EE2211: Spring 2023
Training Error of Nested Models

1 For Regression

Suppose we have a regression problem with a one-dimensional dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$. Each $x_i \in \mathbb{R}$ and $y_i \in \mathbb{R}$. Consider the training step of minimizing the squared ℓ_2 error

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^2} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - \bar{\mathbf{w}}^\top \bar{\mathbf{x}}_i\right)^2 \tag{1}$$

where

$$\bar{\mathbf{w}} = \begin{bmatrix} b \\ w \end{bmatrix} \quad \text{and} \quad \bar{\mathbf{x}}_i = \begin{bmatrix} 1 \\ x_i \end{bmatrix}. \tag{2}$$

Note that (1) is equivalent to

$$\min_{f \in \mathcal{F}_{\text{affine}}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2 \tag{3}$$

where $\mathcal{F}_{\text{affine}}$ is the class of affine functions, which can be written as $\mathcal{F}_{\text{affine}} = \{f(x) = b + wx : b, w \in \mathbb{R}\}$. Now, note that the class of affine functions $\mathcal{F}_{\text{affine}}$ is a subset of $\mathcal{F}_{\text{quad}} = \{f(x) = w_0 + w_1 x + w_2 x^2 : b, w_1, w_2 \in \mathbb{R}\}$. This is written as $\mathcal{F}_{\text{affine}} \subseteq \mathcal{F}_{\text{quad}}$. We know that for any function g, one has $\min_{x \in A} g(x) \leq \min_{x \in B} g(x)$ if $B \subseteq A$, since we have "more elements" to minimize over in A. Thus, if the optimization can in (1) can be done exactly (e.g., using the pseudo-inverse formula),

$$\min_{f \in \mathcal{F}_{\text{quad}}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2 \leq \min_{f \in \mathcal{F}_{\text{affine}}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2. \tag{4}$$

or equivalently,

$$\frac{1}{m} \sum_{i=1}^{m} \left(y_i - (w_0^* + w_1^* x_i + w_2^* x_i^2)\right)^2 \leq \frac{1}{m} \sum_{i=1}^{m} \left(y_i - (b^* + w^* x_i)\right)^2. \tag{5}$$

In other words, the squared loss of when we minimize over the set of quadratic functions cannot be larger than that when we optimize over affine functions.

More generally, if we have two classes of functions $\mathcal{G} \subseteq \mathcal{F}$ (we say that these function classes are *nested*), then

$$\min_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2 \leq \min_{f \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2. \tag{6}$$

Thus, if we increase the model order, the ℓ_2 loss (squared error loss) cannot go up.

Exercise 1. *Prove that for a one-dimensional dataset of size m, a polynomial order of $m-1$ (number of training samples minus 1) suffices to drive the optimized ℓ_2 loss to zero. When can we get away with a polynomial order of $< m-1$?*

Exercise 2. *Generalize the observation in Remark 1 to a d-dimensional dataset.*

Exercise 3. *Further generalize the preceding observations to the use of basis functions other than polynomials (e.g., sinusoids). What is the key property of polynomials we are using?*

2 For Classification

The situation for classification is similar, yet more subtle. The training error (one minus the training accuracy) for a one-dimensional dataset $\{(x_i, y_i)\}_{i=1}^m$ using classifier $f : \mathbb{R} \to \{-1, +1\}$ is

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \neq f(x_i)\}. \tag{7}$$

By the very same logic as for the regression case,

$$\min_{f \in \mathcal{F}_{\text{quad}}} \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \neq f(x_i)\} \leq \min_{f \in \mathcal{F}_{\text{affine}}} \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \neq f(x_i)\}. \tag{8}$$

So if we are minimizing the zero-one loss, the training error cannot go up if we are using a model class $\mathcal{F}$ that contains another one $\mathcal{G}$; see (6) for the regression case.

However, what we have done in this class for binary classification is slightly different. Since optimizing the zero-one loss is intractable (because it is non-differentiable), we have encoded the labels as $\{-1, +1\}$ and treated the problem as a regression problem whose targets are $\{-1, +1\}$. We then minimized the ℓ_2 error

$$\frac{1}{m}\sum_{i=1}^{m} \big(y_i - f(x_i)\big)^2 \tag{9}$$

over functions f belonging to different function classes $\mathcal{F}$, say $\mathcal{F}_{\text{affine}}$ or $\mathcal{F}_{\text{quad}}$. This is, however, *not* minimizing the vanilla training error or zero-one loss in (7). Hence, we cannot conclude that training error does not increase when we minimize f over a function class $\mathcal{F}$ that contains $\mathcal{G}$. In many other machine learning approaches to binary classification, one uses proxies or surrogates to the zero-one loss that are more computationally friendly. For example, one uses the *hinge loss* for *support vector machines* and one uses the *exponential loss* for *boosting*. In our class, for the sake of pedagogical purposes and for unifying the approaches for regression and classification, we have chosen to use the approach in (9) for binary classification.

In most cases of practical relevance though, the training error will not increase as we use nested function classes $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \mathcal{F}_3 \subseteq \ldots$ even though we are optimizing a slightly different loss function in (9).