COMP9414 Artificial Intelligence

Tutorial week 3 – Multi-layer neural networks

1 Problem description

We want to model the evolution of biomass in a bioprocess, which is periodically fed with a compound A, resulting in an increase in the mass of living matter (biomass) and a decrease - as it grows - in the existing substrate (food). A schematic of the bioprocess can be seen in Figure 1.

When the bioprocess receives compound A, it increases the mass of living matter and decreases the existing substrate as it grows. The dynamics of the variables can be observed in Figure 2.

The problem, then, consists of using a multi-layer neural network to approximate the bioprocess, where compound A and substrate are considered as inputs to the network and biomass as its output. To achieve this, the basic rules of data preprocessing, neural network design, training, and testing must be followed.

Three data files representing the inputs and the target are available, as shown in Table 1.

Table 1: Files provided with the data.

File	Bioprocess's meaning	Neural model's meaning
compoundA.txt	Compound A	First input
substrate.txt	Substrate	Second input
biomass.txt	Biomass	Output

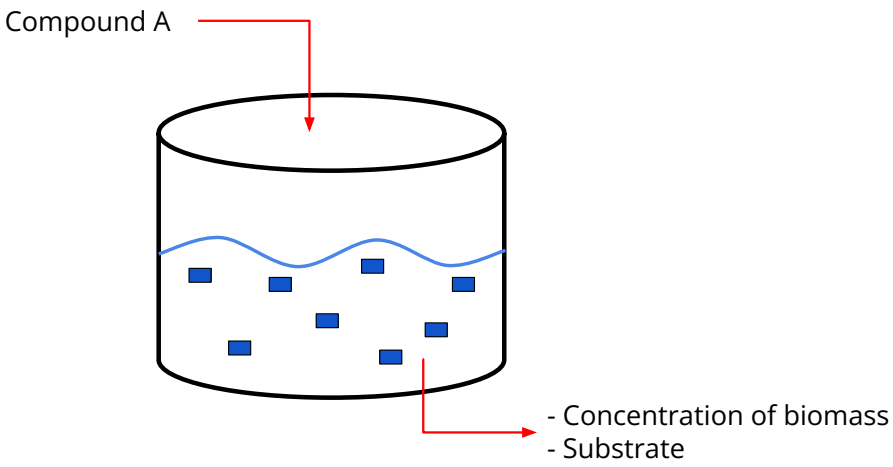

Figure 1: Bioprocess fed with compound A, which affects the concentration of biomass and the existing substrate.

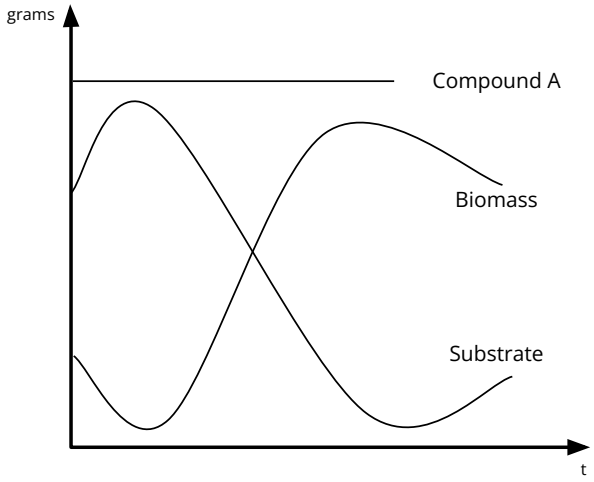

Figure 2: Dynamics of the process among the variables: compound A, substrate, and biomass.

2 Activities

2.1 Data preprocessing

In this section, it is expected that you analyse the provided data and perform any required preprocessing. Some of the tasks during preprocessing might include the ones shown below; however, not all of them are necessary and you should evaluate each of them against the results obtained.

(a) Identify the variation range for input and output variables.

(b) Plot each variable to observe the overall behaviour of the bioprocess.

(c) In case outliers are detected, correct the data accordingly. For instance, as we are dealing with variables measured in grams, no value should be less than zero. A simple correction is to replace such values with a zero value.

(d) Split the data into two subsets: training and validation.

2.2 Design of the neural network

2.2.1 Neural network libraries (TensorFlow and Keras)

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and deploying deep learning (DL) models, especially neural networks. TensorFlow allows you to define and train neural networks and to run models efficiently on CPUs and GPUs.

There are several reasons to use TensorFlow:

- **Ease of Use:** High-level APIs (such as Keras) make it easy to prototype and train models.
- **Scalability:** Works across a range of environments, from small devices to large data centres, with support for distributed training.
- **Community Support:** A large ecosystem with many tutorials, pretrained models, and third-party tools.

To install TensorFlow, you can simply run the following command in your terminal:

```
pip install tensorflow
```

For further information, see the following resources:

- TensorFlow Homepage: `https://www.tensorflow.org/`
- TensorFlow Tutorials: `https://www.tensorflow.org/tutorials`

Keras is a high-level neural networks API written in Python. It is capable of running on top of TensorFlow and is designed for fast experimentation. Keras enables you to build models with just a few lines of code (e.g., `model = Sequential([...])`) and supports quick prototyping. It is now the official high-level API of TensorFlow (`tf.keras`). Although Keras is very easy to use, you might avoid it if you need very fine-grained control over model behaviour.

Since Keras is bundled with TensorFlow, you usually don't need to install it separately. You can start using Keras in a Python environment with the following code:

```
from tensorflow import keras
from tensorflow.keras import layers
```

For further information about Keras, see the following resources:

- Keras Homepage: `https://keras.io/`
- Keras Tutorials: `https://www.tensorflow.org/guide/keras`

2.2.2 Neural network architecture

In this section, it is expected that you select and design a neural architecture for biomass estimation. Consider the following steps.

(a) Design the multi-layer neural network. Given the number of training samples, propose a neural architecture taking into account Ni – number of inputs, Nh – number of units in the hidden layer, and No – number of outputs.

(b) Decide the number of layers and their respective activation functions.

(c) Remember it's recommended that your network accomplish the maximal number of parameters Nw $<$ (number of samples)/10.

(d) Create the neural network using Keras and TensorFlow.

2.3 Training

In this section, you have to train your proposed neural network. Consider the following steps.

(a) Split the dataset considering training, validation, and test sets.

(b) Decide the training parameters such as loss function, optimiser, batch size, learning rate, and episodes.

(c) Train the neural model and verify the loss values during the process.

(d) Verify possible overfitting problems.

2.4 Validating the neural model

(a) Assess your results by plotting the network response for the test inputs against the test targets. For instance, Figure 3 shows the results for 400 points.

(b) Compute error indexes to complement the visual analysis. Use IA, RMS, and RSD.

$$\text{IA} = 1 - \frac{\sum_{i=1}^{n}(o_i - p_i)^2}{\sum_{i=1}^{n}(|o_i'| + |p_i'|)^2}$$

$$\text{RMS} = \sqrt{\frac{\sum_{i=1}^{n}(o_i - p_i)^2}{\sum_{i=1}^{n} o_i^2}}$$

$$\text{RSD} = \sqrt{\frac{\sum_{i=1}^{n}(o_i - p_i)^2}{N}}$$

with:

- o_i: observed values.
- p_i: predicted values.
- N: number of data samples.
- o_m: observations' mean value.
- $o_i' = o_i - o_m$

- $p'_i = p_i - o_m$

(c) Test your network against the test set provided, including plots and error indexes.

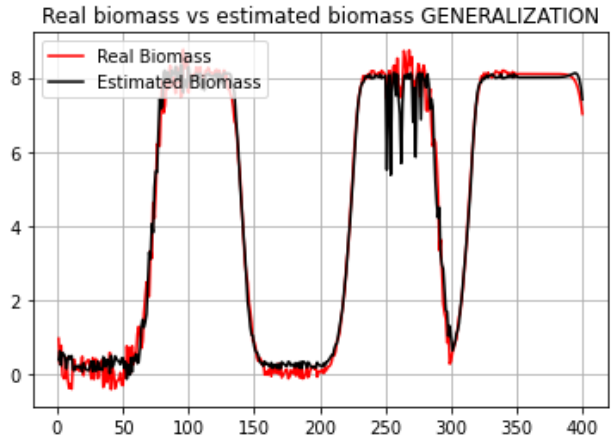

Figure 3: Groundtruth and estimated biomass using a neural network with a test set of 400 data points.