Ahmed Ibrahim

ECE 9039/9309 MACHINE LEARNING

Last Lecture

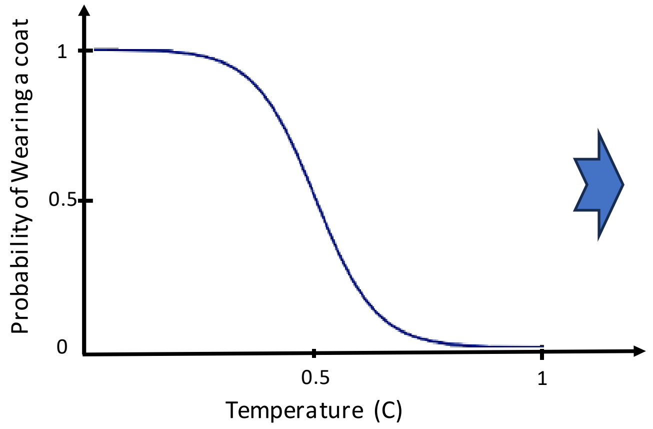
- Principal Component Analysis (PCA) (Tut)
- Classification
  - How does classification work?
  - Types of Classifiers
  - Linear Classifiers (Tut)
- Classification Metrics (Tut)
  - A confusion matrix
- Metrics vs Loss
- Logistic Regression

Outlines

- Probabilistic Classifiers
  - Logistic Regression
  - Naïve Bayes Classifier
- Instance-based classifiers
  - K-Nearest Neighbors (K-NN)
- More Classifiers
  - Linear SVM
  - Kernel SVM
- Clustering
  - K-means
  - K-means++
  - Gaussian Mixture Model

Probabilistic Classifiers

Recall: Capturing Uncertainty

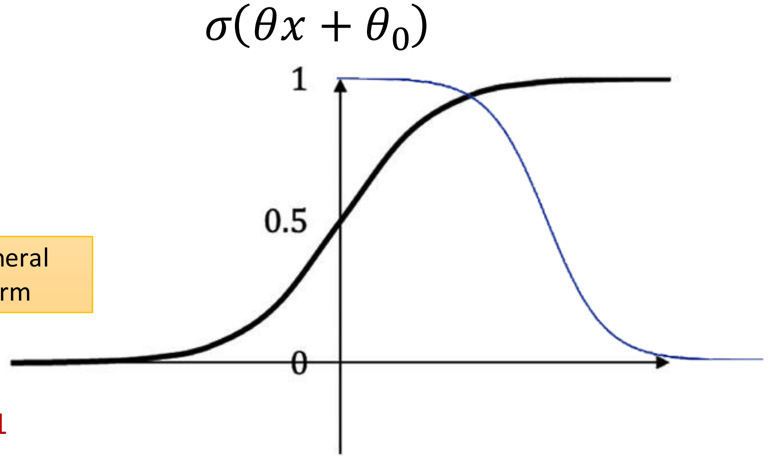


$$\sigma(\theta x + \theta_0) = \frac{1}{1 + e^{-(\theta x + \theta_0)}}$$

$$\hat{P}(y|x,\theta) = \frac{1}{1 + e^{-(\theta x + \theta_0)}}$$

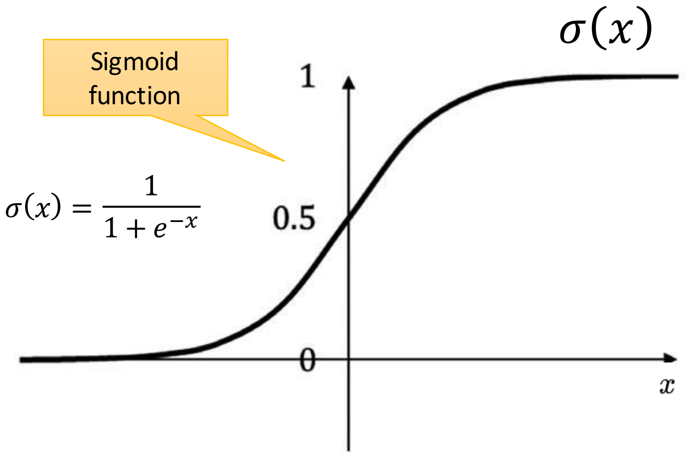
$$\hat{P}(y|x,\theta) = \frac{1}{1 + e^{-(\theta^T x + \theta_0)}}$$ … equ$_1$

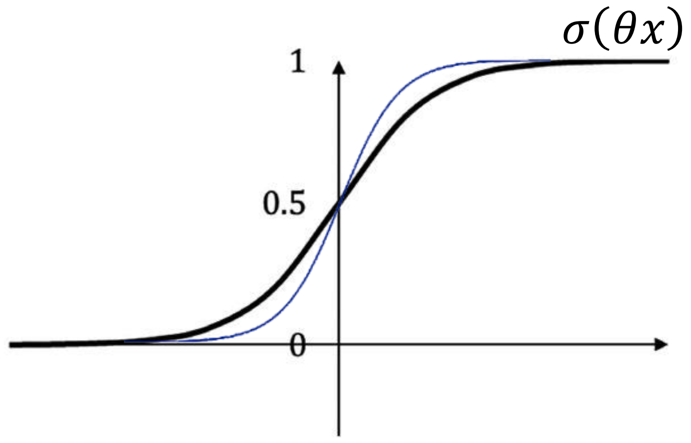
General form

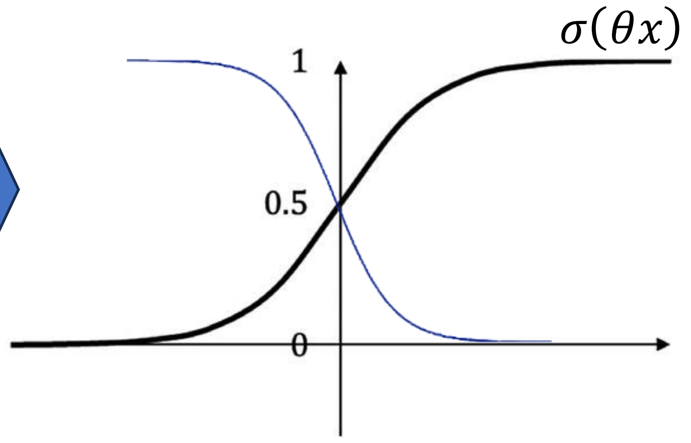


Sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$







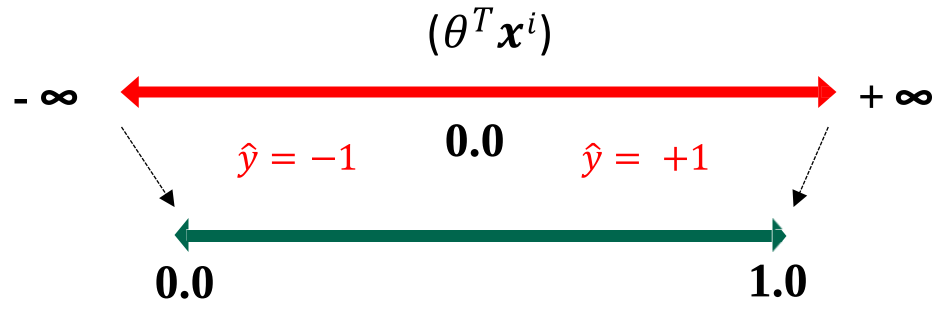
Recall: Using Probability in Classification

$$\hat{P}(y|x,\theta) = \frac{1}{1+e^{-(\theta^T x+\theta_0)}}$$

Simplify -> $\hat{P}(y|x,\theta) = \frac{1}{1+e^{-(\theta^T x)}}$

$(\theta^T x) \rightarrow +\infty\,,\ \hat{P}= 1$

$(\theta^T x) \rightarrow -\infty\,,\ \hat{P}= 0$



Logistic Regression

- $\hat{P}$ -> estimate of class probability
- Learning parameters of logistic regression:

Training Data: m Observations (x^i, y)

x^1	x^2	y
2	1	+1
1	1	-1
0	1	-1
2	0	+1
3	1	+1

To learn $\hat{\theta}$, we need to define empirical classification error

Likelihood Function

- Empirical classification error => Likelihood function
- No $\hat{\theta}$ achieves perfect prediction (normally)
- Likelihood $l(\theta)$ => Measures the quality fit for the model with coefficients θ.

Likelihood Function (cont.)

x_1	x_2	y
3	1	+1

If the Model is good, the Model should predict +1

Pick θ to maximize

$$P(y=+1|\boldsymbol{x},\theta)=P(y=+1|\boldsymbol{x}[1]=3,\boldsymbol{x}[2]=1,\theta)$$

- Your classifier is extremely good if:

$$P(y=+1|\boldsymbol{x},\theta)\sim 1$$

Likelihood Function (cont.)

x_1	x_2	y
0	2	−1

If the Model isgood, the Model should predict −1

Pick θ to maximize

$$P(y=-1|\boldsymbol{x},\theta) = P(y=-1|\boldsymbol{x}[1]=0, \boldsymbol{x}[2]=2, \theta)$$

- Your classifier is extremely good if:

$$P(y=-1|\boldsymbol{x},\theta) \sim 0$$

Likelihood Function (cont.)

x_1	x_2	y	Choose θ to maximize
2	1	+1	$P(y = +1 \mid \boldsymbol{x}_1 = 2, \boldsymbol{x}_2 = 1, \theta)$
1	1	-1	$P(y = -1 \mid \boldsymbol{x}_1 = 1, \boldsymbol{x}_2 = 1, \theta)$
0	1	-1	$P(y = -1 \mid \boldsymbol{x}_1 = 0, \boldsymbol{x}_2 = 1, \theta)$
2	0	+1	$P(y = +1 \mid \boldsymbol{x}_1 = 2, \boldsymbol{x}_2 = 0, \theta)$
3	1	+1	$P(y = +1 \mid \boldsymbol{x}_1 = 3, \boldsymbol{x}_2 = 1, \theta)$
2	2	-1	$P(y = -1 \mid \boldsymbol{x}_1 = 2, \boldsymbol{x}_2 = 2, \theta)$
4	4	-1	$P(y = -1 \mid \boldsymbol{x}_1 = 4, \boldsymbol{x}_2 = 4, \theta)$
…	..	…	…

How do we combine them into one single measure of Quality?

$$l(\theta) = P\left(y^{(1)} \middle| \boldsymbol{x}^{(1)}, \theta\right) . P\left(y^{(2)} \middle| \boldsymbol{x}^{(2)}, \theta\right) \ldots\ldots P\left(y^{(m)} \middle| \boldsymbol{x}^{(m)}, \theta\right)$$

$$l(\theta) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)$$

Likelihood Function (cont.)

$$\ln \frac{a}{b} = \ln a - \ln b$$

$$\ln ab = \ln a + \ln b$$

Log Review

- In probability theory, multiplying probabilities is a fundamental concept used to find the joint probability of independent events occurring.
- Goal=> Choose the coefficients θ that **maximizes** the likelihood. This is done by multiplying probabilities.

$$l(\theta) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)$$

- The Math is simplified by using log-likelihood: Taking the natural log

$$\ln\left(l(\theta)\right) = ll(\theta) = ln\left(\prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)\right)$$

Likelihood Function (cont.)

$$\ln(l(\theta)) = ll(\theta) = ln\left(\prod_{i=1}^{m} P(y^{(i)}|\boldsymbol{x}^{(i)}, \theta)\right)$$

$$\ln(l(\theta)) = ll(\theta) = \sum_{i=1}^{m} ln\left(P(y^{(i)}|\boldsymbol{x}^{(i)}, \theta)\right)$$

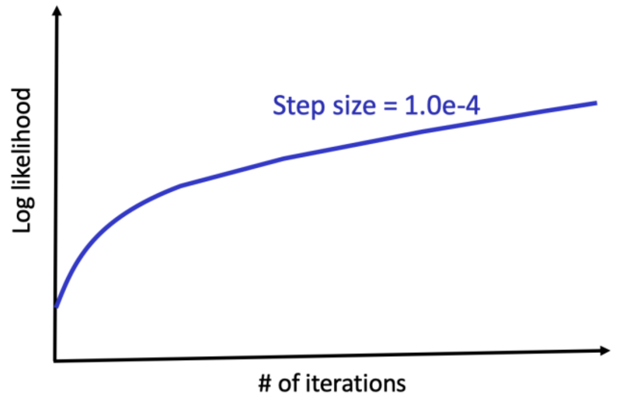
$$\ln\frac{a}{b} = \ln a - \ln b$$

$$\ln ab = \ln a + \ln b$$

Log Review

- Finding θ:

Using the Gradient Descend Algorithm



Logistic Regression Model

- The probability model to predict y = -1, could be written as:

$$P(y = -1 \mid \boldsymbol{x}, \theta) = 1 - P(y = +1 \mid \boldsymbol{x}, \theta)$$

- From equ1 (in slide #5), we could say:

$$P(y = -1 \mid \boldsymbol{x}, \theta) = 1 - P(y = +1 \mid \boldsymbol{x}, \theta)$$

$$= 1 - \frac{1}{1+e^{-\theta^T \boldsymbol{x}}}$$

$$= \frac{1+e^{-\theta^T \boldsymbol{x}} - 1}{1+e^{-\theta^T \boldsymbol{x}}}$$

$$= \frac{e^{-\theta^T \boldsymbol{x}}}{1+e^{-\theta^T \boldsymbol{x}}}$$

$$P(y = +1 \mid \boldsymbol{x}, \theta) = \frac{1}{1+e^{-\theta^T \boldsymbol{x}}}$$

$$P(y = -1 \mid \boldsymbol{x}, \theta) = \frac{e^{-\theta^T \boldsymbol{x}}}{1+e^{-\theta^T \boldsymbol{x}}}$$

$$\ln(l(\theta)) = ll(\theta) = \sum_{i=1}^{m} ln\left(P(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta)\right)$$

Naïve Bayes Classifier

Bayes Rule!

Bayes' Theorem, is a fundamental principle in probability theory that describes how to update the **probabilities** of hypotheses when given **evidence**.

- Problem statement:
  - Given features X1,X2,…,Xn
  - Predict a label Y
- A good strategy is to predict:

$$\arg\max_Y P(Y|X_1,\ldots,X_n)$$

- Use Bayes Rule!

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

You would test + if you had a disease

The prior probability: the frequency of the disease in the population

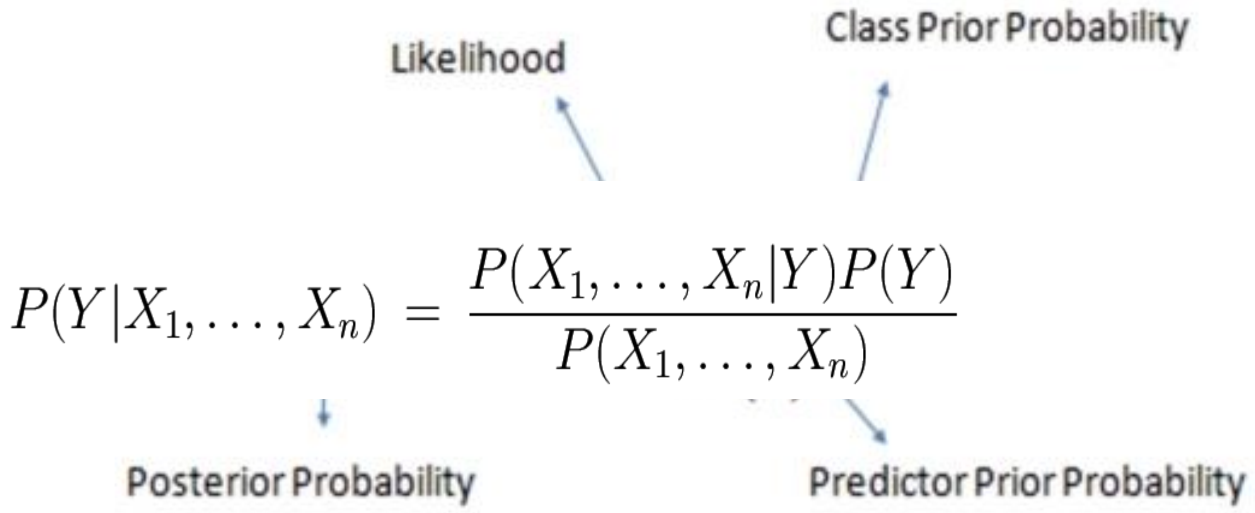
Having a disease

You tested +

Probability of texting +

Naïve Bayes Classifier

- Another lazy learning algorithm
- Based on the Bayes Theorem
- Every feature being classified is independent of the value of any other feature



$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Practical Exercise

- Step 1: Convert the data set into a frequency table

Part of the dataset

Weather	Playing outside
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Total	5	9

Practical Exercise (cont.)

- Step 2: Create a Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Likelihood table				
Weather	No	Yes		
Overcast	-	4	= 4/14	0.28
Rainy	3	2	= 5/14	0.36
Sunny	2	3	= 5/14	0.36
All	5	9		
	= 5/14	= 9/14		
	0.36	0.64		

Practical Exercise (cont.)

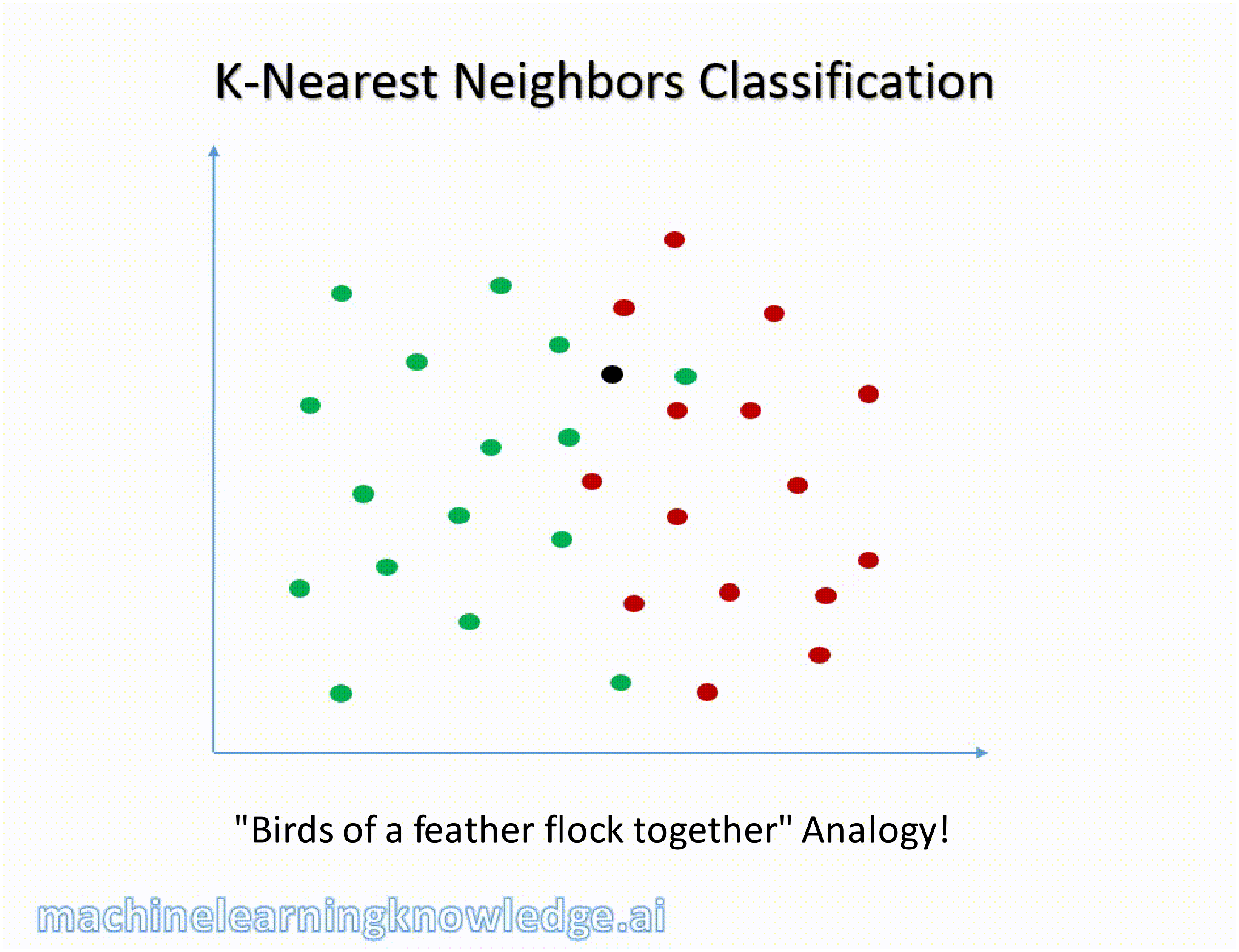
- Step 3: Now, use the Naïve Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.
- Now, you can figure out if the players will play if the weather is sunny by using the Likelihood table as follows:
  - P(Sunny|Yes) = 3/9 = 0.33; P(Sunny) = 5/14 = 0.36; P(Yes)= 9/14 = 0.64
  - P(Yes|Sunny) = P(Sunny|Yes) * P(Yes) / P(Sunny)
  - Now, P(Yes|Sunny) = 0.33 * 0.64 / 0.36 = 0.60

More Supervised Learning Algorithms: KNN & SVM

K-Nearest Neighbors (K-NN)

K-Nearest Neighbors (K-NN)

- What is KNN? A **non-parametric, instance-based** learning algorithm for classification and regression.
- Classifies new data points based on the k nearest data points in the training set.
- A lazy supervised learning



K-Nearest Neighbors (K-NN)

- It is called a **lazy learning** algorithm because it doesn't have a specialized training phase.
  - **Eager Learning** – This refers to a type of ML algorithm that constructs a **generalized model** before receiving data for prediction.
- K-Nearest Neighbor's most significant advantage is that the algorithm can make predictions **without training**; new data can be added.
- The most significant drawback of the algorithm is its struggle to compute distances effectively when dealing with data of high dimensionality.

The KNN Algorithm

1. **Choose k** – Define the number of nearest neighbors to consider (e.g., k=3, k =5).
2. **Distance Calculation** – Measure the distance between the new data point and each training point using a distance metric (e.g., Euclidean distance).
3. **Identify Neighbors** – Select the k data points closest to the new point.
4. **Majority Vote** – Determine the most frequent class label among the k neighbors.
5. **Prediction** – Assign the new data point to the majority class.

Choosing the Right k

- k affects model complexity and accuracy.
- Low k (high variance): Overfitting, sensitive to noise.
  - The prediction is based on a small number of nearest neighbors, so it's largely influenced by their specific values.
- High k (high bias): Underfitting, less sensitive to noise.
  - The prediction is more stable and generalizes better, but it may also be too simple to capture the underlying pattern well, leading to high bias.
- Experimentation and validation techniques are crucial for finding the optimal k.

Choosing the Right k (cont.)

- **Grid Search**
  - This extensive approach involves trying out a predefined range of k values and evaluating the model's performance on each one. Metrics like accuracy, precision, recall, or F1 score are used for evaluation.
  - The k value that yields the best performance metric is chosen as the optimal k.
- **Cross-validation** (more details in later slides)
  - The model is trained on one-fold while the remaining folds are used for validation.
  - This process is repeated for each fold, and the average performance across all folds is calculated for each k value. The k value with the highest average performance is chosen as optimal k.

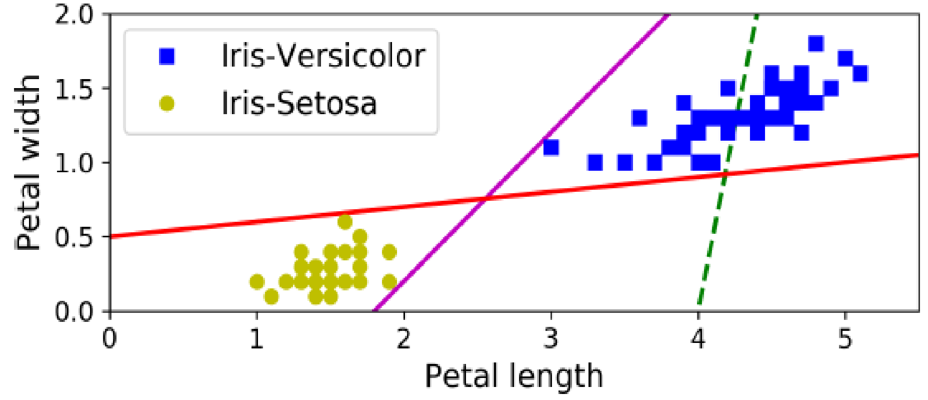
K-Nearest Neighbors (cont.)

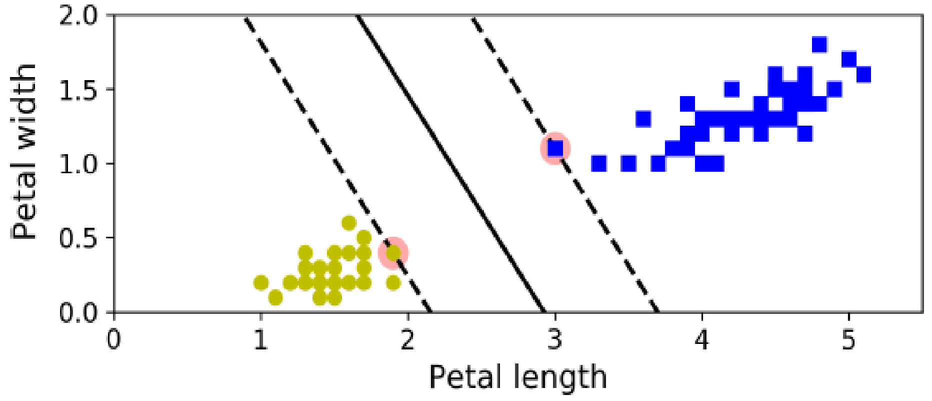
- For regression, KNN predicts the output for a new data point by averaging the values of the K nearest neighbors to that data point and using this average as the predicted value for the new data point.
- KNN is a non-parametric method, meaning it does not make any <u>underlying assumptions about the distribution of data</u>, which can be an advantage in certain settings. However, it can be computationally expensive for large datasets since it requires calculating the distance between the new point and all points in the dataset.
- Additionally, KNN can suffer from the **curse of dimensionality** as the number of features grows, making the <u>distance metric less meaningful</u>.

Support Vector Machine

Support Vector Machine

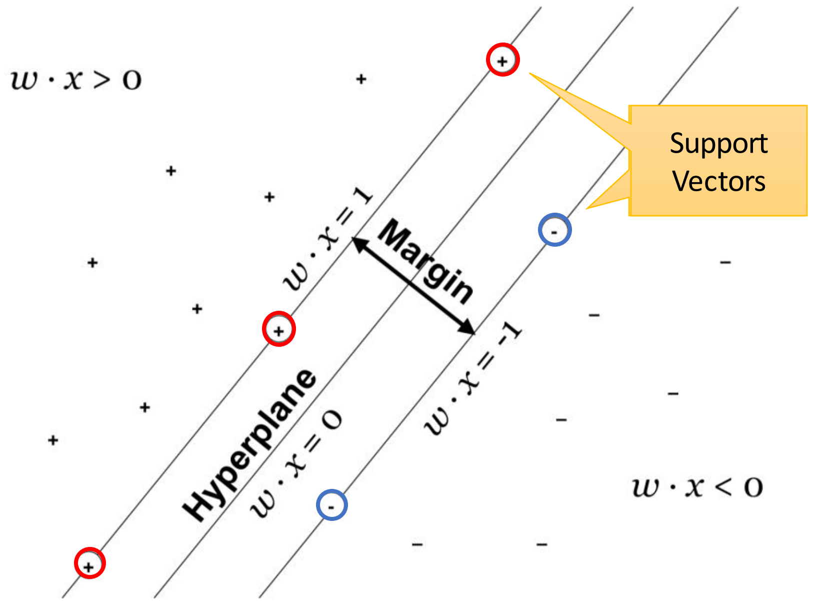
- A very powerful ML model capable of performing **linear** or **nonlinear** classification, regression, and even outlier detection.
- It is one of the most popular models in ML; SVMs are particularly well suited for the classification of complex but small- or medium-sized datasets.
- Think of an SVM classifier as fitting the widest possible street between classes (shown by parallel dashed lines)





Support Vector Machine

- **Hyperplane ($\boldsymbol{w.x = 0}$)** – This is the decision boundary that separates the two classes in the feature space. The hyperplane is defined by the weight vector w and the feature vector $\boldsymbol{x}$, and the equation $\boldsymbol{w.x = 0}$ represents points that lie exactly on this decision boundary.
- **Margins ($\boldsymbol{w.x = \pm 1}$)** – These are the boundaries of the margin around the hyperplane. The margins are parallel to the hyperplane and are at the maximum distance from it where the nearest training examples (the support vectors) can be found. The equations $\boldsymbol{w.x = 1}$ and $\boldsymbol{w.x = -1}$ represent the margins. The width of the margin is inversely proportional to the norm of w.



- **Support Vectors** – These are the data points that lie closest to the decision boundary and are the most difficult to classify. They are the critical elements of the training set because they define the margin.
- **Margin Width** – This is indicated by the bidirectional arrow perpendicular to the hyperplane, showing the distance between the two margins, which is maximized in SVM to increase the classifier's generalization ability.

Feature Scaling

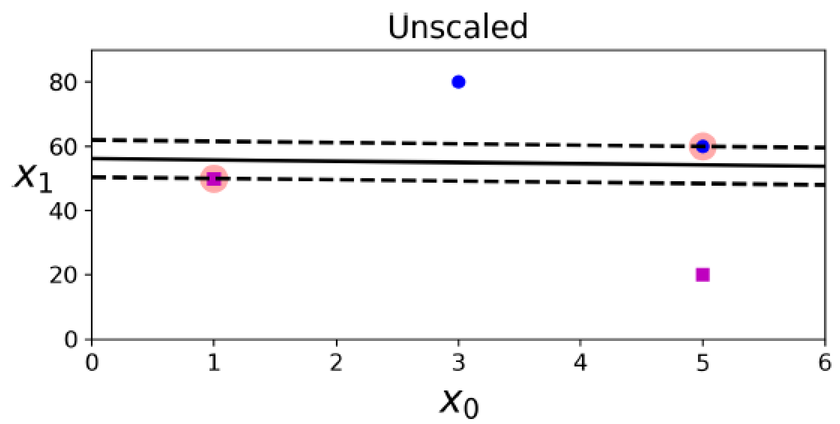


Figure A

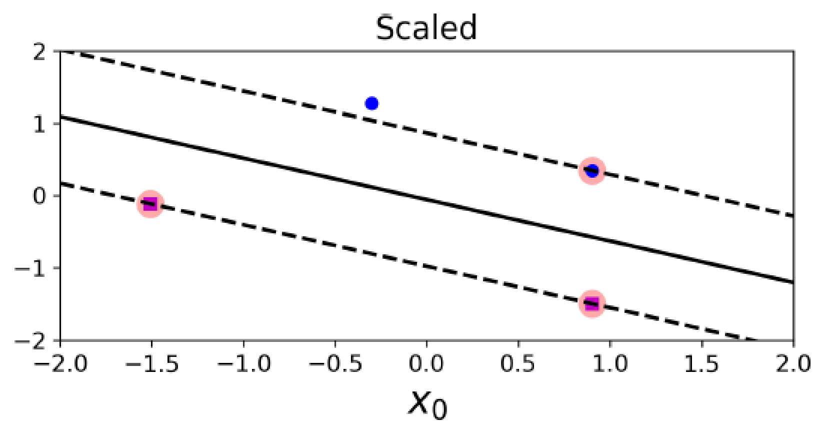
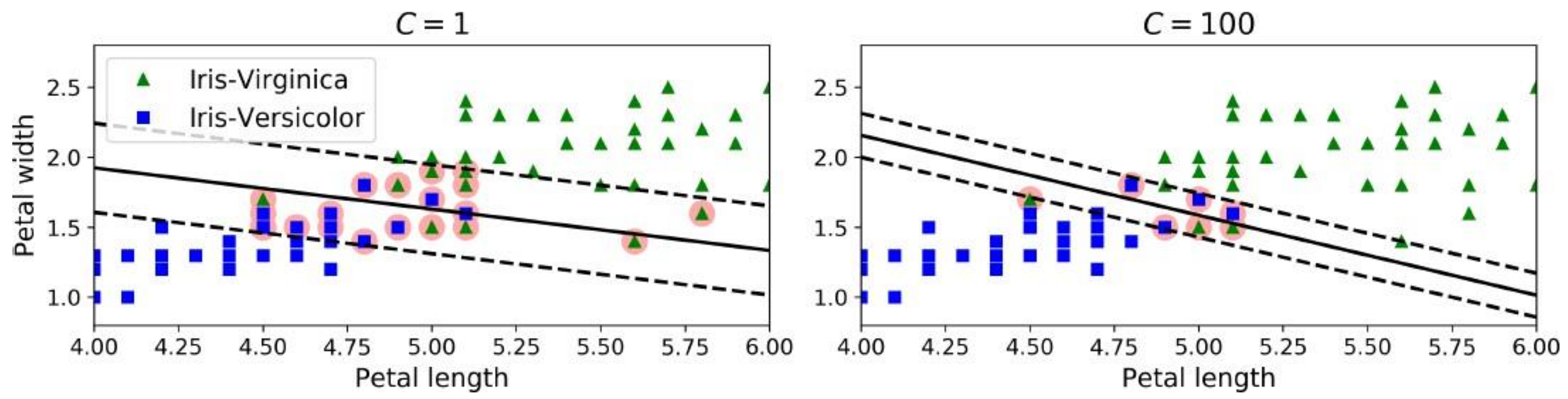


Figure B

- SVMs are **sensitive** to the feature scales.
- In Figure A, the vertical scale is much larger than the horizontal scale, so the widest possible street is close to the horizontal.
- After feature scaling, the decision boundary looks much better (Figure B).

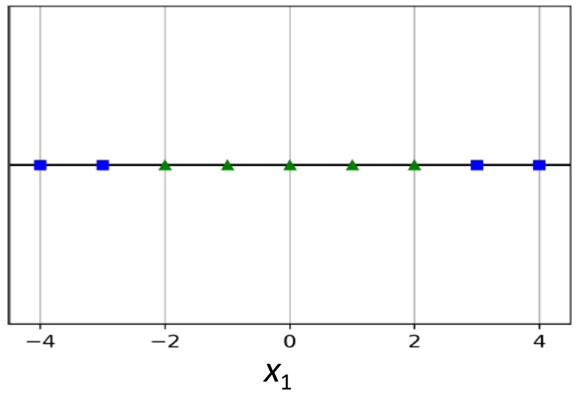
Soft Margin Classification

- The objective is to find a good balance between keeping the street as large as possible and limiting the **margin violations** (i.e., instances that end up in the middle of the street or even on the wrong side).
- This is called "soft margin classification".
- In Scikit-Learn's SVM class, you can control this balance using the C hyperparameter: a smaller C value leads to a wider street but more margin violations.

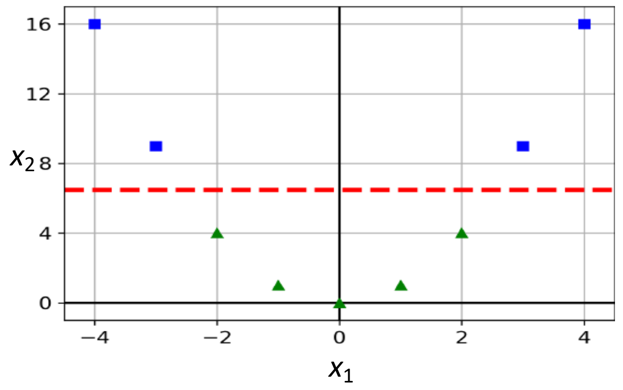


Nonlinear Datasets

- Many datasets are not even close to being linearly separable.
- Adding More Features – One common technique to handle nonlinear datasets is feature engineering, which involves creating new features based on the existing ones. This can be done in various ways, including adding **polynomial features**.
- This is because increasing the dimensionality can "unfold" the data, making it possible to draw a linear separation.



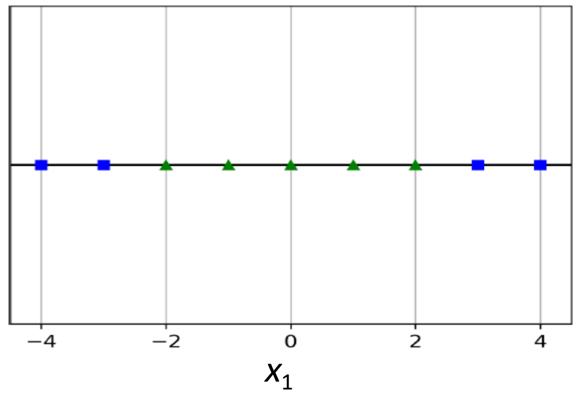
Simple dataset with just one feature x_1



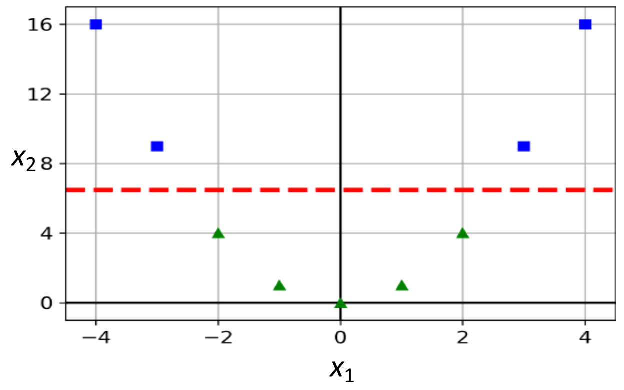
The resulting 2D dataset is perfectly linearly separable (add a second feature $x_2 = (x_1)^2$)

Kernel SVM

- Kernel SVM is an extension of the basic SVM algorithm that allows for more complex data transformations and decision boundaries.
- Kernel SVM uses a kernel function to compute the dot product of data points in a high-dimensional space without transforming the data into that space. Common kernel functions include:
    - **Linear Kernel** is essentially the dot product of two vectors in the input space.
    - **Polynomial Kernel** allows for polynomial decision boundaries in the original feature space.



Simple dataset with just one feature x_1



The resulting 2D dataset is perfectly linearly separable (add a second feature $x_2 = (x_1)^2$)

Attendance

You can use the provided link if you don't have a mobile phone or if your phone lacks a QR-Code reader – https://forms.gle/qVEm2UtSeMeFNTrW6

Multiclass Classification

Binary vs. Multiclass Classification

- **Binary Classification**: When we must categorize given data into **two** distinct classes.
- Example – Based on the given health conditions of a person, we must determine whether the person has a particular disease or not.

- **Multiclass Classification**: The number of classes is **more than two** classes.
- For Example – Based on data about different species of flowers, we must determine which species our observation belongs to.

Confusion Matrices for Multiclass Classification

- Let's say that we had a more significant number of test examples and that we obtained the following confusion matrix:

		predicted class		
		cold	allergy	strep throat
actual class:	cold	25	8	7
	allergy	6	15	3
	strep throat	5	4	33

- What is the accuracy of the model?
  - Total # of test cases = 106
  - # of accurately classified instances = 25 + 15 + 33 = 73
  - Accuracy = 73/106 = 0.6886 or 68.9%

Confusion Matrices for Multiclass Classification

		predicted class		
		cold	allergy	strep throat
actual class:	cold	25	8	7
	allergy	6	15	3
	strep throat	5	4	33

- How many test cases of strep throat are there?
  5 + 4 + 33 = 42
- How many actual colds were misdiagnosed?
  8 + 7 = 15
- What percentage of actual colds were correctly diagnosed?
  total # of colds = 25 + 8 + 7 = 40, Correctly diagnosed = 25, %Correctly diagnosed = 25/40 = 0.625 or 62.5%

Exercise: Evaluating Classifier Performance in Credit Card Fraud Detection

- You are provided with the performance of two different fraud detection models on a set of 400 credit card transactions. Both classifiers have achieved the same overall accuracy, but their performance in detecting fraudulent transactions varies.

		predicted by model A	
		fraud	not fraud
actual:	fraud	100	10
	not fraud	40	250

		predicted by model B	
		fraud	not fraud
actual:	fraud	80	30
	not fraud	20	270

  - Overall accuracy = (100+250)/400 = 0.875
  - Overall accuracy = (80+270)/400 = 0.875

- Your task is to analyze the confusion matrices of both models to understand their strengths and weaknesses.

Exercise (cont.)

		predicted by model A: fraud	predicted by model A: not fraud
actual:	fraud	100	10
	not fraud	40	250

		predicted by model B: fraud	predicted by model B: not fraud
actual:	fraud	80	30
	not fraud	20	270

- Overall accuracy = (100+250)/400 = 0.875
- Precision $= \frac{100}{100+40} = \frac{100}{140}$ = 0.7143
- Recall (or True Positive Rate) $= \frac{100}{100+10} = \frac{100}{110}$ = 0.9091
- False Positive Rate $= \frac{40}{40+250} = \frac{40}{290}$ = 0.1379

- Overall accuracy = (80+270)/400 = 0.875
- Precision $= \frac{80}{80+20} = \frac{80}{100}$ = 0.8
- Recall (or True Positive Rate) $= \frac{80}{80+30} = \frac{80}{110}$ = 0.7273
- False Positive Rate $= \frac{20}{20+270} = \frac{20}{290}$ = 0.0690

Exercise (cont.)

		predicted by model A: fraud	predicted by model A: not fraud
actual:	fraud	100	10
	not fraud	40	250

		predicted by model B: fraud	predicted by model B: not fraud
actual:	fraud	80	30
	not fraud	20	270

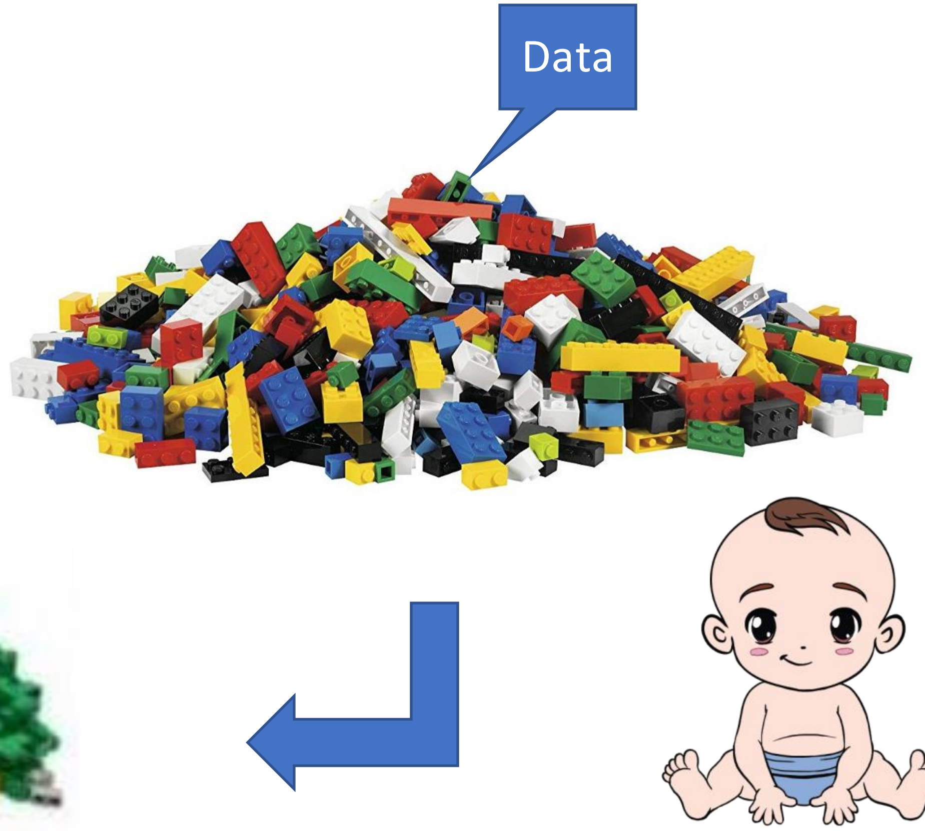
	Model A	Model B
Overall accuracy	0.875	0.875
Precision	0.7143	0.8
Recall(or True Positive Rate)	0.9091	0.7273
False Positive Rate	0.1379	0.0690

- Classifier A has a higher recall, indicating it is better at identifying actual **fraud cases** but with a higher false positive rate. This means it incorrectly flags more non-fraudulent transactions as fraudulent.
- Classifier B has higher precision and a lower false positive rate, suggesting it is more conservative with labeling transactions as fraudulent. Still, when it does, it is more likely to be correct.

Unsupervised Learning

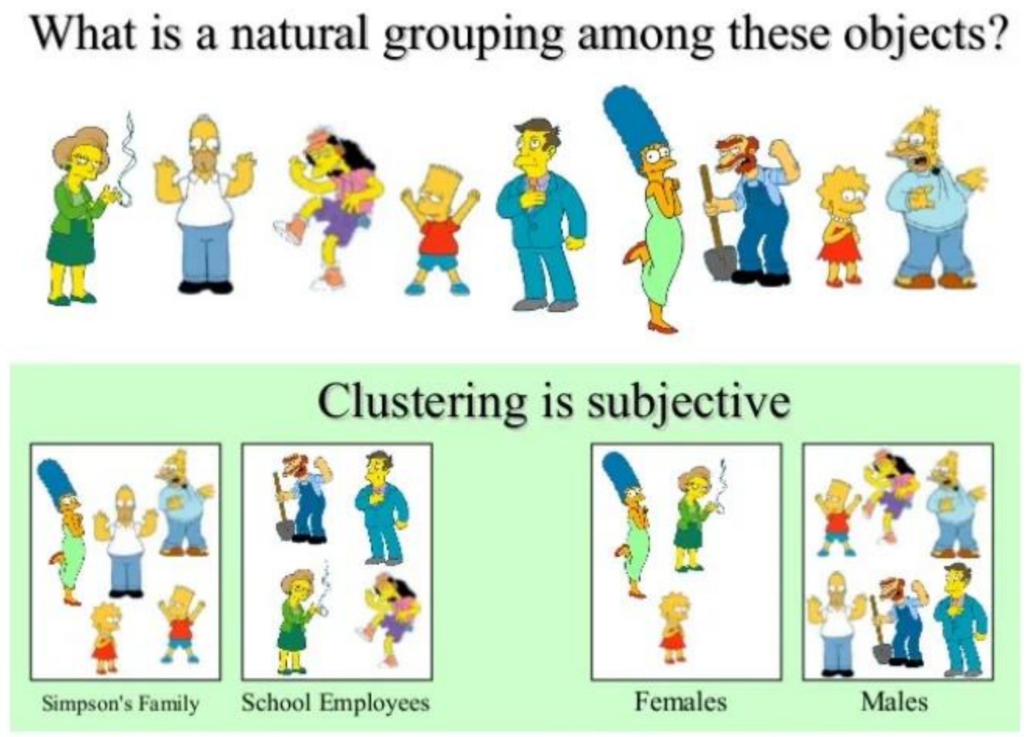
Recall: Unsupervised Learning

- Discover hidden groups (patterns) within the data
- NO pre-existing knowledge
- Example
  - Task: detect groups (patterns) of similar objects
  - Input: a lot of complex and unlabeled data

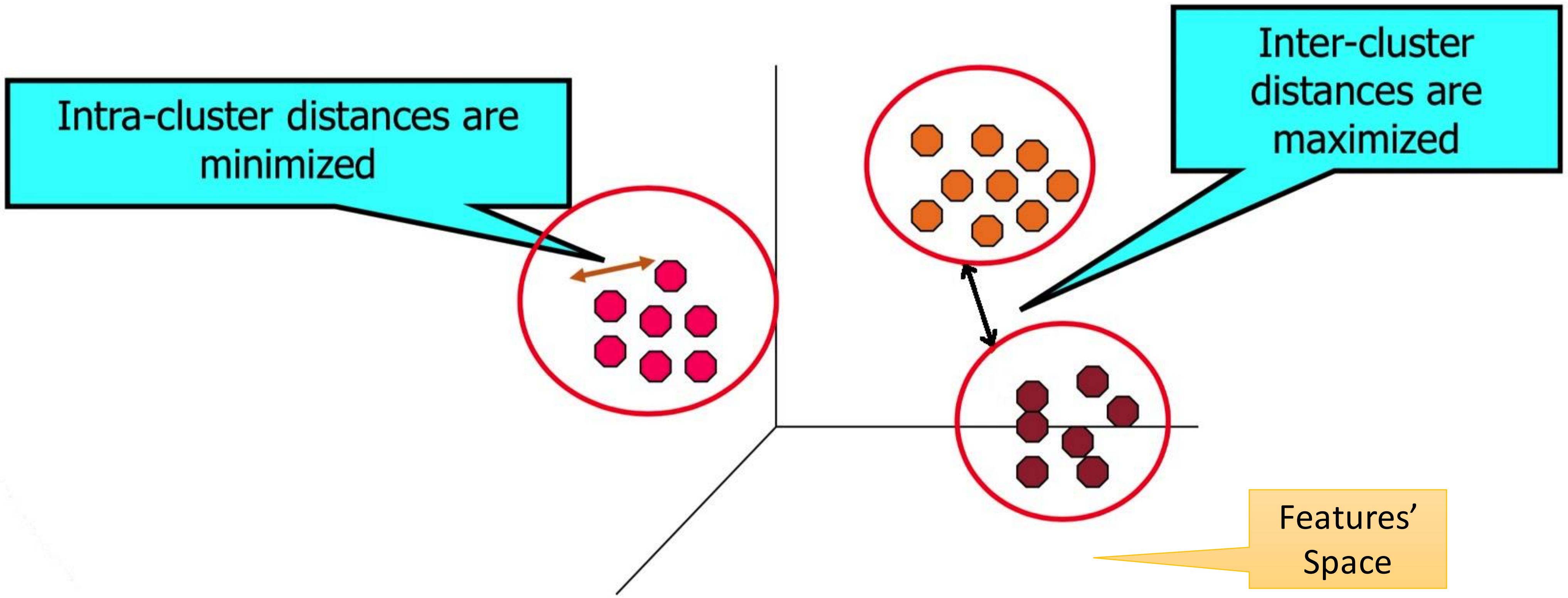


Clustering

- "Clustering is an unsupervised ML technique that finds **certain patterns/structures** in the unlabeled data to **segregate** them into different groups according to their properties (distribution placement)." Such groups are called **clusters**.
- Finding the groups of observations in the **population** such that the observations are
  - **Homogeneous** within the group (high intra-class similarity)
  - **Heterogeneous** between the groups (low inter-class similarity)



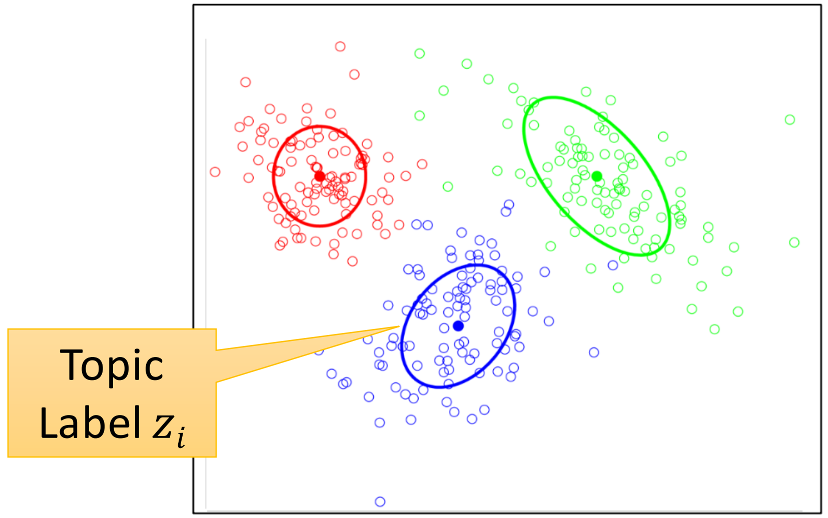
Intra & Inter-Clusters Distaces


Intra-cluster distances are minimized
Inter-cluster distances are maximized
Features' Space

- Clustering is not "right" or "wrong" – different clustering/clustering criteria can reveal different information about the data.

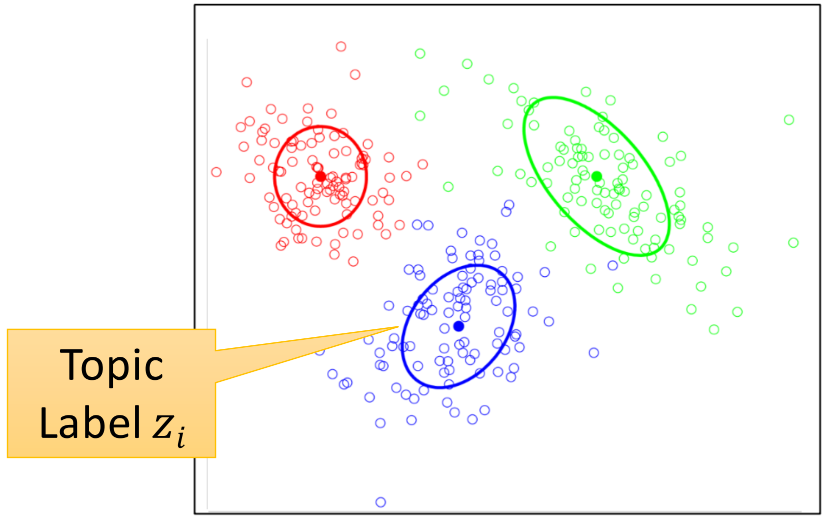
Clustering (cont.)

- Clustering: No need to know the labels
- In clustering, we must define what it means for two or more observations to be similar or different.
- What if some of the labels are known? These labels will work as **ground truth** (to be addressed later!)
- Example: Clustering a set of documents by topic
  - Discover groups (clusters) of related articles/websites/tweets
- How can we find a structured representation of the data
  - Input: documents as vectors x_i
  - Output: cluster labels z_i



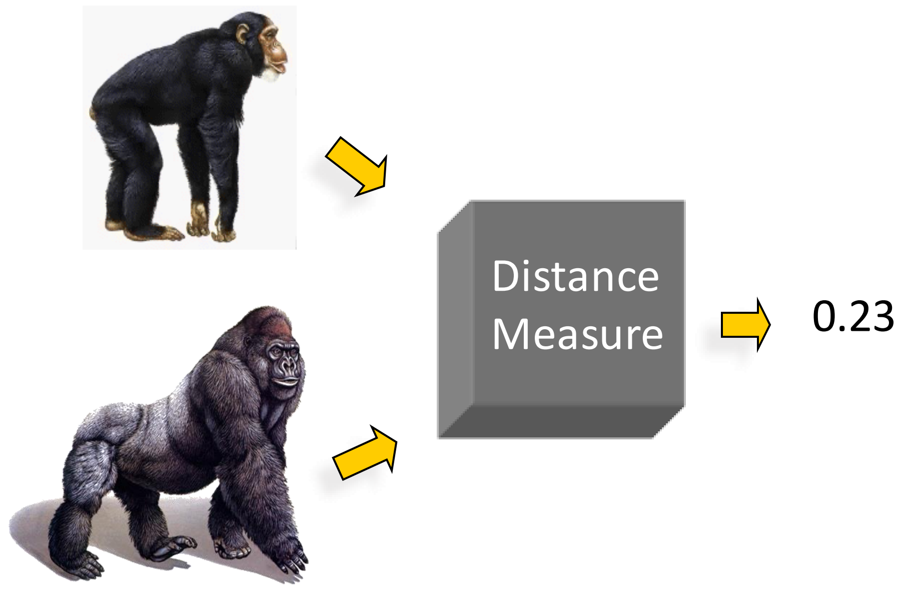
Clustering (cont.)

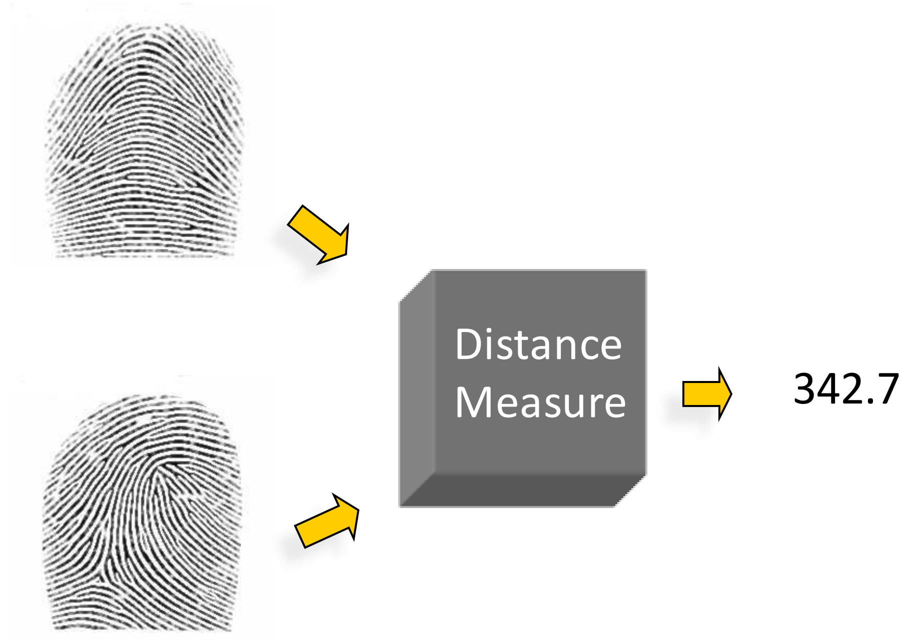
- A cluster could be defined by **center** & **shape/spread**
- Assign observation x_i (doc) to cluster k (topic label) if a **distance** to cluster center (ignoring shape) is closer than others.



Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$.





Source: Eamonn Keogh

Distance Measures (cont.)

- **Euclidean Distance** – Most common distance measure, used in methods like K-means and hierarchical clustering. Represents the straight-line distance between two points in Euclidean space.
- **Manhattan Distance** – Measures the distance between two points by summing the absolute differences of their coordinates.
- **Cosine Similarity** – It's a similarity measure that calculates the cosine of the angle between two vectors. Used when the magnitude of the vectors is not important, but the orientation is.
- **Jaccard Similarity / Distance** – Measures similarity between finite sets. Useful for comparing similarity between sets, like documents as sets of terms.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

$$d(p,q) = \sum_{i=1}^{n} |q_i - p_i|$$

$$d(p,q) = \frac{p \cdot q}{||p|| \cdot ||q||}$$

$1 - J(A,B)$ Where, $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$

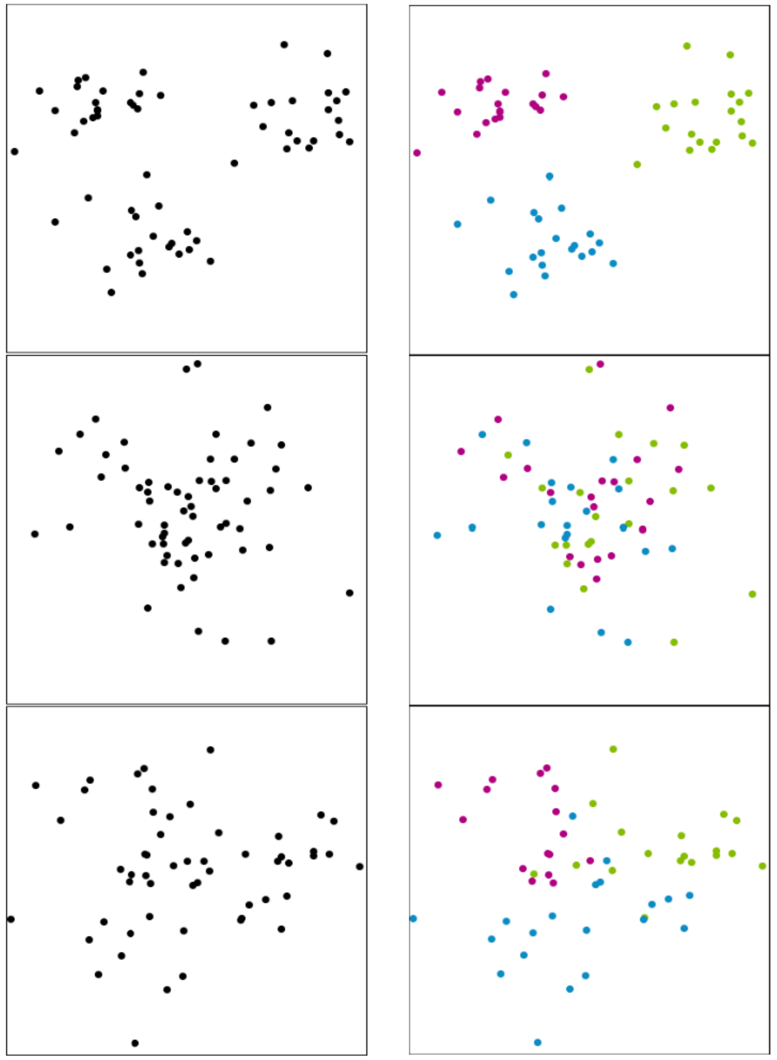
P.S. p and q represent two data points in a feature space. Each data point is typically represented as a vector of attributes (or features).

Hope for unsupervised Learning

Easy

Impossible

In Between



Partitional Clustering

- **Partitional clustering** or *non-hierarchical*: A division of objects into **non-overlapping** subsets (clusters) such that each object is in exactly one cluster.
- The *non-hierarchical* methods divide a dataset of N objects into 'K' clusters.
- **Algorithms**: K-means, Gaussian mixtures, hierarchical clustering, etc.

k-means Clustering

k-means Clustering

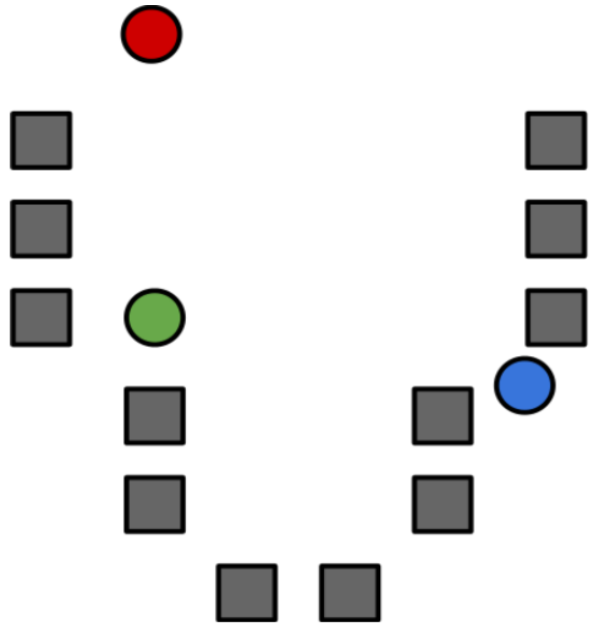
- K-means always maintains exact K clusters
    - Clusters represented as centroids (“center of mass”)
- Tends to **converge quickly**
- Sensitive to the choice of **initial** centroids
- Must choose K!

Basic algorithm:

```
Step 0: Choose $K$ cluster centroids
Step 1: Assign points to the closest centroid
Step 2: Recompute cluster centroids
Step 3: Goto 1
```

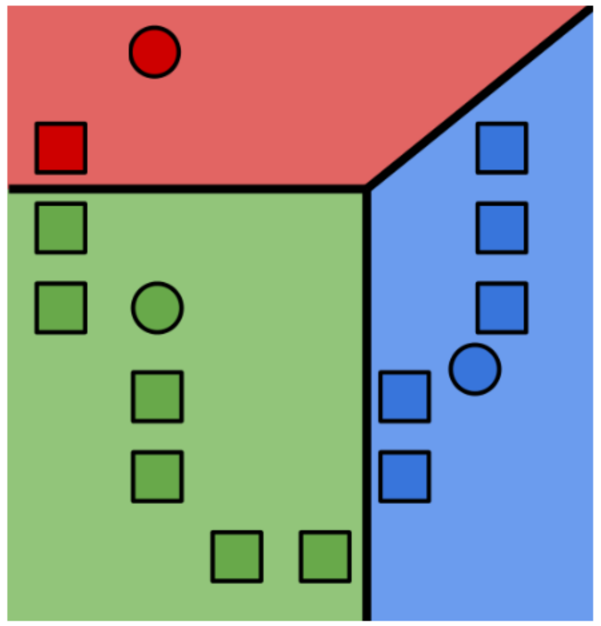
k-means Clustering Algorithm

- Step 0: Choose K cluster centroids (randomly): $\mu_1, \mu_2, \ldots, \mu_k$



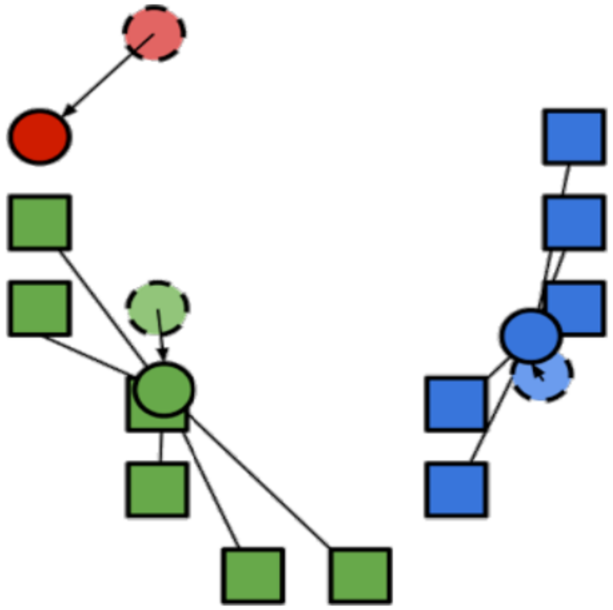
- Step 1: Assign observations to the closest cluster center: $z_i \leftarrow \arg\min_j \|\mu_j - \boldsymbol{x}_i\|^2$

$$\mu_j = \frac{1}{n_j} z \sum_{i:\, z_i = j} \boldsymbol{x}_i$$

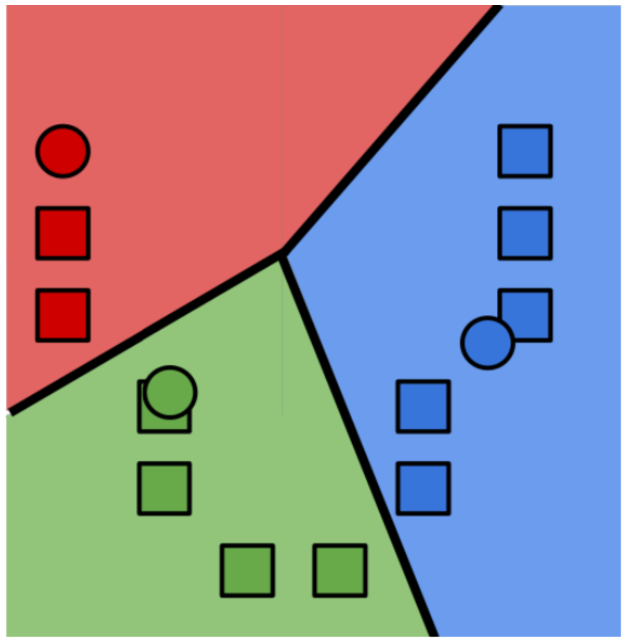


k-means Clustering Algorithm (cont.)

- Step 2: Revise cluster centers as the mean of assigned observations



- Step 3: Repeat steps 1 and 2 until convergence



k-means Clustering Algorithm (cont.)

- Expectation-Maximization (EM) – EM is a more general iterative optimization framework used for various ML models, including clustering.
- The **E-step** assigns the data points to the closest cluster.
- The **M-step** is computing the centroid of each cluster.

- The Objective Function:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \left\| x^i - \mu_k \right\|^2$$

$$w_{ik} = \begin{cases} 1 & \mathit{if\ data\ point\ } x_i \mathit{\ belongs\ to\ clusetr\ } k \\ 0 & \mathit{otherwise} \end{cases}$$

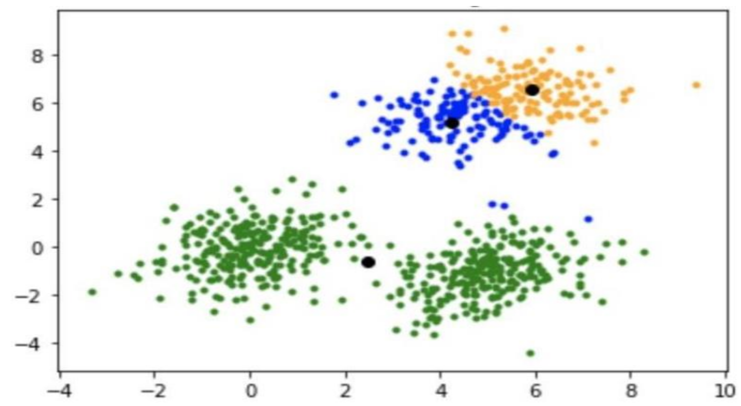
μ_k : *the centroid of the* x_i *cluster*

- Minimizing J ensures that data points are as close as possible to the centroids of their assigned clusters, which is how K-means clusters data effectively.

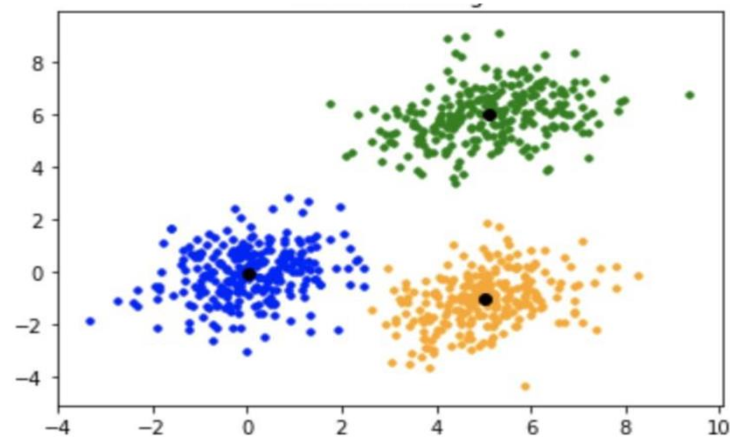
Smart Initialization with k-means++

- K-means++ tackles the sensitivity to the initial centroid problem by providing a more innovative way to initialize the centroids. It follows a **probabilistic** approach that ensures the initial centroids are:
  - **Spread out**: This helps capture diverse regions of the data and avoids early convergence to poor solutions.
  - **Further apart**: This reduces the chances of empty clusters and imbalanced sizes.

Poor Clustering



Ideal Clustering



How k-means++ works?

1. Choosing the first cluster center **uniformly** at random from data points
   - "uniformly" means that every data point has an equal probability of being selected as the first cluster center. This is a probabilistic way to ensure that there is no bias towards any point in the data space when initializing the cluster centers.
2. For each observation x, compute distance $d(x)$ to the nearest cluster center
3. Choose a new cluster center from amongst data points, with a probability of x being chosen proportional to $d(x)2$
4. Repeat Steps 2 and 3 until k centers have been chosen

- k-means++ Pos/Con – <u>Computationally costly</u> relative to random initialization, but often <u>converge more rapidly</u>

Assessing the Quality of the Clustering

- Ideal clustering is characterized by minimal **intra-cluster** distance and maximal **inter-cluster distance**.
- As k-means is trying to minimize the sum of squared distances:

$$\sum_{j=1}^{k} \sum_{i:z_i=j} \|\mu_j - \boldsymbol{x}_i\|^2$$

- Measure of quality of given clustering (Internal criterion)

Finding Optimal k

- Can refine clusters more and more to the data **overfitting**!
- The extreme case of k=N: can set each cluster center equal to a data point
- There are two major approaches to finding the optimal number of clusters:
  - Domain knowledge
  - Data-driven approach
    - **Empirical Method**: $k = N/2$; N: Total number of data points
    - **Silhouette Coefficient**
    - **Elbow method –** measure compactness

Silhouette$_{\text{(si·luh·wet)}}$ Coefficient

- The Silhouette Coefficient is a measure of how similar an object (a point) is to its own cluster (**cohesion**) compared to other clusters (**separation**).

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

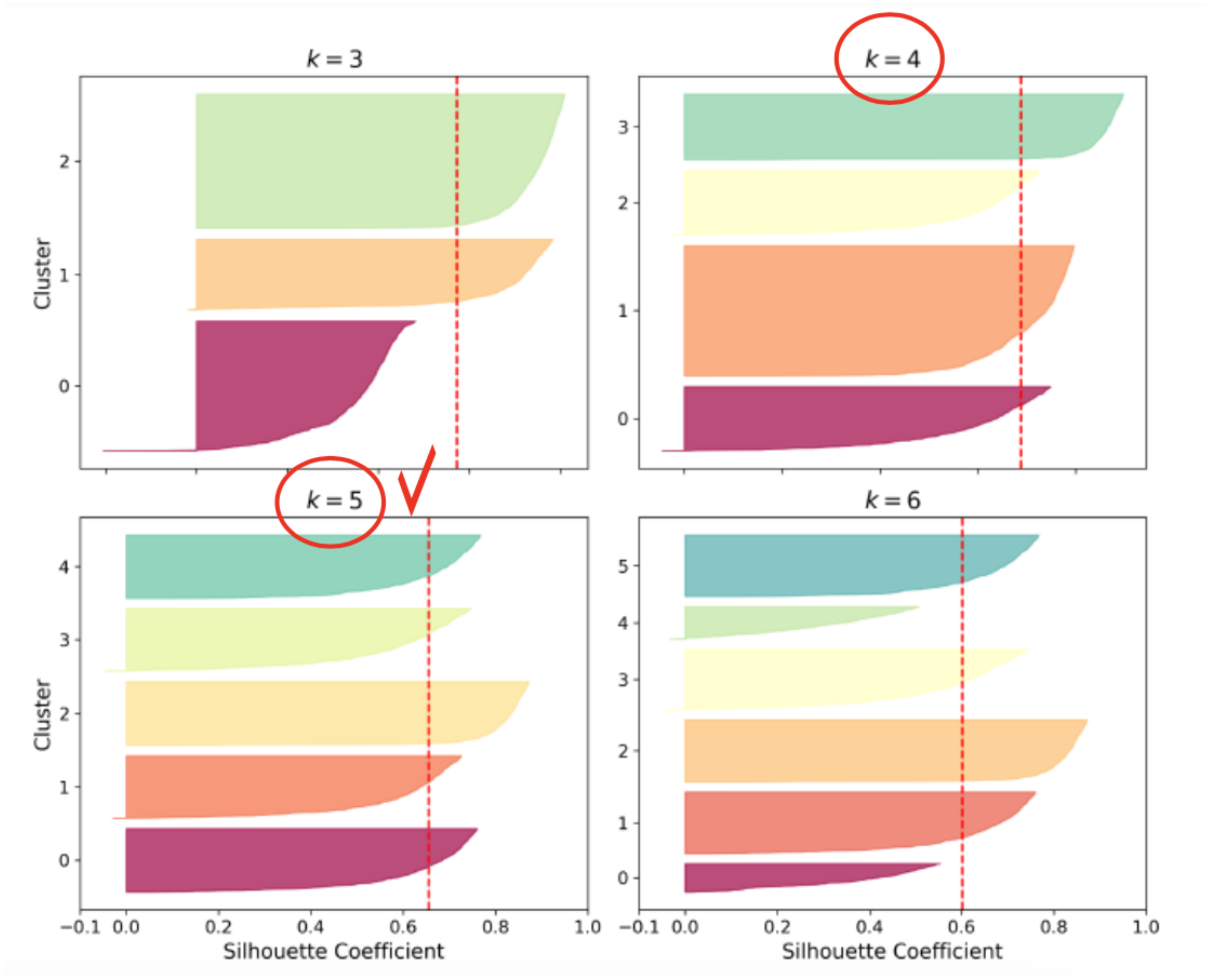
$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j),$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

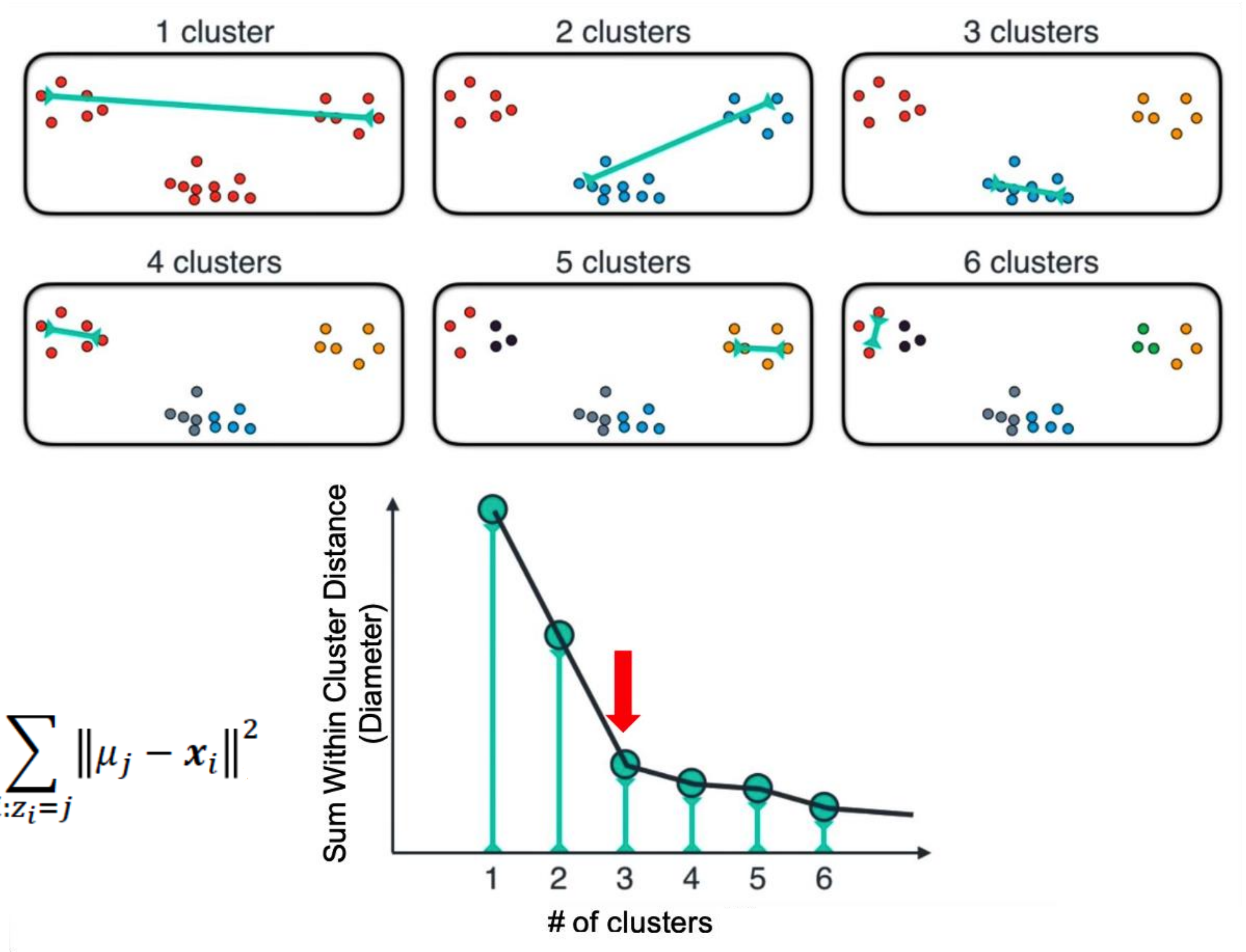
- $a(i)$ is the average distance of point i from all other points in its cluster
- $b(i)$ is the smallest average distance of i to all points in any other cluster.
  - $b(i)$ is found by measuring the average distance of i from every point in cluster A, the average distance of i from every point in cluster B, and taking the smallest resulting value.
- 1 means much closer to own cluster, 0 means, equally close to own cluster as to another cluster. Negative means closer to another cluster.

Silhouette Coefficient - Example

A Silhouette Coefficient versus the number of clusters diagram is used to determine a dataset's optimal number of clusters when using clustering algorithms such as K-means.



Find k: Elbow Method



$$\text{Sum of Within-Cluster Distances} = \sum_{j=1}^{k} \sum_{i:z_i=j} \left\|\mu_j - \boldsymbol{x}_i\right\|^2$$

Source:
https://www.youtube.com/watch?v=q71Niz856KE

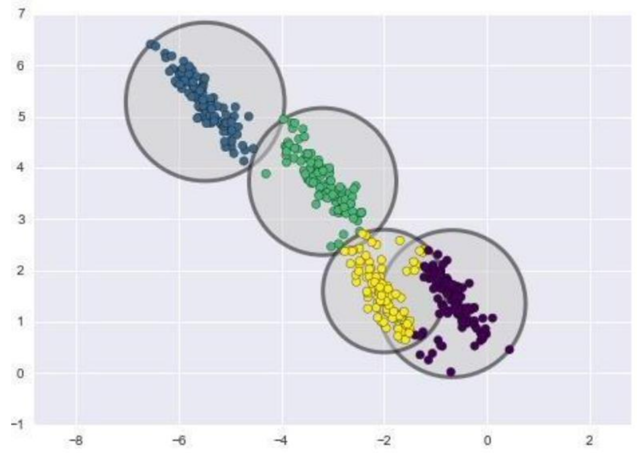
Data Preparation for Cluster Analysis

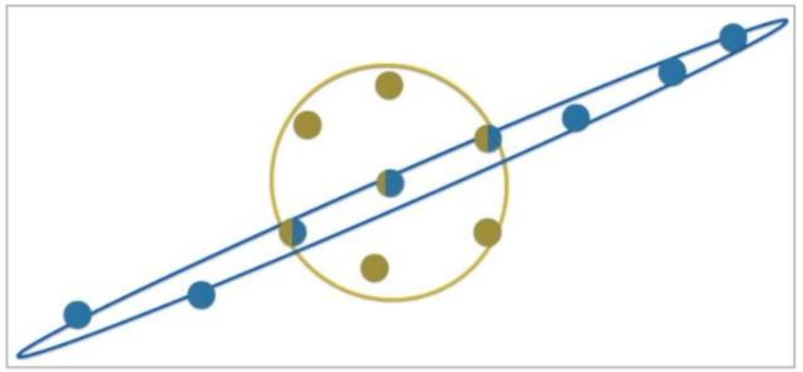
- Why do we need to prepare data for clustering?
  - Variables have incomparable units (price in $, area in square-meter)
  - Variables with the same unit may be significantly different in terms of their scales and variances
  - Data in original raw form may lead to bias in clustering
  - Clusters may be heavily dependent on one feature
- Solution: Normalization

Mixture Models: Model-Based Clustering

Gaussian Mixture Models (GMM) & EM

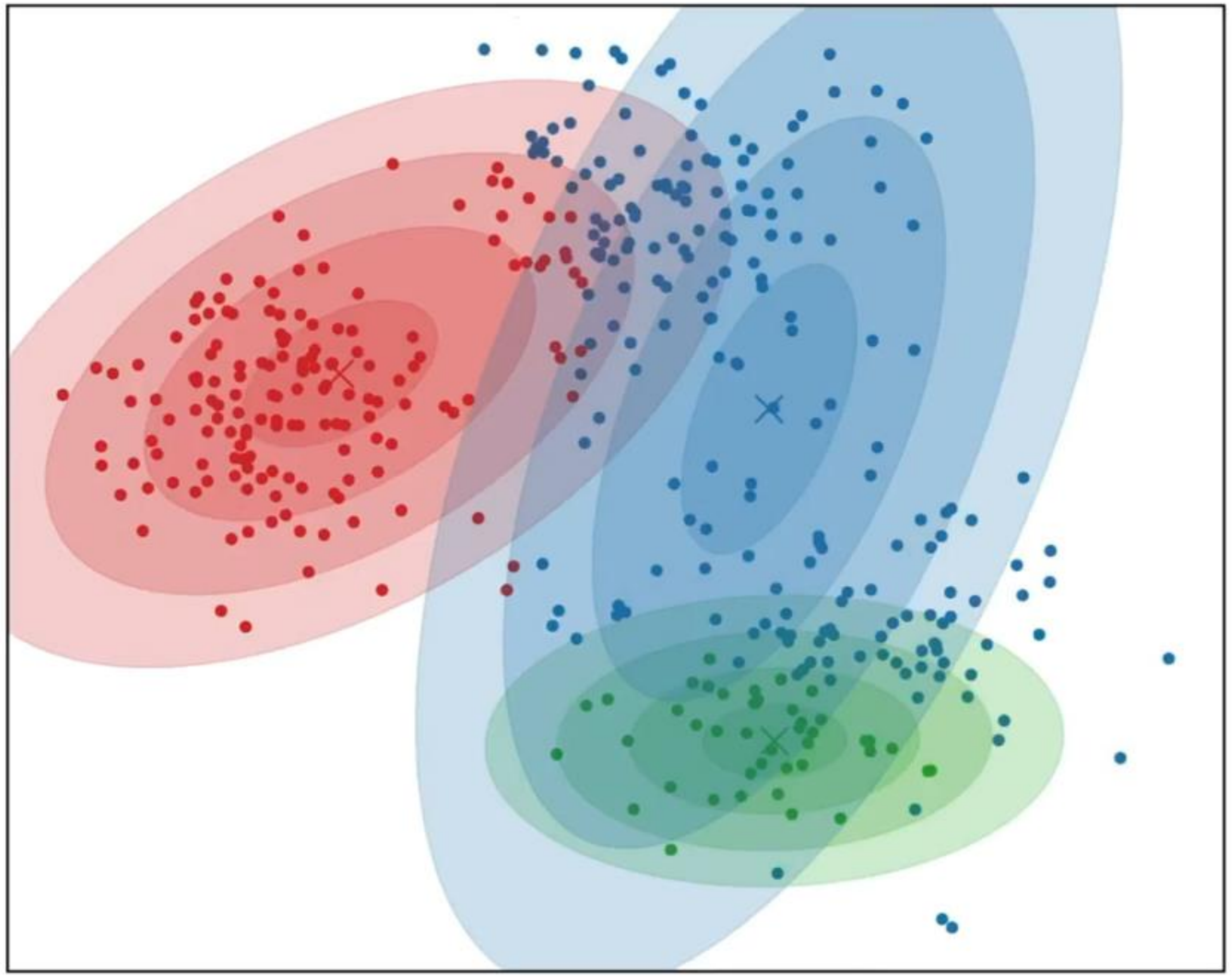
- GMM is **a probabilistic model** that assumes instances were generated from a **mixture** of several *Gaussian distributions* with unknown parameters.
- GMM can represent clusters with different sizes and orientations, unlike k-means.
- **Mixture-models –** Soft Classification
  - Help to catch uncertainty in clustering
  - Accounts for cluster shapes, not just centers





Source: https://towardsdatascience.com/

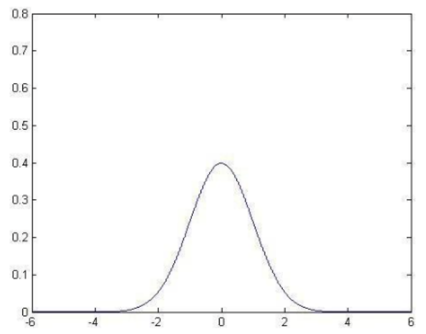
GMM & EM



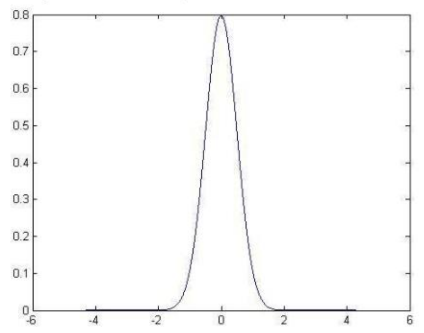
Gaussian Distribution Example

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

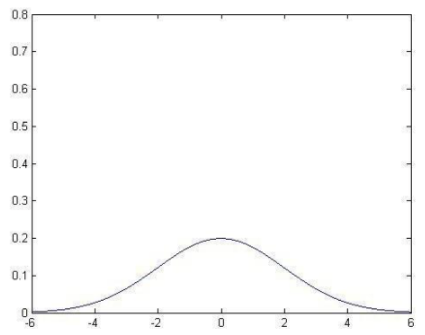
$\mu = 0, \sigma = 1$



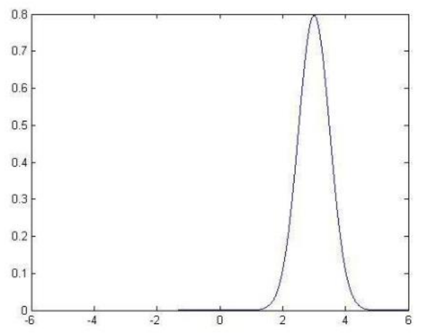
$\mu = 0, \sigma = 0.5$



$\mu = 0, \sigma = 2$



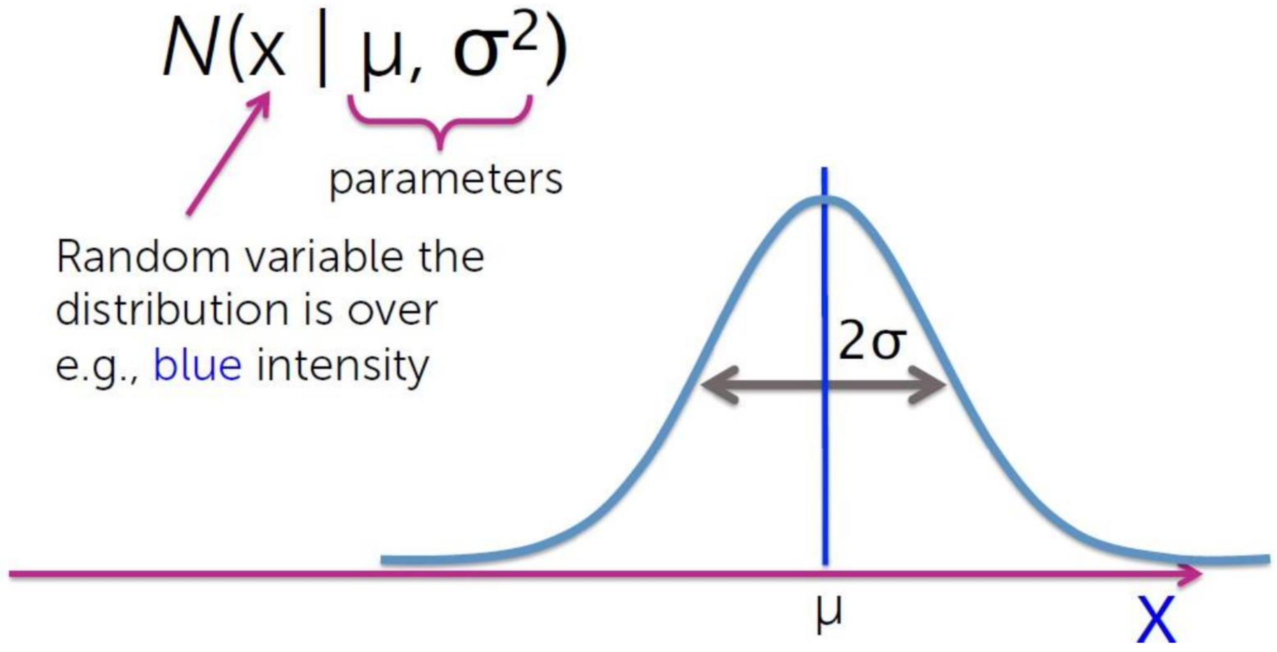
$\mu = 3, \sigma = 0.5$



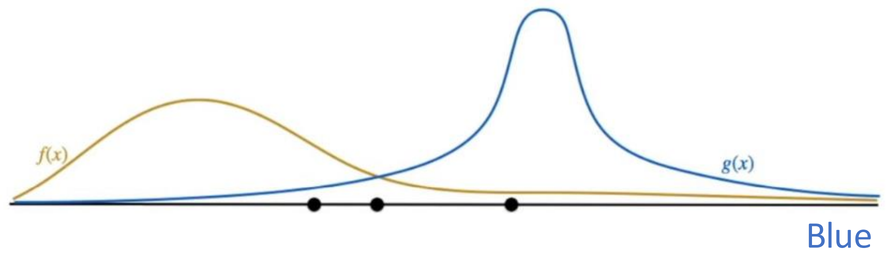
$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)^2$$

Notating a 1D Gaussian Distribution

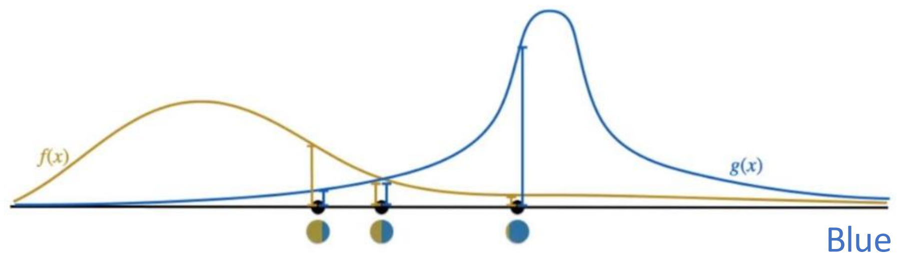

$N(x \mid \underbrace{\mu, \sigma^2}_{\text{parameters}})$
parameters
Random variable the distribution is over e.g., blue intensity
2σ
μ
X

GMM Steps



1. Initialize Clusters (Gaussian distributions)

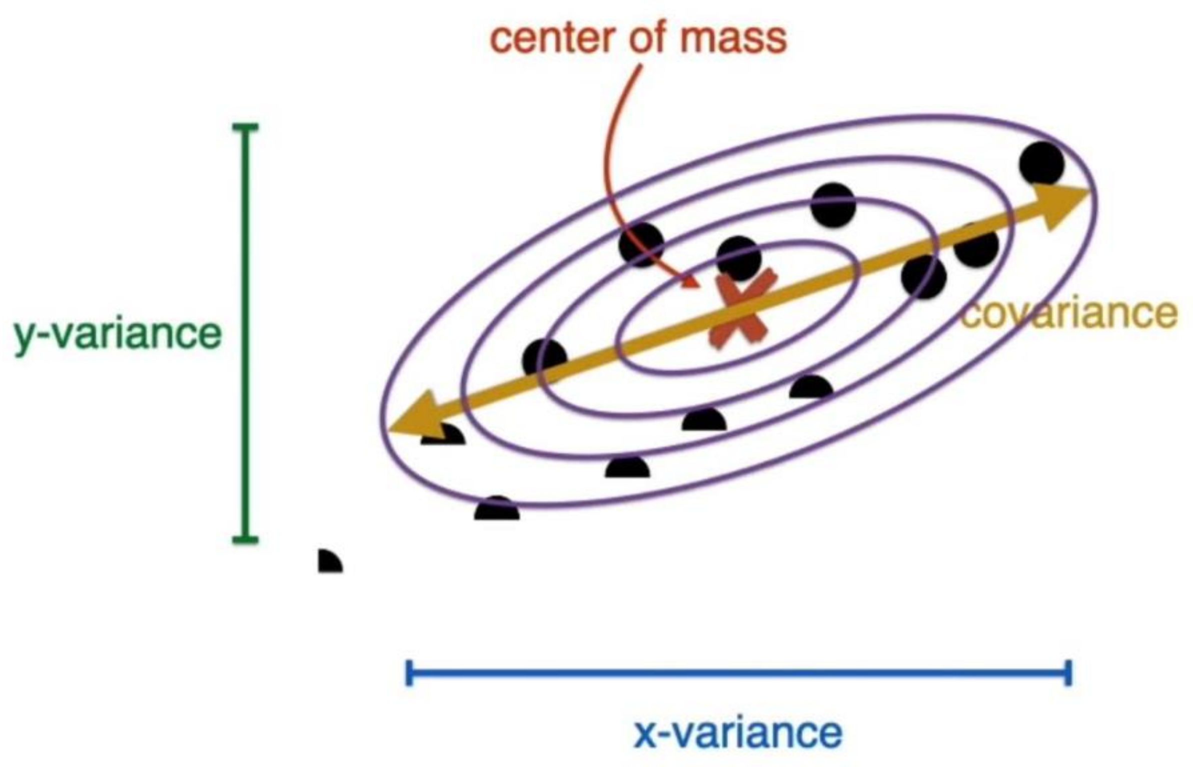
- **Choose the Number of Components**: Decide on the number of Gaussian distributions (clusters) to fit the data. This can be based on **domain knowledge** or **heuristic methods**.
- **Initial Parameters**: Initialize the parameters of the Gaussian components. This can involve setting initial means and mixture coefficients either randomly, by using the results of another clustering method (e.g., K-means), or based on prior knowledge.



2. Coloring Data Points (Hard or soft Assignment)

- **Cluster Membership**: fitting a GMM to the data, each data point is associated with each cluster to a certain degree, represented by a probability.
- This probability indicates how likely it is for a data point to belong to each cluster.

GMM Steps (cont.)

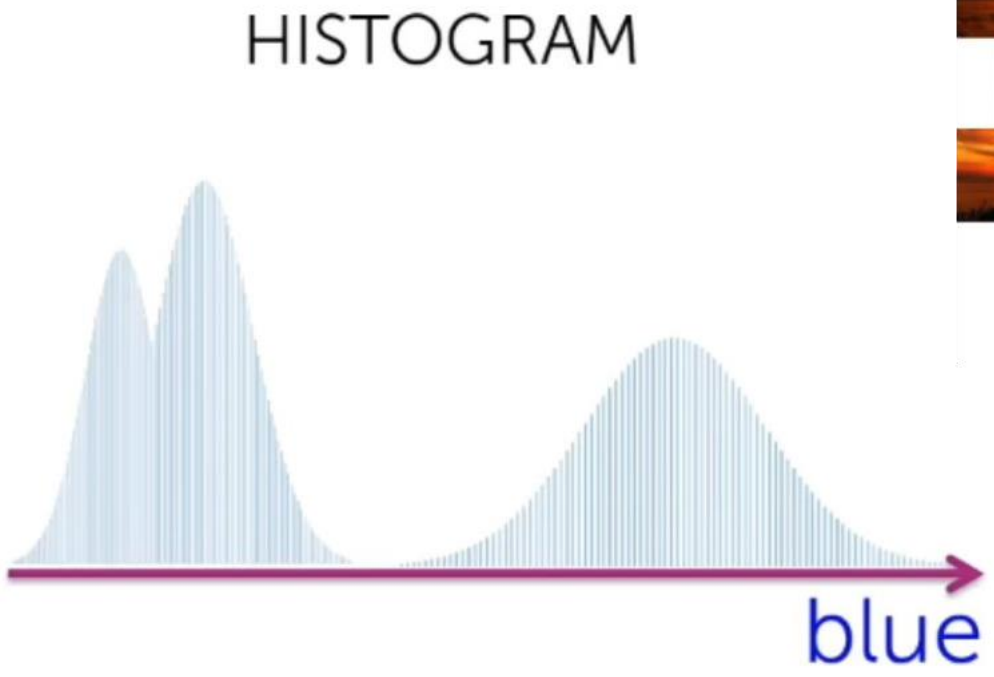


3. Fitting a Gaussian

- The **EM algorithm** is used to iteratively estimate the parameters that maximize the likelihood of the data given the model.
- After each iteration of the E-step and M-step, check if the convergence criterion is met.
- This could be a small change in the log-likelihood of the data given the model, or a predefined number of iterations.
- If not converged, return to the E-step.

Motivating Application: Clustering images

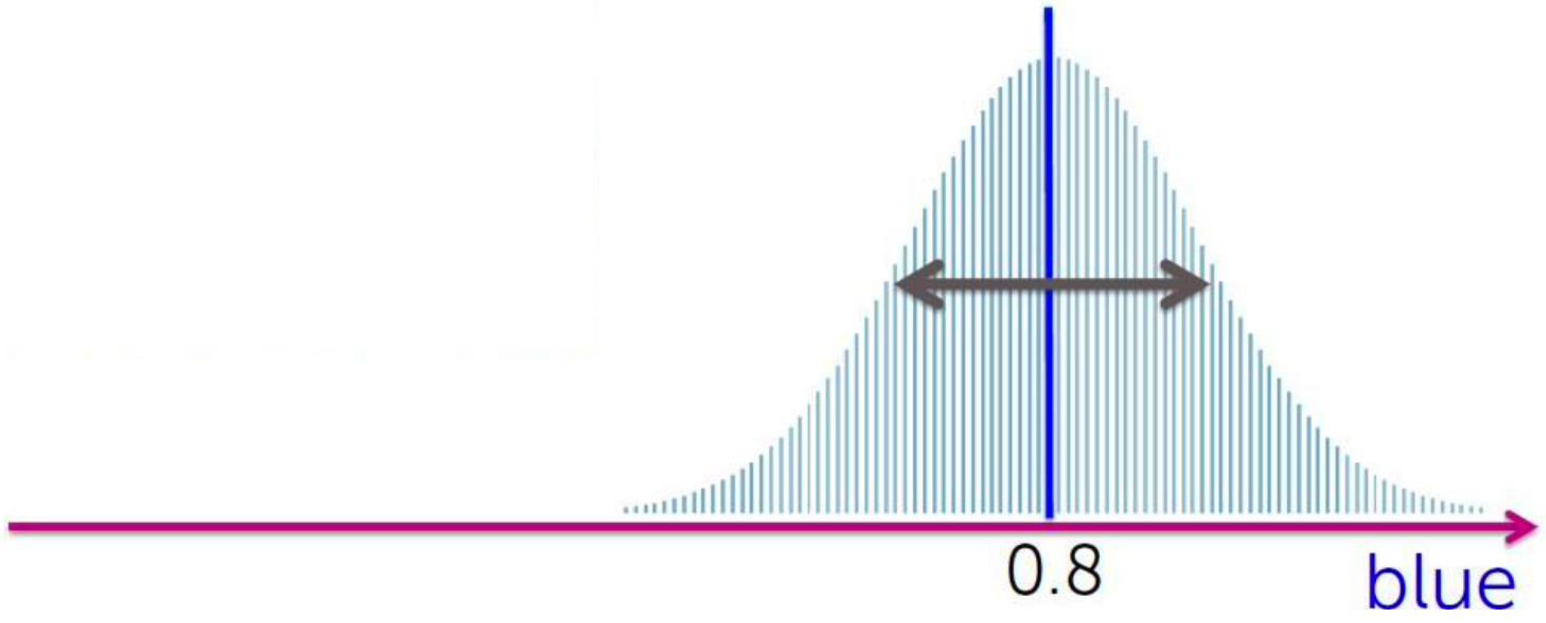
- Discover groups of unlabeled images



- Every image has a variation of RGB intensities.

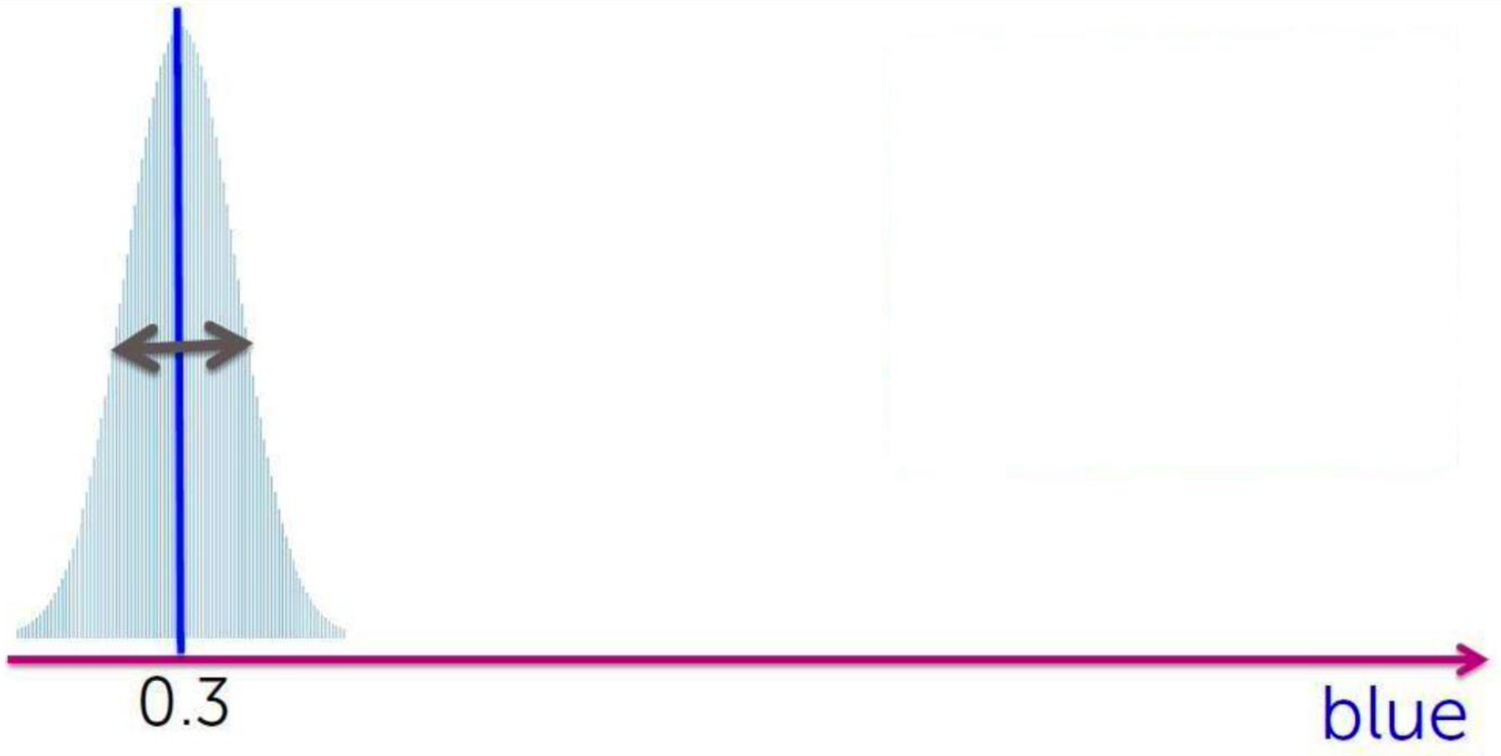
Distribution Over All Cloud Images

- Let’s look at just the Blue dimension



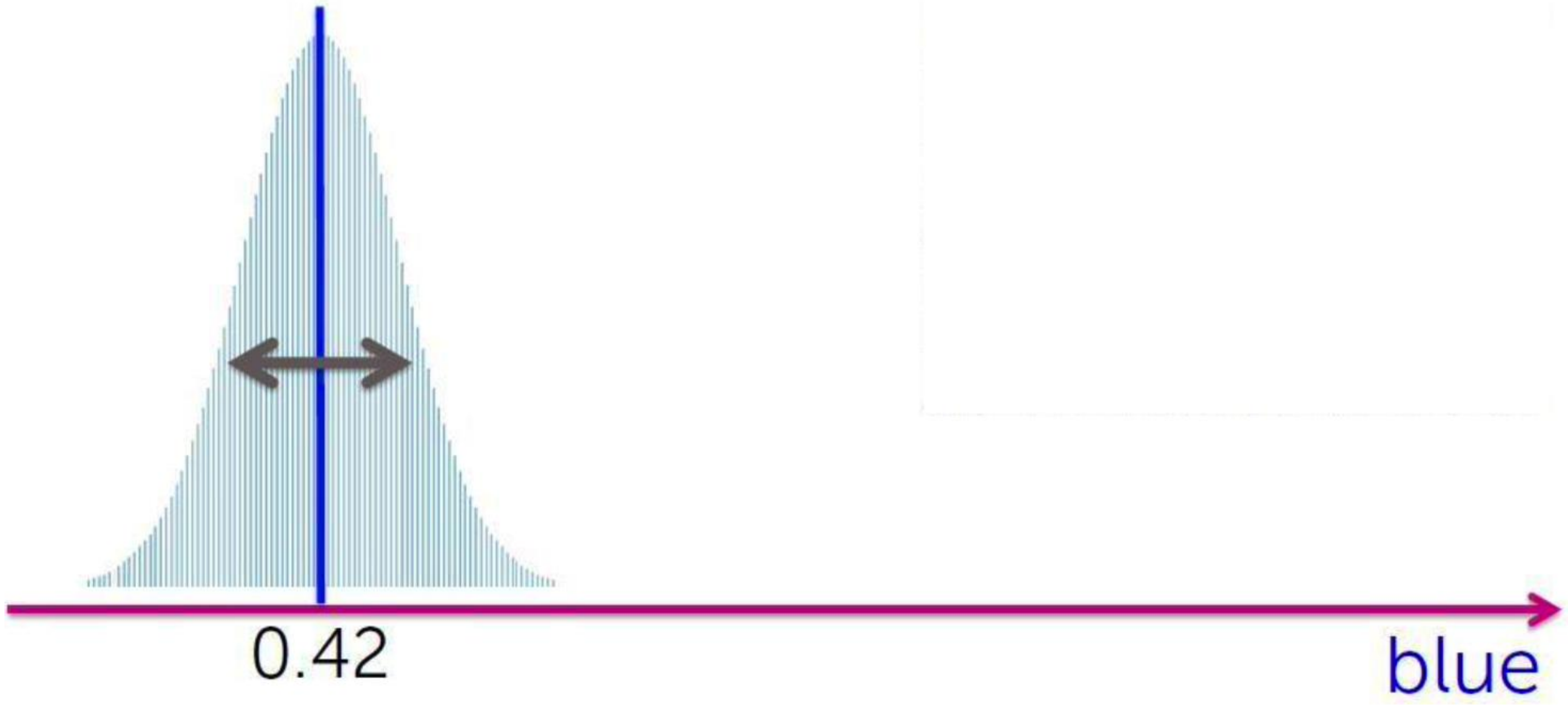
Distribution Over All Sunset Images

- We still focus on the at Blue dimension



Distribution Over All Forest Images

- We still focus on the at Blue dimension

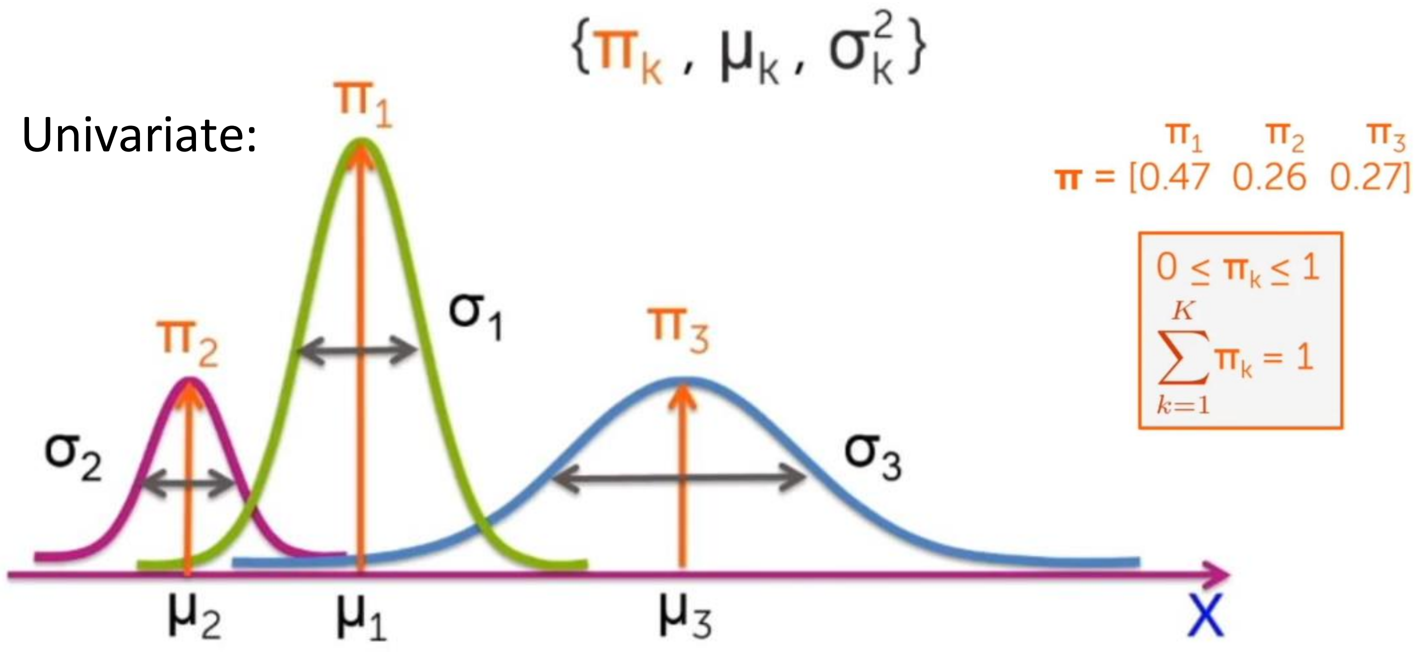


GMM Algorithm: Combination of Weighted Gaussians

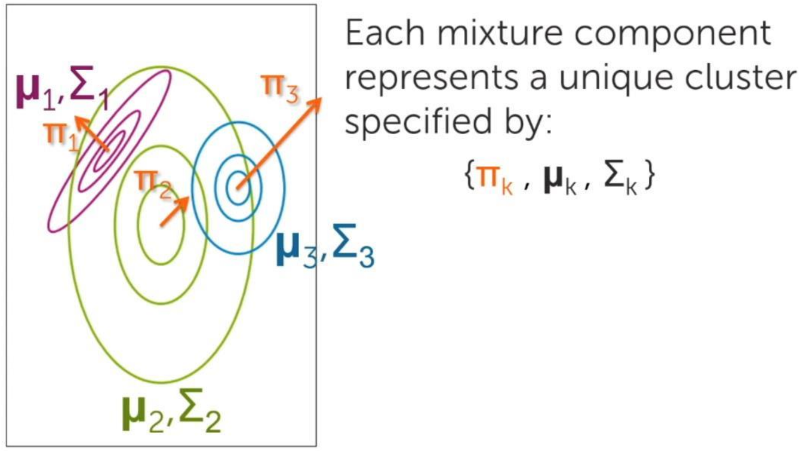
Given observation x_i from cluster k, what is the likelihood of seeing x_i (likelihood of observing a data point given that it came from Gaussian k):

$$N(x_i|\mu_k, \Sigma_k)$$

- Univariate:



- Multivariate:


μ_1, Σ_1
π_1
π_2
π_3
μ_3, Σ_3
μ_2, Σ_2
Each mixture component represents a unique cluster specified by:
$\{\pi_k, \mu_k, \Sigma_k\}$

Source: https://www.youtube.com/watch?v=DODphRRL79c

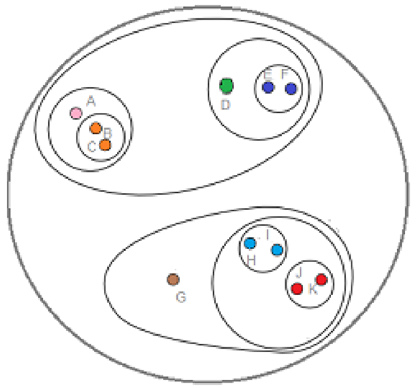
Expectation Maximization (EM) Algorithm

- We need the parameters of each Gaussian (i.e., variance, mean, and weight) to cluster our data, but we need to know which sample belongs to what Gaussian to estimate those very same parameters.

1. Start with random Gaussian parameters (θ)
2. Repeat the following until we converge:
   - **Expectation Step**: Compute p(zi = k | xi, θ). In other words, does sample i look like it came from cluster k?
   - **Maximization Step**: Update the Gaussian parameters (θ) to fit points assigned to them (maximize the likelihood that each sample came from the distribution).

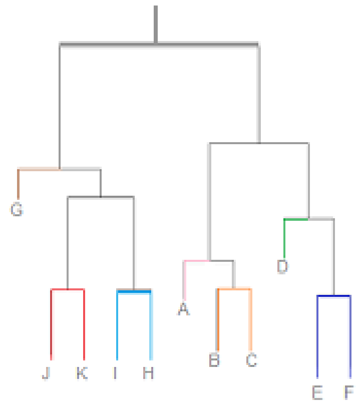
Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering

Hierarchical clustering is characterized by developing a hierarchy or tree-like structure (dendrogram).



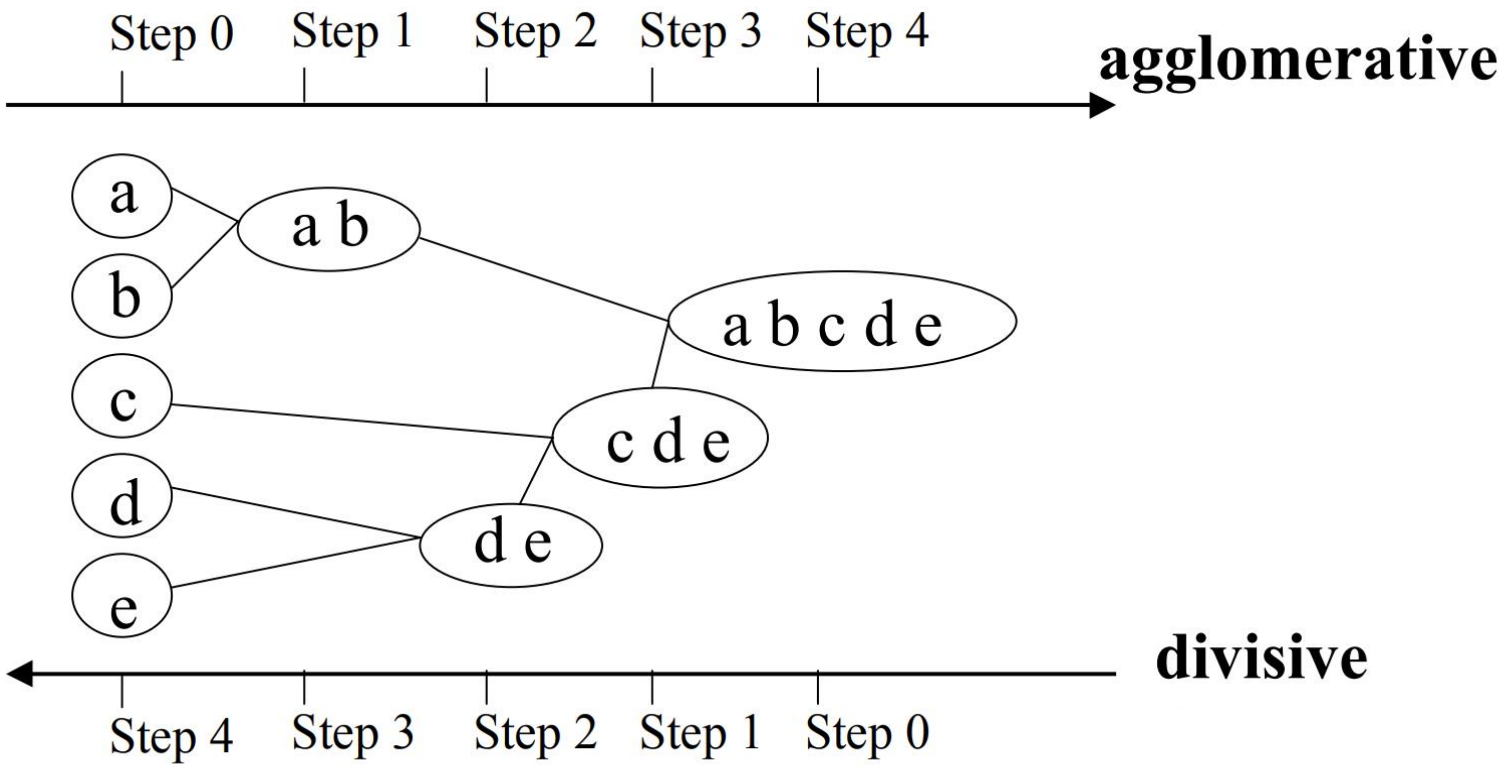
Bottom-up Clustering (Agglomerative)

- Agglomerative clustering starts with each object in a separate cluster.
- Clusters are formed by grouping objects into bigger and bigger clusters.

Top-down Clustering (Divisive)

- Divisive clustering starts with all the objects grouped in a single cluster.
- Clusters are divided or split until each object is in a separate cluster.

Hierarchical Clustering



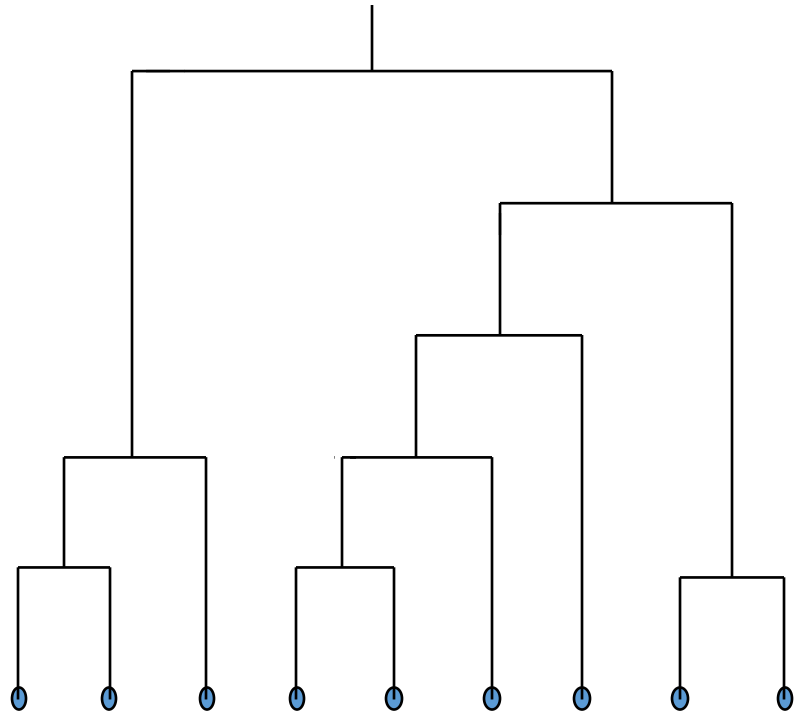
Credit for Marco BOTTA

Agglomerative Clustering

- Agglomerative (bottom-up): merge clusters iteratively.
  - start by placing each object in its own cluster.
  - merge these atomic clusters into larger and larger clusters.
  - until all objects are in a single cluster.
- The majority of hierarchical methods fall into this category, with their primary distinction being how they define the **similarity between clusters**. The most common Agglomerative method is the **centroid method**.
- In the **centroid method**, the distance between two clusters is the distance between their **centroids** (means for all the variables).

Hierarchical Clustering

- Clustering is obtained by cutting the dendrogram at the desired level: each connected component forms a cluster.
- Use **distance matrix** as clustering criteria. This method does not require the number of clusters k as input but needs a termination condition.


Dendrogram

What Is Good Clustering?

- **Internal criterion** – A good clustering method will produce high-quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
- **External criterion** – Compare a clustering against prior or expert-specified knowledge (i.e., **ground truth**) using a certain clustering quality measure (will be discussed later).
- **Relative criterion** – Directly compare different clustering solutions, usually those obtained via different parameter settings for the same algorithm.

Next

- Decision Tree
- Random Forest

Thank you