دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس یادگیری ماشین

تمرین چهارم

نام و نام خانوادگی	**میلاد محمدی**
شماره دانشجویی	**810100462**
تاریخ ارسال گزارش	**1401/10/09**

سوال اول

We denote our points:

$$x_1 = \begin{matrix}-1\\0\end{matrix}, x_2 = \begin{matrix}0\\1\end{matrix}, x_3 = \begin{matrix}1\\0\end{matrix}$$

In order to find w and b, first we need to find α:

$$\max\ W(\alpha) = \sum_{i=0}^{n} \alpha_i - \frac{1}{2} \sum_{i=0,j=0}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$W(\alpha) = \alpha_1 + \alpha_2 + \ \alpha_3 - \frac{1}{2}\alpha_1^2 - \frac{1}{2}\alpha_2^2 - \frac{1}{2}\alpha_3^3 - \alpha_1\alpha_3$$

$$subject\ to: \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

This gives us: $\alpha_3 = \alpha_1 + \alpha_2$

$$W(\alpha) = 2\alpha_2 + 2\alpha_1 - \alpha_2^2 - 2\alpha_1^2 - 2\alpha_2\alpha_1$$

We take Lagrange to find extremum:

$$\frac{\delta W}{\delta \alpha_2} = 2 - 2\alpha_2 - 2\alpha_1 = 0$$

$$\frac{\delta W}{\delta \alpha_1} = 2 - 2\alpha_2 - 4\alpha_1 = 0$$

We can easily get: $\alpha_1 = 0, \alpha_2 = 1$ this gives us: $\alpha_3 = 1$

Now we can calculate w:

$$w = \sum_{i=1}^{s} \alpha_i y_i x_i$$

$$w = 0 + \left(1 * 1 * \begin{matrix}0\\1\end{matrix}\right) + \left(1 * (-1) * \begin{matrix}1\\0\end{matrix}\right) = \begin{matrix}-1\\1\end{matrix}$$

And by using formula below we get $b = 0$

$$\alpha_i(y_i((w.x_i) + b) - 1 = 0$$

This gives line equation as:

$$(-1\ \ 1)\begin{pmatrix}X_2\\X_3\end{pmatrix} + 0 = 0 \rightarrow x_2 = \ x_3 \quad \rightarrow \quad \text{SVs are: } x_2, x_3$$

سوال دوم

قسمت الف)

The sample $S = x^1, x^2, \dots, x^m$ includes m examples. The Kernel (Gram) matrix K is an $m \times m$ matrix including inner products between all pairs of examples i.e., $k_{i,j} = k(x_i, x_j)$.

$$k(x_i, x_j) = \varphi^T(x_i)\varphi(x_j) = \varphi^T(x_j)\varphi(x_i) = K(x_j, x_i)$$

قسمت ب)

$$(\| \varphi(x_i) - \varphi(x_j) \|)^2 \leq 2$$

$$(\| \varphi(x_i) - \varphi(x_j) \|)^2 = -2\varphi^T(x_i)\varphi(x_j) + \varphi^T(x_i)\varphi(x_i) + \varphi^T(x_j)\varphi(x_j)$$

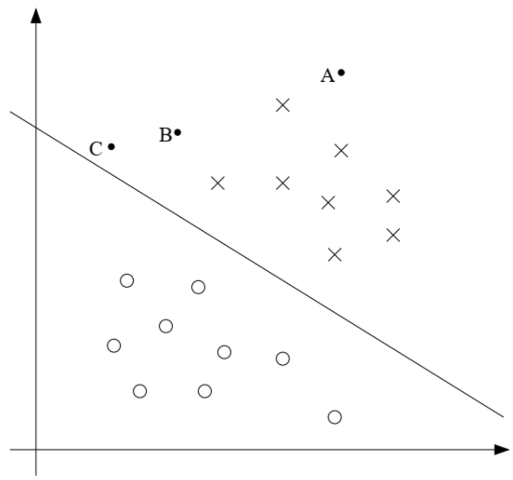
Since $K(x_i, x_i) = 1$:

$$(\| \varphi(x_i) - \varphi(x_j) \|)^2 = -2K(x_i, x_j) + 1 + 1, \qquad K(x_i, x_j) > 0$$

We simply get:

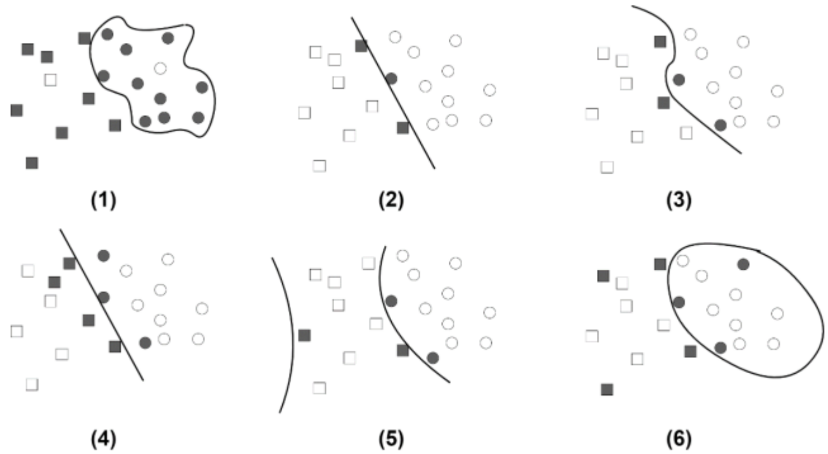
$$(\| \varphi(x_i) - \varphi(x_j) \|)^2 \leq 2$$

قسمت پ)



Given that SVM has found a separating line that has the maximum margin, to classify a new point such as A that is further away from the others, we need to obtain the $f(x; a, b)$. For all values that are on the decision boundary, the value is zero. Now, the only parameter that is important to determine whether this point is in the positive or negative class is the value of b.

سوال سوم



(2) Linear soft margin with C=1. It has a narrower margin and accepts less error.

(4) Linear soft margin with C=10. According to the SVs, it has a wider margin and accepts more error.

(6) Hard margin with $x_i.x_j + (x_i.x_j)^2$. This kernel creates a circular shape and corresponds to figure 6.

(1) Hard margin with $exp(-10\|xi - xj\|^2)$. Gamma is big, influence is big.

(3) Hard margin with $exp(-\frac{1}{10}\|xi - xj\|^2)$. Gamma is small, influence is small.

(5) Hard margin with $exp(-\frac{1}{10}\|xi - xj\|^2)$. Gamma is small, influence is small.

سوال چهارم

قسمت الف)

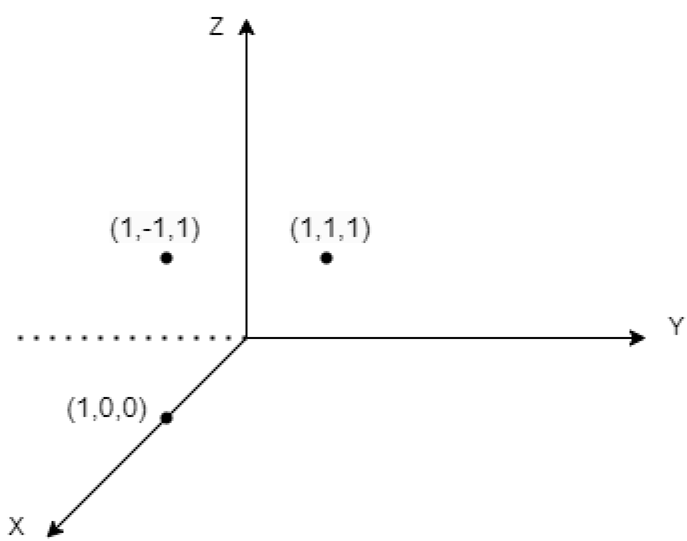
Using $\varphi(x)$ we get:

$$(1,0,0) \rightarrow \mathrm{y} = -1$$

$$(1,-1,1) \rightarrow \mathrm{y} = +1$$

$$(1,1,1) \rightarrow \mathrm{y} = +1$$

These points are easily separable with a plane: $z = \frac{1}{2}$



قسمت ب)

We change our constraints to equality type and use Lagrange:

$$L(w,\lambda) = \frac{1}{2} \| w \|_2^2 + \sum_{i=1}^{3} \lambda_i (y_i (w^T \varphi(x_i) + b) - 1)$$

$$\frac{\delta L(w,\lambda)}{\delta w} = w + \sum_{i=1}^{3} \lambda_i y_i \varphi(x_i) = 0$$

$$\frac{\delta L(w,\lambda)}{\delta b} = \sum_{i=1}^{3} \lambda_i y_i = 0$$

We get:

$$\mathrm{w}_1 - \lambda_1 + \lambda_2 + \lambda_3 = 0$$

$$-\lambda_1 + \lambda_2 + \lambda_3 = 0 \quad (*)$$

$(*) \rightarrow$ This gives us: $w_1 = 0$

By using our constraint, we get:

$$y_i(w^T \varphi(x_i) + b) = 1$$

$$\varphi(x_i) = (1{,}0{,}0) \rightarrow -1 * \left((w_1, w_2, w_3) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b \right) = 1 \rightarrow -w_1 - b = 1 \rightarrow b = -1$$

$$\varphi(x_i) = (1,-1{,}1) \rightarrow 1 * \left((w_1, w_2, w_3) \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + b \right) = 1 \rightarrow -w_2 + w_3 - 1 = 1 \qquad (**)$$

$$\varphi(x_i) = (1{,}1{,}1) \rightarrow 1 * \left((w_1, w_2, w_3) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + b \right) = 1 \rightarrow w_2 + w_3 - 1 = 1 \qquad (***)$$

$(**)(***) \rightarrow \; w_3 = 2, w_2 = 0$

$W = (0{,}0{,}2)$, $b = -1$

Margins is:

$$margin = \frac{1}{\| w \|} = \frac{1}{2}$$

سوال پنجم

قسمت الف)

For $n = 5$, we must have at least 3 votes:

$$\sum_{k=3}^{5} \binom{5}{k} (0.51)^k (1-0.51)^{5-k} \approx 0.51$$

قسمت ب)

For $n = 9$, we must have at least 5 votes:

$$\sum_{k=5}^{9} \binom{9}{k} (0.51)^k (1-0.51)^{5-k} \approx 0.52$$

قسمت پ) 100 درصد.

خیر. در محاسبات بالا فرض ما بر این بوده است که خطاها بین مدل‌ها ارتباطی ندارند ولی در واقعیت این فرض معمولا برقرار نیست.

قسمت ت)

$$\sum_{k=3}^{5} \binom{5}{k} (0.5)^k (1-0.5)^{5-k} \approx 0.5$$

استفاده از ensemble learning لزوما موجب افزایش دقت نهایی نمی‌شود.

1 Question 6

2 Part A

Linear: For Data that are linearly separable. Meaning there is a straight line that can be used as our classifier.
Polynomial: This kernel maps data to its polynomial feature space. And makes the data linearly separable in that dimension. It's used when data is not linearly separable.
RBF: RBF is used when there is no prior knowledge about data. It's very costly and is usually approximated by other algorithms.
Sigmoid:this function is equivalent to a two-layer, perceptron model of the neural network, which is used as an activation function for artificial neurons.

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
!pip install -U scikit-learn --user
```

[38]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import pandas as pd
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay, confusion_matrix
from sklearn.svm import SVC
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.model_selection import train_test_split
```

[9]:
```
df = pd.read_csv('/content/drive/MyDrive/iris.csv')
```

[10]:
```
df.loc[df["species"] == "setosa", "species"] = 0
df.loc[df["species"] == "versicolor", "species"] = 1
df.loc[df["species"] == "virginica", "species"] = 2
```

[11]:
```
y= df['species']
```

[12]:
```
X = df[['petal_length','petal_width']]
```

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=101)
```

3 Part B

i. SVM with Linear Kernel, one-vs-rest

[14]:
```
matplotlib.rcParams['figure.figsize'] = (15, 10)

feature_1, feature_2 = np.meshgrid(np.linspace(X_train['petal_length'].min(),␣
↪X_train['petal_length'].max()),np.linspace(X_train['petal_width'].min(),␣
↪X_train['petal_width'].max()))
grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T
svc = SVC(kernel='linear',decision_function_shape='ovr').
↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
y_pred = np.reshape(svc.predict(grid), feature_1.shape)
display = DecisionBoundaryDisplay(xx0=feature_1, xx1=feature_2, response=y_pred)
display.plot()
display.ax_.scatter(X_train['petal_length'], X_train['petal_width'], c=y_train,␣
↪edgecolor="black")

plt.show()
```

```
/root/.local/lib/python3.8/site-packages/sklearn/base.py:409: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

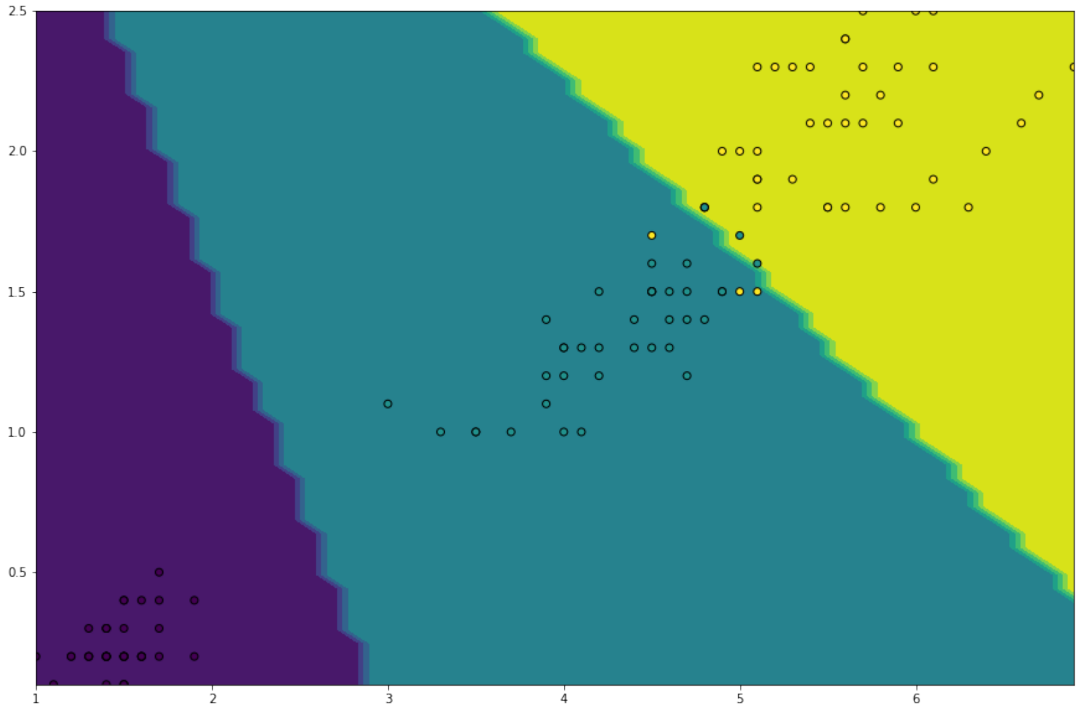


Figure 1: Area of classes with linear kernel(one-vs-rest)

ii. SVM with Linear Kernel, one-vs-one

[15]:
```
matplotlib.rcParams['figure.figsize'] = (15, 10)

feature_1, feature_2 = np.meshgrid(np.linspace(X_train['petal_length'].min(),␣
↪X_train['petal_length'].max()),np.linspace(X_train['petal_width'].min(),␣
↪X_train['petal_width'].max()))
grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T
svc = SVC(kernel='linear',decision_function_shape='ovo').
↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
y_pred = np.reshape(svc.predict(grid), feature_1.shape)
display = DecisionBoundaryDisplay(xx0=feature_1, xx1=feature_2, response=y_pred)
display.plot()
display.ax_.scatter(X_train['petal_length'], X_train['petal_width'], c=y_train,␣
↪edgecolor="black")

plt.show()
```

```
/root/.local/lib/python3.8/site-packages/sklearn/base.py:409: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

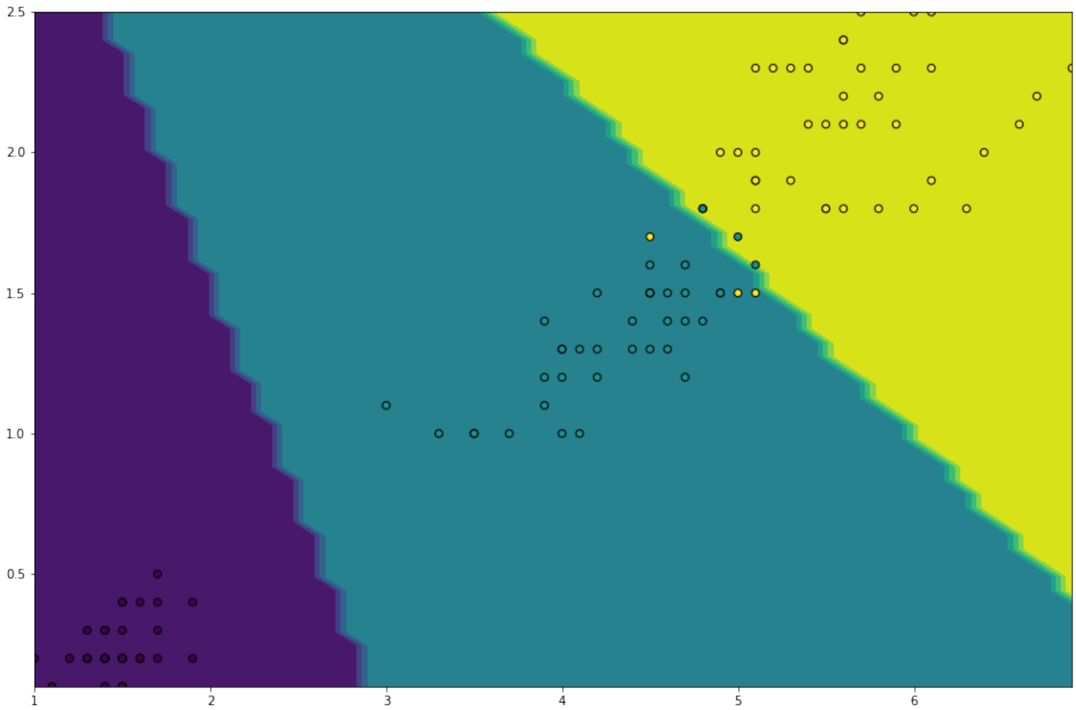


Figure 2: Area of classes with linear kernel (one-vs-one)

iii. SVM with RBF Kernel, one-vs-rest

[16]:
```
matplotlib.rcParams['figure.figsize'] = (15, 10)

feature_1, feature_2 = np.meshgrid(np.linspace(X_train['petal_length'].min(),␣
 ↪X_train['petal_length'].max()),np.linspace(X_train['petal_width'].min(),␣
 ↪X_train['petal_width'].max()))
grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T
svc = SVC(kernel='rbf',decision_function_shape='ovr').
 ↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
y_pred = np.reshape(svc.predict(grid), feature_1.shape)
display = DecisionBoundaryDisplay(xx0=feature_1, xx1=feature_2, response=y_pred)
display.plot()
display.ax_.scatter(X_train['petal_length'], X_train['petal_width'], c=y_train,␣
 ↪edgecolor="black")

plt.show()
```

```
/root/.local/lib/python3.8/site-packages/sklearn/base.py:409: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

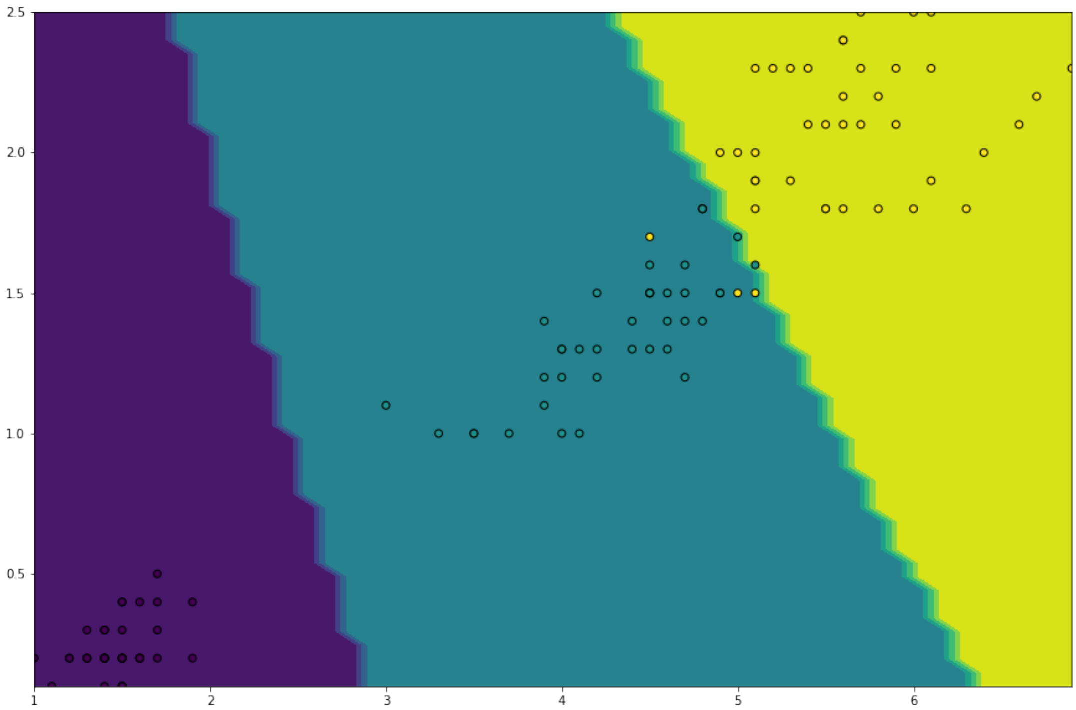


Figure 3: Area of classes with RBF kernel (one-vs-rest)

iv. SVM with Polynomial Kernel(d=5), one-vs-rest

[18]:

```
matplotlib.rcParams['figure.figsize'] = (15, 10)

feature_1, feature_2 = np.meshgrid(np.linspace(X_train['petal_length'].min(), X_train['petal_length'].max()),np.linspace(X_train['petal_width'].min(), X_train['petal_width'].max()))
grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T
svc = SVC(kernel='poly',degree=5,decision_function_shape='ovr').fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
y_pred = np.reshape(svc.predict(grid), feature_1.shape)
display = DecisionBoundaryDisplay(xx0=feature_1, xx1=feature_2, response=y_pred)
display.plot()
display.ax_.scatter(X_train['petal_length'], X_train['petal_width'], c=y_train, edgecolor="black")

plt.show()
```

```
/root/.local/lib/python3.8/site-packages/sklearn/base.py:409: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

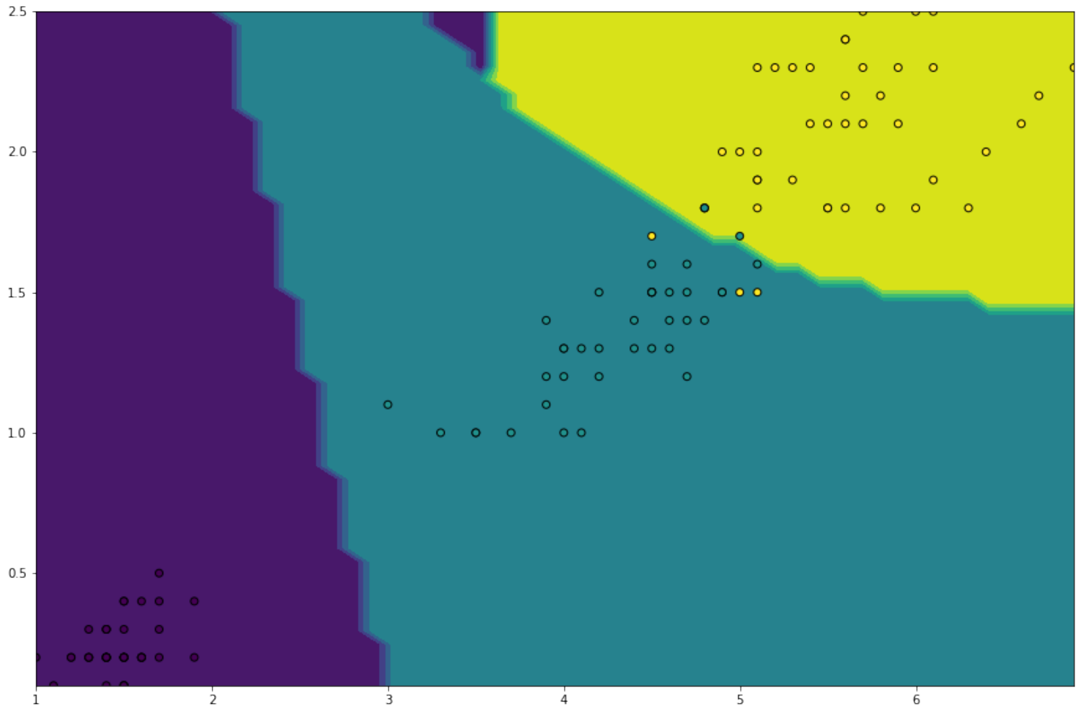


Figure 4: Area of classes with polynomial kernel (one-vs-rest)

4 Part C

i. Linear

[42]:
```
svc = SVC(kernel='linear',decision_function_shape='ovr').
 ↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
```

[43]:
```
y_pred_train = svc.predict(X_train)
print('Accuracy on training dataset:')
accuracy_score(y_train.astype('int'), y_pred_train)
```

```
Accuracy on training dataset:
```

[43]: 0.95

[44]:
```
y_pred_test = svc.predict(X_test)
print('Accuracy on test dataset:')
accuracy_score(y_test.astype('int'), y_pred_test)
```

```
Accuracy on test dataset:
```

[44]: 1.0

[45]:
```
cm = confusion_matrix(y_test.astype('int'), y_pred_test, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

[45]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f2d0f3b7790>

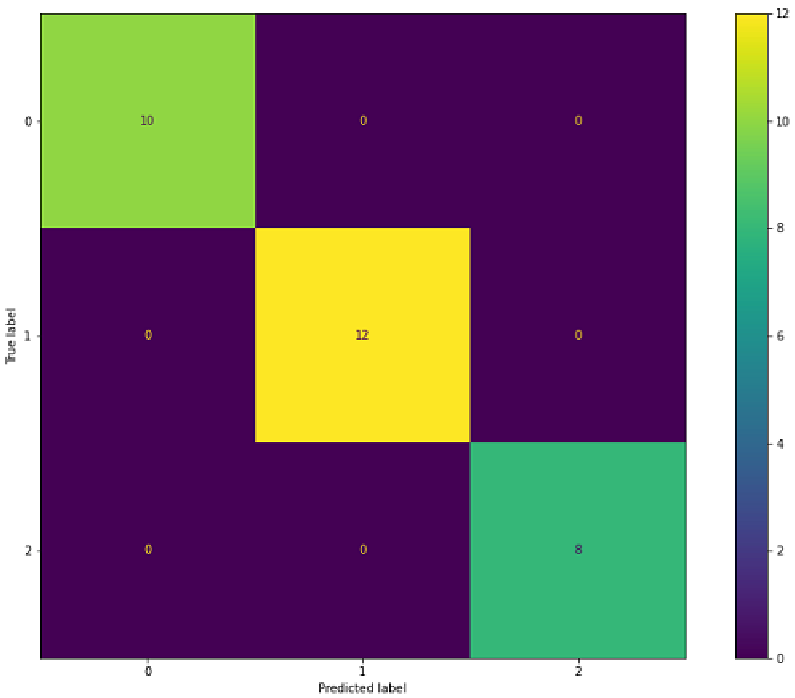


Figure 5: Confusion matrix of test data with linear kernel

ii. RBF

[46]:
```
svc = SVC(kernel='rbf',decision_function_shape='ovr').
 ↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
```

[47]:
```
y_pred_train = svc.predict(X_train)
print('Accuracy on training dataset:')
accuracy_score(y_train.astype('int'), y_pred_train)
```

```
Accuracy on training dataset:
```

[47]:
```
0.9416666666666667
```

[48]:
```
y_pred_test = svc.predict(X_test)
print('Accuracy on test dataset:')
accuracy_score(y_test.astype('int'), y_pred_test)
```

```
Accuracy on test dataset:
```

[48]:
```
0.9333333333333333
```

[49]:
```
cm = confusion_matrix(y_test.astype('int'), y_pred_test, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

[49]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f2d0f3b1ee0>
```

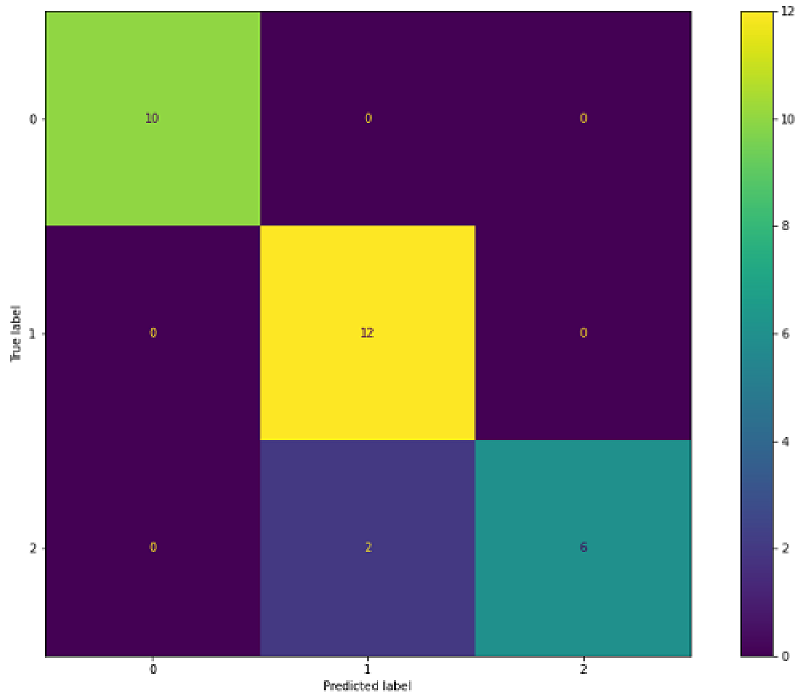


Figure 6: Confusion matrix of test data with RBF kernel

iii. Polynomial

[50]:
```
svc = SVC(kernel='poly',degree=5,decision_function_shape='ovr').
 ↪fit(X_train[['petal_length','petal_width']], y_train.astype('int'))
```

[51]:
```
y_pred_train = svc.predict(X_train)
print('Accuracy on training dataset:')
accuracy_score(y_train.astype('int'), y_pred_train)
```

```
Accuracy on training dataset:
```

[51]:
```
0.9583333333333334
```

[52]:
```
y_pred_test = svc.predict(X_test)
print('Accuracy on test dataset:')
accuracy_score(y_test.astype('int'), y_pred_test)
```

```
Accuracy on test dataset:
```

[52]:
```
0.9666666666666667
```

[53]:
```
cm = confusion_matrix(y_test.astype('int'), y_pred_test, labels=svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

[53]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f2d0e515d30>
```

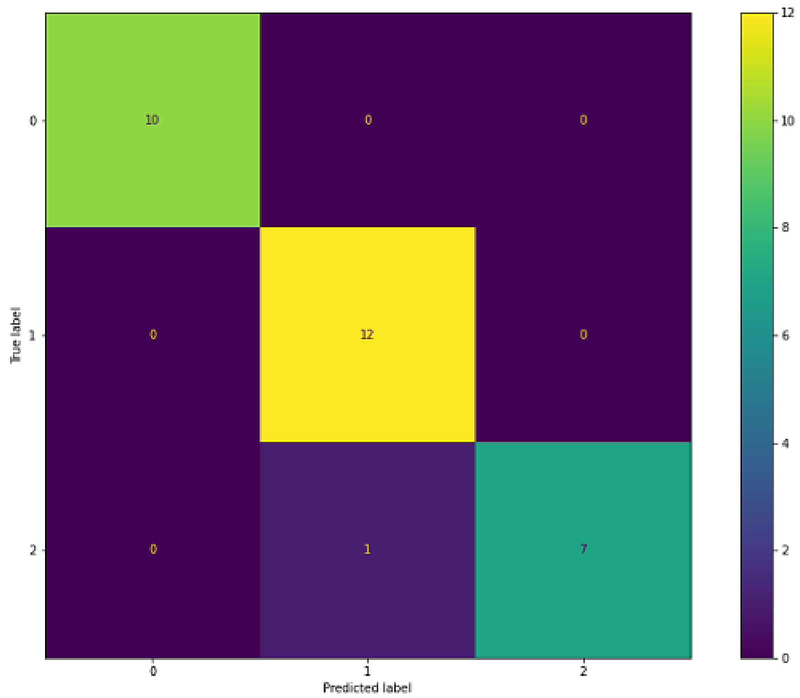


Figure 7: Confusion matrix of test data with polynomial kernel

iv. **Comparison**

v. **Our data has 3 classes. According to the figure below, it can be seen that these data are linearly separable (except for a few points). For this reason, the linear kernel is the most accurate among the kernels. Poly, rbf kernels cannot make a good distinction by taking the data in another space, and they are not as suitable as the linear kernel for these data.**

[56]:
```
sns.scatterplot(data=df, x="petal_length", y="petal_width", hue='species')
```

[56]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2d0e42b7f0>

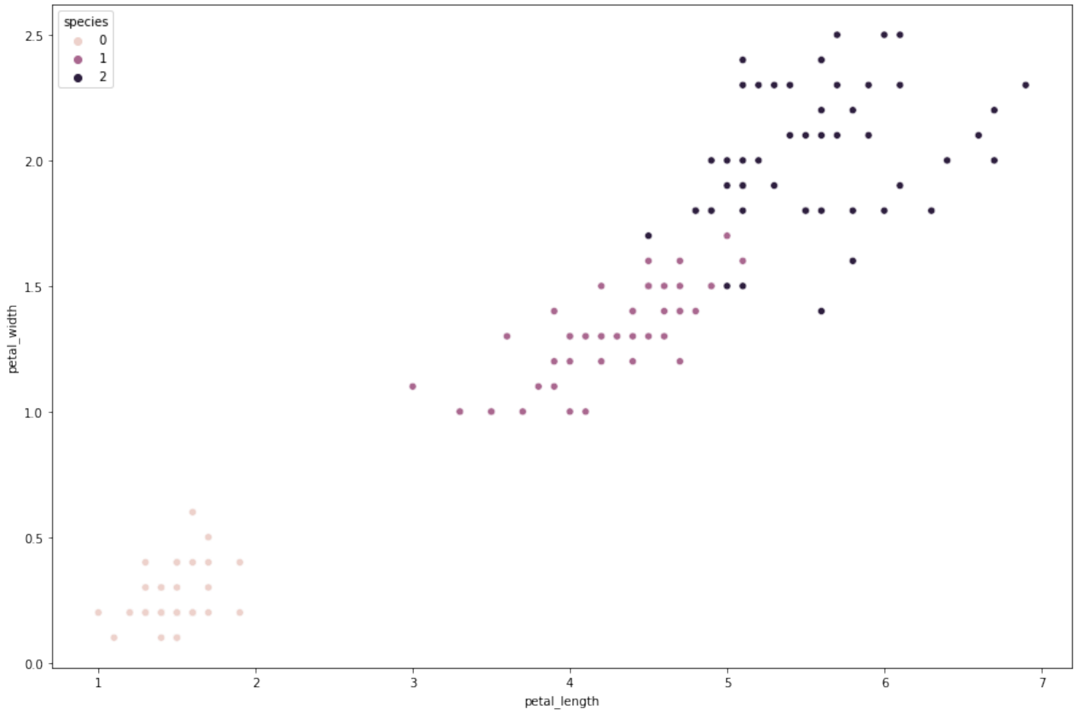


Figure 8: Data distribution

5 Question 7

[1]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn import svm
```

[2]:
```
x = np.linspace(0,2,100)
er = np.random.random_sample(size=100)/2 - 0.5

y = np.sin(x**2) + er
```

[3]:
```
fig, ax = plt.subplots()
plt.plot(x, y, 'o')

plt.show()
```

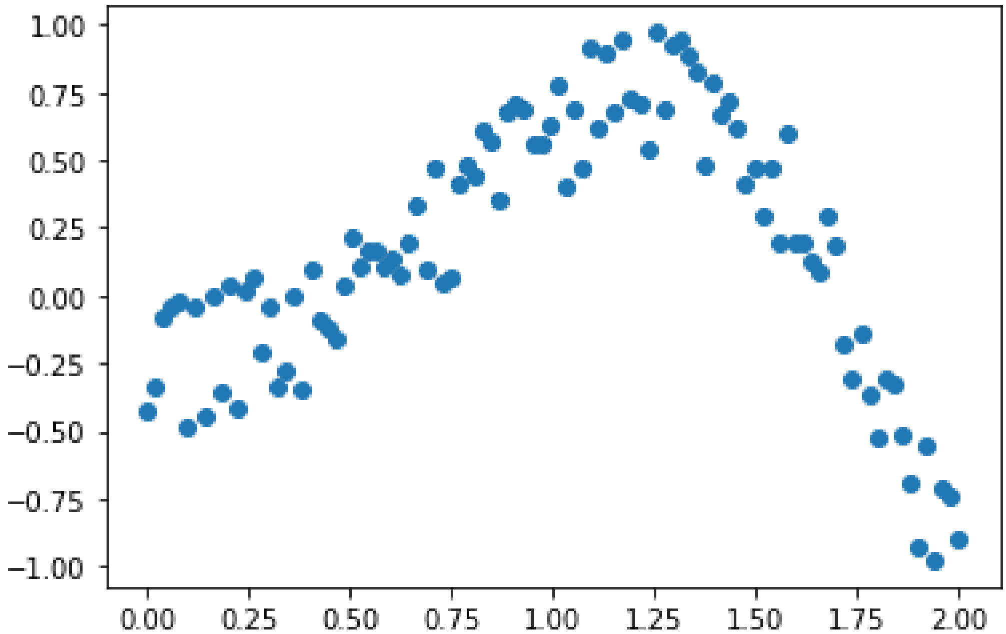


Figure 9: Data distribution

[4]:
```
x = x.reshape(-1,1)
```

6 Part A

[12]:
```
param_grid = {'C': [15,20,10,9,8,7,6,5,4,3,2,1,0.0001,0.00001,0.01,0.1],'gamma':
↪[1,2,3,4,5,6,7,8,9,10,0.00001,0.001,0.01,0.1]}
```

i. Best gamma and C values for polynomial kernel

[6]:
```
poly_regressor = svm.SVR(kernel='poly',degree=3,epsilon=0.1, coef0=1)

grid = GridSearchCV(poly_regressor, param_grid=param_grid, cv=5)
grid.fit(x, y)
opt_poly = grid.best_estimator_
print('The best gamma and C values for polynomial kernel:'+str(opt_poly))
```

```
The best gamma and C values for polynomial kernel:SVR(C=0.01, coef0=1, gamma=10,
kernel='poly')
```

ii. Best gamma and C values for RBF kernel

[13]:
```
rbf_regressor = svm.SVR(kernel='rbf')

grid = GridSearchCV(rbf_regressor, param_grid=param_grid, cv=5)
grid.fit(x, y)
opt_rbf = grid.best_estimator_
print('The best gamma and C values for RBF kernel:'+str(opt_rbf))
```

```
The best gamma and C values for RBF kernel:SVR(C=15, gamma=1)
```

iii. Best gamma and C values for linear kernel

[8]:
```
linear_regressor = svm.SVR(kernel='linear')

grid = GridSearchCV(linear_regressor, param_grid=param_grid, cv=5)
grid.fit(x, y)
opt_linear = grid.best_estimator_
print('The best gamma and C values for linear kernel:'+str(opt_linear))
```

```
The best gamma and C values for linear kernel:SVR(C=0.0001, gamma=1,
kernel='linear')
```

7 Part B

i. Polynomial

[9]:
```
plt.scatter(x, y)
plt.plot(x, opt_poly.predict(x), color = 'red')
plt.show()
```

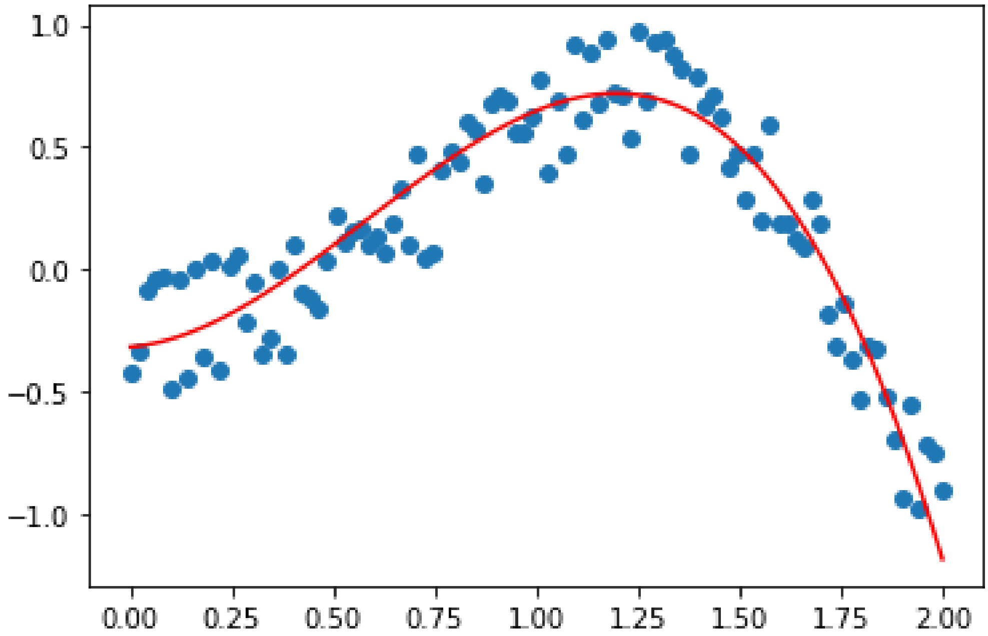


Figure 10: Graph estimated by polynomial kernel with input data

ii. RBF

[14]:
```
plt.scatter(x, y)
plt.plot(x, opt_rbf.predict(x), color = 'red')
plt.show()
```

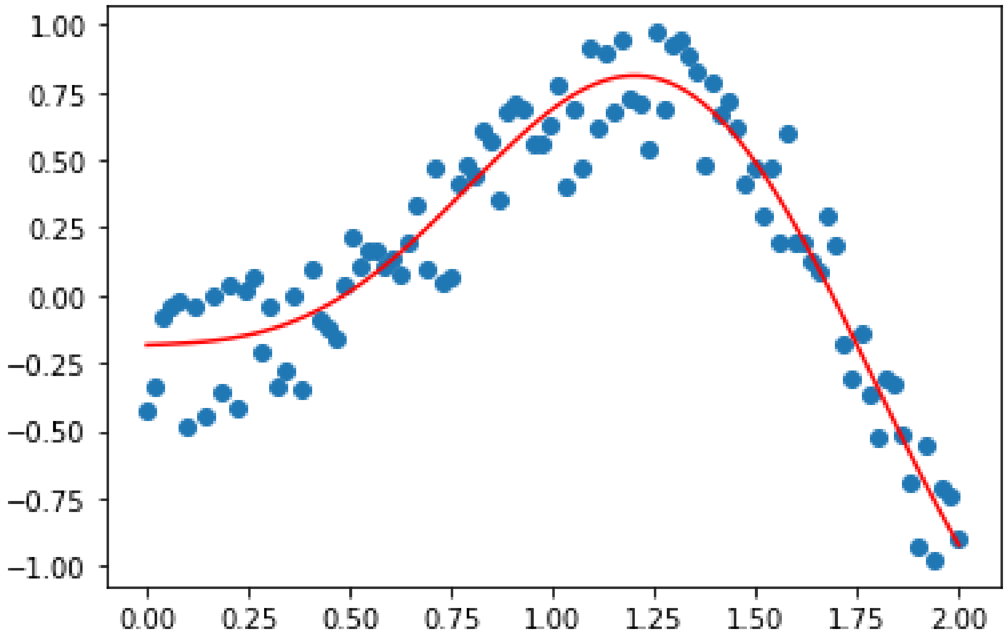


Figure 11: Graph estimated by RBF kernel with input data

iii. Linear

[11]:
```
plt.scatter(x, y)
plt.plot(x, opt_linear.predict(x), color = 'red')
plt.show()
```

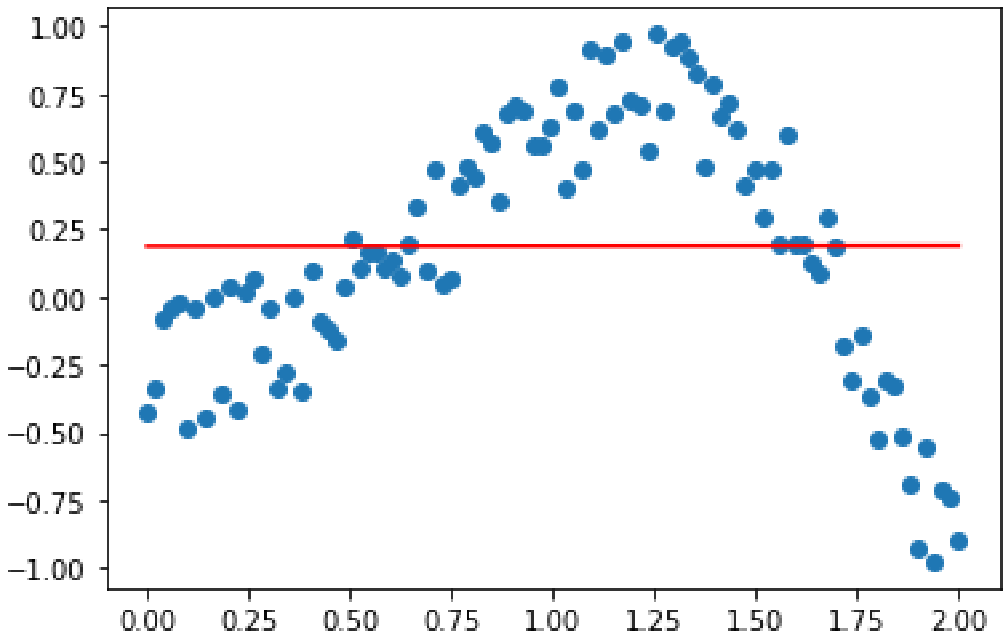


Figure 12: Graph estimated by linear kernel with input data

a: Comparison

[18]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

[23]:
```
print('error of polynomial kernel = '+ str(mean_squared_error(y, opt_poly.
 ↪predict(x))))
print('error of linear kernel = '+str(mean_squared_error(y, opt_linear.predict(x))))
print('error of rbf kernel = '+str(mean_squared_error(y, opt_rbf.predict(x))))
```

```
error of polynomial kernel = 0.028307844545877373
error of linear kernel = 0.22710749630541716
error of rbf kernel = 0.023383956438304408
```

i. **According to the shape of the function, the linear kernel has the worst result and cannot predict the curve correctly. The rbf kernel moves the data to a space with more dimensions than the polynomial and has less error than the others.**