Deadline Prediction using Ordinal Regression

Joshua Cook, Byoungwook Jang, Aditya Mahara

February 17, 2015

1 Background

StudentLife was a study conducted by Dartmouth College's computer science department that collected passive and automatic sensing data over a 10 week period [3]. The goal of this study was to assess students' mental health based on their behavior. The students' behaviors were determined by processing the collected data using Machine Learning algorithms. However, for this data set to be useful , additional studies need to be conducted that attempt to predict metrics that can be used by the school's leadership and faculty to make real-time adjustments throughout the term.

2 Scope and Goal

The goal of our study is to implement the artificial neural network that can accurately predict the number of deadlines from the set 0, 1, 2, or 3 based on the student's behavior on the previous day. This way, no compliance is required from students other than simply downloading an application. We first treated this as a classification problem proposing Naive Bayes and SVM methods. However, since the ordering of classes has a relationship to one another, we reformulated our objective as an ordinal regression problem.

3 DataSet

The dataset is collected from 48 undergraduate and graduate students at Dartmouth over the 10 week spring term (March 27, 2013 to June 5, 2013). Within the StudentLife sensor datasets there are ten different data fields such as physical activity, audio inferences, conversation inferences, bluetooth scan, light sensor, GPS, phone charge, phone lock, WiFi, and WiFi location [3]. All sensor data were available as csv files and were organized by participants. First, we imported these datasets in a meaningful way into MATLAB. The timestamps in the raw datasets were in Unix time stamp format so the time information we obtained had a resolution of 1 second. To process the information associated with these timestamps we wrote a code to convert the Unix timestamp into month-day-year within a period from March 27, 2013 to June 5, 2013 (i.e 71 days). We also wrote codes to extract example sets and feature sets using these datasets which is described in detail below.

To test our algorithm, an initial set of training set of 4000 examples, testing set of 1696 examples with 170 features each, is used. Some information on the examples and features

along with the data processing necessary to create these example sets and feature vectors are presented below:

4 Examples and Features

To create an example set we use information about deadlines per day for each student. The StudentLife dataset has information from 44 students for 71 days with deadline information. Since our algorithm is trying to predict the deadline for the next day, we will be able to use information from 44 students for 70 days as examples. Therefore initially we have a total example set of 3080 (44days $*$ 70 students). We refer to this set as 'Dataset I.' Next we scanned over the columns of Dataset I to check for any missing data points. All the examples with any missing feature were excluded from our algorithm analysis. This reduced the number of examples to 1886. This new set without any missing feature information is referred to as 'Dataset II.'

4.1 Unequal label classes

The frequency of different deadline labels (0,1,2,3) is not equal throughout Dataset I and Dataset II. The total occurrence of deadline labels (0,1,2,3) in Dataset I was 2438, 496,135,11; and in Dataset II was 1446, 345,89,6, respectively. Since distribution of examples with specific labels wasn't equally distributed we replicated the datasets with lower number of label examples. Using Dataset II we replicated examples for label 1,2,3 by a factor of 4, 16, and 241.

In addition to replicating examples, next we will explore modifying learning objective per label to create a better classification for labels with low occurrences.

4.2 Features

Features were extracted to represent daily activity using sensor information through the duration of the study for those specific 44 students for whom we have deadline information available. For our algorithm analysis we have used three out the ten feature sets available and we plan to explore more in the coming weeks before our final submission.

To construct a feature vector, we used a simplistic way to capture information about frequency of occurrence of a certain classifier per sensor. Brief descriptions on what these features represent and how we extracted them are given below:

Audio
The raw data file for audio has two columns. First column has timestamp information and the second column was information on audio inference where audio inference is classified as 0, 1, 2, or 3 that represents silence, voice, noise, or unknown respectively. The audio classifier runs 24/7 with duty cycling. It makes audio inferences for 1 minute, then pauses for 3 minutes before restart. If the conversation classifier detects that there is a conversation going on, it will keep running until the conversation is finished. It generates one audio inference

every 2 to 3 seconds [3].

Physical Activity
The raw data file for physical activity has two columns. First column has timestamp information and the second column was information on activity inference where activity inference is classified as 0, 1, 2, or 3 that represents stationary, walking, running, or unknown, respectively. The activity classifier runs 24/7 with duty cycling. To avoid draining the battery, it makes activity inferences continuously for 1 minutes, then pauses for 3 minutes before restart collecting activity inferences again. It generates one activity inference every 2 to 3 seconds depending on Smartphone's accelerometer sampling rate [3].

Conversation
The raw data file for conversation has two columns. First column represents a timestamp where a conversation began and the second column is the timestamp when the conversation ends.

4.3 Feature Implementation

In Fig. 1 we have a histogram representation of a feature vector for an example feature set (i.e. Audio). Using 'Audio' feature we compute the frequency of occurrence for silence, voice, and noise, for every hour per student. The histogram represents an example of a feature vector for 1 day for 1 student as we can see which parts of the day he/she was mostly silent and which parts of the days were mostly in noisy environments or where he/she was talking. Using this technique, we extracted 72 features (24 hours x 3 labels) for audio data.

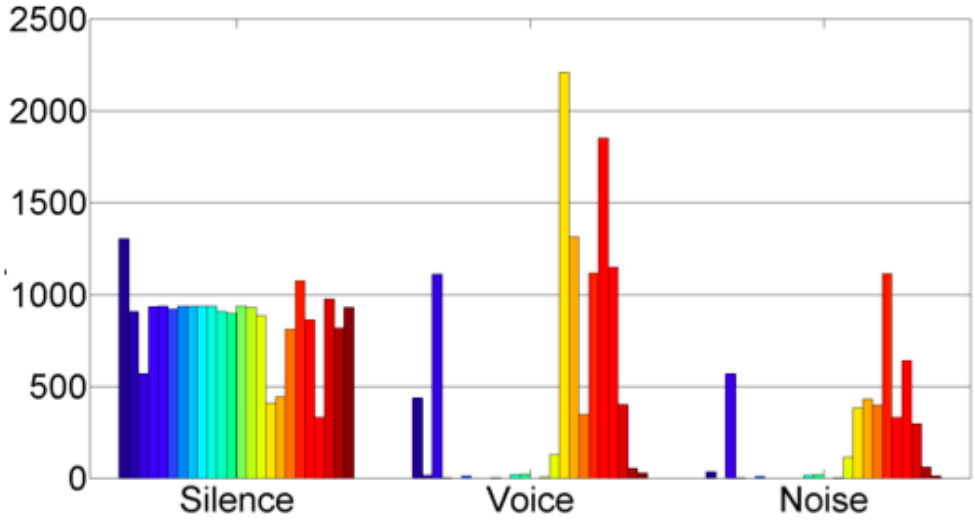

Figure 1: Feature vector profile for Audio Data

Similar techniques are used to extract 96 features (24 hours x 4labels) for physical activity and 6 features for conversation. Next steps during feature extraction will involve extracting information not based solely on frequency of occurrence, but using more elaborate information. Some of these features to be explored will be parameters such as duration of time

between events, distance travelled per unit of time by a student, hours spent in the library, usage of the gym, and so on. All these information can be extracted from the StudentLife dataset that's available to us.

5 Implementation

Jianlin Cheng's paper, *A Neural Network Approach to Ordinal Regression*, implements the artificial neural network (ANN) to perform the ordinal regression task [1]. In order to implement the neural network, the algorithm modularized into two major parts: 1) forward propagation and backpropagation, and 2) batch gradient descent. The detailed implementation tutorial can be found from Andrew Ng's coursera course [2].

5.1 Notations

The following notations are going to be used in the cost functions.

$$(x^{(i)}, y^{(i)}) = \text{i-th training example} \tag{1}$$
$$L = \text{total number of layers in the neural network} \tag{2}$$
$$s_l = \text{number of nodes in layer l} \tag{3}$$
$$a_i^{(l)} = \text{activation of unit } i \text{ in layer } l \tag{4}$$
$$\theta_{ij}^{(l)} = \text{matrix of weights from j-th node in layer } l \text{ to i-th node in layer } l+1 \tag{5}$$

As mentioned in class, the cost function of the logistic regression is as follows

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta^2 \tag{6}$$

As we are using the logistic function for each activation node, we can sum rewrite the cost function for the neural network as follows

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} (\log h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k)] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_l+1} (\theta_{ji}^{(l)})^2 \tag{7}$$

5.2 Algorithm

The steps for the neural network algorithm is as follows:

Given the training set $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\}$

- Initialize the θ matrix
- for $i = 1$ to m

  Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \cdots, L$

- Perform back propagation to compute the gradient of $J(\theta)$

With the gradient, we performed the bath gradient descent until we reached the stopping criteria. The stopping criteria is given as follows. Given δ_1, δ_2, and $\delta_3 > 0$, we need to satisfy the following three criteria to stop the gradient descent

$$||\theta - \theta_{new}|| < \delta_1 \tag{8}$$
$$||J(\theta) - J(\theta_{new})|| < \delta_2 \tag{9}$$
$$||\nabla J(\theta)|| < \delta_3 \tag{10}$$

6 Results

Using the Artifical Neural Network modified for the Ordinal Neural Network(ONN) case, firstly we use 100 training and testing examples with equal number of all deadline labels examples with a architecture of 1 layer and 10 nodes. Next we used 4000 training examples and 1696 testing examples to test two architecture: i. 1 Layer 10 nodes, ii. 2 Layers with 10 nodes each. For each arrangement we plot the error per example as a function of the lambda value we used. In all cases the training error was less than the testing error.

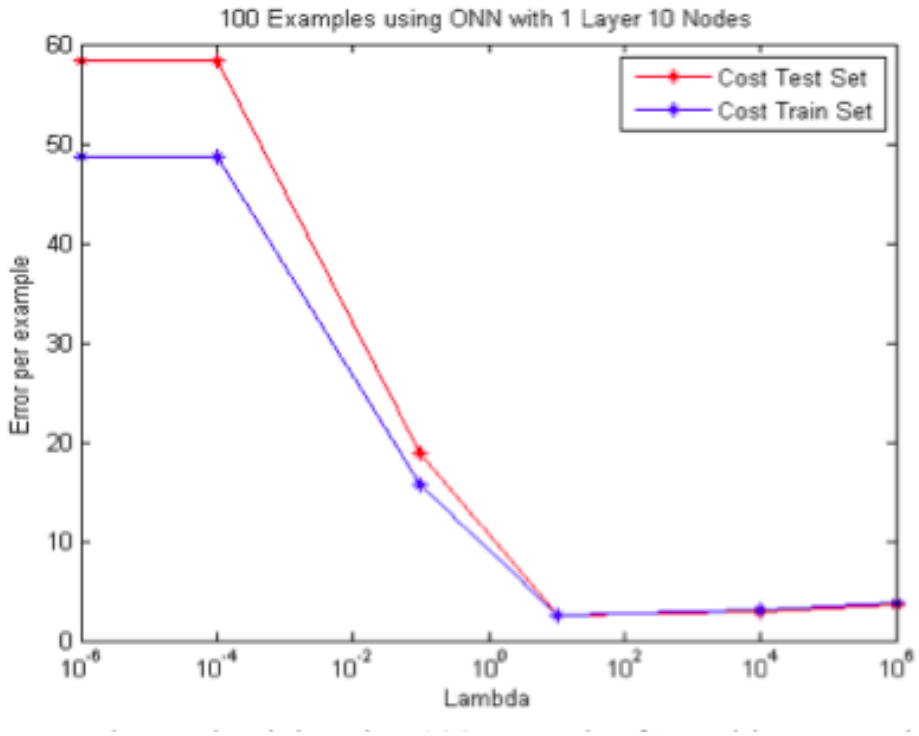

Figure 2: Error per examples vs. lambda using 100 examples for architecture using 1 layer and 10 nodes

As seen in Fig. 2. using 100 training and testing examples with architecture of 1 layer and 10 nodes we see that with increasing lambda, the error per example goes down. It seems like to lower values of lambda (10^{-2} to 10) there is over fitting. Also, as seen in Fig. 3. when we use all examples and use the identical architecture, the error per example goes down;

however the error seems to be scaled down. This happens since we use many more examples the average error per examples. In both cases we see over fitting for lower values of lambda. In both cases we do not see issues with under fitting.

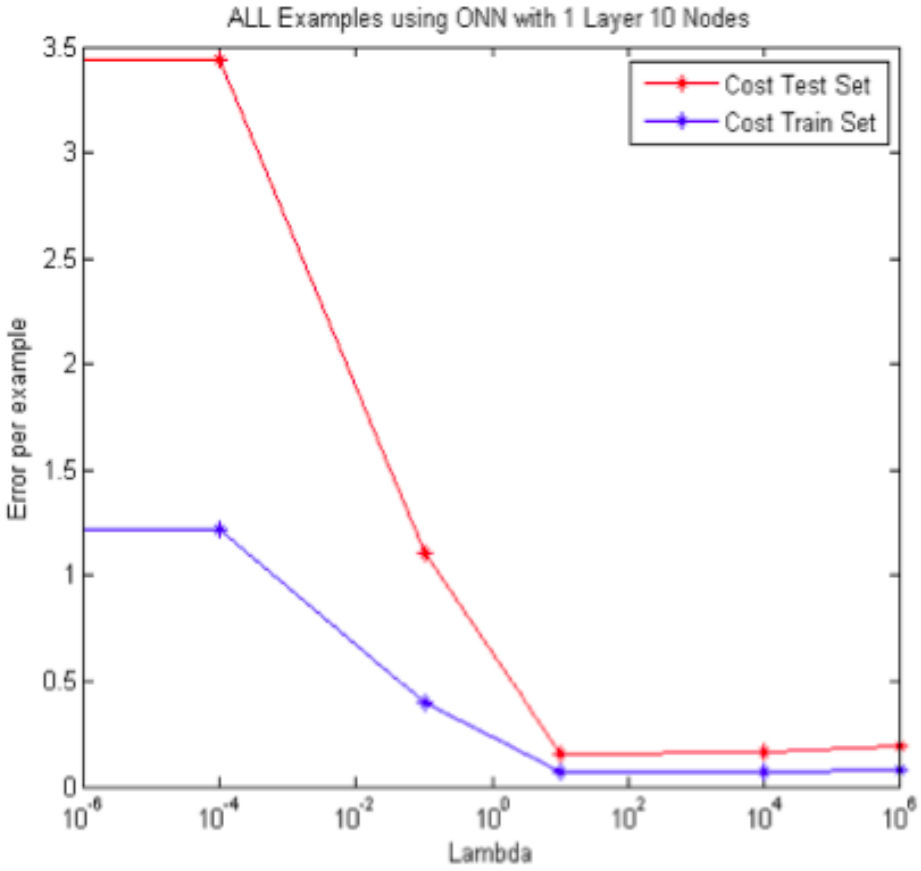

Figure 3: Error per examples vs. lambda using all examples for architecture using 1 layer and 10 nodes

When we use architecture with 2 layers with 10 nodes each, we get a error per example plot as shown in Fig.4. This doesn't make a lot of sense to us since there seems to be one value of lambda for which the error is maximized and there seems to be no issues caused by over fitting and underfitting. Further analysis for architectures with additional layers and nodes will need to be conducted before we can conclude anything from these preliminary results.

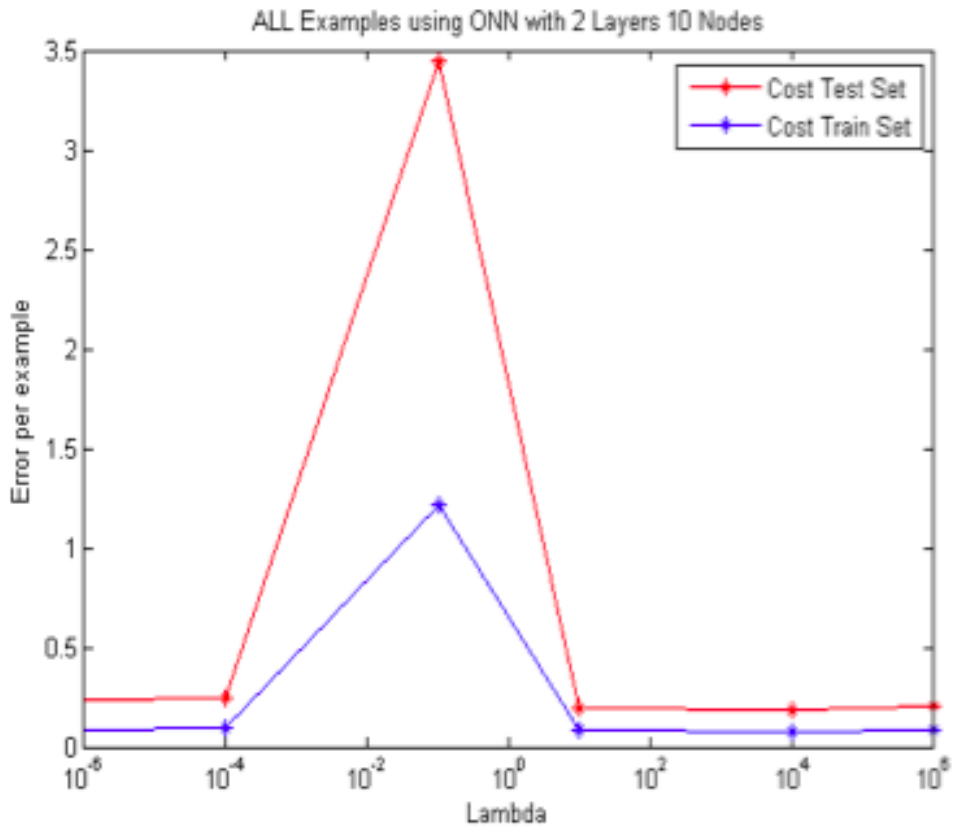

Figure 4: Error per examples vs. lambda using all examples for architecture using 2 layers and 10 nodes each

Some of the things we plan to do next are analyze the propagation of error as a function of system architecture (nodes/layers) to get a sense of which architecture might perform the best for this application. In addition to that we plan to analyze the error for each deadline label separately to see how the non uniformity of distribution of examples (per label) is affecting the performance of our algorithm.

7 Future Work

Now that the neural network has been implemented, we can begin to explore how different design choices affect our results. During training, we eliminated training examples that had incomplete feature vectors. In the final implementation of the ANN, we are going to investigate averaging methods in order to fill in these missing data. This will hopefully increase the number of unique examples we have for training and testing. In addition, there is more sensing data that has not been processed from the study that will give our ANN more insight into making accurate predictions. Once these adjustment on features have been made, different ANN architectures will be explored for varying values of lambda and parameter initialization values.

References

[1] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. *Neural Networks, 2008. IJCNN 2008*, pages 1279–1284, 2008.

[2] Andrew Ng. Coursera - machine learning. *https://www.coursera.org/course/ml*.

[3] Zhenyu Chen Andrew T. Campbell et el. Rui Wang, Fanglin Chen. Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. *In Proceedings of the ACM Conference on Ubiquitous Computing*, 2014.