Efficient Probit Regression

Release 0.1.0

Christian Peters & Samuel Toporovskij

Feb 06, 2022

API DOCUMENTATION

This is the documentation of our code base.

CHAPTER

ONE

EFFICIENT PROBIT REGRESSION API

Here you can find the documentation of most of our functions and classes.

1.1 Probit Model

class efficient_probit_regression.probit_model.**PGeneralizedProbitSGD**(*p*, *initial_learning_rate=0.1*, *power_t=0.5*)

Stochastic Gradient descent for probit regression. Adapts the learning rate in each iteration using inverse scaling.

Parameters

- **p** (*int*) – The order of the probit model.
- **initial_learning_rate** (*float*) – The initial learning rate.
- **power_t** (*float*) – Inverse scaling is used to adapt the learning rate in each iteration. The update formula is learning_rate = initial_learning_rate / power(cur_iteration, power_t)

get_params()

The function get_params() returns the estimated parameters. (revision)

new_sample(*x*, *y*)

Performs one step of SGD on a new sample x, y.

Parameters

- **x** (*numpy.ndarray*) –
- **y** (*int*) –

class efficient_probit_regression.probit_model.**ProbitSGD**(*initial_learning_rate=0.1*, *power_t=0.5*)

Parameters

- **initial_learning_rate** (*float*) –
- **power_t** (*float*) –

efficient_probit_regression.probit_model.**p_gen_norm_cdf**(*x*, *p*)

Returns the cumulative densitiy until a point x for p>= 1.

efficient_probit_regression.probit_model.**p_gen_norm_pdf**(*x*, *p*)

Returns the densitiy at a point x for p>= 1.

1.2 Sampling

class efficient_probit_regression.sampling.**ReservoirSampler**(*sample_size*, *d*)

Implementation of a reservoir sampler as described in "A general purpose unequal probability sampling plan" by M. T. Chao, adapted here for row sampling of datasets consisting of a data matrix X and a label vector y.

Parameters

- **sample_size** (*int*) – Number of rows in the resulting sample.
- **d** (*int*) – Second dimension of the sample. The whole sample will have a dimension of (sample_size, d).

get_sample()

Returns the sample of X and the sample of y.

insert_record(*row*, *label*, *weight*)

Insert a data record consisting of a row and a label. The record will be sampled with a probability that is proportional to the given weight.

Parameters

- **row** (*numpy.ndarray*) –
- **label** (*float*) –
- **weight** (*float*) –

efficient_probit_regression.sampling.**compute_leverage_scores**(*X*, *p=2*, *fast_approx=False*)

Computes leverage scores.

Parameters **X** (*numpy.ndarray*) –

efficient_probit_regression.sampling.**fast_QR**(*X*, *p=2*)

Returns Q of a fast QR decomposition of X.

efficient_probit_regression.sampling.**leverage_score_sampling**(*X*, *y*, *sample_size*, *augmented=False*, *online=False*, *round_up=False*, *precomputed_scores=None*, *p=2*, *fast_approx=False*)

Draw a leverage score weighted sample of X and y without replacement.

Parameters

- **X** (*numpy.ndarray*) – Data matrix.
- **y** (*numpy.ndarray*) – Label vector.
- **sample_size** (*int*) – Sample size.
- **augmented** (*bool*) – Whether to add the additive 1/W term, where W is the sum of all weights.
- **online** (*bool*) – Compute online leverage scores in one pass over the data.
- **round_up** (*bool*) – Round the leverage scores up to the nearest power of two.
- **precomputed_scores** (*Optional[numpy.ndarray]*) – To avoid recomputing the leverage scores every time, pass the precomputed scores here.
- **p** (*int*) – The order of the p-generalized probit model.
- **fast_approx** (*bool*) – Whether to use the fast leverage score approximation algorithm.

Returns

X_reduced The reduced data matrix.

y_reduced The reduced label vector.

w The corresponding sample weights.

efficient_probit_regression.sampling.**logit_sampling**(*X*, *y*, *sample_size*)
Logit sampling from 2018 Paper On Coresets for Logistic Regression.

Returns X_reduced, y_reduced, weights

Parameters

- **X** (*numpy.ndarray*) –
- **y** (*numpy.ndarray*) –
- **sample_size** (*int*) –

efficient_probit_regression.sampling.**online_ridge_leverage_score_sampling**(*X*, *y*, *sample_size*, *augmentation_constant=None*, *lambda_ridge=1e-06*)
Sample X and y proportional to the online ridge leverage scores.

Parameters

- **X** (*numpy.ndarray*) –
- **y** (*numpy.ndarray*) –
- **sample_size** (*int*) –
- **augmentation_constant** (*Optional[float]*) –
- **lambda_ridge** (*float*) –

efficient_probit_regression.sampling.**truncated_normal**(*a*, *b*, *mean*, *std*, *size*, *random_state=None*)
Use rejection sampling if the interval [a, b] covers at least a probability mass of 5%. Otherwise use the implementation given in scipy.stats.truncnorm.

The parameters a and b specify the actual interval where the probability mass is located, mean and std specify the original normal distribution.

Parameters

- **a** (*numpy.ndarray*) –
- **b** (*numpy.ndarray*) –
- **mean** (*numpy.ndarray*) –
- **std** (*numpy.ndarray*) –

efficient_probit_regression.sampling.**uniform_sampling**(*X*, *y*, *sample_size*)
Draw a uniform sample of X and y without replacement.

Parameters

- **X** (*numpy.ndarray*) – A matrix of size (n, d).
- **y** (*numpy.ndarray*) – A vector of size (n,).
- **sample_size** (*int*) – The size of the sample that should be drawn.

Returns The reduced sample (X_reduced, y_reduced).

1.3 Datasets

class efficient_probit_regression.datasets.**BaseDataset**(*add_intercept=True*, *use_caching=True*, *cache_dir=None*)

get_X()
Returns the data matrix from the data object.

get_beta_opt(*p*)
Returns the optimized/estimated parameters beta.

Parameters **p** (*int*) – the order of the p-generalized probit-model.

Returns returns the estimated parameters

get_d()
Returns the dimension of the data. d can also be regarded as the number of features.

get_n()
Returns the number of rows of the data.

get_y()
Returns the target data y.

class efficient_probit_regression.datasets.**Covertype**(*use_caching=True*)
Dataset Homepage: https://archive.ics.uci.edu/ml/datasets/Covertype

get_name()
Returns name of the data set.

class efficient_probit_regression.datasets.**Example2D**

get_name()
Returns name of data set.

load_X_y()
Loads data set and returns X and y.

class efficient_probit_regression.datasets.**Iris**(*use_caching=True*)

get_name()
" Returns name of data set.

load_X_y()
Loads data and returns X and y.

class efficient_probit_regression.datasets.**KDDCup**(*use_caching=True*)
Dataset Homepage: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

get_name()
Returns name of data set.

class efficient_probit_regression.datasets.**Webspam**(*drop_sparse_columns=True*, *use_caching=True*)
Dataset Source: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#webspam

get_name()
Returns name of data set.

load_X_y()
Loads data and returns X and y.

efficient_probit_regression.datasets.**add_intercept**(*X*)
Adds intercept.

1.4 Experiments

class efficient_probit_regression.experiments.**BaseExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*)

Parameters

- **p** (*int*) –
- **num_runs** (*int*) –
- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **results_filename** (*str*) –

get_config_grid()
Returns a list of configurations that are used to run the experiments.

abstract get_reduced_X_y_weights(*config*)
Abstract method that each experiment overrides to return the reduced matrix X, label vector y and weights that correspond to an experimental config.

optimize(*X*, *y*, *w*)
Optimize the Probit regression problem given by X, y and w.

Parameters

- **X** – Data matrix.
- **y** – Label vector.
- **w** – Weights.

run(*parallel=False*, *n_jobs=4*)
Run the experiment.

Parallel = False parallel is set to False by default.

N_jobs = 4 n_jobs is set to 4 by default.

class efficient_probit_regression.experiments.**BaseExperimentBayes**(*dataset*, *num_runs*, *min_size*, *max_size*, *step_size*, *prior_mean*, *prior_cov*, *samples_per_chain*, *num_chains*, *burn_in*)

Parameters

- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **num_runs** (*int*) –
- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **prior_mean** (*numpy.ndarray*) –

- **prior_cov** (*numpy.ndarray*) –
- **samples_per_chain** (*int*) –
- **num_chains** (*int*) –
- **burn_in** (*int*) –

abstract get_method_name()
Returns the name of the method, like "uniform", "leverage", or "leverage_online".

class efficient_probit_regression.experiments.**LeverageScoreSamplingExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*, *only_compute_once=True*, *online=False*, *round_up=True*, *fast_approx=False*)

Parameters **dataset** (efficient_probit_regression.datasets.BaseDataset) –

get_reduced_X_y_weights(*config*)
Returns reduced X, y and weights.

run(***kwargs*)
Run the experiment.

Parallel = False parallel is set to False by default.

N_jobs = 4 n_jobs is set to 4 by default.

class efficient_probit_regression.experiments.**LeverageScoreSamplingExperimentBayes**(*dataset*, *num_runs*, *min_size*, *max_size*, *step_size*, *prior_mean*, *prior_cov*, *samples_per_chain*, *num_chains*, *burn_in*)

Parameters

- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **num_runs** (*int*) –
- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **prior_mean** (*numpy.ndarray*) –
- **prior_cov** (*numpy.ndarray*) –

- **samples_per_chain** (*int*) –
- **num_chains** (*int*) –
- **burn_in** (*int*) –

get_method_name()
Returns the name of the method, like "uniform", "leverage", or "leverage_online".

class efficient_probit_regression.experiments.**LewisSamplingExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*, *fast_approx=True*)

Parameters **dataset** (efficient_probit_regression.datasets.BaseDataset) –

get_reduced_X_y_weights(*config*)
Returns reduced X, y and weights.

class efficient_probit_regression.experiments.**LogitSamplingExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*)

Returns reduced X, y and weights.

Parameters

- **p** (*int*) –
- **num_runs** (*int*) –
- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **results_filename** (*str*) –

get_reduced_X_y_weights(*config*)
Abstract method that each experiment overrides to return the reduced matrix X, label vector y and weights that correspond to an experimental config.

class efficient_probit_regression.experiments.**OnlineLeverageScoreSamplingExperimentBayes**(*dataset*, *num_runs*, *min_size*, *max_size*, *step_size*, *prior_mean*, *prior_cov*, *samples_per_chain*, *num_chains*, *burn_in*)

Parameters

- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **num_runs** (*int*) –

- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **prior_mean** (*numpy.ndarray*) –
- **prior_cov** (*numpy.ndarray*) –
- **samples_per_chain** (*int*) –
- **num_chains** (*int*) –
- **burn_in** (*int*) –

get_method_name()
Returns the name of the method, like "uniform", "leverage", or "leverage_online".

class efficient_probit_regression.experiments.**OnlineRidgeLeverageScoreSamplingExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*)

Parameters **dataset** (efficient_probit_regression.datasets.BaseDataset) –

get_reduced_X_y_weights(*config*)
Returns reduced X, y and weights.

class efficient_probit_regression.experiments.**SGDExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*)

Parameters **dataset** (efficient_probit_regression.datasets.BaseDataset) –

get_reduced_X_y_weights(*config*)
In SGD, no reduction is performed.

optimize(*X*, *y*, *w*)
Applies SGD in one pass over the data.

class efficient_probit_regression.experiments.**UniformSamplingExperiment**(*p*, *num_runs*, *min_size*, *max_size*, *step_size*, *dataset*, *results_filename*)

Parameters **dataset** (efficient_probit_regression.datasets.BaseDataset) –

get_reduced_X_y_weights(*config*)
Reduces the sample weights and returns them.

class efficient_probit_regression.experiments.**UniformSamplingExperimentBayes**(*dataset*, *num_runs*, *min_size*, *max_size*, *step_size*, *prior_mean*, *prior_cov*, *samples_per_chain*, *num_chains*, *burn_in*)

Parameters

- **dataset** (efficient_probit_regression.datasets.BaseDataset) –
- **num_runs** (*int*) –
- **min_size** (*int*) –
- **max_size** (*int*) –
- **step_size** (*int*) –
- **prior_mean** (*numpy.ndarray*) –
- **prior_cov** (*numpy.ndarray*) –
- **samples_per_chain** (*int*) –
- **num_chains** (*int*) –
- **burn_in** (*int*) –

get_method_name()

Returns the name of the method, like "uniform", "leverage", or "leverage_online".

1.5 Settings

PYTHON MODULE INDEX

e

INDEX

W