ECE 901: Large-scale Machine Learning and Optimization **Fall 2016**

Lecture 2 — 09/08

Lecturer: Dimitris Papailiopoulos *Scribe: Yifei Liu & Fan Gao*

2.1 Empirical Risk Minimization (ERM)

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y)\}$, where $(\boldsymbol{x}_i, y_i)$'s are independently and identically distributed (i.i.d.) following some underlying distribution. Then ERM aims to find a model from certain hypothesis class $\mathcal{H}$ that minimizes the empirical risk,

$$\min_{\text{models}\in\mathcal{H}} \sum_i \text{disagree}(\text{model}(\boldsymbol{x}_i), y_i)$$

When we apply machine learning methods on a dataset, usually we will split the data into three folds with the following proportion, (training set : validation set: test set) = (2:1:1). ERM is related to the first two folds. We can use cross validation or holdout set to perform parameter tuning, see Chapter 7.7 (page 222) of *Introduction to Statistical Learning Theory* for more detail.

We care about two questions on ERM:

- When does the ERM concentrate around the true risk?
- How does the hypothesis class affect the ERM?

We will try to answer these questions in the following sections.

For a machine learning problem, our goal is to find a hypothesis h_s with the smallest expected risk, which is defined as

$$R[h_s] = \mathbb{E}\{f(h_s(\boldsymbol{x}), y)\}, \tag{2.1}$$

where $f(\cdot, \cdot)$ is a function that measures the disagreement between predicted label and true label, and the expectation is taken with respect to the underlying distribution. However, in practice, we don't know the exact form of the underlying distribution. Therefore, we use the empirical risk instead and try to find the minimizer of it. Define the empirical risk of a hypothesis h to be

$$\hat{R}_n[h] = \frac{1}{n}\sum_{i=1}^{n} f(h(\boldsymbol{x}_i), y_i), \tag{2.2}$$

and define the generalization error to be

$$\epsilon_{\text{gen}}[h] = |\hat{R}_n[h] - R[h]|. \tag{2.3}$$

To make the exposition simpler, we assume that our loss function f is between 0 and 1, namely, $0 \leq f(a,b) \leq 1, \forall a, b$. To answer the first question, we need the following powerful Hoeffding's Inequality.

Theorem 2.1 (Hoeffding's Inequality). *Let $X_1, X_2, \cdots, X_n$ be i.i.d. random variables on $\mathbb{R}$, and $\forall i, 0 \leq X_i \leq 1$. Let $S_n = \frac{1}{n}\sum_i X_i$, then*

$$Pr(|S_n - \mathbb{E}[S_n]| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \tag{2.4}$$

Exercise: How many samples do we need to bound $Pr(|S_n - \mathbb{E}[S_n]| \geq \epsilon) \leq 0.1$? $n = \frac{3}{2}\epsilon^{-2}$.

Denote $R_i = f(h(\boldsymbol{x}_i), y_i)$, then $\hat{R}_n = \frac{1}{n}\sum_{i=1}^n R_i$ and

$$\mathbb{E}(R_i) = \mathbb{E}(\hat{R}_n) = \mathbb{E}\{f(h(\boldsymbol{x}), y)\} = R[h] \tag{2.5}$$

Corollary 2.2. *Using Theorem 2.1 and that the loss function is between 0 and 1, we get the following* concentration *result,*

$$Pr(|\hat{R}_n[h] - R[h]| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \tag{2.6}$$

Lemma 2.3 (Union Bound). *For n sets A_1, A_2, ..., A_n,*

$$Pr\left(\bigcup_{i=1}^n A_i\right) \leq \sum_{i=1}^n Pr(A_i) \tag{2.7}$$

Let $\mathcal{H}$ be a finite set of predictors, hence

$$Pr\left(\bigcup_{h\in\mathcal{H}} \{|\hat{R}_n[h] - R[h]| \geq \epsilon\}\right) \leq 2|\mathcal{H}|e^{-2n\epsilon^2} \tag{2.8}$$

Exercise: How many samples do we need to bound $Pr\left(\bigcup_{h\in\mathcal{H}}\{|\hat{R}_n[h] - R[h]| \geq \epsilon\}\right) \leq \delta$?

$$n \geq \frac{1}{2\epsilon^2}\log\left(\frac{2|\mathcal{H}|}{\delta}\right) = O\left(\frac{\log|\mathcal{H}| + \log(\delta^{-1})}{\epsilon^2}\right) \tag{2.9}$$

What if the set $\mathcal{H}$ is not finite? For example, the decision rule in *support vector machine*: $h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b)$. Obviously $|\mathcal{H}| = \infty$, so this bound doesn't work. But if we consider the floating-point in computer, the set $\mathcal{H}_{float}$ is finite. Assuming that we work on a 64-bit computer and the dimension of $\boldsymbol{x}$ is d, then $|\mathcal{H}_{float}| = 2^{64(d+1)}$ and the sample size n should be $O_\delta(d/\epsilon^2)$.

Let's loot at another example *neural networks*. For a three-layer network (input layer,

hidden layer and output layer): the input $\boldsymbol{x} \in \mathbb{R}^p$, the output $\boldsymbol{y} \in \mathbb{R}^q$, the weight matrices $W_1 \in \mathbb{R}^{m \times p}$ and $W_2 \in \mathbb{R}^{q \times m}$, the model is

$$y_k = \sigma\left(\sum_{i=1}^{m} (W_2)_{k,i} \sigma\left(\sum_{j=1}^{p} (W_1)_{i,j} x_j\right)\right) \tag{2.10}$$

where σ is the sigmoid function. $|\mathcal{H}_{float}| = 2^{64(mp+mq)}$ and the sample size n should be $O_\delta(m(p+q)/\epsilon^2)$.