My fancy report on FIDDLE

For PHS 7045

Michael Throolin

Introduction

Electronic health record (EHR) data and machine learning tools have been useful in the realm of developing patient risk stratification models. However, EHR is often suffers from the curse of dimensionality and has irregular sampling. As such, before a researcher can use machine learning tools they have to preprocess the data, which is often tedious and time consuming. The preprocessing often requires decisions regarding variable selection, data resampling, and dealing with missing values. Such decisions vary across studies. FIDDLE (Tang et al. 2020) was proposed as an algorithm to streamline this process.

FIDDLE (Flexible Data-Driven Pipeline) is a systematic approach to transform structured EHR data into ML-ready representations. It is a data-driven approach which allows for some customization. The preprocessing steps of FIDDLE, as described by (Tang et al. 2020), involve three main stages:

Pre-filter:

- Remove rows with timestamps outside the observation period and eliminate rarely occurring variables to speed up downstream analysis.

Transform:

- Time-invariant data are concatenated into a table.
- Time-dependent data are concatenated into a tensor, with non-frequent variables represented by their most recent recorded value and frequent variables represented by summary statistics.
- Missing values are handled using carry-forward imputation with a "presence mask" to track imputed values.

Post-filter:

- Features that are unlikely to be informative, such as those that are almost constant across examples, are removed. Duplicated features are also combined into a single feature.

These preprocessing steps are designed to systematically transform structured EHR data into feature vectors that can be used for machine learning models.

IDDLE code was developed in Python (github repo). For my project I am translating this code to R. This is part of a group assignment I am doing for my EHR class where we will propose improvements to FIDDLE. I will code these improvements in R.

Description of the solution plan

From my preliminary reading of the python code, I have identified the following steps to translate the code to R:

1. Change items stored as a data frame to a data table
2. Use Rcpp to speed up the code where possible.
3. Parallelize what I can. As this is data preprocessing I can probably split the data up to do this once I have all the functions up and running.

After I have finished the main translation, I plan to explore a couple changes to the FIDDLE pipeline:

- Use a different imputation method for missing values. The last value carried forward may not be feasible in every pipeline.
- Use a different method for removing features that are unlikely to be informative. The method used in the FIDDLE pipeline is based on constant thresholds rather than an score-based system. I could implement some feature importance metric for this filtering.

Preliminary results

For the sake of this midterm, I have preprogrammed the pre-filter step in R and C++. I simulated some data to verify both codes produce the same results, then used microbenchmark to run the Rcpp code 100 times and to determine which was faster, my data.table code or my Rcpp code. You can check my code inside the code folder of this project's github repo.

A summary of the simulated data:

- 995165 rows of data
- 10000 subject-specific IDs
- 100 variables, 90 numerical and 10 categorical

- Time measurements between 0 and 10

Below I show how the data looks after it is processed as well as the timing results of the pre-filter code in R versus C++. It appears that using data.table in R was much faster than using Rcpp.

A sample of how the data looks after the pre-filter step:

```
       ID     t    var               value
   <char> <num> <char>              <char>
1:   5245   6.7  var16  0.345702948448439
2:   3573   7.7  var60  0.569509514591554
3:   4249   5.4  var22 -0.844704918926738
4:   7414   6.6  var41 0.0686402958837423
5:   6896   2.1  var96  -0.40937368185828
6:   4251   3.8  var29  0.915518839576742
```

Timing Results:

```
Unit: seconds
        expr      min       lq     mean   median       uq      max neval
   R_version 1.153373 1.338064 1.475449 1.344413 1.615523 1.925873     5
 Cpp_version 4.136790 4.499828 4.580618 4.533053 4.859891 4.873525     5
```

Appendix

References

Tang, Shengpu, Parmida Davarmanesh, Yanmeng Song, Danai Koutra, Michael W Sjoding, and Jenna Wiens. 2020. "Democratizing EHR analyses with FIDDLE: a flexible data-driven preprocessing pipeline for structured clinical data." *Journal of the American Medical Informatics Association*, October. https://doi.org/10.1093/jamia/ocaa139.