Introduction to machine-learning using scikit-learn

QLSC612

May 2025

By

Mohammad Torabi & Michelle Wang

McGill University

ORIGAMI Lab

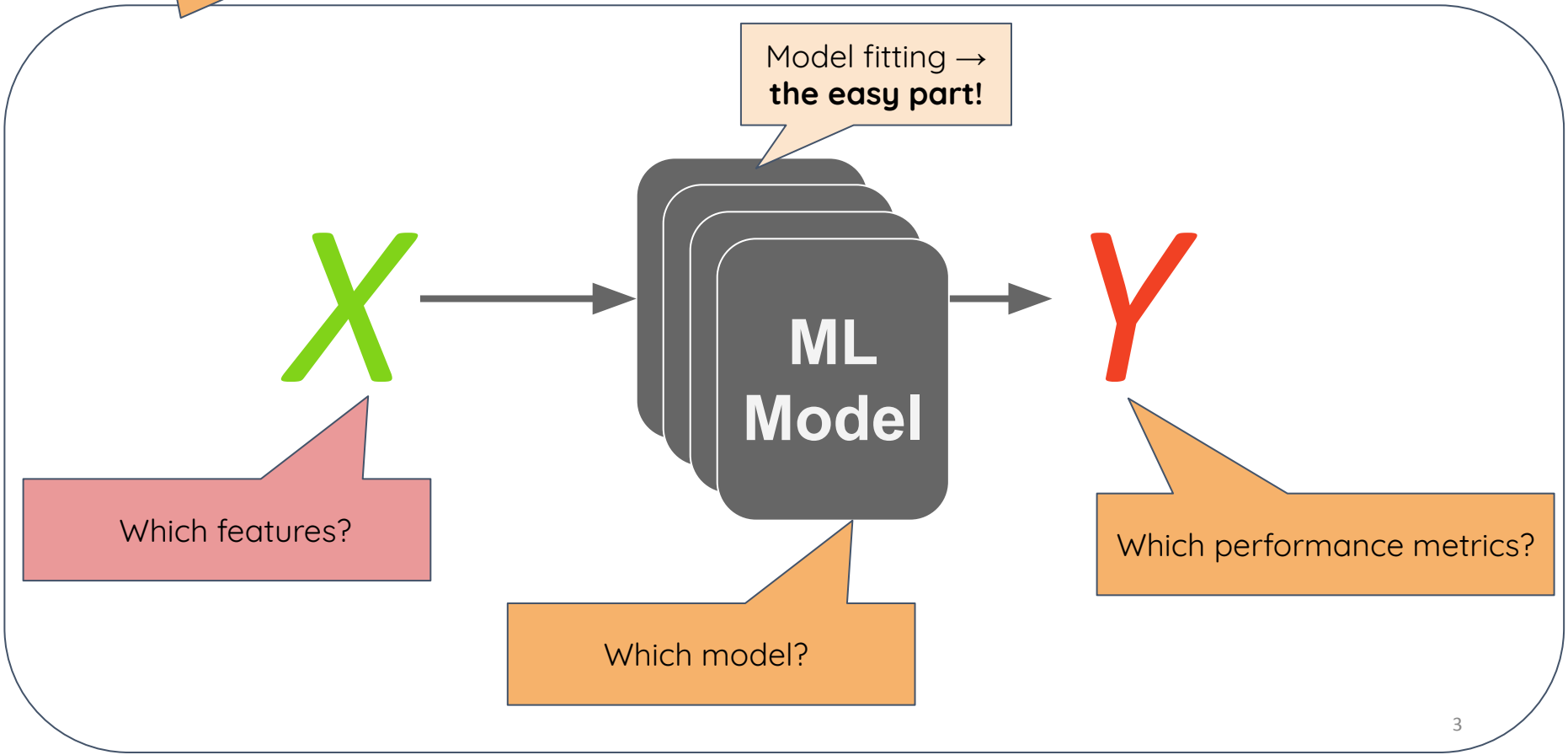
neuro

Montreal Neurological Institute-Hospital

Objectives

- Part 2: Unsupervised learning
  - Dimensionality reduction
  - Clustering
  - *Coding example: PCA, k-means*
  - *Coding example: fMRI site prediction*

How do I validate my design choices?

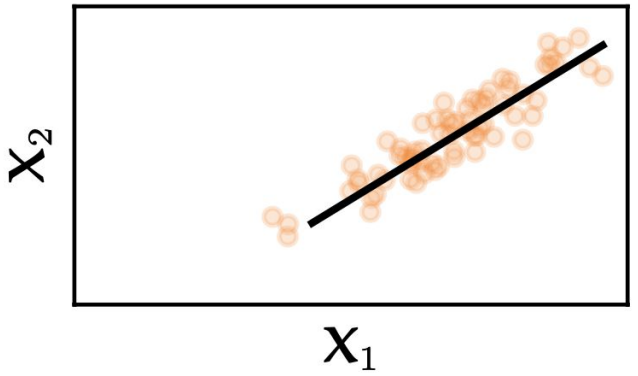


Unsupervised learning

- Learning without knowing the true labels
- Objectives
  - Dimensionality reduction of features
  - Grouping of samples based on **“similarity”**
- Techniques
  - Feature Transformation/Projection
  - Clustering

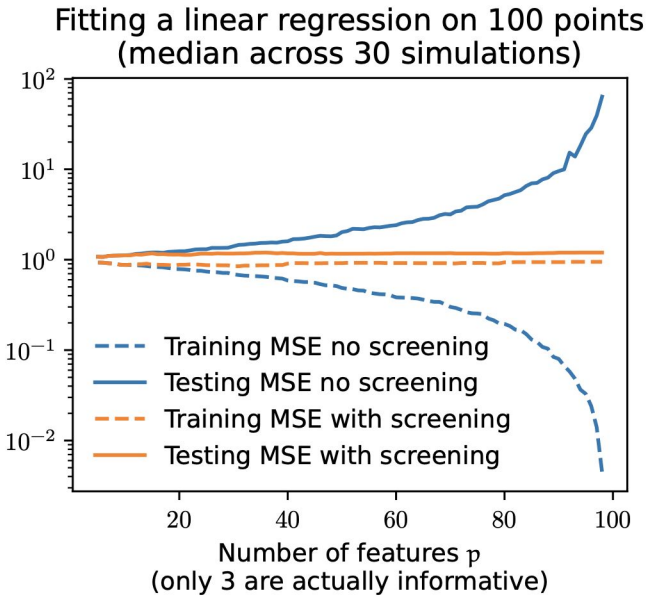
Dimensionality Reduction

Data is almost 1-dimensional
BUT represented as
2-dimensional



Dimensionality Reduction

- High dimensional data:
  - Model complexity increases -> unstable solution
  - Risk of overfitting: fitting exactly training data but failing on test data

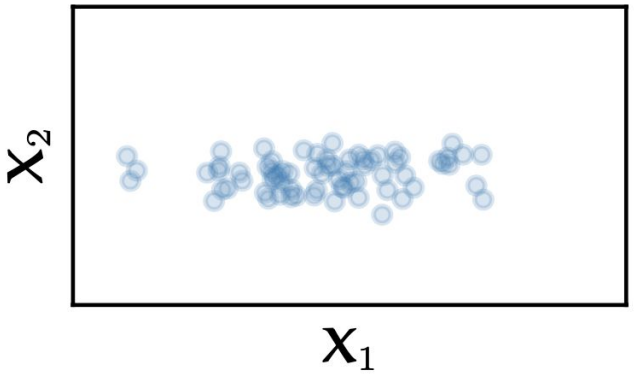


Dimensionality Reduction

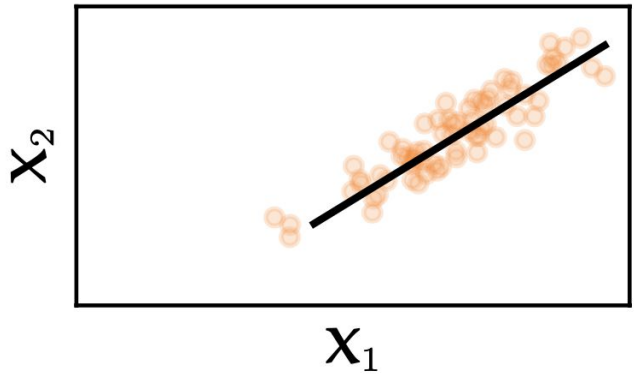
- **Curse of Dimensionality:** large number of input features can dramatically impact the performance of ML algorithms
- Techniques:
  - Feature selection (usually supervised)
  - Feature transformation (usually unsupervised)
- Feature transformation is useful for
  - Information compression
  - Data artifact clean-up
  - Visualization

Feature Transformation vs. Feature Selection

Maybe OK to drop X2

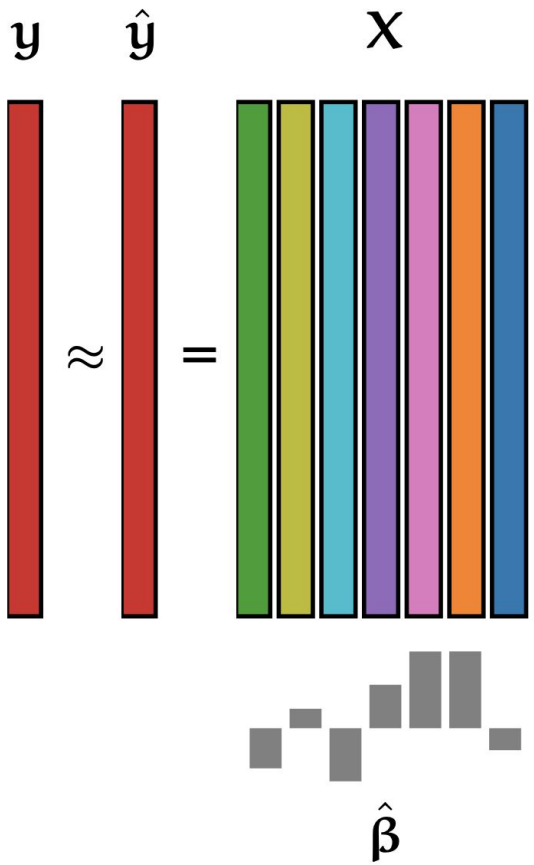


Data is low-dimensional BUT no feature can be dropped



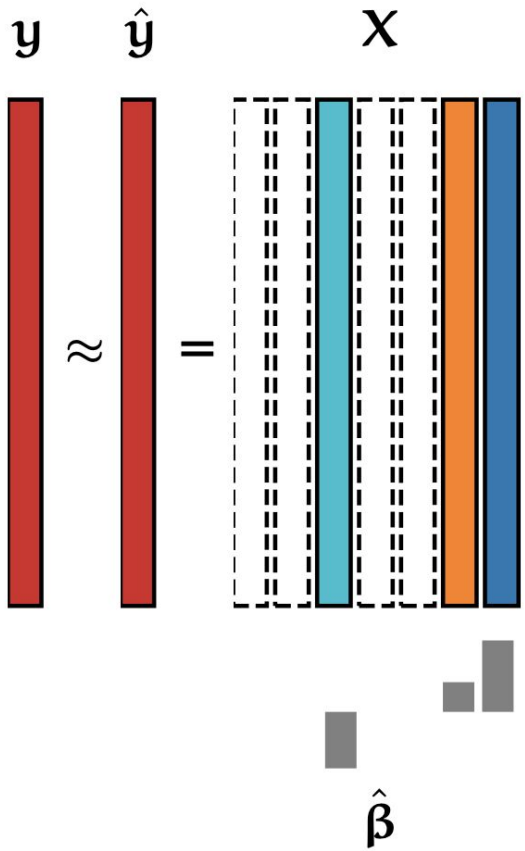
Feature Selection

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$

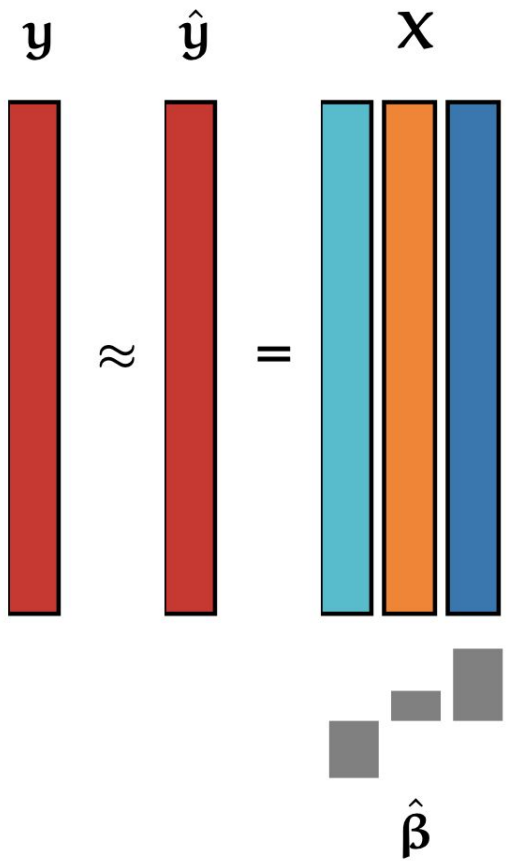


Feature Selection

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$

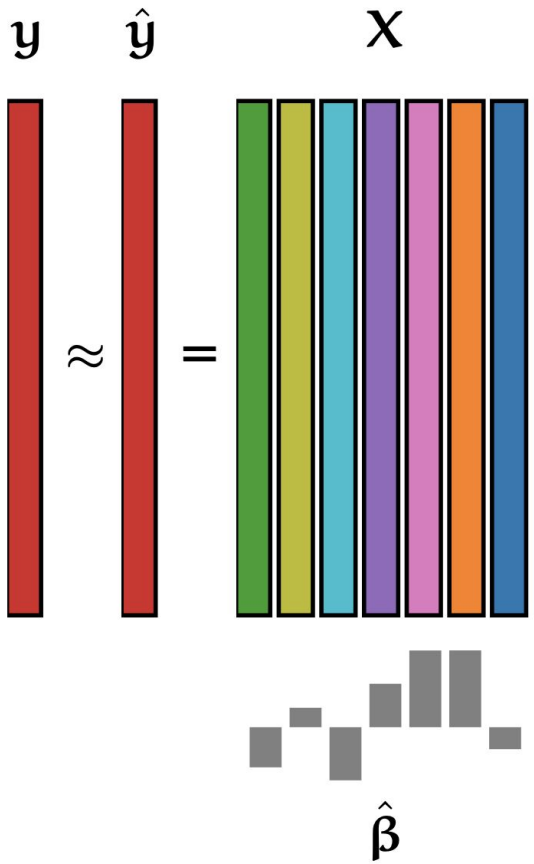


Feature Selection



Feature Transformation

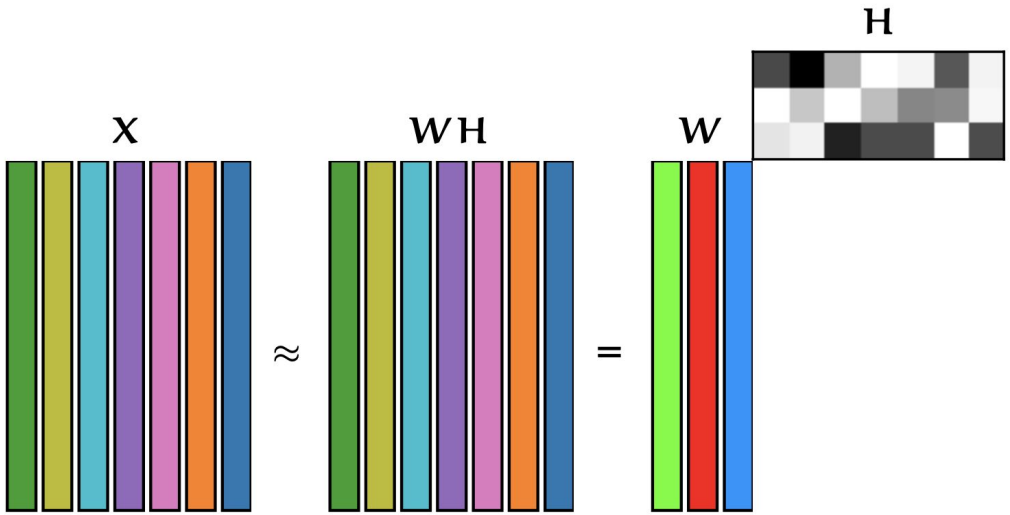
$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$



Feature Transformation

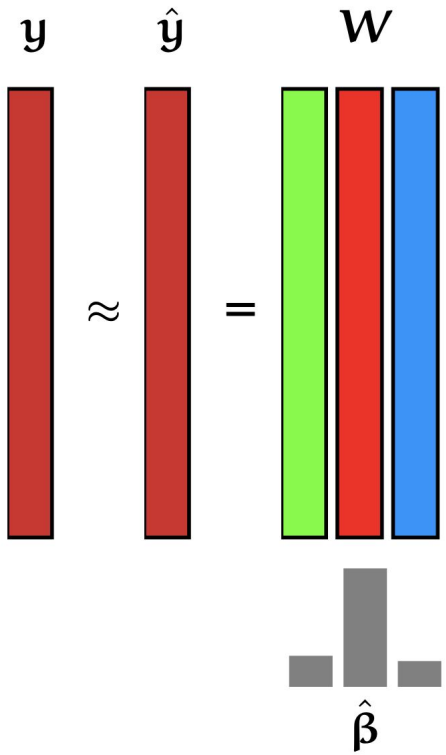
Minimize

$$\|\mathbf{X} - \boldsymbol{W}\mathbf{H}\|_F^2 = \sum_{i,j} (\mathbf{X}_{i,j} - (\boldsymbol{W}\mathbf{H})_{i,j})^2$$



Feature Transformation

$$\hat{\mathbf{y}} = \boldsymbol{W}\hat{\boldsymbol{\beta}}$$

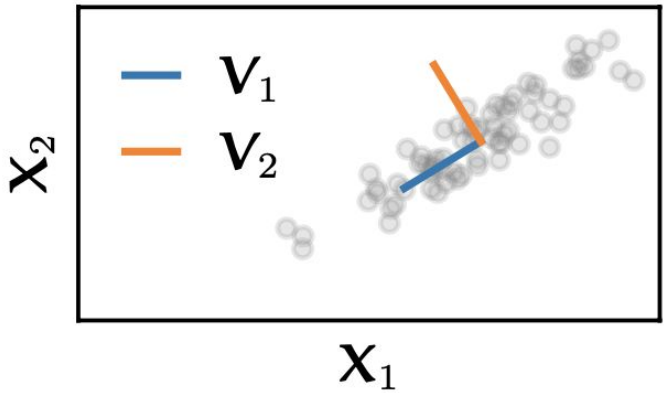


Feature Transformation

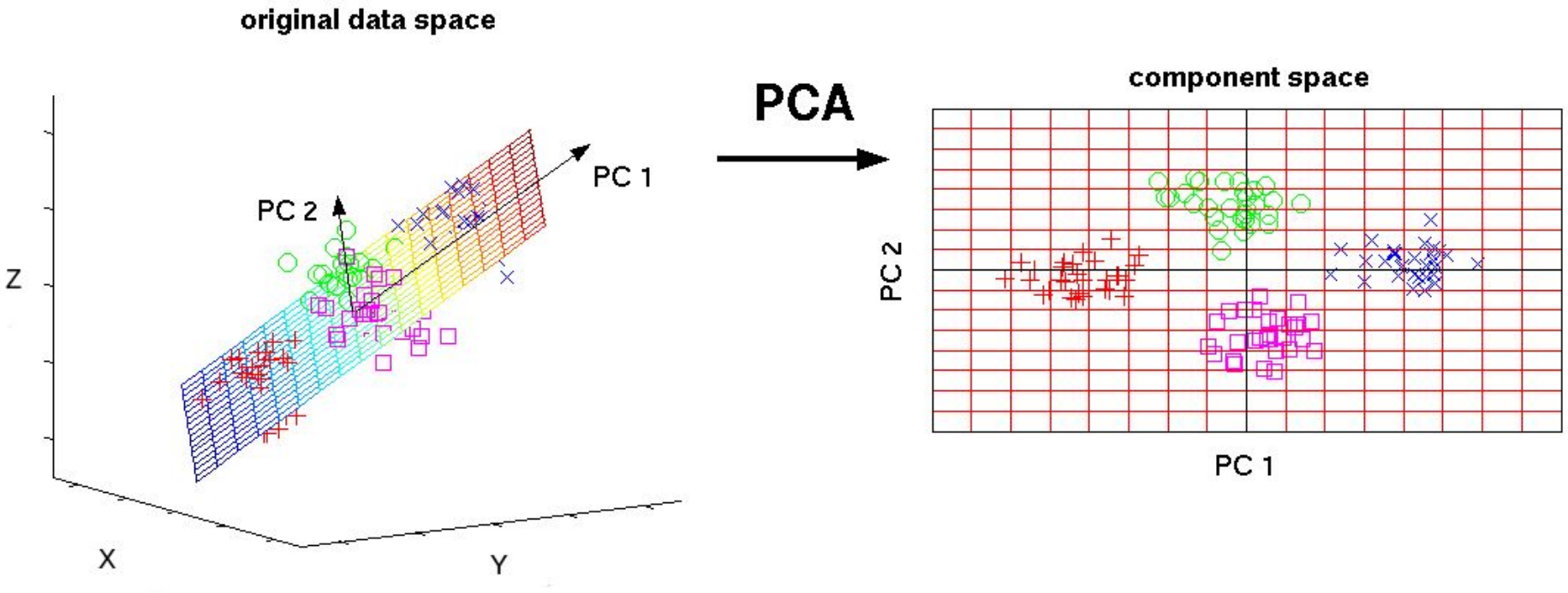
- Change of basis (i.e. “perspective”) for data representation
- Techniques with different priors on data generation process:
  - Singular Value Decomposition (SVD)
  - Principal Component Analysis (PCA)
  - Independent Component Analysis (ICA)
  - Non-negative Matrix Factorization (NMF)

Principal Component Analysis (PCA)

- PCA finds a set of components (eigenvectors) that are orthogonal and capture the maximum variance in data.
- These components form the basis of the new space that the data will be transformed to
- Truncating the components to keep only the first k components gives the best rank-k approximation of X and transforms X to k-dim space

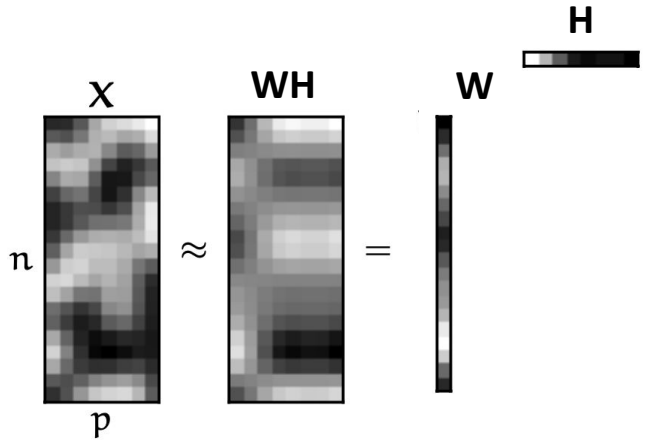


Principal Component Analysis (PCA)



Principal Component Analysis (PCA)

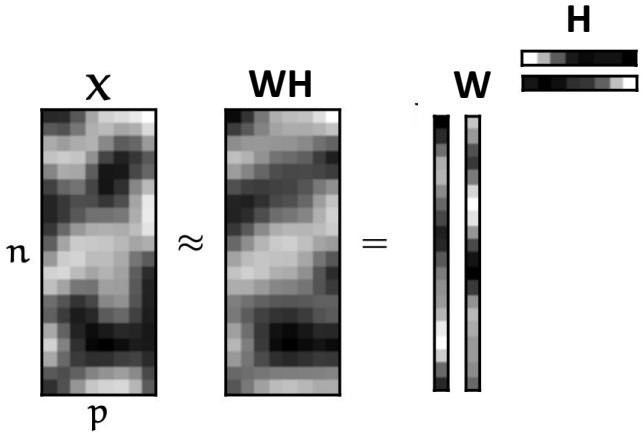
Reconstructing with 1 principal component:



Explained variance: 0.53

Principal Component Analysis (PCA)

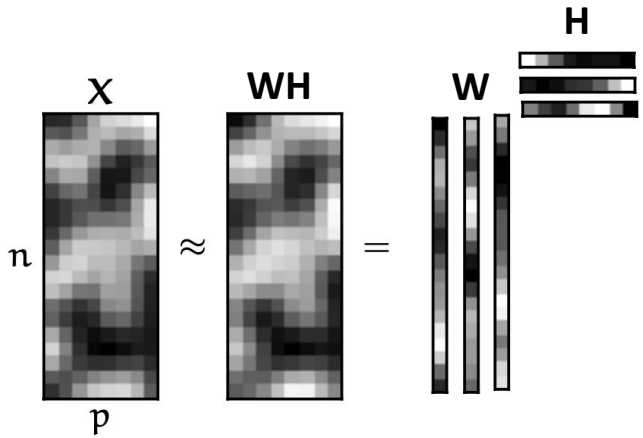
Reconstructing with 2 principal components:



Explained variance: 0.84

Principal Component Analysis (PCA)

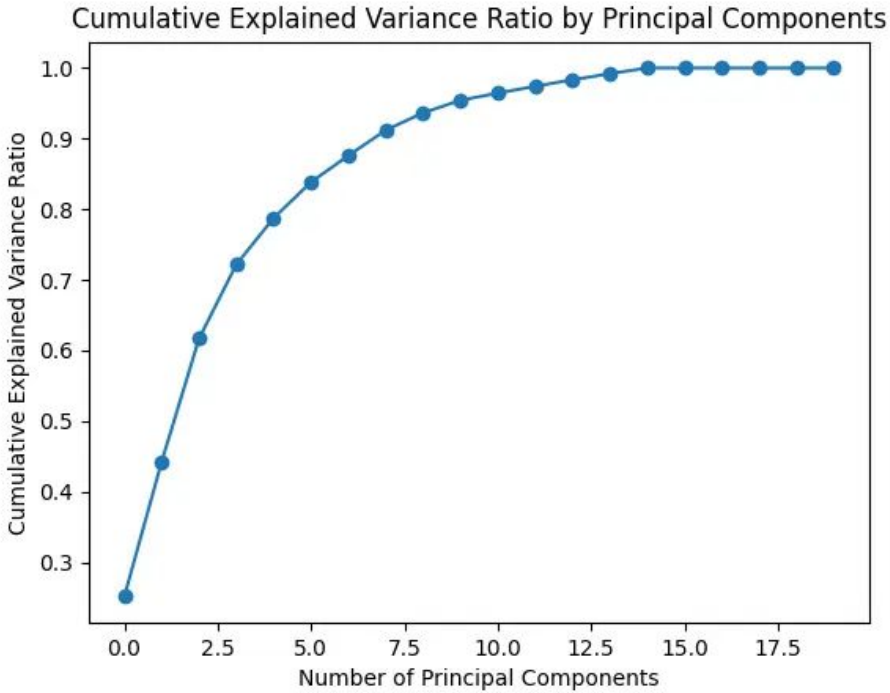
Reconstructing with 3 principal components:



Explained variance: 0.97

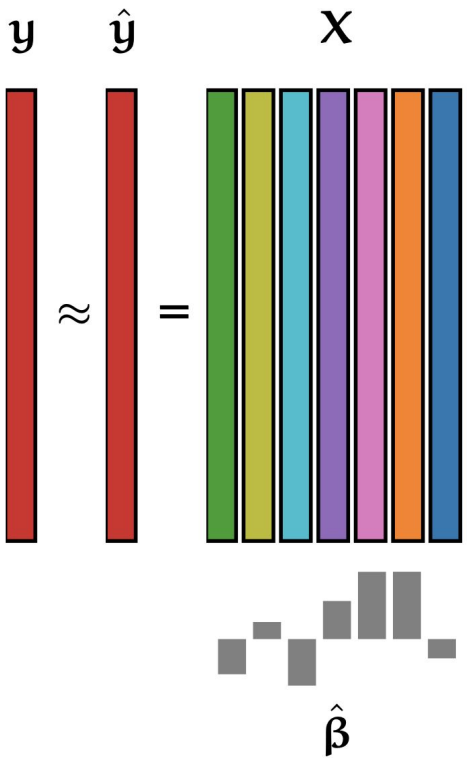
Principal Component Analysis

- Variance explained

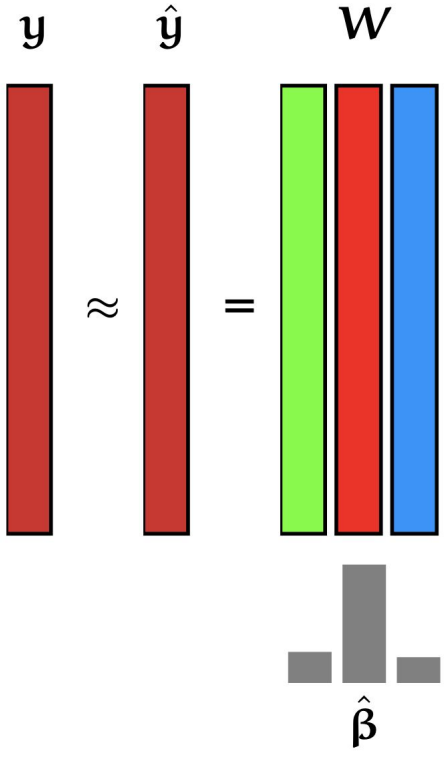


Principal Component Analysis (PCA)

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$



$$\hat{\mathbf{y}} = \boldsymbol{W}\hat{\boldsymbol{\beta}}$$



Principal Component Analysis

- sklearn

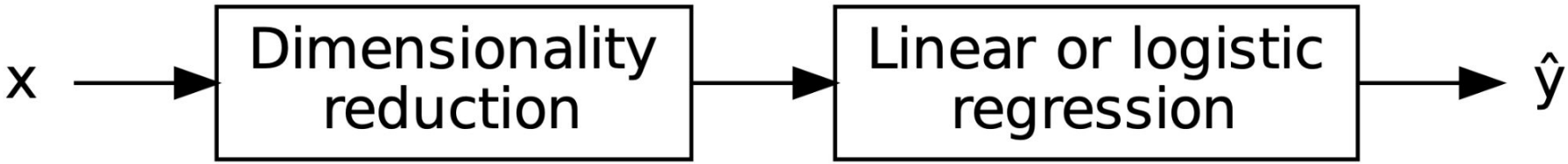
```
transformer = PCA(n_components=N)
transformer.fit(X)
transformed_X = transformer.transform(X)
print(transformed_X.shape) #(n_samples, n_components)
```

```
transformer = PCA(n_components=N)
transformed_X = transformer.fit_transform(X)
```

sklearn.decomposition.PCA

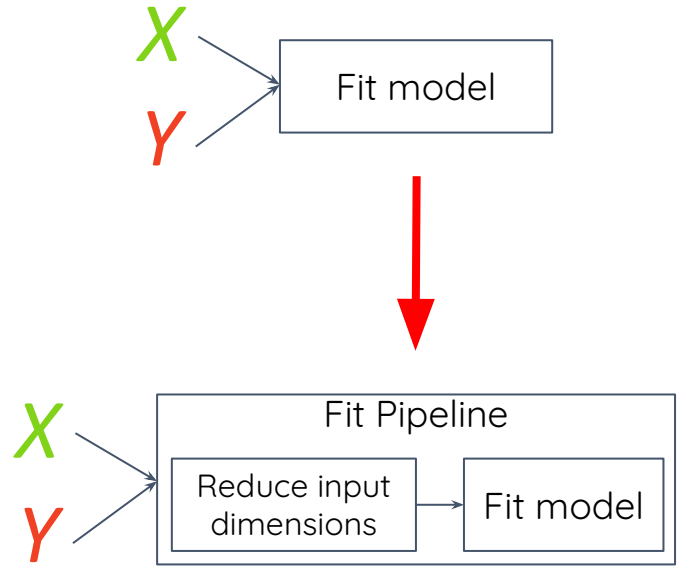
Dimensionality Reduction

- A preprocessing step



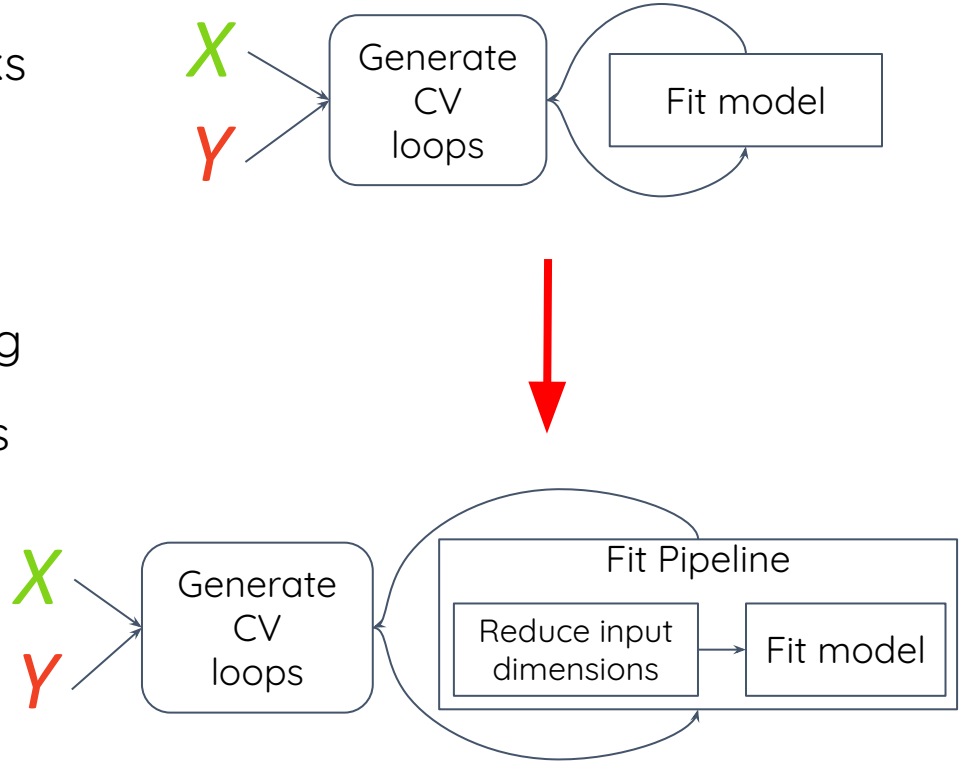
Pipelines

- Chain various “preprocessing” tasks in your analysis
  - Feature scaling
  - Dimensionality reduction
- Avoids mistakes e.g. double dipping
- Simplifies changes to your analysis



Pipelines

- Chain various “preprocessing” tasks in your analysis
  - Feature scaling
  - Dimensionality reduction
- Avoids mistakes e.g. double dipping
- Simplifies changes to your analysis
- Simplifies cross-validation



Pipelines

- Remember! Feature transformation only on the training data!

```
transformer = PCA(n_components=N)
transformed_train = transformer.fit_transform(X_train)
Transformed_test = transformer.transform(X_test)
```

sklearn.pipeline.make_pipeline

Pipelines

```
pipeline = make_pipeline(PCA(n_components=N), LinearRegression())
pipeline.fit(X_train, y_train)
Y_train_predicted = pipeline.predict(X_train)
y_test_predicted = pipeline.predict(X_test)
```

sklearn.pipeline.make_pipeline

Clustering

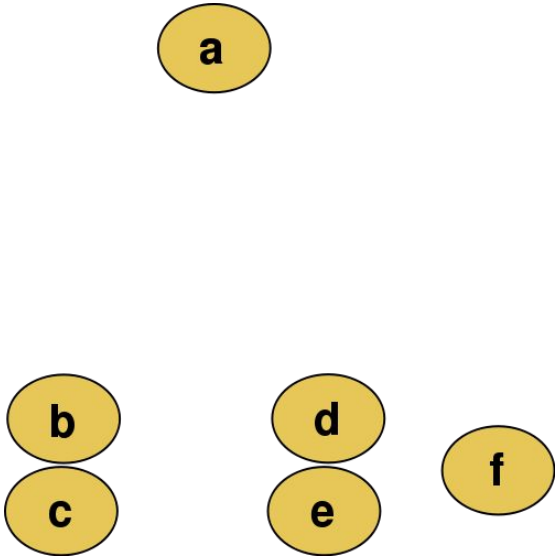
- Grouping observations together without knowing their true labels
- Using distance (i.e. similarity) between samples (often in a high-dimensional space!)
- Things to consider
  - Hyperparameters (e.g. n_clusters, distance metric)
  - Scalability

Clustering

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers

Clustering

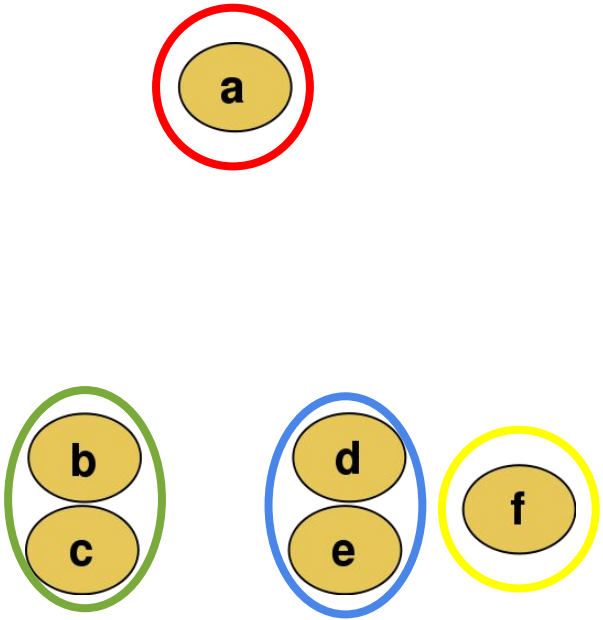
- Which samples will cluster together?



Sample space

Clustering

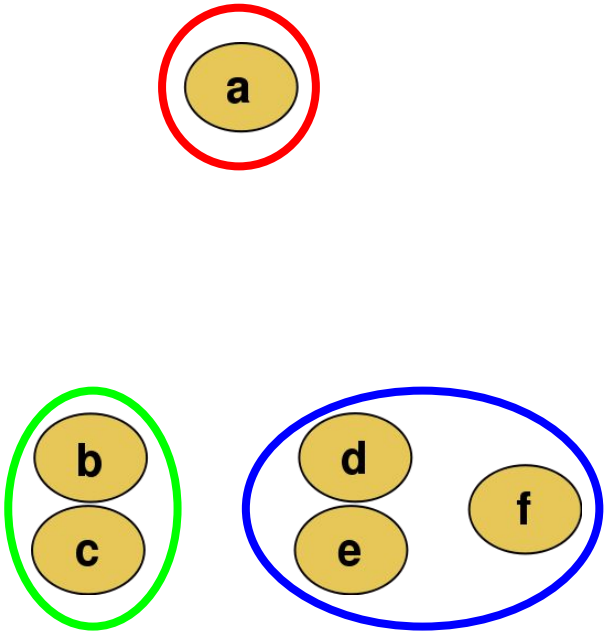
- Which samples will cluster together if n_clusters=4?



Sample space

Clustering

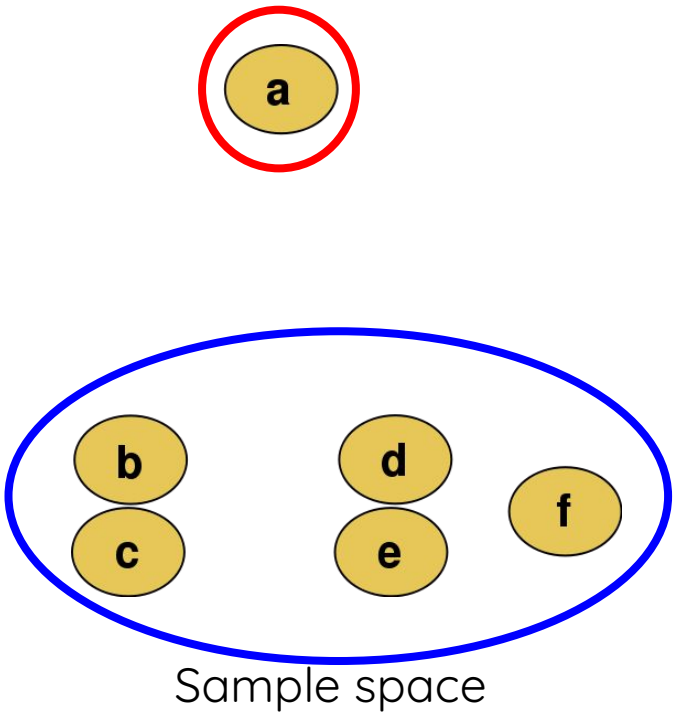
- Which samples will cluster together if n_clusters=3?



Sample space

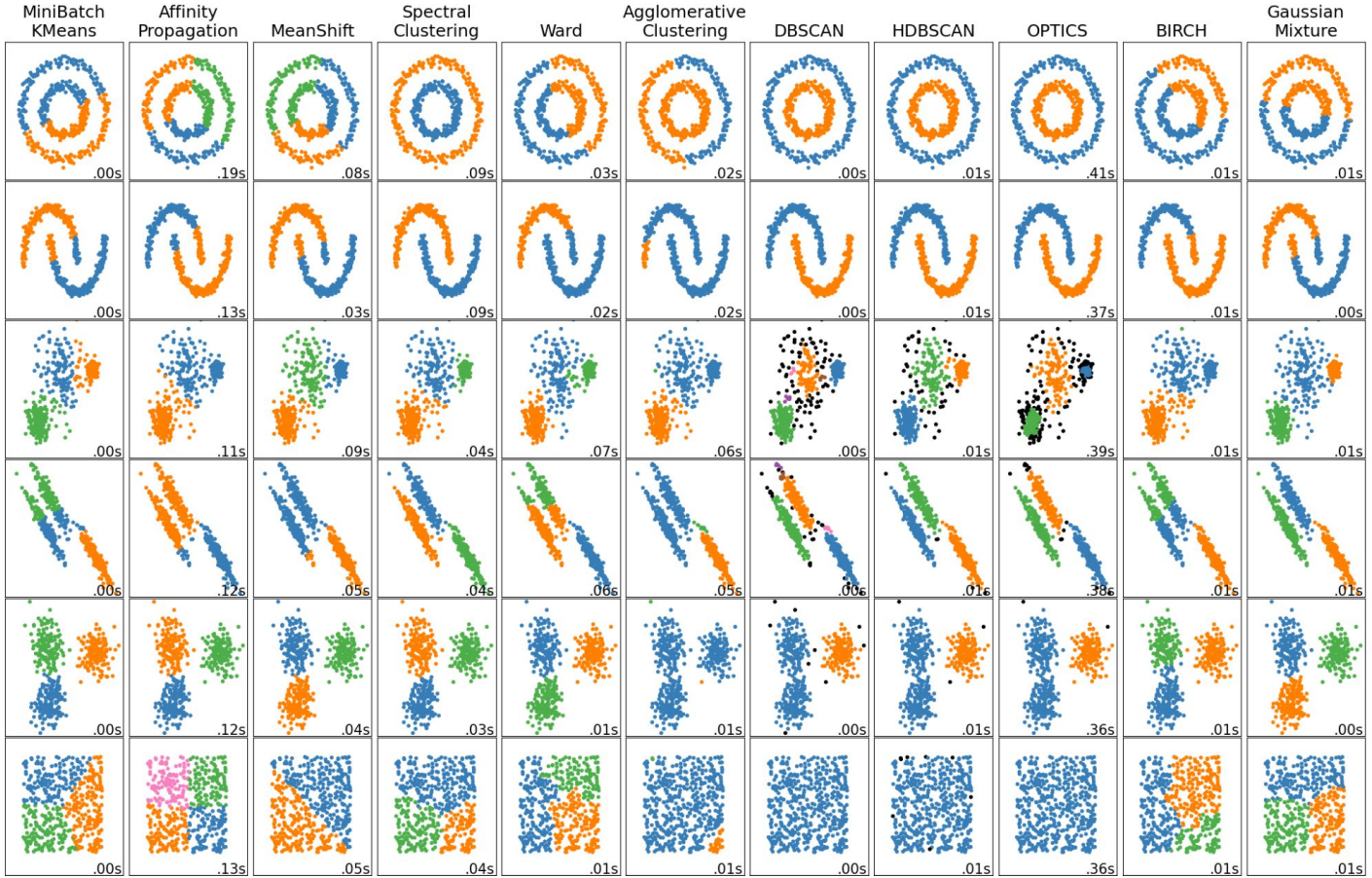
Clustering

- Which samples will cluster together if n_clusters=2?



Clustering

... is in the eyes of the beholder



K-means Clustering

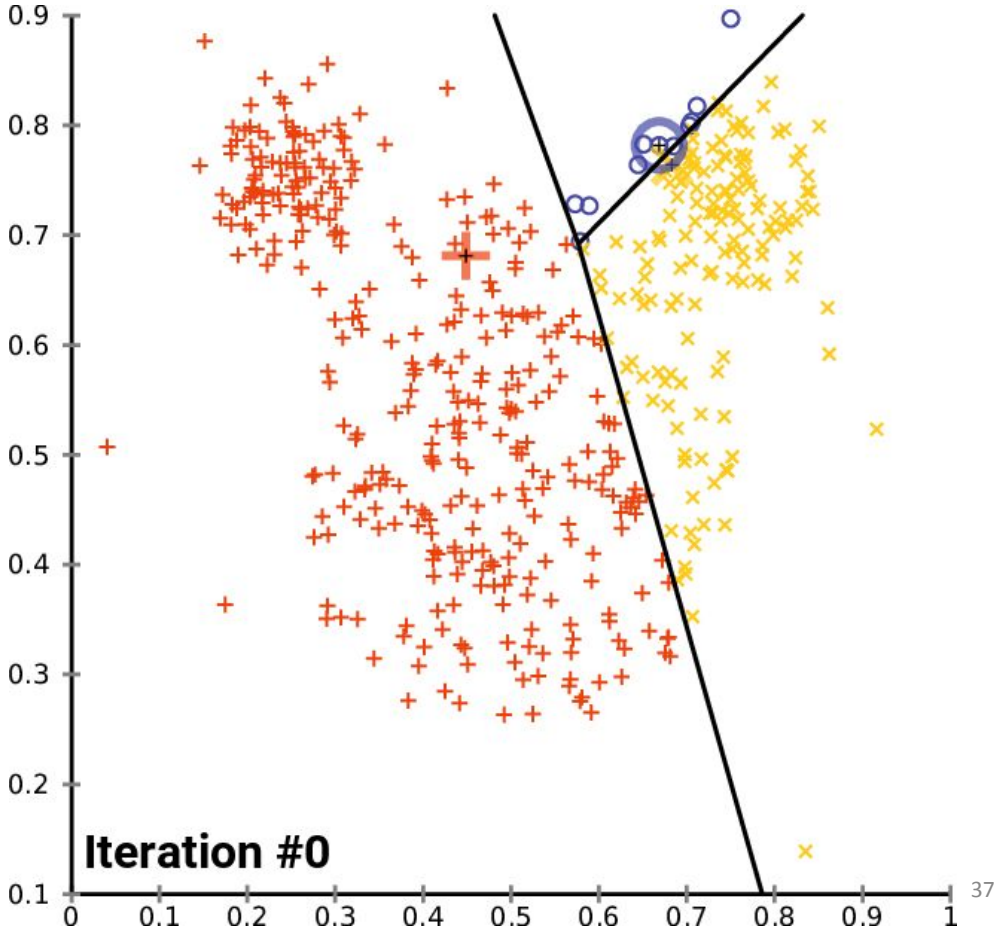
- The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares/distance
- Each centroid represents a cluster
- This algorithm requires the number of clusters to be specified
- Often uses Euclidean distance
- Very high-dimensional spaces result in inflated Euclidean distances (an instance of curse of dimensionality!)
    - Run a dimensionality reduction algorithm (e.g. PCA) prior to k-means clustering

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

K-means Clustering

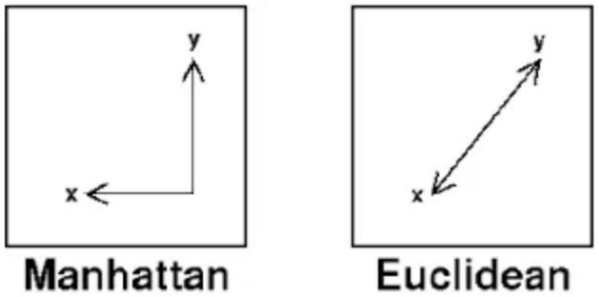
Steps:

1. **Initialization:** Choose k random centroids
2. **Assignment step:** Assign each observation to the cluster with the nearest centroid
3. **Update step:** Recalculate centroids (means) for observations assigned to each cluster



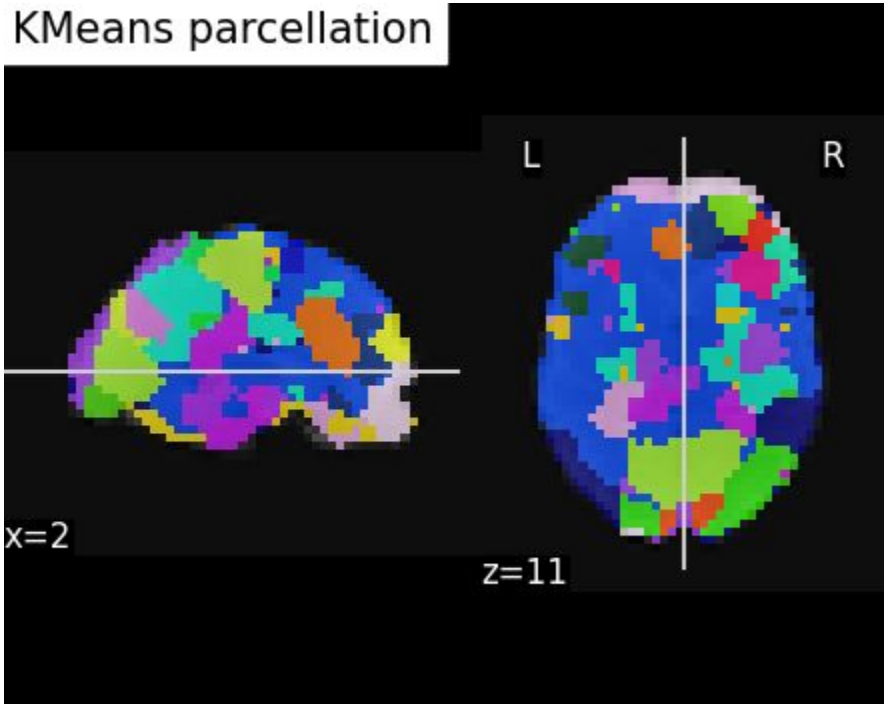
Distance Metrics

- Euclidean distance
- Manhattan distance
- Hamming distance
- Correlation (1 - corr)



Biological Example

- Parcellate brain voxels based on their fMRI signals
- How do we know if the clustering is done well? What metrics do we have?



Evaluation of clusters

- Internal validation (without true labels):
  - Silhouette Coefficient
- external validation (with true labels):
  - Rand Index (RI) and Adjusted Rand Index (ARI)
  - Mutual Information

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

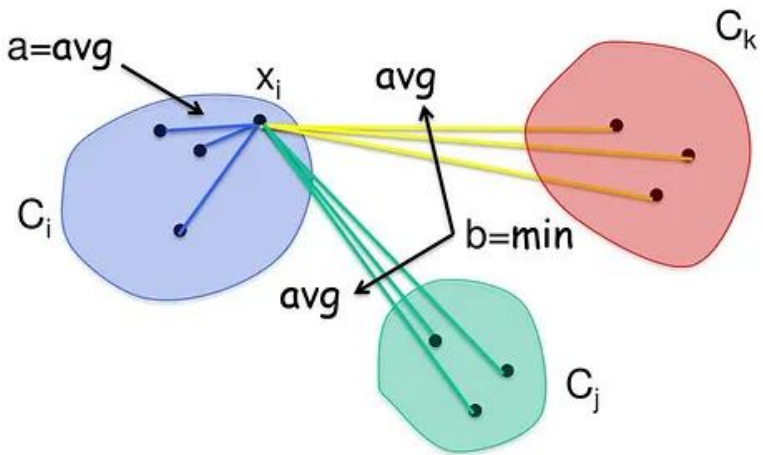
$$\mathrm{RI} = \frac{a + b}{C_2^{n_{samples}}}$$

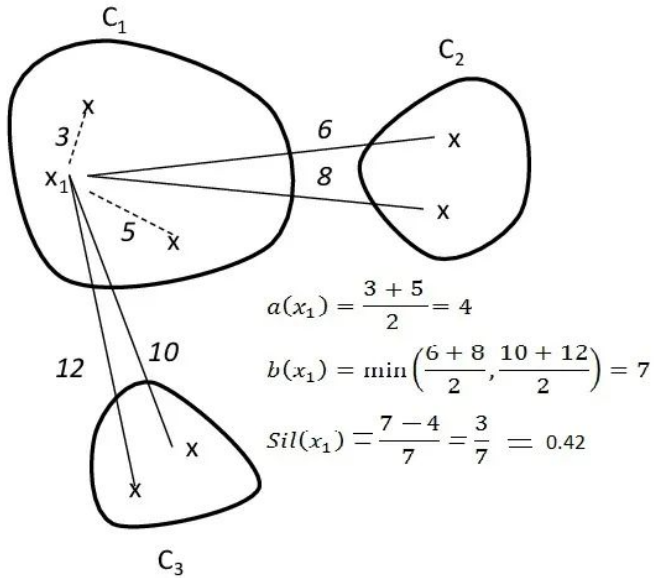
$$\mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]}$$

Evaluation of clusters

- Internal validation (without true labels):
  - Silhouette Coefficient

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





[Silhouette Index]

Project

- **Data:** X: connectivity features derived from fMRI data, y: fMRI site label for each participant.
- **Visualize the data:** you have to apply dimensionality reduction first
- **Classification (supervised learning):** Use different models to classify and predict the fMRI site using the connectivity features and compare their performance. We will use scikit-learn pipeline for this purpose, which chains different transformations. Then we will fit the whole pipeline on our data using cross-validation.
- **Clustering (unsupervised learning):** use K-means clustering to cluster the participants. Find the best number of clusters and evaluate the clustering performance.