Artificial Neural Networks (ANNs)

Introduction to ANNs

ANNs have been considered as attractive and powerful tools to predict and approximate linear, nonlinear and even complex models, based only on input-output data mapping. Actually, an ANN consists of input and output layers and at least one hidden interconnection layer. A general architecture of ANN with N1 inputs, N2 outputs and L hidden layers is depicted in the following Figure:

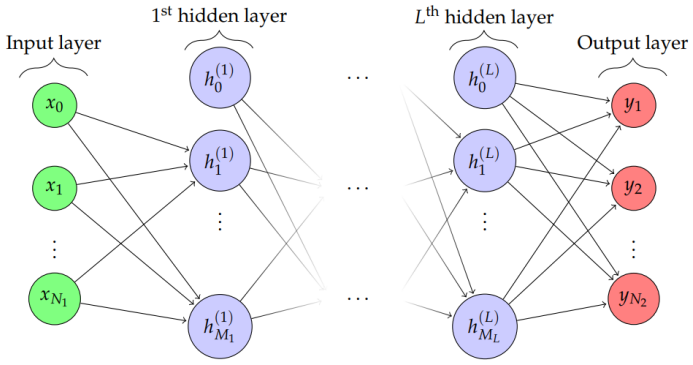

Figure1: Network graph of N_1 input units, N_2 output units and L-layer perceptron where each hidden layer contains M_j hidden units

ANNs can manipulate information just like the human brain thanks to the computational features of their basic units (also called nodes or neurons) which take a set of inputs, multiply them by weighted values and put them through an activation function. The schematic structure of the i^{th} hidden artificial neuron at the j^{th} hidden layer can be depicted as the following Figure:

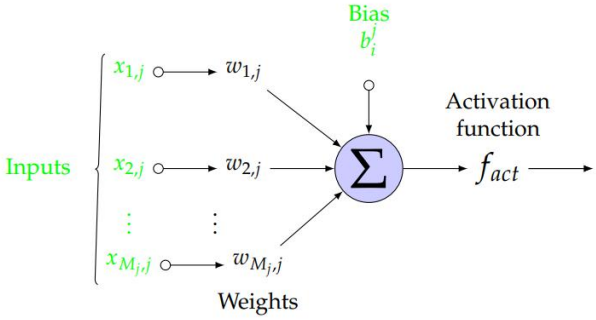

Figure 2: Structure of a single artificial neuron in a neural network.

There are several topologies of NNs in deep learning and they can be classified into two groups of algorithms. The~first group contains the ones that were used for supervised deep learning problems such as fully-connected feed-forward algorithms (Multi-Layer Perceptron, Radial Basis Network, etc.), recurrent NNs algorithms (long short term memory, gated recurrent unit, gated feed-forward, etc.) and convolutions NNs algorithms (deep convolutional NNs, deep convolutional inverse graphics network, deconvolutional network, etc.). The second group contains the ones that were used for unsupervised deep learning problems such as restricted Boltzmann machine algorithms (deep belief network, deep Boltzmann machine, etc.) and ML auto-encoder algorithms (variational auto-encoder, denoising auto-encoder, sparse auto-encoder, etc.).

Data collection:

The first and the most important step in the supervised learning process is gathering the data. In other words, to carry out good training, vast amounts of real-world data (Big Data) is required since the more data we provide to the system, the faster the model can learn and improve. Besides, the collected dataset should be well distributed throughout the operation range so as to represent the behaviour of the fuel cell in each operating power point.

Inputs and outputs selection:

Another factor that can improve the accuracy of the learned function is the selection of the inputs and outputs since the accuracy is strongly dependent on how the inputs are represented. The inputs should be entered as a feature vector that contains enough information to properly predict the output; but also, it should not be too large due to the dimensionality curse effect.

Data division (training, validation and test):

When enough data is available, the next step is to split this data into three subsets which are training, validation and test. The training dataset needs to be fairly large and contains a variety of data in order to contain all the needed information. Many researchers have proposed a training set of 70%, 80% and 90%; where the rest of data were divided between the validation and test.

Designing the Network

Based on Figure 2, the output of the h (j) i hidden layer unit can be calculated as:

$$h_i^{(j)} = f_{act}\left[\left(\sum_{i=0}^{M_j} w_{i,j}x_{i,j}\right) + b_i^j\right]$$

where, j = [1,2, . . . , L] refers to the j^{th} hidden layer, i = [1,2, . . . , M_j] refers to the i^{th} neuron in the hidden layer j, M_j = [M_1,M_2, . . . , M_L] refers to the number of neurons at each layer, x ∈ Rm are numerical inputs, w ∈ Rm are weights associated with the inputs, b ∈ R are biases. fact is the activation function which is used to introduce nonlinearity into the output of the artificial neuron. Actually, this is important since most of data in the real world is nonlinear and the neurons should learn these nonlinear representations. There are many activation functions that can be used in practice such as sigmoid, tanh, ReLu, etc.. The k^{th} output layer unit can be calculated as:

$$y_k = f_{act}^{out}\left[\left(\sum_{i=0}^{M_L} w_{i,L}h_i^{(j)}\right) + b_k\right]$$

where k = [1,2, . . . , N_2] and f_{out}^{act} is the output transfer function.