Computer Programming Report

肖易佳　计 *83*　*2018011347*

Preparation

Mechanism

The classification task can be converted as: given an input vector of 10 dimension, we need to output +1 or -1 by using the perceptron.

Algorithm

(a) parameters: η, w(a vector), b

(b) Initialize: initialize w and b using random data

(c) Load data: load training data and test data, and split training data into 2 subsets - training and validating, so that we can know whether the model is well-training before really applying it to test set.

(d) Model:

$$y = wx + b$$

(e) Training: we need to update the parameters in this section. First, there will be some instances mis-classified by our perceptron, our goal(loss function) is to minimize the sum of distances of all those mis-classified. So we define:

$$LOSS(w, b) = \sum_{iinM} -y_i * (wx_i + b)$$

Where M = {x| x is mis-classified instances}

Learning outcomes:

Author(s): Yijia Xiao

(f) Update: Using gradient descend, we need to find the position where $\frac{\partial L(w,b)}{\partial w} = 0$, and $\frac{\partial L(w,b)}{\partial b} = 0$.
Then we can update w and x:

$$w = w + \eta * x_i * y_i$$

$$b = b + \eta * y_i$$

(g) End training: we can end when accuracy $> \epsilon$ or Loss $< \Delta$; we can also quit training as said in exercise instruction - end at random(but this cannot guarantee the best result).

Experiment 1

(a) On TrainingSet-2 and On TestSet: When we change the learning rate (in my test, from 1e-5 1e-3), the error rate on TestSet is more stable than that of TraingingSet-2.
I think the reason is that the scale of TestSet is bigger than the TrainingSet-2.

(b) Comparison of **Fixed increment rule** and the **Variable increment rule**: When we use the fixed learning rate, the problem is that we have to first find the best learning rate(usually by fixing a step and use for-loop to find the best rate), and besides that, if the training epoch number is too large, the model may just go past the best state, as we can see in the text files I uploaded: in some files the loss function and learning rate will increase after a certain point, and then converge again.

Experiment 2

(a) Number of training epochs: I selected to train 10000 epochs, and output the accuracy and loss on validation set every 100 epochs; as for

random epochs, I just need to keep the average of number at 10000. My observation: If we train the model a fixed number of rounds, then the accuracy mainly rely on the size of training data and the similarity between training and testing datasets(maybe we can use cross-entropy, but I am not sure); and as for the random epochs training, the accuracy and loss will fluctuate, based on the actual training rounds(I think this is similar to the previous problem - the accuracy is not stable). When training set is bigger, and more similar to the test set, the training accuracy is higher, and loss function is usually smaller.

(b) Another finding: In the experiment, I divided the training set into two subsets - one is the actual training set and the other is validation set. I find if the set is small, like 40(Set-No.2), it is usually useless to perform the validation process, because the learning rate I use is relatively small, so the the model is just not well-trained in such a small set(25 to 35 instances), if we test the trained model on the validation sets, the results may just be the same, because the model's parameters are undertrained.