Supervised, Unsupervised and Semi-supervised Learning for fMRI Images

Mila Kankanala
Department of ECE
Andrew ID: mkankana
mila@cmu.edu

Dhruv Nathawani
Department of ECE
Andrew ID: dnathawa
dnathawa@andrew.cmu.edu

Abstract

Supervised learning is carried out to predict labels from fMRI images given a training set. Unsupervised learning is used to obtain the missing voxels in a given distribution of fMRI images. Semi-supervised is used to classify test data using labelled and unlabelled training data.

Introduction

The following paper is broken down into three sections: Part 1, 2 and 3. Part 1 deals with having to build the best possible classifier to predict Y (label) from all of the remaining information (X (brain scan), subject id, event time and voxel locations). Part 2 required the prediction of voxel values: given partial fMRI images (i.e., containing values for only a subset of the voxels) the remaining voxels in the images were predicted and the RMSE was calculated. Part 3 consists of additional analysis computed using the data provided and the results obtained on doing so.

Part 1 - Supervised Learning

1.1 Problem Statement

Each row of the training database consists of 5903 voxels taken from a single fMRI scan. 501 training samples were provided along with their respective voxels. There exists three possible labels denoting the following states:

- No Event.
- Early Stop: Successful stop to an early stop signal.
- Correct Go: Correct button press (within ~500ms) on a trial with no stop signal.

The training is carried out using this database and it's tested on 1001 test samples using different classifiers. The additional details provided about the data (subjectsTrain/ subjectsTest, eventsTrain/eventsTest, x, y, z) weren't used in our implementation.

1.2 Classification

The result required was a 1001x3 matrix where each column corresponded to the probability of the training data falling in that class. In our implementation, we fed in absolute 1s and 0s as the accuracy significantly dropped on feeding in probabilities of the classes. The different methods tested out to classify the test data are covered in the following parts.

1.2.1 Support Vector Machine with RBF kernel

To begin with, a one vs one multi-class SVM classifier was used. Here, a vote is made using three SVM models, each built with one class vs all the others and the best is chosen to classify that test data. Hard voting was used instead of classifying based on probabilities as the latter gave very poor accuracy results. Therefore the outputs the columns denote 0 if it's not classified as that label and 1 if it is that label.

Details of the SVM are as follows:

The Gaussian RBF kernel used is given by,

$$k(x,\ x') = exp(-|x - x'| \ / \ 2\sigma^2)$$

With the kernel width set to 100 and the hinge weight set to 50.

On using this method, an accuracy of 0.55 was achieved. However the majority of the test data was found to be classified with label 3 when this model was used. This exhibits a selection bias as the fraction of training data with label 3 is significantly greater than the others.

Simultaneously, a one vs all multi-class SVM was also tested. The accuracy obtained using this model was 0.54 so it was temporarily discarded in favour of the one vs one classifier.

1.2.2 Cross Validation

In an effort to improve the classifying accuracy, limit overfitting and selection bias, a 10 fold cross validation was carried out during the training. Initially LOOCV (Leave one out cross validation) was tested but it didn't affect the accuracy. On implementing CV, the accuracy improved to 0.566 for one vs one multi SVM. The value of *n* denoting the number of folds of CV was varied between 3, 5 and 10 but it was found to be best at 10 folds. (Figure 2) Additionally, the one vs all model tested better when run with a 10 fold CV and gave an accuracy of 0.58.
The cross validation formula used:

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

K= N (For LOOCV)
K= 10 (For 10 fold CV)

1.2.3 Principal Component Analysis

Since the number of features are 5903, it made sense to perform dimensionality reduction to use the more prominent and dependent features for classification. Therefore, PCA was carried out and only the features with non-zero eigenvalues were chosen. However, on applying PCA on both the CV resulting multi-SVM model and non CV multi-SVM model, there was no observed change in the accuracy. That could most probably be because the training model is not getting any additional useful information to help classify better, only a reduced number from the previous case without PCA.

1.3 Analysis and Results

The accuracies for the different methods were compared and the highest accuracy which was obtained by using multi-SVM one vs all with RBF kernel with kernel width set to 100 and 10 fold CV was chosen as the final result. The number of submission to Autolab was limited to only 16 to prevent the risk of overfitting and getting less accuracy on the test

The chart with the different tested models and their accuracies are shown in Figure 1. The last bar was the chosen classifier.

The accuracy on the test set was found to be 0.52. This is lower than the holdout accuracy and a suggestion as to why it is so would be that the holdout data probably had more instances of label 3 classes and since the classifier to tuned better for that class it did a

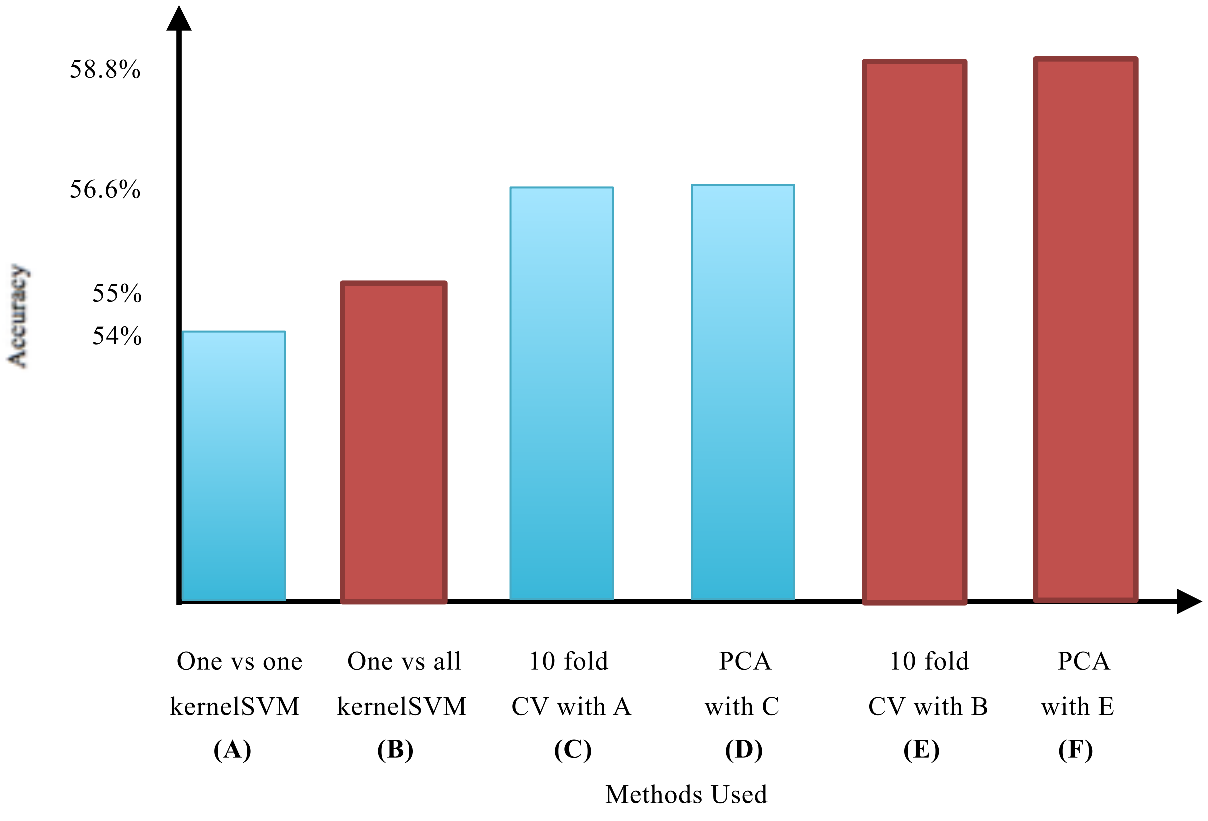

Figure 1: Accuracy vs Methods Used

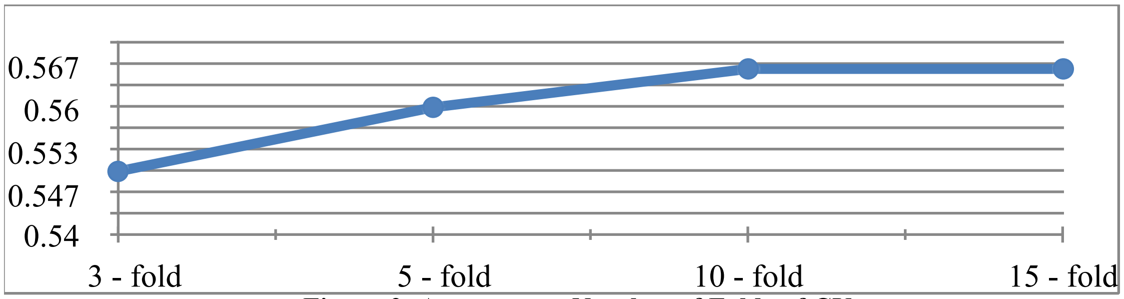

Figure 2: Accuracy vs Number of Folds of CV

better job of classification. However the test data probably had a more even distribution of the different classes and the model suffered on the accuracy due to that.

2 Unsupervised Learning

2.1 Problem Statement

The task of the project was to predict missing fMRI data using the data given to us. We chose to use MATLAB because it is easy to code(when dealing with huge matrices) and it has a large number of inbuilt functions. We stacked up the known and unknown data to create a matrix with a few missing values. The missing values were represented as NaN. We used many different approaches before we got our best result using Regularized Expectation Maximization. The approaches we used are described below.

2.2.1 Sanity Check

Average of all values was a way to make sure we were working accurately with the data and the csv file.

2.2.2 Average over all rows

We submitted the average values over all the rows which led to a surprisingly good result. This might be because the data is highly correlated row wise.

2.2.3 PCA

PCA did not work as well as we hoped. It gave us a very low accuracy and it ran very slowly. So we needed a change of approach.

2.2.4 PPCA

PPCA is an extension on PCA where the Expectation Maximization Algorithm to estimate the principle subspace. We chose PPCA because Probabilistic principal component analysis is preferred to other algorithms, such as the alternating least squares algorithm for handling missing data. PPCA assumes that the values are missing at random through the data set. An expectation-maximization algorithm is used for both complete and missing data.
PPCA worked extremely well when tested on when tested on smaller simulated data. So we expected to get a good result on the fMRI data. But It never finished running for the entire data set on MATLAB. We chose to move onto another approach instead of trying to optimize the code because we felt the input matrix was too large for the MATLAB PPCA function to handle.

2.2.5 kNN Imputation

The MATLAB knnimpute(Data) replaces NaNs in Data with the corresponding value from the nearest-neighbor column. The nearest-neighbor column is the closest column in Euclidean distance. If the corresponding value from the nearest-neighbor column is also NaN, the next nearest column is used. This worked very well on smaller data but gave average results on the real fMRI data.

We were able to improve this result by transposing the matrix and then feeding it into the K-nn impute function and then transposing it back. This function was built by default to have features as rows and observations as columns(because bio technologists deal with loads of features so to keep the matrix small they use features as rows).

Both the rows and the columns k nearest neighbors had a correlation with the missing data(Even though the rows had a higher correlation). So then we averaged out the results of the knnimpute with closest column and the knnimpute with closest rows(after transposing the matrix) and this gave an improvement.

2.2.6 Regularised EM

The MATLAB knnimpute(Data) replaces NaNs in Data with the corresponding value from the nearest-neighbor column. The nearest-neighbor column is the closest column in Euclidean distance. If the corresponding value from the nearest-neighbor column is also NaN, the next nearest column is used. This worked very well on smaller data but gave average results on the real fMRI data.

The EM algorithm penalizes the likelihood which is the mutual information between the missing data and the incomplete data (or the conditional entropy of the missing data given the observations). We can assume the missing data that has a strong (probabilistic) relation with the observations, which implies that the missing data has little uncertainty given the observations. This implies the observations contain a lot of information about the missing data and we can infer the missing data from the observations with a small error rate. This information can be measured by Kolmogorov (or algorithmic) mutual information based on the theory of Kolmogorov complexity or Shannon mutual information based on Shannon information theory.

We got an RMS error of 0.483 using REM. This algorithm gave us the best result of all of our other approaches.

REGEM:

```
        Sample size:                    2502
        Unique missingness patterns:        2
        Percentage of values missing:      18.49
        Stagnation tolerance:              1.00e-02
        Maximum number of iterations:      30
        Initialization of missing values by mean substitution.
        One multiple ridge regression per record:
        ==> one regularization parameter per record.
```

```
Performing regularization on correlation matrices.

Ilter is no. of Iterations
peff is the no. of degrees of freedom for each missing pattern
D(Xmis) is the rms change of missing values
|D(Xmis)|/|Xmis| is the scaled relative change of missing values

Iter     mean(peff)      |D(Xmis)|       |D(Xmis)|/|Xmis|
 1       3.27e+02        2.755e-01          1.248e+01
 2       5.14e+02        1.264e-01          4.571e-01
 3       6.23e+02        7.228e-02          1.819e-01
 4       6.94e+02        4.739e-02          1.024e-01
 5       7.46e+02        3.394e-02          6.748e-02
 6       7.88e+02        2.589e-02          4.883e-02
 7       8.22e+02        2.061e-02          3.747e-02
 8       8.52e+02        1.699e-02          3.006e-02
 9       8.77e+02        1.428e-02          2.476e-02
10       8.99e+02        1.224e-02          2.088e-02
11       9.19e+02        1.068e-02          1.798e-02
12       9.39e+02        9.717e-03          1.617e-02
13       9.56e+02        8.690e-03          1.433e-02
14       9.73e+02        7.975e-03          1.304e-02
15       9.87e+02        7.274e-03          1.181e-02
16       1.00e+03        6.821e-03          1.100e-02
17       1.02e+03        6.366e-03          1.021e-02
18       1.03e+03        6.072e-03          9.688e-03
```

2.3 Results/Plots

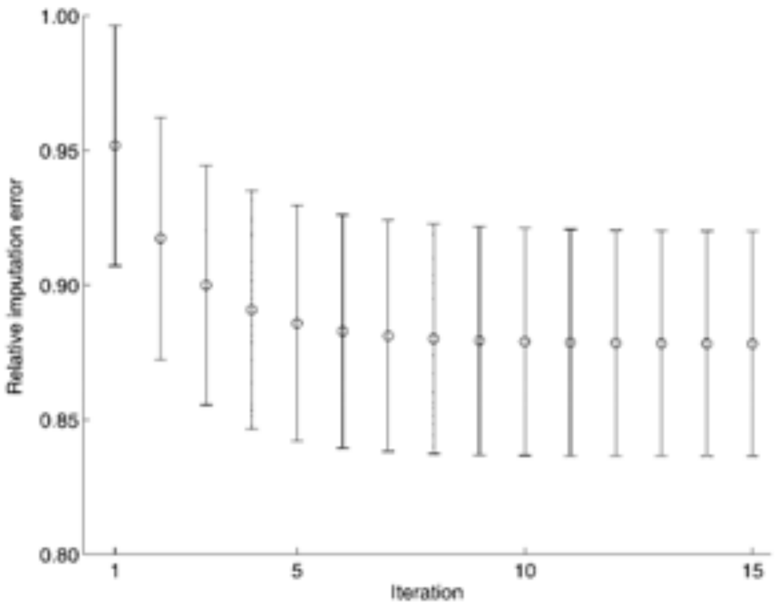

Figure 3: RMS relative imputation errors showing the Mean (circles) and standard deviation (bars) for the first 15 iterations of the regularized EM algorithm.

3 Semi-supervised Learning

The data from the two parts are used to conduct semi-supervised learning over the original Xtest data from Part 1.

3.1 Self-training

Self-training is carried out over the unlabelled portion of the training set. The data from Part 1 consists of 501 training data points each of which has a label associated with it. Additionally from part 2, The 1001 training data with the 5903 voxels function as the unlabelled data. Using both the 501 labelled data and 1001 unlabelled data, the test data of Part 1 can be classified. In self-training, the following steps are carried out:

- The labelled data is used to train the classifier (an SVM classifier is used)
- This model is run over the unlabelled data to obtain pseudo results
- The whole data set (1502 data points) is used to classify the test data in Part 1.

This process improved the accuracy from 58.8 that we had obtained in Part 1 to 60.2.

4.1 Semi-supervised clustering

In this method, search-based semi-supervised clustering is carried out. Here, the training data-labels from part 1 is used to bias the search for appropriate partition. This labelled data is used to generate the initial seed clusters, as well as the use of constraints generated from labeled points to guide the clustering. The clustering model used is a probabilistic style K-Means, where the labeled data provides prior information about the conditional distribution of hidden category labels.

Here, K-Means clustering is done by finding the distance between observations by using the Euclidean distance formula between the data points and associating the label to the unlabelled point based on which labelled data it's closest to.

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

The number of clusters K is varied between 10 to 30 in increments of 2 and the optimum result is found at 22 clusters.

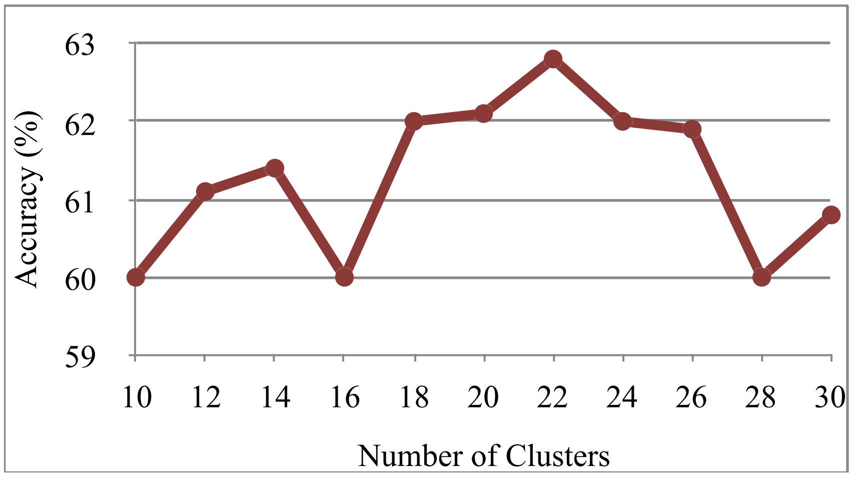

Figure 4: Accuracy vs Number of K-means Clusters used in semi-supervised learning

An accuracy of 62.6 is obtained by carrying out K-means clustering (using number of clusters = 22) for the unlabelled data followed by running the test data with this model which is a definite improvement over just using supervised learning from the 501 data points in Part 1.

References

[1] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney, (2002), Semi-supervised Clustering by Seeding, (ICML-2002)

[2] A Comparison of Methods for Multi-class SupportVector Machines, Chih-Wei Hsu and Chih-Lin

[3] http://vikas.sindhwani.org/svmlin.html

[4] http://www.mathworks.com/help/stats/ppca.html

[5] http://www.mathworks.com/help/bioinfo/ref/knnimpute.html

[6] http://climate-dynamics.org/software/#regem

[7] The Regularised EM Algorithm, Haifeng Li, Keshu Zhang and Tao Jiang.

[8] http://www.mathworks.com/help/stats/pca.html