CS 725 – Project Report

Shantanu Thakoor	Shudhatma Jain	Karan Vaidya	Sai Sandeep
13D100003	130050024	130050019	130050052

November 19, 2016

Introduction: Emotion Classification

Initially, we had proposed the project of emotion recognition from facial images. We had decided to implement a convolutional neural network based approach, in order to do 7-way emotion classification, based on the 7 basic emotions in human psychology.

However, datasets for this were not publicly available. We had submitted a request at the time of proposing the project, however we were unable to get usable data from this. We found a small public dataset, and worked with that. We achieved 84% accuracy on binary classification (positive vs negative emotion) and 42% accuracy on 7-way classification. However due to the small size of the dataset, we were unable to have results that generalized to our faces as well (the images in the dataset were of few people, always centered on the face, preprocessed, etc). Hence our original goal, of constructing a timeline of used emotions while performing some activity, could not be met.

Introduction: Mortality Prediction

We changed our project to that of mortality prediction based on medical records after consulting with TA and Professor. We found an interesting dataset based on `www.physionet.org/challenge/2012/set-a` We also implemented the methods and ideas presented in the paper found at `http://www.cinc.org/archives/2012/pdf/0261.pdf`

The dataset consisted of time series data with significant gaps. Hence our goal was not to find a best-performing model, but to learn techniques of handling time series information and data cleansing techniques.

Dataset Description

The dataset consists of medical records of patients given as a time series. It includes 6 vital records (such as blood pressure, heart rate, etc.) and 25 non-vital records (such as percentage of certain substances in the body). Note that we did not use any domain knowledge, and results can certainly be improved if insights from the world of medicine were to be incorporated in the models, especially for feature generation and data cleaning.

One key aspect is that, there are a lot of gaps in the data; not all medical records were measured at each timestamp. In particular, non-vital medical records were taken very sparingly. Hence, handling the data correctly and filling the gaps was very important. This problem was not minor: 83% of all timestamp data is missing, and 97% of non-vitals is missing! Since the

sparsity is so high, simple techniques like filling with dataset average would not work: all the datapoints would start to look very similar. In addition, we had to be very careful to use only data from the past while filling gaps (else information from the future would seep backwards and poison the data). The length of the time series was also different for each patient.

In addition to this data, we also have the time spent in the ICU and ages, of all the patients. This is not missing for anyone. Also, since our original datasets were sourced from a competition, they are anonymized and random noise has been added to each datapoint.

Training Approaches Used

We mainly considered two training approaches. We could either find a representative of each feature for each patient, and learn on that (hence having one training point per patient). However, this does not work well since the time series is very long for some patients, and they are healthy at the start; hence the representative of the sample would be very biased towards being healthy. In addition, this would make it difficult to predict mortality at the start of every series.

Instead, we use each timestamp as one training point; since we would like to predict mortality early for patients, we give the same label to all timestamps in the time series. Hence, we would be more likely to detect potential danger to the patient, early on.

Tackling Time Series Data

We used the idea of having an exponentially weighted average of features to fill the gaps in the time series; in cases where this was not available, we used the dataset mean. Again, we note that we can only use past data when filling information for a certain timestamp. This step introduces a hyperparameter: how much to scale the difference in time by. Currently, we give a reading weight exponential in how long ago it happened compared to the timestamp being filled.

Another idea we had was to include the variance. This was especially apt for the medical domain, since readings often show erratic changes near time of death.

We also note that we performed oversampling in order to remove the bias of the dataset towards a certain label. We tried adding random noise and thus perform dataset augmentation, but this was giving almost the same result. This may be due to the initial random noise in the dataset.

From Classification to Regression

Although predicting mortality seems to be a classical classification problem, having binary labels might not be very useful for a time series data. Since data at the beginning is very similar irrespective of mortality at the end of the series, it would be unfair to predict a high mortality chance based on that. Hence, we had the idea of exponentially decaying the labels; this has the effect of making the label a probability of dying after that particular timestamp. Our accuracies greatly improved after adding this, since we could now predict on earlier data more easily. The "death-likelihood" falls of exponentially with time as we go back from the time of death. This idea was our own, and not taken from the above paper.

Since we still have probabilistic semantics, we can also still use the cross-entropy loss as our loss function. We find this interpretation satisfying and convincing.

We do the final prediction based on whether the output score is greater than a particular threshold; this ideal threshold has been searched for. The paper presented an idea of a fuzzy

threshold where we give a random output for scores in a certain range; however this did not improve our scores significantly.

Neural Network Implementation

We trained a feed-forward neural network with 2 hidden layers of widths 100 and 30, with ReLU activation function at hidden layer nodes and sigmoid at the final output. Training was done with cross-entropy loss, using the exponentially decayed labels as explained above. Classification was done as: if more than X timestamps have predicted a score greater than Y, then we classify as dead, else alive. Hence, we view this as an ensemble method of sorts. In effect, voting is being done and if greater than a certain threshold of timestamps predict death, that is chosen as the label. Here we give unequal weightage to timestamps by not having the threshold as 0.5; this is because, timestamps near the end are more accurate than near the beginning. We were getting a training accuracy of around 66%, and test accuracy of around 89%. This difference can be attributed to the higher oversampling done in the training set. We also noticed that there was a higher bias towards predicting life than death.

Future work may include giving exponentially weighted importance to timestamps based on time from death; hence by reducing the cost of an earlier timestamp mispredicting, we may be able to get better estimates.

We have used the TensorFlow library for implementing this neural network.

Adaboost Implementation

Continuing on the ensemble methods trend, we then explored the Adaboost algorithm for this task, using decision trees as the base model (frequently cited as the best out-of-the-box model for classification tasks). However, we were getting worse results here: around 53% for training, and 11% for testing. In fact, the model almost always predicts death. This may be explained as: it is always predicting the most likely label in the training set. In other words, very little expressive power is with the model.

A possible explanation is: since adaboost with decision trees would try to use a different subset of features for each base model, and each feature has so many gaps (remember, 97% of the data is missing!), that many models using a subset of the features are weak. In other words, the features are very weak on their own to perform this sort of task.

Acknowledgements and Conclusion

We would like to acknowledge the dataset source and paper we have mentioned above.

For future work, we have the following suggestions. We could train an RNN or LSTM in order to take advantage of the sequential nature of the data, and hence learn such variations in the time series and exploit them in order to make predictions. In our own framework, we could also do better data wrangling or feature extraction and dimensionality reduction. Finally, a different loss function may be used to train the neural network (since final prediction is done based on voting mechanism etc).

We feel that we have learned quite a lot, and are very happy with the insights and ideas we had during the course of this project.