HW3-Mechine Learning- Theoretical questions

1. **Clustering**

   a. K-medoid is more robust to noise than K-mean. That because it not using the mean point as the center of a cluster (like k-means does). *K*-medoids uses an actual point in the cluster to represent it. Medoid (set of cluster representatives) is the most centrally located object of the cluster, with minimum sum of distances to other points.

   Therefore, we can compare the medoid to the median. We know that median is more forgiven to outliers than arithmetic mean that used in K-means.

   b. Prove that μ which minimizes the term $\sum_{i=1}^{m}(x_i-\mu)^2$ is the mean of m examples.

   mean of m examples is: $\bar{x}=\frac{1}{m}\sum_{i=1}^{m}x_i$

   By using Maximum Likelihood Estimation statistic method:

$$L(\mu)=\prod_{i=1}^{m}\frac{1}{x_i\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}=\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m\prod_{i=1}^{m}\frac{1}{x_i}e^{-\frac{\left((\sum_{i=1}^{m}x_i)^2-2\sum_{i=1}^{m}x_i\mu+m\mu^2\right)}{2\sigma^2}}$$

$$l(\mu)=\ln\big(L(\mu)\big)=-\frac{((\sum_{i=1}^{m}x_i)^2-2\sum_{i=1}^{m}x_i\,\mu+m\mu^2)}{2\sigma^2}$$

$$\frac{dl(\mu)}{d\mu}=\left.\frac{2\sum_{i=1}^{m}x_i-2m\mu}{2\sigma^2}\right|_{\hat{\mu}}=0$$

$$\hat{\mu}=\frac{\sum_{i=1}^{m}x_i}{m}=\bar{x}\ \blacksquare$$

   <u>Bonus</u>: prove that the medoid which minimizes the term $\sum_{i=1}^{m}|(x_i-\mu)|$ is the median of m examples.

   First, we define the median for an odd and even numbers of observations.

   Median for odd m: $Med=x_{\frac{m+1}{2}}$

   Median for even m: $Med=\frac{x_{\frac{m}{2}}+x_{\frac{m}{2}+1}}{2}$

   We assume that $x_1<x_2<x_3\ldots\ldots<x_m$

$$\sum_{i=1}^{m}|(x_i-\mu)|=\sum_{i=2}^{m}|(x_i-\mu)|+(x_m-x_1)\quad when\ \mu\in[x_i,x_m]$$

   For an odd m:

$$\sum_{i=1}^{m}|(x_i-\mu)|=\left|x_{\frac{m+1}{2}}-\mu\right|+(x_m-x_1)+(x_{m-1}-x_2)+\cdots$$
$$=\left|x_{\frac{m+1}{2}}-\mu\right|+const$$

   If we derive the function $\frac{d\ \sum_{i=1}^{m}|(x_i-\mu)|}{d\mu}$ and equal to zero, we will find that the minimum of this function is the vertex: $(x_{\frac{m+1}{2}},const)$.

   So, we can say that for an odd m the median: $\mu=x_{\frac{m+1}{2}}$ minimize the function.

For an even m:

$$\sum_{i=1}^{m}|(x_i-\mu)| = \left|x_{\frac{m}{2}}-\mu\right| + \left|x_{\frac{m+2}{2}}-\mu\right| + const$$

If we derive the function and equal to zero, we will get that:

$\mu = median = \frac{x_{\frac{m}{2}}+x_{\frac{m}{2}+1}}{2}$ minimize the function of $\sum_{i=1}^{m}|(x_i-\mu)|$ for an even m.

∎

2. SVM

For linear kernel we expected to see a linear separation between classes (A D are the images that fit for this). The value of the hyper parameter C (penalization for miss classification) is the one that define which minimal margin will be chosen to distinguish between classes. The bigger C is -the margin will be smaller.

For large C- hard margin, we cannot accept a leakage from our margin, and there is less miss classification. We can see in figure D there we see perfect classification and no miss classification.

If we compare this to A, that there is a pink labels near the separation line and we know that the distance from the line to the support vectors in each area is equal. So, we can assume that in figure A we can accept miss classification and that is fit to a small C.

Small C-soft margin, large margin, we can accept an error in classification.

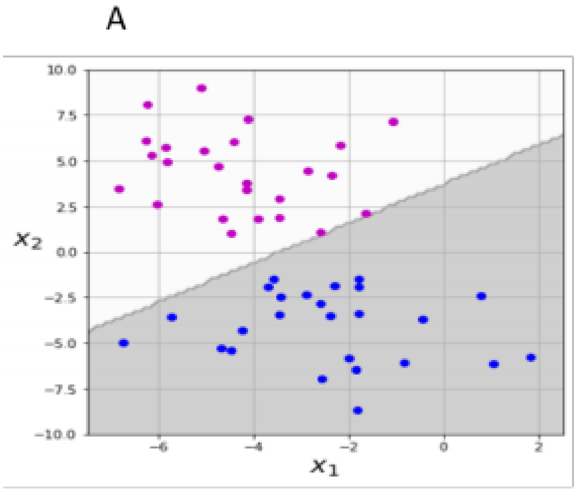

Figure 1: Linear kernel with C=0.01

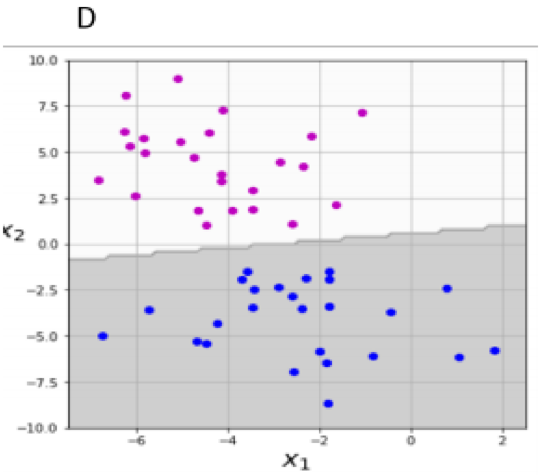

Figure 2: Linear kernel with C=1

In the polynomial kernel we calculate the dot product by increasing the power of the kernel: $k(x_i, x_j) = (x_i \cdot x_j + r)^d$ d is the degree of the polynomial. We want to measure the similarity between x_i, x_j. For low order, the classification will be look as low degree curve (almost linear, but not). We can match 2nd order polynomial kernel to image C.

For higher order like in 10th order polynomial kernel the learning is very good so the classification will be accurate, and the shape will be tight around the examples - image F

C

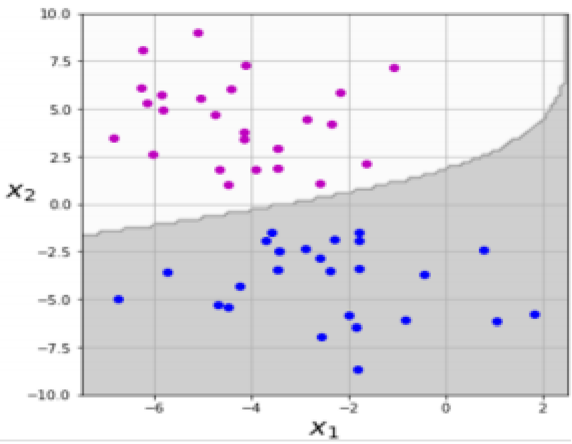

Figure 3: 2^{nd} *order polynomial kernel*

F

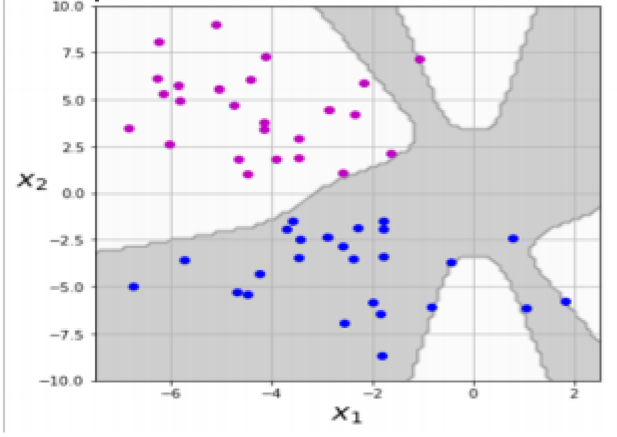

Figure 4: 10^{th} *order polynomial kernel*

For RBF kernel the hyperparameter γ is the one that supervise the classification. This kernel is with Gaussian distribution $\gamma = \frac{1}{2\sigma^2}$.

For low γ the variance is high so the classification will surround the examples but not very tight around them-like in image E.

As we increase the value of gamma the shape of the class will be tighter-like in image B, and sometime in can cause overfitting.

E

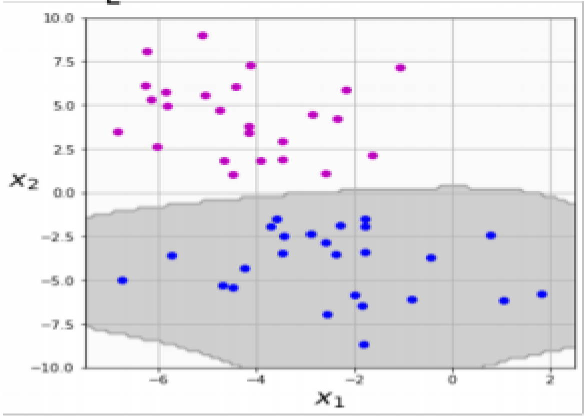

Figure 5 RBF kernel with $\gamma = 0.2$

B

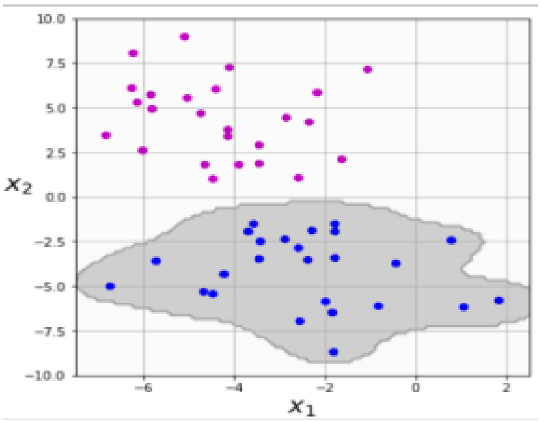

Figure 6: RBF kernel with $\gamma = 1$

1.	Linear kernel with C=0.01	A
2.	Linear kernel with C =1	D
3.	2^{nd} order polynomial kernel	C
4.	10^{th} order polynomial kernel	F
5.	RBF kernel with $\gamma = 0.2$	E
6.	RBF kernel with $\gamma = 1$	B

3. Capability of generalization

a. The scientific term of balance that Einstein meant to in ML aspect is generalization. If we general our model, a simpler hypothesis representation and there is less prone to overfitting. We can say that we seek the tradeoff between good fitting and complexity.

b. AIC rewards goodness of fit- $2\ln(\hat{L})$ how well the classification fit for our example. And also the AIC includes a penalty -2p , is an increasing function that present the number of estimated parameters - The higher it is the high level of complexity.

c. If this balance is violated to the fitting side, we expect to see overfitting (too good fitting), and it can cause a problem to run the model on other data (not the same data that train the model). So, the learning is like memorize-no deep understanding and a good learning model.
If the opposite happened, the balance has been violated to the other side: high complexity, it can cause underfitting and the model will fail and not succeed to learn at all.

d. The aiming for the AIC is to do statistical criterion that estimates the quality of each model (compering between models). By calculate the balance of the lack of fit and the model complexity. We want to reduce the AIC because Lower AIC values represent a better fit model, it indicates a better balance between fit and complexity. And for high AIC (more than 2) the model is to complex or do overfitting and this not the desired outcome.