10-716: Advanced Machine Learning **Spring 2019**

Lecture 16: March 26

Lecturer: Pradeep Ravikumar *Scribes: Xuan Li, Yujie Wei*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

16.1 Computation complexity

16.1.1 Motivations

In the previous lecture, we finished our discussion on lower bounds. Starting from this lecture, we will dive into a new topic, **computational complexity**, which is adopted to measure how the bound of an excess error is related to the available computational resources. The motivations of taking into account the computational complexity of learning algorithms originate from following interesting observations:

- With a large amount of training data and limiting computational resources, a learning algorithm might not able to achieve the desired excess error within a reasonable amount of time. Therefore, it is necessary to consider the tradeoff between the amount of data and the amount of computational resources when designing a learning algorithm.
- A learning problem is often set up as finding proper estimators for a set of natural parameters. Since an estimator can be viewed as a function of sampled data, we are interested in whether the complexity of a function will affect the performance of an estimator. In other words, if we allow a particular estimator to have more operations than others, can it obtain a lower excess error?
- Human brains are capable of learning efficiently from observations, suggesting that there are learning algorithms whose computational complexity is near linear w.r.t the amount of data. Can we find such learning algorithms?

With these motivating questions in mind, we will start from setting up a theoretical framework that takes into account the computational complexity of a learning algorithm.

16.1.2 Excess error decomposition

Recall that in the first half of the course we developed an Empirical Risk Minimization(ERM) framework which bounds excess error $\varepsilon(\theta_n, \theta^*)$ by decomposing it into several terms as shown below:

$$\begin{aligned}\varepsilon(\theta_n, \theta^*) &= R(\theta_n) - R(\theta^\star) &(16.1)\\ &= \underbrace{R(\theta_n) - R(\theta_0^*)}_{\text{estimation error}} + \underbrace{R(\theta_0^*) - R(\theta^\star)}_{\text{approximation error}} &(16.2)\\ &= \varepsilon_{\text{est}} + \varepsilon_{\text{app}} &(16.3)\end{aligned}$$

where θ_n is the best estimator $\theta \in \Theta_0$ that minimizes the empirical risk and θ^* is the optimal parameter that minimizes the actual risk. The error ε_{app} measures how closely estimators $\theta_0 \in \Theta_0$ can approximate the optimal parameter θ^* while the error ε_{est} measures the effect of measuring empirical risk on sampled data rather than the actual risk.

Notice that the excess error is determined by the number of training samples and the capacity of the selected subset of parameters Θ_0 (or the capacity of the family of functions). One of the key findings of our previous discussions is that we can bound the estimation error by using Rademacher Complexity to measure the complexity of a family of functions. Larger family of functions can have smaller approximation errors ε_{app} but lead to larger estimation errors ε_{est}.

In practice, θ_n is often obtained by using some optimization algorithms which can minimize $R(\theta_n)$ up to a particular predefined tolerance ρ. We define such an approximation solution as $\tilde{\theta}_n$ and know that:

$$R(\tilde{\theta}_n) = R(\theta_n) + \rho \tag{16.4}$$

Now we instead care about the excess error between $\tilde{\theta}_n$ and θ^* which contains three terms:

$$\begin{aligned}
\varepsilon(\tilde{\theta}_n, \theta^*) &= R(\tilde{\theta}_n) - R(\theta^\star) && (16.5)\\
&= \underbrace{R(\tilde{\theta}_n) - R(\theta_n)}_{\text{optimization error}} + \underbrace{R(\theta_n) - R(\theta_0^*)}_{\text{estimation error}} + \underbrace{R(\theta_0^*) - R(\theta^\star)}_{\text{approximation error}} && (16.6)\\
&= \varepsilon_{\text{opt}} + \varepsilon_{\text{est}} + \varepsilon_{\text{app}} && (16.7)
\end{aligned}$$

where the additional term ε_{opt} is determined by the predefined tolerance ρ.

Recall that both the estimation error and approximation error can be bounded by Rademacher complexity as discussed in previous lectures, we have:

$$\begin{aligned}
\varepsilon(\tilde{\theta}_n, \theta^*) &= \varepsilon_{\text{opt}} + \varepsilon_{\text{est}} + \varepsilon_{\text{app}}\\
&\le (2r_n + \rho) + 2r_n + \varepsilon_{\text{app}}\\
&= 4r_n + \rho + \varepsilon_{\text{app}}
\end{aligned}$$

16.1.3 Problem setup

The decomposition of the excess error leads to a complicated compromise which can be formalized as an optimization problem as shown below:

$$\min_{d,\rho,n} \varepsilon(\tilde{\theta}_n, \theta^*) = \varepsilon_{\text{opt}} + \varepsilon_{\text{est}} + \varepsilon_{\text{app}} \tag{16.8}$$

$$\text{s.t.} \begin{cases} n \le n_{max} \\ T \le T_{max} \end{cases} \tag{16.9}$$

The optimization problem has three variables: the capacity of a parameter space $d = |\Theta_0|$ which is usually determined by the model we select, the number of samples n which can be altered by just using a subset of all available n_{max} samples, and the desired tolerance ρ within the alloted training time T_{max}. The excess error changes as any one of the variables changes. Their relationship is shown below:

Table 16.1: Typical variations when d, n, ρ increase

	d	n	ρ
ε_{app}	$\downarrow$		
ε_{est}	$\uparrow$	$\downarrow$	
ε_{opt}			$\uparrow$
T	$\uparrow$	$\uparrow$	$\downarrow$

The optimization problem defined above contains two following cases:

- Small-scale learning problems: For small-scale learning problems where computing time is not limited, the maximum number of samples n_{max} is the constraint. In this case, we can select a arbitrary small rho so that the optimization error ε_{opt} becomes trivial and the excess error is dominated by ε_{app} and ε_{est}. Simply take $n = n_{max}$ and we can simplify the problem so as to just consider the approximation-estimation tradeoff only without taking into account the computational complexity.
- Large-scale learning problems: For large-scale learning problems which are constrained by the maximum computing time T_{max}. We need to consider the approximation-estimation-optimization tradeoff. In practice people tend to release the tolerance ρ so as to use more data within the alloted time. In other words, we tend to solve a large-scale problem *approximately* rather than solving a small-scale problem *precisely*.

16.1.4 A gradient descent example

Now let's examine the approximation-estimation-optimization tradeoff under a vanilla gradient descent setting. A gradient descent algorithm can be viewed as solving iterative quadratic optimization problems where each iteration takes a constant time. Therefore, analyzing the number of iterations is equivalent to analyzing the computation time.

Example: Choosing a proper $T(n, d, \rho)$ for gradient descent.

Suppose the objective function is convex and differentiable. Also we suppose the objective function has a condition number $\kappa = \lambda_{max}/\lambda_{min}$ where λ is the eigen spectrum of the Hessian matrix. From the convergence analysis of gradient descent, after running gradient descent m iterations, we have:

$$\rho = \theta(m) - \tilde{\theta}_n \leq \exp(-m/k) \tag{16.10}$$

Knowing that the constant time needed for each iteration in gradient descent is $O(nd)$, the number of iterations needed to reach tolerance ρ is $O(\kappa \log(\frac{1}{\rho}))$. Therefore, the time needed to reach tolerance ρ is $O(nd\kappa \log(\frac{1}{\rho}))$. In other words, the computational complexity constraint for gradient descent becomes:

$$nd\kappa \log(1/\rho) \leq T_{max}$$

16.2 Stability

Notice that in the previous section we decompose the excess error into three terms ε_{app}, ε_{est}, and ε_{opt}. However, in large-scale learning algorithms, the estimation error is very large (i.e, deep neural networks, $O(D^2/N)$), which make it unable to generalize. In other words, decomposing the excess error into ε_{app} and ε_{est} does not lead to a tight bound. Thus, instead we should revisit the excess error and look into the statistical performance of $\tilde{\theta}_n$. If we decompose the overall error into optimization error and estimation error such that

$$R(\tilde{\theta}_n) - R(\theta_0^\star) = \underbrace{R(\tilde{\theta}_n) - R(\hat{\theta}_n)}_{\text{optimization error}} + \underbrace{R(\hat{\theta}_n) - R(\theta_0^\star)}_{\text{estimation error}} \tag{16.11}$$

The question is can we analyze the statistical performance with regularization provided with the algorithm. To analyze, we further decompose the expression as

$$\begin{aligned}
&\mathbb{E}\big[R(\tilde{\theta}_n) - R(\theta_0^\star)\big] && && (16.12)\\
&= \mathbb{E}\big[R(\tilde{\theta}_n) - R_n(\tilde{\theta}_n)\big] && text(estimationerror) && (16.13)\\
&+ \mathbb{E}\big[R_n(\tilde{\theta}_n) - R_n(\hat{\theta}_n)\big] && \text{(optimization error)} && (16.14)\\
&+ \mathbb{E}\big[R_n(\hat{\theta}_n) - R_n(\theta_0^\star)\big) && (\leq 0) && (16.15)\\
&+ \mathbb{E}\big[R_n(\theta_0^\star) - R(\theta_0^\star)\big] && (= 0) && (16.16)\\
& && && (16.17)
\end{aligned}$$

So the final question is can we bound the estimation error term $\mathbb{E}\big[R(\tilde{\theta}_n) - R_n(\tilde{\theta}_n)\big]$?

Definition 16.1 *an algorithm is $\epsilon-stable$ if for all S*

$$\sup_z |\ell\big(\tilde{\theta}(S), Z\big) - \ell\big(\tilde{\theta}(S'), Z\big)| \leq \epsilon$$

where

$$S : \{\{Z_i\}_1^n,\ Z_i \overset{iid}{\sim} \mathbb{P}\}$$

$$S' : \{\{Z_j\}_{j\neq i} \cup \{Z_i'\},\ Z_i' \overset{iid}{\sim} \mathbb{P}\}$$

in other words, S differs from S' in the ith sample only.

Theorem 16.2 *If an algorithm is $\epsilon-stable$, then*

$$\epsilon_{est}^A \leq \epsilon$$

Proof: what we want to prove is

$$\epsilon_{est}^A = \mathbb{E}_S\big[R_n\big(\tilde{\theta}(S)\big) - R\big(\tilde{\theta}(S)\big)\big] \leq \epsilon$$

Since

$$\begin{aligned}
\mathbb{E}_S\ \big[R_n\big(\tilde{\theta}(S)\big)\big] &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{S,Z_i}\big[\ell\big(\tilde{\theta}(S), Z_i\big)\big]\\
&= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{S^i,Z_i'}\big[\ell\big(\tilde{\theta}(S^i), Z_i'\big)\big]
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}_S\ \big[R\big(\tilde{\theta}(S)\big)\big] &= \mathbb{E}_{S,Z}\big[\ell\big(\tilde{\theta}(S), Z\big)\big]\\
&= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{S,Z_i'}\big[\ell\big(\tilde{\theta}(S), Z_i'\big)\big]
\end{aligned}$$

We can derive

$$\begin{aligned}
&\mathbb{E}_S\ \left[R_n\big(\tilde{\theta}(S)\big)\right] - \mathbb{E}_S\ \left[R\big(\tilde{\theta}(S)\big)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{S^i,Z_i'}\left[\ell\big(\tilde{\theta}(S^i), Z_i'\big)\right] - \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{S,Z_i'}\left[\ell\big(\tilde{\theta}(S), Z_i'\big)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{S,S^i,Z_i'}\left[\ell\big(\tilde{\theta}(S^i), Z_i'\big) - \ell\big(\tilde{\theta}(S), Z_i'\big)\right] \\
&\le \epsilon
\end{aligned}$$

■

Example: stability of gradient descent
Under the assumptions:

- $\ell(\theta, Z)$ is convex
- $||\nabla\ell(\theta, Z) - \nabla\ell(\theta', Z)|| \le \beta||\theta - \theta'||$
- $\ell(\theta, Z) - \ell(\theta', Z) \le L||\theta - \theta'||$

Theorem 16.3 *Under the above assumptions, with step size $\eta \le \frac{1}{\beta}$,*

$$\epsilon_{STAB} \le 2\eta\ \frac{L^2 T}{n}$$