Introduction to PCA, k-means clustering, and k nearest neighbors classification

Yulia Newton, Ph.D.

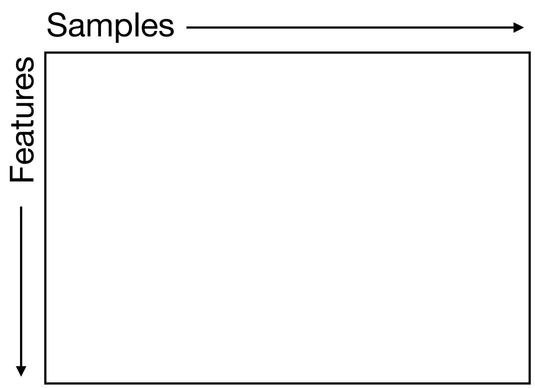
CS156, Introduction to Artificial Intelligence

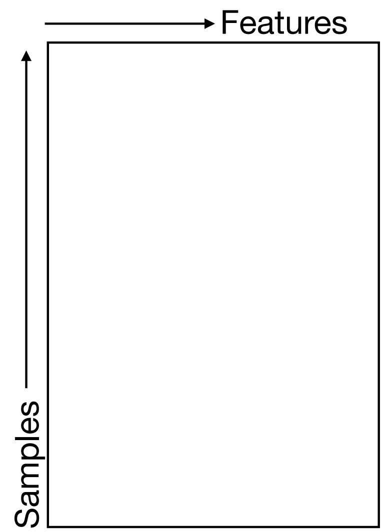
San Jose State University

Spring 2021

Data dimensionality

- "Dimensionality" refers to the number of features each training/ validation/testing observation has





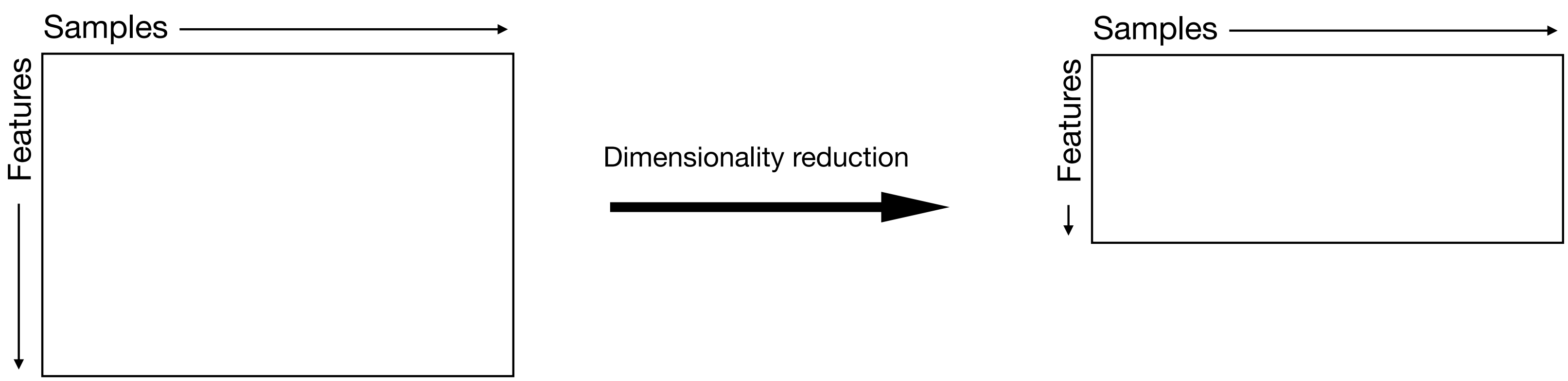
Before you jump into building predictive models

- Not all of the features in your dataset might be informative or relevant to the prediction task you are solving
  - Irrelevant information adds unnecessary noise
  - Reducing dimensionality can help improve model performance
  - Noise can cause model overfitting
- The curse of dimensionality
  - Models built on high-dimensional data require a lot of observations in order to avoid overfitting
  - Reducing dimensionality can help avoid model overfitting

- Remember: “garbage in - garbage out”

Dimensionality reduction

- "Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension." - Wikipedia



Feature space reduction

- Dimensionality reduction is also known as a feature space reduction

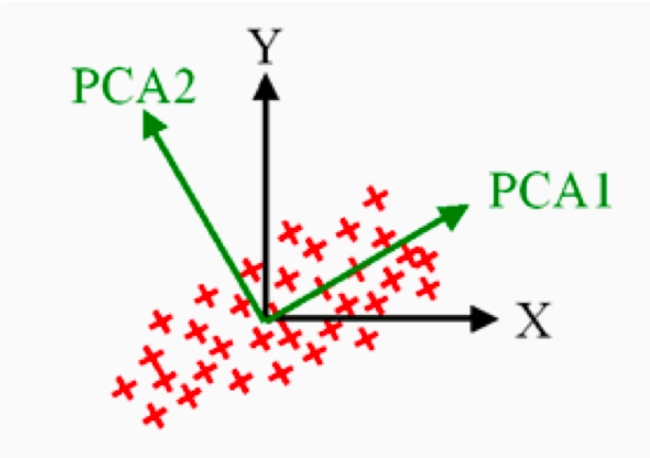
- Can be achieved through
  - Reducing the number of input/independent variables
  - Transformation/projection/rotation of the data from high-dimensional feature space into low-dimensional feature space

Dimensionality reduction during data exploration stage

- Projecting data into 2-D or 3-D space prior to building a predictive model can show patterns intrinsic to the data
- Clustering the data prior to building a predictive model can tell us if the chosen feature space is predictive of the output variable

Principal component analysis

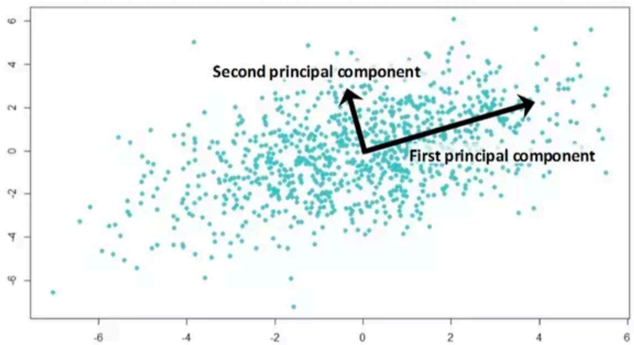
- Principal component analysis (PCA) is one of the most commonly used methods to perform data projection and dimensionality reduction
- Core idea - rotate axes in the direction of the biggest variance



https://amva4newphysics.wordpress.com

PCA steps (conceptual understanding)

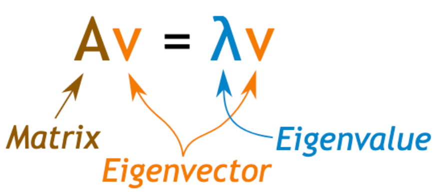
- Find the direction in the data with the highest variance in the hyperspace (feature space) - that is the first principal component
- Find the direction in the data with the next highest variance, which is orthogonal to previously identified principal components
- Continue until the number of desired principal components is reached



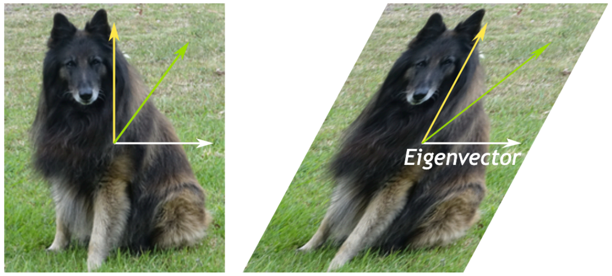
https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python

What are principal components?

- Principal components are eigenvectors of the covariance matrix
  - From linear algebra, an eigenvector is a vector that changes only by a scalar under a linear transformation
    - Does not change direction in a transformation



www.mathsisfun.com



www.mathsisfun.com

Mathematics behind PCA

- Compute covariance matrix of the input data
  - $X_1, X_2, \ldots, X_n$ are random variables

$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$$

- Covariance matrix is a matrix of covariances between random variables *i* and *j* in each *(i, j)* entry of the matrix
  - On the diagonal, it's just variance of the random variable
  - According to the law of large numbers, the average value of a random variable converges to expected value

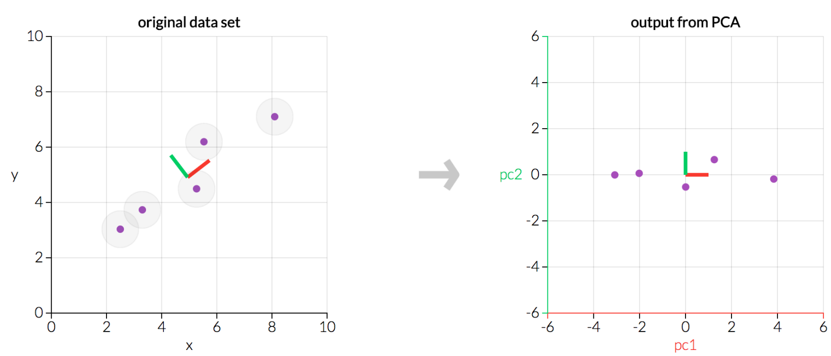
$$\mathrm{K}_{X_i X_j} = \operatorname{cov}[X_i, X_j] = \mathrm{E}[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])]$$

$$\mathrm{K}_{\mathbf{X}\mathbf{X}} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_n - \mathrm{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_n - \mathrm{E}[X_n])] \end{bmatrix}$$

en.wikipedia.org

Principal components are not the same as the original features

- Once the data has been rotated into the principal component space, original features no longer can be assigned to the PC axes
  - Principal components are linear combinations of original features
  - Different original features might contribute to principal components with different influence
  - Principal component space is a new transformed space



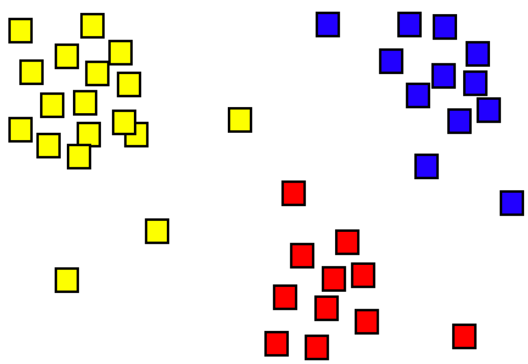
https://setosa.io/ev/principal-component-analysis

Let’s play with some online examples

- https://setosa.io/ev/principal-component-analysis/

Clustering

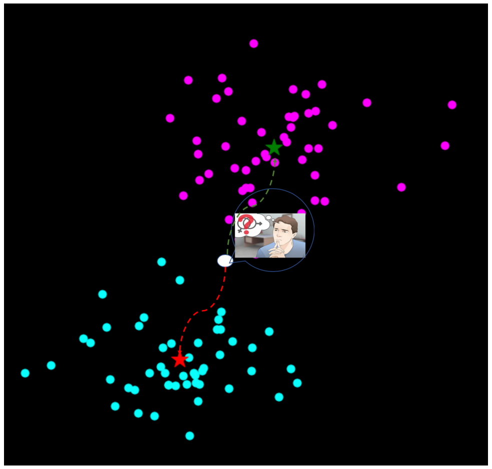
- “Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.” - Wikipedia
- Clustering assigns labels based on the input variable values



en.wikipedia.org

k-mean clustering

- This method aims to partition data into k parts/clusters
- The idea is to represent clusters by centroids and assign a data point to a cluster based on which centroid it is the closest to/most similar to



https://towardsdatascience.com

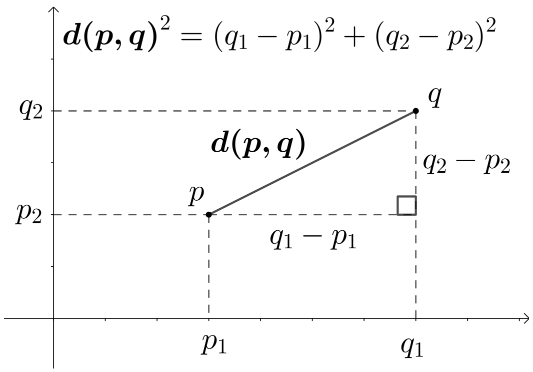
k-mean clustering (cont'd)

- Steps:
  - Select k parameter
  - Select k centroids by selecting random data points from the dataset without replacement
    - Other ways to generate initial centroids can be utilized
  - Iterate until convergence (no significant change in centroids and/or assignment of data points to the clusters does not change or a predetermined number of iterations):
    - For each data point compute distance or similarity between the data point and each centroid
    - Assign each data point to 1 out of k clusters based on the smallest distance/largest similarity
    - Recompute the centroid for each cluster 1…k based on the data points in that cluster

Distance/similarity measure

- Depending on the data/task different measures of distance/similarity might be more appropriate
  - Distance similarity can be measured in low- or high-dimensional space
- Most often used measures of distance:
  - Euclidean distance
  - Hamming distance
  - Manhattan distance
  - Minkowski distance
- Most often used measures of similarity:
  - Correlation coefficient
  - Cosine similarity
  - Jaccard similarity

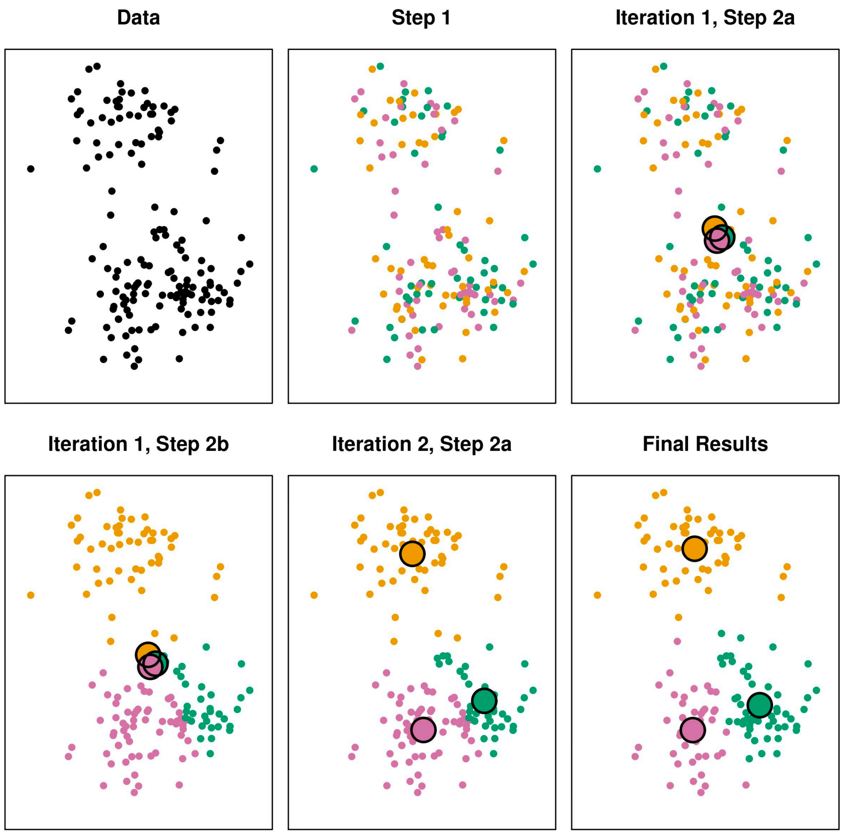
Euclidean distance in 2-D space:



en.wikipedia.org

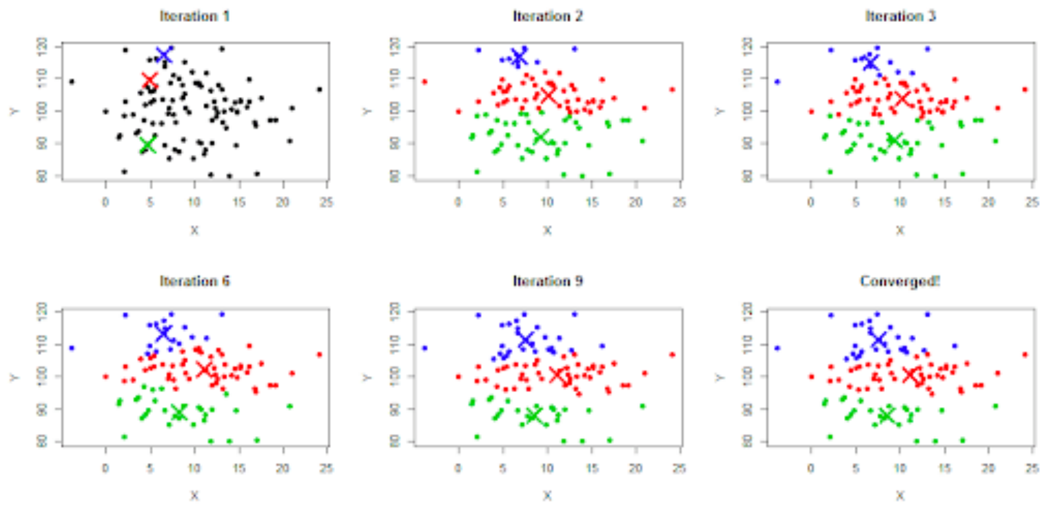
- Useful explanation of some of these measures:
  - https://medium.com/@gshriya195/top-5-distance-similarity-measures-implementation-in-machine-learning-1f68b9ecb0a3

Demonstration of k-means clustering steps (toy example 1)



https://stackoverflow.com/questions/51263331/kmeans-save-each-iteration-step

Demonstration of k-means clustering steps (toy example 2)



https://stackoverflow.com/questions/60312401/when-using-the-k-means-clustering-algorithm-is-it-possible-to-have-a-set-of-dat

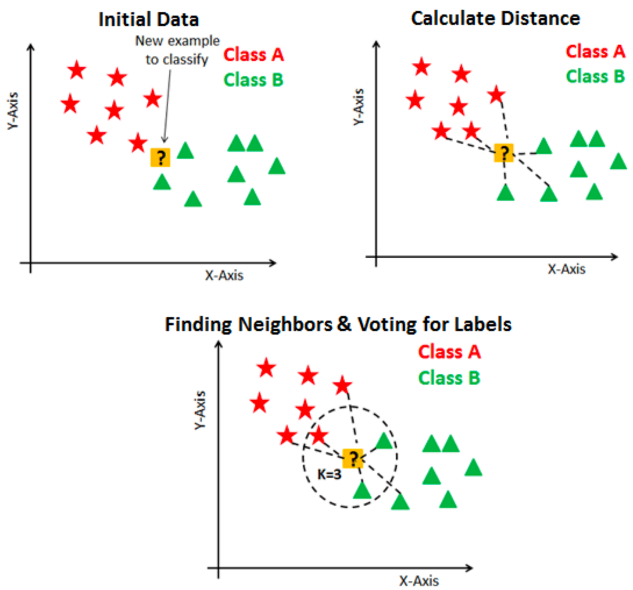
k-nearest neighbors classification

- k-nearest neighbors algorithm is a ML classification method for classifying observations into classes/labels
  - The output is a class membership
- No generalized model
  - Based on training data and labels
  - No explicit model training step
- An instance-based learning type of algorithm
  - Also called lazy learning
- Related to the k-means clustering algorithm

k-nearest neighbors method

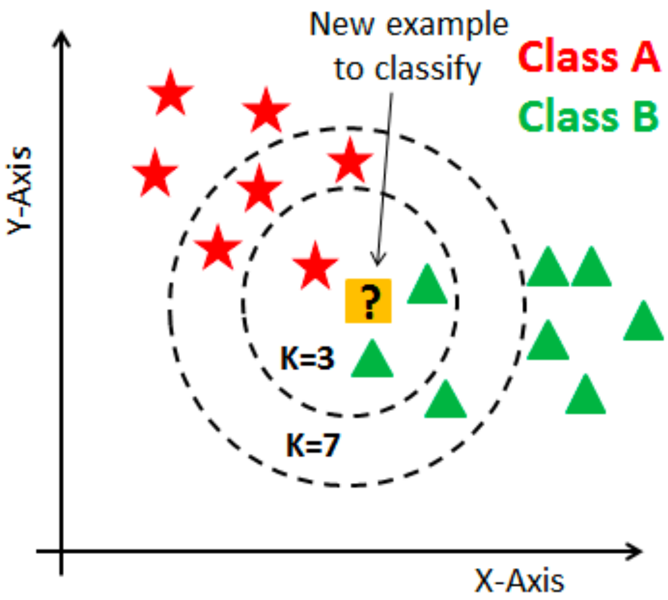
- Steps for classifying a new data point:
  - Select k parameter
  - Compute the distance/similarity between every data point in the training data and the new data point
  - Identify k closest (most similar) data points
  - Assign the label of the new data point based on majority vote
    - Different ways of assigning the class can be utilized
      - E.g. weighted voting, etc.

k-nearest neighbors toy example



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

The choice of k hyper-parameter might dramatically affect the results



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Let’s look at some code examples

- General dimensionality reduction examples on Iris dataset
  - scikitlearn libraries
    - PCA
    - LDA
    - t-SNE
  - *Dimensionality_reduction.ipynb*
- k-means clustering of synthetic 2-D data
  - *kmeans.synthetic_data.ipynb*
- k-nearest neighbors classification
  - *knn.synthetic_data.ipynb*