Detecting Distracted Driving with Neural Nets

Joseph Blubaugh

16 November 2016

Overview

The first pass analysis used summarized statistical measures to fit both a random forest and neural network model to the data. The results primarily showed that aggregating the data over an entire event prevented the two algorithms from detecting texting events. Also, because the data was aggregated, there were very few data points to model between testing and training sets. The random forest algorithm was a better choice than the neural network because of the large number of predictor variables relative to the number of observations. Both models however, were guilty of overfitting. For the next phase of this analysis, I am focussing on looking at the data in its original form. I have chosen to focus on the neural network rather than the random forest mainly because of the data structure. In the previous analysis the aggregated data set contained 119 observations with 40 predictor variables. Since I am using the original data the data set contains more than 1 million rows on 11 predictor variables.

Neural Nets applied to Distracted Driving

In order to standardize the results of my analysis, I have elected to use the R package caret to train all of my models (http://topepo.github.io/caret/index.html). Caret is a modeling framework that allows you to run many different types of models with different combinations of parameter sets at the same time. It can easily use models from other popular packages and offers a rich set of validation tests and diagnostic plots. Another benefit to using this package is enabling parallel computing of cross validation tasks.

Time Omission

Time has been intentionally omitted from all of the models. The training/testing strategies in this analysis include either splitting the data roughly in the middle of the simulation or sampling from the entire simulation. Since texting events happen during specific fixed intervals of each simulation, a neural net will learn that time is an important factor in determining texting and the results will be biased. I discovered this when I initially included the time variable in the split data set. The training set performance measured very high with an overall accuracy of .95, but the testing accuracy was 0 because the model only saw time observations between 0-365 seconds and so it predicted no texting for all observations >365.

For the following models I am using the nnet package which ships with base R. I am using k=10 cross validation for all models and Im training on different combinations of size and decay parameters. Training and testing sets are all approximately a 50/50 split.

The following sections describe the values in the model results table:

Data Processing:

- Original: Data in its original form. Variables are the 8 emotions measured at .03 second intervals.
- Differencing: First order differening of each original observation.
- Moving Avg: Running averages n=30 for all of the emotions.
- 1/2 Sec Cut: Time is cut into half second intervals with the average value of each emotion recorded over the interval.
- 1/2 Sec Diff: First order differncing of the 1/2 second cut data.
- 1/2 Sec Cut Stat: In addition to mean, sd, min, max, iqr, and median are calculated.

Data Split:

- 365 Split: The data are split at the 365 second of the simulation. This is approximately half way through the entire simulation and is in between texting events for all subjects.
- Entire Sim: The training set is randomly selected from the entire simulation. The testing set are all observations not selected for the training set.

Model Specifics:

- MaxItr: Max Number of iterations allowed for the alogrithm to converge.
- Converged: Whether the model convered or not during training. If the model did not converge then it was still improving when it hit the iteration limit.
- Size: Number of hidden nodes in the model.
- Decay: The rate of decay towards zero placed on the weights when they do not update after an interation.
- Training and Testing: Total percentage of correctly predicted observations.

General Model Form:

- Texting ~ Subject + Age + Gender + Anger + Contempt + Disgust + Fear + Joy + Sad + Surprise + Neutral

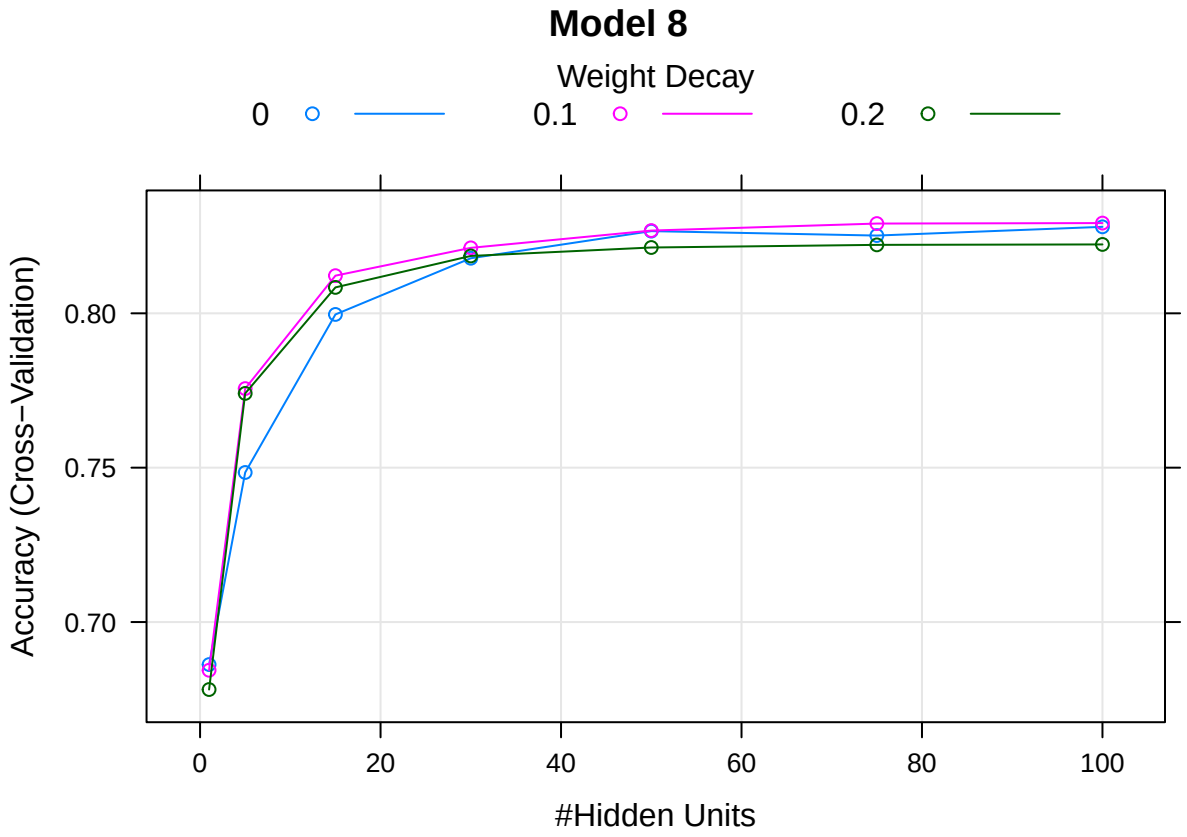
Model Results:

Model	Data Processing	Data Split	MaxItr	Converged	Size	Decay	Training	Testing
Model 1:	Original	365 Split	500	No	15	.20	.783	.690
Model 2:	Original	Entire Sim	500	No	100	.10	.798	.796
Model 3:	Differencing	365 Split	500	Yes	15	.05	.594	.571
Model 4:	Differencing	Entire Sim	500	Yes	15	.05	.592	.591
Model 5:	Moving Avg	365 Split	500	Yes	30	.05	.525	.527
Model 6:	Moving Avg	Entire Sim	500	Yes	10	.00	.528	.528
Model 7:	1/2 Sec Cut	365 Split	500	Yes	100	.00	.884	.722
Model 8:	1/2 Sec Cut	Entire Sim	500	Yes	100	.10	.839	.816
Model 9:	1/2 Sec Diff	365 Split	500	Yes	100	.10	.716	.647
Model 10:	1/2 Sec Diff	Entire Sim	100	Yes	100	.10	.687	.625
Model 11:	1/2 Sec Cut Stat	365 Split	100	Yes	100	.00	.849	.717
Model 12:	1/2 Sec Cut Stat	Entire Sim	100	Yes	75	.10	.826	.809

Interesting Observations

- Model 8 currently has the best overall performance.
- Overall the models trained over the entire simulation fair better than models trained on the time split data sets. This suggests that there are could be distinct differences between two texting segments.
- Model 2 took 3 days to train on 6 cpu. Its interesting that it still did not converge during this time and would have continued improving if given more iterations. The testing and training scores are virtually even so Model 2 could probably be given more iterations.
- Overall the models built on data that was differenced perform the worst. The differnced models using the 30 second intervals is much improved over the the original differencing models.
- Models 2 and 8 are using the same dataset except that model 8 has cut the data into 30 second intervals. Performance actually improves when you look at the data at a slightly higher level indicating that looking at the raw data measured every .03 seconds might actually make inference more difficult

```
#############################################################
## Model 8: Training and Evaluation

## Set Cross Validation
fit.control = trainControl(method = "cv", number = 10)

## Create combination of model parameters to train on
search.grid = expand.grid(decay = c(0, .1, .2),
                          size = c(1, 5, 15, 30, 50, 75, 100))

## Limit the iterations and weights each model can run
maxIt = 100; maxWt = 15000

fit = train(Texting ~ . - Time, mdl.08.train, method = "nnet",
            trControl = fit.control,
            tuneGrid = search.grid,
            MaxNWts = maxWt,
            maxit = maxIt)

40053 samples, 11 predictors
2 classes: '0', '1'

Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 40053, 40052, 40052, 40053, 40052, 40053, ...
Resampling results across tuning parameters:

  decay  size  Accuracy   Kappa
  0.0       1  0.6862218  0.3201710
  0.0       5  0.7484432  0.4673945
  0.0      15  0.7996313  0.5822628
  0.0      30  0.8178326  0.6214050
  0.0      50  0.8266183  0.6403386
  0.0      75  0.8251806  0.6368128
  0.0     100  0.8280119  0.6437033
  0.1       1  0.6843809  0.3225985
  0.1       5  0.7756104  0.5279786
  0.1      15  0.8121924  0.6082412
  0.1      30  0.8212030  0.6276012
  0.1      50  0.8267981  0.6394863
  0.1      75  0.8290677  0.6442134
  0.1     100  0.8292249  0.6447110 **Best Model**
  0.2       1  0.6781336  0.3088442
  0.2       5  0.7740603  0.5246459
  0.2      15  0.8083948  0.5996885
  0.2      30  0.8185739  0.6216010
  0.2      50  0.8213152  0.6274478
  0.2      75  0.8221691  0.6291242
  0.2     100  0.8223039  0.6294092
```



Neural Network Confusion Matrix

```
      Predicted
Actual     0     1
     0 23398  2520
     1  4172 14413
```

(Training Set) Overall Performance: 0.8391441

```
      Predicted
Actual     0     1
     0 22738  2941
     1  4735 14089
```

(Testing Set) Neural Net Overall Performance 0.816965

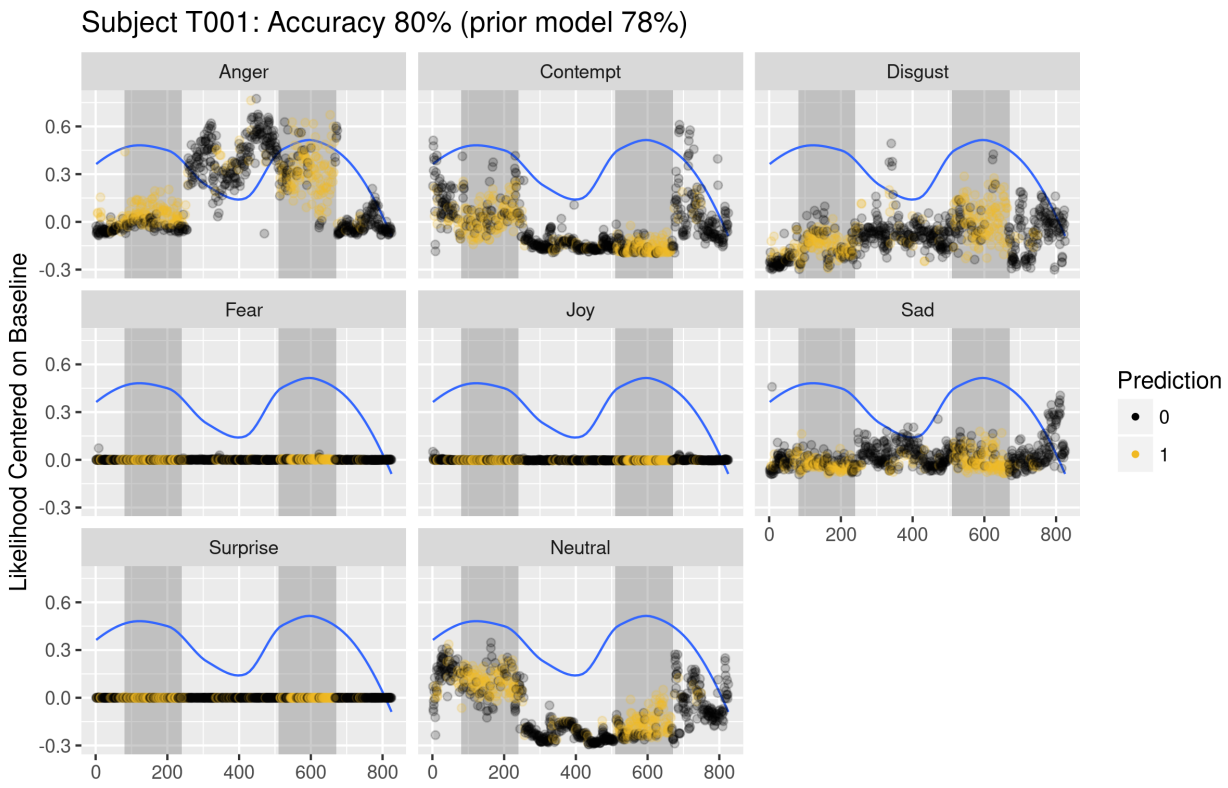


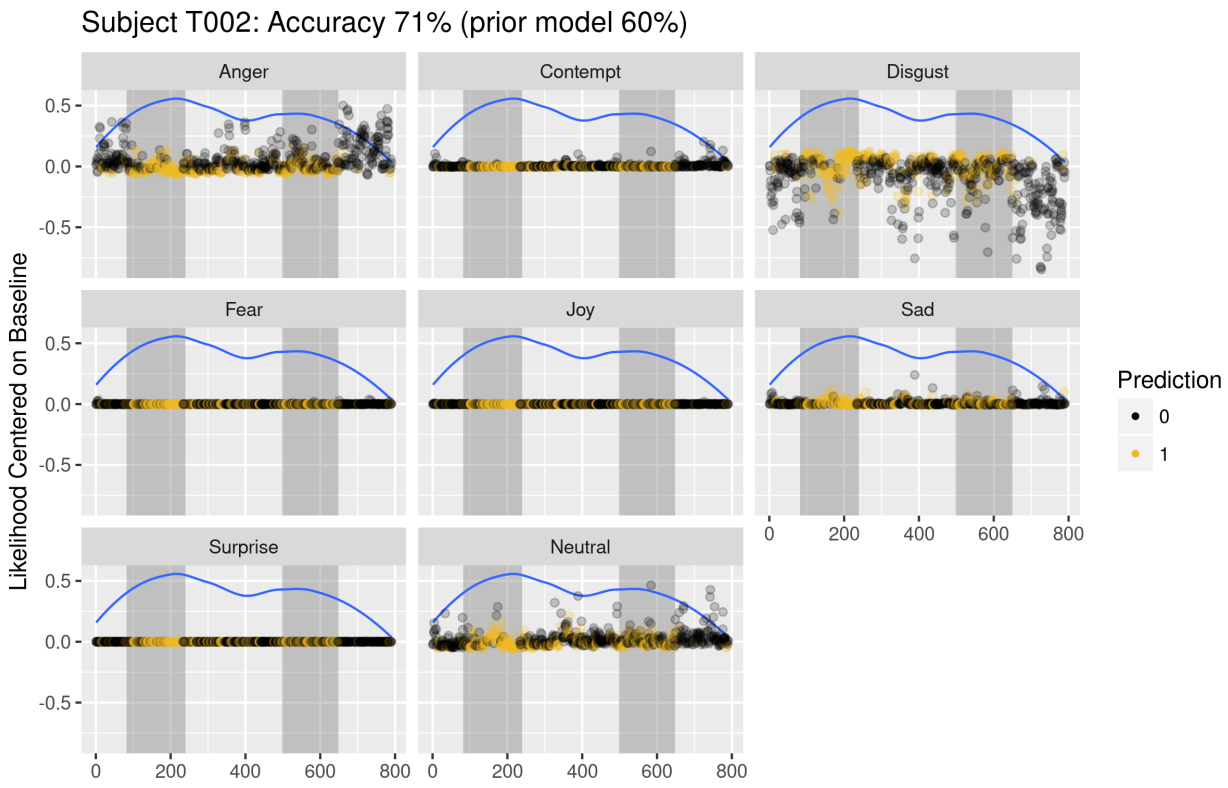
Figure 1: Model 8 Prediction

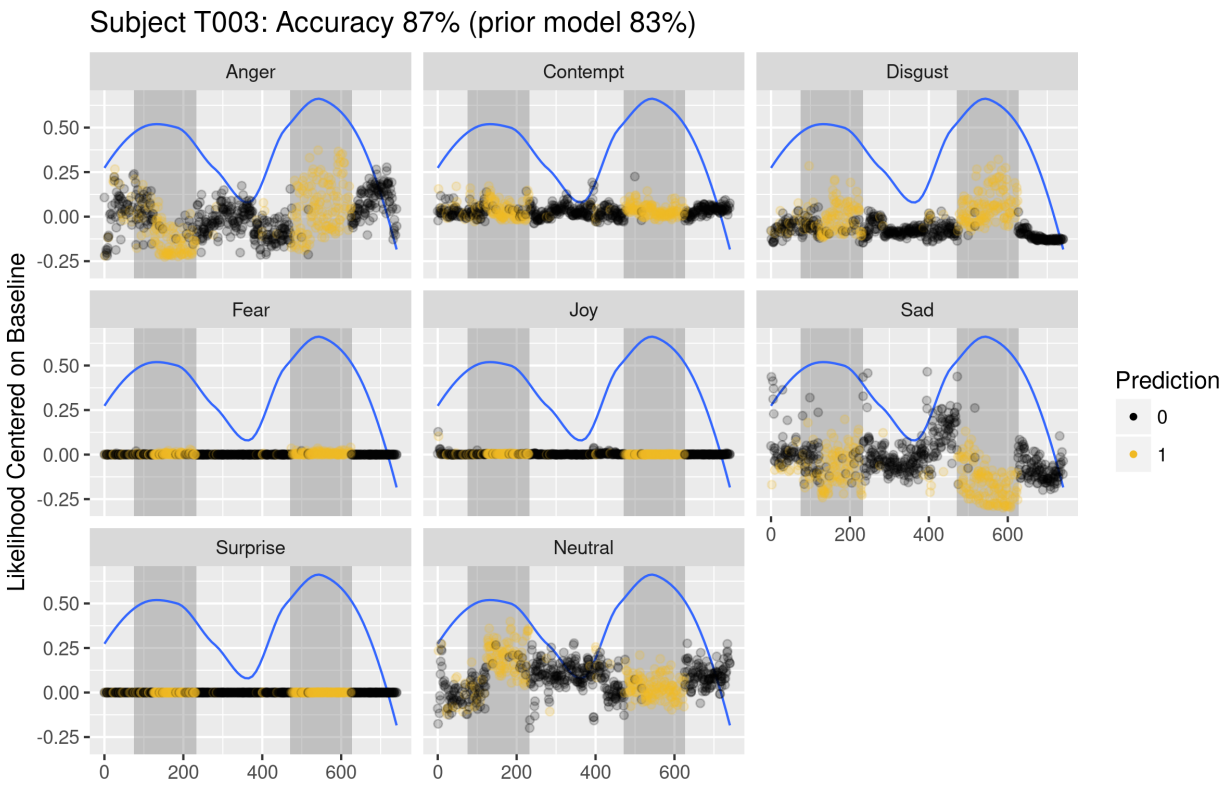
Plots

The following 3 plots overlay the prediction for model 8 against the test sets for the first 3 subjects. Yellow dots represent predictions for texting events and the black dots are no texting events. The blue curve is a loess curve for the probability prediction. It shows the overall trend for the predictions by time. Compared to the model in the previous analysis, all 3 Subjects accuracy has increased.

```
## `geom_smooth()` using method = 'loess'
```

```
## `geom_smooth()` using method = 'loess'
## `geom_smooth()` using method = 'loess'
```




Subject T003: Accuracy 87% (prior model 83%)
Anger
Contempt
Disgust
Fear
Joy
Sad
Surprise
Neutral
Likelihood Centered on Baseline
0.50
0.25
0.00
-0.25
0
200
400
600
Prediction
0
1

Accuracy by Subject:

Table 2: Table continues below

	T022	T007	T086	T006	T083	T035	T018	T081	T076	T020	T064
Train	0.980	0.922	0.958	0.938	0.951	0.952	0.929	0.922	0.946	0.934	0.926
Test	0.971	0.949	0.948	0.938	0.937	0.933	0.928	0.927	0.918	0.917	0.912

Table 3: Table continues below

	T012	T009	T074	T088	T013	T032	T003	T080	T011	T015	T016
Train	0.921	0.909	0.922	0.933	0.883	0.918	0.892	0.901	0.903	0.867	0.891
Test	0.907	0.907	0.905	0.900	0.883	0.880	0.878	0.873	0.865	0.864	0.859

Table 4: Table continues below

	T044	T060	T005	T079	T008	T039	T082	T010	T073	T029	T024
Train	0.885	0.907	0.855	0.894	0.868	0.849	0.859	0.845	0.844	0.813	0.812
Test	0.856	0.856	0.848	0.840	0.840	0.837	0.829	0.829	0.826	0.820	0.810

Table 5: Table continues below

	T042	T046	T051	T001	T017	T061	T084	T066	T077	T036	T040
Train	0.848	0.843	0.856	0.853	0.798	0.797	0.812	0.843	0.795	0.791	0.774
Test	0.809	0.807	0.805	0.798	0.798	0.797	0.788	0.787	0.770	0.768	0.762

Table 6: Table continues below

	T021	T033	T014	T031	T019	T004	T002	T023	T041	T054	T034
Train	0.803	0.767	0.806	0.795	0.726	0.723	0.714	0.740	0.693	0.739	0.697
Test	0.747	0.746	0.739	0.732	0.726	0.712	0.711	0.705	0.681	0.677	0.675

	T047	T025	T027	T038
Train	0.697	0.690	0.539	0.551
Test	0.674	0.670	0.516	0.513

Conclusions and Next Steps

So far the neural net models have shown encouraging results in the ability to detect texting events for subjects based on facial expressions. It clearly helps model performance to consider each individual separately. In my next session I will focus on trying to extract, interpret, and visualize model weights. I would also like to look into some additional neural net models that have recursive features and that can possibly handle the time variable.