Online Dynamic Programming

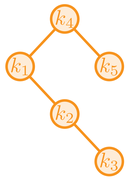
Holakou Rahmanian Manfred K. Warmuth

An Example of the Prediction Game

Fix a set of n ordered keys $k_1 < \ldots < k_n$.
In each of the T trials

1. The learner randomly predicts with a *Binary Search Tree (BST)*
2. The adversary reveals the *search probabilities* for all the keys $\boldsymbol{p} \in [0,1]^n$ s.t. $\sum_i p_i = 1$
3. The learner incurs a loss of *average search cost* of the predicted BST i.e. $\mathbb{E}\left[\sum_{i=1}^n \text{depth}(k_i) \cdot p_i\right]$

$n = 5, \boldsymbol{p} = (.1, .05, .05, .5, .3)$



average search cost

$1 \times .5$

$+2 \times .1 + 2 \times .3$

$+3 \times .05$

$+4 \times .05 = 1.65$

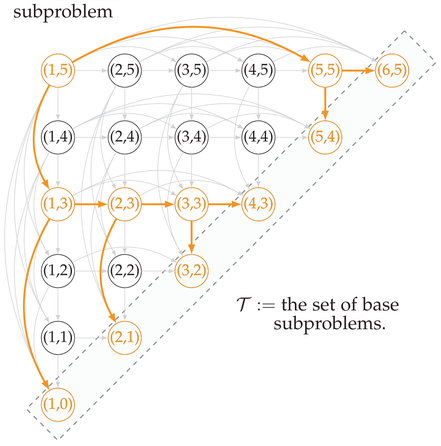
The Dynamic Programming Representation

DAG of Subproblems

- *Vertices V encode the subproblems*
- *Edges E encode the dependency between subproblems:* (i,j) to $(i, r-1)$ and $(r+1, j)$ for $i \le r \le j$

$$\overbrace{\text{OPT}(i,j)}^{\text{The subproblem with } k_i \ldots k_j} = \begin{cases} 0 & j = i-1 \\ \min_{i \le r \le j} \{\text{OPT}(i, r-1) & \\ \qquad + \text{OPT}(r+1, j) & \\ \qquad + \sum_{k=i}^{j} p_k\} & i \le j \end{cases}$$

$s :=$ the final subproblem



$\mathcal{T} :=$ the set of base subproblems.

Multiedges encode the recursive calls

- A **multiedge** is the pair of edges of form (i,j) to $(i, r-1)$ and $(r+1, j)$
- The associated loss with any outgoing multiedge m from (i,j) is defined as $\ell_m = \sum_{k=i}^{j} p_k$

Multipaths encode the solutions

- A **multipath** starts with a single multiedge at s, continues with successor multiedges at the internal nodes, and ends at $\mathcal{T}$

Benefits of multipath representation:

- *Every BST can be represented by a multipath and vice versa*
- *Linear loss over the components (i.e. multiedges)*
- *Works for a large class of min-sum dynamic programmings ($M_v :=$ the set of outgoing multiedges from vertex v)*

$$\text{OPT}(v) = \min_{m \in M_v} \left\{ \ell_m + \sum_{u:(v,u) \in m} \text{OPT}(u) \right\}$$

where for all vertices $v \in V$

1. $|M_v| = k$ *for a fixed* k
2. *m's in M_v do not overlap*

Algorithm #1 – *Expanded Hedge (EH)*

Maintain a **distribution** W on all multipaths. With each **multiedge** m, associate a weight w_m. Exploiting the linear loss, let W be in *canonical product form* so that the multiplicative update factorizes over multiedges and sampling becomes easy:

Weight Space – Canonical Product Form

1. $W(\boldsymbol{\pi}) = \prod_{m \in \boldsymbol{\pi}} w_m$
2. $\sum_{m \in M_v} w_m = 1$ for all $v \in V - \mathcal{T}$
3. $\sum_{\boldsymbol{\pi}} W(\boldsymbol{\pi}) = 1$

Prediction Sample from W using 1 and 2 above

Multiplicative update for each multipath $\boldsymbol{\pi}$ results in multiplicative updates per multiedge m:

Update
$$W^{\text{new}}(\boldsymbol{\pi}) \propto W(\boldsymbol{\pi}) \exp(-\eta\, \boldsymbol{\pi} \cdot \boldsymbol{\ell}) = \prod_{m \in \boldsymbol{\pi}} \underbrace{w_m \exp\left[-\eta\, \ell_m\right]}_{:= \widehat{w}_m}$$

Going back to the canonical product form:

Projection – Generalized Weight Pushing
Goal: For each multiedge m, find factor f_m s.t. the new weights $w_m^{\text{new}} := \widehat{w}_m f_m$ are in canonical product form.

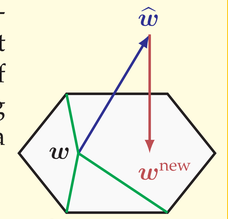
Solution:
1. $Z_v := 1, \quad \forall v \in \mathcal{T}$
2. $Z_v := \sum_{m \in M_v} \widehat{w}_m \prod_{u:(v,u) \in m} Z_u, \ \forall v \in V - \mathcal{T}$
3. $f_m := \left(\prod_i Z_{u_i}\right) / Z_v$, for each multiedge m from v to u_1 and u_2

Regret Bounds
$$\mathbb{E}[L_{\text{EH}}] - L_{\text{best}} \le \sqrt{2 \log(4)}\, n^{\frac{3}{2}} \sqrt{T} + n^2 \log(4)$$

Algorithm #2 – *Component Hedge (CH)*

We can also work with **edges** as components. With each edge e, associate a weight w_e. Arbitrarily distribute the loss ℓ_m of each multiedge m over its edges. Having a linear loss, it is sufficient to maintain a **mean vector** $\boldsymbol{w}$ of multipaths:

$$\mathbb{E}[\boldsymbol{\pi} \cdot \boldsymbol{\ell}] = \mathbb{E}[\boldsymbol{\pi}] \cdot \boldsymbol{\ell} = \boldsymbol{w} \cdot \boldsymbol{\ell}$$



$\boldsymbol{w}$ will be in a polytope with a small number of facets:

Weight Space – 2-Flow Polytope

1. $w_{\text{out}}(s) = 2$
2. For any multiedge $m = \{e, e'\}$, $w_e = w_{e'}$
3. $w_{\text{out}}(v) = 2 \times w_{\text{in}}(v)$, for each vertex $v \in V - \mathcal{T} - \{s\}$

We sample with the same expectation as $\boldsymbol{w}$:

Prediction Decompose $\boldsymbol{w}$ into a convex combination of multipaths. Decomposition is done iteratively via a greedy approach which removes one multipath at a time.

Multiplicative update per edge:

Update For each edge $e \in E$, $\widehat{w}_e = w_e \exp(-\eta \ell_e)$

Going back to the polytope with relative entropy projection:

Projection – Iterative Bregman Projections
Goal: Find $\boldsymbol{w}^{\text{new}} := \underset{\boldsymbol{w} \in \text{2-flow polytope}}{\arg\min} \Delta(\boldsymbol{w} || \widehat{\boldsymbol{w}})$.
Solution: Project onto the flow constraint of a particular vertex and then repeatedly cycle through the vertices

Regret Bounds
$$\mathbb{E}[L_{\text{CH}}] - L_{\text{best}} \le 4\sqrt{2}\, n\, (\log n)^{\frac{1}{2}} \sqrt{T} + 16\, n \log n$$

Other Combinatorial Objects

- *CH has better bounds, but it is hard to obtain the projections.*

Problem	FPL	EH	CH
Optimal Binary Search Trees	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n\, (\log n)^{\frac{1}{2}} \sqrt{T})$
Matrix-Chain Multiplications	—	$\mathcal{O}(n^{\frac{3}{2}}\, D^3 \sqrt{T})$	$\mathcal{O}(n\, (\log n)^{\frac{1}{2}}\, D^3 \sqrt{T})$
Knapsack	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n\, (\log nC)^{\frac{1}{2}} \sqrt{T})$
Rod Cutting	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n\, (\log n)^{\frac{1}{2}} \sqrt{T})$
Weighted Interval Scheduling	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n^{\frac{3}{2}} \sqrt{T})$	$\mathcal{O}(n\, (\log n)^{\frac{1}{2}} \sqrt{T})$

⋆ $D :=$ the maximum dimensionality of the matrices in the matrix-chain multiplication problem

⋆ $C :=$ the capacity of the knapsack in the knapsack problem