

DecMeg2014 Model Documentation - 3rd Place

Nathan Hammes

Location: Indianapolis, IN
Email: nhammes@gmail.com

1 Summary/General Approach

The final model implements preprocessing, followed by pooled classification using lasso and SVM^{light}'s standard and transductive support vector machines (SVMs), using the associated MATLAB MEX interface[4,3]. An important final reclassification step with the output of the 3 classifiers achieves much improved accuracy. For preprocessing, the data is filtered with notch, low-pass, and averaging filters. Sensors 1-162 are then removed, and only information corresponding to 0-0.5s post-stimulus are retained.

After the 3-step classification, the probabilistic results are simply added to achieve preliminary labels and a cross-validated accuracy of just above .70. A probabilistic interpretation of the medians of the preliminary class labels allows for a weighted distance determination of each trial from each class mean (of the medians), resulting in a final cross-validated accuracy of .736.

2 Feature Selection

Data is initially filtered with a 50Hz notch filter for powerline noise and a 100Hz low-pass filter, followed by convolution with an averaging filter. Based upon 6-fold overall single-subject decoding accuracies, the averaging filter that is convolved along the temporal dimension of the data takes the form

$$a_t = \sum_{i=-q}^q b_i y_t \quad q < t < 375 - q \quad (1)$$

with $q = 5$ and the weights $b_i = \frac{1}{2q}$ for $|i| < q$ and $b_i = \frac{1}{4q}$ otherwise. Each time series \mathbf{y} is from a given sensor and comprises a set of y_t given $t = 1, 2, 3 \dots 375$. See [2] for more information. Just before the pooled classification process, data from both the magnetometers and gradiometers corresponding to sensors 1 through 162 are discarded, and the values at each remaining sensor are z-scored within each subject, identically to the z-scoring process performed by the benchmark code. Z-scoring in this case means to center each feature of the data at 0 and give each a standard deviation of 1.

For post-lasso/SVM/transductive SVM, features are developed similarly, but no z-scoring is performed, and only the gradiometers from sensors 163 to 306 are used with the full second of information post-stimulus.

3 Modeling and Training

As described above, the 3 classifiers used are lasso and 2 types of support vector machines. The *lasso* used is MATLAB’s native implementation, and the parameters were not changed from that of the benchmark, i.e. $\alpha = .9$ corresponding to an elastic net with strong L1 regularization, and $\gamma = .005$. Both the standard and transductive SVMs used the default parameter tuning. Although at this point, the transductive SVM achieves a score of .70 overall, lasso achieves .69, and normal SVM is somewhere in between, all outputs are simply combined. There is literature available for probabilistic interpretation of SVMs, but I did not delve into the topic. The resulting accuracy is slightly above .70.

The most interesting observation and best contribution I believe my model can provide is based on an observation made from classifying the data of a single subject based on each trial’s distance from the true means of the positive and negative classes. When this is done using all 306 sensors and all filtered data post-stimulus, accuracies are $\sim 90\%$ for each of the training subjects. Using this information, a probabilistic representation of the means of the initially classified data is developed by partitioning the **non-standardized (i.e. non z-scored)** data 10-fold and keeping the medians of each class for each partition (a partition in this case represents 90% of the data). This process is repeated 9 more times to collect 100 points for the median of each of the $96 \cdot 250 = 24000$ features.

At this point, for a given feature, the Bhattacharyya distance is a reliable measure of the distance of the positive-class mean distribution from that of the negative class. Taking μ_p , μ_n , σ_p , and σ_n to be the respective means and the standard deviations of the positive and negative classes at a given feature, the Bhattacharyya distance is calculated as[1]

$$B = \frac{1}{4} \ln \left(\frac{1}{4} \left(\frac{\sigma_p^2}{\sigma_n^2} + \frac{\sigma_n^2}{\sigma_p^2} + 2 \right) \right) + \frac{1}{4} \left(\frac{(\mu_p - \mu_n)^2}{\sigma_p^2 + \sigma_n^2} \right) \quad (2)$$

The Manhattan/city block distance of each trial feature from the mean (of its medians) is calculated, multiplied to the corresponding B for that feature as a weight, and summed with the distances from all other features to achieve final distances for that trial from each class. Final classification labels are assigned so as to equally distribute the trials between the two classes.

4 Functions/Code

I created 2 solutions - one using and one without the computationally intensive transductive SVMs. Any times mentioned are those for a system with an i5 processor and 8GB of RAM. Functions are named clearly according to the tasks they perform:

Entire program

- **DecMegRunAll** - Runs the full program with all 3 classifiers, adding paths as needed. Takes approximately 17.5 hours.

- **Filters.mat** - not a function, but 2 MATLAB filter objects 'NotchPower50', and 'LowPass100', filters for 50Hz powerline noise and 100Hz or greater frequencies, respectively. Used in prepConvOccFiles, DecMegPostclassification, and DecMegPostclassificationJustSVMAndLasso.
- **createYStruct** - Creates and stores the structure variable 'yStruct' to be used for ensuing programs for labeling the training subjects.
- **loadIds** - Loads Ids from each subject, and stores them in two different forms ('idsCell' and 'idsAll') in a separate file 'idsAll.mat'
- **prepConvOccFiles** - Performs preprocessing on the original data and stores it in separate ~40MB files prefixed with 'ConvOcc' (Convolution and only Occipital sensors)
- **createFeaturesNoTime** - Original createFeatures function as part of the official competition code with a couple adjustments - it only requires one input, and only performs reshaping.
- **createFeaturesZScoreOnly** - Only the z-scoring portion of the original createFeatures function.
- **DecMegCrossValandTestLasso** - Loads the ConvOcc training files, creates a lasso classifier from them and classifies the testing data from respective ConvOcc files. Results are stored in 'DecMeglassoCell.mat' with variable 'lassoCell'. Commented code in the center of the file allows for cross-validation of the 16 subjects.
- **DecMegSVMPrepped** - Loads the 'prepped' ConvOcc training data, creates a classifier, and classifies each testing subject from its respective ConvOcc file, one at a time. Results are stored in 'DecMegtestRegCell.mat' with variable 'testRegCell'.
- **DecMegTransSVMPrepped** - Loads the 'prepped' ConvOcc training data, loads each testing subject from its respective ConvOcc file, creates a classifier, and then classifies that test subject. Results are stored in 'DecMegtestTransCell.mat' with variable 'testTransCell'. An extra text file 'trans_predictions' is output by the program but is unused. Takes approximately 17 hours.
- **DecMegPostclassification** - Loads the probabilistic predictions stored in 'DecMeglassoCell.mat', 'DecMegtestRegCell.mat', and 'DecMegtestTransCell.mat'. Loads the variables in 'idsAll.mat'. Loads original data, performs an alternative preprocessing and selection of data (only the gradiometers at the back of the sensor array, using the full 250 timepoints post-stimulus), and runs the weighted distance calculation on each subject. Final output is a submission file 'DecMegSubmission.csv'. Additional commented code allows for visualization of the locations in the array where the medians have not crossed, which gives insight into where the subject's data is most separable. The setting of the random number generator should output a file with an accuracy of .71404 on the private leaderboard. I had not previously seeded the generator, and the file corresponding to the 3rd place submission resulted in .71316 on the leaderboard.

Extra Functions for a Run of Lasso and Standard SVM Only

- **DecMegRunJustLassoAndSVM** - Runs the program with 2 classifiers (Lasso and the normal SVM), adding paths as needed. Takes approximately 23 minutes.
- **DecMegPostclassificationJustLassoAndSVM** - Identical to DecMeg-Postclassification, but only loads Lasso and standard SVM predictions. Final output is a submission file 'DecMegSubmissionLassoSVMOnly.csv', which achieves a score on the private leaderboard of .70270, and has the potential to save some time for those who would like a less computationally intensive version of the code.

5 Dependencies

The solution was successfully generated using MATLAB 2013a on a 64-bit Windows machine with 8GB of RAM. The function *lasso* is a part of the Statistics Toolbox, and the required files for SVM^{light} and the corresponding MATLAB MEX interface are provided in the model code's zip folder with prior permission from authors as needed. Most of this code also executed successfully using a computer with MATLAB 2010b installed, and it is doubtful any version of MATLAB between these two will cause any issues with code execution.

6 How To Generate the Solution

1. Unzip the provided compressed folder in a suitable location.
2. Place the data from the competition into the 'DecMeg2014ThirdPlaceCodeAndDoc' folder.
3. Open a compatible MATLAB version.
4. Navigate within MATLAB to within the 'DecMeg2014ThirdPlaceCodeAndDoc' folder.
5. Run either DecMegRunAll.m or DecMegRunJustLassoAndSVM.m, as desired. The solution should output the appropriate submission file.

A warning: the first 2 times I attempted to use the MEX interface, MATLAB closed unexpectedly without error. The code ran without issue on any computer I tried after these two initial hangups. I believe the provided functional code should not cause problems, but I feel I must mention the issue regardless.

7 Additional Observations

The choice of q for the averaging filter above can boost scores significantly, yet it varies depending on which subject one is working with. I found with $q=5$, my best 6-fold single subject decoding accuracy was achieved at .870. If the proper q between 1 and 20 is chosen and known ahead of time for each subject, this can be bumped to .877.

“Postclassifying” after the initial supervised classification, as mentioned above, is the most important step of the model. This step improved scores on subjects by as much as 16% (for subject 8, from .67-.83) and only hurt the score of subject 6 (.67 down to .63). By simply comparing test subjects to the training subjects by the Manhattan distances of the means of the original data or their derivatives, it can be seen that subject 18 on the public leaderboard is quite dissimilar from the others, while those subjects on the private leaderboard seem to be relatively similar to the training subjects.

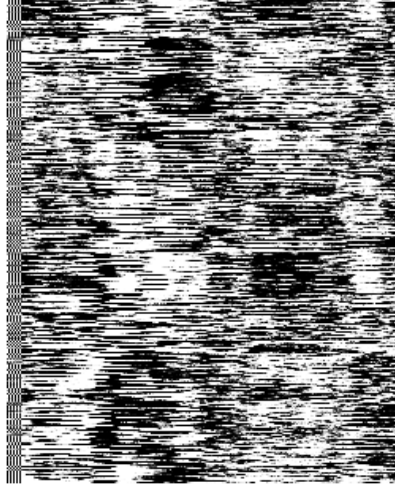
Unfortunately, as a result, public and private leaderboard scores were extremely rough estimates of each other for most contestants. As mentioned by another Kagglers, fchollet, in the forums, the variability we were seeing between subjects on such a small test set makes this a difficult problem, and I by no means claim that my model was the 3rd-best in predicting unseen subjects. It is only the 3rd-best at predicting the 4 subjects in the test set, and would need some adjustment to be more truly generalizable.

8 Simple Features and Methods

I’ll reiterate here Alexandre’s (who achieved 1st place by a wide margin on both leaderboards) statement from earlier in the competition: that the best model could possibly be obtained by deriving information from the data of each test subject. It is relatively easy to throw pooled or subject-to-subject classifiers at the data, but improving the transfer learning process seemed to require some form of combining or refreshing the results of the classifiers based on the distribution of the data of each test subject. The simple method explained above achieved at least 3% improved cross-validation accuracy, and more complex methods can achieve even greater improvements.

9 Figures

Below is a figure that is a grayscale version of the output of the commented code of DecMegPostclassification.



(a) $+ \text{ class } > - \text{ class }$

Fig. 1: A 306-sensor by 250-timepoint representation of the number of times (out of 100 total) in subject 17 when the positive class median is greater than that of the negative class. The structure of noise can be seen by noting the approximately 10 left-most timepoints where (I assume) the subject has not yet reacted to the stimulus.

References

1. G.B. Coleman, H.C. Andrews, "Image Segmentation by Clustering", Proc IEEE, Vol. 67, No. 5, pp. 773-785, 1979.
2. MATLAB's page for moving average filters: <http://www.mathworks.com/help/econ/filtering.html>.
3. T. Briggs. MATLAB Interface for SVM-Light. <http://sourceforge.net/projects/mex-svm/> File: 'svm_mex601r14.zip'. Date: 14.09.2009. Accessed 06.2014.
4. T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999. <http://svmlight.joachims.org/>. Version: 6.02. Date: 14.08.2008. Accessed 06.2014.