Early prediction of sepsis from clinical data using generative longitudinal modelling

Gleb Tikhonov

March 25, 2020

1 Introduction

Multivariate longitudinal data naturally arise in many biomedical applications. In this course project you will implement and apply some of the non-deep multivariate modelling techniques from Lecture 2 to analyse longitudinal data on incentive care unit (ICU) patients. Particularly, you will be asked to perform dynamic sepsis detection based on the time-series of vital signs and clinical measurements from the ICU patients.

This course project topic is fundamentally based on the the PhysioNet Computing in Cardiology Challenge 2019. However, to complete the project you are not required neither to participate in the Challenge itself, nor the course project grading will be based solely on the predictive performance that you achieve.

From technical point of view, in this project you are expected to get hands on experience on how to combine high-level and low-level machine learning routines in order to tailor your models to particular study designs. Further, you will obtain a more in-depth understanding of relationship between various model designs. Finally, you will learn how to couple fitted generative models with Bayes classifier.

2 Data acquiring and exploration

- Read the detailed PhysioNet2019 Challenge description from the Challenge webpage. The publicly available datasets can be downloaded via a link in that page or via this direct link.

- Focus your attention on the evaluation criteria of the challenge. Try to understand why such custom scoring strategy is used and why its practical utility is higher compared to standard scores used in machine learning (e.g. AUROC, accuracy).

- Read the downloaded data to the computational environment of your choice and form a single data matrix, where rows correspond to different samples and columns to different types of measurements.

- Assess which column of the resulted matrix it is worth to consider as outcomes that are going to be modelled and which as the covariates. Think whether it makes sense to transform some of the covariates or create some new ones? For example, the binary time-series of whether the patient is septic or not can be transformed into covariate that represents time before/after getting sepsis diagnosis.

- Investigate what is the a) structure of missing values in the data; b) variable types (e.g. unconstrained continuous, non-negative continuous, binary, categorical, ordinal, etc); c) empirical distribution of the data. Perhaps you can come up with some illustrative visualization of the dataset. Feel free to apply other data mining techniques of your choice.

3 Building a generative model

We will build the generative modelling framework for this data in a step-by-step manner. However, please note that the guidelines in this section cover only one optional way of how you can approach the problem. Please feel free to experiment with any alternative ideas that you may come up with.

We will use the following notation. We will index the observed patients with $p = 1 \dots P$, where P is the total number of patients. For each patient we will denote the matrix of modelled outcomes as $Y^{(p)} = \left[\boldsymbol{y}_1^{(p)T}, \dots, \boldsymbol{y}_{N_p}^{(p)T}\right]^T$ and the matrix of covariates as $X^{(p)} = \left[\boldsymbol{x}_1^{(p)T}, \dots, \boldsymbol{x}_{N_p}^{(p)T}\right]^T$.

1. Linear model with heterogeneous noise.

   We will start with a over-simplistic model, where we ignore all of multivariate data aspects and assume only linear relationship between covariates and outcome expectations.

   $$\boldsymbol{y}_i^{(p)} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_i^{(p)}), D), \qquad p = 1 \dots P, i = 1 \dots N_p$$

   $$\boldsymbol{\mu}(\boldsymbol{x}) = B\boldsymbol{x}, \qquad D = bdiag(\sigma_1^2, \dots \sigma_J^2)$$

   This model is parametrized by matrix of linear regression coefficients B and marginal variances $\sigma_1^2, \dots \sigma_J^2$. Please remember that some of the observation are missing, so you shall account for this feature properly.

2. Extension to non-linearity.

   Next we will replace the linear relationship between the $\boldsymbol{y}_i^{(p)}$ and $\boldsymbol{x}_i^{(p)}$ with a non-linear relationship. Since there is quite a lot of training data, you may prefer something flexible — e.g. neural network seems to be a viable option.

   $$\boldsymbol{y}_i^{(p)} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_i^{(p)}), D), \qquad p = 1 \dots P, i = 1 \dots N_p$$

   $$\boldsymbol{\mu}(\boldsymbol{x}) = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

   Here $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is some neural network that that takes the vector of covariates as input and returns the expectations for the data distribution $\boldsymbol{\mu}(\boldsymbol{x})$. It is parametrized by a set of weights and biases denoted as $\boldsymbol{\theta}$.

3. Extension to noise correlated across the outcomes.

   Both previous models did not acknowledge that given the multivariate and longitudinal nature of our data, the residuals may not be independent. First we will address the non-independence across the observed outcomes.

   $$\boldsymbol{y}_i^{(p)} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_i^{(p)}), \Sigma), \qquad p = 1 \dots P, i = 1 \dots N_p$$

   Here the Σ is no more assumed to be diagonal and therefore may capture the positive or negative correlations in the residuals. There are different ways to parametrize a covariance matrix Σ.

4. Extension to noise that accounts longitudinal design.

   Finally we will augment our generative model to account for longitudinal design of the data. Namely, we would account for the temporal structure of observations for each patient. We will use multivariate Gaussian process (GP) for this purpose, defined as a product of latent factors H with Gaussian process priors and latent loadings Λ. Denoting $\bar{\boldsymbol{y}}^{(p)} = \left[\boldsymbol{y}_1^{(p)T}, \dots, \boldsymbol{y}_{N_p}^{(p)T}\right]^T$, $\bar{\boldsymbol{\mu}}^{(p)} = \left[\boldsymbol{\mu}(\boldsymbol{x}_1^{(p)})^T, \dots, \boldsymbol{\mu}(\boldsymbol{x}_{N_p}^{(p)})^T\right]^T$, and marginalizing the Gaussian terms out, we would retrieve the following model

   $$\bar{\boldsymbol{y}}^{(p)} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}^{(p)}, \bar{\Sigma}^{(p)}), \qquad p = 1 \dots P$$

   $$\bar{\Sigma}^{(p)} = \sum_{r=1}^{R} K_r^{(p)} \otimes \Lambda_{r\cdot}^T \Lambda_{r\cdot} + I_{N_p} \otimes D$$

Here $\bar{\Sigma}^{(p)}$ is the cross-covariance matrix for all observations corresponding to patient p. $K_r^{(p)}$ is the GP prior covariance matrix of the r-th latent factor. Apart of the NN parameters and marginal noise variances, such model's parameter list also include the latent loadings Λ and hyperparameters of GP's covariance functions.

Please think what criticism can be put on the last model specified above. Try to devise and describe some remedies of how these drawbacks could be potentially resolved. What types of readily human-assessable inference can be acquired from this generative model?

4 Generative models for constructing Bayes classifier

The whole previous section is devoted to how-to build a more realistic generative model for the observations. However, the practical objective is actually rather different. Hence, you will use the constructed generative models as proxy to the true data generation mechanism to build a Bayes classifier and get probabilistic estimate on the probabilities and timing of sepsis for test patients. Effectively, given the model parameters that you've learned for the generative models, you will obtain the likelihoods of observing the test data under several alternative hypotheses.

The resulted probabilistic estimates then shall be turned into the predictions in line with the PhysioNet2019 Challenge criteria.

Since it is impossible to acquire the private test dataset for the challenge, you can use cross-validation and compute the predictive performances on the same publicly available dataset. You shall test different generative models (1-4 from the previous section) and may further investigate various configurations of those (number of layers/nodes in neural network, number of latent factors, etc).

5 Advices on practical implementation

In principle, the project can be completed using a wide variety of numerical framework. However, in my opinion it is most natural to implement the machine learning routines either with TensorFlow or with PyTorch. Therefore, any prior experience with these libraries would be very helpful, but if you are willing to get a practical experience in those, I an fully convinced that you will succeed as well.

Depending on how the upcoming project question-answer sessions would evolve, this course project description may get further updated with various useful notes and technical details.