CS 74 Project Milestone: Predicting Stress from Android Sensor Data

Nishant Kumar, Minsoo Kim

Feb 17, 2015

I. Project Goal

Stress is an indicator for a wide variety of both physical and mental health disorders. Although stress is relatively easily self assessed and addressed, it often goes unnoticed as a natural part of everyday life. Unaddressed in this way, especially over a prolonged period of time, stress may lead to more serious mental health disorders such as depression or increase the likelihood of physical disorders such as heart attack.

An automatic stress level prediction system that can infer a user's level of stress from sensory input will be helpful in addressing the issues caused by stress. Because sensory data is gathered continuously, such a system can monitor a user's stress level with little interruption and alert/remind the user of his or her current stress level or make recommendations based on it.

We propose an application that can predict a user's stress levels by analyzing the raw sensor data from his/her android device. This application then in theory can sit on a centralized web server or on the user's Android device itself.

II.Scope

The current scope of the project is to demonstrate that a machine learning algorithm provided enough raw sensor data from a smartphone can learn a model to predict a user's stress levels.

III. Data

The data used to learn this model is from the Dartmouth StudentLife study. This is available on the CS department's website. This data was collected from student volunteers from the CS-65 Smartphone programming class offered last year. Student volunteers were given android phones which had an in built app which recorded raw sensor data such as:

Audio inputs
Activity inferences
Conversation logs
GPS location
Bluetooth
GPS location
WiFi
WiFi location,
Light sensor

Phone lock
Phone Charge
Education data - Deadlines etc.

Data collection monitoring
Students were sent reminder emails if there were any gaps in the data collection or the data was not getting uploaded at all.

Incentive
The student volunteers were offered incentives such as free T-Shirts, Google Nexus smartphones etc.

Privacy
The student volunteers' identities were hidden by using random ids. The call logs and sms logs were hashed.

Survey Data
The dataset also contains self reported stress level data ranging between 1 and 5. The original mapping used in the dataset is as follows:

Stress Level Id	**Description**
1	A little stressed
2	Definitely stressed
3	Stressed out
4	Feeling good
5	Feeling great

We have used a new mapping in order to show a monotonically increasing values of stress. Its shows as follows:

Stress Level Id	**Description**
1	Feeling great
2	Feeling good

3	A little stressed
4	Definitely stressed
5	Stressed out

The data collected is stored in csv and JSON formats.

IV.Data Processing

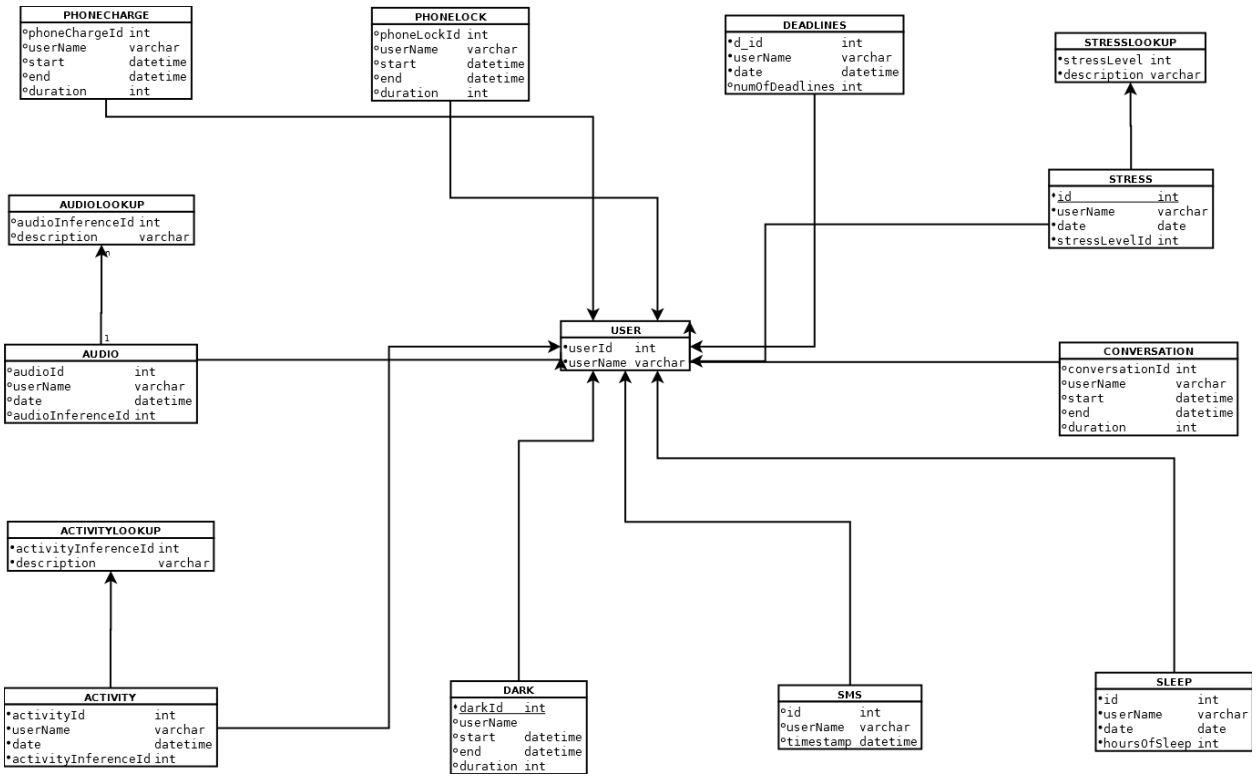
The project requires certain amount of Software Engineering effort. Its described as follows:

a. We have written several parsers in Java to read the data that is in csv and JSON formats.

b. The respective parsers then further pass on the data to data adaptors that store all this sensor data in a database. The data adaptors are implemented in Java and Hibernate which provide the Data Access Object(DAO) layer over the database.

c. The database is implemented in MySQL database server.

The database schema diagram is as follows:



Every individual sensor data is stored in its own table. The tables further should have a foreign key relationship with the USER table, indicating that they cannot store data about any user that is not present in the USER table.

This database is then queried to generate input for the Neural Network to train on. For the milestone, the sensor data we used for the features is as follows:

Conversation logs
Light data
Phone lock
Phone Charge
Sleep

Stress data acts as our y - label.

If the sensor data is recorded multiple times in a given day for a given user, we add up all the data. Eg: the conversation data for a given day and given user is added up to indicate the number of hours a user has conversation logs on that day.

The stress data on the other hand for a given day and user is averaged up.

V. Neural network

1. **Motivation**

Models of regression and classification that involve linear combinations of fixed basis functions are useful analytical tools, but they often run into trouble when the model we desire to approximate has high dimensionality. The relationship between human behavior and phenomena such as stress is most likely high dimensional and nonlinear. Therefore we use neural networks to approximate a model of the relationship between behavior and stress.

2. **Training**

i) Gradient descent

To train a neural network, we wish to find a weight matrix w which minimizes the error function $E(w)$. Since we cannot find an analytical solution to $dE(w)=0$, we use gradient descent to optimize the weight matrix. We choose the weight update to be a small step in the direction of the negative gradient. The gradient descent update rule is given by:

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E(\mathrm{w}^{(\tau)})$$

where the parameter η is the learning rate. The above rule applies to the *batch gradient descent* method, which evaluates the gradient based on the entire training set before updating the weight matrices.

An online version of gradient descent, whose weight update rule is given by:

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla \mathrm{E}_n(\mathrm{w}^{(\tau)})$$

is known as stochastic gradient descent, and updates the weight matrix based on each training example. Stochastic gradient descent simplifies the gradient by computing one for each training example, but has been demonstrated to perform on par with batch methods, while taking less running time.

ii) Backpropagation

In order to evaluate the gradient of $E(\mathrm{w})$ for a feed-forward network, we use error backpropagation. The backpropagation algorithm is derived by applying the chain rule for partial derivatives and obtaining the derivatives of the error function with respect to the the neural network weights.

We omit the full derivation of the backpropagation algorithm, which was referenced from Section 5.3.1 of Bishop.

3. **Algorithm**

Based on the algorithm provided by Bishop, we build a neural network with the following functions.

First, the activation function for the hidden units are given by

$$\mathrm{h(a)} = \tanh(\mathrm{a})$$

where

$$\tanh(\mathrm{a}) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

The hyperbolic tangent function has its derivative given by

$$\mathrm{h}'(\mathrm{a}) = 1 - \mathrm{h(a)}^2$$

We use a euclidean error function given by

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2$$

where y_k is the activation output unit k and t_k is the corresponding target value.

The forward propagation is performed by:

$$a_j = \sum_{i=0}^{D} {w_{ji}}^{(1)} x_i$$

$$z_j = \tanh(a_j)$$

$$y_k = \sum_{j=0}^{M} {w_{kj}}^{(2)} z_j$$

Finally, to obtain the gradients, we first compute

$$\delta_k = y_k - t_k$$

and

$$\delta_j = (1 - {z_j}^2)\sum_{k=1}^{K} w_{kj}\,\delta_k$$

and compute

$$\frac{\partial E_n}{\partial {w_{ji}}^{(1)}} = \delta_j x_i \qquad \frac{\partial E_n}{\partial {w_{kj}}^{(2)}} = \delta_k z_j$$

VI. Challenges

1. **Random initialization**

We encountered a symmetry problem in our initial implementation of the neural network. Initializing the parameters to random positive values between 0 and 1, while an intuitive choice, causes a problem known as symmetry, in that all activation values of the hidden layer take on similar values. In this case, the neural network will be unable to learn. Since we used the hyperbolic tangent function, an initialization of all parameters to positive values has a high likelihood of causing all of the hidden

unit layer activations to take on the value of 1. This means that the error signals propagated to these units will take the same value, which means that the weight updates for the hidden units will be identical. This causes the neural network to "get stuck".

We resolved the issue by initializing the parameters to random values within a range of -0.12 to 0.12. The specific values were reached via trial and error of best performance, and recommendations from literature.

2. Data Processing

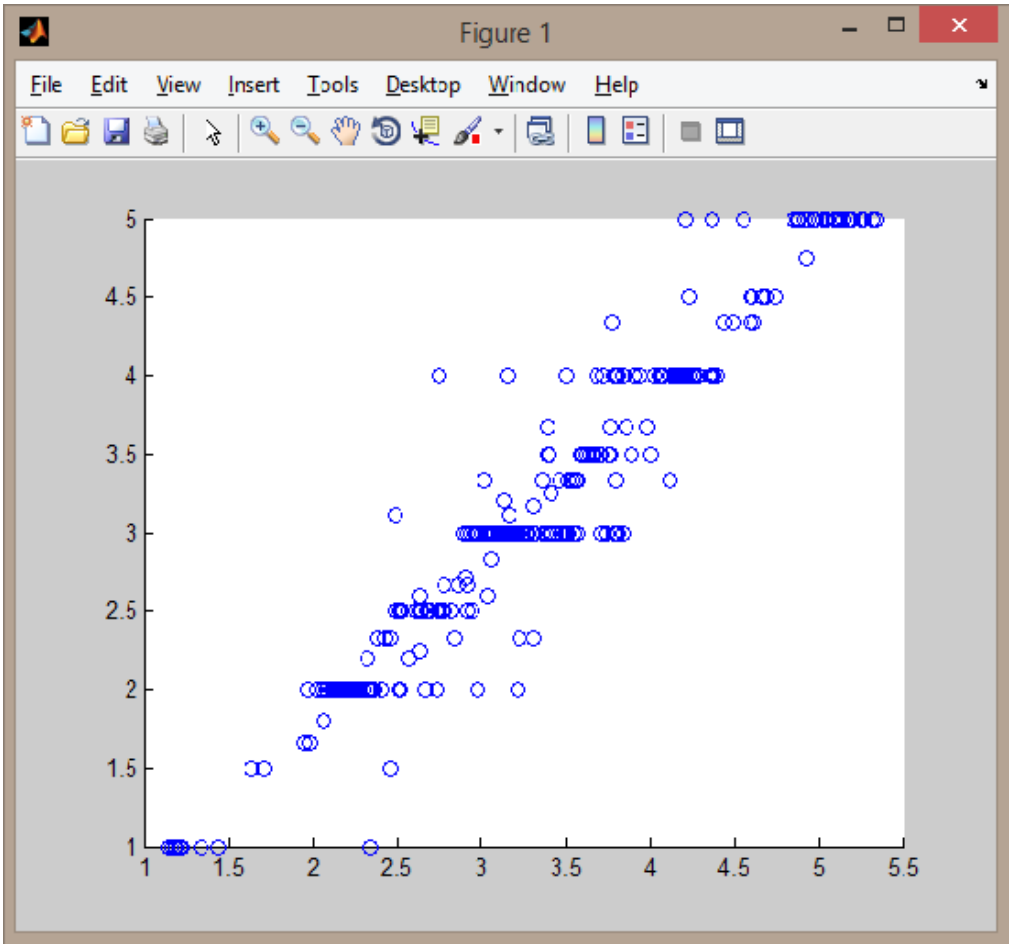
We wrote our own Java classes and databases in order to make the whole project self-contained, however we found the using STATA from some data processing operations saved much time. We plan to continue using both in conjunction with one another.

VII. Results

Our first set of results are from training on a subset of 330 training examples. These examples are produced by removing all training examples which did not have the full set of feature data, meaning there was at least one sensor data from the user that was unable to be collected for the corresponding time period, for the user. Our results indicate that with this descriptive subset of examples, we are able to accurately fit the data, despite the fact that we are only using the a subset of the entire feature data, namely the five features phone-lock, phone-charge, sleep, dark.

Figure 1 plots the predicted Y and actual Y. The scatterplot shows a slope of 1, which reflects the desired relationship between the actual Y values and those predicted by the neural network after training. (Actual Y=Pred Y). The training hyperparameters are: 330 training examples, 200 hidden units, 0.01 learning rate, and 500,000 iterations. After 500,000 iterations, the MSE is roughly 0.1.

Figure 1: Scatterplot of Predicted Y vs Actual Y



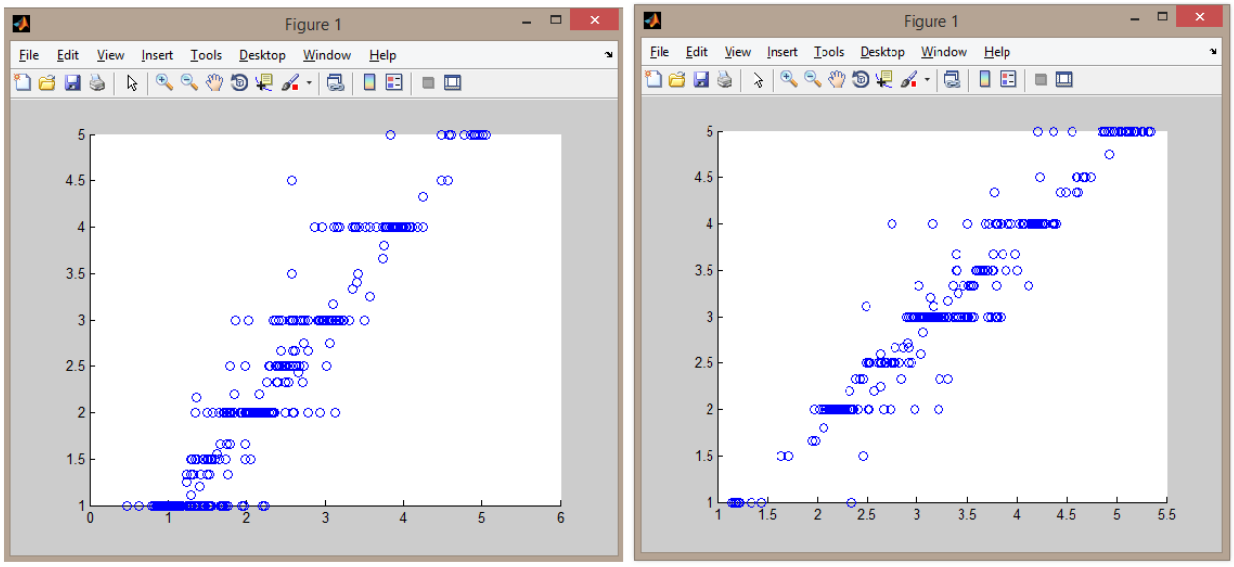
Next, we show a series of results which were used to determine the optimal learning rates and number of hidden units.

1) **Learning rate and Hidden unit choices**

i) Learning rate

Figure 2 shows the results of two training sessions, obtained by training the neural network with the same hyperparameters, the only difference being the learning rate. The left scatterplot has been trained with a learning rate of 0.001, while the right is identical to Figure 1 above, with a learning rate of 0.01.

Figure 2: Comparison of learning rates 0.001 and 0.01



We find little difference between the learning rates, and since the smaller learning rate of 0.01 produces almost identical results while taking roughly 1/10 of the time, we opt to use 0.01 as our learning rate.

We also tested the learning rate of 0.1, but found it to be too unreliable, with training failures occurring too often.

ii) Minimum required hidden units

We test the optimal number of hidden units. We begin with a small number(20). We found that for 20 hidden units and 0.1 learning rate, the MSE reached 0.4 at 150,000 iterations but for the next 150,000 iterations, remained at 0.4. We terminated the training at this point.

Next we ran 20 hidden units with 0.01 learning rate. The MSE reached 0.51 at 600,000 iterations but remained there for the next 200,000 iterations.

Therefore, we concluded that 20 hidden units were not enough to fully model our data.

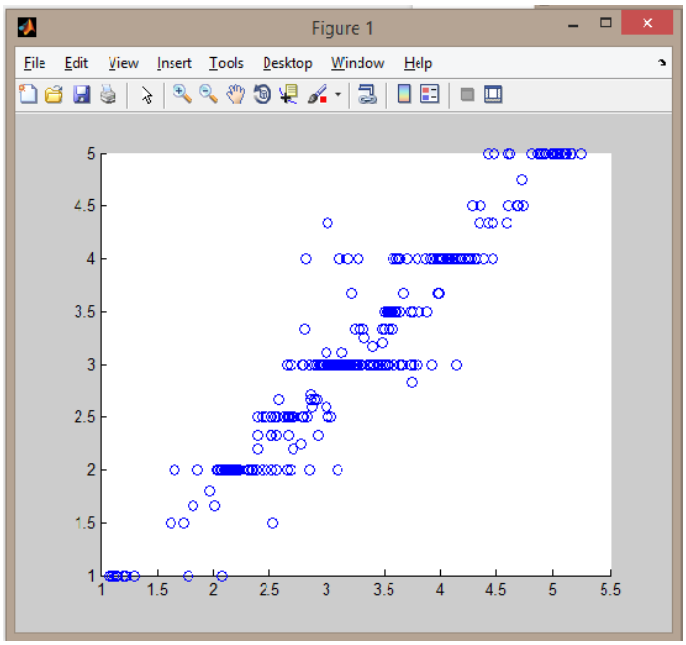
iii) Optimal hidden units

Next we try values 100, 200, and 300 for our hidden units.

We increased the number of hidden units to 100, with a 0.01 learning rate.

At 7 minutes of training time, the running time was much faster than the 30 minutes of 200 hidden units with the same learning rate.

Figure 3: 100 hidden units vs 200 hidden units



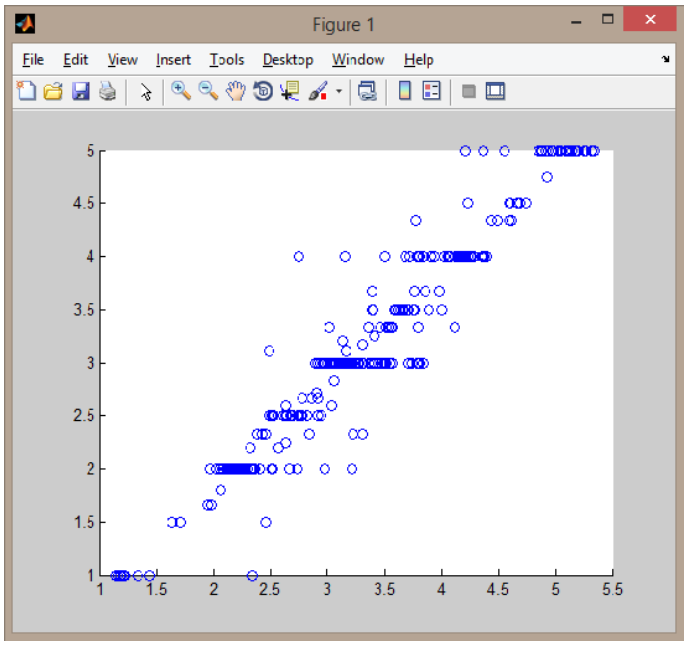
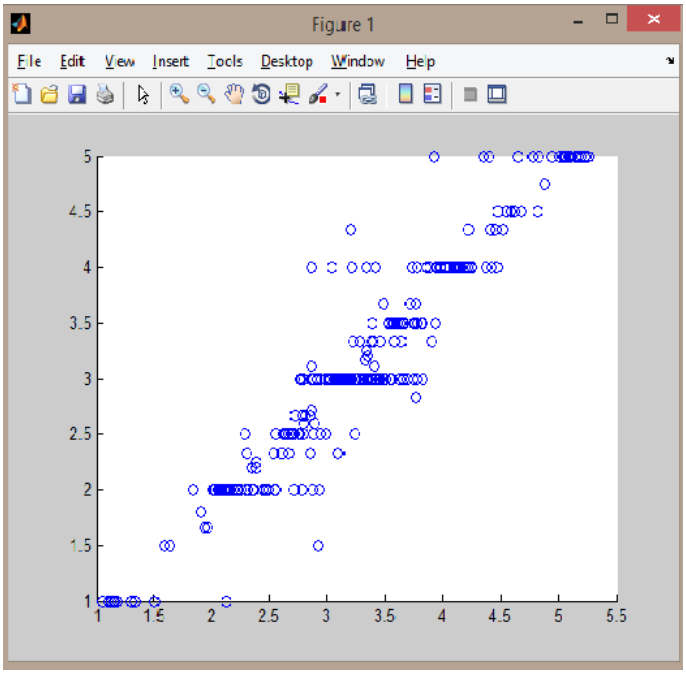
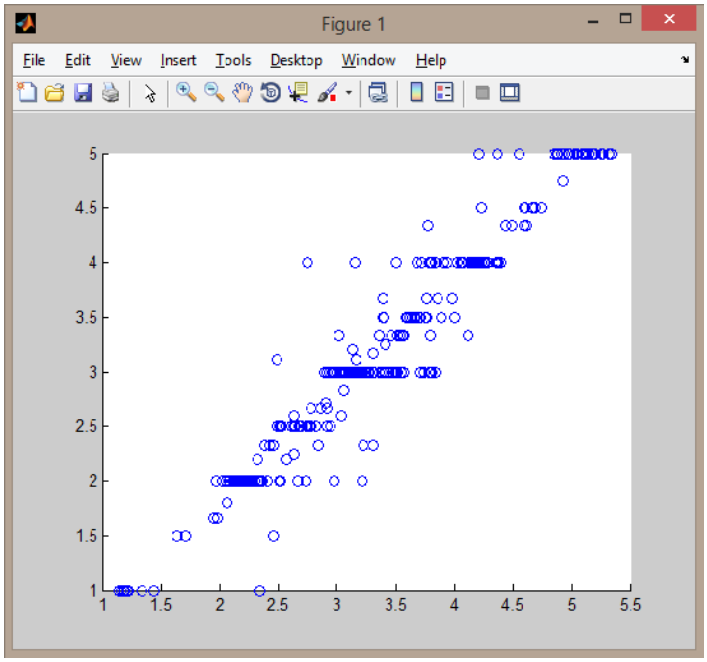


Figure 4: 300 hidden units vs 200 hidden units





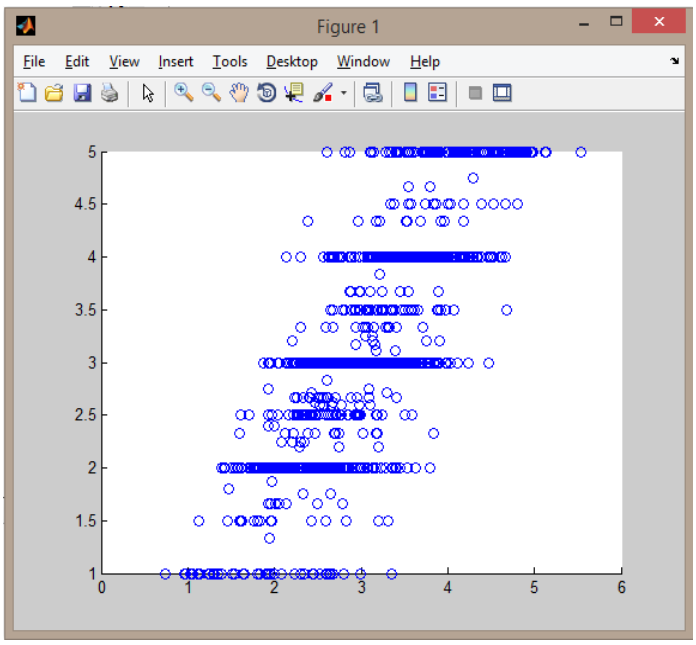
The training took 15 minutes for 300 hidden units, longer than 200 hidden units, but differences were minimal.

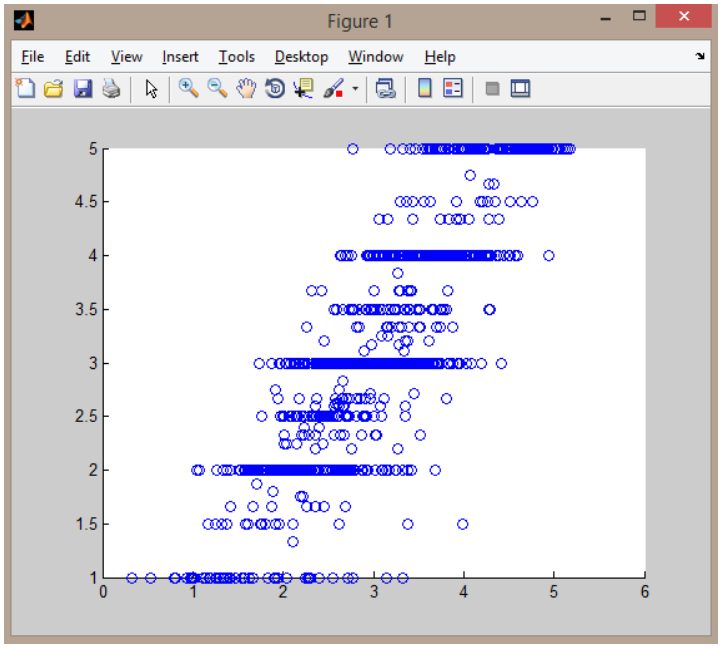
In conclusions, out of all hyperparameter choices we tried, we found the choice of 0.01 to be a good choice for the learning rate considering the accuracy and the training time, and found any number between 100 and 300 to be good choices for the hidden units.

2) Full dataset: A preview

Finally, after completing our tests on the smaller subset, we tested our neural network on the full dataset of 1270 examples (selected from the 3200 examples and removing examples without corresponding reported stress values). In the interest of time, we terminated the training at an MSE of 0.4 and 0.3. As figure 5 shows, the results are not as accurate as that of the smaller subset, but the scatterplot takes the general desired shape. Since error was continuing to decrease when we terminated the training, we believe that we can continue training up to an MSE of 0.2 without any issues. This and further optimization will be our goal going forward.

Figure 5: Full dataset (1270 rows), termination with MSE=0.4 vs MSE=0.3





VI. Discussion

For the milestone goal, we began by training a neural network on a subset of the entire data set. We produced this data by removing all training examples which did not have the full set of feature data. In effect, we chose the most descriptive examples in our data set and trained a model on them. Our results showed that with this descriptive subset of examples, we were able to reliably fit the training data, despite the fact that we used only a subset of the entire feature data, namely the five features phone-lock, phone-charge, sleep, dark. Further, we performed a preliminary training of the larger dataset of 1270 examples, which included missing sensor values, and found that the results showed a reasonable fit.

Given that the number of features are not many, it is in some ways surprising to see that we were able to find reliable fits of the data, and it is not easy to see what the precise functional relationships between these 5 variables would be, such that with only 5 variables, one can predict the stress level of a user.

Our best guess, from analyzing the dataset itself, is that there may actually be numerous relationships that can be found within this dataset of only 5 features. For example, from looking through the dataset, we can observe things such as the following. The "dark" feature, although it shows no clear overall relationship with the stress variable, seems to have a very clear relationship with medium to low stress, in that when the "dark" value is less than 5, the stress value is almost always lower than 3. We may hypothesize, for example, that such a value may indicate that the user is spent a highly socially active day relative to the avg user, wherein high social activity has been shown to be indicative of either a low stress state, or to actually cause a reduction in stress. Similarly, we can readily observe things such as the fact that there is only a handful of users who slept less than 7 hours and reported a stress value less than or equal to 2, whereas increasing the sleep value by one hour, to 7 hours, suddenly gives us dozens of users who reported a stress value less than or equal to 2.

We surmise that in fact, human behavior, as far as stress is concerned, may actually be much more predictable than our intuition might tell us. One reason such a finding may have eluded us is that until now, we did not have adequate tools to describe numerically a person's behavior to a high degree of fidelity. As far as we are aware, the Dartmouth StudentLife study is one of the first studies ever to measure human behavior in such a continuous and detailed fashion, and this has only been possible because of the advancement of smartphone sensor technology.

Our preliminary findings give us confidence that in light of these new sources of data about human behavior, it will be a worthwhile goal to employ machine learning techniques to analyzing these data and perhaps produce results which may not have been feasible before. For our final report, we hope to produce further results to validate our theories.

VIII. Future Goals

1. We plan to explore more options for neural network optimization, and cross validate our results.

2. We plan to add more features such as Activity inferences and Audio inferences.

3. We will expand the already existing features as necessary.

IX. References:

1. Discussions with Professor Lorenzo Torresani.

2. *Wang, Rui, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. "StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones." In Proceedings of the ACM Conference on Ubiquitous Computing. 2014.*

3. *S. E. Taylor, W. T. Welch, H. S. Kim, and D. K. Sherman. Cultural differences in the impact of social support on psychological and biological stress responses. Psychological Science, 18(9):831–837, 2007*

4. *N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In Proc. of PervasiveHealth , 2011*

5. *CS65 Smartphone Programming. http://www.cs.dartmouth.edu/~campbell/cs65/cs65.html*

6. *Depression. http://www.nimh.nih.gov/health/topics/depression/index.shtml*

7. *StudentLife Dataset 2014. http://studentlife.cs.dartmouth.edu/*

8. *C. M. Aldwin.Stress, coping, and development: An integrative perspective. Guilford Press, 2007*

9. *Eftekhar B, Mohammad K, Ardebili HE, Ghodsi M, Ketabchi E. Comparison of artificial neural network and logistic regression models for prediction of mortality in head trauma based on initial clinical data. BMC Medical Informatics and Decision Making 2005;5:3. doi:10.1186/1472-6947-5-3.*

10. *Bishop, Christopher M. Pattern Recognition and Machine Learning*

11. *Ng, Andrew. Machine Learning Course Materials (Stanford, Coursera)*

12. *http://www.mysql.com/*