Analysis of Autism Data using PCA and multiclass Naive Bayes model

Sreyam Sengupta, under the supervision of Dr. Koel Das

Details

Subject: Advanced Experimental Physics Laboratory
Course code: PH4201
Name: Sreyam Sengupta
Roll: 13MS121

Contents

1 Introduction

In neuroscience, particularly in EEG experiments, there is often a need to deal with multi-dimensional data containing different classes. For example, in the dataset I worked with, there is data from 1045 observations (say children, some of whom are autistic and some are not). The data from each child has 1280 entries. However, these 1280 entries are not always independent - some of the entries are correlated with the others. There are two main problems. The first problem is how to reduce the dimensionality of the data to make it more tractable. It is not easy to deal with 1045×1280 matrices. And in the realm of EEG experiments, this is a small matrix - typical observations can have around 10^5 entries, and there may be thousands of such observations.
The answer is PCA - principal components analysis. PCA begins by finding the covariance matrix of the data matrix, and then finds the eigenvalues and eigenvectors of the covariance matrix. The eigenvalues of the covariance matrix are the variance of the data along the corresponding eigenvector. Since the covariance matrix is real and symmetric, the eigenvectors can be made orthonormal. The eigenvalues are ranked in descending order. Generally, the first few eigenvalues account for a large fraction of the total variance. So we consider only those directions (i.e. the corresponding eigenvectors) and discard the rest.
Now that we have thrown away a large number of dimensions while still retaining much of the variance of the data, the next step is how to classify the data, assuming it is multiclass. Suppose we have data for crows and sparrows. We have taken data for many dimensions, including body mass, total length, body volume, wingspan, beak size, size of egg, leg length and so on. After PCA, we see the greatest variation is in body mass, body volume and wingspan. So we keep only these dimensions and discard the rest. Now the problem is, suppose we have an unknown data point with given body mass, body volume and wingspan. How do we tell whether it is a crow or a sparrow? This is where discriminant analysis comes in. We need a way to classify the data, so that when we get an unknown data point, we can use the classifier to predict which class (here, crow or sparrow) the data belongs to. To train the classifier, we need some training data, i.e. data whose class we already know. This data is fed to the classifier. Then, we can use our classifier to classify the unknown data point. In this case, we already know the test data - both the training data and the test data are taken from the same dataset, which is known. The point of this exercise is to see how good the PCA and Naive Bayes model are for a certain set of parameters.

2 The initial dataset

These were the matrices in the original matlab file:

1. *Data*: 1045×1280 double matrix. Contains 1045 observations, each having 1280 dimensions.

2. *TrainData*: 941×1280 double matrix. Contains 941 of the 1045 observations (about 90%) for training.

3. *ValidData*: 104×1280 double matrix. Contains the other 104 of the 1045 observations (about 10%) for testing.

4. *TrainIndex*: 941×1 double matrix. Contains the index numbers of the training data. For example, if the 10th element of *TrainIndex* is "579" it means the 10th row of *TrainData* is identical to the 579th row of *Data*.

5. *ValidIndex*: 104×1 double matrix. Similar to *TrainIndex*, but for *ValidData* instead of *TrainData*.

6. *TrainLabels*: 941×1 double matrix. Contains the classification (which is binary, either 0 or 1) of the training data. For example, if the 10th element of *TrainLabels* is "0" it means the 10th row of *TrainData* (which is the 579th row of *Data*) belongs to class "0". "0" and "1" stand for autistic or non-autistic, I guess "0" means non-autistic (Null hypothesis) and "1" means autistic, but I am not sure. It does not matter here.

7. *ValidLabels*: 104×1 double matrix. Similar to *TrainLabels*, but for *ValidData* instead of *TrainData*.

3 Performing principal components analysis (PCA) on the training data

The following are matrices I created. These resulted from using the *princomp()* function with *TrainData* as input:

1. *coeff*: 1280×1280 double matrix. The principal component coefficients of the matrix *TrainData*. *TrainData* contains 941 entries (rows), each having 1280 components (columns). We do PCA on n d-dimensional vectors. Here $n = 941$, $d = 1280$ (here the dimensionality of a vector is defined as how many components it has, i.e. what dimensional space it lives in, not how many indexes it requires, which is 1). So, doing PCA on those n d-dimensional vectors gives us d d-dimensional vectors. If you think of the original n datapoints as forming a d-dimensional ellipsoid, the principal components can be thought of as the principal axes of that ellipsoid. The 1st column of *coeff* contains the components of the 1st principal axis (as in the longest one, the one having the largest eigenvalue), the 2nd column is the 2nd principal axis and so on.

2. *score*: 941×1280 double matrix. The scores of the principal components. Unimportant for us.

3. *latent*: 1280×1 double matrix. It contains the principal component variances, i.e. the eigenvalues of the covariance matrix of *TrainData*, arranged in descending order in a single column. For example if the 8th element of *latent* is "4.3694e+03" it means the 8th column of *coeff* is an eigenvector of the covariance matrix of *TrainData* with eigenvalue 4.3694×10^3.

4. *tsquared*: 941×1 double matrix. Hotelling's t-squared characteristic. Useful for finding extremal points in the data. Unimportant for us.

5. *explained*: 1280×1 double matrix. Percentage of the total variance due to each principal component. For example, if the 8th element of *explained* is "1.5765" it means the 8th principal axis, i.e. the 8th column of *coeff* accounts for 1.5765% of the total variance in the data. Can be thought of as *latent*, scaled so that the sum of elements is 100.

6. *mu*: 1×1280 double matrix. Estimated means of each variable, i.e. estimated mean of each column of *TrainData*.

Next, we perform some analysis.

4 Analysing the results of PCA on training data

We plotted the contents of *explained* using the *pareto()* function, which shows both a bar graph and a cumulative distribution. The graph is attached below. In it we clearly see that the first 9 principal axes together account for over 70% of the variance, so we will discard the rest. We reduce the dimensionality of the data from 1280 to 9. We do this by defining a new matrix:

coeff9: 1280×9 double matrix. This matrix contains the first 9 columns of *coeff*, which together account for over 70% variance.

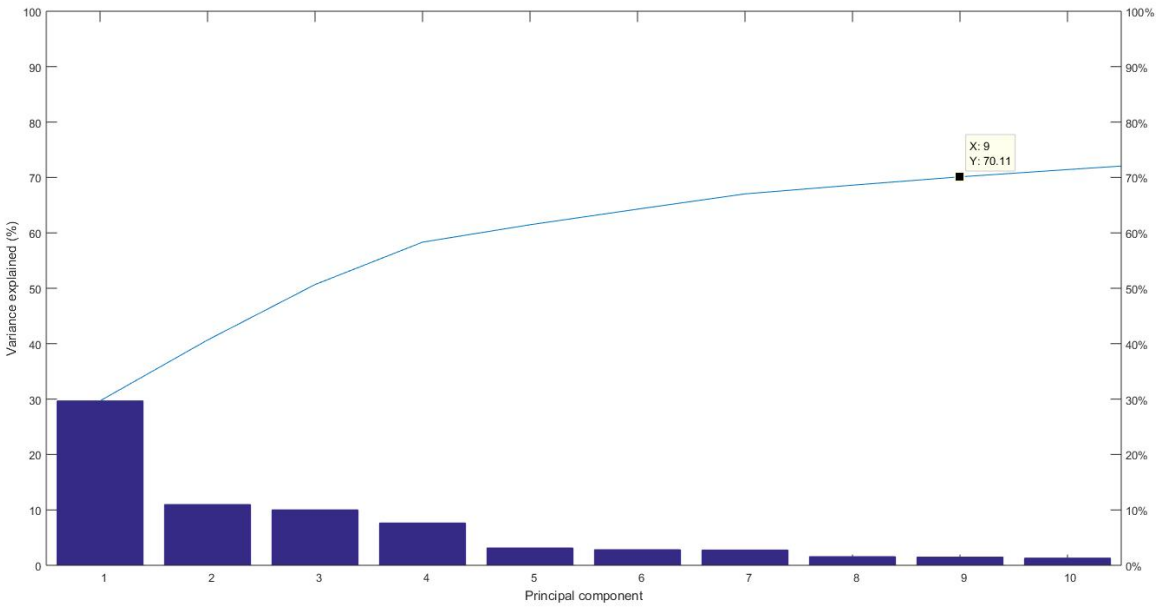

Figure 1: Variance explained (%) vs Principal components

5 Reducing dimensionality of training data and test data

Now that we have the first 9 principal axes, we simply multiply *TrainData* and *ValidData* with *coeff9* to reduce their dimensions. Two new matrices are defined:

1. *reduced_TrainData*: 941×9 double matrix. It is simply *TrainData*×*coeff9*.
2. *reduced_ValidData*: 104×9 double matrix. It is simply *ValidData*×*coeff9*.

6 Classifying the training data using a multiclass Naive Bayes model

We classified the training data twice, first using the entire *TrainData* to classify and predicting with the *ValidData*. The next time we used the *reduced_TrainData* to classify and predicted on the *reduced_ValidData*. Two new matrices are defined:

1. *label*: 104×1 double matrix. The predicted labels of *ValidData* when classified using *TrainData*.
2. *reduced_label*: 104×1 double matrix. The predicted labels of *reduced_ValidData* when classified using *reduced_TrainData*.

7 Checking the accuracy of prediction

To do this, we define two variables: *full*, to measure the accuracy of Naive Bayes classification using the original *TrainData* and predicting on the *ValidData*, and *reduced*, which measures the accuracy of the Naive Bayes classification using *reduced_TrainData* and predicting on *reduced_ValidData*.
Neither are very good. The former yields a success rate of 69.23% while the latter yields 57.69%.

8 MATLAB code

```
>> load('Aut_Data_sreyam_1stfold.mat')
>> [coeff,score,latent,tsquared,explained,mu] = princomp(TrainData);
>> figure
>> pareto(explained)
>> xlabel('Principal component')
>> ylabel('Variance explained (%)');
>> coeff9 = coeff(:,1:9);
>> reduced_TrainData = TrainData*coeff9;
>> reduced_ValidData = ValidData*coeff9;
```

```
>> Mdl = fitcnb(TrainData, TrainLabels);
>> reduced_Mdl = fitcnb(reduced_TrainData, TrainLabels);
>> label = predict(Mdl,ValidData);
>> reduced_label = predict(reduced_Mdl,reduced_ValidData);
>> full=0.0;
>> reduced=0.0;
>> goodness_of_Mdl = zeros(104,1);
>> goodness_of_reduced_Mdl = zeros(104,1);
>> for i = 1:104
if label(i,1) == ValidLabels(i,1)
goodness_of_Mdl(i,1) = 1;
else
goodness_of_Mdl(i,1) = 0;
end
end
>> for i = 1:104
if reduced_label(i,1) == ValidLabels(i,1)
goodness_of_reduced_Mdl(i,1) = 1;
else
goodness_of_reduced_Mdl(i,1) = 0;
end
end
>> for i = 1:104
full = full + goodness_of_Mdl(i,1);
reduced = reduced+goodness_of_reduced_Mdl(i,1);
end
>> full = full/104;
>> reduced = reduced/104;
>> full

full =

    0.6923

>> reduced

reduced =

    0.5769
```

9 Conclusion

The classification using Naive Bayes model seems pretty bad - perhaps another algorithm does it better.
What I did was only a tiny fraction of what I could have done. A satisfactory start to the project could have been to perform the 10-fold classification - I could only do 1 fold. I have no excuses for this.

10 Acknowledgements

I am grateful to Dr. Koel Das for providing me the opportunity to work in such a field at all, even though I had no background in pattern classification.
I thank Arpita Saha Chowdhury (project JRF, Neuro lab), Tiasha Saha Roy (PhD student, Neuro lab) and Rohit Bhagwat (MS student, Neuro lab) for their help and guidance. Without them this project report would never have been written.