Team Meeting

26 Aug 2022 / 3:15 PM / Dual delivery

Attendees

Ate, Stefan, Xavier, Jiadong, Zexi, Ni

Agenda

Overall Approach - Ni

Sparse PLS & Variable Selection- Ate

Variable Selection by RF - Ni

Spartan & Data Preprocessing - Xavier, Stefan

Deep learning - Zexi

To Do

Overall Approach

Variable selection methods wrt data cleaning

- Variables given by Shuai
- Variables containing missing values
- Variables with low variance
- Variables with high covariance

Variable selection methods wrt modeling

- PLS / PCA
- SelectFromModel - Logistic Regression, Random Forest, Adaboost

1. For each of the five datasets, calculate no. of missing values for each column (variable)

   => These columns are the ones that may need to be removed

2. Join all five datasets to find common variables - **dataset 1 ~ 420k**

3. Drop the variables by different variable selection methods wrt data cleaning - **dataset 2 ~ 400k**
4. Find intersection between dataset to and the selected variables (by modeling) - **dataset 3 ~ 500-1000**

Sparse PLS & Variable Selection - Ate

In the nature paper, the following steps are taken:

Step 1: Discovery Analysis on the NTR dataset.

The epigenome-wide association study (EWAS) identified 243 epigenome-wide significant ($p < 1.20 \times 10-7$, Bonferroni correction for 411,169 tests) differentially methylated positions (DMPs) between MZ and DZ twins.

Step 2: Replication Analysis (Other four datasets)

Replication analysis in four independent twin cohorts revealed strong concordance of effects: correlations of effect sizes ranged from 0.84 to 0.97. The number of DMPs that replicated following Bonferroni correction for 243 tests ranged from 5 to 186.

Step 3: Sensitivity analysis (NTR and BSGS)

These cohorts also had DNA methylation data available for non-twins. These analyses included a comparison of MZ twins to non-twins (parents and siblings) rather than DZ twins, comparison of single MZ twins to single DZ twins (random exclusion of one twin from each pair), and sex-stratified analyses (Supplementary Data 2). The 243 sites showed highly consistent effect sizes across all analyses.

Step 4: Meta-Analysis

We next combined all blood EWAS results in a meta-analysis (total sample size = 5723, 88% of samples), which revealed 834 Bonferroni-significant CpGs, hereafter referred to as "MZ-DMPs" : 497 (60%) of which had a lower methylation level in MZ twins (MZ-hypo-DMPs) and 337 had a higher methylation level (MZ-hyper-DMPs).

Sparse PCA and Sparse PLS

- In python, the command **sklearn.decomposition**.SparsePCA can be used to derive the sparse PCA.
  [sklearn.decomposition.SparsePCA — scikit-learn 1.1.2 documentation]

- In r, we have a package called spls for sparse PLS.
  [Sparse PLS on R - Qiita]

- In python, we have the following package to perform PLS
  sklearn.cross_decomposition.PLSRegression — scikit-learn 1.1.2 documentation

- In python, the package py-ddspls can be used for data-driven-sparse PLS.
  py-ddspls · PyPI

Variable Selection by Random Forest - Ni

```
from sklearn.feature_selection import SelectFromModel
```

This method can also apply to: LogisticRegression, AdaBoostRegressor

Results after variable selection using random forest

Random Forest	Original - 833	Round 1 - 319	Round 2 - 121
E-Risk	0.8115	0.7787	0.8286
BSGS	0.8158	0.8014	0.7724

The AUC didn't change much but the model complexity has been reduced.

(The selected variable would be different every time you run it)

Reference:
https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f

Note:

The sparse SVC we found last week is an old version for SVC
http://scikit-learn.sourceforge.net/0.8/modules/generated/scikits.learn.svm.sparse.SVC.html

For the latest version of SVC, the default of input data is already assumed to be a sparse matrix

Parameters:: **X : *{array-like, sparse matrix} of shape (n_samples, n_features) or (n_samples, n_samples)***
Training vectors, where `n_samples` is the number of samples and `n_features` is the number of features.
For kernel="precomputed", the expected shape of X is (n_samples, n_samples).

Spartan & Data Preprocessing - Xavier, Stefan

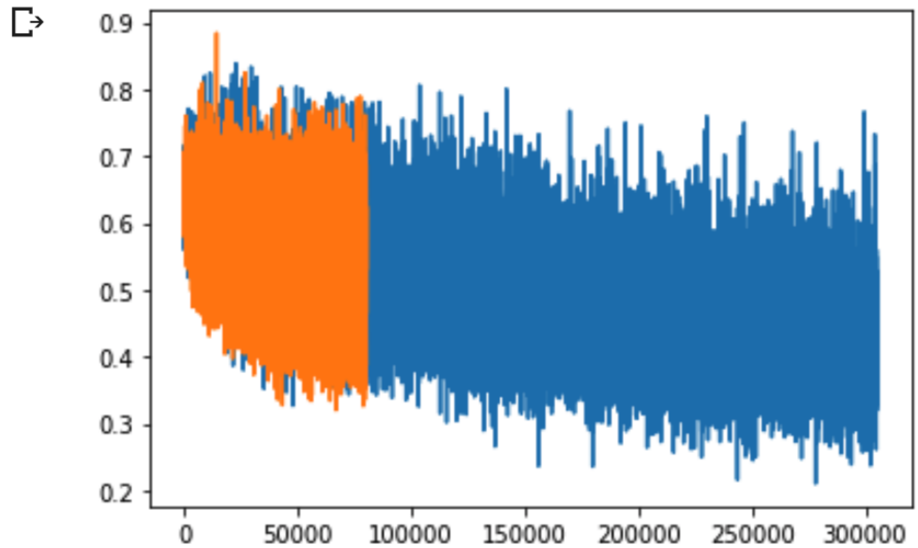
Dataset Name	Total Columns	Common Columns & probes deletion
E-MTAB	485,577	427,376
BSGS	485,577	
Denmark	485,577	
AMDTSS	479,597	
ERISK	430,867	

Dataset Name	Column With Missing Value	Total Missing Value	Average Missing Value Per Column	Max Missing Value Per Column	Total row number
E-Risk	0	0	0	0	
BSGS	183201	740468	4.0	564	614
Denmark	0	0	0	0	
E-MTAB	18339	32283	1.8	32	648
AMDTSS	0	0	0	0	

Missing values: Use KNN / avg to impute

Deep learning - Zexi

Implemented drop out, 0.2, smaller learning steps from 0.001 to 0.0001. 1 hidden layer (833 * 16*1), batch size of 64 => result in 0.8 AUC, significant improvement, but still not converge because run out of memory. Following graph shows the training loss and test loss.




```
epoch 15350     average training loss : 0.5084542036056519     average test loss:0.46139341592788696   test auc : 0.8033420264657587
epoch 15400     average training loss : 0.39761537313461304    average test loss:0.5528178215026855    test auc : 0.8031005505650536
epoch 15450     average training loss : 0.48474523425102234    average test loss:0.49628716707229614   test auc : 0.8033420264657587
epoch 15500     average training loss : 0.3486678898334503     average test loss:0.541593611240387     test auc : 0.8030522553849125
epoch 15550     average training loss : 0.36942604184150696    average test loss:0.5281187891960144    test auc : 0.8030522553849126
epoch 15600     average training loss : 0.5186009407043457     average test loss:0.5540214776992798    test auc : 0.8031005505650536
epoch 15650     average training loss : 0.37619540095329285    average test loss:0.5705896019935608    test auc : 0.8030522553849126
epoch 15700     average training loss : 0.4175921380519867     average test loss:0.7041391730308533    test auc : 0.8029556650246304
epoch 15750     average training loss : 0.3719558119773865     average test loss:0.5490924715995789    test auc : 0.8032454361054767
epoch 15800     average training loss : 0.4437069594860077     average test loss:0.5741021633148193    test auc : 0.8030039602047716
epoch 15850     average training loss : 0.48334673047065735    average test loss:0.49767351150512695   test auc : 0.8029073698444896
epoch 15900     average training loss : 0.3593886196613312     average test loss:0.5583887696266174    test auc : 0.8031005505650536
epoch 15950     average training loss : 0.2878909409046173     average test loss:0.5052749514579773    test auc : 0.8029073698444895
epoch 16000     average training loss : 0.46109670400619507    average test loss:0.6154130697250366    test auc : 0.8030522553849125
epoch 16050     average training loss : 0.3314194083213806     average test loss:0.5016703605651855    test auc : 0.8029073698444895
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-19-9bad266625de> in <module>
     13     batch_train_loss = []
     14     #calculate output
---> 15     output_train = model(x_train.to(device))
     16     #calculate loss
     17     train_loss = loss_fn(output_train,y_train.reshape(-1,1).to(device))

RuntimeError: CUDA out of memory. Tried to allocate 2.00 MiB (GPU 0; 15.90 GiB total capacity; 15.16 GiB already allocated; 21.75 MiB free; 15.16 GiB reserved in total by PyTorch)
If reserved memory is >> allocated memory try setting max_split_size_mb to avoid fragmentation.  See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

Questions for Shuai

1. Use modeling to select variables from all 5 datasets, then use the selected feature to train E-Risk
2. Use 80% from each of the 5 dataset to do variable selection and training, then the rest 20% on testing
3. Use 3 datasets to do the model training and variable selection, then the rest 2 on testing (cross validation?)

To Do

1. Schedule a short meeting with Shuai