MACHINE LEARNING

With content adapted from the thesis PhD “Desenvolvimento de Modelos Analíticos de Apoio à Gestão de Instituições do Ensino Superior, com Recurso a Data Mining”, of Maria Martins, 2020

Multi-class classifiers

- So far, we've only deal with binary classifiers
  - *Each input example is classified into one of two classes (one considered positive, the other negative)*
  - *Examples: prediction of the occurrence of precipitation or the occurrence of a given pathology*
- But many of the problems we deal with are multi-class
  - *Each input example is sorted into one of several classes*
  - *Examples: classification of an iris plant (one of 3 species) or automatic recognition of handwritten digits (or other characters)*
- Although there are classifiers intrinsically prepared for multi-class, many others have been designed specifically for binary classification
  - *But even those are easily adaptable to multi-class problems*

Multi-class binary classifiers

Binary classifiers are converted into multi-class classifiers, using, as a rule, one of two strategies:

The one-versus-all strategy and the one-versus-one strategy.

- The one-versus-all strategy – a binary classifier is induced for each of the n classes, looking at the rest as the negative class (then for each input example is chosen the model class that can be predicted most likely)
  - *for example, to classify the iris plants in the 3 species, 3 binary classifiers are created to distinguish: versicolor from the rest, virginica from the rest, and setosa from the rest (already to recognize manuscript digits are required n=10 binary classifiers)*
  - *This is the most commonly used option. Besides presenting a linear efficiency in relation to the number of classes (only n classifiers are required), it also has the advantage of its interpretability. Because each class is represented by a classifier, we can gain knowledge about the class by analyzing the corresponding classifier.*

Multi-class binary classifiers

Binary classifiers are converted into multi-class classifiers, using, as a rule, one of two strategies:

- the one-versus-one strategy – a binary classifier is induced for each pair of classes, which makes a total of n(n-1)/2 classifiers, where n is the number of classes (then for each input example, the most voted class is chosen, that is, the one that was most often chosen in the various direct confrontations)
    - *for example, to classify iris plants in the 3 species, 3 binary classifiers are created to distinguish: versicolor from virginica, versicolor of setose, and setosa of virginica (already to recognize handwritten digits are needed n(n-1)/2=10(9)/2=45 binary classifiers)*
    - *It is a less efficient approach, given its complexity in relation to the number of classes being quadratic (n(n-1)/2 classifiers are required).*
    - *However, it should be noted that each of your binary classifiers only involves a small subset of the data (only the examples classified in one of the two classes) – in one-against-all, each binary classifier uses the entire dataset*

Multi-class and multi-label classifiers

- All Scikit-learn classifiers can be applied to multi-class problems
  - *while random forests (RandomForestClassifier) and neuronal networks (MLPClassifier) are already intrinsically prepared for multi-class,*

    SVM (SVC) are helped from the one-versus-one scheme

    and logistic regression (LogisticRegression) of one-versus-all

    If desired, we may change the mutli-class strategy of these algorithms using the Meta estimators OneVsRestClassifier and OneVsOneClassifier of the sklearn.multiclass module

- Do not confuse multi-class classification with multi-label classification
  - *Unlike multi-class classification, where each input example only assumes one class among the possible n, in the multi-label classification each example can simultaneously assume multiple classes*

    For example, the same film can be classified into historical, biographical and drama genres

Multi-class confusing matrix

How to evaluate multi-class ratings?

- The concept of confusion matrix used in binary classification is easily generalized in case of terms n classes
    - *just assume that the value of row i and column k of the confusion array represents the number of examples of the test set that, belongs to the class C_i, were classified by the model as being of the class C_k*
- For example, if the following confusion matrix mirrors the result of a model in the iris dataset classification, we can complete the following:
    - *The model performs well, since in 38 plants used for testing, it only misclassified one of them: a versicolor plant was classified as virginica*

previsto	setosa	versicolor	virginica
real			
setosa	13	0	0
versicolor	0	12	1
virginica	0	0	12

Metrics for multi-class classification

- The generality of the remaining metrics is defined for binary classification
  - *Therefore, in the evaluation of the multi-class classification, what is usually done is to use the average of one of these metrics over the various classes*
- However, the average of the 'binary' metric to be applied in multi-class evaluation may take different forms:
  - *Macro - the respective binary metric is calculated for each class and then the arithmetic mean of these values is considered*
  - *Micro – calculates the 'binary' metric globally, accounting for all true positives, true negatives, false positives and false negatives, regardless of classes*
  - *Weighted - identical to Macro, but weights the average depending on the number of examples in each class*
- One of the criteria for choosing the type of mean to use is the unbalanced level of the classes
  - *If classes have a very different number of occurrences, the Macro mean is more indicated, since it gives the same importance to all classes*
  - *But if we still want to give some more importance to the majority classes, we can always use the Weighted Average*

Calculation Example of macro and micro means

- The calculation of macro and micro averages is exemplified as two alternative ways to transform the binary metric 'precision' into a multi-class metric,
  - *the same procedure can then be followed to calculate the Macro and Micro average of any other binary classification metric*
  - *It should be said, however, that Scikit-learn already puts at our disposal functions that calculate all these averages (see next slide)*

- Example of the calculation of macro and micro precision for evaluation of the classification of iris plants

  As it is known, the accuracy of a binary classifier is given by the VP/(VP+FP) rate

  - *Therefore, to determine macro accuracy, it is started by calculating this same rate for each of the species (for example, for the species 'setose', it is considered for VP the amount of plants that were correctly classified as setosas, and for FP the amount of plants that were wrongly classified as setose), and then the average of all these values (of all these rates) is calculated)*
  - *To determine Micro accuracy, the overall PV/(VP+FP) is calculated, considering for PV all plants that were correctly classified (regardless of species), and for FP all plants that were misclassified.*

    Note that the micro average of accuracy turns out to be equal to the micro mean, both sensitivity, accuracy (tx of hit) and F1 itself.

Examples of multi-class metrics

- Scikit-learn metrics intended for multi-class are supfixed with macro, micro, or weighted, alluding to the type of average they calculate. Here are some examples:
    - *AUC_macro – arithmetic mean of the AUCs of the classes*
    - *AUC_micro – AUC global, accounting for all true positives and all false positives, regardless of class*
    - *AUC_weighted – arithmetic mean of the AUCs of the classes, weighted by the number of true cases in each class*
    - *f1_score_macro – arithmetic mean of the F1 values of the classes*
    - *f1_score_micro – F1 overall, accounting for all true positives, false negatives and false positives, regardless of class*
    - *f1_score_weighted – arithmetic mean of F1, weighted by the number of occurrences of each class*
    - *precision_score_macro – arithmetic average of the precisions of the classes*
    - *precision_score_micro – overall accuracy, accounting for all true positives and all false positives, regardless of class*
    - *precision_score_weighted, – arithmetic mean of the precisions, weighted by the number of true cases of each class.*
- To realize what interpretation to give the concepts 'positive class' and 'negative class' in the calculation of multi-class metrics, let us retrieve the example shown from the confusion matrix
    - *The poorly classified example represents, at the same time, a false positive (FP) for the virginic species, and a false negative (FN) for the species versicolor.*

Unsupervised learning

- All ML methods we have studied so far were based on supervised learning techniques
- There are other methods, with prominence in other specific data analysis and exploration tasks, which are supported by unsupervised learning techniques
  - *These methods receive as input unclassified data (datasets without variable response)*
  - *Among them, the clustering method*

- The clustering method, also known as the grouping or data segmentation method, is used when we want to create a finite set of categories to segment a heterogeneous population into more homogeneous groups.
  - *It is a technique that allows us to group objects (observations) into homogeneous groups in relation to one or more common characteristics.*
    - In the ideal situation, the objects of each group should have the greatest similarity between them and the smallest between them and the objects belonging to other groups in order to ensure homogeneity within each cluster and heterogeneity between clusters.

Clustering

- Building a cluster involves selecting characteristics and similarity measures between objects
  - *Usually, the similarity measures used are traditional distance measures such as Euclidean.*
- Clustering is sometimes one of the first tasks to be performed when we want to carry out a data mining study, since it allows us to identify groups, from which other more targeted and detailed analyses can be performed, such as certain forecasts.
- In the case of an unsupervised method, initial clusters are not required to be defined at the time the algorithm begins,
  - *because it is not intended to associate with a specific cluster in each instance.*
  - *It is intended, on the contrary, to identify, from the initial set of instances, clusters that translate groups of similar instances.*

Examples of applying the Clustering method

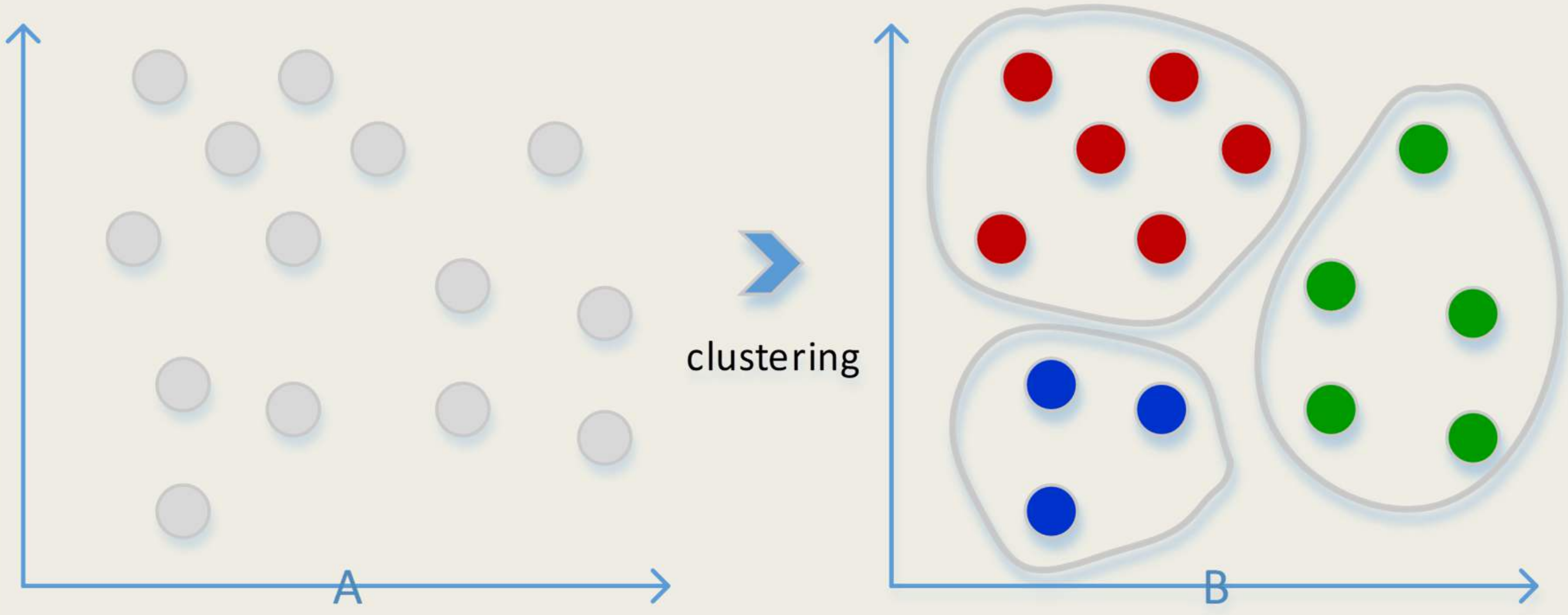
1. To know what different measures should be considered for a new garment intended to be produced
    - *the method will deal with segmenting individuals from a given population sample into several subgroups, each of which has individuals with similar specific characteristics (e.g., waist height and measurement)*
    - *then it will be the average characteristics (height and average waist measurement) of each group found that will serve as a reference for the various sizes of the garment to be produced*
2. Segment students with similar school achievement and behaviours, aiming to predict and analyse their learning processes into more homogeneous groups.
3. Segment students according to their personal and learning characteristics in order to develop and apply more personalized teaching methodologies.

The K-Means algorithm

- K-Means is one of the best-known clustering techniques
- Being applied to unclassified or labelled data (unsupervised method),
    - *aims to find K subgroups in the analyzed data, for a value of K provided*
- To form these subgroups (clusters), try to find k centroids that adequately represent the center of each of them:
    - *begins by "planting" K centroids, randomly distributed in the characteristic space*
    - *then itterily,*
        - associates to each of the centroids the instances closest to them,
        - adjusts each of the centroids to its new group of instances
    - *completing this process when all clusters stabilize, i.e. do not change (or equivalently when centroids do not change position)*

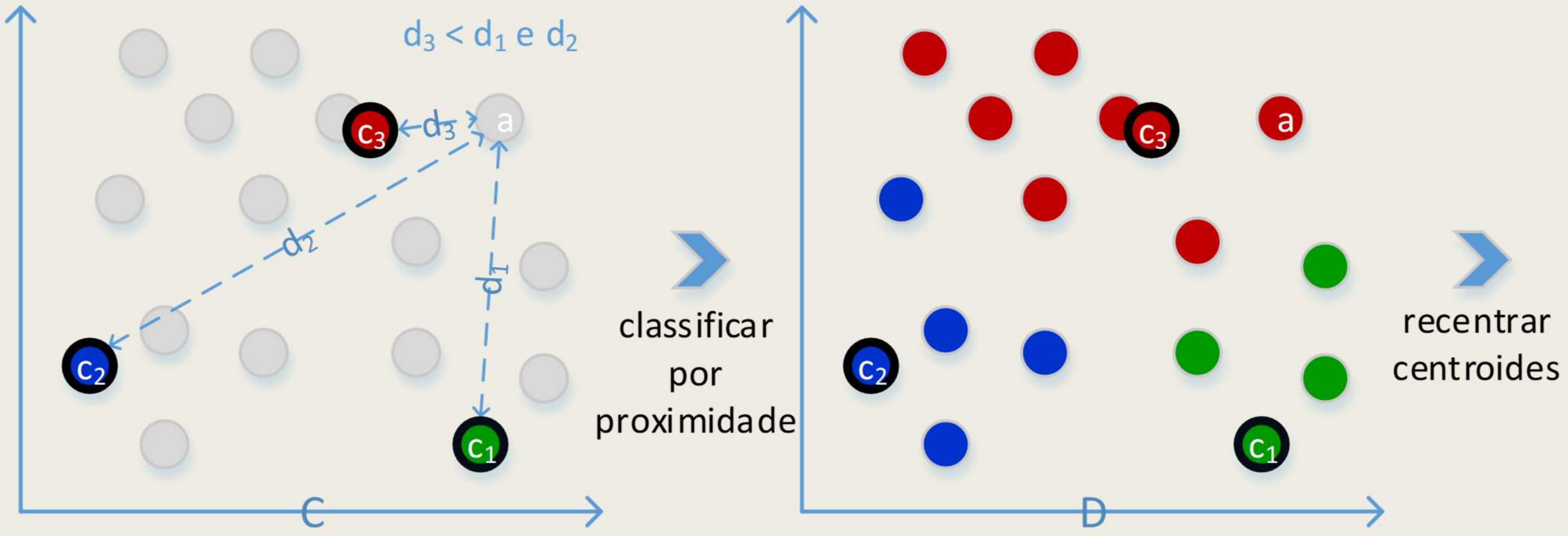
K-Means - Exemplifying for K=3

- Suppose we have a set of 2D points (represented observations/instances with two attributes)
    - *and that the goal is to group these points into 3 different clusters that allow us to discover, within each of them, a pattern that features them*
- The following figure illustrates a possible result when the goal is then to separate the data into 3 groups (k=3)



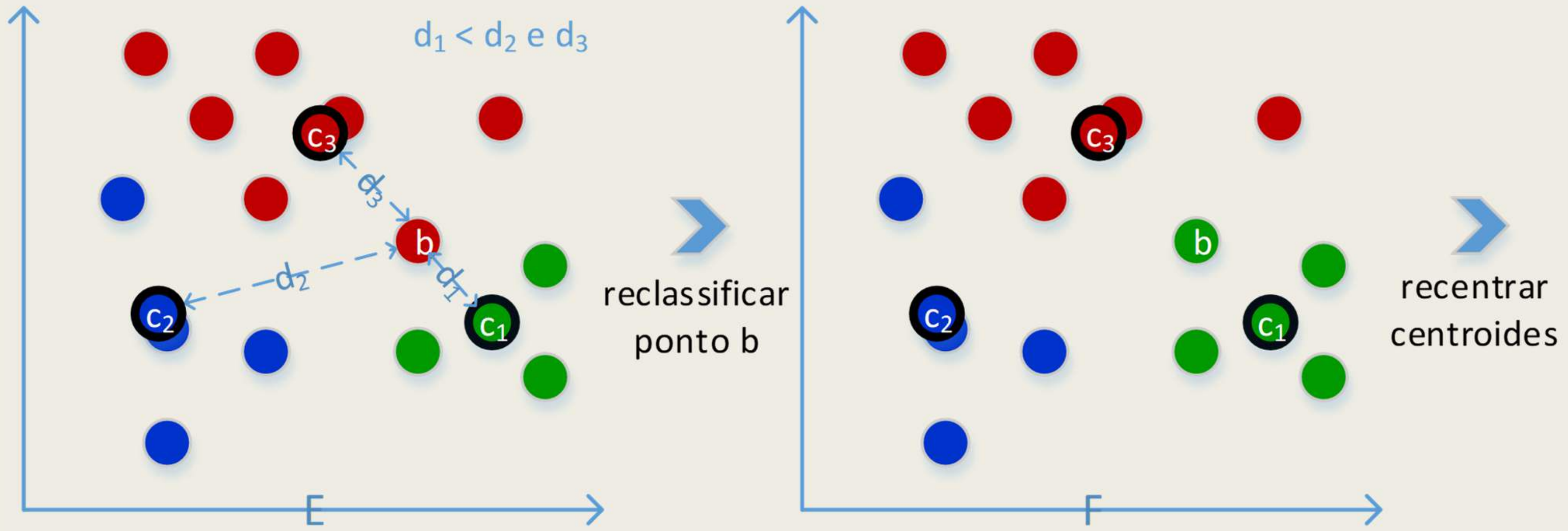
K-Means - Exemplifying for K=3

- The algorithm begins by randomly choosing 3 centroids (C_1, C_2 And C_3, on Graph C)
- Then, it is the distance of each point (instance) to each of the 3 centroids, in order to associate it with the nearest
    - *for example, as illustrated in Graph C, distance d3, point (a) to centroid C_3, is smaller than the distances d_1 And d_2, from that point to the centroids C_1 And C_2, Respectively*
        - therefore, point (a) will be classified as belonging to the Cluster 3
- Graph D shows the points after being associated with their respective centroids



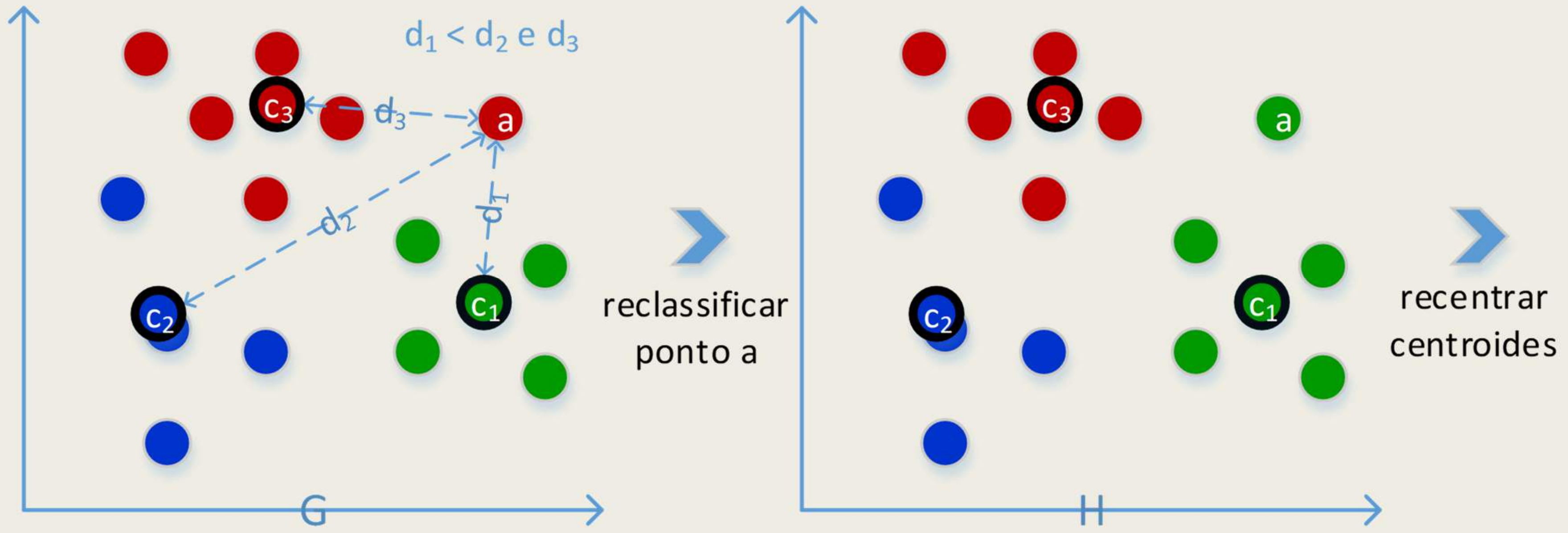
- Then all centroids will be repositioned, reflecting each of them the average (the geometric center) of all the points included in your cluster, resulting in the illustration of Graph E

K-Means - Exemplifying for K=3



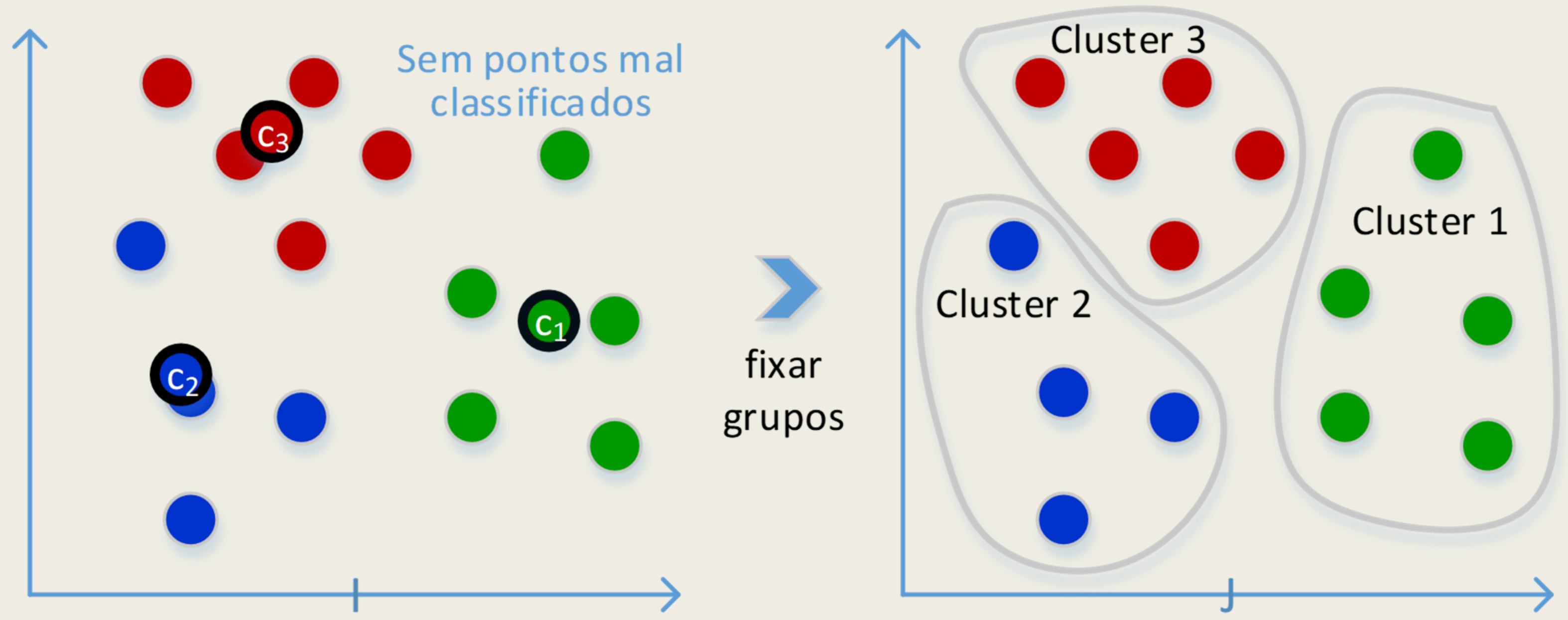
- Again, the distance from each point of the graph (instance) is measured to each of the 3 new centroids, to reclassify some that have become closer to the centroid of another cluster than to its
  - *in the example illustrated in Graph E, the distance d_1, from point (b) to centroid C_1,happened to be less than the distances d_2 and d_3, from that point to the centroids C_2 and C_3, respectively*
    - therefore, point (b) is reclassified from the Cluster 3 for the Cluster 1 (Graph F)
- As clusters 1 and 3 have changed, their centroids are recalculated, resulting in graph G illustration

K-Means - Exemplifying for K=3



- Now is the point (a), which due to the displacement of the centroids, became closer to the centroid of another cluster than to its
    - *Hence transit from Cluster 3 (Graph G) to Cluster 1 (Graph H)*
- The centroids are refocused again C_1 and C_3, as illustrated in Graph I

K-Means - Exemplifying for K=3



- As (although the centroids have moved a little) all the points are still effectively associated with the nearest centroids, the search for the 3 clusters illustrated in Graph J was completed.

Dimensionality reduction

- The goal of reducing the dimensionality of a dataset is to start representing it in a more compact way, reducing the number of predictive variables
  - *in a dataset with fewer columns, it becomes easier to apply sophisticated algorithms, which usually involve great computational effort*
  - *furthermore, algorithms, when applied to high-dimensional datasets, tend to overfit training data, thus compromising their generalization capacity.*
- If fewer dimensions can represent the most important information intrinsic to the data, why not try to exclude the excess dimensions?...
  - *Data reduction leads, however, to some loss of information,*
  - *but the use of more evolved algorithms can largely compensate for this loss*
- Dimensionality reduction can be achieved in different ways
  - *and the reduction of the dataset itself can also be achieved by reducing, by sampling, the number of observations (i.e., decreasing its size), rather than reducing dimensionality – in the background, eliminating rows from the dataset rather than columns.*

Dimensionality Reduction Methods

- Feature Selection

  only a few predictive variables are chosen, without involving any transformation of the same

  - *the information retained by the unselected columns is totally lost*
  - *there are several methods to make this selection, either manual or automatic or semi-automatic*

- Reduction by type transformation

  data are now represented in other formats

  - *for example, time series can be converted to multidimensional data of smaller size and complexity, using wavelet transformations*

- Projection of characteristics (or extraction of characteristics)

  project the data into a new smaller space

  - *Reduction by rotational transformation*

    correlations in the data are used to represent them in a smaller number of dimensions

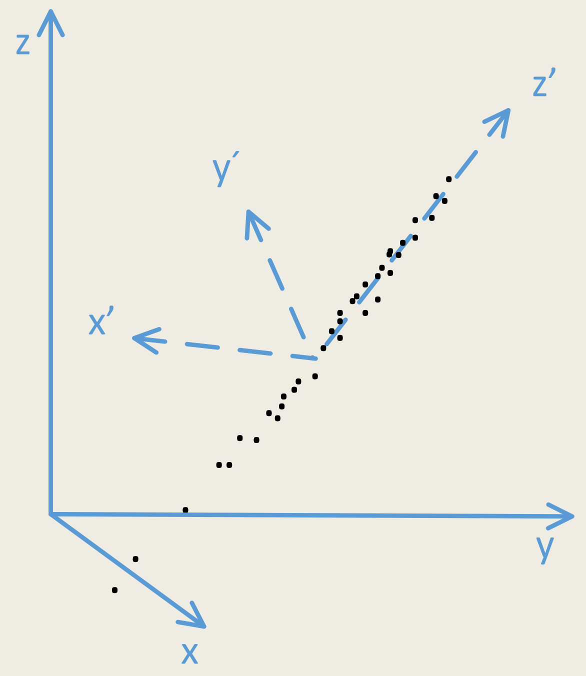
    - the information of all the initial columns will be somehow represented in the new columns, even if they are outnumbered
    - two important methods that make this type of reduction, and the one that we will pay more attention to, are the PCA (main component analysis) and the SVD (singular value decomposition)

  - *Reduction with manifolds (t-SNE method)*

Rotational Transformation

- In real datasets there is always some correlation between our predictive variables, which can even go unnoticed by the data analyst
  - *this correlation translates into redundant information that will penalize the efficiency of the learning algorithm*

- To better understand the effect of rotation on the data, notice the 3D data (3 explanatory variables, x, y and z) of the figure
  - *in this case, if a rotation is applied to the data that transfers it to the new x'y'z', there is a clear advantage, since the data can be (approximately) represented solely by one of the new coordinates: the variable z'*



- Now the important question is how can optimal rotation be determined
  - *which leads to greater removal of redundancies (correlations between variables) – which transitioned the data to the new x'y'z axis system, in the example of the figure*

The methods pursuing this objective are precisely the PCA and the SVD

- *Both integrate scikit-learn decomposition module (PCA and TruncatedSVD)*

PCA - principal component analysis

- PCA is the main method of dimensionality reduction
  - *It is based on a rationale that combines statistical calculation with linear algebra*
- Leaving our mathematical foundation to other more specialized studies, let us focus on the essentials of its operation:
  - *we don't need to indicate how many attributes the reduction will be*
  - *before applying the rotation transformation itself, the method begins by rotating the data set around the source of the axis system, subtracting from each of the observations the average value of the entire set*
  - *searches for the set of orthogonal directions that represent the initial data set*
  - *in general terms, the PCA seeks to map the original dataset to a new attribute space in which the column vectors of the new dataset become orthogonal to each other*
    - each of the new columns now represents, in descending order, a certain percentage of the variance of the data (as much as you can)

      (the perfect rotation would be the one that could represent 100% of the variance in a single column)

Use of Kernel functions in the PCA method

- The PCA is, by its nature, a linear method,
  - *which means we have difficulty dealing with data that is not linearly separable*
  - *and that's the case with most of the datasets that we deal with in the real world*
- One way to capture the nonlinearity of the data is to apply some kind of nonlinear transformations to that data and only then reduce its dimensionality by the PCA method
  - *these transformations are ensured by* Kernel functions, *which project the data into another dimensional space*
- Examples of kernel functions used:
  - *Polynomial*
  - *radial base function*
  - *sigmoidal*
  - *Cosine*
- Transformations with Kernel functions can be effectively useful in separating data
  - *but can also contribute to the overfitting*

    (this effect can be controlled and minimized with training-validation)
- It is recommended to normalize the data before applying the PCA

SVD - decomposition into singular values

- In the SVD method, the data matrix (representing the dataset), is decomposed into the product of 3 arrays with special features that make them easier to analyze and manipulate.
- The SVD method is quite similar to the PCA,
  - *except that factorization in the SVD is done on the data matrix and not on the covariance matrix*
  - *and unlike the PCA, can therefore be applied to sparse matrices*

- Truncated SVD

  The truncated SVD difference from normal SVD is in the number of columns resulting from
  - *while normal SVD produces arrays with n columns, with no number of attributes of the problem,*
  - *truncated SVD produces arrays with a specific number of columns, so that you can reduce dimensionality.*

Dimensionality reduction with manifolds - t-SNE

- It is a set of algorithms that allow us to map multidimensional data to a space with fewer dimensions (usually 2) to be easily graphically visualized
- One of these algorithms, particularly interesting, and which has gained increasing popularity, is the t-SNE (t-distributed stochastic neighbour embedding)
    - *calculates a new representation of training data (we can only transform the data we are trained for, not the test data)*
    - *useful for exploratory data analysis, and is rarely used in supervised learning*
    - *tries to find a two-dimensional representation of the data that preserves the distances between the points*
    - *starts with a random two-dimensional representation for each data point (instances, dataset observations) and then tries to approximate the points that are nearby in the original space and move away from the points that are farther away in that space*
- It is computationally very demanding compared to the other methods
    - *Therefore, when the dataset reaches a high dimensionality (above a few dozen columns), it is recommended that you start first by reducing PCA or SVD*

Solve exercise #26

from the book of exercises