10601A Machine Learning Final Project

Wei Fu Wang , Yu Fang Chang
weifuw@andrew.cmu.edu, yufangc@andrew.cmu.edu

1 Label Prediction

Our task for project part 1 is to train a classifer to predict label from the given data (X (brain scan), subject ID, event time and voxel locations).

1.1 Library and Programming Languages

Matlab and LibSVM

1.2 Approach

- **Centering data**
  Since the range of values of raw data varies widely, the learning algorithm might not work properly without normalization. If the scales for different features are wildly different, objective functions may be governed by few dominant features. Therefore, we consider normalizing the range of all features to ensure feature values implicitly weights all features equally in their representation.
  We have applied the simplest approach, min-max scaling to normalize our data. For each value, they are standardized as:

$$X_{centered} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  By this approach, the data are set to fixed range: 0 to 1.
  However, min-max scaling does not improve our result much. One possible reason may be the original scaling does represent some extent of feature contribution to the objective function and thus we should not normalized them. Since we trained our model mainly with Support Vector Machine, we normalize a vector by converting it to a unit vector. This trains the SVM on the relative values of the features, not the magnitudes. The magnitudes of the features is c̈ancelled outb̈y the normalization and we are left with relative values between 0 and 1.
- **LibSVM with grid search (cross-validation)**
  We followed the instruction by training the model with the basic Gaussian kernel SVM classifier and obtained the probabilities for each of the classes in the data. It results in approximately 0.57 accuracy in the hold out data set. Furthermore, we applied grid search for tuning the parameters (C and gamma for SVM) of our classifiers. We first split the given training data into half, one for training and another for validating our model. Since we have found that the data are quite unbalanced, we have checked to ensure the data distribution was uniform across all the labels. We used 10-fold cross validation. After finding the best parameters with grid search, we train our model with the whole given training data and the best C:50 and gamma:2^{-10}. This results in approximately 0.61 accuracy in the hold out data set. Additionally, we have also tried applying PCA to reduce dimension first and then kernel SVM, but no obvious improvement.

- **Add subject and event as extra feature**
  Since the event and subject Id are given, we have also tried to apply them into our learning model. We found that there are 28 subject Ids in the data and different subject ids correspond to experiments with different conditions and thus it is intuitive that the features and labels may have high relevance with the same subject. Therefore, we add a column of feature representing the subject and normalized them with min-max scaling. With this approach, we treat the information of subject id as a prior, and it turns out to be a better result, 0.64 accuracy in the hold out data set. However, we didn't find out a proper way to add event as another attribute to the objective function.
- **LibSVM with training several model then vote**
  Since our goal was getting the probability of data being classified to each class, another intuitive method is to train multiple models and predict several times to get the probability. We had simply tried to split the training data into 10 parts, trained 10 different models by libSVM with parameters that found by processing grid search. Then predicted each data 10 times, and voted, which is to count the times they were classified to certain class, ended up getting the probability. At first we didnt spend too much time on this voting method because we had gotten greater results by only using SVM predictor(in comparison to 0.646 on hold out set, here was only 0.631). However, after getting unexpected test accuracy and reading other teams sharing about their high accuracy solutions, we found out that many of them used this voting method. We thought one possible reason was that voting effectively reduced the weights of each predictor, thus reduced the risk of bad prediction resulted from overfitting models or any bad selected models. So we believed that if we could have trained multiple models with couple of different learning methods, and voted to get probability, it would have resulted in much better accuracy on the test data set.

Finally, we convert the prediction probabilities into a matrix where the highest probability was labeled 1 and the other two were labeled 0.

1.3 Result

Hold-out	Test
0.646	0.591

Table 1: Accuracy

From above result, we can found that the test accuracy is much lower than hold-out accuracy. It may be caused by the class imbalance in training data (class-3 have also half data (225 points) and class-2 only have 123 points). Our classifier might intrinsically have higher accuracy on class-3 data, and lower accuracy on other class data.

2 Predict missing data

In the second part of this project, our task is to deal with unsupervised learning problem, which is, given partial voxel values as features, to predict the remaining voxel values. Our idea was that there might exists some correlation between features, so we should be able to predict these missing data with existing feature values.

2.1 Library and Programming Languages

Matlab, Python, Scikit Learn

2.2 Approach

- **Regression+PCA**
  In the second part of this project, our task is to deal with unsupervised learning problem, which is, given partial voxel values as features, to predict the remaining voxel values. Our

idea was that there might exists some correlation between features, so we should be able to predict these missing data with existing feature values. In order to predict continuous value instead of the classification problem as part 1, we first tried linear regression model trained by the whole data set. The regression function we used was the matlab built-in function called fitlm. Whats more, to predict 2731 missing voxel value, we have to train 2731 regression models and predict 2731 times. We soon discovered the problem of wasting too much time on training models. Also, among 5903 voxel values there might be some relatively less important features. Considering the reasoning mentioned above, we intuitively used PCA to reduce the dimension from 5903 to 50, or sometimes 100, ended up fairly speeding up the training process. With PCA+linear regression, we reached 0.5 of the RMSE value, which was a good start of this part.

- **Support Vector Regression**
  In order to improve the prediction accuracy, we were trying different kinds of regression models, for instance, support vector regression. Having great experience with support vector machine classifier in project part 1, we decided to try support vector regression in the well-known machine learning library Scikit-Learn. We used Gaussian RBF kernel with gamma=0.00005 and C=50, which was a good set of parameters that wed found by grid search method mentioned above. Considering the variance lost by using dimensionality reduction, we also tried not to apply PCA even if it saved large amount of time. As expected, the result turned out to be greater than previous method, which is 0.494 of RMSE value on the hold out data set.

2.3 Extra idea

Hypothesis
In order to make our regression fit data better, we were trying to test two hypothesis. First, the voxel values of all samples within a class are more similar than that of samples outside the class. Second, some features are more similar than other features. Wed tried to improve our accuracy by adding label and doing clustering under these two assumptions.

- **Adding label**
  First, we intuitively believed, the voxel values of samples within the same class are more similar. Therefore, we classified these samples by class labels into three clusters according to given labels 0,1,3. Then we labeled the unlabeled samples (provided data) to which they belonged to by 1-NN, and predicted missing data class by class. Ideally, we should have gotten better regression to fit data within each class than to fit data as a whole. However, we didnt get expected results by adding this method and the accuracy became even worse unfortunately.

- **Feature clustering**
  Since we assumed that some features are more similar than others, wed tried to clustered features by K-means to find out which features were more similar to each other. Also, another way we found out similar features was processing hierarchical clustering with median linkage to the x,y,z value which are given in project part 1, under the belief that neighbor voxels have stronger correlation. Once we found out similar features, theoretically we could predict missing features more accurately by its similar features. Unfortunately, adding clustering didnt work as well. It didnt reduce the RMSE value effectively. Figure 1 shows that we can divide the feature into several cluster and train them with different model.

2.4 Result

Clustering	PCA+Regression	PCA+Support Vecotr Regression	Support Vecotr Regression
0.529	0.504	0.532	0.494

Table 2: RSME

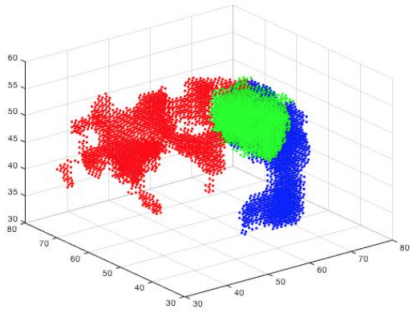

Figure 1: Plot of feature location

3 Semi-supervised Learning

In the final part of our project, we tried semi-supervised classification. With the predicted missing data from part 2, we can treat them as unlabeled data and iteratively learn with the labeled data. The model constructs a support vector machine using all the available data from both the training and working sets. With semi-supervised learning, we hope we can access more information even with unlabeled data. If we only train the Support Vector Machine with labeled data, we may not get good boundary because of the small data size. But if we also consider the unlabeled one and maximize the margin between unlabeled data, we may get a better model.

3.1 Library and Programming Languages

Python, Scikit Learn

3.2 Approach

First, we applied grid search (cross validation) to find the best parameters for the raw training data (501 examples) and trained the initial model based on the labeled data. We then predict the unlabeled data (part 2 examples). Since we trained the model with returning the probability of each label, we selected the examples whose labeling probability dominates the other two as the best prediction, and added them into the labeled data. To determine whether one label dominates others, we take probability 0.8 as the threshold. Afterwards, we trained the model again with the new labeled data and iteratively followed the previous step until we get a convinced label for the whole data.

Algorithm 1 Self Training

Require: Labeled Data (X_l, Y_l), Unlabeled Data (X_u, Y_u)
  while (X_u, Y_u) not empty **do**
    Train a classifier f from (X_l, Y_l).
    Use f to classify all unlabeled items xX_u
    Pick x^* with the highest confidence, move $(x^*, f(x^*))$ to labeled data (X_l, Y_l).
  end while

The algorithm is quite simple and can applied to almost every classifer. However, the algorithm suffers from mistake reinforcement. The algortihm may un-labled a training point if its classification confidence drops below a threshold.

3.3 Result

As the above figure shows, the approach converges in a reasonable speed. It takes approximately 10 minutes to converge and within 30 iterations.

We got the accuracy of 0.642. From the result, we didn't get great improvement. The accuracy is between the hold-out set and the testing set. The possible reason may highly relevant to the accuracy of part 2 since we apply the predicted missing data as the training data. If we have poor performance on part 2, we may got wrong label for the unlabeled data and thus ruined the whole learning model. The first challenge is to improve the accuracy of predicting missing data.