25 Case Study 3: Pattern Recognition

Objectives 25-1
Theory and Examples 25-2
- Description of Myocardial Infarction Recognition 25-2
- Data Collection and Preprocessing 25-3
- Selecting the Architecture 25-6
- Training the Network 25-7
- Validation 25-7
- Data Sets 25-10

Epilogue 25-11
Further Reading 25-12

Objectives

This chapter presents a case study in using neural networks for pattern recognition. In pattern recognition problems, you want a neural network to classify inputs into a set of target categories, e.g., recognize the vineyard that a particular bottle of wine came from, based on a chemical analysis, or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis.

In this chapter we will demonstrate the application of multilayer neural networks to the recognition of heart disease from a reading of the electrocardiogram. We will show each of the steps in the pattern recognition process: data collection, feature extraction, architecture selection, network training and network validation.

Theory and Examples

In pattern recognition (pattern classification) problems we are trying to categorize network inputs into their corresponding classes. Here are a few examples of pattern recognition problems:

- recognition of handwritten zip codes
- spoken word recognition
- disease recognition from a list of symptoms
- fingerprint recognition
- white blood cell classification

In the case study presented in this chapter, we will be looking for patterns in electrocardiogram signals that indicate the presence of a myocardial infarction (heart attack).

Description of Myocardial Infarction Recognition

An electrocardiogram (EKG) is a recording of the electrical activity of the heart over time. It generally consists of an array of different signals recorded at the same time. An EKG can consist of a single signal (also called a *lead*), although the standard EKG that is used for detailed interpretation consists of 12 leads. EKG's with as many as 15 leads are sometimes used. Each lead represents the electrical activity across two points on the body. The 12-lead EKG is determined from 10 electrodes that are placed on specific locations on the body. The calculation of the 12-lead potentials from the 10 electrodes is a somewhat complex calculation, and beyond the scope of this case study. The interested reader is referred to [Dubi00] for a more complete discussion of the EKG.

Through a careful analysis of the EKG, a physician can often determine the health of the heart. The shapes of the signals indicate the path of electrical flow in the heart as various muscles are contracted in a coordinated way to pump blood in and out. If a part of the heart muscle has been damaged because of a lack of blood flow through the coronary arteries (called a myocardial infarction (MI), or heart attack), then the path of electrical flow changes. A well-trained physician can discern from the changes in the EKG, if the heart has been damaged, and where the damage has occurred.

For this case study, we will train a neural network to recognize MI's, using information obtained from a 15 lead EKG.

Data Collection and Preprocessing

The EKG signals used for this case study were obtained from the Physio-Net database [MoMa01]. Data were extracted from the QT data set for healthy patients and patients with MI's. Each EKG consists of 15 leads. The leads are labeled I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6, VX, VY, VZ. Figure 25.1 shows a small portion of the lead I signal for one of the healthy patients.

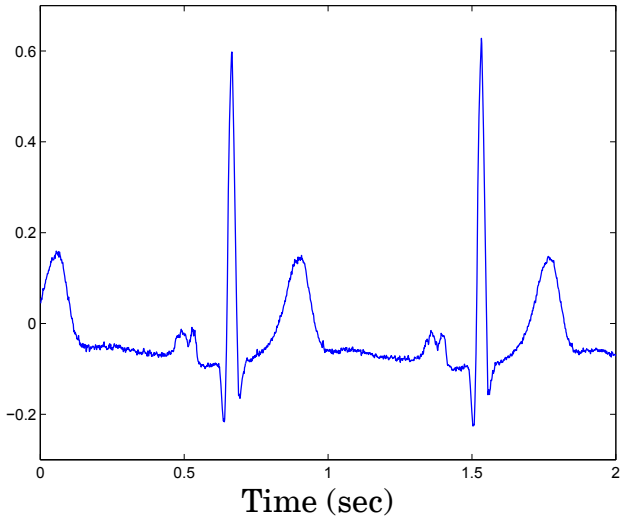

Figure 25.1 Example EKG Signal

Our data set has a total of 447 EKG records. Of these, 79 represent healthy patients, and the remaining 368 have an MI diagnosis. A diagnosis for each record was provided by a physician, but it is possible for the diagnosis to be in error. We will have more to say about this when we come to the validation of the network.

Each EKG consists of 15 leads measured at a rate of 1000 Hz for a period of several minutes. This is an enormous amount of data, and it would be impractical to use the entire EKG as an input to the neural network. As with many pattern recognition problems, we need to perform a *feature extraction* step before using the neural network to execute the pattern recognition step. Feature extraction involves mapping the high-dimensional input space into a space with fewer dimensions, in order to simplify and make more robust the pattern recognition step.

There are a number of general methods for dimensionality reduction. This includes linear methods, like the principal components method that we mentioned in Chapter 22, and nonlinear methods, like manifold learning [TeSi00]. For this case study, instead of using general methods to generate the low-dimensional feature space, we will extract features that are com-

monly used by physicians to detect abnormalities in the EKG. The first step is to consider a typical cycle of an EKG signal, as shown in Figure 25.2.

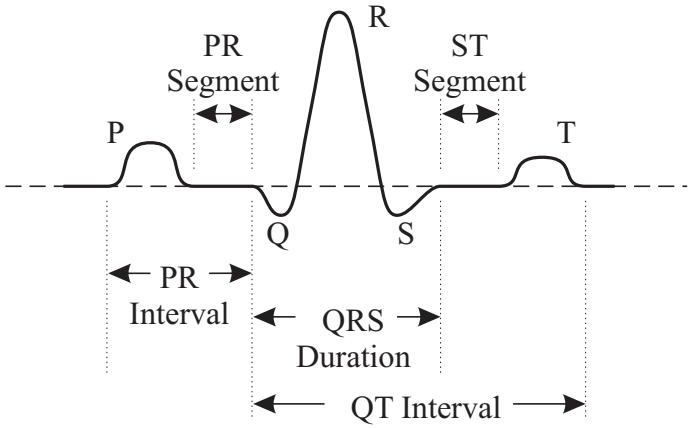

Figure 25.2 Prototype Cycle of an Electrocardiogram Signal

William Einthoven, in the early 1900's, was the first to carefully measure and analyze the EKG. He assigned the letters P, Q, R, S and T to the various deflections shown in the prototype cycle of Figure 25.2. He also described the electrocardiographic features of a number of cardiovascular disorders. He won the Nobel Prize in Medicine in 1924 for his discoveries. His features are still used to this day.

For this study, we have used some of the features that are in standard use by physicians, as well as other features that are related to the prototype cycle [Raff06]. The descriptions of the 47 features that we used are listed below. (If a description refers to the "amplitude" lead, this refers to the square root of the sum of squares of the VX, VY and VZ leads.)

Input Features to the Neural Network

1. age in years
2. gender, -1=female, 1=male
3. maximum heart rate in beats/min
4. minimum heart rate in beats/min
5. average time between heart beats in sec
6. rms deviation of the mean heart rate in beats/sec
7. full width at half maximum for the heart rate distribution
8. average qt interval for lead with max t wave
9. average qt interval for all leads
10. average corrected qt interval for lead with max t wave
11. average corrected qt interval for all leads
12. average qrs interval for all leads
13. average pr interval for lead with maximum p wave
14. rms deviation of pr intervals from average-max p lead

15. average pr interval for all leads
16. rms deviation for pr interval from average-all leads
17. percentage of negative p waves-max p lead
18. average percentage of negative p waves for all leads
19. maximum amplitude of any t wave
20. rms deviation of qt intervals
21. rms deviation of corrected qt intervals
22. average st segment length
23. rms deviation of st segment lengths
24. average heart rate in beats/min
25. rms deviation of heart rate distribution in beats/min
26. average rt angle averaged over all amplitude beats
27. number of missed r waves (beats)
28. % total qt intervals not analyzed or missing
29. % total pr intervals not analyzed or missing
30. % total st intervals not analyzed or missing
31. average number of maxima between t wave end and q
32. rms deviation of rt angle for all beats
33. ave qrs from amplitude lead
34. rms deviation of qrs from amplitude lead
35. ave st segment from amplitude lead
36. rms deviation of st segment from amplitude lead
37. ave qt interval from amplitude lead
38. rms deviation of qt interval from amplitude lead
39. ave bazetts corrected qt interval from amplitude lead
40. rms deviation of corrected qt interval from amplitude lead
41. ave r-r interval from amplitude lead
42. rms deviation of r-r interval from amplitude lead
43. average area under qrs complexes
44. average area under s-t wave end
45. average ratio of qrs area to s-t wave area
46. rms deviation of rt angle within each beat averaged over all beats in amplitude signal
47. st elevation at the start of the st interval for amplitude signal

To summarize, the data set contains 447 records. Each record has 47 input variables, and one target value. The target is 1 for a healthy diagnosis and -1 for an MI diagnosis.

One of the problems with the data set is that there are only 79 records for the healthy diagnosis, while there are 368 records for the MI diagnosis. If we train the network using the sum square error performance index, where all of the errors are weighted equally, the network will be biased to indicate

the MI diagnosis. The ideal solution to this problem would be to collect more data from healthy patients. Let's say that this is not possible in this case, and we need to do what we can with the data available. One possibility is to use a weighted sum square error as the performance index, where errors for healthy patients would be weighted higher than errors for MI patients, so that overall healthy and MI contributions would be equal if each error were equal. Another simple approach is to repeat the healthy records in the data set, so that the total number of healthy records is equal to the number of MI records. This requires extra computation, but that is not a problem in this case study. Since it is the simplest solution, we will use it here.

After the data has been collected, the next step is to divide the data into training, validation and test sets. In this case, we randomly set aside 15% of the data for validation and 15% for testing. For the validation and testing sets, we did not include multiple entries of the healthy records. This was done only for the training set.

The data were normalized using Eq. (22.1), so that the inputs were in the range [-1,1]. Since the tangent-sigmoid transfer function will be used in the output layer of the neural network, the targets were set to values of -0.76 and +0.76, instead of -1 and 1, to prevent training difficulties caused by saturation of the sigmoid functions, as discussed in Chapter 22.

Selecting the Architecture

Figure 25.3 shows the network architecture. We are using the tangent-sigmoid transfer function in both layers. This is the standard network for pattern recognition. There are cases in which two hidden layers are used, but we normally try first with one hidden layer. The number of neurons in the hidden layer, S^1, will depend on the complexity of the decision boundaries needed for the pattern recognition task. This is something that cannot generally be known before training. We will start with 10 neurons in the hidden layer, and then test the network performance after training.

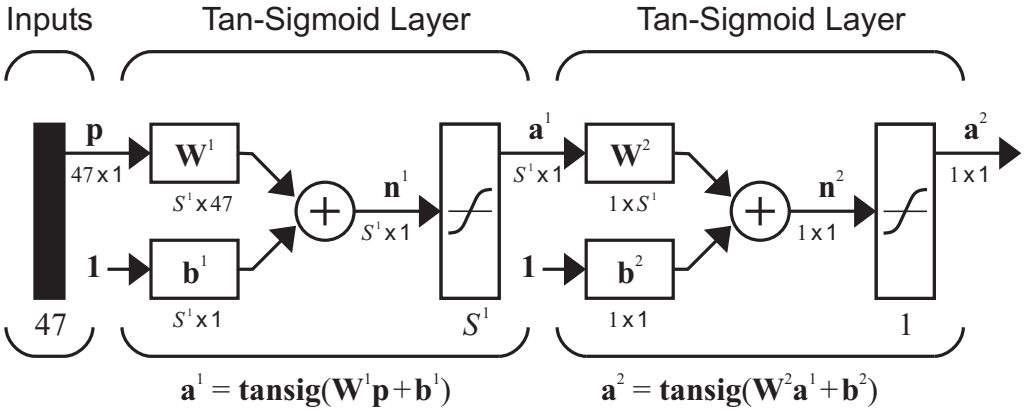

Figure 25.3 Network Architecture

Training the Network

We trained the network using the scaled conjugate gradient algorithm of [Mill93]. This algorithm is very efficient for pattern recognition problems. We used early stopping to prevent network overfitting.

Figure 25.4 illustrates the mean squared error versus iteration number. The blue line shows the validation error, and the black line shows the training error. We used a network with 10 neurons in the hidden layer ($S^1 = 10$). The minimum validation error occurred at iteration 16, as indicated by the circle in Figure 25.4, and the network parameters were saved at this point. Note that the validation error curve does not always fall at each iteration, and it may rise before falling to a lower value. We tested that the validation error was not reduced over 40 iterations before we finally stopped the training.

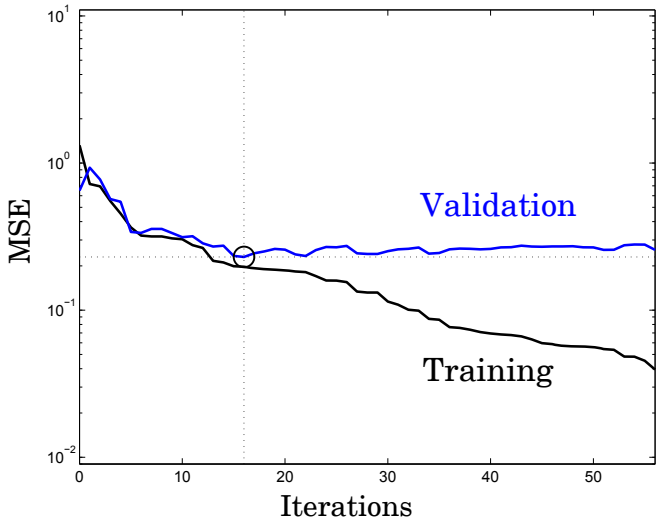

Figure 25.4 Mean Squared Error vs. Iteration Number ($S^1 = 10$)

Validation

As we discussed in previous chapters, an important tool for network validation in function approximation problems is a scatter plot of network outputs versus targets. For pattern recognition problems, the network outputs and targets are discrete variables, so a scatter plot is not particularly useful. Instead of the scatter plot, we use the confusion matrix, which was discussed in Chapter 22. Figure 25.5 shows the confusion matrix for our trained network on the test data. The upper left cell shows that 13 of the 14 healthy EKG's in the test set were classified correctly, while the 2,2 cell shows that 66 of the 71 MI EKG's were classified correctly. A total of 92.9% of the test data were classified correctly. The largest number of mistakes were for MI records that were classified as healthy (5), as shown in cell 1,2.

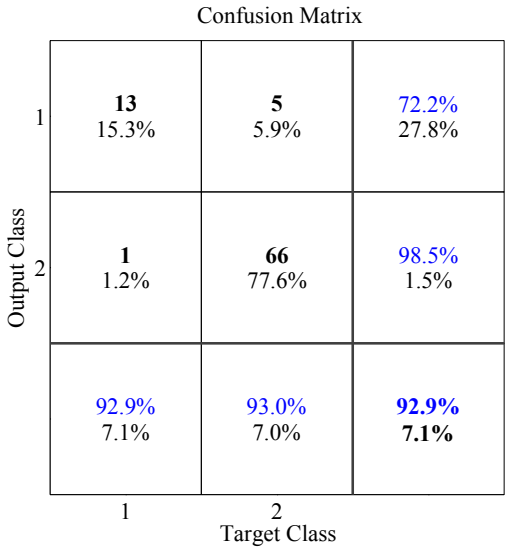

Figure 25.5 Confusion Matrix for Test Data (One Data Division)

Another useful validation tool for pattern recognition problems is the Receive Operating Characteristic (ROC) curve, described in Chapter 22. Figure 25.6 shows the ROC curve (blue line) for the test data. The ideal curve would follow the path from 0,0 to 0,1 and then to 1,1. The curve for this test set is close to the ideal path.

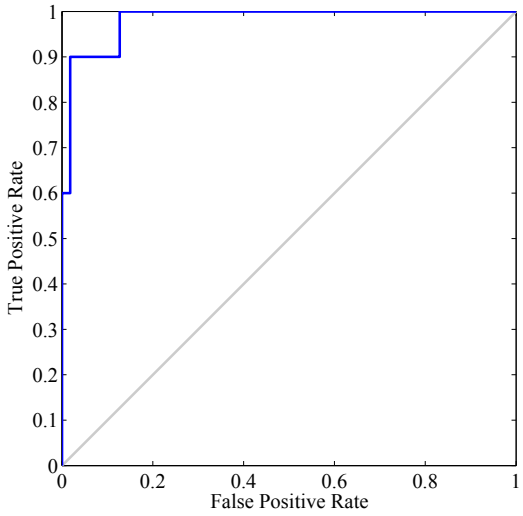

Figure 25.6 Receiver Operating Characteristic Curve (Test Set)

The results shown in Figure 25.5 and Figure 25.6 represent one division of the data into training/validation/testing sets. Because the data set is fairly small, especially in terms of healthy diagnoses, we might wonder how sensitive the results are to the data division. To investigate this sensitivity, we performed a Monte Carlo simulation. The data were divided 1,000 different

times. For each division of the data, a neural network was trained with different random initial weights. The results of the 1,000 trials were averaged together, and the results are shown in Figure 25.7.

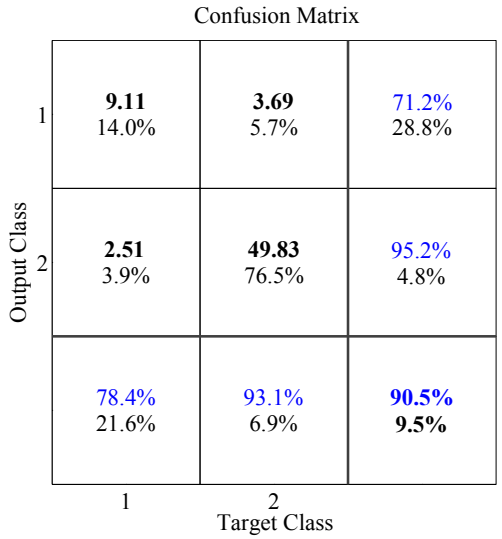

Figure 25.7 Average Test Confusion Matrix for 1,000 Monte Carlo Runs

Figure 25.7 represents the average results over the 1,000 different networks and data divisions. There were approximately 12 healthy patients (on average) in each test set. Of these, more than 9 were correctly diagnosed. There were approximately 54 sick patients in each test set, and approximately 50 were correctly diagnosed. The average testing error was approximately 9.5%. Note that none of the patients in the test set were used to train the neural network, so these numbers should be conservative estimates of how the network should perform on new patients.

The average test results for the Monte Carlo simulation are similar to our original test results. However, in addition to knowing the average results, it is also helpful to look at the distribution of errors. Figure 25.8 shows a histogram of the percentage errors. The average percent error is 9.5%, but there is a significant spread in the distribution of errors. The standard deviation of the percent error is 3.5.

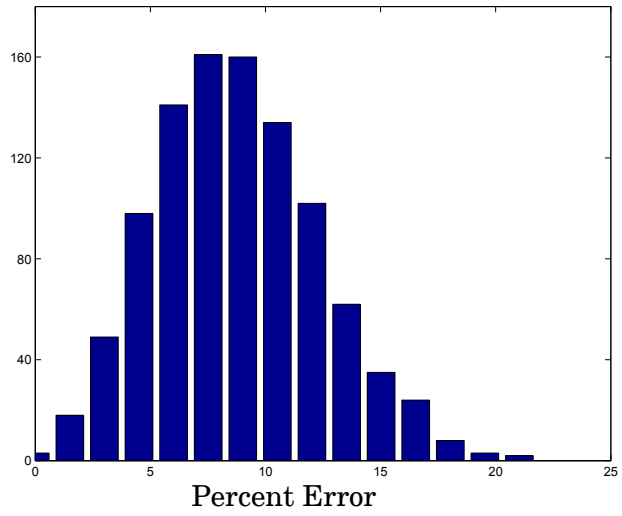

Figure 25.8 Percent Error Histogram (1000 Monte Carlo Trials)

This Monte Carlo process can be helpful in validating the data and the training process. For example, it is possible to identify which patients are misclassified in each Monte Carlo run. Those patients who are consistently misclassified (regardless of the division of the data) can be carefully investigated. These cases can be helpful in two areas. First, they can enable us to refine the data base. If, after reevaluation by clinicians, it is determined that a particular patient was mislabeled in the original data set, the data can be corrected. Second, if we find upon review that the patient was correctly labeled in the original data set, then we can use that patient to help improve the operation of the neural network classification. This may involve identifying new features which capture the key characteristics of the EKG, or it may involve obtaining more data with similar characteristics, in order to reinforce the training of the neural network.

The Monte Carlo process can be also used to help to improve the network performance. By combining the individual networks obtained from the Monte Carlo trials, we can often obtain a more accurate classification. The same input is applied to all of the networks, and the outputs can be combined through a “voting” procedure. We choose the class that is selected by the largest number of networks.

Data Sets

There are two data files associated with this case study:

- ekg_p.txt — contains the input vectors in the ekg data set
- ekg_t.txt — contains the targets in the ekg data set

They can be found with the demonstration software, which is described in Appendix C.

Epilogue

This chapter has demonstrated the use of multilayer neural networks for pattern recognition. In this case study, the pattern recognition network was used to classify EKG records into healthy and myocardial infarction diagnoses.

Most pattern recognition tasks involve a feature extraction step, in which the original data set is reduced in dimension. The features that were extracted from the EKG data consisted of characteristics of the prototype EKG cycle.

A Monte Carlo procedure was used as part of the network validation. The data were randomly divided a number of times into training/validation/test sets, and for each division a neural network was trained with random initial weights. The performances of all of the networks were analyzed to determine expected future performance. In addition, those records that were misclassified by most of the networks, regardless of the data division, were analyzed to assist in refining the data set and improving the pattern recognition.

In the next chapter, we apply neural networks to a clustering problem. We will use a self-organizing feature map network for that application.

Further Reading

[Dubi00] D. Dubin, *Rapid Interpretation of EKG's*, Sixth Edition, Tampa, FL: COVER, 2000.

This book describes the EKG in very clear terms, and leads you through the interpretation in a step-by-step way.

[MoMa01] G.B. Moody, R.G. Mark, and A.L. Goldberger, "PhysioNet: a Web-based resource for the study of physiologic signals," *IEEE Transactions on Engineering in Medicine and Biology*, vol. 20, no. 3, pp: 70-75, 2001.

This paper describes the PhysioNet data base that contains a large variety of recorded physiologic signals. The data base can be found at http://www.physionet.org/.

[Raff06] The features in the data set described in this chapter were designed and extracted by Dr. Lionel Raff, Regents Professor of Chemistry at Oklahoma State University.

[TeSi00] J. B. Tenenbaum, V. de Silva, J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, pp. 2319-2323, 2000.

There are several different approaches to manifold learning, in which data is mapped from a high-dimensional space to a lower-dimensional manifold. This paper introduces a method called Isomap.