Hedging structured concepts

Koolen, W.M.; Warmuth, M.K.; Kivinen, J.

Publication date
2010
Document Version
Final published version
Published in
Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)

Hedging Structured Concepts

Wouter M. Koolen[*]
Advanced Systems Research
Centrum Wiskunde en Informatica
wmkoolen@cwi.nl

Manfred K. Warmuth[†]
Department of Computer Science
UC Santa Cruz
manfred@cse.ucsc.edu

Jyrki Kivinen[‡]
Department of Computer Science
University of Helsinki
jkivinen@cs.helsinki.fi

Abstract

We develop an online algorithm called *Component Hedge* for learning structured concept classes when the loss of a structured concept sums over its components. Example classes include paths through a graph (composed of edges) and partial permutations (composed of assignments). The algorithm maintains a parameter vector with one non-negative weight per component, which always lies in the convex hull of the structured concept class. The algorithm predicts by decomposing the current parameter vector into a convex combination of concepts and choosing one of those concepts at random. The parameters are updated by first performing a multiplicative update and then projecting back into the convex hull. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

1 Introduction

We develop online learning algorithms for structured concepts that are composed of components. For example, sets are composed of elements, permutations of individual assignments, trees have edges as components, etc. The number of components d is considered small, but the number of structured concepts D built from the components is typically exponential in d.

Our algorithms address the following online prediction problem. In each trial the algorithm first produces a concept from the structured class by choosing a concept probabilistically based on its current parameters. It then observes the loss of each concept. Finally, it prepares for the next trial by updating its parameters by incorporating the losses. Since the algorithm "hedges" by choosing the structured concept probabilistically, we analyze the expected loss incurred in each trial. The goal is to develop algorithms with small regret, which is the total expected loss of the online algorithm minus the loss of the best structured concept in the class chosen in hindsight.

We now make a key simplifying assumption on the loss: We assume that the loss of a structured concept in each trial is always the sum of the losses of its components and that the component losses always have range $[0, 1]$. Thus if the concepts are k-element sets chosen out of n elements, then in each trial each element is assigned a loss in $[0, 1]$ and the loss of any particular k-set is simply the sum of the losses of its elements. Similarly for trees, a loss in $[0, 1]$ is assigned to each edge of the graph and the loss of a tree is the sum of the losses of its edges.

We will show that with this simplifying assumption we still have rich learning problems that address a variety of new settings. We give efficient algorithms (i.e. polynomial in d) that serve as an entry point for considering more complex losses in the future.

Perhaps the simplest approach to learning structured concept classes online is Follow the Perturbed Leader (FPL) algorithm [KV05]. FPL adds a random perturbation to the cumulative loss of each individual component, and then plays the structured concept with minimal perturbed loss. FPL is widely applicable, since efficient combinatorial optimization algorithms exist for a broad range of concept classes. Unfortunately, the loss range of the structured concepts enters into the regret bounds that we can prove for FPL. For example,

[*]Supported by BRICKS project AFM2.2. Part of this research was performed while visiting UCSC supported by NSF grant IIS-0917397.

[†]Supported by NSF grant IIS-0917397

[‡]Part of this research was performed while visiting UCSC. Supported by Academy of Finland grant 118653 (Algodan) and the PASCAL Network of Excellence.

for k-sets the loss range is $[0, k]$ because each set contains k elements, for permutations the loss range is $[0, n]$ because each permutation is composed of n assignments, etc.

A second simple approach for learning well compared to the best structured concept is to run the Hedge algorithm of [FS97] with one weight per structured concept. The original algorithm was developed for the so-called expert setting, which in the context of this paper corresponds to learning with sets of size one. To apply this algorithm to our setting, the experts are chosen as the structured concepts in the class we are trying to learn. In this paper we call this algorithm *Expanded Hedge* (EH). It maintains its uncertainty as a probability distribution over all structured concepts and the weight W_C of concept C is proportional to $\exp(-\eta\ell(C))$, where $\ell(C)$ is the total loss of concept C incurred so far and η is a non-negative learning rate.

There are two problems with EH. First, there are exponentially many weights to maintain. However our simplifying assumption assures that $\ell(C)$ is a sum over the losses of the component of C. This implies that W_C is proportional to a product over the components of the structured concept C and this fact can be exploited to still achieve efficient algorithms in some cases. More importantly however, like for FPL, the loss range of the structured concepts usually enters into the best regret bounds that we can prove.

Learning with structured concepts has also been dealt with recently in the bandit domain [CBL09]. However all of this work is based on EH and contains the additional range factors.

Our contribution Our new method, called *Component Hedge* (CH), avoids the additional range factors altogether. Each structured concept C is identified with its incidence vector in $\{0, 1\}^d$ indicating which components are used. The parameter space of CH is simply the convex hull of all concepts in the class $\mathcal{C}$ to be learned. Thus, whereas EH maintains a weight for each structured concept, CH only maintains a weight for each component. The current parameter vector represents CH's first-order "uncertainty" about the quality of each concept. The value of parameter i represents the *usage* of component i in the next prediction. The usages of the components are updated in each trial by incorporating the current losses, and if the usage vector leaves the hull, then it is projected back via a relative entropy projection. The key trick to make this projection efficient is to find a representation of the convex hull of the concepts as a convex polytope with a number of facets that is polynomial in d. We give many applications where this is possible.

We clearly champion the Component Hedge algorithm in this paper because we can prove regret bounds for this algorithm that are tight within constant factors for many structured concept classes. Also it is trivial to enhance CH with a variety of "share updates" that make it robust in the case when the best comparator changes over time [HW98, BW02].

Two instances of CH have appeared before even though this name was not used: learning with k-sets [WK08] and learning with permutations [HW09]. The same polytope we use for paths was also employed in [AHR08] for developing online algorithms for the bandit setting. They avoid the projection step altogether by exploiting a barrier function. The contribution of this paper is to clearly formulate the general methodology of the Component Hedge algorithm and give many more involved combinatorial examples. In the case of permutations we also show how the method can be used to learn truncated permutations. Also in earlier work [TW03] it was pointed out that the Expanded Hedge algorithm can be simulated efficiently in many cases. In particular, the concept class of paths in a directed graph was introduced. However, good bounds were only achieved in very special cases. In this paper we show that CH essentially is optimal for the path problem.

Paper outline We give the basic setup for the structured prediction task, introduce CH and prove its general regret bound in Section 2. We then turn to a list of applications in Section 3: vanilla experts, k-sets, permutations, paths, undirected and directed spanning trees. For each structured concept class we discuss efficient implementation of CH, and derive expected regret bounds for this algorithm. Then in Section 4 we provide matching lower bounds for all examples, showing that the regret of CH is optimal within a constant factor. In Section 5 we compare CH to the existing algorithms EH and FPL. We observe that the best general regret bounds for each algorithm exceed that of CH by a significant range factor. We show that the bounds for these other algorithms can be improved to closely match those of CH whenever the the so-called *unit rule* holds for the algorithms and class. This means any loss vector $\ell \in [0, 1]^d$ can be split into up to d scaled unit loss vectors $\ell_i\, e_i$ and processing these in separate trials always incurs at least as much loss. Unfortunately, for most pairing of the algorithms CH and FPL with the classes we consider in this paper, we have explicit counter examples to the unit rule. Finally, Section 6 concludes with a list of open problems.

2 Component Hedge

Prediction task We consider sequential prediction [HKW98, CBL06] over a structured concept class [KV05, CBL09]. Fix a set of concepts $\mathcal{C} \subseteq \{0, 1\}^d$ of size $D = |\mathcal{C}|$. For example $\mathcal{C}$ could consist of the incidence vectors of subsets of k out of n elements (then $D = \binom{n}{k}$ and $d = n$), or the adjacency matrices of undirected spanning trees on n elements (then $D = (n-1)^{n-2}$ and $d = n(n-1)/2$).

Our online learning protocol proceeds in trials. At trial t, we have to produce a single concept $C^t \in \mathcal{C}$. Then a loss vector $\ell^t \in [0, 1]^d$ is revealed, and we incur loss given by the dot product $C^t \cdot \ell^t$. Although each

Table 1: Example structured concept classes

Case	U	D	d
Experts	1	n	n
k-Sets	k	$\binom{n}{k}$	n
Permutations	n	$n!$	n^2
Paths (from source via $\leq n$ intermediate nodes to sink)	$n+1$	$n! \cdot \mathrm{e} - o(1)$	$n(n+1)+1$
Undirected spanning trees	$n-1$	n^{n-2}	$n(n-1)/2$
Directed spanning trees w. fixed root	$n-1$	n^{n-2}	$(n-1)^2$

component suffers loss at most 1, a concept may suffer loss up to $U \coloneqq \max_{C \in \mathcal{C}} |C|$. We allow randomized algorithms. Thus the expected loss of of the algorithm at trial t is $\mathbb{E}[\mathbf{C}^t] \cdot \ell^t$, where the expectation is over the internal randomization of the algorithm. Our goal is to minimize our *(expected) regret* after T trials

$$\sum_{t=1}^{T} \mathbb{E}[\mathbf{C}^t] \cdot \ell^t - \min_{C \in \mathcal{C}} \sum_{t=1}^{T} C \cdot \ell^t.$$

That is, the difference between our cumulative expected loss and the loss of the best concept in hindsight.

Note that the ith component of $\mathbb{E}[\mathbf{C}^t]$ is the probability that component i is "used in" concept $\mathbf{C}^t$. We therefore call $\mathbb{E}[\mathbf{C}^t]$ the *usage vector*. This vector becomes the internal parameter of our algorithm. The set of all usages vector is the convex hull of the concepts.

2.1 Component Hedge

Two instances of CH appeared before in the literature [HW09, WK08]. Here we give the algorithm in its general form, and prove a general regret bound. The algorithm CH maintains its uncertainty about the best structured concept as a usage vector w^t in $\mathrm{conv}(\mathcal{C}) \subseteq [0,1]^d$, the convex hull of the concepts $\mathcal{C}$. The initial weight w^0 is typically the usage of the uniform distribution on concepts. CH predicts in trial t by decomposing w^{t-1} into a convex combination[1] of the concepts $\mathcal{C}$, then sampling $\mathbf{C}^t$ according to its weight in that convex combination. The expected loss of CH is thus $w^{t-1} \cdot \ell^t$. The updated weight w^t is obtained by trading off the relative entropy with the linear loss:

$$w^t \coloneqq \operatorname*{argmin}_{w \in \mathrm{conv}(\mathcal{C})} \triangle(w \| w^{t-1}) + \eta w \cdot \ell^t, \qquad \text{where} \qquad \triangle(w \| v) = \sum_{i \in [d]} \Big(w_i \ln \frac{w_i}{v_i} + v_i - w_i \Big).$$

It is easy to see that this update can be split into two steps: an unconstrained update followed by relative entropy projection into the convex hull:

$$\begin{aligned} \widehat{w}^t &\coloneqq \operatorname*{argmin}_{w \in \mathbb{R}^d} \triangle(w \| w^{t-1}) + \eta w \cdot \ell^t \\ w^t &\coloneqq \operatorname*{argmin}_{w \in \mathrm{conv}(\mathcal{C})} \triangle(w \| \widehat{w}^t). \end{aligned}$$

It is easy to see that $\widehat{w}_i^t = w_i^{t-1} \mathrm{e}^{-\eta \ell_i^t}$, that is, the old weights are simply scaled down by the exponentiated losses. The result of the relative entropy projection w^t unfortunately does not have a closed form expression.

For CH to be efficiently implementable, the hull has to be captured by polynomial in d many constraints. This will allow us to efficiently decompose any point in the hull as a convex combination of at most $d+1$ concepts. The trickier part is to efficiently implement the projection step. For this purpose one can use generic convex optimization routines. For example this was done in the context of implementing the entropy regularized boosting algorithm [WGV08]. We proceed on a case by case basis and often develop iterative algorithms that locally enforce constraints and do multiple passes over all constraints. See Table 1 for a list of structured concept classes we consider in this paper.

2.2 Regret bounds

As in [HW09], the analysis is split into two steps paralleling the two update steps. Essentially the unnormalized update step already gives the regret bound and the projection step does not hurt. For any usage vector

[1] This decomposition usually is far from unique.

$w^{t-1} \in \mathrm{conv}(\mathcal{C})$, loss vector $\ell^t \in \{0,1\}^d$ and any comparator concept C,

$$\begin{aligned}(1-\mathrm{e}^{-\eta})w^{t-1}\cdot\ell^t \;&\le\; \underbrace{\triangle(C\|w^{t-1})-\triangle(C\|\widehat{w}^t)+\eta\, C\cdot\ell^t}_{\sum_i w_i^{t-1}(1-\mathrm{e}^{-\eta\ell_i^t})}\\ &\le\; \triangle(C\|w^{t-1})-\triangle(C\|w^t)+\eta\, C\cdot\ell^t\end{aligned}$$

The first inequality is obtained by bounding the exponential using the inequality $1-\mathrm{e}^{-\eta x} \ge (1-\mathrm{e}^{-\eta})x$ for $x \in [0,1]$ as done in [LW94]. The second inequality is an application of the Generalized Pythagorean Theorem [HW01], using the fact that w^t is a Bregman projection of $\widehat{w}^t$ into the convex set $\mathrm{conv}(\mathcal{C})$, which contains C. We now sum over trials and obtain, abbreviating $\ell^1+\ldots+\ell^T$ to $\ell^{\le T}$,

$$(1-\mathrm{e}^{-\eta})\sum_{t=1}^{T} w^{t-1}\cdot\ell^t \;\le\; \triangle(C\|w^0)-\triangle(C\|w^T)+\eta C\cdot\ell^{\le T}.$$

Recall that $w^{t-1}\cdot\ell^t$ equals the expected loss $\mathbb{E}[\mathbf{C}^t]\cdot\ell^t$ of CH in trial t. Also, relative entropies are nonnegative, so we may drop the second one, giving us the following bound on the total loss of the algorithm:

$$\sum_{t=1}^{T}\mathbb{E}[\mathbf{C}^t]\cdot\ell^t \;\le\; \frac{\triangle(C\|w^0)+\eta C\cdot\ell^{\le T}}{1-\mathrm{e}^{-\eta}}.$$

To proceed we have to expand the prior w^0. We consider the *symmetric balanced* case, i.e. where the concept class is invariant under permutation of the components, and every concept uses exactly U components. Paths may have different lengths and hence do not satisfy these requirements. All other examples from Table 1 do. In this balanced symmetric case we take w^0 to be the usage of the uniform distribution on concepts, satisfying $w_i^0 = U/d$ for each component i. It follows that $\triangle(C\|w^0) = U\ln(d/U)$, because any comparator C is a $0/1$ vector that also uses exactly U components.

Let $\ell^\star$ denote $\min_{C\in\mathcal{C}} C\cdot\ell^{\le T}$, the loss of the best concept in hindsight. Then by choosing $\eta = \sqrt{\frac{2U\ln(d/U)}{\ell^\star}}$ as a function of $\ell^\star$, we obtain the following general expected regret bound for CH:

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right]-\ell^\star \;\le\; \sqrt{2\ell^\star U\ln(d/U)}+U\ln(d/U). \tag{1}$$

The best-known general regret bounds for Expanded Hedge [FS97] and Follow the Perturbed Leader [HP05] are:

$$\mathbb{E}\left[\ell_{\mathrm{EH}}\right]-\ell^\star \;\le\; \sqrt{2\ell^\star U\ln D}+U\ln D \tag{2}$$

$$\mathbb{E}\left[\ell_{\mathrm{FPL}}\right]-\ell^\star \;\le\; \sqrt{4\ell^\star U d\ln d}+3Ud\ln d \tag{3}$$

where $D=|\mathcal{C}|$. Specific values for U, D and d in each application are listed in Table 1. We remark that if only an upper bound $\hat{\ell} \ge \ell^\star$ is available, then we can still tune η as a function of $\hat{\ell}$ to achieve these bounds with $\hat{\ell}$ under the square roots instead of $\ell^\star$. Moreover, standard heuristics can be used to tune η "online" when no good upper bound on $\ell^\star$ is given, which increase the expected regret bounds by at most a constant factor. (e.g. [CBFH$^+$97, HP05]).

We are not concerned with small multiplicative constants (e.g. 2 vs 4), but the gap between (1) and both (2) and (3) is significant. To compare, observe that $\ln D$ is of order $U\ln d$ in all our applications. Thus, the EH regret bound is worse by a factor $\sqrt{U}$, while FPL is worse by a bigger factor $\sqrt{d}$. Moreover, in Section 4 we show for the covered examples that our expected regret bound (1) for CH is optimal up to constant scaling.

Some concept classes have special structure that can be exploited to improve the regret bounds of FPL and EH down to that of CH. We consider one such property, called the *unit rule* in Section 5.

3 Applications

We consider the following structured concept classes: experts, k-sets, truncated permutations, source-sink paths, undirected and directed spanning trees. In each case we discuss implementation of CH and obtain a regret bound. Matching lower bounds are presented in Section 4.

3.1 Experts

The most basic example is the vanilla expert setting. In this case, the set of "structured" concepts equals the set of n standard basis vectors in $\mathbb{R}^n$. We will see that in this case Component Hedge see gracefully degrades to the original Hedge algorithm. First, the parameter spaces of both algorithms coincide since the convex

hull of the basis vectors equals the probability simplex. Second, the predictions coincide since a vector in the probability simplex decomposes uniquely into a convex combination of basis vectors. Third, the parameter updates are the same, since the relative entropy projection of a non-negative weight vector into the probability simplex amounts to re-normalizing to unity.

In fact on this simple task CH, EH and FPL each coincide with Hedge. For CH and EH this is obvious. For FPL this fact was observed in [KW05, Kal05] by using log-of-exponential perturbations instead of exponential perturbations used in the original paper [KV05]. Thus, we obtain following regret bound for all algorithms:

$$\mathbb{E}\left[\ell_{\text{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star \ln n} + \ln n.$$

3.2 k-sets

The problem of learning with sets of k out of n elements was introduced in [WK08] and applied to online Principal Component Analysis (PCA). Their algorithm is an instance of CH, and we review it here. The convex hull of k-sets equals the set of $w \in \mathbb{R}^n_+$ that satisfy the following constraints:

$$w_i \leq 1 \quad \text{for all } i \in [n] \qquad\qquad \text{and} \qquad\qquad \sum_{i=1}^{n} w_i = k. \tag{4}$$

Relative entropy projection into this polytope amounts to re-normalizing the sum to k, followed by redistributing the mass of the components that exceed 1 over the remaining components so that their ratios are preserved. Finally, each element of the convex hull of sets can be greedily decomposed into a convex combination of n k-sets by iteratively removing sets in the convex combination while always setting the coefficient of the new set as high as possible. Both projection and decomposition take $O(n^2)$ time [WK08].

Regret bound By (1), the regret of CH on sets is

$$\mathbb{E}\left[\ell_{\text{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star k \ln(n/k)} + k \ln(n/k).$$

We give a matching lower bound in Section 4.

3.3 Truncated permutations

The second instantiation of CH that has appeared is the problem of permutations [HW09]. Here we consider a slightly generalized task: *truncated permutations* of k out of n elements. A truncated permutation fills k slots with distinct elements from a pool of n elements. Equivalently, a truncated permutation is a maximal matching in the complete bipartite graph between $[k]$ and $[n]$. Truncated permutations extend k-sets by linearly ordering the selected k elements.

Results to search queries are usually in the form of a truncated permutation; of all n existing documents, only the top k are displayed in order of decreasing relevance. Predicting with truncated permutations is thus a model for learning the best search result.

Matching polytope We write $i \leftarrow j$ for the component that assigns item j to slot i. Now the convex hull of truncated permutations consists of all $w \in \mathbb{R}^{k \times n}_+$ (see [Sch03, Corollary 18.1b]) satisfying the following k *row* (left) and n *column* (right) constraints:

$$\sum_{j \in [n]} w_{i \leftarrow j} = 1 \quad \text{for all } i \in [k] \qquad\qquad \text{and} \qquad\qquad \sum_{i \in [k]} w_{i \leftarrow j} \leq 1 \quad \text{for all } j \in [n]. \tag{5}$$

Relative entropy projection The relative entropy projection of $\widehat{w}$ into the convex hull of truncated permutations $w = \operatorname{argmin}_{w \text{ s.t. } (5)} \triangle(w \| \widehat{w})$ has no closed form solution. By convex duality, $w_{i \leftarrow j} = \widehat{w}_{i \leftarrow j} \mathrm{e}^{-\lambda_i - \mu_j}$, where λ_i and μ_j are the Lagrange multipliers associated to the row and column constraints (5), which minimize

$$\sum_{i \in [k]\,;\, j \in [n]} \widehat{w}_{i \leftarrow j} \mathrm{e}^{-\lambda_i - \mu_j} + \sum_{i \in [k]} \lambda_i + \sum_{j \in [n]} \mu_j.$$

under the constraint that $\mu \geq \mathbf{0}$. This dual problem, which has $2n$ variables and n constraints, may be optimized directly using numerical convex optimization software. Another approach is to iteratively reestablish each violated constraint beginning from $\mu = \mathbf{0}$ and $\lambda = \mathbf{0}$. In full permutation case ($k = n$), this process is called *Sinkhorn balancing*. It is known to converge to the optimum, see [HW09] for an overview of efficiency and convergence results of this iterative method.

Decomposition Our decomposition algorithm for truncated permutations interpolates between the decomposition algorithms used for k-sets and full permutations [WK08, HW09]. Assume w lies in the hull of truncated permutations, i.e. the constraints (5) are satisfied. To measure progress, we define a score $s(w)$ as the number of zero components in w plus the number of column constraints that are satisfied with equality.

Our algorithm maintains a truncated permutation C that satisfies the following invariant: C hits all columns whose constraints are satisfied with equality by w, and avoids all components with weight zero in w. Such a C can be established in time $O(k^2 n)$ using augmenting path methods (see [Sch03, Theorem 16.3]).

Let l be the minimum weight of the components used by C, and let h be the maximum column sum of the columns untouched by C. So by construction $h < 1$. If $l = 1$ then $w = C$ and we are done. Otherwise, let $\alpha = \min\{l, 1 - h\}$, and set $w' = (w - \alpha C)/(1 - \alpha)$. It is easy to see that the vector w' satisfies (5), and that $s(w') > s(w)$. It is no longer the case that C satisfies the invariant w.r.t. w'. However, we may compute a weight k matching C' that satisfies the invariant by executing at most $s(w') - s(w)$ many augmenting path computations, which each cost $O(kn)$ time. We describe how this works below. After that we simply recurse on w' and C'. The resulting convex combination is αC plus $(1 - \alpha)$ times the result of the recursion.

The number of iterations is bounded by the score $s(w)$, which is at most kn. Thus, the total running time is $O(k^2 n^2)$.

We now show that C can be improved to C' satisfying the invariant by a single augmenting path computation per violated requirement. Let C^* be a size k matching satisfying the invariant for w'. Such a matching always exists because w' lies in the matching polytope. Let $j \in [n]$ be a problematic column, i.e. either C matches j to a row i but $w'_{i \leftarrow j} = 0$, or C does not match j while its column constraint is tight for w'. From j, alternately follow edges from C and C^*. Since C and C^* are both matchings, this can not lead to a cycle, so it must lead to a path. Since all rows are matched, this path must end at a column. The path can not end at a column whose constraint is forced in both C and C^*. So it must end at a column whose constraint is not tight. Incorporating this augmenting path into C corrects the violated requirement without creating any new violations.

Regret bound By (1), the regret of CH on truncated permutations is

$$\mathbb{E}\left[\ell_{\text{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star k \ln n} + k \ln n.$$

We obtain a matching lower bound in Section 4.

3.4 Paths

The online shortest path problem was considered by [TW03, KV05], and by various researchers in the bandit setting (see e.g. [CBL09, AHR08] and references therein). We develop expected regret bounds for CH for the "full information setting". Our regret bound improves the bounds given in [TW03, KV05] which have the additional range factors in the square root.

Consider the a directed graph on the set of nodes $[n] \cup \{s, t\}$. Each trial we have to play a walk from the source node s to the sink node t. As always, our loss is given by the sum of the losses of the edges that our walk traverses. Since each edge loss is nonnegative (it lies in $[0, 1]$ by assumption) it is never beneficial to visit a node more than once. Thus w.l.o.g. we restrict attention to paths.

As an example, consider the full directed graph on $[n] \cup \{s, t\}$. Paths of length $k + 1$ through this graph use k distinct internal nodes in order, and therefore are in 1-1 correspondence with truncated permutations of size k. Paths thus generalize truncated permutations by allowing all lengths simultaneously.

Unit flow polytope To implement CH efficiently, we have to succinctly describe the convex hull of paths. Unfortunately, we can not hope to write down linear constraints that capture the convex hull *exactly*. For if we could, then we could solve the *longest path* problem, which is known to be NP complete, by linear programming. Fortunately, there is a slight relaxation of the convex hull of paths that is describable by few constraints, namely the polytope of so-called unit flows. Even better, we will see that this relaxation does not hurt predictive performance at all.

A *unit flow* $w \in \mathbb{R}^d_+$ is described by the following constraints:

$$1 = \sum_{j \in [n]+t} w_{s,j} \qquad \text{and} \qquad \sum_{j \in [n]+s} w_{j,i} = \sum_{j \in [n]+t} w_{i,j} \quad \text{for each } i \in [n]. \tag{6}$$

We think of $w_{i,j}$ as describing the amount of flow from node i to j. The left constraint ensures that one unit of flow leaves the source s. The right constraint enforces that at internal nodes inflow equals outflow. It easily follows that one unit of flow enters the sink t.

The unit flow polytope is not bounded, but it has the right "bottom". Namely, the vertices of the unit flow polytope are the s-t paths, see [Sch03, Section 10.3]. The unit flow polytope is the Minkowski sum of the

convex hull of s-t paths and the conic hull (nonnegative linear combinations) of directed cycles. Moreover, each unit flow can be decomposed into at most d paths and cycles, by iterative greedy removal of a directed cycle or paths containing the edge of least non-zero weight in time $O(n^4)$.

Since the unit flow polytope does have polynomially many constraints, we may efficiently run CH on it. Each round, it produces a flow. We then decompose this flow into paths and cycles, and throw away the cycles. We then sample a path from the remaining convex combination of paths.

Relative entropy projection To run CH, we have to compute the relative entropy projection of an arbitrary vector in $\mathbb{R}^d_+$ into the flow polytope (6). This is a convex optimization problem in $d \approx n^2$ variables with constraints. By Slater's constraint condition, we have strong duality. So equivalently, we may solve the concave dual problem, which only has $n+1$ variables and is unconstrained. The dual problem can therefore be solved efficiently by numerical convex optimization software.

Say we want to find w, the relative entropy projection of $\widehat{w}$ into the flow polytope. Since each edge appears in exactly two constraints with opposite sign, the solution has the form $w_{i,j} = \widehat{w}_{i,j} \mathrm{e}^{\lambda_i - \lambda_j}$ for all $i, j \in [n] \cup \{s, t\}$, where λ_i is the Lagrange multiplier associated with node i (and $\lambda_t = 0$). The vector λ maximizes

$$\lambda_s - \sum_{i \neq t \,;\, j \neq s} \widehat{w}_{i,j} \mathrm{e}^{\lambda_i - \lambda_j}$$

That is, we have to find a single scale factor e^{λ_i} for each node i, such that scaling each edge weight by the ratio of the factors of its nodes reestablishes the flow constraints (6).

We propose the following iterative algorithm. Start with all λ_i equal to zero. Then pick a violated constraint, say at node i, and reestablish it by changing its associated λ_i. That is, we execute either

$$\mathrm{e}^{\lambda_s} \leftarrow \frac{1}{\sum_{j \in [n]+t} \widehat{w}_{s,j} \mathrm{e}^{-\lambda_j}} \qquad \text{or} \qquad \mathrm{e}^{\lambda_i} \leftarrow \sqrt{\sum_{j \in [n]+s} \widehat{w}_{j,i} \mathrm{e}^{\lambda_j} \Big/ \sum_{j \in [n]+t} \widehat{w}_{i,j} \mathrm{e}^{-\lambda_j}} \qquad \text{for some } i \in [n].$$

In our experiments, this algorithm converges quickly. We leave its thorough analysis as an open problem.

Decomposition Find any s-t path with non-zero weights on all edges in time $O(n^2)$. Subtract that path, scaled by its minimum edge weight. This creates a new zero, maintains flow balance, and reduces the source's outflow. After at most n^2 iterations the source has outflow zero. Discard the remaining conic combination of directed cycles. The total running time is $O(n^4)$.

Regret bound for the complete directed graph Since paths have different lengths, we aim for a regret bound that depends on the length of the comparator path. To get such a bound, we need a prior usage vector w^0 that favors shorter paths. To this end, consider the distribution $\mathbb{P}$ that distributes weight 2^{-k} uniformly over all paths of length $k \leq n$, and assigns weight 2^{-n} to the paths of length $n+1$. This assures that $\mathbb{P}$ is normalized to 1. Since there are $n!/(n-k+1)!$ paths of length k, the probability of a path P of length k equals

$$\mathbb{P}(\mathbf{P} = P) \;=\; \frac{(n-k+1)!}{2^k n!} \qquad \text{if } k \leq n \text{ and} \qquad \mathbb{P}(\mathbf{P} = P) \;=\; \frac{1}{2^n n!} \qquad \text{if } k = n+1.$$

Also, the expected path length $\mathbb{E}[\mathbf{P} \cdot \mathbf{1}]$ is $2 - 2^{-n}$. We now set $w^0 \coloneqq \mathbb{E}[\mathbf{P}]$, i.e. the usage of $\mathbb{P}$. There are three kinds of edges. We have one direct edge s, t, we have $2n$ boundary edges of the form s, j or i, t, and we have $n(n-1)$ internal edges of the type i, j. A simple computation shows that their usages are (for $n \geq 3$)

$$w^0_{s,t} = \frac{1}{2}, \qquad\qquad w^0_{s,j} \;,\; w^0_{i,t} = \frac{1}{2n}, \qquad\qquad w^0_{i,j} = \frac{1 - 2^{-(n-1)}}{2n(n-1)}.$$

Let P be a comparator path of length k. If $k = 1$ then $\triangle(P \| w^0) = \ln 2$. Otherwise, still for $n \geq 3$,

$$\begin{aligned}
\triangle(P \| w^0) \;&=\; -2 \ln \frac{1}{2n} - (k-2) \ln \frac{1 - 2^{-(n-1)}}{2n(n-1)} + \mathbb{E}[\mathbf{P} \cdot \mathbf{1}] - k \\
&=\; (k-2) \ln\bigl(2n(n-1)\bigr) + 2 \ln 2n + (k-2) \ln \left(1 + \frac{2^{-(n-1)}}{1 - 2^{-(n-1)}}\right) - 2^{-n} - (k-2) \\
&\leq\; k \ln 2 - (k-2) \frac{1 - 2^{-n+2}}{1 - 2^{-n+1}} + 2(k-1) \ln n \;\leq\; 2k \ln n.
\end{aligned}$$

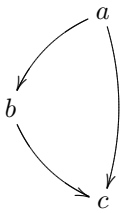
By tuning η as before, the regret of CH with prior w^0 w.r.t. a comparator path of length k is

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{4 \ell^\star k \ln n} + 2k \ln n.$$

This new regret bound improves known results in two ways. First, it does not have the range factors, which in the case of paths usually turn out to be the diameter of the graph, i.e. the length of the longest s-t path. Second, some previous bounds only hold for acyclic graphs. Our bound holds for the complete graph.

Figure 1: EH is not CH on paths

(a) Graph



(b) Usages after update $(1/3, 1, 1)$

Case	ab, bc	ac
EH and CH prior	$1/2$	$1/2$
EH after update	$1/4$	$3/4$
CH after update	$1/3$	$2/3$

(c) Usages after update $(1/2, 1, 1)$

Case	ab, bc	ac
EH and CH prior	$1/2$	$1/2$
EH after update	$1/3$	$2/3$
CH after update	$\frac{\sqrt{17}-1}{8}$	$\frac{9-\sqrt{17}}{8}$

Regret bound for an arbitrary graph We discussed the full graph as a first application of CH. For prediction on an arbitrary graphs we simply design a prior w^0 with zero usage on all edges that are not present in the graph. We could either use graph-specific knowledge, or we could use our old w^0, disable edges by setting their usage to zero, and project back into the flow polytope. Relative entropy projection never revives zeroed edges. The regret bound now obviously depends on the graph via the prior usage w^0.

3.4.1 Expanded Hedge and Component Hedge are different on paths

An efficient dynamic programming-based algorithm for EH was presented in [TW03]. This algorithm keeps one weight per edge, just like CH. These weights are updated using the *weight pushing algorithm*. This algorithm performs relative entropy projection on full distributions on paths. Like CH, weight pushing finds a weight of each node, and scales each edge weight by the ratio of its nodes weights. We now show that CH and EH are different on graphs. Consider the graph shown in Figure 1a. Say we use prior $\mathbb{P}$ with weight $1/2$ on both paths (a, b, c) and (a, c). Then the usages are $(1/2, 1/2, 1/2)$ for (ab, bc, ac). Now multiply edge ab by $1/3$ (that is, we give it loss $\ln 3$), and both other edges by 1 (we give them loss zero). The resulting usages of EH and CH are displayed in Table 1b. The usages are different, and hence, so are the expected losses. In most cases (as shown e.g. in Table 1c), the updated usages of CH are irrational while the prior usages and the scale factors of the update are rational. On the other hand, EH always maintains rationality.

3.5 Spanning trees

Whereas paths connect the source to the sink, spanning trees connect every node to every other node. Undirected spanning trees are often used in network-level communication protocols. For example, the *Spanning Tree Protocol* (IEEE 802.1D) is used by mesh networks of Ethernet switches to agree on a single undirected spanning tree, and thus eliminate loops by disabling redundant links. Directed spanning trees are used for asymmetric communication, for example for streaming multimedia from a central server to all connected clients. In either case, the cost of a spanning tree is the sum of the costs of its edges.

Learning spanning trees was pioneered by [KGCC07] for learning dependency parse trees. They discuss efficient methods for parameter estimation under log-loss and hinge loss. [CBL09] derive a regret bound for undirected spanning trees in the bandit setting. We instantiate CH to both directed and undirected trees and give the first regret bound without the range factor.

Three kinds of directed spanning trees are common. Spanning trees with a fixed root, spanning trees with a single arbitrary root, and arborescences (or spanning forests) with multiple roots. We focus on a fixed root. The other two models can be simulated by a fixed root. To simulate arborescences, add a dummy as the fixed root, and put the root selection cost of node i along the path from the dummy to i. Furthermore, to force a single root, increase the cost of all edges leaving the dummy by a fixed huge amount.

Tree polytope To characterize the convex hull of directed trees on n nodes with fixed root 1, we use a trick based on flows from [MW95] that makes use of auxiliary variables $f^k_{i,j}$:

$$0 \le f^k_{i,j} \le w_{i,j}, \quad \sum_{i,j} w_{i,j} = n-1, \quad \underbrace{\sum_{j\neq i} f^k_{j,i}}_{k\text{-flow into } i} + \underbrace{\mathbf{1}_{i=1}}_{k\text{-source at 1}} = \underbrace{\sum_{j\neq i} f^k_{i,j}}_{k\text{-flow out of } i} + \underbrace{\mathbf{1}_{i=k}}_{k\text{-sink at } k}, \quad \text{for } i,j,k \in [n]. \tag{7}$$

The intuition is as follows. A tree has $n-1$ edges, and every node can be reached from the root. We enforce this by having a separate flow channel f^k for each non-root node k. We place a unit of flow into this channel at the root. Each intermediate node satisfies flow equilibrium. Finally, the target node k consumes the unit of flow destined for it. The first equation ensures that each edge's usage is sufficient for the flow that traverses that edge. The undirected tree polytope is constructed based on the directed tree polytope by considering

the above $w_{i,j}$ as auxiliary variables, an imposing the constraint $w_{i,j} + w_{j,i} = v_{i,j}$. Now v are the weights sought.

Relative entropy projection The relative entropy projection of $\widehat{w}$ into the convex hull of directed spanning trees $w = \operatorname{argmin}_{w \text{ s.t. } (7)} \triangle(w\|\widehat{w})$ has no closed form solution. By convex duality, the solution satisfies

$$w_{i,j} \;=\; (n-1)\,\frac{\widehat{w}_{i,j}\mathrm{e}^{\sum_{k\neq 1}\max\{0,\mu_j^k-\mu_i^k\}}}{\sum_{i,j\neq i}\widehat{w}_{i,j}\mathrm{e}^{\sum_{k\neq 1}\max\{0,\mu_j^k-\mu_i^k\}}}, \qquad\qquad f_{ij}^k = \begin{cases} w_{i,j} & \text{if } \mu_j^k > \mu_i^k, \\ 0 & \text{if } \mu_j^k < \mu_i^k, \end{cases}$$

where μ_i^k, the Lagrange multipliers associated to the flow balance constraints, maximize

$$\sum_{k\neq 1}\left(\mu_k^k - \mu_1^k\right) - (n-1)\ln\Bigg(\sum_{i,j\neq i}\widehat{w}_{i,j}\mathrm{e}^{\sum_{k\neq 1}\max\{0,\mu_j^k-\mu_i^k\}}\Bigg).$$

This unconstrained concave maximization problem in $\approx n^2$ variables seems easier than the primal problem, which has $\approx n^3$ variables and constraints. Note however that the objective is not differentiable everywhere. Alternatively, we may again proceed by iteratively reestablishing constraints locally, starting from some initial assignment to the dual variables μ. This approach is analogous to Sinkhorn balancing.

Decomposition We have no special-purpose tree decomposition algorithm, and therefore resort to a general decomposition algorithm for convex polytopes that is based on linear programming. Let w be in the tree polytope. Choose an arbitrary vertex C (i.e. a spanning tree) by minimizing a linear objective over the current polytope. Now use linear programming to find the furthest point w' in the polytope on the ray from C through w. At least one more inequality constraint is tight for w'. Thus w' lies in a convex polytope of at least one dimension lower. Add this inequality constraint as an equality constraint, recursively decompose w', and express w as a convex combination of C and the decomposition of w'. The recursion bottoms out at a vertex (i.e. a spanning tree) and the total number of iterations is at most d.

Regret bound By (1), the regret $\mathbb{E}\left[\ell_{\text{CH}}\right] - \ell^\star$ of CH on undirected and directed spanning trees is at most

$$\sqrt{2\ell^\star(n-1)\ln(n/2)} + (n-1)\ln(n/2) \qquad\qquad \sqrt{2\ell^\star(n-1)\ln(n-1)} + (n-1)\ln(n-1)$$

We provide matching lower bounds in Section 4.

4 Lower bounds

Whereas it is easy to get some regret bounds with additional range factors, we show that CH is essentially optimal in all our applications. We leverage the following lower bound for the vanilla expert case:

Theorem 1 *There are positive constants c_1 and c_2 s.t. any online algorithm for q experts with loss range $[0, U]$ can be forced to have expected regret at least*

$$c_1\sqrt{\ell^\star U\ln q} + c_2\ln q. \tag{8}$$

This type of bound was recently proven in [AWY]. Note that c_1 and c_2 are independent of the number of experts, the range of the losses and the algorithm. Earlier versions of the above lower bound using many quantifier and limit arguments are given in [CBFH$^+$97, HW09]. We now prove lower bounds for our structured concept classes by embedding the original expert problem into each class and applying the above theorem. This type of reduction was pioneered in [HW09] for permutations.

The general reduction works as follows. We identify q structured concepts $C_1, \ldots, C_q$ in the concept class $\mathcal{C} \subseteq \{0,1\}^d$ to be learned that partition the d components. Now assume we have an online algorithm for learning class $\mathcal{C}$. From this we construct an algorithm for learning with q experts with loss range $[0, U]$. Let $\ell \in [0, U]^q$ denote the loss vector for the expert setting. From this we construct a loss vector $L \in [0,1]^d$ for learning $\mathcal{C}$: $L := \sum_{i=}^q \frac{\ell_i}{U} C_i$. That is, we spread the loss of expert i, evenly among the U many components used by concept C_i. Second, we transform the predictions as follows. Say our algorithm for learning $\mathcal{C}$ predicts with any structured concept $C \in \mathcal{C}$. Then we play expert i with probability $C_i \cdot C/U$. The expected loss of the expert algorithm now equals the transformed loss of the algorithm for learning concepts in $\mathcal{C}$:

$$\mathbb{E}[\ell_{\mathbf{i}}] \;=\; \sum_{i=1}^q \frac{C_i \cdot C}{U}\ell_i \;=\; C \cdot \sum_{i=1}^q \frac{\ell_i}{U} C_i \;=\; C \cdot L$$

This also means that the expected loss of the expert algorithm equals the expected loss of the algorithm for learning the structured class. This implies that the expected regret of the algorithm for learning $\mathcal{C}$ is at least the expected regret of the expert algorithm. The lower bound (8) for the regret in the expert setting is thus also a lower bound for the regret of the structured prediction task.

k-sets We assume that k divides n. Then we can partition $[d]$ with n/k sets, where set i uses components $(i-1)k+1,\ldots,ik$. The resulting lower bound has leading factor $\sqrt{k \ln \frac{n}{k}}$, matching the upper bound for CH within constant factors.

Truncated permutations We can partition the n^2 assignments into n full permutations. For example, the n cyclic shifts of the identity permutation achieve this. The truncations to length k of those n permutations partition the kn components in the truncated case. The lower bound with leading factor $\sqrt{k \ln n}$ again matches the regret bound of CH within constant factors.

Spanning trees As observed in [Gus83], the complete undirected graph has $(n-1)/2$ edge-disjoint spanning trees. Hence we get a lower bound with leading factor $\sqrt{(n-1)\ln((n-1)/2)}$. Each undirected spanning tree can be made directed by fixing a root. So there are at least as many disjoint directed spanning trees with a fixed root. In both cases we match the regret of CH within a constant factor.

Paths Consider the directed graph on $[n] \cup s,t$ that has n/k disjoint s-t paths of length $k+1$ connecting source to sink. By construction, we can embed n/k experts with loss range $[0,k]$ into this graph, so the regret has leading factor at least $\sqrt{k \log(n/k)}$. This graph is a subgraph of the complete directed graph $s \to K_n \to t$. Moreover, nature can force the algorithm to essentially play on the disjoint path graph by giving all edges outside it sheer infinite loss in a sheer infinite number of trials. This shows that the regret w.r.t. a comparator path of length k through the full graph has leading factor at least $\sqrt{k \log(n/k)}$.

A lower bound on the regret for arbitrary graphs is difficult to obtain since various interesting problems can be encoded as path problems. For example, the expert problem where each expert has a different loss range can be encoded into a graph that has a disjoint path of each length $1, 2, \ldots n$. The optimal algorithm for such expert problems was recently found in [AW], but its regret has no closed form expression. It might be that the regret of CH is tight within constant factors for all graphs, but this question remains open.

5 Comparison to other algorithms

CH is a new member of an existing ecosystem. Other algorithms for structured prediction are EH[LW94] and FPL [KV05]. We now compare them.

Efficiency FPL can be readily applied efficiently to our examples of structured concept classes: k-sets take $O(n)$ per trial using variants of median-finding, truncated permutations take $O(k^2 n)$ per trial using the Hungarian method for minimum weight bipartite matching, paths take $O(n^2)$ per trial using Dijkstra's shortest path algorithm and spanning trees take $O(n^2)$ per trial using either Prim's algorithm or Chu–Liu/Edmonds's algorithm for finding a minimum weight spanning tree.

EH can be efficiently implemented for k-sets [WK08] and paths [TW03] using dynamic programming, and for spanning trees [KGCC07] using the Matrix-Tree Theorem by Kirchoff (undirected) and Tutte (directed). An approximate implementation based on MCMC sampling could be built for permutations based upon [JSV04].

In most cases FPL and EH are faster than CH. This may be partly due to the novelty of CH and the lack of special-purpose algorithms for it. On the other hand, FPL solves a linear minimization problem, which is intuitively simpler than minimizing a convex relative entropy.

5.1 Improved regret bounds with the unit rule

On the other hand, we saw in Section 2.2 that the general regret bound for CH (1) improves the guarantees of EH (1) by a factor $\sqrt{U}$ and those of FPL (3) by a larger factor $\sqrt{d}$. It is an open question whether these factors are real or simply an artifact of the bounding technique (see Section 6). We now give an example of a property of structured concept classes that makes these range factors vanish.

We say that a prediction algorithm has the *unit rule* on a given structured concept class $\mathcal{C}$ if its worst-case performance is achieved when in each trial only a single expert has nonzero loss. Without changing the prediction algorithm, the unit rule immediately improves its regret bound by reducing the effective loss range of each concept from $[0,U]$ to $[0,1]$. The improved regret bounds are (c.f. (2) and (3))

$$\mathbb{E}\left[\ell_{\text{EH}}\right] \le \ell^\star + \sqrt{2\ell^\star \ln D} + \ln D \tag{9}$$

$$\mathbb{E}\left[\ell_{\text{FPL}}\right] \le \ell^\star + \sqrt{4\ell^\star U \ln d} + 3U \ln d \tag{10}$$

The unit rules for EH and FPL on experts have been observed before [KV05, AWY08]. We reprove them here for completeness. The unit rule holds for both EH and FPL on sets, and for EH on undirected trees. It fails for EH and FPL on permutations, and for EH on directed trees.

We prove the unit rule for EH on sets here, and counter it for EH on directed trees. Proofs and counterexamples for the other cases are similar, and omitted for lack of space.

5.1.1 Unit rule holds for EH on k-sets

Fix an expert i, and let j be an arbitrary other expert. We claim that if we hand out loss to i, then the usage of j increases. For each k-set S, we denote the prior weight of S by W_S. We abbreviate

$$Z_i \coloneqq \sum_{S:i\in S} W_S, \qquad Z_{\neg i} \coloneqq \sum_{S:i\notin S} W_S, \qquad Z_j \coloneqq \sum_{S:j\in S} W_S, \qquad Z_{\neg j} \coloneqq \sum_{S:j\notin S} W_S,$$
$$Z_{i\wedge j} \coloneqq \sum_{S:i\in S,j\in S} W_S, \quad Z_{\neg i\wedge j} \coloneqq \sum_{S:i\notin S,j\in S} W_S, \quad Z_{i\wedge\neg j} \coloneqq \sum_{S:i\in S,j\notin S} W_S, \quad Z_{\neg i\wedge\neg j} \coloneqq \sum_{S:i\notin S,j\notin S} W_S.$$

Theorem 2 *Assume that the prior weights have product structure, i.e. $W_S \propto \prod_{i\in S} w_i$. Then*

$$Z_j = \mathbb{P}(j\in\mathbf{S}^1) \;\le\; \mathbb{P}(j\in\mathbf{S}^2|\ell^1=\delta_i) = \frac{Z_{i\wedge j}\mathrm{e}^{-\eta}+Z_{\neg i\wedge j}}{Z_i\mathrm{e}^{-\eta}+Z_{\neg i}}.$$

Proof: With some rewriting, the claim is equivalent to

$$Z_iZ_j \;\ge\; Z_{i\wedge j} \qquad \text{and also} \qquad Z_{i\wedge\neg j}Z_{\neg i\wedge j} \;\ge\; Z_{i\wedge j}Z_{\neg i\wedge\neg j}$$

Define

$$R(n,k) \coloneqq \sum_{\substack{S\subseteq[n]\\|S|=k}} \prod_{i\in S} w_i.$$

We now show that $R(n,k+1)R(n,m) \ge R(n,k)R(n,m+1)$ for all $0\le k<m<n$. The proof proceeds by induction on n. The case $n=0$ is trivial. Now suppose that the claim holds up to n. We need to show it for $n+1$. For $n>0$, we have

$$R(n,k) \;=\; \mathbf{1}_{k>0}w_nR(n-1,k-1)+\mathbf{1}_{k<n}R(n-1,k). \tag{11}$$

Suppose that the induction hypothesis holds up to n. We must show that for all $0\le k<m<n+1$

$$R(n+1,k+1)R(n+1,m) \ge R(n+1,k)R(n+1,m+1).$$

By (11), this is equivalent to

$$\begin{aligned}(w_{n+1}R(n,k)+\mathbf{1}_{k<n}R(n,k+1))\,(\mathbf{1}_{m>0}w_{n+1}R(n,m-1)+\mathbf{1}_{m\le n}R(n,m)) \;\ge\\ (\mathbf{1}_{k>0}w_{n+1}R(n,k-1)+\mathbf{1}_{k\le n}R(n,k))\,(\mathbf{1}_{m+1>0}w_{n+1}R(n,m)+\mathbf{1}_{m<n}R(n,m+1))\end{aligned}$$

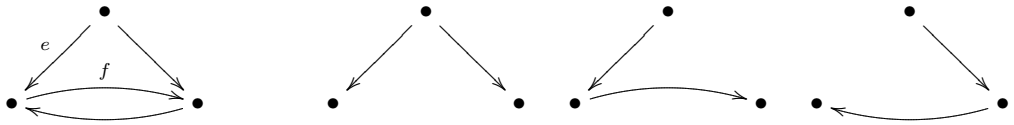
Now we expand, and use $0\le k<m<n+1$ to eliminate indicators. It remains to show

$$\begin{pmatrix}(w_{n+1})^2R(n,k)R(n,m-1) \;+\\ w_{n+1}R(n,k)R(n,m) \;+\\ w_{n+1}R(n,k+1)R(n,m-1) \;+\\ R(n,k+1)R(n,m)\end{pmatrix} \;\ge\; \begin{pmatrix}\mathbf{1}_{k>0}(w_{n+1})^2R(n,k-1)R(n,m) \;+\\ \mathbf{1}_{k>0}\mathbf{1}_{m<n}w_{n+1}R(n,k-1)R(n,m+1) \;+\\ w_{n+1}R(n,k)R(n,m) \;+\\ \mathbf{1}_{m<n}R(n,k)R(n,m+1)\end{pmatrix}$$

We now show that this inequality holds line-wise. Lines with active indicators trivially hold. If $k-1=m$, the second line holds with equality. Otherwise, and for the other lines we use the induction hypothesis. ■

5.1.2 Unit rule fails for EH on directed trees

The unit rule is violated for EH on directed trees. Consider this graph (left) and its three directed spanning trees (right):



Note that we may always restrict attention to a given graph G by assigning zero prior weight to all spanning trees of the full graph that use edges outside G. Now if we put a unit of loss on edge e, the usage of f decreases, and vice versa, contradicting the unit rule. Call the prior weights on directed trees W_A, W_B, W_C. Then the usages satisfy

$$W_A+W_B = \mathbb{P}(e\in\mathbf{T}^1) \;\ge\; \mathbb{P}(e\in\mathbf{T}^2|\ell^1=\delta_f) = \frac{W_A+W_B\mathrm{e}^{-\eta}}{W_A+W_B\mathrm{e}^{-\eta}+W_C},$$
$$W_B = \mathbb{P}(f\in\mathbf{T}^1) \;\ge\; \mathbb{P}(f\in\mathbf{T}^2|\ell^1=\delta_e) = \frac{W_B\mathrm{e}^{-\eta}}{W_A\mathrm{e}^{-\eta}+W_B\mathrm{e}^{-\eta}+W_C}.$$

6 Conclusion

We developed the Component Hedge algorithm for online prediction over structured expert classes. The advantage of CH is that it has a general regret bound without the range factors that typically plague EH and FPL. We considered several example concept classes, and showed that the lower bound is matched in each case.

Open problems While the unit rule is one method for proving regret bounds for EH and FPL that are close to optimum, there might be other proof methods that show that EH and FPL perform as well as CH when applied to structured concepts. We know of no examples of structured concept classes where EH and FPL are clearly suboptimal. Resolving the question of whether such examples exist is our main open problem.

The prediction task for each structured concept class can be analyzed as a two-player zero-sum game versus nature which tries to maximize the regret. The paper [AWY08] gave an efficient implementation of the minimax optimal algorithm for playing against an adversary in the vanilla expert setting. Actually, the key insight was that the unit rule holds for the optimal algorithm in the vanilla expert case. This fact made it possible to design a balanced algorithm that incurs the same loss no matter which sequence of unit losses is chosen by nature. Unfortunately, the optimum algorithm does not satisfy the unit rule for any of the structured concept classes considered here. However, there might be some sort of relaxation of the unit rule that still leads to an efficient implementation of the optimum algorithm.

In this paper the loss of a structured concept C always had the form $C \cdot \ell$, where ℓ is the loss vector for the components. This allowed us to maintain a mixture of concepts w and predict with a random concept $\mathbf{C}$ s.t. $\mathbb{E}[\mathbf{C}] = w$. By linearity, the expected loss of such a randomly drawn concept $\mathbf{C}$ is the same as the loss of the mixture w. For regression problems with for example the convex loss $(C \cdot \ell - y)^2$ our algorithm can still maintain a mixture w, but now the expected loss of $\mathbf{C}$, i.e. $\mathbb{E}[(\mathbf{C} \cdot \ell - y)^2]$, is typically larger than the loss $(w \cdot \ell - y)^2$ of the mixture. We are confident that in this more general setting we can still get good regret bounds compared to the best mixture chosen in hind-sight. All we need to do is replace CH with the more general "Component Exponentiated Gradient" algorithm, which would do an EG update on the parameter vector w and project the updated vector back into the hull of the concepts.

In general, we believe that we have a versatile method of learning with structured concept classes. For example it is easy to augment the updates with a "share update" [HW98, BW02] for the purpose of making them robust against sequences of examples where the best comparator changes over time. We also believe that our methods will get rid of the additional range factors in the bandit setting [CBL09] and that gain versions of the algorithm CH also have good regret bounds.

At the core of our methods lies a relative entropy regularization which results in a multiplicative update on the components. In general, which relative entropy to choose is always one of the deepest questions. For example in the case of learning k-sets, a sum of binary relative entropies over the component can be used that incorporates the $w_i \leq 1$ constraints into the relative entropy term. In general incorporating inequality constraints into the relative entropy seems to have many advantages. However how to do this is an open ended research question.

References

[AHR08] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *In Proceedings of the 21st Annual Conference on Learning Theory (COLT*, 2008.

[AW] Jacob Abernethy and Manfred K. Warmuth. Repeated games against budgeted adversaries. Unpublished manuscript.

[AWY] Jake Abernethy, Manfred K. Warmuth, and Joel Yellin. When random play is optimal against an adversary. Journal version of [AWY08], in progress.

[AWY08] Jacob Abernethy, Manfred K. Warmuth, and Joel Yellin. Optimal strategies for random walks. In *Proceedings of The 21st Annual Conference on Learning Theory*, pages 437–446, July 2008.

[BW02] Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

[CBFH+97] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

[CBL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[CBL09] Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[Gus83] Dan Gusfield. Connectivity and edge-disjoint spanning trees. *Information Processing Letters*, 16(2):87–89, 1983.

[HKW98] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

[HP05] Marcus Hutter and Jan Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, April 2005.

[HW98] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[HW01] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[HW09] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, July 2009.

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[Kal05] Adam Kalai. A perturbation that makes "Follow the Leader" equivalent to "Randomized Weighted Majority". Private communication, December 2005.

[KGCC07] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the Matrix-Tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.

[KV05] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[KW05] Dima Kuzmin and Manfred K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, pages 684–686. Springer-Verlag, June 2005. Open problem.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994. Preliminary version appeared in the Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, 1989.

[MW95] Thomas L. Magnanti and Laurence A. Wolsey. Optimal trees. In M. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503–615. North-Holland, 1995.

[Sch03] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

[TW03] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.

[WGV08] Manfred K. Warmuth, Karen Glocer, and S.V.N. Vishwanathan. Entropy regularized LPBoost. In Yoav Freund, László Györfi, György Turán, and Thomas Zeugmann, editors, *Proceedings of the 19th International Conference on Algorithmic Learning Theory (ALT '08)*, pages 256–271. Springer-Verlag, October 2008.

[WK08] Manfred K. Warmuth and Dima Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, October 2008.