Team Meeting

17 Aug 2022 / 10:00 AM / ZOOM

Attendees

Zexi, Stefan, Xavier, Ate, Ni, Jiadong, Shuai, John, Susan

Agenda

1. **Introduction to the selected machine learning models**
2. **Results of the trained models**
3. **Big data - hpc / Spartan**
4. **Questions from us**
5. **TO DO**

Part 1: Introduction to the selected machine learning models

Current machine learning models:

SVM, Logistic Regression (l2 norm - ridge regression), MNB, Adaboosting, Random Forests, Gradient Boosting;

Stacking, Voting (combination of the first five models, gradient boosting was rejected due to high computation complexity);

Stacking Model

```python
# get a stacking ensemble of models
from sklearn.ensemble import StackingClassifier
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('lr', LogisticRegression(tol = 0.001, solver = 'sag', penalty = 'l2', C = 30)))
    level0.append(('rf', RandomForestClassifier(n_estimators = 500, min_impurity_decrease = 1e-06, max_depth = 50, criterion = 'gini')))
    level0.append(('gb', GradientBoostingClassifier(n_estimators = 300, max_depth = 5, learning_rate = 0.5)))
    level0.append(('svm', SVC(kernel = 'rbf', gamma = 'scale', degree = 1, decision_function_shape = 'ovr', C = 20)))
    level0.append(('mnb', MultinomialNB()))
    # define meta learner model
    level1 = LogisticRegression(tol = 0.001, solver = 'sag', penalty = 'l2', C = 30)
    # define the stacking ensemble
    model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
    return model
```

Python

PCA and PLS (dimension reduction method)

Pytorch (deep learning method).

Table 6: Machine learning methods applicable in classification problems

Method	Pros	Cons
SVM	• Insensitive to over-fitting	• Might not be suitable for large data-set
Logistic regression	• Good accuracy for many simple data-sets • Good performance for linearly separable data-sets	• High dimension data tend to over-fit the model • Feature selection/ dimensionality reduction is necessary
Bayesian algorithm	• Good performance on small-scale data • suitable for incremental training • Not sensitive to missing data • relatively simple algorithm	• Need to calculate the prior probability • Classification decision has error rate • Not good if the sample attribute is related
Adaboosting	• High-precision classifier • Simple to implement • Overfitting is not easy to occur	• The number of iterations is not easy to set • Sensitive to data imbalance • Training is time-consuming
Stochastic gradient boosting	• Avoid the problems from overfitting • high performance in high dimensional data • Using boosting algorithms • Consistent approximation	• High computational complexity
Random forests	• Not computationally intensive • high generalization accuracy • Low classification error rates • Little need to tune parameters • Robust and does not overfit	• For very large data sets, the size of the trees can take up a lot of memory. • Poor performance on imbalanced data

Part 2: Results for these models

AUC	Stacking	Voting	PLS	Nature
E-Risk	**0.8516**	**0.8357**	**0.8134**	**0.739**
BSGS	**0.8092**	**0.8031**	**0.7597**	**0.774**
Denmark	**0.6301**	**0.6587**	**0.6933**	**0.563**
E-MTAB	**0.6356**	**0.6782**	**0.6652**	**0.522**
AMDTSS	**0.7173**	**0.7139**		**0.648**

1. From the training Loss, clearly not converge => parameters: 18,000 epoch, 1.1 * 10^11 number of parameters, training takes 14 minutes on GPU.
2. The auc of test dataset stays around 0.665, there could be many reasons, need to further investigate reasons, 1. write codes for test loss 2. Smaller learning rate 3. potentially because not enough data 883 columns, 1000+ rows

```
training loss : 0.2484523355960846      test auc : 0.6515535529897681
8
training loss : 0.017954792827367783    test auc : 0.6578428611658688
2
training loss : 0.025135910138487816    test auc : 0.6555430395193842
1
training loss : 0.09713796526193619     test auc : 0.6569980287243031
.3
training loss : 0.01765984669327736     test auc : 0.6578897963015113
'6
training loss : 0.07118618488311768     test auc : 0.6565286773678776
'6
training loss : 0.07706234604120255     test auc : 0.6565286773678776
6
training loss : 0.12084238231182098     test auc : 0.6566225476391626
5
training loss : 0.07480235397815704     test auc : 0.6547920773491035
.7
training loss : 0.10140485316514969     test auc : 0.6554961043837417
3
training loss : 0.05426829308271408     test auc : 0.6568572233173753
8
training loss : 0.03883696720004082     test auc : 0.6555899746550268
```

Part 3: HPC (SPARTAN)

Done:

1. Created project directory on Spartan
   - /data/gpfs/projects/punim1257/Group31
2. Installation for the required packages
   - foss/2019b
   - iomklc/2019.05
   - foss/2020b
   - fosscuda/2019b
3. Run a demo code for E-Risk dataset (833 columns)

```
1 node and 1 core:
spartan-bm084.hpc.unimelb.edu.au Group31_1n1c
The AUC baseline is: 0.7931818181818182
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best Score:  0.8003239657385539
Best Params:  {'tol': 1e-07, 'solver': 'lbfgs', 'penalty': 'l2', 'C': 1}
```

(Output result of demo code)

4. Downloaded the 450K E-Risk dataset on Spartan

```
[stefan@spartan-login2 Group31]$ ls -ahl
total 13G
drwxrwsr-x 2 haozex punim1257 4.0K Aug 12 15:41 .
drwxrws--- 6 root   punim1257 4.0K Aug  4 14:21 ..
-rwxrwxrwx 1 haozex punim1257  402 Aug 11 14:45 1n1c.slurm
-rw-r--r-- 1 haozex punim1257  24M Aug 12 15:32 ERisk_data.csv
-rw-r--r-- 1 haozex punim1257  68K Aug 12 15:41 .ERisk_data.csv.swp
-rw-r--r-- 1 stefan punim1257  13G Aug 11 17:27 GSE105018_NormalisedData.csv
-rw-r--r-- 1 haozex punim1257 1.4K Aug 11 14:14 logistic_regression.py
-rw-r--r-- 1 haozex punim1257 760K Aug 12 15:38 Probes_to_exclude.txt
-rw-rw-r-- 1 haozex punim1257 9.7K Aug 12 15:30 slurm-38399909.out
[stefan@spartan-login2 Group31]$
```

To Do:

- Useless Probe Deletion
- Write MPI to implement concurrent processing on SPARTAN
- Combine the dimension reduction method and best model into one single python file
- Apply the model with full dataset (to see whether there is a problem regarding high dimensional data - whether the AUC will decrease? Whether overfitting may occur?)

Questions from us

1. Normalization method resulted in low AUC
2. Dimension reduction or variable selection? Interpretation or AUC?
3. We are currently working on python, do we need to have another version in R?

To Do

1. Variable selection - missing values, variance, mixOmics (sparse pls:=splss)
2. Try the common probes first

Steps:

1. Preprocessing (Ate)
2. Sparse PLS and Sparse PCA (Ate)
3. Sparse Machine Learning Methods

Jiadong Mao to Everyone 10:44 AM

1, spls + ML algorithms

2, ML algorithms with feature selection

2, Sparse ML algo