Learning from Delayed Outcomes via Proxies with Applications to Recommender Systems

Timothy A. Mann [*1] Sven Gowal [*1] András György [1] Ray Jiang [1] Huiyi Hu [1] Balaji Lakshminarayanan [1] Prav Srinivasan [1]

Abstract

Predicting delayed outcomes is an important problem in recommender systems (e.g., if customers will finish reading an ebook). We formalize the problem as an adversarial, delayed online learning problem and consider how a proxy for the delayed outcome (e.g., if customers read a third of the book in 24 hours) can help minimize regret, even though the proxy is not available when making a prediction. Motivated by our regret analysis, we propose two neural network architectures: Factored Forecaster (FF) which is ideal if the proxy is informative of the outcome in hindsight, and Residual Factored Forecaster (RFF) that is robust to a non-informative proxy. Experiments on two real-world datasets for predicting human behavior show that RFF outperforms both FF and a direct forecaster that does not make use of the proxy. Our results suggest that exploiting proxies by factorization is a promising way to mitigate the impact of long delays in human-behavior prediction tasks.

1 Introduction

Predicting delayed outcomes is an important problem in recommender systems and online advertising. The problem is aggravated by a stream of new items for recommendation while successful recommendations are determined by a delayed outcome (e.g., on the order of weeks or months). The actual time that an outcome is delayed is less important than the number of times a forecaster is queried for a new item without receiving any feedback. For example, in a popular recommender system an outcome delayed by a week might mean that the forecaster makes millions of predictions for a popular new item without incorporating any feedback into the forecaster's model.

[*]Equal contribution [1]DeepMind, London, UK. Correspondence to: Timothy A. Mann <timothymann@google.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Motivating Example: Consider an online marketplace for ebooks. Our problem is to recommend ebooks that a customer will finish within 90 days of its purchase. Similarly to YouTube video recommendations (Davidson et al., 2010), when a customer visits, the marketplace (1) generates a small set of ebook candidates, (2) scores each of the candidates, and (3) presents them in descending order of predicted probability of finishing the book. When an ebook is purchased, the outcome is unknown for 90 days. The approach taken by much of the prior work on learning with delay (Weinberger & Ordentlich, 2002; Mesterharm, 2005; Joulani et al., 2013; Quanrud & Khashabi, 2015) would require waiting the whole 90 days to see the outcome. However, new books are added to the marketplace every day. If we waited 90 days before we can accurately predict the engagement probability of a new book, the marketplace will miss out on many potential sales. While generalization can help to mitigate this problem, per product memorization is important for large scale recommender systems (Cheng et al., 2016). A complementary approach could make use of a less-delayed proxy for the delayed outcome. For example, one day after a purchase we can define a proxy based on the furthest page reached in the ebook. Clearly, this proxy provides some information about the eventual outcome. On the other hand, we are interested in predicting the eventual outcome, not the proxy. This leads to the main question studied in this paper: How can we learn to predict delayed outcomes via a proxy?

The Problem: We formulate learning from delayed outcomes as a full information online learning problem where the goal is to minimize regret (Hazan et al., 2016). The online learning framework is ideal for predicting human responses (e.g., recommender systems), because it does not make distributional assumptions about the data. In our motivating example, ebook candidates are not generated by a fixed distribution, since new books are being added and the popularity of books changes over time. For these reasons, we model the problem as a delayed, adversarial online learning problem with less-delayed proxies, where instances are selected by an oblivious adversary but the proxies and outcomes are sampled in a probabilistic manner conditioned on the instances. Furthermore, we assume that the number of proxy symbols and the number of outcomes are small relative to the number of instances (e.g., the number of ebooks in an online marketplace).

Proposed Solution: We suggest that a proxy for the delayed outcome can be exploited by breaking the prediction problem into two subproblems: (a) predicting the proxy from instances, and (b) predicting the delayed outcome from the proxy. The intuition is that the first model can be updated quickly, since the proxy is less delayed than the outcome. On the other hand, the model predicting the outcome can only be updated when outcomes are revealed but it only uses the proxy as input. Thus, it can generalize across instances. We analyze the regret of this factored approach and find that it scales with $(D+N)$ where D is the outcome delay and N is the total number of items. On the other hand, the regret of forecasters that ignore the proxy may scale with $(D \times N)$. Thus, proxy information can mitigate the impact of delay.

Practical Neural Implementations: Using this intuition we propose a factored neural network-based online learner, called factored forecaster (FF), based on two modules – one predicts the proxies, while the other predicts the delayed outcomes. However, we found that the selected proxies are not always sufficiently informative of the outcomes in both of our experimental domains. To mitigate this problem, we introduce a second neural network-based learner, called "factored with residual" forecaster (RFF), that introduces a residual correction term. We compare both of these factored algorithms to an algorithm that ignores the proxies, called the direct forecaster (DF), on two experimental domains: (1) predicting commit activity for GitHub repositories, and (2) predicting engagement with items acquired from a popular marketplace. In both of these domains, RFF outperforms both DF and FF.

Contribution: This paper offers three main contributions:

1. We formalize the problem of learning from delayed outcomes via proxies as a full-information online learning problem. Many prior works have analyzed learning with delayed outcomes (e.g., Weinberger & Ordentlich, 2002; Mesterharm, 2005; Joulani et al., 2013); however, to our knowledge, this is the first work to consider using proxies to mitigate the impact of delay.

2. We analyze the regret of learning with proxies. In particular, factorization helps when the forecaster is forced to make a burst of predictions for the same instance (i.e., predictions for a popular new item are requested many times before any outcome is revealed).

3. Finally, we introduce a practical neural network implementation, RFF, for exploiting proxies. Our experiments provide evidence that RFF performs as well as a delayed learner when the proxies are not informative of the outcomes and outperforms a delayed learner when instances are chosen adversarially and the proxies are informative of the outcomes.

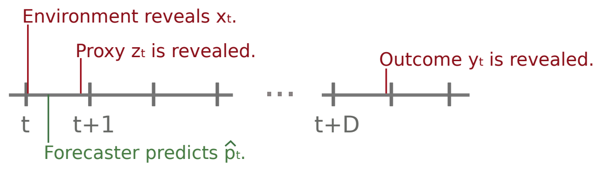


Figure 1. A timeline depicting prediction of a delayed outcome. At the beginning of round t, the environment reveals the instance x_t. Next the forecaster makes a prediction $\hat{p}_t$ given x_t. At the end of round t, the proxy z_t is revealed only after the forecaster makes a prediction. Finally, the outcome y_t is revealed at the end of round $t + D$ (after prediction $\hat{p}_{t+D}$ is made).

2 Formal Problem Description & Approach

The formal setting in this paper is motivated by online marketplaces where we want to predict consumer engagement with purchased products. An instance $x \in \mathcal{X}$ represents features about a user and a content being considered for recommendation, while a label $y \in \mathcal{Y}$ indicates what kind of engagement occurred, with the simplest being $\mathcal{Y} = \{0, 1\}$ where $y = 0$ represents "no engagement" and $y = 1$ represents "successful engagement". A proxy $z \in \mathcal{Z}$ represents an observation made after the prediction, but before the outcome is revealed. Typically, the number of proxy symbols will be much smaller than the number of instances.

Above, $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ are nonempty finite sets representing the sets of potential instances, outcomes, and proxies, respectively. We denote the number of potential instances $|\mathcal{X}|$ by N, and the $(d-1)$-dimensional simplex by Δ^d for any $d \geq 2$. Let $T > 0$ denote the number of prediction rounds, and for simplicity we assume that every outcome is observed with a fixed delay $D \geq 0$ ($D = 0$ implies no delay). Before an episode begins, the environment generates $(x_t, y_t, z_t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ for each round $t \in \{1, 2, \dots, T\}$.

Assumption 1. *Let $g(\cdot|x)$ be a conditional probability distribution over $\mathcal{Z}$ for all $x \in \mathcal{X}$, and $h(\cdot|z)$ be a conditional probability distribution over $\mathcal{Y}$ for all $z \in \mathcal{Z}$. The sequence $\langle x_t \rangle_{t=1}^T$ is generated arbitrarily. The sequence $\langle z_t \rangle_{t=1}^T$ is generated randomly such that $z_t \sim h(\cdot|x_t)$ and is conditionally independent of $\langle z_1, \dots, z_{t-1}, z_{t+1}, \dots, z_T \rangle$ given x_t. Furthermore, the sequence $\langle y_t \rangle_{t=1}^T$ is generated randomly such that and $y_t \sim g(\cdot|z_t)$ and it is conditionally independent of $\langle y_1, \dots, y_{t-1}, y_{t+1}, \dots, y_T \rangle$ given z_t.*

That is, we make no assumption about how the instances are selected, while the proxies and outcomes are generated stochastically from the unknown distributions h and g.

In our theoretical analysis, for clarity, we consider a setup where each outcome is observed with a fixed delay D, while the proxies are revealed with no delay, as described by the following model:[1] At each round $t \geq 1$, using the instance x_t, the forecaster makes a prediction $\hat{p}_t \in \mathcal{P} = \{\phi \mid \phi : \mathcal{X} \to \Delta^{|\mathcal{Y}|}\}$ and incurs instantaneous loss $f_t(\hat{p}_t) = -\ln(\hat{p}_t(y_t|x_t))$. At the end of round t, the forecaster receives a proxy symbol z_t and a delayed outcome y_{t-D} (if $t - D > 0$). Figure 1 depicts a single prediction round with a delayed outcome. The goal of the forecaster is to minimize the (expected) regret

$$\mathbb{E}[\text{Regret}_T] = \min_{\omega \in \mathcal{P}} \mathbb{E}\left[\sum_{t=1}^{T} f_t(\hat{p}_t) - f_t(\omega)\right] \tag{1}$$

over T rounds, where the expectation is taken with respect to the proxies $\langle z_t \rangle_{t=1}^T$ and outcomes $\langle y_t \rangle_{t=1}^T$ (note that $\mathcal{P}$ is compact, so the minimum exists). Under Assumption 1, the optimal predictor is defined by

$$p^*_{\mathrm{F}}(y|x) = \sum_{z \in \mathcal{Z}} g(y|z)h(z|x)$$

for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

In the usual setups considered in the literature, the forecaster does not have access to the proxies and hence needs to deal with the delays of the outcomes directly. Joulani et al. (2016) introduce a strategy for converting a base online learner into a delayed online learner. We refer to this approach or any other approach that learns a direct relationship between instances and outcomes as a *direct forecaster* for delayed outcomes, since it does not make use of the proxy outcomes.

According to Joulani et al. (2013), the regret in adversarial online learning with delay D is at least $\Omega\left((D+1)N\ln\left(\frac{T}{N(D+1)}\right)\right)$. Applying the approach of Joulani et al. (2016), it is not difficult to prove that there exists a forecaster with regret $O\left((D+1)N|\mathcal{Y}|\ln\left(\frac{T}{N}+1\right)\right)$ (see the supplementary material for a formal statement and proof). Notice that both the lower and upper bounds are logarithmic in the number of rounds T but scale multiplicatively (i.e., $D \times N$) with the delay D and the number of instances N. However, this bound is particular to the adversarial setting. If the instances are chosen at each round according to a fixed distribution, then the regret bound scales additively (i.e., $D+N$) (Joulani et al., 2013). Unfortunately, an online marketplace with a stream of new items resembles adversarially generated instances more than independent and identically distributed instances. Our goal is to try to recover an additive scaling between instances and delay even when instances are selected adversarially.

Proposed Approach: Given the model described by Assumption 1, it is natural to consider a factored model

$$\hat{p}_t(y|x_t) = \sum_{z \in \mathcal{Z}} \hat{g}_t(y|z)\hat{h}_t(z|x_t) \ , \tag{2}$$

where $\hat{h}$ approximates the conditional distribution h that generates the proxies and $\hat{g}$ estimates the conditional probability g of the delayed outcome given a proxy. The key advantage of a factored model is that the estimator $\hat{h}$ can be updated at each round based on the proxy. While the estimator $\hat{g}$ can only be updated when a delayed outcome is revealed, its predictions do not depend on specific instances. Note that under Assumption 1, the optimal predictor p^*_{F} has a factored form.

For simplicity, below we assume that $\hat{g}$ and $\hat{h}$ are Laplace estimators given the outcomes: for $t \geq 1$, given a sequence of observations $\langle a_1, \ldots, a_{t-1} \rangle$ taking values in a finite set $\mathcal{A}$, for the tth symbol the Laplace estimator assigns probability $q_t(a) = \frac{\sum_{s=1}^{t-1} \mathbb{I}\{a_s = a\} + \alpha}{t - 1 + |\mathcal{A}|\alpha}$ for all $a \in \mathcal{A}$ with $\alpha = 1$ (Cesa-Bianchi & Lugosi, 2006).[2] The resulting algorithm is shown in Algorithm 1, with its regret analyzed in the next theorem.

Algorithm 1 Factored Forecaster for Delayed Outcomes

```
Require: $\mathcal{X}$ {Instance space.}, $\mathcal{Y}$ {Outcome space.}, $\mathcal{Z}$ {Proxy space.}, $D \geq 0$ {Delay in number of rounds.}, Base {Base forecaster.}, $T \geq 1$ {Total number of rounds.}
 1: Before the game, the environment selects:
      1. instances $\langle x_1, x_2, \ldots, x_T \rangle \in \mathcal{X}^T$ arbitrarily;
      2. proxies $\langle z_1, z_2, \ldots, z_T \rangle \in \mathcal{Z}^T$
         independently with $z_t \sim h(\cdot|x_t)$; and
      3. outcomes $\langle y_1, y_2, \ldots, y_T \rangle \in \mathcal{Y}^T$
         independently with $y_t \sim g(\cdot|z_t)$.
 2: For each $x \in \mathcal{X}$, initialize Base estimator $\hat{h}^{(x)}$.
 3: For each $z \in \mathcal{Z}$, initialize Base estimator $\hat{g}^{(z)}$.
 4: for $t = 1, 2, \ldots, T$ do
 5:     Environment reveals $x_t$.
 6:     Predict $\hat{p}_t$ where
        $\forall_{y \in \mathcal{Y}}\ , \hat{p}_t(y) = \sum_{z \in \mathcal{Z}} \hat{g}^{(z)}(y)\hat{h}^{(x_t)}(z)$.
 7:     Incur loss $f_t(\hat{p}_t) = -\ln(\hat{p}_t(y_t))$.
 8:     Environment reveals $z_t$ and $y_{t-D}$ (if $t - D > 0$).
 9:     Update $\hat{h}^{(x_t)}$ with $z_t$ and $\hat{g}^{(z^{t-D})}$ with $y_{t-D}$ using Base (if $t - D > 0$).
10: end for
```

[1] We consider delayed proxies in our experiments.

[2] A somewhat better, asymptotically optimal estimator can be obtained by selecting $\alpha = 1/2$; this is known as the Krichevsky-Trofimov estimator (Cesa-Bianchi & Lugosi, 2006). While our results can easily be applied to this estimator, we choose to present them with the Laplace estimator because it simplifies the form of our bounds.

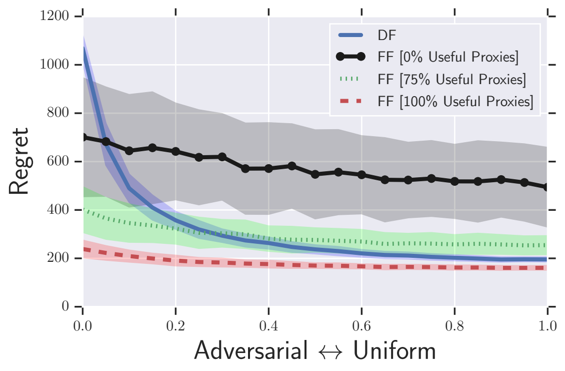


Figure 2. Comparison of regret (averaged over 200 trials) on randomly generated prediction tasks for a direct forecaster (DF) and factored forecasters (FFs) as the instances are generated from an adversarial schedule (towards the left) to a uniform schedule (towards the right). Shaded regions denote ± 1 standard deviation.

Theorem 1. *Suppose that Assumption 1 holds. Let $D \geq 0$ be the delay applied to outcomes. Then the expected regret of Algorithm 1 with a Laplace estimator as* Base *satisfies*

$$\mathbb{E}\left[\mathrm{Regret}_T\right] \leq$$
$$O\Bigg(\underbrace{(D+1)|\mathcal{Y}||\mathcal{Z}|\ln\left(\frac{T}{|\mathcal{Z}|}\right)}_{(a)} + \underbrace{|\mathcal{Z}|N\ln\left(\frac{T}{N}\right)}_{(b)}\Bigg).$$

The proof of the theorem is presented in the supplementary material. There are three terms in the bound of Theorem 1. The first term, (a), depends on the delay D but not on the number of instances N (instead of N it depends on $|\mathcal{Z}|$). The second term, (b), depends on N but not the delay. Thus, the regret scales additively with respect to the delay and the number of instances, while the regret of the direct forecaster scales multiplicatively (i.e., $\Omega(DN)$).

Figure 2 compares the regret of a direct forecaster (DF) and factored forecasters (FFs) in a synthetic prediction domain where the outcomes are generated by an unknown factored model. In this task, $T = 1000$ and $D = 100$. Details of the prediction domain are given in the supplementary material. Towards the left-hand side, the prediction task is generated according to an adversarial schedule that forces the forecaster to make consecutive predictions for the same instance. Towards the right, the schedule is gradually interleaved by a uniform distribution over instances. It is also interesting to consider the case when the factorization assumption (Assumption 1) is violated. The three FF curves differ by the fraction of the times that the proxy signal is sampled from the true model versus uniform noise. With 100% useful proxies, FF has consistent performance, while DF struggles with the adversarially generated schedule. With 0% useful proxies (random noise), FF incurs large regret relative to DF in most cases. However, the case where FF is given 75% useful proxies, demonstrates that FF can outperform DF in adversarial settings even when the proxy signal is diluted with noise. This suggests that the FF approach may be useful even when we cannot identify high-quality proxies.

3 Neural Network Architectures for Learning from Delayed Outcomes

Based on our analysis, we propose three neural network architectures for learning from delayed outcomes via proxies.

Direct forecaster (DF): We use a single neural network to predict the distribution over outcomes given an instance (Figure 3(a)). This approach ignores the proxies.

Factored forecaster (FF): This approach instantiates Algorithm 1 with two neural networks (Figure 3(b) and 3(c)). The first (Figure 3(b)) predicts a distribution over proxies given an instance (that is, it implements $\hat{h}_t$), while the second (Figure 3(c)) predicts an outcome distribution given a proxy (implementing $\hat{g}_t$). The final prediction is obtained according to (2).

Factored with Residual forecaster (RFF): Similarly to the factored approach, this method learns two neural networks (Figure 3(b) and 3(d)). However, the neural network that predicts an outcome distribution has two towers. The first tower only uses the proxy to predict the logits (implementing $\hat{g}_t$ for the probabilities), while the second tower is an instance-dependent residual correction that can help to correct predictions when the proxies are not informative of the delayed outcomes. Denoting the coordinates of the exponentiated Δ-logits by $1 + \delta_t(y|x_t)$ (where δ_t is typically close to 0), the final prediction is obtained as $\hat{p}_t(y|x_t) = \sum_{z\in\mathcal{Z}}(1 + \delta_t(y|x_t, z))\hat{g}_t(y|z)\hat{h}_t(z|x_t)$. We train the residual tower with a separate loss and stop backpropogation from that loss to the first tower. This ensures that the second tower is treated as a residual correction helping to preserve generalization across instances.

For all experiments, unless stated otherwise, we update network weights using Stochastic Gradient Descent[3] with a learning rate of 0.1 minimizing the negative log-loss. Except for the networks predicting the outcome distribution from proxies (Figure 3(c) and left tower in Figure 3(d)), all network towers have two hidden layers. Their output layer is sized appropriately (to output the correct number of class logits) and use a `softmax` activation. We apply L_2 regularization on the weights with a scale parameter of 0.01. The networks predicting the outcome distribution from proxies do not contain hidden layers, they do not use regularization

[3]We also tried other optimizers but found that most were not sensitive enough to changes in the distribution over instances as they keep track of a historical average over gradients.

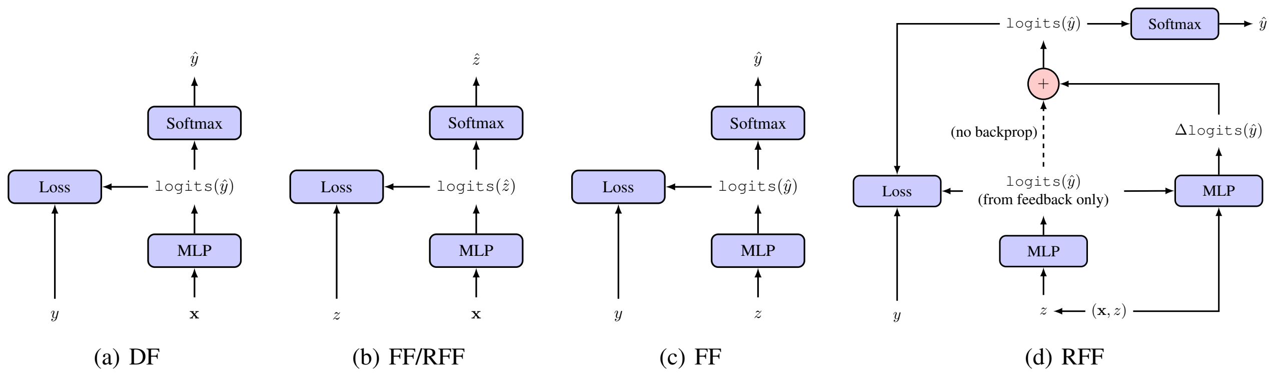


Figure 3. Neural network architectures for delayed prediction, where x denotes an instance, y denotes an outcome, $\hat{y}$ denotes a predicted outcome distribution, z denotes a proxy, and $\hat{z}$ denotes a predicted distribution over proxies. The direct model (a) directly predicts $\hat{y}$ an outcome distribution from an instance x. The factored model is defined by two neural networks (b) and (c), where (b) predicts $\hat{z}$ the proxy distribution given an instance x and (c) predicts $\hat{y}$ an outcome distribution from an proxy z. Finally, the factored with residual model is defined by two networks (b) and (d), where (d) uses both an instance x and proxy z to predict $\hat{y}$ an outcome distribution.

or bias in their output layer. Training samples are stored in a fixed-sized FIFO replay buffer from which we sample uniformly.

For the experiment explained in Section 4.1, the networks predicting the outcome distribution from proxies use a learning rate of 1. Network towers have two hidden layers with 40 and 20 units. The training buffer has a size of 1,000. We start training once we have 128 examples in the buffer and perform one gradient step with a batch size of 128 every four rounds.

For the experiment explained in Section 4.2, the networks predicting the outcome distribution from proxies use a learning rate of 0.1. Network towers have two hidden layers with 20 and 10 units. The training buffer has a size of 3,000. We start training once we have 500 examples in the buffer and perform 20 gradient steps with a batch size of 128 every 1,000 rounds.[4]

In both experiments, hyperparameters such as the learning rate and the coefficient of the L_2 regularization were tuned on DF first, using a simple grid search. The hyperparameters that do not relate to DF (such as the number of hidden layers and the L_2 regularization coefficient of the networks predicting the proxies) are tuned in a second step using a simple grid search, as well.

4 Experiments & Results

We compare the regret of DF, FF, and RFF in two domains: (1) predicting the commit activity of GitHub repositories, and (2) predicting engagement with items acquired from a marketplace.

[4]To simulate less frequent updates to the networks.

4.1 GitHub Commit Activity

The goal is to predict the number of commits made to repositories from GitHub[5] - a popular website hosting open source software projects. Given a repository, the question we want the online learner to answer is “will there be at least three commits in the next three weeks?”. This information could be used to predict churn rate. For example, GitHub could potentially intervene by sending a reminder email. We obtained historical information about commits to GitHub repositories from the BigQuery GitHub database.[6] We started with 100,000 repositories and filtered out repositories with fewer than five unique days with commits between May 1, 2017 and January 8, 2018. This resulted in about 8,300 repositories. In our experiments, an adversary selects both a repository and a timestamp. The outcome is one if there were at least three commits over the 21 days following the chosen timestamp and zero otherwise. The proxy is based on the number of commits over the first seven days following the chosen timestamp. These were mapped to three values: (1) no commits, (2) one commit, and (3) more than one commits. The proxy is delayed by one week and the outcomes are delayed by three weeks. We simulate a scenario where the forecaster receives one sample every 10 minutes.

The adversary initially selects repositories from a subset with a low number of commits - one or fewer commits over the past month. After four and a half weeks, the adversary switches to a distribution that samples from repositories with two or more commits within the past week. Figure 4 shows the outcome probabilities for each proxy under the

[5]http://www.github.com
[6]https://cloud.google.com/bigquery/public-data/github

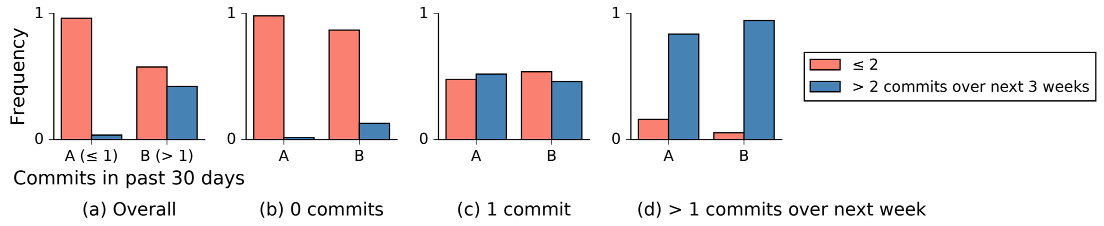


Figure 4. Comparison of outcome distributions for each proxy under different adversarial modes (A and B). (a) shows the change in overall outcome distributions between the two modes, while (b-d) show the marginal distributions conditioned on each proxy.

two distributions used by the adversary. The outcome distribution does not have a fixed relationship with the proxies. Thus, this makes the task more difficult for the factored architectures but the direct architecture is not affected. In this experiment, the historical information about a repository defines an instance. We used binary features to represent the programming languages present in a repository as well as time bucketized counts of historical commit activity for that repository.

To calculate the regret, we subtract the loss of an (almost) optimal forecaster. Since we do not have access to an optimal forecaster, we trained two models with the same architecture as DF on both modes used by the adversary, on non-delayed data. We trained these models using the `Adam` optimizer for 10,000 steps and using an initial learning rate of 0.0005.

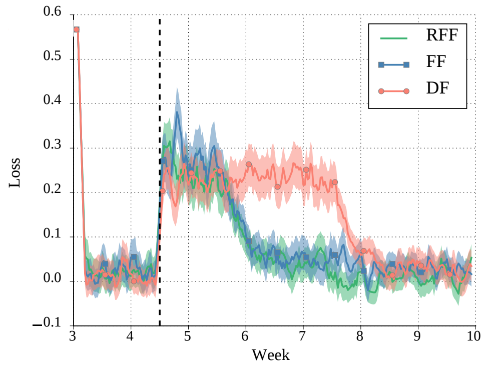
Figure 5(a) compares the loss of the direct and factored architectures averaged over 200 independent trials. The vertical dashed line indicates the time at which the adversary switches from its initial distribution to a distribution over high commit repositories. Due to the outcome delay, all algorithms suffer the same initial loss for roughly 3 weeks. Then all three algorithms quickly achieve a low loss until the adversary changes the distribution over repositories. Finally, *the factored architectures (FF and RFF) recover roughly 2.5 weeks more quickly than DF* (as can be seen in Figure 5(a) from week 5.5–8). Figure 5(b) shows the regret for the same experiment. FF and RFF achieve smaller regret (than DF) as they are better able to adapt to changing distributions. We can also clearly observe that FF is not able to maintain as low loss as RFF, since it is unable to correct the error coming from the slightly incorrect factorization assumption.

4.2 Engagement with Marketplace Items

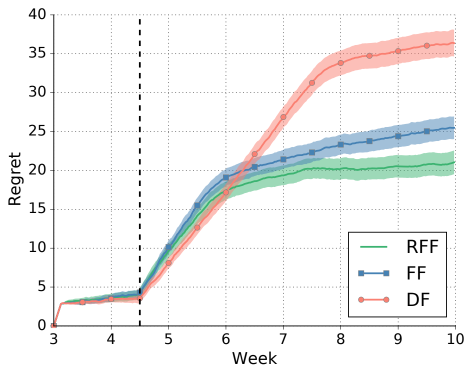
For a popular marketplace with personalized recommendations, we want to predict the probability that after acquiring an item a user will (1) engage with that item more than once, and (2) not delete that item within 7 days after acquiring it (e.g., a user who downloads a new ebook will read at least two chapters and not delete this ebook from their device). The proxies are measured two days after an item is acquired and the possible outcomes are: (1) *Deleted:* The item was deleted. (2) *Zero Engagements:* The user engaged with the item zero times but did not delete it. (3) *One Engagement:* The user engaged with the item exactly once and did not delete it. (4) *Many Engagements:* The user engaged with the item two or more times and did not delete it. The outcomes are the same as the proxies but measured seven days after conversion.

We collected 21 days of data between the 10th and 31st of January 2018. For each item with at least 100 conversions, we stored the empirical probability of each proxy and the probability of each outcome conditioned on proxies. In addition, we stored the empirical probability that an item was acquired (if shown) on each of the 21 days. We consider two distribution schedules (Figure 6). The first schedule, which we refer to as Staggered (Figure 6(a)), subsampled 5 items for each day (at noon) that appear for the first time on that day in the marketplace (i.e., the candidate items with higher indices only appear on increasingly later days). This schedule simulates items continually being added to the marketplace. This is a very hard setting in which we expect DF to perform poorly, due to the need to constantly make predictions about new instances. The second schedule, which we refer to as Uniform, is derived by subsampling 100 items uniformly (Figure 6(b)) and creating a 21 day schedule with half-day intervals (for the afternoons, the distributions are obtained as the average distributions for the actual and the next days, based on the frequencies from the 21 logged days). This schedule is more favorable for DF because instances are sampled according to a slowly shifting distribution. Overall, the proxies are delayed by two days and the outcomes are delayed by seven days. We simulated a scenario where the forecaster receives one sample every 40 seconds.

In this experiment, we encoded an item/instance using a 100-dimensional one-hot encoding specifying items by index. Similarly to the GitHub experiment, to calculate the regret, we subtract the loss of an (almost) optimal forecaster. We trained 42 models (one for each half-day interval) with the same architecture as DF on all modes used by the adversary (one model for each half-day interval). For each model, we

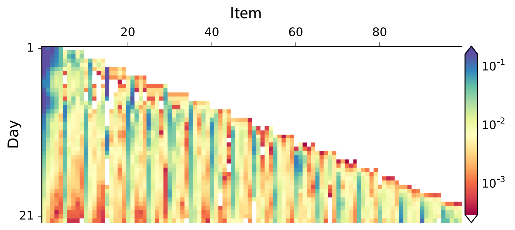


(a)

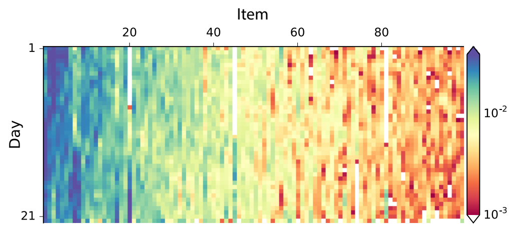


(b)

Figure 5. Comparisons of (a) average loss and (b) regret (averaged over 200 independent trials with (a) further averaged over five consecutive datapoints) in a task where the adversary shifts the distribution of instances partway through the episode (indicated by the dashed vertical line). We measure regret with respect to an optimally trained model. The shaded areas represent 95% confidence intervals.

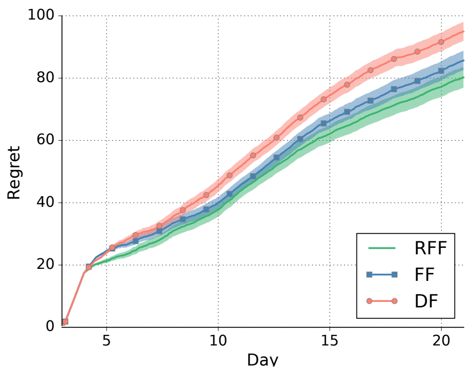


(a) Staggered Schedule

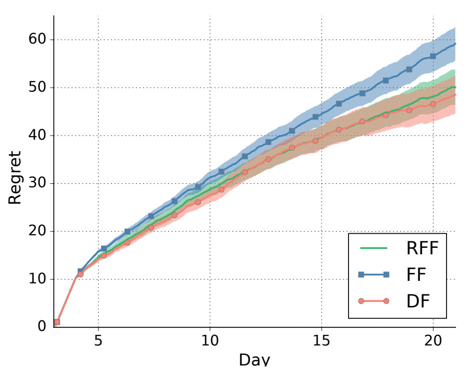


(b) Uniform Schedule

Figure 6. Two different item distribution schedules derived from an actual marketplace. The X-axis corresponds to 100 item indices sorted in decreasing order of total conversions, while the Y-axis corresponds to days. (a) Empirical distribution (logarithmic scale) with new items being added each day. (b) More stable empirical distribution of uniformly sampled items.



(a)



(b)

Figure 7. Regret (and 95% confidence intervals) of DF, FF, and RFF averaged over 200 independent trials on items sampled from different schedules. (a) In the staggered schedule, the factored approaches outperform DF, which is crippled by delay. (b) The uniform schedule demonstrates how RFF can achieve similar performance to DF, when the factorization assumption does not help. (Note the difference in the scale of the Y-axis).

used the Adam optimizer and trained for 40,000 steps using non-delayed data, with an initial learning rate of 0.0005.

Figure 7 compares the average regret of all three forecasters over 200 independent trials for both adversarial schedules. For the Staggered schedule (Figure 7(a)), FF and RFF outperform DF, as expected. DF must wait until the outcomes become available, but FF and RFF are able to generalize to new instances. RFF performs slightly better than FF, indicating that RFF can mitigate the incorrect factorization assumption as delayed outcomes become available.

The Uniform schedule (Figure 7(b)) is easier since the distribution over instances is shifting slowly. DF, FF, and RFF all achieve smaller regret compared to the Staggered schedule. In the Uniform schedule, DF outperforms FF because the factorization assumption is violated by the data. However, RFF achieves similar results to DF because its residual tower allows it to mitigate the error introduced by the incorrect factorization assumption.

5 Related Work

Chapelle (2014) proposes a model for learning from delayed conversions. However, this approach does not take advantage of potential proxies for delayed outcomes. Learning from delayed outcomes is also related to survival analysis (Yu et al., 2011; Fernández et al., 2016), where the goal is to model the time until a delayed event. A significant difference of our work is the use of proxies.

A large body of literature exists in online learning for both the adversarial and stochastic partial monitoring settings with delayed outcomes, analyzing the regret (see, e.g., Weinberger & Ordentlich, 2002; Mesterharm, 2005; Agarwal & Duchi, 2011; Joulani et al., 2013; 2016). However, none of the settings takes potential proxies into consideration to mitigate the impact of delay.

Cesa-Bianchi et al. (2018) consider an adversarial learning problem with delayed bandit-based feedback. In their setting, the observed loss function is the sum of D adversarially chosen loss functions for the actions taken in the previous D rounds. This differs significantly from our setting where proxy signals are *additional* information to the forecaster. Furthermore, our analysis does not restrict the proxies to a particular relationship with the delayed outcomes.

In a fully stochastic online learning setting, where the instances and outcomes are sampled from the same distribution at each round, the regret (Joulani et al., 2013) or mistake bounds (Mesterharm, 2005) scale with $D + N$ (i.e., delay plus number of instances; the exact dependence on the time horizon T depends on the loss function). However, the stochastic assumption is not realistic for online marketplaces because new products are being added on a regular basis. Thus, the distribution over instances is not independent and identically distributed from day to day.

6 Discussion

We have presented a way to leverage proxies to learn faster in scenarios where the outcomes are significantly delayed. Our theoretical analysis shows that under our factorization assumption, the regret of the factored approach which exploits intermediate observations scales as $D + N$ unlike a naive approach that scales as $D \times N$.

While we have presented instances and proxies as disjoint sets, note that in practice, we can encode some information from an instance into the proxy. For example, in the motivating example, an instance corresponds to a book. From the instance, we can extract features that may generalize, such as the length of the ebook. A proxy could encode both how much the customer has read so far and the length of the ebook. This technically increases the number of proxy symbols, but could significantly improve the predictive power of a factored forecaster.

While we have focused on full-information online learning, it may also be interesting to consider learning with bandit feedback. However, applying bandit algorithms in a production setting can be challenging, since decisions are typically made by several components (not a single agent). Nevertheless, this is an interesting direction of future work.

We have presented experimental results on a dataset from GitHub as well as a dataset from a real marketplace, and showed that our algorithms can learn faster when the chosen proxies are helpful, and can gracefully recover the baseline performance when the selected proxies are not informative of the delayed outcomes. We believe that the proposed approach can be beneficial in many real-world applications where the goal is to optimize for long-term value. It would be interesting to extend our theoretical analysis to ranking measures.

Acknowledgements

We would like to thank David Silver for helpful discussions regarding learning from delayed signals and Tor Lattimore and Todd Hester for reviewing this manuscript.

References

Agarwal, A. and Duchi, J. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 873–881, 2011.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Cesa-Bianchi, N., Gentile, C., and Mansour, Y. Nonstochastic bandits with composite anonymous feedback. In Bubeck, S., Perchet, V., and Rigollet, P. (eds.), *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pp. 750–773, 2018.

Chapelle, O. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1097–1105, 2014. ISBN 978-1-4503-2956-9.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7–10, 2016.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 2nd edition, 2006.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., et al. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296. ACM, 2010.

Fernández, T., Rivera, N., and Teh, Y. W. Gaussian processes for survival analysis. In *Advances in Neural Information Processing Systems*, pp. 5021–5029, 2016.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Joulani, P., György, A., and Szepesvári, C. Online learning under delayed feedback. In *Proceedings of the* 30^{th} *International Conference on Machine Learning*, 2013.

Joulani, P., György, A., and Szepesvári, C. Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms. In *AAAI*, volume 16, pp. 1744–1750, 2016.

Mesterharm, C. On-line learning with delayed label feedback. In *International Conference on Algorithmic Learning Theory*, pp. 399–413. Springer, 2005.

Quanrud, K. and Khashabi, D. Online learning with adversarial delays. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1270–1278. Curran Associates, Inc., 2015.

Weinberger, M. J. and Ordentlich, E. On delayed prediction of individual sequences. *IEEE Transactions on Information Theory*, 48(7):1959–1976, 2002.

Yu, C.-N., Greiner, R., Lin, H.-C., and Baracos, V. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In *Advances in Neural Information Processing Systems*, pp. 1845–1853, 2011.

A Analysis of the Direct Forecaster

Algorithm 2 Direct Forecaster for Delayed Outcomes

```
Require: $\mathcal{X}$ {Instance space.}, $\mathcal{Y}$ {Outcome space.}, $D \geq 0$ {Delay in # rounds.}, Base {Base forecaster.}, $T \geq 1$ {Total rounds.}
1: Before the game, the environment selects:
     1. instances $\langle x_1, x_2, \ldots, x_T \rangle \in \mathcal{X}^T$ arbitrarily, and
     2. instances $\langle y_1, y_2, \ldots, y_T \rangle \in \mathcal{Y}^T$ arbitrarily.
2: For each $x \in \mathcal{X}$, initialize Base estimator $\hat{p}^{(x)}$.
3: for $t = 1, 2, \ldots, T$ do
4:     Environment reveals $x_t$.
5:     Predict $\hat{p}_t = \hat{p}^{(x_t)}$.
6:     Incur loss $f_t(\hat{p}_t) = -\ln(\hat{p}_t(y_t))$.
7:     Update $\hat{p}^{(x_{t-D})}$ with $y_{t-D}$ (if $t - D > 0$).
8: end for
```

In this section we formally introduce and analyze a direct forecaster. The forecaster, given in Algorithm 2, creates a separate estimator for each instance $x \in \mathcal{X}$. Thus, it converts the problem into $|\mathcal{X}|$ subproblems. Similarly to the factored forecaster, we consider the case when the Base forecasters (for each $x \in \mathcal{X}$) are of the form

$$\hat{p}_t(y|x) = \frac{\sum_{s=1}^{t-1} \mathbb{I}\{x_s = x \wedge y_s = y\} + \alpha}{\sum_{s=1}^{t-1} \mathbb{I}\{x_s = x\} + \alpha|\mathcal{Y}|}\,, \tag{3}$$

for some $\alpha \geq 0$ where the sums are defined to be 0 if $t \leq 1$ (notable special cases are the Krichevsky-Trofimov estimator for $\alpha = 1/2$ and the Laplace estimator for $\alpha = 1$, see, e.g., Cesa-Bianchi & Lugosi 2006). For $t \leq 0$, $\hat{p}_t(y|x)$ is defined to be 1 for any pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Following Joulani et al. (2016), we characterize the effect of the delayed observations by considering the so-called prediction drift of the non-delayed forecaster Base given in (3), which looks at the loss difference of the predictions of the non-delayed and the delayed forecasters, $\hat{p}_s(y_s|x_s)$ and $\hat{p}_{s-D}(y_s|x_s)$, respectively.

Lemma 1. *Let $\alpha > 0$ and $D \geq 0$. Then for any sequence $\langle (x_s, y_s) \rangle_{s=1}^T \in (\mathcal{X} \times \mathcal{Y})^T$, predictor (3) satisfies*

$$\sum_{s=1}^{T} \ln\left(\frac{\hat{p}_s(y_s|x_s)}{\hat{p}_{s-D}(y_s|x_s)}\right) \leq D|\mathcal{X}||\mathcal{Y}| \ln\left(\frac{T-1}{\alpha|\mathcal{X}||\mathcal{Y}|} + 1\right) \tag{4}$$

The expression on the left-hand side of (4) is called the cumulated prediction drift.

Proof. For any t, define $\#_t(x) = \sum_{s=1}^{t-1} \mathbb{I}\{x_s = x\}$ and $\#_t(y, x) = \sum_{s=1}^{t-1} \mathbb{I}\{y_s = y \wedge x_s = x\}$, where the sums are defined to be 0 if $t \leq 1$. Then prediction drift can be bounded as follows:

$$\begin{aligned}
\sum_{s=1}^{T} \ln\left(\frac{\hat{p}_s(y_s|x_s)}{\hat{p}_{s-D}(y_s|x_s)}\right) &= \sum_{s=1}^{T} \ln\left(\frac{\frac{\#_s(y_s,x_s)+\alpha}{\#_s(x_s)+\alpha|\mathcal{Y}|}}{\frac{\#_{s-D}(y_s,x_s)+\alpha}{\#_{s-D}(x_s)+\alpha|\mathcal{Y}|}}\right) \leq \sum_{s=1}^{T} \ln\left(\frac{\#_s(y_s,x_s)+\alpha}{\#_{s-D}(y_s,x_s)+\alpha}\right) \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \sum_{s=1}^{\#_T(x,y)} \Big(\ln(\#_s(y,x)+\alpha) - \ln(\#_{s-D}(y,x)+\alpha)\Big) \\
&\leq \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} D\Big(\ln(\#_T(x,y)+\alpha) - \ln(\alpha)\Big) = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} D \ln\left(\frac{\#_T(x,y)}{\alpha} + 1\right)\,,
\end{aligned} \tag{5}$$

where the first inequality holds since $\#_{s-D}(x_s) \leq \#_s(x_s)$, while the second inequality holds since at most the last D positive and the first D negative terms are not canceled in the middle row. By Jensen's inequality, using the concavity of

$\ln(\cdot)$ and that $\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \#_T(x,y) = T-1$, (5) can be bounded from above as

$$(5) \le D|\mathcal{X}||\mathcal{Y}| \ln\left(\frac{T-1}{\alpha|\mathcal{X}||\mathcal{Y}|}+1\right) \tag{6}$$

finishing the proof of the lemma. □

The following theorem bounds the regret of Algorithm 2 with Laplace estimators as Base. The analysis follows that of Joulani et al. (2016) and is presented for completeness.

Theorem 2. *Let $D \ge 0$ be the delay applied to outcomes, $\mathcal{X}$ be the set of instance symbols, $\mathcal{Y}$ be the set of possible outcomes, and $\langle (x_1,y_1),(x_2,y_2),\dots,(x_T,y_T)\rangle$ be an arbitrary sequence of instance and outcome pairs. For any competitor $p \in \{\phi \mid \phi : \mathcal{X} \to \Delta^{|\mathcal{Y}|}\}$, the regret of Algorithm 2 with a Laplace estimator as* Base *satisfies*

$$\mathrm{Regret}_T(p) = \sum_{t=1}^{T} \ln\left(\frac{p(y_t|x_t)}{\hat{p}_{t-D}^{(x_t)}(y_t)}\right) \le O\left((D+1)|\mathcal{Y}||\mathcal{X}|\ln\left(\frac{T}{|\mathcal{X}|}+1\right)\right) . \tag{7}$$

Proof. Let $R(T,\mathcal{Y})$ be a concave function upper bounding the regret of Base applied to an arbitrary sequence of length T with symbols from $\mathcal{Y}$. First we rewrite the regret of the algorithm with delayed feedback as the sum of the non-delayed regret and the prediction drift:

$$\begin{aligned}\mathrm{Regret}_T(p) &= \sum_{t=1}^{T} f_t(\hat{p}_{t-D}) - f_t(p) = \sum_{t=1}^{T} \ln\left(\frac{p(y_t|x_t)}{\hat{p}_{t-D}^{(x_t)}(y_t)}\right)\\ &= \underbrace{\sum_{t=1}^{T} \ln\left(\frac{p(y_t|x_t)}{\hat{p}_{t}^{(x_t)}(y_t)}\right)}_{(a)} + \underbrace{\sum_{t=1}^{T} \ln\left(\frac{\hat{p}_{t}^{(x_t)}(y_t)}{\hat{p}_{t-D}^{(x_t)}(y_t)}\right)}_{(b)} .\end{aligned}$$

Term (b) can be bounded by Lemma 1. Next, we consider term (a). Writing it as the sum of regret over prediction tasks for each instance $x \in \mathcal{X}$, we obtain

$$\begin{aligned}\sum_{t=1}^{T} \ln\left(\frac{p(y_t|x_t)}{\hat{p}_{t}^{(x_t)}(y_t)}\right) &= \sum_{x\in\mathcal{X}}\sum_{t=1}^{T} \mathbb{I}\{x=x_t\} \ln\left(\frac{p(y_t|x)}{\hat{p}_{t}^{(x)}(y_t)}\right)\\ &\le \sum_{x\in\mathcal{X}} R\left(\sum_{t=1}^{T}\mathbb{I}\{x=x_t\},\mathcal{Y}\right)\\ &\le |\mathcal{X}| R\left(\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\sum_{t=1}^{T}\mathbb{I}\{x=x_t\},\mathcal{Y}\right)\\ &= |\mathcal{X}| R\left(\frac{T}{|\mathcal{X}|},\mathcal{Y}\right) ,\end{aligned}$$

where the first inequality follows since R is an upper bound on the regret of each individual prediction task, while the second inequality follows from applying Jensen's inequality to the concave function R. By Remark 9.3 in Cesa-Bianchi & Lugosi (2006), for the Laplace and the Krichevsky-Trofimov estimator, $R(T,\mathcal{Y}) = O(|\mathcal{Y}|\ln T)$. Thus, term (a) is bound by $O\left(|\mathcal{X}||\mathcal{Y}|\ln\left(\frac{T}{|\mathcal{X}|}\right)\right)$. Combining with the bound on term (b) finishes the proof.

□

B Proof of Theorem 1

In this section we prove Theorem 1. We start with presenting an intermediate bound on the regret, which allows us to directly use regret bounds available for the Base algorithm, given in Theorem 2.

Lemma 2. *Suppose Assumption 1 holds. Then the expected regret of Algorithm 1 can be bounded as*

$$\mathbb{E}\left[\mathrm{Regret}_T\right] \le \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{g(y_t|z_t)}{\hat{g}_t(y_t|z_t)}\right)\right] + \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{h(z_t|x_t)}{\hat{h}_t(z_t|x_t)}\right)\right] . \tag{8}$$

Proof. By definition,

$$\begin{aligned}\mathbb{E}\left[\mathrm{Regret}_T\right] &= \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{\sum_{z'\in\mathcal{Z}} g(y_t|z')h(z'|x_t)}{\sum_{z\in\mathcal{Z}} \hat{g}_t(y_t|z)\hat{h}_t(z|x_t)}\right)\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{T} \frac{1}{p^*_{\mathrm{F}}(y_t|x_t)}\left(\sum_{z\in\mathcal{Z}} g(y_t|z)h(z|x_t)\right) \ln\left(\frac{\sum_{z\in\mathcal{Z}} g(y_t|z)h(z|x_t)}{\sum_{z\in\mathcal{Z}} \hat{g}_t(y_t|z)\hat{h}_t(z|x_t)}\right)\right] .\end{aligned}$$

Applying the log-sum inequality (Cover & Thomas, 2006, Theorem 2.7.1)[7] for the inner summations over z, we obtain

$$\mathbb{E}_S\left[\mathrm{Regret}_T\right] \le \mathbb{E}_S\left[\sum_{t=1}^{T}\sum_{z\in\mathcal{Z}} \frac{g(y_t|z)h(z|x_t)}{p^*_{\mathrm{F}}(y_t|x_t)} \ln\left(\frac{g(y_t|z)h(z|x_t)}{\hat{g}_t(y_t|z)\hat{h}_t(z|x_t)}\right)\right] \tag{9}$$

Now define z'_t conditionally independently of the other variables $\langle x_s, y_s, z_s, z'_s\rangle_{s\neq t}$ via the conditional distribution $\Pr[z'_t = z|x_t, y_t] = \frac{g(y_t|z)h(z|x_t)}{p^*_{\mathrm{F}}(y_t|x_t)}$. Then, under our assumptions, z_t and z'_t have the same conditional distribution given x_t and y_t, and so by the independence of z_t and $\hat{g}_t$ and $\hat{h}_t$,

$$\begin{aligned}(9) &= \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{E}_{z'_t}\left[\ln\left(\frac{g(y_t|z'_t)h(z'_t|x_t)}{\hat{g}_t(y_t|z'_t)\hat{h}_t(z'_t|x_t)}\right)\Big|x_t, y_t\right]\right] = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{E}_{z_t}\left[\ln\left(\frac{g(y_t|z_t)h(z_t|x_t)}{\hat{g}_t(y_t|z_t)\hat{h}_t(z_t|x_t)}\right)\Big|x_t, y_t\right]\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{g(y_t|z_t)h(z_t|x_t)}{\hat{g}_t(y_t|z_t)\hat{h}_t(z_t|x_t)}\right)\right] = \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{g(y_t|z_t)}{\hat{g}_t(y_t|z_t)}\right)\right] + \mathbb{E}\left[\sum_{t=1}^{T} \ln\left(\frac{h(z_t|x_t)}{\hat{h}_t(z_t|x_t)}\right)\right] .\end{aligned}$$

□

Now we are ready to prove Theorem 1.

Proof of Theorem 1. To prove the theorem, we bound the two terms on the right-hand side of (8) in Lemma 2 separately. Applying Theorem 2 (with $D \leftarrow D$, $\mathcal{X} \leftarrow \mathcal{Z}$, and $\mathcal{Y} \leftarrow \mathcal{Y}$), the first term can be bounded by

$$O\left((D+1)|\mathcal{Y}||\mathcal{Z}| \ln\left(\frac{T}{|\mathcal{Z}|}\right)\right) .$$

Writing the second term as

$$\sum_{t=1}^{T} \ln\left(\frac{h(z_t|x_t)}{\hat{h}_t(z_t|x_t)}\right) = \sum_{x\in\mathcal{X}}\sum_{t=1}^{T} \mathbb{I}\{x_t = x\} \ln\left(\frac{h(z_t|x)}{\hat{h}_t(z_t|x)}\right),$$

the inner sum can be bounded by the regret of the prediction algorithm $\hat{h}^{(x)}$ applied to a sequence of length $T_x = \sum_{t=1}^{T} \mathbb{I}\{x_t = x\}$, which is $O(|\mathcal{Z}| \ln T_x)$ by Remark 9.3 of Cesa-Bianchi & Lugosi (2006). Thus,

$$\sum_{t=1}^{T} \ln\left(\frac{h(z_t|x_t)}{\hat{h}_t(z_t|x_t)}\right) = O\left(\sum_{x\in\mathcal{X}} |\mathcal{Z}| \ln T_x\right) \le O\left(|\mathcal{Z}| N \ln\left(\frac{T}{N}\right)\right)$$

where the last inequality follows from Jensen's inequality since $\sum_{x\in\mathcal{X}} T_x = T$. This completes the proof of the theorem. □

[7] For any non-negative numbers $a_1, \ldots, a_T$ and $b_1, \ldots, b_T$, $\left(\sum_{i=1}^{T} a_i\right) \ln \frac{\sum_{i=1}^{T} a_i}{\sum_{i=1}^{T} b_i} \le \sum_{i=1}^{T} a_i \ln \frac{a_i}{b_i}$, where, by convention, $0 \ln 0 = 0$, $a \ln \frac{a}{0} = \infty$ if $a > 0$ and $0 \ln \frac{0}{0} = 0$.

C Synthetic Prediction Task

In this section we describe precisely how the synthetic prediction task used at the end of Section 2 is defined. We generated a prediction task by sampling two stochastic matrices (with rows summing up to 1) – an $|\mathcal{X}| \times |\mathcal{Z}|$ matrix H (instances to proxies) and a $|\mathcal{Z}| \times |\mathcal{Y}|$ matrix G (proxies to outcomes). To generate these matrices, we first created a matrix $R_{\mathcal{Z}}$ (and $R_{\mathcal{Y}}$) where each row was a one hot encoding of an element sampled uniformly from $\mathcal{Z}$ (or $\mathcal{Y}$). Next we generated a matrix $U_{\mathcal{Z}}$ (and $U_{\mathcal{Y}}$) with a uniform distribution in each row. The final matrix $P_{\mathcal{Z}}$ (and $P_{\mathcal{Y}}$) was created by interpolating $P_{\mathcal{Z}} = (1-\epsilon)R_{\mathcal{Z}} + \epsilon U_{\mathcal{Z}}$ (and $P_{\mathcal{Y}} = (1-\epsilon)R_{\mathcal{Y}} + \epsilon U_{\mathcal{Y}}$) for some $\epsilon \in [0, 1]$.

The instances were $\mathcal{X} = \{1, 2, \ldots, N\}$. Let $\mu \in [0, 1]$, and $U_1, U_2, \ldots, U_T$ be independent, uniform random variables over $\mathcal{X}$. The instances were selected according to

$$x_t = \begin{cases} U_t & \text{with probability } \mu\text{; and} \\ \min(N, \lfloor t/D \rfloor + 1) & \text{otherwise.} \end{cases}$$

Once the instances were selected, the proxy was selected by sampling $z_t \in \mathcal{Z}$ according the distribution specified by the x_t^{th} row of the matrix H. Finally, the outcome was selected by sampling $y_t \in \mathcal{Y}$ according to the distribution specified by the z_t^{th} row of the matrix G.

When $\mu = 0$, the schedule is adversarial with a pattern that looks like:

$$x_1 = 1, x_2 = 1, \ldots, x_D = 1, x_{D+1} = 2, x_{D+2} = 2, \ldots, x_{2D} = 2, ..., x_{ND} = N, x_{ND+1} = N, \ldots, x_T = N \enspace .$$

On the other hand, when $\mu = 1$, the instances are sampled uniformly from $\mathcal{X}$ at each round.

In our experiments we used the following parameter choices: the number of rounds $T = 1000$, the delay $D = 100$, the number of instances $N = 10$, the number of proxies $|\mathcal{Z}| = 4$, and the number of outcomes $|\mathcal{Y}| = 5$.