NEURAL NETWORK FROM SCRATCH USING NUMPY

Introduction and Objective

In this part a neural network will be implemented using NumPy, a basic library of python. The neural network will have three layers, the input layer, a hidden layer and the output layer. This network will be a multi-class classifier and for the purpose of the exercise there will be three output classes.

Methods

The implementation of this network will have two principal parts: first we will define some internal functions that will help us clarify the code and then we will create a model class with several methods.

The class NN (Artificial Neural Network) has two methods:

First the reserved method __init__, this method is called when an object from the class is created.

Then, the method fit computes the training of the model, inside this method various tasks are performed, such as train/test/validation splitting, weight initialization, forward pass function calling, backpropagation, and computation of the cost. The validation cost will be calculated every 10 iterations and the test cost will be calculated after all the training is done.

There are two other classes, one for each layer type: hidden_layer class and Output_layer class. Both classes have similar methods, initialize_weights method takes the number of hidden neurons and output neurons and initializes the weights randomly. Forward method computes the forward propagation and gives Y or Z vector depending on the layer type. Backprop method performs the backpropagation outputting the derivatives. Those derivatives will be the input for the next method, update_weights, which will multiply the derivatives with the learning rate and perform the update.

Apart from those methods, there are other external functions that help us to keep the code clear:

- ***Get_data() and Get_data_2()*** Functions that load the datasets, split it in features and target and normalizes X, the feature vectors.
- ***Encode_labels(Y)*** Function that applies a one hot encoder to the target or to the labels.
- ***Y2indicator(Y)*** Function that outputs the index of the Y vectors.
- ***Cost(T,Y)*** Function that computes the categorical cross entropy from the prediction and the label.
- ***Error_rate(T,Y)*** Function that computes the error rate from the target and the prediction as np.mean(Target!=prediction)
- ***Sigmoid(a) / Softmax(a)*** Computation of the sigmoid and softmax functions
- ***Accuracy(T,Y)*** Function that calculates the accuracy as np.mean(T == Y)

Results

We are going to evaluate this neural network with two datasets loaded from sklearn: Iris dataset where three classes of iris flowers are presented. It has four features: sepal length, sepal width, petal length and petal width.

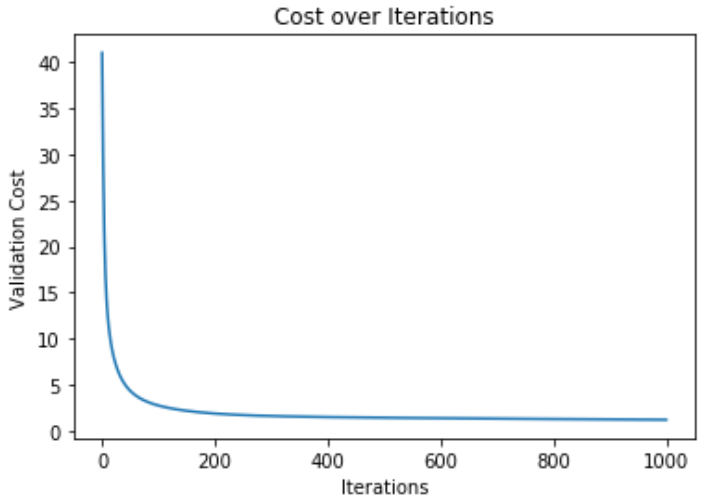
Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

And Wine dataset where three different wine classes will be classified depending on 13 features: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium and so on.

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

The first dataset has 150 samples so we will use 20 samples for validation, 20 samples for testing and the rest for training.

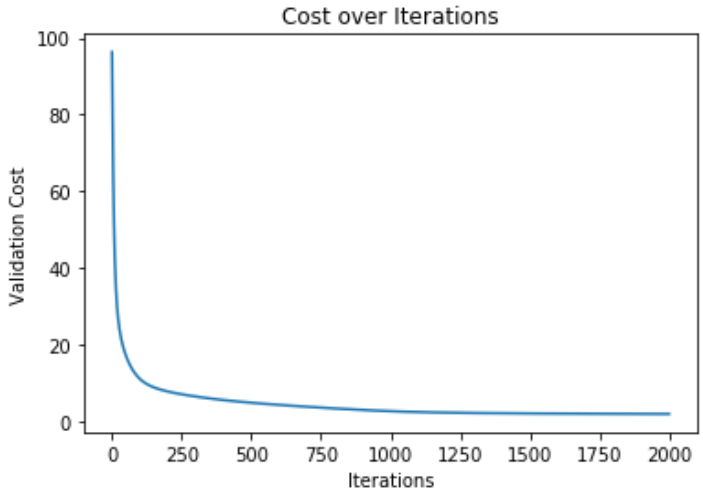
After 10000 iterations, the model achieved a 100% accuracy with a loss of 1.21 and the test set also got a 100% accuracy with a loss of 1.77, which means that 20 out of 20 testing examples were correctly classified. In the next graph one can see the cost over all the iterations:



The second dataset has 178 samples, and this will be divided in the form of 20 validation samples, 20 testing samples and the rest for training.

After 20000 iterations the model is capable of achieving a 95% validation accuracy, which means classifying correctly 19 samples out of 20. In the test set, the network had an accuracy of 90% (18 of 20).

The next graph shows the cost over iterations in the validation of the second dataset:



Discussion

In this project I it has been shown how to implement a neural network from scratch using object-oriented programming.

In order to create a larger network, some more advanced programming should be done. Future work could be to receive the number of hidden layers from the user and create a network depending on that decision.

References

1) https://www.kaggle.com/

2) https://scikit-learn.org/stable/

3) https://www.tensorflow.org/tutorials/keras/classification