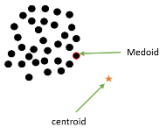
Ameer Lawen 316184456

Machine Learning in Healthcare
Homework #3

1. Clustering:

a. K-medoid clustering is a more robust algorithm to noise compared with K-means clustering, as both try to find clusters in the data in an unsupervised manner based on the geometry and the determined metric used to define vicinity. As we learned, K-means algorithm iteratively: computes centroids, assign points based on their closest computed centroid and repeats again till convergence. The centroids therefore most definitely will not be included in the set of given points, so it is a generated parameter that might be influenced critically by the nature of a cluster, mainly when the data is considerably noisy and/or contains significant outliers, so the centroid, which is the reference point of each cluster, might be influenced by noise. On the other hand, K-medoid's cluster reference must be a sample in the cluster. Even though there might be noise and outliers in a given dataset, an obvious cluster will not be affected severely by extreme samples, outliers will not bias sharply the center of interest of a cluster (outliers are not as many as standard data, otherwise outliers will not be considered outliers but another distinct cluster) because the medoid is a part of the cluster, so in order to lower the 'Energy' of the entire system, the medoid of a noisy cluster will be among the 'main' subgroup of the cluster and never will be away of this main subgroup, thus not affected severely even by an extreme outlier. Qualitative visual elaboration:



If a centroid is biased due to outliers, the main cluster might be partitioned if there is another adjacent cluster. On the other hand, we can see that the medoid although be biased, but still remains in the adjacency of the main cluster because it can't obtain a value that is not among the data values, and it shall be in the adjacency of the main cluster not the noise/outliers to minimize the cost function because grouped outliers are sparse. Thus k-medoid is more robust to noise. One more perspective, L1 penalty gives less weight to distant points than L2 penalty, so distant points less affect the loss function, in a way we can look at it as 'less overfitting' in a case of distant points.

b. Pertaining the 1D special case of this algorithm, we will try to prove that the centroid that minimizes the loss term is the mean of m examples:

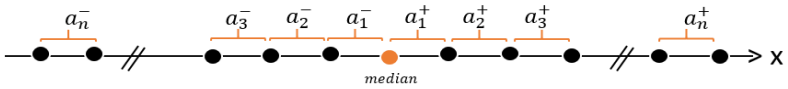
$$\min_{\mu}\left\{\sum_{i=1}^{m}(x_i-\mu)^2\right\} \rightarrow$$

$$0=\frac{d}{d\mu}\left(\sum_{i=1}^{m}(x_i-\mu)^2\right)=\sum_{i=1}^{m}-2\cdot(x_i-\mu)=-2\cdot\sum_{i=1}^{m}x_i+2\cdot\sum_{i=1}^{m}\mu$$

$$0=-2\cdot\sum_{i=1}^{m}x_i+2\cdot m\cdot\mu, \quad as\ \mu\ is\ constant$$

$$\mu=\frac{\sum_{i=1}^{m}x_i}{m}\ which\ is\ the\ definition\ of\ mean\ of\ x$$

c. Let us assume for simplicity and without loss of generality, that we have a 1D dataset, and the dataset has odd number of samples, $m = 2n + 1$. We aspire to show that the loss function is minimized when μ is the median (part of the samples).



The spacings between the points are shown and matched above, for simplicity, a_j^+ is the spacing between the $(j-1)^{th}$ and j^{th} ($x_{j-1}^+ \; and \; x_j^+$accordingly) ordered samples that are larger than the median, and a_j^- is the space between the $(j-1)^{th}$ and j^{th} ordered samples that are smaller than the median. The loss term in this state is:

$$J_{\mu_{med}} = \sum_{i=1}^{m} |x_i - \mu_{median}| = \sum_{j=1}^{n} (x_j^+ - \mu_{median}) - \sum_{j=1}^{n} (x_j^- - \mu_{median}) =$$

$$= (\boldsymbol{n \cdot a_1^+ + (n-1) \cdot a_2^+ + \cdots + 2 \cdot a_{n-1}^+ + a_n^+})$$
$$+ (\boldsymbol{n \cdot a_1^- + (n-1) \cdot a_2^- + \cdots + 2 \cdot a_{n-1}^- + a_n^-})$$
$$\triangleq \boldsymbol{\Delta^+ + \Delta^-}$$

Let us try to substitute μ as the nearest sample on the left side of the median (x_1^-) *without loss of generality*, the loss value will be:

$$J = \sum_{i=1}^{m} |x_i - x_1^-| = (\mu_{median} - x_1^-) + \sum_{j=1}^{n} (x_j^+ - x_1^-) - \sum_{j=2}^{n} (x_j^- - x_1^-) =$$

$$= a_1^- + \sum_{j=1}^{n} \left(x_j^+ - (\mu_{median} - a_1^-)\right) - \sum_{j=2}^{n} \left(x_j^- - (\mu_{median} - a_1^-)\right)$$

$$= a_1^- + \sum_{j=1}^{n} (x_j^+ - \mu_{median} + a_1^-) - \sum_{j=2}^{n} (x_j^- - \mu_{median} + a_1^-)$$

$$= a_1^- + n \cdot a_1^- + \sum_{j=1}^{n} (x_j^+ - \mu_{median}) - (n-1) \cdot a_1^- - \sum_{j=2}^{n} (x_j^- - \mu_{median})$$

$$= 2 \cdot a_1^- + \sum_{j=1}^{n} (x_j^+ - \mu_{median}) - \sum_{j=2}^{n} (x_j^- - \mu_{median})$$

$$= 2 \cdot a_1^- - a_1^- + \sum_{j=1}^{n} (x_j^+ - \mu_{median}) - \sum_{j=2}^{n} (x_j^- - \mu_{median}) + a_1^-$$

$$\{as\ a_1^- = -(x_1^- - \mu_{median})\} \rightarrow = a_1^- + \sum_{j=1}^{n} (x_j^+ - \mu_{median}) - \sum_{j=1}^{n} (x_j^- - \mu_{median})$$

$$\therefore \boldsymbol{J_{x_1^-} = a_1^- + \Delta^+ + \Delta^- = a_1^- + J_{\mu_{med}} > J_{\mu_{med}}}$$

Inductively, changing the reference stepwise to the left (or to the right, of the median) brings about larger loss value gradually. Therefore, the medoid that minimizes the loss term is the median of m (odd) samples. For even number of examples, the same procedure can be applied, and the medoid must be in the center of the ordered samples to minimize loss, although in some cases the median is set to be the mean of the two central nodes in the ordered set of examples, which produce a value that isn't included in the dataset, the reference example that minimizes the loss function relies on the distribution of the data. It might be the $\left(\frac{n}{2}\right)^{th}$ *or* $\left(\frac{n}{2}+1\right)^{th}$ example, and by induction it can be proven similarly.

2. SVM:

Classification based on SVM algorithm is a very potent approach and can be optimized by altering the kernel which is the 'metric' of the system. The algorithm will try to maximize the margins between the different classes, based on the kernel and the hyper-parameter C which defines the user's tolerance for misclassification (degree of overfitting). Now we will do matching between the figures and the listed settings.

Figure	Setting
A	*1*
B	*6*
C	*3*
D	*2*
E	*5*
F	*4*

- Elaborations:
  - For setting 1 and 2 the corresponding figures are A and D accordingly as stated above. It is obvious that the border applied by the algorithm between the two groups is a linear border, thus the kernel for both cases is a linear kernel. Now in order to differentiate and match correctly, it is possible to notice that in figure A., the defined border is set with higher proximity to points, mainly the purple points. A border that is not deterred by the adjacent points even though it is possible to set another 'conservative' border (i.e. in figure D.) indicates that the algorithm is relatively more tolerable to misclassifications in the training set thus there is less 'overfitting' and more generalization, meaning that between figure A. and D., the hyper-parameter C of the classifier in figure A. is lower than the hyper-parameter C of the classifier in figure D. as we see in figure D., the border is more complied to every point and tries to maximize the margins even for distant points of the main cluster, thus it over-fits the data meaning that the hyper-parameter C is relatively large.
  - For setting 3 and 4 the corresponding figures are C. and F. accordingly as stated above. For polynomial kernels, we expect to get segregation of the data by non-linear borders, but also we expect not to get closed loops/enclosed areas. For figure C. It is relatively straight forward to recognize that the separator is 2^{nd} order polynomial due to its simple non-linearity, for figure F. we can see that the borders are quite complex, segregating without obtaining enclosed areas, so it is not a section of a Gaussian plot thus the kernel is not RBF kernel. The remaining setting is 10^{th} order polynomial kernel which implies a complex behavior of border (setting 4) a behavior that we observe in figure F.
  - For setting 5 and 6 the corresponding figures are E. and B. accordingly as stated above. Other than elimination of figures A., C., D. and F., we are searching for borders that are obtainable by doing a 'section' for a 3D Gaussian mixture, which dictates to get a form of enclosed spaces or 'islands' due to the morphology of a Radial basis function/Gaussians. Figure E. and B. apply for this criteria. We can see that in figure B. the blue cluster is more tightly enclosed and overly-fitted to the training set (the observed data). On the other

hand, figure E. represents a more general border, less tightly-fitted to the data. Relative over-fitting is dictated by relatively higher value of the hyper-parameter γ, because γ is inversely related with the variance of the Gaussian distribution. Meaning that higher γ implies lower variance thus less generalizability and more over-fitting. For that, setting 6 (higher γ) is matched with figure B. and therefore setting 5 is matched with figure E.

3. Capability of generalization:

a. The scientific term of the balance that Einstein meant to in Machine Learning aspect is the Bias-Variance tradeoff. To clarify this phenomenon, we aim to dissect the two extremities of such tradeoff. On the first hand, if a model perfectly fits the training set, it is generally more complex and requires adjustment and learning of large number of parameters. It would be over-specific and would lack generalizability and reproduction of such performance on the test set or other futuristic data. In other words, it has low predetermined bias but high variance. On the other hand, a less complex model has high bias and low variance, which means the model is more stiff and less prone to variance in the data, it is relatively easier to learn due to the small number of parameters which are meant to be adjusted. It may be easy to be constructed and generalization is more effective, but it may lead to under-fitting. In other words, it may be not as simple as possible, but simpler, in Einstein's lexicon. Risking its performance to be near-chance level.

b. In order to asses how each of the two terms in AIC affect the balance stated in a. we should look at AIC mathematic definition:

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

- The parameter- p, represents the total number of learned parameters. As stated above, a more complex model requires relatively large number of parameters to be learned based on the training set, which reduces the models bias but increases its variance. Large p has a higher probability to over-fit the data and to be detrimental to generalization.
- The parameter- $\hat{L}$, represents the estimated likelihood. The higher the L the more likely the classifications results brought about by the model to match the learned class, meaning that the model better fits the data, with a risk for over-fitting if not controlled tightly. The natural logarithm is an ascending and injective function meaning that it would not affect the trend of L, but adding the minus sign adjusts this transformation, as likelihood measure ranges between 0 and 1, so $-2\ln(\hat{L})$, decreases as likelihood increases.

c. The bias-variance unstable balance is critical to be satisfied. If it gets violated, as stated above, it may lead to over- or under-fitting of the data. Violation of the balance is detrimental for generalization if the bias decreases and variance increases (over-fitting), or detrimental for learning and satisfactory conclusions if the bias increases and variance decreases (under-fitting).

d. Ideally, we want our model to best fit the data while preserving its simplicity. For clarification, at first, we shall understand the range that $-2\ln(\hat{L})$ occupies:

$$\hat{L} \in [0,1] \rightarrow$$

$$we\ shall\ define\ T(\hat{L})\ as\ the\ transformation\ of\ \hat{L} \rightarrow$$

$$T(\hat{L}) = -2 \cdot \ln(\hat{L}) \rightarrow$$

$$\therefore T(\hat{L}) \in [0, \infty)$$

Meaning that, $\lim_{L \to 0} T(\hat{L}) = \infty\ and\ \lim_{L \to 1} T(\hat{L}) = 0$. So as likelihood increases, this measure (T(L)) decreases drastically in a logarithmic manner, and AIC will be affected dramatically. After this elaboration, we will further dissect AIC's behavior.

Let us examine the extreme states of over-fitting and under-fitting and asses how this meter is affected accordingly:

- In the case of over-fitting, there are many learned parameters, therefore p is relatively large, but on the other hand, in over-fitting, the likelihood measure on the training data is large (hence over-fitting), leading for T(L) to be nearly 0.
- In the case of under-fitting, there are few learned parameters, therefore p is relatively small, but on the other hand, in under-fitting, the likelihood measure on the training data is away from being perfect if the dataset is complex (remember, bias is large), leading T(L) to be relatively high.

  Either ways, AIC would get high values in case of violation of the balance (because of high p or low L). Finally, in order to find the best balance, a model should contain low p and high L, meaning that it mustn't over-learn/over-fit, thus p should be small, and ideally, likelihood measure must be high for good fit, meaning that T(L) tends to zero (minimization of T(L)), these two combination imply that in order to improve a model, we should minimize it's AIC measure or alternatively use AIC measure for model selection- the best model based on this criteria has the lowest AIC measure.