CS230

Using EMRs to Predict Patient Outcomes

Gaurab Banerjee
Stanford University
gbanerje@stanford.edu

Abstract

This project looks at utilizing free-text diagnosis notes to predict patient outcomes. As a baseline, ICD9 code are mapped to mortality using logisitic regression. Following this, ICD9 codes are used as the input in a multilayer-perceptron classifier to predict mortality. In contrast to this, a NLP approach is taken utilizing the free text notes where BERT is used for pretraining and then a simple linear classifier layer is used to output hospital mortality.

1 Introduction

Electronic Medical Records (EMRs) are the main form of storing information crucial to patient care such as unique identifiers, comments from caregivers, and lab test results. The amount of information stored within these EMRs provides large datasets which many data scientists have used to unearth trends linking different factors of patient care to onset of different forms of disease, or in our case, mortality rate.

EMRs contain information that can be generally abstracted to a couple of categories: coded and non-coded. Work has been done in the past with coded items which are easier to work with since they are generally binary or categorical variables. Indeed, in this work we rely on this to establish our baselines. The piece of EMRs worked with less often is their free-text information. Physicians, surgeons, and other major caregivers will often chart free-text and then nursing staff or medical staff will later code this information. This work primarily wants to focus on the efficacy of using this free-text information and seeks to compare the accuracy of using free-text as opposed to coded information. For the logistic regression and MLP-classifier, the inputs are the ICD9 codes and the outputs are whether the patient has expired or not. For the NLP piece of this project, the input is the doctors' free-text diagnosis notes and the output is once again, the mortality indicator.

Hospitals have a unique incentive to care about mortality prediction since ICU deaths count as inpatient deaths. Many use systems such as trauma scoring already. Studies have shown, though, that mortality predictions work better than trauma scoring in ICUs. Currently, mortality prediction involve a multitude of factors and complexity and, yet, are increasingly being used for clinical decision support. Consider the widely accepted Abbreviated Injury Scale which requires twenty minutes per patient.

The first scores proposed for mortality prediction were APACHE and SAPS and these were hand-computed with doctors determining the important features. Most ICU mortality predictors today rely on logistic regression. SICULA, created by Pirracchio et al. (2015), used multiple learning algorithms to get better mortality prediction and performed better than any of its component algorithms alone. This hinted at the usefulness of more robust learning algorithms in this space.

While critically ill patients are less than 10% of patients, they account for nearly one percent of the gross domestic product. A better mortality prediction algorithm to aid medical providers and families has much usefulness.

2 Dataset and Features

The data used here is from MIMIC-III (Medical Information Mart for Intensive Care III), containing information recorded in various EMRs from Beth Israel Deaconess Medical Center's Intensive Care Unit such as demographics, vital signs, laboratory tests, and more. There are about 1.1 million rows in this set and several columns represneting features. We choose some very specific features listed below.

The dataset includes a flag HOSPITAL_EXPIRE_FLAG, which has a value of 0 or 1 depending on whether a patient died in the hospital or not. The goal of this project is to predict mortality using free-text using the free-text provided in columns such as DIAGNOSIS: free text doctor's notes with no systematic ontology and which can be extremely vague.

The baseline algorithm is logistic regression. The HOSPITAL_EXPIRE_FLAG is the output and the input is the ICD9_CODE which is a categorical value where certain codes are linked to certain diagnoses (e.g. 280 is coded to anemia).

Most of the data processing was able to be accomplished using PANDAS and

The data was provided in multiple disjoint data sets. First, the data had to be merged using the patient ID as the linking variable. Following this, it was necessary to set up one-hot encoding for the categorical variable, ICD9_CODE column, since it would create a very sparse vector otherwise. From here, it was possible to do logistic regression and get some preliminary results. This same dataset could also be used for the MLP-classifier.

For the NLP approach, the data was a bit different. I removed all columns except 'DIAGNOSIS' and 'HOSPITAL_EXPIRE_FLAG'. From here, I was able to create tensors and later could use scikit's train_test_split to construction the necessary training and testing sets.

3 Methods

My code is at https://github.com/gbanerjee01/DLproject for the duration of this class. The dataset is not provided as you must get a special access granted by doing the required trainings listed on the MIMIC website.

3.1 Logistic Regression

Logistic regression is a form of binary classification with model:

$$log\frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta$$

where x represents the predictor variables, β represents the predictors, and $p(x)$ is as follows:

$$\frac{exp(\beta_0 + x \cdot \beta)}{1 + exp(\beta_0 + x \cdot \beta)}$$

We predict Y = 1 when $p \geq 0.5$ (when $\beta_0 + x \cdot \beta$ is nonnegative) and Y = 0 when $p < 0.5$ (when $\beta_0 + x \cdot \beta$ is negative). The decision boundary that separates the two classes is the solution of $\beta_0 + x \cdot \beta = 0$, and the distance of each point from the decision boundary is given by $\frac{\beta_0}{||\beta||} + x \cdot \frac{\beta}{||\beta||}$.

Because logistic regression is used for classification problems, the outputs correspond to mortality predictions given ICD9-CODE. This is the "simplest" algorithm used and sets the baseline.

3.2 Neural Networks

Neural networks work with some type of input, several hidden layers and activation functions which then result in an output. There is a loss minimization happening through forward and back propagation. In my usage of scikit's MLP classifier I set the parameters to be solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2) for the primary model parameters. The train/test split ratio is 0.8/0.2.

The data representation in this case is PANDAS dataframes into Numpy arrays. According to SciKit's page, the "MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters."

3.3 NLP and BERT

Bert is a Google pre-training approach that allows us to use a lot of transfer learning as identified in lecture and apply to our own smaller dataset. This allows word embeddings to be handled by BERT as long as we provide our data in appropriate tokenized format. I handled most of the architecture using a PyTorch implementation using Google Colaboratory since it was easiest there. Unlike a LSTM-RNN approach, BERT is a Transformer. The power of BERT is that it is a contextual model and not context-free meaning we can harness the power of relationships in our data. I used the Kana (2019) advice and implementation type which suggested the Hugging Face PyTorch Interface for BERT with a TF backend. I used an Adam optimizer with a learning rate of 2e-5 and 4 epochs since this seemed to work in other implementation of BERT. Changing this around did not appear to help. Once the BERT implementation was brought in, and the data was properly formatted, then the classification could be done. Since the data itself wasn't too large, this went fairly fast.

4 Results and Discussion

The logistic regression on ICD-9 codes has an AUC that is 0.43. When normalized, nearly 100% of 0's and 1's were predicted as 0's, saying all the patients were predicted to survive. The actual classification of the algorithm was poor. The first hypothesis for this is that deaths were not represented significantly enough in the data and were deeply overshadowed by the 0 flag in quantity.

The MLP-classifier, on the other hand, did much better. There was a 0.99 score on the neural network implementation. The first thought here is that there might again just be severe overfitting or perhaps the majority of patients fall into the same categories of codes, thereby misrepresenting the complete dataset if the test dataset cannot truly uniformly draw from the entire dataset.

Compared to this, the BERT classifier performed more poorly. This may be due to the extreme medical nature of the DIAGNOSIS free text column and on further examination some of the technical terms were incorrectly represented in the embeddings and were fragmented. It might also be that the actual notes are in fragments leading to this type of poor result.

5 Conclusion/Future Work

I think it would be really interesting to perform Principal Component Analysis to figure out why the MLP classifier did so well. Furthermore, it would be a proper next step to use the much larger body of text found in the column 'NOTES' for each patient and see how extraneous information in free text form might perform at this problem.

6 Contributions

Since I was the only team member, I did the whole project. All of the code inspiration and help I got (mostly for BERT and TensorFlow) is in the citations.

References

[1] Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3, 160035.

[2] Pirracchio, R., Petersen, M. L., Carone, M., Rigon, M. R., Chevret, S., & van der Laan, M. J. (2015). Mortality prediction in intensive care units with the Super ICU Learner Algorithm (SICULA): a population-based study. The Lancet Respiratory Medicine, 3(1), 42-52.

[3] Szarvas, G., Farkas, R., Iván, S., Kocsor, A., & Busa Fekete, R. (2006, December). Automatic extraction of semantic content from medical discharge records. In i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data.

[4] http://www.realworldnlpbook.com/blog/training-sentiment-analyzer-using-allennlp.html

[5] Marcin, J. P., & Pollack, M. M. (2002). Triage scoring systems, severity of illness measures, and mortality prediction models in pediatric trauma. Critical care medicine, 30(11), S457-S467.

[6] Meredith, J. W., Evans, G., Kilgo, P. D., MacKenzie, E., Osler, T., McGwin, G., Cohn, S., Esposito, T., Gennarelli, T., Hawkins, M., & Lucas, C. (2002). A comparison of the abilities of nine scoring algorithms in predicting mortality. Journal of Trauma and Acute Care Surgery, 53(4), 621-629.

[7] Johnson, A. E., & Mark, R. G. (2017). Real-time mortality prediction in the Intensive Care Unit. In AMIA Annual Symposium Proceedings (Vol. 2017, p. 994). American Medical Informatics Association.

[8] Rish, Irina (2001). An empirical study of the naive Bayes classifier (http://www.
research.ibm.com/people/r/rish/papers/RC22230.pdf) (PDF). IJCAI Workshop on Empirical Methods in AI.

[9] Vassar MJ, Lewis FR Jr, Chambers JA, et al: Prediction of outcome in intensive care unit trauma patients: A multicenter study of Acute Physiology and Chronic Health Evaluation (APACHE), Trauma and Injury Severity Score (TRISS), and a 24-hour intensive care unit (ICU) point system. J Trauma 1999; 47:324–329

[10] Wong DT, Barrow PM, Gomez M, et al: A comparison of the Acute Physiology and Chronic Health Evaluation (APACHE) II score and the Trauma-Injury Severity Score (TRISS) for outcome assessment in intensive care unit trauma patients. Crit Care Med 1996; 24:1642–1648

[11] Rutledge R, Fakhry S, Rutherford E, et al: Comparison of APACHE II, Trauma Score, and Injury Severity Score as predictors of outcome in critically injured trauma patients. Am J Surg 1993; 166:244–247

[12] Liang J, Zhou Z: Application of APACHE II scoring in ICU trauma patients. Chin J Traumatol 1998; 1:58–60

[13] Rhee KJ, Baxt WG, Mackenzie JR, et al: APACHE II scoring in the injured patient. Crit Care Med 1990; 18:827–830

[14] Klem SA, Pollack MM, Glass NL, et al: Resource use, efficiency, and outcome prediction in pediatric intensive care of trauma patients. J Trauma 1990; 30:32–36

[15] Johnson, A., Pollard, T., Mark, R. (2016). MIMIC-III Clinical Database. PhysioNet. doi:10.13026/C2XW26

[16] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[18] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[19] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch.

[20] Kana, M. (2019, October 26). BERT for dummies- Step by Step Tutorial. Retrieved from https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03.

[21] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[22] Alsentzer, E., Murphy, J. R., Boag, W., Weng, W. H., Jin, D., Naumann, T., McDermott, M. (2019). Publicly available clinical BERT embeddings. arXiv preprint arXiv:1904.03323.

[23] Travis E. Oliphant. Python for Scientific Computing, Computing in Science Engineering, 9, 10-20 (2007), DOI:10.1109/MCSE.2007.58

[24] K. Jarrod Millman and Michael Aivazis. Python for Scientists and Engineers, Computing in Science Engineering, 13, 9-12 (2011), DOI:10.1109/MCSE.2011.36

[25] Travis E. Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

[26] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

[27] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

[28] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)