ICU Mortality Prediction

Course project for CS-E5890

This project leverages the concepts covered in Lecture 3, “Deep generative modelling for biomedical data” to predict patient survival. You will build a deep generative model (Variational autoencoder) that can be used to learn the data generating function and then perform the predictive task of binary classification.

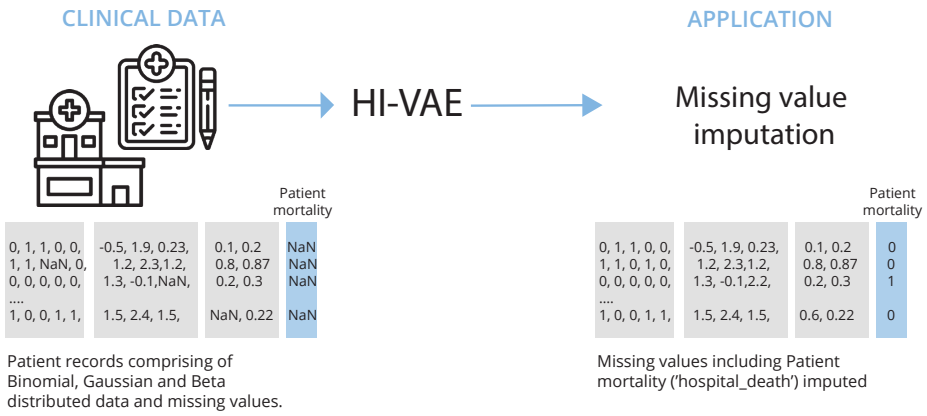
Introduction

There is a pressing need for the development of methods that can predict mortality rates in Intensive Care Units (ICU) so that the efficacy of medications, surgery and treatment regimens can be evaluated. In this project, you will build a generative model that can be predict if a patient will survive given a summary of their clinical health record. You will not make use of temporal data, instead the data has already been summarised to a single entry (row) for each patient.

Data

The data that you will use for this exercise is from MIT's GOSSIS community initiative and summarises a patient's first 24 hours of intensive care. The data comprises of 91713 instances with 186 covariates each. The covariates are from disparate likelihoods (i.e. heterogeneous data) and some measurements are missing. More information about the dataset and the dataset itself can be found here: https://www.kaggle.com/c/widsdatathon2020/overview . You will have to log-in to Kaggle and accept the competition rules to download the data. The true labels for the test data is not publicly available. Hence, you shall partition/hold-out 300 instances (for example) from the original data to form a test dataset.

Objective



Since this dataset comprises of heterogeneous covariates and missing values, you will make use of HI-VAE. The HI-VAE source code with examples on how to run the model is available at: https://github.com/probabilistic-learning/HI-VAE.

You must choose the appropriate likelihood for each covariate (follow the same structure as the code repository, in particular data_types.csv). In this scenario, we are trying to predict the 'hospital_death' binary covariate for the test data. Since HI-VAE is designed as a fully unsupervised generative model, you should include 'hospital_death' as a covariate in the training step. To perform binary classification on the test data, we shall simply impute the value of the 'hospital_death' covariate in the test data. In other words, the value of the 'hospital_death' covariate for each instance in the test data will be treated as missing and its value will be imputed using HI-VAE.

Report your model performance using the Area under the Receiver Operating Characteristic (ROC) curve between the predicted mortality and the observed target ('hospital_death').

Note: the objective of this project is to get a hands-on experience with deep generative models on real-world healthcare data. Your models are not required to compete with the Kaggle Challenge's leaderboard.

Prerequisite knowledge

The prerequisites listed below are just to inform you about the expected technical knowledge. They are not a criteria and you can learn the required technical skills.

- Python programming skills.
- Knowledge of python libraries like numpy, TensorFlow, matplotlib, etc.
- Machine learning - basic principles, deep learning and Bayesian data analysis.

Bonus tasks

- Explore various neural network architectures for the encoder and decoder. Also, tweak the various hyper-parameter choices. How does the choice of architecture and hyper-parameters affect your model's performance?

- Examine your choice of likelihood for the covariates. How is your model's performance affected by using, say Gaussian likelihood, for all covariates?
- Explore various choices of latent space ($\mathbf{z}$) dimension. Does your model's performance improve with higher latent space dimension? What about the model's computation time?
- Visualise the two dimensional latent space ($\mathbf{z}$) for the training data. Also, colour the points using the 'hospital_death' covariate. Do you observe any patterns? Compare the two dimensional representation with that obtained using other techniques like PCA.
- Compare your model's performance with other methods like conditional VAE and deep logistic regression.

Project report

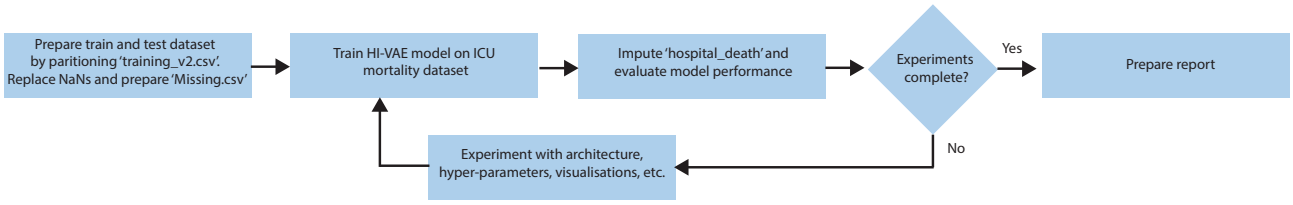
For the project report, follow the template on the MyCourses webpage. Be sure to include the following:

- A brief summary of your model including the various design choices that you made.
- Describe the various experiments you performed and your observations. See the 'Bonus tasks' section for experiment ideas. You are also encouraged to explore your own ideas!
- The performance of your model and the size of the test dataset that you used.

Useful tips and resources

- Familiarise yourself with the dataset. Make use of the 'pandas' library to explore the training set.
- The missing values are represented by NaNs. Be sure to replace the NaNs with zero and create the mask appropriately for the training and test dataset (a csv containing the positions of the different missing values in the data as in the GitHub repository). Approximately 33% of the training data is missing.
- The file 'WiDS Datathon 2020 Dictionary.csv' contains more information on the covariates.

- Make sure that the covariate ‘hospital_death’ is marked as missing for all instances in the test set. You will impute the value using the generative model trained on the training set.



Sample project workflow.

- A nice tutorial on VAEs:
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

- The HI-VAE source code can be found at:
https://github.com/probabilistic-learning/HI-VAE

- The HI-VAE model is proposed in the paper: Nazabal, A., Olmos, P. M., Ghahramani, Z., & Valera, I. (2018). Handling incomplete heterogeneous data using VAEs. arXiv preprint arXiv:1807.03653.
(https://arxiv.org/abs/1807.03653)

- The Kaggle completion and more information can be found here:
https://www.kaggle.com/c/widsdatathon2020/overview

- A nice explanation of Area under the Receiver Operating Characteristic (AUROC) can be found here:
https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

- Another nice tutorial on AUC scoring:
https://www.kaggle.com/learn-forum/53782