Learning to Predict Independent of Span

Hado van Hasselt [*]   Richard S. Sutton [†]

August 20, 2015

Abstract

We consider how to learn multi-step predictions efficiently. Conventional algorithms wait until observing actual outcomes before performing the computations to update their predictions. If predictions are made at a high rate or span over a large amount of time, substantial computation can be required to store all relevant observations and to update all predictions when the outcome is finally observed. We show that the exact same predictions can be learned in a much more computationally congenial way, with uniform per-step computation that does not depend on the span of the predictions. We apply this idea to various settings of increasing generality, repeatedly adding desired properties and each time deriving an equivalent span-independent algorithm for the conventional algorithm that satisfies these desiderata. Interestingly, along the way several known algorithmic constructs emerge spontaneously from our derivations, including dutch eligibility traces, temporal difference errors, and averaging. This allows us to link these constructs one-to-one to the corresponding desiderata, unambiguously connecting the 'how' to the 'why'. Each step, we make sure that the derived algorithm subsumes the previous algorithms, thereby retaining their properties. Ultimately we arrive at a single general temporal-difference algorithm that is applicable to the full setting of reinforcement learning.

1 Learning long-term predictions

The *span* of a multi-step prediction is the number of steps elapsing between when the prediction is made and when its target or ideal value is known. We consider the case in which predictions are made repeatedly, at each of a sequence of discrete time steps. For example, if on each day we predict what a stock market index will be in 30 days, then the span is 30, whereas if we predict at each hour what the stock market index will be in 30 days, then the span is $30 \times 24 = 720$.

The span may vary for individual predictions in a sequence. For example, if we predict on each day what the stock-market index will be at the end of the year, then the span will be much longer for predictions made in January than it is for predictions made in

---

[*]Google DeepMind

[†]Reinforcement Learning and Artificial Intelligence Laboratory
 Department of Computing Science, University of Alberta
 Edmonton, Alberta, Canada T6G 2E8

December. If the span may vary in this way, then we consider the span of the prediction sequence to be the maximum possible span of any individual prediction in the sequence. For example, the span of a daily end-of-year stock-index prediction is 365. Often the span is infinite. For example, in reinforcement learning we often learn value functions that are predictions of the discounted sum of all future rewards in the potentially infinite future (Sutton and Barto 1998).

In this paper we consider computational and algorithmic issues in efficiently learning long-term predictions, defined as predictions of large integer span. Predictions could be long term in this sense either because a great deal of clock time passes, as in predicting something at the end of the year, or because predictions are made very often, with a short time between steps (e.g., as in high-frequency financial trading). The per-step computational complexity of some algorithms for learning accurate predictions depends on the span of the predictions, and this can become a significant concern if the span is large. Therefore, we focus on the construction of learning algorithms whose computational complexity per time step (in both time and memory) is constant (does not scale with time) and independent of span.

This paper features two recurring themes, the first of which is the repeated spontaneous emergence of, often well-known, algorithmic constructs, directly from our derivations. We start each derivation by formalizing a desired property and constructing an algorithm that fulfills it, without considering computationally efficiency. Then, we derive a span-independent algorithm that results on each step in exactly the same predictions. Interestingly, each time a specific algorithmic construct emerges, demonstrating a clear connection between the desideratum (the 'why') and the algorithmic construct (the 'how'). For instance, the desire to be independent of span leads to a dutch eligibility trace, which was previously derived only in the more specific context of online temporal difference (TD) learning (van Seijen and Sutton 2014).

The second theme is that we unify the algorithms at each step. Each time, we make sure to obtain an algorithm that is strictly more general than the previous ones, so that in the end we obtain one single algorithm that can fulfill all the desiderata while remaining computationally congenial.

2   Outline of the paper

In this section, we briefly describe the high-level narrative of the paper, without going into technical detail. In each of the Sections 3 to 8, we describe and formalize one or more desirable properties for our algorithms and then derive a computationally congenial algorithm that achieves this exactly. We build up to the final, most general, algorithm that is ultimately derived in Section 8 to highlight the connections between desired properties and algorithmic constructs. Making these connections clear is one of the main goals of this paper.

Specifically, in Section 3 we derive a span-independent algorithm to update the predictions for a single final outcome. The algorithm is *offline* in the sense that does not change its predictions before observing the outcome. The dutch trace emerges spontaneously, which shows that this trace is closely tied to the requirement of span-independent computation. This emergence is surprising and intriguing because it shows that these traces

are not specific to online TD learning, for which they were first proposed (van Seijen and Sutton 2014).

In Section 4 we derive span-independent updates that update the predictions *online*, towards interim targets that temporarily stand in for the final outcome. We show that the desire to be online results in the spontaneous emergence of TD errors (Sutton 1984; Sutton 1988). In this paper we are mostly agnostic to the origin of the interim targets. These may for instance be given by external experts or by own online predictions, as in standard TD learning (e.g., see Sutton and Barto 1998).

It can be beneficial to be able to switch smoothly between online and offline updates, on a step-by-step basis, for instance when we do not full trust some of the interim targets that we would use for our online updates. This allows us to have the best of both worlds: the online predictions stay trustworthy even if some interim targets are wrong, and we are still able to use any useful information immediately when it is observed. In Section 5 we consider how to do this efficiently and from our derivation an update emerges that averages the online weights in a separate trusted weight vector. This is interesting because such averaging is known to improve the convergence rates of online learning algorithms (Polyak and Juditsky 1992; Bach and Moulines 2013), but seems to only rarely be used in reinforcement learning (as noted, e.g., by Szepesvári 2010).

Some interim targets may be so informative that we want their effect to persist in the predictions even after observing the final outcome. For instance, if the final outcome is stochastic and the interim targets are drawn independently from the same distribution it makes sense to average these instead of committing fully only to the final outcome. In the extreme, we might see an interim target that we trust so much that we do not even care about the actual outcome anymore, for instance because the interim target already takes into account all possible outcomes from that moment rather than only the specific one that will happen to materialize this time, resulting in a more accurate prediction on average than a single final outcome. In Section 6, we formalize these ideas and show they lead naturally to a form of TD(λ) (Sutton 1988; Sutton and Barto 1998).

The λ parameter that governs the amount of persistency of the interim targets can be interpreted as representing a degree of trust: if we trust an interim target fully ($\lambda = 0$) we do not need to consider later observations, while if we distrust it fully ($\lambda = 1$) it will be replaced by later targets and leave no trace in the final predictions. This is a different notion of trust than the one considered for the smooth switching between online and offline updates, where the trust was relative to the actual final outcome rather than the expected outcome. These two forms of trust are compatible and complimentary, and in Section 7 we show how to combine them into a single algorithm.

Up to Section 7, we have only considered predicting a single final outcome in an episodic setting. In Section 8 we consider how to deal with two important generalizations of the problem setting: cumulative returns, and soft terminations. Cumulative returns allow us to see part of the return on each step, and allow us to start learning from these partial returns immediately in the online setting. Soft terminations allow us to learn about predictions that may conditionally terminate even if the actual process continues, and they allow for non-episodic predictions that may terminal softly on each step rather than completely at a single point in time. This leads to a single final algorithm that subsumes all previous algorithms as special cases. The algorithm is similar to the conventional TD(λ) algorithm but with important differences that ensure that it is exactly equivalent to the desired, but

inefficient, algorithm and therefore inherits all its desirable properties.

Because our final algorithm is novel, it is appropriate to analyze it. In Section 9 we prove that the algorithm is convergent under typical mild conditions, and that it converges to the same solution as similar previous algorithms, including TD(λ).

We conclude the paper with a short discussion in Section 10.

3 Independence of span and the emergence of traces

We start with a supervised learning setting—predicting the final numeric outcome of an episodic process. An episode of the process starts at time $t = 0$ and moves stochastically from state to state generating feature vectors ϕ_t until termination with a final numeric outcome Z at final time T. For example, Z could be the price of a particular stock that we want to predict, and each episode may be a year, such that time T corresponds to the end of the year.

We consider the general case of multi-step predictions ($T > 1$), where a prediction is made on each step. The standard supervised learning setting is a special case where in each episode we only make one prediction (such that, without loss of generality, we can take $T = 1$).

Our predictions are linear, that is, the prediction at time t is the inner product of ϕ_t and a learned weight vector θ, denoted $\phi_t^\top \theta$. The algorithms are indifferent to the origins of the features, which may be handcrafted or learned.[1] The weights have an initial value θ_0 that is presumably due to previous episodes. We analyze how the weights change in a single episode (and thus we do not include the episode number in our notation).

At the final time, when Z is observed, we can update all the predictions towards the target as in the classical least mean squares (LMS) algorithm defined by the updates:

$$\theta_{t+1} \doteq \theta_t + \alpha_t \phi_t \left(Z - \phi_t^\top \theta_t \right), \qquad t = 0, \ldots, T - 1, \tag{1}$$

where $\alpha_t > 0$ is a step-size parameter that may vary from time step to time step (e.g., as a function of the state at that time). We call this a *forward view*, because to update the prediction at time t we need to look forward in time to the outcome Z which is observed at the later time T.

To perform the updates (1) we have to wait until Z is known and then do the update for all previous time steps t. This requires storing and then computing updates for all the preceding feature vectors. The required computational resources scale with the span of the prediction (the maximum length of an episode), which is what we wish to avoid. We seek incremental computations whose per-time-step complexity is $O(n)$, where n is the number of parameters, and that result in the same weights as (1) at the end of the episode. That is, the incremental updates should compute the same θ_T as (1) if they are given the same input (the same θ_0, the same sequence $\{\phi_t\}_{t=0}^{T-1}$, and the same Z).

It may seem that the best we can hope for is to approximate the result computed by the LMS algorithm, because of the strict computational restriction. Such a trade off between computation and accuracy is not uncommon. We will however now derive an algorithm

---

[1]This includes, for instance, the case where ϕ_t is the last hidden layer of a neural network.

that finds the exact same final predictions with much more congenial computation, by carefully analyzing the total change to the weight vector due to the LMS algorithm.

The final step of the algorithm in (1) can be rewritten as

$$
\begin{aligned}
\boldsymbol{\theta}_T &= \boldsymbol{\theta}_{T-1} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}\left(Z - \boldsymbol{\phi}_{T-1}^\top \boldsymbol{\theta}_{T-1}\right) \\
&= \boldsymbol{\theta}_{T-1} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}Z - \alpha_{T-1}\boldsymbol{\phi}_{T-1}\boldsymbol{\phi}_{T-1}^\top \boldsymbol{\theta}_{T-1} \\
&= \left(\mathbf{I} - \alpha_{T-1}\boldsymbol{\phi}_{T-1}\boldsymbol{\phi}_{T-1}^\top\right)\boldsymbol{\theta}_{T-1} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}Z \\
&= \mathbf{F}_{T-1}\boldsymbol{\theta}_{T-1} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}Z\,.
\end{aligned}
$$

Here $\mathbf{F}_t \doteq \mathbf{I} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top$ is a *fading* matrix that will be important throughout this paper. Now, continuing,

$$
\begin{aligned}
\boldsymbol{\theta}_T &= \mathbf{F}_{T-1}\left(\mathbf{F}_{T-2}\boldsymbol{\theta}_{T-2} + \alpha_{T-2}\boldsymbol{\phi}_{T-2}Z\right) + \alpha_{T-1}\boldsymbol{\phi}_{T-1}Z && \text{(expanding } \boldsymbol{\theta}_{T-1}) \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}\boldsymbol{\theta}_{T-2} + \left(\mathbf{F}_{T-1}\alpha_{T-2}\boldsymbol{\phi}_{T-2} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}\right)Z && \text{(regrouping)} \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}\left(\mathbf{F}_{T-3}\boldsymbol{\theta}_{T-3} + \alpha_{T-3}\boldsymbol{\phi}_{T-3}Z\right) + \left(\mathbf{F}_{T-1}\alpha_{T-2}\boldsymbol{\phi}_{T-2} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}\right)Z && \\
& && \text{(recursing on } \boldsymbol{\theta}_{T-2}) \\
&= \mathbf{F}_{T-1}\mathbf{F}_{T-2}\mathbf{F}_{T-3}\boldsymbol{\theta}_{T-3} + \left(\mathbf{F}_{T-1}\mathbf{F}_{T-2}\alpha_{T-3}\boldsymbol{\phi}_{T-3} + \mathbf{F}_{T-1}\alpha_{T-2}\boldsymbol{\phi}_{T-2} + \alpha_{T-1}\boldsymbol{\phi}_{T-1}\right)Z && \\
& && \text{(regrouping)} \\
& \ \ \vdots && \text{(recursing further)} \\
&= \underbrace{\mathbf{F}_{T-1}\mathbf{F}_{T-2}\cdots\mathbf{F}_0\boldsymbol{\theta}_0}_{\doteq\, \boldsymbol{a}_{T-1}} \ + \ \underbrace{\left(\sum_{t=0}^{T-1}\mathbf{F}_{T-1}\mathbf{F}_{T-2}\cdots\mathbf{F}_{t+1}\alpha_t\boldsymbol{\phi}_t\right)Z}_{\doteq\, \boldsymbol{e}_{T-1}} && \\
&= \boldsymbol{a}_{T-1} + \boldsymbol{e}_{T-1}Z\,, && (2)
\end{aligned}
$$

where $\boldsymbol{a}_t$ and $\boldsymbol{e}_t$ are two auxiliary memory vectors.

Importantly, the auxiliary vectors can be updated without knowledge of Z, and with complexity independent of span and proportional to the number of features. The $\boldsymbol{a}_t$ vector stores the effect of the initial weights on the updated weights. It is initialized as $\boldsymbol{a}_0 = \boldsymbol{\theta}_0$ and can then be updated efficiently with

$$
\begin{aligned}
\boldsymbol{a}_t &\doteq \mathbf{F}_t\mathbf{F}_{t-1}\cdots\mathbf{F}_0\boldsymbol{\theta}_0 \\
&= \mathbf{F}_t\left(\mathbf{F}_{t-1}\cdots\mathbf{F}_1\boldsymbol{\theta}_1\right) \\
&= \mathbf{F}_t\boldsymbol{a}_{t-1} \\
&= \boldsymbol{a}_{t-1} - \alpha_t\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top \boldsymbol{a}_{t-1} \\
&= \boldsymbol{a}_{t-1} + \alpha_t\boldsymbol{\phi}_t(0 - \boldsymbol{\phi}_t^\top \boldsymbol{a}_{t-1})\,, \qquad t = 1,\ldots,T-1. && (3)
\end{aligned}
$$

The $\boldsymbol{e}_t$ vector is analogous to the conventional eligibility trace (see: Sutton 1988; Sutton and Barto 1998, and references therein) but has a special form as first proposed by van Seijen and Sutton (2014). It is initialized to $\boldsymbol{e}_{-1} = \mathbf{0}$ (or, equivalently, to $\boldsymbol{e}_0 = \alpha_0\boldsymbol{\phi}_0$) and then updated according to

$$
\boldsymbol{e}_t \doteq \sum_{k=0}^{t}\mathbf{F}_t\mathbf{F}_{t-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k
$$

$$= \sum_{k=0}^{t-1} \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

$$= \mathbf{F}_t \underbrace{\sum_{k=0}^{t-1} \mathbf{F}_{t-1} \mathbf{F}_{t-2} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k}_{= \, \boldsymbol{e}_{t-1}} \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

$$= \mathbf{F}_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t \tag{4}$$

$$= \boldsymbol{e}_{t-1} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t$$

$$= \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), \qquad t = 0, \ldots, T-1. \tag{5}$$

An eligibility trace of this special form is called a *dutch trace* (van Hasselt, Mahmood, and Sutton 2014). For comparison, the conventional accumulating trace that it replaces can be written as $\boldsymbol{e}_{-1} \doteq \mathbf{0}$ and $\boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t.$[2]

The emergence of the dutch trace here is surprising and intriguing because, in contrast to previous work (van Seijen and Sutton 2014; van Hasselt, Mahmood, and Sutton 2014), the dutch trace has arisen in a setting without temporal-difference (TD) learning. Eligibility traces are not specific to TD learning at all; they are more fundamental than that. The need for eligibility traces seems to arise whenever one tries to learn long-term predictions in an efficient manner, that is, with computational complexity that is independent of predictive span.

The auxiliary vectors $\boldsymbol{a}_t$ and $\boldsymbol{e}_t$ are updated on each time step $t < T$ and then, after observing Z at time T, are used to compute $\boldsymbol{\theta}_T = \boldsymbol{a}_{T-1} + \boldsymbol{e}_{T-1} Z$, as in (2). This way we achieve exactly the same final result as the forward view (1), but with an algorithm whose time and memory complexity per step is uniformly $O(n)$ and independent of span. The complete algorithm can be summarized as:

$$\boldsymbol{a}_0 \doteq \boldsymbol{\theta}_0 \,, \quad \text{then } \boldsymbol{a}_{t+1} \doteq \boldsymbol{a}_t + \alpha_t \boldsymbol{\phi}_t (0 - \boldsymbol{\phi}_t^\top \boldsymbol{a}_t), \qquad t = 1, \ldots, T-1 \,,$$
$$\boldsymbol{e}_{-1} \doteq \mathbf{0} \,, \quad \text{then } \boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), \quad t = 0, \ldots, T-1 \,, \tag{6}$$
$$\boldsymbol{\theta}_T \doteq \boldsymbol{a}_{T-1} + Z \boldsymbol{e}_{T-1} \,.$$

The vector $\boldsymbol{a}_{T-1}$ can be interpreted as storing the remaining effect of the initial weights $\boldsymbol{\theta}_0$ after all updates in (1) have concluded. The trace $\boldsymbol{e}_{T-1}$ can be interpreted as storing all we need to know about the feature vectors that were observed during the episode. Together, these vectors allow us to replace all the T updates of the forward view with one fully equivalent update at the end of the episode.

We call the span-independent algorithm (6) the *backward view* corresponding to the forward view defined in (1), because on each step all updates only use information that is available at that time step: we only look backwards in time. The advantage of this is that the updates can be computed immediately and we do not have to wait and store observations until later. Until recently, exactly equivalences between forward and backward views were only known to exist for algorithms that update their predictions in batch (Sutton and Barto 1998). Van Seijen & Sutton (2014) were the first to derive an online backward

---

[2]We incorporate the step size into both trace updates. This is a slight deviation from the way these traces are usually written to allow for time-changing step sizes and increased generality.
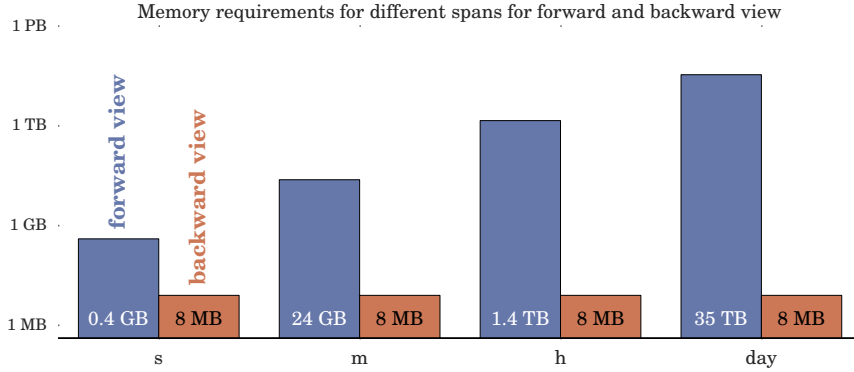
Figure 1: **Example memory requirements.** *The bars show memory requirements for the conventional algorithm (1) and the span-independent algorithm (6). The brown bars of the conventional algorithm (1) that stores all observations over the duration of a second, minute, hour, or day, when one million features of 4 bytes each are observed every 10 ms. The blue bars on the right show the memory required by the span-independent algorithm (6) in the same setting, which is 8 MB for any span.*

view that was exactly equivalent to its forward view in terms of the learned predictions. The derivation in this section shows that such equivalences exist more generally, including for the LMS update in (1).

When we consider the episode as a whole, there is no gain in total computation time for the span-independent algorithm (6) compared to the conventional algorithm (1): both algorithms use $O(nT)$ computation for the entire episode. However, in the span-independent algorithm the computation is spread out more evenly with a uniform per-step complexity of $O(n)$, whereas the conventional algorithm performs the bulk of computation at the end, when we finally observe Z.

Additionally, there is a gain in terms of required memory. Algorithm (1) needs to store all previously observed features, leading to memory requirements of order $O(nT)$, whereas algorithm (6) only needs to store $\boldsymbol{a}$ and $\boldsymbol{e}$ and therefore has constant span-independent memory requirements of order $O(n)$. In real-world problems, for instance in robotics, it is not uncommon to extract millions of features from the sensory inputs at each step (e.g., Montemerlo and Thrun 2003), where each step lasts only a fraction of a second. Consider a robot that generates one million features each 10 ms, where each feature is a real number represented with single precision using 4 bytes of memory. Figure 1 shows the resulting memory requirements for the conventional algorithm and the span-independent algorithm for different spans of the predictions. The required storage of the conventional algorithm ranges from 0.4 GB for predictions spanning one second to about 35 Terabytes for predictions spanning one day. While a few Gigabytes of on-board storage is feasible with today's resources, several Terabits will be a significant burden for an autonomous

7

mobile robot. Concretely, this means that with the conventional algorithm we would have to restrict either the number of features, the frequency at which we make predictions, or the maximum amount of wall time a prediction can span. The span-independent algorithm scales much better. In the example in Figure 1 it only needs to store two vectors with 10^6 components of 4 bytes each, resulting in memory requirements that are constant at 8 Megabytes.

4 Online updating and the emergence of TD errors

The algorithms in the previous section do not make any changes to the predictions during the episode; these are *offline* algorithms. Yet it would sometimes make sense to update the predictions during an episode, especially if the span is very long and we do not want to wait that long before we start learning. In this section we introduce an online forward view and derive a span-independent algorithm (the backward view) that on each time step computes the exact same predictions.

An online algorithm cannot update the predictions towards the final outcome Z during the episode, because Z is not yet available. Instead, if we only have observations up to a horizon $h < T$ we may want to move the predictions for all earlier times $t < h$ towards some informed guess of what the final outcome will be. Such a guess plays the role of a target for the updates, like Z in the forward view (1), but it is used prior to Z being available; it is an *interim target*. We use Z^h to denote the interim target at time h, which may be based on all the data available up that horizon. The interim target might be from a human expert or it might be, as in TD learning, the current prediction corresponding to the feature vector ϕ_h. Interim targets at times closer to T might produce more accurate predictions. In the example of the stock market, as we get closer to the end of the year we may be able to more accurately estimate the final stock price. For now we consider the general case and do not specify the source of the interim targets Z^h, for $h = 1, \ldots, T - 1$. Notationally it is convenient to define $Z^T \doteq Z$. Note that the time index on Z^h is a superscript rather than a subscript. Our convention is that the superscript position is reserved for the upper limit of the data considered available in an online update. The subscript position is used for the time step whose prediction is being modified.

To clarify the notation under online updating, we introduce the notation $\boldsymbol{\theta}_t^h$ for the weights at step t based on all the data up through time h. Using these double subscripts, what we previously called $\boldsymbol{\theta}_t$ would now be $\boldsymbol{\theta}_t^T$, because these weight vectors depend on Z which is considered to arrive at time T. The complete set of online updates is then

$$
\begin{aligned}
\boldsymbol{\theta}_0^h &\doteq \boldsymbol{\theta}_0, & h &= 0, \ldots, T\,; \\
\boldsymbol{\theta}_{t+1}^h &\doteq \boldsymbol{\theta}_t^h + \alpha_t \phi_t \left(Z^h - \phi_t^\top \boldsymbol{\theta}_t^h \right), & t &= 0, \ldots, h-1, \quad h = 1, \ldots, T\,; \qquad (7) \\
&= \mathbf{F}_t \boldsymbol{\theta}_t^h + \alpha_t \phi_t Z^h.
\end{aligned}
$$

This online algorithm defines a set of h updates for each interim horizon $h \in \{1, \ldots, T\}$. For each horizon, all predictions are updated towards the latest, and presumably best available, interim target. Although most updates do not involve the final outcome Z, this algorithm is still considered a forward view, because the prediction at some time t (e.g., $\phi_t^\top \boldsymbol{\theta}_t^h$) is updated using interim targets that arrive later (e.g., Z^h).

We can write out all the double-subscripted weight vectors in a triangle as

$$
\begin{aligned}
&\boldsymbol{\theta}_0^0 \\
&\boldsymbol{\theta}_0^1 \quad\;\; \boldsymbol{\theta}_1^1 \\
&\;\vdots \qquad \vdots \qquad \ddots \\
&\boldsymbol{\theta}_0^h \quad\;\; \boldsymbol{\theta}_1^h \quad \cdots \quad \boldsymbol{\theta}_h^h \\
&\boldsymbol{\theta}_0^{h+1} \; \boldsymbol{\theta}_1^{h+1} \;\; \cdots \;\; \boldsymbol{\theta}_h^{h+1} \;\; \boldsymbol{\theta}_{h+1}^{h+1} \\
&\;\vdots \qquad \vdots \qquad\quad\;\; \vdots \qquad\;\; \vdots \qquad \ddots \\
&\boldsymbol{\theta}_0^T \quad\;\; \boldsymbol{\theta}_1^T \quad \cdots \quad \boldsymbol{\theta}_h^T \quad\;\; \boldsymbol{\theta}_{h+1}^T \quad \cdots \quad \boldsymbol{\theta}_T^T \, .
\end{aligned} \tag{8}
$$

The computation proceeds from top to bottom, one row at a time from left to right. Each row starts with the same initial values $\boldsymbol{\theta}_0^h = \boldsymbol{\theta}_0$ and then computes the sequence $\boldsymbol{\theta}_1^h$, $\boldsymbol{\theta}_2^h$ and so on If we really computed all the different weight vectors in the triangle then the algorithm would be inefficient. With a computational complexity of $O(nh)$ on each time $h \in \{1, \ldots, T\}$, the computation would not be constant per time step and would not be independent of the span of the prediction; the last row alone has complexity $O(nT)$ and it can only be computed after observing Z at time T. However, instead of computing the whole triangle, perhaps there is a way to incrementally compute just the diagonal, to somehow obtain $\boldsymbol{\theta}_{h+1}^{h+1}$ from $\boldsymbol{\theta}_h^h$ efficiently on each step. If this can be done, then the entire computation will be of uniform $O(n)$ complexity per time step, independent of span.

To find an efficient update along the diagonal of the triangle, notice first that the forward view (7) already provides a way to efficiently move right one step in the triangle. In other words, we can get to $\boldsymbol{\theta}_{h+1}^{h+1}$ from $\boldsymbol{\theta}_h^{h+1}$ for any h with constant $O(n)$ computation. If we can find an efficient way to step down in the triangle, that is to get to $\boldsymbol{\theta}_h^{h+1}$ from $\boldsymbol{\theta}_h^h$ for any h with constant $O(n)$ computation, then we can combine these two steps into a single $O(n)$ update. To see if this is possible, we first write down explicitly how each weight vector in the triangle depends on the initial weights and the interim targets. Similar to our derivation of the final weights in (2) in the previous section, we can unroll the forward-view updates repeatedly starting from $\boldsymbol{\theta}_t^h$ and obtaining

$$
\begin{aligned}
\boldsymbol{\theta}_t^h &= \mathbf{F}_{t-1}\boldsymbol{\theta}_{t-1}^h + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z^h && \text{(applying (7) to } \boldsymbol{\theta}_t^h) \\
&= \mathbf{F}_{t-1}(\mathbf{F}_{t-2}\boldsymbol{\theta}_{t-2}^h + \alpha_{t-2}\boldsymbol{\phi}_{t-2}Z^h) + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z^h && \text{(applying (7) to } \boldsymbol{\theta}_{t-1}^h) \\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}\boldsymbol{\theta}_{t-2}^h + (\mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2} + \alpha_{t-1}\boldsymbol{\phi}_{t-1})\, Z^h && \text{(regrouping)} \\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}(\mathbf{F}_{t-3}\boldsymbol{\theta}_{t-3}^h + \alpha_{t-3}\boldsymbol{\phi}_{t-3}Z^h) + (\mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2} + \alpha_{t-1}\boldsymbol{\phi}_{t-1})\, Z^h && \\
&&& \text{(applying (7) to } \boldsymbol{\theta}_{t-2}^h) \\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}\mathbf{F}_{t-3}\boldsymbol{\theta}_{t-3}^h + (\mathbf{F}_{t-1}\mathbf{F}_{t-2}\alpha_{t-3}\boldsymbol{\phi}_{t-3} + \mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2} + \alpha_{t-1}\boldsymbol{\phi}_{t-1})\, Z^h && \\
&\;\;\vdots && \text{(continuing until we reach } \boldsymbol{\theta}_0^h) \\
&= \underbrace{\mathbf{F}_{t-1}\cdots\mathbf{F}_0\boldsymbol{\theta}_0}_{\boldsymbol{a}_{t-1}} \;+\; \underbrace{\left(\sum_{j=0}^{t-1}\mathbf{F}_{t-1}\cdots\mathbf{F}_{j+1}\alpha_j\boldsymbol{\phi}_j\right)}_{\boldsymbol{e}_{t-1}} Z^h && \\
&= \boldsymbol{a}_{t-1} + \boldsymbol{e}_{t-1}Z^h \, . && \tag{9}
\end{aligned}
$$

Notice that $\boldsymbol{a}_{t-1}$ and $\boldsymbol{e}_{t-1}$ depend only on time t and not on the data horizon h. We can use this result to find the difference between $\boldsymbol{\theta}_h^{h+1}$ and $\boldsymbol{\theta}_h^h$ as

$$
\begin{aligned}
\boldsymbol{\theta}_h^{h+1} - \boldsymbol{\theta}_h^h &= (\boldsymbol{a}_{h-1} + \boldsymbol{e}_{h-1}Z^{h+1}) - (\boldsymbol{a}_{h-1} + \boldsymbol{e}_{h-1}Z^h) \\
&= \boldsymbol{e}_{h-1}(Z^{h+1} - Z^h) .
\end{aligned}
\tag{10}
$$

Here we see the emergence of a temporal difference error $Z^{h+1} - Z^h$. We know from the previous section that $\boldsymbol{e}_{h-1}$ can be computed incrementally with (5) and therefore does not have to be recomputed from scratch for each new observation. Therefore, we now have an efficient way to compute $\boldsymbol{\theta}_h^{h+1}$ from $\boldsymbol{\theta}_h^h$ for any h with constant $O(n)$ computation per step that is independent of span. Now that we have $\boldsymbol{\theta}_h^{h+1}$ we can efficiently compute $\boldsymbol{\theta}_{h+1}^{h+1}$ using (7), and we can merge these two steps to compute $\boldsymbol{\theta}_{h+1}^{h+1}$ directly from $\boldsymbol{\theta}_h^h$. The complete update can then be written as

$$
\begin{aligned}
\boldsymbol{\theta}_{h+1}^{h+1} &= \mathbf{F}_h \boldsymbol{\theta}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h Z^{h+1} && \text{(using (7))} \\
&= \mathbf{F}_h \left(\boldsymbol{\theta}_h^h + \boldsymbol{e}_{h-1}(Z^{h+1} - Z^h) \right) + \alpha_h \boldsymbol{\phi}_h Z^{h+1} && \text{(using (10))} \\
&= \mathbf{F}_h \boldsymbol{\theta}_h^h + \mathbf{F}_h \boldsymbol{e}_{h-1}(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^{h+1} && \\
&= \mathbf{F}_h \boldsymbol{\theta}_h^h + (\boldsymbol{e}_h - \alpha_h \boldsymbol{\phi}_h)(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^{h+1} && \text{(using (4))} \\
&= \mathbf{F}_h \boldsymbol{\theta}_h^h + \boldsymbol{e}_h(Z^{h+1} - Z^h) - \alpha_h \boldsymbol{\phi}_h(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^{h+1} && \\
&= \mathbf{F}_h \boldsymbol{\theta}_h^h + \boldsymbol{e}_h(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^h && \\
&= (\mathbf{I} - \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top) \boldsymbol{\theta}_h^h + \boldsymbol{e}_h(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^h && \text{(by definition of } \mathbf{F}_h\text{)} \\
&= \boldsymbol{\theta}_h^h + \boldsymbol{e}_h(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h \left(Z^h - \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right) .
\end{aligned}
$$

This update holds for all $h \geq 1$. The update for $\boldsymbol{\theta}_1^1$ is given directly by (7) as

$$
\boldsymbol{\theta}_1^1 = \boldsymbol{\theta}_0^1 + \alpha_0 \boldsymbol{\phi}_0 (Z^1 - \boldsymbol{\phi}_0^\top \boldsymbol{\theta}_0^1) .
$$

For any Z^0 we can rewrite this as

$$
\begin{aligned}
\boldsymbol{\theta}_1^1 &= \boldsymbol{\theta}_0^0 + \alpha_0 \boldsymbol{\phi}_0 (Z^1 - \boldsymbol{\phi}_0^\top \boldsymbol{\theta}_0^0) && \text{(using } \boldsymbol{\theta}_0^1 \doteq \boldsymbol{\theta}_0 \doteq \boldsymbol{\theta}_0^0\text{)} \\
&= \boldsymbol{\theta}_0^0 + \alpha_0 \boldsymbol{\phi}_0 (Z^1 - Z^0 + Z^0 - \boldsymbol{\phi}_0^\top \boldsymbol{\theta}_0^0) && \\
&= \boldsymbol{\theta}_0^0 + \alpha_0 \boldsymbol{\phi}_0 (Z^1 - Z^0) + \alpha_0 \boldsymbol{\phi}_0 (Z^0 - \boldsymbol{\phi}_0^\top \boldsymbol{\theta}_0^0) && \\
&= \boldsymbol{\theta}_0^0 + \boldsymbol{e}_0(Z^1 - Z^0) + \alpha_0 \boldsymbol{\phi}_0 (Z^0 - \boldsymbol{\phi}_0^\top \boldsymbol{\theta}_0^0) , && \text{(using } \boldsymbol{e}_0 \doteq \alpha_0 \boldsymbol{\phi}_0\text{)}
\end{aligned}
$$

which means that then the update derived above for $h \geq 1$ in fact also holds for $h = 0$. For concreteness, we will define $Z^0 \doteq 0$, even though this value does not actually affect any of the weights.

Now that we have an update that can compute $\boldsymbol{\theta}_{t+1}^{t+1}$ from $\boldsymbol{\theta}_t^t$ for any t, we can drop the redundant superscript. The resulting algorithm is

$$
\begin{aligned}
\boldsymbol{e}_{-1} &\doteq \mathbf{0}, \text{ then } \boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), && t = 0, \dots, T-1, \\
\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \boldsymbol{e}_t \left(Z^{t+1} - Z^t \right) + \alpha_t \boldsymbol{\phi}_t (Z^t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t), && t = 0, \dots, T-1.
\end{aligned}
\tag{11}
$$

By construction this backward view is equivalent to the less efficient forward view (7) in the sense that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t^t$ for all t. In contrast to the offline backward view (6) that we derived

in the previous section, we no longer need to compute and store the auxiliary vector $\boldsymbol{a}_t$. All relevant information that was contained therein is now stored directly in the online weights $\boldsymbol{\theta}_t$.

Although the online backward view yields different predictions during the episode, the final weights $\boldsymbol{\theta}_T$ are exactly equal to those computed by the conventional LMS algorithm (1) that constituted our first, offline, forward view. In terms of the triangle in (8), the online forward view (7) computes the whole triangle, the online backward view (11) efficiently computes only the diagonal, the offline forward view (1) computes only the last row, and the offline backward view (6) from the previous section computes only the final weights. All three algorithms ultimately result in the same final weights.

5 Unifying online and offline learning and the emergence of averaging

The online algorithms from the previous section do not quite subsume the offline algorithms from Section 3. Although they all reach the same weights by the end of the episode, during the episode their weights are different. The offline algorithm does not change the weights during the episode, and the online algorithm must change them.

One might think that the online algorithm is always better because it can immediately use any incoming relevant information, but it is not so. Suppose the interim targets are always wildly wrong (say due to a poor human 'expert'). They would cause the weights of the online algorithm to also be wildly wrong for all steps except the last one at the end of the episode. In this case the weights of the online algorithm would be worse than those of the offline algorithm almost all of the time.

Because interim targets can sometimes be misleading, we might want to reduce their effect on some steps, based on how much we trust these targets. In this section $\tilde{\boldsymbol{\theta}}_t$ denotes the online weights, which are computed by the span-independent backward view (11). The unified weights $\boldsymbol{\theta}_t$ take into account the degree of trust for each interim target and are used to make our predictions. If we trust Z^t fully, we want to obtain the same weights as in the online algorithm, so that $\boldsymbol{\theta}_t = \tilde{\boldsymbol{\theta}}_t$. If we do not trust Z^t at all, we want the predictions to remain unchanged, so that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$. For intermediate degrees of trust, $\beta_t \in (0, 1)$, the algorithm should smoothly move from one extreme to the other, so the final result of the update should be something like

$$\boldsymbol{\theta}_{t+1} = (1 - \beta_{t+1})\boldsymbol{\theta}_t + \beta_{t+1}\tilde{\boldsymbol{\theta}}_{t+1} \,. \tag{12}$$

The above reasoning may sound plausible, but is it sound? In this section we construct a forward view for partially trusted interim targets, and then derive an equivalent span-independent backward view. It turns out the resulting update indeed changes the weights precisely as in (12).

In the forward view, the online weights that always trust the latest interim target fully will be denoted $\tilde{\boldsymbol{\theta}}_t^h$ to differentiate them from the trusted interim weights $\boldsymbol{\theta}_t^h$. If we have data up to horizon h but we trust the latest interim target Z^h only with degree $\beta_h \in [0, 1]$, then the predictions prior to h should update towards Z^h only to this degree and for the rest, with degree $(1 - \beta_h)$, fall back on earlier interim targets. Similarly we update towards

Z^{h-1} only as far as this interim target was trusted, so with total degree $(1 - \beta_h)\beta_{h-1}$, and then further fall back to Z^{h-2} with total degree $(1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2}$, and so on. If we do not trust any of the interim targets observed between the time t of making the prediction and the current horizon h, the predictions should remain wherever they were at time t. This can be achieved by updating the prediction at time t towards the then-current prediction $\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^t$ with the remaining degree $(1 - \beta_h)(1 - \beta_{h-1})\cdots(1 - \beta_{t+1})$. Note that the different multipliers sum to one:

$$\beta_h \; + \; (1 - \beta_h)\beta_{h-1} \; + \; (1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2} \; + \; \ldots \; + \; (1 - \beta_h)\cdots(1 - \beta_{t+1}) = 1 \,.$$

The total forward view for a prediction at time t with a horizon of h is therefore given by

$$
\begin{aligned}
\boldsymbol{\theta}_{t+1}^h = \; & \boldsymbol{\theta}_t^h + \beta_h \alpha_t \boldsymbol{\phi}_t (Z^h - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \\
& + (1 - \beta_h)\beta_{h-1}\alpha_t \boldsymbol{\phi}_t (Z^{h-1} - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \\
& + (1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2}\alpha_t \boldsymbol{\phi}_t (Z^{h-2} - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \\
& \quad \vdots \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+2})\beta_{t+1}\alpha_t \boldsymbol{\phi}_t (Z^{t+1} - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+2})(1 - \beta_{t+1})\alpha_t \boldsymbol{\phi}_t (\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \,.
\end{aligned}
$$

(now grouping terms $\alpha_t \boldsymbol{\phi}_t(\cdot - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h)$ with total weight equal to one)

$$
\begin{aligned}
= \; & \boldsymbol{\theta}_t^h + \alpha_t \boldsymbol{\phi}_t \Big(\beta_h Z^h + \\
& \qquad (1 - \beta_h)\beta_{h-1}Z^{h-1} + \\
& \qquad (1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2}Z^{h-2} + \\
& \qquad \vdots \\
& \qquad (1 - \beta_h)\cdots(1 - \beta_{t+2})\beta_{t+1}Z^{t+1} + \\
& \qquad (1 - \beta_h)\cdots(1 - \beta_{t+2})(1 - \beta_{t+1})\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h \Big) \\
= \; & \boldsymbol{\theta}_t^h + \alpha_t \boldsymbol{\phi}_t (Z_t^h - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h) \\
= \; & \mathbf{F}_t \boldsymbol{\theta}_t^h + \alpha_t \boldsymbol{\phi}_t Z_t^h \,, \qquad t = 0, \ldots, h-1; \quad h = 1, \ldots, T, \qquad (13)
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad Z_t^h \doteq \; & \beta_h Z^h \\
& + (1 - \beta_h)\beta_{h-1}Z^{h-1} \\
& + (1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2}Z^{h-2} \\
& + \ldots \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+2})\beta_{t+1}Z^{t+1} \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+2})(1 - \beta_{t+1})\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t \\
= \; & \beta_h Z^h + (1 - \beta_h)Z_t^{h-1} \,, \qquad t = 0, \ldots, h-1; \quad h = 1, \ldots, T \,, \text{ and} \qquad (14)
\end{aligned}
$$

$$Z_t^t \doteq \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^t \qquad\qquad\qquad t = 0, \ldots, T-1. \qquad (15)$$

12

To derive a span-independent variant of algorithm (13), we first identify how a general weight vector $\boldsymbol{\theta}_t^h$ depends on the combined interim targets and the observed feature vectors by applying the recursive definition in (13) repeatedly, yielding

$$
\begin{aligned}
\boldsymbol{\theta}_t^h &= \mathbf{F}_{t-1}\boldsymbol{\theta}_{t-1}^h + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z_{t-1}^h && \text{(applying (13) to } \boldsymbol{\theta}_t^h\text{)}\\
&= \mathbf{F}_{t-1}\left(\mathbf{F}_{t-2}\boldsymbol{\theta}_{t-2}^h + \alpha_{t-2}\boldsymbol{\phi}_{t-2}Z_{t-2}^h\right) + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z_{t-1}^h && \text{(applying (13) to } \boldsymbol{\theta}_{t-1}^h\text{)}\\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}\boldsymbol{\theta}_{t-2}^h + \mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2}Z_{t-2}^h + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z_{t-1}^h\\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}(\mathbf{F}_{t-3}\boldsymbol{\theta}_{t-3}^h + \alpha_{t-3}\boldsymbol{\phi}_{t-3}Z_{t-3}^h) + \mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2}Z_{t-2}^h + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z_{t-1}^h\\
&&& \text{(applying (13) to } \boldsymbol{\theta}_{t-2}^h\text{)}\\
&= \mathbf{F}_{t-1}\mathbf{F}_{t-2}\mathbf{F}_{t-3}\boldsymbol{\theta}_{t-3}^h + \mathbf{F}_{t-1}\mathbf{F}_{t-2}\alpha_{t-3}\boldsymbol{\phi}_{t-3}Z_{t-3}^h + \mathbf{F}_{t-1}\alpha_{t-2}\boldsymbol{\phi}_{t-2}Z_{t-2}^h + \alpha_{t-1}\boldsymbol{\phi}_{t-1}Z_{t-1}^h\\
&\vdots\\
&= \underbrace{\mathbf{F}_{t-1}\cdots\mathbf{F}_0\boldsymbol{\theta}_0^t}_{\doteq\ \boldsymbol{a}_{t-1}} + \sum_{k=0}^{t-1}\mathbf{F}_{t-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k Z_k^h\\
&= \boldsymbol{a}_{t-1} + \sum_{k=0}^{t-1}\mathbf{F}_{t-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k Z_k^h\,. && (16)
\end{aligned}
$$

The last step uses the fact that the initial trusted weights $\boldsymbol{\theta}_0^t$ are equal to the initial online weights, such that $\boldsymbol{\theta}_0^t = \tilde{\boldsymbol{\theta}}_0^t = \boldsymbol{\theta}_0$ for any t, which means $\boldsymbol{a}_t$ is the same as before. Notice that we have not yet used the definition of Z_t^h in any way; the derivation so far holds for any combined target.

We now first examine if we can efficiently go down in the triangle, that is, to get to $\boldsymbol{\theta}_h^{h+1}$ from $\boldsymbol{\theta}_h^h$:

$$
\begin{aligned}
\boldsymbol{\theta}_h^{h+1} - \boldsymbol{\theta}_h^h &= \left(\boldsymbol{a}_{h-1} + \sum_{k=0}^{h-1}\mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k Z_k^{h+1}\right) && (\boldsymbol{\theta}_h^{h+1}\text{ from (16)})\\
&\quad -\left(\boldsymbol{a}_{h-1} + \sum_{k=0}^{h-1}\mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k Z_k^h\right) && (\boldsymbol{\theta}_h^h\text{ from (16)})\\
&= \sum_{k=0}^{h-1}\mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k\left(Z_k^{h+1} - Z_k^h\right) && \text{(merge sums, cancel }\boldsymbol{a}_{h-1}\text{)}\\
&= \sum_{k=0}^{h-1}\mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k\left(\beta_{h+1}Z^{h+1} + (1-\beta_{h+1})Z_k^h - Z_k^h\right)\\
&&& \text{(using (14) on }Z_k^{h+1}\text{)}\\
&= \sum_{k=0}^{h-1}\mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k\beta_{h+1}(Z^{h+1} - Z_k^h)
\end{aligned}
$$

$$= \beta_{h+1} \underbrace{\left(\sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \right) Z^{h+1}}_{= \, \boldsymbol{e}_{h-1}} \quad - \quad \beta_{h+1} \underbrace{\sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k Z_k^h}_{= \, \boldsymbol{\theta}_h^h - \boldsymbol{a}_{h-1}, \text{ from (16)}}$$

$$= \beta_{h+1} \boldsymbol{e}_{h-1} Z^{h+1} \quad - \quad \beta_{h+1}(\boldsymbol{\theta}_h^h - \boldsymbol{a}_{h-1})$$

$$= \beta_{h+1} \underbrace{(\boldsymbol{a}_{h-1} + \boldsymbol{e}_{h-1} Z^{h+1})}_{= \, \tilde{\boldsymbol{\theta}}_h^{h+1}, \text{ from (9)}} \quad - \quad \beta_{h+1} \boldsymbol{\theta}_h^h$$

$$= \beta_{h+1}(\tilde{\boldsymbol{\theta}}_h^{h+1} - \boldsymbol{\theta}_h^h) \tag{17}$$

Thus $\boldsymbol{\theta}_h^{h+1}$ can be written as a simple combination of the previous trusted weights $\boldsymbol{\theta}_h^h$ and the interim online weights $\tilde{\boldsymbol{\theta}}_h^{h+1}$ with

$$\boldsymbol{\theta}_h^{h+1} = (1 - \beta_{h+1})\boldsymbol{\theta}_h^h + \beta_{h+1}\tilde{\boldsymbol{\theta}}_h^{h+1} \, . \tag{18}$$

We can plug this value into the definition of $\boldsymbol{\theta}_{h+1}^{h+1}$ to find

$$\boldsymbol{\theta}_{h+1}^{h+1} = \mathbf{F}_h \boldsymbol{\theta}_h^{h+1} + \alpha_h Z_h^{h+1} \boldsymbol{\phi}_h \qquad\qquad\qquad \text{(using (13))}$$

$$= \mathbf{F}_h\left((1 - \beta_{h+1})\boldsymbol{\theta}_h^h + \beta_{h+1}\tilde{\boldsymbol{\theta}}_h^{h+1} \right) + \alpha_h Z_h^{h+1} \boldsymbol{\phi}_h \qquad \text{(using (18))}$$

$$= \mathbf{F}_h\left((1 - \beta_{h+1})\boldsymbol{\theta}_h^h + \beta_{h+1}\tilde{\boldsymbol{\theta}}_h^{h+1} \right) + \alpha_h \left(\beta_{h+1} Z^{h+1} + (1 - \beta_{h+1}) Z^h \right) \boldsymbol{\phi}_h \, .$$
$$\text{(using (14))}$$

$$= \mathbf{F}_h\left((1 - \beta_{h+1})\boldsymbol{\theta}_h^h + \beta_{h+1}\tilde{\boldsymbol{\theta}}_h^{h+1} \right) + \alpha_h \left(\beta_{h+1} Z^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right) \boldsymbol{\phi}_h \, .$$
$$\text{(using (15))}$$

Now we group the terms depending on whether they are trusted (multiplied with β_{h+1}) or untrusted (multiplied with $(1 - \beta_{h+1})$) to simply further to

$$\boldsymbol{\theta}_{h+1}^{h+1} = (1 - \beta_{h+1}) \underbrace{\left(\mathbf{F}_h \boldsymbol{\theta}_h^h + \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right)}_{= \boldsymbol{\theta}_h^h, \text{ using } \mathbf{F}_h = \mathbf{I} - \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top} \quad + \quad \beta_{h+1} \underbrace{\left(\mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^{h+1} + \alpha_h Z^{h+1} \boldsymbol{\phi}_h \right)}_{= \tilde{\boldsymbol{\theta}}_{h+1}^{h+1}, \text{ using (7)}}$$

$$= (1 - \beta_{h+1})\boldsymbol{\theta}_h^h + \beta_{h+1}\tilde{\boldsymbol{\theta}}_{h+1}^{h+1} \, .$$

All superscripts now match their corresponding subscripts and so we can write down an algorithm that is equivalent to the forward view in the sense that $\boldsymbol{\theta}_t^t = \boldsymbol{\theta}_t$ for all t, with

$$\begin{aligned}
\boldsymbol{e}_{-1} &\doteq 0, \text{ then } \boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), & t &= 0, \ldots, T-1, \\
\tilde{\boldsymbol{\theta}}_{t+1} &\doteq \tilde{\boldsymbol{\theta}}_t + \boldsymbol{e}_t \left(Z^{t+1} - Z^t \right) + \alpha_t \boldsymbol{\phi}_t (Z^t - \boldsymbol{\phi}_t^\top \tilde{\boldsymbol{\theta}}_t), & t &= 0, \ldots, T-1, \\
\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \beta_{t+1}(\tilde{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\theta}_t), & t &= 0, \ldots, T-1,
\end{aligned} \tag{19}$$

where $\beta_t \in [0, 1]$ is the degree of trust we place in Z^t. The first two lines compute the online weights, and are equal to the online backward view (11) from the previous section.

14

The last line effectively computes a weighted running average the online weights, according to the sequence $\{\beta_t\}_{t=1}^T$.

Algorithm (19) subsumes the previous algorithms. If $\beta_t = 1$, $\forall t$, then the predictions are equal those of the online algorithm on each step. If $\beta_t = 0$ for $t \in \{1, \ldots, T-1\}$ and $\beta_T = 1$, then the predictions are equal those of the offline algorithm. As long as we trust the final outcome, such that $\beta_T = 1$, then all these algorithms result in exactly the same weights at the end of the episode, and algorithm (19) allows us to be flexible about how much we change the predictions during the episode, without requiring us to commit to either fully online or fully offline updates for the whole episode.

6   Bootstrapping

So far we have always fully trusted the actual final outcome. All interim targets have been deemed irrelevant by the end of the episode, leaving no effect on the computed final weights. There are cases in which we do not want to discard all interim targets. For instance, consider a stock that crashes down just before the end of the year. Certainly our updated predictions should include the possibility of such a crash, but we may not want to predict it will always crash just before the year's end. Similarly, suppose it rains on a certain date for which we want to predict the weather. It then seems wasteful to ignore the sunny weather on the days leading up to that date. These are examples of cases in which the interim targets are almost as informative as the final outcome. In some cases, an interim target may even be *more* informative. For instance, it may be due to a highly-trusted expert that takes into account all possible outcomes from that point in time. Surely, this expert should not be ignored completely in favor of one random final outcome.

The general idea of updating predictions using other predictions, such as the interim targets, is called *bootstrapping* (see, e.g., Sutton and Barto 1998). One way to obtain persistence of interim targets in our final predictions, and to achieve bootstrapping, is to drop the requirement in the previous section that we trust the final outcome fully, and allow $\beta_T < 1$. The eventual target for our updates is then a weighted average of the interim targets and the final outcome. For instance, if $\beta_t = 1/2$ for all $t \in \{1, \ldots, T\}$ the final updates will place a weight of $\beta_T = 1/2$ on the final outcome, a weight of $(1 - \beta_T)\beta_{T-1} = 1/4$ on the interim target immediate before then, and so on.

The notion of trust from the previous section applies a single degree of trust for an interim target uniformly to all prior predictions, but this is not always desirable. Using this definition of trust, if we fully trust an interim target then it replaces all earlier interim targets. However, even if an interim target if fully trusted for the most recent prediction, it may be inherently less trustworthy for earlier predictions. To illustrate this, consider flipping a coin three times and predicting the total number of heads. The possible final outcomes are 0, 1, 2, and 3 heads. If a trusted expert tells us before the first flip that the coin is fair, that is equivalent to observing a trustworthy interim target of $Z^1 = 1.5$. Suppose then the first two flips both result in heads, such that the only remaining possible final outcomes are 2 and 3. If the coin is indeed fair, an interim target of 2.5 would now be trustworthy target for the prediction made after observing two heads. However, we would probably not want to replace the earlier interim target of 1.5 for the first prediction.

15

Unfortunately, this is exactly what happens with the algorithm from the previous section.

This suggests a different notion of trust, based on the degree of trust we place in an interim target as a stand-in for the *expected* final outcome rather than the *actual* (random) outcome. If an interim target precisely matches the expected outcome at that point in time, then later targets can then only be noisier or more specific but not more informative, and it should never be replaced by later targets.

Suppose, concretely, that we fully trust Z^t under this new notion of trust. Then, the update for the prediction made at time $t-1$ should disregard any targets that arrive later, including even the final outcome. Conversely, if we do not trust Z^t at all, then it should leave no trace in the final updates. More generally we can update towards Z^t with an intermediate degree of trust $\eta_t \in [0, 1]$. If the next interim target Z^{t+1} is trusted with degree η_{t+1}, we then update our prediction at time t towards it with a total weight of $(1 - \eta_t)\eta_{t+1}$. The update towards Z^{t+2} will get a total weight of $(1 - \eta_t)(1 - \eta_{t+1})\eta_{t+2}$, and so on until we reach either the final outcome or the current data horizon. The latest interim target (and the final outcome) is always trusted fully until we move to the next time horizon. Therefore, at horizon h we always place any remaining weight on Z^h and update towards it with total weight $(1 - \eta_t)(1 - \eta_{t+1}) \cdots (1 - \eta_{h-1})$.

The corresponding total update to the prediction at time t with a current horizon h is then

$$
\begin{aligned}
\boldsymbol{\theta}_{t+1}^h = \boldsymbol{\theta}_t^h &+ \eta_{t+1}\alpha_t\boldsymbol{\phi}_t(Z^{t+1} - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h) \\
&+ (1 - \eta_{t+1})\eta_{t+2}\alpha_t\boldsymbol{\phi}_t(Z^{t+2} - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h) \\
&+ (1 - \eta_{t+1})(1 - \eta_{t+2})\eta_{t+3}\alpha_t\boldsymbol{\phi}_t(Z^{t+3} - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h) \\
&+ \ldots \\
&+ (1 - \eta_{t+1})\cdots(1 - \eta_{h-2})\eta_{h-1}\alpha_t\boldsymbol{\phi}_t(Z^{h-1} - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h) \\
&+ (1 - \eta_{t+1})\cdots(1 - \eta_{h-1})\alpha_t\boldsymbol{\phi}_t(Z^h - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h) \\
\doteq \boldsymbol{\theta}_t^h &+ \alpha_t\boldsymbol{\phi}_t\left(Z_t^h - \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_t^h\right).
\end{aligned}
\tag{20}
$$

where we have grouped the updates into a single update towards a combined target Z_t^h, just as in the previous section. This update is perhaps more familiar when we change the notation slightly. For all t, we define $\lambda_t \doteq 1 - \eta_t$ such that λ_t essentially specifies to what degree we *dis*trust Z^t. The combined target is then defined as

$$
\begin{aligned}
Z_t^h \doteq\ &(1 - \lambda_{t+1})Z^{t+1} \\
&+ \lambda_{t+1}(1 - \lambda_{t+2})Z^{t+2} \\
&\cdots \\
&+ \lambda_{t+1}\cdots\lambda_{h-2}(1 - \lambda_{h-1})Z^{h-1} \\
&+ \lambda_{t+1}\cdots\lambda_{h-1}Z^h.
\end{aligned}
\tag{21}
$$

This target Z_t^h is known as a λ-return (Sutton and Barto 1998). The version that truncates at the current horizon h was first proposed by van Seijen and Sutton (2014). The total set of updates is

$$
\boldsymbol{\theta}_0^t \doteq \boldsymbol{\theta}_0\,, \qquad\qquad\qquad t = 0, \ldots, T\,;
$$

16

$$\boldsymbol{\theta}_{t+1}^{h} \doteq \boldsymbol{\theta}_{t}^{h} + \alpha_{t}\boldsymbol{\phi}_{t}(Z_{t}^{h} - \boldsymbol{\phi}_{t}^{\top}\boldsymbol{\theta}_{t}^{h}), \tag{22}$$
$$\doteq \mathbf{F}_{t}\boldsymbol{\theta}_{t}^{h} + \alpha_{t}\boldsymbol{\phi}_{t}Z_{t}^{h}, \qquad\qquad t = 0,\ldots,h-1; \quad h = 1,\ldots,T.$$

If we ultimately distrust all interim targets, then $\lambda_t = 1$ for all t and $Z_t^h = Z^h$. The algorithm then reduces to the online algorithm from Section 4. Otherwise, at least some interim targets persist and contribute to the final weights. At the other extreme, if we trust all interim targets, then $\lambda_t = 0$ for all t and $Z_t^h = Z^{t+1}$. Then, the updates reduce to single-step updates that only use the immediate next interim target. In that case each update depends only on the immediate next time step and we can drop the superscript h and the forward view (22) reduces to an efficient span-independent algorithm

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_{t} + \alpha_{t}\boldsymbol{\phi}_{t}(Z^{t+1} - \boldsymbol{\phi}_{t}^{\top}\boldsymbol{\theta}_{t}), \quad t = 0,\ldots,T.$$

Apart from this special case, the forward view (22) is computationally inefficient and we desire an efficient span-independent algorithm to get from $\boldsymbol{\theta}_h^h$ to $\boldsymbol{\theta}_{h+1}^{h+1}$. In the previous section we derived how a weight vector depends on any sequence of combined targets Z_t^h, independent on the definition of those targets. We repeat the result of that derivation, as first given in (16), here for clarity:

$$\boldsymbol{\theta}_{t}^{h} = \boldsymbol{a}_{t-1} + \sum_{k=0}^{t-1} \mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_{k}\boldsymbol{\phi}_{k}Z_{k}^{h} \qquad t = 0,\ldots,h-1; \quad h = 1,\ldots,T.$$

Because this equation holds regardless of the definition of Z_t^h, we can apply it to the current algorithm. In particular we use it to try to find an efficient algorithm to go from $\boldsymbol{\theta}_h^h$ to $\boldsymbol{\theta}_h^{h+1}$. If this is possible, we can then use the update (22) to go from $\boldsymbol{\theta}_h^{h+1}$ to $\boldsymbol{\theta}_{h+1}^{h+1}$. We start by writing out the difference as

$$\boldsymbol{\theta}_{h}^{h+1} - \boldsymbol{\theta}_{h}^{h} = \sum_{k=0}^{h-1} \mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_{k}\boldsymbol{\phi}_{k}Z_{k}^{h+1} - \sum_{k=0}^{h-1} \mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_{t}\boldsymbol{\phi}_{k}Z_{k}^{h}$$

$$(\boldsymbol{a}_{h-1} \text{ cancels})$$

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_{k}\boldsymbol{\phi}_{k}\left(Z_{k}^{h+1} - Z_{k}^{h}\right). \tag{23}$$

The combined targets Z_k^{h+1} and Z_k^h share many terms: going back to (21) we can see that all interim targets up to Z^{h-1} will have the exact same multipliers. These terms cancel, and the remaining difference is given by

$$Z_{k}^{h+1} - Z_{k}^{h} = \underbrace{\lambda_{k+1}\cdots\lambda_{h-1}(1-\lambda_{h})Z^{h} + \lambda_{k+1}\cdots\lambda_{h}Z^{h+1}}_{\text{due to } Z_{k}^{h+1}} - \underbrace{\lambda_{k+1}\cdots\lambda_{h-1}Z^{h}}_{\text{due to } Z_{k}^{h}}$$

$$= \lambda_{k+1}\cdots\lambda_{h}\left(Z^{h+1} - Z^{h}\right). \tag{24}$$

We can then continue from (23) with

$$\boldsymbol{\theta}_{h}^{h+1} - \boldsymbol{\theta}_{h}^{h} = \sum_{k=0}^{h-1} \mathbf{F}_{h-1}\cdots\mathbf{F}_{k+1}\alpha_{k}\boldsymbol{\phi}_{k}\left(Z_{k}^{h+1} - Z_{k}^{h}\right)$$

17

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \lambda_{k+1} \cdots \lambda_h (Z^{h+1} - Z^h) \qquad \text{(using (24))}$$

$$= \lambda_h \underbrace{\left(\sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \lambda_{k+1} \cdots \lambda_{h-1} \alpha_k \boldsymbol{\phi}_k \right)}_{\doteq\, \boldsymbol{e}_{h-1}} (Z^{h+1} - Z^h)$$

$$= \lambda_h \boldsymbol{e}_{h-1} (Z^{h+1} - Z^h) . \qquad (25)$$

We again encounter the TD error $Z^{h+1} - Z^h$ and, more importantly, a new trace vector $\boldsymbol{e}_t$ that can be updated efficiently with

$$\boldsymbol{e}_t \doteq \sum_{k=0}^{t} \mathbf{F}_t \cdots \mathbf{F}_{k+1} \lambda_{k+1} \cdots \lambda_t \alpha_k \boldsymbol{\phi}_k$$

$$= \sum_{k=0}^{t-1} \mathbf{F}_t \cdots \mathbf{F}_{k+1} \lambda_{k+1} \cdots \lambda_t \alpha_k \boldsymbol{\phi}_k \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

$$= \lambda_t \mathbf{F}_t \underbrace{\sum_{k=0}^{t-1} \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \lambda_{k+1} \cdots \lambda_{t-1} \alpha_k \boldsymbol{\phi}_k}_{\boldsymbol{e}_{t-1}} \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

$$= \lambda_t \mathbf{F}_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t \qquad (26)$$

$$= \lambda_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \lambda_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}) .$$

This trace is similar to the one we encountered before, but with the difference that the value of the vector decays towards zero by multiplication with λ_t on each step. Predictions made prior to a fully trusted interim target (for which $\lambda_t = 0$) will never be affected by later interim targets because the trace vector is set to zero. The extent to which the trace extends backward in time depends on the extent to which we have not yet trusted the corresponding interim targets.

We now combine the derived update from $\boldsymbol{\theta}_h^h$ to $\boldsymbol{\theta}_h^{h+1}$ with the update from $\boldsymbol{\theta}_h^{h+1}$ to $\boldsymbol{\theta}_{h+1}^{h+1}$ to derive a single efficient update, given by

$$\boldsymbol{\theta}_{h+1}^{h+1} = \mathbf{F}_h \boldsymbol{\theta}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h Z^{h+1} \qquad \text{(using (22))}$$

$$= \mathbf{F}_h \left(\boldsymbol{\theta}_h^h + \lambda_h \boldsymbol{e}_{h-1}(Z^{h+1} - Z^h) \right) + \alpha_h \boldsymbol{\phi}_h Z^{h+1} \qquad \text{(using (25))}$$

$$= \mathbf{F}_h \boldsymbol{\theta}_h^h + \lambda_h \mathbf{F}_h \boldsymbol{e}_{h-1}(Z^{h+1} - Z^h) + \alpha_h \boldsymbol{\phi}_h Z^{h+1}$$

$$= \mathbf{F}_h \boldsymbol{\theta}_h^h + (\boldsymbol{e}_h - \alpha_h \boldsymbol{\phi}_h)\left(Z^{h+1} - Z^h \right) + \alpha_h \boldsymbol{\phi}_h Z^{h+1}$$
$$\qquad\qquad \text{(using } \lambda_h \mathbf{F}_h \boldsymbol{e}_{h-1} = \boldsymbol{e}_h - \alpha_h \boldsymbol{\phi}_h \text{, from (26))}$$

$$= \mathbf{F}_h \boldsymbol{\theta}_h^h + \boldsymbol{e}_h \left(Z^{h+1} - Z^h \right) + \alpha_h \boldsymbol{\phi}_h Z^h$$

$$= (\mathbf{I} - \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top) \boldsymbol{\theta}_h^h + \boldsymbol{e}_h \left(Z^{h+1} - Z^h \right) + \alpha_h \boldsymbol{\phi}_h Z^h$$

$$= \boldsymbol{\theta}_h^h + \boldsymbol{e}_h \left(Z^{h+1} - Z^h \right) + \alpha_h \boldsymbol{\phi}_h \left(Z^h - \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right) .$$

This concludes our derivation because the value of $\boldsymbol{\theta}_{h+1}^{h+1}$ is now defined fully in terms of the previous weights on the diagonal $\boldsymbol{\theta}_h^h$ and other quantities that are either directly

18

available upon reaching our new data horizon $h+1$ or, in the case of the trace vector, can be computed with constant $O(n)$ computation per step. The span-independent algorithm with persistent interim targets is

$$
\begin{aligned}
&\boldsymbol{e}_{-1} \doteq \boldsymbol{0}, \text{ then } \boldsymbol{e}_t \doteq \lambda_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \lambda_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), && t = 1, \ldots, T-1, \\
&\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \boldsymbol{e}_t \left(Z^{t+1} - Z^t \right) + \alpha_t \boldsymbol{\phi}_t (Z^t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t) && t = 0, \ldots, T-1.
\end{aligned}
\tag{27}
$$

Compared to the online backward view (11), the only difference is the appearance of λ_t in the update of the trace. If $\lambda_t = 1$ for all t, we regain the online backward view precisely, demonstrating that the new algorithm is strictly more general. In contrast to the averaging backward view (19), from the previous section, we see that for the notion of trust corresponding to λ we do not need to maintain separate online and trusted weights. Instead, the degree of trust is used to scale the trace vector $\boldsymbol{e}_t$ down accordingly. If $\lambda_t < 1$ for any $t \in \{1, \ldots, T\}$, the corresponding interim targets have a lasting effect on the final weight vector and therefore for the first time we may obtain predictions that differ not just during the episode but also at its end.

7 Combining two notions of trust and the emergence of averaged TD(λ)

Algorithm (27) is a strict generalization of the online algorithm (11), but it does not subsume the offline algorithm (6), or the averaging algorithm (19) that switches smoothly between online and offline updates. In this section, we combine the ideas from the last two sections to arrive at an algorithm that generalizes and subsumes all previous algorithms, thereby unifying all that came before into a single, general-purpose algorithm.

An offline version of the TD(λ) algorithm can be obtained by using the online algorithm in (27) to update an online weight vector $\tilde{\boldsymbol{\theta}}_t$ and then defining the trusted weight vector $\boldsymbol{\theta}_t$ to remain equal to the initial weights until the last step, at which time we replace them with the online weights. An algorithm that switches smoothly between the offline and online cases can then be obtained similar to before, resulting in

$$
\begin{aligned}
&\boldsymbol{e}_{-1} \doteq \boldsymbol{0}, \text{ then } \boldsymbol{e}_t \doteq \lambda_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \lambda_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), && t = 0, \ldots, T-1, \\
&\tilde{\boldsymbol{\theta}}_{t+1} \doteq \tilde{\boldsymbol{\theta}}_t + \boldsymbol{e}_t \left(Z^{t+1} - Z^t \right) + \alpha_t \boldsymbol{\phi}_t (Z^t - \boldsymbol{\phi}_t^\top \tilde{\boldsymbol{\theta}}_t), && t = 0, \ldots, T-1, \\
&\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \beta_{t+1} (\tilde{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\theta}_t), && t = 0, \ldots, T-1.
\end{aligned}
\tag{28}
$$

The first two lines are the online algorithm (27), from the previous section. The last line is equal to the last line in the unified algorithm without persistency of interim targets, as given in (19), but now using the online weights that use persistent interim targets weighted according to λ-returns, as computed in the first two lines. When $\lambda_t = 1$ for all t we regain the averaging algorithm (19) without persistent interim targets. When $\beta_t = 1$ for all t we regain the online algorithm (27) with persistent interim targets. So, we have again successfully unified all previously seemingly difference approaches to trust and have arrived at a single general algorithm that subsumes all that came before.

The merits of λ-returns are well known (Sutton 1988; Sutton and Barto 1998) but the β-weighting of the online weights is novel to this paper, and it is appropriate to discuss it in a little more detail. So far we have considered only a single episode, but a major

potential benefit of including β appears when we consider multiple episodes. For clarity, consider the extreme case where all episodes last only a single step. Then λ plays no role because there is no interim within each episode; there is only a beginning and an end. If we use m to enumerate episodes, such that Z_m is the true outcome of episode m, then the updates for single-step episodes are

$$\tilde{\boldsymbol{\theta}}_{m+1} = \tilde{\boldsymbol{\theta}}_m + \alpha_m(Z_m - \boldsymbol{\phi}_m^\top \tilde{\boldsymbol{\theta}}_m)\boldsymbol{\phi}_m\,,$$
$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \beta_{m+1}(\tilde{\boldsymbol{\theta}}_{m+1} - \boldsymbol{\theta}_m)\,,$$

where we used the fact that the final weights of episode n are the first weights of episode $m + 1$. Using β, we can do something here that we cannot do with λ alone: we can weight the relative impact of different episodes. For instance, we can choose to keep the weights and the predictions stationary over multiple episodes, by setting $\beta_m = 0$, to reduce the impact of the final outcomes observed in those episodes on our predictions. Another possibility is to decay the trust, for instance according to $\beta_m = \frac{1}{m}$, such that we trust the outcome of the first episode fully ($\beta_1 = 1$) and then reduce the trust for each subsequent episode. Such a choice of β makes sense if we view the trust we place in outcomes as being relative to the trust we place in the predictions we already have. As our predictions improve over time, the outcomes become relatively less trustworthy. With this definition of trust, the trusted weights are the average of the weights of all previous episodes: $\boldsymbol{\theta}_m = \frac{1}{m}\sum_{i=0}^m \tilde{\boldsymbol{\theta}}_i$. This specific algorithm is interesting because the predictions according to the averages $\boldsymbol{\theta}_m$ are known to converge to the optimal predictions faster than the predictions according to any sequence of online weights $\tilde{\boldsymbol{\theta}}_m$ (Polyak and Juditsky 1992; Bach and Moulines 2013). This shows that the notion of trust as provided by β gives us something that cannot be obtained with λ alone. To our knowledge, algorithm (28) is the first to generalize this idea of averaging online weights, in a principled fashion, to long-term predictions.

8 Generalizing to cumulative returns and soft terminations

In this section, we discuss how to extend our algorithms to handle soft terminations and cumulative returns. Both extensions generalize the episodic final-outcome setting considered above, and the algorithm we derive in this section will subsume all previous algorithms.

Often, we want to predict the cumulation of a signal $\{X_t\}_{t=0}^T$ rather than a single final outcome. In the episodic setting, with termination at time T, we then aim to predict

$$Z_t^T \doteq X_{t+1} + X_{t+2} + \ldots + X_T\,,$$

where, as before, t is the time of the prediction. In contrast to final outcomes, these cumulative outcomes depend on the time step of the prediction because at later time steps there will be less signal left to accumulate before the episode ends. We call this time-dependent outcome the *cumulative return*.

We may wish to update our predictions online, before observing the full cumulative return. To do so, we need to define interim targets to temporarily take the place of the

actual return. An interim target Z_t^h up to a horizon $h < T$ should in any case include the part of the signal that was already observed. In addition, we introduce a residual prediction P_h that stands in for the unseen part of the signal, from horizon h to the end of the episode T. The full interim target at time t up to horizon h is then

$$Z_t^h = X_{t+1} + X_{t+2} + \ldots + X_h + P_h \,.$$

The residual prediction P_h may for instance be given by an external expert or by our own predictions at time h. For now we are agnostic to its origin and consider the general case. In any case, we define $P_T = 0$ because at termination there is no remaining signal left to predict. If all signals except the one coinciding with termination are zero, we regain the final-outcome setting where the last signal $X_T = Z$ takes the role of the final outcome. If we further define the interim target at each horizon to be equal to the residual prediction at that horizon, such that $Z^h = P_h$, we are back in the online final-outcome setting. Therefore, cumulative returns are strictly more general than final outcomes.

So far, we have considered episodic predictions where each episode ends with a single final outcome that we wish to predict. The algorithms extend naturally to multiple episodes by using the final weights of the episode as the initial weights of the next episode. However, some predictive questions do not fit nicely into this strictly episodic format because they are better thought of as terminating *softly* on each time step.

A soft termination is a conceptual and potentially partial termination of the signal. Such a soft termination could represent a probability of termination, for instance when we want to take into account the probability of a robot breaking while learning in a simulation in which it never actually does. Or the soft termination could represent a desire to trade off the imminence and the magnitude of a signal, for instance when we do not just want more money rather than less, but we also want it sooner rather than later. In both cases the prediction is about a diminishing version of the 'raw' signal (e.g., money). Here, we are not concerned with the potential reasons for using soft terminations and consider the general case where the termination of the prediction can vary per time step and be anywhere between full continuation and full termination. Soft terminations allow us to ask more general predictive questions, and even to simultaneously consider multiple predictions that may resolve at different times.

Soft terminations can be modeled by using a continuation parameter $\gamma_t \in [0, 1]$ to denote the amount of termination of our prediction upon reaching time t. This quantity is often called a discount factor, because it discounts the impact of later outcomes compared to earlier ones. If $\gamma_t = 1$, no termination happens at time t; if $\gamma_t = 0$, the prediction terminates fully, even if the trajectory may continue. We consider general sequences of γ_t and only require that eventually every prediction resolves completely, potentially asymptotically, such that $\prod_{i=t}^{\infty} \gamma_i = 0$ for all t. The episodic setting considered in the previous sections is a special case where $\gamma_t = 0$ only if $t = T$ is the final step of the episode, and $\gamma_t = 1$ on all other steps.

We are now almost ready to formulate the target for a prediction about a discounted cumulative signal. In order to maintain full generality and compatibility with previous sections, we immediately include persistency of the residual predictions according to a sequence of $\{\lambda_t\}_{t=0}^{\infty}$ and trust of the resulting combined targets according to a sequence $\{\beta_t\}_{t=0}^{\infty}$. For any horizon $h > t$, the combined interim target for the prediction at time

t should include at least the immediate next signal X_{t+1}. Beyond this first signal we continue with γ_{t+1} and observe the residual prediction P_{t+1}. We trust this prediction with degree $1 - \lambda_{t+1}$ as a stand-in for the expected cumulative discounted return, and so its total multiplier is $\gamma_{t+1}(1 - \lambda_{t+1})$. If we trust this prediction fully, so that $\lambda_t = 0$, there is no need to continue further. Otherwise, we continue for as much as we have not yet terminated according to both γ_{t+1} and λ_{t+1}, so that the next signal X_{t+2} gets a total weight of $\gamma_{t+1}\lambda_{t+1}$. This line of reasoning then continues until we reach the horizon of our current data at time h, at which point we place any remaining trust on the most recent residual prediction P_h. All together, this gives us a combined target for the prediction at time t with data up to time $h > t$:

$$
\begin{aligned}
Z_t^h \doteq\; & X_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1})P_{t+1} \\
& + \gamma_{t+1}\lambda_{t+1}(X_{t+2} + \gamma_{t+2}(1 - \lambda_{t+2})P_{t+2}) \\
& + \gamma_{t+1}\lambda_{t+1}\gamma_{t+2}\lambda_{t+2}(X_{t+3} + \gamma_{t+3}(1 - \lambda_{t+3})P_{t+3}) \\
& \;\;\vdots \\
& + \gamma_{t+1}\cdots\gamma_{h-2}\lambda_{t+1}\cdots\lambda_{h-2}(X_{h-1} + \gamma_{h-1}(1 - \lambda_{h-1})P_{h-1}) \\
& + \gamma_{t+1}\cdots\gamma_{h-1}\lambda_{t+1}\cdots\lambda_{h-1}(X_h + \gamma_h P_h)\,.
\end{aligned}
\tag{29}
$$

This can be written recursively as

$$
Z_t^h \doteq X_{t+1} + \gamma_{t+1}\big((1 - \lambda_{t+1})P_{t+1} + \lambda_{t+1}Z_{t+1}^h\big)\,,
\tag{30}
$$
$$
\text{and}\;\; Z_t^t \doteq P_t\,,
\tag{31}
$$

Each of these interim targets is trusted according to the trust associated with its horizon, β_h. Recall from Section 5 that this trust propagates backward. If we trust the last target Z_t^h fully, there is no need to consider earlier targets. Otherwise, we multiply Z_t^h with its associated trust β_h and continue to the previous target with degree $(1 - \beta_h)$. This then continues until either we find a target we fully trust, or we reach the then-current predictions $\phi_t^\top \theta_t^t$ at time t, when we made the prediction we are currently updating. Our final interim targets are then given by

$$
\begin{aligned}
\bar{Z}_t^h \doteq\; & \beta_h Z_t^h \\
& + (1 - \beta_h)\beta_{h-1}Z_t^{h-1} \\
& + (1 - \beta_h)(1 - \beta_{h-1})\beta_{h-2}Z_t^{h-2} \\
& \;\;\vdots \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+2})\beta_{t+1}Z_t^{t+1} \\
& + (1 - \beta_h)\cdots(1 - \beta_{t+1})\phi_t^\top \theta_t^t \\
=\; & \beta_h Z_t^h + (1 - \beta_h)\bar{Z}_t^{h-1}
\end{aligned}
\tag{32}
$$
$$
\text{and}\;\; \bar{Z}_t^t \doteq \phi_t^\top \theta_t^t
\tag{33}
$$

where θ_t^t are the trusted weights for time t that we will compute. The forward-viewing update is then given by

$$
\theta_0^t \doteq \theta_0 \qquad\qquad\qquad t = 0,\ldots,T
$$

22

$$\boldsymbol{\theta}_{t+1}^h \doteq \boldsymbol{\theta}_t^h + \alpha_t \boldsymbol{\phi}_t (\bar{Z}_t^h - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t^h), \qquad t = 0, \ldots h-1; \quad h = 0, \ldots, T, \qquad (34)$$

$$\doteq \mathbf{F}_t \boldsymbol{\theta}_t^h + \alpha_t \boldsymbol{\phi}_t \bar{Z}_t^h, \qquad t = 0, \ldots h-1; \quad h = 0, \ldots, T.$$

To find an efficient backward view, again we can start by writing down explicitly the value of $\boldsymbol{\theta}_t^h$. Following a derivation identical to the one for when we first consider trusted weights, as shown in (16) but with the complex target $\bar{Z}_k^h$ replacing Z_k^h, we obtain

$$\boldsymbol{\theta}_t^h = \boldsymbol{a}_{t-1} + \sum_{k=0}^{t-1} \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \bar{Z}_k^h \qquad t = 0, \ldots, h-1; \quad h = 1, \ldots, T. \qquad (35)$$

As before, we can apply this to both $\boldsymbol{\theta}_h^{h+1}$ and $\boldsymbol{\theta}_h^h$ to find the difference

$$\boldsymbol{\theta}_h^{h+1} - \boldsymbol{\theta}_h^h = \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \bar{Z}_k^{h+1} \quad - \quad \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \bar{Z}_k^h$$

$$(\boldsymbol{a}_{h-1} \text{ cancels})$$

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \left(\bar{Z}_k^{h+1} - \bar{Z}_k^h \right)$$

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \left(\beta_{h+1} Z_k^{h+1} + (1 - \beta_{h+1}) \bar{Z}_k^h - \bar{Z}_k^h \right) \qquad \text{(from (32))}$$

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \beta_{h+1} \left(Z_k^{h+1} - \bar{Z}_k^h \right)$$

$$= \beta_{h+1} \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k Z_k^{h+1} \quad - \quad \beta_{h+1} \underbrace{\sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \bar{Z}_k^h}_{\doteq \boldsymbol{\theta}_h^h - \boldsymbol{a}_{h-1}, \text{ from (35)}}$$

$$= \beta_{h+1} \left(\boldsymbol{a}_{h-1} + \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k Z_k^{h+1} \right) \quad - \quad \beta_{h+1} \boldsymbol{\theta}_h^h. \qquad (36)$$

The first part of this result, within the brackets, looks familiar: notice the similarity to (35). The term can be interpreted as the intermediate result $\tilde{\boldsymbol{\theta}}_h^{h+1}$ of a forward view with targets Z_t^h, as defined in (30), and updates defined by

$$\begin{aligned} \tilde{\boldsymbol{\theta}}_0^t &\doteq \tilde{\boldsymbol{\theta}}_0 & t = 0, \ldots, T \\ \tilde{\boldsymbol{\theta}}_{t+1}^h &\doteq \tilde{\boldsymbol{\theta}}_t^h + \alpha_t \boldsymbol{\phi}_t (Z_t^h - \boldsymbol{\phi}_t^\top \tilde{\boldsymbol{\theta}}_t^h), & t = 0, \ldots h-1; \quad h = 0, \ldots, T. \end{aligned} \qquad (37)$$

This is an online algorithm, comparable to the algorithm for λ-returns derived before, but implicitly including cumulative returns and discounting through the definition of Z_t^h. The complete forward view (34) can then be interpreted as switching smoothly between this online algorithm and an offline algorithm that in the extreme can delay updating

23

the predictions indefinitely. Analogous to the interim weights shown in (35), the interim weights of this online algorithm satisfy

$$\tilde{\boldsymbol{\theta}}_t^h = \boldsymbol{a}_{t-1} + \sum_{k=0}^{t-1} \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k Z_k^h \qquad t = 0, \ldots, h\,;\quad h = 1, \ldots, T\,. \qquad (38)$$

We can then continue our derivation from (36) with

$$\boldsymbol{\theta}_h^{h+1} - \boldsymbol{\theta}_h^h = \beta_{h+1} \underbrace{\left(\boldsymbol{a}_{h-1} + \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k Z_k^{h+1} \right)}_{= \tilde{\boldsymbol{\theta}}_h^{h+1},\text{ using } (38)} - \beta_{h+1} \boldsymbol{\theta}_h^h$$

$$= \beta_{h+1} (\tilde{\boldsymbol{\theta}}_h^{h+1} - \boldsymbol{\theta}_h^h)\,.$$

This implies that if we have $\boldsymbol{\theta}_h^h$ and $\tilde{\boldsymbol{\theta}}_h^{h+1}$ we can then compute $\boldsymbol{\theta}_h^{h+1}$ efficiently using

$$\boldsymbol{\theta}_h^{h+1} = (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h + \beta_{h+1} \tilde{\boldsymbol{\theta}}_h^{h+1}\,. \qquad (39)$$

For now, we ignore the question of how to obtain $\tilde{\boldsymbol{\theta}}_h^{h+1}$ and first focus on how to go from $\boldsymbol{\theta}_h^{h+1}$ to $\boldsymbol{\theta}_{h+1}^{h+1}$ and, resultingly, from $\boldsymbol{\theta}_h^h$ to $\boldsymbol{\theta}_{h+1}^{h+1}$. We can now derive

$$\boldsymbol{\theta}_{h+1}^{h+1} = \mathbf{F}_h \boldsymbol{\theta}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h \bar{Z}_h^{h+1} \qquad \text{(using (34))}$$

$$= \mathbf{F}_h \left(\beta_{h+1} \tilde{\boldsymbol{\theta}}_h^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h \right) + \alpha_h \boldsymbol{\phi}_h \bar{Z}_h^{h+1} \qquad \text{(using (39))}$$

$$= \mathbf{F}_h \left(\beta_{h+1} \tilde{\boldsymbol{\theta}}_h^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h \right) + \alpha_h \boldsymbol{\phi}_h \left(\beta_{h+1} Z_h^{h+1} + (1 - \beta_{h+1}) \bar{Z}_h^h \right)$$
$$\qquad \text{(using (32))}$$

$$= \mathbf{F}_h \left(\beta_{h+1} \tilde{\boldsymbol{\theta}}_h^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h \right) + \alpha_h \boldsymbol{\phi}_h \left(\beta_{h+1} Z_h^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right)$$
$$\qquad \text{(using (33))}$$

$$= \beta_{h+1} \underbrace{\left(\mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h Z_h^{h+1} \right)}_{\doteq\, \tilde{\boldsymbol{\theta}}_{h+1}^{h+1},\text{ using } (37)} + (1 - \beta_{h+1}) \underbrace{\left(\mathbf{F}_h \boldsymbol{\theta}_h^h + \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top \boldsymbol{\theta}_h^h \right)}_{= \boldsymbol{\theta}_h^h}$$
$$\qquad \text{(regrouping)}$$

$$= \beta_{h+1} \tilde{\boldsymbol{\theta}}_{h+1}^{h+1} + (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h\,.$$

We see that $\tilde{\boldsymbol{\theta}}_h^{h+1}$ is not needed and instead we can use $\tilde{\boldsymbol{\theta}}_{h+1}^{h+1}$. Therefore, if $\tilde{\boldsymbol{\theta}}_{h+1}^{h+1}$ can be computed with constant computation independent of span, then $\boldsymbol{\theta}_{h+1}^{h+1}$ can be computed from $\boldsymbol{\theta}_h^h$ efficiently as well, using

$$\boldsymbol{\theta}_{h+1}^{h+1} = (1 - \beta_{h+1}) \boldsymbol{\theta}_h^h + \beta_{h+1} \tilde{\boldsymbol{\theta}}_{h+1}^{h+1}\,. \qquad (40)$$

It now remains to be shown that the online weights $\tilde{\boldsymbol{\theta}}_{h+1}^{h+1}$ can be computed efficiently. We start with the difference between $\tilde{\boldsymbol{\theta}}_{h}^{h+1}$ and $\tilde{\boldsymbol{\theta}}_{h}^{h}$ for which we can use (38) to obtain

$$\tilde{\boldsymbol{\theta}}_{h}^{h+1} - \tilde{\boldsymbol{\theta}}_{h}^{h} = \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \left(Z_k^{h+1} - Z_k^h \right) . \tag{41}$$

We can find the difference $Z_k^{h+1} - Z_k^h$ from the definition in (29). All terms not involving P_h, X_{h+1} or P_{h+1} cancel, leaving us with

$$Z_k^{h+1} - Z_k^h = \underbrace{\gamma_{k+1} \cdots \gamma_h \lambda_{k+1} \cdots \lambda_{h-1} \left((1 - \lambda_h) P_h + \lambda_h X_{h+1} + \lambda_h \gamma_{h+1} P_{h+1} \right)}_{\text{from } Z_k^{h+1}}$$

$$- \underbrace{\gamma_{k+1} \cdots \gamma_h \lambda_{k+1} \cdots \lambda_{h-1} P_h}_{\text{from } Z_k^h}$$

$$= \gamma_{k+1} \cdots \gamma_h \lambda_{k+1} \cdots \lambda_{h-1} \left(-\lambda_h P_h + \lambda_h X_{h+1} + \lambda_h \gamma_{h+1} P_{h+1} \right)$$

$$= \gamma_{k+1} \cdots \gamma_h \lambda_{k+1} \cdots \lambda_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) .$$

Here we see the emergence of a general form of the classical temporal-difference error for cumulative discounted returns: $\delta_h \doteq X_{h+1} + \gamma_{h+1} P_{h+1} - P_h$.[3] Using this, we can now continue from (41) with

$$\tilde{\boldsymbol{\theta}}_{h}^{h+1} - \tilde{\boldsymbol{\theta}}_{h}^{h} = \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \left(Z_k^{h+1} - Z_k^h \right)$$

$$= \sum_{k=0}^{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \gamma_{k+1} \cdots \gamma_h \lambda_{k+1} \cdots \lambda_h \delta_h$$

$$= \gamma_h \lambda_h \underbrace{\left(\sum_{k=0}^{h-1} \gamma_{k+1} \cdots \gamma_{h-1} \lambda_{k+1} \cdots \lambda_{h-1} \mathbf{F}_{h-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \right)}_{\doteq \, \boldsymbol{e}_{h-1}} \delta_h$$

$$= \gamma_h \lambda_h \boldsymbol{e}_{h-1} \delta_h , \tag{42}$$

where, similar to before, the trace vector $\boldsymbol{e}_t$ can be updated recursively according to

$$\boldsymbol{e}_t \doteq \sum_{k=0}^{t} \gamma_{k+1} \cdots \gamma_t \lambda_{k+1} \cdots \lambda_t \mathbf{F}_t \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k$$

$$= \sum_{k=0}^{t-1} \gamma_{k+1} \cdots \gamma_t \lambda_{k+1} \cdots \lambda_t \mathbf{F}_t \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

---

[3]If we use our current predictions as interim targets, the TD error is $\delta_t = X_{t+1} + \gamma_{t+1} \boldsymbol{\phi}_{t+1}^\top \boldsymbol{\theta}_t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_{t-1}$. This TD error uses the weights at two consecutive time steps and is therefore slightlt different from the classic TD error defined, using only the current weights $\boldsymbol{\theta}_t$, as $\delta_t = X_{t+1} + \gamma_{t+1} \boldsymbol{\phi}_{t+1}^\top \boldsymbol{\theta}_t - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t$. The difference is important to achieve exact equivalence, although it is also possible to rewrite the new algorithms to use a more standard TD error.

$$= \gamma_t \lambda_t \mathbf{F}_t \sum_{k=0}^{t-1} \gamma_{k+1} \cdots \gamma_{t-1} \lambda_{k+1} \cdots \lambda_{t-1} \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k \quad + \quad \alpha_t \boldsymbol{\phi}_t$$

$$= \gamma_t \lambda_t \mathbf{F}_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t \tag{43}$$

$$= \gamma_t \lambda_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \gamma_t \lambda_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}).$$

The trace now decays according to both λ and γ. Using this trace vector, we can compute $\tilde{\boldsymbol{\theta}}_h^{h+1}$ efficiently from $\tilde{\boldsymbol{\theta}}_h^h$ using (42). We now combine this with the final step to $\tilde{\boldsymbol{\theta}}_{h+1}^{h+1}$ from $\tilde{\boldsymbol{\theta}}_h^{h+1}$ and derive

$$\tilde{\boldsymbol{\theta}}_{h+1}^{h+1} = \mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h Z_h^{h+1} \qquad\qquad\qquad\qquad (\text{using (37)})$$

$$= \mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^{h+1} + \alpha_h \boldsymbol{\phi}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} \right) \qquad\qquad (\text{using (30), (31)})$$

$$= \mathbf{F}_h \left(\tilde{\boldsymbol{\theta}}_h^h + \gamma_h \lambda_h \boldsymbol{e}_{h-1} \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) \right) + \alpha_h \boldsymbol{\phi}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} \right)$$
$$(\text{using (42)})$$

$$= \mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^h + \gamma_h \lambda_h \mathbf{F}_h \boldsymbol{e}_{h-1} \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) + \alpha_h \boldsymbol{\phi}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} \right)$$

$$= \mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^h + \left(\boldsymbol{e}_h - \alpha_h \boldsymbol{\phi}_h \right) \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) + \alpha_h \boldsymbol{\phi}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} \right)$$
$$(\text{using } \gamma_h \lambda_h \mathbf{F}_h \boldsymbol{e}_{h-1} = \boldsymbol{e}_h - \alpha_h \boldsymbol{\phi}_h, \text{ from (43)})$$

$$= \mathbf{F}_h \tilde{\boldsymbol{\theta}}_h^h + \boldsymbol{e}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) + \alpha_h \boldsymbol{\phi}_h P_h$$

$$= (\mathbf{I} - \alpha_h \boldsymbol{\phi}_h \boldsymbol{\phi}_h^\top) \tilde{\boldsymbol{\theta}}_h^h + \boldsymbol{e}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) + \alpha_h \boldsymbol{\phi}_h P_h$$

$$= \tilde{\boldsymbol{\theta}}_h^h + \boldsymbol{e}_h \left(X_{h+1} + \gamma_{h+1} P_{h+1} - P_h \right) + \alpha_h \boldsymbol{\phi}_h \left(P_h - \boldsymbol{\phi}_h^\top \tilde{\boldsymbol{\theta}}_h^h \right)$$

$$= \tilde{\boldsymbol{\theta}}_h^h + \boldsymbol{e}_h \delta_h + \alpha_h \boldsymbol{\phi}_h \left(P_h - \boldsymbol{\phi}_h^\top \tilde{\boldsymbol{\theta}}_h^h \right).$$

All weights again have matching sub- and superscripts and an equivalent TD algorithm, in the sense that $\tilde{\boldsymbol{\theta}}_t = \tilde{\boldsymbol{\theta}}_t^t$ and $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t^t$, for the fully general case including cumulative discounted returns is given by

$$\begin{array}{ll} \boldsymbol{e}_{-1} \doteq \mathbf{0}, \text{ then } \boldsymbol{e}_t \doteq \gamma_t \lambda_t \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t (1 - \gamma_t \lambda_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1}), & t = 0, \ldots, T-1, \\ \tilde{\boldsymbol{\theta}}_{t+1} \doteq \tilde{\boldsymbol{\theta}}_t + \boldsymbol{e}_t \delta_t + \alpha_t \boldsymbol{\phi}_t (P_t - \boldsymbol{\phi}_t^\top \tilde{\boldsymbol{\theta}}_t), & t = 0, \ldots, T-1, \\ \boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \beta_{t+1} (\tilde{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\theta}_t), & t = 0, \ldots, T-1. \end{array} \tag{44}$$

The first two lines constitute the online algorithm we just derived. The last line is from (40) and extends this algorithm to include smooth switching between offline and online updates. If $\beta_t = 1$ for all t, the algorithm reduces to a variant of TD(λ) known as true online TD(λ) (van Seijen and Sutton 2014), but extended to include general, potentially non-constant, sequences of $\{\alpha_t\}$, $\{\gamma_t\}$ and $\{\lambda_t\}$. The extension to averaging according to β_t is new to this paper.

Soft termination generalizes the episodic setting we considered previously. This means that as far as the learning update is concerned, we do not have to treat steps on which the process actually terminates and restarts for a new episode in any special way. To see how this works, we first renumber the time steps on consecutive episodes: if the first episode ends at time T, the initial time step of the second episode will be taken to be T rather than 0. If the second episode lasts T' steps, the third episode is then taken to

26

begin on $T + T'$, and so on. Together with the requirement that $\gamma_T = 0$ on every actual termination, this is sufficient to get updates that are completely equivalent to treating the subsequent episodes completely separately. Notice that the update on termination at some time T, and resulting in $\boldsymbol{\theta}_T$, uses the residual prediction on termination P_T only when multiplied with γ_T. Previously we required that $P_T = 0$ because there is no further signal to predict. This is now no longer necessary, because $\gamma_T = 0$ already fulfills the requirement that $\gamma_T P_T = 0$. If for instance we use our current predictions, such that $P_{t+1} \doteq \boldsymbol{\phi}_{t+1}^\top \boldsymbol{\theta}_t$ for all t, we can simply keep the update as is even though P_T will then be the prediction for the cumulative return of the next episode, because $\boldsymbol{\phi}_T$ is now defined to be its first feature vector. Therefore, both hard and soft termination can be handled seamlessly using algorithm (44), and that will be our final, general algorithm.

9   Convergence Analysis

The algorithm (44) differs from related earlier algorithms such as TD(λ) in a few subtle but important ways. The most notable differences are the updates to the traces and the averaging due to β_t. Known results on convergence therefore do not automatically transfer to this new algorithm and it is appropriate to take a moment to analyze it.

The convergence of the trusted weights $\boldsymbol{\theta}$ depends on the convergence of the online weights $\tilde{\boldsymbol{\theta}}$ and so we must investigate these jointly. The online weights, in turn, depend on the sequences of parameters and residual predictions that are supplied. We want our analysis to be general, which means we want to be able to handle general sequences of discounts $\{\gamma_t\}_{t=1}^\infty$, persistency parameters, $\{\lambda_t\}_{t=1}^\infty$, and residual predictions $\{P_t\}_{t=1}^\infty$. Naturally, if any of these can change completely arbitrarily, we can have no hope of converging to any predeterminable solution. Therefore, as in Sutton et al. (2014), we allow the features, discounts and persistency parameters to be stationary functions of an underlying unobserved state, such that $\boldsymbol{\phi}_t \doteq \boldsymbol{\phi}(S_t)$, $\gamma_t \doteq \gamma(S_t)$ and $\lambda_t \doteq \lambda(S_t)$ for some fixed functions $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^n$, $\gamma : \mathcal{S} \to [0, 1]$ and $\lambda : \mathcal{S} \to [0, 1]$, where $\mathcal{S}$ is a state space and $S_t \in \mathcal{S}$ is the, unobserved, state of the world at time t. We assume there is a steady-state distribution over these states such that all expectations used below are well-defined with respect to a distribution over states defined by $\lim_{t \to \infty} \Pr(S_t = s)$. This setting generalizes the more standard approach where $\gamma_t = \gamma$ and $\lambda_t = \lambda$ are constants, because now these parameters can still change over time, but it avoids the possibility of arbitrary non-stationarity that would ruin convergence.

We first consider convergence when the residual predictions are also due to a fixed function of state, for instance because they are due to otherwise stationary experts or oracles.

Theorem 1. *Let* $X_t \doteq X(S_t)$, $\boldsymbol{\phi}_t \doteq \boldsymbol{\phi}(S_t)$, $\gamma_t \doteq \gamma(S_t)$, $\lambda_t \doteq \lambda(S_t)$ *and* $P_t \doteq P(S_t)$ *all be fixed functions of (unobserved) states* $S_t \in \mathcal{S}$, *with a stable steady-state distribution* d. *Then, if* $\sum_{t=0}^\infty \alpha_t = \sum_{t=0}^\infty \beta_t = \infty$, *and* $\sum_{t=0}^\infty \alpha_t^2 < \infty$, *algorithm* (44) *converges almost surely to the fixed-point solution*

$$\boldsymbol{\theta}_* \doteq \mathbb{E}[\boldsymbol{\phi}_t \boldsymbol{\phi}_t]^{-1} \, \mathbb{E}[Z_t^\infty \boldsymbol{\phi}_t] \ .$$

Proof. We start by analyzing the online weights $\tilde{\boldsymbol{\theta}}_t$. Because of the equivalence of the forward and backward views, we can investigate the forward view, which is easier to

analyze. In other words, instead of investigating $\lim_{t\to\infty}\tilde{\boldsymbol{\theta}}_t$ as updated through (44), we investigate $\lim_{t\to\infty}\tilde{\boldsymbol{\theta}}_t^t$ as updated through (37). By construction, the end result is exactly the same. The asymptotic forward view as the horizon goes to infinity is

$$\tilde{\boldsymbol{\theta}}_{t+1}^\infty = \tilde{\boldsymbol{\theta}}_t^\infty + \alpha_t\boldsymbol{\phi}_t(Z_t^\infty - \boldsymbol{\phi}_t^\top\tilde{\boldsymbol{\theta}}_t^\infty)\,, \qquad\qquad t = 0,\dots,$$

$$\text{where } Z_t^\infty \doteq \lim_{h\to\infty} Z_t^h\,, \qquad\qquad t = 0,\dots\,.$$

Because Z_t^h does not depend on the weights, this is a standard stochastic gradient-descent update $\tilde{\boldsymbol{\theta}}_{t+1} = \tilde{\boldsymbol{\theta}}_t - \alpha_t\nabla_{\tilde{\boldsymbol{\theta}}}l(\tilde{\boldsymbol{\theta}})|_{\tilde{\boldsymbol{\theta}}_t}$ on the quadratic loss function

$$l(\tilde{\boldsymbol{\theta}}) \doteq \mathbb{E}\Big[(Z_t^\infty - \boldsymbol{\phi}_t^\top\tilde{\boldsymbol{\theta}})^2\Big]\,.$$

If the step sizes are suitably chosen, for instance such that $\sum_{t=0}^\infty \alpha_t = \infty$ and $\sum_{t=0}^\infty \alpha_t^2 < \infty$ (Robbins and Monro 1951), and if the means and variances of Z_t^∞ and $\boldsymbol{\phi}_t$ are well-defined and bounded for all t, this update converges to the fixed-point solution $\boldsymbol{\theta}_*$ that minimizes the quadratic loss (cf. Kushner and Yin 2003), such that

$$\lim_{t\to\infty}\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_* \doteq \mathbb{E}\big[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top\big]^{-1}\mathbb{E}[\boldsymbol{\phi}_t Z_t^\infty]\,.$$

It is straightforward to see that $\boldsymbol{\theta}_t$ will have the same limit; it suffices to have $\sum_{t=0}^\infty \beta_t = \infty$. $\square$

Although convergence is already guaranteed when $\beta_t = 1$ for all t, recent work has shown that for similar stochastic gradient algorithms the optimal rate of convergence is attained if β_t decreases much faster than α_t, specifically when $\beta_t = O(t^{-1})$ while $\alpha_t = \alpha$ for some constant α (Bach and Moulines 2013). More generally, it seems likely that convergence also holds if $\sum_{t=0}^\infty \beta_t^2 < \infty$ and $\sum_{t=0}^\infty \alpha_t^2 = \infty$. The observation that β_t should perhaps decrease over time for faster learning may seem at odds with our introduction of this parameter as a degree of trust. However, these two views are quite compatible if we consider β_t to be the degree of trust we place in the online updates relative to the trust we place in our current predictions due to the trusted weights. When the trust in the predictions increases over time, the relative trust in the inherently noisy online targets should then decrease.

Although Theorem 1 is already fairly general, it does not cover the important case when the residual predictions additionally depend on the weights we are updating. It makes sense to use the predictions we trust most and therefore we now consider what happens when $P_t \doteq \boldsymbol{\phi}_t^\top\boldsymbol{\theta}_{t-1}$. Notice that we have to use $\boldsymbol{\theta}_{t-1}$ rather than $\boldsymbol{\theta}_t$, because P_t is used in the computation of $\boldsymbol{\theta}_t$ and so the latter is not yet available when we compute P_t. The analysis of this case is more complex than the previous one, because P_t is no longer a constant function of state. This means the update is no longer a standard gradient-descent update on a quadratic loss, because the target Z_t^h for the online weights itself depends on the trusted weights that we are simultaneously updating.

The results by Bach and Moulines (2013) on stochastic gradient descent indicate that perhaps the most interesting case is where β_t decreases faster than α_t, such that $\lim_{t\to\infty}\beta_t/\alpha_t = 0$. This suggests an analysis on two time scales is appropriate.

Theorem 2. *Let* $X_t \doteq X(S_t)$, $\boldsymbol{\phi}_t \doteq \boldsymbol{\phi}(S_t)$, $\gamma_t \doteq \gamma(S_t)$ *and* $\lambda_t \doteq \lambda(S_t)$ *all be fixed bounded functions of (unobserved) states* $S_t \in \mathcal{S}$, *with a stable steady-state distribution* d. *Define* $P_t \doteq \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_{t-1}$. *Then, if* $\sum_{t=0}^\infty \alpha_t = \sum_{t=0}^\infty \beta_t = \infty$, $\sum_{t=0}^\infty \alpha_t^2 < \infty$, *and* $\lim_{t\to\infty} \frac{\beta_t}{\alpha_t} = 0$, *algorithm* (44) *converges almost surely to the TD fixed-point solution* $\boldsymbol{\theta}_*$ *that minimizes the mean-squared projected Bellman error (Sutton, Szepesvári, and Maei 2008; Sutton et al. 2009), such that*

$$\mathbb{E}\big[(Z_t(\boldsymbol{\theta}_*) - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_*)\boldsymbol{\phi}_t^\top\big]\, \mathbb{E}[\boldsymbol{\phi}_t \boldsymbol{\phi}_t]^{-1}\, \mathbb{E}\big[\boldsymbol{\phi}_t(Z_t(\boldsymbol{\theta}_*) - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_*)\big] = 0\,, \tag{45}$$

where

$$Z_t(\boldsymbol{\theta}) \doteq X_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1})\boldsymbol{\phi}_t^\top \boldsymbol{\theta} + Z_t(\boldsymbol{\theta})\,, \quad \forall \boldsymbol{\theta}\,.$$

Proof. In two-time-scale analyses, we are allowed to analyze the faster updates as if the slower updates have stopped. This means that in analyzing the updates to the online weights $\tilde{\boldsymbol{\theta}}_t$, we can assume the trusted weights $\boldsymbol{\theta}_t$ are constant to analyze where $\tilde{\boldsymbol{\theta}}_t$ converges towards as a function of the stationary $\boldsymbol{\theta}_t$. On the other hand, when we analyze the slower updates to the trusted weights $\boldsymbol{\theta}_t$ we are allowed to assume the faster updates to the online weights converge completely between each two steps. For more detail on analyzing stochastic approximations on two times scales, we refer to Borkar (1997), Borkar (2008), Kushner and Yin (2003), and Konda and Tsitsiklis (2004).

We first analyze the convergence of the faster updates to the online weights, where we can assume that the trusted weights are stationary at some value $\boldsymbol{\theta}$. Then, using $P_t \doteq \boldsymbol{\phi}_t^\top \boldsymbol{\theta}$, the targets for the updates of the forward view are

$$Z_t^t(\boldsymbol{\theta}) = \boldsymbol{\phi}_t^\top \boldsymbol{\theta}\,,$$
$$Z_t^h(\boldsymbol{\theta}) = X_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1})\boldsymbol{\phi}_t^\top \boldsymbol{\theta} + \gamma_{t+1}\lambda_{t+1} Z_{t+1}^h(\boldsymbol{\theta})\,,$$

where we have extended the notation slightly to make the dependence of $Z_t^h(\boldsymbol{\theta})$ on $\boldsymbol{\theta}$ explicit. Notice that the residual predictions on each time step depend on the same stationary trusted weights $\boldsymbol{\theta}$. Because of the assumed stationarity of $\boldsymbol{\theta}$, the updates to the weights $\tilde{\boldsymbol{\theta}}_t^h$ of the forward view can again be considered standard stochastic gradient updates and therefore these weights converge towards the fixed point $\tilde{\boldsymbol{\theta}}_*(\boldsymbol{\theta})$, where again we make the dependence on $\boldsymbol{\theta}$ explicit, defined by

$$\tilde{\boldsymbol{\theta}}_*(\boldsymbol{\theta}) = \mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top\big]^{-1}\, \mathbb{E}[\boldsymbol{\phi}_t Z_t^\infty(\boldsymbol{\theta})]\,,$$

where $Z_t^\infty(\boldsymbol{\theta}) = \lim_{h\to\infty} Z_t^h(\boldsymbol{\theta})$ denotes the limit of the target of the update as the horizon grows to infinity. In an episodic setting, $Z_t^\infty = Z_t^T$ for all $t < T$, where T denotes the first termination after time k. More in general, Z_t^∞ is always well-defined because we require $\prod_{t=0}^\infty \gamma_t = 0$.

For the analysis of the slower updates to $\boldsymbol{\theta}_t$, we can now assume the faster time scale has already converged to its fixed point $\tilde{\boldsymbol{\theta}}_*(\boldsymbol{\theta}_t)$ for the current weights. Therefore, we analyze the update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta_{t+1}(\tilde{\boldsymbol{\theta}}_*(\boldsymbol{\theta}_t) - \boldsymbol{\theta}_t)$$
$$= \boldsymbol{\theta}_t + \beta_{t+1}(\mathbb{E}\big[\boldsymbol{\phi}_k \boldsymbol{\phi}_k^\top\big]^{-1}\, \mathbb{E}[\boldsymbol{\phi}_k Z_k^\infty(\boldsymbol{\theta}_t)] - \boldsymbol{\theta}_t)\,.$$

This is a stochastic-approximation update that, under the conditions that $\sum_{t=0}^{\infty} \beta_t = \infty$ and $\sum_{t=0}^{\infty} \beta_t^2 < \infty$, converges almost surely to the fixed point $\boldsymbol{\theta}_*$ that satisfies

$$\boldsymbol{\theta}_* = \mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top\big]^{-1} \mathbb{E}[\boldsymbol{\phi}_t Z_t^\infty(\boldsymbol{\theta}_*)] \ .$$

If we multiply both sides with $\mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top\big]$, this implies that $\mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_*\big] = \mathbb{E}[\boldsymbol{\phi}_t Z_t^\infty(\boldsymbol{\theta}_*)]$ and therefore, by moving both terms to the same side and then multiplying with $\mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top\big]^{-1}$,

$$\mathbb{E}\big[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top\big]^{-1} \mathbb{E}\big[\boldsymbol{\phi}_t (Z_t^\infty(\boldsymbol{\theta}_*) - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_*)\big] = \mathbf{0} \ .$$

It follows immediately that $\boldsymbol{\theta}_*$ minimizes the mean-squared projected Bellman error completely to zero, as desired. $\qquad\square$

10   Discussion

In this paper, we have considered how to answer predictive questions with algorithms that use constant computation per time step that is proportional to the number of learned weights, and that is independent of the span of the prediction. We considered both final and cumulative outcomes, under online and offline updating, with and without persistency of the residual predictions we encounter during an episode, and with hard and soft termination. In the end, we obtained a single general algorithm that can be used for all these different predictive questions, which is shown in (44). This algorithm is guaranteed to be convergent under typical, fairly mild, technical conditions.

Some extensions remain for future work. In particular, we have not considered how different policies of behavior can influence our predictions, and as a result have not talked about the problem of control in which the goal is to find the optimal policy for a given (reward) signal. Our analysis already extends naturally to the prediction of action values, from which control policies can be easily distilled. Then, using a form of policy iteration (Bellman 1957; Howard 1960), we can repeatedly switch between estimating and improving the policy to tackle the problem of optimal control. However, to properly and fully include adaptable policies, we would in addition need to carefully consider the problem of learning off-policy, about action-selection policies that differ from the one used to generate the data (Sutton and Barto 1998). This is consistent but orthogonal to the ideas outlined in this paper, and such off-policy predictions (including those about the greedy and, ultimately, optimal policy) are learnable through a proper use of rejection sampling, as in Q-learning, or importance sampling (Precup, Sutton, and Singh 2000; Precup and Sutton 2001; Maei 2011; Sutton et al. 2014; van Hasselt, Mahmood, and Sutton 2014; Mahmood, van Hasselt, and Sutton 2014).

All algorithms considered in this paper are in a sense descendent from a linear stochastic gradient, or LMS, update. The main idea of span-independent computation is more general and can be applied quite naturally to other settings, including for instance non-linear functions such as deep neural networks (LeCun, Bengio, and Hinton 2015; Mnih et al. 2015) or to quadratic-time linear-function algorithms as in LSTD (Bradtke and Barto 1996). Not all updates may have fully equivalent span-independent counterparts, but even then it may be more important to be independent of span than to be exactly equivalent.

References

Bach, F. and Moulines, E. (2013). Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. *Advances in Neural Information Processing Systems* 26, pp. 773–781.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems & Control Letters* 29(5), pp. 291–294.

Borkar, V. S. (2008). Stochastic approximation. *Cambridge Books*.

Bradtke, S. J. and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning* 22, pp. 33–57.

Howard, R. A. (1960). *Dynamic programming and Markov processes*. MIT Press.

Konda, V. R. and Tsitsiklis, J. N. (2004). Convergence rate of linear two-time-scale stochastic approximation. *Annals of applied probability* 14(2), pp. 796–819.

Kushner, H. J. and Yin, G. (2003). *Stochastic approximation and recursive algorithms and applications*. Vol. 35. Springer Science & Business Media.

LeCun, Y., Bengio, Y., and Hinton, G. (May 2015). Deep learning. *Nature* 521(7553), pp. 436–444.

Maei, H. R. (2011). Gradient temporal-difference learning algorithms. PhD thesis. University of Alberta.

Mahmood, A. R., van Hasselt, H. P., and Sutton, R. S. (2014). Weighted importance sampling for off-policy learning with linear function approximation. *Advances in Neural Information Processing Systems* 27, pp. 3014–3022.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature* 518(7540), pp. 529–533.

Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In: *IEEE International Conference on Robotics and Automation, 2003*. Vol. 2. IEEE, pp. 1985–1991.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization* 30(4), pp. 838–855.

Precup, D. and Sutton, R. S. (2001). Off-policy temporal-difference learning with function approximation. In: *Proceedings of the eighteenth International Conference on Machine Learning*. Morgan Kaufmann, pp. 417–424.

Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility traces for off-policy policy evaluation. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, pp. 766–773.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics* 22(3), pp. 400–407.

Sutton, R. S. (1984). Temporal credit assignment in reinforcement learning. PhD thesis. University of Massachusetts.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning* 3, pp. 9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* The MIT press, Cambridge MA.

Sutton, R. S., Szepesvári, Cs., and Maei, H. R. (2008). A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems* 21, pp. 1609–1616.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ACM, pp. 993–1000.

Sutton, R. S., Mahmood, A. R., Precup, D., and van Hasselt, H. P. (2014). A new Q(λ) with interim forward view and Monte Carlo equivalence. *JMLR W&CP* 32(1), pp. 568–576.

Szepesvári, Cs. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 4(1), pp. 1–103.

van Hasselt, H. P., Mahmood, A. R., and Sutton, R. S. (2014). Off-policy TD(λ) with a true online equivalence. In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence.* AUAI Press.

van Seijen, H. and Sutton, R. S. (2014). True online TD(λ). *JMLR W&CP* 32(1), pp. 692–700.