Machine learning in healthcare: HW3

Hadassa Malka, 13-15.01.2021

1) Clustering:

a) In the K-medoids algorithm, the goal is to find the K medoid points (representative points of K clusters of our dataset) which minimizes the sum of L1 distances between them and their assigned data points (non-medoid points of their respective cluster). Meaning, only actual data points can be considered as the cluster representative, differing from K-means algorithm in which the representative (centroid) is any point of the space which minimizes the sum of L2 distances between them and their assigned data points (of respective cluster). We can then understand that:

- If the dataset presents outliers with large values, K-medoid will still take an existing data point in the same dataset to reduce the sum of L1 distances between the large outlier and the medoid; whereas the K-means algorithm may take a value (centroid) which is relatively far from the other points of the cluster in order to minimise the distance between them. In that sense, K-means is more sensitive to outliers, making **K-centroid more robust to outliers** (noise).

- Let be p_1, p_2 points of n dimensions in the dataset. K-centroid uses the L1 (Manhattan) distance between two points, defined by:

$$d_{L1}(p_1, p_2) = \sum_{i=1}^{n} |p_{1_i} - p_{2_i}|$$

K-means uses the L2 (Euclidean) distance:

$$d_{L2}(p_1, p_2) = \sqrt{\sum_{i=1}^{n} (p_{1_i} - p_{2_i})^2}$$

Regarding outliers, we will prefer using the **L1 norm as metrics because it shows more robustness to them** (as a higher norm parameter will increase the sensibility to noise, due to the fact that we take the square of the distances of each axis, meaning we can have greater values for the distance while having high values of outliers in some dimensions).
Paper used to support this point: *"On the Surprising Behavior of Distance Metrics in High Dimensional Space",* by Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Kiem (2001).

Overall, we can conclude that **K-centroids is more robust to outliers than K-means.**

b) We would like to prove that for the 1D case of K-means, the centroid $\mu_{optimal}$ which minimizes the term $\sum_{i=1}^{m}(x_i - \mu)^2$ is the mean of m examples.

$$\mu_{optimal} = \arg_{\mu} \min \left(\sum_{i=1}^{m} (x_i - \mu)^2 \right)$$

Let's derivate the expression $g(\mu) = \sum_{i=1}^{m}(x_i - \mu)^2$ according to μ:

$$\frac{\partial}{\partial \mu} \left(\sum_{i=1}^{m} (x_i - \mu)^2 \right) = 2 \sum_{i=1}^{m} (x_i - \mu)$$

Let's find the second derivative according to μ, in order to be sure that $\frac{\partial g(\mu)}{\partial \mu} = 0$ will give us a minimum for μ.

$$\frac{\partial^2}{\partial \mu^2} \left(\sum_{i=1}^{m} (x_i - \mu)^2 \right) = -2 \sum_{i=1}^{m} 1 \rightarrow \frac{\partial^2 g(\mu)}{\partial \mu^2} = -2m \rightarrow \frac{\partial^2 g(\mu)}{\partial \mu^2} < 0$$

$$\rightarrow \frac{\partial g(\mu)}{\partial \mu}|_{\mu=\mu_0} = 0 \leftrightarrow \mu_0 \; is \; a \; minimum \; of \; g(\mu)$$

Therefore:

$$2 \sum_{i=1}^{m} (x_i - \mu_{optimal}) = 0 \rightarrow \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} \mu_{optimal} \rightarrow \sum_{i=1}^{m} x_i = m \cdot \mu_{optimal}$$

$$\rightarrow \boldsymbol{\mu_{optimal} = \frac{\sum_{i=1}^{m} x_i}{m}}$$

As requested, **in the 1D K-means, the centroid $\boldsymbol{\mu_{optimal}}$ which minimizes the term $\boldsymbol{\sum_{i=1}^{m}(x_i - \mu)^2}$ is the mean of $\boldsymbol{m}$ examples.**

c) We would like to prove that for the 1D case of K-medoids, the centroid $\mu_{optimal}$ which minimizes the term $\sum_{i=1}^{m}|x_i-\mu|$ is the median of m examples.

$$\mu_{optimal} = \arg_{\mu}\min\left(\sum_{i=1}^{m}|x_i-\mu|\right)$$

The expression $g(\mu)=\sum_{i=1}^{m}|x_i-\mu|$ is piecewise differentiable according to μ (not differentiable where each term is zero, but we can take the derivative at each part zero, and consider zero as part of the increasing function). Let's write for comfort:

$$G_1 = \{indices\ i\ for\ which\ \mu \geq x_i\}$$
$$G_2 = \{indices\ i\ for\ which\ \mu < x_i\}$$

$$\frac{\partial}{\partial\mu}\left(\sum_{i=1}^{m}|\mu-x_i|\right)=\sum_{i=1}^{m}sign(\mu-x_i)=\sum_{i\in G_1}1+\sum_{i\in G_2}-1=|G_1|-|G_2|$$

An absolute value of a linear function is convex (a decreasing linear function followed by its symmetric increasing linear function), and since the sum of convex functions give a convex function, $g(\mu)$ is a convex function, admitting a minimum where $\frac{\partial}{\partial\mu}(\sum_{i=1}^{m}|\mu-x_i|)=0$

$$\frac{\partial}{\partial\mu}\left(\sum_{i=1}^{m}|\mu-x_i|\right)=0\leftrightarrow|G_1|=|G_2|$$

In other words, the μ which will minimize the cost function $g(\mu)$ is the member of the dataset which will bring the same number of elements in $|G_1|$ and $|G_2|$, meaning **the value in the dataset for which there will be the same number of items $\boldsymbol{x_{i\in G_1}(\geq\mu)}$ and $\boldsymbol{x_{i\in G_2}(<\mu)}$**, and that is exactly the definition of the median.

Overall we get:

$$\boldsymbol{\mu_{optimal} = median(\{x_i\}_{i=1}^{m})}$$

2) SVM:

We would like to assign each image to its respective setting. Each figure represents a set of points from two different classes (the purple and the blue dots), according to the value of their two features x_1, x_2.

a) Linear kernels:

For a 2D data, we expect a linear kernel setting to give a straight line (linear combination of the features in their space) as the hyperplane of the problem (separating the two classes). So the corresponding pictures would be A and D. For this type of kernel, the hyperparameter C is a trade-off between the maximization of the margins (between the hyperplane and the closest points) and the correctness of the training examples classification.

The higher is C, the higher we put importance on classifying correctly the samples at the expense of accepting smaller margins (a too high C can lead to overfitting).

On the opposite, the smaller is C, the higher we put importance on increasing the margins at the expense of correctness rate for the classification, also known as accuracy (a too small C can lead to underfitting), and at the expense of classifying some data points inside the margins as well.

Based on those explanations, we can notice that in the picture A, the hyperplane chosen increases the margins but admits data points inside the margins (we can see them touching the line in the red class) therefore **A corresponds to the smaller C** (C=0.01 in configuration 1.). Hence, the picture **D corresponds to the higher C** (C=1 in configuration 2.), since the hyperplane increases the margin without admitting any data points inside them / misclassifications.

b) RBF kernels:

RBF kernels aim to generate new features (based on the original ones) using the distance between a center and their respective data points. Therefore, such a transformation will give a hyperplane with clusters shape (the decision boundary function will be a contour around a specific class).

That said, we can understand that type of kernel corresponds to the pictures B and E. Now, the hyperparameter used is γ, which determines how far from the decision boundary hyperplane we want a single training sample to reach.

For a high value of γ, we would like to have a hyperplane which maximizes the margins while taking in consideration only the closest data points, meaning the data points close to the boundary condition will have a lot of weights in order to

balance in the shape of the curve to their benefits, in term of reducing the misclassifications (we expect then to have a more wiggled hyperplane, more fitted to points of the class).
On the opposite, for a low value of gamma, we would like to also take in consideration the data points which are farer from the hyperplane (putting less weight on the closest points). We then expect the hyperplane to be smoother, less wiggled than with using a higher value of gamma.

Considering pictures B and D, we can see that the decision boundary in B is way more wiggled, puts more weights in maximizing the margins for the closest points than in D, therefore we can conclude that **the highest γ value corresponds to the picture B** ($\gamma = 1$, meaning setting 6.) while **the lowest γ value corresponds to the picture E** ($\gamma = 0.2$, meaning setting 5.), which is smoother and puts less weight to the importance of maximizing the margins between the closest points and the hyperplane.

c) Polynomial kernels:

Polynomial kernels aim to generate new features using a polynomial relation between the original features. Therefore, besides the fact that we have only two pictures left, we expect the boundary function to have a polynomial shape which is the case in the pictures C and F.

The higher is the dimension of the polynomial kernel, the more our boundary function will have a complex shape (in terms of variation of the slope direction. Meaning we expect a higher number of fixed points). For a degree of two, we expect the hyperplane to have a slope which is always increasing or always decreasing (characteristic of polynomial function of degree 2, as its derivative is linear), which is the case in **picture C** (setting 3.). We remain with a more complex shape for **picture F, which corresponds to a 10^{th} order polynomial kernel** (setting 4.).

Overall, we get the following matches:

- $\mathbf{1} \leftrightarrow \boldsymbol{A}$
- $\mathbf{2} \leftrightarrow \boldsymbol{D}$
- $\mathbf{3} \leftrightarrow \boldsymbol{C}$
- $\mathbf{4} \leftrightarrow \boldsymbol{F}$
- $\mathbf{5} \leftrightarrow \boldsymbol{E}$
- $\mathbf{6} \leftrightarrow \boldsymbol{B}$

3) Capability of generalization:

We would like to study a new criterion for choosing a parsimonious model in GMM called Akaike Information Criterion (AIC).

a) *"Everything should be made as simple as possible but not simpler"*:

The famous quote of Einstein corresponds to the **generalization** of models in machine learning aspects. Indeed, that term refers to the ability of a model to adapt to every new situation (data) it has not seen before. If the model was trained on a specific train set, it should also work on every other set different for the training one, avoiding overfitting to the training set (in other words, it should be made as simple as possible to adapt to any other set). But, if the model is too simple (meaning not working well on any set), it means that it is underfitting (in other words, it shouldn't be made too much simple).

b) Terms of AIC:

The formula of AIC is given by:

$$AIC = 2p - 2 \cdot \ln(\hat{L})$$

Let's study the impact on each of the term on generalization of machine learning models:

- $2p$: corresponds to twice the number of learned parameters. As this term increases, it means we are increasing the number of parameters learnt on a specific set, meaning it will lead to fit more and more the model on the set (if this term is too big, we talk about overfitting, the model will be too complex).
  We want this term to be as small as possible.

- $2 \cdot \ln(\hat{L})$: corresponds to twice the natural logarithm of the estimated likelihood of the model on the training set given the p parameters. As the likelihood decreases, it means we less and less succeed in fitting the model on the specific dataset; the model becomes more and more simple, and if the term is too small, we talk about underfitting the model on the set; the model is too simple.
  We want this term to be as big as possible.

In terms of generalisation, we want to find a trade-off (balance) between the number of parameters used (a big number increases the complexity of the model) and the log-likelihood of the model on the dataset using those parameters (a small number decreases the complexity of the model).

c) Violation of the balance:

If the balance between a too much or too simple model is violated, we get either one of the following option:

- **Underfitting:** Happens if the model is too much simple. In that case, we have a too small log-likelihood of the model on the training set using the parameters, meaning it does not succeed in fitting well the model for the problem.

- **Overfitting:** Happens if the model is too complex. In that case, the number of parameters is too high, leading to overfit the model on the training set.

d) Choice of AIC:

Given the definition of AIC, we can understand that we want to lower the number of parameters (hence lowering the complexity) while increasing the log-likelihood estimation (hence avoiding too much simplicity).

Therefore **we would like to take the AIC which is the lowest** (small $2p$ and high $2 \cdot \ln(\hat{L})$).