

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



Course Outline



- Introduction to AI
- The History of AI
- Python
- Search
- Informed Search
- Beyond Classical Search
- Adversarial Search
- Constraint Satisfaction Problems
- Fuzzy Logic
- **Introduction to Machine Learning**

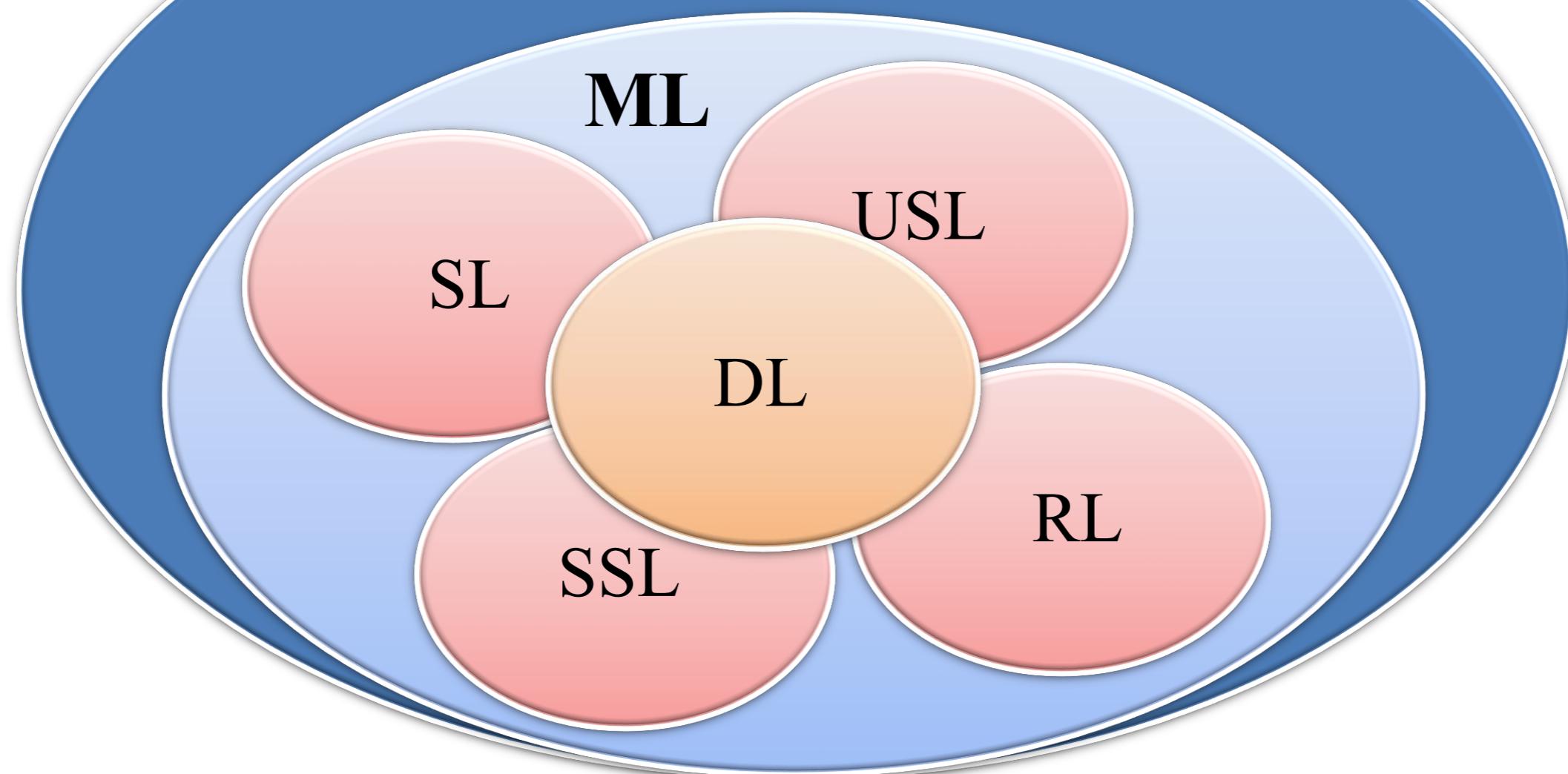
Overview of Machine Learning



A.I. is the study of
how to make
computers do
things at which, at
the moment, people
are better.



Artificial Intelligence





Quotes about ML

“A breakthrough in machine learning would be worth ten Microsofts”

– *(Bill Gates, Chairman, Microsoft)*

“Machine learning is the next Internet”

– *(Tony Tether, Director, DARPA)*

“Machine learning is the hot new thing”

– *(John Hennessy, President, Stanford)*

“Web rankings today are mostly a matter of machine learning”

– *(Prabhakar Raghavan, Dir. Research, Yahoo)*

“Machine learning is going to result in a real revolution”

– *(Greg Papadopoulos, CTO, Sun)*

“Machine learning is today’s discontinuity”

– *(Jerry Yang, CEO, Yahoo)*

Why ML?



Because Machine can drive
your car for you!



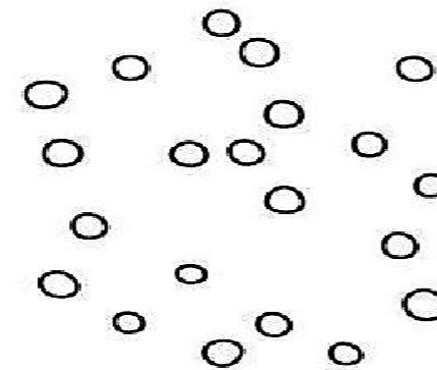
Because Machine can detect
50 eye diseases!



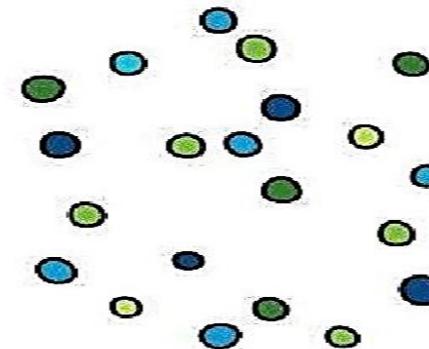
Because machine can unlock
your phone with your face!

Data

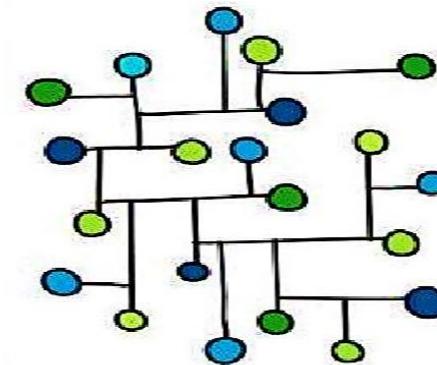
Data



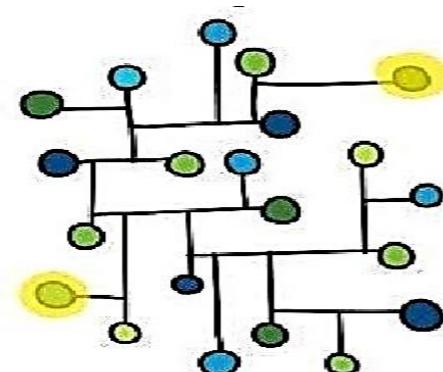
Information



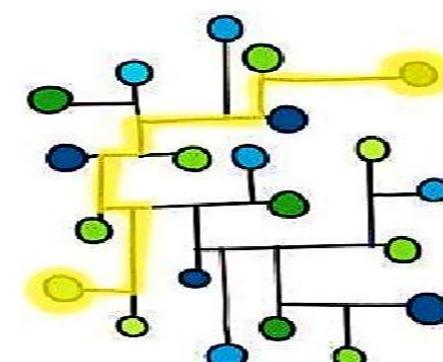
Knowledge



Insight



Wisdom

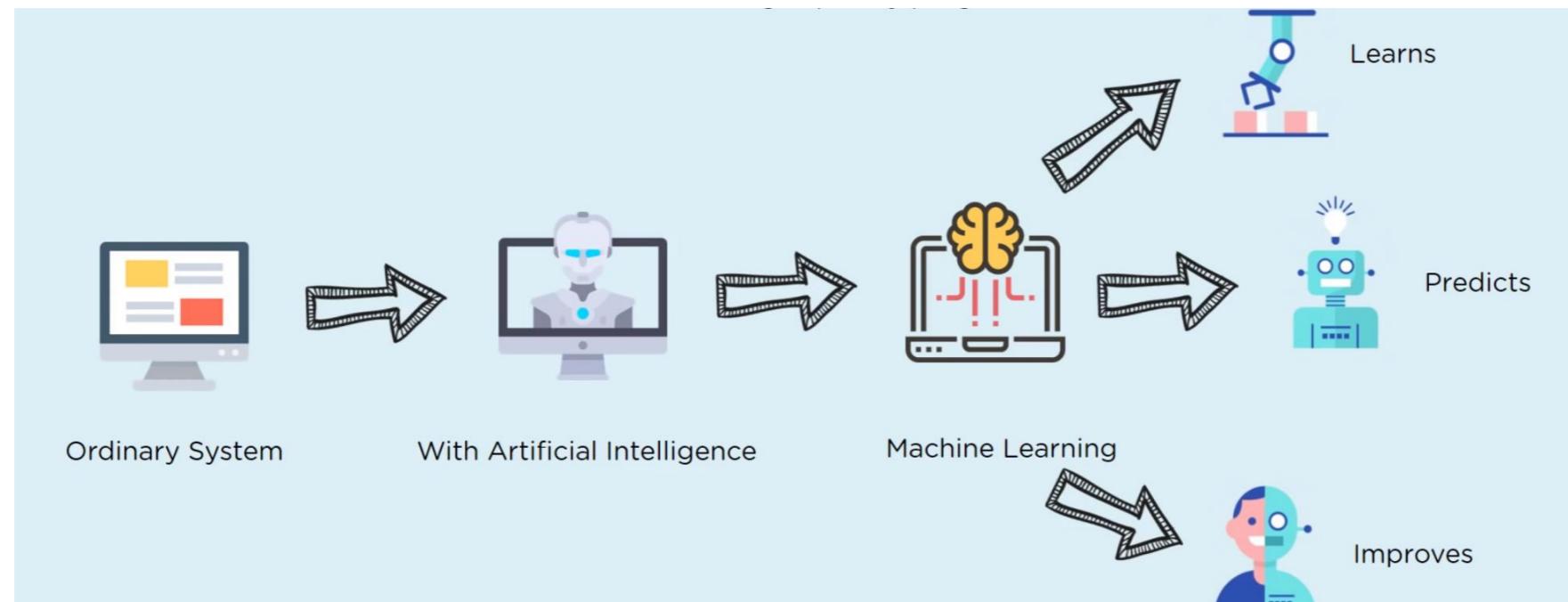


Impact

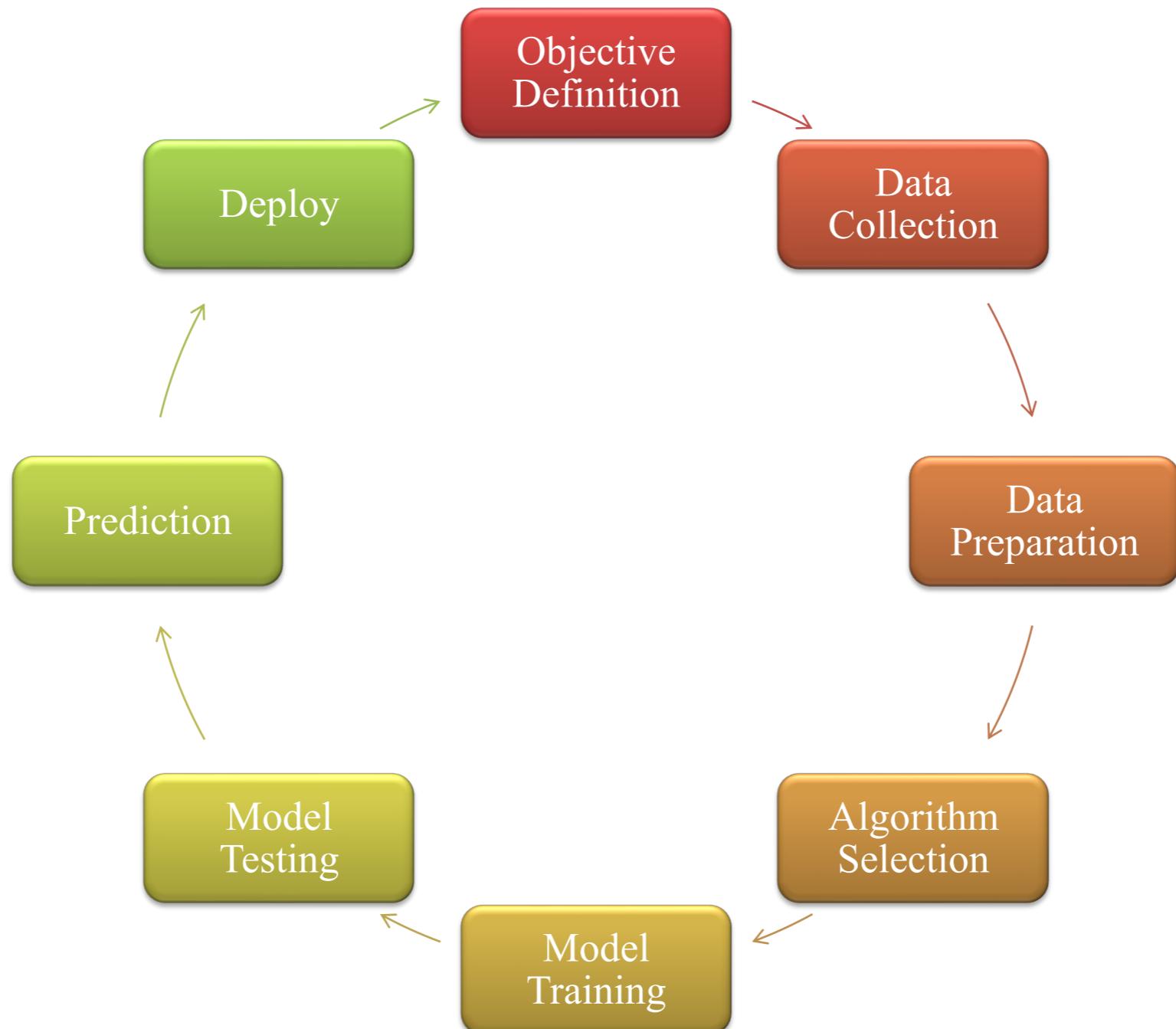


What is ML?

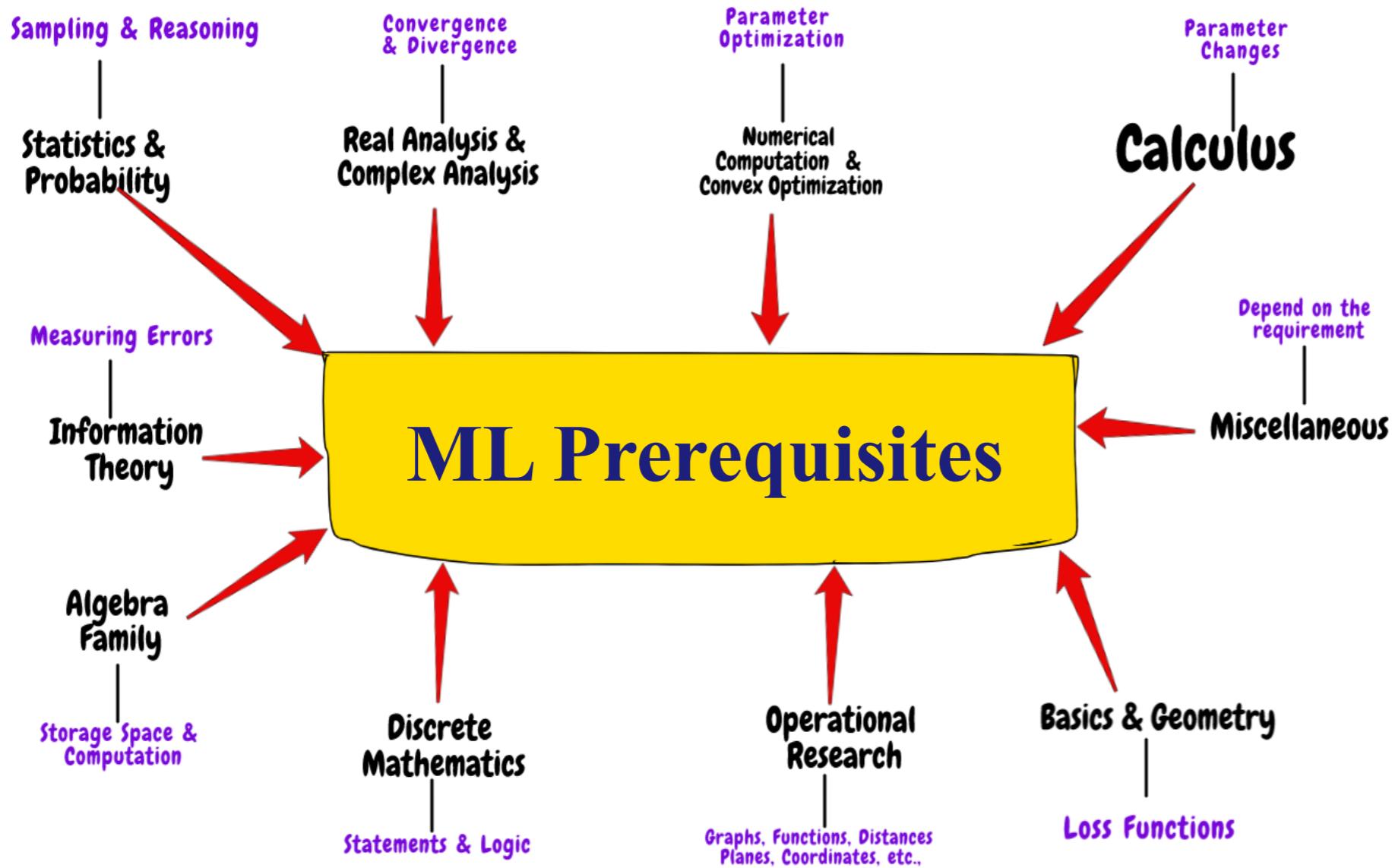
Machine learning is the science of making computers **learn** and **act** like human by **feeding data** and information without being explicitly programmed.



Steps



Prerequisites





The stock price will increase or decrease?



Predicting the age of a person based on the height, weight, health, and other factors?

Do you want
to detect an
anomaly?
**(anomaly
detection)**



Want to detect money withdrawal anomalies!

Do you want to
discover
structure in
unexplored data?
(Clustering)



Finding groups of customers with similar behavior
given a large database of customer data containing
their demographics and past records



What Is ML?

- Informally;
 - **Automating automation**
 - Getting computers to **program themselves**
 - Writing software is the **bottleneck**
 - Let the **data** do the work instead!

ML is a Magic?



No, more like gardening!

Seeds = Algorithms

Nutrients = Data

Gardener = You

Plants = Programs



The machine learning framework

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

$$f(\text{apple}) = \text{"apple"}$$

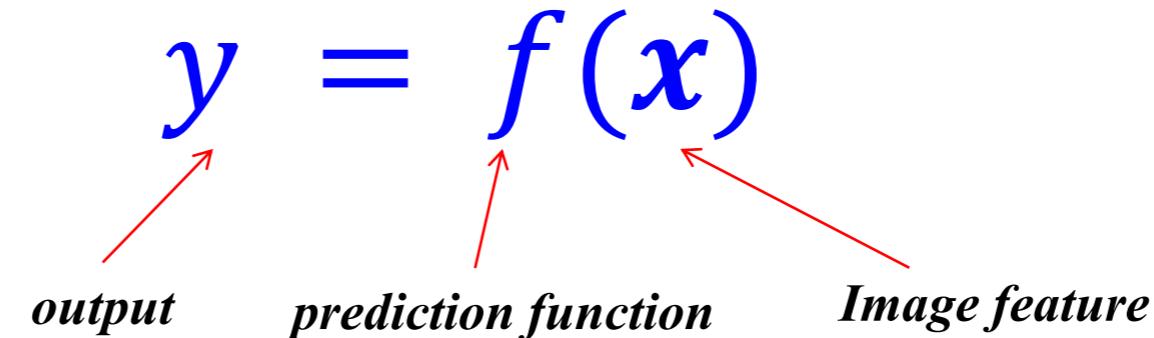
$$f(\text{tomato}) = \text{"tomato"}$$

$$f(\text{cow}) = \text{"cow"}$$

The framework

$$y = f(\mathbf{x})$$

output *prediction function* *Image feature*



The diagram illustrates the components of a prediction function. At the top, the equation $y = f(\mathbf{x})$ is written in blue. Below it, three labels are centered under their respective parts: "output" under y , "prediction function" under f , and "Image feature" under \mathbf{x} . Red arrows point from each label to its corresponding part in the equation.

Training set:

- Given a **training set** of labeled examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set.

Validation set:

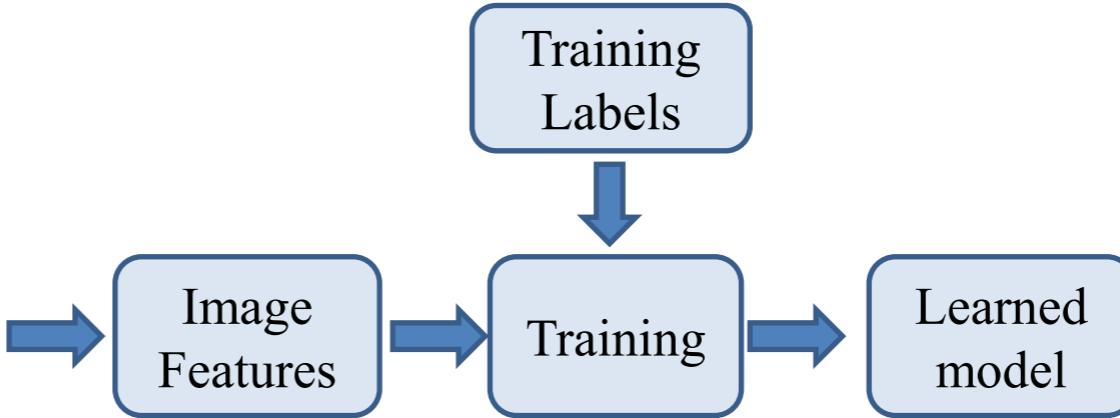
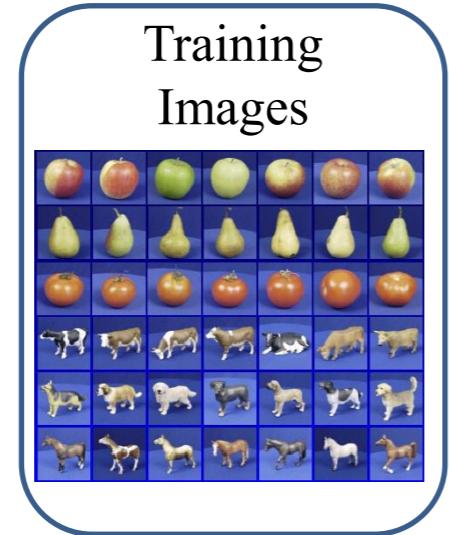
- A set of examples used to tune the parameters.

Testing:

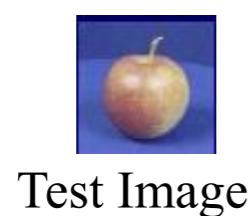
- Apply f to a never-before-seen *test example* \mathbf{x} and output the predicted value $y = f(\mathbf{x})$.

What Is ML?

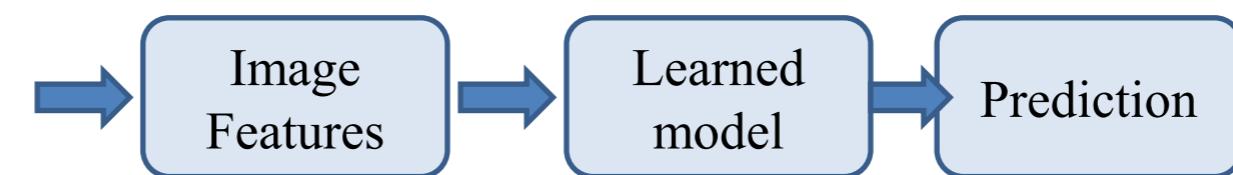
Training



Testing



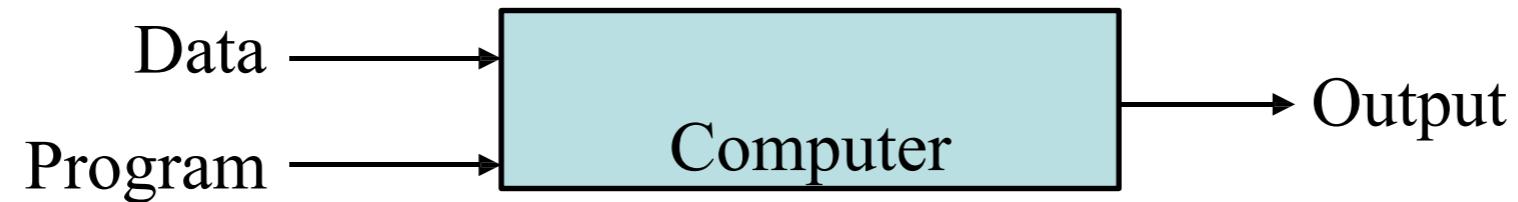
Test Image



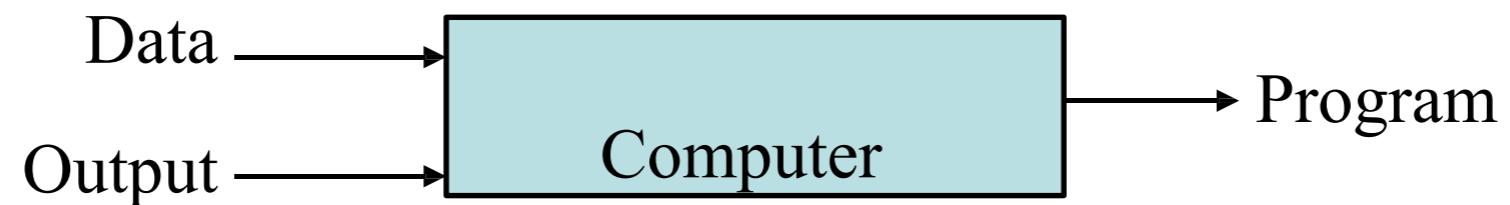
What Is ML?



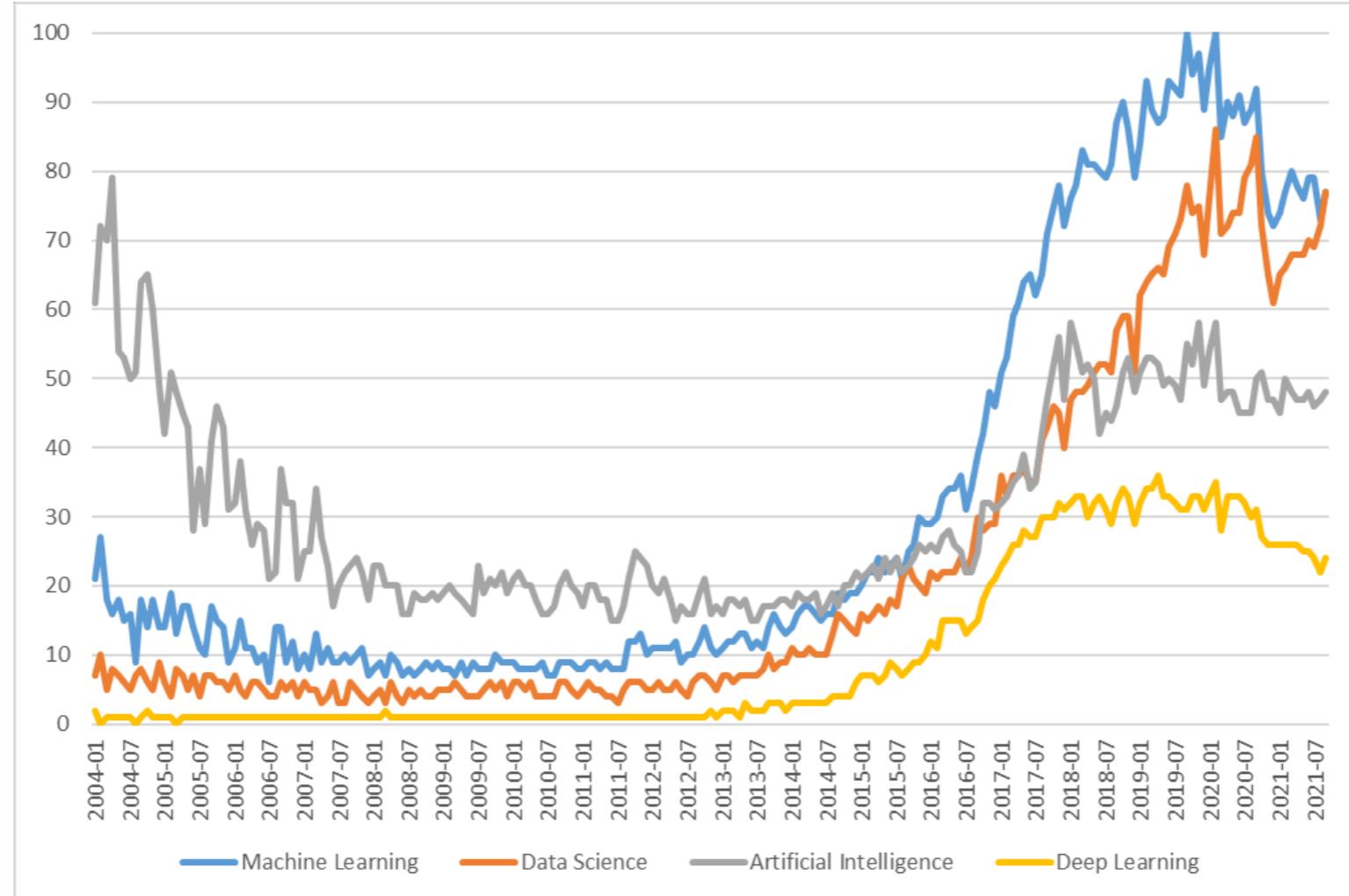
Traditional Programming



Machine Learning



ML Popularity



Google Trends

- Tens of thousands of machine learning algorithms Hundreds new every year
- Every machine learning algorithm has three components:
 - **Representation**
 - **Evaluation**
 - **Optimization**

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- etc.



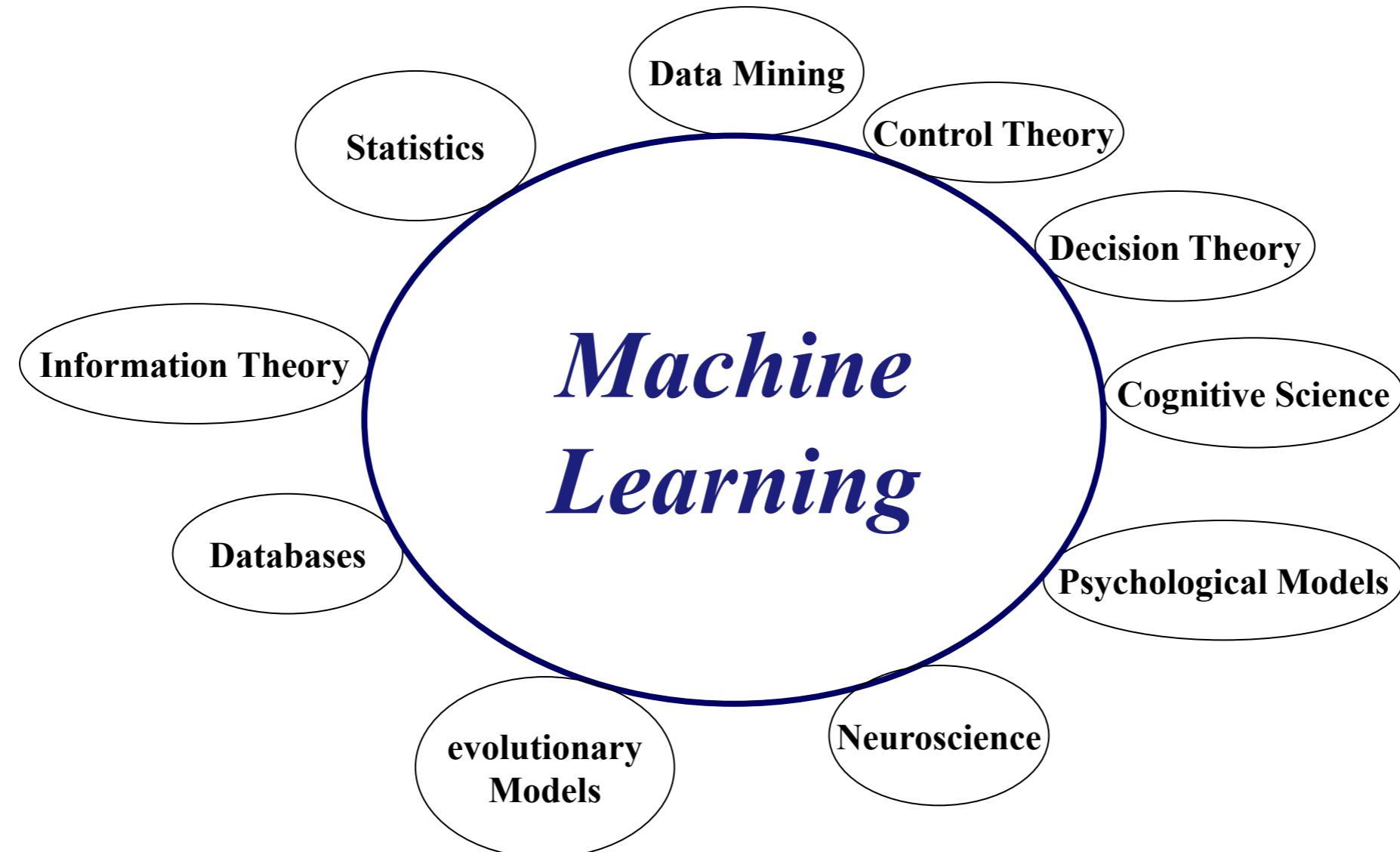
Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence



Optimization

- **Combinatorial optimization**
 - e.g., Greedy search
- **Convex optimization**
 - e.g., Gradient descent
- **Constrained optimization**
 - e.g., Linear programming



Machine learning is primarily concerned with the accuracy and effectiveness of the computer system.

Why “Learn”?

- Machine learning is **programming computers** to **optimize a performance criterion** using **example data** or **past experience**.
- There is no need to “learn” to calculate payroll
- Learning is used when:
 - Human expertise does not exist (navigating on Mars),
 - Humans are unable to explain their expertise (speech recognition)
 - Solution changes in time (routing on a computer network)
 - Solution needs to be adapted to particular cases (user biometrics)

“Learning”?

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Build a model that is a good and useful approximation to the data.

Learning Associations

- Basket analysis:

$P(Y | X)$ probability that somebody who buys X also buys Y where X and Y are products/services.

Example: $P(\text{chips} | \text{beer}) = 0.7$

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Classification

- Example: Credit scoring
 - Differentiating between **low-risk** and **high-risk** customers from their *income* and *savings*

Discriminant:

IF income > θ_1 AND savings > θ_2

THEN low – risk ELSE high – risk

- **Machine Learning**
Study of algorithms that
 - improve their performance
 - at some task
 - with experience
- **Optimize** a performance criterion using example data or experience.
- **Role of Statistics:** Inference from a sample
- **Role of Computer science:** Efficient algorithms to
 - Solve the optimization problem
 - Representing and evaluating the model for inference

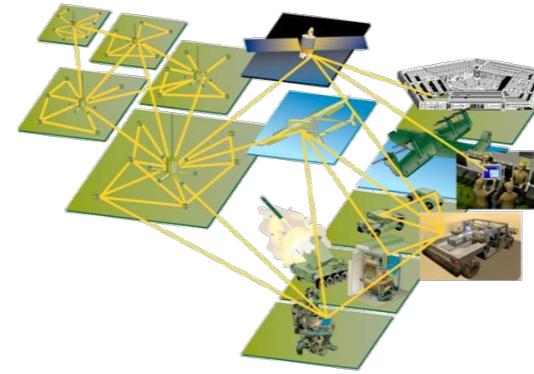
Commercial Motivation for ML

- Lots of data is being collected and warehoused
 - Web data, e-commerce
 - purchases at department/ grocery stores
 - Bank/Credit Card transactions
- Computers have become cheaper and more powerful
- Competitive Pressure is Strong
 - Provide better, customized services for an *edge* (e.g. in Customer Relationship Management)

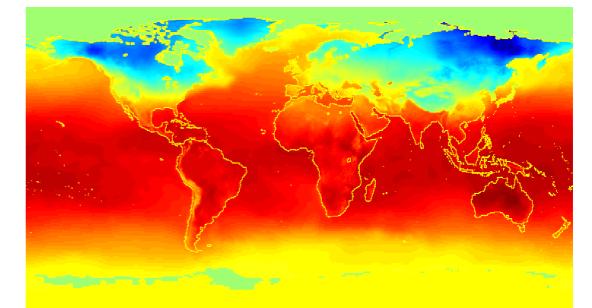
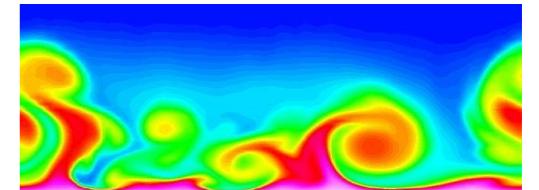


Scientific Motivation for ML

- Data collected and stored at enormous speeds (GB/hour)
 - remote sensors on a satellite
 - telescopes scanning the skies
 - microarrays generating gene expression data
 - scientific simulations generating terabytes of data
- Traditional techniques infeasible for raw data
- Machine Learning may help scientists
 - in classifying and segmenting data
 - in Hypothesis Formation



5 petabytes
(~5,000 years of MP3 audio)



Practical Machine Learning



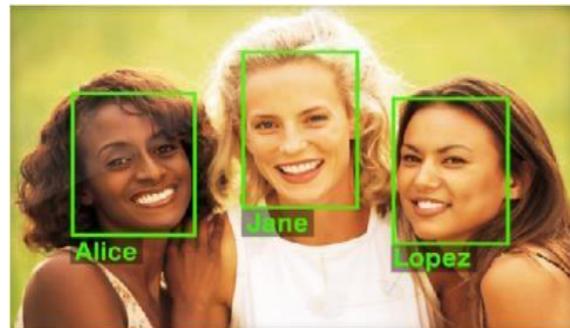
Spam Detection



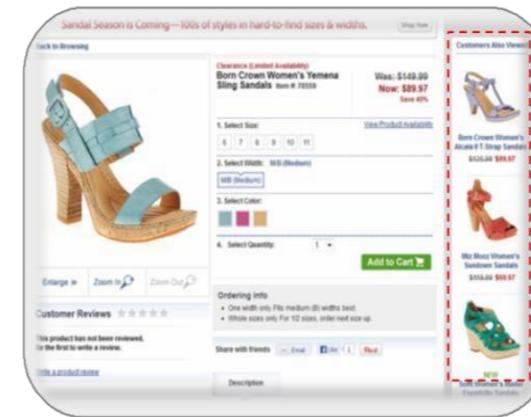
Credit Card Fraud Detection



Speech Recognition



Face Detection



Product Recommendation



Sentiment Analysis

This **robot** uses artificial intelligence
with deep learning to recycle better

#techthatmatters

Bulk Handling Systems



Faults

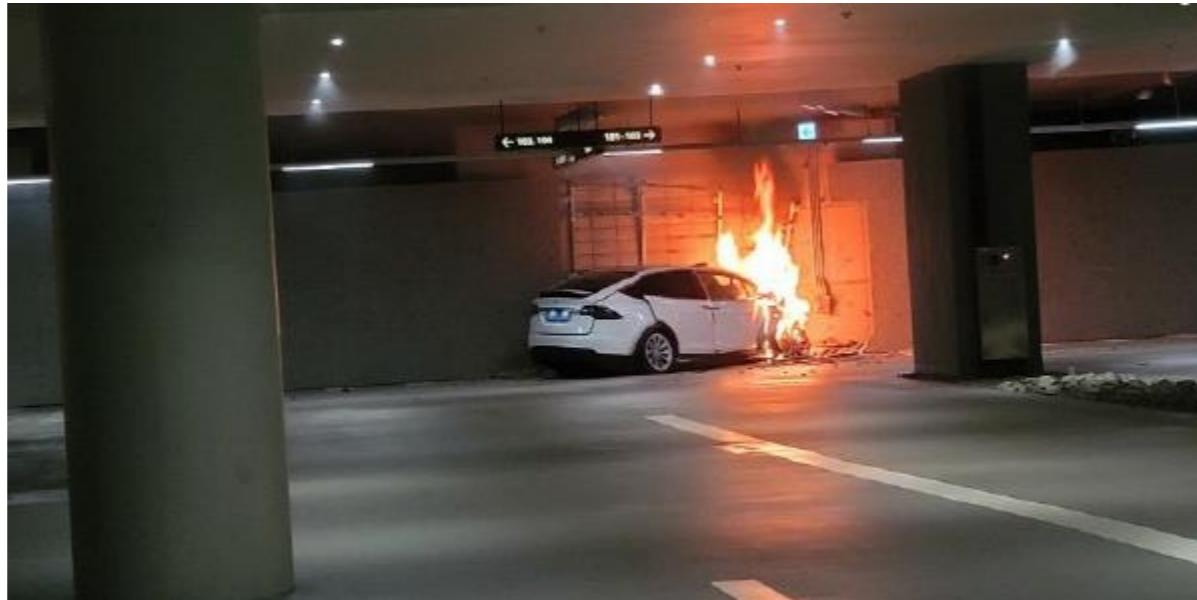
- Toyota pauses Paralympics self-driving buses after one hits visually impaired athlete



<https://www.theguardian.com/technology/2021/aug/28/toyota-pauses-paralympics-self-driving-buses-after-one-hits-visually-impaired-athlete>

Faults

- South Korea investigates fatal crash of Tesla Model X



<https://www.reuters.com/article/us-tesla-southkorea-crash-idINKBN28K0LA>

Faults

- And many others...



A threat for employees?





Office: #432, Dayang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University

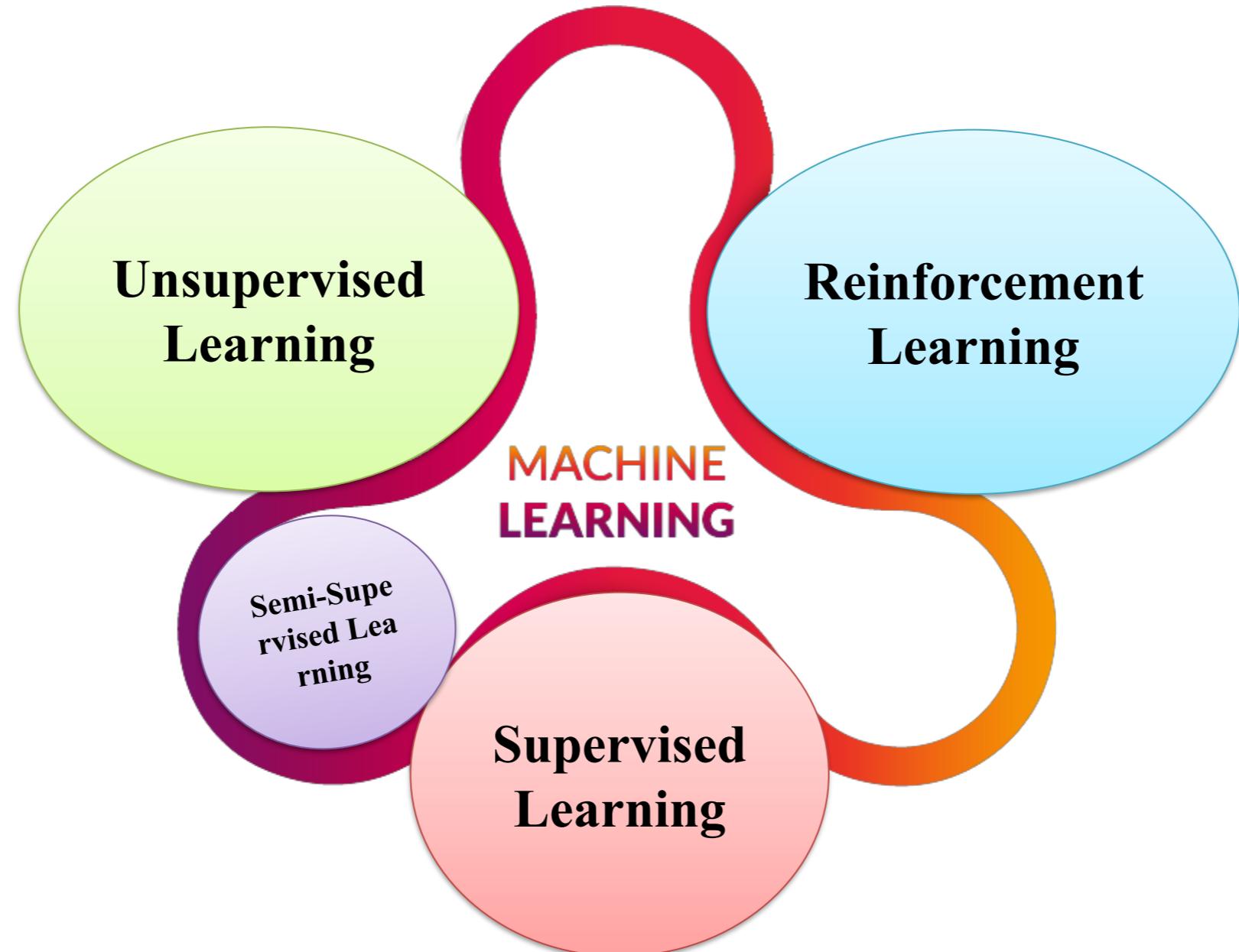


- Introduction to AI
- The History of AI
- Python
- Search
- Informed Search
- Beyond Classical Search
- Adversarial Search
- Constraint Satisfaction Problems
- Fuzzy Logic
- Introduction to Machine Learning
- **Types of Machine Learning**

Overview of Machine Learning



Types of Learning





Machine Learning

- Human being learns from past experiences.
- Machines have no “**experiences**”.
- Machines learn from **data**
- ML Goal:
 - Learn a **target function** that can be used to predict the values of a discrete class attribute,
 - e.g., approve or not-approved, and high-risk or low risk.
 - Known as **Supervised learning, classification**, or **inductive learning**.

Data

- **Data:**
 - A set of data records (also called examples, instances or cases)
 - Described by k attributes: A_1, A_2, \dots, A_k .
 - Class: each example is labelled with a pre-defined class.
- **Goal:**
 - To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

Use-case

Emergency Reception

- An **emergency room** in a hospital measures 17 variables (e.g., blood pressure, age, etc.) of newly admitted patients.
- **Decision:** whether to put a new patient in an intensive-care unit.
- **Problem:** to predict high-risk patients and discriminate them from low-risk patients.

Use-case

Credit Card

- A credit card company receives thousands of applications for new cards.
- Each application contains information about an applicant,
 - Age
 - Marital status
 - Annual salary
 - Outstanding debts
 - Credit rating
 - etc.
- **Problem:**
 - To decide whether an application should be approved,
 - or to classify applications into two categories, approved and not approved.

ML Tasks

(1) Classification

- assigning a category to each item

(2) Regression

- predicting a value for each item

(3) Ranking

- learning to order items.

(4) Clustering

- partitioning items into homogeneous subsets

(5) Dimensionality reduction (manifold learning)

- transforming a representation of items into a lower-dimensional one

(6) Anomaly Detection

Ingredients of ML

- **Training examples**
- **Feature vectors (patterns)**
- **Labels**
- **Hyperparameters**
 - Free parameters not determined by the learning algorithm but specified as input to the learning algorithm
- **Validation sample**
 - For tuning the hyperparameters Test sample
- **Hypothesis set**
 - A set of functions mapping to the set of labels

Types of Machine Learning



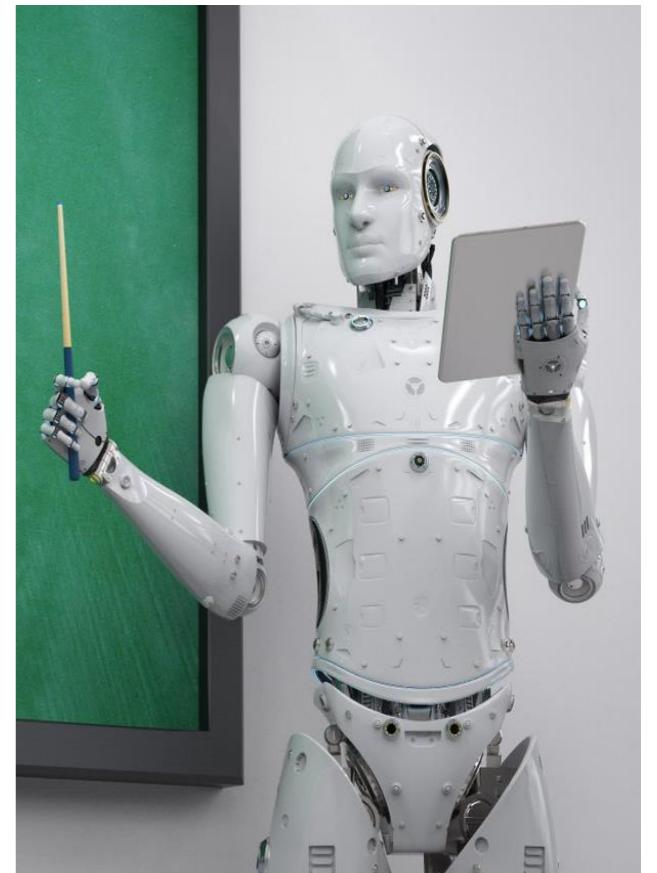
- **Supervised Learning (SL)**
 - Learn a mapping from the input to an output using a set of labeled examples
- **Unsupervised Learning (UL)**
 - Find the regularities of data using a set of unlabeled examples
- **Semi-supervised Learning (SSL)**
 - The training sample consisting of both labeled and unlabeled data

Types of Machine Learning



- **Reinforcement Learning (RL)**
 - During learning, the correct answers are not provided but hints or delayed rewards
- **On-line Learning**
 - Training and testing phases are intermixed.
- **Active Learning**
 - The learner adaptively or interactively collects training examples

- Try to **model** relationships and dependencies between the target prediction output and the input features
- Used to **predict** the output values for new data based on those relationships which it learned from the previous data sets.



- **Goal:** to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

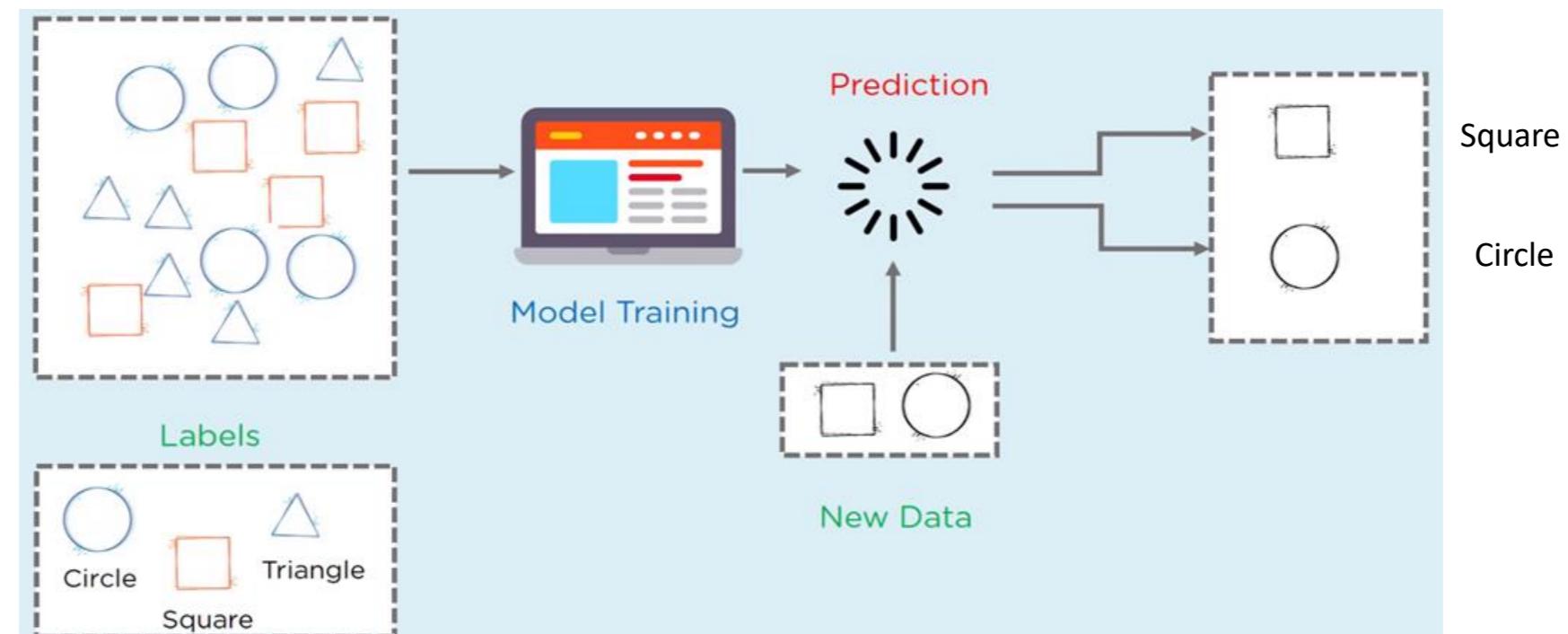
$$Y = f(X)$$

X : input variables

Y : an output variable

$f(\cdot)$: an algorithm to learn the mapping function from the input to the output.

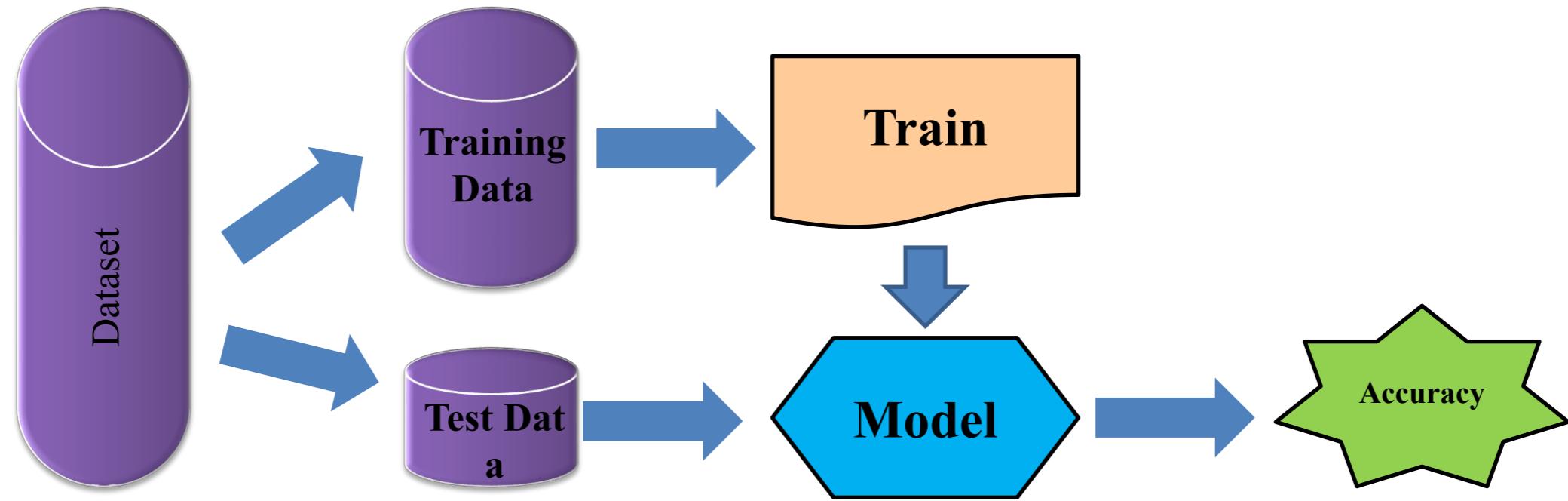
- SL enable machines to **classify / predict** objects, problems or situations based on labeled data fed to the machine.



SL process: two steps

- **Learning (training):** Learn a model using the **training data**
- **Testing:** Test the model using **unseen test data** to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classification}}{\text{total number of test cases}}$$





What is Learning?

Given

a **data set** D ,

a **task** T , and

a **performance measure** M ,

a computer system is said to **learn** from D to perform the task T if after learning the system's performance on T improves as measured by M .

SL problems

- **Classification:**
 - Output: a category
 - e.g., “red” or “blue” or “disease” and “no disease”.
- **Regression:**
 - Output: a real value,
 - e.g., “dollars” or “weight”.

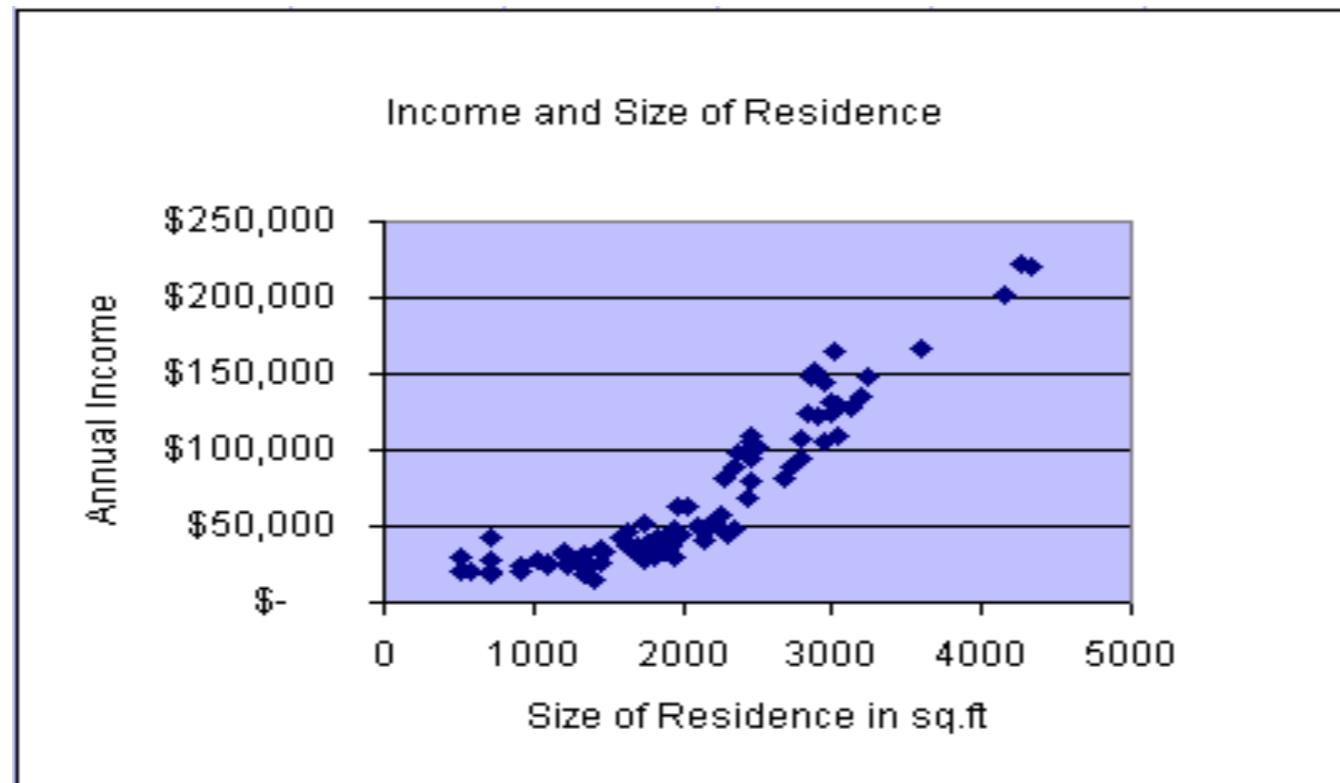
Algorithms

• Regression

- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Statistical Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)
- **Random forest** for classification and regression problems
- **Support vector machines** for classification problems
- **Artificial neural networks (ANN)**
- **Deep neural networks (DNN)**
- **Linear discriminant analysis**
- **Decision trees**
- **Similarity learning**
- **Bayesian logic**
- **Supervised classifier**
- **Probabilistic Learning**

Linear Regression

- Maps an input to an output based on example input-output pairs,
- Predicts continuous valued output.



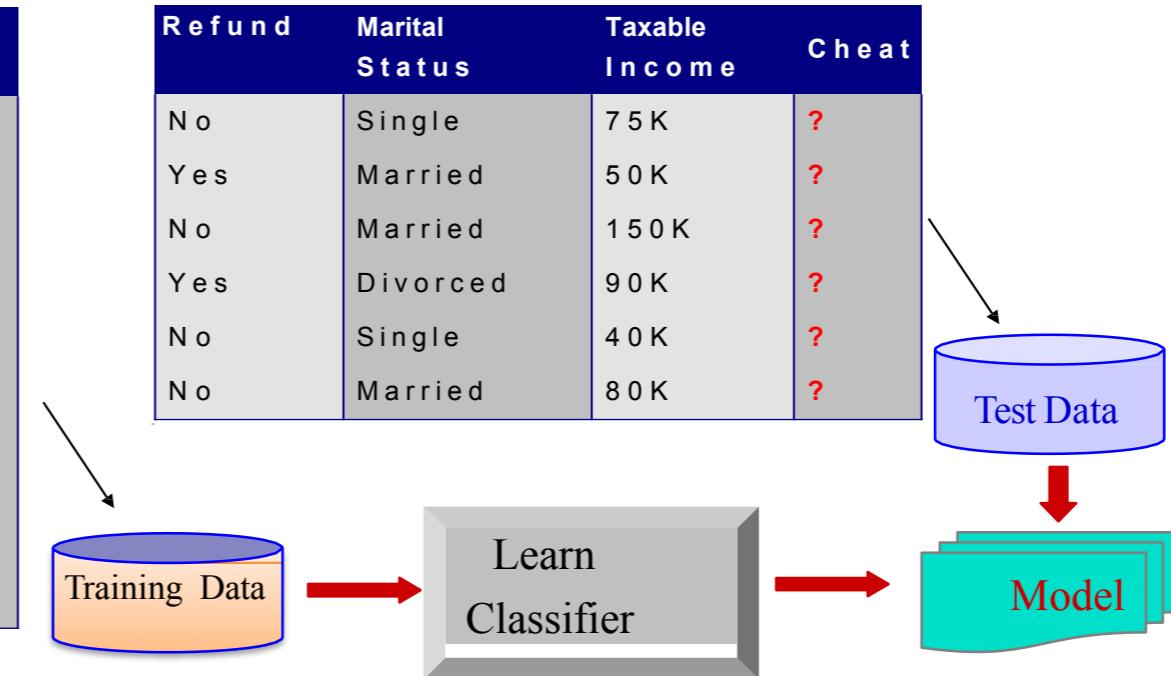
Classification

- Given a collection of records (**training set**)
 - Each record contains a set of **attributes**, one of the attributes is the **class**.
- Find a **model** for class attribute as a function of the values of other attributes.
- Goal: **previously_unseen** records should be assigned a class as accurately as possible.
 - A **test set** is used to determine the accuracy of the model.
 - Usually, the given data set is divided into training and test sets
 - with **training set** used to **build the model** and **test set** used to **validate it**.

Classification

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

categorical categorical continuous class



Many classifiers to choose;

- Logistic regression
- Neural networks
- Naïve Bayes
- Bayesian network
- SVM
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- ...

Classification:

Application 1

- **Direct Marketing**
 - **Goal:** Reduce cost of **mailing** by **targeting** a set of consumers likely to buy a new cell-phone product.
 - **Approach:**
 - Use the data for a similar product introduced before.
 - We know which customers decided to buy and which decided otherwise. This $\{buy, don't\;buy\}$ decision forms the *class attribute*.
 - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
 - Type of business, where they stay, how much they earn, etc.
 - Use this information as input attributes to learn a classifier model.

Classification: Application 2

- **Fraud Detection**
 - **Goal:** Predict fraudulent cases in credit card transactions.
 - **Approach:**
 - Use credit card transactions and the information on its account-holder as attributes.
 - When does a customer buy, what does he buy, how often he pays on time, etc.
 - Label past transactions as fraud or fair transactions. This forms the class attribute.
 - Learn a model for the class of the transactions.
 - Use this model to detect fraud by observing credit card transactions on an account.

Summary

- SL is best suited to problems where there is a **set of available reference points** or a **ground truth** with which to train the algorithm.

But those aren't always available!!!

- UL algorithms try to use techniques on the input data to **mine for rules, detect patterns, and summarize and group the data points** which help in deriving meaningful insights and describe the data better to the users.



Learn patterns from (unlabeled) data.

- **UL** is where you only have input data (X) and no corresponding output variables.
- The goal for UL is to **model** the underlying structure or distribution in the data in order to learn more about the data.

Problems

- **Clustering:** where you want to discover the inherent groupings in the data, such as *grouping customers by purchasing behavior*.
- **Association:** where you want to discover rules that describe large portions of your data, such as *people that buy X also tend to buy Y*.

Approaches

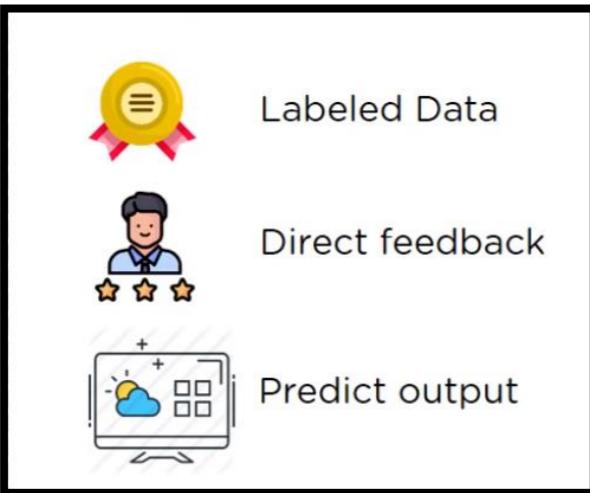
- Clustering (similarity-based)
- Density estimation (e.g., EM algorithm)
- Performance Tasks
- Understanding and visualization
- Anomaly detection
- Information retrieval
- Data compression

Comparison

Supervised

vs.

Unsupervised



Algorithms

- K-means clustering,
- Hierarchical clustering,
- Unsupervised soft-clustering,
- Affinity propagation clustering
- Self-organizing map learning
- Autoencoders
- Adversarial autoencoders
- Non-parametric Bayesian Learning
- Generative Deep Neural networks (GDNN)
- Apriori,
- PCA
- Mixture model
- Gaussian mixture model, Expectation Maximization (EM)
- Dirichlet process

Clustering

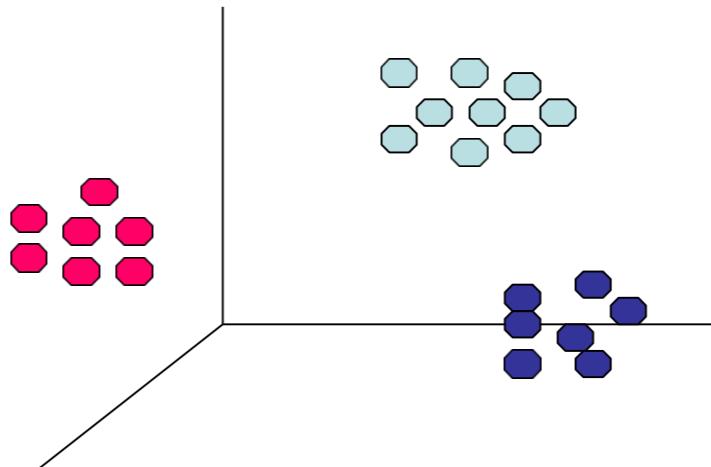
- Clustering is the task of partitioning the dataset into groups, called clusters.
- The goal is to split up the data in such a way that points within single cluster are very similar and points in different clusters are different.
- It determines grouping among unlabeled data.
- Given **a set of data points**, each having a **set of attributes**, and a **similarity measure among them**, find clusters such that
 - Data points in one cluster are more similar to one another.
 - Data points in separate clusters are less similar to one another.
- Similarity Measures:
 - **Euclidean Distance** if attributes are continuous.
 - Other Problem-specific Measures.

Clustering

- **Euclidean Distance**-based Clustering in 3-D space.

Intracluster distances
are minimized

Intercluster distances
are maximized



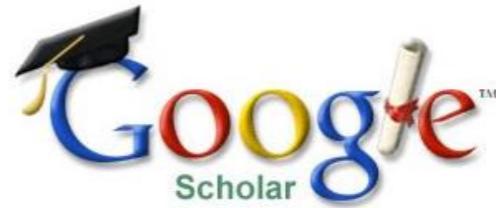
Clustering Strategies

- **K-means**
 - Iteratively re-assign points to the nearest cluster center.
- **Agglomerative clustering**
 - Start with each point as its own cluster and iteratively merge the closest clusters.
- **Mean-shift clustering**
 - Estimate modes of probability density function.
- **Spectral clustering**
 - Split the nodes in a graph based on assigned links with similarity weights.

Clustering: Application 1

- **Market Segmentation:**
 - **Goal:** subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.
 - **Approach:**
 - Collect different attributes of customers based on their geographical and lifestyle related information.
 - Find clusters of similar customers.
 - Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

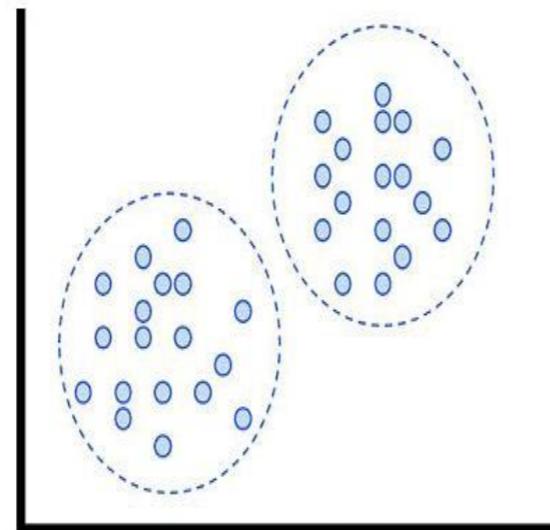


- **Document Clustering:**
 - **Goal:** To find groups of documents that are similar to each other based on the important terms appearing in them.
 - **Approach:** To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.
 - **Gain:** Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents.
 - **Gain:** Information retrieval can utilize the clusters to relate a new document or search term to clustered documents.

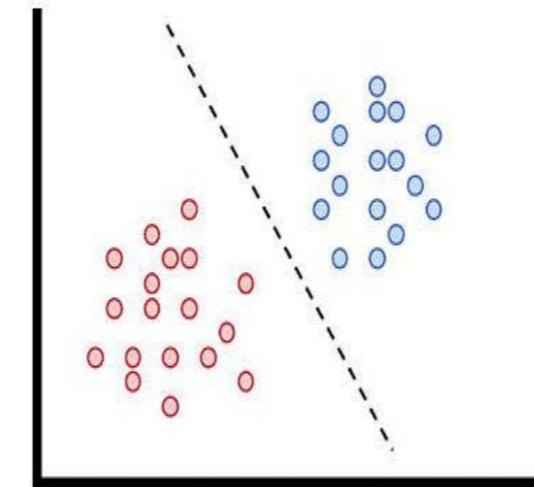
Comparison

Classification vs. Clustering

- **Classification**
 - Used in SL technique where predefined labels are assigned to instances by properties.
- **Clustering**
 - Used in UL where similar instances are grouped, based on their features or properties.



Clustering



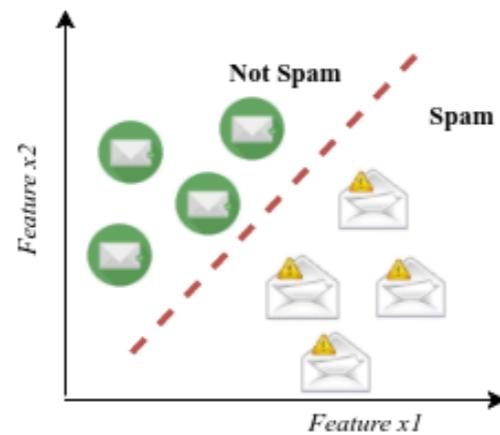
Classification

Comparison



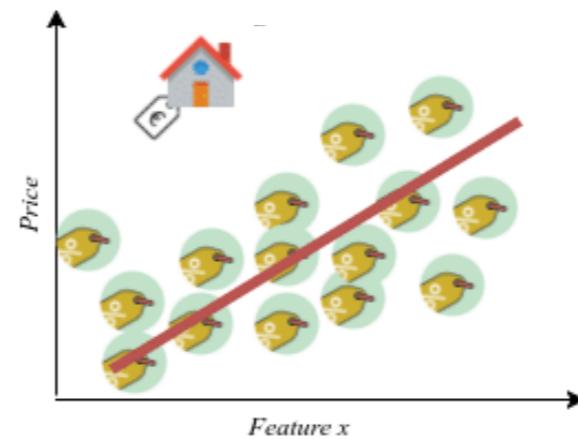
Regression	Classification	Clustering
SL	SL	UL
map an input to an output based on example input-output pairs		partitioning the dataset into groups, e.g. clusters
predicts continuous valued output	predicts discrete number of values	Split up the data in such a way that points within single cluster are very similar and points in different clusters are different.
used to predict the numeric data instead of labels.	data is categorized under different labels	determines grouping among unlabeled data.
identify the distribution trends based on the available data or historic data	the labels are predicted for the data.	

Classification



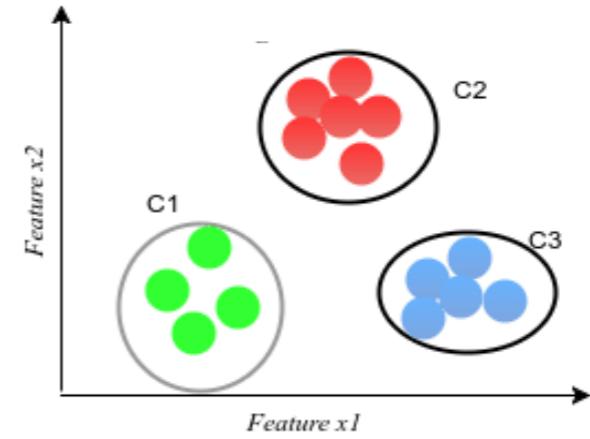
Spam filtering as a classification task

Regression



House price estimation as a regression task

Clustering



customers are grouped into three different categories based on their purchasing behavior.

Association Rule Discovery

- Given a set of **records** each of which contain some number of items from a given collection;
 - Produce **dependency rules** which will **predict** occurrence of an item based on occurrences of other items.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

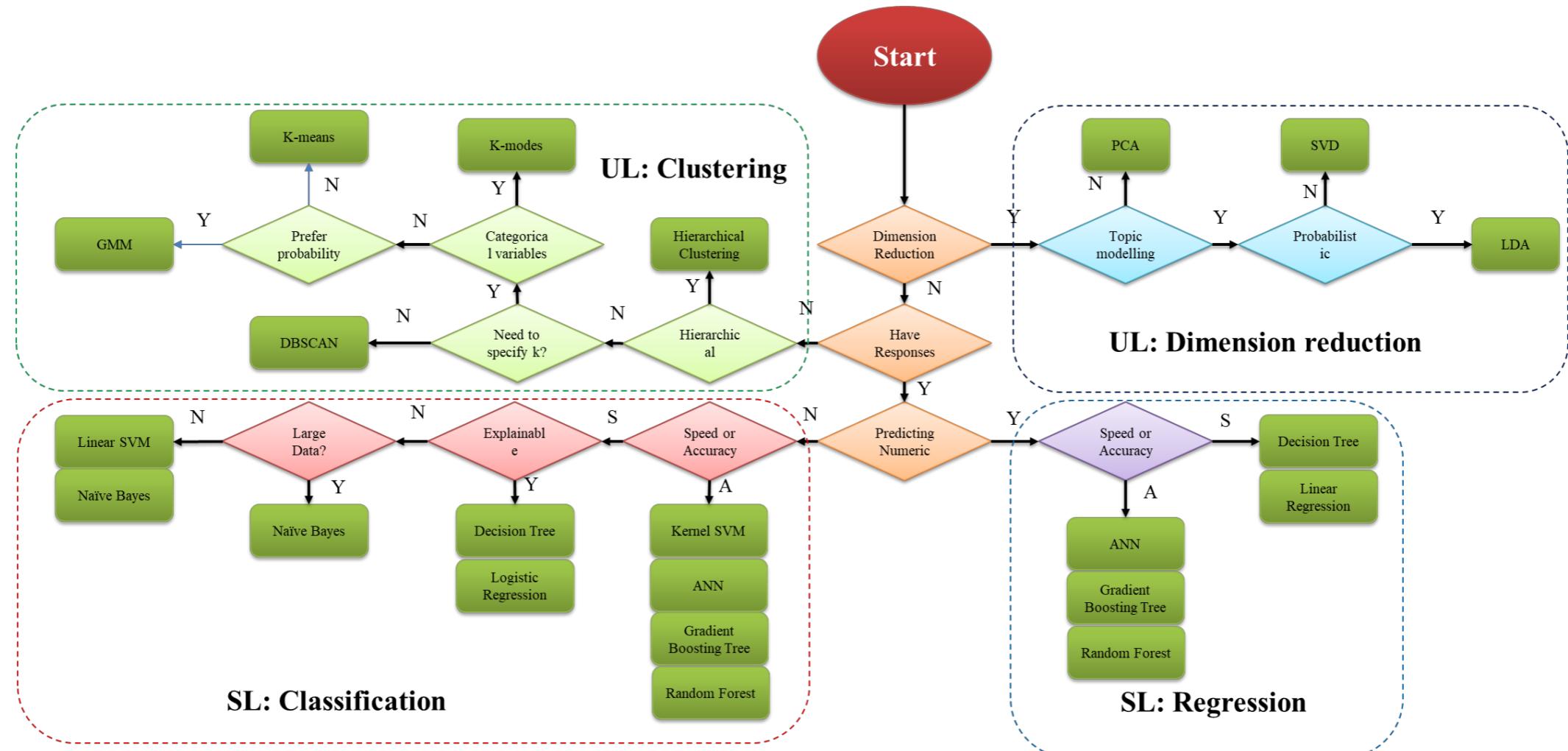
Rules Discovered:

{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

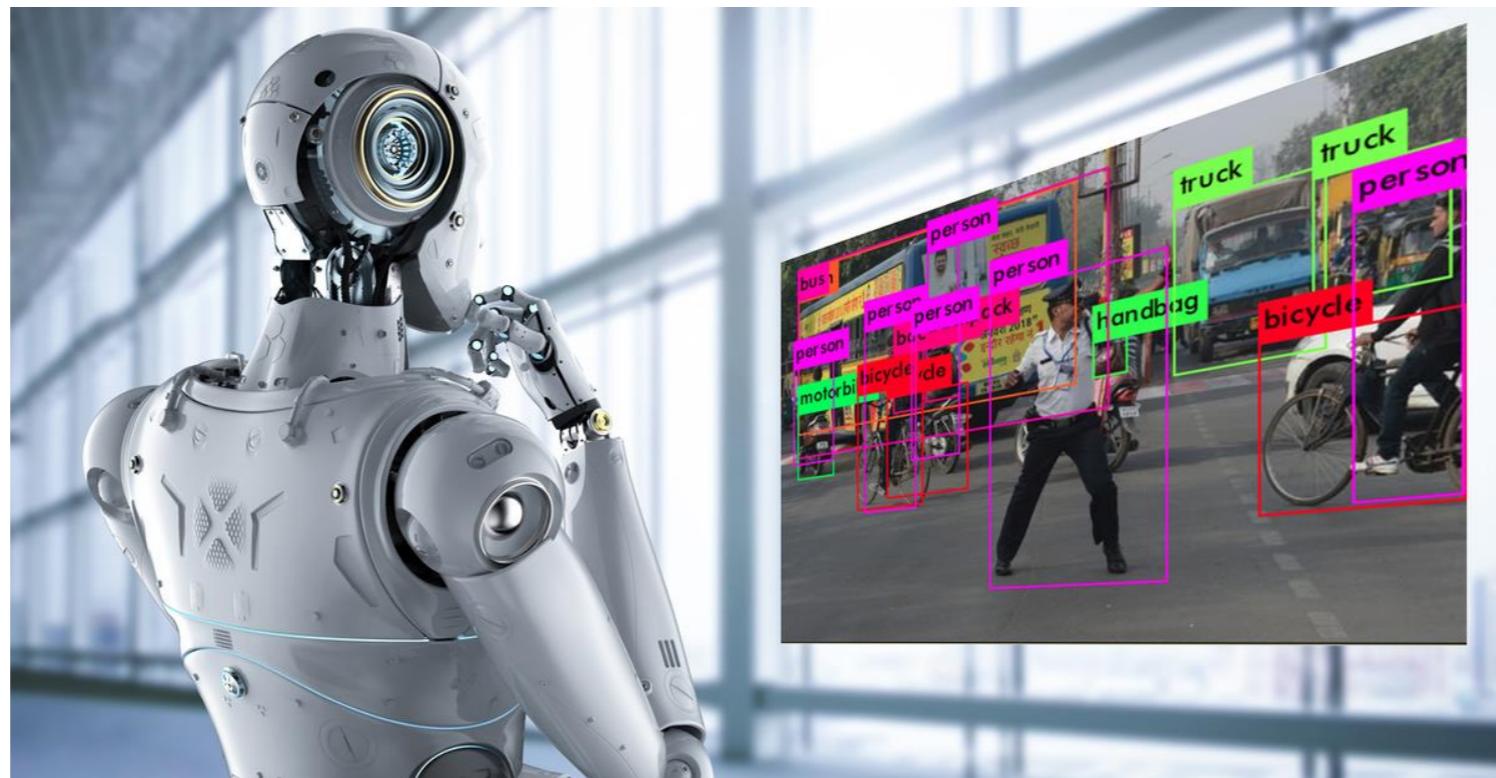
Association Rule Discovery: Application 2

- **Supermarket shelf management.**
 - **Goal:** To identify items that are bought together by sufficiently many customers.
 - **Approach:** Process the point-of-sale data collected with barcode scanners to find dependencies among items.
 - A classic rule--
 - If a customer buys diaper and milk, then he is very likely to buy beer.
 - So, don't be surprised if you find six-packs stacked next to diapers!

SL and UL Cheat-sheet



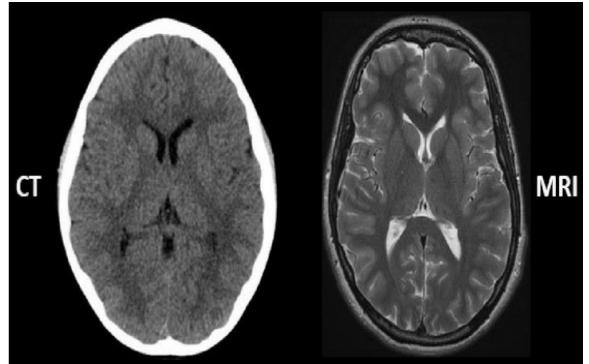
- A training dataset with both **labeled** and **unlabeled** data.
- This method is particularly useful when extracting relevant features from the data is difficult,
- and labeling examples is a time-intensive task for experts.



- **Problems:** where you have a large amount of input data (X) and only some of the data is labeled (Y).
- These problems sit in between both SL and UL.
- A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.

Applications

- **Medical images**
 - e.g. CT scans or MRIs
- A trained radiologist can go through and label a small subset of scans for tumors or diseases.
- It would be too time-intensive and costly to manually label all the scans
- but the deep learning network can still benefit from the small proportion of labeled data and improve its accuracy compared to a fully unsupervised model.



Reinforcement learning

- An agent learns how to behave in an environment by performing actions and seeing the results.



- **Video games** are full of reinforcement cues.
- Complete a level and earn a badge.
- Defeat the bad guy in a certain number of moves and earn a bonus. Step into a trap — game over.
- These cues help players learn how to improve their performance for the next game.
- Without this feedback, they would just take random actions around a game environment in the hopes of advancing to the next level.



Comparison



SL

- Labeled dataset
- Establish relationship between input and output
- Generate output for new data points
- Reliable models but expensive and limited
- Classification: Associative classifiers, Decision Trees, Instance Learning, Bayesian Learning, Kernel machines, Neural Networks, Genetic Algorithms, etc.
- Regression: Linear Regression, ...

UL

- Unlabeled dataset
- Decipher structure of the data
- Output attributes are not defined
- Clustering: K-means, DBScan, Hierarchical algorithms, Self Organizing Maps, etc.
- Associations: Apriori, FP-Growth, ...

RL

- Maximizing the rewards from the results
- Aka. credit assessment learning
- Additional decision about rewards
- Explore the tradeoff between exploring and exploiting the data

Topics:

- **Policies**: what actions should an agent take in a particular situation
- **Utility estimation**: how good is a state (used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
 - Game playing
 - Robot in a maze
 - Multiple agents, partial observability, ...

Steps

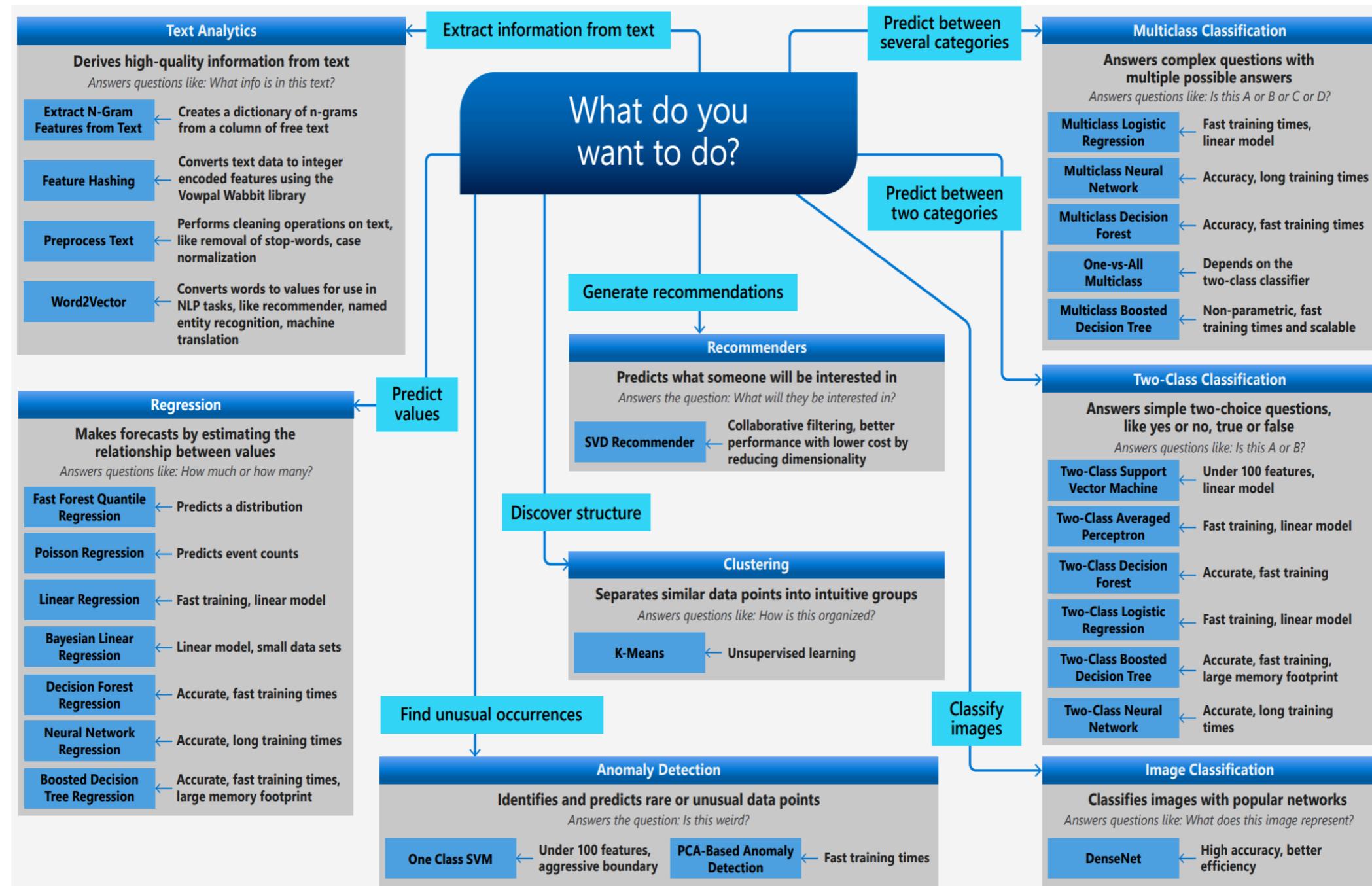
- In order to produce intelligent programs (also called agents), reinforcement learning goes through the following steps:
 - **Input state** is observed by the agent.
 - **Decision making function** is used to make the agent perform an action.
 - After the action is performed, the agent receives **reward** or reinforcement from the environment.
 - The **state-action pair information** about the reward is stored.



Algorithms

- Q-Learning
- R-Learning
- MDP / POMDP
- Temporal Difference (TD)
- Deep Adversarial Networks
- Multi-armed bandit
- Actor-critic
- Deep reinforcement learning (DRL)

ML Algorithms Cheat Sheet





Office: #432, Daeyang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



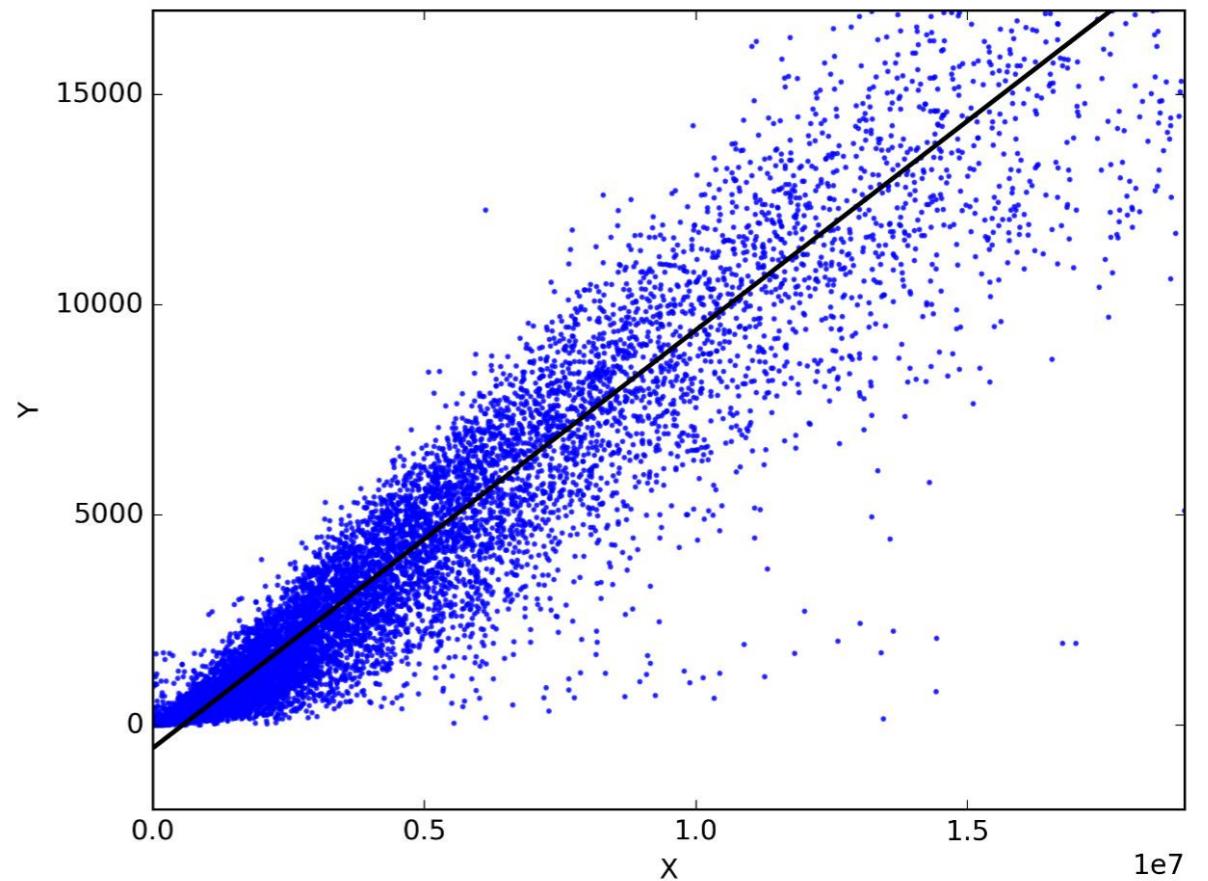


Course Outline

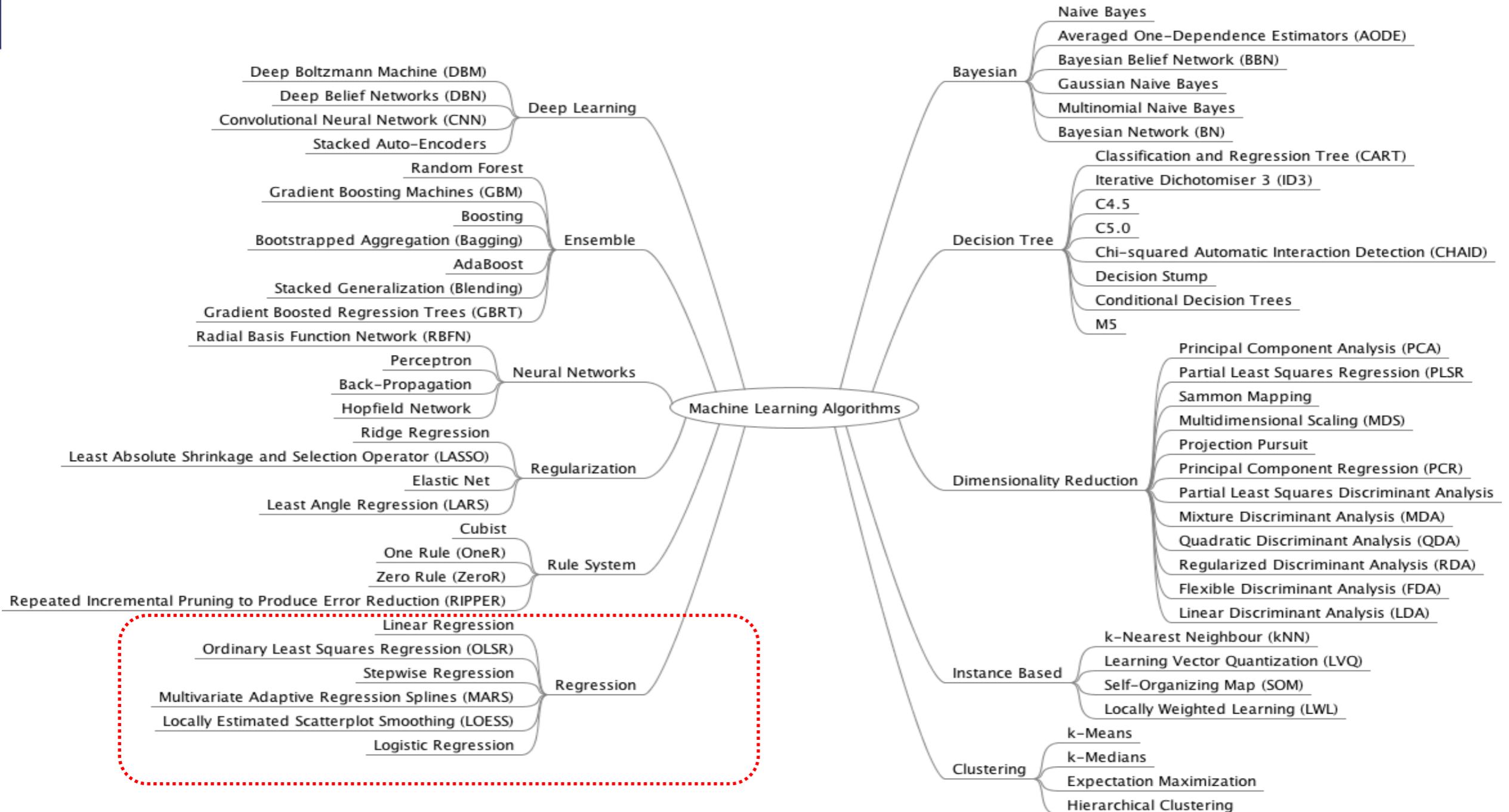
- Introduction to AI
- The History of AI
- Python
- Search
- Informed Search
- Beyond Classical Search
- Adversarial Search
- Constraint Satisfaction Problems
- Fuzzy Logic
- Introduction to Machine Learning
- Types of Machine Learning
- **SL: Linear Regression**

LINEAR REGRESSION

- Definition
- Applications areas
- Use cases
- Selection criteria
- Mathematical representation
- Errors
- Model Evaluation
- Gradient Descent
- Overfitting
- Regularization
- Multivariable Linear Regression



Machine Learning Algorithms



Regression

- **Regression analysis:** a form of predictive modelling technique.
 - Investigates the relationship between a **dependent** (target) and **independent variable (s)** (predictor).
- Types:
 - **Linear Regression**
 - Logistic Regression
 - Polynomial Regression
 - Stepwise Regression
 - Ridge Regression
 - Lasso Regression
 - ElasticNet Regression
 - ...

Linear Regression

- **Linear Regression:** quantifies the relationship between one or more **predictor variables** and one **outcome variable**.
- a.k.a **multiple regression, multivariate regression, ordinary least squares (OLS)**.
- **Types:**
 - **Simple Linear Regression:**
 - Input(X): is a single variable
 - **Multiple Linear Regression:**
 - Multiple input variables(X)

Application areas

- Determining the strength of predictors,
- Forecasting and effect,
- Trend forecasting,
- ...



Use-cases

- Evaluating **sales** and estimating the **progress**
- Understanding the **price change** impacting your **business**
- Analyzing **risk factor** in financial services and insurance domain



Use-cases

- Generating insights on **consumer behavior**, profitability, and other business factors.
- Evaluation of **trends**; making **estimates**, and **forecasts**.
- Determining **marketing effectiveness**, pricing, and promotions on sales of a product.
- Assessment of **risk** in **financial** services and insurance domain.
- Studying **engine performance** from test data in automobiles.
- Calculating **causal relationships** between parameters in biological systems.
- Conducting **market research** studies and customer survey results analysis.
- **Astronomical** data analysis.
- Predicting **house prices** with the increase in sizes of houses.

Selection Criteria

- **Classification and regression capabilities,**
 - Regression models predict a continuous variable, such as the sales made on a day or predict temperature of a city.
- **Error rates**
 - Linear regression is weaker than other algorithms in terms of reducing error rates.
- **Data compatibilities,**
 - Linear regression relies on continuous data to build regression capabilities.
- **Data quality**
 - Each missing value removes one data point that could optimize the regression. In simple linear regression, outliers can significantly disrupt the outcomes.
- **Computational complexity,**
 - Linear regression is often not computationally expensive, compared to decision trees and clustering algorithms.
- **Comprehensible and transparent**
 - Linear regression can be represented by simpler mathematical notations to anyone and can be understood easily.

Mathematical Representation

- **Linear Regression** is a **linear** model,
- e.g., a **linear relationship** between the input variables (x) and a single output variable ($h_{\theta}(x)$),

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

- The **output** varies linearly based upon the **input**.
 - “ $h_{\theta}(x)$ ”: the output determined by input “ x ”.
 - “ θ_1 ” (**slope**): determines how much value of “ x ” has **impact** on “ $h_{\theta}(x)$ ”.
 - “ θ_0 ”: the **constant**

Mathematical Representation

Imagine, we are predicting distance travelled ($h_{\theta}(x)$) from speed (x). Our linear regression model representation for this problem would be:

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

or

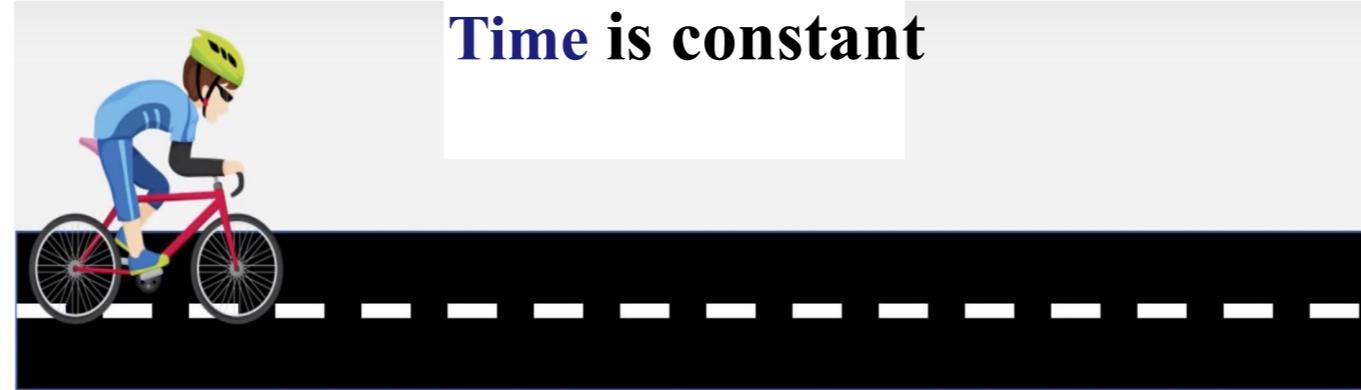
$$\text{Distance} = \theta_0 + \theta_1 * \text{speed}$$

θ_0 : coefficient

θ_1 : intercept

Time is constant

Speed = 10m/s



Distance = 36km

Speed = 20m/s

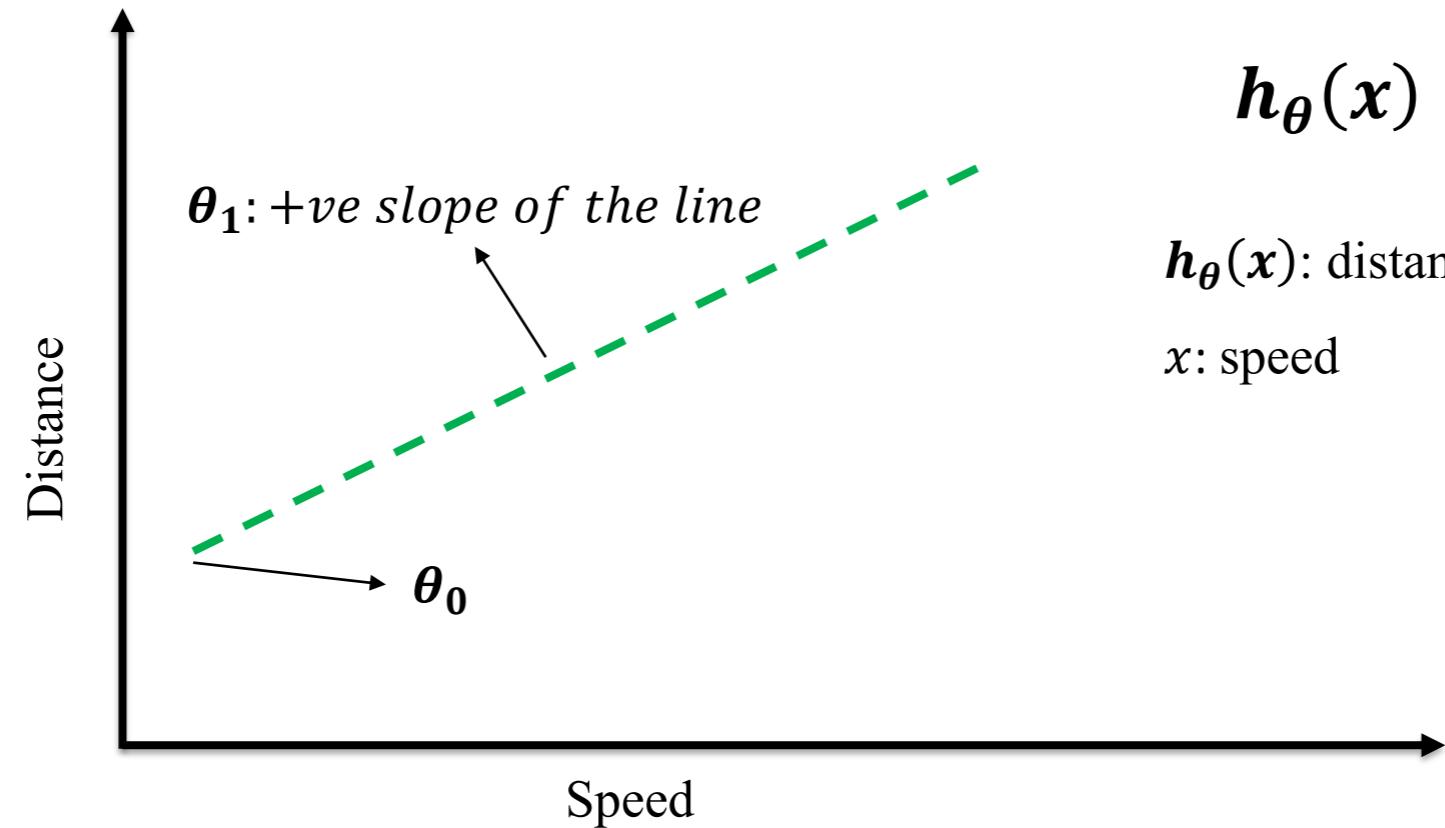


Distance = 72km

Speed = 30m/s



Distance = ?



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

$h_{\theta}(x)$: distance travelled in fixed interval of time

x : speed

- **Positive relationship:** as the speed increases, distance also increases

Distance is constant

Speed = 10m/s



Time = 100 s

Speed = 20m/s



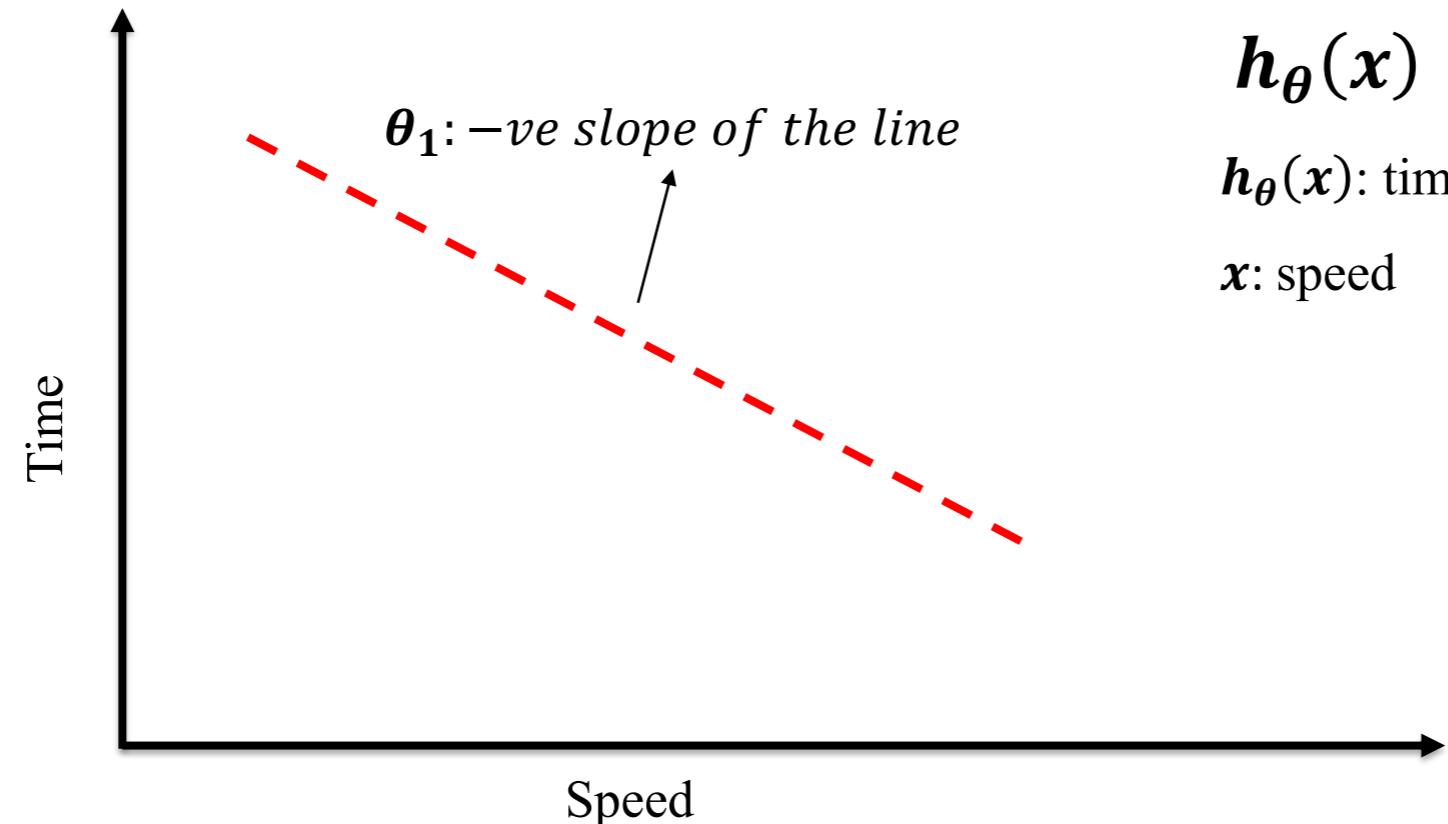
Time = 50 s

Speed = 30m/s



Time = ?

- If distance is assumed to be constant:



$$h_\theta(x) = \theta_0 + \theta_1x$$

$h_\theta(x)$: time taken to travel a fixed distance

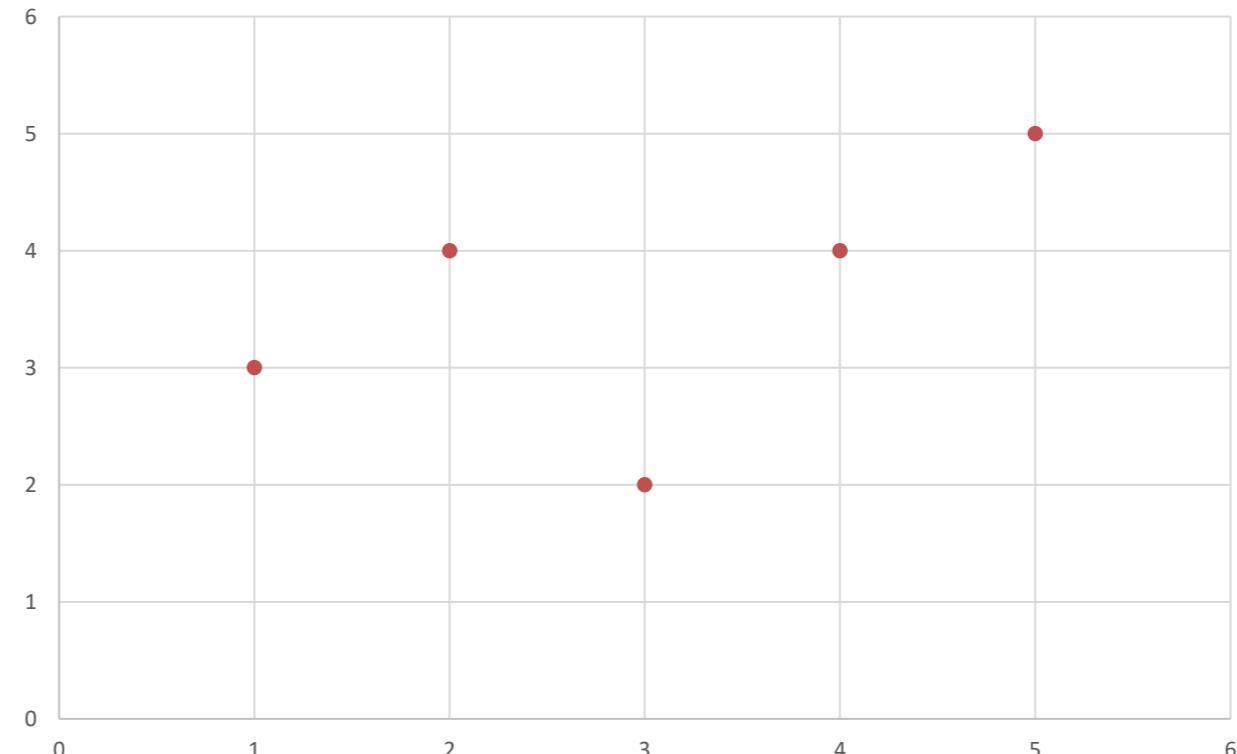
x : speed

- **Negative relationship:** as the speed increases, time decreases.

Mathematical representation

- Consider the following dataset:

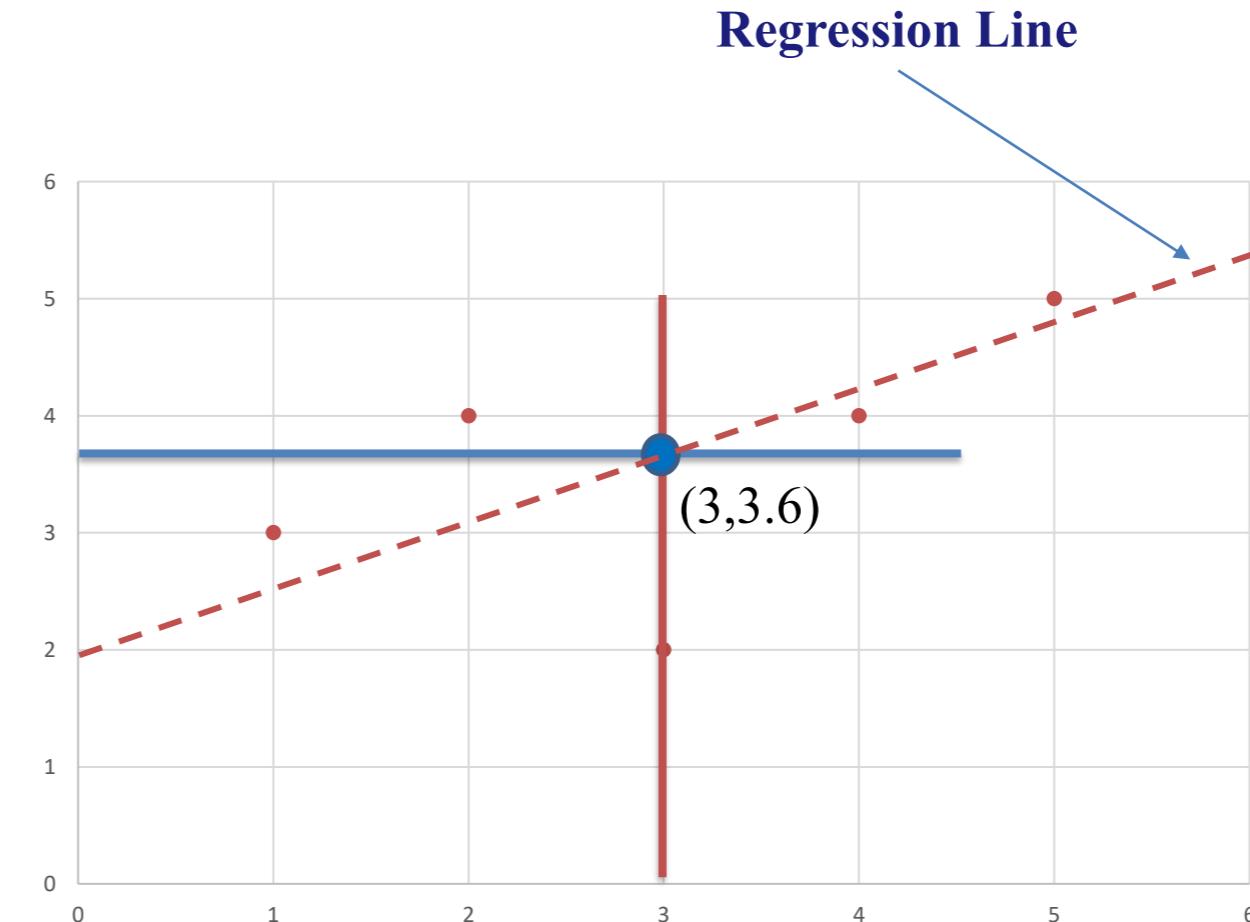
x	$h_{\theta}(x)$
1	3
2	4
3	2
4	4
5	5



Mathematical representation

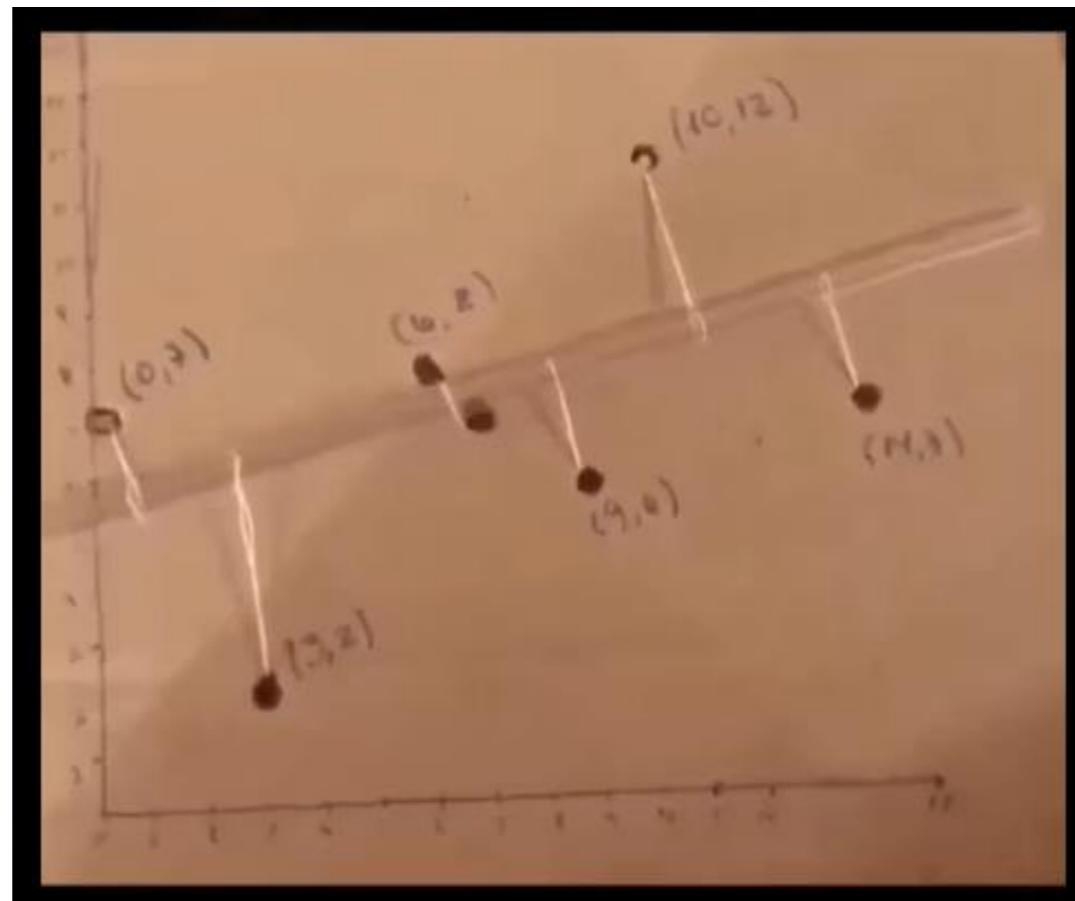
x	$h_{\theta}(x)$
1	3
2	4
3	2
4	4
5	5

Mean: 3 3.6



- **Objective:** find a line that passes the mean point and it well fit?

Linear Regression



Mathematical representation

- Regression equation to find the best line:

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

- **Learning procedure:** how do we find these coefficients?

Approaches:

- **Ordinary Least Square Method**
- **Gradient Descent Approach,**
- etc.

Mathematical representation

- **Least Square Method:** to find or predict the best fit line

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

$$\theta_1 = \frac{\sum(x - \bar{x})(h_{\theta}(x) - \bar{h}_{\theta}(x))}{\sum(x - \bar{x})^2}$$

Mathematical representation

x	$h_\theta(x)$	$x - \bar{x}$	$h_\theta(x) - \bar{h}_\theta(x)$	$(x - \bar{x})^2$	$(x - \bar{x})(h_\theta(x) - \bar{h}_\theta(x))$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8

$$\sum = 10$$

$$\sum = 4$$

$$\theta_1 = \frac{\sum (x - \bar{x})(h_\theta(x) - \bar{h}_\theta(x))}{\sum (x - \bar{x})^2} = \frac{4}{10}$$

Mathematical representation

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

$$\theta_1 = \frac{\sum(x - x_i)(h_{\theta}(x) - h_{\theta}(x^{(i)}))}{\sum(x - x_i)^2} = \frac{4}{10} = 0.4$$

$$h_{\theta}(x) = 0.4x + \theta_0$$

$$3.6 = 0.2 \times 3 + \theta_0$$

$$3.6 = 1.2 + \theta_0$$

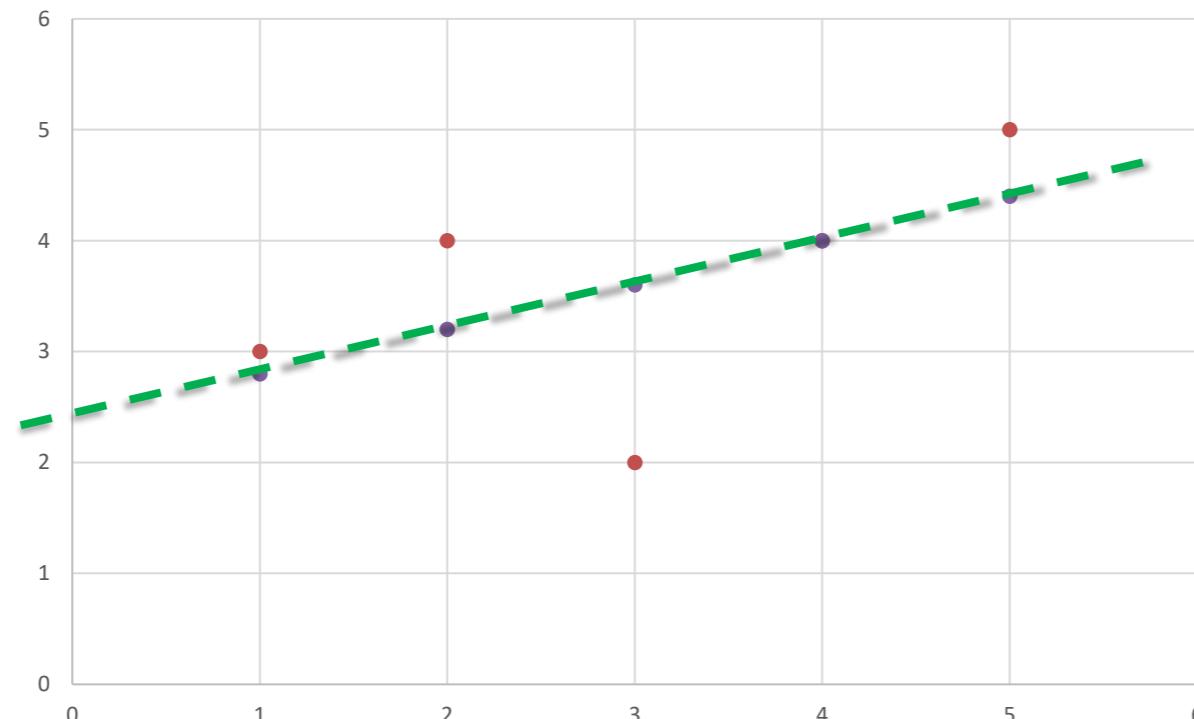
$$\theta_0 = 3.6 - 1.2$$

$$\theta_0 = 2.4$$

- The **regression line**: $h_{\theta}(x) = 0.4x + 2.4$

Mathematical representation

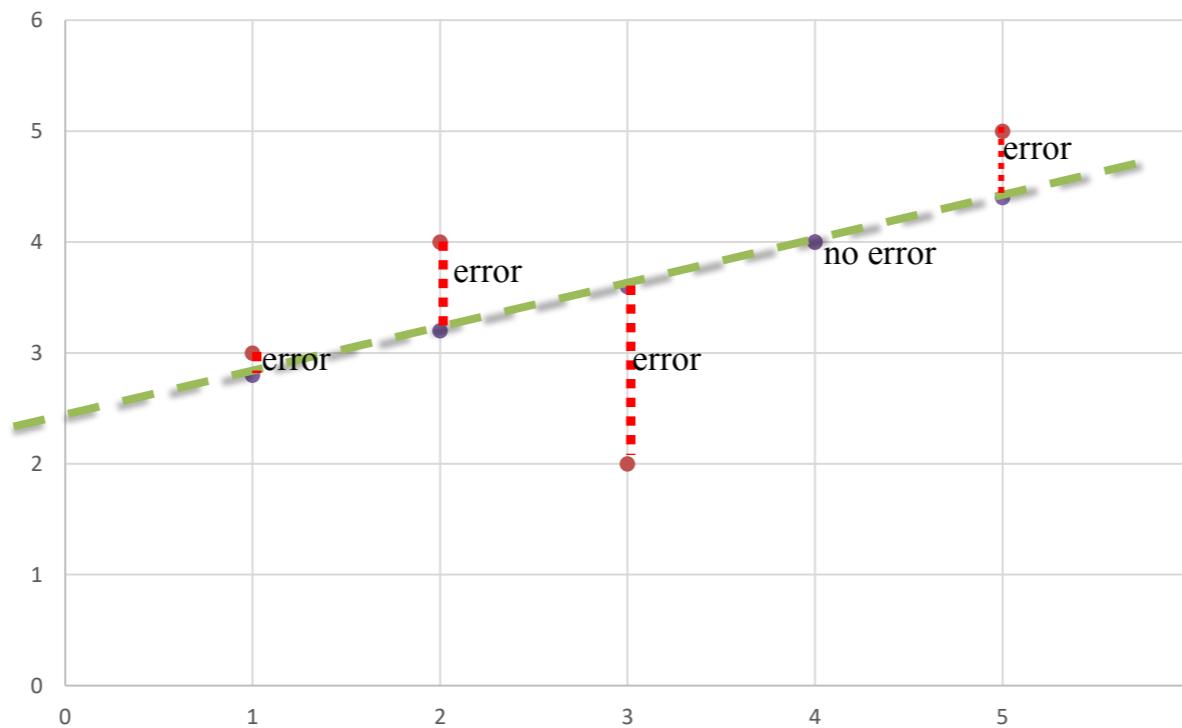
$$h_{\theta}(x) = 2.4 + (0.4 \times x)$$



x	$\hat{h}_{\theta}(x)$
1	2.8
2	3.2
3	3.6
4	4
5	4.4
6	4.8

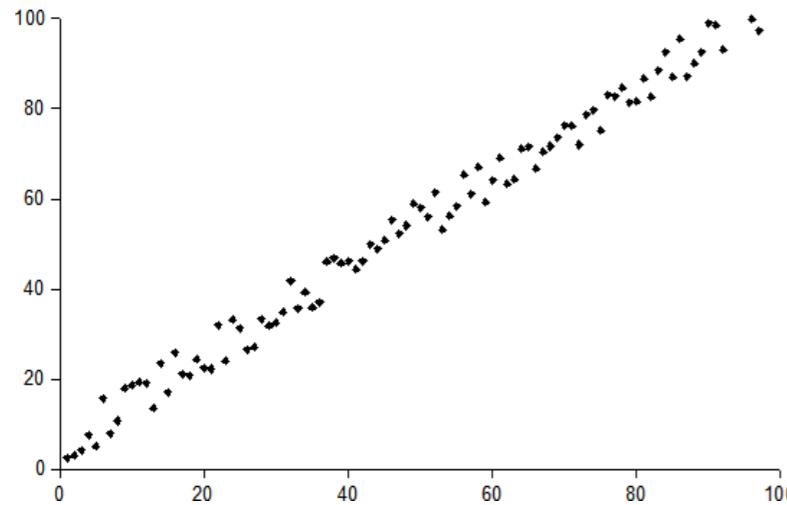
Mathematical representation

- Calculate the distance between the **actual** and **predicted values**.
- Objective: **to reduce the distances**.



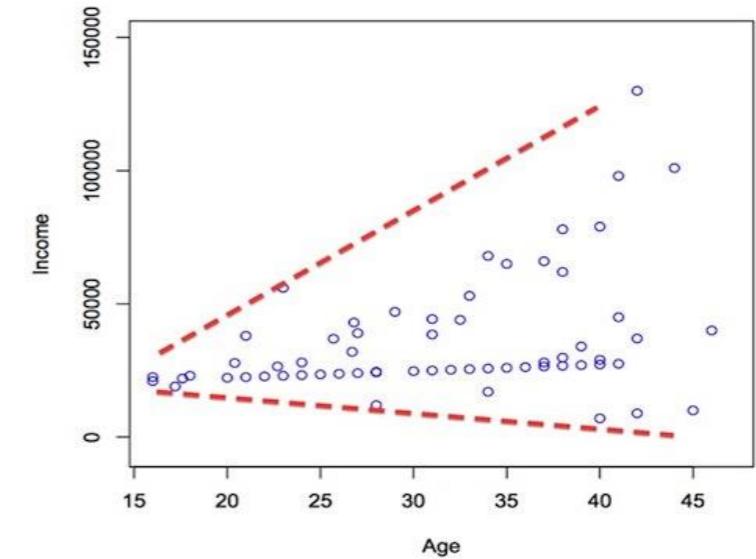
Errors

Homoscedasticity



Error has constant variance

Heteroscedasticity



Error has no constant variance

Possible reasons of arising Heteroscedasticity:

1. **Outliers:** The data set has a large range between the largest and the smallest observed values.
2. The model is not correctly specified.
3. The observations are mixed with different measures of scale.
4. Incorrect transformation of data is used to perform the regression.
5. Skewness in the distribution of a regressor, and may be some other sources.

Regression Line

Finding the best fit line

- The algorithm performs “ n ” number of iterations for different values of “ θ_1 ”.

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

- Computes the **predicated** value and compares it with the **actual value** and finds the distance, at each iteration.
- Best fit line:** the value of “ θ_1 ” for which the distance is minimum.

Evaluate the model

Evaluation Metrics:

- R-Square (R^2)
- Adjusted R-Square
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Mean Squared Percentage Error (MSPE)
- Mean Absolute Percentage Error (MAPE)
- Root Mean Squared Logarithmic Error (RMSLE)

Evaluation metrics

Mean Squared Error (MSE)

- MSE measures the average of the squares of the **errors**,
 - e.g., the average squared **difference** between the estimated values and the actual values.

$$MSE = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - \hat{h}_{\theta}(x^{(i)}))^2$$

$h_{\theta}(x^{(i)})$: the actual expected output

$\hat{h}_{\theta}(x^{(i)})$: the model's prediction.

Evaluation Metrics

R-squared

- A statistical measure of **how close** the data are to the fitted regression line.

$$R^2 = 1 - \frac{\sum_{i=1}^m (h_\theta(x^{(i)}) - \hat{h}_\theta(x^{(i)}))^2}{\sum_{i=1}^n (h_\theta(x^{(i)}) - \bar{h}_\theta(x^{(i)}))^2}$$

- R-squared ranges between 0 and 1:
 - 0: the model explains none of the variability of the response data around its mean.
 - 1: model explains all the variability of the response data around its mean.

“The higher the R-squared, the better the model fits your data. But not for all cases”

Evaluation Metrics

Adjusted R-squared

- Shows how well terms fit a curve or line but adjusts for the number of terms in a model.
- Considers the **marginal improvement** added by an additional term in the model.

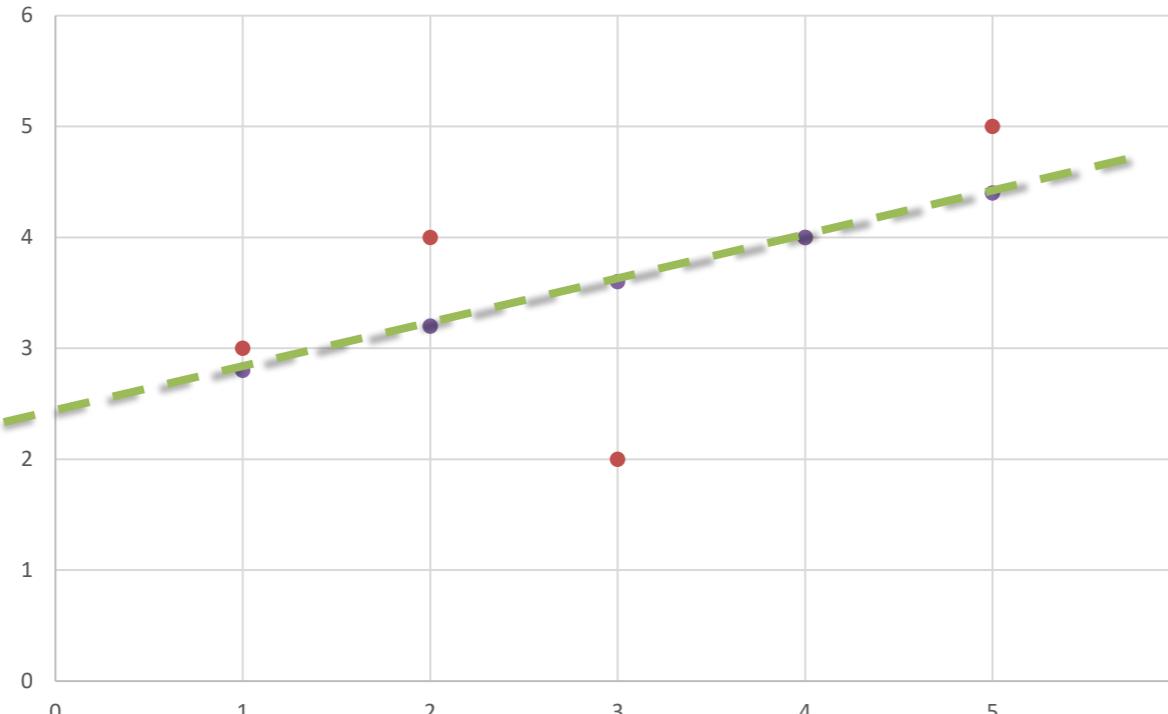
$$R_{adj}^2 = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1} \right]$$

- n : the total number of observations
- k : is the number of predictors

Note) adjusted R-squared will always be less than or equal to R-squared.

Test the model

- In case of **R-squared**;



x	$\hat{h}_\theta(x)$
1	2.8
2	3.2
3	3.6
4	4
5	4.4
6	4.8

$$R^2 = 1 - \frac{\sum (h_\theta(x^{(i)}) - \hat{h}_\theta(x^{(i)}))^2}{\sum (h_\theta(x^{(i)}) - \bar{h}_\theta(x^{(i)}))^2}$$

Test the model

$$R^2 = 1 - \frac{\sum_{i=1}^n (h_\theta(x^{(i)}) - \hat{h}_\theta(x^{(i)}))^2}{\sum_{i=1}^n (h_\theta(x^{(i)}) - \bar{h}_\theta(x^{(i)}))^2}$$

x	$h_\theta(x)$	$h_\theta(x) - \bar{h}_\theta(x)$	$(h_\theta(x) - \bar{h}_\theta(x))^2$	$\hat{h}_\theta(x)$	$h_\theta(x) - \hat{h}_\theta(x)$	$(h_\theta(x) - \hat{h}_\theta(x))^2$
1	3	-0.6	0.36	2.8	0.2	0.04
2	4	0.4	0.16	3.2	0.8	0.64
3	2	-1.6	2.56	3.6	-1.6	2.56
4	4	0.4	0.16	4.0	3.34	0.0
5	5	1.4	1.96	4.4	0.6	0.36

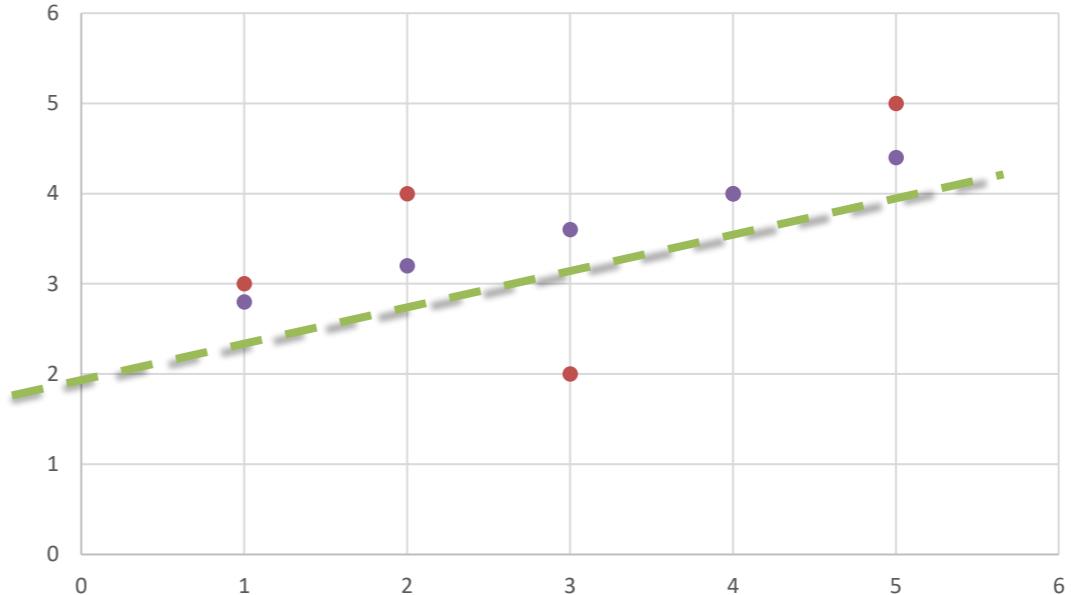
$$\sum = 5.2$$

$$\sum = 3.6$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (h_\theta(x^{(i)}) - \hat{h}_\theta(x^{(i)}))^2}{\sum_{i=1}^n (h_\theta(x^{(i)}) - \bar{h}_\theta(x^{(i)}))^2} = \frac{3.6}{5.2} \approx 0.69$$

Test the model

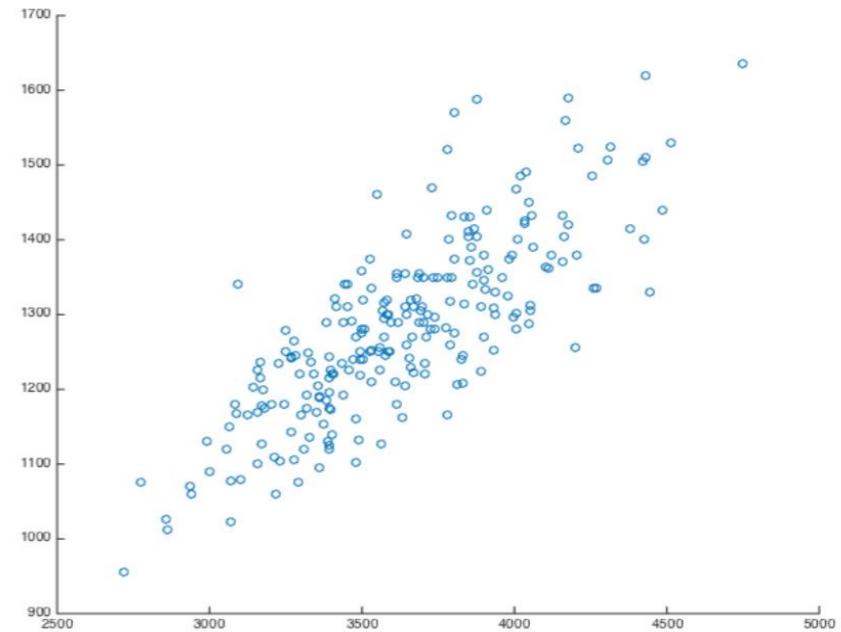
$$R^2 \approx 0.69$$



Conclusion: It means that this is **not a good fit**.

Least Square Method

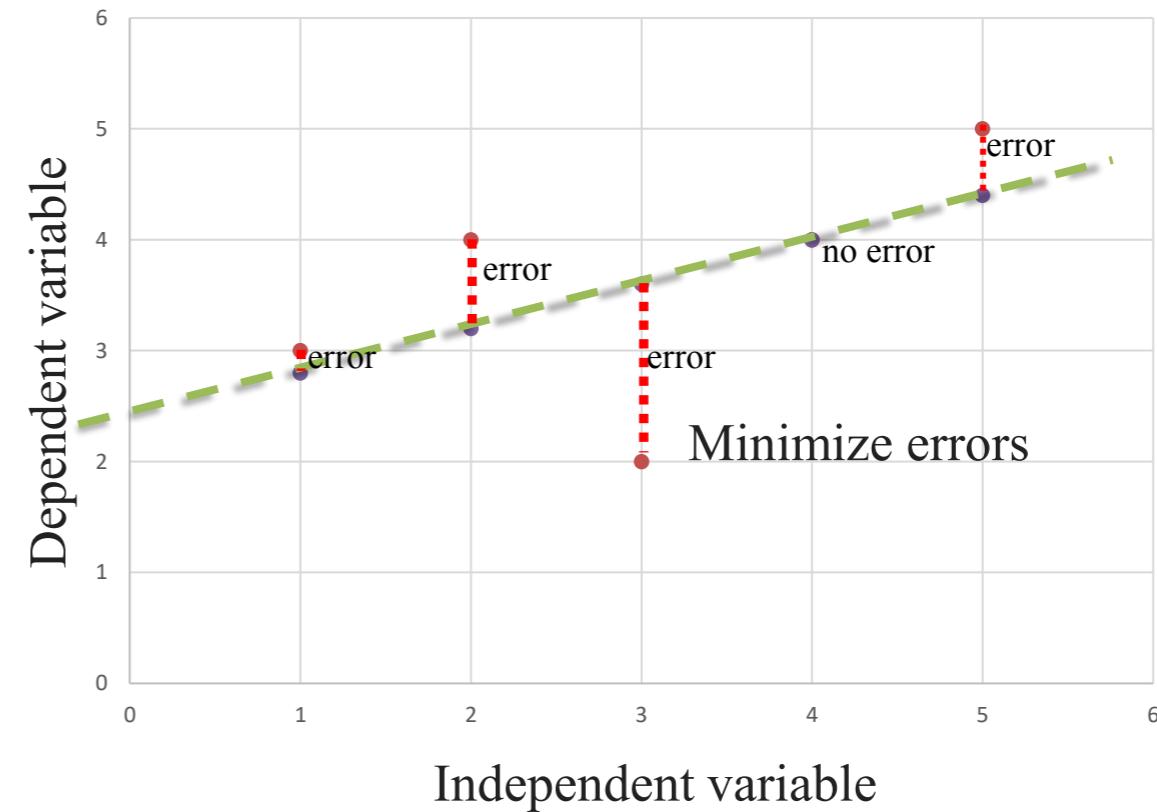
- Suppose we have this dataset:
- Is the data **consistent** with our model?
- Is our model **correct**?
- How can we **estimate** the parameters of our model?
- **Precision?**



Least Square Method

- We want to **minimize** the error of our model.
- The total error of this model is the sum of all errors of each point.

$$D = \sum_{i=1}^m d_i^2$$



- d_i : Distance between line and i^{th} point.
- m : Total number of points

Least Square Method

- Ordinary Least Mean Square to find the **coefficients**.

Slope

$$\theta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(h_\theta(x_i) - \bar{h}_\theta(x))}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Intercept

$$\theta_0 = \bar{h}_\theta(x) - \theta_1 \bar{x}$$

- \bar{x} : the mean value of input variable x .
- $\bar{h}_\theta(x)$: the mean value of output variable $h_\theta(x)$.

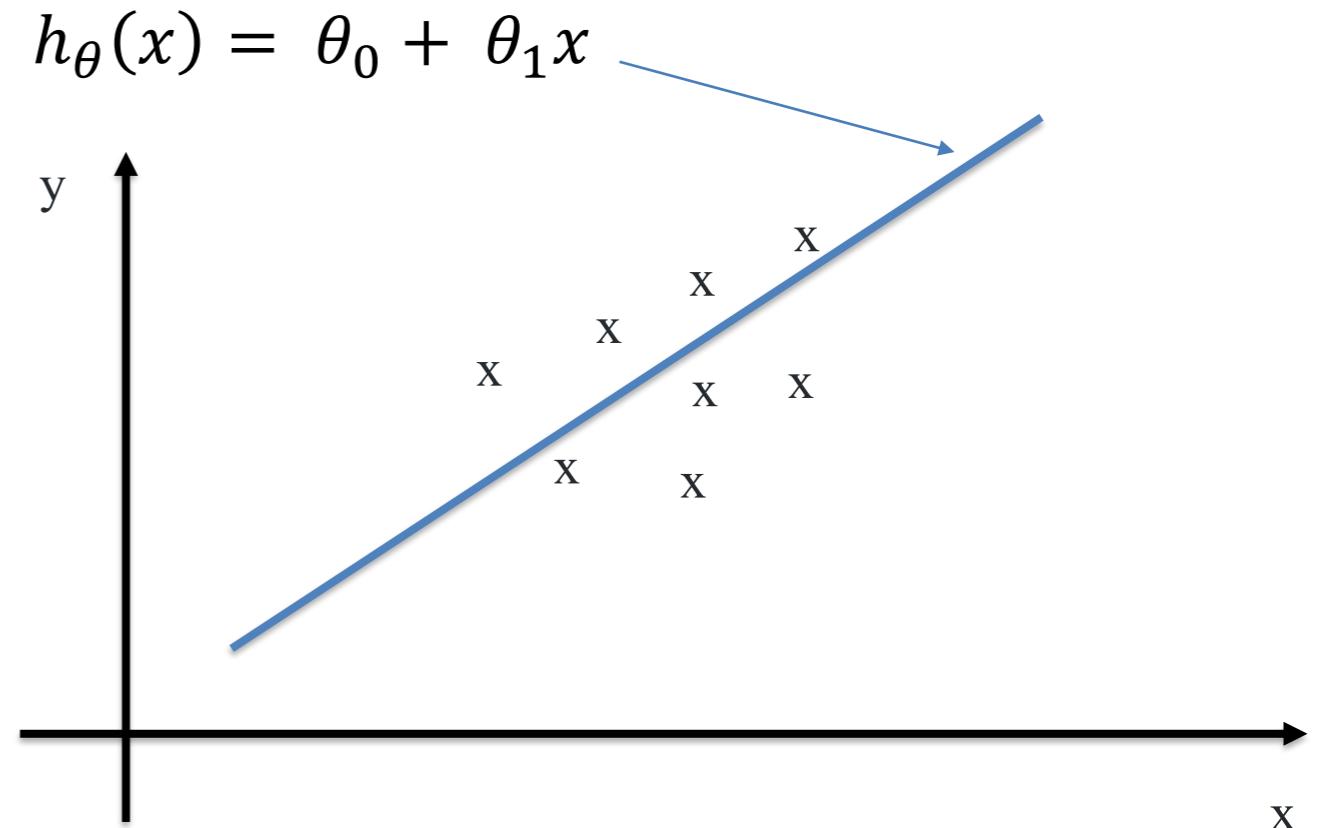
Problem representation

- **Hypothesis of a Univariate Linear Regression**

Hypothesis:

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

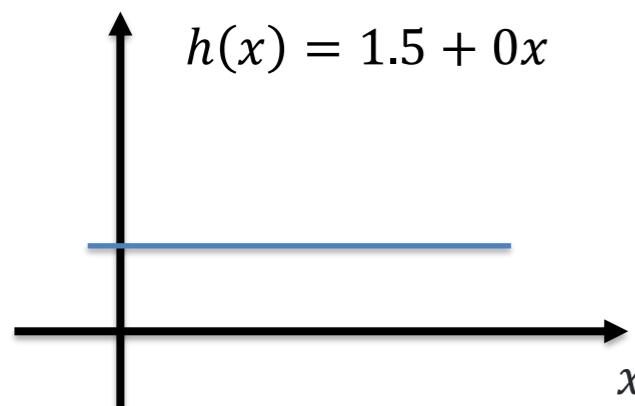
Parameters: θ_0 's



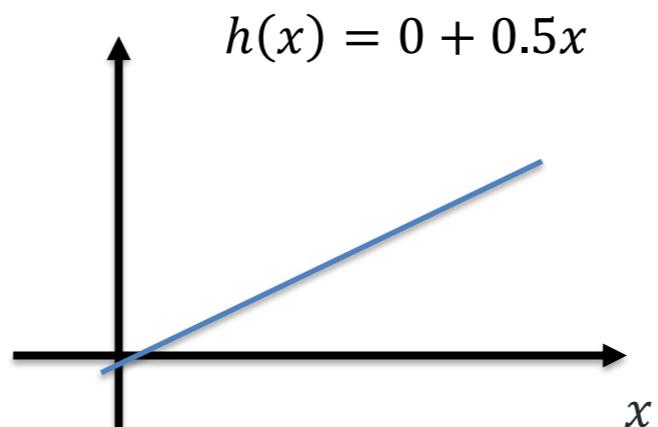
- **Objective:** find the **regression line**.

- Different values results in different regression lines.

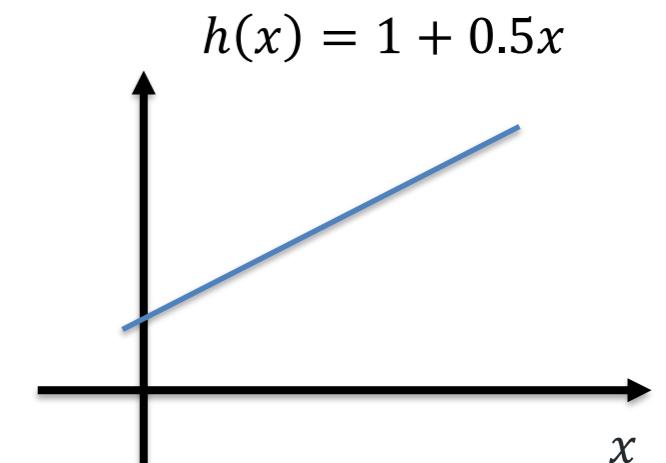
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



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



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



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

Cost Function

$$\min_{\theta_0 \theta_1} \frac{1}{2n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)})^2$$

Cost function: $J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)})^2$

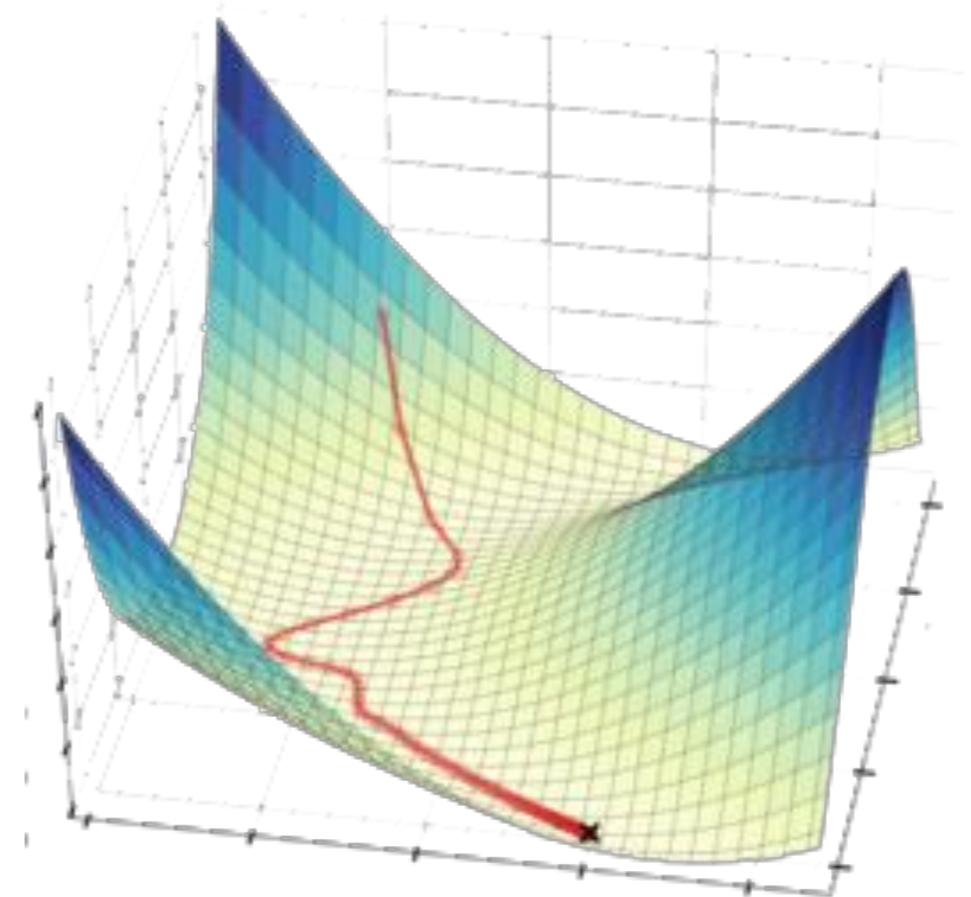
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

How: by minimizing the cost function

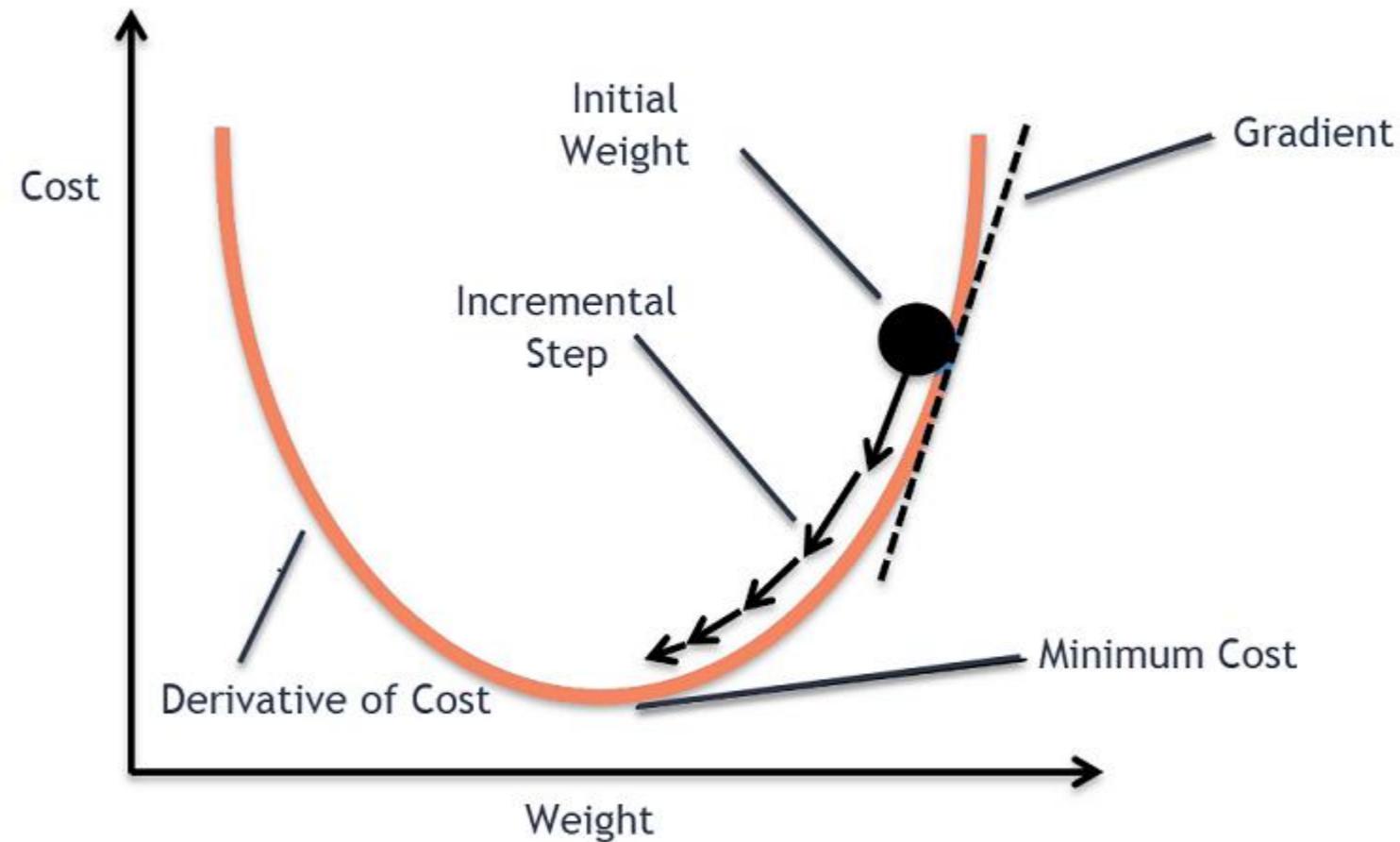
Solution: Gradient Descent

Gradient Descent

- An optimization algorithm
- Used to minimize the **cost function**.



Gradient Descent



Gradient Descent

Gradient descent algorithm

- Steps:
 - Compute the gradient, e.g., slope
 - Make a step (move) in the direction of opposite to the gradient

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

GD Algorithm

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)})^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n (\theta_0 + \theta_1 x^{(i)} - h_\theta(x)^{(i)})^2$$

- For θ_0 case and θ_1 case:

$$\text{For } \theta_0 \quad j = 0: \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)})$$

$$\text{For } \theta_1 \quad j = 1: \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)}).x^{(i)}$$

Gradient Descent

Gradient descent algorithm

Repeat until **convergence** {

$$\theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^n (\hat{h}_\theta(x^{(i)}) - h_\theta(x)^{(i)}). x^{(i)}$$

}

Gradient Descent

Gradient descent algorithm

- Steps:
 - Compute the gradient, e.g. slope
 - Make a step (move) in the direction of opposite to the gradient

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

simultaneous update θ_0, θ_1

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

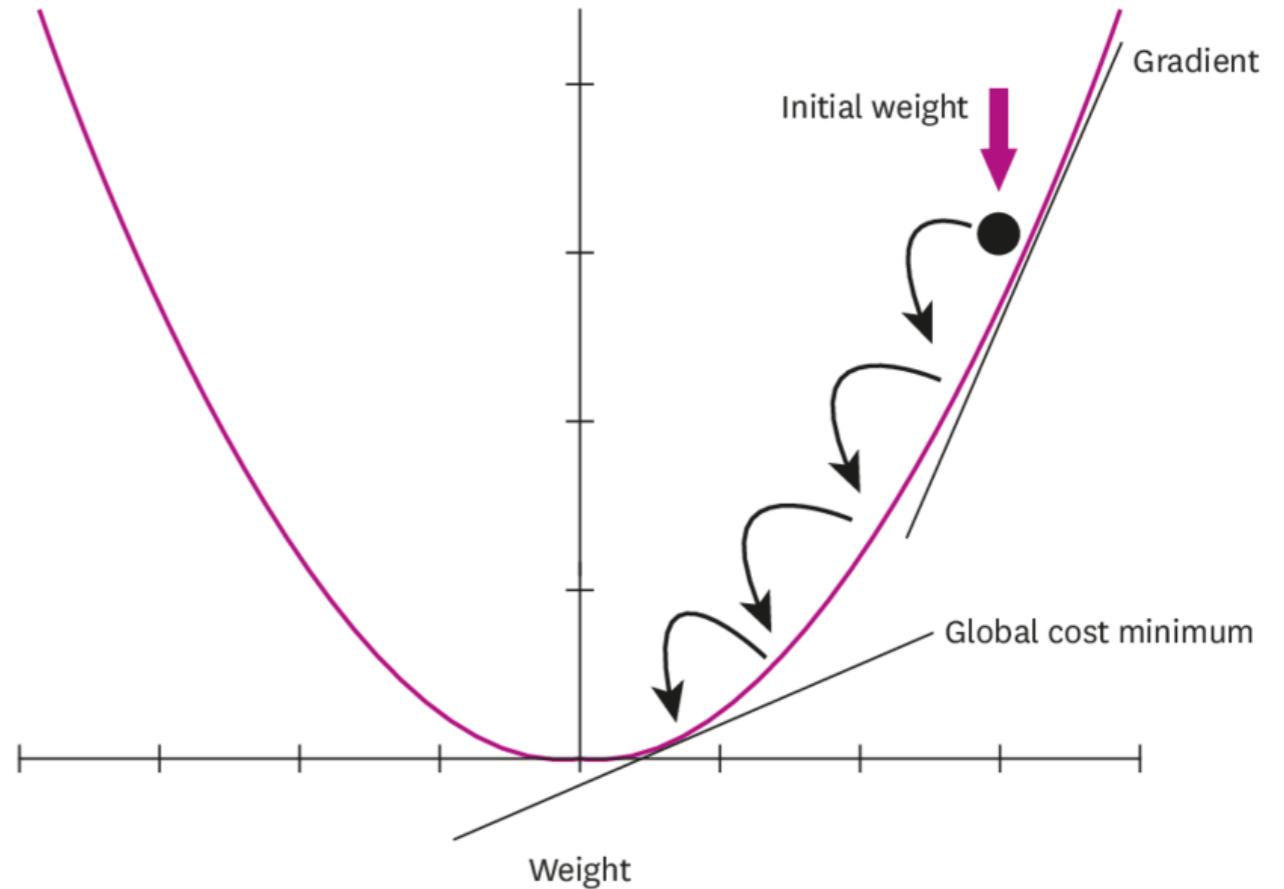
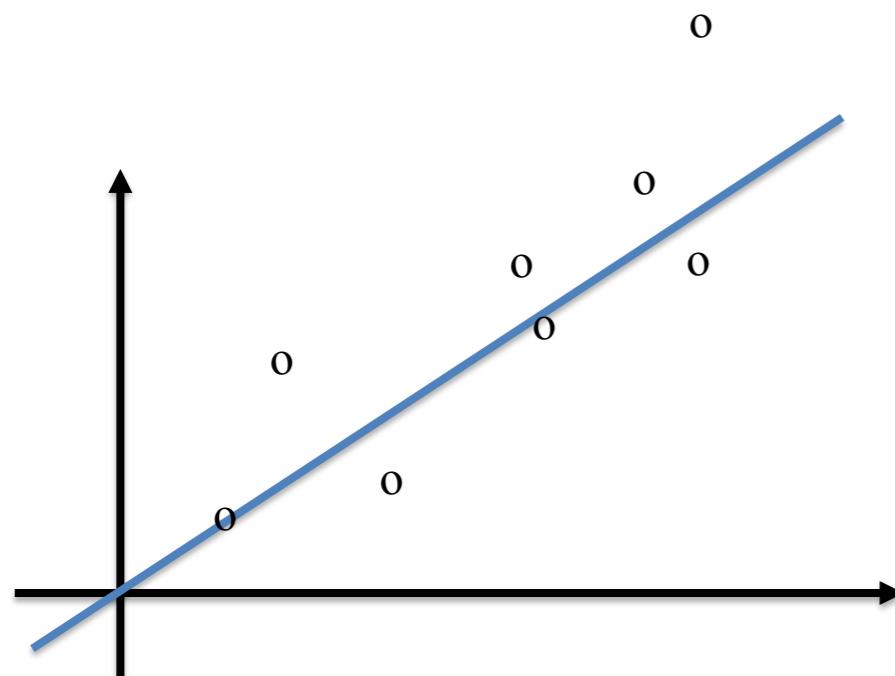
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

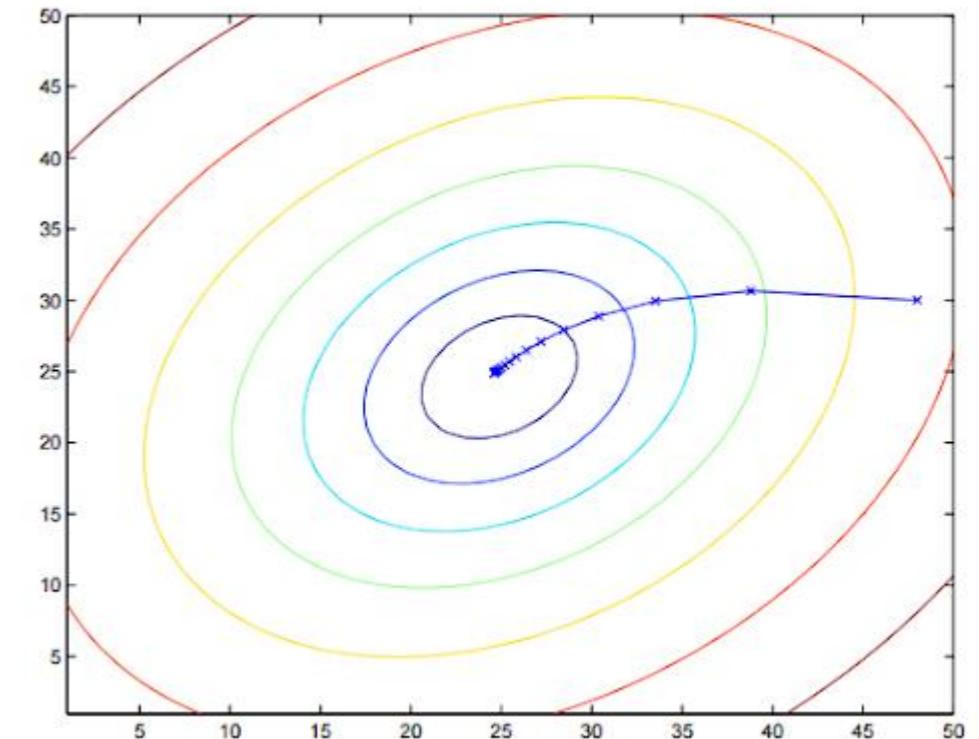
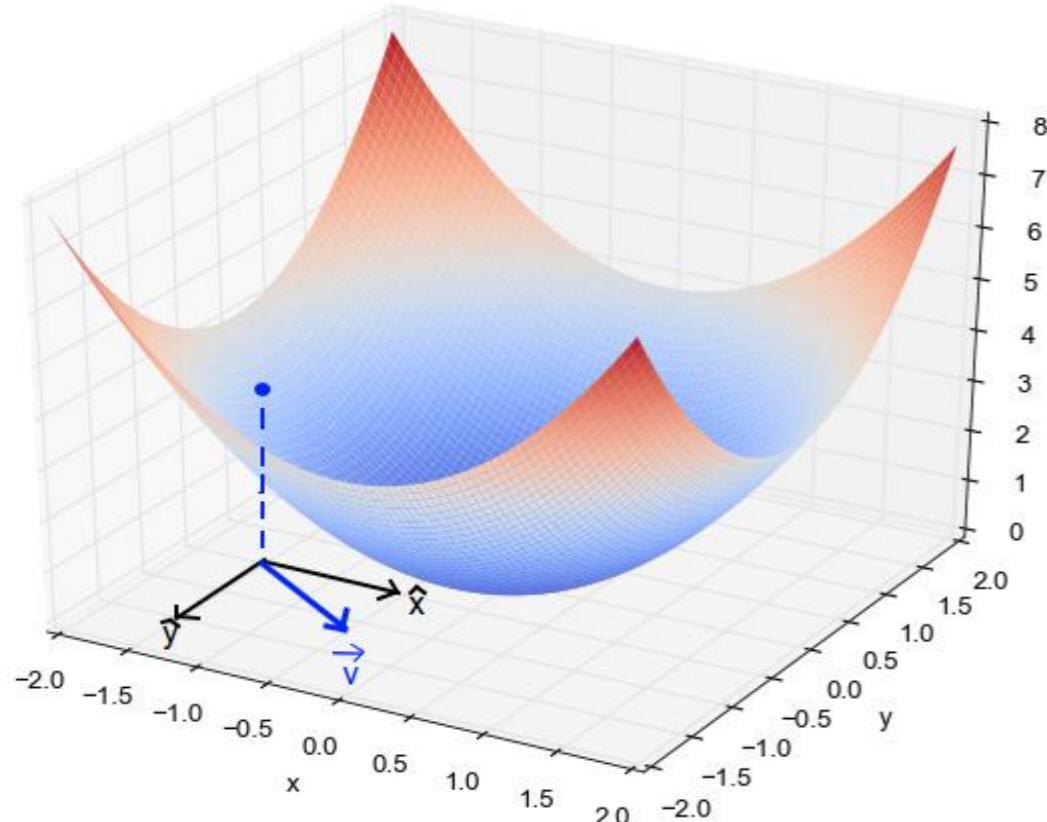
Gradient Descent

- One parameter:



Gradient Descent

- Two parameters:

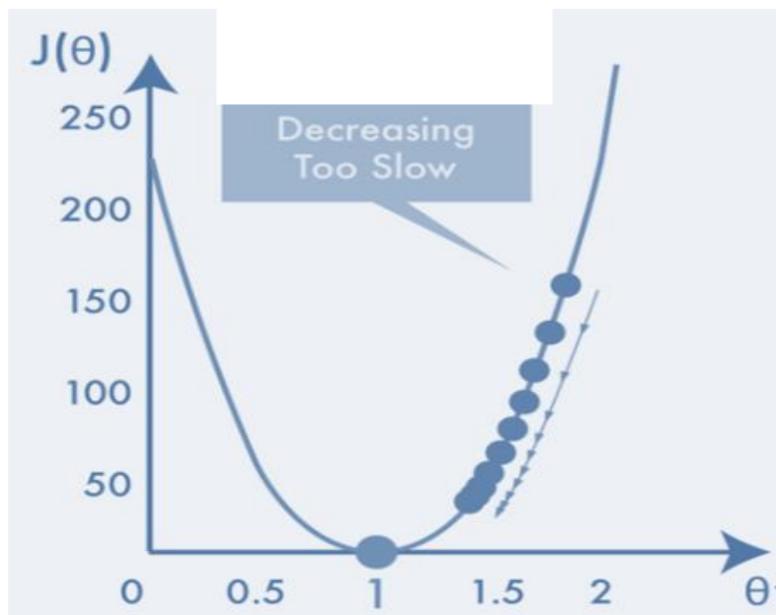


Python: `matplotlib.pyplot.contour()`

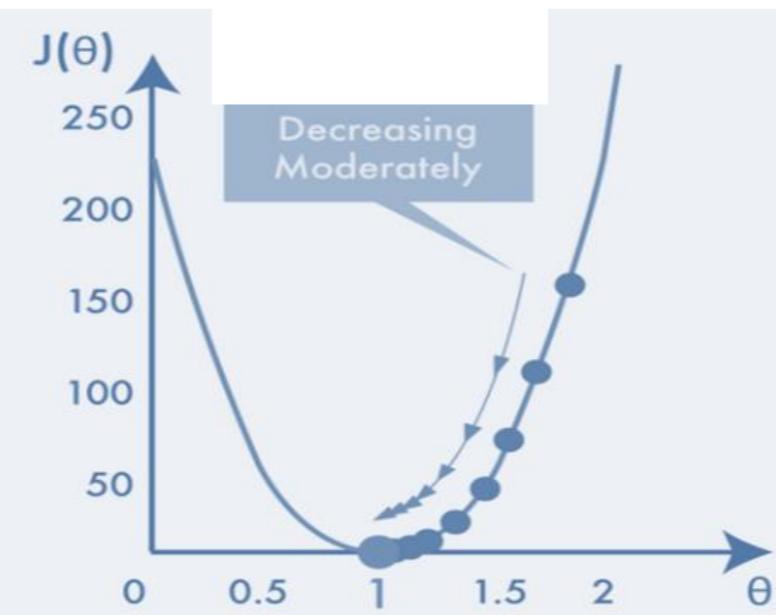
Learning Rate

- The size of step to take towards a local minima.
 - Large learning rate: **overshooting**
 - Small learning rate: **too long training process**

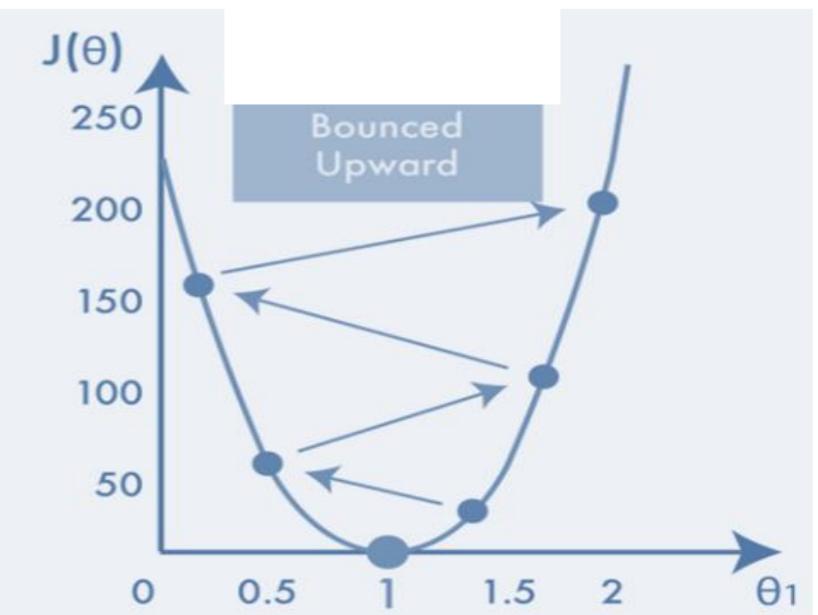
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



small α

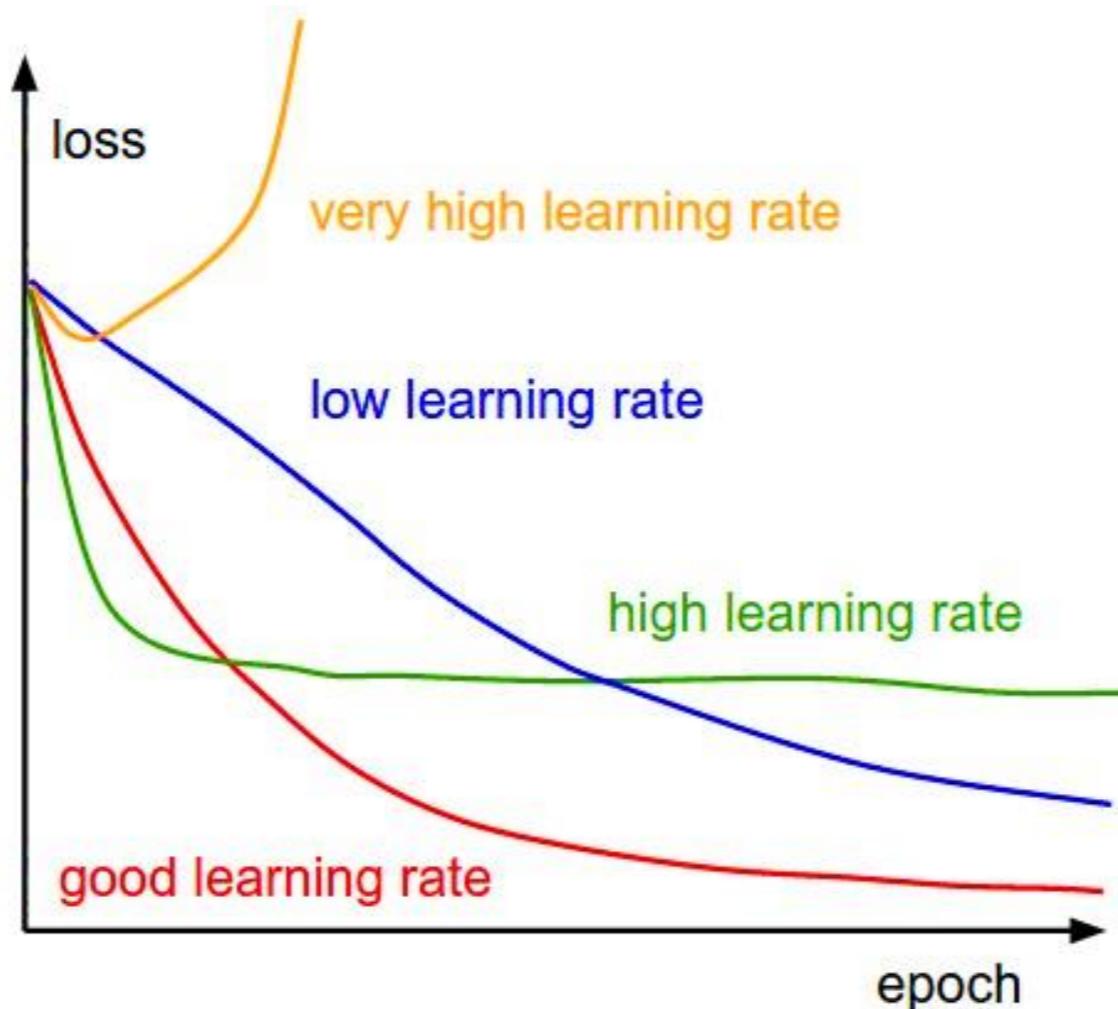


moderate α



large α

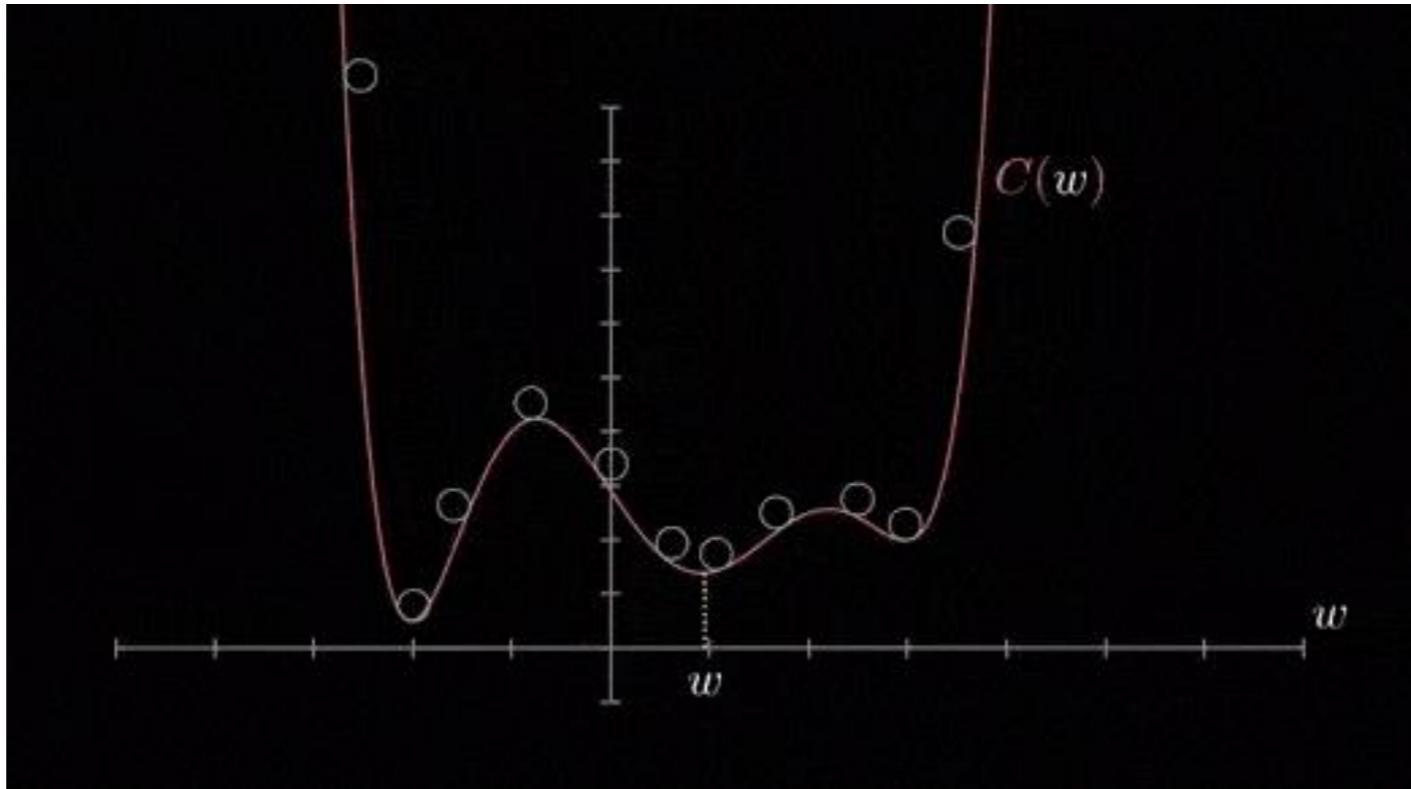
Learning Rate



- Note: as the gradient decreases while moving towards the local minima, the size of step decreases.

Global Minima

- GD does not guarantee the global minima.





Office: #432, Dayang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



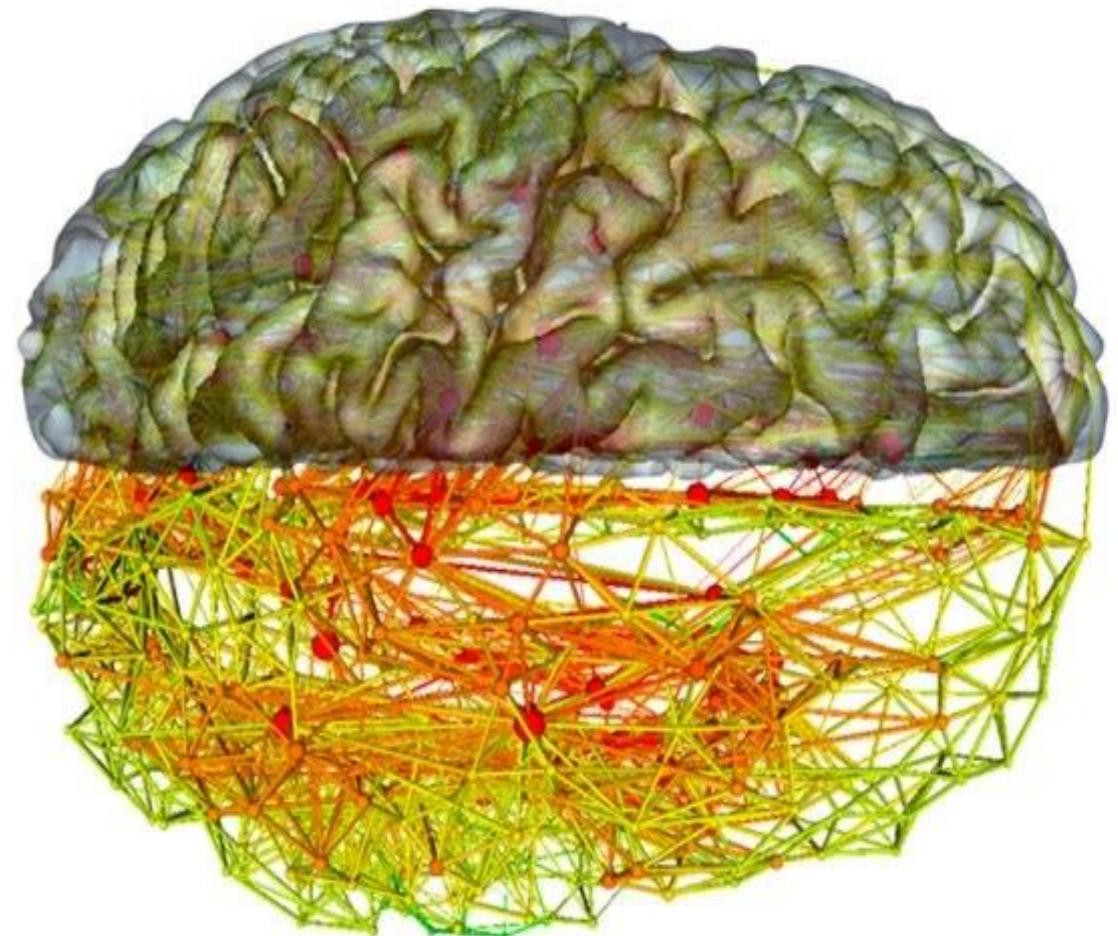


Course Outline

- Introduction to AI
- The History of AI
- Python
- Search
- Informed Search
- Beyond Classical Search
- Adversarial Search
- Constraint Satisfaction Problems
- Fuzzy Logic
- Introduction to Machine Learning
- Types of Machine Learning
- SL: Linear Regression
- **SL: Artificial Neural Networks**

ARTIFICIAL NEURAL NETWORKS

- Introduction
- Activation Function
- Learning in ANN
- History
- Applications
- Forward Propagation Algorithm
- Backpropagation Algorithm
- Hyperparameters
- Optimization

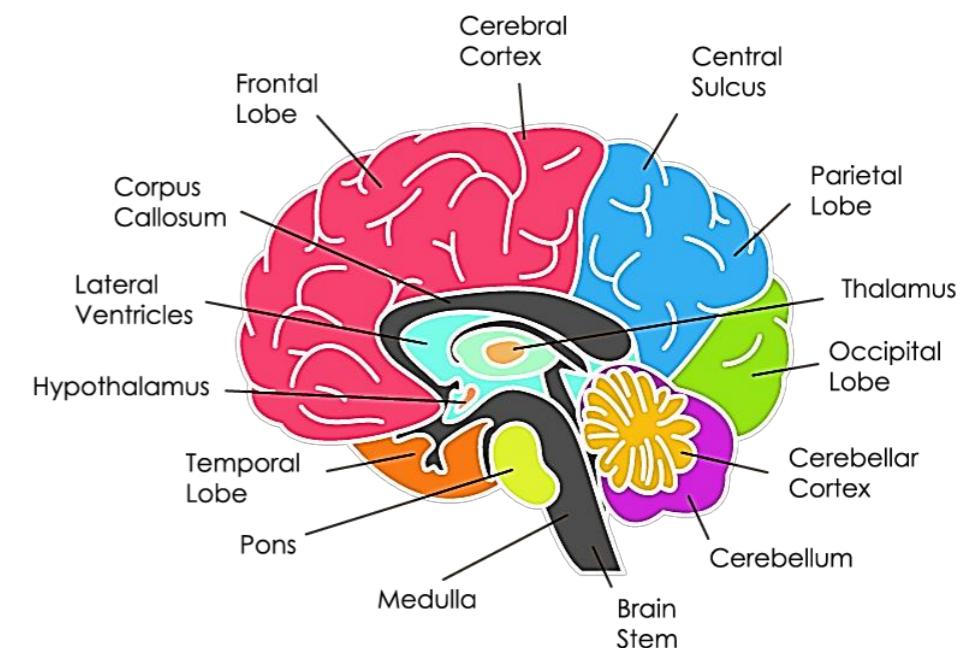


Neural Networks

- Neural Networks:
 - **Biological models**
 - **Artificial models**
- **Goal:**
 - To produce artificial systems capable of sophisticated **computations** similar to the **human brain**.

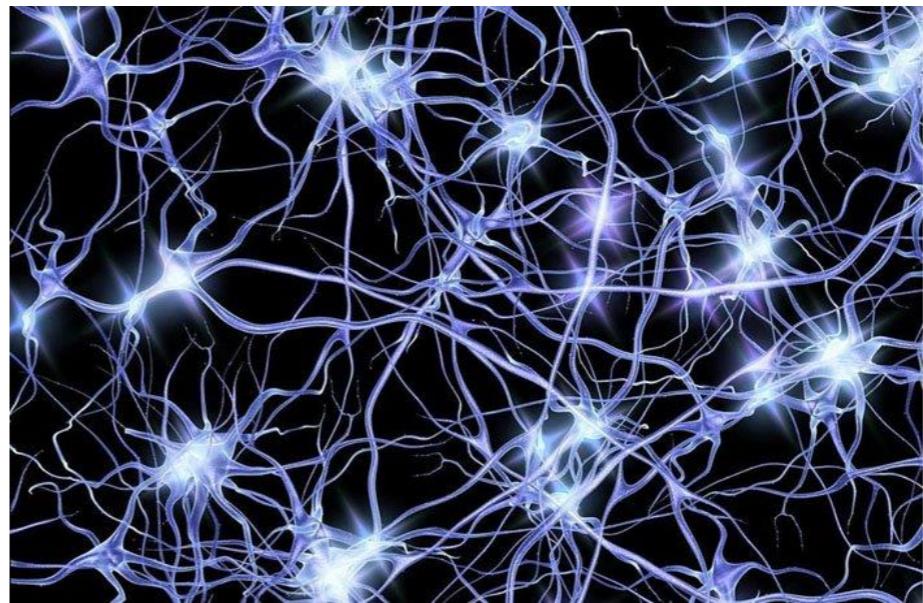
The Brain

- Human brain “**computes**” in an entirely different way from conventional digital computers.
- The brain is highly **complex, nonlinear, and parallel**.
- Key features of the biological brain:
 - Experience shapes the wiring through plasticity,



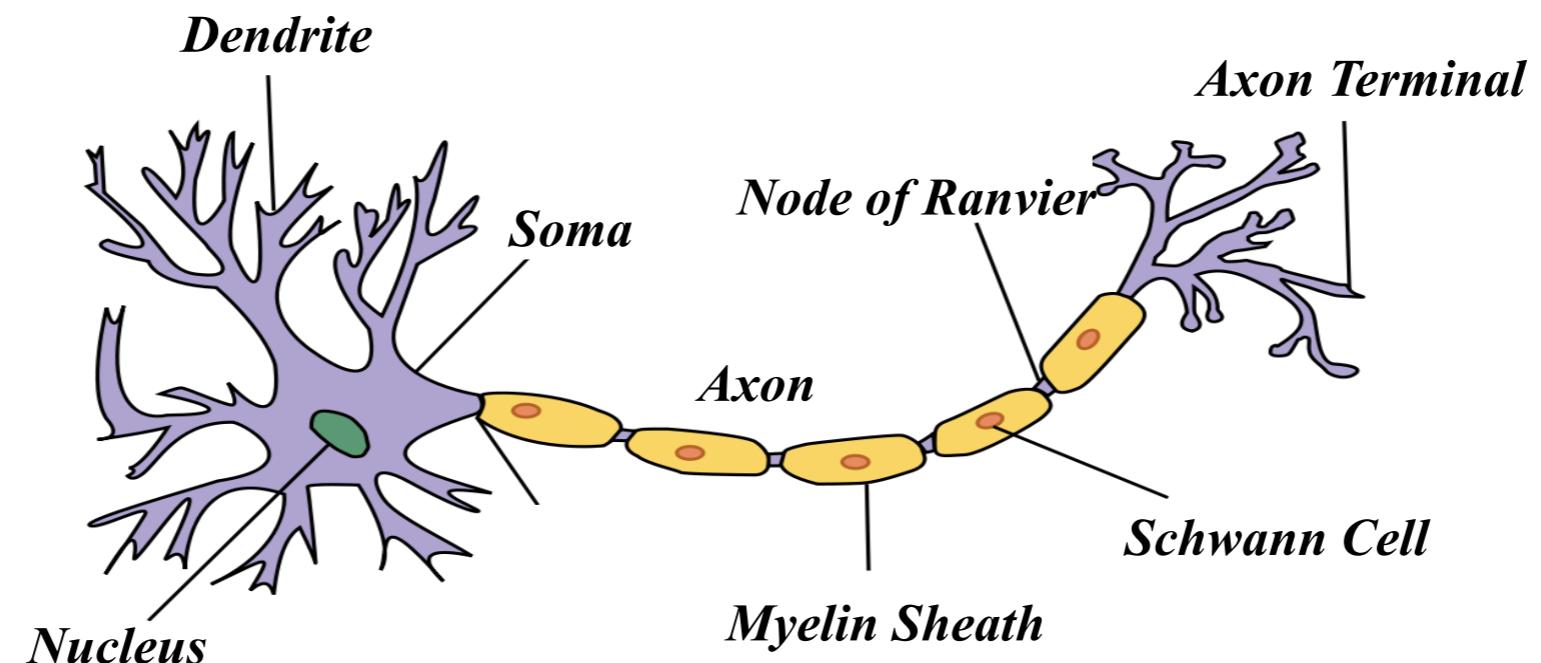
Human Neurons

- Number of **neurons**
 - $\sim 10^{10}$
- Neuron **switching time**
 - ~ 0.001 second
- **Connections** per neuron
 - $\sim 10^{4-5}$
- **Scene recognition** time
 - ~ 0.1 second
- 100 inference steps doesn't seem like enough
 - much parallel computation



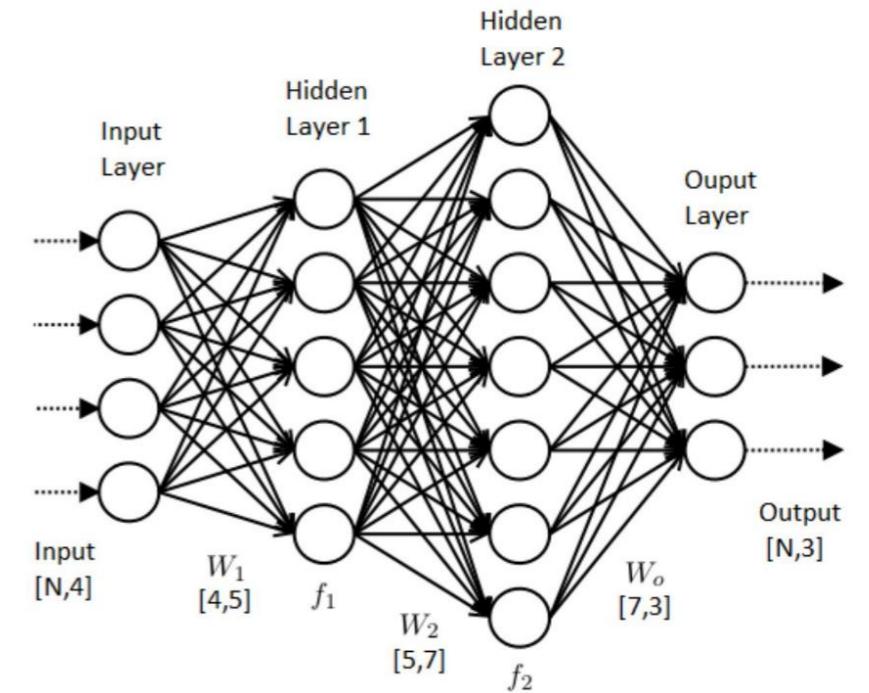
Human Neurons

- A biological neuron basically:
 - Receives inputs from other sources (**dendrites**),
 - Merges them in some way (**soma**),
 - Performs an operation on the result (**axon**),
 - Outputs the result - possibly to other neurons (**axon terminals**).



- Massively **parallel distributed processor**
- Made up of simple **processing units**.
- **Unit:** has a natural propensity for storing experimental knowledge and making it available for use.
- Neural networks **resemble the brain:**
 - **Knowledge:** acquired from the environment through a **learning process**.
 - Inner neuron connection **strengths**, e.g., **weights**, to store the acquired knowledge.
- Learning by adjusting:
 - **Learning algorithm, Weights, topology, ...**

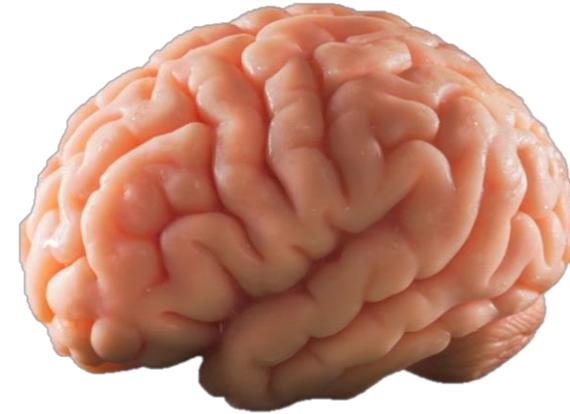
- “**Learning**” to perform tasks by considering **examples**.
- Based on a collection of connected **nodes Artificial Neurons**.
- Each connection can transmit a signal from one artificial neuron to another.
- Neuron receives a signal and can **process** it and then **pass** it.



Definition

- “An **information processing system** that has been developed as a generalization of mathematical models of human cognition or neurobiology, based on the assumptions that:
 - Information processing occurs at many simple elements called **neurons**.
 - Signals are passed between neurons over connection **links/channels**.
 - Each connection link has an associated **weight**, which typically multiplies the signal transmitted.
 - Each neuron applies an **activation function** (usually non-linear) to its net input (sum of weighted input signals) to **determine** its output signal.”

Biological Neural Network



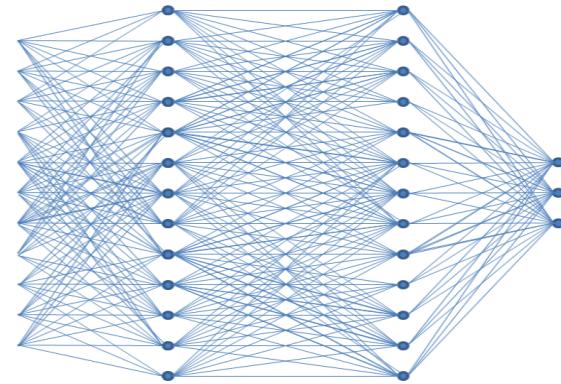
Dendrites

Soma

Axon

Axon terminals

Artificial Neural Network



Weights,

Input function
(summation function)

Activation function
(transfer function)

Output

- The procedure:
 - **Input values** enter the neuron via the **connections**.
 - The **inputs** are multiplied with the **weighting factor** of their respective connection.
 - The modified inputs are fed into a **summation function**.
 - The result from the summation function is sent to a **transfer function**.
 - The neuron **outputs** the result of the transfer function into other neurons.



Epoch
000,141

Learning rate
0.03

Activation
Sigmoid

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?



X_1 8 neurons



X_2 6 neurons



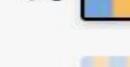
X_1^2 4 neurons



X_2^2 2 neurons



X_1X_2

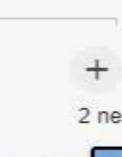
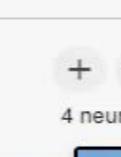
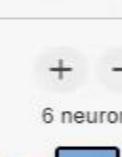


$\sin(X_1)$



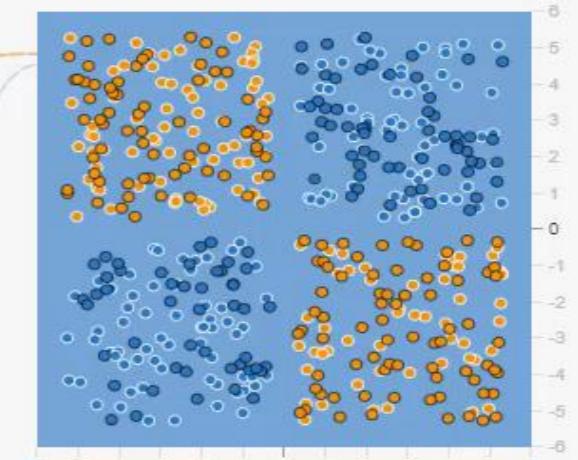
$\sin(X_2)$

4 HIDDEN LAYERS

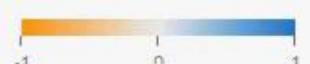


OUTPUT

Test loss 0.501
Training loss 0.499



Colors shows data, neuron and weight values.

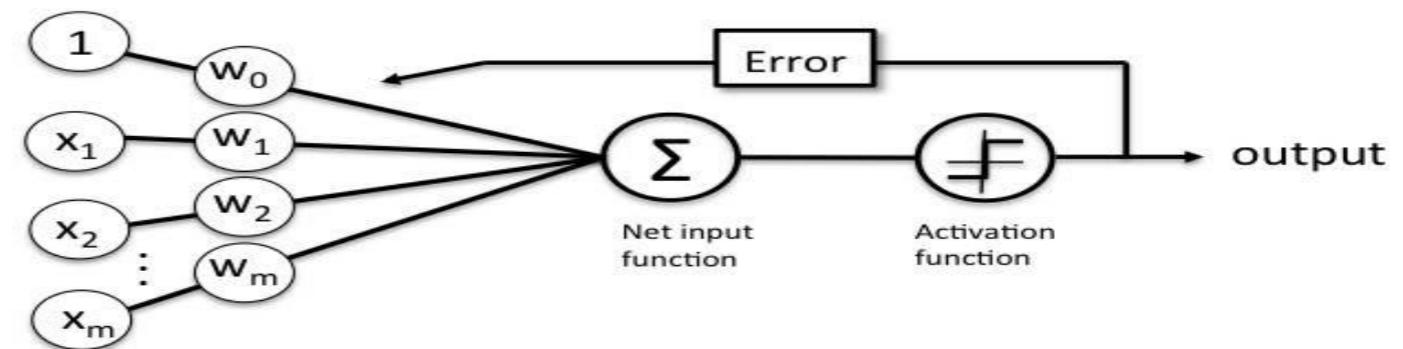


Show test data

Discretize output

This is the output from one neuron.
Hover to see it larger.

- Types
 - Single-layered NN
 - e.g., perceptron.

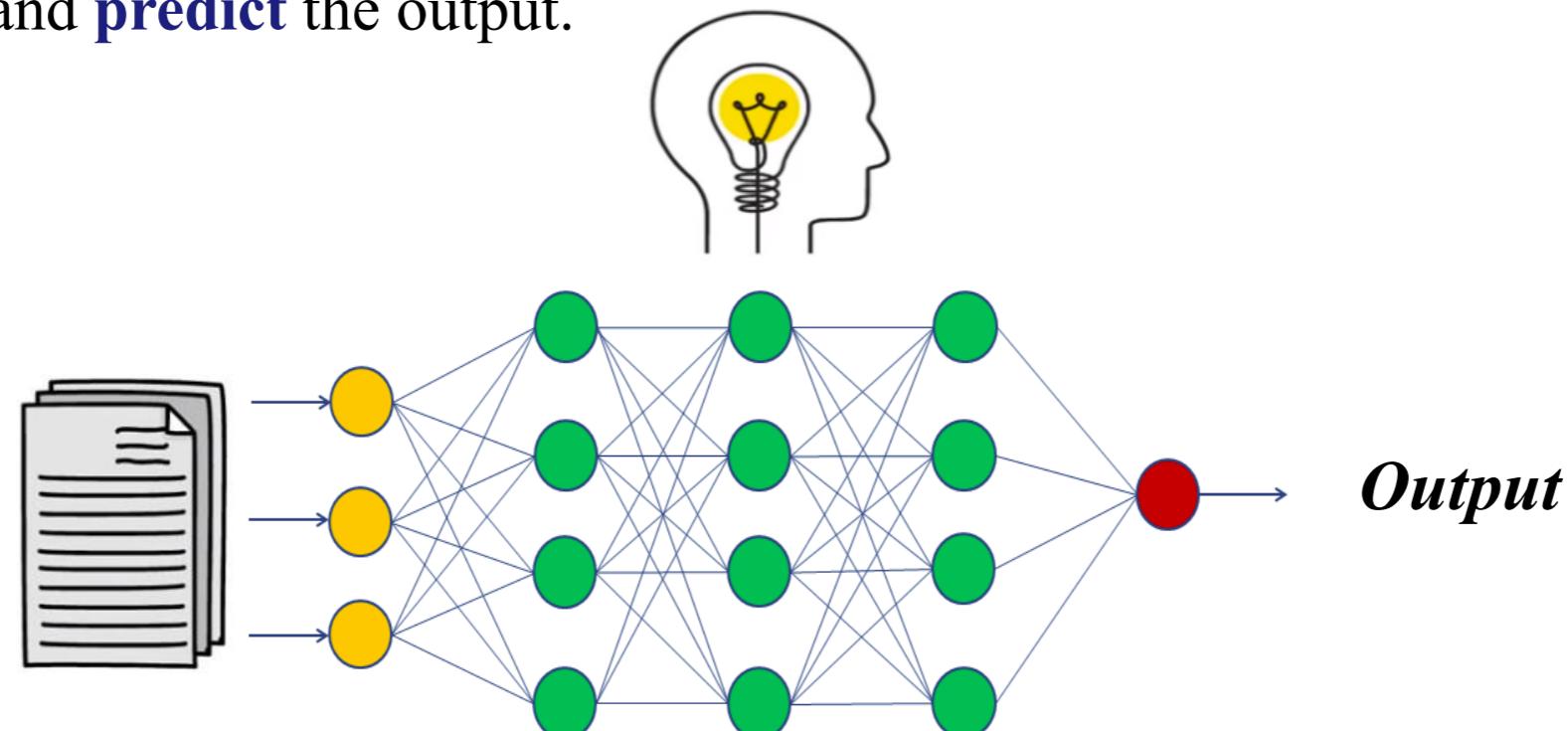


- Multi-layered NN

In nutshell,

ANN:

- takes in **data**,
- **train** itself to recognize the **patterns** in the data
- and **predict** the output.

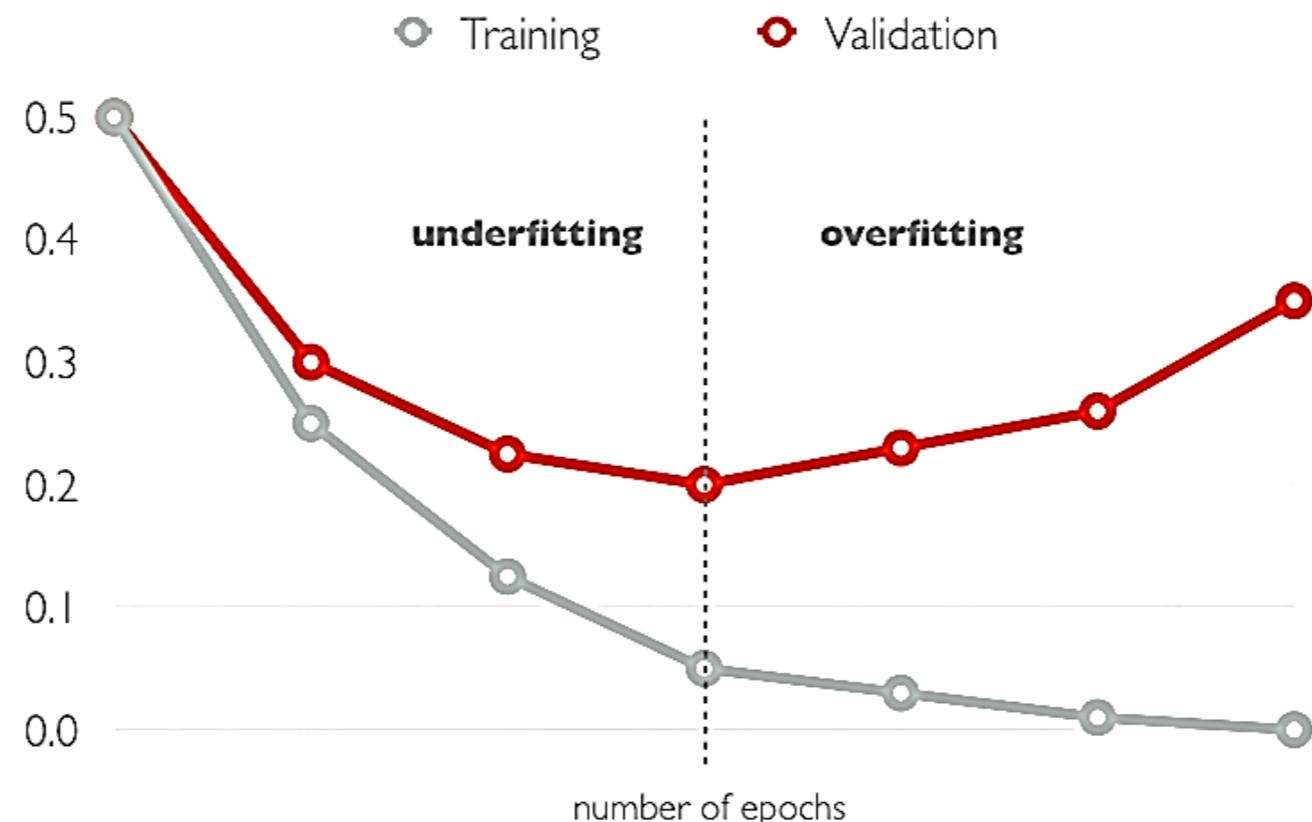


- **Batch:**
 - Instead of passing the entire dataset into the ANN at once, divide the dataset into number of batches.
- **Iteration:**
 - Running the procedure over each batch.
 - For example, if we have 10,000 images as data and a batch size of 200, then an epoch should contain $10,000/200 = 50$ iterations.
- **Epoch:** one iteration over entire dataset.

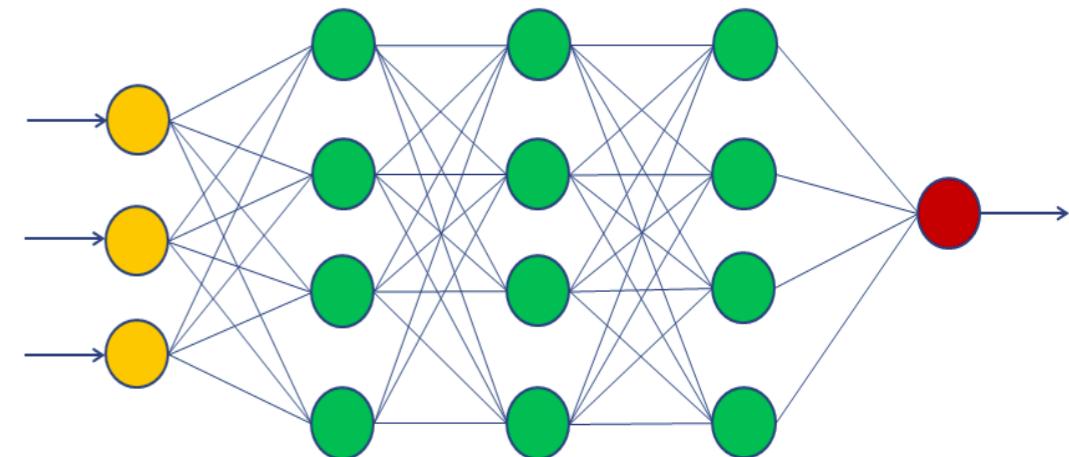
Putting it together

- Validation set:

- The best configuration selection!

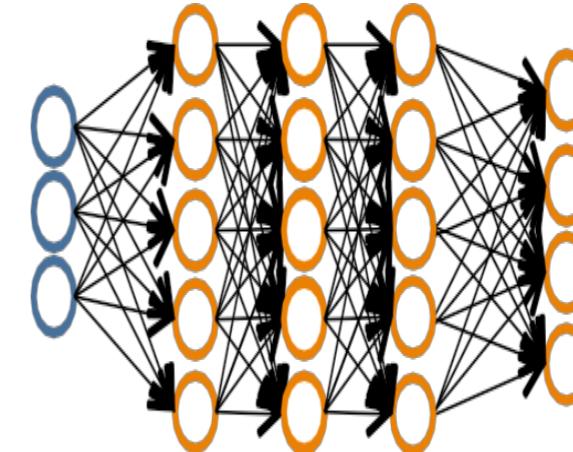
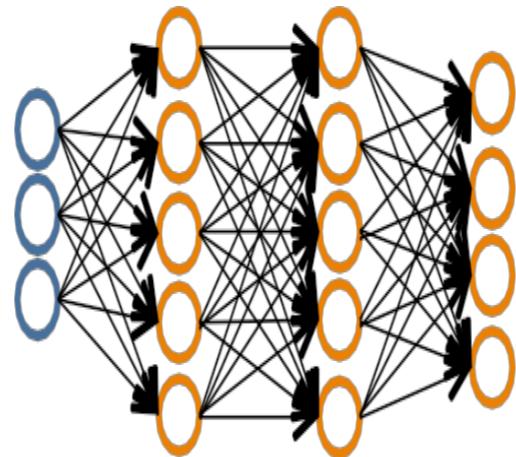
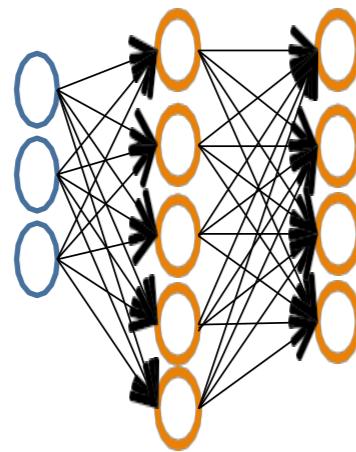


- **Input Layer**
- **Hidden Layer(s)**
- **Activation Function:** defines the output of that node given an input or set of inputs.
- **Weights**, adjusts as learning proceeds.
- **Biases**; as a **threshold** to shift the activation function.
- **Output Layer** represents the results.



Architecture

- Pick a network **architecture**



- Number of **input units**: Dimension of features
- Number **output units**: Number of classes
- Reasonable default: **1 hidden layer**,

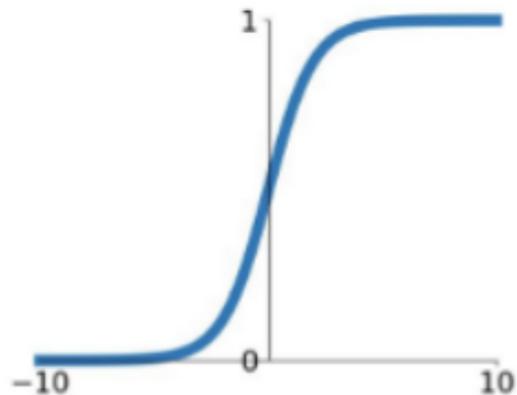
Activation Function

- Determines whether a particular node be **activated** or not.
- Only the activated neuron **transmits** data to the next layer.
- e.g.
 - Sigmoid Function
 - Rectified Linear Unit (ReLU)
 - Leaky Rectified Linear Unit
 - Hyperbolic Tangent (tanh)
 - Exponential Linear Units (ELU)
 - ...

Activation Function

Sigmoid Function (σ)

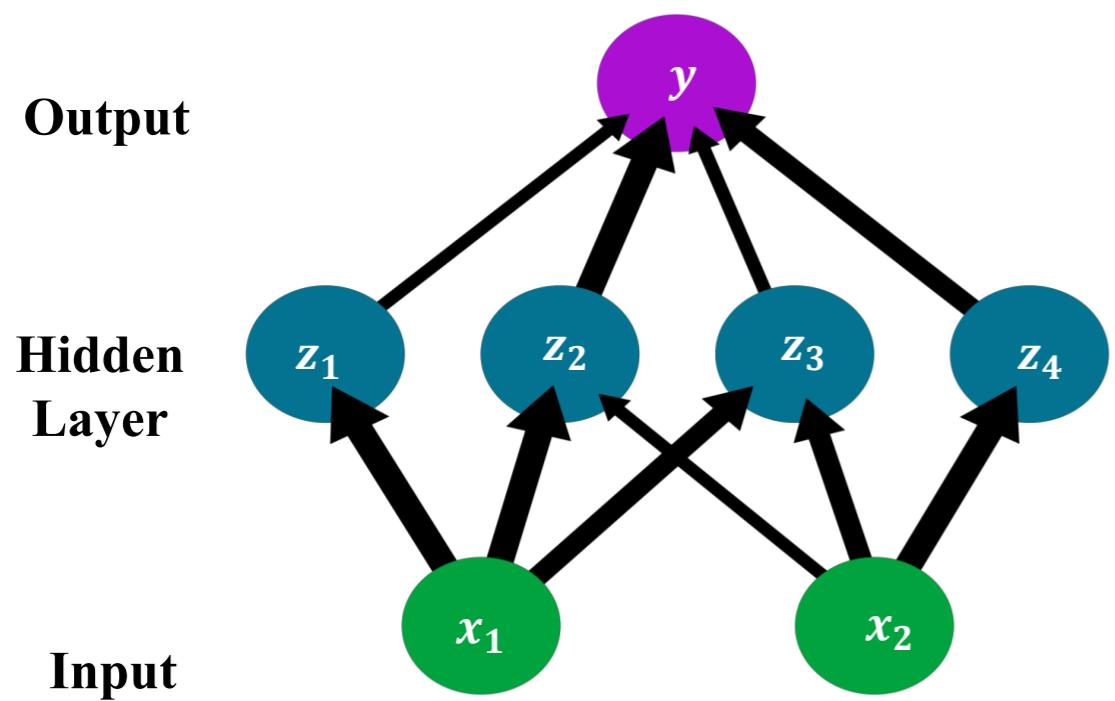
$$g(z) = \frac{1}{(1+e^{-z})}$$



- Range from [0,1].
- Not zero centered.
- Have exponential operations.
- Mostly for output layer where the output is binary.
- For the hidden layers, it is slow.
- Simplification during backpropagation.

Activation Function

- Neural Network with sigmoid activation functions



Loss

$$J = \frac{1}{2}(y - y^*)^2$$

Output (Sigmoid)

$$y = \frac{1}{1 + \exp(-b)}$$

Output (Linear)

$$b = \sum_{j=0}^D \beta_j z_j$$

Hidden (Sigmoid)

$$z_j = \frac{1}{1 + \exp(-a_j)}, \forall j$$

Hidden (Linear)

$$a_j = \sum_{i=0}^M \alpha_{ji} x_i, \forall i$$

Input
given $x_i, \forall i$

- Inputs are flexible
 - Any real values
 - Highly correlated or independent
- Target function may be discrete-valued, real-valued, or vectors of discrete or real values
 - Outputs are real numbers between 0 and 1
- Resistant to errors in the training data
- Long training time
- Fast evaluation
- The function produced can be difficult for humans to interpret

Learning in ANN

- **Learning:** the method of **modifying** the weights of connections between the neurons.
- Goal: to find weights and biases that **minimize** the **cost function**.
- Types of Learning:
 - **Supervised Learning;**
 - The input vector is presented to the network, which will give an output vector.
 - This output vector is compared with the desired output vector.
 - An error signal is generated, if there is a difference between the actual output and the desired output vector.
 - On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

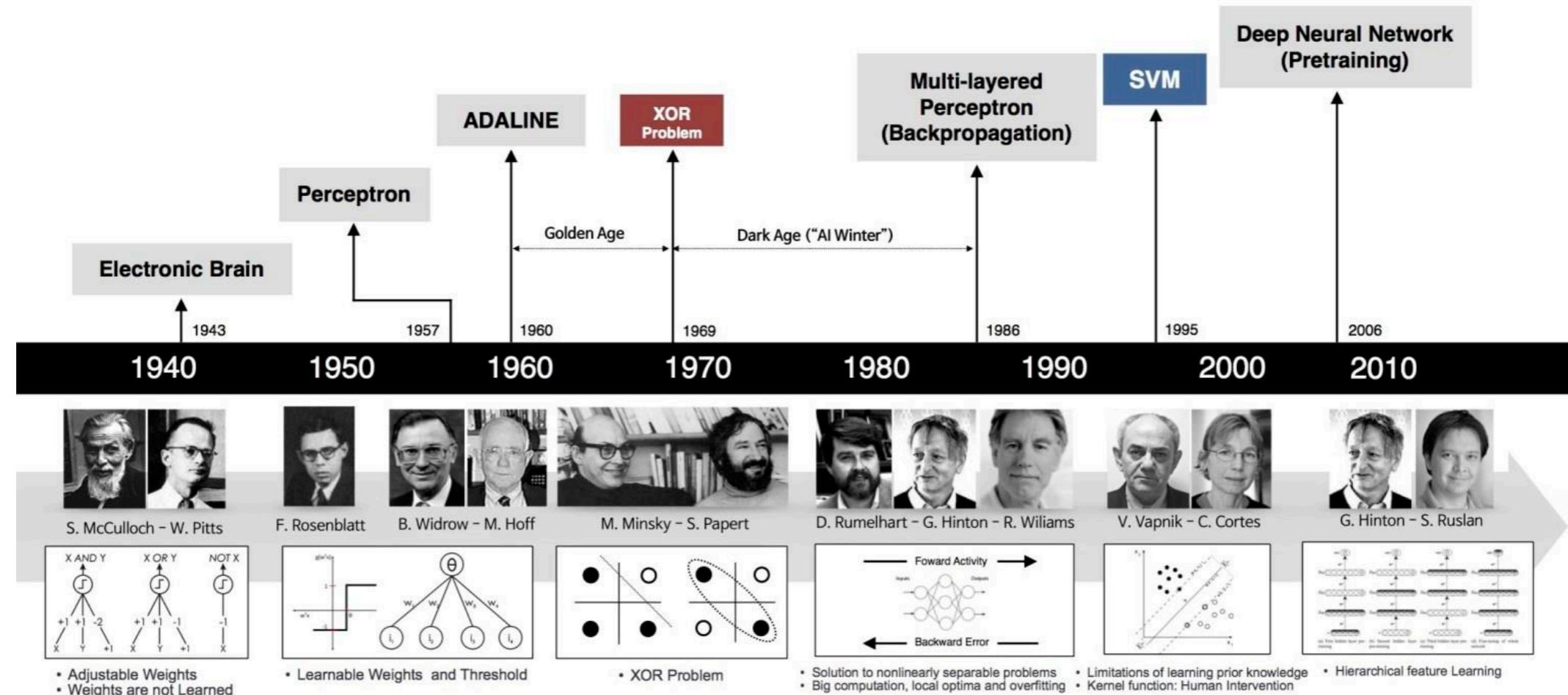
Learning in ANN

- **Unsupervised Learning;**
 - Forms clusters based on the input vectors of similar type.
 - Upon arrival of a new input pattern is applied, the neural network gives an output response indicating the class to which the input pattern belongs.
 - No feedback from the environment as to what should be the desired output and if it is correct or incorrect.
 - The network itself must discover the patterns and features from the input data, and the relation for the input data over the output.

Learning in ANN

- **Reinforcement Learning;**
 - The network receives some feedback from the environment.
 - This makes it somewhat similar to supervised learning.
 - However, the feedback obtained here is evaluative not instructive,
 - After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.

History

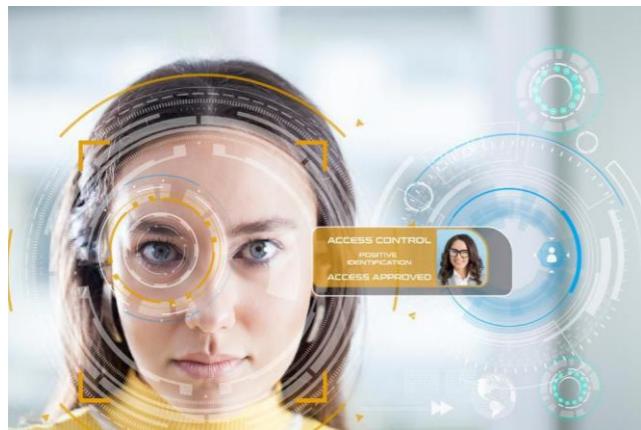


Applications



- Application areas
 - Language processing
 - Character recognition
 - Pattern recognition
 - Signal processing
 - Prediction
 - ...

Applications



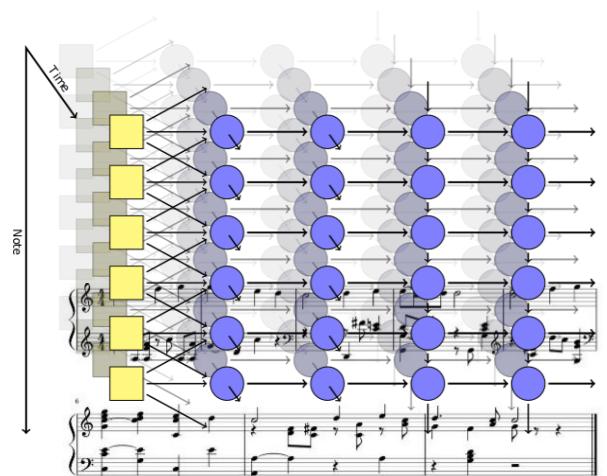
Facial Recognition



Real-time Translation



Image Compression



Music Composition

Winter is here. Go to
the store and buy some
snow shovels.

Winter is here. Go to the store and buy
some snow shovels.

Handwriting Recognition

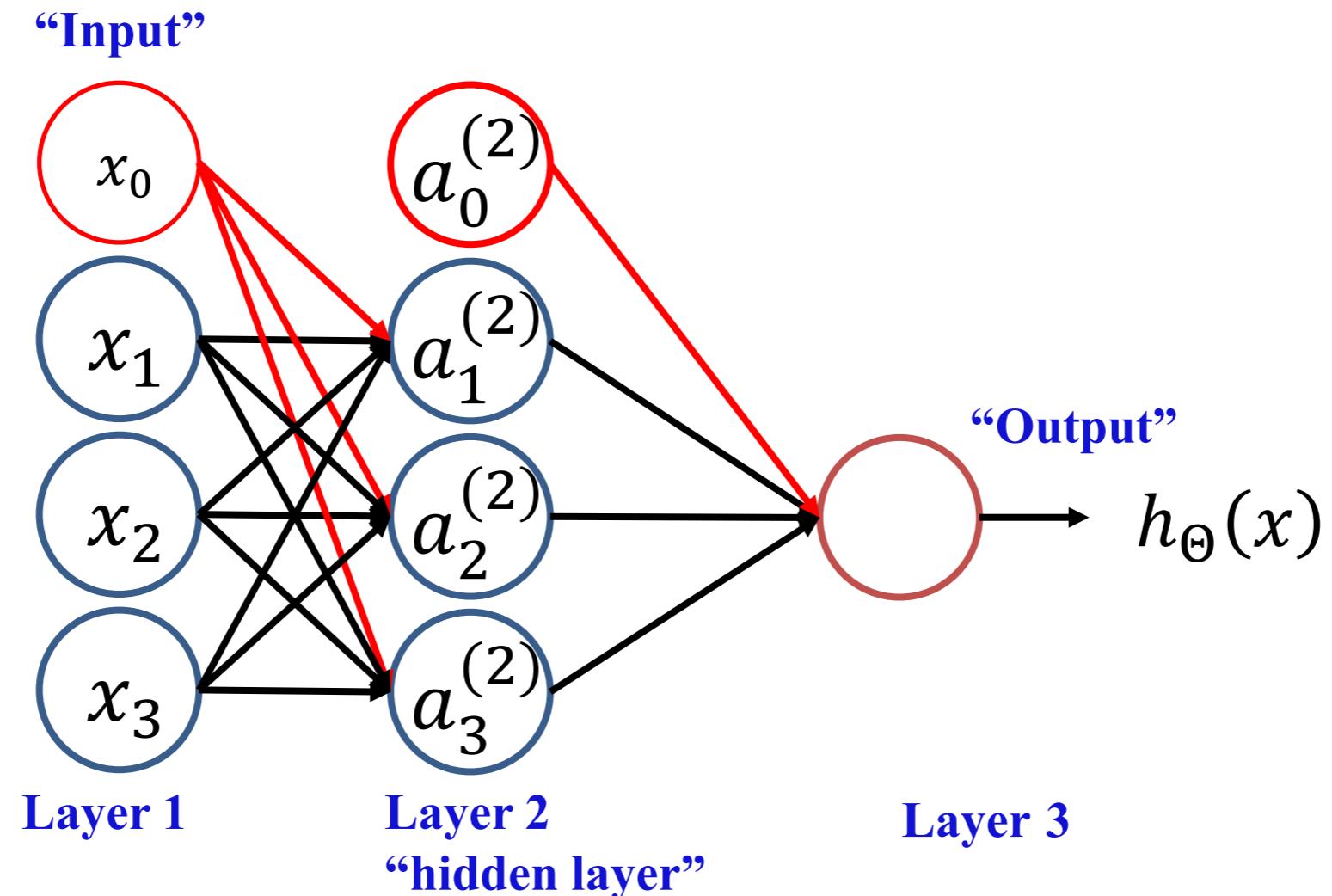


Stock Prediction

Advantages of ANN

- **Nonlinearity**
- **Input-output mapping**
- **Additivity**
- **Evidential response**
- **Contextual information**
- **Fault tolerance**
- **VLSI implementability**
- **Uniformity of analysis and design**
- **Neurobiological analogy**

- An ANN composed of artificial neurons.



A network with only a single hidden layer is conventionally called "**shallow**".

Forward Propagation Algorithm

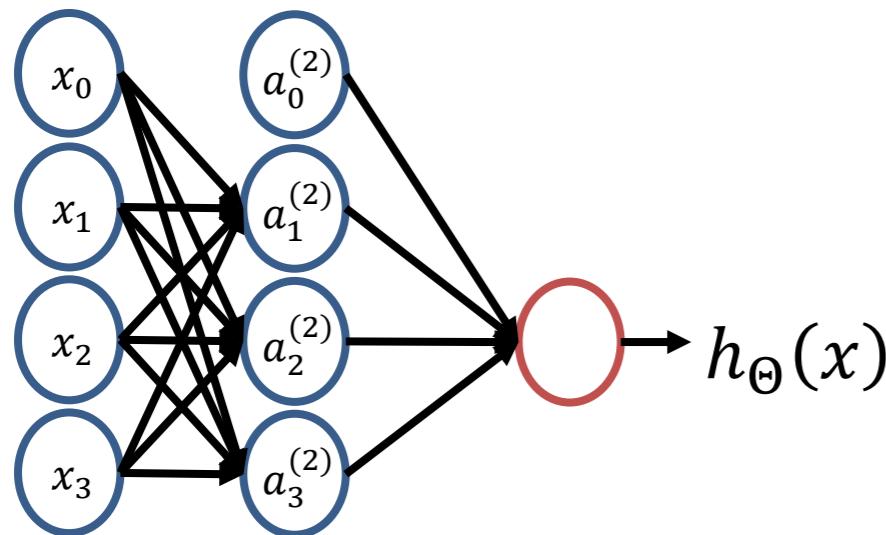
- In ANN, the neurons of one layer are connected to the next layer through **channels**.
- Each of the channels is assigned a numerical values, i.e., **weights**.
- The input is **multiplied** by the correspond weight.
- Their **sum** is sent as the input to the next **neuron** in the **hidden layer**.
- Each **neuron** is associated with a numerical value, i.e., the **bias**, which is added to the input sum.
- The result will be passed to the **Activation function**.
- The result of activation function determines if the neuron will be **activated or not**.
- An activated neuron **transmits data** to the neurons of the **next layer** over the **channels**.
- In the **output layer**, the neuron with the **highest** value determines the **output**.

Backpropagation Algorithm

- If through the forward propagation, the network **failed** to **predict** correctly, the network need to be **trained**.
- The **predicted output** is compared with the **actual output** to realize the error e.g., the probability.
- The magnitude of **error** indicates how wrong the networks is.
- This information is then transferred **backward** through the network.
- Based on the received information, the **weights** are **adjusted**.
- The process of “**forward propagation**” and “**backpropagation**” is iteratively performed till the network can predict the output correctly in most of the cases.
- Q) how long it may take time! Seconds, minutes, hours, months, ...

	Actual Output	Error
↓	0	-0.5
↑	1	+0.6
↓	0	-0.1

Forward Propagation Representation



$a_i^{(j)}$ = “activation” of unit i in layer j

$\Theta^{(j)}$ = matrix of **weights** controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right)$$

$$h_{\Theta}(x) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right)$$

Forward Propagation Representation

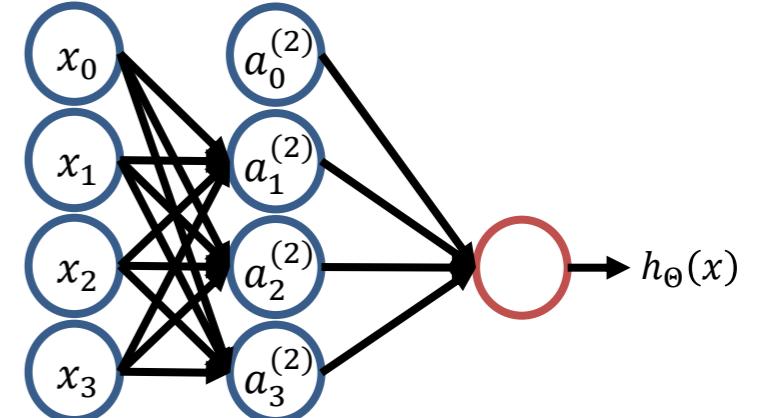
Vectorized Implementation

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g(z_1^{(2)})$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g(z_2^{(2)})$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g(z_3^{(2)})$$

$$h_{\theta}(x) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g(z^{(3)})$$

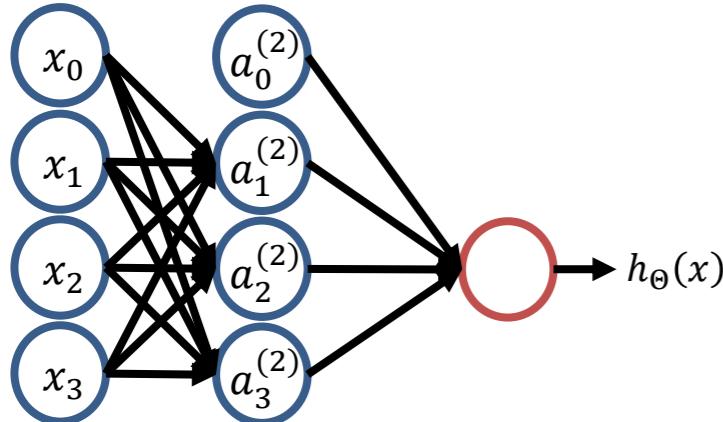


“Pre-activation”

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

Forward Propagation Representation



“Pre-activation”

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$a_1^{(2)} = g(z_1^{(2)})$$

$$\begin{aligned} z^{(2)} &= \Theta^{(1)}x = \Theta^{(1)}a^{(1)} \\ a^{(2)} &= g(z^{(2)}) \end{aligned}$$

$$a_2^{(2)} = g(z_2^{(2)})$$

$$\text{Add } a_0^{(2)} = 1 \quad \textbf{Bias unit}$$

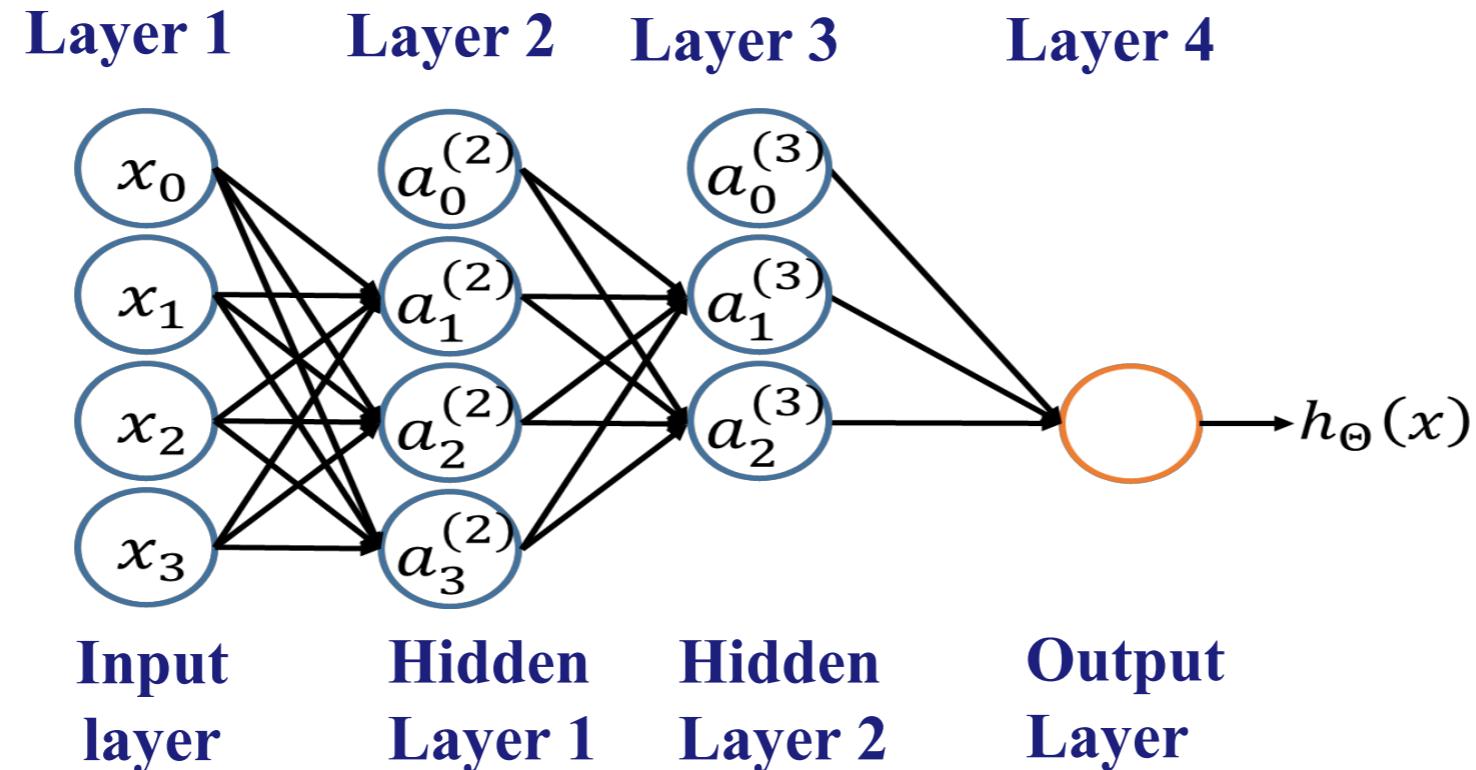
$$a_3^{(2)} = g(z_3^{(2)})$$

$$z^{(3)} = \Theta^{(2)}a^{(2)}$$

$$h_{\Theta}(x) = g(z^{(3)})$$

$$\mathbf{h}_{\Theta}(x) = \mathbf{a}^{(3)} = \mathbf{g}(\mathbf{z}^{(3)})$$

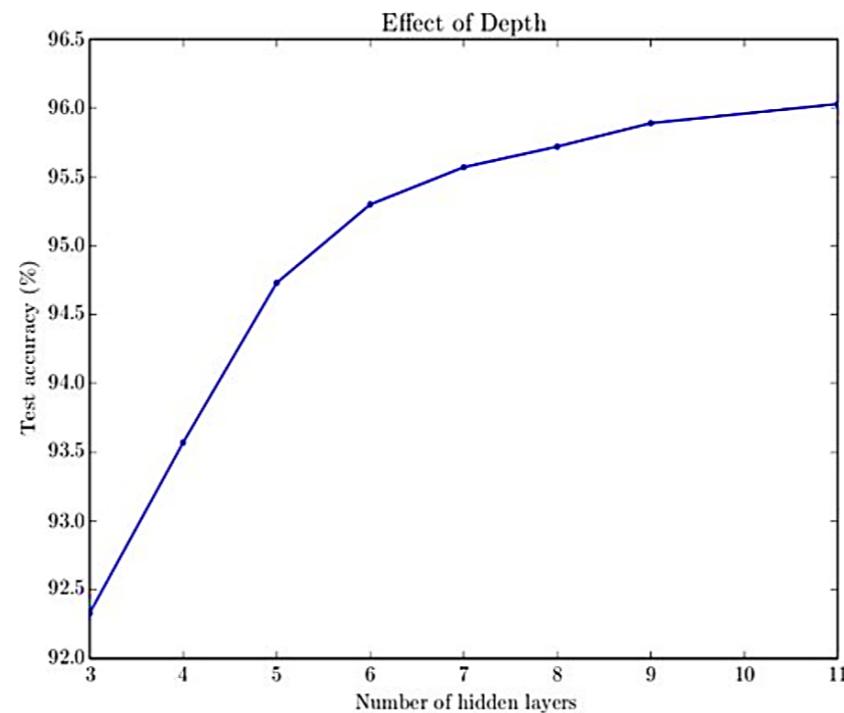
Deep Neural Networks



An ANN having two or more hidden layers counts as “**Deep Neural Networks**”.

Hyperparameters

- How many **hidden layers** do we need?
- In ANNs, hidden layers are required *iff* the data must be separated **non-linearly**.



The deeper networks, the better generalizing.

Hyperparameters

- In case of supervised learning, in order to prevent **over-fitting**, the number of neurons:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

- N_i : number of input neurons
- N_o : number of output neurons
- N_s : number of samples in training data set
- α : number of nonzero weights for each neuron (an arbitrary scaling factor) {2~10} [trial and error to minimize MSE]

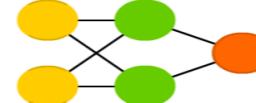
A lot of terminologies for NN!

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

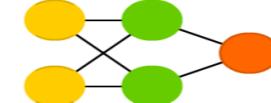
Perceptron (P)



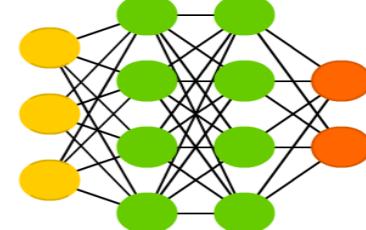
Feed Forward (FF)



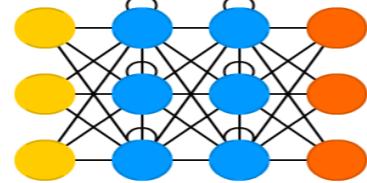
Radial Basis Network (RBF)



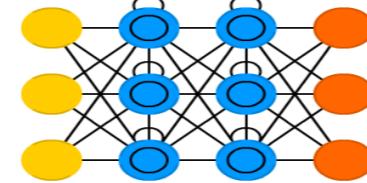
Deep Feed Forward (DFF)



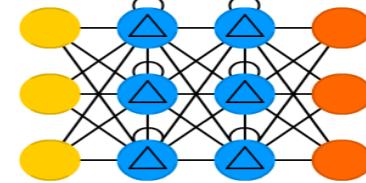
Recurrent Neural Network (RNN)



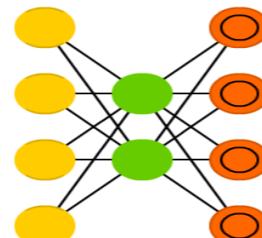
Long / Short Term Memory (LSTM)



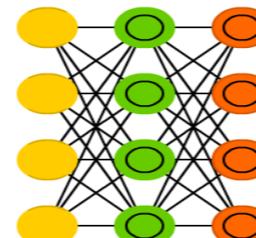
Gated Recurrent Unit (GRU)



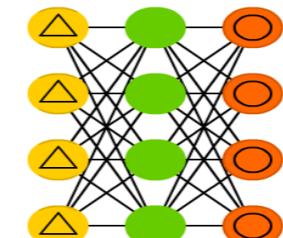
Auto Encoder (AE)



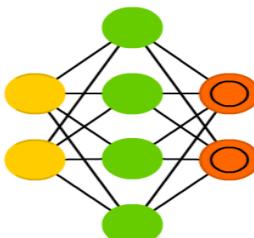
Variational AE (VAE)



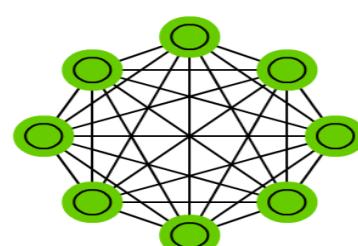
Denoising AE (DAE)



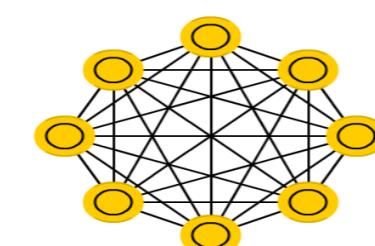
Sparse AE (SAE)



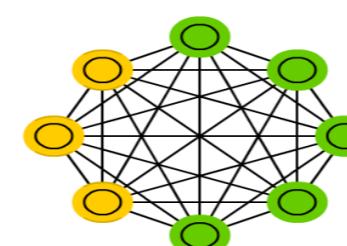
Markov Chain (MC)



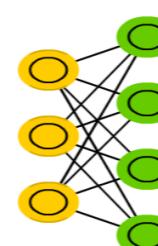
Hopfield Network (HN)



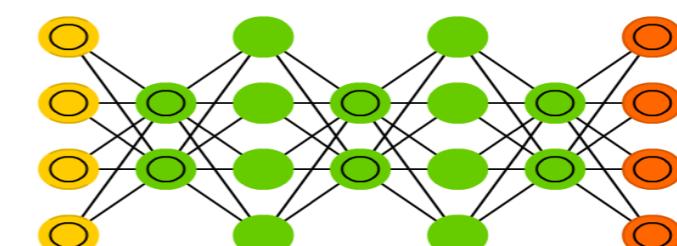
Boltzmann Machine (BM)



Restricted BM (RBM)

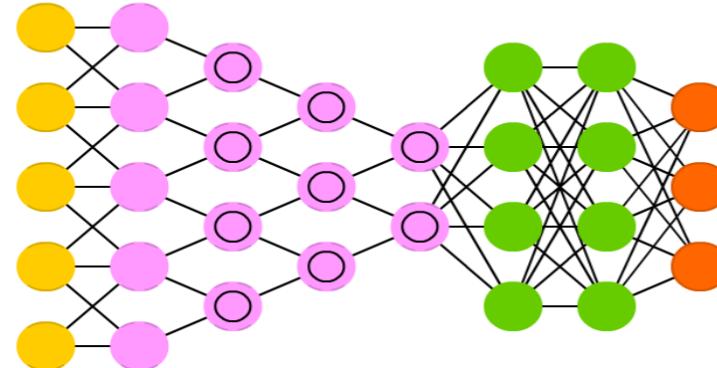


Deep Belief Network (DBN)

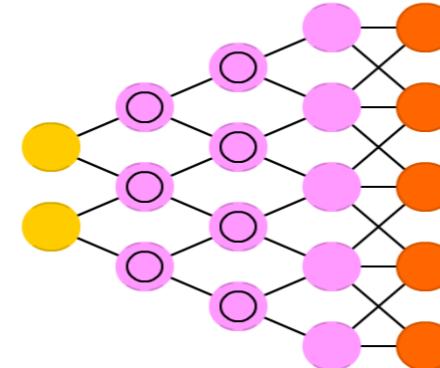


A lot of terminologies for NN!

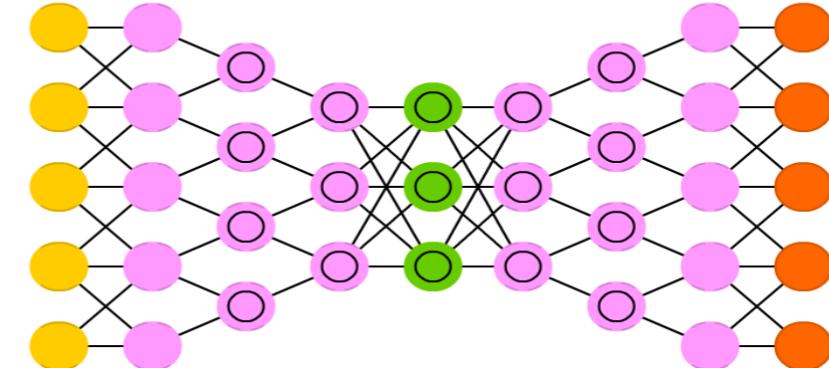
Deep Convolutional Network (DCN)



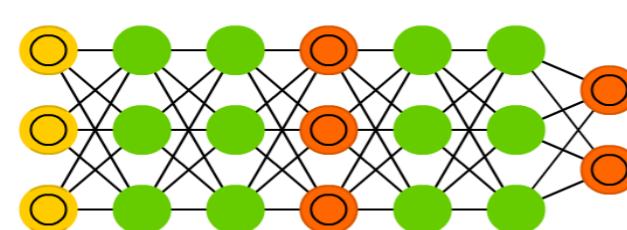
Deconvolutional Network (DN)



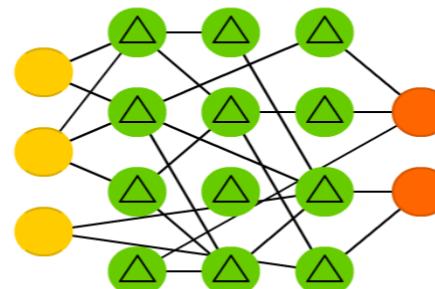
Deep Convolutional Inverse Graphics Network (DCIGN)



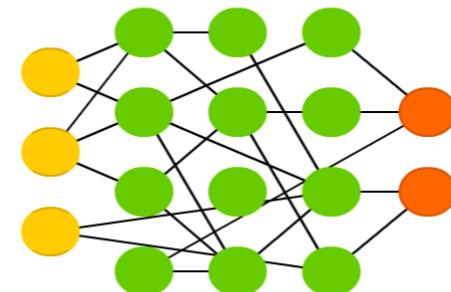
Generative Adversarial Network (GAN)



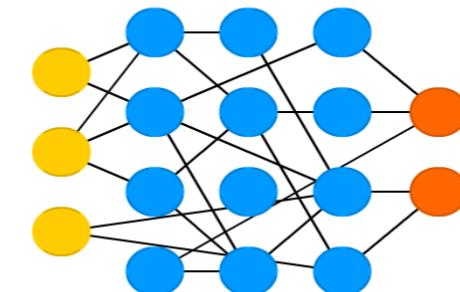
Liquid State Machine (LSM)



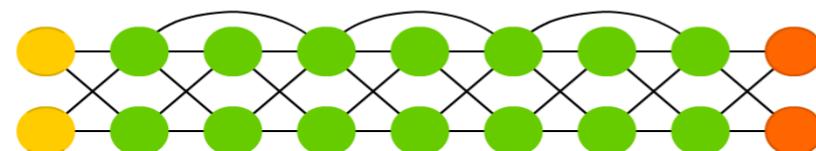
Extreme Learning Machine (ELM)



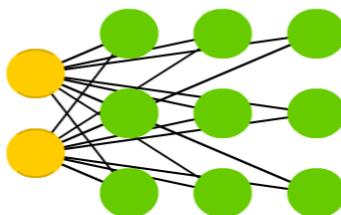
Echo State Network (ESN)



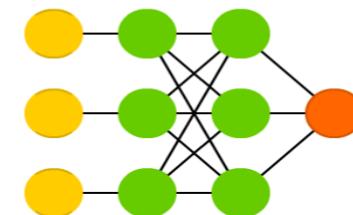
Deep Residual Network (DRN)



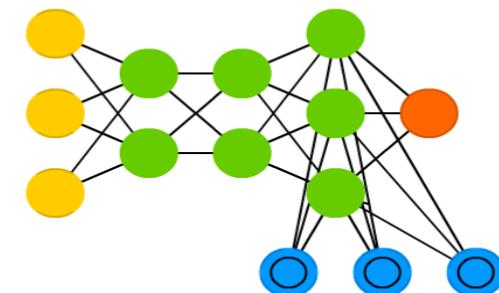
Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)



Multi-class Classification

One-vs-all Algorithm

A computer vision example:



Pedestrian



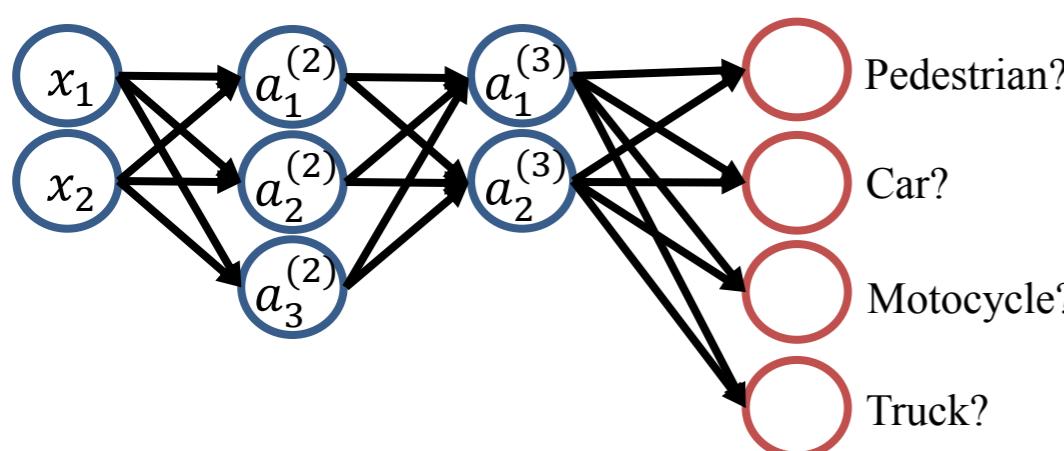
Car



Motorcycle



Truck



Goal:

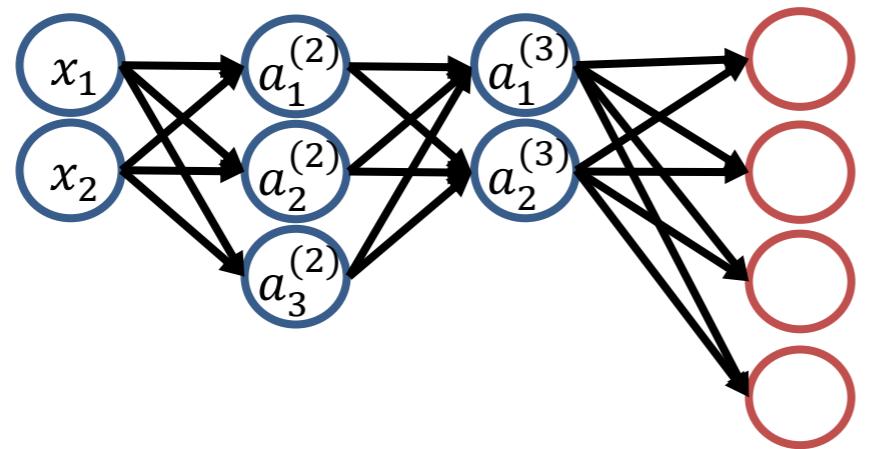
$$h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{When pedestrian}$$

$$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{When car}$$

$$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \text{When motorcycle}$$

etc.

Multi-class Classification



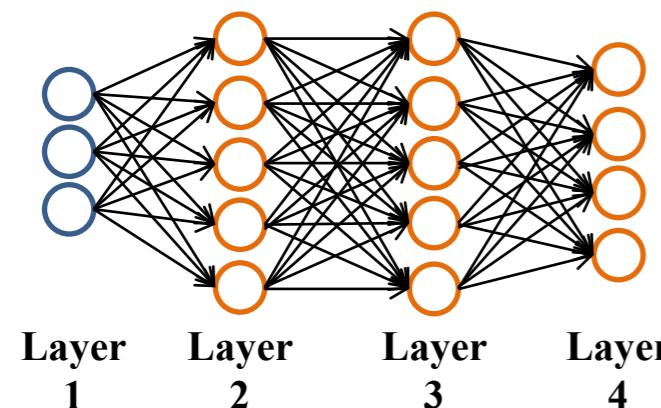
$$h_{\Theta}(x) \in \mathbb{R}^4$$

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

$y^{(i)}$ one of

$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
pedestrian	car	motorcycle	truck

ANN for Classification



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

Types of classification:

Binary classification

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

1 output unit

$$h_{\Theta}(x) \in \mathbb{R}$$

$$S_L = 1$$

'one output'

$$K = 1$$

'one unit'

Multi-class classification (K classes)

$$y \in \mathbb{R}^K \quad \text{E.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

pedestrian car motorcycle truck

$$h_{\Theta}(x) \in \mathbb{R}^K$$

K output units

$$S_L = K$$

$$K \geq 3$$

Cost Function

$$h_{\Theta}(x) \in \mathbb{R}^K$$

$(h_{\Theta}(x))_i = i^{th} output$

$$J(\Theta) = \frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_{\theta}(x^{(i)})_k + (1 - y_k^{(i)}) \log (1 - h_{\theta}(x^{(i)}))_k \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_j^{(l)})^2$$

- **Goal:** to **minimize** the **cost function**

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

- **Gradient Descent**

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$$

Backpropagation Representation

- For each output unit (layer $L = 4$)

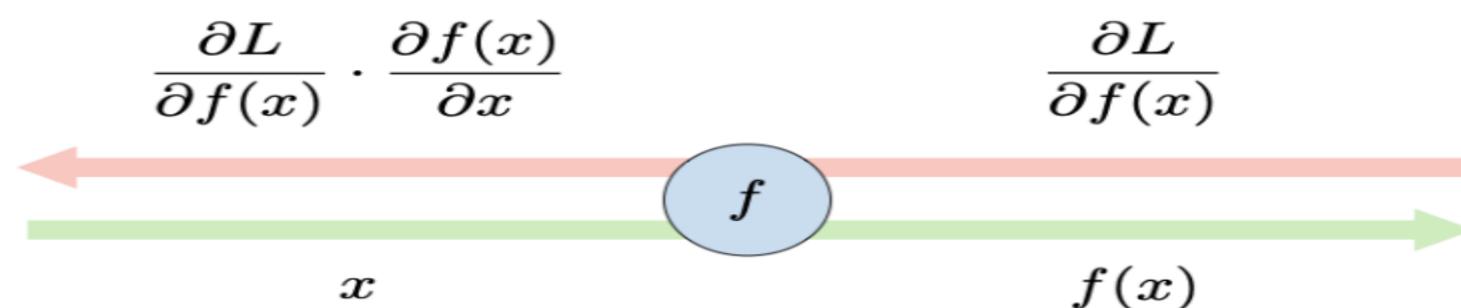
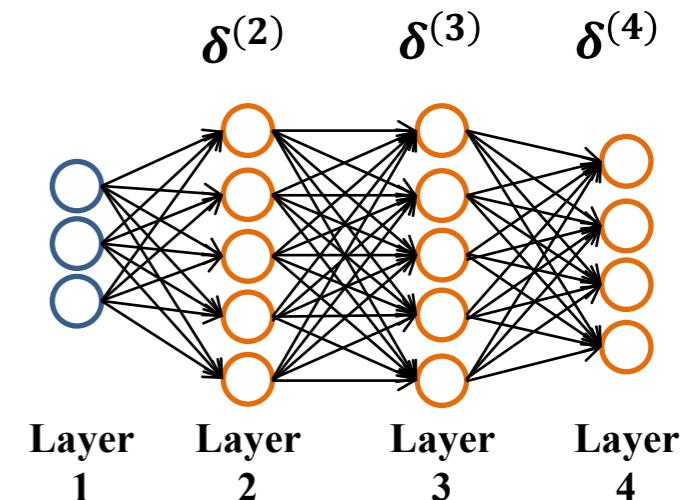
$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \cdot g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \cdot g'(z^{(2)})$$

- Finally:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a^{(l)} \delta_i^{(l+1)} \quad \text{if } \lambda = 0$$



Backpropagation Representation

1. *Training set:* $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
2. *Set* $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$
3. *for* $i = 1$ *to* m
 - 3.1. *Set* $a^{(1)} = x^{(i)}$
 - 3.2. *Forward propagation to compute:* $a^{(l)} \quad \forall l = 2, 3, \dots, L$
 - 3.3. *Using* $y^{(i)}$ *compute* $\delta^{(L)} = a^{(L)} - y^{(i)}$
 - 3.4. *compute* $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$
 - 3.5. $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Backpropagation Representation

$$\begin{cases} \text{if } j \neq 0 \rightarrow D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \\ \text{if } j = 0 \rightarrow D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \end{cases}$$

$$\therefore \frac{\partial}{\partial \Theta_{ij}^{(l)} J(\Theta)} = D_{ij}^{(l)}$$

Conclusion

1. Neural Networks...

- provide a way of learning features
- are highly nonlinear prediction functions
- (can be) a highly parallel network of logistic regression classifiers
- discover useful hidden representations of the input

2. Backpropagation...

- provides an efficient way to compute gradients
- is a special case of reverse-mode automatic differentiation

Appropriate problems for ANN

- Input is high-dimensional discrete or real-valued
(e.g., raw sensor input)
- Output is discrete or real valued
- Output is a vector of values
- Possibly noisy data
- Long training times accepted
- Fast evaluation of the learned function required.
- Not important for humans to understand the weights



Office: #432, Dayang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



Course Outline



- Introduction to AI
- The History of AI
- Python
- Search
 - Informed Search
 - Beyond Classical Search
 - Adversarial Search
 - Constraint Satisfaction Problems
 - Fuzzy Logic
- **Machine Learning**
 - Supervised Learning
 - **Unsupervised Learning**

Problem
Solving

Knowledge,
reasoning, and
planning

Artificial Intelligence

Learning

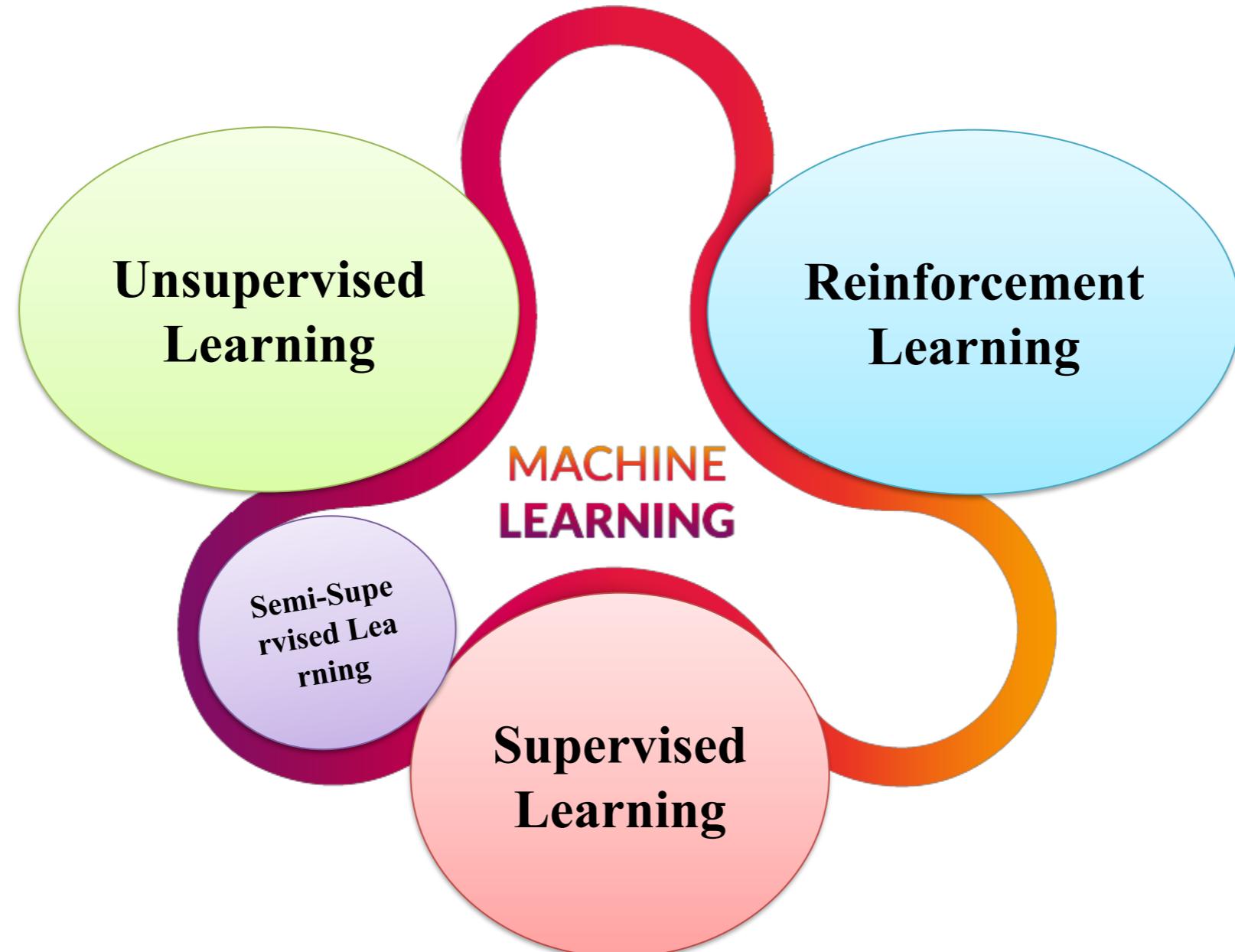
Communications
, perceiving, and
acting

UNSUPERVISED LEARNING

- Principles
- Problems
- Approaches
- Algorithms
- Clustering
- Clustering Methods
- Applications
- K-means
- Hierarchical



Types of Learning



- UL algorithms try to use techniques on the input data to **mine for rules, detect patterns, and summarize and group the data points** which help in deriving meaningful insights and describe the data better to the users.



Learn patterns from (unlabeled) data.

Problems

- **Clustering:** where you want to discover the inherent groupings in the data, such as *grouping customers by purchasing behavior*.
- **Association:** where you want to discover rules that describe large portions of your data, such as *people that buy X also tend to buy Y*.

Approaches

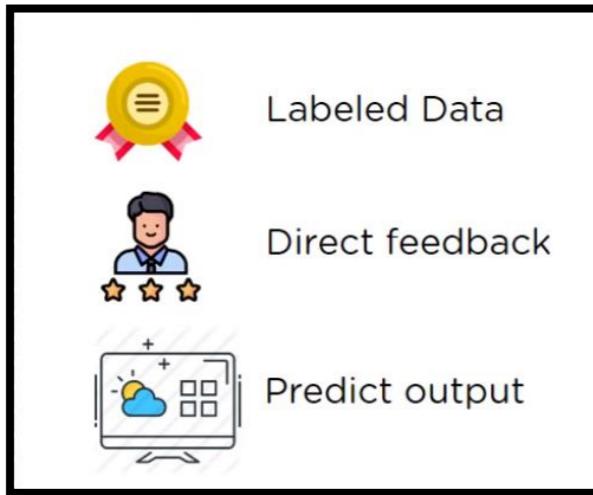
- Clustering (similarity-based)
- Density estimation (e.g., EM algorithm)
- Performance Tasks
- Understanding and visualization
- Anomaly detection
- Information retrieval
- Data compression

Comparison

Supervised

vs.

Unsupervised



Algorithms

- K-means clustering,
- Hierarchical clustering,
- Unsupervised soft-clustering,
- Affinity propagation clustering
- Self-organizing map learning
- Autoencoders
- Adversarial autoencoders
- Non-parametric Bayesian Learning
- Generative Deep Neural networks (GDNN)
- Apriori,
- PCA
- Mixture model
- Gaussian mixture model, Expectation Maximization (EM)
- Dirichlet process

CLUSTERING



- **Cluster:** a collection of data points aggregated together because of certain **similarities**.
- **Clustering:** given a set of data points, each having a set of attributes, and a **similarity measure** among them, find clusters such that:
 - Maximize **intra-cluster similarity**
 - Minimize **inter-class similarity**
 - Classes / labels for each instance are **derived from the data**,
 - e.g., unsupervised classification
- **Similarity Measures:**
 - Distance
 - Other Problem-specific Measures.

Distance measures

- **Euclidean Distance:**

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

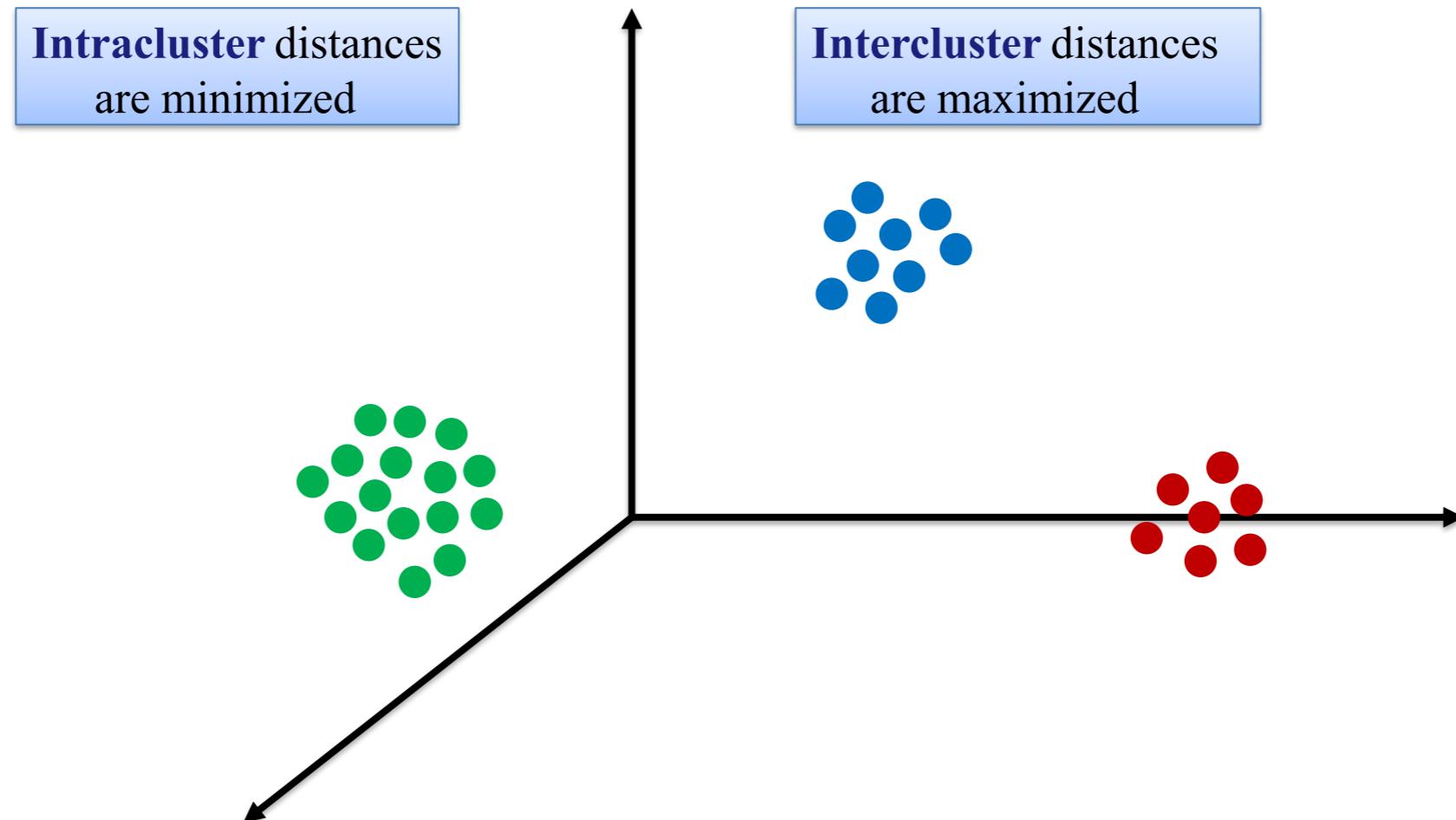
- **Manhattan distance**

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- **Correlation distance**

$$d(x, y) = 1 - r(x, y)$$

- **Euclidean Distance**
 - - Clustering in 3-D space.



Distance measures

- **Distance of sequences**
 - **Hamming Distance**
 - the number of positions at which the corresponding symbols are different.

ACCTTG

Distance is 3

TACCTG

- **Levenshtin distance;**
 - The minimum number of single-character edits required to change one word into the other.

H		O	N	D	A	
H	Y	U	N	D	A	I

Distance is 3

Clustering Methods

• Density-based

- Consider the cluster as the dense region having some similarities and differences from the lower dense region of the space.

- e.g., DBSCAN:

• Hierarchical

- Form a tree-type structure based on the hierarchy.

- e.g., CURE

• Positioning

- Partition the objects into k clusters and each partition forms one cluster.

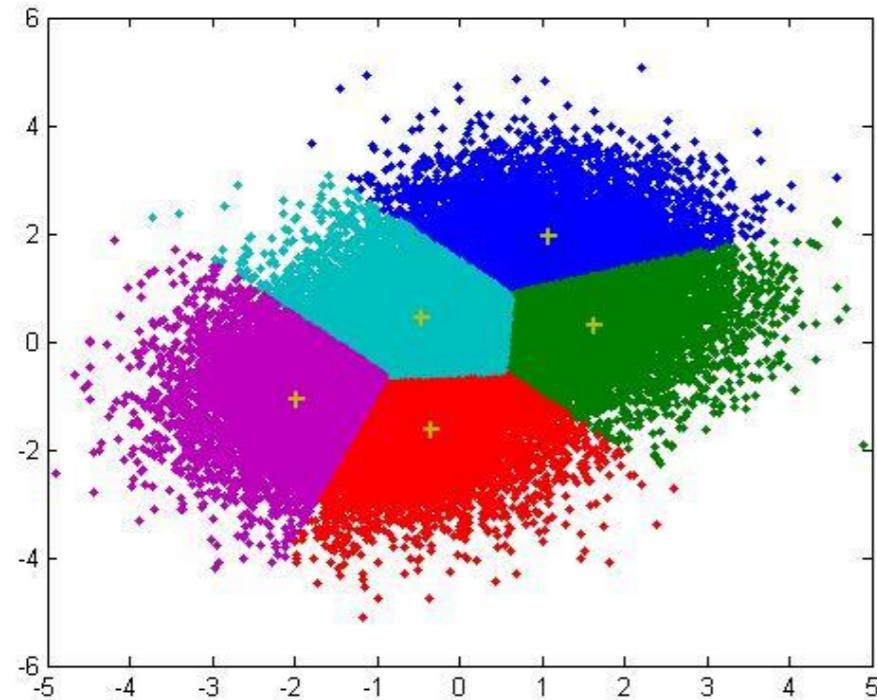
- e.g., K-means

• Grid-based

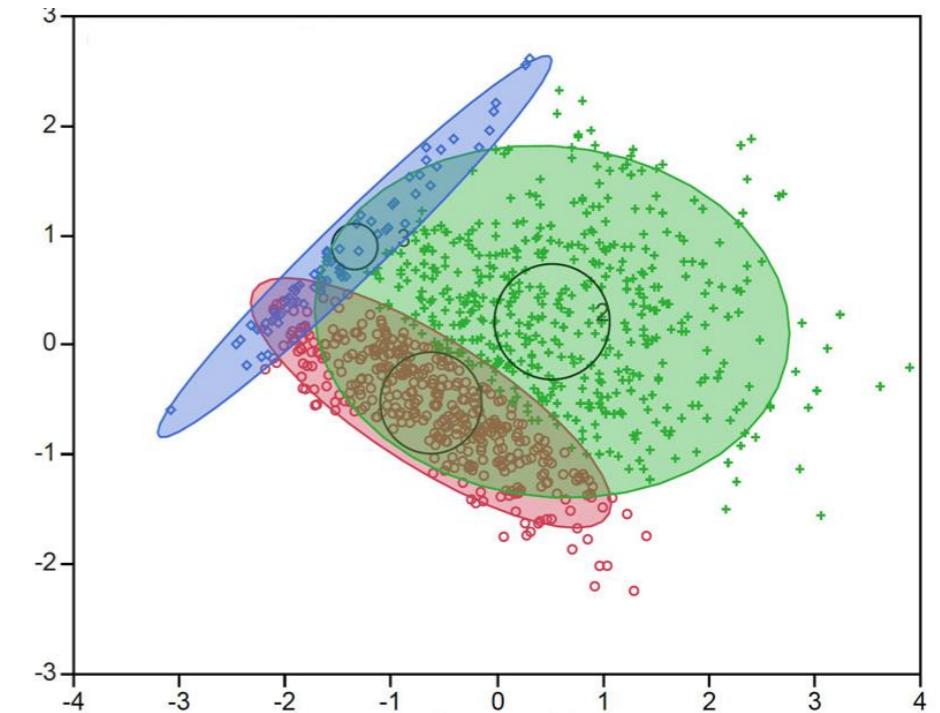
- The data space is formulated into a finite number of cells that form a grid-like structure

- e.g., sting: statistical information grid

Geometry of clustering

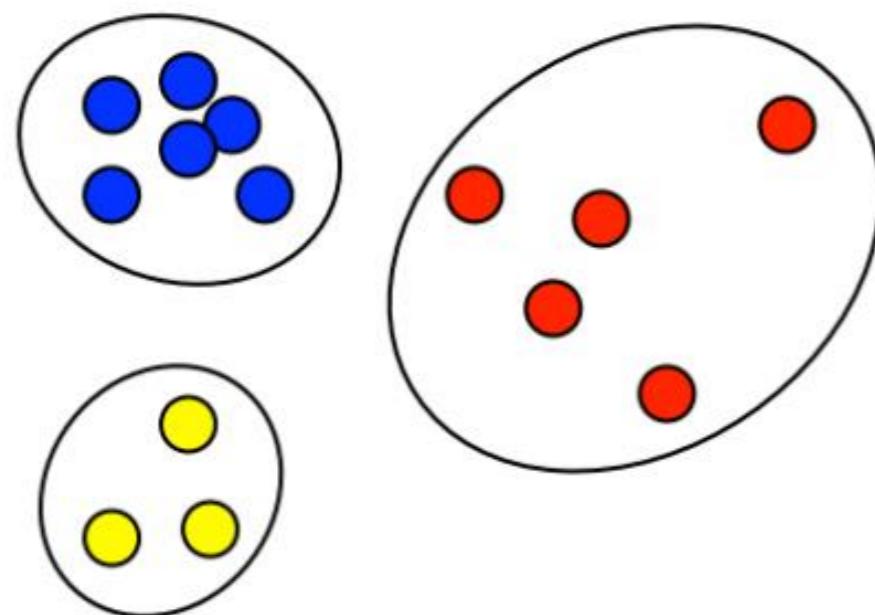


k -means

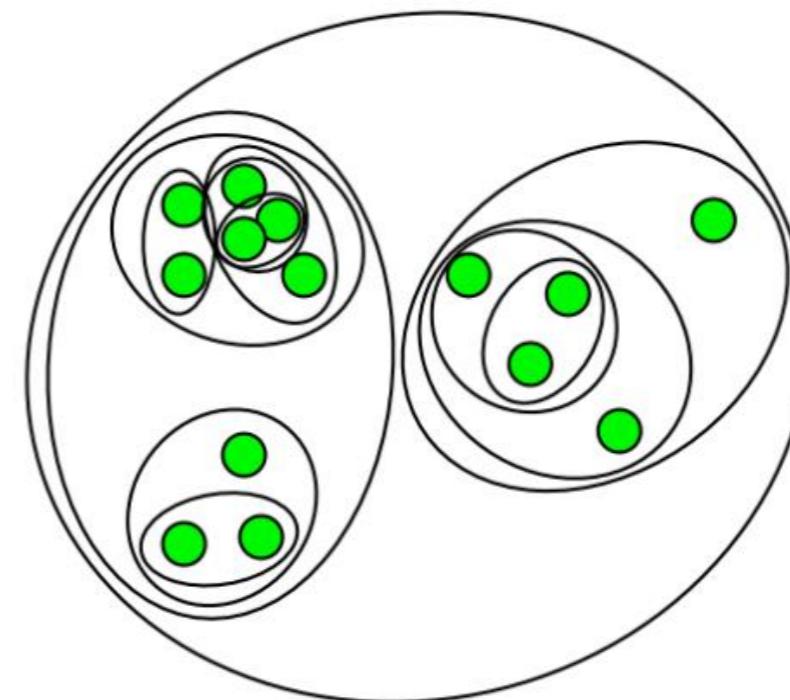


Gaussian mixture model

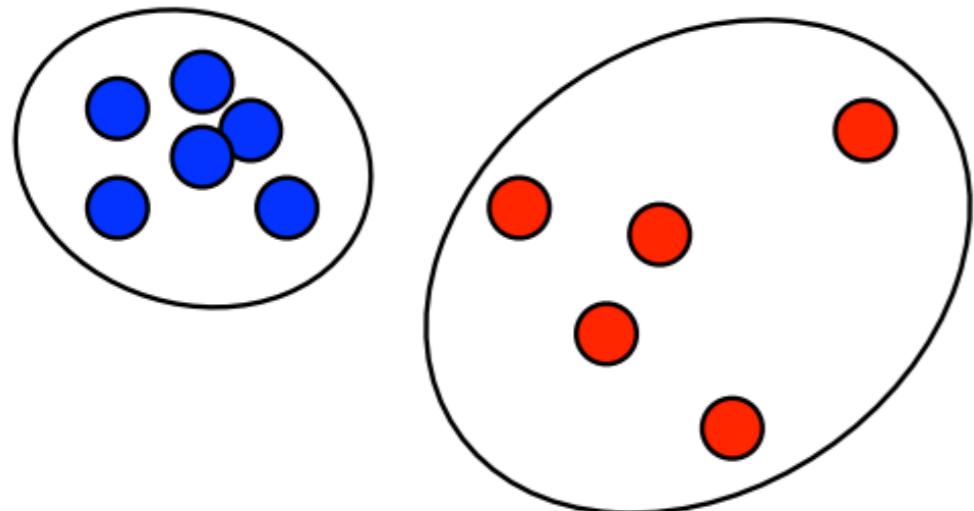
Partitioned clustering



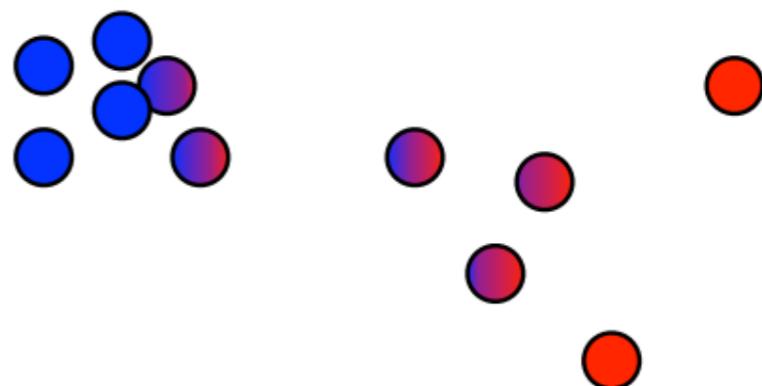
Hierarchical clustering



Crisp



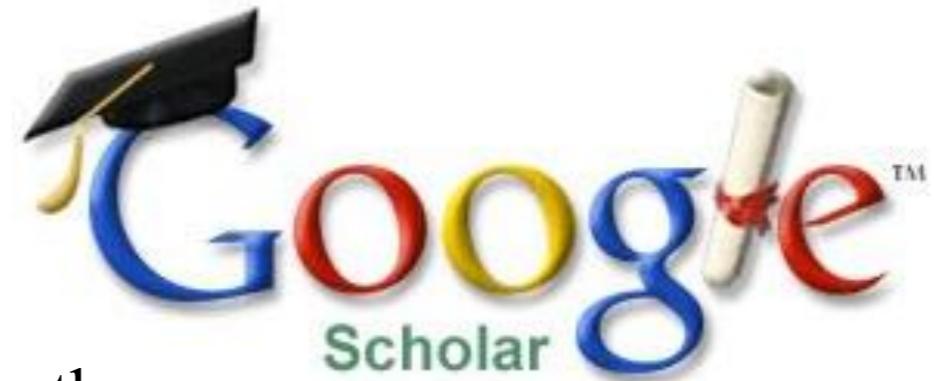
Fuzzy



Applications

Google Scholar

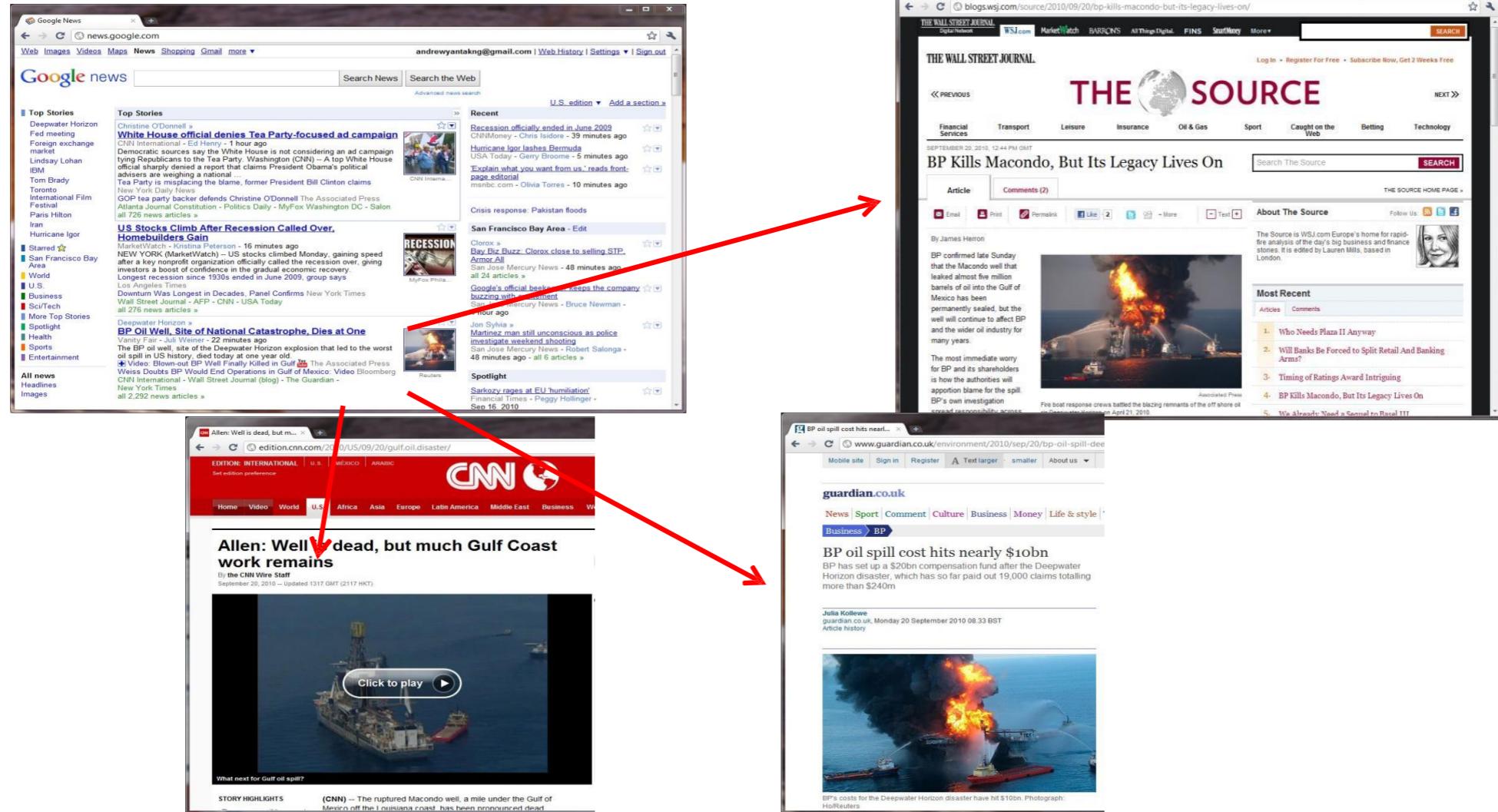
- **Goal:** To find groups of documents that are similar to each other
 - e.g. based on the important terms appearing in them.



Clustering

Applications

- News Clustering



The figure displays four web browser windows showing news coverage of the Deepwater Horizon oil spill:

- Google News (Top Left):** Shows a search results page for "Deepwater Horizon". Headlines include "Christine O'Donnell backs Tea Party-focused ad campaign" and "U.S. Stocks Climb After Recession Called Over". A red arrow points from this window to the CNN window.
- CNN (Bottom Left):** Shows a news article titled "Allen: Well dead, but much Gulf Coast work remains". A red arrow points from this window to the WSJ Source window.
- WSJ Source (Top Right):** Shows a news article titled "BP Kills Macondo, But Its Legacy Lives On". It includes a large image of a burning oil rig.
- Guardian.co.uk (Bottom Right):** Shows a news article titled "BP oil spill cost hits nearly \$10bn". It also includes a large image of a burning oil rig.

Clustering



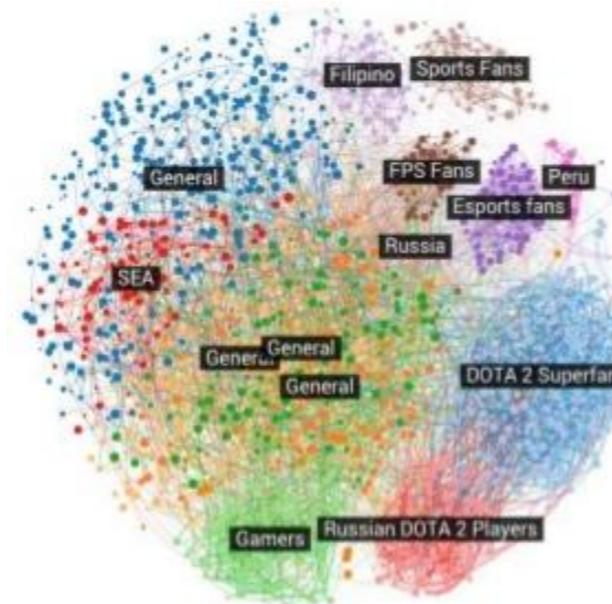
Applications



Market Segmentation



Image Compression

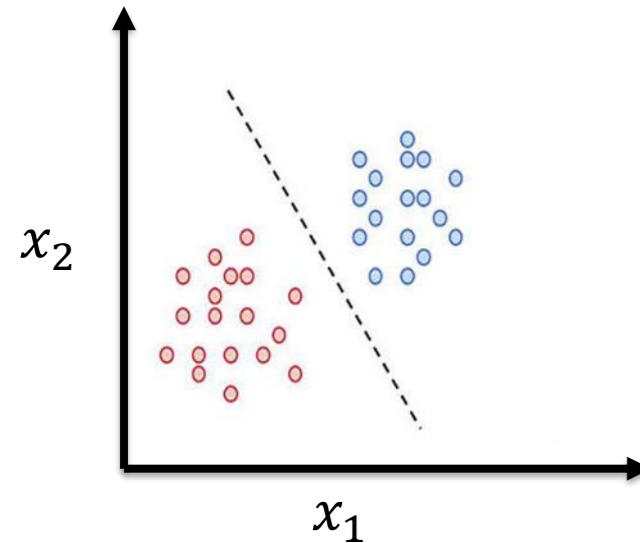


Social network analysis



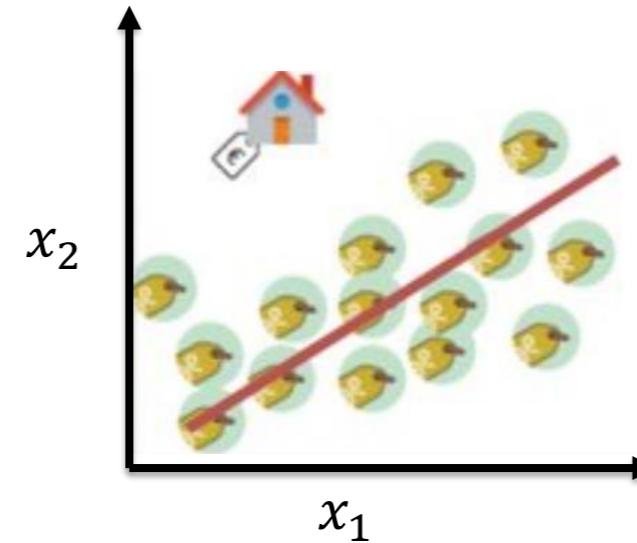
Data center
(computation clustering)

Classification



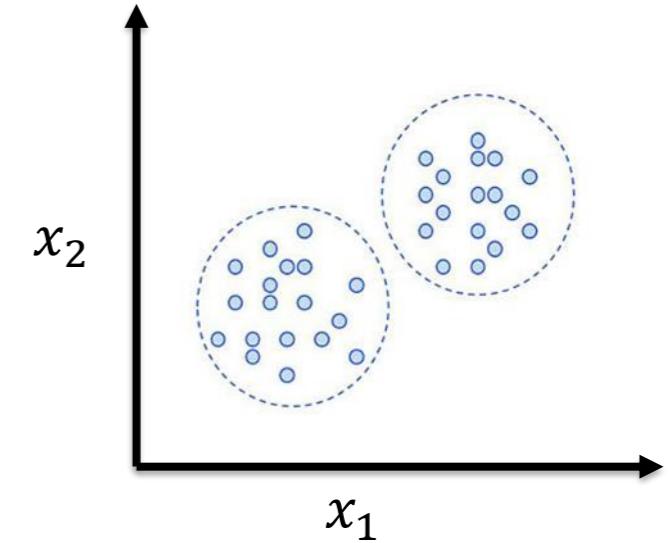
Spam filtering

Regression



House price estimation

Clustering



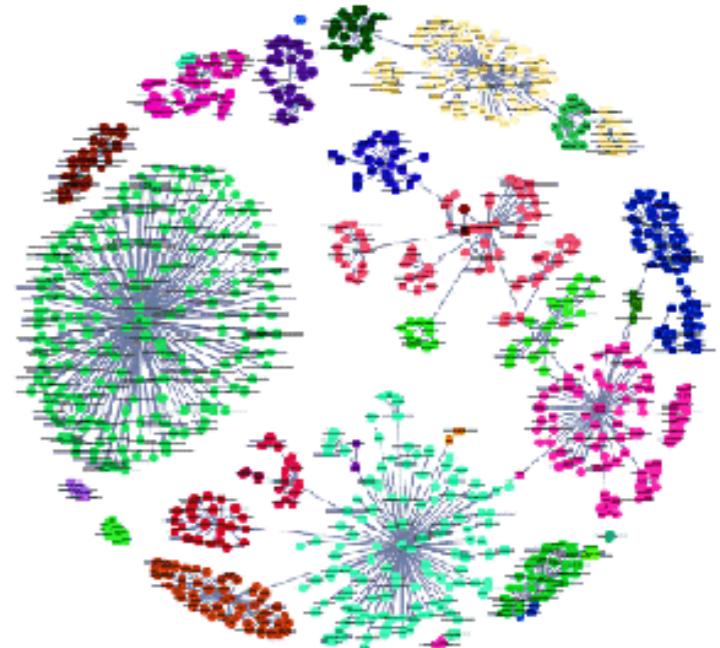
Customers Clustering

K-means

A **partitioned** and **non-fuzzy** technique.

Goal: group similar data points together and discover underlying patterns.

Centroids: to represent clusters by optimizing the squared error function.



K-means

The Algorithm

Randomly selects K points, e.g., centroids.

Repeat:

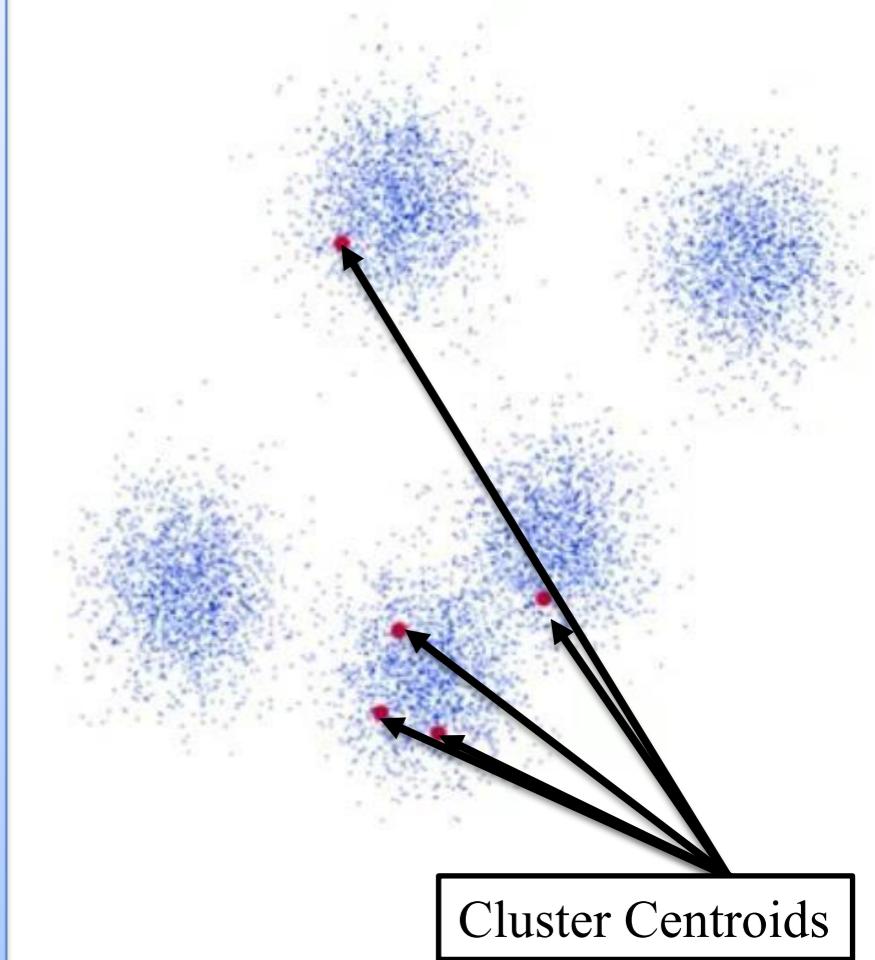
 Assign each data point to the nearest cluster center

 Update the center point according to the mean of the data points.

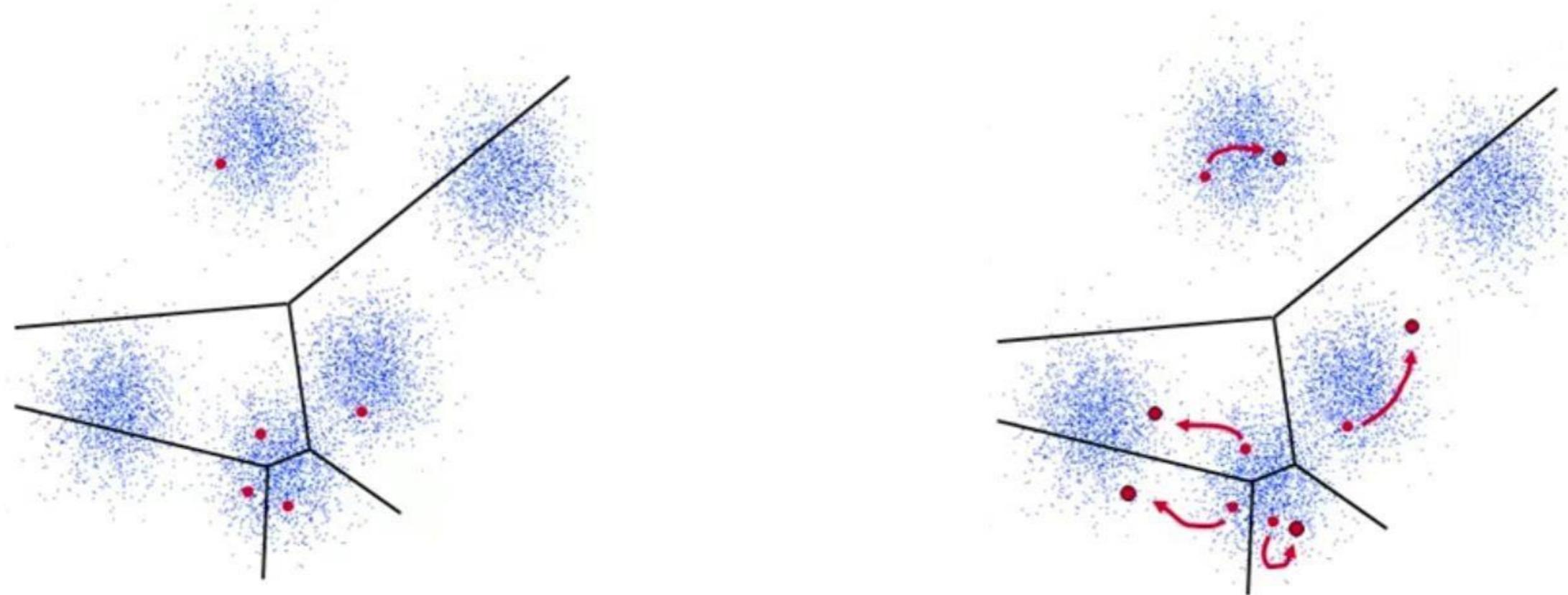
Terminate:

 The centroids have stabilized.

 The defined number of iterations has been achieved.



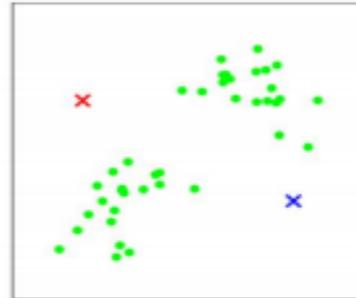
K-means



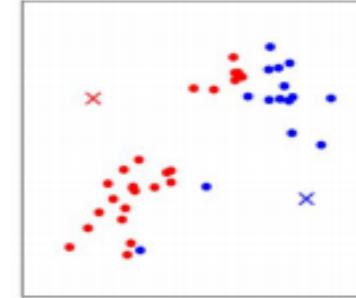
Graphical Representation



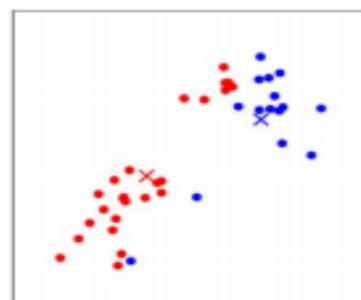
Data



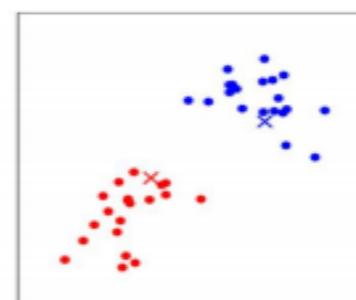
Randomly select the centroids



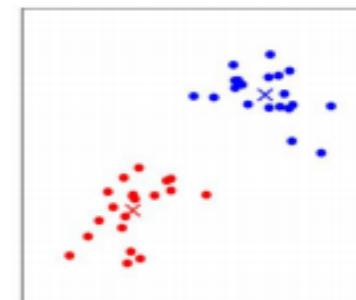
Assign data points to the clusters



Update the centroids

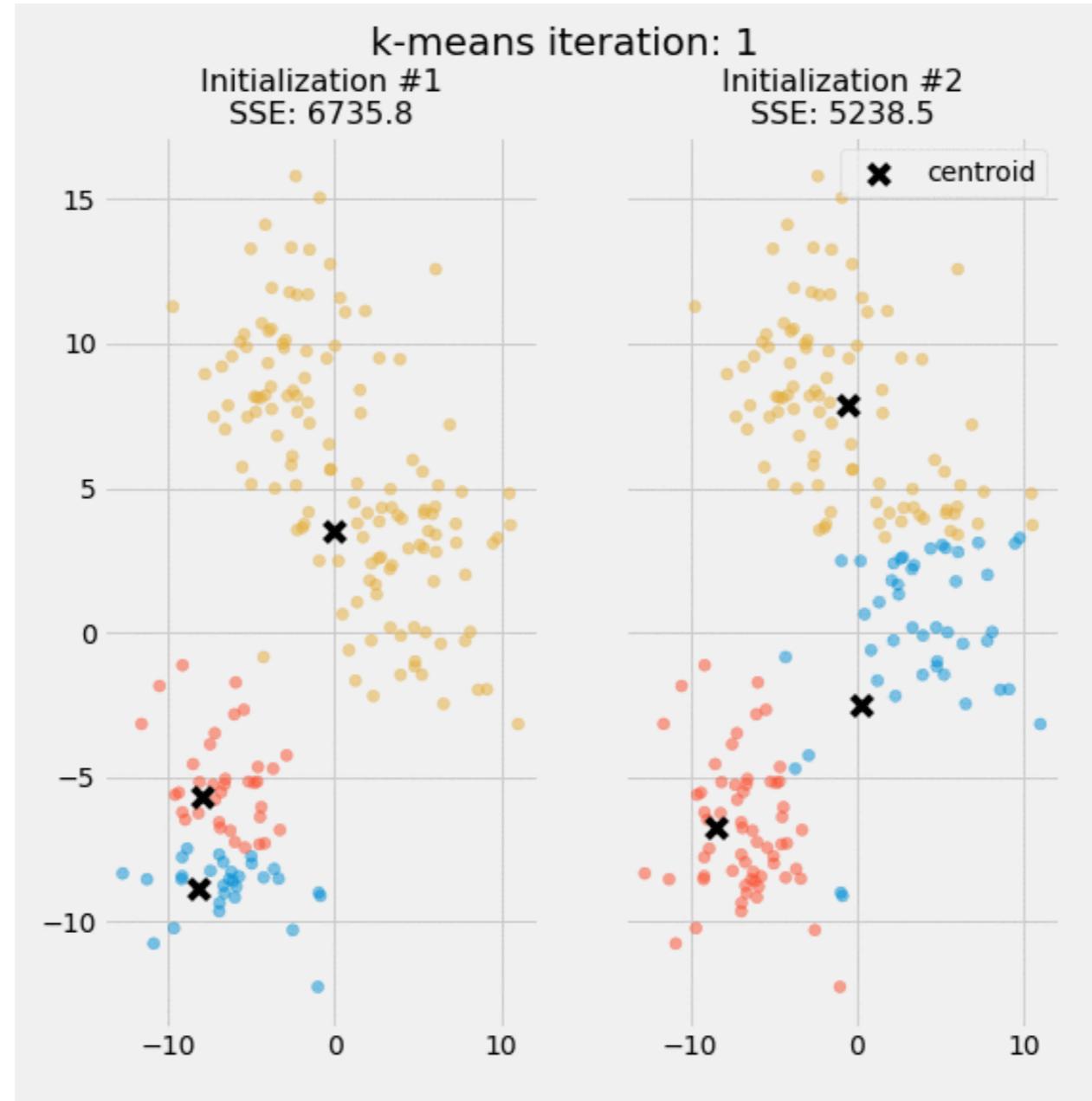


Assign data points to the clusters

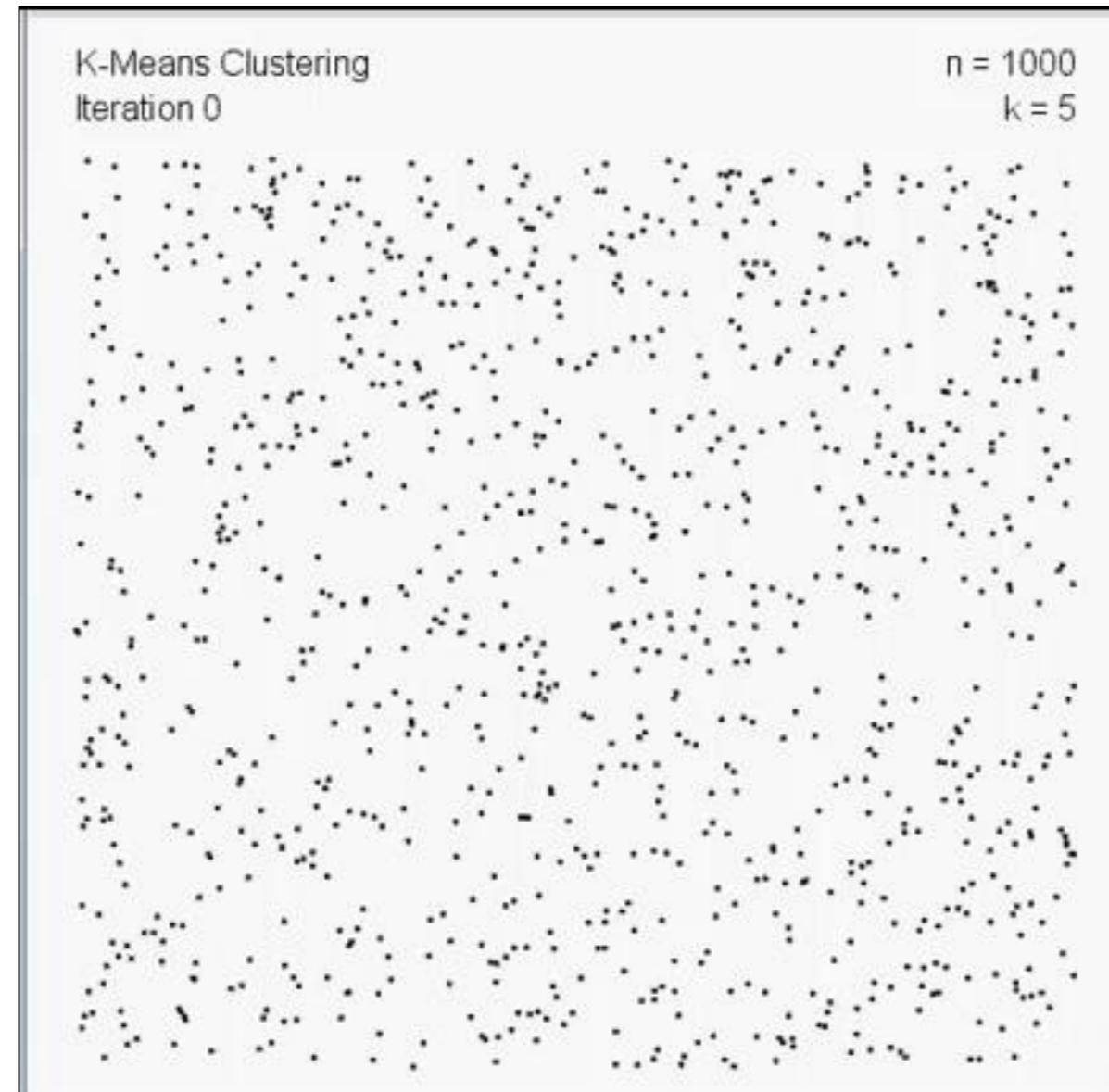


Update the centroids

Graphical Representation



Graphical Representation



K-means

Mathematical Representation

Input:

K : Number of clusters:

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

No label for the training set.

No bias, e.g. $x_0 = 1$

Mathematical Representation

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

repeat until convergence:

{

for $i = 1$ *to* m

$$c^i := \arg \min_k \|x^{(i)} - \mu_k\|$$

for $k = 1$ *to* K

$$\mu_k := \frac{\sum_{i=1}^m 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)}=k\}}$$

}

K-means

Cost function

$$J(C^{(1)}, C^{(2)}, \dots, C^{(m)}, \mu_1, \mu_2, \dots, \mu_k) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Objective

$$\min_{\substack{C^{(1)}, C^{(2)}, \dots, C^{(m)}, \\ \mu_1, \mu_2, \dots, \mu_k}} J(C^{(1)}, C^{(2)}, \dots, C^{(m)}, \mu_1, \mu_2, \dots, \mu_k)$$

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

repeat

{

for $i = 1$ **to** m

$$C^i = \arg \min_k \|x^{(i)} - \mu_k\|$$

for $k = 1$ **to** K

μ_k = average of points assigned to cluster k

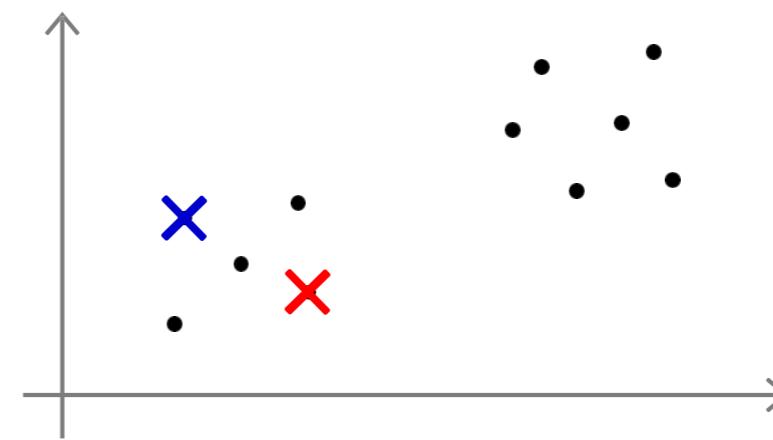
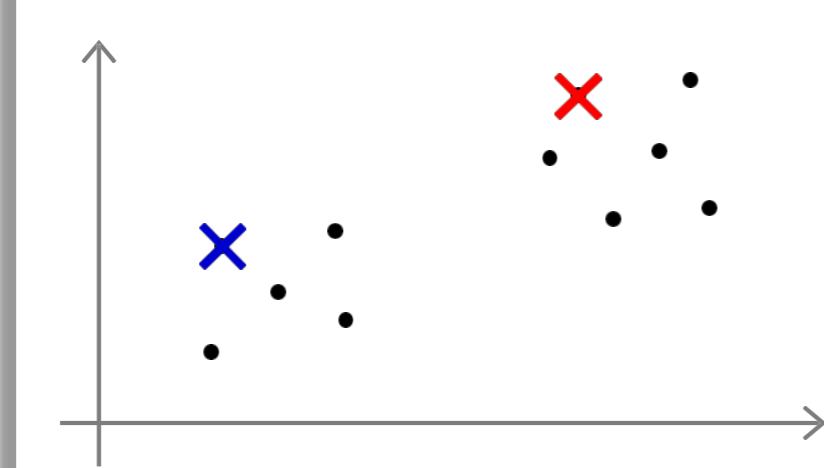
}

Centroid initialization

The initial value: $K \leq m$

Randomly select K training sample

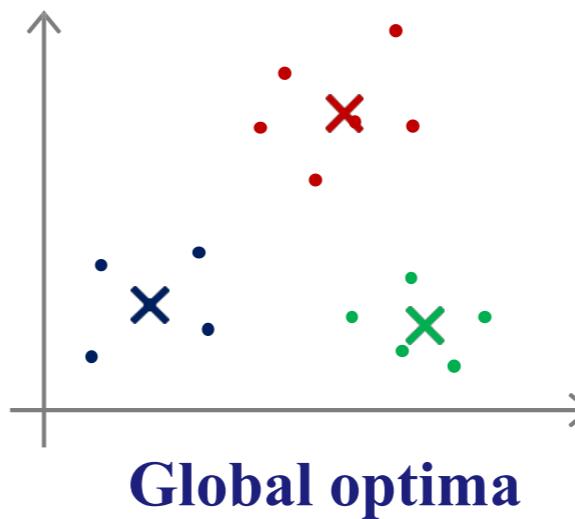
Assign the centroids to the clusters



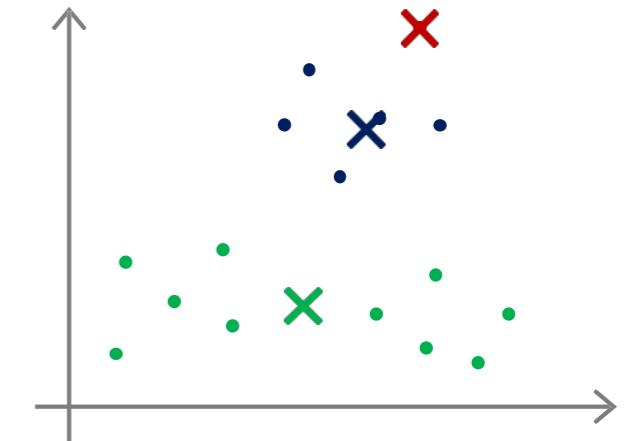
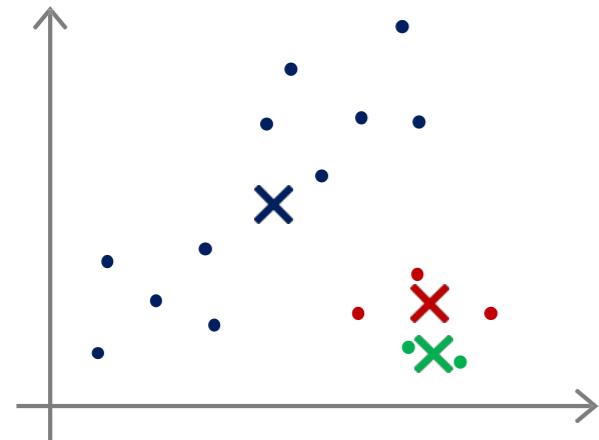
K-means

Optimization

Local optima



Global optima



K-means failed!

Avoid local optima

for $t = 1$ to MAX

{

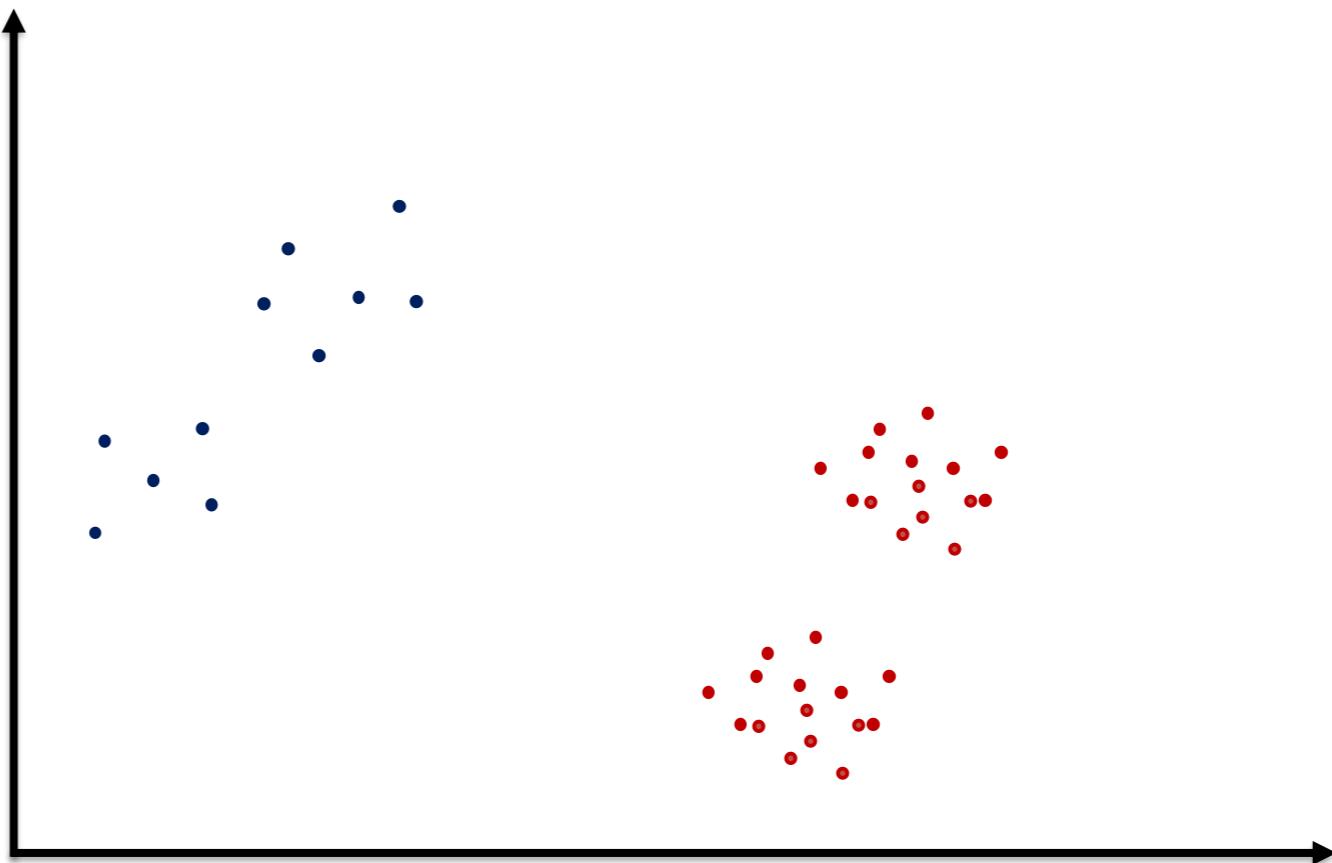
randomly initialize cluster centroids $\mu_1, \mu_2, \dots, \mu_k$

run K-means to get $C^{(1)}, C^{(2)}, \dots, C^{(3)}, \mu_1, \mu_2, \dots, \mu_k$

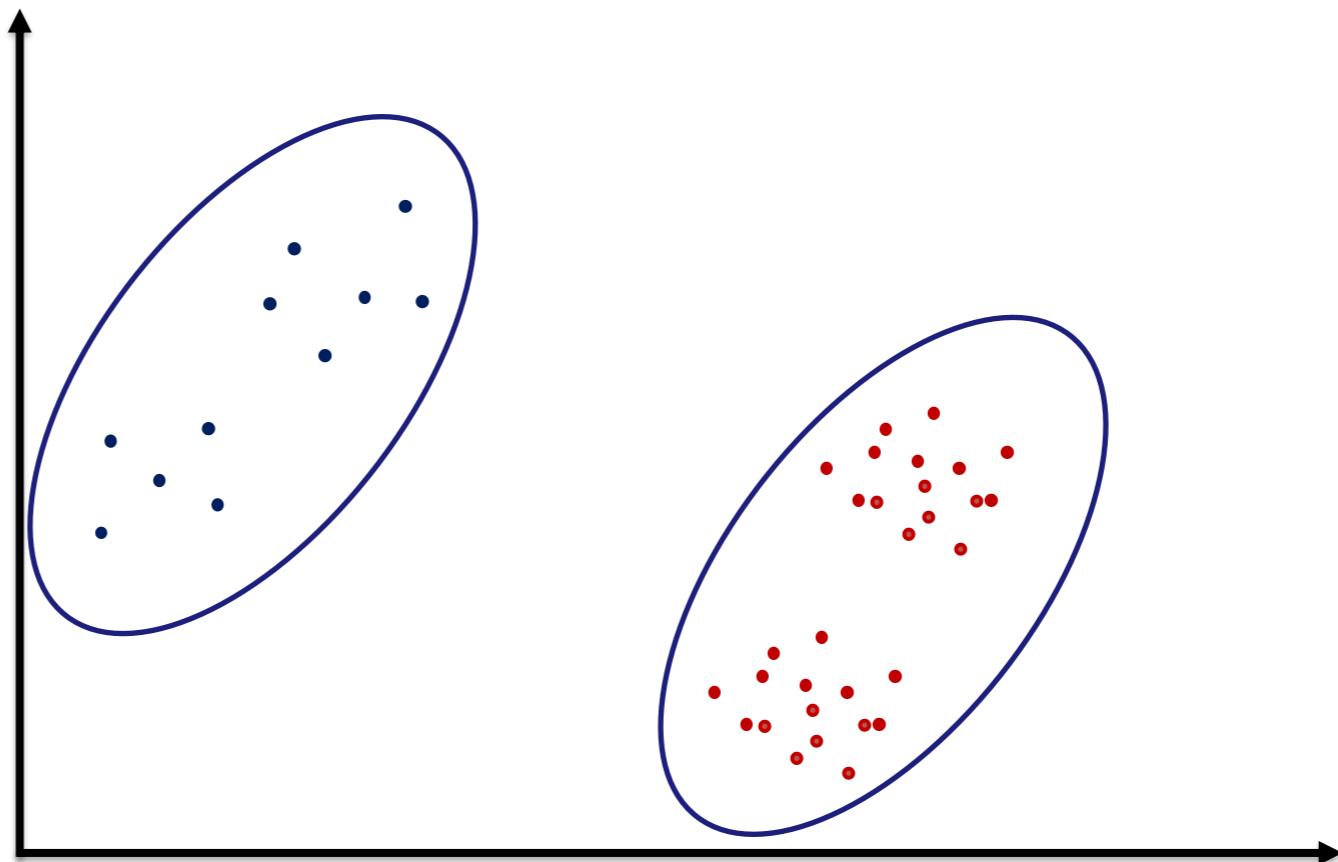
compute cost function $J(C^{(1)}, C^{(2)}, \dots, C^{(m)}, \mu_1, \mu_2, \dots, \mu_k)$

} peak clustering with **minimum cost**

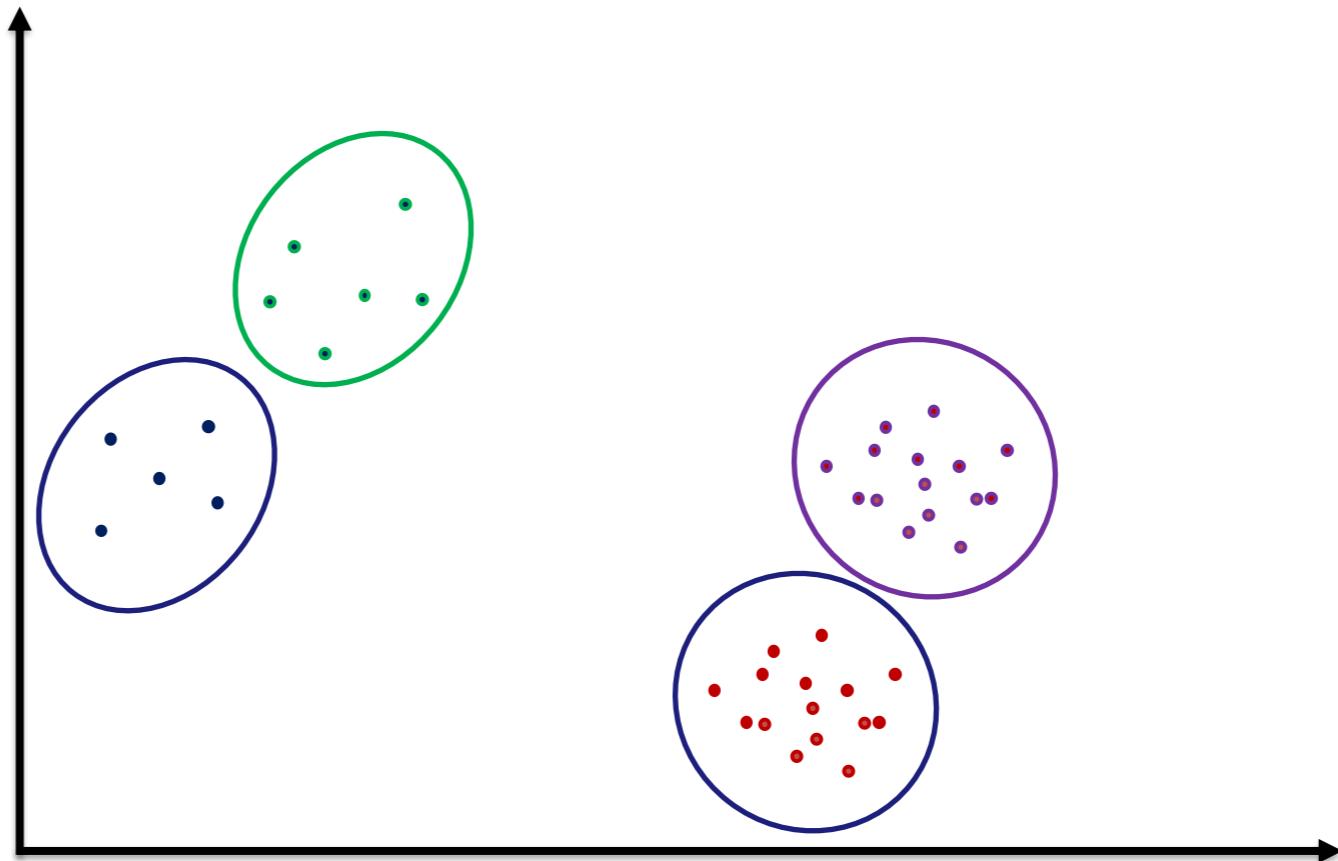
Choosing the best K !



Choosing the best K!

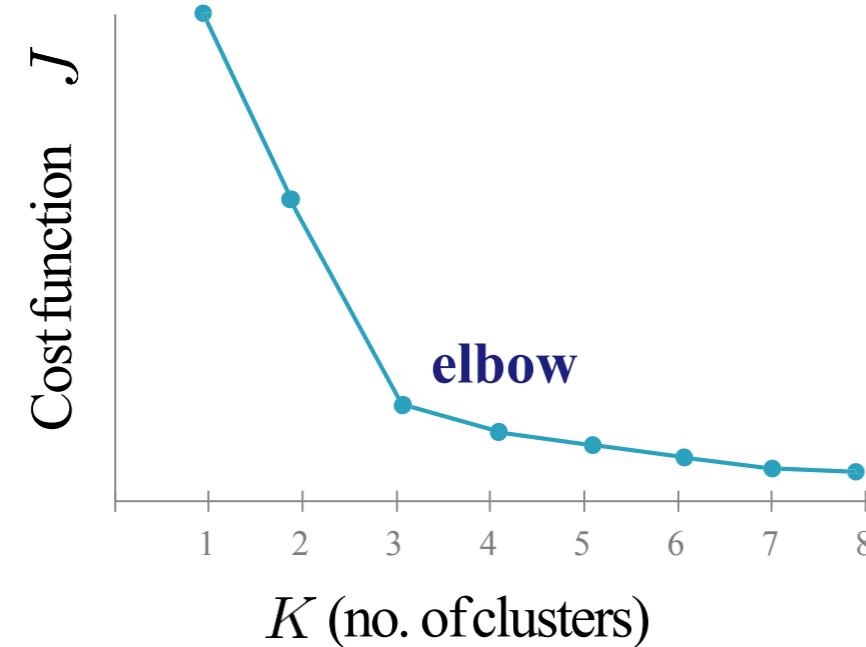


Choosing the best K!

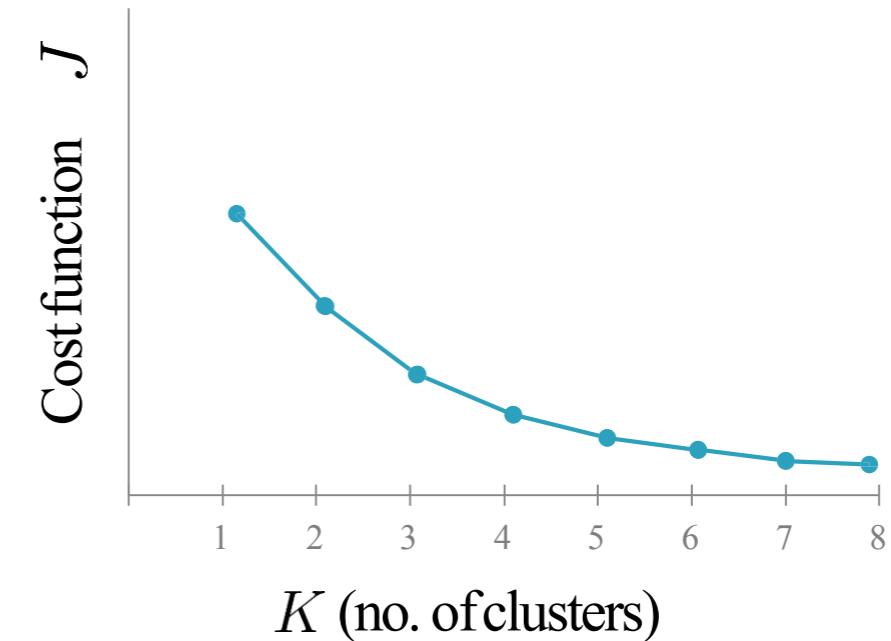


Choosing the best K!

The “elbow” method



3 clusters is suggested!



No elbow!

Improve the cluster by backpropagation

Division

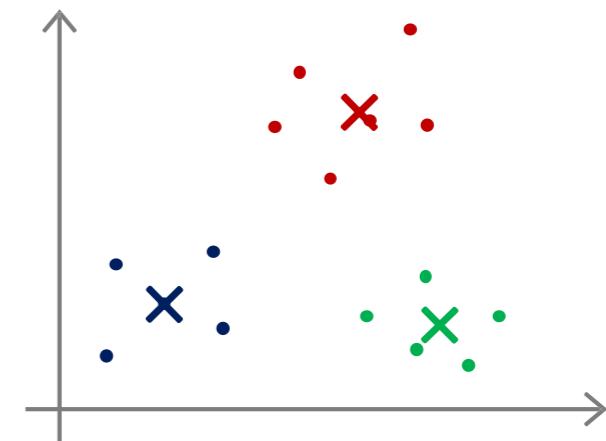
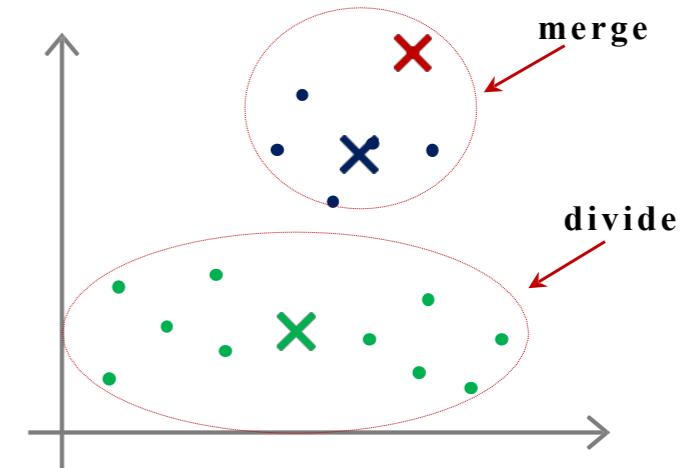
Divide a cluster with the **maximum error** into two clusters

Apply k-means to the data points of this cluster using $K = 2$.

Merge

Merge two cluster that are very close to each other.

Merge two clusters with **minimum** increase of the **cost function**.



“two-part maker” algorithm

Start with all the points in one cluster

While the number of clusters is less than K

 measure the total error

 for every cluster:

 perform K-means clustering with $k = 2$ on the given cluster

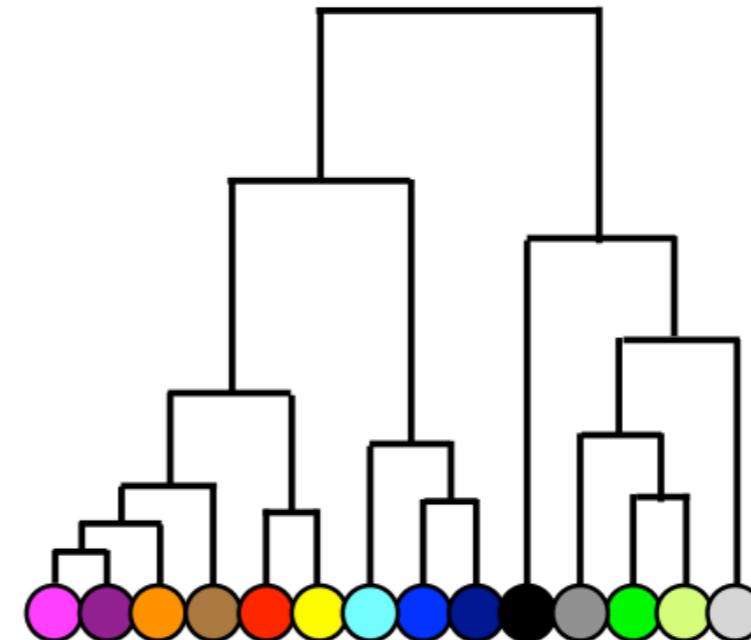
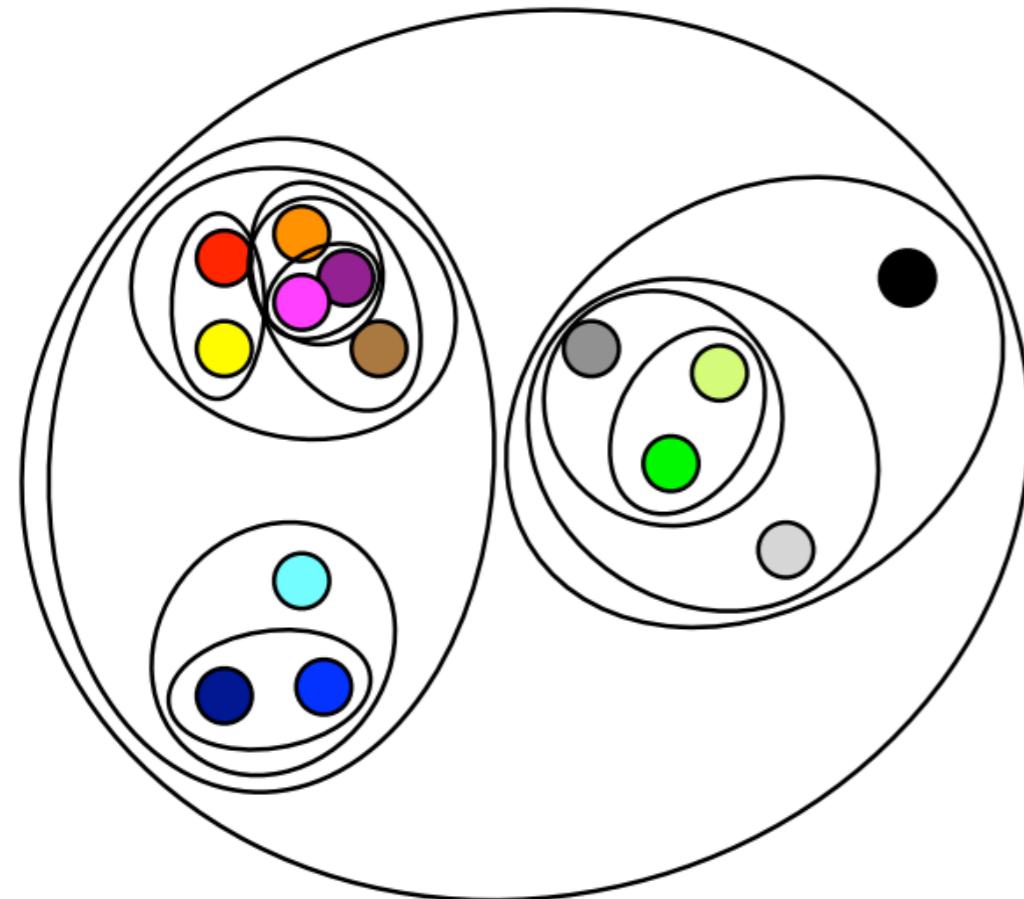
 measure the total error after splitting

 choose the cluster that gives the lowest error

Hierarchical clustering



Hierarchical clustering is usually depicted as a dendrogram (tree)



Hierarchical clustering

First **merge** very similar data points

Gradually, by merging the smaller clusters, **generate bigger clusters**.

Initially, each data point is considered as a cluster.

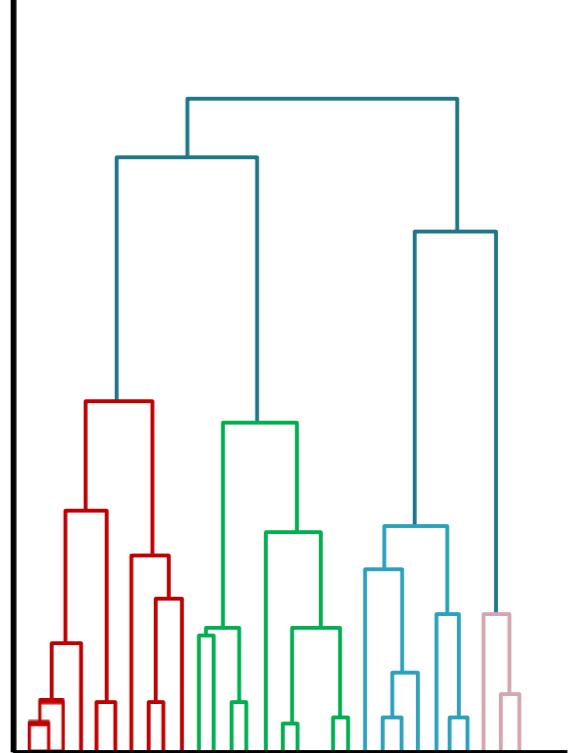
Repeat {

Select the closest two cluster to each other

Merge those two clusters

Stop when there is only one cluster

}



Hierarchical clustering

To define the closest two clusters to each other:

Closest pair (single-...)

Furthest pair (all-)

Average distance of all pairs

“enter” method (minimum dispersion)

Clustering



Comparison

Algorithm	Parameters	Scalability	Use-case	Metrics
k-means	Number of clusters	Very large $n_{samples}$, medium $n_{clusters}$ with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distance between points
Affinity propagation	Damping, sample preference	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distance between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large n_{sample} and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distance between points
Agglomerative clustering	Number of clusters, linkage type, distance	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints, non-Euclidean distance	Any pairwise distance
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster size	Distance between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distance to center
Birch	Branching factor, threshold, optional global cluster	Large $n_{clusters}$ and $n_{samples}$	Large dataset, outlier removal, data reduction	Euclidean distance between points



Office: #432, Daeyang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



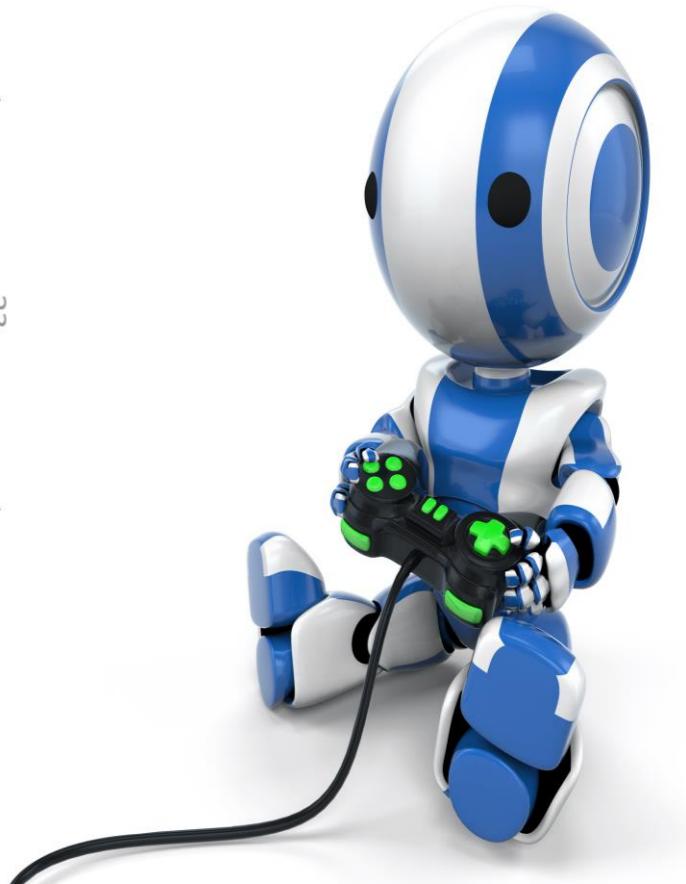
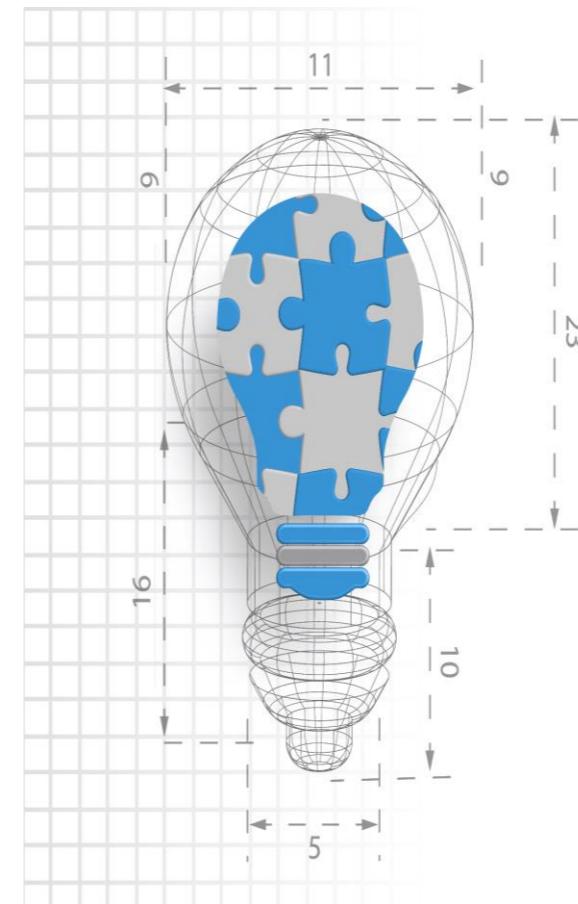
Course Outline



- Introduction to AI
- The History of AI
- Python
- Search
 - Search
 - Informed Search
 - Beyond Classical Search
 - Adversarial Search
 - Constraint Satisfaction Problems
 - Fuzzy Logic
- **Machine Learning**
 - Supervised Learning
 - Unsupervised Learning
 - **Reinforcement Learning**

REINFORCEMENT LEARNING

- Reinforcement Learning (RL)
- Markov Decision Process (MDP)
- Q-Learning



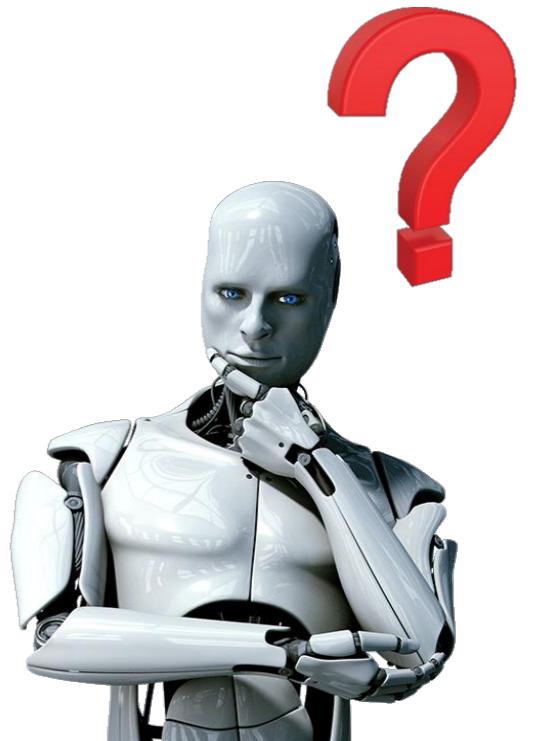
Machine learning



**Supervised
Learning**



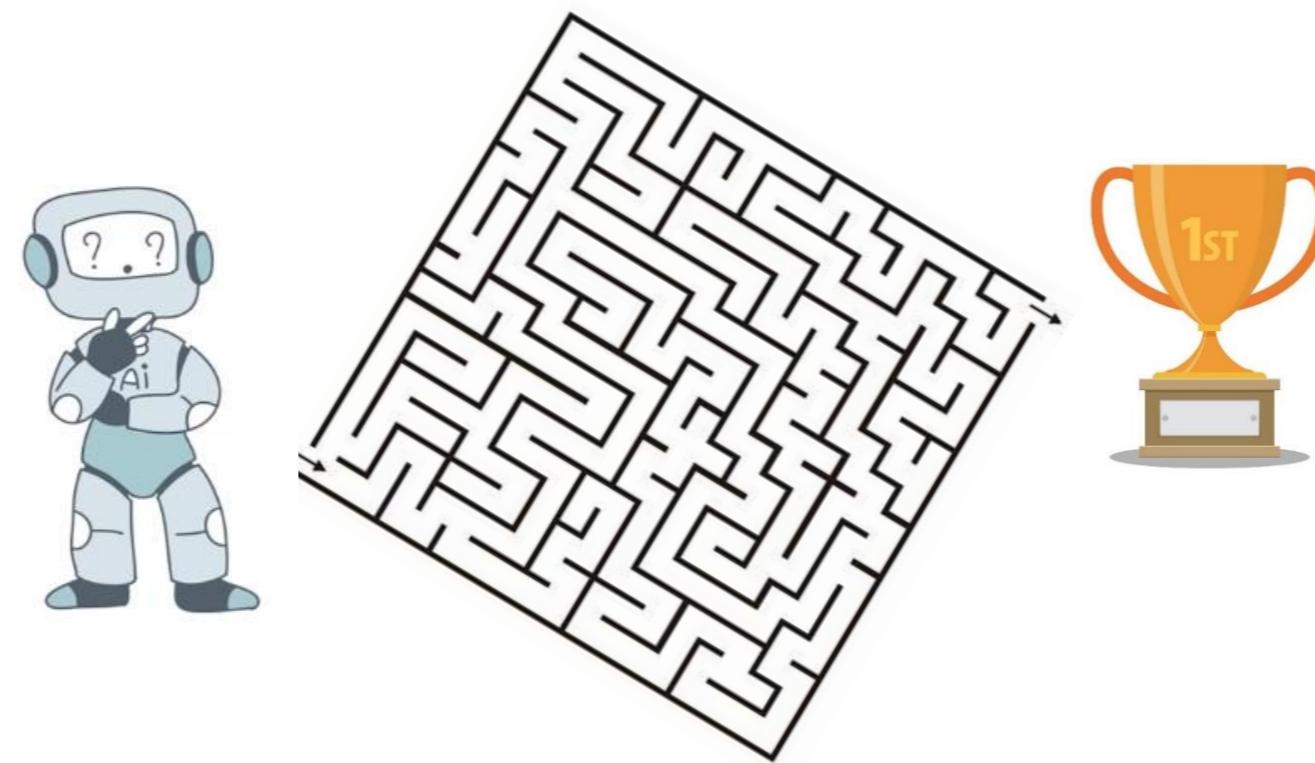
**Unsupervised
Learning**



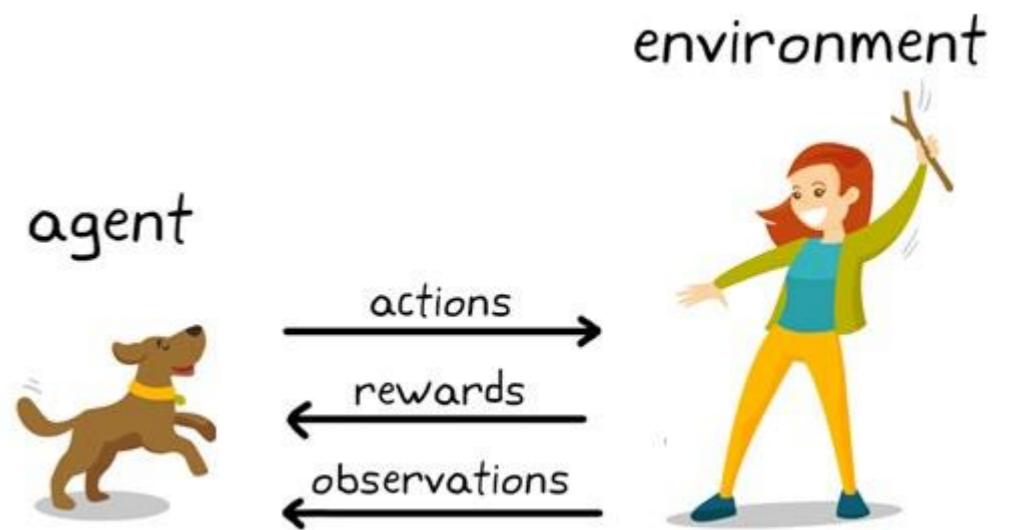
**Reinforcement
Learning**

Reinforcement Learning

- An agent learns to behave in an **environment** by performing **actions** and observing the **results**.



- The agent has no information about the **environment**.
- **No training data**.
- The agent learns from **experience**.
- **Input**: an initial state
- **Training**: based upon the input.
- **Output**: various possibilities
- **Rewards**; upon a rational decision
- Ultimate objective: **maximum reward**.
- Finish it with some outcome either **GOAL** or **FAIL**.



Comparison



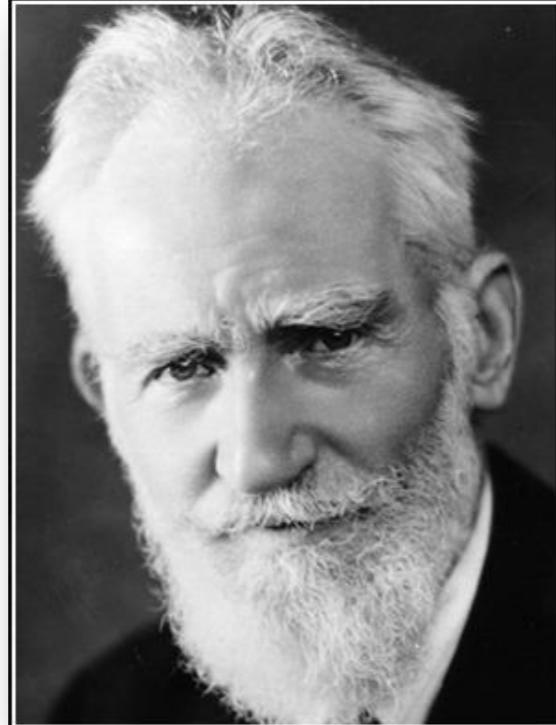
Supervised Learning



Reinforcement Learning



Unsupervised Learning



We learn from experience that men
never learn anything from
experience.

— *George Bernard Shaw* —

- RL Systems:
 - **Agent**
 - **Environment**
- **Agent**
 - Takes some **action** based on the observations.
 - In every step, **updates** its knowledge and decides the **next action** to perform.
 - The loop keep continuing till the agent reaches a **terminal state**, where the agent completes the assigned task.
 - By successfully completing the process, the agent received a **reward**.



Games

1- RL agent collects **state S_0** from the environment.

2- Based on the **state S_0** , the agent takes **action A_0** ,

Initially the action is random

3- The environment is now in a new **state S_1** .

4- The agent now gets a **reward R_1** from the environment.

<https://youtu.be/3BVXynuhNfI>

5- The RL loop goes on until the agent is dead or reaches the destination.

[states, action, rewards]

Definitions

- **Agent:**
 - The RL algorithm that learns from trial and error.
- **Environment:**
 - The world through which the agent moves.
- **Action (A):**
 - All the possible steps that the agent can take.
- **State (S):**
 - Current condition returned by the environment.
- **Reward (R):**
 - An instant return from the environment to appraise the last actions.
- **Policy (π):**
 - The approach that the agent uses to determine the next action based on the current state.



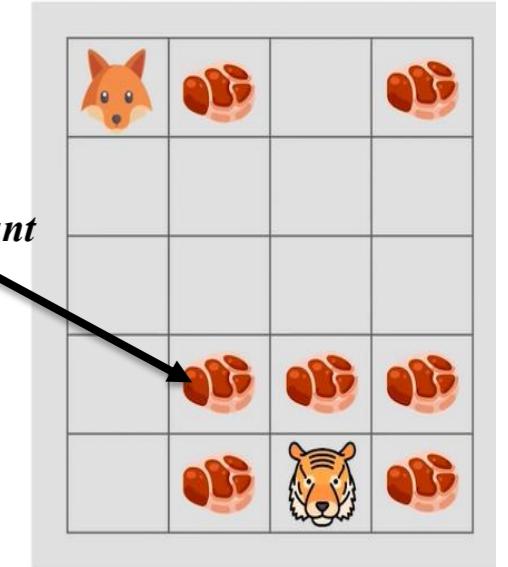
Definitions

- **Value (V):** the expected long-term return with discount,
 - e.g., opposed to the short-term reward R .
- **Discount (γ):** an action that redirect the agent to failure state,
 - Hence must be avoided.
- **Action-value (Q):** like the Value, except, it takes an extra parameter, the current action (A).
- **Exploitation:** using the already known exploited information to heighten the rewards.
- **Exploration:** exploring and capturing more information about an environment.

- **Regret:**
 - The difference between reward of a possible action and reward of the action that has been taken.
 - Penalizes mistakes wherever they occur during the run
 - Minimizing regret requires optimally learning to be optimal



- **Reward maximization theory:**
 - Maximize the rewards by taking the best possible action.
- **Discounting** works based on **Gamma-value γ**
 - The smaller gamma, the larger discount value.
 - **γ closer to 0**: prevent the agent for more exploration in dangerous area.
 - **γ closer to 1**: encourage the agent explore more.



Types of RL

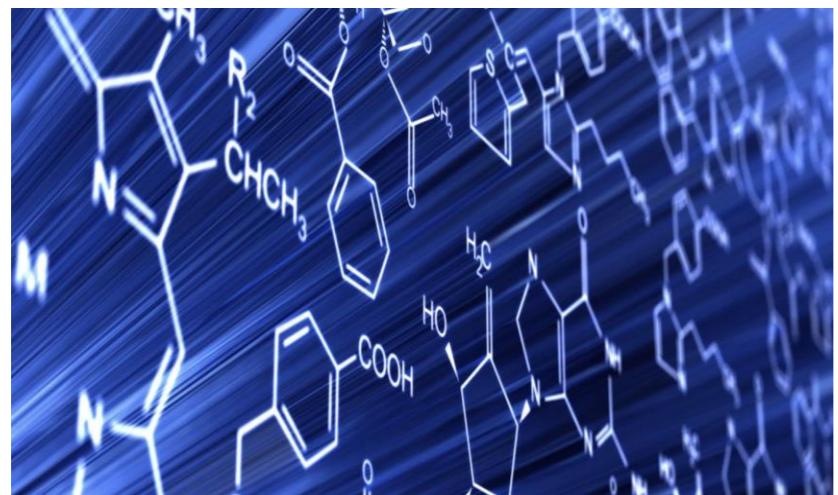
- **Positive:**

- Event occurrence because of **specific behavior**.
- Increases the strength and the frequency of the behavior
- Impacts positively on the action taken by the agent.

- **Negative:**

- Strengthening of a behavior that occurs because of a **negative condition**
 - e.g., should have stopped or avoided.
- Helps to define the minimum stand of performance.

RL Applications



Chemistry; optimization of chemical reactions



Robotics for industrial automation



Video Games



UAV

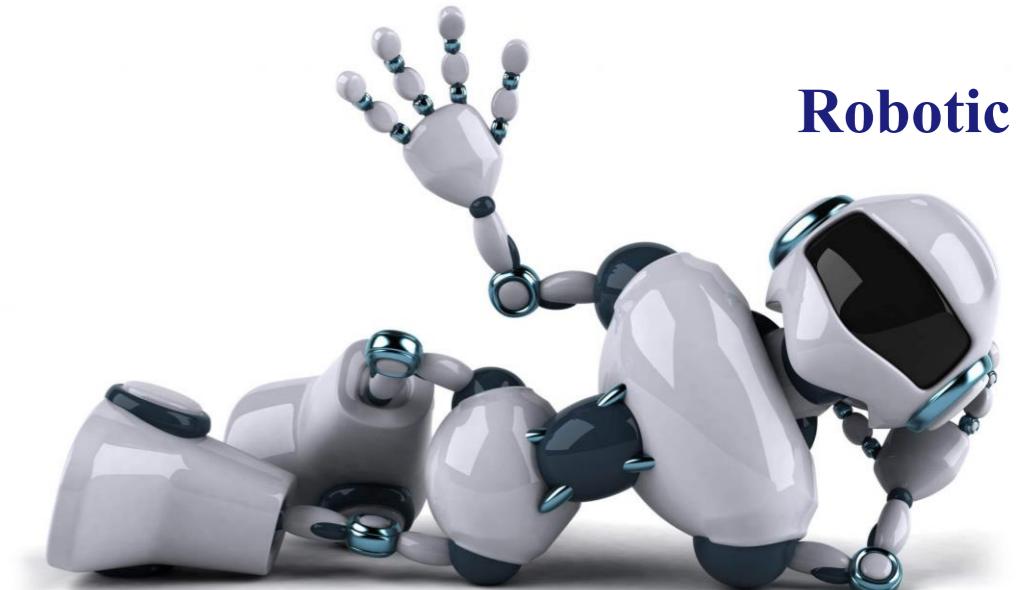
RL Applications



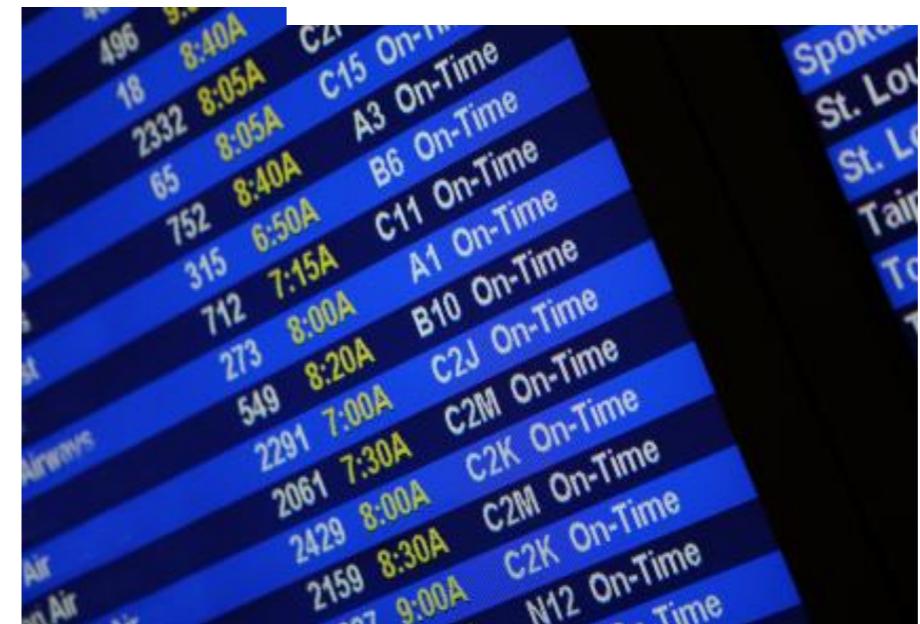
Mail Service



Go Game



Robotic



Flight Schedule

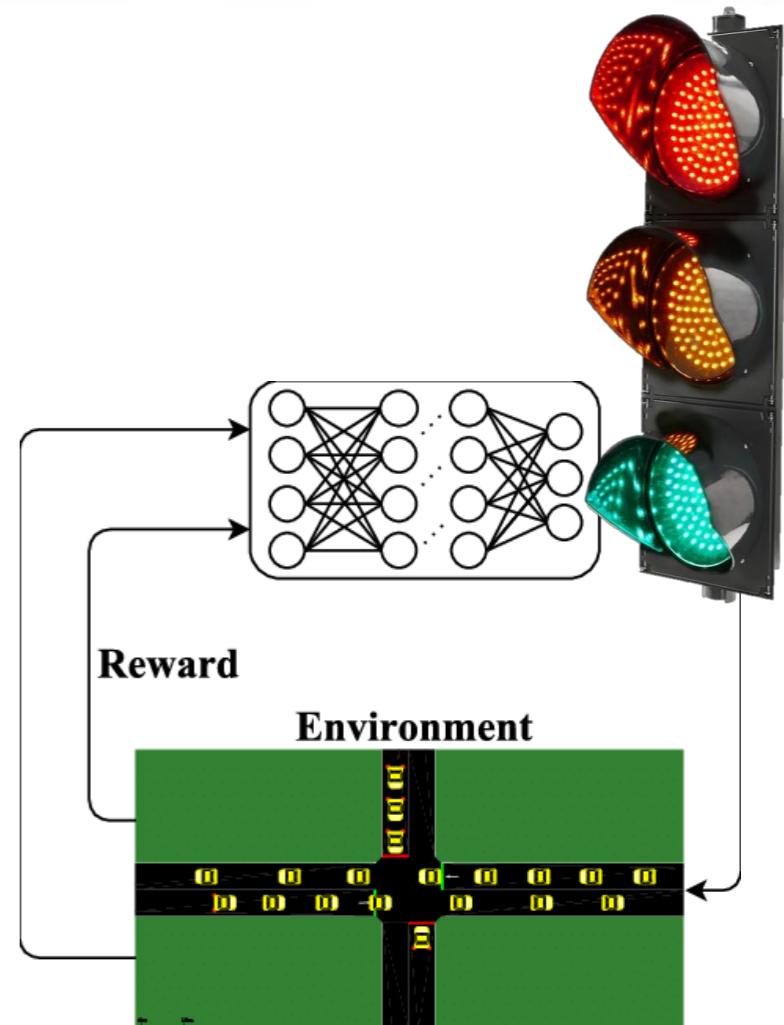
RL Applications



Aircraft Control



Autonomous Car



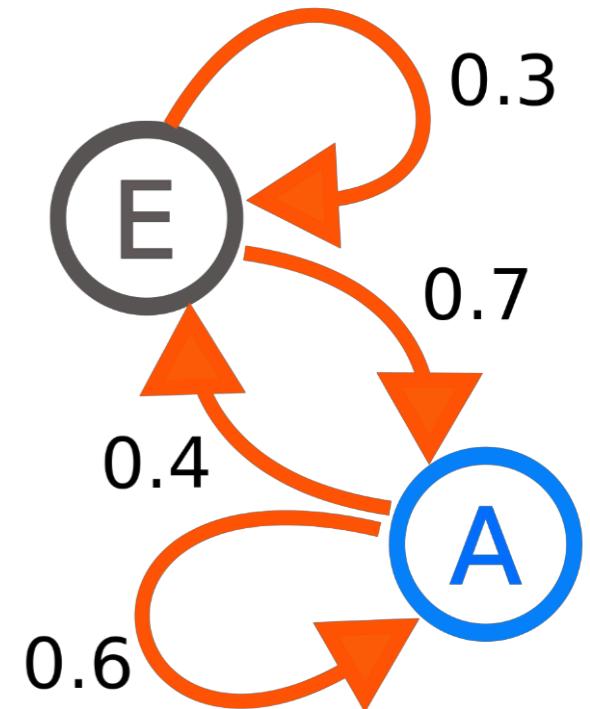
Traffic Light Control

- **Markov decision process (MDP):**
 - The mathematical approach for mapping a solution in RL.
 - A discrete time stochastic control process.
 - Employed for non-deterministic search problems.
 - Useful for studying optimization problems solved via dynamic programming and RL.
 - **Markov property:** “future is independent of the past give the present”.

- **Andrey Andreyevich Markov**
- 1856–1922
- Russian mathematician
- St. Petersburg University
- Best known for his work on:
 - Stochastic processes.
 - Markov chains
 - Markov processes



- **Markov chain:**
 - A stochastic model.
 - A mathematical system that experiences **transitions** from one **state** to another according to certain probabilistic **rules**.
 - The probability of each event depends only on the state attained in the previous event.



- MDP consist of:
 - **Stochastic process**
 - **Decision maker:** observes the process and select actions that influence its development over time.
 - The decision maker receives a series of **rewards:**
- **Positive rewards**
- **Negative rewards**

- **Markov property:**
 - **Outcome** of an action depend only on **current state**
 - **Independent** from the **future** and the **past states**.

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, \dots, S_0 = s_0)$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

Formulation

$$\left(\begin{array}{l} \mathbf{T}, S, A_s, p_t(\cdot | s, a), r_t(s, a) : \\ \quad t \in \mathbf{T}, s \in S, a \in A_s \end{array} \right)$$

- $T \in [0, \infty)$: **set of decision epochs**
 - Points in time when the decision maker decides on and then executes an action
 - If T is **discrete** (finite or countably infinite) set of stages
 - If T is **finite**, assume that no decision is taken in the final decision epoch.
- **Finite horizon**; T is finite, $T = \{1, 2, 3, \dots, N\}$
- **Infinite horizon**; T is infinite, $T = \{1, 2, 3, \dots\}$.

Formulation

$$\left(\begin{array}{c} T, \mathcal{S}, \mathcal{A}_s, p_t(\cdot | s, a), r_t(s, a) : \\ t \in T, s \in \mathcal{S}, a \in \mathcal{A}_s \end{array} \right)$$

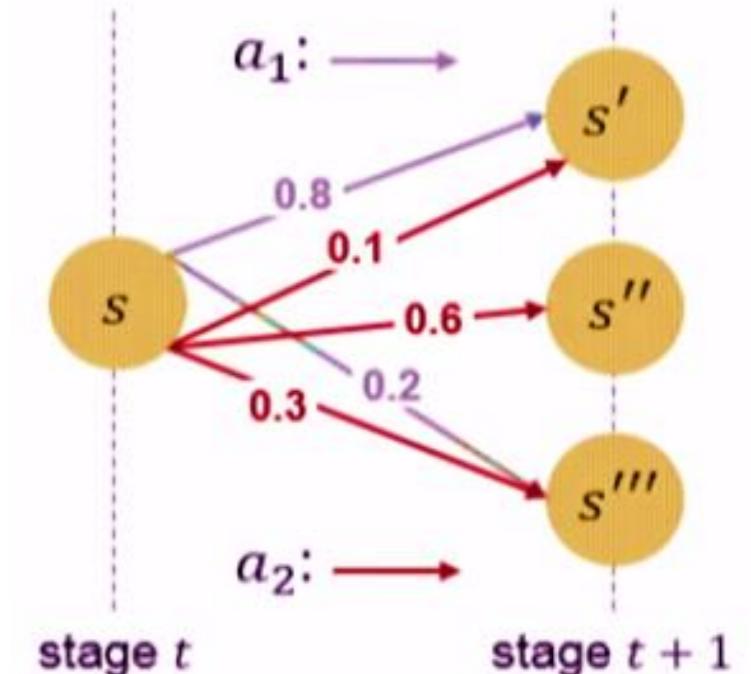
- **\mathcal{S} : state space**
 - The processes take values incountable S .
- **\mathcal{A}_s : action space**
 - Set of possible actions the state of the system is $s \in S$
 - $A = \bigcup_{s \in S} \mathcal{A}_s$: set of all possible actions

Formulation

$$\left(T, S, A_s, \mathbf{p}_t(\cdot | s, a), \mathbf{r}_t(s, a) : \right)$$

$$t \in T, s \in S, a \in A_s$$

- $\mathbf{p}_t(\cdot | s, a)$: **transition probability**
 - Specify how the state of the system changes from one decision epoch to the next (give that T is discrete)
- $\mathbf{r}_t(s, a)$: **rewards**
 - The decision maker receives a reward $r_t(s, a)$ (either profit or cost) as a result of choosing action ' a ' when the system is in the state ' s ' at time ' t '.
 - $r_N(s)$: terminal rewards in case that a MDP has finite horizon N



- **Decision rule δ_t**

- Tell the decision maker how to choose the action to be taken in a given decision epoch $t \in T$.
- For a Markovian and deterministic decision rule, $\delta_t: S \rightarrow A$

S	A
1	Up
2	Left
3	Right
4	Up
5	Up

Decision rule example

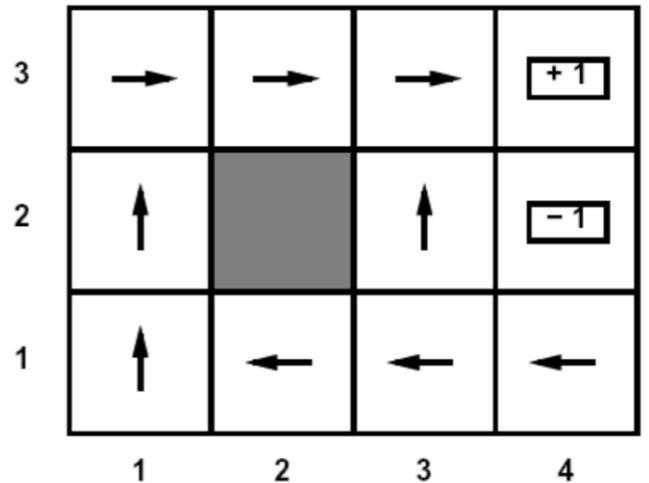
- **Policy π**

- A sequence of decision rule $\delta_1, \delta_2, \delta_3, \dots$ for every decision epoch.
- **Stationary π :** if the same decision rule is used in every decision epoch.

The **goal** is to find a **policy π** that **maximizes** the expected future reward.

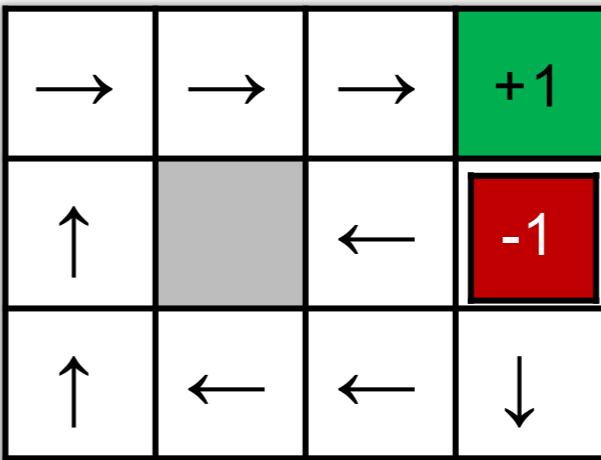
Policies

- **Optimal plan:** sequence of actions, from start to a goal.
- For MDPs (non-deterministic), we focus on **policies**
 - Policy: map of states to actions
 - $\pi(s)$ gives an action for state s
- **Goal:** an **optimal policy** $\pi^*: S \rightarrow A$
 - e.g., it Maximizes the expected utility if followed.

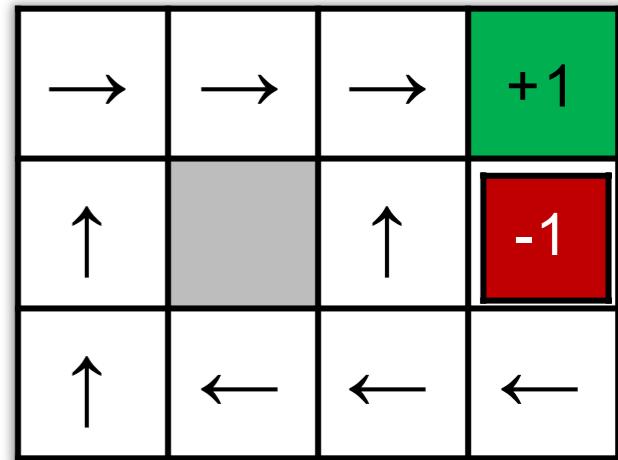


Optimal Policies

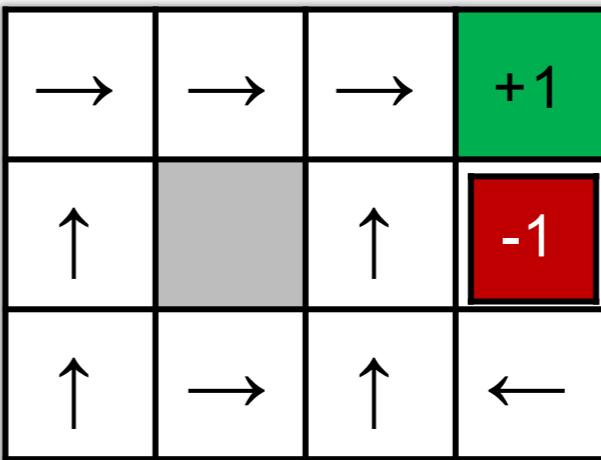
- Which sequence of optimal policies matches?
- $R(s)$: each step cost in different policies
 - If cost is less, select a longer way.
 - If the cost is high, take the shortest path.



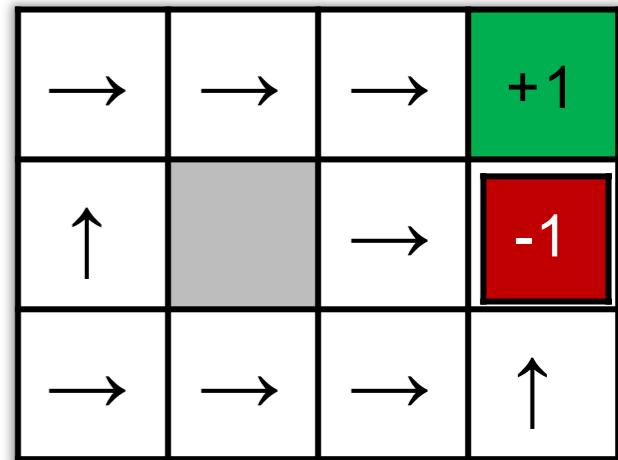
$$R(s) = -0.01$$



$$R(s) = -0.03$$



$$R(s) = -0.4$$



$$R(s) = -2.0$$

- **Value function $v_t(s_t)$**
 - Maximum total expected reward starting in state s_t from ‘ t ’ decision epoch onward.
- **Bellman optimality equation:** to find the value of one state



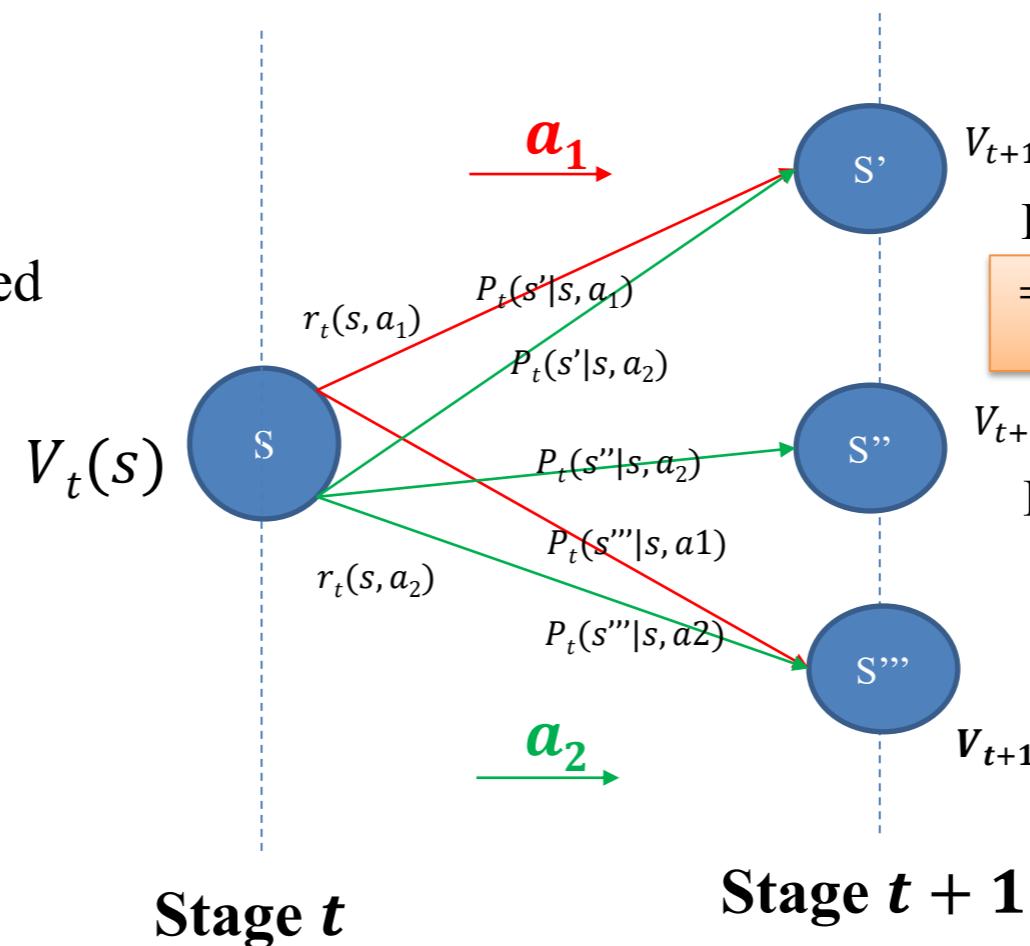
Richard Ernest Bellman
(1920~1984)

$$V_t(s_t) = \max_{a_t \in A_{s_t}} \{ r_t(s_t, a_t) + E[v_{t+1}(s_{t+1})] \}$$

Find an action a_t that maximizes $\{$
 Expected immediate reward in period t
 Expected maximum total remaining reward in period $t + 1, t + 2, \dots$

$$V_t(s_t) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p(j|s_t, a_t) v_{t+1}(j) \right\}$$

- Objective:** to select action a_1 or action a_2 that maximizes the expected reward.



Maximum total expected reward in this state

If action a_1 :

$$= r_t(s, a_1) + [v_{t+1}(s') * P_t(s'|s, a_1) + v_{t+1}(s'') * P_t(s''|s, a_1) + v_{t+1}(s''') * P_t(s'''|s, a_1)]$$

$V_{t+1}(s')$

If action a_2 :

$$= r_t(s, a_2) + v_{t+1}(s') * P_t(s'|s, a_2) + v_{t+1}(s'') * P_t(s''|s, a_2) + v_{t+1}(s''') * P_t(s'''|s, a_2)$$

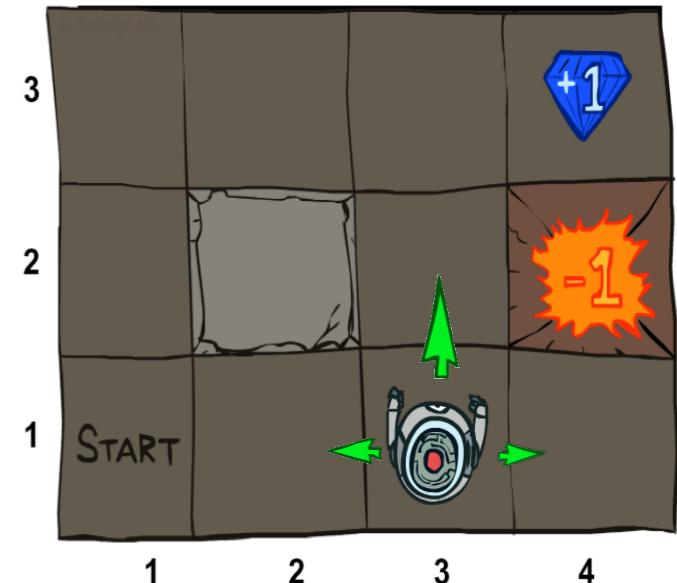
$V_{t+1}(s'')$

$V_{t+1}(s''')$

The action with the biggest value will be selected.

Example: Grid World

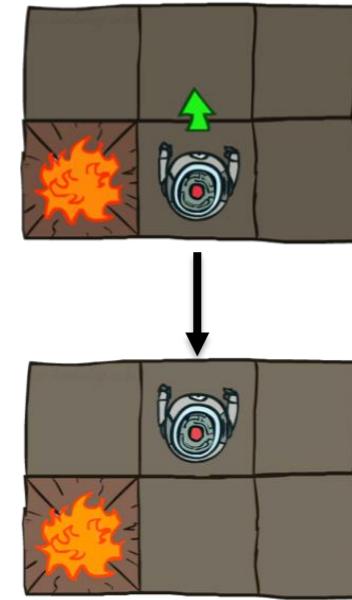
- **A maze-like problem**
 - The agent lives in a grid.
 - Walls block the agent's path.
- **Noisy movement:** actions do not always go as planned
 - In the current position, if agent takes action North:
 - Assumption:
 - 80% of the time if the agent select ‘North’ it will go one cell up.
(if there is no wall there)
 - 10%: West; 10%: East
 - These conditions are known as **non-deterministic** movement.
 - The agent receives rewards each time step:
 - “Living” reward (can be negative); cost of each step
 - Additional reward at pit or target (good or bad) and will exit the grid world afterward
 - **Goal:** maximize sum of rewards



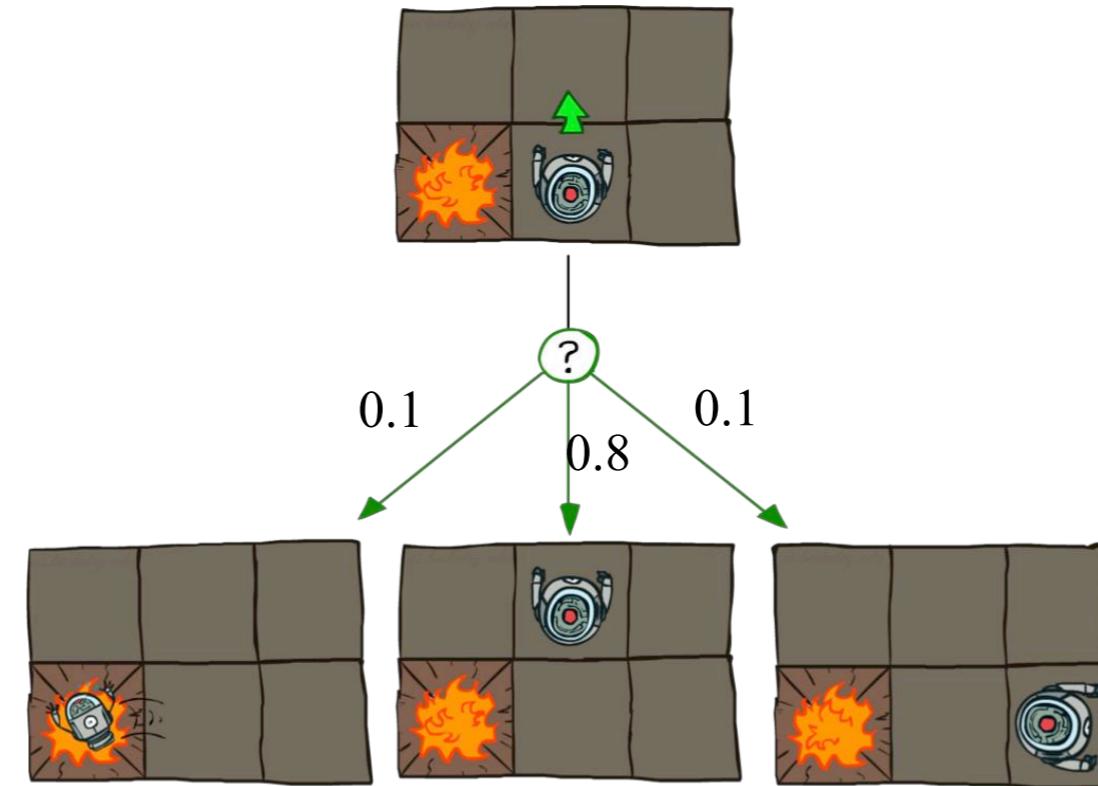
MDP

- A Grid can be designed in two environment:

Deterministic Grid World



Non-deterministic (Stochastic) Grid World



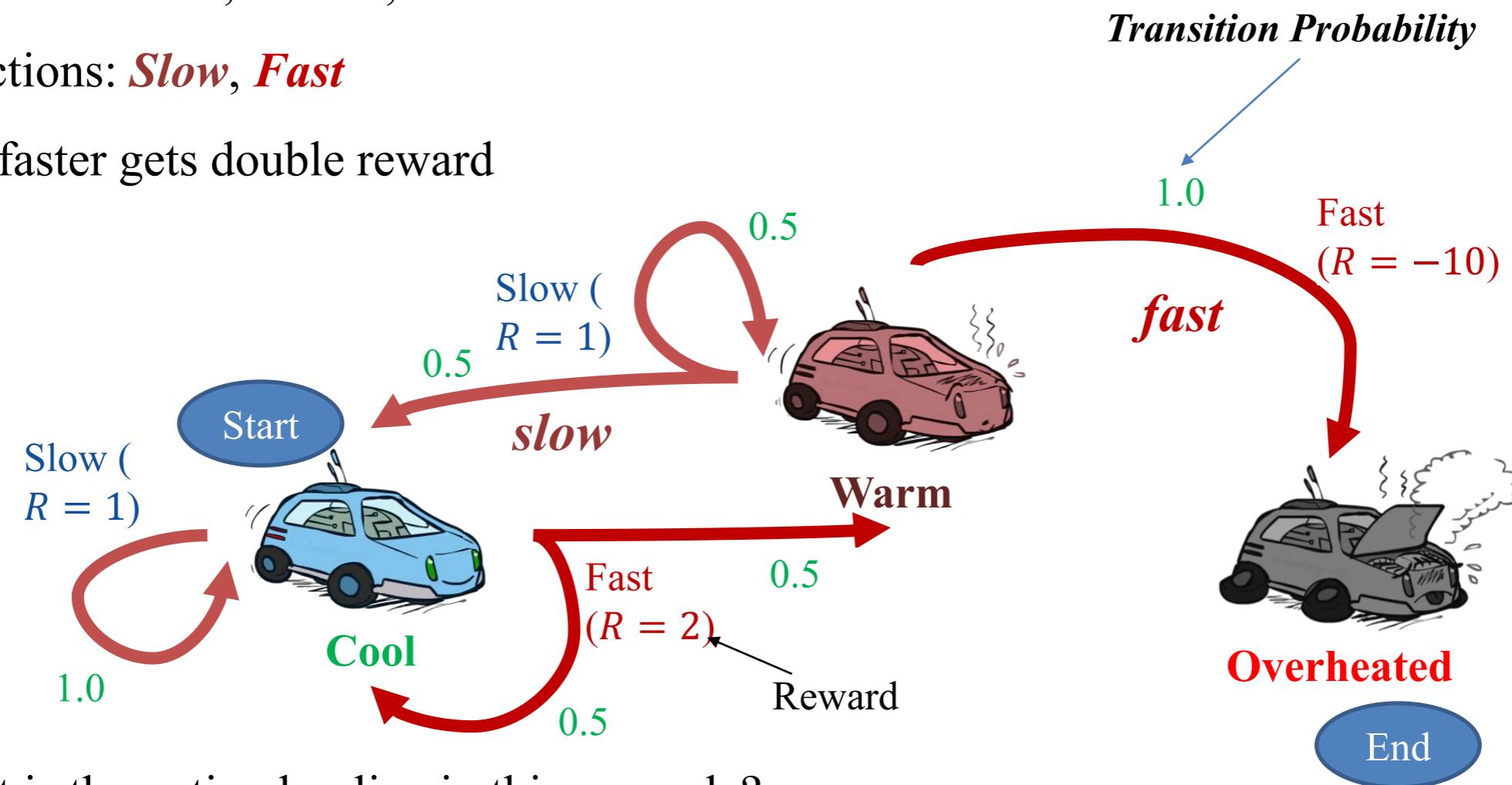
The agent knows what action to take!

The agent does not know what action to take!
Takes an action and see the result.

MDP is used for non-deterministic search problems!

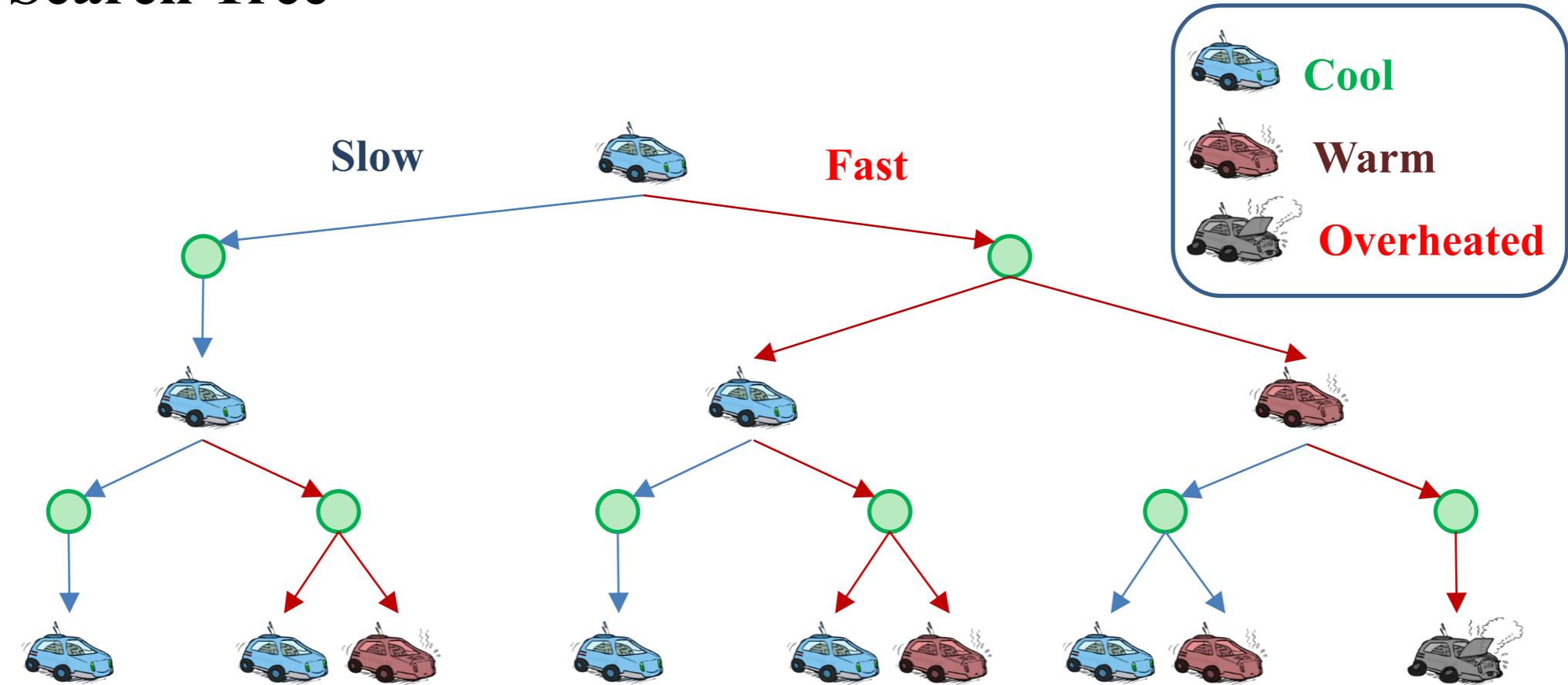
Example: racing

- A robot car wants to travel far and quickly.
- Three states: **Cool**, **Warm**, **Overheated**
- Two actions: **Slow**, **Fast**
- Going faster gets double reward



- What is the optimal policy in this example?

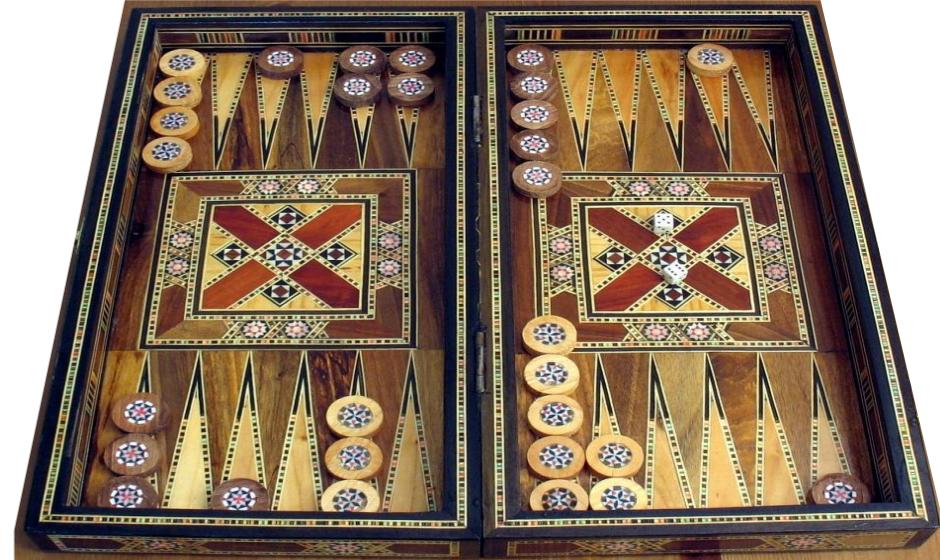
Racing Search Tree



- Can we use **expectimax** for MDP directly?

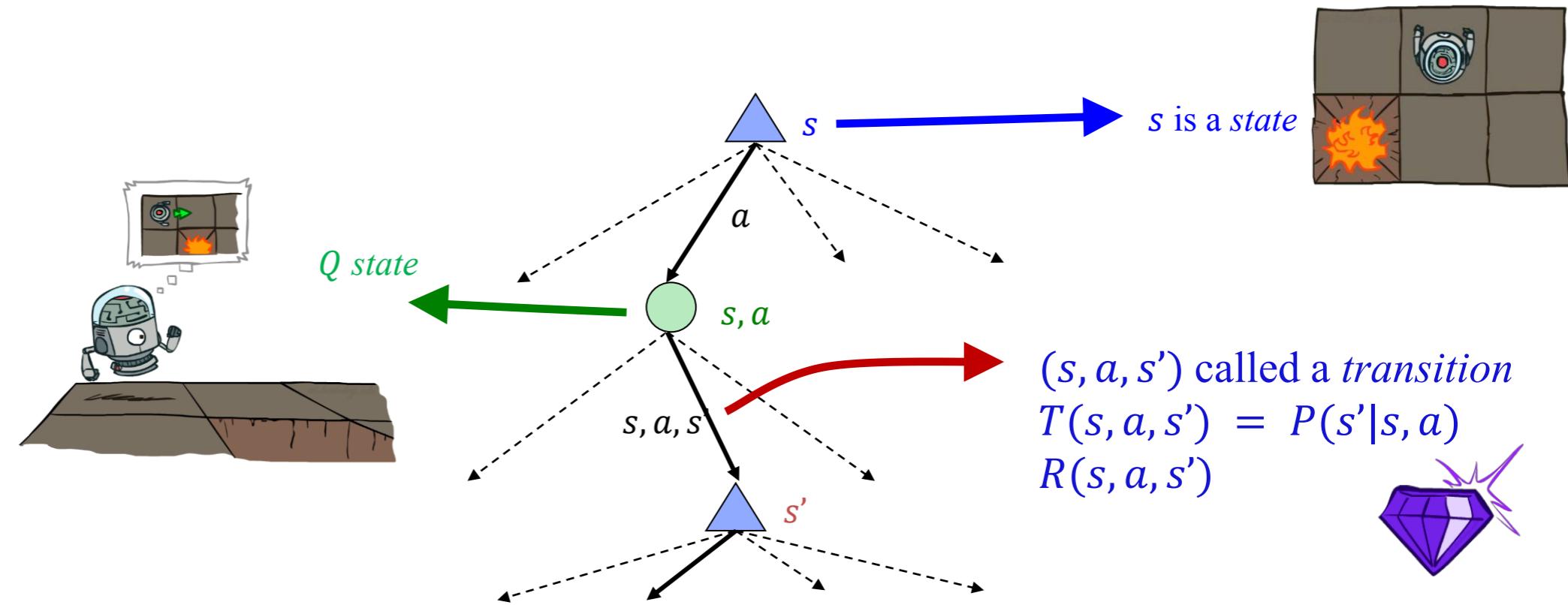
Expectimax Algorithm

- Used to solve MDPs for non-deterministic search problem.
- Maximizes the average (expected) reward.
- There is a probabilistic model of how the opponent (or environment) will behave in any state.
- Not 100% safe, you might lose.
- **Expectimax:** what to do in the current state!
- **Policy** gives a plan for all states.



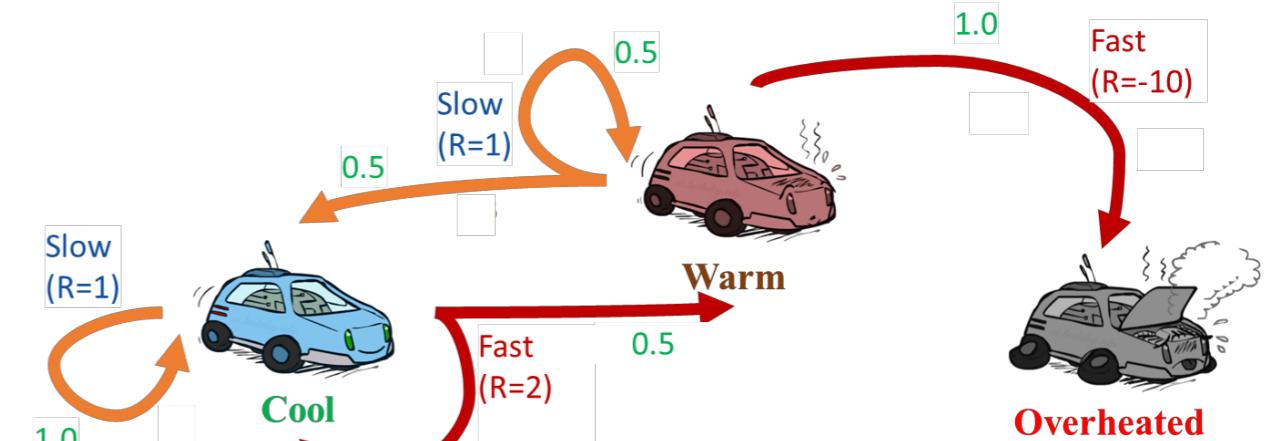
Search Tree; General form

- MDP state: projects an Expectimax-like search tree.
- Several choices are available in each state.



So far...

- **An MDP:**
 - A set of states $s \in S$
 - A set of actions (per state) A
 - A model $T(s, a, s')$
 - A reward function $R(s, a, s')$
- **Objective:** find a policy $\pi(s)$.
- **New twist:** don't know T or R , so must try out actions!
- **Big idea:** Compute all averages over T using sample outcomes!



Types

- **Offline mode (MDP):**

- If MDP is known,
- e.g., T and R are known

- **Online learning (RL):**

- If MDP is Unknown,
- e.g., T and R are unknown

Reinforcement Learning with Online Interactions



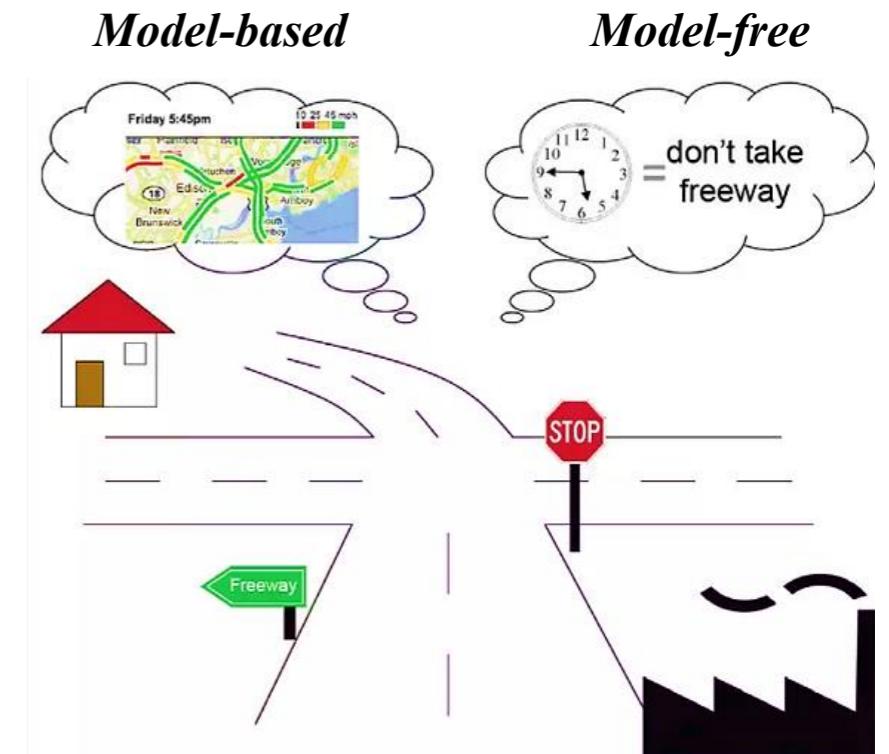
Offline Reinforcement Learning



Learning Modes



- **Model-based:** build a model of the environment
 - Learn an approximate model based on experience
 - Memory-intensive learning method
- **Model-free:** learn a policy without any model
 - Temporal difference methods (TD)
 - Requires limited episodic memory (though more helps)



Model-Based Learning

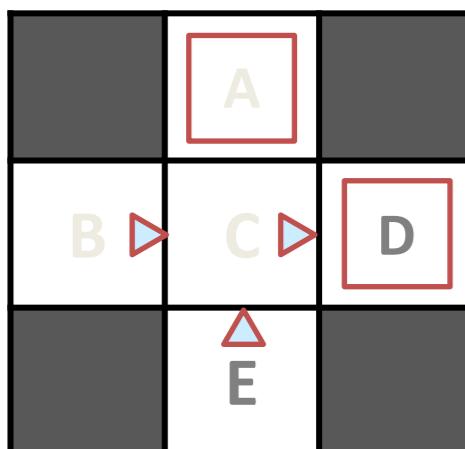
- Learn an approximate model based on experiences
- Solve for values as if the learned model were correct
- **Procedure:**
 - Step 1: **Learn empirical MDP model**
 - Count outcomes s' for each s, a
 - Find estimation of \hat{T} and \hat{R}
 - Step 2: **Solve the learned MDP**
 - e.g., value iteration

Learning Models

Model-Based Learning

Example:

Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Estimated model

$T(s, a, s')$

$T(B, \text{east}, C) = 1.00$
 $T(C, \text{east}, D) = 0.75$
 $T(C, \text{east}, A) = 0.25$
...

$R(s, a, s')$

$R(B, \text{east}, C) = -1$
 $R(C, \text{east}, D) = -1$
 $R(D, \text{exit}, x) = +10$
...

Learning Models

Example

- **Goal:** the expected age of students in a class.

- If $P(A)$ is given:

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \dots$$

- If $P(A)$ is unknown: then **sampling** is a solution.

- Sample: $[a_1, a_2, \dots, a_N]$

Model-based

$$\hat{P}(a) = \frac{\# \text{ of students with age } (a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Model-Free

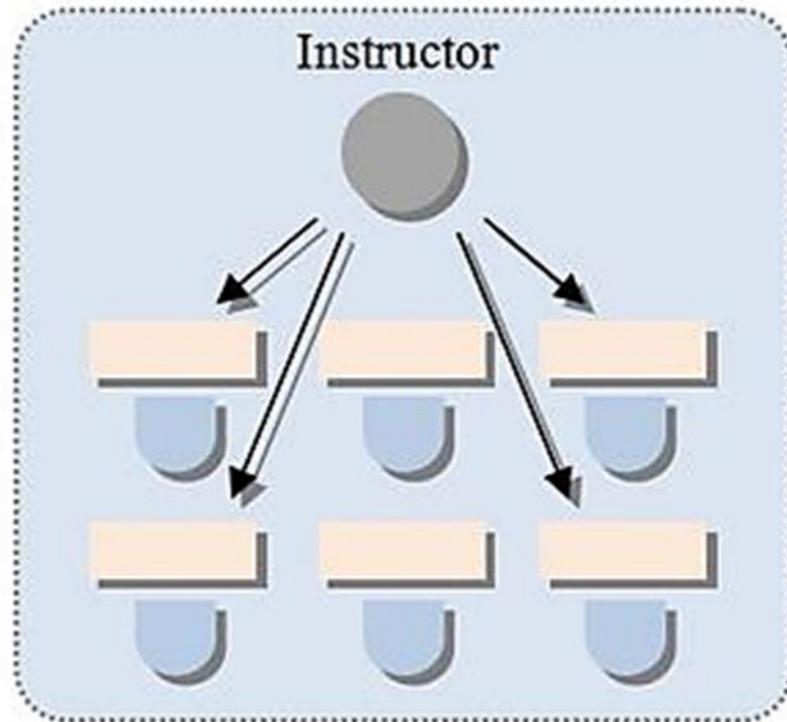
$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Learning Models

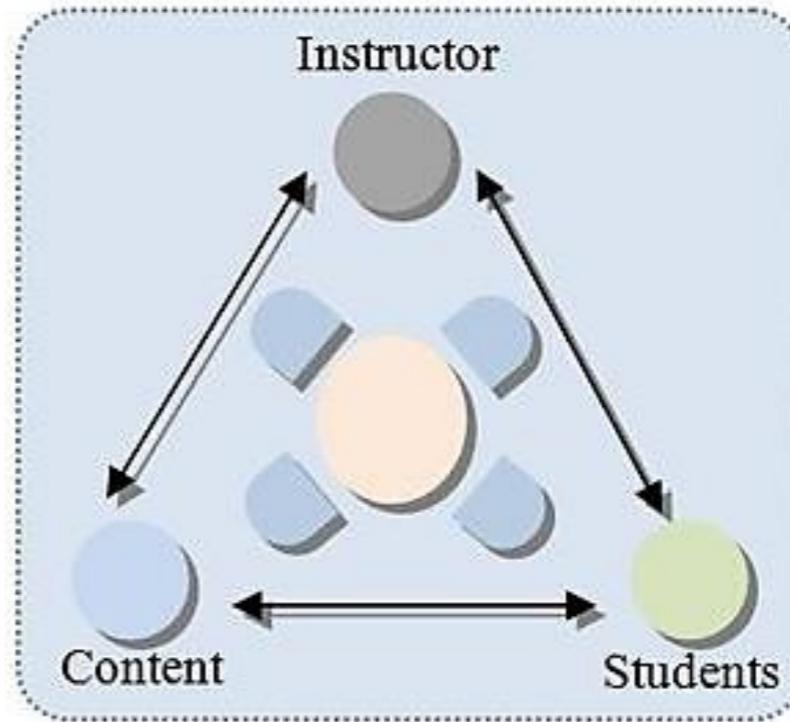
Model-free Learning

- Types:

- Passive



- Active



Passive RL

- Simplified task: policy evaluation
 - **Input:** a fixed policy $\pi(s)$
 - Unknown **transitions** $T(s, a, s')$
 - Unknown **rewards** $R(s, a, s')$
 - **Goal:** learn the state values
- In this case:
 - Learner is “along for the ride”
 - No choice about what actions to take
 - Just execute the policy and learn from experience
 - This is NOT offline planning! You actually take actions in the world.

Passive RL

Direct Evaluation

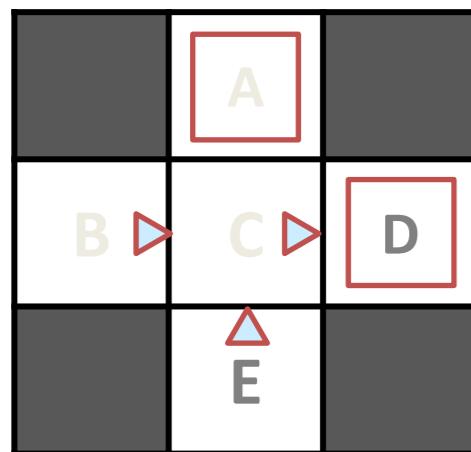
- **Goal:** Compute values for each state under π
- **Idea:** Average together observed sample values
 - Act according to π
 - Every time visit a state, write down what the sum of discounted rewards turned out to be the Average those samples

Learning Models

Example: Direct Evaluation

- The policy is given to the agent and must follow.

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

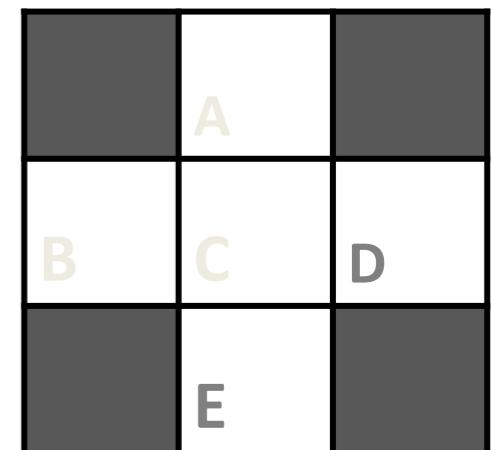
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

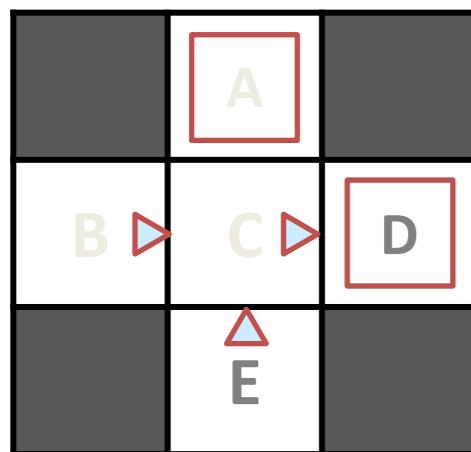
Output Values



Learning Models

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

- B, east, C, -1
- C, east, D, -1
- D, exit, x, +10

Episode 2

- B, east, C, -1
- C, east, D, -1
- D, exit, x, +10

Episode 3

- E, north, C, -1
- C, east, D, -1
- D, exit, x, +10

Episode 4

- E, north, C, -1
- C, east, A, -1
- A, exit, x, -10

Output Values

	-10	
A	+8	+4
B	C	D

	-2	
E		

Direct Evaluation

- **Advantages**
 - Easy to understand
 - Doesn't require any knowledge of T , R
 - Eventually computes the correct average values, using just sample transitions

Output Values

	-10 A	
+8 B	+4 C	+10 D
		-2 E

- **Disadvantages**
 - Wastes information about state connections
 - Each state must be learned separately
 - High time complexity!

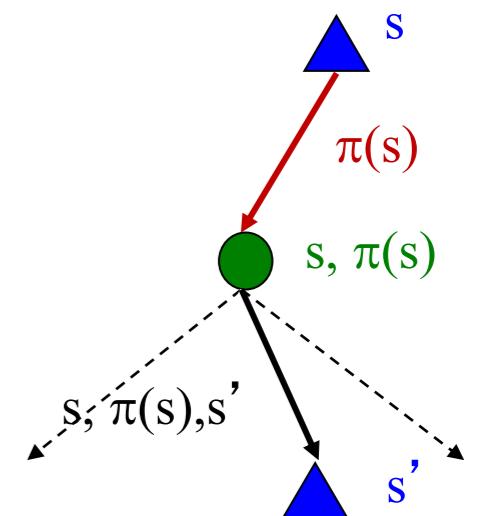
Learning Models

Policy Evaluation

- Simplified Bellman updates calculate V for a fixed policy π :
 - Each round, replace V with a one-step-look-ahead layer over V

$$V_0^\pi(s) = 0, \forall s$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')], \forall s$$



Policy Evaluation

- **Goal:** improve the V estimation via:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')], \forall s$$

- **First idea:** Take samples of outcomes s' (by taking the action!) and average

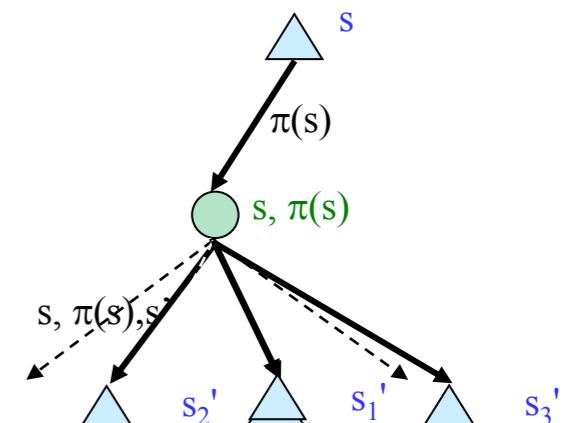
$$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

...

$$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$



Policy Evaluation

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')], \forall s$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

- In the extreme case, we can just take one sample ($n = 1$)

$$V_{k+1}^{\pi}(s) \leftarrow sample$$

- But this is very high variance!
- Second idea:** Make use of the value of $V_k^{\pi}(s)$. Use running average.

$$V_{k+1}^{\pi}(s) \leftarrow (1 - \alpha)V_k^{\pi}(s) + \alpha \times sample$$

Exponential Moving Average

- Exponential moving average
 - The running interpolation update: $\bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
 - Makes recent samples more important:

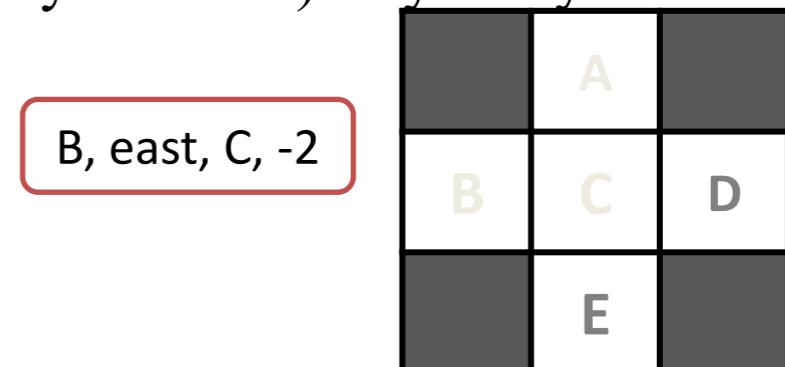
$$\bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \dots}{1 + (1 - \alpha) + (1 - \alpha)^2}$$

- Decreasing learning rate (α) can give converging averages
 - $\alpha = 0$: the agent learn nothing (exclusively exploiting prior knowledge),
 - $\alpha = 1$: the agent consider only the most recent information

Learning Models

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')], \forall s$$

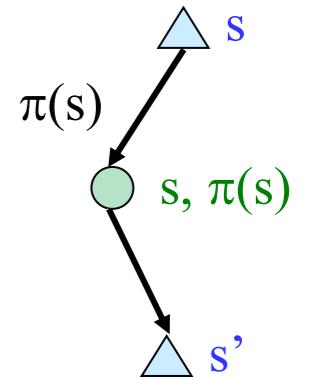
- You only have access to a stochastic environment.
- You cannot fully control which state you will be at and directly jump to each of the states one by one to update V^{π} .
- **Third idea:** Only update one state s at a time (the state you are in) as you try out the policy in the environment



sample= $R(s, \pi(s), s') + \gamma V_k^{\pi}(s') = -2 + 1 * V^{\pi}(C) = -2$

Temporal Difference (Value) Learning

- Putting the ideas together: Temporal Difference (Value) Learning!
- Task: Given policy π , learn state value V^π
- Learn from every experience
 - Update $V^\pi(s)$ each time we experience a transition (s, a, s', r)
 - Likely outcomes s' will contribute updates more often
 - Move values toward latest sample (running average)
- Sample of $V(s)$:** $sample = R(s, \pi(s), s') + \gamma V^\pi(s')$
- Update to $V(s)$:** $V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$
- Equivalent to:** $V^\pi(s) \leftarrow V^\pi(s) + \alpha(sample - V^\pi(s))$



Temporal Difference (Value) Learning

- States

	A	
B	C	D
	E	

$$\gamma = 1, \alpha = 1/2$$

- Observed state changes

	0	
0	0	8
	0	

	0	
-1	0	8
	0	

	0	
0	3	0
	0	

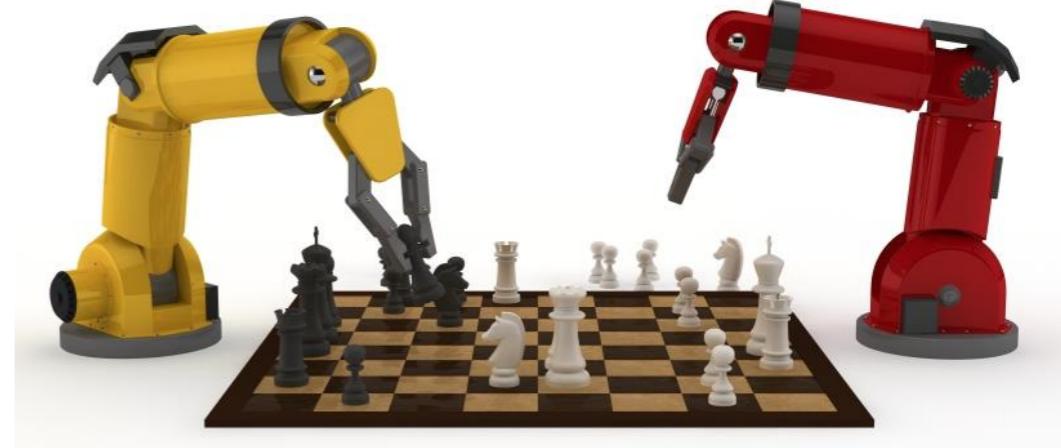
$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)[R(s, \pi(s), s') + \gamma V^\pi(s')]$$

Active RL

- **Full RL:** optimal policies (like value iteration)
 - The agent select can select any action
 - Unknown transitions $T(s, a, s')$
 - Unknown rewards $R(s, a, s')$
 - Take an actions at a time step (no given policy π)
- **Goal:** learn the optimal policy / values
- In this case:
 - Learner makes choices!
 - Main challenge: **exploration and exploitation!**

Active RL

- ‘Q’: stands for **quality**.
- **Model-free**
- **value-based**
- **Off-policy**
- **Goal:** to learn a policy,
 - To tell an agent what action to take under what circumstances.



Q-Learning Algorithm

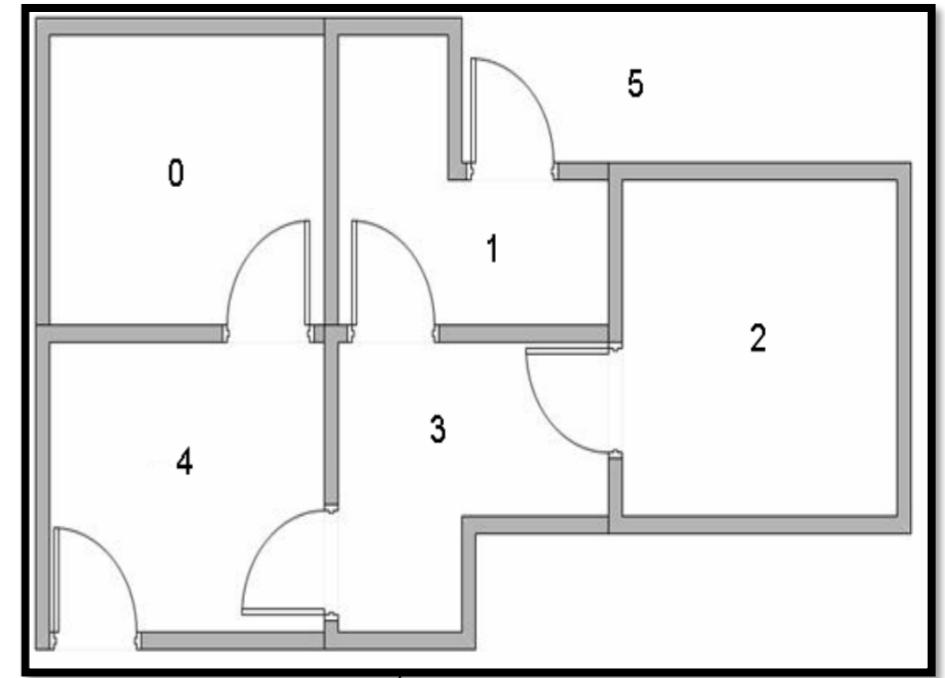
1. Set the γ , and environment **rewards** in matrix R.
2. Initialize **matrix Q** to zero
3. Select a random **initial state**
4. Set initial state = **current state**
5. Select one among all possible actions for the **current state**
6. Using this possible **action**, consider going to the next state
7. Get maximum **Q value** for this next state based on all possible actions

*Compute: $Q(state, action) = R(state, action) + \text{Gamma} * \text{Max } [Q(next\ state, all\ actions)]$*

8. Repeat above steps until current state = goal state

Example

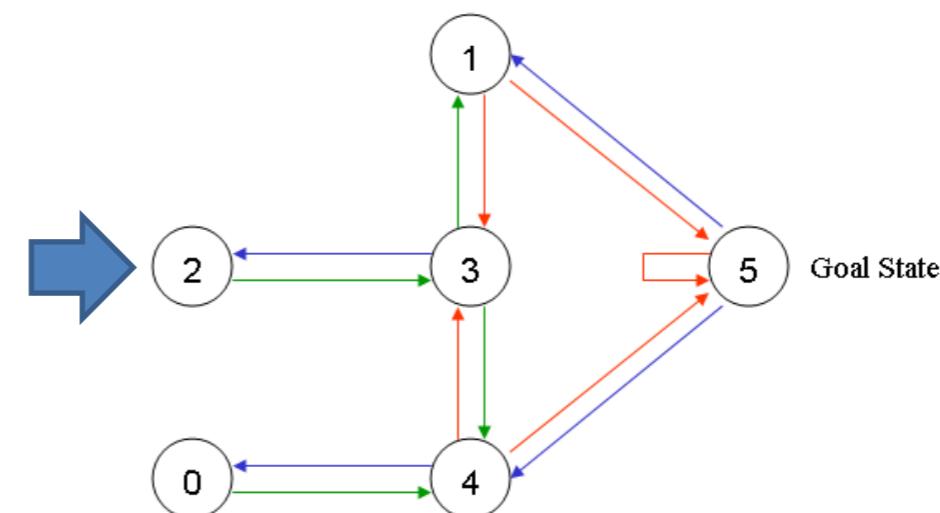
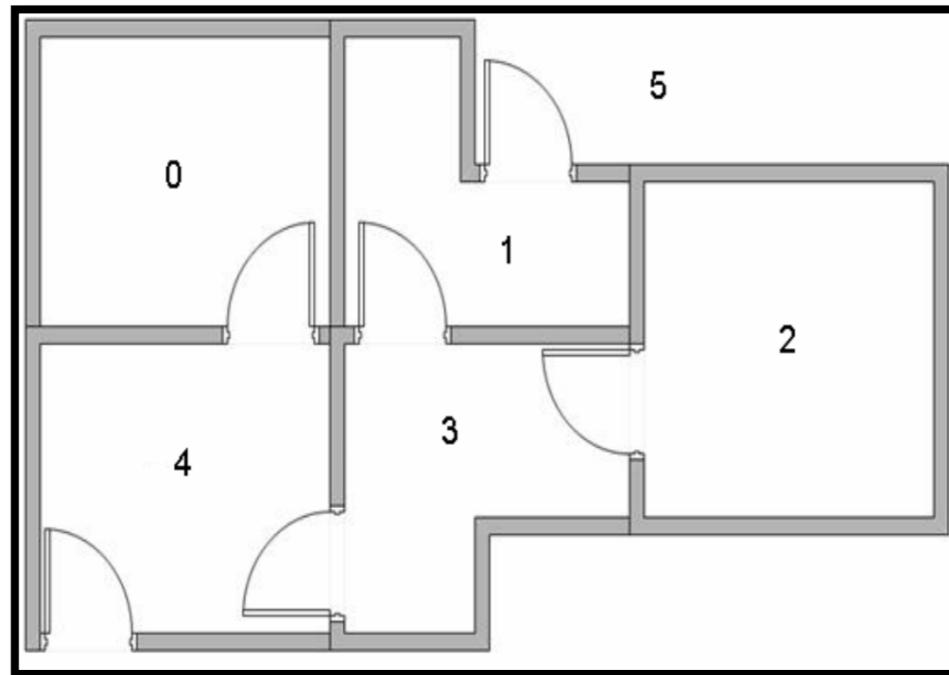
- Initial state: any one of the rooms (0,1,2,3,4)
- Goal: to reach outside the building room (room #5)



Q-Learning

Example

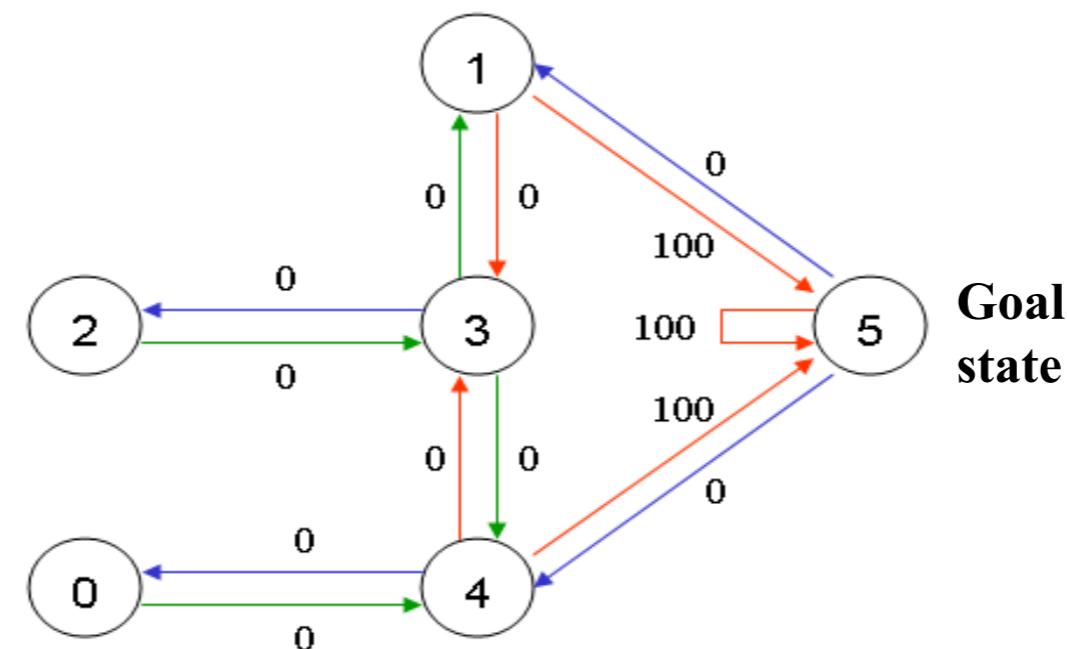
- Graph representation



Example

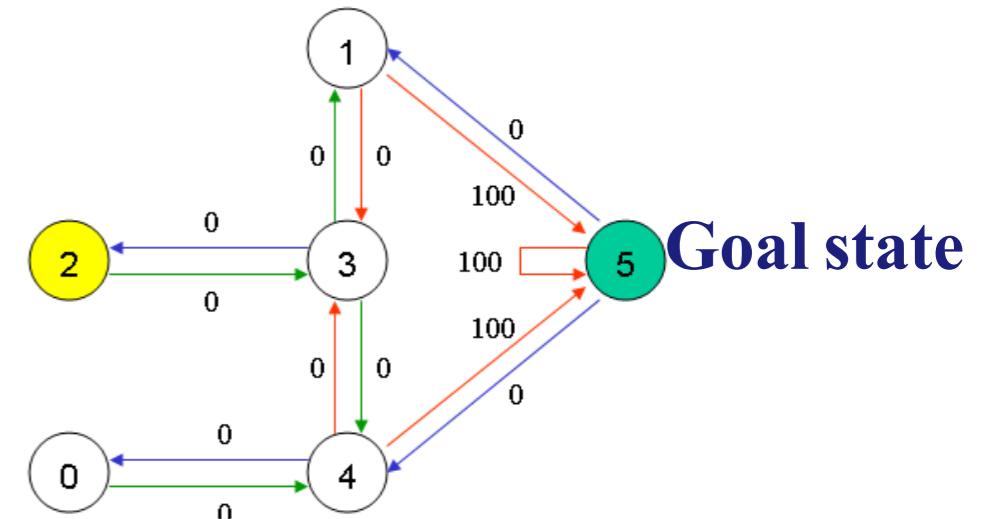
- **Rewards**

- **100**: for the doors that lead directly to the goal (outside)
- **0**: for the doors not directly connected to the target room



Q-learning

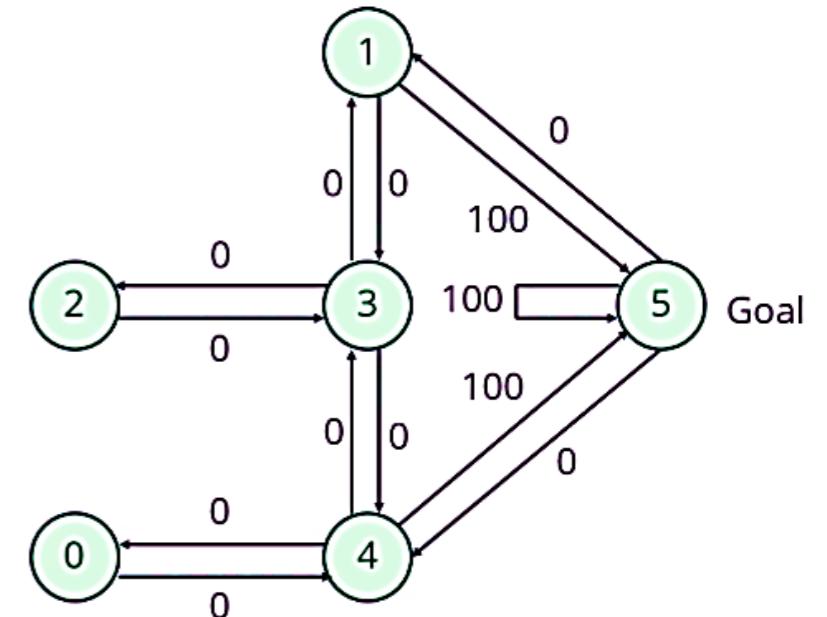
- **State:** the rooms
- **Action:** Agent's movement from one room to another room
- Example) agent traverse from room 2 to 5
 - **initial state:** state 2
 - State 2 -> state 3
 - State 3 -> state (2, 1, 4)
 - State 4 -> state 5 (**Goal**)



Q-learning example

- Matrix R:
- The state diagram and the instant reward values

$$\begin{matrix}
 & \textbf{0} & \textbf{1} & \textbf{2} & \textbf{3} & \textbf{4} & \textbf{5} \\
 \textbf{0} & -1 & -1 & -1 & -1 & 0 & -1 \\
 \textbf{1} & -1 & -1 & -1 & 0 & -1 & 100 \\
 \textbf{2} & -1 & -1 & -1 & 0 & -1 & -1 \\
 \textbf{3} & -1 & 0 & 0 & -1 & 0 & -1 \\
 \textbf{4} & 0 & -1 & -1 & 0 & -1 & 100 \\
 \textbf{5} & -1 & 0 & -1 & -1 & 0 & 100
 \end{matrix}$$



Q-learning example

- **Matrix Q:**
 - The memory of what the agent has learnt through experience
- Note:
 - γ closed to zero: the agent will tend to consider only immediate rewards
 - γ is close to one: the agent will consider future rewards with greater weight

- **First step:**
 - Set the value of the learning parameter, $\gamma = 0.8$
 - Set the initial state as Room 1
 - Initialize matrix Q as a zero matrix:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- **Second step:**
 - From room 1 the agent can go either to room 3 or 5,
 - If goes to 5
 - From 5, calculate maxim Q value for this state based on all possible actions

	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

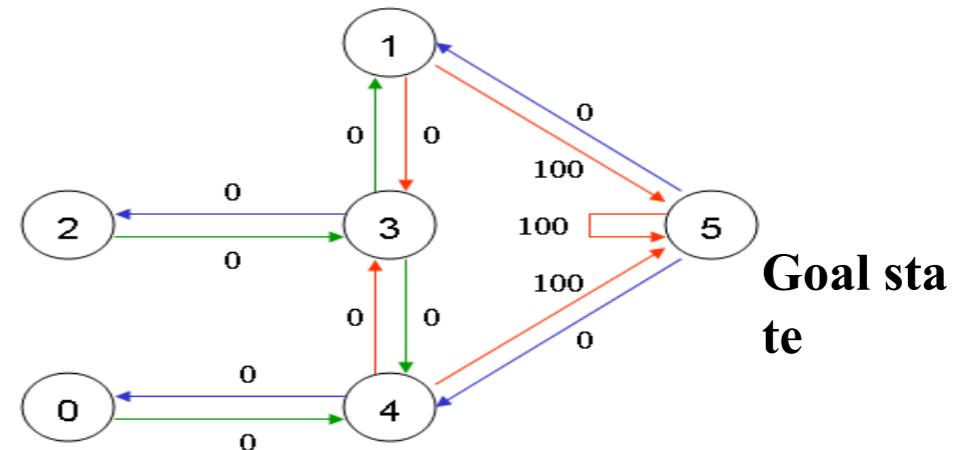
$$Q(state, action) = R(state, action) + \gamma * \text{Max}[Q(next\ state, all\ actions)]$$

$$Q(1,5) = R(1,5) + 0.8 * \text{Max}[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100$$

- The next state, 5, now becomes the current state.
- State 5 is the goal state.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

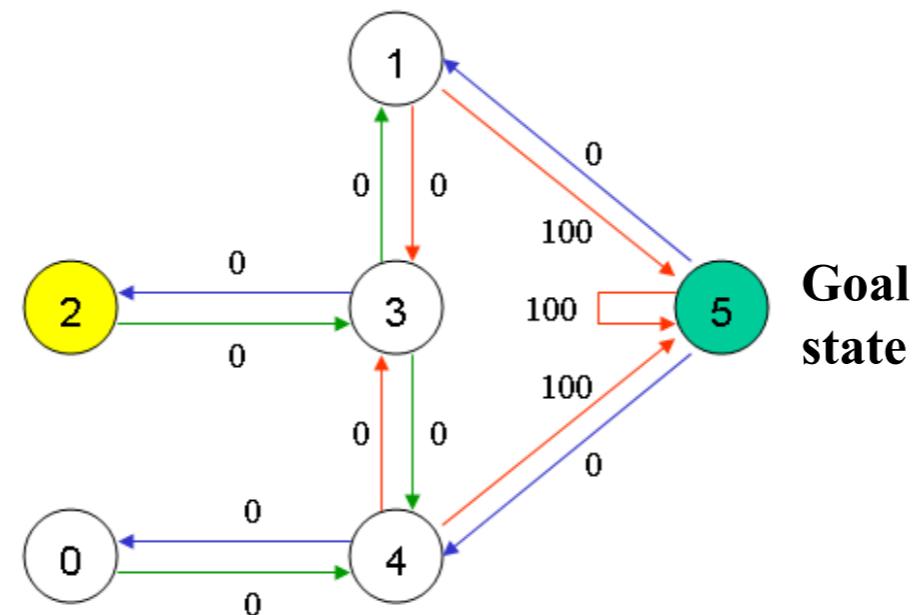
- **Initial state:** randomly select 3
- The agent can go to 1, 2, or 4.
- Randomly select 1.



$$Q(3, 1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 2), Q(1, 5)] = 0 + 0.8 * \text{Max}(0, 100) = 80$$

$$Q = \begin{matrix} & \begin{matrix} \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \end{matrix} \\ \begin{matrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix}$$

- The next state, 1, now becomes the current state.
- Repeat the inner loop of the Q-learning algorithm



- The agent is in room 5.
- The agent can go to 1, 4, or 5.

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

- 5 is the goal state, finish this episode.

$$Q = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 100 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 80 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

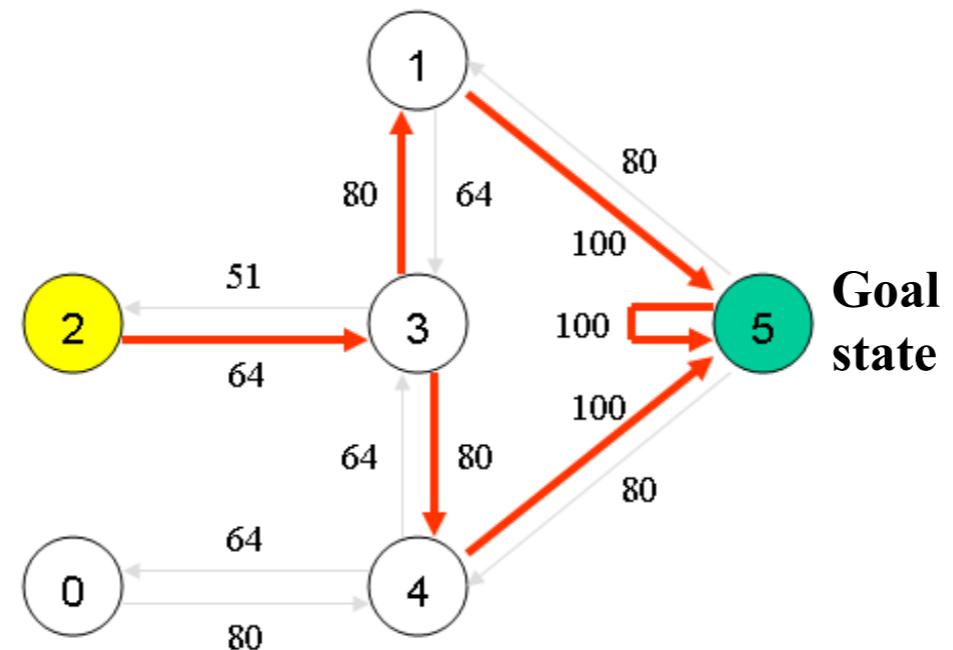
- If agent learns more through further episodes, will finally reach convergence values in matrix Q like:

$$Q = \begin{bmatrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{bmatrix}$$

- Matrix Q Normalization

$$Q = \begin{bmatrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{bmatrix}$$

- Optimal paths



- If MDP is known, e.g., T and R :

Offline Solution

Goal	Technique
Compute V^*, Q^*, π^*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

- If MDP is Unknown, e.g., T and R :

Model-Based

Goal	Technique
Compute V^*, Q^*, π^*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Model-Free

Goal	Technique
Compute V^*, Q^*, π^*	Q-learning
Evaluate a fixed policy π	TD/Value Learning



Office: #432, Daeyang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr

Artificial Intelligence

Md. Jalil Piran, PhD

Professor (Associate)

Computer Science and Engineering

Sejong University



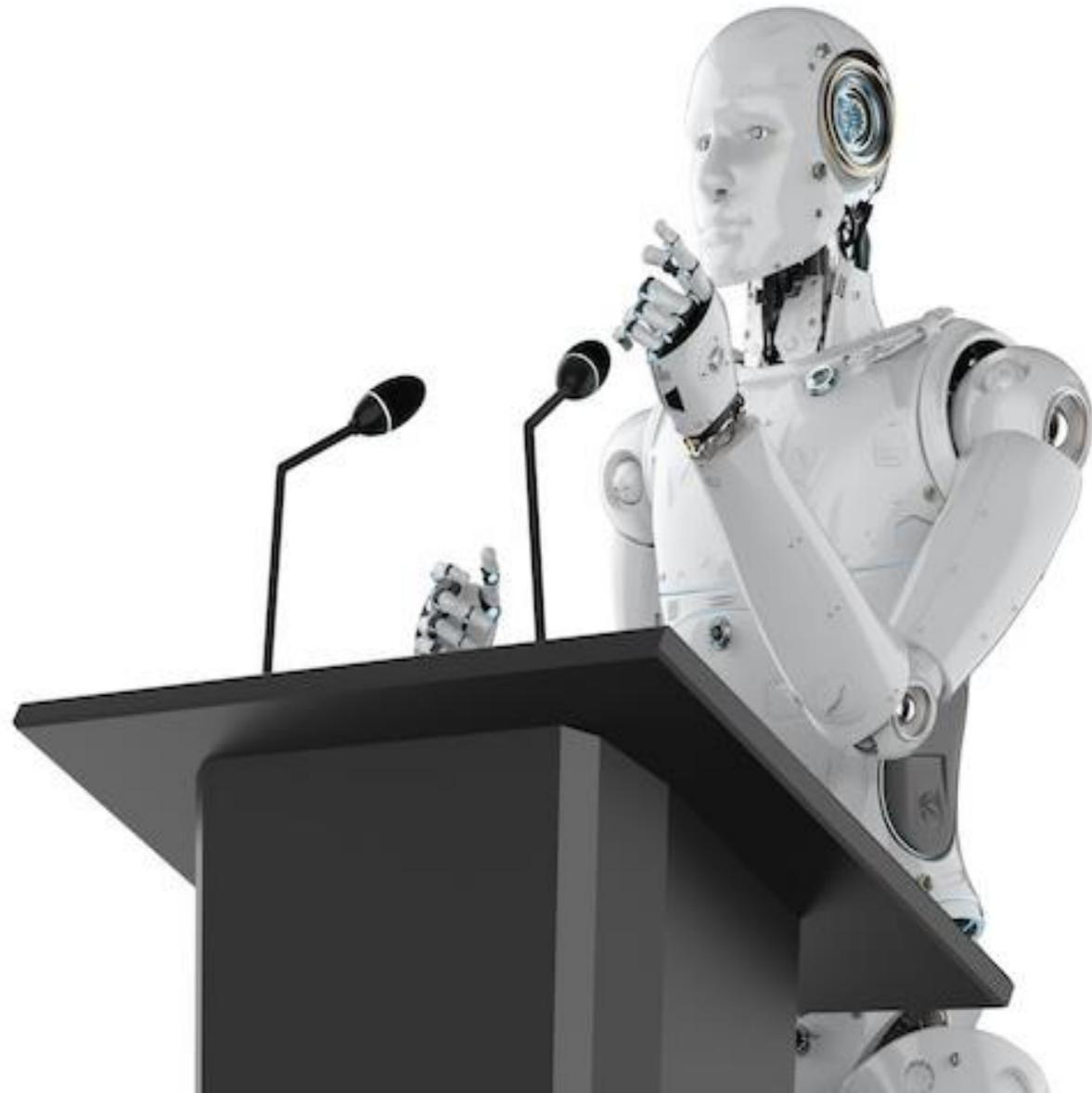
Course Outline



- Introduction to AI
- The History of AI
- Python Programming
- Problem Solving by Search
 - Informed Search
 - Beyond Classical Search
 - Adversarial Search
 - Constraint Satisfaction Problems
- Fuzzy Logic
- Machine Learning
 - Supervised Learning
 - Unsupervised Learning
 - Reinforcement Learning
- **Natural language Processing**

NATURAL LANGUAGE PROCESSING

- Topics
 - Basics
 - History
 - Tasks
 - Components
 - Pipeline
 - ML Models for NLP
 - NLP Libraries in Python



Data Sources

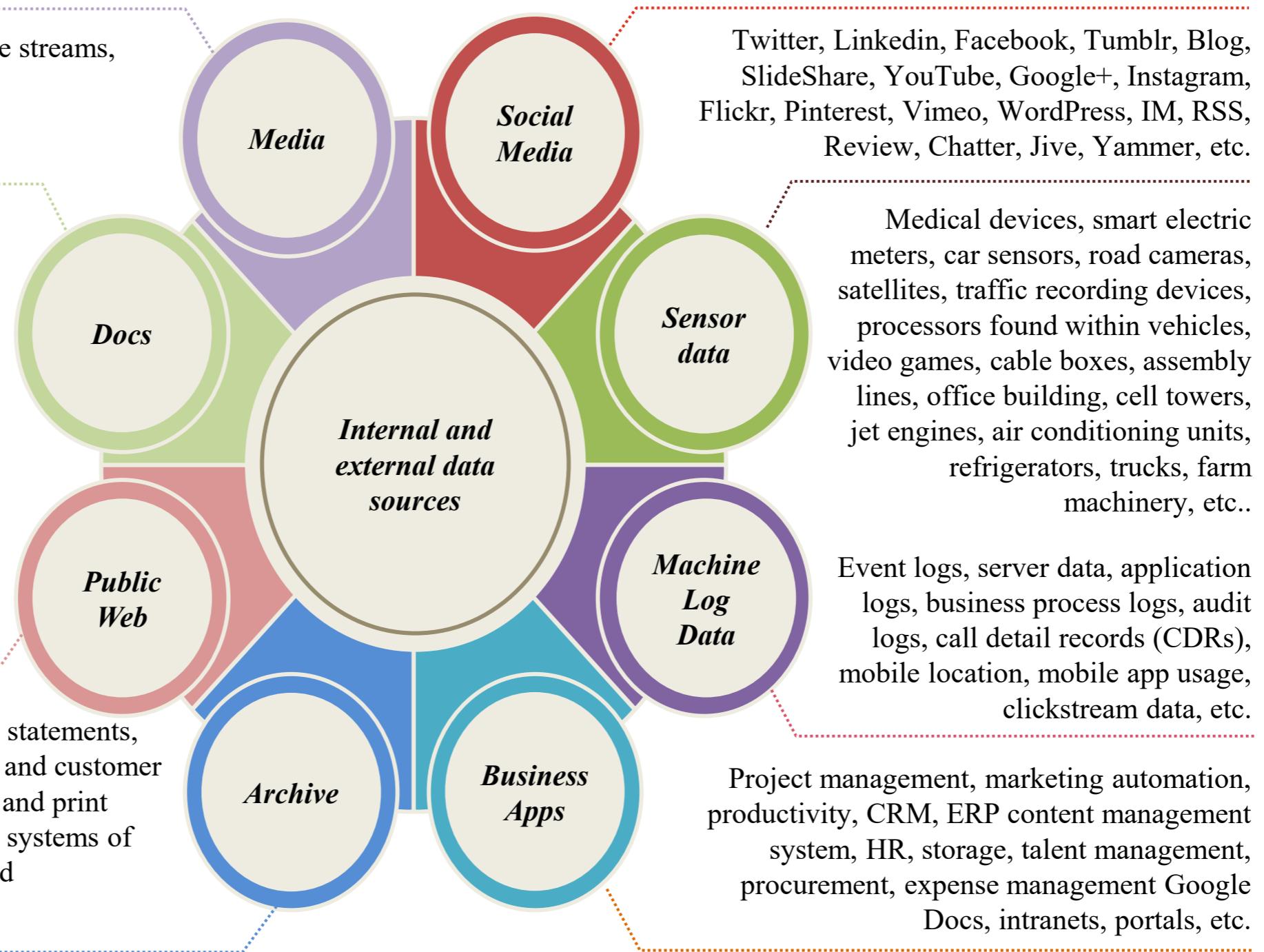


Images, videos, audio, Flash, live streams, podcasts, etc.

XLS, PDF, CSV, email, Word, PPT, HTML, HTML 5, plain text, XML, JSON, etc.

Government, weather, competitive, traffic, regulatory, compliance, health care services, economic, census, public finance, stock, OSINT, the World Bank, SEC/Edgar, Wikipedia, IMDb, etc.

Archives of scanned documents, statements, insurance forms, medical record and customer correspondence, paper archives, and print stream files that contain original systems of record between organizations and their customers



Data Classification by Structure



STRUCTURED

- Relational DB
- Excel Spreadsheets
- JSON (tabular format)
- CSV Files

SEMI- STRUCTURED

- JSON Data
- XML
- Log Files
- NoSQL DB

UNSTRUCTURED

- Text Documents
- Images and Videos
- Audios
- Social Networks Data
- Emails



Natural Languages

- Natural languages refer to the human way of communication.
 - Speech
 - Text



Language Terminologies



- **Phonetics and Phonology** (sounds)
- **Morphology** (structure of words)
- **Syntax** (structure of sentences)
- **Semantics** (meaning)
- **Pragmatics** (language use)
- **Discourse and Dialogue** (units larger than single utterance)

Language Terminologies

- **Morphology**

- The study of how words are formed from minimal meaning-bearing units (**morphemes**)
- **Types:**
 - **Stems**
 - **Affixes**
- **Examples:**
 - **Plural:** cat-s, fox-es, fish
 - **Tense:** walk-s, walk-ed
 - **Nominalization:** kill-er, fuzz-iness
 - **Compounding:** book-case, over-load, wash-cloth



Language Terminologies

- **Syntax**

- The study of how sentences are formed by **grouping** and **ordering** words.
- The allowable structures in the language: sentences, phrases, affixes
 - -ing,
 - -ed,
 - -ment,
 - ...
- Based on **Substitution / Movement / Coordination Tests**

Language Terminologies

- **Semantics**

- The study of the **meaning of words, intermediate constituents and sentences**.

- e.g., the meaning(s) of texts in the language.

- **Examples:**

- **Words:** “purchase” vs. “buy”, “hot” vs. “cold”

- **Sentences:** “Mary has a new car”

$$\exists x, y \text{ Having}(x) \wedge \text{Haver}(\text{Mary}, x) \wedge$$
$$\text{HadThing}(y, x) \wedge \text{Car}(y) \wedge \text{HasAge}(y, z) \wedge$$
$$<(z, 1 \text{ year})$$

- “Mary’s car is old”?

- Symbolic structure that corresponds to objects and relations in some world being represented

Language Terminologies

- **Pragmatics**

- The study of the **meaning** of a sentence that comes from **context-of-use**.

- **Examples:**

“Yesterday, she did much better”

“The judge denied the prisoner’s request because he was cautious/dangerous”

“Can you pass me the salt?”

- The study of how language is used to achieve goals:

- e.g., convince someone to quit smoking.



Language Terminologies

- **Part-of-Speech (POS):** the category of a word
 - Noun,
 - Verb,
 - Preposition,
 - ...
- **Bag-of-words (BoW):**
 - A featurization that uses a vector of word counts (or binary) ignoring order.
- **N-gram:**
 - A consecutive sequence of words in a text, e.g., N (2-5 is common), .

Challenges

Non-standard English

Great job @justinbieber! Were SOO PROUD of what youve accomplished! U taught us 2 #neversaynever & you yourself should never give up either♥

Segmentation issues

the New York-New Haven Railroad
the New York-New Haven Railroad

Idioms

dark horse
get cold feet
lose face
throw in the towel

Neologisms

unfriend
Retweet
bromance

World knowledge

Mary and Sue are sisters.
Mary and Sue are mothers.

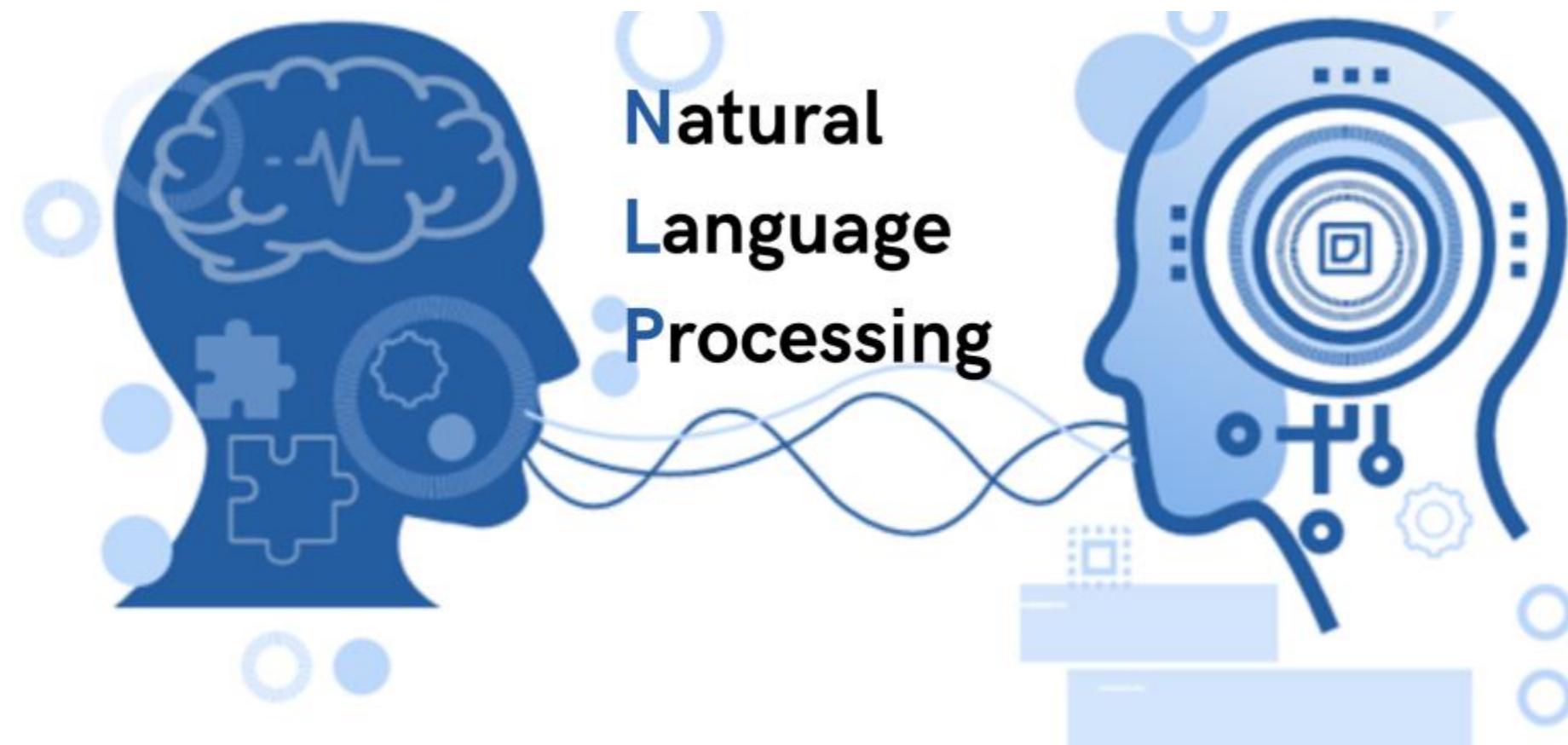
Tricky entity names

Where is *A Bug's Life* playing ...
Let It Be was recorded ...
... a mutation on the *for* gene ...

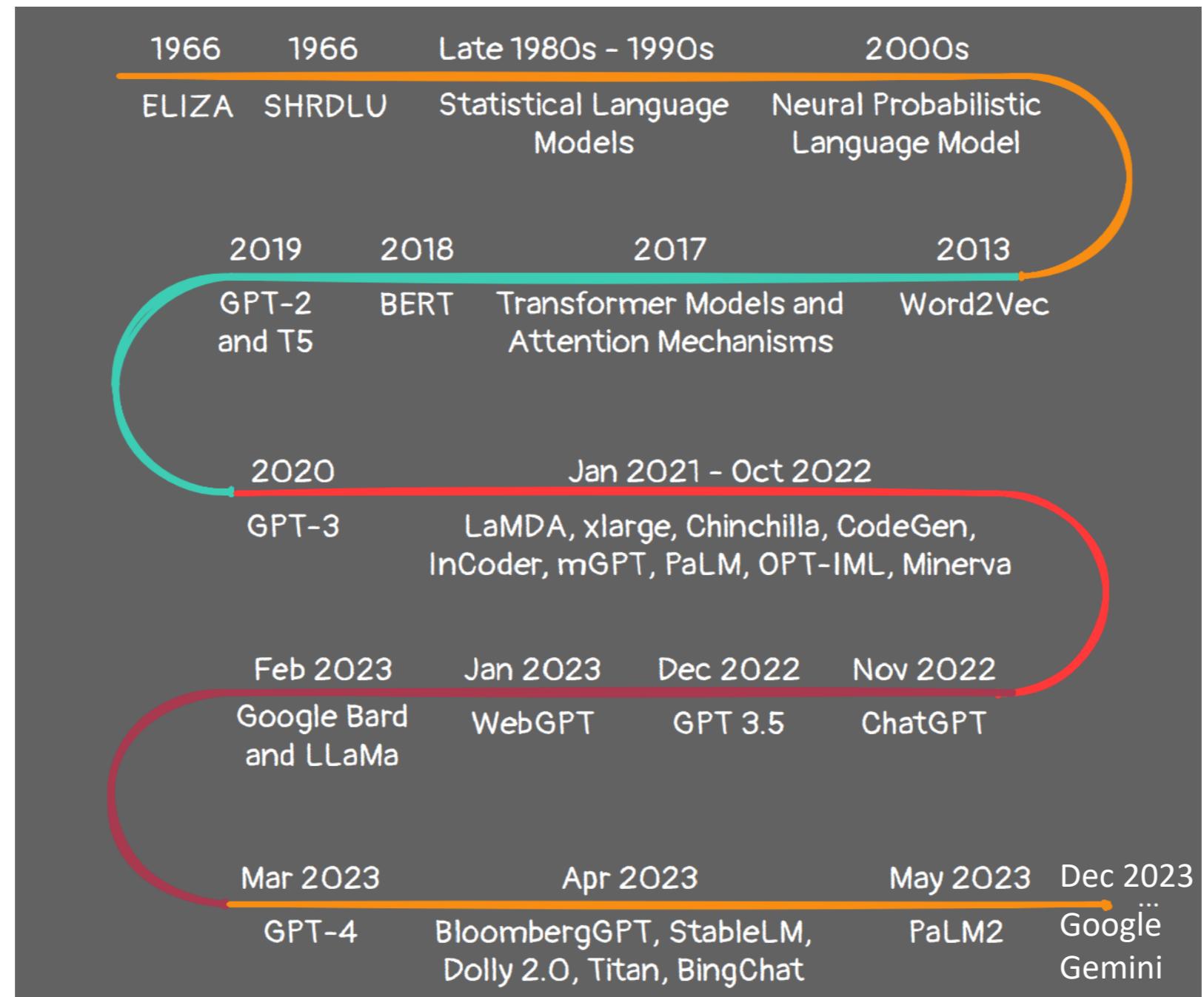
Natural Language Processing



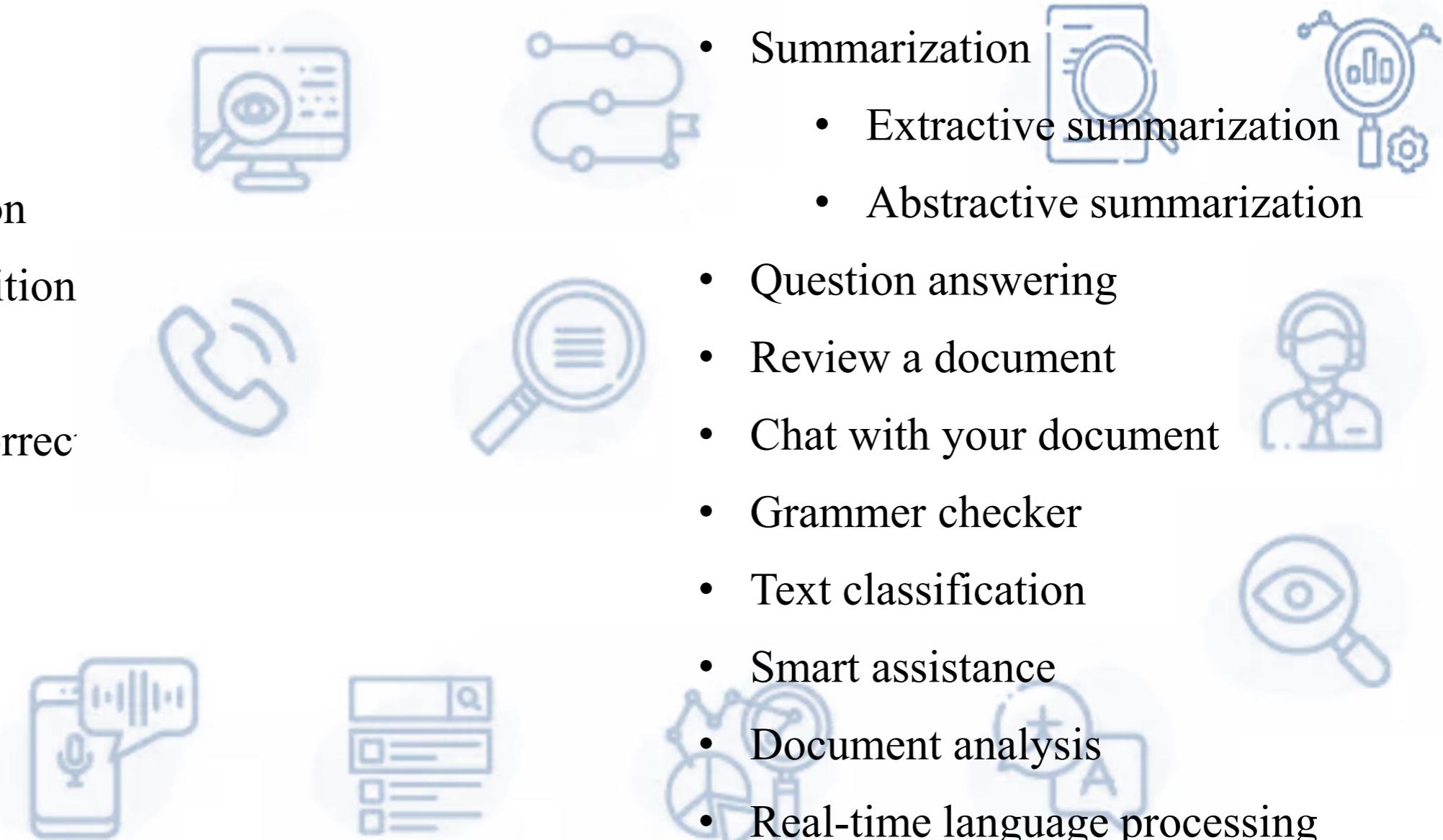
- Natural Language Processing (NLP) refers to the branch of AI that allows machines to understand natural languages.



NLP History



- Machine Translation
- Semantic Analysis
- Toxicity Classification
- Named entity recognition
- Spam detection
- Grammatical error correction
- Topic modeling
- Text generation
 - Autocomplete
 - Chatbots
- Information retrieval



- Summarization
 - Extractive summarization
 - Abstractive summarization
- Question answering
- Review a document
- Chat with your document
- Grammer checker
- Text classification
- Smart assistance
- Document analysis
- Real-time language processing
- Similarity check

Level 1

Spam detection

Related emails...



Advertisements...



Part-of-speech (POS) tagging

ADJ ADJ NOUN VERB ADV

Colorless green ideas sleep furiously.

Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

Level 2

Sentiment analysis

Best roast chicken in San Francisco!



The waiter ignored us for 20 minutes.

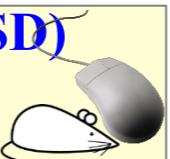


Coreference resolution

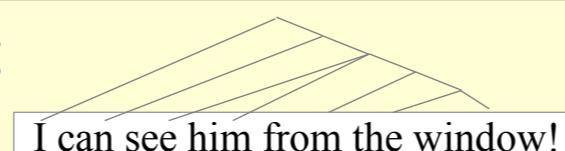
Carter told Mubarak he shouldn't run again.

Word sense disambiguation (WSD)

I need new batteries for my **mouse**.



Parsing



I can see him from the window!

Machine translation (MT)

나는 인공 지능을 좋아합니다.



I love Artificial Intelligence.

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30



Level 3

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?



Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

Summarization

The Dow Jones is up

The S&P500 jumped

Housing prices rose



Economy is good

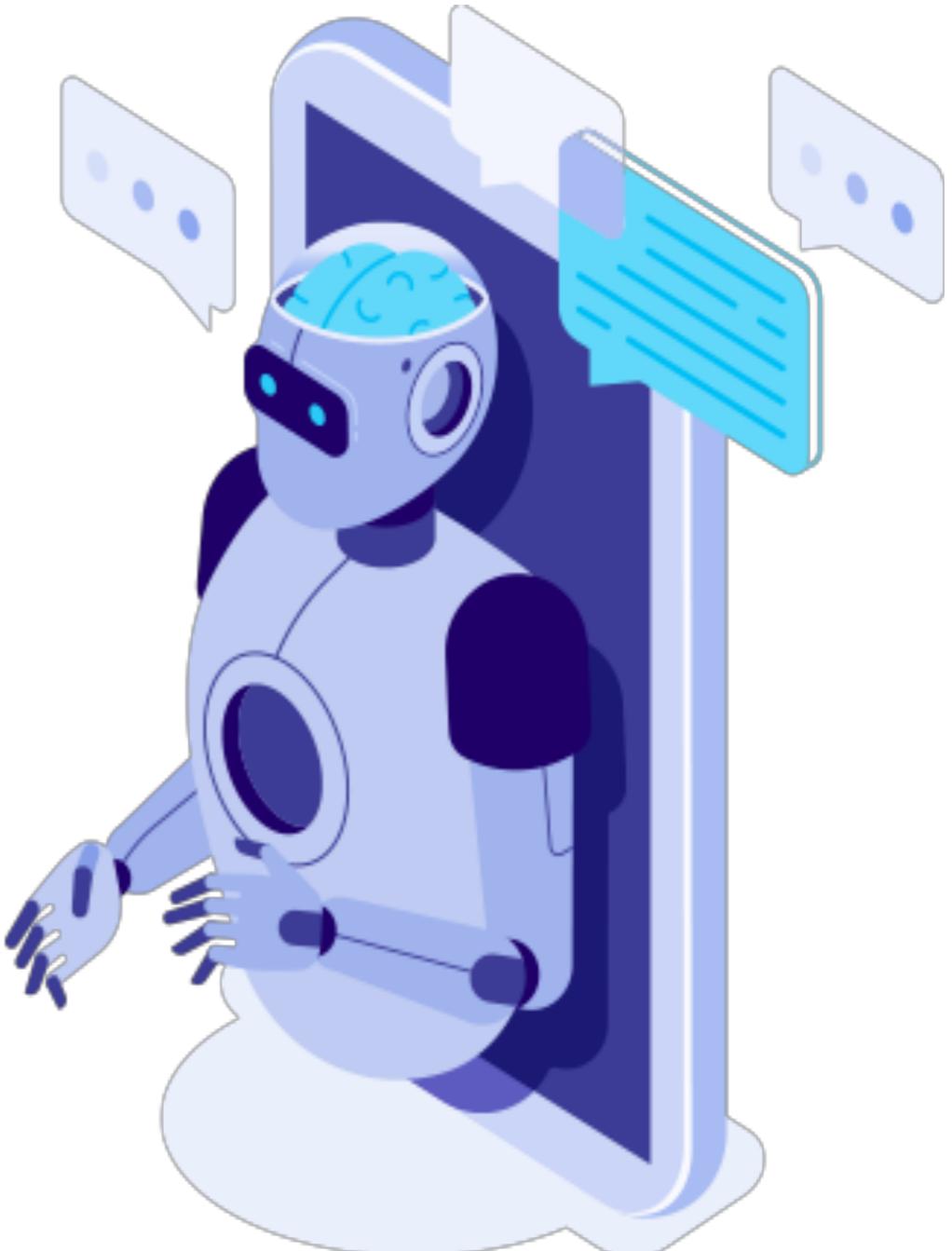
Dialog

Where is Citizen Kane playing in SF?



Castro Theatre at 7:30. Do you want a ticket?

- **Machine Translation (MT)**
 - A sub-field of computational linguistics
 - Investigates the use of software to translate text or speech from one language to another.
 - Types:
 - **Statistical Machine Translation (SMT)**
 - **Neural Machine Translation (NMT)**



- **Information Extraction (IE)**

- Automatically extracting structured information from unstructured documents.
- Goal: Map a document collection to structured database.
- Examples:
 - Complex searches (“Find me all the jobs in advertising paying at least \$50,000 in Boston”)
 - Statistical queries (“How has the number of jobs in accounting changed over the years?”)

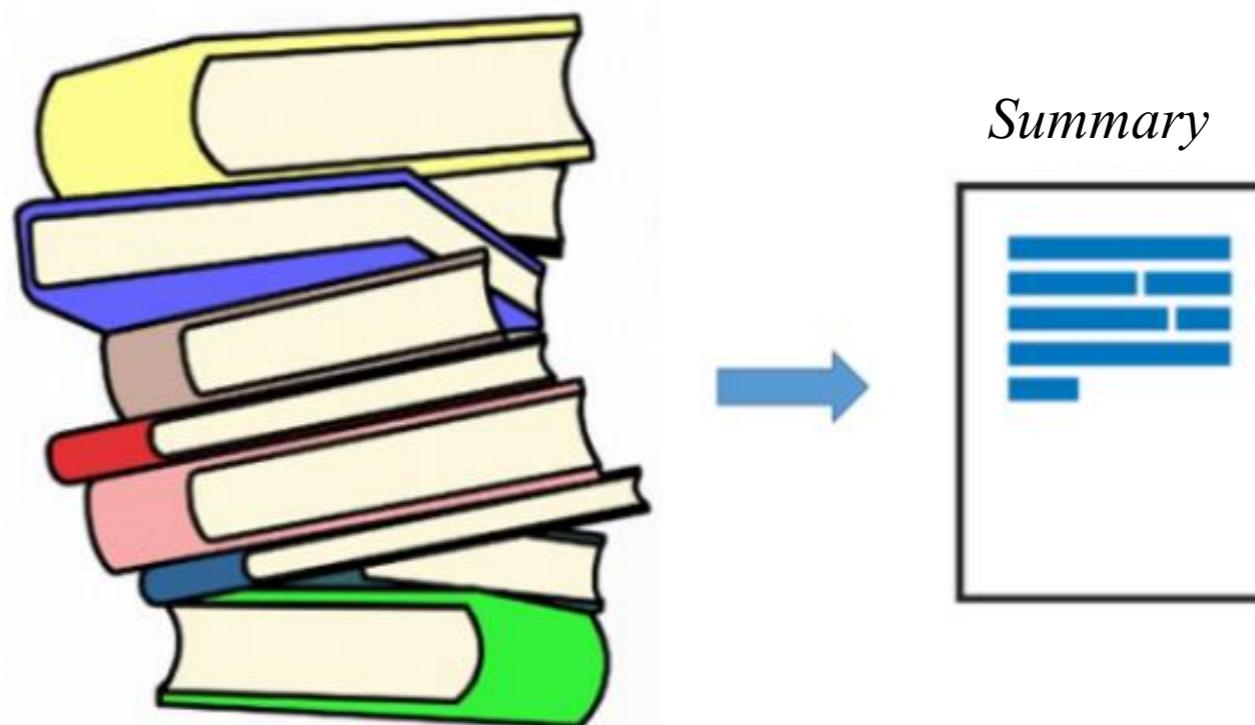


- **Information Extraction (IE)**

"Artificial Intelligence (AI) is a field of computer science that focuses on creating intelligent machines capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, and language understanding. AI systems use algorithms and large datasets to learn patterns and make decisions, with applications ranging from natural language processing and computer vision to robotics and autonomous vehicles. As AI continues to advance, ethical considerations around its use, transparency, and accountability become crucial. AI technologies, such as neural networks, play a pivotal role in transforming industries, automating processes, and enhancing human capabilities."

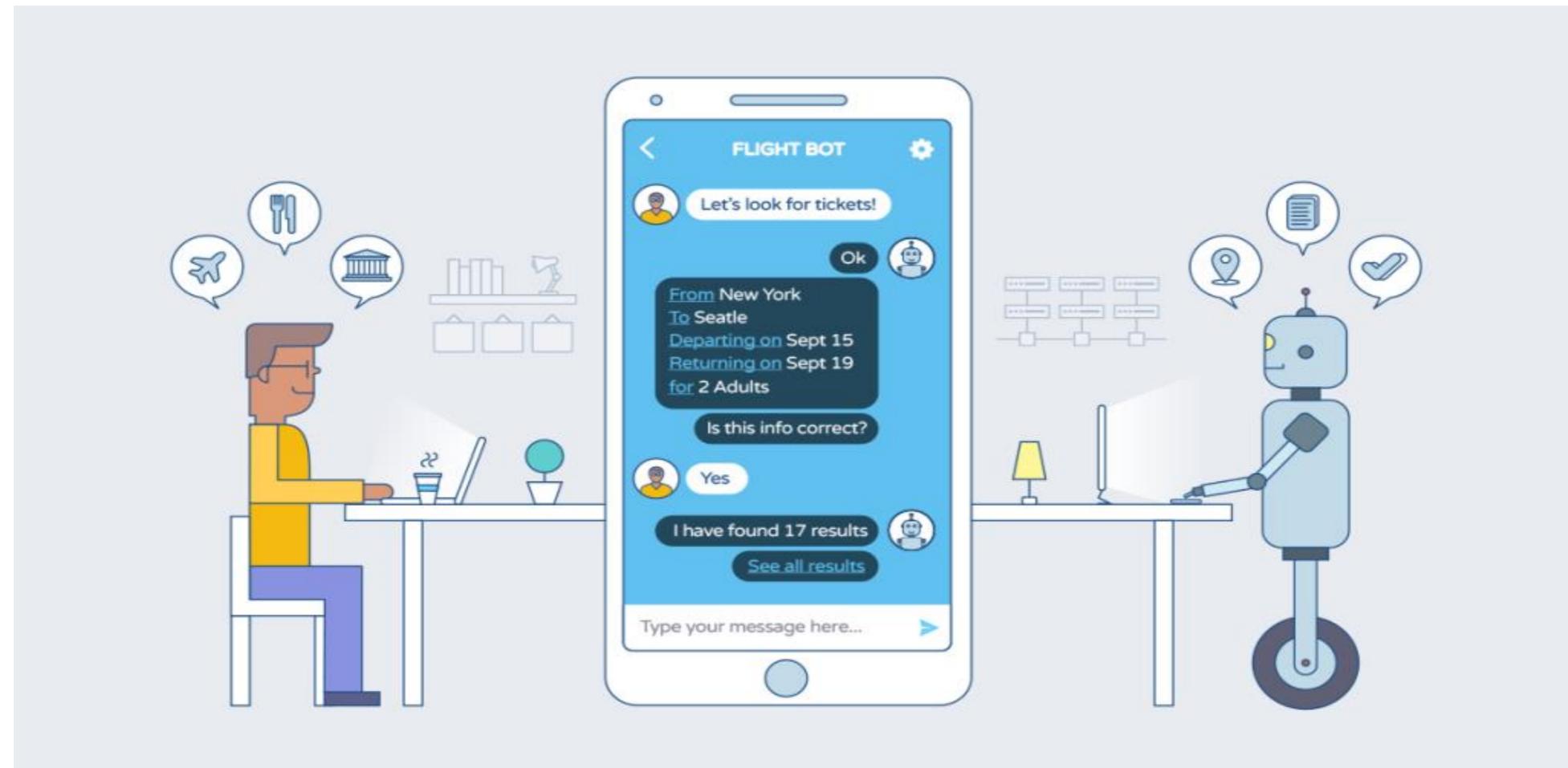
- **Definition of AI:** A field of computer science focusing on creating intelligent machines.
- **Tasks of AI:** learning, reasoning, problem-solving, perception, and language understanding.
- **AI Applications:** natural language processing, computer vision, robotics, and autonomous vehicles.
- **Ethical Considerations:** Ethical considerations around AI use, transparency, and accountability are highlighted.
- **AI Technologies:** neural networks.

- **Text Summarization**

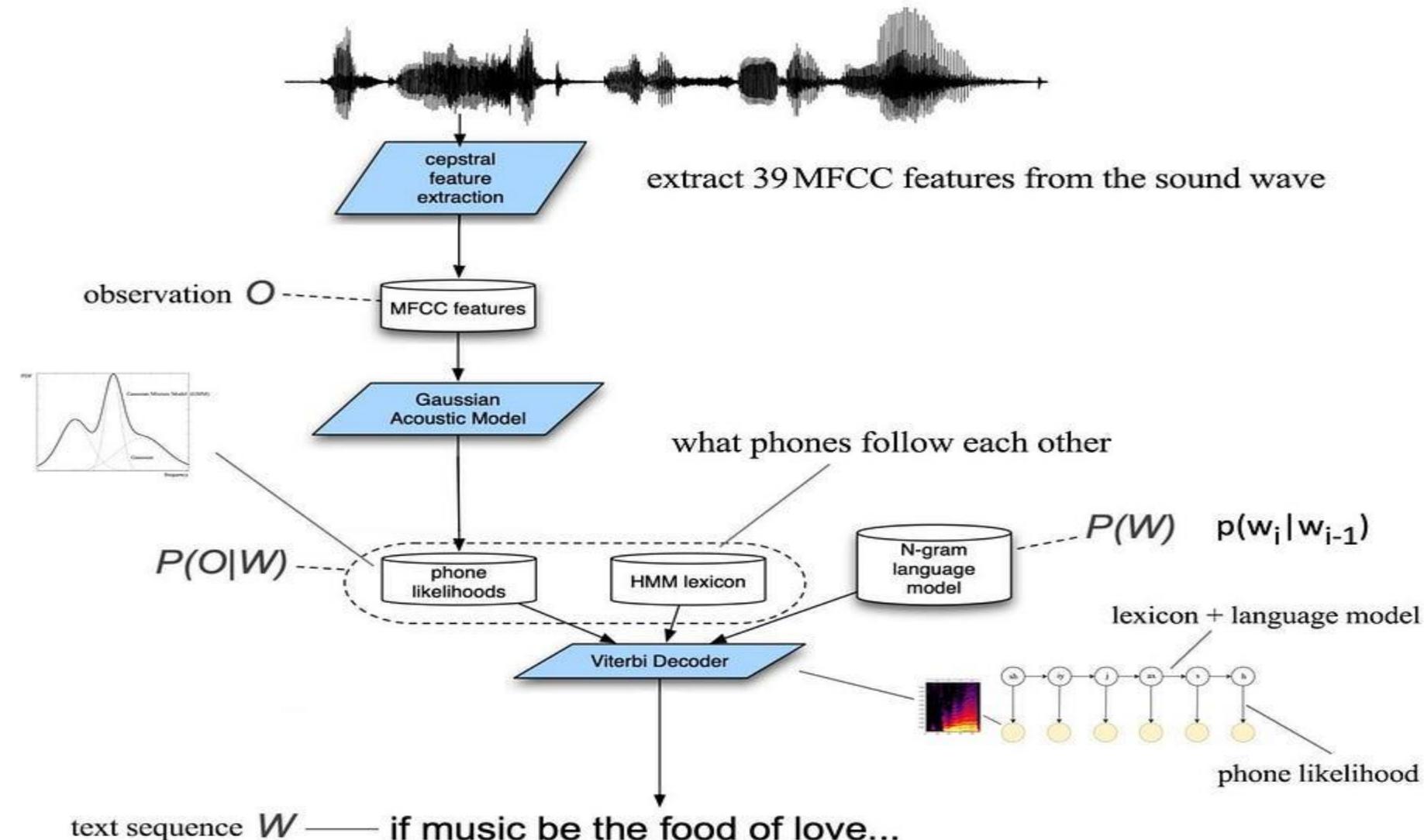


- **Chatbots**

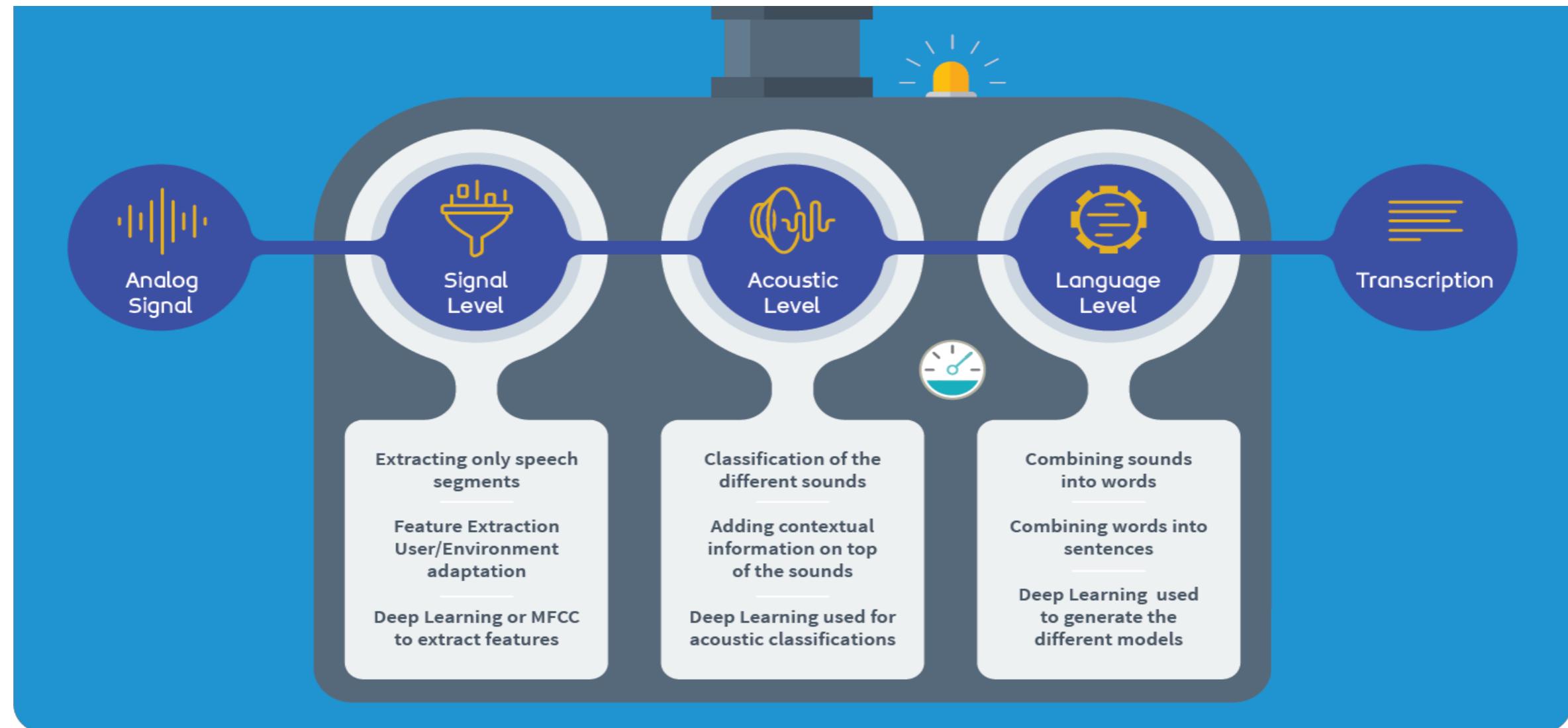
- A computer program which conducts a conversation via auditory or textual methods



- Speech Recognition



- Speech Recognition



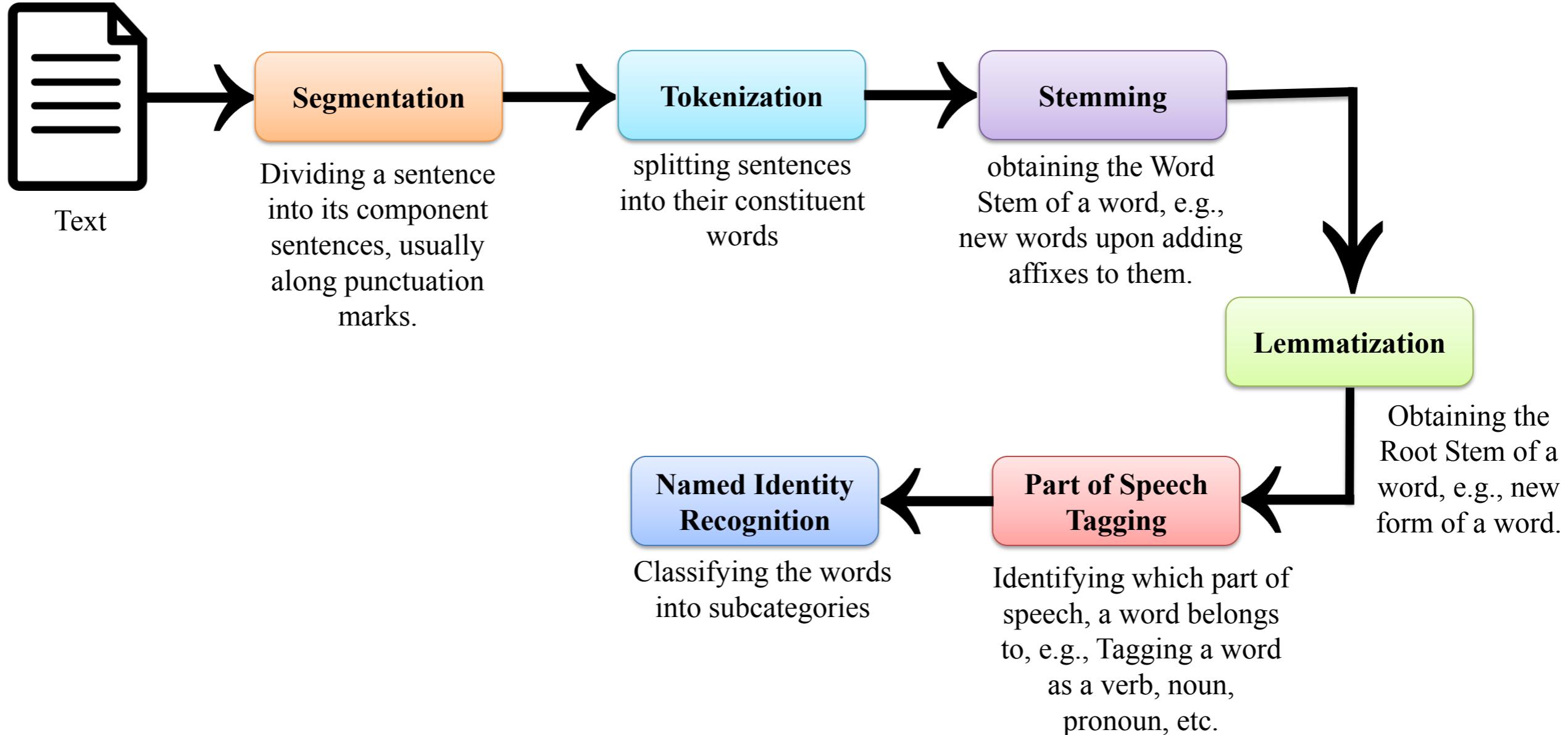
NLP Components



- **Natural language understanding**
 - Mapping the given input in the natural language into a useful representation
 - Different level of analysis required:
 - Morphological analysis
 - Syntactic analysis
 - Semantic analysis
 - Discourse analysis
- **Natural language generation**
 - Taking some formal representation of what you want to say and working out a way to express it in a natural language.
 - Producing output in the natural language from some internal representation
 - Different level of synthesis required:
 - Deep planning (what to say)
 - Syntactic generation



NLP Pipeline



- Recurrent Neural Networks (RNNs)
- Seq2Seq (Sequence-to-Sequence)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Transformers

NLP Libraries in Python



- Natural Language Toolkit (NLTK),
- Spacy
- Textblob
- TensorFlow
- Pytorch
- Scikit-learn
- ...



Office: #432, Daeyang AI Center,

Phone: 010-8999-8586,

email: piran@sejong.ac.kr