**OBJECTIVE**

In this study, I am investigating if a deep network can learn to adequately represent a temporal sequence as a single vector. By casting this problem as a multi-class multi-label forecasting task, I am able to test how much information about the future does this representation hold. Given a clinical time series of **XT** = **{x1, …., xT}**, I train a classifier to predict a discrete label **ŷT+m** denoting the total IBD-related charges incurred in the period **T** to **T+m**.

**DATASET**

My experiments use a research registry of IBD patients collected at the University of Pittsburgh Medical Center (UPMC) tertiary care center for digestive disorders. The registry contains datasets of timestamped clinical events such as pathological lab results, active drug prescriptions, diagnostic and surgical procedures, encounters with care providers , Quality-of-Life survey responses, and billing data. I merged these events into a clinical time series with 53 variables for each patient. I resample them to a monthly rate and record the total measurements in each monthly window. After excluding patients being treated for cancer and/or who had a transplant (to prevent confounding the model), I have 2550 patients’ data whose treatment courses vary from 32 months to 11 years.

**IMPUTATION**

The dataset has numerous missing values, as is to be expected in EHR/EMR data. A variable might be missing because the clinician expected its value to be normal and chose not to measure it. The patterns of missingness might themselves hold important information; to capture this, I parametrically trade-off between the empirical mean and the imputed value of the missing variable. I impute variables based on certain assumptions about clinical practice:

1. Diagnostics, Surgeries, Encounters and all Prescriptions are imputed with 0. If it was not recorded, it most likely did not happen.
2. Lab values and HBI scores are assumed to have a homeostatic nature. They are forward-filled with the last observed value. When the entire variable is missing, it is imputed with a clinically normal value as defined by domain experts.

**MODEL**

The model consists of an *encoder* which learns a single vector that represents the input sequence (*hidden state*), and *classifiers* that predict an outcome. The encoder is a single GRU layer extended to handle irregularly-sampled sequences. The output of the encoder is fed to a classifier which makes the final prediction. The prediction error – “loss” – is backpropagated through the network back to the encoder. The model then updates its parameters to reduce this loss and make more accurate predictions.

**Target Replication**

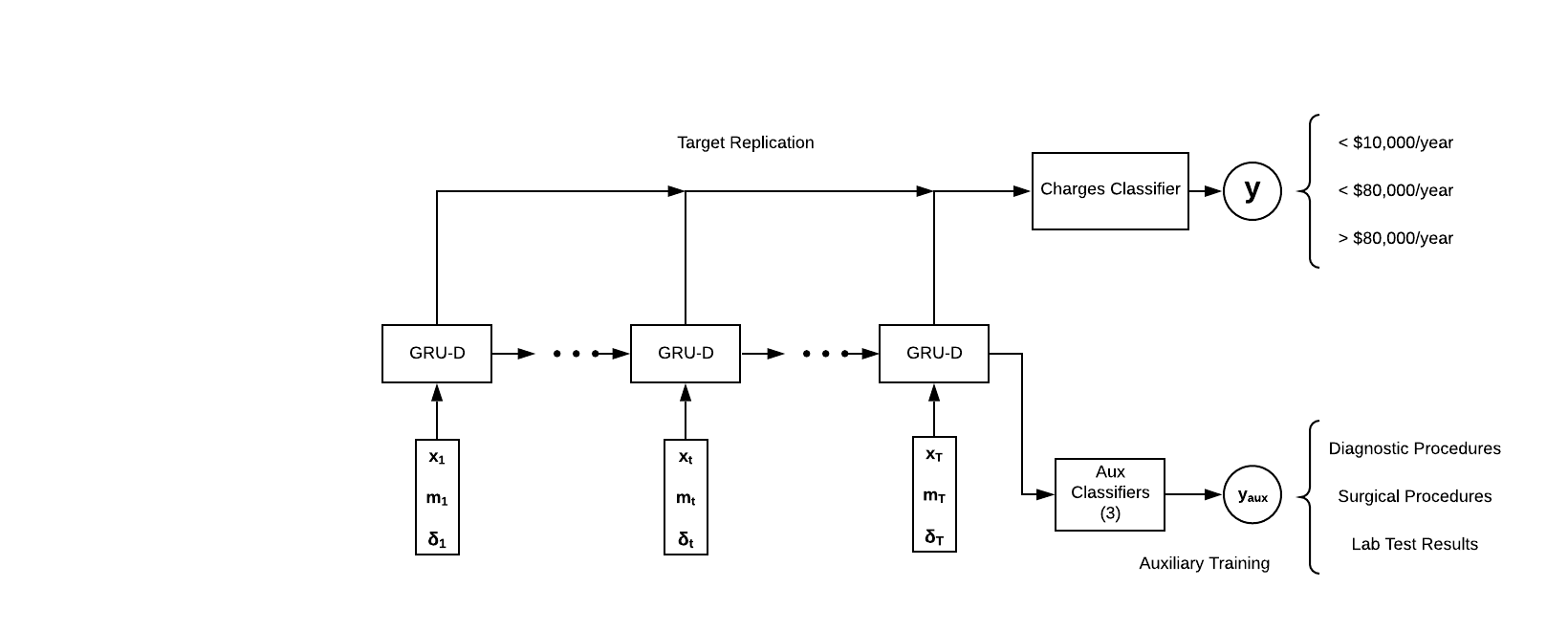
One difficulty with training RNNs is that this information must be passed throughout the sequence back in time; this can result in the algorithm converging on local optima that might be quite suboptimal. A simple workaround is to replicate the prediction task at each step in the sequence, and aggregating this prediction error over the entire sequence. This strategy intuitively makes sense as it causes the model to learn even interim representations that have higher predictive power for the charge classification task. My proposed architecture incorporates Target Replication by predicting the total charges in the target period at every step in the training period.

Figure . Model Architecture

**Auxiliary/MultiTask Training**

Multitask Learning (MTL) refers to the idea of sharing what is learned on parallel related tasks trained simultaneously (Caruana 1997). While not a concept unique to deep recurrent networks, it has recently received renewed interest in NLP tasks. In classifying clinical time series, it has been shown to improve regularization and extracting features that can generalize across multiple related tasks (Lipton 2017, Lee 2015, and more). My proposed architecture incorporates MTL by training the model on 3 auxiliary tasks:

1. The diagnostic procedures likely to be conducted in the target period,
2. Lab results likely to be abnormal in the target period, and
3. If any IBD-related surgeries are likely to be conducted in the target period.

Target Replication and MTL are used only while training the model.

**GRU-D**

The GRU-D cell is a vanilla GRU cell which incorporates trainable decay parameters for each variable in the input sequence. The decay rates are a function of **δt** which encodes since how long the variable has been missing.

**Classifier**

My model consists of 4 classifiers (3 for auxiliary, 1 for charges). Each contains two fully-connected layers of neurons which learn features most important to its prediction task. Classifiers use an activation function to transform the network’s linear outputs to a non-linear space. I use a Leaky Relu activation function at the outputs.

**Preliminary Results**

The models’ performances are only evaluated on the charge prediction task, and not on any of the auxiliary or replicated target tasks. All models are trained on 80% of the data and tested on 10%, with the remaining 10% used as a validation set. Throughout the training process, I track the training loss, validation loss and validation accuracy at each epoch, and choose the version with the least validation loss. To explore regularization effects, I record and plot the training and validation loss at each epoch.

To evaluate the model, I report the following metrics against the test set: Accuracy, Brier Loss, class-specific Area Under the ROC Curve (AUC), Precision, Recall, and F1-Score. Brier Loss ∈ [0,1] is a metric to evaluate the quality of probabilistic predictions, with 0 being the best and 1 being the worst.

Classification Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Low** | **Mid** | **High** | **Support-Weighted Average** |
| **Precision** | 0.9762 | 0.4547 | 0.5484 | 0.9137 |
| **Recall** | 0.8934 | 0.7199 | 0.8411 | 0.8787 |
| **F1-Score** | 0.9330 | 0.5574 | 0.6640 | 0.8906 |
| **AUC** | 0.9379 | 0.9096 | 0.9603 | 0.9376 |
| **Brier Loss** | 0.1408 | - | 0.0395 | 0.1336 |
| **Support** | 13888 | 1039 | 1070 | 15997 |

**Accuracy: 0.8786**

Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Predicted Low** | **Predicted Mid** | **Predicted High** |
| **Actual Low** | 12408 | 836 | 644 |
| **Actual Mid** | 194 | 748 | 97 |
| **Actual High** | 109 | 61 | 900 |