Topic 5

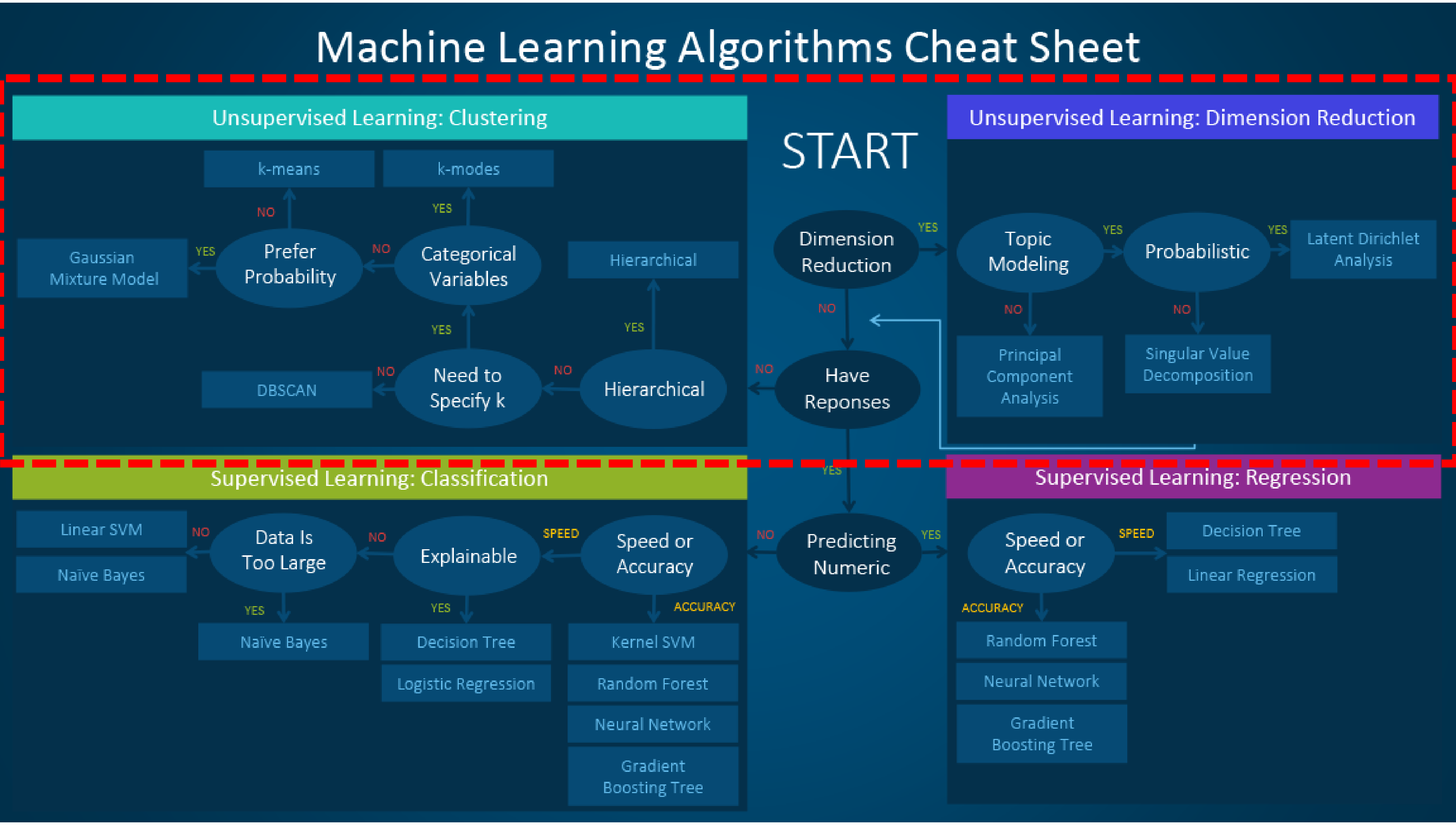
Python Unsupervised Machine Learning

ST0249 (AIML) AI & MACHINE LEARNING

Learning Outcomes

- Understand Unsupervised Learning
  - Discuss types of unsupervised learning
  - Explain the challenges of unsupervised learning
  - Explain pre-processing and scaling
  - Understand Clustering
  - Explain dimensionality reduction
  - Anomaly Detection

Unsupervised Learning



source https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

Unsupervised Learning

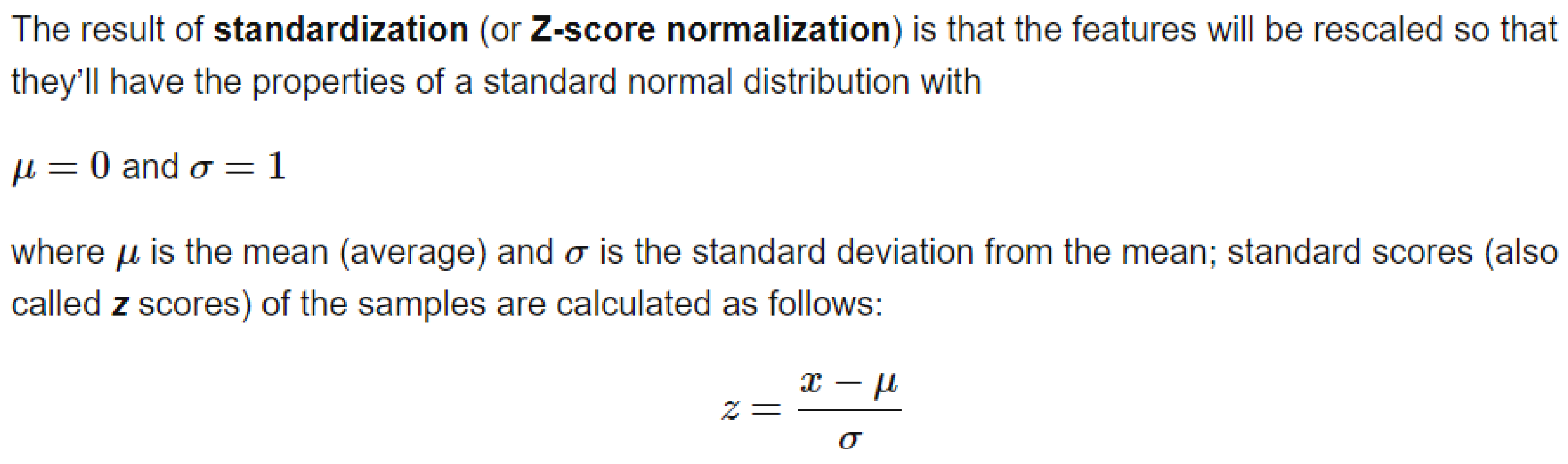
- In unsupervised learning, data points have no labels associated with them. Instead, the goal of an unsupervised learning algorithm is to **organize the data in some way** or **to describe its structure**.
- This can mean **grouping** it into **clusters** or finding different ways of looking at complex data so that it appears simpler or more organized.
- This can also mean **dimension reduction**.

Challenges of unsupervised learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- It is intrinsically more difficult than supervised learning because there is no gold standard (like an outcome/target variable) and no single objective (like test set accuracy)
- Data is unlabelled. Initial values for algorithms such as k-means number of centroids, k, is unknown.
- Scaling of variables/features matters

Pre-processing and feature scaling

Standardizing the features so that they are centred around 0 with a standard deviation of 1 is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.



The result of **standardization** (or **Z-score normalization**) is that the features will be rescaled so that they'll have the properties of a standard normal distribution with

$\mu = 0$ and $\sigma = 1$

where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called **z** scores) of the samples are calculated as follows:

$$z = \frac{x - \mu}{\sigma}$$

source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Pre-processing and feature scaling

Some examples of algorithms where feature scaling matters are:

- k-nearest neighbours with an Euclidean distance measure if want all features to contribute equally
- k-means (see k-nearest neighbours)
- logistic regression, SVMs, perceptrons, neural networks etc. if you are using gradient descent/ascent-based optimization, otherwise some weights will update much faster than others.
- linear discriminant analysis, principal component analysis, kernel principal component analysis since you want to find directions of maximizing the variance (under the constraints that those directions / eigenvectors / principal components are orthogonal); you want to have features on the same scale since you'd emphasize variables on "larger measurement scales" more.

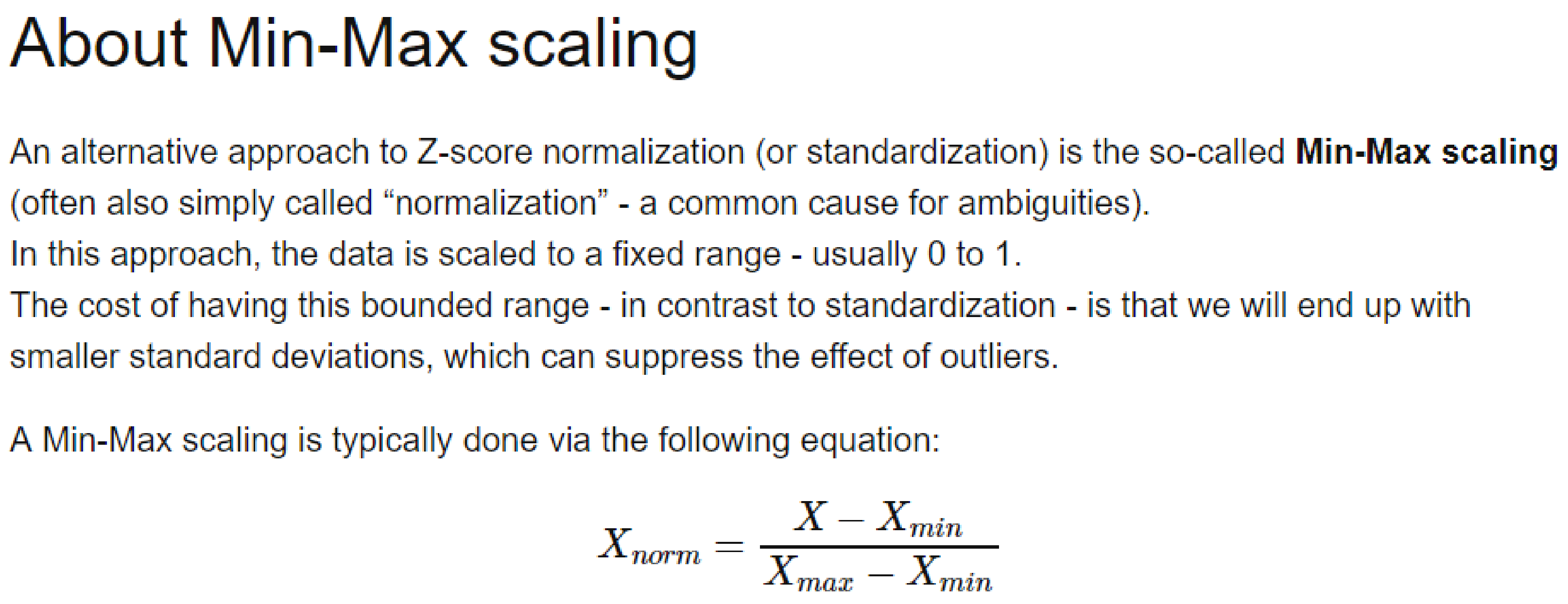
source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Pre-processing and feature scaling

In addition, we'd also want to think about whether we want to "standardize" or "normalize" our data. Some algorithms assume that our data is centred at 0. For example, if we initialize the weights of a small multi-layer perceptron with tanh activation units to 0 or small random values centred around zero, we want to update the model weights "equally." As a rule of thumb I'd say: When in doubt, just standardize the data, it shouldn't hurt.

source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Pre-processing and feature scaling



About Min-Max scaling

An alternative approach to Z-score normalization (or standardization) is the so-called **Min-Max scaling** (often also simply called “normalization” - a common cause for ambiguities).
In this approach, the data is scaled to a fixed range - usually 0 to 1.
The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.

A Min-Max scaling is typically done via the following equation:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

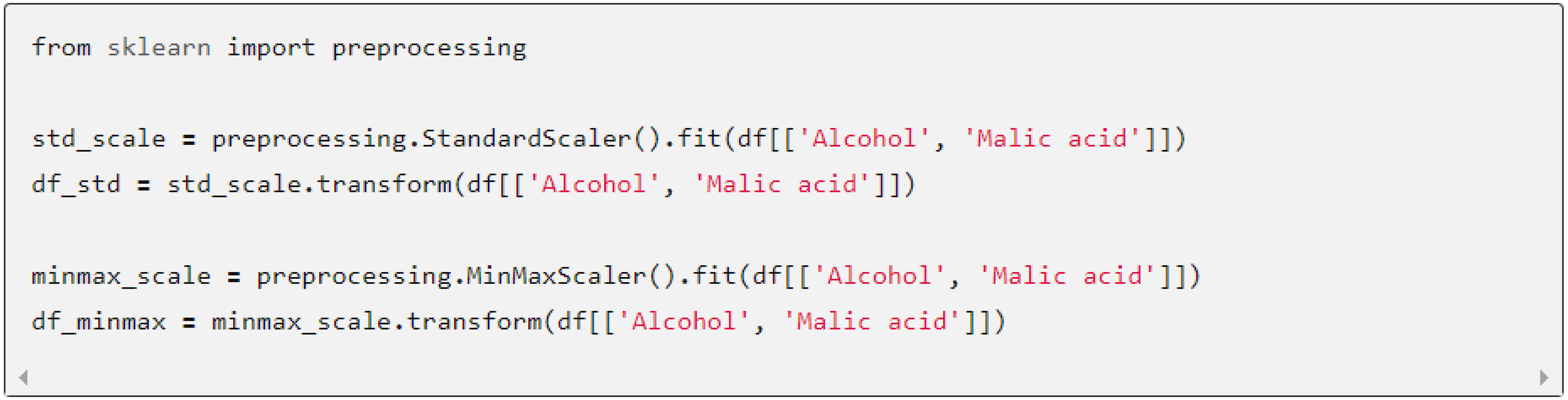
source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Z-score standardization or Min-Max scaling?

- “Standardization or Min-Max scaling?” - There is no obvious answer to this question: it really depends on the application.
- For example, in **clustering analyses,** standardization may be especially crucial in order to compare similarities between features based on certain distance measures. Another prominent example is the **Principal Component Analysis**, where we usually prefer standardization over Min-Max scaling, since we are interested in the components that maximize the variance.
- However, this doesn’t mean that Min-Max scaling is not useful at all! A popular application is **image processing**, where pixel intensities have to be **normalized to fit within a certain range** (i.e., 0 to 255 for the RGB colour range). Also, typical **neural network algorithm** require data that on a 0-1 scale.

source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

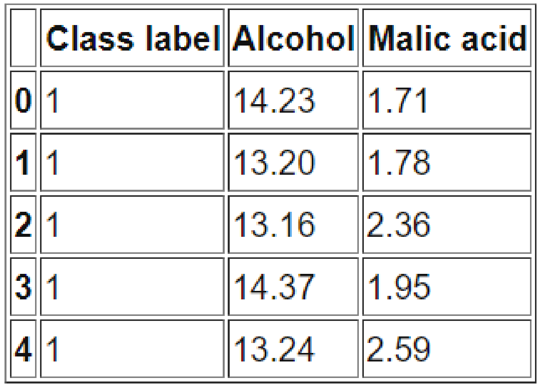
Standardization & Scaling in scikit-learn



```
from sklearn import preprocessing

std_scale = preprocessing.StandardScaler().fit(df[['Alcohol', 'Malic acid']])
df_std = std_scale.transform(df[['Alcohol', 'Malic acid']])

minmax_scale = preprocessing.MinMaxScaler().fit(df[['Alcohol', 'Malic acid']])
df_minmax = minmax_scale.transform(df[['Alcohol', 'Malic acid']])
```



	Class label	Alcohol	Malic acid
0	1	14.23	1.71
1	1	13.20	1.78
2	1	13.16	2.36
3	1	14.37	1.95
4	1	13.24	2.59

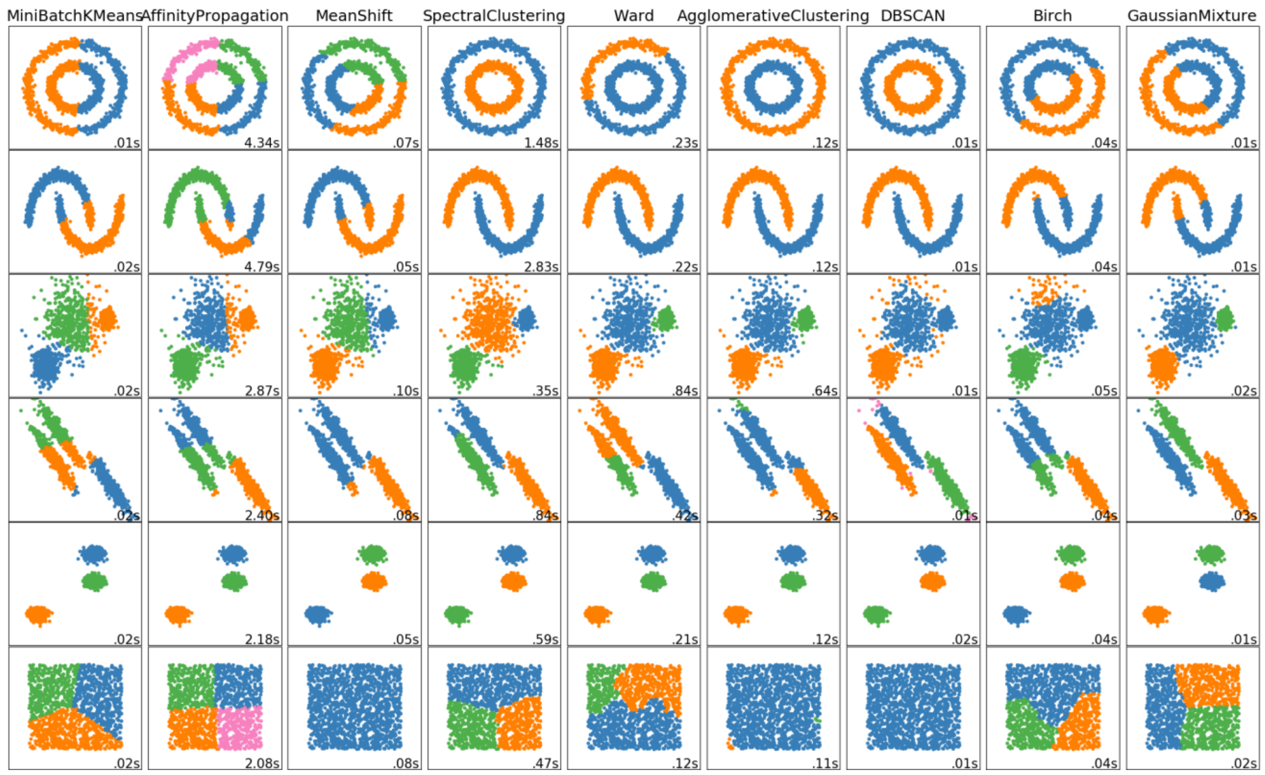
source http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Clustering

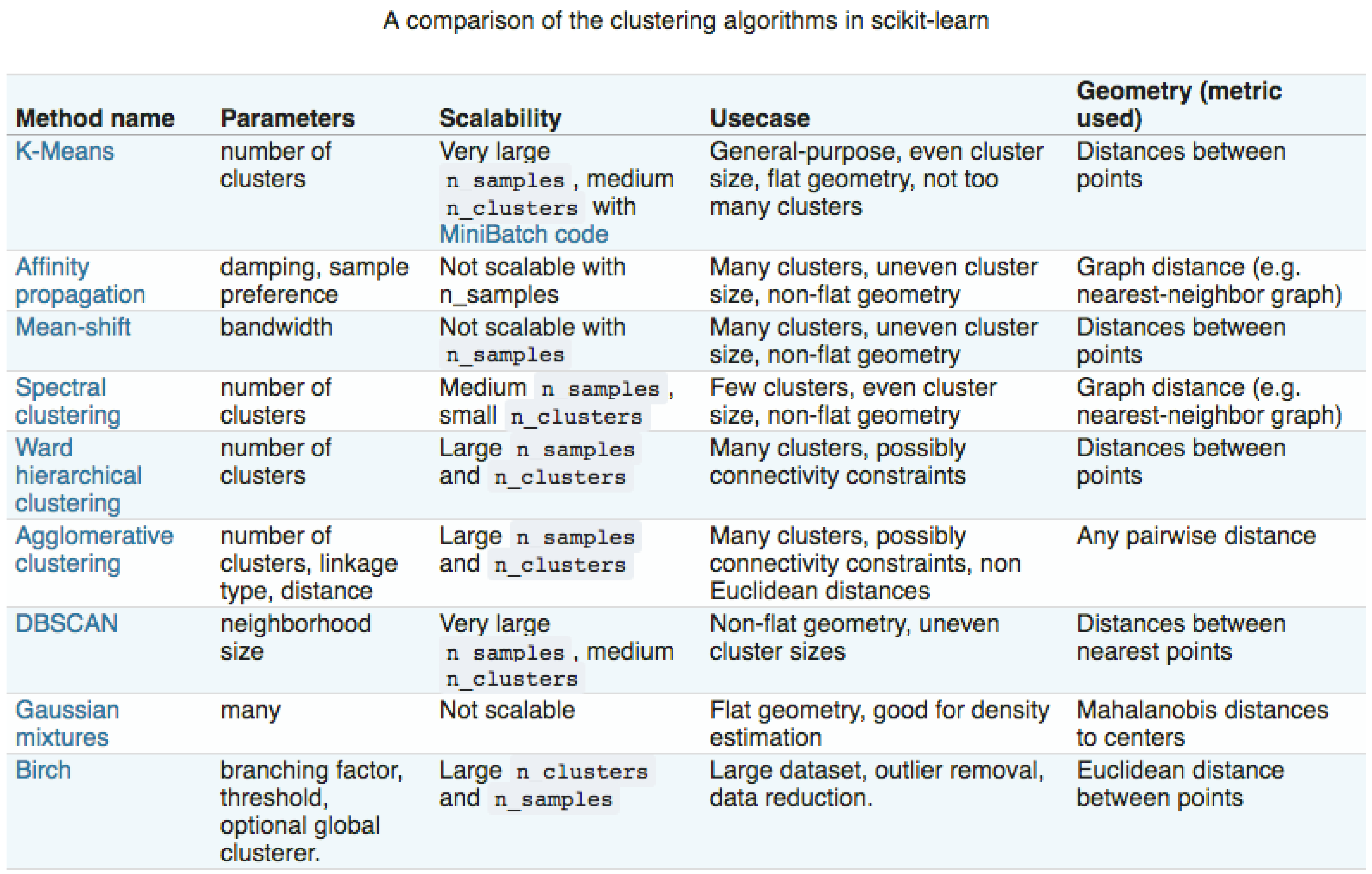
What is clustering?

- **Clustering**, in machine learning, is a method of grouping data points into similar clusters. It is also called **segmentation**.
- Over the years, many clustering algorithms have been developed. Almost all clustering algorithms use the features of individual items to find similar items.
- For example, you might apply clustering to find similar people by demographics. You might use clustering with text analysis to group sentences with similar topics or sentiment.
- Clustering is called a non-supervised learning technique because it can be used in unlabeled data. Indeed, clustering is a useful first step for discovering new patterns, and requires little prior knowledge about how the data might be structured or how items are related.
- Clustering is often used for exploration of data prior to analysis with other more predictive algorithms.

Comparison of clustering algorithms



source http://scikit-learn.org/stable/modules/clustering.html



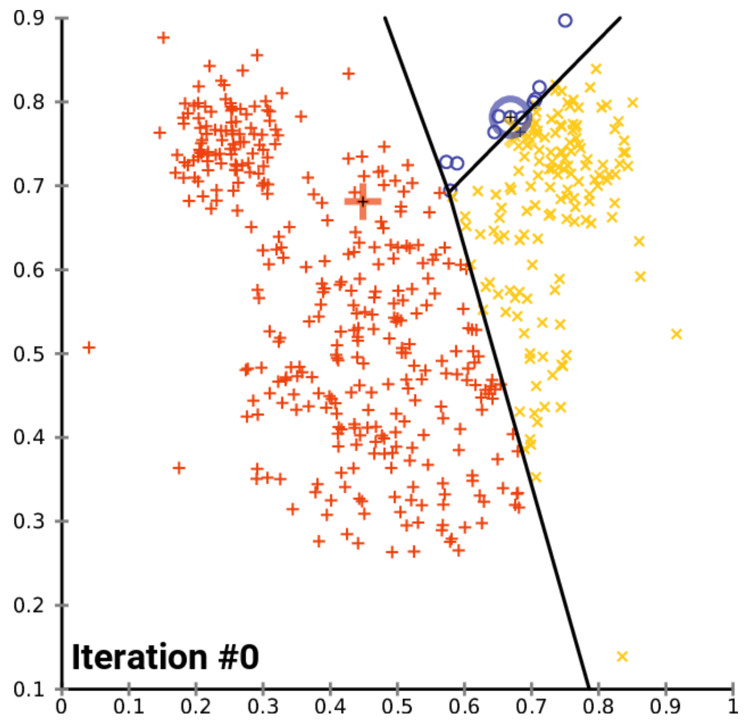
A comparison of the clustering algorithms in scikit-learn

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	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

source http://scikit-learn.org/stable/modules/clustering.html

K-means Clustering

- When you configure a clustering model using the k-means method, you must specify a target number k indicating the number of **centroids** you want in the model.
- The centroid is a point that is representative of each cluster.
- The K-means algorithm assigns each incoming data point to one of the clusters by minimizing the within-cluster sum of squares.



K-means algorithm

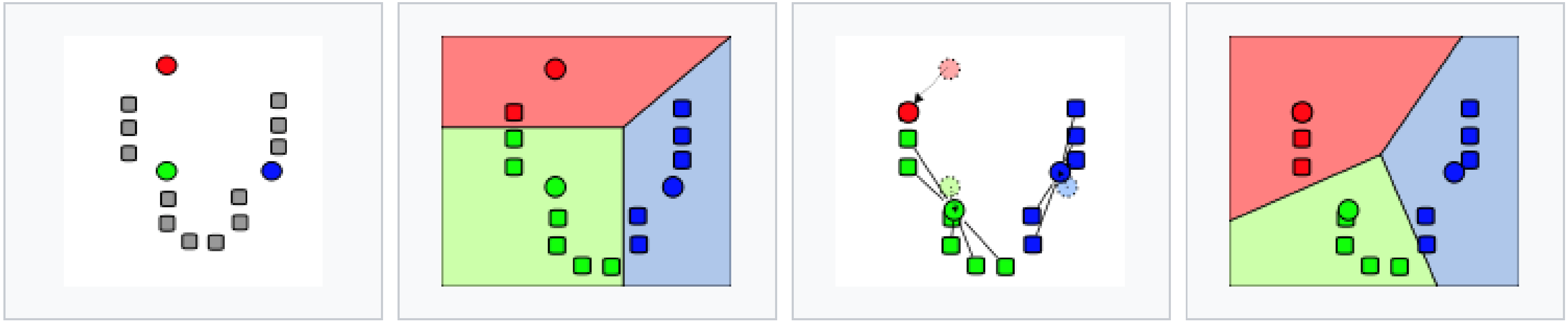
Given an initial set of k means $m_1^{(1)},\ldots,m_k^{(1)}$, the algorithm proceeds by alternating between two steps:

Assignment step: Assign each observation to the cluster whose mean has the least squared Euclidean distance, this is intuitively the "nearest" mean.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

K-means algorithm

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

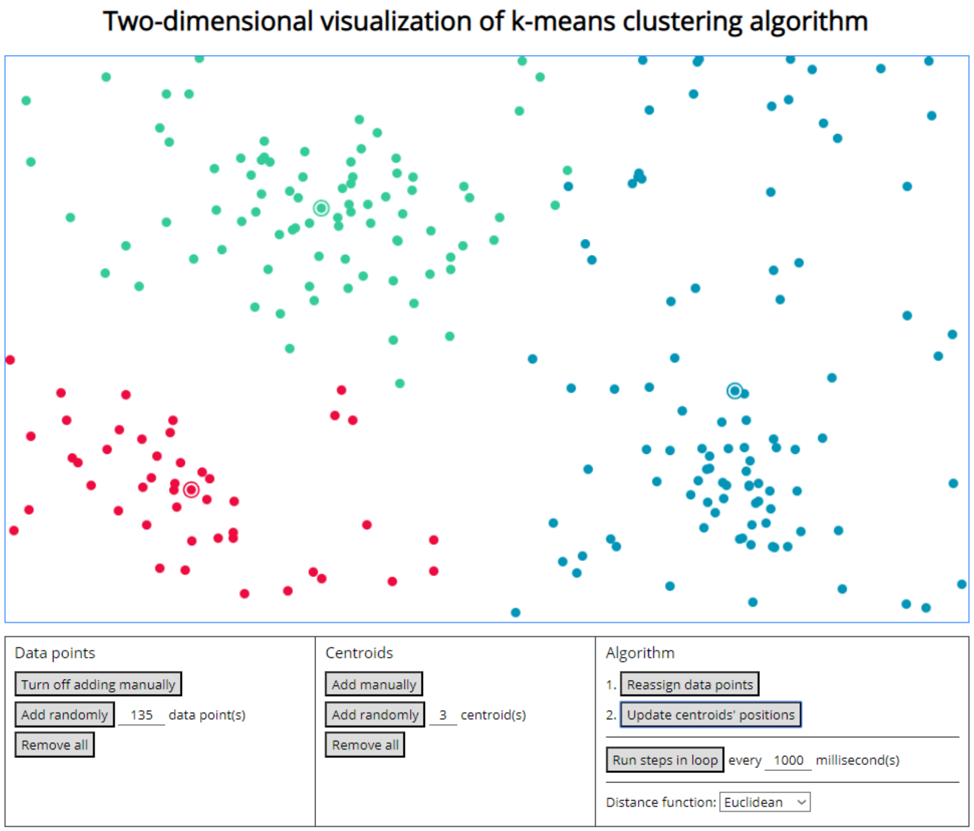
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Activity

https://hckr.pl/k-means-visualization/



Two-dimensional visualization of k-means clustering algorithm

135 data point(s)

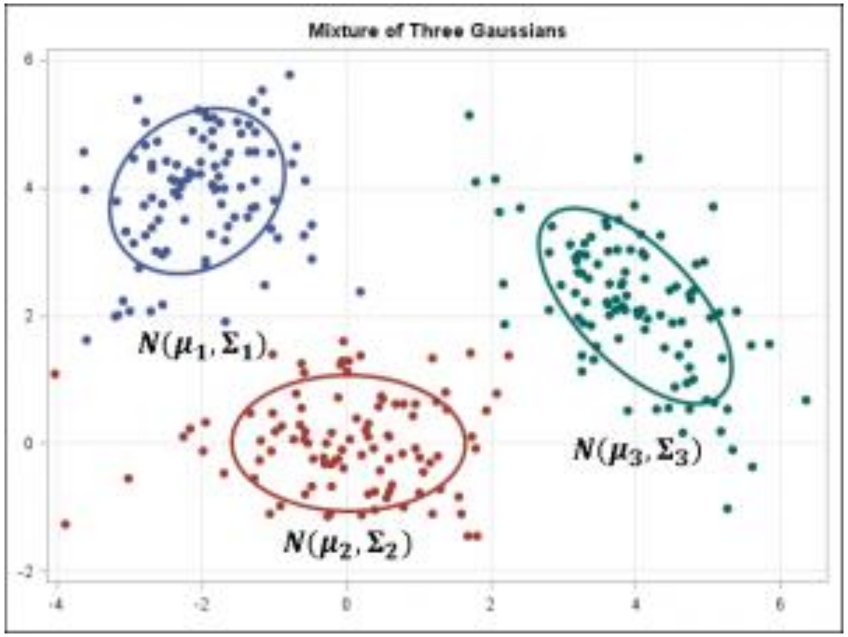
3 centroid(s)

every 1000 millisecond(s)

Distance function: Euclidean

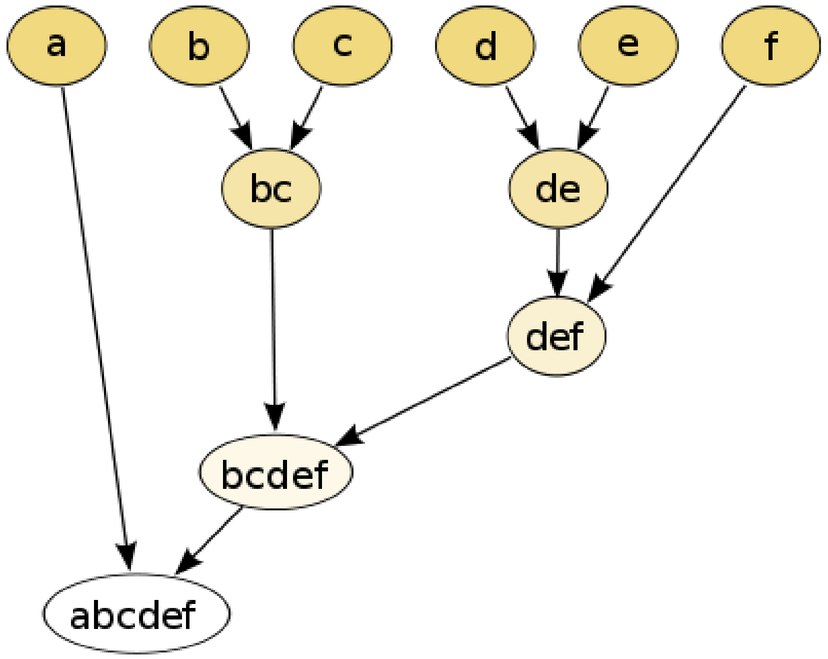
GMM (Gaussian mixture model) clustering

- GMM clustering aims to partition n observations into k clusters. K-means define hard assignment: the samples are to be and only to be associated to one cluster.
- GMM, however define a soft assignment for each sample.
- Each sample has a probability to be associated with each cluster.



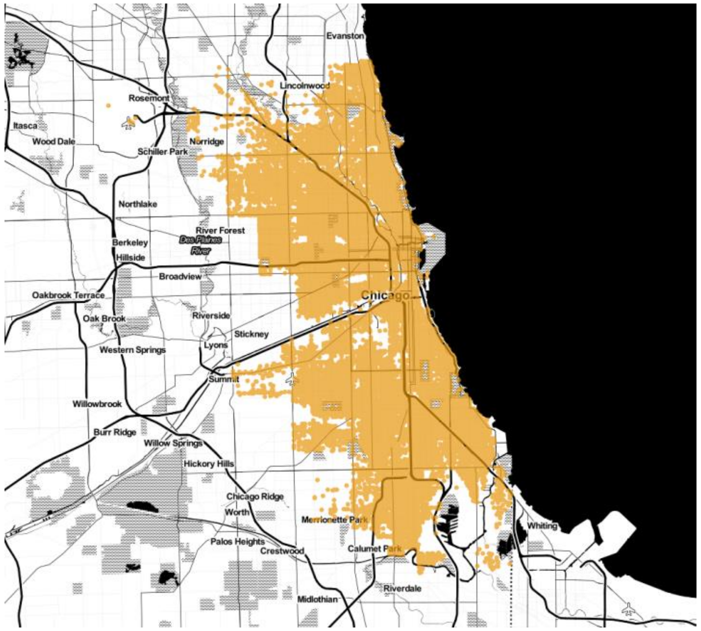
Hierarchical Clustering

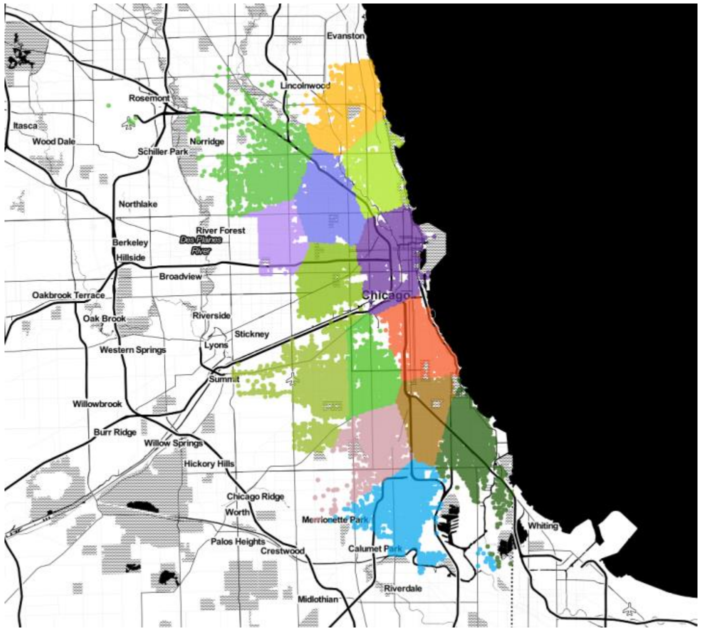
- Hierarchical partitions can be visualized using a tree structure (a dendrogram).
- It does not need the number of clusters as an input and the partitions can be viewed at different levels of granularities (i.e., can refine/coarsen clusters) using different K.



Activity

https://arkadiuszkondas.com/clustering-chicago-robberies-locations-with-k-means-algorithm/





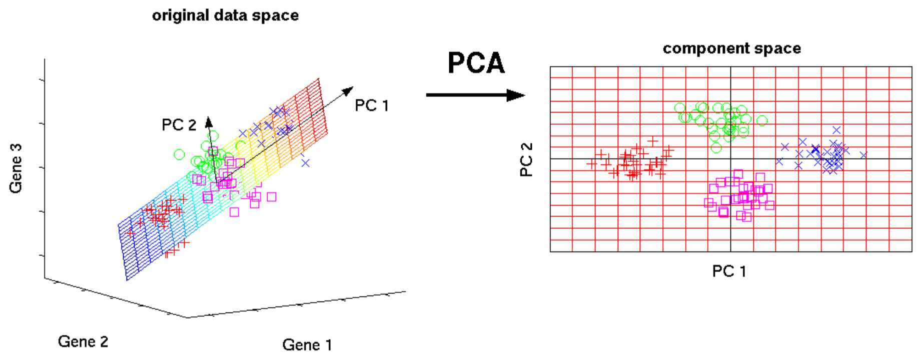
Dimension Reduction

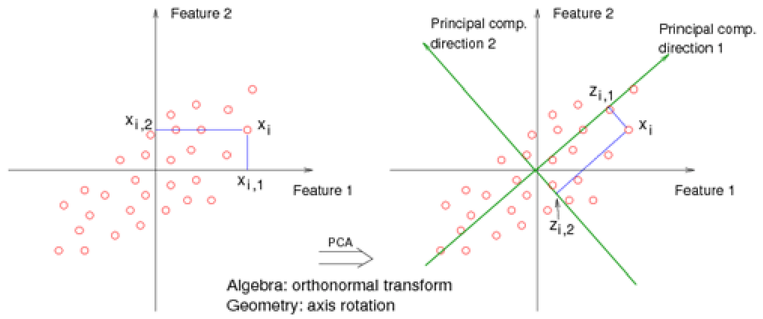
Understanding Dimension Reduction

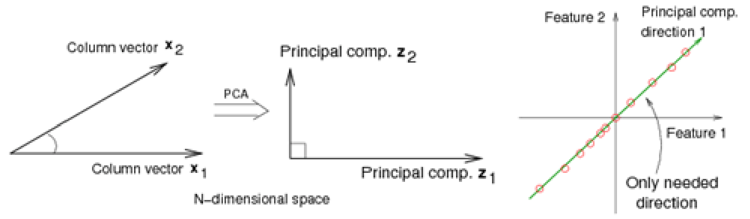
- We generally do not want to feed a large number of features directly into a machine learning algorithm since **some features may be irrelevant** or the "intrinsic" dimensionality may be smaller than the number of features.
- Principal component analysis (PCA), singular value decomposition (SVD), and linear discriminant analysis (LDA) all can be used to perform **dimension reduction**.

Principal Component Analysis (PCA)

- PCA is an unsupervised clustering method which maps the original data space into a lower dimensional space while preserving as much information as possible.
- The PCA basically finds a subspace that most preserves the data variance, with the subspace defined by the dominant eigenvectors of the data's covariance matrix.

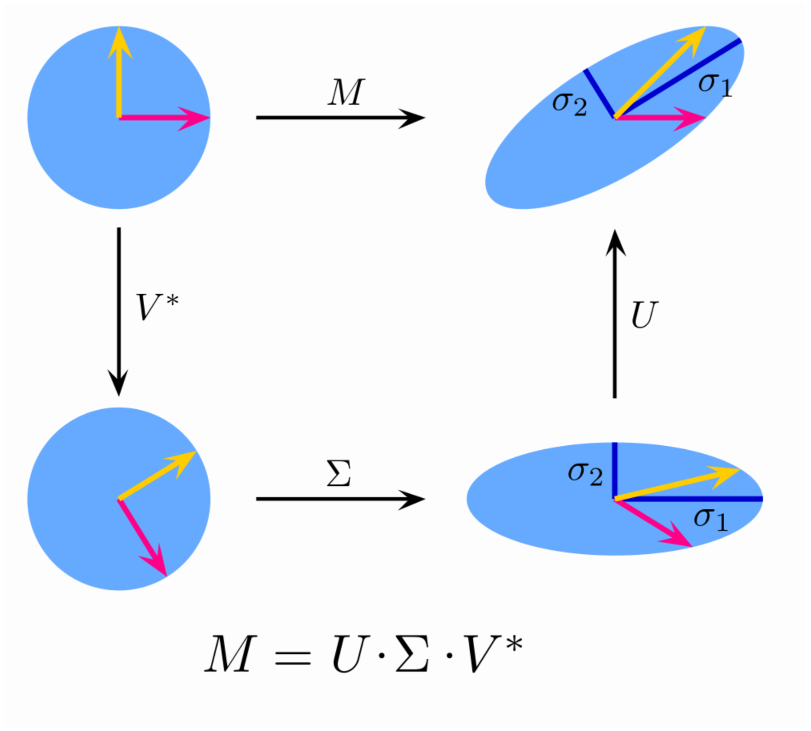






Singular Value Decomposition (SVD)

- The SVD is related to PCA in the sense that SVD of the centred data matrix (features versus samples) provides the dominant left singular vectors that define the same subspace as found by PCA.
- In linear algebra, the singular-value decomposition (SVD) is a factorization of a real or complex matrix.



Singular Value Decomposition (SVD)

- However, SVD is a more versatile technique as it can also do things that PCA may not do. For example, the SVD of a user-versus-movie matrix is able to extract the user profiles and movie profiles which can be used in a recommendation system.
- In addition, SVD is also widely used as a topic modelling tool, known as latent semantic analysis, in natural language processing (NLP).

Linear Discriminant Analysis (LDA)

A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.

The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions.

Note: Unlike PCA, LDA is not an unsupervised learning algorithm and requires labelled data

https://scikit-learn.org/stable/modules/lda_qda.html

LDA

- LDA is based upon the concept of searching for a linear combination of variables (predictors) that best separates two classes (targets).
- To capture the notion of separability, Fisher defined the following score function.
- Given the score function, the problem is to estimate the linear coefficients that maximize the score.

https://www.saedsayad.com/lda.htm

$$Z = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_d x_d$$

$$S(\beta) = \frac{\beta^T \mu_1 - \beta^T \mu_2}{\beta^T C \beta}$$

Score function

$$\Big\downarrow$$

$$S(\beta) = \frac{\overline{Z}_1 - \overline{Z}_2}{\text{Variance of Z within groups}}$$

Comparison of LDA and PCA 2D projection of Iris dataset

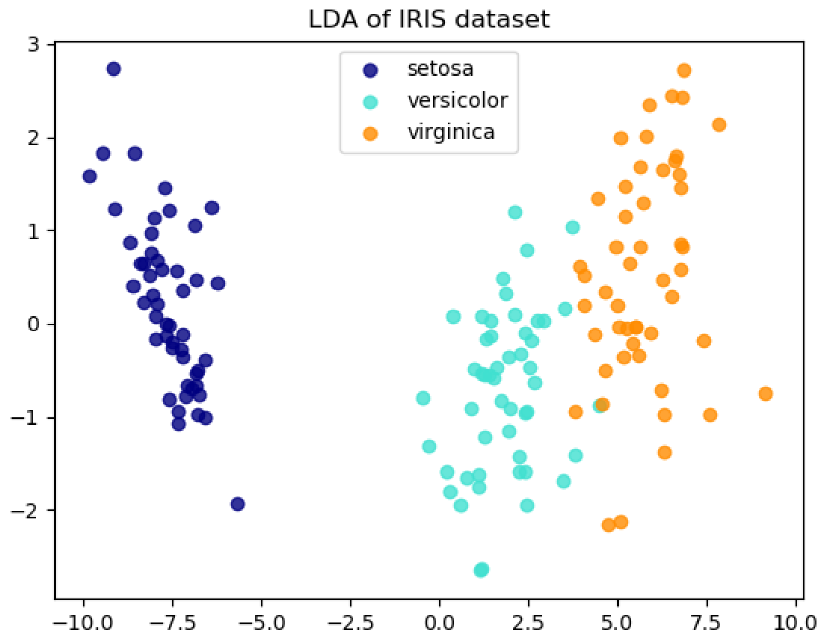
The Iris dataset represents 3 kind of Iris flowers (Setosa, Versicolour and Virginica) with 4 attributes: sepal length, sepal width, petal length and petal width.

Principal Component Analysis (PCA) applied to this data identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data. Here we plot the different samples on the 2 first principal components.

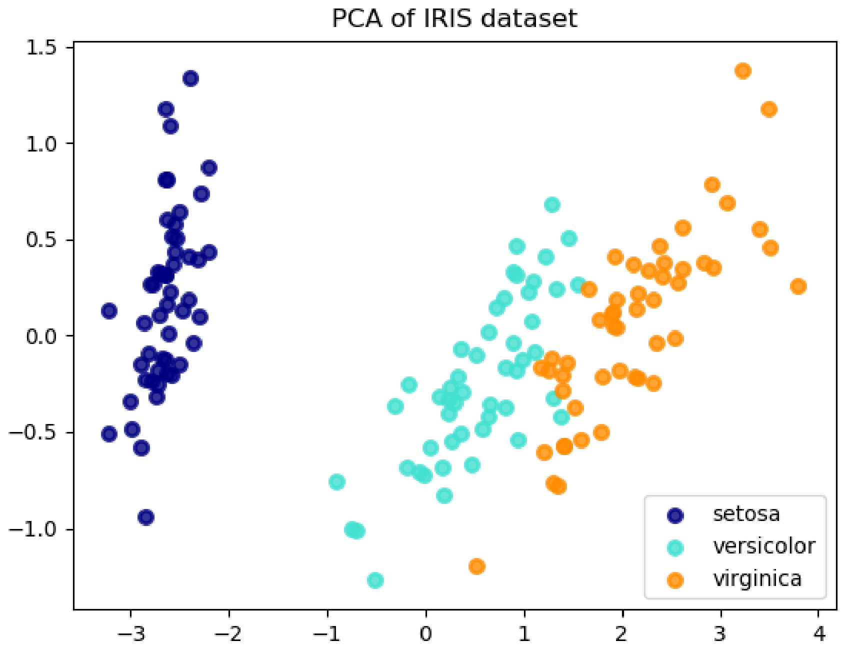
Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance *between classes*. In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.

https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html

LDA vs PCA



LDA of IRIS dataset
● setosa
● versicolor
● virginica
3
2
1
0
−1
−2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

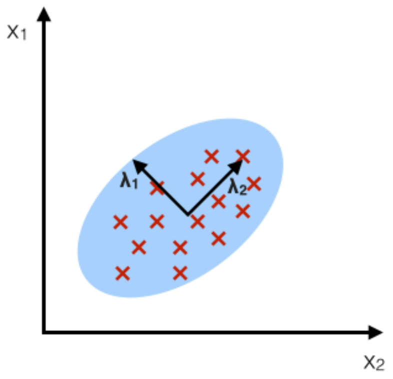


PCA of IRIS dataset
● setosa
● versicolor
● virginica
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4

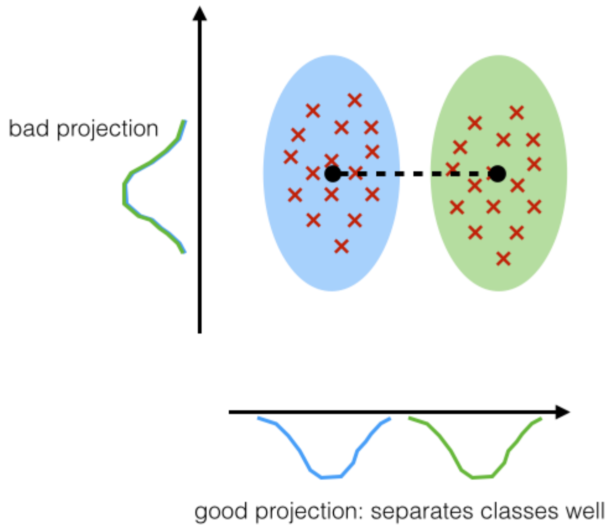
LDA vs PCA

https://sebastianraschka.com/Articles/2014_python_lda.html

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



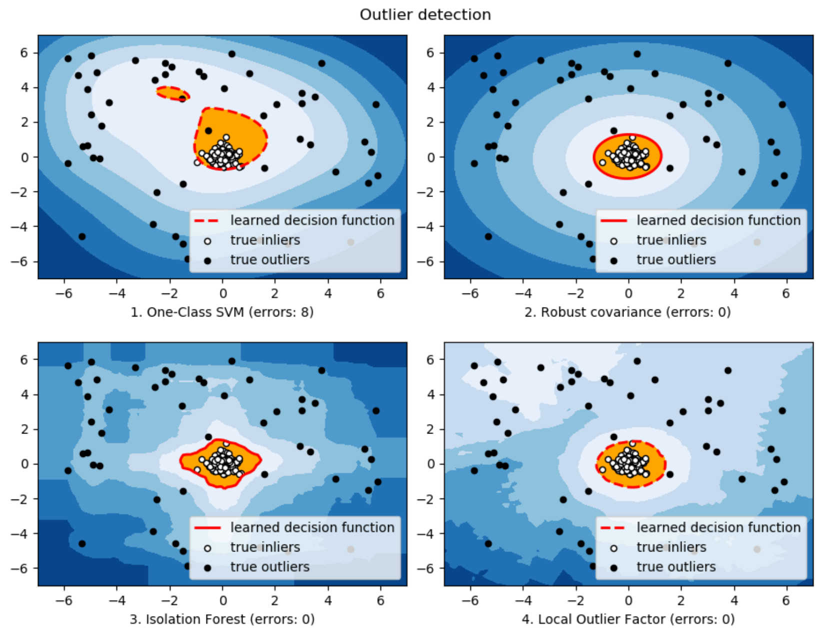
Anomaly Detection

Anomaly Detection

Anomaly detection encompasses many important tasks in machine learning:

- Identifying transactions that are potentially fraudulent.
- Learning patterns that indicate that a network intrusion has occurred.
- Finding abnormal clusters of patients.
- Checking values entered into a system.

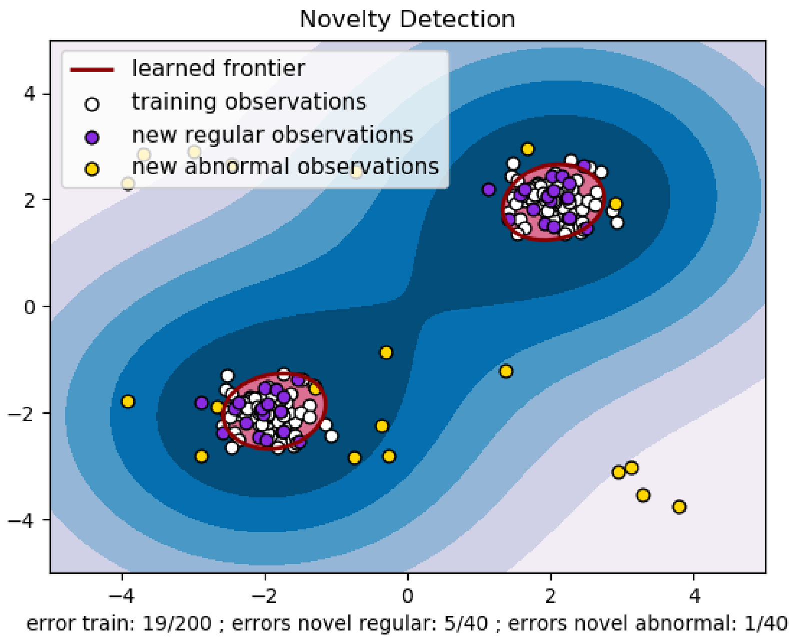
Because anomalies are rare events by definition, it can be difficult to collect a representative sample of data to use for modelling. The algorithms included in this category have been especially designed to address the core challenges of building and training models by using imbalanced data sets.



source http://scikit-learn.org/stable/modules/outlier_detection.html#novelty-detection

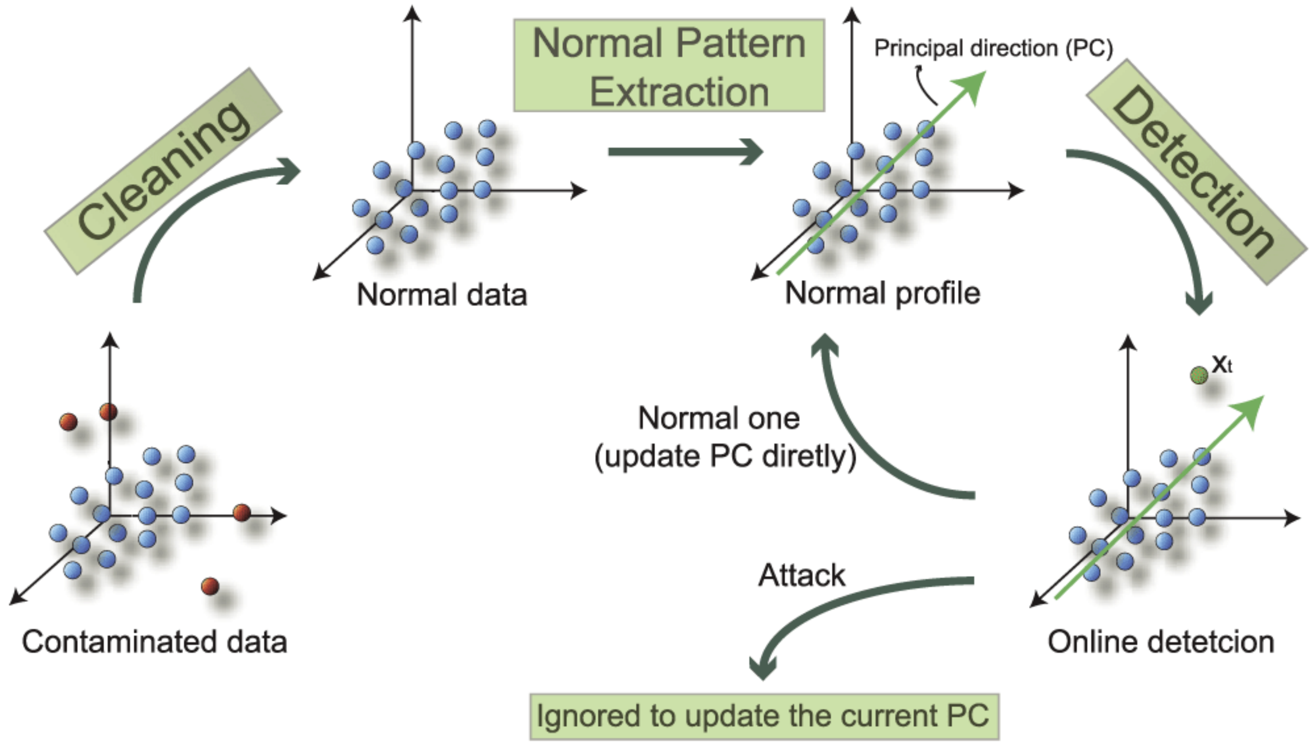
One-class SVM

- In one-class SVM, the support vector model is trained on data that has only one class, which is the “normal” class.
- It infers the properties of normal cases and from these properties can predict which examples are unlike the normal examples.
- This is useful for anomaly detection because the scarcity of training examples is what defines anomalies: that is, typically there are very few examples of the network intrusion, fraud, or other anomalous behaviour.



PCA-based anomaly detection

- The PCA-Based Anomaly Detection module solves the problem by analysing available features to determine what constitutes a "normal" class, and applying distance metrics to identify cases that represent anomalies.



Summary

What we have learnt:

- Types of unsupervised learning
- Challenges of unsupervised learning
- Pre-processing and scaling
- Clustering
- Dimensionality Reduction
- Anomaly Detection