Online Learning of Non-stationary Sequences

by

Claire E. Monteleoni

A.B. Harvard University, 1998

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fufillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
June 2003

Signature of Author ....
Department of Electrical Engineering and Computer Science
May 19, 2003

Certified by: ....
Tommi S. Jaakkola
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: ....
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Online Learning of Non-stationary Sequences
by
Claire E. Monteleoni

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We consider an online learning scenario in which the learner can make predictions on the basis of a fixed set of experts. The performance of each expert may change over time in a manner unknown to the learner. We formulate a class of universal learning algorithms for this problem by expressing them as simple Bayesian algorithms operating on models analogous to Hidden Markov Models (HMMs). We derive a new performance bound for such algorithms which is considerably simpler than existing bounds. The bound provides the basis for learning the rate at which the identity of the optimal expert switches over time. We find an analytic expression for the *a priori* resolution at which we need to learn the rate parameter. We extend our scalar switching-rate result to models of the switching-rate that are governed by a matrix of parameters, i.e. arbitrary homogeneous HMMs. We apply and examine our algorithm in the context of the problem of energy management in wireless networks. We analyze the new results in the framework of Information Theory.

Thesis Supervisor: Tommi S. Jaakkola
Title: Associate Professor of Electrical Engineering and Computer Science

Contents

Chapter 1

Introduction

1.1 Problem Overview

Online Learning refers to a problem in Machine Learning in which training examples are only received one at a time, and the learner must make a prediction at each time-step. We consider a problem framework in which the learner knows nothing about the generation of the observations, i.e. the learner is not given any process statistics for the observation sequence, let alone any stationarity information. Thus there are no statistical assumptions that the learner can safely make about the observation sequence; observations could even be generated online by an adaptive adversary. It is important to differentiate this framework from some other solution techniques referred to as "online," in which the learner has access to the data in batch form, but chooses to process examples sequentially, for performance gains or due to resource constraints. Here the limitation of viewing data sequentially is part of the problem, as is the requirement that the learner make a prediction on each example, and seek to minimize cumulative prediction loss over the examples seen.

Since no statistical assumptions can be made in advance, a key problem is to be able to adapt to the potentially non-stationary nature of the observation sequence. It may be possible to model the sequence as being non-stationary due to occasional switches between different stationary processes that in turn govern the observations. Thus the ability to shift emphasis and resources from one prediction method, or "expert," to another, in response to such changes in the observations, would clearly be valuable in many applications, from financial portfolio management to energy management in wireless networks.

More concretely, we consider here an online learning scenario in which the learner can make predictions on the basis of a fixed set of experts. The learner does not know the mechanisms by which the experts arrive at their predictions, but is able to observe the losses incurred by the experts on each new observation, where loss is a deterministic function of a prediction, and the true observation. At each learning iteration t, the learner observes the predictions of the experts, $a_{1,t}, \ldots, a_{n,t}$, and then must make its own prediction $\hat{y}_t$. Then the learner observes the true observation value, y_t, and can update its model. It can compute a loss function, $L(\cdot\,, y_t)$ on it's own prediction and on the predictions of each of the experts. The learner's goal is to minimize its cumulative loss over the learning iterations.

1.1.1 Possible Approaches

In the typical online learning framework of the type we consider, in a classification problem for example, the learner can gain information on the prediction accuracy of its current model, by predicting the label of the incoming example, before observing the true label. Then it can update its model in light of the prediction error incurred on that example. The problem becomes how the learner ought to represent and update its model, in order to minimize prediction error over time, subject to limits on computational resources such as access to examples only one at a time, and not being allowed to store all previously seen examples.

In the framework we have described, no sampling assumptions can be made about the sequence to be predicted, and thus the design of algorithms has to be guided by relative performance measures. For example, we can aim to do as well as the best method chosen in hindsight, where hindsight refers to full knowledge of the sequence to be predicted. The analysis of algorithms is therefore focused on establishing bounds on the regret, or the difference between the cumulative loss of the algorithm and the loss of the best method in the same class, chosen in hindsight. Note that the observation sequences that would give rise to the maximal regret for an algorithm are not unpredictable (quite the opposite). Designing algorithms on the basis of optimizing worst-case regret can therefore be useful in practice. Other relative performance measures can be constructed by comparing across systematic variations in the sequence [Foster and Vohra, 1999].

Previous work has taken the following approaches to this learning problem. A simple learner might try to quickly identify a single best expert to rely on throughout the prediction task [Littlestone and Warmuth, 1989]. Such predictions are suboptimal when we can, perhaps with hindsight, identify segments in the observation sequence where a different expert consistently outperfoms others. When considering switches from one expert to another in response to the changes in the observation sequence [Herbster and Warmuth, 1998], the learner has to make some assumptions about the level of non-stationarity expected in the observations. This can be done, for example, by setting the rate at which switches are expected to occur between such segments of observations.

In previous work, this switching-rate parameter had to be set *a priori*. However, any fixed setting of the rate parameter (such as zero which corresponds to finding a single best expert) is liable to lead to poor predictions when little or no prior knowledge is available about the process governing the observations. The learner could reap substantial gains by adjusting the expected switching-rate on the basis of the observations. Thus we seek to better model the possible non-stationarity of the observation sequence by learning its switching dynamics online.

1.2 Related Work

1.2.1 Online Learning

We have outlined the type of online learning problem we are concerned with. Within this framework, Littlestone and Warmuth proved relative loss bounds for algorithms that have access to a fixed set of experts, and attempt to do as well as the best expert chosen with hindsight [1989]. Herbster and Warmuth extended this work to algorithms whose cumulative loss can be bounded relative to the loss of the best k-partitioning of the sequence, where a partitioning defines segment boundaries and maps segments to experts [1998]. The bound on the learner's regret is expressed in terms of the match between the actual switching-rate (or the best in hindsight) and the rate at which the learner assumes that the switching occurs. Similar algorithms, also with relative loss guarantees, have been developed for adaptive game playing [Freund and Schapire, 1999], online portfolio management [Helmbold *et al.*, 1996], paging [Blum *et al.*, 1999], and the k-armed bandit problem [Auer *et al.*, 1995].

1.2.2 Information Theory

Coding

Coding theory is a principled way of establishing and analyzing limits on the compressibility of information, which we consider due to the duality between prediction and compression [Feder *et al.*, 1992]. Feder *et al.* showed that, in the upper and lower bounds, the redundancy of a sequence is equivalent to its predictability, and conversely, incompressibility is equivalent to unpredictability [1992]. We consider here results which pertain to instantanous codes, i.e. coding that can be done sequentially, in regards to our focus on online learning.

Universal Source Coding refers to the problem of being able to code all input sources that are i.i.d. and have entropy below a certain rate, with a single instantanous coding scheme, and achieve arbitrarily small probability of error (asymptotically with increasing block size) [Cover and Thomas, 1991; Rissanen, 1984]. In order to achieve universal source coding, a larger block size is needed to ensure good performance on all the input sources considered, than would be needed with knowledge of the true input source. This is a different problem framework than ours however, as our problem does not provide limitations on the observation process, such as bounded entropy, or stationarity assumptions.

Feder and Merhav give a Hierarchical Universal Coding algorithm, which at first seems similar to the hierarchical learning algorithm we introduce in Chapter 2, but the problem we solve is different. The `HUC` algorithm can achieve the minimal redundancy, i.e. the channel capacity, when the encoder knows that the active class, i.e. the true input source, must fall into a known set of classes [1996]. In this framework, the set of possible channels is known, as well as the one-to-one mapping from source to channel. In our framework however, the learner is not given the information that the observed source must belong to a known set of classes.

Universal Prediction

Feder *et al.* extend the notion of universal source coding to universal prediction of binary sequences [1992]. This notion of universality is stronger however, in that the input sequence is arbitrary, and optimality is judged by minimizing regeret with respect to the best predictor in a given class, computed in hindsight. They derive a universal prediction algorithm from an incremental, variable-rate coding algorithm due to Ziv and Lempel [1978], and show that it asymptotically attains the minimum error of any finite-state predictor. The universal framework considered by [Littlestone and Warmuth, 1989; Haussler *et al.*, 1998; Herbster and Warmuth, 1998] is this one, but with an attempt to provide bounds that hold for finite sequence length, as opposed to asymptotically, and where the comparison class of predictors is finite [Haussler *et al.*, 1998].

1.2.3 Stochastic Complexity

Another area of related work is the notion of stochastic complexity [Rissanen, 1986; 1989]. Rissanen shows that optimal prediction is possible with the model that is formed by computing the stochastic complexity of the sequence to be learned. The stochastic complexity of a sequence is defined as its complexity in relation to a certain class of probabilistic models. Formally, that value is the "fewest number of binary digits with which the data can be encoded," by using that class of models [1986]. In this framework, there are no given statistical assumptions on the data generation process. However to analyze the stochastic complexity of a sequence, specifically to upper bound it with respect to a class of probabilistic models, such assumptions are needed. This is not the case in the analysis done in [Littlestone and Warmuth, 1989; Herbster and Warmuth, 1998], and in this work, in which upper bounding the regret of online learning algorithms, with respect to a class of prediction models, does not require any assumptions about the observation sequence.

1.3 Our Approach

We consider learning algorithms of the type from [Herbster and Warmuth, 1998; Littlestone and Warmuth, 1989] which rely on a collection of experts. We re-derive these online learning algorithms from the point of view of simple Bayesian prediction methods operating on models similar to HMMs, where the hidden state dynamics corresponds to a learner's model of switching between experts or segments in the sequence. This switching-rate characterizes how quickly the corresponding learner can respond to changes. Existing relative loss bounds are not formulated in such a way as to provide reasonable guarantees for such algorithms, and specifically to learn the parameter controlling the switching-rate at the appropriate resolution. We derive bounds on the regret of such algorithms that are considerably simpler than previous results, and whose form facilitates such guarantees. We introduce a hierarchical algorithm for updating the switching-rate parameter, based on the observation sequence, while performing the learning task. Using the new performance bound, we derive an analytic form for the computation of the optimal resolution at which to learn the switching-rate parameter, which can be computed *a priori*, independent of the process to be observed. We then generalize the relative loss bound to models that allow a matrix of parameters to govern the switching-rate, as opposed to a single scalar.

Chapter 2

Approach and Analysis

Our approach to the problem of better modeling non-stationary sequences, by learning their switching-rates online, starts with deriving existing online learning algorithms [Herbster and Warmuth, 1998; Littlestone and Warmuth, 1989], but starting from a different view-point than previously in the literature. These algorithms can be seen as simple Bayesian estimation algorithms operating on HMM models, in which the hidden states correspond to the set of experts, and the observations relate to expert predictions. Thus the state transition probabilities in the HMM model are the parameters governing the rate of switching from favoring one expert as the current best predictor of observations, to favoring another expert. We will re-derive existing algorithms to show that the updates on the weights of each expert are analogous to performing Bayesian updates of an appropriately defined HMM. In Section 2.1 we will specify the form of HMM for which performing such Bayesian updates yields existing online learning algorithms.

2.1 Deriving Online Learning Algorithms from HMMs

We consider here Bayesian algorithms for combining expert opinions in non-stationary environments. As introduced in Chapter 1, the learner has access to n experts, $a_1, \ldots, a_n$. Each expert makes a prediction at each time-step over a finite (known) time period $t = 1, \ldots, T$. We denote the i^{th} expert at time t as $a_{i,t}$ to suppress any details about how the experts arrive at their predictions and what information is available to facilitate the predictions. These details may vary from one expert to another and may change over time. From the point of view of deriving the combination method we can think of the experts as making probabilistic predictions of the form

$$P(y_t|a_{i,t}) = P(y_t|a_i, y_1, \ldots, y_{t-1}) \tag{2.1}$$

where y_t is the observation at time t. This form is akin to specifying emission probabilities in a generalized HMM.

It is important to note that, throughout this work, the form of graphical model that we refer to is more general than a typical HMM, for which the term is standardly used. In our model, we allow the current observation to depend on past observations, not just hidden state. It is a conditional Markov model in that conditioning on the observations induces a Markov distribution over the hidden states. For the sake of simplifying terminology however, we will continually use the term "HMM" to refer to our model class.

As described in Chapter 1, after making its prediction, the learner receives the true observation, y_t, and can then apply a loss function L to its prediction and the experts' predictions, where L measures the prediction loss given the true observation. Since the learner does not know the mechanisms by which the experts arrive at their predictions, it ought to exploit the information obtained by observing the losses of the experts. By defining the log-likelihood of the observation given the expert as the negative of the expert's prediction loss on that observation, the learner can then perform Bayesian updates of its distribution over experts. Bayesian updating is the optimal way to update a statistical model, assuming the predictions were generated from such a model. In cases where this assumption need not hold, Bayesian updating still has good relative performance guarantees. It entails updating the model parameters so as to minimize the negative log-likelihood of the observations given the model [Berger, 1985].

Thus the learner should update an HMM in which the hidden state is the identity of the expert that is the best predictor of the current observation, and the emission probabilities are defined as

$$P(y_t|a_{i,t}) = e^{-L(i,t)} \tag{2.2}$$

where $L(i,t)$ is the loss of expert i at time t. This allows the interpretation of the prediction loss of expert i at time t as

$$L(i,t) = -\log P(y_t|a_{i,t}) \tag{2.3}$$

where "log" refers to the natural logarithm throughout this work.

Such a model makes predictions by conditioning over the experts' predictions, subject to the distribution $p_t(i) = P(i|y_1, \ldots, y_{t-1})$, its current weighting of the experts, which is updated as forward probabilities in an HMM (which we will show below). Then the prediction loss of such an algorithm is naturally given by

$$\begin{aligned} L(p_t, t) &= -\log \sum_{i=1}^{n} p_t(i) P(y_t|a_{i,t}) && (2.4) \\ &= -\log \sum_{i=1}^{n} p_t(i) e^{-L(i,t)} && (2.5) \end{aligned}$$

This general form need not only pertain to models in which the prediction function is the weighted mean of the experts' predictions, and prediction loss is is expressible as log-loss, i.e. the negative log-likelihood mentioned above. As shown in [Haussler *et al.*, 1998; Herbster and Warmuth, 1998], by appropriate setting of the constants, c

and η the condition

$$L(\texttt{pred}(\vec{a},\, t), t) \leq -c \, \log \sum_{i=1}^{n} p_t(i)\, e^{-\eta L(i,t)} \tag{2.6}$$

(where $\vec{a}$ is the set of expert predictions and `pred` is the algorithm's prediction function), holds as an upper bound for many possible joint choices of loss function and prediction function. Since all these other pairings are related, through appropriate choices of c and η, to using the log-loss function and the weighted mean prediction function, in which case $c = \eta = 1$, we do not include these constants in our exposition.

For the purpose of deriving these algorithms, we make the following assumptions, although they are not required for our new relative loss bound. We assume that conditioned on all the past observations, the experts' current predictions are independent of one another. Additionally, we assume that experts make predictions causally, i.e. they do not have access to future observations at the time they predict. Now to evaluate $p_t(i)$ we actually only need access to the history of expert losses $L(i, \tau)$, $i = 1, \ldots, n$, $\tau = 1, \ldots, t-1$. For example, in the case of finding the best single expert, we would update $p_t(i)$ according to

$$\begin{aligned} p_t(i) &= \frac{1}{Z_t}\, p_{t-1}(i)\, P(y_{t-1}|a_{i,t-1}) && (2.7)\\ &= \frac{1}{Z_t}\, p_{t-1}(i)\, e^{-L(i,t-1)} && (2.8) \end{aligned}$$

where Z_t normalizes the distribution. This is consistent with Bayesian estimation, with the identity of the best expert as a discrete parameter. These updates are analogous to the weight updates of the online learning algorithm `Static-expert`, due to [Littlestone and Warmuth, 1989].

For a more general model, we can allow the identity of the best expert to change over time. We focus here on a simple Markov model of the switching between experts, $P(i_t|i_{t-1}, \alpha)$, where α denotes the parameters of the assumed transition model. For now, the parameter vector α can be viewed as fixed. The updating of the weighting over the experts is in this case analogous to computing forward probabilities in an HMM:

$$\begin{aligned} p_t(i) &= \frac{1}{Z_t} \sum_{j=1}^{n} p_{t-1}(j)\, P(y_{t-1}|a_{j,t-1})\, P(i|j, \alpha) && (2.9)\\ &= \frac{1}{Z_t} \sum_{j=1}^{n} p_{t-1}(j)\, e^{-L(j,t-1)}\, P(i|j, \alpha) && (2.10) \end{aligned}$$

where Z_t normalizes the distribution. We assume that the weighting is initially uniform $p_1(i) = 1/n$, since the learner has no knowledge about the experts, so introducing an arbitrary preference for a given expert could hurt performance if that expert turns out to be a poor predictor.

When the Markov dynamics consists of only two types of transitions, a self transition with probability $1-\alpha$, and a transition to another expert with probability $\alpha/(n-1)$, for a scalar $0 \le \alpha \le 1$, the update equation reduces to the `Fixed-share` algorithm, due to [Herbster and Warmuth, 1998]. We will restrict ourselves in this chapter to this simple fixed share Markov model although our analysis extends directly to any Markov (Chapter 3) and higher order models of switching.

The cumulative loss of such an online learning algorithm is the loss incurred over a time horizen of T training iterations,

$$L_T(\alpha) = \sum_{t=1}^{T} L(p_t, t) \tag{2.11}$$

We use this quantity as a performance metric for these types of algorithms.

2.1.1 Existing Relative Loss Bounds

Having specified the existing online learning algorithms, we can now state the existing relative loss bounds of such algorithms.

For the `Static-expert` algorithm, the existing relative loss bound is given by the following theorem.

Theorem 2.1.1 *[Herbster and Warmuth, 1998] Let $L_T(\mathtt{alg})$ be the cumulative loss of the `Static-expert` algorithm on an arbitrary sequence of T observations, and let $L_T(a_i^*)$ be the cumulative loss of the best expert (in the given set of experts) for that particular sequence, chosen in hindsight. Then*

$$L_T(\mathtt{alg}) \le L_T(a_i^*) + \log n \tag{2.12}$$

Proof This is proven in [Herbster and Warmuth, 1998], but here we use our notation. When the algorithm is instantiated with a pairing of a loss function and prediction function that satisfy the condition (2.6), then

$$L(p_t, t) \le -\log \sum_{i=1}^{n} p_t(i)\, e^{-L(i,t)} = -\log \sum_{i=1}^{n} p_t(i)\, P(y_t|a_i, y_1, \ldots, y_{t-1}) \tag{2.13}$$

Thus the cumulative loss of the algorithm over the sequence of T observations can be expanded as follows

$$L_T(\mathtt{alg}) = \sum_{t=1}^{T} L(p_t, t) \le -\sum_{t=1}^{T} \log \sum_{i=1}^{n} p_t(i)\, P(y_t|a_i, y_1, \ldots, y_{t-1}) \tag{2.14}$$

$$= -\sum_{t=1}^{T} \log P(y_t|y_1,\ldots,y_{t-1}) \tag{2.15}$$

$$= -\log\, p_1(y_1) \prod_{t=2}^{T} P(y_t|y_1,\ldots,y_{t-1}) \tag{2.16}$$

$$= -\log P(y_1,\ldots,y_T) \tag{2.17}$$

We can compute the probability of the joint observation sequence by summing over all hidden states, the probability of the observation sequence given that hidden state at every time-step, times the probability of being in that hidden state at every time-step.

$$-\log P(y_1,\ldots,y_T) = -\log \sum_{i=1}^{n} P(y_1,\ldots,y_T|a_{i,1},\ldots,a_{i,T})P(a_{i,1},\ldots,a_{i,T}) \tag{2.18}$$

In the `Static-expert` algorithm (i.e. updates of the form (2.8)), no switches between hidden states are modeled. Thus we can expand as follows

$$= -\log \sum_{i=1}^{n} p_1(i) P(y_1|a_{i,1}) \prod_{t=2}^{T} P(y_t|a_i, y_1,\ldots,y_{t-1}) \tag{2.19}$$

$$= -\log\, \frac{1}{n} \sum_{i=1}^{n} e^{-L(i,1)} \prod_{t=2}^{T} e^{-L(i,t)} \tag{2.20}$$

$$= -\log\, \frac{1}{n} \sum_{i=1}^{n} e^{-\sum_{t=1}^{T} L(i,t)} \tag{2.21}$$

Since $-\log(\cdot)$ decreases monotonically, we can upper bound this by the same function of any of the terms in the summation, yielding

$$L_T(\texttt{alg}) \leq -\log\, \frac{1}{n} e^{-\sum_{t=1}^{T} L(i,t)} = L_T(a_i) + \log n \tag{2.22}$$

Since this holds for all i, this yields the result, by choosing to compare with the loss of a_i^*, the cumulative loss minimizing expert.

□

For the `Fixed-share`(α) algorithm, the existing relative loss bound is given by the following theorem.

Theorem 2.1.2 *[Herbster and Warmuth, 1998] Let $L_T(\alpha)$ be the cumulative loss of the **`Fixed-share`** algorithm on an arbitrary sequence of T observations, where $0 \leq \alpha \leq 1$, and for any $k < T$, let L_T(best k-partitioning) be the cumulative loss of the best segmentation of that sequence into k segments and mapping of each of the* *segments to a member of the given expert set, computed in hindsight. Then*

$$L_T(\alpha) \leq L_T(\text{best } k\text{-partitioning}) + (T-1)[H(\alpha^*) + D(\alpha^*||\alpha)] + k\log(n-1) + \log n \tag{2.23}$$

where $\alpha^* = k/(T-1)$ *is the hindsight-optimal (cumulative loss minimizing) setting of switching-rate parameter,* α*, given* k*, and where* $D(\alpha^*\|\alpha) = \alpha^* \log \frac{\alpha^*}{\alpha} + (1-\alpha^*)\log\frac{1-\alpha^*}{1-\alpha}$.

We refer the reader to [Herbster and Warmuth, 1998] for the proof.

The existing relative loss bound (Theorem 2.1.1) for the `Fixed-share` algorithm is not formulated in a way that can provide sufficiently strong guarantees for our purposes. Specifically the value of the bound does not go to zero when $\alpha = \alpha^*$, which makes it inaccurate for measuring the cost of not using α^*, when only discrete values of α are tested in learning that parameter online. Thus we cannot use the existing bound to obtain tight guarantees for our computation of the optimal resolution at which to learn α (Section 2.5).

For the goal of designing an algorithm that can learn the switching-rate online, a bound based on the comparison class of `Fixed-share(`α`)` algorithms will provide stronger guarantees than the existing bound which compares to the loss of the best k-partitioning, a quantity that need not be attainable by any `Fixed-share(`α`)` algorithm.

2.2 A New Relative Loss Bound

We will provide here a substantially simpler bound, and one whose form is based on the comparison class of all `Fixed-share(`α`)` algorithms.

Theorem 1 *Let* $L_T(\alpha)$ *be the cumulative loss of the* ***Fixed-share*** *algorithm on an arbitrary sequence of* T *observations, where* $0 \leq \alpha \leq 1$*, and let* $L_T(\alpha^*) = \min_\alpha L_T(\alpha)$ *be the cumulative loss of the best such algorithm chosen in hindsight for the same sequence. Then*

$$L_T(\alpha) \;\; \leq \;\; L_T(\alpha^*) + (T-1)\, D(\alpha^*\|\alpha) \tag{2.24}$$

where $D(\alpha^*\|\alpha) = \alpha^* \log \frac{\alpha^*}{\alpha} + (1-\alpha^*)\log\frac{1-\alpha^*}{1-\alpha}$.

Proof As we show in Chapter 3, this is a direct result of Theorem 3, the Main Theorem for arbitrary, homogeneous HMMs.

Unlike earlier results, this worst-case bound vanishes when $\alpha = \alpha^*$ and does not depend directly on the number of experts. The dependence on n may appear indirectly through α^*, however. In comparison with the existing bound [Herbster and Warmuth, 1998], note that the cumulative loss of the best k-partitioning is less than the cumulative loss of Fixed-share(α^*), since for a given sequence, the loss of the best k-partitioning may not be attainable by any Fixed-share(α) algorithm. Thus the difference in comparison class explains why the existing bound does not go to zero for $\alpha = \alpha^*$. Note also that, in the new bound, while the regret appears proportional to T, this dependence vanishes for any reasonable learning algorithm that is guaranteed to find $\alpha = \alpha^* + \mathcal{O}(1/\sqrt{T})$, as we will show in Section 2.5.

The bound holds for a more general class of algorithms than the ones we have described above. Some of our assumptions in deriving the algorithms, namely that expert predictions are causal, and conditionally independant across experts, given the previous observations, are not required for the bound. In fact, the bound does not require any assumptions at all about the sequence of observations, nor the experts. The bound can be generalized to arbitrary transition probabilities in the HMM interpretation, which we will show in Chapter 3, and give the Main Theorem and proof in the general form.

2.2.1 Information Theoretic Interpretation

The form of the new bound (Theorem 1) is more intuitive than the existing bound for such algorithms (Theorem 2.1.1). This difference can be explained in an information theoretic framework, as follows. First note that it differs from the form of Theorem 2.1.1, by no longer including $H(\alpha^*)$ in the term that scales with T. The information theoretic explanation in [Herbster and Warmuth, 1998], for the form of their bound, is that it is the bound on the expected optimal code length for coding whether a shift between which expert is currently best will occur at any time-step, but by coding with α as the probability of a shift occuring, as opposed to α^*, the true probability. This formula, the bound on the expected optimal codelength when coding with α, if the true probability is actually α^*, is of the form:

$$H(\alpha^*) + D(\alpha^*||\alpha) \tag{2.25}$$

[Cover and Thomas, 1991]. Our new bound just contains the second quantity in the term that scales with T. We use a related analysis to the information theoretic one used in [Herbster and Warmuth, 1998], but the important difference is that even by coding the shifts with α^*, one has to pay $H(\alpha^*)$ in the expected optimal codelength. So what our bound actually formulates is the difference between the expected optimal codelength when coding with α from that when coding with α^*. Thus the $H(\alpha^*)$ terms cancel, and we are left with $D(\alpha^*||\alpha)$. This also explains why we could not obtain strong guarantees by using the existing bound in solving for the optimal search resolution for learning α. Since the dependence on α is the same in the two bounds, optimizing their bound to solve for the resolution would give the same resolution as ours, but the bound on the regret from learning α at that resolution would be much looser than ours (Section 2.4). This is because even at the optimal $\alpha = \alpha^*$, there is always the additional $H(\alpha^*)$ term in their bound, since their bound was formulated with respect to the best k-partitioning.

2.3 Algorithm `Learn-`α

We now give an algorithm to learn the switching-rate simultaneously with updating the weighting over the experts. Since the cumulative loss $L_T(\alpha)$ of each `Fixed-share` algorithm running with switching parameter α can be interpreted as a negative log-probability, we can evaluate the posterior distribution over the switching-rate according to

$$p_t(\alpha) = P(\alpha|y_1, \ldots, y_{t-1}) \propto e^{-L_{t-1}(\alpha)} \tag{2.26}$$

assuming a uniform prior over $\alpha \in [0, 1]$. Note that, as before, we use $p_t(\alpha)$ as a predictive weighting and thus it does not include the observation at the same time point. We can view this algorithm as finding the single best "α-expert," where the collection of α-experts is given by `Fixed-share` algorithms running with different switching-rates, α.

Allocation of Computational Resources

As mentioned in Chapter 1, the online learning framework derives from certain limitations on resources: access to a data-set only one sample at a time, and a memory limit preventing the learner from storing all previously viewed examples. Additionally, one can analyze solutions to such problems based on computation time, and memory usage. In light of limited computation time and memory, we will consider a finite resolution version of this algorithm, allowing only m possible choices for the switching-rate, $\alpha_j = j/(m + 1)$, where $j = 1, \ldots, m$. For a sufficiently large m we expect to be able to find $\alpha_j \approx \alpha^*$, i.e., suffer only a minimal additional loss due to not being able to represent the hindsight optimal value.

Let $p_{t,j}(i)$ be the distribution over the α-experts, b_j, defined by the j^{th} `Fixed-share` algorithm taking parameter $\alpha = \alpha_j$, and let $p_t^{top}(j)$ be the top-level algorithm producing a weighting over these α-experts. The top level algorithm is given by

$$\begin{aligned} p_t^{top}(j) &= \frac{1}{Z_t} p_{t-1}^{top}(j) P(y_{t-1}|b_{j,t-1}) && (2.27) \\ &= \frac{1}{Z_t} p_{t-1}^{top}(j) e^{-L(j,t-1)} && (2.28) \\ &= \frac{1}{Z_t} p_{t-1}^{top}(j) \sum_{i=1}^{n} p_{t-1,j}(i) P(y_{t-1}|a_{i,t-1}) && (2.29) \end{aligned}$$

where $p_1^{top}(j) = 1/m$, $p_{1,j}(i) = 1/n$, and $p_{t,j}(i)$ is updated according to (2.10). The loss of the top-level algorithm per time-step is defined as

$$L^{top}(p_t^{top}, t) = -\log \sum_{j=1}^{m} p_t^{top}(j) e^{-L(p_{t,j},t)} \tag{2.30}$$

$$= -\log \sum_{j=1}^{m} \sum_{i=1}^{n} p_t^{top}(j) p_{t,j}(i) P(y_t|a_{i,t}) \tag{2.31}$$

as is appropriate for a hierarchical Bayesian method.

2.4 Hierarchical Relative Loss Bound

We can now derive a relative loss bound for the top-level algorithm. The resulting bound can be used to find the optimal search resolution for the switching-rate parameter by minimizing the overall loss.

The tradeoff in finding the optimal learning resolution, i.e. the optimal search resolution in the space of $0 \leq \alpha \leq 1$, is between 1) the ability to identify the best `Fixed-share` expert, which degrades for larger m, and 2) the ability to find α_j whose loss is close to that of α^*, the optimal α for that sequence, which improves for larger m. The regret arising from having to consider a number of non-optimal values of the parameter in the search comes from the relative loss bound associated with tracking the best single expert, i.e. the relative loss bound for the `Static-expert` algorithm. [Herbster and Warmuth, 1998; Littlestone and Warmuth, 1989]. From that bound (Theorem 2.1.1) we see that this regret is simply $\log(m)$ in our context, since we are using m α-experts.

Theorem 2 *Let L_T^{top} be the loss of the hierarchical* ***`Learn-`α*** *algorithm for any sequence of T observations and any $m \geq 2$ as the search resolution. Let $L_T(\alpha^*) = \min_\alpha L_T(\alpha)$ be the cumulative loss of the best* ***`Fixed-share`*** *algorithm chosen in hindsight for that sequence, and let α_{j^*} be the best discrete choice (at resolution m) of the switching-rate chosen in hindsight for that sequence. Then*

$$L_T^{top} \leq L_T(\alpha^*) + \log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \tag{2.32}$$

Proof The result follows directly from successive application of the two relative loss bounds, the `Static-Expert` bound and our new `Fixed-share` bound.

When running `Learn-`α`(`m`)`, m possible values α_j are tested. The loss of the top-level algorithm, which updates its distribution over these α-experts, or sub-algorithms running `Fixed-share(`α_j`)`, is updated via the `Static-Expert` algorithm, and thus adheres to the relative loss bound (Theorem 2.1.1)

$$L_T^{top} \leq L_T(\alpha_{j^*}) + \log(m) \tag{2.33}$$

where $L_T(\alpha_{j^*})$ is the loss of the cumulative loss minimizing `Fixed-share(`α_j`)` chosen in hindsight for the same sequence.

Now we replace the $L_T(\alpha_{j^*})$ term with our new relative loss bound (Theorem 1) for `Fixed-share` algorithms running with arbitrary settings of the parameter α, yielding

$$L_T^{top} \;\leq\; L_T(\alpha^*) + \log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \tag{2.34}$$

$\square$

2.5 Computing the Optimal Learning Resolution

Having introduced a new online learning algorithm, `Learn-`α, it is important to quantify its performance in regards to its computation time, and memory usage. By solving for the optimal resolution at which to learn the switching-rate parameter, we are able to specify how many sub-algorithms need to be run in order to achieve near-optimal prediction accuracy, for a given number of learning iterations.

Since the relative loss bound we attain from Section 2.4 is thus

$$L_T^{top} - L_T(\alpha^*) \;\leq\; \log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \tag{2.35}$$

in order to find the optimal resolution for learning α, we must solve for m that minimizes the worst-case regret:

$$\log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \tag{2.36}$$

To facilitate this optimization we can proceed to bound the relative entropy term by using the fact that $|\alpha^* - \alpha_{j^*}| \leq \frac{1}{2(m+1)}$ for the majority of the range of α^*. This is because `Learn-`α tests α_j values spaced at $\frac{1}{(m+1)}$ intervals along the range from zero to one, so (unless it is towards the endpoints of the range) the farthest α^* could be from some setting of α_j is half that interval, or $\frac{1}{2(m+1)}$.

Lemma 2.5.1 *For large m, the bound is*

$$\leq \log(m) + (T-1)\, \frac{1}{m^2} \tag{2.37}$$

Proof We first note that in the original expression, in order for $D(\alpha^* || \alpha_{j^*})$ to be finite, α_{j^*} must being bounded away from zero and one, which is actually the case, since `Learn-`α tests $\alpha_j = \frac{j}{(m+1)}$ just on the range $j \in \{1, \ldots, m\}$.

Now define $\delta = |\alpha^* - \alpha_{j^*}|$. So $\delta \leq \frac{1}{2(m+1)}$ for almost all values of α^*, except for α^* in $[0, \frac{1}{2(m+1)})$, or $(1 - \frac{1}{2(m+1)}, 1]$, (ranges that are smaller for larger m, but for which we can at least bound $\delta \leq \frac{1}{(m+1)}$). Now, in the case $\alpha^* \geq \alpha_{j^*}$, the relative entropy term is approximately

$$\alpha^* \log \frac{\alpha^*}{\alpha^* - \delta} + (1 - \alpha^*) \log \frac{(1 - \alpha^*)}{(1 - \alpha^*) + \delta} \tag{2.38}$$

Applying Jensen's inequality, this is

$$\leq \log\left[\alpha^* \frac{\alpha^*}{\alpha^* - \delta} + (1-\alpha^*)\frac{(1-\alpha^*)}{(1-\alpha^*)+\delta}\right] \tag{2.39}$$

$$= \log\left[1 + \frac{\delta^2}{(\alpha^* - \delta)(1-\alpha^* + \delta)}\right] \tag{2.40}$$

Now using the upper bound on log, i.e. $\log x \leq x - 1$, we attain

$$\log\left[1 + \frac{\delta^2}{(\alpha^* - \delta)(1-\alpha^* + \delta)}\right] \leq \frac{\delta^2}{(\alpha^* - \delta)(1-\alpha^* + \delta)} \tag{2.41}$$

Note that this term will never have zero in the denominator. This follows from α_{j^*} being bounded away from zero and one.

We can continue to upper bound this quantity by

$$\frac{\delta^2}{(\alpha^* - \delta)(1-\alpha^* + \delta)} \leq 4\delta^2 \tag{2.42}$$

where the upper bound comes from the maximizing $\alpha^* = \frac{1}{2} + \delta$. By symmetry, the approximation bound still holds for the case $\alpha^* \leq \alpha_{j^*}$, where the right hand side of Equation 2.42 comes from the maximizing $\alpha^* = \frac{1}{2} - \delta$.

Now plugging into the regret bound (2.36) yields

$$\log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \leq \log(m) + (T-1)\, 4\delta^2 \tag{2.43}$$

$$\leq \log(m) + (T-1)\, \frac{4}{4(m+1)^2} \tag{2.44}$$

$$\leq \log(m) + (T-1)\, \frac{1}{m^2} \tag{2.45}$$

for almost all values of α^*. (Note that for α^* in the small intervals mentioned above, using the same analysis but the looser bound on δ, the loss bound is thus $\leq \log(m) + (T-1)\, \frac{4}{m^2}$).

□

We can now minimize with respect to m, yielding $m^* = \mathcal{O}(\sqrt{T})$ as the optimal resolution at which to learn α.

2.5.1 Interpretation of Optimal Resolution

Having solved for the optimal learning resolution allows us to address the resource allocation issues mentioned above, as the resolution tells us how many sub-algorithms must be run in parallel. This quantity scales with the number of training iterations for which one would like to judge the algorithm's performance, but it does so sublinearly. An additional benefit of our solution is that the resolution at which the parameter needs to be learned is not affected by n, the number of original experts.

We can now evaluate the value of the regret bound for our `Learn-`α algorithm, at the optimizing m. Taking the upper bound shown in Lemma 2.5.1 on our hierarchical loss bound (Theorem 2), we attain

$$L_T^{top} \;\leq\; L_T(\alpha^*) + \log(m) + (T-1)\, D(\alpha^* || \alpha_{j^*}) \tag{2.46}$$

$$\leq\; L_T(\alpha^*) + \log(m) + (T-1)\, \frac{1}{m^2} \tag{2.47}$$

This holds for all m, and thus plugging in the optimizing m^* yields

$$L_T^{top} \;\leq\; L_T(\alpha^*) + \frac{1}{2}\log(T) + O(1) \tag{2.48}$$

So the regret bound is

$$L_T^{top} - L_T(\alpha^*) \;\leq\; \frac{1}{2}\log(T) + O(1) \tag{2.49}$$

Thus we have shown an algorithm to learn α online, while performing its original learning task, that can do almost as well as the hindsight-optimal `Fixed-share` algorithm for that sequence, with regret upper bounded by half the natural logarithm of the number of training examples. This is significant, because without knowledge of the observation sequence, one does not know α^* for that sequence, and thus there is no justifiable way to set α for the `Fixed-share` algorithm, *a priori*.

We note that the accuracy corresponding to our optimal resolution, i.e. $\frac{1}{\sqrt{T}}$, is the same as the asymptotic accuracy of estimating a parameter from a sequence of T i.i.d. observations of that parameter. In the case of learning α, these observations correspond to whether the identity of the current best expert switched from one time-step to the next. So our resolution computation, based on worst-case regret bounds for a learner with no prior knowledge about the observation sequence, actually agrees with the learning resolution for a "typical" sequence.

2.6 Stochastic Complexity Theoretic Analysis

We now compare our results with the notion of stochastic complexity [Rissanen, 1986; 1989]. In the framework of the predictive Minimum Description Length (MDL) criterion, Rissanen defines the stochastic complexity of a sequence relative to a class of models as

$$C(\vec{x}) \;=\; \min_k \{L(\vec{x}|k) + \log^*(k) + c_1\} \tag{2.50}$$

where the form is exactly as in the formula for "(semi) *predictive* (stochastic) *complexity*" [1986], with a few symbols swapped for this exposition. Formally, in the stochastic complexity framework, $L(\vec{x}|k) = -\sum_t \log f_{k,\hat{\theta}(t)}(x_{t+1}|\vec{x}^t)$, where $\hat{\theta}(t)$ is the current estimate of θ, the parameter vector with k components, and $f_{k,\hat{\theta}(t)}(x_{t+1}|\vec{x}^t)$ is the probability of the $(t+1)$th datum, given the the previous observations, and $\hat{\theta}(t)$.

For such a metric, the upper bound is given by the following theorem, which we will quote from [Rissanen, 1986] with minor notation changes.

Theorem 2.6.1 *[Rissanen, 1986] Let the family of densities satisfy the conditions for independence for each k and $\theta \in \mathbf{\Omega}^k$, namely, $f_{k,\theta}(\vec{x}) = \prod_{t=1}^{T} f_{k,\theta}(x_t)$, and let $f_{k,\theta}(x_t)$ be three times continuously differentiable with respect to θ in the interior of a compact set $\mathbf{\Omega}^k$. Further, let the central limit theorem hold for some estimates $\hat{\theta}(\vec{x})$ of θ in the interior points such that the first four moments of $\sqrt{T}(\hat{\theta}(\vec{x}) - \theta)$ converge. Then $C(\vec{x})$ is optimal in that for all k and for all $\theta \in \mathbf{\Omega}^k$,*

$$C(\vec{x}) \leq -E_{k,\theta} \log f_{k,\theta}(\vec{x}) + \frac{k}{2} \log T + o(\log T) \tag{2.51}$$

where $C(\vec{x})$ is defined by Equation 2.50, and T is the length of the sequence $\vec{x}$.

We refer the reader to [Rissanen, 1986] for the proof.

We compare this to the relative loss bound for algorithms of the form `Learn-`α`(`m`)`, evaluated at the optimizing m.

$$L_T^{top} \quad \leq \quad L_T(\alpha^*) + \frac{1}{2}\log(T) + O(1) \tag{2.52}$$

Here we are considering sequences of length T, and models for which there is only one parameter, α, so $k = 1$. In this case, the stochastic complexity bound is

$$C(\vec{x}) \leq -E_{\alpha} \log f_{\alpha}(\vec{x}) + \frac{1}{2} \log T + o(\log T) \tag{2.53}$$

where α is a scalar. This bound holds for all α, i.e. all predictive models using one parameter, and all settings of that parameter.

The bounds are quite similar. $L_T(\alpha^*)$ is not necessarily less than $-E_{\alpha} \log f_{\alpha}(\vec{x})$, since the expectation is taken over all predictive models with one parameter, whereas $L_T(\alpha^*)$ refers to the best model of the type we specified in this chapter, HMMs with switching-rate α, i.e. learners running `Fixed-share(`α`)`. Our bound on the loss of `Learn-`α`(`m^*`)` can be seen as an upper bound on the statistical complexity of a sequence with respect to that class of models. For "typical" sequences (assuming the switches are i.i.d.), our worst-case bound is comparable to Rissanen's bound.

In stating the stochastic complexity bound, Rissanen terms as "optimal," prediction algorithms that adhere to it [1986], so in a preliminary comparison of the bounds, `Learn-`α`(`m^*`)` seems to be optimal in the stochastic complexity theoretic framework. Additionally our bound makes absolutely no assumptions about the sequence or α^*, whereas the value of the stochastic complexity bound depends on the process that generated the sequence.

Chapter 3

Generalized Result

3.1 Generalized Framework

To better model a non-stationary observation sequence, a richer model of the switching dynamics could increase the learner's prediction accuracy. So far we have treated models of the switching-rate of the sequence that use a single parameter, α. We now consider learning algorithms that use many parameters to model the switching dynamics of the observation process. While the model is homogeneous with time, we allow the transition probabilities to vary by hidden states. In the view of the previous exposition, this corresponds to allowing different switching-rates based on which expert the model currently believes to be best. Thus we allow a matrix of parameters to govern the switching dynamics between the hidden states, as opposed to a single scalar.

In the view of HMMs, this generalized bound now covers any generalized HMM that is homogenous with time, yet has an arbitrary transition matrix. Thus by specifying such a transition matrix, an online learning algorithm can be derived from Bayesion updates of such an HMM, in the method we described in Chapter 2, and any such algorithm is subject to the relative loss bound we present below. Here the transition probabilities between hidden states are specified per hidden state, as opposed to the simple scenario in Chapter 2. Thus we are extending our single parameter result to the multi-dimensional parameter case.

3.1.1 Preliminaries

We now consider an arbitrary homogeneous HMM with n hidden states, $a_1, \ldots, a_n$, parameterized by Θ, an $n \times n$ matrix, where θ_{ij} is the probability of transitioning from hidden state i to hidden state j from one time interval to the next. Thus stated, this implies that $\sum_j \theta_{ij} = 1, \forall i$. From such a model, we can derive an online learning algorithm that is a generalization of the one derived in Section 2.1. To generalize the

Bayesian update step of the distribution over hidden states, now

$$p_t(i) = \frac{1}{Z_t}\sum_{j=1}^{n} p_{t-1}(j)\, e^{-L(j,t-1)}\, P(i|j,\theta_{ji}) \tag{3.1}$$

$$= \frac{1}{Z_t}\sum_{j=1}^{n} p_{t-1}(j)\, e^{-L(j,t-1)}\, \theta_{ji} \tag{3.2}$$

where Z_t normalizes the distribution, and $p_1(i) = \frac{1}{n}$. The loss function is specified as in Section 2.1, i.e. $L(i,t) = -\log P(y_t|a_{i,t}) = -\log P(y_t|a_i, y_1, \ldots, y_{t-1})$, as are the remaining algorithm preliminaries.

3.2 Main Result for Arbitrary Transition Matrices

We now state the generalized theorem.

Theorem 3 (Main Theorem) *Let $L_T(\Theta)$ be the cumulative loss of an online learning algorithm of the type we proposed, (using Bayesian updating of a homogeneous HMM), on an arbitrary sequence of observations, where $0 \le \theta_{ij} \le 1$, $\sum_j \theta_{ij} = 1$, and let $L_T(\Theta^*) = \min_\Theta L_T(\Theta)$ be the loss of the best such algorithm chosen in hindsight for the same sequence. Then*

$$L_T(\Theta) - L_T(\Theta^*) \le (T-1)\sum_i \rho_i^* D(\Theta_i^*||\Theta_i) \tag{3.3}$$

$$\le (T-1)\max_i D(\Theta_i^*||\Theta_i) \tag{3.4}$$

where $D(\cdot||\cdot)$ is the relative entropy measure between two discrete distributions, $D(\Theta_i^*||\Theta_i) = \sum_j \theta_{ij}^* \log\frac{\theta_{ij}^*}{\theta_{ij}}$, and ρ_i^* is the posterior probability of being in hidden state i, at any time-step but the final one, computed in hindsight with the best such HMM for that sequence.

Proof Given in Appendix A.

3.3 Derivation of Single Parameter Case

We can now derive a simple case in which the learner's model describes the switching dynamics with a single parameter α, via a particular function stated below. This is the case discussed in earlier chapters, that yields the `Fixed-share` algorithm [Herbster and Warmuth, 1998], and for which we stated Theorem 1, and designed the algorithm `Learn-`α.

Lemma 3.3.1 *Theorem 3 (Main Theorem) implies Theorem 1 (Single Parameter Case):*

$$L_T(\alpha) - L_T(\alpha^*) \le (T-1)D(\alpha^*||\alpha) \tag{3.5}$$

Proof From Theorem 3, we have that

$$L_T(\Theta) - L_T(\Theta^*) \leq (T-1)\sum_i \rho_i^* D(\Theta_i^*||\Theta_i) \tag{3.6}$$

In the case we are considering, the distribution over hidden states is the distribution maintained over the experts, and there are n experts. We will now instantiate the parameters in Θ, for the special case we are considering, i.e. HMM's in which the transition probabilities do not vary by expert (i.e. starting state), and are specified by

$$\theta_{ij} = \begin{cases} (1-\alpha) & i = j \\ \frac{\alpha}{n-1} & i \neq j \end{cases} \tag{3.7}$$

Substituting in these parameter settings, and the definition of relative entropy, the bound becomes

$$\begin{aligned}
L_T(\Theta) - L_T(\Theta^*) &\leq (T-1)\sum_{i=1}^{n} \rho_i^* \sum_{j=1}^{n} \theta_{ij}^* \log \frac{\theta_{ij}^*}{\theta_{ij}} && (3.8) \\
&= (T-1)\sum_{i=1}^{n} \rho_i^* \left[\theta_{ii}^* \log \frac{\theta_{ii}^*}{\theta_{ii}} + \sum_{j\neq i}^{n} \theta_{ij}^* \log \frac{\theta_{ij}^*}{\theta_{ij}}\right] && (3.9) \\
&= (T-1)\sum_{i=1}^{n} \rho_i^* \left[(1-\alpha^*) \log \frac{(1-\alpha^*)}{(1-\alpha)} + \sum_{j\neq i}^{n} \frac{\alpha^*}{n-1} \log \frac{\frac{\alpha^*}{n-1}}{\frac{\alpha}{n-1}}\right] && (3.10) \\
&= (T-1)\sum_{i=1}^{n} \rho_i^* \left[(1-\alpha^*) \log \frac{(1-\alpha^*)}{(1-\alpha)} + \alpha^* \log \frac{\alpha^*}{\alpha}\right] && (3.11)
\end{aligned}$$

Within the brackets, we now have the expression for the relative entropy between two Bernoulli variables, and so we continue to attain

$$\begin{aligned}
&= (T-1)\sum_{i=1}^{n} \rho_i^* D(\alpha^*||\alpha) && (3.12) \\
&= (T-1)D(\alpha^*||\alpha)\sum_{i=1}^{n} \rho_i^* && (3.13)
\end{aligned}$$

We attain the final result, since the posterior distribution over states sums to one, and noting that by definition of the transition probabilities in this scenario, the parameter vector Θ can be fully instantiated by instantiating the scalar α. Thus we attain the exact form of Theorem 1, i.e.

$$L_T(\alpha) - L_T(\alpha^*) \leq (T-1)D(\alpha^*||\alpha) \tag{3.14}$$

□

3.4 Implications

The bound we have given covers all online learning algorithms that can be derived as Bayesian updates of any generalized HMM that is homogenous with time. To interpret the bound, note that if we are able to compare with the loss of the best such model computed in hindsight, the relative loss bound is just an average of the relative entropies between rows of the transition matrix of the algorithm and that of the optimal such transition matrix, where the average is weighted by the hindsight frequencies of being in each state. Note that if we are instantiating the value of the bound, meaning we can compare with Θ^*, then we can compute ρ^* from Θ^*, approximately, with better accuracy for larger T. Additionally, since a convex combination can always be upper bounded by its largest term, we note that the relative loss is upper bounded by the largest value of $D(\Theta_i^*||\Theta_i)$, i.e. for the row, i, whose distribution differs most between the algorithm's transition matrix, and the hindsight-optimal one. Thus, even without assuming large T, the bound is feasible to compute.

The bound is tighter than previous relative loss bounds [Herbster and Warmuth, 1998] on online learning algorithms that fall into the general class that our bound covers. Moreover, unlike previous bounds, our bound does not scale with the number of hidden states (or "experts" under that interpretation). The states appear only in the convex combination, which as explained above, is upper bounded by its largest term. Thus increasing the complexity of our model, by increasing the number of hidden states, will not directly affect the loss bound.

It is now possible to interpret our bound just in terms of maximum likelihood parameter estimation in generalized HMMs, as opposed to the online learning framework. It is actually a result about the prediction accuracy of HMM parameter estimation algorithms that do sequential Bayesian updates (i.e. only viewing one observation at a time), versus the prediction accuracy of such models when the parameter estimation is done by viewing the observations in batch. So if all the observations are available, we can bound the regret for doing the parameter estimation sequentially, i.e. in one pass through the data, as opposed to the multiple passes required for the batch version of maximum likelihood parameter estimation in HMMs. This view highlights the convenience of the bound's lack of dependance on the number of hidden states of the HMM.

Chapter 4

Application to Wireless Networks

4.1 Energy/Performance Tradeoff in IEEE 802.11

We applied the `Learn-`α algorithm to an open problem in computer networks to demonstrate its improvement over existing online learning algorithms, and to validate our computation of the optimal learning resolution. The problem is that of managing the tradeoff between energy consumption and performance in wireless nodes of the IEEE 802.11 standard [IEEE, 1999]. Since a node cannot receive packets while asleep, yet maintaining the awake state drains energy, the existing protocol uses a fixed polling time at which a node should wake from the sleep state and poll its neighbors for buffered packets. Polling at fixed intervals, however, does not respond optimally to current network activity. We applied our algorithm to learn the polling time online, and to allow the polling time to change with time, in response to changes in network activity.

Such a problem is clearly an appropriate application for an online learning algorithm, such as the `Fixed-share` algorithm due to [Herbster and Warmuth, 1998]. Additionally however, this problem would benefit from application of the `Learn-`α algorithm. Since we are concerned with wireless, mobile nodes, there is no principled way to set the switching-rate parameter *a priori*, since network activity varies not only over time, but across location, and the location of the mobile node is allowed to change. We can therefore expect an additional benefit from learning the switching-rate.

4.2 Previous Work

Previous work aimed at improving upon the IEEE standard includes Krashinsky and Balakrishnan's [Krashinsky and Balakrishnan, 2002] `Bounded Slowdown` (`BSD`) algorithm which uses an adaptive control loop to change polling time based on network conditions. Several features of this algorithm are deterministic, in that it uses parameterized exploration intervals, and the tradeoff is not managed optimally. A machine learning approach to this problem was proposed by Steinbach, using Reinforcement Learning [Steinbach, 2002]. This approach imposes the assumption that network activity possesses the Markov property, which is an unrealistic assumption.

4.3 Formulation of Algorithm for Application

For this application, we instantiate the experts as deterministic algorithms assuming constant polling times. Thus we use n experts, each corresponding to a different but fixed polling time in milliseconds (ms): $T_i : i \in \{1 \ldots n\}$. The experts form a discretization over the range of possible polling times. In the `Learn-`α algorithm, we maintain m α-experts. The α-experts are `Fixed-share` sub-algorithms that each maintain a distribution over experts, $p_{t,j}(i) = p_{t;\alpha_j}(i)$, which is initialized to the uniform weighting. Here $\alpha_j = j/(m+1)$ is the jth sub-algorithm's definition of the probability that the optimal polling time changes from one discrete T_i to another, and thus $p_{t;\alpha_j}(i)$ is updated according to Equation 2.10. The `Learn-`α algorithm, i.e. the top of the hierarchy, chooses its current polling time T_t, i.e. its prediction of the current amount of time it ought to sleep, using the weighted mean,

$$T_t = \sum_{j=1}^{m} p_t^{top}(j)\, T_{t,j} \tag{4.1}$$

where $T_{t,j}$ is the prediction of the jth sub-algorithm of how long to sleep for, computed as the weighted mean over the experts

$$T_{t,j} = \sum_{i=1}^{n} p_{t,j}(i)\, T_i \tag{4.2}$$

The sub-algorithms do not actually sleep for this amount of time though, as the polling time is chosen by the top-level. The algorithm's polling time is thus

$$T_t = \sum_{j=1}^{m} \sum_{i=1}^{n} p_t^{top}(j)\, p_{t,j}(i)\, T_i \tag{4.3}$$

Using the loss function L that we will define in the next section, we update $p_t^{top}(j)$, the distribution over α-experts, according to Equation 2.29.

This particular problem imposes the constraint that the learning algorithm can only receive observations, and perform learning updates, when it is awake. Thus our subscript t here signifies only wake times, not every time epoch at which bytes might arrive.

4.3.1 Objective Function

To fully specify the algorithm, we need to instantiate the loss function L. The objective at each learning iteration is to choose the polling time T_t that minimizes both the energy usage of the node, and the network latency it introduces by sleeping. We define the loss function so as to reflect the tradeoff inherent in these conflicting goals.

Specifically, we will design a loss function that is directly proportional to appropriate estimates of these two quantities.

In the simple scenario in which we tested our algorithm, the observation, y_t, that the learning algorithm receives upon awakening, is the number of bytes that arrived while it slept during the previous interval. We denote this quantity as I_t, and the length of time that the node slept upon awakening at time t, as T_t. Since the algorithm was not actually operating in an 802.11 wireless node, we did not have a way to directly observe our current energy usage. Instead we modeled it as proportional to $\frac{1}{T_t}$. This is based on the design that the node wakes only after an interval T_t to poll for buffered bytes, and the fact that it consumes less energy when asleep than awake. Additionally there is a constant spike of energy needed to change from sleeping to awake state, so the more times a node polls during a given time horizen, the higher the energy consumption.

We modeled the latency introduced into the network due to sleeping for T_t ms as proportional to I_t. In other words, there was an increased latency for each byte that was buffered during sleep, by the amount of its wait-time in the buffer. Since our learning algorithm does not perform measurement or learning while asleep, and only observes the aggregated number of bytes that arrived while it slept, we have to approximate the amount of total latency its chosen sleep time introduced, based on the individual buffer wait-times of each of the I_t packets. To estimate the average latency that each of the I_t buffered packets would have experienced, without knowledge of the byte arrival times, we can use the maximal entropy assumption. This models all the bytes as arriving at a uniform rate during the sleep interval, T_t, under which assumption the average wait-time per byte would be $\frac{T_t}{2}$. Thus the total latency introduced by sleeping for T_t is approximated by the number of bytes that arrived in that time, times the average wait-time per byte, yielding $\frac{T_t}{2}I_t$.

The form of our proposed loss function is thus:

$$L(t) = \gamma \frac{T_t I_t}{2} + \frac{1}{T_t} \qquad : \gamma > 0 \tag{4.4}$$

In our weight updates however, we apply this loss function to each expert i, indicating the loss that would have accrued had the algorithm used T_i instead of T_t as it's polling time. So the equivalent loss per expert i is:

$$L(i,t) = \gamma \frac{I_t T_i^2}{2T_t} + \frac{1}{T_i} \tag{4.5}$$

where the first term scales I_t to the number of bytes that would have arrived had the node slept for time T_i instead of T_t, under the assumption discussed above that the bytes that arrived at a uniform rate. Note that the objective function is a sum of convex functions and therefore admits a unique minimum.

The parameter $\gamma > 0$ allows for scaling between the units of information and time, as well as the ability to encode a preference for the ratio between energy and latency that the particular node, protocol or host network favors. For the purposes of experiments, we set it through calibration on the same traces with the existing protocol and other algorithms for this domain, so that the polling times attained would lie within the ranges of what had been deemed acceptable to those in the networks community.

Note that this is one of many possible loss functions that are proportional to the tradeoff that must be optimized for this application.

Application of Loss Function

The loss function is applied exactly as in our exposition of the `Learn-`α algorithm (Section 2.3), except only at time-steps when the algorithm is awake. At each wake interval, the loss function specified above is applied to each of the experts, i.e. values of possible polling time T_i, by the sub-algorithms. Using this loss function, we define the probability of the observation given the expert to be $e^{-L(i,t)}$. Then the loss of the jth sub-algorithm, or α-expert, is given by

$$L(p_{t,j}, t) = -\log \sum_{i=1}^{n} p_{t,j}(i)\, e^{-L(i,t)} \tag{4.6}$$

and the loss of the top-level of the hierarchy is given by Equation 2.31.

Form of Loss Function

From the look of the loss function for the experts, it may not seem that this can be derived directly from an HMM, i.e. that $P(y_t|a_{i,t}) = e^{-L(i,t)}$. The concern is that loss is not only a function of t and i, but of T_t, the amount of time that the algorithm slept for upon awakening at time t. And T_t, in turn, is a function of $p_t^{top}(j)$, because at any given learning iteration the amount of time that the algorithm decides to sleep is a function of its current distribution over the α-experts. Note however, that as described in Chapter 2, the form of models from which we can derive online learning algorithms is more general than typical HMM's in that the emissions probabilities can also depend on previous observations, i.e. $P(y_t|a_{i,t}) = P(y_t|a_i, y_1, \ldots, y_{t-1})$. So as long as the p_t^{top} distribution is only a function of previous observations, then our loss function is still derivable from the type of model discussed in previous chapters. This is in fact the case. Each sub-algorithm, or α-expert, has a constant value for its α setting, and its distribution over experts is a function of this value and previous observations. The distribution that the top of the hierarchy maintains over the α-experts is a function of the observations and m, which is a constant, since it is a parameter to the algorithm. This follows from the fact that the exact set of α_j values being used is directly computable from m, and thus coupled with the observations, the losses of the α-experts are computable as well, since the form of their updates is fixed. Thus this loss function could correspond to an HMM, and is congruous with our discussions thus far. Additionally, via our computation of the optimal resolution (Section 2.5), m itself could be computed *a priori* and incorporated into the loss.

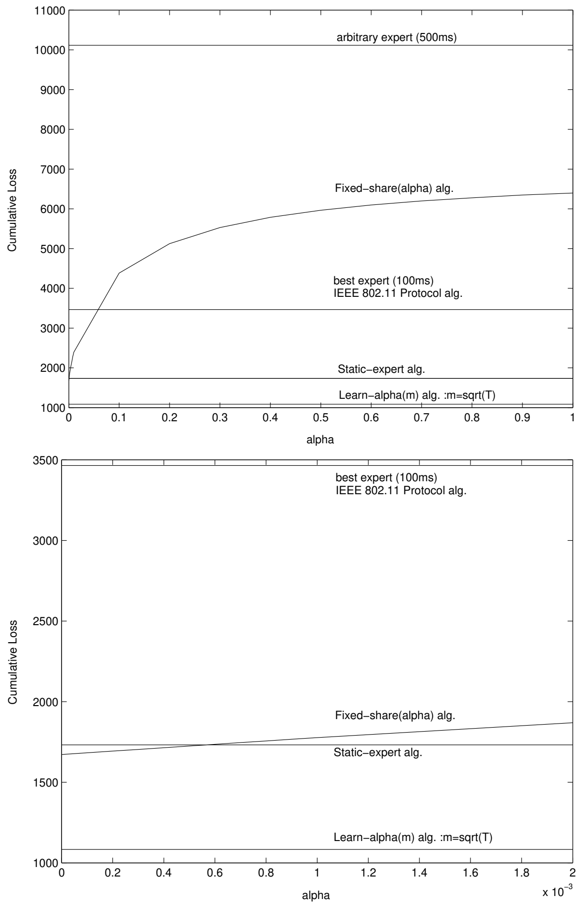

Figure 4-1: Cumulative loss comparison. The curve indicates the cumulative loss of `Fixed-share`(α) along the range of α values. We compare to the cumulative loss on the same trace of the 802.11 protocol, `Static-expert`, and `Learn-`α`(`m^*`)`. The bottom figure zooms in on the first 0.002 of the range for α.

4.4 Experiments

We ran experiments on traces of real network activity, using publicly available data from [Berkeley, 1996], a UC Berkeley home dial-up server that monitored users accessing HTTP files from home. The traces were composed of multiple overlapping HTTP connections, originating from multiple users, all passing through the collection node, over a duration of several days.

Since the traces just provided us with the start and end times of each connection, and the total number of bytes transferred during the connection, for each connection we smoothed the total number of bytes uniformly over $10ms$ intervals spanning its duration. In all the network trace experiment results below, $\gamma = 1.0 \times 10^{-7}$.

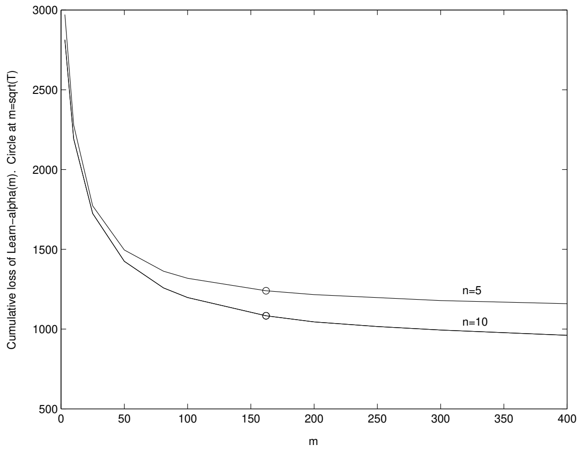

Figure 4-2: Cumulative loss of `Learn-`α`(m)`. Circles at $m = \sqrt{\mathcal{T}} \doteq 162$. By our computation for m^*, after $m = \mathcal{O}(\sqrt{\mathcal{T}})$, cumulative loss should level off.

4.4.1 Results and Analysis

Figure 4-1 compares the cumulative losses of the various algorithms on a 4 hour trace, with observation epochs every $10ms$. This corresponds to approximately 26,100 training iterations for the learning algorithms. Note that in the typical online learning scenario, $\mathcal{T}$, the number of learning iterations, i.e. the time horizen parameter to the loss bounds, is just the number of observation epochs. In this application, the number of training epochs need not match the number of observation epochs, since the application involves sleeping during many observation epochs, and learning is only done upon awakening. Since in these experiments the performance of the learning algorithms (`Static-expert`, `Fixed-share`, and `Learn-`α) are compared by each algorithm using n experts spanning the range of $1000ms$ at regularly spaced intervals of $100ms$, in order to get an *a priori* estimate of $\mathcal{T}$, we assume a mean sleep interval of $550ms$, i.e. the mean of the experts.

It is important to note however that the polling times of each of the runs graphed are set by the algorithm being run, and so for the learning algorithms, the polling times change online. Another possible caveat of this experimental evaluation is that the experts incurr losses without their predicted polling times actually being realized, and the cumulative loss of each learning algorithm is in turn computed as a function of these expert losses. The form of the expert loss function addresses this by taking the proportion of incoming bytes that would have arrived had that expert's polling time been used, although the assumption of a uniform byte arrival rate is introduced. Additionally, this issue pertains to each of the three online learning algorithms being compared, with the only significant difference being how the distribution over experts is updated. This lends credibility to making this sort of comparison.

Since we calibrated our objective function (by choice of γ), to match the level of this tradeoff deemed acceptable in the networks community, it is no surprise that the deterministic setting of $100ms$, used by the existing IEEE 802.11 protocol, was the best expert. `Static-expert`, the online learning algorithm with relative loss bounds against the best fixed expert, achieved lower cumulative loss however, since it can predict with the best linear combination of the experts, as apposed to any one of their values, i.e., it can choose the optimal smoothed value over the desired range, as opposed to being limited by the discretization imposed by the experts.

On this particular trace, the optimal α for `Fixed-share` turns out to be extremely low, and so for most settings of α, one would be better off using a `Static-expert` model. Yet as the second graph shows, there is a value for α below which it is beneficial to use `Fixed-share`, which lends validity to our fundamental goal of being able to quantify the level of non-stationarity of a process, in order to better model it. This also highlights the strength of the Learn-α algorithm, since without prior knowledge of the stochastic process to be observed, there is no optimal way by which to set α.

Since Learn-α tracks the performance of `Fixed-share` with the best α, we might expect that there exists an α for which Fixed-share(α)'s performance is better than that of Learn-α(m^*). We did not find this to be the case. This could be an artifact of the resolution of our experiments with `Fixed-share`, but is also explainable using the logic for why `Static-expert` beat the best expert. Learn-α can harness the predictions of the best linear combination of its sub-algorithms, each of which operate with a fixed α. Thus it has the potential to do better than any given one.

Figure 4-2 shows the cumulative loss of Learn-α as a function of m, the resolution at which to learn α. We see that choosing m proportional to $\sqrt{\mathcal{T}}$, where $\mathcal{T}$ is the number of learning iterations, matches the point in the curve beyond which one cannot significantly reduce loss by increasing m. As expected, the performance of the algorithm levels off after the optimal m that we can compute *a priori*. Our results also verify that the optimal resolution m is not a function of the number of experts n.

Chapter 5

Conclusion

5.1 Summary

In summary, first we re-derived existing online learning algorithms as simple Bayesian estimation algorithms, updating appropriately defined probabilistic models. We then proved a substantially simpler relative loss bound for such online learning algorithms than previously in the literature, applicable for any sequence of observations. The loss bound formed the basis for an analytic expression of the optimal resolution at which to learn the switching-rate parameter. We analyzed the results using tools from Information Theory. We then generalized the proof of the bound to apply to richer models of non-stationary sequences, extending the result from modeling the switching-rate with a single scalar, to using a matrix of parameters to model the switching dynamics of the sequence. We gave an algorithm to learn the switching-rate parameter online, simultaneous to learning the target concept. The optimal learning resolution we computed was shown to be appropriate in the context of an application of our new algorithm to an open problem in wireless networks.

5.2 Future Work

Aside from the evident extensions which can already be unravelled from our general result, i.e. specifiying particular transition matrices, and instantiating the specific online learning algorithms they yield, there are several interesting directions for future work. A more general result may be possible, where the form of graphical model need not be homogenous with time, or where the structure admits more parameters than we currently treat. We have not yet extended our approach of learning α, and computing the optimal resolution for learning α, to learning the whole Θ matrix, but that may be feasible as well. Further analysis of the current result, especially in light of existing work in Information Theory, may yield a way to prove that our optimized hierarchical loss bound is tight, i.e. that it is also the lower bound on the relative loss of online learning algorithms that simultaneously learn the switching-rate, achievable by the `Learn-`α algorithm we presented.

5.3 Acknowledgements

I am greatly indebted to my research advisor, Tommi Jaakkola. I would like to thank him not only for his brilliant, elegant answers to all my questions, and for many patient explanations, but for setting an inspiring example of successfully combining wisdom, piercing intelligence, enthusiasm, and kindness.

I would like to thank Jessica Howe, Jason Rennie, and Ronny Krashinsky for various forms of help with the application to 802.11 wireless energy management. I have also benefited from discussions with Hari Balakrishnan, Huizhen (Janey) Yu, Adam Kalai, and Jaeyeon Jung.

Appendix A

Proof of Main Theorem

We consider an arbitrary homogeneous HMM, parameterized by Θ, where θ_{ij} is the probability of transitioning from state i to state j from one time interval to the next. Thus stated, this implies that $\sum_j \theta_{ij} = 1, \forall i$.

We first analyze the cumulative loss achieved by the online learning algorithm derived from our HMM formulation, i.e.,

$$L_T(\Theta) = \sum_{t=1}^{T} L(p_t, t) \tag{A.1}$$

Lemma A.0.1 *This cumulative loss (A.1) can be expanded to the following form:*

$$L_T(\Theta) = -\log \left[\sum_{i_1,\ldots,i_T} p_1(i_1) e^{-L(i_1,1)} \prod_{t=2}^{T} e^{-L(i_t,t)} P(i_t|i_{t-1}, \Theta) \right]$$

where $\{i_1, \ldots, i_T\}$ is the set of all possible state sequences.

Proof Using the formulations introduced in Chapter 2, we can expand Equation A.1 as follows.

$$\begin{aligned}
L_T(\Theta) &= -\sum_{t=1}^{T} \log \sum_{i=1}^{n} p_t(i) P(y_t|a_{i,t}) && \text{(A.2)}\\
&= -\sum_{t=1}^{T} \log \sum_{i=1}^{n} p_t(i) P(y_t|a_i, y_1, \ldots, y_{t-1}) && \text{(A.3)}\\
&= -\sum_{t=1}^{T} \log P(y_t|y_1, \ldots, y_{t-1}) && \text{(A.4)}\\
&= -\log\, p_1(y_1) \prod_{t=2}^{T} P(y_t|y_1, \ldots, y_{t-1}) && \text{(A.5)}\\
&= -\log P(y_1, \ldots, y_T) && \text{(A.6)}
\end{aligned}$$

We can compute the probability of the joint observation sequence by summing over all possible hidden state sequences $\{i_1, \ldots, i_T\}$, the probability of the observation sequence given that hidden state sequence, weighted by the probability of that hidden state sequence.

$$
\begin{aligned}
& L_T(\Theta) = -\log P(y_1, \ldots, y_T) \\
= \;& -\log\left[\sum_{i_1,\ldots,i_T} P(y_1,\ldots,y_T|i_1,\ldots,i_T)P(i_1,\ldots,i_T|\Theta)\right] && \text{(A.7)} \\
= \;& -\log\left[\sum_{i_1,\ldots,i_T} p_1(i_1)P(y_1|i_1,\ldots,i_T)\prod_{t=2}^{T} P(y_t|i_1,\ldots,i_T,y_1,\ldots,y_{t-1})P(i_t|i_1,\ldots,i_{t-1},\Theta)\right] \\
= \;& -\log\left[\sum_{i_1,\ldots,i_T} p_1(i_1)P(y_1|i_1)\prod_{t=2}^{T} P(y_t|i_t,y_1,\ldots,y_{t-1})P(i_t|i_{t-1},\Theta)\right] && \text{(A.8)} \\
= \;& -\log\left[\sum_{i_1,\ldots,i_T} p_1(i_1)e^{-L(i_1,1)}\prod_{t=2}^{T} e^{-L(i_t,t)}P(i_t|i_{t-1},\Theta)\right] && \text{(A.9)}
\end{aligned}
$$

□

We now seek to bound the cumulative loss $L_T(\Theta)$ relative to the cumulative loss the same algorithm running with Θ^*, i.e. the cumulative loss minimizing setting of this matrix of parameters (computed in hindsight). i.e., $\Theta^* : L_T(\Theta^*) = \min_\Theta L_T(\Theta)$.

Now, define $\vec{s} = \{i_1, \ldots, i_T\}$

$$
\begin{aligned}
P(\vec{s}|\Theta) &= p_1(i_1)\prod_{t=2}^{T} P(i_t|i_{t-1},\Theta) && \text{(A.10)} \\
\phi(\vec{s}) &= \prod_{t=1}^{T} e^{-L(i_t,t)} && \text{(A.11)}
\end{aligned}
$$

With this notation we can write

$$
\begin{aligned}
L_T(\Theta) - L_T(\Theta^*) &= -\log\left[\sum_{\vec{s}} \phi(\vec{s})P(\vec{s}|\Theta)\right] + \log\left[\sum_{\vec{r}} \phi(\vec{r})P(\vec{r}|\Theta^*)\right] && \text{(A.12)} \\
&= -\log\left[\frac{\sum_{\vec{s}} \phi(\vec{s})P(\vec{s}|\Theta)}{\sum_{\vec{r}} \phi(\vec{r})P(\vec{r}|\Theta^*)}\right] && \text{(A.13)} \\
&= -\log\left[\frac{\sum_{\vec{s}} \phi(\vec{s})P(\vec{s}|\Theta^*)\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)}}{\sum_{\vec{r}} \phi(\vec{r})P(\vec{r}|\Theta^*)}\right] && \text{(A.14)} \\
&= -\log\left[\sum_{\vec{s}} \left(\frac{\phi(\vec{s})P(\vec{s}|\Theta^*)}{\sum_{\vec{r}} \phi(\vec{r})P(\vec{r}|\Theta^*)}\right)\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)}\right] && \text{(A.15)}
\end{aligned}
$$

$$= \; -\log\left[\sum_{\vec{s}} Q(\vec{s}|\Theta^*)\,\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)}\right] \tag{A.16}$$

where $Q(\vec{s}|\Theta^*)$ is a distribution over the choices of experts along the sequence and also summarizes all the information about the observation sequence together with Θ^*.

We now attempt to write $P(\vec{s}|\Theta)/P(\vec{s}|\Theta^*)$ more compactly. The probability of any sequence of hidden states is now defined by counting the number of transitions.

$$\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)} = \frac{p_1(i_1)\prod_{i=1}^n \prod_{j=1}^n (\theta_{ij})^{n_{ij}(\vec{s})}}{p_1(i_1)\prod_{i=1}^n \prod_{j=1}^n (\theta^*_{ij})^{n_{ij}(\vec{s})}} \tag{A.17}$$

where $n_{ij}(\vec{s})$ is the number of transitions from state i to state j, in a sequence $\vec{s} = i_1, \ldots, i_T$.

Now we note that $\sum_j n_{ij}(\vec{s})$, is the number of times the sequence was in state i, except at the final time-step. Thus $\sum_j n_{ij}(\vec{s}) = (T-1)\hat{\rho}_i(\vec{s})$, where $\hat{\rho}_i(\vec{s})$ is the empirical estimate, from the sequence $\vec{s}$, of the marginal probability of being in state i, at any time-step except the final one. Thus we can write

$$n_{ij}(\vec{s}) \;=\; T'\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s}) \tag{A.18}$$

where $T' = T - 1$, and $\hat{\theta}_{ij}(\vec{s}) = n_{ij}(\vec{s})/\sum_j n_{ij}(\vec{s})$ is the empirical estimate of the probability of that particular state transition, on the basis of $\vec{s}$.

We can now simplify to attain

$$\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)} \;=\; \prod_{i=1}^n \prod_{j=1}^n \left(\frac{\theta_{ij}}{\theta^*_{ij}}\right)^{T'\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})} \tag{A.19}$$

$$= \; \exp\left\{T'\sum_{i=1}^n \sum_{j=1}^n \hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\log(\frac{\theta_{ij}}{\theta^*_{ij}})\right\} \tag{A.20}$$

Substituting the result back into Equation A.16, yields

$$L_T(\Theta) - L_T(\Theta^*) \;=\; -\log\left[\sum_{\vec{s}} Q(\vec{s}|\Theta^*)\,\frac{P(\vec{s}|\Theta)}{P(\vec{s}|\Theta^*)}\right] \tag{A.21}$$

$$= \; -\log\left[\sum_{\vec{s}} Q(\vec{s}|\Theta^*)\,\exp\left\{T'\sum_{i=1}^n \sum_{j=1}^n \hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\log(\frac{\theta_{ij}}{\theta^*_{ij}})\right\}\right] \tag{A.22}$$

Note that this is an exact expression for the difference between the cumulative losses. The expression depends on the actual sequence of observations only through the distribution Q and Θ^*. We can find an upper bound on this difference by maximizing the above result over all valid choices for the distribution Q. The only constraint we impose is that Θ^* has to remain the optimal setting of the matrix of parameters. We can express this constraint as

$$\frac{d}{d\Theta} L_T(\Theta)_{\Theta=\Theta^*} = \vec{0} \tag{A.23}$$

Lemma A.0.2 *Constraint (A.23) is equivalent to*

$$\sum_{\vec{s}} Q(\vec{s}|\Theta^*)\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s}) = \rho_i^*\theta_{ij}^* \tag{A.24}$$

where $\rho_i^* = E_{\vec{s}\sim Q}\,[\hat{\rho}_i(\vec{s})]$.

Proof We can start by optimizing Equation A.22 with respect to Θ, since the $L_T(\Theta^*)$ term in this equation is constant with respect to Θ. So our objective function is

$$J(\Theta) \quad = \quad -\log\left[E_{\vec{s}\sim Q}\ \exp\left\{T'\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\log(\frac{\theta_{ij}}{\theta_{ij}^*})\right\}\right] \tag{A.25}$$

The constraint given, is equivalent to

$$\frac{d}{d\theta_{ij}} L_T(\Theta)_{\Theta=\Theta^*} = 0 \quad \forall i,j \tag{A.26}$$

In our optimization, we will encode the constraint on Θ, that $\forall i$, $\sum_j \theta_{ij} = 1$, using a Lagrange multiplier, λ_i. So we fix i, and evaluate

$$\frac{d}{d\theta_{ij}}\left[J(\Theta) + \lambda_i(1-\sum_{j'}\theta_{ij'})\right]_{\Theta=\Theta^*} \tag{A.27}$$

$$= \frac{d}{d\theta_{ij}} J(\Theta)_{\Theta=\Theta^*} - \lambda_i \tag{A.28}$$

$$= \frac{d}{d\theta_{ij}}\left[-\log\left[E_{\vec{s}\sim Q}\ \exp\left\{T'\sum_{i'=1}^{n}\sum_{j'=1}^{n}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta_{i'j'}^*})\right\}\right]\right]_{\Theta=\Theta^*} - \lambda_i \tag{A.29}$$

$$= -\frac{d}{d\theta_{ij}}\left[E_{\vec{s}\sim Q}\ \exp\left\{T'\sum_{i'=1}^{n}\sum_{j'=1}^{n}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta_{i'j'}^*})\right\}\right]_{\Theta=\Theta^*} - \lambda_i \tag{A.30}$$

since the denominator yielded from taking the derivative of the log term, is equal to one, for $\Theta = \Theta^*$. Continuing,

$$
\begin{aligned}
&= -E_{\vec{s}\sim Q}\left[\frac{d}{d\theta_{ij}}\exp\left\{T'\sum_{i'=1}^{n}\sum_{j'=1}^{n}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta^*_{i'j'}})\right\}\right]_{\Theta=\Theta^*} - \lambda_i && \text{(A.31)}\\
&= -E_{\vec{s}\sim Q}\left[\exp\left\{T'\sum_{i'j'}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta^*_{i'j'}})\right\}T'\frac{d}{d\theta_{ij}}\sum_{i'j'}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta^*_{i'j'}})\right]_{\Theta=\Theta^*}\\
&\quad -\lambda_i && \text{(A.32)}\\
&= -E_{\vec{s}\sim Q}\left[\exp\left\{T'\sum_{i'j'}\hat{\rho}_{i'}(\vec{s})\hat{\theta}_{i'j'}(\vec{s})\log(\frac{\theta_{i'j'}}{\theta^*_{i'j'}})\right\}T'\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\frac{1}{\theta_{ij}}\right]_{\Theta=\Theta^*} - \lambda_i && \text{(A.33)}\\
&= -\frac{T'}{\theta^*_{ij}}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] - \lambda_i && \text{(A.34)}
\end{aligned}
$$

Now setting the derivative (Equation A.27) equal to zero, we attain

$$
\begin{aligned}
0 &= -\frac{T'}{\theta^*_{ij}}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] - \lambda_i && \text{(A.35)}\\
\theta^*_{ij} &= -\frac{T'}{\lambda_i}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] && \text{(A.36)}
\end{aligned}
$$

This holds $\forall j$. Now we solve for λ_i in the constraint, by summing both sides on j.

$$
\begin{aligned}
\sum_j \theta^*_{ij} &= -\sum_j \frac{T'}{\lambda_i}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] && \text{(A.37)}\\
1 &= -\frac{T'}{\lambda_i}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\sum_j\hat{\theta}_{ij}(\vec{s})\right] && \text{(A.38)}\\
\lambda_i &= -T'E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\right] && \text{(A.39)}
\end{aligned}
$$

We now define $\rho^*_i = E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\right]$, and plug in λ_i to solve Equation A.36, yielding

$$
\begin{aligned}
\theta^*_{ij} &= \frac{-T'}{-T'\rho^*_i}E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] && \text{(A.40)}\\
E_{\vec{s}\sim Q}\left[\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\right] &= \rho^*_i\theta^*_{ij} && \text{(A.41)}
\end{aligned}
$$

which holds $\forall i, j$.

□

We now wish to maximize Equation A.25 with respect to Q and subject to the mean constraint (A.41). Since $-\log(\cdot)$ is a convex function, we can upper bound the expression by moving the expectation outside the logarithm:

$$
\begin{aligned}
-\log\left[E_{\vec{s}\sim Q}\,\exp\left\{T'\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\log(\frac{\theta_{ij}}{\theta^*_{ij}})\right\}\right] \leq \\
E_{\vec{s}\sim Q}\left[-T'\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\rho}_i(\vec{s})\hat{\theta}_{ij}(\vec{s})\log(\frac{\theta_{ij}}{\theta^*_{ij}})\right]
\end{aligned}
\tag{A.42}
$$

$$
= \quad -T'\sum_{i=1}^{n}\sum_{j=1}^{n}\rho^*_i\theta^*_{ij}\log(\frac{\theta_{ij}}{\theta^*_{ij}}) \tag{A.43}
$$

$$
= \quad T'\sum_{i=1}^{n}\rho^*_i\sum_{j=1}^{n}\theta^*_{ij}\log(\frac{\theta^*_{ij}}{\theta_{ij}}) \tag{A.44}
$$

$$
= \quad T'\sum_{i=1}^{n}\rho^*_i D(\Theta^*_i\|\Theta_i) \tag{A.45}
$$

where we have used the mean constraint and the definition of the relative entropy $D(\cdot\|\cdot)$ for discrete distributions. We have thus shown that

$$
L_T(\Theta) - L_T(\Theta^*) \leq (T-1)\sum_{i=1}^{n}\rho^*_i D(\Theta^*_i\|\Theta_i) \tag{A.46}
$$

This completes the proof.

$\square$

Bibliography

[Auer *et al.*, 1995] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[Berger, 1985] James O. Berger. *Statistical decision theory and Bayesian analysis.* Springer-Verlag, New York, 1985.

[Berkeley, 1996] Berkeley. UC Berkeley home IP web traces. In *http://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html*, 1996.

[Blum *et al.*, 1999] Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *IEEE 40th Annual Symposium on Foundations of Computer Science*, page 450, New York, New York, October 1999.

[Cover and Thomas, 1991] Thomas Cover and Joy Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[Feder and Merhav, 1996] Meir Feder and Neri Merhav. Hierarchical universal coding. *IEEE Transactions on Information Theory*, 42(5):1354–1364, 1996.

[Feder *et al.*, 1992] Meir Feder, Neri Merhav, and Michael Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38(4):1258–1270, 1992.

[Foster and Vohra, 1999] Dean P. Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:7–35, 1999.

[Freund and Schapire, 1999] Yoav Freund and Robert Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[Haussler *et al.*, 1998] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Trans. on Information Theory*, 44(5):1906–1925, 1998.

[Helmbold *et al.*, 1996] David P. Helmbold, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. On-line portfolio selection using multiplicative updates. In *International Conference on Machine Learning*, pages 243–251, 1996.

[Herbster and Warmuth, 1998] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[IEEE, 1999] IEEE. Computer society LAN MAN standards committee. In *IEEE Std 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*, August 1999.

[Krashinsky and Balakrishnan, 2002] Ronny Krashinsky and Hari Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. In *MobiCom 2002*, Atlanta, GA, September 2002.

[Littlestone and Warmuth, 1989] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[Rissanen, 1984] Jorma Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, IT-30(4):629–636, 1984.

[Rissanen, 1986] Jorma Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986.

[Rissanen, 1989] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore, 1989.

[Steinbach, 2002] Carl Steinbach. A reinforcement-learning approach to power management. In *AI Technical Report, M.Eng Thesis*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May 2002.

[Ziv and Lempel, 1978] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.