Shahar Rashty 312465305

HW3 - ML in HealthCare

1.1 Clustering

a. Prove that the median minimizes the term $\sum_{i=1}^{m}|x_i - \mu|$

Let us sort the x values such that :

$$x_1 < x_2 < \cdots < x_m$$

Median - is the value separating the higher half from the lower half of a data sample

Since we sorted our terms we can say that $x_{\frac{n+1}{2}}$ is the median!

Given that $\mu \in [x_1, x_m]$

Assuming m is odd,

$$f(\mu) = \sum_{i=1}^{m}|x_i - \mu|$$

To minimize this tern let's differentiate it

$$f'(m) = -\sum_{i=1}^{m} sign(x_i - \mu) = \sum_{1}^{m}\begin{cases} 1\ if\ (x_i - \mu) > 0 \\ -1\ if\ (x_i - \mu) < 0 \end{cases}$$

$$= 1 * len\big((x_i - \mu) > 0\big) - 1 * len\big((x_i - \mu) < 0\big)$$

Since we want to minimize f(mu) we will demand that the derivative shall be equals to zero,

Hence we got :

$$f'(\mu) = 1 * len\big((x_i - \mu) > 0\big) - 1 * len\big((x_i - \mu) < 0\big) = 0$$

Which means we must have the same number of Xs before and after the index of the position of the Xi that satisfy the condition $(x_i - \mu) > 0$ for the first time, which means that $\mu = x_{\frac{n+1}{2}}$ – the middle index at our sorted population, hence μ must be the median !

The term we are trying to minimize is a measure of how far away each data point is from μ. The goal is to find a value that minimizes this sum, meaning that it is the closest possible value to all the data points. the median is the value that minimizes the sum of absolute differences because it separates the data set into two equal halves and it is equidistant from the maximum and minimum values, making it the closest possible value to all the data points.

A sufficient condition for the median to be a minimum of the loss function (the sum of absolute differences between a given value and each data point) is that the data set is having odd number of examples ,this way we can assure that the median of the population is part of the population , if we had even number of

examples the median will be the mean of the 2 middle values and therefore cannot be chosen as a medoid in K-medoids algorithm.

b. To prove that the centroid μ minimizes the term $\sum_{i=1}^{n}(x_i-\mu)^2$, we will differentiate this term and compare to zero:

$$f'(\mu)=\frac{d(\sum_{i=1}^{n}(x_i-\mu)^2)}{d\mu}=0$$

$$\sum_{i=1}^{n}-2(x_i-\mu)=0$$

$$-2\left(\sum_{i=1}^{n}x_i-n\cdot\mu\right)=0$$

$$\sum_{i=1}^{n}x_i=n\mu$$

$$\frac{\sum_{i=1}^{n}x_i}{n}=\mu$$

$$f''(\mu)=2n>0\rightarrow minima$$

Which is the definition of the mean of n examples.

c. Yes, the K-medoids is more robust to noise than the K-means algorithm. K-means and K-medoids are both clustering algorithms that aim to divide a dataset into k clusters, where each cluster is characterized by a centroid or medoid. The main difference between the two algorithms is the way in which the centroid or medoid is defined. In K-means, the centroid is a point that is calculated as the mean of all the points in a cluster, while in K-medoids, the medoid is a point that is part of the dataset and is chosen as the one that minimizes the dissimilarity between itself and all the other points in the cluster.

The use of medoids instead of means makes K-medoids more robust to noise and outliers, as the medoid is less sensitive to these elements than the mean. Therefore, K-medoids is considered to be more robust to noise than K-means. However, K-medoids algorithm tend to be more computationally expensive than K-means as we are comparing all points in the cluster to find the medoid.

1.2 SVM

Support Vector Machines (SVMs) is a type of supervised machine learning algorithm that can be used for classification and regression tasks. The main idea behind SVMs is to find the best boundary, or hyperplane, that separates different classes in the dataset. The boundary is chosen in such a way that it maximizes the margin, or the distance, between the boundary and the closest data points from different classes, called support vectors.

For figures A and D, the decision boundary is linear, hence they can only match the settings 1 or 2.

The hyperparameter C means-" How much do you penalize" . C controls the trade-off between maximizing the margin (the distance between the decision boundary and the closest training data points) and minimizing the misclassification errors. A smaller C value results in a larger margin, but allows for more misclassification errors. A larger C value results in a smaller margin, but allows for fewer misclassification errors. So, with C1=1 and C2=0.01, C2 will result in a larger margin as it is a smaller value and allows more misclassification errors .

The margin in figure A is smaller than the margin in figure D , so A match the larger C ,

A=2 and D=1.

The Radial Basis Function (RBF) kernel is a popular kernel function used in Support Vector Machines (SVMs) to transform the input data into a higher-dimensional feature space. The RBF kernel maps the input data into a space where it becomes linearly separable, allowing the SVM to find a non-linear decision boundary.

Gamma, kernel coefficient for RBF ,hyperparameter that means " How much we fit the training data"-. The higher Gamma the more we fit the training data. So too high of a Gamma can cause overfitting.

figure B tends more to overfit and is more complex than in figure E – hence B must have larger gamma than E.

E=5 and B=6.

For the Polynomial kernel, the higher the order of the polynomial kernel the better the model fits the data – high order will tend to overfitting.

In figure F we can see that the decision boundary is very complex and overfitting the data , thus it matches a higher order polynomial kernel(10th order),

we can conclude that **F=4**.

2nd order polynomial kernel can be represented as parabolic decision boundary, thus, the only figure that the decision line matches this, is figure C.

C = 3

1.3 Kernel function

$$< \phi(x1)^T * \phi(x2) \geq \left(\phi_{1(x1)}, \phi_{2(x1)}, \phi_{3(x1)} \dots\right)^T * \left(\phi_{1(x2)}, \phi_{2(x2)}, \phi_{3(x2)}, \dots.\right) =$$

$$= \phi_{0(x1)} * \phi_{0(x2)} + \phi_{1(x1)} * \phi_{1(x2)} + \phi_{2(x1)} * \phi_{2(x2)} + \cdots =$$

$$= \left(x_1^0 * e^{-\frac{x1^2}{2}} * x_2^0 * e^{-\frac{x_2^2}{2}} + \frac{x_1^1}{\sqrt{1!}} * e^{-\frac{x1^2}{2}} * \frac{x_2^1}{\sqrt{1!}} * e^{-\frac{x2^2}{2}} + \frac{x_1^2}{\sqrt{2!}} * e^{-\frac{x1^2}{2}} * \frac{x_2^2}{\sqrt{2!}} * e^{-\frac{x2^2}{2}} \dots..\right) =$$

$$= e^{-\frac{x1^2}{2}} * e^{-\frac{x_2^2}{2}} * \left(\frac{(x_1 * x_2)^0}{0!} + \frac{(x_1 * x_2)^1}{1!} + \frac{(x_1 * x_2)^2}{2!} \dots\right) =$$

$$= e^{-\frac{x1^2}{2}} * e^{-\frac{x_2^2}{2}} * e^{x_1 * x_2} = e^{-0.5(x_1 - x_2)^2}$$

The red equal sign is due to taylor series - $\sum(z^k/k!) = e\hat{}z$

And the one after that is due to $(a+b)^2 = a^2 + 2ab + b^2$

We can see that by we got kernel function! Will call it $k(x_1, x_2)$ by definition

It gets 2 vectors and return a scalar

Kernel

- Kernel $k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$,

$$< \phi(x1)^T * \phi(x2) >= k(x_1, x_2) = e^{-0.5(x_1 - x_2)^2}$$

We can see that our kernel function is a function of subtraction between x1 and x2 :

$$k(x_1, x_2) = e^{-0.5(x_1 - x_2)^2} = f(x_1 - x_2)$$

And the kernel function is symmetric:

$$k(x_1, x_2) = e^{-0.5(x_1 - x_2)^2} = k(x_2, x_1) = e^{-0.5(x_2 - x_1)^2}$$

1.4 Capability of generalization

a. The scientific term for the balance that Einstein meant to in machine learning aspect is the trade-off between bias, and variance that effect the model's complexity and generalization ability. Generalization refers to a model's ability to perform well on unseen data, as opposed to just the data it was trained on. Bias refers to the difference between the expected predictions of a model and the true values of the data. A model with high bias is simpler and tends to make consistent but not very accurate predictions. On the other hand, variance refers to the variability of a model's predictions for a given input. A model with high variance is sensitive to small fluctuations in the training data-overfit the training set, and thus has bad generalization ability – will not perform well on an unseen data.
   Occam's Razor principle states that, given multiple models that explain the data, the simplest model (the one with the fewest number of parameters) is preferred, as it is less likely to overfit and thus have better generalization-large variance and large bias. The AIC criterion can be seen as a way to balance the goodness of fit of the model (expressed by the likelihood term) with the simplicity of the model (expressed by the number of parameters), thus helping to find a simple (as possible) model with a good generalization.

b. The term 2p in the AIC formula represents the number of parameters in the model, which is a measure of the complexity of the model. A model with a high number of parameters is more likely to overfit the training data and have poor generalization. Therefore, this term serves as a bias term, as it favors simpler models (models with fewer parameters) which in general have lower bias.On the other hand, the term -2ln(^L) in the AIC formula represents the goodness of fit of the model based on the likelihood of the data given the model parameters. A model with a high likelihood is more likely to fit the training data well, but it may not generalize well to unseen data. Therefore, this term serves as a variance term, as it favors models that fit the data well but in general have higher variance.
   In summary, the balance between the two terms 2p and -2ln(^L) in AIC is aimed to find the optimal trade-off between model complexity (bias) and goodness of fit (variance) in order to achieve good generalization performance.

c. If the balance between the two terms in the AIC formula is violated which means that the balance between bias and variance was violated, two options are likely to happen:
   Overfitting: If the model is too complex and has a high number of parameters, it may fit the training data very well (high likelihood term) but it may not generalize well to unseen data. This is because the model has learned noise in the training data and is not able to generalize to new situations. This is

known as overfitting and it happens when the bias is too low.

Underfitting: If the model is too simple and has a low number of parameters, it may not fit the training data well (low likelihood term) but it may generalize well to unseen data. This is because the model has not learned enough information from the data to make accurate predictions. This is known as underfitting and it happens when the variance is too low.

d. We are aiming to minimize the AIC.

The AIC value is a trade-off between the goodness of fit of the model (measured by the likelihood term) and the complexity of the model (measured by the number of parameters).

Our goal is to find the balance between the lowest value of 2p and the highest value of 2ln(L), that will give us the best results from the model so it will be simpler , won't over/under fit and will has a good generalization ability.

1.5 Linear binary classffier with different cost function

1. The goal is to minimize the loss, hence we will use the derivative to find the minima:

$$\frac{\partial L}{\partial w} = -\frac{2}{n}\sum_{1}^{n} x_i(y_i - x_i^T w) = 0$$

$$\sum_{1}^{n} x_i(y_i - x_i^T w) = 0$$

$$\sum_{1}^{n} x_i y_i - x_i x_i^T w = 0 \rightarrow \sum_{1}^{n} x_i y_i - (\sum_{1}^{n} x_i x_i^T) * w = 0$$

$$w = \left(\sum_{1}^{n} x_i x_i^T\right)^{-1} * \sum_{1}^{n} x_i y_i$$

$L''(w) = -\frac{2}{n} * \sum_1^n -x_i x_i^T$>0 -> minima

The optimal weights will minimize the loss .

2. The loss of the batch will be accumulated, we will update the weights as follow:

$$w = w_0 - \eta * \left(-\frac{2}{4}\right)\sum_{i=1}^{4} x_i * (y_i - x_i^T * w)$$

3. With the given D= [((1,0),+1) , ((1,1),+1) ,((2,0),-1)] we can calculate numerically the optimal w:

$$w = \left(\sum_{1}^{n} x_i x_i^T\right)^{-1} * \sum_{1}^{n} x_i y_i$$

$$= \left(\begin{pmatrix}1\\0\end{pmatrix}(1\ 0) + \begin{pmatrix}1\\1\end{pmatrix}(1\ 1) + \begin{pmatrix}2\\0\end{pmatrix}(2\ 0)\right)^{-1} * \left(\begin{pmatrix}1\\0\end{pmatrix} * 1 + \begin{pmatrix}1\\1\end{pmatrix} * 1 + \begin{pmatrix}2\\0\end{pmatrix} * -1\right) =$$

$$= \left(\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} + \begin{pmatrix}4 & 0\\0 & 0\end{pmatrix}\right)^{-1} * \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}6 & 1\\1 & 1\end{pmatrix}^{-1} * \begin{pmatrix}0\\1\end{pmatrix} =$$

$$= \frac{1}{5} * \begin{pmatrix}1 & -1\\-1 & 6\end{pmatrix} * \begin{pmatrix}0\\1\end{pmatrix} = \frac{1}{5}\begin{pmatrix}-1\\6\end{pmatrix}$$

The optimal weights are $\mathrm{w}_1 = -\frac{1}{5}$ and $\mathrm{w}_2 = \frac{6}{5}$

Now we will find the decision boundary :

We are in 2D so the decision boundary will be a line , y=m*x+b , in our case b=0 ,no bias,

decision boundary equation : $f(x_1, w) = w^T * x = 0 \rightarrow \frac{1}{5} * (-1 \quad 6) * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \rightarrow$

$$\boldsymbol{x_2 = \frac{1}{6} x_1}$$

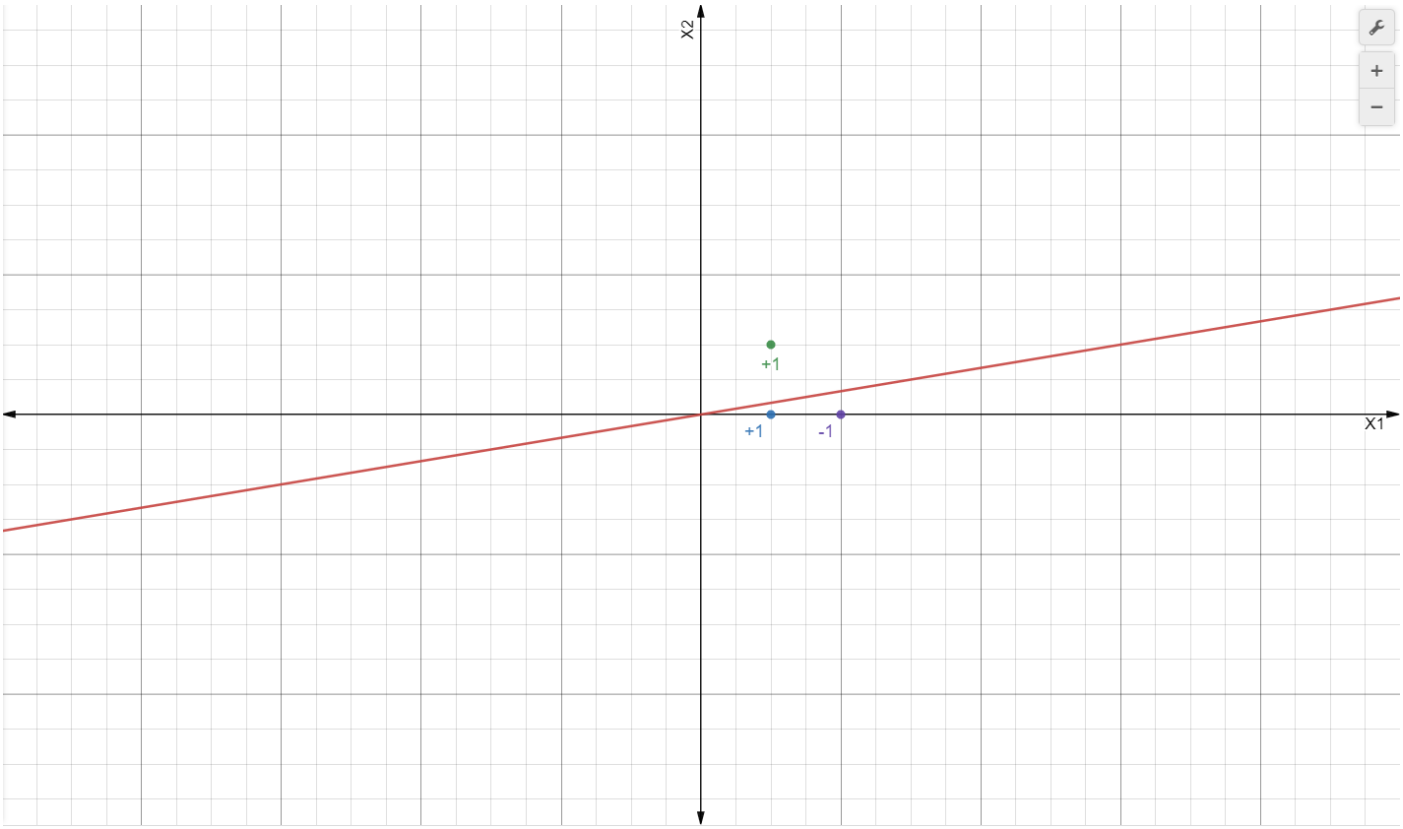

Figure 1 decision boundary with the given dataset (showing the correct labels)

As we can see at figure 1, point (1,0) was miss-labeled as -1 , instead of +1, as it below the decision boundary line .

I can conclude that different tasks required different loss functions and that the choice of the cost function can have a significant impact on the performance of the model .

Here our model failed at a fairly simple task – because of an inadequate choice of loss function for the task – classification , we had better used cross entropy loss to get better results .