Online Learning of Smooth Functions

Introduction

Online learning is a model of machine learning in which a learning algorithm, or learner, trains on data that gets revealed sequentially. This contrasts with a lot of other machine learning, where the learner might train on big batches of data at once. Online learning can be useful for predicting something that naturally evolves over time, such as the weather.

We specifically consider the online learning of real-valued functions, which take in numerical inputs and return a numerical output. In our model, the learner tries to predict the outputs of such a function, f, on given inputs. As more data is revealed over time, we should expect the learner's predictions to become more and more accurate; however, this requires *smoothness conditions* that guarantee that the functions being learned behave "nicely."

Finally, we consider an *adversarial* model, in which an adversary chooses inputs and the hidden function f in order to maximize the learner's errors. This contrasts with statistical models, where the inputs are chosen at random. Because of this, the learners are measured by their worst-case performance.

The Setup

Fix a domain X and a class $\mathcal{F}$ of functions from X to $\mathbb{R}$, both of which are known to the learner. In our model, a learning algorithm A learns $\mathcal{F}$ as follows:

- An adversary first selects $f \in \mathcal{F}$.
- Learning then proceeds in trials. On trial $i \geq 0$:
  - The adversary selects and gives A an input $x_i \in X$;
  - A produces $\hat{y}_i \in \mathbb{R}$, its prediction for $f(x_i)$ based on all previously revealed points $(x_j, f(x_j))$;
  - The adversary reveals the true value of $f(x_i)$.

In order to study how well algorithms can learn real-valued functions, we need a way to measure their performance. To do this, fix some real $p \geq 1$. On each trial i, if the learner A makes a mistake, it gains a penalty term $|\hat{y}_i - f(x_i)|^p$. These terms are then summed.

Recall that we are also interested in analyzing worst-case performance. We can define a function called the opt function, which measures the best possible worst-case error for a learner against some class of functions $\mathcal{F}$, depending on the parameter p:

opt

For $p \geq 1$ and a class $\mathcal{F}$ of real-valued functions, $\text{opt}_p(\mathcal{F})$ is the best upper bound on the sum of penalties, $\sum_{i \geq 1} |\hat{y}_i - f(x_i)|^p$, a learner can guarantee against any adversary while learning $\mathcal{F}$.

Note that opt is a sum of penalties, so smaller values of opt indicate better performance from learners.

This can be imagined as a game between the learner and the adversary. The adversary knows exactly what the learner will do at each step, for any combination of inputs; its goal is to maximize the learner's total penalty. In response, the learner's goal is to make its worst-case performance as good as possible, so that the adversary can take the least advantage.

Finally, the parameter p affects how harshly small errors are punished compared to big errors. When p is large, small errors are punished more leniently, which makes it easier for the learner to have a small total penalty. Thus $\text{opt}_p(\mathcal{F})$ will tend to decrease as p increases.

The Single-Variable Problem

We first consider the case where inputs are real numbers between 0 and 1. First, we construct classes $\mathcal{F}$ of functions on $[0, 1]$:

$\mathcal{F}_q$

For $q \geq 1$, $\mathcal{F}_q$ is the class of absolutely continuous functions $f : [0, 1] \to \mathbb{R}$ such that

$$\int_0^1 |f'(x)|^q dx \leq 1.$$

$\mathcal{F}_\infty$ is the class of functions $f : [0, 1] \to \mathbb{R}$ such that $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$ for all $x_1, x_2 \in [0, 1]$.

By Jensen's inequality, $\mathcal{F}_q$ shrinks as q increases; thus $\mathcal{F}_1$ is the broadest class and $\mathcal{F}_\infty$ is the narrowest. Broader classes contain more chaotic functions and are harder for the learner, so $\text{opt}_p(\mathcal{F}_q)$ increases as q decreases.

Previously, it was shown that:

- If $p = 1$ or $q = 1$, then $\text{opt}_p(\mathcal{F}_q) = \infty$;
- If $p \geq 2$ and $q \geq 2$, then $\text{opt}_p(\mathcal{F}_q) = 1$;
- For $\varepsilon \in (0, 1)$ and $q \geq 2$, $\text{opt}_{1+\varepsilon}(\mathcal{F}_q) = \Theta(\varepsilon^{-\frac{1}{2}})$, where the constant factors do not depend on q.

We extended this to results in the previously unexplored case $q \in (1, 2)$, which contains more chaotic functions:

Results in the Single-Variable Case

- For $\varepsilon \in (0, 1)$, $\text{opt}_2(\mathcal{F}_{1+\varepsilon}) = \Theta(\varepsilon^{-1})$;
- For $\varepsilon \in (0, 1)$ and $p \geq 2 + \varepsilon^{-1}$, $\text{opt}_p(\mathcal{F}_{1+\varepsilon}) = 1$.

A Multivariable Generalization

We can also generalize to functions which take in d numerical inputs between 0 and 1 and return a number (functions from $[0, 1]^d$ to $\mathbb{R}$):

$\mathcal{F}_{q,d}$

For $q \geq 1$ and $d \in \mathbb{Z}_{>0}$, $\mathcal{F}_{q,d}$ is the class of functions $f : [0, 1]^d \to \mathbb{R}$ such that any function $g : [0, 1] \to \mathbb{R}$ formed by fixing $d - 1$ arguments of f is in $\mathcal{F}_q$. $\mathcal{F}_{\infty,d}$ is defined similarly.

As before, larger p and q are less harsh on the learner, thus decreasing opt.
In this case, we found the following bounds:

Results in the Multivariable Case

- For $p, q \geq 1$ and $d \in \mathbb{Z}_{>0}$, $\text{opt}_p(\mathcal{F}_{q,d}) \geq d^p \cdot \text{opt}_p(\mathcal{F}_q)$;
- For $p \geq 1$ and $d \in \mathbb{Z}_{>0}$:
  - If $p < d$ then $\text{opt}_p(\mathcal{F}_{\infty,d}) = \infty$;
  - If $p > d$ then $\text{opt}_p(\mathcal{F}_{\infty,d}) \leq \frac{(2^d - 1)d^p}{1 - \frac{2^d}{2^p}}$.

Implications

Our results show that opt is finite for many classes of functions, including the broad classes of single-variable functions where $q < 2$, as well as some classes of multivariable functions that likely show up more in practice. This means that algorithms can learn quite effectively, getting rapidly decreasing errors as they pick up more information, as long as the functions are reasonably "well-behaved." In practice, this could help forecasters refine their models, e.g. by uncovering that some overlooked inputs are missing from the model.