Individual Sequence Prediction using Memory-efficient Context Trees

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer

***Abstract*— Context trees are a popular and effective tool for tasks such as compression, sequential prediction, and language modeling. We present an algebraic perspective of context trees for the task of individual sequence prediction. Our approach stems from a generalization of the notion of margin used for linear predictors. By exporting the concept of margin to context trees, we are able to cast the individual sequence prediction problem as the task of finding a linear separator in a Hilbert space, and to apply techniques from machine learning and online optimization to this problem. Our main contribution is a memory efficient adaptation of the Perceptron algorithm for individual sequence prediction. We name our algorithm the *Shallow Perceptron* and prove a *shifting* mistake bound, which relates its performance with the performance of any sequence of context trees. We also prove that the Shallow Perceptron grows a context tree at a rate that is upper-bounded by its mistake-rate, which imposes an upper-bound on the size of the trees grown by our algorithm.**

***Index Terms*— context trees, online learning, Perceptron, shifting bounds**

I. Introduction

Universal prediction of individual sequences is concerned with the task of observing a sequence of symbols one-by-one and predicting the identity of each symbol before it is revealed. In this setting, no assumptions regarding the underlying process that generates the sequence are made. In particular, we do not assume that the sequence is generated by a stochastic process. Over the years, individual sequence prediction has received much attention from game theorists [1]–[3], information theorists [4]–[7], and machine learning researchers [8]–[10]. *Context trees* are a popular type of sequence predictors. Context trees are unique in that the number of previous symbols they use to make each prediction is context dependent, rather than being a constant. In this paper, we exploit insights from machine learning and online optimization to present an algebraic perspective on context tree learning and individual sequence prediction. Our alternative approach becomes possible after we cast the sequence prediction problem as a linear separation problem in a Hilbert space.

The investigation of individual sequence prediction began with a series of influential papers by Robbins, Blackwell and Hannan [1]–[3]. This line of work revolved around the *compound sequential Bayes* predictor, a randomized prediction algorithm that is guaranteed to perform asymptotically as well as the best constant prediction. Cover and Shenhar [5] extended this result and gave an algorithm that is guaranteed to perform asymptotically as well as the best k'th order Markov predictor, where k is a parameter of the algorithm. Feder et al. [6] presented a similar algorithm[1] with a similar theoretical guarantee, and a faster convergence rate than earlier work. We call these algorithms *fixed order predictors*, to emphasize their strong dependence on the prior knowledge of the order k of the Markov predictor.

The fixed order assumption is undesirable. The number of previous symbols needed to make an accurate prediction is usually not constant, but rather depends on the identity of the recently observed symbols. For example, assume that the sequence we are trying to predict is a text in the English language and assume that the last observed symbol is the letter "q". In English, the letter "q" is almost always followed by the letter "u", and we can confidently predict the next symbol in the sequence without looking farther back. However, a single symbol does not suffice in general in order to make accurate predictions and additional previous symbols are required. This simple example emphasizes that the number of previous symbols needed to make an accurate prediction

A preliminary version of this paper appeared at Advances in Neural Information Processing Systems 17 under the title "The Power of Selective Memory: Self-Bounded Learning of Prediction Suffix Trees"

O. Dekel is with Microsoft Research.

S. Shalev-Shwartz is with the Department of Computer Science and Engineering, The Hebrew University.

Y. Singer is with Google Research.

Part of this work was supported by the Israeli Science Foundation grant number 522-04 while all three authors resided at the Hebrew University.

Manuscript received January 24 2008; revised September 30 2008.

[1]We refer to the predictor described in Eq. (43) of [6], and not to the IP predictor presented later in the same paper.

depends on the identity of those symbols. Even setting a global upper-bound on the maximal number of previous symbols needed to make a prediction may be a difficult task. Optimally, the prediction algorithm should be given the freedom to look as far back as needed to make accurate predictions.

Feder et al. [6] realized this and presented the *incremental parsing* predictor (IP), an adaptation of the Lempel-Ziv compression algorithm [11] that incrementally constructs a context tree predictor. The *context* of each prediction is defined as the suffix of the observed sequence used to predict the next symbol in the sequence. A context tree is the means for encoding the context length required to make a prediction, given the identity of the recently observed symbols. More precisely, the sequence of observed symbols is read backwards, and after reading each symbol the context tree tells us whether we have seen enough to make a confident prediction or whether we must read off an additional symbol and increase the context length by one. The desire to lift the fixed order assumption also influenced the design of the *context tree weighting* algorithm (CTW) [7], [12], [13] and various related statistical algorithms for learning variable length Markov models [14]–[16]. These works, however focused on probabilistic models with accompanying analyses, which centered on likelihood-based regret and generalization bounds.

We now give a more formal definition of context trees in a form that is convenient for our presentation. For simplicity, we assume that the alphabet of the observed symbols is $\Sigma = \{-1, +1\}$. We discuss relaxations to non-binary alphabets in Sec. VI. Let $\Sigma^\star$ denote the set of all finite-length sequences over the alphabet Σ. Specifically, $\Sigma^\star$ includes ϵ, the empty sequence. We say that a set $V \subset \Sigma^\star$ is *suffix-closed* if for every $\mathbf{s} \in V$, every suffix of $\mathbf{s}$, including the empty suffix ϵ, is also contained in V. A context tree is a function $\mathcal{T} : V \to \Sigma$, where V is a suffix-closed subset of $\Sigma^\star$. Let $x_1, \ldots, x_{t-1}$ be the symbols observed until time $t-1$. We use $\mathbf{x}_j^k$ to denote the subsequence $x_j, \ldots, x_k$, and for completeness, we adopt the convention $\mathbf{x}_t^{t-1} = \epsilon$. We denote the set of all suffixes of $\mathbf{x}_1^k$ by, $\mathrm{suf}\big(\mathbf{x}_1^k\big)$, thus $\mathrm{suf}\big(\mathbf{x}_1^k\big) = \big\{\mathbf{x}_j^k \,\big|\, 1 \le j \le k+1\big\}$. A context tree predicts the next symbol in the sequence to be $\mathcal{T}(\mathbf{x}_{t-i}^{t-1})$, where $\mathbf{x}_{t-i}^{t-1}$ is the longest suffix of $\mathbf{x}_1^{t-1}$ contained in V.

Our goal is to design online algorithms that incrementally construct context tree predictors, while predicting the symbols of an input sequence. An algorithm of this type maintains a context tree predictor in its internal memory. The algorithm starts with a default context tree. After each symbol is observed, the algorithm has the option to modify its context tree, with the explicit goal of improving the accuracy of its predictions in the future.

A context tree $\mathcal{T}$ can be viewed as a rooted tree. Each node in the tree represents one of the sequences in V. Specifically, the root of the tree represents the empty sequence ϵ. The node that represents the sequence $\mathbf{s}_1^k = s_1, \ldots, s_k$ is the child of the node representing the sequence $\mathbf{s}_2^k$. When read backwards, the sequence of observed symbols defines a path from the root of the tree to one of its nodes. Since the tree is not required to be complete, this path can either terminate at an inner node or at a leaf. Each of the nodes along this path represents a suffix of the observed sequence. The node at the end of the path represents the longest suffix of the observed sequence that is a member of V. This suffix is the context used to predict the next symbol in the sequence, and the function $\mathcal{T}$ maps this suffix to the predicted symbol. This process is illustrated in Fig. 1.

Our presentation roughly follows the evolutionary progress of existing individual sequence prediction algorithms. Namely, we begin our presentation with the assumption that the structure of the context tree is known a-priori. This assumption is analogous to the fixed order assumption mentioned above. Under this assumption, we show that the individual sequence prediction problem can be solved through a simple embedding into an appropriate Hilbert space followed with an application of techniques from online learning and convex programming. We next lift the assumption that the context tree structure is fixed ahead of time, and present algorithms that incrementally learn the tree structure with no predefined limit on the maximal tree depth. The construction of these algorithms requires us to define a more sophisticated embedding of the sequence prediction problem into a Hilbert space, but applies the same algorithms on the embedded problem. A major drawback of this approach is that it may grow very large context trees, even when it is unnecessary. The space required to store the context tree grows with the length of the input sequence, and this may pose serious computational problems when predicting very long sequences. Interestingly, this problem also applies to the IP predictor of Feder et al. [6] and to the unbounded-depth version of the CTW algorithm [12]. The most important contribution of this paper is our final algorithm, which overcomes the memory inefficiency problem underscored above.

We evaluate the performance of our algorithms using the game-theoretic notion of *regret*. Formally, let $\mathcal{C}$ be a comparison class of predictors. For example, $\mathcal{C}$ can be the set of all context tree predictors of depth k. Had the

input sequence $\mathbf{x}_1^T$ been known up-front, one could have chosen the best predictor from $\mathcal{C}$, namely, the predictor that makes the least number of mistakes on $\mathbf{x}_1^T$. The regret of an online prediction algorithm with respect to the class $\mathcal{C}$ is the difference between the average number of prediction mistakes made by the algorithm and the average number of prediction mistakes made by the best predictor in $\mathcal{C}$.

Cover [17] showed that any *deterministic* online predictor cannot attain a vanishing regret universally for all sequences. One way to circumvent this difficulty is to allow the online predictor to make *randomized* predictions and to analyze its expected regret. This approach was taken, for example, in the analysis of the IP predictor [6].

Another way to avoid the difficulty observed by Cover is to slightly modify the regret-based model in which we analyze our algorithm. A common approach in learning theory is to associate a confidence value with each prediction, and to use the *hinge-loss* function to evaluate the performance of the algorithm, instead of simply counting errors. Before giving a formal definition of these terms, we first need to generalize our previous definitions and establish the important concept of a *margin-based* context tree. A margin-based context tree is a context tree with real-valued outputs. In other words, it is a function $\tau : V \to \mathbb{R}$, where as before, V is a suffix-closed subset of $\Sigma^\star$. If $\mathbf{x}_{t-i}^{t-1}$ is the longest suffix of $\mathbf{x}_1^{t-1}$ contained in V, then $\mathrm{sign}(\tau(\mathbf{x}_{t-i}^{t-1}))$ is the binary prediction of the next symbol and $|\tau(\mathbf{x}_{t-i}^{t-1})|$ is called the *margin* of the prediction. The margin of a prediction should be thought of as a degree of confidence in that prediction.

The hinge-loss attained by a context tree τ, when it attempts to predict the last symbol in the sequence $\mathbf{x}_1^t$, is defined as

$$\ell(\tau, \mathbf{x}_1^t) \;=\; \left[1 - x_t\, \tau(\mathbf{x}_{t-i}^{t-1})\right]_+ \quad , \tag{1}$$

where $[a]_+ = \max\{0, a\}$. Note that $\ell(\tau, \mathbf{x}_1^{t-1})$ is zero iff $x_t = \mathrm{sign}(\tau(\mathbf{x}_{t-i}^{t-1}))$ and $|\tau(\mathbf{x}_{t-i}^{t-1})| \geq 1$. The hinge-loss is a convex upper bound on the indicator of a prediction mistake. When considering deterministic individual sequence predictors in later sections, we bound the number of prediction mistakes made by our algorithm using the cumulative hinge-loss suffered by any competing predictor from $\mathcal{C}$. Interestingly, the two techniques discussed above, randomization and using the notions of margin and hinge-loss, are closely related. We discuss this relation in some detail in Sec. III.

A margin-based context tree can be converted back into a standard context tree simply by defining $\mathcal{T}(\mathbf{s}) = \mathrm{sign}(\tau(\mathbf{s}))$ for all $\mathbf{s} \in V$. Clearly, $\mathcal{T}$ and τ are entirely equivalent predictors in terms of the symbols they predict, and any bound on the number of mistakes made by τ applies automatically to $\mathcal{T}$. Therefore, from this point on, we put aside the standard view of context trees and focus entirely on margin-based context trees. The advantage of the margin-based approach is that it enables us to cast the context tree learning problem as the problem of linear separation in a Hilbert space. Linear separation is a popular topic in machine learning, and we can harness powerful machine learning tools to our purposes. Specifically, we use the Perceptron algorithm [18]–[20] as well as some of its variants. We also use a general technique for online convex programming presented in [21], [22].

Main Results

We now present an overview of the main contributions of this paper, and discuss how these contributions relate to previous work on the topic. We begin with Sec. II in which we make the simplifying assumption that the set V is finite, fixed, and known in advance to the algorithm. Under this strong assumption, a simple application of the Perceptron algorithm to our problem results in a deterministic individual sequence predictor with various desirable qualities. Following [23], we generalize Novikoff's classic mistake bound for the Perceptron algorithm [20] and prove a bound that compares the performance of the Perceptron with the performance of a competitor that is allowed to change with time. In particular, let $\tau_1^\star, \ldots, \tau_T^\star$ be an arbitrary sequence of context-tree predictors from $\mathcal{C}$. We denote by $L^\star$ the cumulative hinge-loss suffered by this sequence of predictors on the symbol sequence $x_1, \ldots, x_T$, namely, $L^\star = \sum_{t=1}^T \ell(\tau_t^\star, \mathbf{x}_1^t)$. Our goal is to bound the number of prediction mistakes made by our algorithm in terms of $L^\star$. Also define

$$\begin{aligned} S \;&=\; \sum_{t=2}^{T} \sqrt{\sum_{\mathbf{s}\in V} \left(\tau_t^\star(\mathbf{s}) - \tau_{t-1}^\star(\mathbf{s})\right)^2} \qquad \text{and} \\ U \;&=\; \max_t \sqrt{\sum_{\mathbf{s}\in V} \left(\tau_t^\star(\mathbf{s})\right)^2} \quad . \end{aligned} \tag{2}$$

The variable S represents the amount by which the sequence $\tau_1^\star, \ldots, \tau_T^\star$ changes over time and U is the maximal norm over the context-trees in this sequence. Letting $\hat{x}_1, \ldots, \hat{x}_T$ denote the sequence of predictions made by our algorithm, we prove the following mistake bound

$$\frac{|\{t : \hat{x}_t \neq x_t\}| - L^\star}{T} \;\leq\; \frac{(S+U)\sqrt{L^\star} + (S+U)^2}{T} \quad . \tag{3}$$

A bound of this type is often called a *shifting* or a *drifting* bound, as it permits the predictor we are competing against to change with time. When the performance of our bound is compared to a fixed context tree predictor, S in Eq. (3) simply becomes zero.

The assumption of a fixed Markov order k is equivalent to assuming that V contains all possible sequences of length at most k, and is therefore a special case of our setting. Under certain additional constraints, we can compare the bound in Eq. (3) with existing bounds for sequential prediction. In this setting, the expected regret of the fixed order predictor of Feder et al. [6] with respect to the class of all fixed k-order context tree predictors is $O(\sqrt{2^k/T})$. The expected regret of Cover and Shenhar's fixed order predictor [5] is $O(2^k/\sqrt{T})$. A meaningful comparison between these bounds and Eq. (3) can be made in the special case where the sequence is *realizable*, namely, when there exists some $\tau^\star \in \mathcal{C}$ such that $\ell(\tau^\star, \mathbf{x}_1^t) = 0$ for all t. In this case, Eq. (3) reduces to the bound,

$$\frac{|\{t : \hat{x}_t \neq x_t\}|}{T} \leq \frac{2^k}{T} \quad . \tag{4}$$

This regret bound approaches zero much faster than the bounds of Feder et al. [6] and Cover and Shenhar [5]. Other advantages of our bound are the fact that we compete against sequences of context trees that may change with time, and the fact that our bound does not hold only in expectation.

In Sec. III we make an important digression in our development of a memory efficient context tree algorithm in order to show how the fixed order predictor of [6] can be recaptured and derived directly using our approach. Concretely, we cast the individual sequence prediction problem as an online convex program and solve it using an online convex programming procedure described in [21], [22]. Interestingly, the resulting algorithm turns out to be precisely the fixed order predictor proposed in [6], and the $O(\sqrt{2^k/T})$ expected regret bound proven by [6] follows immediately from the convergence analysis of the online convex programming scheme we present.

In Sec. IV we return to the main topic of this paper and relax the assumption that V is known in advance. By embedding the sequence prediction problem into a Hilbert space, we are able to learn context tree predictors of an arbitrary depth. As in the previous sections, we still rely on the standard Perceptron algorithm as the bound in Eq. (3) still holds. Once again, considering the realizable case, where the sequence is generated by a k-order context tree, the regret of our algorithm is $O(2^{2k}/T)$, while the expected regret of Feder et al.'s IP predictor is $O(k/\log(T) + 1/\sqrt{\log(T)})$. Similar to the IP predictor and other unbounded-depth context tree growing algorithms, our algorithm grows context trees that may become excessively large.

In Sec. V we overcome the memory inefficiency problem mentioned above and present the main contribution of this paper, which is the *Shallow Perceptron* algorithm for online learning of context trees. Our approach balances the two opposing requirements presented above. On one hand, we do not rely on any a-priori assumptions on V, and permit the context tree to grow as needed to make accurate predictions. On the other hand, we only use a short context length when it is sufficient to make accurate predictions. More precisely, the context trees constructed by the shallow Perceptron grow only when prediction mistakes are made, and the total number of nodes in the tree is always upper-bounded by the number of prediction mistakes made so far. We again prove a shifting mistake bound similar to the bound in Eq. (3). In the case of the Shallow Perceptron, the mistake bound implicitly bounds the number of nodes in the context tree. All of our bounds are independent of the length of the sequence, a property which makes our approach suitable for predicting arbitrarily long sequences.

II. Margin-Based Context Trees and Linear Separation

In this section we cast the context tree learning problem as the problem of finding a separating hyperplane in a Hilbert space. As described above, a context tree over the alphabet $\{-1, +1\}$ is a mapping from V to the set $\{-1, +1\}$, where V is a suffix-closed subset of $\{-1, +1\}^\star$. Throughout this section we make the (rather strong) assumption that V is finite, fixed, and known to the algorithm. This assumption is undesirable and we indeed lift it in the next sections. However, as outlined in Sec. I, it enables us to present our algebraic view of context trees in its simplest form.

Let $\mathcal{H}$ be a Hilbert space of functions from V into $\mathbb{R}$, endowed with the inner product

$$\langle \nu, \mu \rangle = \sum_{\mathbf{s} \in V} \nu(\mathbf{s}) \mu(\mathbf{s}) \tag{5}$$

and the induced norm $\|\mu\| = \sqrt{\langle \mu, \mu \rangle}$. The Hilbert space $\mathcal{H}$ is isomorphic to the $|V|$-dimensional vector space, $\mathbb{R}^{|V|}$, whose elements are indexed by sequences or strings from V. We use the more general notion of a Hilbert space since we later lift the assumption that V is fixed and known in advance, and it may become impossible to bound $V|$ by a constant.

A margin-based context tree is a vector in $\mathcal{H}$ by definition. The input sequence can also be embedded in

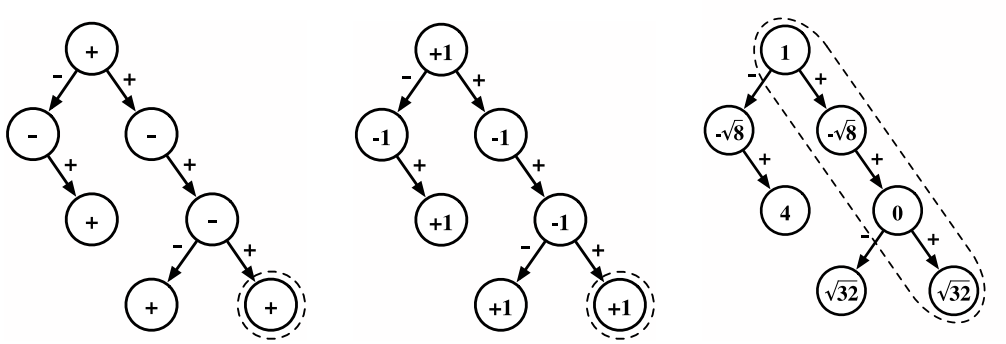

Fig. 1. An example of a context tree (left), along with its equivalent margin-based context tree (center), and an equivalent context function (right). The context associated with each node is indicated on the edges of the tree along the path from the root to that node. The output associated with each node is provided inside the node. The nodes and values that constitute the prediction for the input sequence $(+ - + + +)$ are designated with a dashed line.

$\mathcal{H}$ as follows. Let k be any positive integer and let $\mathbf{x}_1^k$ be any sequence of k symbols from Σ. We map $\mathbf{x}_1^k$ to a function $\phi \in \mathcal{H}$ as follows,

$$\phi(\mathbf{s}_1^i) = \begin{cases} 1 & \text{if } \mathbf{s}_1^i \text{ is longest suffix of } \mathbf{x}_1^k \text{ s.t. } \mathbf{s}_1^i \in V \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Returning to our sequence prediction problem, let $\mathbf{x}_1^{t\text{-}1}$ be the sequence of observed symbols on round t, and let ϕ_t be its corresponding vector in $\mathcal{H}$. Furthermore, let $\mathbf{x}_{t-i}^{t\text{-}1}$ denote the longest suffix of $\mathbf{x}_1^{t\text{-}1}$ contained in V. Then for any margin-based context tree $\tau \in \mathbb{R}^{|V|}$ we have that

$$\tau(\mathbf{x}_{t\text{-}i}^{t\text{-}1}) \;=\; \langle \phi_t, \tau \rangle \quad . \tag{7}$$

Geometrically, τ_t can be viewed as the normal of a separating hyperplane in $\mathcal{H}$. We predict that the next symbol in the sequence is $+1$ if the vector ϕ_t falls in the positive half-space defined by τ_t, that is, if $\langle \phi_t, \tau_t \rangle \geq 0$. Otherwise, we predict that the next symbol in the sequence is -1.

Embedding margin-based context trees in $\mathcal{H}$ also provides us with a natural measure of tree complexity. We define the complexity of the margin-based context tree to be the squared-norm of the vector τ, namely

$$\|\tau\|^2 \;=\; \sum_{\mathbf{s} \in V} \tau^2(s) \quad .$$

Next, we use the equivalence of context trees and linear separators to devise a context tree learning algorithm based on the Perceptron algorithm [18]–[20]. The Perceptron, originally formulated for the task of binary classification, observes a sequence of inputs and predicts a binary outcome for each input. Before any symbols are revealed, the Perceptron sets the initial margin-based context tree, τ_1, to be the zero vector in $\mathcal{H}$. On round t, the Perceptron is given the input ϕ_t, as defined in Eq. (6), and predicts the identity of the next symbol in the sequence to be the sign of $\langle \tau_t, \phi_t \rangle$, where τ_t is the margin-based context tree it currently holds in memory. Immediately after making this prediction, the next symbol in the sequence, x_t, is revealed and the Perceptron constructs τ_{t+1}. The Perceptron applies a conservative update rule, which means that if x_t is correctly predicted, then the next context tree τ_{t+1} is simply set to be equal to τ_t. However, if a prediction mistake is made, the Perceptron sets

$$\tau_{t+1} \;=\; \tau_t + x_t \phi_t \quad .$$

While we focus in this section on vector-based representations, it is worth describing the resulting update in its functional form. Viewing τ_{t+1} as a function, from V to $\mathbb{R}$, the updates described above amounts to,

$$\tau_{t+1}(\mathbf{s}) \;=\; \begin{cases} \tau_t(\mathbf{s}) + x_t & \text{if } \; \mathbf{s} = \mathbf{x}_{t\text{-}i}^{t\text{-}1} \\ \tau_t(\mathbf{s}) & \text{otherwise} \end{cases} \quad ,$$

where $\mathbf{x}_{t\text{-}i}^{t\text{-}1}$ is the longest suffix of $\mathbf{x}_1^{t\text{-}1}$ contained in V. This update implies that only a *single* coordinate of τ_t is modified as the Perceptron constructs τ_{t+1}.

We now state and prove a mistake bound for the Perceptron algorithm. This analysis not only provides a bound on the number of sequence symbols that our algorithm predicts incorrectly, but also serves as an important preface to the analysis of the algorithms presented in the next sections. The primary tool used in our analysis is encapsulated in the following general lemma, which holds for any application of the Perceptron algorithm and is not specific to the case of context tree learning. It is

a generalization of Novikoff's classic mistake bound for the Perceptron algorithm [20]. This lemma can also be derived from the analyses presented in [23], [24].

Lemma 1 *Let $\mathcal{H}$ be a Hilbert space and Let $\{(\phi_t, x_t)\}_{t=1}^T$ be a sequence of input-output pairs, where $\phi_t \in \mathcal{H}$, $\|\phi_t\| \leq R$, and $x_t \in \{-1,+1\}$ for all $1 \leq t \leq T$. Let $u_1^\star, \ldots, u_T^\star$ be a sequence of arbitrary functions in $\mathcal{H}$. Define $\ell_t^\star = [1 - x_t \langle u_t^\star, \phi_t \rangle]_+$, $L^\star = \sum_{t=1}^T \ell_t^\star$, $S = \sum_{t=2}^T \|u_t^\star - u_{t-1}^\star\|$, and $U = \max_t \|u_t^\star\|$. Let M denote the number of prediction mistakes made by the Perceptron algorithm when it is presented with $\{(\phi_t, x_t)\}_{t=1}^T$. Then,*

$$M - \sqrt{M}\, R(U+S) \;\leq\; L^\star \quad .$$

The proof of the lemma is given in the appendix.

Applying Lemma 1 in our setting is a straightforward matter due to the equivalence of context tree learning and linear separation. Note that the construction of ϕ_t as described in Eq. (6) implies that $\|\phi_t\| = 1$, thus we can set $R = 1$ in the lemma above and get that the number of mistakes, M, made by the Perceptron algorithm satisfies $M - \sqrt{M}(U+S) \;\leq\; L^\star$. The latter inequality is a quadratic equation in $\sqrt{M}$. Solving this inequality for M (see Lemma 8 in the appendix) yields the following corollary.

Corollary 2 *Let $x_1, x_2, \ldots, x_T$ be a sequence of binary symbols. Let $\tau_1^\star, \ldots, \tau_T^\star$ be a sequence of arbitrary margin-based trees. Define $\ell_t^\star = \ell(\tau_t^\star, \mathbf{x}_1^t)$, $L^\star = \sum_{t=1}^T \ell_t^\star$, $S = \sum_{t=2}^T \|\tau_t^\star - \tau_{t-1}^\star\|$ and $U = \max_t \|\tau_t^\star\|$. Let M denote the number of prediction mistakes made by the Perceptron algorithm when it is presented with the sequence of binary symbols. Then,*

$$M \;\leq\; L^\star + (S+U)^2 + (S+U)\sqrt{L^\star} \quad .$$

Note that if $\tau_t^\star = \tau^\star$ for all t, then S equals zero and we are essentially comparing the performance of the Perceptron to a single and fixed margin-based context tree. In this case, Thm. 2 reduces to a bound due to Gentile [24]. If $L^\star$ also equals zero then Thm. 2 reduces to Novikoff's original analysis of the Perceptron [20]. In the latter case, it is sufficient to set $\tau^\star(\mathbf{s})$ to either 1 or -1 in order to achieve a hinge-loss of zero on each round. We can thus simply bound $\|\tau^\star\|^2$ by $|V|$ and the bound in Thm. 2 reduces to

$$\frac{M}{T} \;\leq\; \frac{|V|}{T} \quad .$$

As mentioned in Sec. I, this bound approaches zero much faster than the bounds of Feder et al. [6] and of Cover and Shenhar [5].

III. Randomized Predictions and the Hinge-Loss

In the previous section we reduced the individual sequence prediction problem to the task of finding a separating hyperplane in a Hilbert space. This perspective enabled us to use the Perceptron algorithm for sequence prediction and to bound the number of prediction mistakes in terms of the cumulative hinge-loss of any sequence of margin-based context trees. An alternative approach, taken for instance in [6], [17], is to derive an algorithm that makes *randomized* predictions and to prove a bound on the *expected* number of prediction mistakes. In this section, we describe an interesting relation between these two techniques. This relation enables us to derive the first algorithm presented in [6] directly from our setting.

Assume that we have already observed the sequence $\mathbf{x}_1^{t-1}$ and that we are attempting to predict the next symbol x_t. Let τ_t be a margin-based context tree whose output is restricted to the interval $[-1,+1]$. We use τ_t to make the randomized prediction $\hat{x}_t$, where

$$\forall\; a \in \{+1,-1\} \;:\; \mathbb{P}(\hat{x}_t = a) \;=\; \frac{1 + a\tau_t(\mathbf{x}_{t-i}^{t-1})}{2} \quad . \tag{8}$$

It is easy to verify that

$$\begin{aligned} \mathbb{E}[\hat{x}_t \neq x_t] \;&=\; \frac{1 - x_t \tau_t(\mathbf{x}_{t-i}^{t-1})}{2} \\ &=\; \frac{[1 - x_t \tau_t(\mathbf{x}_{t-i}^{t-1})]_+}{2} \\ &=\; \frac{\ell(\tau_t, \mathbf{x}_1^t)}{2} \quad . \end{aligned} \tag{9}$$

In words, the cumulative hinge-loss of a deterministic margin-based context tree τ_t translates into the expected number of prediction mistakes made by an analogous randomized prediction rule.

When making randomized predictions, our goal is to attain a small *expected regret*. Specifically, let $x_1, \ldots, x_T$ be a sequence of binary symbols and let $\tau_1, \ldots, \tau_T$ be the sequence of margin-based context trees constructed by our algorithm as it observes the sequence of symbols. Assume that each of these trees is a function from V to $[-1,1]$ and let $\hat{x}_1, \ldots, \hat{x}_t$ be the random predictions made by our algorithm, where each $\hat{x}_t$ is sampled from the probability distribution defined in Eq. (8). Let $\tau^\star : V \to [-1,1]$ be a margin-based context tree, and let $x_1^\star, \ldots, x_T^\star$ be a sequence of random variables distributed according to $\mathbb{P}[x_t^\star = a] = (1 + a\tau^\star(\mathbf{x}_{t-i}^{t-1}))/2$ for $a \in \{+1,-1\}$. Assume that $\tau^\star$ is the tree that minimizes

$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[x_t^\star \neq x_t]$, and define the expected regret as

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\,\hat{x}_t \neq x_t\,] - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\,x_t^\star \neq x_t\,] \quad .$$

Using Eq. (9), the above can be equivalently written as $\frac{1}{2}$ times

$$\frac{1}{T}\sum_{t=1}^{T}\ell(\tau_t, \mathbf{x}_1^t) - \frac{1}{T}\sum_{t=1}^{T}\ell(\tau^\star, \mathbf{x}_1^t) \quad . \tag{10}$$

Our goal is to generate a sequence of margin-based context trees $\tau_1, \ldots, \tau_T$ that guarantees a small value of Eq. (10), for any input sequence of symbols. This new problem definition enables us to address the context tree learning problem within the more general framework of *online convex programming*.

Convex programming focuses on the goal of finding a vector in a given convex set that minimizes a convex objective function. In online convex programming, the objective function changes with time and the goal is to generate a sequence of vectors that minimizes the respective sequence of objective functions. More formally, online convex programming is performed in a sequence of rounds where on each round the learner chooses a vector from a convex set and the environment responds with a convex function over the set. In our case, the vector space is $\mathcal{H}$ and we define the convex subset of $\mathcal{H}$ to be $[-1,+1]^{|V|}$. Choosing a vector in $[-1,+1]^{|V|}$ is equivalent to choosing a margin-based context tree $\tau : V \rightarrow [-1,+1]$. Therefore, on round t of the online process the learner chooses a margin-based context tree τ_t. Then, the environment responds with a loss function over $[-1,+1]^{|V|}$. In our case, let us slightly overload our notation and define the loss function over $[-1,+1]^{|V|}$ to be $\ell_t(\tau) = \ell(\tau, \mathbf{x}_1^t)$. We note that given $\mathbf{x}_1^t$ the hinge-loss function is convex with respect to its first argument and thus ℓ_t is a convex function in τ over $[-1,+1]^{|V|}$.

Since we cast the context tree learning problem as an online convex programming task, we can now use a variety of online convex programming techniques. In particular, we can use the algorithmic framework for online convex programming described in [22]. For completeness, we present a special case of this framework in the Appendix. The resulting algorithm can be described in terms of two context trees. The first tree, denoted $\tilde{\tau}_t$ is a counting tree, which is defined as follows. For $t = 1$ we set, $\tilde{\tau}_1 \equiv 0$, and for $t > 1$,

$$\tilde{\tau}_{t+1}(\mathbf{s}) \;=\; \begin{cases} \tilde{\tau}_t(\mathbf{s}) + x_t & \text{if } \mathbf{s} = \mathbf{x}_{t\text{-}i}^{t\text{-}1} \\ \tilde{\tau}_t(\mathbf{s}) & \text{otherwise} \end{cases} \tag{11}$$

Note that the range of the $\tilde{\tau}_t$ is not $[-1,+1]$ and therefore it can not be used directly for defining a randomized prediction. We thus construct a second tree by scaling and thresholding $\tilde{\tau}_t$. Formally, the second tree is defined as follows: For all $\mathbf{s} \in V$,

$$\tau_t(\mathbf{s}) \;=\; \min\left\{1\,,\,\max\left\{-1\,,\,\tilde{\tau}_t(\mathbf{s})\,\sqrt{|V|/t}\,\right\}\right\} \quad . \tag{12}$$

Finally, the algorithm randomly chooses a prediction according to the distribution, $\mathbb{P}[\hat{x}_t = 1] = (1+\tau_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1}))/2$. Based on the definition of τ_t we can equivalently express the probability of predicting the symbol 1 as

$$\mathbb{P}[\hat{x}_t = 1] = \begin{cases} 1 & \text{if } \tilde{\tau}_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1}) \geq \sqrt{\frac{t}{|V|}} \\ 0 & \text{if } \tilde{\tau}_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1}) \leq -\sqrt{\frac{t}{|V|}} \\ \frac{1}{2} + \frac{\sqrt{|V|}\,\tilde{\tau}_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1})}{2\sqrt{t}} & \text{otherwise} \end{cases} \tag{13}$$

Surprisingly, the algorithm we obtain is a variant of the first algorithm given in [6]. Comparing the above algorithm with the Perceptron algorithm described in the previous section, we note two differences. First, the predictions of the Perceptron are deterministic and depend only on the sign of $\tau_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1})$ while the predictions of the above algorithm are randomized. Second, the Perceptron updates the tree only after making incorrect predictions, while the above algorithm updates the tree at the end of each round. In later sections, we rely on this conservativeness property to keep the tree size small. Nonetheless, the focus of this section is on making the connection between our framework and existing work. The following theorem provides a regret bound for the above algorithm.

Theorem 3 *Let $x_1, x_2, \ldots, x_T$ be a sequence of binary symbols. Let $\tau^\star : V \rightarrow [-1,+1]$ be an arbitrary margin based tree and let $x_1^\star, \ldots, x_T^\star$ be the randomized predictions of $\tau^\star$, namely, $\mathbb{P}[x_t^\star = a] = (1 + a\tau^\star(\mathbf{x}_{t\text{-}i}^{t\text{-}1}))/2$. Assume that an online algorithm for context trees is defined according to Eq. (13) and Eq. (11) and is presented with the sequence of symbols. Then,*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\,\hat{x}_t \neq x_t\,] - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\,x_t^\star \neq x_t\,] \;\leq\; \sqrt{\frac{|V|}{T}} \quad .$$

The proof follows from the equivalence between context function learning and online convex programming. For completeness, we sketch the proof in the Appendix.

We have shown how the randomized algorithm of [6] can be derived directly from our setting, using the online convex programming framework. Additionally, we can compare the expected regret bound proven in [6] with the bound given by Thm. 3. [6] bounds the expected regret with respect to the set of all k-order context tree

predictors by $O(\sqrt{2^k/T})$. In our setting, we set V to be the set of all binary strings of length at most k, and the bound in Thm. 3 also becomes $O(\sqrt{2^k/T})$. In other words, the generic regret bound that arises from the online convex programming framework reproduces the bound given in [6].

IV. Learning Context Trees of Arbitrary Depth

In the previous sections, we assumed that V was fixed and known in advance. We now relax this assumption and permit our algorithm to construct context trees of arbitrary depth. To this end, we must work with infinite dimensional Hilbert spaces, and thus need to redefine accordingly our embedding of the sequence prediction problem.

Let $\mathcal{H}$ be the Hilbert space of square integrable functions $f : \Sigma^\star \to \mathbb{R}$, endowed with the inner product

$$\langle g, f \rangle = \sum_{\mathbf{s} \in \Sigma^\star} g(\mathbf{s}) f(\mathbf{s}) \quad , \tag{14}$$

and the induced norm $\|g\| = \sqrt{\langle g, g \rangle}$. Note that the sole difference between the inner products defined in Eq. (14) and in Eq. (5) is in the support of f, which is extended in Eq. (14) to $\Sigma^\star$.

To show how the context tree learning problem can be embedded in $\mathcal{H}$, we map both symbol-sequences and context trees to functions in $\mathcal{H}$. Our construction relies on a predefined *decay parameter* $\alpha > 0$. Let $x_1, \ldots, x_k$ be any sequence of symbols from Σ. We map this sequence to the function $f \in \mathcal{H}$, defined as follows,

$$f(\mathbf{s}_1^i) = \begin{cases} 1 & \text{if } \mathbf{s}_1^i = \epsilon \\ e^{-\alpha\, i} & \text{if } \mathbf{s}_1^i \in \mathrm{suf}\left(\mathbf{x}_1^k\right) \\ 0 & \text{otherwise} \end{cases} \quad , \tag{15}$$

where, as before, $\mathrm{suf}\left(\mathbf{x}_1^k\right)$ denotes the set of all suffixes of $\mathbf{x}_1^k$. The decay parameter α mitigates the effect of long contexts on the function f_t. This idea reflects the assumption that statistical correlations tend to decrease as the time between events increases, and is common to many context tree learning approaches [8], [9], [13]. Comparing the definition of ϕ from Eq. (6) and the above definition of f we note that in the former only a single element of ϕ is non-zero while in the latter all suffixes of $x_1, \ldots, x_k$ are mapped to non-zero values.

Next, we turn to the task of embedding margin-based context trees in $\mathcal{H}$. Let $\tau : V \to \mathbb{R}$ be a margin-based context tree. We map τ to the function $g \in \mathcal{H}$, defined by

$$g(\mathbf{s}_1^i) = \begin{cases} \tau(\epsilon) & \text{if } \mathbf{s}_1^i = \epsilon \\ \left(\tau(\mathbf{s}_1^i) - \tau(\mathbf{s}_2^i)\right) e^{\alpha\, i} & \text{if } \mathbf{s}_1^i \neq \epsilon \text{ and } \mathbf{s}_1^i \in V \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

We say that g is the *context function* which represents the context tree τ. Our assumption that $\mathcal{H}$ includes only squared integrable functions implicitly restricts our discussion to trees that induce a square integrable context function. We discuss the implications of this restriction at the end of this section, and note that any tree with a bounded depth induces a square integrable context function.

The mapping from a context tree τ to a context function g is a bijective mapping. Namely, every τ is represented by a unique g and vice versa. This fact is stated in the following lemma.

Lemma 4 *Let $g \in \mathcal{H}$ be a context function. Let V be the smallest suffix-closed set that contains $\{\mathbf{s} : g(\mathbf{s}) \neq 0\}$, and let $\tau : V \to \mathbb{R}$ be a margin-based context tree defined from the context function g such that for all $\mathbf{s}_1^k \in V$,*

$$\tau(\mathbf{s}_1^k) \;=\; g(\epsilon) + \sum_{i=0}^{k-1} g(\mathbf{s}_{k-i}^k) e^{-\alpha\,(i+1)} \quad .$$

Then, the mapping defined by Eq. (16) maps $\tau(\cdot)$ back to $g(\cdot)$.

The proof is deferred to the appendix.

Returning to our sequence prediction problem, let $\mathbf{x}_1^{t\text{-}1}$ be the sequence of observed symbols on round t, and let f_t be its corresponding function in $\mathcal{H}$. Also, let $\tau_t : V_t \to \mathbb{R}$ be the current context tree predictor and let g_t be its corresponding context function. Finally, let $\mathbf{x}_{t\text{-}i}^{t\text{-}1}$ denote the longest suffix of $\mathbf{x}_1^{t\text{-}1}$ contained in V_t. Then, the definition of f_t from Eq. (15) and Lemma 4 immediately imply that

$$\tau_t(\mathbf{x}_{t\text{-}i}^{t\text{-}1}) \;=\; \langle f_t, g_t \rangle \quad . \tag{17}$$

In the light of Lemma 4, the problem of learning an accurate context tree can be reduced to the problem of learning an accurate context function $g \in \mathcal{H}$. We define the complexity of τ to be the squared norm of its corresponding context function. Written explicitly, the squared norm of a context function g is

$$\|g\|^2 \;=\; \left(\sum_{\mathbf{s} \in \Sigma^\star} g^2(s) \right) \quad . \tag{18}$$

Since we assumed that $\mathcal{H}$ is square integrable, the norm of g is finite. The decay parameter α clearly affects

input: decay parameter α
initialize: $V_1 = \{\epsilon\}$, $g_1(\mathbf{s}) = 0 \ \forall \mathbf{s} \in \Sigma^\star$
for $t = 1, 2, \ldots$ **do**
  Predict: $\hat{x}_t = \text{sign}\left(\sum_{i=0}^{t-1} e^{-\alpha\, i}\ g_t\left(\mathbf{x}_{t-i}^{t-1}\right)\right)$
  Receive x_t
  if $(\hat{x}_t = x_t)$ **then**
   $V_{t+1} = V_t$
   $g_{t+1} = g_t$
  else
   $P_t = \{\mathbf{x}_{t-i}^{t-1} : 0 \le i \le t-1\}$
   $V_{t+1} \ = \ V_t \ \cup \ P_t$
   $g_{t+1}(\mathbf{s}) = \begin{cases} g_t(\mathbf{s}) + x_t\ e^{-\alpha\, i} & \text{if } \mathbf{s} = \mathbf{x}_{t-i}^{t-1} \in P_t \\ g_t(\mathbf{s}) & \text{otherwise} \end{cases}$
end for

Fig. 2. The arbitrary-depth Perceptron for context tree learning.

the definition of context tree complexity: adding a (unit weight) new node of depth i to a tree τ increases the complexity of the corresponding context function by $e^{2\,\alpha\, i}$.

We can now adapt the Perceptron algorithm to the problem of learning arbitrary-depth context trees. In our case, the input on round t is f_t and the output is x_t. The Perceptron predicts the label on round t to be the sign of $\langle g_t, f_t \rangle$, where the context function $g_t \in \mathcal{H}$ plays the role of the current hypothesis. We also define V_t to be the smallest suffix-closed set which contains the set $\{\mathbf{s} : g_t(\mathbf{s}) \neq 0\}$ and we picture V_t as a rooted tree. The Perceptron initializes g_1 to be the zero function in $\mathcal{H}$, which is equivalent to initializing V_1 to be a tree of a single node (the root) which assigns a weight of zero to the empty sequence. After predicting a binary symbol and receiving the correct answer, the Perceptron defines g_{t+1}. If x_t is correctly predicted, then g_{t+1} is simply set to be equal to g_t. Otherwise, the Perceptron updates its hypothesis using the rule $g_{t+1} = g_t + x_t f_t$. In this case, the function g_{t+1} differs from the function g_t only on inputs $\mathbf{s}$ for which $f_t(\mathbf{s}) \neq 0$, namely on every $\mathbf{x}_{t-i}^{t-1}$ for $0 \le i \le t-1$. For these inputs, the update takes the form $g_{t+1}(\mathbf{x}_{t-i}^{t-1}) = g_t(\mathbf{x}_{t-i}^{t-1}) + x_t e^{-\alpha\, i}$. The pseudo-code of the Perceptron algorithm applied to the context function learning problem is given in Fig. 2.

We readily identify a major drawback with this approach. The number of non-zero elements in f_t is $t-1$. Therefore, $|V_{t+1}| - |V_t|$ may be on the order of t. This means that the number of new nodes added to the context tree on round t may be on the order of t, and the size of V_t my grow quadratically with t. To underscore the issue, we refer to this algorithm as the *arbitrary-depth* Perceptron for context tree learning.

Implementation shortcuts, such as the one described in [16], can reduce the space complexity of storing V_t to $O(t)$, however even memory requirements that grow linearly with t can impose serious computational problems. Consequently, the arbitrary-depth Perceptron may not constitute a practical choice for context tree learning, and we present it primarily for illustrative purposes. We resolve this memory growth problem in the next section, where we modify the Perceptron algorithm such that it utilizes memory more conservatively.

The mistake bound of Lemma 1 assumes that the maximal norm of f_t, where f_t is given in Eq. (15), is bounded. To show that the norm of f_t is bounded, we use the fact that $\|f_t\|^2$ can be written as a geometric series and therefore can be bounded based on the decay parameter α as follows,

$$\|f_t\|^2 \ = \ \sum_{i=0}^{t} e^{-2\alpha\, i} \ = \ \frac{1 - e^{-2\alpha\, t}}{1 - e^{-2\alpha}} \ \le \ \frac{1}{1 - e^{-2\alpha}} \ . \tag{19}$$

Applying Lemma 1 with the above bound on $\|f_t\|$ we obtain the following mistake bound for the arbitrary-depth Perceptron for context tree learning.

Theorem 5 *Let $x_1, x_2, \ldots, x_T$ be a sequence of binary symbols. Let $g_1^\star, \ldots, g_T^\star$ be a sequence of arbitrary context functions, defined with decay parameter α. Define $\ell_t^\star = \ell(g_t^\star, \mathbf{x}_1^t)$, $L^\star = \sum_{t=1}^T \ell_t^\star$, $S = \sum_{t=2}^T \|g_t^\star - g_{t-1}^\star\|$, $U = \max_t \|g_t^\star\|$ and $R = 1/\sqrt{1 - e^{-2\alpha}}$. Let M denote the number of prediction mistakes made by the arbitrary-depth Perceptron with decay parameter α when it is presented with the sequence of symbols. Then,*

$$M \ \le \ L^\star + R^2\,(S+U)^2 + R\,(S+U)\sqrt{L^\star} \ .$$

Proof: Using the equivalence of context function learning and linear separation along with Eq. (19), we apply Lemma 1 with $R = 1/\sqrt{1 - e^{-2\alpha}}$ and obtain the inequality,

$$M - R\,(U+S)\sqrt{M} - L^\star \le 0 \ . \tag{20}$$

Solving the above for M (see Lemma 8 in the appendix) proves the theorem. ∎

The mistake bound of Thm. 5 depends on U, the maximal norm of a competing context function $g_t^\star$. As mentioned before, since we assumed that $\mathcal{H}$ is square integrable, U is always finite. However, if a context tree $\tau : V \to \mathbb{R}$ contains a node of depth k, then Eq. (16) implies that the induced context function has a squared norm of at least $\exp(2\,\alpha\, k)$. Consequently, the

mistake bound given in Thm. 5 is at least $R^2\,U^2 = \Omega\left(\frac{\exp(2\,\alpha\,k)}{1-\exp(-2\,\alpha)}\right) \geq \Omega(k)$. Put another way, after observing a sequence of T symbols, we cannot hope to compete with context trees of depth $\Omega(T)$. This fact is by no means surprising. Indeed, the following simple lemma implies that no algorithm can compete with a tree of depth $T-1$ after observing only T symbols.

Lemma 6 *For any online prediction algorithm, there exists a sequence of binary symbols $x_1, \ldots, x_T$ such that the number of prediction mistakes is T while there exists a context tree $\tau^\star : V \to \mathbb{R}$ of depth $T-1$ that perfectly predicts the sequence. That is, for all $1 \leq t \leq T$ we have $\mathrm{sign}(\tau^\star(\mathbf{x}_1^{t\text{-}1})) = x_t$ and $|\tau^\star(\mathbf{x}_1^{t\text{-}1})| \geq 1$.*

Proof: Let $\hat{x}_t$ be the prediction of the online algorithm on round t, and set $x_t = -\hat{x}_t$. Clearly, the online algorithm makes T prediction mistakes on the sequence. In addition, let $V = \cup_{t=1}^{T} \{\mathbf{x}_1^{t\text{-}1}\}$ and $\tau^\star(\mathbf{x}_1^{t\text{-}1}) = x_t$. Then, the depth of $\tau^\star$ is $T-1$ and $\tau^\star$ perfectly predicts the sequence. ■

To conclude this section, we discuss the effect of the decay parameter α as implied by Thm. 5. On one hand, a large value of α causes R to be small, which results in a better mistake bound. On the other hand, the construction of a context function g from a given margin-based context tree τ depends on α (see Eq. (16)), and in particular, $\|g_t^\star\|$ increases with α. Therefore, as α increases, so do U and S.

To illustrate the effect of α, consider again the realizable case, in which the sequence is generated by a context tree with r nodes, of maximal depth k. If V is known ahead of time, we can run the algorithm defined in Sec. II and obtain the mistake bound $M \leq r$. However, if V is unknown to us, but we do know the maximal depth k, we can run the same algorithm with V set to contain all strings of length at most k, and obtain the mistake bound $M \leq 2^k$. An alternative approach is to use the arbitrary-depth Perceptron described in this section. In that case, Eq. (16) implies that $\|g^\star\|^2 \leq r\,\exp(2\,\alpha\,k)$ and the mistake bound becomes $M = (R\,U)^2 \leq \frac{r\,\exp(2\,\alpha\,k)}{1-\exp(-2\,\alpha)}$. Solving for α yields the optimal choice of $\alpha = \frac{1}{2}\log(1+\frac{1}{k}) \approx \frac{1}{2\,k}$ and the mistake bound becomes approximately $r\,k$, which can be much smaller than 2^k if $r \ll 2^k$. The optimal choice of α depends on $\tau_1^\star, \ldots, \tau_T^\star$, the sequence of context functions the algorithm is competing with, which we do not know in advance. Nevertheless, no matter how we set α, our algorithm remains asymptotically competitive with trees of arbitrary depth.

V. The Shallow Perceptron

As mentioned in the previous section, the arbitrary-depth Perceptron suffers from a major drawback: the memory required to hold the current context tree may grow linearly with the length of the sequence. We resolve this problem by aggressively limiting the growth rate of the context trees generated by our algorithm. Specifically, before adding a new path to the tree, we prune it to a predefined length. We perform the pruning in a way that ensures that the number of nodes in the current context tree never exceeds the number of prediction mistakes made so far. Although the context tree grows at a much slower pace than in the case of the arbitrary-depth Perceptron, we can still prove a mistake bound similar to Thm. 5. This mistake bound naturally translates into a bound on the growth-rate of the resulting context tree.

Recall that every time a prediction mistake is made, the Perceptron performs the update $g_{t+1} = g_t + x_t f_t$, where f_t is defined in Eq. (15). Since the non-zero elements of f_t correspond to $\mathbf{x}_{t\text{-}i}^{t\text{-}1}$ for all $0 \leq i \leq t-1$, this update adds a path of depth t to the current context tree. Let M_t denote the number of prediction mistakes made on rounds 1 through t. Instead of adding the full path to the context tree, we limit the path length to d_t, where d_t is proportional to $\log(M_t)$. We call the resulting algorithm the *shallow Perceptron* for context tree learning, since it grows shallow trees. Formally, define the abbreviations, $\beta = \frac{\ln 2}{2\alpha}$ and $d_t = \lfloor \beta \log(M_t) \rfloor$. We set

$$g_{t+1}(\mathbf{s}) = \begin{cases} g_t(\mathbf{s}) + x_t e^{-\alpha\,i} & \text{if } \exists i \leq d_t \text{ s.t } \mathbf{s} = \mathbf{x}_{t\text{-}i}^{t\text{-}1} \\ g_t(\mathbf{s}) & \text{otherwise} \end{cases} \tag{21}$$

An analogous way of stating this update rule is obtained by defining the function $\nu_t \in \mathcal{H}$

$$\nu_t(\mathbf{s}) \;=\; \begin{cases} -x_t e^{-\alpha\,i} & \text{if } \exists i \leq d_t \text{ s.t } \mathbf{s} = \mathbf{x}_{t\text{-}i}^{t\text{-}1} \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

Using the definition of ν_t, we can state the shallow Perceptron update from Eq. (21) as

$$g_{t+1} \;=\; g_t + x_t f_t + \nu_t \quad . \tag{23}$$

Namely, the update is obtained by first applying the standard Perceptron update, and then altering the outcome of this update by adding the vector ν_t. It is convenient to conceptually think of ν_t as an additive noise which contaminates the updated hypothesis. Intuitively, the standard Perceptron update guarantees positive progress, whereas the additive noise ν_t pushes the updated hypothesis slightly off its course.

Next, we note some important properties of the shallow Perceptron update. Primarily, it ensures that the maximal depth of V_{t+1} always equals d_t. This property trivially follows from the fact that d_t is monotonically non-decreasing in t. Since V_{t+1} is a binary tree of depth $\lfloor \beta \log(M_t) \rfloor$, then it must contain less than

$$2^{\lfloor \beta \log(M_t) \rfloor} \;\leq\; 2^{\beta \log(M_t)} \;=\; M_t^{\beta}$$

nodes. Therefore, any bound on the number of prediction mistakes made by the shallow Perceptron also yields a bound on the size of its context tree hypothesis. For instance, when $\alpha = \frac{1}{2}\log(2)$, we have $\beta = 1$ and the size of V_{t+1} is upper bounded by M_t.

Another property which follows from the bound on the maximal depth of V_t is that

$$\langle g_t, f_t + x_t \nu_t \rangle \;=\; \langle g_t, f_t \rangle + x_t \langle g_t, \nu_t \rangle \;=\; \langle g_t, f_t \rangle \quad , \tag{24}$$

where the last equality holds simply because $g_t(\mathbf{s}) = 0$ for every sequence s whose length exceeds d_t. As defined above, the shallow Perceptron makes its prediction based on the entire sequence $\mathbf{x}_1^{t\text{-}1}$, and confines the usage of the input sequence to a suffix of d_t symbols before performing an update. A simple but powerful implication of Eq. (24) is that we can equivalently limit $\mathbf{x}_1^{t\text{-}1}$ to length d_t before the Perceptron extends its prediction. This property comes in handy in the proof of the following theorem, which bounds the number of prediction mistakes made by the shallow Perceptron.

Theorem 7 *Let $x_1, x_2, \ldots, x_T$ be a sequence of binary symbols. Let $g_1^\star, \ldots, g_T^\star$ be a sequence of arbitrary context functions, defined with a decay parameter α. Define $\ell_t^\star = \ell(g_t^\star, \mathbf{x}_1^t)$, $L^\star = \sum_{t=1}^T \ell_t^\star$, $S = \sum_{t=2}^T \|g_t^\star - g_{t-1}^\star\|$, $U = \max_t \|g_t^\star\|$ and $R = 1/\sqrt{1-e^{-2\alpha}}$. Let M denote the number of prediction mistakes made by the shallow Perceptron with a decay parameter α when it is presented with the sequence of symbols. Then,*

$$M \;\leq\; L^\star + R^2 \Big(S + 3\,U\Big)^2 + R\Big(S + 3\,U\Big)\sqrt{L^\star} \quad .$$

Proof: Let $\hat{f}_t = f_t + x_t \nu_t$. In other words,

$$\hat{f}_t(\mathbf{s}) = \begin{cases} e^{-\alpha\, i} & \text{if } \; \exists i \;\; \text{s.t} \;\; \mathbf{s} = \mathbf{x}_{t-i}^{t\text{-}1} \;\; \wedge \;\; i \leq d_t \\ 0 & \text{otherwise} \end{cases} \quad . \tag{25}$$

Since $g_t(\mathbf{s}) = 0$ for every s which is longer than d_t, we have $\langle g_t, f_t \rangle = \left\langle g_t, \hat{f}_t \right\rangle$. Additionally, we can rewrite the shallow Perceptron update, defined in Eq. (23), as

$$g_{t+1} \;=\; g_t + x_t \hat{f}_t \quad . \tag{26}$$

The above two equalities imply that the shallow Perceptron update is equivalent to a straightforward application of the Perceptron to the input sequence $\{(\hat{f}_t, x_t)\}_{t=1}^T$, which enables us to use the bound in Lemma 1. Eq. (25) also implies that the norm of $\hat{f}_t$ is bounded,

$$\|\hat{f}_t\|^2 = \sum_{i=0}^{d_t} e^{-2\alpha\, i} \;=\; \frac{1 - e^{-2\alpha\, d_t}}{1 - e^{-2\alpha}} \;\leq\; \frac{1}{1 - e^{-2\alpha}} \quad ,$$

so Lemma 1 is applied with $R = 1/\sqrt{1-e^{-2\alpha}}$, and we have,

$$M - R\,(S+U)\sqrt{M} \;\leq\; \sum_{t=1}^{T} \Big[1 - x_t \Big\langle g_t^\star, \hat{f}_t \Big\rangle\Big]_+ \quad . \tag{27}$$

We focus on the term $\Big[1 - x_t \Big\langle g_t^\star, \hat{f}_t \Big\rangle\Big]_+$. Since $\hat{f}_t = f_t + x_t \nu_t$, we rewrite

$$\Big[1 - x_t \Big\langle g_t^\star, \hat{f}_t \Big\rangle\Big]_+ \;=\; [1 - x_t \langle g_t^\star, f_t \rangle - \langle g_t^\star, \nu_t \rangle]_+ \quad .$$

Using the Cauchy-Schwartz inequality and the fact that the hinge-loss is a Lipschitz function, we obtain the upper bound

$$\Big[1 - x_t \Big\langle g_t^\star, \hat{f}_t \Big\rangle\Big]_+ \;\leq\; [1 - x_t \langle g_t^\star, f_t \rangle]_+ + \|g_t^\star\|\,\|\nu_t\| \quad .$$

Recall that $\ell_t^\star = [1 - x_t \langle g_t^\star, f_t \rangle]_+$. Summing both sides of the above inequality over $1 \leq t \leq T$ and using the definitions of $L^\star$ and U, we obtain

$$\sum_{t=1}^{T} \Big[1 - x_t \Big\langle g_t^\star, \hat{f}_t \Big\rangle\Big]_+ \;\leq\; L^\star + U \sum_{t=1}^{T} \|\nu_t\| \quad . \tag{28}$$

On rounds where the Shallow Perceptron makes a correct prediction, $\|\nu_t\| = 0$. If a prediction mistake is made on round t, then

$$\begin{aligned} \|\nu_t\|^2 \;&=\; \sum_{i=d_t+1}^{t} e^{-2\,\alpha\, i} \;\leq\; \frac{e^{-2\alpha\,(d_t+1)}}{1 - e^{-2\alpha}} \\ &=\; R^2\, e^{-2\alpha\,(\lfloor \beta\, \log(M_t) \rfloor + 1)} \\ &\leq\; R^2\, e^{-2\alpha\,\beta\, \log(M_t)} \\ &=\; \frac{R^2}{M_t} \quad , \end{aligned}$$

and therefore $\|\nu_t\| \leq R\,/\sqrt{M_t}$. Summing both sides of this inequality over $1 \leq t \leq T$ gives

$$\sum_{t=1}^{T} \|\nu_t\| \;\leq\; R \sum_{i=1}^{M} \sqrt{1/i} \quad .$$

Since the function $1/\sqrt{x}$ is monotonically decreasing in x, we obtain the bound,

$$\begin{aligned} \sum_{i=1}^{M} \sqrt{1/i} \;&\leq\; 1 + \int_1^M \sqrt{1/x}\; dx \\ &=\; 1 + 2\sqrt{M} - 2 \;\leq\; 2\sqrt{M} \quad . \end{aligned}$$

input: decay parameter α
initialize: $\beta = \frac{\ln 2}{2\alpha}$, $V_1 = \{\epsilon\}$,
$g_1(\mathbf{s}) = 0 \ \forall \mathbf{s} \in \Sigma^\star$, $M_0 = 0$, $d_0 = 0$
for $t = 1, 2, \ldots$ **do**
Predict: $\hat{x}_t = \text{sign}\left(\sum_{i=0}^{d_{t-1}} e^{-\alpha\, i}\ g_t\left(\mathbf{x}_{t-i}^{t-1}\right)\right)$
Receive x_t
if $(\hat{x}_t \neq x_t)$ **then**
Set: $M_t = M_{t-1}$, $d_t = d_{t-1}$,
$V_{t+1} = V_t$, $g_{t+1} = g_t$
else
Set: $M_t = M_{t-1} + 1$
Set: $d_t = \lfloor \beta \log(M_t) \rfloor$
$P_t = \{\mathbf{x}_{t-i}^{t-1} : 0 \leq i \leq d_t\}$
$V_{t+1} = V_t \cup P_t$
$g_{t+1}(\mathbf{s}) = \begin{cases} g_t(\mathbf{s}) + x_t\, e^{-\alpha\, i} & \text{if } \mathbf{s} = \mathbf{x}_{t-i}^{t-1} \in P_t \\ g_t(\mathbf{s}) & \text{otherwise} \end{cases}$
end for

Fig. 3. The shallow Perceptron for context tree learning.

Recapping, we showed that $\sum_{t=1}^{T} \|\nu_t\| \leq 2\,R\sqrt{M}$. Combining this inequality with Eq. (28) gives

$$\sum_{t=1}^{T} \left[1 - x_t \left\langle g_t^\star, \hat{f}_t \right\rangle\right]_+ \leq L^\star + 2\,R\,U\,\sqrt{M} \quad .$$

Combining this inequality with Eq. (27) and rearranging terms gives

$$M - R\left(S + 3U\right)\sqrt{M} - L^\star \leq 0 \quad . \tag{29}$$

Solving the above for M (see again Lemma 8 in the appendix) concludes the proof. ∎

VI. Discussion

In this paper, we addressed the widely studied problem of individual sequence prediction using well known machine learning tools. By recasting the sequence prediction problem as the problem of linear separation in a Hilbert space, we gained the ability to use several of-the-shelf algorithms to learn context tree predictors. However, the standard algorithms lacked an adequate control of the context tree size. This drawback motivated the derivation of the Shallow Perceptron algorithm.

A key advantage of our approach is that our prediction algorithm updates its context tree in a conservative manner. In other words, if the predictor stops making prediction mistakes, the context tree stops growing. For example, take the infinitely alternating binary sequence $(-1, +1, -1, +1, \ldots)$. This sequence is trivially realized by a context tree of depth 1. For this sequence, even the simple arbitrary-depth predictor presented in Sec. IV would grow a depth 1 tree and then cease making updates. On the other hand, the IP predictor of Feder et al. [6] would continue to grow its context tree infinitely, even though it is clearly unnecessary.

In contrast to typical information-theoretic algorithms for sequence prediction, the Perceptron-based algorithms presented in this paper do not rely on randomized predictions. Throughout this paper, we sidestepped Cover's impossibility result [17], which states that deterministic predictors cannot have a vanishing regret, universally for all sequences. We overcame the difficulty by using the hinge-loss function as a proxy for the error (indicator) function. As a consequence of this choice, our bounds are not proper regret bounds, and can only be compared to proper regret bounds in the realizable case, where the sequence is deterministically generated by some context tree. On the other hand, when the sequence is indeed realizable, the convergence rates of our bounds are superior to those of randomized sequence prediction algorithms, such as those presented in [5], [6]. Additionally, our approach allows us to prove shifting bounds, which compare the performance of our algorithms with the performance of any predefined sequence of margin-based context trees.

Our algorithms can be extended in a number of straightforward ways. First, when the size of the symbol alphabet is not binary, we can simply replace the binary Perceptron algorithm with one of its multiclass classification extensions (see for example Kessler's construction in [25] and [26]). It is also rather straightforward to obtain a multiclass variant of the Shallow Perceptron algorithm, using the same techniques used to extend the standard Perceptron to multiclass problems. Another simple extension is the incorporation of side information. If the side information can be given in the form of a vector in a Hilbert space $\tilde{\mathcal{H}}$, then we can incorporate it into our predictions by applying the Perceptron algorithm in the product space $\mathcal{H} \times \tilde{\mathcal{H}}$.

This work also gives rise to a few interesting open problems. First, it is worth investigating whether our approach could be used for compression. A binary sequence $x_1, \ldots, x_T$ can be compressed using our deterministic predictor by transmitting only the indices of the symbols that are incorrectly predicted. Thus, the average number of prediction mistakes made by our algorithm is precisely the compression ratio of the induced compressor. A seemingly more direct application of our techniques to the compression problem, namely one that does not make a detour through the prediction problem,

could yield better theoretical guarantees. Another direction which deserves more attention is the relationship between randomization and margin-based approaches. The connections between the two seem to run deeper than the result provided in Sec. III. Finally, it would be very interesting to prove an expected shifting regret bound, that is, a bound with respect to a sequence of competitors rather than with respect to a single competitor, for any of the randomized sequential predictors referenced in this paper. We leave these questions open for future research.

ACKNOWLEDGMENT

We would like to thank Tsachy Weissman and Yaacov Ziv for correspondences on the general framework.

REFERENCES

[1] H. Robbins, "Asymptotically subminimax solutions of compound statistical decision problems," in *Proceedings of the 2nd Berkeley symposium on mathematical statistics and probability*, 1951, pp. 131–148.

[2] D. Blackwell, "An analog of the minimax theorem for vector payoffs," *Pacific Journal of Mathematics*, vol. 6, no. 1, pp. 1–8, Spring 1956.

[3] J. Hannan, "Approximation to Bayes risk in repeated play," in *Contributions to the Theory of Games*, M. Dresher, A. W. Tucker, and P. Wolfe, Eds. Princeton University Press, 1957, vol. III, pp. 97–139.

[4] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions in Information Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[5] T. M. Cover and A. Shenhar, "Compound Bayes predictors for sequences with apparent Markov structure," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 6, pp. 421–424, June 1977.

[6] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of individual sequences," *IEEE Transactions on Information Theory*, vol. 38, pp. 1258–1270, 1992.

[7] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "Context tree weighting: a sequential universal source coding procedure for FSMX sources," in *Proceedings of the IEEE International Symposium on Information Theory*, 1993, p. 59.

[8] D. P. Helmbold and R. E. Schapire, "Predicting nearly as well as the best pruning of a decision tree," *Machine Learning*, vol. 27, no. 1, pp. 51–68, Apr. 1997.

[9] F. Pereira and Y. Singer, "An efficient extension to mixture techniques for prediction and decision trees," *Machine Learning*, vol. 36, no. 3, pp. 183–199, 1999.

[10] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006.

[11] J. Ziv and A. Lempel, "Compression of individual sequences via variable rate coding," *IEEE Transactions on Information Theory*, vol. 24, pp. 530–536, 1978.

[12] F. M. J. Willems, "Extensions to the context tree weighting method," in *Proceedings of the IEEE International Symposium on Information Theory*, 1994, p. 387.

[13] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, 1995.

[14] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, no. 2, pp. 117–150, 1996.

[15] P. Buhlmann and A. Wyner, "Variable length markov chains," *The Annals of Statistics*, vol. 27, no. 2, pp. 480–513, 1999.

[16] G. Bejerano and A. Apostolico, "Optimal amnesic probabilistic automata, or, how to learn and classify proteins in linear time and space," *Journal of Computational Biology*, vol. 7, no. 3/4, pp. 381–393, 2000.

[17] T. M. Cover, "Behavior of sequential predictors of binary sequences," *Trans. 4th Prague Conf. Information Theory Statistical Decision Functions, Random Processes*, 1965.

[18] S. Agmon, "The relaxation method for linear inequalities," *Canadian Journal of Mathematics*, vol. 6, no. 3, pp. 382–392, 1954.

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–407, 1958, (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[20] A. B. J. Novikoff, "On convergence proofs on perceptrons," in *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. XII, 1962, pp. 615–622.

[21] S. Shalev-Shwartz and Y. Singer, "Convex repeated games and fenchel duality," in *Advances in Neural Information Processing Systems 20*, 2006.

[22] S. Shalev-Shwartz, "Online learning: Theory, algorithms, and applications," Ph.D. dissertation, The Hebrew University, 2007.

[23] N. Cesa-Bianchi and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," in *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006, pp. 483–498.

[24] C. Gentile, "The robustness of the p-norm algorithms," *Machine Learning*, vol. 53, no. 3, 2002.

[25] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[26] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, Mar 2006.

APPENDIX

Proof: [*of Lemma 1*] We prove the lemma by bounding $\langle u_T^\star, \tau_{T+1} \rangle$ from above and from below, starting with an upper bound. Using the Cauchy-Schwartz inequality and the definition of U we get that

$$\langle u_T^\star, \tau_{T+1} \rangle \;\leq\; \|u_T^\star\| \, \|\tau_{T+1}\| \;\leq\; U \, \|\tau_{T+1}\| \quad . \tag{30}$$

Next, we upper bound $\|\tau_{T+1}\|$ by $\sqrt{M}$. The Perceptron update sets $\tau_{t+1} = \tau_t + \rho_t x_t \phi_t$, where $\rho_t = 1$ if a prediction mistake occurs on round t, and $\rho_t = 0$ otherwise. Expanding the squared norm of τ_{t+1} we get

$$\begin{aligned} \|\tau_{t+1}\|^2 &= \|\tau_t + \rho_t x_t \phi_t\|^2 \\ &= \|\tau_t\|^2 + 2\rho_t x_t \langle \tau_t, \phi_t \rangle + \rho_t \|\phi_t\|^2 \quad . \end{aligned}$$

If $\rho_t = 1$ then a prediction mistake is made on round t and $x_t \langle \tau_t, \phi_t \rangle \leq 0$. Additionally, we assume that $\|\phi_t\| \leq R$. Using these two facts gives

$$\|\tau_{t+1}\|^2 \;\leq\; \|\tau_t\|^2 + R^2 \rho_t \quad .$$

If $\rho_t = 0$ then $\tau_{t+1} = \tau_t$ and the above clearly holds as well. Since $\tau_1 \equiv 0$, we obtain that, for all t, $\|\tau_{t+1}\|^2 \leq R^2 \sum_{i=1}^{t} \rho_i$, and in particular $\|\tau_{T+1}\| \leq R\sqrt{M}$. Plugging this fact into Eq. (30) gives the upper bound

$$\langle u_T^\star, \tau_{T+1} \rangle \;\leq\; R\,U\,\sqrt{M} \quad . \tag{31}$$

Next we derive a lower bound on $\langle u_T^\star, \tau_{T+1}\rangle$. Again, using the fact that $\tau_{t+1} = \tau_t + \rho_t x_t \phi_t$ gives

$$\begin{aligned}\langle u_t^\star, \tau_{t+1}\rangle &= \langle u_t^\star, \tau_t + \rho_t x_t \phi_t\rangle \\ &= \langle u_t^\star, \tau_t\rangle + \rho_t x_t \langle u_t^\star, \phi_t\rangle \quad .\end{aligned}$$

The definition of the hinge-loss in Eq. (1) implies that $\ell_t^\star = [1 - x_t \langle u_t^\star, \phi_t\rangle]_+ \geq 1 - x_t \langle u_t^\star, \phi_t\rangle$. Since the hinge-loss is non-negative, we get

$$\rho_t x_t \langle u_t^\star, \phi_t\rangle \;\geq\; \rho_t(1-\ell_t^\star) \;\geq\; \rho_t - \ell_t^\star \quad .$$

Overall, we have shown that

$$\langle u_t^\star, \tau_{t+1}\rangle \;\geq\; \langle u_t^\star, \tau_t\rangle + \rho_t - \ell_t^\star \quad .$$

Adding the null term $\langle u_{t-1}^\star, \tau_t\rangle - \langle u_{t-1}^\star, \tau_t\rangle$ to the above and rearranging terms, we get

$$\langle u_t^\star, \tau_{t+1}\rangle \;\geq\; \langle u_{t-1}^\star, \tau_t\rangle + \langle u_t^\star - u_{t-1}^\star, \tau_t\rangle + \rho_t - \ell_t^\star \quad .$$

Using the Cauchy-Schwartz inequality on the term $\langle u_t^\star - u_{t-1}^\star, \tau_t\rangle$, the above becomes

$$\langle u_t^\star, \tau_{t+1}\rangle \;\geq\; \langle u_{t-1}^\star, \tau_t\rangle - \|u_t^\star - u_{t-1}^\star\|\,\|\tau_t\| + \rho_t - \ell_t^\star \quad .$$

Again using the fact that $\|\tau_t\| \leq R\sqrt{M}$, we have

$$\langle u_t^\star, \tau_{t+1}\rangle \;\geq\; \langle u_{t-1}^\star, \tau_t\rangle - R\sqrt{M}\,\|u_t^\star - u_{t-1}^\star\| + \rho_t - \ell_t^\star \quad .$$

Applying this inequality recursively, for $t = 2, \ldots, T$, gives

$$\begin{aligned}\langle u_T^\star, \tau_{T+1}\rangle \;\geq\; \langle u_1^\star, \tau_2\rangle &- R\sqrt{M}\sum_{t=2}^{T}\|u_t^\star - u_{t-1}^\star\| \\ &+ \sum_{t=2}^{T}\rho_t - \sum_{t=2}^{T}\ell_t^\star \quad .\end{aligned} \tag{32}$$

Since $\tau_1 \equiv 0$, our algorithm necessarily invokes an update on the first round. Therefore, $\tau_2 = x_1\phi_1$, and $\langle u_1^\star, \tau_2\rangle = x_1 \langle u_1^\star, \phi_1\rangle$. Once again, using the definition of the hinge-loss in Eq. (1), we can lower bound, $\langle u_1^\star, \tau_2\rangle \geq 1 - \ell_1^\star$. Plugging this inequality back into Eq. (32) gives

$$\langle u_T^\star, \tau_{T+1}\rangle \;\geq\; -R\sqrt{M}\sum_{t=2}^{T}\|u_t^\star - u_{t-1}^\star\| + \sum_{t=1}^{T}\rho_t - \sum_{t=1}^{T}\ell_t^\star \quad .$$

Using the definitions of S and M, we rewrite the above as

$$\langle u_T^\star, \tau_{T+1}\rangle \;\geq\; -R\sqrt{M}S + M - L^\star \quad .$$

Comparing the lower bound given above with the upper bound in Eq. (31) proves the lemma. ∎

Proof: [*of Thm. 3*] Since our randomized algorithm is similar to the randomized algorithm of [6], one can prove Thm. 3 using the proof technique of [6]. An alternative route, which we adopt here, is to directly use general regret bounds for online convex programming. For completeness, let us first describe a setting which is a special case of the algorithmic framework given in [22] for online convex programming. This special case is derived from Fig. 3.2 in [22] by choosing the strongly convex function to be $\frac{1}{2}\|\mathbf{w}\|^2$ with a domain S and the dual update scheme to be according to Eq. (3.11) in [22]. Let $S \subset \mathbb{R}^n$ be a convex set and let $\Pi : \mathbb{R}^n \to S$ be the Euclidean projection onto S, that is, $\Pi(\tilde{\tau}) = \arg\min_{\tau\in S}\|\tau - \tilde{\tau}\|$. Denote $U = \max_{\tau\in S}\frac{1}{2}\|\tau\|^2$ and let L be a constant. The algorithm maintains a vector $\tilde{\tau}_t$ which is initialized to be the zero vector, $\tilde{\tau}_1 = (0, \ldots, 0)$. On round t, the algorithm sets $c_t = \sqrt{tL/U}$ and predicts $\tau_t = \Pi(\tilde{\tau}_t/c_t)$. Then it receives a loss function $\ell_t : S \to \mathbb{R}$. Finally, the algorithm updates $\tilde{\tau}_{t+1} = \tilde{\tau}_t - \lambda_t$ where λ_t is a sub-gradient of ℓ_t computed at τ_t. Assuming that $\|\lambda_t\| \leq \sqrt{2\,L}$ for all t, Corollary 3 in [22] implies the bound

$$\forall \tau^\star \in S, \quad \tfrac{1}{T}\sum_{t=1}^{T}\ell_t(\tau_t) - \tfrac{1}{T}\sum_{t=1}^{T}\ell_t(\tau^\star) \;\leq\; 4\sqrt{\tfrac{LU}{T}} \quad . \tag{33}$$

In our case, let $S = [+1, -1]^{|V|}$. This choice of S implies that S is a convex set and that $\tau^\star \in S$. For each round t, define $\ell_t(\tau) = [1 - x_t \langle \tau, \phi_t\rangle]_+$ and note that if $\tau \in S$ then $\lambda_t = -x_t\phi_t$ is a subgradient of ℓ_t at τ. Therefore, the update of $\tilde{\tau}_t$ given in Eq. (11) coincides with the update $\tilde{\tau}_{t+1} = \tilde{\tau}_t - \lambda_t$ as required. It is also simple to verify that the definition of τ_t given in Eq. (12) coincides with $\tau_t = \Pi(\tilde{\tau}_t/c_t)$. We have thus shown that the algorithm defined according to Eq. (13) and Eq. (11) is a special case of the online convex programming setting described above. To analyze the algorithm we note that $U = \max_{\tau\in S}\frac{1}{2}\|\tau\|^2 = \frac{|V|}{2}$ and that for all t we have $\|\lambda_t\| = 1$. Therefore, Eq. (33) gives

$$\forall \tau^\star \in S, \quad \tfrac{1}{T}\sum_{t=1}^{T}\ell_t(\tau_t) - \tfrac{1}{T}\sum_{t=1}^{T}\ell_t(\tau^\star) \;\leq\; 2\sqrt{\tfrac{|V|}{T}} \quad . \tag{34}$$

Finally, making predictions based on τ_t yields the randomized prediction given in Eq. (13). Thus, using Eq. (9) we obtain that

$$\ell_t(\tau_t) \;=\; 2\,\mathbb{E}[\hat{x}_t \neq x_t] \quad \text{and} \quad \ell_t(\tau^\star) \;=\; 2\,\mathbb{E}[y_t^\star \neq x_t] \quad .$$

Combining the above equalities with Eq. (34) concludes our proof. ∎

Proof: [*of Lemma 4*] Let g' denote the function obtained by applying the mapping defined in Eq. (16) to τ. Our goal is thus to show that $g' \equiv g$. Let $\mathbf{s} \in \Sigma^\star$ be

an arbitrary sequence. If $\mathbf{s} = \epsilon$ then $g'(\epsilon) = \tau(\epsilon) = g(\epsilon)$. If $\mathbf{s} \notin V$ then $g'(\epsilon) = 0$ and the definition of V implies that $g(\epsilon) = 0$ as well. We are left with the case $\mathbf{s} = s_1, \ldots, s_k \in V$. In this case we get that,

$$\begin{aligned} g'(\mathbf{s}_1^k) &= (\tau(\mathbf{s}_1^k) - \tau(\mathbf{s}_2^k))\, e^{\alpha\, k} \\ &= \Big(g(\epsilon) + \sum_{i=0}^{k-1} g(\mathbf{s}_{k-i}^k) e^{-\alpha\,(i+1)} \\ &\quad - g(\epsilon) - \sum_{i=0}^{k-2} g(\mathbf{s}_{k-i}^k) e^{-\alpha\,(i+1)}\Big)\, e^{\alpha\, k} \\ &= g(s_{k-(k-1)}, \ldots, s_k)\, e^{-\alpha\,(k-1+1)}\, e^{\alpha\, k} \\ &= g(\mathbf{s}_1^k) \quad . \end{aligned}$$

■

Lemma 8 *Let x, b, c be non-negative scalars such that, $x - b\sqrt{x} - c \leq 0$, then, $x \leq c + b^2 + b\sqrt{c}$.*

Proof: Denote $Q(y) = y^2 - b\, y - c$ and note that Q is a convex second degree polynomial. Thus, $Q(\sqrt{x}) \leq 0$ whenever $\sqrt{x}$ is between the two roots of $Q(y)$,

$$r_{1,2} = \frac{b}{2} \pm \sqrt{\left(\frac{b}{2}\right)^2 + c} \quad .$$

In particular, $\sqrt{x}$ is smaller than the larger root of $Q(y)$, and thus

$$\sqrt{x} \leq \frac{b}{2} + \sqrt{\left(\frac{b}{2}\right)^2 + c} \quad .$$

Since both sides of the above are non-negative we obtain that

$$\begin{aligned} x &\leq \left(\frac{b}{2} + \sqrt{\left(\frac{b}{2}\right)^2 + c}\right)^2 \\ &= \frac{b}{2}^2 + c + b\sqrt{\left(\frac{b}{2}\right)^2 + c} \\ &\leq c + b^2 + b\sqrt{c} \quad . \end{aligned}$$

■

Ofer Dekel joined Microsoft Research in 2007, after receiving his Ph.D. in computer science from the Hebrew University of Jerusalem. His research interests include statistical learning theory, online prediction, and theoretical computer science.

Shai Shalev-Shwartz received the Ph.D. degree in computer science from The Hebrew University of Jerusalem, in 2007. From 2007 through 2009 he was a research Assistant Professor of Computer Science at Toyota Technological Institute at Chicago. He is now an Assistant Professor of computer science at the Hebrew University of Jerusalem. His research interests are in the areas of machine learning, online prediction, and optimization techniques.

Yoram Singer is a senior research scientist at Google. From 1999 through 2007 he was an associate professor of computer science and engineering at the Hebrew University of Jerusalem. From 1995 through 1999 he was a member of the technical staff at AT&T Research. He received his Ph.D. in computer science from the Hebrew University in 1995. His research focuses on the design, analysis, and implementation of statistical learning algorithms.