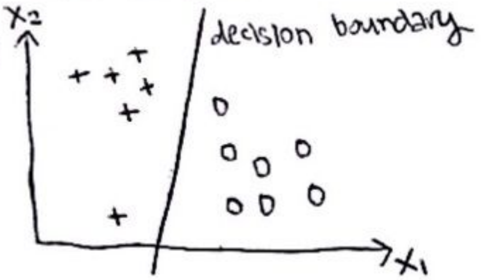
Yohandi - 120040025

DDA3020: Homework II

1(1)

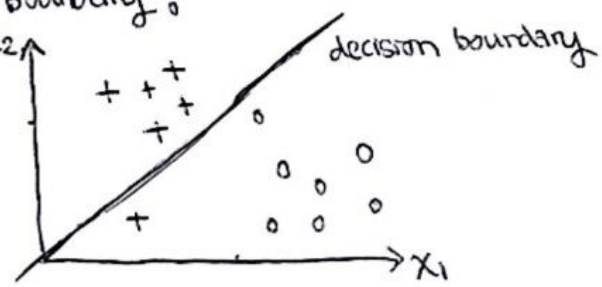


The data are linearly separable. This implies that logistic regression will find a line that perfectly fits the data.

$\Rightarrow$ No classification errors for training datasets

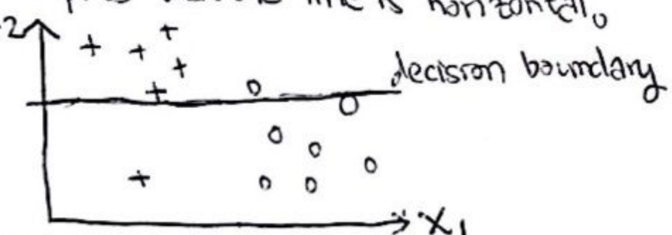
$\Rightarrow$ The line is not unique (add margin)

(2) Since w_0 is regularized to 0, we must have (0, 0) in the decision boundary.



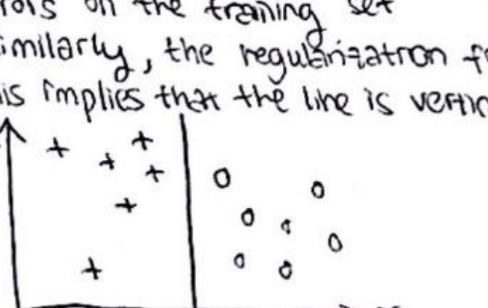
$\Rightarrow$ The method makes one classification error on the training set

(3) The regularization forces $w_1 = 0$. This implies that the line is horizontal.



$\Rightarrow$ The method makes two classification errors on the training set

(4) Similarly, the regularization forces $w_2 = 0$. This implies that the line is vertical



$\Rightarrow$ The method makes no classification error on the training set

2(1) $\phi(x_1) = [1,0,0]^T \quad \phi(x_2) = [1,2,2]^T$

it is best decision to have the plane with $[0,1,1]^T$ and pass the $[1,1,1]^T$ point. We have $w \parallel [0,1,1]^T$

(2) Since $[1,1,1]^T$ connects two points and intersects the decision boundary, we have the margin distance as the distance between $[1,1,1]^T$ and $[1,0,0]^T$ that is $\sqrt{2}$

(3) $\Rightarrow w \parallel [0,1,1]^T \Rightarrow w = [0,k,k]^T$

$\Rightarrow \|w\| = \frac{1}{\sqrt{2}} \Rightarrow \sqrt{k^2+k^2} = \frac{1}{\sqrt{2}}$

This implies that $k = \frac{1}{2} \Rightarrow w = [0, \frac{1}{2}, \frac{1}{2}]^T$

(4) $-1(0+\frac{1}{2}\cdot 0+\frac{1}{2}\cdot 0+w_0) \geq 1$

$1(0+\frac{1}{2}\cdot 2+\frac{1}{2}\cdot 2+w_0) \geq 1$

$\Rightarrow w_0 \leq -1$ and $w_0 \geq -1$

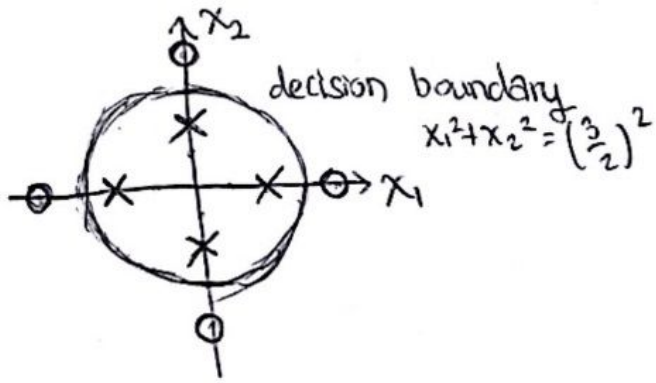
$\Rightarrow w_0 = -1$

(5) $f(x) = w_0 + w^T\phi(x)$

$= -1 + 0 + \frac{1}{2}\cdot\sqrt{2}x + \frac{1}{2}\cdot x^2$

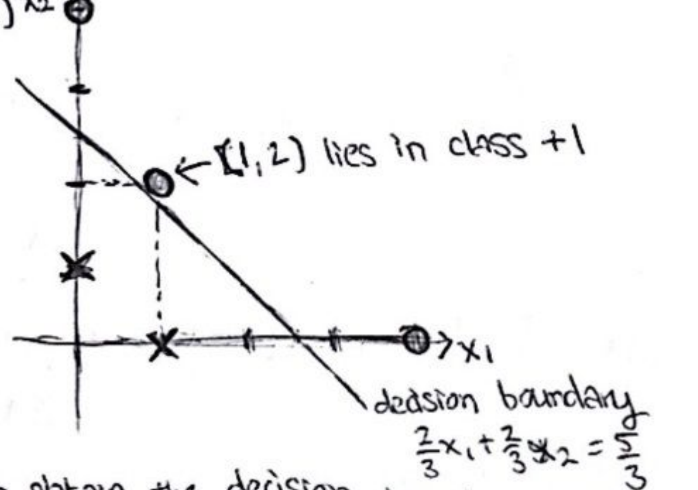
$= -1 + \frac{\sqrt{2}}{2}x + \frac{1}{2}x^2$

3(1) Denote ✖ as class -1 and **O** as class +1



As a SVM classifier always try to make a decision boundary to be as wide as possible, an equation $x_1^2+x_2^2=(\frac{3}{2})^2$ is used as the decision boundary. This way, each point in both classes will have exactly $\frac{1}{2}$ unit distance from the boundary

i2)



To obtain the decision boundary, we first must notice that initially we have supporting vectors: $S_1=\begin{pmatrix}1\\0\end{pmatrix}, S_2=\begin{pmatrix}0\\1\end{pmatrix}, S_3=\begin{pmatrix}4\\0\end{pmatrix}$, $S_4=\begin{pmatrix}0\\4\end{pmatrix}$. Adding bias: $\tilde{S}_1=\begin{pmatrix}1\\0\\1\end{pmatrix}, \tilde{S}_2=\begin{pmatrix}0\\1\\1\end{pmatrix}$, $\tilde{S}_3=\begin{pmatrix}4\\0\\1\end{pmatrix}, \tilde{S}_4=\begin{pmatrix}0\\4\\1\end{pmatrix}$

Equations are:

$$\alpha_1\tilde{S}_1\cdot\tilde{S}_1+\alpha_2\tilde{S}_2\cdot\tilde{S}_1+\alpha_3\tilde{S}_3\cdot\tilde{S}_1+\alpha_4\tilde{S}_4\cdot\tilde{S}_1=-1$$
$$\alpha_1\tilde{S}_1\cdot\tilde{S}_2+\alpha_2\tilde{S}_2\cdot\tilde{S}_2+\alpha_3\tilde{S}_3\cdot\tilde{S}_2+\alpha_4\tilde{S}_4\cdot\tilde{S}_2=-1$$
$$\alpha_1\tilde{S}_1\cdot\tilde{S}_3+\alpha_2\tilde{S}_2\cdot\tilde{S}_3+\alpha_3\tilde{S}_3\cdot\tilde{S}_3+\alpha_4\tilde{S}_4\cdot\tilde{S}_3=+1$$
$$\alpha_1\tilde{S}_1\cdot\tilde{S}_4+\alpha_2\tilde{S}_2\cdot\tilde{S}_4+\alpha_3\tilde{S}_3\cdot\tilde{S}_4+\alpha_4\tilde{S}_4\cdot\tilde{S}_4=+1$$

$$\Rightarrow \left.\begin{array}{l} 2\alpha_1+\alpha_2+5\alpha_3+\alpha_4=-1\\ \alpha_1+2\alpha_2+\alpha_3+5\alpha_4=-1\\ 5\alpha_1+\alpha_2+17\alpha_3+\alpha_4=1\\ \alpha_1+5\alpha_2+\alpha_3+17\alpha_4=1\end{array}\right\} \begin{array}{l}\alpha_4=1\\ \alpha_1=\frac{2}{3}\\ \alpha_2=-\frac{10}{3}\\ \alpha_3=0\end{array}$$

$$\tilde{w}=\sum_{i=1}^{4}\alpha_i\tilde{S}_i=\begin{pmatrix}\frac{2}{3}\\ \frac{2}{3}\\ -\frac{5}{3}\end{pmatrix}$$

$$\Rightarrow w=\begin{pmatrix}\frac{2}{3}\\ \frac{2}{3}\end{pmatrix} \text{ and } b=-\frac{5}{3}$$

For label [1;2], we have

$$w\begin{pmatrix}1\\2\end{pmatrix}+b=\begin{pmatrix}\frac{2}{3}\\ \frac{2}{3}\end{pmatrix}\begin{pmatrix}1\\2\end{pmatrix}-\frac{5}{3}=\frac{1}{3}>0$$

(class +1)

4. We first have a lagrange function:

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2+\sum_{i=1}^{m}\alpha_i(1-y_i(w^Tx_i+b))$$

based on lecture 7, the stationary condition of the function is obtained by partially derivate $\mathcal{L}$ to w and b and put it equals to 0. we have:

$$\frac{\partial\mathcal{L}}{\partial w}=0\Rightarrow\frac{\partial(\frac{1}{2}\|w\|^2+\sum_{i=1}^{m}\alpha_i+\sum_{i=1}^{m}-(\alpha_iy_i(w^Tx_i+b)))}{\partial w}=0$$

$$\Rightarrow\frac{1}{2}\cdot2w-\sum_{i=1}^{m}\alpha_iy_iX_i=0$$

$$\Rightarrow w=\sum_{i=1}^{m}\alpha_iy_ix_i$$

$$\frac{\partial\mathcal{L}}{\partial b}=0\Rightarrow\frac{\partial(\frac{1}{2}\|w\|^2+\sum_{i=1}^{m}\alpha_i(1-y_iw^Tx_i)-\sum_{i=1}^{m}\alpha_iy_ib)}{\partial b}=0$$

$$\Rightarrow-\sum_{i=1}^{m}\alpha_iy_i=0 \text{ or } \sum_{i=1}^{m}\alpha_iy_i=0$$

now, by having a stationary condition for the function, we have dual representation of the maximum margin problem. the problem is as follows:

$$\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jy_iy_jx_i^Tx_j$$

such that $\sum_{i=1}^{m}\alpha_iy_i=0$, $\alpha_i\geq0, \forall i$

to classify, we are interested in the value of $\text{sgn}(w^Tx+b)$ as a prediction.

$$w^Tx+b=\sum_{i=1}^{m}\alpha_iy_ix_ix+b$$

margin distance is defined as the nearest point; denote this as γ. we have:

$$\gamma=\frac{y_i(w^Tx+b)}{\|w\|}$$

$$\gamma^2=\frac{(y_i(w^Tx+b))^2}{\|w\|^2}=\frac{1}{w^Tw}$$

$$\Rightarrow\frac{1}{\gamma^2}=w^Tw=\sum_{i=1}^{m}\alpha_iy_i\left(\sum_{j=1}^{m}\alpha_jy_jx_jx_j\right)$$

$$=\sum_{i=1}^{m}\alpha_iy_i(w^Tx-b)$$

$$=\sum_{i=1}^{m}\alpha_i\underbrace{y_i(w^Tx+b)}_{1}-2\underbrace{\sum_{i=1}^{m}\alpha_iy_ib}_{0}$$

(L7S12)

$$=\sum_{i=1}^{m}\alpha_i \text{ (shown)}$$

Programming Question

Yohandi [SID: 120040025]

Support Vector Machine

In this problem we are asked to write a program that construct support vector machine models with different kernel functions and slack variable. The kernel functions vary from standard linear separator function, polynomial separator function, radical basis function, and sigmoid function.

To train an SVM model, we solve a typical optimization problem of the following form:

$min\frac{1}{2}||w||^2 s.t. y_i(w^T x_i + b) \geq 1$, for all i.

We use one of the methods, namely Lagrange function and KKT conditions, to alter the optimization problem. Consequently, the problem is modified into the dual problem. Suppose we have a lagrange function, which is denoted as $L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \alpha_i(1 - y_i(w^T x_i + b))$. Now, to achieve a turning point (maximum or minimum) results, we use a method from calculus named partial derivative. Derivate it with respect to w and b and set it to 0, we obtain: $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$. Detail derivation can be found in the written part or lecture slides.

Note that in the previous part, we need to achieve both conditions so that we can get the required results. After exclude both variables w and b, we get the previously described dual representation for the maximum margin problem. That being said, we want to solve the optimization problem by finding:

$max \sum_{i=1}^{m} \alpha_i - \sum_{i=1,j=1}^{m,m} \alpha_i \alpha_j y_i y_j x_i^T x_j$ such that $\sum_{i=1}^{m} \alpha_i y_i = 0$

Solving this allows our machine to model and train the SVM. According to lecture slides, with an assumption S is the set of support vectors, we will get $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ and $b = \frac{1}{S} \sum_{j \in S}(y_i - \sum_{i=1}^{m} \alpha_i y_i x_i^T x_j)$. Again, the detail derivation can be found in the written part or lecture slides.

In implementing, we are provided with training and testing datatset , namely train.txt and test.txt. Those datasets have 120 training data and 30 testing data, respectively. It covers 3 classes, corresponding to setosa, versicolor, virginica. They are derived from the Iris dataset, contains 3 classes of 50 instances each, where each class refers to a type of iris plant. We are asked to classify each iris plant as one of the three possible types.

In this task, we use the SVM function from python sklearn package. For multi-class SVM we use one vs rest strategy. Since the basic form of SVM is given, it is no longer necessary to derive it. Numpy is used for the vector manipulation.

Hyperparameter Settings

According to the official scikit-learn website, the C-Support Vector Classification contains a plenty of parameters. Those are `C` that denotes the regularization parameter, `kernel` that specifies the kernel type to be used in the algorithmm, `degree` that specifies the degree of the polynomial kernel function, `gamma` that specifies the kernel coefficient, etc.

As required in the question, there are some variants that we are to implement in this task. The settings for each question are as followings:

Question 1 We are asked to calculate using the standard SVM model (linear separator) for Question 1. With that case, we set the `kernel` to "linear" to obtain the required linear separator. Since we need to calculate the SVM without slack, we must set `C` to a large value. The question requests that the software be simulated using `C = 10 ** 5`. Therefore, `C`'s hyperparameter is now set to 10^5.

Question 2 In question 2, the problem requests that we compute the SVM with slack. First, we set the `kernel` to "linear" to obtain the requested linear separator. Since we are required to employ slack, a tiny amount of `C` is set. The question requests us to use `C = 0.1 * t` for every $1 \leq t \leq 10$ and fit it to the algorithm on the training dataset before validating it on the testing dataset. As a consequence, we employ ten distinct values of `C`.

Question 3 In this question, we are to implement SVM with kernel functions and slack variables. We experiment with different kernel functions in this task. However, for all parts, a constant gamma is fixed to 1.

Question 3.1 In this part, we set `kernel` to "poly" and set the `degree` to 2. We also set `C` to 1.

Formula: $(y < x, x' > +r)^d$

Question 3.2 In this part, we set `kernel` to "poly" and set the `degree` to 3. We also set `C` to 1.

Question 3.3 In this part, we set `kernel` to "rbf". As `gamma` is equal to $\frac{\sigma^2}{2}$, we set it to 0.5.

Formula: $e^{-y||x-x'||^2}$

Question 3.4 In this part, we set `kernel` to "sigmoid". As `gamma` is equal to $\frac{1}{d(X)}$, in which $d(X)$ is the dimension of X, we set it to 0.25.

Formula: $tanh(y < x, x' > +r)$

Data Loading

The program uses `read_csv()` function, which is provided in the `pandas` library, to read the information in the file. As the given data is separated by a tab, we use `sep = '\t'` as the parameter in the `read_csv` function. Notice that the loading of the data is using relative path instead of absolute path. Hence, when executing the code of the model, one should place the CSV file under the same directory of the model python file.

For each files, we extract the data to some variables. Denote it as `X_train`, `y_train`, `X_test`, and `y_test`. Training dataset is split into `X_train` and `y_train` according to the attribute feature values and the class value. Similarly, testing dataset is split into `X_test` and `y_test`. To obtain the first n columns, a command `.iloc[:,:n].values` can be used. For the rest of columns, a command `.iloc[:,n:].values` is used instead.

As an additional note, we also want to denote each class with a number; hence, for Iris Setosa, Iris Versicolor, and Iris Virginica, we propose them as 0, 1, and 2, respectively.

Training

An *one vs rest* strategy is used as the decision function for the training. `svm.SVC`, which is located in the sklearn library, supports the implementation. SVC implies an algorithm that classifies hyperplane of the dataset linearly. For non-linear separation, the approach is called as SVM.

According to the official scikit-learn website, the default value for the decision function is `ovr`, which denotes the *one vs rest* strategy. Because of that reason, we do not need to confuse the value of the decision function.

A classifier `clf` is firstly defined. After that, we want to fit both `X_train` and `y_train` as learning materials for the classifier. With this, scikit will process the classifier to execute the training process.

Testing

After a classifier `clf` being trained, we can use `clf.predict(X_test)` to get the predicted result. We can manually compare it with `y_test` and maintain a variable to count the number of missclassifications in the predicted result. However, another method `clf.score(X_test, y_test)` also exists to show directly the accuracy of the classification. Both methods serve the same purpose; hence, it does not matter to select which method that will be used.

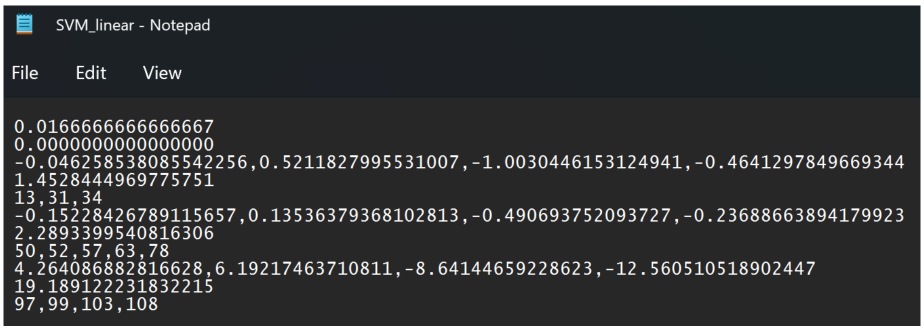
Aside from using `X_test`, in which we will get the information for the testing error, we can also use `X_train` to use the data that previously used as training materials to get the training error.

For the missclassification error, we simply count the number of missclassified data (according to the first described method). After that, we return that value divided by the number of original data. With an assumption that the the number of missclassified data is stored in a variable named $missclassifications$ In equation, the loss is shown as: $error = \frac{missclassifications}{|data|}$

Result and Analysis

Linear Kernel without Slack Variable

As asked in the question, we are to classify the iris class by using the standard SVM model (linear separator). As the used kernel do not use slack variable, `C` is set as a relatively high value such as 10^5. From the program, we obtain:



Based on the classification error in the training dataset, although it is seen that the accuracy is almost 100%, this still implies that the dataset is not linearly separable. However, for the testing dataset, the accuracy is shown as 100%. This finding is possible due to the number of testing data points that is much smaller than the number of training data points; hence, it is keen to be missclassified.

Linear Kernel with Slack Variable

As asked in the question, we are to classify the iris class by using the standard SVM model (linear separator). As the used kernel utilize slack variables, `C` is set as the required values, $[0.1, 0.2, ..., 1]$. From the program, we obtain:

SVM_slack - Notepad

File Edit View

```
0.0250000000000000
0.0333333333333333
-0.1470921538569825,0.3508769716586304,-0.7189940004257296,-0.3420055299780925
1.8921299070140811
10,13,14,15,31,34
0.29336431520205764
-0.15360322226233802,0.11058593459008659,-0.48024710009784577,-0.2207520412128427
2.3452434434132177
40,41,43,44,45,46,48,50,52,53,54,55,56,57,58,59,63,64,65,66,67,70,71,73,77,78
0.2704603293981356
-0.09176465640249276,0.08722690842768177,-1.105546183144634,-0.898991564830173
7.220857460580074
80,81,83,84,86,89,91,93,96,97,103,104,107,108,109,111,112,115,116,117,119
0.2806432599587305

0.0250000000000000
0.0333333333333333
-0.1546953806046849,0.39157525196948617,-0.7651819374495734,-0.35435917518989174
1.8964214397668673
13,14,31,34
0.29854137765418015
-0.15228426789115657,0.13536379368102813,-0.490693752093727,-0.23688663894179923
2.2893399540816306
43,45,46,48,50,52,53,55,56,57,58,59,63,64,65,66,67,71,73,78
0.281029805509726
0.07164497321760521,0.240000000000002,-1.3760315684645459,-1.162193297623472
7.457591188546857
80,81,83,89,91,93,96,97,103,104,107,108,111,112,116,117,119
0.28568933474783487

0.0166666666666667
0.0333333333333333
-0.08461491940959265,0.44526590762693274,-0.8404228999803343,-0.39549567986979794
```

If we focus on both training and testing error, we have:

Error Type	C = 0.1	C = 0.2	C = 0.3	C = 0.4	C = 0.5	C = 0.6	C = 0.7	C = 0.8	C = 0.9	C = 1.0
Training	0.025	0.025	0.167	0.167	0.025	0.025	0.025	0.025	0.025	0.025
Testing	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033

It is noticeable that all `C` values serve the same accuracy for the testing data. However, if we take a look when `C` $= 0.3$ or 0.4, they serve a better accuracy for the training data. Even with slack variable, the training error still have the same accuracy with the one without slack variable; however, it is noticeable that the testing error is still higher than the first method.

Polynomial Kernel (Second Order)

SVM_poly2 - Notepad

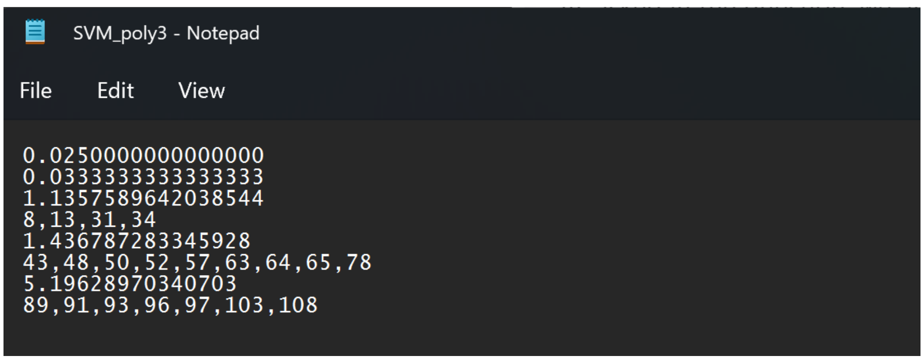
File Edit View

```
0.0250000000000000
0.0333333333333333
1.2957988860269352
13,14,31,34
1.5915205976595659
43,46,48,50,52,56,57,58,63,64,65,66,73,78
5.7293028470990714
80,83,89,91,93,96,97,103,108,112,116,119
```

Although this method have a relatively low training error, it still does not serve a better accuracy compared with the normal linear separator. However, this method is still descent and suggested as the accuracy reaches out to almost 98%.

Notice that the testing error is a bit higher compared to the training error. This is possible since the total data in the given training dataset is way lower than testing dataset, making it deviates either lower or higher than the expected average.

Polynomial Kernel (Third Order)



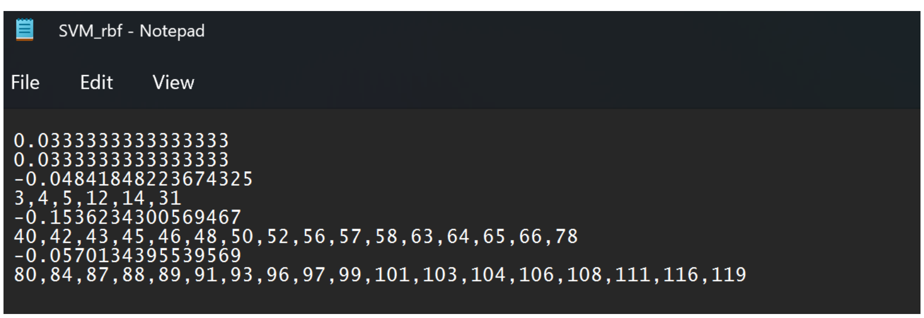
SVM_poly3 - Notepad

File Edit View

```
0.0250000000000000
0.0333333333333333
1.1357589642038544
8,13,31,34
1.436787283345928
43,48,50,52,57,63,64,65,78
5.19628970340703
89,91,93,96,97,103,108
```

Similar to the second order, the polynomial kernel for the third order also serves a comparatively same accuracies. With the same reasons, this method will still be descent.

Radial Basis Function Kernel



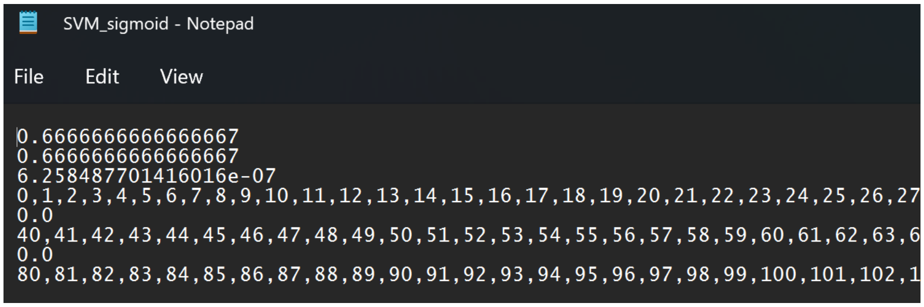
SVM_rbf - Notepad

File Edit View

```
0.0333333333333333
0.0333333333333333
-0.04841848223674325
3,4,5,12,14,31
-0.1536234300569467
40,42,43,45,46,48,50,52,56,57,58,63,64,65,66,78
-0.0570134395539569
80,84,87,88,89,91,93,96,97,99,101,103,104,106,108,111,116,119
```

Although the accuracy for the radial basis function kernel serves an arguably high percentage, it is still noticeable that the training error is slightly higher than all previous methods. However, as 96.7% is a highly great number, the method is still good for use if the datasets share similarities with the given ones.

Sigmoid Kernel



SVM_sigmoid - Notepad

File Edit View

```
0.6666666666666667
0.6666666666666667
6.258487701416016e-07
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
0.0
40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,6
0.0
80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,1
```

After looking at the result, especially for both training error and testing error parts, we notice that the errors are significantly higher than the other results. This truly implies that sigmoid method is deeply not suggested to be used for the given datasets.