Machine Learning Applications with Python

Neda Hantehzadeh

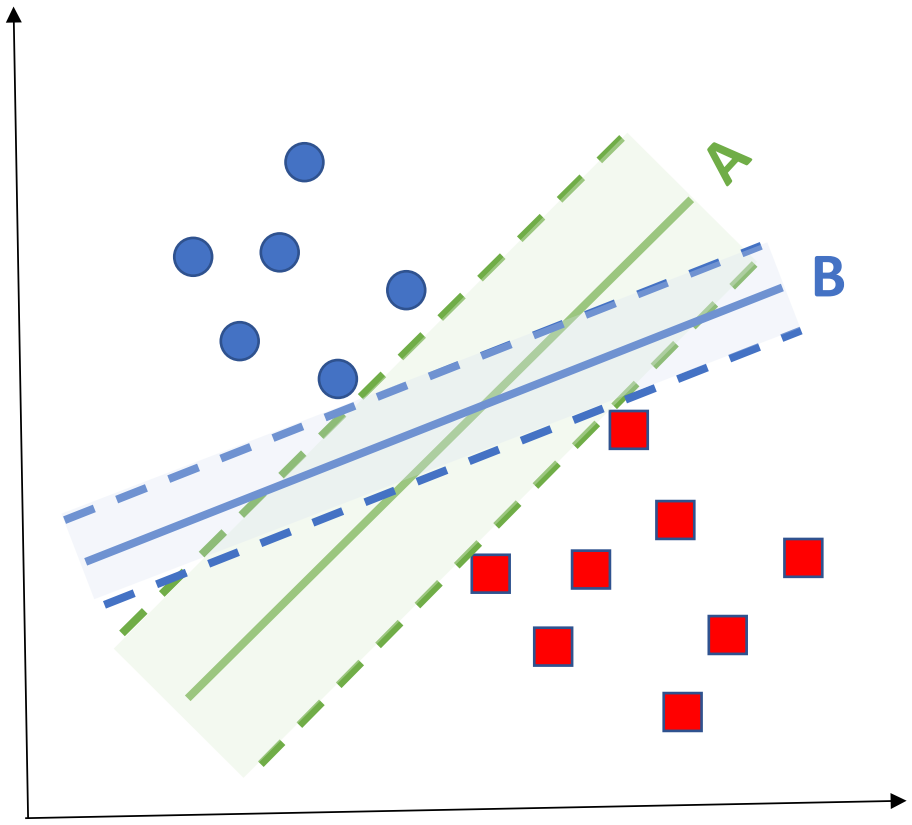
Week 6

Outline

- What is Support Vector Machine (SVM)?
- SVM Kernels and Kernel-trick
- Mathematics set up for SVM
- Application of SVM for classification
- Parameter tuning for SVM
- Application of SVM for Regression and learning curve

Solution of Mid-term Exam

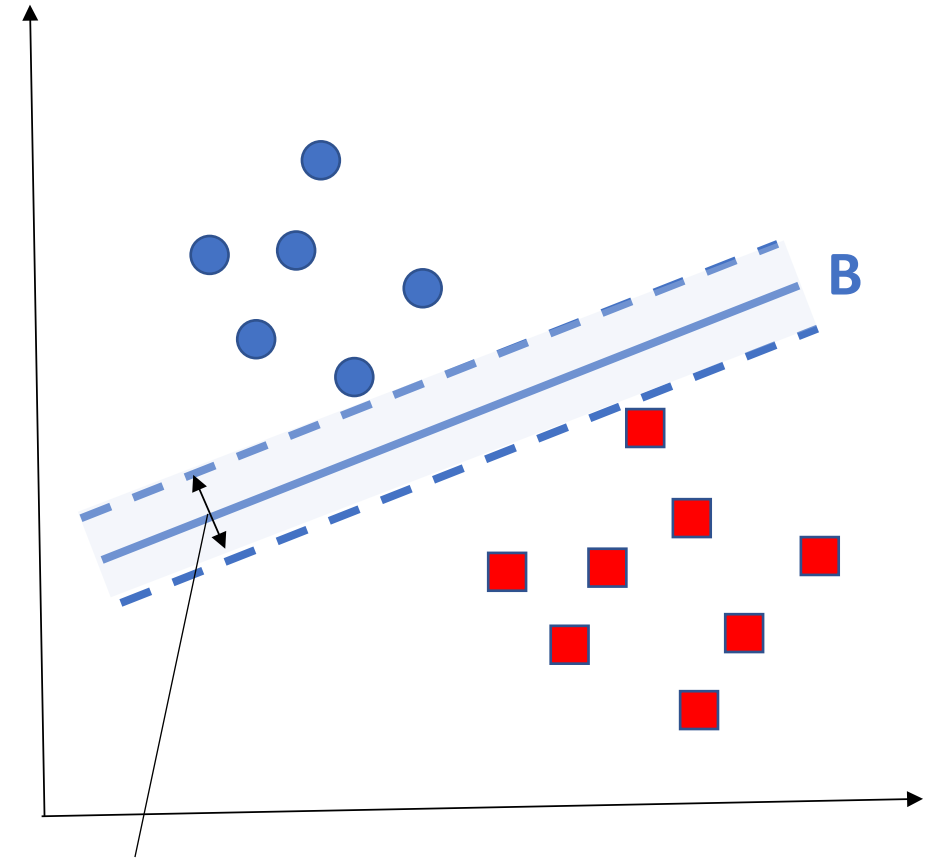
What is SVM?

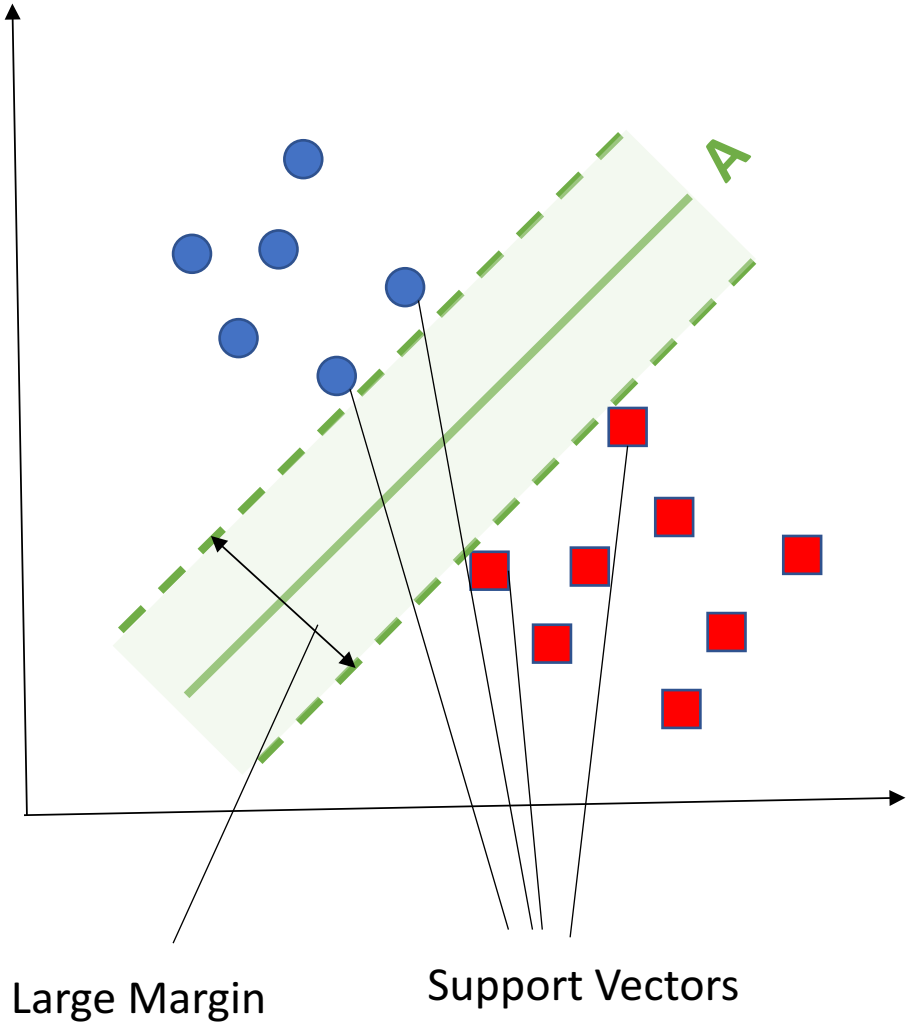


Applications:

- Classification
- Regression

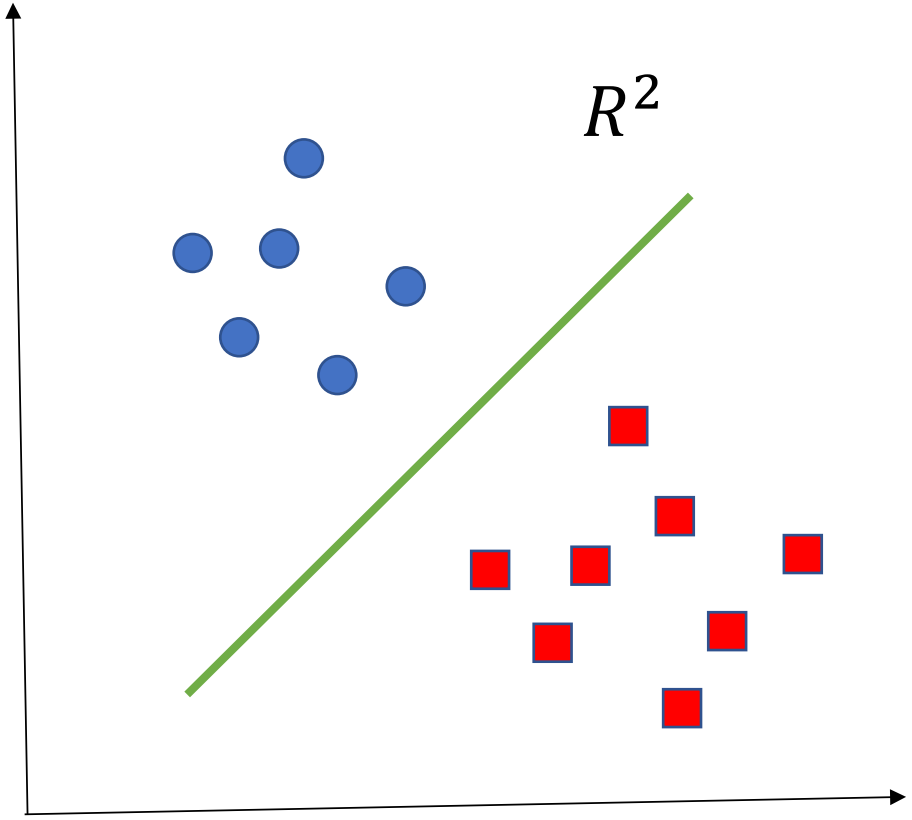
What is SVM?

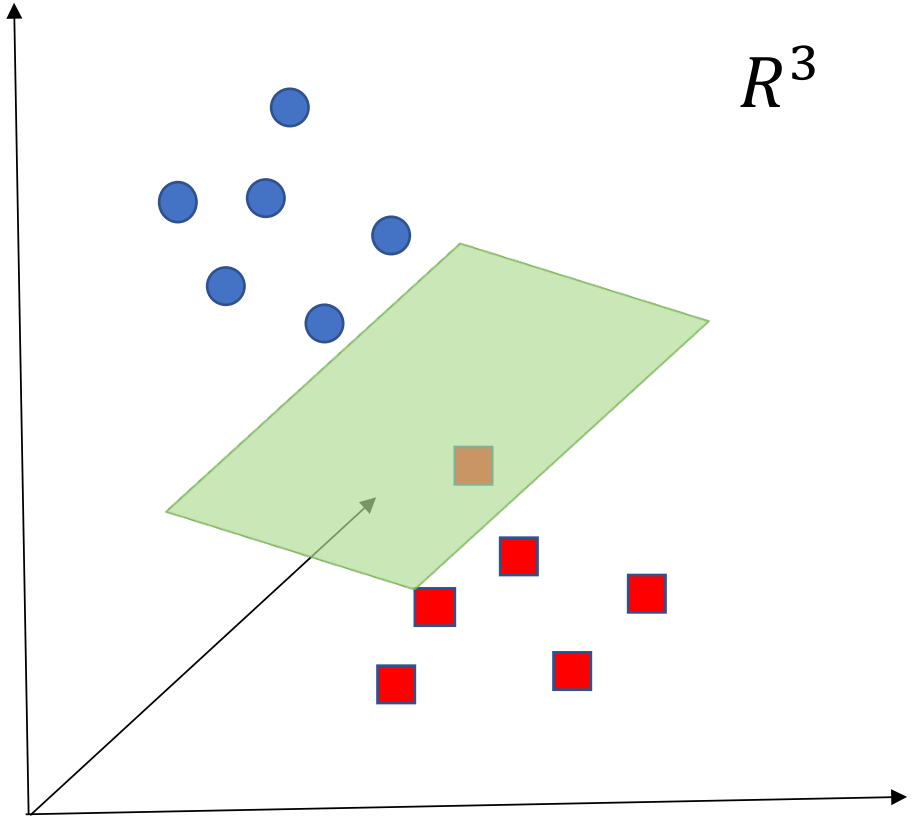




What is a hyperplane?

In N dimensional space , A hyperplane is a linear decision surface of dimension N-1 that best separates the classes points with maximum margin





Equation of a Hyperplane

- Equation of a line is $y = mx + c$
- Equation of a hyperplane is $w^T x = 0$

- Consider writing the equation of a line as $y - mx - c = 0$ then:

$$\text{W}=\begin{pmatrix} -c \\ -m \\ 1 \end{pmatrix} \text{ and } \mathrm{x} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

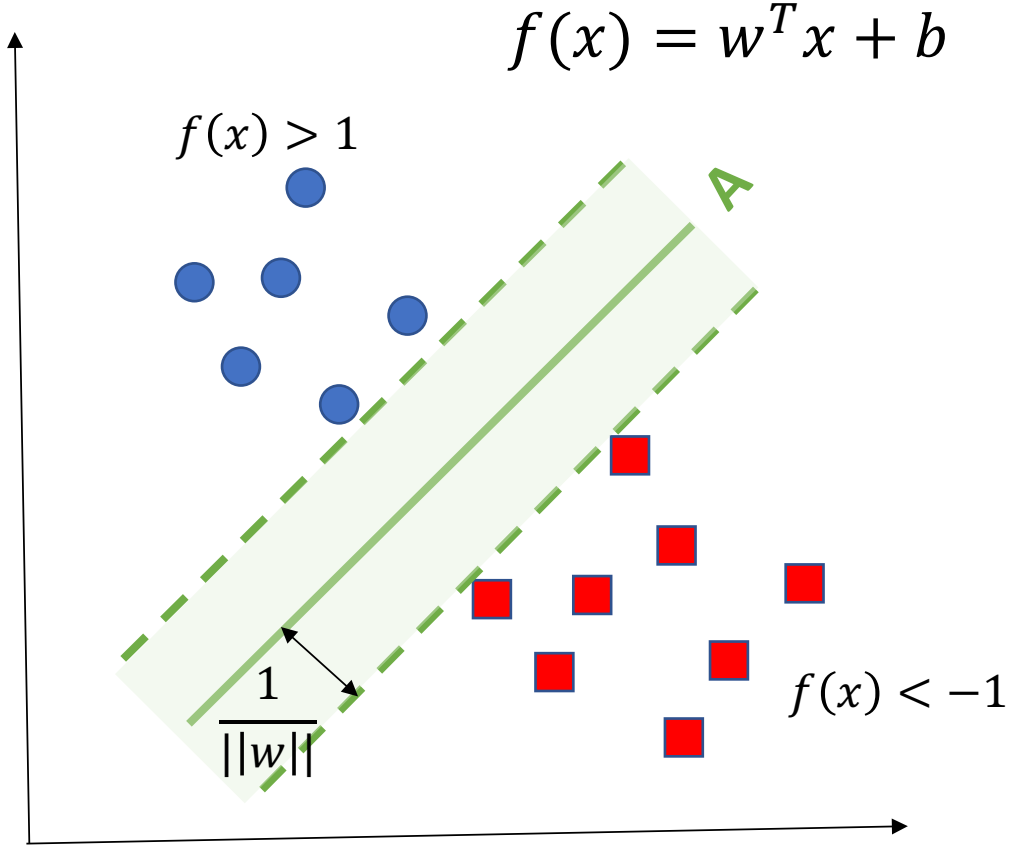
Mathematical Set up for SVM

Constrains:

$$f_i(wx + b) \geq 1 \; where \; 1 \leq i \leq n$$

Goal:

Maximize $\frac{2}{||w||}$

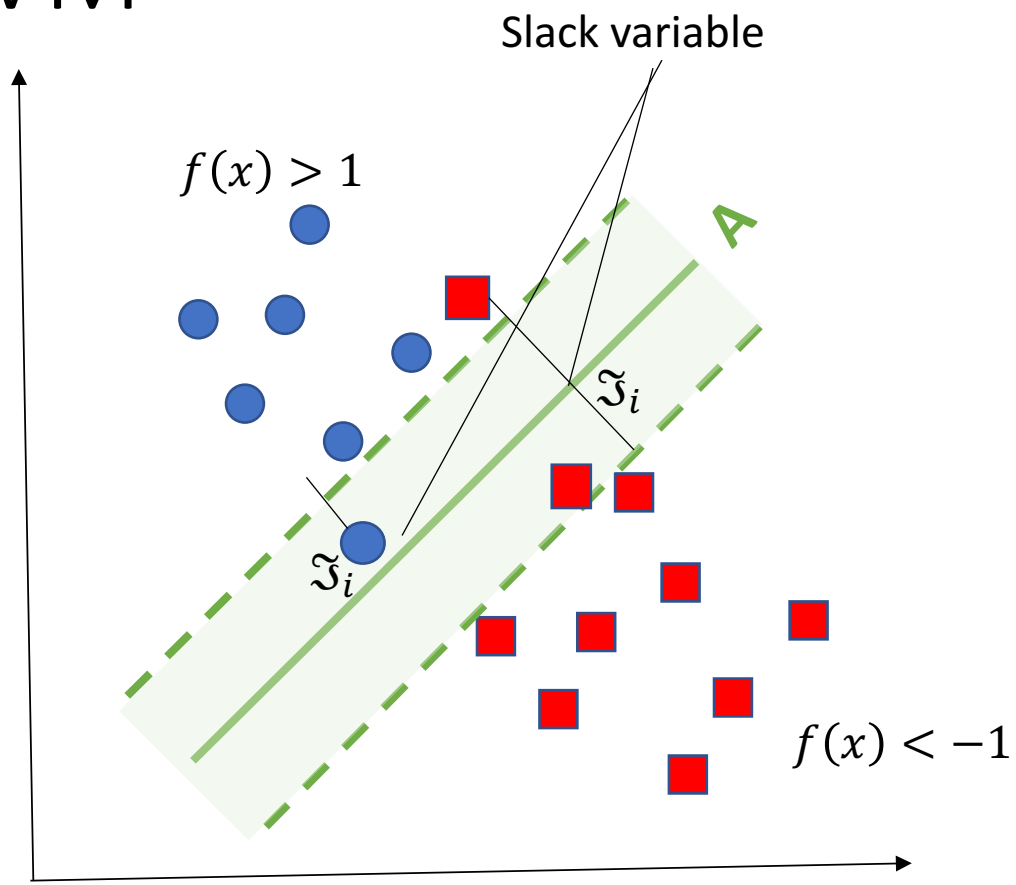


Optimization for minimize $(w, b)\{\frac{||w||^2}{2} \; subject \; to \; f_i(wx + b) \geq 1$
$for \; any \; i = 1,2, \ldots n$

Mathematical Set up for SVM

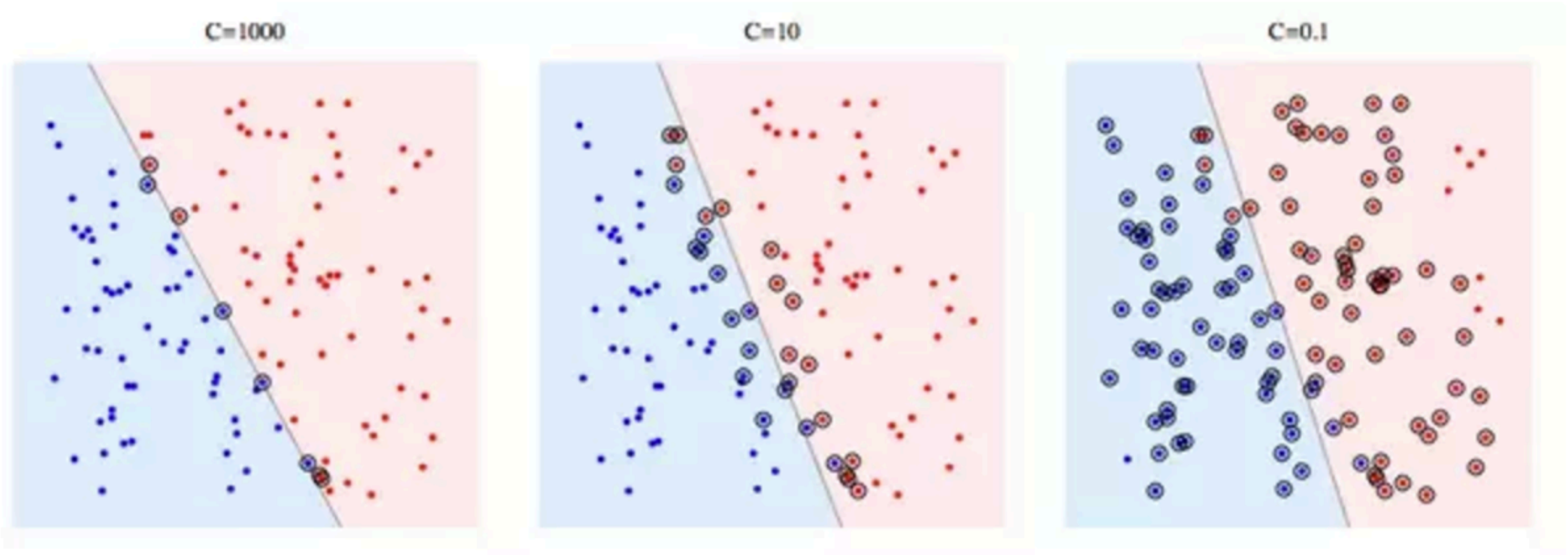
C is the **Regularization parameter that** measures how much to penalize the slack variable**:**

- If C→ 0 we don't penalize slack variable
- Large C ,penalize slack variable more, results in small margin
- Small C, penalize slack variable less, results in larger margin

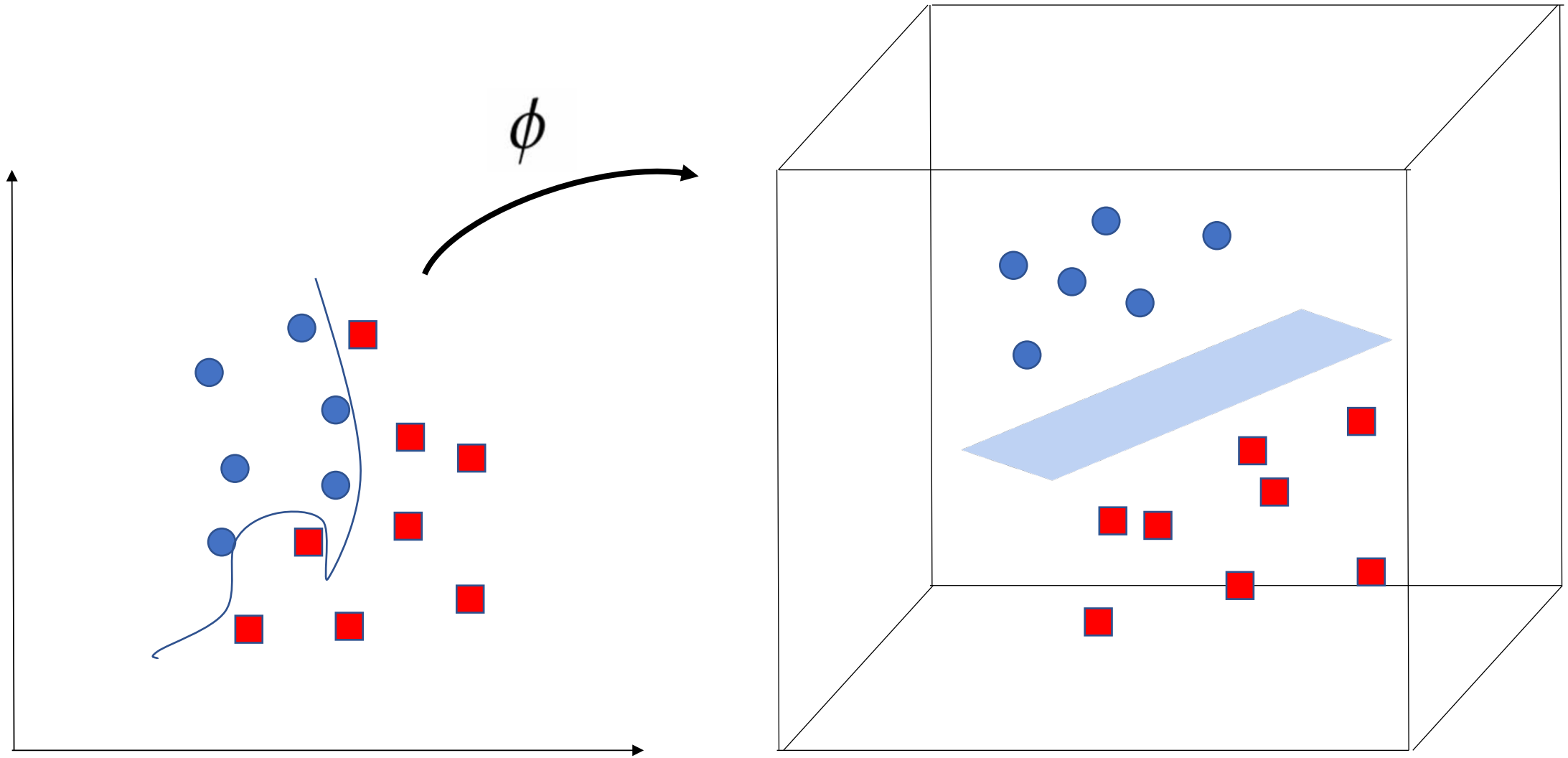


Optimization for minimize $(w, b)\{\frac{||w||^2}{2} + C\sum \Im_i \;\; subject\ to\ f_i(wx + b) \geq 1 - \Im_i$
$for\ any\ i = 1,2, \ldots n$

Examples of Regularization Parameter (C)



Non-Linear SVM - Kernel Trick



SVM Kernels

- A standard SVM is a type of linear classification using dot product but…
- In 1992, Boser, Guyan, and Vapnik replacing each dot product with a nonlinear kernel function (such as a Gaussian radial basis function or Polynomial kernel.

SVM Kernel Mathematics

If $\oplus$ is the kernel function that maps x_i to $\oplus\ (x_i)$, the linear SVM constrain changes to:

Optimization for minimize $(w, b)\{\frac{||w||^2}{2} + C\sum \Im_i\ \ subject\ to\ f_i(w.\oplus\ (x_i) + b) \geq 1 - \Im_i$
$for\ any\ i = 1,2, \ldots n$

SVM Kernel Examples

- Quadratic Kernel:

$$k(x,z) = (x^T z)^2 \quad or \quad (1 + x^T z)^2$$

degree

- Polynomial Kernel:

$$k(x,z) = (x^T z)^d \quad or \quad (1 + x^T z)^d$$

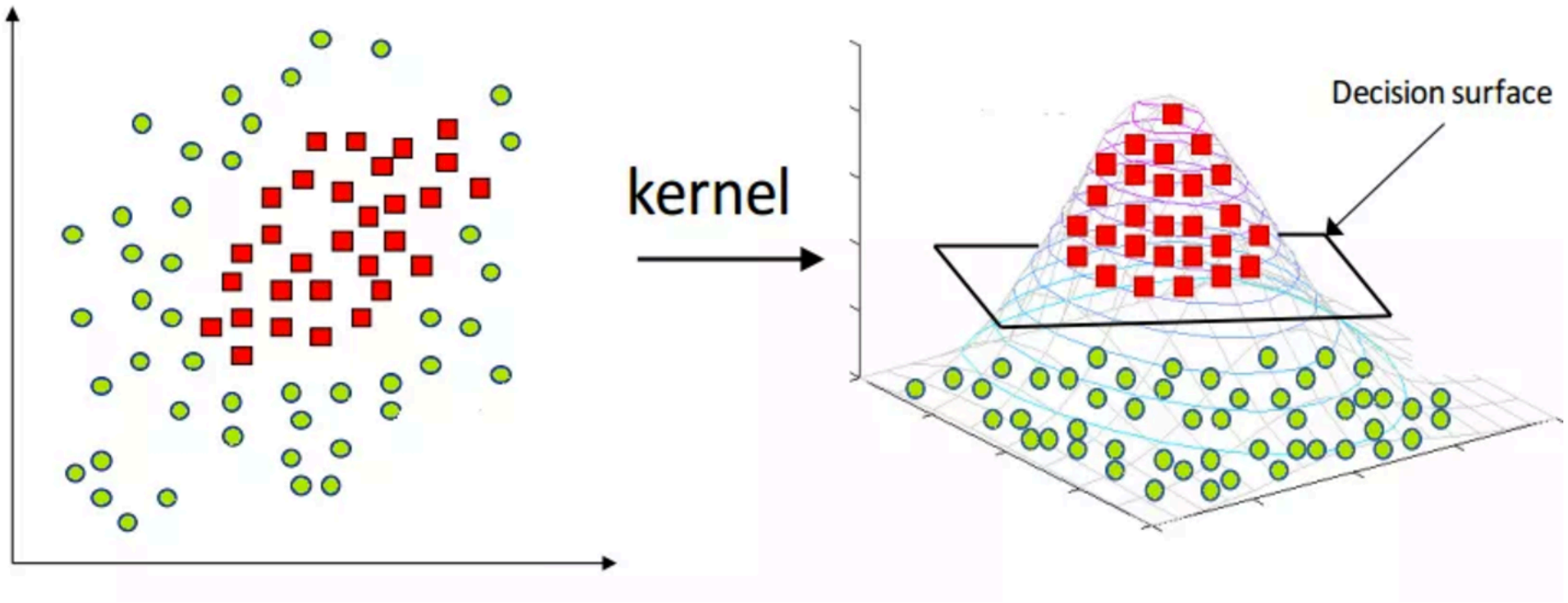
gamma

- Radial Basis Function (RBF):

$$k(x,z) = \exp[-\gamma ||\mathbf{x} - \mathbf{z}||^2]$$

You usually find the best d and Gamma hyper-parameters using Grid-Search

RBF Gamma Variable



Why SVM is a popular technique?

- While logistic regression, decision trees and … techniques are capable of solving linear problems , SVM is able to solve both linear and non-linear problems
- It can work with smaller dataset
- It’s complexity with the dimension of data increases linearly

Important tips to implement SVM for Python!

Data Preparation

- SVM requires the data to be numerical…Solution: all categorical data need to be converted to numerical data using ‘one-hot-encoding’ or ‘label-encoding’
- SVM in Python can not handle missing values. Solution: remove data with missing values or use some form of data imputing. The sklearn.preprocessing.Imputer
- Data needs to be normalized to range from 0 to 1 or -1 to 1. Solution: The sklearn.preprocessing.StandardScaler module can use used for such normalization.

SVM for Multiclass Classification

- **One vs. Rest Approach (OvR):** This strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

- **One vs. One Approach (OvO):** one trains $K(K-1)/2$ binary classifiers for a K-way multi-class problem; each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes.

In svm.svc implementation, ’decision_function_shape’ parameter provides the option to choose one of two strategy. Default is OvO.

Iris data set Example

- Number of classes: 3

-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

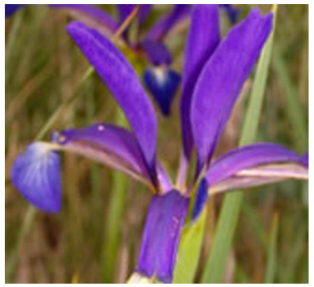
- Attributes: 4
  1. sepal length in cm
  2. sepal width in cm
  3. petal length in cm
  4. petal width in cm

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936



Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1476008

https://archive.ics.uci.edu/ml/datasets/iris

Data Load and Preparation

```
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features.
y = iris.target
print (X.shape)
print (y.shape)
```

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
 np.arange(y_min, y_max, h))
X_plot = np.c_[xx.ravel(), yy.ravel()]
```

SVM with different Kernels

C is the Cost or Regularization Parameter

```
# Create the linear SVC model object
C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=C, decision_function_shape='ovr').fit(X, y
Z = svc.predict(X_plot)
Z = Z.reshape(xx.shape)
```

Could be ‘ovr’ or ‘ovo’

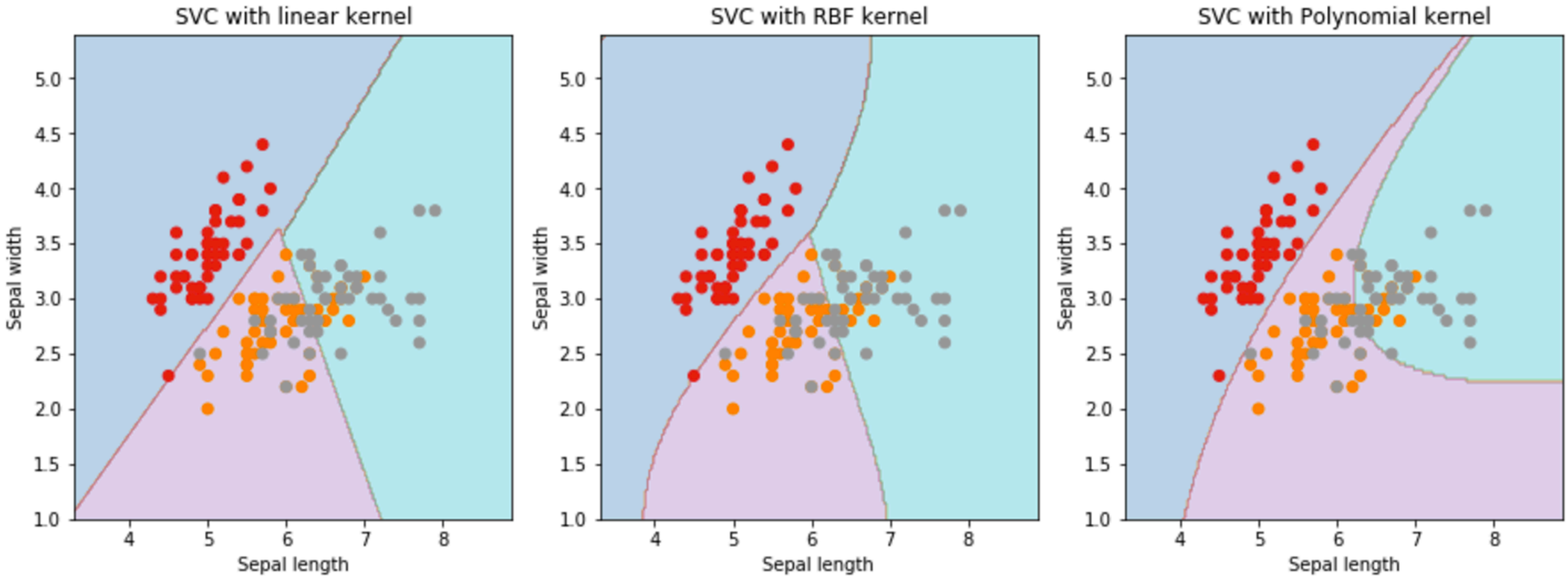
```
# Create the rbf non-linear SVC model object
C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='rbf', C=C, decision_function_shape='ovr').fit(X, y)
```

```
# Create the non-linear polynomial SVC model object
C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='poly', degree=3, C=C, decision_function_shape='ovr').
```

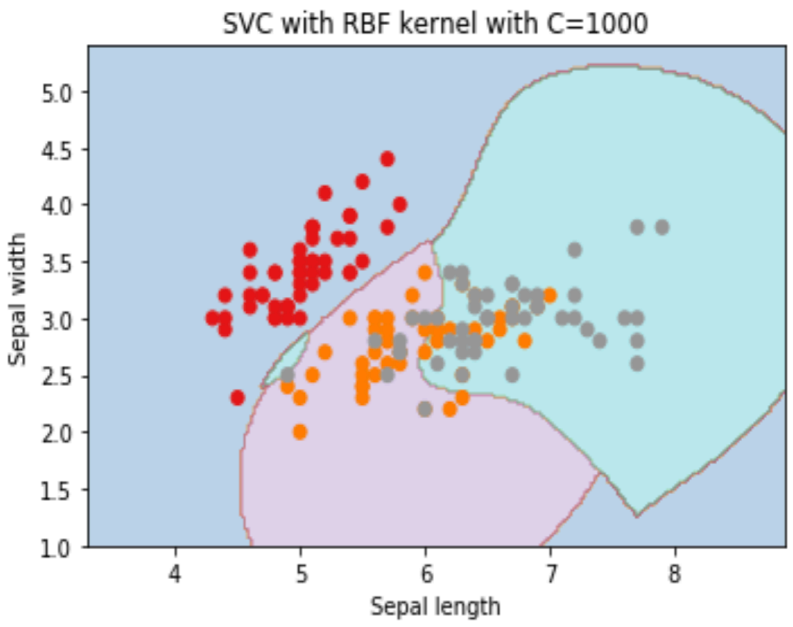
Plot of the Results

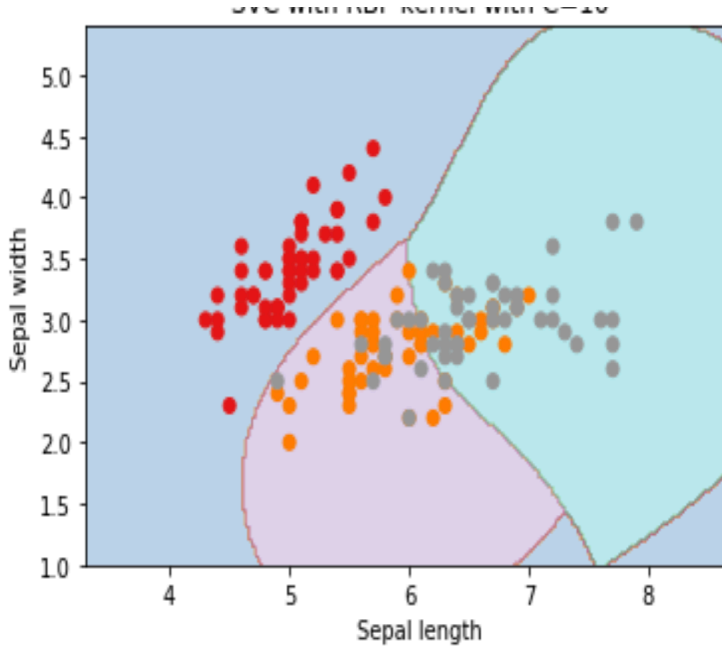
```
plt.contourf(xx, yy, Z, cmap=plt.cm.tab10, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with Polynomial kernel')
```

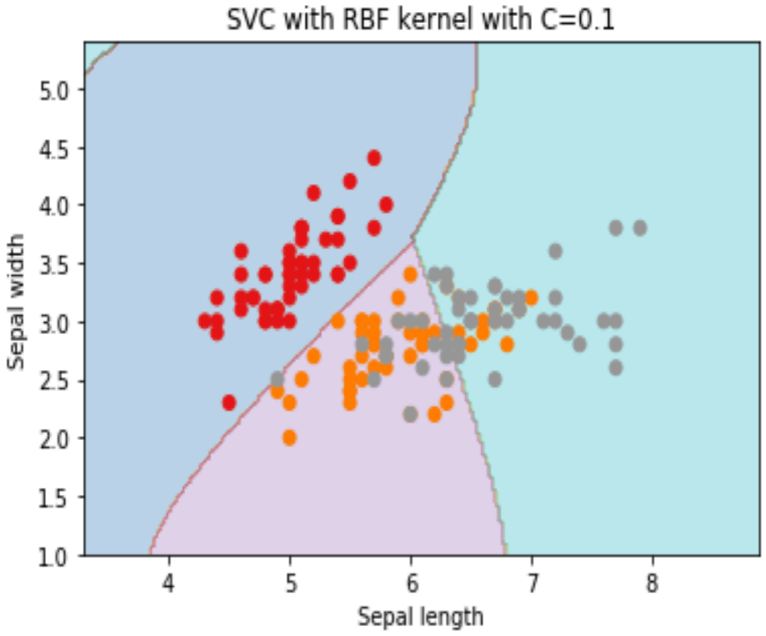
Plot of the Results



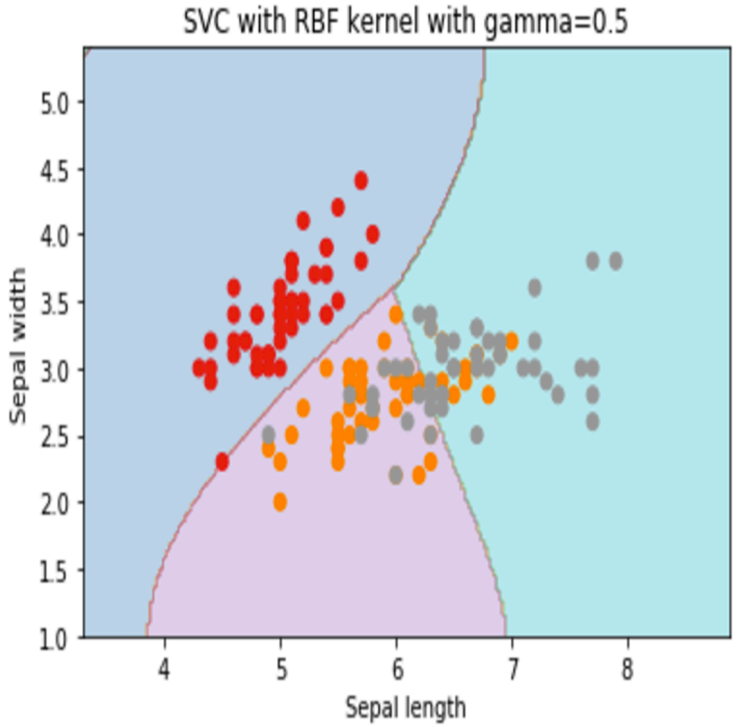
Effect of C and Gamma Variable in RBF Kernel

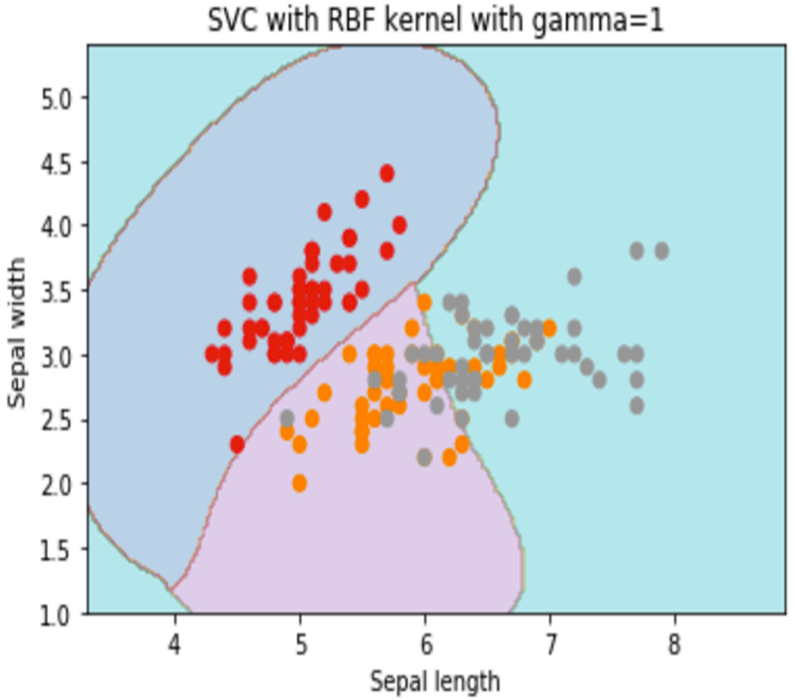


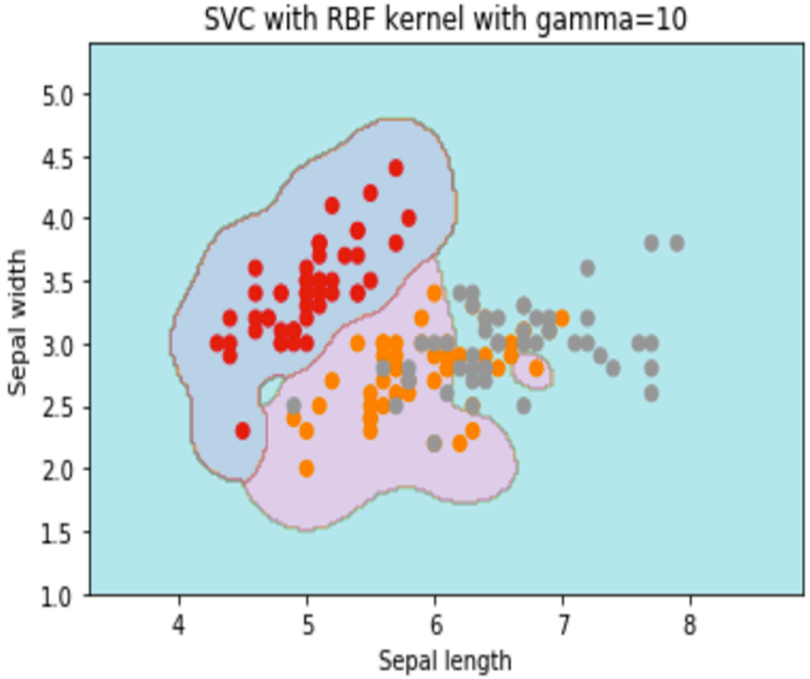




Effect of C and Gamma Variable in RBF Kernel



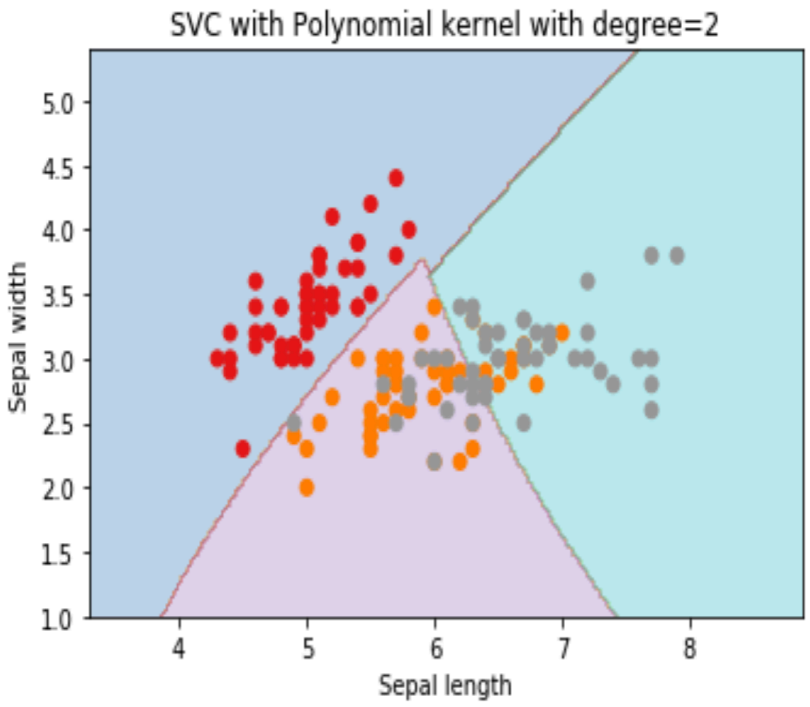


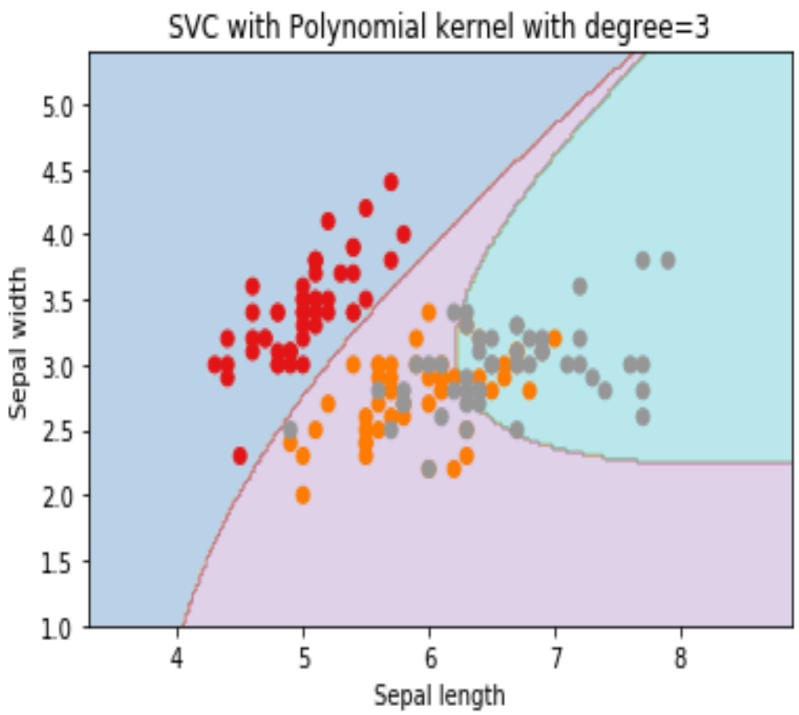


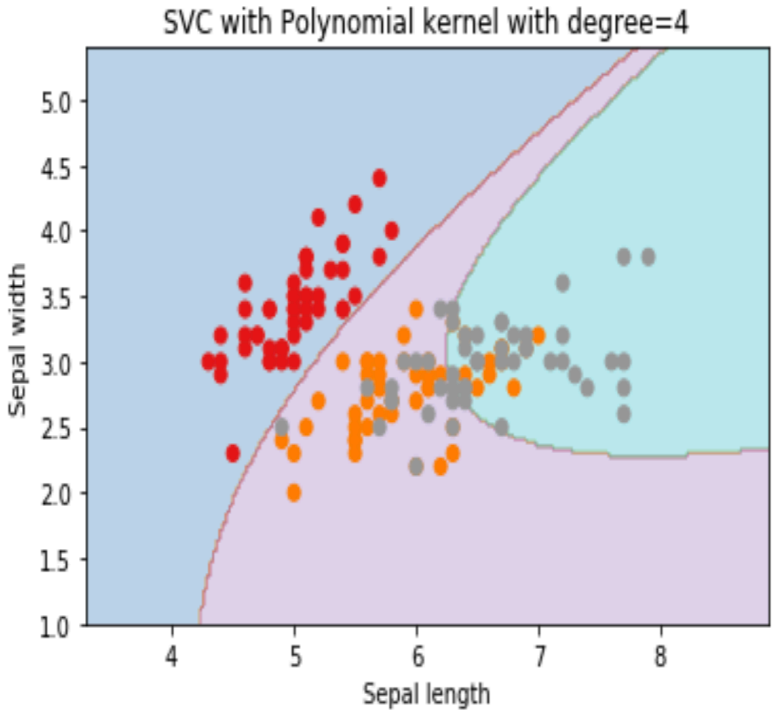
In Class Coding: (10 mins)

Calculate the effect of ‘degree’ using Polynomial Kernel in Python for degree=2,3,4

Polynomial Kerner with Degree=2,3,4







‘GridsearchCV’ to Find the Best Parameters

```
parameters = [{'kernel': ['rbf'],
               'gamma': [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5],
               'C': [1, 10, 100, 1000]},
              {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

print("# Tuning hyper-parameters")
print()

clf = GridSearchCV(svm.SVC(decision_function_shape='ovr'), parameters, cv=5
clf.fit(X_train, y_train)
```

```
Best parameters set found on development set:

{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}

Grid scores on training set:

0.652 (+/-0.112) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.652 (+/-0.112) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.652 (+/-0.112) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
```

Predict using test data

```
y_true, y_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, y_pred))
print()
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        11
          1       0.60      0.82      0.69        11
          2       0.83      0.62      0.71        16

avg / total       0.81      0.79      0.79        38
```

tp/(tp+fp)

tp/(tp+fn)

Weighted harmonic mean of precision and recall

Number of occurrences of each class in y-true

SVM for Regression

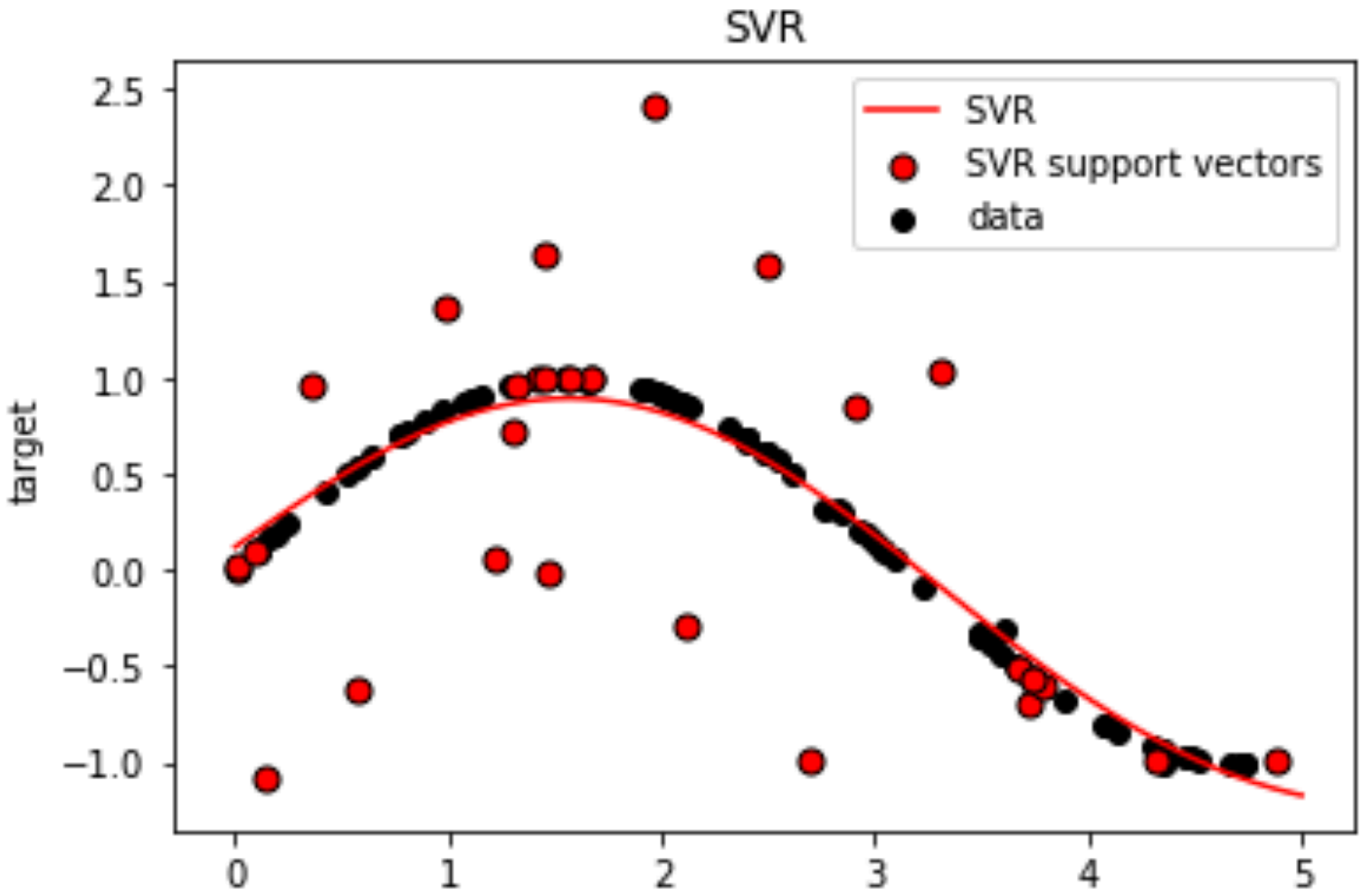
```
from sklearn.svm import SVR
```

```
# Generate sample data
X = 5 * rng.rand(10000, 1)
y = np.sin(X).ravel()

# Add noise to targets
y[::5] += 3 * (0.5 - rng.rand(X.shape[0] // 5))

X_plot = np.linspace(0, 5, 100000)[:, None]
```

```
train_size = 100
svr = GridSearchCV(SVR(kernel='rbf', gamma=0.1), cv=5,
                   param_grid={"C": [1e0, 1e1, 1e2, 1e3],
                               "gamma": np.logspace(-2, 2, 5)})
```

```
svr.fit(X[:train_size], y[:train_size])
```

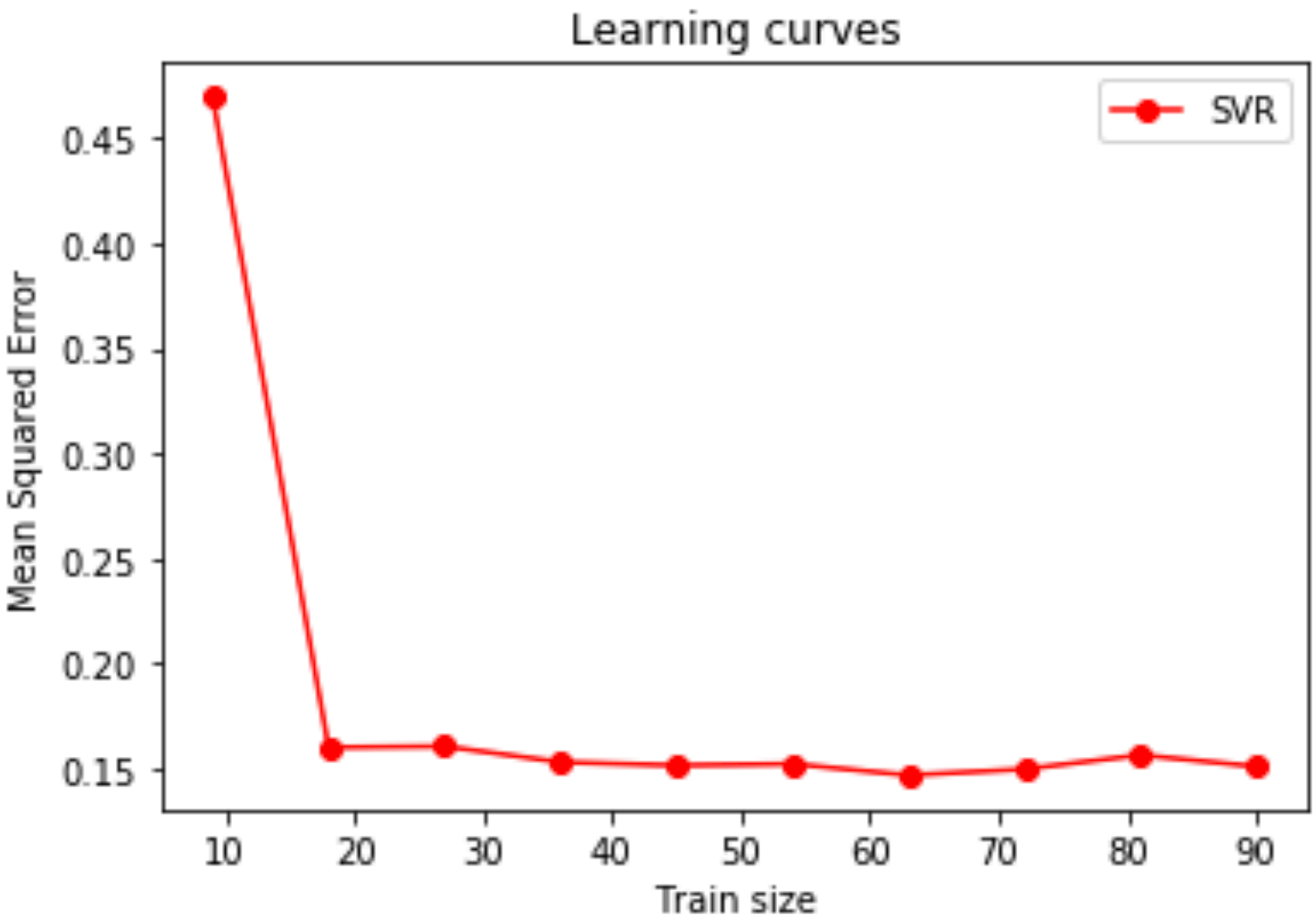
```
# Look at the results
sv_ind = svr.best_estimator_.support_  #gives us the best support vectors f
##plot the support vectors
plt.scatter(X[sv_ind], y[sv_ind], c='r', s=50, label='SVR support vectors',
            zorder=2, edgecolors=(0, 0, 0))
#plot the actual data
plt.scatter(X[:100], y[:100], c='k', label='data', zorder=1,
            edgecolors=(0, 0, 0))
plt.plot(X_plot, y_svr, c='r',label='SVR ')
plt.ylabel('target')
plt.title('SVR')
plt.legend()

# Visualize training and prediction time
plt.show()
```


SVR
SVR
SVR support vectors
data
target
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5

Learning Curve

```
from sklearn.model_selection import learning_curve
```

```
svr = SVR(kernel='rbf', C=1e1, gamma=0.1)
train_sizes, train_scores_svr, test_scores_svr = \
    learning_curve(svr, X[:100], y[:100], train_sizes=np.linspace(0.1, 1, 1
                   scoring="neg_mean_squared_error", cv=10)
print (test_scores_svr.shape)
plt.plot(train_sizes, -test_scores_svr.mean(1), 'o-', color="r",
         label="SVR")
plt.xlabel("Train size")
plt.ylabel("Mean Squared Error")
plt.title('Learning curves')
plt.legend(loc="best")

plt.show()
```


Learning curves
SVR
Mean Squared Error
0.45
0.40
0.35
0.30
0.25
0.20
0.15
10
20
30
40
50
60
70
80
90
Train size