**EXERCICE 1 : - ETAPE A - Classical Approach**

Un réservoir est destiné à alimenter un réseau d'irrigation est représenté à la figure ci-dessous.

|  |  |
| --- | --- |
|  | La hauteur d’eau h(m) est mesurée à l’aide d’un capteur (-10 +10V) de constante de temps négligeable. Le débit d’entrée qe(m3/s) est déterminé à partir de la tension (-10 +10V) appliquée à la pompe de constante de temps négligeable. La hauteur minimum est de 0 m, la hauteur maximum est de 0,6 m. Le débit d’entrée minimum est de 0 m3/s, le débit maximum est de 1,1 10-4 m3/s. |

En supposant que l'équation dynamique (*donc structure du modèle connu*) qui régit le système peut s'écrire sous la forme autour d’un point de fonctionnement :

h(k+1)= a h(k) + b qe(k) (1)

**EXERCICE 1 : - ETAPE A**

1) Estimer les paramètres et du modèle décrit par (1) à partir des données k établies à la période d'échantillonnage Te=1s autour de ce même point de fonctionnement sans bruit de mesure issue du capteur (campagne *cuve1.mat*) puis avec bruit de mesure issue du capteur (campagne *cuve2.mat*).

Les paramètres pourront être estimés selon la méthode des moindres carrés à l’aide de 50% de données pour l’estimation, le reste des données servant à l’étape de validation au travers un modèle de prédiction décrit par :

(k+1)= h(k) + qe(k) (2)

2) Comparer vos résultats vis-à-vis de et ainsi que l’erreur de prédiction h(k+1)-(k+1) suivant un classique critère quadratique pour les deux campagnes.

Nous avons également réalisé deux campagnes de mesures (*cuve3.mat* et *cuve4.mat*) autour du même point de fonctionnement relevées à la période d'échantillonnage Te=1s.

3) A partir de l’un des modèles établis, que pouvez-vous en déduire sur les campagnes enregistrées ?

**EXERCICE 1 : - ETAPE B – Deep Learning Approach**

4) Etablir un modèle de prédiction à partir d’une approche basée par Réseaux de Neurones en s’attachant à considérer comme entrée h(k) et qe(k) en comme sortie désirée h(k+1).

Vous choisirez un NN à une couche cachée composée de n neurones avec des fonctions d’activation de type tangente hyperbolique (variant entre -1 et +1) en sortie également. L’algorithme d’optimisation sera choisi parmi ceux proposé. Dans un premier temps, tester l’algorithme ci-dessous en indiquant ce que vaut n.

5) Indiquer les performances (\*) du NN vis-à-vis du nombre de neurones (n=1 à 30) dans la couche cachée.

6) Indiquer, pour un nombre de neurones, les performances (\*) du NN vis-à-vis du nombre d’époques maximum de training considéré (par pas de 100 100 jusqu’à 1000).

7) Indiquer, pour un nombre de neurones, les performances (\*) du NN vis-à-vis du nombre de données considérées pour le training (0.1 à 0.9) pour chaque campagne.

\* les performances pourront être considérées au regard d’un critère quadratique aux questions précédentes cf 2).

clear all;echo off;close all

load cuve1.mat % Données sans bruit

%%%% NNN

X = [h1(1:62) qe1(1:62)]';

Y = h1(2:63)';

% Initialize params for trainTestSplit

trainingSize = 0.5;

% Split the dataset into training and test sets.

[Xtrain, Ytrain, Xval, Yval] = trainTestSplit(X', Y', trainingSize);

Xtrain=Xtrain';Ytrain=Ytrain';Xval=Xval'; Yval=Yval';

% Specify neural network architecture

layers = [...

sequenceInputLayer(size(Xtrain, 1)), ... % Input data layer

fullyConnectedLayer(1, ... % Hidden layer 1 with 1 nodes

'WeightsInitializer', 'narrow-normal', ...

'BiasInitializer', 'narrow-normal'), ...

tanhLayer, ... % Activation function 1

fullyConnectedLayer(1, ... % Output layer with 1 node

'WeightsInitializer', 'narrow-normal', ...

'BiasInitializer', 'narrow-normal'), ...

tanhLayer, ... % % Activation function 2

regressionLayer ... % Label encode

];

% Spcify hyperparameters for neural network

% 'sgdm' - Stochastic gradient descent with momentum.

% 'adam' - Adaptive moment estimation (ADAM).

% 'rmsprop' - Root mean square propagation (RMSProp).

options = trainingOptions(...

'sgdm', ... % sgdm optimization

'InitialLearnRate', 0.01, ... % Set the initial learning rate to 0.01

'MaxEpochs', 800, ... % Maximum number of epochs to train algorithm

'ValidationData', {Xval, Yval}, ... % Dataset to use as the validation set

'ValidationFrequency', 100, ... % Frequency to validate the network at regular intervals

'Verbose', true, ... % Outputs information about training

'Shuffle', 'every-epoch', ... % Data shuffling for every epoch

'Plots', 'training-progress' ... % Show plots for every epoch

);

% Function to visualize the layers object

analyzeNetwork(layers)

[dnn, trainingInfo] = trainNetwork(Xtrain, Ytrain, layers, options);

% Perform classification with DNN

ypred\_dnn = predict(dnn,X);

figure(1);plot([Y' ypred\_dnn'])