

Peak Engines

Contents

Module peak_engines	1
Sub-modules	1
Classes	1
Class RidgeRegressionModel	1
Parameters	1
Examples	2
Instance variables	2
Methods	2
Class WarpedLinearRegressionModel	3
Parameters	3
Examples	3
Instance variables	3
Methods	3
Class Warper	4
Methods	4
Module peak_engines.peak_engines_impl	4
Functions	4
Function RidgeRegressionModel	4
Function WarpedLinearRegressionModel	5

Module peak_engines

Sub-modules

- [peak_engines.peak_engines_impl](#)

Classes

Class RidgeRegressionModel

```
class RidgeRegressionModel(init0=None, fit_intercept=True, normalize=True,
    score='loocv', grouping_mode='all', num_groups=0, grouper=None, tolerance=0.0001)
```

Implements regularized regression with regularizers fit so as to maximize the performance on the specified cross-validation metric.

Parameters

init0 : object, default=None Functor that can be used to change the starting parameters of the optimizer.

fit_intercept : bool, default=True Whether to center the target values and feature matrix columns.

normalize : bool, default=True Whether to rescale the target vector and feature matrix columns.

score : {'loocv', 'gcv'}, default='loocv' Cross-validation metric to use when fitting regularization parameters:

- 'loocv' will fit regularization parameters so as to maximize the leave-one-out cross-validation
- 'gcv' will fit regularization parameters so as to maximize the generalized cross-validation

grouping_mode : {'all', 'none'}, **default**='all' How to group regularization parameters:

- 'all' will use a single regularization parameter for all regressors.
- 'none' will use a separate regularization parameter for each regressor.

num_groups : **int**, **default**=0 If greater than zero, partition regressors and assign regressors of similar magnitude to the same regularizer.

grouper : **object**, **default**=None Customize how regularization parameters are grouped.

tolerance : **float**, **default**=0.0001 The tolerance for the optimizer to use when deciding to stop the objective. With a lower value, the optimizer will be more stringent when deciding whether to stop searching.

Examples

```
>>> from sklearn.datasets import load_boston
>>> from peak_engines import RidgeRegressionModel
>>> X, y = load_boston(return_X_y=True)
>>> model = RidgeRegressionModel().fit(X, y)
           # Default to Leave-one-out CV with a single regularizer
>>> model = RidgeRegressionModel(grouping_mode='none').fit(X, y)
           # Use separate regularizers for each regressor
>>> model = RidgeRegressionModel(num_groups=2).fit(X, y)
           # Use two regularizers and assign regressors of similar magnitude to the same
           # regularizer
>>> model = RidgeRegressionModel(
           grouper=lambda X, y: [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]).fit(X, y)
           # Use two regularizers: one for the first three variables; and one for the rest
```

Instance variables

Variable **coef_**

Return the regression coefficients.

Variable **regularization_**

Return the fitted regularization parameters.

Variable **within_tolerance_**

Return True if the optimizer found parameters within the provided tolerance.

Methods

Method **fit**

```
def fit(self, X, y)
```

Fit the ridge regression model.

Method **get_params**

```
def get_params(self, deep=True)
```

Get parameters for this estimator.

Method **predict**

```
def predict(self, X_test)
```

Predict target values.

Method `set_params`

```
def set_params(self, **parameters)
```

Set parameters for this estimator.

Class `WarpedLinearRegressionModel`

```
class WarpedLinearRegressionModel(init0=None, fit_intercept=True, normalize=True,
    num_steps=1, tolerance=0.0001)
```

Warped linear regression model fit so as to maximize likelihood.

Parameters

init0 : object, default=None Functor that can be used to change the starting parameters of the optimizer.

fit_intercept : bool, default=True Whether to center the target values and feature matrix columns.

normalize : bool, default=True Whether to rescale the target vector and feature matrix columns.

num_steps : int, default=1 The number of components to use in the warping function. More components allows for the model to fit more complex warping functions but increases the chance of overfitting.

tolerance : float, default=0.0001 The tolerance for the optimizer to use when deciding to stop the objective. With a lower value, the optimizer will be more stringent when deciding whether to stop searching.

Examples

Instance variables

Variable `noise_stddev_`

Return the fitted noise standard deviation.

Variable `noise_variance_`

Return the fitted noise variance.

Variable `regressors_`

Return the regressors of the latent linear regression model.

Variable `warper_`

Return the warper associated with the model.

Variable `within_tolerance_`

Return True if the optimizer found parameters within the provided tolerance.

Methods

Method `fit`

```
def fit(self, X, y)
```

Fit the warped linear regression model.

Method `get_params`

```
def get_params(self, deep=True)
```

Get parameters for this estimator.

Method predict

```
def predict(self, X_test)
```

Predict target values.

Method predict_latent_with_stddev

```
def predict_latent_with_stddev(self, X_test)
```

Predict latent values along with the standard deviation of the error distribution.

Method predict_logpdf

```
def predict_logpdf(self, X_test)
```

Predict target values with a functor that returns the log-likelihood of given target values under the model's error distribution.

Method set_params

```
def set_params(self, **parameters)
```

Set parameters for this estimator.

Class Warper

```
class Warper(impl)
```

Warping functor for a dataset's target space.

Methods**Method compute_latent**

```
def compute_latent(self, y)
```

Compute the warped latent values for a given target vector.

Method compute_latent_with_derivative

```
def compute_latent_with_derivative(self, y)
```

Compute the warped latent values and derivatives for a given target vector.

Method invert

```
def invert(self, z)
```

Invert the warping transformation.

Module peak_engines.peak_engines_impl

Machine Learning Toolkit

Functions**Function RidgeRegressionModel**

```
def RidgeRegressionModel(...)
```

Constructs a ridge regression model

Function WarpedLinearRegressionModel

```
def WarpedLinearRegressionModel(...)
```

Constructs a warped linear regression model

Generated by *pdoc* 0.8.1 (<https://pdoc3.github.io>).