lucidmode

Release 0.4.3.13

IFFranciscoME

Feb 13, 2022

CONTENTS

`lucidmode` is built using similar elements and structure as the API of *scikit-learn* and *TensorFlow*, so in order to preserve standard references like `fit`, `predict`, `predict_proba` and others.

Next, read some more details about What are the key elements of *Explainable Artificial Intelligence*, and what tools does `lucidmode` provide.

CHAPTER ONE

INTRODUCTION

This is an introduction to lucidmode . . . .

CHAPTER

TWO

INSTALLATION

The installation is straight forward, you can use `pip` and clone or dowload a particular version from `github` repository.

2.1 Using `pip`

You can install `lucidmode`, and automatically all the dependencies, using pip:

```
pip install lucidmode
```

2.2 Cloning from Github

For the latest development version, first get the source from Github:

```
git clone https://github.com/lucidmode/lucidmode.git
```

Then navigate into the local `lucidmode` directory and if you run the following line it will install the package and all its dependencies:

```
python setup.py install
```

Either option you choose, for the full use of the `lucidmode` package, you will need to have installed some depencies, all of them are listed in the `requirements.txt` file:

```
# -- Generic
pandas>=1.1.4
numpy>=1.20

# -- Object Description
rich>=9.5

# -- Visualization
jupyter>=1.0
plotly>=4.14

# -- Documentation
numpydoc
```

Those are just the lines with dependencies names and versions, you can check the full file Here

CHAPTER

THREE

MODELS

3.1 Multi-Layer Perceptron

`class lucidmode.models.NeuralNet`(*hidden_l*, *hidden_a*, *output_n*, *output_a*, *cost=None*, *hidden_r=None*, *output_r=None*, *optimizer=None*)

Artificial Neural Network: Feedforward multilayer pereceptron.

It supports a wide variations of topologies, from number of hidden layers, number of hidden neurons per layer, one input layer and one output layer where both of them could have from 1 to N number of neurons.

Parameters

hidden_l: list (of int) Number of neurons to include per hidden layer.

hidden_a: list (list of str, with length hidden_l) Activation of hidden layers

output_n: int Number of neurons in output layer

output_a: str Activation of output layer (str)

hidden_r / output_r: list (of str, of size l_hidden) list with each pre-layer weights and biases regularization criteria, options are:

- 'l1': Lasso regularization $|b|$
- 'l2': Ridge regularization $|b|^2$
- 'elasticnet': $C(L1 - L2)$
- 'dropout': Randomly (uniform) select N neurons in layer and turn its weight to 0

cost: str cost information for model.

- 'function': 'binary-logloss', 'multi-logloss', 'mse'
- 'reg': {'type': ['l1', 'l2', 'elasticnet'], 'lambda': 0.001, 'ratio': 0.01}

init: str initialization of weights specified from compile method

`fit`(*x_train*, *y_train*, *x_val=None*, *y_val=None*, *epochs=10*, *alpha=0.1*, *verbosity=3*, *random_state=1*, *callback=None*, *randomize=False*)

Train a previously specified (formed) model according to specified parameters.

Parameters

x_train: np.array / pd.Series Features data with nxm dimensions, n = observations, m = features

y_train: np.array / pd.Series Target variable data, dimensions of: nx1 por binary classification and nxm for multi-class

x_val: np.array / pd.Series Same as x_train but with data considered as validation

y_val: np.array / pd.Series Same as y_train but with data considered as validation

epochs: int Epochs to iterate the model training

alpha: float Learning rate for Gradient Descent

cost_f: str Cost function, options are according to functions

verbosity: int level of verbosity to show progress 3: cost train and cost val at every epoch

callback: dict whether there is a stopping criteria or action {'earlyStopping': {'metric': 'acc', 'threshold': 0.80}}

Returns

history: dict with dynamic keys and iterated values of selected metrics

`formation`(*cost=None*, *optimizer=None*, *init=None*, *metrics=None*)

Neural Network Model Formation.

Parameters

cost: dict Details of the cost function. Includes the following elements:

- 'cost_f': Cost function by its name, options are: {'logloss', 'mse'}
- 'cost_r': Cost regularization

optimizer: dict, str type: Name of method for optimization params: parameters according to method

init: weight initialization

metrics: metrics to monitor training

Returns

self: Modifications on instance of class

`init_weights`(*input_shape*, *init_layers*, *random_state=1*)

Weight initialization of a model that was previously instantiated by a topology formation process

Parameters

input_shape: int number of features (inputs) in the model

init_layers: list (of str, with size of n_layers) list with each layer criteria for weights initialization, with options:

- 'common-uniform': Commonly used factor & uniformly distributed random weights [1]
- 'xavier_uniform': Xavier factor & uniformly distributed random weights [1]
- 'xavier_normal': Xavier factor & standard-normally distributed random weights [1]
- 'he-standard': Factor formulatated according to [2]

References

- **[1]** X. Glorot and Y. Bengio, “Understanding the difficulty oftraining deep feedforward neural networks. International Conference on Artificial Intelligence and Statistics”, 2010.
- **[2]** He et al, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034, doi: 10.1109/ICCV.2015.123.

`inspect`(**params*)

Method for model inspection, which consists in a terminal print of the model topology and values through the use of the inspect method from the rich package for rich text and beautiful formatting in the terminal.

Parameters

params: list With the parameters to select which element to include in the console print, all the elements included in the list will be considered in conjunction, the options are the following:

- ‘help’: Show full help for the model
- ‘methods’: Show just callable methods
- ‘private-l1’ Priavate and layer 1 methods (beginning with single underscore)
- ‘private-l2’ Priavate and layer 2 methods (beginning with double underscore)

`predict`(*X*, *threshold=0.5*)

Computes a class or value prediction given the inherited model of the class.

Parameters

x_train: np.array Array with n-dimensional samples to generate the predictions from.

threshold: float Threshold value for the classification case. Default is 0.5

`predict_proba`(*X*)

Given the input samples, generates the class probability predictions for all the classes specified in the target variable. Inherits the model, hyperparameters and execution conditions from the class after the fit method is called.

3.2 Logistic Regression

`class lucidmode.models.LogisticRegression`(*penalty='elasticnet'*)

Logistic Regression model under construction …

Parameters

- ‘l1’: Lasso regularization [math:|*b*|]

- ‘l2’: Ridge regularization [math:|*b*|^2]

- ‘elasticnet’: [math:*C(L1 - L2)*]

CHAPTER FOUR

FUNCTIONS

`lucidmode` requires … .

4.1 Cost functions

`lucidmode.functions.cost`(*Y_hat*, *Y*, *type*)

Cost functions

Parameters

Y_hat: np.array Predicted values

Y: np.array Ground truth or real values

type: str One of the following options:

- 'sse': sum of squared errors
- 'mse': mean of squared errors
- 'binary-logloss': binary cross-entropy
- 'multi-logloss': multi-class cross-entropy

Returns

cost: np.float32

The binary cross-entropy or logloss cost function was utilized for both of the implemented models.

$$J(w) = -\frac{1}{m} \sum_{i=1}^{m} \big[y_i \; log(p_i) + (1 - y_i) \; log(1 - p_i) \big]$$

where:

- m: Number of samples.
- w: Model weights.
- y_i: The *i-th* ground truth (observed) output.
- p_i: The *i-th* probabilistically forecasted output.

4.2 Metrics

`lucidmode.tools.metrics.metrics`(*y*, *y_hat*, *type*, *use='learning'*)

Statistical and performance metrics for regression and classification, for single class One-Vs-One, for multiclass One-Vs-Rest.

Parameters

y: np.array Ground truth data

y_hat: np.array Predicted data

type: str The type of model is going to be tested. The options are: ‘classification’, ‘regression’

use: str

- ‘learning’: To measure performance of models in the learning process
- ‘information’: To measure information aspects for generalization goals

CHAPTER

FIVE

VISUALIZATIONS

`lucidmode` provides also tools for data and model visualizations.

5.1 OHLC Class

`lucidmode.tools.visualizations.`**`ohlc_class`**(*p_ohlc*, *p_theme*, *p_data_class*, *p_vlines*)

OHLC Candlestick plot with color indicator of class prediction success or failure.

Parameters

p_ohlc: pd.DataFrame, dict With OHLC Price data Open, Hight, Low, Close for one particular time period

p_theme: dict, optional Aesthetics and labels for the plot

p_data_class: array, list With the correct class, so a visual distinction will be made if prediction is correct or incorrect

p_vlines: list, optional With timestamp values to generate vertical lines at those values.

Returns

plot_ohlc_class: plotly A plotly object to use in a .show() or iplot(), plot()

CHAPTER

SIX

EXAMPLES

`lucidmode` requires … s.

CHAPTER

SEVEN

ROADMAP

lucidmode requires . . . .

lucidmode A Lucid Framework for Interpretable Machine Learning Models	
Author:	IFFranciscoME - if.francisco.me@gmail.com
Version:	**v0.4.1-beta1.0**
License:	GPL-3.0 License.
Repository:	https://github.com/lucidmode/lucidmode

7.1 Understanding Versions

https://semver.org/

CHAPTER

EIGHT

RELEASE HISTORY

8.1 v0.4-beta1.0

Calculation of several metrics for classification

sensitivity (TPR), specificity (TNR), accuracy (acc), likelihood ratio (positive), likelihood ratio (negative), confusion matrix (binary and multiclass), confusion tensor (binary for every class in multi-class)

Sequential Class

- Move the cost_f and cost_r parameters to be specified from formation method, leave the class instantiation with just the model architecture.
- Move the init_weights method to be specified from formation method.

Execution

- Create formation method in the Sequential Class, with the following parameters init, cost, metrics, optimizer.
- Store selected metrics in Train and Validation History

Visualizations

- Select metrics for verbose output.

8.2 v0.3-beta1.0

Regularization

- L1, L2 and ElasticNet on weights and biases, location: gradients
- L1, L2 and ElasticNet on weights and biases, location: cost function

Numerical Stability

- in functions.py, in cost, added a 1e-25 value to A, to avoid a divide by zero and invalid multiply cases in computations of np.log(A)

Data Handling

- train and validation cost

Visualization

- print: verbose of cost evolution

Documentation

- Improve README

8.3 v0.2-beta1.0

Files

- complete data set: MNIST
- complete data set: 'fashion-MNIST'

Tests passed

- fashion MNIST
- previous release tests

Topology

- single hidden layer (tested)
- 1 - 2 hidden layers (tested)
- different activation functions among hidden layer

Activation functions

- For hidden -> Sigmoid, Tanh, ReLU (tested and not working)
- For output -> Softmax

Cost Functions

- 'binary-logloss' (Binary-class Cross-Entropy)
- 'multi-logloss' (Multi-class Cross-Entropy)

Metrics

- Confusion matrix (Multi-class)
- Accuracy (Multi-class)

8.4 v0.1-beta1.0

Tests passed

- Random XOR data classification

Sequential model

- hidden_l: Number of neurons per hidden layer (list of int, with length of l_hidden)
- hidden_a: Activation of hidden layers (list of str, with length l_hidden)
- output_n: Number of neurons in output layer (1)
- output_a: Activation of output layer (str)

Layer transformations

- linear

Activation functions

- For hidden -> Sigmoid, Tanh
- For output -> Sigmoid (Binary)

Weights Initialization

- Xavier normal, Xavier uniform, common uniform, according to [1]

Training Schemes

- Gradient Descent

Cost Functions

- Sum of Squared Error (SSE) or Residual Sum of Squares (RSS)

Metrics

- Accuracy (Binary)

INDEX