MATLAB

Class 5: data reduction

What is data reduction?

Can you describe your crush along the following dimensions: Hair, height, eye colour, smile, personality

What is data reduction?

Can you describe your crush along the following dimensions: Hair, height, eye colour, smile, personality

Can you describe your crush on the same dimensions using fewer words?

Why would you want to reduce your dimensions

Data sometimes comes to us in a high dimensional space (e.g., EEG recordings from 200+ electrodes). While we are interested in global patterns we may not be interested in every electrode. We want a way to summarize information across the hundred of variables (i.e., electrodes)

Data reduction

Like a questionnaire in psychology research, observations of all your measures (i.e., questions on a questionnaire) can be described by latent variables that may be of interest to you (i.e., personality)

Data reduction

Are you witty?

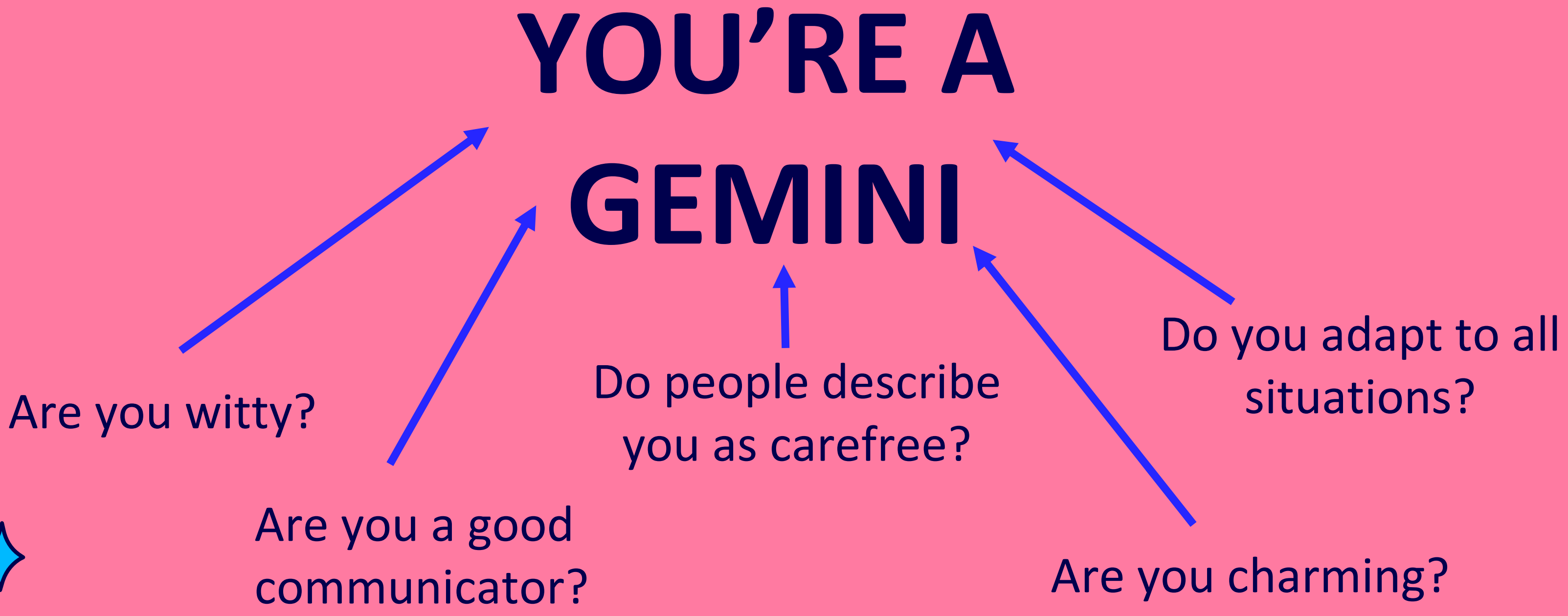
Are you a good communicator?

Do people describe you as carefree?

Are you charming?

Do you adapt to all situations?

Data reduction



YOU'RE A GEMINI

Are you witty?

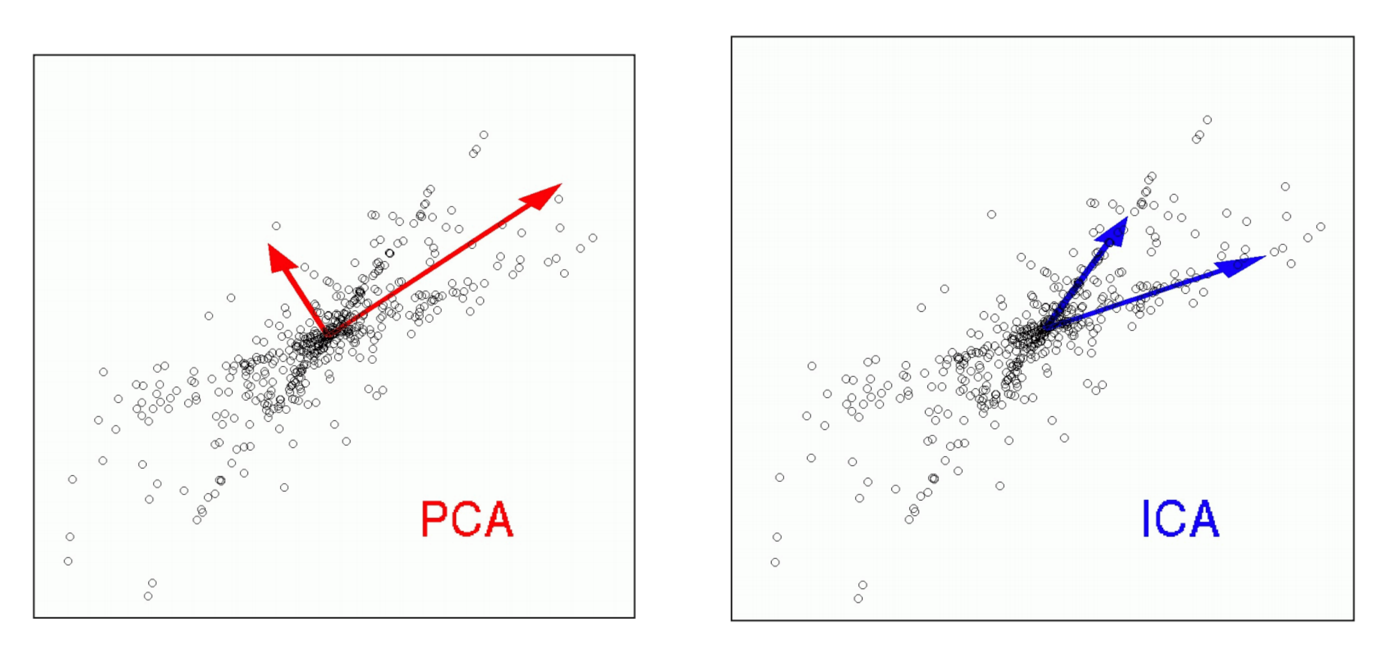
Are you a good communicator?

Do people describe you as carefree?

Are you charming?

Do you adapt to all situations?

ICA vs PCA



Principal Component Analysis

Principal Component Analysis (PCA) extracts relevant orthogonal features, that we call **Principal Components**, from a large pool of variables.

In **PCA** we attempt to find dimensions that maximize variance

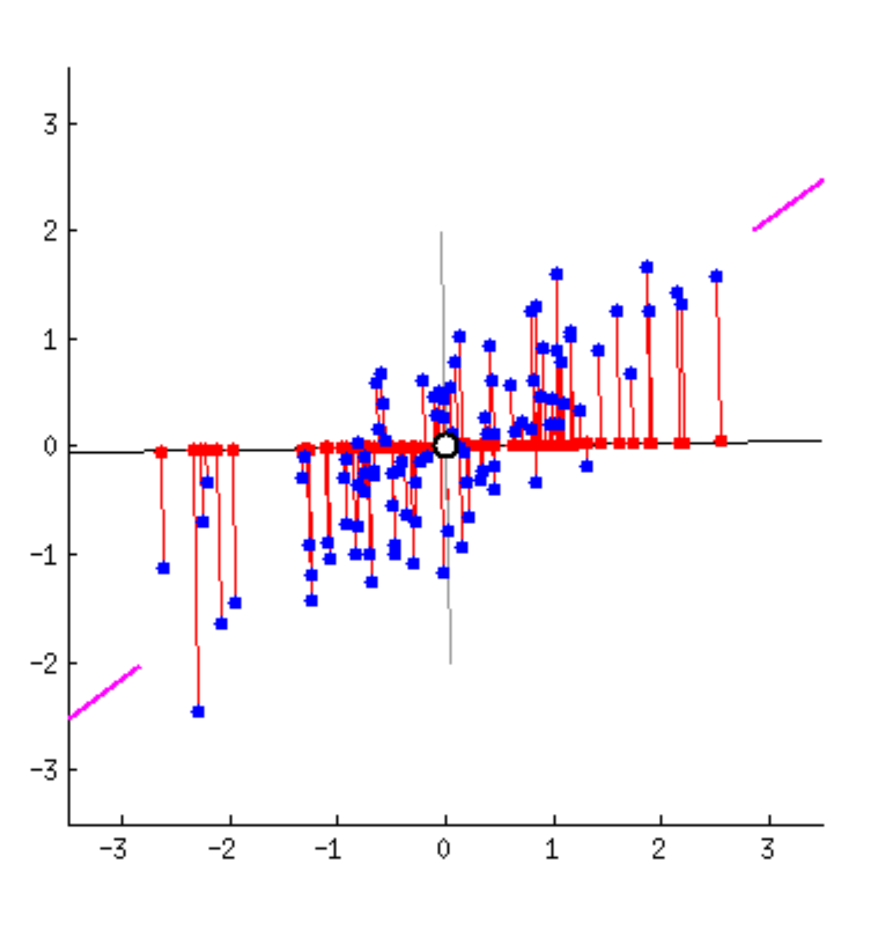
Principal Co relevant

orthogona

Compone es.

In **PCA** we at aximize

variance



PCA

[coeff, score, latent, tsquared, explained, mu] = pca(**X**)

X -> data matrix (m by n) to apply PCA (rows observations, col variables)

Coeff -> matrix (n by n) PCA coefficients each col is one PC, rows are weights for your variables

Score -> matrix (m by n) how much a given observation 'scores' or 'loads' onto a PC

PCA

[coeff, score, latent, tsquared, explained, mu] = pca(**X**)

Latent -> eigenvalues of the covariance matrix of X

Tsquared -> Sum of squares of standardized scores

Explained -> Percent variance explained per component

Mu -> estimated means of each variable

PC

[coef

Normalizing data

In Generally, you want to normalize your data **BEFORE** running a PCA both by subtracting the mean of each column and dividing by the standard deviation of each column

You can't compare apples to oranges or cm to volts

MUCH BETTER

A Note

Since PCA is a **linear transformation**, translating our data into a new space, we can simply rotate it back to where it began:

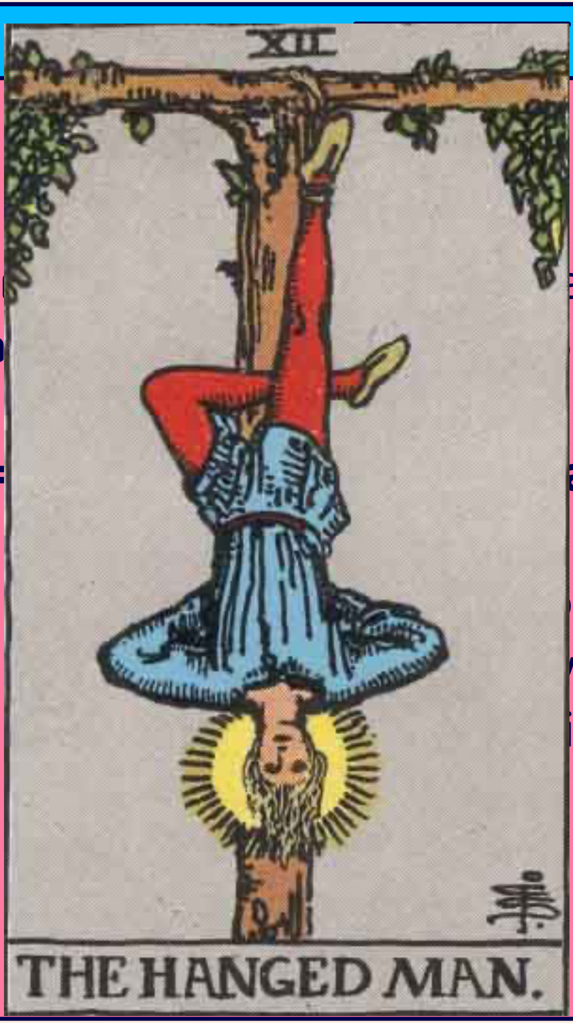
```
Original_data = score*coeff ' + mean(data);
```

Where the **score** is the loadings of the components and **coeff** is the weights matrix that was used to rotate your data.
Notice how we add back the mean of our Original data.

A Note

Since PCA is a **linea** ating our data into a
new space, we ca where it began:

Original_data = ata);



Where the **score** is onents and **coeff** is
the weights matrix our data.
Notice how we add iginal data.

A Note

Since PCA is a **linea**

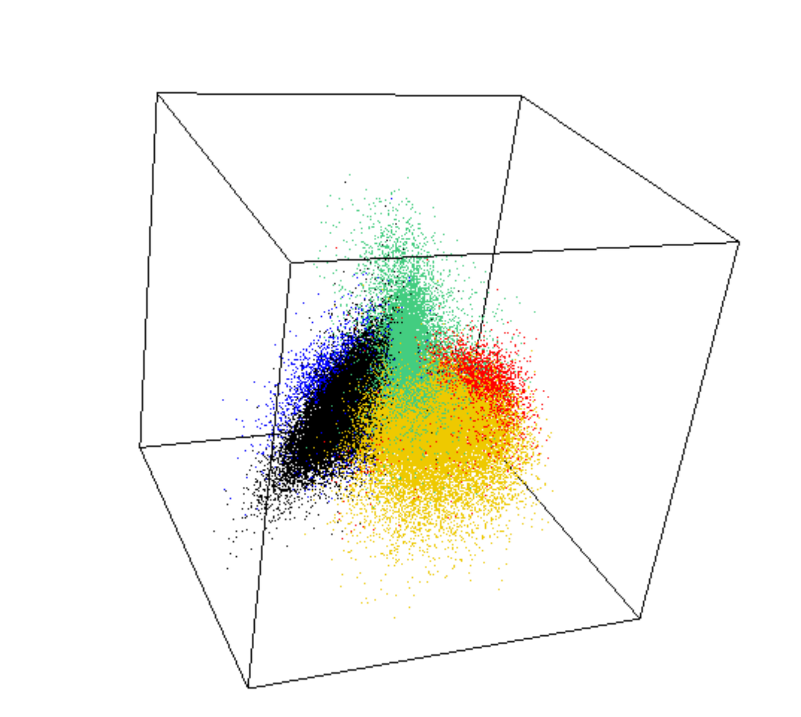
ating our data into a new space, we ca where it began:

Original_data = **ata);**

Where the **score** is onents and **coeff** is the weights matrix our data.
Notice how we add iginal data.

A Note

Since PCA is tr

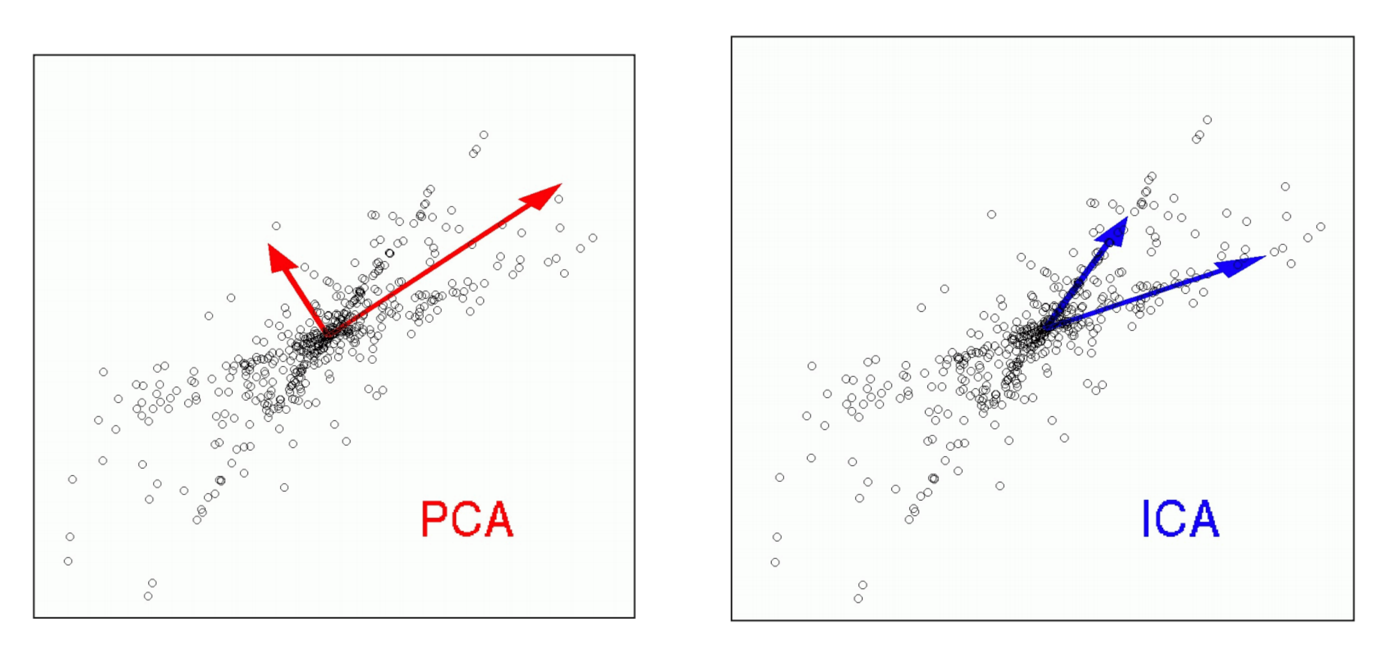


, we can simply rotate it bac

Original_c

Where the **sco** nts and **coeff** is the weights m data. Notice how we add b

ICA

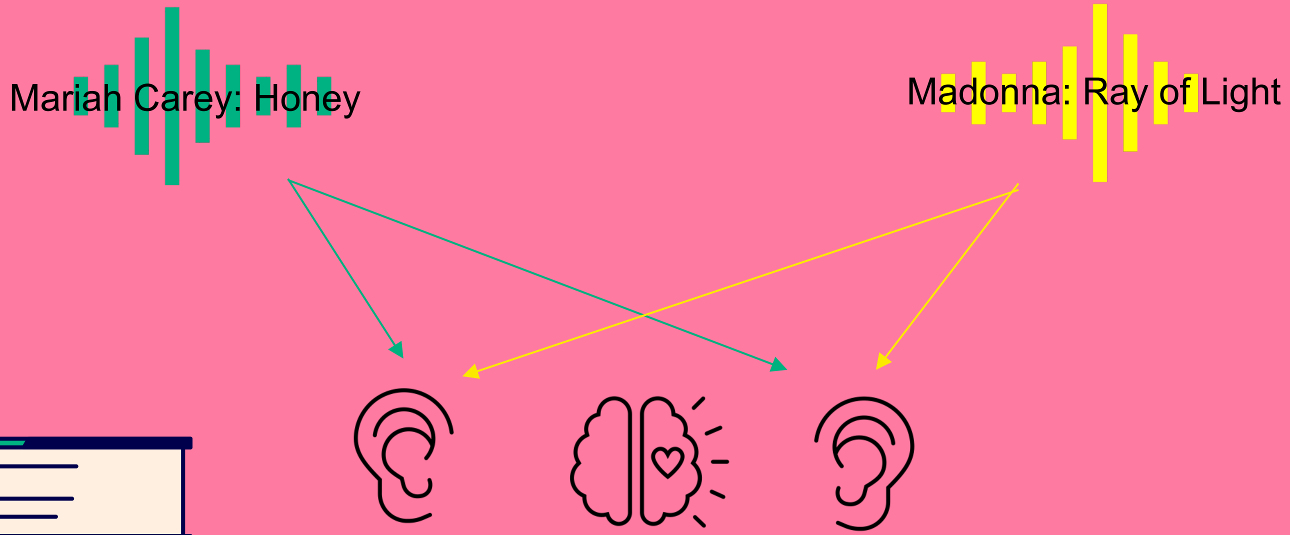


Blind Source Separation Problem

Problem described in mathematics where there are two signals (i.e., sources) that you must disentangle from one another. But how do we go about solving this problem?

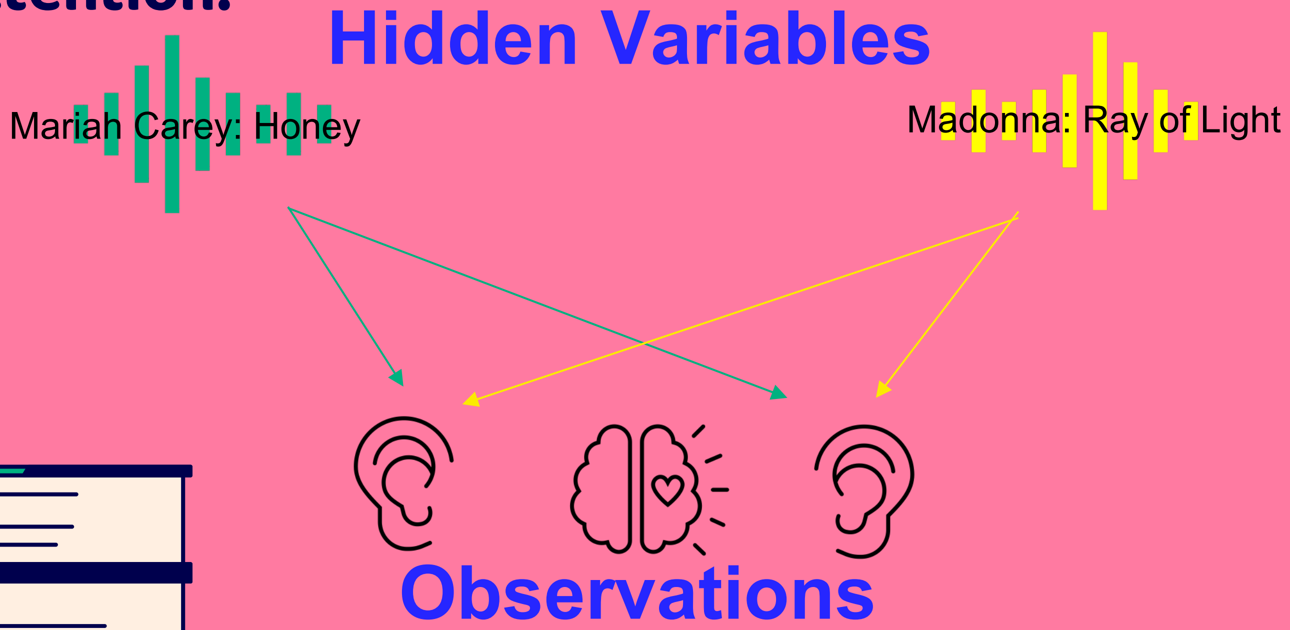
Blind Source Separation Problem

Our brain does it naturally through selective attention:



Blind Source Separation Problem

Our brain does it naturally through selective attention:

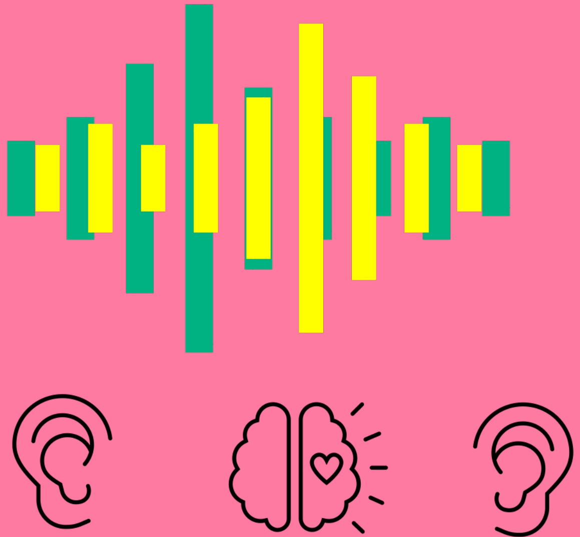


Hidden Variables

Observations

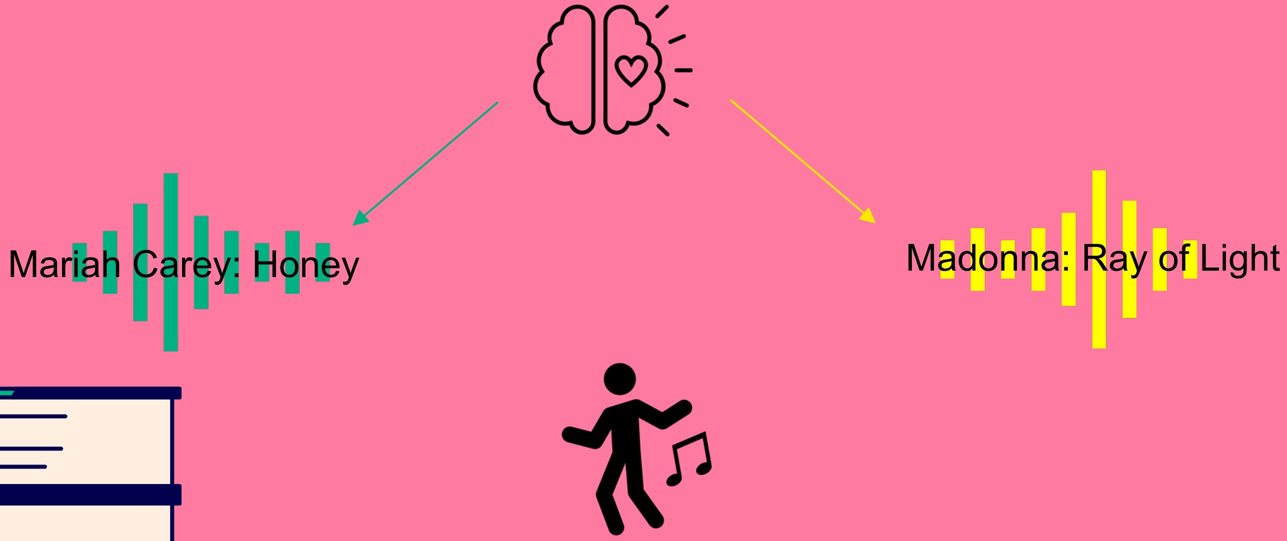
Blind Source Separation Problem

Our brain does it naturally through selective attention:



Blind Source Separation Problem

Our brain does it naturally through selective attention:



Mariah Carey: Honey

Madonna: Ray of Light

Independent Component Analysis

Independent component analysis is one way we have approached this problem mathematically.

This algorithm attempts to decompose a complex signal into its many sub-signals or sources

Independent Component Analysis

Thus, it answers a different question than PCA

Whereas PCA answers the question how can you represent your data with new dimensions, ICA asks what sources make up your complex signal

Independent Component Analysis

ICA finds components that are **statistically independent** from one another (i.e., independent sources)

There are various methods to solve this (e.g., FastICA, infomax etc.) That rely on different strategies like minimizing the **Mutual Information** between components

ICA

[coeff, score, latent, tsquared, explained, mu] = pca(**X**)

X -> data matrix (m by n) to apply PCA (rows observations, col variables)

Coeff -> matrix (n by n) PCA coefficients each col is one PC, rows are weights for your variables

Score -> matrix (m by n) how much a given observation 'scores' or 'loads' onto a PC

A Note

ICA is typically used to remove components or sources from your data (i.e., clean the data of artifacts)

Similarly, ICA is used to extract sources from noisy data

This is a type of feature extraction if you believe your data is coming from a fixed number of independent sources which are linearly mixed together.

A Note

Underdetermined matrices occur when you have less observations than you have sources (i.e., more songs mixed together than microphones to record the songs) . This may make finding the sources more difficult.

Sometimes we know how many sources we want to unmix from our data, or the source itself, other times we do not know the number or identity of the source

Comparison

PCA

Finds orthogonal components

Finds components that maximize variance

Features ordered in terms of variance explained

ICA

Finds Statistically Independent (⬇️ MI) components

Solution for the blind source separation problem

Finds components that share no Mutual Information

We can similarly reduce data by categorizing it into groups

K-means

The previous techniques allow you to reduce data in terms of their dimensions, attempting to describe observations along new dimensions that are more interesting to you

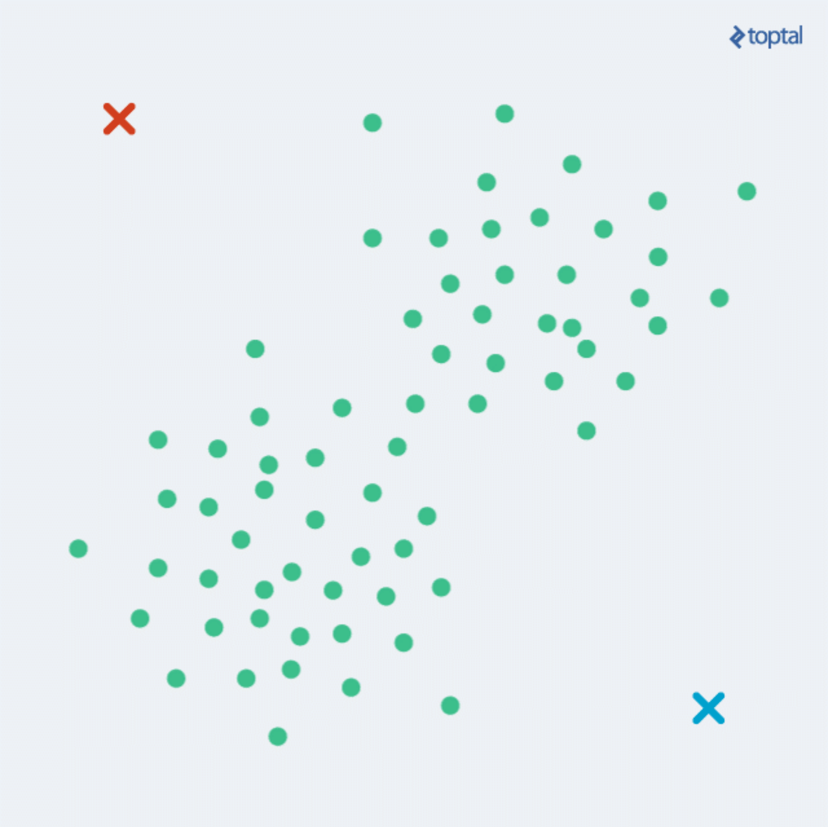
We can also reduce data by clustering or grouping them into a smaller number of cliques

K-means

K-means

An algorithmic way to determine which cluster a given observation belongs to

Based on a simple strategy where you move k clusters around a feature space F trying to reduce the distance between your data and the cluster k



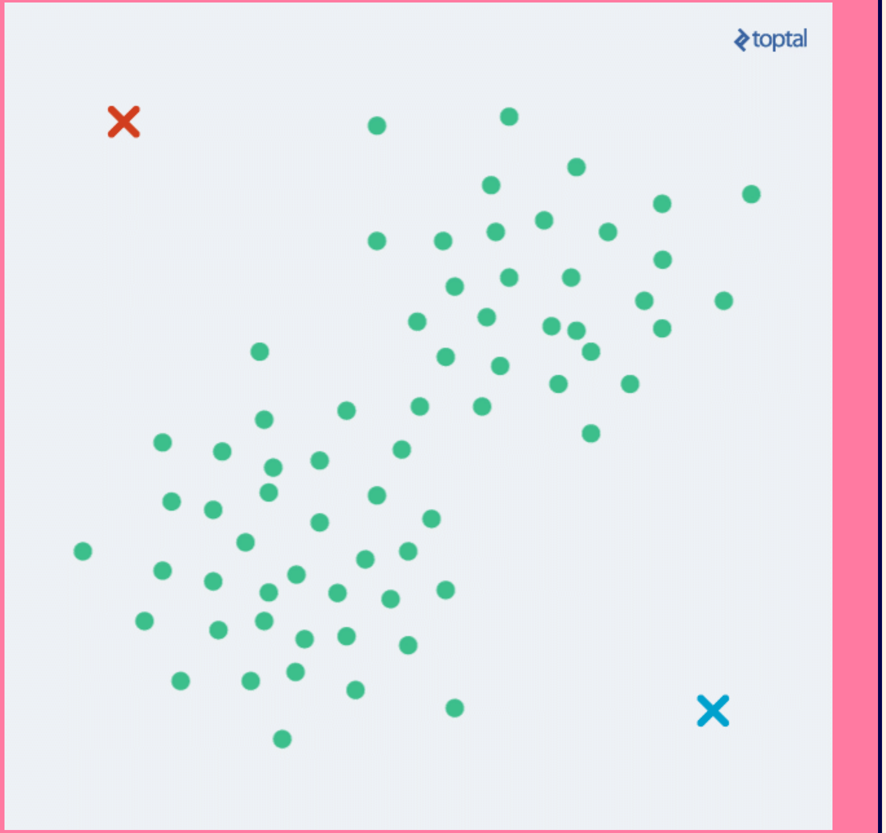
YOURMUMBRIDGE

K-means

We randomly place two centroids k1 (red) and k2 (blue).

We then compute the distance between each data point and both centroids.

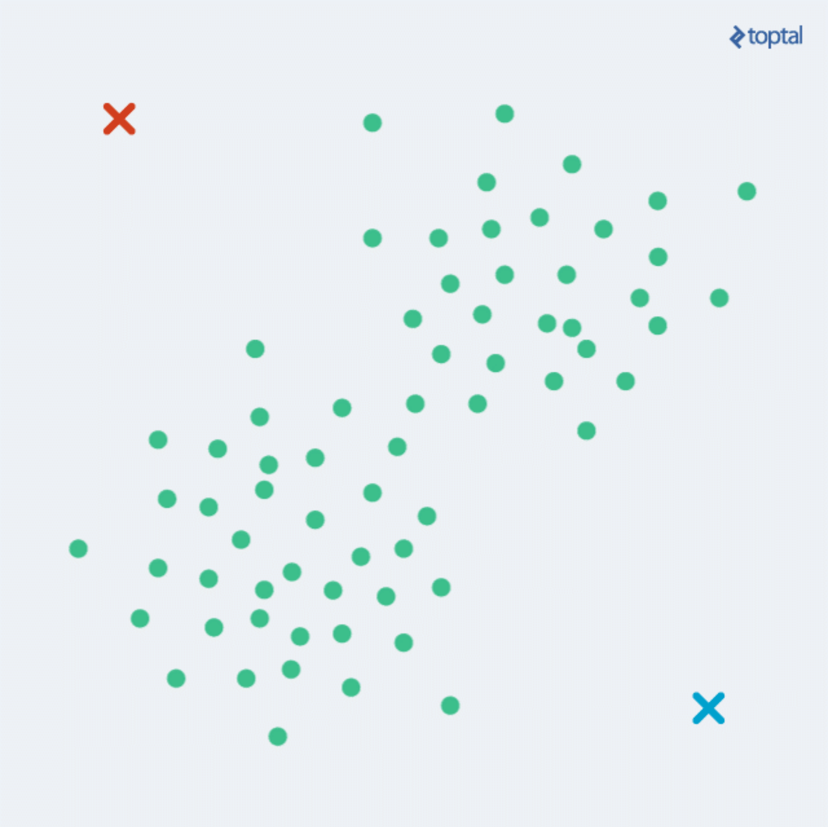
Classify each data point based on which centroid they are closest too



K-means

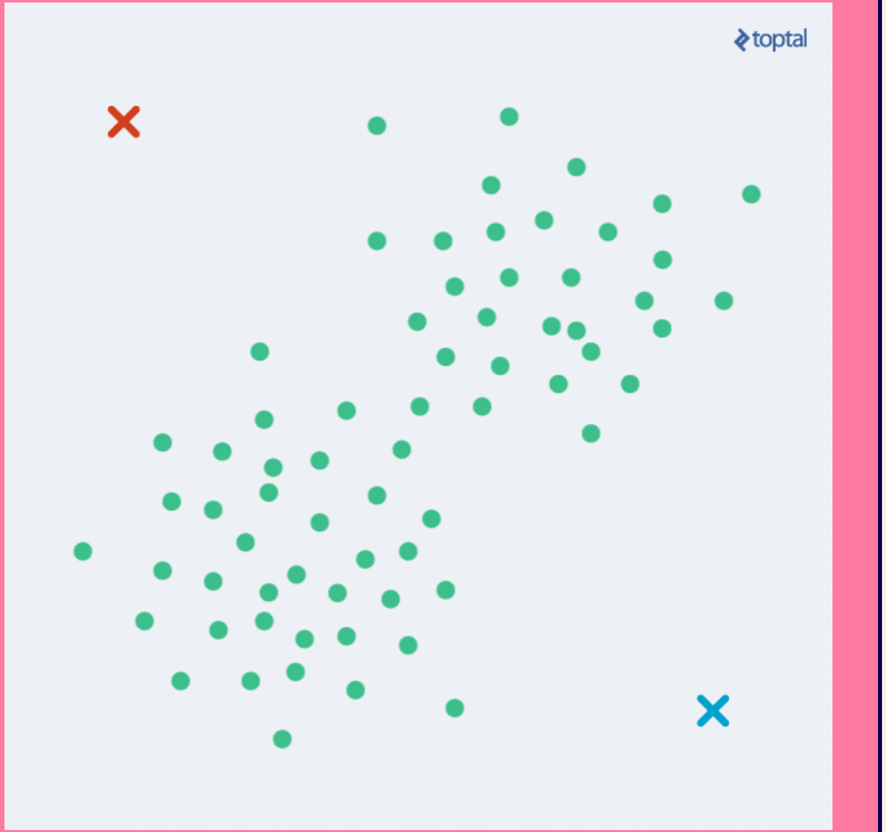
Move the centroids (red and blue) to the center of the data they are associated with

Repeat the above steps until the centroids stop moving around the feature space and settle down



K-means

It is necessary to standardize your data BEFORE running K-means. You want your features on equal footing. Features with large means or variance could be weighted more than your other features (i.e. meters vs micro volts)



Determining how many k is too many k

You can do this with a ***training*** and ***testing*** set. This is highly recommended as to avoid overfitting (see class 7)

Splitting your data into different subsets is common in machine learning algorithms

Normally you will have a large dataset to **train** on (i.e., teach your algorithm) and a smaller set to **test** if your algorithm works (i.e., see if it still works with other data)

Determining how many k is too many k

If you do not know *apriori* how many different classes or categories exist in your data, you need to compute this parameter

The optimal values of k are determined several ways:

Calinski-Harabasz criterion

Silhouette plots

Davies Bouldin index

Determining how many k is too many k

Calinski-Harabasz criterion

Ratio of between cluster dispersion against within cluster dispersion

Tight-knit clusters that are further apart get better scores, are described as better defined clusters

Determining how many k is too many k

Silhouette plots

For each point, measures how similar the observation is to its cluster. Values range from -1 to 1, unrelated to very related respectively

When looking at silhouette plots, one should expect to see all positive values that are very high (a 'box' per cluster)

Determining how many k is too many k

Davies Bouldin index

Compares the distance between clusters against the size of the cluster itself. Favours small clusters that are father apart