Lecture 7: Boosting

CSE 427: Machine Learning

Md Zahidul Hasan

Lecturer, Computer Science
BRAC University

Spring 2023

Weak Learners

We have already seen that there are hypothesis classes that can be learned very accurately with high probability but yet they are computationally infeasible. But what if we could learn a hypothesis class that performs weakly and somehow aggregate these weak learners to get arbitrarily good performance? In fact, we could also directly calibrate the bias-complexity trade-off. A weak binary classifier is one that performs slightly better than a coin toss.

Weak Learners

A learning algorithm $\mathcal{A}$ is said to be a $\lambda-$weak learner with $0 \leq \lambda \leq \frac{1}{2}$ for a hypothesis class $\mathcal{H}$ if there exists a sample size $m \geq poly(\frac{1}{\delta})$ such that for any $\delta \in (0,1)$, for any distribution $\mathcal{D}$ over $\mathcal{X}$, for any labeling function $f : \mathcal{X} \to \{1,-1\}$, if the realizability assumption holds, then when the algorithm is run with a sample size of at least m, then the following holds:

$$Pr[R_{\mathcal{D}}(h) \leq \frac{1}{2} - \lambda] \geq 1 - \delta.$$

Learning Intervals Using Thresholds

An interval function is defined as follows where $b \in \{-1, +1\}$ and $\theta_1, \theta_2 \in \mathbb{R}$:

$$h_{\theta_1,\theta_2,b}(x) = \begin{cases} -b & \textit{if } \theta_1 \leq x \leq \theta_2 \\ +b & \textit{otherwise} \end{cases}$$

We will weakly learn an interval function using the class of threshold functions (i.e. $f_{\theta,b}(x) = \textit{sign}(\theta - x).b$ where $b \in \{+1, -1\}$). Notice that an interval divides the real line into 3 pieces. Using a threshold function, we can always match the sign of any two of the pieces. One of the three pieces will have at most $\frac{1}{3}$ probability of containing a point. Except this one, let's match the signs of the other pieces. In that case, our error is at most $\frac{1}{3} = \frac{1}{2} - \lambda$ with $\lambda = \frac{1}{6}$.

Efficient Implementation of ERM for Learning Intervals

Solving this problem for multidimensional points can be done independently for each dimension. So, we solve this problem for the one-dimensional case only. Suppose, we are given a sample of m points $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$ and a distribution vector D so that D_i represents the importance of (x_i, y_i) in predicting the generalization error $R_{\mathcal{D}}(h)$. So,

$$R_{\mathcal{D}}(h) = \sum_{i=1}^{m} D_i 1_{h(x_i) \neq y_i}$$

We want to find a $\theta \in \mathbb{R}$ so that $h_{\theta,1}(x) = sign(\theta - x)$ can minimize this error. The case for $h_{\theta,-1}$ can be solved analogously. Notice that, first we have to sort the points based on their x_i. We will have to find a $\theta \in \mathbb{R}$. Our search space is huge. But we can reduce it down to $\Theta = \{\theta_0, \theta_1, \cdots, \theta_m\}$ where for $1 \leq i < m$, $\theta_i = \frac{x_i + x_{i+1}}{2}$, $\theta_0 = -\infty$ and $\theta_m = +\infty$.

Efficient Implementation of ERM for Learning Intervals

We can rewrite the generalization in terms of the choice of θ as follows:

$$R_{\mathcal{D},\theta}(h) = \sum_{i:y_i=1} D_i 1_{\theta<x_i} + \sum_{i:y_i=-1} D_i 1_{\theta>x_i}$$

From this we can derive that,
$R_{\mathcal{D},\theta_{i+1}} = R_{\mathcal{D},\theta_i} - D_i 1_{y_i=1} + D_i 1_{y_i=-1} = R_{\mathcal{D},\theta_i} - D_i y_i$.
By using this formula, we can compute the generalization error if we chose $m+1$ different θ.
Complexity Analysis: The sorting takes $\mathcal{O}(m \log m)$ calculations. Then computing the generalization errors take $\mathcal{O}(m)$ operations. So, the overall computational complexity is $\mathcal{O}(m + m \log m)$.

AdaBoost

AdaBoost

input:

training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$

weak learner WL

number of rounds T

initialize $\mathbf{D}^{(1)} = (\frac{1}{m}, \ldots, \frac{1}{m})$.

for $t = 1, \ldots, T$:

invoke weak learner $h_t = \mathrm{WL}(\mathbf{D}^{(t)}, S)$

compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$

let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$

update $D_i^{(t+1)} = \dfrac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$ for all $i = 1, \ldots, m$

output the hypothesis $h_s(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\right)$.

Adaboost Explained

- Adaboost has access to a weak learner that can return a weak hypothesis with error $\frac{1}{2} - \lambda$ for $\lambda \in (0, \frac{1}{2}]$.
- Initially AdaBoost assumes that all of the samples are equally important during learning. So, it initializes the weights as $D^{(1)} = (\frac{1}{m}, \cdots, \frac{1}{m})$.
- In each iteration i, AdaBoost invokes the weak learner with the distribution $D^{(i)}$. Then it computes the estimated generalization error for the returned hypothesis.
- The returned hypothesis is assigned a weight that is inversely proportionally to the error it makes.
- AdaBoost then updates the distribution. A data sample gets high probability if it had high probability in the previous round and also if the returned hypothesis makes a mistake on this data instance. It's identified as a problematic data instance.
- The final prediction is a weighted aggregate of the individual predictions of all the returned hypotheses.

Empirical Risk Bound for AdaBoost

Theorem

Let S be a training set and assume that AdaBoost returns a weak hypothesis at each iteration with $\epsilon_t \leq \frac{1}{2} - \lambda$. Then the training error of the output hypothesis of AdaBoost is bounded as follows:

$$R_S(h_{AB}) = \frac{1}{m}\sum_{i=1}^{m} 1_{h_{AB}(x_i)\neq y_i} \leq e^{-2\lambda^2 T}$$

Proof: For each iteration t, let $f_t = \sum_{i=1}^{t} w_i h_i$ and for each f_t, let's define $Z_t = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f_t(x_i)} \geq \frac{1}{m}\sum_{i=1}^{m} 1_{f_t(x_i)\neq y_i} = R_S(f_t)$. We just need to prove that, $R_S(h_{AB}) = R_S(f_T) \leq Z_T \leq e^{-2\lambda^2 T}$. In fact, we can prove that, $\frac{Z_t}{Z_{t-1}} \leq e^{-2\lambda^2}$ for all t. Using induction, we can see that,

$$D_i^{(t+1)} = \frac{D_i^{(t)} e^{-w_t y_i h_t(x_i)}}{\sum_{j=1}^{m} D_j^{(t)} e^{-w_t y_j h_t(x_j)}} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_t(x_j)}}$$

Empirical Risk for AdaBoost Upper Bound

So,

$$\frac{Z_t}{Z_{t-1}} = \frac{\frac{1}{m}\sum_{i=1}^{m} e^{-y_i f_t(x_i)}}{\frac{1}{m}\sum_{i=1}^{m} e^{-y_i f_{t-1}(x_i)}}$$

$$= \frac{\sum_{i=1}^{m} e^{-y_i f_t(x_i)}}{\sum_{i=1}^{m} e^{-y_i f_{t-1}(x_i)}} = \frac{\sum_{i=1}^{m} e^{-y_i (f_{t-1}(x_i)+w_t h_t(x_i))}}{\sum_{i=1}^{m} e^{-y_i f_{t-1}(x_i)}}$$

$$= \frac{\sum_{i=1}^{m} e^{-y_i f_{t-1}(x_i)-y_i w_t h_t(x_i)}}{\sum_{i=1}^{m} e^{-y_i f_{t-1}(x_i)}} = \sum_{i=1}^{m} \frac{e^{-y_i f_{t-1}(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_{t-1}(x_j)}} e^{-y_i w_t h_t(x_i)}$$

$$= \sum_{i=1}^{m} D_i^{(t)} e^{-y_i w_t h_t(x_i)}$$

$$= \sum_{i:y_i h_t(x_i)=1} D_i^{(t)} e^{-w_t} + \sum_{i:y_i h_t(x_i)=-1} D_i^{(t)} e^{w_t}$$

$$= (1-\epsilon_t) e^{-w_t} + \epsilon_t e^{w_t}$$

$$= \frac{1-\epsilon_t}{\sqrt{\frac{1}{\epsilon_t}-1}} + \epsilon_t \sqrt{\frac{1}{\epsilon_t}-1}$$

$$= 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Upper Bound Proof

We want to maximize a function of the form $f(a) = a(1-a)$. But this function is strictly increasing in the interval $[0, \frac{1}{2}]$. Since, $\epsilon_t \leq \frac{1}{2} - \lambda$,

$$2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 2\sqrt{(\tfrac{1}{2}-\lambda)(\tfrac{1}{2}+\lambda)} = \sqrt{1-4\lambda^2}$$

Since, $1 + x \leq e^x$ for all $x \in \mathbb{R}$, $1 - 4\lambda^2 \leq e^{-4\lambda^2}$. So, we have $\sqrt{1-4\lambda^2} \leq e^{-2\lambda^2}$.
So, we have proved that, $\frac{Z_t}{Z_{t-1}} \leq e^{-2\lambda^2}$.
So, $\frac{Z_T}{Z_0} = \frac{Z_T}{Z_{T-1}} \frac{Z_{T-1}}{Z_{T-2}} \dots \frac{Z_1}{Z_0} \leq e^{-2T\lambda^2}$.
Since $f_0 \equiv 0$, $Z_0 = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i 0} = 1$. So,
$R_S(h_{AB}) \leq Z_T \leq e^{-2T\lambda^2}$.
From this, we can say that, $\lim_{T\to\infty} R_S(h_{AB}) = 0$. So, having too many iterations can lead to overfitting.

Managing the Bias-Complexity Trade-Off using T

In the previous example, our weak hypothesis class was the following: $\mathcal{H}_{DS} = \{h : h(x) = sign(x - \theta).b, \theta \in \mathbb{R}, b \in \{-1, 1\}\}$
And the strong hypothesis class is the following: $L(\mathcal{H}_{DS}, T) = \{f : f(x) = sign(\sum_{i=1}^{T} w_i h_i(x)), w \in R^T, \forall i, h_i \in \mathcal{H}_{DS}\}$
It can be proven that $L(\mathcal{H}_{DS}, T)$ includes all piece-wise constant classifiers with at most T pieces. Therefore,

$$VCdim(L(\mathcal{H}_{DS}, T)) \geq T + 1.$$

So, as T increases, so does the VC dimension of our hypothesis class. If the VC dimension increases, then the complexity of the hypothesis increases. As the complexity increases, the approximation error of the hypothesis is reduced. But the estimation error increases. So, it gets more difficult to find best the hypothesis in the class. Hence, the number of iterations T directly influences the bias vs complexity trade-off. So, by tuning T, we attain the optimum performance of boosting.