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

# create the model. The parameter alpha represents the regularization
magnitude
linear_reg = Ridge(alpha=1.0)

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

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

# evaluate the model performance using the root mean square error metric
print("Root mean squared error (RMSE): %.2f" % sqrt(mean_squared_error(y_
test, predictions)))

'Output':
Root mean squared error (RMSE): 3.74
```

Take note of the following:

- The method Ridge(alpha=1.0) initializes a linear regression model with regularization, where the attribute 'alpha' controls the magnitude of the regularization parameter.

Logistic Regression with Regularization

This code block here is also similar to the example in Chapter 20 on logistic regression. The model will predict the three species of flowers from the Iris dataset. The addition to this code segment is the inclusion of a regularization term to the logistic model using the 'RidgeClassifier' package.

```
# import packages
from sklearn.linear_model import RidgeClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
# 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 logistic regression model
logistic_reg = RidgeClassifier()

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

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

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

'Output':
Accuracy: 0.76
```

In the preceding code block, logistic regression with regularization is implemented by the method 'RidgeClassifier()'. The reduced accuracy observed in this example when regularization is applied to logistic regression is because the algorithm is restricting the values of the model parameters to prevent high variance on a dataset that is fairly simplistic and already has high accuracy on test samples without regularization.

This chapter discusses the role of regularization in linear models like linear and logistic regression. Other forms of regularization exist for other model types such as early stopping for neural networks (to be discussed later in Chapter 34). Regularization is an important technique when designing machine learning models. The next chapter will discuss and implement another important machine learning algorithm known as support vector machines.

CHAPTER 22

Support Vector Machines

Support vector machine (SVM) is a machine learning algorithm for learning classification and regression models. To build intuition, we will consider the case of learning a classification model with SVM. Given a dataset with two target classes that are linearly separable, it turns out that there exists an infinite number of lines that can discriminate between the two classes (see Figure 22-1). The goal of the SVM is to find the best line that separates the two classes. In higher dimensions, this line is called a hyperplane.

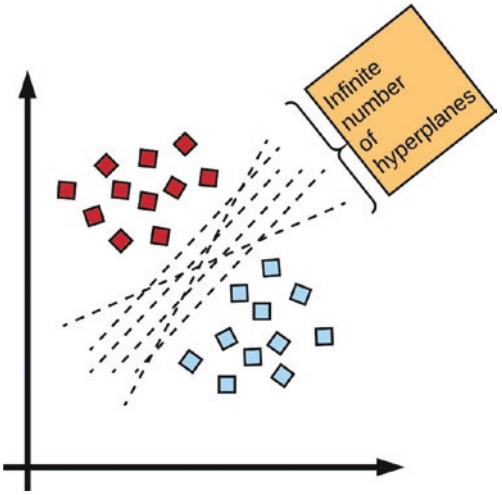

Figure 22-1. *Infinite set of discriminants*

What Is a Hyperplane?

A hyperplane is a line or more technically called a discriminant that separates two classes in n-dimensional space. When a hyperplane is drawn in 2-D space, it is called a line. In 3-D space, it is called a plane, and in dimensions greater than 3, the discriminant is called a hyperplane (see Figure 22-2). For any n-dimensional world, we have n-1 hyperplanes.

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_22