of functions
- This can continued over time and also can be deployed in both single/multi-agent systems
- Algorithms – Actor Critic learning, Q learning, Monte-carlo methods, etc.

Reinforcement learning examples

- A group of robots have been deployed in an unknown territory
- The objective is for them to collaboratively find the navigation path to reach a particular destination/goal
- Can use reinforcement learning where achieving the goal/getting closer to the goal gives a positive reward.
Negative reward otherwise
- Can share the information among robots (multi-agent system)

Semi-Supervised Learning

- A Mix of Supervised and Unsupervised learning



Semi-supervised learning

- Labeled data is expensive/difficult to get
- Unlabeled data is cheap/easier to get
- The idea is to use smaller amount of labelled data with larger amount of unlabeled data to creating the training/testing datasets
- Algorithms - Self Training, Generative models

Semi-Supervised Support Vector
Machines, etc.

Semi-supervised learning applications

- Web page classification
- Speech to text conversion
- Video/image generation

Bayesian learning

- ➊ Bayes Theorem:

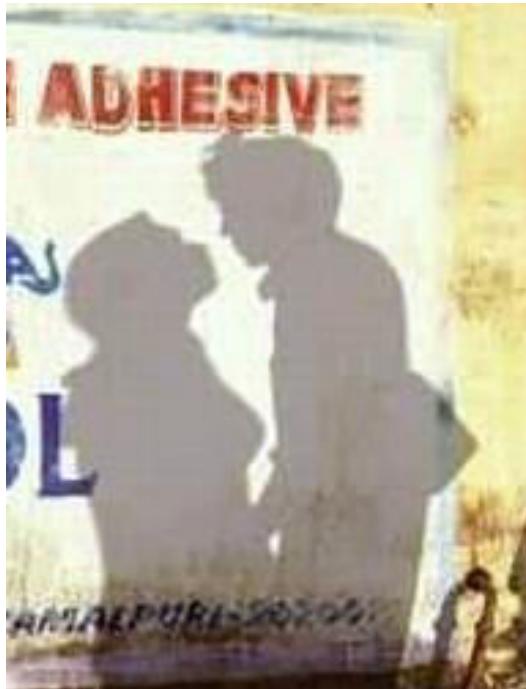
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- Naïve bayesian, Multinomial bayesian, Bayesian networks, Hidden markov model
- Applications: Sentiment analysis, medical diagnosis
- Needs some initial knowledge

Dimensionality Reduction

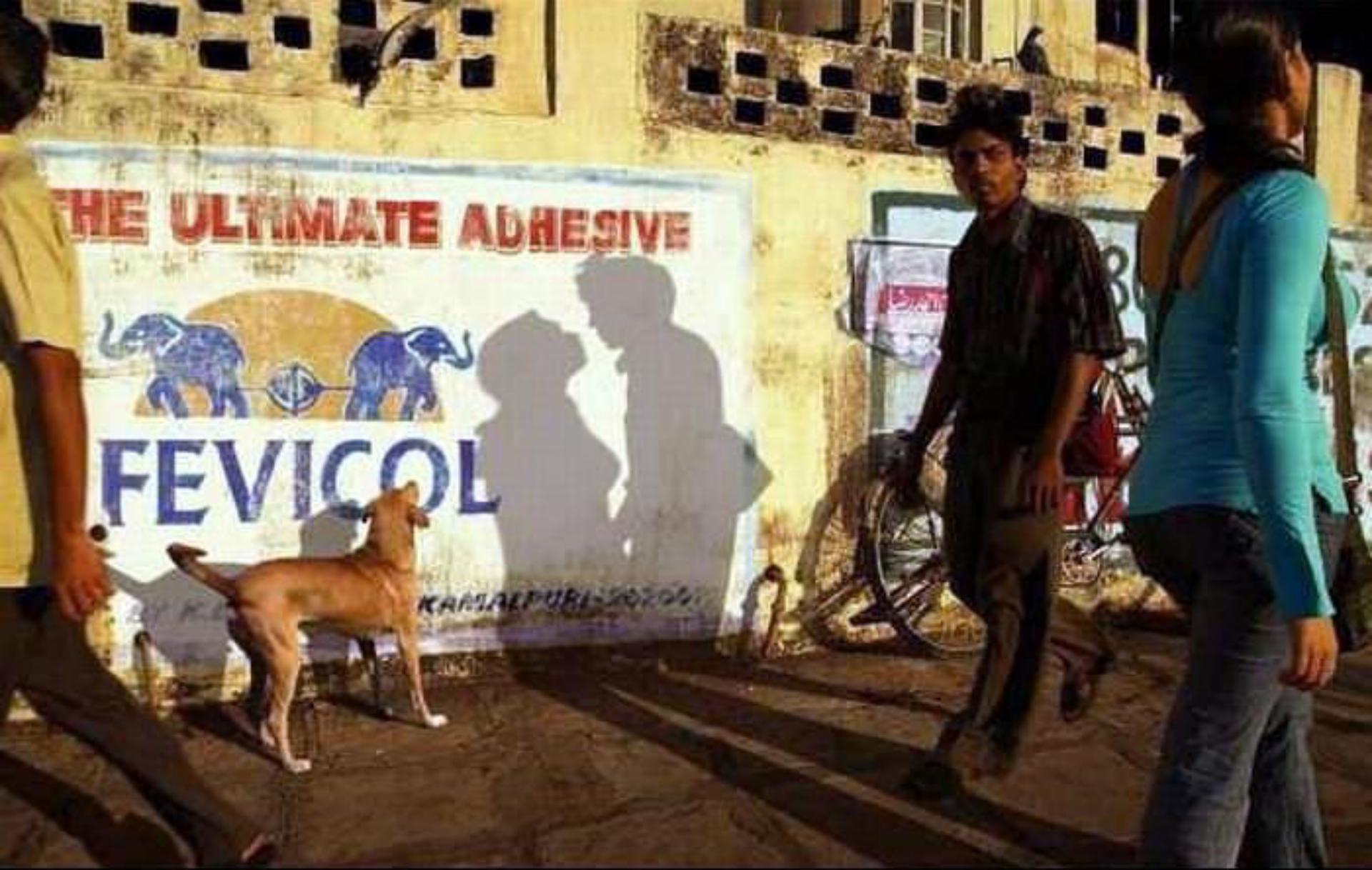
- Too many attributes!!!
- Which attributes to choose for the feature vector?
- 3D picture → 2D Image
- Dimensionality Reduction
- Principal Component Analysis (PCA) is a commonly used technique

Dimensionality Reduction - Challenges



While **dimensionality reduction** is an important tool in machine learning/data mining, we must always be aware that it can distort the data in misleading ways.

Above is a two dimensional projection of an intrinsically three dimensional world....



Original photographer unknown

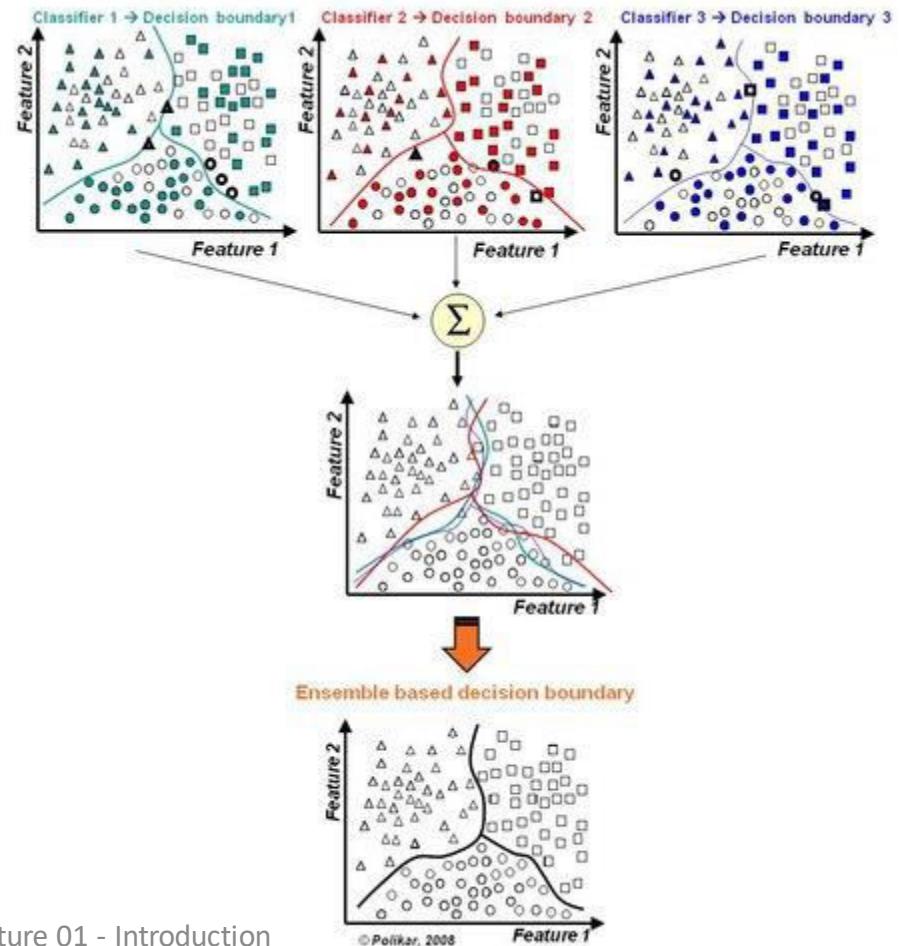
See also www.cs.gmu.edu/~jessica/DimReducDanger.htm

Lecture 01 Introduction

(c) eamonn keogh 37

Ensemble Learning

- Often, multiple classifiers need to be combined to solve a real-world problem.



Machine learning on Big Data and GPGPU computing

- Use large unstructured data sets for learning
(Call records, Social media data, etc.)
- Two main approaches
- Use a Big Data Platform (e.g. Apache Hadoop, Apache Spark)
- Use a Cloud based Big Data Analytics platform
(Amazon AWS Services, Microsoft Azure ML)
- *GPUs to speed up the learning (particularly in Deep learning)*

Things to consider in Selecting a ML Algorithm

- If there's an algorithmic way instead of ML, use it!!! (ML is messy)
- Refer the literature!!!
- Try different ML algorithms (no single algorithm is the best)
- Check the dataset against the usage/strength of each algorithm (e.g. RNNs, ARIMA is good in time-series predictions)
- Be mindful of 'external factors' (e.g. seasonal effects, RL if you don't have data, Clustering if you have unlabeled data, etc.)
- Test your algorithm(s) with test data and select the best performing one for production (include the test results in your thesis/publications)
- No algorithm will be perfect! (There will be an error. The objective is to keep the error at an acceptable rate)

Popular Frameworks/Tools

- Scikit-learn - Python (Anaconda Python Distribution)
- R (R studio)
- Matlab/Octave (can export DLLs)
- Weka (Java based)
- Java OpenNLP/Python NLTK (Natural language processing + ML)
- Apache Spark (part of the Apache Hadoop platform)
- Google Tensorflow (Python library for Deep neural networks)
- Apache Keras (Python library of neural networks)
- Theano (Python library for Multicore processing of DNNs)
- Amazon AWS Services/Microsoft Azure ML (Cloud based ML)

Commonly used python libraries

- NumPy
 - Matrix algebra
- Pandas
 - Data Frames, Series
- Matplotlib
 - Visualization



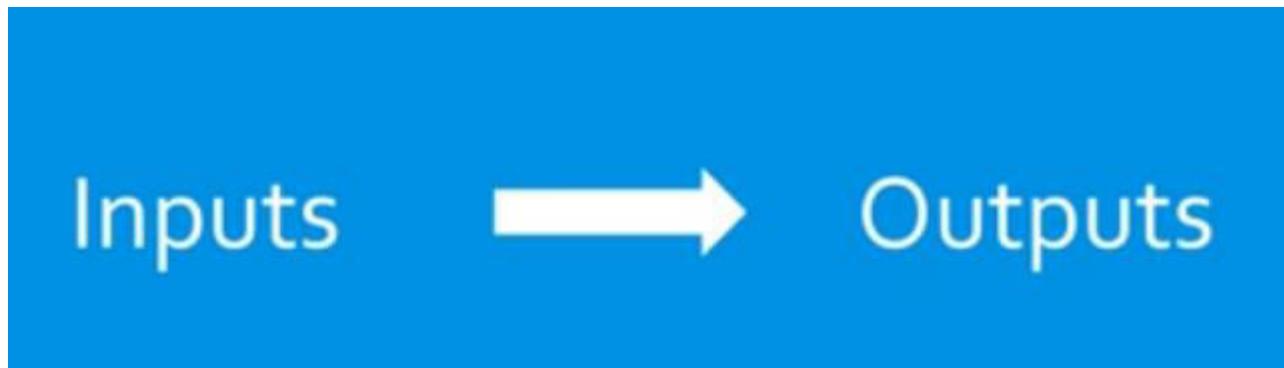
Resources

- Coursera – Andrew Ng. Machine Learning
- Udacity – Intro to Machine Learning,
Reinforcement Learning
- Python Machine Learning – Sebastian Raschka
- Advance Machine Learning with Python –
John Hearty
- Machine Learning – Tom Mitchell
- Many more!!!

Questions ?

Introduction to optimization

- Choosing a set of inputs to produce the best possible output



In practice

- Allocate a set of scarce resources
- Design a system optimally
- Choosing a set of control variables

So.. What is mathematical optimization anyway?

- Optimization comes from the same root as optimal , which means best. When you optimize something, you are "making it best".
- But "best" can vary. If you're a football player, you might want to maximize your running yards, and also minimize your fumbles. Both maximizing and minimizing are types of optimization problems.

Mathematical Optimization in the "Real World"

Mathematical Optimization is a branch of applied mathematics which is useful in many different fields. Here are a few examples:

- Manufacturing
- Production I
- Inventory control
- Transportation
- Scheduling
- Networks
- Finance
- Engineering
- Mechanics
- Economics
- Control engineering
- Marketing
- Policy Modeling

- Machine learning is an application of optimization

Warehouse Placement

Warehouse Location



Minimize
Shipment
Time

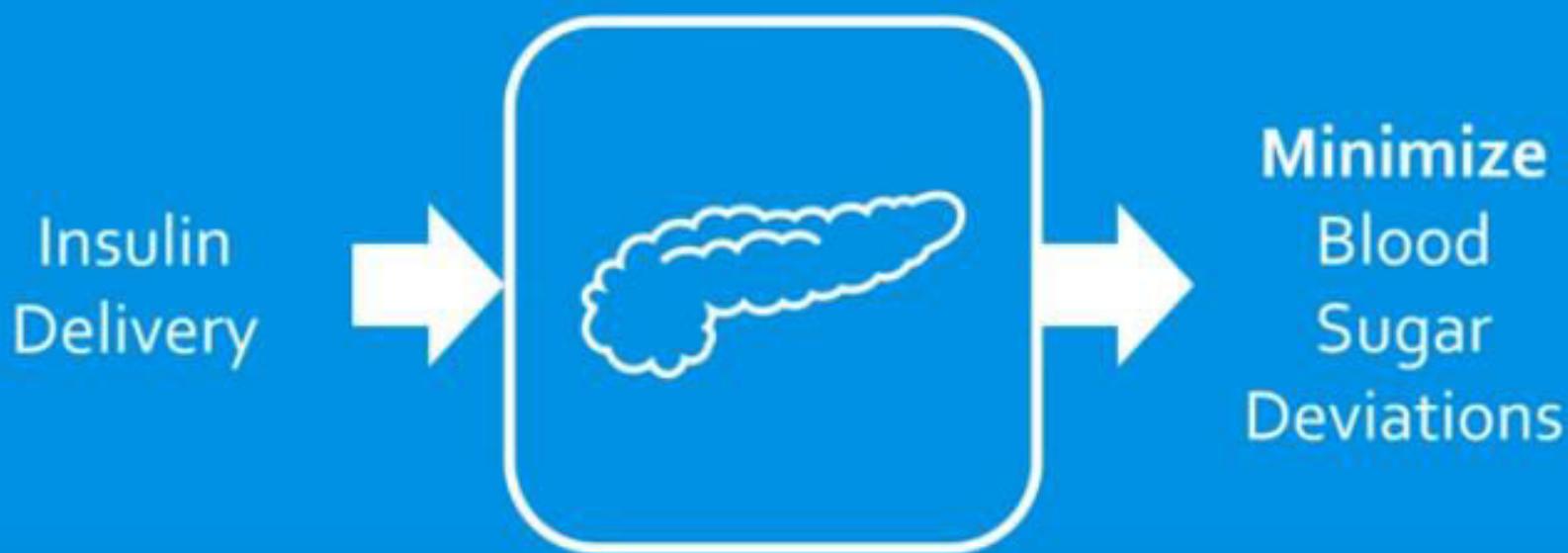
Bridge Construction

Design



Maximize
Load
Bearing

Artificial Pancreas



What is Optimization?

- **Optimization** is the mathematical discipline which is concerned with finding the maxima and minima of functions, possibly subject to constraints.

Optimization algorithm

- Is a set of steps that is used to solve an optimization problem

Optimization Vocabulary

Your basic optimization problem consists of...

- The objective function, $f(x)$, which is the output you're trying to maximize or minimize.

Optimization Vocabulary

Your basic optimization problem consists of...

- The objective function, $f(x)$, which is the output you're trying to maximize or minimize.
- Variables, $x_1 x_2 x_3$ and so on, which are the inputs —things you can control. They are abbreviated x_n to refer to individuals or x to refer to them as a group.

Optimization Vocabulary

Your basic optimization problem consists of...

- The objective function, $f(x)$, which is the output you're trying to maximize or minimize.
- Variables, $x_1 x_2 x_3$ and so on, which are the inputs — things you can control. They are abbreviated x_n to refer to individuals or x to refer to them as a group.
- Constraints, which are equations that place limits on how big or small some variables can get. Equality constraints are usually noted $h_n(x)$ and inequality constraints are noted $g_n(x)$.

Optimization Vocabulary

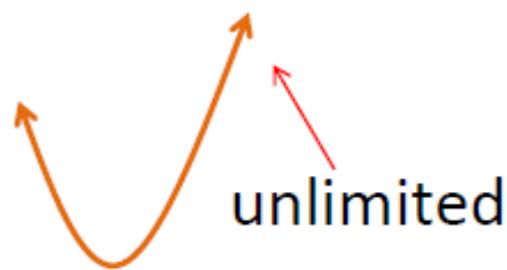
A football coach is planning practices for his running backs.

- His main goal is to maximize running yards — this will become his **objective function**.
- He can make his athletes spend practice time in the weight room; running sprints; or practicing ball protection. The amount of time spent on each is a **variable**.
- However, there are limits to the total amount of time he has. Also, if he completely sacrifices ball protection he may see running yards go up, but also fumbles, so he may place an upper limit on the amount of fumbles he considers acceptable. These are **constraints**.

Note that the variables influence the objective function and the constraints place limits on the domain of the variables.

Types of Optimization Problems

- Some problems have constraints and some do not



Types of Optimization Problems

- Some problems have constraints and some do not
- There can be one variable or many

The diagram illustrates a set of variables arranged in two rows. The top row contains four variables: x_1 (top-left), x_4 (center-left), x_2 (center-right), and x_3 (top-right). The bottom row contains four variables: x_7 (bottom-left), x_6 (center-left), x_5 (center-right), and x_8 (bottom-right). This visual representation serves as a conceptual model for optimization problems involving multiple variables.

Types of Optimization Problems

- Some problems have constraints and some do not
- There can be one variable or many
- Variables can be discrete (integers) or continuous





Continuous



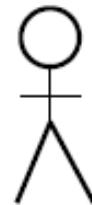
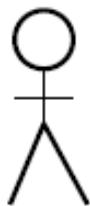
Discrete



Binary

Types of Optimization Problems

- Some problems have constraints and some do not
- There can be one variable or many
- Variables can be discrete (integers) or continuous
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).



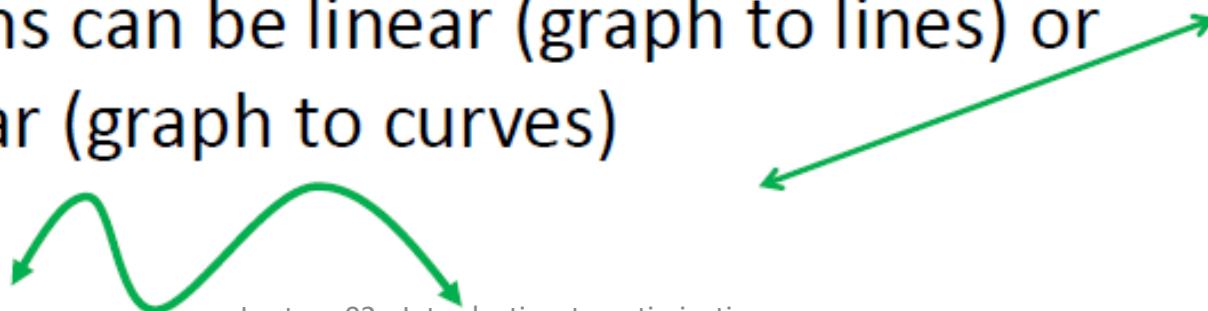
Types of Optimization Problems

- Some problems have constraints and some do not
- There can be one variable or many
- Variables can be discrete (integers) or continuous
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).
- Systems can be deterministic (specific causes produce specific effects) or stochastic (involve randomness/ probability).



Types of Optimization Problems

- Some problems have constraints and some do not
- There can be one variable or many
- Variables can be discrete (integers) or continuous
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).
- Systems can be deterministic (specific causes produce specific effects) or stochastic (involve randomness/probability).
- Equations can be linear (graph to lines) or nonlinear (graph to curves)



What do we optimize?

- A real function of n variables

$$f(x_1, x_2, \dots, x_n)$$

- with or without constraints

Unconstrained optimization

$$\min f(x, y) = x^2 + 2y^2$$

Optimization with constraints

$$\min f(x, y) = x^2 + 2y^2$$

$$x > 0$$

or

$$\min f(x, y) = x^2 + 2y^2$$

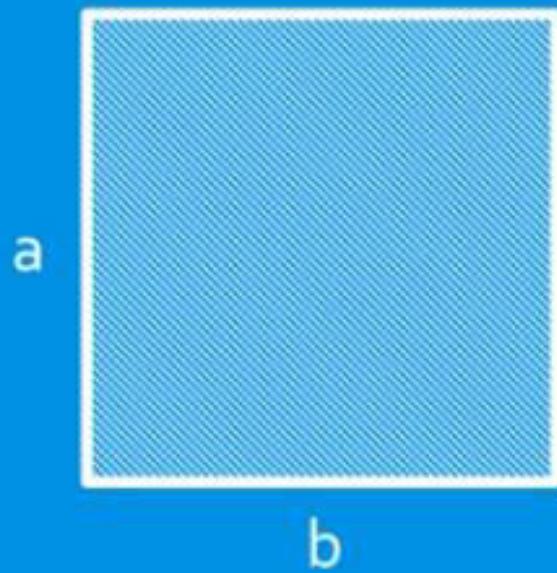
$$-2 < x < 5, y \geq 1$$

or

$$\min f(x, y) = x^2 + 2y^2$$

$$x + y = 2$$

Lagrangian
Multipliers may
be used to
transform the
constraints to a
dual problem
without
constraints



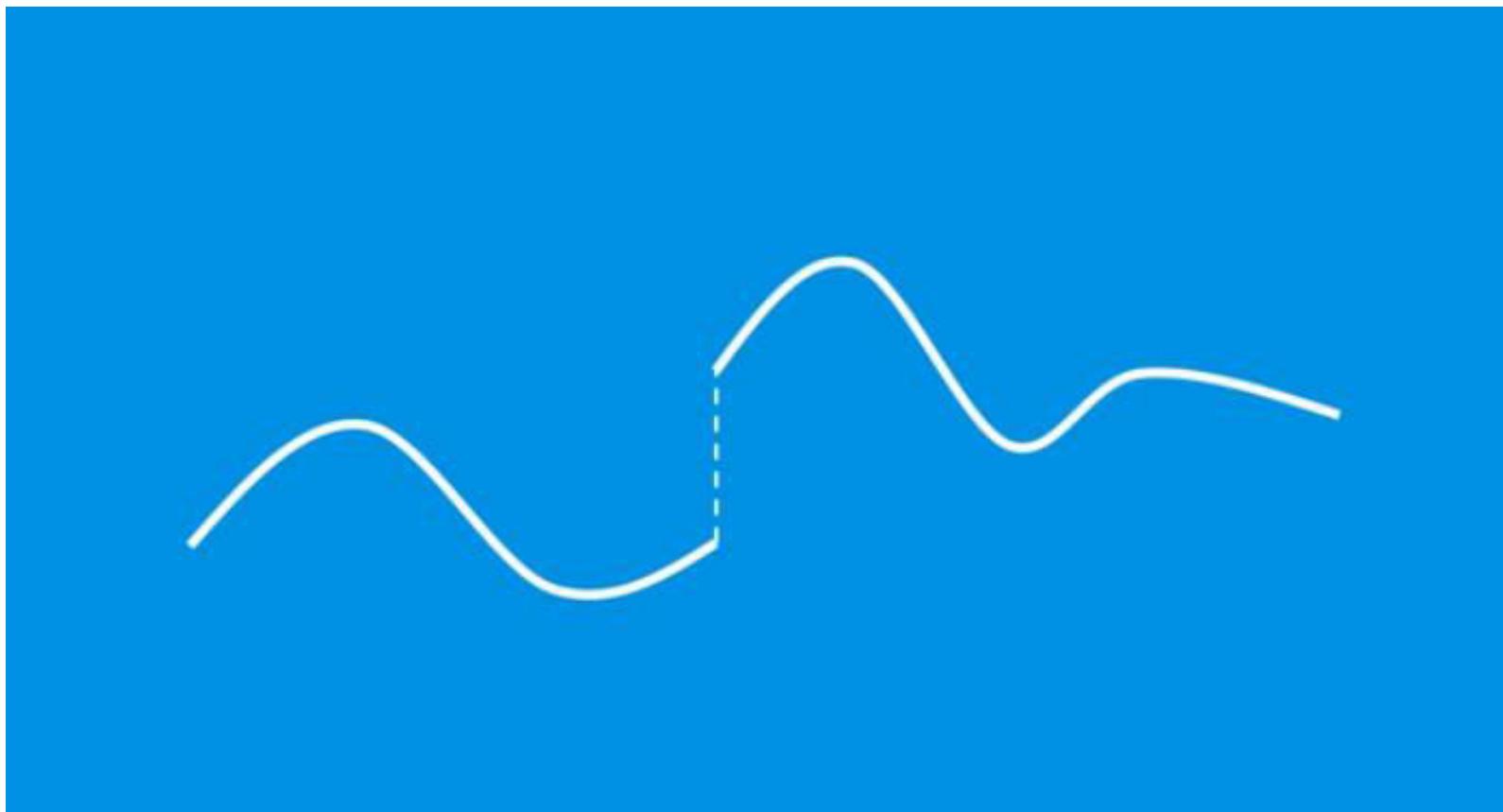
Inequality Constraint

$$a \times b \leq 5$$

Equality Constraint

$$a + b = 10$$

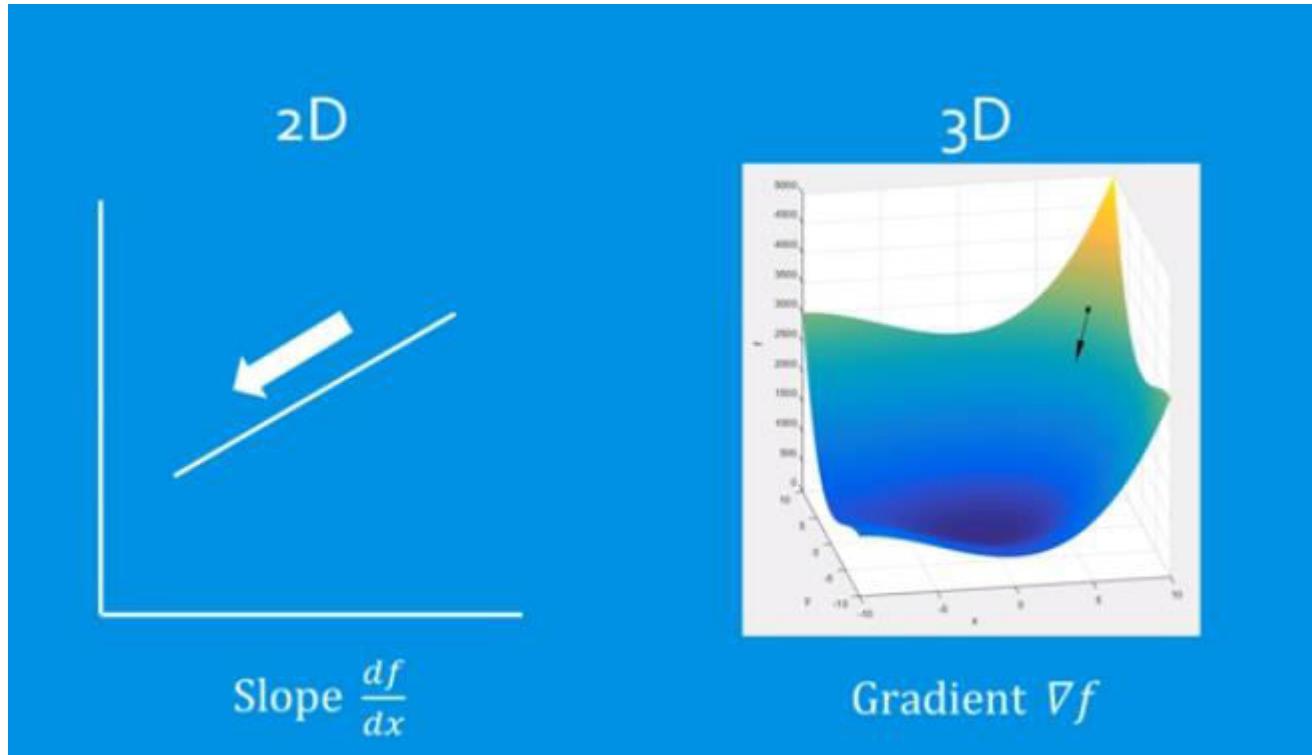
$$\frac{a}{a+b} = 3$$



Optimization algorithms

- Gradient based algorithms
- Gradient free algorithms

Gradient



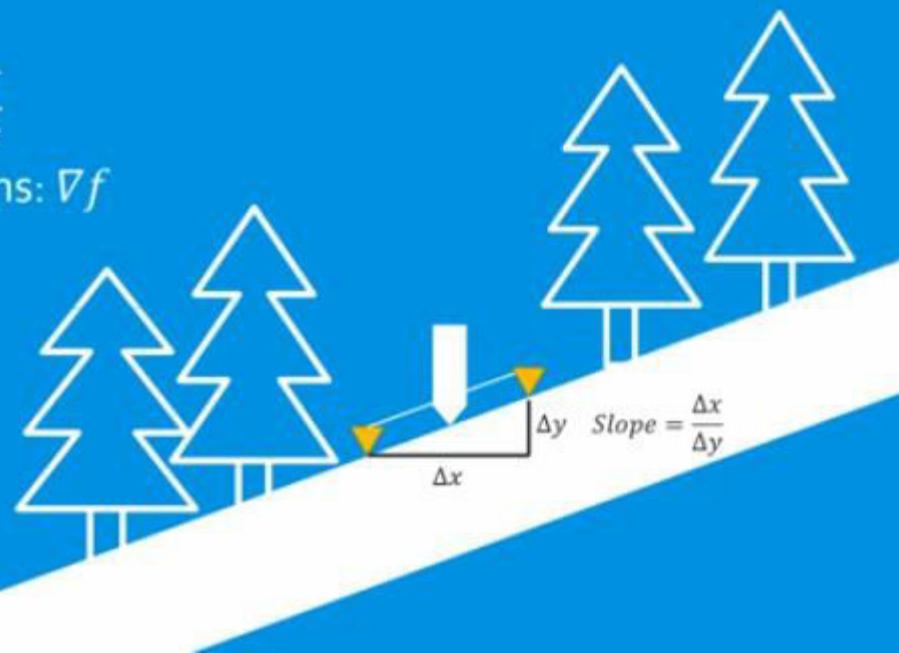




Search Direction

One Dimension: $\frac{df}{dx}$

Multiple Dimensions: ∇f



Search Direction



Step Size





3 Steps

Search Direction

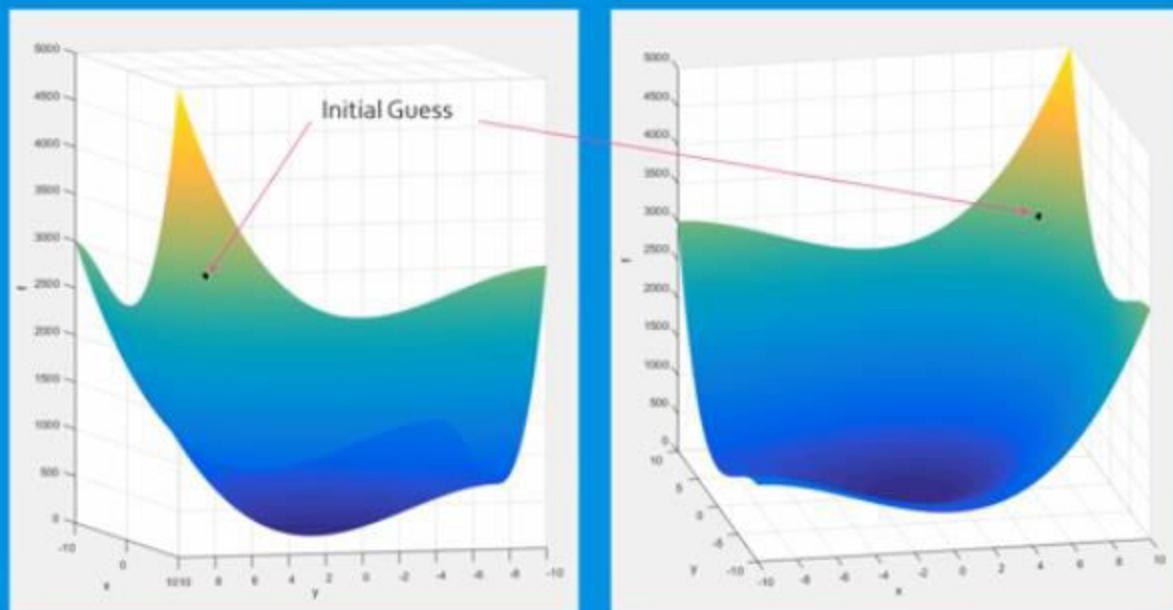
Step Size

Convergence Check

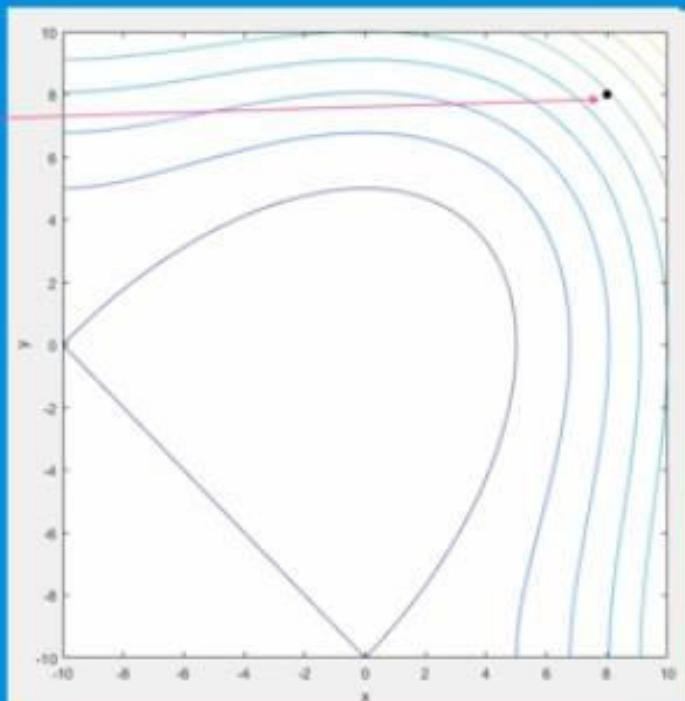
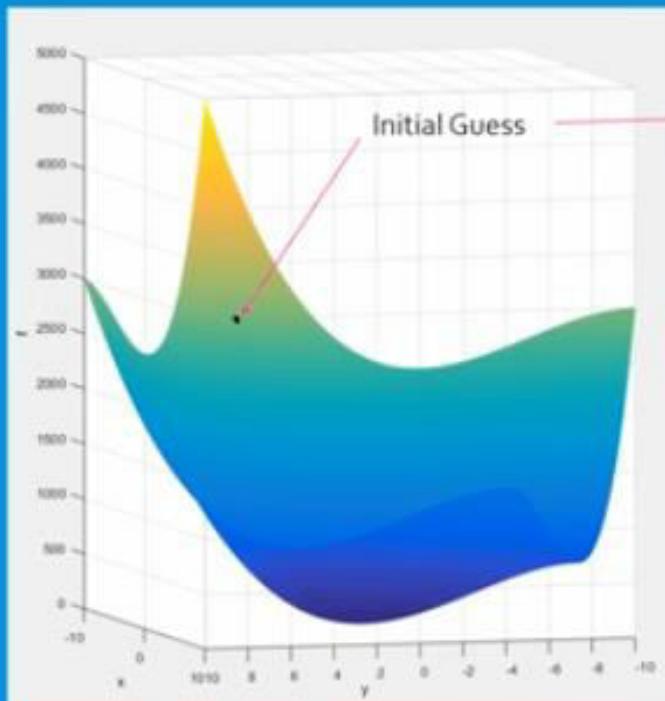
Minimize

$$f(x, y) = x^3 + 15x^2 + y^3 + 15y^2$$

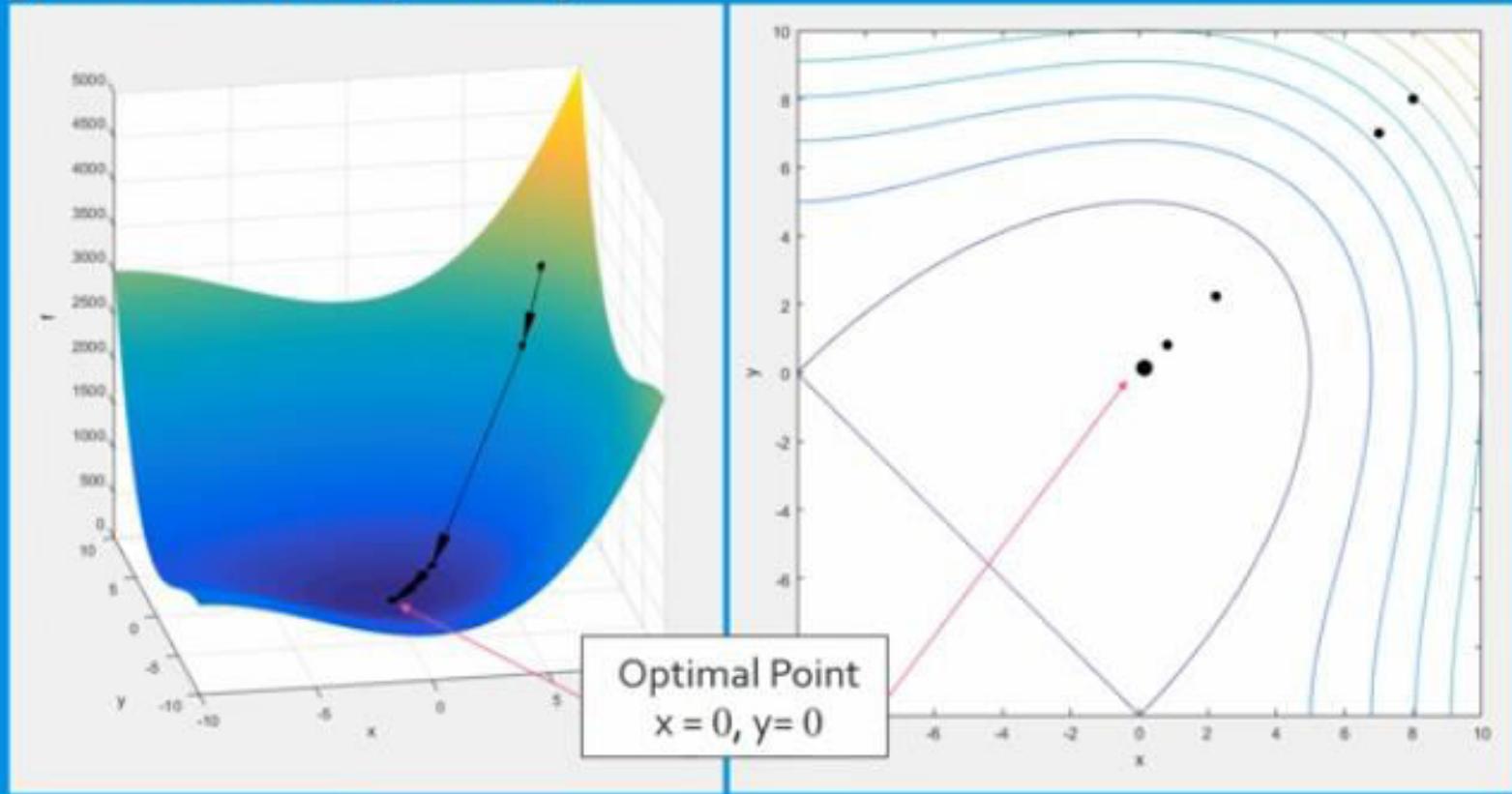
$$f(x, y) = x^3 + 15x^2 + y^3 + 15y^2$$



$$f(x, y) = x^3 + 15x^2 + y^3 + 15y^2$$

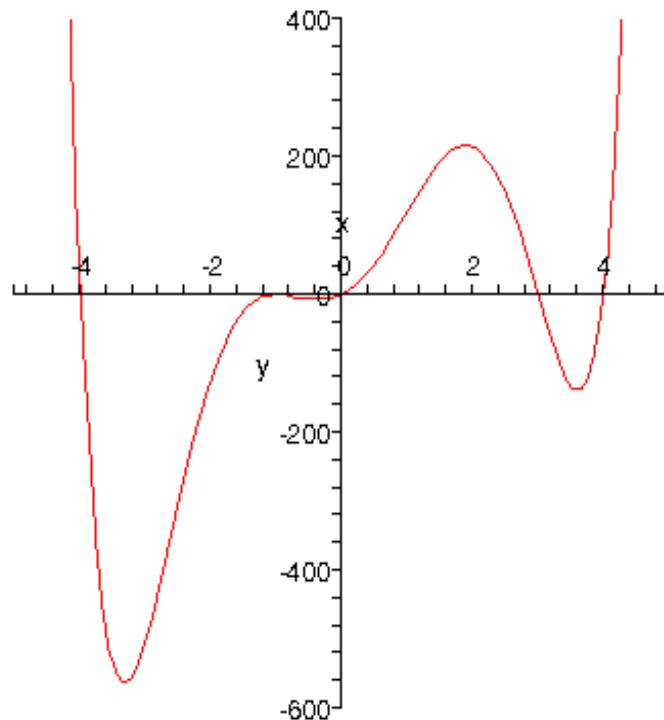


$$f(x, y) = x^3 + 15x^2 + y^3 + 15y^2$$



Lets Optimize

- Suppose we want to find the minimum of the function



- Gradient descent
- Stochastic gradient descent
- Coordinate descent
- Frank—Wolfe algorithm
- Landweber iteration
- Random coordinate descent
- Conjugate gradient method
- Derivation of the conjugate gradient method
- Nonlinear conjugate gradient method
- Biconjugate gradient method
- Biconjugate gradient stabilized method

GRADIENT BASED OPTIMIZATION

- Pros
 - Widely Used
 - Fast
 - Scales Well
- Cons
 - Requires smooth gradients
 - Cost of computing gradients
 - Local minima

Gradient free algorithms

- Do not require derivatives
- Can be used for optimization probs where derivatives can't be used
- Makes them more flexible
- Generally much slower than gradient based algorithms

X

No Derivatives Needed



SUMMARY

- Common Gradient Free Algorithms
- Genetic Algorithms, Particle Swarm
- Simulated Annealing, Nelder-Mead
- Easy to Implement
- No derivatives required
- Slow for large problems
- No guarantee of optimal solution

Optimization Cont.

- A prerequisite to the use of optimization techniques is to mathematically model a system.
- Optimization problem components:
 - Decision variables
 - Objective function (to maximize or minimize)
 - Constraints (requirements or limitations)
- Find the values of the decision variables,
that maximize/ minimize object function value,
while staying within the constraint.

Optimization Cont.

- A prerequisite to the use of optimization techniques is to mathematically model a system.
- Optimization problem components:
 - Decision variables
 - Objective function (to maximize or minimize)
 - Constraints (requirements or limitations)
- Find the values of the decision variables, **Single Objective**
that maximize/ minimize object function value,
while staying within the constraint.

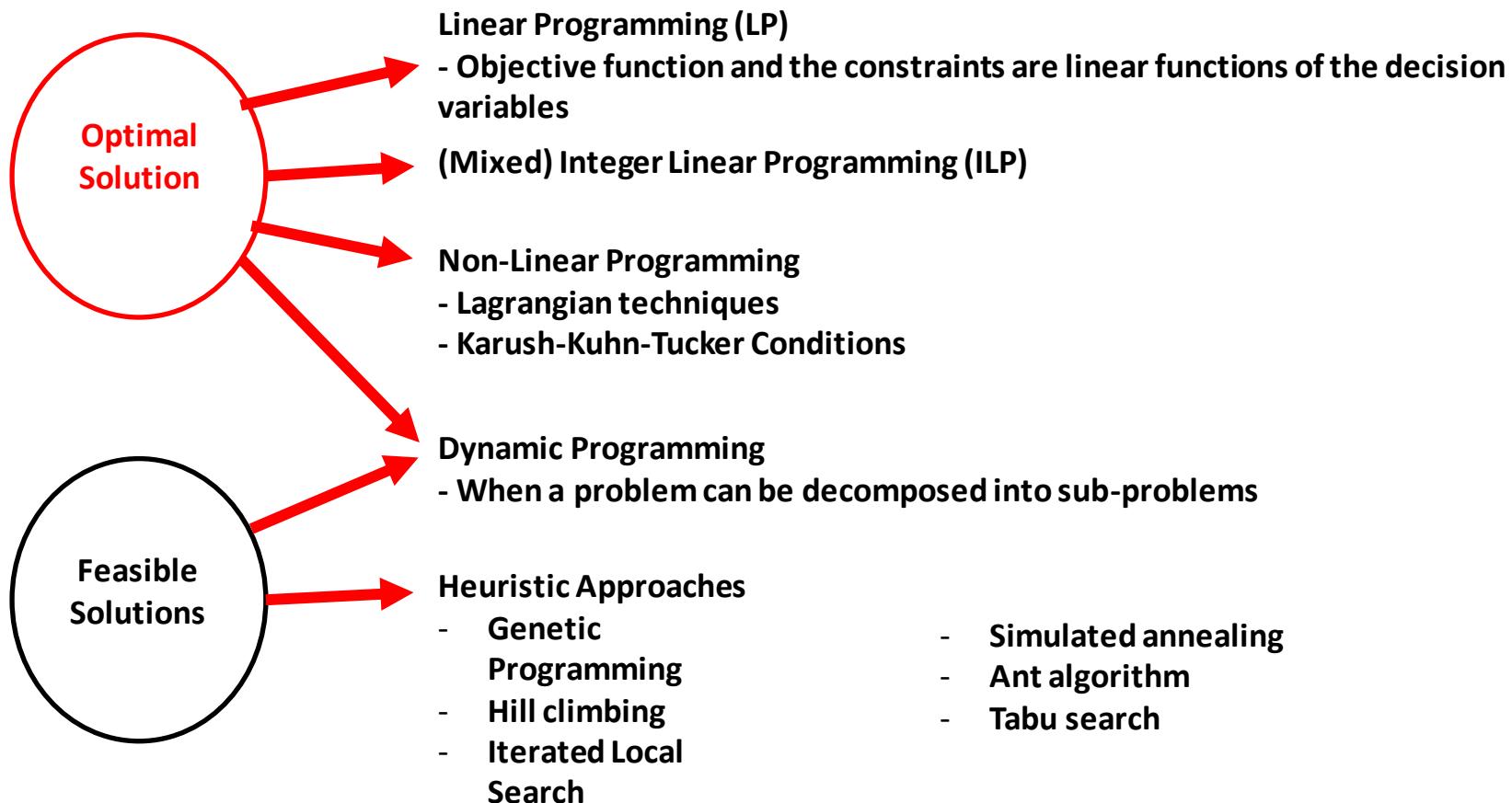
Multi-Objectives

Optimization Cont.

- Feasible Solutions:
Solutions that satisfy all constraints
- Optimal Solution:
The feasible solution with the largest objective value
(for maximization problems) OR
smallest objective value (for minimization problems)

Optimization algorithms – Another classification

Optimization Techniques – Another Classification



Linear Programming

- Decision Variables
- Objective Function (to maximize or minimize)
- Constraints (requirements or limitations)

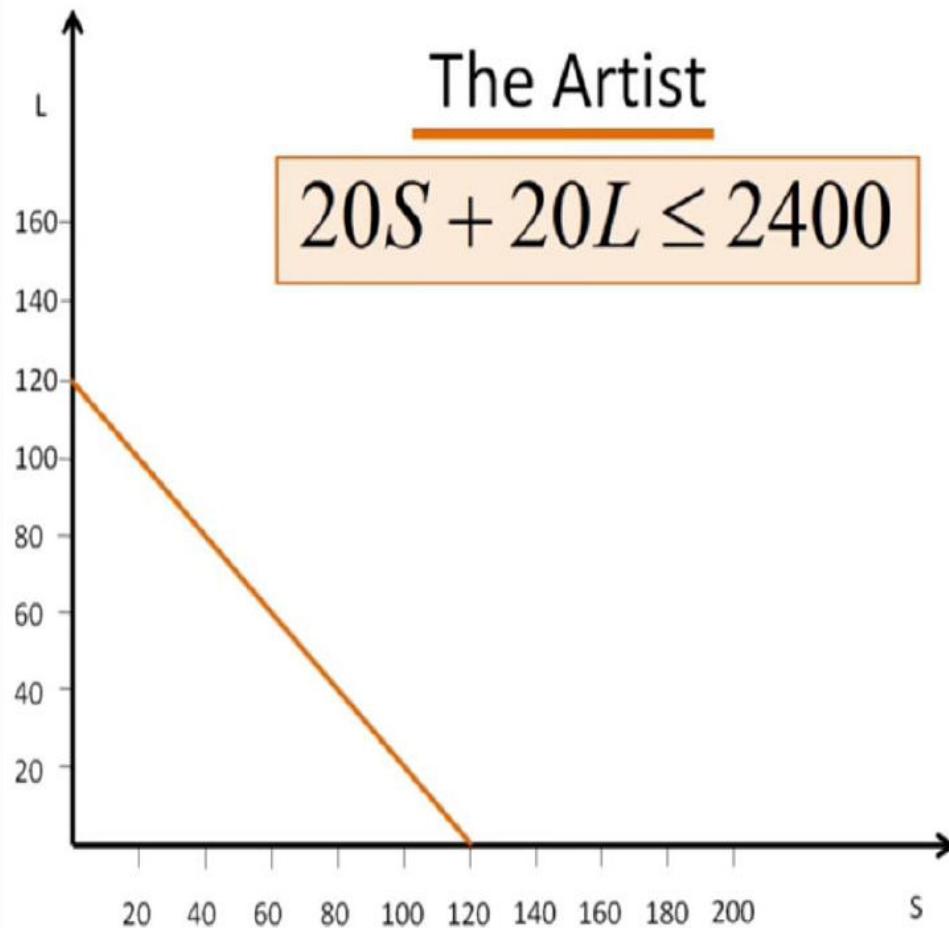
Linear Programming Solving Methods

- Graphical modeling
- Simplex method
- Optimization solver

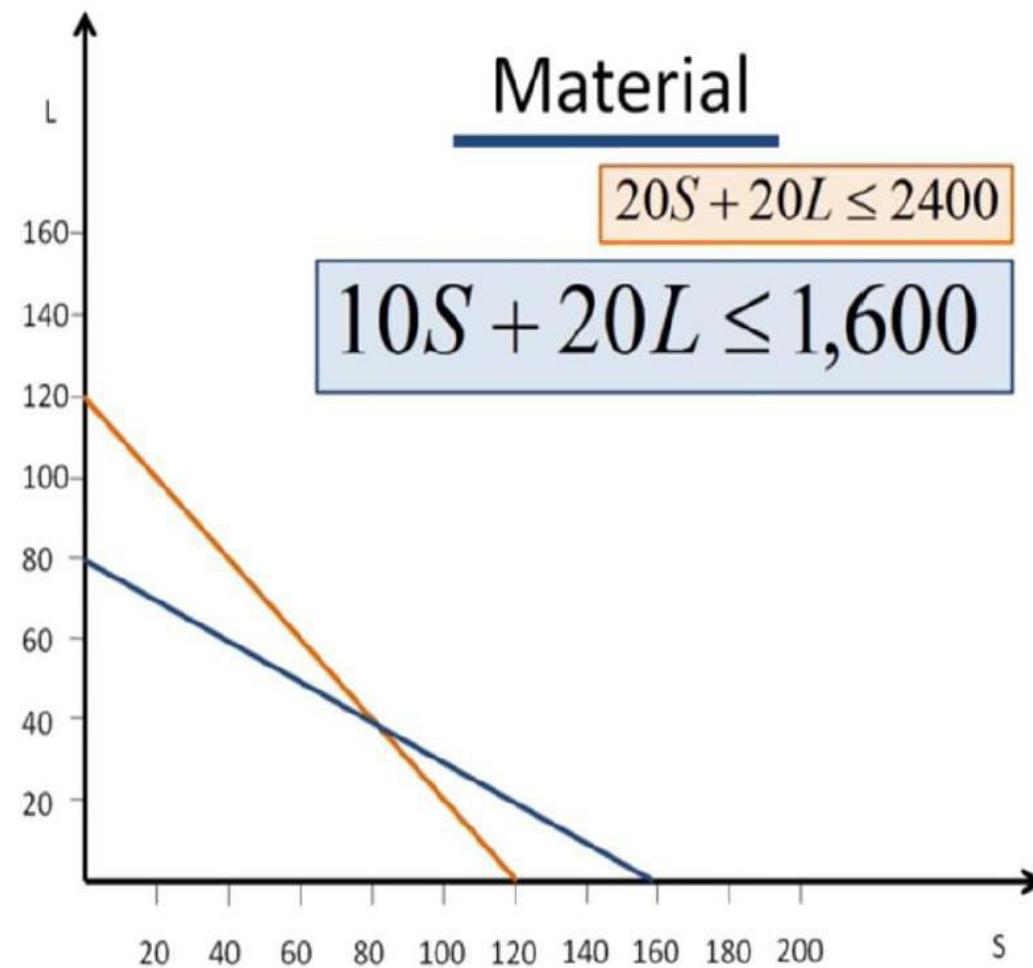
Linear Programming

- A small cake realizes earnings contribution of \$10 and large cake of \$12.
Small Cake: S
Large Cake: L
Object function: Maximize $10S + 12L$
- Each cake, irrespective of size, takes 20 minutes of baker's time, who works 40 hours per week.
 $20S + 20L \leq 2400$
- A small cake and large cake require 10oz and 20oz of flour, respectively. A total of 1600oz of flour is available.
 $10S + 20L \leq 1600$
- A small cake occupies 2 units of storage space, while a large cake occupies 3 units of storage space. Total available storage is 260 units.
 $2S + 3L \leq 260$

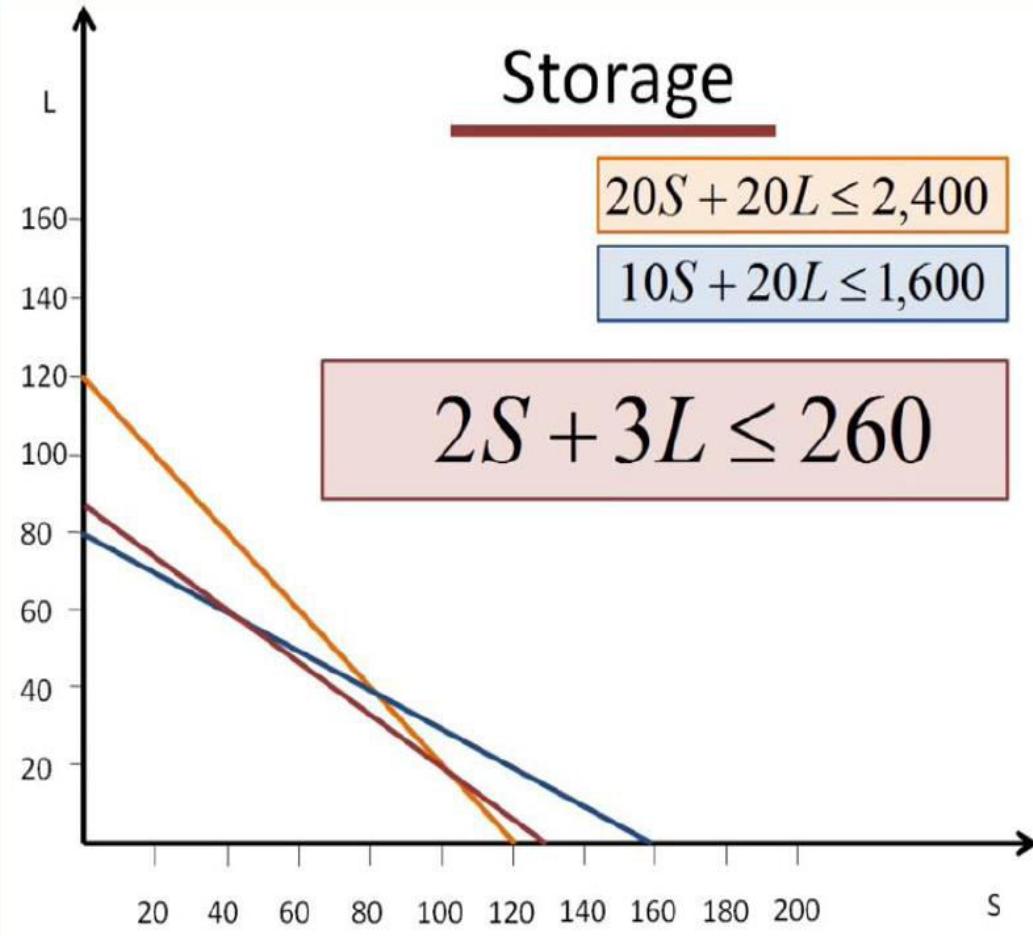
Linear Programming Graphical Method



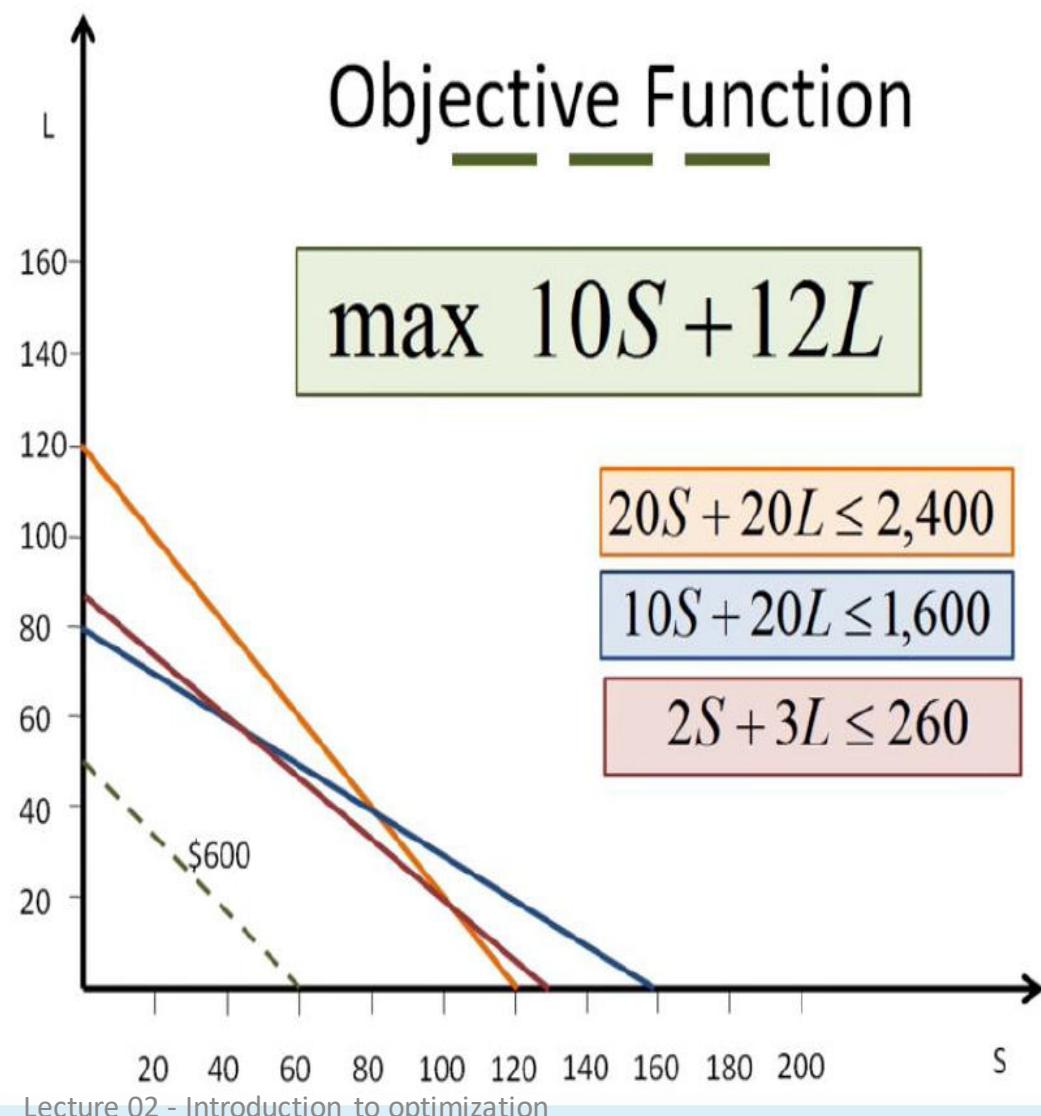
Linear Programming Graphical Method



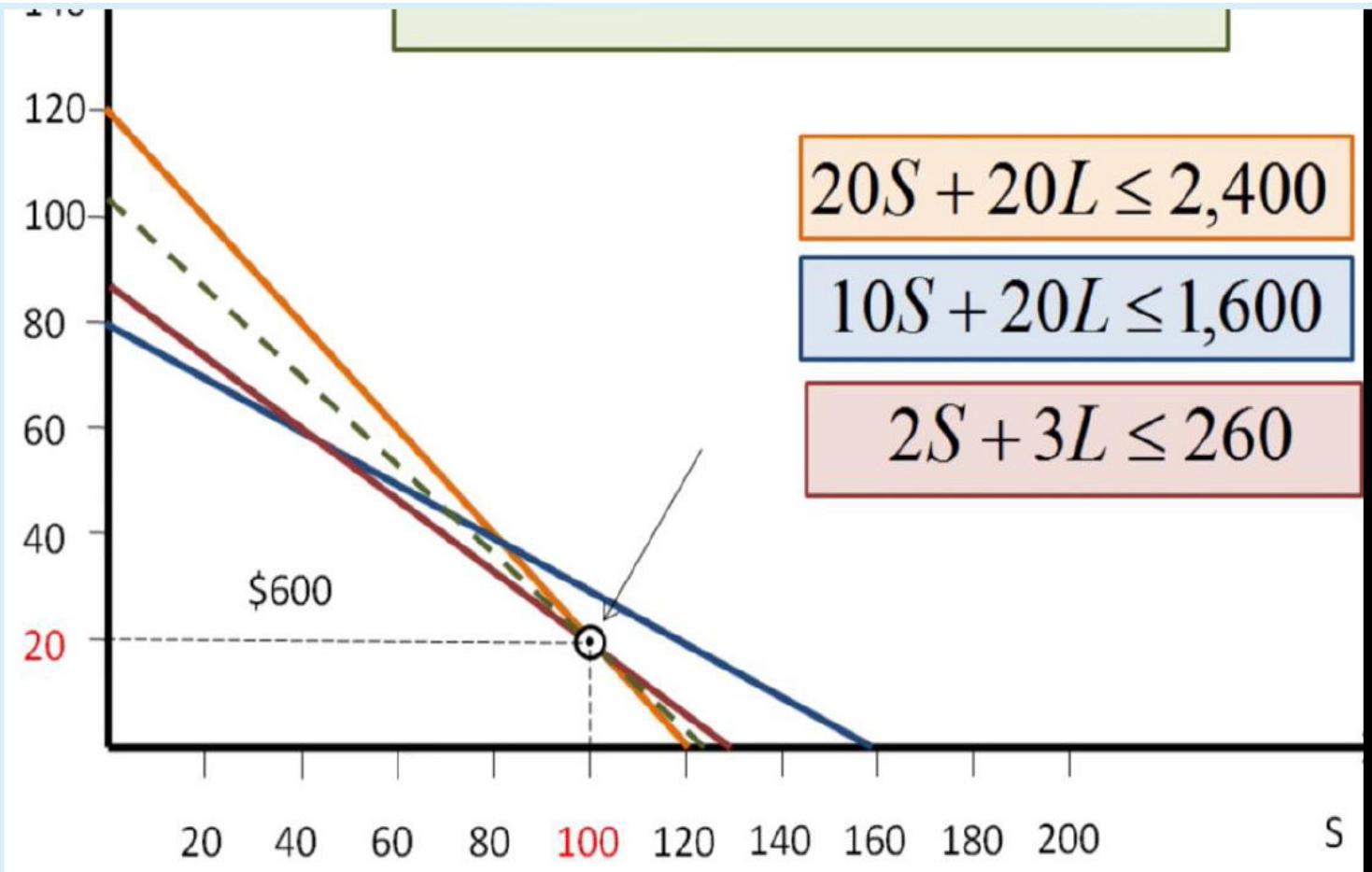
Linear Programming Graphical Method



Linear Programming Graphical Method



Linear Programming Graphical Method



Linear Programming - Simplex Method

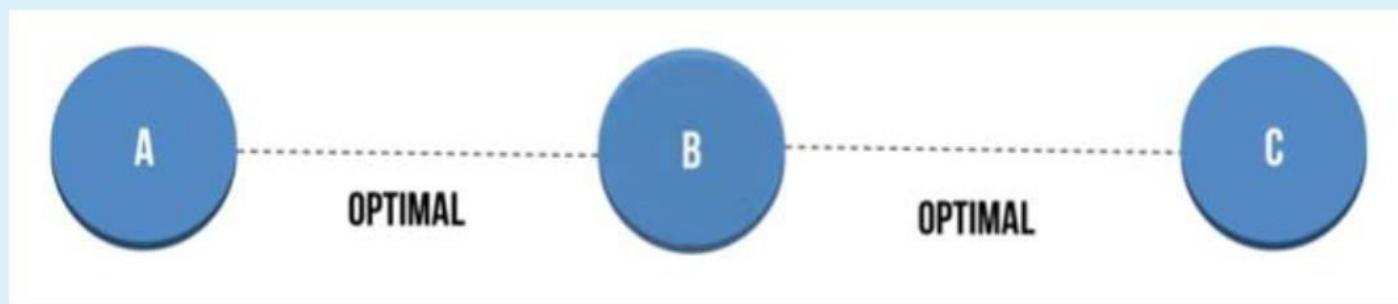
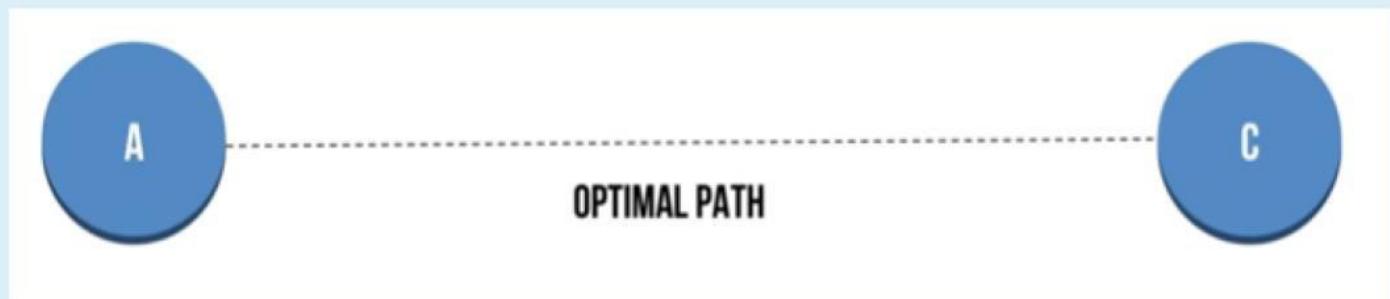
Linear Programming – Solver demo

Integer Linear Programming (ILP)

- Used when the variables have to be integers
- MILP (Mixed Integer Linear Programming)

Dynamic Programming

Dynamic programming

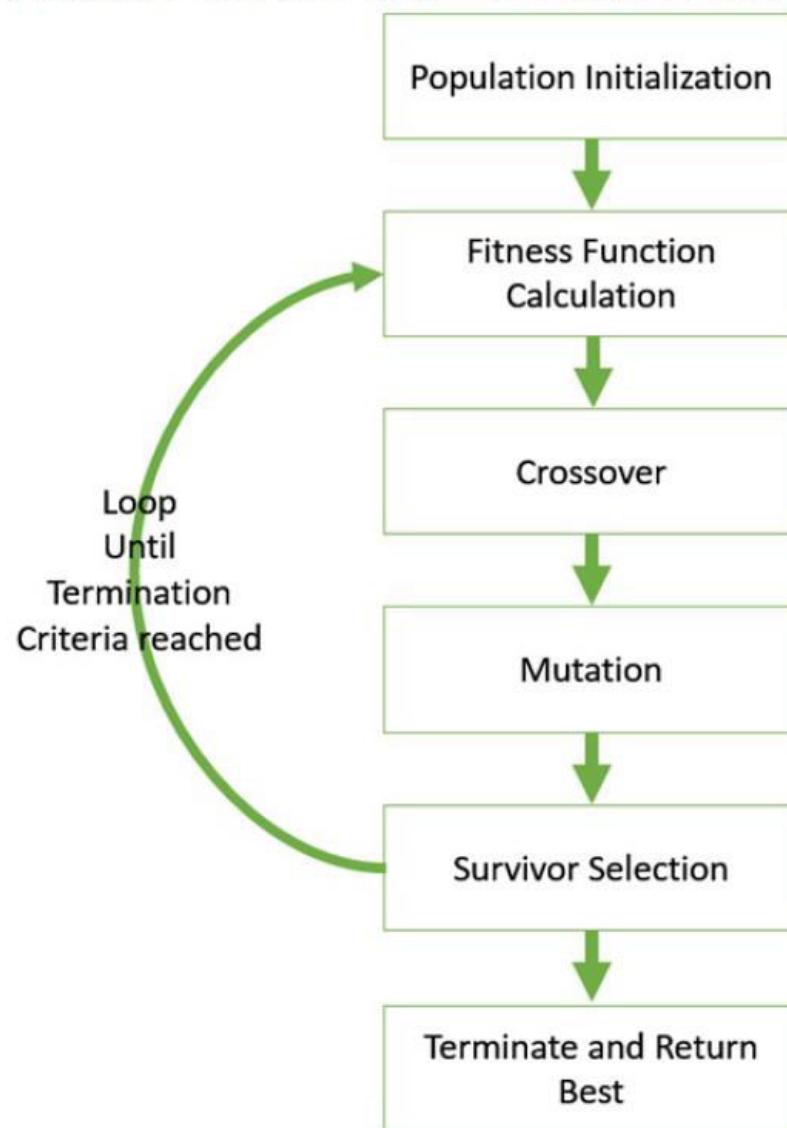


GENETIC ALGORITHM

INTRODUCTION

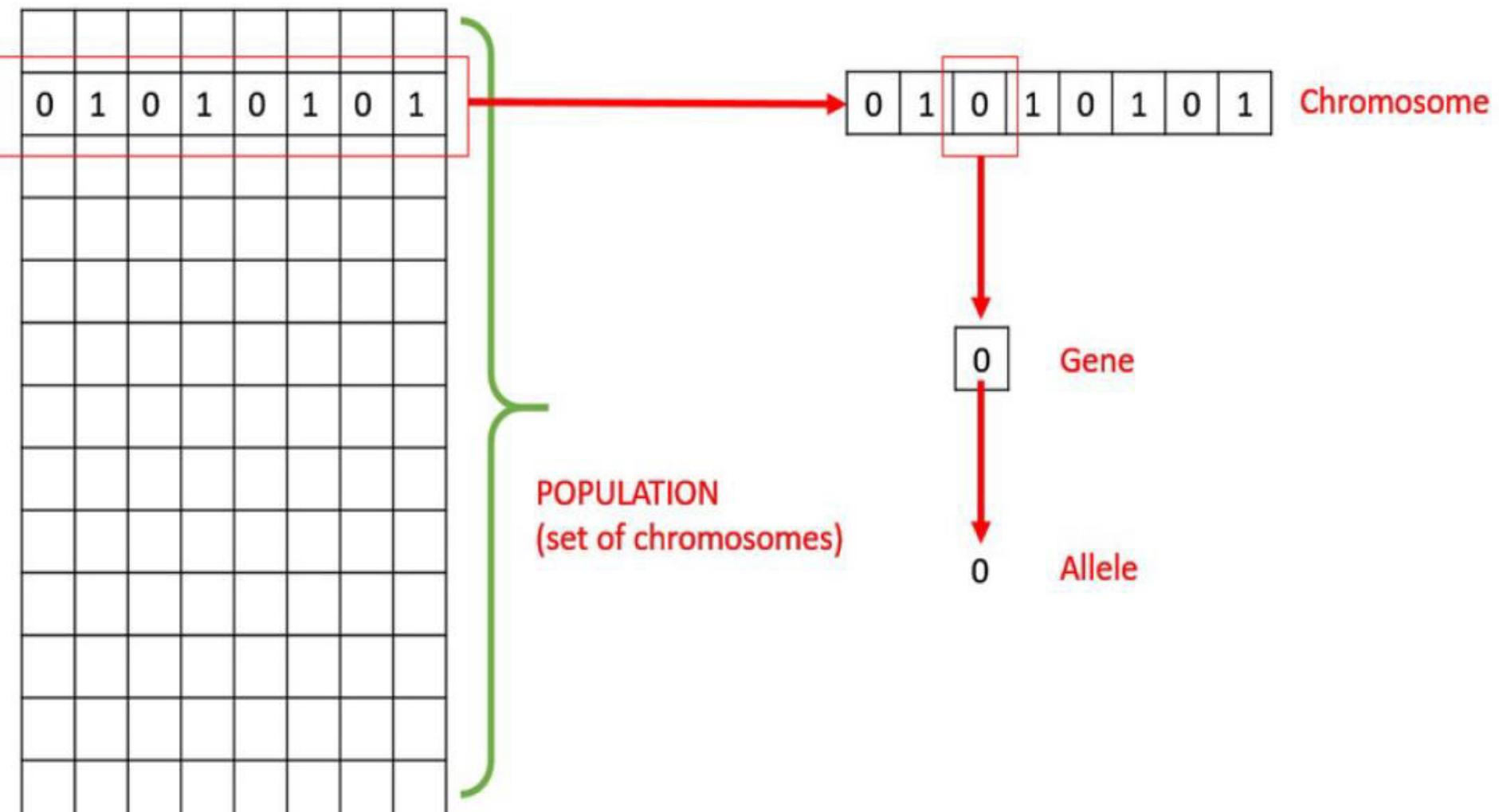
- Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.

BASIC STRUCTURE OF GENETIC ALGORITHM



BASIC TERMINOLOGY

- **Population** — It is a subset of all the possible (encoded) solutions to the given problem.
- **Chromosomes** — A chromosome is one such solution to the given problem.
- **Gene** — A gene is one element position of a chromosome.
- **Allele** — It is the value a gene takes for a particular chromosome.



- Fitness Function — A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- Genetic Operators — These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

GENOTYPE REPRESENTATION

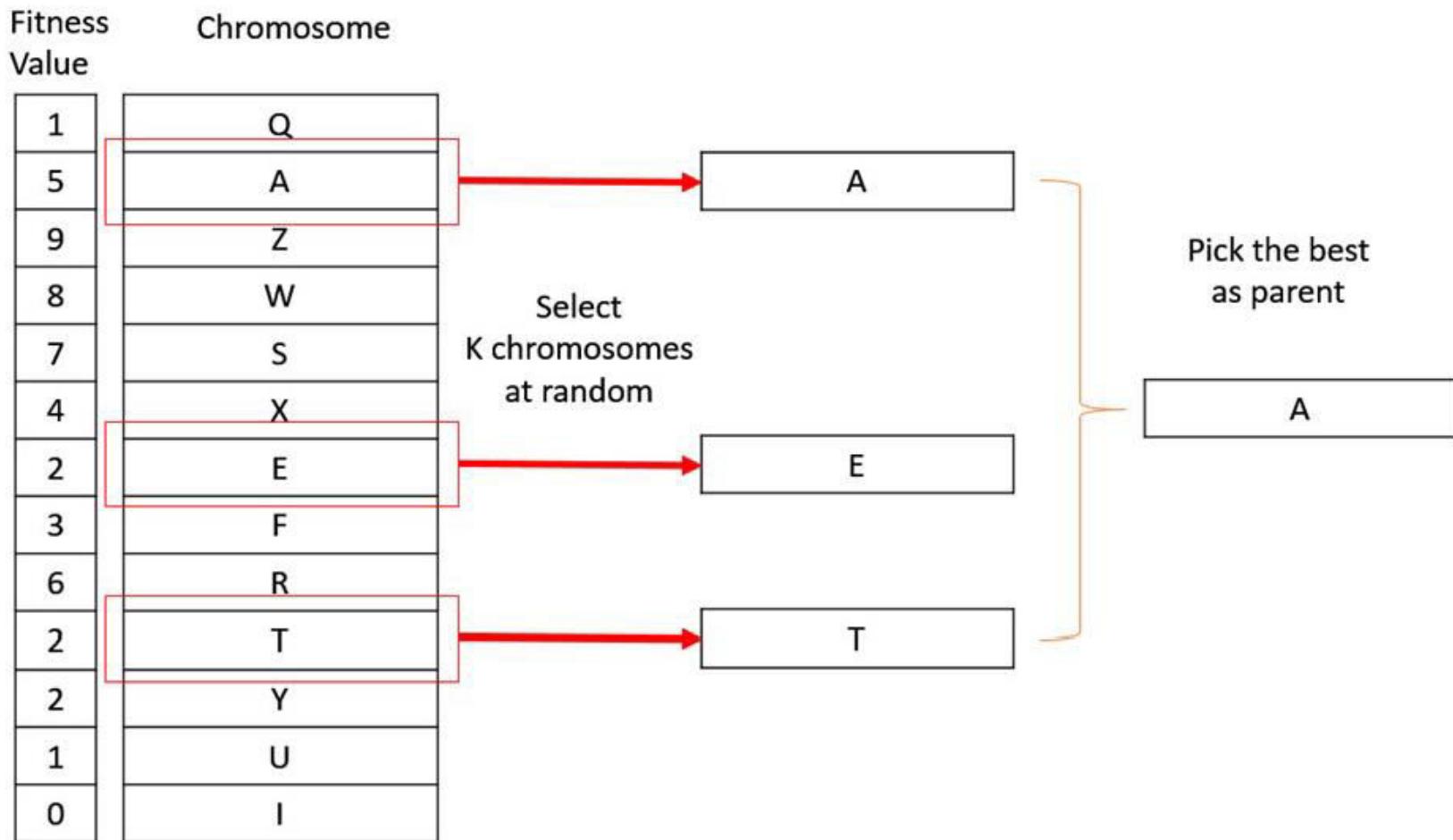
- BINARY REPRESENTATION

0	0	1	0	1	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---

- REAL VALUED REPRESENTATION

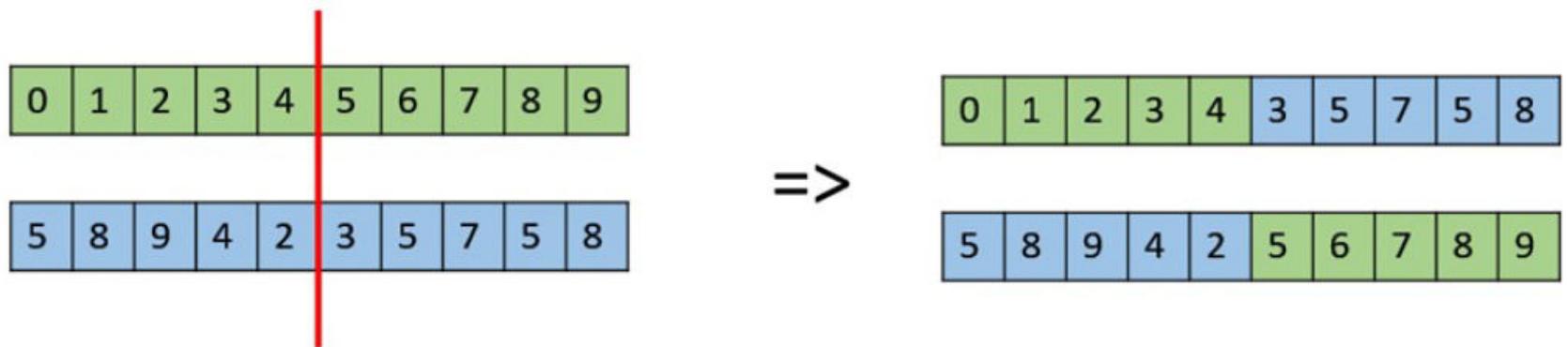
0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

- Tournament Selection



GA - CROSSOVER

- one parent is selected and one or more off-springs are produced using the genetic material of the parents
- **One Point Crossover**



GA - MUTATION

- used to maintain and introduce diversity in the genetic population
- Mutation Operators:

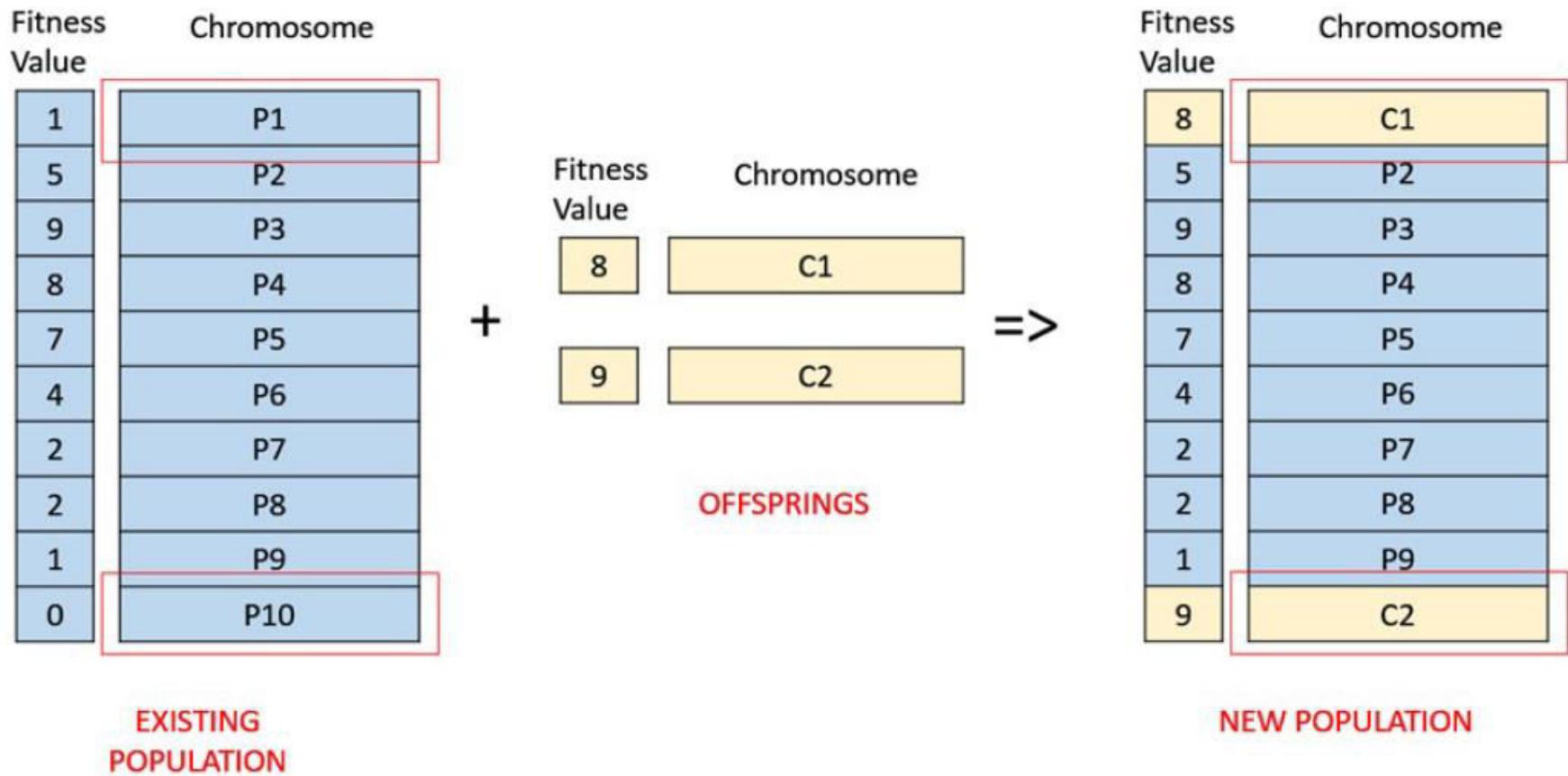
- **Bit Flip Mutation**

0 0 1 1 0 1 0 0 1 0	=>	0 0 1 0 0 1 0 0 1 0
---------------------------------------	----	---------------------------------------

- **Random Resetting**
 - **Swap Mutation**

1 2 3 4 5 6 7 8 9 0	=>	1 6 3 4 5 2 7 8 9 0
---------------------------------------	----	---------------------------------------

Fitness Based Selection



GA - demo

GA – sample code

Multi-objective optimization

- Most real-world problems are multi-objective optimization problems
- Have to find a trade-off between multiple objectives
- One method is to derive a single objective by having a weighted sum of the multiple objectives

Summary

- Optimization is the process of finding the optimal set of inputs to maximize an objective function, with or without constraints
- Gradient based algorithms
 - Gradient descent, Stochastic gradient descent.....
- Gradient free algorithms
 - Genetic algorithms, simulated annealing....

Summary

- Optimal solution - Linear programming, ILP
- Feasible solutions (Heuristic based) – Genetic algorithms, Simulated annealing
- Machine learning is an application of optimization

Lecture 2 – Linear Regression

Regression

- Supervised learning technique
- Used when the value that you want to predict is a **continuous** variable
 - e.g. - Predicting the **height** of a person based on the nationality, age, gender....
 - Predicting the housing **price** based on the floor area, no. of floors, city.....
 - Predicting the **value** of a share based on the company revenue, profit,.....

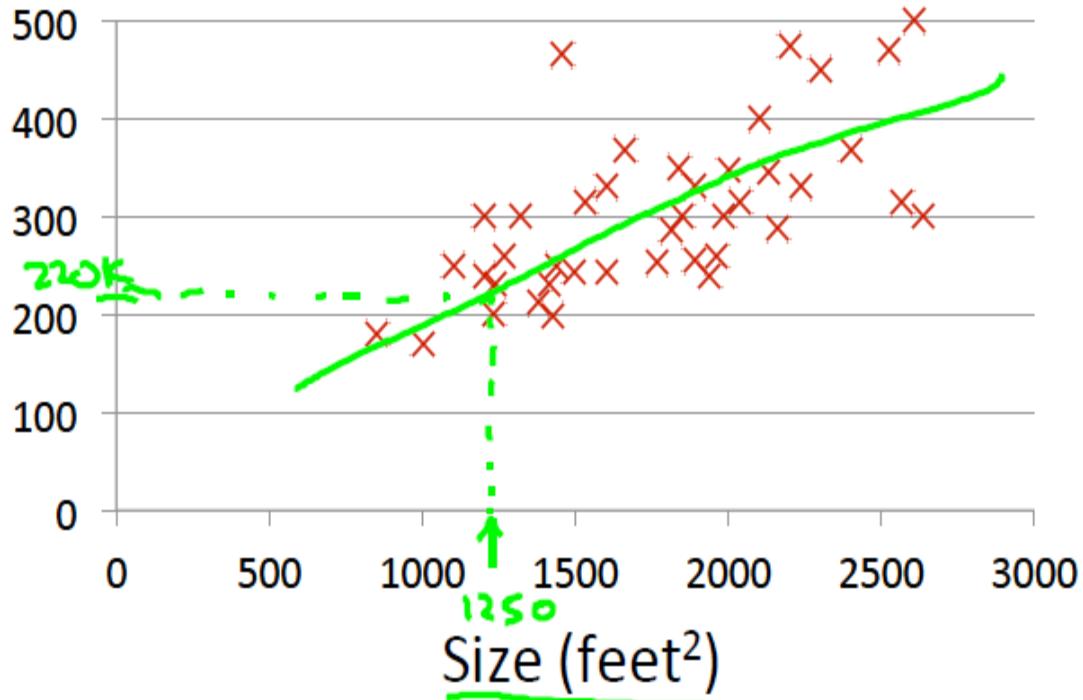
Outline

- Why regression?
- Linear regression with one variable
- Linear regression with multiple variables
- Polynomial regression
- Dealing with common issues
- Normal equation

Linear regression with single variable

Housing Prices (Portland, OR)

Price
(in 1000s
of dollars)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

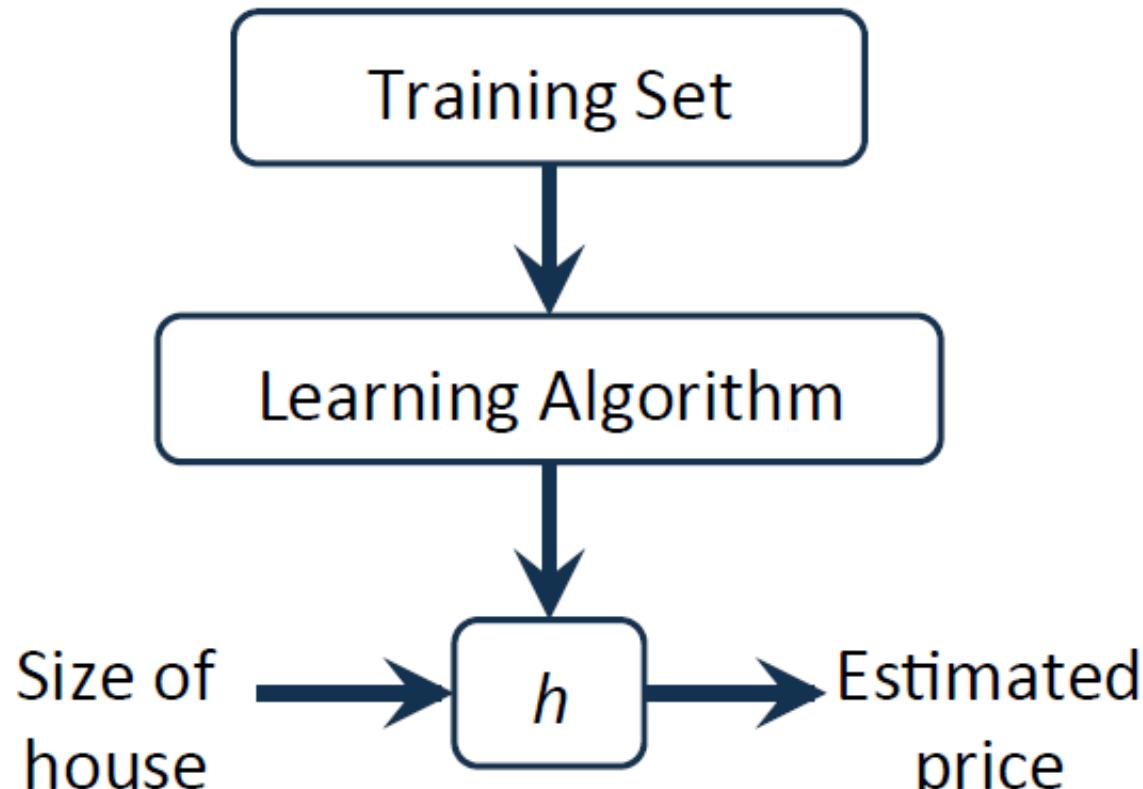
Classification: Discrete-valued output

Training Data Example

Living area (feet ²)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
:	:

- TO DO: Write the notation.....

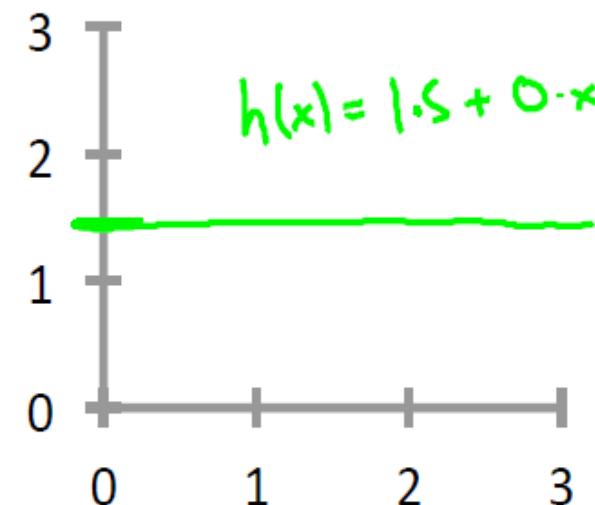
Learning process



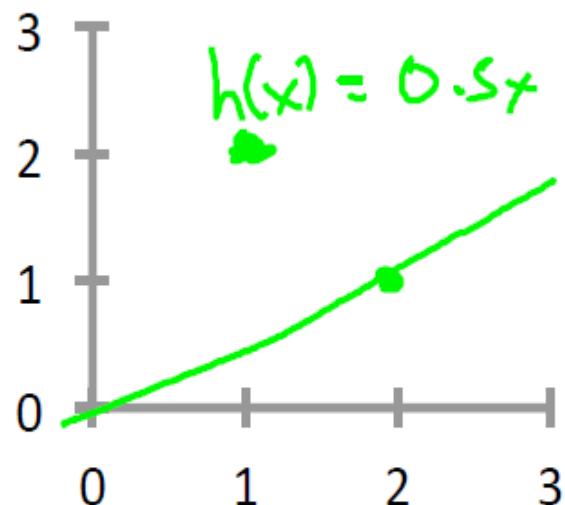
- TO DO – Write the Hypothesis

Hypothesis

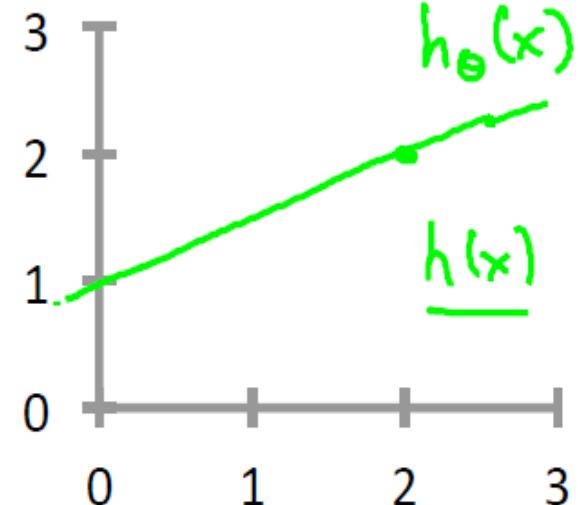
$$\underline{h_{\theta}(x)} = \theta_0 + \theta_1 x$$



$$\begin{aligned} \rightarrow \theta_0 &= 1.5 \\ \rightarrow \theta_1 &= 0 \end{aligned}$$



$$\begin{aligned} \rightarrow \theta_0 &= 0 \\ \rightarrow \theta_1 &= 0.5 \end{aligned}$$



$$\begin{aligned} \rightarrow \theta_0 &= 1 \\ \rightarrow \theta_1 &= 0.5 \end{aligned}$$

Cost Function

- TO DO

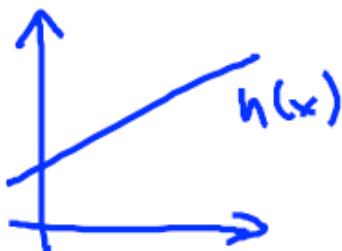
Simplified

Hypothesis:

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

Parameters:

$$\underline{\theta_0, \theta_1}$$



Cost Function:

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

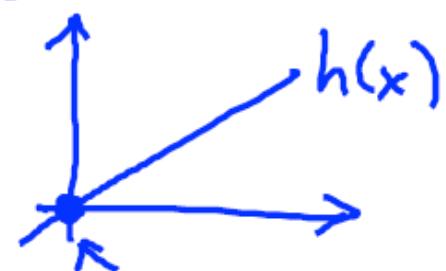
Goal: minimize $J(\theta_0, \theta_1)$

$$\nearrow \theta_0, \theta_1$$

$$h_{\theta}(x) = \underline{\theta_1 x}$$

$$\theta_0 = 0$$

$$\underline{\theta_1}$$



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

$$\min_{\theta_1} \underline{J(\theta_1)}$$

$$\underline{\theta_1, x^{(i)}}$$

Hypothesis vs. Cost function

- TO DO

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

Parameters: θ_0, θ_1

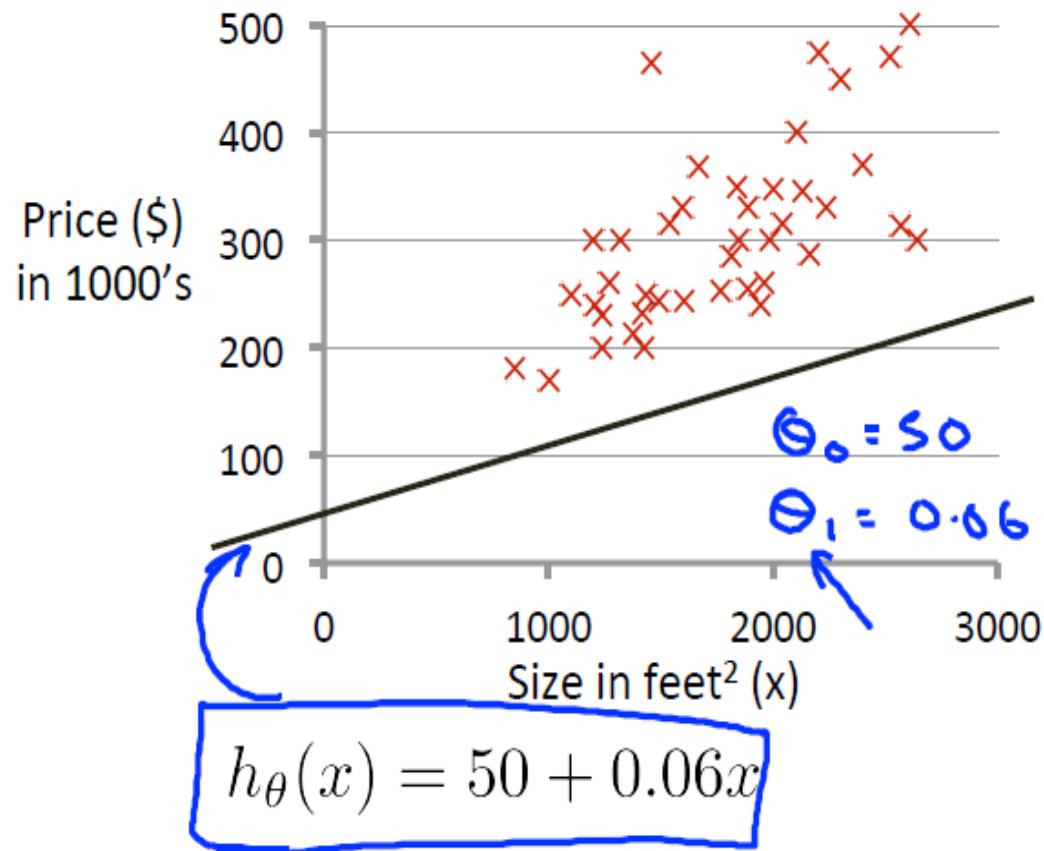
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

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

.

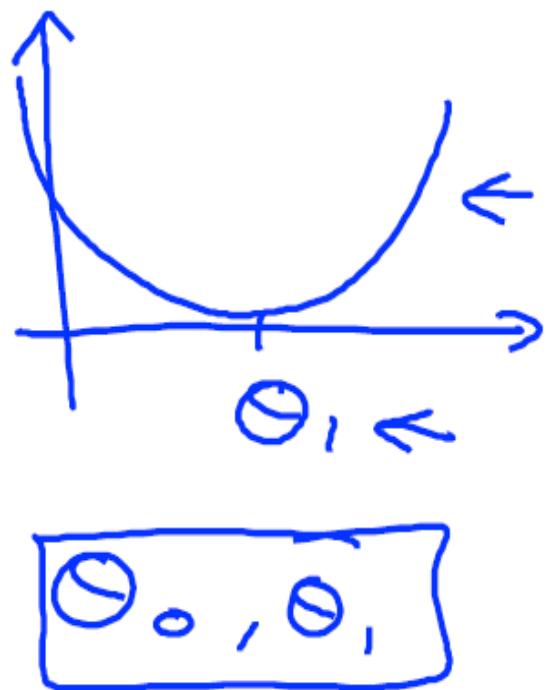
$$\underline{h_{\theta}(x)}$$

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

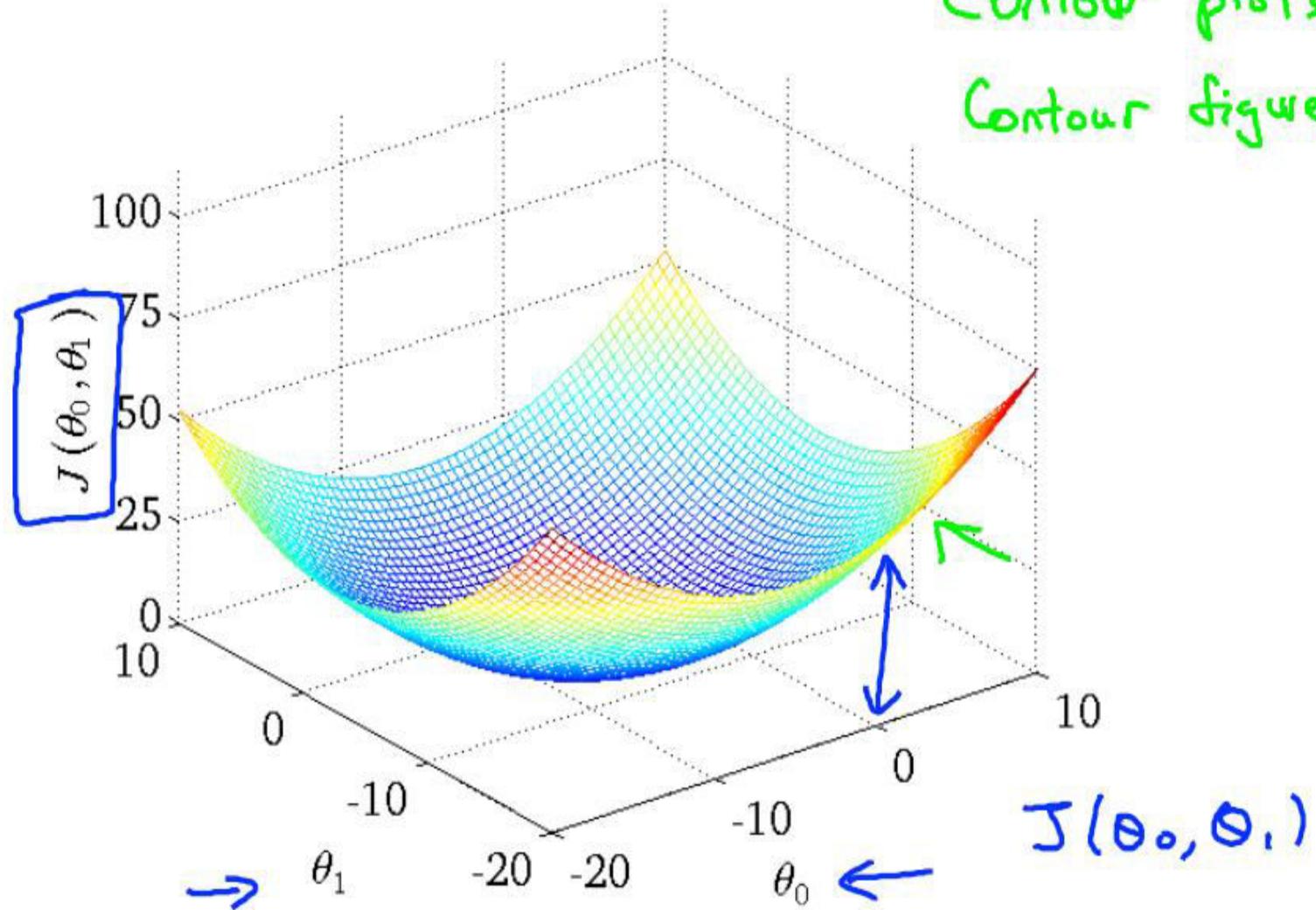


$$J(\underline{\theta_0, \theta_1})$$

(function of the parameters θ_0, θ_1)

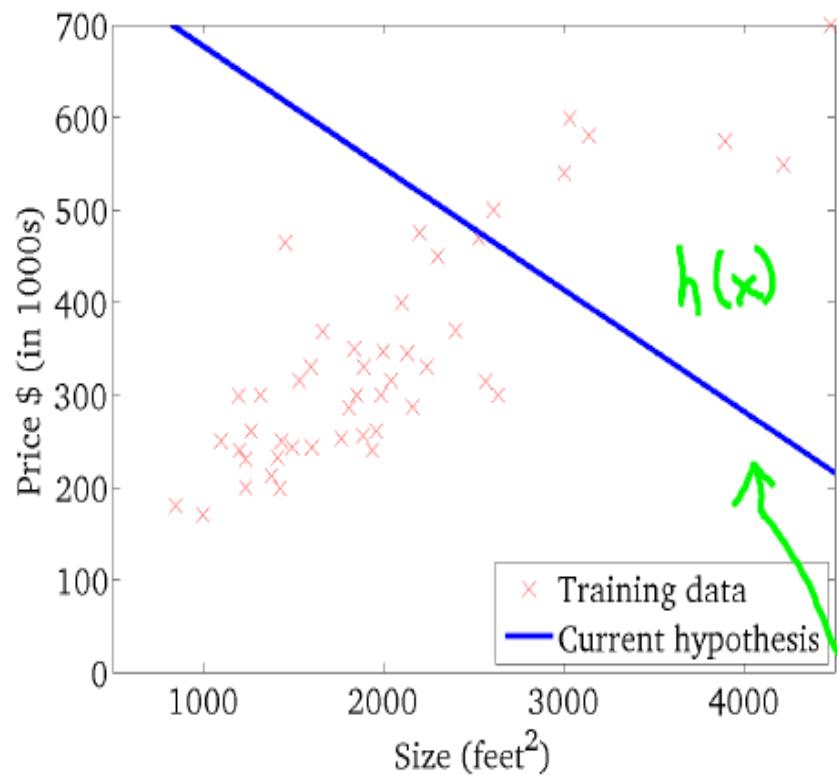


Contour plots
Contour figures -



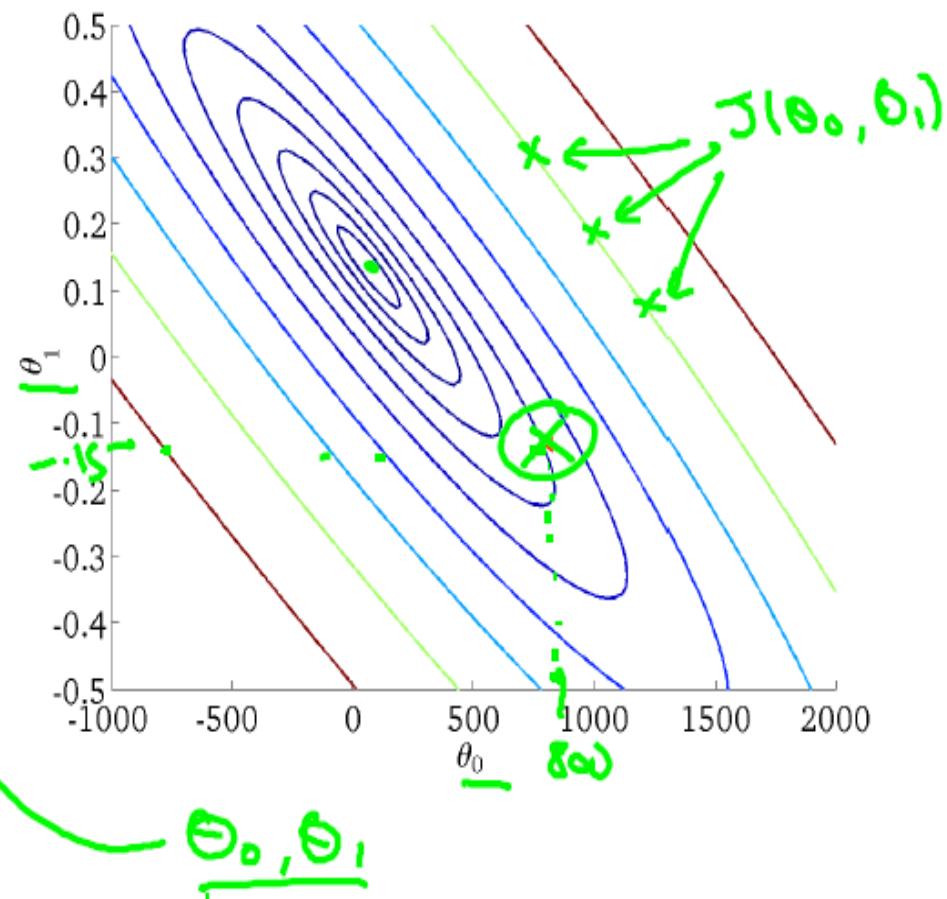
$$h_{\theta}(x)$$

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



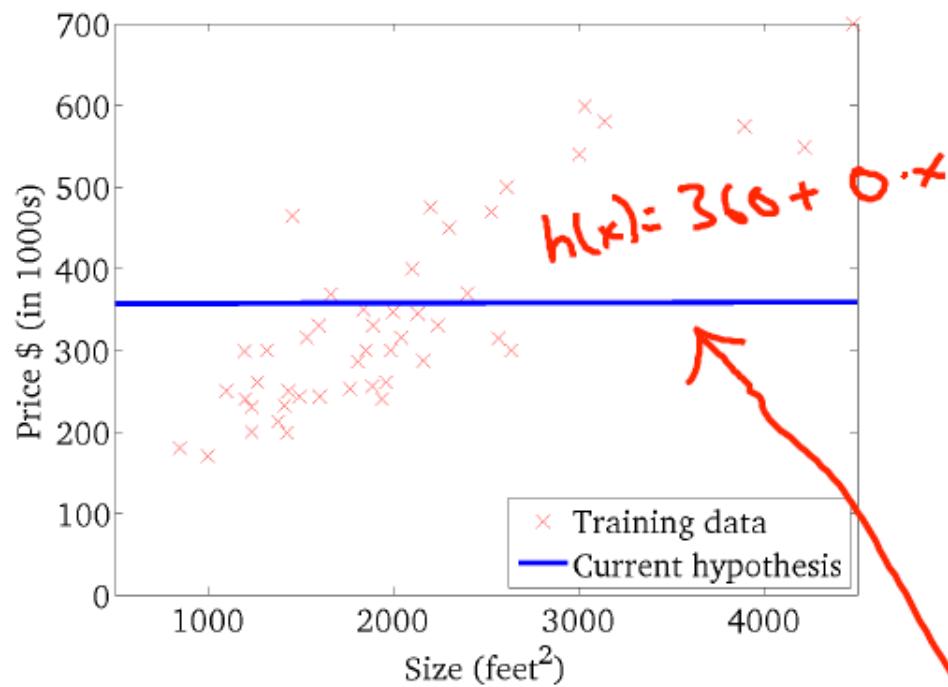
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



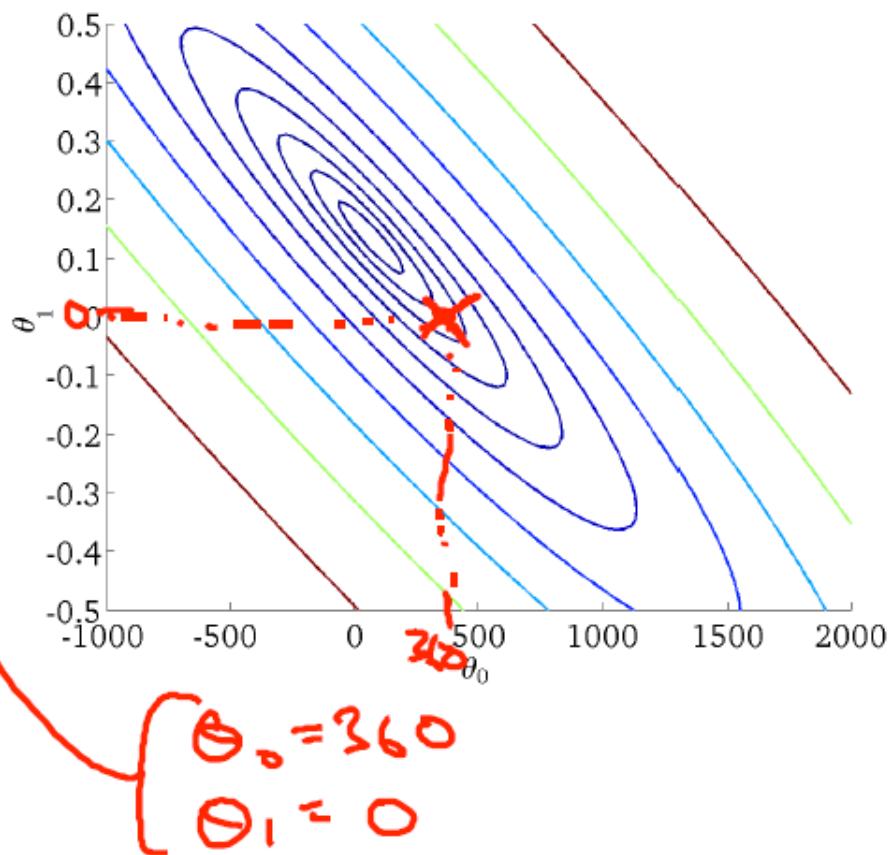
$$h_{\theta}(x)$$

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



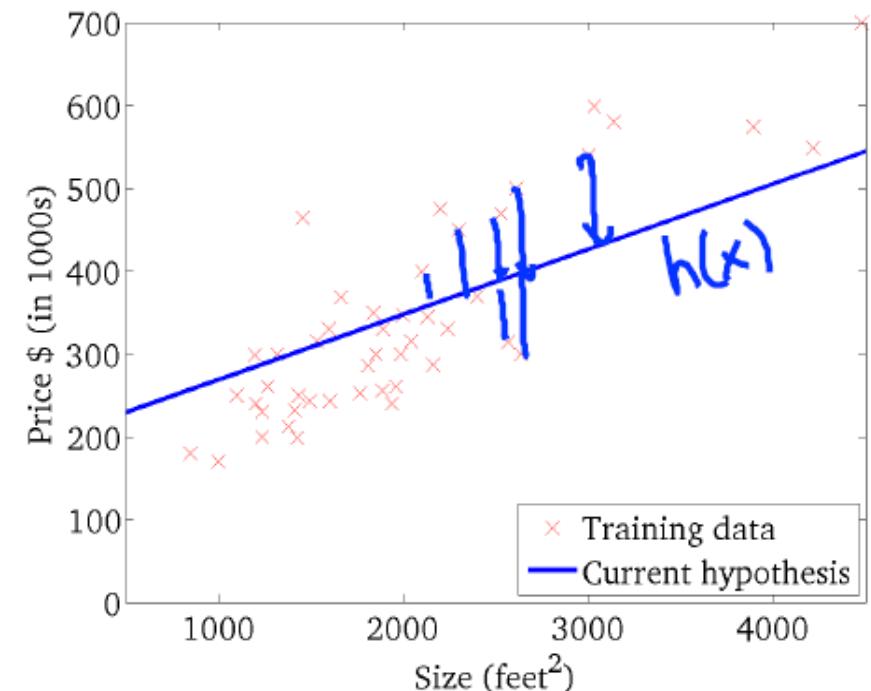
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



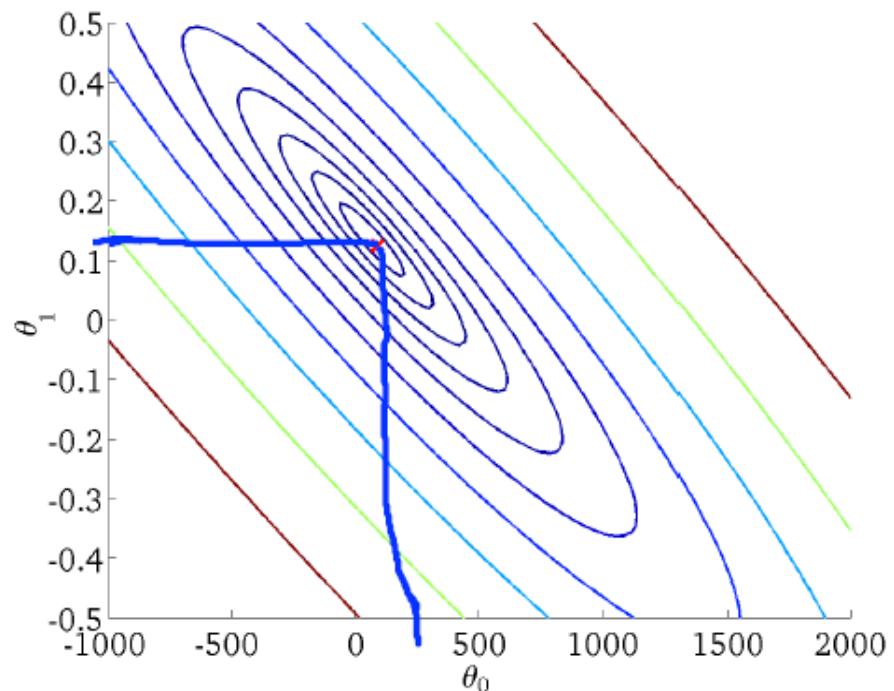
$$h_{\theta}(x)$$

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



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

(function of the parameters θ_0, θ_1)



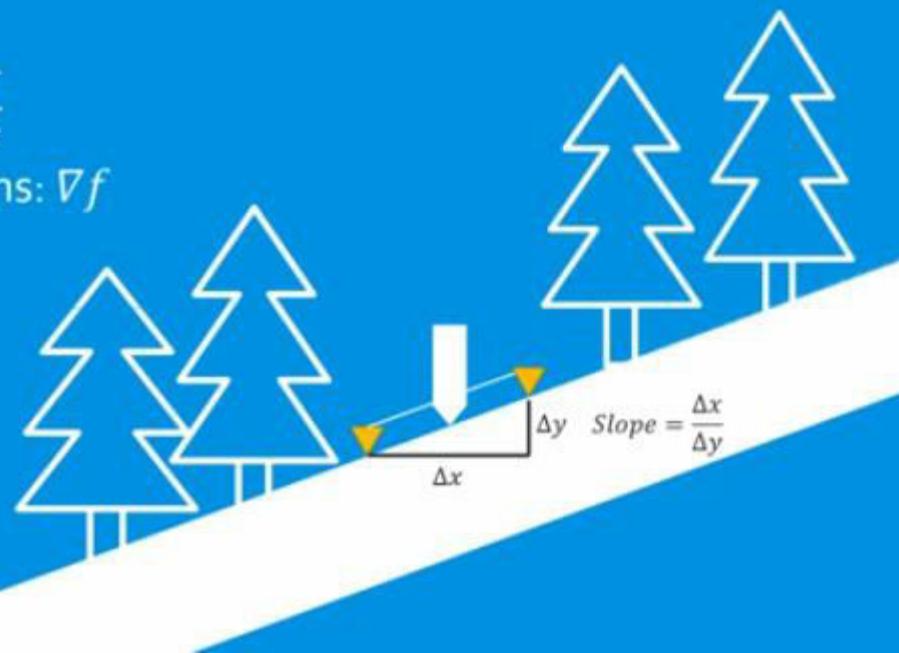




Search Direction

One Dimension: $\frac{df}{dx}$

Multiple Dimensions: ∇f



Search Direction



Step Size





3 Steps

Search Direction

Step Size

Convergence Check

Gradient Descent

Have some function $\underline{J(\theta_0, \theta_1)}$ $\mathcal{J}(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$

Want $\min_{\theta_0, \theta_1} \underline{J(\theta_0, \theta_1)}$ $\min_{\theta_0, \dots, \theta_n} \underline{\mathcal{J}(\theta_0, \dots, \theta_n)}$

Outline:

- Start with some $\underline{\theta_0, \theta_1}$ (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing $\underline{\theta_0, \theta_1}$ to reduce $\underline{J(\theta_0, \theta_1)}$
until we hopefully end up at a minimum

Gradient descent algorithm

repeat until convergence {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

learning rate derivative

} (simultaneously update
 $j = 0$ and $j = 1$)

$$\min_{\theta_1} J(\theta_1) \quad \theta_1 \in \mathbb{R}.$$

- TO DO: Positive and Negative gradients

Gradient descent algorithm

```
repeat until convergence {  
     $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$   
    (for  $j = 1$  and  $j = 0$ )  
}
```

Linear Regression Model

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

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

- TO DO: Derivation

Gradient Descent algorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \right]$$

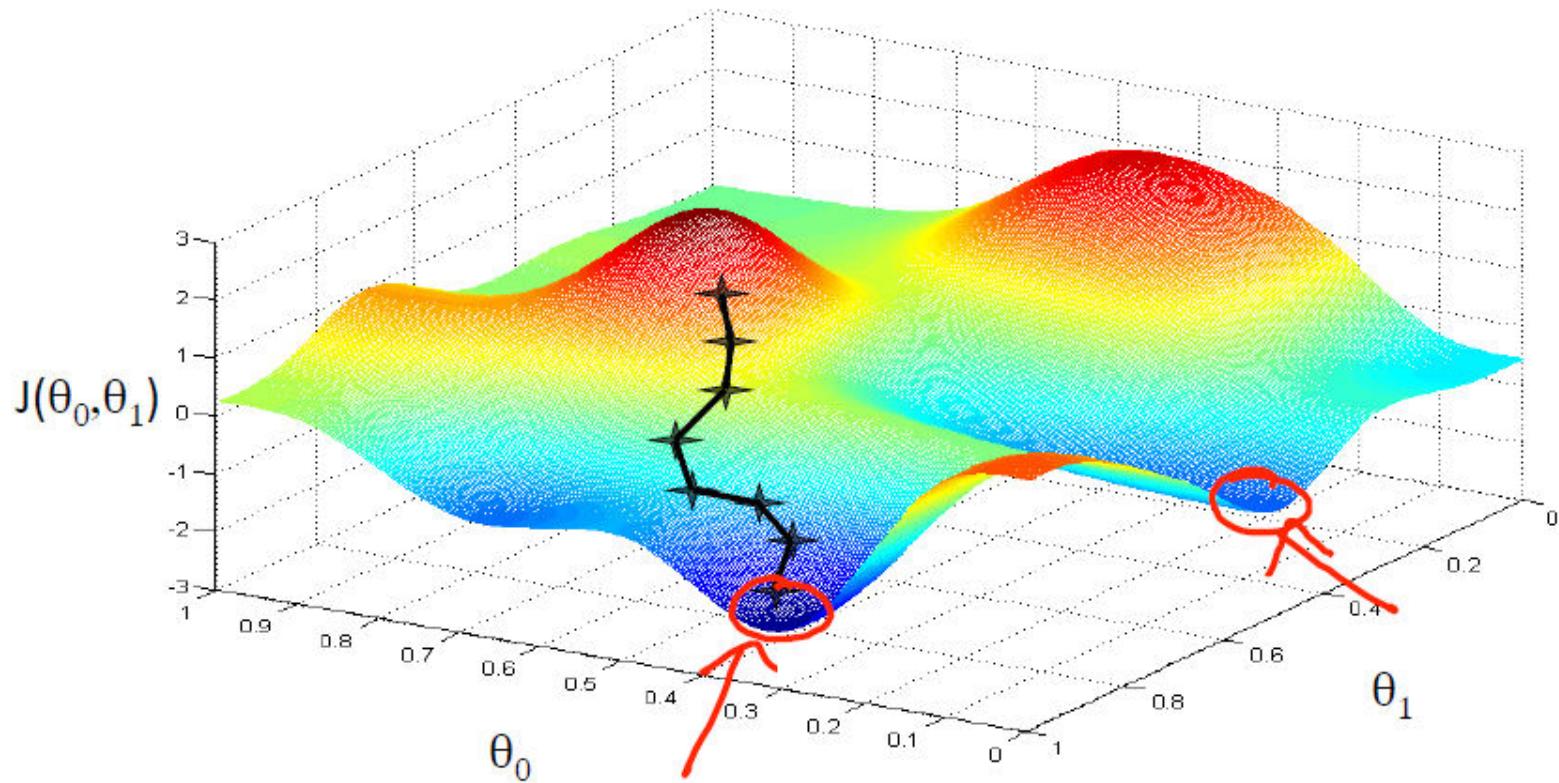
$$\theta_1 := \theta_1 - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right]$$

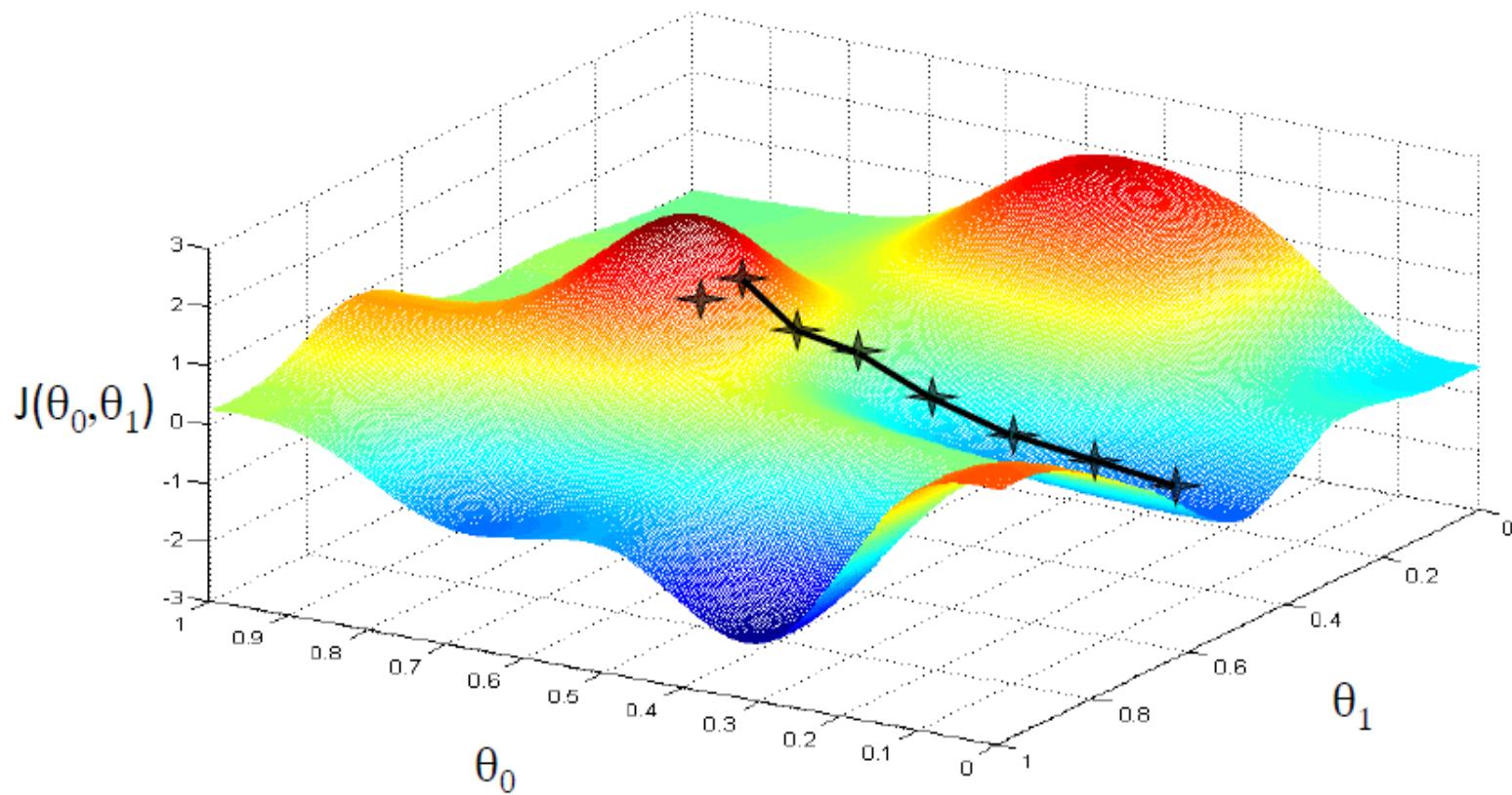
}

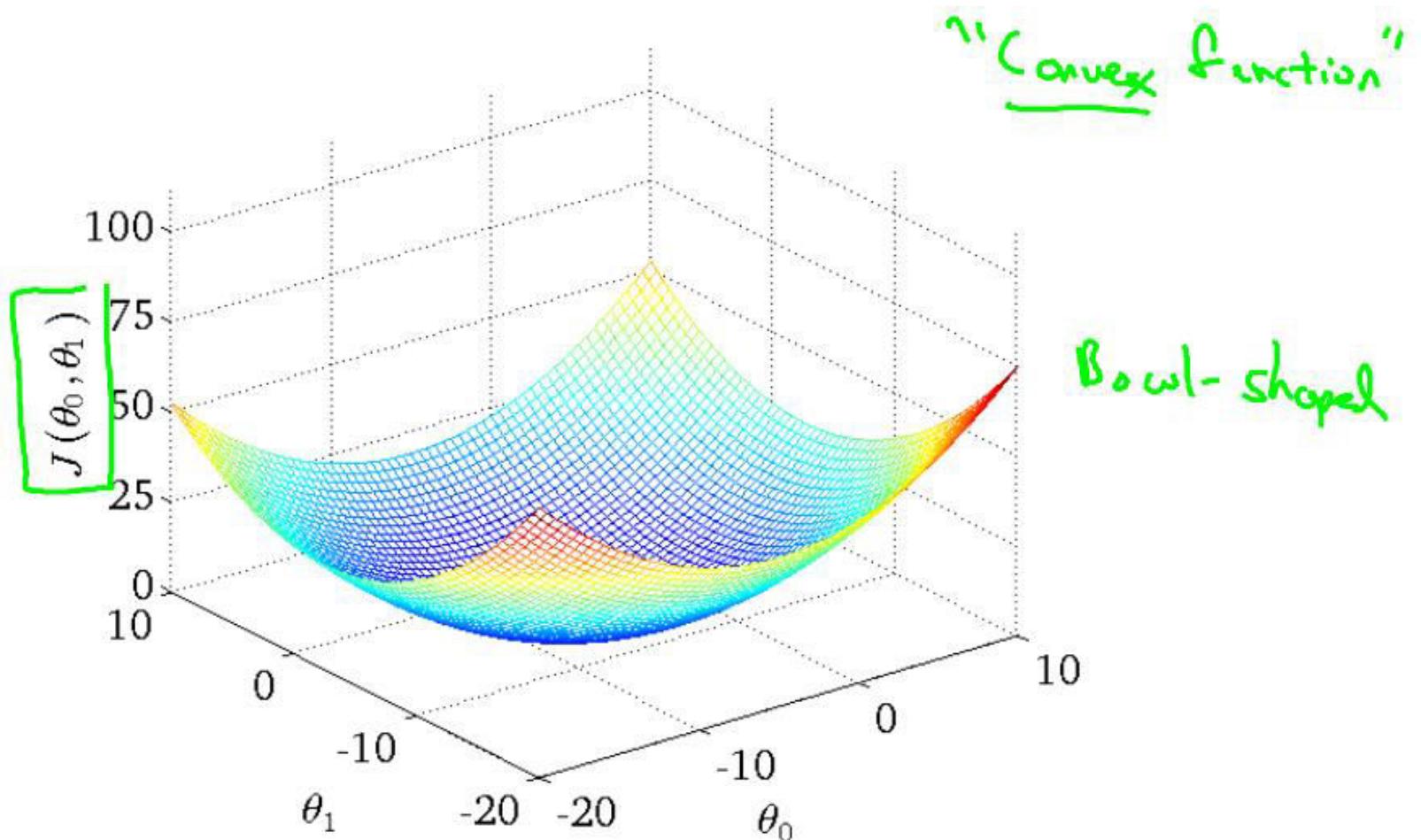
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

update
 θ_0 and θ_1
simultaneously

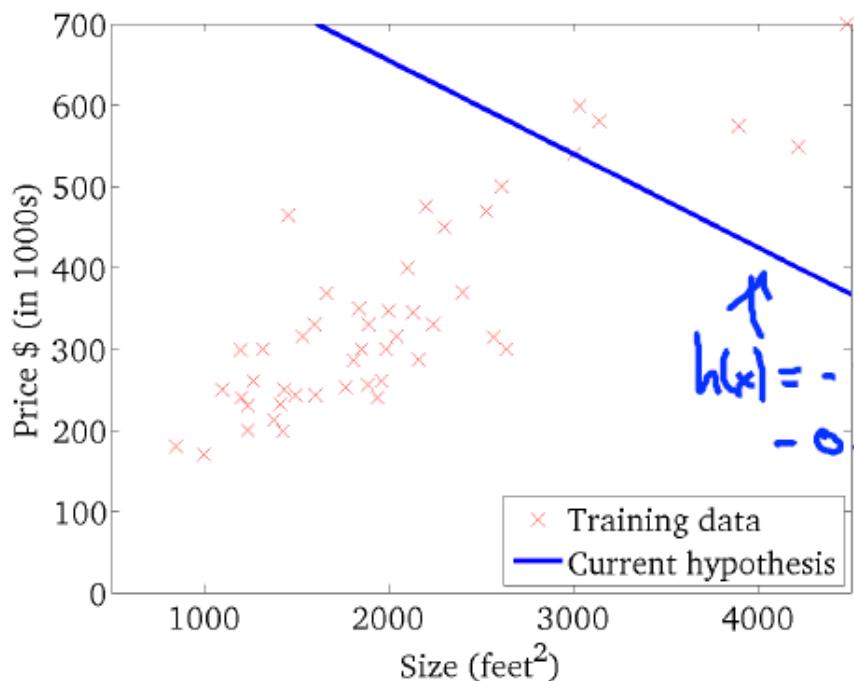
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$



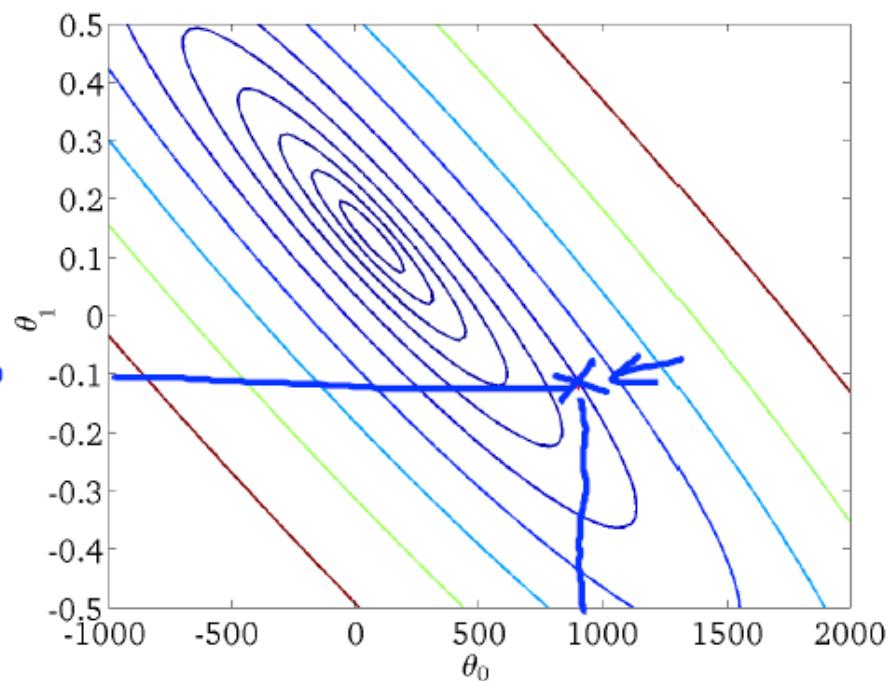




$\underline{h_\theta(x)}$
 (for fixed θ_0, θ_1 , this is a function of x)

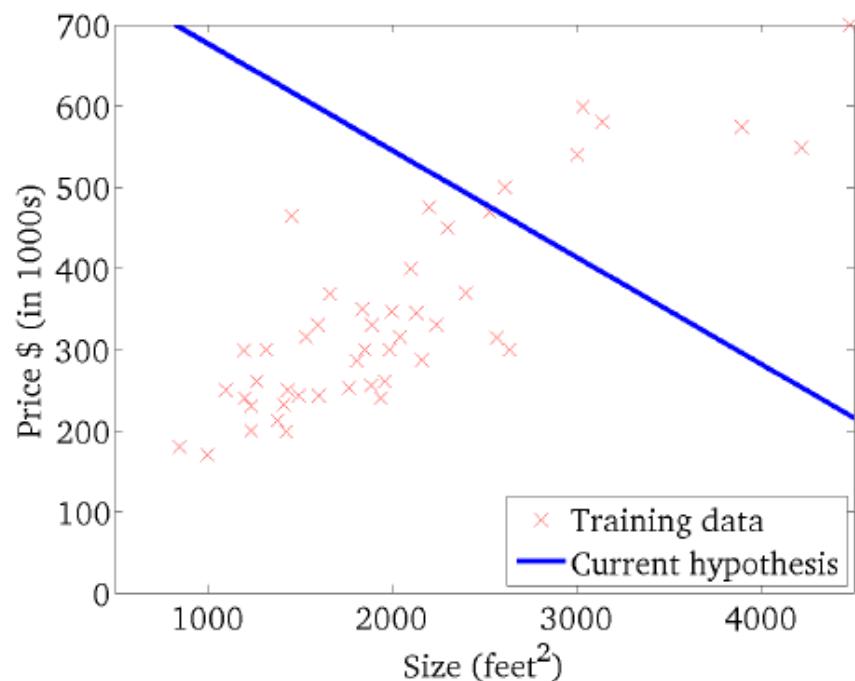


$\underline{J(\theta_0, \theta_1)}$
 (function of the parameters θ_0, θ_1)



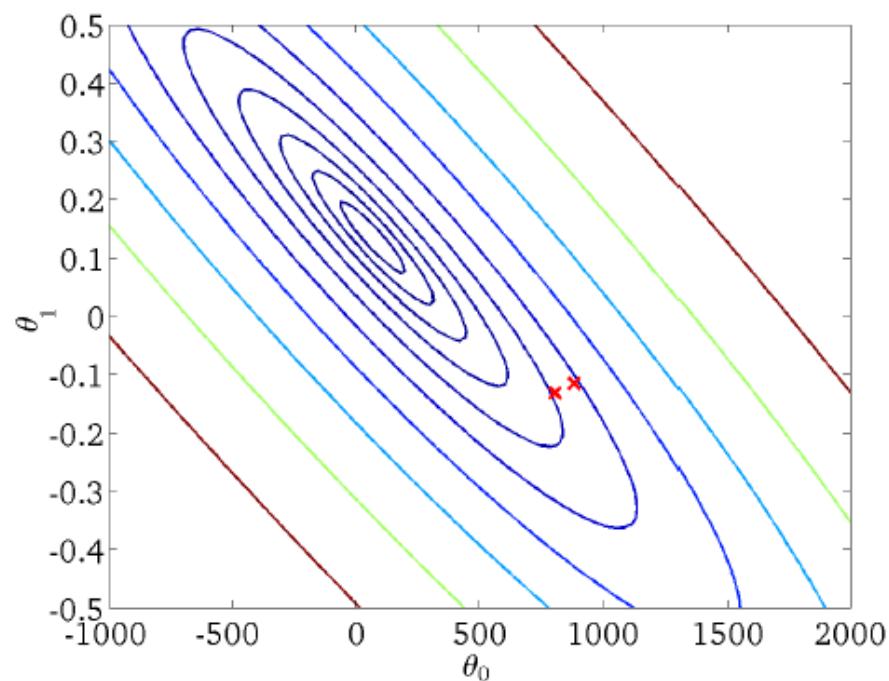
$$h_{\theta}(x)$$

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



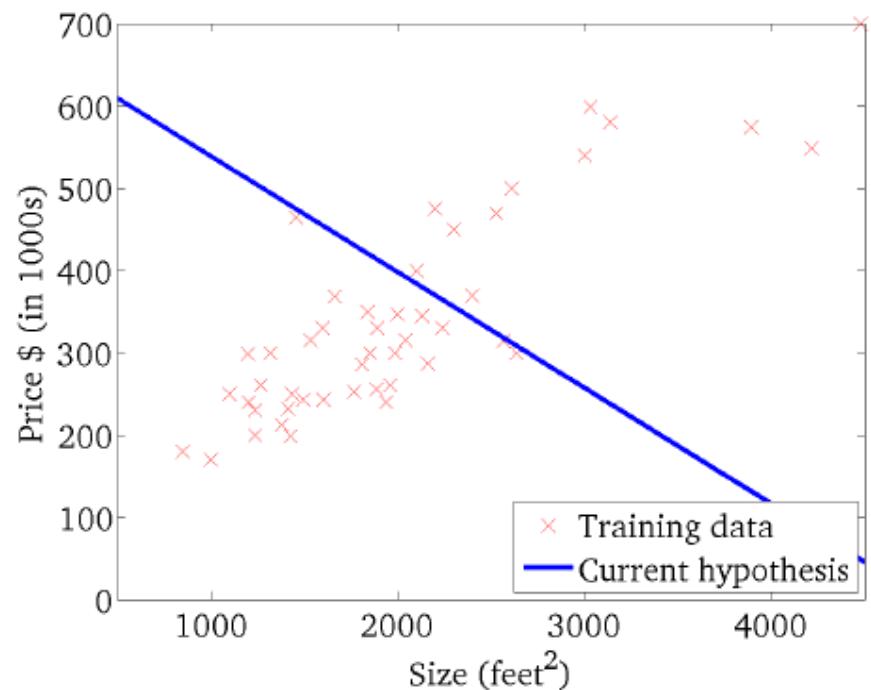
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



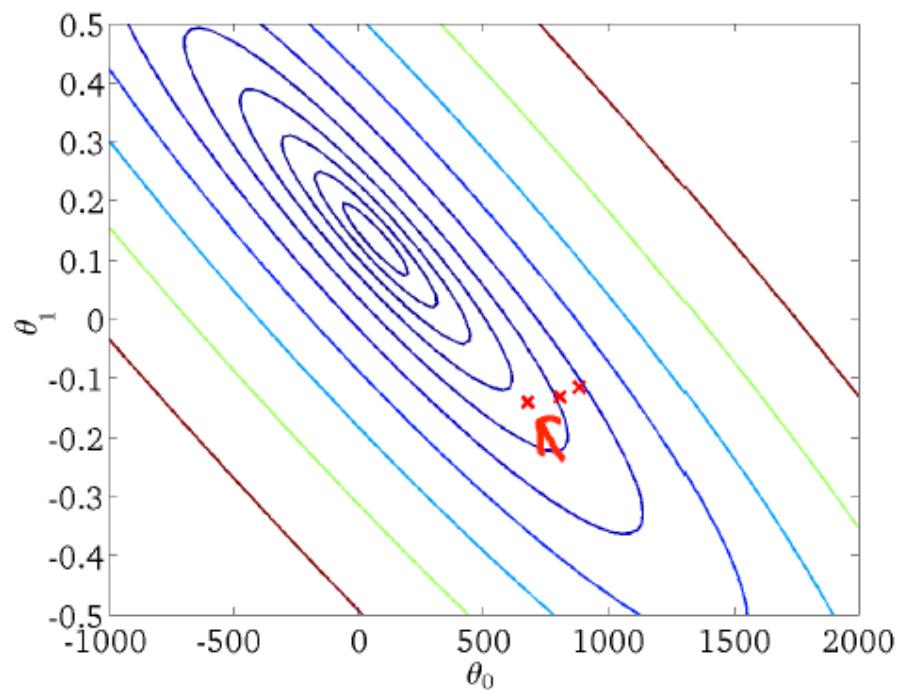
$$h_{\theta}(x)$$

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



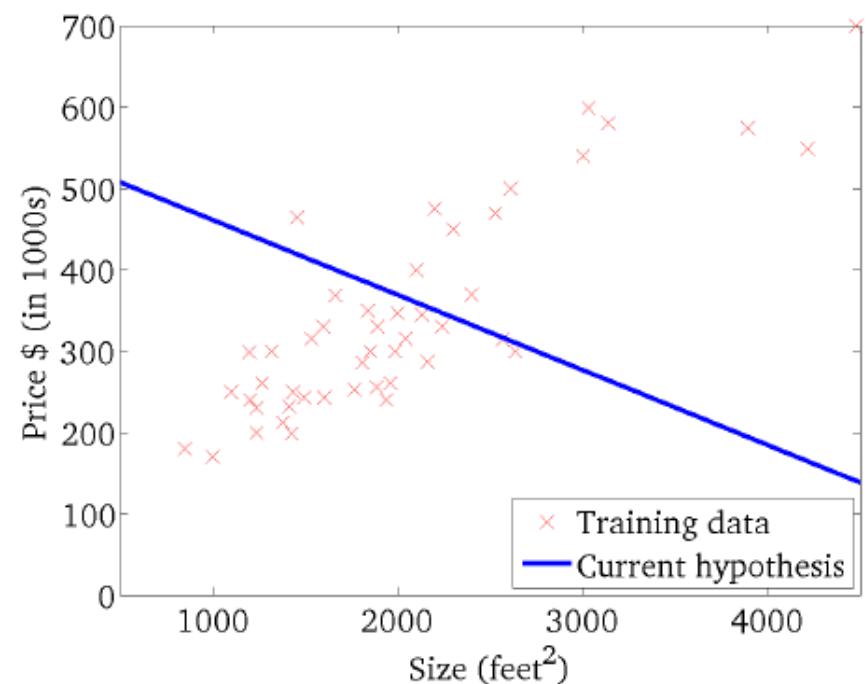
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



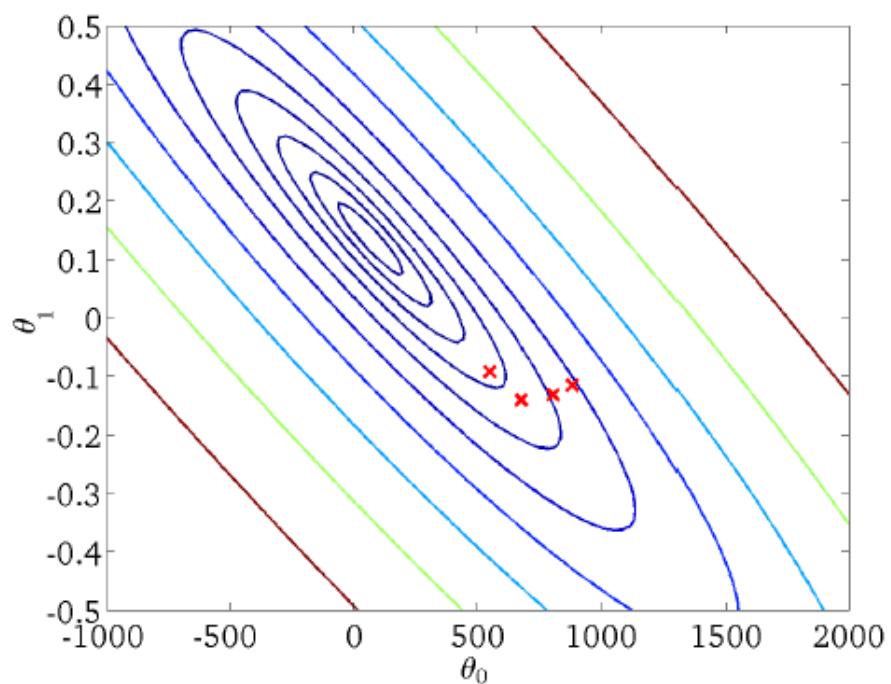
$$h_{\theta}(x)$$

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



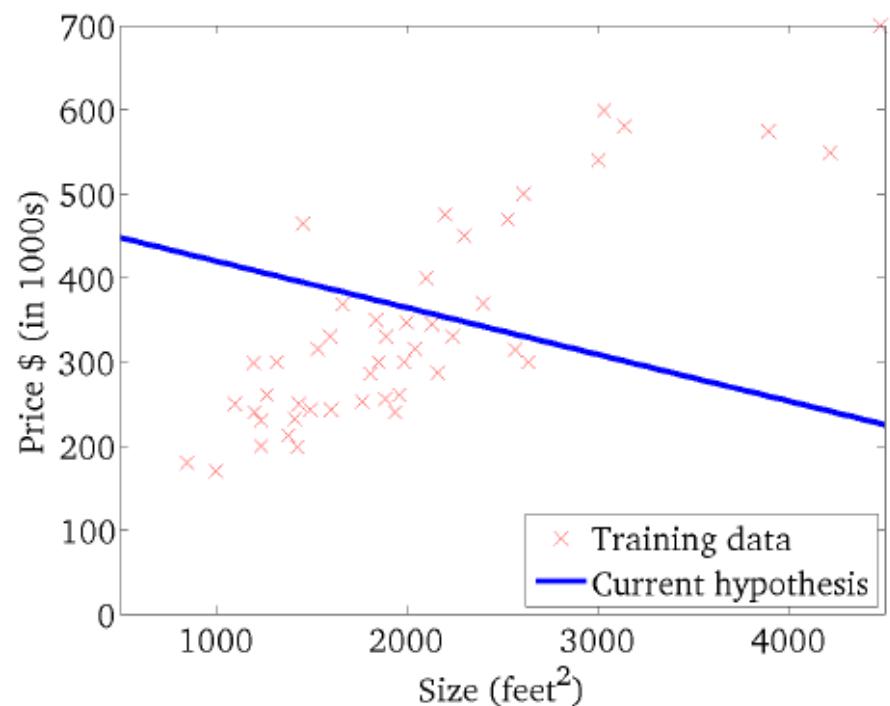
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



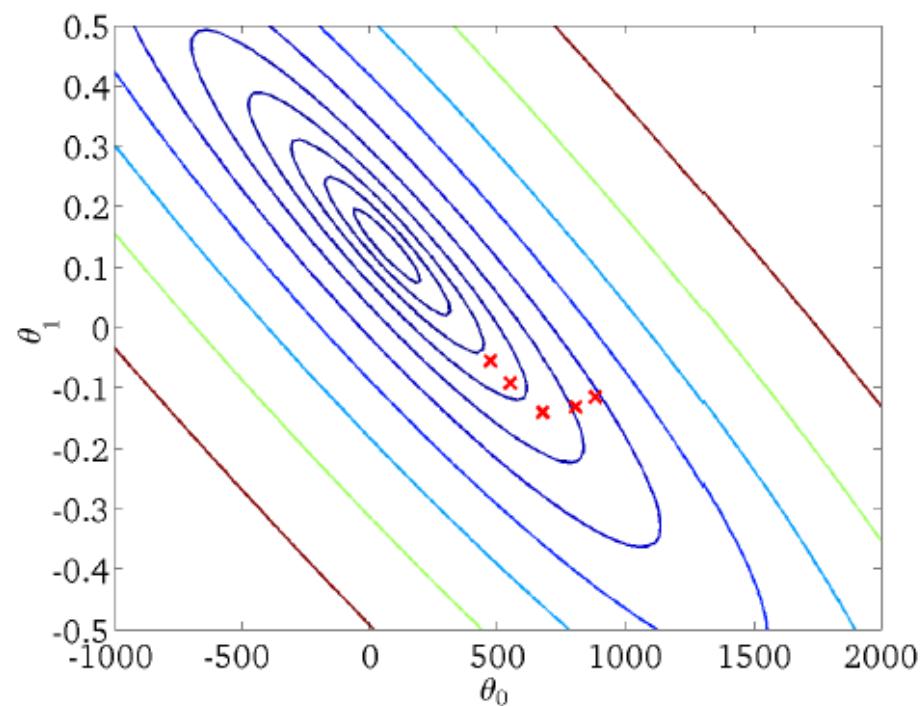
$$h_{\theta}(x)$$

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

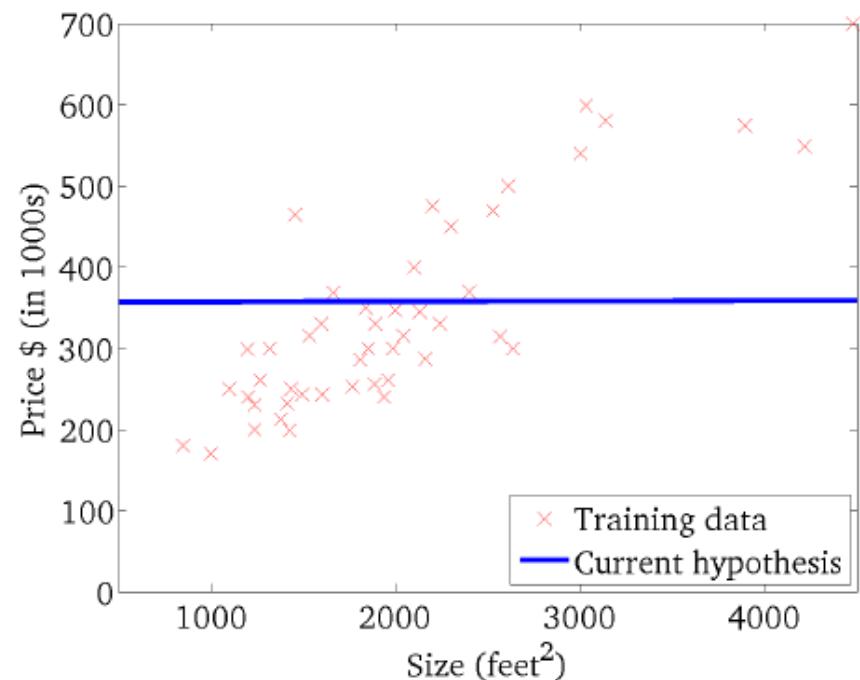


$$J(\theta_0, \theta_1)$$

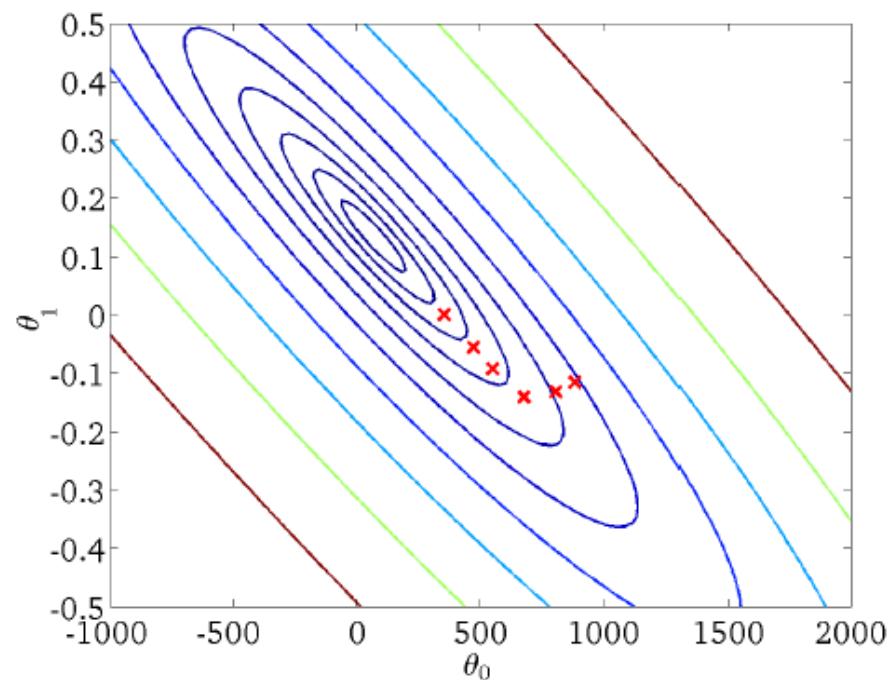
(function of the parameters θ_0, θ_1)



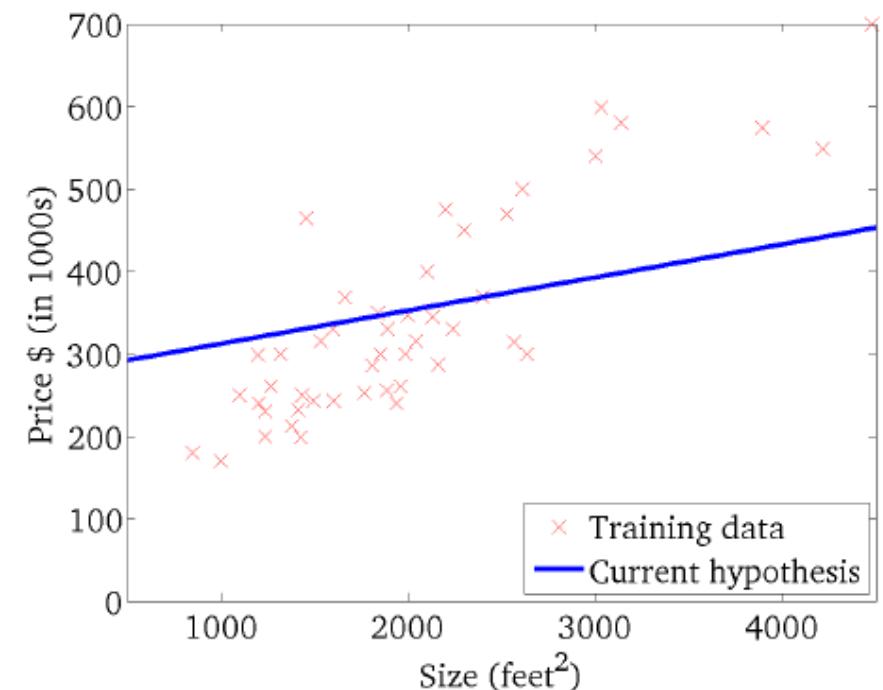
$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)



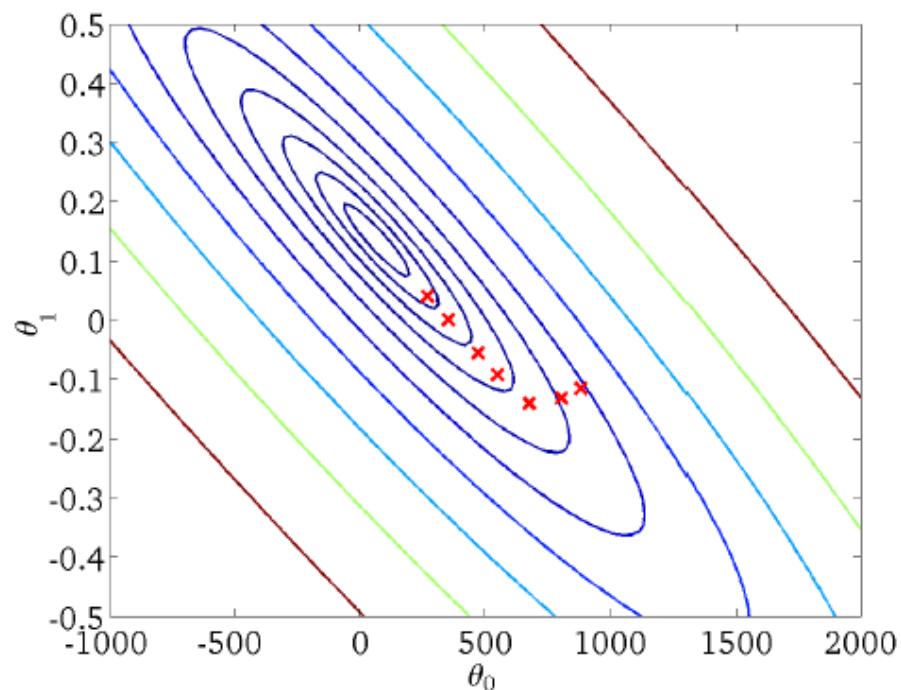
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)



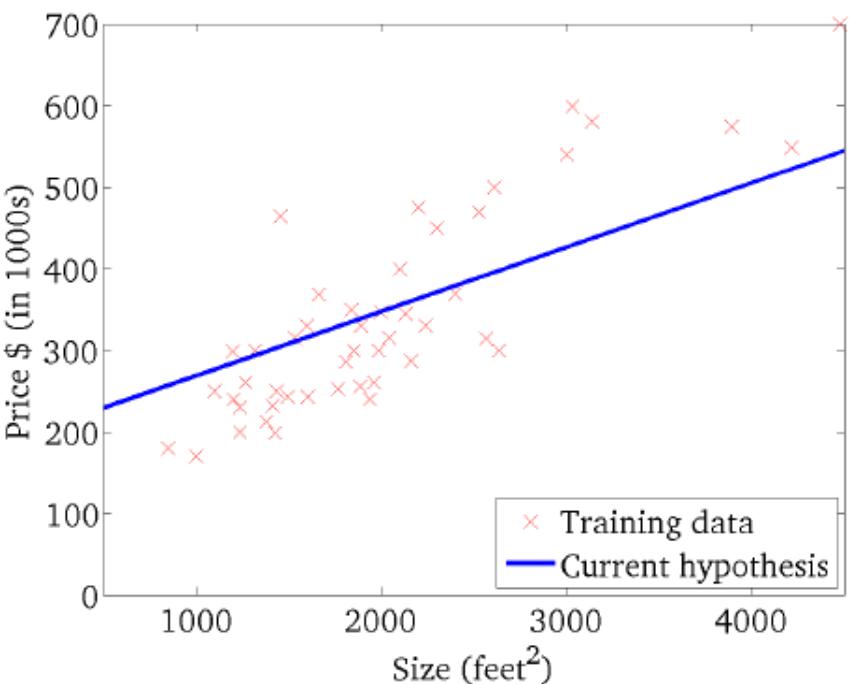
$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)



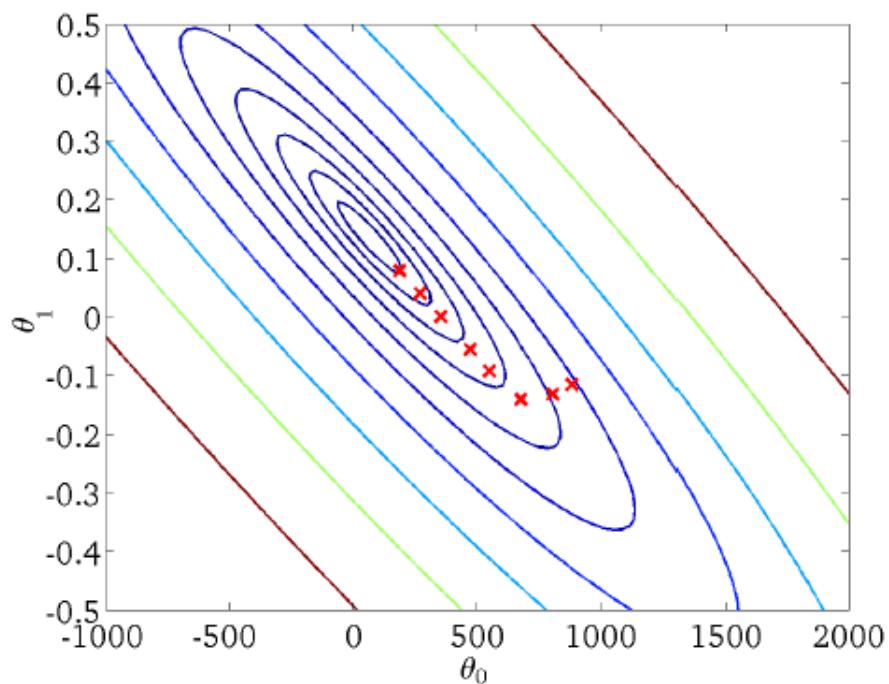
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)



$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)

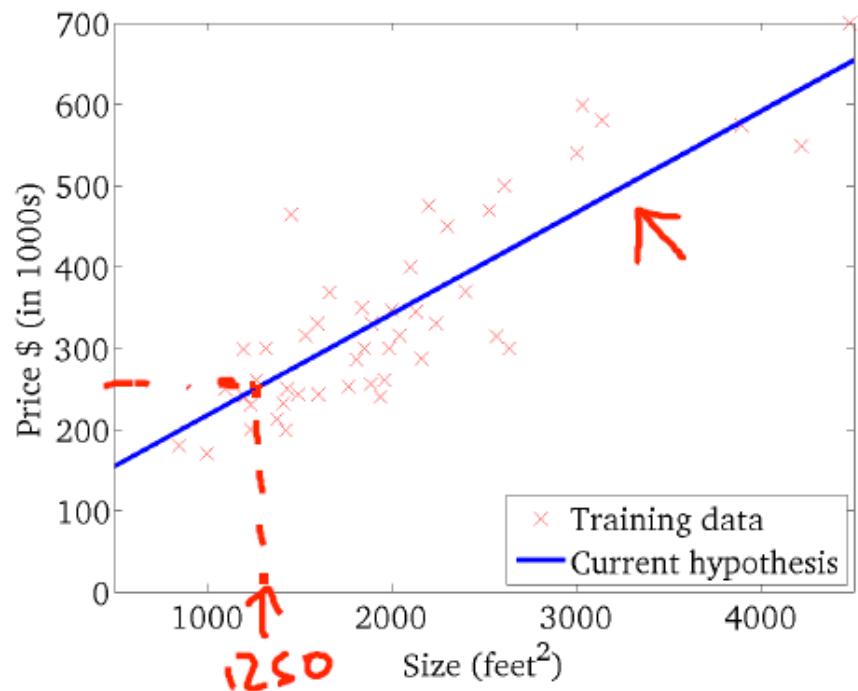


$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)



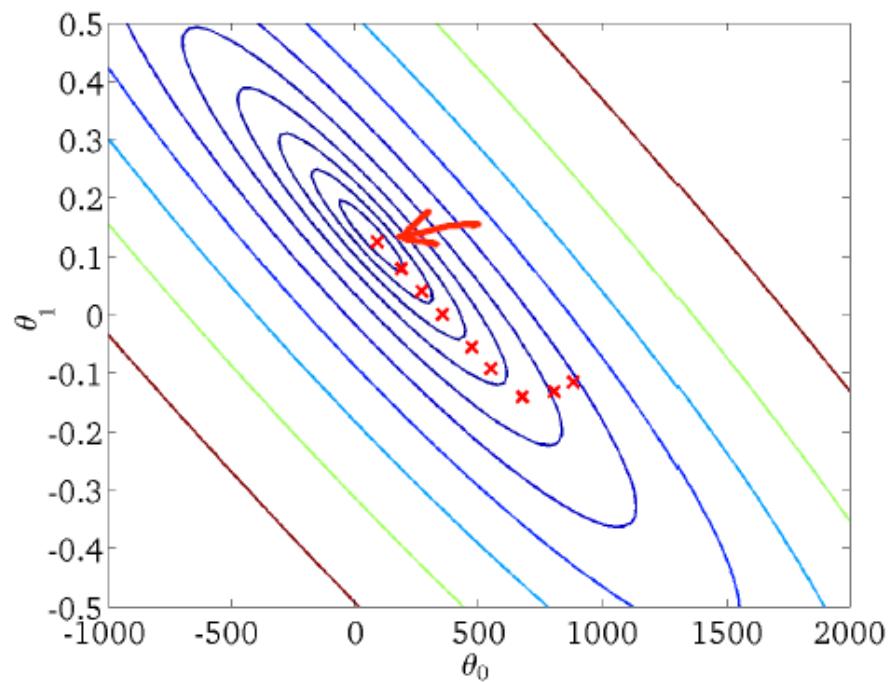
$$h_{\theta}(x)$$

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



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

(function of the parameters θ_0, θ_1)



“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.

$$\rightarrow \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

Linear regression with multiple variables

Multiple features (variables)

Multiple features (variables).

Size (feet ²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Notation:

- n = number of features $n=4$
- $x^{(i)}$ = input (features) of i^{th} training example.
- $x_j^{(i)}$ = value of feature j in i^{th} training example.

$\underline{x}^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$

$x_3^{(2)} = 2$

Hypothesis

Hypothesis:

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

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

E.g. $\underline{h_{\theta}(x)} = \underline{80} + \underline{0.1x_1} + \underline{0.01x_2} + \underline{3x_3} - \underline{2x_4}$

↑ ↑ ↑
 age

- TO DO: Vector notation

Hypothesis: $\underline{h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n}$

$$\xrightarrow{x_0=1}$$

Parameters: $\underline{\theta_0, \theta_1, \dots, \theta_n}$ Θ n+1 - dimensional vector

Cost function:

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

Gradient descent:

Repeat {
 $\rightarrow \theta_j := \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n) \right] \boxed{J(\theta)}$
 }
 ↑ (simultaneously update for every $j = 0, \dots, n$)

Gradient Descent

Previously ($n=1$):

Repeat {



$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \underbrace{\frac{\partial}{\partial \theta_0} J(\theta)}_{}$$



$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \quad (\text{simultaneously update } \theta_0, \theta_1)$$

}

New algorithm ($n \geq 1$):

Repeat {

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

$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad (\text{simultaneously update } \theta_j \text{ for } j = 0, \dots, n)$$

}

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\rightarrow \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \quad \dots$$

Polynomial regression

Housing prices prediction

$$h_{\theta}(x) = \theta_0 + \theta_1 \times \boxed{\text{frontage}} + \theta_2 \times \boxed{\text{depth}}$$

x_1
-

Area

$$\times = \underline{\text{frontage} * \text{depth}}$$

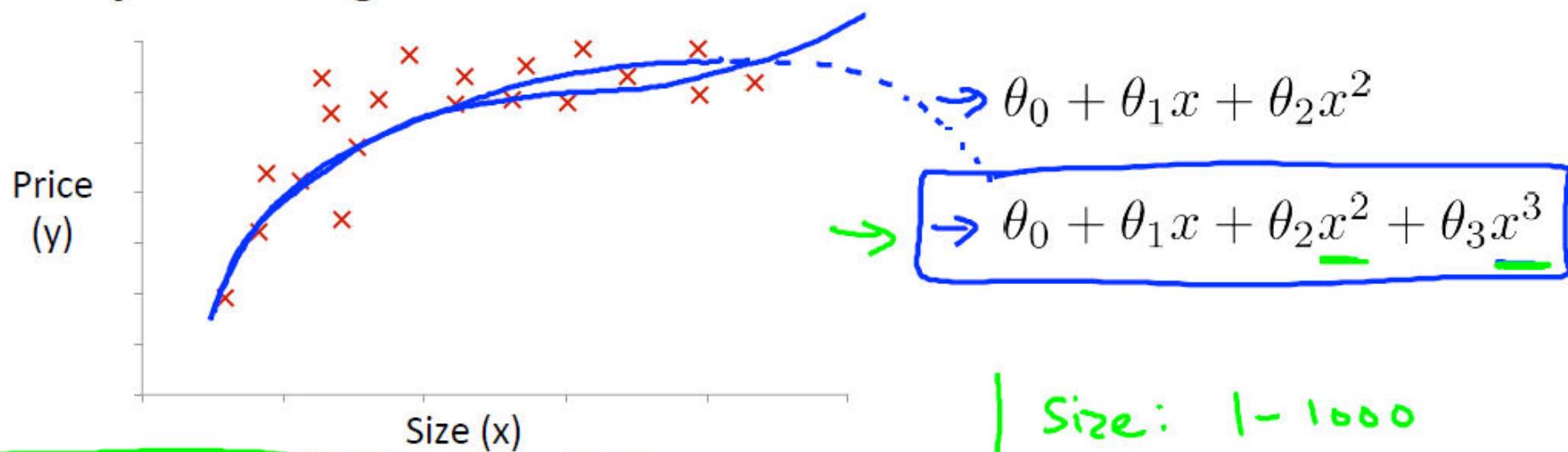


$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

↑ land area

Polynomial regression

Polynomial regression



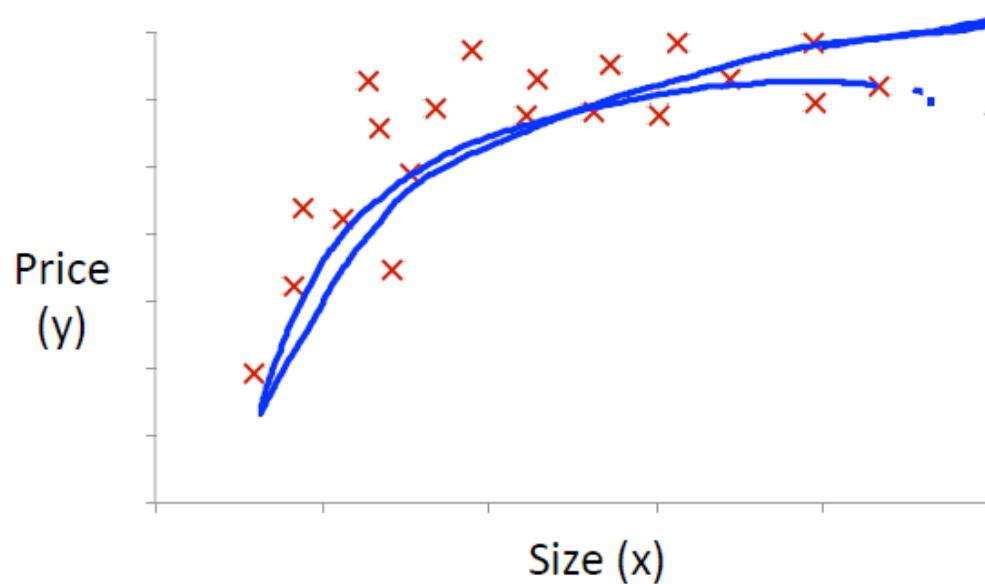
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

$$\begin{aligned} \rightarrow x_1 &= (\text{size}) \\ \rightarrow x_2 &= (\text{size})^2 \\ \rightarrow x_3 &= (\text{size})^3 \end{aligned}$$

Size: 1 - 1000
Size²: 1 - 1000,000
Size³: 1 - 10⁹

Choice of feature

Choice of features



$$\rightarrow h_{\theta}(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$

$$\rightarrow h_{\theta}(x) = \theta_0 + \theta_1(\text{size}) + \theta_2 \sqrt{(\text{size})}$$

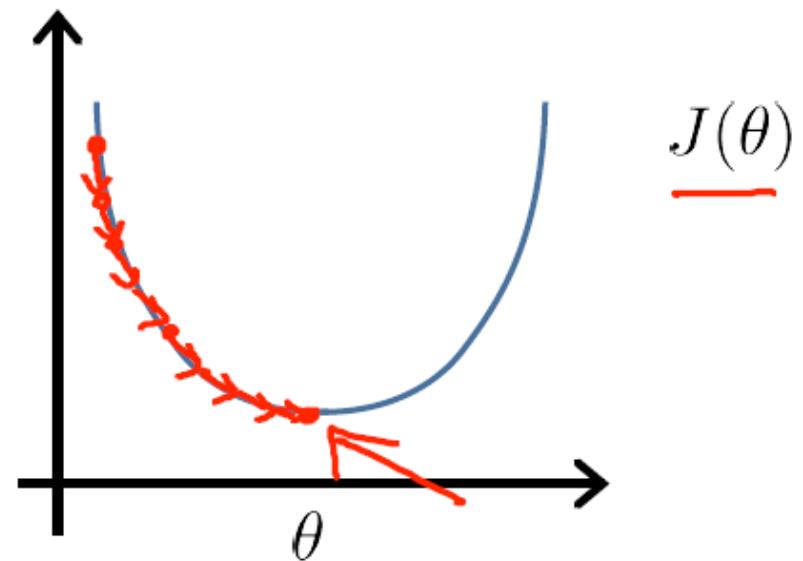


Gradient descent for polynomial regression

- Since the cost function is not convex now, we might get stuck in local minima
- No straight-forward solution for this
- Have to try different initial values for Θ

Normal equation

Gradient Descent



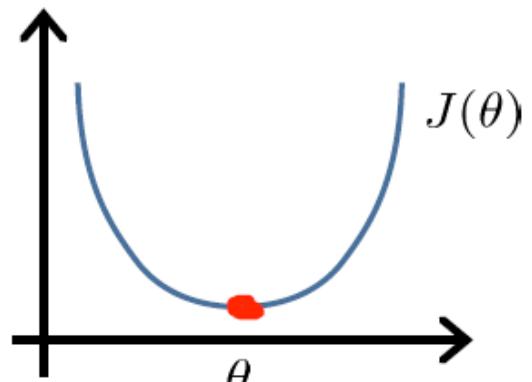
Normal equation: Method to solve for θ
analytically.

Intuition: If 1D ($\theta \in \mathbb{R}$)

$$\rightarrow J(\theta) = a\theta^2 + b\theta + c$$

$$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$$

Solve for θ



$$\theta \in \mathbb{R}^{n+1}$$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) = \dots \stackrel{\text{set}}{=} 0 \quad (\text{for every } j)$$

Solve for $\theta_0, \theta_1, \dots, \theta_n$

Examples: $m = 4$.

x_0	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$

$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$

$m \times (n+1)$

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

Gradient Descent vs Normal Equation

Gradient Descent	Normal Equation
<ul style="list-style-type: none">• Need to choose alpha	<ul style="list-style-type: none">• No need to choose alpha
<ul style="list-style-type: none">• Need many iterations	<ul style="list-style-type: none">• Do not need iterations
<ul style="list-style-type: none">• Works well even when n is large	<ul style="list-style-type: none">• Need to compute $O(n^3)$• Slow if n is very large

m training examples, n features.

Gradient Descent

- • Need to choose α .
- • Needs many iterations.
- Works well even when n is large.

$$\overbrace{n = 10^6}^{n \text{ is large}}$$

Normal Equation

- • No need to choose α .
- • Don't need to iterate.
- Need to compute $(X^T X)^{-1}$
- $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
- Slow if n is very large.

$$\begin{array}{l} n = 100 \\ n = 1000 \end{array}$$

$$n = 10000$$

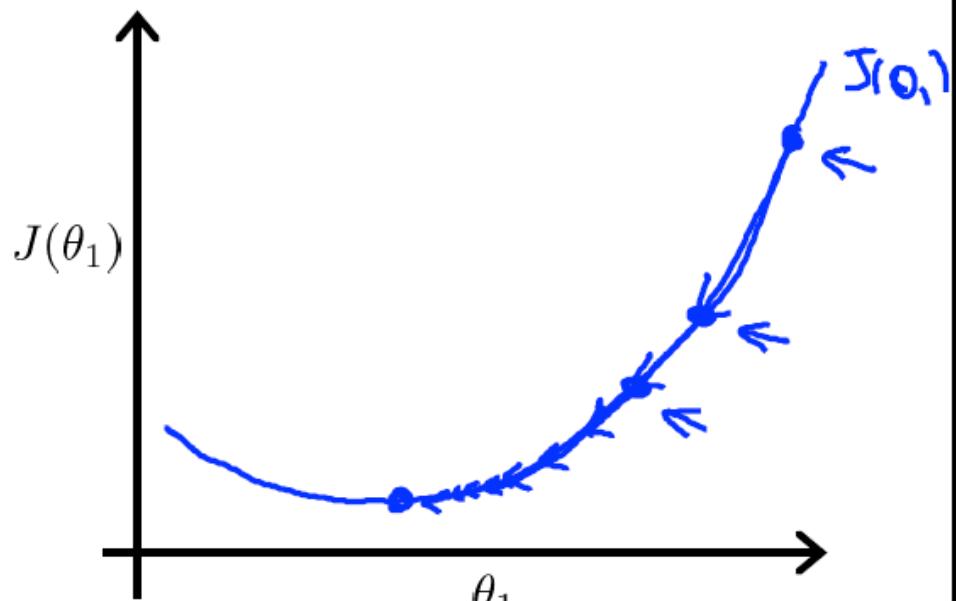
Common issues and how to overcome them

- Learning rate
- Feature Scaling
- Overfitting and underfitting (bias and variance)

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

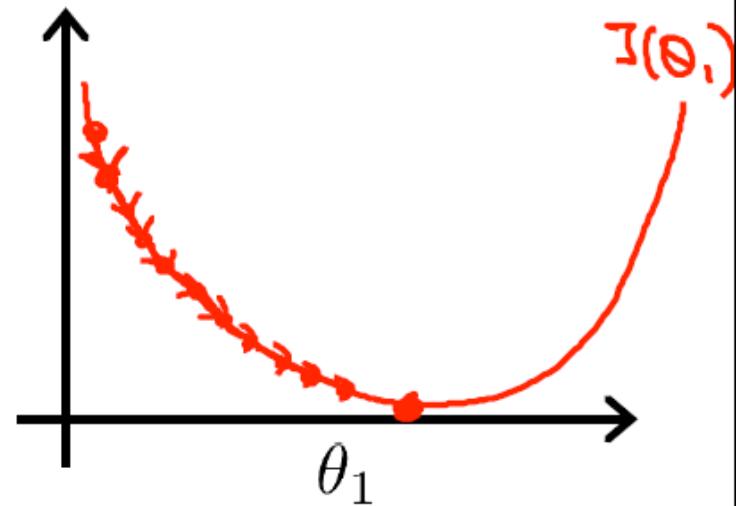
As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



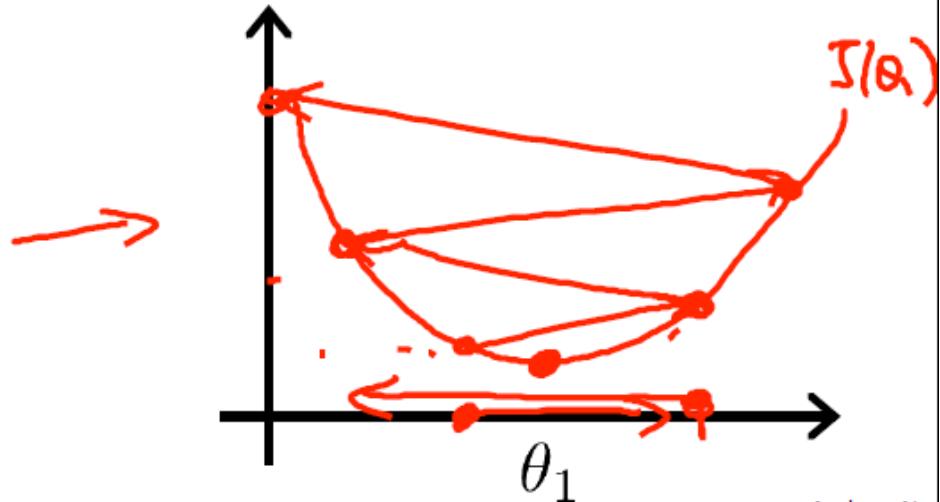
Learning rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



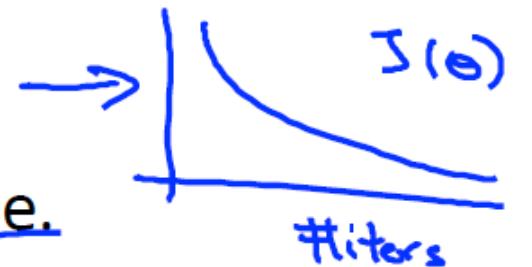
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



SUMMARY

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge. (Slow converge also possible)



To choose α , try

$$\dots, \underbrace{0.001}_{\uparrow}, \underbrace{0.003}_{\approx 3x}, \underbrace{0.01}_{\uparrow}, \underbrace{0.03}_{\approx 3x}, \underbrace{0.1}_{\uparrow}, \underbrace{0.3}_{\approx 3x}, \underbrace{1}_{\uparrow}, \dots$$

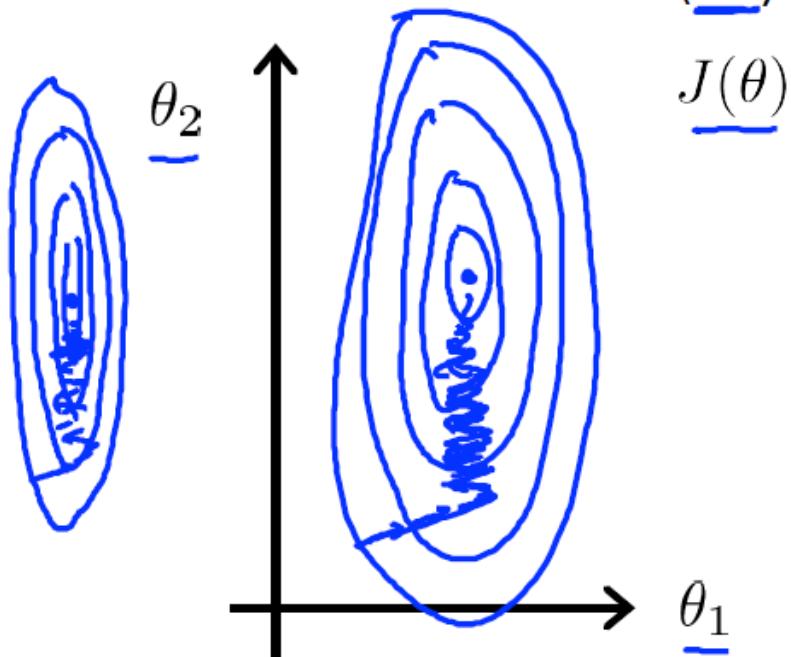
Feature scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. $x_1 = \text{size } (0\text{-}2000 \text{ feet}^2)$ ↪

$x_2 = \text{number of bedrooms } (1\text{-}5)$ ↪

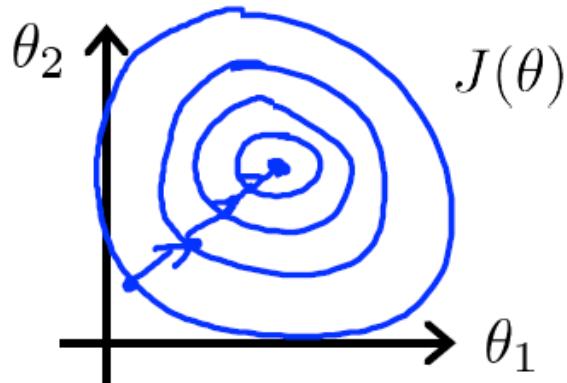


→ $x_1 = \frac{\text{size (feet}^2)}{2000}$ ↪

→ $x_2 = \frac{\text{number of bedrooms}}{5}$ ↪

$0 \leq x_1 \leq 1$

$0 \leq x_2 \leq 1$



Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

$$x_0 = 1$$

$$6 \leq x_1 \leq 3 \quad \checkmark$$

$$-2 \leq x_2 \leq 0.5 \quad \checkmark$$

$$-100 \leq x_3 \leq 100 \quad \times$$

$$-0.0001 \leq x_4 \leq 0.0001 \quad \times$$

$$\boxed{-1 \leq x_i \leq 1}$$

$$-3 \rightarrow 3 \quad \checkmark$$

$$-\frac{1}{3} \rightarrow \frac{1}{3} \quad \checkmark$$

- TO DO : Mean normalization

Evaluating your hypothesis

Evaluating your hypothesis

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
<hr/>	
1427	199
1380	212
1494	243

70%

Training set

30%

Test set

\rightarrow

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

\rightarrow

$(x_{test}^{(1)}, y_{test}^{(1)})$
 $(x_{test}^{(2)}, y_{test}^{(2)})$
⋮
 $(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test} = no.
of test example
 $(x_{test}^{(1)}, y_{test}^{(1)})$

U1

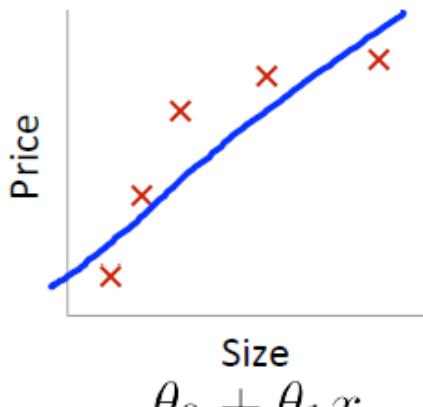
Training/testing procedure for linear regression

Training/testing procedure for linear regression

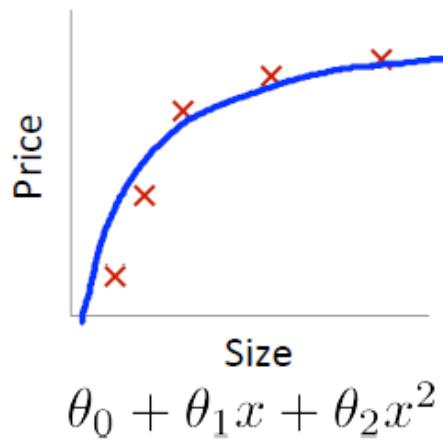
- - Learn parameter $\underline{\theta}$ from training data (minimizing training error $J(\theta)$) 70%
- Compute test set error:

$$J_{\text{test}}(\theta) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left(h_{\theta}(x_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)} \right)^2$$

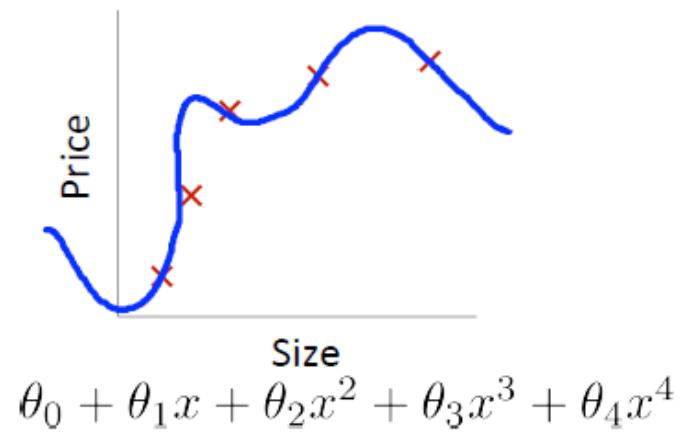
Bias/variance



High bias
(underfit)
 $d=1$



"Just right"
 $d=2$



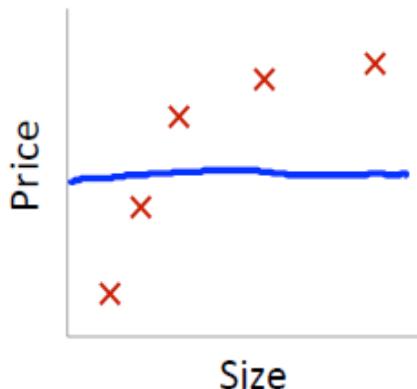
High variance
(overfit)
 $d=4$

Linear regression regularization

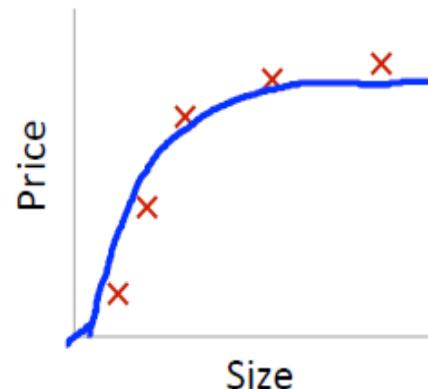
Linear regression with regularization

Model:
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

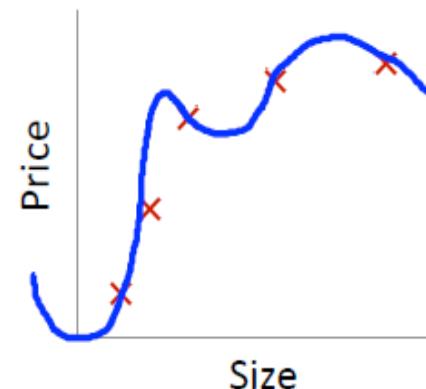
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^m \theta_j^2$$



Large λ ←



Intermediate λ ←
"Just right"



→ Small λ
High variance (overfit)
→ $\lambda = 0$

→ High bias (underfit)
 $\lambda = 10000$. $\theta_1 \approx 0, \theta_2 \approx 0, \dots$
 $h_{\theta}(x) \approx \theta_0$

Model selection

- $\rightarrow d = \text{degree of polynomial}$ ↓
- $d=1$ 1. $\underline{h_\theta(x) = \theta_0 + \theta_1 x} \rightarrow \Theta^{(1)} \rightarrow J_{\text{test}}(\Theta^{(1)})$
- $d=2$ 2. $\underline{h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2} \rightarrow \Theta^{(2)} \rightarrow J_{\text{test}}(\Theta^{(2)})$
- $d=3$ 3. $\underline{h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3} \rightarrow \Theta^{(3)} \rightarrow J_{\text{test}}(\Theta^{(3)})$
- ⋮ ⋮ ⋮
- $d=10$ 10. $\underline{h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}} \rightarrow \Theta^{(10)} \rightarrow J_{\text{test}}(\Theta^{(10)})$

Choose $\underline{\theta_0 + \dots + \theta_5 x^5} \leftarrow$

How well does the model generalize? Report test set error $J_{\text{test}}(\underline{\theta^{(5)}})$.

$\Theta^{(5)}$

$\Theta_0, \Theta_1, \dots$

Problem: $J_{\text{test}}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

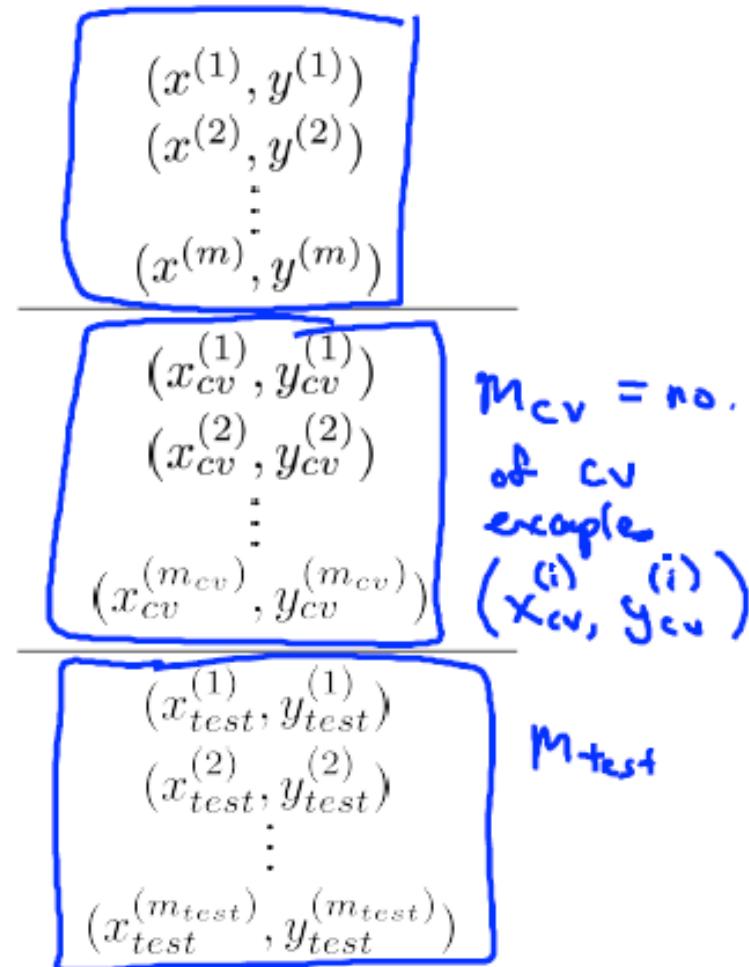
Evaluating your hypothesis

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Annotations:

- Handwritten blue text: "50%" above the first five rows.
- Handwritten blue bracket: A curly brace groups the first five rows under the heading "Training set".
- Handwritten blue bracket: A curly brace groups the next two rows under the heading "Cross validation (cv)".
- Handwritten blue bracket: A curly brace groups the last two rows under the heading "test set".
- Handwritten blue arrow: An arrow points from the "Training set" bracket to the top box containing $(x^{(1)}, y^{(1)})$ through $(x^{(m)}, y^{(m)})$.
- Handwritten blue arrow: An arrow points from the "Cross validation (cv)" bracket to the middle box containing $(x_{cv}^{(1)}, y_{cv}^{(1)})$ through $(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$.
- Handwritten blue arrow: An arrow points from the "test set" bracket to the bottom box containing $(x_{test}^{(1)}, y_{test}^{(1)})$ through $(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$.



Train / validation / test error

Train/validation/test error

Training error:

$$\rightarrow J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cross Validation error:

$$\rightarrow J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

$$\rightarrow J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Overfitting and Underfitting

- If the linear regression/learning algorithm, is suffering from overfitting, getting more training data may help.
- If the linear regression/learning algorithm is suffering from underfitting, the hypothesis has to be changed (the order of the hypothesis has to be increased/or made more complex in the case of Linear Regression)
- Else, can adjust the regularization term to fix overfitting or underfitting

Summary

- Linear regression is used to predict continuous values based on historical data
- Supervised learning technique (as historical values are there)
- Linear regression with single feature
- Linear regression with multiple features
- Polynomial regression

Summary

- Gradient descent (iterative method to solve regression problems)
- Normal equation (analytical method to solve regression problems)
- Common issues
 - Learning rate
 - Feature scaling
 - Overfitting and underfitting

Lecture 4 – Logistic Regression for Classification

Classification

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant/ Benign ? -

Classification

- Email: Spam / Not Spam?
 - Online Transactions: Fraudulent (Yes / No)?
 - Tumor: Malignant / Benign ?
- $y \in \{0, 1\}$
- 0: "Negative Class" (e.g., benign tumor)
- 1: "Positive Class" (e.g., malignant tumor)
- $y \in \{0, 1, 2, 3\}$

- TO DO: Why not linear regression?

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

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

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

(Classification)

Logistic Regression Model

Want $0 \leq h_\theta(x) \leq 1$

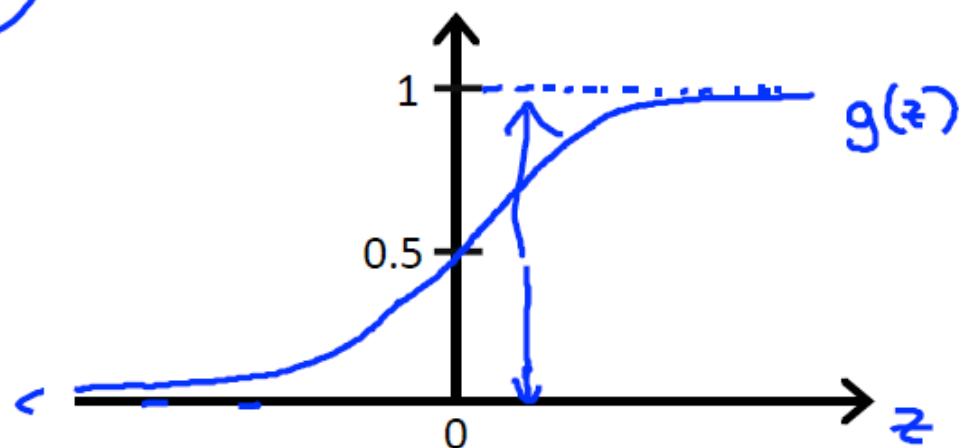
$$h_\theta(x) = g(\theta^T x)$$

$$\rightarrow g(z) = \frac{1}{1 + e^{-z}}$$

$\theta^T x$

- ↳ Sigmoid function
- ↳ Logistic function

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



Parameters $\underline{\theta}$.

Logistic regression

$$h_{\theta}(x) = g(\theta^T x)$$

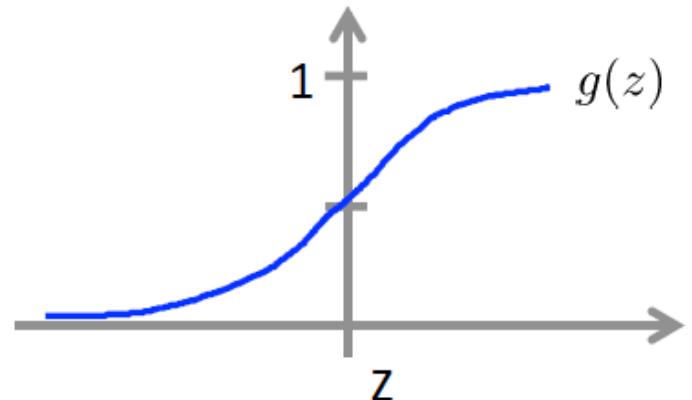
$$g(z) = \frac{1}{1+e^{-z}}$$

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

$$\theta^T x \geq 0$$

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

$$\theta^T x < 0$$



$$g(z) \geq 0.5 \\ \text{when } z \geq 0$$

$$h_{\theta}(x) = g(\theta^T x)$$

$$g(z) < 0.5 \\ \text{when } z < 0$$

- TO DO: Probabilistic interpretation

TO DO: Decision boundary

TO DO: Non linear Decision boundary

Training

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

set:

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad \mathbb{R}^{n+1}$$

$$x_0 = 1, y \in \{0, 1\}$$

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

How to choose parameters θ ?

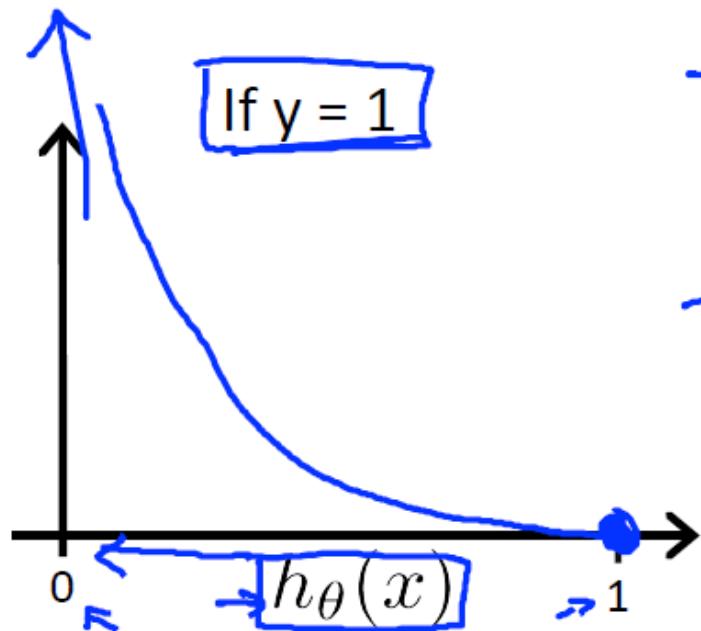
TO DO: Cost function

TO DO: Logistic regression cost function

Logistic regression cost function

Logistic regression cost function

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

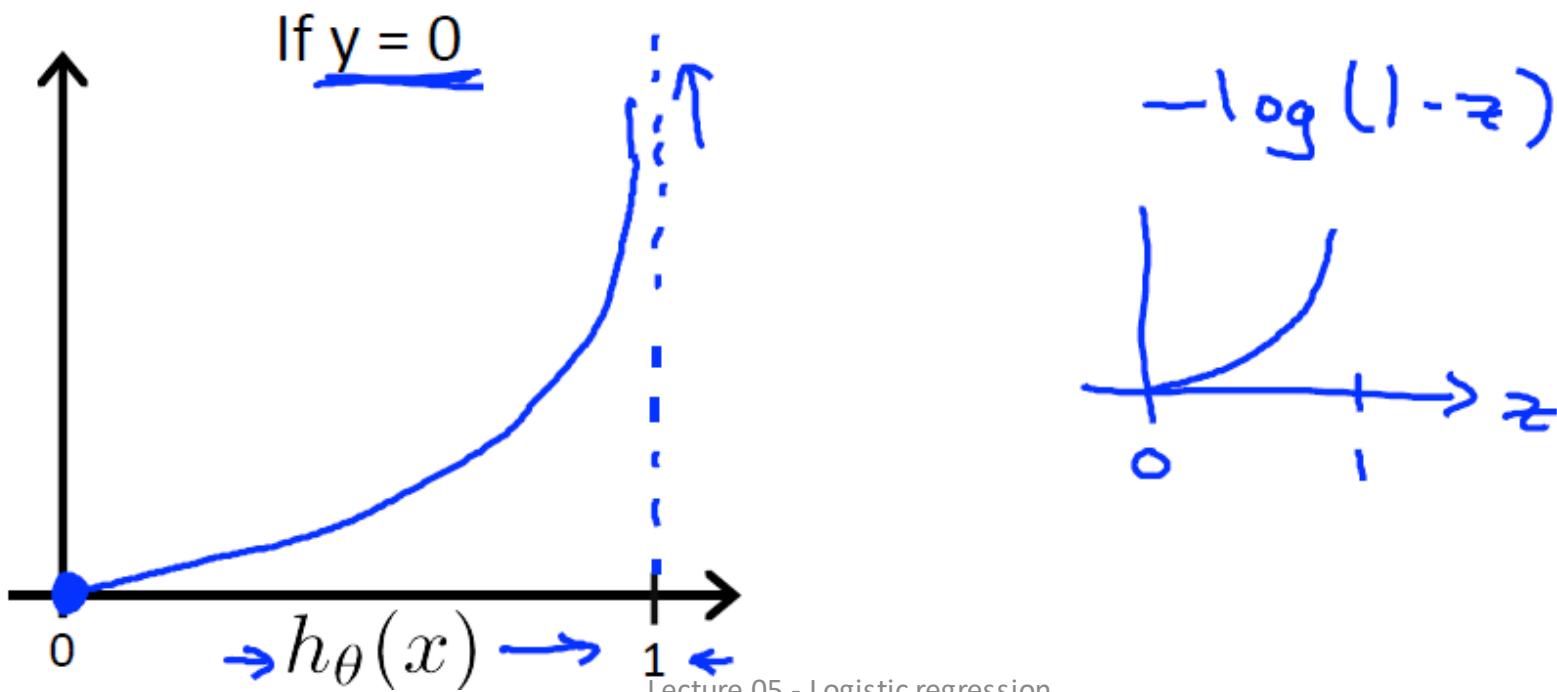


- Cost = 0 if $y = 1$, $h_{\theta}(x) = 1$
 - [But as $h_{\theta}(x) \rightarrow 0$
Cost $\rightarrow \infty$]
- Captures intuition that if $h_{\theta}(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

Logistic regression cost function

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



TO DO: Simplified cost function

Logistic regression cost function

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

To fit parameters θ :

$$\min_{\theta} J(\theta) \quad \text{Get } \underline{\theta}$$

To make a prediction given new x :

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

$$\underline{\text{ply}=1 | x; \theta)}$$

Gradient Descent

Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all θ_j)

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

Gradient Descent

Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all θ_j)

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad \text{for } i=0 \dots n$$

$$h_\theta(x) = \Theta^T x$$

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

Algorithm looks identical to linear regression!

Multiclass classification

Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

$$y=1 \quad y=2 \quad y=3 \quad y=4$$

Medical diagrams: Not ill, Cold, Flu

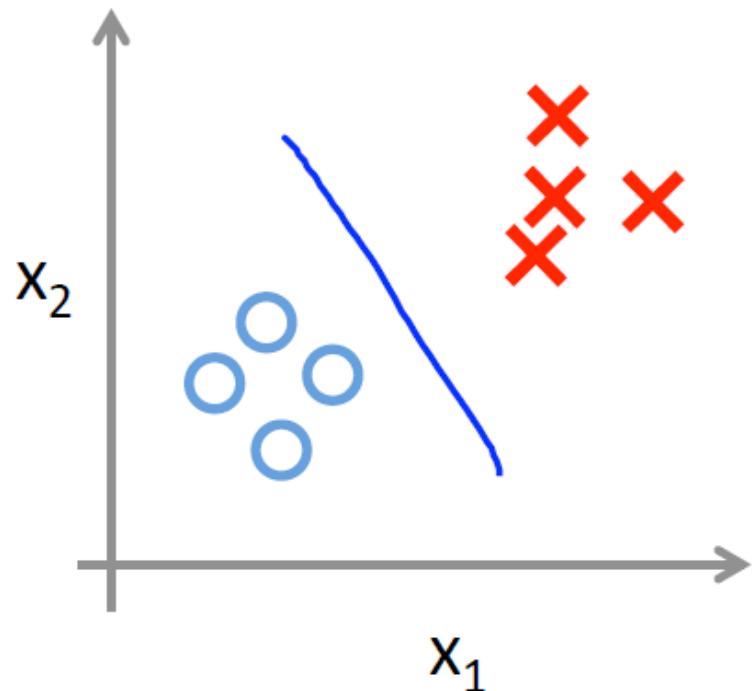
$$y=1 \quad 2 \quad 3$$

Weather: Sunny, Cloudy, Rain, Snow

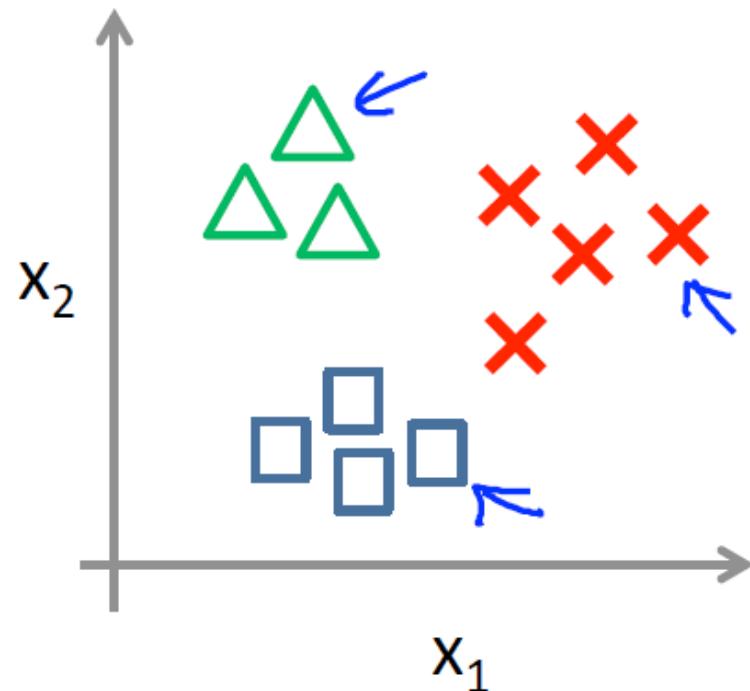
$$\begin{array}{cccc} y=1 & 2 & 3 & 4 \\ \hline 0 & + & 1 & 2 \end{array} \leftarrow$$

Andrew Ng

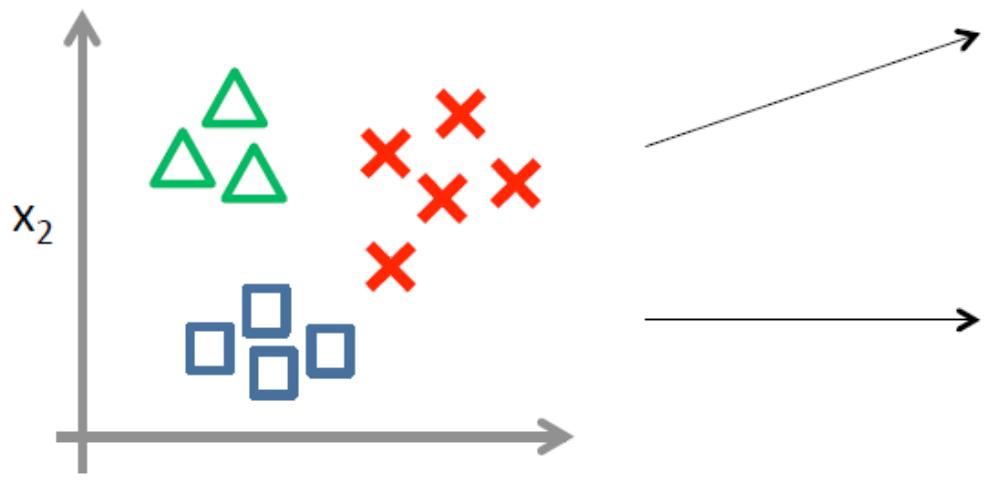
Binary classification:



Multi-class classification:

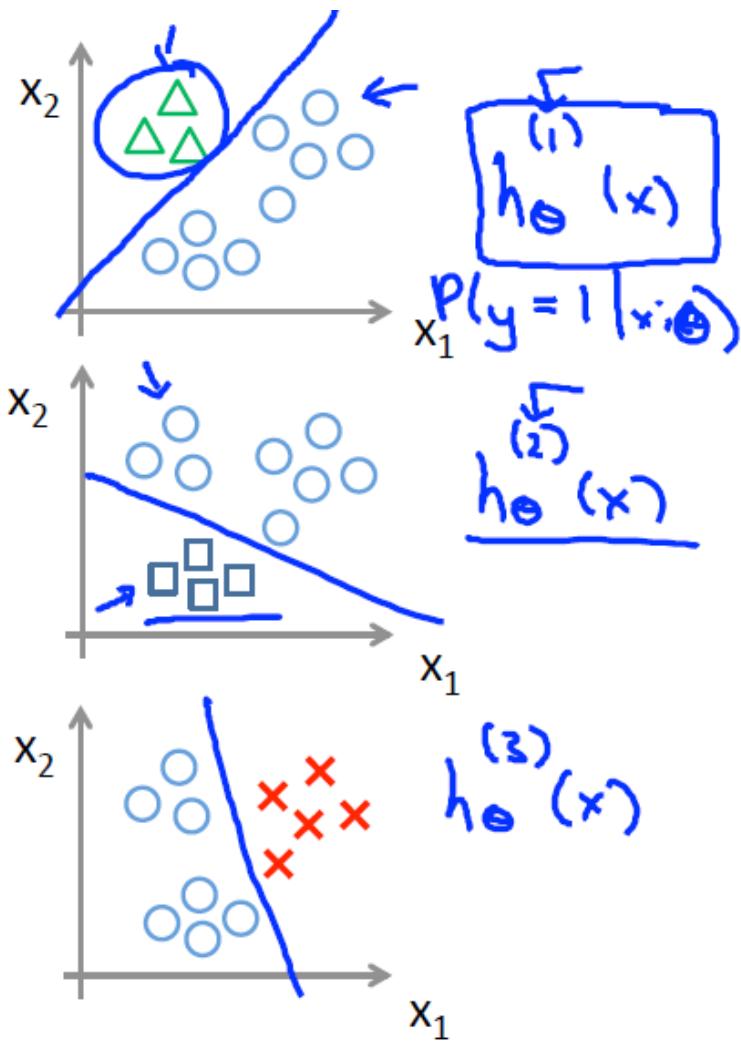


One-vs-all (one-vs-rest):



Class 1: ←
Class 2: ←
Class 3: ←

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



One-vs-all

One-vs-all

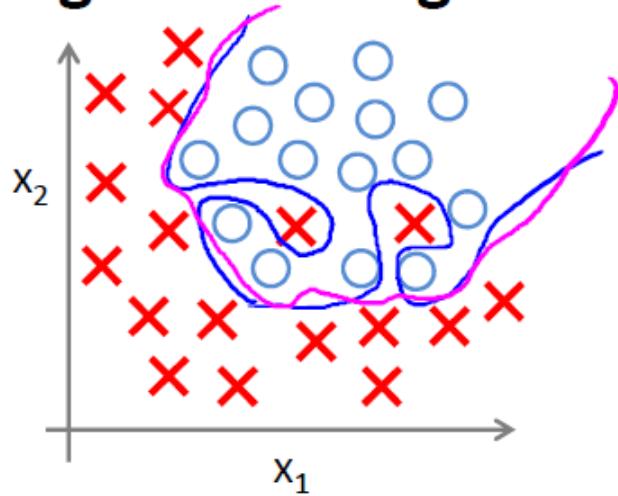
Train a logistic regression classifier $\underline{h_{\theta}^{(i)}(x)}$ for each class i to predict the probability that $\underline{y = i}$.

On a new input \underline{x} , to make a prediction, pick the class i that maximizes

$$\max_i \underline{\underline{h_{\theta}^{(i)}(x)}}$$

Regularized logistic regression

Regularized logistic regression.



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

Cost function:

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

$\theta_1, \theta_2, \dots, \theta_n$

Gradient descent

Gradient descent

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \leftarrow$$

$(j = \cancel{X}, 1, 2, 3, \dots, n)$
 $\theta_1, \dots, \theta_n$

}

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

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

Optimization algorithms

- **Algorithms**

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Given θ , we have code that can compute

$$\begin{aligned} & - J(\theta) \\ & - \frac{\partial}{\partial \theta_j} J(\theta) \end{aligned}$$

(for $j = 0, 1, \dots, n$)

- **Advantages**

- No need to manually pick alpha
- Often faster than gradient descent.

- **Disadvantages:**

- More complex

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

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

$\theta_1 = 5, \theta_2 = 5$.

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

```
function [jVal, gradient] = costFunction(theta)
    jVal = (theta(1)-5)^2 + ...
            (theta(2)-5)^2;
    gradient = zeros(2,1);
    gradient(1) = 2*(theta(1)-5);
    gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...  
= fminunc(@costFunction, initialTheta, options);
```

$\theta \in \mathbb{R}^2 \quad d \geq 2$.

Lecture 5 – Support Vector Machines

TO DO: Intuition

TO DO: Dot product

TO DO: Margin

- SVM is called a ‘large margin classifier’ as it tries to maximize the margin between the positive and negative support vectors

Linear SVM – Representation

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem*:

Find \mathbf{w} and b such that

$$= \frac{2}{\|\mathbf{w}\|} \quad \text{is maximized; and for all } \{(\mathbf{x}_i, y_i)\}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1; \quad \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- A better formulation ($\min ||\mathbf{w}|| = \max 1/ ||\mathbf{w}||$):

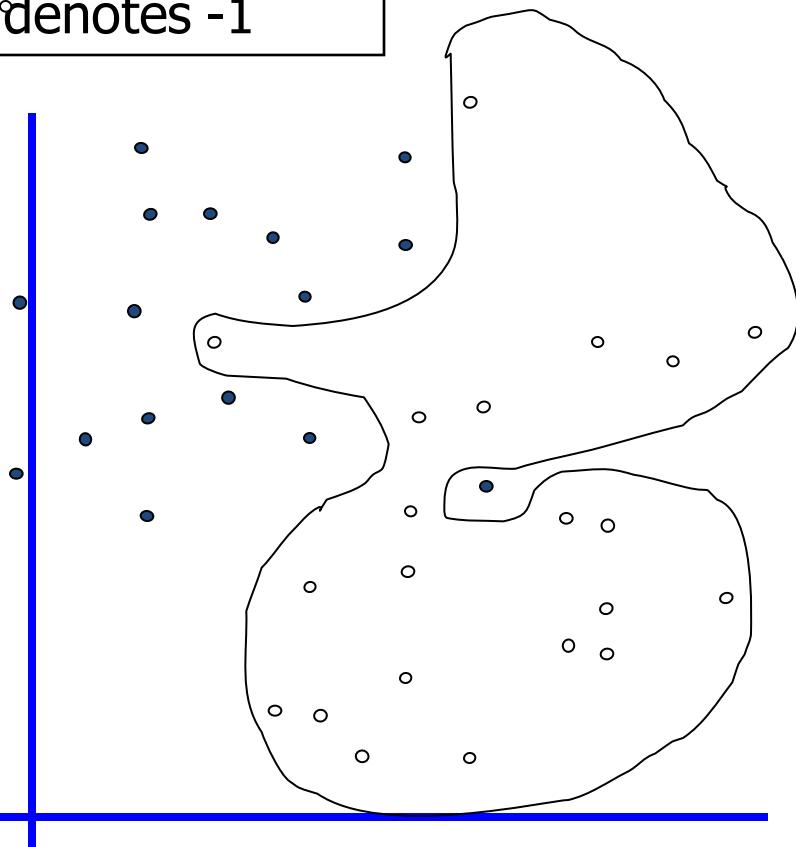
Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \text{ is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

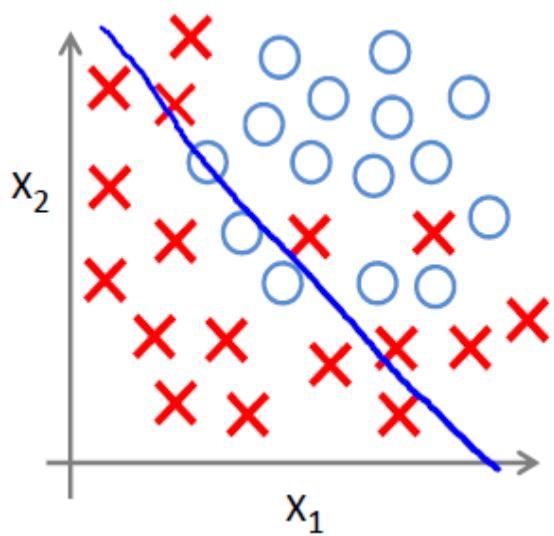
Dataset with noise

• denotes +1
• denotes -1

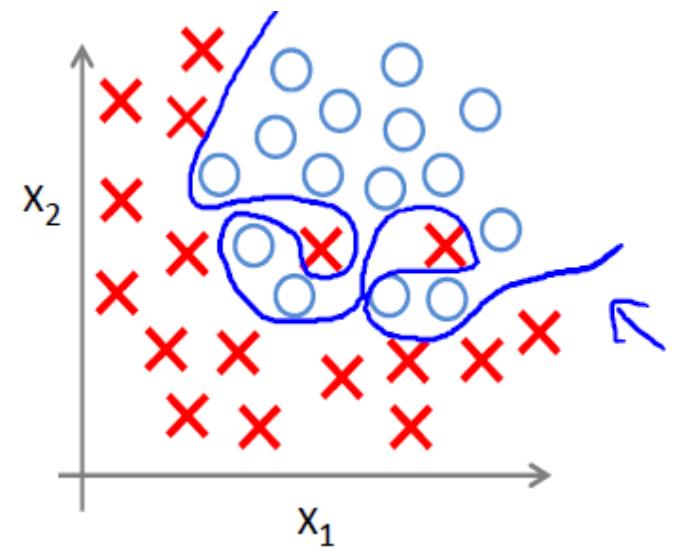
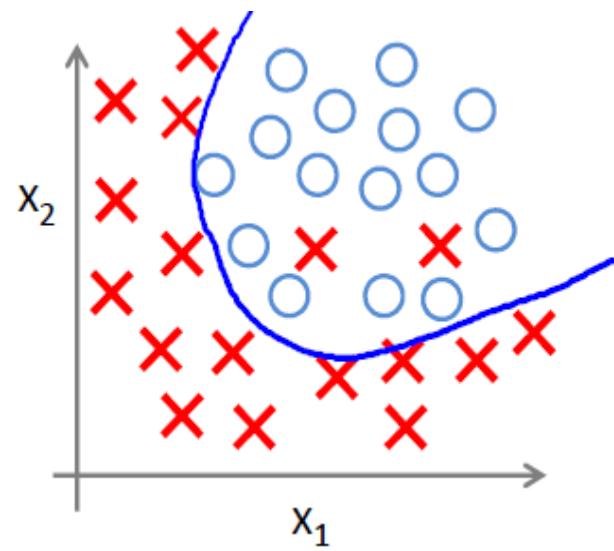


- **Hard Margin:** So far we require all data points be classified correctly
 - No training error
- **What if the training set is noisy?**
 - **Solution 1:** use very powerful kernels

OVERFITTING!



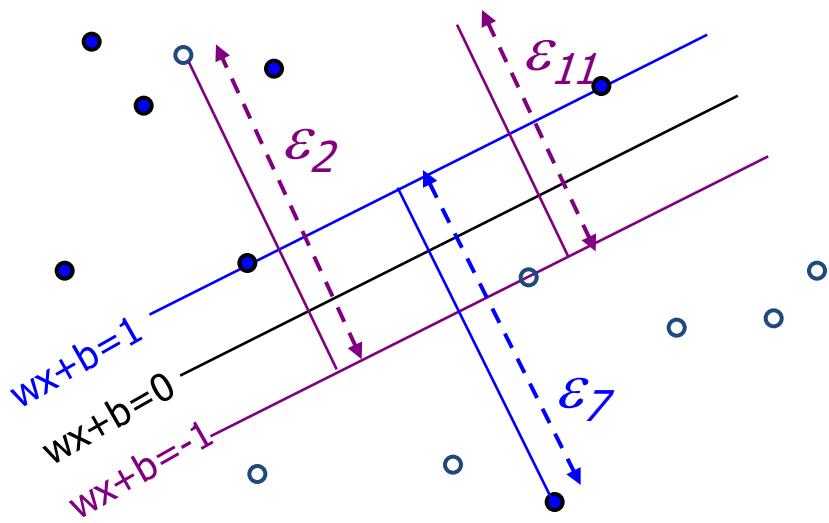
Underfitting



Overfitting

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.



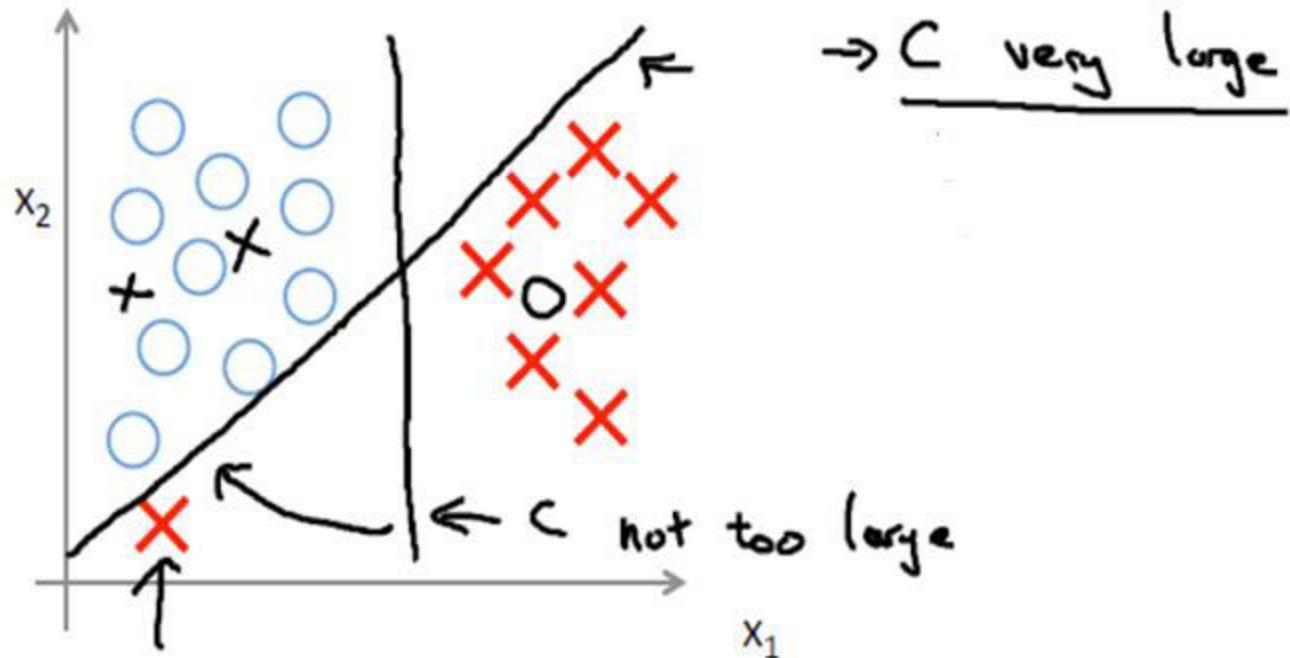
What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} | \mathbf{w} |^2 + C \sum_{k=1}^R \varepsilon_k$$

Large margin classifier in presence of outliers

Large margin classifier in presence of outliers



Hard Margin v.s. Soft Margin

- The old formulation:

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} |\mathbf{w}|^2 \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

- The new formulation incorporating slack variables:

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} |\mathbf{w}|^2 + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$

- Parameter C can be viewed as a way to control underfitting/overfitting – Regularization parameter

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \frac{1}{2} |\mathbf{w}|^2 - \sum \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$ is maximized

$$\begin{aligned}\partial Q / \partial w &= 0 \Rightarrow \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \\ \partial Q / \partial b &= 0 \Rightarrow \sum \alpha_i y_i = 0\end{aligned}$$

Linear SVMs: Summary

- The classifier is a *separating hyperplane*.
- The most “important” training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers α_i .
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u} + b$$

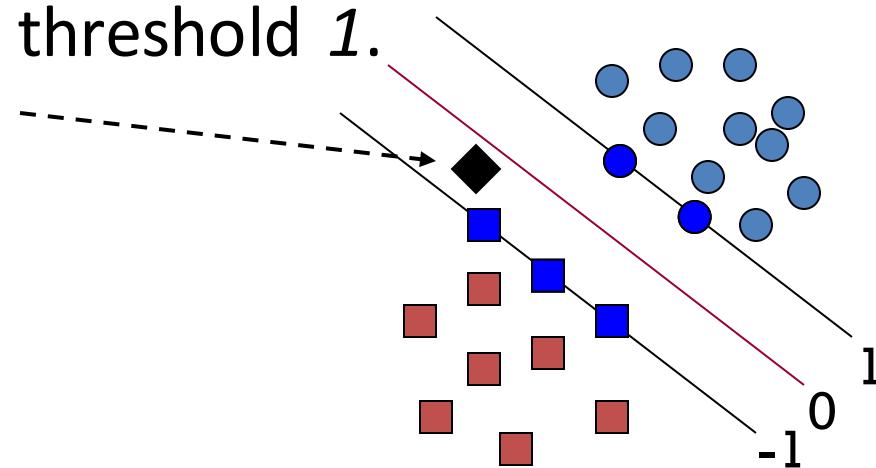
Classification with SVMs

- Given a new point \mathbf{u} , we can score its projection onto the hyperplane normal:
 - i.e., compute score: $\mathbf{w} \cdot \mathbf{u} + b = \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u} + b$
 - Decide class based on whether $<$ or $>$ 1
 - Can set confidence threshold 1.

Score > 1 yes

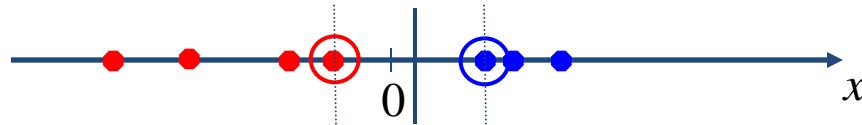
Score < -1 no

Else: don't know

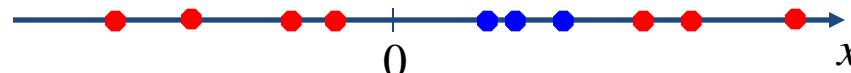


Non-linear SVMs

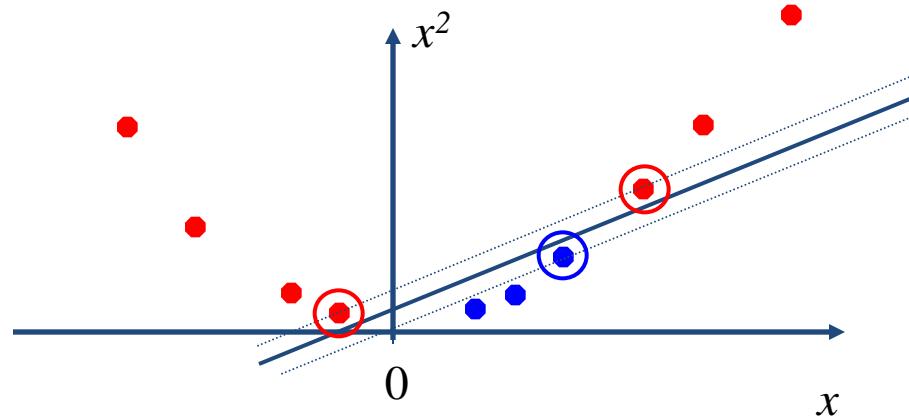
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

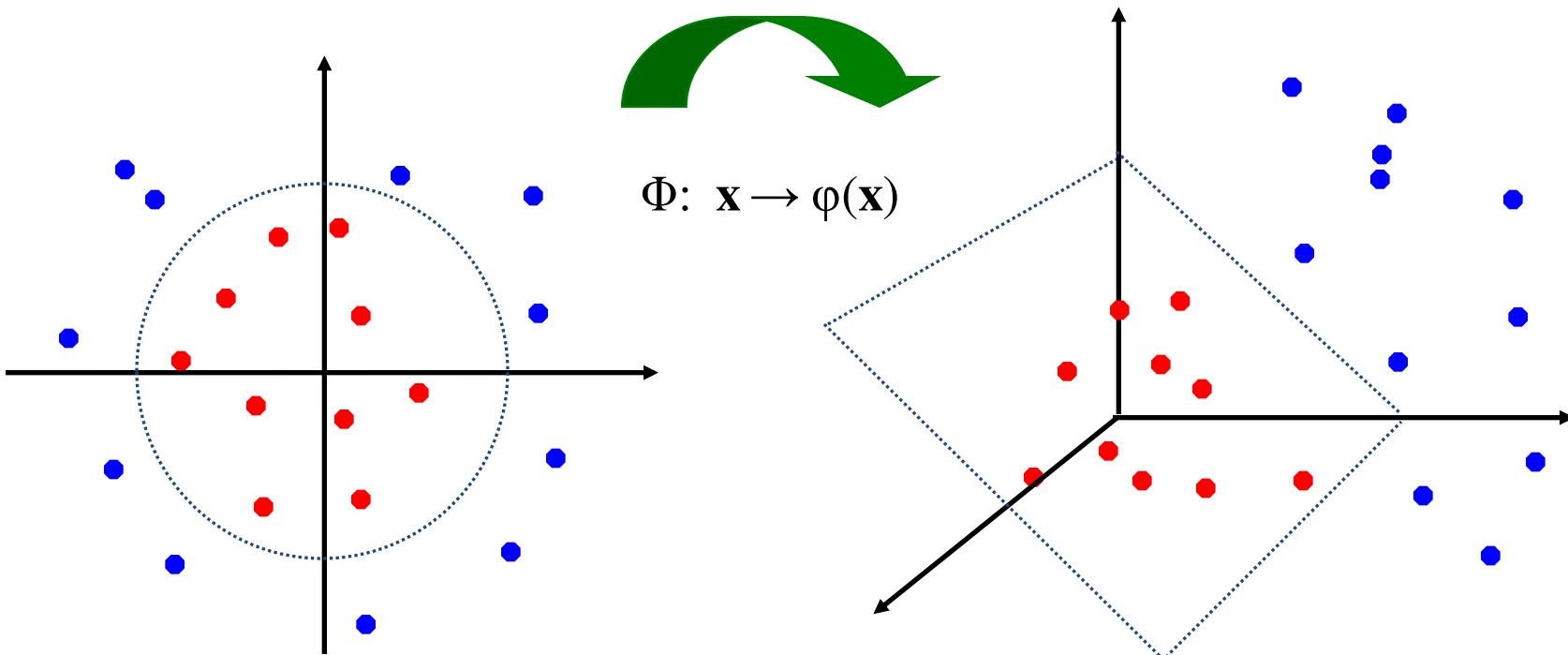


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on dot product between vectors
 $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the dot product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

Kernel function

- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$

- Mercer's theorem

Kernels

- Why use kernels?
 - Make non-separable problem separable.
 - Map data into better representational space

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p$

- Gaussian (radial-basis function network):

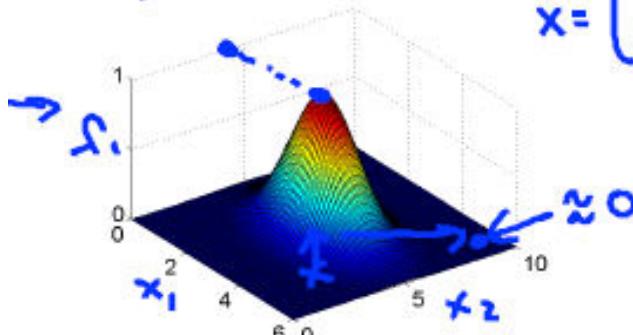
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i \cdot \mathbf{x}_j + \beta_1)$
- String
- *chi Square*

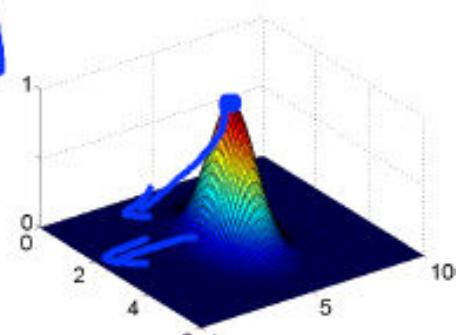
Example:

$$\rightarrow l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$$

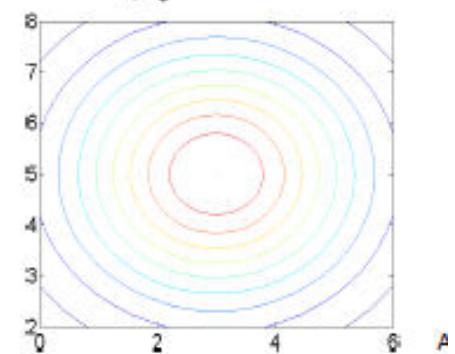
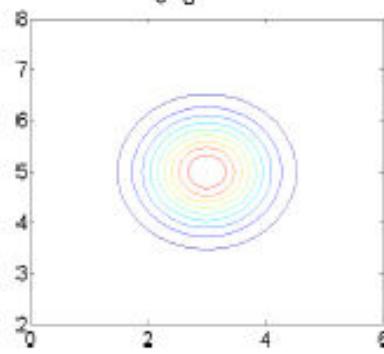
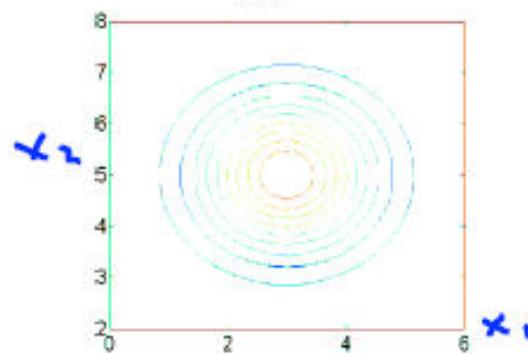
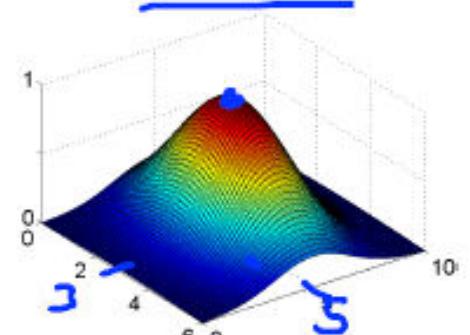
$$\rightarrow \sigma^2 = 1$$



$$x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad \sigma^2 = 0.5$$



$$\sigma^2 = 3$$



TO DO: Classification using Kernel methods

SVM Parameters

- C – higher C – Lower bias/High Variance
 - Overfitting
- C – smaller C – High bias/Low Variance
 - Underfitting
- σ^2 – Large- f varies smoothly – Higher bias/Low variance
 - Underfitting
- σ^2 – Small - f varies sharply – Lower bias/High variance
 - Overfitting

Note: C has the opposite behavior to λ in linear/logistic regression

Properties of SVM

- Non-parameterized model (Parameterized vs non-parameterized)
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Ability to use kernel functions to separate non linearly separable classes
- Can work with large no. of features

Some Issues

- Choice of kernel
 - Gaussian or polynomial kernel is default
 - if ineffective, more elaborate kernels are needed
 - domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
 - e.g. σ in Gaussian kernel
 - σ is the distance between closest points with different classifications
 - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- Optimization criterion – Hard margin v.s. Soft margin
 - a lengthy series of experiments in which various parameters are tested

SVM Applications

- **SVM has been used successfully in many real-world problems**
 - text (and hypertext) categorization
 - image classification
 - bioinformatics (Protein classification, Cancer classification)
 - hand-written character recognition
- **Could work well even when your training set is relatively small**

Multi-class classification

- One-vs-All
- Built into many libraries

SVR (support vector regression)

- Using the same idea for regression
- The optimization function now tries to fit the support vectors, instead of trying to maximize the margin between positive and negative support vectors

Classification using Decision Trees

Muditha Tissera

Learning Outcomes

- **LO3:** Gain an understanding on the supervised learning techniques.

Session Outcome

After completing this session you will be able to;

Explain

Decision trees and their usage

Decision tree growing with splitting

Decision tree pruning

Rule extraction from decision tree

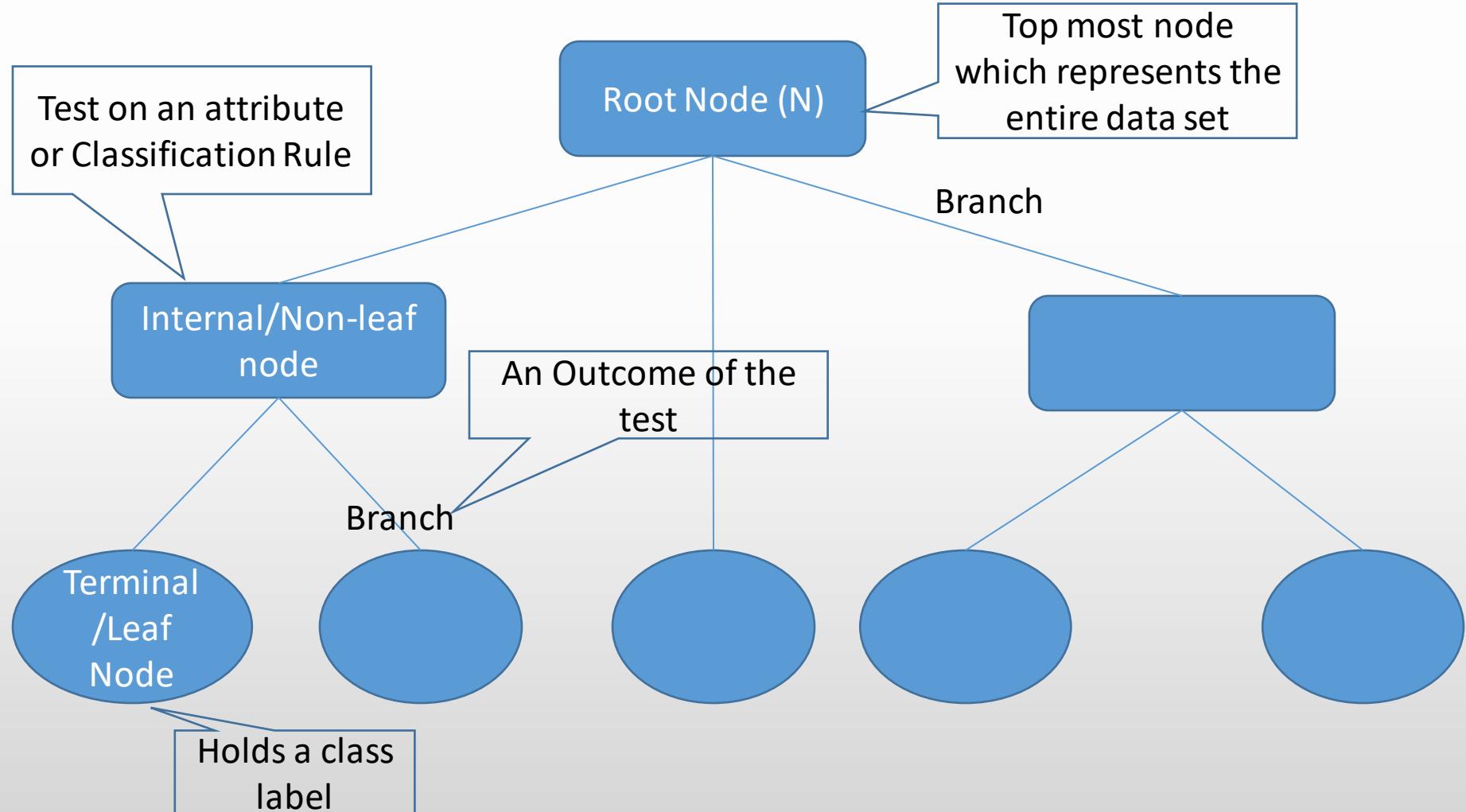
Random forests



What is a Decision Tree?

- A predictive model used in Machine Learning, Statistics and Data mining.
- A tree structure (Directed, acyclic graph) starting with some observations about an item (Data) to conclusions (target value).
- Makes a prediction on the basis of a series of decisions along the branch of a tree.
- Representing possible paths.
- Used for classification and regression.

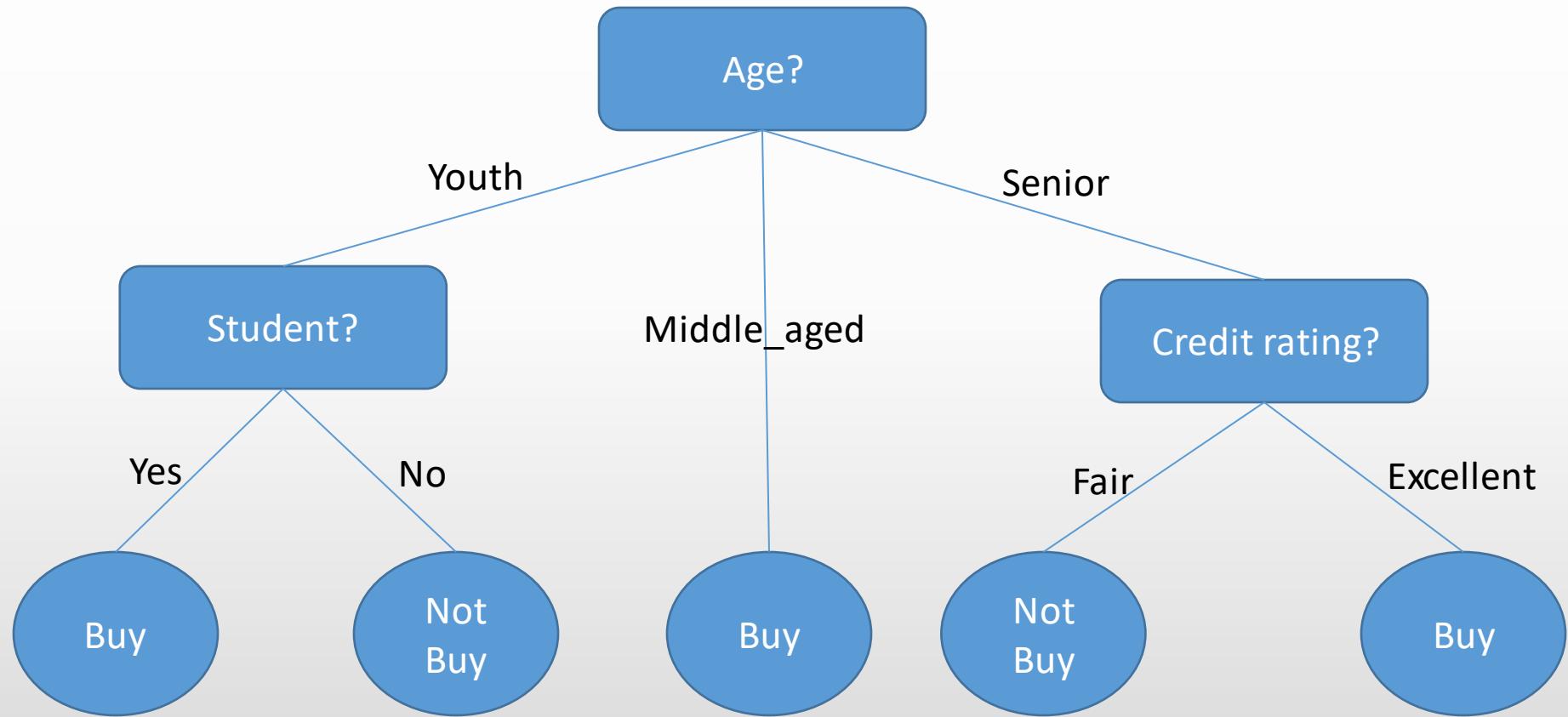
Decision tree structure



Class labeled training samples from a customer database:

#	Age	Income	Student	Credit_rating	Class:buys_computer
1	youth	High	no	fair	no
2	youth	High	no	excellent	no
3	Middle-aged	High	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	Middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	Middle-aged	medium	no	excellent	yes
13	Middle-aged	High	yes	fair	yes
14	senior	medium	no	excellent	no

Example: Concept of Buying a computer



Types of decision trees

Univariate
Vs.
Multivariate decision trees

Binary
Vs.
n-ary decision trees

Univariate Vs. Multivariate decision trees:

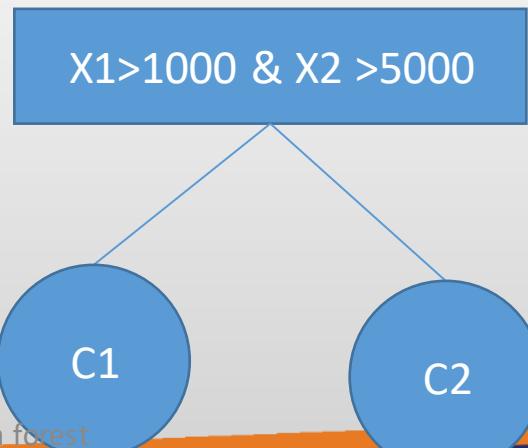
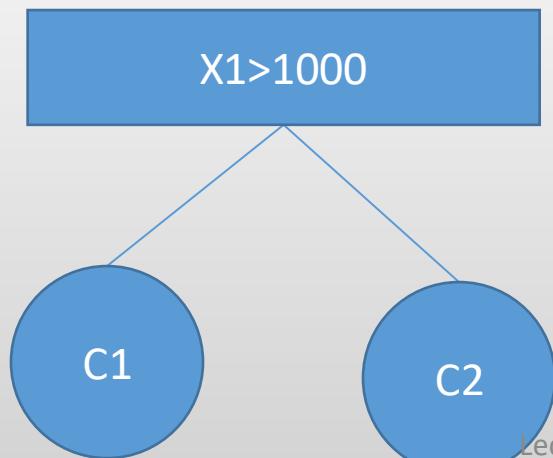
Restrict the number of variables/attributes used per split.

Univariate:

- Each split at a non-terminal node
- involves test on a single attribute.

Multivariate:

- Each split at a non-terminal node
- involves test on multiple attributes.
- [Conjunctions of univariate Splits]

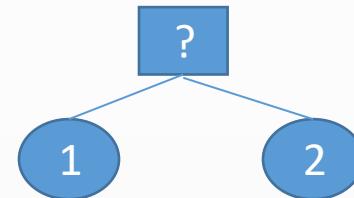


Binary Vs. n-ary decision trees:

Restrict the number of children per node.

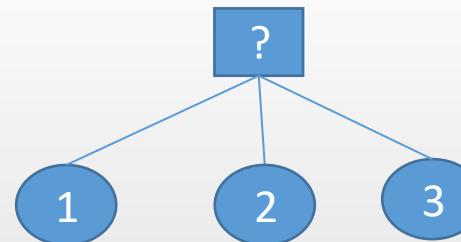
Binary (2-ary)

Split has **2** branches



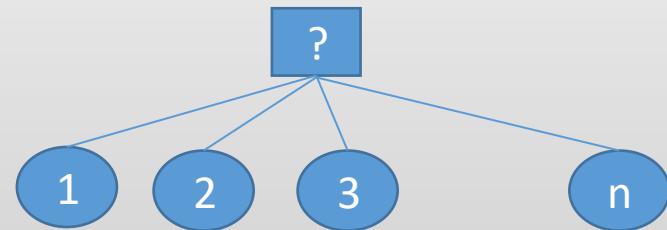
Ternary (3-ary)

Split has **3** branches



N-ary

Split has **n** branches



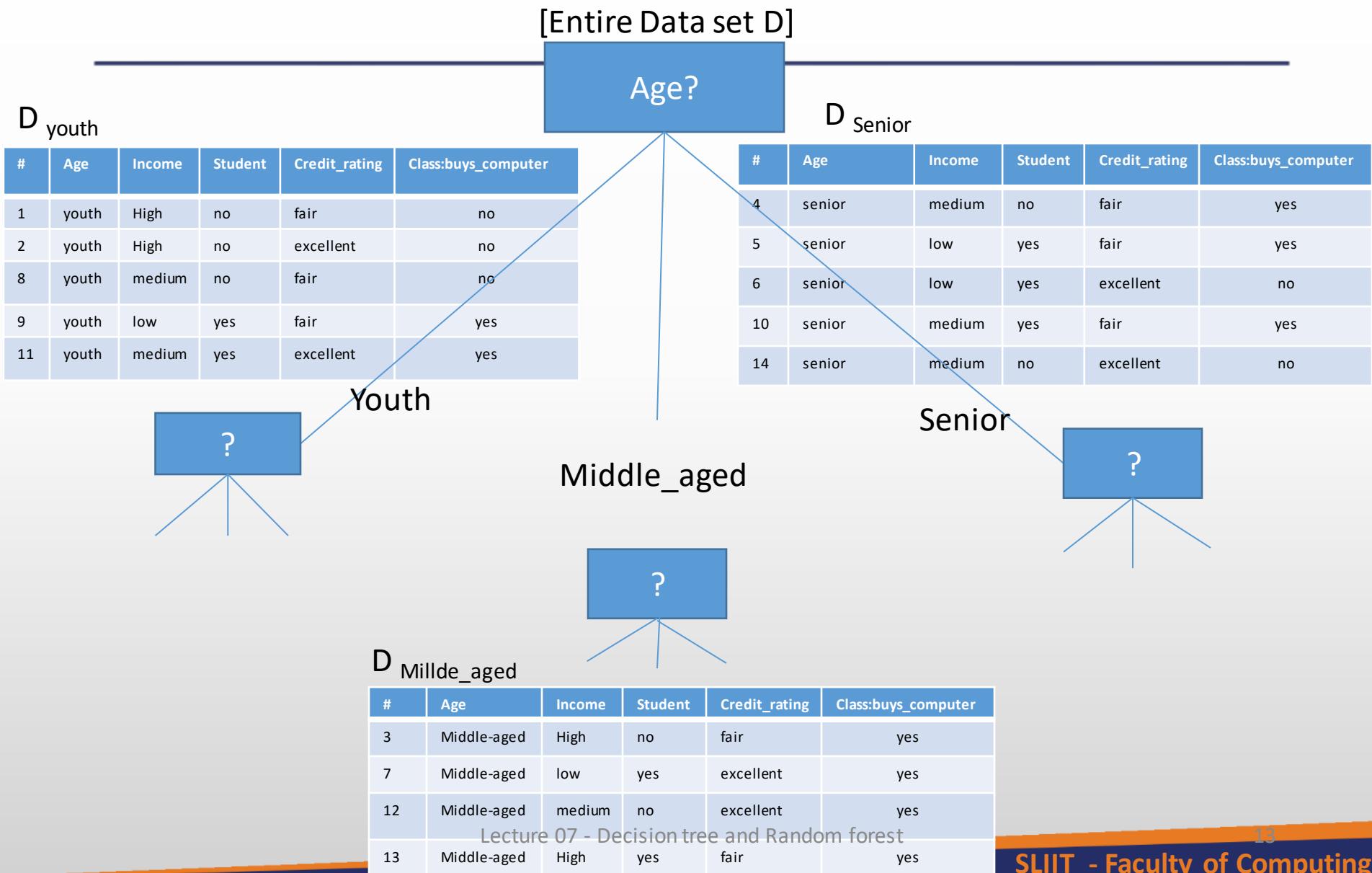
Decision Tree Learning/Induction

Construction of a decision tree from class labeled training samples.

Decision Tree Learning/Induction

- A top-down approach is common.
- Start with a training set of samples with associated class labels.
- Recursively split the training set into smaller subsets as the tree is being built.
- Splitting is done based on a test (splitting rule) on an attribute/variable of the dataset.

Splitting attribute



Attribute selection measures

Determines the most appropriate attribute to partition/split samples at each node.

Methods available:

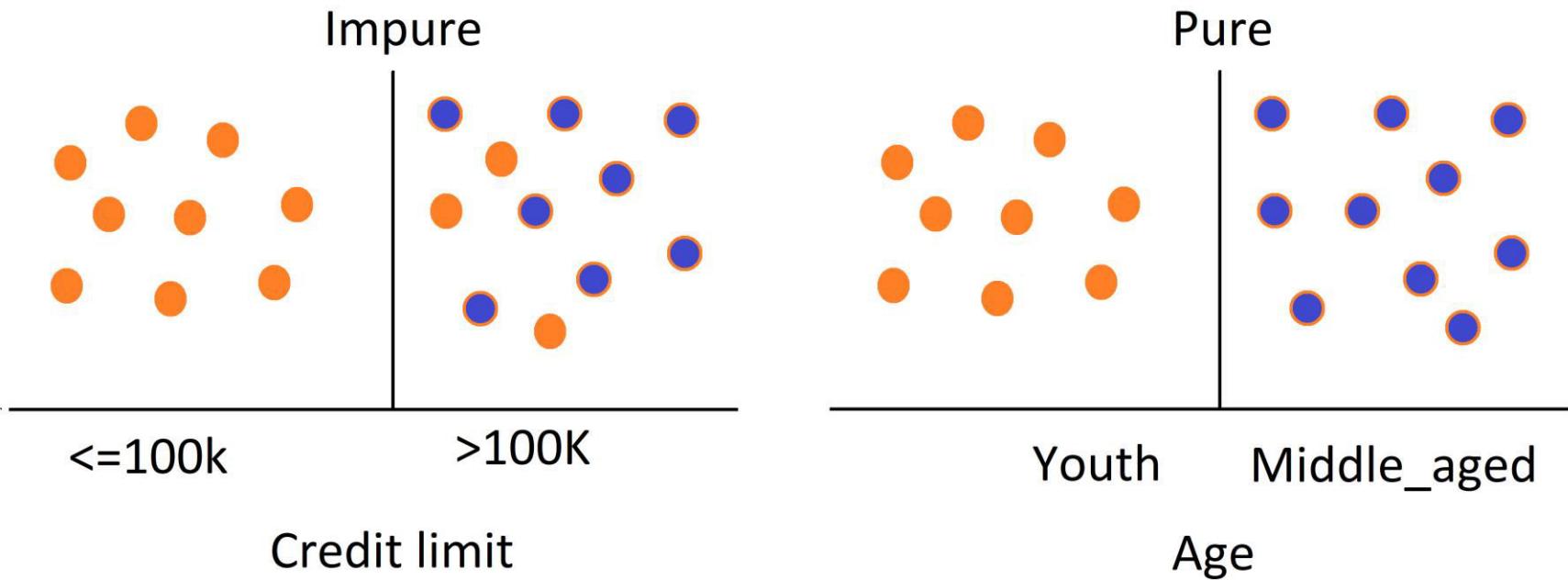
Information Gain (used in ID3 algorithm)

Gain Ratio (used in C4.5 algorithm)

Gini Index (used in CART algorithm)

Entropy

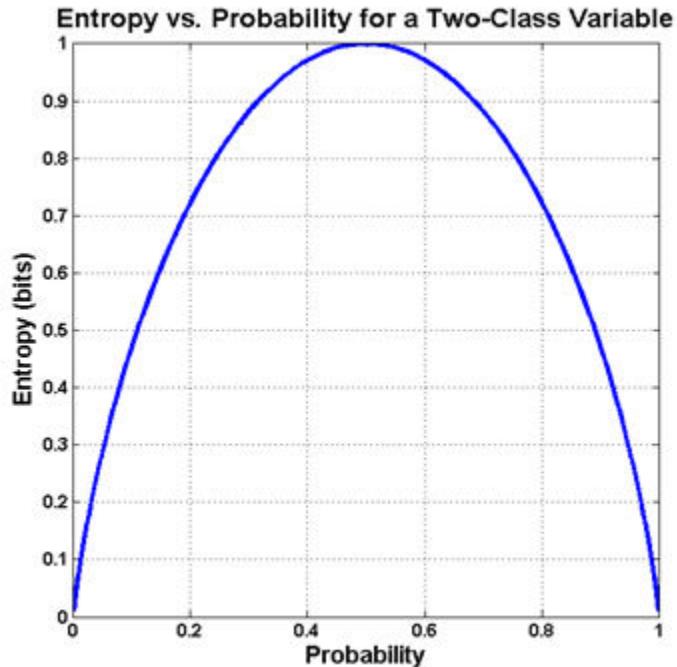
Which test is more information to correctly classify samples? Credit limit or Age?
(Buys a computer :Yes or No)



$$\text{Entropy/Info}(D) = \sum_{i=1}^m -p_i \log_2 (p_i)$$

D= Data set/ partition (training set of class labeled samples)

p_i = Probability that an arbitrary sample in the data set belongs to class i.
(ratios of elements of each label in the set)



$$P_i = |C_i, D| / |D|$$

Class label attribute has m distinct values defining m distinct classes. Ex. Buys_computer has 2 (m) values. (yes, no)

$|C_i, D|$ = number of samples in class C_i in D .

$|D|$ = number of samples in set/partition D

Information Gain

Information Gain is calculated using the entropy.

Entropy of a partition:

Measures the level of **impurity** in a partition data.

The expected information needed to classify a given partition(D) with minimum impurity is given by

Entropy OR Info(D).

Calculate Information Gain

$$\text{Information gain}_{(\text{attribute})} = \text{Entropy}_{(\text{Parent})} - (\text{weighted}) \text{ Average Entropy}_{(\text{Children})}$$

- Information gain is the difference between the amount of information that is needed to correctly make a prediction before (ie. Parent Entropy) and after (ie. Average entropy of children)a split is made.
- Gain is the reduction of entropy due to the splitting on the attribute A.

$$\text{Entropy/Info}(D) = \sum_{i=1}^m -pi \log_2 (pi)$$

calculate Entropy of a Node :

Example (**Parent**): class label attribute “*Buys_computer*”

D = entire data set of 14 records

m=2 {class C₁ (9 records)=“yes”, class C₂(5 records)=“no”}

Expected information to classify a sample in D (Info(D))

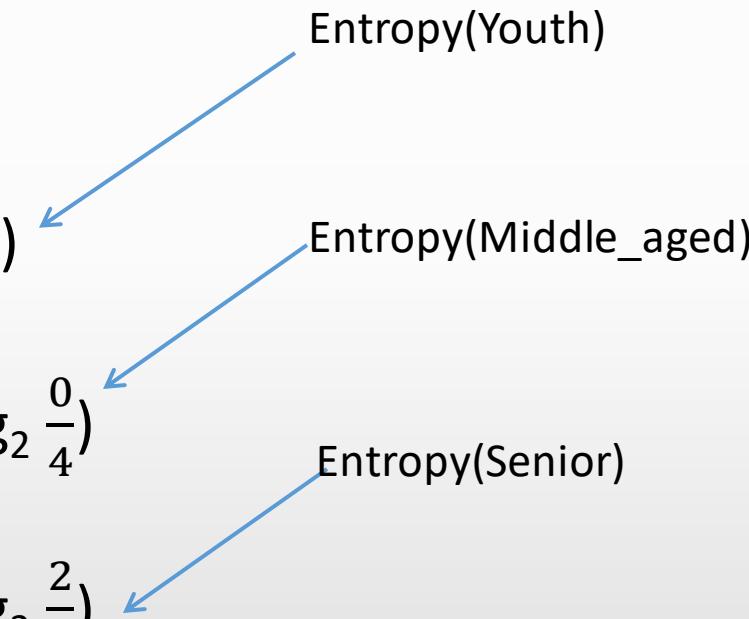
$$\text{Info}(D) / \text{Entropy} = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940 \text{ bits.}$$

Entropy = 0.940 bits.

(Weighted) Average Entropy (Children)

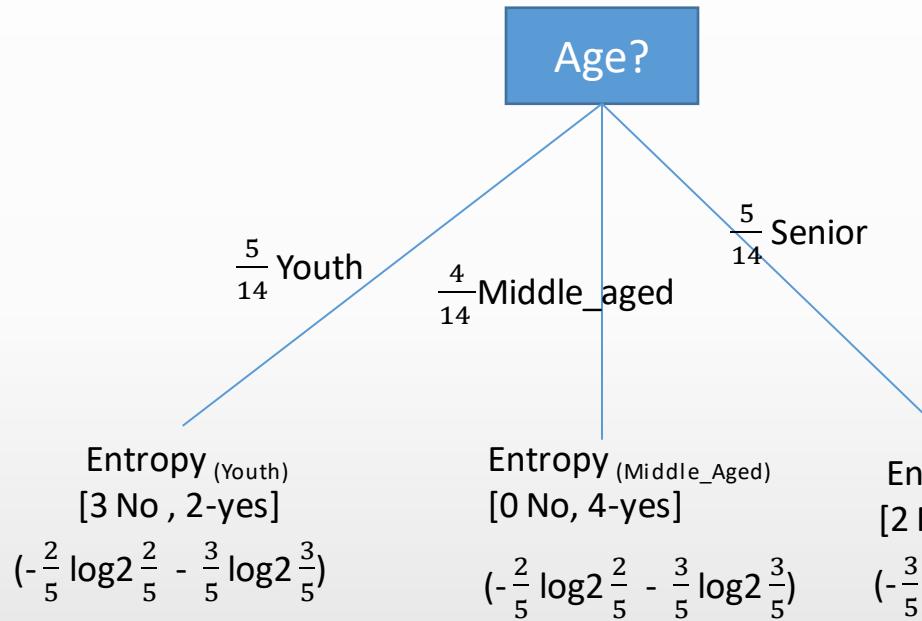
$$\text{Info}_{(A)}(D) = \sum_{j=1}^m \frac{|D_j|}{|D|} * \text{Info}(D_j)$$

Ex:

$$\begin{aligned}\text{Info}_{(\text{Age})}(D) &= \frac{5}{14} * \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) \\ &\quad + \frac{4}{14} * \left(-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) \\ &\quad + \frac{5}{14} * \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right)\end{aligned}$$


$$= 0.694 \text{ bits.}$$

Calculate Information Gain



$$\text{Entropy}(D) / \text{Info}(D) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940 \text{ bits}$$

$$\begin{aligned}
 \text{Info}_{(\text{Age})}(D) &= \frac{5}{14} * \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) \\
 &\quad + \frac{4}{14} * \left(-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) \\
 &\quad + \frac{5}{14} * \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) \\
 &= 0.694 \text{ bits.}
 \end{aligned}$$

(Weighted) Average Entropy (Children)

Information Gain_(Age)

$$\begin{aligned}
 &= \text{Info}_{(\text{Buys_computer})} - \text{Info}_{(\text{Age})} \\
 &= 0.940 - 0.694 \\
 &= 0.246 \text{ bits}
 \end{aligned}$$

How to select the splitting attribute for a node using Information gain

1) Compute Entropy of the parent node

$$\text{Info}_{(\text{Buys_computer})} = 0.940$$

2) Compute Information gain of each attribute.

(Ex: Age, income, student, credit rating)

$$\begin{aligned}\text{Gain}_{(\text{Age})} &= \text{Info}_{(\text{Buys_computer})} - \text{Info}_{(\text{Age})} \\ &= 0.940 - 0.694 \\ &= 0.246 \text{ bits}\end{aligned}$$

$$\text{Gain}_{(\text{Income})} = \text{Info}_{(\text{Buys_computer})} - \text{Info}_{(\text{Income})} = 0.029 \text{ bits}$$

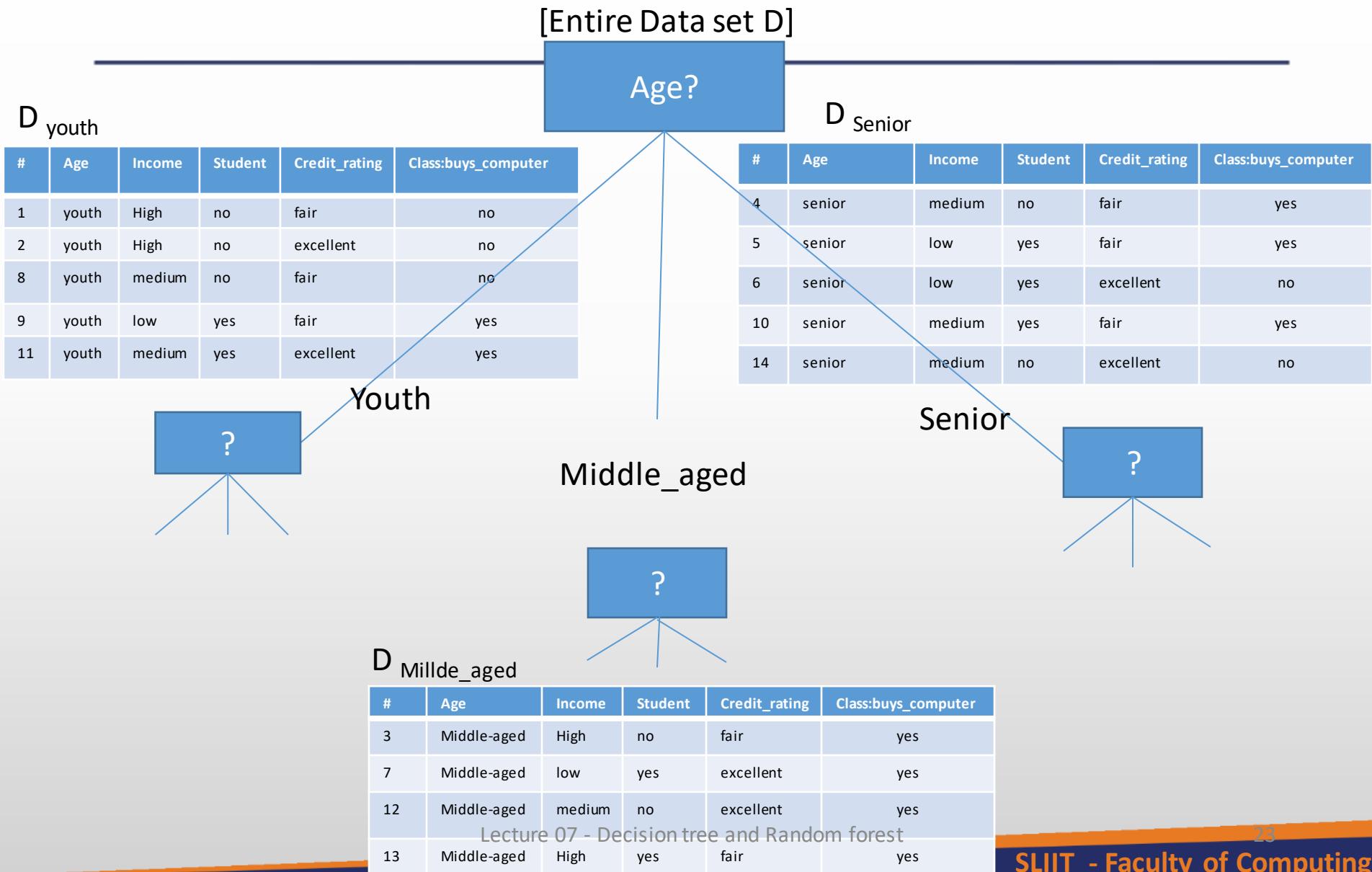
$$\text{Gain}_{(\text{student})} = \text{Info}_{(\text{Buys_computer})} - \text{Info}_{(\text{student})} = 0.151 \text{ bits}$$

$$\text{Gain}_{(\text{Credit_rating})} = \text{Info}_{(\text{Buys_computer})} - \text{Info}_{(\text{Credit_rating})} = 0.048 \text{ bits}$$

3) The attribute with the highest information gain is selected as the splitting attribute.

Ex: Age has the highest information gain. So that it is selected as the splitting attribute at Node N (root node).

Splitting attribute



Generate_Decision_Tree Algorithm

Input:

1) Data partition D

a set of training samples and their associated class labels.

2) Attribute list

the set of candidate attributes

3) Attribute_selection_method

a procedure to determine the splitting criterion that “**best**” partitions the data samples into individual classes.

This criterion consists of a **splitting_attribute** and possibly either a **split point** or **splitting subset**.

Output: A decision tree

Generate_Decimal_Tree

Create node N;

if samples in D are all of the same class C **then**,

return N as a leaf node labeled with the class C

if attribute_list is empty **then**

return N as a leaf node labeled with the majority class in D ;

Apply **Attribute_Selection_Method**(D , Attribute_list) to find the best *splitting criterion*;

Label node N with *splitting criterion*;

If *splitting_attribute* is discrete-valued and multiway splits allowed **then** //not restricted to binary tree

attribute_list <- *attribute_list* - *splitting criterion* //remove *splitting attribute*

for each outcome j of *splitting criterion* //partition the samples and grow subtrees for each partition.

let D_j be the set of data samples in D satisfying outcome j ; // a partition.

if D_j is empty **then**

attach a leaf labeled with the majority class in D to node N ;

else

attach the node returned by **Generate_Decimal_Tree**(D_j , Attribute_list) to node N

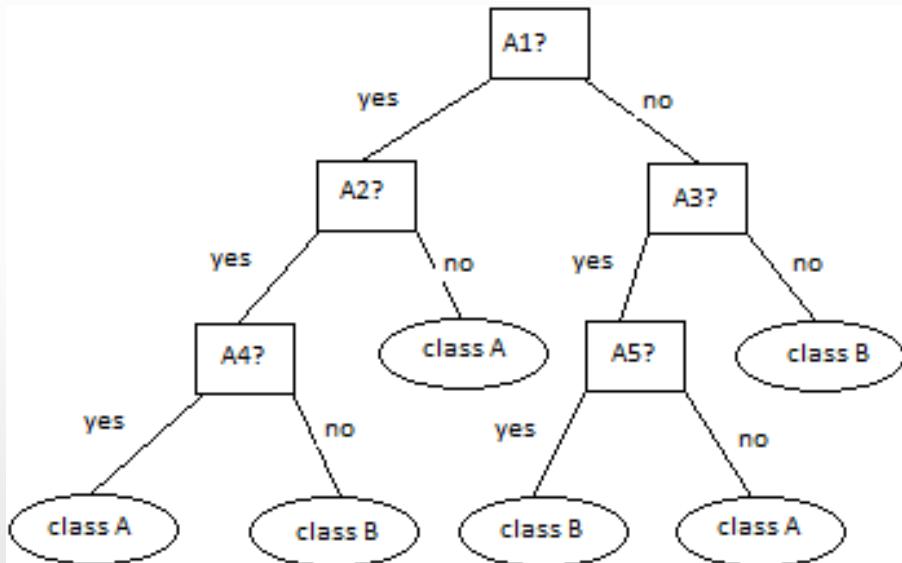
end for

return N ;

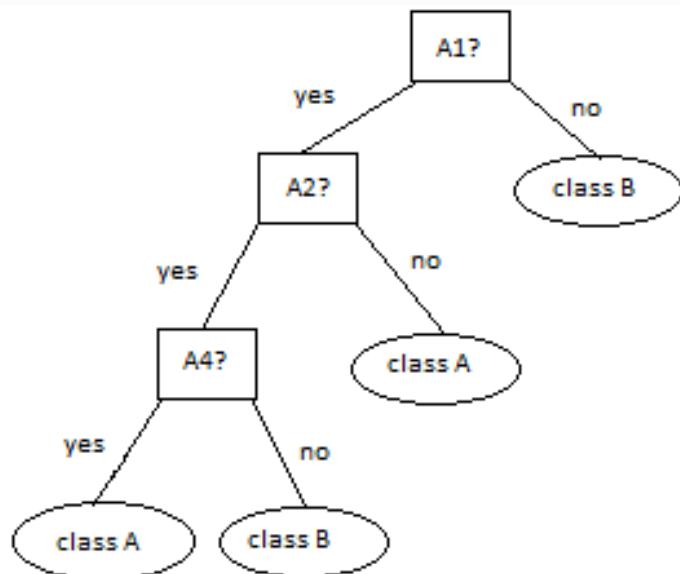
Decision Tree Pruning

- Technique to reduce the size (complexity) by removing least-reliable branches.
- Why pruning?
 - Address the problem of overfitting the data;
 - Improves the accuracy.
 - Tree becomes smaller and less complex;
 - Easier to comprehend.

Decision Tree Pruning Cont.



Unpruned tree



Pruned tree

Decision Tree Pruning Cont.

- Can prevent overfitting
- Two approaches:
 1. Pre-pruning
 1. Post-pruning (backward pruning)

Pre-pruning

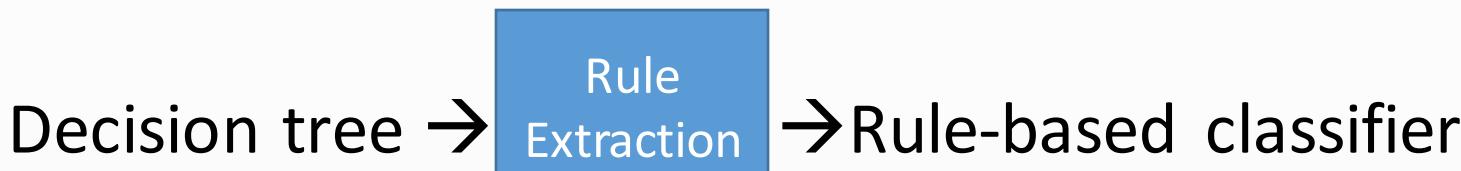
- Tree is pruned by halting its construction early.
- Upon halting node becomes a leaf.
- Assess the goodness of a split using measurement such as;
 - Statistical significance
 - Information gain
 - Gini Index and so on.
- If measurement of the split < threshold value
 - Further splitting of the subtree is halted.

Post-pruning

- Common approach.
- Remove a subset of a given tree in a fully-grown tree and replace with a leaf node.
- Leaf := Most frequent class among subtree. Ex: A3 in the diagram
- Post-pruning techniques:
 - Cost based pruning
 - Reduced error pruning
 - Maximum error pruning
 - Error complexity pruning

Rule extraction from Decision Tree

- The more the decision tree is large the more the complexity and difficult to comprehend.



Rule-based classifiers uses set of IF-THEN rules.

IF <i>condition</i>	THEN <i>conclusion</i>
<p><i>Rule antecedent(precondition)</i></p> <p>one or more attributes that are Logically ANDed.</p> <p>IF Age = youth AND Student = yes</p>	<p><i>rule consequent</i></p> <p><i>Class prediction</i></p> <p>THEN buys computer = yes</p>

Rule extraction from Decision Tree Cont.

Method:

- One rule is created from each path from root to a leaf.
- Splitting criterion at each node from root to one before leaf(except leaf) along the path are joined with logical **AND** to form an antecedent (IF part)
- Leaf node class label prediction form the rule consequent (THEN part).

Rule #	Rule antecedent	Rule consequent
R1	IF age =youth AND student =no	THEN buys_computer = no
R2	IF age =youth AND student =yes	THEN buys_computer = yes
R3	IF age =middle_aged	THEN buys_computer = yes
R4	IF age =senior AND credit_rating= excellent	THEN buys_computer = yes
R5	IF age =senior AND credit_rating=fair	THEN buys_computer = yes

Characteristics of Decision Tree Rules

- A disjunction (logical OR) is implied between each of the extracted rules.
- Mutually exclusive.
 - No rule conflicts because no two rules will be triggered for the same sample.
- Exhaustive.
 - One rule for possible attribute-value combination.
 - So no default rule is required.
- Unordered.

Rule Induction directly from data.

- IF-THEN rules can be extracted directly from the training data. (Without having to generate a decision tree first)

Ex:

Sequential covering algorithm

Associative classification algorithms

Advantages and Disadvantages of Decision Trees

Advantages:

- Construction does not require any domain knowledge or parameter setting.
- Good for exploratory knowledge discovery.
- Can handle high dimensional data.
- Representation of acquired knowledge is easy to assimilate by human. (Intuitive)
- Learning and classification steps of Decision Tree induction are simple and fast.
- High accuracy.(But depend on selected data)
- Used as a basis for many commercial rule induction systems.

Disadvantages:

- Highly depend on the training data. Thus overfitting.
- Trees becomes very complex unless an appropriate thresholds are set as termination conditions.
- Use greedy approach and locally optimal (not globally optimal).

Random Forests

- An ensemble classifier that produces multiple decision trees.
- Use randomly selected subsets of training samples to produce multiple decision trees.
- Each decision tree is independently produced without any pruning.
- Grow the forest up to a user-defined number of trees (k).
- A new unlabeled data input is thus evaluated against all decision trees created in the ensemble.
- This ensemble method increases the accuracy of the output.

Sampling training samples for random forest construction(multiple decision trees)

Bootstrap Method (Sampling with replacement):

Sample the training samples uniformly ***with replacement.***

ie. Each time a sample is selected, it is equally likely to be selected again and included in the next sample training set.

Ex. .632 bootstrap

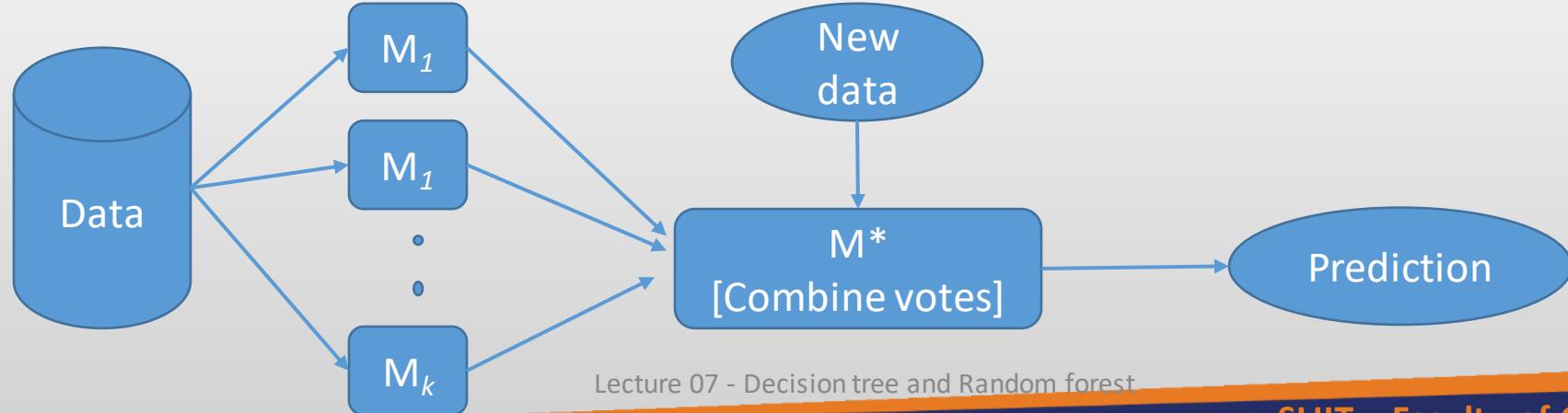
Ensemble methods

Methods that use a combination of Models.

Ex: Multiple decision trees

An ensemble method combines a series of k learned models(classifiers/predictors) with the aim of creating an improved composite model M^*

$$M_1, M_2, M_3, \dots, M_k \rightarrow M^*$$



Ensemble methods

Two techniques available:

1. Bagging (Bootstrap aggregation)

1. Boosting

Bagging

Given a data set D of d samples,

1. Generate k number of sample training sets D_1 to D_k (bootstrap samples) using sampling with replacement.
2. For each sample, a model(classifier) is learned.
3. For a new input, each model returns a prediction, which counts as one vote.
4. Bagged classifier M^* , counts the votes and assign the class label **with the most votes for the new input**.
5. If the models predict continuous values, bagged classifier get the result from each model and **average** them for the final outcome.

M^* has the improved accuracy over a single predictor derived from D.

Algorithm: Bagging

Input:

D , a set of d training samples.

K , the number of models in the ensemble.

A learning scheme (decision tree, backpropogation etc.)

Output:

A composite model M^*

Algorithm: Bagging Cont.

Method:

1. **for** $i=1$ to k **do** // create k models
2. Create bootstrap sample, D_i by sampling D with replacement;
3. Use D_i to derive a model, M_i
4. **end for**

To use the composite model on a sample x :

1. **if** classification **then**
2. Let each of the k models classify X and return the majority vote;
3. **If** regression **then**
4. Let each of the k models predict a value for X and return the average predicted values.

Boosting

- A series of k classifiers are **iteratively** learned.
- After the first classifier is learned, next classifier is learned paying more attention to the **misclassified** samples of the previous iteration.
- Do this iteratively k times and the final boosted classifier M^* , combines the votes of each individual classifier.
- Each classifier's vote is a function of its accuracy.
- Boosting can be extended for the prediction of continuous values.

Adaboost algorithm

1. D data set has d class labeled training samples. $(X_1, y_1), (X_2, y_2), \dots, (X_d, y_d)$
2. Assign each sample an equal weight of $\frac{1}{d}$.
3. A sample D_i of size d is derived from D data set (training samples). Sampling with replacement is used.
4. Classifier M_i is learned from D_i .
5. Test the model using same D_i and calculate the error rate of the model. Proceed only if the error rate is less than 0.5

If the sample is misclassified, $err(X_j) = 1$

Otherwise, $err(X_j) = 0$

Model error
rate

Weight of
the
sample

$$error(M_i) = \sum_j^d w_j * err(X_j)$$

Misclassification
error of sample X_j

Adaboost algorithm Cont.

6. Update weights using the error rate of the classifier M_i

Method of updating the weights:

If a sample is **correctly** classified in round i then,

weight of the sample is multiplied by $\frac{\text{error}(M_i)}{1-\text{error}(M_i)}$

7. Weight of all the samples are normalized. (both correctly and incorrectly classified)

Method of normalizing the weights:

new normalized weight : = sample weight * $\frac{\text{sum of old weights}}{\text{sum of new weights}}$

After normalization, the sum of weights remains same (ie. 1)

How an ensemble predict the class label of a new sample?

- Assign a score/weight to each classifier's vote based on how well the classifier performed.
- The lower the classifier error rate, the greater the accuracy. Therefore, the higher the score/weight for the classifier's vote.

Classifier's error rate  → Accuracy  → Score/weight 

The weight of classifier M_i 's vote is

$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

- Get the class prediction from each classifier with the weight of the model.
- Sum up the weights of each predicted class.
- **Class which has the highest sum is selected.**

Adaboost algorithm

Input:

- D , a set of d class-labeled training samples
- K , the number of rounds (one classifier is generated per round)
- A classification learning scheme

Output:

A composite model

Adaboost algorithm Cont.

Method:

01. Initialize the weight of each sample in D to $1/d$;
02. *for i=1 to k do// for each round*
03. *sample D with replacement according to the sample weights to obtain Di;*
04. *use training set Di to derive a model Mi;*
05. *compute error(Mi), the error rate of Mi*
06. *if error(Mi) >.5 then*
07. *reinitialize the weights to 1/d*
08. *go back to step 3 and try again;*
09. *end if*
10. *for each sample in Di that was correctly classified do*
11. *multiply the weight of the sample by error(Mi)/(1-error(Mi));*
//Update weights.
12. *normalize the weight of each sample;*
13. *end for*

To use the composite model to classify sample X:

1. initialize weight of each class to 0;
2. for i= 1 to k do // for each classifier:
3. $w_i = \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$; //weight of the classifier's vote
4. $C = M_i(X)$; // get class prediction for X from M_i
5. Add w_i to weight for class c
6. end for
7. return the class with the largest weight;

Bagging Vs. Boosting

Boosting:

Focusses on misclassified samples.

High accuracy but

Risks on overfitting the resulted composite model to such data.

Bagging:

Less model overfitting

Bagging and Boosting both have improved accuracy than a single model. But boosting tends to achieve greater accuracy.

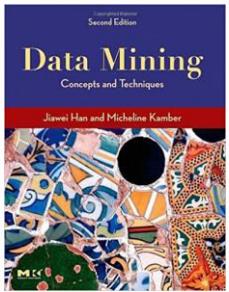
Other ensemble methods

- Boolean operator based ensemble methods
- ML based ensemble methods
- Stack ensemble

Random Forest: Advantages & Disadvantages

- Advantages
 - Works well with small training sets
 - Can be easily parallelized
- Disadvantages
 - Can fail to predict rare outcomes or rare predictions (since it's based on bootstrap sampling)
 - Need to guess the no. of trees
 - Overfitting can occur

Reference:



Data Mining (Concepts and Techniques)
-Second Edition
-Jiawei Han and Micheline Kamber

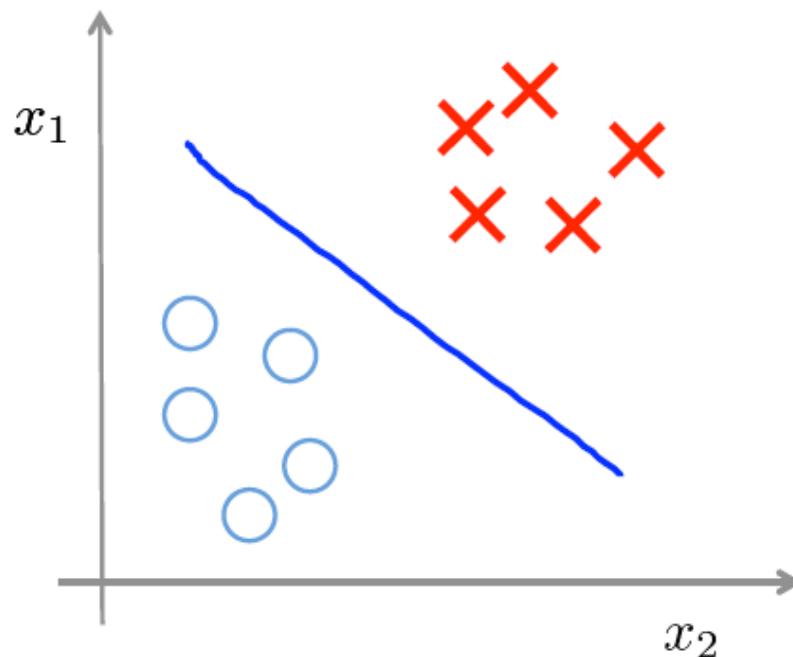
Lecture 8 – Unsupervised learning

Unsupervised learning

- Unlabeled data
- Two main applications
 - Clustering (data mining, recommendation systems)
 - Dimensionality reduction (data compression, visualization)

Supervised learning

Supervised learning

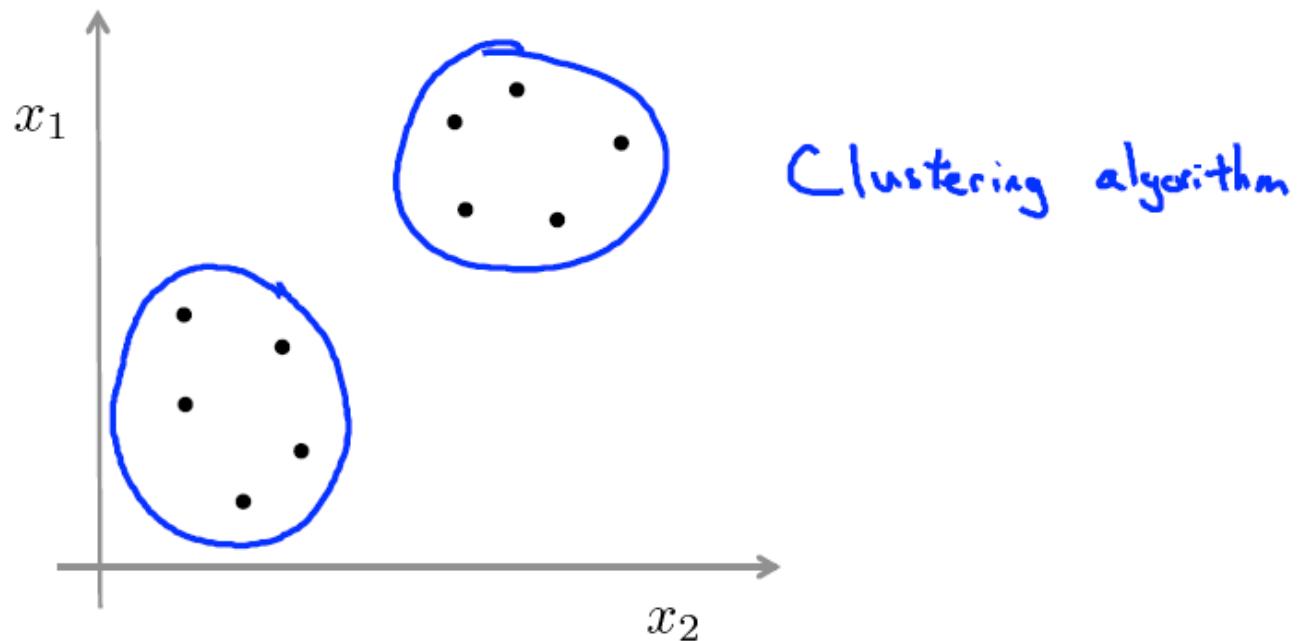


Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})\}$



Unsupervised learning

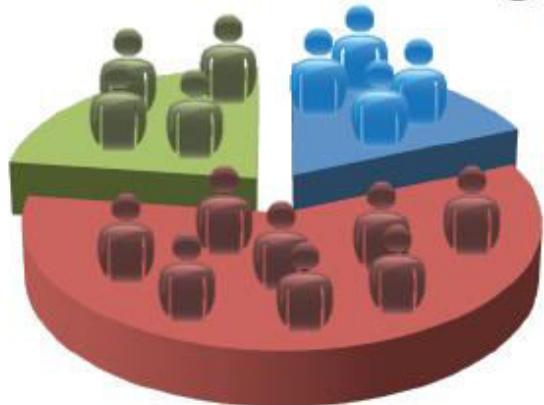
Unsupervised learning



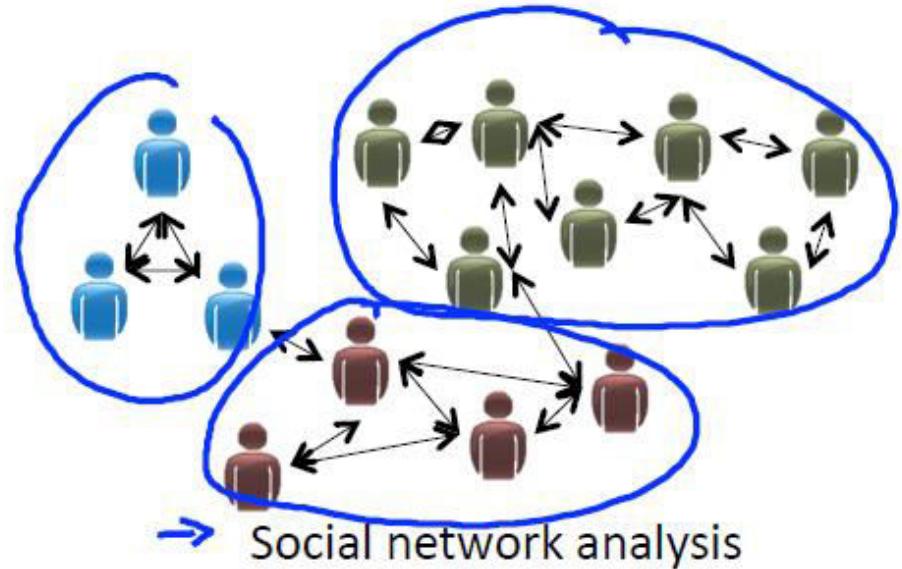
Training set: $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \underline{x}^{(3)}, \dots, \underline{x}^{(m)}\}$ ←

Applications of clustering

Applications of clustering



→ Market segmentation



→ Organize computing clusters

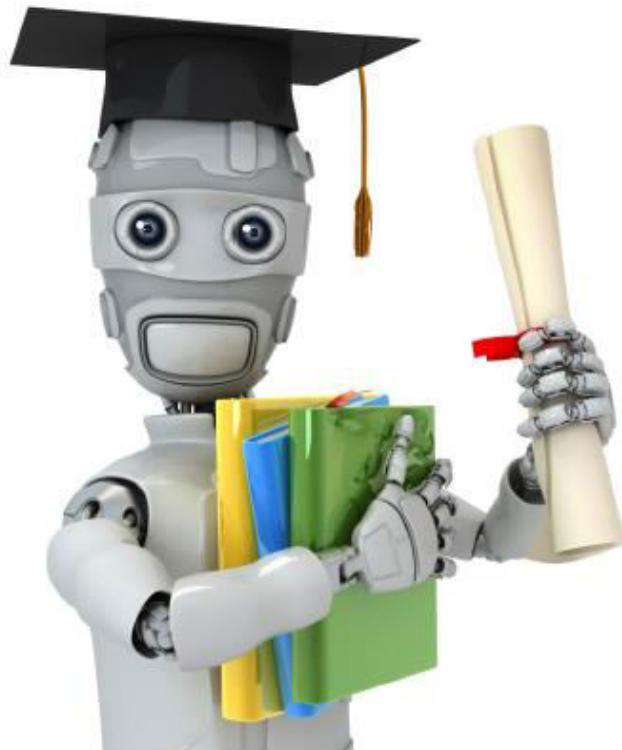


→ Astronomical data analysis

Clustering algorithms

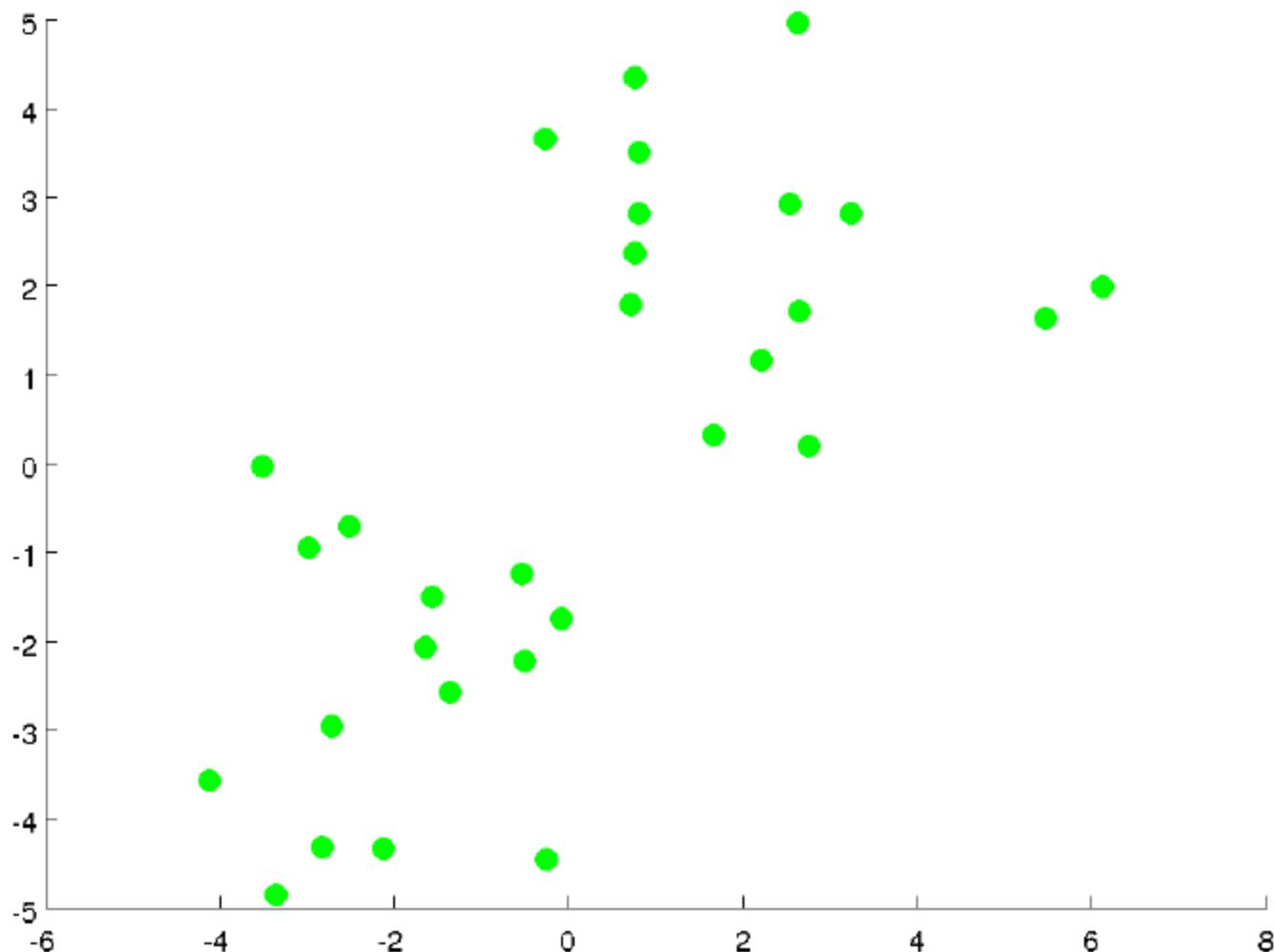
- K-means
- Self Organizing maps
- Auto-encoders/Deep belief networks

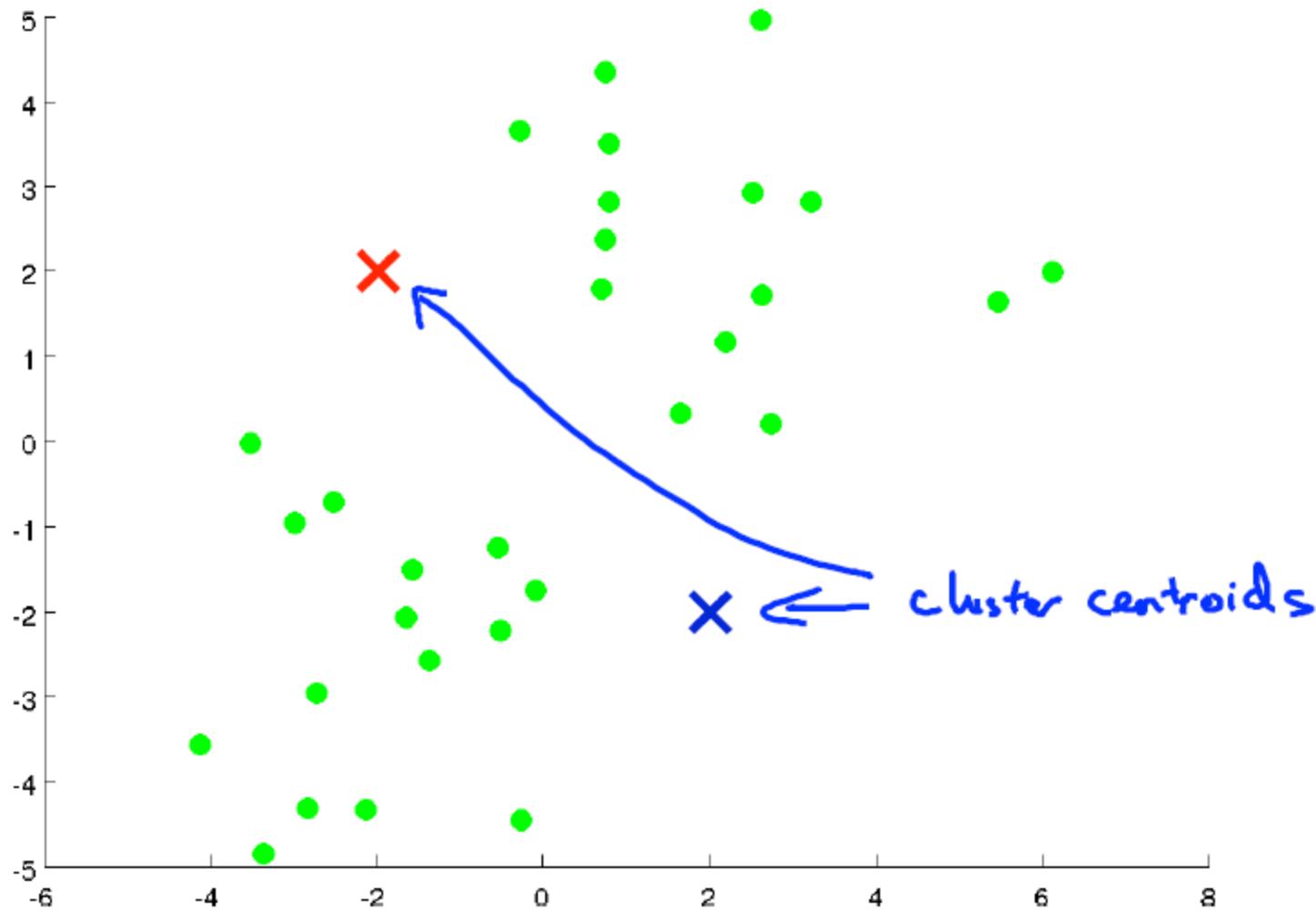
K-means algorithm

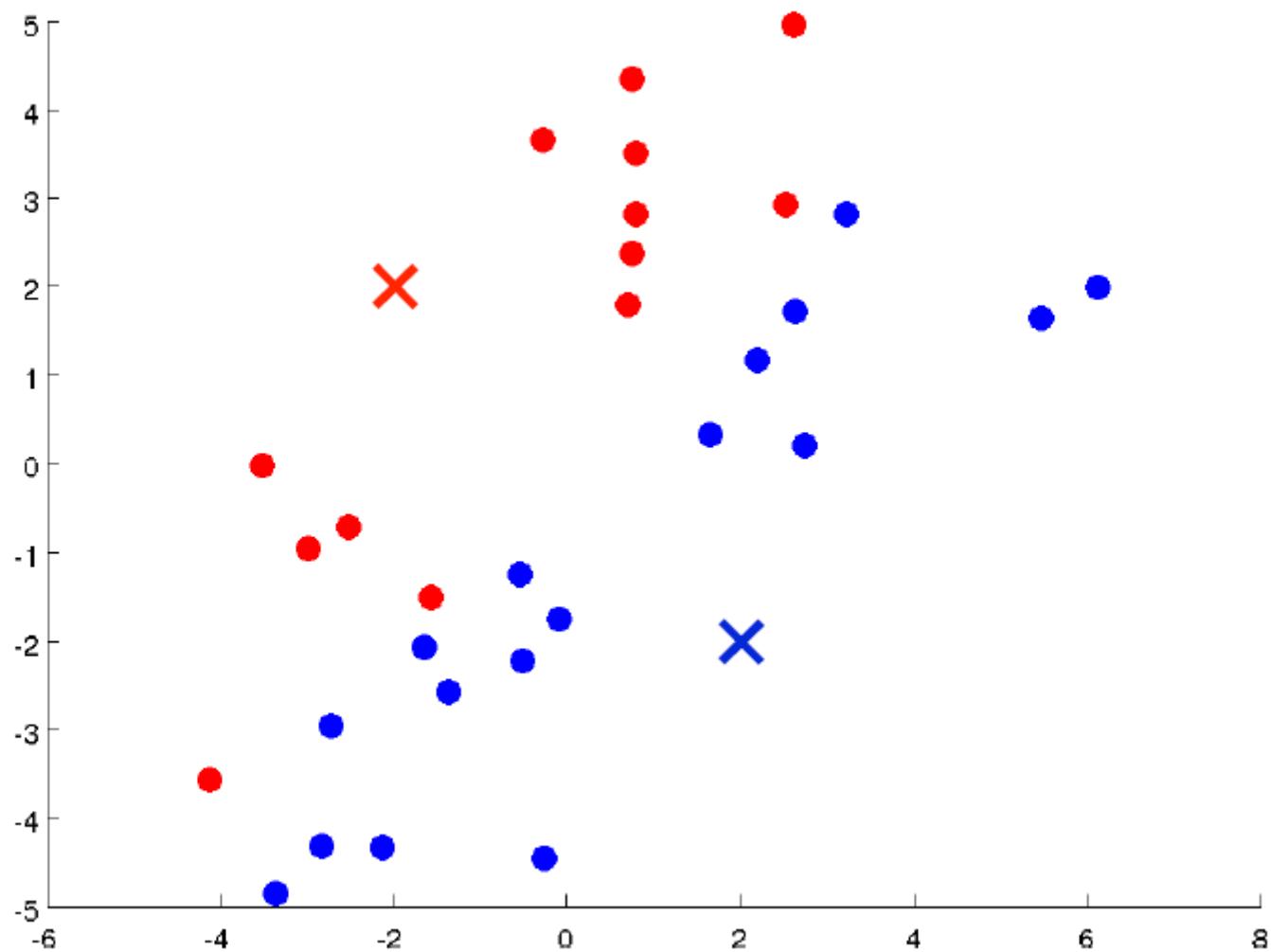


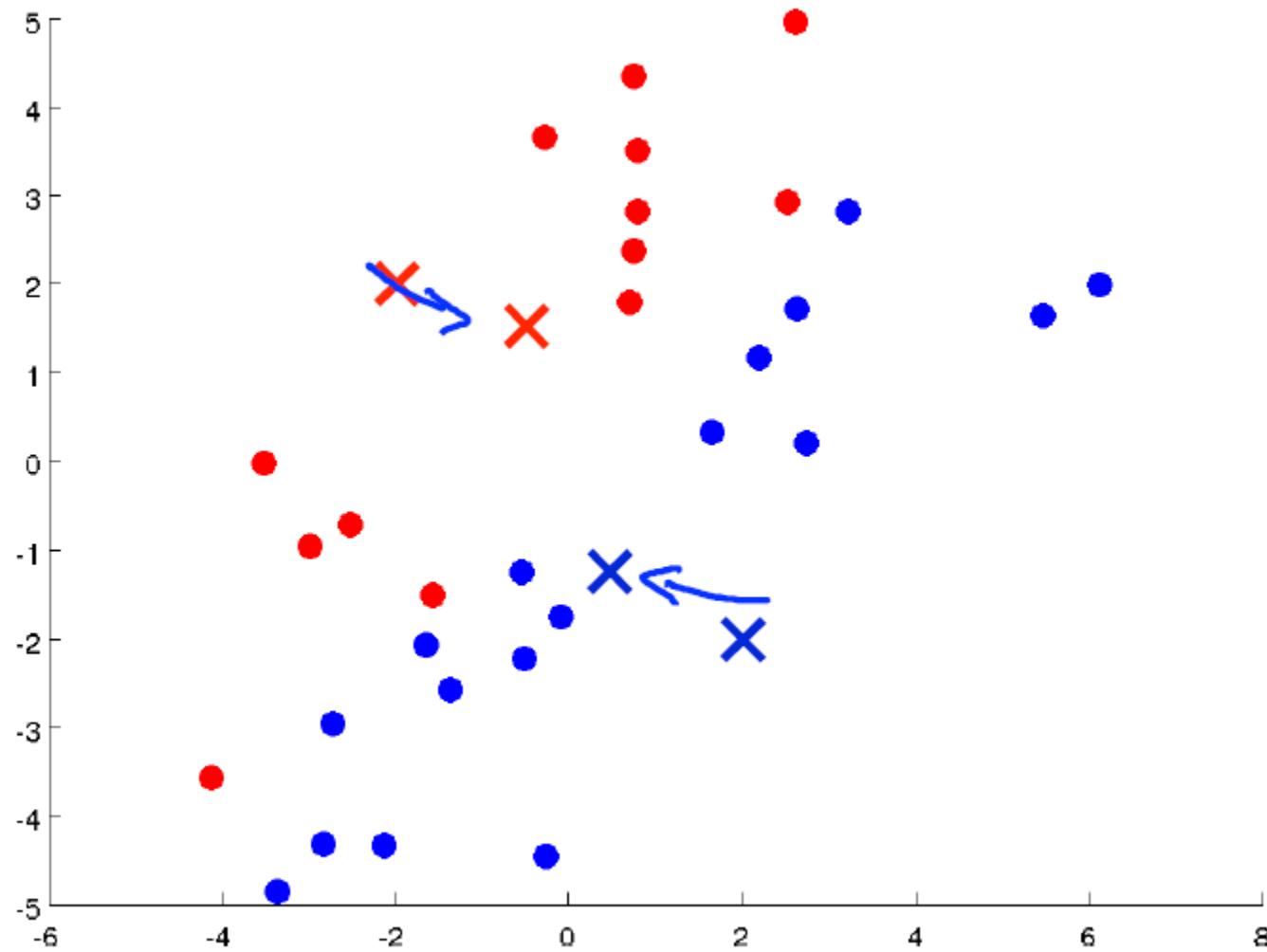
Machine Learning

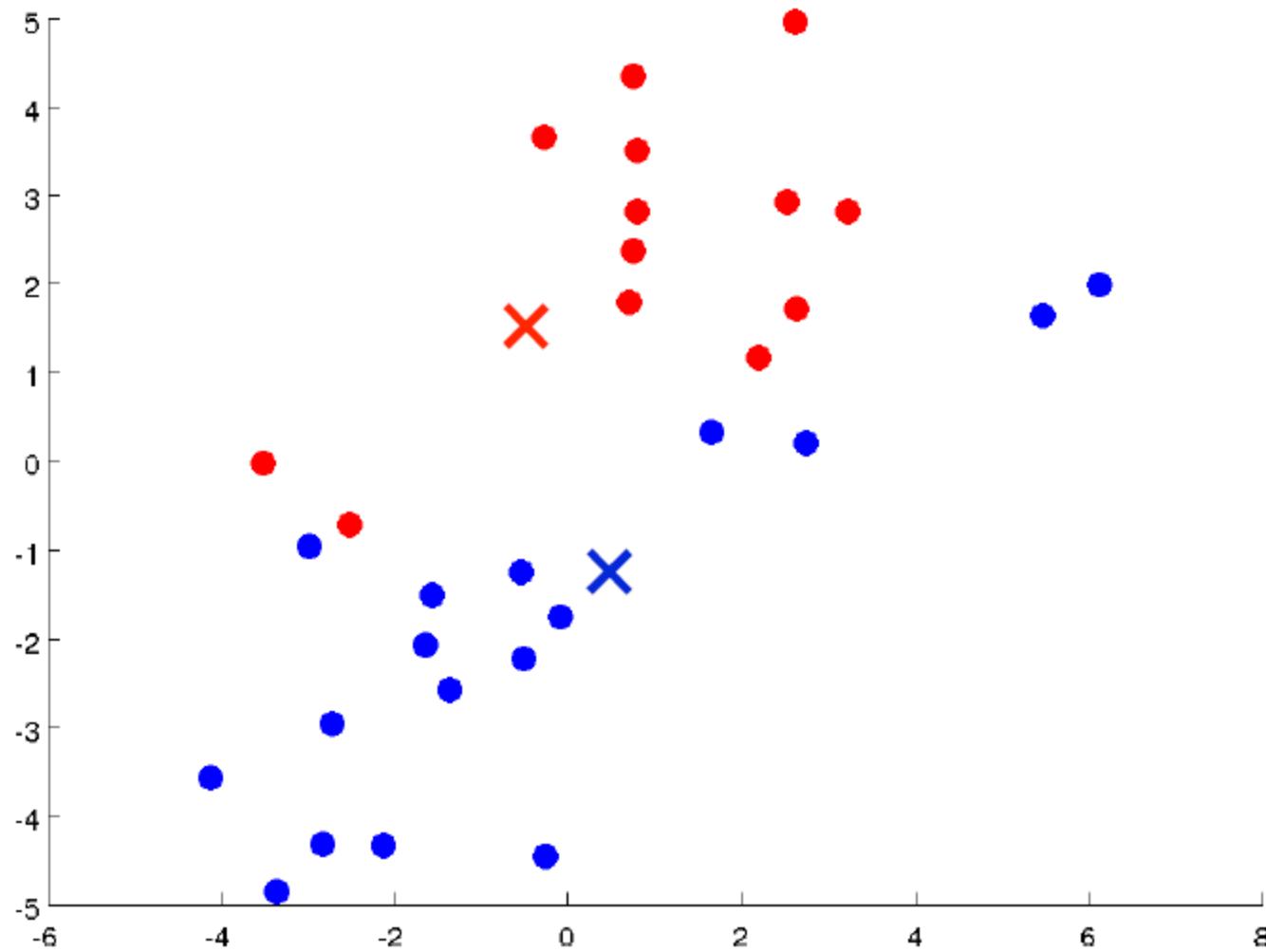
Clustering
K-means
algorithm

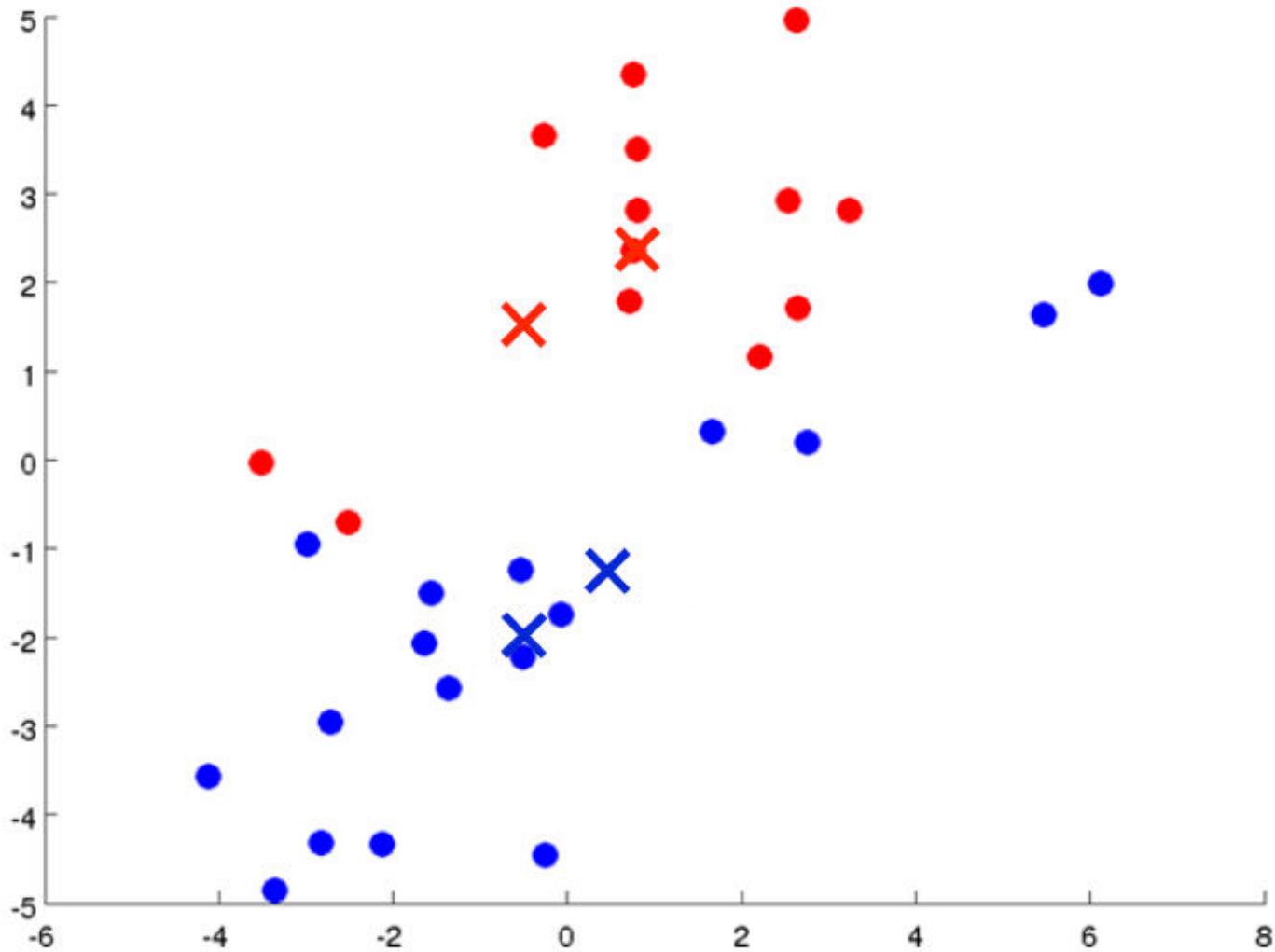


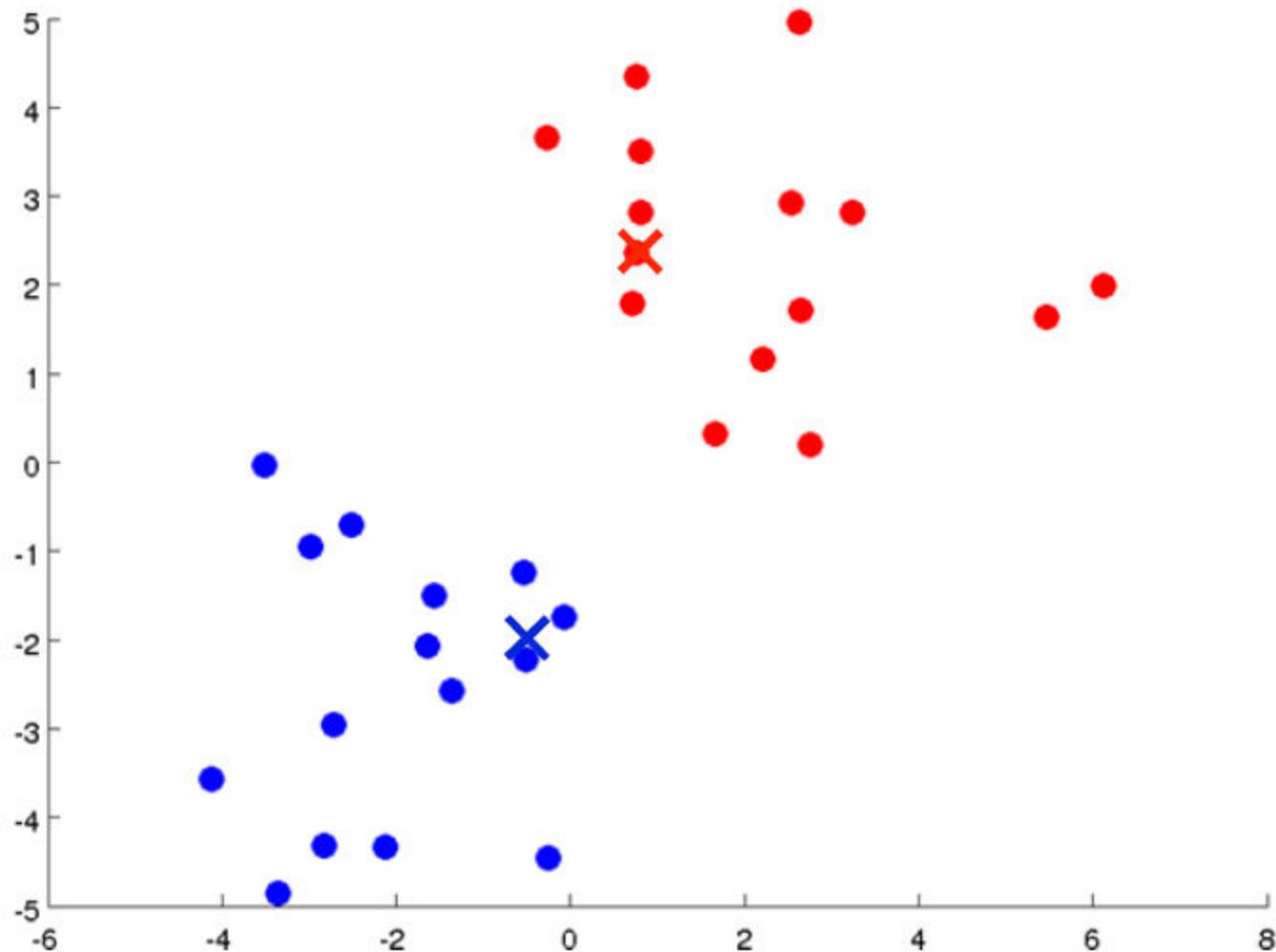


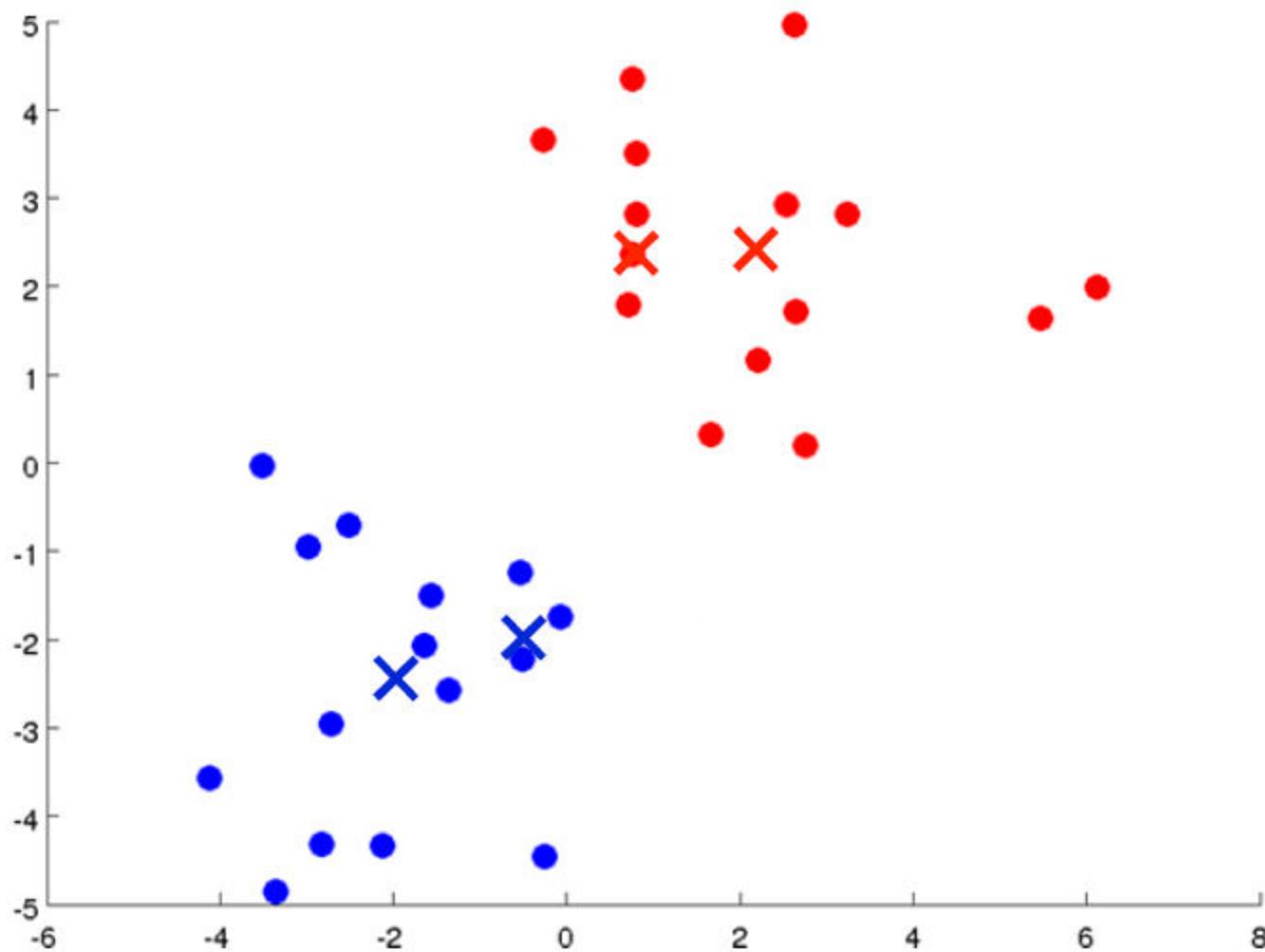


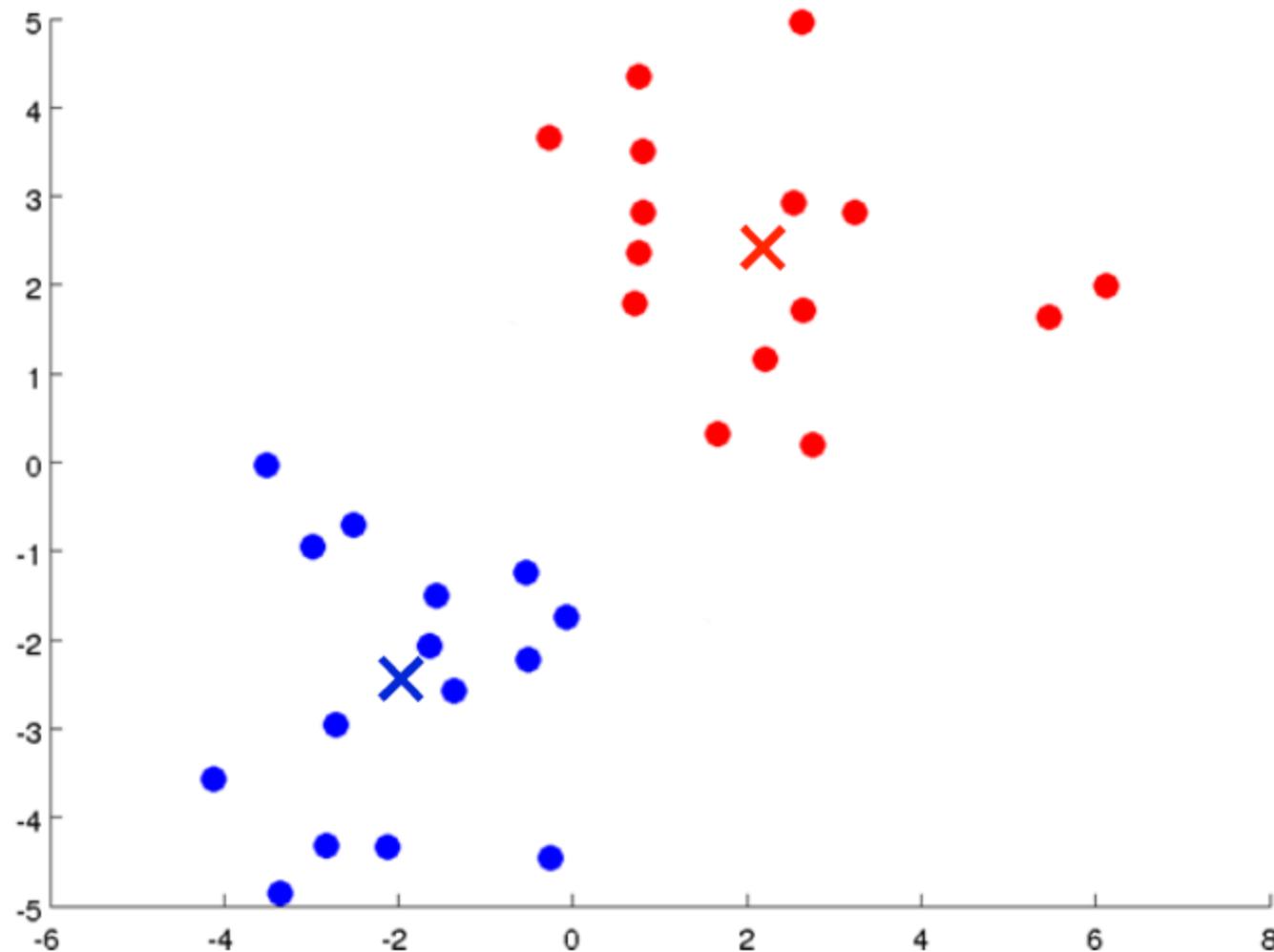












K-means algorithm

K-means algorithm

Input:

- K (number of clusters) 
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ 

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

for $i = 1$ to m

$c^{(i)} :=$ index (from 1 to K) of cluster centroid
closest to $x^{(i)}$

Cluster Assignment

for $k = 1$ to K

$\mu_k :=$ average (mean) of points assigned to cluster k

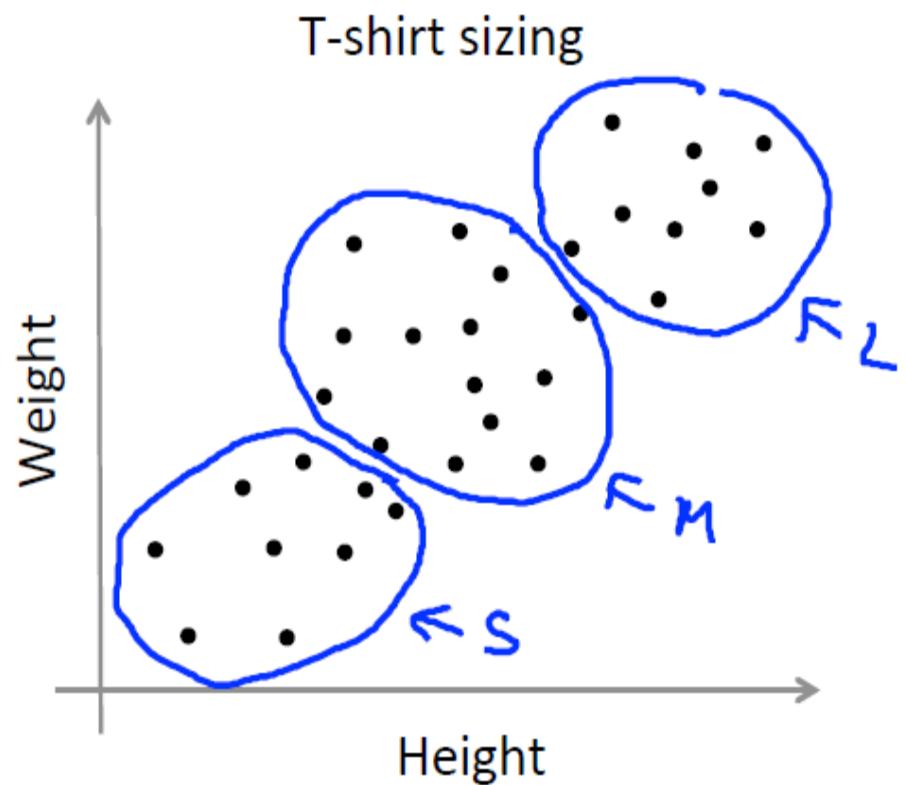
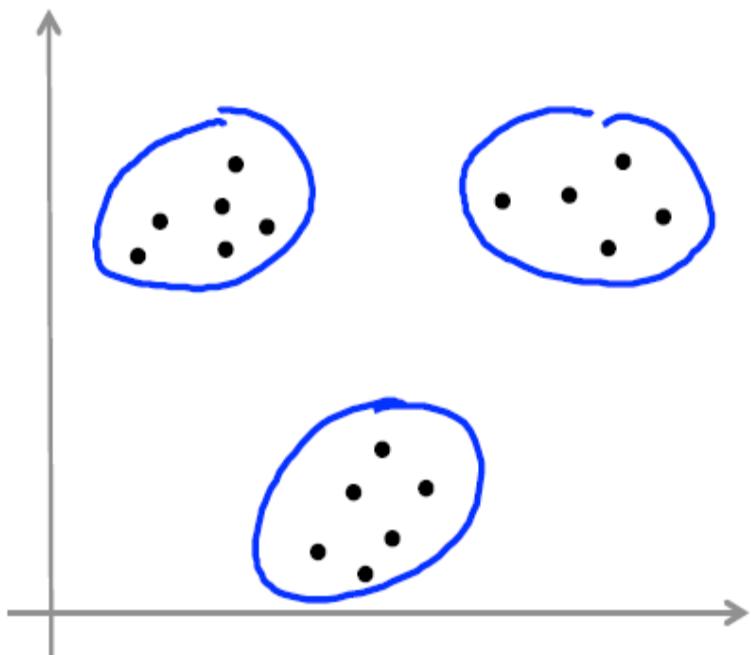
Cluster Moving

}

Non-separated clusters

K-means for non-separated clusters

S, M, L



K-means optimization

K-means optimization objective

$c^{(i)}$ = index of cluster (1,2,...,K) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Optimization objective:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

Random initialization

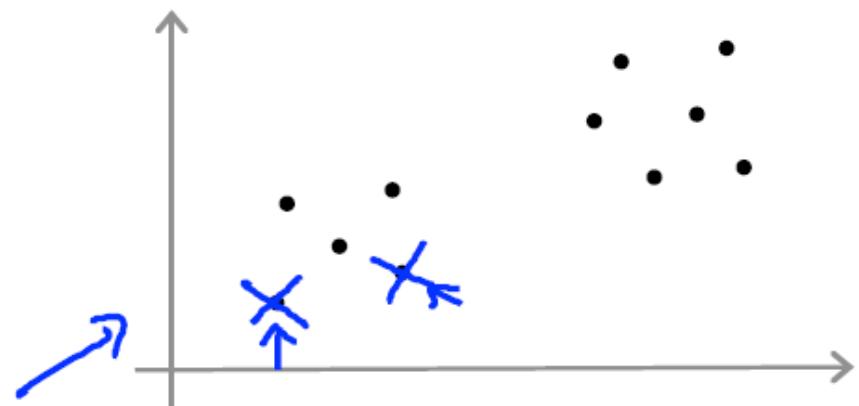
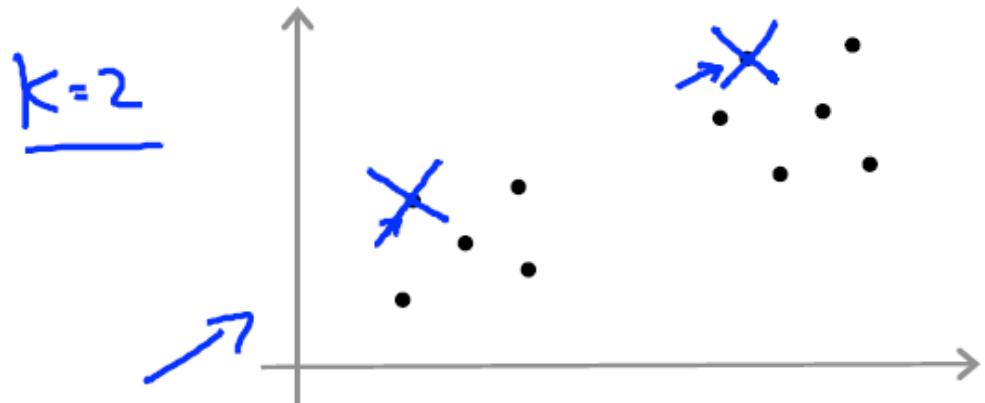
Random initialization

Should have $K < m$

Randomly pick K training examples.

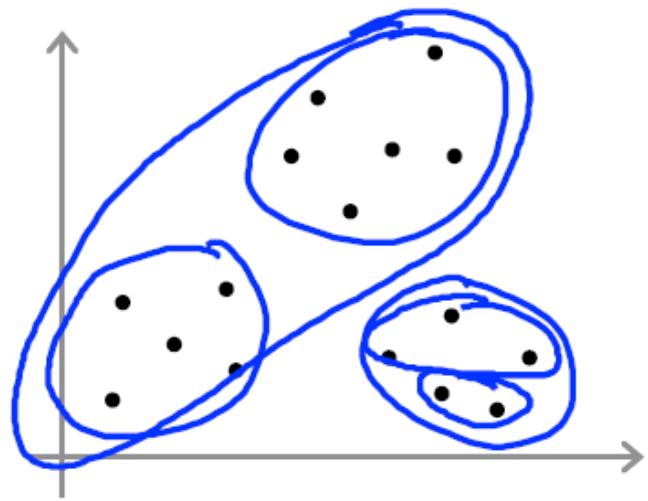
Set μ_1, \dots, μ_K equal to these K examples.

$$\mu_1 = x^{(i)}$$
$$\mu_2 = x^{(j)}$$

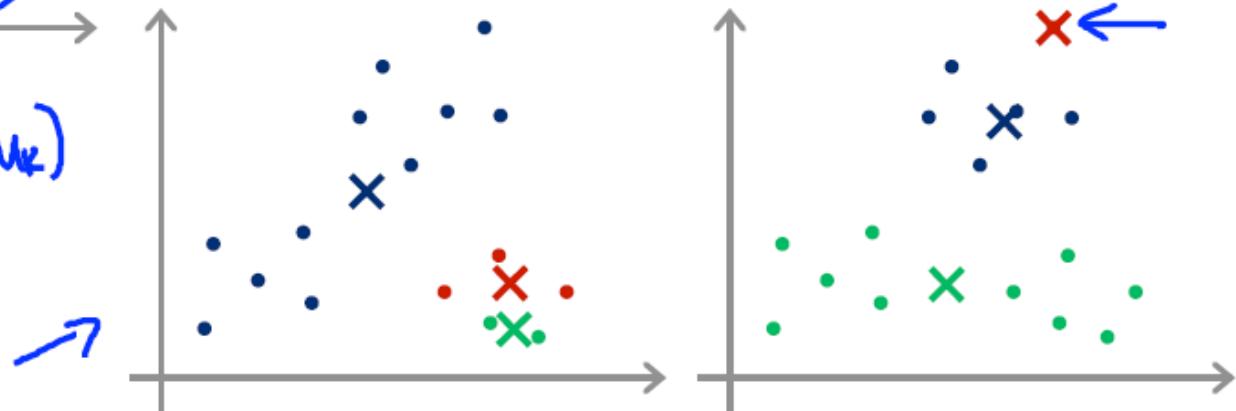


Local Optima

Local optima



$$\mathcal{J}(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k)$$



Random initialization

Random initialization

For i = 1 to 100 {

 Randomly initialize K-means.

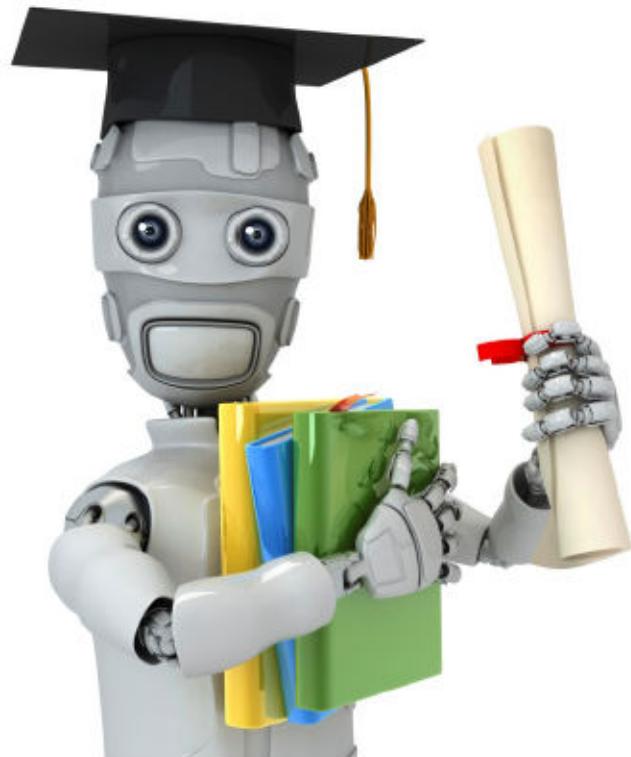
 Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

 Compute cost function (distortion)
 $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

}

Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

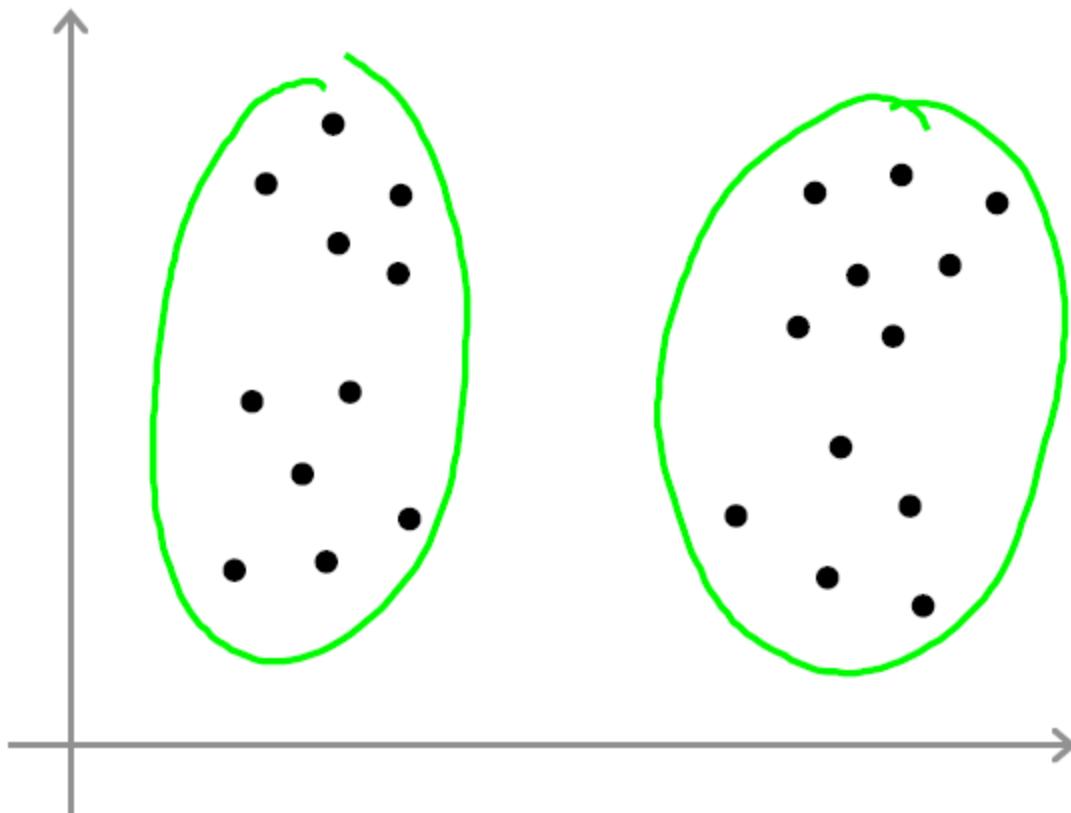
Choosing number of clusters



Machine Learning

Clustering
Choosing the
number of clusters

What is the right value of K?

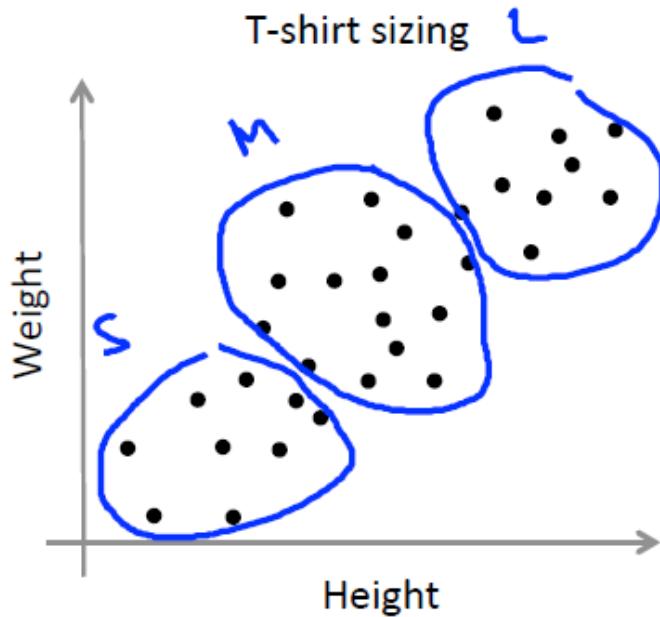


Choosing the value of K

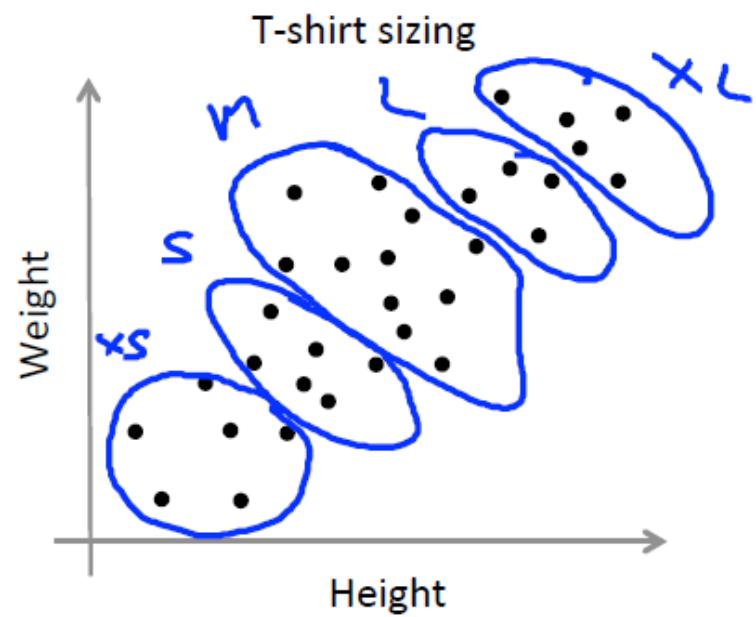
Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

$K=3$ S, M, L

E.g.

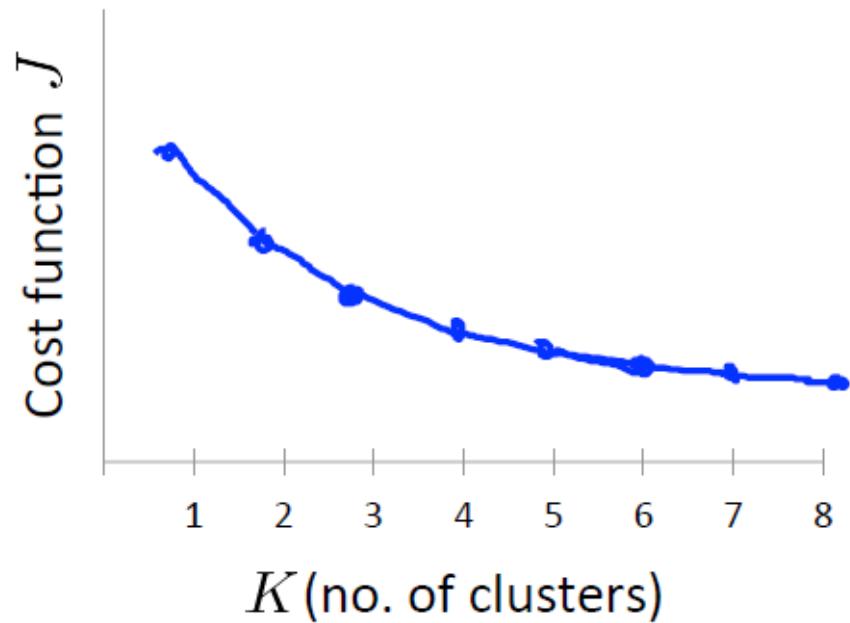
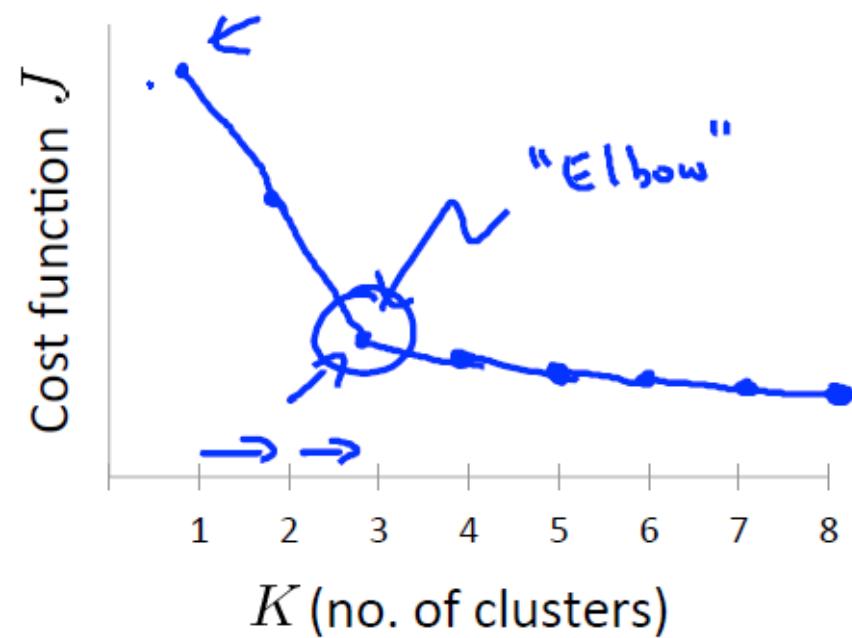


$K=5$ XS, S, M, L, XL

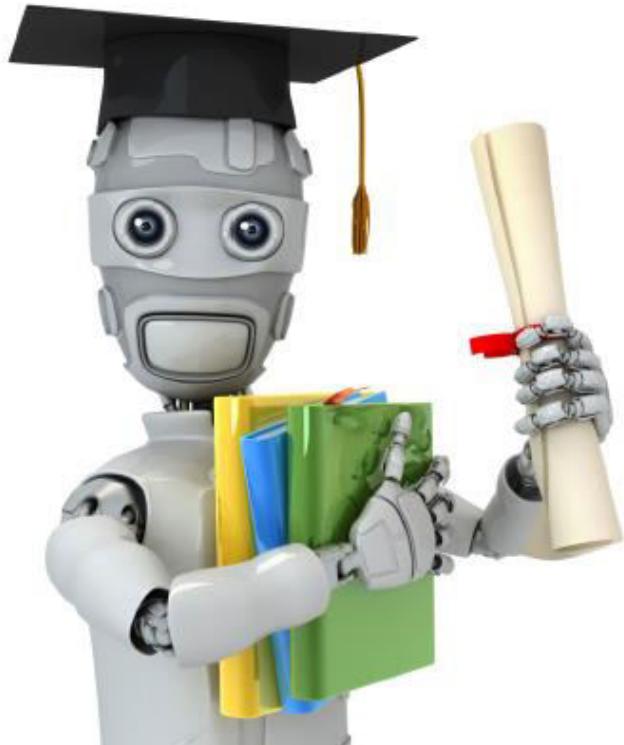


Choosing the value of K

Elbow method:



Dimensionality Reduction : data



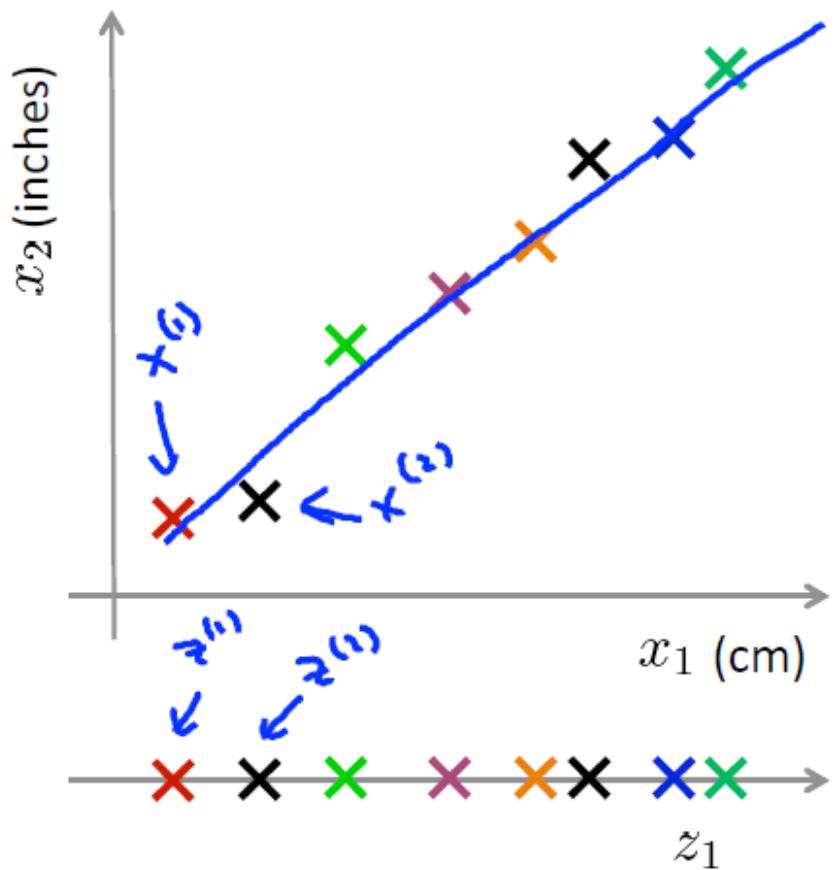
Dimensionality Reduction

Motivation I:
Data Compression

Machine Learning

Data compression

Data Compression



Reduce data from
2D to 1D

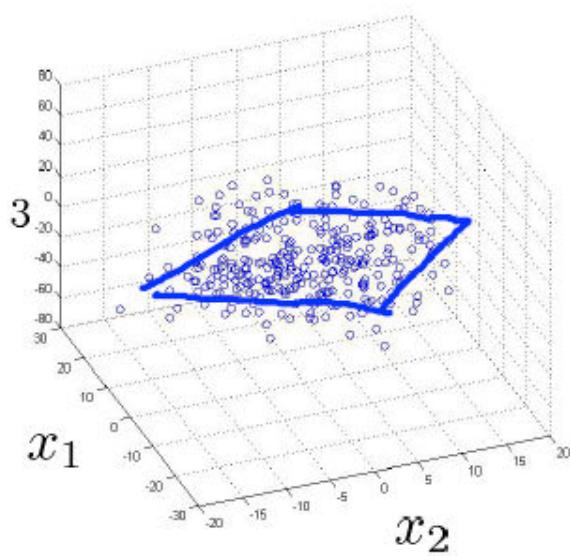
$$\begin{aligned} x^{(1)} \in \mathbb{R}^2 &\rightarrow z^{(1)} \in \mathbb{R} \\ x^{(2)} \in \mathbb{R}^2 &\rightarrow z^{(2)} \in \mathbb{R} \\ &\vdots \\ x^{(m)} \in \mathbb{R}^2 &\rightarrow z^{(m)} \in \mathbb{R} \end{aligned}$$

Data compression

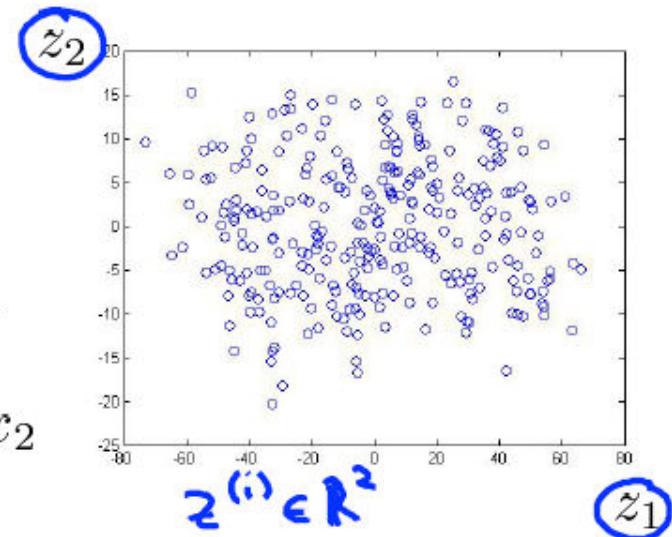
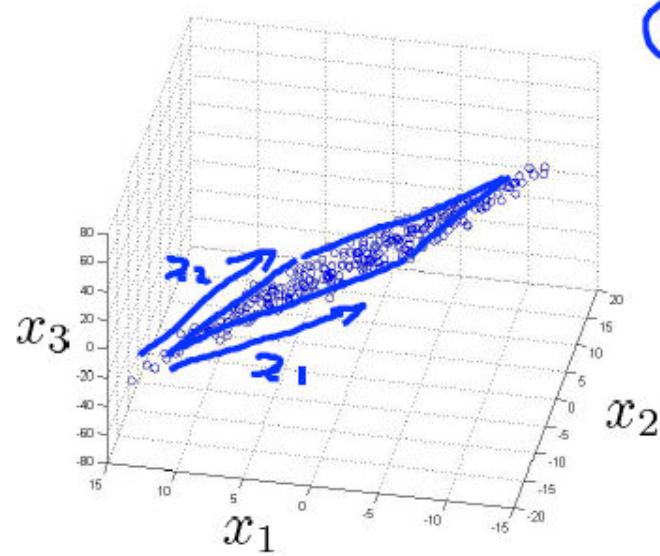
Data Compression

$10000 \rightarrow 1000$

Reduce data from 3D to 2D



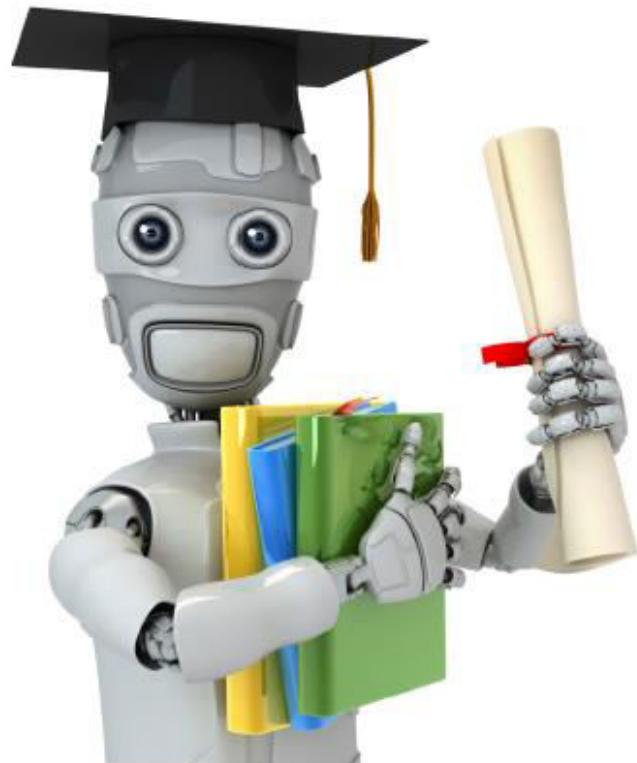
$$x^{(i)} \in \mathbb{R}^3$$



$$z^{(i)} \in \mathbb{R}^2$$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

Data visualization



Machine Learning

Dimensionality Reduction

Motivation II:
Data Visualization

Data Visualization

$$x \in \mathbb{R}^{50}$$

$$x^{(i)} \in \mathbb{R}^{50}$$

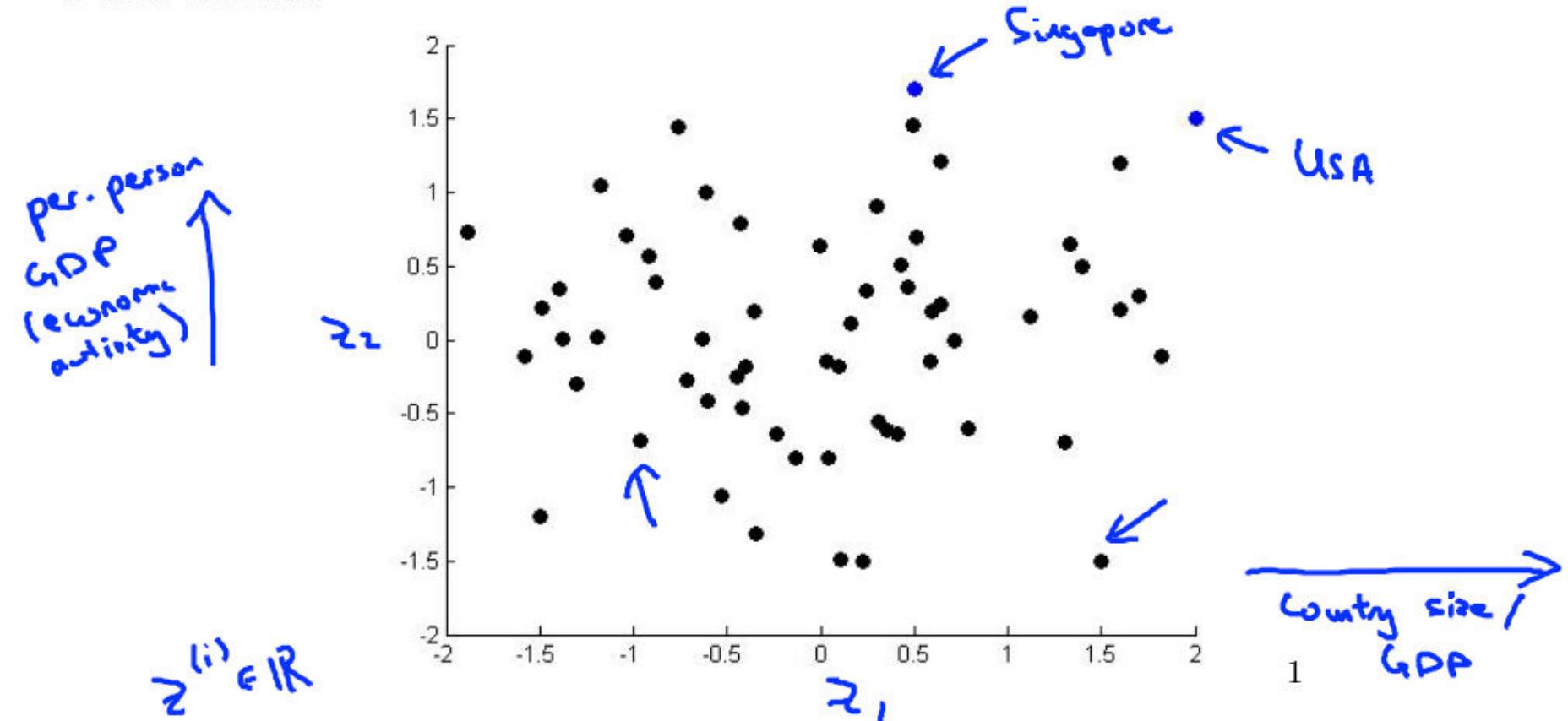
x_6

Country	x_1 GDP (trillions of US\$)	x_2 Per capita GDP (thousands of intl. \$)	x_3 Human Develop- ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	Mean household income (thousands of US\$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...

Data Visualization

Country	z_1	z_2	$z^{(i)} \in \mathbb{R}^2$
Canada	1.6	1.2	
China	1.7	0.3	Reduce data
India	1.6	0.2	from 500
Russia	1.4	0.5	to 2D
Singapore	0.5	1.7	
USA	2	1.5	
...	

Data Visualization



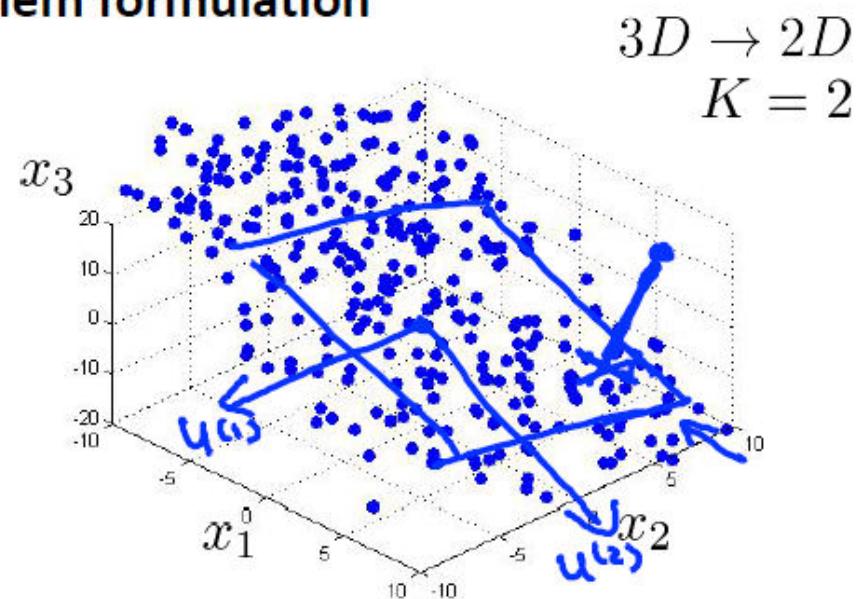
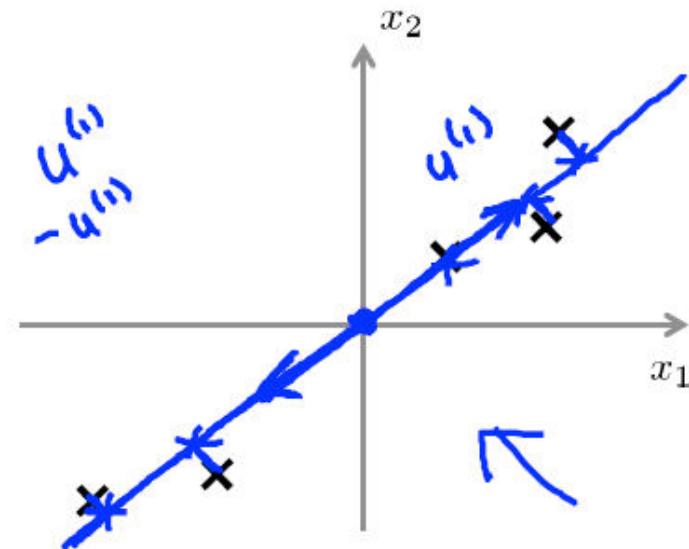
Dimensionality reduction

- Principle Component Analysis (PCA)
- Auto-encoders
- Linear Discriminant Analysis (LDA)
- Genetic algorithms

Principle component analysis

Principle Component Analysis (PCA) problem formulation

Principal Component Analysis (PCA) problem formulation

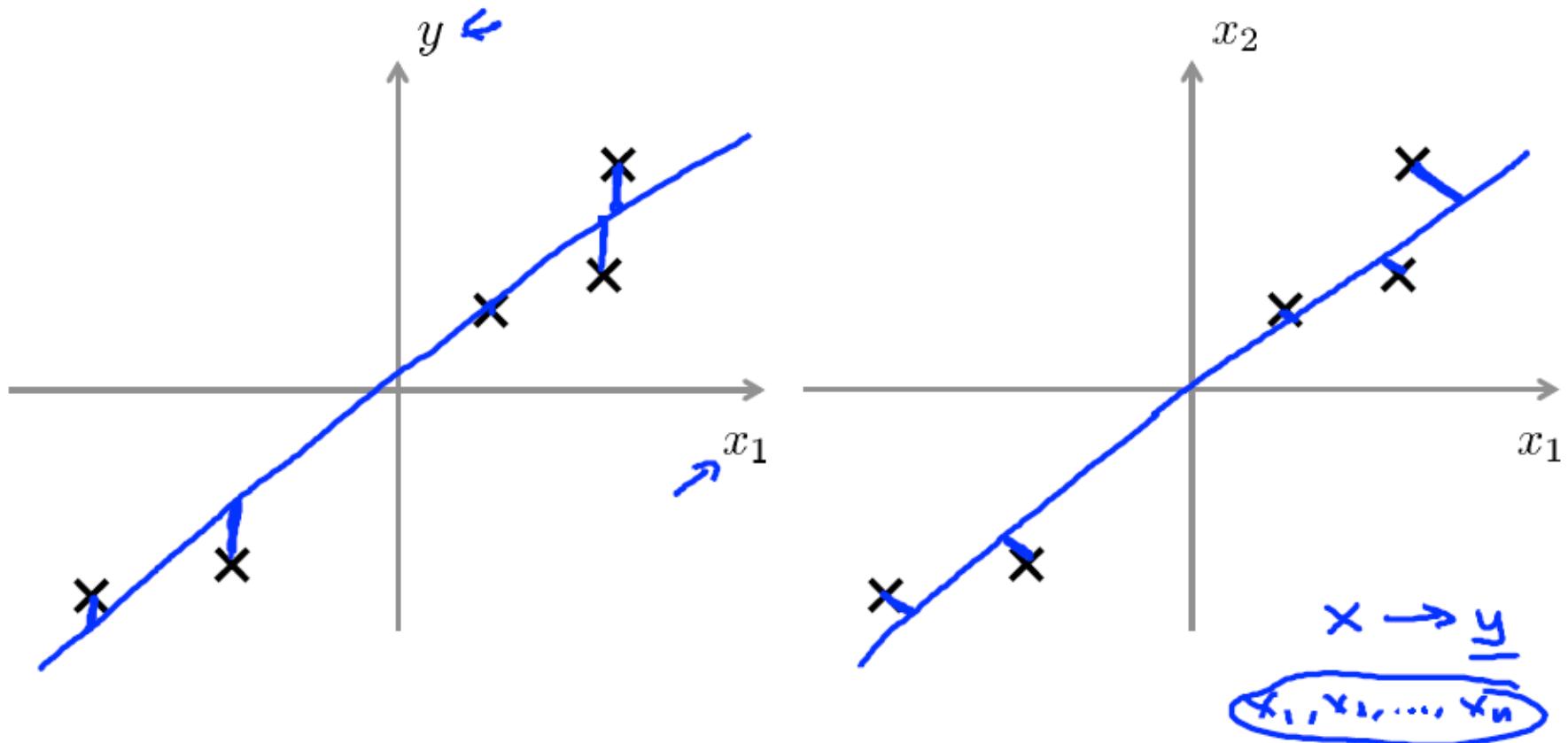


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n -dimension to k -dimension: Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is not linear regression

PCA is not linear regression



Data preprocessing

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $\underline{x_j^{(i)} - \mu_j}$.

If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

PCA algorithm

Principal Component Analysis (PCA) algorithm

Reduce data from n -dimensions to k -dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m} \sum_{i=1}^n (x^{(i)}) (x^{(i)})^T$$

$n \times 1$ $1 \times n$

Sigma

Compute "eigenvectors" of matrix Σ :

- $[U, S, V] = \text{svd}(\Sigma)$;

→ Singular value decomposition
eig(Sigma)

$n \times n$ matrix.

TO DO: PCA Algorithm

PCA algorithm summary

Principal Component Analysis (PCA) algorithm summary

→ After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

$$\text{Sigma} = \frac{1}{m} \sum_{i=1}^m (x^{(i)})(x^{(i)})^T$$

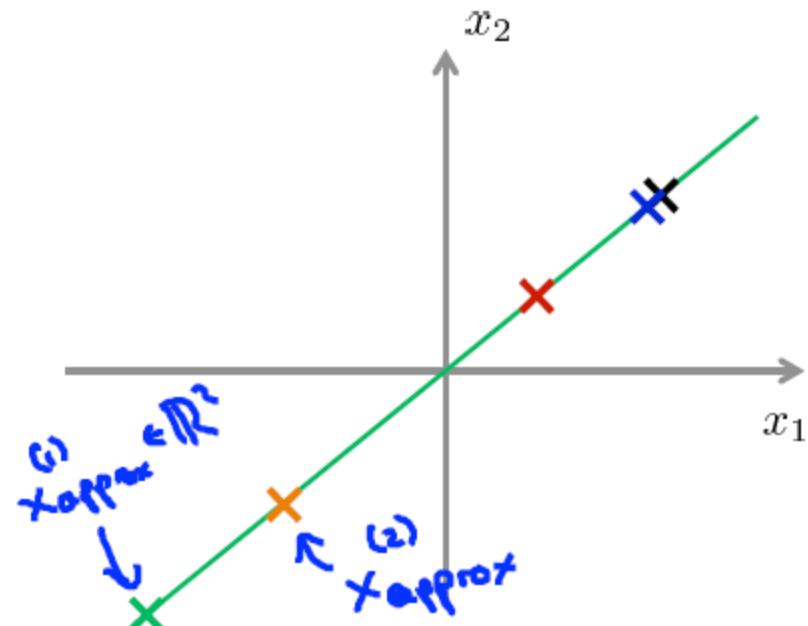
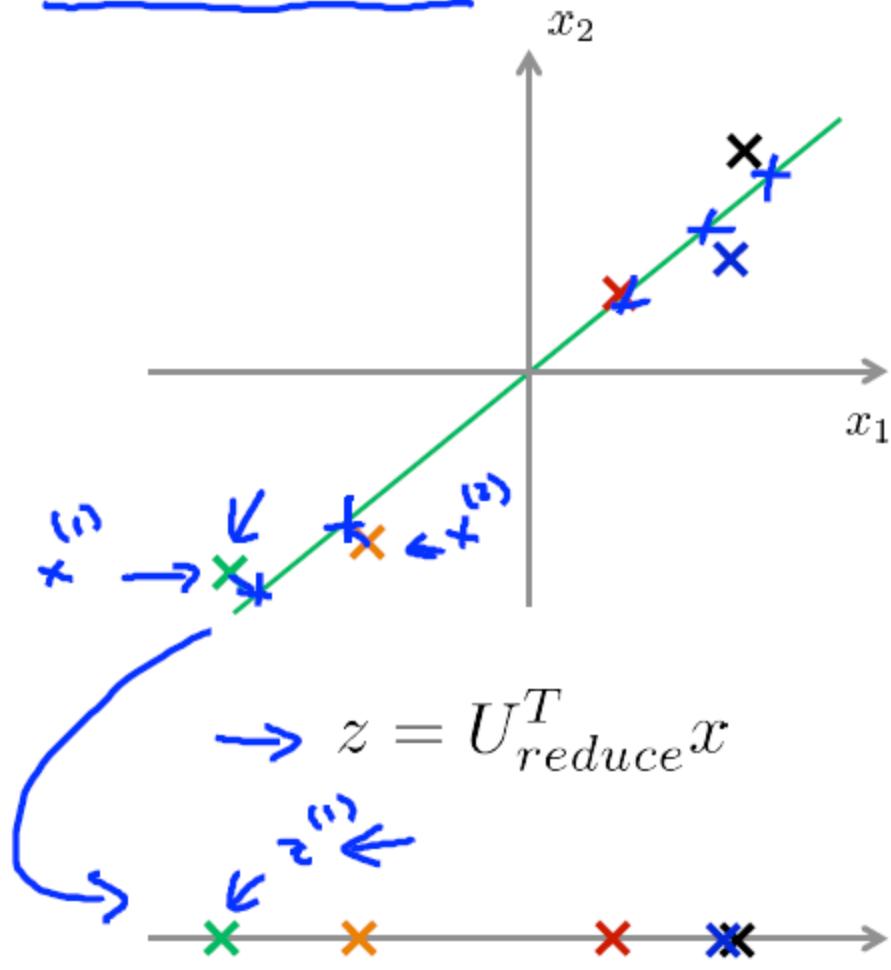
→ `[U,S,V] = svd(Sigma);`

→ `Ureduce = U(:,1:k);`

→ `z = Ureduce' * x;`

To do

Reconstruction from compressed representation



Choosing K – number of PCA components

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01 \quad \begin{matrix} 0.01 \\ \text{---} \\ (1\%) \end{matrix}$$

99%

~~“99% of variance is retained”~~

To Do: Choosing k

Choosing K – number of PCA components

Choosing k (number of principal components)

→ $[U, S, V] = \text{svd}(\text{Sigma})$

Pick smallest value of k for which

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}} \geq 0.99$$

$k=100$

(99% of variance retained)

Supervised learning speedup

Supervised learning speedup

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

Extract inputs:

Unlabeled dataset: $\underline{x}^{(1)}, \underline{x}^{(2)}, \dots, \underline{x}^{(m)} \in \mathbb{R}^{10000}$
 $\downarrow PCA$

$\underline{z}^{(1)}, \underline{z}^{(2)}, \dots, \underline{z}^{(m)} \in \mathbb{R}^{1000}$

New training set:

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

$$h_{\Theta}(z) = \frac{1}{1 + e^{-\Theta^T z}}$$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA
only on the training set. This mapping can be applied as well to
the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test
sets

Application of PCA

Application of PCA

- Compression
 - Reduce memory/disk needed to store data
 - Speed up learning algorithm ←
- Choose k by % of variance retain

Visualization

$k=2$ or $k=3$

Bad use of PCA (to prevent overfitting)

Bad use of PCA: To prevent overfitting

→ Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$. — 10000

Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\rightarrow \min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \boxed{\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2}$$

PCA is sometimes used where it shouldn't be

Design of ML system:

- - Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- - ~~Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
- - Train logistic regression on $\{(\cancel{z^{(1)}}, y^{(1)}), \dots, (\cancel{z^{(m)}}, y^{(m)})\}$
- - Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

- How about doing the whole thing without using PCA?
- Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $\underline{z^{(i)}}$.

Summary

- Unsupervised learning is applied on unlabeled data
- Two main applications
 - Clustering (data mining, recommendation systems)
 - Dimensionality reduction (data compression, visualization)

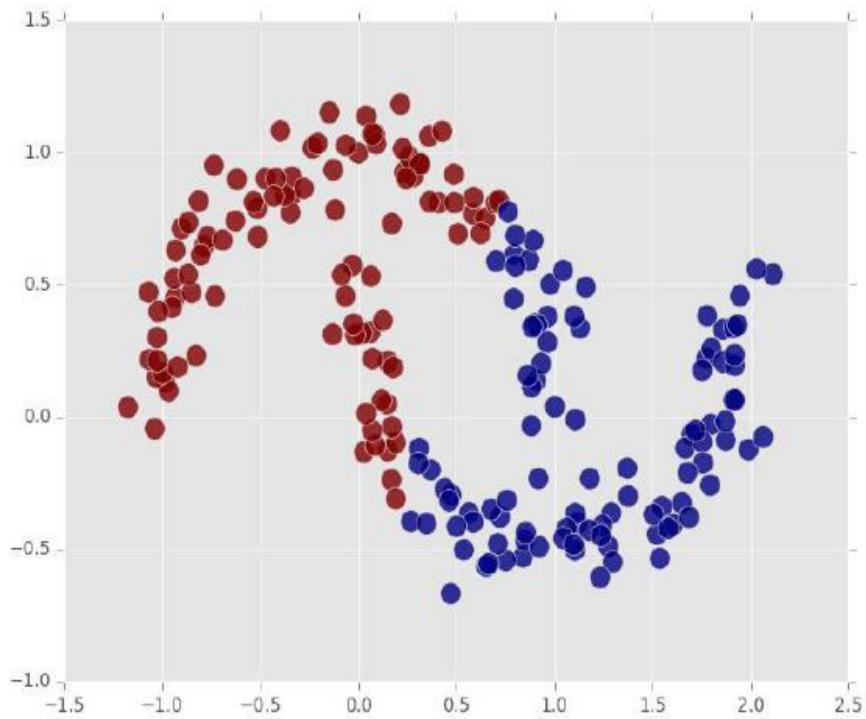
Lecture 9 - Clustering

K-means is the default

- A very simple algorithm
- Lots of resources
- Lots of implementations
- Scales to very large datasets

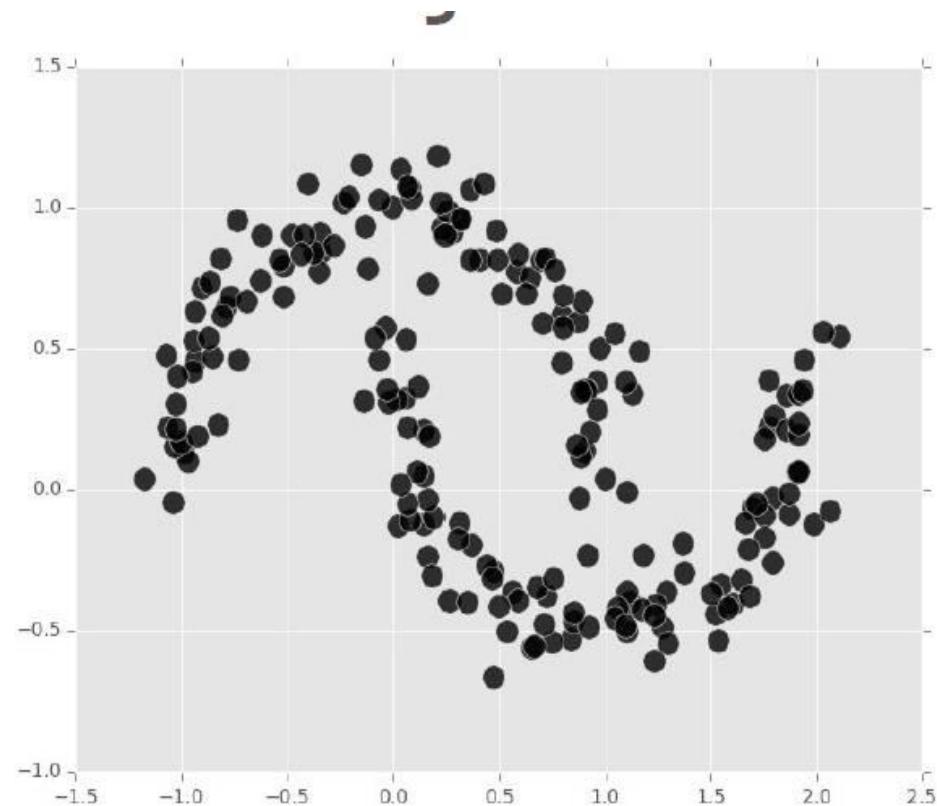
K-means isn't the only answer

- How to choose K?
- Sometimes choosing K is impossible.
- Spherical, convex clusters only.



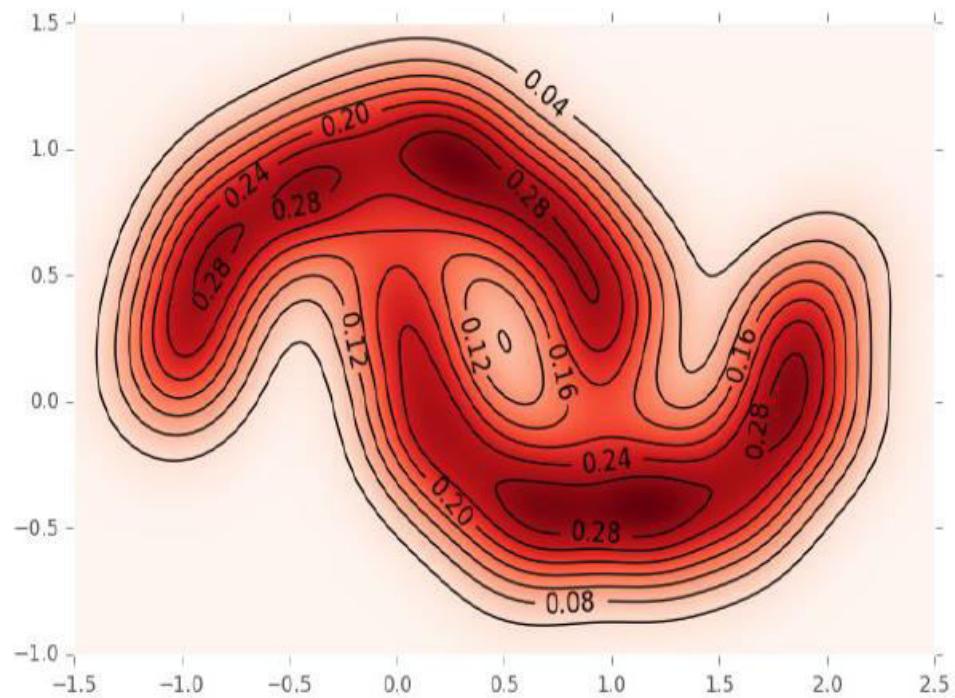
Enter density-based clustering

- **Premise:** data is drawn from probability density function (PDF).



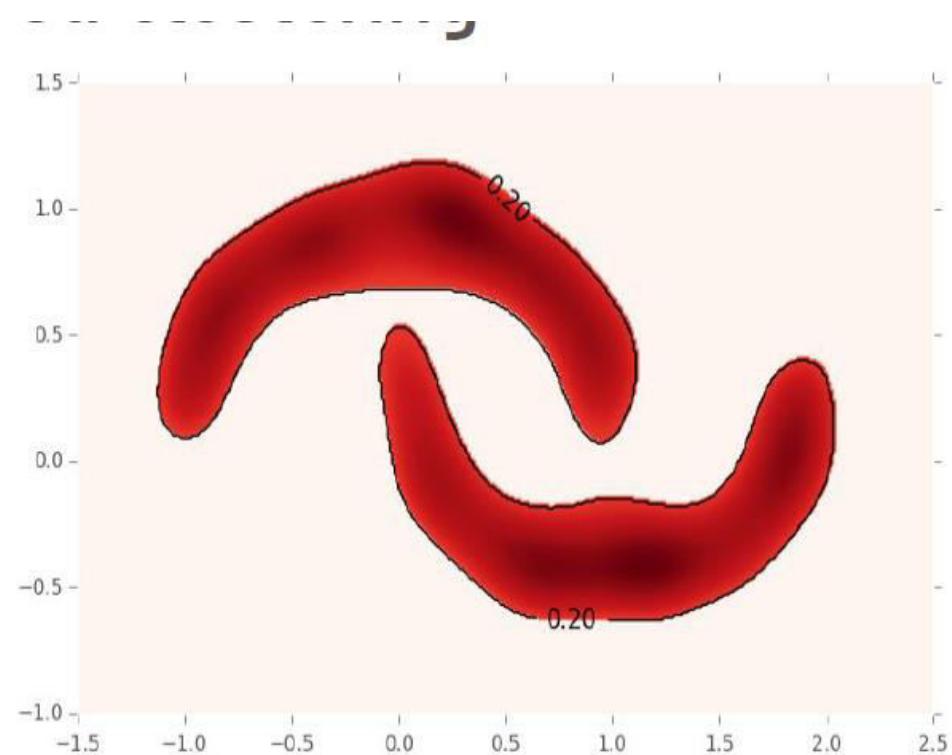
Enter density-based clustering

- **Premise:** data is drawn from probability density function (PDF).
- Use data to estimate the PDF



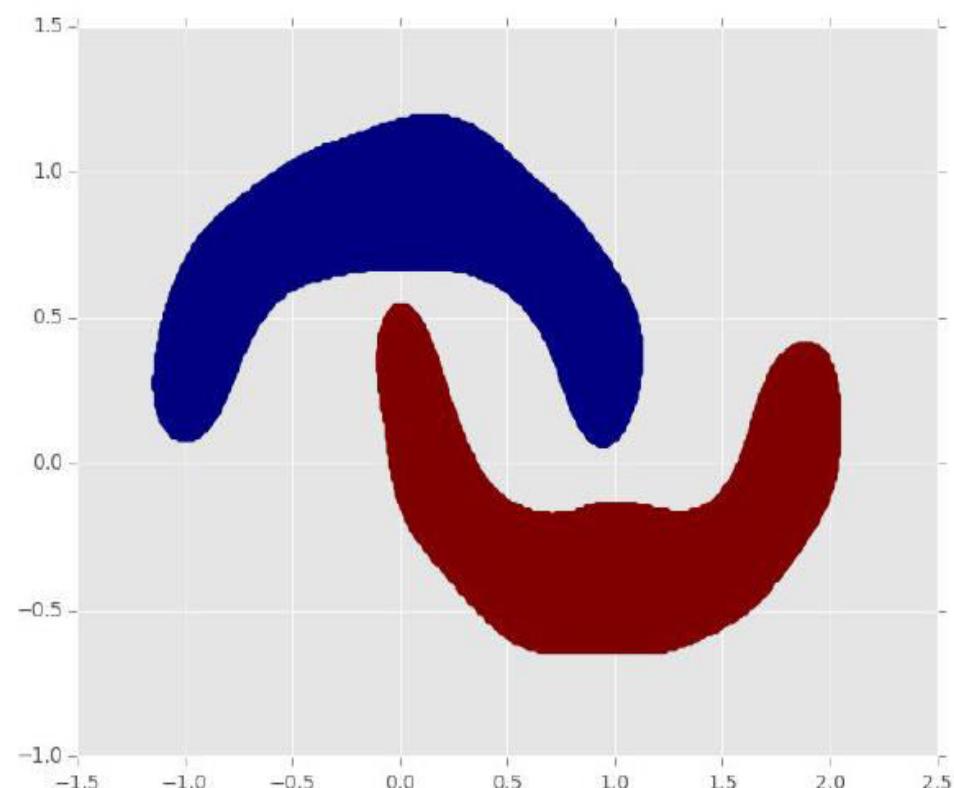
Enter density-based clustering

- Choose a threshold and get the *upper level set*



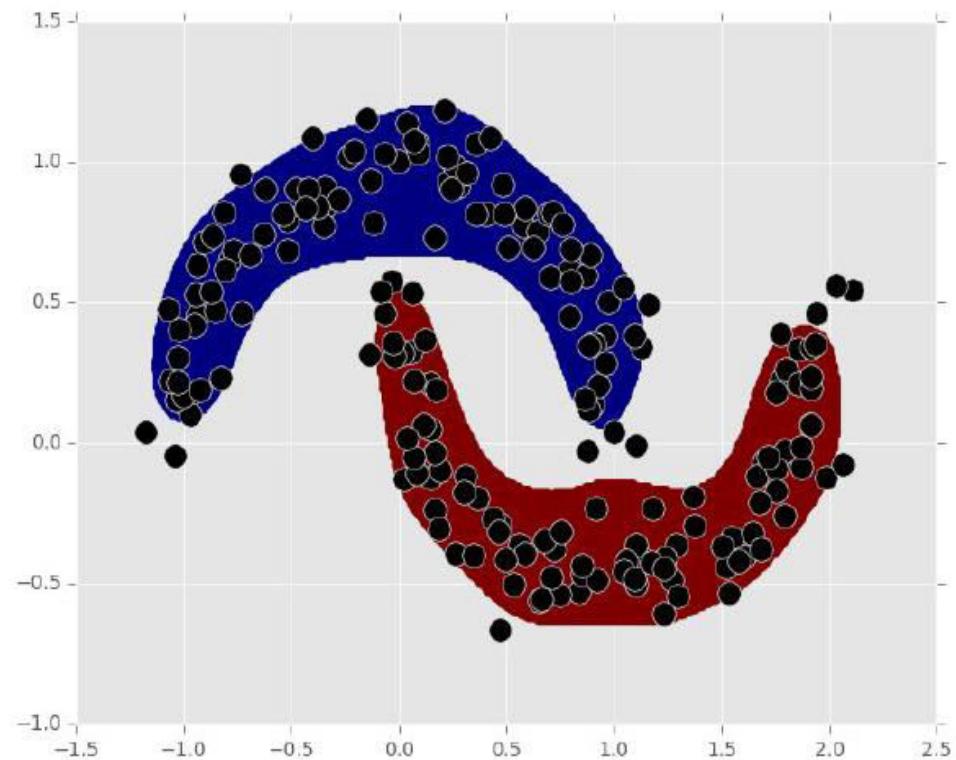
Enter density-based clustering

- Choose a threshold and get the *upper level set*
- Find connected components of the upper level set



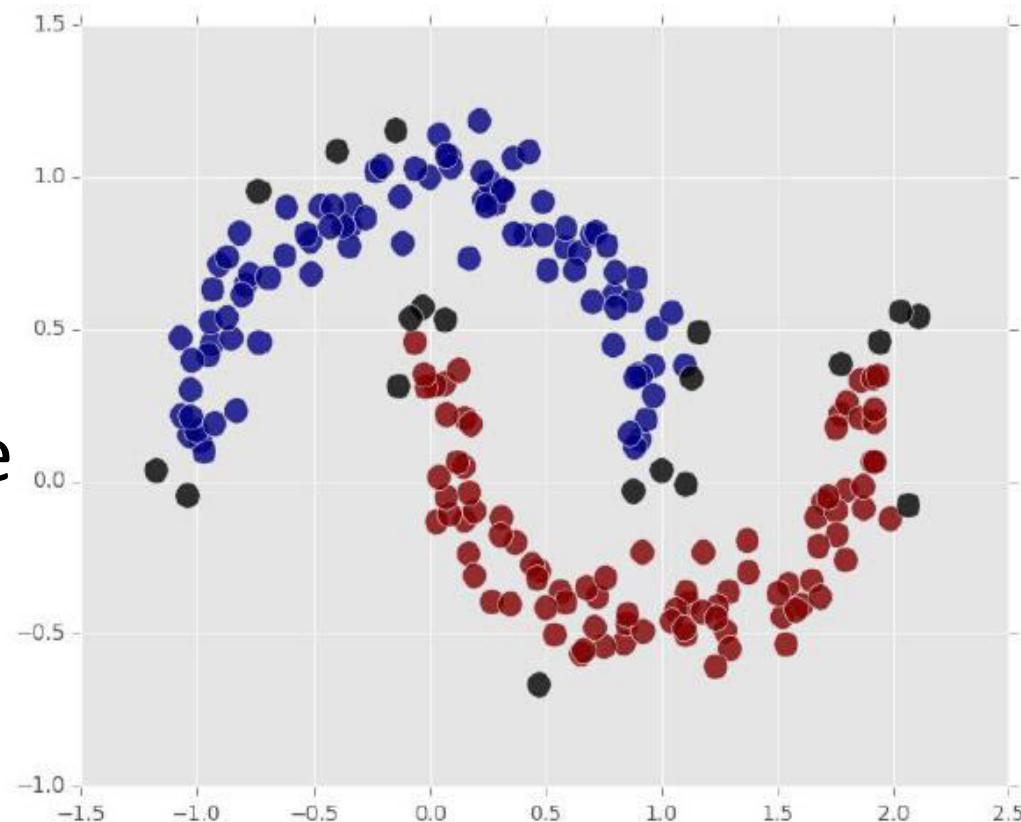
Enter density-based clustering

- Intersect the data with the connected components



Enter density-based clustering

- Intersect the data with the connected components
- Assign points to the corresponding cluster



Pros and cons

- Recovers more complex cluster shapes.
- Don't need to know K.
- Automatically find outliers.
- Requires a distance function.
- Not as scalable as K-means.

DBScan

Concepts: Preliminary

- **DBSCAN is a density-based algorithm**
- DBScan stands for Density-Based Spatial Clustering of Applications with Noise
- Density-based Clustering locates regions of high density that are separated from one another by regions of low density

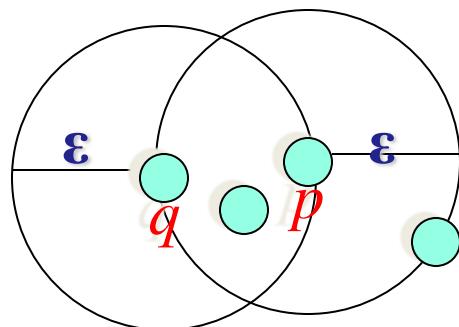
Density = number of points within a specified radius (Eps)

ε -Neighborhood

- ε -Neighborhood – Objects within a radius of ε from an object.

$$N_\varepsilon(p) : \{q \mid d(p, q) \leq \varepsilon\}$$

- “High density” - ε -Neighborhood of an object contains at least MinPts of objects.



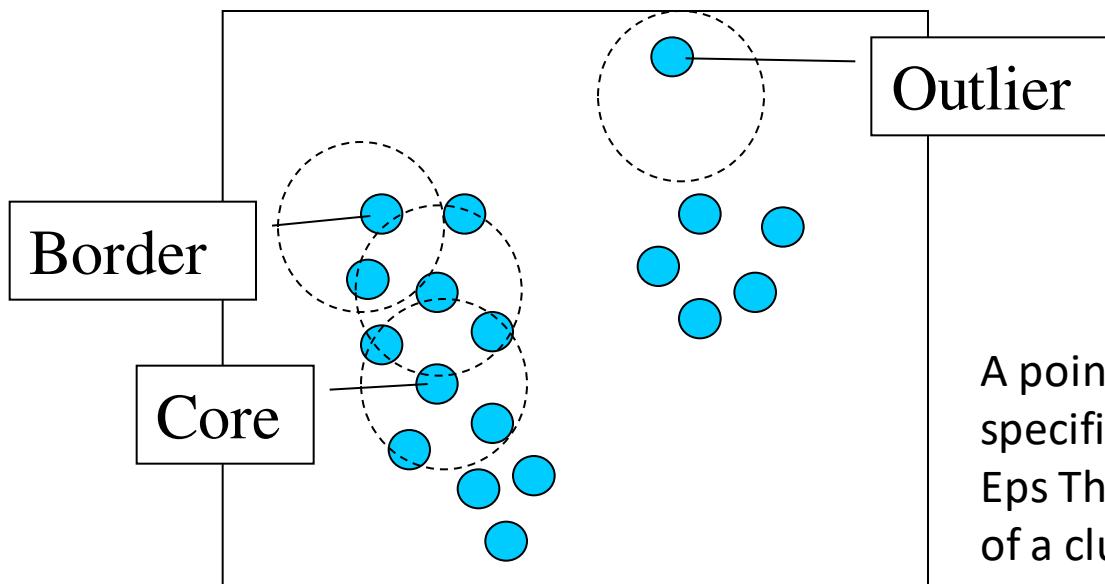
ε -Neighborhood of p

ε -Neighborhood of q

Density of p is “high” ($\text{MinPts} = 4$)

Density of q is “low” ($\text{MinPts} = 4$)

Core, Border & Outlier



Given ϵ and $MinPts$, categorize the objects into three exclusive groups.

A point is a **core point** if it has more than a specified number of points ($MinPts$) within Eps . These are points that are at the interior of a cluster.

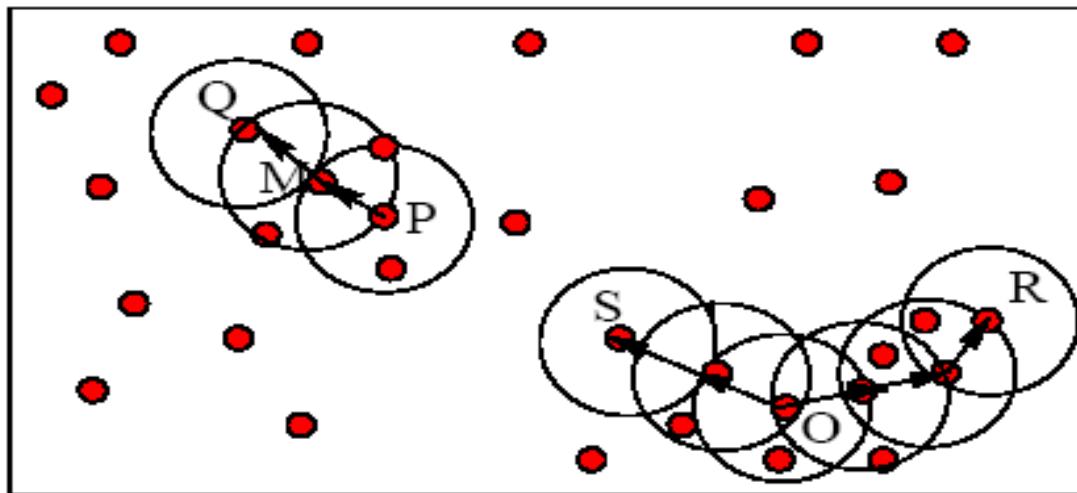
$$\epsilon = 1\text{unit}, MinPts = 5$$

A **border point** has fewer than $MinPts$ within Eps , but is in the neighborhood of a core point.

A **noise point** is any point that is not a core point nor a border point.

Example

- M, P, O, and R are core objects since each is in an Eps neighborhood containing at least 3 points



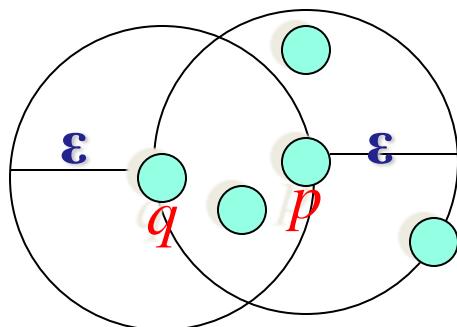
Minpts = 3

Eps=radius
of the circles

Density-Reachability

■ Directly density-reachable

- An object q is directly density-reachable from object p if p is a core object and q is in p 's ϵ -neighborhood.

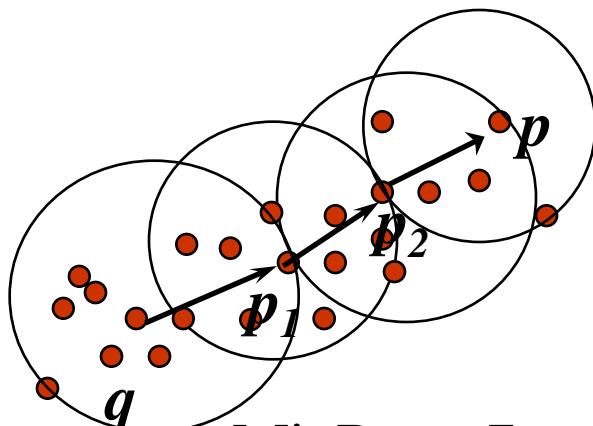


- q is directly density-reachable from p
- p is not directly density-reachable from q
- Density-reachability is asymmetric.

MinPts = 4

Density-reachability

- Density-Reachable (directly and indirectly):
 - A point p is directly density-reachable from p_2 ;
 - p_2 is directly density-reachable from p_1 ;
 - p_1 is directly density-reachable from q ;
 - $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain.



- **p is (indirectly) density-reachable from q**
- **q is not density- reachable from p**

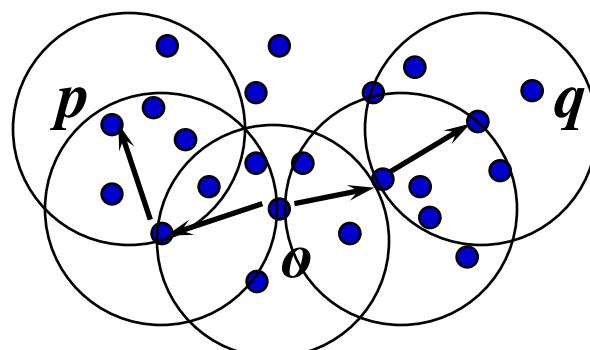
Density-Connectivity

■ Density-reachable is not symmetric

- not good enough to describe clusters

■ Density-Connected

- A pair of points p and q are density-connected if they are commonly density-reachable from a point o .

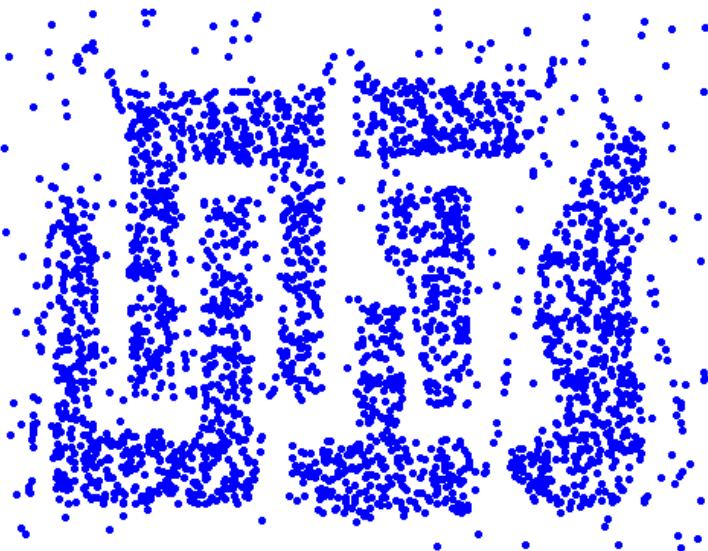


■ Density-connectivity is symmetric

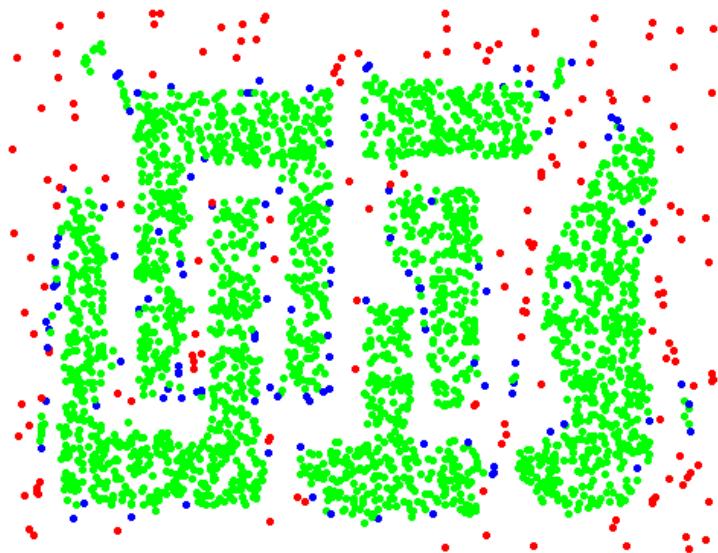
DBSCAN: The Algorithm

- Arbitrary select a point p
- Retrieve all points density-reachable from p wrt Eps and $MinPts$.
- If p is a core point, a cluster is formed.
- If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.
- The points that don't belong to any cluster are the outliers

Example



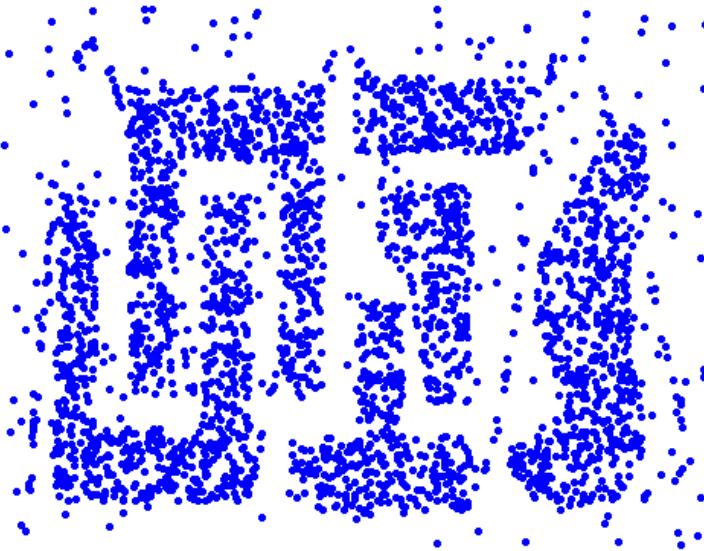
Original Points



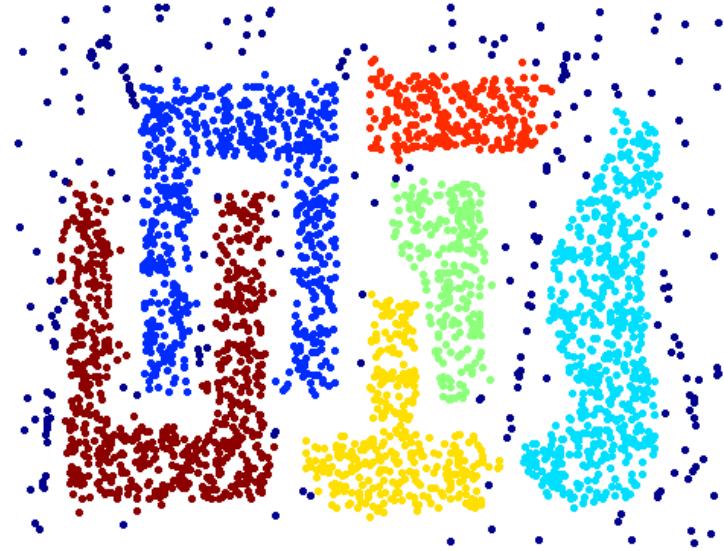
Point types: **core**,
border and **outliers**

$\varepsilon = 10$, MinPts = 4

When DBSCAN Works Well



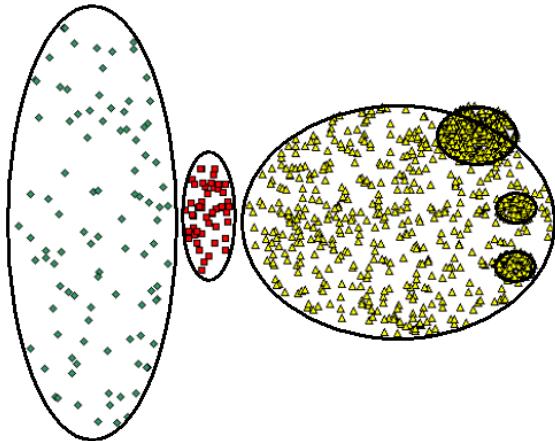
Original Points



Clusters

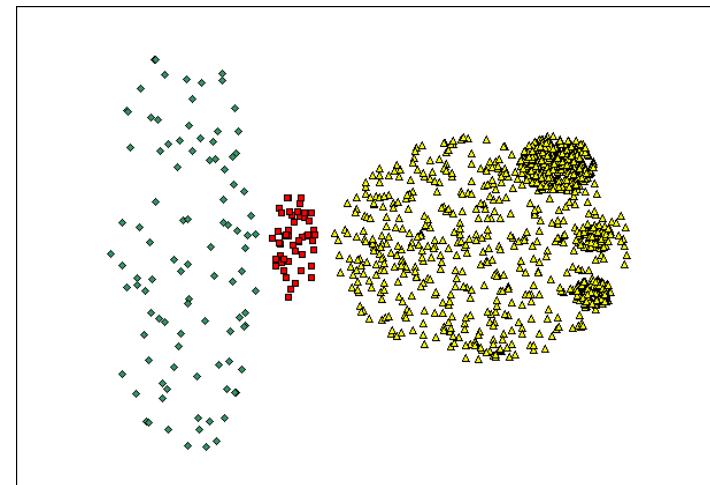
- Resistant to Noise
- Can handle clusters of different shapes and sizes

When DBSCAN Does NOT Work Well

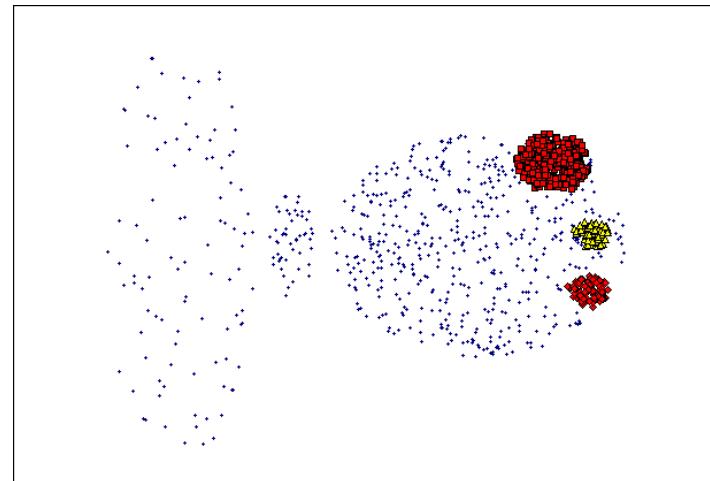


Original Points

- Cannot handle Varying densities
- sensitive to parameters



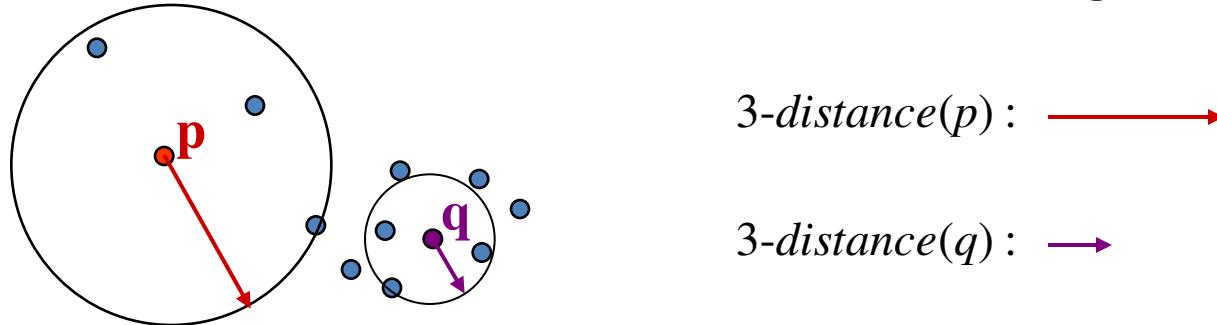
(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

Determining the Parameters ε and $MinPts$

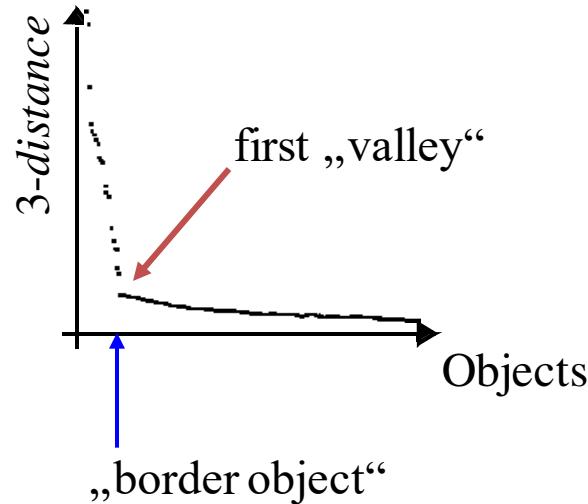
- Cluster: Point density higher than specified by ε and $MinPts$
- Idea: use the point density of the least dense cluster in the data set as parameters – but how to determine this?
- Heuristic: look at the distances to the k -nearest neighbors



- Function $k\text{-distance}(p)$: distance from p to its k -nearest neighbor
- $k\text{-distance plot}$: k -distances of all objects, sorted in decreasing order

Determining the Parameters ε and $MinPts$

- Example k -distance plot



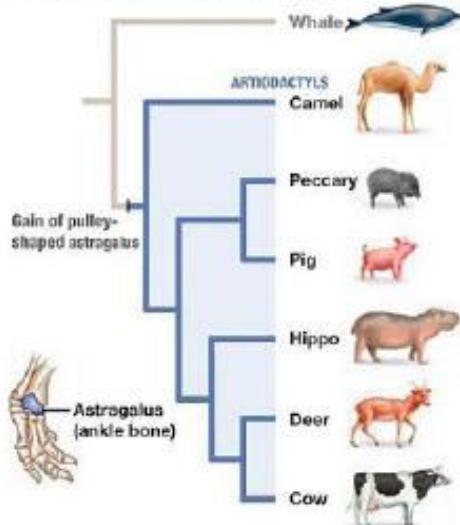
- Heuristic method:
 - Fix a value for $MinPts$ (default: $2 \times dimensions - 1$)
 - User selects “border object” o from the $MinPts$ -distance plot; ε is set to $MinPts$ -distance(o)

DBScan visualization

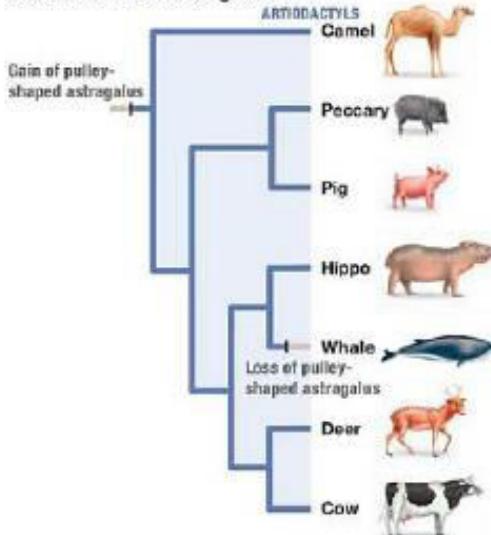
<http://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>

Hierarchical Clustering

(a) The astragalus is a synapomorphy that identifies artiodactyls as a monophyletic group.



(b) If whales are related to hippos, then two changes occurred in the astragalus.



(c) Data on the presence and absence of SINE genes support the close relationship between whales and hippos.

Locus	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Cow	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	
Deer	0	0	0	0	0	0	1	?	1	1	1	1	1	1	?	1	1	0	0	
Whale	1	1	1	1	1	1	1	0	?	1	0	1	1	0	0	?	1	0	0	
Hippo	0	?	0	1	1	1	1	0	1	1	0	1	1	0	0	0	?	1	0	0
Pig	0	0	0	?	0	0	0	0	?	0	0	0	0	0	0	0	0	1	1	1
Peccary	?	?	?	?	?	?	?	?	?	2	2	2	2	2	2	2	?	2	1	1
Camel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

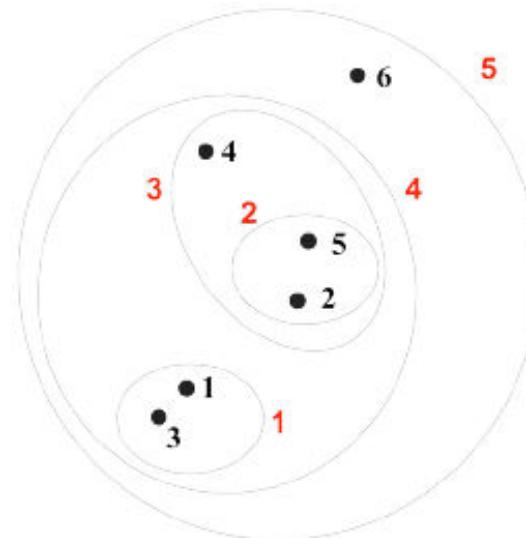
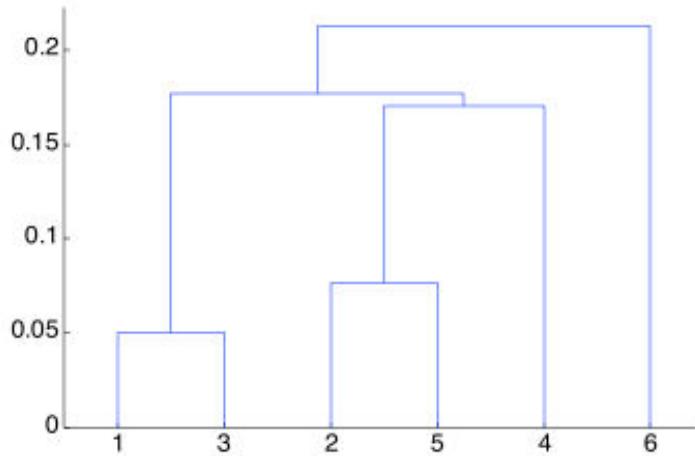
1 = gene present
0 = gene absent
? = still undetermined

Whales and hippos share four unique SINE genes (4, 5, 6, and 7)

© 2010 Pearson Education, Inc.

Hierarchical Clustering

- Produces a set of **nested clusters** organized as a hierarchical tree
- Can be visualized as a **dendrogram**
 - A tree-like diagram that records the sequences of merges or splits



Strengths of Hierarchical Clustering

- No assumptions on the number of clusters
 - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- Hierarchical clustering may correspond to meaningful taxonomies
 - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Hierarchical Clustering Algorithms

- Two main types of hierarchical clustering
 - **Agglomerative:**
 - Start with the points as individual clusters
 - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
 - **Divisive:**
 - Start with one, all-inclusive cluster
 - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
 - Merge or split one cluster at a time

Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm:

 Compute the distance matrix between the input data points

 Let each data point be a cluster

Repeat

 Merge the two closest clusters

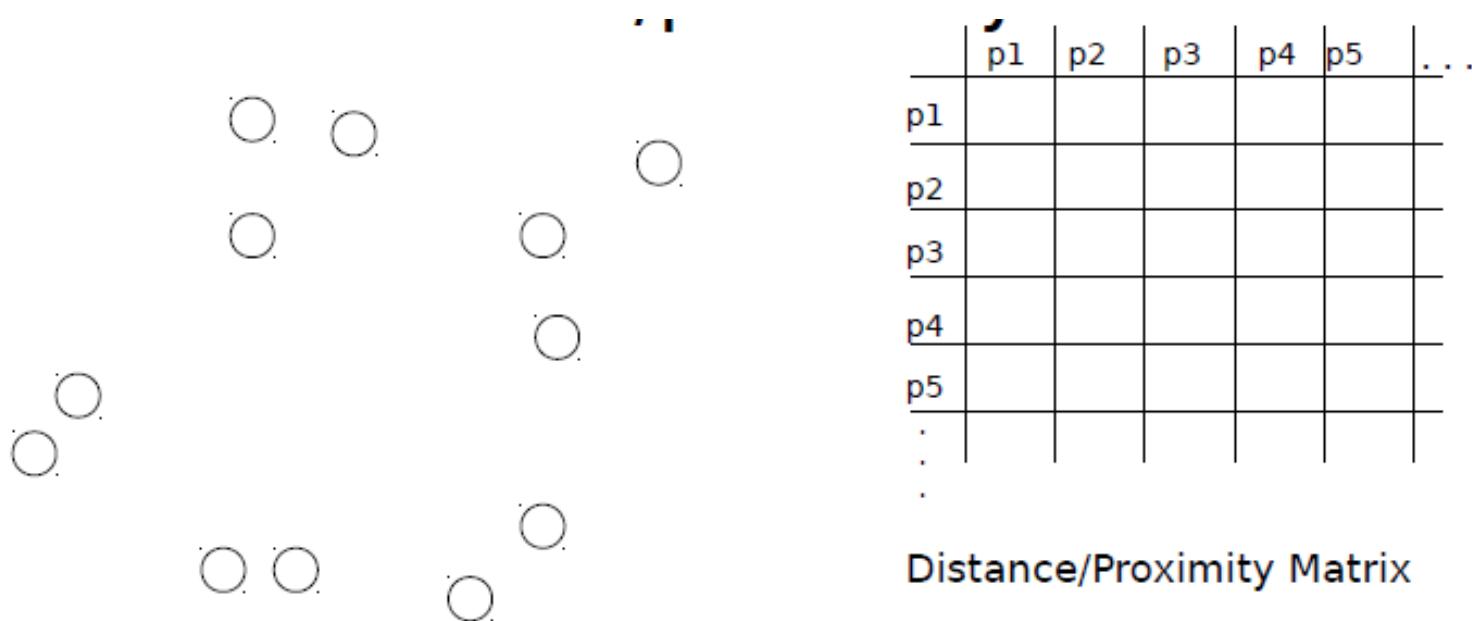
 Update the distance matrix

Until only a single cluster remains

- Key operation is the computation of the distance between two clusters
 - Different definitions of the distance between clusters lead to different algorithms

Inout / Initial Setting

- Start with clusters of individual points and a distance matrix



Intermediate State

- After some merging steps, we have some clusters

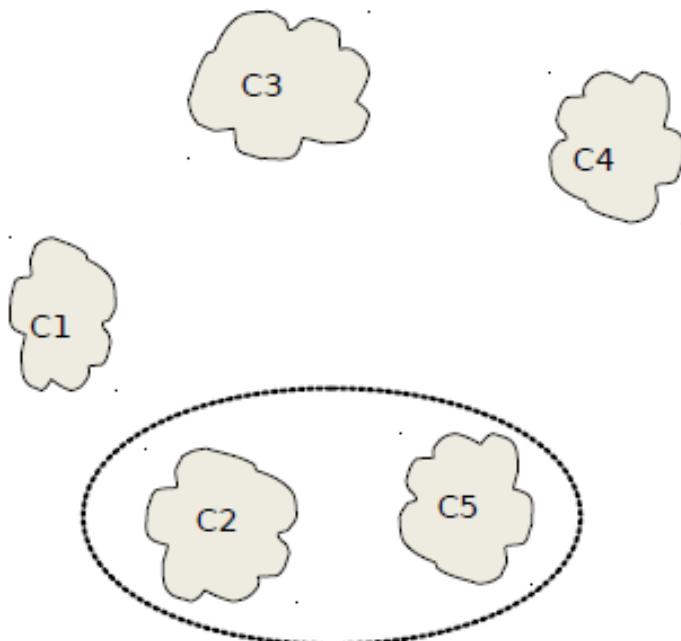


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix

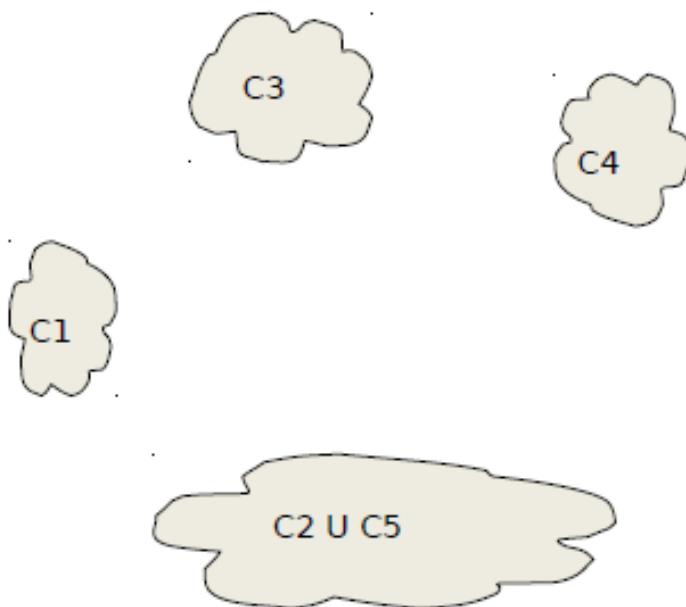


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

After Merging

- How do we update the distance matrix?



		C1	C3	C4
C1		?		
C2 U C5	?	?	?	
C3		?		
C4		?		

Distance between two clusters

- Each cluster is a set of points
- How do we define distance between two sets of points
 - Lots of alternatives
 - Not an easy task

Distance between two clusters

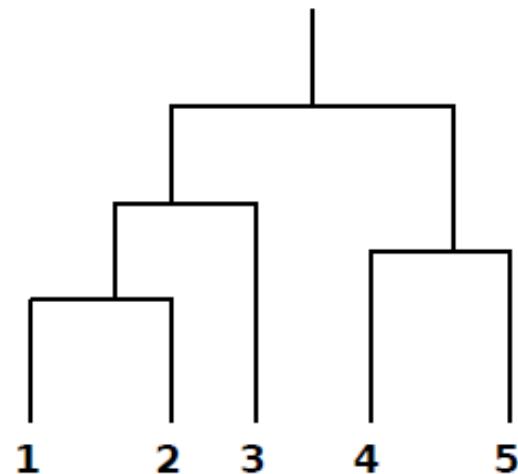
- Single-link distance between clusters C_i and C_j
- C_j is the minimum distance between any object in C_i and any object in C_j
- The distance is defined by the two most similar objects

$$D_{\text{single}} = \min_{x,y} \{d(x, y) \mid x \in C_i, y \in C_j\}$$

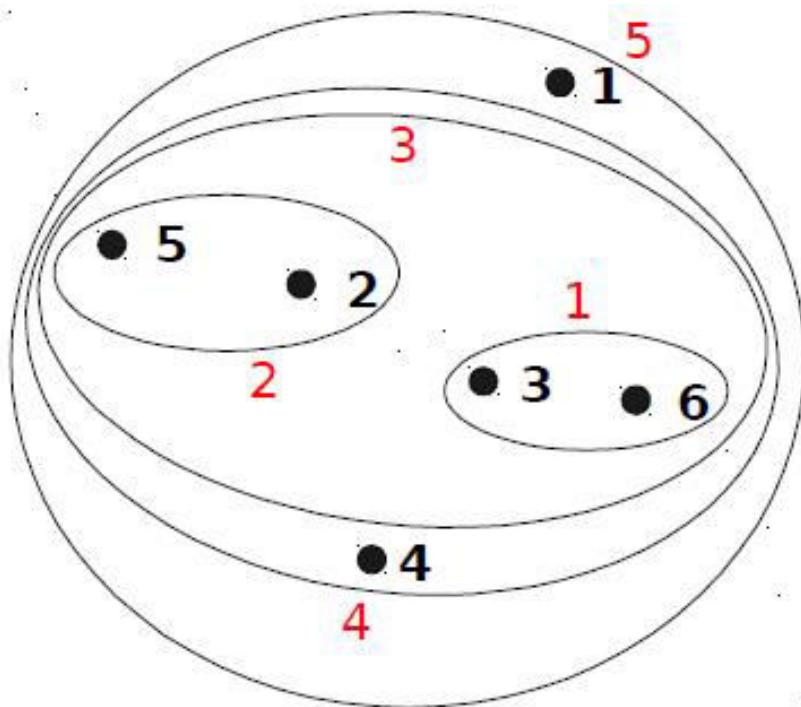
Single-link clustering example

- Determined by one pair of points, i.e., by one link in the proximity graph.

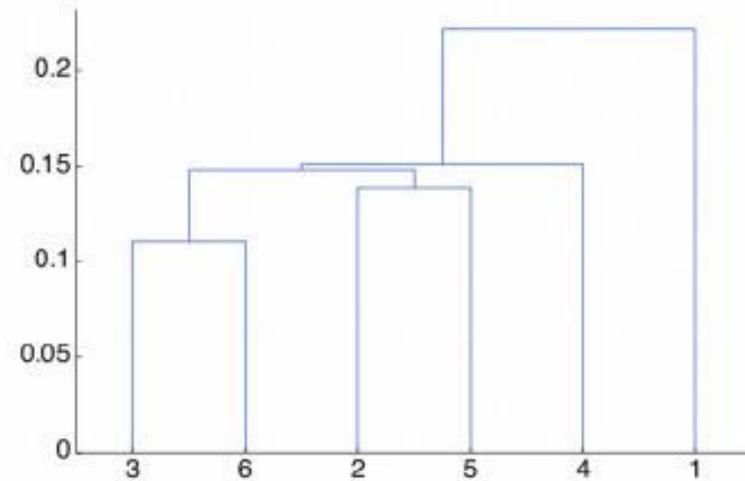
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link clustering example

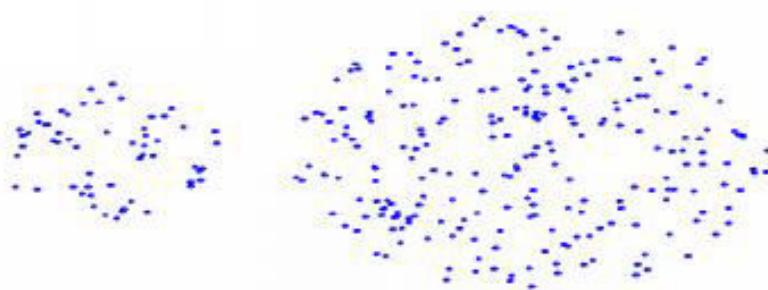


Nested Clusters

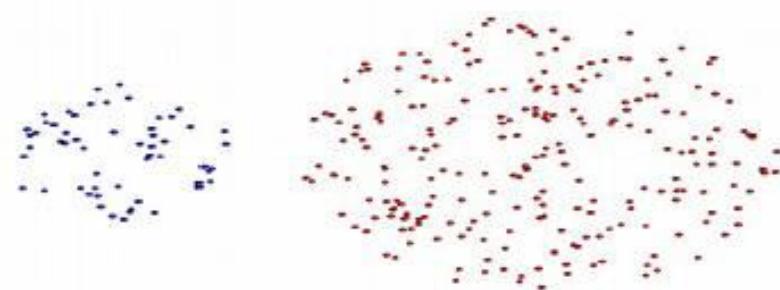


Dendrogram

Strengths of single-link clustering



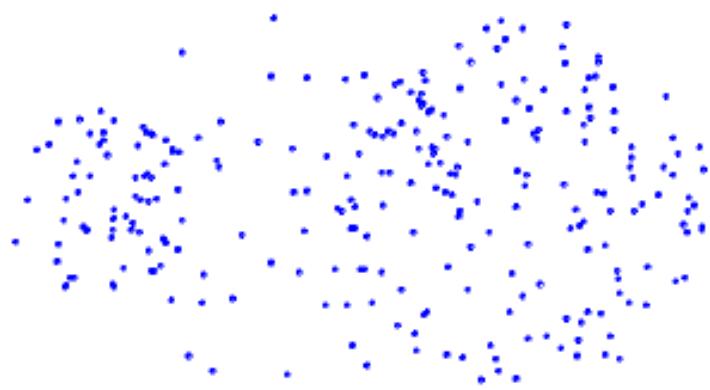
Original Points



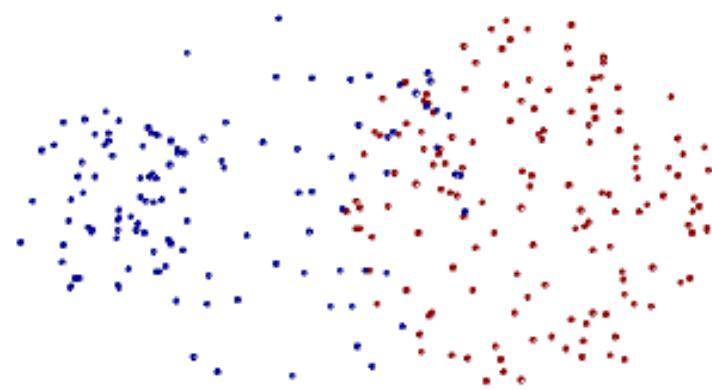
Two Clusters

- Can handle non-elliptical shapes

Limitations of single-link clustering



Original Points



Two Clusters

- Sensitive to noise and outliers
- It produces long, elongated clusters

Distance between two clusters

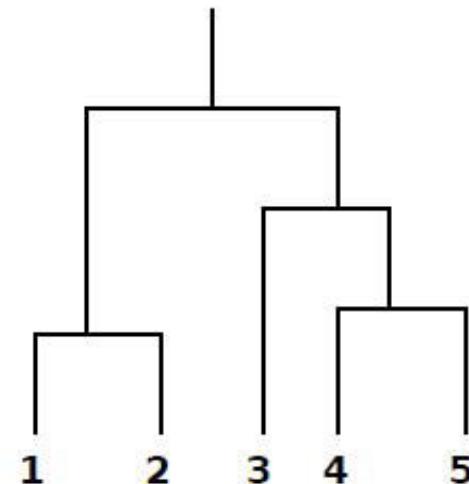
- **Complete-link distance** between clusters Ci and Cj is the **maximum distance** between any object in Ci, and any object in Cj;
- The distance is **defined by the two most dissimilar objects**

$$D_{\text{complete}} = \max_{x,y} \{d(x, y) \mid x \in C_i, y \in C_j\}$$

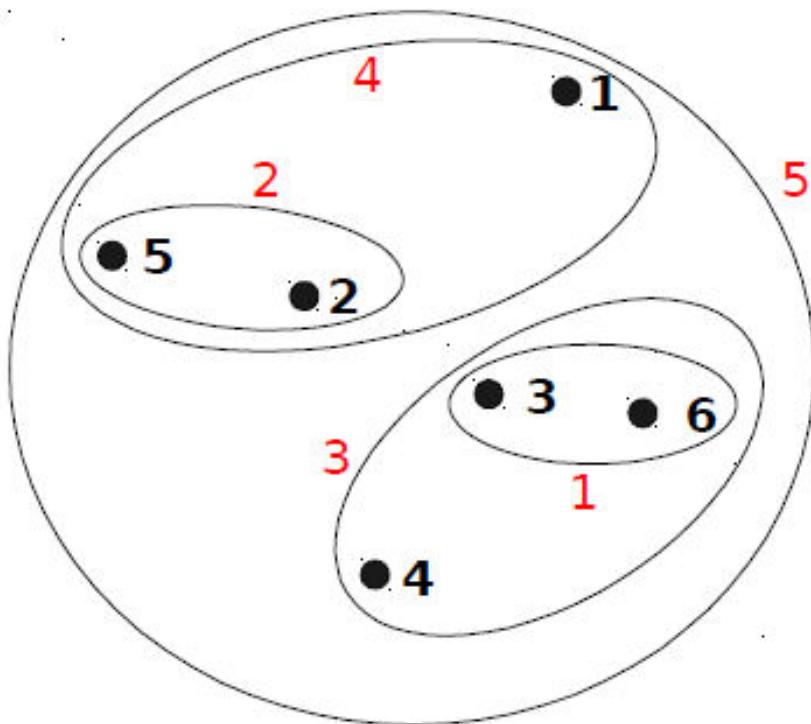
Complete-link clustering example

- Distance between clusters is determined by the two most distant points in the different clusters

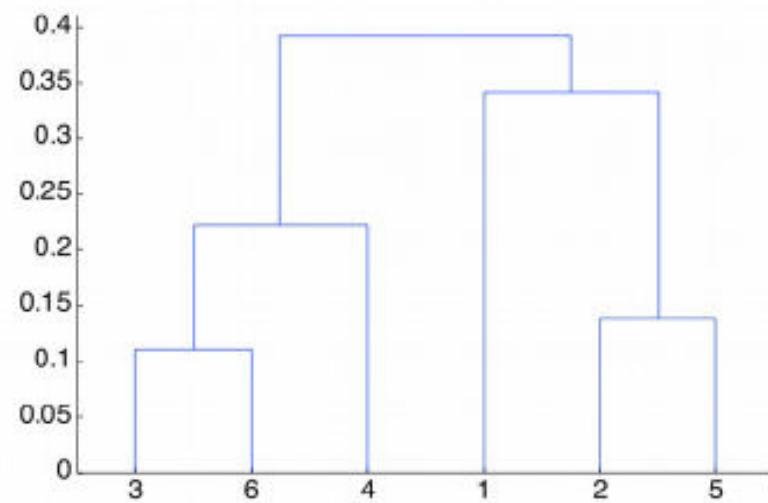
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link clustering: example

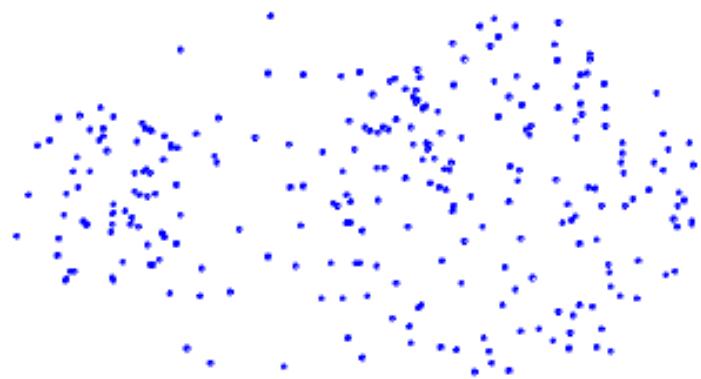


Nested Clusters

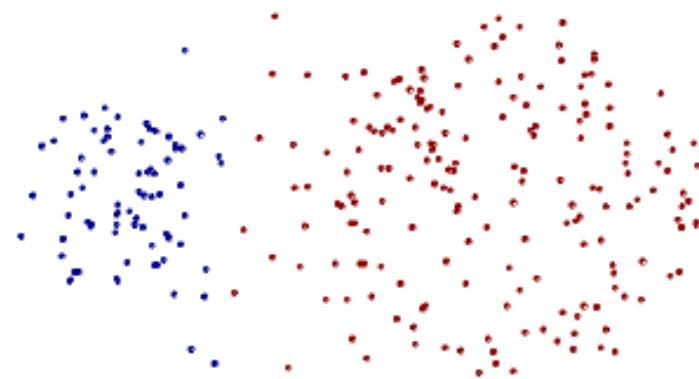


Dendrogram

Strengths of complete-link clustering



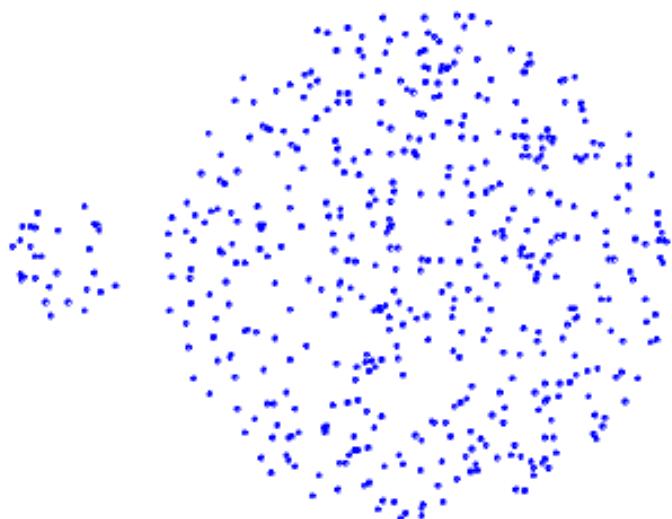
Original Points



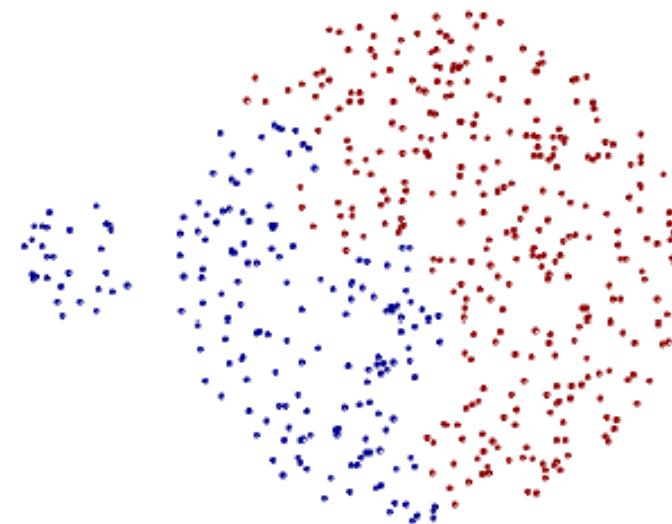
Two Clusters

- More balanced clusters (with equal diameter)
- Less susceptible to noise

Limitations of complete-link clustering



Original Points



Two Clusters

- Tends to break large clusters
- All clusters tend to have the same diameter – small clusters are merged with larger ones

Distance between two clusters

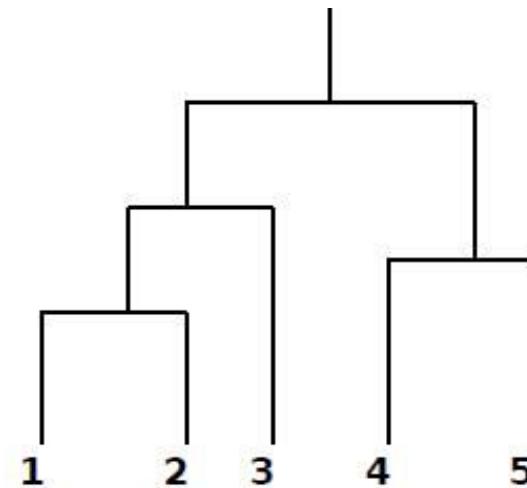
- Group average distance between clusters Ci
- and Cj is the average distance between any object in Ci and any object in Cj

$$D_{\text{average}} = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

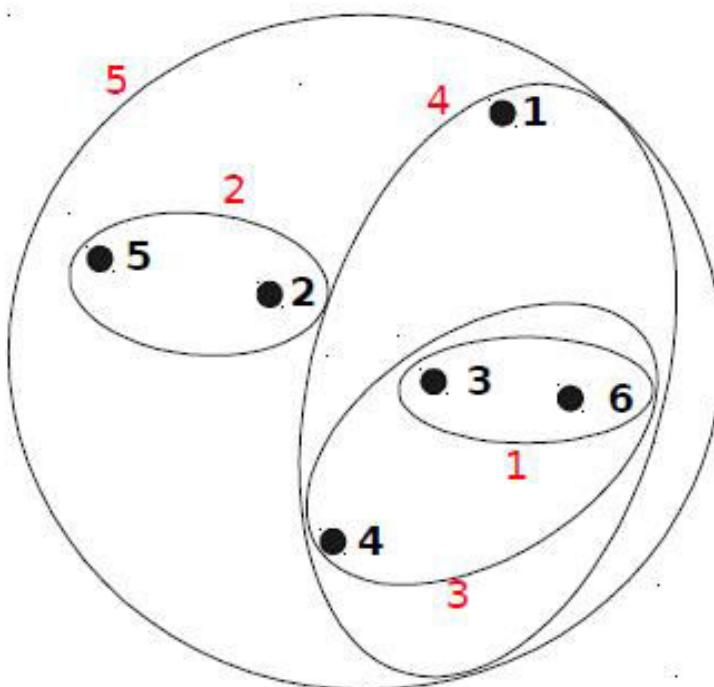
Average-link clustering example

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

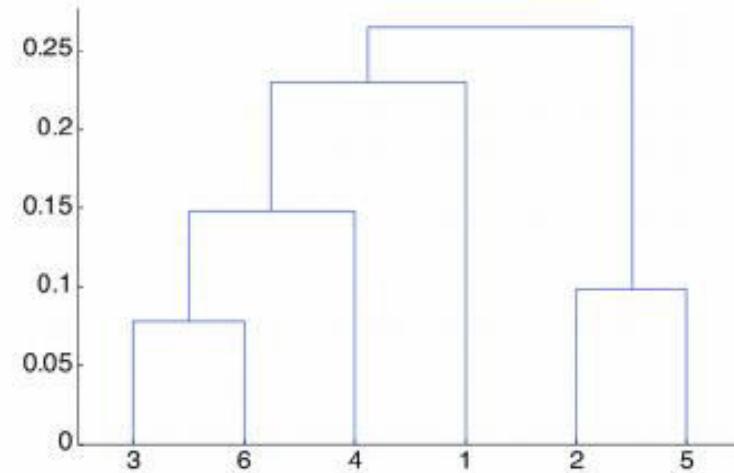
I1	I2	I3	I4	I5	
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link clustering: example



Nested Clusters



Dendrogram

Average-link clustering discussion

- Compromise between Single and Complete Link
- **Strengths**
 - Less susceptible to noise and outliers
- **Limitations**
 - Biased towards globular clusters

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{\text{centroids}}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- **Ward's distance** between clusters C_i and C_j is the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters in cluster C_{ij}

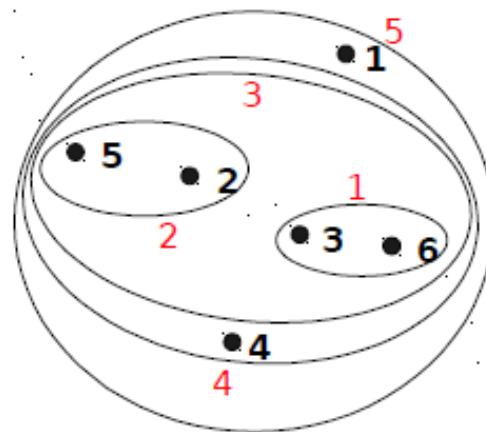
$$D_W(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i : centroid of C_i
- r_j : centroid of C_j
- r_{ij} : centroid of C_{ij}

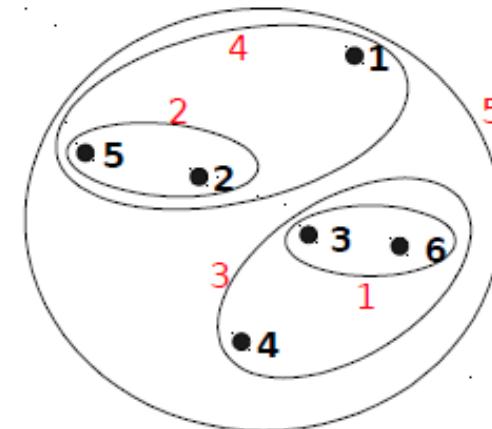
Ward's distance for clusters

- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
 - Can be used to initialize k-means

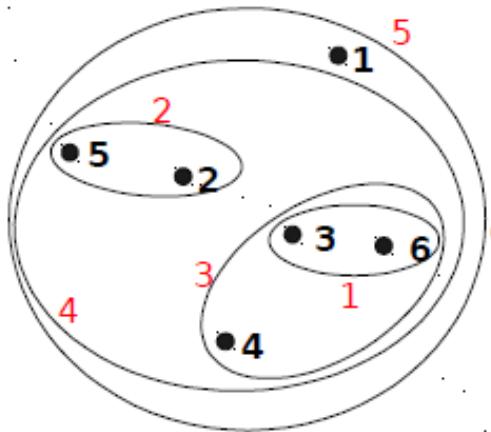
Hierarchical Clustering comparison



MIN

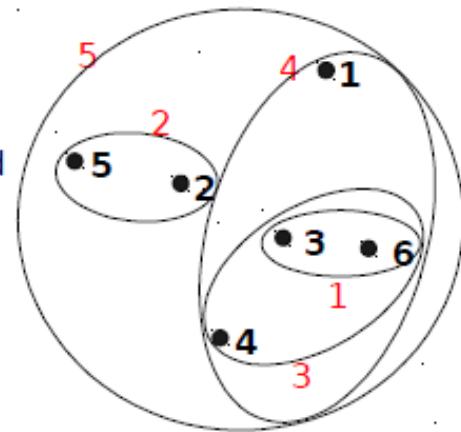


MAX



Group Average

Ward's Method



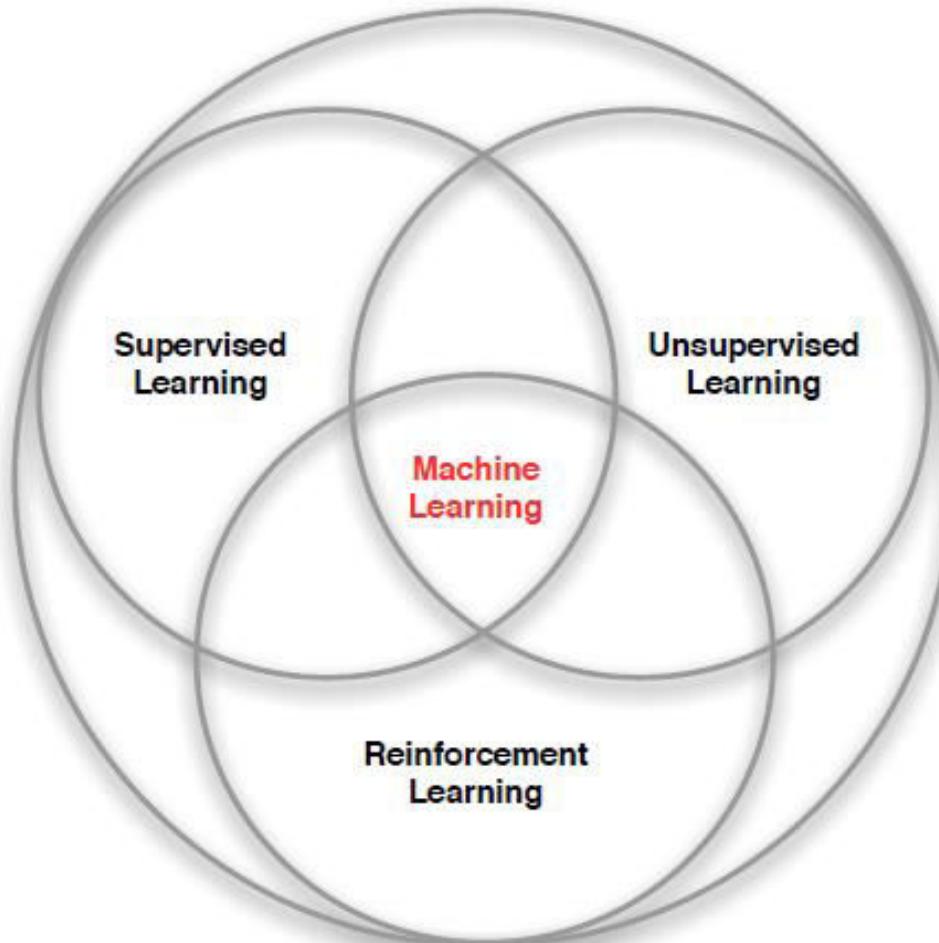
Hierarchical Clustering: Time and Space requirements

- For a dataset X consisting of n points
- $O(n^2)$ space; it requires storing the distance matrix
- $O(n^3)$ time in most of the cases
 - There are n steps and at each step the size n^2 distance matrix must be updated and searched
 - Complexity can be reduced to $O(n^2 \log(n))$ time for some approaches by using appropriate data structures

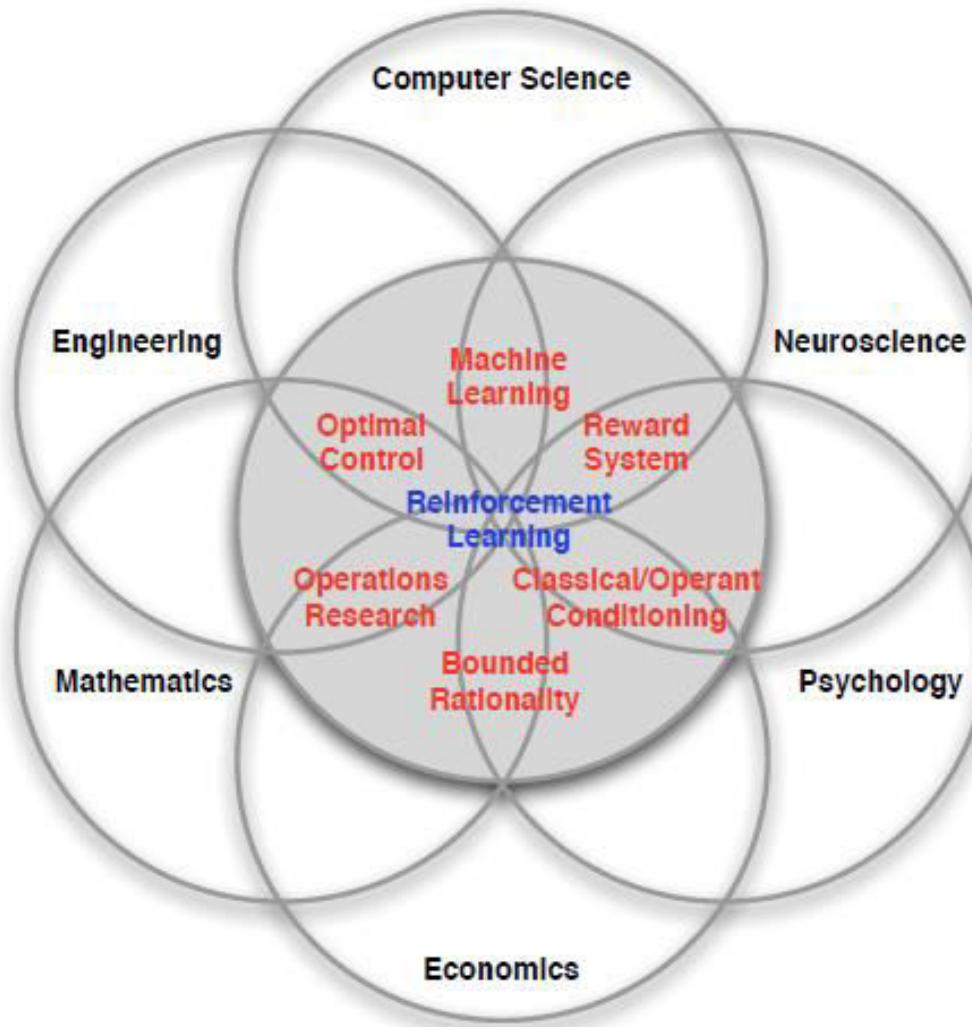
Lecture 11 – Introduction to Reinforcement learning

Branches of machine learning

Branches of Machine Learning



Many Faces of Reinforcement Learning



Characteristics of RF

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Examples of RF - Reinforcement Learning

- Fly stunt maneuvers in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different Atari games better than humans

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Reinforcement learning is based on the **reward hypothesis**

- Definition (Reward Hypothesis) : All goals can be described by the maximization of expected cumulative reward

Examples of Rewards

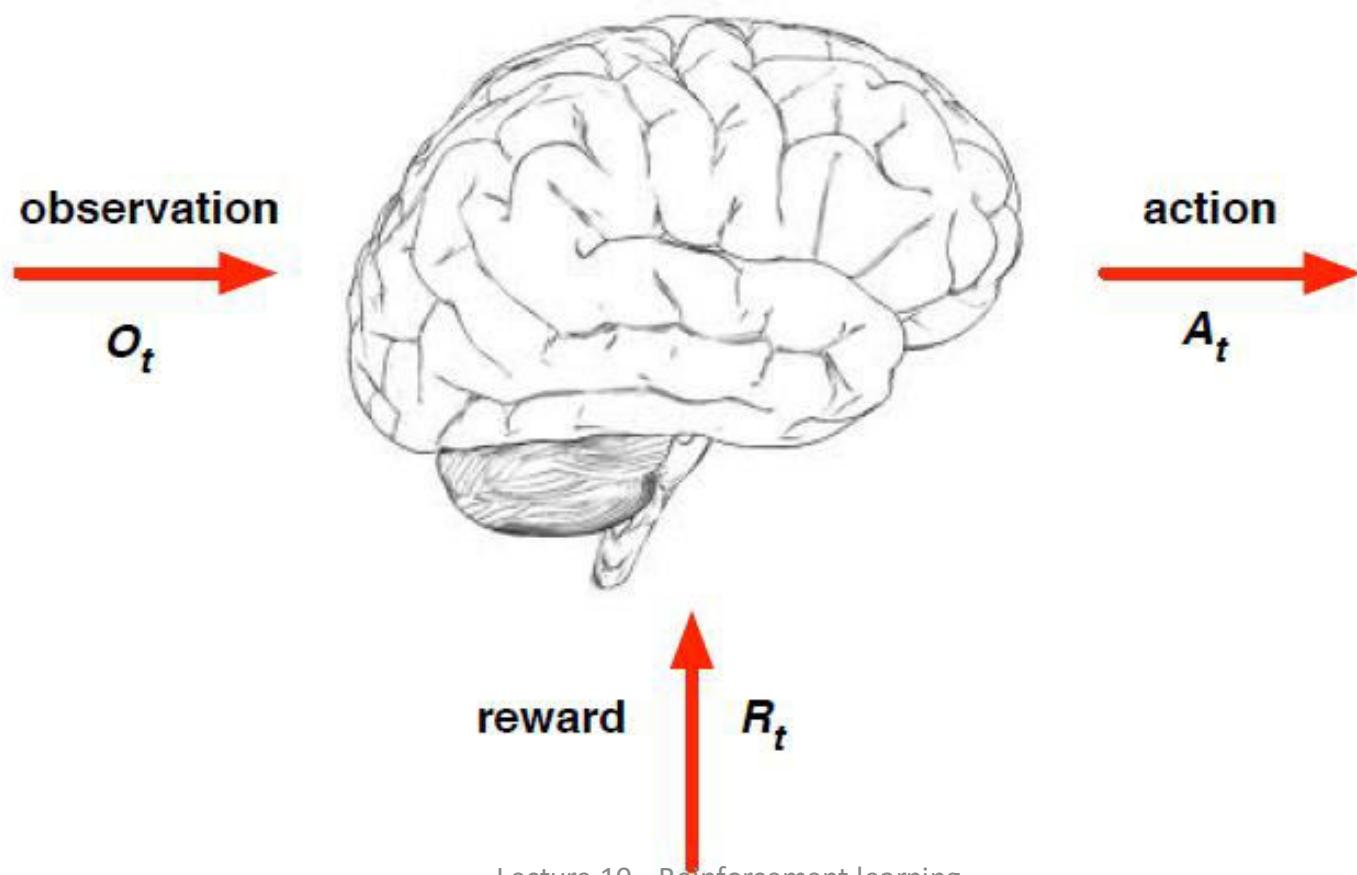
- Fly stunt manoeuvres in a helicopter
 - +ve reward for following desired trajectory
 - -ve reward for crashing
- Defeat the world champion at Backgammon
 - +/-ve reward for winning/losing a game
- Manage an investment portfolio
 - +ve reward for each \$ in bank
- Control a power station
 - +ve reward for producing power
 - -ve reward for exceeding safety thresholds
- Make a humanoid robot walk
 - +ve reward for forward motion
 - -ve reward for falling over
- Play many different Atari games better than humans
 - +/-ve reward for increasing/decreasing score

Sequential Decision Making

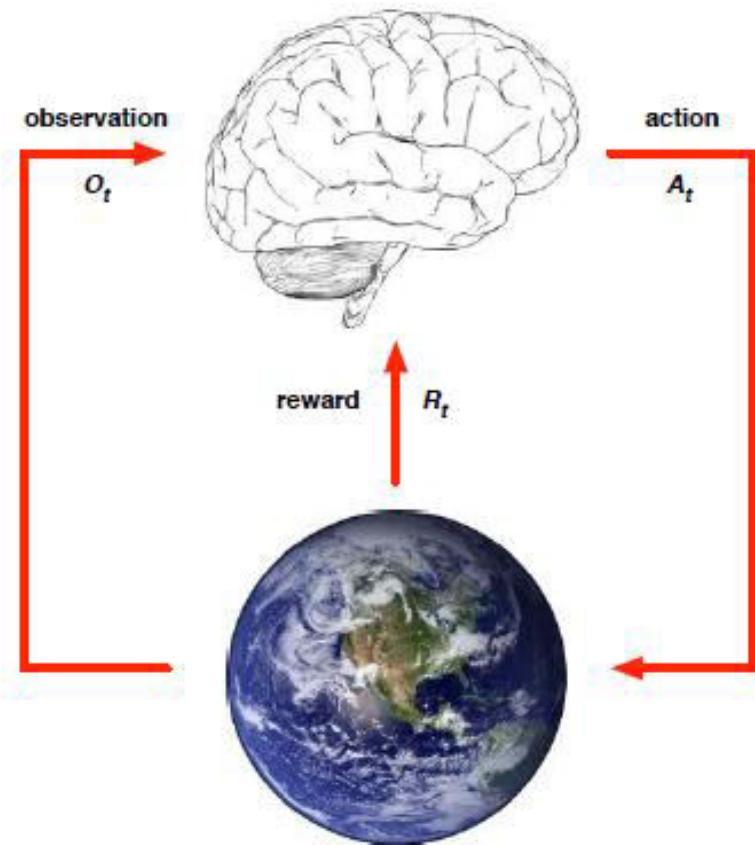
- Goal: select actions to maximise total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
 - A financial investment (may take months to mature)
 - Refueling a helicopter (might prevent a crash in several hours)
 - Blocking opponent moves (might help winning chances many moves from now)

Agent and Environment

Agent and Environment



Agent and Environment



- At each step t the agent:
 - Executes action A_t
 - Receives observation O_t
 - Receives scalar reward R_t
- The environment:
 - Receives action A_t
 - Emits observation O_{t+1}
 - Emits scalar reward R_{t+1}
- t increments at env. step

History and State

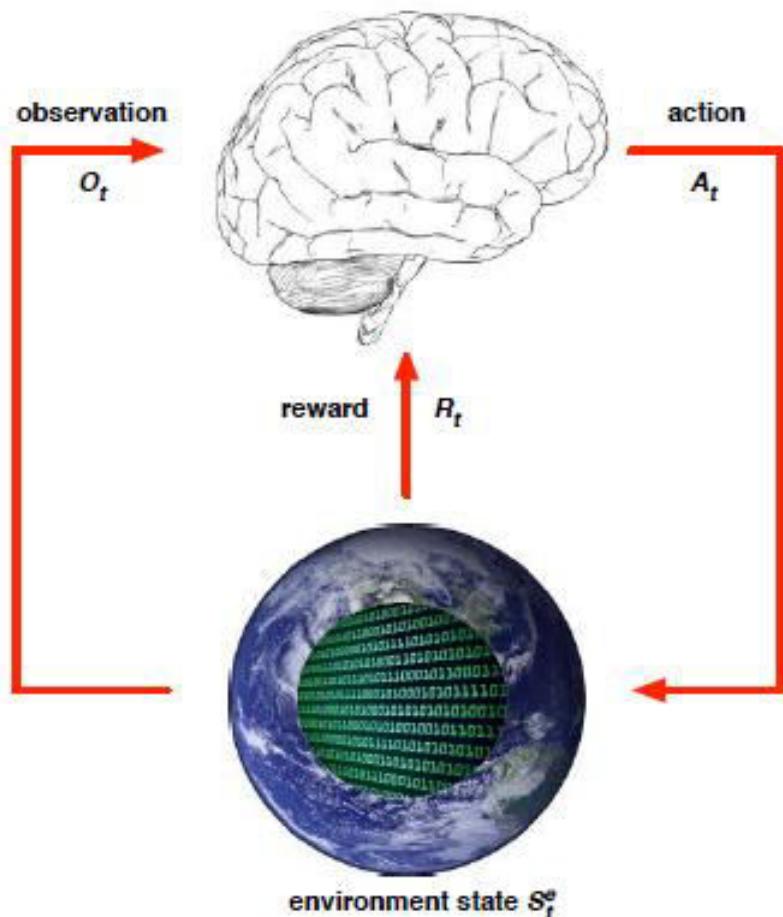
- The **history** is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_t, R_t$$

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
 - The agent selects actions
 - The environment selects observations/rewards
- **State** is the information used to determine what happens next
- Formally, state is a function of the history: $S_t = f(H_t)$

Environment State

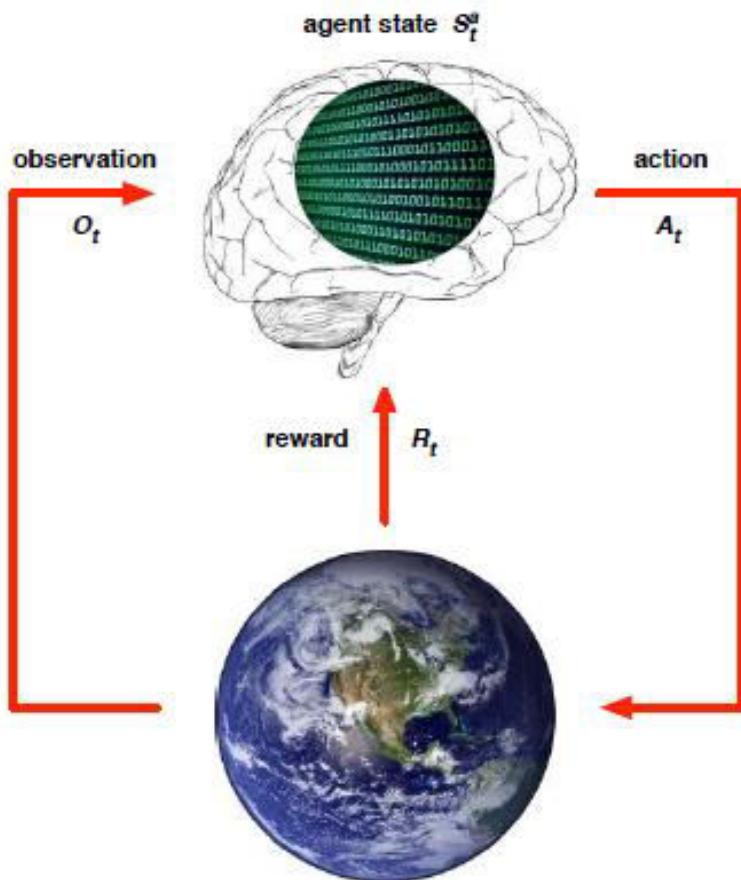
Environment State



- The **environment state** S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

Agent State

Agent State



- The **agent state** S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

Information State

Information State

An **information state** (a.k.a. **Markov state**) contains all useful information from the history.

Definition

A state S_t is **Markov** if and only if

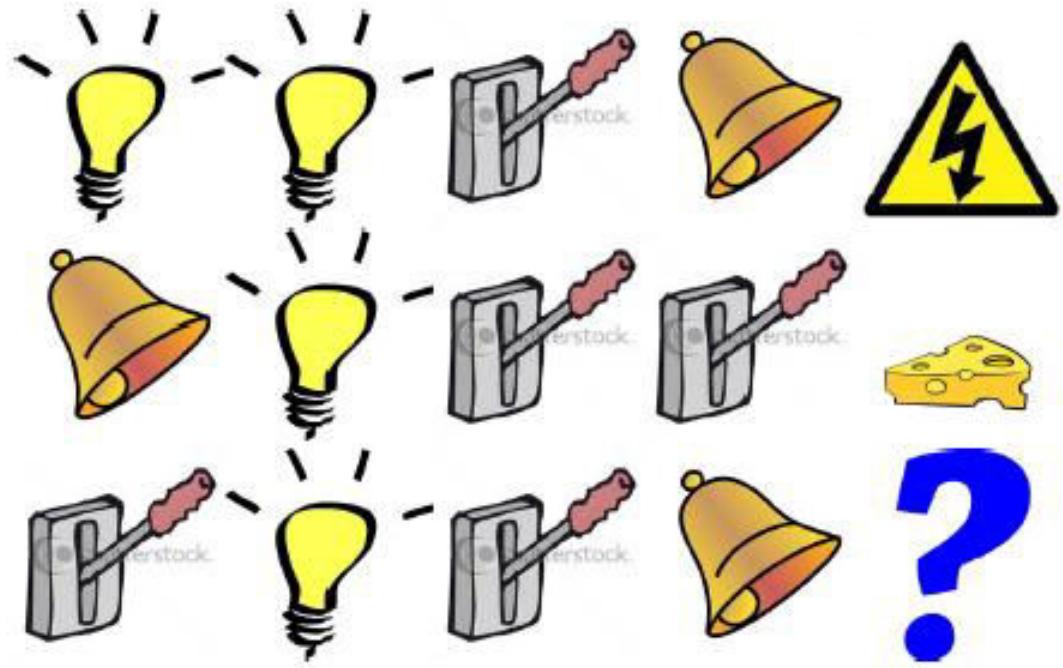
$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \dots, S_t]$$

- “The future is independent of the past given the present”

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

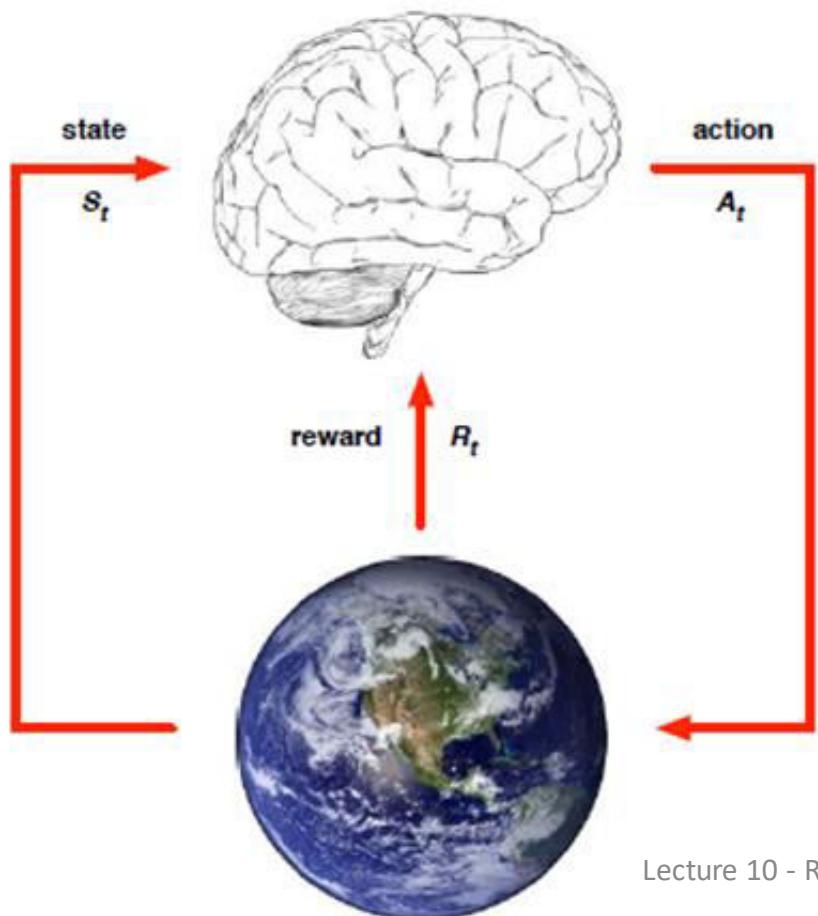
Rat Example



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Fully Observable Environment Markov

Fully Observable Environments



Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

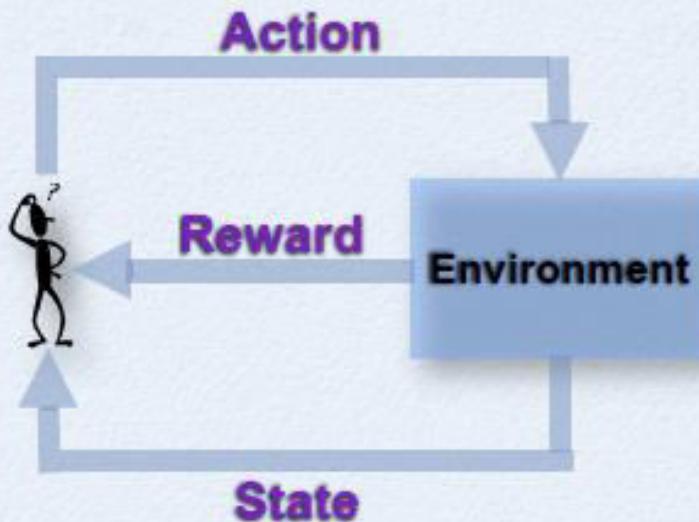
- Agent state = environment state = information state
- Formally, this is a **Markov decision process (MDP)**

Partially Observable Environment POM

Partially Observable Environments

- Partial observability: agent **indirectly** observes environment:
 - A robot with camera vision isn't told its absolute location
 - A trading agent only observes current prices
 - A poker playing agent only observes public cards
- Now agent state \neq environment state
- Formally this is a **partially observable Markov decision process** (POMDP)
- Agent must construct its own state representation S_t^a , e.g.
 - Complete history: $S_t^a = H_t$
 - **Beliefs** of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], \dots, \mathbb{P}[S_t^e = s^n])$
 - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

REINFORCEMENT LEARNING



- **RL:** A class of learning problems in which an agent interacts with an unfamiliar, dynamic and stochastic environment
- **Goal:** Learn a policy to maximize some measure of long-term reward
- **Interaction:** Modeled as a MDP or a POMDP

Components of RL Agent

An RL agent may include one or more of these components:

- **Policy**: agent's behavior function
- **Value function**: how good is each state and/or action
- **Model**: agent's representation of the environment

Policy

Policy

- A **policy** is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

Value Function

Value Function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

Model

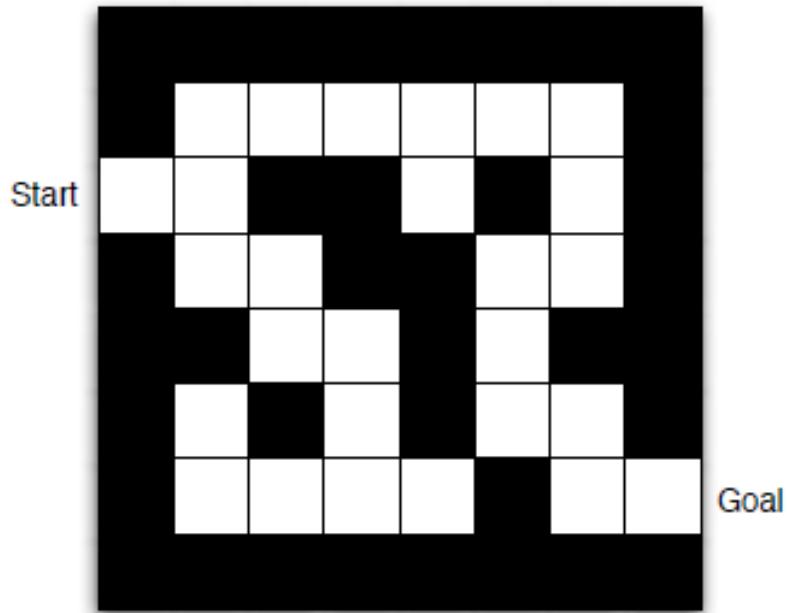
Model

- A **model** predicts what the environment will do next
- \mathcal{P} predicts the next state
- \mathcal{R} predicts the next (immediate) reward, e.g.

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

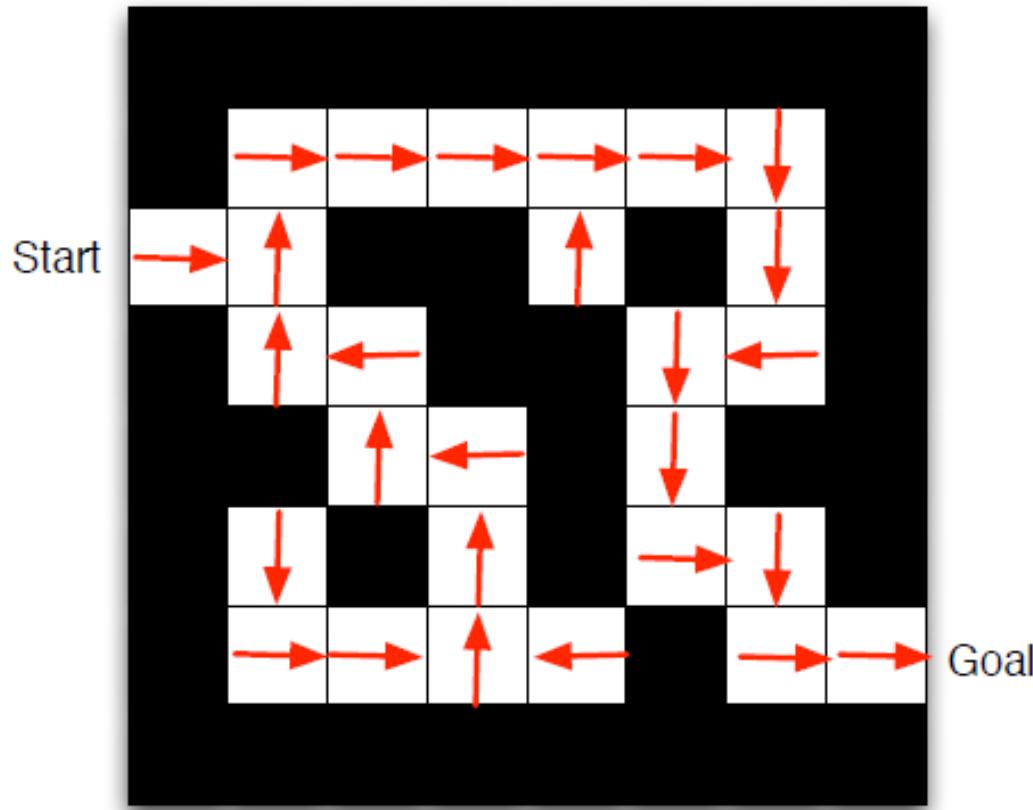
$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Maze Example



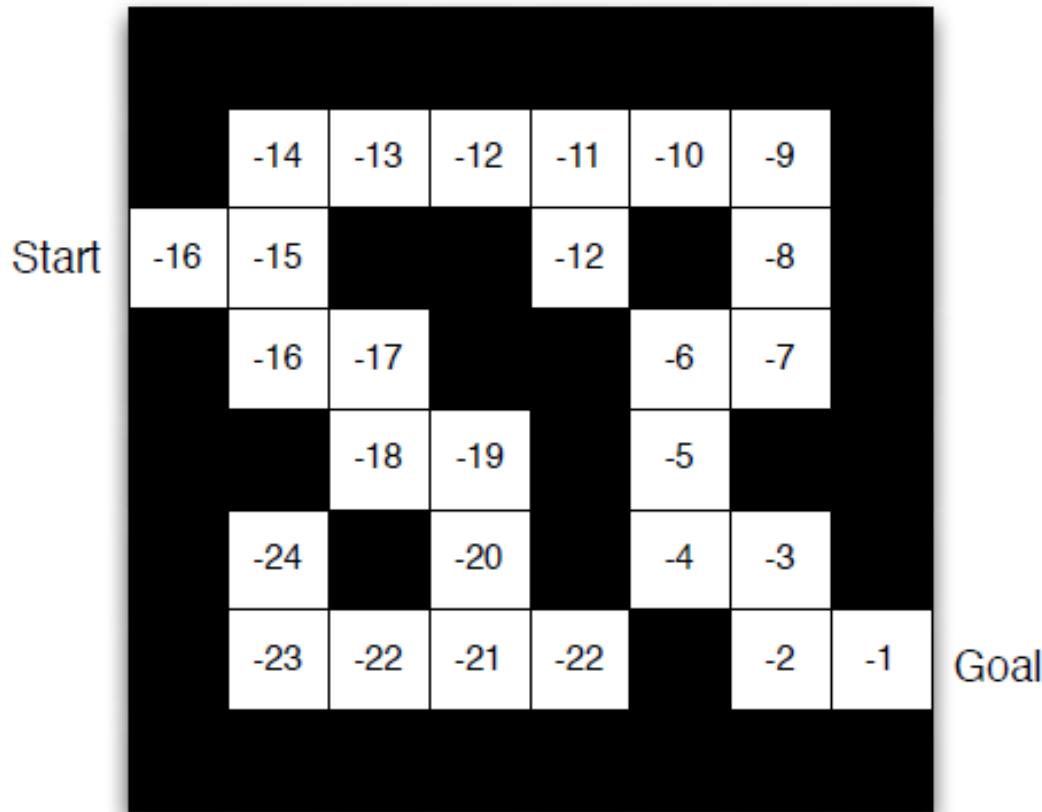
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



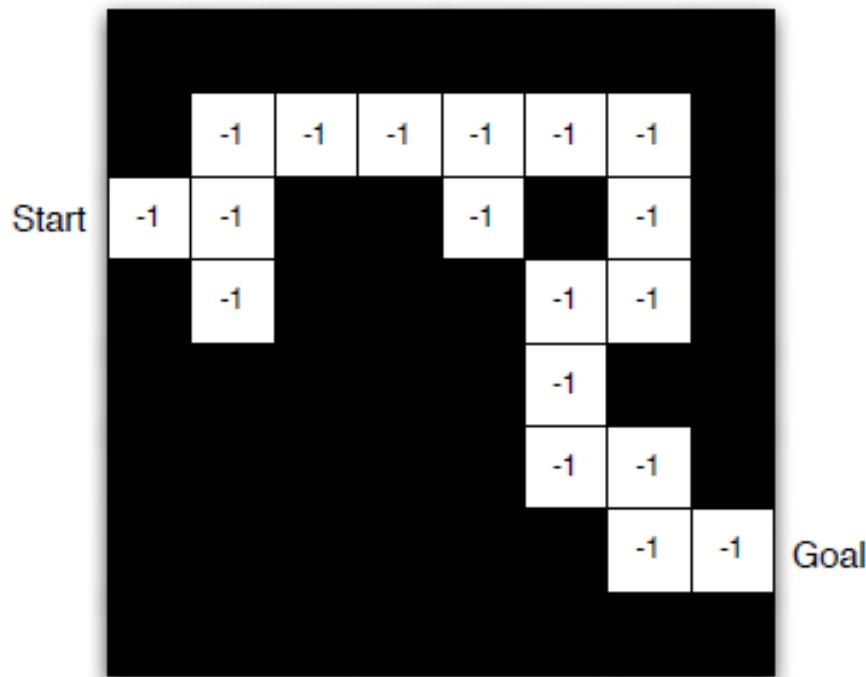
- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



- Numbers represent value $v_\pi(s)$ of each state s

Maze Example: Model



- Agent may have an internal model of the environment
 - Dynamics: how actions change the state
 - Rewards: how much reward from each state
 - The model may be imperfect
-
- Grid layout represents transition model $\mathcal{P}_{ss'}^a$,
 - Numbers represent immediate reward \mathcal{R}_s^a from each state s (same for all a)

Categories of RL Agents (1)

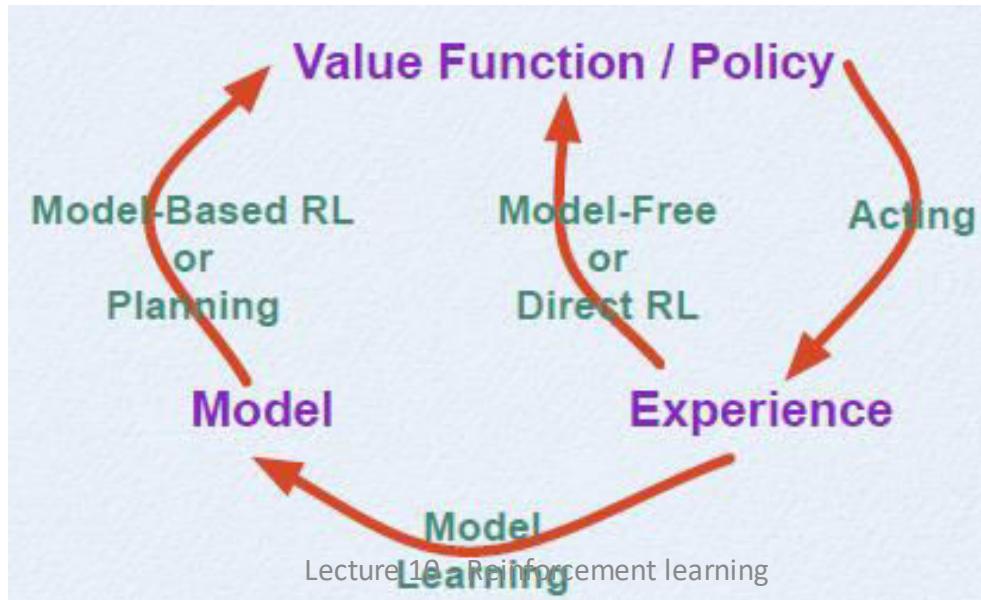
- Value Based
 - No Policy (Implicit)
 - Value Function
- Policy Based
 - Policy
 - No Value Function
- Actor Critic
 - Policy
 - Value Function

Categories of RL Agents (2)

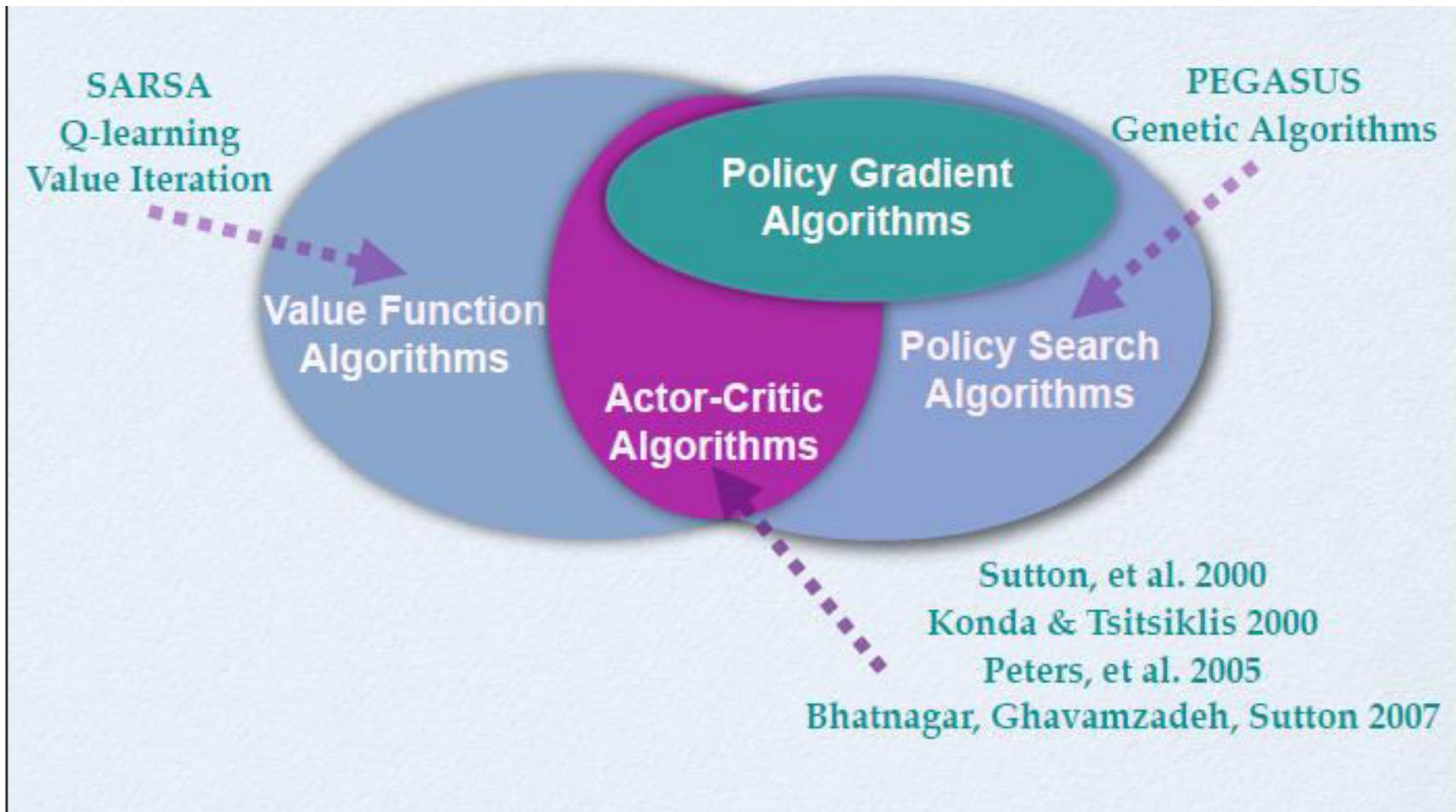
- Model Free
 - Policy and/or Value Function
 - No Model
- Model Based
 - Policy and/or Value Function
 - Model

MODEL BASED vs. MODEL FREE RL

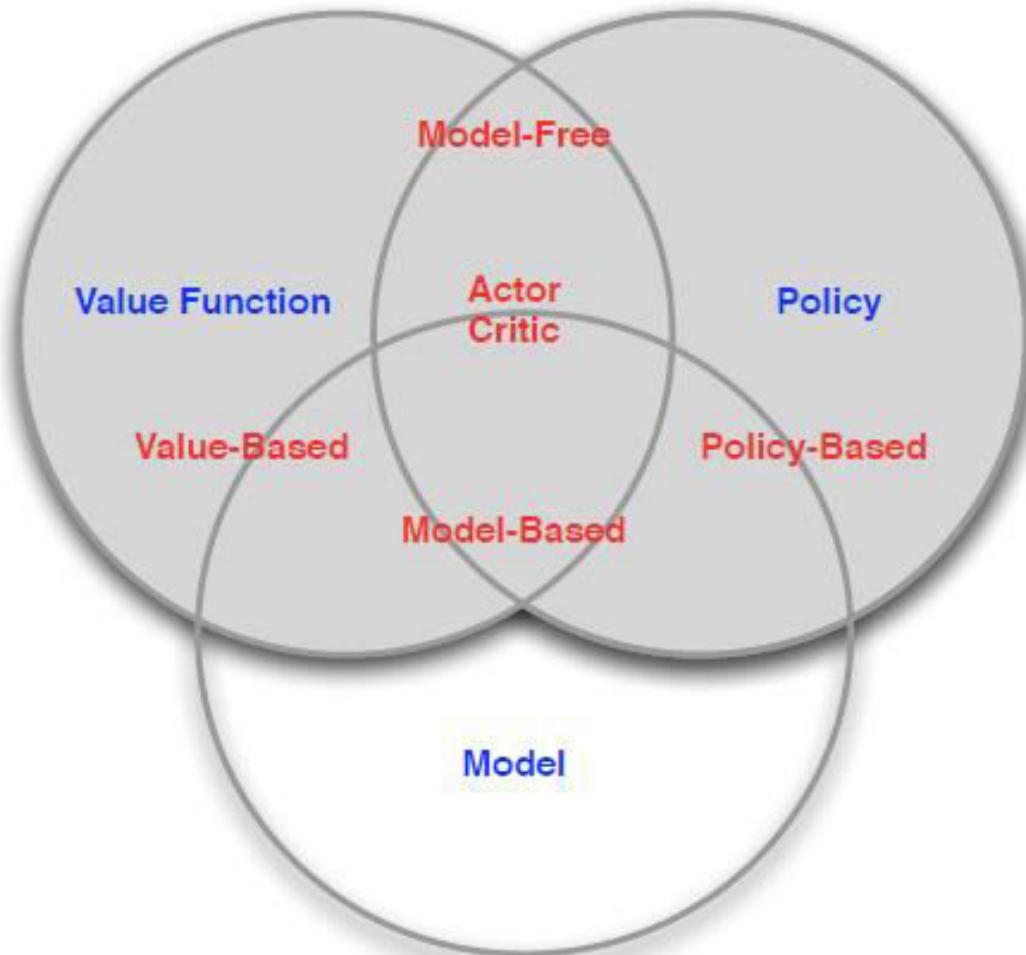
- What is model?
 - state transition distribution and reward distribution
- Model Based RL:
 - model is not available, but it is explicitly learned
- Model Free RL:
 - model is not available and is not explicitly learned



Reinforcement Learning Solutions



RL Agent Taxonomy



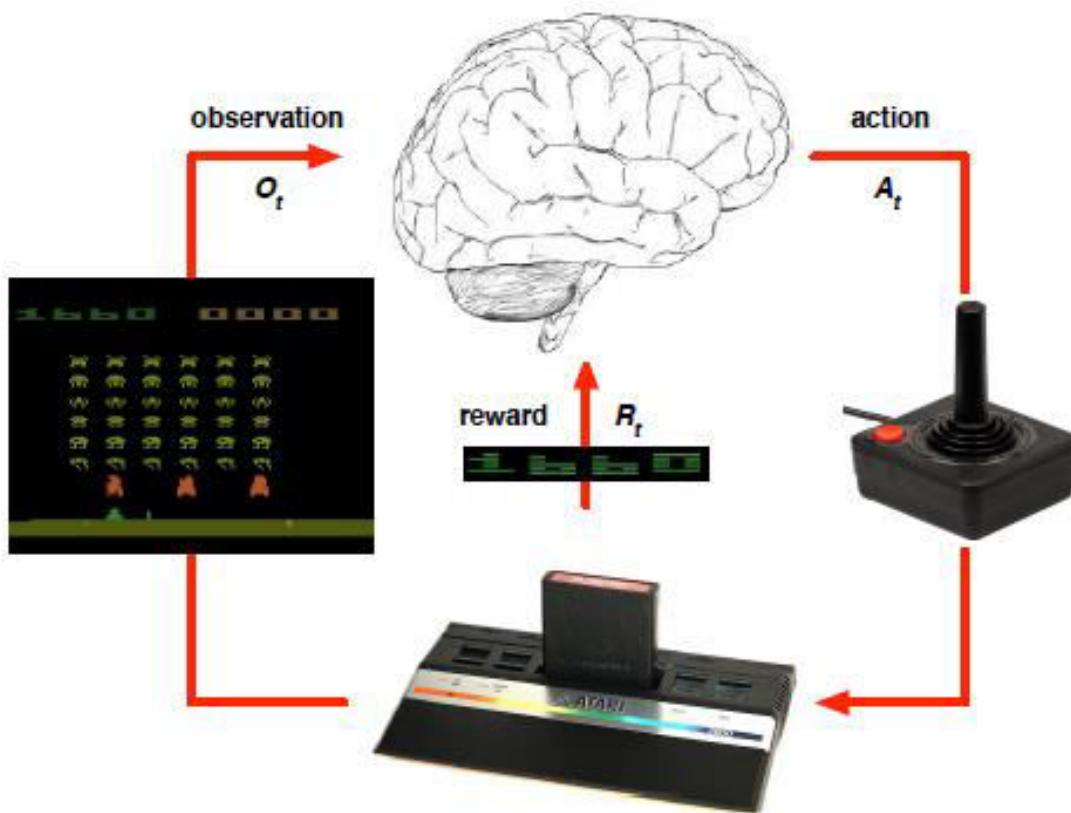
Learning and Planning

Two fundamental problems in sequential decision making

- Reinforcement Learning:
 - The environment is initially unknown
 - The agent interacts with the environment
 - The agent improves its policy
- Planning:
 - A model of the environment is known
 - The agent performs computations with its model (without any external interaction)
 - The agent improves its policy
 - aka. deliberation, reasoning, introspection, pondering, thought, search

RL Learning

Atari Example: Reinforcement Learning

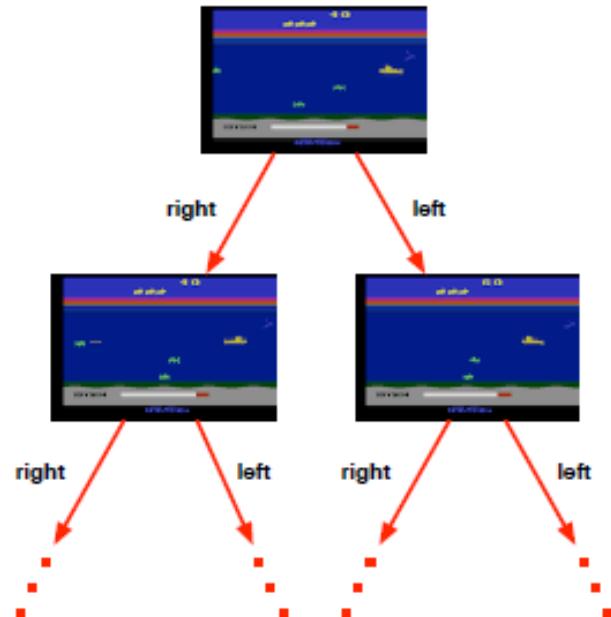


- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Planning

Atari Example: Planning

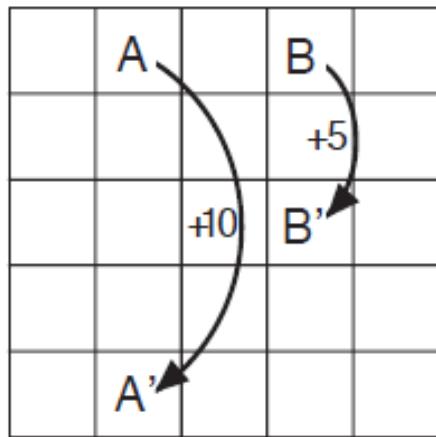
- Rules of the game are known
- Can query emulator
 - perfect model inside agent's brain
- If I take action a from state s :
 - what would the next state be?
 - what would the score be?
- Plan ahead to find optimal policy
 - e.g. tree search



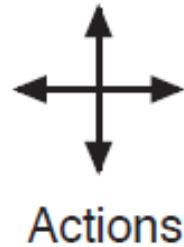
Prediction and Control

- Prediction: evaluate the future
 - Given a policy
- Control: optimize the future
 - Find the best policy

Gridworld Example: Prediction



(a)

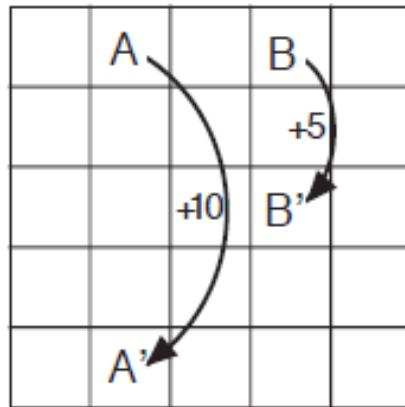


3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

What is the value function for the uniform random policy?

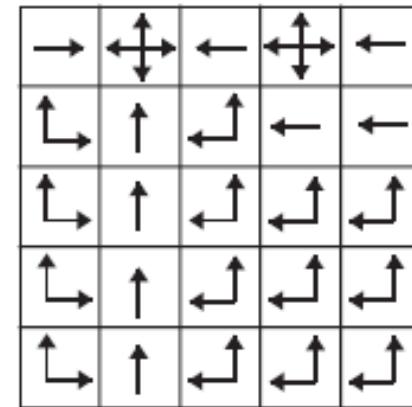
Gridworld Example: Control



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*



c) π_*

What is the optimal value function over all possible policies?
What is the optimal policy?

Exploration and Exploitation (1)

- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way

Exploration and Exploitation (2)

- Exploration finds more information about the environment
- Exploitation exploits known information to maximize reward
- It is usually important to explore as well as exploit

Exploration and Exploitation examples

- Restaurant Selection
 - Exploitation Go to your favourite restaurant
 - Exploration Try a new restaurant
- Online Banner Advertisements
 - Exploitation Show the most successful advert
 - Exploration Show a different advert
- Oil Drilling
 - Exploitation Drill at the best known location
 - Exploration Drill at a new location
- Game Playing
 - Exploitation Play the move you believe is best
 - Exploration Play an experimental move

Learning Modes

- Offline Learning
 - Learning while interacting with a simulator
- Online Learning
 - Learning while interacting with the environment

Offline Learning

- Agent interacts with a simulator
- Rewards/costs do not matter
 - no exploration/exploitation tradeoff
- Computation time between actions is not critical
- Simulator can produce as much as data we wish
- Main Challenge
 - How to minimize time to converge to optimal policy

Online Learning

- No simulator - Direct interaction with environment
- Agent receives reward/cost for each action
- Main Challenges
 - Exploration/exploitation tradeoff
 - Should actions be picked to maximize immediate reward or to maximize information gain to improve policy
 - Real-time execution of actions
 - Limited amount of data since interaction with environment is required

Q learning

- $Q[s,a]$
 - s - state that we are in
 - a - action we can take
 - Q - value of taking the action a at state s

$$Q[s,a] = \text{immediate reward} + \text{discounted reward}$$

How to use Q?

- $\Pi(s) = \operatorname{argmax}_a(Q[s,a])$
- $\Pi^*(s) = Q^*[s,a]$

Q learning procedure - details

Q learning – update rule

Q learning - Exploration

Q learning procedure

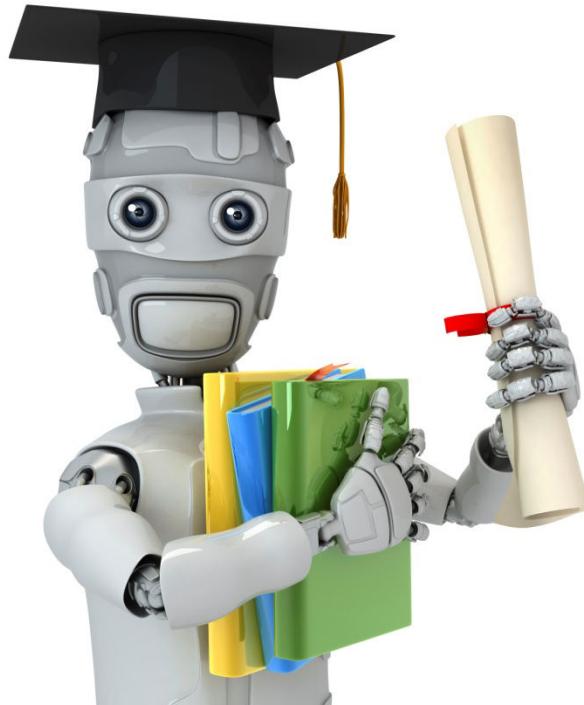
- Interact with the environment and identify the current state – s
- Iterate over time $\langle s, a, s', r \rangle$
- Update the Q values
- Repeat until converge

Deep Reinforcement learning

- Combining Deep Neural nets with Reinforcement learning
- E.g. Deep Q learning (have a Deep NN instead of a matrix)

Summary

- RL is used when labeled or unlabeled data is not available
- Based on finding the optimum action a , that gives the maximum reward r at a state s
- Model, Value function, Policy
- Q learning as a value iteration, model free algorithm
- Can be combined with other learning algorithms (e.g. Deep Q learning)



Machine Learning

Anomaly detection

Problem
motivation

Anomaly detection example

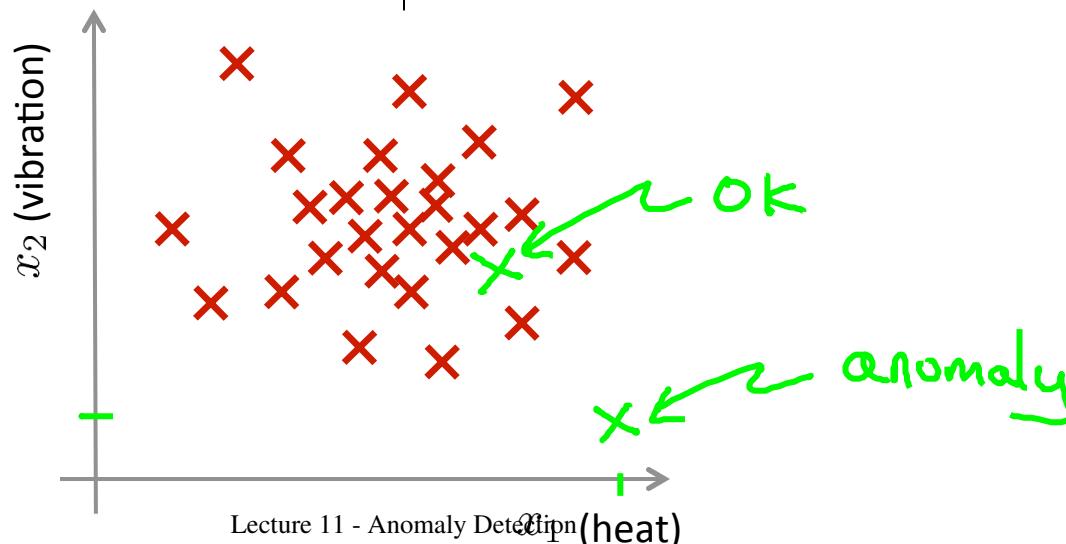
Aircraft engine features:

- x_1 = heat generated
- x_2 = vibration intensity

...

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

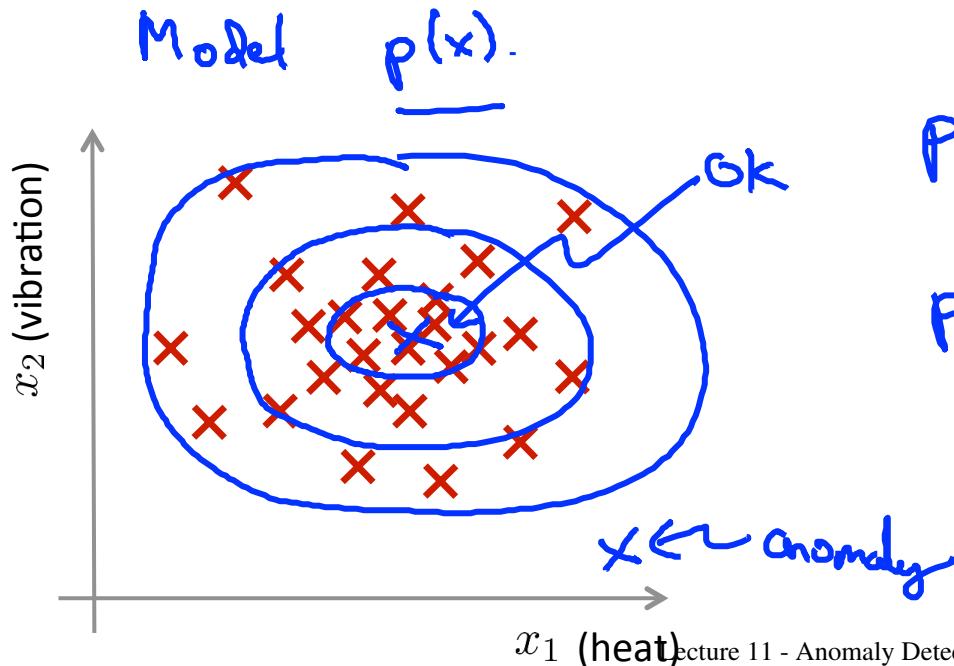
New engine: x_{test}



Density estimation

→ Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

→ Is x_{test} anomalous?



$p(x_{test}) < \varepsilon \rightarrow$ flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ OK

Anomaly detection example

→ Fraud detection:

→ $x^{(i)}$ = features of user i 's activities

→ Model $p(x)$ from data.

→ Identify unusual users by checking which have $p(x) < \varepsilon$

$$\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \quad p(x)$$

→ Manufacturing

→ Monitoring computers in a data center.

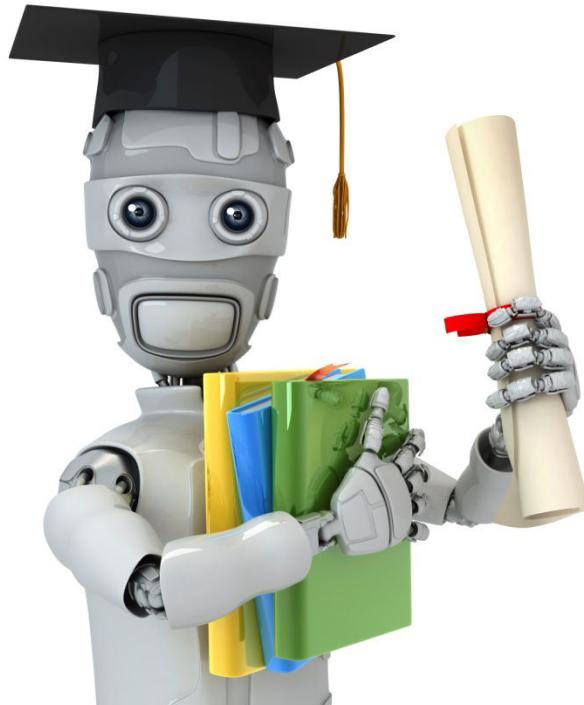
→ $x^{(i)}$ = features of machine i

x_1 = memory use, x_2 = number of disk accesses/sec,

x_3 = CPU load, x_4 = CPU load/network traffic.

...

$$p(x) < \varepsilon$$



Machine Learning

Anomaly detection

Gaussian distribution

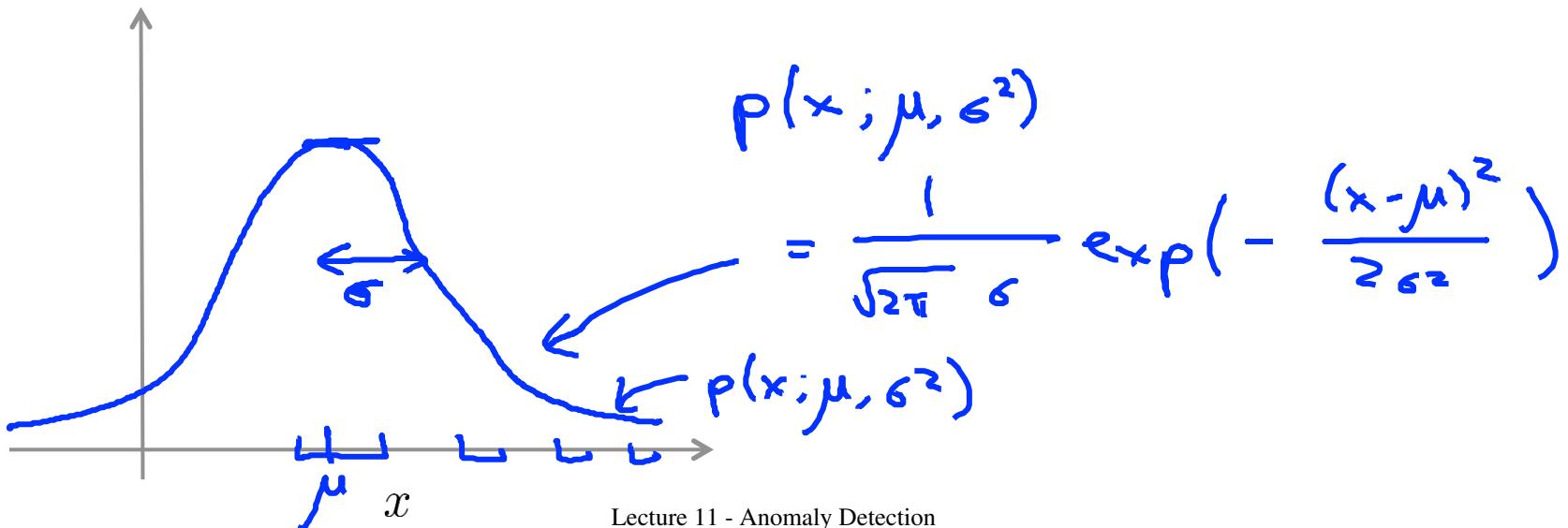
Gaussian (Normal) distribution

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ , variance σ^2 .

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

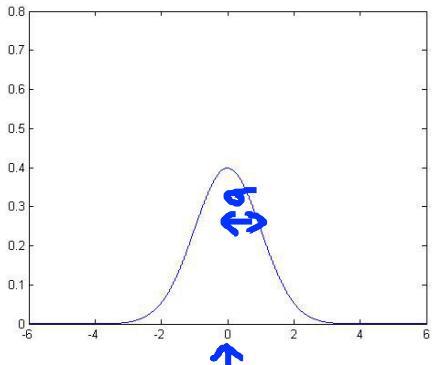
\curvearrowleft "distributed as"

σ standard deviation

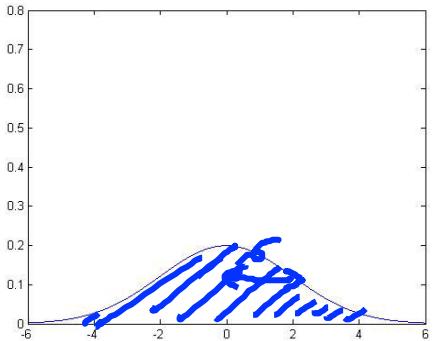


Gaussian distribution example

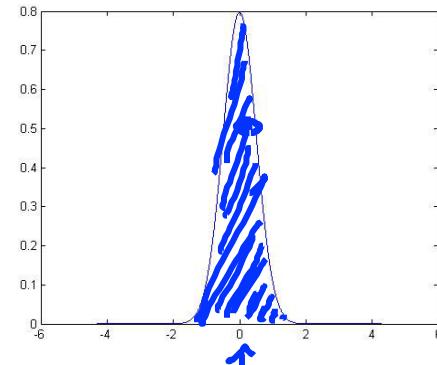
$$\rightarrow \mu = 0, \sigma = 1$$



$$\rightarrow \mu = 0, \sigma = 2$$

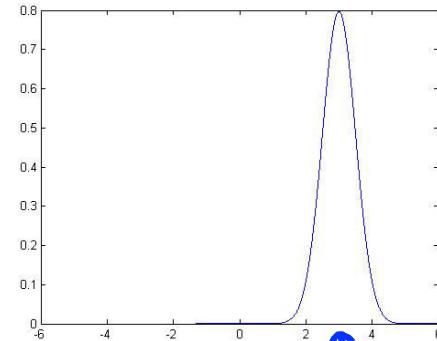


$$\rightarrow \mu = 0, \sigma = 0.5$$



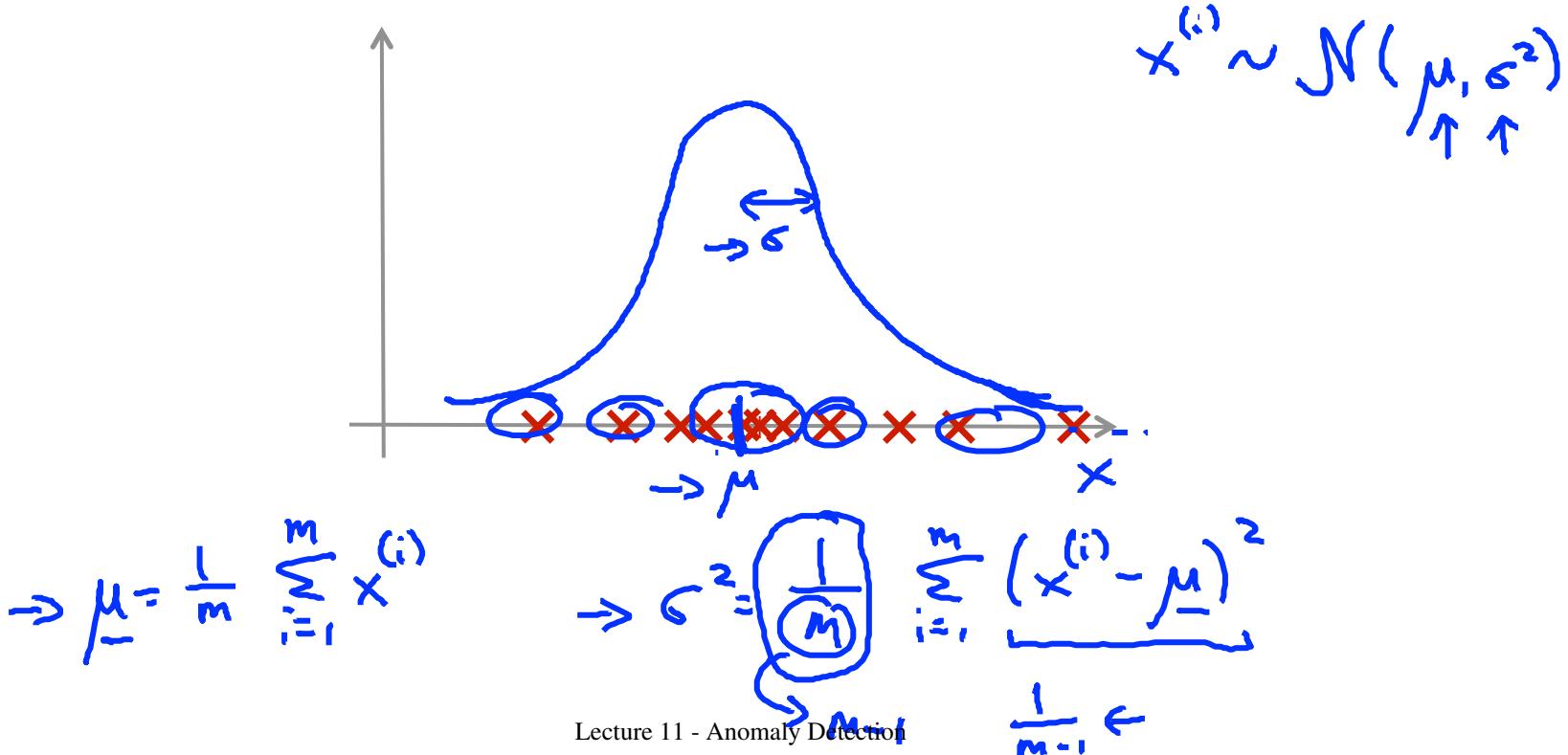
$$\zeta^2 = 0.25$$

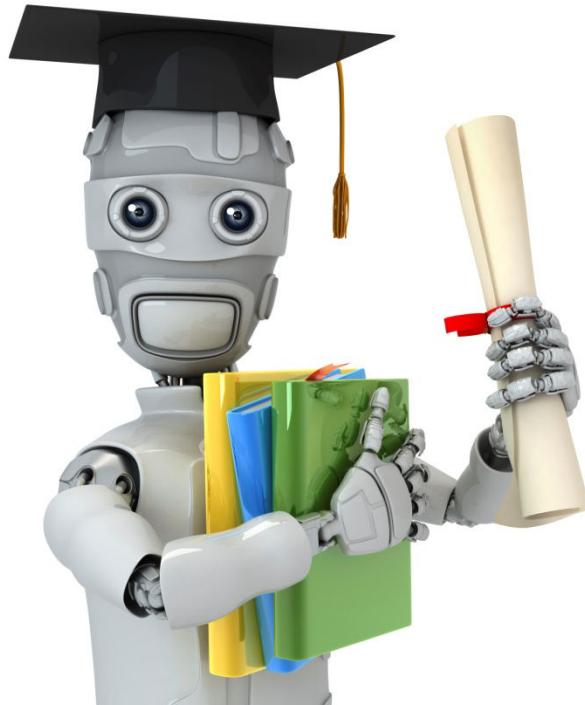
$$\rightarrow \mu = 3, \sigma = 0.5$$



Parameter estimation

→ Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$





Machine Learning

Anomaly detection

Algorithm

→ Density estimation

→ Training set: $\{x^{(1)}, \dots, x^{(m)}\}$

Each example is $x \in \mathbb{R}^n$

→ $p(x)$

$$= \boxed{p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2) \dots p(x_n; \mu_n, \sigma_n^2)}$$

$$= \boxed{\prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)}$$

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

$$x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$$

$$\sum_{i=1}^n i = 1+2+3+\dots+n$$

$$\prod_{i=1}^n i = 1 \times 2 \times 3 \times \dots \times n$$

Anomaly detection algorithm

- 1. Choose features x_i that you think might be indicative of anomalous examples.

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

- 2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\rightarrow \boxed{\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}}$$

$$p(x_j; \mu_j, \sigma_j^2)$$

$$\rightarrow \boxed{\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2}$$

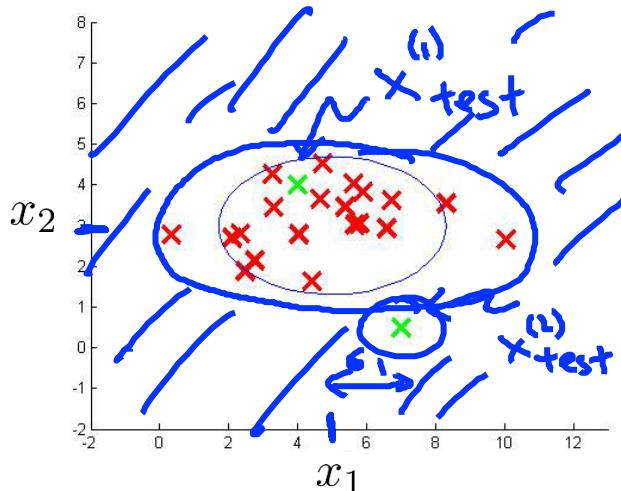
$$\begin{aligned} & \mu_1, \mu_2, \dots, \mu_n \\ & \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^m x^{(i)} \end{aligned}$$

- 3. Given new example x , compute $p(x)$:

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

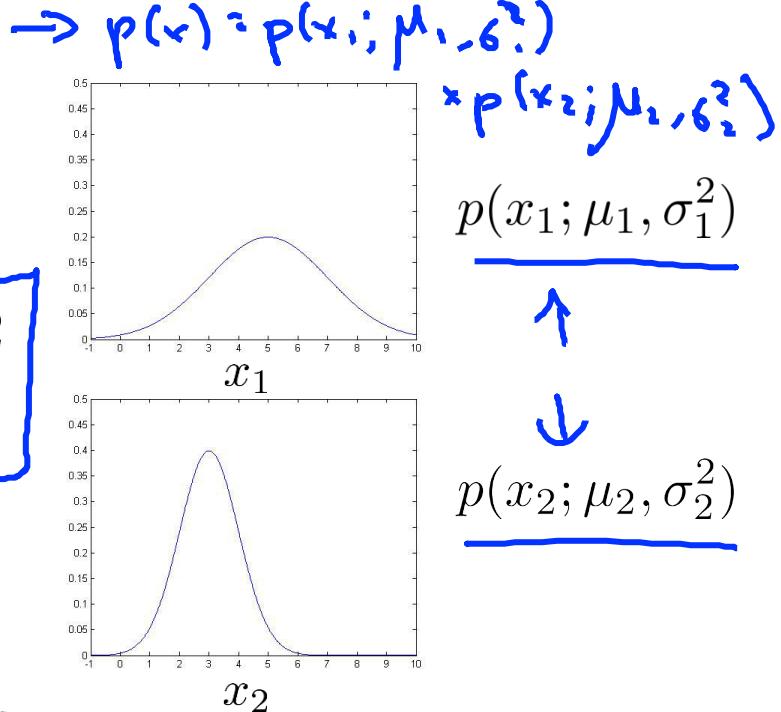
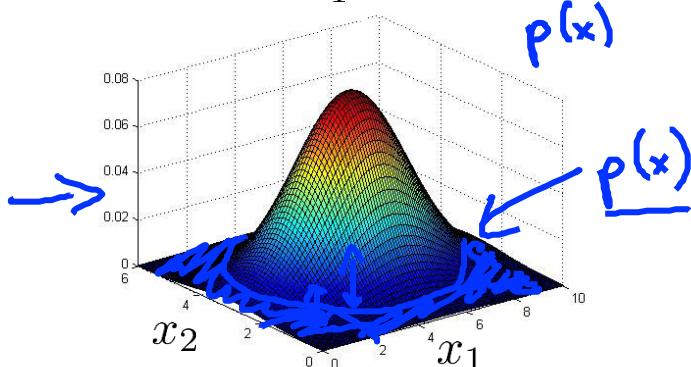
Anomaly if $\underline{p(x) < \varepsilon}$

Anomaly detection example



$$\begin{aligned} \mu_1 &= 5, \underline{\sigma_1} = 2 \\ \mu_2 &= 3, \underline{\sigma_2} = 1 \end{aligned}$$

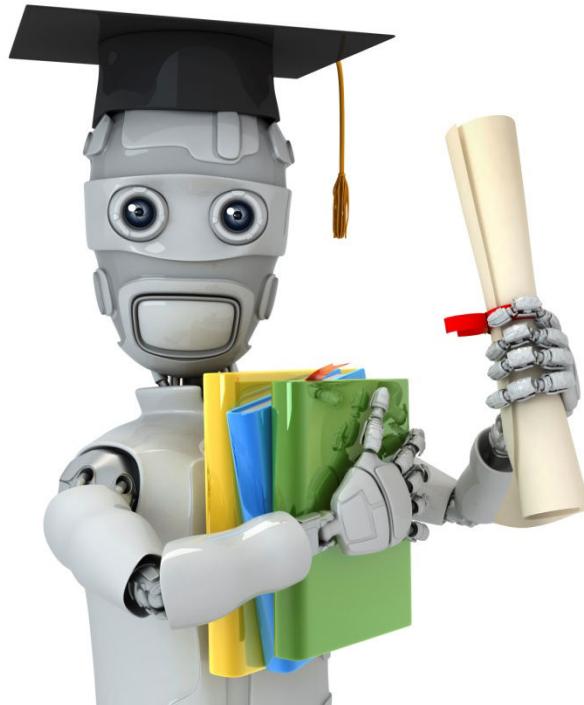
$\zeta^2 = 4$



$$\underline{\varepsilon = 0.02}$$

$$p(x_{test}^{(1)}) = \underline{0.0426} \geq \varepsilon$$

$$p(x_{test}^{(2)}) = \underline{0.0021} < \varepsilon$$



Machine Learning

Anomaly detection

Developing and
evaluating an anomaly
detection system

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

- Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).
- Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)
- Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
- Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

$$y = 1$$

Aircraft engines motivating example

- 10000 good (normal) engines
- 20 flawed engines (anomalous) 2 - 50 $y = 1$
- Training set: 6000 good engines ($y = 0$) $p(x) = p(x_1; \mu_1, \sigma^2_1) \dots p(x_n; \mu_n, \sigma^2_n)$
- CV: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)
- Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Alternative:

Training set: 6000 good engines

→ CV: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

→ Test: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Algorithm evaluation

- Fit model $p(x)$ on training set $\{x^{(1)}, \dots, x^{(m)}\}$
- On a cross validation/test example x , predict

$(x_{\text{test}}^{(i)}, y_{\text{test}}^{(i)})$



$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

$y=0$

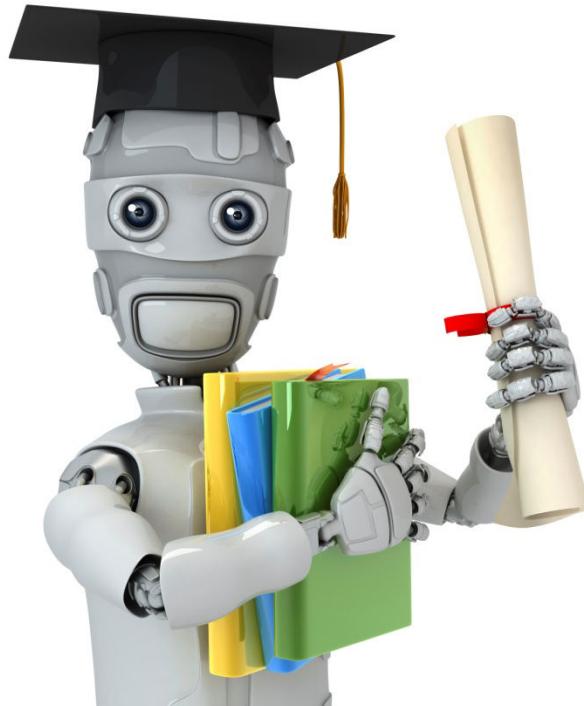
Possible evaluation metrics:

- - True positive, false positive, false negative, true negative
- - Precision/Recall
- - F_1 -score

CV

Test set

Can also use cross validation set to choose parameter ε



Machine Learning

Anomaly detection

Anomaly detection
vs. supervised
learning

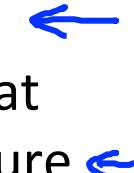
Anomaly detection

vs.

Supervised learning

- Very small number of positive examples ($y = 1$). (0-20 is common).
- Large number of negative ($y = 0$) examples. 
- Many different “types” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like;
- future anomalies may look nothing like any of the anomalous examples we've seen so far.

Large number of positive and negative examples. 

Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set. 

Spam 

Anomaly detection

vs.

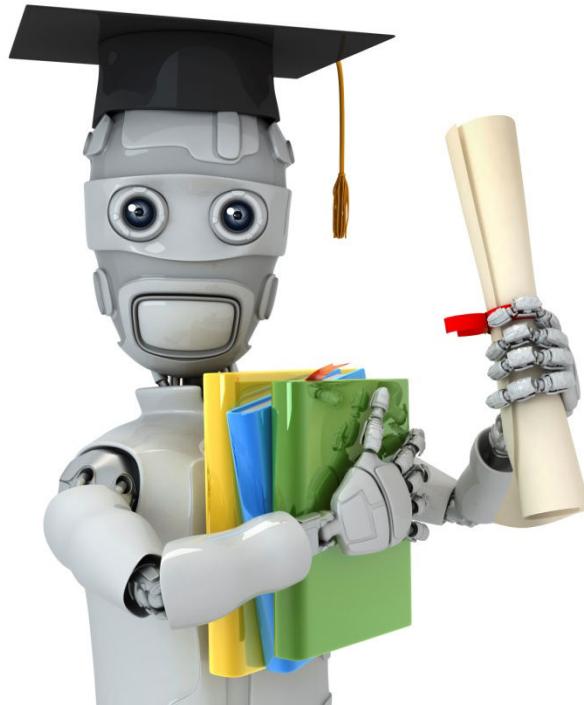
Supervised learning

- • Fraud detection $y=1$
- • Manufacturing (e.g. aircraft engines)
- • Monitoring machines in a data center

⋮

- Email spam classification ←
- Weather prediction (sunny/rainy/etc).
- Cancer classification ←

⋮

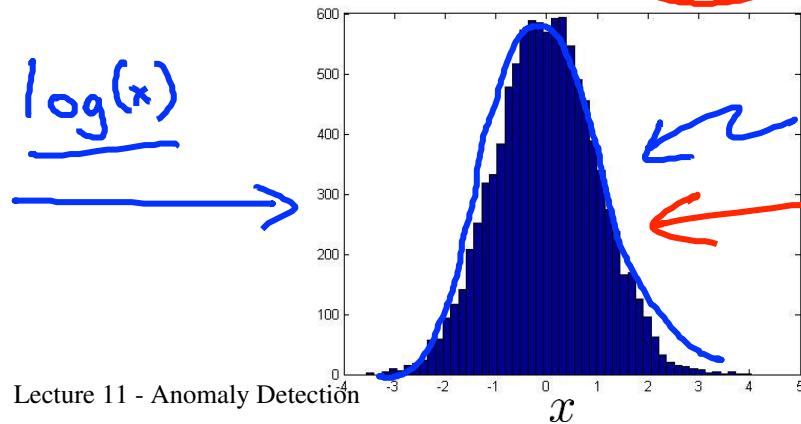
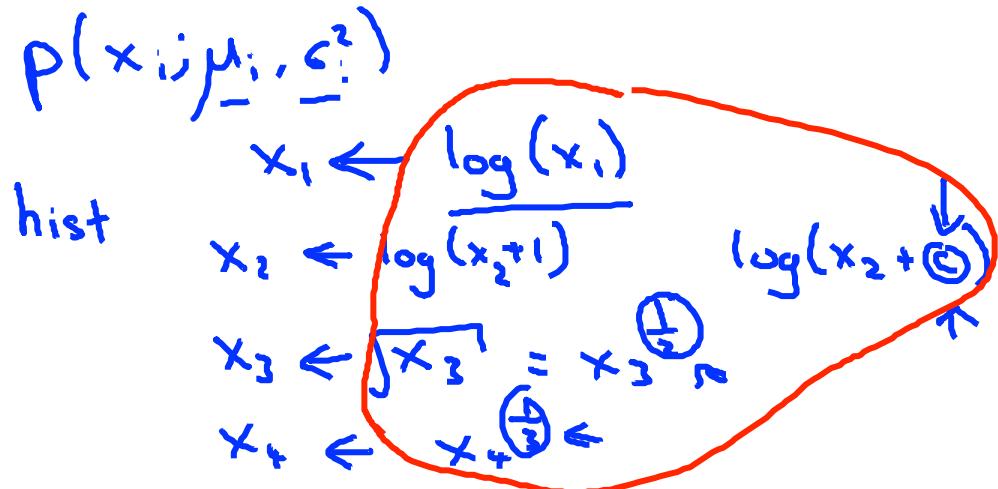
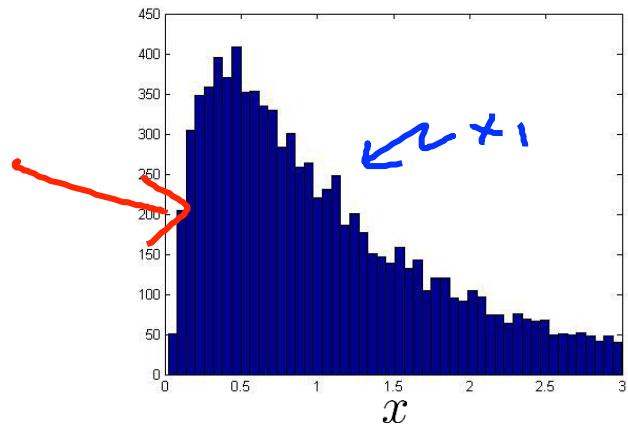
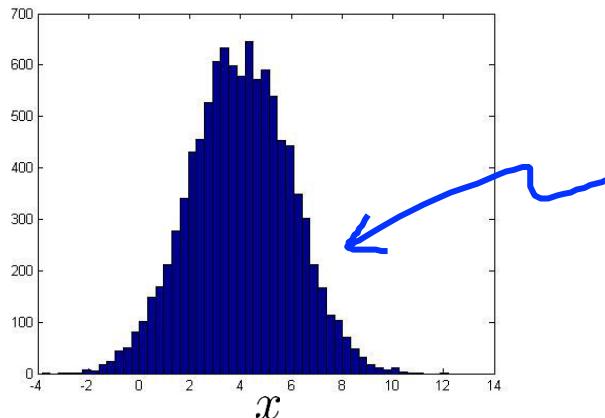


Machine Learning

Anomaly detection

Choosing what
features to use

Non-gaussian features

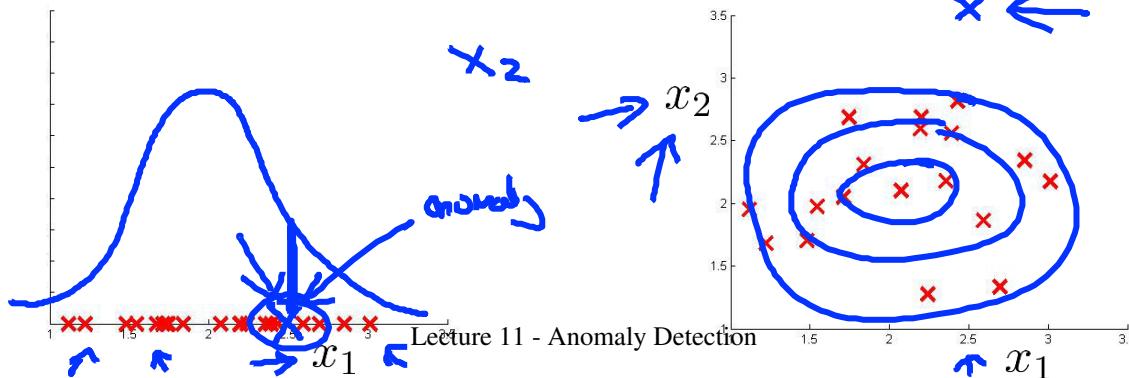


→ Error analysis for anomaly detection

[Want $p(x)$ large for normal examples x .
 $p(x)$ small for anomalous examples x .

Most common problem:

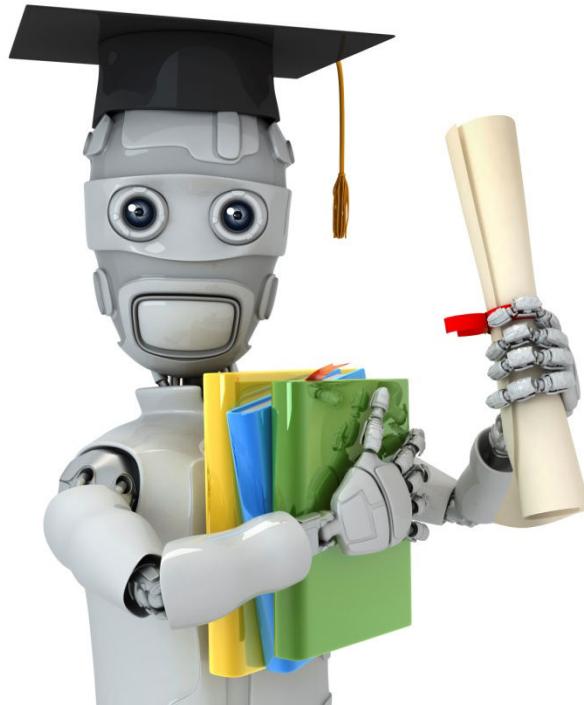
[$p(x)$ is comparable (say, both large) for normal and anomalous examples



- Monitoring computers in a data center
- Choose features that might take on unusually large or small values in the event of an anomaly.
 - x_1 = memory use of computer
 - x_2 = number of disk accesses/sec
 - x_3 = CPU load ←
 - x_4 = network traffic ←

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

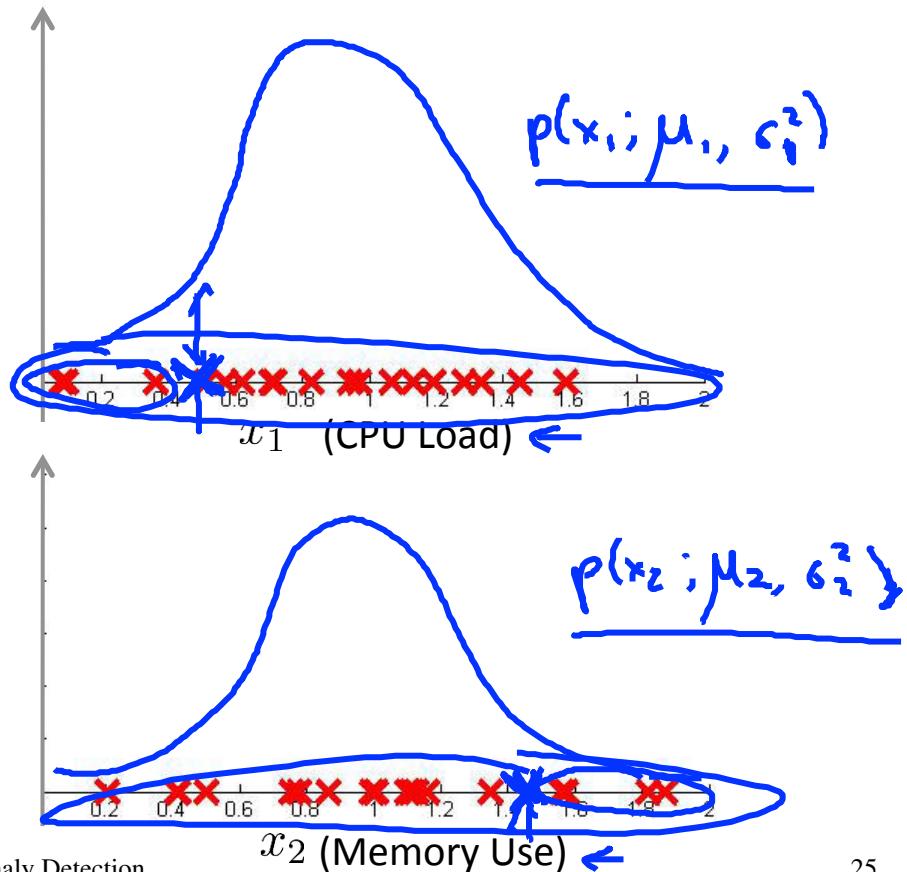
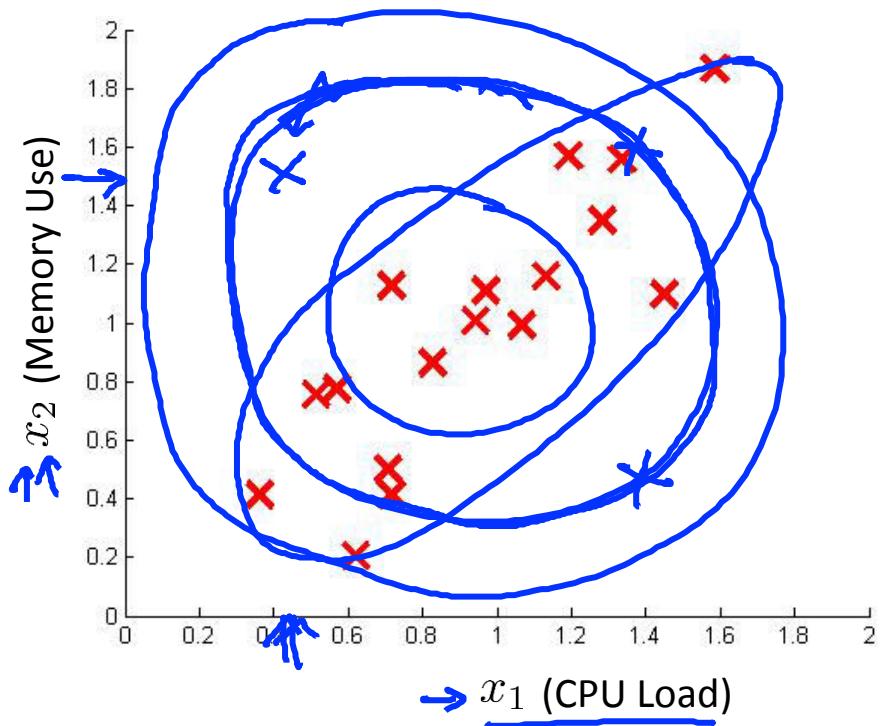


Machine Learning

Anomaly detection

Multivariate
Gaussian distribution

Motivating example: Monitoring machines in a data center



Multivariate Gaussian (Normal) distribution

→ $x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \dots$, etc. separately.
Model $p(x)$ all in one go.
Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

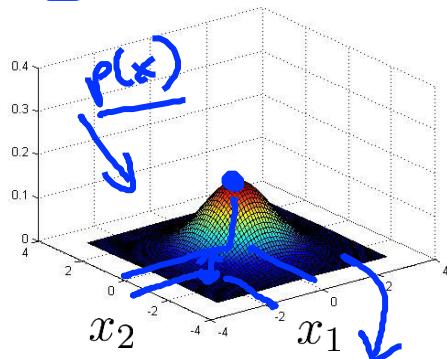
$$p(x; \mu, \Sigma) =$$

$$\frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

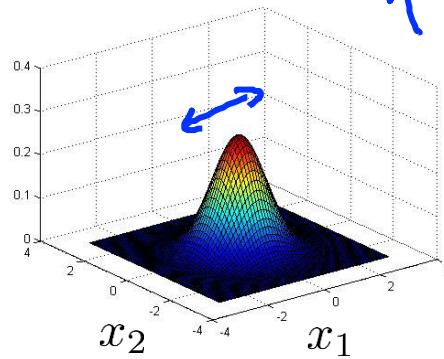
$|\Sigma| = \text{determinant of } \Sigma \quad | \det(\Sigma)$

Multivariate Gaussian (Normal) examples

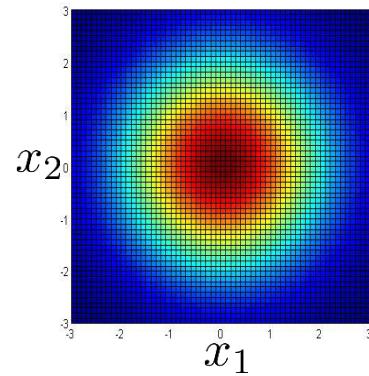
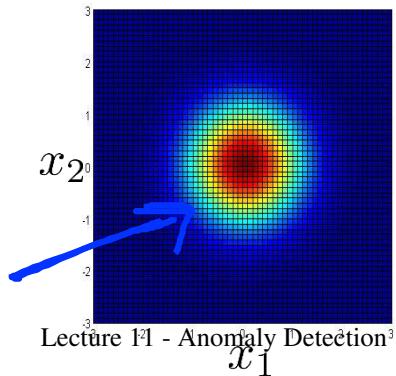
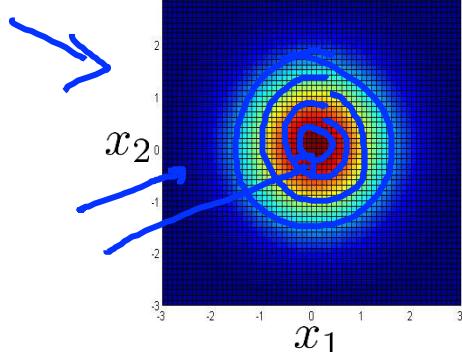
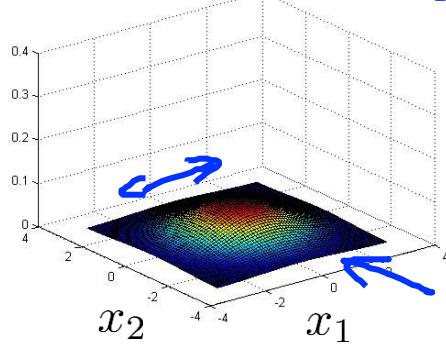
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$

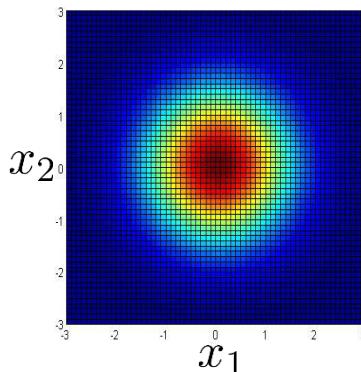
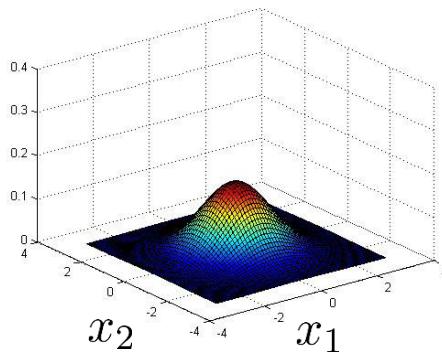


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

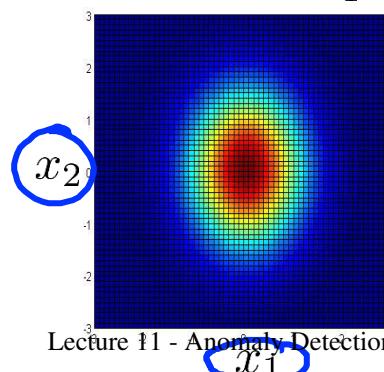
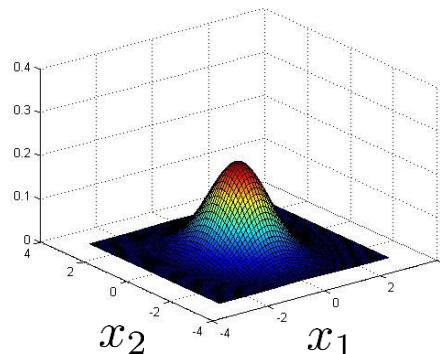


Multivariate Gaussian (Normal) examples

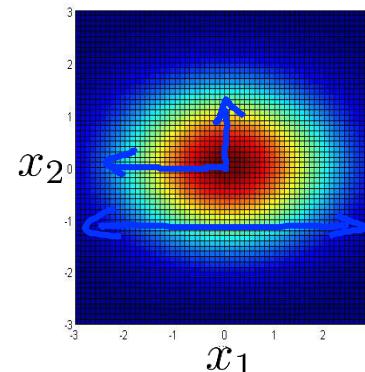
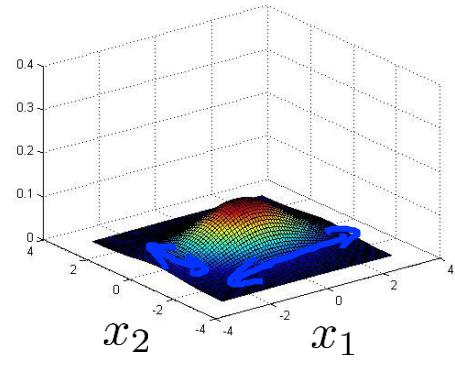
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$$

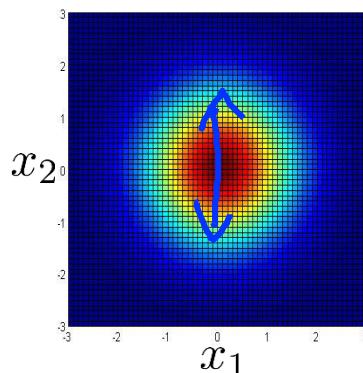
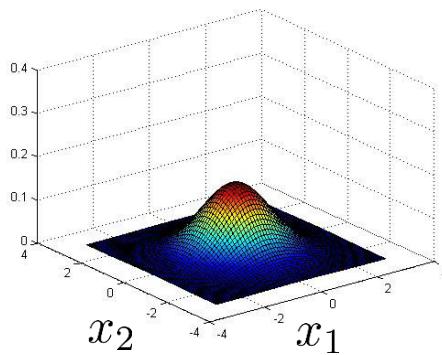


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

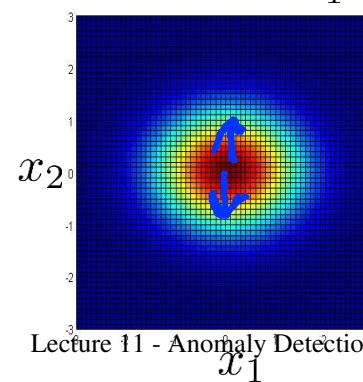
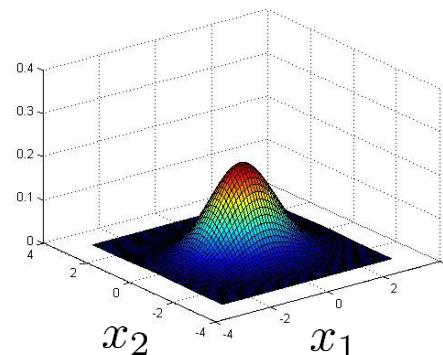


Multivariate Gaussian (Normal) examples

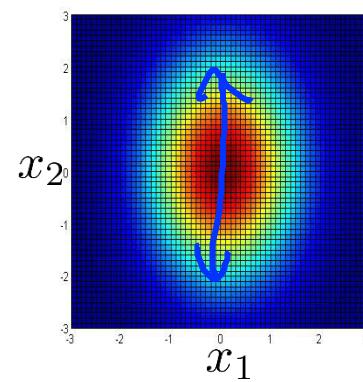
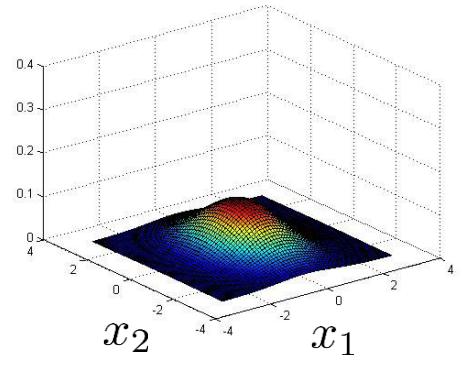
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$$



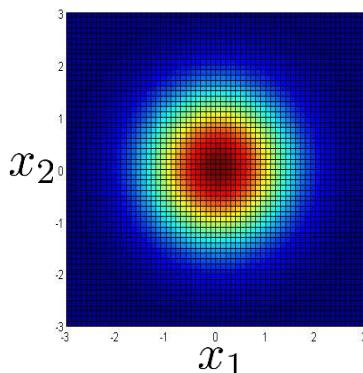
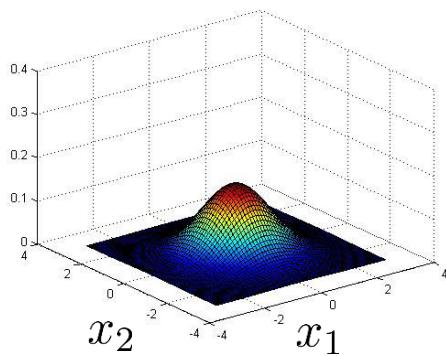
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$



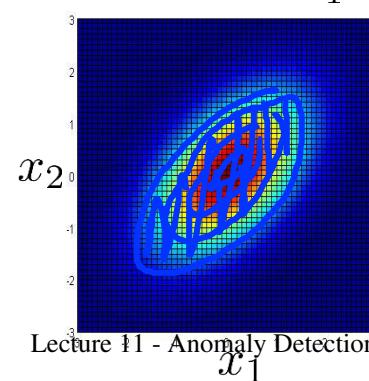
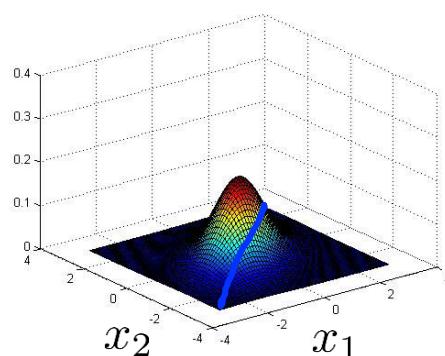
Lecture 11 - Anomaly Detection

Multivariate Gaussian (Normal) examples

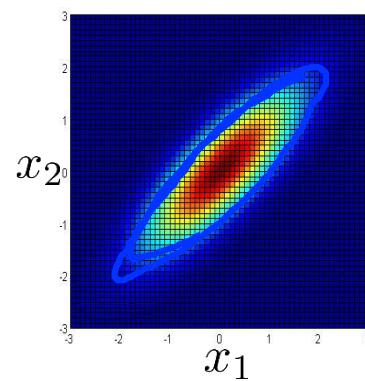
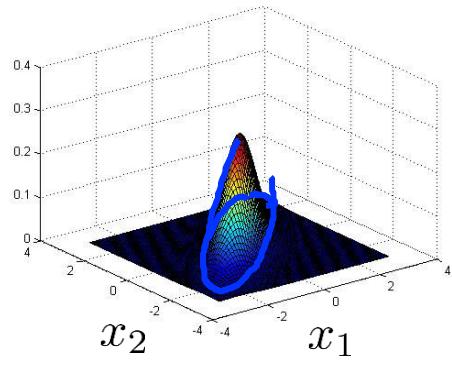
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

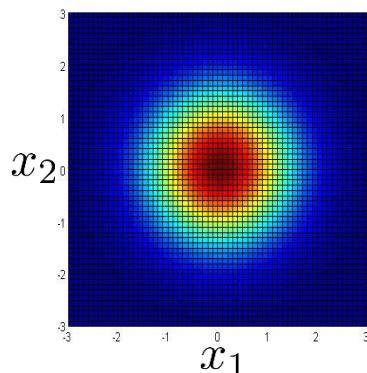
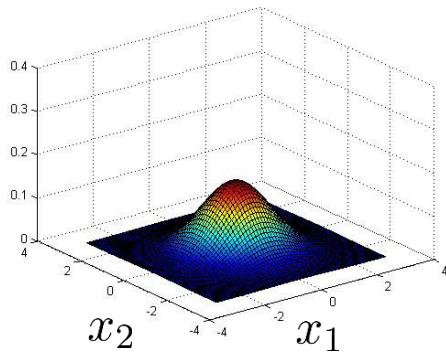


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

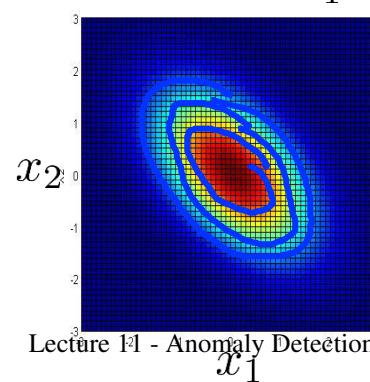
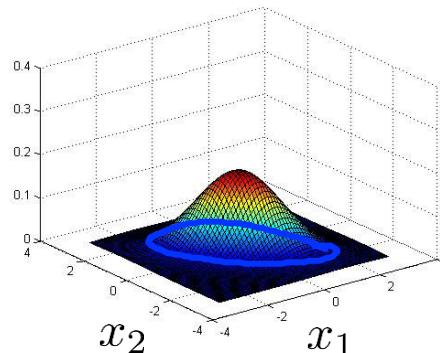


Multivariate Gaussian (Normal) examples

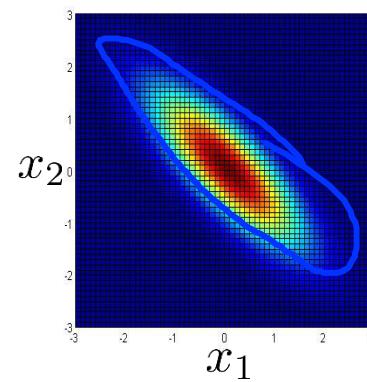
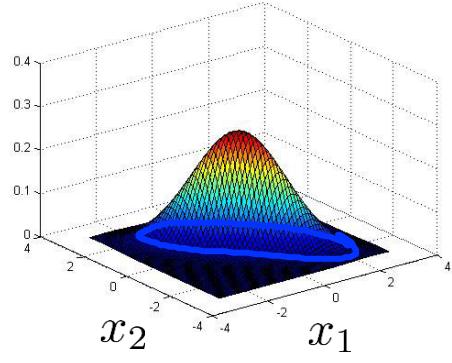
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

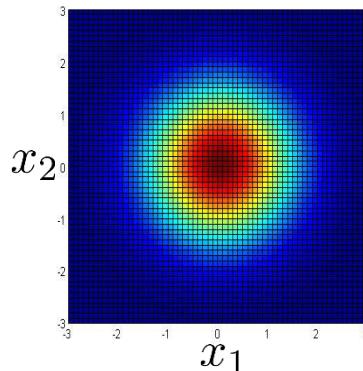
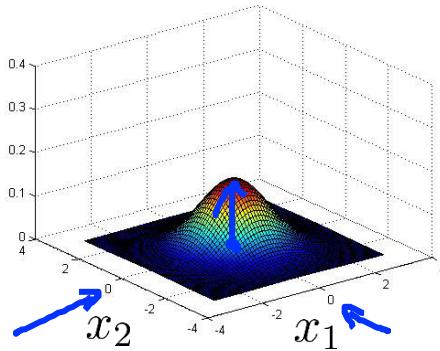


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

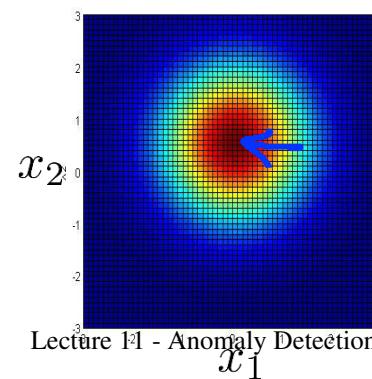
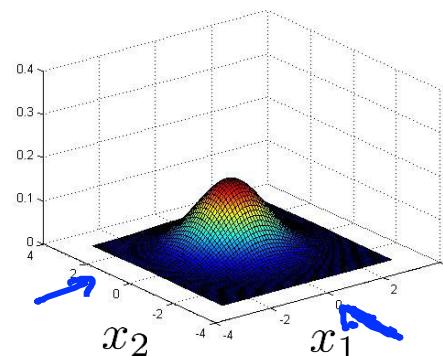


Multivariate Gaussian (Normal) examples

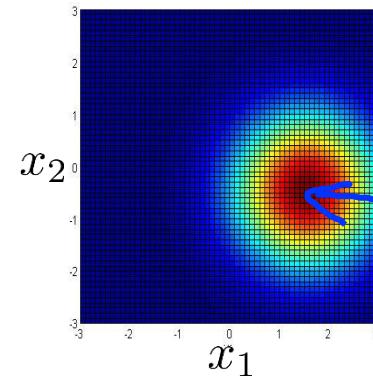
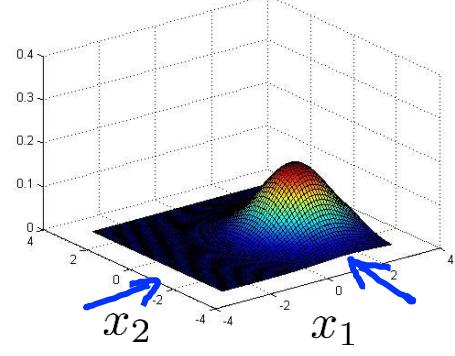
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

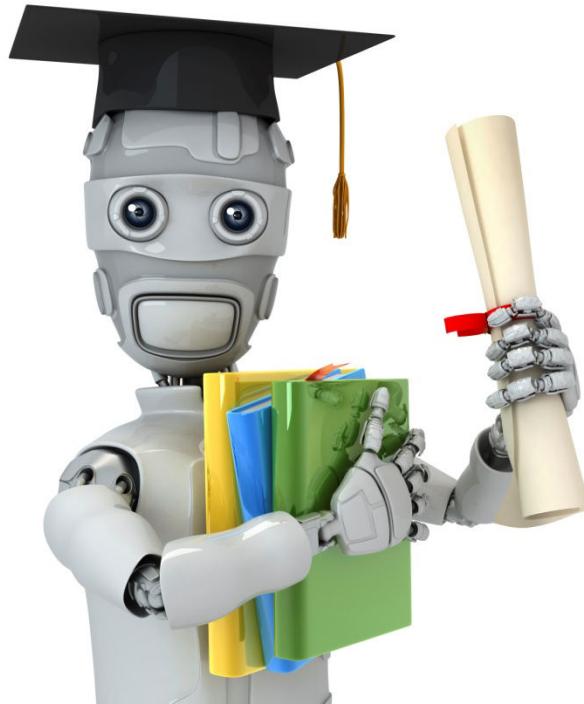


$$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





Machine Learning

Anomaly detection

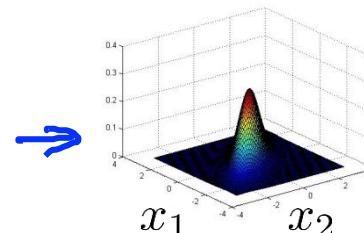
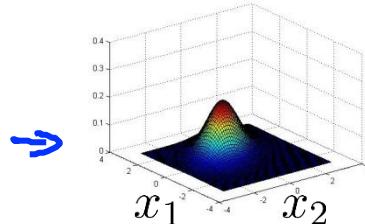
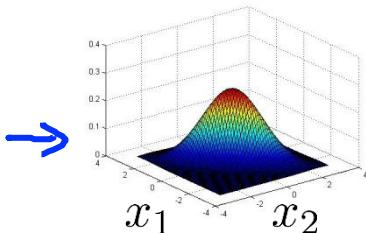
Anomaly detection using
the multivariate
Gaussian distribution

Multivariate Gaussian (Normal) distribution

Parameters μ, Σ

$$\mu \in \mathbb{R}^n \quad \Sigma \in \mathbb{R}^{n \times n}$$

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$



Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$$x \in \mathbb{R}^n$$

$$\rightarrow \boxed{\mu} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\rightarrow \boxed{\Sigma} = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

Anomaly detection with the multivariate Gaussian

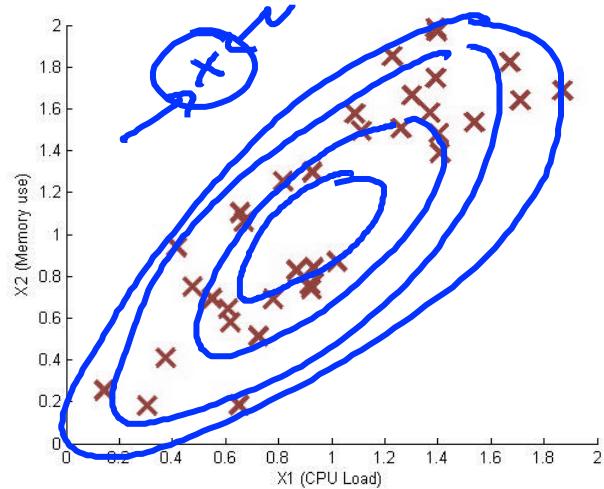
1. Fit model $p(x)$ by setting

$$\left[\begin{array}{l} \mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \\ \Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T \end{array} \right]$$

2. Given a new example x , compute

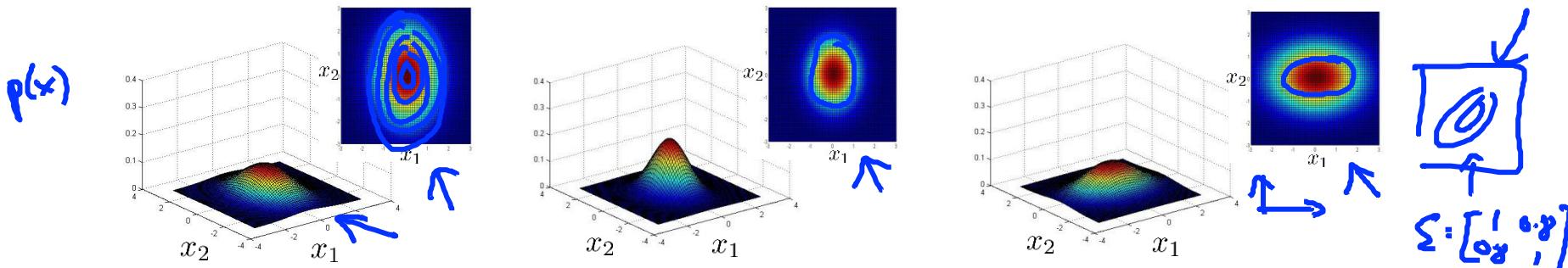
$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Flag an anomaly if $p(x) < \varepsilon$



Relationship to original model

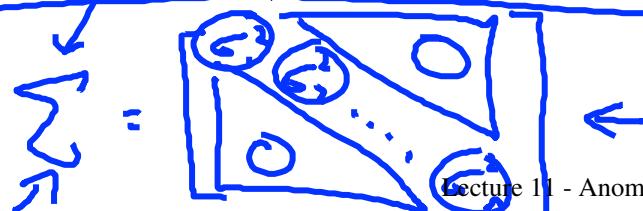
Original model: $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$



Corresponds to multivariate Gaussian

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

where



→ Original model

$$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.

$$\rightarrow x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$$

→ Computationally cheaper (alternatively, scales better to large $n=10,000, n=100,000$)

OK even if m (training set size) is small

vs. → Multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

→ Automatically captures correlations between features

$$\Sigma \in \mathbb{R}^{m \times m}$$

$$\underline{\Sigma^{-1}}$$

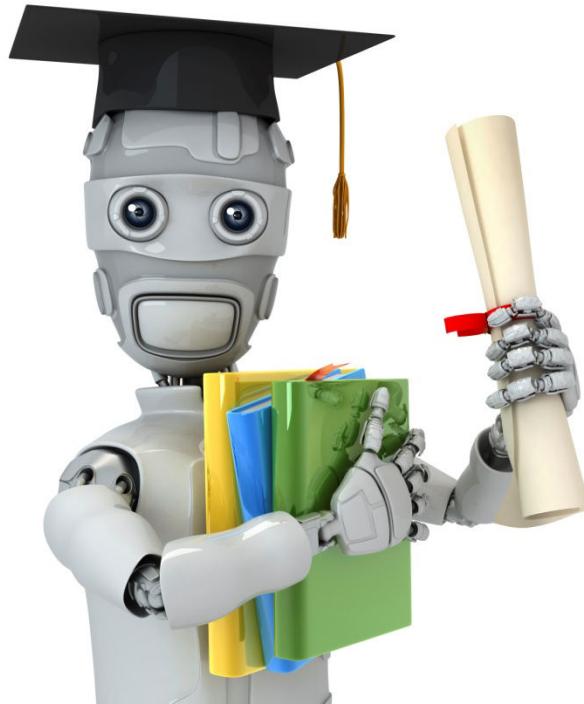
Computationally more expensive

$$\rightarrow \Sigma \sim \frac{n^2}{2}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 = x_4 + x_5 \end{bmatrix}$$

Must have $m > n$ or else Σ is non-invertible.

$$\underline{m \geq n}$$



Machine Learning

Recommender Systems

Problem formulation

Example: Predicting movie ratings

→ User rates movies using ~~one to five stars~~
~~zero~~

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	6
Romance forever	5	?	0	0
Cute puppies of love	?	4	0	0
Nonstop car chases	5	0	5	4
Swords vs. karate	0	0	?	?

$$n_u = 4$$

$$n_m = 5$$

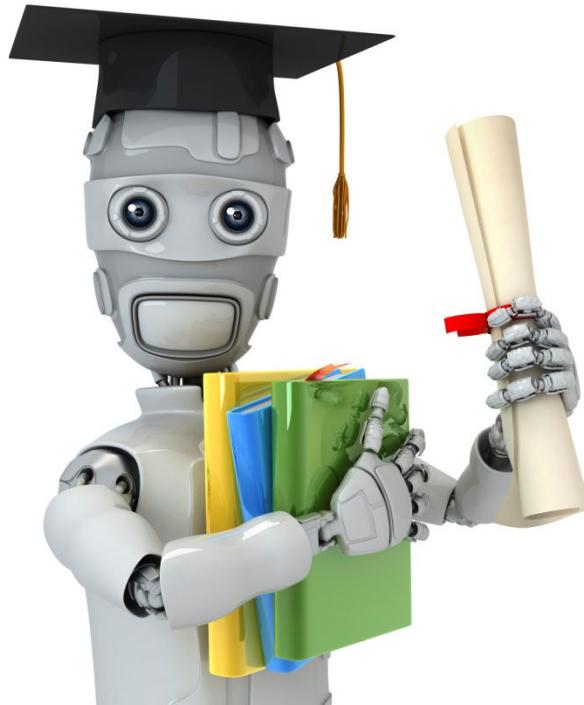


→ n_u = no. users

→ n_m = no. movies

→ $r(i, j) = 1$ if user j has rated movie i
 $y^{(i,j)}$ = rating given by user j to movie i
(defined only if $r(i, j) = 1$)

$$0, \dots, 5$$



Machine Learning

Recommender Systems

Content-based
recommendations

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	$n_u = 4, n_m = 5$	$x_o = 1$	$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$
Love at last	1	5	5	0	0		
Romance forever	2	5	?	?	0		
Cute puppies of love	3	?	4	0	?		
Nonstop car chases	4	0	0	5	4		
Swords vs. karate	5	0	0	5	?		

For each user j , learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie $(\theta^{(j)} \text{ with } x^{(i)})$ stars.

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \leftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

$$(\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Problem formulation

- $r(i, j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating by user j on movie i (if defined)
- $\theta^{(j)}$ = parameter vector for user j
- $x^{(i)}$ = feature vector for movie i
- For user j , movie i , predicted rating: $(\theta^{(j)})^T(x^{(i)})$
- $m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i : r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$$\theta^{(j)} \in \mathbb{R}^{n+1}$$

Optimization objective:

To learn $\underline{\theta^{(j)}}$ (parameter for user j):

$$\rightarrow \min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

To learn $\underline{\theta^{(1)}}, \underline{\theta^{(2)}}, \dots, \underline{\theta^{(n_u)}}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$\Theta^{(1)}, \dots, \Theta^{(n_u)}$

Optimization algorithm:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$J(\theta^{(1)}, \dots, \theta^{(n_u)})$

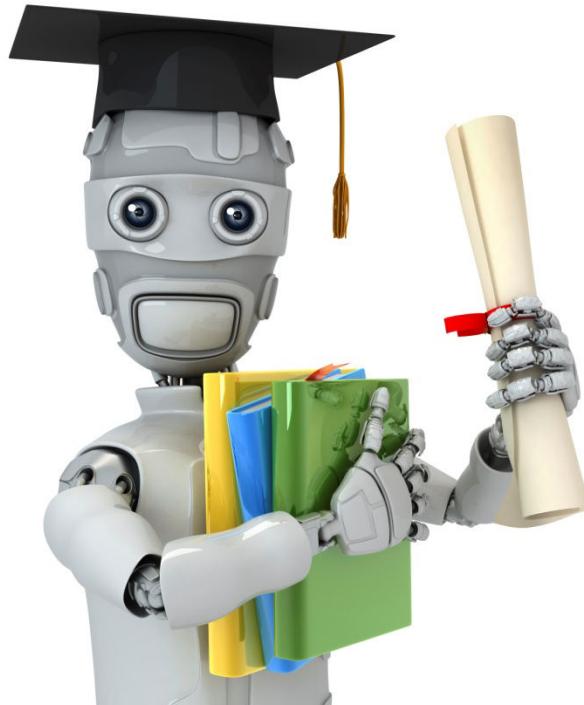
Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0)$$

~~m^(j)~~

$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$



Machine Learning

Recommender Systems

Collaborative filtering

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9



Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)	$x_o = 1$
$x^{(1)}$ Love at last	5	5	0	0	? 1.0	? 0.0	
Romance forever	5	?	?	0	[?]	[?]	
Cute puppies of love	?	4	0	?	[?]	[?]	
Nonstop car chases	0	0	5	4	[?]	[?]	
Swords vs. karate	0	0	5	?	[?]	[?]	

$\rightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

$\theta^{(j)}$

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 10$

$x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$

$x^{(N)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$

Optimization algorithm

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (x_k^{(i)})^2$$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Collaborative filtering

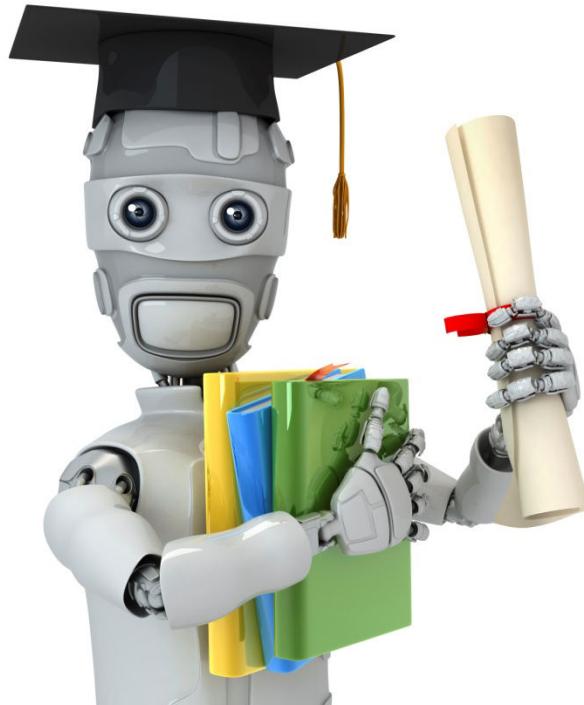
Given $x^{(1)}, \dots, x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$

$$\begin{matrix} r^{(i,j)} \\ y^{(i,j)} \end{matrix}$$



Given $\theta^{(1)}, \dots, \theta^{(n_u)}$,
can estimate $x^{(1)}, \dots, x^{(n_m)}$

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \dots$



Machine Learning

Recommender Systems

Collaborative
filtering algorithm

Collaborative filtering optimization objective

Given $x^{(1)}, \dots, x^{(n_m)}$, estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$$(i,j) : r(i,j) = 1$$

$$x \in \mathbb{R}^n$$

$$\theta \in \mathbb{R}^n$$

$$x_1 = 1$$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, estimate $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Minimizing $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$$\min_{x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$



Collaborative filtering algorithm

~~$x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$~~
 ~~θ~~
 ~~θ_1~~
 ~~θ_n~~

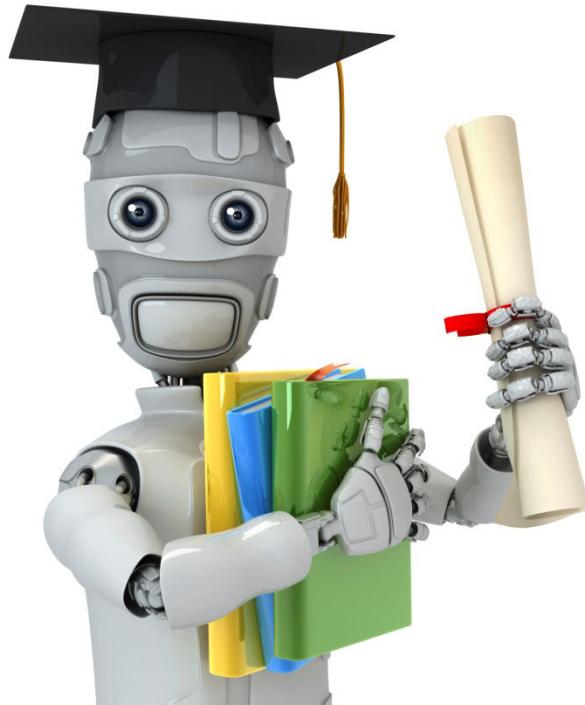
- 1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values.
- 2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \dots, n_u, i = 1, \dots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$
$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

$\frac{\partial J}{\partial x_k^{(i)}}$

- 3. For a user with parameters $\underline{\theta}$ and a movie with (learned) features \underline{x} , predict a star rating of $\underline{\theta}^T \underline{x}$.

$$(\underline{\theta}^{(i)})^T (\underline{x}^{(i)})$$



Machine Learning

Recommender Systems

Vectorization:
Low rank matrix
factorization

Collaborative filtering

$$n_m = 5$$

$$n_u = 4$$

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$



 $y^{(i,j)}$

Collaborative filtering

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings:

$$\times \Theta^T \leftarrow$$

$$(\Theta^{(1)})^T (x^{(1)})$$

$$(i, j)$$

$$\begin{bmatrix} (\theta^{(1)})^T (x^{(1)}) & (\theta^{(2)})^T (x^{(1)}) & \dots & (\theta^{(n_u)})^T (x^{(1)}) \\ (\theta^{(1)})^T (x^{(2)}) & (\theta^{(2)})^T (x^{(2)}) & \dots & (\theta^{(n_u)})^T (x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T (x^{(n_m)}) & (\theta^{(2)})^T (x^{(n_m)}) & \dots & (\theta^{(n_u)})^T (x^{(n_m)}) \end{bmatrix}$$

$$\rightarrow X = \begin{bmatrix} -(x^{(1)})^T \\ -(x^{(2)})^T \\ \vdots \\ -(x^{(n_m)})^T \end{bmatrix} \quad \Theta = \begin{bmatrix} -(\Theta^{(1)})^T \\ -(\Theta^{(2)})^T \\ \vdots \\ -(\Theta^{(n_u)})^T \end{bmatrix}$$

Lecture 12: Recommender Systems

Low rank matrix factorization

Finding related movies

For each product i , we learn a feature vector $\underline{x}^{(i)} \in \mathbb{R}^n$.

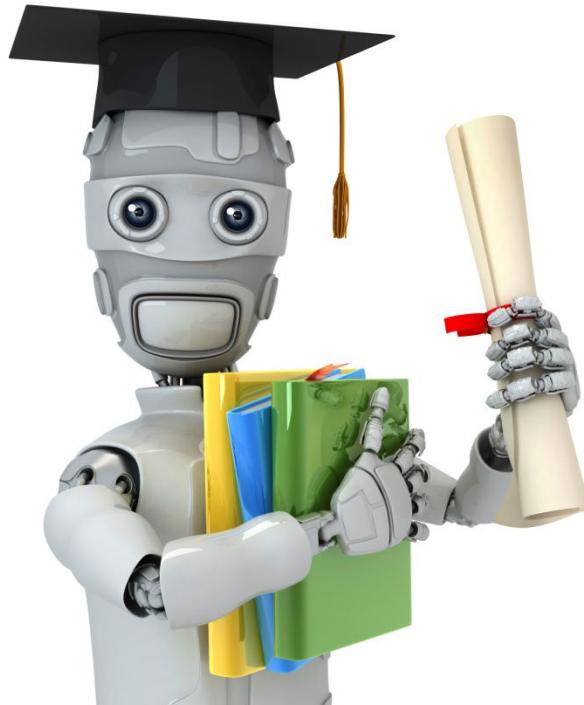
$\rightarrow x_1 = \text{romance}$, $x_2 = \text{action}$, $x_3 = \text{comedy}$, $x_4 = \dots$

How to find movies j related to movie i ?

small $\|x^{(i)} - x^{(j)}\|$ \rightarrow movie j and i are "similar"

5 most similar movies to movie i :

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.



Machine Learning

Recommender Systems

Implementational
detail: Mean
normalization

Users who have not rated any movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

↓

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

$$\min_{x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$n=2$

$$\underline{\Theta}^{(s)} \in \mathbb{R}^2$$

$$\underline{\Theta}^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{\lambda}{2} [(\underline{\Theta}_1^{(s)})^2 + (\underline{\Theta}_2^{(s)})^2] \leftarrow$$

$$(\underline{\Theta}^{(s)})^T \underline{x}^{(i)} = 0$$

Mean Normalization:

$$Y = \begin{bmatrix} \rightarrow & 5 & 5 & 0 & 0 & ? & -2.5 \\ \rightarrow & 5 & ? & ? & 0 & ? & -2.5 \\ Y = & ? & 4 & 0 & ? & ? & -2 \\ \rightarrow & 0 & 0 & 5 & 4 & ? & \vdots \\ \rightarrow & 0 & 0 & 5 & 0 & ? & \vdots \end{bmatrix}$$

$$\mu = \begin{bmatrix} \rightarrow & 2.5 \\ \rightarrow & 2.5 \\ \rightarrow & 2 \\ \rightarrow & 2.25 \\ \rightarrow & 1.25 \end{bmatrix} \rightarrow \underline{Y} =$$

$$\begin{bmatrix} \circled{2.5} & \circled{2.5} & \circled{-2.5} & \circled{-2.5} & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j , on movie i predict:

$$\rightarrow (\underline{\theta}^{(s)})^T (\underline{x}^{(i)}) + \underline{\mu_i}$$

\downarrow
learn $\underline{\theta}^{(s)}, \underline{x}^{(i)}$

User 5 (Eve):

$$\underline{\theta}^{(s)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$(\underline{\theta}^{(s)})^T (\underline{x}^{(i)}) + \boxed{\underline{\mu_i}}$$