EWSNet User Guide

2021

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The transition from one steady-state to another alternative steady-state occurs in many complex systems, such as financial markets, human societies, climate systems, and various other domains [1]. Transitions may be abrupt and irreversible (i.e., catastrophic) or smooth and reversible (i.e., non-catastrophic), and can occur due to gradual external forcing or random fluctuations in the system. In such scenarios, on crossing a threshold (known as a tipping or bifurcation point), structural changes occur in the underlying system. This is often termed a critical transition, prior to which the system's return to an equilibrium slows down - a phenomenon known as critical slowing down (CSD) [1]-[2]. The phenomenon of CSD is related to the fact that the real part of the dominant eigenvalue of the system goes to zero at the bifurcation point. In all such cases, where the dominant eigenvalue approaches zero close to the tipping point irrespective of catastrophic or non-catastrophic transitions, the phenomenon of CSD persists, and there exist statistical indicators that forewarn the vicinity of a tipping point [3]-[5]. Understanding the causes of sudden transitions and forecasting them using statistical indicators have recently emerged as an important area of research due to the management implications of preventing catastrophes in natural systems.

The traditional approach of forecasting critical transitions relies on CSD based indicators such as variance, autocorrelation, and skewness showing an increasing trend before a transition. In this work, we take a different direction and propose using machine learning techniques to anticipate transitions in complex dynamical systems.

CHAPTER

TWO

MACHINE LEARNING

Maintaining a level of abstraction, human intelligence, and decision-making can be understood as analogies to mathematical functions. The inference that we make is essentially a function that processes a set of input sensory signals, mapping it to some meaningful target domain. For example, when one infers "*that is an apple*," it can be understood as a function that maps visual sensory signals to a known space of objects. What constitutes knowledge and intelligence then is learning such functions, and human beings learn from experience.

On similar lines, one way to enable machines to make inferences is to model the task using a parameterized function that maps data in some input space to the desired target space. Machine *learning* then refers to finding the optimal parameters for the function that best explains the data for the task at hand. Utilizing samples of data in the input and target spaces, one can learn the function that maps the input domain to the target domain [6]. This enables performing inference on a new/unseen data point, popularly known as supervised learning, and constitutes the base for the functioning of EWSNet.

Deep Learning: Deep neural networks are essentially parameterized functions, proven to be universal function approximators [7]. Efficiently learning from data using classical machine learning models often required handcrafting efficient feature representations for the data. The advent of convolutional neural networks enabled stacking multiple neural network layers and going deep to automatically extract features important for learning from raw input data.

In their original form, neural networks were built to process I.I.D. data samples in fixed dimensional spaces. On the contrary, most real-life data, such as time-series, text, audio, video, etc., are sequential and of variable length. To process such sequential data and capture the inherent dependence that is present, novel neural network architectures have been proposed, the most popular among them being recurrent neural networks (RNN). RNNs process sequential data in an iterative manner, keeping a memory of data at the previous time steps. However, capturing long-term dependencies is difficult with RNNs. An LSTM is a sophisticated RNN architecture that addresses this issue by enabling the addition and deletion of information to a common memory propagated across time steps.

CHAPTER

THREE

EWSNET

EWSNet is a deep neural network, a parameterized function that employs a LSTM and a fully convoluted network. The LSTM captures long-term dependencies in the sequential time series data and the fully convolutional sub-module helps automatically extract complex nonlinearities from the data, requisite to learn the characteristics indicative of a future transition.

EWSNet can classify catastrophic, non-catastrophic, and no transitions in raw time series data learning fundamental properties that characterize transitions and critical slowing down, which helps it distinguish a no transition from the other two variants. EWSNet makes none but one presumption that the test data must belong to a similar distribution as training data. By a similar distribution, it is meant that test data have critical behavior (bifurcations) similar to underlying bifurcations of time series data that are deployed for training. The present EWSNet is not specific to models on which it is trained but critical behavior specific. There is always a choice to update and retrain EWSNet to cover an exhaustive space of such critical behaviors under each transition label and increase its prediction's robustness.

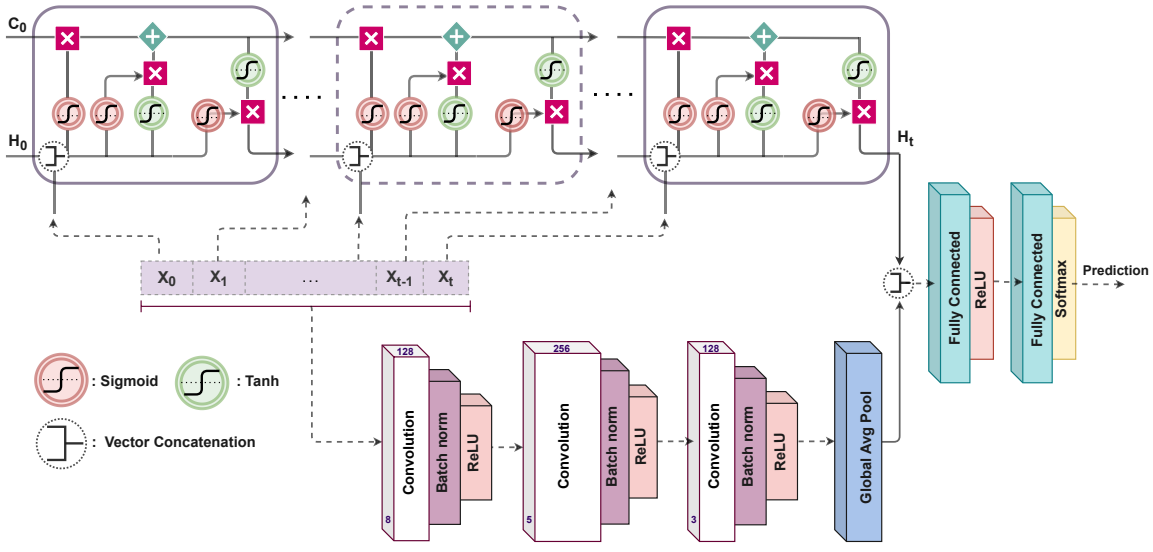

Fig. 1: The EWSNet Architecture.

CHAPTER

FOUR

DATA GENERATION

We generate stochastic time series that serve as training data pertaining to the three labels (catastrophic, non-catastrophic, and no transition). Time series are obtained by simulating nine well-studied models that depict the above transitions in diverse systems, such as ecological, climatic, and systems biology. We consider data prior to tipping to train the EWSNet. We train EWSNet with various non-linear models for variance in the training set that adds to the robustness of the EWSNet. In the process, we first obtain the deterministic tipping by plotting the bifurcation diagram in XPPAUT [8]. We also vary parameters apart from the control (bifurcation parameter) wherever possible. It is to be noted that an extra parameter is varied in models [1], [3], [6], [7], and [9] (see SI, Table S1) only after reconfirming that a particular bifurcation persists within the range. To confirm this, we plot a two-parameter bifurcation diagram using XPPAUT. This aids in more variability across time series.

Further, we solve the system using the Euler-Maruyama method with integration step size $dt = 0.01$. We choose different values of another parameter from a range, other than the control parameter as in models [1],[3],[6],[7], and [9]. In such cases, the respective parameter value is picked up randomly for each simulation within the mentioned range which remains fixed for all time steps and varies only across time series.

We generate 0.1 million time series, each for Dataset-W and Dataset-C. However, in Dataset-C, we train with different time series by simulating models for $\kappa = -0.2$ and 0.2. At the same time, the corresponding test data contain time series generated from the underlying models for high amplitude autocorrelation coefficient values $[\kappa = -0.64, 0.64, -0.8, 0.8]$. Time series for both training and testing are generated in equal proportions for all three labels. Model [1]-[5] correspond to the label critical (catastrophic) transition, [6]-[8] replicate smooth (non-catastrophic) transition, and model 9 represent no-transition. All the chosen parameter values for model [1]-[9] are taken from published literature cited in the main text and supplementary information. Variation in parameters to add variability to the datasets is done only after plotting both one-parameter and two-parameter bifurcations.

Here, we explain in detail the data generation process for model [1]. The deterministic dynamics of model [1] can be represented by the below differential equation:

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

where $r = 1$ represents the maximum growth rate, $K \in [8, 10]$ is the carrying capacity, and $b = 1$ is the half saturation constant. c is the bifurcation parameter which is varied in the range $1 - 3$ (see Fig. 1). Time series generated by simulating model [1] correspond to the label catastrophic transition. In generating the stochastic time series, the parameters r, b are fixed, while K takes a uniform random number in the range [8,10] in order to add variability across time series. This is done by confirming that the saddle-node

bifurcation persists with the help of a two-parameter bifurcation in the above range for c for the given K value (Fig. 2). The deterministic bifurcation point is $c = 2.13$ for $K = 8$, and $c = 2.6$ for $K = 10$. We generate time series data by simulating for 400 time steps prior to a tipping. Suppose we generate n time series by simulating model [1] for 400 time steps as c varies in the range $1 - 1.8$ with time, n independent K values are picked up from $U[8, 10]$ one for each simulation with all other parameters fixed. This also allows for variance in dataset and reduce bias if any, as pre-transition time series that are used to train the EWSNet are at varying distances from the tipping.

Bifurcation embedded time series for all the other models are plotted (Figs. 3-10). The red and green curves in each figure represent the stable and unstable states of a deterministic system, respectively. The stochastic trajectory corresponding to each system is presented in blue.

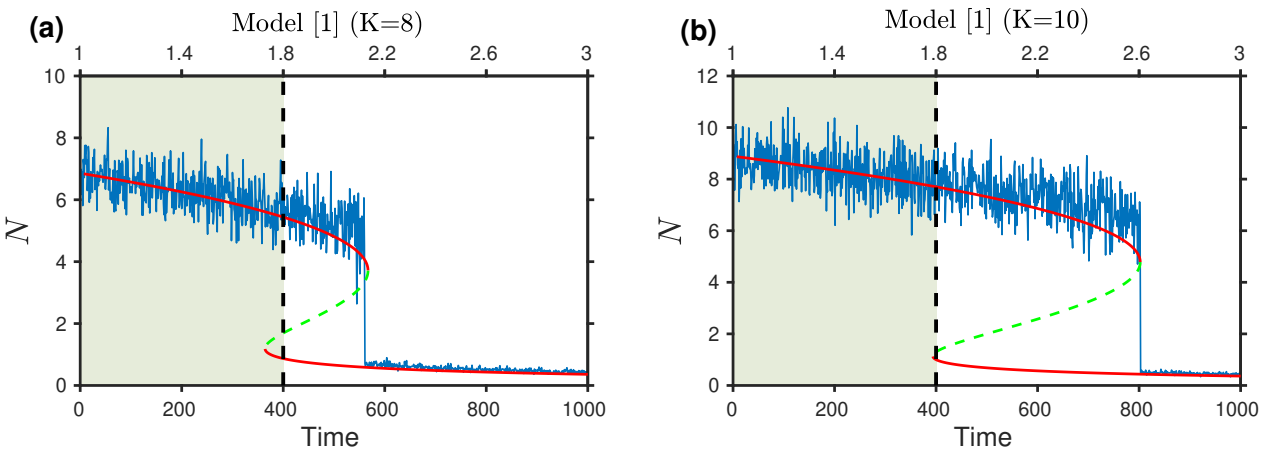

Fig. 1: Stochastic time series embedded over bifurcation diagram for Model [1]: In (a) $K = 8$, and (b) $K = 10$, respectively. The bifurcation parameter c (maximum grazing rate) is varying in the range $1 - 3$. Other parameter values are $r = 1$ and $b = 1$. The shaded region marks the pre-transition time series used for training EWSNet.

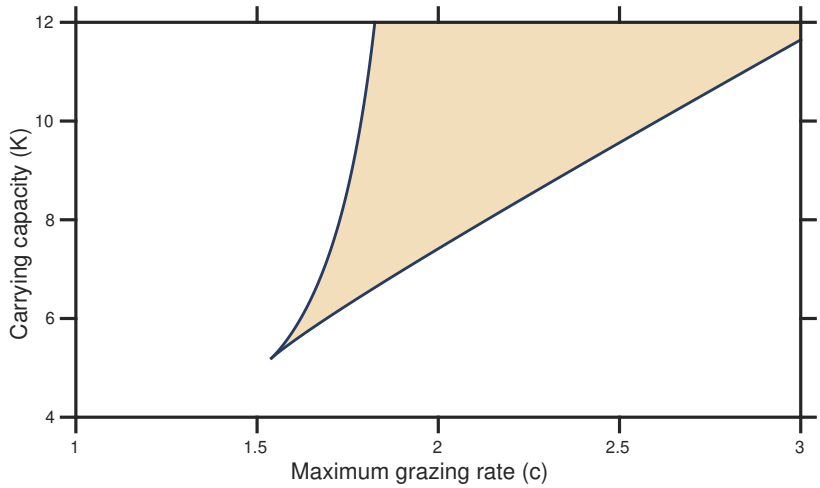

Fig. 2: Phase diagram of the overharvesting model in (K, c)- plane: The shaded region represents bistability and rest is monostable. The curve separating the monostable region from the bistable region is a saddle-node bifurcation curve. Other parameter values are $r = 1$ and $b = 1$.

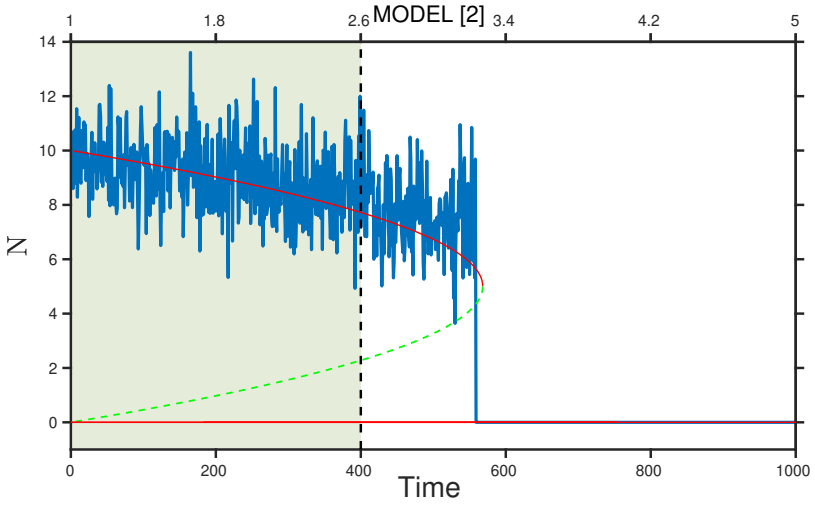

Fig. 3: Stochastic time series embedded over bifurcation diagram for Model [2]: The bifurcation parameter c (maximum grazing rate) is varying in the range $1 - 5$. Other parameter value is $K = 11$. The shaded region marks the pre-transition time series used for training EWSNet.

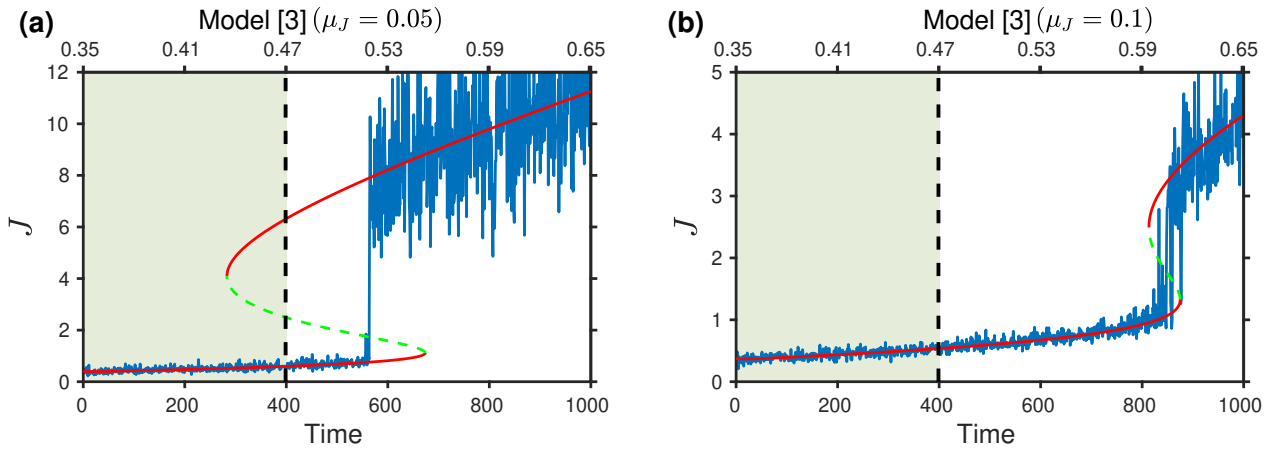

Fig. 4: Stochastic time series embedded over bifurcation diagram for Model [3]: In (a) $\mu_J = 0.05$, and (b) $\mu_J = 0.1$, respectively. The bifurcation parameter μ_P (predator mortality rate) is varying in the range $0.35 - 0.65$. Other parameter values are $b = 1$, $c = 1$, $\mu_A = 0.01$. The shaded region marks the pre-transition time series used for training EWSNet.

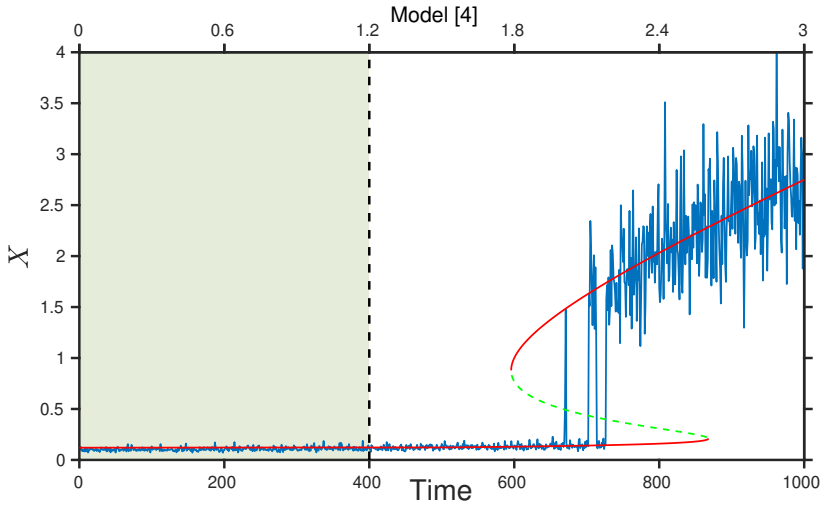

Fig. 5: Stochastic time series embedded over bifurcation diagram for Model [4]: The bifurcation parameter a (maximum transcription rate) is varying in the range $0 - 3$. Other parameter is $r = 0.1$. The shaded region marks the pre-transition time series used for training EWSNet.

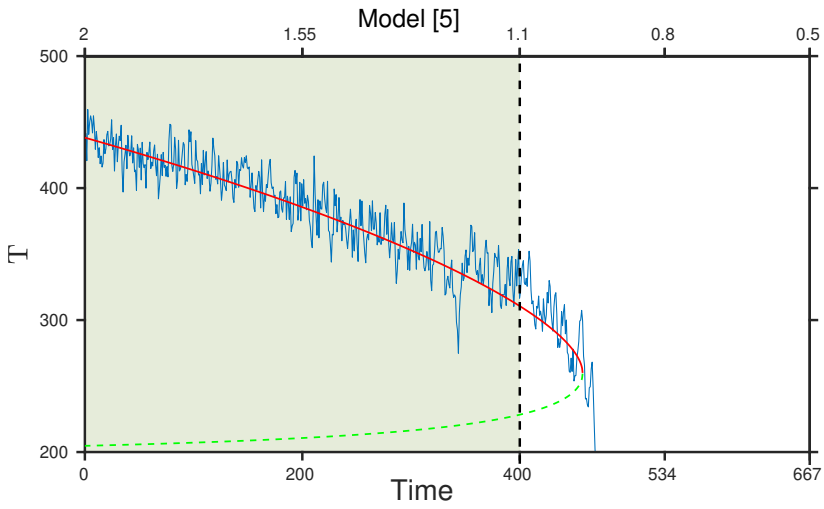

Fig. 6: Stochastic time series embedded over bifurcation diagram for Model [5]: The bifurcation parameter μ (relative intensity of solar radiation) is varying in the range $0.5 - 2$. Other parameter values are $a = 2.81$, $b = 0.009$, $c = 1.5 \times 10^8$, $\epsilon = 0.69$, $I_0 = 0.03$, $\sigma = 1.251 \times 10^{-12}$. The density T represent temperature in Kelvin. The shaded region marks the pre-transition time series used for training EWSNet.

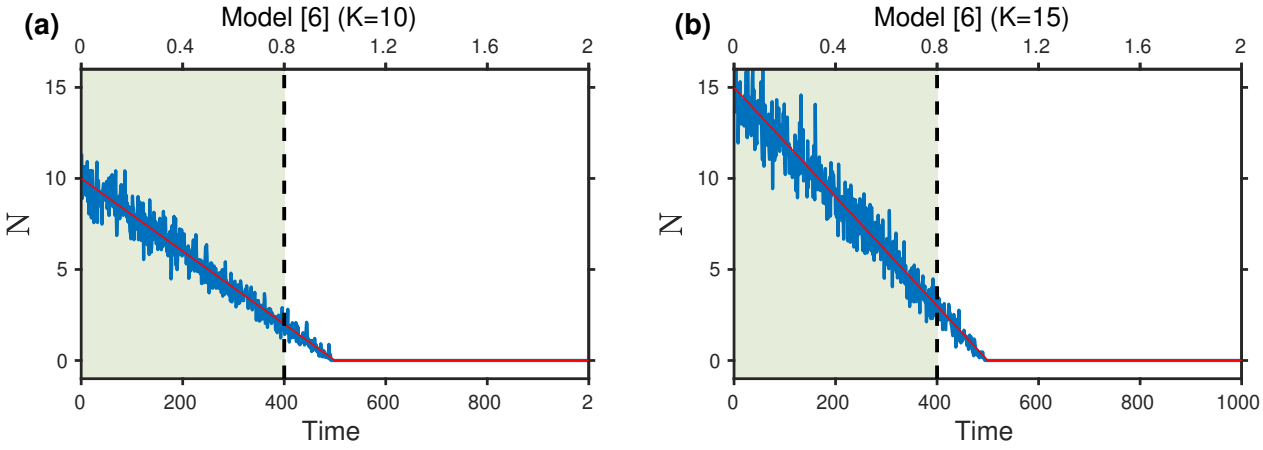

Fig. 7: Stochastic time series embedded over bifurcation diagram for Model [6]: In (a) $K = 10$ and (b) $K = 15$, respectively. The bifurcation parameter c (maximum grazing rate) is varying in the range $0 - 2$. Other parameter value is $r = 1$. The shaded region marks the pre-transition time series used for training EWSNet.

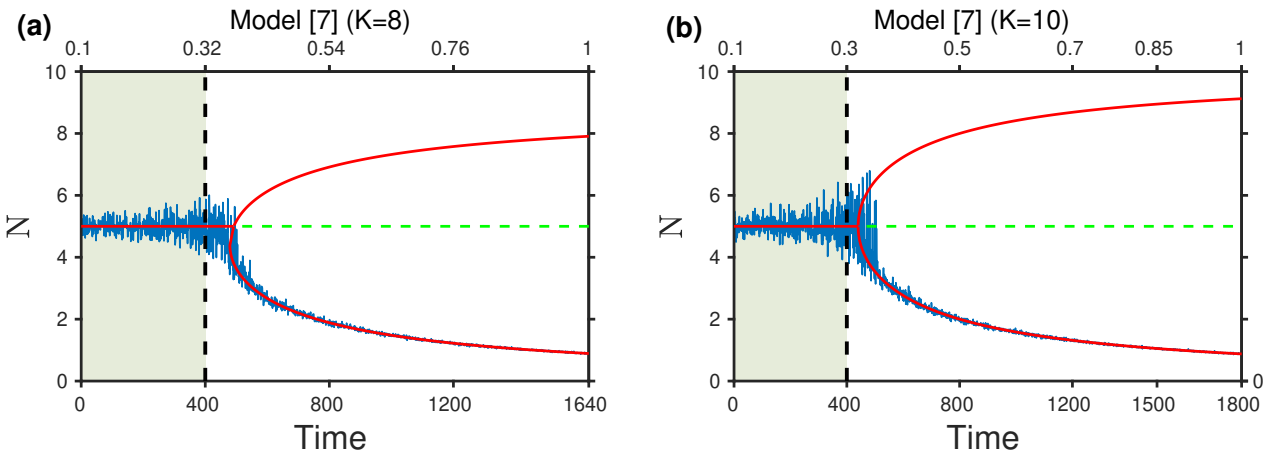

Fig. 8: Stochastic time series embedded over bifurcation diagram for Model [7]: In (a) $K = 8$ and (b) $K = 10$, respectively. The bifurcation parameter r (maximum growth rate) is varying in the range $0.1 - 1$. Other parameter values are $c = 0.8$, $N_c = 5$, $I = 4$. The shaded region marks the pre-transition time series used for training EWSNet.

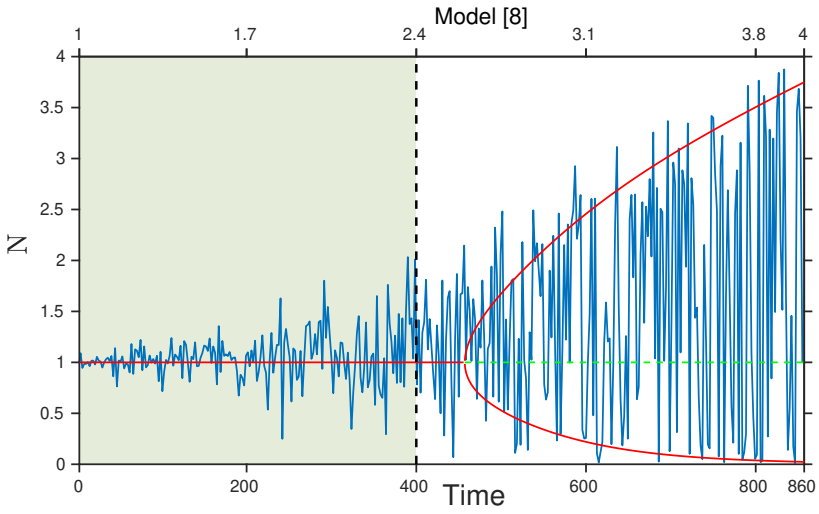

Fig. 9: Stochastic time series embedded over bifurcation diagram for Model [8]: The bifurcation parameter K (carrying capacity of resource) is varying in the range $1-4$. Other parameter values are $r = 0.5$, $a = 0.4$, $b = 0.6$, $e_1 = 0.6$, and $d_1 = 0.15$. The shaded region marks the pre-transition time series used for training EWSNet

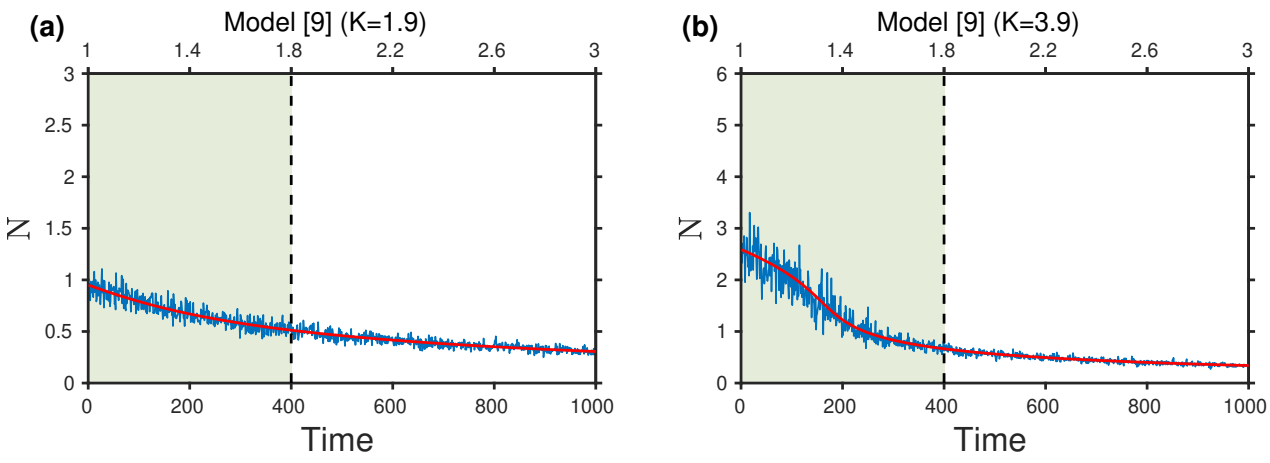

Fig. 10: Stochastic time series embedded over deterministic steady states for Model [9]: In (a) $K = 1.9$ and (b) $K = 3.9$, respectively. The control parameter c (maximum grazing rate) is varying in the range $1 - 3$. Other parameter values are $r = 1$, and $b = 1$. The shaded region marks the time series used for training EWSNet.

Function for stochastic time series generation

`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

FIVE

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

SIX

DATA LOADING & UTILS

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

SEVEN

MODEL TRAINING AND EVALUATION

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 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.
- **learning_rate** – Initial learning rate.

Returns The trained model.

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.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Marten Scheffer (2009). *Critical transitions in nature and society*. (Princeton University Press).

[2] Marten Scheffer, Jordi Bascompte, William A Brock, Victor Brovkin, Stephen R Carpenter, Vasilis Dakos, Hermann Held, Egbert H Van Nes, Max Rietkerk, and George Sugihara. Early-warning signals for critical transitions. *Nature*, 461(7260):53–59, 2009.

[3] Carpenter, Stephen R and Cole, Jonathan J and Pace, Michael L and Batt, Ryan and Brock, WA and Cline, Timmothy and Coloso, Jim and Hodgson, James R and Kitchell, Jim F and Seekell, David A and others. Early warnings of regime shifts: a whole-ecosystem experiment. *Science*, 332(6033):1079–1082, 2011.

[4] Scheffer, Marten and Carpenter, Steve and Foley, Jonathan A and Folke, Carl and Walker, Brian. Catastrophic shifts in ecosystems. *Nature*, 413(6856):591–596, 2001.

[5] Vasilis Dakos, Stephen R Carpenter, William A Brock, Aaron M Ellison, Vishwesha Guttal, Anthony R Ives, Sonia Kefi, Valerie Livina, David A Seekell, Egbert H van Nes, and others. Methods for detecting early warnings of critical transitions in time series illustrated using simulated ecological data. *PloS One*, 7(7):e41010, 2012.

[6] Anzai, Yuichiro (2009). *Pattern recognition and machine learning*. (Elsevier).

[7] Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron and Bengio, Yoshua (2016). *Deep Learning*. (MIT press Cambridge)

[8] Ermentrout, Bard (2002). *Simulating, analyzing, and animating dynamical systems: A guide to XPPAUT for researchers and students*. SIAM Press.

PYTHON MODULE INDEX

S

INDEX