
DATA
features x samples
PREPROCESSING
normalization
outliers
dimensionality reduction
feature selection
un/supervised
filters/wrappers/embedded
correlation, mutual info
projection
global/local
non/linear
kernel/ANN
un/supervised
PCA, LDA, Isomap/LLE, SOM
REGRESSION
CLUSTERING (unsupervised)
DBSCAN
K-Means
Hierarchical clustering
EM (probability distribut°)
CLASSIFICATION (supervised)
naive Bayes
Perceptron - SVM
Adaboost (weighted seq) ⇒ boosting
bagging
kNN
rule-based methods
random forest
VALIDATION
cross-validat°
train/test sets
bootstrapping
overfitting.

DATA: what are your raw data?
Sometimes you have to extract your features,
This part is really context dependent.
Usually ML starts when you have extracted your features,

PREPROCESSING: this comes when you have extracted your features,

- sometimes you need to convert your features to numerical values,
  or to normalize them, e.g. for PCA you need to have the same range of values that the feature can reach because you'll compare variances, and they are expressed in the ^{2} of your base unit.

- sometimes you know you have outliers in your data, some methods are very sensitive to instance outliers like k-means.
  Outliers can be detected via distance/density methods: you have to know some structure of your data to estimate a property that will allow you to discriminate outliers from "good" points.

- if you have "too many" features, you can apply techniques to reduce your dimensionality. They are of 2 types: - project° or combinat° of your features
  - feature selection.

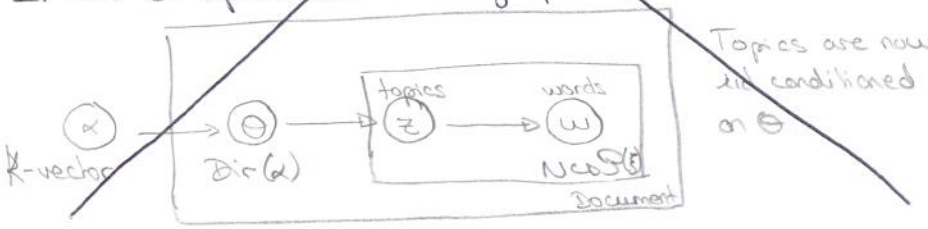
- The simplest example of projection method is PCA = principal analysis component. It will work as a linear combination of your original features and rank the linear combinations according to how much they explain the global variance in the dataset. It's an unsupervised method. It's based on the diagonalizat° of the covariance matrix of your dataset, the new dimensions are the eigenvectors found, so they are orthogonal. The weight you get for each new dimension is the normalized variance explained by this new axis. Then you can choose to keep m new features because they keep k% (eg 90%) of the variance of your original dataset
  People have adapted PCA to a non-linear context, using a kernel. The idea is to project w. the kernel your original space to another space, in which you can apply PCA
  ↳ to be confirmed.

  Another variat° of project° methods are the one focusing on explaining the local variat° rather than the global as PCA. They are based on a definit° of the neighborhood which is to be preserved: Isomap, LLE...

  Self-Organizing Map is an example of a neural-network used for dimensionality reduction. It's unsupervised. You choose ahead the structure of the grid you want your data to be, and then you learn on an unsupervised manner to associate data points with 1 neuron of your predefined grid. It uses competitive learning bw the neurons. At the end, your grid has learned some of the local interact° in your dataset.

  This step can also take place in an supervised context, e.g. LDA for Linear Discriminant Analysis

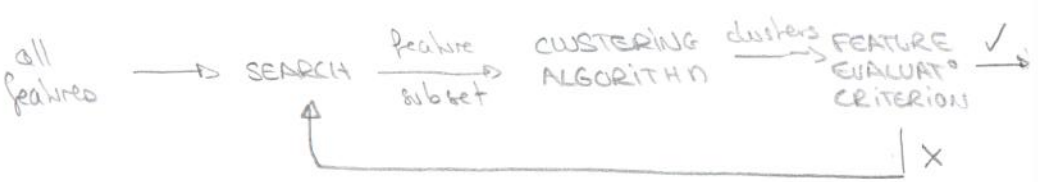
It's a way of automatically discovering associat° bw observed instances and categories (more high-level). The most cited example are about words (= observed examples) and the type of text those words are contained in. It builds an internal model of joint dirichlet distribut°, based on texts you already know the type (that's the supervised part). Basically you learn which words are associated w. which type of texts. The K latent variables correspond to the K distribut° of probability. It can be represented as a graphical model



- The feature selection methods are also interesting because they allow to keep the initial features and not a combinaison of them so it's easier to interpret.

Most of the feature select° methods which have been dvlped are supervised: they rate the feature subsets according to "are they useful to determine class?"

You can do an individual evaluat°, known as feature ranking, or a subset evaluat° which implies a search in the feature subset space. For subset evaluat°, we distinguish 3 categories of methods: filters, wrappers and embedded methods. Filters are the one totally independent of the classificat°/clustering you do afterwards: it's usually fast and generalizable but it could not give the best results. Filter methods can be based on χ^2, informat°, correlat° fct°. Wrappers use a learning method to assess the relative usefulness of subsets of variables. Better results usually than filters, but computationally expensive and prone to overfitting



Embedded methods imply feature select° in the process of training, it has a lower cost than wrappers. The search is guided by the learning process
ex: forward select° w. decision tree.

REGRESSION: this is the part of machine learning where what we try to learn and predict is a quantitative value.

The simple form is the method of least squares (gives an estimator). It's usually supervised.

CLUSTERING we call clustering the process of assigning a discrete label to the data (a class) but on an unsupervised manner. The idea is to uncover some of the dataset properties by grouping examples together.

You can find that some of the goals of clustering are common with manifold learning = locally-based projections as Isomap, LLE, SOM where there is already a process of choosing which link bw features make a "neighbor", will belong to the same cluster.

- The easiest method to illustrate this part is K-means: you choose k, the number of wanted clusters. You initiate the k cluster centers randomly or with a "smarter" method depending on the context. It's an iterative algorithm where you alternate 2 steps until the clusters don't change anymore: (1) you assign each data point to a cluster center (the closest according to the distance metric you use); (2) you move the centers to the mean of (barycenter w. equal weights) of the labeled datapoints (label = cluster label).
This method is sensitive to the initializat°, that's why you can use a "smarter" method to place your cluster centers, or you can test it several times w. ≠ initializat°.

- Hierarchical clustering is also a method based on a (distance) metric. You first group together the 2 datapoints that are the closest in the dataset, and you iterate like that until you satisfy: a nb of clusters & an evaluat° fct° or a max distance. It's usually represented with a tree.

- DBSCAN = density based spatial clustering of applications w. noise. You have to specify 2 parameters: a distance ε and the min nb of points in a radius ε for this points to be considered as a cluster.
It's a really simple algorithm, you don't have to provide the nb of wanted clusters. You still have to understand enough your data to provide (ε, minPts), and it can be difficult if your clusters don't have the same "ideal" parameters (ε, minPts).

- You can also build a model of a mixture of probabilities and fit it on your data via the EM (espectat° - maximizat°) algorithm. You can choose the type of probabilities you want (G, Poisson...) and the nb of components. Be careful the more parameters you add to your model, the longer / more difficult it will be for the EM algo to find a good fit. The EM algo is a very customizable algo, especially by the evaluat° fct° you choose (usually sth that penalizes models that don't give high enough probabilities where there are enough data points).

EXTRA

- **Naive Bayes Classifier**: apply Bayes' theorem w. strong independence assumptions bw feature.

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

$k \in \{1,\dots,K\}$: classes

$\prod_{i=1}^{n}$: indep bw features.

$x = (x_1, \dots, x_n)$

- **Model ensembles**:
  - **bagging**: generate rd variat° on the training set by resampling, learn a classifier on each, and combine the results by voting
    ↘ variance, ↗ bias
  - **boosting**: training examples have weights, and these are varied so that each new classifier focuses on the examples the previous ones tended to get wrong
  - **stacking**: the outputs of individual classifiers become the inputs of a "higher-level" learner that figures out how best to combine them.

- **Perceptron**: single-layer neuron
  linear classifier

Σ