Low-Rank Factorizations for Fast Inference in Structured Models

Justin Chiu* [1] Yuntian Deng* [2] Alexander Rush [1]

[1]Cornell Tech and [2]Harvard University

Introduction

Structured distributions, i.e. distributions over combinatorial spaces, are commonly used to learn latent discrete representations from observed data. However, scaling these models is bottlenecked by the high computational and memory complexity of marginalization:

- Hidden Markov Model (HMM): $O(L^2)$
- Probabilistic Context-Free Grammar (PCFG): $O(L^3)$

As prior work has found that scaling the size is necessary for performance, we propose a general method for scaling inference in structured models that admit exact inference via dynamic programming. We:

- Express marginalization as matrix-vector products
- Trade off speed and expressivity via rank

Structured Models

We model observations $x = (x_1, \dots, x_T)$ via combinatorial latent structure z, where each latent nodes z_i has an associated discrete label set $[L]$. As the structure z is unobserved, we must perform training and evaluation via marginalization:

$$p(x) = \sum_z p(x, z).$$

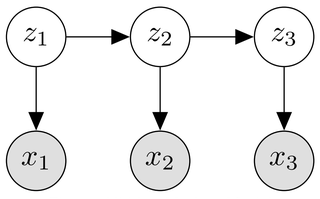

Figure: Hidden Markov Model

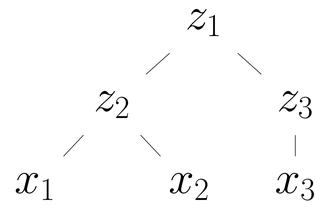

Figure: Probabilistic Context-Free Grammar

Hypergraph Representation

- Hypergraphs consist of nodes and hyperedges
  - Hyperedge consists of a head node and set of tail nodes
- Perform marginalization by traversing hypergraph
  - Aggregate scores from tails to head via a matrix-vector product

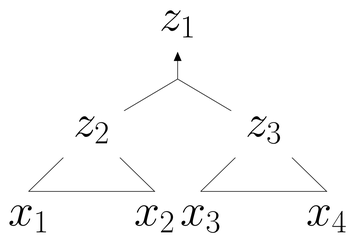

Figure: Hyperedge representation for PCFG

Hypergraph Marginalization

For each hyperedge e in topological order:

- Combine tail scores α_1, α_2 into joint tail scores β_v
- Apply score matrix Ψ_e and aggregate in head scores α_u

Algorithm 1 Hypergraph marginalization / belief prop

for $u \leftarrow v$ hyperedge e topologically **do**
- $\beta_v \leftarrow \alpha_{v_1} \alpha_{v_2}^\top$ ▷ $O(L^{|e|})$
- $\alpha_u \xleftarrow{+} \Psi_e \beta_v$ ▷ $O(L^{|e|+1})$

return $\alpha_S^\top \mathbf{1}$

Low-Rank Factorization

Marginalization is bottlenecked by

- The number of hyperedges
- The cost of each matrix-vector product

We reduce the cost of the latter via a low-rank R factorization. We can factor a score matrix Ψ into $U \in \mathbb{R}^{L \times R}, V \in \mathbb{R}^{L^{|e|} \times R}$:

$$\boxed{\Psi} \times \boxed{\beta} = \boxed{U} \times \left(\boxed{V^\top} \times \boxed{\beta} \right)$$

Low-Rank Hypergraph Marginalization

Applying the low-rank factorization to the scoring matrices yields Algorithm 2:

Algorithm 2 Low-rank marginalization

for $u \leftarrow v_1, v_2$ hyperedge e topologically **do**
- $\beta_v \leftarrow \alpha_{v_1} \alpha_{v_2}^\top$ ▷ $O(L^{|e|})$
- $\gamma \leftarrow V_e^\top \beta_v$ ▷ $O(L^{|e|} R)$
- $\alpha_u \xleftarrow{+} U_e \gamma$ ▷ $O(LR)$

return $\alpha_S^\top \mathbf{1}$

- HMM marginalization goes from $O(L^2)$ to $O(LR)$
- PCFG marginalization goes from $O(L^3)$ to $O(L^2 R)$

Expressivity

- Rank constraints limit expressivity
- Only need to constrain subsets of matrices for speedups
  - Transition matrix for HMMs
  - Subset of transitions for PCFGs

Experiments: Accuracy and Speed

On language modeling on PennTreebank, Low-rank HMMs achieve similar accuracy at faster speeds.

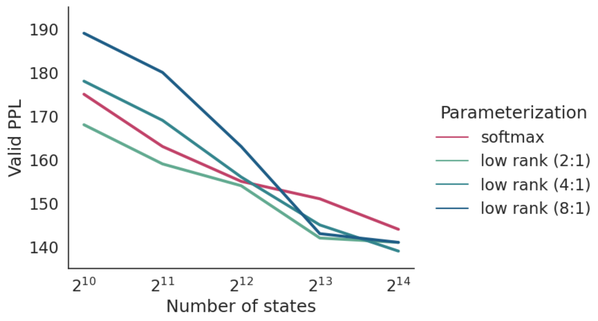

Figure: HMM Accuracy

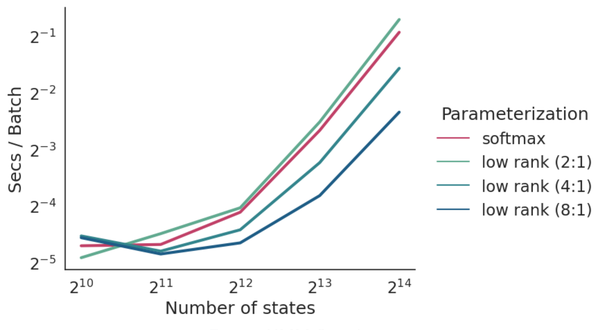

Figure: HMM Speed

Speed-Accuracy Frontier

Low-rank HMMs also dominate softmax HMMs on the speed-accuracy frontier.

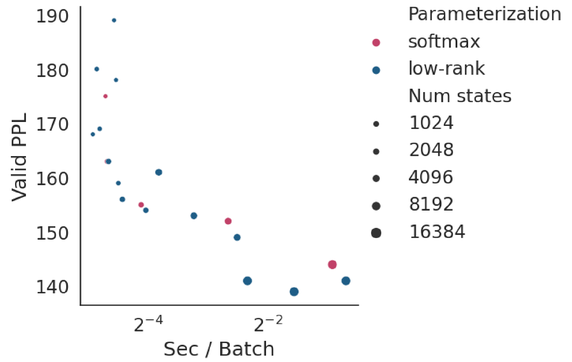

Figure: Speed-accuracy frontier for HMMs

PCFG Results

Low-rank PCFGs (LPCFG) also achieve similar accuracy at faster speeds, but scaling is not as large as LHMMs.

$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{P}\rvert$	Model	R	PPL	Secs/batch
30	60	PCFG	-	252.60	0.23
		LPCFG	8	247.02	0.27
		LPCFG	16	250.59	0.27
60	120	PCFG	-	234.01	0.33
		LPCFG	16	217.24	0.28
		LPCFG	32	213.81	0.30
100	200	PCFG	-	191.08	1.02
		LPCFG	32	203.47	0.64
		LPCFG	64	194.25	0.81

Conclusion

Low-rank constraints speed up marginalization

- Only need to constrain bottleneck parameters
- Most effective with large models

Please see the paper for more experiments:

- Combination with other structure that admit fast matrix-vector products
- More datasets (polyphonic music, video) and models (hidden semi-Markov)