Data Analysis and Machine-Learning

Chapter 6.2:

Support Vector Machine (2)

Parameter Tuning (SVC, SVR)



Now that we have understood the mathematical and algorithmic frameworks of the SVM, let us investigate the specific features of SVM by visualizing and comparing each parameter.

#Import essential libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.svm import SVC, SVR

from sklearn import datasets

#Generate Samples

x, y = datasets.make\_classification(

    n\_samples=50, n\_features=2, n\_informative=2, n\_classes=2, n\_redundant=0,

    n\_clusters\_per\_class=2, random\_state=123

)

np.c\_[y, x]

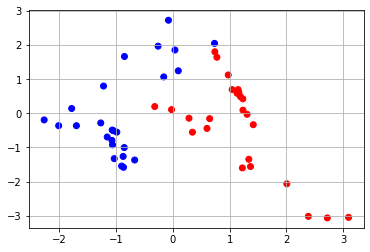
x1, x2 = x[:,0], x[:,1]

#Visualize

plt.scatter(x1, x2, c=y, cmap='bwr')

plt.grid()

plt.show()



The parameter C tunes the decision boundaries, i.e., how much misclassification to be allowed. If C is too small, too much misclassifications may result in underfits, while overvalued C may result in overfits to the training set.

The parameter Gamma tunes the curvature boundaries. If the parameter for gamma is too small, curvature range may become too large, resulting in underfits. If the gamma value is too large, on the other hand, overfits may occur as every single datum can be classified with each respective curvature range.

Linear or non-linear (e.g., poly, rbf) kernels can be selected depending on the characteristics of the classification datasets.

For better understanding, consider following illustrations (SVC, for classifications).

#Tuning SVM Parameters for SVC (C, Gamma, Kernel)

plt.figure(figsize=(20,15))

plt.subplot(4,2,1)

svm\_model = SVC(kernel='linear', C=0.05).fit(x, y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = linear, C=0.05')

plt.axis('off')

plt.subplot(4,2,2)

svm\_model = SVC(kernel='linear', C=10000).fit(x,y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = linear, C=10000')

plt.axis('off')

plt.subplot(4,2,3)

svm\_model = SVC(kernel = 'poly', C=0.5).fit(x, y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = poly, C=0.5')

plt.axis('off')

plt.subplot(4,2,4)

svm\_model = SVC(kernel = 'poly', C=10000).fit(x, y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = poly, C = 10000')

plt.axis('off')

plt.subplot(4,2,5)

svm\_model = SVC(kernel = 'rbf', C=1, gamma=0.05).fit(x, y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = rbf, gamma = 0.05')

plt.axis('off')

plt.subplot(4,2,6)

svm\_model = SVC(kernel = 'rbf', C=1, gamma=10).fit(x, y)

dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = rbf, gamma = 10')

plt.axis('off')

plt.subplot(4,2,7)

svm\_model = SVC(kernel = 'rbf', C=1, gamma=100).fit(x, y)

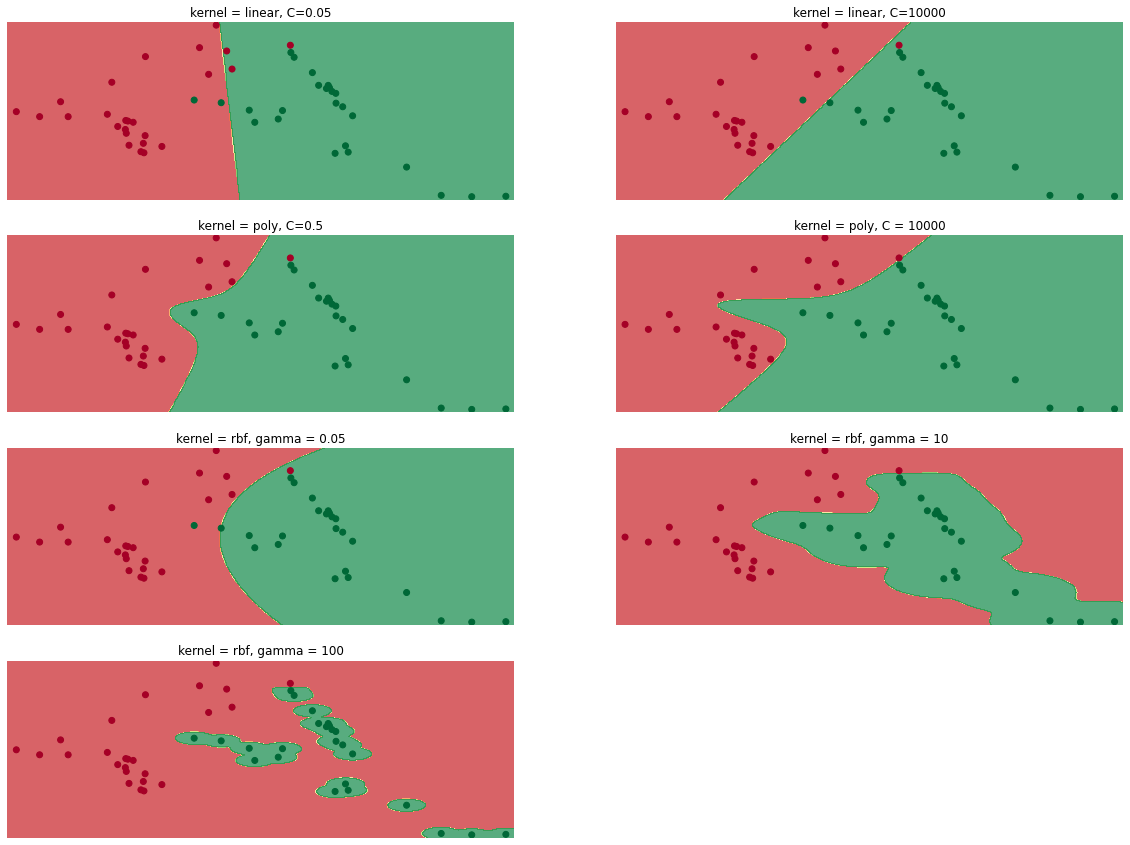
dt.dimensionchange(svm\_model, x1, x2, cmap='RdYlGn', alpha=0.7)

plt.scatter(x1, x2, c=y, cmap='RdYlGn')

plt.title('kernel = rbf, gamma = 100')

plt.axis('off')

plt.show()



For continuous dataset, support vector regression (SVR) algorithm can be used, as follows, tuning with the same parameters (C, Gamma).

#Generate Sample Dataset

np.random.seed(1234)

x\_train = np.linspace(0, 1, 80)

y\_train = np.sin(1.8 \* np.pi \* x\_train) + (np.random.randn(80)/10)

x\_test = np.linspace(0, 1, 20)

y\_test = np.sin(1.8 \* np.pi \* x\_test) + (np.random.randn(20)/10)

#Visualize data

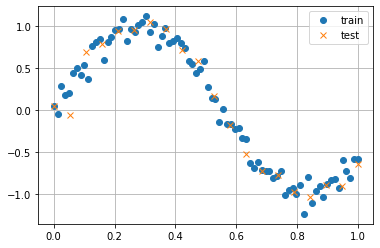
plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.legend()

plt.grid()

plt.show()



# Import SVR

from sklearn.linear\_model import LinearRegression

from sklearn.svm import SVR

#Tuning SVM parameters for SVR (C, Gamma)

plt.figure(figsize=(20,15))

plt.subplot(4,2,1)

x\_train = x\_train.reshape(-1,1)

linearModel = LinearRegression().fit(x\_train, y\_train)

yhat = linearModel.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, linearModel.predict(x\_train))

plt.legend()

plt.title('linear regression')

plt.grid()

plt.subplot(4,2,2)

svm\_model = SVR(C=1, gamma=1).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=1, gamma=1 (underfit)')

plt.grid()

plt.subplot(4,2,3)

svm\_model = SVR(C=1, gamma=10).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=1, gamma=10')

plt.grid()

plt.subplot(4,2,4)

svm\_model = SVR(C=10, gamma=1).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=10, gamma=1')

plt.grid()

plt.subplot(4,2,5)

svm\_model = SVR(C=10, gamma=10).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=10, gamma=10')

plt.grid()

plt.subplot(4,2,6)

svm\_model = SVR(C=10, gamma=10000).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=10, gamma=10000 (overfit)')

plt.grid()

plt.subplot(4,2,7)

svm\_model = SVR(C=10000, gamma=1).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=10000, gamma=1')

plt.grid()

plt.subplot(4,2,8)

svm\_model = SVR(C=10000, gamma=10).fit(x\_train, y\_train)

yhat = svm\_model.predict(x\_train)

plt.plot(x\_train, y\_train, 'o', label='train')

plt.plot(x\_test, y\_test, 'x', label='test')

plt.plot(x\_train, svm\_model.predict(x\_train))

plt.legend()

plt.title('SVR, C=10000, gamma=10')

plt.grid()

