ET-287 – Signal Processing using Neural Networks

3. Feedforward Neural Networks

Professor Sarah Negreiros de Carvalho Leite
Aeronautics Institute of Technology
Electronic Engineering Division
Department of Telecommunications

sarahnc@ita.br, sarah.leite@gp.ita.br
room: 221

Course Syllabus

Unit 1 – Introduction and brief review of linear algebra using Python.

Unit 2 - Introduction to Neural Networks.

a) What is an artificial neural network?

b) The human brain and models of a neuron.

c) Artificial intelligence and neural networks.

d) Neural network architecture.

e) Activation functions.

Unit 3 - Feedforward Neural Networks.

a) Perceptron.

b) Multilayer perceptron.

c) Neural networks with radial basis activation function.

d) Extreme learning machines.

e) Convolutional neural networks.

Unit 4 - Recurrent Neural Networks.

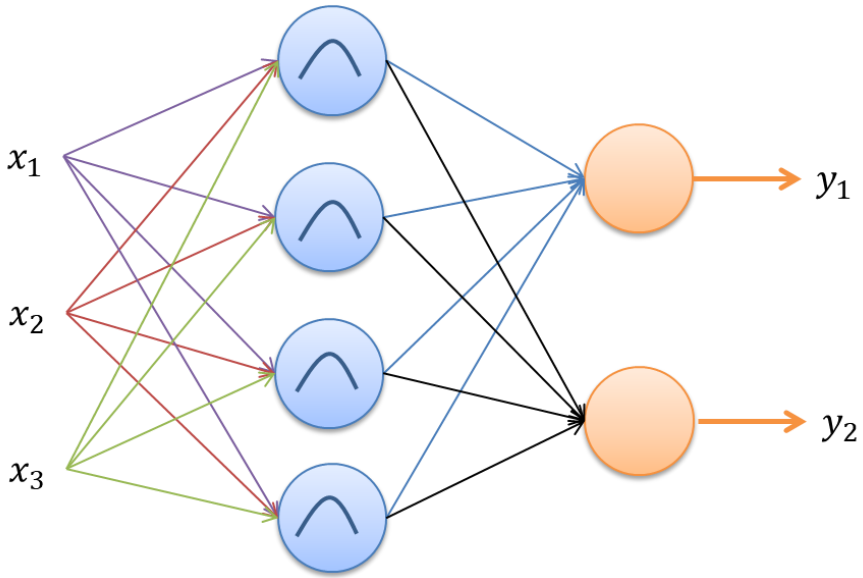
Unit 5 – Considerations about Deep Learning.

Radial Basis Function Networks

Radial Basis Function Networks

The Radial Basis Function (RBF) Network is also capable of solving classification problems and approximating functions.

Unlike MLP, RBF networks typically have only one intermediate layer where a radial basis activation function is employed, usually the Gaussian function.



Radial Basis Function Networks

The Gaussian activation function is of the form:

$$f(||x - c||) = e^{-\frac{||x-c||^2}{2s^2}}$$

x: input data vector,

c: center of the radial basis function,

s: radius of the radial basis function.

The activation of the neurons in the intermediate layer considers the distance $||x - c||$ between the input data x and the synaptic weight (center of the radial basis function) c.

Radial Basis Function Networks

A radial basis activation function is characterized by having a response that decreases (or increases) with the distance to a central point.

The model parameters include:

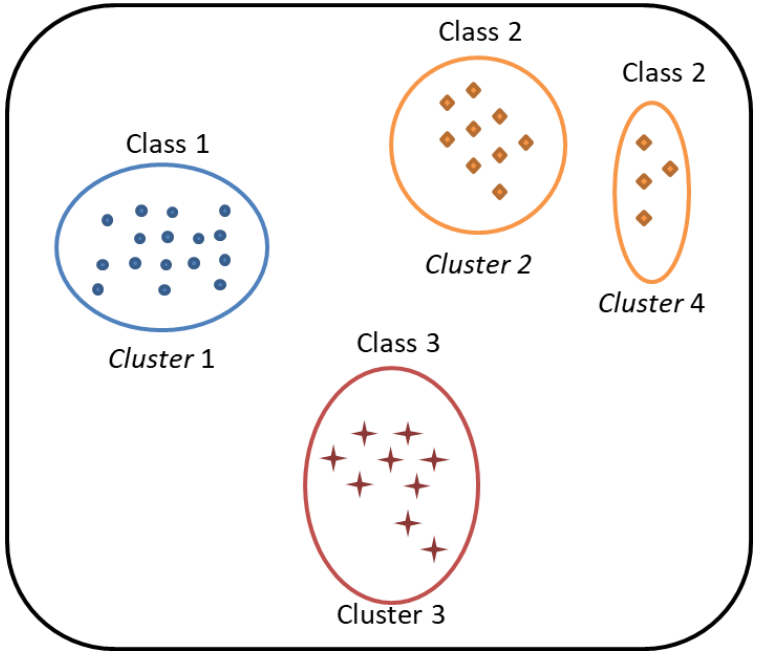
- Center
- Rate of decrease (or increase) in each direction.

Each neuron in the intermediate layer of the RBF network defines a hyperellipsoid that groups input data, constructing local approximators.

Radial Basis Function Networks

Clustering in the RBF network with 4 neurons in the intermediate layer.

The RBF network is an excellent option for well-defined problems.



RBF network training

Typically, the training of the RBF network involves an unsupervised stage and a supervised stage.

Unsupervised stage: defines the number of neurons and the parameters of the radial functions.

The classical approach involves the use of clustering techniques. Through unsupervised learning, the centers of the clusters are positioned in regions where the input data is concentrated.

Traditionally, the *k*-means algorithm is used, allowing the partitioning of the pattern space into *k* groups.

K-means algorithm

1. Arbitrarily choose *k* input vectors and set them as the *k* centers.
2. Evaluate all other input vectors and associate each one with the nearest group.
3. Calculate the new positions of the *k* centers.
4. Repeat step 2 until convergence is reached. If a given input vector is not allocated to the nearest group, it must be reallocated, and the centers of both groups recalculated.

There is no rule to determine the value of *k*.
One option is to choose *k* as the number of classes in the problem. However, *k* can also be a larger value.

RBF network training

Once the input vectors belonging to each cluster are established, it is possible to determine the center (mean value) and the radius (standard deviation) of each group.

Supervised stage: Responsible for adjusting the synaptic weights of the neurons in the output layer. The first layer of the RBF network groups input data into clusters, transforming potentially nonlinear input data into linear form. This way, linear classification or regression algorithms can be applied in the output layer of the RBF network.

Exercises

1. Consider the sine function approximation problem again, this time considering an RBF.

Extreme Learning Machine

Extreme Learning Machine (ELM)

Feedforward neural networks produce an output that is a linear combination of the activations of neurons from the previous layer.

ELM takes this result to the extreme. In contrast to classical approaches that employ heavy algorithms to determine the weights of each neural network neuron, **in ELM, the weights of the hidden layer are assigned randomly**, generating a linear system whose solution represents the weights of the output layer.

It is immediately intuitive that an arbitrary assignment of weights to the neurons in the hidden layer allows for a **faster** structuring of the neural network. On the other hand, the **number of neurons** required for a good solution and generalization of the network is **much higher**, potentially exceeding the number of available training samples.

The choice of the ideal number of neurons to compose the hidden layer can be determined through cross-validation methods.

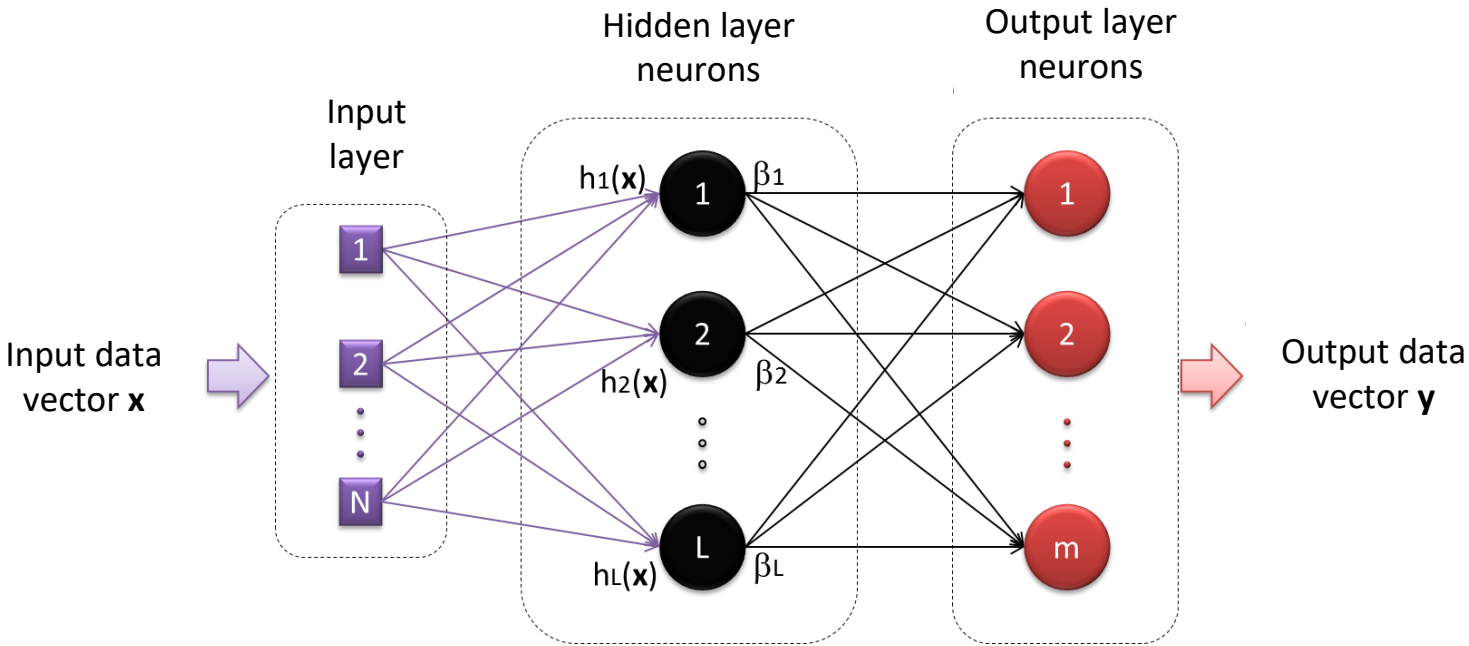
ELM

The ELM network, like MLP and RBF, also possesses **universal approximation** capability and can be employed to solve problems of various natures, such as classification, regression, clustering, sparse approximation, compression, feature learning, among others.

This network is a **highly efficient** machine learning method, as it does not use an iterative training process for parameter adjustment, making the training phase of the network quite fast. Additionally, its implementation is **easy** to understand and can be used to solve complex problems.

ELM

The network has *N* inputs that are mapped to *L* neurons in the hidden layer. The outputs of the neural network are determined by *m* neurons in the output layer.



Implementation of an ELM

1. Create the matrix of random weights W for the hidden layer. This matrix have dimensions $(L+1) \times N$, where L is the number of hidden layer neurons plus 1 to account for the bias and N is the number of inputs.

2. Assemble the intermediate layer matrix given by:

$$H_{int} = X.W^T$$

where X is the feature matrix of the neural network input data.

3. The matrix H_{int} undergoes the activation function and generates the matrix H. From it, we can determine the output weight matrix of the ELM neural network:

$$W = H.Y$$

4. With the weight array completely determined, we can now use the test data to determine the network output:

$$Y = H.W$$

Exercises

Implement an ELM network to address the iris plant classification problem once again. Compare the three solutions that you proposed, making considerations regarding the accuracy of the classification system and the computational cost.

Exercises

Consider the breast cancer diagnosis problem using an ELM network. Build an ELM network that takes the features of abnormal cells as input and produces the diagnosis as output: benign or malignant tumor. Observe how the parameters of the neural network affect the diagnostic performance:

i. Number of neurons in the hidden layer,
ii. Activation function,
iii. Learning rate.

With the parameters fixed, do a 5-cross fold validation and calculate the mean and standard deviation of the network's accuracy rate. What can be done to reduce performance variability?