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

Lecture 20: April 9

Lecturer: Pradeep Ravikumar *Scribes: Avikalp Srivastava, Hanyang Li, Haixing Yin*

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

20.1 Continuing Algorithmic Weakening

We begin by looking at an example for the concept of algorithmic weakening.

20.1.1 Denoising Example

Given the signal of interest as X^* in p dimensions, we are going to receive n samples $(y_1, \ldots, y_n)$ which are simply X^* perturbed with some gaussian noise:

$$y_i = X^* + \sigma z_i$$
$$z_i \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_P)$$

A reasonable estimator would be:

$$\bar{y}_n = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{20.1}$$

But, say we have an additional restriction that: $X^* \in S$,

Then the optimal estimator is obtained by projecting $\bar{y}_n$ onto the set S:

$$\tilde{y}_s \in \arg \inf_{z \in S} \|z - \bar{y}_n\|_2 \tag{20.2}$$

One can show that this estimator is also minimax optimal.

If S is specified in a complicated way (eg. exponentially many inequalities), projecting onto this set is hard, and we may want to algorithmically weaken the estimator $\tilde{y}_s$. We can do so by looking at upper bounding the set S with some simpler polytopes, which are easier to project onto (but would be statistically worse).

Let's say we build sets C_1, $C_2, \ldots$ that are progressively tighter constraint sets for the set of interest S.

$$C_1 \supset C_2 \supset ... \supset S \tag{20.3}$$

Given such set constructions, and considering set indices i, j s.t. $i < j$, we have the following:

- It is easier to project onto C_i than onto C_j.
- Given less time, we might only be able to use the easy C_i projection, but with more time we can get a statistically better C_j projection.
- However, in large sample regime, C_i projection can give similar error compared to C_j projection.

Therefore, given a large number of samples, we can get away with projection to a looser set.

Next, we look at a very simple case to illustrate this concept.

20.1.1.1 Lift and Project

Consider the following set S, which is just an ℓ_1-ball:

$$S = \{x \in \mathbb{R}^p \mid \sum_{i=1}^{p} |x_i| \leq 1\} \tag{20.4}$$

This is convex, and shouldn't be hard to project onto. But say we can only work with linear inequalities. One way to represent S using just linear inequalities is:

$$S = \{x \in \mathbb{R}^p \mid \sum_{i=1}^{p} z_i x_i \leq 1, \forall z \in \{-1, 1\}^p\} \tag{20.5}$$

This contains 2^p linear inequalities to specify the polytope. This seems hard, but in general projection to ℓ_1 ball is easy. The reason is use of the strategy *lift and project*.

Instead of defining a polytope over just x, lift and project method defines a polytope over x and z (set $S_{x,z}$), and then uses projection to obtain a set with just x. In this example, we have $S_{x,z}$ defined by $p + 1$ inequalities:

$$S_{x,z} = \{(x, z) \in \mathbb{R}^{2p} \mid \sum_{i=1}^{p} z_i \leq 1,\ -z_i \leq x_i \leq z_i,\ i = 1, ..., p\} \tag{20.6}$$

Projecting this to the first p coordinates gives us back the set S:

$$S = \{x \in \mathbb{R}^p \mid \exists z \in \mathbb{R}^p \quad s.t \sum z_i \leq 1,\ -z_i \leq x_i \leq z_i\ ,\ i = 1, ..., p\} \tag{20.7}$$

For more complicated sets S, lift and project returns outer bounds $(C_1, C_2, \dots)$ instead of giving back S.

Depending on the method used for lift and project, one can get a hierarchy of sets. We mention two particular hierarchies that are used ([PRIHIE]), but don't discuss them further:

- Sherali-Adams Hierarchy - a strictly LP hierarchy.
- Lasserre/Farillo Hierarchy - involving SDPs (semi-definite constraints over matrices).

20.1.2 Another Example: CUT polytope hierarchy

This example is presented with the goal of showing the behavior where solving with small sample set is exponential but large sample set is polynomial in time.

Consider the CUT polytope, which is a convex hull of a set of 1-rank matrices:

$$\text{CUT}_{m\times m} = conv\{mm^T \mid m \in \{-1,+1\}^p\} \subseteq \mathbb{R}^{p\times p}$$

This is convex, but complex since the number of vertices is 2^p. Projecting onto this is super-polynomial.

A convex relaxation named elliptope is given by:

$$\mathcal{E}_{m\times m} = \{X \in \mathbb{R}^{p\times p} \mid X \succeq 0,\ X_{ii} = 1,\ i = 1, ..., p\}$$

An even simpler relaxation is derived using the nuclear norm (sum of singular values)

$$N_{m\times m} = \{X \in \mathbb{R}^{p\times p} \mid \|X\|_* \leq \sqrt{p}\}$$

The hierarchy is:

$$\text{CUT}_{m\times m} \subset \mathcal{E}_{m\times m} \subset N_{m\times m}$$

As an aside, we begin a discussion with application of techniques from uniform laws and generalization analysis, and we'll connect it back to the CUT polytope discussion.

Given any set C, and the denoising setup as before in section 20.1.1, the optimal estimator is given as:

$$\hat{X}_n(C) \in \arg\inf_{X\in C} \|X - \bar{y}_n\|_2^2 \tag{20.8}$$

We'll try to compute the statistical performance of $\hat{X}_n(C)$.

Here, instead of Rademacher complexity, it's easier to deal with Gaussian square complexity:

$$G(D) = \mathbb{E}_{g\sim N(\mathbf{0},\mathbf{I})}[\sup_{a\in D}\langle a, g\rangle^2] \tag{20.9}$$

We also define some other notations:

$$B_2(1) = \{X \mid \|X\|_2 \leq 1\} \qquad \text{(unit } \ell_2\text{-ball)} \tag{20.10}$$

$$T_C(X^*) = \text{CONE}\{X - X^* \mid X \in C\} \qquad \text{(cone of all directions leading away from } X^*) \tag{20.11}$$

Theorem 20.1

$$\mathbb{E}[\|\hat{X}_n(C) - X^*\|_2^2] \leq \frac{\sigma^2}{n} G(T_C(X^*) \cap B_2(1))$$

Intuition: The larger C is, the larger the Gaussian squared complexity, requiring larger number of samples to keep the expected error the same.

Proof:

$$y_i = x^* + \sigma z_i$$

$$\bar{y}_n = x^* + \frac{\sigma}{\sqrt{n}} z \qquad \text{(where } z \text{ is a standard gaussian)}$$

Now, the stationary condition for constrained optimization involves setting the dot product of the gradient with all feasible directions to be non-negative. The gradient here is $2(\hat{X}_n(C) - \bar{y}_n)$, so:

$$\begin{aligned} & 2\langle \hat{X}_n(C) - \bar{y}_n, X - \hat{X}_n(C) \rangle \geq 0 \qquad \forall X \in C \\ \implies & 2\langle \hat{X}_n(C) - \bar{y}_n, \hat{X}_n(C) - X^* \rangle \leq 0 \\ \implies & \langle \hat{X}_n(C) - X^* - \frac{\sigma}{\sqrt{n}} z, \hat{X}_n(C) - X^* \rangle \leq 0 \qquad \text{(by substituting } \bar{y}_n\text{)} \end{aligned}$$

$$\begin{aligned} \implies \|\hat{X}_n(C) - X^*\|_2^2 \leq & \ \langle \hat{X}_n(C) - X^*, \frac{\sigma}{\sqrt{n}} z \rangle \\ & = \|\hat{X}_n(C) - X^*\|_2 \langle \frac{\hat{X}_n(C) - X^*}{\|\hat{X}_n(C) - X^*\|_2}, \frac{\sigma}{\sqrt{n}} z \rangle \\ & \leq \frac{\sigma}{\sqrt{n}} \|\hat{X}_n(C) - X^*\|_2 \sup_{a \in B_2(1) \cap T_C(X^*)} |\langle a, g \rangle| \\ \implies \|\hat{X}_n(C) - X^*\|_2 \leq & \ \frac{\sigma}{\sqrt{n}} \sup_{a \in D} |\langle a, g \rangle| \qquad \text{(defining } D \text{ as } B_2(1) \cap T_C(X^*)\text{)} \end{aligned}$$

Squaring and taking the expectation gives us the result in the theorem. ■

Now that we have a bound on the error ($\epsilon(p) = \mathbb{E}[\|\hat{X}_n(C) - X^*\|_2^2]$) , we loop back and think of the three quantities of interest: error ($\epsilon(p)$), number of samples ($n(p)$), and time taken ($t(p)$) - for $\text{CUT}_{m\times m}$, $\mathcal{E}_{m\times m}$, and $N_{m\times m}$.

First, see that:

$$n(p) \geq \sigma^2 G(T_C(X^*) \cap B_2(1)) \implies \epsilon(p) \leq 1$$

We fix the error bound at 1 and look at the tradeoff between $n(p)$ and $t(p)$:

$$\begin{aligned} \text{CUT}_{m\times m} &: n(p) = c_1\sqrt{p}, \quad t(p) = \text{super-poly}(p) \\ \mathcal{E}_{m\times m} &: n(p) = c_2\sqrt{p}, \quad t(p) = p^{2.25} \\ N_{m\times m} &: n(p) = c_3\sqrt{p}, \quad t(p) = p^{1.5} \end{aligned}$$

where $0 < c_1 < c_2 < c_3$

This means that if we have a constant number of more samples, we can solve $N_{m\times m}$ or $\mathcal{E}_{m\times m}$ instead of $\text{CUT}_{m\times m}$ to get the same error bound.

20.2 Sequential Prediction

In the most basic version of the sequential prediction problem, the predictor or forecaster observes one after another the elements of a sequence $y_1, y_2, ...$ of symbols. At each time $t = 1, 2, ...$, before the tth symbol of the sequence is revealed, the forecaster guesses its value y_t on the basis of the previous $t - 1$ observations.

20.2.1 Classical statistical theory

In the classical statistical theory of sequential prediction, the sequence of elements, which we call outcomes, is assumed to be realization of a stationary stochastic process.

20.2.1.1 Framework

Sequence: $y_1, y_2, ...$

Goal: predict y_t given $y_1, ..., y_{t-1}$

Machine Learning approach: Model $P(y_t|y_1, ..., y_{t-1})$

20.2.1.2 Limitation

The ML approach won't work in some cases:

1. The sequential data may not arise from any stochastic model.
2. Stochastic model varies with time t, which is non-stationary.

20.2.2 Learning with expert advice

Now, we abandon the basic assumption that the outcomes are generated by an underlying stochastic process and view the sequence $y_1, y_2, ...$ as the product of some unknown and unspecified mechanism. To contrast it with stochastic modeling, this approach has often been referred to as prediction of *individual sequences*.

In basic model, the performance of the forecaster is measured by the loss accumulated during many rounds of prediction, where loss is scored by some fixed loss function. Since we want to avoid any assumption on the way the sequence to be predicted is generated, there is no obvious baseline against which to measure the forecaster's performance. To provide such a baseline, we introduce a class of *reference forecasters*, also called *experts*. These experts make their prediction available to the forecaster before the next outcome is revealed. The forecaster can then make its own prediction depending on the experts' "advice" with the aim of keeping its cumulative loss close to that of the best expert in the class.

We can view an expert as a truly black box of unknown computational power, possibly with access to private sources of side information. In some applications, the class of experts is collectively regarded as a statistical model, where an expert in the class represents an optimal forecaster for some given "state of nature".

Notice that assuming the sequence is completely arbitrary is a too strong assumption for our prediction problem. Thus, we just want a relatively good forecaster as our baseline model. That is to say, our objective is to do at least as good as the best expert without any assumptions about the distribution of data.

20.2.2.1 Protocol

Prediction with expert advice is based on the following protocol for sequential decisions:

The decision maker is a forecaster whose goal is to predict arbitrary sequence $y_1, y_2, ..., y_t \in \mathcal{Y}$ where $\mathcal{Y}$ is our outcome space. The forecaster's predictions $\hat{p}_1, \hat{p}_2, ... \in \mathcal{D}$ where $\mathcal{D}$ is our decision space which is typically convex subset, but not necessary. In some special cases we take $\mathcal{D} = \mathcal{Y}$, but in general they may be different. For example, $\mathcal{Y}$ might be a set of binary values $\{0, 1\}$, while $\mathcal{D}$ might be a set of continuous values $[0, 1]$.

The forecaster computes its predictions in a sequential fashion, and its predictive performance is compared to that of a set of reference forecasters that we call experts. More precisely, at each time t the forecaster has access to the set $\{f_{E,t}, E \in \epsilon\}$ of expert predictions $f_{E,t} \in \mathcal{Y}$, where ϵ is a fixed set of indices for the experts which could be finite or infinite set.

The predictions of forecaster and experts are scored using a nonnegative loss function $\ell : \mathcal{D} \times \mathcal{Y} \to \mathbb{R}$.

This prediction protocol can be naturally viewed as the following repeated game between forecaster who makes guesses $\hat{p}_t$, and "environment" who chooses the expert advice $\{f_{E,t} \in \mathcal{Y}\}$ and sets the true outcomes y_t.

For each round $t = 1, 2, ...,$

1. Environment chooses y_t, $[f_{E,t}]$, and $E \in \epsilon$
2. Environment reveals expert advice $[f_{E,t}]$, and $E \in \epsilon$
3. Forecaster chooses $\hat{P}_t \in \mathcal{D}$
4. Environment reveals y_t; Forecaster incurs a loss $\ell(\hat{P}_t, y_t)$; Experts incur loss $\ell(f_{E,t}, y_t)$

20.2.2.2 Notation

- $R_{E,n} = \sum_{t=1}^{n}(l(\hat{P}_t, y_t) - l(f_{E,t}, y_t) \equiv$ (cumulative) regret w.r.t expert E
  (As it measures how much the forecaster regrets, in hindsight, of not having followed the advice of expert E)
- $r_{E,t} = l(\hat{P}_t, y_t) - l(f_{E,t}, Y_t) \equiv$ instantaneous regret
- $R_{E,n} = \sum_{t=1}^{n} r_{E,t}$
- $L_n = \sum_{t=1}^{n} l(\hat{P}_t, y_t)$
- $L_{E,n} = \sum_{t=1}^{n} l(f_{E,t}, y_t)$
- $R_{E,n} = L_n - L_{E,n}$

The forecaster's goal is to keep as small as possible the cumulative regret, which is to do as well as the best expert ($\max_{E \in \epsilon} R_{E,n}$ is small).

$$\max_{E \in \epsilon} R_{E,n} = o(n) \iff \frac{1}{n} \max_{E \in \epsilon} R_{E,n} \xrightarrow{n \to \infty} 0$$

$$R_{E,n} = L_n - L_{E,n} \iff \frac{1}{n}(L_n - \min_{E \in \epsilon} L_{E,n}) \xrightarrow{n \to \infty} 0 \quad \forall Y_t \quad \forall f_{E,t}$$

where $L^* = \min_{E \in \epsilon} L_{E,n}$

Note that the forecaster has no assumption on Y_t and $f_{E,t}$ here.

20.2.3 Example

Consider the problem of predicting an unknown sequence $y_1, y_2, ...$ of bits $y_t \in \{0, 1\}$. At each time t, the forecaster first makes its guess $\hat{P}_t \in \{0, 1\}$ for y_t. Then the true bit y_t is revealed and the forecaster finds out whether its prediction was correct. The forecaster makes a prediction $\hat{P}_t$ based on the advice of N experts. This advice takes the form of a binary vector $(f_{1,t}, ..., f_{N,t})$, where$f_{i,t} \in \{0, 1\}$.

Goal: to bound the number of time steps t in which $\hat{P}_t \neq y_t$, or equivalently , to bound the number of mistakes made by the forecaster.

To simplify, we hold a strong assumption:, there is some expert i that makes no mistakes. $\exists i \in [N]$ s.t $f_{i,t} = Y_t \ \forall t$, where i is unknown to the forecaster. Then, one strategy is as follows:

- Start with assigning a weight w_j to each expert j: $w_j = 1 \ \forall j \in [N]$
- for each time step t, we go with majority: $\hat{P}_t = \begin{cases} 1 & if \ \sum w_i \mathbb{1}(f_{i,t} == 1) / \sum w_i > \frac{1}{2} \\ 0 & otherwise \end{cases}$
- After y_t is revealed, if $\hat{p}_t \neq y_t$, set $w_k = 0 \ \forall k$ s.t $f_{i,t} \neq y_t$

Claim: The number of rounds in which the forecaster makes a mistake is $m \leq \lfloor log_2 N \rfloor$

Proof: Let W_m be the sum of the weights of all experts after the forecaster has made m mistakes.

$W_m = \sum_{i=1}^{n} w_{i,m}$, initially, $m = 0$ and $W_0 = N$

When the forecaster makes its mth mistake, at least half of the experts that have been always correct so far make their first mistake because the forecaster takes the majority vote of the experts. This implies that $W_m \leq \frac{W_{m-1}}{2}$.

Recalling that expert i never makes a mistake, we know that $w_i = 1$, which implies that $W_m \geq 1$.

Using above together, we thus find that

$$1 \leq W_m \leq \frac{W_{m-1}}{2} \leq \frac{W_0 \equiv N}{2^m}$$

Solving for m gives the claimed inequality

$$m \leq \lfloor log_2 N \rfloor$$

20.2.4 Application

The concepts and methods of sequential prediction can be applied to many fields, such as:

- Repeated games
- Data compression (to encode a sequence coming in)
- Gambling/ investment without assuming its distribution
- Online learning

References

[BL] Cesa-Bianchi, Nicolo and Lugosi, Gabor, "Prediction, Learning, and Games," 2006

[PRIHIE] https://www.win.tue.nl/ nikhil/hierarchies/madhur.pdf