# AdaBoost

Saturday, September 29, 2018 22:01

*For the glory of God*

## What is AdaBoost ?

- AdaBoost, short for Adaptive Boosting, is one of popular boosting techniques developed in 1996.

  Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers.

  A weak classifier is simply a classifier that performs poorly but performs better than random guessing.

  e.g. Classifying a person as male or female based on their heights only.

- AdaBoost was the first really successful boosting algorithm for binary classification.

  (It was actually the best algorithm for binary algorithm ; however, its reputation fell into the shade before ANN)

- AdaBoost has been used in conjunction with other learning algorithms to improve their performance.

## Intuitive explanation on AdaBoost

Let's say you are given the task of listing out all the people in the city of San Francisco who are taller than 5'7", weigh less than 190 lbs, and are between the ages of 28 and 41. Now the problem is that you are supposed to do this without the help of machines. All you are allowed to do is take a look at the person and determine whether or not that person qualifies. How do we do it?

You may or may not be good at estimating these parameters just by looking at the person. So to improve the accuracy, you get three people to help you out. The first person is really good at guessing the height, the second person is really good at guessing the weight and the third person is really good at guessing the age. Individually, they may not be all that useful to you, because they can do only one simple task. But if you combine them together and filter out all the people, you have a very good chance of getting an accurate list of people who qualify. Wouldn't you agree? This is the concept behind AdaBoost.

↓

how does it work ? we will get there.

## How does it work ?

- In order for me to explain it, let me decompose the word Adaptive - Boosting.

a) Adaptive

- AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.
- At the beginning, we give equal weights to all of training data.

- But on each round, the weights of incorrectly classified examples are increased such that weak learner is forced to focus on the next round. For correctly classied examples are decreased in their weights.

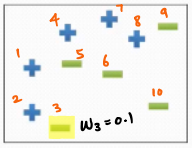
b) Boosting

- The basic concept behind AdaBoost is to create a strong classifier by the conjuncture of many weak classifiers.

Example ( Reference : Microsoft )

· Let's suppose that the training data are given for binary classification.

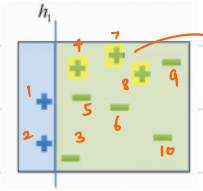
· Step 1) Set the initial weight




weight of $x_i = \frac{1}{n}$ ; where n is the number of training data

$= \frac{1}{10}$

· Step 2) Run the weak learning algorithm to get a weak classifier.




These are misclassified based on the $h_1$ boundary. ( say female with height > 170 cm )

e.g. female | male ($h_1$) ; classification rule = male if height is greater than 170 cm.

- we may need to calculate the misclassification rate ;

$$\text{Misclassification error} \equiv \frac{\sum_{i=1}^{N} w_i^{(m)} \left(y_i \neq G_m(x_i)\right)}{\sum_{i=1}^{N} w_i^{(m)}}$$

$$= \frac{(0.1 \times 1) + (0.1 \times 1) + (0.1 \times 1)}{0.1 + 0.1 + 0.1 + \cdots + 0.1} \quad ; \; N = 10 \text{ and } 3 \text{ for } y_i \neq G_m(x_i)$$

$$= 0.3$$

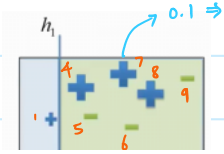
- For the next step, let's calculate the optimal weight for $h_1$ classifier.

$$\beta_{\text{optimal}} \equiv \frac{1}{2} \ln\left(\frac{1 - \text{Error}}{\text{Error}}\right)$$

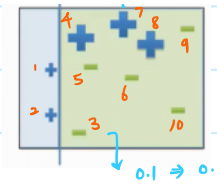
$$= \frac{1}{2} \ln\left(\frac{1 - 0.3}{0.3}\right)$$

$$= 0.42365$$

· Step 3) Update weights




{ Increase the weights on the misclassified data




Increase the weights on the misclassified data

Decrease the weights on the correctly classified data

0.1 ⇒ 0.0657 (Decrease)

- In order to update the weight, we're going to use the equation as follows ;

$$w_T^{(m+1)} \equiv w_T^{(m)} \exp\left(-y_T \beta_m G_m(x_T)\right)$$

- For the data point 4,

$$w_{4,\text{new}} = w_{4,\text{old}} \cdot \exp(-1 \cdot 0.42 \cdot -1)$$

Note that $y_4 = 1$ but $G_m(x_4) = -1$ (misclassification)

$$= 0.1 \cdot e^{0.42}$$

$$= 0.1 \times 1.52$$

$$= 0.152$$

- For the data point 1,

$$w_{1,\text{new}} = w_{1,\text{new}} \cdot \exp(-1 \cdot 0.42 \cdot 1)$$

$$= 0.1 \times e^{-0.42}$$

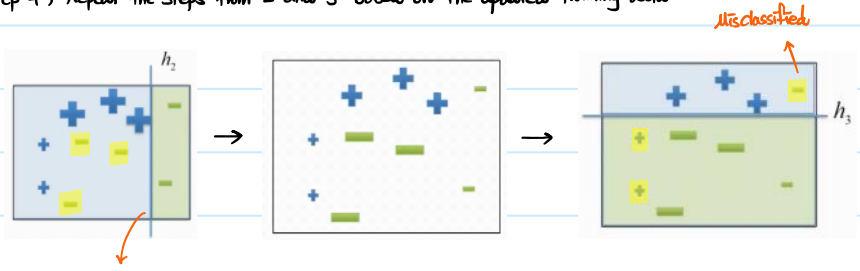
$$= 0.1 \times 0.657$$

$$= 0.0657$$

- As can be seen, Incorrect classification would receive higher weights.

↳ It prompts our classifiers to pay more attention to them in the next iteration.

· Step 4 ) Repeat the steps from 2 and 3 based on the updated training data

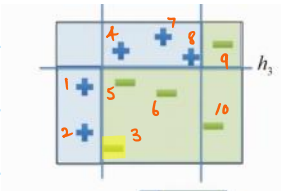



Let's say that the weak algorithm predicts the boundary based on the updated training data.

( The algorithm might try to classify the updated data point as exact as possible )

· Step 5 ) Combine all weak classifiers

- In this case, we have three weak classifiers ;

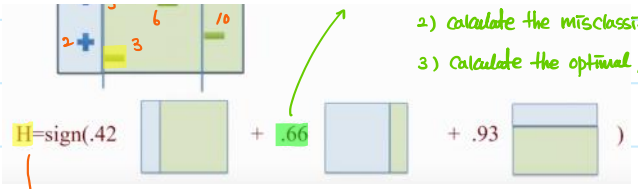



It would be calculated with the same procedure explained in step 2.

e.g. 1) classify the data based on $h_2$ classifier

2) calculate the misclassification rate

3) calculate the optimal $\beta$




2) calculate the misclassification rate

3) Calculate the optimal $\beta$

$\Rightarrow$ Strong classifier $= \Sigma \beta \cdot$ weak classifier

Say that this is a strong weak classifier. (sign means either + or -)

- Step 6) Run the strong classifier to conduct binary classification for the training data.

e.g. for the data point 3, $H = 0.42(-) + 0.66(+) + 0.93(-) = \begin{cases} + & \text{if } H > 0 \\ - & \text{if } H < 0 \end{cases}$ ; Binary classification

## Derivation of AdaBoost

- To begin with, let us suppose that we have a data set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where each $x_i$ has an $y_i \in [-1, 1]$.
- The loss function is defined as the exponential loss ;

$$L(y, f(x)) = \exp(-y f(x))$$ ; The goal is to minimize the loss function.

Note that squared error is not suitable for classification because ;

- In classification, you only care about predicting the right class.
- In regression, you need to minimize the distance between predicted and actual values.

- At step m of Adaboost, stage-wise Additive modeling is used. It generalizes the following equation ;

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$ ; where $\beta_m$ is a coeff. and $G_m$ is a classifier (or weak learner)

- Therefore, the loss function becomes ;

$$L(y, f(x)) = \sum_{n=1}^{N} \exp(-y_n f(x_n))$$ ; total error

$$= \sum_{n=1}^{N} \exp\left(-y_n f_{m-1}(x_n) - y_n \beta_m G_m(x_n)\right)$$

$$= \sum_{n=1}^{N} \exp(-y_n f_{m-1}(x_n)) \exp(-y_n \beta_m G_m(x_n))$$

$$= \sum_{n=1}^{N} w_n^{(m)} \exp(-y_n \beta_m G_m(x_n))$$ ; where $w_n^{(m)} = \exp(-y_n f_{m-1}(x_n))$

we can define these as weights since they are constant w.r.t. $\beta_m$ and $G_m$

($\because$ we're going to minimize the loss function w.r.t. $\beta_m$ and $G_m$)

- So, we are going to choose $G_m$ and $\beta_m$ such that it minimizes the loss function ;

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{n=1}^{N} w_n^{(m)} \exp(-\beta_m y_n G_m(x_n))$$

Problem 2-1.

Show that for each $\beta > 0$, the solution for $G_m(x)$ is given by ;

$$G_m = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))$$

Solution)

· Let us take a look the loss function.

$$L_m = \sum_{i=1}^{n} w_i^{(m)} \exp(-y_i \beta_m G_m(x_i))$$

$$= \sum_{y_i = G_m(x_i)} w_i^{(m)} \exp(-\beta_m) + \sum_{y_i \neq G_m(x_i)} w_i^{(m)} \exp(\beta_m) \quad ; \; y_i \neq G_m(x_i) \text{ means 'Misclassification'}$$

↓

Note that $y_i = G_m(x_i)$ means they are all either -1 or 1 ; such that multiplication turns out always positive values.

$$= \exp(\beta_m) \sum_{i=1}^{N} w_i^{(m)} I(G_m(x_i) \neq y_i) + \exp(-\beta_m) \sum_{i=1}^{N} w_i^{(m)} (1 - I(G_m(x_i) \neq y_i))$$

$$= \underbrace{(e^{\beta_m} - e^{-\beta_m})}_{\downarrow} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)) + \underbrace{e^{-\beta_m} \sum_{i=1}^{N} w_i^{(m)}}_{\downarrow}$$

It's always $> 0$ if $\beta > 0$ — It's a constant w.r.t. $G_m$ !

· Hence, in order to minimize the function,

$$\boxed{G_m = \underset{G_m}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))}$$

Problem 2-2.

Show that the optimal $\beta$ is given by $\beta_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}$

Solution)

· Let's recall the loss function.

$$L_m = \sum_{y_i = G_m(x_i)} w_i^{(m)} \exp(-\beta_m) + \sum_{y_i \neq G_m(x_i)} w_i^{(m)} \exp(\beta_m)$$

· In order to obtain the optimal $\beta$,

$$\beta_m = \underset{\beta_m}{\operatorname{argmin}} \left( \sum_{y_i = G_m(x_i)} w_i^{(m)} \exp(-\beta_m) + \sum_{y_i \neq G_m(x_i)} w_i^{(m)} \exp(\beta_m) \right)$$

· Let's set the derivative of the function in the argmin to zero and solve for $\beta_m$.

$$\frac{d}{d\beta_m} \left( \sum_{y_i = G_m(x_i)} w_i^{(m)} \exp(-\beta_m) + \sum_{y_i \neq G_m(x_i)} w_i^{(m)} \exp(\beta_m) \right) = 0$$

$$\Leftrightarrow \left( -\sum_{y_i = G_m(x_i)} w_i^{(m)} \exp(-\beta_m) + \sum_{y_i \neq G_m(x_i)} w_i^{(m)} \exp(\beta_m) \right) = 0$$

$$\Leftrightarrow \sum_{y_i \neq G_m(x_i)} w_i^{(m)} e^{\beta_m} = \sum_{y_i = G_m(x_i)} w_i^{(m)} e^{-\beta_m}$$

$$\Leftrightarrow \frac{e^{\beta_m}}{e^{-\beta_m}} = \frac{\sum_{y_i = G_m(x_i)} w_i^{(m)}}{\sum_{y_i \neq G_m(x_i)} w_i^{(m)}}$$

$$\Leftrightarrow e^{2\beta_m} = \frac{\sum_{y_i = G_m(x_i)} w_i^{(m)}}{\sum_{y_i \neq G_m(x_i)} w_i^{(m)}} \quad ; \text{ where } e^{\beta_m} \cdot (e^{-\beta_m})^{-1} = e^{\beta_m} \cdot e^{\beta_m} = e^{\beta_m + \beta_m} = e^{2\beta_m}$$

$$\Leftrightarrow \beta_m = \frac{1}{2} \ln \left( \frac{\sum_{y_i = G_m(x_i)} w_i^{(m)}}{\sum_{y_i \neq G_m(x_i)} w_i^{(m)}} \right)$$

$$= \frac{1}{2} \ln \left( \frac{\sum_{i=1}^{N} w_i^{(m)} - \sum_{y_i \neq G_m(x_i)} w_i^{(m)}}{\sum_{y_i \neq G_m(x_i)} w_i^{(m)}} \right) \quad ; \text{ where } \sum_{y_i = G_m(x_i)} w_i^{(m)} = \sum_{i=1}^{N} w_i^{(m)} - \sum_{y_i \neq G_m(x_i)} w_i^{(m)}$$

$$= \frac{1}{2} \ln \left( \frac{1}{\epsilon_{trm}} - 1 \right) \quad ; \text{ Since we define } \epsilon_{trm} = \frac{\sum_{i=1}^{N} w_i^{(m)} (y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$$

$$= \frac{1}{2} \ln \left( \frac{1 - \epsilon_{trm}}{\epsilon_{trm}} \right)$$

$$\therefore \beta_{optimal} = \frac{1}{2} \ln \left( \frac{1 - \epsilon_{trm}}{\epsilon_{trm}} \right)$$

Problem 2-3

Show that the weight for the next iteration can be updated as $w_i^{(m+1)} = w_i^{(m)} \exp(-\beta_m y_i G_m(x_i))$

Solution)

· Since we defined the weight as following ;

$$w_n^{(m)} \equiv \exp(-y_n f_{m-1}(x_n))$$

· We can use the expression for the weight of next iteration.

$$w_n^{(m+1)} = \exp(-y_n f_m(x_n)) \quad \cdots \text{ (A)}$$

· If we can recall the function $f_m$, it should be ;

$$f_m(x_n) = f_{m-1}(x_n) + \beta_m G_m(x_n) \quad \cdots \text{ (B)}$$

· Let us substitute Ⓑ into Ⓐ, then we have ;

$$w_n^{(m+1)} = \exp\left(-y_n\left(f_{m-1}(x_n) + \beta_m G_m(x_n)\right)\right)$$

$$= \exp\left(-y_n f_{m-1}(x_n)\right)\exp\left(-y_n \beta_m G_m(x_n)\right)$$

$$= w_n^{(m)} \exp\left(-y_n \beta_m G_m(x_n)\right)$$

$$\therefore\ w_T^{(m+1)} = w_T^{(m)} \exp\left(-y_T \beta_m G_m(x_T)\right)$$