FEASIBILITY OPTIMALITY OF PERIODWISE STATIC PRIORITY POLICIES FOR A QUALITY OF SERVICE MODEL IN WIRELESS NETWORKS AND CONVERGENCE ANALYSIS FOR AN ONLINE RECOMMENDATION SYSTEM

BY

ANH TRUONG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Advisers:

Assistant Professor Negar Kiyavash
Professor P. R. Kumar

ABSTRACT

In the first part of this thesis, we consider a proposed Quality of Service (QoS) model in which a set of clients require their own timely-throughput from an access point, with packet deadlines restricted to be in one period. It is known that two debt-based policies, including time-based debt and weighted delivery-based debt, are feasibility optimal in the sense that they can fulfill the requirements of all sets of feasible clients. We analyze why these policies are optimal by considering a class of periodwise static priority policies. We prove that this latter class of policies can achieve whatever a history-dependent policy can, i.e., it suffices to consider only this class of policies for such a scheduling problem. Our approach proceeds by investigating the submodularity of the complement of the idle time function. We thereby show that the set defined by the timely-throughput constraints is a polymatroid, from which the optimality within the class of periodwise static priority policies follows.

The second part of the thesis analyzes the convergence of an algorithm for the problem of learning with expert advice. At the present time, several web-based recommendation systems use votes from experts or other users to recommend objects to other customers. We apply the 'learning from expert advice' framework for this system, and propose a recommendation algorithm that uses a weighted update rule. Often, recommendation algorithms make assumptions that do not hold in practice, such as requiring a large number of good objects, presence of experts with the identical taste as the user receiving the recommendation, or experts who vote on all or a majority of objects. Our algorithm relaxes these assumptions by allowing an arbitrary proportion of bad objects as well as arbitrary tastes of experts. Moreover, it can deal with the issues that arise because of the existence of sleeping-experts, i.e., experts who are not available for voting at all rounds. A key attribute of our approach is to define the concept of the best expert on the basis of both availability and accuracy of experts. We then prove that the algorithm converges almost surely to the best expert(s) regardless of whether the predictions of experts are binary or continuous valued. Moreover, we derive an upper bound on loss of the proposed algorithm by comparing it to the loss of an appropriately defined 'current best' and show that the regret of our algorithm is logarithmic in the number of experts. Besides theoretical performance guarantees, we present simulation results that show the proposed algorithm outperforms Dsybil, the current state-of-the-art recommendation algorithm.

To my parents, for their love and support. To my brother and sister, for their encouragement.

ACKNOWLEDGMENTS

I would like to thank Professor P. R. Kumar and Professor Negar Kiyavash for their enthusiastic advising, support and discussion during the project. I also thank Professor Vivek Borkar for his helpful comments and suggestions on improving some of the proofs. This thesis could not have materialized without their support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

QoS is of increasing demand in current networks due to the emergence of many real-time applications such as video streaming or Voice over Internet Protocol (VoIP). In wireless networks, this requires even more attention due to the intrinsic unreliability of the channels. One of the problems of current interest is to find scheduling policies to serve a set of clients when there are multiple clients in use at the same time. A previous work [1] introduced a QoS model that takes into account both channel unreliabilities and deadline constraints. It was established there that two debt-based policies, namely, time-based debt and weight-based debt, can satisfy the requirement on the average number of packets delivered in each period, under the above constraints. These polices were proven to be feasibility optimal, i.e., they can achieve all long-term timely-throughputs that the more general class of history-dependent policies (defined below in Definition 5) can. In fact, these policies belong to a class of policies that we call *periodwise static priority policies* (see Definition 4). In this thesis, we study why such a class must necessarily contain an optimal policy. We answer this by showing that the set of feasible timely-throughput vectors is a polymatroid because of the submodularity of the complement of the unavoidable idle time function.

In the second part of this thesis, we turn to the problems posed by recommendation systems. In recent years, online learning algorithms have gained popularity through the widespread usage of online recommendation systems, such as Netflix, Digg, or Rotten Tomatoes. These systems solicit opinions from users on items such as movies or news articles. We refer to the users, who vote for items, as “experts,” and the items under consideration as “objects.” Usually, recommendation systems offer a satisfaction scale (e.g., a five-star rating scale in Rotten Tomatoes) from which an expert can choose to make a prediction about a particular object. After receiving the ratings of experts for an object, the system calculates its own recommendation score. For example, Rotten Tomatoes releases an average rating of experts voting for an object. In our formulation of the problem, after a consumer, say Alice, consumes the object, her feedback (outcome) is used to update the weights of the experts. The weight evolution is a crucial aspect of a recommendation system. Many weight updating rules have been proposed in the learning literature, and the performance of the algorithms is evaluated by proving bounds on their regret [2–5].

In the second part of this thesis, we address the problem of designing a provably-good recommendation system. Our contributions are as follows:

- We suggest a recommendation algorithm that uses a weighted average updating of experts' weights similar to that of [6]. We prove that this updating rule will converge to the best expert whenever there is one such expert. Alternately it will switch between the best experts, if multiple equally good best experts are present. Moreover, we derive an upper bound on the loss of the proposed algorithm. Unlike a leading recommendation algorithm, Dsybil [7], we do not require that a high fraction of objects in the voting pool be good in order to obtain performance guarantees.

- While in the setup of Dsybil [7], experts can only cast positive, discretely valued votes, we allow experts to rate an object with any continuous value in [0,1]. The majority of common learning algorithms assume that all experts are available at all rounds. Our algorithm allows experts to refrain from voting at some rounds as this is a more realistic assumption for online recommendation systems.

- While for the proof of convergence of our algorithm, we make some assumptions on the availability and prediction distribution of experts, these assumptions are relaxed in the simulation results. In fact, our simulations show that the proposed algorithm outperforms the current state-of-the-art algorithm, Dsybil, in more general settings than were assumed in the derivation of the convergence proofs.

The rest of this thesis is organized as follows: Chapter 2 describes some related work on scheduling in networks. Chapter 3 presents the QoS model in detail and some definitions, as well as previous results on the two optimal priority policies. Chapter 4 establishes that there indeed exists a periodwise static priority policy that is optimal. Chapter 5 begins the second part of the thesis by presenting some preliminary concepts and definitions on learning with expert advice. Chapter 6 summarizes some previous work, covering a variety of frameworks and settings for this kind of problem. We then describe our problem setup in Chapter 7, including the loss function and the updating rule. In Chapter 8, convergence results, based on an analysis of weight evolution, are derived. Subsequently, Chapter 9 presents an upper bound on the loss of the algorithm, and simulation results are given in Chapter 10. Finally, we summarize both parts in Chapter 11.

CHAPTER 2

RELATED WORK

The problem of scheduling wireless networks to support Quality of Service (QoS) has been studied by two approaches. In the first one, the objective is to minimize the mean delay. Dua and Bambos [8] introduced a ‘switch-over’ policy to minimize the expected cost of buffering or dropping packets, solvable by a dynamic programming framework. Shakkottai and Srikant [9] considered the optimality vis-à-vis the number of packets lost. They proposed a modified policy from Earliest Due Date (EDD) policy, namely Feasible Earliest Due Date (FEDD), and pointed out that it is optimal for the case of a deterministic arrival process. In another approach, the maximization of throughput is taken into account. Tassiulas and Ephremides [10] have proposed a maximal throughput policy for multiple queues served by only one server. The decision of choosing which queue to serve is based on the change of connectivity of queues with the server and queue lengths. They have shown that this policy minimizes delay if the queues are symmetric. Considering a model featuring multi-services, Whittle [11] has considered a problem of allocating activities under an index system, with different objective functions and constraints.

Hou, Borkar and Kumar [1] have proposed a framework for a QoS model in which a vector of timely-throughputs is required by a set of clients, where packets are counted towards the throughput only if they are delivered within a certain deadline, all under unreliable channels. They introduced two debt-based policies that are proven to be optimal in the sense that they satisfy all feasible requirements. It turns out that these policies belong to the set of static priority policies, but the authors do not address the broader question of why this class is optimal in terms of realizing any vector that is achievable in the general class of history-dependent policies. Yao [12] has examined the problem of optimization of a linear objective over a polymatroid. Motivated by the above-mentioned open problem, in this thesis, we exploit the polymatroid property of the set of feasible timely-throughputs and establish the optimality of the class of the static priority policies.

CHAPTER 3

MODEL OF QUALITY OF SERVICE

In this section, we describe the QoS model defined in [1], for which we will prove the optimality of a class of policies. The model is that of a wireless network containing a set of N clients and an access point. Each client has an application to be served by the access point. In a noise susceptible environment, it is natural to assume that every channel between clients and access point is unreliable, i.e., each transmitted packet is not always successful. Each client i has a probability of successful transmission, denoted by p_i, which can be different from that of other clients. We suppose that time slots are divided into periods, each of length τ, and each packet has a deadline of τ. At the beginning of each period, each client generates a packet. The access point then chooses a packet among them to serve. In each period, only after it succeeds in transmitting that packet, does the access point choose one of the other remaining packets to serve next. If at the end of the period, a packet has not been delivered, it is discarded. Besides the delay constraint, each client requires a delivery ratio q_i, for $1 \leq i \leq N$, defined as the long-term average of the number of packets delivered per period. For such unreliable channels, the access point seeks a strategy to fulfill all clients' requirements.

Definition 1. *A client set* $\{1, 2, ..., N\}$ *is feasible if its required delivery ratios are satisfied by some scheduling policy, i.e., the time average delivery ratio vector is at least* $[q_i]$*, with probability 1.*

Definition 2. *A scheduling policy is feasibility optimal if it can fulfill all feasible sets of clients.*

Definition 3. *A work-conserving (or non-idling) policy is a policy which attempts to deliver all packets without resting whenever there is a still undelivered packet.*

Without loss of generality, we consider only work-conserving policies in this thesis. However, even under a work-conserving policy, the access point is forced to be idle for a certain fraction of time, which happens after all clients have been served (i.e., all packets are delivered), and there is no more work in the system. We define the expected fraction of idle time for a set of clients S as follows:

$$I(S) := E\{\frac{1}{\tau}(\tau - \sum_{i \in S} \gamma_i)^+\},$$

where γ_i is Geom(p_i), a random variable representing the number of time slots needed by client i in order for its packet to be successfully transmitted.

Let us define the busy time $B(S) := \tau - I(S)$ as the complement of the idle time, and note that the total busy time of all clients cannot exceed the busy time of the system. Hou, Borkar and Kumar in [1] obtain the following necessary and sufficient condition for a set of clients to be feasible:

Theorem 1. *The necessary and sufficient condition for a feasible set is:*

$$\sum_{i \in S} \frac{q_i}{\tau p_i} \leq 1 - I(S) \text{ for all } S \subseteq \{1, 2, ..., N\}.$$

Note that the left-hand side above denotes the expected total fraction of time that the access point needs to devote to the set of clients S.

We now consider the class of debt-based policies. These policies calculate, at the beginning of each period k, the "debts" that the access point owes clients. For the case of time-based debt, the debt of client i is defined as the difference between the implied workload, $w_i = \frac{q_i}{\tau p_i}$, and the actual average number of time slots given to client i, before period k. For the case of weight-delivery-based debt, the debt of client i is the difference between the required delivery ratio q_i and the actual average number of packets delivered in each period, weighted by $\frac{1}{p_i}$. In debt-based policies, the set of clients is arranged (at the beginning of each period) in the descending order of debts to be served.

Theorem 2. *The above debt-based policies are feasibility optimal.*

It is worth noting that these debt-based policies are contained in the following class of *periodwise static priority policies.*

Definition 4. *A periodwise static priority policy is a policy which maintains the priority order of the clients throughout one period regardless of the transmission of packets in that period. When a packet is delivered, it is dropped from the priority order. However, the order is revised at the beginning of the next period, possibly based on the entire previous history of packet transmissions and deliveries.*

CHAPTER 4

OPTIMALITY OF PERIODWISE STATIC PRIORITY POLICIES

So far, the reason for feasibility optimality of the class of periodwise static priority policies is not well understood. By this we mean that a policy in this class can attain at least as much, componenentwise, as any throughput vector that is achievable in the more general class of history-dependent policies:

Definition 5. *A history-dependent policy is a policy which uses the information of the clients from the past to the current time, to decide which packet is to be served in any slot.*

We note that a history dependent policy can keep reevaluating actions during a period, and can, for example, change the order of clients within that period if some clients have had unsuccessful attempts.

Yao [12] has examined the problem of linear optimization over a polymatroid.

Definition 6. *([12]) The polytope*

$$\mathcal{P} = \left\{ x \geq 0, \middle| \sum_{i \in S} x_i \leq f(S), \forall S \subseteq \mathcal{X} \right\}$$

is a polymatroid if function f satisfies the following properties:

(a) $f(\phi) = 0$.

(b) f *is non-decreasing.*

(c) f *is submodular, i.e., if* $E, F \subseteq S$, *then* $f(E)+f(F) \geq f(E \cup F)+f(E \cap F)$.

Note that, if we reverse all the above inequalities, then we say that the function f is supermodular.

For the problem of optimizaton of a linear function over a polymatroid, there is a simply defined "priority" policy that is optimal.

Theorem 3. *([12]) x_π defined below is the optimal solution for the linear maximization problem over a polymatroid.*

$$\begin{aligned}
&x_{\pi_1} = f(\{\pi_1\}),\\
&x_{\pi_2} = f(\{\pi_1, \pi_2\}) - f(\{\pi_1\}),\\
&...\\
&x_{\pi_N} = f(\{\pi_1, \pi_2, ..., \pi_N\}) - f(\{\pi_1, \pi_2, ..., \pi_{N-1}\}).
\end{aligned}$$

Returning to the problem of supporting timely-throughput under reliability, in order to prove the optimality of periodwise static priority policies, it suffices to show that the convex set of expected throughputs that is achievable in a period is a polymatroid. The reason is that any long-term average of throughput vectors must also lie in this convex set. Therefore, our goal in this part of the thesis is to prove that the following set of vectors is indeed a polymatroid:

$$\mathcal{P} = \left\{ (q'_1, q'_2, ..., q'_N) \middle| \sum_{i \in S} x_i \leq f(S), \forall S \subseteq \{1, 2, \ldots, N\}, x_i = \frac{q'_i}{\tau p_i}, \text{and } x_i \geq 0, \forall i, \right\}.$$

From above, $f(S) := 1 - I(S)$.

We will show that this set is indeed a polymatroid by demonstrating a submodular property of the complement of the idle time function.

Theorem 4. *The function $f(S) := 1 - I(S)$ satisfies the following properties:*

(a) $f(\phi) = 0$.

(b) f *is non-decreasing.*

(c) f *is submodular, i.e., if* $E, F \subseteq S$, *then* $f(E) + f(F) \geq f(E \cup F) + f(E \cap F)$.

The first property is obvious for f. The second property follows from the fact that the idle time function $I(\cdot)$ is a non-increasing function. Thus, it remains only to check property (c) to establish the submodularity of f.

Proof. Since $f(S)$ is the complement of $I(S)$, it turns out that we need to prove the supermodularity of the expected idle time function, i.e.,

$$I(E \cup F) + I(E \cap F) \geq I(E) + I(F), \text{ for all } E, F \subseteq S. \tag{4.1}$$

We will prove this using induction. The base case is when the set of clients is a singleton: $S = \{i\}$. The only possible subsets of S are $\{i\}$ and ϕ, and the supermodularity is obvious.

Assuming that the supermodularity holds for any set of N clients, we need to prove it also holds for any set of $N+1$ clients. First, note that, as in the base case, the proof is trivial when any one of the two possible sets, namely E and F, is ϕ or the whole set. The proof also holds when $E \cup F \subseteq \{1, 2, ..., N\}$, by the induction assumption. Therefore, we only need to consider the cases when $E \cup F = \{1, 2, ..., N+1\}$.

1. Suppose $E \cap F = \phi$.
   For simplicity, assume that $E = \{1, 2, ..., k\}$ and $F = \{k+1, k+2, ..., N+1\}$. Suppose client i requires t_i slots in order to be successfully delivered, for $1 \leq i \leq N+1$. Note that t_i is a random variable.
   Denote the version of the right-hand side of (4.1), where instead of the expected idle time we take the random idle time, by simply RHS, and similarly, the analog of the left-hand side of (4.1) by LHS. Also denote $h(\{1, 2, ..., k\})$ as the idle time if the only clients are the set $\{1, 2, ..., k\}$.
   Suppose $\sum_{i=1}^{N+1} t_i \leq \tau$. Then

$$\begin{aligned} RHS &= h(\{1, 2, ..., k\}) + h(\{k+1, k+2, ..., N+1\}) \\ &= \tau - \sum_{i=1}^{k} t_i + \tau - \sum_{i=k+1}^{N+1} t_i \\ &= \tau - \sum_{i=1}^{N+1} t_i + \tau \\ &= h(\{1, 2, ..., N+1\}) + h(\phi) \\ &= LHS. \end{aligned}$$

Suppose $\sum_{i=1}^{N+1} t_i > \tau$ and $\sum_{i=1}^{k} t_i \leq \tau$

$$\begin{aligned} RHS &= h(\{1,2,...,k\}) + h(\{k+1,k+2,...,N+1\}) \\ &= \tau - \sum_{i=1}^{k} t_i + \Big[\tau - \sum_{i=k+1}^{N+1} t_i\Big]^+ \\ &\leq \tau - \sum_{i=1}^{k} t_i + \Big[\sum_{i=1}^{k} t_i\Big]^+ \\ &= \tau \\ &= h(\{1,2,...,N+1\}) + h(\phi) \\ &= LHS. \end{aligned}$$

Suppose $\sum_{i=1}^{N+1} t_i > \tau$ and $\sum_{i=1}^{k} t_i > \tau$

$$\begin{aligned} RHS &= h(\{1,2,...,k\}) + h(\{k+1,k+2,...,N+1\}) \\ &= 0 + I(\{k+1,k+2,...,N+1\}) \\ &\leq \tau \\ &= h(\{1,2,...,N+1\}) + h(\phi) \\ &= LHS. \end{aligned}$$

2. Suppose $E \cap F = \{1,2,...,M\}$, i.e., they have M clients in common, with $1 \leq M \leq N-1$. For simplicity, assume that $E = \{1,2,...,M,M+1,...,k\}$ and $F = \{1,2,...,M,k+1,k+2,...,N+1\}$, then $E \cap F = \{1,2,...,M\}$.

   As above, consider first the case $\sum_{i=1}^{N+1} t_i \leq \tau$, then

$$\begin{aligned} RHS &= h(\{1,2,...,M,M+1,...,k\}) \\ &+ h(\{1,2,...,M,k+1,k+2,...,N+1\}) \\ &= \tau - \sum_{i=1}^{k} t_i + \tau - \sum_{i\in\{1,2,...,M\}\cup\{k+1,k+2,...,N+1\}} t_i \\ &= \tau - \sum_{i=1}^{N+1} t_i + \tau - \sum_{i=1}^{M} t_i \\ &= h(\{1,2,...,N+1\}) + h(\{1,2,...,M\}) \\ &= LHS. \end{aligned}$$

Suppose $\sum_{i=1}^{N+1} t_i > \tau$ and $\sum_{i=1}^{k} t_i \le \tau$

$$\begin{aligned}
RHS &= h(\{1,2,...,M,M+1,...,k\}) \\
&+ h(\{1,2,...,M,k+1,k+2,...,N+1\}) \\
&= \tau - \sum_{i=1}^{k} t_i + \left[\tau - \sum_{i\in\{1,2,...,M\}\cup\{k+1,k+2,...,N+1\}} t_i\right]^+ \\
&\le \tau - \sum_{i=1}^{k} t_i + \left[\sum_{i=M+1}^{k} t_i\right]^+ \\
&= \tau - \sum_{i=1}^{M} t_i \\
&= h(\{1,2,...,N+1\}) + h(\{1,2,...,M\}) \\
&= LHS.
\end{aligned}$$

Suppose $\sum_{i=1}^{N+1} t_i > \tau$ and $\sum_{i=1}^{k} t_i > \tau$ and $\sum_{i=1}^{M} t_i \le \tau$

$$\begin{aligned}
RHS &= h(\{1,2,...,M,M+1,...,k\}) \\
&+ h(\{1,2,...,M,k+1,k+2,...,N+1\}) \\
&= \left[\tau - \sum_{i\in\{1,2,...,M\}\cup\{k+1,k+2,...,N+1\}} t_i\right]^+ \\
&\le \left[\tau - \sum_{i=1}^{M} t_i\right]^+ \\
&= \tau - \sum_{i=1}^{M} t_i \\
&= h(\{1,2,...,N+1\}) + h(\{1,2,...,M\}) \\
&= LHS.
\end{aligned}$$

Suppose $\sum_{i=1}^{N+1} t_i > \tau$ and $\sum_{i=1}^{k} t_i > \tau$ and $\sum_{i=1}^{M} t_i > \tau$

$$\begin{aligned}
RHS &= h(\{1,2,...,M,M+1,...,k\}) \\
&+ h(\{1,2,...,M,k+1,k+2,...,N+1\}) \\
&= 0 \\
&= LHS.
\end{aligned}$$

Note that since the above proof holds for random variables t_i, for $1 \le i \le N+1$, it also holds for the expected values, and so the proof of submodularity is complete.

$\square$

Next, we define the following class of policies:

Definition 7. *A randomized periodic static priority policy is a periodwise static priority policy which, at the beginning of each period, chooses randomly an order, according to a fixed distribution, for all clients and keeps that order for that period.*

Our main result is Theorem 5 which proves the existence of a randomized periodwise static priority policy.

Theorem 5. *If there exists a history-dependent policy which is feasibility optimal, then there exists a randomized periodic static priority policy, which is also feasibility optimal.*

Proof. Let $\mathscr{C}$ be the convex hull defined by the constraints describing the polytope $\mathcal{P}$. Consider an ordering, which could be any permutation over the N clients, $\pi = [\pi_1, \pi_2, ..., \pi_N]$. In this order, client π_1 is served first, and client π_n is served last. If the randomized periodwise static priority policy (here, we only consider work-conserving policies) picks this order, its expected busy times spent on the client sets are described by the following equalities:

$$\begin{aligned}
&x_{\pi_1} = f(\{\pi_1\}),\\
&x_{\pi_2} = f(\{\pi_1, \pi_2\}) - f(\{\pi_1\}),\\
&...\\
&x_{\pi_N} = f(\{\pi_1, \pi_2, ..., \pi_N\}) - f(\{\pi_1, \pi_2, ..., \pi_{N-1}\}).
\end{aligned}$$

Because the number of time slots available for clients is bounded by τ, $\mathscr{C}$ is a closed and bounded convex set. It follows that if every extreme point of the polytope is some periodwise static priority policy, then the randomized periodwise static priority policy class can realize any history-dependent policy by time sharing. We therefore only need to show that every extreme point of $\mathscr{C}$ is some static priority policy x_π. As noted in Theorem 3, it is shown in [12] that, for a polymatroid, every extreme point indeed corresponds to a static priority policy, proving the result. $\square$

CHAPTER 5

BACKGROUND ON LEARNING FROM EXPERTS

We now turn to problems of learning with experts, which arise, for example, in learning literature and have applications in current online prediction systems such as Netflix, where users receive recommendations on movies to rent based on votes from consumers. The overall system consists of experts and a (master) algorithm. The task of experts is to make predictions on the quality of objects, and the algorithm attempts to evaluate an object for a user based on the ratings of the experts.

For instance, in the case of movies, experts are users who vote for movies they have watched, and the items under consideration are naturally the movies. The goal of the algorithm is to give rating predictions for movies for each user based on the prediction of experts and the compatibility of their former predictions with user's taste.

Formally, the framework can be summarized as follows. Let us assume that the predictions lie in a set $\mathcal{X}$. These predictions can be binary or continuous valued. In each round t, each expert i provides his prediction x_t^i for a given object. Based on predictions of experts, the algorithm decides which value $\hat{y}_t \in \mathcal{X}$ to predict for that object. After consumption, the user reveals his rating y_t in the set $\mathcal{Y}$. This rating (called outcome) is used to calculate losses of experts and the algorithm. The performance of each expert is a function of his aggregate loss so far and is used to weigh his prediction in the future.

Loss of each predictor (the expert or the algorithm) is defined as a non-negative function $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ of the outcome and the prediction value. Moreover, loss function is usually chosen to be a convex function. Some popular loss function choices are:

(i) Absolute loss: $\ell(\hat{y}_t) = |\hat{y}_t - y_t|$.

(ii) Squared loss: $\ell(\hat{y}_t) = (\hat{y}_t - y_t)^2$.

(iii) Relative entropy loss: $\ell(\hat{y}_t) = (1 - y_t) \ln\left(\frac{1-y_t}{1-\hat{y}_t}\right) + y_t \ln\left(\frac{y_t}{\hat{y}_t}\right)$.

(iv) Hellinger loss: $\ell(\hat{y}_t) = \frac{1}{2}\left((\sqrt{1 - y_t} - \sqrt{1 - \hat{y}_t})^2 + (\sqrt{y_t} - \sqrt{\hat{y}_t})^2\right)$.

The prediction of the algorithm is based on the level of trust it has in the value of the prediction of each expert for a particular user. This trust value for an expert i is called his weight and is denoted by $p_t^i \in \mathcal{W}$. The larger the weight of an expert, the larger is his influence on the prediction of the algorithm. At each round after calculating loss, the algorithm updates weights for experts.

These steps are summarized as follows:

```
**for** $t = 1 \to ...$ **do**
    Experts provide predictions $x_t^i \in \mathcal{X}$.
    Algorithm computes its prediction $\hat{y}_t \in \mathcal{X}$ based on $x_t^i, p_t^i$, for all $i$.
    Outcome $y_t \in \mathcal{Y}$ is revealed.
    Loss update: Algorithm and experts update losses $\ell(\hat{y}_t), \ell(x_t^i)$, respectively.
    Weight update: Weights get updated according to $p_{t+1}^i = f(p_t^i, \ell(\hat{y}_t), \ell(x_t^i))$.
**end for**
```

Often the weight update rule takes either the multiplicative or additive form. In the former, the weight of an expert is multiplied by a factor smaller than one to reduce the trust in him if his prediction was off, i.e., he incurred a big loss, or the weight remains unchanged if experts had a reliable prediction. Thus, good experts obtain bigger weights than poor ones as time progresses. Similarly the algorithm can use an additive rule to update rules.

The goal of research in learning with expert advice is to find an algorithm, i.e., predictor, whose loss is not much greater than the loss of the "best expert," defined as the one whose loss is minimal over the set of experts for a time horizon of interest. Define "current regret" as the difference between the algorithm's loss and an expert's loss at round t, $r(x_t^i) := \ell(\hat{y}_t) - \ell(x_t^i)$ and "cumulative regret" as the accumulated difference up to time n:

$$R_n^i := \sum_{t=1}^{n} \left(\ell(\hat{y}_t) - \ell(x_t^i)\right). \tag{5.1}$$

Thus, the target is to guarantee that the regret with respect to the best expert remains as small as possible. Specifically a vanishing per round regret is desirable, where per round regret is the averaged cumulative regret over n rounds. A common approach for designing the predictor is the weighted majority decision rule established by Littlestone and Warmuth [13]. This algorithm uses as its prediction a value equal to the weighted average predictions of experts. A different strategy is to follow the best expert. In each round, the algorithm calculates the loss of each expert and selects the expert with minimum current loss (ties are arbitrarily broken). We now present a simple example to see how these strategies work.

In the movie rating system, assume that predictions and outcomes are binary, i.e., $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. Specifically, 1 represents a good rating and 0 represents a bad rating. Moreover, assume an absolute loss function $L_n^i := \sum_{t=1}^{n} |x_t^i - y_t|$ is in place. At the beginning, all experts are given equal weight. Subsequently, the weight of expert i at round n is calculated by the multiplicative rule $p_n^i = \beta^{L_n^i} p_{n-1}^i$, where $\beta < 1$ is the factor to punish experts with poor performance. Given predictions x_t^i and weights p_t^i of all experts, the weighted majority algorithm first calculates $\hat{y}_t$ as

$$\hat{y}_t = \frac{\sum_i p_t^i x_t^i}{\sum_i p_t^i}.$$

The algorithm then predicts 1 if $\hat{y}_t > 1/2$ or 0 otherwise. The special case where $\beta = 0$ is called the "halving" algorithm. At each round, experts with bad predictions have weights taken off. The algorithm needs to only follow the rest of the experts in the next rounds by forming an average (weighted majority), or follow an arbitrarily expert chosen among the good ones (follows the best expert). Either way, after some rounds, weights of all experts except the best expert (assumed to exist) will be eliminated, and the algorithm follows the best expert thereafter.

CHAPTER 6

RELATED WORK

In most of the existing literature on learning with experts, it is assumed that all experts are available at all rounds [4, 5, 13]. With this assumption, the first class of predictors is based on the weighted majority algorithm, introduced in the pioneering work of Littlestone and Warmuth [13]. This algorithm follows the group of experts whose sum of their weights exceeds that of the rest, and, if there is a bad prediction, it punishes all members of that group by scaling their weights with a factor smaller than one. This algorithm was later extended by Bianchi et al. [5] to a weighted average prediction rule, and the scaling factor for weight updates could be flexibly chosen from a wide range of values. Bianchi and Lugosi [14] proposed a framework for design and analysis of prediction algorithms using the concept of potential function. Their framework includes special cases of polynomially weighted average and exponentially weighted average predictors mentioned in the framework of Littlestone and Warmuth [13].

Another way to formulate the problem of learning from experts' advice is by use of competitive online learning in which the algorithm is compared to a class of prediction strategies. One such method is aggregated forecaster, introduced by Vovk [4]. This forecaster makes predictions so that the potential function of the accumulated regret is non-increasing, or equivalently, its loss is no more than an exponential potential function. A simpler version of such a predictor was presented by Kivinen and Warmuth [15]. A weighted average model is deployed in this version, and interestingly, it coincides with the aggregated forecast if the loss function is logarithmic. Yet another such method is a defensive forecaster also proposed by Vovk [16]. This forecaster (predictor) makes predictions so that a law of probability (e.g., law of large numbers) describing the agreement between the predictions and the outcomes is satisfied. A comparison between the aggregate forecaster and defensive forecaster was given by Chernov et al. [17] under two different settings.

The above results however do not directly apply to online recommendation systems where the assumption that all experts are always available does not hold true. Specifically, most of classic approaches, e.g., Littlestone and Warmuth [13], discount the weight of an expert any time he makes a mistake, and leave it unchanged only if he makes a correct prediction. Thus, a good expert attending most of the rounds may end up with a lower weight than a poor expert participating in just a few rounds.

To avoid this, Yu et al. [7] encourage participation of the experts in voting by increasing their weights when they provide good advice. Moreover, they reward experts who vote on objects with fewer votes, as opposed to safe-playing experts who only vote for already popular objects. Yu et al. [7] provide an upper bound on the expected loss of their algorithm. However, their analysis is carried out under some strong assumptions such as that a high percentage of objects in the recommendations system are good (Alice likes them), and a small number of good experts have voted on a large fraction of good objects. The requirement of having a high percentage of good objects is partially due to the fact that their suggested algorithm does not incorporate negative feedback in updating the weights of experts.

In the setting where experts are not necessarily available all the time, Freund et al. [6], Kleinberg et al. [18], and Blum and Mansour [2] have considered so-called specialized experts or sleeping experts. Freund et al. [6] presented an average update rule in which the weight of each available expert is updated according to the quality of his prediction and the weighted prediction of the algorithm. In effect, the algorithm finds a pseudoexpert (average expert) whose loss is equal to the average loss of available experts and follows his prediction. This algorithm is actually a kind of mixture forecaster corresponding to a distribution over the set of experts, and its goal is to find a distribution that minimizes the regret. Kleinberg et al. [18] assumed that in each round, all available experts have an unknown but fixed payoff distribution. The algorithm then chooses the best expert based on the ordered arrangement of the expected payoffs. They propose algorithms that achieve nearly optimal regret bounds, i.e., optimal up to a constant or a sublogarithmic factor. Blum and Mansour [2] proposed time selection functions and modification rules to convert an external regret (i.e., regret of the algorithm with respect to best expert) to an internal regret (i.e., regret of the algorithm choosing an expert when another expert was chosen). The time selection function [6] is indeed the indicator function with value equal to one when the expert under consideration is available. They also derived an upper bound on external regret, though this regret was only considered within a fraction of time depicted by a selection function.

CHAPTER 7

SETTING AND MODEL

We now present the mathematical model and our notation. The set of all experts is denoted by $E = \{1, 2, ..., N\}$. At each round t, the set of available experts is denoted by E_t, where $E_t \subseteq E$. We assume that E_t is stationary and that the probability distribution of available experts is symmetric, i.e., it is invariant under permutations. (Note that, in the simulations, we relax this assumption and consider different availabilities for experts.)

At the beginning, each expert is given equal weight, i.e., $p_0^i = p_0^j$ for all $i \neq j$. As described in Chapter 5, the weight of expert i at time t is denoted by $p_t^i \in [0, 1]$, and his prediction for a given object is $x_t^i \in [0, 1]$. Here, we assume that E_t and x_t are independent of each other, which is a reasonable assumption since the correctness of an expert has no relation to his availability.

At time t, knowing the predictions (i.e., opinions) of awake experts for an object, the algorithm calculates the weighted prediction for that object, $\hat{y}_t$, as follows

$$\hat{y}_t = \frac{\sum\limits_{i \in E_t} p_t^i x_t^i}{\sum\limits_{i \in E_t} p_t^i}. \tag{7.1}$$

Thereby, the system provides a rating for every object (rated by the experts).

Recommendations are user specific, and to serve different users, the system maintains different weights for the experts, based on the feedback of the corresponding user.

Upon consumption of an object, the user, say Alice, provides her feedback y_t to the system, indicating her opinion about the object. Throughout this paper, y_t is assumed to be binary 0 or 1, corresponding to a bad or good rating, respectively.

After obtaining Alice's feedback, the weights of experts are updated as follow:

$$p_{t+1}^i = \begin{cases} p_t^i \frac{x_t^i}{\hat{y}_t} & \text{if } i \in E_t, y_t = 1, \\ p_t^i \frac{1-x_t^i}{1-\hat{y}_t} & \text{if } i \in E_t, y_t = 0, \\ p_t^i & \text{if } i \notin E_t. \end{cases} \tag{7.2}$$

It is worth noting that in (7.2) only weights for available experts, i.e., the ones in E_t, are updated. For experts not in E_t, their weights remain unchanged.

Let us consider the case when we have a good object (according to Alice, of course). For such an object, a good prediction algorithm is supposed to recommend a value greater than a half. Therefore, the update rule of (7.2) increases the weight of an expert if he has a recommendation better than the average. A similar explanation holds for the case of bad objects.

In this thesis, we use the relative entropy log loss, that is, the natural logarithm of the difference between the outcome and the predicted value:

$$L_t = -I\{y_t = 1\} \ln \hat{y}_t - I\{y_t = 0\} \ln(1 - \hat{y}_t). \tag{7.3}$$

CHAPTER 8

CONVERGENCE OF THE ALGORITHM

In this chapter, we analyze the weight evolution and the convergence of the algorithm. We start with experts making binary predictions and then proceed to the continuous case.

Let us rewrite the update rule in (7.2) in matrix form. Let

$$p_t = \begin{bmatrix} p_t^1 \\ p_t^2 \\ \vdots \\ p_t^N \end{bmatrix}.$$

Then $p_{t+1} = D_t p_t$, where

$$D_t = \begin{bmatrix} d_t^1 & 0 & \dots & 0 \\ 0 & d_t^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_t^N \end{bmatrix}.$$

Since each expert updates his weight depending on his previous weight only, D_t is a diagonal matrix. Each diagonal element is given by (7.2), i.e.,

$$d_t^i = I\{i \notin E_t\} + I\{i \in E_t\}\Big[I\{y_t = 1\}\frac{x_t^i}{\hat{y}_t} + I\{y_t = 0\}\frac{1 - x_t^i}{1 - \hat{y}_t}\Big].$$

Define

$$q_t^i := I\{i \notin E_t\} + I\{i \in E_t\}\Big[I\{y_t = 1\}x_t^i + I\{y_t = 0\}(1 - x_t^i)\Big],$$

and the corresponding normalization factor

$$\bar{q}_t^i := I\{i \notin E_t\} + I\{i \in E_t\}\Big[I\{y_t = 1\}\hat{y}_t + I\{y_t = 0\}(1 - \hat{y}_t)\Big].$$

Note that $d_t^i = \frac{q_t^i}{\bar{q}_t^i}$. Hence, $D_t = Q_t \bar{Q}_t$, with

$$Q_t = \begin{bmatrix} q_t^1 & 0 & \dots & 0 \\ 0 & q_t^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_t^N \end{bmatrix},$$

and

$$\bar{Q}_t = \begin{bmatrix} \frac{1}{\bar{q}_t^1} & 0 & \dots & 0 \\ 0 & \frac{1}{\bar{q}_t^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\bar{q}_t^N} \end{bmatrix}.$$

Since Q_t and $\bar{Q}_t$ are diagonal matrices, by induction,

$$p_t = \Big(\prod_{m=0}^{t-1} Q_m \bar{Q}_m \Big) p_0.$$

Now, consider the ratio between the weights of experts i and j at time t,

$$\xi_t^{ij} := \frac{p_t^i}{p_t^j} = \frac{\Big(\prod_{m=0}^{t-1} q_m^i \frac{1}{\bar{q}_m^i} \Big) p_0^i}{\Big(\prod_{m=0}^{t-1} q_m^j \frac{1}{\bar{q}_m^j} \Big) p_0^j}.$$

We have,

$$\ln \xi_t^{ij} = \ln \frac{p_0^i}{p_0^j} + \sum_{m=0}^{t-1} (\ln q_m^i - \ln q_m^j) + \sum_{m=0}^{t-1} (\ln \bar{q}_m^j - \ln \bar{q}_m^i).$$

The following lemma is important for our main results.

Lemma 1. *Given the above settings and assumptions, if* $E(\ln q_m^i) > E(\ln q_m^j)$ *for* $i \neq j$*, then* $p_t^j \to 0 \quad a.s.$ *and* $p_t^i \to 1 \quad a.s.$

Proof. Taking the limit of $\frac{1}{t} \ln \xi_t^{ij}$ as $t \to \infty$, using the law of large numbers, with the assumptions that $\{q_m^i\}_1^\infty$ and $\{q_m^j\}_1^\infty$ are i.i.d., as are $\{\bar{q}_m^i\}_1^\infty$ and

$\{\bar{q}_m^j\}_1^\infty$, we obtain

$$\frac{1}{t}\ln \xi_t^{ij} \to E(\ln q_m^i) - E(\ln q_m^j) + E(\ln \bar{q}_m^j) - E(\ln \bar{q}_m^i).$$

However, since we have assumed that the distribution of available experts is symmetric, $\bar{q}_m^i$ and $\bar{q}_m^j$ have the same distribution. Hence, $E(\ln \bar{q}_m^i) = E(\ln \bar{q}_m^j)$. Therefore,

$$\frac{1}{t}\ln \xi_t^{ij} \to E(\ln q_m^i) - E(\ln q_m^j).$$

It follows that if $E(\ln q_m^i) > E(\ln q_m^j)$, then $\ln \xi_t^{ij} \to \infty$, i.e., $\frac{p_t^i}{p_t^j} \to \infty$. Note that from the update rule (7.2), $\sum_{i=1}^n p_t^i = 1$ for every t. Therefore,

$$p_t^j \to 0 \quad a.s. \text{ and } p_t^i \to 1 \quad a.s. \tag{8.1}$$

□

Now, we consider whether the above inequality, $E(\ln q_m^i) > E(\ln q_m^j)$, holds. Define the time proportion of occurrence of a set of experts A as follows:

$$\Gamma(A) := \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} I\{E_t = A\} \quad \forall A \subseteq \{1, 2, ..., N\}.$$

Note that Γ is a probability measure and that the above definition is well defined since E_t is stationary. Define the accuracy of expert i, μ^i, to be the probability of correct predictions, with respect to user Alice.

8.1 Binary predictions

We first analyze the algorithm where x_t^i is allowed to take only two values, ε and $1-\varepsilon$, depending on the values of outcome y_t. Here, ε is a small positive value to guarantee that $\ln \varepsilon$ does not diverge to infinity. Consider the case when expert i is correct, which happens with probability μ^i. If $y_t = 1$, then $x_t^i = 1-\varepsilon$, and $q_t^i = 1-\varepsilon$. Similarly, if $y_t = 0$, then $x_t^i = \varepsilon$, and $q_t^i = 1-\varepsilon$.

A similar derivation holds for the case when the expert is not correct. It follows that, if $i \in E_t$

$$q_m^i = \begin{cases} 1-\varepsilon & \text{w.p } \mu^i, \\ \varepsilon & \text{w.p } 1-\mu^i. \end{cases}$$

So, the expected value of $\ln q_m^i$ is given by

$$\begin{aligned} E(\ln q_m^i) &= \sum_{A:i\in A} \Gamma(A)\Big(\mu^i \ln(1-\varepsilon) + (1-\mu^i)\ln\varepsilon\Big) \\ &= \Big(\sum_{A:i\in A} \Gamma(A)\Big)\Big(\mu^i \ln(1-\varepsilon) + (1-\mu^i)\ln\varepsilon\Big). \end{aligned} \tag{8.2}$$

Let us define the availability of expert i as the proportion of time that expert i is available, which is the first factor of (8.2)). Also define the "index" of an expert i as the second factor in the right-hand side in (8.2), which is a non-decreasing function of his accuracy. The best expert is then defined as the expert whose index is highest, i.e., $E(\ln q_m^i) > E(\ln q_m^j)$ for all $j \neq i$. Our main results for the binary case is as follows.

Theorem 6. *In the binary setting, if there is only one best expert, then the algorithm converges to that expert. If there exists more than one best expert, then the algorithm alternates between those experts.*

Proof.

1. We begin with the first case when there is only one expert attaining the highest index. From Lemma 1, the weight of expert j converges to zero for all $j \neq i$, while the weight of expert i converges to 1. The algorithm then follows that best expert. Note that from (8.2), the performance of an expert depends on both his accuracy and availability. In our convergence analysis, it has been assumed that the distribution of available experts is symmetric and hence all experts are equally available. Therefore, the algorithm will follow the most accurate expert.[1]

2. For the second case, without loss of generality, assume that there exists more than one expert who achieves the maximum value in (8.2), i.e.,

[1]In the simulation results, we present a more interesting comparison of the performance of our algorithm where both availability and accuracy are taken into account. There, the index is redefined as the product of accuracy and availability of an expert. The "best" expert must achieve the highest product. Thus, this index encourages experts to vote for objects, since it prefers experts who both participate in voting as well as provide correct recommendations.

there exist experts i, j such that $E(\ln q_m^i) = E(\ln q_m^j) > E(\ln q_m^k)$ for all $k \neq i, j$. The proof is based on the law of the iterated logarithms. We recall the law of iterated logarithm (LIL): Given X_n i.i.d., variance σ^2, let

$$S_n := \sum_{m=1}^{n} X_m.$$

Then,

$$\limsup_{n\to\infty} \frac{S_n}{\sqrt{n \log\log n}} = \sqrt{2}\sigma \quad a.s.$$

Also,

$$\liminf_{n\to\infty} \frac{S_n}{\sqrt{n \log\log n}} = -\sqrt{2}\sigma \quad a.s.$$

Thus, the sum S_n will change signs infinitely often. Applying LIL for $X_m = \ln q_m^i$, along with the fact that $Prob\{p_t^i = p_t^j\} = 0$, one can see that $(\ln q_m^i - \ln q_m^j) > 0$ or $(\ln q_m^i - \ln q_m^j) < 0$ alternately. Therefore the scheme will switch between the two experts infinitely often. However, S_n can change only by a bounded amount in each step, from which it follows that the switchings become rarer and rarer, i.e., the algorithm stays with the same expert for longer and longer times.

□

8.2 Continuous predictions

When x_t^i can take continuous values in $[0, 1]$, μ_i needs to be analyzed to a finer degree. For a given tolerance a, define the corresponding accuracy $\mu_i(a)$ of an expert i as the percentage of time that his recommendations lie within a distance a from Alice's feedback, i.e.,

$$|x_t^i - y_t| < a.$$

Suppose first that expert i is correct. If $y_t = 1$, then x_t^i is not far from 1 by a distance greater than a, i.e., $x_t^i > 1 - a$, and x_t^i is assumed to be uniformly distributed in $[1 - a, 1]$. It follows that $q_t^i = x_t^i$ and q_t^i has the same distribution in this interval. Similarly, if $y_t = 0$, then $x_t^i < a$, and x_t^i is assumed to be uniformly distributed in $[0, a]$. Therefore, $q_t^i = 1 - x_t^i$ and q_t^i has uniform distribution in $[1 - a, 1]$.

The similar derivation holds for the case when expert i is not correct. Specifically, if $y_t = 1$, then $x_t^i < 1 - a$, $q_t^i = x_t^i$ and x_t^i is assumed to be uniformly distributed in $[1 - a, 1]$. If $y_t = 0$, then $x_t^i > a$, $q_t^i = 1 - x_t^i$ and x_t^i is assumed to be uniformly distributed in $[0, a]$.

So far, our assumption on the prediction of expert i, x_t^i, is that it is piecewise uniformly distributed[2] as depicted in Figure 8.1.

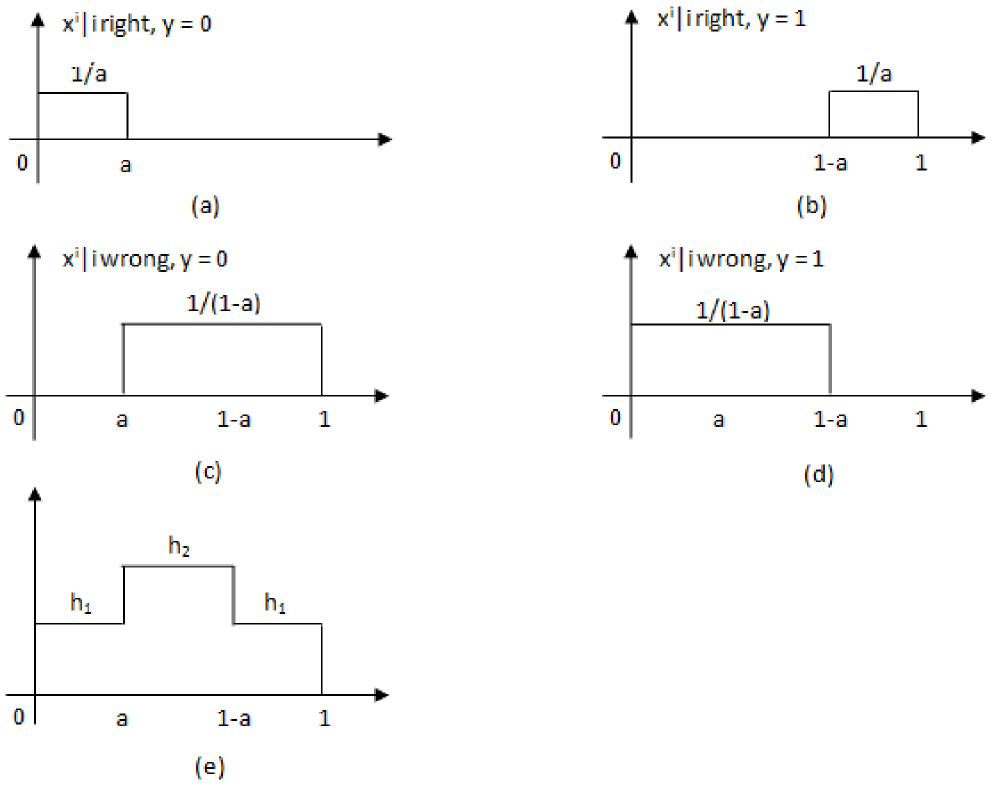

Figure 8.1: Distribution of predictions of expert i. (a-d) correspond to the conditional distributions given that expert i is right or wrong and the value of the outcome is 0 or 1; (e) corresponds to the distribution of expert i.

Note that from this figure, the distributions of experts' predictions are different from each other due to the values of h_1, h_2.

$$h_1 = \frac{\mu^i}{2a} + \frac{1 - \mu^i}{2(1 - a)},$$

$$h_2 = \frac{1 - \mu^i}{1 - a}.$$

[2]In the simulation, we also investigated some other distributions such as Poisson or exponential, with random variables taking values only on [0,1].

For experts whose predictions are usually good, the corresponding values of h_2 are small and, consequently, those of h_1 are large.

From the above, the probability distribution function of q_t^i is derived as

$$f_{q^i}(x) = \begin{cases} \frac{1-\mu^i(a)}{1-a} & \text{if } x \in [0, 1-a], \\ \frac{\mu^i(a)}{a} & \text{if } x \in [1-a, 1]. \end{cases}$$

The index of expert i is given by

$$E(\ln q_t^i) \quad = \Big(\sum_{A: i \in A} \Gamma(A) \Big) \Big(\tfrac{\mu^i(a)}{a} \int_{1-a}^{1} \ln x dx + \tfrac{1-\mu^i(a)}{1-a} \int_0^{1-a} \ln x dx \Big). \quad (8.3)$$

The same results as in Theorem 6 can be applied for this case with continuous values. Note that the argument holds since the index of expert i is a non-decreasing function of accuracy for that expert μ^i. We will prove this by showing the positivity of the first derivative of $E(\ln q_t^i)$ with respect to μ^i. Indeed, because $\int \ln x = x \ln x - x$,

$$\begin{aligned} \frac{\partial E(\ln q_t^i)}{\partial \mu^i} &= \frac{1}{a} \int\limits_{1-a}^{1} \ln x dx - \frac{1}{1-a} \int\limits_{0}^{1-a} \ln x dx \\ &= \frac{1}{a} \left[-1 - (1-a)\ln(1-a) + 1 - a \right] - \frac{1}{1-a} \left[(1-a)\ln(1-a) - (1-a) \right] \\ &= -\frac{1}{a} + \left[1 - a - (1-a)\ln(1-a) \right] \left[\frac{1}{a} + \frac{1}{1-a} \right] \\ &= -\frac{1}{a} + \left[1 - \ln(1-a) \right] \frac{1}{a} \\ &= -\frac{\ln(1-a)}{a} \\ &> 0, \end{aligned}$$

as desired.

CHAPTER 9

UPPER BOUND ON LOSS

In this chapter, we analyze the loss of the algorithm, which is defined by a logarithmic function according to (7.3). Let us rewrite the update rule in (7.2),

$$\begin{aligned} p_{t+1}^i &= p_t^i \left[I\{i \notin E_t\} + I\{i \in E_t\} \left(I\{y_t = 1\} \frac{x_t^i}{\hat{y}_t} + I\{y_t = 0\} \frac{1 - x_t^i}{1 - \hat{y}_t} \right) \right] \\ &= p_t^i f(x_t^i, y_t). \end{aligned}$$

By induction, the weight of expert i at time n is

$$\begin{aligned} p_n^i &= p_0^i \prod_{t=1}^{n} f(x_t^i, y_t) \\ &= \frac{1}{N} \prod_{t=1}^{n} f(x_t^i, y_t), \end{aligned} \tag{9.1}$$

where the second equality follows from the original uniformity of weights of experts.

Now, define the 'current best' expert by $i_t^* = \arg\max_{i \in E_t} \alpha_t^i \beta_t^i$, where

$$\alpha_t^i = \frac{\sum\limits_{t'=1}^{t} I\{i \in E_{t'}\}}{t},$$

$$\beta_t^i = \frac{\sum\limits_{t'=1}^{t} \left[I\{y_{t'} = 1\} \ln x_{t'}^i + I\{y_{t'} = 0\} \ln(1 - x_{t'}^i) \right]}{\sum\limits_{t'=1}^{t} I\{i \in E_{t'}\}}.$$

Intuitively, α_t^i is the availability accumulated up to time t, or the number of occurrences of expert i over t rounds. β_t^i is the accuracy accumulated up to time t of expert i, calculated by his performance over the number of predictions up to t. By this definition, the 'current best' expert is the one who is currently available, and whose product of accumulated availability and accuracy up to time t is largest. We will compare the loss of the algorithm to that of the 'current best' expert by deriving the regret as in (4.1).

$$\begin{aligned}
\sum_{t=1}^{n} l(\hat{y}_t) - \sum_{t=1}^{n} l(x_t^{i_t^*}) &= \sum_{t=1}^{n} \left[-I\{y_t = 1\} \ln \hat{y}_t - I\{y_t = 0\} \ln(1 - \hat{y}_t)\right] \\
&\quad - \sum_{t=1}^{n} I\{i_t^* \in E_t\} \left[-I\{y_t = 1\} \ln x_t^{i_t^*} - I\{y_t = 0\} \ln(1 - x_t^{i_t^*})\right] \\
&= \sum_{t=1}^{n} I\{i_t^* \in E_t\} \left[I\{y_t = 1\} \ln \frac{x_t^{i_t^*}}{\hat{y}_t} + I\{y_t = 0\} \ln \frac{1 - x_t^{i_t^*}}{1 - \hat{y}_t}\right] \\
&= \sum_{t=1}^{n} \ln f(x_t^{i_t^*}, y_t) \\
&= \ln \prod_{t=1}^{n} f(x_t^{i_t^*}, y_t) + \ln p_0 - \ln p_0 \\
&= \ln p_0 \prod_{t=1}^{n} f(x_t^{i_t^*}, y_t) - \ln p_0, \\
&= \ln \frac{1}{N} \prod_{t=1}^{n} f(x_t^{i_t^*}, y_t) + \ln N, \qquad (9.2)
\end{aligned}$$

where p_0 is the initial weight of the 'current best' expert, which is $1/N$. Without loss of generality, assume that the set of experts ranked by the descending order of accuracy is $\{1, 2, ..., N\}$. Obviously, if expert 1 is always present, then the current best expert is expert 1. Otherwise, the current expert could be any expert whose accumulated product is currently highest. Thus, this current best expert need not be expert 1.

Now assume that expert 1 is always available. Recall that the weight of each expert is always less than one, so $\ln p_n^1 \leq 0$, or equivalently, from (9.1)

$$\ln \frac{1}{N} \prod_{t=1}^{n} f(x_t^1, y_t) \leq 0.$$

From the above observation,

$$\ln \frac{1}{N} \prod_{t=1}^{n} f(x_t^{i_t^*}, y_t) \leq \ln \frac{1}{N} \prod_{t=1}^{n} f(x_t^1, y_t) \leq 0.$$

From (9.2),

$$\sum_{t=1}^{n} l(\hat{y}_t) - \sum_{t=1}^{n} l(x_t^{i_t^*}) \leq \ln N.$$

One can see that by applying the average update rule (7.1), the algorithm suffers an amount of loss no more than the loss of the 'current best' expert plus the term $\ln N$.

CHAPTER 10

SIMULATION RESULTS

We consider a data set consisting of 20 objects, 10 experts and 1000 rounds. So, each object has a 1000 $\times$ 10 rating table containing the votes of the 10 experts over 1000 rounds.

Table 10.1: Recommendations of experts for an object.

Outcome	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
1	0.15	0.2	0.85		0.23	0.6	0.9	0.95	0.7	
1	0.5		0.8	0.4	0.83		0.2	0.5	0.6	0.9
0	0.4	0.1	0.15	0.06	0.5	0.8			0.5	0.4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 10.1 gives an example of the recommendations for an object. The predictions of experts take values in $[0, 1]$, while outcomes (user's feedback after consumption) take only two values, 0 or 1. In our simulation, we attempt to create different scenarios for the distribution of experts' predictions. In addition, we change the accuracy and availability of experts, as well as the fraction p of bad objects. We compare our algorithm with Dsybil, the algorithm proposed in [7]. One of the problems in comparing these algorithms is that the definitions of loss for each algorithm are different. We therefore consider the following common definition, the total number of mistakes that the algorithm makes:

$$L_t = \sum_{v=1}^{t} |y_v - round(\hat{y}_v)|.$$

Above, "round" is a function that approximates a real number by its nearest integer. In Dsybil, $round(\hat{y}_v) = \hat{y}_v$ since $\hat{y}_v$ is 0 or 1.

In the simulations, we consider the following two cases. In the first case, there is one expert who is always correct. That expert is not necessarily awake at all times, but whenever he is, he provides the right recommendation. In the

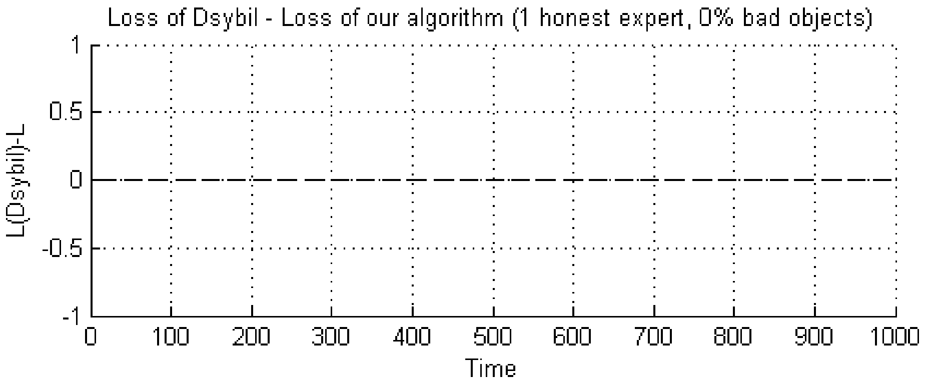

Figure 10.1: Comparison of loss with one honest expert and 0% of bad objects.

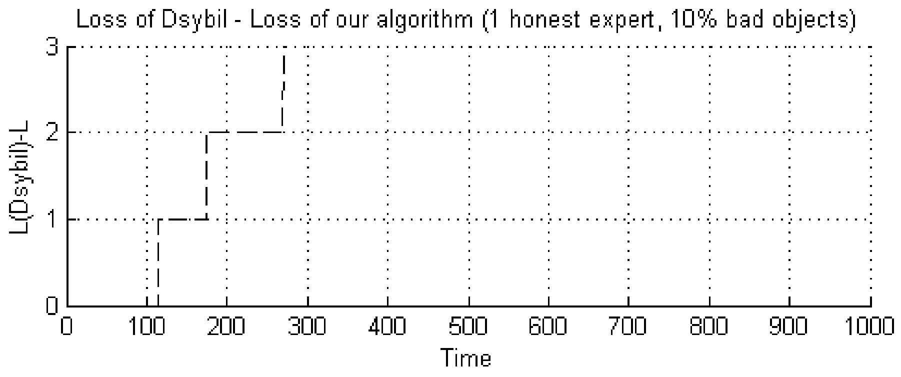

Figure 10.2: Comparison of loss with one honest expert and 10% of bad objects.

second case, there are two best experts who are correct almost (at very high accuracy) always. Figure 10.1 shows the comparison of the losses when there is no bad object, for the case of a single best expert. In this case, Dsybil and our algorithm only need to follow the best expert, and both have the same performance. However, if the percentage of bad objects is increased (to 10% in Figure 10.2 and 30% in Figure 10.3), while still assuming a single correct expert, our algorithm outperforms Dsybil due to the following explanation.

In our algorithm, all experts express their opinions regardless of the quality of the object, while in Dsybil, the experts only vote on good objects, and do not provide any recommendations for the bad objects. Thus, our algorithm still follows the "best" expert (the one with recommendation close to 0) and does not suffer any additional loss. In contrast, knowing that the object is bad, the "best" experts in Dsybil do not provide their opinion concerning the object. Dsybil therefore must depend on other experts (who are usually not

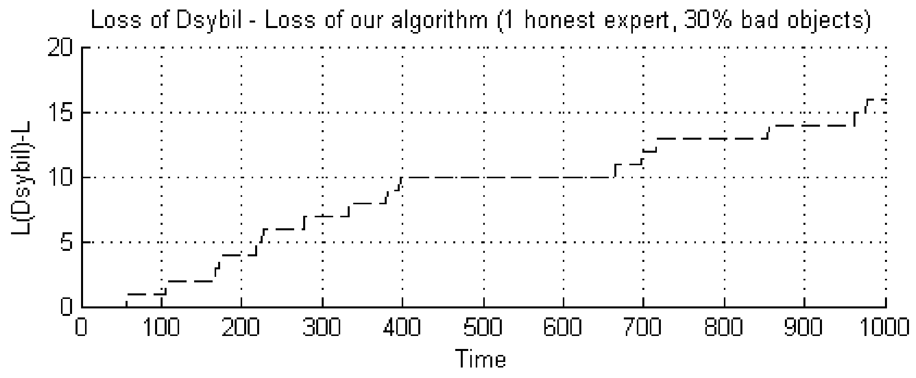

Figure 10.3: Comparison of loss with one honest expert and 30% of bad objects.

honest experts) rather than the "best" experts, and the recommendations will consequently follow the opinions of others. The additional loss is partially induced by this shortcoming of Dsybil. Therefore, for Dsybil to perform acceptably, a high percentage of the objects must be good. Clearly, such a requirement is quite unrealistic for an online recommendation system.

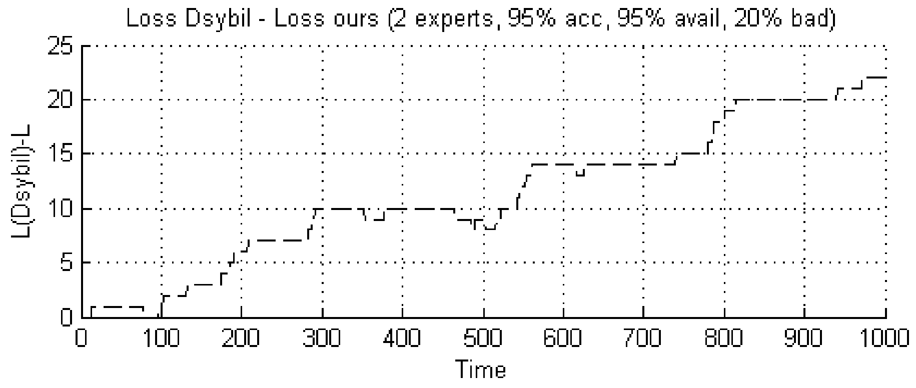

Figure 10.4: Comparison of loss with two best experts and 20% of bad objects.

In Figure 10.4, we analyze the case where two best experts have 95% accuracy and 95% availability. Our algorithm still outperforms Dsybil because it converges faster to one of the best experts. Figure 10.5 and Figure 10.6 show that Dsybil keeps switching between the two experts and thus suffers a higher loss, while our algorithm tracks one of the best experts after some number of rounds.

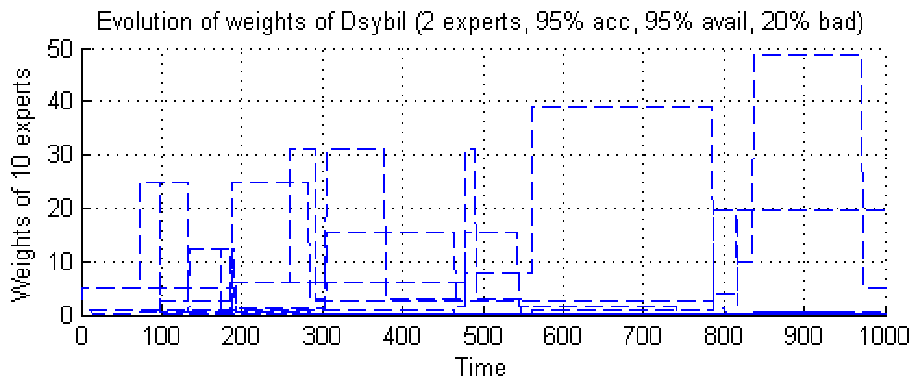

Figure 10.5: Evolution of weights in Dsybil with two best experts.

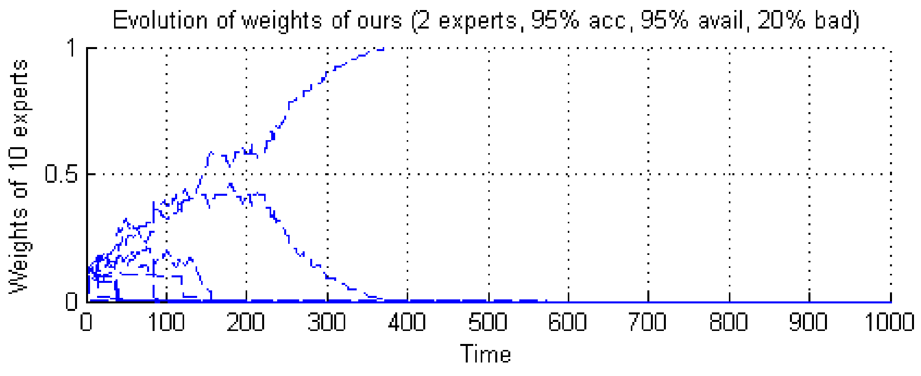

Figure 10.6: Evolution of weights in our algorithm with two best experts.

CHAPTER 11

CONCLUDING REMARKS

In the first part, we address the feasibility optimality of the class of periodwise static priority policies for the problem of providing QoS in unreliable wireless networks. When the deadline is limited to one period, by establishing the submodularity of the busy time function, we have shown that it is enough to consider that class, instead of the more general class of history-dependent policies. For the general problem where packet deadlines are greater than one period, the existence of a feasibility optimal randomized stationary periodwise static priority policy remains an open problem.

In the second part, we have considered the problem of designing an online recommendation system. We have proposed a recommendation algorithm that uses an average-weight update rule and proved its convergence to the best expert. Moreover, we derived an upper bound on its loss. A key advantage of our algorithm, besides the theoretical guarantees, is that we have relaxed many of the assumptions commonly made by recommendation systems that do not hold in practice. For instance, these algorithms often require a large fraction of good objects in the voting pool, the presence of experts with the same taste as the user receiving the recommendation, or experts who vote on all or majority of objects. Our simulation results show that the proposed algorithm outperforms the current state-of-art recommendation algorithm, Dsybil [7], even in a more general setting.

REFERENCES

[1] I.-H. Hou, V. Borkar, and P. R. Kumar, "A theory of QoS for wireless," in *Proceedings of Infocom 2009*, Rio de Janeiro, Brazil, 2009, pp. 486–494.

[2] A. Blum and Y. Mansour, "From external to internal regret," *The Journal of Machine Learning Research*, vol. 8, pp. 1307–1324, 2007.

[3] D. Haussler, J. Kivinen, and M. Warmuth, "Tight worst-case loss bounds for predicting with expert advice," University of California at Santa Cruz, Santa Cruz, CA, Tech. Rep. UCSC-CRL-94-36, 1994.

[4] V. G. Vovk, "Aggregating strategies," in *Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT '90)*, San Francisco, CA, 1990, pp. 371–386.

[5] C. Bianchi, Y. Freund, D. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth, "How to use expert advice," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC '93)*, 1993, pp. 382–391.

[6] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Using and combining predictors that specialize," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, El Paso, Texas, May 1997, pp. 334–343.

[7] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao, "Dsybil: Optimal sybil-resistance for recommendation systems," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2009, pp. 283–298.

[8] A. Dua and N. Bambos, "Deadline constrained packet scheduling for wireless networks," in *IEEE 62nd Vehicular Technology Conference, VTC-2005-Fall*, Dallas, TX, 2005, pp. 196–200.

[9] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," in *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia (WOWMOM '99)*, New York, NY, 1999, pp. 35–42.

[10] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, pp. 466–478, 1993.

[11] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of Applied Probability*, vol. 25, pp. 287–298, 1988.

[12] D. D. Yao, "Dynamic scheduling via polymatroid optimization," in *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures.* London, UK: Springer-Verlag, 2002, pp. 89–113.

[13] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, Nov. 1989, pp. 256–261.

[14] N. C. Bianchi and G. Lugosi, *Prediction, Learning, and Games.* New York, NY: Cambridge University Press, 2006.

[15] J. Kivinen and M. K. Warmuth, "Averaging expert predictions," in *Computational Learning Theory: 4th European Conference (EuroCOLT 99)*, Nordkirchen, Germany, 1999, pp. 153–167.

[16] V. Vovk, "Competitive on-line learning with a convex loss function," *Computing Research Repository*, vol. abs/cs/0506041, 2005.

[17] A. Chernov, Y. Kalnishkan, F. Zhdanov, and V. Vovk, "Supermartingales in prediction with expert advice," *Journal of Theoretical Computer Science*, vol. 411, pp. 2647–2669, 2010.

[18] R. D. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," in *21st Annual Conference on Learning Theory - COLT 2008*, Helsinki, Finland, 2008, pp. 425–436.