MiraPy

Release v0.1.0

Swapnil Sharma, Akhil Singhal

Jul 11, 2019

CONTENTS

MiraPy

MiraPy is a Python package for Deep Learning in Astronomy. It is built using Keras for developing ML models to run on CPU and GPU seamlessly. The aim is to make applying machine learning techniques on astronomical data easy for astronomers, researchers and students.

Github repository: mirapy-org/mirapy

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Introduction

MiraPy is a Python package for Deep Learning in Astronomy. It is built using Keras for developing ML models to run on CPU and GPU seamlessly. The aim is to make applying machine learning techniques on astronomical data easy for astronomers, researchers and students.

1.1.1 Applications

MiraPy can be used for problem solving using ML techniques and will continue to grow to tackle new problems in Astronomy. Following are some of the experiments that you can perform right now:

- Classification of X-Ray Binaries using neural network
- Astronomical Image Reconstruction using Autoencoder
- Classification of the first catalog of variable stars by ATLAS
- HTRU1 Pulsar Dataset Image Classification using Convolutional Neural Network
- Variable star Classification using Recurrent Neural Network (RNN)
- 2D visualization of feature sets using Principal Component Analysis (PCA)
- Curve Fitting using Autograd (basic implementation)

There are more projects that we will add soon and some of them are as following:

- Feature Engineering (Selection, Reduction and Visualization)
- Classification of different states of GRS1905+105 X-Ray Binaries using Recurrent Neural Network (RNN)
- Feature extraction from Images using Autoencoders and its applications in Astronomy

You can find the applications MiraPy in our tutorial repository.

In future, MiraPy will be able to do more and in better ways and we need your suggestions! Tell us what you would like to see as a part of this package on Slack.

1.1.2 Installation

You can download the package using *pip* package installer:

```
pip install mirapy
```

You can also build from source code:

```
git clone --recursive https://github.com/mirapy-org/mirapy.git
cd mirapy
pip install -r requirements.txt
python setup.py install
```

1.1.3 Contributing

MiraPy is far from perfect and we would love to see your contributions to open source community! MiraPy is open source, built on open source, and we'd love to have you hang out in our community.

1.1.4 About Us

MiraPy is developed by Swapnil Sharma and Akhil Singhal as their final year 'Major Technical Project' under the guidance of Dr. Arnav Bhavsar at Indian Institute of Technology, Mandi.

1.2 Installation

You can download the package using *pip* package installer:

```
pip install mirapy
```

You can also build from source code:

```
git clone --recursive https://github.com/mirapy-org/mirapy.git
cd mirapy
pip install -r requirements.txt
python setup.py install
```

1.3 Tutorials

We have a set of MiraPy tutorials for problem-solving in Astronomy using Deep Learning. You can find the Jupyter notebooks in our Github repository. Following are the short descriptions of MiraPy applications:

- **Astronomical Image Reconstruction using Autoencoder**

  Encoder-decoder networks can be trained for noise removal from blurry image. We can use MiraPy for astronomical image reconstruction by training a simple denoising autoencoder using some images of galaxies and nebulae in Missier catalog.

- **ATLAS variable star Classification**

  We demonstrate how to use MiraPy to classify variable stars using features extracted from light curves. These features are available in ATLAS catalog. We use deep neural network for the same.

- **OGLE variable star Classification**

  We demonstrate how to use MiraPy to classify variable stars using light-curves available in OGLE variable star catalogs. We use Recurrent Neural Network (RNN) in the classification model.

- **HTRU1 Batched Dataset Classification**

MiraPy can be used for the classification of pulsars and non-pulsars in dataset released by HTRU1 survey. The dataset contains 60000 images which are classified using Convolutional Neural Network (CNN).

- **X-Ray Binary Classification**

  Tutorial demonstrates how to use Fully-Connected Neural (FCN) network to classify features of pulsar, non-pulsar and black hole systems.

- **2D and 3D visualisation**

  We demonstrate how to use MiraPy to visualize a feature dataset using 2D and 3D graphs. For this purpose, we use Pricipal Component Analysis (PCA) for feature reduction.

1.4 License

1.5 API Reference

This page contains auto-generated API reference documentation[1].

1.5.1 `mirapy`

Subpackages

`mirapy.autoencoder`

Submodules

`mirapy.autoencoder.models`

[1] Created with sphinx-autoapi

Module Contents

class mirapy.autoencoder.models.**Autoencoder**

Base Class for autoencoder models.

compile(*self*, *optimizer*, *loss*)

Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train(*self*, *x*, *y*, *batch_size=32*, *epochs=100*, *validation_data=None*, *shuffle=True*, *verbose=1*)

Trains the model on the training data with given settings.

Parameters

- **x** – Numpy array of training data.
- **y** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.
- **batch_size** – Number of samples per gradient update.
- **validation_data** – Numpy array of validation data.
- **shuffle** – Boolean. Shuffles the data before training.
- **verbose** – Value is 0, 1, or 2.

predict(*self*, *x*)

Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

plot_history(*self*)

Plots loss vs epoch graph.

save_model(*self*, *model_name*, *path='models/'*)

Saves a model into a H5py file.

Parameters

- **model_name** – File name.
- **path** – Pa

load_model(*self*, *model_name*, *path='models/'*)

Loads a model from a H5py file.

Parameters

- **model_name** – File name.
- **path** – Pa

summary(*self*)

class mirapy.autoencoder.models.**DeNoisingAutoencoder**(*img_dim*, *activation='relu'*, *padding='same'*)

Bases: *mirapy.autoencoder.models.Autoencoder*

De-noising Autoencoder used for the astronomical image reconstruction.

Parameters

- **img_dim** – Set. Dimension of input and output image.
- **activation** – String (activation function name).
- **padding** – String (type of padding in convolution layers).

compile (*self*, *optimizer*, *loss*)
Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train (*self*, *x*, *y*, *batch_size=32*, *epochs=100*, *validation_data=None*, *shuffle=True*, *verbose=1*)
Trains the model on the training data with given settings.

Parameters

- **x** – Numpy array of training data.
- **y** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.
- **batch_size** – Number of samples per gradient update.
- **validation_data** – Numpy array of validation data.
- **shuffle** – Boolean. Shuffles the data before training.
- **verbose** – Value is 0, 1, or 2.

predict (*self*, *x*)
Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

show_image_pairs (*self*, *original_images*, *decoded_images*, *max_images*)
Displays images in pair of images in grid form using Matplotlib.

Parameters

- **original_images** – Array of original images.
- **decoded_images** – Array of decoded images.
- **max_images** – Integer. Set number of images in a row.

mirapy.classifiers

Submodules

mirapy.classifiers.models

Module Contents

class mirapy.classifiers.models.**Classifier**
Base class for classification models. It provides general abstract methods required for applying a machine learning techniques.

compile (*self*, *optimizer*, *loss='mean_squared_error'*)
Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

save_model (*self*, *model_name*, *path='models/'*)
Saves a model into a H5py file.

Parameters

- **model_name** – File name.
- **path** – Path of directory.

load_model (*self*, *model_name*, *path='models/'*)
Loads a model from a H5py file.

Parameters

- **model_name** – File name.
- **path** – Pa

train (*self*, *x_train*, *y_train*, *epochs*, *batch_size*, *reset_weights*, *class_weight*, *validation_data*, *verbose*)
Trains the model on the training data with given settings.

Parameters

- **x_train** – Numpy array of training data.
- **y_train** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.
- **batch_size** – Number of samples per gradient update.
- **reset_weights** – Boolean. Set true to reset the weights of model.
- **class_weight** – Dictionary. Weights of classes in loss function.
- **validation_data** – Numpy array of validation data.
- **verbose** – Value is 0, 1, or 2.

predict (*self*, *x*)
Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

plot_history (*self*)
Plots loss vs epoch graph.

reset (*self*)
Resets all weights of the model.

class mirapy.classifiers.models.**XRayBinaryClassifier** (*activation='relu'*)
Bases: *mirapy.classifiers.models.Classifier*

Classification model for X-Ray Binaries.

Parameters **activation** – String (activation function name).

compile (*self*, *optimizer=Adam(lr=0.0001*, *decay=1e-06)*, *loss='mean_squared_error'*)
Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train (*self*, *x_train*, *y_train*, *epochs=50*, *batch_size=100*, *reset_weights=True*, *class_weight=None*, *validation_data=None*, *verbose=1*)
Trains the model on the training data with given settings.

Parameters

- **x_train** – Numpy array of training data.
- **y_train** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.
- **batch_size** – Number of samples per gradient update.
- **reset_weights** – Boolean. Set true to reset the weights of model.
- **class_weight** – Dictionary. Weights of classes in loss function during training.
- **validation_data** – Numpy array of validation data.
- **verbose** – Value is 0, 1, or 2.

predict (*self*, *x*)
Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

Returns Predicted class for Input data.

class mirapy.classifiers.models.**AtlasVarStarClassifier** (*activation='relu'*, *input_size=22*, *num_classes=9*)
Bases: *mirapy.classifiers.models.Classifier*

Classification model for variable star features in ATLAS catalog.

Parameters

- **activation** – String (activation function name).
- **input_size** – Integer. Dimension of Feature Vector.
- **num_classes** – Integer. Number of Classes.

compile (*self*, *optimizer=Adam(lr=0.01*, *decay=0.01)*, *loss='mean_squared_error'*)
Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train (*self*, *x_train*, *y_train*, *epochs=50*, *batch_size=100*, *reset_weights=True*, *class_weight=None*, *validation_data=None*, *verbose=1*)
Trains the model on the training data with given settings.

Parameters

- **x_train** – Numpy array of training data.
- **y_train** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.

- **batch_size** – Number of samples per gradient update.
- **reset_weights** – Boolean. Set true to reset the weights of model.
- **class_weight** – Dictionary. Weights of classes in loss function during training.
- **validation_data** – Numpy array of validation data.
- **verbose** – Value is 0, 1, or 2.

predict(*self*, *x*)
Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

Returns Predicted class for Input data.

class mirapy.classifiers.models.**OGLEClassifier**(*activation='relu'*, *input_size=50*, *num_classes=5*)
Bases: *mirapy.classifiers.models.Classifier*

Feature classification model for OGLE variable star time-series dataset.

Parameters

- **activation** – String (activation function name).
- **input_size** – Integer. Dimension of Feature Vector.
- **num_classes** – Integer. Number of Classes.

compile(*self*, *optimizer='adam'*, *loss='categorical_crossentropy'*)
Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train(*self*, *x_train*, *y_train*, *epochs=50*, *batch_size=100*, *reset_weights=True*, *class_weight=None*, *validation_data=None*, *verbose=1*)
Trains the model on the training data with given settings.

Parameters

- **x_train** – Numpy array of training data.
- **y_train** – Numpy array of target data.
- **epochs** – Integer. Number of epochs during training.
- **batch_size** – Number of samples per gradient update.
- **reset_weights** – Boolean. Set true to reset the weights of model.
- **class_weight** – Dictionary. Weights of classes in loss function.
- **validation_data** – Numpy array of validation data.
- **verbose** – Value is 0, 1, or 2.

predict(*self*, *x*)
Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

Returns Predicted class for Input data.

class mirapy.classifiers.models.**HTRU1Classifier**(*input_dim*, *activation='relu'*, *padding='same'*, *dropout=0.25*, *num_classes=2*)

Bases: *mirapy.classifiers.models.Classifier*

CNN Classification of pulsars and non-pulsars data released by HTRU survey as Data Release 1. The dataset has same structure as CIFAR-10 dataset.

Parameters

- **input_dim** – Set. Dimension of input data.
- **activation** – String. Activation function name.
- **padding** – Sting. Padding type.
- **dropout** – Float between 0 and 1. Dropout value.
- **num_classes** – Integer. Number of classes.

compile(*self*, *optimizer*, *loss='categorical_crossentropy'*)

Compile model with given configuration.

Parameters

- **optimizer** – Instance of optimizer.
- **loss** – String (name of loss function) or custom function.

train(*self*, *x_train*, *y_train*, *epochs=100*, *batch_size=32*, *reset_weights=True*, *class_weight=None*, *validation_data=None*, *verbose=1*)

predict(*self*, *x*)

Predicts the output of the model for the given data as input.

Parameters **x** – Input data as Numpy arrays.

Returns Predicted class for Input data.

mirapy.fitting

Submodules

mirapy.fitting.losses

Module Contents

mirapy.fitting.losses.**negative_log_likelihood**(*y_true*, *y_pred*)

Function for negative log-likelihood error.

Parameters

- **y_true** – Array of true values.
- **y_pred** – Array of predicted values.

Returns Float. Loss value.

mirapy.fitting.losses.**mean_squared_error**(*y_true*, *y_pred*)

Function for mean squared error.

Parameters

- **y_true** – Array of true values.
- **y_pred** – Array of predicted values.

Returns Float. Loss value.

mirapy.fitting.models

Module Contents

class mirapy.fitting.models.**Model1D**
Base class for 1-D model.

__call__(*self*, *x*)
Return the value of evaluate function by calling it.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the evaluate function.

evaluate(*self*, *x*)
Return the value of a model of the given input.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the model.

set_params_from_array(*self*, *params*)
Sets the parameters of the model from an array.

Parameters **params** – Array of parameter values.

get_params_as_array(*self*)
Returns the parameters of the model as an array.

class mirapy.fitting.models.**Gaussian1D**(*amplitude=1.0*, *mean=0.0*, *stddev=1.0*)
Bases: *mirapy.fitting.models.Model1D*

One dimensional Gaussian model.

Parameters

- **amplitude** – Amplitude.
- **mean** – Mean.
- **stddev** – Standard deviation.

evaluate(*self*, *x*)
Return the value of Gaussian model of the given input.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the model.

set_params_from_array(*self*, *params*)
Sets the parameters of the model from an array.

Parameters **params** – Array of parameter values.

get_params_as_array(*self*)
Returns the parameters of the model as an array.

mirapy.fitting.optimizers

Module Contents

class mirapy.fitting.optimizers.**ParameterEstimation**(*x*, *y*, *model*, *loss_function*, *callback=None*)

Base class of parameter estimation of a model using regression.

Parameters

- **x** – Array of input values.
- **y** – Array of target values.
- **model** – Model instance.
- **loss_function** – Instance of loss function.
- **callback** – Callback function.

regression_function(*self*, *params*)

Return the output of loss function.

Parameters **params** – Array of new parameters of the model.

Returns Output of loss function.

get_model(*self*)

Returns a copy of model used in estimation.

Returns Model instance.

fit(*self*)

Fits the data into the model using regression.

Returns Returns the result.

Package Contents

class mirapy.fitting.**Model1D**

Base class for 1-D model.

__call__(*self*, *x*)

Return the value of evaluate function by calling it.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the evaluate function.

evaluate(*self*, *x*)

Return the value of a model of the given input.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the model.

set_params_from_array(*self*, *params*)

Sets the parameters of the model from an array.

Parameters **params** – Array of parameter values.

get_params_as_array(*self*)

Returns the parameters of the model as an array.

class mirapy.fitting.**Gaussian1D**(*amplitude=1.0*, *mean=0.0*, *stddev=1.0*)
Bases: *mirapy.fitting.models.Model1D*

One dimensional Gaussian model.

Parameters

- **amplitude** – Amplitude.
- **mean** – Mean.
- **stddev** – Standard deviation.

evaluate(*self*, *x*)
Return the value of Gaussian model of the given input.

Parameters **x** – Array of 1-D input values.

Returns Return the output of the model.

set_params_from_array(*self*, *params*)
Sets the parameters of the model from an array.

Parameters **params** – Array of parameter values.

get_params_as_array(*self*)
Returns the parameters of the model as an array.

mirapy.fitting.**mean_squared_error**(*y_true*, *y_pred*)
Function for mean squared error.

Parameters

- **y_true** – Array of true values.
- **y_pred** – Array of predicted values.

Returns Float. Loss value.

mirapy.fitting.**negative_log_likelihood**(*y_true*, *y_pred*)
Function for negative log-likelihood error.

Parameters

- **y_true** – Array of true values.
- **y_pred** – Array of predicted values.

Returns Float. Loss value.

class mirapy.fitting.**ParameterEstimation**(*x*, *y*, *model*, *loss_function*, *callback=None*)
Base class of parameter estimation of a model using regression.

Parameters

- **x** – Array of input values.
- **y** – Array of target values.
- **model** – Model instance.
- **loss_function** – Instance of loss function.
- **callback** – Callback function.

regression_function(*self*, *params*)
Return the output of loss function.

Parameters **params** – Array of new parameters of the model.

Returns Output of loss function.

get_model (*self*)
Returns a copy of model used in estimation.

Returns Model instance.

fit (*self*)
Fits the data into the model using regression.

Returns Returns the result.

mirapy.utils

Submodules

mirapy.utils.utils

Module Contents

mirapy.utils.utils.**get_psf_airy** (*n*, *nr*)
Calculates Point Spread Function.

Parameters

- **n** –
- **nr** –

Returns Numpy array of Point Spread Function

mirapy.utils.utils.**image_augmentation** (*images*, *image_data_generator*, *num_of_augumentations*, *disable=False*)
Form augmented images for input array of images

Parameters

- **images** – numpy array of Images.
- **image_data_generator** – Keras image generator object.
- **num_of_augumentations** – Number of augmentations of each image.
- **disable** – Bool. Disable/enable tqdm progress bar.

Returns Numpy array of augmented images.

mirapy.utils.utils.**psnr** (*img1*, *img2*)
Calculate Peak Signal to Noise Ratio value.

Parameters

- **img1** – Float. Array of first image.
- **img2** – Float.Array of second image.

Returns Float. PSNR value of x and y.

mirapy.utils.utils.**append_one_to_shape** (*x*)
Reshapes input.

Parameters **x** – Array input.

Returns Reshaped array.

mirapy.utils.utils.**unpickle**(*file*)
Unpickle and read file.

Parameters **file** – Pickle file to read.

Returns Data loaded from pickle file.

mirapy.utils.utils.**to_numeric**(*y*)
Convert numpy array of array of probabilities to numeric array.

Parameters **y** – Numpy array.

Returns Numpy array of classes.

mirapy.utils.utils.**accuracy_per_class**(*y_true*, *y_pred*)
Computes accuracy per class.

Parameters

- **y_true** – True class.
- **y_pred** – Predicted class.

Returns

Package Contents

mirapy.utils.**get_psf_airy**(*n*, *nr*)
Calculates Point Spread Function.

Parameters

- **n** –
- **nr** –

Returns Numpy array of Point Spread Function

mirapy.utils.**image_augmentation**(*images*, *image_data_generator*, *num_of_augumentations*, *disable=False*)
Form augmented images for input array of images

Parameters

- **images** – numpy array of Images.
- **image_data_generator** – Keras image generator object.
- **num_of_augumentations** – Number of augmentations of each image.
- **disable** – Bool. Disable/enable tqdm progress bar.

Returns Numpy array of augmented images.

mirapy.utils.**psnr**(*img1*, *img2*)
Calculate Peak Signal to Noise Ratio value.

Parameters

- **img1** – Float. Array of first image.
- **img2** – Float.Array of second image.

Returns Float. PSNR value of x and y.

mirapy.utils.**append_one_to_shape**(*x*)
Reshapes input.

Parameters **x** – Array input.

Returns Reshaped array.

mirapy.utils.**unpickle**(*file*)
Unpickle and read file.

Parameters **file** – Pickle file to read.

Returns Data loaded from pickle file.

mirapy.utils.**to_numeric**(*y*)
Convert numpy array of array of probabilities to numeric array.

Parameters **y** – Numpy array.

Returns Numpy array of classes.

mirapy.utils.**accuracy_per_class**(*y_true*, *y_pred*)
Computes accuracy per class.

Parameters

- **y_true** – True class.
- **y_pred** – Predicted class.

Returns

mirapy.visualization

Submodules

mirapy.visualization.visualize

Module Contents

mirapy.visualization.visualize.**visualize_2d**(*x*, *y*)
Function for 2D visualization of data using Principal Component Analysis (PCA).

Parameters

- **x** – Array of features.
- **y** – Array of target values.

mirapy.visualization.visualize.**visualize_3d**(*x*, *y*)
Function for 3D visualization of data using Principal Component Analysis (PCA).

Parameters

- **x** – Array of features.
- **y** – Array of target values.

Package Contents

mirapy.visualization.**visualize_2d**(*x*, *y*)
Function for 2D visualization of data using Principal Component Analysis (PCA).

Parameters

- **x** – Array of features.
- **y** – Array of target values.

mirapy.visualization.**visualize_3d**(*x*, *y*)
Function for 3D visualization of data using Principal Component Analysis (PCA).

Parameters

- **x** – Array of features.
- **y** – Array of target values.

Package Contents

mirapy.**__minimum_python_version__ = 3.5**

exception mirapy.**UnsupportedPythonError**
Bases: Exception

1.5.2 load_dataset

Module Contents

load_dataset.**load_messier_catalog_images**(*path*, *img_size=None*, *disable_tqdm=False*)
Data loader for Messier catalog images. The images are available in *messier-catalog-images* repository of MiraPy organisation.

Parameters

- **path** – String. Directory path.
- **img_size** – Final dimensions of the image.
- **disable_tqdm** – Boolean. Set True to disable progress bar.

Returns Array of images.

load_dataset.**prepare_messier_catalog_images**(*images*, *psf*, *sigma*)
Function to apply convolution and add noise from poisson distribution on an array of images.

Parameters

- **images** – Array of images.
- **psf** – Point Spread Function (PSF).
- **sigma** – Float. VStandard deviation.

Returns Original image arrays and convolved image arrays.

load_dataset.**load_xray_binary_data**(*path*, *standard_scaler=True*)
Loads X Ray Binary dataset from directory.

Parameters

- **path** – Path to the directory.
- **standard_scaler** – Bool. Standardize data or not.

Returns Dataset and Class labels.

load_dataset.**load_atlas_star_data**(*path*, *standard_scaler=True*, *feat_list=None*)
Loads ATLAS variable star dataset from directory.

Parameters

- **path** – Path to the directory.
- **standard_scaler** – Bool. Standardize data or not.
- **feat_list** – List of features to include in dataset.

Returns Dataset and Class labels.

load_dataset.**load_ogle_dataset**(*path*, *classes*, *time_len=50*, *pad=False*)
Loads OGLE variable star time series data from directory.

Parameters

- **path** – Path to the directory.
- **classes** – Classes to include in dataset.
- **time_len** – Length of time series data.
- **pad** – Bool. Pad zeroes or not.

Returns Dataset and Class labels.

load_dataset.**load_htru1_data**(*data_dir='htru1-batches-py'*)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

m

INDEX

Symbols

A

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

X