(1) (1.0 pt) Which logical operation can not be realized by one hyperplane?

1- and

2- or

3- xor

(2) (1.0 pt) which option is <u>not</u> correct about bagging:

1- each base learner is trained on a sample which on average contains 36.8% repeated examples

2- the examples misclassified by the former base learners will be emphasized in the latter ones

3- each sample contains the same number of examples as the given training data

(3) (1.0 pt) Diversity among base learners in random forest is higher than in bagging, since in each node of random tree the split point is chosen :

1- using a random subset of the examples

2- randomly

3- among a random subset of the features

(4) (1.0 pt) In Adaboost, the weight correspond to each base learner in the final decision, is higher:

1- if the weighted error of the base learner is higher

2- if the weighted error of the base learner is lower

3- either of 1 or 2 might happen

(5) (1.0 pt) The Adaboost algorithm (slide 21) can not continue if the weighted training error of:

1- a base learner approaches to zero

2- the base learner is lower than 0.5

3- the ensemble approaches to zero

(6) (5.0 pt) For the given data, implement the Adaboost algorithm (slide 21), and choose the option that represents the accuracy obtained in the last iteration. Consider the base learners as perceptron (as you have implemented in question 2 of exercise 3). For your implementation use the following information:

Perceptron :

In each iteration compute the examples that have been misclassified and from them take the one which correspond to the maximum weight given by Adaboost (in the case of having more than one maximum take the one that is coming earlier in the data).

The weight update in Adaboost is: $D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$

The weights are initialized to zero.

Perceptron should run 201 iterations.

Adaboost:

The weights initialized to one.

Number of base classifiers: 4

Note : consider learning the bias (augment your data with a column of ones).

Data:

x1 = ([.1,.2,.4,.8, .8, .05,.08,.12,.33,.55,.66,.77,.88,.2,.3,.4,.5,.6,.25,.3,.5,.7,.6])

x2 = ([.2,.65,.7,.6, .3,.1,.4,.66,.77,.65,.68,.55,.44,.1,.3,.4,.3,.15,.15,.5,.55,.2,.4])

labels = ([1,1,1,1,1,1,1,1,1,1,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1])

1- 0.94

2- 0.91
3- 0.96