**Part 1: Methods and Accuracies**

We attempted to apply different methods to the training set. The sections below have been separated by the categories given in the project directions outline.

Generative Method

|  |  |  |  |
| --- | --- | --- | --- |
| Training Method | Training Set | Error type | Error |
| GMM |  | Cross Validation |  |
| GMM |  | Cross Validation |  |

Discriminative Method

Some of the more accurate methods were SVM’s trained on the word counts:

|  |  |  |  |
| --- | --- | --- | --- |
| Training Method | Training Set | Error type | Error |
| SVM | train\_img | Leaderboard | 0.7809 |
| SVM, C = 300,K=50\* | train\_img | Leaderboard | 0.8009 |

\*Matlab Function: fitcsvm(X,Y, ‘BoxConstraint’, 300, ‘KernelScale’,50)

Due to the success of the straightforwardly training an SVM, we attempted concatenating the features together to see if it would produce better results. Unfortunately this was not the case:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Training Method | | Training Set | | Error type | | Error |
| SVM | train\_img,train\_cnn\_feat | | Cross Validation | | 0.6270 | |

We also attempted random forests, which produced relatively agreeable results:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Training Method | | Training Set | | Error type | | Error |
| Random Forests | words\_train | | Cross Validation | | 0.75 | |

Instance-based Method

|  |  |  |  |
| --- | --- | --- | --- |
| Training Method | Training Set | Error type | Error |
| KNN | words\_train | Cross Validation | 0.6318 |
| KNN | train\_img | Cross Validation | 0.5353 |
| KNN | train\_cnn\_feat | Cross Validation | 0.5744 |

Regularization Method

Dimensionality Reduction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DR Method | Training method | Training set | Error type | Error |
| PCA | SVM | train\_img | Cross Validation | 0.5044 |
| PCA | SVM | train\_cnn\_feat | Cross Validation | 0.5724 |
| PCA | SVM | words\_train | Cross Validation | 0.7854 |
| PCA | SVM | words,color,cnn,img\_prob | Cross Validation | 0.7914 |
| PCA | KNN | train\_img | Cross Validation | 0.5322 |
| PCA | KNN | train\_cnn\_feat | Cross Validation | 0.5638 |
| PCA | KNN | words\_train | Cross Validation | 0.5920 |

**Part 2: Analysis of Results**

Our MVP: SVM

One of the earlier methods we tried was the SVM on the word counts. As it became apparent that this simple method produced the highest accuracies, we worked to tweak the SVM by optimizing its training parameters. Matlab 2016b allows an option called ‘OptimizeHyperparameters’ in the fitcsvm model. This then produces a table denoting the relationship between the objective function and various parameters such as BoxConstraint () and KernelScale (a scaling of data points before the kernel is applied, effectively scaling the margin on either side of the decision hyperplane). Choosing the parameters that minimized the objective function improved the resulting SVM accuracies by a few percentage points.

Dimensionality Reduction via PCA

Running PCA before other training models did not seem to improve the accuracies. This can be partially attributed to the large size of the resulting models when many principal components were utilized. When we concatenated word count, color, cnn features and img\_prob, we kept 500 of 9000 principal components for an 83% reconstruction accuracy. This yielded a model of 82MB, exceeding the allowed upload threshold of 50MB. Ultimately, the stipulation on filesize restricts the PCA’s ability to be of practical use.

Weighting Multiple Methods by Majority Vote

Another major idea to improve the net accuracy of our models was to aggregate the information from multiple models. This was based on the assumption that, if we had multiple tests with reasonable accuracies (say, >66%), conducting a majority vote from each method to determine the final predictions should result in a higher accuracy than each individual test. We also had the idea assign different weights to each test, by framing the proposition as a convex optimization problem and solving via Matlab’s cvx plugin.

However, this method ran into a large issue; we did not end up having many credible tests. When we tried a majority vote with our top three methods (SVM on word count, optimized SVM on word count and random forests), the optimized weights produced the same results as equal weighting, which ended up producing predictions that matched 98.2% with the best method, in our case the optimized SVM.

**Part 3: Visualization of Results**