CS6781 - Theoretical Foundations of Machine Learning

Lecture 17: Intro to Boosting

April 14^{th}, 2020

Lecturer: Nika Haghtalab *Readings: None*
Scribe: Rishi Bommasani and William Gao

1 Strong and Weak Learners

The main focus of our learning algorithms in the PAC model have been to achieve *arbitrarily small* generalization error, $\mathrm{err}_{\mathcal{D}}(h)$. In today's lecture, we consider relaxing this strong requirement and instead learning a hypothesis that performs slightly better than random guessing. We explore whether there is an inherent difference between having algorithms that always guarantee arbitrarily small generalization error and those who are only capable of guaranteeing much larger error rates.

Let us recall learnability in the PAC model, hereafter called *strong learnability*.

Definition 1.1 (Strong Learnability)**.** A hypothesis class $\mathcal{C}$ is *strongly learnable*, if there is an algorithm that for any realizable distribution $\mathcal{D}$, any $\epsilon > 0$, and any $\delta > 0$, uses $\mathrm{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, \mathrm{VCDim}(\mathcal{H}))$ samples from $\mathcal{D}$ and returns a classifier h such that

$$\Pr\left(\mathrm{err}_{\mathcal{D}}(h) \leq \varepsilon\right) \geq 1 - \delta. \tag{1}$$

The algorithm performing this learning task is call a *strong learner*.

Definition 1.2 (Weak Learnability)**.** A hypothesis class $\mathcal{C}$ is γ-*weakly learnable* for some $\gamma > 0$, if there is an algorithm that for any realizable distribution $\mathcal{D}$ and any $\delta > 0$, uses $\mathrm{poly}(\frac{1}{\delta}, \mathrm{VCDim}(\mathcal{H}))$ samples from $\mathcal{D}$ and returns a classifier h such that

$$\Pr\left(\mathrm{err}_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma\right) \geq 1 - \delta. \tag{2}$$

The algorithm performing this learning task is call a γ-*weak learner*.

Note that if a learner ignores the samples and guesses $+1, -1$ uniformly at random, it has an error of exactly $1/2$. A weak learner is one that can essentially beats this random prediction by a small $\gamma > 0$.

A natural question to ask is whether strong learnability and weak learnability are inherently different. That is, are there classes that are weakly learnable but not strongly? Freund [1995] and Schapire [1990] showed that weak learnability is equivalent to strong learnability, i.e., there is a *boosting algorithm* that uses a weak learner on an adaptively designed short sequence of distributions and gives a strong learner.

Theorem 1.3. *A concept class $\mathcal{C}$ is weakly PAC-learnable iff it is strongly PAC-learnable.*

To see how applying a weak learner to an adaptively designed short sequence of distributions can result in a strong learner, we start with a warm up that applies to a specific class of weak-learners. We then see how the same ideas can be generalized to work for any weak learner.

2 Warm up

In this section, we see how a special class of weak learners can be boosted to strong learners. These weak learners are special, because in addition to predicting $+1$ or -1, they can also tell us that they they don't know the answer. To make sure that these learners do "learn", there must be a small fraction of the data where they can make better than random predictions.

Formally consider a classifier $g(x) \in \{+1, -1, \text{not sure}\}$. For any $\mathcal{D}$ some $\epsilon < 1/2$ and some $\epsilon'' > 0$, we say that $g(\cdot)$ can $(1-\epsilon', \epsilon)$-learn if

1. On at most $1 - \epsilon'$ fraction of data, g says *not sure*.
2. Conditioned on making a prediction, g has error of at most ϵ.

Let us first show that $g(\cdot)$ that $(1-\epsilon', \epsilon)$-learns is a special type of weak learner. Let $h(x) \in \{-1, +1\}$ be a classifier such that on input x randomly outputs $+1$ or -1 with equal probability if $g(x) =$ not sure. Otherwise, $h(x) = g(x)$ which is either $+1$ or -1. Note that

$$\text{err}_{\mathcal{D}}(h) \leq \frac{1}{2}(1-\epsilon') + \epsilon'\epsilon \leq \frac{1}{2} - \gamma,$$

when $\gamma = \epsilon'(0.5 - \epsilon) > 0$. So h is a weak learner.

What we will first show is that if we have a $(1-\epsilon', \epsilon)$-weak learner of the above type, we can yield a strong learner with error $\varepsilon = 2\epsilon$. In particular, we can construct weak learners $g_1, \ldots, g_T$ as given in Algorithm 1.

Algorithm 1: Decision List Boosting Algorithm

Initialize $P_1 \leftarrow \mathcal{U}\left(\mathcal{S} \triangleq \{(x_1, y_1), \ldots, (x_m, y_m)\}\right)$
for $t \leftarrow 1, \ldots, T$ **do**
  g_t is a $(1-\epsilon', \epsilon)$-weak learner on P_t
  $P_{t+1} \leftarrow P_t \mid \{x : g_t(x) = \textit{not sure}\}$
end

We can then define our strong learner by outputting the decision of g_1 if g_1 makes a prediction and backing off to g_2 if it doesn't and repeating this iteratively as needed. If all of $g_1, \ldots, g_T$ indicate that they are *not sure*, we can then make a random prediction. Setting $T = \frac{1}{\epsilon'} \ln(\frac{1}{2\epsilon})$ ensures the error associated with random guesses is at most $\frac{1}{2}(1-\epsilon')^T \leq \epsilon$. On the other hand, conditioned on any of the weak learners making a prediction, the error (by the definition of the weak learner) associated with that is at most ϵ. The resulting strong learner therefore achieves error $\varepsilon = 2\epsilon$.

3 A General Boosting Algorithm

In the previous section, we made explicit use of the fact that the weak learners considered are able to identify when they are uncertain. For traditional learners whose hypotheses have the simple label

space of $\{-1, 1\}$, there is no immediate analogue. This results in two issues with the approach given in section 2 and we introduce the workarounds that we will consider in this lecture.

1. **Training.**
   Issue: We cannot train h_{t+1} on the distribution $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \mid \{h_t(x) = \textit{not sure}\}$.
   Workaround: Train h_{t+1} on $\mathcal{D}_{t+1}$ which is $\mathcal{D}_t$ with the examples that h_t gets wrong up-weighted.

2. **Prediction.**
   Issue: We cannot apply h_{t+1} conditional on $h_1, \ldots, h_t$ being *not sure*.
   Workaround: Take the weighted majority vote of $h_1, \ldots, h_T$ with weights related to the quality of each of the h_i.

These workarounds give rise to the template given in Algorithm 2.

Algorithm 2: Template for Boosting Algorithms for traditional weak learners

Initialize $P_1 \leftarrow \mathcal{U}\left(\mathcal{S} \triangleq \{(x_1, y_1), \ldots, (x_m, y_m)\}\right)$
for $t \leftarrow 1, \ldots, T$ **do**
  h_t is the hypothesis generated by a γ_t-weak learner on P_t
  P_{t+1} is defined in terms of P_t with higher weights on $\{x_i : h_t(x_i) \neq y_i\}$
end
Return: $h_{\text{final}}(x) = \text{sign}\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right)$

3.1 AdaBoost

We next introduce the celebrated `AdaBoost` algorithm [Freund and Schapire, 1995], which is a specific instantiation of the template we previously introduced. In particular, the weighting schema for defining P_{t+1} and the voting weights α_i will be tightly connected and set adaptively.

Theorem 3.1. *Let h_{final} be the output of AdaBoost. Then* $\text{err}_{\mathcal{S}}(h_{\text{final}}) \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)$.

Corollary 3.2. *Under the assumption that $\forall t, \gamma_t \geq \gamma$, setting $T = \mathcal{O}\left(\frac{1}{\gamma^2} \ln\left(\frac{1}{\varepsilon}\right)\right)$ implies that* `AdaBoost` *generates a hypothesis h_f that achieves error at most ε. That is,* $\text{err}_{\mathcal{S}}(h_{\text{final}}) \leq \varepsilon$.

In ADABoost, distribution P_{t+1} is intuitively designed so that weak learning on distribution P_{t+1} would result in new information about how we should label each points. At the very least, we want hypothesis h_t not to be a good learner for P_{t+1}, i.e., the algorithm has to output a new $h_{t+1} \neq h_t$ in the next round. So, P_{t+1} is designed so that $\text{err}_{P_{t+1}}(h_t) = \frac{1}{2}$. Intuitively, this is the *hardest* new distributions for the algorithm, in the way that h_t is doing just as bad/good as random guessing. Next, we formally prove that $\text{err}_{P_{t+1}}(h_t)$.

Algorithm 3: `AdaBoost` Algorithm [Freund and Schapire, 1995]

Initialize $P_1 \leftarrow \mathcal{U}\left(\mathcal{S} \triangleq \{(x_1, y_1), \ldots, (x_m, y_m)\}\right)$

for $t \leftarrow 1, \ldots, T$ **do**

  h_t is the hypothesis generated by the weak learner on P_t with error $\varepsilon_t \triangleq \mathrm{err}_{P_t}(h_t)$

  $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

  $P_{t+1}(x_i) \leftarrow \dfrac{P_t(x_i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

  where Z_t is the natural normalizing factor $\sum_{j\in[n]} P_t(x_j)\exp(-\alpha_t y_j h_t(x_j))$

end

Return: $h_{\text{final}}(x) = \mathsf{sign}\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right)$

Lemma 3.3. $\forall t \in [T-1], \mathrm{err}_{P_{t+1}}(h_t) = \frac{1}{2}$

Proof. We will demonstrate this by showing that the sets $\{x_i : h_t(x_i) = y_i\}$ and $\{x_i : h_t(x_i) \neq y_i\}$ have equal probability mass under P_{t+1}.

$$\Pr_{P_{t+1}}\left(\{x_i : h_t(x_i) = y_i\}\right) = \frac{1-\varepsilon_t}{Z_t}\exp\left(\left(\frac{-1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)\right)\right) = \frac{\sqrt{\varepsilon_t(1-\varepsilon_t)}}{Z_t} \tag{3}$$

$$\Pr_{P_{t+1}}\left(\{x_i : h_t(x_i) \neq y_i\}\right) = \frac{\varepsilon_t}{Z_t}\exp\left(\left(\frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)\right)\right) = \frac{\sqrt{\varepsilon_t(1-\varepsilon_t)}}{Z_t} \tag{4}$$

as required. □

Corollary 3.4. $Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$

Proof.

$$\Pr_{P_{t+1}}\left(\{x_i : h_t(x_i) = y_i\}\right) + \Pr_{P_{t+1}}\left(\{x_i : h_t(x_i) \neq y_i\}\right) = 1 \implies Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \tag{5}$$

□

At a high level, what we have to show is that if h_{final} misclassifies a point x, i.e., the weighted majority of functions $h_1, \ldots, h_T$ similarly misclassifies a point, then the weight of x in P_T is very large. In the next lecture we will formally show that this weight would be at least $\frac{1}{m}\frac{1}{\prod_{t=1}^{T} Z_t}$. This means that the total empirical error of h_{final} is at most $\prod_{t=1}^{T} Z_t$. This is because the sum of all the x_i probabilities is 1, the sum of the probabilities for the points which h_{final} misclassifies is at most 1. Consequently, there are at most $m\prod_{t=1}^{T} Z_t$ misclassified points, which further implies the error on $\mathcal{S}$ is at most $\prod_{t=1}^{T} Z_t$ as desired.

What would remain then is to show that $\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \to 0$. Intuitively, the latter calculation hold because

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t 2\sqrt{(\frac{1}{2}-\gamma_t)(\frac{1}{2}+\gamma_t)} = \prod_t \sqrt{(1-2\gamma_t)(1+2\gamma_t)}$$

is the geometric mean of the values that are bounded away from 1. This gap between the values and 1 ensures that the geometric mean is smaller than the arithmetic one and therefore it shrinks quickly as $t \to \infty$. We will see a formal proof of these arguments and Theorem 3.1 in the next lecture.

References

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121 (2):256–285, 1995.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.