01/06/2018

Pattern Recognition - Homework 2

Author: Ali Berrada

Program: MAIA 2017/19

Notes on attached Python scripts:

- problem1.py : the code for Problem 2.1 that computes the accuracy for each combination from the options. The data it is fed are “data/train_problem1.txt” and “data/test_problem1.txt” which were obtained from the original data after splitting into 75%-25% and then shuffling the train set.

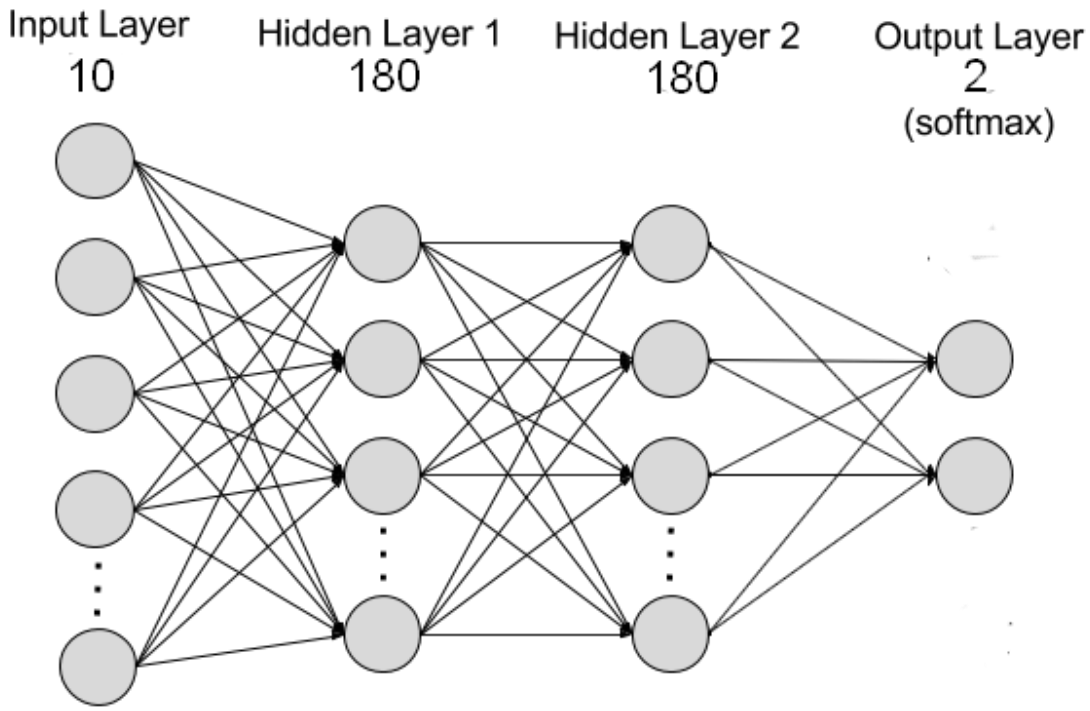
- problem2_cross-data-gen.py : the code for Problem 2.2 to generate different train/test splits for cross-validation.

- problem2_best-accuracy.py : the code for Problem 2.2 that computes the accuracy for each combination from the options with cross-validation approach. The data it is fed are “data/train_<i>.txt” and “data/test_<i>.txt” which were obtained from “problem2_cross-data-gen.py”. The results (accuracies, models, AUC plots) for all the experiments are saved in an auto created folder within the same directory.

- problem2_mymodel/mytest.py : the code for Problem 2.2 that loads the saved model weights and computes the accuracy given a data set. The labels of the data are automatically replaced with “1” if the labels are strictly greater than 0 and with “0” if the labels are less or equal 0. The only variables to change are “testFile” and “delimiter”.

Problem 2.1

The architecture of the network I have used is as follows:



As it can be seen, the input layer accepts a sample of 10 features. The 2 hidden layers consist of 180 nodes applying either the ReLu or tanh activation function. The output layer is a softmax function with 2 possible classes (‘0’ or ‘1’) applying a cross entropy loss function.

Depending on the experiment, a batch normalization was performed after the input layer and after the first hidden layer.

The network weights were updated using the stochastic gradient descent optimizer with a learning rate of 0.01 for 50 epochs and with a batch size of 36.

The network was trained with 75% of the total dataset and the rest was used to test the accuracy. Combinations between different options gave the following results for 12 unique experiments:

Weights initialization	Batch normalization	Activation	Accuracy (%)
Zeros	**Yes**	**relu**	50
		tanh	50
	No	**relu**	50
		tanh	50
Random	**Yes**	**relu**	58.9
		tanh	76.7
	No	**relu**	75.7
		tanh	59.35
Xavier	**Yes**	**relu**	84.8
		tanh	83.5
	No	**relu**	78.45
		tanh	76

With weights initialized to zero we are achieving the worst accuracy. It is 50% in this situation because the data is well balanced between the 2 possible classes and the network always gives the same prediction. This type of initialization is not appropriate as it serves no purpose to cancel the activations which in turn result in no gradient decent update during the backpropagation phases.

The initialization following uniform Xavier distribution gave better results in comparison with uniformly distributed random weights.

Batch normalization, overall, improved the accuracy except in the case where random weights and ReLu were used at the same time

The highest accuracy (84.8%) was obtained with a combination of Xavier weight initialization, batch normalization and ReLu activation. The AUC was 0.93 as can be seen in the figure below. From multiple runs with different training sets, this combination of options can lead to accuracy above 85%.

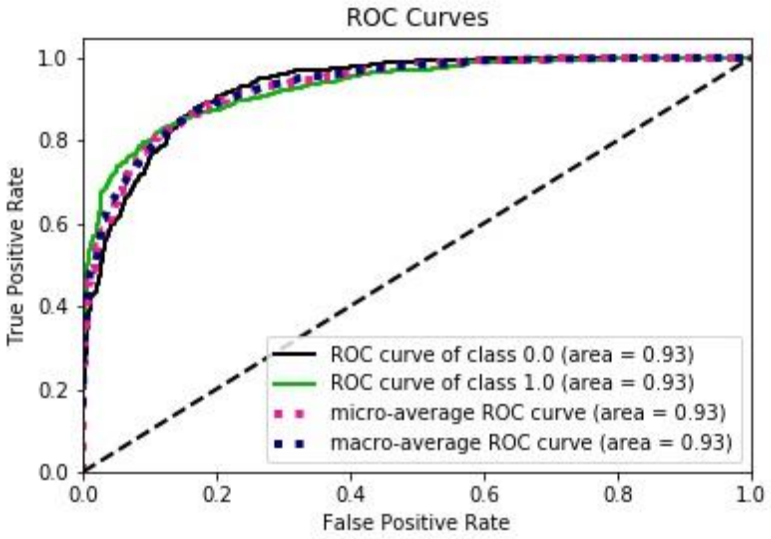


Figure 1 : Experiment with Xavier + Batch Normalization + ReLu

As a side information, the same experiments were made with the Adam optimizer (learning rate = 0.001) and the results were slightly higher for almost all the experiments. With Adam optimizer, the combination of Xavier weight initialization, batch normalization and ReLu activation (as discussed earlier) gave on average an accuracy of 84.23% while with SGD it was very low. This only shows that when it comes to fine tuning in the field of machine learning, we need to rely on different experiments before choosing the final network configuration.

Problem 2.2

For this problem, a 5-fold cross-validation was performed leading to the following results:

Weights	Batch norm	Activation	Cross-1	Cross-2	Cross-3	Cross-4	Cross-5	Average
Zeros	**Yes**	**relu**	50	50	50	50	50	50
		tanh	50	50	50	50	50	50
	No	**relu**	50	50	50	50	50	50
		tanh	50	50	50	50	50	50
Random	**Yes**	**relu**	61.06	57.87	60.25	58.75	59.5	59.48
		tanh	77.75	77.5	76.62	77.87	76.62	77.27
	No	**relu**	77.25	74.31	76.75	77.81	72.87	75.8
		tanh	58.62	58	60.18	59.12	45.62	56.31
Xavier	**Yes**	**relu**	84.25	84.56	85.68	83.81	83.87	84.43
		tanh	82.81	84	83.625	83	84.5625	83.6
	No	**relu**	79.25	78.56	77.75	79	78.625	78.63
		tanh	77.75	76.87	76.75	77	76.875	77.05

These results confirm the previous discussion and therefore the model I will choose to be run on new data is the Xavier initialization with Batch normalization and ReLu activation as performed on the 3rd cross-validation data.

Testing on the original 8000 data samples, the accuracy obtained was 85.91%.

On “unseen before” dataset of also 8000 samples generated from the same distribution as the original, the accuracy was 79.43%