

Machine Learning Week 8



outline

Week	Topics
8	Unsupervised Learning: Clustering 1- K Means Clustering
9	Unsupervised Learning: Clustering II – Hierarchical Clustering
10	Unsupervised Learning: EM Algorithm
11	Pengujian Unsupervised Learning
12	Reinforcement Learning
13	Feature reduction
14	Ensemble Learning



Clustering

- Unsupervised Learning -

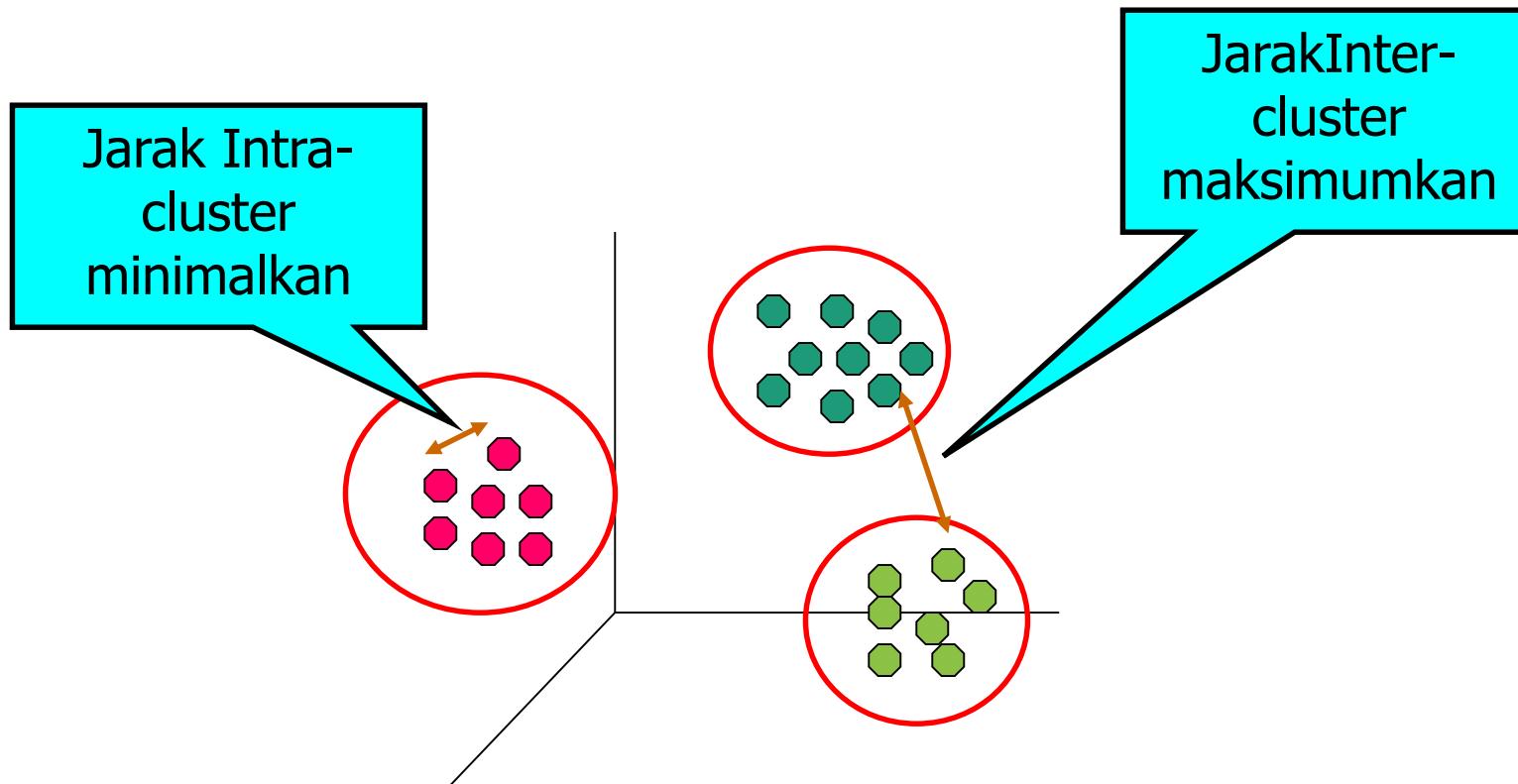


Clustering - Basic Concept

- Cluster analysis groups data objects based only on information found in the data that describes the objects and their relationships.
- The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
- The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.



Clustering - Basic Concept (cont.)



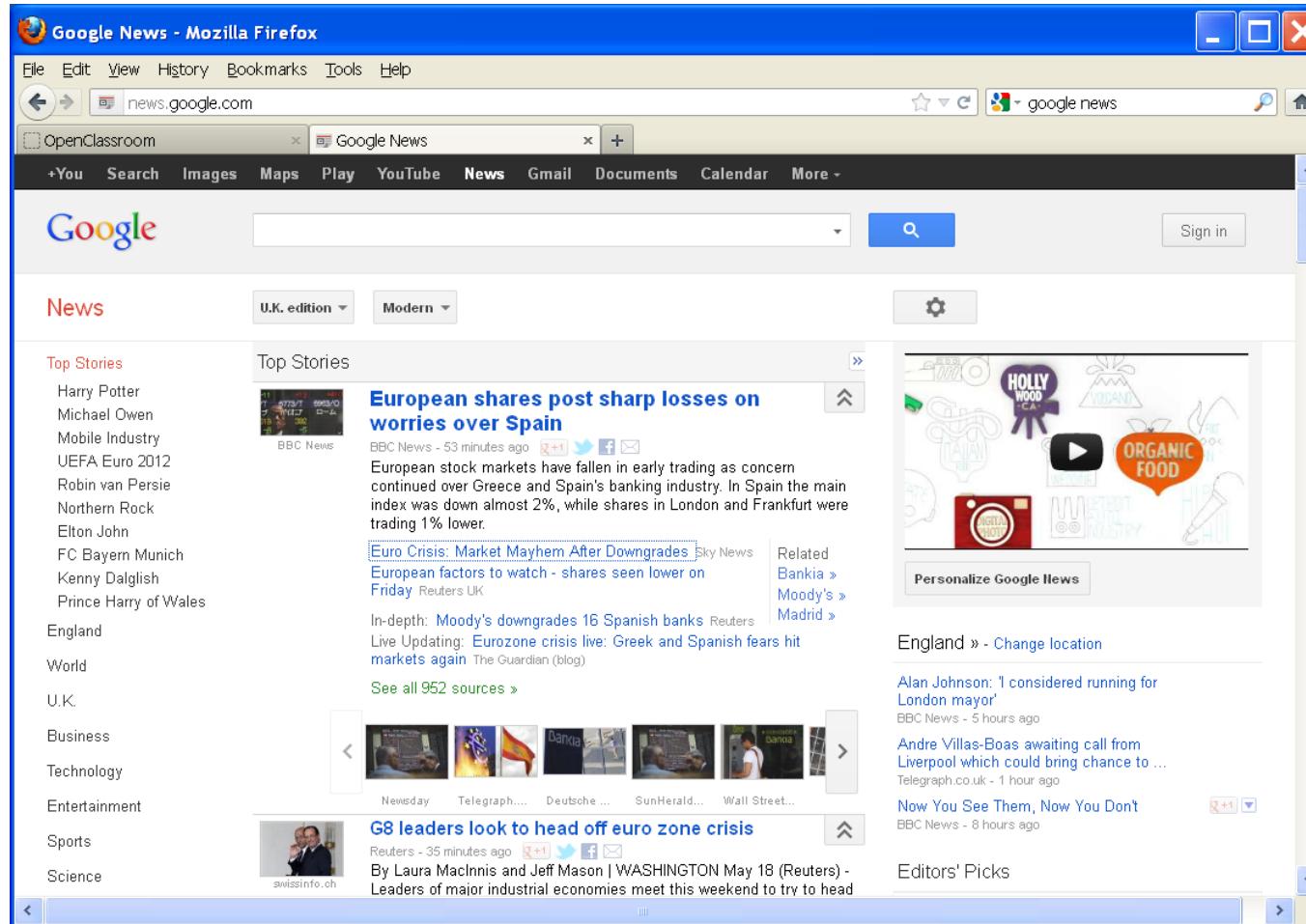
Clustering - Basic Concept (cont.)

- Clustering can be regarded as a form of classification in that **it creates a labeling of objects** with class (cluster) labels.
- However, it derives these labels only from the data. For this reason, cluster analysis is sometimes referred to as **unsupervised classification**.



Application

- Real Applications: Google News



Application

- A technique demanded by many real world tasks
 - **Bank/Internet Security:** fraud/spam pattern discovery
 - **Biology:** taxonomy of living things such as kingdom, phylum, class, order, family, genus and species
 - **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
 - **Climate change:** understanding earth climate, find patterns of atmospheric and ocean
 - **Finance:** stock clustering analysis to uncover correlation underlying shares
 - **Image Compression/segmentation:** coherent pixels grouped
 - **Information retrieval/organisation:** Google search, topic-based news
 - **Land use:** Identification of areas of similar land use in an earth observation database
 - **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
 - **Social network mining:** special interest group automatic discovery



Clustering - Basic Concept (cont.)

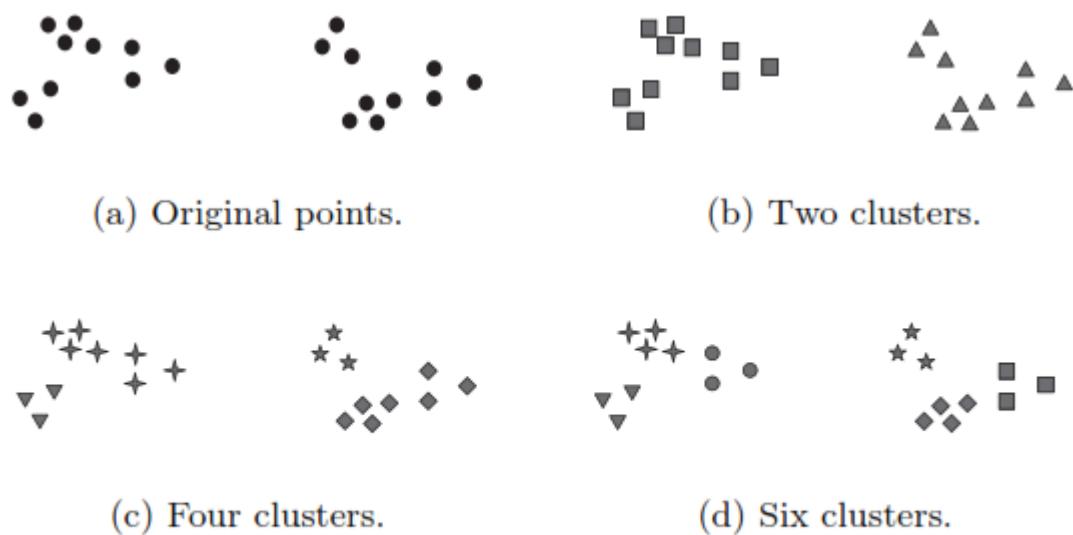


Figure 8.1. Different ways of clustering the same set of points.

- This figure illustrates that the definition of a cluster is imprecise and that the best definition depends on the nature of data and the desired results.



Different Types of Clusterings

- Hierarchical versus Partitional
- Exclusive versus Overlapping versus Fuzzy
- Complete versus Partial



Hierarchical versus Partitional

- A **partitional clustering** is simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- If we permit clusters to have subclusters, then we obtain a **hierarchical clustering**, which is a set of nested clusters that are organized as a tree.
- Each node (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters), and the root of the tree is the cluster containing all the objects.
- Often, but not always, the leaves of the tree are singleton clusters of individual data objects.





(a) Original points.



(b) Two clusters.



(c) Four clusters.



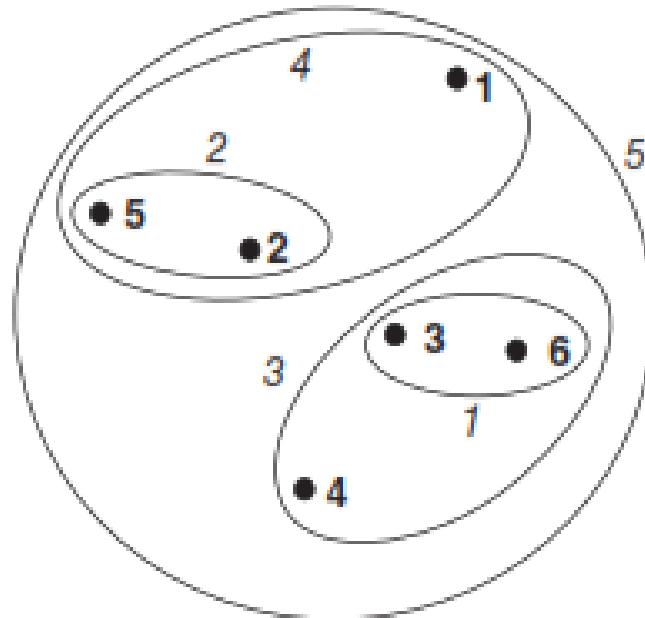
(d) Six clusters.

Figure 8.1. Different ways of clustering the same set of points.

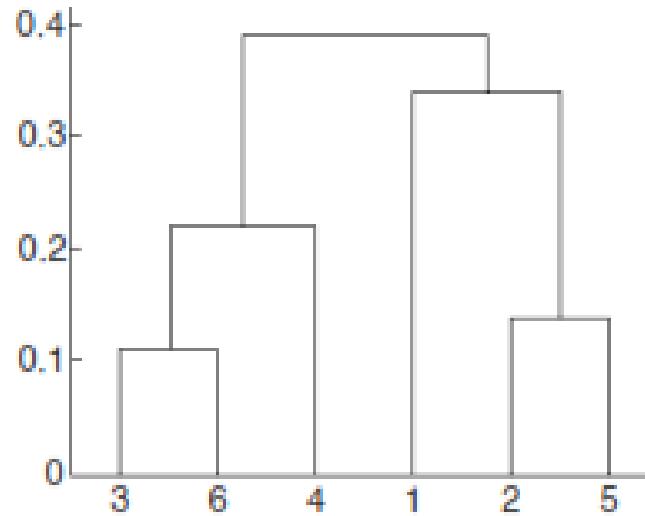
- Taken individually, each collection of clusters in Figures 8.1 (b–d) is a **partitional clustering**.
- If we allow clusters to be nested, then one interpretation of Figure 8.1(a) is that it has two subclusters (Figure 8.1(b)), each of which, in turn, has three subclusters (Figure 8.1(d)).



Some representations of hierarchical clustering



Nested cluster diagram.



Dendrogram.



Hierarchical versus Partitional (cont.)

- A hierarchical clustering can be viewed as a sequence of partitional clusterings.
- A partitional clustering can be obtained by taking any member of that sequence; i.e., by cutting the hierarchical tree at a particular level.



Exclusive vs. Overlapping vs. Fuzzy

- Exclusive: each object is assigned into a single cluster.
- Overlapping or non-exclusive: an object can simultaneously belong to more than one group (class).
 - A non-exclusive clustering is also often used when, for example, an object is “between” two or more clusters and could reasonably be assigned to any of these clusters.
- In a fuzzy clustering, every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn’t belong) and 1 (absolutely belongs) → clusters are treated as fuzzy sets.



Exclusive vs. Overlapping vs. Fuzzy (cont.)

- In fuzzy clustering, we often impose the additional constraint that the sum of the weights for each object must equal 1.
- Similarly, probabilistic clustering techniques compute the probability with which each point belongs to each cluster, and these probabilities must also sum to 1.
- A fuzzy or probabilistic clustering does not address true multiclass situations, such as non-exclusive clustering.
- In practice, a fuzzy or probabilistic clustering is often converted to an exclusive clustering by assigning each object to the cluster in which its membership weight or probability is highest



Complete vs. Partial

- A complete clustering assigns every object to a cluster.
 - For example, an application that uses clustering to organize documents for browsing needs to guarantee that all documents can be browsed
- A partial clustering does not assign every object to a cluster.
 - Some objects in a data set may not belong to well-defined groups, even some objects may represent noise or outliers.
 - For example, some newspaper stories may share a common theme, such as global warming, while other stories are more generic or one-of-a-kind.



K-Means Clustering



Prototype-based clustering

- K-means adalah prototype based clustering dan merupakan one-level partitioning dari objek yang ada pada data
- **K-means**
 - Mendefinisikan prototype dalam bentuk centroid(mean of a group of points)
 - A centroid almost never corresponds to an actual data point
- **K-medoid**
 - Prototype dalam bentuk medoid (the most representative point for a group of points)
 - A medoid, by its definition, must be an actual data point

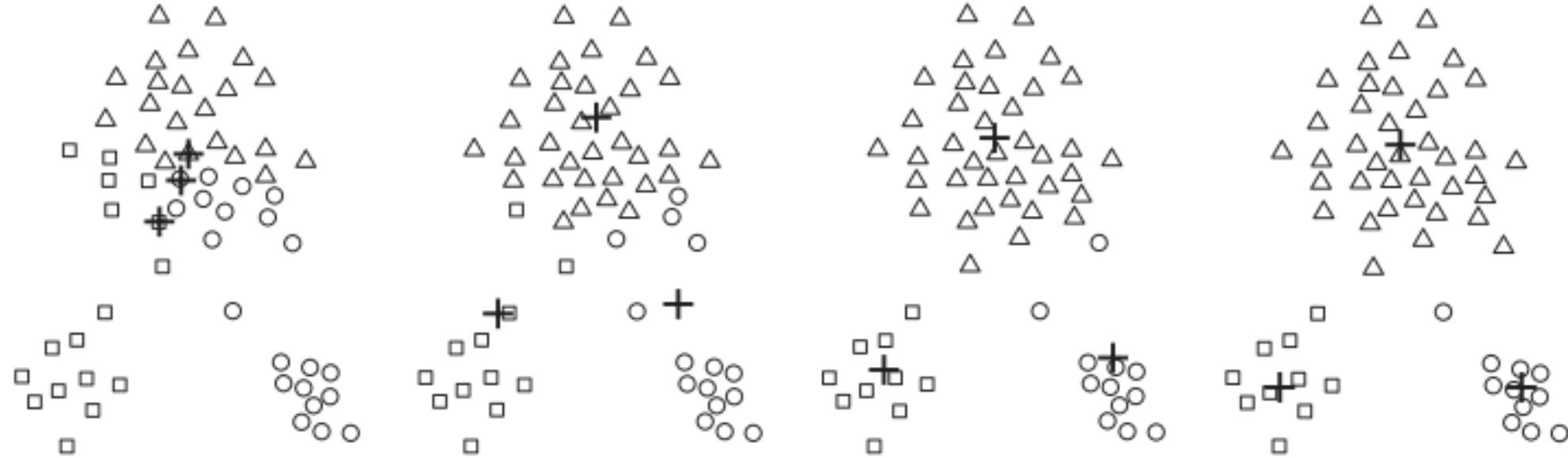
Konsep K-Means Clustering

- Mengelompokkan n objek ke dalam K cluster berdasarkan nilai atribut dari objek tersebut.
- K adalah jumlah cluster, berupa bilangan integer positif.
- Merupakan jenis hard clustering → 1 objek hanya dapat menjadi anggota dari 1 cluster secara eksklusif.

The Basic K-means Algorithm

Algorithm 8.1 Basic K-means algorithm.

- 1: Select K points as initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning each point to its closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** Centroids do not change.
-



(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

(d) Iteration 4.

Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

- Centroid dilambangkan dengan “+”
- All points belonging to the same cluster have the same marker shape.

Some Distance Measures

Distance/ Dissimilarity Measures

- Interval-scaled variables
 - Euclidean distance
 - Manhattan distance
 - Minkowski distance
- Categorical variables
- Ordinal variables

Interval-scaled variables

- Continuous measurements of a roughly linear scale.
- Contoh: tinggi badan, berat badan, suhu ruangan, koordinat lintang dan bujur.
- Beberapa ukuran distance yang dapat digunakan:
 - Euclidean Distance
 - Minkowski Distance
 - Manhattan Distance

Mathematic requirement for distance measures

- $d(x, y) \geq 0$ Jarak tidak boleh negatif.
- $d(x, x) = 0$ Jarak objek terhadap dirinya sendiri adalah 0.
- $d(x, y) = d(y, x)$ Jarak merupakan fungsi simetris.
- $d(x, y) \leq d(x, z) + d(z, y)$ jarak objek x terhadap objek y secara langsung dalam sebuah ruang tidak melebihi jarak jika dilewatkan ke objek lain (*triangular inequality*).

Euclidean Distance

- Ukuran jarak yang paling sering digunakan untuk data numerik.
- Jarak Euclidean antara 2 titik atau tuple, $x = (x_1, x_2, \dots, x_d)$ dan $y = (y_1, y_2, \dots, y_n)$ pada ruang dimensi d adalah

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

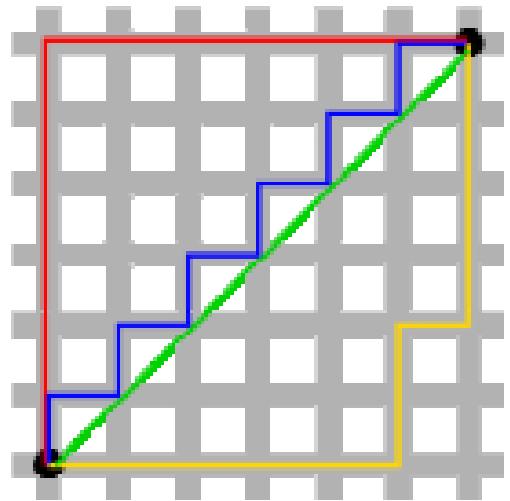
Manhattan (City Block Distance)

- Jumlah jarak dari semua atribut.
- Jarak Manhattan antara 2 titik atau tuple, $x = (x_1, x_2, \dots, x_d)$ dan $y = (y_1, y_2, \dots, y_n)$ pada ruang dimensi d adalah

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

Examples:

- Tentukan jarak titik A(3,10) dan titik B(6,8) menggunakan:
 - Jarak Euclidean
 - Jarak Manhattan



Manhattan versus Euclidean distance:

The red, blue, and yellow lines represent Manhattan distance. They all are of same length as **12**.

The green line represent Euclidian distance of length **$6\sqrt{2} \approx 8.48$** .

Minkowski Distance

- Generalisasi dari jarak Euclidean dan Manhattan.
- Jarak Manhattan antara 2 titik atau tuple, $x = (x_1, x_2, \dots, x_d)$ dan $y = (y_1, y_2, \dots, y_n)$ pada ruang dimensi d adalah

$$d_{min}(x, y) = \left(\sum_{i=1}^d |x_i - y_i|^p \right)^{\frac{1}{p}}, p \geq 1$$

- Jika $p = 1 \rightarrow$ Manhattan distance
- Jika $p = 2 \rightarrow$ Euclidean distance

Normalisasi Data

- Data di tiap atribut umumnya dinormalisasi terlebih dahulu.
 - untuk menghindari *overweighing* atribut dengan range data besar (misalnya, income) terhadap atribut dengan range data kecil (misalnya, IPK).
- Beberapa cara untuk normalisasi antara lain:
 - Min-max normalization
 - Z-score normalization

Min-Max Normalization

Data dinormalisasi ke range $[x_{min}, x_{max}]$ dengan cara:

$$x' = x_{min} + [(x_{max} - x_{min}) \times \frac{x - x_{min_data}}{(x_{max_data} - x_{min_data})}]$$

Dimana:

x adalah nilai data semula

x' adalah data x yang telah dinormalisasi

x_{min} dan x_{max} adalah nilai minimum dan maksimum *range* data normalisasi

x_{min_data} dan x_{max_data} adalah nilai minimum dan maksimum pada data *input*

Z-Score Normalization

Data dinormalisasi berdasarkan nilai mean dan standar deviasi dari atribut dengan cara

$$v' = \frac{v - \bar{A}}{\sigma_A}$$

dimana:

v adalah nilai atribut A

\bar{A} adalah nilai rata-rata atribut A

σ_A adalah standar deviasi dari atribut A

Categorical Variables

- Contoh categorical variables → warna : merah, kuning, hijau.
- Jarak Manhattan antara 2 objek x dan y yang dideskripsikan dengan categorical variables adalah:

$$d(x, y) = \frac{p - m}{p}$$

p adalah jumlah variabel

m adalah jumlah variabel yang bernilai sama pada objek x dan y

Object identifier	Color (categorical)	Shape (categorical)	Material (categorical)
A	Yellow	Triangle	Wood
B	Red	Rectangle	Paper
C	Green	Triangle	Wood
D	Yellow	Triangle	Wood

$$d(B, A) = \frac{3 - 0}{3} = 1$$

$$d(C, A) = \frac{3 - 2}{3} = 0,333$$

Ordinal Variables

- Hampir sama dengan categorical variables, namun nilai-nilainya memiliki urutan yang berarti.
- Contoh: Temperature → high, medium, low.

Ordinal Variables

- Misalkan f adalah sebuah variabel dari himpunan ordinal variables yang mendeskripsikan n objek.
- Jarak atau dissimilarity terkait atribut f adalah sebagai berikut:
 - Nilai atribut f pada objek ke- i adalah x_{if} dan atribut f memiliki M_f urutan state, merepresentasikan ranking $1, \dots, M_f$.
 - Ubah setiap nilai x_{if} dengan ranking yang bersesuaian, $r_{if} \in \{1, \dots, M_f\}$.
 - Ubah jarak setiap variabel ke range $[0.0, 1.0]$ dengan cara mengubah ranking r_{if} dari objek ke- i pada variabel f dengan cara
$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$
 - Dissimilarity antar objek x dan y diukur berdasarkan nilai z_{if} menggunakan interval-scaled variables.

Object identifier	Atribut 1 (ordinal)	Rank r_{if} ($M_f = 3$)	z_{if}
1	Low	1	$(1-1) / (3-1) = 0$
2	High	3	$(3-1) / (3-1) = 1$
3	Medium	2	$(2-1) / (3-1) = 0,5$
4	Low	1	$(1-1) / (3-1) = 0$

Back to K-Means Algorithm

Centroids and Objective Functions

- Updating centroid depends on :
 - The proximity measure for the data
 - The objective function (the goal of the clustering), for example:
minimize the **sum of the squared error (SSE)** of each point to its closest centroid.
- In other words, we calculate the error of each data point, i.e., its Euclidean distance to the closest centroid, and then compute the total sum of the squared errors.
- Given two different sets of clusters that are produced by two different runs of K-means, we prefer the one with the smallest SSE → this means that the prototypes (centroids) of this clustering area better representation of the points in their cluster.

Centroids and Objective Functions (cont.)

$$\text{SSE} = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} \text{dist}(\mathbf{c}_i, \mathbf{x})^2$$

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Symbol	Description
\mathbf{x}	An object.
C_i	The i^{th} cluster.
\mathbf{c}_i	The centroid of cluster C_i .
\mathbf{c}	The centroid of all points.
m_i	The number of objects in the i^{th} cluster.
m	The number of objects in the data set.
K	The number of clusters.

Algorithm 8.1 Basic K-means algorithm.

```
1: Select  $K$  points as initial centroids.  
2: repeat  
3:   Form  $K$  clusters by assigning each point to its closest centroid.  
4:   Recompute the centroid of each cluster.  
5: until Centroids do not change.
```

- Steps 3 and 4 of the K-means algorithm directly attempt to minimize objective function (i.e. SSE).
- However, the actions of K-means in Steps 3 and 4 are only guaranteed to find a **local minimum with respect to the SSE** since they are based on optimizing the SSE for specific choices of the centroids and clusters, rather than for all possible choices.

Centroids and Objective Functions (cont.)

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid
cosine	mean	Maximize sum of the cosine similarity of an object to its cluster centroid

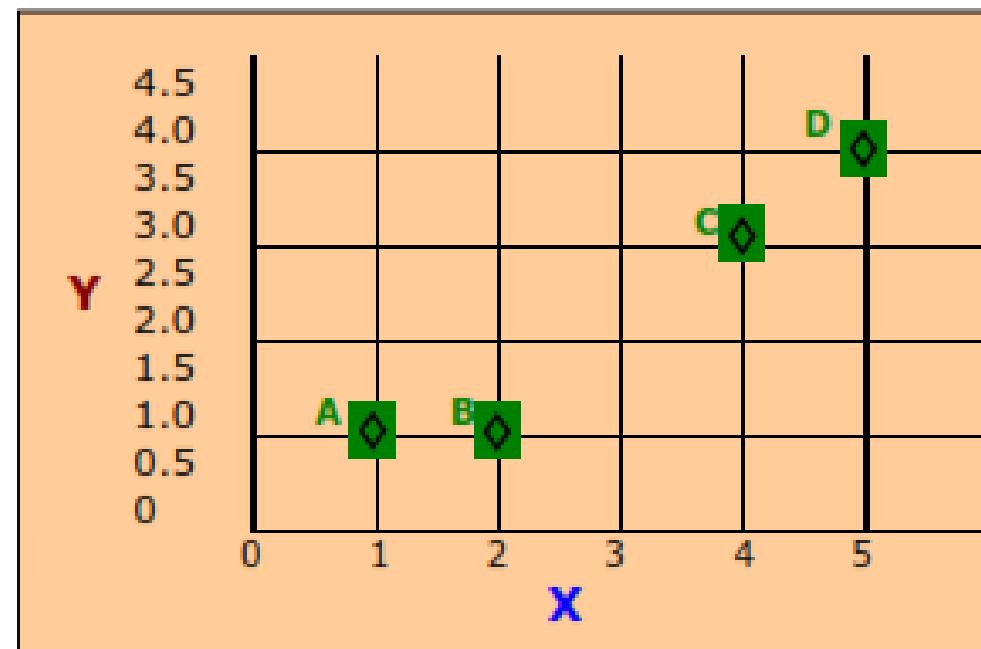
Example of K-Means Clustering

Example : K-Mean Clustering

Objects : 4 medicines as A, B, C, D.

Attributes : 2 as X is weight & Y is PH

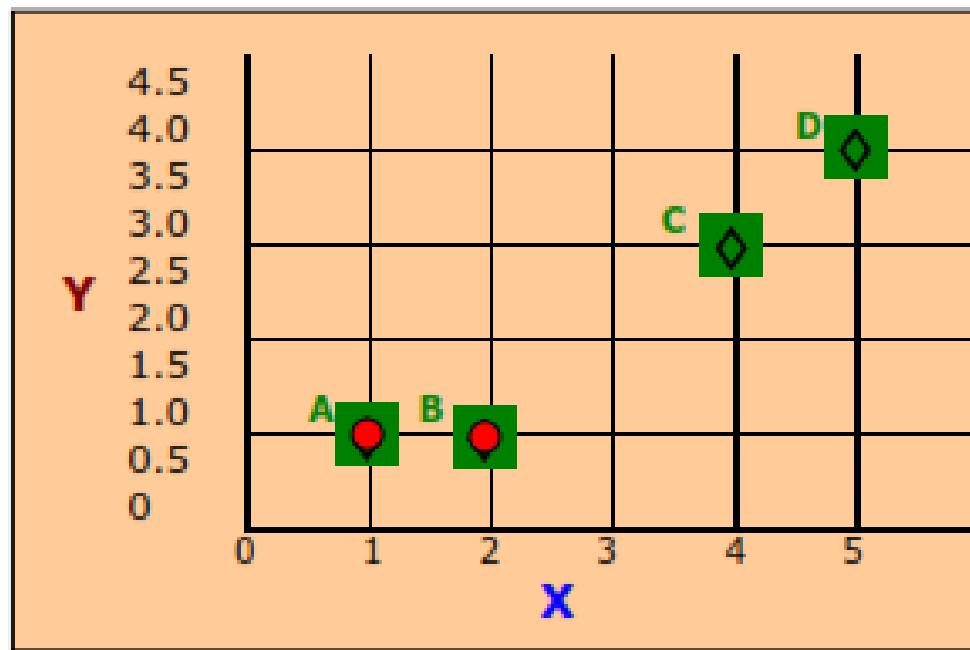
Objects	Attributes	
	X	Y
A	1	1
B	2	1
C	4	3
D	5	4



Initial value of centroids:

Suppose medicine **A** and medicine **B** be first centroids. If **P₁** and **P₂** denote coordinates of the centroids, then **P₁ = (1, 1)** and **P₂ = (2, 1)**.

Let centroid **P₁** be for cluster group-1
Let centroid **P₂** be for cluster group-2.



1. Iteration 0

(a) Objects Clusters centers stated before : Objects : A , B, C, D

Group-1 has center $P_1 = (1, 1)$;

Attributes : X and Y

Group-2 has center $P_2 = (2, 1)$;

	A	B	C	D
X	1	2	4	5
Y	1	1	3	4

(b) Calculate distances between cluster center to each object

- 1st, calculate the Euclidean distances from centroid P_1 to each point A, B, C, D. It is the 1st row of the distance matrix.
- 2nd, calculate the Euclidean distances from centroid P_2 to each point A, B, C, D. It is the 2nd row of the distance matrix.

The ways to calculate just two distance matrix elements D_{13} and D_{23} are :

$$D_{13} = \sqrt{(C_x - P_{1x})^2 + (C_y - P_{1y})^2} = \sqrt{(4 - 1)^2 + (3 - 1)^2} = 3.61$$

$$D_{23} = \sqrt{(C_x - P_{2x})^2 + (C_y - P_{2y})^2} = \sqrt{(4 - 2)^2 + (3 - 1)^2} = 2.83$$

Similarly calculate other elements $D_{11}, D_{12}, D_{14}, D_{21}, D_{22}, D_{24}$

(c) Distance matrix becomes

$$D^0 = \left\{ \begin{array}{cccc} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \\ A & B & C & D \end{array} \right\}$$

1st row indicates **group-1** cluster
2nd row indicates **group-2** cluster

(d) Objects clustering into groups:

Assign group to each object based on the minimum distance. Thus,

medicine **A** is assigned to **group 1**;

medicine **B** is assigned to **group 2**,

medicine **C** is assigned to **group 2**, and

medicine **D** is assigned to **group 2**.

Group Matrix : matrix element is **1** if the object is assigned to that group

$$G^0 = \left\{ \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ A & B & C & D \end{array} \right\}$$

1st row as **group 1**
2nd row as **group 2**

Objective function (SSE)

$$\begin{aligned} \text{SSE} &= D_{11} + D_{22} + D_{23} + D_{24} \\ &= 0 + 0 + 2.83 + 4.24 \\ &= 7.07 \end{aligned}$$

2. Iteration 1 :

The cluster groups have new members. Compute the new centroids of each group. **Repeat the process of iteration indicated below.**

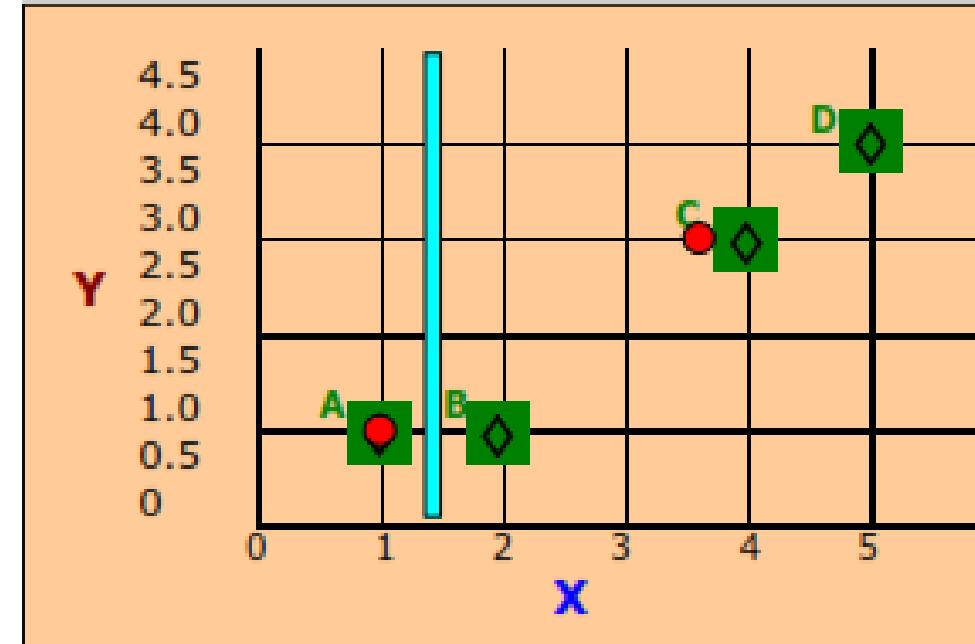
Group-1 has one member **A**, the centroid remains as $P_1 = (1, 1)$.

Group-2 now has 3 members **B**, **C**, **D**, so centroid is the average of their coordinates:

$$P_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3} \right)$$

$$= (11/3, 8/3)$$

$$= (3.67, 2.67)$$



(a) Objects Clusters centers stated above : Objects : **A , B, C, D**

Group-1 has center $P_1 = (1, 1)$;

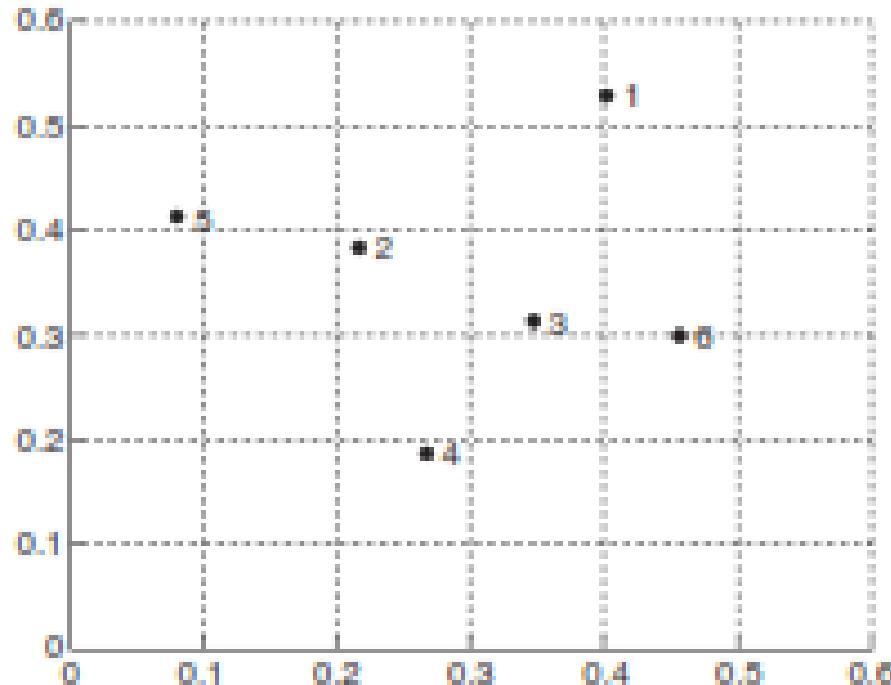
Group-2 has center $P_2 = (3.67, 2.67)$;

Attributes : **X** and **Y**

	A	B	C	D
X	1	2	4	5
Y	1	1	3	4

Tugas

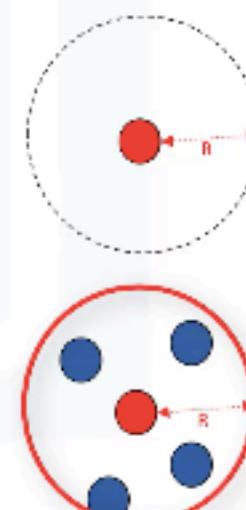
- Lakukanlah clustering dari sample data berikut. Cobalah dengan jumlah cluster K=2 dan K=3.
- Cobalah juga dengan beberapa inisialisasi centroid yang berbeda.
- Analisis hasil clustering berdasarkan hasil percobaan yang telah dilakukan.
- Berikan kesimpulan dari hasil yang diperoleh.



Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

What is DBSCAN?

- DBSCAN (**Density-Based Spatial Clustering of Applications with Noise**)
 - Is one of the most common clustering algorithms
 - Works based on density of objects
- R (Radius of neighborhood)
 - Radius (R) that if includes enough number of points within, we call it a dense area
- M (Min number of neighbors)
 - The minimum number of data points we want in a neighborhood to define a cluster



How DBSCAN Works

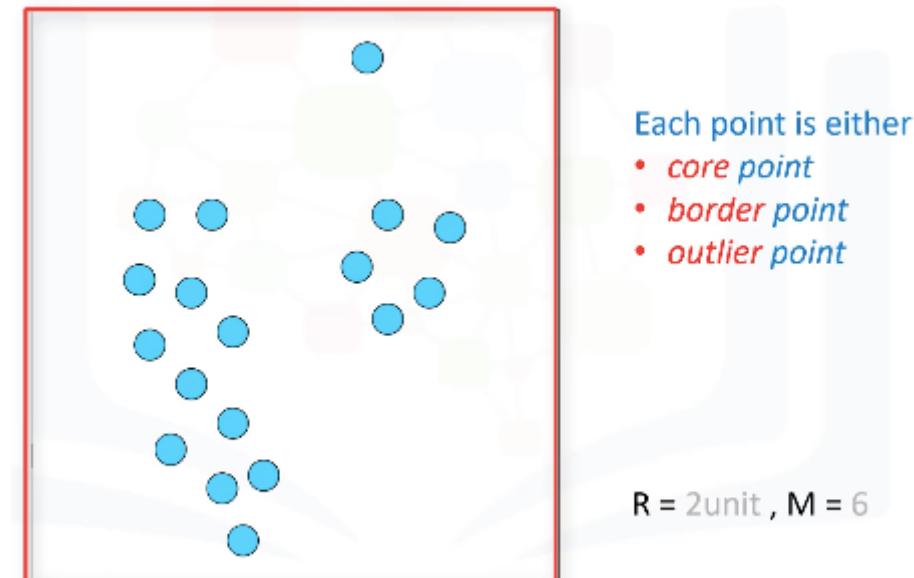
Let's define radius as 2 units. For the sake of simplicity, assume it as radius of 2 centimeters around a point of interest.

Also, let's set the minimum point, or M, to be 6 points including the point of interest.

To see how DBSCAN works, we have to determine the type of points.

Each point in our dataset can be either a core, border, or outlier point.

But the whole idea behind the DBSCAN algorithm is to visit each point, and find its type first. Then we group points as clusters based on their types.

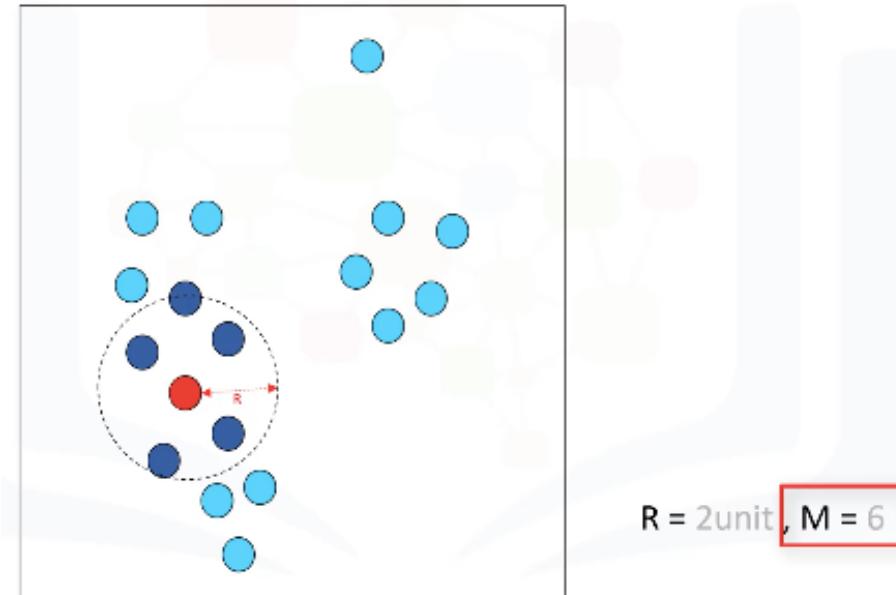


How DBSCAN Works

Let's pick a point randomly. First we check to see whether it's a core data point.

So, what is a core point?

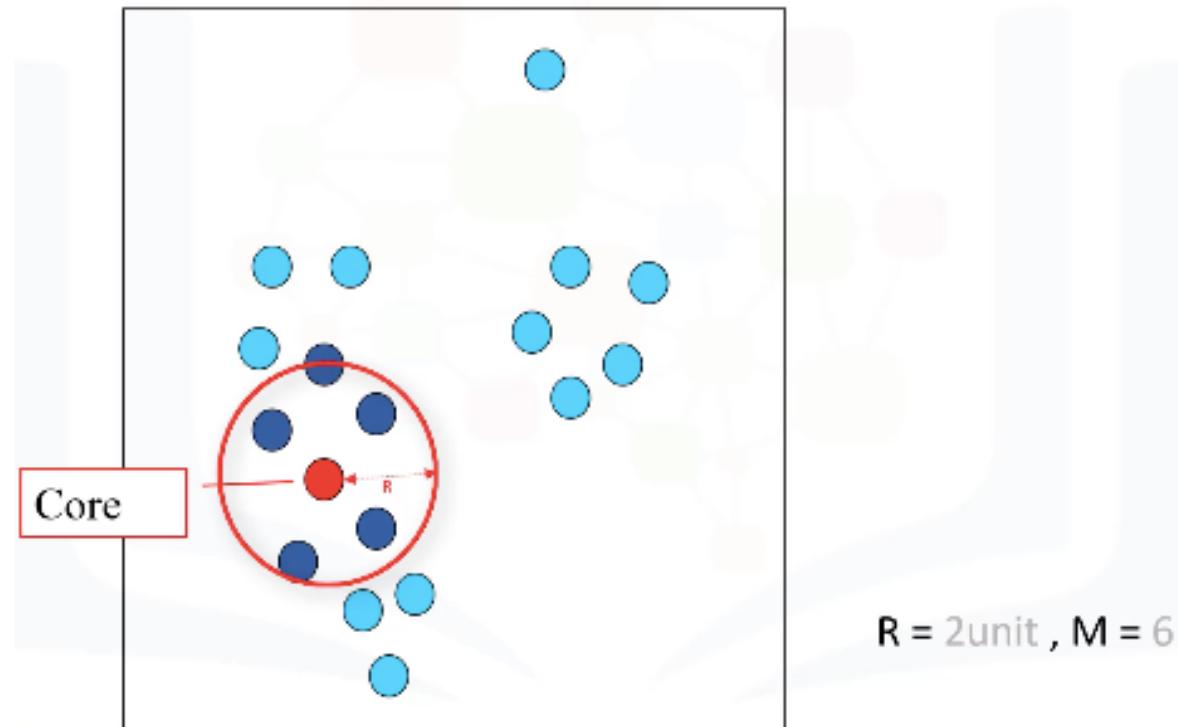
A data point is a core point if, within R-neighborhood of the point, there are at least M points.



How DBSCAN Works

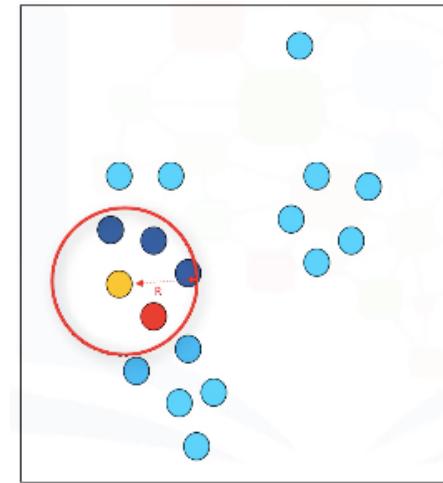
For example, as there are 6 points in the 2-centimeter neighbor of the red point, we mark this point as a core point.

Ok, what happens if it's NOT a core point?



How DBSCAN Works

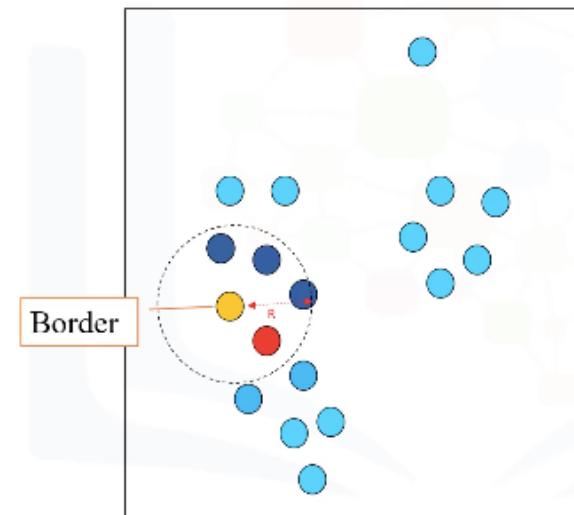
Let's look at another point. Is this point a core point? No.



R = 2unit , M = 6

As you can see, there are only 5 points in this neighborhood, including the yellow point.

So, what kind of point is this one? In fact, it is a "border" point.



R = 2unit , M = 6

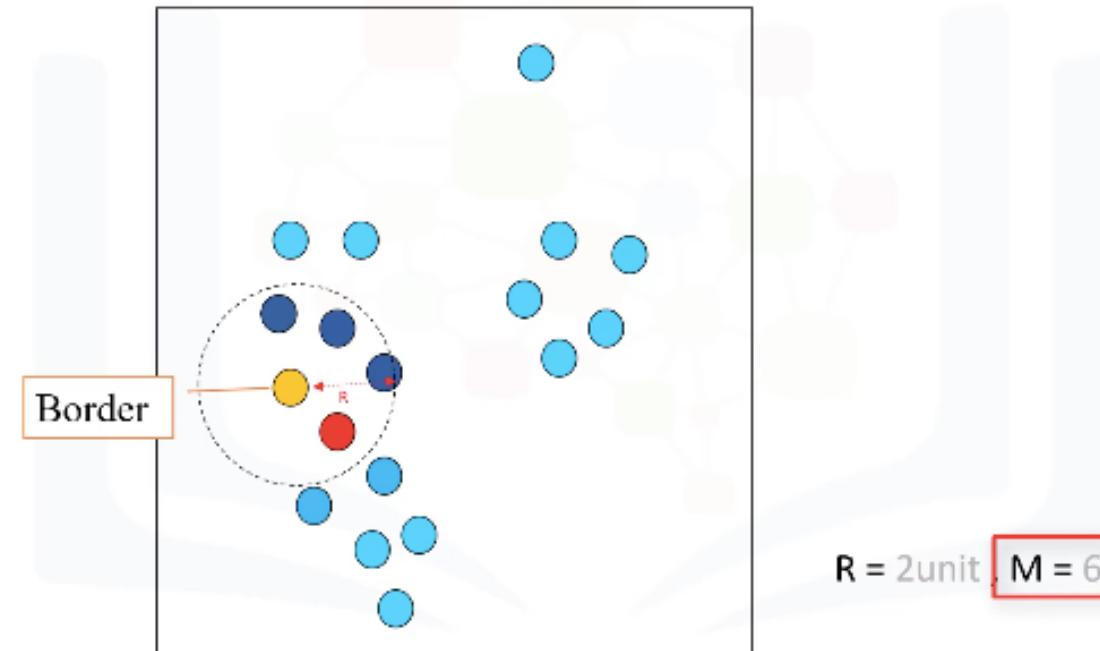
How DBSCAN Works

What is a border point? A data point is a BORDER point if:

- a. Its neighborhood contains less than M data points, or
- b. It is reachable from some core point.

Here, Reachability means it is within R-distance from a core point.

It means that even though the yellow point is within the 2-centimeter neighborhood of the red point, it is not by itself a core point, because it does not have at least 6 points in its neighborhood.



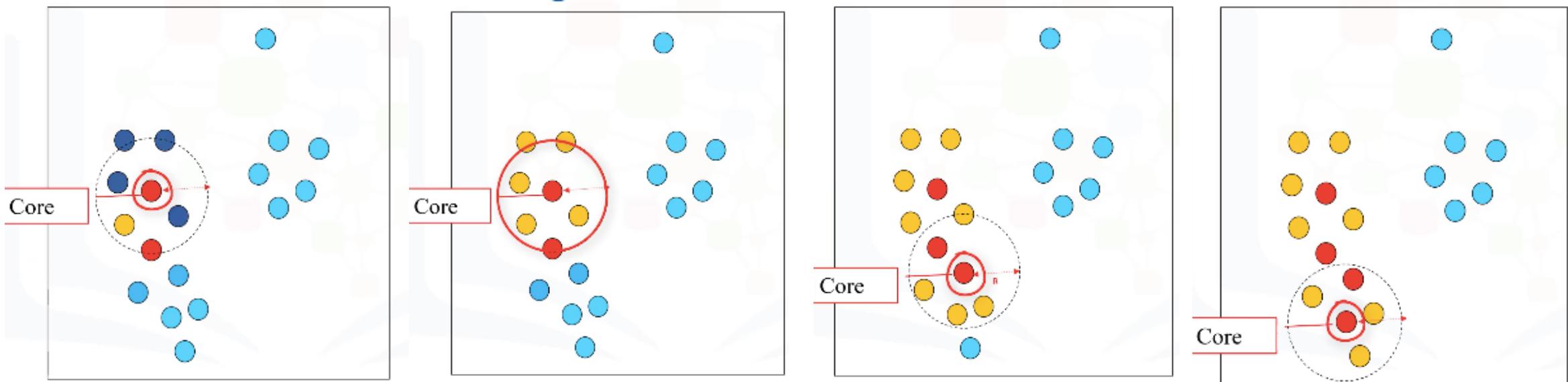
How DBSCAN Works

We continue with the next point.

As you can see it is also a core point.

And all points around it, which are not core points, are border points.

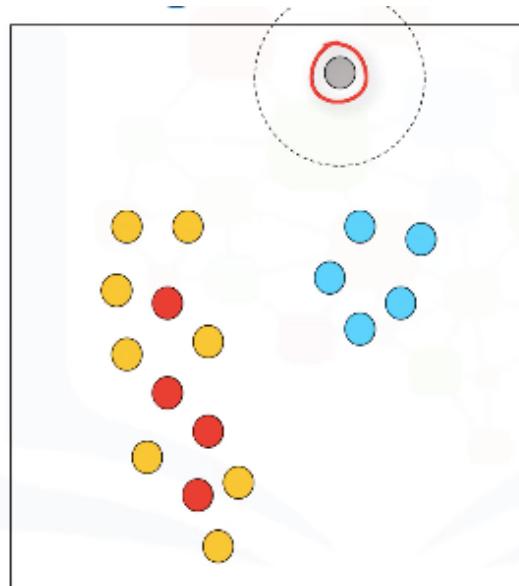
Next core point. And next core point.



How DBSCAN Works

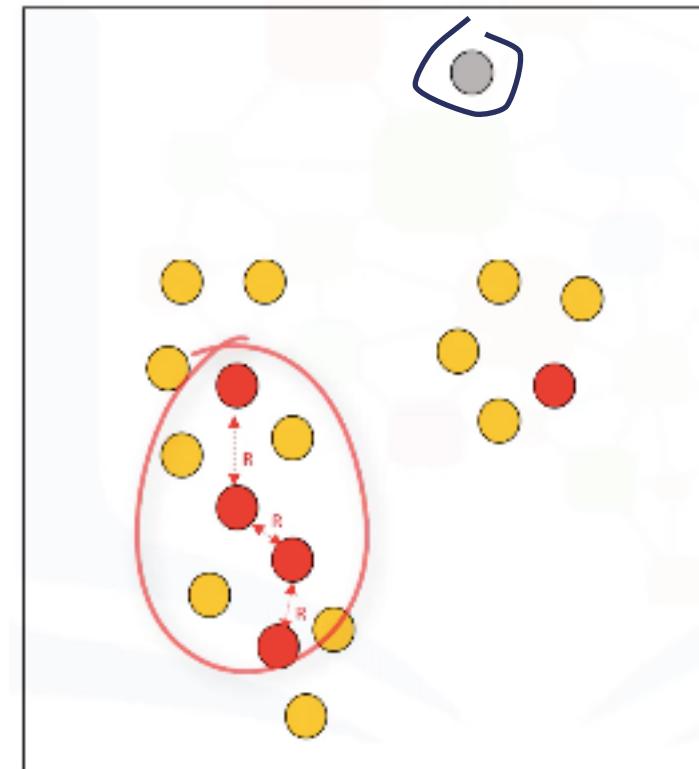
Let's take this point. You can see it is not a core point, nor is it a border point. So, we'd label it as an outlier.

What is an outlier? An outlier is a point that: Is not a core point, and also, is not close enough to be reachable from a core point.



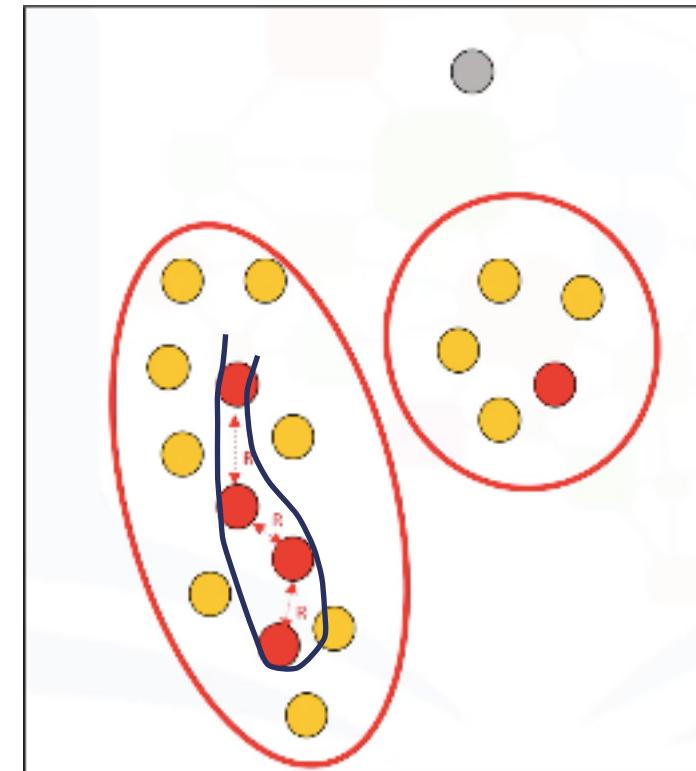
How DBSCAN Works

We continue and visit all the points in the dataset and label them as either Core, Border, or Outlier. The next step is to connect core points that are neighbors, and put them in the same cluster.



How DBSCAN Works

So, a cluster is formed as at least one core point, plus all reachable core points, plus all their borders.
It simply shapes all the clusters and finds outliers as well.



DBSCAN Advantages

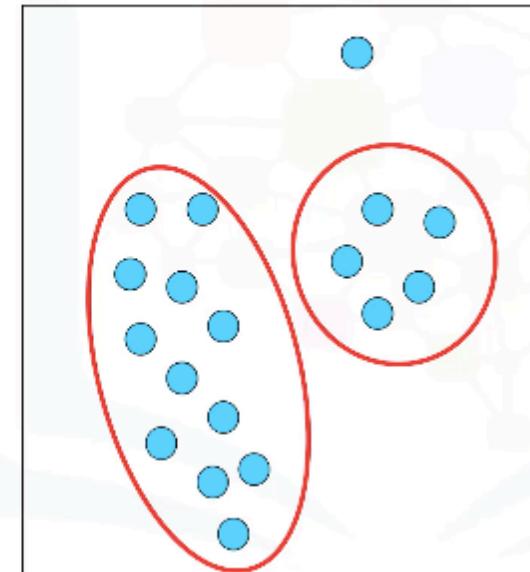
Let's review this one more time to see why DBSCAN is cool.

DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by a different cluster.

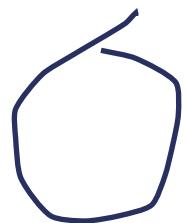
DBSCAN has a notion of noise, and is robust to outliers.

On top of that, DBSCAN makes it very practical for use in many real world problems because it does not require one to specify the number of clusters, such as K in k-Means.

Advantages of DBSCAN



1. Arbitrarily shaped clusters
2. Robust to outliers
3. Does not require specification of the number of clusters



Hierarchical Clustering

Overview

- Agglomerative Vs Divisive Clustering
- Contoh Agglomerative
- Latihan dan diskusi

Hierarchical Clustering

- Hierarchical Clustering adalah metode analisis kelompok yang berusaha untuk membangun sebuah hirarki kelompok data.
- Strategi pengelompokannya umumnya ada 2 jenis yaitu Agglomerative (Bottom-Up) dan Divisive (Top-Down). (Pada bagian ini akan dibatasi hanya menggunakan konsep Agglomerative).
- **Algoritma Agglomerative Hierarchical Clustering :**
 1. Hitung Matrik Jarak antar data.
 2. Ulangi langkah 3 dan 4 hingga hanya satu kelompok yang tersisa.
 3. Gabungkan dua kelompok terdekat berdasarkan parameter kedekatan yang ditentukan.
 4. Perbarui Matrik Jarak antar data untuk merepresentasikan kedekatan diantara kelompok baru dan kelompok yang masih tersisa.
 5. Selesai.

Rumus Umum

- Membentuk Matrik Jarak,
 - misal dengan Manhattan Distance :
 - atau menggunakan Euclidian Distance :
- Pengelompokan data secara hierarki:
 - Single linkage:
 - Complete Linkage:
 - Average Linkage:

$$D_{man}(x, y) = \sum_{j=1}^d |x_j - y_j|$$

$$D(x_2, x_1) = \sqrt{\sum_{j=1}^d |x_{2j} - x_{1j}|^2}$$

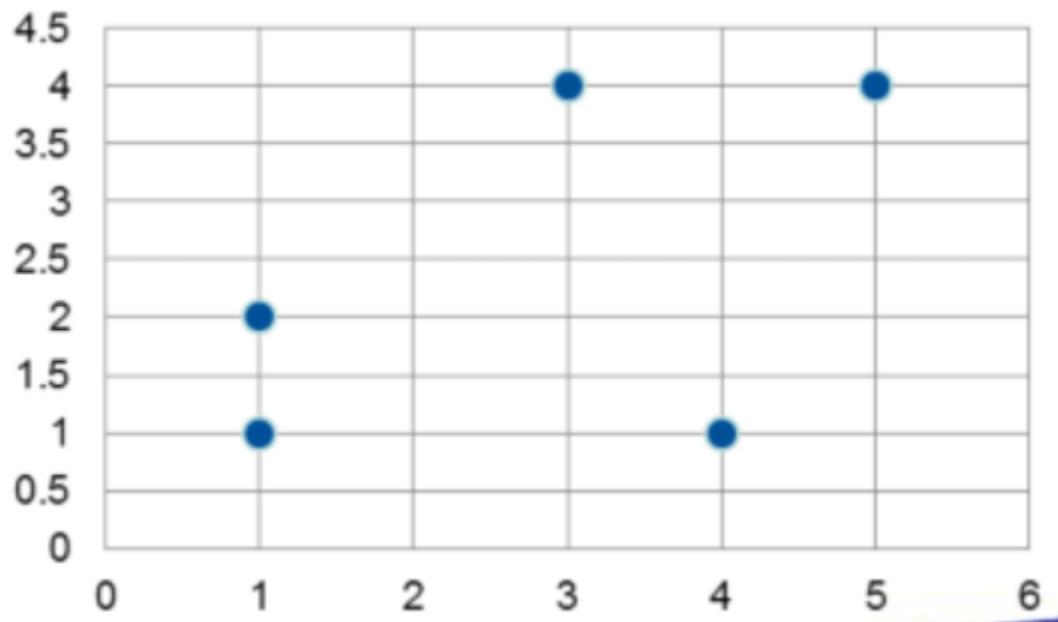
$$d_{uv} = \min\{d_{uv}\}, d_{uv} \in D$$

$$d_{uv} = \max\{d_{uv}\}, d_{uv} \in D$$

$$d_{uv} = average\{d_{uv}\}, d_{uv} \in D$$

Contoh Studi Kasus

Data	Fitur x	Fitur y
1	1	1
2	4	1
3	1	2
4	3	4
5	5	4



Selesaikan hierarchical clustering (agglomerative) dengan menggunakan min linkage, complete linkage , average linkage!

Single Linkage

Step 1. Hitung Matrik Jarak antar data.

	1	2	3	4	5
1	0				
2		0			
3			0		
4				0	
5					0

Data	Fitur x	Fitur y
1	1	1
2	4	1
3	1	2
4	3	4
5	5	4

- Jarak masing2 data
 - $D(1,2): 3$
 - $D(1,3):1$
 - $D(1,4):5$
 - $D(1,5):7$
 - $D(2,3):4$
 - $D(2,4):4$
 - $D(2,5):4$
 - $D(3,4):4$
 - $D(3,5):6$
 - $D(4,5):2$

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Iterasi 1: (langkah 3) penggabungan 2 jarak terdekat

Dman	1	2	3	4	5
1	0	3	■	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

- Dengan memperlakukan data sebagai kelompok, selanjutnya kita pilih jarak dua kelompok yang terkecil.
- terpilih kelompok 1 dan 3, sehingga kedua kelompok ini digabungkan.
- Langkah 4: perbarui matriks jarak

Single Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3	2	4	5
1,3	0	3	4	?
2		0	4	4
4			0	2
5				0

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)
 - $D((1,3),2)$: $\min(D(1,2), D(3,2)) : \text{Min}(3,4) = 3$
 - $D((1,3),4)$: $\min(D(1,4), D(3,4)) : \text{Min}(5,4) = 4$
 - $D((1,3),5)$: ?
- $D(2,4):4$
- $D(2,5):4$
- $D(3,4):4$
- $D(3,5):2$
- $D(4,5)$

Single Linkage

Step 3. gabungkan 2 kelompok data yg paling dekat

	1,3	2	4	5
1,3	0	3	4	6
2		0	4	4
4			0	2
5				0

Single Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3	2	4,5
1,3	0	3	4
2		0	?
4,5			0

- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)
 - $D((1,3),2)$: $\min(D(1,2), D(3,2)) : \text{Min}(3,4) = 3$
 - $D((1,3),(4,5))$: $\min(D(1,4), D(1,5), D(3,4), D(3,5)) : \text{Min}(5,7,4,6) = 4$
 - $D(2,(4,5))$:

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Single Linkage

Step 3. gabungkan 2 kelompok data yg paling dekat

	1,3	2	4,5
1,3	0	3	4
2		0	4
4,5			0

Single Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3,2	4,5
1,3,2	0	4
4,5	4	0

- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)
 - $D((1,3,2),(4,5))$:
- kelompok (132) dan (45) digabung untuk menjadi kelompok tunggal dari lima data, yaitu kelompok (13245) dengan jarak terdekat 4.

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Dendrogram

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

→

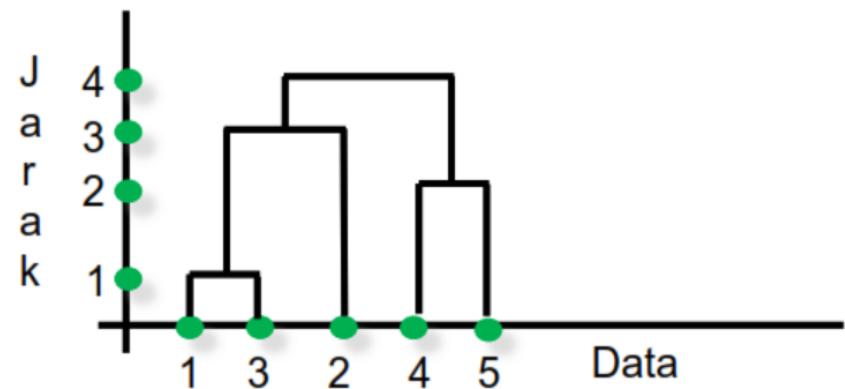
Dman	(13)	2	4	5
(13)	0	3	4	6
2	3	0	4	4
4	4	4	0	2
5	6	4	2	0

→

Dman	(45)	(13)	2
(45)	0	4	4
(13)	4	0	3
2	4	3	0

→

Dman	(132)	(45)
(132)	0	4
(45)	4	0



Complete Linkage

Step 1. Hitung Matrik Jarak antar data.

	1	2	3	4	5
1	0				
2		0			
3			0		
4				0	
5					0

Data	Fitur x	Fitur y
1	1	1
2	4	1
3	1	2
4	3	4
5	5	4

- Jarak masing2 data
 - $D(1,2): 3$
 - $D(1,3):1$
 - $D(1,4):5$
 - $D(1,5):7$
 - $D(2,3):4$
 - $D(2,4):4$
 - $D(2,5):4$
 - $D(3,4):4$
 - $D(3,5):6$
 - $D(4,5):2$

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Iterasi 1: (langkah 3) penggabungan 2 jarak terdekat

Dman	1	2	3	4	5
1	0	3	■	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

- Dengan memperlakukan data sebagai kelompok, selanjutnya kita pilih jarak dua kelompok yang terkecil.
- terpilih kelompok 1 dan 3, sehingga kedua kelompok ini digabungkan.
- Langkah 4: perbarui matriks jarak

Complete Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3	2	4	5
1,3	0	4	5	?
2		0	4	4
4			0	2
5				0

- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)

- $D((1,3),2)$: $\max(D(1,2), D(3,2)) : \text{Max}(3,4) = 4$
- $D((1,3),4)$: $\max(D(1,4), D(3,4)) : \text{Max}(5,4) = 5$
- $D((1,3),5)$: ?

- $D(2,4):4$
- $D(2,5):4$

- $D(3,4):4$
- $D(3,5):2$

- $D(4,5)$

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Complete Linkage

Step 3. gabungkan 2 kelompok data yg paling dekat

	1,3	2	4	5
1,3	0	4	4	6
2		0	4	4
4			0	2
5				0

Complete Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3	2	4,5
1,3	0	4	7
2		0	?
4,5			0

- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)
 - $D((1,3),2)$: $\max(D(1,2), D(3,2)) : \min(3,4) = 4$
 - $D((1,3),(4,5))$: $\max(D(1,4), D(1,5), D(3,4), D(3,5))$
 - $\max(5,7,4,6) = 7$
 - $D(2,(4,5))$:

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Complete Linkage

Step 3. gabungkan 2 kelompok data yg paling dekat

	1,3	2	4,5
1,3	0	4	7
2		0	4
4,5			0

Complete Linkage

Step 4. Hitung ulang Matrik Jarak antar data.

	1,3,2	4,5
1,3,2	0	7
4,5	7	0

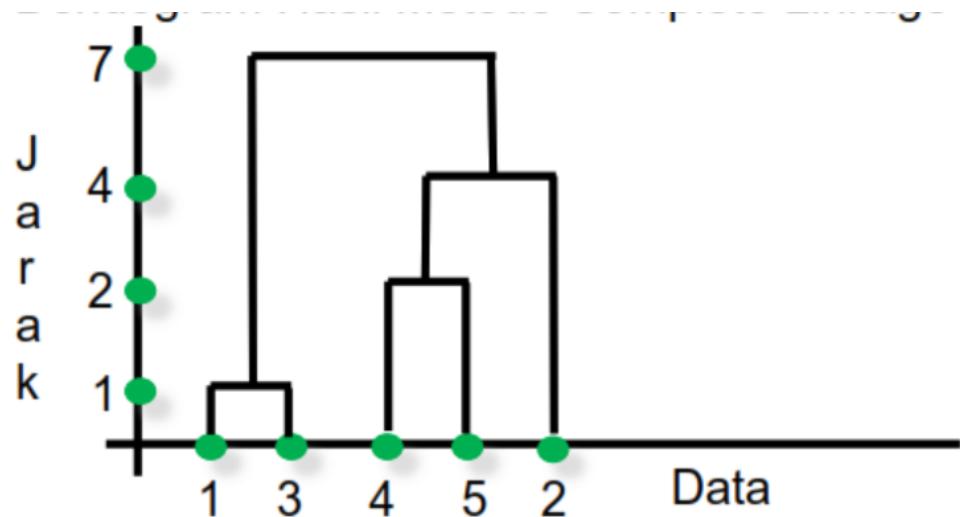
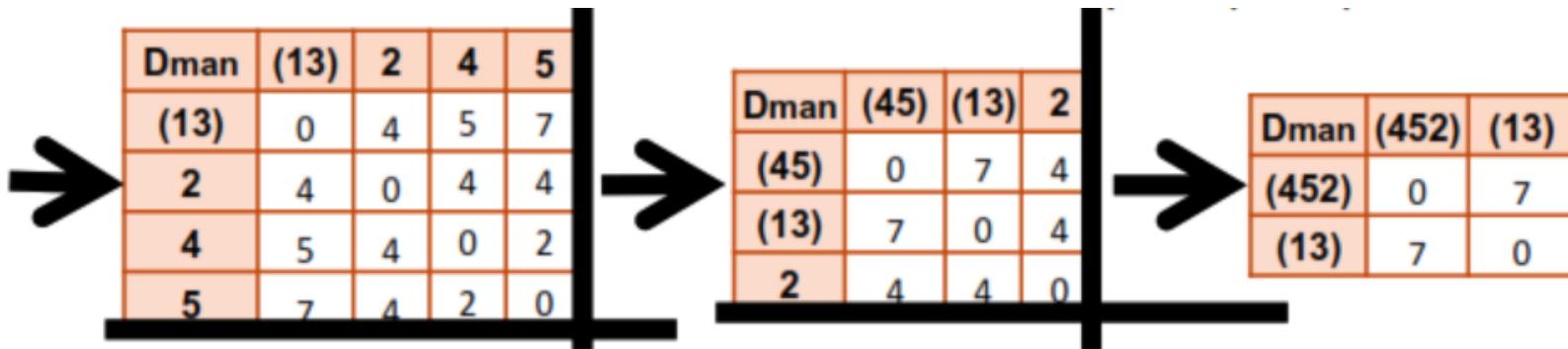
- Jarak masing2 data (untuk data cluster dengan individu, pakai single linkage)
 - $D((1,3,2),(4,5))$:
- kelompok (132) dan (45) digabung untuk menjadi kelompok tunggal dari lima data, yaitu kelompok (13245) dengan jarak terjauh 7.

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Dendogram

AlternatiV Dendogram

Dman	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0



Single Vs Complete



Expectation Maximization Algorithm

Overview

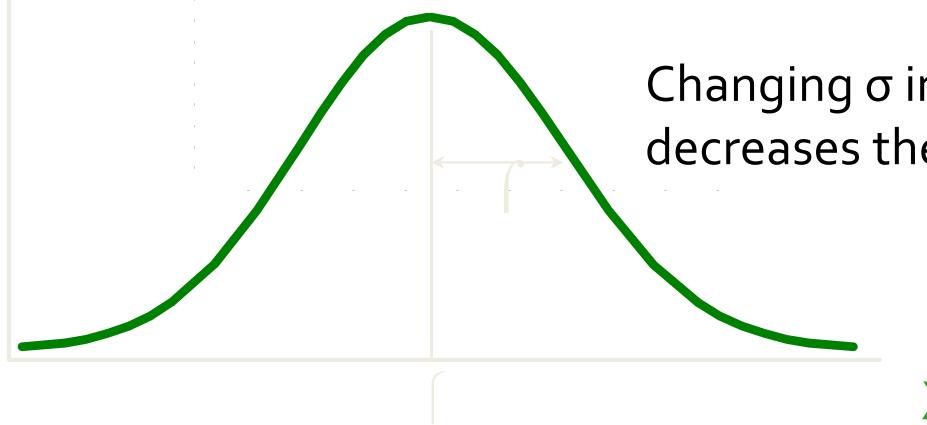
- Distribusi normal
- Probabilitas dan likelihood

Normal distribution

The Normal Distribution

$f(X)$

Changing μ shifts the distribution left or right.



Changing σ increases or decreases the spread.

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

The Normal Distribution: as mathematical function (pdf)

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

Note constants:

$\pi=3.14159$

$e=2.71828$

This is a bell shaped curve
with different centers and
spreads depending on μ
and σ

The Normal PDF

$$\int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 1$$

It's a probability function, so no matter what the values of μ and σ , must integrate to 1!

Normal distribution is defined by its mean and standard dev.

$$E(X)=\mu = \int_{-\infty}^{+\infty} x \frac{1}{\sigma \sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx$$

$$\text{Var}(X)=\sigma^2 = \int_{-\infty}^{+\infty} x^2 \frac{1}{\sigma \sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx - \mu^2$$

Standard Deviation(X)= σ

**The beauty of the normal curve:

No matter what μ and σ are, the area between $\mu-\sigma$ and $\mu+\sigma$ is about 68%; the area between $\mu-2\sigma$ and $\mu+2\sigma$ is about 95%; and the area between $\mu-3\sigma$ and $\mu+3\sigma$ is about 99.7%. Almost all values fall within 3 standard deviations.

68-95-99.7
Rule

y

x

68% of
the data

95% of the data

99.7% of the data

Copyright, Robert Niles, <http://www.robertniles.com/stats/stdev.shtml>

68-95-99.7

Rule
in Math
terms...

$$\int_{\mu-\sigma}^{\mu+\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .68$$

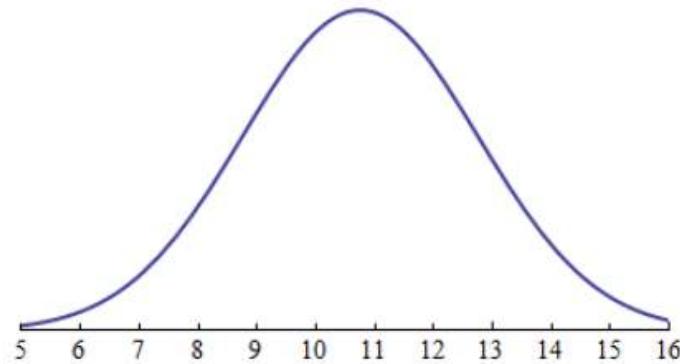
$$\int_{\mu-2\sigma}^{\mu+2\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .95$$

$$\int_{\mu-3\sigma}^{\mu+3\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .997$$

Probabilitas dan likelihood

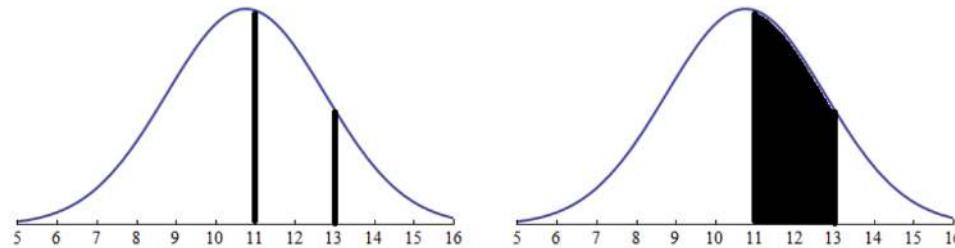
Probabilitas

- **Probability** is the measure of the likelihood that an event will occur.
- The basic idea is out of all given occurrences, what is the certainty that a specific event will occur?
- Let us say we have a normal distribution graph of the average marks of students in a surprise test. (this concept will apply to all continuous distributions)



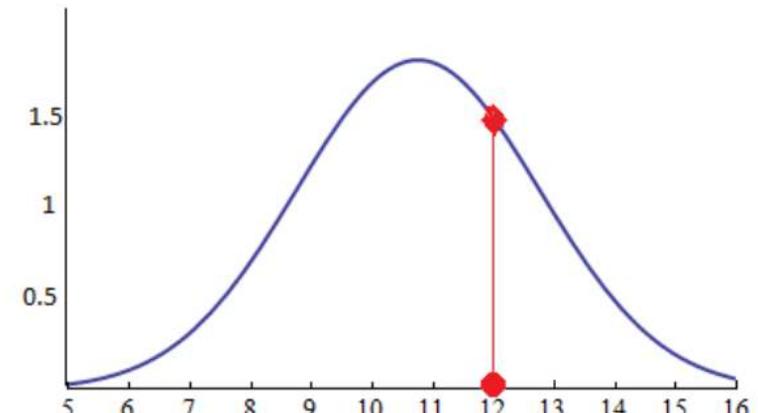
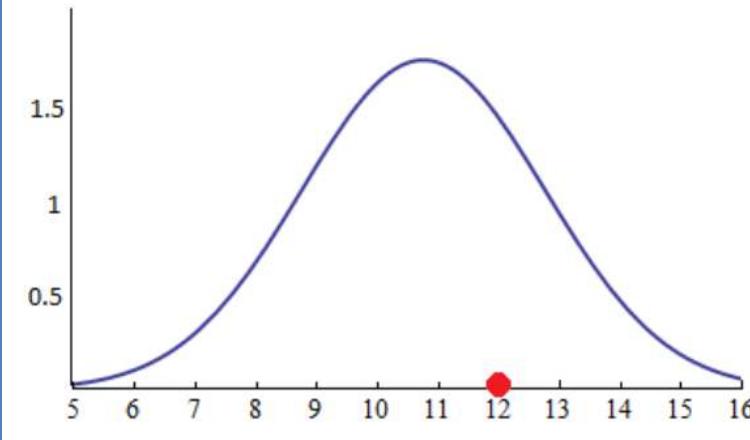
- Now, the probability that a randomly selected student will have marks between 11–13 marks is the area under the curve between those 2 points.
- mathematically,

$$P(\text{marks between 11 and 13 marks} \mid \text{mean}=11 \text{ and std} = 3) = 0.31$$



Likelihood

- likelihood function (often simply a likelihood) is a function of parameters within the parameter space that describes the probability of obtaining the observed data.
- $L(\text{mean}=11 \text{ and std} = 3 | \text{student scored } 12 \text{ marks}) = 1.48$



- Probabilities are the areas under fixed distribution

$P(\text{data}|\text{distribution})$

- Likelihoods are the y-axis values for fixed data points with distributions that can be moved.

$L(\text{distribution}|\text{data})$

- Finally, Probability quantifies anticipation (of outcome), likelihood quantifies trust (in the model).

Bayesian Theori

Bayes' Theorem

- Bayes' Theorem shows the relationship between a conditional probability and its inverse.
- i.e. it allows us to make an inference from
- the probability of a hypothesis given the evidence to
- the probability of that evidence given the hypothesis
- and vice versa

Bayes' Theorem

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

- $P(A)$ – the PRIOR PROBABILITY – represents your knowledge about A before you have gathered data.
- e.g. if 0.01 of a population has schizophrenia then the probability that a person drawn at random would have schizophrenia is 0.01

Bayes' Theorem

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

- $P(B|A)$ – the CONDITIONAL PROBABILITY – the probability of B, given A.
- e.g. you are trying to roll a total of 8 on two dice. What is the probability that you achieve this, given that the first die rolled a 6?

Bayes' Theorem

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

- So the theorem says:
- The probability of A given B is equal to the probability of B given A, times the prior probability of A, divided by the prior probability of B.

A Simple Example

• Mode of transport:

- Car
- Bus
- Train

• Suppose that Bob is late one day.

• His boss wishes to estimate the probability that he traveled to work that day by car.

• He does not know which mode of transportation Bob usually uses, so he gives a prior probability of 1 in 3 to each of the three possibilities.

$$P(\text{car}) = 0.33$$

$$P(\text{bus}) = 0.33$$

$$P(\text{train}) = 0.33$$

$$P(\text{late}|\text{car}) = 0.5$$

$$P(\text{late}|\text{bus}) = 0.20$$

$$P(\text{late}|\text{train}) = 0.01$$

Probability he is late:

50%

20%

1%

$$P(\text{car}|\text{late}) = \text{????}$$

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

$$P(\text{late}) =$$

$$P(\text{late}|\text{car}) * P(\text{car}) + P(\text{late}|\text{bus}) * P(\text{bus}) + P(\text{late}|\text{train}) * P(\text{train})$$

A Simple Example

- $P(A|B) = P(B|A) P(A) / P(B)$
- $P(\text{car}|\text{late}) = P(\text{late}|\text{car}) \times P(\text{car}) / P(\text{late})$
- $P(\text{late}|\text{car}) = 0.5$ (he will be late half the time he drives)
- $P(\text{car}) = 0.33$ (this is the boss' assumption)
- $P(\text{late}) = 0.5 \times 0.33 + 0.2 \times 0.33 + 0.01 \times 0.33$

(all the probabilities that he will be late added together)
- $$\begin{aligned}P(\text{car}|\text{late}) &= 0.5 \times 0.33 / 0.5 \times 0.33 + 0.2 \times 0.33 + 0.01 \times 0.33 \\&= 0.165 / 0.71 \times 0.33 \\&= 0.7042\end{aligned}$$

EM algorithm

EM algorithm

- The EM algorithm is an iterative optimization method that finds the maximum likelihood estimate (MLE) of parameters in problems where hidden/missing/latent variables are present

K-Means →
EM

- Boot Step:

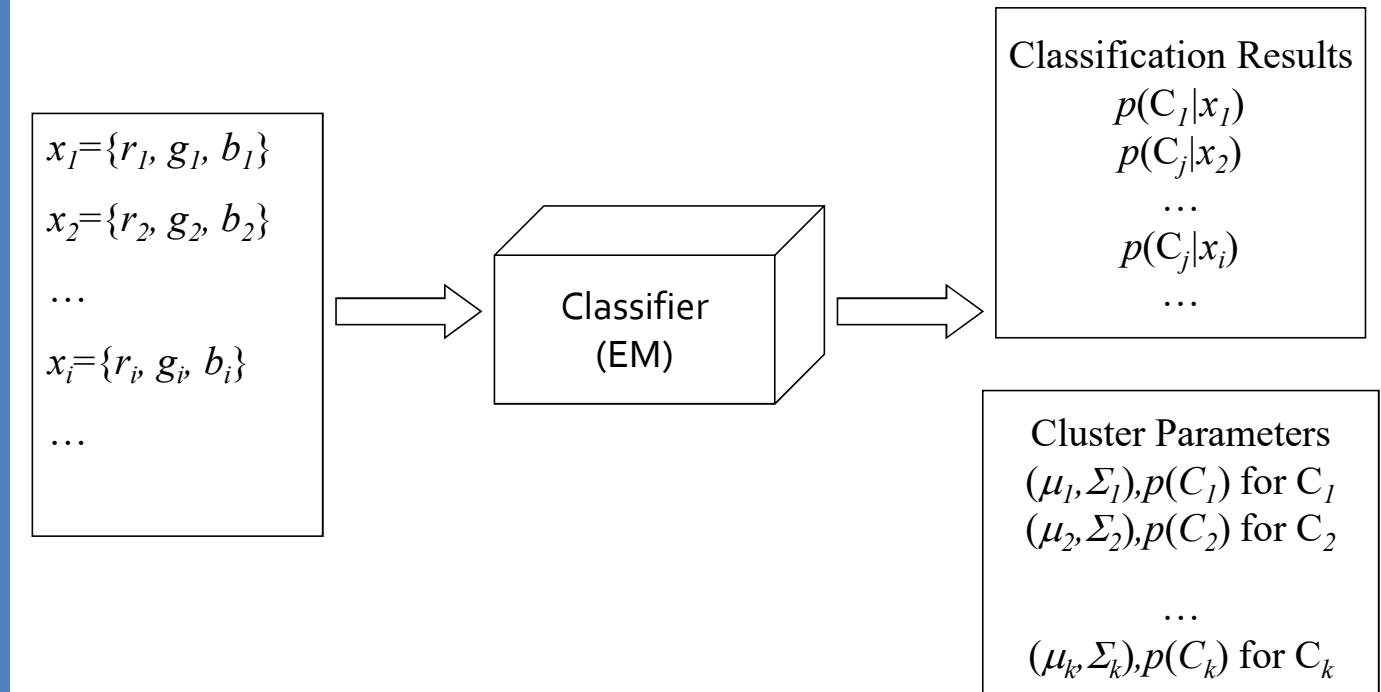
- Initialize K clusters: C_1, \dots, C_K
 (μ_j, Σ_j) and $P(C_j)$ for each cluster j .

- Iteration Step:

- Estimate the cluster of each data
 $p(C_j | x_i) \xrightarrow{\text{blue arrow}} \text{Expectation}$
- Re-estimate the cluster parameters

$(\mu_j, \Sigma_j), p(C_j)$ For each cluster $j \xrightarrow{\text{blue arrow}} \text{Maximization}$

EM Classifier



EM Classifier (Cont.)

Input (Known)

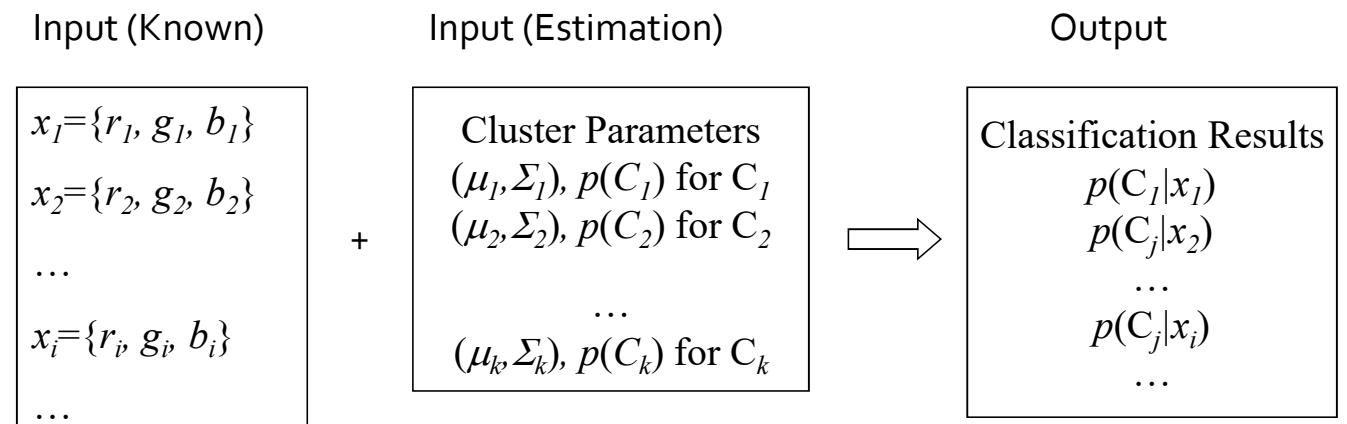
$$\begin{aligned}x_1 &= \{r_1, g_1, b_1\} \\x_2 &= \{r_2, g_2, b_2\} \\&\dots \\x_i &= \{r_i, g_i, b_i\} \\&\dots\end{aligned}$$

Output (Unknown)

$$\begin{aligned}&\text{Cluster Parameters} \\&(\mu_1, \Sigma_1), p(C_1) \text{ for } C_1 \\&(\mu_2, \Sigma_2), p(C_2) \text{ for } C_2 \\&\dots \\&(\mu_k, \Sigma_k), p(C_k) \text{ for } C_k\end{aligned}$$

$$\begin{aligned}&\text{Classification Results} \\&p(C_1|x_1) \\&p(C_j|x_2) \\&\dots \\&p(C_j|x_i) \\&\dots\end{aligned}$$

Expectation Step



$$p(C_j | x_i) = \frac{p(x_i | C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i | C_j) \cdot p(C_j)}{\sum_j p(x_i | C_j) \cdot p(C_j)}$$

Maximization Step

Input (Known)

$$\begin{aligned}x_1 &= \{r_1, g_1, b_1\} \\x_2 &= \{r_2, g_2, b_2\} \\&\dots \\x_i &= \{r_i, g_i, b_i\} \\&\dots\end{aligned}$$

Input (Estimation)

$$\begin{aligned}\text{Classification Results} \\ p(C_1|x_1) \\ p(C_j|x_2) \\ \dots \\ p(C_j|x_i) \\ \dots\end{aligned}$$

Output

$$\begin{aligned}\text{Cluster Parameters} \\ (\mu_1, \Sigma_1), p(C_1) \text{ for } C_1 \\ (\mu_2, \Sigma_2), p(C_2) \text{ for } C_2 \\ \dots \\ (\mu_k, \Sigma_k), p(C_k) \text{ for } C_k\end{aligned}$$

$$\mu_j = \frac{\sum_i p(C_j | x_i) \cdot x_i}{\sum_i p(C_j | x_i)}$$

$$\Sigma_j = \frac{\sum_i p(C_j | x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j | x_i)}$$

$$p(C_j) = \frac{\sum_i p(C_j | x_i)}{N}$$

EM Algorithm

- Boot Step:

- Initialize K clusters: C_1, \dots, C_K
 (μ_j, Σ_j) and $P(C_j)$ for each cluster j .

- Iteration Step:

- Expectation Step

$$p(C_j | x_i) = \frac{p(x_i | C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i | C_j) \cdot p(C_j)}{\sum_j p(x_i | C_j) \cdot p(C_j)}$$

- Maximization Step

$$\mu_j = \frac{\sum_i p(C_j | x_i) \cdot x_i}{\sum_i p(C_j | x_i)} \quad \Sigma_j = \frac{\sum_i p(C_j | x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j | x_i)} \quad p(C_j) = \frac{\sum_i p(C_j | x_i)}{N}$$

Data	NIlai
1	2
2	4
3	1
4	5
5	7

Example

- Initialization: $K=2$
 - $P(C_1)=0.5$
 - $P(C_2)=0.5$
 - $\mu(C_1) = (1, 2)$
 - $\mu(C_2) = (3, 4)$
 - $\Sigma(C_1) = (1, 1)$
 - $\Sigma(C_2) = (2, 2)$

Data	Nilai
1	2
2	4
3	1
4	5
5	7

Example

- $P(C_1)=0.5$
- $P(C_2)=0.5$
- $\mu(C_1) = (2)$
- $\mu(C_2) = (5)$
- $\sum(C_1) = (1)$
- $\sum(C_2) = (2)$
- Expectation
 - $P(C_1|data_1) = P(data_1|C_1)*P(C_1)/P(data_1)$
 - $P(C_2|data_1) = P(data_1|C_2)*P(C_2)/P(data_1)$
 - $P(data_1|C_1) = 0.398$

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

Data	Nilai	P(x C1)	p(x C2)	P(C1 x)	P(C2 x)	my	sigma	Pc1	Pc2
1	2	0.3989423	0.021024	0.570458	0.046423	1	2	1	0.5
2	4	0.053991	0.155348	0.077203	0.343024	2	5	2	
3	1	0.2419707	0.003653	0.346	0.008067				
4	5	0.0044318	0.199471	0.006337	0.440452				
5	7	1.487E-06	0.073381	2.13E-06	0.162033				

Data	Nilai	P(x C1)	p(x C2)	P(C1 x1)	P(C2 x1)	my	sigma	Pc1	Pc2
1	2	0.548791	0.019931	0.599057	0.036724	1.827428	0.7119	0.2	0.2
2	4	0.0203588	0.209616	0.022224	0.386233	4.809504	1.538041		
3	1	0.3464651	0.002318	0.378199	0.00427				
4	5	0.0004768	0.256341	0.00052	0.472328				
5	7	3.867E-09	0.054513	4.22E-09	0.100444				

Data	Nilai	P(x C1)	p(x C2)	P(C1 x1)	P(C2 x1)	my	sigma	Pc1	Pc2
1	2	0.8940706	0.010643	0.606386	0.015714	1.66781	0.386906	0.2	0.2
2	4	0.0009134	0.272406	0.000619	0.402174	4.789857	1.104047		
3	1	0.5794402	0.000541	0.392994	0.000798				
4	5	6.047E-07	0.354191	4.1E-07	0.522919				
5	7	1.138E-16	0.039553	7.72E-17	0.058395				



CLUSTER EVALUATION



OVERVIEW

- Motivation
- Evaluation type
 - Unsupervised
 - Supervised
- Unsupervised Evaluation
 - Cohesion
 - Separation
- Supervised Clustering /External Measure



MOTIVATION



MOTIVASI

- Dalam supervised classification, evaluasi model klasifikasi yang dihasilkan adalah bagian integral dari proses pengembangan model klasifikasi, dan ada langkah-langkah evaluasi yang diterima dengan baik (misalnya, akurasi) dan prosedur (misalnya validasi silang).
- Akan tetapi, karena sifat dari clustering itu sendiri, cluster evaluation adalah sesuatu yang belum banyak dikembangkan atau banyak dipakai dalam analisa cluster

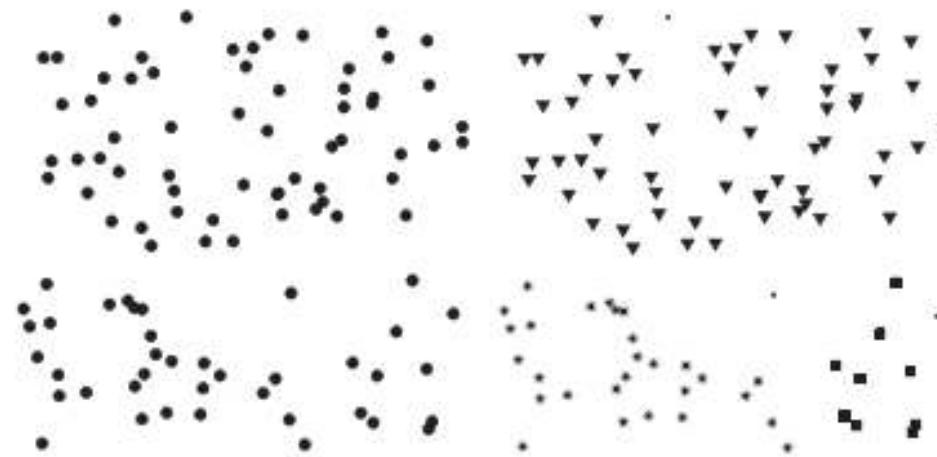
Motivasi

- Masing-masing algoritma mendefinisikan jenis clusternya sendiri, sehingga masing-masing clustering memerlukan evaluasi yg berbeda
 - For instance, K-means clusters might be evaluated in terms of the SSE, but for density-based clusters, which need not be globular, SSE would not work well at all.



MOTIVATION

- Cluster evaluation should be a part of any cluster analysis.
 - A key motivation is that almost every clustering algorithm will find clusters in a data set, even if that data set has no natural cluster structure.



(a) Original points.



(b) Three clusters found by DBSCAN.



(c) Three clusters found by K-means.

(d) Three clusters found by complete link.



OTHER MOTIVATIONS FOR CLUSTER VALIDATION

1. Menentukan kecenderungan pengelompokan satu set data, yaitu, membedakan apakah struktur non-acak benar-benar ada dalam data.
2. Menentukan jumlah cluster yang benar.
3. Mengevaluasi seberapa baik hasil analisis kluster sesuai dengan data tanpa referensi ke informasi eksternal.
4. Membandingkan hasil analisis kluster dengan hasil yang diketahui secara eksternal, seperti label kelas yang disediakan secara eksternal.
5. Membandingkan dua set cluster untuk menentukan mana yang lebih baik.

- Perhatikan bahwa item 1, 2, dan 3 tidak menggunakan informasi eksternal apa pun — supervised method— sedangkan item 4 membutuhkan informasi eksternal.
- Butir 5 dapat dilakukan dengan cara yang diawasi atau tidak diawasi.
- Perbedaan lebih lanjut dapat dibuat sehubungan dengan item 3, 4, dan 5: Apakah kita ingin mengevaluasi seluruh clustering atau hanya cluster individu?

EVALUATION



TYPES OF EVALUATION



VARIOUS TYPE OF CLUSTER VALIDITY

- Unsupervised
- Supervised
- Relative



UNSUPERVISED

- Measures the goodness of a clustering structure without respect to external information.
 - An example of this is the SSE.
- Unsupervised measures of cluster validity are often further divided into two classes:
 - measures of cluster cohesion (compactness, tightness), which determine how closely related the objects in a cluster are, and
 - Measures of cluster separation (isolation), which determine how distinct or well separated a cluster is from other clusters.
- Unsupervised measures are often called internal indices because they use only information present in the data set.



SUPERVISED

- Measures the extent to which the clustering structure discovered by a clustering algorithm matches some external structure.
- An example of a supervised index is entropy, which measures how well cluster labels match externally supplied class labels.
- Supervised measures are often called external indices because they use information not present in the data set.

RELATIVE

- Compares different clusterings or clusters.
- A relative cluster evaluation measure is a supervised or unsupervised evaluation measure that is used for the purpose of comparison.
- Thus, relative measures are not actually a separate type of cluster evaluation measure, but are instead a specific use of such measures.
- As an example, two K-means clusterings can be compared using either the SSE or entropy.



UNSUPERVISED

UNSUPERVISED CLUSTER EVALUATION USING COHESION AND SEPARATION

UNSUPERVISED CLUSTER EVALUATION USING THE PROXIMITY MATRIX.





UNSUPERVISED CLUSTER EVALUATION USING COHESION AND SEPARATION

- Many internal measures of cluster validity for partitional clustering schemes are based on the notions of cohesion or separation.
- In general, we can consider expressing overall cluster validity for a set of K clusters as a weighted sum of the validity of individual clusters,

$$\text{overall validity} = \sum_{i=1}^K w_i \text{ validity}(C_i).$$

- The validity function can be cohesion, separation, or some combination of these quantities.
- The weights will vary depending on the cluster validity measure. In some cases, the weights are simply 1 or the size of the cluster, while in other cases they reflect a more complicated property, such as the square root of the cohesion.

PROTOTYPE-BASED VIEW OF COHESION AND SEPARATION

- **Cluster Cohesion:** Measures how closely related are objects in a cluster
 - Example: SSE
- **Cluster Separation:** Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error

- Cohesion is measured by the within cluster sum of squares (SSE)

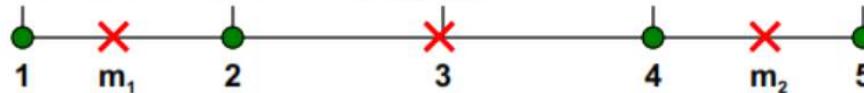
$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

- Separation is measured by the between cluster sum of squares

$$BSS = \sum |C_i| (m - m_i)^2$$

- Where $|C_i|$ is the size of cluster i

- Example: SSE
 - BSS + WSS = constant



K=1 cluster: $WSS = (1 - 3)^2 + (2 - 3)^2 + (4 - 3)^2 + (5 - 3)^2 = 10$

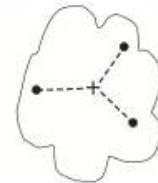
$$BSS = 4 \times (3 - 3)^2 = 0$$

$$Total = 10 + 0 = 10$$

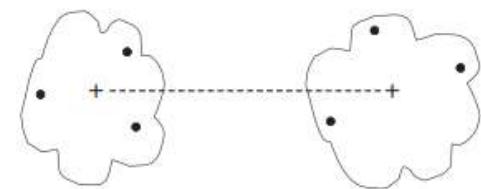
K=2 clusters: $WSS = (1 - 1.5)^2 + (2 - 1.5)^2 + (4 - 4.5)^2 + (5 - 4.5)^2 = 1$

$$BSS = 2 \times (3 - 1.5)^2 + 2 \times (4.5 - 3)^2 = 9$$

$$Total = 1 + 9 = 10$$



(a) Cohesion.



(b) Separation.

$$cohesion(C_i) = \sum_{\mathbf{x} \in C_i} proximity(\mathbf{x}, \mathbf{c}_i)$$

$$separation(C_i, C_j) = proximity(\mathbf{c}_i, \mathbf{c}_j)$$

$$separation(C_i) = proximity(\mathbf{c}_i, \mathbf{c})$$

- There are two measures for separation because the separation of cluster prototypes from an overall prototype is sometimes directly related to the separation of cluster prototypes from one another.



OVERALL MEASURES OF COHESION AND SEPARATION

- The previous definitions of cluster cohesion and separation gave us some simple and well defined measures of cluster validity that can be combined into an overall measure of cluster validity by using a weighted sum.
- However, we need to decide what weights to use.
- Not surprisingly, the weights used can vary widely, although typically they are some measure of cluster size.

- Note that any unsupervised measure of cluster validity potentially can be used as an objective function for a clustering algorithm and vice versa.
- The cluster evaluation measure I corresponds to traditional K-means and produces clusters that have good SSE values.
- The other measures produce clusters that are not as good with respect to SSE, but that are more optimal with respect to the specified cluster validity measure.



EVALUATING INDIVIDUAL CLUSTERS

- Many of these measures of cluster validity also can be used to evaluate individual clusters.
- For example, we can rank individual clusters according to their specific value of cluster validity, i.e., cluster cohesion or separation.
- A cluster that has a high value of cohesion may be considered better than a cluster that has a lower value.
- This information often can be used to improve the quality of a clustering.
- If, for example, a cluster is not very cohesive, then we may want to split it into several sub clusters.
- On the other hand, if two clusters are relatively cohesive, but not well separated, we may want to merge them into a single cluster.



EVALUATING INDIVIDUAL CLUSTERS AND OBJECTS

- We can also evaluate the objects within a cluster in terms of their contribution to the overall cohesion or separation of the cluster.
- Objects that contribute more to the cohesion and separation are near the “interior” of the cluster.
- Those objects for which the opposite is true are probably near the “edge” of the cluster.



THE SILHOUETTE COEFFICIENT

- The popular method of silhouette coefficients combines both cohesion and separation.
- The following steps explain how to compute the silhouette coefficient for an individual point (we use distances, but an analogous approach can be used for similarities)
 1. For the i^{th} object, calculate its average distance to all other objects in its cluster. Call this value a_i .
 2. For the i^{th} object and any cluster not containing the object, calculate the object's average distance to all the objects in the given cluster. Find the minimum such value with respect to all clusters; call this value b_i .
 3. For the i^{th} object, the silhouette coefficient is $s_i = (b_i - a_i) / \max(a_i, b_i)$.



The value of the silhouette coefficient can vary between -1 and 1 . A negative value is undesirable because this corresponds to a case in which a_i , the average distance to points in the cluster, is greater than b_i , the minimum average distance to points in another cluster. We want the silhouette coefficient to be positive ($a_i < b_i$), and for a_i to be as close to 0 as possible, since the coefficient assumes its maximum value of 1 when $a_i = 0$.



SC	Representasi
0.71 – 1.00	Baik
0.51 – 0.70	Sedang
0.26 – 0.50	Buruk
≤ 0.25	Berada di klaster lain

- We can compute the average silhouette coefficient of a cluster by simply taking the average of the silhouette coefficients of points belonging to the cluster.
- An overall measure of the goodness of a clustering can be obtained by computing the average silhouette coefficient of all points.



2. UNSUPERVISED CLUSTER EVALUATION USING THE PROXIMITY MATRIX

- Measuring Cluster Validity via Correlation
- Judging a Clustering Visually by Its Similarity Matrix



MEASURING CLUSTER VALIDITY VIA CORRELATION

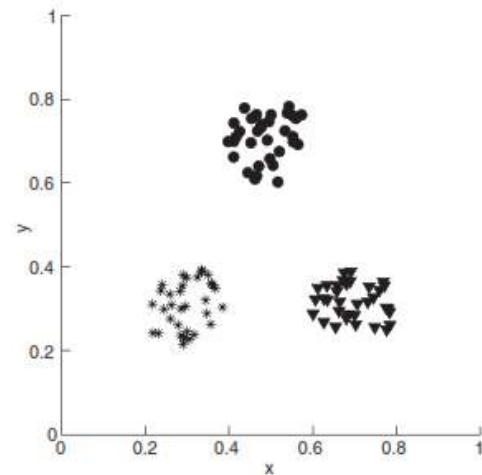
- If we are given the similarity matrix for a data set and the cluster labels from a cluster analysis of the data set, then we can evaluate the “goodness” of the clustering by looking at the correlation between the similarity matrix and an ideal version of the similarity matrix based on the cluster labels.
- (With minor changes, the following applies to proximity matrices, but for simplicity, we discuss only similarity matrices.)

- More specifically, an ideal cluster is one whose points have a similarity of 1 to all points in the cluster, and a similarity of 0 to all points in other clusters.
- Thus, if we sort the rows and columns of the similarity matrix so that all objects belonging to the same class are together, then an ideal similarity matrix has a block diagonal structure.
- In other words, the similarity is non-zero, i.e., 1, inside the blocks of the similarity matrix whose entries represent intra-cluster similarity, and 0 elsewhere.

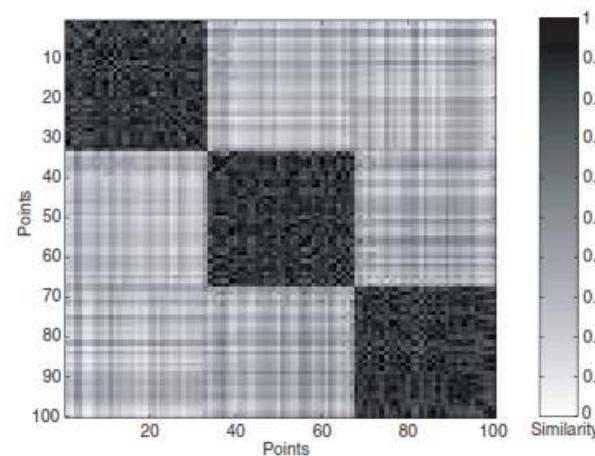
- High correlation between the ideal and actual similarity matrices indicates that the points that belong to the same cluster are close to each other, while low correlation indicates the opposite.
- Since the actual and ideal similarity matrices are symmetric, the correlation is calculated only among the $n(n-1)/2$ entries below or above the diagonal of the matrices.
- Consequently, this is not a good measure for many density- or contiguity-based clusters, because they are not globular and may be closely intertwined with other clusters.

JUDGING A CLUSTERING VISUALLY BY ITS SIMILARITY MATRIX

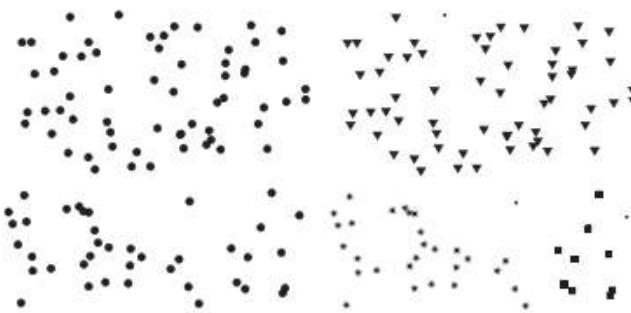
- In theory, if we have well-separated clusters, then the similarity matrix should be roughly block-diagonal.



(a) Well-separated clusters.



(b) Similarity matrix sorted by K-means cluster labels.



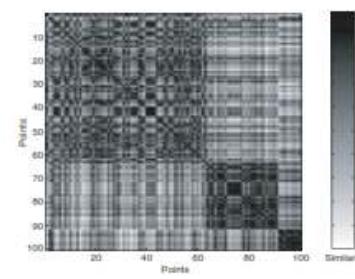
(a) Original points.

(b) Three clusters found by DBSCAN.

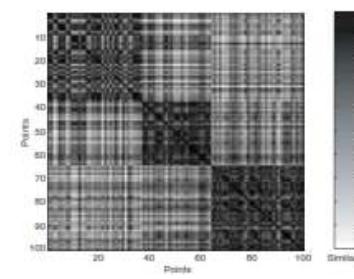


(c) Three clusters found by K-means.

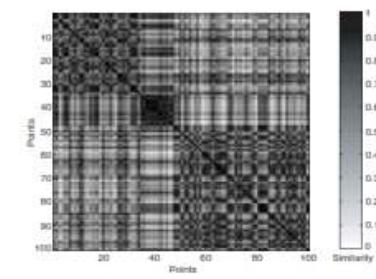
(d) Three clusters found by complete link.



(a) Similarity matrix sorted by DBSCAN cluster labels.



(b) Similarity matrix sorted by K-means cluster labels.



(c) Similarity matrix sorted by complete link cluster labels.

- This approach may seem hopelessly expensive for large data sets, since the computation of the proximity matrix takes $O(m^2)$ time, where m is the number of objects.
- Alternative: We can take a sample of data points from each cluster, compute the similarity between these points, and plot the result.
- It may be necessary to oversample small clusters and undersample large ones to obtain an adequate representation of all clusters.

DETERMINING THE CORRECT NUMBER OF CLUSTERS

- Various unsupervised cluster evaluation measures can be used to approximately determine the correct or natural number of clusters.
- Thus, we can try to find the natural number of clusters in a data set by looking for the number of clusters at which there is a knee, peak, or dip in the plot of the evaluation measure when it is plotted against the number of clusters.

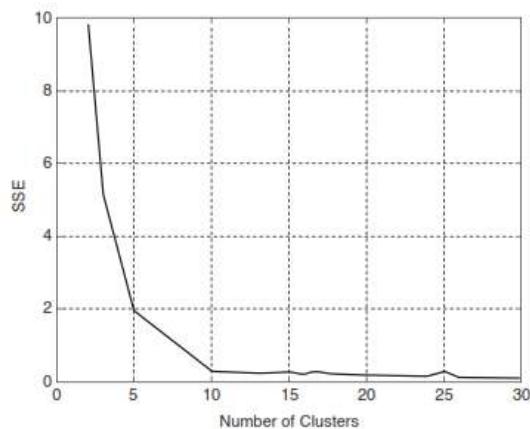
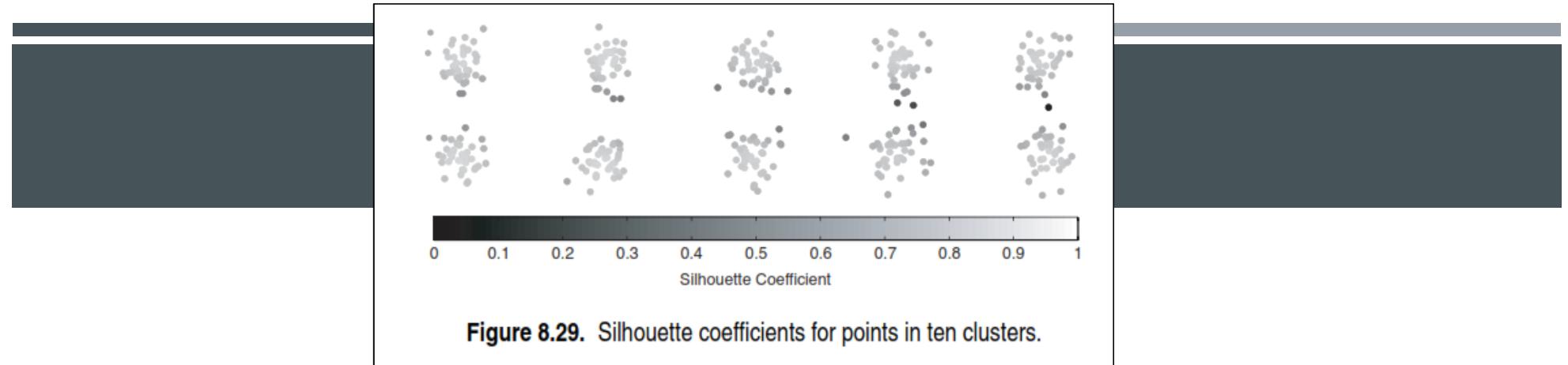


Figure 8.32. SSE versus number of clusters for the data of Figure 8.29.

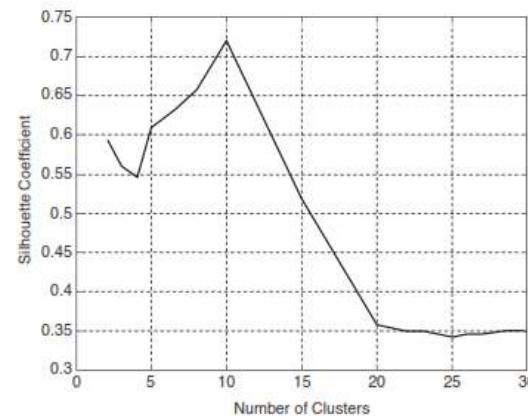


Figure 8.33. Average silhouette coefficient versus number of clusters for the data of Figure 8.29.



CLUSTERING TENDENCY

- One obvious way to determine if a data set has clusters is to try to cluster it.
- To address this issue, we could evaluate the resulting clusters and only claim that a data set has clusters if at least some of the clusters are of good quality.
- If the clusters are uniformly poor, then this may indeed indicate that there are no clusters in the data.



SUPERVISED METHOD





SUPERVISED MEASURES OF CLUSTER VALIDITY

- But why is this of interest? After all, if we have the class labels, then what is the point in performing a cluster analysis?
- Motivations for such an analysis are the comparison of clustering techniques with the “ground truth” or the evaluation of the extent to which a manual classification process can be automatically produced by cluster analysis.

CLASSIFICATION-ORIENTED MEASURE FOR CLUSTER VALIDITY

- In the case of classification, we measure the degree to which predicted class labels correspond to actual class labels, but for the measures just mentioned, nothing fundamental is changed by using cluster labels instead of predicted class labels.
- Examples:
 - Entropy
 - Purity
 - Precision
 - Recall

ENTROPY

The degree to which each cluster consists of objects of a single class.

$$p_{ij} = \frac{m_{ij}}{m_i} \quad e_i = - \sum_{j=1}^L p_{ij} \log_2 p_{ij} \quad e = \sum_{i=1}^K \frac{m_i}{m} e_i$$

Keterangan:

- p_{ij} : the probability that a member of cluster i belongs to class j
- m_{ij} : the number of objects of class j in cluster i
- m_i : the number of objects in cluster i
- e_i : entropy of cluster i
- e : the total entropy
- L is the number class; K is the number of cluster

PURITY

Another measure of the extent to which a cluster contains objects of a single class.

$$p_i = \max_j p_{ij} \quad \text{purity} = \sum_{i=1}^K \frac{m_i}{m} p_i$$

Keterangan:

p_i : purity of cluster i

purity : the total purity



PRECISION AND RECALL

J

Precision: The fraction of a cluster that consists of objects of a specified class.

The precision of cluster i with respect to class j is $precision(i, j) = p_{ij}$.

Recall: The extent to which a cluster contains all objects of a specified class.

The recall of cluster i with respect to class j is $recall(i, j) = m_{ij}/m_j$, where m_j is the number of objects in class j .

- A true positive (TP) decision assigns two similar documents to the same cluster, a true negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit.
- A (FP) decision assigns two dissimilar documents to the same cluster. A (FN) decision assigns two similar documents to different clusters.
- The *Rand index* () measures the percentage of decisions that are correct.
-

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Table 8.9. K-means clustering results for the *LA Times* document data set.

Cluster	Enter-tainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

Table 8.9. K-means clustering results for the *LA Times* document data set.

Cluster	Enter-tainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

- PI, metro= 506/677
- PI,entertainment=3/677
- PI,financial=677
- PI,foreign=40/677
- PI,National=96/677
- PISports=27/677
- EI=



- Example: consider cluster I and the Metro class of Table 8.9.
- The precision is $506/677 = 0.75$
- The recall is $506/943 = 0.26$

Feature Extraction

Principal Component Analysis (PCA)



PCA Toy Example

Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18



If each component is stored in a byte,
we need $18 = 3 \times 6$ bytes



PCA Toy Example

Looking closer, we can see that all the points are related geometrically: they are all the same point, scaled by a factor:

$$\begin{array}{c|c} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & = 1 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 3 \\ 6 \\ 9 \end{matrix} & = 3 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array} \quad \begin{array}{c|c} \begin{matrix} 2 \\ 4 \\ 6 \end{matrix} & = 2 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 5 \\ 10 \\ 15 \end{matrix} & = 5 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array} \quad \begin{array}{c|c} \begin{matrix} 4 \\ 8 \\ 12 \end{matrix} & = 4 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 6 \\ 12 \\ 18 \end{matrix} & = 6 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array}$$



PCA Toy Example

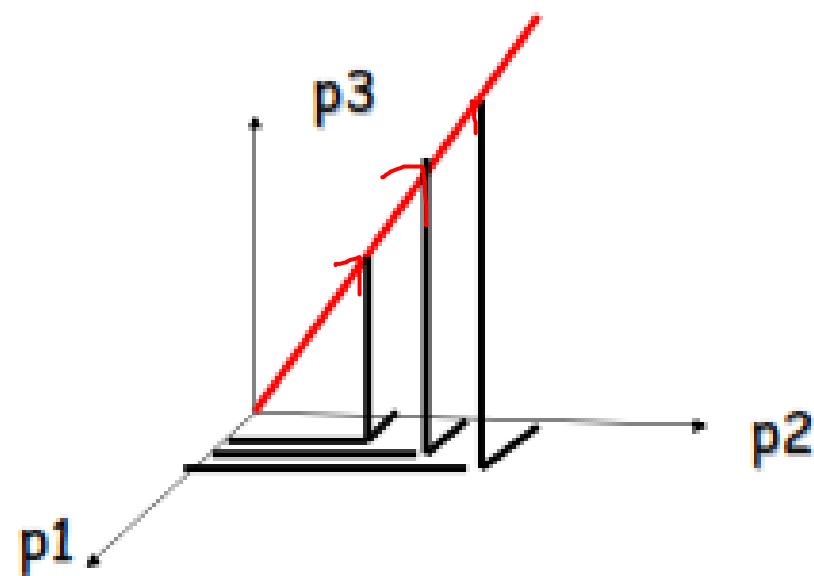
$$\begin{array}{c|c} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & = 1 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 3 \\ 6 \\ 9 \end{matrix} & = 3 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array} \quad \begin{array}{c|c} \begin{matrix} 2 \\ 4 \\ 6 \end{matrix} & = 2 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 5 \\ 10 \\ 15 \end{matrix} & = 5 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array} \quad \begin{array}{c|c} \begin{matrix} 4 \\ 8 \\ 12 \end{matrix} & = 4 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 6 \\ 12 \\ 18 \end{matrix} & = 6 * \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array}$$

They can be stored using only 9 bytes (50% savings!):
Store one point (3 bytes) + the multiplying constants (6 bytes)



Geometrical Interpretation

View each point in 3D space.

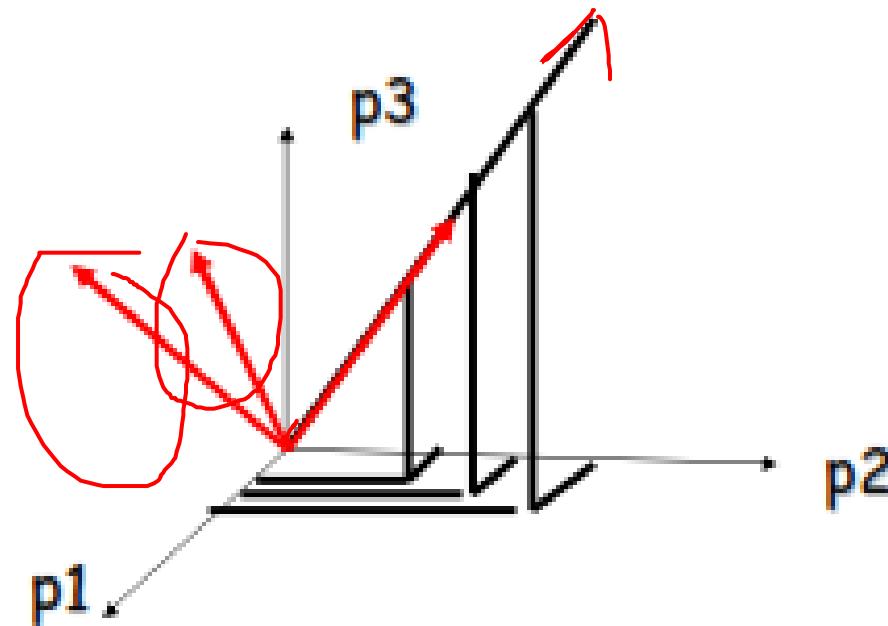


But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.



Geometrical Interpretation

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every point has only one non-zero coordinate: we only need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the points (6 bytes).

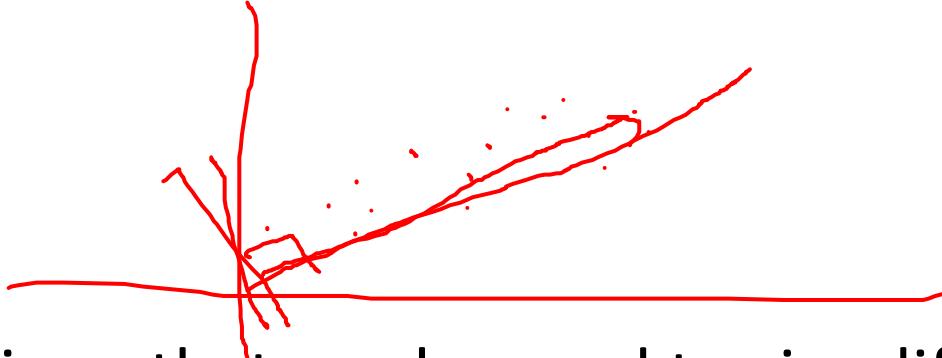


PCA

- Given a set of points, how do we know if they can be compressed like in the previous example?
- The answer is to look into the correlation between the points.
- The tool for doing this is called PCA.



PCA



- PCA is a technique that can be used to simplify a dataset.
- It is a linear transformation that chooses a new coordinate system for the data set such that:
 - greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component),
 - the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.



PCA

- By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset.
- This is the principal component.
- PCA is a useful statistical technique that has found application in:
 - fields such as face recognition and image compression
 - finding patterns in data of high dimension.



PCA - Algorithm

- Step 1: Get some data
- Step 2: Subtract the mean
- Step 3: Calculate the covariance matrix
- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix
- Step 5: Choosing components and forming a feature vector
- Step 6: Deriving the new data set
- Getting the old data back



Concept behind PCA

- Covariance Matrix
- Eigenvalue and Eigenvector



Covariance

- Variance and Covariance are a measure of the “spread” of a set of points around their center of mass (mean).
- Variance – measure of the deviation from the mean for points in one dimension e.g. heights.

$$var(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{(n - 1)}$$



Covariance

- Covariance is a measure of how much each of the dimensions vary from the mean with respect to each other.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions.
 - Example: the relationship between the number of hours studied and marks obtained.

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$

X *Y*



Covariance

- If we have a 3-dimensional data set $(\underline{X}, \underline{Y}, \underline{Z})$, then we could measure the 3 kinds of covariance:
 - $\text{Cov}(\underline{X}, \underline{Y})$
 - $\text{Cov}(\underline{X}, \underline{Z})$
 - $\text{Cov}(\underline{Y}, \underline{Z})$
- The covariance between one dimension and itself is the variance.
 - $\text{Cov}(\underline{X}, \underline{X}) = \text{var}(X)$



Example

- Example : 2 dimensional data set
 - H : number of hours studied for a subject
 - M : marks obtained in that subject
- Question: is there any relationship between the number of hours studied for a subject and the marks obtained in that subject?

Data	Hours(H)	Mark(M)
	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42



Solution

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54



Covariance

- What is the interpretation of covariance calculations?
- Example: 2 dimensional data set
 - H: number of hours studied for a subject
 - M: marks obtained in that subject
 - Covariance value : 104.54
 - *What does this value mean?*



Covariance

- Exact value is not as important as it's sign.
- A positive value of covariance indicates both dimensions increase or decrease together.
 - e.g. as the number of hours studied increases, the marks in that subject increase.
- A negative value indicates while one increases the other decreases , or vice-versa.
 - e.g. active social life at PSU vs performance in CS dept.
- If covariance is zero: the two dimensions are independent of each other.
 - e.g. heights of students vs the marks obtained in a subject



Covariance Matrix

- If we have a 3-dimensional data set (X, Y, Z), then we could measure the 3 kinds of covariance:
 - $\text{Cov}(X, Y)$
 - $\text{Cov}(X, Z)$
 - $\text{Cov}(Y, Z)$
- A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix
→ **covarian matrix.**



Covariance Matrix

- Covariance matrix between 3 dimensional dataset (X, Y, Z) will have 3 rows and 3 columns.

$$C = \begin{pmatrix} cov(x, x) & cov(x, y) & cov(x, z) \\ cov(y, x) & cov(y, y) & cov(y, z) \\ cov(z, x) & cov(z, y) & cov(z, z) \end{pmatrix}$$

- Note that covariance matrix is symmetrical because $cov(X, Y) = cov(Y, X)$



Exercise

The table below displays scores on math, English, and art tests for 5 students. Note that data from the table is represented in matrix **A**, where each column in the matrix shows scores on a test and each row shows scores for a student.

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

⇒
$$\begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

A

Given the data represented in matrix **A**, compute the variance of each test and the covariance between the tests.



Eigenvector

$$Av = \lambda v$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix} \rightarrow \lambda_1 \text{ } v_1$$

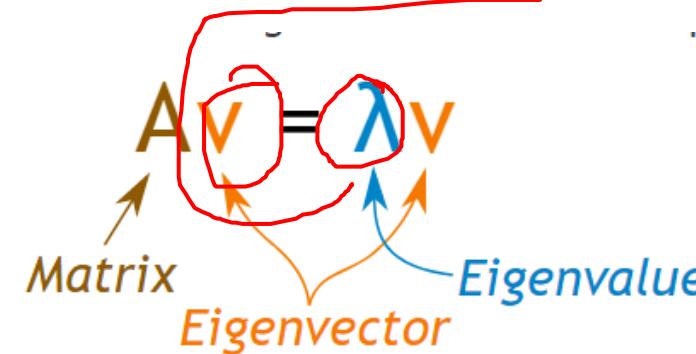
$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

A *v* *λ₁*

Figure 2.2: Example of one non-eigenvector and one eigenvector



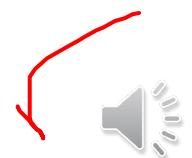
Eigenvector (cont.)



$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

- In the first example, the resulting vector is not an integer multiple of the original vector → the example vector is not eigenvector.
- In the second example, the example is exactly 4 times the vector we began with → the example vector is eigenvector.



Eigenvector (cont.)

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

- The vector $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ represents an arrow pointing from the origin $(0,0)$ to the point $(3,2)$.
- The other matrix, the square one, can be thought of as a transformation matrix.
- If you multiply this matrix on the left of a vector, the answer is another vector that is transformed from it's original position.
- Eigenvectors can only be found for square matrices. And, not every square matrix has eigenvectors.



Eigenvector (cont.)

- All the eigenvectors of a matrix are perpendicular (orthogonal), i.e. at right angles to each other, no matter how many dimensions you have.
 - This is important because it means that you can express the data in terms of these perpendicular eigenvectors, instead of expressing them in term of x and y axis.
- In order to keep eigenvectors standard, eigenvector is usually scaled to make it have a length of 1, so that all eigenvectors have the same length. (orthonormal)
 - The length of a vector doesn't affect whether it's an eigenvector or not, whereas the direction does.



Eigenvector (cont.)

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

is an eigenvector, and the length of that vector is

$$\sqrt{(3^2 + 2^2)} = \underline{\sqrt{13}}$$

so we divide the original vector by this much to make it have a length of 1.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix}$$



Eigenvalue

2x2

- Eigenvectors and eigenvalues always come in pairs.
- From the previous example, the eigenvalue is 4.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

- No matter what multiple of the eigenvector we took before we multiplied it by the square matrix, we would always get 4 times the scaled vector as our result



Eigenvalue and Eigenvector

- Jika \mathbf{A} adalah matriks $n \times n$, maka vector tak nol \mathbf{x} di dalam R^n dinamakan vector eigen dari \mathbf{A} jika \mathbf{Ax} adalah kelipatan scalar dari \mathbf{x} , yaitu:

$$\mathbf{Ax} = \lambda \mathbf{x}$$

untuk suatu scalar λ .

- Scalar λ disebut **nilai eigen** dari \mathbf{A} dan \mathbf{x} dikatakan **vector eigen** yang bersesuaian dengan λ .



Menghitung Nilai Eigen

- Untuk mencari nilai eigen matriks A yang berukuran $n \times n$ maka kita tuliskannya kembali sebagai :

$$Ax = \lambda x$$

atau

$$(A - \lambda I)x = 0$$

- Dan persamaan di atas akan mempunyai penyelesaian jika

$$|A - \lambda I| = 0$$



Menghitung Nilai Eigen

- Contoh:
- Tentukan nilai eigen dan vector eigen untuk matriks:
- $A = \begin{bmatrix} 5 & -1 \\ -2 & 4 \end{bmatrix}$
- Solusi:
- $\begin{vmatrix} 5 - \lambda & -1 \\ -2 & 4 - \lambda \end{vmatrix} = 0$



Mendapatkan vector eigen

- Diperoleh nilai eigen 6 dan 3.

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = 0$$

$$\begin{bmatrix} 5 - \lambda & -1 \\ -2 & 4 - \lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

- Untuk nilai eigen 6 diperoleh:

$$\begin{bmatrix} -1 & -1 \\ -2 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

- Atau

$$x_1 = -x_2$$



PCA - Algorithm

- Step 1: Get some data
- Step 2: Subtract the mean
- Step 3: Calculate the covariance matrix ✓
- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix ✓
- Step 5: Choosing components and forming a feature vector
- Step 6: Deriving the new data set
- Getting the old data back



Step 1: Get some data

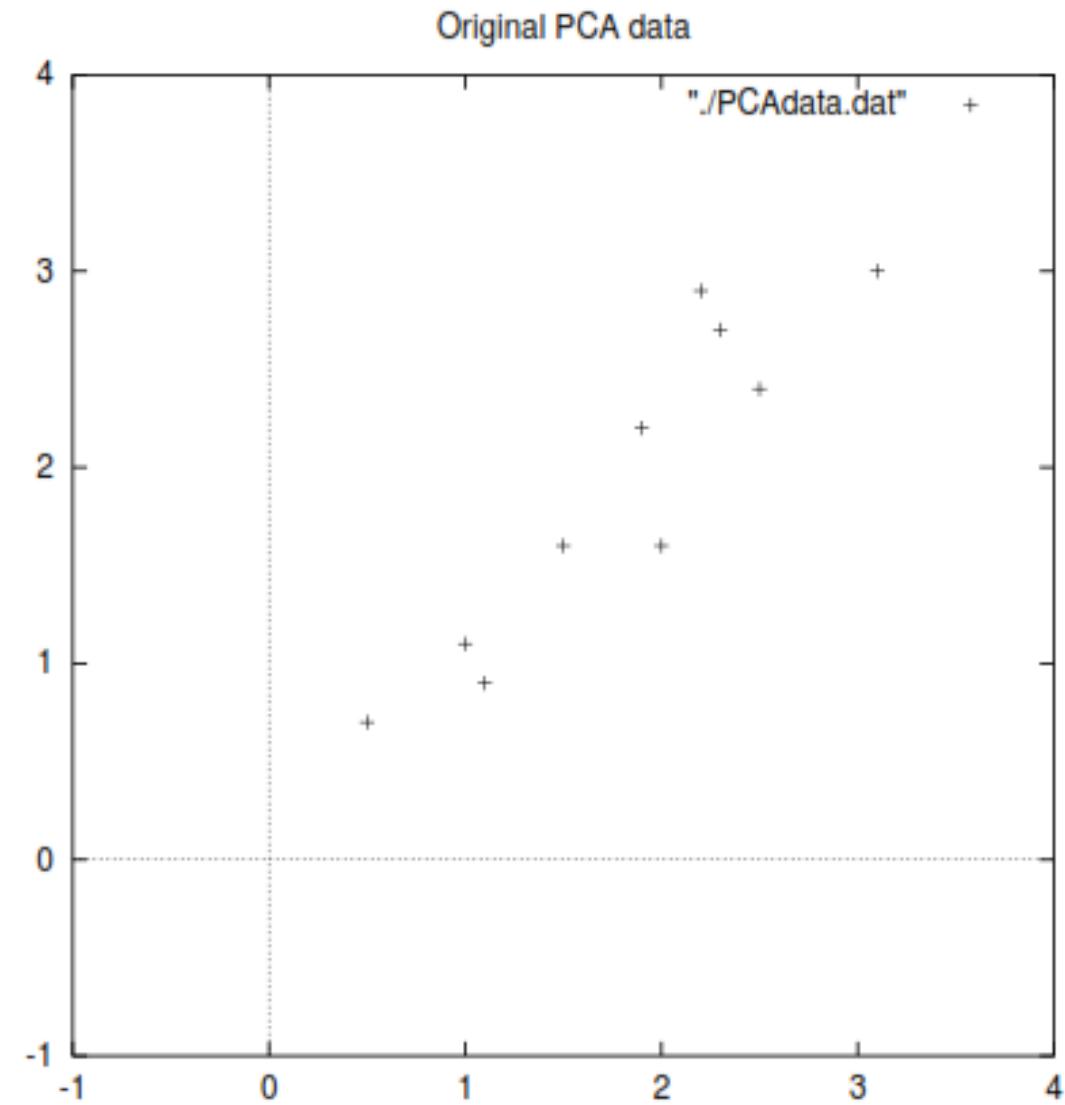
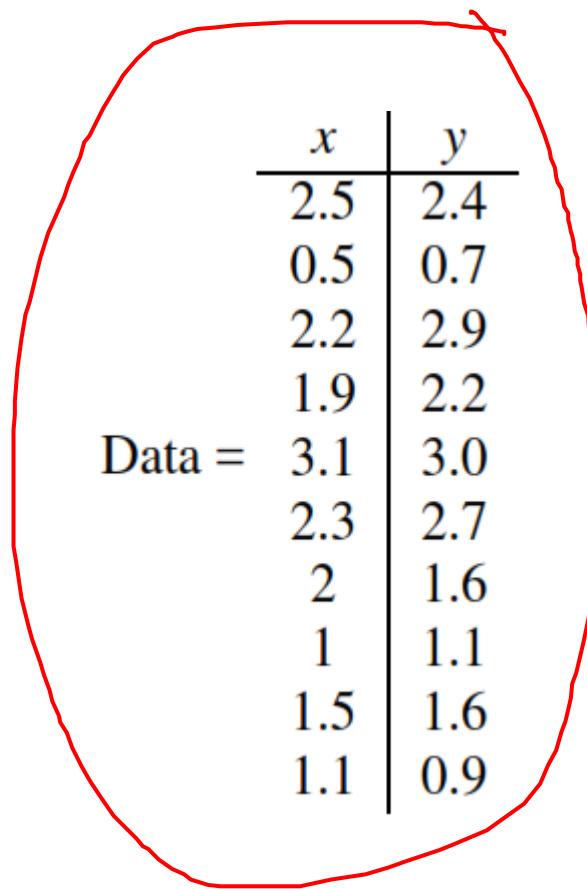


Figure 3.1: PCA example data,



Step 2: Subtract the mean

- For PCA to work properly, you have to subtract the mean from each of the data dimensions.
- The mean subtracted is the average across each dimension.
- So, all the x values have (the mean of the \bar{x} values of all the data points) subtracted, and all the y values have \bar{y} subtracted from them.
- This produces a data set whose mean is zero.

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

DataAdjust =



Step 3: Calculate the covariance matrix

- Since the data is 2 dimensional, the covariance matrix will be 2×2 .

$$cov = \begin{pmatrix} \text{Cov}(xx) & \text{Cov}(xy) \\ \text{Cov}(yx) & \text{Cov}(yy) \end{pmatrix}$$
$$\begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- So, since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.



Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

- Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix.

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

- It is important to notice that these eigenvectors are both unit eigenvectors ie. their lengths are both 1.



Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

- The plot of the data in Figure 3.2 show you how the data has quite a strong pattern.
- As expected from the covariance matrix, they two variables do indeed increase together.
- On top of the data, both the eigenvectors are plotted as well.
- They appear as diagonal dotted lines on the plot.
- They are perpendicular to each other.

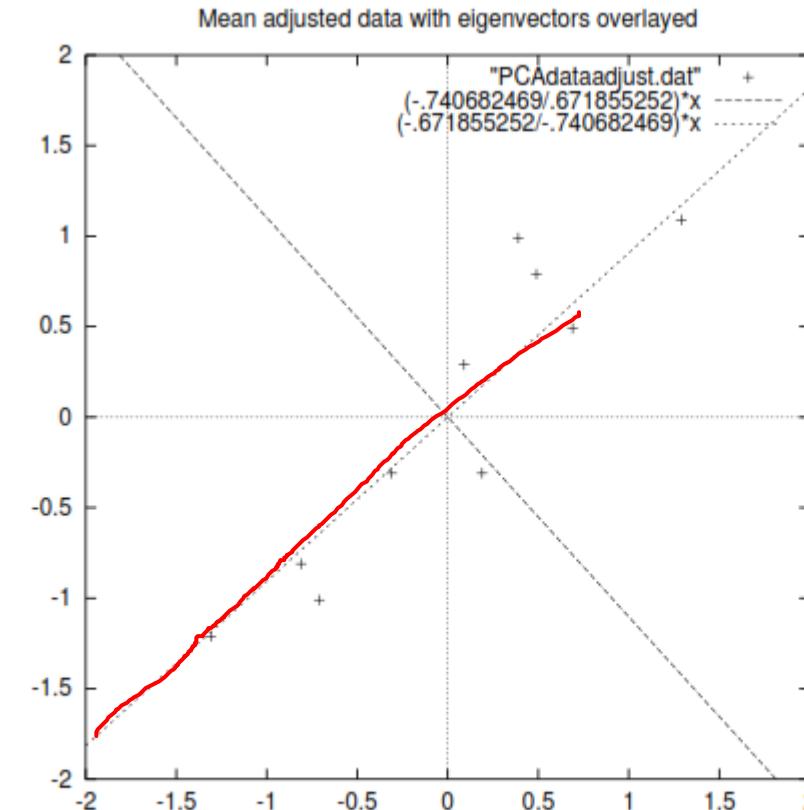


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.



Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

The eigenvectors provide us with information about the patterns in the data.

- The **first eigenvectors** goes through the middle of the points, like drawing a line of best fit. That eigenvector is showing us how these two data sets are related along that line.
- The **second eigenvector** gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

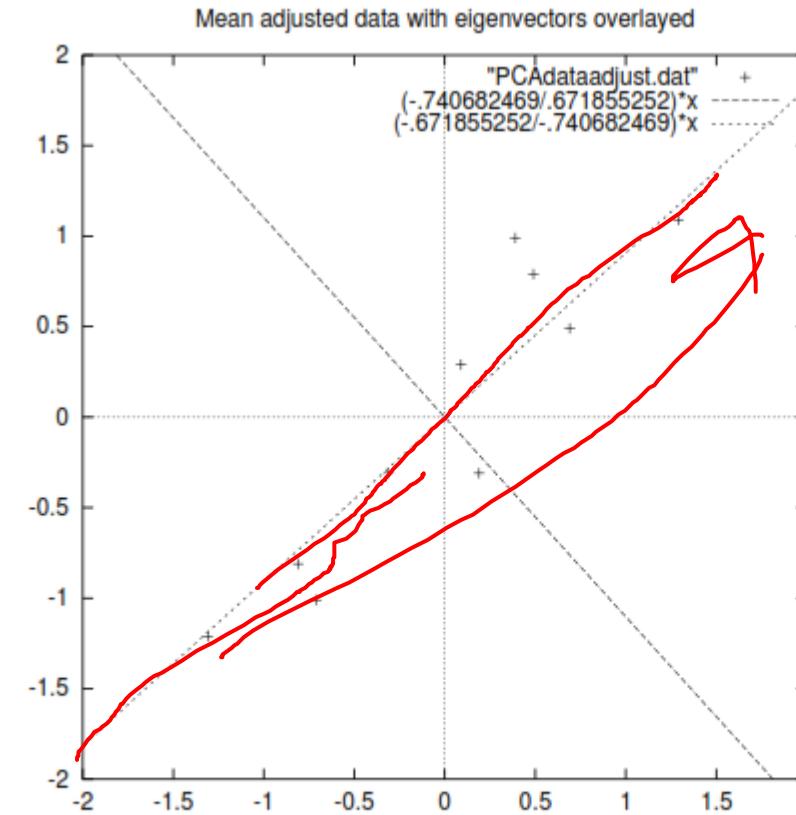


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.



Step 5: Choosing components and forming a feature vector

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

- The eigenvector with the highest eigenvalue is the principle component of the data set.
- In our example, the eigenvector with the largest eigenvalue was the one that pointed down the middle of the data.
- It is the most significant relationship between the data dimensions.

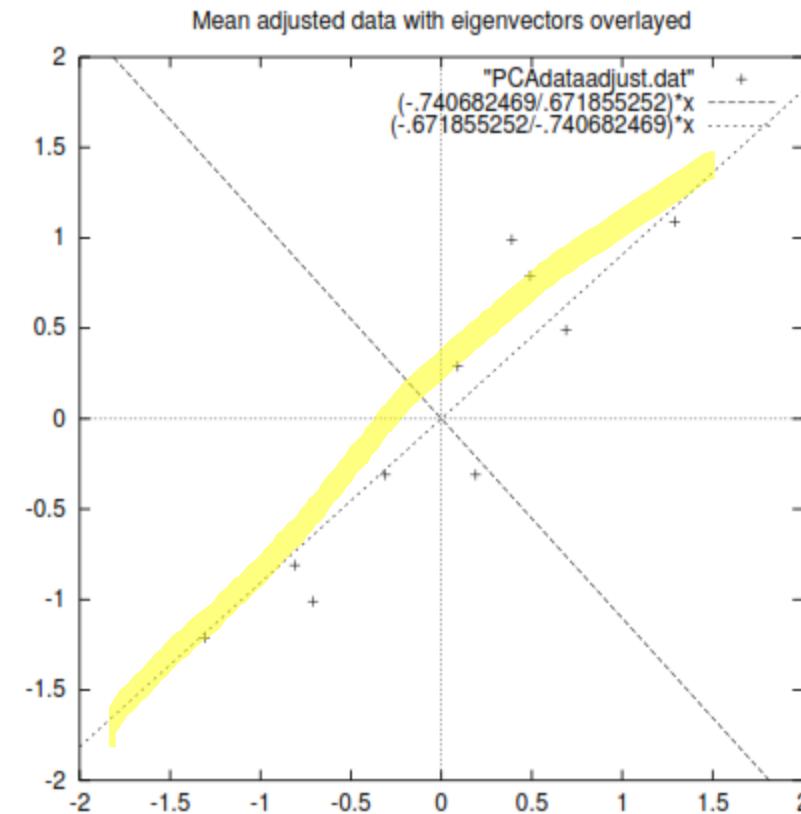


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.



Step 5: Choosing components and forming a feature vector

- In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, **highest to lowest**.
- This gives you the components in order of significance.
- Now, if you like, **you can decide to ignore the components** of lesser significance.
- You do lose **some information**, but if **the eigenvalues** are small, you don't lose much.
- If you leave **out some components**, the final **data set** will have less dimensions than the original.
- To be precise, if you originally **have n dimensions** in your data, and so you calculate **n eigenvectors** and **eigenvalues**, and then you choose only the first p eigenvectors, then the final data **set has only p dimensions**.



Step 5: Choosing components and forming a feature vector

- The feature vector is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

$$\text{FeatureVector} = (\text{eig}_1 \text{ eig}_2 \text{ eig}_3 \dots \text{ eig}_n)$$

- Given our example set of data, and the fact that we have 2 eigenvectors, we have two choices. We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

- or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$



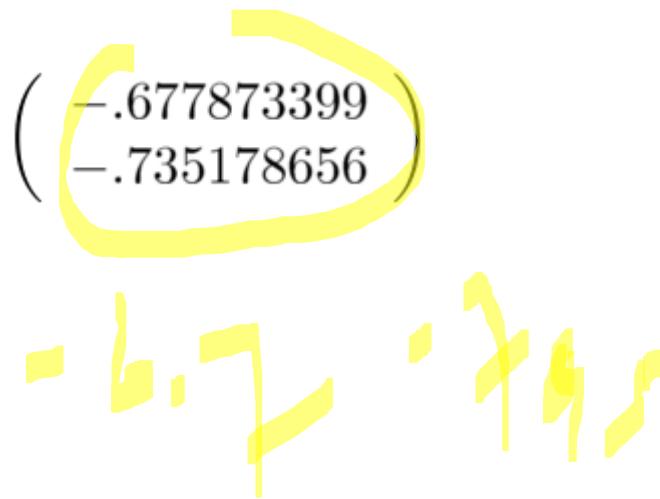
Step 6: Deriving the new data set

- Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

$$\text{FinalData} = \text{RowFeatureVector} \times \text{RowDataAdjust}$$

- RowFeatureVector* is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top.
- RowDataAdjust* is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension.
- FinalData* is the final data set, with data items in columns, and dimensions along rows.





DataAdjust =

<i>x</i>	<i>y</i>
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01



Step 6: Deriving the new data set

- We have changed our data from being in terms of the axes x and y , and now they are in terms of our 2 eigenvectors.
- In the case of when the new data set has reduced dimensionality, i.e. we have left some of the eigenvectors out, the new data is only in terms of the vectors that we decided to keep.



In the case of keeping both eigenvectors for the transformation, we get the data and the plot found in Figure 3.3.

	<i>x</i>	<i>y</i>
	-.827970186	-.175115307
	1.77758033	.142857227
	-.992197494	.384374989
	-.274210416	.130417207
Transformed Data=	-1.67580142	-.209498461
	-.912949103	.175282444
	.0991094375	-.349824698
	1.14457216	.0464172582
	.438046137	.0177646297
	1.22382056	-.162675287

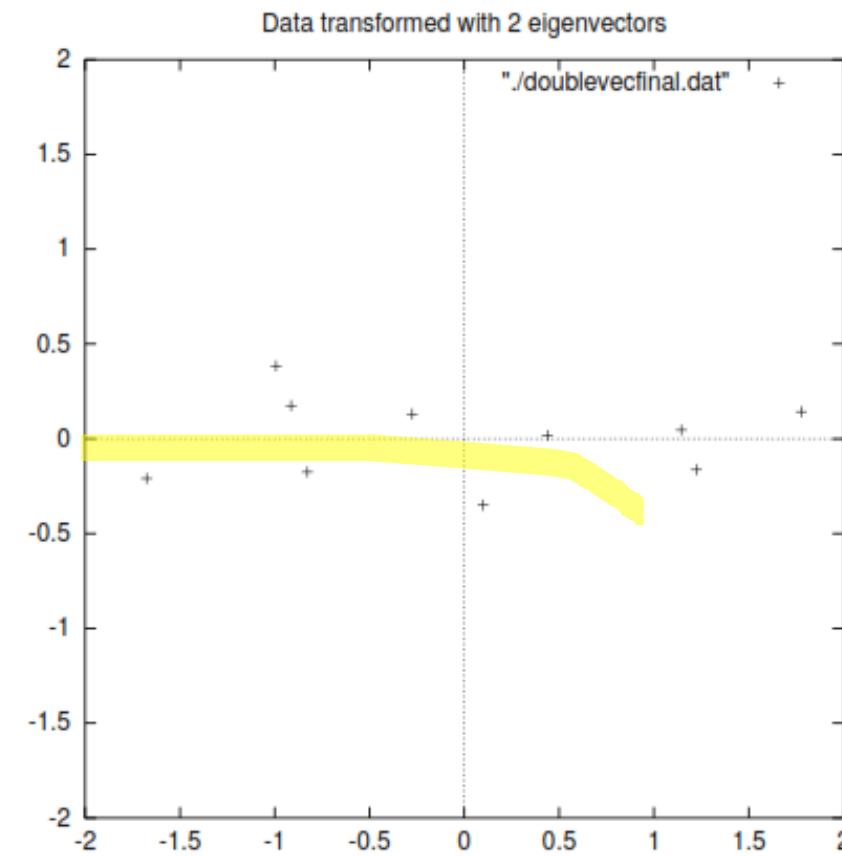


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.



- In the case of keeping only one eigenvectors for the transformation, we get the data only in one column.
- This data set is exactly the first column of the other.
- So, if you were to plot this data, it would be 1-dimensional, and would be points on a line in exactly the x positions of the points in the plot in Figure 3.3.

Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056



Getting the old data back

- It is obviously of great concern if you are using the PCA transform for data compression.
- If we took all the eigenvectors in our transformation will we get exactly the original data back.
- If we have reduced the number of eigenvectors in the final transformation, then the retrieved data has lost some information.



Getting the old data back

- Recall that the final transform is this:

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

- which can be turned around so that, to get the original data back:

$$RowDataAdjust = RowFeatureVector^{-1} \times FinalData$$

- However, when we take all the eigenvectors in our feature vector, it turns out that the inverse of our feature vector is actually equal to the transpose of our feature vector.
- This is only true because the elements of the matrix are all the unit eigenvectors of our data set. This makes the return trip to our data easier, because the equation becomes:

$$RowDataAdjust = RowFeatureVector^T \times FinalData$$

Getting the old data back

- But, to get the actual original data back, we need to add on the mean of that original data (remember we subtracted it right at the start). So, for completeness:

$$\text{RowOriginalData} = (\text{RowFeatureVector}^T \times \text{FinalData}) + \text{OriginalMean}$$

Getting the old data back

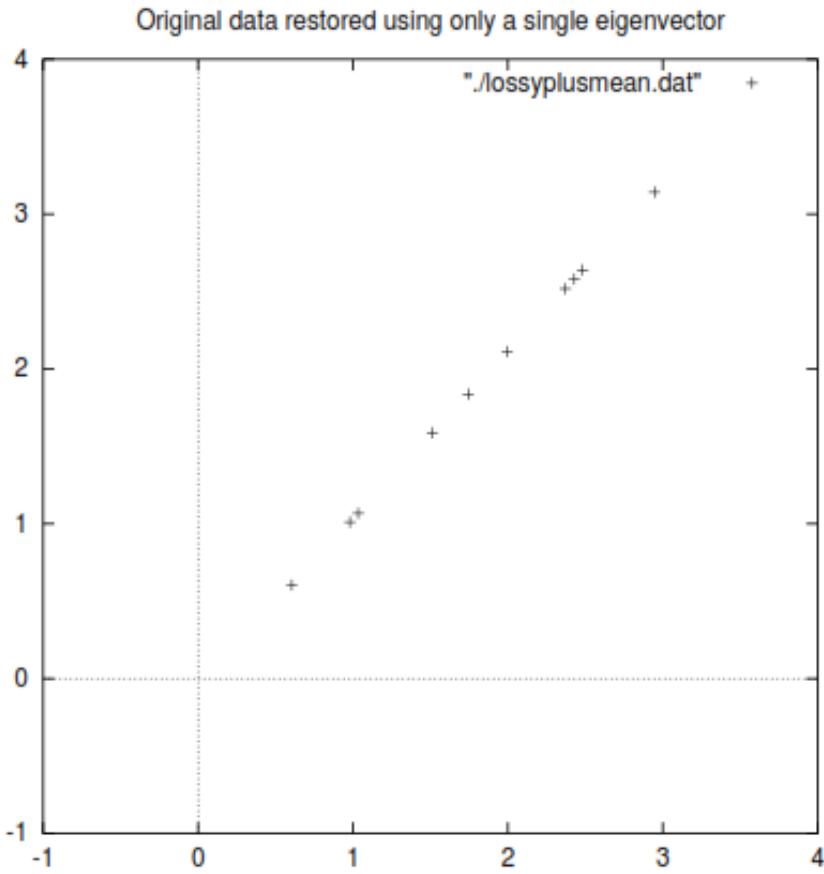


Figure 3.5: The reconstruction from the data that was derived using only a single eigenvector

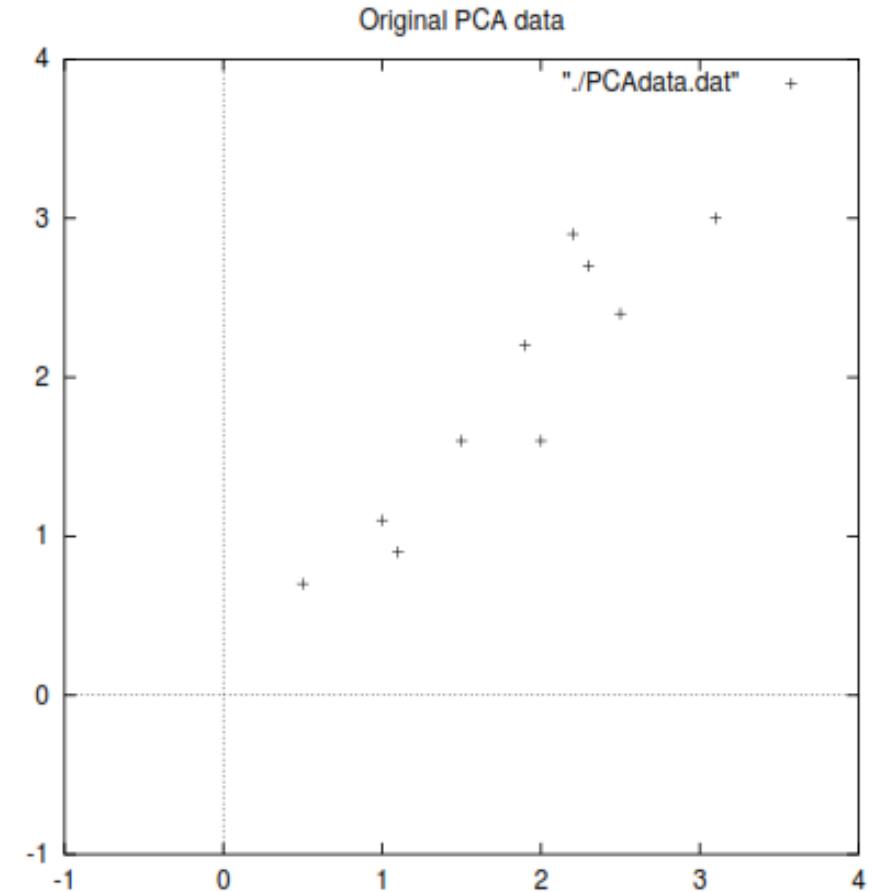


Figure 3.1: PCA example data,

Getting the old data back

- If you compare it to the original data plot in Figure 3.1, then you will notice:
 - the variation along the principle eigenvector (see Figure 3.2 for the eigenvector overlayed on top of the mean-adjusted data) has been kept
 - the variation along the other component (the other eigenvector that we left out) has gone.

Curse Dimensionality and Feature Reduction

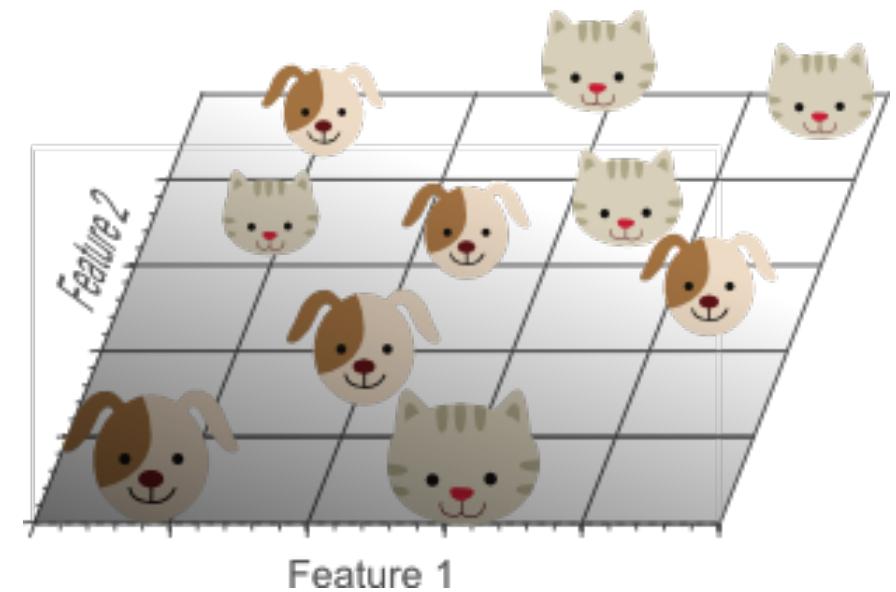


The Curse of Dimensionality



Contoh

- Contoh membedakan anjing dan kucing.
- Hal-hal yang membedakan anjing dan kucing (feature 1 dan feature 2)
- Contoh
 - the average red color, the average green color and the average blue color of the image → 3 features.



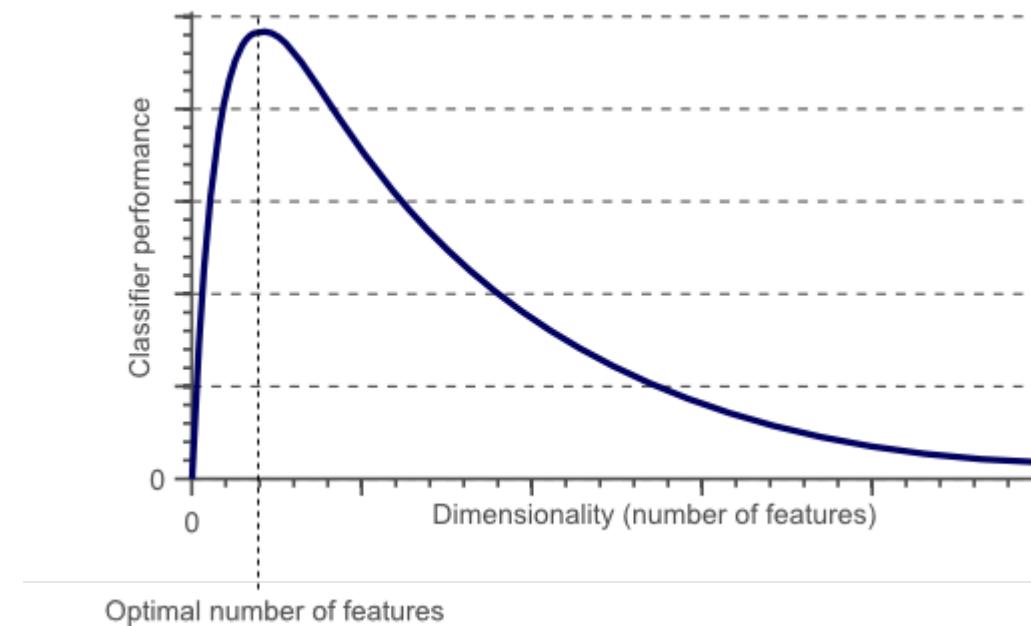
Contoh

- 3 feature tidak cukup untuk classifier
 - average edge atau gradient intensity pada X dan Y direction → 5 features.
- Untuk menambah akurasi, bisa dengan menambah fitur : texture, statistical moment dll
 - Apakah semakin banyak feature akan semakin detail?



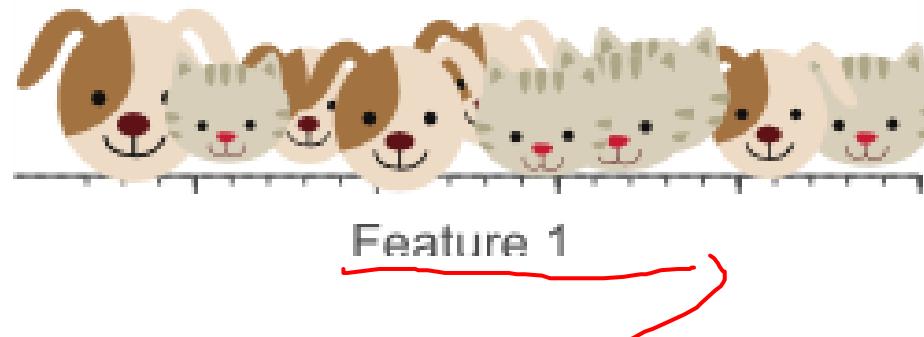
Curse of dimensionality

- As the dimensionality increases, the classifier's performance increases until the optimal number of features is reached. Further increasing the dimensionality without increasing the number of training samples results in a decrease in classifier performance.



Single feature

- Misalkan dalam melakukan klasifikasi menggunakan single feature. Data training yang digunakan 10

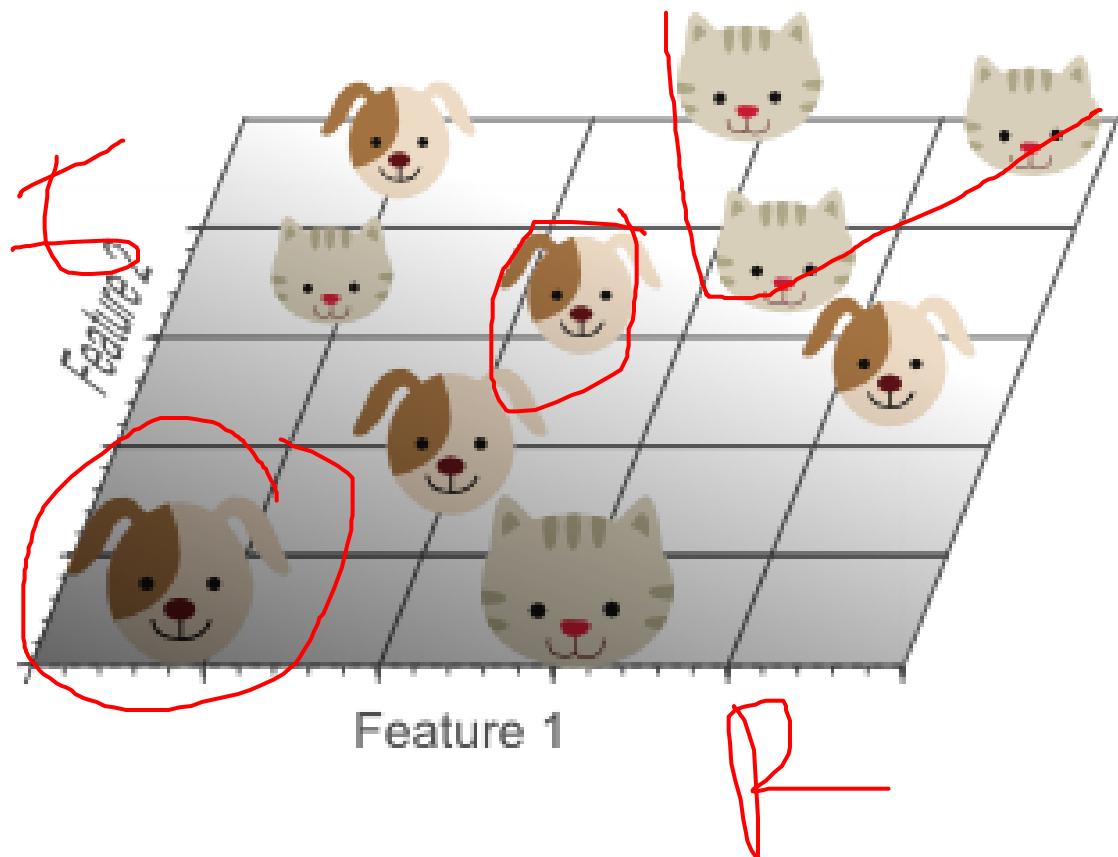


- A single feature does not result in a perfect separation of our training data.



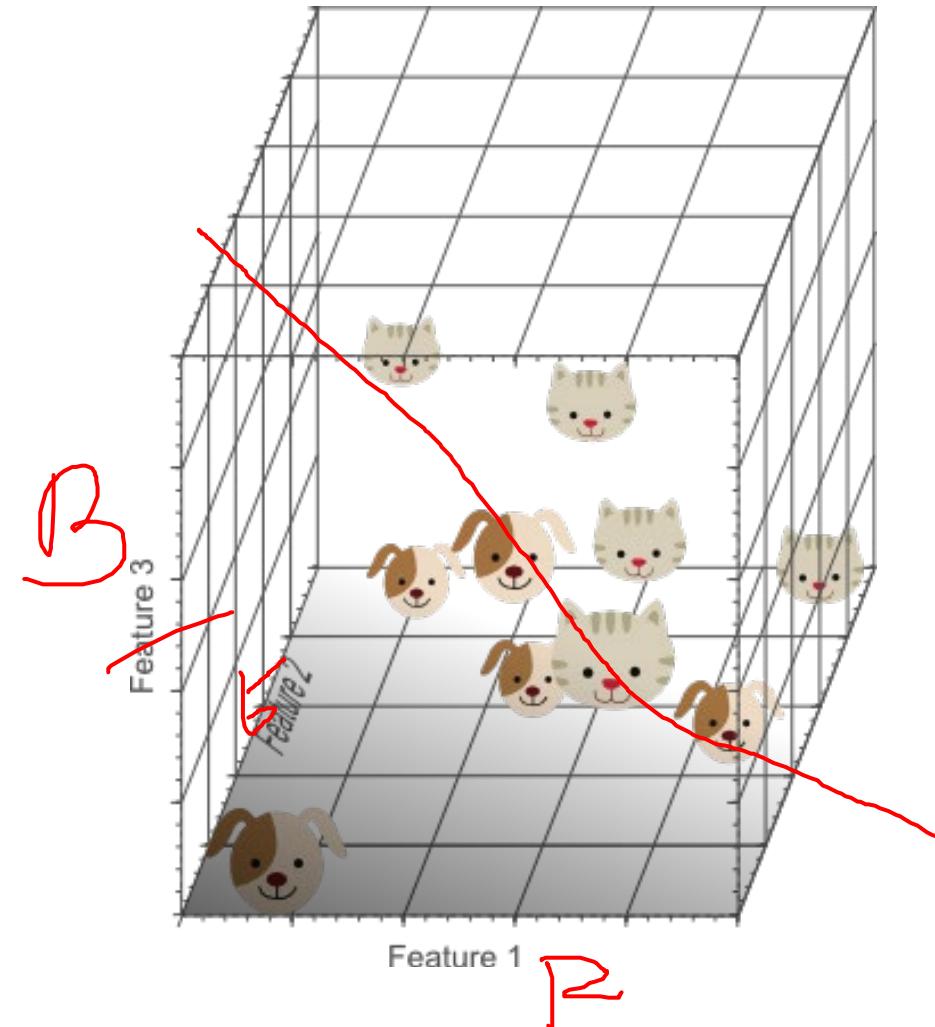
2 Feature

- Menambahkan 1 fitur lagi, misal rata2 warna hijau:
- Adding a second feature still does not result in a linearly separable classification problem: No single line can separate all cats from all dogs in this example.

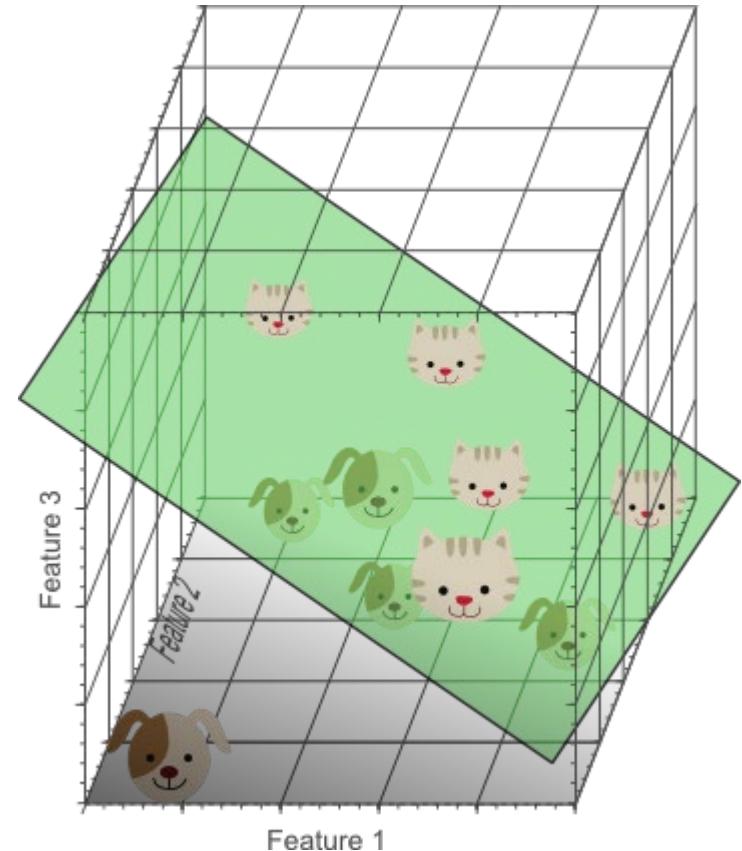


3 feature

- Menambahkan fitur yang ketiga (warna biru)
- Adding a third feature results in a linearly separable classification problem in our example. A plane exists that perfectly separates dogs from cats.

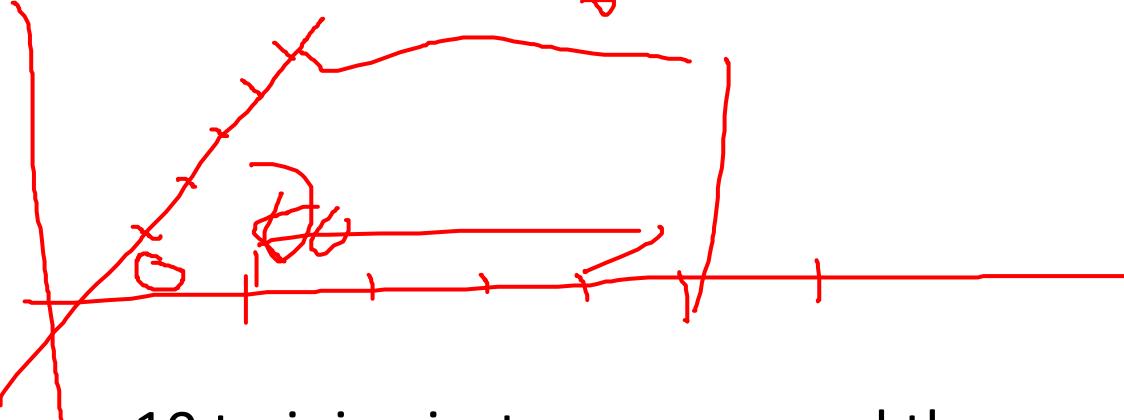


- In the three-dimensional feature space, we can now find a plane that perfectly separates dogs from cats. This means that a linear combination of the three features can be used to obtain perfect classification results on our training data of 10 images:



- The above illustrations might seem to suggest that increasing the number of features until perfect classification results are obtained is the best way to train a classifier.
- However, note how the density of the training samples decreased exponentially when we increased the dimensionality of the problem.





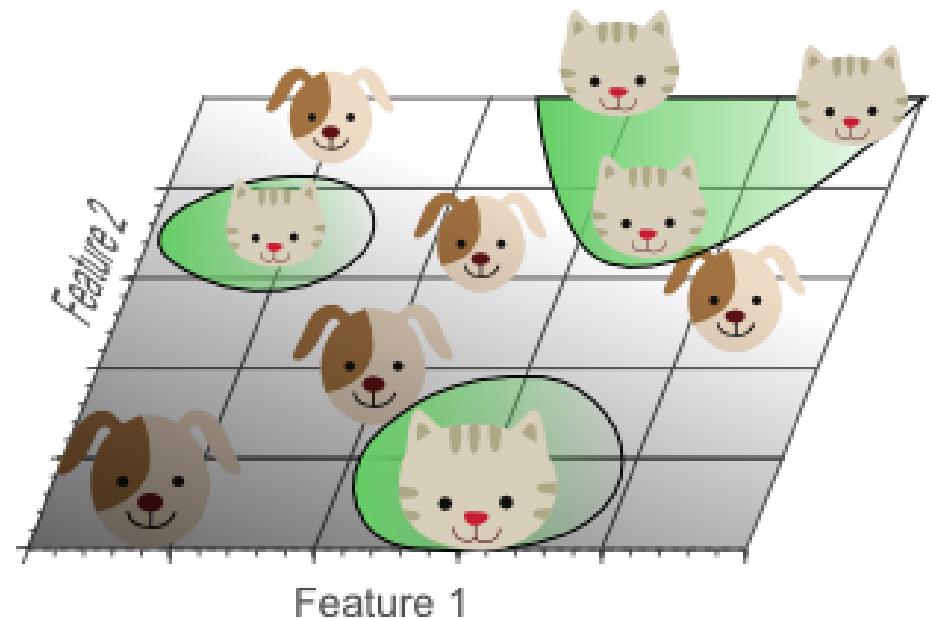
- In the 1D case, 10 training instances covered the complete 1D feature space, the width of which was 5 unit intervals.
 - The sample density was $10/5 = \mathbf{2 \text{ samples/interval}}$.
- In the 2D case however, 10 training instances cover a 2D feature space with an area of $5 \times 5 = 25$ unit squares.
 - The sample density was $10/25 = \mathbf{0.4 \text{ samples/interval}}$. $\frac{25}{10}$
- Finally, in the 3D case, the 10 samples had to cover a feature space volume of $5 \times 5 \times 5 = 125$ unit cubes.
 - The sample density was $10/125 = \mathbf{0.08 \text{ samples/interval}}$. $\frac{125}{10}$



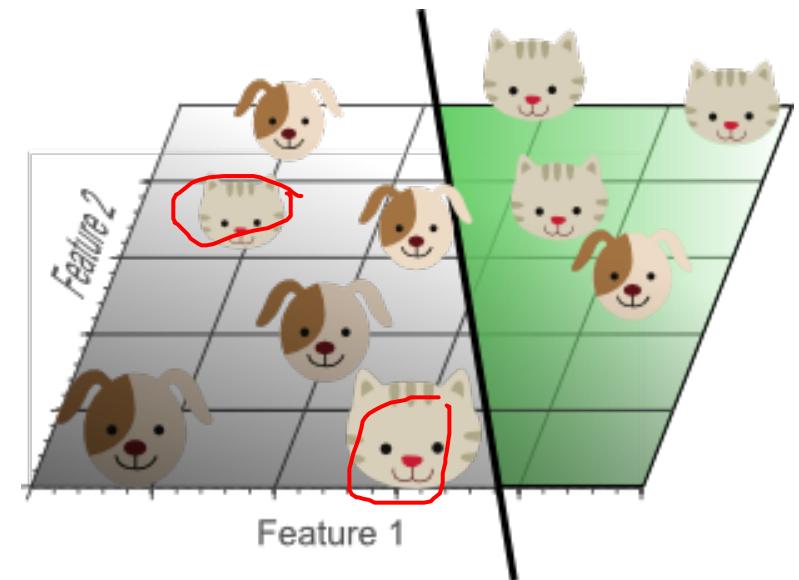
- If we would keep adding features, the dimensionality of the feature space grows, and becomes sparser and sparser.
- Due to this sparsity, it becomes much more easy to find a separable hyperplane because the likelihood that a training sample lies on the wrong side of the best hyperplane becomes infinitely small when the number of features becomes infinitely large.

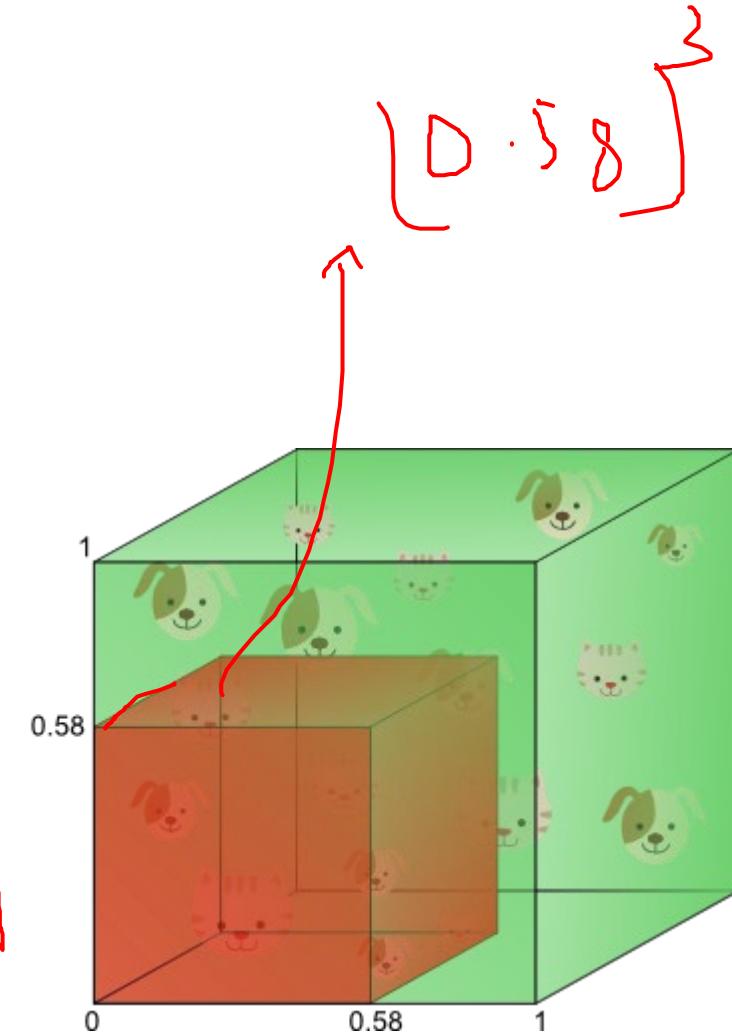
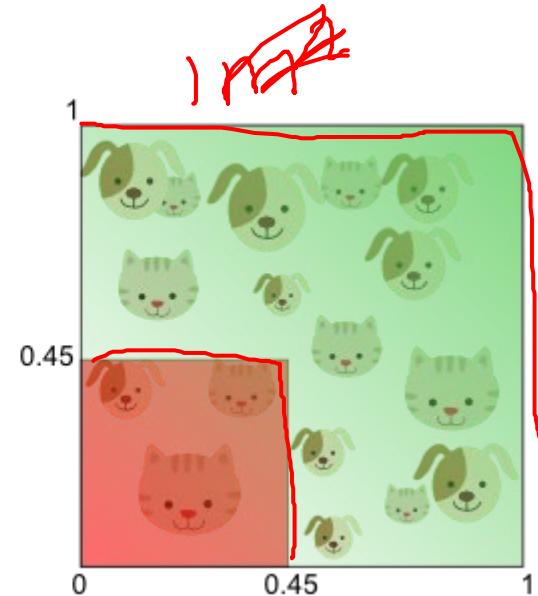
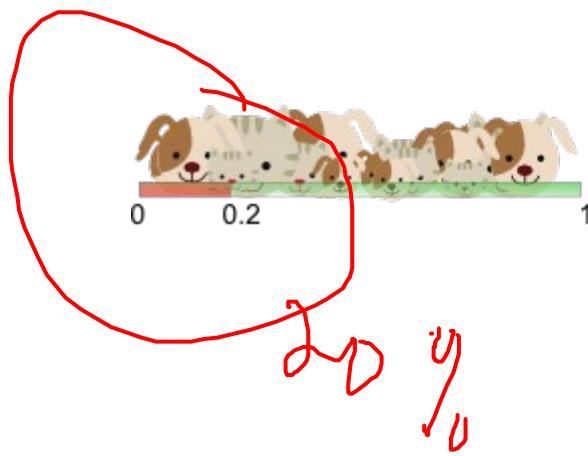


- However, if we project the highly dimensional classification (3D features) result back to a lower dimensional space (2D features), a serious problem associated with this approach becomes evident:
- Using too many features results in overfitting. The classifier starts learning exceptions that are specific to the training data and do not generalize well when new data is encountered.
- Because of this, the resulting classifier would fail on real-world data, consisting of an infinite amount of unseen cats and dogs that often do not adhere to these exceptions.



- This figure shows the result of a linear classifier that has been trained using only 2 features instead of 3:
- Although the simple linear classifier using 2 features seems to perform worse than the non-linear classifier using 3 features this simple classifier generalizes much better to unseen data because it did not learn specific exceptions that were only in our training data by coincidence.
- In other words, by using less features, the curse of dimensionality was avoided such that the classifier did not overfit the training data.





20% → 58%



- Dengan kata lain, jika jumlah training data fixed, maka overfitting akan terjadi jika dimensi nya terus ditambahkan
 - in other words, if the amount of available training data is fixed, then overfitting occurs if we keep adding dimensions.
- Maka jika jumlah dimensi ditambah, harus juga diimbangi dengan penambahan training data (penambahan secara eksponensial), untuk menghindari overfitting
 - On the other hand, if we keep adding dimensions, the amount of training data needs to grow exponentially fast to maintain the same coverage and to avoid overfitting.



Dimensionality Reduction



Why Dimensionality reduction?

- machine learning and data mining techniques may not be effective for high dimensional data
 - Curse of Dimensionality
- The intrinsic dimension may be small.
 - For example, the number of genes responsible for a certain type of disease may be small.



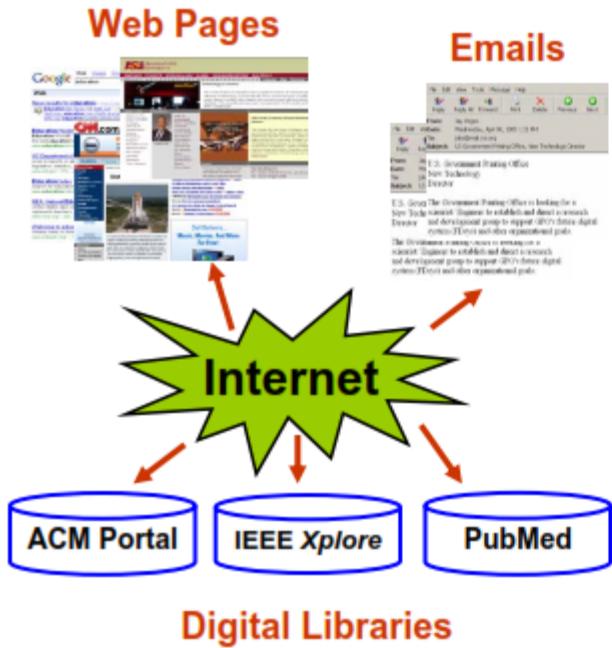
Why Dimensionality reduction? (cont.)

- Visualization: projection of high-dimensional data onto 2D or 3D.
- Data compression: efficient storage and retrieval.
- Noise removal : positive effect on query accuracy.



Application of Dimensionality Reduction

Document Classification

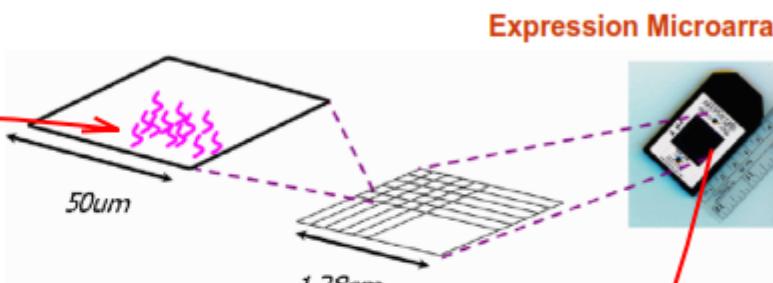
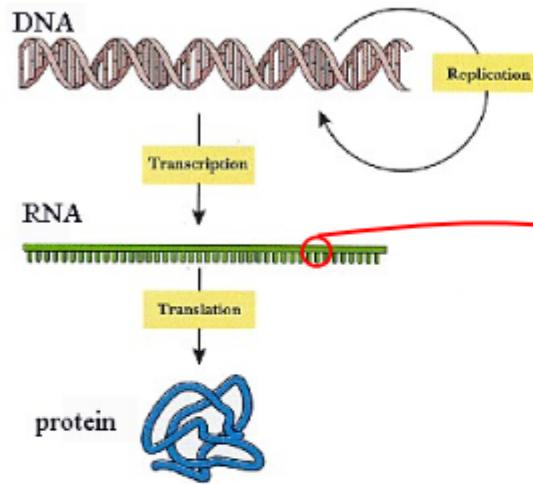


Documents	Terms		C		
	T ₁	T ₂	T _N	
D ₁	12	0	6	Sports
D ₂	3	10	28	Travel
⋮	⋮	⋮	⋮	⋮	⋮
D _M	0	11	16	Jobs

- **Task:** To classify unlabeled documents into categories
- **Challenge:** thousands of terms
- **Solution:** to apply dimensionality reduction



Gene Expression Microarray Analysis



- **Task:** To classify novel samples into known disease types (disease diagnosis)
- **Challenge:** thousands of genes, few samples
- **Solution:** to apply dimensionality reduction

Gene Sample \	M23197_at	U66497_at	M92287_at	...	Class
Sample 1	261	88	4778	...	ALL
Sample 2	101	74	2700	...	ALL
Sample 3	1450	34	498	...	AML
...

Expression Microarray Data Set



Other Types of High-Dimensional Data



Face images



Handwritten digits



Major Techniques of Dimensionality Reduction

- Feature selection
- Feature extraction (reduction)



Feature Selection

Definition:

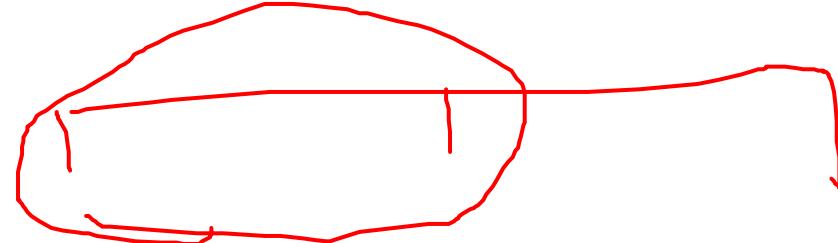
- A process that chooses an optimal subset of features according to a objective function.
- Proses memilih bagian dari fitur sesuai dengan fungsi objektive

Objectives:

- To reduce dimensionality and remove noise
- To improve mining performance
 - Speed learning
 - Predictive accuracy



Feature Reduction



- Feature reduction refers to the mapping of the original high-dimensional data onto a lower-dimensional space
- Given a set of data points of p variables $\{x_1, x_2, \dots, x_n\}$ Compute their low-dimensional representation:

$$x_i \in \Re^d \rightarrow y_i \in \Re^p \quad (p \ll d)$$

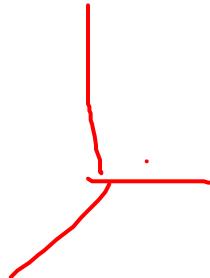
- Criterion for feature reduction can be different based on different problem settings.
 - Unsupervised setting: minimize the information loss
 - Supervised setting: maximize the class discrimination



Feature Reduction vs. Feature Selection

- Feature reduction

- All original features are used
 - The transformed features are linear combinations of the original features



- Feature selection

- Only a subset of the original features are selected

- Continuous versus discrete



Feature Reduction vs. Feature Selection

- Feature extraction approaches transform original data in such way, and the output are not in original domain space.
- Feature selection approaches select the most significant subset of features (genes) from its original domain space, so the output are still in original domain space.



Machine Learning

Tom Heskes



Learning goals: machine learning

- Can you?
 - Tell the difference between supervised, unsupervised and reinforcement learning and between regression and classification
 - Explain the principles behind Ockham's razor and how it relates to a model's inductive bias, its generalization performance and the risk of overfitting

Content

- Introduction
 - Supervised, reinforcement, unsupervised learning
- Supervised learning: classification, regression
- Generalisation vs. overfitting

Introduction

- Learning is an essential aspect of intelligence
 - Flexible behavior
 - Self-improvement
 - Less initial knowledge needed
 - No repetition of the same stupid behavior again and again

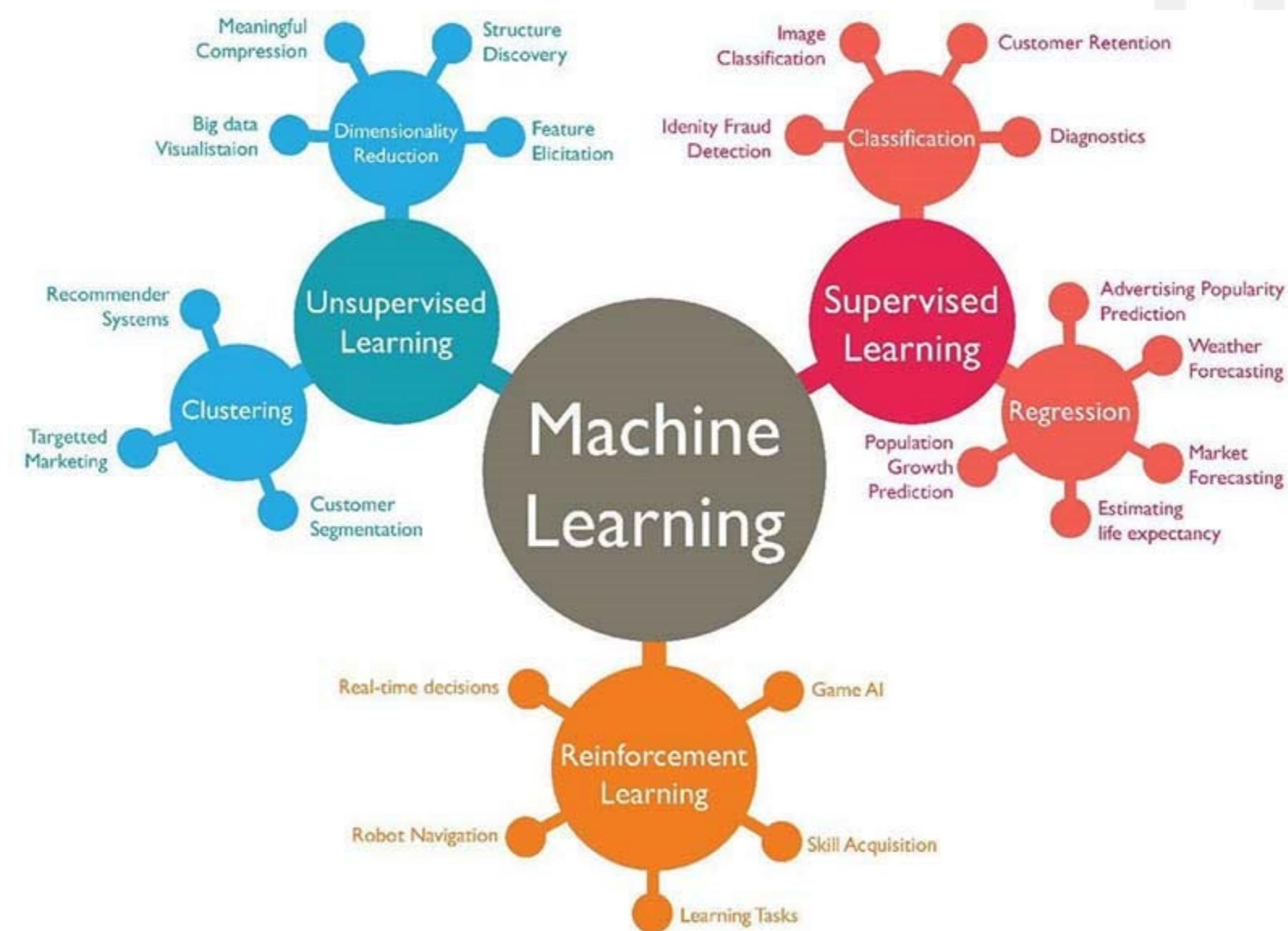


Definition

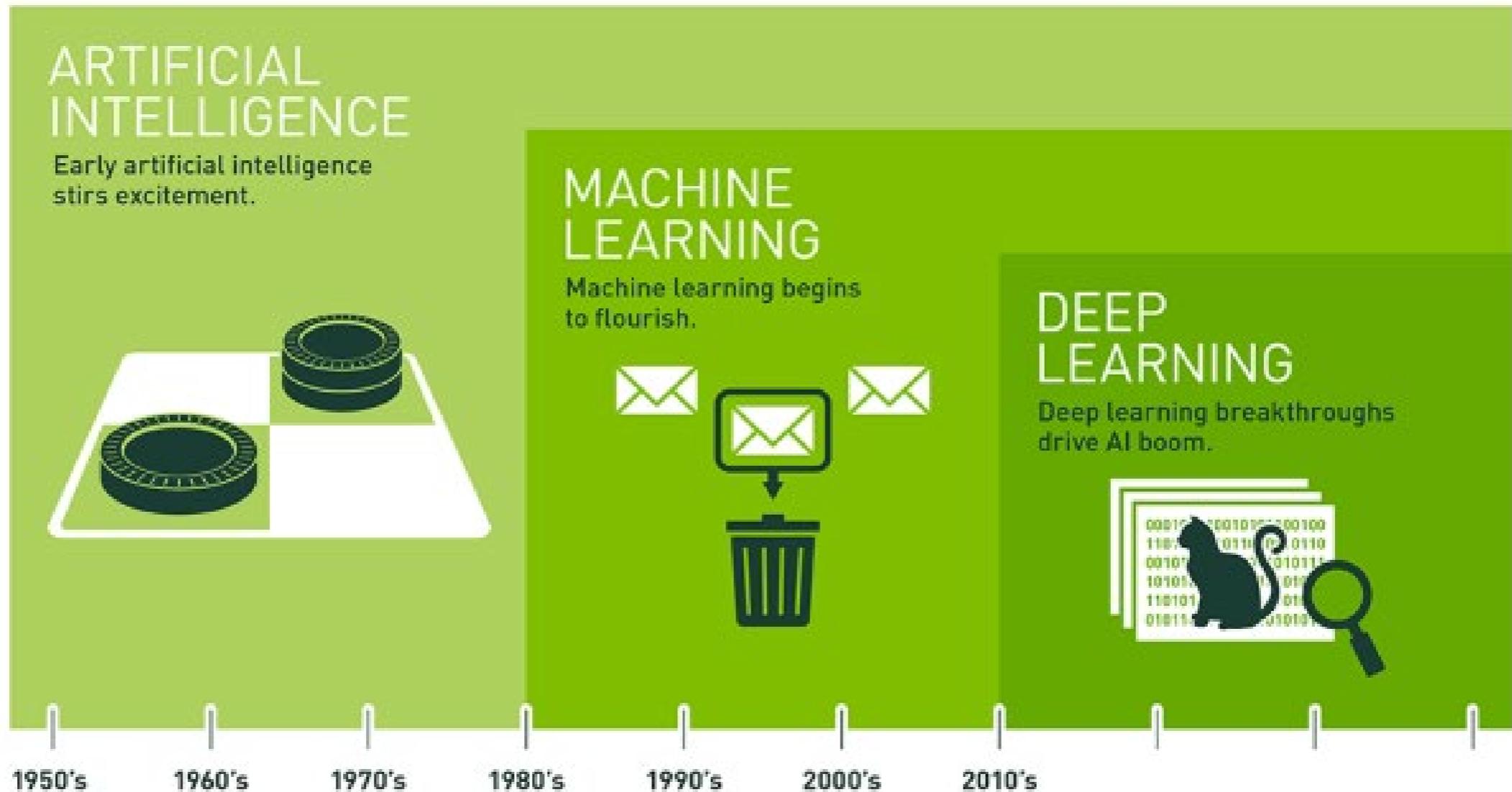
- Arthur Samuel (1959). Machine Learning:
Field of study that gives computers the ability
to learn without being explicitly programmed
- Tom Mitchell (1998). Well-posed learning problem: A computer
program is said to *learn* from experience E with respect to some
task T and some performance measure P , if its performance on T ,
as measured by P , improves with experience E



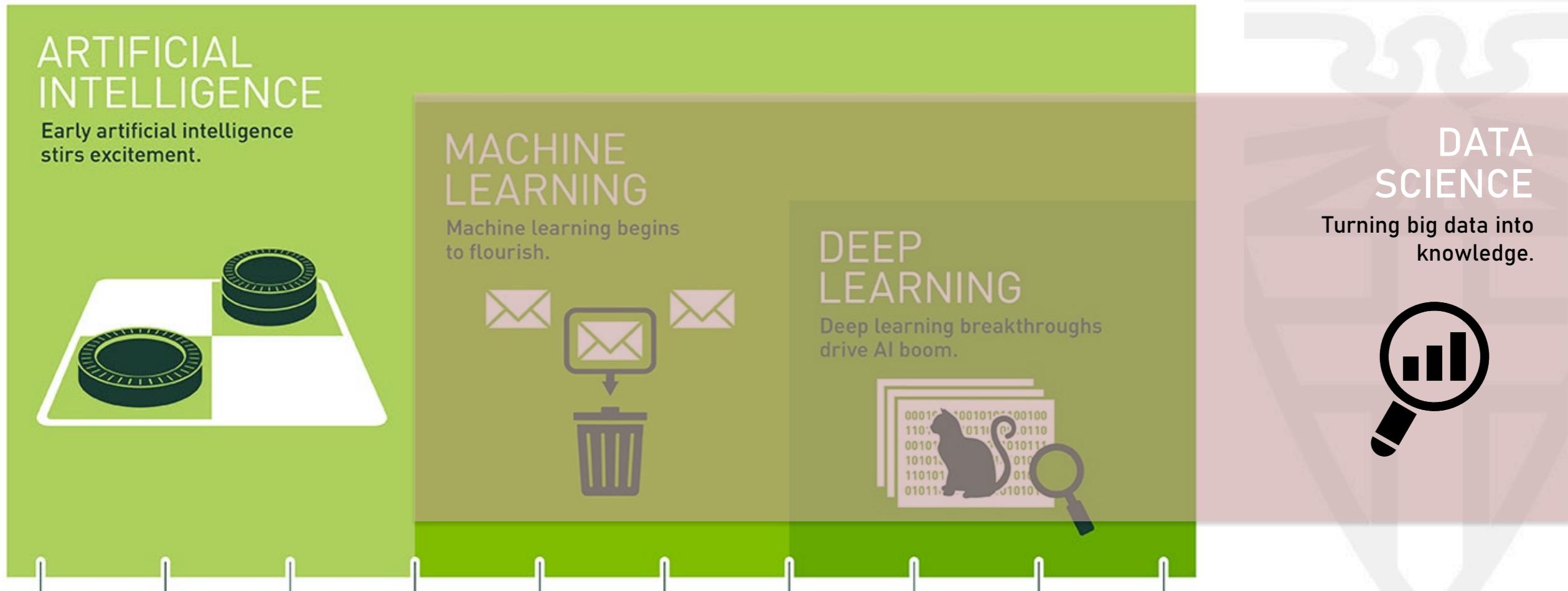
Types and Applications of Machine Learning



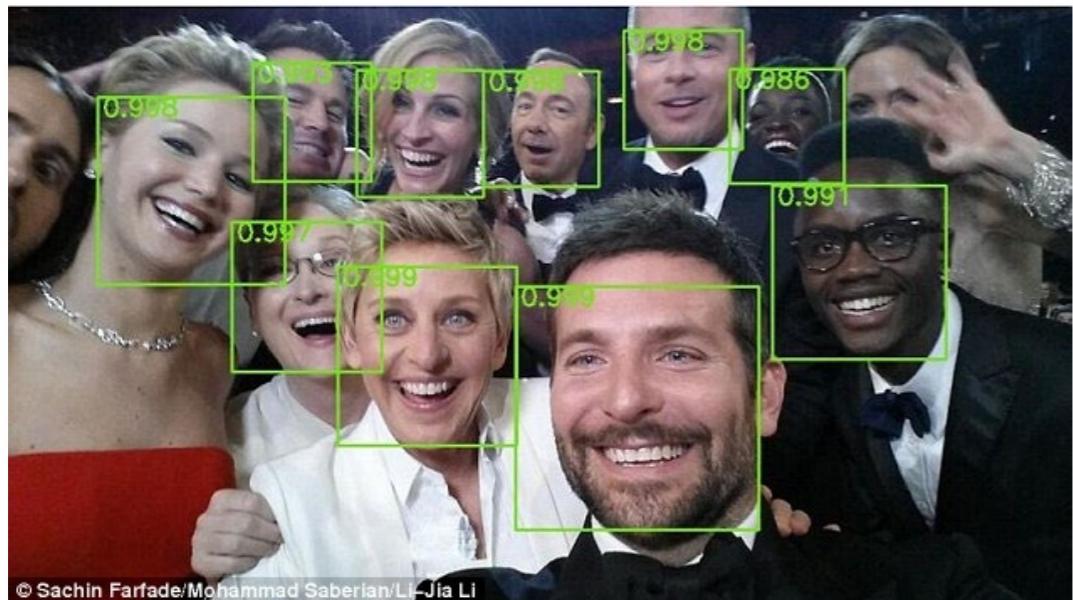
Artificial Intelligence, Machine Learning, Deep Learning



Data Science



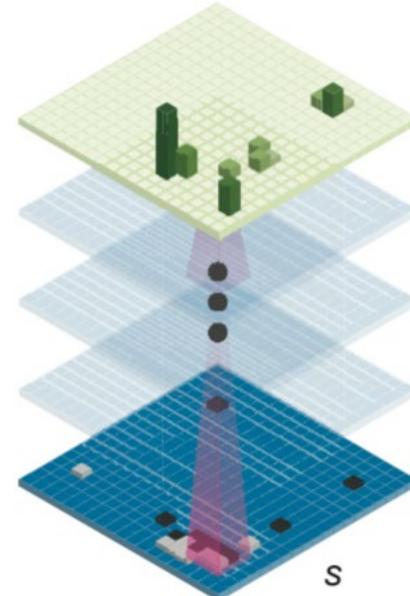
Some applications



face detection and
image tagging

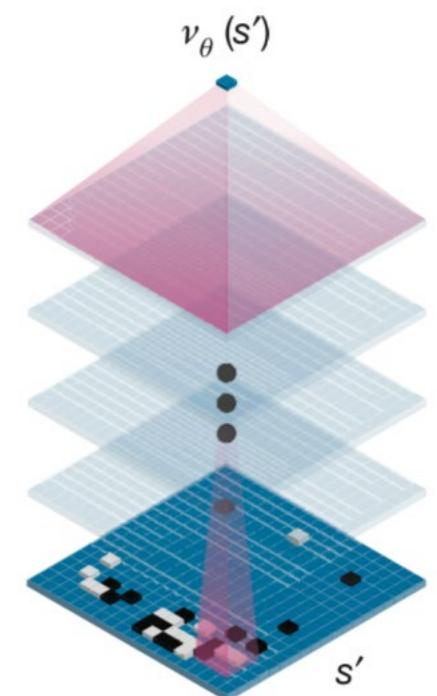
Policy network

$$p_{\sigma/\rho}(a|s)$$



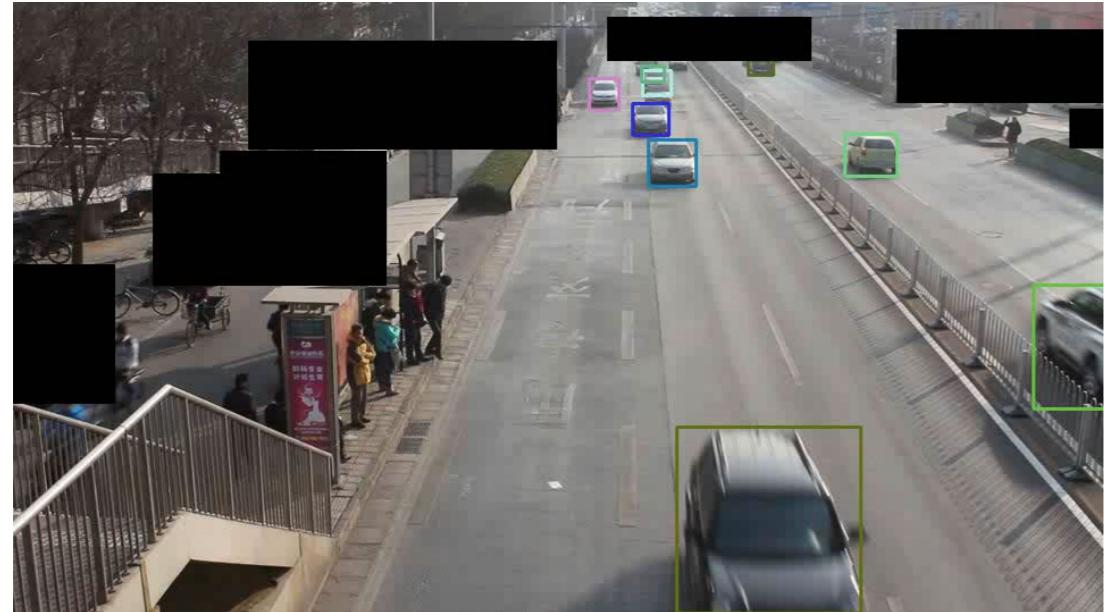
Value network

$$v_{\theta}(s')$$



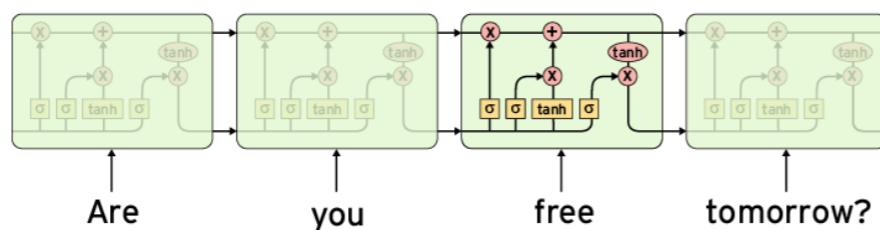
playing go

Some applications



counting cars

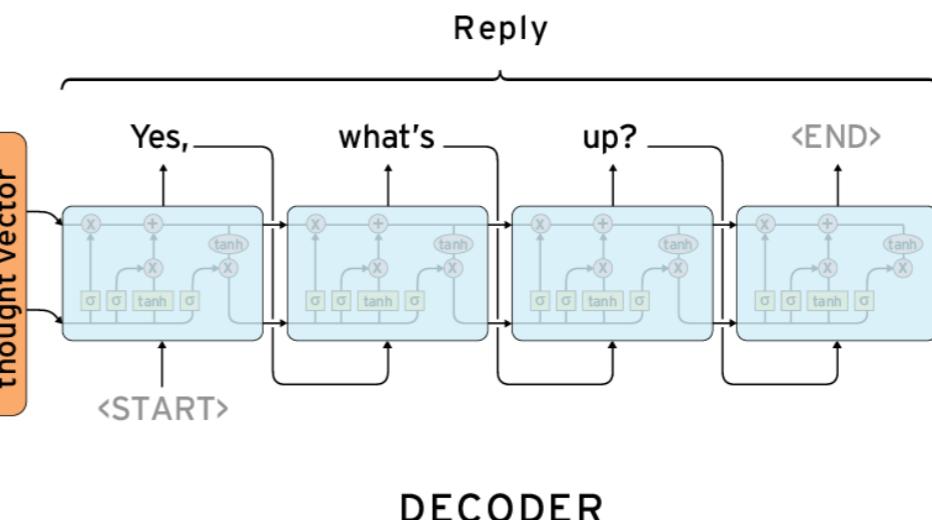
ENCODER



Incoming Email

chatbot

speech recognition



Reply

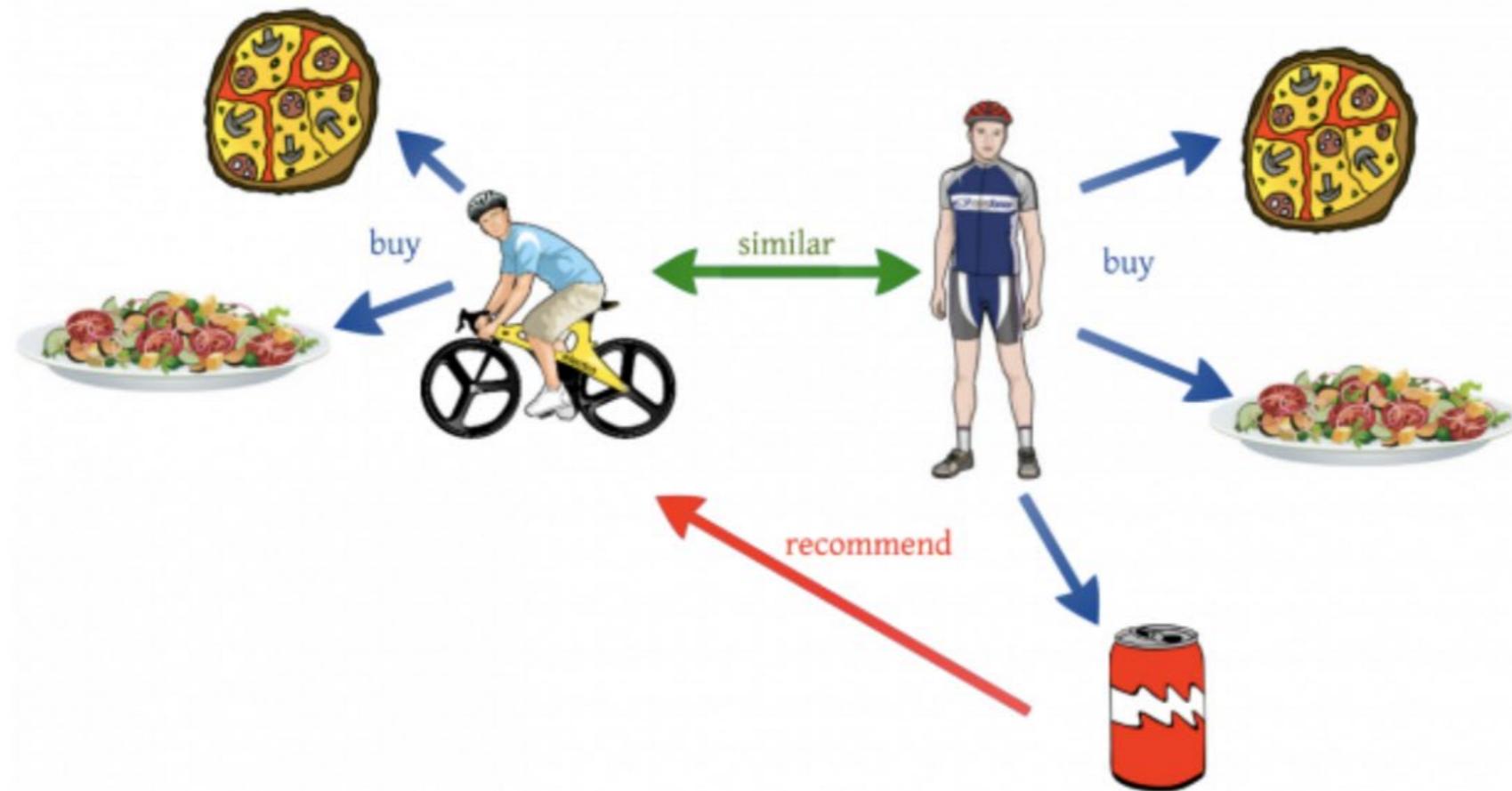
DECODER

Radboud University Nijmegen



Some applications

- Recommender systems / collaborative filtering
 - Tell you what you might like, based on what **similar** people liked
 - Used by: Amazon, Google, Netflix, Albert Heijn, etc

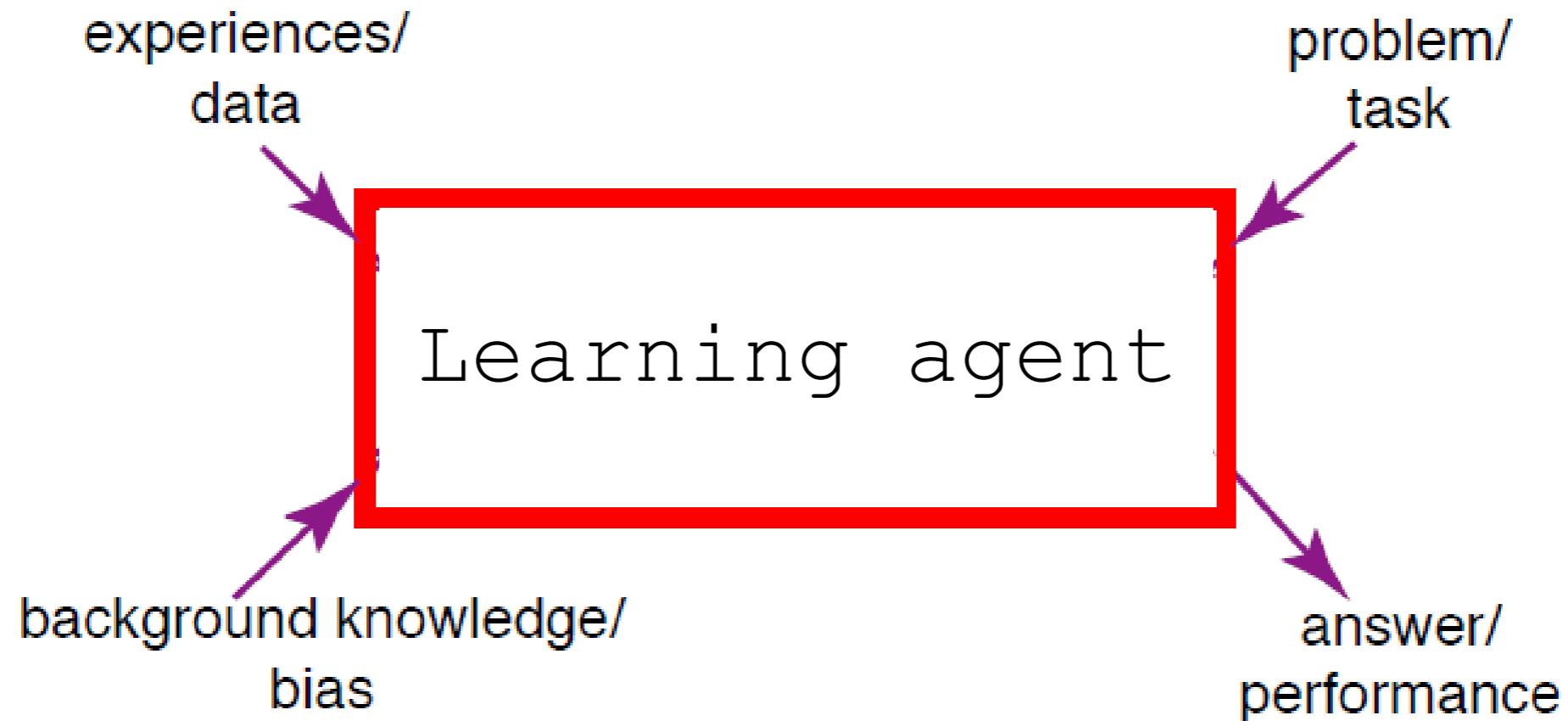


What to learn from?

- Learning can come about by
 - Being told (programming and data input)
 - **Being given examples** (induction)
 - Being given analogies
 - Doing (from self-generated experience)



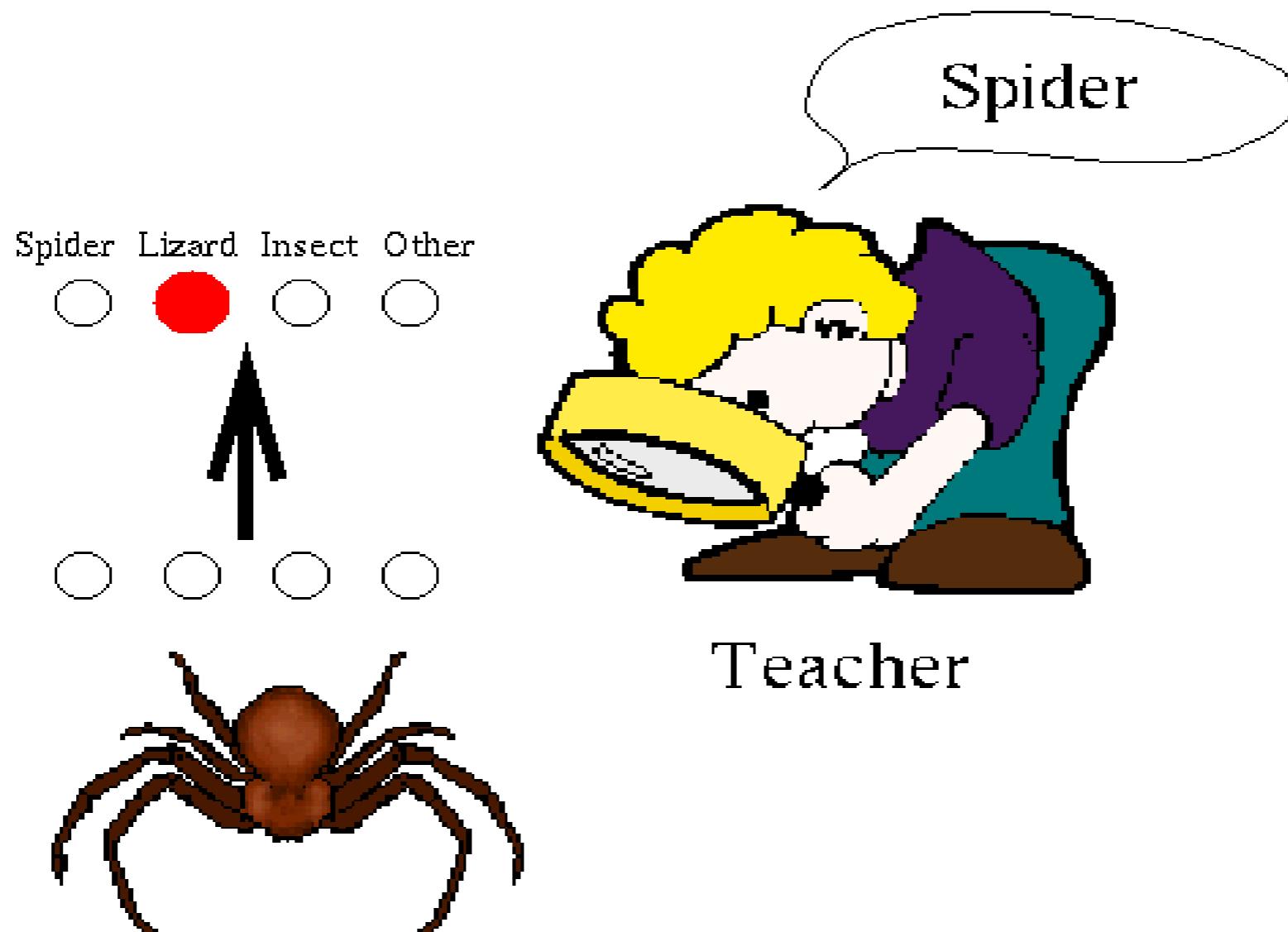
Learning Architecture



Types of Feedback

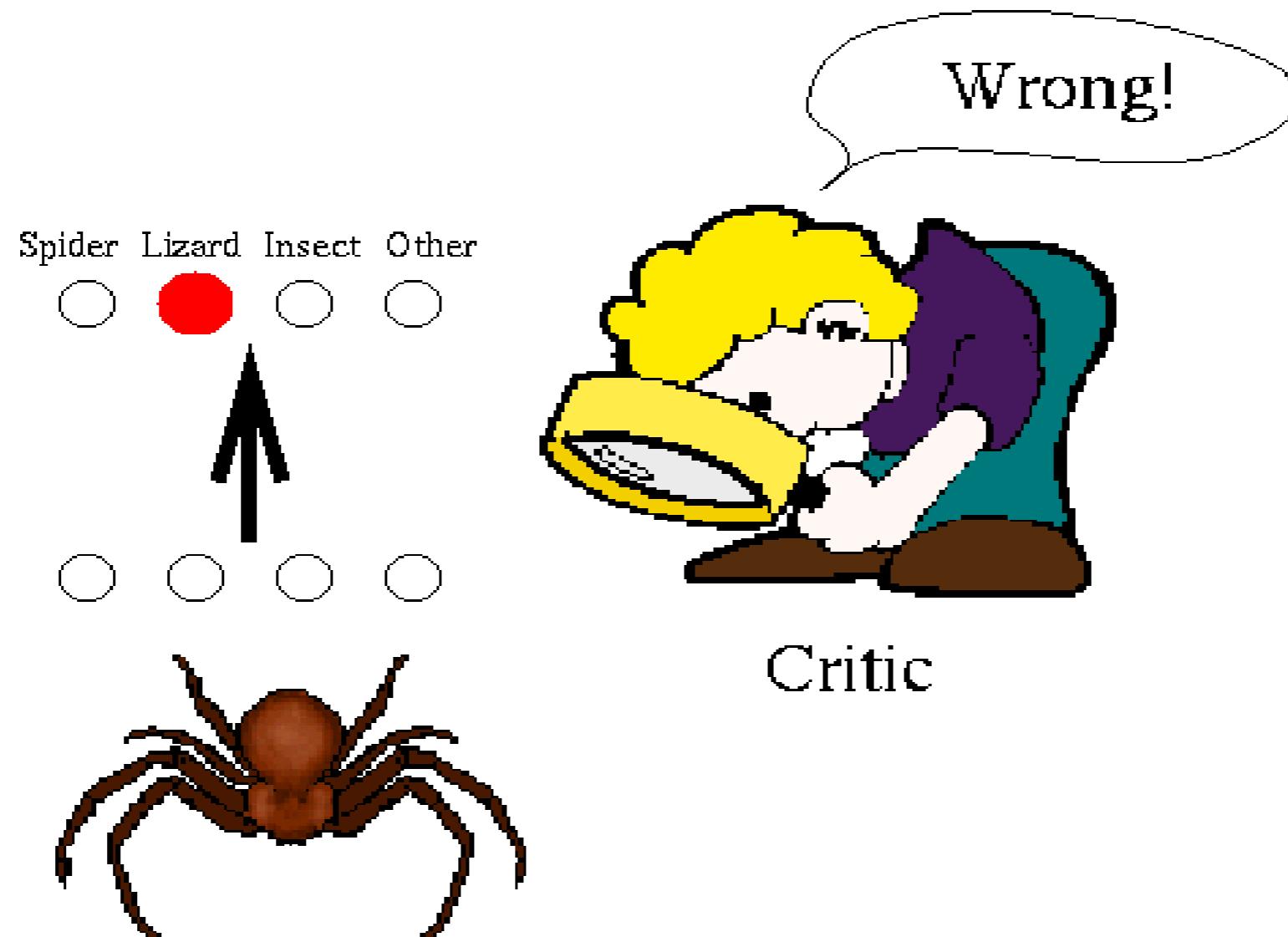
- Feedback divides learning algorithms into:
 - Supervised learning
 - Reinforcement learning
 - Unsupervised learning

Supervised learning

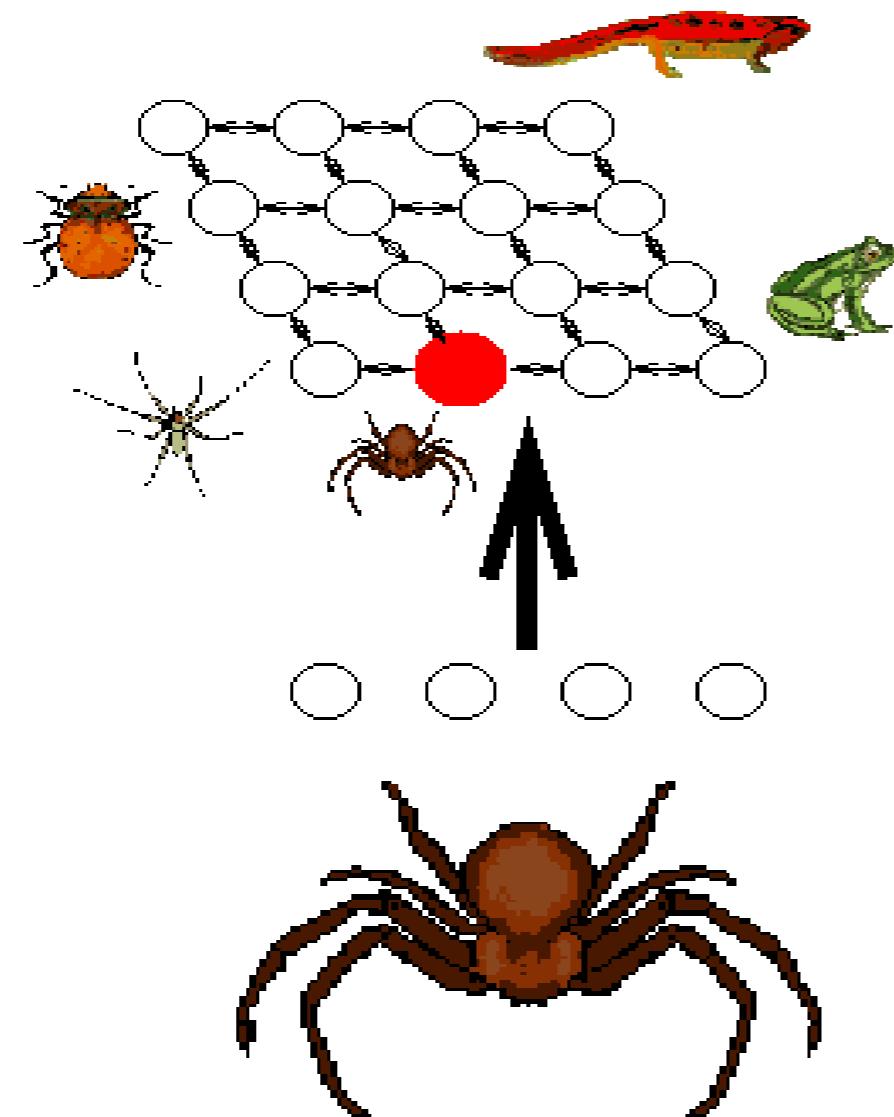


Thanks to Simon Dennis for the pictures

Reinforcement learning



Unsupervised learning



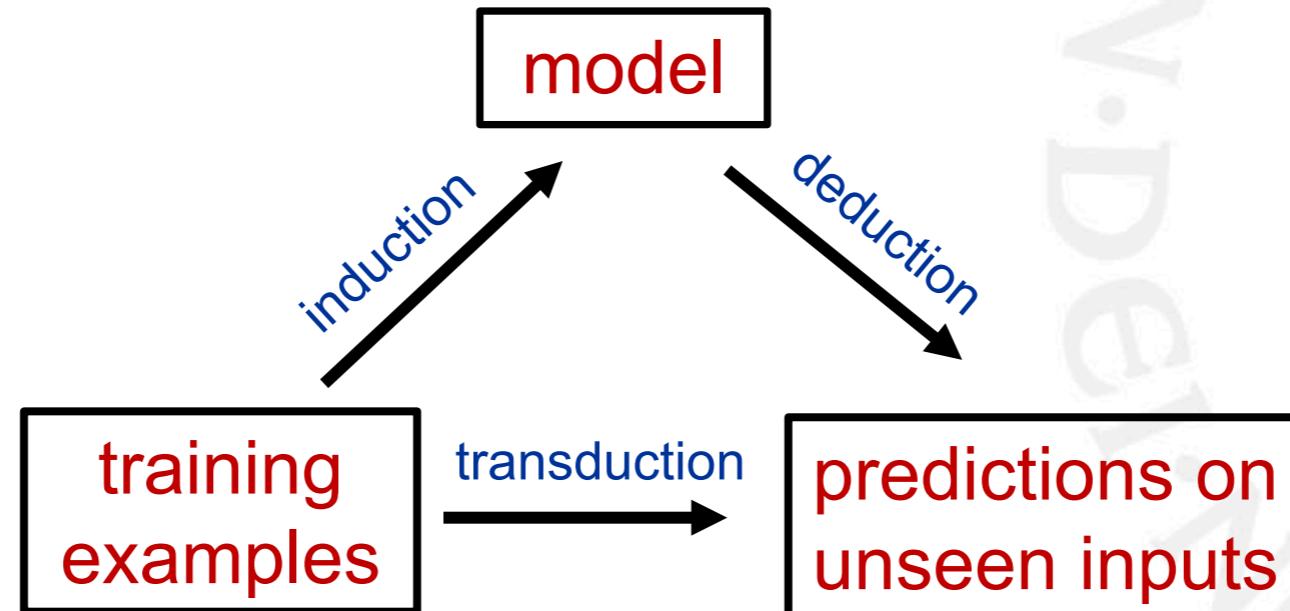
Map Layer

Next: Supervised Learning

- Classification, regression
- Generalization vs. overfitting
- Ockham's razor

Supervised Learning

- Teacher provides a collection of training examples for a target function f : $\{(x_1, y_1 = f(x_1)), (x_2, y_2 = f(x_2)), \dots, (x_n, y_n = f(x_n))\}$



Supervised Learning

- Fundamental problem of induction:

Find a hypothesis which **generalizes well**, i.e., performs well on unseen examples

- Useless to just memorize the training set

Generalization

Is the object in this picture a tree?

Yes



because it's green....



No



because I've never seen a tree
with **exactly** that many leaves before...

Generalization

Is the object in this picture a tree?

Yes



because it's green....

No



because I've never seen a tree
like this before...

Inductive Bias

- Preference for/restriction to some hypotheses based on prior knowledge (info not in the training set)
- This preference is called **inductive bias**
- Many possible biases – which is right?
- Good heuristic is **Ockham's razor**:

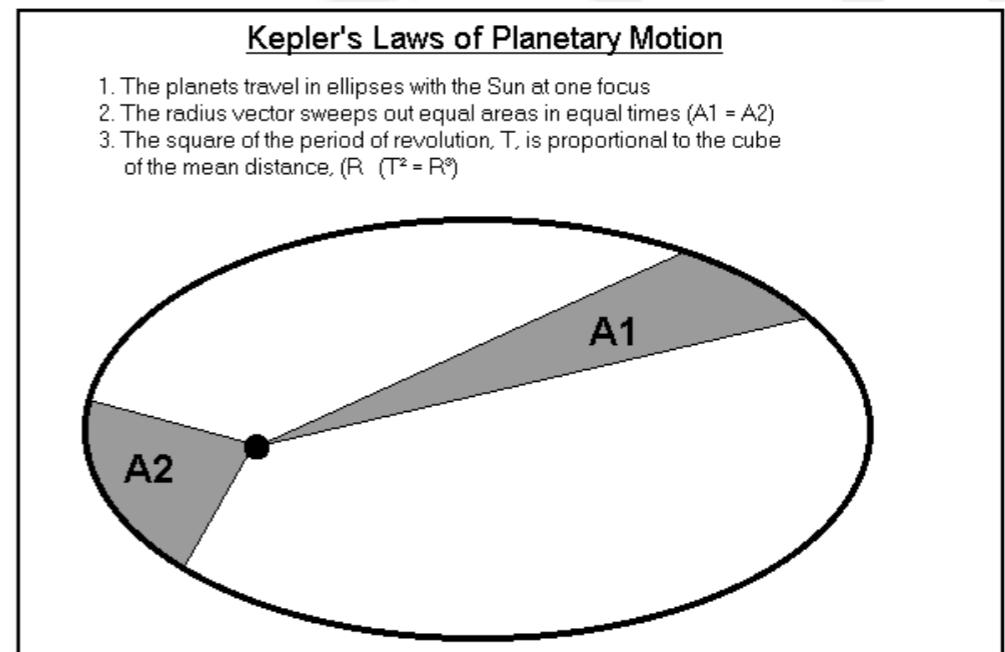
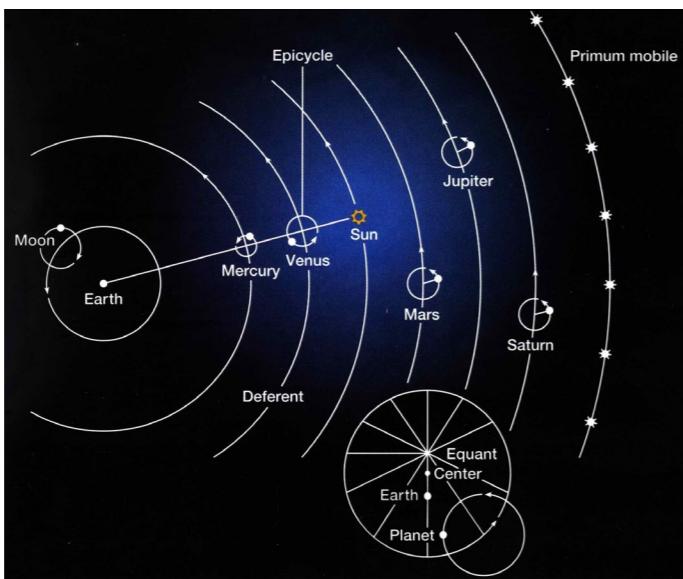
prefer the **simplest** hypothesis consistent with data



Ockham chooses a razor

Ockham's Razor

- Humans use Ockham's razor all the time
- Next number: 2, 4, 6, 8, ...
- Ptolemaeus vs Kepler

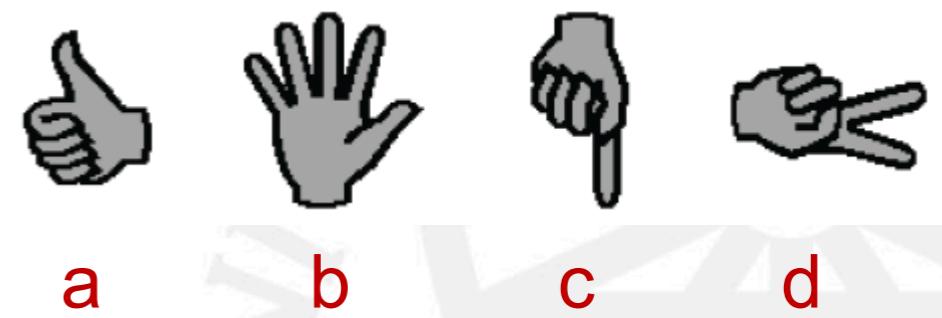
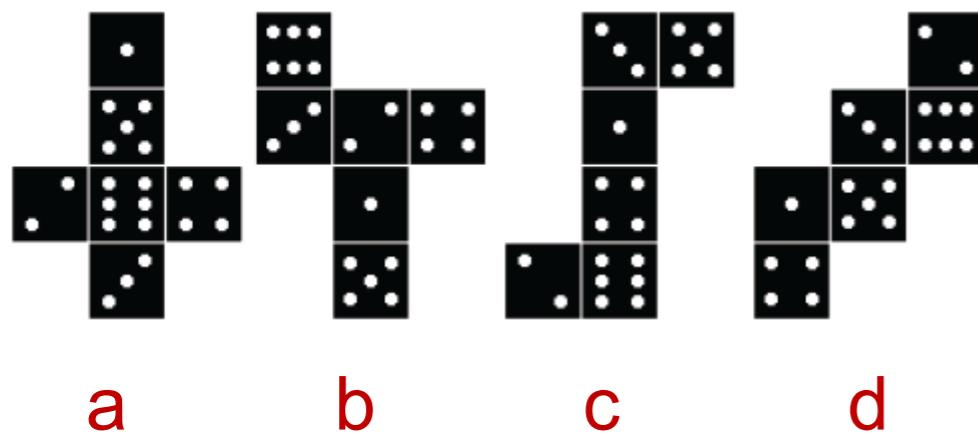
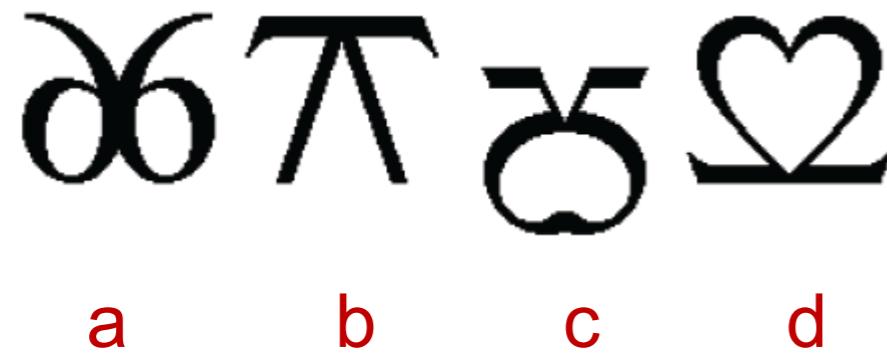


Which is the odd one out?

Go to kahoot.it



Odd-one-out



FLOW SNIP TRAP DRAWBACK

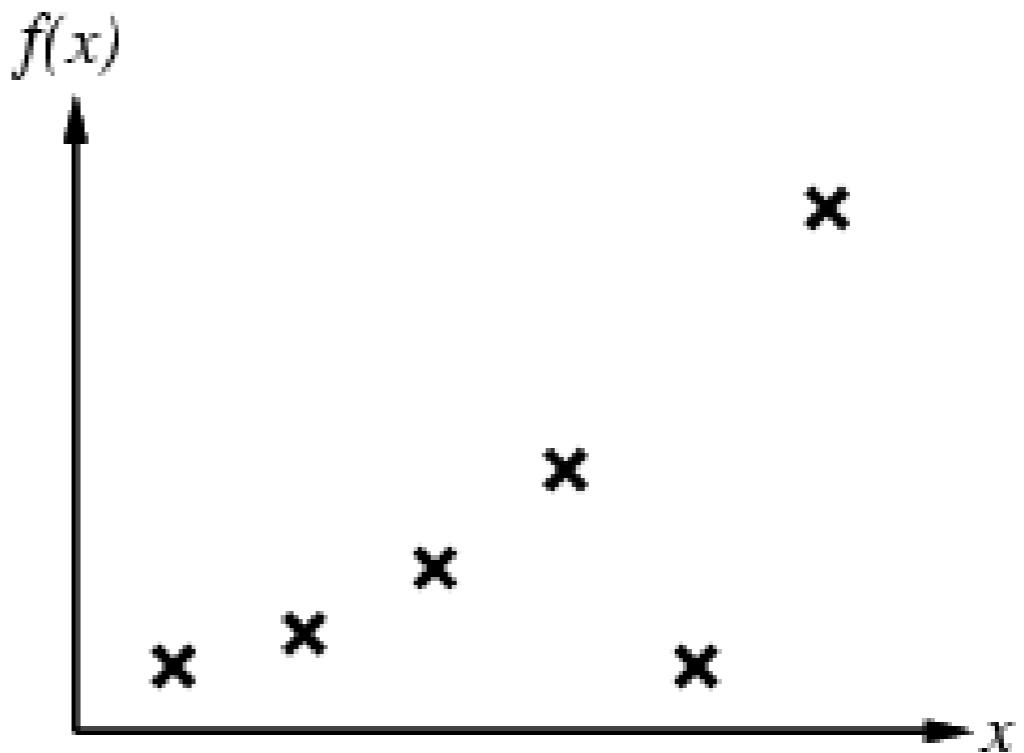
a b c d e

Supervised Learning

- Given:
 - a set of **input features** X_1, \dots, X_n
 - a set of **output targets** Y_1, \dots, Y_k
 - a set of **training examples** where the values for the input features and the target features are given for each example
 - a **new example**, where only the values for the input features are given predict the values for the target features for the new example.
- Regression: the output targets are **continuous** (predict the temperature tomorrow, stock values, ...)
- Classification: the output targets are **discrete** (true/false, yes/no, small/medium/large, ...)

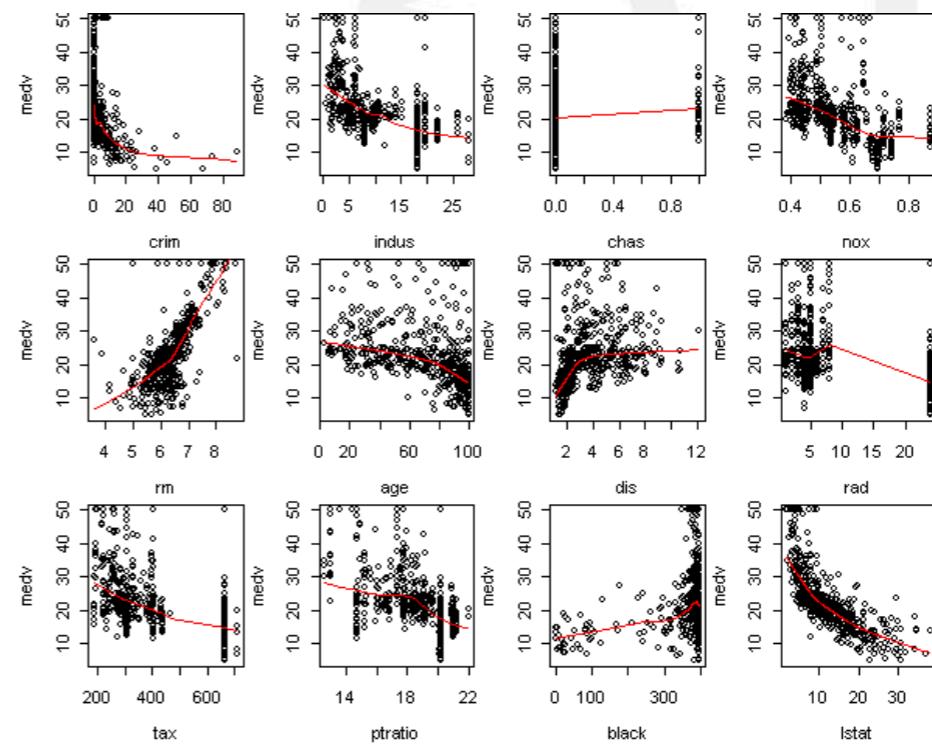
Regression

- Curve fitting
- Construct/adjust line to agree with outputs on training set

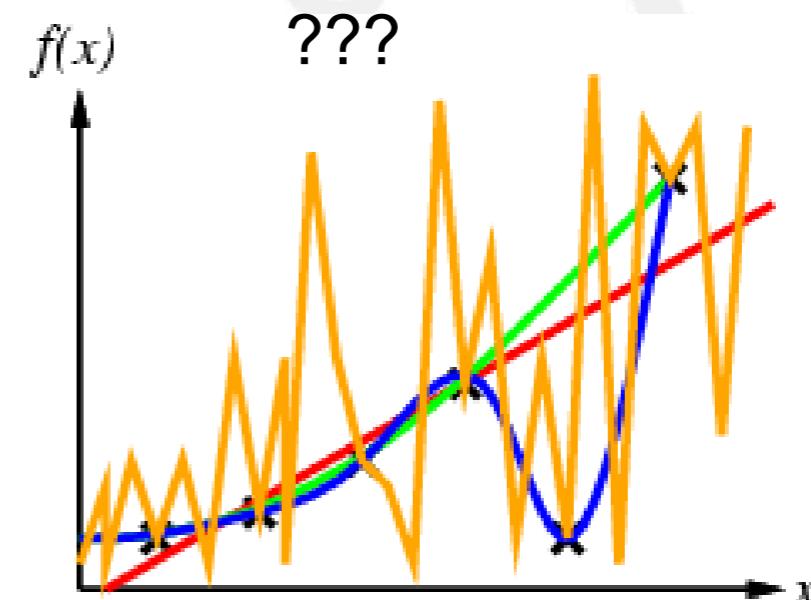
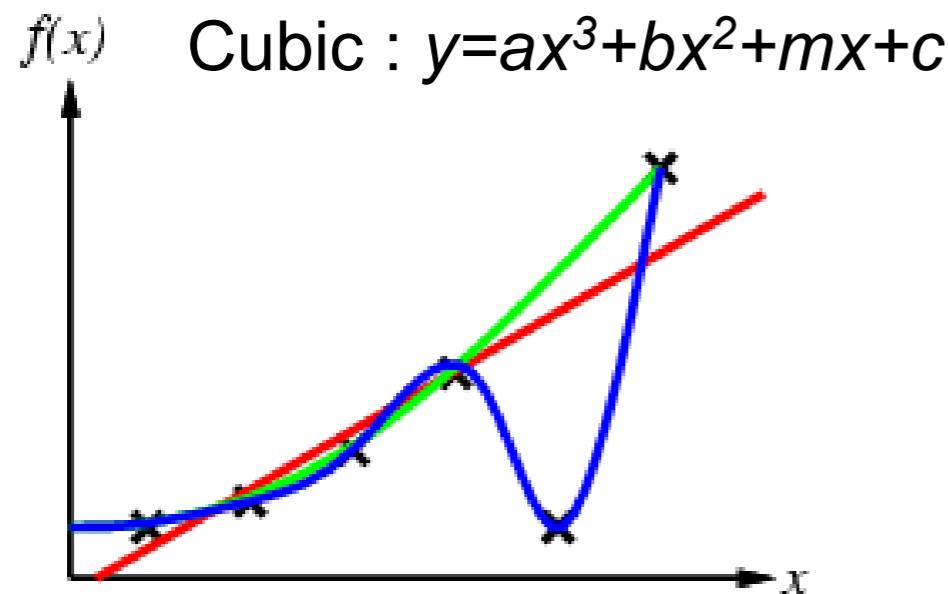
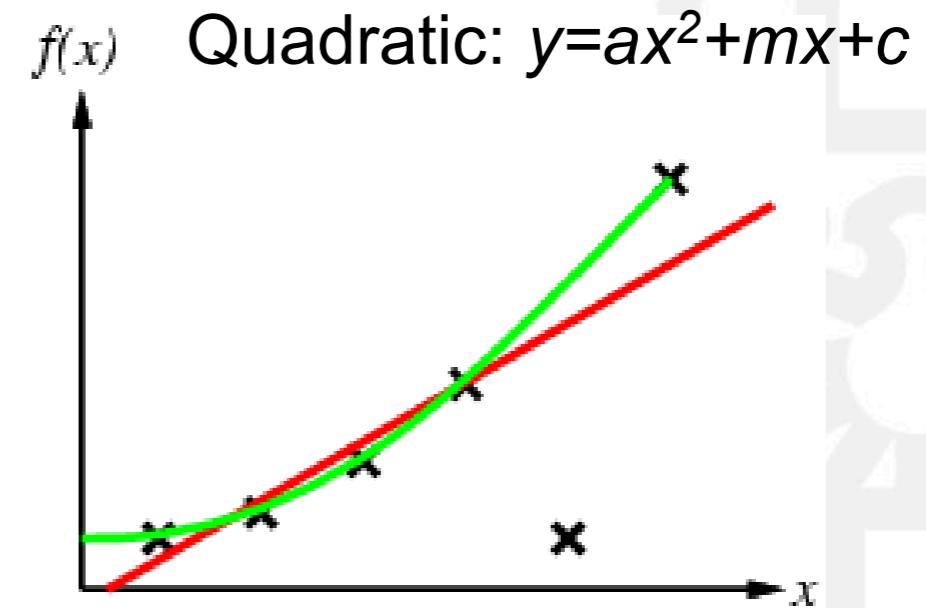
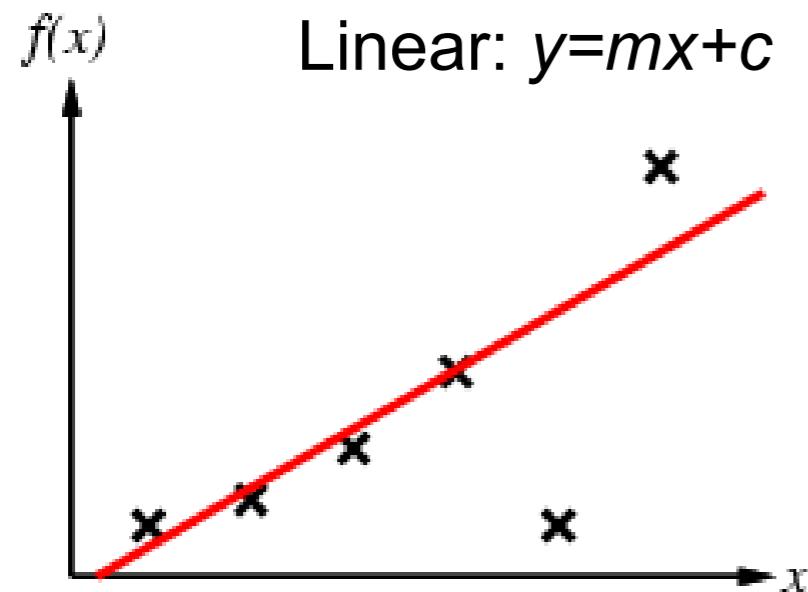


Example

- Predict housing price based on:
 - Number of rooms
 - Size
 - Neighborhood (how???)
 - ...
- Actual data set: Boston housing



Model Selection



Overfitting

- Which model is the best?
- Linear is simplest (2 numbers to find), but not such a good fit
- Cubic has zero error, but appears to **overfit**

Classification

- Boolean/binary classification: each example is classified as true or false

$$f(x) \rightarrow \{\text{true}, \text{false}\}$$

or as positive or negative

$$f(x) \rightarrow \{+1, -1\}$$

- Representation: each example is described by a set of attribute values (Boolean, discrete, continuous).

Evaluating Predictions

- Suppose Y is an output and i refers to an example:
 - Y_i is the value of output Y on example i
 - \hat{Y}_i is the predicted value of output Y on example i
 - The error of the prediction is a measure of how close \hat{Y}_i is to Y_i
- There are (too...) many possible errors that can be defined and measured

Measures of Errors

For continuous outputs (regression):

- Sum of squares error:

$$\sum_i (\hat{Y}_i - Y_i)^2$$

For binary outputs (classification) $Y \in \{0,1\}$, the output \hat{Y} of a classifier is often a probability between 0 and 1

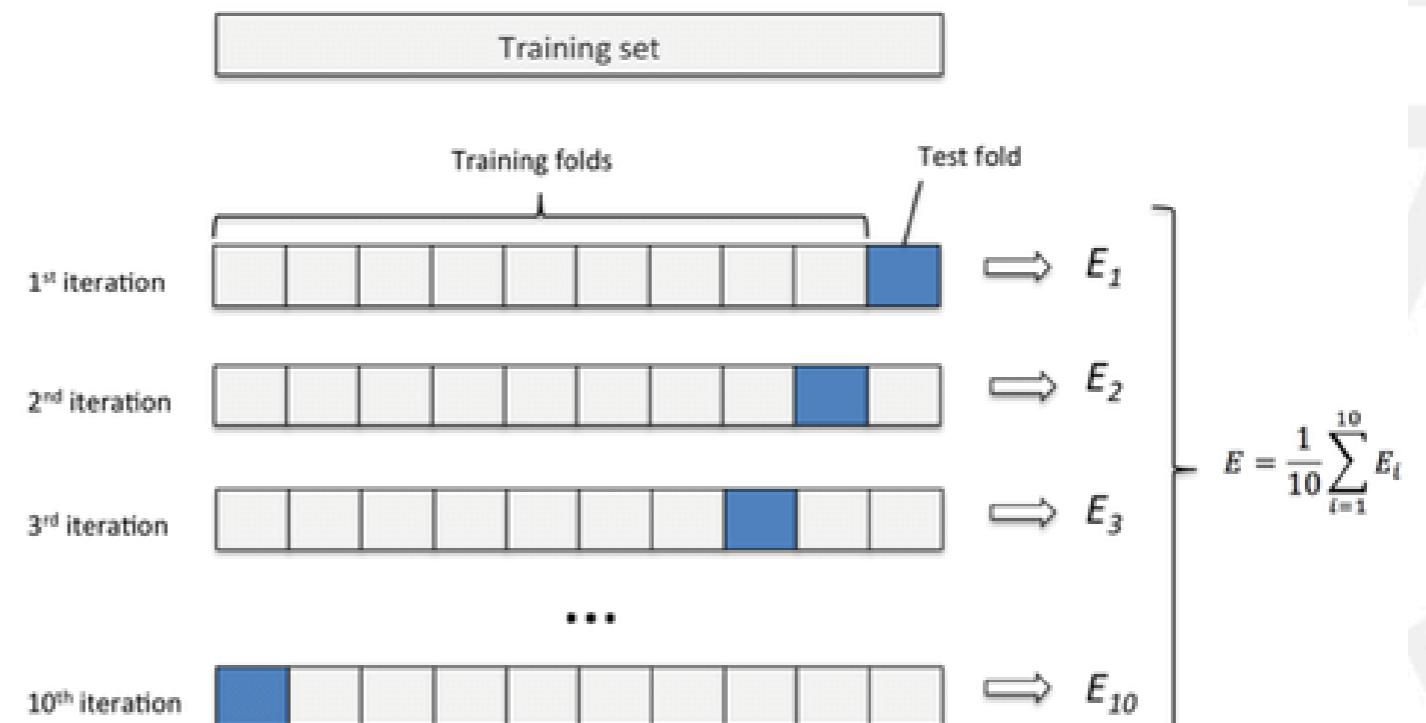
- Entropy:

$$-\sum_i [Y_i \log \hat{Y}_i + (1 - Y_i) \log (1 - \hat{Y}_i)]$$

Goal: try to get the predictions \hat{Y}_i as close as possible to the targets Y_i

Training and Test Sets

- **Warning:** predictions that minimize the error on given (training) data, do not always minimize the error for future predictions
- To evaluate how well a learner will work on future predictions, we divide the examples into:
 - **training examples** that are used to train the model
 - **test examples** that are used to evaluate the model
- **Cross-validation:** do this repetitively in a structured way



Reinforcement Learning

Tom Heskes

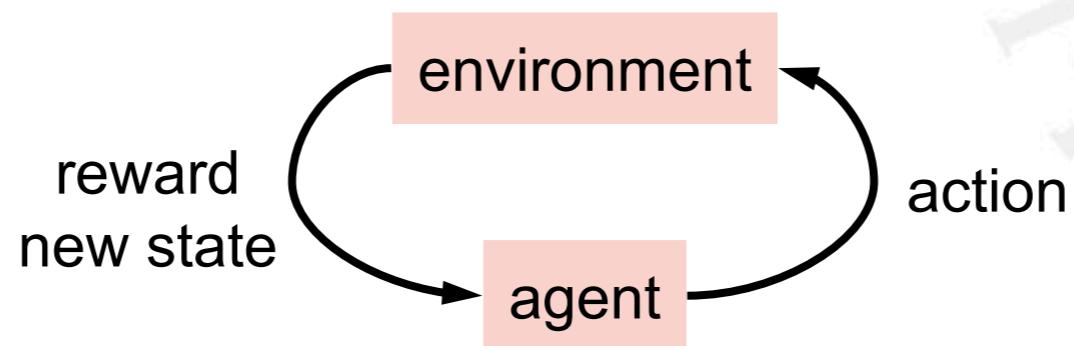


Learning goals: reinforcement learning

- Can you?
 - Tell what distinguishes a reinforcement learning from supervised learning
 - Explain the main elements: state, action, reward
 - Understand Bellman's equations and its relation to reinforcement learning
 - Understand the main ideas behind deep Q-learning

Before

- Supervised learning
 - classification, regression
- Unsupervised learning
 - clustering
- Reinforcement learning
 - more general than supervised/unsupervised learning
 - learn from interaction with the environment to achieve a goal



Next part

- Examples
- Defining a RL problem
 - Markov Decision Process
- Solving a RL problem
 - Q-learning



Robot in a room

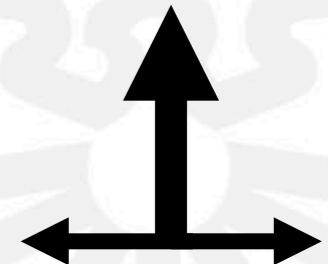
			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT

80% move UP

10% move LEFT

10% move RIGHT



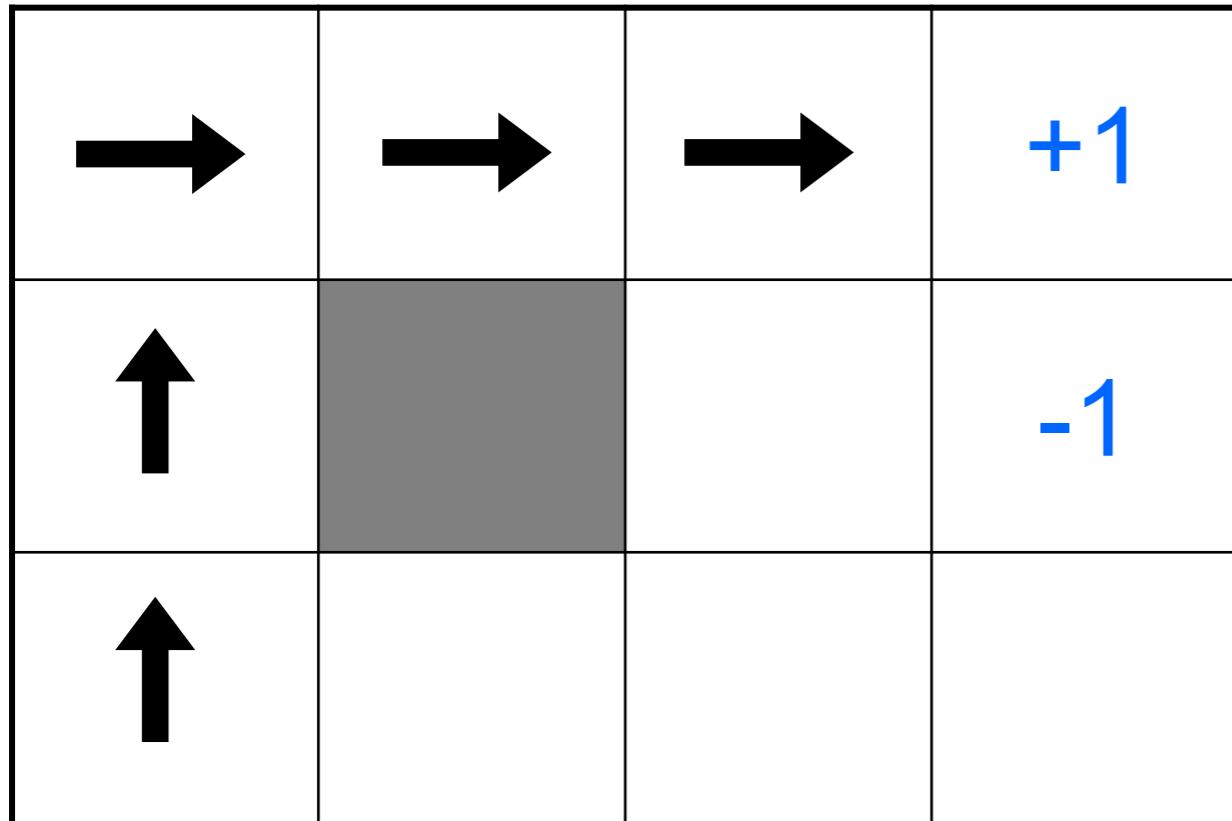
reward +1 at [4,3], -1 at [4,2]

reward -0.04 for each step

- states
- actions
- rewards

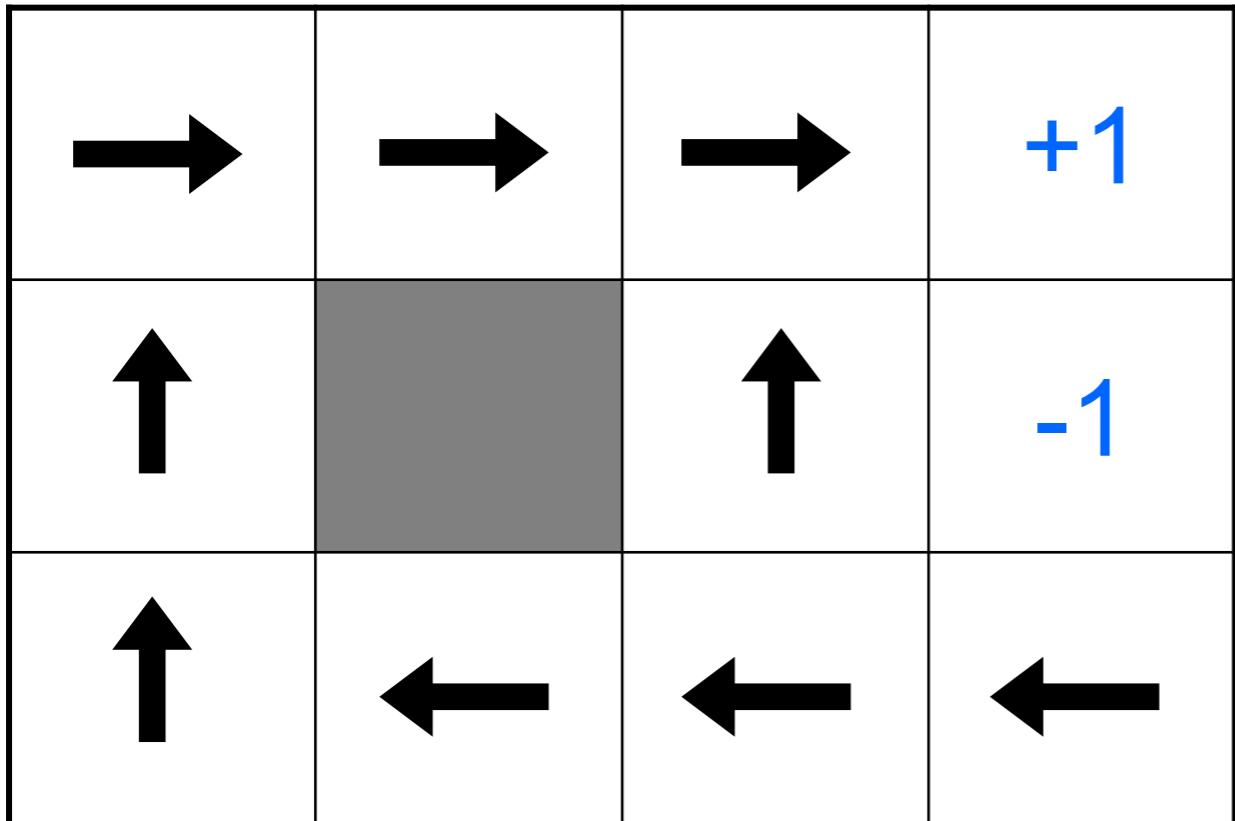
what's the solution?

Is this a solution?



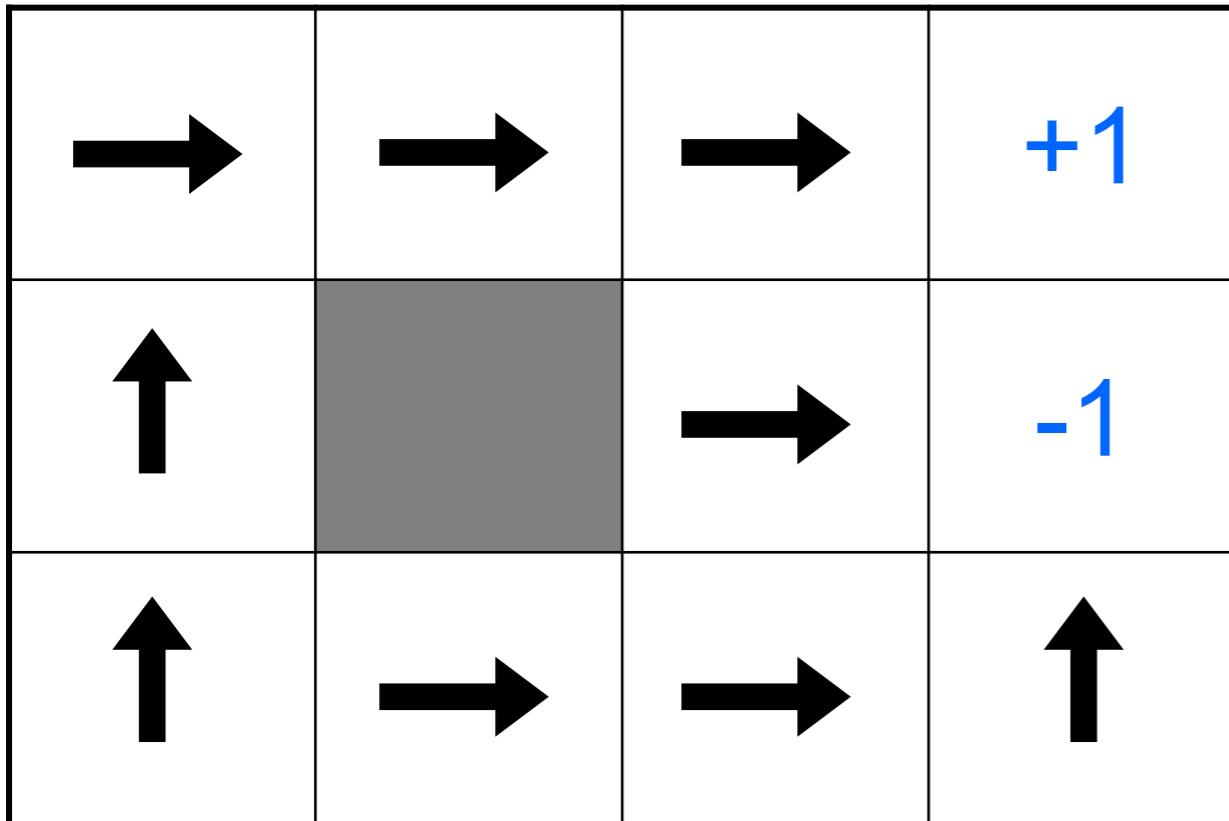
- Only if actions have deterministic consequences: not in this case, since actions are **stochastic**
- We need a **policy**: mapping from each state to an action

Optimal policy (1)



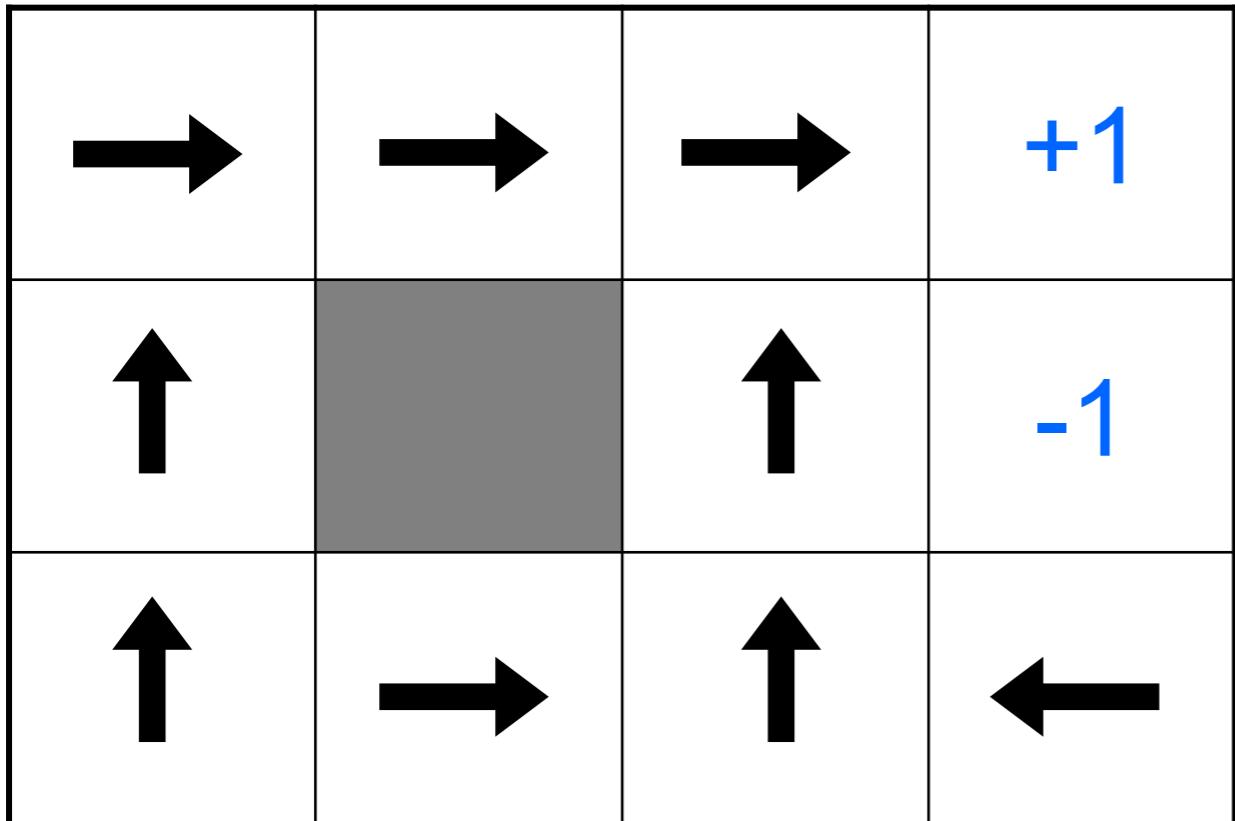
- with no reward/penalty for taking steps

Optimal policy (2)



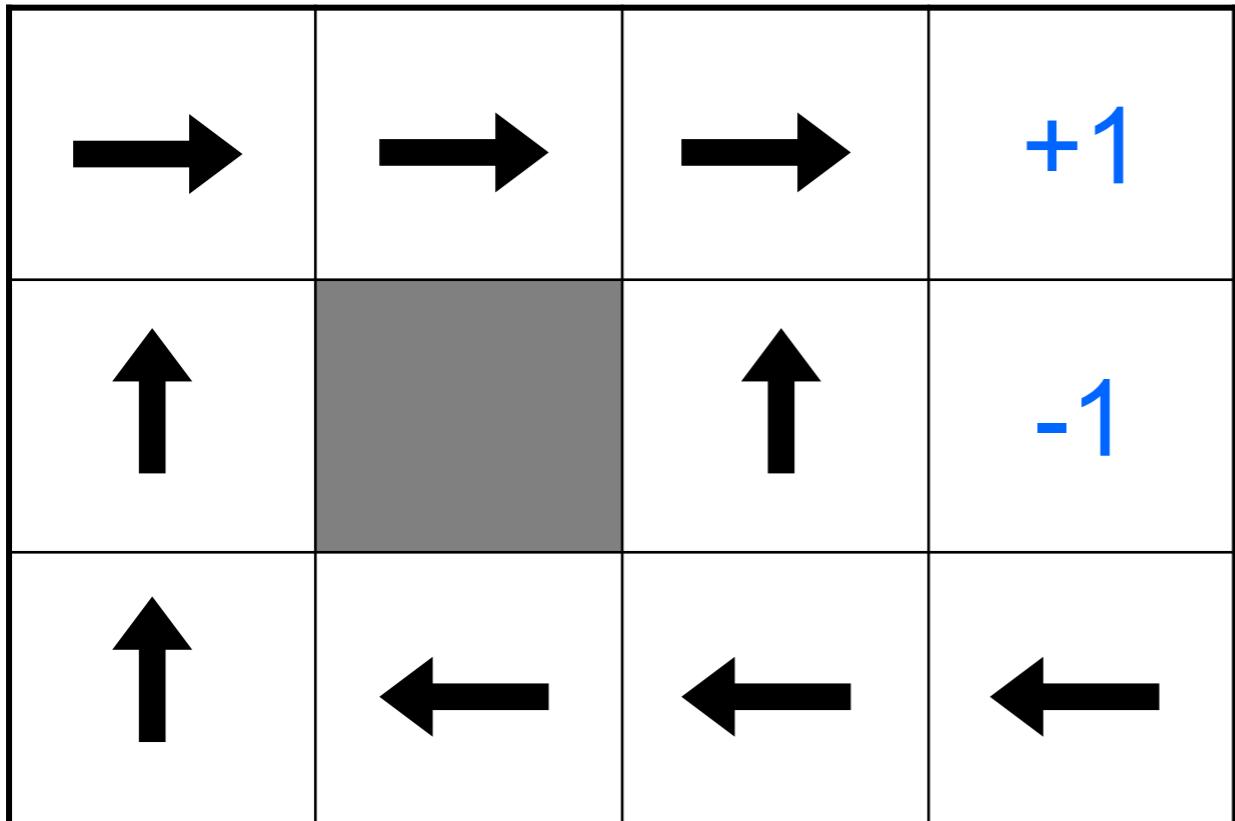
- with reward -2 for each step: strong penalty for moving around

Optimal policy (3)



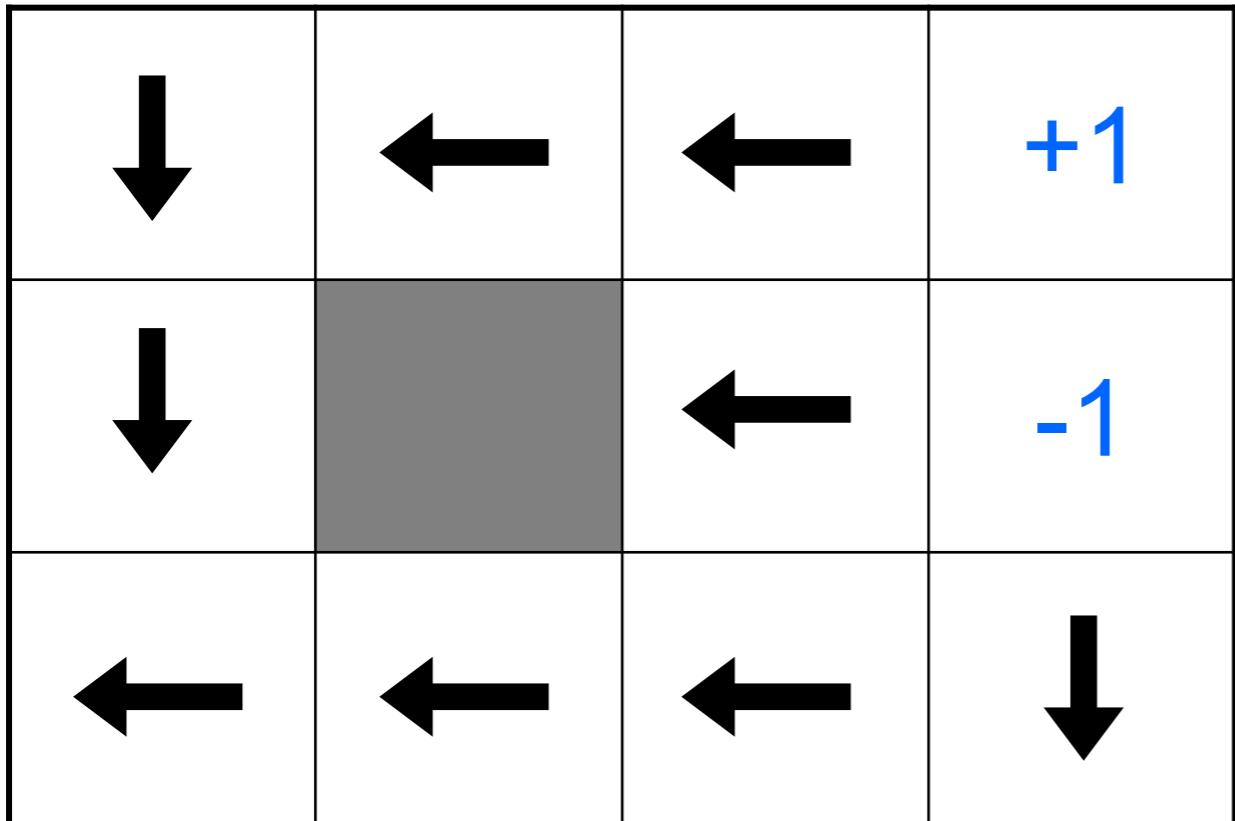
- with a small penalty, reward = -0.1, for taking steps

Optimal policy (4)



- with a tiny penalty, reward = -0.04, for taking steps

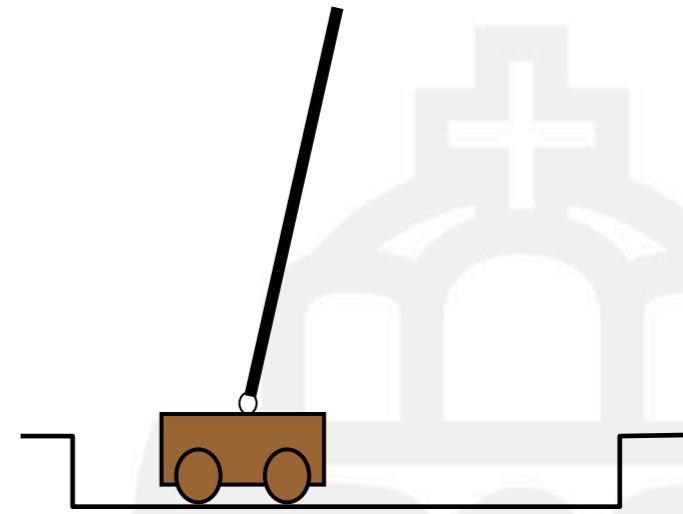
Optimal policy (5)



- with a positive reward = +0.01 for walking around

Other examples

- Pole-balancing
- Alphago/Alphazero
- Helicopter flying
- No teacher to tell you the good and bad actions
 - rewards could be delayed (a lot...)
- Similar to control theory
 - more general, fewer constraints
- Explore the environment and learn from experience
 - not just blind search, try to be smart about it

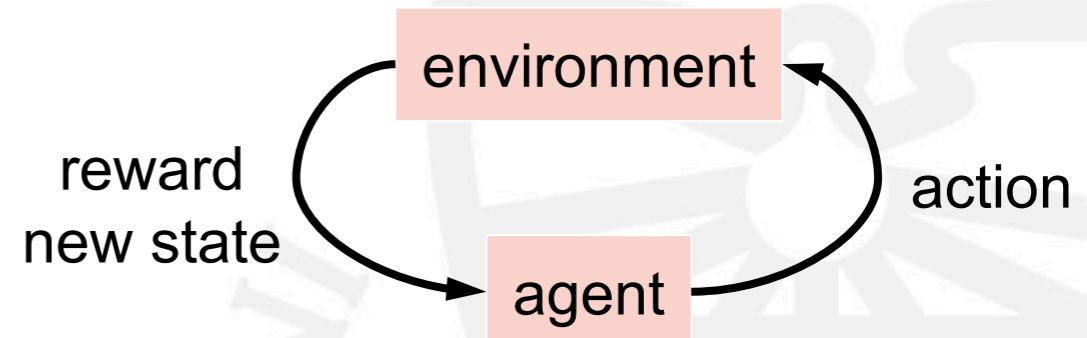


Reinforcement learning to play go



Markov Decision Process (MDP)

- Set of states S , set of actions A , initial state S_0
- Transition model $P(s,a,s')$
 - $P([1,1], \text{up}, [1,2]) = 0.8$
- Reward function $r(s)$
 - $r([4,3]) = +1$
- Goal: maximize cumulative reward in the long run
- Policy: mapping from S to A
 - $\pi(s)$ (deterministic action given state) or $\pi(s,a)$ (stochastic: probability of action given state)



The challenges of reinforcement learning

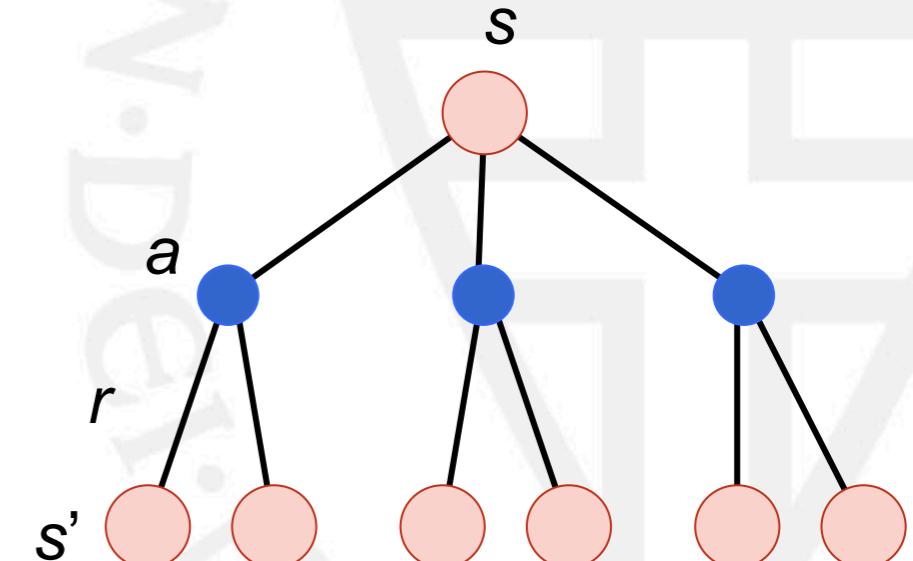
- Transitions and rewards usually not known
 - Need to experience and then estimate
- How to change the policy based on experience
 - Either indirectly, by first learning the value function
 - Or directly: policy-based reinforcement learning
- How to explore the environment
 - Exploration-exploitation trade-off
 - Often implemented by adding random “moves” with some probability

Computing return from rewards

- Episodic versus continuing tasks
 - Episodic: “game over” after a finite number of steps, e.g., playing a game of go
 - Continuing: “never”-ending, e.g., a personal assistance robot
- Additive rewards
 - $V(s_0, s_1, \dots) = r(s_0) + r(s_1) + r(s_2) + \dots$
 - Infinite value for continuing tasks
- Discounted rewards
 - $V(s_0, s_1, \dots) = r(s_0) + \gamma \times r(s_1) + \gamma^2 \times r(s_2) + \dots$
 - Value is bounded if the rewards are bounded
 - $\gamma < 1$ is the discount factor (typically 0.9 or higher)

Value functions

- **State value function:** $V^\pi(s)$
 - expected return when starting in state s and following policy π
- **State-action value function:** $Q^\pi(s,a)$
 - expected return when starting in state s , performing action a , and following policy π
- Useful for finding the optimal policy
 - Can estimate from experience
 - Pick the best action using $Q^\pi(s,a)$
- Famous **Bellman equation**



$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma V^\pi(s')] = \sum_a \pi(s, a) Q^\pi(s, a)$$

Optimal value functions

- There's a set of *optimal* policies
 - V^π defines a partial ordering on policies
 - The optimal policies have the same optimal value function:

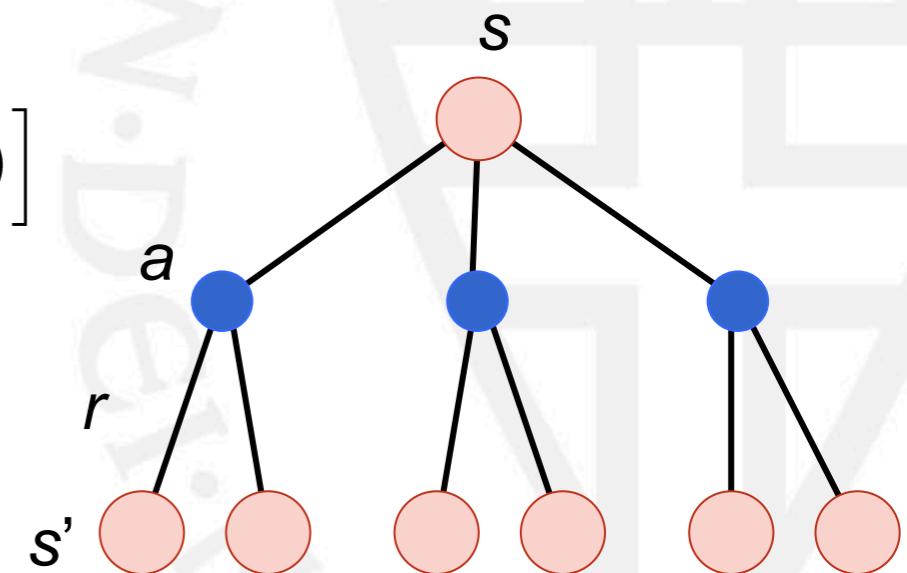
$$V^*(s) = \max_{\pi} V^\pi(s)$$

- Bellman optimality equation

$$V^*(s) = \max_a \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma V^*(s')]$$

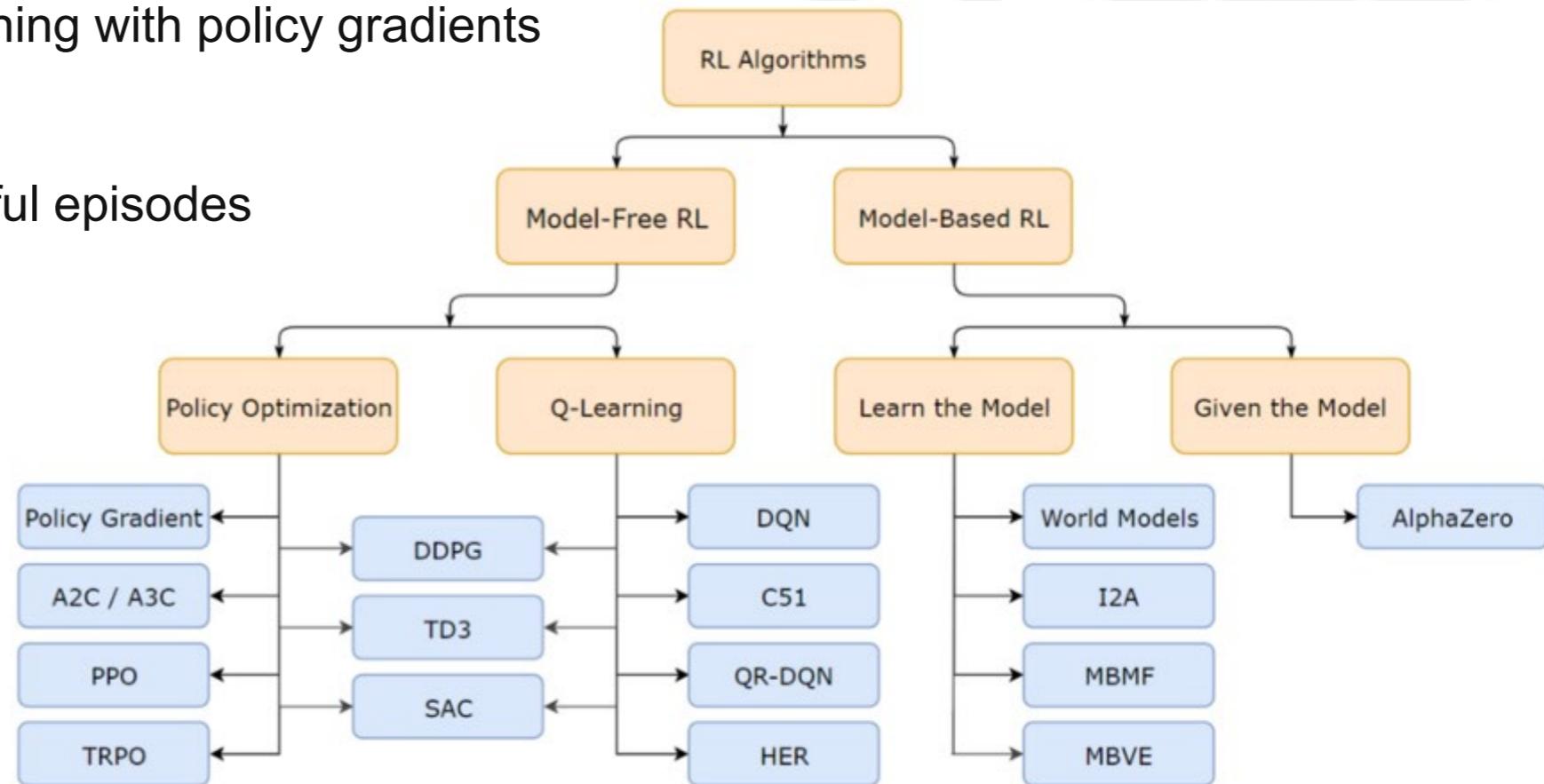
- System of many non-linear equations
 - “Just” solve for $V^*(s)$
 - Then easy to extract the optimal policy/policies
- Having $Q^*(s,a)$ makes it even simpler

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$



Reinforcement learning

- Approximate solution of Bellman's optimal equation
- Many different variants...
 - on-policy versus off-policy, model-free versus model-based
 - policy-based learning: directly learning the policy
 - deep reinforcement learning with policy gradients
 - actor-critic algorithms
 - self-imitation learning:
only remember successful episodes



Q-learning

- Use any policy to estimate

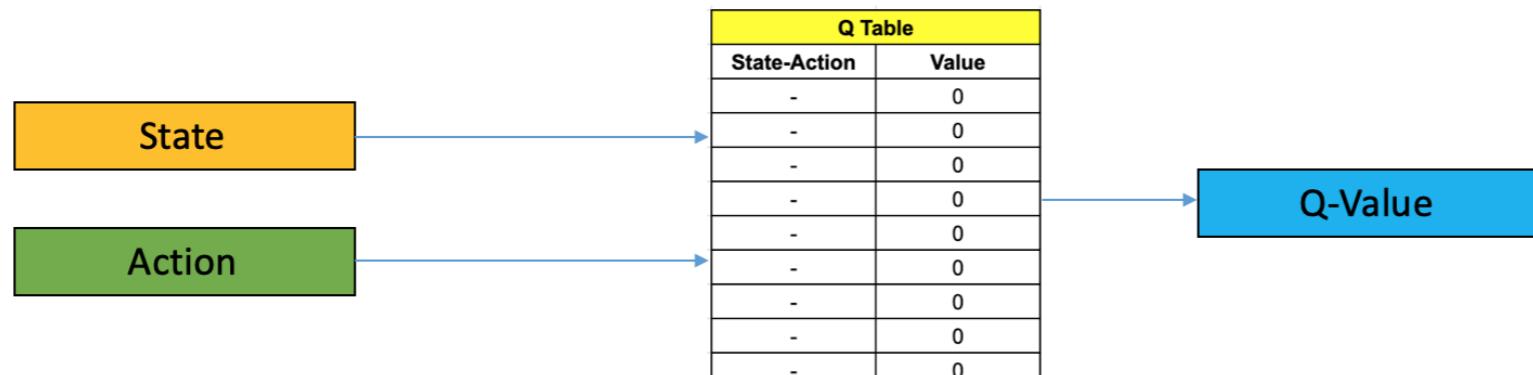
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- $Q(s,a)$ directly approximate the Belmann optimal $Q^*(s,a)$
- Does require updating every (s,a) pair
- Infeasible for a large number of states and actions: use function approximation, e.g., with a deep neural network → deep reinforcement learning
- Off-policy, temporal-difference algorithm
- New tricks abound, e.g., experience replay, curriculum learning (start simple)

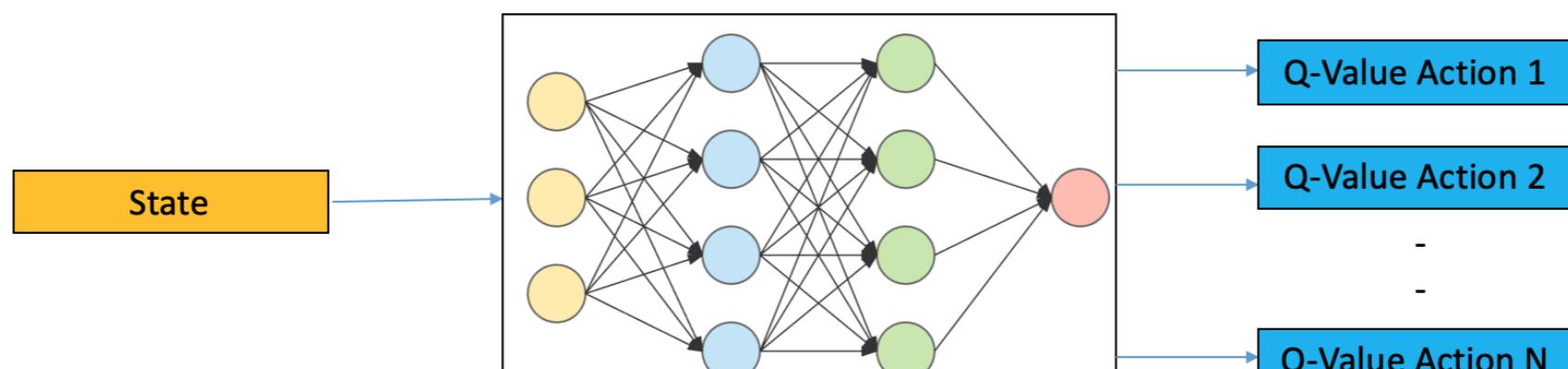
temporal difference

estimate of future
optimal value

Deep Q-learning



Q Learning



Deep Q Learning

Learning to play Atari games



Reinforcement learning outlook

- Learning based on very little information
- Need to be able to repeat many, many times (lots of episodes)...
 - which is why you see a lot of success in games, much less in “real-world” problems
- Many different variants and clever tricks

