A Tutorial on Online Supervised Learning with Applications to Node Classification in Social Networks

Alexander Rakhlin
University of Pennsylvania

Karthik Sridharan
Cornell University

August 31, 2016

We revisit the elegant observation of T. Cover [Cov65] which, perhaps, is not as well-known to the broader community as it should be. The first goal of the tutorial is to explain—through the prism of this elementary result—how to solve certain sequence prediction problems by modeling sets of solutions rather than the unknown data-generating mechanism. We extend Cover's observation in several directions and focus on computational aspects of the proposed algorithms. The applicability of the methods is illustrated on several examples, including node classification in a network.

The second aim of this tutorial is to demonstrate the following phenomenon: it is possible to predict as well as a combinatorial "benchmark" for which we have a certain multiplicative approximation algorithm, even if the exact computation of the benchmark given all the data is NP-hard. The proposed prediction methods, therefore, circumvent some of the computational difficulties associated with finding the best model given the data. These difficulties arise rather quickly when one attempts to develop a probabilistic model for graph-based or other problems with a combinatorial structure.

1 The basics of bit prediction

Consider the task of predicting an unknown sequence $\boldsymbol{y} = (y_1, \dots, y_n)$ of ± 1's in a streaming fashion. At time $t = 1, \dots, n$, a forecaster chooses $\widehat{y}_t \in \{\pm 1\}$ based on the history $y_1, \dots, y_{t-1}$ observed so far. After this prediction is made, the value y_t is revealed to the forecaster. The average number of mistakes incurred on the sequence is

$$\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}, \tag{1}$$

where $\mathbf{1}\{S\}$ is 1 if S is true, and 0 otherwise. A randomized algorithm $\mathcal{A}$ is determined by the means

$$\widehat{q}_t = \widehat{q}_t(y_1, \dots, y_{t-1}) \in [-1, 1], \qquad t = 1, \dots, n \tag{2}$$

of the distributions $\mathcal{A}$ puts on the outcomes $\{\pm 1\}$ at time t. The expected average number of mistakes made on the sequence $\boldsymbol{y}$ by a randomized algorithm $\mathcal{A}$ is

$$\mu_{\mathcal{A}}(\boldsymbol{y}) = \mathbb{E}\left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right], \tag{3}$$

where the expectation is with respect to the random choices $\widehat{y}_t$, drawn from the distributions with means $\widehat{q}_t(y_1, \ldots, y_{t-1})$, $t = 1, \ldots, n$.

Whenever a prediction algorithm $\mathcal{A}$ has low expected error on some sequence $\boldsymbol{y}$, it must be at the expense of being worse on other sequences. Why? On average over the 2^n sequences, the algorithm necessarily incurs an error of 1/2. Indeed, denoting by $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ a sequence of independent unbiased ± 1-valued (Rademacher) random variables, it holds that

$$\frac{1}{2^n} \sum_{\boldsymbol{y}} \mu_{\mathcal{A}}(\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\varepsilon}} \mathbb{E} \left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1} \{ \widehat{y}_t \neq \varepsilon_t \} \right] = \frac{1}{2} \tag{4}$$

by an elementary inductive calculation, keeping in mind that $\widehat{q}_t = \widehat{q}_t(\varepsilon_1, \ldots, \varepsilon_{t-1})$. As a consequence, it is impossible to compare prediction algorithms when all sequences are treated equally.

Evidently, any algorithm $\mathcal{A}$ induces a function $\mu_{\mathcal{A}}$ on the hypercube $\{\pm 1\}^n$, whose average value is 1/2. Cover [Cov65] asked the converse: given a function $\phi : \{\pm 1\}^n \to \mathbb{R}$, is there an algorithm $\mathcal{A}$ with the property

$$\forall \boldsymbol{y}, \quad \mu_{\mathcal{A}}(\boldsymbol{y}) = \phi(\boldsymbol{y}). \tag{5}$$

In words, if we specify the average number of mistakes we are willing to tolerate for each sequence, is there an algorithm that achieves the goal? If such an algorithm exists, we shall say that ϕ is *achievable*. Let us call ϕ *stable* if

$$|\phi(\ldots, +1, \ldots) - \phi(\ldots, -1, \ldots)| \leq \frac{1}{n} \tag{6}$$

for any coordinate, keeping the rest fixed. Cover's observation [Cov65] is now summarized as

Lemma 1. Suppose $\phi : \{\pm 1\}^n \to \mathbb{R}$ is stable. Then

$$\phi \text{ is achievable if and only if } \mathbb{E}\phi = 1/2,$$

where the expectation is under the *uniform* distribution.

That is, for any function ϕ that does not change too fast along any edge of the hypercube, there exists an algorithm that attains the average number of mistakes given by ϕ if and only if ϕ is 1/2 on average over all 2^n sequences.

As an immediate consequence, for any stable ϕ, $\mathbb{E}\phi \geq 1/2$ is equivalent to existence of an algorithm with

$$\forall \boldsymbol{y}, \quad \mu_{\mathcal{A}}(\boldsymbol{y}) \leq \phi(\boldsymbol{y}).$$

This latter version of the Lemma will be used in the sequel, and we shall say that ϕ is achievable even if (5) holds with an inequality.

Perhaps, it is worth emphasizing the following message of the lemma:

Existence of a forecasting strategy with a given mistake bound for an arbitrary sequence can be verified by checking a probabilistic inequality.

The proof of the more general multi-class statement (Lemma 2) appears in the appendix; it uses backward induction, and may be viewed as a "potential function" argument.

Example 1. Let $\bar{\boldsymbol{y}} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{y_i = 1\}$ denote the proportion of +1's in the sequence. Take

$$\phi(\boldsymbol{y}) = \min\{\bar{\boldsymbol{y}}, 1 - \bar{\boldsymbol{y}}\} + Cn^{-1/2}.$$

It is an exercise to show that the mean of this function with respect to the uniform distribution is at least 1/2 for an appropriate constant C, and that ϕ is stable. Hence, *there exists* a randomized prediction algorithm $\mathcal{A}$ that takes advantage of imbalanced sequences, in the sense that

$$\forall \boldsymbol{y}, \quad \mu_{\mathcal{A}}(\boldsymbol{y}) \le \min\{\bar{\boldsymbol{y}}, 1 - \bar{\boldsymbol{y}}\} + Cn^{-1/2}. \tag{7}$$

For instance, if the sequence $\boldsymbol{y}$ ends up having 30% of 1's, the algorithm, in expectation, will incur roughly 30% proportion of errors (the extra term $Cn^{-1/2}$ is small for large enough n). Notably, the mistake guarantee holds for *any* sequence without any stochasticity assumption on its nature[1] and the imbalance of the sequence need not be known until the very end of the n rounds. The existence of such a prediction strategy may seem rather surprising, and for the intrigued reader that attempts to solve this problem, let us give a hint: no deterministic method will work.

2 Modeling solutions through ϕ

As we have seen, there is no algorithm that can predict sequences uniformly better than another algorithm. Thankfully, we do not care about *all* sequences. A typical prediction problem has some structure that informs us of the sequences we should focus on. The structure is often captured through a stochastic description of the generative process, such as an i.i.d. or an autoregressive assumption. The stochastic assumption, however, may not be justified in applications that involve complex interactions and time dependencies, such as in the social network example below.

The approach in this tutorial is different: we provide a non-stochastic description of the "prior knowledge" via the function ϕ. The function specifies the expected proportion of mistakes we are willing to tolerate on each sequence. We tilt ϕ down towards the sequences we care about, at the expense of making it larger on some other sequences that we are unlikely to see anyway. Lemma 1 guarantees existence of a prediction strategy with proportion of mistakes given by ϕ, as long as ϕ is stable and at least 1/2 on average. Furthermore, given ϕ, the algorithm is simple to state, as we will see below.

In 1950's, David Hagelbarger [Hag56] and Claude Shannon [Sha53] at the Bell Labs built the so-called "mind reading machines" to play the game of matching pennies. According to some accounts,[2] the machine was consistently predicting the sequence of bits produced by untrained human players better than 50%.[3] Of course, the only reason a machine can predict consistently better than chance is that humans cannot enter "truly random" sequences.

How can we design such a machine? Using the approach outlined above, we would need to capture the possible patterns of behavior we might see in the sequences and encode this knowledge in ϕ. We have already seen in Example 1 how to take advantage of imbalanced sequences. Of course,

[1] In [Bla95], D. Blackwell draws parallels between an almost sure version of (7) (based on "approachability") and the corresponding i.i.d. statement.

[2] http://backup.itsoc.org/review/meir/node1.html

[3] Here is a modern version of this machine: http://www.mindreaderpro.appspot.com by Y. Freund and colleagues.

this may not be the only structure available, and we shall now describe a few general approaches to building ϕ.

The first basic construction will be called *aggregation*. Suppose $\phi_1, \dots, \phi_N$ are N stable functions, each satisfying $\mathbb{E}\phi_i \geq 1/2$. It is then possible to show that the best-of-all aggregate

$$\phi(\boldsymbol{y}) = \min_{j \in \{1,\dots,N\}} \phi_j(\boldsymbol{y}) + C_{n,N} \tag{8}$$

is stable and satisfies $\mathbb{E}\phi \geq 1/2$ for

$$C_{n,N} = \sqrt{\frac{c \log N}{n}},$$

with an absolute constant c. The penalty $C_{n,N}$ for aggregating N "experts" depends only logarithmically on N, and diminishes when n is large. A reader familiar with the literature on prediction with expert advice will recognize the form of $C_{n,N}$ as a *regret bound* of the Exponential Weights algorithm.[4]

Another useful (and most-studied) way to construct ϕ is by taking a subset $F \subseteq \{\pm 1\}^n$ and letting

$$\phi(\boldsymbol{y}) = \mathrm{d_H}(\boldsymbol{y}, F) + C_{n,F}, \tag{9}$$

the normalized Hamming distance between $\boldsymbol{y}$ and the set F, penalized by $C_{n,F}$. Recall that the normalized Hamming distance is

$$\mathrm{d_H}(\boldsymbol{y}, F) \triangleq \min_{\boldsymbol{w} \in F} \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{y_t \neq w_t\},$$

where $\boldsymbol{w} = (w_1, \dots, w_n)$. The definition in (9) automatically ensures stability of ϕ, and the smallest $C_{n,F}$ that guarantees $\mathbb{E}\phi \geq 1/2$ is

$$C_{n,F} = \frac{1}{2n} \mathbb{E} \max_{\boldsymbol{w} \in F} \langle \boldsymbol{\varepsilon}, \boldsymbol{w} \rangle \triangleq \mathscr{R}(F),$$

the *Rademacher averages* of the set F.[5] By Lemma 1, there is a randomized prediction algorithm that incurs $C_{n,F}$ proportion of mistakes on any sequence in F, and the performance degrades linearly with the distance to the set.

Observe that the ϕ function in Example 1 can be written as (9) with $F = \{-\mathbf{1}, \mathbf{1}\}$. This is the simplest nontrivial set F, since matching the performance of a singleton $F = \{\boldsymbol{w}\}$ simply amounts to outputting this very sequence.

As one makes F a larger subset of the hypercube, the Hamming distance from any $\boldsymbol{y}$ decreases, yet the overall penalty $C_{n,F}$ becomes larger. On the extreme of this spectrum is $F = \{\pm 1\}^n$. Insisting on a small error on this set is not possible, and, indeed,

$$\mathscr{R}(\{\pm 1\}^n) = \frac{1}{2n} \mathbb{E} \max_{\boldsymbol{w} \in \{\pm 1\}^n} \langle \boldsymbol{\varepsilon}, \boldsymbol{w} \rangle = \frac{1}{2},$$

the performance of random guessing.

The goal is now clear: for the problem at hand, we would like to define F to be large enough to capture the underlying structure of solutions, yet not too large. In Section 5 we come back to this issue when discussing combinatorial relaxations of F.

[4]In contrast to an algorithmic proof of the regret bound, we derived $C_{n,N}$ as a value necessary to ensure $\mathbb{E}\phi \geq 1/2$.

[5]We include the factor 1/2 in the definition of Rademacher averages.

3 Application: node classification

We now discuss an application of Lemma 1. Let $G = (V, E)$ be a known undirected graph representing a social network. At each time step t, a user in the network opens her Facebook page, and the system needs to decide whether to classify the user as type "−1" or "+1", say, in order to decide on an advertisement to display. We assume here that the feedback on the "correct" type is revealed to the system after the prediction is made. The more natural *partial information* version of this problem is outside the scope of this short tutorial, and we refer the reader to [RS16].

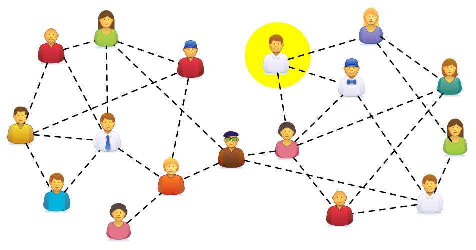
Figure 1: Two-community structure.

The prediction should be made based on all the information revealed thus far (the types of users in the network that appeared before time t), the global graph structure, and the position of the current user in the network. In Section 8 we will also discuss the version of this problem where covariate information about the users is revealed, but at the moment assume that the graph itself provides enough signal to form a prediction. A fascinating question is: what types of ϕ functions capture the graph structure relevant to the problem? Below we provide two examples, only scratching the surface of what is possible.

3.1 Community structure

Suppose we have a hunch that the type of the user (+1 or −1) is correlated with the community to which she belongs. For simplicity, suppose there are two communities, more densely connected within than across (see Figure 1). To capture the idea of correlating communities and labels, we set ϕ to be small on labelings that assign homogenous values within each community. We make the following simplifying assumptions: (i) $|V| = n$, (ii) we only predict the label of each node once, and (iii) the order in which the nodes are presented is fixed (this assumption is easily removed). Smoothness of a labeling $\boldsymbol{y} \in \{\pm 1\}^n$ with respect to the graph may be computed via

$$\sum_{(u,v)\in E} \mathbf{1}\{\boldsymbol{y}(u) \neq \boldsymbol{y}(v)\} = \frac{1}{4} \sum_{(u,v)\in E} (\boldsymbol{y}(u) - \boldsymbol{y}(v))^2 \tag{10}$$

where $\boldsymbol{y}(v) \in \{\pm 1\}$ is the label in $\boldsymbol{y}$ that corresponds to vertex $v \in V$. This function in (10) counts the number of disagreements in labels at the endpoints of each edge. The value is also known as the size of the cut induced by $\boldsymbol{y}$ (the smallest possible being MinCut). As desired, the function in (10) gives a smaller value to the labelings that are homogenous within the communities. A more concise way to write (10) is in terms of the graph Laplacian

$$\boldsymbol{y}^{\mathsf{T}} L \boldsymbol{y}, \tag{11}$$

where $L = D - A$, the diagonal matrix D contains degrees of the nodes, and A is the adjacency matrix.

Unfortunately, the function in (11) is not stable. It also has an undesirable property, illustrated by the following example. The cut size is $n-1$ for a star graph, where $n-1$ nodes, labeled as +1, are connected to the center node, labeled as −1. The large value of the cut does not capture the simplicity of this labeling, which is only one bit away from being a constant +1.

Instead, we opt for the indirect definition (9). More precisely, we define

$$F_\kappa = \left\{ \boldsymbol{y} \in \{\pm 1\}^n \ : \ \boldsymbol{y}^\top L \boldsymbol{y} \le \kappa \right\} \tag{12}$$

for $\kappa \ge 0$, and then set

$$\phi(\boldsymbol{y}) = \mathrm{d_H}(\boldsymbol{y}, F_\kappa) + C_{n, F_\kappa}. \tag{13}$$

Parameter κ should be larger than the value of MinCut, for otherwise the set F_κ is empty. The function ϕ has the interpretation as the proportion of vertices whose labels need to be flipped to achieve the value at most κ for the cut, compensated by the Rademacher averages of the set F_κ. While we can give some straightforward bounds on the Rademacher averages of F_κ, the investigation of this value for various graphs, including random ones, is an interesting research question.

While MinCut is computationally easy, the calculation of ϕ becomes NP-hard in general if we allow $[-1, 1]$-valued weights $w_{(u,v)}$ on the edges and define F_κ with respect to the *weighted* Laplacian

$$\sum_{(u,v) \in E} w_{(u,v)} (\boldsymbol{y}(u) - \boldsymbol{y}(v))^2. \tag{14}$$

Such a definition can be used to model trust-distrust networks, and we certainly would like to develop computationally efficient methods for this problem. Somewhat surprisingly, this is possible in certain cases, even though evaluating ϕ is computationally hard. See Sections 6 and 7 for details.

3.2 Exercise: predicting voting preferences

Suppose n individuals (connected via the known social network as in the previous example) arrive to the voting station one by one, and we are predicting whether they will vote for Grump or for Blinton. After our prediction is made, the voter reveals her true binary preference. Suppose we know the individual's place in the network and the voting preferences of the individuals observed thus far. Our task is to design an online prediction algorithm that makes as few mistakes as possible.

Suppose we have prior knowledge that Grump supporters may be described well by a ball in the network (a ball with center v and radius r is the set of vertices at most r hops away from v), but the center of this ball in the network is not known. Suppose each individual has at most d friends. We leave it as an exercise to design a ϕ function for this prediction problem.

4 Extension to multi-class prediction

We now extend the result of Cover to k-ary outcomes, i.e. $y_t \in \{1, \ldots, k\}$. As before, the expected prediction error is given by

$$\mathbb{E}\left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{ \widehat{y}_t \neq y_t \right\} \right],$$

but uniformly random guessing now incurs an expected cost of $1 - 1/k$. By the same token, on average over the k^n sequences, any algorithm must incur the expected cost of $1 - 1/k$.

Let us define a couple of shorthands. We shall use the notation $a_{1:t} \triangleq \{a_1, \ldots, a_t\}$, $[k] \triangleq \{1, \ldots, k\}$, and denote the set of probability distributions on k outcomes by Δ_k.

We shall say that $\phi : [k]^n \to \mathbb{R}$ is stable if for any coordinate (and holding the rest fixed),

$$\max_{r \in [k]} \phi(\dots, r, \dots) - \frac{1}{k} \sum_{i=1}^{k} \phi(\dots, i, \dots) \leq \frac{1}{nk}. \tag{15}$$

We now overload the notation and define a randomized forecasting strategy as a collection of distribution-valued functions of histories:

$$\widehat{q}_t = \widehat{q}_t(y_{1:t-1}) \in \Delta_k.$$

Lemma 2. Suppose $\phi : [k]^n \to \mathbb{R}$ is stable. Then

$$\phi \text{ is achievable if and only if } \mathbb{E}\phi = 1 - \tfrac{1}{k},$$

where the expectation is under the *uniform* distribution on $[k]^n$.

Once again, it follows from the Lemma that $\mathbb{E}\phi \geq 1 - \frac{1}{k}$ is equivalent to existence of a strategy with $\mu_{\mathcal{A}} \leq \phi$. Lemma 1 can be seen as a special case of Lemma 2 for $k = 2$.

5 Computation

By repeatedly referring to "existence of a prediction strategy" in the previous sections, we, perhaps, gave the impression that these methods are difficult to find. Thankfully, this is not the case.

5.1 The exact algorithm

The proofs of Lemma 1 and 2 are constructive, and the algorithms are easy to state. For binary prediction ($k = 2$), the randomized algorithm is defined on round t by the mean

$$q_t^*(y_{1:t-1}) = n \cdot \mathbb{E}[\phi(y_{1:t-1}, -1, \varepsilon_{t+1:n}) - \phi(y_{1:t-1}, +1, \varepsilon_{t+1:n})] \tag{16}$$

of the distribution on the outcomes $\{\pm 1\}$, where the expectation is over independent Rademacher random variables $\varepsilon_{t+1}, \dots, \varepsilon_n$. The prediction $\widehat{y}_t \in \{\pm 1\}$ is then a random draw such that $\mathbb{E}\widehat{y}_t = q_t^*$.

The two evaluations of ϕ in (16) are performed at neighboring vertices of the hypercube differing in the t-th coordinate. If the function values are equal in expectation, the mean q_t^* is equal to zero, which corresponds to the uniform distribution on $\{\pm 1\}$. In this case, the function ϕ does not provide any guidance on which prediction to prefer. On the other hand, if the absolute difference is $1/n$ in expectation (the largest allowed by stability), the distribution is supported on one of the outcomes and prediction is deterministic. Between these extremes, the difference in values of ϕ measures the influence of t-th coordinate on the potential function ϕ, where the past outcomes $y_1, \dots, y_{t-1}$ have been fixed and future is uniformly-random. We emphasize that Rademacher random variables for future rounds are purely an outcome of the minimax analysis, as we assume no generative mechanism for the sequence $\boldsymbol{y}$.

For $k > 2$, the randomized algorithm is defined on round t by a distribution[6] on $\{1, \dots, k\}$, and the optimal choice is given by

$$q_t^*(y_{1:t-1}) = \underset{q \in \Delta_k}{\operatorname{argmin}} \max_{j \in [k]} \left\{ -q^\top \boldsymbol{e}_j - n\mathbb{E}\phi(y_{1:t-1}, j, u_{t+1:n}) \right\} \tag{17}$$

where the expectation is with respect to $u_{t+1}, \dots, u_n$, each independent uniform on $[k]$. Given that the values $\mathbb{E}\phi(y_{1:t-1}, j, u_{t+1:n})$ have been computed for each j, the minimization in (17) is performed by a simple water-filling $O(k)$-time algorithm which can be found in the proof of Lemma 2 (see also [RS16]). The actual prediction is then a random draw $\widehat{y}_t \sim q_t^*$.

5.2 Randomization

Computing the expectations in (16) and (17) may be costly. Thankfully, a doubly-randomized strategy works by drawing a random sequence per iteration. For binary prediction, the algorithm on round t becomes: draw independent Rademacher random variables $\varepsilon_{t+1}, \dots, \varepsilon_n$, compute

$$\widetilde{q}_t^*(y_{1:t-1}, \varepsilon_{t+1:n}) = n[\phi(y_{1:t-1}, -1, \varepsilon_{t+1:n}) - \phi(y_{1:t-1}, +1, \varepsilon_{t+1:n})] \tag{18}$$

and draw $\widehat{y}_t$ from the distribution on $\{\pm 1\}$ with the mean $\widetilde{q}_t^*$. This randomized strategy was essentially proposed in [CBS11].

For $k > 2$, we draw uniform independent $u_{t+1}, \dots, u_n$, solve for

$$\widetilde{q}_t^*(y_{1:t-1}, u_{t+1:n}) = \underset{q \in \Delta_k}{\operatorname{argmin}} \max_{j \in [k]} \left\{ -q^\top \boldsymbol{e}_j - n\phi(y_{1:t-1}, j, u_{t+1:n}) \right\} \tag{19}$$

and then draw prediction $\widehat{y}_t$ from the distribution $\widetilde{q}_t^*$.

Lemma 3. The doubly-randomized strategies (18) and (19) enjoy, in expectation, the same mistake bounds as, respectively, (16) and (17).

In the binary prediction case, the proof of Lemma 3 is immediate by the linearity of expectation. The analogous argument for (19) is more tricky and follows from a more general *random playout* technique introduced in [RSS12]. This technique also yields a proof of Lemma 3 (for both binary and multi-class cases) for *adaptively chosen* sequences of outcomes, an issue we have not yet discussed (see also Section 8 below).

[6]We hope the difference in the meaning of q_t^* as a distribution (for $k > 2$) vs a mean of a distribution (for $k = 2$) will not cause confusion.

6 Relaxations and the power of improper learning

In the rest of the tutorial, we focus on the binary prediction problem for simplicity. Recall that the computation in (16) involves drawing random bits $\varepsilon_{t+1}, \ldots, \varepsilon_n$ and evaluating the ϕ function on two neighboring vertices of the hypercube. If ϕ is defined as

$$\phi(\boldsymbol{y}) = \mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F) + C_{n,F}, \tag{20}$$

then computing (16) or (18) involves comparing two distances to the set F, as shown in Figure 2. Note that the knowledge of $C_{n,F}$ is not needed, as this value cancels off in the difference.

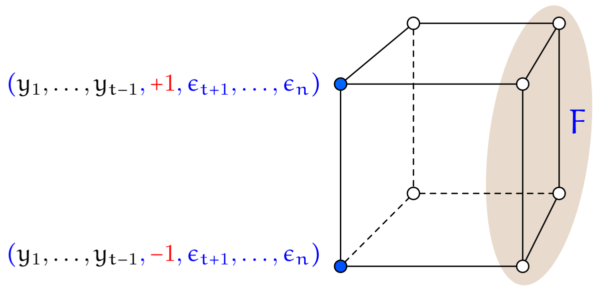
$(y_1, \ldots, y_{t-1}, +1, \epsilon_{t+1}, \ldots, \epsilon_n)$
$(y_1, \ldots, y_{t-1}, -1, \epsilon_{t+1}, \ldots, \epsilon_n)$
F

Figure 2: Randomized strategy involves computing the difference of normalized Hamming distances from neighboring vertices to F.

Since

$$\mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F) = \min_{\boldsymbol{w} \in F} \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{w_t \neq y_t\right\} = \frac{1}{2} - \frac{1}{2n} \max_{\boldsymbol{w} \in F} \boldsymbol{w}^{\mathsf{T}} \boldsymbol{y}, \tag{21}$$

we may extend the function $\mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F)$ to any $F \subseteq [-1,1]^n$ by defining it as the right-hand side of (21). The extended $\mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F)$ is still stable in the sense of (6).

Suppose that calculating the distance $\mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F)$ is computationally expensive, due to the combinatorial nature of $F \subset \{\pm 1\}^n$. Let $F' \subseteq [-1,1]^n$ be a set containing F, and suppose that $\mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F')$ is easier to compute. The following observation is immediate:

Observation 1. Let $F \subseteq F' \subseteq [-1,1]^n$. Algorithms (16) and (18) with

$$\phi'(\boldsymbol{y}) = \mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F') + \mathscr{R}(F'),$$

enjoy a mistake bound

$$\mathbb{E}\left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] \leq \min_{\boldsymbol{w} \in F}\left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\left\{w_t \neq y_t\right\}\right] + \alpha \times \mathscr{R}(F) \tag{22}$$

for $\alpha \geq \mathscr{R}(F')/\mathscr{R}(F)$.

Proof. By construction, ϕ' is achievable, and

$$\mu_{\mathcal{A}}(\boldsymbol{y}) \leq \phi'(\boldsymbol{y}) \leq \mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F) + \mathscr{R}(F') \leq \mathrm{d}_{\mathrm{H}}(\boldsymbol{y}, F) + \alpha \times \mathscr{R}(F). \tag{23}$$

$\square$

By relaxing the set F to a larger set F', we may gain in computation while suffering a multiplicative factor α in the Rademacher complexity. Crucially, this factor *does not multiply the Hamming distance* but only the term $\mathscr{R}(F)$. The latter is typically of lower order and diminishing with n. We may summarize the observation as

$$\begin{pmatrix}\text{online}\\ \text{mistakes}\end{pmatrix} \le \begin{pmatrix}\text{offline combinatorial}\\ \text{benchmark}\end{pmatrix} + \alpha \times \begin{pmatrix}\text{additive } o(1)\\ \text{error}\end{pmatrix}.$$

There is a reason we belabor this simple observation. In the literature on online learning, it has been noted that one may guarantee

$$\begin{pmatrix}\text{online}\\ \text{mistakes}\end{pmatrix} \le \alpha \times \begin{pmatrix}\text{offline combinatorial}\\ \text{benchmark}\end{pmatrix} + \begin{pmatrix}\text{additive } o(1)\\ \text{error}\end{pmatrix}.$$

when one has a multiplicative approximation algorithm for the benchmark. The performance of the algorithm in this case is compared to

$$\alpha \times \min_{\boldsymbol{w}\in F}\left[\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\left\{w_t \neq y_t\right\}\right].$$

However, the bound easily becomes vacuous (say, the error rate of the offline benchmark is 5% and the multiplicative factor is a constant or logarithmic in n). The version where α only enters the remainder term seems much more attractive.

The key to obtaining (22) is the **improper** nature of the prediction methods (16) or (18): the prediction $\widehat{y}_t$ need not be in any way consistent with any of the models in F. Informally:

> Using improper learning methods, it may be possible to predict as well as a combinatorial benchmark (plus a lower-order term) even when computing this very benchmark given all the data is NP-hard.

To make the statement more meaningful, we will show that α can be upper bounded—for some interesting examples—in a way that does not render the mistake guarantee vacuous. We start with a simple example in Section 6.1, and then present a more complex machinery based on Constraint Satisfaction in Section 7.

6.1 Example: node classification

Consider the example in Section 3.1, and suppose, additionally, that the undirected graph $G = (V, E, W)$ has weights on edges. The weight on edge (i, j) is denoted by $W_{i,j}$, and $W_{i,j} \equiv 0$ when (i, j) is not an edge. Positive and negative edges may model friend/foe or trust/distrust networks.

We define F_κ as in (12), with the understanding that L is now the weighted graph Laplacian: $L = D - W$ with D the diagonal matrix, $D_{i,i} = \sum_{j=1}^{n}|W_{i,j}|$. As before, set $\phi(\boldsymbol{y}) = \mathrm{d_H}(\boldsymbol{y}, F_\kappa) + C_{n,F_\kappa}$. Why would evaluation of $\mathrm{d_H}(\cdot, F_\kappa)$ be computationally hard? First, if we can evaluate ϕ for any κ, we can also find the value

$$\min_{\boldsymbol{w}\in\{\pm1\}^n} \boldsymbol{w}^{\mathsf{T}} L \boldsymbol{w}. \tag{24}$$

However, if all the edge weights are -1, then (24) becomes

$$\min_{\boldsymbol{w}\in\{\pm 1\}^n} \sum_{(i,j)\in E} 1 + w_i w_j, \tag{25}$$

which may be recognized as the value of MinUnCut, an NP-hard problem in general. Hence, we cannot hope to evaluate the Hamming distance to F_κ exactly. Our first impulse is to approximate the value in (24); in the case of MinUnCut this can be done with a multiplicative factor of $O(\sqrt{\log n})$. However, it is not clear how to turn such a multiplicative approximation into a mistake bound with the factor 1 in front of the combinatorial benchmark. Yet, such a bound *is* possible, as we show next.

Following Observation 1, we set

$$F'_\kappa = \left\{\boldsymbol{w} \in [-1,1]^n \ : \ \boldsymbol{w}^\top L \boldsymbol{w} \le \kappa\right\} \tag{26}$$

and extend

$$\mathrm{d_H}(\boldsymbol{y}, F'_\kappa) = \frac{1}{2} - \frac{1}{2n} \max_{\boldsymbol{w}\in F'_\kappa} \boldsymbol{w}^\top \boldsymbol{y}, \tag{27}$$

as in (21). Note that evaluating $\mathrm{d_H}(\boldsymbol{y}, F'_\kappa)$ amounts to maximization of a linear function subject to a quadratic constraint $\boldsymbol{w}^\top L\boldsymbol{w} \le \kappa$ and a box constraint $\boldsymbol{w} \in [-1,1]^n$. This can be accomplished with standard optimization toolboxes.

It remains to estimate $\mathscr{R}(F'_\kappa)$. To this end, notice that

$$F'_\kappa \subset \left\{\boldsymbol{w} \in \mathbb{R}^n : \boldsymbol{w}^\top \boldsymbol{w} \le n, \boldsymbol{w}^\top L \boldsymbol{w} \le \kappa\right\} \subset \left\{\boldsymbol{w} \in \mathbb{R}^n : \boldsymbol{w}^\top M \boldsymbol{w} \le 1\right\} \tag{28}$$

with $M = \frac{1}{2n} I + \frac{1}{2\kappa} L$. Hence, we can upper bound

$$2n \cdot \mathscr{R}(F'_\kappa) = \mathbb{E}_{\boldsymbol{\varepsilon}} \max_{\boldsymbol{w}\in F'_\kappa} \boldsymbol{w}^\top \boldsymbol{\varepsilon} \le \mathbb{E}_{\boldsymbol{\varepsilon}} \max_{\boldsymbol{w}^\top M \boldsymbol{w} \le 1} \boldsymbol{w}^\top \boldsymbol{\varepsilon} \le \mathbb{E}_{\boldsymbol{\varepsilon}} \sqrt{\boldsymbol{\varepsilon}^\top M^{-1} \boldsymbol{\varepsilon}} \le \sqrt{\sum_{j=1}^n \lambda_j(M)^{-1}}, \tag{29}$$

where $\lambda_j(M)$ is the jth eigenvalue of M. The upper bound in (29) depends on the spectrum of the underlying network's Laplacian with an added regularization term $\frac{1}{2n} I$. It is an interesting research direction to find tighter upper bounds, especially when the social network evolves according to a random process.

To summarize, we relaxed F_κ to a larger set F'_κ, for which computation can be performed with an off-the-shelf optimization toolbox. Further, we derived a (rather crude) upper bound on the Rademacher averages of this set. In our calculations, however, we did not obtain an upper bound on the multiplicative gap α between the original Rademacher averages $\mathscr{R}(F_\kappa)$ and the larger value $\mathscr{R}(F'_\kappa)$. Hence, the price we paid for efficiently computable solutions remains unknown. In the next section, we present a generic way to glean this payment from known approximations to integer programs.

7 Computational hierarchies

Unlike the rest of the tutorial, this section is not self-contained. Our aim is to sketch the technique in [RS15], hiding the details under the rug. We also refer the reader to the literature on approximation algorithms based on semidefinite and linear programming (see e.g. [GM12] and references therein).

7.1 Relaxing the optimization problem

Consider the set

$$F_\kappa = \{\boldsymbol{w} \in \{\pm 1\}^n \ : \ \mathcal{C}(\boldsymbol{w}) \le \kappa\} \tag{30}$$

for some $\mathcal{C} : \{\pm 1\}^n \to \mathbb{R}$, which we call a *constraint*. The definitions (12) and (14) in terms of graph Laplacian and weighted graph Laplacian are two examples of such a definition. A more general example is Constraint Satisfaction: let $\mathscr{C}$ be a collection of functions $\mathfrak{z} : \{\pm 1\}^n \to \mathbb{R}$ and set

$$\mathcal{C}(\boldsymbol{w}) = \sum_{\mathfrak{z} \in \mathscr{C}} \mathfrak{z}(\boldsymbol{w}).$$

Recall that computing a prediction amounts to evaluating the weighted Hamming distance from $\boldsymbol{y} \in \{\pm 1\}^n$ to F_κ, which—in view of (21)—is equivalent to finding the value

$$\mathsf{OPT}_1(\kappa, \boldsymbol{y}) \ \triangleq \ \max_{\boldsymbol{w} \in F_\kappa, \ \boldsymbol{w}^\top \boldsymbol{y} \ge \beta} \ \beta \tag{31}$$

and then setting

$$\phi(\boldsymbol{y}) = \frac{1}{2} - \frac{1}{2n}\mathsf{OPT}_1(\kappa, \boldsymbol{y}) + C_{n,F_\kappa}. \tag{32}$$

Observation 1 suggests that if OPT_1 cannot be easily computed for the set F_κ, we should aim to find a larger set F'_κ for which this optimization is easier. A twist here is that we will not write down the definition of F'_κ explicitly (although it can be understood as a projection of a certain higher-dimensional object). Instead, let us replace $\mathsf{OPT}_1(\kappa, \boldsymbol{y})$ in (32) by a value $\mathsf{SDP}_1(\kappa, \boldsymbol{y})$ of some other optimization problem (to be specified in a bit) and set

$$\phi'(\boldsymbol{y}) = \frac{1}{2} - \frac{1}{2n}\mathsf{SDP}_1(\kappa, \boldsymbol{y}) + C_{n,F'_\kappa}. \tag{33}$$

As before, the condition $\mathbb{E}\phi'(\boldsymbol{\varepsilon}) \ge 1/2$ for achievability of this function implies that the smallest value of the constant is

$$C_{n,F'_\kappa} = \frac{1}{2n}\mathbb{E}\left[\mathsf{SDP}_1(\kappa, \boldsymbol{\varepsilon})\right]. \tag{34}$$

Before defining SDP_1, let us state a version of Observation 1:

Observation 2. If for any $\boldsymbol{y} \in \{\pm 1\}^n$ it holds that

$$\mathsf{SDP}_1(\kappa, \boldsymbol{y}) \le \mathsf{OPT}_1(\alpha \cdot \kappa, \boldsymbol{y}) \tag{35}$$

for some $\alpha \in \mathbb{R}$, then using the algorithm (16) or (18) with ϕ' in (33), we guarantee a regret bound of

$$\mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \ne y_t\right\}\right] \le \min_{\boldsymbol{w} \in F_\kappa}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{w_t \ne y_t\right\}\right] + \mathscr{R}(F_{\alpha \cdot \kappa}) \tag{36}$$

for any sequence $\boldsymbol{y}$.

Proof. Immediate from (34) and (35), and the fact that F'_κ defined by SDP_1 contains F_κ. □

We now define SDP_1, along with two more auxiliary optimization problems, and prove (35).

7.2 Setting up auxiliary optimization problems

OPT_1 is an optimization of a linear objective over vertices of the hypercube, under the restriction $\mathcal{C}(\mathbf{x}) \le \kappa$. In the literature, much effort has been devoted to analyzing a dual problem: minimization of $\mathcal{C}(\mathbf{x})$, possibly subject to a linear constraint. Our plan of attack is to define dual auxiliary optimization formulations, then use "integrality gap" for these problems, and pass back to the primal objective OPT_1 in order to prove (35).

Let us define the set of probability distributions on those vertices of the hypercube that yield the value of at least β for the linear objective:

$$\mathcal{D}(\beta, \boldsymbol{y}) \triangleq \Delta(\{\boldsymbol{w} \in \{\pm 1\}^n,\ \boldsymbol{w}^\top \boldsymbol{y} \ge \beta\}). \tag{37}$$

Define the optimization problem

$$\mathsf{OPT}_2(\beta, \boldsymbol{y}) \triangleq \min_{\boldsymbol{w} \in \{\pm 1\}^n,\ \boldsymbol{w}^\top \boldsymbol{y} \ge \beta} \mathcal{C}(\boldsymbol{w}) = \min_{p \in \mathcal{D}(\beta, \boldsymbol{y})} \mathbb{E}_{\boldsymbol{z} \sim p} \mathcal{C}(\boldsymbol{z}), \tag{38}$$

the minimum constraint value achievable on the vertices of the hypercube, given that the linear objective value is at least β. The second equality in (38) holds true because the minimum of a linear (in p) objective is attained at a singleton, a vertex of the hypercube.

Both (31) and (38) are combinatorial optimization problems, which may be computationally intractable. A common approach to approximating these hard problems is to pass from distributions to *pseudo-distributions*. Roughly speaking, a pseudo-distribution at "level r" only behaves like a distribution for tuples of variables of size up to r. Associated to a pseudo-distribution is a notion of a pseudo-expectation, denoted by $\widehat{\mathbb{E}}$. We refer to [Bar14, CT12, Rot13] for details.

Let $\widehat{\mathcal{D}}(\beta, \boldsymbol{y})$ be the set of suitably defined pseudo-distributions with the property

$$\widehat{\mathcal{D}}(\beta, \boldsymbol{y}) \subseteq \widehat{\mathcal{D}}(\beta', \boldsymbol{y})$$

whenever $\beta' \le \beta$ (see [RS15] for the precise definitions of these sets in terms of semidefinite programs). Define a relaxation of (38) as

$$\mathsf{SDP}_2(\beta, \boldsymbol{y}) \triangleq \min_{\widehat{p} \in \widehat{\mathcal{D}}(\beta, \boldsymbol{y})} \widehat{\mathbb{E}}_{\boldsymbol{z} \sim \widehat{p}} \mathcal{C}(\boldsymbol{z}) \tag{39}$$

and let

$$\mathsf{SDP}_1(\kappa, \boldsymbol{y}) \triangleq \max_{\exists \widehat{p} \in \widehat{\mathcal{D}}(\beta, \boldsymbol{y}) \text{ s.t. } \widehat{\mathbb{E}}_{\boldsymbol{z} \sim \widehat{p}} \mathcal{C}(\boldsymbol{z}) \le \kappa} \beta\ . \tag{40}$$

We write "SDP" here because relaxations we have in mind arise from semidefinite relaxations, but the arguments below are generic and hold for other approximations of combinatorial optimization problems.

The *integrality gap* of the dual formulation is

$$\alpha(\boldsymbol{y}) \triangleq \max_{\beta} \frac{\mathsf{OPT}_2(\beta, \boldsymbol{y})}{\mathsf{SDP}_2(\beta, \boldsymbol{y})}, \tag{41}$$

with $\alpha \triangleq \max_{\boldsymbol{y}} \alpha(\boldsymbol{y})$. We emphasize that we define the gap for the dual problems. Next, we show that the gap appears when relating SDP_1 to OPT_1.

Lemma 4. For any $\boldsymbol{y} \in \{\pm 1\}^n$,

$$\mathsf{SDP}_1(\kappa, \boldsymbol{y}) \le \mathsf{OPT}_1(\alpha(\boldsymbol{y}) \cdot \kappa, \boldsymbol{y}). \tag{42}$$

Proof of Lemma 4. Let us fix $\boldsymbol{y}$ and for brevity omit it from OPT and SDP definitions. First, it holds that

$$\mathsf{SDP}_2(\mathsf{SDP}_1(\kappa)) \le \kappa. \tag{43}$$

Indeed, $\mathsf{SDP}_1(\kappa)$ is the value of β that guarantees existence of some pseudo-distribution $\widehat{p}$ in $\widehat{\mathcal{D}}(\beta, \boldsymbol{y})$ that respects the constraint of κ on average. Hence, for this value of β, the minimum in $\mathsf{SDP}_2(\beta)$ will include this pseudo-distribution, and, hence, the value of the minimum is at most κ. By the same token,

$$\mathsf{OPT}_1(\mathsf{OPT}_2(\beta)) \ge \beta. \tag{44}$$

Indeed, $\mathsf{OPT}_2(\beta)$ is the value of $\mathcal{C}(\boldsymbol{w})$ achieved by some $\boldsymbol{w} \in \{\pm 1\}^n$ satisfying $\boldsymbol{w}^\intercal \boldsymbol{y} \ge \beta$. The maximization in OPT_1 includes this $\boldsymbol{w}$ by the definition of F_κ, and, hence, the maximum is larger than β.

Next, by the definition of the integrality gap and (43),

$$\mathsf{OPT}_2(\mathsf{SDP}_1(\kappa)) \le \alpha \cdot \mathsf{SDP}_2(\mathsf{SDP}_1(\kappa)) \le \alpha \cdot \kappa. \tag{45}$$

By (44) and (45),

$$\mathsf{SDP}_1(\kappa) \le \mathsf{OPT}_1(\mathsf{OPT}_2(\mathsf{SDP}_1(\kappa))) \le \mathsf{OPT}_1(\alpha \cdot \kappa) \tag{46}$$

because the value of $\mathsf{OPT}_1(\kappa)$ is nondecreasing in κ. □

It remains to provide concrete examples where $\alpha(\boldsymbol{y})$ is bounded in a non-trivial manner.

7.3 Back to node classification

Recall the node classification example discussed in Section 6.1. In view of Lemma 4, to conclude the mistake bound (36) we only need to get an estimate on α. For the case $\mathcal{C}(\boldsymbol{w}) = \boldsymbol{w}^\intercal L \boldsymbol{w}$, the integrality gap defined in (41) is the ratio of an integer quadratic program (IQP) subject to linear constraint (38) and its relaxed version. We turn to [GS13, Theorem 6.1], which tells us that within $O(r)$ levels of Lasserre hierarchy one can solve the IQP subject to linear constraints with the gap of at most

$$O(\max\{\lambda_r^{-1}, 1\})$$

where λ_r is the r-th eigenvalue of the normalized graph Laplacian. One can verify that $\mathscr{R}(F_\kappa)$ grows as $\sqrt{\kappa}$, and thus we essentially pay an extra factor of $O(\max\{\lambda_r^{-1/2}, 1\})$ for having a polynomial-time algorithm, with the computational complexity of $O(n^r)$.

There are several points worth emphasizing:

- As another manifestation of improper learning, the prediction algorithm does not need to "round the solution." The integrality gap only appears in the mistake analysis. This means that any improvement in the gap analysis of semidefinite or other relaxations immediately translates to a tighter mistake bound for the same prediction algorithm.
- It is the expected gap (with respect to a random direction $\boldsymbol{\varepsilon}$) that enters the bound, a quantity that may be smaller than the worst-case gap. It would be interesting to quantify this gain.

As an alternative to definition (30), one may combine the linear objective and the constraint in a single "penalized" form. Such an approach allows one to use Metric Labeling approximation algorithms [KT02], and we refer to [RS15] for more details.

8 Incorporating covariates

In most applications of online prediction, some additional side information is available to the decision-maker before she makes the prediction. Consider the following generalization of the problem introduced in Section 1. At time $t = 1, \dots, n$, the forecaster observes side information $x_t \in \mathcal{X}$, makes a forecast $\widehat{y}_t \in \{\pm 1\}$ (based on the history $y_1, \dots, y_{t-1}$ and $x_1, \dots, x_t$), and then the value y_t is observed.

Let ϕ be a function of two sequences: $\phi : \mathcal{X} \times \{\pm 1\}^n \to \mathbb{R}$. The function ϕ is stable if

$$|\phi(\boldsymbol{x}; y_{1:t-1}, +1, y_{t+1:n}) - \phi(\boldsymbol{x}; y_{1:t-1}, -1, y_{t+1:n})| \le 1/n, \tag{47}$$

where $\boldsymbol{x} = (x_1, \dots, x_n)$. We will prove the following generalization of Lemma 1.

Lemma 5. Let $\phi : \mathcal{X} \times \{\pm 1\}^n \to \mathbb{R}$ be stable, and suppose that x_t's are i.i.d. Then there exists a prediction strategy such that

$$\forall \boldsymbol{y} \in \{\pm 1\}^n, \qquad \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{\widehat{y}_t \neq y_t\}\right] \le \mathbb{E}\phi(\boldsymbol{x}; \boldsymbol{y}) \tag{48}$$

if and only if

$$\mathbb{E}\phi(\boldsymbol{x}; \boldsymbol{\varepsilon}) \ge 1/2. \tag{49}$$

Above, the expectation on the left-hand side of (48) is over the randomization of the algorithm and the x's, while on the right-hand side the expectation is over the x's. In (49), the expectation is both over the x's and over the independent Rademacher random variables.

An attentive reader will notice that the guarantee (48) is not very interesting because x_t's are chosen independently while y_t's are fixed ahead of time. The issue would be resolved if y_t's could be chosen by Nature *after* seeing x_t. Let us call such a Nature *semi-adaptive*, and reserve the word *adaptive* for Nature that chooses y_t's based also on the full history of $\{(x_s, y_s, \widehat{y}_s)\}_{s=1}^{t-1}$ (including learner's predictions).

Lemma 6. Lemma 5 holds for an adaptive Nature. That is, (49) is equivalent to existence of a prediction algorithm (given in (51) below) with

$$\mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] \leq \mathbb{E}\phi(\boldsymbol{x};\boldsymbol{y}), \tag{50}$$

where each $y_t \in \{\pm1\}$ may be chosen arbitrarily by Nature based on the history $\{(x_s, y_s, \widehat{y}_s)\}_{s=1}^{t-1}$ and x_t.

Inspecting the proof of Lemma 6, we see that the optimal doubly-randomized strategy is to draw $x_{t+1:n}$ and $\varepsilon_{t+1:n}$ and then set the mean of the distribution to be

$$\widetilde{q}_t^*(\boldsymbol{x}, y_{1:t-1}, \varepsilon_{t+1:n}) = n\left[\phi(\boldsymbol{x}; y_{1:t-1}, -1, \varepsilon_{t+1:n}) - \phi(\boldsymbol{x}; y_{1:t-1}, +1, \varepsilon_{t+1:n})\right]. \tag{51}$$

Notice that the coordinates $x_{1:t}$ of $\boldsymbol{x}$ are the actual observations, while $x_{t+1:n}$ are hallucinated. If x's are i.i.d., these hypothetical observations are available, for instance, if one has access to a pool of unlabeled data. In fact, the statement of Lemma 6 holds verbatim for any stochastic process governing evolution of x's, as long as we can "roll out the future" according to this process.

Finally, we state one more extension of Cover's result, lifting any stochastic assumptions on the generation of the x's.

Lemma 7. Let $\phi : \mathcal{X}^n \times \{\pm1\}^n \to \mathbb{R}$ be stable, and assume that both x_t and y_t are chosen by Nature adversarially and adaptively. Then existence of a strategy with

$$\mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] \leq \mathbb{E}\phi(\boldsymbol{x};\boldsymbol{y})$$

is equivalent to

$$\forall \boldsymbol{z}, \quad \mathbb{E}_{\boldsymbol{\varepsilon}}\phi(\boldsymbol{z}_1, \boldsymbol{z}_2(\varepsilon_1), \ldots, \boldsymbol{z}_n(\varepsilon_{1:n-1}); \boldsymbol{\varepsilon}) \geq \frac{1}{2}, \tag{52}$$

where $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ is an $\mathcal{X}$-valued predictable process with respect to the dyadic filtration $\{\sigma(\varepsilon_1, \ldots, \varepsilon_t)\}_{t\geq0}$ generated by i.i.d. Rademacher $\varepsilon_1, \ldots, \varepsilon_n$.

In Section 1, we introduced a "canonical" way to define $\phi(\boldsymbol{y})$ for the case of no side information. The analogous canonical definition that takes side information into account is as follows. Let $\mathcal{F}$ be a class of functions $\mathcal{X} \to \{\pm1\}$. If $\mathcal{F}$ is chosen well, one of the functions in this class will explain, approximately, the relationship between x_t's and y_t's. It is then natural to take the projection

$$F|_{\boldsymbol{x}} = \{(f(x_1), \ldots, f(x_n)) : f \in \mathcal{F}\},$$

the set of vertices of the hypercube achieved by evaluating some $f \in \mathcal{F}$ on the given data. We may now define

$$\phi(\boldsymbol{x};\boldsymbol{y}) = \mathrm{d_H}(\boldsymbol{y}, F|_{\boldsymbol{x}}) + C_{n,F|_{\boldsymbol{x}}},$$

as before. The function ϕ defined in this way is indeed small if $\boldsymbol{y}$ is close to the values of some $f \in \mathcal{F}$ on the data.

It remains to give an expression for $C_{n,F|\boldsymbol{x}}$. For the i.i.d. side information case of Lemma 6, the condition $\mathbb{E}\phi(\boldsymbol{x};\boldsymbol{\varepsilon}) \geq 1/2$ means that the smallest value of $C_{n,F|\boldsymbol{x}}$ ensuring achievability is

$$\frac{1}{2n}\mathbb{E}_{\boldsymbol{\varepsilon},\boldsymbol{x}}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\varepsilon_t f(x_t)\right],$$

the Rademacher averages of $\mathcal{F}$. For the adversarial case of Lemma 7, condition (52) means that the smallest value of $C_{n,F|\boldsymbol{x}}$ is

$$\sup_{\boldsymbol{z}} \mathscr{R}^{\text{seq}}(\mathcal{F},\boldsymbol{z}),$$

where

$$\mathscr{R}^{\text{seq}}(\mathcal{F},\boldsymbol{z}) \triangleq \frac{1}{2n}\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\varepsilon_t f(\boldsymbol{z}_t(\varepsilon_{1:t-1}))\right]$$

is the *sequential* Rademacher complexity [RST15].

9 Discussion and Research Directions

The prediction results discussed in this tutorial hold for arbitrary sequences – even for those chosen adversarially and adaptively in response to forecaster's past predictions. Treating the prediction problem as a multi-stage game against Nature has been very fruitful, both for the theoretical analysis and for the algorithmic development. Even though we discuss maliciously chosen sequences, it is certainly not our aim to paint any prediction problem as adversarial. Rather, we view the "individual sequence" results as being *robust* and applicable in situations when modeling the underlying stochastic process is difficult. For instance, one may try to model the node prediction problem described in Section 3.1 probabilistically—e.g. as a Stochastic Block Model—but such a model is unlikely to be true in the real world. Of course, the ultimate test is how the two approaches perform on real data. In the node classification example, the methods discussed in this tutorial performed very well in our own experiments, often surpassing the performance of more classical machine learning methods. Perhaps it is worth emphasizing that the prediction algorithms developed here are very distinct from these classical methods, and, if anything, this tutorial serves the purpose of enlarging the algorithmic toolkit.

We presented some very basic ideas, only scratching the surface of what is possible. Among some of the most interesting (to us) and promising research directions are:

- Develop linear or sublinear time methods for solving prediction problems on large-scale graphs.
- Run more experiments on real-world data and explore the types of functions ϕ that lead to good prediction performance.
- Develop partial-information versions of the problem. Some initial steps for contextual bandits were taken in [RS16, SLKS16].
- Analyze the setting of constrained sequences. That is, develop methods when Nature is not fully adversarial, yet also not i.i.d.

- Develop efficient prediction methods that go beyond i.i.d. covariates.

For more additional questions or clarifications, please feel free to email us.

A Proofs

Proof of Lemma 2. Define functions $\mathbf{Rel}_t : [k]^t \to \mathbb{R}$ as

$$\mathbf{Rel}_n(y_1, \ldots, y_n) = -\phi(y_1, \ldots, y_n)$$

and

$$\mathbf{Rel}_{t-1}(y_1, \ldots, y_{t-1}) = \mathbb{E}_{y_t \sim \mathrm{Unif}[k]} \mathbf{Rel}_t(y_1, \ldots, y_t) + \frac{1}{n}\left(1 - \frac{1}{k}\right), \tag{53}$$

with $\mathbf{Rel}_0(\varnothing)$ being a constant. We desire to prove that there is an algorithm such that

$$\forall \boldsymbol{y} \in [k]^n, \qquad \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] - \phi(y_1, \ldots, y_n) = 0.$$

Consider the last time step n and write the above expression as

$$\mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n-1} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\} + \frac{1}{n}\mathbf{1}\left\{\widehat{y}_n \neq y_n\right\} + \mathbf{Rel}_n(y_1, \ldots, y_n)\right]. \tag{54}$$

Let $\mathbb{E}_{n-1}$ denote the conditional expectation given $\widehat{y}_1, \ldots, \widehat{y}_{n-1}$. We shall prove that there exists a randomized strategy for the last step such that for any $y_n \in [k]$,

$$\mathbb{E}_{n-1}\left[\frac{1}{n}\mathbf{1}\left\{\widehat{y}_n \neq y_n\right\}\right] + \mathbf{Rel}_n(y_1, \ldots, y_n) = \mathbf{Rel}_{n-1}(y_1, \ldots, y_{n-1}). \tag{55}$$

This last statement is translated as

$$\min_{q_n \in \Delta_k} \max_{y_n \in [k]} \left\{\mathbb{E}_{n-1}\left[\frac{1}{n}\mathbf{1}\left\{\widehat{y}_n \neq y_n\right\}\right] + \mathbf{Rel}_n(y_1, \ldots, y_n)\right\} = \mathbf{Rel}_{n-1}(y_1, \ldots, y_{n-1}). \tag{56}$$

Writing $\mathbf{1}\left\{\widehat{y}_n \neq y_n\right\} = 1 - \boldsymbol{e}_{\widehat{y}_n}^{\mathsf{T}} \boldsymbol{e}_{y_n}$, the left-hand side of (56) is

$$\frac{1}{n} \min_{q_n \in \Delta_k} \max_{y_n \in [k]} \left\{1 - q_n^{\mathsf{T}} \boldsymbol{e}_{y_n} + n\mathbf{Rel}_n(y_1, \ldots, y_n)\right\}. \tag{57}$$

The stability condition (15) means that we can choose q_n to *equalize* the choices of y_n. Let $\psi(1), \ldots, \psi(k)$ be the sorted values of

$$n\mathbf{Rel}_n(y_1, \ldots, y_{n-1}, 1), \ldots, n\mathbf{Rel}_n(y_1, \ldots, y_{n-1}, k),$$

in non-increasing order. In view of the stability condition,

$$\sum_{i=1}^{k} (\psi(i) - \psi(k)) \le 1.$$

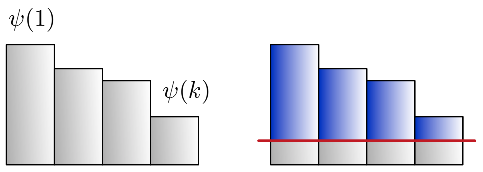

Figure 3: Under the stability condition, water-filling is optimal.

Hence, q_n can be chosen so that all $\psi(i) - q_n(i)$ have the same value (see Figure 3). One can check that this is the minimizing choice for q_n. Let q_n^* denote this optimal choice. The common value of $\psi(i) - q_n^*(i)$ can then be written as

$$\psi(k) - \frac{1}{k}\left(1 - \sum_{i=1}^{k}(\psi(i) - \psi(k))\right) = \frac{1}{k}\sum_{i=1}^{k}\psi(i) - \frac{1}{k}$$

and hence (57) is equal to

$$\frac{1}{n}\left(1 - \frac{1}{k}\right) + \frac{1}{k}\sum_{i=1}^{k}\mathbf{Rel}_n(y_1, \dots, y_{n-1}, i). \tag{58}$$

This value is precisely $\mathbf{Rel}_{n-1}(y_1, \dots, y_{n-1})$, as per Eq. (53), thus verifying (56). Repeating the argument for $t = n-1$ until $t = 0$, we find that

$$\mathbf{Rel}_0(\varnothing) = -\mathbb{E}\phi + \left(1 - \frac{1}{k}\right) = 0,$$

thus ensuring existence of an algorithm with (54) equal to zero. The other direction of the statement is proved by taking sequences $\boldsymbol{y}$ uniformly at random from $[k]^n$, concluding the proof.

When $k = 2$, the solution q_t^* takes on a simpler form

$$q_t^*(y_1, \dots, y_{t-1}) = n[\mathbb{E}\phi(y_1, \dots, y_{t-1}, -1, \varepsilon_{t+1}, \dots, \varepsilon_n) - \mathbb{E}\phi(y_1, \dots, y_{t-1}, +1, \varepsilon_{t+1}, \dots, \varepsilon_n)],$$

which is found by equating the two alternatives in (57). □

Proof of Lemma 6. As in the proof of Lemma 2, define functions $\mathbf{Rel}_t : (\mathcal{X} \times \{\pm 1\})^t \to \mathbb{R}$ as

$$\mathbf{Rel}_n(x_{1:n}; y_{1:n}) = -\phi(x_{1:n}; y_{1:n})$$

and

$$\mathbf{Rel}_{t-1}(x_{1:t-1}; y_{1:t-1}) = \mathbb{E}_{\varepsilon_t, x_t}\mathbf{Rel}_t(x_{1:t-1}, x_t; y_{1:t-1}, \varepsilon_t) + \frac{1}{2n}, \tag{59}$$

with $\mathbf{Rel}_0(\varnothing)$ being a constant. Having observed $x_{1:n-1}, y_{1:n-1}$ and x_n at the present time step, we solve

$$\min_{q_n}\max_{y_n}\left\{\mathbb{E}\left[\frac{1}{n}\mathbf{1}\{\widehat{y}_n \neq y_n\}\right] + \mathbf{Rel}_n(x_{1:n}, y_{1:n})\right\} \tag{60}$$

The same steps as in Lemma 2 (for binary prediction) lead to the solution

$$q_n^*(x_{1:n}, y_{1:n-1}) = n[\phi(x_{1:n}, y_{1:n-1}, -1) - \phi(x_{1:n}, y_{1:n-1}, +1)]. \tag{61}$$

We remark that q_n^* depends on x_n, as given by the protocol of the problem. Then (60) equals to

$$\mathbb{E}_{\varepsilon_n} \mathbf{Rel}_n(x_{1:n}; y_{1:n-1}, \varepsilon_n) + \frac{1}{2n} \tag{62}$$

We now take expectation over x_n on both sides:

$$\mathbb{E}_{x_n} \min_{q_n} \max_{y_n} \left\{ \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] + \mathbf{Rel}_n(x_{1:n}; y_{1:n}) \right\} = \mathbf{Rel}_{n-1}(x_{1:n-1}; y_{1:n-1}) \tag{63}$$

The argument continues back to $t = 0$, with

$$\mathbf{Rel}_t(x_{1:t}; y_{1:t}) = -\mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \phi(x_{1:n}; y_{1:t}, \varepsilon_{t+1:n}) + \frac{n-t}{2n} \tag{64}$$

and

$$q_t^*(x_{1:t}, y_{1:t-1}) = n\mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \left[\phi(x_{1:n}, y_{1:t-1}, -1, \varepsilon_{t+1:n}) - \phi(x_{1:n}, y_{1:t-1}, +1, \varepsilon_{t+1:n})\right]. \tag{65}$$

Finally,

$$\mathbb{E}\phi(x_{1:n}; \varepsilon_{1:n}) = \frac{1}{2}$$

means that $\mathbf{Rel}_0(\varnothing) = 0$. The algorithm in (65) is not implementable: it requires the knowledge of P_X. However, all we need is to be able to sample $x_{t+1:n} \sim P_X$ (or have access to unlabeled data), draw independent Rademacher $\varepsilon_{t+1:n}$, and define

$$\widetilde{q}_t^*(x_{1:n}, y_{1:t-1}, \varepsilon_{t+1:n}) = n\left[\phi(x_{1:n}, y_{1:t-1}, -1, \varepsilon_{t+1:n}) - \phi(x_{1:n}, y_{1:t-1}, +1, \varepsilon_{t+1:n})\right]. \tag{66}$$

The proof that this strategy yields the same expected mistake bound against adaptive Nature relies on the technique we term random playout. In this case the proof is not difficult. Consider (60) at step t and use the inductive definition of $\mathbf{Rel}_t$ in (64):

$$\min_{q_t} \max_{y_t} \left\{ \mathbb{E}\left[\frac{1}{n}\mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] - \mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \phi(x_{1:n}; y_{1:t}, \varepsilon_{t+1:n}) + \frac{n-t}{2n} \right\} \tag{67}$$

In the above minimum over q_t, let us choose the randomized strategy with the mean given by (66), thus passing to an upper bound of

$$\max_{y_t} \left\{ \mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \mathbb{E}_{\widehat{y}_t \sim \widetilde{q}_t^*} \left[\frac{1}{n}\mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] - \mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \phi(x_{1:n}; y_{1:t}, \varepsilon_{t+1:n}) \right\} + \frac{n-t}{2n} \tag{68}$$

which, in turn, is upper bounded via Jensen's inequality by

$$\mathbb{E}_{x_{t+1:n}, \varepsilon_{t+1:n}} \max_{y_t} \left\{ \mathbb{E}_{\widehat{y}_t \sim \widetilde{q}_t^*} \left[\frac{1}{n}\mathbf{1}\left\{\widehat{y}_t \neq y_t\right\}\right] - \phi(x_{1:n}; y_{1:t}, \varepsilon_{t+1:n}) \right\} + \frac{n-t}{2n}. \tag{69}$$

The choice of $\widetilde{q}_t^*$ makes the two possibilities for $y_t\{\pm 1\}$ identical in terms of their value (such a strategy is called an *equalizer*) and is optimal. With routine algebra, the above expression is equal to

$$\mathbb{E}_{x_{t+1:n},\varepsilon_{t+1:n}}\mathbb{E}_{\varepsilon_t}\phi(x_{1:n}; y_{1:t-1}, \varepsilon_t, \varepsilon_{t+1:n}) + \frac{n-t+1}{2n}. \tag{70}$$

Taking expectation with respect to x_t yields $\mathbf{Rel}_{t-1}(x_{1:t-1}; y_{1:t-1})$, completing the recursion step. Moreover, because of this value is optimal, so is the randomized strategy and, hence, the inequality in the above proof is an equality.

Regarding the required stability condition on ϕ, we see that it is simply that (65) is within the range $[-1, 1]$. In particular, it is implied by the assumed stability condition.

Finally, because of the order of expectation, minimum, and maximum in (63), the choice of y_t for Nature may depend on x_t (which is drawn from an unknown distribution), on q_t (but not on $\widehat{y}_t$), and on the history. This ensures that the prediction strategy works against an adaptive Nature.

□

References

[Bar14] B. Barak. Sum of squares upper bounds, lower bounds, and open questions, 2014. Lecture notes.

[Bla95] D. Blackwell. Minimax vs. bayes prediction. *Probability in the Engineering and Informational Sciences*, 9:pp 53–58, 1995.

[CBS11] N. Cesa-Bianchi and O. Shamir. Efficient online learning via randomized rounding. In *Advances in Neural Information Processing Systems*, pages 343–351, 2011.

[Cov65] T. Cover. Behaviour of sequential predictors of binary sequences. In *Proc. 4th Prague Conf. Inform. Theory, Statistical Decision Functions, Random Processes*, 1965.

[CT12] E. Chlamtac and M. Tulsiani. Convex relaxations and integrality gaps. In *Handbook on semidefinite, conic and polynomial optimization*, pages 139–169. Springer, 2012.

[GM12] B. Gärtner and J. Matousek. *Approximation algorithms and semidefinite programming.* Springer Science & Business Media, 2012.

[GS13] V. Guruswami and A. K. Sinop. Rounding Lasserre SDPs using column selection and spectrum-based approximation schemes for graph partitioning and Quadratic IPs. *arXiv preprint arXiv:1312.3024*, 2013.

[Hag56] D.W. Hagelbarger. Seer, a sequence extrapolating robot. *Electronic Computers, IRE Transactions on*, pages 1–7, 1956.

[KT02] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

[Rot13] T. Rothvoß. The lasserre hierarchy in approximation algorithms. *Lecture Notes for the MAPSP*, pages 1–25, 2013.

[RS15] A. Rakhlin and K. Sridharan. Hierarchies of relaxations for online prediction problems with evolving constraints. In *COLT*, 2015.

[RS16] A. Rakhlin and K. Sridharan. BISTRO: An efficient relaxation-based method for contextual bandits. In *International Conference on Machine Learning*, 2016.

[RSS12] A. Rakhlin, O. Shamir, and K. Sridharan. Relax and randomize: From value to algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2150–2158, 2012.

[RST15] A. Rakhlin, K. Sridharan, and A. Tewari. Sequential complexities and uniform martingale laws of large numbers. *Probability Theory and Related Fields*, 161(1-2):111–153, 2015.

[Sha53] C.E. Shannon. A mind-reading machine. *Bell Laboratories memorandum*, 1953.

[SLKS16] V. Syrgkanis, H. Luo, A. Krishnamurthy, and R. E. Schapire. Improved regret bounds for oracle-based adversarial contextual bandits. *CoRR*, abs/1606.00313, 2016.