sklearn.grid_search.RandomizedSearchCV

« *class* `sklearn.grid_search.`**`RandomizedSearchCV`**(*estimator*, *param_distributions*, *n_iter=10*, *scoring=None*, *fit_params=None*, *n_jobs=1*, *iid=True*, *refit=True*, *cv=None*, *verbose=0*, *pre_dispatch='2*n_jobs'*, *random_state=None*, *error_score='raise'*) [source]

Randomized search on hyper parameters.

RandomizedSearchCV implements a “fit” and a “score” method. It also implements “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.

If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

Read more in the User Guide.

Parameters: **estimator** : estimator object.

A object of that type is instantiated for each grid point. This is assumed to implement the scikit-learn estimator interface. Either estimator needs to provide a `score` function, or `scoring` must be passed.

Previous

param_distributions : dict

Dictionary with parameters names (string) as keys and distributions or lists of parameters to try. Distributions must provide a `rvs` method for sampling (such as those from scipy.stats.distributions). If a list is given, it is sampled uniformly.

n_iter : int, default=10

Number of parameter settings that are sampled. n_iter trades off runtime vs quality of the solution.

«

scoring : string, callable or None, default=None

A string (see model evaluation documentation) or a scorer callable object / function with signature `scorer(estimator, X, y)`. If `None`, the `score` method of the estimator is used.

fit_params : dict, optional

Parameters to pass to the fit method.

n_jobs : int, default=1

Number of jobs to run in parallel.

pre_dispatch : int, or string, optional

Controls the number of jobs that get dispatched during parallel execution. Reducing this number can be useful to avoid an explosion of memory consumption when more jobs get dispatched than CPUs can process. This parameter can be:

- None, in which case all the jobs are immediately created and spawned. Use this for lightweight and fast-running jobs, to avoid delays due to on-demand spawning of the jobs
- An int, giving the exact number of total jobs that are spawned
- A string, giving an expression as a function of n_jobs, as in ‘2*n_jobs’

Previous

iid : boolean, default=True

If True, the data is assumed to be identically distributed across the folds, and the loss minimized is the total loss per sample, and not the mean loss across the folds.

cv : int, cross-validation generator or an iterable, optional

Determines the cross-validation splitting strategy. Possible inputs for cv are:

- None, to use the default 3-fold cross-validation,
- integer, to specify the number of folds.
- An object to be used as a cross-validation generator.
- An iterable yielding train/test splits.

For integer/None inputs, if `y` is binary or multiclass, **`StratifiedKFold`** used. If the estimator is a classifier or if `y` is neither binary nor multiclass, **`KFold`** is used.

Refer User Guide for the various cross-validation strategies that can be used here.

refit : boolean, default=True

Refit the best estimator with the entire dataset. If “False”, it is impossible to make predictions using this RandomizedSearchCV instance after fitting.

verbose : integer

Controls the verbosity: the higher, the more messages.

random_state : int or RandomState

Pseudo random number generator state used for random uniform sampling from lists of possible values instead of scipy.stats distributions.

error_score : ‘raise’ (default) or numeric

Value to assign to the score if an error occurs in estimator fitting. If set to ‘raise’, the error is raised. If a numeric value is given, FitFailedWarning is raised. This parameter does not affect the refit step, which will always raise the error.

Attributes:

grid_scores_ : list of named tuples

Contains scores for all parameter combinations in param_grid. Each entry corresponds to one parameter setting. Each named tuple has the attributes:

- `parameters`, a dict of parameter settings
- `mean_validation_score`, the mean score over the cross-validation folds
- `cv_validation_scores`, the list of scores for each fold

best_estimator_ : estimator

Estimator that was chosen by the search, i.e. estimator which gave highest score (or smallest loss if specified) on the left out data. Not available if refit=False.

best_score_ : float

Score of best_estimator on the left out data.

best_params_ : dict

Parameter setting that gave the best results on the hold out data.

See also:

`GridSearchCV`

Does exhaustive search over a grid of parameters.

`ParameterSampler`

A generator over parameter settins, constructed from param_distributions.

Notes

The parameters selected are those that maximize the score of the held-out data, according to the scoring parameter.

If *n_jobs* was set to a value higher than one, the data is copied for each parameter setting(and not *n_jobs* times). This is done for efficiency reasons if individual jobs take very little time, but may raise errors if the dataset is large and not enough memory is available. A workaround in this case is to set *pre_dispatch*. Then, the memory is copied only *pre_dispatch* many times. A reasonable value for *pre_dispatch* is *2 * n_jobs*.

Methods

`decision_function`(X)	Call decision_function on the estimator with the best found parameters.
`fit`(X[, y])	Run fit on the estimator with randomly drawn parameters.
`get_params`([deep])	Get parameters for this estimator.
`inverse_transform`(Xt)	Call inverse_transform on the estimator with the best found parameters.
`predict`(X)	Call predict on the estimator with the best found parameters.
`predict_log_proba`(X)	Call predict_log_proba on the estimator with the best found parameters.
`predict_proba`(X)	Call predict_proba on the estimator with the best found parameters.
`score`(X[, y])	Returns the score on the given data, if the estimator has been refit.
`set_params`(**params)	Set the parameters of this estimator.
`transform`(X)	Call transform on the estimator with the best found parameters.

«

__init__(*estimator*, *param_distributions*, *n_iter=10*, *scoring=None*, *fit_params=None*, *n_jobs=1*, *iid=True*, *refit=True*, *cv=None*, *verbose=0*, *pre_dispatch='2*n_jobs'*, *random_state=None*, *error_score='raise'*) [source]

decision_function(*X*) [source]

Call decision_function on the estimator with the best found parameters.

Only available if `refit=True` and the underlying estimator supports `decision_function`.

Parameters:	**X** : indexable, length n_samples Must fulfill the input assumptions of the underlying estimator.

Previous

`fit`(*X*, *y=None*) [source]

Run fit on the estimator with randomly drawn parameters.

Parameters: **X** : array-like, shape = [n_samples, n_features]

Training vector, where n_samples in the number of samples and n_features is the number of features.

y : array-like, shape = [n_samples] or [n_samples, n_output], optional

Target relative to X for classification or regression; None for unsupervised learning.

«

`get_params`(*deep=True*) [source]

Get parameters for this estimator.

Parameters: **deep: boolean, optional** :

If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns: **params** : mapping of string to any

Parameter names mapped to their values.

`inverse_transform`(*Xt*) [source]

Call inverse_transform on the estimator with the best found parameters.

Only available if the underlying estimator implements `inverse_transform` and `refit=True`.

Parameters: **Xt** : indexable, length n_samples

Previous

Must fulfill the input assumptions of the underlying estimator.

`predict`(*X*) [source]

Call predict on the estimator with the best found parameters.

Only available if `refit=True` and the underlying estimator supports `predict`.

Parameters: **X** : indexable, length n_samples

Must fulfill the input assumptions of the underlying estimator.

«

`predict_log_proba`(*X*) [source]

Call predict_log_proba on the estimator with the best found parameters.

Only available if `refit=True` and the underlying estimator supports `predict_log_proba`.

Parameters: **X** : indexable, length n_samples

Must fulfill the input assumptions of the underlying estimator.

`predict_proba`(*X*) [source]

Call predict_proba on the estimator with the best found parameters.

Only available if `refit=True` and the underlying estimator supports `predict_proba`.

Parameters: **X** : indexable, length n_samples

Must fulfill the input assumptions of the underlying estimator.

Previous

score(*X*, *y=None*) [source]

Returns the score on the given data, if the estimator has been refit.

This uses the score defined by `scoring` where provided, and the `best_estimator_.score` method otherwise.

Parameters:	**X** : array-like, shape = [n_samples, n_features] Input data, where n_samples is the number of samples and n_features is the number of features. **y** : array-like, shape = [n_samples] or [n_samples, n_output], optional Target relative to X for classification or regression; None for unsupervised learning.
Returns:	**score** : float

«

Notes

- The long-standing behavior of this method changed in version 0.16.
- It no longer uses the metric provided by `estimator.score` if the `scoring` parameter was set when fitting.

set_params(***params*) [source]

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The former have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns:	**self** :

transform(*X*) [source]

Previous

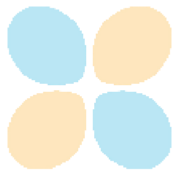
Call transform on the estimator with the best found parameters.

Only available if the underlying estimator supports `transform` and `refit=True`.

Parameters:	**X** : indexable, length n_samples Must fulfill the input assumptions of the underlying estimator.

Examples using `sklearn.grid_search.RandomizedSearchCV`

«



Comparing randomized search and grid search for hyperparameter estimation

Previous