



## Answers for theoretical questions

Edited by **Nathan Berdugo, 209570571**

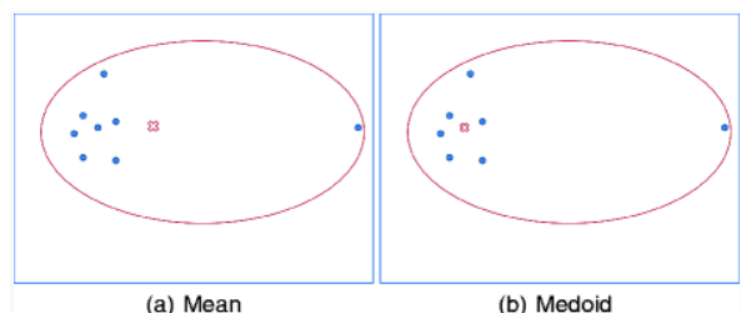
### 1. Clustering

a. Yes, K-medoid is indeed more robust to noise than the K-means algorithm. Whereas K-medoid and K-means algorithm share common characteristics (as they both try to minimize the distance between an example to a point in space), they have some intrinsic differences. Here I will explain the difference that makes K-means more sensitive to noise (and outliers) than K-medoid.

As explained in the question introduction, **K-medoid** algorithm uses L1 minimization in order to achieve dissimilarity minimization between the examples within the dataset. An important difference between K-means and K-medoid is that *in K-medoid, the center of a cluster is chosen from the actual dataset*, while K-means not necessarily picks a 'centroid' from within the dataset.

By determining the center of a cluster with a point in space which is not included in the dataset, **K-means** allows noise or extreme data to influence center-point selection, since the algorithm - as its name implies - is based on the mean. As we learned in the course lectures, as well as in other courses (especially in the Statistics 094423), **the mean is very sensitive to extreme values**. By using K-means, we give great weight to those noises and extreme values (outliers) that can significantly affect the determination of the cluster center.

This effect will be reflected to a lesser extent in the K-medoid algorithm since as stated, the center is determined from the dataset and therefore, noise nor outliers will not be reflected in this method.



**Figure 1** – A demonstration of a 2D-dataset with an extreme value (on the right edge of the ellipse). One can notice that in K-means (a), the centroid is highly influenced by the outliers as an intrinsic characteristic of the mean. While K-medoid (b) chooses an example **from within the dataset** as the center of the cluster. <sup>1</sup>

b. To-be-proved: For 1D case of K-means, the centroid ( $\mu$ ) which minimizes the term

$\sum_{i=1}^m (x_i - \mu)^2$  is the mean of  $m$  examples.

**Proof:**

Let :  $\bar{x} \in \mathbb{R}$  is the mean of a population  $m$

$$\begin{aligned} \sum_{i=1}^m (x_i - \mu)^2 &= \sum_{i=1}^m ((x_i - \bar{x}) + (\bar{x} - \mu))^2 \\ &= \sum_{i=1}^m (x_i - \bar{x})^2 + 2 \sum_{i=1}^m (x_i - \bar{x})(\bar{x} - \mu) + \sum_{i=1}^m (\bar{x} - \mu)^2 \\ &= \sum_{i=1}^m (x_i - \bar{x})^2 + 2(\bar{x} - \mu) \sum_{i=1}^m (x_i - \bar{x}) + m(\bar{x} - \mu)^2 \end{aligned}$$

$$\sum_{i=1}^m (x_i - \bar{x}) = \sum_{i=1}^m x_i - m\bar{x} = \sum_{i=1}^m x_i - m \cdot \frac{1}{m} \sum_{i=1}^m x_i \equiv 0$$

$$\Rightarrow \sum_{i=1}^m (x_i - \mu)^2 = \sum_{i=1}^m (x_i - \bar{x})^2 + m(\bar{x} - \mu)^2$$

One can notice that in order to minimize the term above,  $\mu$  should be valued as  $\bar{x}$  (i.e.  $\mu = \bar{x}$ ).



**Bonus.** To-be-proved: For K-medoid, the medoid which minimizes the term  $\sum_{i=1}^m |x_i - \mu|$  is the median of  $m$  examples, given that  $\mu$  belongs to the dataset.

**Proof (in next page):**

I was assisted by the proof that appears in <sup>2</sup>

<sup>2</sup>/ The median minimizes the sum of absolute deviations (the  $\ell_1$  norm), StackExchange:  
<https://math.stackexchange.com/questions/113270/the-median-minimizes-the-sum-of-absolute-deviations-the-ell-1-norm>

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

and suppose:  $x_1 < x_2 < \dots < x_n$

Let's try and find out how  $f(\mu)$  behaves when changing the value of  $\mu$ :

if  $\mu < x_1$ :

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

As the value of  $\mu$  increases, the value of  $f(\mu)$  decreases,

implying for a strictly monotonically decreasing function

until  $\mu$  reaches  $x_1$

$$\Rightarrow f(\mu) > f(x_1) \quad \forall \mu < x_1$$

Now let:  $x_k \leq \mu \leq \mu + d \leq x_{k+1}$ , for  $d \in \mathbb{R}$  and  $1 \leq k \leq m$

$$\begin{aligned} f(\mu + d) &= \sum_{i=1}^m |x_i - (\mu + d)| = \sum_{i=1}^k (\mu + d - x_i) + \sum_{i=k+1}^m (x_i - (\mu + d)) = \\ &= dk + \sum_{i=1}^k (\mu - x_i) - d(m - k) + \sum_{i=1}^k (x_i - \mu) \\ &= d(2k - m) + \sum_{i=1}^k (\mu - x_i) + \sum_{i=1}^k (x_i - \mu) \\ &= d(2k - m) + f(\mu) \end{aligned}$$

$$\Rightarrow d(2k - m) = f(\mu + d) - f(\mu) \quad / \text{ divide by } d \rightarrow 0$$

$$\Rightarrow 2k - m = \frac{f(\mu + d) - f(\mu)}{d}$$

$\Rightarrow$  we result with the *derivative definition* of  $f(\mu)$

$$\Rightarrow \begin{cases} f'(\mu) < 0 & \text{if } 2k < m \Rightarrow k < \frac{m}{2} \\ f'(\mu) > 0 & \text{if } 2k > m \Rightarrow k > \frac{m}{2} \\ f'(\mu) = 0 & \text{if } 2k = m \Rightarrow k = \frac{m}{2} \end{cases}$$

By the definition of the median - which is any value such that


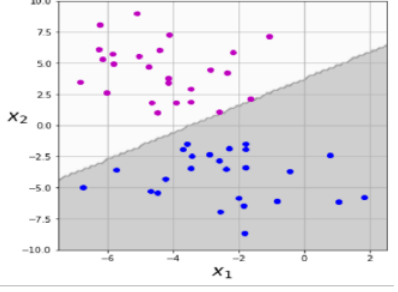

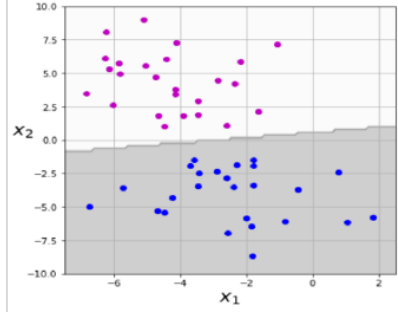
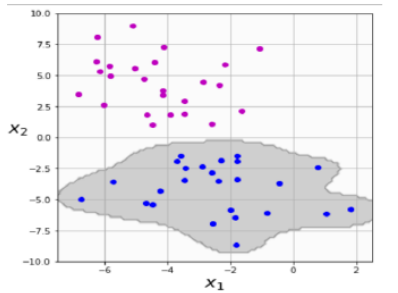
half of the population ( $\frac{m}{2}$ ) is lower than the proposed median,

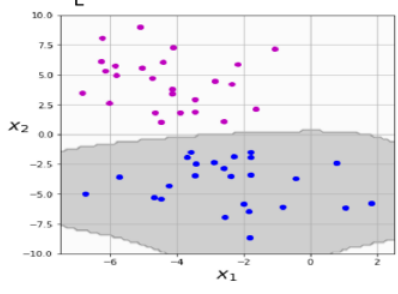
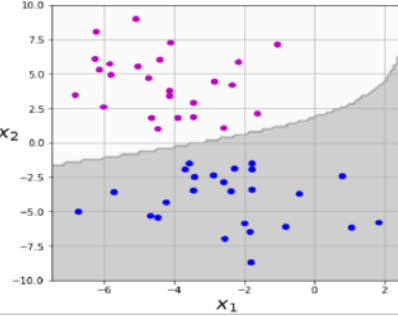
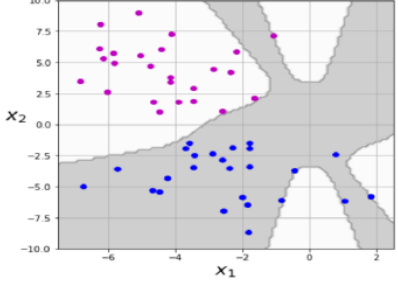
and the other half is greater than it - one can conclude

by the behaviour of the function above ( $f(\mu)$ ) that the value that minimize it

is  $k = \frac{m}{2}$  which is by definition the median of the ordered population  $m$ .

## 2. SVM

<p><b>Linear kernel with <math>C=1</math></b></p> 	<p>As we can see, by drawing a simple straight line we got a separation of our dataset, hence the Linear kernel. As we learned in the lectures, the hyperparameter <math>C</math> influences the trade-off between {increasing the distance between the hyperplane and the support vectors} and {decreasing the number of samples which are misclassified by this hyperplane}.</p> <p>By comparing the two Linear Classifiers given in this exercise, one can conclude that the linear kernel with the hyperparameter values as 1 is the highest, thus <i>the Euclidian distance (i.e., the margin) between the hyperplane drawn by this simple line and support vectors will be <b>small</b>, and the number of misclassified samples will be <b>low</b></i> – which might result with <b>overfitting</b> to our data and poor testing/validating performance. In other words, <i>the Bias will be <b>high</b>, while the Variance is <b>low</b>.</i></p>	<p><b>A</b></p> 
<p><b>Linear kernel with <math>C=0.01</math></b></p> 	<p>At first, we can recognize a simple-line-separation, which implies for SVM with the setting of a linear kernel.</p> <p>Additionally, we can see the result of a linear kernel with an hyperparameter (<math>C</math>) which is smaller than the previous <i>by two orders of magnitude</i>. For this reason, the margin is bigger (i.e., the hyperplane is less tightened to the support vectors), and the possibility of resulting with misclassified samples is bigger (even though it seems that in our case, no misclassified samples is found). Determining a small-valued <math>C</math> might result with <b>underfitting</b> which in turn will also lead to poor performance in the testing/validating phase. In other words, <i>the Bias will be <b>low</b>, while the Variance is <b>high</b>.</i></p>	<p><b>D</b></p> 
<p><b>RBF kernel with <math>\gamma = 1</math></b></p>	<p>An RBF (Radial Basis Function) kernel is a common non-linear SVM kernel, which uses Gaussian function in order to separate our data by the squared Euclidian distance between our support vectors.</p> <p>The margin here is defined by the value of <math>\gamma</math>, which in turn is proportional to the inverse of the variance squared. Meaning that <b>the higher the <math>\gamma</math>, the smaller the margin between the support-vectors</b>, resulting in low probability of resulting with misclassified samples but high chances of getting <b>overfitting</b> to the training model. In this case: <i>The Bias will be <b>high</b>, while the Variance is <b>low</b>.</i></p>	<p><b>B</b></p> 

<p><b>RBF kernel with <math>\gamma = 0.2</math></b></p>	<p>At first one can see the radial separation of our data, which implies for RBF kernel.</p> <p>By comparing this separation to the earlier, we can observe a greater margin with high probability of 'capturing' and misclassifying samples, which may lead to <b>underfitting</b> and again to poor results in the testing phase.</p> <p>A small value of <math>\gamma</math> can lead to these results, hence the choice as it is written on the left column.</p>	<p>E</p> 
<p><b>2<sup>nd</sup> order polynomial kernel</b></p>	<p>At first we can easily determine the non-linear separation that has been used as the SVM setting in this example. The degree of the polynomial kernel determines the degree of the function which will try to optimize the separation of our dataset.</p> <p>The lower the polynomial degree, the higher the probability of resulting with misclassified samples. However, and accordingly with previous explanation about this principle, <i>increasing the chance of getting misclassified samples goes side-by-side with <b>underfitting</b>, low bias and high variance.</i></p> <p>In this case, the hyperplane has indeed the characteristic of a small-order polynomial degree, which led to this match I chose.</p>	<p>C</p> 
<p><b>10<sup>th</sup> order polynomial kernel</b></p>	<p>First, we can obviously look at a non-linear separation of our data.</p> <p>The kernel here has been determined with a much higher polynomial degree, which led to the attempt of separating our data by using this 10<sup>th</sup> degree polynome.</p> <p>In contrary to the previous example, this higher degree polynome might result with much lower misclassified samples, and therefore with much higher chances of <b>overfitting</b> to the training data.</p> <p>In this case, <i>the bias is very <b>high</b>, while the variance is <b>low</b>.</i></p>	<p>F</p> 

### 3. Capability of generalization



a. The scientific term of balance in ML aspect, which A. Einstein meant to in his words is **Generalization**.

In ML aspect, we attempt to decrease the dimensions of our data (hence, make our data '*as simple as possible*'), **while maintaining a proper balance between the decrease in complexity and the loss of valuable data**.

By achieving this balance, we can create a model with high generalization capabilities, while avoiding the need of high computational resources. A model which has been manipulated in the training phase to contain *only* valuable, highly-important features, will be able – in the validation/testing phase – to generalize the learning to new datasets, and reach high performance.

In contrary, **a violation of this balance** - by overlooking valuable features and over-simplifying the model – **might lead to an underfitted model** which may result with poor performance in the testing phase, and will represent what A. Einstein called in his words: '*simpler*'.

b. As explained in to intro of the question, the AIC is defined as follows:

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

Where  $p$  is the total number of learned parameters, and  $\hat{L}$  is the estimated likelihood given these parameters. Corresponding to the explanation in the previous section, **the higher  $p$  is**, the higher the dimensions of our model is, and the number of features we include in it increases and the chances of resulting with **overfitting** increases as well. This leads directly to higher complexity, which in turn demands stronger computational resources while some features might not be valuable enough and will not contribute any high-importance data which helps our prediction model.

Of course that the opposite scenario is indeed plausible, when **smaller value of  $p$  might result with underfitting**, and again to poor performance in the testing phase.

Therefore, in AIC one should always try and find the proper value of  $p$  in order to avoid over/underfitting and reach a proper predicting model.

The component  $\hat{L}$  is a measure of how much our model success in reconstructing our data. This serves as a statistical analysis which quantify the resemblance of the prediction (i.e., the performance of our model) and the labeling.

**By maximizing  $\hat{L}$  (i.e., minimizing  $-2\ln(\hat{L})$ ) we reach a model with higher reliability for better performance in the testing phase.**

In conclusion, one should check the AIC when choosing the model it wants to use for learning. The lower AIC, the better is the model and the higher the chances of reaching **generalization** (with exception of *too small value of  $p$*  that led to the low value of AIC, which may imply, as explained, for underfitting).

c. As explained previously, *a violation of this balance* might lead to over/underfitting.

This violation could happen both **by reducing too much the value of  $p$**  and overlooking valuable features resulting with an over-simplifying model – *Underfitting*;

And by leaving the value of  $p$  as high as it can be, with including non-contributing features which **might lead to an overfitted, highly-biased model**.

These two options are likely to result with poor performance in the testing phase.

d. In overall, we are aiming for a *low value of AIC*. This may imply for a simple, reduced-dimension model (**low value of  $p$** ) which has been manipulated to overlook non-contributing features and include only the high-valuable features that improve the prediction performance.

In addition, this low value of AIC implies for a high value of  $\hat{L}$ , which demonstrate the high ability of our model to predict the labeling (high likelihood).

Note that not always the best AIC is the lowest one, as the value of  $p$  should be determined by the balance of minimizing the complexity and avoiding over-simplicity of the model (as explained in more depth earlier). Thus, *one should look for the lowest AIC that includes the proper value of  $p$* .