Algorithms and Uncertainty, Winter 2018/19 Lecture 24 (4 pages)

Boosting

Instructor: Thomas Kesselheim

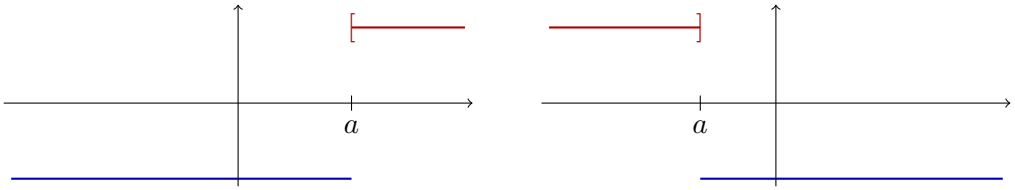
Often, you may face the situation that you have a way to classify data points in a reasonably good way but not accurately enough. For example, the reason might be that you are using hypotheses that are not expressive enough. Today, we will get to know a powerful technique that allows us to "boost" the accuracy of classification on the training set.

1 A Motivating Example

Let us start with a motivating example. While this example itself feels a little trivial, it hopefully does not take too much imagination to see that such problems also arise in more complex scenarios.

Suppose you have to classify points on the real line $\mathbb{R}$. All you start from is the set of decision stumps: These are the hypotheses of the form

$$h(x) = \begin{cases} -1 & \text{if } x < a \\ 1 & \text{if } x \geq a \end{cases} \quad \text{or} \quad h(x) = \begin{cases} -1 & \text{if } x > a \\ 1 & \text{if } x \leq a \end{cases} .$$



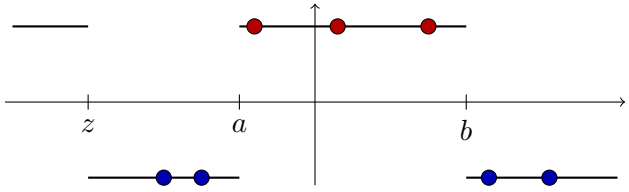
Note that we can write such a hypothesis very succinctly by two parameters $w_1 \in \mathbb{R}$, $w_2 \in \{-1, 1\}$ such that $h_{w_1,w_2}(x) = w_2 \cdot \text{sign}(x - w_1)$.[1]

Suppose your ground truth is actually that all $x \in [a, b]$ are positive and all $x \notin [a, b]$ are negative. Even in this very simple example, you cannot get a training error below $\frac{1}{3}$. For example, if $S = \{(-1, -1), (0, +1), (1, -1)\}$, one of the points will always be classified incorrectly.

However, a *linear combination of classifiers* will perform well. Given any training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, define h^* for any $z < \min_i x_i$ by

$$h^*(x) = \text{sign}\left(h_{a,1}(x) + h_{b,-1}(x) + h_{z,-1}(x)\right) .$$

For each i, this ensures that $h^*(x_i) = y_i$, so $\text{err}_S(h^*) = 0$. Note that this is still not the ground truth: Below z our classification is incorrect.



Our goal today will be exactly to extend this idea to general hypothesis classes.

[1] Redefine the sign function to be +1 at 0 for this purpose.

2 Problem Statement

Suppose we are given a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and we would like to come up with a hypothesis h such that $\mathrm{err}_S(h)$ is close to 0. All we have is a weak learning algorithm: Given any weights $p_1, \ldots, p_m$ with $\sum_i p_i = 1$ (so the weights define a probability distribution over S), it computes a hypothesis h_p such that

$$\mathrm{err}_p(h_p) := \sum_{i:h_p(x_i)\neq y_i} p_i \leq \frac{1}{2} - \gamma$$

for some fixed $\gamma > 0$.[2] Note that for $p_i = \frac{1}{m}$, this is exactly the definition of the training error. That is, the hypothesis h_p makes sure that a weighted version of the training error is small.

Example 24.1. *We come back to the example of decision stumps but the ground truth being defined by an interval $[a, b]$. We can show that using a decision stump h that minimizes $\mathrm{err}_p(h)$ fulfills the above guarantee with $\gamma = \frac{1}{6}$. That is, the empirical risk minimizer using decision stumps is a weak learner if the ground truth is defined by an interval.*

To show that $\gamma = \frac{1}{6}$, observe that for every vector p and every $a < b$

$$\sum_{i:x_i<a} p_i \leq \frac{1}{3} \quad \text{or} \quad \sum_{i:a\leq x_i\leq b} p_i \leq \frac{1}{3} \quad \text{or} \quad \sum_{i:x_i>b} p_i \leq \frac{1}{3} \ .$$

Classifying two of the the kinds of data points (below a, between a and b, above b) is easy by a decision stump. It error will be at most $\frac{1}{3} = \frac{1}{2} - \frac{1}{6}$.

We will show that this is enough to design a *strong learning algorithm*: For any $\epsilon > 0$, for any training set S, it will use the weak learner to compute hypotheses $h_1, \ldots, h_T$ and $\alpha_1, \ldots, \alpha_T$ such that h^* with $h^*(x) = \mathrm{sign}(\alpha_1 h_1(x) + \ldots + \alpha_T h_T(x))$ fulfills $\mathrm{err}_S(h^*) \leq \epsilon$.

3 Boosting via Experts Algorithms

Given a weak learning algorithm, we can come up with a strong learning algorithm using no-regret experts algorithms in a very surprising way. To this end, we define an expert for every $i \in \{1, \ldots, m\}$. That is, each expert corresponds to a data point in our training set.

It remains to define the cost vectors. We will compute hypotheses $h_1, h_2, \ldots$ as follows. In each step t, the experts algorithm uses a probability distribution $p^{(t)}$ over its experts. Input this to the weak learner and call the output h_t. Let $\ell_i^{(t)} = 1$ if $h_t(x_i) = y_i$ and 0 otherwise. That is, an expert has a cost of 1 if it was correctly classified. This sounds counter-intuitive first but makes a lot of sense: The experts algorithm moves the probability distribution towards experts of low cost. Consequently, the weak learner should classify these points better in the next round.

By the weak-learner property, we have for all t

$$\sum_{i=1}^{m} p_i^{(t)} \ell_i^{(t)} \geq \frac{1}{2} + \gamma$$

because $\mathrm{err}_{p^{(t)}}(h_t) \leq \frac{1}{2} - \gamma$.

[2] The boosting framework can be extended such that this bound only holds with a certain probability.

Furthermore, by the regret definition for all i'

$$\sum_{t=1}^{T}\sum_{i=1}^{m} p_i^{(t)}\ell_i^{(t)} \leq \sum_{t=1}^{T}\ell_{i'}^{(t)} + \text{Regret}^{(T)} \ .$$

In combination, this implies that for all i'

$$\sum_{t=1}^{T}\ell_{i'}^{(t)} \geq T\left(\frac{1}{2}+\gamma\right) - \text{Regret}^{(T)} \ .$$

That is, in $\frac{T}{2}+\gamma T - \text{Regret}^{(T)}$ steps, i' is classified correctly. If $\text{Regret}^{(T)} < \gamma T$, using the majority vote of $h_1,\dots,h_T$ is the correct classification for all of S.

Using our regret bound for Multiplicative Weights, $\text{Regret}^{(T)} \leq 2\sqrt{T\ln m}$, we would need $T > \frac{4\ln m}{\gamma^2}$.

4 AdaBoost

The algorithm *AdaBoost* (for adaptive boosting) uses exactly the ideas described in the previous section. It is an adapted version of the Multiplicative Weights algorithm. A more careful and tailored analysis gives us a much better guarantee. In particular, we will get a bound for every T, independent of m, and do not have to know γ.

The algorithm works as follows. The most striking difference to the no-regret algorithm is that there is no global learning rate η. Instead, the weight update in step t uses a factor η_t, which depends on the current error.

- Initially set $w_i^{(1)} = 1$ for all i
- In step $t = 1,\dots,T$
  - Compute $W^{(t)} = \sum_{i=1}^{m} w_i^{(t)}$, $p_i^{(t)} = w_i^{(t)}/W^{(t)}$
  - Let h_t be the outcome of the weak learner on $p^{(t)}$
  - Compute $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} p_i^{(t)}$ (the error of h_t on $p^{(t)}$)
  - Let $\eta_t = \frac{1}{2}\ln\left(\frac{1}{\epsilon_t}-1\right)$
  - Update $w_i^{(t+1)} = w_i^{(t)}\mathrm{e}^{-\eta_t y_i h_t(x_i)}$
- Return h^* defined by $h^*(x) = \text{sign}\left(\sum_{t=1}^{T}\eta_t h_t(x)\right)$

Theorem 24.2. *The AdaBoost algorithm fulfills* $\text{err}_S(h^*) \leq \exp(-2\gamma^2 T)$.

Proof. Let $g_t = \sum_{t'=1}^{t}\eta_{t'}h_{t'}(x)$. By this definition $w_i^{(t)} = \mathrm{e}^{-y_i g_{t-1}(x_i)}$ and $h^*(x) = \text{sign}(g_T(x))$.

Like in the analysis of the Multiplicative Weights algorithm, we consider the change of the sum of weights, $W^{(t)} = \sum_{i=1}^{m} w_i^{(t)}$. We will show that

$$W^{(t+1)} \leq \mathrm{e}^{-2\gamma^2}W^{(t)} \ . \tag{1}$$

This implies that $W^{(T+1)} \leq \mathrm{e}^{-2\gamma^2 T}W^{(1)} = \mathrm{e}^{-2\gamma^2 T}m$.

Furthermore, for every i with $h^*(x_i) \neq y_i$, we have to have $y_i g_T(x_i) \leq 0$ because the product of two reals of different signs is always non-positive. This means that for i also $w_i^{(T+1)} =$

$e^{-y_i g_T(x_i)} \geq 1$. For all other i, we use that $w_i^{(T+1)} \geq 0$ to get that $W^{(T+1)} \geq |\{i \mid h^*(x_i) \neq y_i\}|$ and so

$$\mathrm{err}_S(h^*) \leq \frac{1}{m} W^{(T+1)} \leq e^{-2\gamma^2 T} \ .$$

So, it remains to show Equation (1). The weight after step t is

$$W^{(t+1)} = \sum_{i=1}^{m} w_i^{(t+1)} = \sum_{i=1}^{m} w_i^{(t)} e^{-y_i \eta_t h_t(x_i)} \ .$$

So, the weight changes as

$$\frac{W^{(t+1)}}{W^{(t)}} = \sum_{i=1}^{m} \frac{w_i^{(t)}}{W^{(t)}} e^{-y_i \eta_t h_t(x_i)} = \sum_{i=1}^{m} p_i^{(t)} e^{-y_i \eta_t h_t(x_i)} = \sum_{i:h_t(x_i)=y_i} p_i^{(t)} e^{-\eta_t} + \sum_{i:h_t(x_i)\neq y_i} p_i^{(t)} e^{\eta_t} \ .$$

By definition $\sum_{i:h_t(x_i)\neq y_i} p_i^{(t)} = \epsilon_t$ and $e^{\eta_t} = \sqrt{1/\epsilon_t - 1}$, so

$$\begin{aligned} \frac{W^{(t+1)}}{W^{(t)}} &= (1-\epsilon_t) e^{-\eta_t} + \epsilon_t e^{\eta_t} = (1-\epsilon_t) \frac{1}{\sqrt{1/\epsilon_t - 1}} + \epsilon_t \sqrt{1/\epsilon_t - 1} \\ &= (1-\epsilon_t) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \ . \end{aligned}$$

By the property of a weak learner, we have $\epsilon_t \leq \frac{1}{2} - \gamma$, so

$$\frac{W^{(t+1)}}{W^{(t)}} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 2\sqrt{\left(\frac{1}{2}-\gamma\right)\left(\frac{1}{2}+\gamma\right)} = \sqrt{1-4\gamma^2} \leq \sqrt{e^{-4\gamma^2}} = e^{-2\gamma^2} \ .$$

This shows Equation (1) and completes our proof. □

5 The Downside of Boosting: Increased VC Dimension

We derived that given any training set S the training error will be at most $\exp(-2\gamma^2 T)$ when using T iterations of AdaBoost. One may be tempted to set T as high as possible because this makes the error smaller and smaller. It is important to remember that this improved accuracy will be bought by overfitting.

In more formal terms: The VC dimension of the class of hypotheses that can be produced by AdaBoost in T iterations grows in T. If we set $T \geq \frac{1}{2\gamma^2} \ln(2d)$ then $\mathrm{err}_S(h^*) = 0$ on any set S of size at most d. This also means that a set of size d is shattered.

Consequently, one needs to be cautious when applying boosting: It is a reasonable tool derive better classification but there is a usual trade-off between training errors and overfitting.

References

Freund, Yoav; Schapire, Robert E (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 55: 119. (Original AdaBoost paper)