Unsupervised Learning

Why do we need unsupervised learning?

Unsupervised learning becomes important once we have data that don't have a label.

The idea is to come up with algorithms that are able to detect structures in a dataset.
For example, we might wanna try to find the clusters in the dataset.

K-Means Clustering

K-Means is an algorithm that allows us to detect clusters in dataset. A cluster can be seen as a group of points that have something in common. This typically means that the points are close to each other.

Algorithm:

1. Randomly pick n arbitary points as cluster centers.
2. Now let's compute the distance from every data point to each cluster center.
3. Assign each point to the closest cluster center.
4. We now try to optimize the positions of the cluster centers. Therefore, we **minimize** to total of the **quadratic distances** between each point of a cluster center and the cluster center.
5. Move the cluster centers.

The cost function looks as follows:

$$J = \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

Note: Solving the problem is not trivial (NP-hard!). Therefore, k-means algorithm only hopes to find the global minimum and possibly gets stuck at a local minimum.

Visualization: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Stability of k-means

Will the output of K-Means always be the same?

No, the result depends on the initial position of the cluster centers. Therefore, we can say that k-means **is not stable**.

Let's consider the following example:

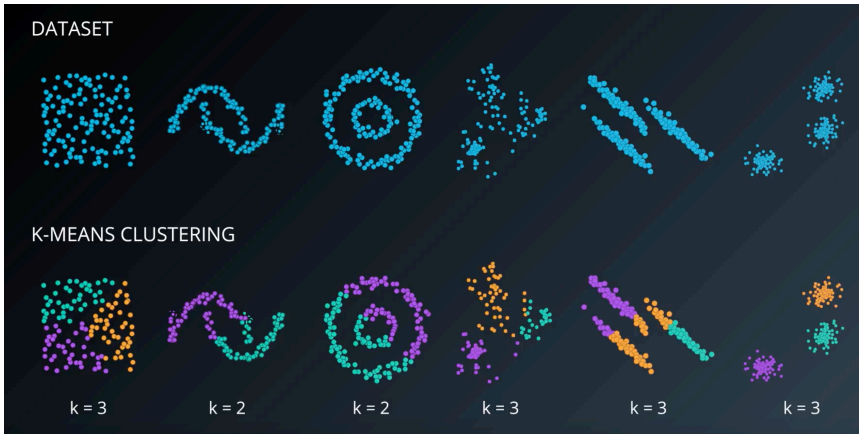
As we can see initializing k-means with the points shown in the image results in a very strange partitioning. Definitely, a partitioning we don't want.
However, if we initialize the points differently, we might get another partitioning with cluster centers on the left/right.

There is no guarantee that k-means converges to the "right" solution.

Downsides of k-means

K-Means does only work well if we know the number of cluster centers and if the data can be described by clusters that are **circular** or spherical or hyper-spherical in higher dimensions.

Let's take a look at the following examples to understand where k-means clustering fails.



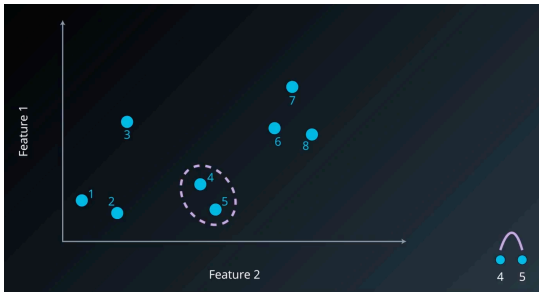
Hierarchical clustering

As the name already implies "hierarchical clustering" is a method of cluster analysis which seeks to build a hierarchy of clusters.
One specific form of hierarchical clustering is **single-link clustering**.

Single-link clustering

Algorithm:

1. Compute the distance between each and every point in the dataset.
2. Select the two points which are closest to each other.
3. Group the two points into a cluster
4. Also we note down that we've just added the points to a cluster in a hierarchical tree.

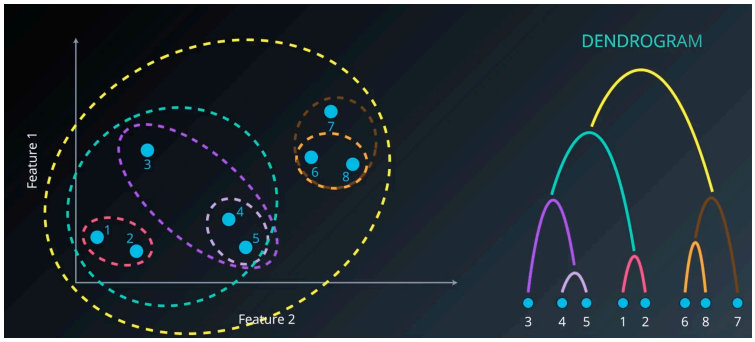


5. Proceed with step 1-4.

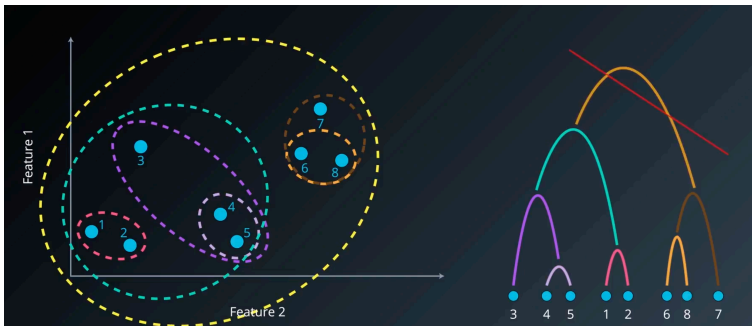
Important: If we want to compute the distance between a point to a cluster, we basically compute the distance between the point and the point from the cluster that is closest.

Note: Hierarchical clustering algorithms typical differ in the way how distances between points-clusters, respectively, clusters-clusters get computed.

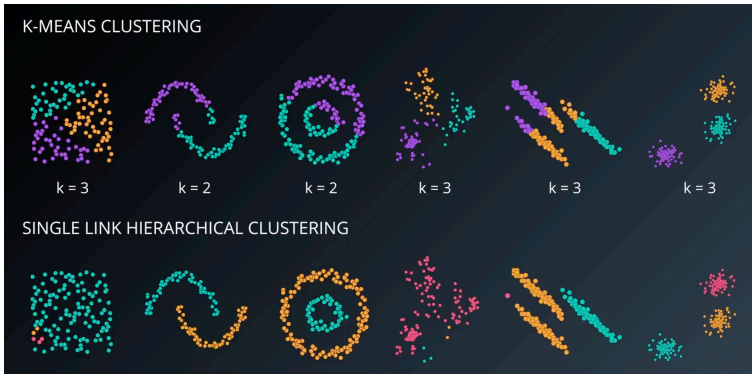
The final hierarchical cluster tree is also called **dendrogram**.



When we run the algorithm we need to tell the algorithm how many clusters we are looking for.
Based on the number of clusters we can limit the height of the dendrogram.



Comparison K-Means / Single-link clustering



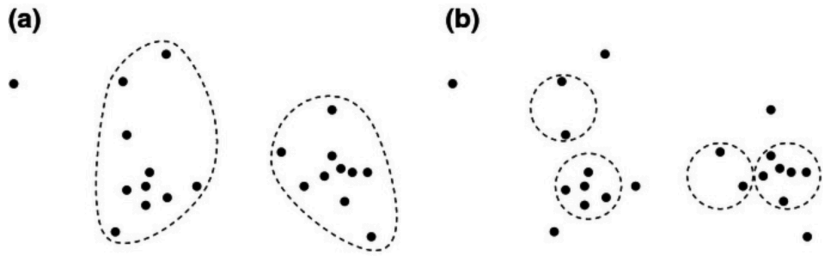
Complete-Link clustering

Complete-link clustering basically works in the same way as single-link clustering.
The only difference is the way how we compute distances between clusters.

In complete-linkage clustering, the link between two clusters contains all element pairs, and the distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved. The method is also known as farthest neighbour clustering.

Shortcomings of single and complete link clustering

- **Single linkage:** Often suffers from chaining, that is because, we only need a single pair of points to be close to merge two clusters. Therefore, clusters can be too spread out and not compact enough.
- **Complete linkage:** Often suffers from crowding, that is because, a point can be closer to points in other clusters than to points in its own cluster. Therefore, the clusters are compact, but not far enough apart.

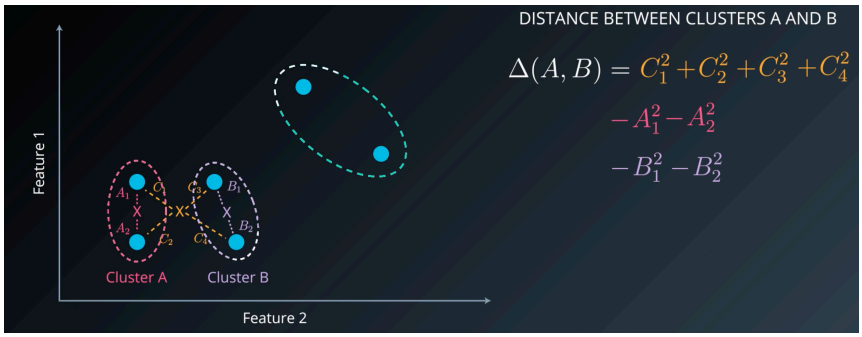


Average Link Clustering

We simply compute the average distance between all points of the cluster to all other points of another cluster.

Ward's Method

Tries to minimize the variance when merging two clusters. Ward's method tries to find a central point between two clusters. This can be done simply by averaging all the points.
We can then compute the sum of the squared distances between all points of two clusters to the central point. From this sum we then substract the existing variance of both clusters.



DB-Scan

DB-Scan stands for **density-based spatial clustering of applications with noise**.

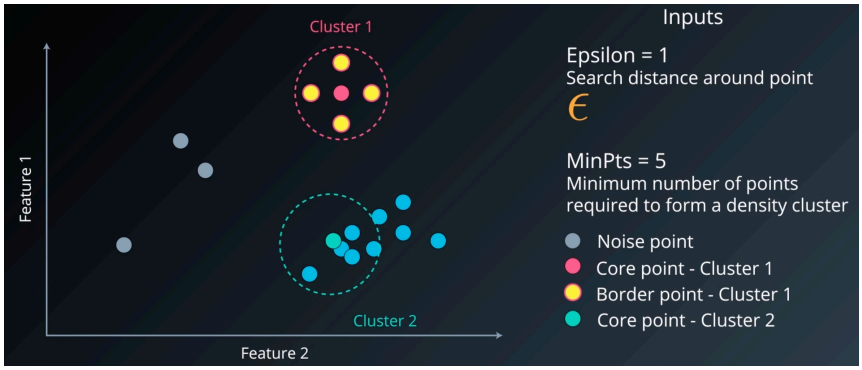
In DB-Scan not every point needs to be part of a cluster. Points that are not part of a cluster are considered as noise.

Algorithm:

1. Select an arbitary point in our dataset
2. Let's take a look at the surrounding neighbourhood of the point. Circular region with distance ϵ away

from the point. 

3. If there are less than n_{min} data points in the circular region, we mark the point as noise.
   If there are more than n_{min} data points in the circular region (incl. points which are already marked as noise), we have found a cluster.
   Note: The point we've selected is called **core "point**. The other points are called **border points**.
4. Dependent on whether we found a cluster or not we proceed as follows:
   If we found a cluster, we proceed by performing the same check with the border points. If this point satisfies your cluster criterion, we merge the cluster with our previous cluster.
   If not enough point were found, we proceed with step 1.



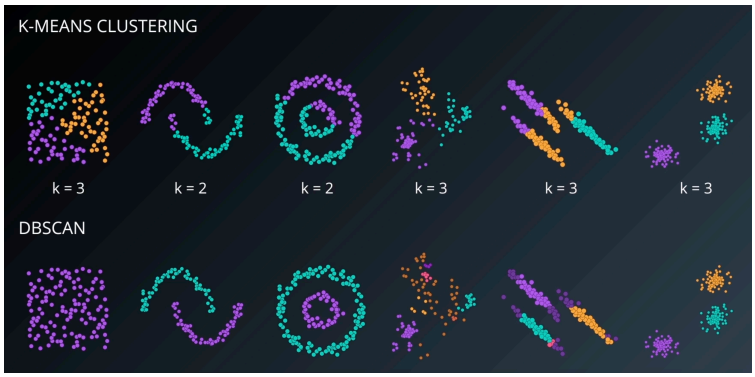
Advantages:

- We don't have to specify the number of clusters
- Flexibility in the shapes and sizes of clusters
- Able to deal with noise
- Able to deal with outliers

Disadvantage:

- Border points that are reachable from two clusters are assigned to the cluster that visits them first. So, DB-Scan is not guaranteed to return the same clustering all the time for this kind of data.

Comparison K-Means / DB-Scan

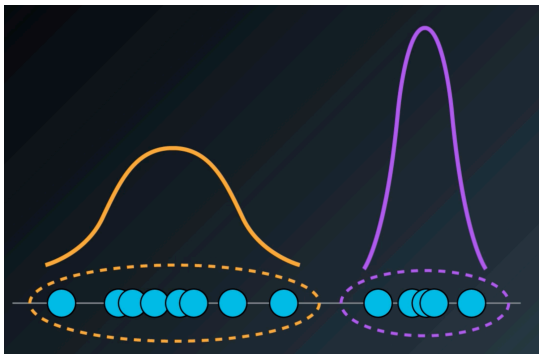


Gaussian Mixture Models

Gaussian Mixture Model clustering is a soft-clustering algorithm which means that every point belongs to **every** cluster we have with a certain probability.

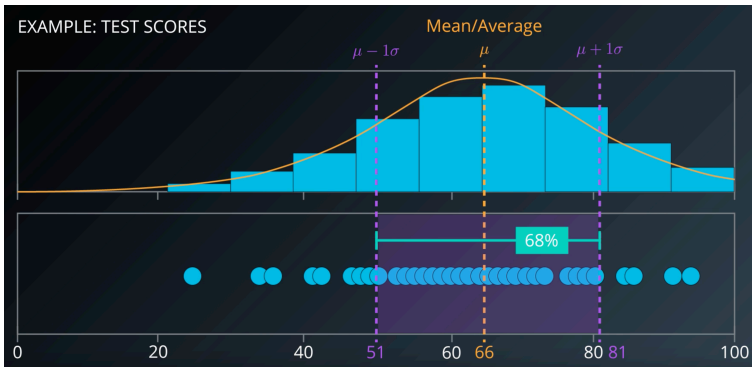
It's a little bit like saying... Hmm, these set of points look like they come from a Gaussian distribution that looks like this. This distribution forms our cluster.

Formally, we try to approximate your distribution by a set of Gaussian distributions.



Property of a Gaussian

One important property of a Gaussian distribution is the fact that the area between $\mu - 1\sigma$ and $\mu + 1\sigma$ contains 68% of all data points.
The area between $\mu - 2\sigma$ and $\mu + 2\sigma$ contains 95% of the points.
The area between $\mu - 3\sigma$ and $\mu + 3\sigma$ contains 99% of the points.



EM (Expectation Maximization) - Algorithm

Random variables:

- Z ... Hidden variable. Distribution selector
- X_i ... A normal distribution described by μ, Σ

$P(Z = i)$ ... Probability that we encounter distribution i
$P(X = x|Z = i) = p_{X,Z}(x, z_i)$ ... Probability that we encounter sample x in the distribution i.

What is the chance that we encounter sample x?

$$p_{X,Z}(x, z_i) = p_{X|Z}(x, z_i) \cdot P(Z = i)$$

$$p(x) = \sum_{i=1}^{c} p_{X|Z}(x, z_i) \cdot P(Z = i)$$

Finding the correct parameters

We now need to estimate the parameters μ, σ as well as the Z (the assignment of every point to a distribution).

$$ln(X|\mu, \sigma, \pi) = \sum_{n=1}^{N} ln(p(x_n)) = \sum_{n=1}^{N} ln(\sum_{i=1}^{c} p_{X|Z}(x_n, z_i) \cdot P(Z = i))$$

Note: π is the mixing coefficient that weights each Gaussian distribution.

The first idea we could come up with would be a maximum likelihood estimation. However, this does not work since there's no closed form solution to this problem. Hence we need to perform what's called **Expectation Maximization (EM) technique**.

Algorithm

1. Initialize the means μ_i, covariances Σ_i and mixing coefficients π_i.
   Typically initialization is done using k-means clustering.

2. **Estimation step**
   Evaluate the responsibilities for every data point using the current parameter values:

   $$\lambda_i(x) = \frac{\pi_i \cdot N(x|\mu_i, \Sigma_i)}{\sum_{j=1}^{c} \pi_j \cdot N(x|\mu_j, \Sigma_j)}$$

3. **Maximization step**
   Re-estimate the parameters using the current responsibilities:

   $$\mu_i = \frac{\sum_{n=1}^{N} \lambda_i(x_n) \cdot x_n}{\sum_{n=1}^{N} \lambda_i(x_n)}$$

   $$\Sigma_i = \frac{\sum_{n=1}^{N} \lambda_i(x_n) \cdot (x_n - \mu_i) \cdot (x_n - \mu_i)^T}{\sum_{n=1}^{N} \lambda_i(x_n)}$$

$$\pi_i = \frac{1}{N} \sum_{n=1}^{N} \lambda_i(x_n)$$

4. **Evaluate log likelihood**
   If there is no convergence, return to step 2.

Advantages / Disadvantages

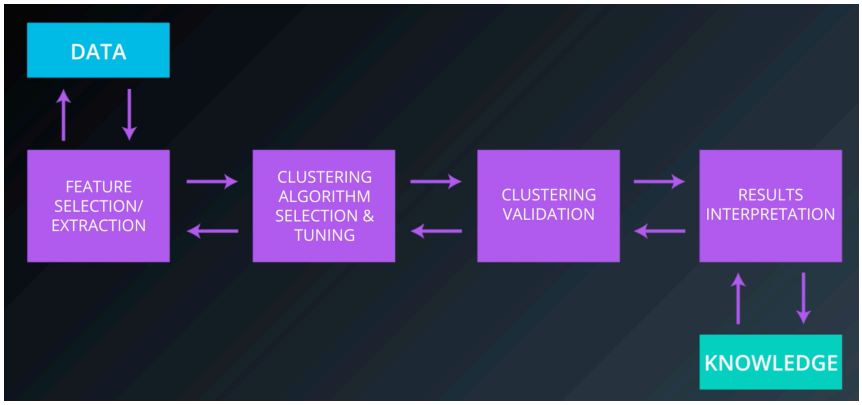
Advantages:

- Soft-Clustering (sample membership of multiple clusters)
- Cluster shape flexiblity

Disadvantages:

- Sensitive to initalization values
- Possible to converge to a local optimum
- Slow convergence rate

Cluster analysis process



Feature selection:
Choosing distinguishing features from a set o candidates. In other words, we select the best ones.

Feature extraction:
Transform the data to generate more useful features. For instance, by applying PCA

Clustering validation selection and tuning:
Select a clustering method and proxiimity measure (e.g. euclidean distance, cosine distance, pearson correlation)

Cluster validation:
Use scoring methods to validate the quality of a method. Such scoring methods are also called "indices".

Cluster validation

Cluster validation is the procedure of evaluation the results of a clustering objectively and quantitatively.

We differentiate between three categories of validation indices:

- **External indices:**
  External indices can be used if data are labeled.
- **Internal indices:**
  Measure the fit between the data and structure using only the data.
- **Relative indices:**
  Indicate which clustering structure is better in terms of *compactness* and *separability*. Basically all internal indices can be seen as relative indices.

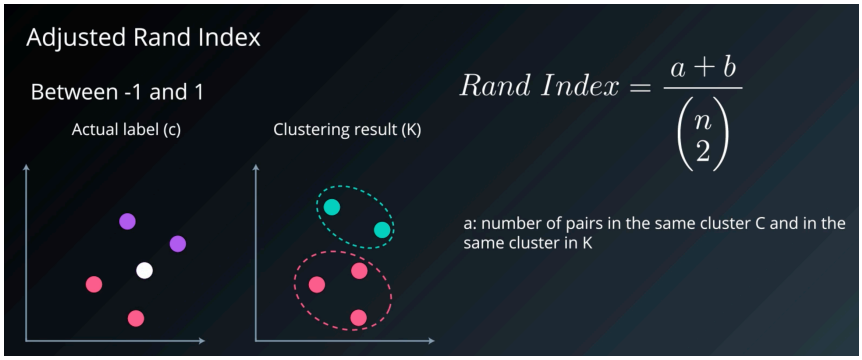
Compactness: Measures how close elements of a cluster are from each other.
Separability: Measures how far / distinct clusters are from each other.

External indices

The following indices can be used if we have a labeled dataset:

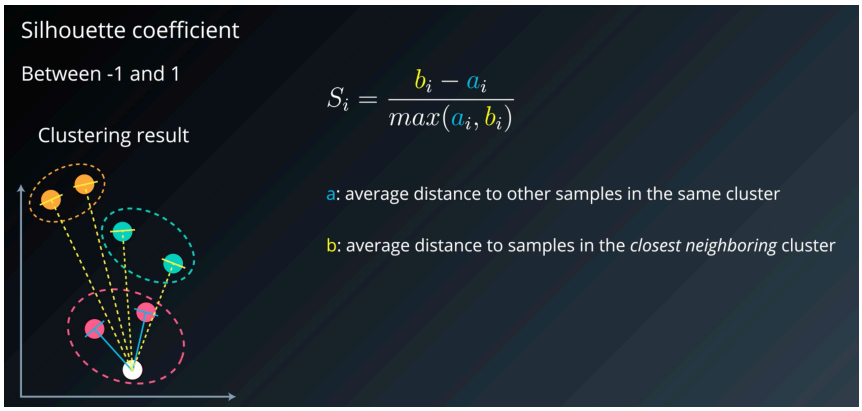
- Adjusted Rand index
- Fawlks and Mallows
- NMI measure
- Jaccard
- F-measure
- Purity

Adjusted Rand index

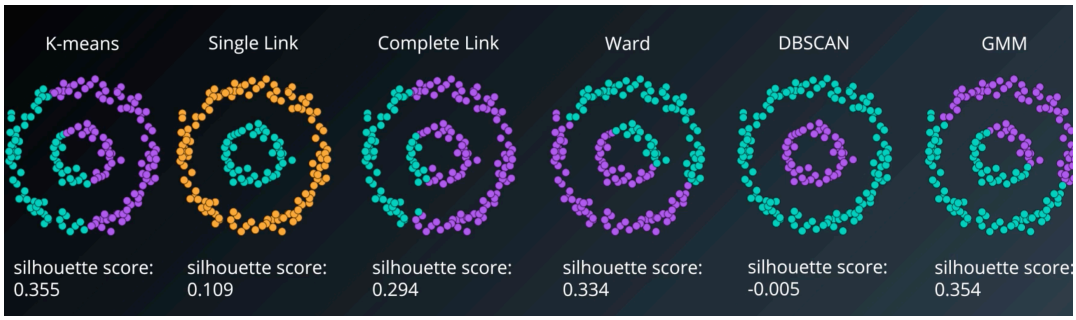


Internal indices

If we have unlabeled data we can use the so-called *Silhouette coefficient* measure the quality of our clustering.



Comparison - Silhouette coefficient for different datasets:



Note: Since the Silhouette coefficient look for compact cluster, it fails for ring-shaped structures.

Feature Scaling

We use feature scaling to make sure that some features won't dominate the result when adding them together.

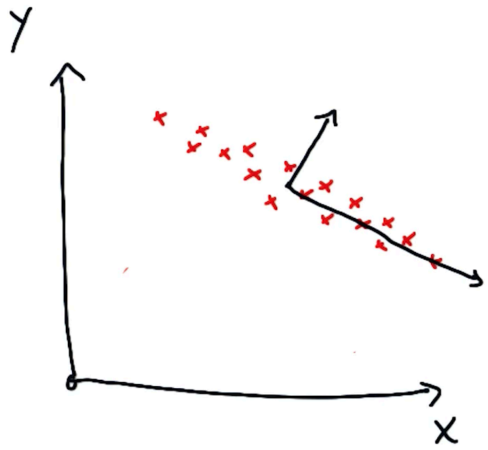
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Principal Component Analysis (PCA)

PCA is used for dimensionality reduction. It finds a new coordinate by translation and rotation.

- It moves the center of the coordinate system to the center of the data.
- It moves the x-axis into th_{}e principal axis of variation (the axis where we see most variation)
- Further axis become orthogonal axis of variation

Basically, PCA is driven by the assumption that there is a small number of features in our dataset that actually drives the patterns. Out of these feature we then try to form *composite features* that more directly probe the underlying phenomenon.



How to find the principal component?

We choose the principal components to be the direction that has the *largest variance* since it retains the maximum amount of information in the original data.
This is simply because it minimizes the information loss. We can see the *information loss* as the *distance between a feature and the hyperplane*.

Latent variables

Latten (hidden) variables are cannot be measured directly, but the are somehow driving the phenomenon behind the scenes. For instance, the size of an house might be a good indicator for its price. However, the size cannot be measured directly. Instead, it's a combination of square footage and the number of rooms.

When should we use PCA?

- Latent features driving the patterns in data
- Dimensionality reduction
    - Visualize high-dimensional data
    - Reduce noise (by throwing away the less important principal components)
    - Make other algorithms (regression, classification) work better for fewer inputs

Random projection

Random projection is another dimensionality reduction method that is computationally more effective than PCA. While PCA always looks for a hyperplane that maximizes the variance of the projection, random projection simply chooses a random hyperplane and projects the data on it.

This can expresses as a simple matrix multiplication of the form:

The following equation reduces the data from d dimensions to k dimensions.

$X_{kxN}^{RP} = R_{kxd} \cdot X_{dxN}$ where R is some random matrix representing a set of hyperplanes.

Johnson-Lindenstrauss Lemma

The reason why dimensionality reduction works with random hyperplanes is the so-called *Johnson-Lindenstrauss lemma*.

A dataset of N points in high-dimensional euclidean space can be mapped down to a space in much lower dimension in a way that preserves the distance between the points to a large degree.

The following property gets preserved:

$$(1-\epsilon)||u-v||^2 < ||p(u)-p(v)||^2 < (1+\epsilon)||u-v||^2$$

ϵ ... Accepted level of error. This is a parameter we are allowed to choose.
u,v ... Points in the dataset
p(u), p(v) ... Points on the projected hyperplane

Independent Component Analysis

ICA takes a mixture of independent sources and tries to isolate these independent sources.

It assumes that components are statistically independent and they also must **not** have non-gaussian distribution.

Note: The central limit tells us that the sum of independent variables tends towards a gaussian distribution.

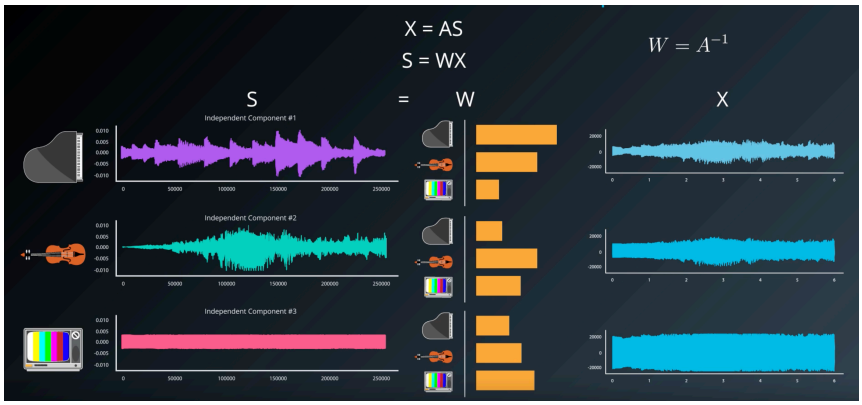
To extract the independent components we assume that every mixed signal is generated by multiplying the individual component S by a mixing matrix A.

So, S can be expressed by the equation:

$$S = AX$$

If we now assume that A is invertible, we can simply obtain the original signal by calculating:

$$X = A^{-1}S$$



Applications

ICA is widely used in medical applications, for instance, EEG (separate the different signals obtained from multiple sensor readings) or MRI (locate the region of a signal).