Machine Learning

1. Supervised and unsupervised learning

Yannick Le Cacheux

CentraleSupélec - Université Paris Saclay

September 2024

Table of Contents

Outline

Why machine learning?

- **Question:** How do you write precise instructions (*e.g.* a computer program) to distinguish a dog from a cat?

(a) A dog.

(b) A cat.

Figure: A dog and a cat. Even a two-years old can tell which is which.

Recognizing cats and dogs

- Dogs have longer snouts.

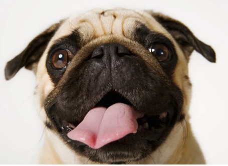


Figure: Short snoot.

- Cats have pointy ears.

Figure: Pointy boy.

- What is an ear anyway? Something that has roughly the shape of a triangle I guess.

Figure: Nice ear.

Why machine learning?

- **Question:** How do you write precise instructions (*e.g.* a computer program) to distinguish a dog from a cat?

Mostly: you can't

Others have tried. They have failed.

What is machine learning?

- **Arthur Samuel (1959)**: *"Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed."*
- **Tom Mitchell (1998)**: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

What is it used for?

- Machine learning is a must for:
  - Image recognition
  - Object detection
  - Speech recognition
  - ...
- And now also for:
  - Machine translation
  - Board games A.I. (go, chess...)
  - Robotics
  - ...
- And of course:
  - Conversational AI (ChatGPT, ...)
  - Image and video generation
  - ...

Outline

Supervised versus unsupervised learning

Supervised learning

Annotations, usually called *labels* or *targets*, are provided with the data.

Unsupervised learning

No labels are provided with the data.

Supervised versus unsupervised learning

Supervised learning

Annotations, usually called *labels* or *targets*, are provided with the data.

Unsupervised learning

No labels are provided with the data.

Other learning paradigms

There also exist other paradigms such as reinforcement learning, semi-supervised learning, self-supervised learning...
But in this course, we will mostly focus on supervised and unsupervised learning

Supervised versus unsupervised learning

For a same dataset, we can be either in a supervised or unsupervised learning setting depending on whether we use explicit labels or not.

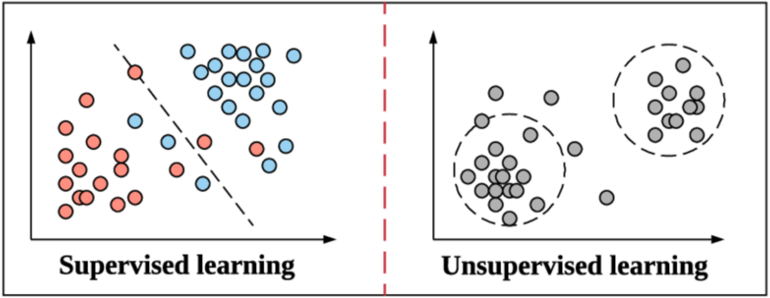


Figure: Left: Supervised learning with labeled points, from red and blue classes. The task is classification: tell wheither a new point is red or blue. Right: Unsupervised learning with unlabeled points. The task is to group points into clusters.

Examples of **supervised** learning tasks

- Classification: predict a category
  - Image classification
  - Fraud detection in financial transactions
- Regression: predict a quantitative value
  - Stock price prediction
  - Sales forecast
- Other applications:
  - Translation from one language to another
  - Image captioning
  - Speech recognition

Targets

Can you identify what the labels are for each of these applications?

Examples of **unsupervised** learning tasks

- Clustering: identify groups of similar points in data
  - Customer segmentation in marketing
  - Document clustering in text analysis
  - Gene expression clustering in bioinformatics
- Dimensionality reduction: reduce the number of dimensions of the input data while keeping as much information as possible
  - Visualization of high dimensional data
  - Image compression
  - Representation learning: e.g. learn meaningful representations of users and movies for recommendation systems
- ...

Outline

A first supervised learning algorithm

Let's introduce a first supervised learning method: **K-Nearest Neighbors** or **KNN**.

- KNN is a versatile and intuitive algorithm
- Used for both classification and regression tasks
- Instance-based learning: no explicit training phase
- Relies on the principle: "Similar inputs have similar outputs"

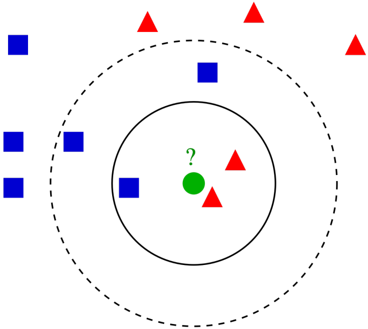


Figure: KNN illustration

How KNN Works

To classify a new, unlabeled data point (green dot in the center):

- Calculate distance to all training points
- Find the K nearest neighbors:
  - *e.g.* the 3 shapes inside the solid circle for $K = 3$
  - or the 5 shapes inside the dotted circle for $K = 5$
- Output the majority among the K neighbors
  - red triangle for $K = 3$ (2 out of 3)
  - blue square for $K = 5$ (3 out of 5)

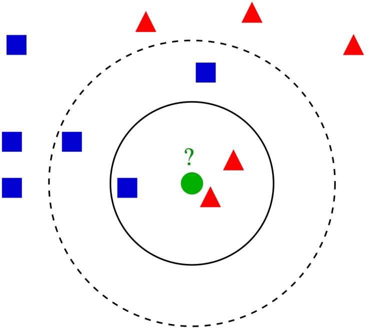


Figure: KNN illustration

Formalization

Assume we have a set of N points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding labels $\{y_1, \ldots, y_N\}$. The points $\mathbf{x}_n$ have dimension D, such that

$$\forall n \qquad \mathbf{x}_n = \begin{pmatrix} x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,D} \end{pmatrix} = (x_{n,1}, \ldots, x_{n,D})^\top \in \mathbb{R}^D$$

A note on notations

We will use the notations $\mathbf{x}_n$, y_n, D etc a lot throughout this course.

Assume we are also provided with a distance d, for example the Euclidean distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

Given a new point $\mathbf{x}$, we want to predict its label y.

Formalization (continued)

To estimate the label $\hat{y}$ of $\mathbf{x}$, we proceed as follows:

1. Calculate distances to all training points:

$$d_n = d(\mathbf{x}, \mathbf{x}_n) \quad \forall n \in \{1, \dots, N\}$$

2. Find indices of K nearest neighbors:

$$\mathcal{N}_K(\mathbf{x}) = \text{argsort}_K(d_1, \dots, d_N)$$

   where argsort_K returns the indices of the K smallest values.

3. Predict the most common class among neighbors:

$$\hat{y} = \text{mode}\{y_i : i \in \mathcal{N}_K(\mathbf{x})\}$$

Variants of KNN

- We can also use KNN for regression: predict the average of neighbor values:

$$\hat{y} = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x})} y_i$$

- In some variants, the neighbors can be weighted:

$$\hat{y} = \frac{\sum_{i \in \mathcal{N}_K(\mathbf{x})} w_i y_i}{\sum_{i \in \mathcal{N}_K(\mathbf{x})} w_i}$$

  with a weigh function $w(\cdot)$ usually decreasing with the distance d (e.g., $w_n = \frac{1}{d_n}$ or $w_n = e^{-d_n}$)

- Other variant: Fixed Radius NN. Instead of K neighbors, we use all points within a fixed radius R. Define the neighborhood as:

$$\mathcal{N}_R(\mathbf{x}) = \{i : d(\mathbf{x}, \mathbf{x}_i) \leq R\}$$

  Then apply similar formulas as in standard or weighted KNN.

KNN decision areas

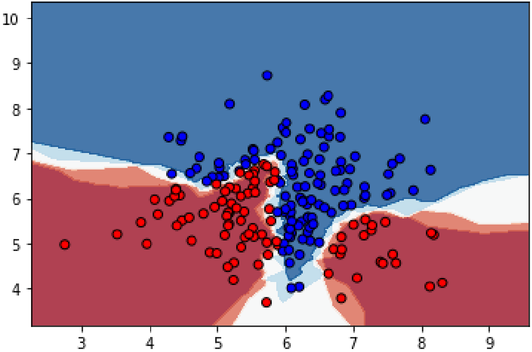


Figure: Decision boundaries for KNN classification with $K = 4$ neighbors. The colors represent the predicted class and associated confidence for each region.

Limits of KNN

- Are we done with (supervised) machine learning? Have we found the perfect classification and regression algorithm?

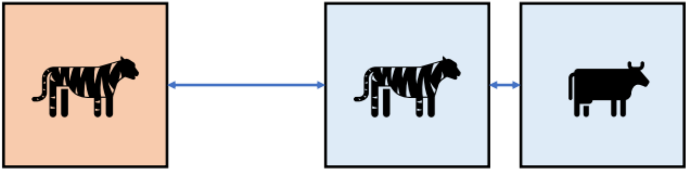


Figure: Measuring pixel distances. The center image is closer to the image on the right than the one on the left. Is it a cow?

- (Un)fortunately, it looks like **no, we are not**.

The curse of dimensionality

In high-dimensional spaces, the distances between points become less meaningful.

Outline

Unsupervised learning

We will now shortly introduce a first unsupervised learning algorithm: **K-Means**.

K-Means is a *clustering* algorithm.

Clustering

Clustering is the task of dividing a set of objects into groups, or **clusters**, in such a way that **objects in the same cluster are more similar** to each other than to those in other clusters.

A word of caution

- How can we group these vehicles?
  - Motor / no motor?
  - Touches the ground / does not touch the ground?
  - Single passenger / multiple passengers?

Clustering can be subjective

There can be a part of subjectivity in how we define groups or the idea of "close to each other"

Back to Euclidean space

- Let's focus on a simple, 2-dimensional, Euclidean example for now.

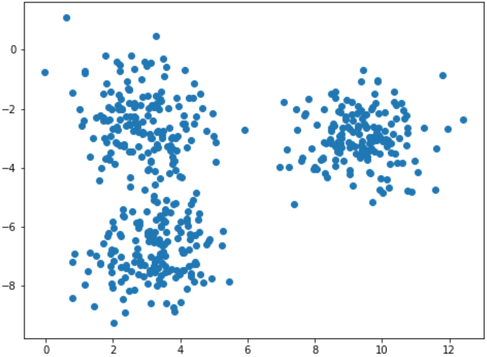


Figure: How many clusters are there?

- How could we define what would constitute "good" clusters here?

What makes “good” clusters?

- Intuitively, we want points in the same cluster to be:
  - Close to each other
  - Far from points in other clusters
- How can we do that? For example, we could:
  - Define a representative point for each cluster
  - Assign points to the closest representative
- This way we can measure ”closeness” using distance to the representative
- A natural choice for the representative is the center of the cluster

Key ideas

- Use cluster centers as representatives
- Assign points to the nearest center
- Minimize the total distance between points and their assigned centers

Formalization

Given N unlabeled points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_n \in \mathbb{R}^D$, we want to:

- Find K representatives $\{\mathbf{c}_k\}_{k \in [1,K]} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}, \quad \mathbf{c}_k \in \mathbb{R}^D$
- Find an assignment a that assigns each point $\mathbf{x}_n$ to one cluster $\mathbf{c}_k$:
  - $a : \mathbb{R}^D \to [1, K]$
  - define $a_n := a(\mathbf{x}_n)$ the assignment of point $\mathbf{x}_n$ $\forall n$

Objective:

$$\underset{\text{w.r.t. } \{a_n\}_n, \{\mathbf{c}_k\}_k}{\text{minimize}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{c}_{a_n}\|_2^2$$

- That is, we want to find centroids and assignments such that each point $\mathbf{x}_n$ is as close as possible to its centroid $\mathbf{c}_{a_n}$

Formalization

Given N unlabeled points $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\quad \mathbf{x}_n \in \mathbb{R}^D$, we want to:

- Find K representatives $\{\mathbf{c}_k\}_{k\in[1,K]} = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$, $\quad \mathbf{c}_k \in \mathbb{R}^D$
- Find an assignment a that assigns each point $\mathbf{x}_n$ to one cluster $\mathbf{c}_k$:
  - $a : \mathbb{R}^D \to [1, K]$
  - define $a_n := a(\mathbf{x}_n)$ the assignment of point $\mathbf{x}_n$ $\forall n$

Objective:

$$\underset{\text{w.r.t. } \{a_n\}_n, \{\mathbf{c}_k\}_k}{\text{minimize}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{c}_{a_n}\|_2^2$$

- That is, we want to find centroids and assignments such that each point $\mathbf{x}_n$ is as close as possible to its centroid $\mathbf{c}_{a_n}$

One tiny issue

This problem is NP-hard

K-means: a greedy heuristic approach

Objective (reminder):

$$\underset{\text{w.r.t. } \{a_n\}_n, \{\mathbf{c}_k\}_k}{\text{minimize}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{c}_{a_n}\|_2^2$$

The K-means algorithm is a greedy heuristic for this objective, that alternates between two steps:

1. Optimize assignments $\{a_n\}_n$ given fixed representatives, also called *centroids* $\{\mathbf{c}_k\}_k$
2. Optimizing centroid locations $\{\mathbf{c}_k\}_k$ given fixed assignments $\{a_n\}_n$

This approach is computationally efficient, and reasonably intuitive.

K-means algorithm: formalization

- Initialize centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ randomly
- Assign each point $\mathbf{x}_n$ to the nearest centroid $\mathbf{c}_k$

$$a(\mathbf{x}_n) = \text{argmin}_{k \in [1,K]} \|\mathbf{x}_n - \mathbf{c}_k\|_2^2$$

- Update clusters as the mean of the assigned points:

$$\mathbf{c}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x} \in \mathcal{C}_k} \mathbf{x}$$

(Mean of points in cluster $\mathbf{c}_k$)

$$\mathcal{C}_k = \{\mathbf{x} | a(\mathbf{x}) = k\}$$

(Set of points $\mathbf{x}$ assigned to cluster $\mathbf{c}_k$)

Repeat

Repeat the last 2 steps until the assignments a_n stop changing.

K-Means: illustration

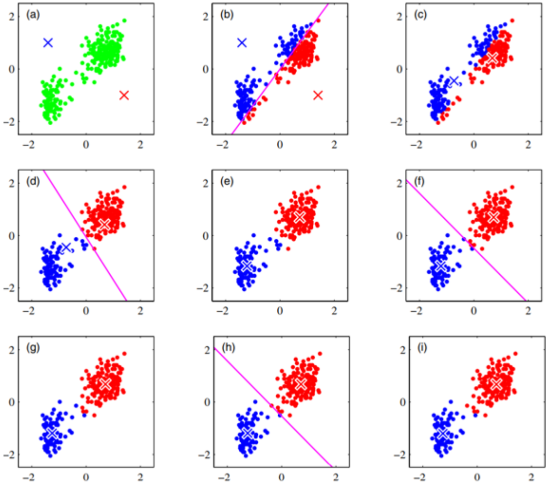


Figure: (a) Random initialization of the 2 centroids. (b) Assignment step: points are assigned to the nearest centroid. (c) Update step: centroid are updated to the mean of their assigned points. (d-h) Repeat. (i) Algorithm has converged. From Bishop.

Outline

Questions about K-Means

Will this algorithm always converge?

-

Does it always produce the same solution?

-

Does it find the optimal solution?

-

Questions about K-Means

Will this algorithm always converge?

- This is the hardest of the 3 questions. Answer in a few slides.

Does it always produce the same solution?

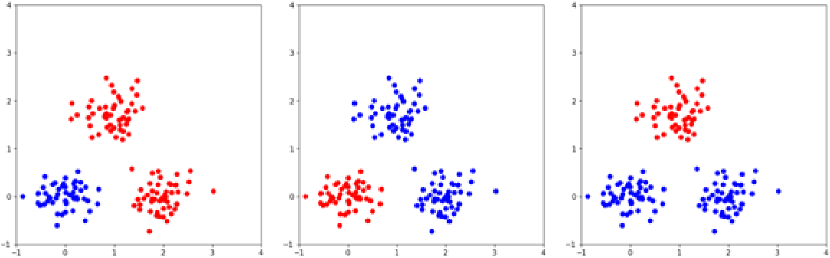
- **Hint:** Think about the initialization step.

Does it find the optimal solution?

- **Hint:** Consider the answer to the previous question.

Does K-Means always produce the same, optimal solution?

- No, different initializations can lead to different results



K-Means in practice

It is good practice to run the algorithm several times

Convergence of K-Means

Outline of the proof of convergence:

- Let's consider the cost function $J(\{\mathbf{c}_k\}_k, \{a_n\}_n) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{c}_{a_n}\|_2^2$
- Given fixed clusters $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, assigning points decreases the cost: $a(\mathbf{x}) = \text{argmin}_{k \in [1,K]} \|\mathbf{x} - \mathbf{c}_k\|_2^2$
- Given a fixed assignment $\{a_1, \ldots, a_N\}$, updating clusters decreases the cost: $\mathbf{c}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x} \in \mathcal{C}_k} \mathbf{x}$
- So each iteration of K-Means decreases the cost J
- The cost J is > 0 and there is only a finite, though quite large, number of possible assignments $\{a_1, \ldots, a_N\}$
- So K-means always converges

(Convergence is reached when the assignment stops changing)

Convergence of K-Means

Outline of the proof of convergence:

- Let's consider the cost function
  $J(\{\mathbf{c}_k\}_k, \{a_n\}_n) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{c}_{a_n}\|_2^2$
- Given fixed clusters $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$, assigning points decreases the cost:
  $a(\mathbf{x}) = \text{argmin}_{k \in [1,K]} \|\mathbf{x} - \mathbf{c}_k\|_2^2$
- Given a fixed assignment $\{a_1, \dots, a_N\}$, updating clusters decreases the cost: $\mathbf{c}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x} \in \mathcal{C}_k} \mathbf{x}$
- So each iteration of K-Means decreases the cost J
- The cost J is > 0 and there is only a finite, though quite large, number of possible assignments $\{a_1, \dots, a_N\}$ [1]
- So K-means always converges

(Convergence is reached when the assignment stops changing)

[1] Specifically, K^N

Question time!

- Which clusters will we find if we apply K-Means with $K = 2$?

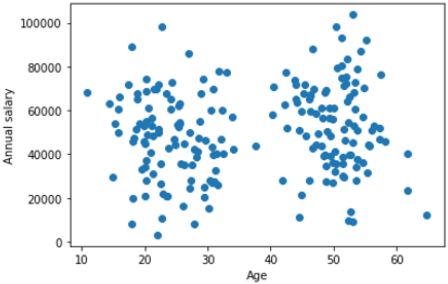
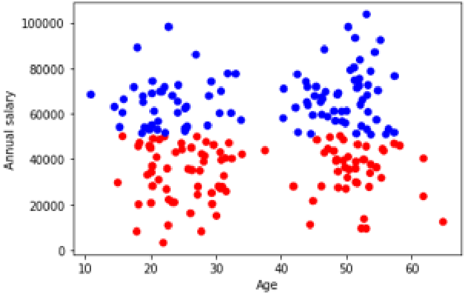


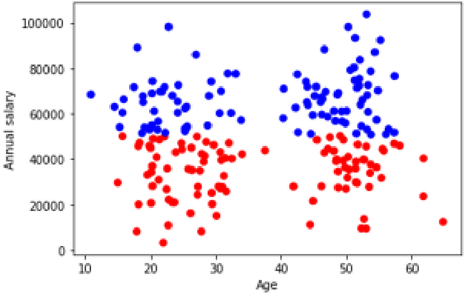
Figure: Some points forming 2 clusters.

~~(Trick) question time!~~ Answer time



- These ones!
- Why?

~~(Trick) question time!~~ Answer time



- These ones!
- Why? Because the scale of each variable (age and salary) is very different.

K-Means in practice

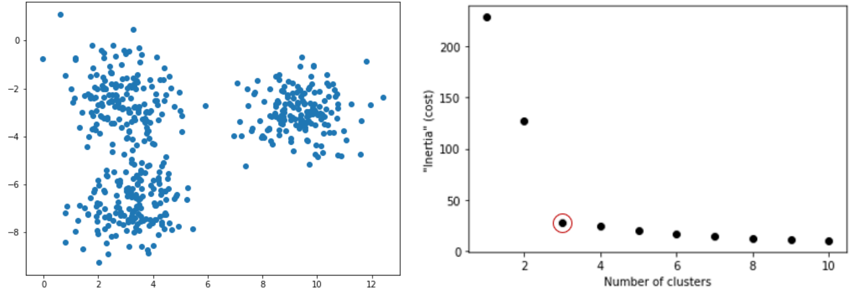
Standardization: it is often useful to "standardize" variables:

$$x_{\text{stand}} = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ the standard deviation of all x.

How to choose K: elbow method

- K is easy to estimate in 2 dimensions. Not so much when $D = 200$.
- The (*theoretically* optimal) cost $\sum_{n=1}^{N} \left\| \mathbf{x}_n - \mathbf{c}_{a(\mathbf{x}_n)} \right\|_2^2$, sometimes also called the *inertia*, decreases with the number of clusters K.



- We can select a value corresponding to a sharp decrease, $K = 3$ in this example ("elbow method")

How to choose K, alternative method: silhouette score

- For point $\mathbf{x}_n$, we define u_n the average distance to points of the corresponding cluster $\mathcal{C}_{a_n}$:

$$u_n = \frac{1}{|\mathcal{C}_{a_n}|} \sum_{\mathbf{x}_j \in \mathcal{C}_{a_n}} d(\mathbf{x}_n, \mathbf{x}_j)$$

- We also define $v_n = \min\limits_{\substack{k \\ k \neq a_n}} \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_j \in C_k} d(\mathbf{x}_n, \mathbf{x}_j)$

How to choose K, alternative method: silhouette score

- For point $\mathbf{x}_n$, we define u_n the average distance to points of the corresponding cluster $\mathcal{C}_{a_n}$:

$$u_n = \frac{1}{|\mathcal{C}_{a_n}|} \sum_{\mathbf{x}_j \in \mathcal{C}_{a_n}} d(\mathbf{x}_n, \mathbf{x}_j)$$

- We also define $v_n = \min_{\substack{k \\ k \neq a_n}} \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_j \in C_k} d(\mathbf{x}_n, \mathbf{x}_j)$

  i.e. the average distance to points in the closest different cluster $\mathcal{C}_k$, $k \neq a_n$
- The *silhouette score* s_n of point $\mathbf{x}_n$ is

$$s_n = \frac{v_n - u_n}{\max\{u_n, v_n\}} \quad \text{(and } s_n = 0 \text{ if } |\mathcal{C}_{a_n}| = 1). \qquad -1 \leq s_n \leq 1 \forall n$$

- An average silhouette score $S = \frac{1}{N} \sum s_n$ close to 1 corresponds to "good" clusters

How to choose K, alternative method: silhouette score

- For point $\mathbf{x}_n$, we define u_n the average distance to points of the corresponding cluster $\mathcal{C}_{a_n}$:

Warning

I am actually not a big fan of the silhouette score.

- We also define $v_n = \min\limits_{\substack{k \\ k \neq a_n}} \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_j \in C_k} d(\mathbf{x}_n, \mathbf{x}_j)$

  i.e. the average distance to points in the closest different cluster $\mathcal{C}_k$, $k \neq a_n$
- The *silhouette score* s_n of point $\mathbf{x}_n$ is

$$s_n = \frac{v_n - u_n}{\max\{u_n, v_n\}} \quad \text{(and } s_n = 0 \text{ if } |\mathcal{C}_{a_n}| = 1). \qquad -1 \leq s_n \leq 1 \forall n$$

- An average silhouette score $S = \frac{1}{N} \sum s_n$ close to 1 corresponds to “good” clusters

How to choose K, alternative method: silhouette score

- For point $\mathbf{x}_n$, we define u_n the average distance to points of the corresponding cluster $\mathcal{C}_{a_n}$:

Warning

I am actually not a big fan of the silhouette score.

- We also define $v_n = \min \frac{1}{\ } \sum \quad d(\mathbf{x}_n, \mathbf{x}_i)$

K-Means in practice

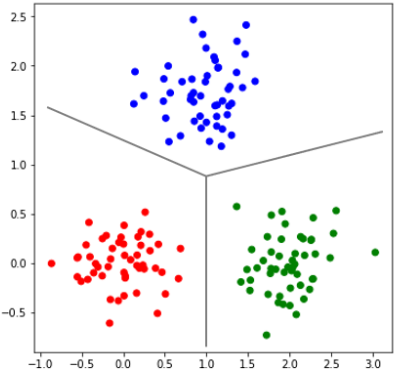
Why? Because it makes it too easy to simply choose the K that gives the "best" silhouette score $S(K)$. This can make us forget that the choice of K is always somehow arbitrary. In the elbow at least, this part is clear.

$$s_n = \frac{v_n - u_n}{\max\{u_n, v_n\}} \quad \text{(and } s_n = 0 \text{ if } |\mathcal{C}_{a_n}| = 1\text{)}. \qquad -1 \leq s_n \leq 1 \forall n$$

- An average silhouette score $S = \frac{1}{N} \sum s_n$ close to 1 corresponds to "good" clusters

Limits of K-means

- K-means can be considered "linear" in that cluster boundaries are linear.



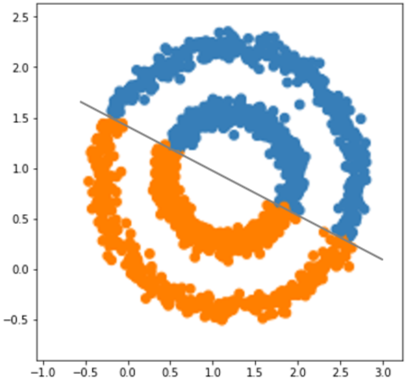


Figure: Left: linearly separable clusters. Right: non-linearly separable clusters.

Setting up a Python Environment with Conda

1. Download and install Miniconda
2. Create a new Conda environment:
   ```
   conda create -n mlclass python=3.10
   ```
3. Activate the environment:
   ```
   conda activate mlclass
   ```
4. Install necessary packages using pip:
   ```
   pip install scikit-learn jupyter matplotlib
   ```
5. Launch Jupyter Notebook:
   ```
   jupyter notebook
   ```

Alternatively, you may use Google Collab, Visual Studio Code, or any other method that lets you read, modify and run iPython Notebooks.