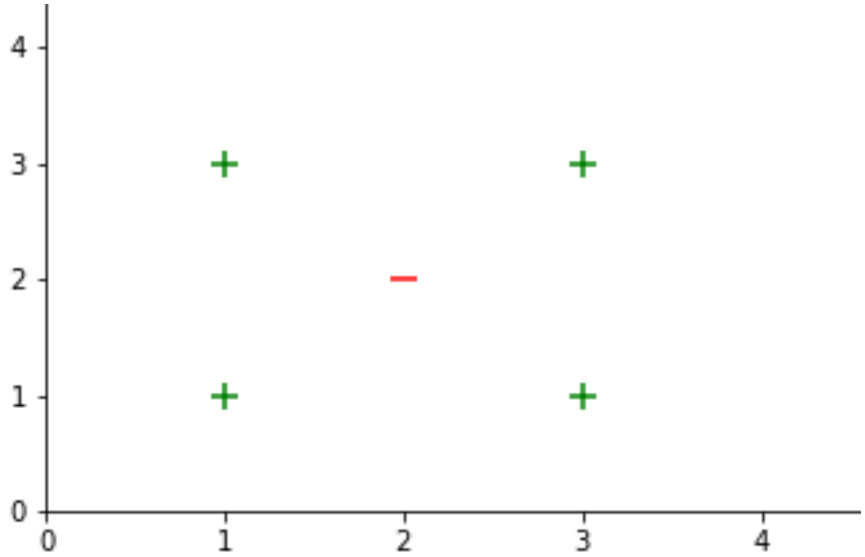
Boosting Decision Stumps



While studying for a Machine Learning exam, I came across this fascinating question. I say it's fascinating because of the amount of time it made me think about it and by doing so – gaining a deeper understanding in the specific algorithm used to solve the problem. In addition, it's such a simple question… and that's why I think its fascinating.

The question went something like:

"By using only Decision Stumps as classifiers, are you able to reach a zero-classification error on the data set above? If yes – how many iterations would it require? If not – explain why."

If you'd like to see the solution, skip to page 5.

Throughout pages 2-4 I explain the process done to figure out the solution.

Enjoy.

At first, thinking about simple decision boundaries around the ' – ' placed in the centre (2, 2) was the easy solution, but we have the Decision-Stump constraint. Then, I started thinking how AdaBoost (boosting algorithm which uses weak learners such as Decision-Stumps every iteration) could be used to reach the zero-classification error. Afterall, the only 4 decision stumps I thought about at first were: vertical line at 1.5, classifying all the points to the right as ' – ' and the points to the left as ' + ' (= weak learner, classification error of 40%), second would be a vertical line at 2.5, third a horizontal line at 1.5 and the fourth would be a horizontal line at 2.5. Together, I assumed using AdaBoost, id reach a zero-classification error. This assumption led me to calculate the 4 iterations manually, thinking it would be great preparation for the exam. I used the following algorithm:

Input:

- Training set $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$
- A weak learner generator: WL
- Number of rounds T

Initialize $D^{(1)} = (\frac{1}{m}, \dots, \frac{1}{m})$

For $t = 1, \dots, T$:

- Invoke a weak learner: $h_t \leftarrow WL(D^{(t)}, S)$
- Compute: $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} 1_{[y_i \neq h_t(x_i)]}$
- Let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$
- Update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))}$

Output hypothesis $h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x))$

After calculating the 4 iterations, I found out I was still unable to classify 2 points from the dataset correctly using the final hypothesis. Resulting in a 40% classification error.

This made me dive into the internet looking for examples of datasets which AdaBoost can't reach a zero-classification error. The only examples I found were datasets which failed since it was impossible to invoke a weak learner for such a data set (usually the best was a 50% error – not a weak learner…). But, for our dataset we can invoke a weak learner, therefore, I concluded we must be able to reach the desired zero-classification error.

This is when I dove into the mathematic proofs presented in class, finding a very helpful Theorem which states:

Theorem 1: Let S be a training set and assume that at each iteration of AdaBoost, the weak learner returns a hypothesis for which $\epsilon_t \leq \frac{1}{2} - \gamma$. Then, the training error of h_s - the output hypothesis of AdaBoost is at most

$$L_s(h_s(x)) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}[y_i \neq h_s(x_i)] \leq \exp(-2\gamma^2 T)$$

Bounding the error was just what I was looking for!

So, I thought to bound it by 0, unfortunately the exp function never reaches 0. But this brought me across a great trick. Our dataset, size m, in our case containing m = 5 points – can only have a classification error of: {0, 0.2, 0.4, 0.6, 0.8, 1}, so if we would bound the classification error by less than 0.2, we would reach the next value in the set – which is 0!

Now I believed its possible to be done, but gamma was a tricky parameter to find because of the epsilon which was changing every iteration based on the update of the vector D (see the second line in the for loop from the algorithm above). So, I implanted the algorithm and hoped to find that after a certain amount of iteration id reach the desired zero-classification error.

This didn't happen. Epsilon was converging to 0.5 based on the 4 weak learners I presented above. Meaning we need to use a very small gamma, causing the T (number of iterations) to grow as well.

This can be seen from the equation reached from developing the above bound to isolate T:

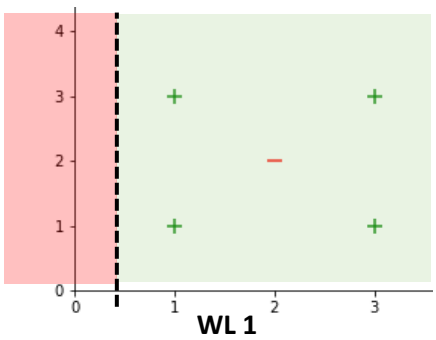
$$\boldsymbol{T \geq \left\lfloor 1 + \frac{\log m}{2\gamma^2} \right\rfloor}$$

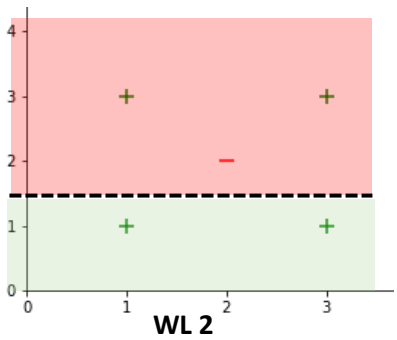
On the one hand I was a little frustrated for not yet finding the number of iterations needed (I want to say I still had faith its possible, but I started questioning a few assumptions I made a long the way…) On the other hand – I became familiar with the inner workings of AdaBoost, the output hypothesis and intuition behind boosting algorithms, so I was overall happy.

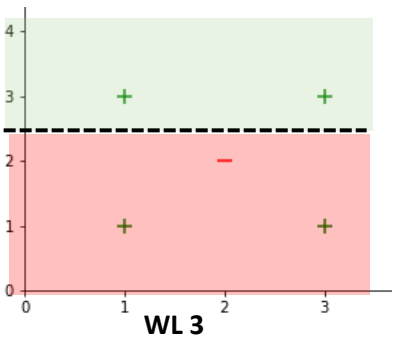
I needed to think outside the box, so I decided to run SciKit-Learn's AdaBoost for 1 iteration and see which weak learner they invoked at first… surprisingly, they reached only a 20% error after 1 iteration, and that solved it all for me.

There was always another Decision-Stump I didn't think about yet, classifying all the points as ' + ' and reaching a 20% classification error. By using this weak learner as the first weak learner in the AdaBoost algorithm – we could essentially break the balance created by the 4 weak learners I was using until now.

The algorithm reached the zero-classification error after 3 iterations. Using the following weak learners and calculating the vector D as shown in the algorithm above:

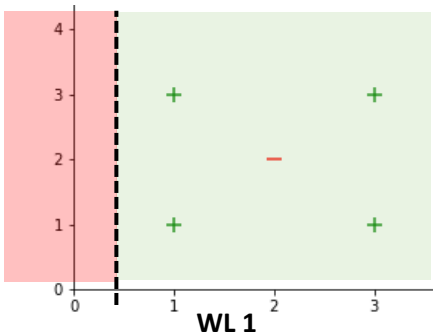


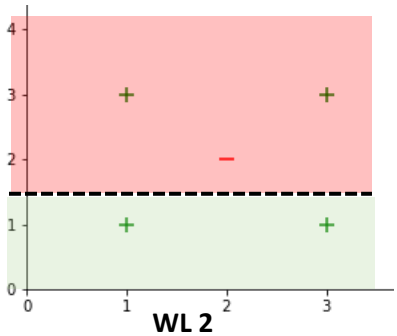


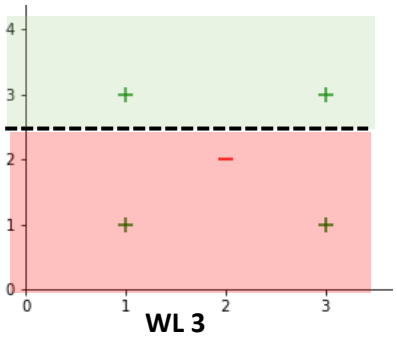


Solution

It is possible to reach a zero-classification error using only Decision-Stumps as classifiers.
After 3 iterations, by running AdaBoost and invoking the following Decision-Stumps:







We successfully achieve a zero-classification error.

Output details (based on AdaBoost algorithm from page 2):

```
t = 1:
        Hypothesis: [1 1 1 1 1]
        Epsilon: 0.2
        W: 0.69
t = 2:
        Hypothesis: [ 1 -1 -1  1 -1]
        Epsilon: 0.25
        W: 0.55
t = 3:
        Hypothesis: [-1  1 -1 -1  1]
        Epsilon: 0.17
        W: 0.8

Output:
        Value each point      >>> [0.44, 0.95, -0.66, 0.44, 0.95]:
        Output hypothesis     >>> [ 1  1 -1  1  1]
        True label            >>> [ 1  1 -1  1  1]
        Classification error  >>> 0.0%
```

Hopefully I was able to share some of the joy I received from this beautiful question.

Machine Learning has been a significant course for me, happy to have been able to share a glimpse of it with you.

Jacob Link

August 2022