Kernels

Kernel is a mathematical procedure for extending the feature space of a dataset to learn non-linear decision boundaries between different classes. The mathematical details of kernels are beyond the scope of this text. Suffice to say that a kernel can be seen as a mathematical function that captures similarity between data samples.

Linear Kernel

The support vector classifier is the same as a linear kernel. It is also known as a linear kernel because the feature space of the support vector classifier is linear.

Polynomial Kernel

The kernel can also be expressed as a polynomial. With this, a support vector classifier is trained on higher-dimensional polynomial features without manually adding an exponential number of polynomial features to the dataset. Adding a polynomial kernel to the support vector classifier enables the classifier to learn a non-linear decision boundary.

Radial Basis Function or the Radial Kernel

The radial basis function or radial kernel is another non-linear kernel that enables the support vector classifier to learn a non-linear decision boundary. The radial kernel is similar to adding multiple similarity features to the space. For the radial basis function, a hyper-parameter called gamma, γ, is used to control the flexibility of the non-linear decision boundary. The smaller the gamma value, the less complex (or flexible) the non-linear discriminant becomes, but a larger value for gamma leads to a more flexible and sophisticated decision boundary that tightly fits the non-linearity in the data, which can inadvertently lead to overfitting. This is illustrated in Figure 22-11. RBF is a popular kernel option used in practice.

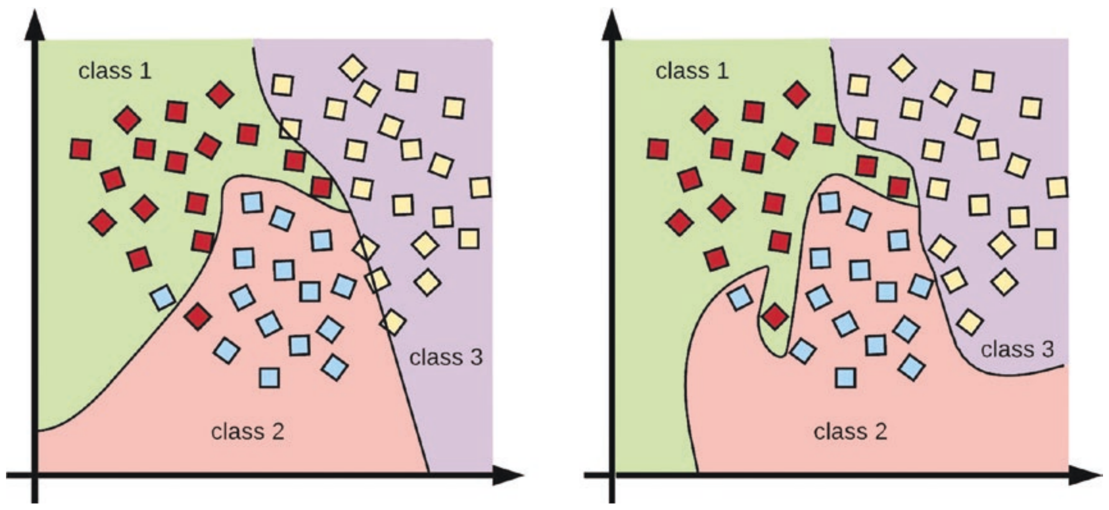

***Figure 22-11.** An illustration of adjusting the radial basis function γ parameter, together with the C parameter of the support vector classifier to fit a non-linear decision boundary. Left: RBF kernel with $C = 1$ and $\gamma = 10^{-3}$. Right: RBF kernel with $C = 1$ and $\gamma = 10^{-5}$.*

When using the radial kernel with the support vector classifier, the values of C and gamma are hyper-parameters that are tuned to find an appropriate level of model flexibility that generalizes to new examples when deployed.

In practice, a linear kernel or support vector classifier sometimes surprisingly performs well when used to map a function to non-linear data. This observation follows Occam's razor which suggests that it is advantageous to select the simplest hypothesis to solve a problem in the presence of more complex options.

Also, with regard to choosing the best set of C and gamma, γ, to avoid overfitting, a grid search is used to explore a range of values for the hyper-parameters and come up with the combination that performs best on test data. The grid search is used in conjunction with cross-validation approaches. However, the grid search procedure can be potentially computationally expensive.

Support vector machines perform well with high-dimensional data. However, they are preferred for small or medium-sized datasets. For humongous datasets, SVMs become computationally infeasible. Another limitation is that the performance of SVMs is known to plateau at some point, even when there exist large training samples. This is one of the motivations and advantages of deep neural networks.

Support Vector Machines with Scikit-learn

In Scikit-learn, **SVC** is the SVM package for classification, while **SVR** is the SVM package for regression. The attribute 'gamma' in both the SVC and SVR methods controls the flexibility of the decision boundary, and the default kernel is the radial basis function (rbf).

SVM for Classification

In this code example, we will build an SVM classification model to predict the three species of flowers from the Iris dataset.

```
# import packages
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from math import sqrt

# load dataset
data = datasets.load_iris()

# separate features and target
X = data.data
y = data.target

# split in train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True)

# create the model
svc_model = SVC(gamma='scale')

# fit the model on the training set
svc_model.fit(X_train, y_train)

# make predictions on the test set
predictions = svc_model.predict(X_test)
```

```
# evaluate the model performance using accuracy metric
print("Accuracy: %.2f" % accuracy_score(y_test, predictions))
```

```
'Output':
Accuracy: 0.95
```

SVM for Regression

In this code example, we will build an SVM regression model to predict house prices from the Boston house-prices dataset.

```
# import packages
from sklearn.svm import SVR
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# load dataset
data = datasets.load_boston()

# separate features and target
X = data.data
y = data.target

# split in train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True)

# create the model
svr_model = SVR(gamma='scale')

# fit the model on the training set
svr_model.fit(X_train, y_train)

# make predictions on the test set
predictions = svr_model.predict(X_test)

# evaluate the model performance using the root mean squared error metric
print("Mean squared error: %.2f" % sqrt(mean_squared_error(y_test,
predictions)))
```

```
'Output':
Root mean squared error: 7.58
```

In this chapter, we surveyed the support vector machine algorithm and its implementation with Scikit-learn. In the next chapter, we will discuss on ensemble methods that combine outputs of multiple classifiers or weak learners to build better prediction models.

CHAPTER 23

Ensemble Methods

Ensemble learning is a technique that combines the output of multiple classifiers also called weak learners to build a more robust prediction model. Ensemble methods work by combining a group of classifiers (or models) to get an enhanced prediction accuracy. The idea behind an “ensemble” is that the performance from the average of a group of classifiers will be better than each classifier on its own. So each classifier is called a “weak” learner.

Ensemble learners are usually high-performing algorithms for both classification and regression tasks and are mostly competition-winning algorithms. Examples of ensemble learning algorithms are Random Forest (RF) and Stochastic Gradient Boosting (SGB). We will motivate our discussion of ensemble methods by first discussing decision trees because ensemble classifiers such as RF and SGB are built by combining several decision tree classifiers.

Decision Trees

Decision trees, more popularly known as classification and regression trees (CART), can be visualized as a graph or flowchart of decisions. A branch connects the nodes in the graph, the last node of the graph is called a terminal node, and the topmost node is called the root. As seen in Figure 23-1, when constructing a decision tree, the root is at the top, while the branches connect nodes at lower layers until the terminal node.