Building Deep Learning Models

Jenna Reps, Egill Fridgeirsson, Chungsoo Kim, Henrik John, Seng Chan You, Xiaoyong Pan

2022-08-08

Contents

1 Introduction

1.1 DeepPatientLevelPrediction

Patient level prediction aims to use historic data to learn a function between an input (a patient's features such as age/gender/comorbidities at index) and an output (whether the patient experienced an outcome during some time-at-risk). Deep learning is example of the the current state-of-the-art classifiers that can be implemented to learn the function between inputs and outputs.

Deep Learning models are widely used to automatically learn high-level feature representations from the data, and have achieved remarkable results in image processing, speech recognition and computational biology. Recently, interesting results have been shown using large observational healthcare data (e.g., electronic healthcare data or claims data), but more extensive research is needed to assess the power of Deep Learning in this domain.

This vignette describes how you can use the Observational Health Data Sciences and Informatics (OHDSI) `PatientLevelPrediction` package and `DeepPatientLevelPrediction` package to build Deep Learning models. This vignette assumes you have read and are comfortable with building patient level prediction models as described in the `BuildingPredictiveModels` vignette. Furthermore, this vignette assumes you are familiar with Deep Learning methods.

1.2 Background

Deep Learning models are build by stacking an often large number of neural network layers that perform feature engineering steps, e.g embedding, and are collapsed in a final softmax layer (basically a logistic regression layer). These algorithms need a lot of data to converge to a good representation, but currently the sizes of the large observational healthcare databases are growing fast which would make Deep Learning an interesting approach to test within OHDSI's Patient-Level Prediction Framework. The current implementation allows us to perform research at scale on the value and limitations of Deep Learning using observational healthcare data.

In the package we have used torch and tabnet but we invite the community to add other backends.

Many network architectures have recently been proposed and we have implemented a number of them, however, this list will grow in the near future. It is important to understand that some of these architectures require a 2D data matrix, i.e. |patient|x|feature|, and others use a 3D data matrix |patient|x|feature|x|time|. The FeatureExtraction Package has been extended to enable the extraction of both data formats as will be described with examples below.

Note that training Deep Learning models is computationally intensive, our implementation therefore supports both GPU and CPU. It will automatically check whether there is GPU or not in your computer. A GPU is highly recommended for Deep Learning!

1.3 Requirements

Full details about the package requirements and instructions on installing the package can be found here.

1.4 Integration with PatientLevelPrediction

The `DeepPatientLevelPrediction` package provides additional model settings that can be used within the `PatientLevelPrediction` package `runPlp()` function. To use both packages you first need to pick the deep learning architecture you wish to fit (see below) and then you specifiy this as the modelSettings inside `runPlp()`.

```r
# load the data
plpData <- PatientLevelPrediction::loadPlpData('locationOfData')

# pick the set<Model> from  DeepPatientLevelPrediction
deepLearningModel <- DeepPatientLevelPrediction::setResNet()

# use PatientLevelPrediction to fit model
deepLearningResult <- PatientLevelPrediction::runPlp(
    plpData = plpData,
    outcomeId = 1230,
    modelSettings = deepLearningModel,
    analysisId = 'resNetTorch',
    ...
  )
```

2 Non-Temporal Architectures

We implemented the following non-temporal (2D data matrix) architectures:

2.1 Simple MLP

2.1.1 Overall concept

A multilayer perceptron (MLP) model is a directed graph consisting of an input layer, one or more hidden layers and an output layer. The model takes in the input feature values and feeds these forward through the graph to determine the output class. A process known as 'backpropogation' is used to train the model. Backpropogation requires labelled data and involves iteratively calculating the error between the MLP model's predictions and ground truth to learn how to adjust the model.

2.1.2 Examples

To use the package to fit a MLP model you can use the `setDeepNNTorch()` function to specify the hyper-parameter settings for the MLP.

```
#singleLayerNN(inputN = 10, layer1 = 100, outputN = 2, layer_dropout = 0.1)
deepset <- setDeepNNTorch(
  units=list(c(10,63), 128),
  layer_dropout=c(0.2),
  lr =c(1e-4),
  decay=c(1e-5),
  outcome_weight = c(1.0),
  batch_size = c(100),
  epochs= c(5),
  seed=NULL
  )

mlpResult <- PatientLevelPrediction::runPlp(
    plpData = plpData,
    outcomeId = 3,
    modelSettings = deepset,
    analysisId = 'DeepNNTorch',
    analysisName = 'Testing Deep Learning',
    populationSettings = populationSet,
    splitSettings = PatientLevelPrediction::createDefaultSplitSetting(),
    sampleSettings = PatientLevelPrediction::createSampleSettings(),  # none
    featureEngineeringSettings = PatientLevelPrediction::createFeatureEngineeringSettings(), # none
    preprocessSettings = PatientLevelPrediction::createPreprocessSettings(),
    executeSettings = PatientLevelPrediction::createExecuteSettings(
      runSplitData = T,
      runSampleData = F,
      runfeatureEngineering = F,
      runPreprocessData = T,
      runModelDevelopment = T,
      runCovariateSummary = F
    ),
    saveDirectory = file.path(testLoc, 'DeepNNTorch')
  )
```

2.2 ResNet

2.2.1 Overall concept

2.2.2 Examples

2.3 TabNet

2.3.1 Overall concept

2.3.2 Examples

2.4 Transformer

2.4.1 Overall concept

2.4.2 Examples

3 Acknowledgments

Considerable work has been dedicated to provide the `DeepPatientLevelPrediction` package.

```
citation("DeepPatientLevelPrediction")
```

```
##
## To cite package 'DeepPatientLevelPrediction' in publications use:
##
##   Reps J, Fridgeirsson E, Chan You S, Kim C, John H (2021).
##   _DeepPatientLevelPrediction: Deep learning function for patient level
##   prediction using data in the OMOP Common Data Model_.
##   https://ohdsi.github.io/PatientLevelPrediction,
##   https://github.com/OHDSI/DeepPatientLevelPrediction.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {DeepPatientLevelPrediction: Deep learning function for patient level prediction using da
##     author = {Jenna Reps and Egill Fridgeirsson and Seng {Chan You} and Chungsoo Kim and Henrik John
##     year = {2021},
##     note = {https://ohdsi.github.io/PatientLevelPrediction, https://github.com/OHDSI/DeepPatientLeve
##   }
```

Please reference this paper if you use the PLP Package in your work:

Reps JM, Schuemie MJ, Suchard MA, Ryan PB, Rijnbeek PR. Design and implementation of a standardized framework to generate and evaluate patient-level prediction models using observational healthcare data. J Am Med Inform Assoc. 2018;25(8):969-975.