EWSNet

Sahil Sidheekh

Jun 09, 2021

CONTENTS:

1 Inference & Finetuning using Pretrained EWSNet 3

2 Data Generation and Preprocessing 5

3 Data Loading 7

4 Model Training and Evaluation 9

5 Indices and tables 11

Python Module Index 13

Index 15

This is the official code documentation for the Early Warning Signal Network (EWSNet), a deep learning model trained using simulated stochastic time series data for anticipating transitions and regime shifts in complex dynamical systems.

CHAPTER

ONE

INFERENCE & FINETUNING USING PRETRAINED EWSNET

class `src.inference.ewsnet.`**`EWSNet`**(*ensemble=1*, *weight_dir=None*, *prefix=''*, *suffix='.h5'*)

This is a wrapper class to load pretrained EWSNet models and perform inference on custom data points. To instanatiate this inference wrapper, you need to have pretrained weights stored in a local directory. Supports ensembling multiple models for increased reliability and robustness.

Initializing the EWSNet wrapper.

Parameters

- **`ensemble`** (*`int, optional`*) – A variable indicating the no. of models to ensemble and average predictions over.
- **`weight_dir`** (*`str, optional`*) – Path to the directory to load the weights from.
- **`prefix`** (*`str, optional`*) – Prefix for the weight filenames
- **`suffix`** (*`str, optional`*) – Suffix for the weight filenames

Attributes

- **model** A list of size *ensemble* holding the corresponding models, that are instances of type model class:*tf.keras.Model*

Note: Note that the model weights should be saved as $_PREFIX_$i$_SUFFIX_ where i corresponds to the index of the model in the ensemble.

Note: Once loading of the weights is successfull, use the predict() function to test custom time series data using EWSNet.

`build_model`()

Function to define and build the neural network architecture for EWSNet

`finetune`(*X*, *y*, *freeze_feature_extractor=True*, *learning_rate=5e-05*, *batch_size=512*, *tune_epochs=5*)

Function to finetune EWSNet on a custom dataset. By default finetunes all models in the ensemble based on the given data and set of parameters.

Parameters

- **`X`** (*`np.array, required`*) – The data points (univariate timeseries) to finetune EWSNet on. Dimension - (N x D) or (N x 1 x D) where *N* denotes the no. of samples and *D* denotes the no. of time steps.
- **`y`** (*`np.array, required`*) – The target labels corresponding to the data points (X). Dimension - (N,) or (N x 1) where *N* denotes the no. of samples.

- **`freeze_feature_extractor`** (*`bool, optional`*) – A boolean flag that determines the part of the network to be finetuned. When set to False. the entire network is finetuned. When set to True, only the fully connected layers are finetuned and the feature extraction blocks are frozen.
- **`learning_rate`** (*`float, optional`*) – The learning rate for finetuning the models.
- **`batch_size`** (*`int, optional`*) – The batch size for finetuning the models.
- **`tune_epochs`** (*`int, optional`*) – The no. of epochs for finetuning the models.

`load_model`(*weight_dir*, *prefix*, *suffix*)

Function to load the model from the weights present in the given directory

Parameters

- **`weight_dir`** (*`str,`*) – Path to the directory to load the weights from.
- **`prefix`** (*`str,`*) – Prefix for the weight filenames
- **`suffix`** (*`str,`*) – Suffix for the weight filenames

`predict`(*x*)

Function to make predictions using EWSNet.

Parameters **`x`** (*`1 dimensional np.array or list , required`*) – The datapoint (univariate timeseries) to test for future transitions

Returns A tuple consisting of the predicted label and the predictoin probability for each class.

CHAPTER

TWO

DATA GENERATION AND PREPROCESSING

`src.generate_data.`**`model1`**(*n_max*, *c*, *c_max*, *r*, *b*, *M*, *corr*)

Generate multiple stochastic time series by solving the following differential equation using the Euler Maruyama method:

$\frac{dN}{dt} = rN(1 - \frac{N}{K}) - \frac{cN^2}{b^2+N^2}$.

The values of *sigma* and *k* are varied within a range in order to add variability across time series to generate time series with increased variablility.

Change the value of corr to generate time series when system is perturbed with colored noise of different amplitude.

Parameters

- **`k`** – carrying capacity
- **`r`** – maximum growth rate
- **`c`** – maximum grazing rate
- **`b`** – half saturation constant
- **`M`** – Number of time series to be generated for the given set of parameter values
- **`n_max`** – number of timesteps
- **`corr`** – Correlation time
- **`x0`** – initial condition.

Note: Modify the stochastic differential equation to generate time series data from other models

CHAPTER THREE

DATA LOADING

src.utils.generic_utils.**load_dataset_at**(*index*, *normalize_timeseries=False*, *verbose=True*, *is_timeseries=True) -> (<built-in function array>*, *<built-in function array>*)

Loads a Univaraite Dataset indexed by *utils.constants*. The dataset is loaded as a pandas DataFrame and preprocessed to replace missing values with zero.

Note: The dataset should be such that the first column corresponds to the target class label. i.e a dataset consisting of N time series of length T would be a dataframe of dimension Nx(T+1) where the first column corresponds to the class labels.

Parameters

- **index** – Integer index, set inside *utils.constants* that refers to the dataset.
- **normalize_timeseries** – Bool / Integer. Determines whether to normalize the timeseries. If False, does not normalize the time series. If True / int not equal to 2, performs standard sample-wise z-normalization. If 2: Performs full dataset z-normalization.
- **verbose** – Whether to describe the dataset being loaded.

Returns A tuple of shape (X_train, y_train, X_test, y_test, is_timeseries). For legacy reasons, is_timeseries is always True.

src.utils.generic_utils.**plot_roc**(*y_test*, *y_score*, *figname='none'*, *n_classes=3*)

Plots the ROC Curve given the target and the prediction probabilities

Parameters

- **y_test** – The target class labels
- **y_score** – The models output prediction probabilities
- **figname** – Name of the figure for saving.
- **n_classes** – Number of classes.

CHAPTER

FOUR

MODEL TRAINING AND EVALUATION

`src.model_training.exp_utils.evaluate_model`(*model: tensorflow.python.keras.engine.training.Model*, *dataset_id*, *dataset_prefix*, *batch_size=128*, *test_data_subset=None*, *cutoff=None*, *normalize_timeseries=False*, *error_analysis=False*)

Evaluates a given Keras Model on the provided dataset.

Parameters

- **`model`** – A Keras Model.
- **`dataset_id`** – Integer id representing the dataset index containd in *utils/constants.py*.
- **`dataset_prefix`** – Name of the dataset. Used for weight saving.
- **`batch_size`** – Size of each batch for evaluation.
- **`test_data_subset`** – Optional integer id to subset the test set. To be used if the test set evaluation time is significantly.
- **`cutoff`** – Optional integer which slices of the first *cutoff* timesteps from the input signal.
- **`normalize_timeseries`** – Bool / Integer. Determines whether to normalize the timeseries.

  If False, does not normalize the time series. If True / int not equal to 2, performs standard sample-wise

    z-normalization.

  If 2: Performs full dataset z-normalization.

Returns The test set accuracy of the model.

`src.model_training.exp_utils.train_model`(*model: tensorflow.python.keras.engine.training.Model*, *dataset_id*, *dataset_prefix*, *epochs=50*, *batch_size=128*, *val_subset=5000*, *cutoff=None*, *normalize_timeseries=False*, *learning_rate=1e-05*, *prediction=False*)

Trains a provided Model, given a dataset id.

Parameters

- **`model`** – A Keras Model.
- **`dataset_id`** – Integer id representing the dataset index containd in *utils/constants.py*.
- **`dataset_prefix`** – Name of the dataset. Used for weight saving.

- **`epochs`** – Number of epochs to train.
- **`batch_size`** – Size of each batch for training.
- **`val_subset`** – Optional integer id to subset the test set. To be used if the test set evaluation time significantly surpasses training time per epoch.
- **`cutoff`** – Optional integer which slices of the first *cutoff* timesteps from the input signal.
- **`normalize_timeseries`** – Bool / Integer. Determines whether to normalize the time-series. If False, does not normalize the time series. If True / int not equal to 2, performs standard sample-wise z-normalization. If 2: Performs full dataset z-normalization.
- **`learning_rate`** – Initial learning rate.

Returns The trained model.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

src.model_training.exp_utils, 9

src.utils.generic_utils, 7

INDEX

B

build_model() (*src.inference.ewsnet.EWSNet method*), 3

E

evaluate_model() (*in module src.model_training.exp_utils*), 9

EWSNet (*class in src.inference.ewsnet*), 3

F

finetune() (*src.inference.ewsnet.EWSNet method*), 3

L

load_dataset_at() (*in module src.utils.generic_utils*), 7

load_model() (*src.inference.ewsnet.EWSNet method*), 4

M

model1() (*in module src.generate_data*), 5

P

plot_roc() (*in module src.utils.generic_utils*), 7

predict() (*src.inference.ewsnet.EWSNet method*), 4

S

src.model_training.exp_utils (*module*), 9

src.utils.generic_utils (*module*), 7

T

train_model() (*in module src.model_training.exp_utils*), 9