Higgs CP state - Machine Learning

August 2018

1 Setup

1.1 Python - Anaconda

Python version used for this project is Python2. Anaconda is the suggested distribution of Python to be used both on personal computer and Prometheus supercomputer.

Steps to follow in order to Anaconda:

1. Download Anaconda
2. Create new Python2 environment in Anaconda
3. Install any needed Python libraries in the newly created environment
4. Assign the value of `ANACONDA_PYTHON_PATH` variable in setup file to path of Python in the new environment

1.2 Environment variables

To facilitate usage of Python scripts and remove the need to provide paths etc. each time, setup file template was prepared. `setup_template` can be found in main folder of the project. It should be copied to `slurm_scripts` folder. Then the content of the file should be edited to reflect paths used by the user and appropriate lines should be uncommented.
The variables in the file are:
`ANACONDA_PYTHON_PATH` - path to Python from Anaconda distribution
`WORKDIR` - path to the project
`A1A1_DATA` - location where A1A1 dataset/where it should be downloaded for later use
`A1RHO_DATA` - location where A1Rho dataset/where it should be downloaded for later use
`RHORHO_DATA` - location where RhoRho dataset/where it should be downloaded for later use
`DATA_SOURCE` - address from where the datasets should be downloaded

Example content of the setup file:

```
export ANACONDA_PYTHON_PATH=/net/people/plguser/anaconda2/envs/envname/
bin
export WORKDIR=/net/people/plguser/ml_higgs
export A1A1_DATA=/net/scratch/people/plguser/higgs_data/data_a1a1/
export A1RHO_DATA=/net/scratch/people/plguser/higgs_data/data_a1rho/
export RHORHO_DATA=/net/scratch/people/plguser/higgs_data/data_rhorho/
export DATA_SOURCE="http://example.com/"
```

After variable's value is provided, adequate line from the file should be uncommented. Then, when all values were added, content of the file should be evaluated using following command in the command line:

```
source setup
```

1.3 Prometheus: Slurm

Prometheus uses Slurm job scheduler which demands special scripts to be run in order to submit a job. Some useful slurm script can be found in `slurm_scripts` folder and can be used as they are or as examples of how to prepare new slurm scripts.
To submit a job, following command should be used:

```
sbatch slurm_script_name.sh
```

Useful Slurm/Prometheus commands include:
`sacct` - "displays accounting data for all jobs and job steps in the Slurm job accounting log or Slurm database"
`pro-jobs` - "view of jobs scheduled in queuing system"
`pro-jobs-history` - "historical data of completed jobs"

1.4 Prometheus - other comments

The code on Prometheus should be stored in Home directory, `/net/people/plguser/`, while the data on which calculations are carried out should be stored in Scratch, `/net/scratch/people/plguser/`.
When running in batch mode, grant intended for the computations should be set as default or appropriate parameter should be added to slurm run scripts (`#SBATCH -A grant_ID`).

1.5 Python libraries

Python libraries used in the project include:

- argparse
- inspect
- matplotlib
- numpy

- os
- re
- sklearn
- sklearn.ensemble
- sklearn.metrics
- sys
- tensorflow
- xgboost (installation: `conda install --name envname -c conda-forge xgboost`)

As not all of them are used in each of the scripts, they can be installed only when necessity comes.

2 Code repository structure

```
ml_higgs
├── plot_scripts
├── slurm_scripts
└── thesis-helpers
```

`ml_higgs` - Python scripts

`plot_scripts` - Python scripts to prepare plots/heatmaps

`slurm_scripts` - slurms scripts to be used to run Python scripts in batch mode

`thesis-helpers` - other useful scripts

3 Main python file

Script `main.py` is used to run training of classifiers, given appropriate parameters. Full list of parameters than can be passed are presented in Table 1.

4 Slurm scripts

Slurm scripts can be found in `slurm_scripts` directory and should be used to run training scripts on Prometheus in batch mode. The names of output files correspond to job name eg. slurm-12345678_X.out for job with id 12345678 where X refers to "subjob" if one script runs multiple jobs.

Available scripts:

- train_ABCmethods.sh

Table 1: main.py script args

Flags	Options	Default	Description	Classifier
-t –type	nn_rhorho nn_a1rho nn_a1a1 boosted_trees svm random_forest train_rhorhoZ	nn_rhorho	Type of trained classifier	NNs Boosted Trees SVM Random Forest
-l –layers		6	Number of hidden layers of NN	NNs
-s –size		100	Number of neurons in hidden layer	NNs
–lambda		0.0	Smearing parameter	NNs
-m –method	A B C	A	Method of calculating features	NNs Boosted Trees SVM Random Forest
-o –optimizer	GradientDescentOptimizer AdadeltaOptimizer AdagradOptimizer ProximalAdagradOptimizer AdamOptimizer FtrlOptimizer ProximalGradientDescentOptimizer RMSPropOptimizer	AdamOptimizer	NN optimiser	NNs
-i –input			Path to input data	NNs Boosted Trees SVM Random Forest
-d –dropout		0.2	Dropout for NN	NNs
-e –epochs		25	Number of epochs in NN training	NNs
-f –features	Model-Oracle Model-OnlyHad Model-Benchmark Model-1 Model-2 Model-3		Name of feature set to be used as an input	NNs Boosted Trees SVM Random Forest
–treedepth		5	Tree depth in Boosted Trees	Boosted Trees
–miniset	t / true / 1 / yes f / false / 0 / no	false	Use smaller training set of size 100.000	NNs Boosted Trees SVM Random Forest
–svm_c			SVM C parameter	SVM
–svm_gamma			SVM gamma parameter	SVM
–forest_max_feat	log2 sqrt	sqrt	Number of features considered when looking for the best split	Random Forest
–forest_max_depth		10	Max depth of tree in Random Forest	Random Forest
–forest_estimators		10	Number of trees in the forest	Random Forest
–unweighted	t / true / 1 / yes f / false / 0 / no	false	Unweighted data or original weights	Random Forest
–z_noise_fraction		0.5	Z boson noise fraction	NNs Boosted Trees SVM Random Forest

- train_dropout.sh
- train_features_a1a1.sh
- train_features_a1rho.sh
- train_features_rhorho.sh
- train_optimizers.sh
- train_rhorho_unweighted.sh
- train_features_noise_rhorhoZ.py
- train_rhorho_different_nn_structure.sh
- train_rhorho_lambda.sh
- train_boostedtrees_maxdepth.sh
- train_svm_gridsearch.sh
- train_randomforest_gridsearch.sh
- train_all_methods.sh

4.1 train_ABCmethods.sh

Used to compare results obtained with feature sets calculated with A, B or C method. Trains NN on RhoRho dataset with Model-1 features, dropout 0.2, 250 epochs and default NN structure.

4.2 train_dropout.sh

Used to compare results obtained with different dropout values (0.0 - 0.5, each 0.1). Trains NN on RhoRho dataset with Model-OnlyHad features, 250 epochs and default NN structure.

4.3 train_features_{a1a1, a1rho, rhorho}.sh

Used to compare results obtained with different feature sets (Model-Oracle, Model-OnlyHad, Model-Benchmark, Model-1, Model-2) on each of RhoRho, A1Rho, A1A1 datasets. Uses 250 epochs, 0.2 dropout and default NN structure.

4.4 train_optimizers.sh

Used to compare results obtained with different optimisers (GradientDescentOptimizer, AdadeltaOptimizer, AdagradOptimizer, ProximalAdagradOptimizer, AdamOptimizer, FtrlOptimizer, ProximalGradientDescentOptimizer, RMSPropOptimizer). Trains NN on RhoRho dataset with Model-1 features, dropout 0.2, 250 epochs and default NN structure.

4.5 train_rhorho_unweighted.sh

Used to train NN on different feature sets (Model-Oracle, Model-OnlyHad, Model-Benchmark, Model-1, Model-2) with unweighted data. Uses RhoRho dataset, 250 epochs, 0.2 dropout and default NN structure.

4.6 train_features_noise_rhorhoZ.py

Used to compare results obtained with different Z boson noise values (0.0 - 0.9, each 0.1) for different feature sets (Model-Oracle, Model-OnlyHad, Model-Benchmark, Model-1, Model-2). Trains NN on RhoRho dataset with Model-1 features, dropout 0.0, 250 epochs and default NN structure.

4.7 train_rhorho_different_nn_structure.sh

Used to compare results obtained with different NN structure. Tested number of hidden layers: 2-10, each 1, tested number of neurons in hidden layers: 20, 50 and 100-800, each 100. Uses RhoRho dataset, 0.2 dropout and 150 epochs.

4.8 train_rhorho_lambda.sh

Used to compare results obtained on RhoRho dataset with Model-3 feature set with different lambda smearing parameter. Uses 50 epochs, dropout 0.2, 6 hidden layers, 300 neurons each.

4.9 train_boostedtrees_maxdepth.sh

Used to compare results obtained on RhoRho dataset and Model-OnlyHad feature set with boosted trees with tree max depths of 3-20, each 1.

4.10 train_svm_gridsearch.sh

Used to compare results obtained on RhoRho dataset and Model-Oracle feature set with boosted trees with C and gamma parameter in range 10^{-3}, 10^{3} (logarithmic scale). Miniset (100.000 train examples) option is used.

4.11 train_randomforest_gridsearch.sh

Used to train Random Forest on RhoRho dataset with different feature sets (Model-Oracle, Model-OnlyHad, Model-Benchmark, Model-1, Model-2) and max depth of tree equal to number of features in given feature set. Number of estimators: 128, functions used to determine number of features considered while looking for the best split in tree: log2 and sqrt.

4.12 train_all_methods.sh

Used to compare NN, Boosted Trees, Random Forest and SVM on RhoRho dataset and each of feature sets (Model-Oracle, Model-OnlyHad, Model-Benchmark, Model-1, Model-2).
NN configuration: 250 epochs, dropout 0.2, 6 hidden layers, 300 neurons in each hidden layer.
Boosted Trees configuration: equal to number of features in feature set.
SVM configuration: C = 10, gamma = 0.1, miniset (100.000 training examples) option.
Random forest: max depth: 30, number of estimators: 300, funtion used to determine number of features used while looking for the best split: sqrt.

5 Plot scripts

5.1 plot_scripts/AUC_plots.py

Arguments of the script are presented in Table 2.

Table 2: AUC_plots.py script args

Flags	Description
-i –input	Paths to input data
-o –output	Output file name
-s –sizes	Ranges of x and y axes Order: x0 x1 y0 y1
-t –colors_t	Colors of training plots
-v –colors_v	Colors of validation plots
-x –legendloc	Localtion of legend Possible values as in Mathplotlib
-l –labels	Labels of data
–only_validation	Show only validation scores Options: true / t / yes / 1 / false / f / 0 / no

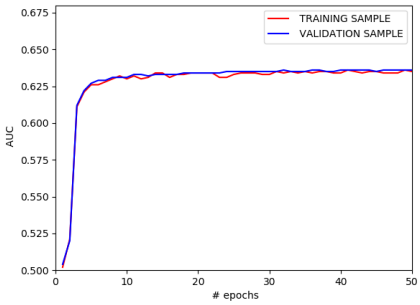
Example of usage:

```
~/anaconda2/envs/tensorflow/bin/python plot_scripts/AUC_plots.py
-i plot_scripts/ABC_methods/slurm-11559843_0.out
plot_scripts/ABC_methods/slurm-11559843_1.out
plot_scripts/ABC_methods/slurm-11559843_2.out
-o ABC_methods.png -v maroon olivegreen indigo -t b b b
-l "Method A" "Method B" "Method C" --only_validation yes
```

`-s 0 150 0.64 0.65 -x "lower right"`

Example of plot can be seen in Figure 5.1.

Figure 1: Example plot made with AUC_plots.py



5.2 `plot_scripts/boosted_trees_depth_plots.py`

Arguments of the script are presented in Table 3.

Table 3: boosted_trees_depth_plots.py script args

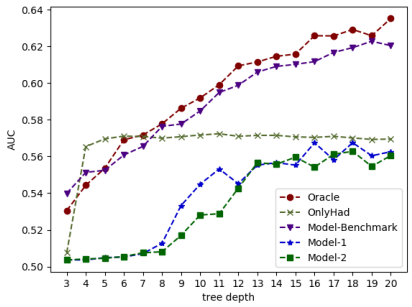
Flags	Description
-i –input	Paths to input data
-o –output	Output file name
-c –colors	Colors of plots
-l –labels	Labels of data
-m –markers	Markers of data

Example of usage:

```
~/anaconda2/envs/tensorflow/bin/python
plot_scripts/boosted_trees_depth_plots.py
-o boosted_trees_depth2.png -i plot_scripts/boosted_trees/Oracle
plot_scripts/boosted_trees/OnlyHad
 plot_scripts/boosted_trees/Model-Banchmark
plot_scripts/boosted_trees/Model-1 plot_scripts/boosted_trees/Model-2
-l Oracle OnlyHad Model-Benchmark Model-1 Model-2
-m o x v "*" s -c maroon darkolivegreen indigo mediumblue darkgreen
```

Example of plot can be seen in Figure 5.2.

Figure 2: Example plot made with boosted_trees_depth_plots.py



5.3 plot_scripts/lambda_plots.py

Arguments of the script are presented in Table 4.

Table 4: lambda_plots.py script args

Flags	Description
-i –input	Paths to input data
-o –output	Output file name
-c –colors	Colors of plots
-l –labels	Labels of data

Example of usage:

```
~/anaconda2/envs/tensorflow/bin/python plot_scripts/lambda_plots.py
-i plot_scripts/ABC_methods -o ABC_methods_plot.png -c r g b
-l "Method A" "Method B" "Method C"
```

Example of plot can be seen in Figure 5.3.

5.4 plot_scripts/Z_plots.py

Arguments of the script are presented in Table 5.

Example of usage:

```
~/anaconda2/envs/tensorflow/bin/python plot_scripts/Z_plots.py
-i plot_scripts/z_particle_50epochs
-c maroon darkolivegreen indigo mediumblue darkgreen
-o Z_noise.png -m o x v "*" s
```

Example of plot can be seen in Figure 5.4.

Figure 3: Example plot made with lambda_plots.py

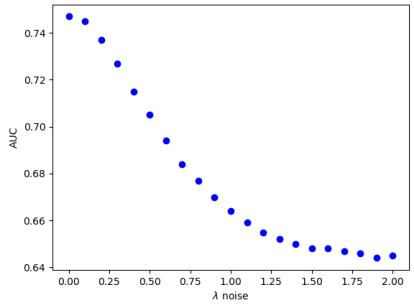


Table 5: Z_plots.py script args

Flags	Description
-i –input	Paths to input data
-o –output	Output file name
-c –colors	Colors of plots
-m –markers	Markers of data

5.5 `thesis-helpers/nn_sizes_heatmap.py`

Arguments of the script are presented in Table 6.

Table 6: nn_sizes_heatmap.py script args

Flags	Description
-i –input	Paths to input data
-o –output	Output file name

Example of usage:

```
~/anaconda2/envs/tensorflow/bin/python thesis-helpers/nn_sizes_heatmap.py
-i thesis-helpers/nn_structure_oracle_00
-o nnStructure_oracle_00.png
```

Example of plot can be seen in Figure 5.5.

5.6 `thesis-helpers/svm_finetuning_heatmap.py`

Arguments of the script are presented in Table 7.

Figure 4: Example plot made with Z_plots.py

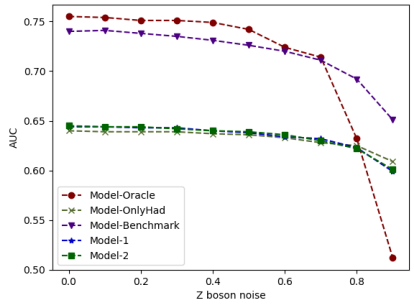


Figure 5: Example plot made with nn_sizes_heatmap.py

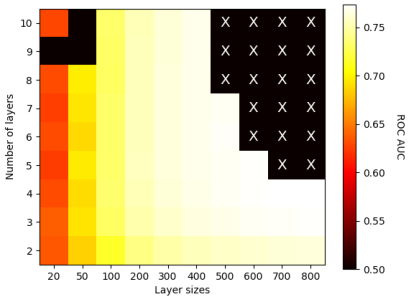


Table 7: svm_finetuning_heatmap.py script args

Flags	Description
-i –input	Paths to input data
-o –output	Output file name

Example of usage:

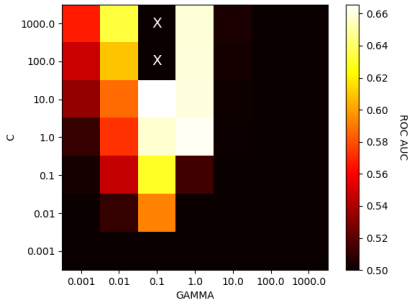
```
~/anaconda2/envs/tensorflow/bin/python thesis-helpers/svm_finetuning_heatmap.py
-i thesis-helpers/svm_finetuning/
-o thesis-helpers/svm_heatmap.png
```

Example of plot can be seen in Figure 5.6.

6 New ML techniques scripts

All new classifiers are run with the usage of `main.py` script and with providing appropriate parameters. The output files contain accuracy and ROC AUC score.

Figure 6: Example plot made with svm_finetuning_heatmap.py



6.1 `train_boostedtrees.py`

Classifier: XGBClassifier from xgboost library
Classifier parameters:

- max depth of tree

6.2 `train_randomforest.py`

Classifier: RandomForestClassifier from sklearn.ensemble library
Classifier parameters:

- max depth of tree
- number of estimators
- max number of features considered while looking for the best split

6.3 `train_svm.py`

Classifier: svm.SVC from sklearn library
Classifier parameters:

- C
- gamma

Comments: rbf kernel

7 Other python scripts - changes

Most of the scripts were changed to use environment variables set with setup file eg. for data paths and well as parameters passed from command line.
`a1a1.py`, `a1rho.py`, `rhorho.py` - refactored

`data_utils.py` - added "miniset" option
`data_utilsZ.py` - no changes
`download_a1a1.py`, `download_a1rho.py`, `download_rhorho.py`, `download_rhorhoZ.py` - refactored; script takes three arguments: 1) "-d"/"–datasets" - number of datasets to download with default value of 2, 2) "-o"/"–output" - directory in which the downloaded data should be saved, default value: "RHORHO_DATA" / "A1RHO_DATA" / "A1A1_DATA" environment variable value, 3) "-s"/"–source" - address from which data should be downloaded, default value: "DATA_SOURCE" environment variable value
`math_utils.py` - new functions and some common code extracted from `a1a1.py`, `a1rho.py`, `rhorho.py`
`particle.py` - no changes
`prepare_a1a1.py`, `prepare_a1rho.py`, `prepare_rhorho.py`, `prepare_Z_rhorho.py`, `prepare_utils.py` - refactored; script takes two arguments: 1) "-d"/"–datasets" - number of datasets to download with default value of 2, 2) "-i", "–input" - directory in which the downloaded data is saved, default value: "RHORHO_DATA" / "A1RHO_DATA" / "A1A1_DATA" environment variable value
`tf_model.py` - refactored
`train_a1a1.py`, `train_a1rho.py`, `train_rhorho.py`, `train_rhorhoZ.py` - refactored

8 Prometheus - step by step

8.1 Log on to Prometheus

1. Type `ssh plguser@prometheus.cyfronet.pl` in a terminal
2. Log in using the same password which is used for logging on plgrid webpage.

8.2 Prepare directories

1. Create project directory in main user folder, eg.:
   ```
   cd $HOME
   mkdir HiggsCP
   ```
2. Create data directories in Scratch partition, eg.:
   ```
   cd $SCRATCH
   mkdir higgs_data
   cd $higgs_data
   mkdir a1rho a1a1 rhorho
   ```

8.3 Setup Anaconda

1. Pick appropriate Anaconda2 Linux installer from https://repo.continuum.io/archive/ page, eg. Anaconda2-5.2.0-Linux-x86_64.sh

2. Download Anaconda installer to home directory on Prometheus
   ```
   cd $HOME
   wget https://repo.continuum.io/archive/Anaconda2-5.2.0-Linux-x86_64.sh
   ```

3. Change permission of the file
   ```
   chmod a+x Anaconda2-5.2.0-Linux-x86_64.sh
   ```

4. Run the script
   ```
   ./Anaconda2-5.2.0-Linux-x86_64.sh
   ```

5. Create new Anaconda environment called "tensorflow"
   ```
   cd anaconda2/bin/
   ./conda create --no-default-packages -n tensorflow python=2.7
   ```

6. Install any needed libraries, eg. for numpy:
   ```
   ./conda install --name tensorflow numpy
   ```

8.4 Copy project to Prometheus

1. In an other tab of terminal (personal computer), in the directory where the project ml_higgs is downloaded, type:
   ```
   scp ml_higgs.zip plguser@prometheus.cyfronet.pl:~/net/people/plguser/HiggsCP
   ```

2. From Prometheus terminal tab type:
   ```
   unzip $HOME/HiggsCP/ml_higgs.zip
   ```

8.5 Prepare setup file

1. Copy template to setup file
   ```
   cd $HOME/HiggsCP/ml_higgs
   cp setup_template slurm_scripts/setup
   ```

2. Edit contents of the file as described in 1.2
   ```
   vi slurm_scripts/setup
   ```

3. Load the variables
   ```
   source slurm_scripts/setup
   ```

8.6 Prepare data for training

1. Download data, eg. for one rhorho dataset:
   ```
   $ANACONDA_PYTHON_PATH/python2.7 download_a1rho.py -d 1
   ```

2. Prepare data, eg. for one rhorho dataset:
   ```
   $ANACONDA_PYTHON_PATH/python2.7 prepare_rhorho.py -i $RHORHO_DATA -d 1
   ```

8.7 Run Slurm scripts

1. Congratulations! The project is ready to be used! Scripts from slurm_scripts directory can be used by typing:
   `sbatch $WORKDIR/slurm_scripts/script_name.sh`