# keras_ssg_lasso Documentation

*Release 0.1*

**Romain Tavenard**

**Nov 23, 2017**

# Contents

**class** ssgl_classifiers.**SSGL_LogisticRegression**(*dim_input*, *n_classes*, *groups*, *indices_sparse*, *alpha=0.5*, *lbda=0.01*, *n_iter=500*, *batch_size=256*, *optimizer='sgd'*, *verbose=0*)

Semi-Sparse Group Lasso Logistic Regression classifier.

The loss function to minimize is:

$L(X, y, \beta) + (1 - \alpha)\lambda \sum_{l=1}^{m} \sqrt{p_l} \|\beta^l\|_2 + \alpha\lambda\|\beta\|_1$

where $L$ is the logistic loss and $p_l$ is the number of variables in group $l$.

**Parameters**

- **dim_input** (*int*) – Dimension of the input feature space.
- **n_classes** (*int*) – Number of classes for the classification problem.
- **groups** (*list of numpy arrays*) – Affiliation of input dimensions to groups. numpy array of shape *(dim_input, )*. Each group is defined by an integer, each input dimension is attributed to a group.
- **indices_sparse** (*array-like*) – numpy array of shape *(dim_input, )* in which a zero value means the corresponding input dimension should not be included in the per-dimension sparsity penalty and a one value means the corresponding input dimension should be included in the per-dimension sparsity penalty.
- **alpha** (*float in the range [0, 1], default 0.5*) – Relative importance of per-dimension sparsity with respect to group sparsity (parameter $\alpha$ in the optimization problem above).
- **lbda** (*float, default 0.01*) – Regularization parameter (parameter $\lambda$ in the optimization problem above).
- **n_iter** (*int, default 500*) – Number of training epochs for the gradient descent.
- **batch_size** (*int, default 256*) – Size of batches to be used during both training and test.
- **optimizer** (*Keras Optimizer, default "sgd"*) – Optimizer to be used at training time. See https://keras.io/optimizers/ for more details.
- **verbose** (*int, default 0*) – Verbose level to be used for keras model (0: silent, 1: verbose).

**weights_**
numpy.ndarray of shape *(dim_input, n_classes)* – Logistic Regression weights.

**biases_**
numpy.ndarray of shape *(n_classes, )* – Logistic Regression biases.

**fit**(*X*, *y*)
Learn Logistic Regression weights.

**Parameters**

- **X** (*array-like, shape=(n_samples, dim_input)*) – Training samples.
- **y** (*array-like, shape=(n_samples, n_classes)*) – Training labels (formatted as a binary matrix, as returned by a standard One Hot Encoder, see http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html for more details).

**fit_predict**(*X*, *y*)
Fit the model using X and y and then use the fitted model to predict X.

Utility function equivalent to calling fit and then predict on the same data.

**Parameters**

- **X** (*array-like, shape=(n_samples, dim_input)*) – Training samples.
- **y** (*array-like, shape=(n_samples, n_classes)*) – Training labels (formatted as a binary matrix, as returned by a standard One Hot Encoder, see http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html for more details).

**Returns** **labels** – Array of class indices.

**Return type** array, shape=(n_samples,)

**predict**(*X*)
Predict the class of samples in X.

**Parameters** **X** (*array-like, shape=(n_samples, dim_input)*) – Samples to predict.

**Returns** **labels** – Array of class indices.

**Return type** array, shape=(n_samples,)

**predict_probas**(*X*)
Predict the probability of each class for samples in X.

**Parameters** **X** (*array-like, shape=(n_samples, dim_input)*) – Samples to predict.

**Returns** **probas** – Array of class probabilities.

**Return type** array, shape=(n_samples, n_classes)

**class** ssgl_classifiers.**SSGL_MultiLayerPerceptron**(*dim_input*, *n_classes*, *hidden_layers*, *groups*, *indices_sparse*, *alpha=0.5*, *lbda=0.01*, *n_iter=500*, *batch_size=256*, *optimizer='sgd'*, *activation='relu'*, *verbose=0*)
Bases: *ssgl_classifiers.SSGL_LogisticRegression*

Semi-Sparse Group Lasso Multi Layer Perceptron classifier.

**Parameters**

- **dim_input** (*int*) – Dimension of the input feature space.
- **n_classes** (*int*) – Number of classes for the classification problem.
- **hidden_layers** (*tuple (or list) of ints*) – Number of neurons in the hidden layers.
- **groups** (*list of numpy arrays*) – List of groups. Each group is defined by a numpy array of shape *(dim_input, )* in which a zero value means the corresponding input dimension is not included in the group and a one value means the corresponding input dimension is part of the group.
- **indices_sparse** (*array-like*) – numpy array of shape *(dim_input, )* in which a zero value means the corresponding input dimension should not be included in the per-dimension sparsity penalty and a one value means the corresponding input dimension should be included in the per-dimension sparsity penalty.

- **alpha** (*float in the range [0, 1], default 0.5*) – Relative importance of per-dimension sparsity with respect to group sparsity (parameter $\alpha$ in the optimization problem above).
- **lbda** (*float, default 0.01*) – Regularization parameter (parameter $\lambda$ in the optimization problem above).
- **n_iter** (*int, default 500*) – Number of training epochs for the gradient descent.
- **batch_size** (*int, default 256*) – Size of batches to be used during both training and test.
- **optimizer** (*Keras Optimizer, default "sgd"*) – Optimizer to be used at training time. See https://keras.io/optimizers/ for more details.
- **activation** (*Keras Activation function, default "relu"*) – Activation function to be used for hidden layers. See https://keras.io/activations/ for more details.
- **verbose** (*int, default 0*) – Verbose level to be used for keras model (0: silent, 1: verbose).

**weights_**
*list of arrays* – Multi Layer Perceptron weights.

**biases_**
*list of arrays* – Multi Layer Perceptron biases.

**class** ssgl_classifiers.**SSGL_WeightRegularizer**(*l1_reg=0.0*, *l2_reg=0.0*, *groups=None*, *indices_sparse=None*)

Bases: keras.regularizers.Regularizer

Semi-Sparse Group Lasso weight regularizer.

**Parameters**

- **l1_reg** (*float, default 0.*) – Per-dimension sparsity penalty parameter.
- **l2_reg** (*float, default 0.*) – Group sparsity penalty parameter.
- **groups** (*list of numpy arrays or None, default None.*) – List of groups. Each group is defined by a numpy array of shape *(dim_input, )* in which a zero value means the corresponding input dimension is not included in the group and a one value means the corresponding input dimension is part of the group. None means no group sparsity penalty groups numbering must starts at 0 with a continuous increment of 1 ([0,1,2,3...]). Features of the same group must be contiguous.
- **indices_sparse** (*array-like or None, default None.*) – numpy array of shape *(dim_input, )* in which a zero value means the corresponding input dimension should not be included in the per-dimension sparsity penalty and a one value means the corresponding input dimension should be included in the per-dimension sparsity penalty. None means no per-dimension sparsity penalty.

# Python Module Index

## S

# Index