Implementation of SynB-RVM for Uncertain Software Effort Estimation

June 18, 2021

1 Algorithm

This document presents how to implement Synthetic Bootstrap ensemble of Relevance Vector Machines (SynB-RVMs) proposed by Song et al. [1], as an example to demonstrate how to produce effort estimations for software projects following the Artificial Intelligence (AI) scheme. The tool is implemented in Matlab under GNU GLP 3.0 license.

Once the tool is downloaded, there is a folder called *matlab*, containing the codes. Another folder called *data_example* contains the edited software projects for SEE. To use the tool, one needs to configure it by running the script *config.m* or typing in the command window the following line:

```
>> config()
```

This function configures the directories between code scripts and the data set and among all scripts, so that the scripts can call each other and load the data set as if they are in a single one-layer directory.

An example of running the overall implementation of SynB-RVM is given in the script *example_run_SynB_RVM.m* in the directory "*matlab/examples/*".

1.1 Preparing the Training and Test Projects

To simulate a real-world SEE scenario, the original Nasa93 [2] is split into a training and a test set according to the time order of the software projects. Specifically, 90% projects that completed earlier are used as the training data and the remaining 10% projects are retained as the test examples to showcase point effort predictions and prediction intervals produced by SyB-RVM. The 17 features used to describe projects of Nasa93 are in the tool's folder `data_example/nasa93_edited`. The suffix `edited` means that the data set has been revised to be used with Matlab. For example, the feature 'stor' (main memory constraint) with the values of `vl`, `l`, `n`, `h`, `vh` and `xh` are converted to the values of 1, 2, 3, 4, 5 and 6, respectively.

To obtain the data division into training and test sets in one step, one can run the script *get_nasa93.m* or type in the command window the following line:

```
>> [X_train, y_train, X_test, y_test] = get_nasa93()
```

where `X_train` and `X_test` contain the input features of the training and test examples, respectively, and `y_train` and `y_test` contain their corresponding required efforts in person-months. Through *get_nasa93.m*, the training and test examples are pre-processed to prune out three outliers and to conduct feature normalization. If one would like to run the pre-processing separately, this process is implemented in the script *data_preprocess.m* and can be run in the command window by typing the following line:

```
>> [years, X, y] = data_preprocess(Data);
```

where `Data` denotes the original projects of Nasa93, `years` denotes the time of completing each project, and `X` and `y` are their features and actual effort for development.

1.2 Training and Prediction Algorithms of SynB-RVM

The training and prediction algorithms of SynB-RVM are implemented in the script *SynB_RVM.m* in the directory "matlab/train-prediction/". Script *example_run_SynB_RVM.m* in the directory "matlab/examples/" exemplifies the usage of the training and prediction algorithms, producing point and interval predictions with confidence level of 85% for the projects in the test set created in Section 1.1. Type the following command to run this script:

```
>> example_run_SynB_RVM()
```

The predicted point and interval prediction may improve or deteriorate with different parameter settings. Section 1.5 shows how to tune parameters. The parameter setting of SynB-RVM adopted in this example is shown in the script *example_run_SynB_RVM.m* as

```
% SynB-RVM's hyper-parameters
m_bags = 10;        %  #(Bootstrap Bags)
pru_m = 0.1;        % prune rate
tho = 0.01;         % synthetic displacement
```

This script prints the results below:

```
Prediction intervals with CL 0.85 of test projects in Nasa93 are:
   id    actual   point estimate   prediction interval  hit/not
    1    210.00       233.33        109.37 -   357.29      1
    2     48.00       133.33          1.38 -   265.29      1
    3     50.00        55.00          0.00 -   162.96      1
    4     60.00        65.00          0.00 -   172.96      1
    5     42.00       122.22          6.26 -   238.18      1
    6     60.00       144.44         28.48 -   260.41      1
    7    444.00       516.67        408.70 -   624.63      1
    8     42.00       144.44         28.48 -   260.41      1
    9    114.00       183.33         67.37 -   299.30      1
```

The second column presents actual efforts of test projects. These values would not have been available in practice at the prediction stage. They would become available only after the project finishes, if effort information is collected during the development process. The third column reports the corresponding point effort estimations. The fourth column reports the prediction intervals with confidence level of 85%. The last column reports whether or not the prediction interval covers the actual effort for each test project.

1.3 Evaluating The Performance Of A SynB-RVM Model

Performance evaluation for the point estimations and interval predictions are implemented in the scripts *eval_point_pf.m* and *eval_PI_pf.m* in the directory "matlab/evaluate/", respectively. The evaluation of point predictions can be run with the following command:

```
>> eval_point_pf(y_true, y_pred)
```

The input arguments `y_true` and `y_pred` are column vectors consisting of the true and predicted efforts of test projects, respectively. These estimated point efforts are obtained from the function *example_run_SynB_RVM()*. This function prints point predictive performance of SynB-RVM as shown below.

```
ans = structures consisting of

       mae: 57.8214
      mdae: 82.3365
     mlgae: 3.5704
    mdlgae: 4.4108
      mmre: 0.9730
     mdmre: 0.8201
    pred25: 44.4444
    pred15: 44.4444
    pred10: 22.2222
      coor: 0.9536
       lsd: 0.6875
      rmse: 69.2337
     rmdse: 82.3365
        sa: 0.4656
```

To evaluate the prediction intervals (measured in *hit rate* and *relative width*), the following command can be used:

```
>> [hit_rate, relative_width] = eval_PI_pf(PI, y_pre_mean, y_true)
```

The input argument `PI` is a data matrix where each row represents the prediction interval for a test project and `y_pre_mean` is a column vector consisting of the point effort estimations. The output includes the information below:

```
hit_rate = 1
relative_width = 1.6957
```

1.4 Using SynB-RVM In What-If Scenarios

To investigate projects with higher computational and storage constraints, we can increase the values of the resource related features of the test projects in Nasa93, including *data base size constraint*, *time constraint for cpu* and *main memory constraint*.

The generation of test projects is implemented in *get_nasa93_resource(·)* of the directory "matlab/examples/", and can be run by typing the command below:

```
>> get_nasa93_resource('highest')
```

After that, we can observe SynB-RVM's effort estimations for these modified projects. The experiment is implemented in the script *experiment2_resource.m* in the same directory. One can run it by typing the following command:

```
>> experiment2_resource('highest')
```

where the input argument `highest` means that the highest resource constraint is evoked. Point and interval predictions of SynB-RVM are shown below:

```
=======================================
SynB-RVM_ht2d's prediction on the test projects in Nasa93 that
are reconstructed with the highest resource constraint.
Prediction intervals are with confidence level of 0.85.
   id predicted   prediction interval
    1    805.56    689.59 -    921.52
    2    472.22    372.25 -    572.19
    3    516.67    424.70 -    608.64
    4    516.67    424.70 -    608.64
    5    638.89    522.93 -    754.85
    6    527.78    411.82 -    643.74
    7    750.00    642.04 -    857.96
    8    527.78    411.82 -    643.74
    9    527.78    411.82 -    643.74
```

If we assume another extreme scenario where the project manager is considering to have very light computational and storage constraints. In this scenario, we would take the lowest constraints on the database size, time and memory constraint features of Nasa93 projects. This can be implemented by running the function below:

```
>> get_nasa93_resource('lowest')
```

We can then use SynB-RVM to estimate the efforts required to develop those projects with the least computational and storage constraints, by running the command below:

```
>> experiment2_resource('lowest')
```

where the input argument `lowest` means that the lowest constraint of the development resource is evoked. The point and interval predictions of SynB-RVM are shown below:

```
=======================================
SynB-RVM_ht2d's prediction on the test projects in Nasa93
that are reconstructed with the lowest resource constraint.
Prediction intervals are with confidence level of 0.85.
   id predicted   prediction interval
    1    166.67     50.70 -    282.63
    2    128.57     10.32 -    246.82
    3     55.00      0.00 -    162.96
    4     65.00      0.00 -    172.96
    5    125.00      8.04 -    241.96
    6    144.44     36.48 -    252.41
    7    455.56    347.59 -    563.52
    8    125.00     17.04 -    232.96
    9    144.44     36.48 -    252.41
```

Let us now assume a scenario where the project manager would like to give medium (nominal) computational and storage constraints. This can be achieved by replacing the resource related features of the projects in Nasa93 with the *nominal* value. We can implement this by running:

```
>> get_nasa93_resource('balanced')
```

We can then use SynB-RVM to estimate the efforts required to develop those projects with the balanced constraint. by running the below command:

```
>> experiment2_resource('balanced')
```

where the input argument `balanced` means that a nominal constraints are evoked. Point and interval predictions of SynB-RVM are shown below:

```
=======================================
SynB-RVM_ht2d's prediction on the test projects in Nasa93
that are reconstructed with the balanced resource constraint.
Prediction intervals are with confidence level of 0.85.
   id predicted   prediction interval
    1    433.33    301.38 -    565.29
    2    188.89     72.93 -    304.85
    3    277.78    169.81 -    385.74
    4    300.00    192.04 -    407.96
    5    383.33    267.37 -    499.30
    6    283.33    167.37 -    399.30
    7    616.67    508.70 -    724.63
    8    300.00    192.04 -    407.96
    9    316.67    200.70 -    432.63
```

1.5 Decide Parameters of SynB-SVM

This section aims to demonstrate how to tune the parameters of SynB-RVM. The parameter tuning process is implemented in the directory */matlab/example-para_tune/*. To run the parameter tuning, the following command can be used:

```
>> experiment_para_tune()
```

The best parameter configuration decided by this experiment, the point and interval predictions of SynB-RVM with the chosen parameter setting and their predictive performance are shown below:

```
=======================================
The best parameter setting based on this experiment is as below:
     the best number of Bootstrap bags is 30
     the best degree of synthetic displacement is 0.01
     the best pruning rate is 0.1

=======================================
Predictive performance of "Nasa93" with SynB_RVM_ht2d.
---------------------------------------
Overall predictive performance is as below:
     mae = 64.1
     hit_rate = 1.00
     relative width = 1.88
---------------------------------------
Prediction intervals of CL=0.85 of all test projects are as:
   id    actual  predicted   prediction interval     hit/not
    1    210.00     244.00    101.49 -    386.51      1
    2     48.00     166.67     17.57 -    315.76      1
    3     50.00      56.11      0.00 -    182.74      1
    4     60.00      67.78      0.00 -    194.40      1
    5     42.00     137.50      0.00 -    275.46      1
    6     60.00     146.15      7.74 -    284.57      1
    7    444.00     492.59    360.64 -    624.55      1
    8     42.00     140.00      0.37 -    279.63      1
    9    114.00     196.30     53.68 -    338.92      1
```

References

[1] Liyan Song, Leandro L. Minku, and Xin Yao. Software effort interval prediction via Bayesian inference and synthetic Bootstrap resampling. *ACM Transactions on Software Engineering Methodology*, 28(1):1–46, 2019.

[2] Nasa93 doi. `https://doi.org/10.5281/zenodo.268419`, 2008.