National University of Singapore
School of Computing
Tutorial 2:
SUPPORT VECTOR MACHINE

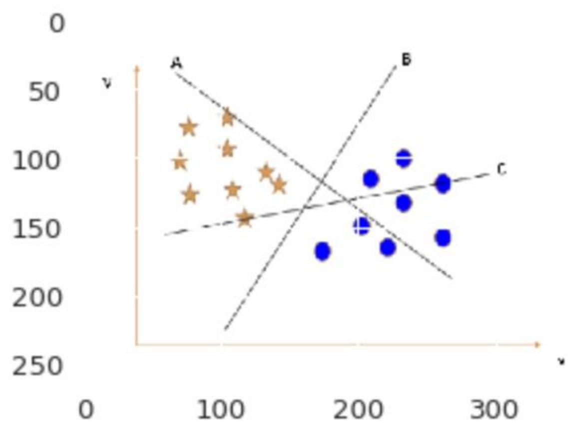
Introduction to Support Vector Machine

Supervised Machine Learning algorithm used to split data. It takes as input data and output a plane (or line) called SVM which separates the data. Its separation is based on some data points (not all data points). These data points are called the support vectors.

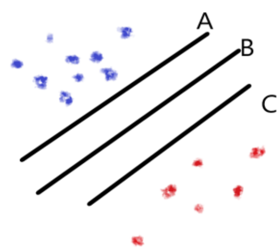
It uses geometry to solve the problem. Plots each data item in n-dimensional space, with values of each feature being particular coordinate. Then finds plane (called SVM) which differentiates different classes.

For multi-class classification problem, it forms n * (n-1)/2 classifiers i.e. separate based on "In class" or "Not in class".

1. Which line can be considered as SVM? (A or B or C)?
   Ans: B



2. All of the lines below will separate the data. Which of these lines is considered as SVM?
   Ans: B



3. Use sklearn SVM.svc classifier for the following classification:

Classify whether a recipe is of Muffin or Cupcakes:

a. You are provided with CSV file “recipes_muffins_cupcakes.csv”. Read the CSV file into Pandas dataframe.

```
import pandas as pd
recipes = pd.read_csv("recipes_muffins_cupcakes.csv")
print(recipes.describe())
```

b. Divide the data into train_data, train_labels, test_data, test_labels keeping train to test data ratio to be 80:20.

```
from sklearn.model_selection import train_test_split

X, y = mf_cp[['Flour', 'Sugar', 'Milk']], mf_cp[['Type']]
np.random.seed(2)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)

print("Shape of X_train = ", X_train.shape)
print("Shape of X_test = ", X_test.shape)
```

c. Define an SVM.svc classifier and fit the train data.

```
model_svm = SVC(kernel='linear')
model_svm.fit(X_train, y_train)
y_pred = model_svm.predict(X_test)
print(np.column_stack((y_pred, y_test)))
```

d. Predict the test_data and print the accuracy of classification.

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred, y_test))
```

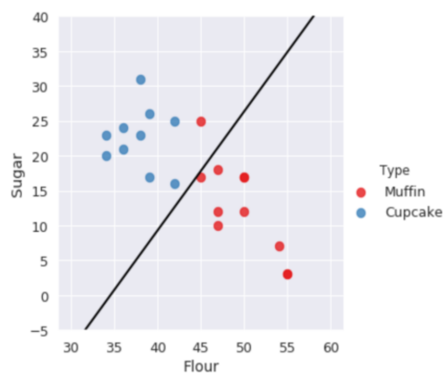
There are 3 important parameters of SVM.

- kernel
- C parameter
- gamma parameter

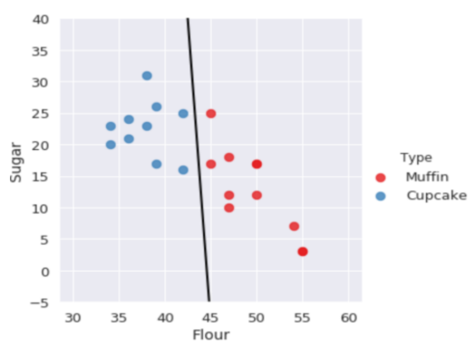
4. Which of these is a valid value of kernel parameter?
   A. Linear
   B. RBF
   C. Poly
   D. All the above

5. Which of the 2 SVMs below do you think have higher C parameter value?
   Ans: B

A.

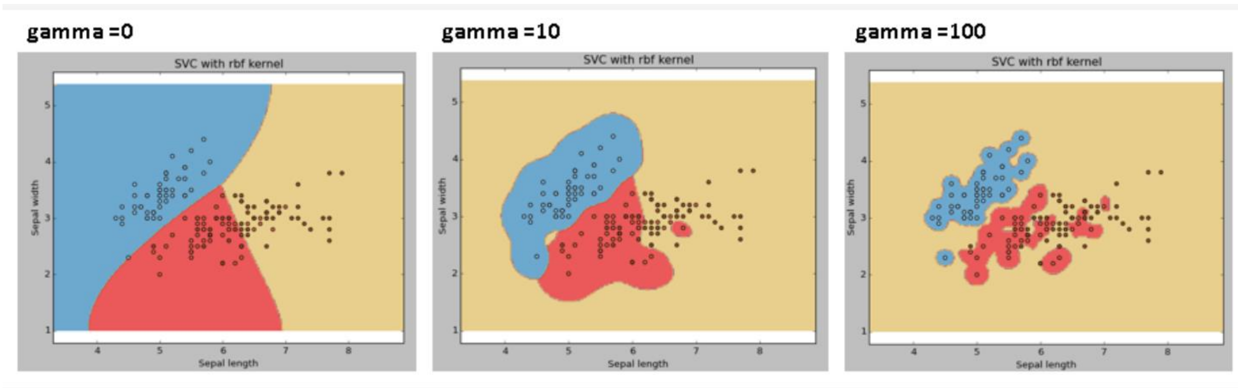


B.



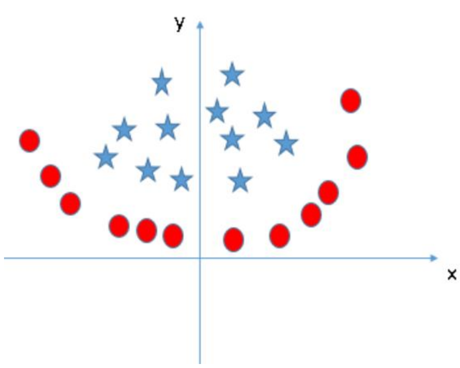
6. Which of the 3 pictures below have highest gamma values?
Ans: C

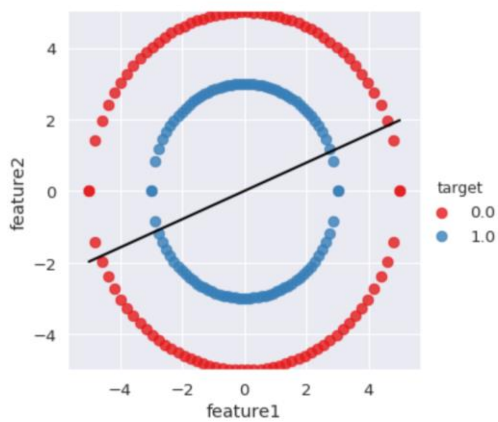
A. B. C.



7. What additional feature would be required to create a linear hyper-plane for segregating the classes in the two diagrams below?

Ans.
Fig 1: |x|
Fig 2: x^2 + y^2





8. Use sklearn svm.SVR classifier for time series data
   a. Use pandas ‘read_csv’ to read ‘price.csv’
      df = pd.read_csv('price.csv')
      df.head()

   b. Create X_train, X_test, y_train, y_test. Put 1st 15 points in training data and 5 points in testing data.

```
# Do not use train_test_split as it will randomise the data
# For time series data we do not want to randomize the data
# Correctly defining train and test

train_arr = np.arange(0, 15)
test_arr = np.arange(15, 20)
print("Train element selected ", train_arr)
print("Test element selected ", test_arr)
X_train, X_test, y_train, y_test = df['sequence'][train_arr], df['sequence'][test_arr],z
df['price'][train_arr], df['price'][test_arr]
print("Length of X_train = ", len(X_train))
print("Length of X_test = ", len(X_test))
```

c. Fit your data into svm.SVR using different kernels.
d. Plot your predicted fitted data vs actual data points.

```
def predict_prices(dates, prices):
    dates = np.expand_dims(dates, axis=1)
    svr_lin  = SVR(kernel='linear', C=1e3)
    svr_poly = SVR(kernel='poly', C=1e3, degree=2)
    svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)

    # Fit regression model
    svr_lin.fit(dates, prices)
    svr_poly.fit(dates, prices)
    svr_rbf.fit(dates, prices)

    plt.scatter(dates, prices, c='k', label='Data')
    plt.plot(dates, svr_lin.predict(dates), c='g', label='Linear model')
    plt.plot(dates, svr_rbf.predict(dates), c='r', label='RBF model')
    plt.plot(dates, svr_poly.predict(dates), c='b', label='Polynomial model')

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Support Vector Regression')
    plt.legend()
    plt.show()

    return svr_rbf, svr_lin, svr_poly
        svr_rbf, svr_lin, svr_poly = predict_prices(X_train, y_train)
```

e. Predict for test data and print Mean Square Error in data points predictions.

```
y_pred = svr_rbf.predict(np.expand_dims(X_test, axis=1))
print(np.column_stack((y_pred, y_test)))
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_pred, y_test))
```