Classifying Human Activity Data With Machine Learning Techniques

By Thomas Hanson

Introduction:

Classification is a kind of machine learning task done on data with an arbitrary number of features and a 'class' or 'label'. The goal of such a task is to develop a model that is able to analyze the features of a datapoint where the true label is unknown and predict the datapoint's label. Modern examples include facial recognition, handwriting parsing, and natural language processing. This task is fully supervised, meaning the model requires data to train on that all has the true label attached to it, unlike clustering methods where the true labels are unknown and an artificial function is applied to group data points together. This work was focused on a dataset of human activity data where the goal was to predict the activity the subject was performing, which was one of: Walking, Standing, Sitting, Laying, Walking Upstairs, or Walking Downstairs. Using two machine learning techniques, namely Gradient Boosted Decision Trees [1] and Neural Networks [2], models were made to accurately predict these labels. One of these models was then used to show that use of all available features was largely unnecessary, and similar results could be achieved using approximately a tenth of the features. All analysis was done in jupyter notebooks run on the University of Rochester's Bluehive Linux Cluster [12].

Data Collection and Preparation:

For my analysis, data was selected by Professor Anand as part of the Data Science Capstone/Practicum course [3a] offered in the University of Rochester's Data Science department [3]. The data itself is the "Human Activity Recognition Using Smartphones Data Set" from the UCI Machine Learning Repository [4] and consists of 561 predictive features (numerical), 10299 datapoints, and a class label with 6 potential values (non-numeric). The data were downloaded and analyzed with the R and Python programming languages [5, 6]. The data are a collection of measurements taken by a smartphone such as acceleration while the subject performed one of the six activities.

The data were partially preprocessed from their raw form before downloaded, specifically each trial of the experiment was aggregated together and presented as one datapoint rather than as a set of points over time. To further assist in analysis, the data was split into a testing and training set using a function provided by Sci-Kit Learn's Model Selection module [7] using a testing size of .2, meaning that only 80% of the dataset was used for training the models while the other 20% was reserved for testing the model's accuracy on novel data. For the neural network, the activity labels were mapped to integers from 0 to 5 to better fit the model.

Exploratory Analysis:

The data was initially analyzed in R [5] to explore the layout of the data, as well as determine some properties of it. A covariance matrix [8] was created on the 561 predictive features, and showed generally low covariances, with the overall mean being 0.0227 and a median of 0.0098, meaning that all of the data is generally distinct from each other. Additionally a bar graph was made using the ggplot2 library [9] of the prevalence of each activity label to see if some were more represented than others, which could bias our model towards those labels.

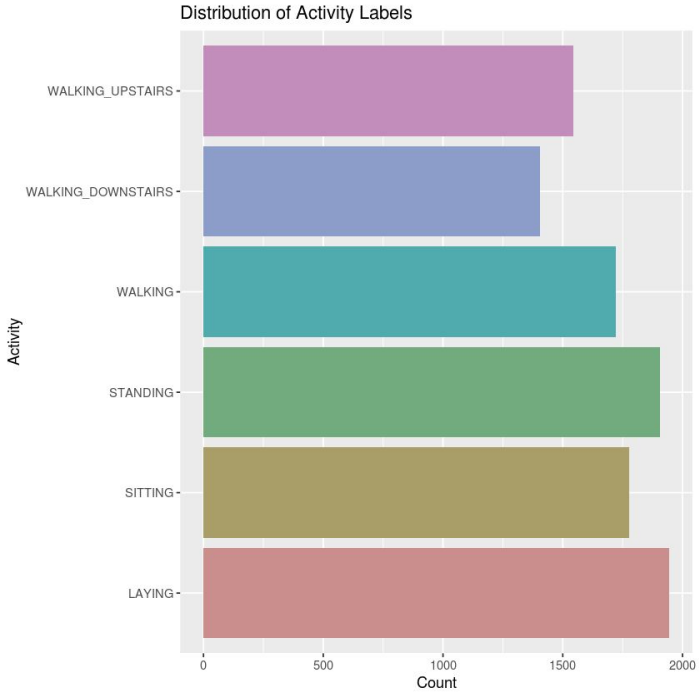

Figure 1: Distribution of Activity in Dataset, shows generally equivalent distribution

This plot shows that all of the labels are similarly well populated, but the difference between walking downstairs (the lowest) and laying (the highest) was something to take note of.

Methods:

Neural Network:

The first model made was a vanilla Feed Forward Neural Network implemented with Pytorch [10]. A neural network (as seen to the right) is a directed acyclic graph consisting of layers of nodes, each of which have an activation function that takes their inputs and creates an output. The network created uses a sigmoid function, also known as the logistic function (used in logistic regression), meaning that each node uses this function to calculate it's output: $f(x) = \frac{1}{1+e^{-x}}$, where x is the sum of the weighted inputs.

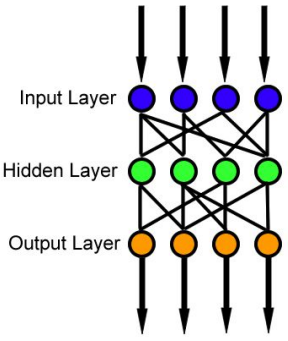

Figure 2: A prototypical neural network, showing input, hidden, output layers, and connections. [2]

Initially, given the size, number of features, and low covariance; it was assumed that a somewhat larger network would be needed. Several network configurations, loss functions, and optimizers were used; including a network 10 layers deep with other a million connections, 3 different loss function/ optimizer pairs, and several hours of training. Eventually the same result continued coming up: an accuracy of less than 0.6 and a model that would classify Standing, Sitting, and Laying together and also the three Walking classes together. In a last ditch effort to improve the model, layers began shrinking and being taken out. Continuing to do this did not reduce accuracy or change the classification, so it continued until the final network was created: a single hidden layer of 100 nodes. This network classified the data almost perfectly and also trained significantly faster, due to it having only ~50,000 connections compared to the >1M of some of the larger networks.

Gradient Boosted Decision Tree:

The second model made was a Gradient Boosted Decision Tree, implemented using Sci-Kit Learn [11]. This is a kind of ensemble method that involves creating an additive model of multiple decision trees based on the residuals of earlier trees. What this means is that the algorithm creates the first decision tree, finds the errors, and trains another decision tree to try and predict those errors. It does this repeatedly, each time adding another tree to the model and taking small steps towards a proper model.

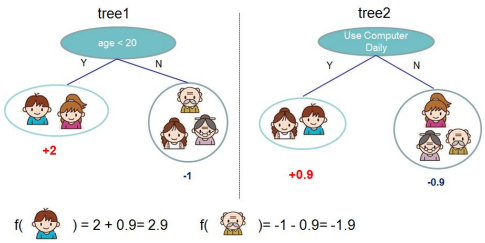

Figure 3: Example of gradient boosted tree evaluation [13]

This model, by design, is able to explain how it achieved it's result, including by reporting feature importances, or how influential each feature was to the final model. This was then used to construct reduced models to determine how many features were needed to achieve a given accuracy. Generally, it showed that the most important feature was significantly more important than others, and then the importance would decline until the 5th feature where it would plateau until the 13th feature where it would plateau again. Using this data, several more models were generated using n many of the most important features from n=1 to n=15. The resulting accuracies were plotted with matplotlib [14] against the number of features used. Lines were added to show the 80% and 90% accuracy lines, as well as the number of features that first achieves these accuracies. Typically the model to first reach 80% had 3 or 4 features, and the first to reach 90% would have 6 features.

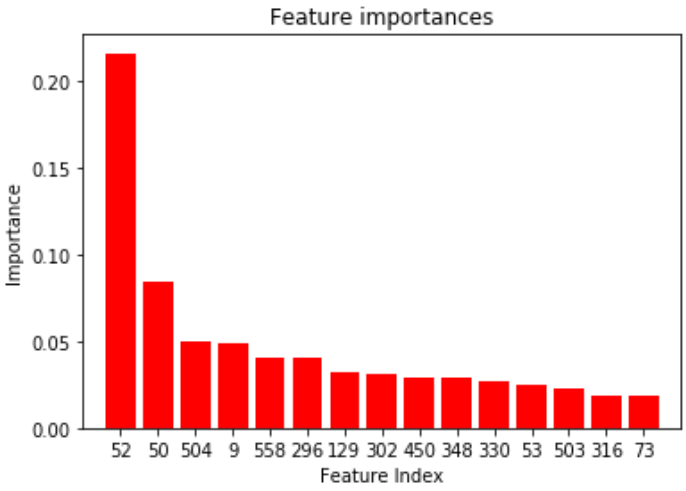

Figure 4: Feature importances returned from Gradient Boosted Decision Tree model.

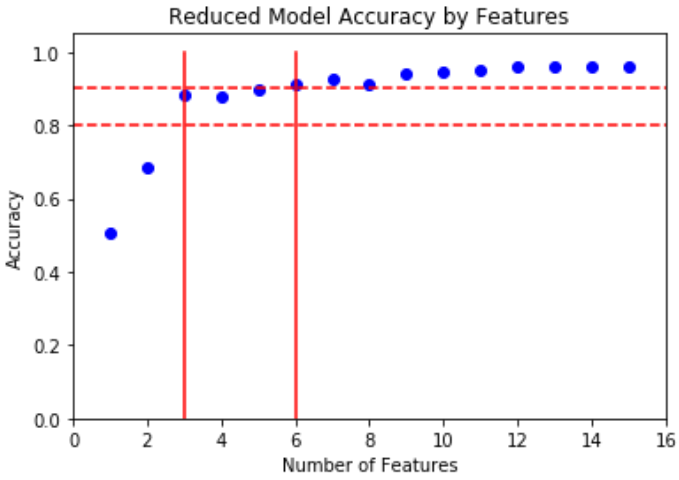

Figure 5: Accuracy against number of features used in reduced model.

Results:

Both algorithms were able to create very strong models, with the Neural Network achieving a testing accuracy of .98 (training accuracy of >.999).

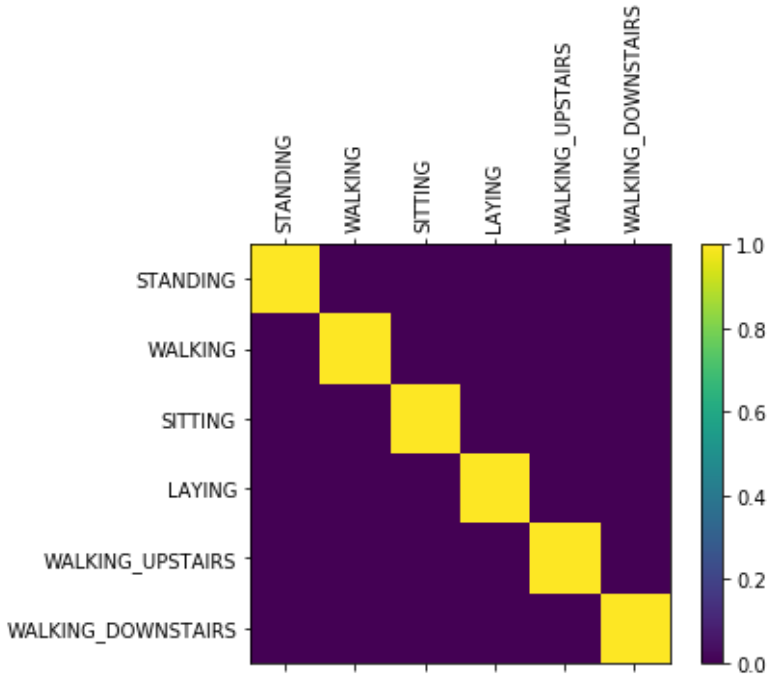

Figure 6: Classification Matrix for Neural Network

This accuracy is so high that the color based classification matrix doesn't even show any errors, and the average number of wrong classifications was 22 out of 2060 predictions.

The Gradient Boosted Decision Trees had a similar level of success, with the full model reaching an accuracy of >.99 regularly, and the reduced model (10 features) achieving ~.95 accuracy.

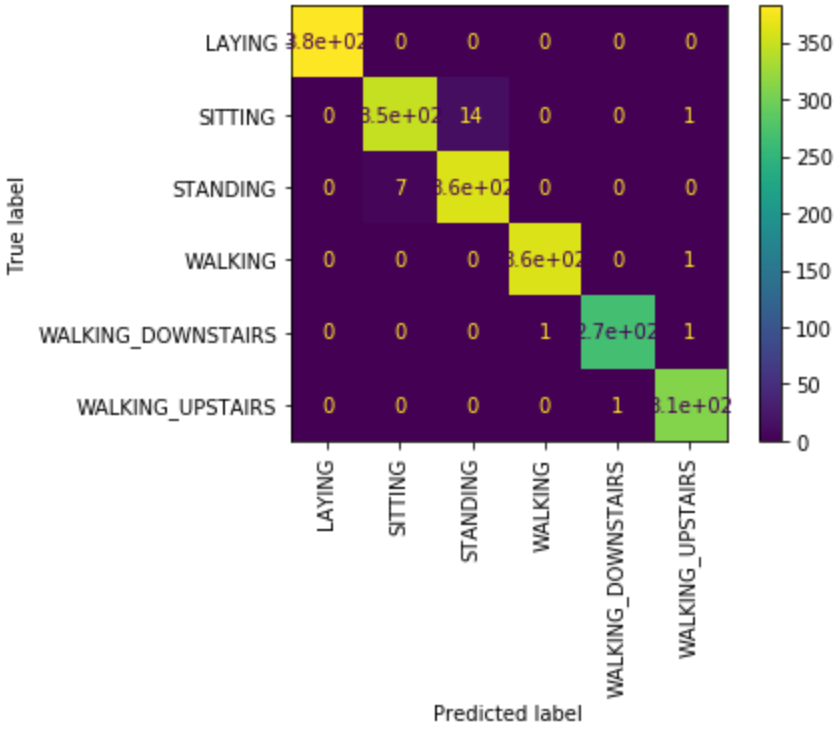

Figure 7: Classification Matrix for Gradient Boosted Decision Tree

From the classification matrix it is possible to see and count exactly the number of errors (only about 26), and that most of them are mixing up standing and sitting. Furthermore, for the full model, a ROC curve was created [15]. On a ROC curve, the true positive rate ($\frac{True\ Positive}{Total\ Positive}$) is graphed against the false positive rate ($\frac{False\ Positive}{Total}$) and the larger the area under the curve is, the stronger the model. Due to the nature of ROC curves, the Sci-Kit Learn implementation can only be created for binary classification (yes or no on a single class), so a curve was created for each of the 6 activity labels and placed on the same graph. Again, due to the strength of the model it is almost impossible to see the curves at all in the top left corner.

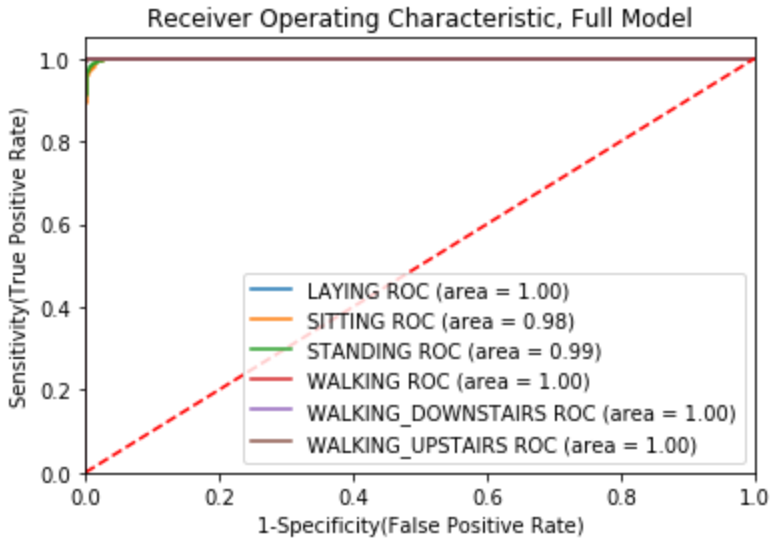

Figure 8: ROC Curve for Full Gradient Boosted Decision Tree Model

References

1. Wikipedia "Gradient Boosting" Page. URL: https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting
2. Wikipedia "Feed Forward Neural Network" Page. URL: https://en.wikipedia.org/wiki/Feedforward_neural_network
3. University of Rochester, Department of Data Science. URL: http://www.sas.rochester.edu/dsc/index.html
    a. University of Rochester, Department of Data Science: Capstone Project. URL: http://www.sas.rochester.edu/dsc/undergraduate/capstone.html
4. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. "Human Activity Recognition Using Smartphones Data Set." URL: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
5. R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/.
6. Python Core Team (2019). Python: A dynamic, open source programming language. Python Software Foundation. URL https://www.python.org/.
7. Sci-Kit Learn. "Train Test Split." URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

8. Wikipedia “Covariance” Page. URL: https://en.wikipedia.org/wiki/Covariance
9. Wickham H (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.
10. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” URL: https://pytorch.org/
11. SKLearn. “Gradient Boosting Classifier” URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
12. University of Rochester, CIRC. Bluehive URL: https://www.circ.rochester.edu/resources.html
13. XGBoost, Introduction to Gradient Boosted Trees. URL: https://xgboost.readthedocs.io/en/latest/tutorials/model.html
14. Matplotlib, “Python 2D plotting library which produces publication quality figures.” URL: https://matplotlib.org/3.1.1/index.html
15. Wikipedia “Receiver operating characteristic” Page. URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic