Multitask Learning with Expert Advice

Jacob Duncan Abernethy
Peter Bartlett
Alexander Rakhlin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-20
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-20.html

January 28, 2007

Acknowledgement

We would like to thank Manfred Warmuth for his knowledge of prediction algorithms and helpful discussions on mixing priors. We would like to thank Alistair Sinclair for his expertise on the MCMC methods. We gratefully acknowledge the support of DARPA under grant FA8750-05-20249.

Multitask Learning with Expert Advice

Jacob Abernethy[1], Peter Bartlett[1,2], and Alexander Rakhlin[1]

[1] Department of Computer Science, UC Berkeley
[2] Department of Statistics, UC Berkeley
{jake,bartlett,rakhlin}@cs.berkeley.edu

Abstract. We consider the problem of prediction with expert advice in the setting where a forecaster is presented with several online prediction tasks. Instead of competing against the best expert separately on each task, we assume the tasks are related, and thus we expect that a few experts will perform well on the entire set of tasks. That is, our forecaster would like, on each task, to compete against the best expert *chosen from a small set of experts*. While we describe the "ideal" algorithm and its performance bound, we show that the computation required for this algorithm is as hard as computation of a matrix permanent. We present an efficient algorithm based on mixing priors, and prove a bound that is nearly as good for the sequential task presentation case. We also consider a harder case where the task may change arbitrarily from round to round, and we develop an efficient approximate randomized algorithm based on Markov chain Monte Carlo techniques.

1 Introduction

A general model of sequential prediction with expert advice is the following. A *forecaster* is given the following *task*: make a sequence of predictions given access to a number of *experts*, where each expert makes its own prediction at every round. The forecaster combines the predictions of the experts to form its own prediction, taking into account each expert's past performance. He then learns the true outcome and suffers some loss based on the difference between the true outcome and its prediction. The goal of the forecaster, in the cumulative sense, is to predict not much worse than the single best expert. This sequence prediction problem has been widely studied in recent years. We refer the reader to the excellent book of Cesa-Bianchi and Lugosi [1] for a comprehensive treatment of the subject.

We consider an extension of this framework where a forecaster is presented with several prediction tasks. The most basic formulation, which we call the *sequential multitask problem* is the following: the forecaster is asked to make a sequence of predictions for task one, then another sequence of predictions for task two, and so on, and receives predictions from a constant set of experts on every round. A more general formulation, which we consider later in the paper, is the *shifting multitask problem*: on every round, the forecaster is asked to make a prediction for some task, and while the task is known to the forecaster, it may change arbitrarily from round to round.

The multitask learning problem is fundamentally a sequence prediction problem, yet we provide the forecaster with extra information for each prediction, namely the task to which this round belongs. This extra knowledge could be quite valuable. In particular, the forecaster may have observed that certain experts have performed well on this task while poorly on others. Consider, for example, an investor that, on each day, would like to make a sequence of trades for a particular stock, and has a selection of trading strategies available to him. We may consider each day a separate prediction task. The behavior of the stock will be quite related from one day to the next, even though the optimal trading strategy may change. How can the investor perform as well as possible on each day, while still leveraging information from previous days?

As the above example suggests, we would like to take advantage of *task relatedness*. This idea is quite general and in the literature several such frameworks have been explored [2–7]. In this paper, we attempt to capture the following intuitive notion of relatedness: experts that perform well on one task are more likely to perform well on others. Of course, if the same best expert is shared across several tasks, then we should not expect to find so many best experts. We thus consider the following problem: given a "small" m, design a multitask forecaster that performs well relative to the best m-sized subset of experts.

The contribution of this paper is the following. We first introduce a novel multitask learning framework within the "prediction with expert advice" model. We then show how techniques developed by Bousquet and Warmuth [8] can be applied in this new setting. Finally, we develop a randomized prediction algorithm, based on an approximate Markov chain Monte Carlo method, that overcomes the hardness of the corresponding exact problem, and demonstrate empirically that the Markov chain mixes rapidly.

We begin in Section 2 by defining the online multitask prediction problem and notation. In Section 3 we provide a reduction from the multitask setting to the single task setting, yet we also show that computing the prediction is as hard as computing a matrix permanent. In Section 4, however, we provide an efficient solution for the *sequential multitask problem*. We attack the more general *shifting multitask problem* in Section 5, and we present the MCMC algorithm and its analysis.

2 Formal Setting

First, we describe the "prediction with expert advice" setting. A *forecaster* must make a sequence of predictions for every round $t = 1, 2, 3, \ldots, T$. This forecaster is given access to a set of N "experts". At every round t, expert i makes prediction $f_i^t \in [0, 1]$. The forecaster is given access to $\boldsymbol{f}^t := (f_1^t, \ldots, f_N^t)$ and then makes a prediction $\hat{p}^t \in [0, 1]$. Finally, the outcome $y^t \in \{0, 1\}$ is revealed, expert i suffers $\ell_i^t := \ell(f_i^t, y^t)$, and the forecaster suffers $\ell(\hat{p}^t, y^t)$, where ℓ is a loss function that is convex in its first argument. We consider the cumulative loss of the forecaster, $\hat{L}^T := \sum_{t \le T} \ell(\hat{p}^t, y^t)$, relative to the cumulative loss of each expert, $L_i^T := \sum_{t \le T} \ell(f_i^t, y^t)$.

In the multitask setting we have additional structure on the order of the sequence. We now assume that the set of rounds is partitioned into K "tasks" and the forecaster knows K in advance. On round t, in addition to learning the predictions $\boldsymbol{f}^t$, the forecaster also learns the *task number* $\kappa(t) \in [K] := \{1, 2, \ldots, K\}$. For convenience, we also define $\tau(k) = \{t \in [T] : \kappa(t) = k\}$, the set of rounds where the task number is k. After T rounds we record the cumulative loss, $L_{k,i}^T$ of expert i for task k, defined as $L_{k,i}^T := \sum_{t \in \tau(k)} \ell_i^t$.

As described in the introduction, we are interested in the *sequential multitask problem*, where we assume that the subsequences $\tau(k)$ are contiguous. We also consider the more general case, the *shifting multitask problem*, where the task may change arbitrarily from round to round. For the remainder of Section 2 and section 3, however, we need not make any assumptions about the sequence of tasks presented.

2.1 The Multitask Comparator Class

We now pose the following question: what should be the goal of the forecaster in this multitask setting? Typically, in the single task expert setting, we compare the performance of the forecaster relative to that of the *best* expert in our class. This is quite natural: we should expect the forecaster to predict only as well as the best information available. Thus, the forecaster's goal is minimize *regret*, $\hat{L}^T - \min_{i=1}^N L_i^T$. We will call the quantity $L_*^T := \min_i L_i^T$ the *comparator*, since it is with respect to this that we measure the performance of the forecaster.

Following this, we might propose the following as a multitask comparator, which we will call the *unrelated* comparator: $L_*^T := \sum_{k=1}^K \min_i L_{k,i}^T$. Here, the forecaster's goal is to minimize loss relative to the best expert on task one, plus loss relative to the best expert on task two, and so on. However, by minimizing the sum over tasks, the forecaster may as well minimize each separately, thus considering every task as independent of the rest.

Alternatively, we might propose a second, which we will call the *fully related* comparator: $L_*^T := \min_i \sum_{k=1}^K L_{k,i}^T$. Here, the forecaster competes against the best expert on all tasks, that is, the single best expert. The forecaster can simply ignore the task number and predict as though there were only one task.

These two potential definitions represent ends of a spectrum. By employing the unrelated comparator, we are inherently expecting that each task will have a different best expert. With the fully related comparator, we expect that one expert should perform well on all tasks. In this paper, we would like to choose a comparator which captures the more general notion of "partial relatedness" across tasks. We propose the following: the goal of the forecaster is to perform as well as the best choice of experts from a *small set*. More precisely, given a positive integer $m \leq N$ as a parameter, letting $\mathcal{S}_m := \{S \subset [N] : |S| = m\}$ be the set of m-sized subsets of experts, we define our comparator as

$$L_*^T := \min_{S \in \mathcal{S}_m} \sum_{k=1}^K \min_{i \in S} L_{i,k}^T. \tag{1}$$

Notice that, for the choice $m = N$, we obtain the unrelated comparator as described above; for the choice $m = 1$, we obtain the fully related comparator.

2.2 Taking Advantage of Task Relatedness

There is a benefit in competing against the constrained comparator described in (1). We are interested in the case when m is substantially smaller than K. By searching for only the m best experts, rather than K, the forecaster may learn faster by leveraging information from other tasks. For example, even when the forecaster arrives at a task for which it has seen no examples, it already has some knowledge about which experts are likely to be amongst the best $S \in \mathcal{S}_m$.

In this paper, we are interested in designing forecasters whose performance bound has the following form,

$$\hat{L}^T \leq c_1 \left(\min_{S \in \mathcal{S}_m} \sum_{k=1}^{K} \min_{i \in S} L_{i,k}^T \right) + c_2 \left(K \log m + m \log \frac{N}{m} \right), \tag{2}$$

where c_1 and c_2 are constants. This bound has two parts, the loss term on the left and the complexity term on the right, and there is an inherent trade-off between these two terms given a choice of m. Notice, for $m = 1$, the complexity term is only $c_2 \log N$, although there may not be a single good expert. On the other hand, when $m = N$, the loss term will be as small as possible, while we pay $c_2 K \log N$ to find the best expert separately for each task. Intermediate choices of m result in a better trade-off between the two terms whenever the tasks are related, which implies a smaller bound.

3 A Reduction to the Single Task Setting

Perhaps the most well-known prediction algorithm in the single-task experts setting, as described at the beginning of Section 2, is the (Exponentially) Weighted Average Forecaster, also known as Randomized Weighted Majority. On round t, the forecaster has a table of the cumulative losses of each expert, $L_1^t, \ldots, L_N^t$, a learning parameter η, and receives the predictions $\boldsymbol{f}^t$. The forecaster computes a weight $w_i^t := e^{-\eta L_i^t}$ for each i, and predicts $\hat{p}^t := \frac{\sum_i w_i^t f_i^t}{\sum_i w_i^t}$. He receives the outcome y^t, suffers loss $\ell(\hat{p}^t, y^t)$, and updates $L_i^{t+1} \leftarrow L_i^t + \ell(f_i^t, y^t)$ for each i.

This simple yet elegant algorithm has the following bound,

$$\hat{L}^t \leq c_\eta \left(\min_i L_i^t + \eta^{-1} \log N \right), \tag{3}$$

where[3] $c_\eta = \frac{\eta}{1 - e^{-\eta}}$ tends to 1 as $\eta \to 0$. The curious reader can find more details of the Weighted Average Forecaster and relative loss bounds in [1]. We will appeal to this algorithm and its loss bound throughout the paper.

[3] Depending on the loss function, tighter bounds can be obtained.

3.1 Weighted Average Forecaster on "Hyperexperts"

We now define a reduction from the multitask experts problem to the single task setting and we immediately get an algorithm and a bound. Unfortunately, as we will see, this reduction gives rise to a computationally infeasible algorithm, and in later sections we will discuss ways of overcoming this difficulty.

We will now be more precise about how we define our comparator class. In Section 2.1, we described our comparator by choosing the best subset $S \in \mathcal{S}_m$ and then, for each task, choosing the best expert in this subset. However, this is equivalent to assigning the set of tasks to the set of experts such that at most m experts are used. In particular, we are interested in maps $\pi : [K] \to [N]$ such that $\text{img}(\pi) := \{\pi(k) : k \in [K]\}$ has size $\leq m$. Define

$$\mathcal{H}_m := \{\pi : [K] \to [N] \text{ s.t. } \text{img}(\pi) \leq m\}.$$

Given this new definition, we can now rewrite our comparator,

$$L_*^T = \min_{S \in \mathcal{S}_m} \sum_{k=1}^{K} \min_{i \in S} L_{i,k}^T = \min_{\pi \in \mathcal{H}_m} \sum_{k=1}^{K} L_{k,\pi(k)}^T. \tag{4}$$

More importantly, this new set $\mathcal{H}_m$ can now be realized as itself a set of experts. For each $\pi \in \mathcal{H}_m$, we can associate a "hyperexpert" to π. So as not to be confused, we now also use the term "base expert" for our original class of experts. On round t, we define the prediction of hyperexpert π to be $f_{\pi(\kappa(t))}^t$, and thus the loss of this hyperexpert is exactly $\ell_{\pi(\kappa(t))}^t$. We can define the cumulative loss of this hyperexpert in the natural way,

$$L_\pi^T := \sum_{t=1}^{T} \ell_{\pi(\kappa(t))}^t = \sum_{k=1}^{K} L_{k,\pi(k)}^T.$$

We may now apply the Weighted Average Forecaster using, as our set of experts, the class $\mathcal{H}_m$. Assume we are given a learning parameter $\eta > 0$. We maintain a $K \times N$ matrix of weights $[w_{k,i}^t]$ for each base expert and each task. For $i \in [N]$ and $k \in [K]$, let $w_{k,i}^t := \exp(-\eta L_{k,i}^t)$. We now define weight v_π^t of a hyperexpert π at time t to be $v_\pi^t := \exp(-\eta L_\pi^t) = \prod_{k=1}^{K} w_{k,\pi(k)}^t$. This gives an explicit formula for the prediction of the algorithm at time t,

$$\hat{p}^t = \frac{\sum_{\pi \in \mathcal{H}_m} v_\pi^t f_{\pi(\kappa(t))}^t}{\sum_{\pi \in \mathcal{H}_m} v_\pi^t} = \frac{\sum_{\pi \in \mathcal{H}_m} \left(\prod_{k=1}^{K} w_{k,\pi(k)}^t \right) f_{\pi(\kappa(t))}^t}{\sum_{\pi \in \mathcal{H}_m} \prod_{k'=1}^{K} w_{k',\pi(k')}^t}. \tag{5}$$

The prediction f_i^t will be repeated many times, and thus we can factor out the terms where $\pi(\kappa(t)) = i$. Let $\mathcal{H}_m^{k,i} \subset \mathcal{H}_m$ be the assignments π such that $\pi(k) = i$, and note that, for any k, $\bigcup_{i=1}^{N} \mathcal{H}_m^{k,i} = \mathcal{H}_m$. Letting

$$u_{k,i}^t := \frac{\sum_{\pi \in \mathcal{H}_m^{k,i}} \prod_{k'=1}^{K} w_{k',\pi(k')}^t}{\sum_{\pi \in \mathcal{H}_m} \prod_{k'=1}^{K} w_{k',\pi(k')}^t} \qquad \text{gives} \qquad \hat{p}^t = \sum_{i=1}^{N} u_{\kappa(t),i} \cdot f_i^t.$$

We now have an exponentially weighted average forecaster that predicts given the set of hyperexperts $\mathcal{H}_m$. In order to obtain a bound on this algorithm we still need to determine the size of $\mathcal{H}_m$. The proof of the next lemma is omitted.

Lemma 1. *Given $m < K$, it holds that $\binom{N}{m} m^{K-m} m! \leq |\mathcal{H}_m| \leq \binom{N}{m} m^K$, and therefore $log|\mathcal{H}_m| = \Theta\left(\log\binom{N}{m} + K\log m\right) = \Theta\left(m\log\frac{N}{m} + K\log m\right)$.*

We now have the following bound for our forecaster, which follows from (3).

Theorem 1. *Given a convex loss function ℓ, for any sequence of predictions $\boldsymbol{f}^t$ and outcomes y^t, where $t = 1, 2, \ldots, T$,*

$$\frac{\hat{L}^T}{c_\eta} \leq \min_{\pi\in\mathcal{H}_m} L_\pi^T + \frac{\log|\mathcal{H}_m|}{\eta} \leq \min_{S\in\mathcal{S}_m} \sum_{k=1}^{K} \min_{i\in S} L_{k,i}^T + \frac{m\log\frac{N}{m} + K\log m}{\eta}.$$

3.2 An Alternative Set of Hyperexperts

We now consider a slightly different description of a hyperexpert. This alternative representation, while not as natural as that described above, will be useful in Section 5. Formally, we define the class

$$\bar{\mathcal{H}}_m := \{(S,\phi) \text{ for every } S \in \mathcal{S}_m \text{ and } \phi : [K] \to [m]\}.$$

Notice, the pair (S,ϕ) induces a map $\pi \in \mathcal{H}_m$ in the natural way: if $S = \{i_1, \ldots, i_m\}$, with $i_1 < \ldots < i_m$, then π is defined as the mapping $k \mapsto i_{\phi(k)}$. For convenience, write $\Psi(S,\phi,k) := i_{\phi(k)} = \pi(k)$. Then the prediction of the hyperexpert (S,ϕ) on round t is exactly the prediction of π, that is f_i^t where $i = \Psi(S,\phi,\kappa(t))$. Similarly, we define the weight of a hyperexpert $(S,\phi) \in \bar{\mathcal{H}}_m$ simply as $v_{S,\phi}^t := \prod_{k=1}^K w_{k,\Psi(S,\phi,k)}^t = v_\pi^t$. Thus, the prediction of the Weighted Average Forecaster with access to the set of hyperexperts $\bar{\mathcal{H}}_m$ is

$$\hat{q}^t = \frac{\sum_{(S,\phi)\in\bar{\mathcal{H}}_m} v_{S,\phi}^t f_{\Psi(S,\phi,\kappa(t))}^t}{\sum_{(S',\phi')\in\bar{\mathcal{H}}_m} v_{S',\phi'}^t}. \tag{6}$$

We note that any $\pi \in \mathcal{H}_m$ can be described by some (S,ϕ), and so we have a surjection $\bar{\mathcal{H}}_m \to \mathcal{H}_m$, yet this is not an injection. Indeed, maps π for which $\mathrm{img}(\pi) < m$ will be represented by more than one pair (S,ϕ). In other words, we have "overcounted" our comparators a bit, and so the prediction $\hat{q}^t$ will differ from $\hat{p}^t$. However, Theorem 1 will also hold for the weighted average forecaster given access to the expert class $\bar{\mathcal{H}}_m$. Notice, the set $\bar{\mathcal{H}}_m$ has size exactly $\binom{N}{m} m^K$, and thus Lemma 1 tells us that $\log|\mathcal{H}_m|$ and $\log|\bar{\mathcal{H}}_m|$ are of the same order.

3.3 Hardness results

Unfortunately, the algorithm for computing either $\hat{p}^t$ or $\hat{q}^t$ described above requires performing a computation in which we sum over an exponentially large number of subsets. One might hope for a simplification but, as the following lemmas suggest, we cannot hope for such a result. For this section, we let $W^t := [w_{k,i}^t]_{k,i}$, an arbitrary nonnegative matrix, and $\phi_m(W^t) := \sum_{\pi\in\mathcal{H}_m} \prod_{k=1}^K w_{k,\pi(k)}^t$.

Lemma 2. *Computing $\hat{p}^t$ as in (5) for an arbitrary matrix nonnegative W^t and arbitrary prediction vector $\boldsymbol{f}^t$ is as hard as computing $\phi_m(W^t)$.*

Proof. Because $\boldsymbol{f}^t$ is arbitrary, computing $\hat{p}^t$ is equivalent to computing the weights $u_{k,i} = \frac{1}{\phi_m(W^t)} \sum_{\pi \in \mathcal{H}_m^{k,i}} \prod_{k'=1}^{K} w^t_{k',\pi(k')}$. However, this also implies that we could compute $\frac{\phi_{m-1}(W^t)}{\phi_m(W^t)}$. To see this, augment W^t as follows. Let $\hat{W}^t := \begin{bmatrix} 1 & & \mathbf{0} \\ & W^t & \\ 1 & & 1 \end{bmatrix}$. If we could compute the weights $u_{k,i}$ for this larger matrix $\hat{W}^t$ then, in particular, we could compute $u_{K+1,N+1}$. However, it can be checked that $u_{K+1,N+1} = \frac{\phi_{m-1}(W^t)}{\phi_m(W^t)}$ given the construction of $\hat{W}^t$.

Furthermore, if we compute $\frac{\phi_{m-1}(W^t)}{\phi_m(W^t)}$ for each m, then we could compute $\prod_{l=1}^{m-1} \frac{\phi_l(W^t)}{\phi_{l+1}(W^t)} = \frac{\phi_1(W^t)}{\phi_m(W^t)}$. But the quantity $\phi_1(W^t) = \sum_{i=1}^{N} \prod_{k=1}^{K} w^t_{k,i}$ can be computed efficiently, giving us $\phi_m(W^t) = \phi_1(W^t) \left(\frac{\phi_1(W^t)}{\phi_m(W^t)} \right)^{-1}$. □

Lemma 3. *Assuming $K = N$, computing $\phi_m(W^t)$ for any nonnegative W^t and any m is as hard as computing Perm(W^t), the permanent of a nonnegative matrix W^t.*

Proof. The permanent $\mathrm{Perm}(W)$ is defined as $\sum_{\pi \in \mathrm{Sym}_N} \prod w^t_{k,\pi(k)}$. This expression is similar to $\phi_N(W)$, yet this sum is taken over only permutations $\pi \in \mathrm{Sym}_N$, the symmetric group on N, rather than all functions from $[N] \to [N]$. However, the set of permutations on $[N]$ is exactly the set of all functions on $[N]$ minus those functions π for which $|\mathrm{img}(\pi)| \leq N - 1$. Thus, we see that $\mathrm{Perm}(W) = \phi_N(W) - \phi_{N-1}(W)$. □

Theorem 2. *Computing either $\hat{p}^t$ or $\hat{q}^t$, as in* (5) *or* (6) *respectively, is hard.*

Proof. Combining Lemmas 2 and 3, we see that computing the prediction $\hat{p}^t$ is at least as hard as computing the permanent of any matrix with positive entries, which is known to be a hard problem. While we omit it, the same analysis can be used for computing $\hat{q}^t$, i.e. when our expert class is $\bar{\mathcal{H}}_m$. □

As an aside, it is tempting to consider utilizing the Follow the Perturbed Leader algorithm of Kalai and Vempala [9]. However, the curious reader can also check that not only is it hard to compute the prediction, it is even hard to find the best hyperexpert and thus the perturbed leader.

4 Deterministic Mixing Algorithm

While the reduction to a single-task setting, discussed in the previous section, is natural, computing the predictions $\hat{p}^t$ or $\hat{q}^t$ directly proves to be infeasible. Somewhat surprisingly, we can solve the *sequential multitask problem* without computing the predictions explicitly.

The problem of multitask learning, as presented in this paper, can be viewed as a problem of competing against comparators which shift within a pool of

size m, a problem analyzed extensively in Bousquet and Warmuth [8]. However, there are a few important differences. On the positive side, we have the extra information that no shifting of comparators occurs when staying within the same task. First, this allows us to design a truly online algorithm which has to keep only K weight vectors, instead of a complete history (or its approximation) as for the Decaying Past scheme in [8]. Second, the extra information allows us to obtain a bound which is independent of time: it only depends on the number of switches between the tasks. On the down side, in the case of the *shifting multitask problem*, tasks and comparators can change at every time step.

In this section, we show how the mixing algorithm of [8] can be adapted to our setting. We design the mixing scheme to obtain the bounds of Theorem 1 for the *sequential multitask problem*, and prove a bound for the *shifting multitask problem* in terms of the number of shifts, but independent of the time horizon.

Algorithm 1 Multitask Mixing Algorithm

Input: η
Initialize $\tilde{\mathbf{w}}_k^0 = \frac{1}{N}\mathbf{1}$ for all $k \in [K]$
for $t = 1$ to T **do**
  Let $k = \kappa(t)$, the current task
  Choose a distribution β_t over tasks
  Set $\tilde{\boldsymbol{z}}^t = \sum_{k'=1}^{K} \beta_t(k') \tilde{\mathbf{w}}_{k'}^{t-1}$
  Predict $\hat{p}^t = \tilde{\boldsymbol{z}}^t \cdot \boldsymbol{f}^t$
  Update $\tilde{w}_{k,i}^t = \left(\tilde{z}_i^t e^{-\eta \ell_i^t}\right) / \left(\sum_{i=1}^{N} \tilde{z}_i^t e^{-\eta \ell_i^t}\right)$ for all $i \in [N]$
  Set $\tilde{\mathbf{w}}_{k'}^t = \tilde{\mathbf{w}}_{k'}^{t-1}$ for any $k' \neq k$.
end for

The above algorithm keeps normalized weights $\tilde{\mathbf{w}}_k^t \in \mathbb{R}^N$ over experts for each task $k \in [K]$ and mixes $\tilde{\mathbf{w}}_k^t$'s together with an appropriate mixing distribution β_t over tasks to form a prediction. It is precisely by choosing β_t correctly that one can pass information from one task to another through sharing of weights. The mixture of weights across tasks is then updated according to the usual exponential weighted average update. The new normalized distribution becomes the new weight vector for the current task. It is important to note that $\tilde{\mathbf{w}}_k^t$ is updated *only* when $k = \kappa(t)$. [4] The following per-step bound holds for our algorithm, similarly to Lemma 4 in [8]. For any $\tilde{\boldsymbol{u}}_t$ and $k' \in [K]$,

$$\hat{\ell}^t \leq c_\eta \left(\tilde{\boldsymbol{u}}_t \cdot \boldsymbol{\ell}^t + \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\mathbf{w}}_{k'}^{t-1} \right) - \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\mathbf{w}}_{\kappa(t)}^{t} \right) + \frac{1}{\eta} \ln \frac{1}{\beta_t(k')} \right) \tag{7}$$

[4] Referring to $\tilde{\mathbf{w}}_k^q$, where q is the last time the task k was performed, is somewhat cumbersome. Hence, we set $\tilde{\mathbf{w}}_{k'}^t = \tilde{\mathbf{w}}_{k'}^{t-1}$ for any $k' \neq k$ and avoid referring to time steps before $t-1$.

where the relative entropy is $\triangle(\boldsymbol{u},\boldsymbol{v}) = \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i}$ for normalized $\boldsymbol{u},\boldsymbol{v} \in \mathbb{R}^N$ (see Appendix A for the proof). If the loss function ℓ is η-exp-concave (see [1]), the constant c_η disappears from the bound.

We first show that a simple choice of β_t leads to the trivial case of unrelated tasks. Intuitively, if no mixing occurs, i.e. β_t puts all the weight on the current task at the previous step, the tasks are uncoupled. This is exhibited by the next proposition, whose proof is straightforward, and is omitted.

Proposition 1. *If we choose $\beta_t(k) = 1$ if $k = \kappa(t)$ and 0 otherwise, then Algorithm 1 yields*

$$\hat{L}^t \le c_\eta \min_{S\in\mathcal{S}_m} \sum_{k=1}^{K} \min_{i\in S} L_{k,i}^t + \frac{c_\eta}{\eta} K \ln N.$$

Of course, if we are to gain information from the other tasks, we should mix the weights instead of concentrating on the current task. The next definition is needed to quantify which tasks appeared more recently: they will be given more weight by our algorithm.

Definition 1. *If, at time t, tasks are ordered according to the most recent appearance, we let the rank $\rho_t(k) \in [K] \cup \{\infty\}$ be the position of task k in this ordered list. If k has not appeared yet, set $\rho_t(k) = \infty$.*

Theorem 3. *Suppose the tasks are presented in an arbitrary order, but necessarily switching at every time step. If we choose $\beta_t(\kappa(t)) = \alpha$ and for any $k \neq \kappa(t)$ set $\beta_t(k) = (1-\alpha)\cdot \frac{1}{\rho_t(k)^2}\frac{1}{Z_t}$ then Algorithm 1 yields*

$$\hat{L}^t \le c_\eta \min_{S\in\mathcal{S}_m} \sum_{k=1}^{K} \min_{i\in S} L_{k,i}^t + \frac{c_\eta}{\eta}\left(m \ln \frac{N}{m-2} + 3T \ln m\right).$$

Here, $Z_t = \sum_{k\in[K],k\neq\kappa(t)} \frac{1}{\rho_t(k)^2} < 2$, $\alpha = 1 - \frac{2}{m}$, and $m > 2$. It is understood that we set $\beta_t(k) = 0$ when $\rho_t(k) = \infty$.

In the theorem above, the number of switches n between the tasks is $T-1$. Now, consider an arbitrary sequence of task presentations. The proof of the above theorem, given in the appendix, reveals that the complexity term in the bound only depends on the number n of switches between the tasks, and not on the time horizon T. Indeed, when continuously performing the same task, we exploit the information that the comparator does not change and put all the weight β_t on the current task, losing nothing in the complexity term. This improves the bound of [8] by removing the $\ln T$ term, as the next corollary shows.

Corollary 1. *Let $\beta_t(k)$ be defined as in Theorem 3 whenever a switch between tasks occurs and let $\beta_t(\kappa(t)) = 1$ whenever no switch occurs. Then for the* shifting multitask problem*, Algorithm 1 yields*

$$\hat{L}^t \le c_\eta \min_{S\in\mathcal{S}_m} \sum_{k=1}^{K} \min_{i\in S} L_{k,i}^t + \frac{c_\eta}{\eta}\left(m \ln \frac{N}{m-2} + 3n \ln m\right).$$

where n is the number of switches between the tasks.

Corollary 2. *With the same choice of β_t as in Corollary 1, for the* sequential multitask problem*, Algorithm 1 yields*

$$\hat{L}^t \le c_\eta \min_{S \in \mathcal{S}_m} \sum_{k=1}^{K} \min_{i \in S} L_{k,i}^t + \frac{c_\eta}{\eta} \left(m \ln \frac{N}{m-2} + 3K \ln m \right).$$

Up to a constant factor, this is the bound of Theorem 1. Additionally to removing the $\ln T$ term, we obtained a space and time-efficient algorithm. Indeed, the storage requirement is only KN, which does not depend on T.

5 Predicting with a Random Walk

In the previous section we exhibited an efficient algorithm which attains the bound of Theorem 1 for the *sequential multitask problem*. Unfortunately, for the more general case of the *shifting multitask problem*, the bound degrades with the number of switches between the tasks. Encouraged by the fact that it is possible to design online algorithms even if the prediction is provably hard to compute, we look for a different algorithm for the *shifting multitask problem*.

Fortunately, the hardness of computing the weights exactly, as shown in Section 3.3, does not immediately imply that *sampling* according to these weights is necessarily difficult. In this section we provide a randomized algorithm based on a Markov chain Monte Carlo method. In particular, we show how to sample a random variable $X^t \in [0,1]$ such that $\mathbb{E}X^t = \hat{q}^t$, where $\hat{q}^t$ is the prediction defined in (6).

Algorithm 2 Randomized prediction

1: Input: Round t; Number R_1 of iterations; Parameter $m < N$; $K \times N$ matrix $[w_{k,i}^t]$
2: **for** $j = 1$ to R_1 **do**
3: Sample $S \in \mathcal{S}_m$ according to $P(S) = \left(\prod_{k=1}^{K} \sum_{i \in S} w_{k,i}^t \right) \Big/ \left(\sum_{S' \in \mathcal{S}_m} \prod_{k=1}^{K} \sum_{i \in S'} w_{k,i}^t \right)$
4: Order $S = \{i_1, \ldots, i_m\}$
5: Sample $\phi : [K] \to [m]$ according to $P(\phi|S) = \left(\prod_{k=1}^{K} w_{k,i_{\phi(k)}}^t \right) \Big/ \left(\prod_{k=1}^{K} \sum_{i \in S} w_{k,i}^t \right)$
6: Set $X_j^t = f_{\Psi(S,\phi,\kappa(t))}^t$
7: **end for**
8: Predict with $\bar{X}^t = \frac{1}{R_1} \sum_{j=1}^{R_1} X_j^t$

Algorithm 2 samples a subset of m experts $S = \{i_1, \ldots, i_m\}$, and then samples a map ϕ from the set of tasks to this subset of experts. If the current task is k, the algorithm returns the prediction of expert $i_{\phi(k)} = \Psi(S, \phi, k)$. We have,

$$P(S,\phi) = P(S)P(\phi|S) = \frac{\prod_{k=1}^{K} w_{k,\Psi(S,\phi,k)}^t}{\sum_{S' \in \mathcal{S}_m} \prod_{k=1}^{K} \sum_{i \in S'} w_{k,i}^t} = \frac{v_{S,\phi}}{\sum_{(S',\phi')} v_{S',\phi'}}.$$

Note that $\hat{q}^t = \sum_{(S,\phi)\in\bar{\mathcal{H}}_m} P(S,\phi) f^t_{\Psi(S,\phi,k)}$, and it follows that $\mathbb{E}X^t = \hat{q}^t$.

Notice that, in the above algorithm, every step can be computed efficiently except for step 3. Indeed, sampling ϕ given a set S can be done by independently sampling assignments $\phi(k)$ for all k. In step 3, however, we must sample a subset S whose weight we define as $\prod_{k=1}^{K}\sum_{i\in S} w^t_{k,i}$. Computing the weights for all subsets is implausible, but it turns out that we can apply a Markov Chain Monte Carlo technique known as the Metropolis-Hastings algorithm. This process begins with subset S of size m and swaps experts in and out of S according to a random process. At the end of R_2 rounds, we will have an *induced* distribution Q_{R_2} on the collection of m-subsets $\mathcal{S}_m$ which will approximate the distribution P defined in step 3 of Algorithm 2.

More formally, the process of sampling S is as follows.

Algorithm 3 Sampling a set of m experts

1: Input: Matrix of $w^t_{k,i}$, $i \in [N]$, $k \in [K]$, and number of rounds R_2
2: Start with some $S_0 \in \mathcal{S}_m$, an initial set of m experts
3: **for** $r = 0$ to $R_2 - 1$ **do**
4: Uniformly at random, choose $i \in [N] \setminus S_r$ and $j \in S_r$. Let $S'_r = S_r \cup i \setminus j$.
5: Calculate $\omega(S_r) = \prod_{k=1}^{K}\sum_{i\in S_r} w^t_{k,i}$ and $\omega(S'_r) = \prod_{k=1}^{K}\sum_{i\in S'_r} w^t_{k,i}$
6: With probability $\min\left\{1, \frac{\omega(S'_r)}{\omega(S_r)}\right\}$, set $S_{r+1} \leftarrow S'_r$, otherwise $S_{r+1} \leftarrow S_r$
7: **end for**
8: Output: S_{R_2}

Definition 2. *Given two probability distributions P_1 and P_2 on a space X, we define the total variation distance $||P_1 - P_2|| = \frac{1}{2}\sum_{x\in X} |P_1(x) - P_2(x)|$.*

It can be shown that the distance $||Q_{R_2} - P|| \to 0$ as $R_2 \to \infty$. While we omit the details of this argument, it follow from the fact that P is the stationary distribution of the Markov chain described in Algorithm 3. More information can be found in any introduction to the Metropolis-Hastings algorithm.

Theorem 4. *If a forecaster predicts according to algorithm 2 with the sampling step 3 approximated by Algorithm 3, then with probability at least $1 - \delta$,*

$$\hat{L}^T \le c_\eta \min_{\pi\in\mathcal{H}_m} \sum_{k=1}^{K} L^T_{k,\pi(k)} + \frac{c_\eta}{\eta}\left(m\log\frac{N}{m} + K\log m\right) + CT\epsilon + CT\sqrt{\frac{\ln\frac{2T}{\delta}}{2R_1}},$$

where R_1 is the number of times we sample the predictions, C is the Lipschitz constant of ℓ, and R_2 is chosen such that $\|Q_{R_2} - P\| \le \epsilon/2$. The last two terms can be made arbitrarily small by choosing large enough R_1 and R_2.

A key ingredient of this theorem is our ability to choose R_2 such that $\|Q_{R_2} - P\| < \epsilon$. In general, since ϵ must depend on T, we would hope that $R_2 = poly(1/\epsilon)$. In other words, we would like to show that our Markov chain has

a fast *mixing time*. In some special cases, one can prove a useful bound on the mixing time, yet such results are scarce and typically quite difficult to prove. On the other hand, this does not imply that the mixing time is prohibitively large. In the next section, we provide empirical evidence that, in fact, our Markov chain mixes extremely quickly.

5.1 Experiments

For small K and N and some matrix $[w^t_{k,i}]_{k,i}$ we can compute the true distribution P on $\mathcal{S}_m$, and we can compare that to the distribution Q_{R_2} induced by the random walk described in Algorithm 3. The graphs in Figure 1 show that, in fact, Q_{R_2} approaches P very quickly even after only a few rounds R_2.

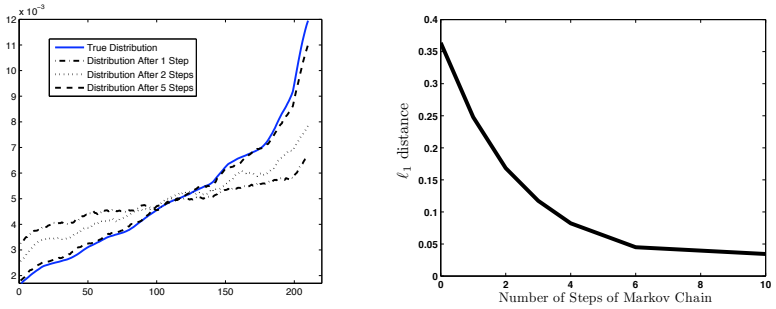

Fig. 1. We generate a random $K \times N$ matrix $[w^t_{k,i}]_{k,i}$, where $K = 5$, $N = 10$, and we consider the distribution on $\mathcal{S}_m$ described in algorithm 2. In the first graph we compare this "true distribution" P, sorted by $P(S)$, to the induced distribution Q_{R_2} for the values $R_2 = 1, 2, 5$. In the second graph we see how quickly the total variation distance $\|P - Q_{R_2}\|$ shrinks relative to the number of steps R_2.

Figure 2 demonstrates the performance Algorithm 2, for various choices of m, on the following problem. We used $R_1 = 5$ and we employ Algorithm 3 to sample $S \in \mathcal{S}_m$ with $R_2 = 700$. On each round, we draw a random $x_t \in X = \mathbf{R}^d$, where in this case $d = 4$, and this x_t is used to generate the outcome and the predictions of the experts. We have $K = 60$ tasks, where each task f_k is a linear classifier on X. If $k = \kappa(t)$, the outcome for round t is $I(f_k \cdot x_t > 0)$. We choose 70 "bad" experts, who predict randomly, and 30 "good" experts which are, themselves, randomly chosen linear classifiers e_i on X: on round t expert i predicts $I(e_i \cdot x_t > 0)$ with 30% label noise. In the plots below, we compare the performance of the algorithm for all values of m to the comparator $\sum_k \min_i L^t_{k,i}$. It is quite interesting to see the tradeoff between short and long-term performance for various values of m.

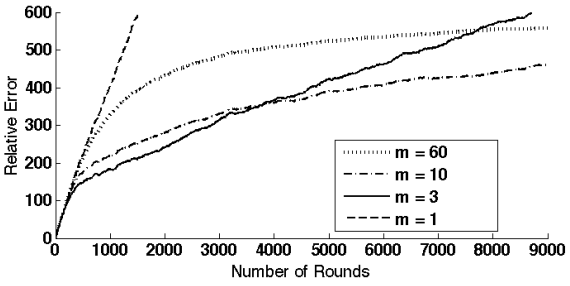

Fig. 2. The performance of Algorithm 2 on a toy example. Large values of m have good long-term performance yet are "slow" to find the best expert. On the other hand, the algorithm learns very quickly with small values of m, but pays a price in the long run. In this example, $m = 10$ appears to be a good choice.

6 Conclusion

We conclude by stating some open problems.

Recall that in Section 3.3 we show a crucial relationship between computing the forecaster's prediction and computing the permanent of a matrix. Interestingly, in very recent work, Jerrum et al [10] exhibit a Markov chain, and a bound on the mixing time, that can be used to efficiently approximate the permanent of an arbitrary square matrix with nonnegative entries. Could such techniques be employed to provide a randomized prediction algorithm with provably fast convergence?

Is it possible to develop a version of the Multitask Mixing Algorithm for the *shifting multitask problem* and prove that performance does not degrade with the number of shifts between the tasks? Are there reasonable assumptions under which Φ in the proof of Theorem 3 depends sublinearly on the number of shifts?

Acknowledgments. We would like to thank Manfred Warmuth for his knowledge of prediction algorithms and helpful discussions on mixing priors. We would like to thank Alistair Sinclair for his expertise on the MCMC methods. We gratefully acknowledge the support of DARPA under grant FA8750-05-20249.

References

1. Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
2. Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *JMLR*, 6:615–637, 2005.
3. Rie K. Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.
4. R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

5. O. Dekel, Y. Singer, and P. Long. Online multitask learning. In *COLT*, 2006.
6. J. Baxter. A model of inductive bias learning. *JAIR*, 12:149–198, 2000.
7. S. Ben-David and R. Schuller. Exploiting task relatedness for mulitple task learning. In *COLT*, pages 567–580, 2003.
8. Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *JMLR*, 3:363–396, 2002.
9. A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.
10. M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *STOC '01*.

A Proofs

Proof (of Lemma 1). We prove the upper bound by considering ways of constructing a function in $\mathcal{H}_m$. We can pick an element of $\mathcal{H}_m$ by first picking an m-sized subset S, and then choose a map $[K] \to S$. There are $\binom{N}{m}$ such subsets, and m^K such maps.

We prove the lower bound by considering the number of maps with image size *exactly* m. Explicitly, define $\rho_m(k) := |\{\pi : [k] \to [N] \text{ s.t. } \mathrm{img}(\pi) = m\}|$. Certainly, $\rho_m(K) \leq |\mathcal{H}_m|$. Notice that $\rho_m(m) = \frac{N!}{(N-m)!}$ since we must choose an ordered sequence of m different numbers from N items. Also, for $k > m$, $\rho_m(k) > m\rho_m(k-1)$. We can see this recursive step by observing that, for every $\pi : [k-1] \to [N]$ where $\mathrm{img}(\pi) = m$, we can contruct m unique functions from $[k] \to [N]$, that also have image size m, by mapping the element k to one of the elements in the image of π. Thus we see that

$$\rho_m(K) > \frac{N!}{(N-m)!} m^{K-m} > \binom{N}{m} m^{K-m} m!$$

□

Proof (of Inequality (7)). The details of this proof can be found in [8, 1].

$$\begin{aligned}
\hat{\ell}^t / c_\eta &\leq \ell(\tilde{\boldsymbol{z}}^t \cdot \boldsymbol{f}^t, y^t)/c_\eta \leq c_\eta^{-1} \sum_{i=1}^N \tilde{z}_i^t \ell_i^t \leq -\frac{1}{\eta} \ln \sum_{i=1}^N \tilde{z}_i^t e^{-\eta \ell_i^t} \\
&= \tilde{\boldsymbol{u}}_t \cdot \boldsymbol{\ell}^t + \frac{1}{\eta} \sum_{i=1}^N u_{t,i} \ln e^{-\eta \ell_i^t} - \frac{1}{\eta} \ln \sum_{i=1}^N \tilde{z}_i^t e^{-\eta \ell_i^t} \\
&= \tilde{\boldsymbol{u}}_t \cdot \boldsymbol{\ell}^t + \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\boldsymbol{z}}^t \right) - \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\mathbf{w}}_{\kappa(t)}^t \right) \\
&\leq \tilde{\boldsymbol{u}}_t \cdot \boldsymbol{\ell}^t + \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\mathbf{w}}_{k'}^{t-1} \right) - \frac{1}{\eta} \triangle \left(\tilde{\boldsymbol{u}}_t, \tilde{\mathbf{w}}_{\kappa(t)}^t \right) + \frac{1}{\eta} \ln \frac{1}{\beta_t(k')}
\end{aligned}$$

□

Proof (of Theorem 3). The proof is an adaptation of the proof of Corollary 9 in [8], taking into account the task information. Let $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_m$ be m arbitrary

comparators. For any $\pi : [K] \mapsto [m]$, $\tilde{\boldsymbol{u}}_{\pi(k)}$ is the comparator used by the environment for task k (known only in the hindsight). For any t,

$$\hat{\ell}^t/c_\eta \le \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(\triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\boldsymbol{z}}^t\right) - \triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_k^t\right)\right),$$

where $k = \kappa(t)$. There are m time points when a task k begins and it is the first task being compared to the comparator $\tilde{\boldsymbol{u}}_{\pi(k)}$. For these m time steps,

$$\hat{\ell}^t/c_\eta \le \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(\triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_k^{t-1}\right) - \triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_k^t\right) + \ln\frac{1}{\alpha}\right),$$

where $\tilde{\mathbf{w}}_k^{t-1} = \tilde{\mathbf{w}}_k^0$, as it has not been modified yet. Otherwise,

$$\hat{\ell}^t/c_\eta \le \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(\triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_{k'}^{t-1}\right) - \triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_k^t\right) + \ln\frac{\rho_t(k')^2 Z_t}{1-\alpha}\right),$$

where k' is the most recent task played against the comparator $\tilde{\boldsymbol{u}}_{\pi(k)}$ (this is still true in the case k is the only task for the comparator $\tilde{\boldsymbol{u}}_{\pi(k)}$ because $\alpha > \frac{1-\alpha}{\rho_t(k)^2 Z_t}$ with the choice of α below). We upper bound

$$\hat{\ell}^t/c_\eta \le \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(\triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_{k'}^{t-1}\right) - \triangle\left(\tilde{\boldsymbol{u}}_{\pi(k)}, \tilde{\mathbf{w}}_k^t\right) + \ln\rho_t(k')^2 + \ln\frac{2}{1-\alpha}\right).$$

We note that for each comparator $\tilde{\boldsymbol{u}}_j$, the relative entropy terms telescope. Recall that $\tilde{\mathbf{w}}_k^0 = \frac{1}{N}\mathbf{1}$ and $\triangle\left(\tilde{\boldsymbol{u}}_j, \frac{1}{N}\mathbf{1}\right) \le \ln N$. Summing over $t = 1, \ldots, T$,

$$\hat{L}^t/c_\eta \le \sum_{k=1}^{K}\sum_{t\in\tau(k)} \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(m\ln N + m\ln\frac{1}{\alpha} + (T-m)\ln\frac{2}{1-\alpha} + 2\Phi\right),$$

where $\Phi = \sum_{t=1}^{T}\ln\rho_t(k')$ and k' is the task previous to $\kappa(t)$ which used the same comparator. We now upper-bound Φ by noting that $\rho_t(k')$ is smaller than the number of time steps δ_t that elapsed since task k' was performed. Note that $\sum_{t=1}^{T}\delta_t \le mT$ as there are m subsequences summing to at most T each. Hence, $\ln\prod_{t=1}^{T}\delta_t$ is maximized when all the terms δ_t are equal (i.e. at most m), resulting in $\Phi \le T\ln m$. Note that Φ, which depends on the sequence of task presentations, is potentially much smaller than $T\ln m$.

Choosing, for instance, $\alpha = 1 - \frac{2}{m}$ whenever $m > 2$,

$$\hat{L}^t/c_\eta \le \sum_{k=1}^{K}\sum_{t\in\tau(k)} \tilde{\boldsymbol{u}}_{\pi(k)} \cdot \boldsymbol{\ell}^t + \frac{1}{\eta}\left(m\ln\frac{N}{m-2} + 3T\ln m\right).$$

The constant 3 can be optimized by choosing a non-quadratic power decay for β_t at the expense of having an extra $T\ln K$ term. Setting the comparators $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_m$ to be unit vectors amounts to finding the best m-subset of N experts. The minimum over the assignment π of tasks to experts then amounts to choosing the best expert out of m possibilities for each task. □

Proof (of Theorem 4). Let $\tilde{P}(S,\phi) = Q_{R_2}(S)P(\phi|S)$, the induced distribution on the (S,ϕ) pairs when the MCMC Algorithm 3 is used for sampling. Let X^t now stand for the random choices f_i^t according to this induced distribution $\tilde{P}$, which is close to P. Then $\mathbb{E}X^t = \sum_{(S,\phi)\in\bar{\mathcal{H}}_m} f^t_{\Psi(S,\phi,k)}\tilde{P}(S,\phi)$. Hence,

$$\begin{aligned}|\hat{q}^t - \mathbb{E}X^t| &= \left|\sum_{(S,\phi)\in\bar{\mathcal{H}}_m} f^t_{\Psi(S,\phi,k)}\left(P(\phi,S) - \tilde{P}(\phi,S)\right)\right| \\ &\leq \sum_{(S,\phi)\in\bar{\mathcal{H}}_m} f^t_{\Psi(S,\phi,k)}P(\phi|S)\,|P(S) - Q_{R_2}(S)| \leq 2\|Q_{R_2} - P\| \leq \epsilon.\end{aligned}$$

Since we sample X^t independently R_1 times, standard concentration inequalities ensure that $P\left(\left|\bar{X}^t - \mathbb{E}X^t\right| \geq \sqrt{\left(\ln\frac{2T}{\delta}\right)/(2R_1)}\right) \leq \frac{\delta}{T}$. Combining with the above result, $P\left(\left|\bar{X}^t - \hat{q}^t\right| \geq \epsilon + \sqrt{\left(\ln\frac{2T}{\delta}\right)/(2R_1)}\right) \leq \frac{\delta}{T}$. Since ℓ is Lipschitz, $\left|\ell(\bar{X}^t, y^t) - \ell(\hat{q}^t, y^t)\right| \leq C\epsilon + C\sqrt{\left(\ln\frac{2T}{\delta}\right)/(2R_1)}$ with probability at least $1-\delta/T$. By the union-bound, with probability at least $1-\delta$,

$$\left|\sum_{t=1}^{T}\ell(\bar{X}^t, y^t) - \sum_{t=1}^{T}\ell(\hat{q}^t, y^t)\right| \leq T\cdot C\epsilon + T\cdot C\sqrt{\frac{\ln\frac{2T}{\delta}}{2R_1}}.$$

Combining with the bound of Theorem 1 (for $\bar{\mathcal{H}}_m$), we obtain the desired result. □