10-601 Project Part 3

Cole Gulino (cgulino@andrew.cmu.edu)
Rushat Gupta Chadha (rguptach@andrew.cmu.edu

Anonymous Author(s)
Affiliation
Address
email

Abstract

Brain imaging using fMRI data can provide many insights into the human brain. By using machine learning techniques, complex processes in the brain can be predicted or modeled. This paper looks at several such techniques used by the 10-601 team of Cole Gulino (cgulino) and Rushat Gupta Chada (rguptach). The first part of the paper focuses on a classification task using feature selection and Principal Component Analysis (PCA) along with a Support Vector Machine (SVM) classifier to predict actions carried out by test subjects. The next part of the paper focuses on using feature selection and PCA along with ridge regression to reconstruct missing features in the form of the data from Part 1 of the paper. The final section covers a method using the reconstructed data from Part 2 and semi-supervised learning to garner higher accuracies for Part 1. This paper will show that by using dimensionality reduction, one can achieve high accuracy on highly dimensional data sets even with simplistic classification and regression techniques. In using these fundamental machine learning techniques, we have been able to model and classify the information of the brain for simple tasks.

1 Project Part 1: Supervised Learning

1.1 Methods Used

Part 1 of the project's goal was to use the BOLD response of brain voxel data in order to predict three different responses from a test subject: (0) No Event, (1) Early Stop, and (3) Correct Go.

The information was provided in a high dimensional space (5903 features). Because the feature space is so high, we needed to use dimensionality reduction in order to ensure that only features relevant for classification were kept. Features that are irrelevant can skew your results by steering your classification away from the relevant features.

The first dimensionality reduction technique that we used was PCA. Shown in the figure below is the explained variance vs the number of components kept in PCA is shown in Figure 1.

The methods that we used for our highest classification accuracy was PCA and feature selection to reduce the dimensionality of the problem. It is clear from Figure 1 that most of the variance is captured in a small number of features.

We first began to try to just capture the values that are shown to have the most variance in Figure 1, but our cross-validation scores were not very successful. In order to get better values, we decided to use grid search found at `http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html` function in scikit-learn in order to test the

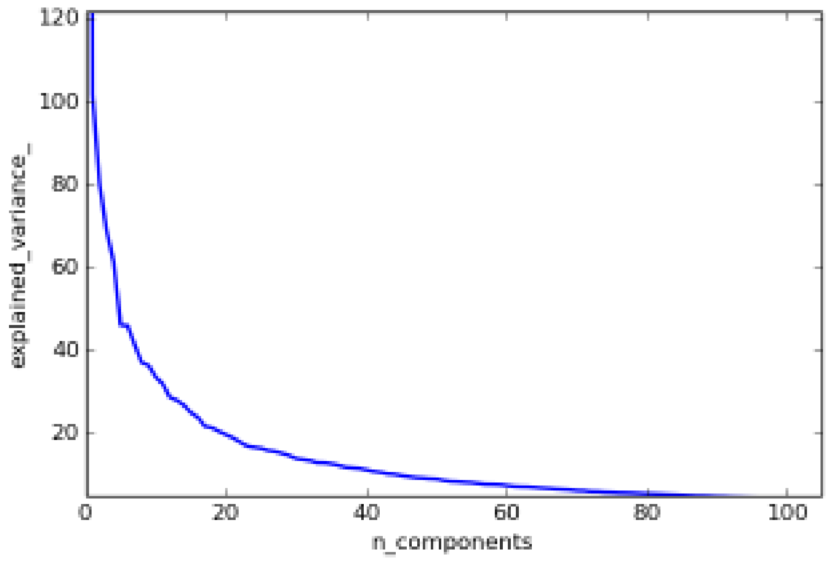

Figure 1: Explained Variance vs Number of Components

number of components we should use to get the best cross-validation score. Figure 2 shows the cross-validation score as a result of the number of principal components.

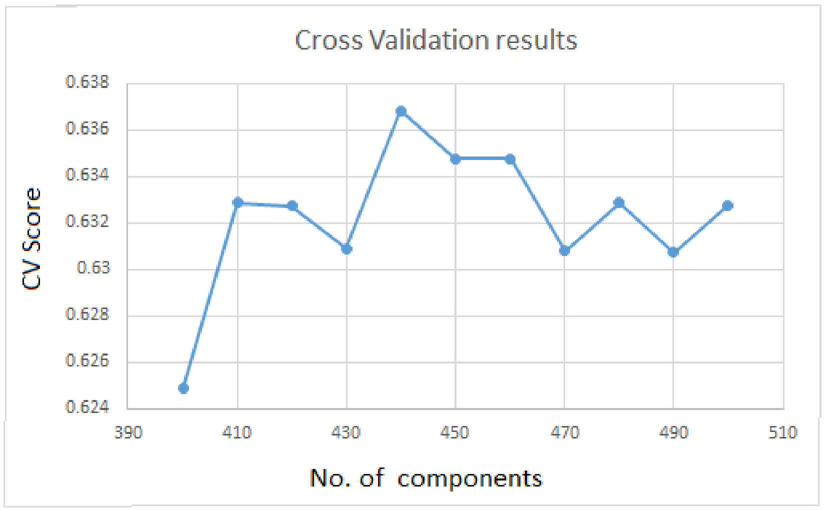

Figure 2: Average CV Score vs. Number of Components

Once we had found the peaks of the cross-validation score vs the number of components for Principal Component Analysis, we tuned it by hand in order to get to our optimal value of 453 components.

After we optimized for PCA, we used feature selection in order to get more useful features from our data. Table 1 shows the cross validation score with feature selection and without.

Once we reduced our features to the optimal space we could find, we trained our SVM classifier. We did not need to play around with the values of the SVM much more than what the recommendations were. We chose the SVM classifier, because it was recommended by the professors, the guest lecturer, and a paper we read on classifying fMRI data [1]. SVM with an RBF Gaussian kernel gave us the best results compared to other classifiers that we tried, including: Gaussian Naive Bayes, Random Forests, and perception.

Table 1: CV Score: With and Without Feature Selection

SelectKBest: k value	**Average CV Score**
0	0.624908333
180	0.634757595

This was the simple method that we used in order to predict the data. By optimizing for feature selection and feature reduction, we were able to get high prediction accuracy in classifying the voxel data.

1.2 Methods Considered, but Not Used

This simple method that we used gave us the best score that we were able to obtain. There were other methods that we tried briefly that gave us a comparable score, but did not give us a better score.

The first thing that we tried was to use a voting method on multiple classifiers. This gave us comparable results to before, but we worried about over-fitting with this method.

The next thing that we tried was to use the (x, y, z) data from the voxels in order do some spatial smoothing. This method was very promising, but the code ran for too long, and we did not have enough time to test it.

The final extra method that we tried was to train a different classifier for every subject. This method was also very promising but was dropped due to time constraints as well. We were getting comparable results by over-fitting each classifier to the individual subject. We trained each classifier on the whole training set and added in multiple versions of the individual subject that the classifier was trained on. We did not have enough time to optimize each classifier, however. We believe that this would have given us better results than our final solution.

2 Project Part 2: Unsupervised Learning

2.1 Methods Used

The title of Project Part 2 is unsupervised learning, but we converted the problem into a supervised learning problem using regression instead of classification like we used for Project Part 1.

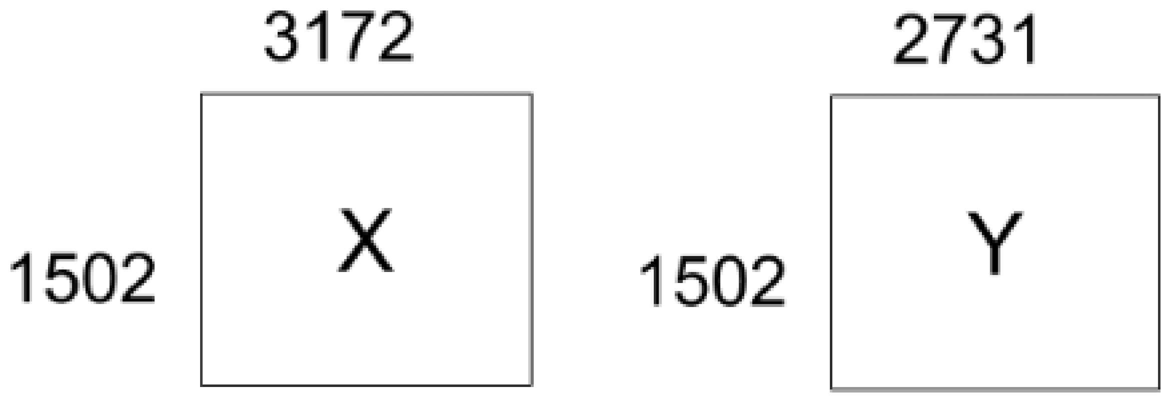

Figure 3: X and Y for Regression

We needed to change our data into a form that could be used by regression. The first thing that we did was to get our X and Y from the training data. We took the rows that corresponded to the indices of the provided data and used that as our X for training the regressor. We then took the rows the corresponded to the indices of the missing data and used that as our Y for training the regressor. An example of this is shown in Figure 3. Figure 3 shows the shape of the X and Y data used to train the regressor.

In order to prevent over-fitting in regression, it is often useful to use a regularization technique like ridge or lasso regression. In practice, we found that we could get similar results for ridge and lasso regression during our trial runs. The main impetus for us choosing ridge regression is the speed of computation compared to lasso regression.

Ridge regression runs in a matter of minutes, while similar performance for lasso regression takes hours with the implementations presented in scikit-learn.

Once we had chosen our regressor, it was then a matter of optimizing the parameters. We again used grid search in order to optimize our parameters. The first thing that we did was to see if we could get optimal performance from using feature reduction like we did in Part 1. Figure 4 shows a graph of cross-validation score versus number of components used in PCA.

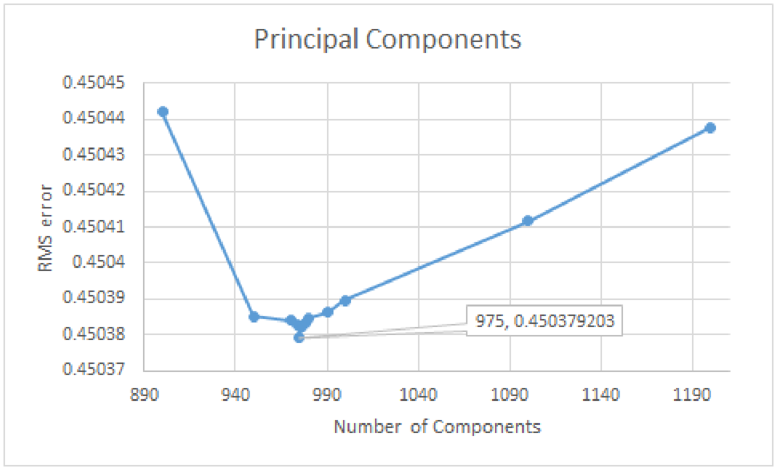

Figure 4: Average CV Score vs. Principal Components

Figure 4 shows that we had a minimum of root mean squared error (RMSE) value whenever the number of components kept were 975. Once we had optimized over PCA, we optimized for the alpha value for Ridge Regression. This is the value that determines how much high weight values that cause over-fitting are penalized. We found that very high alpha values for ridge regression were optimal. Figure 5 shows a graph of the cross-validation score versus alpha values.

By optimizing ridge regression for alpha and getting the optimal number of principal components, we were able to get a RMSE score that is effective in reconstructing the missing features of the voxel data.

3 Project Part 3

The task for Part 3 was to use the processes from Parts 1 and 2 in order to do something interesting with the data. In Part 3, we decided to use regression in order to reconstruct the missing features from the test data in Part 2. We then added that data to the training data from Part 1. We used this data to improve the classification of the test set from Part 1.

The motivation for our take on Project Part 3 is based on the curse of dimensionality. Data that has a high number of features needs a high number of training examples in order to classify properly. Methods like dimension reduction have been shown above to be very effective at improving our classifiers, but we believe that generating new training data would have an even greater effect.

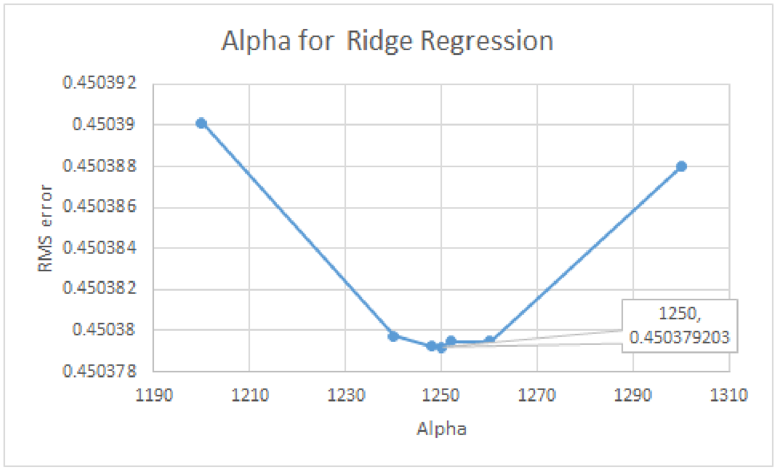

Figure 5: Average CV Score vs. Alpha

Using the ideas from Part 2, we can reconstruct more training examples from the provided data. Once we have our X, we can use a semi-supervised method in order to get a label for this data.

3.1 Methods Used

The first thing that we did was gather the data. We gathered the missing features using our best method for Part 2. Once we had the fully reconstructed features, we then wanted to get a classification for the reconstructed data in order to add it to our training data for the classification task of Part 1.

We tried many parameters and ended up with a value of $k = 7$. We used PCA in order to reduce the dimensions of the training set. We then used k-NN semi-supervised learning in order to get the classifications for the reconstructed data from Part 2. The k-NN algorithm will take the data that is most similar to the data that is already classified and genreate a label for the data that is without one. This unsupervised method allows us to get our labels for the classification problem.

We then added the reconstructed data and its labels into the original training data.

Now that we have more training data, we can then predict the classes of the original test set from Part 1 on the SVM classifier that we used for Part 1. The hope is that this will provide us with a better classification than Part 1.

3.1.1 Results

Once we had trained the classifier, we quantified the results by running a 10-Fold Cross-Validation on both the classifier from Part 1 and the classifier from Part 2. The results are shown in figure 6.

As Figure 6 shows, for 10-Fold Cross-Validation, the classifier from Part 3 performed better on every fold than the the classifier from Part 1.

This result is not very surprising. The more training data you have; the more accuracy your classifier will present.

It seems that generating training data could be a general way to improve the performance of many classifiers. This effect is especially seen in our data, because the number of features is much larger than the number of training instances. Data that is presented like this has a difficult time getting a good classifier.

References

[1] Singh, Vishwajeet & Miyapuram, K.P. & Bapi, Raju S. (2007) Detection of Cognitive States from fMRI data using Machine Learning Techniques, pp. 587-592.

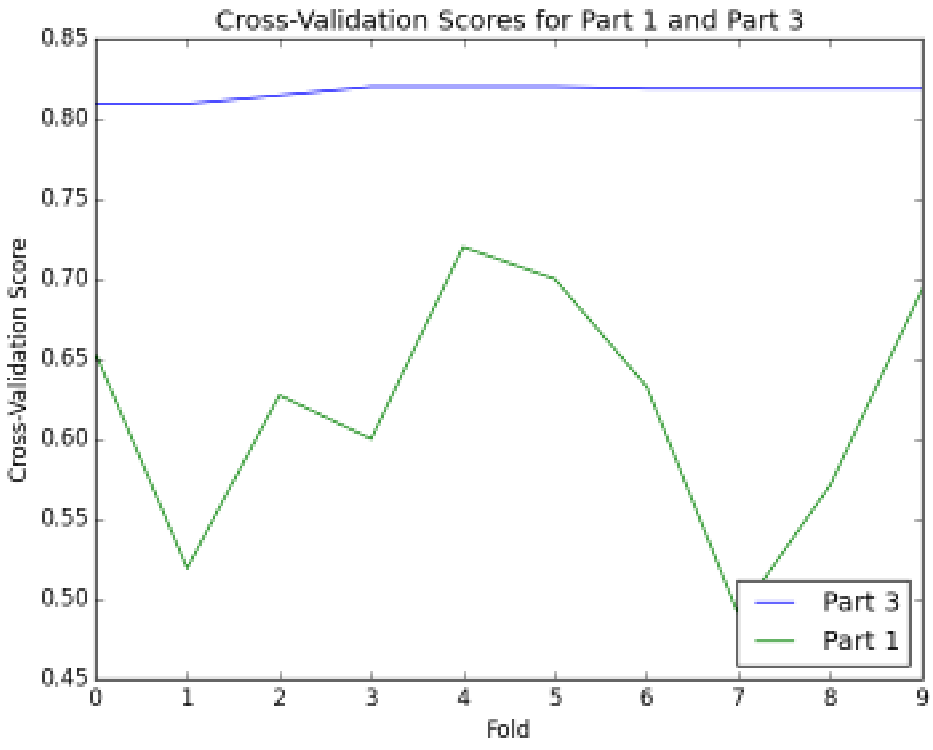

Figure 6: CV Score per Fold for Part 1 and Part 3 Classifiers