1 Boosting with Decision Stump

Consider the following 2-class binary problem

\[ C_{y=-1} = \{(-3, -1), (-3, 1), (3, -1), (3, 1)\} \]
\[ C_{y=1} = \{(-1, -1), (-1, 1), (1, -1), (1, 1)\} \]

Using the error and weight update rule of the discrete AdaBoost, answer the following question

a) What are the first 2 decision stumps, and what are their corresponding thresholds.

b) Are these sufficient to obtain perfect classification?

Solution

a) First decision stump: \( \varphi_1(x) = \begin{cases} 1, & \text{if } x > -2 \\ -1, & \text{otherwise} \end{cases} \)

Second decision stump: \( \varphi_2(x) = \begin{cases} 1, & \text{if } x < 2 \\ -1, & \text{otherwise} \end{cases} \)

b) Not with Discrete AdaBoost. The final vote is obtained by linear combination of the weak learners. Regardless of the choice of weights assigned to each learner, the vote for the negative (black) samples on one of the two sides will be positive (the classifier
function will be equal to 0 if all the weak learner weights are equal for instance). You need at minimum 3 weak learners to correctly classify this problem so that the third weight can play the role of a bias (a decision stump with all the points on one side of it, with a weight equal to the bias).
2 Weak Classifiers

Selecting the proper model of weak classifier is often a very important step when tackling new classification problems. Presented below are a number of such problems; for each dataset:

a) Propose a weak learner that will excel in classifying the data

b) Estimate the number of weak learners necessary to perform the task optimally
Solution

Figure 1:

a) The easiest solution would be using random rectangles or decision stumps, as they will perfectly model the straight boundaries between the two classes. However, if the number of weak learners generated is too small, no rectangles might fit perfectly the boundaries of each 'cell'. In this case a "Random Circles" might better do the job as it is more robust to the initial random generation.

b) Assuming that a sufficient number of weak learners have been generated, the system could perfectly classify the example using 12 random rectangles (there are 12 red squares and 13 white squares, if you put equal weight one for each weak classifier on the red squares, there is no need for a bias w.l.) or 9 decision stumps (4 for each axis and one bias.)

Figure 2:

a) Random Circles are perfect for this task as they perfectly model the radial expansion of the classes.

b) Three boundaries need to be modeled and 3 weak learners are sufficient to properly classify this data (on the contrary to Exercise 1, there is already an odd number of weak learner therefore no other one is required to add a bias.)

Figure 3:

a) Random Projection would allow to 'cut the edges' of the negative (white) class while maintaining perfect classification of the positive class. Random rectangles are another possible choice.

b) A minimum of 3 weak learners would be necessary to solve correctly this problem (see Exercise 1). One random rectangle is also enough.

Figure 4:

a) Random rectangles would be a good starting point, if sufficient examples are generated. Otherwise, Decision stumps are well fitted to this problem.

b) A single random rectangle, if placed perfectly would do the job. Alternatively, 6 decision stumps would also suffice (3 to classify horizontally, 3 to classify vertically, see Exercise 1).

3 The AdaBoost error function (to be done at home)

By differentiating the error function

\[ w_i^{(t+1)} = w_i^{(t)} \exp \left( -\frac{1}{2} \alpha_j y_i \varphi_j(x_i) \right). \]  \hspace{1cm} (1)

with respect to \( \alpha_j \), show that the parameters \( \alpha_j \) in the AdaBoost algorithm are updated using

\[ \alpha_j = \ln \left( \frac{1 - \epsilon_j}{\epsilon_j} \right). \]  \hspace{1cm} (2)
in which \( \epsilon_j \) is defined by

\[
\epsilon_j = \frac{\sum_{i=1}^{N} w_i I(\varphi_j(x_i) \neq y_i)}{\sum_{i=1}^{N} w_i} = \frac{\sum_{i: y_i \neq h(x_i)} w_i}{\sum_i w_i}.
\] (3)

**Solution**

If we differentiate (1) w.r.t \( \alpha_j \) we obtain

\[
\frac{\partial E}{\partial \alpha_j} = -\frac{1}{2} \sum_i w_i y_i \varphi_j(x_i) \exp \left( -\frac{1}{2} \alpha_j y_i \varphi_j(x_i) \right)
\] (4)

Setting this equal to zero and rearranging, we get

\[
-\frac{1}{2} \left( \sum_{i: y_i = h(x_i)} w_i \cdot e^{-\frac{\alpha_j}{2}} - \sum_{i: y_i \neq h(x_i)} w_i \cdot e^{\frac{\alpha_j}{2}} \right) \frac{1}{\sum_i w_i} = 0
\] (5)

By replacing with the definition of the error:

\[
-\frac{1}{2} \left( (1 - \epsilon_j) e^{-\frac{\alpha_j}{2}} - \epsilon \cdot e^{\frac{\alpha_j}{2}} \right) = 0
\] (6)

\[
e^{\alpha_j} = \frac{1 - \epsilon_j}{\epsilon_j}
\] (7)

from which (2) follows directly.