ProteoTorch Documentation

Release 0.1.0

John T. Halloran and Gregor Urban

Nov 25, 2020

CONTENTS

A Python package for deep learning and fast machine learning analysis of MS/MS database search results.

CHAPTER

ONE

KEY FEATURES

ProteoTorch accepts as input a Percolator INput (PIN) file containing target/decoy PSM features. By default, several iterations of deep semi-supervised learning are then performed to classify target and decoy PSMs, and the output PSM scores are recalibrated using the resulting learned parameters.

ProteoTorch provides the following semi-supervised machine learning classifiers:

- Deep neural networks
- Fast Linear SVMs using the L2-SVM-MFN algorithm (equivalent to the recently sped-up Percolator algorithm)
- Linear SVMs using the TRON algorithm (equivalent to these Percolator speedups)
- Linear Discriminant Analysis (+ Gaussian Mixture Models, in development)
- Support to easily swap in any supervised classifier implemented in Python which follow the design of scikit-learn clf object instances, with training function *fit* and testing function *decision_function*

CHAPTER

TWO

ADDITIONAL FEATURES

ProteoTorch provides an ultrafast q-value library (heavily optimized for Python), plotting tools to benchmark/compare MS/MS post-processor results, and an easy-to-use Python API for the MS/MS semi-supervised learning algorithm (with cross-validation) originally implemented in the C++ package Percolator.

CHAPTER

THREE

CONTENTS

3.1 Install

ProteoTorch may be downloaded and installed using:

```
git clone https://github.com/proteoTorch/proteoTorch.git
cd proteoTorch
python3 setup.py build
python3 setup.py install
```

3.1.1 Dependencies

ProteoTorch depends on the following packages:

- PyTorch
- scikit-learn
- numpy
- cython (optional, but highly recommended to build fully sped up q-value library)
- matplotlib (optional, but required for plotting utilities)

3.2 Quickstart

Here, we briefly describe post-processing a PIN file, **test.pin**, using a deep neural network (DNN) as the classifier during semi-supervised learning.

3.2.1 Recalibrating PSMs

After installation, the main ProteoTorch module, **analyze**, may be run from the command line using:

```
proteoTorch --pin test.pin --output_dir testOutput --method 3 --numThreads 10
```

In the above, *--method* specifies a DNN classifier, *--output_dir* specifies the directory to write results to, and *--numThreads* specifies the number of CPU threads to use for all parallelizable ProteoTorch computation (discussed further in the *next section*).

When finished, the recalibrated PSM scores will be written to the tab-delimited file *testOutput/output.txt*. The first few lines of an example output file are:

```
PSMId   score   q-value peptide Label   proteinIds
target_0_6395_3_1       0.999988        0.385129        K.
↪IDSAAETHADAPVVDASPAEDQASEVTEAPHVESAK.S        1       F52H3.7a
target_0_8821_3_1       0.999975        0.385129        R.LHCTAQPMPDGLADDIEGGTVNAR.D
↪  1       F25H5.4
target_0_10580_2_1      0.999968        0.385129        K.LANALEPGAVEVAAAEENADAAAQK.E
↪  1       C10G11.7
target_0_16058_2_1      0.999959        0.385129        K.TVDVISDTGTSFLGGPQSVVDGLAK.A
↪  1       F21F8.7
target_0_17089_2_1      0.999956        0.385129        K.HTDAVAELTDQLDQLNK.A   1
↪  F11C3.3
target_0_16837_2_1      0.999955        0.385129        K.HTDAVAELTDQLDQLNK.A   1
↪  F11C3.3
target_0_27067_3_1      0.999952        0.385129        R.
↪FQSSAVMALQEAAEAYLVGLFEDTNLCAIHAK.R    1       ZK131.3
target_0_15165_2_1      0.999945        0.385129        R.NLQIAQGTPGGLITYGAIDTVNCAK.Q
↪  1       Y39B6A.20
target_0_17414_2_1      0.999943        0.385129        K.VTLEDQIVQTNPVLEAFGNAK.T
↪  1       F11C3.3
```

Further analysis options are discussed in the *next section*.

3.2.2 Q-value analysis plots

Resulting PSM identifications vs q-values may be easily plotted and compared against other methods. Assuming **test.pin** has feature **XCorr** specified in its header, the following may be run on the command line to plot target-decoy competition (TDC) results for both recalibrated ProteoTorch scores and the uncalibrated XCorr scores:

```
proteoTorchPlot --output test.pdf --maxq 0.1  \
    --tdc --dataset test.pin \
    "ProteoTorch DNN":"score":output_dir/output.txt \
    "XCorr":'XCorr':test.pin
```

In the above, *--output* specifies the resulting plot file name (and format), *--maxq* specifies the maximum q-value threshold to plot, *--tdc* runs TDC for all methods, and tab-delimited PSM files are specified by the triple *method:header field:PSM file*. Note that, when TDC is specified, the original pin file must be specified using *--dataset* to properly calculate the *experimental mass*, *scan number* identifiers for each PSM.

Further details and examples are discussed on the *plotting page*.

3.3 Analysis options

The analysis source is provided in the module **proteoTorch.analyze**. Available options are listed below.

3.3.1 Main Options

The following is a list of post-processing options when calling **proteoTorch** from the command line.

- *--pin*: input file in PIN format
- *--method*: machine learning classifier to use during semi-supervised learning
  - Method 0: LDA
  - Method 1: linear SVM, solver TRON
  - Method 2: linear SVM, solver L2-SVM-MFN (Percolator's solver)
  - Method 3: DNN (deep multi-layer perceptron, *default value*)
- *--output_dir*: where to write result files. **Default = model_output/<data_file_name>/<time_stamp>/**
- *--tdc*: Use target-decoy competition to assign q-values (true/false). **Default = true/**
- *--numThreads*: Number of CPU threads to use for parallelizable computations. **Default = 1**)
- *--initDirection*: If >= 0, specifies which feature to use as initial PSM scores during semi-supervised learning. If = -1, automatically find and use the most discriminative feature. **Default = -1**
- *--q*: q-value tolerance when estimating positive training samples. **Default = 0.01**
- *--verbose*: Verbosity. **Default = 1**
- *--output_per_iter_granularity*: Specifies number of iterations to write recalibrated PSM scores. **Default = 5**
- *--write_output_per_iter*: Write recalibrated PSM scores after every *output_per_iter_granularity* iterations (true/false). **Default = true**
- *--maxIters*: Number of semi-supervised learning iterations to run. **Default = 20**
- *--seed*: Random seed when partitioning PSMs into cross-validation bins. **Default = 1**

3.3.2 Deep learning options

- *--dnn_optimizer*: DNN training algorithm to use (sgd or Adam). **Default = Adam**
- *--dnn_num_epochs*: Number of epochs to train DNN. **Default = 50**
- *--deepq*: DNN q-value tolerance when estimating positive training samples. **Default = 0.07**
- *--dnn_lr*: DNN learning rate. **Default = 0.001**
- *--dnn_lr_decay*: Reduce learning rate by this total for all epochs (dnn_lr_decay / dnn_num_epochs applied after each epoch). **Default = 0.02**
- *--dnn_num_layers*: Number of hidden DNN layers. **Default = 3**
- *--dnn_layer_size*: Number of neurons per hidden layer. **Default = 200**
- *--starting_dropout_rate*: Dropout rate for first iteration. **Default = 0.5**
- *--dnn_dropout_rate*: Dropout rate for iterations > 1. **Default = 0.0**

- *--dnn_gpu_id*: GPU ID to use for the DNN model (will switch to CPU mode if no GPU is found or CUDA is not installed). **Default = 0**
- *--dnn_label_smoothing_0*: Label smoothing for training class 0 (decoys). **Default = 0.99**
- *--dnn_label_smoothing_1*: Label smoothing for training class 1 (targets within q-value tolerance). **Default = 0.99**
- *--dnn_train_qtol*: AUC q-value tolerance to measure validation performance. **Default = 0.1**
- *--false_positive_loss_factor*: Multiplicative factor to weight false positives during training. **Default = 4.0**
- *--deepInitDirection*: Produce initial PSM scores by training a large ensemble DNN to speed up training convergence (true/false). **Default = true if *method*=3**
- *--deep_direction_ensemble*: Number of DNN ensembles to train during deep initial direction search. **Default = 30**
- *--load_previous_dnn*: Start iterations from previously saved model (boolean). **Default = false**
- *--previous_dnn_dir*: Previous output directory containing trained dnn weights.

3.3.3 Note on parallelization

Within each iteration of the algorithm, nested cross-validation (CV) is performed. If a DNN classifier is selected (i.e., *--method 3*), the CV folds are run sequentially. This safeguards against the GPU running out of memory and ProteoTorch crashing during analysis.

When an SVM is selected (i.e., *--method 2* or *--method 3*), the CV folds are run in parallel using the number of CPU threads specified by *--numThreads*.

3.4 Plot Utilities

3.4.1 Plotting # of identifications vs q-values

As discussed in the quickstart, *# identifications vs q-value* plots are available after installation by calling **proteoTorchPlot**. Options include

- *--output*: Output file name where the figure will be stored. **Default = figure.png**
- *--maxq*: Maximum q-value to plot to: 0 < q <= 1.0. **Default = 0.1**
- *--tdc*: Perform target-decoy competition (true/false). **Default = true**
- *--dataset*: PIN process which was analyze (only necessary if *tdc* = true).
- *--writeTdcResults*: Write the results of TDC for all methods to new files (true/false). **Default = false**
- *--tdcOutputDir*: Output directory to write TDC competition results. **Default = ‘’**
- *--publish*: Apply plot settings from ProteoTorch paper (true/false). **Default = false**

Furhter details are available in the source, module proteoTorch.plotQvals.

Specifying PSM score files

General tab-delimited files are passed to **proteoTorchPlot** as triples *method:score header field:PSM file*, where *method* specifies the legend name, *score header field* specifies the header column name to use as PSM scores, and *PSM file* is the tab-delimited file name. The input file must contain header fields:

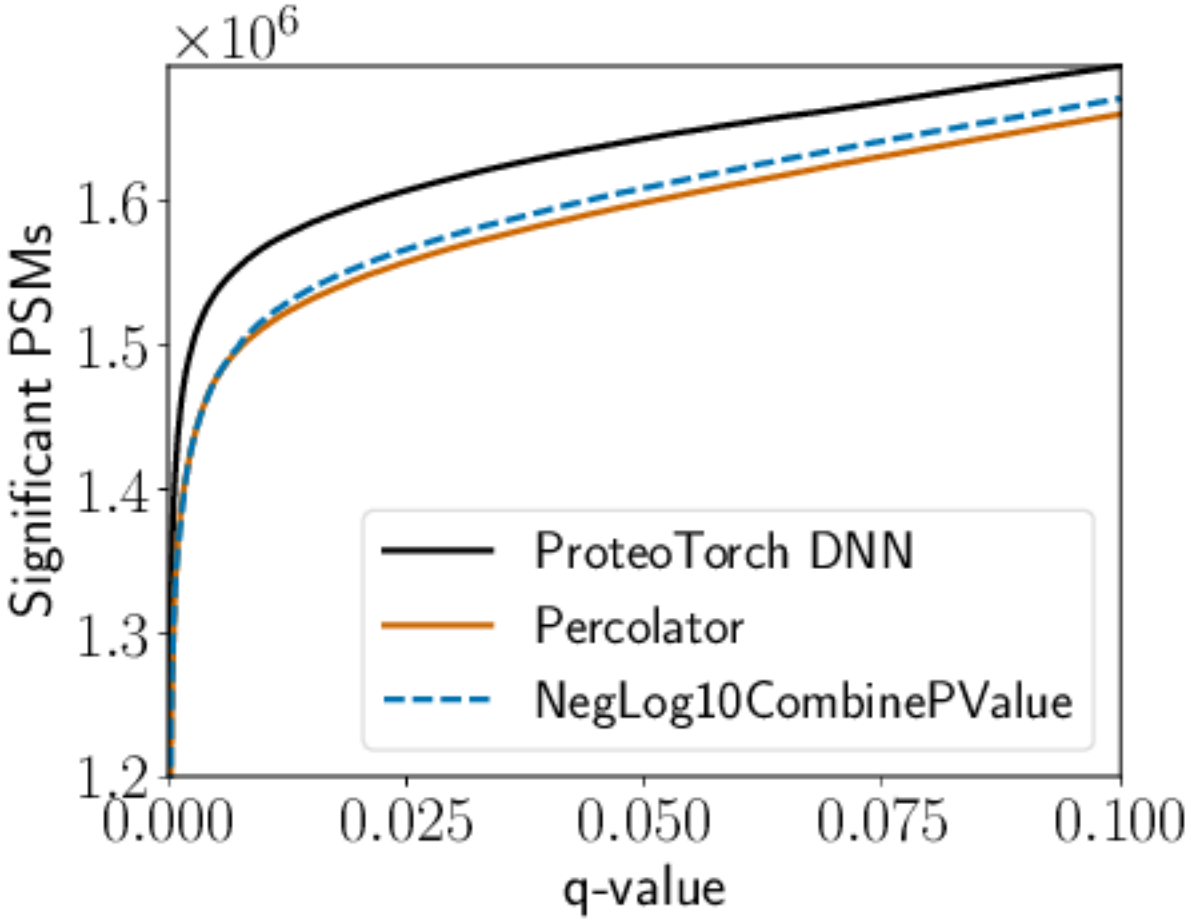
- *score header field* - column of PSM scores to compute q-values with
- *PSMId* - unique PSM IDs used in the analyzed PIN file
- *Label* - whether a PSM is a target (1) or decoy (-1)

If target and decoy results are separated into two tab-delimited files, these may be passed in as the a quartet *method: header field:target file: decoy file*. In this case, the column *Label* does not need to be specified.

As an example, the following plots ProteoTorch and Percolator recalibrated scores for PSMs collected searching a draft of the human proteome dataset using the recently developed high-res MS2 p-value score function, *residue-evidence combined p-value*:

```
proteoTorchPlot --output kim_resev.png --maxq 0.1  \
    --tdc true --dataset kim_resev.pin --publish true \
    "ProteoTorch DNN":"score":output_dir/output.txt \
    "Percolator":'score':kim_resev_percolator.targets.txt:kim_resev_percolator.decoys.
↪txt \
    "NegLog10CombinePValue":'NegLog10CombinePValue':kim_resev.pin
```

resulting in the plot below.



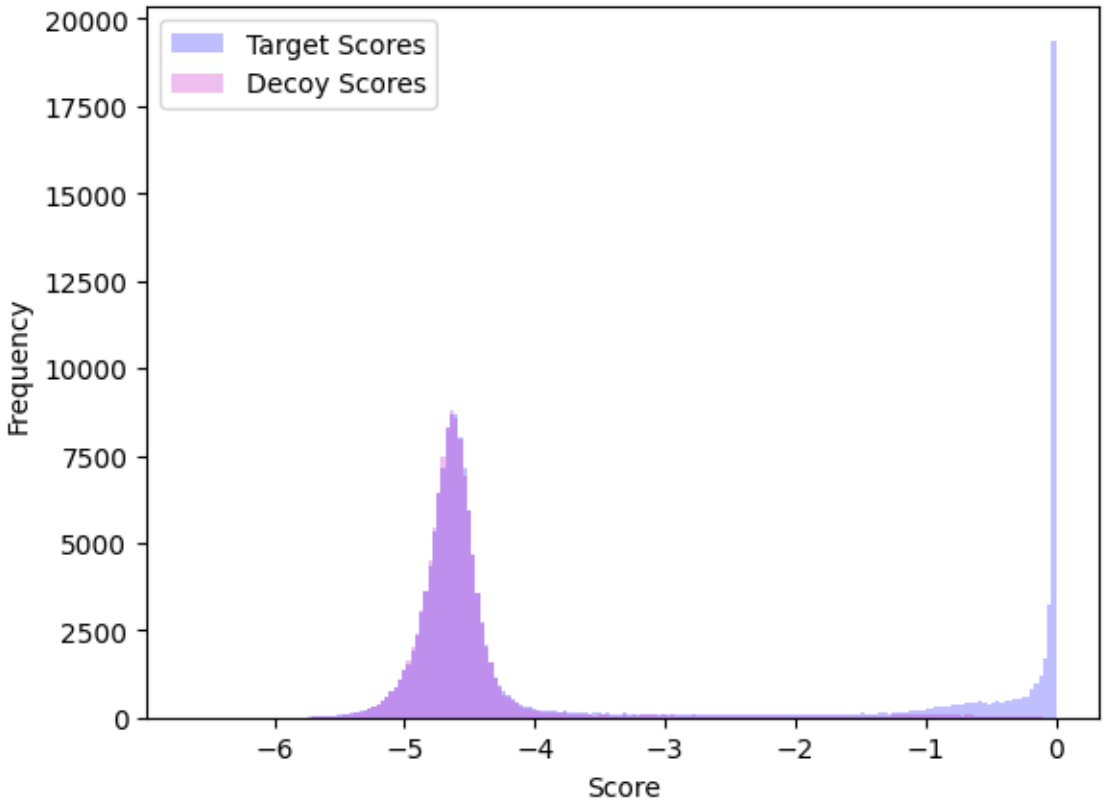
Note that, when TDC is specified, ProteoTorch generally uses the unique keys in header column **PSMId** to map PSMs to the (*experimental mass*, *scan number*) pairs specified in the original PIN file. However, if method is specified as one of {'PeptideProphet', 'Scavager', 'q-ranker'}, it is assumed that *experimental mass* and *scan number* are explicitly provided (with respective header fields **ExpMass** and **ScanNr**) due to potential nonconformity of the supplied **PSMId** with these methods.

3.4.2 Histograms

The following python script shows how to import proteoTorch plotting tools to quickly create a plot of recalibrated target and decoy scores. PSMs were originally collected searching a dataset of SARS-CoV-2 Proteins collected from COVID-19 patiens using the Comet search engine:

```
from math import log
from proteoTorch.plotQvals import load_pin_scores, histogram

filename= 'output_dir/output.txt'
scoreKey = 'score'

# load ProteoTorch output
scores, labels, _ = load_pin_scores(filename, scoreKey)

targets = [log(s) for s,l in zip(scores, labels) if l == 1]
decoys = [log(s) for s,l in zip(scores, labels) if l == -1]

output = 'ihling_comet_proteoTorchDnn_hist.png'

histogram(targets, decoys, output, bins = 200)
```

which generates the plot below.



3.5 Contact

ProteoTorch was developed by John Halloran (@UC Davis), Gregor Urban (@UCI), and Pierre Baldi (@UCI).

More information is available in the official github repo.

Please send questions to jthalloran@ucdavis.edu.

3.5.1 Citation

When using ProteoTorch please cite the following manuscript:

John Halloran, Gregor Urban, David Rocke, and Pierre Baldi. "Deep Semi-Supervised Learning Improves Universal Peptide Identification of Shotgun Proteomics Data." *bioRxiv* (2020) doi: 10.1101/2020.11.12.380881