Scaling Hidden Markov Language Models

March 3, 2021

Abstract

Modern methods for language models based purely on neural networks are opaque and difficult to analyze, while alternative methods based on probabilistic graphical models are interpretable but not performant. We hypothesize that classical methods are not as performant because they have neither been scaled to similar sizes as their modern neural counterparts nor taken advantage of modern parallel hardware. A significant barrier to scaling in classical methods is inference, which typically scales at best polynomially in the size of the model. In order to make probabilistic models performant while maintaining interpretability, we present techniques for circumventing the heavy computational costs of inference. We first focus on scaling one of the simplest models, hidden Markov models, then show our techniques generalize to more complex models, such as probabilistic context-free grammars.

1 Introduction

Neural network-based generative models have led to progress in difficult tasks such as language modeling and machine translation. However, this progress comes at the cost of interpretability. Neural networks are flexible function approximations, but that flexibility results in opaque models that are difficult to analyze. As opposed to post-hoc analysis of models [8], we instead propose to explore the space of models that interpretable from the start. In particular, we focus on probabilistic graphical models, which allow for the execution of arbitrary probabilistic queries. These queries include the ability to calculate conditional and marginal probabilities given evidence. This interpretability comes at a cost. Performing these queries via inference is expensive in both time and space. The computational complexity of inference is a major impediment to scaling graphical models to large sizes, which we hypothesize prevents them from reaching the performance of neural networks. We seek to answer the question of whether we can have graphical models that are both performant and interpretable.

We use language modeling as a benchmark task, as the complex and sometimes long-range phenomena in natural language provides a good testbed. Language model benchmarks are currently dominated by autoregressive neural models, which makes it difficult to analyze what properties of language the models are actually capturing. This is an issue if the goal is to learn from models in order to answer scientific questions.

In this work, we investigate scaling probabilistic graphical models which explicitly reason about latent variables. We focus on the one of the simplest latent variable models, the hidden Markov model (HMM). HMMs posit a simple generative process, that first generates a sequence of latent states then the emissions. Additionally, HMMs make very strong conditional independence assumptions. Despite the simplicity of HMMs, they are still computationally expensive to scale due to the cost of inference.

In order to scale HMMs, we propose a series of choices in parameterization that result in computational speedups for inference in HMMs: sparse emission constraint, state dropout, and an efficient kernel-based softmax. Before diving into these techniques, we first review HMMs.

2 Hidden Markov Models for Language Modeling

Hidden Markov models (HMMs) have a rich history in natural language processing. They have been used for a series of tasks, including speech recognition [5], part of speech tagging [3], and word alignment in machine translation [9]. They have also been used as components in neural models, such as in attention [7]. We focus on applying HMMs to the task of language modeling, where the goal is to model the tokens in a sentence $x = (x_1, \ldots, x_T)$.

HMMs are one of the simplest latent variable models for language modeling with a per-token latent variable. The simplicity is a result of the very strong conditional independence assumption: Every timestep has a single discrete state latent variable for representing context, and HMMs emit

each token given only the corresponding latent state. Additionally, the next state is a function of only the previous state. These independence assumptions result in a bottleneck where all information must flow through the single state variable, causing performance to be limited by the number of states while also giving the model interpretability by limiting dependencies.

Formally, HMMs have the following generative process: for each $t \in [T]$ timestep, first choose a latent state $z_t \in \mathcal{Z}$, where $|\mathcal{Z}|$ is the number of latent states, then choose a token to emit $x_t \in \mathcal{X}$. This defines the joint distribution:

$$p(x, z) = \prod_t p(x_t \mid z_t) p(z_t \mid z_{t-1}), \tag{1}$$

with transitions $p(z_t \mid z_{t-1})$, emissions $p(x_t \mid z_t)$, and a distinguished start state $z_0 = S$.

Training an HMM requires marginalizing over the unobserved $z = (z_1, \ldots, z_T)$ in order to obtain the evidence, $p(x) = \sum_z p(x, z)$, via the forward algorithm. The forward algorithm can be written as a sequence of matrix-vector multiplications: Let the transition operators $\Lambda_t \in \mathbb{R}^{|\mathcal{Z}| \times |\mathcal{Z}|}$ for $t \in [2, \ldots, T]$, with entries

$$[\Lambda_t]_{z_t, z_{t-1}} = p(x_t \mid z_t) p(z_t \mid z_{t-1}),$$

with the first operator given by

$$[\Lambda_1]_{z_1, z_0} = \begin{cases} p(x_1 \mid z_1) p(z_1 \mid z_0 = S) & z_0 = S \\ 0 & \text{otherwise.} \end{cases}$$

The evidence is then given by

$$p(x) = \mathbf{1}^\top \Lambda_1 \Lambda_2 \cdots \Lambda_T \mathbf{1}, \tag{2}$$

where $\mathbf{1} \in \mathbb{R}^{|\mathcal{Z}| \times 1}$ is the column vector of all ones.

On a serial machine, the cost of computing each matrix-vector product in the above equation takes time $O(|\mathcal{Z}|^2)$, resulting in a total running time of $O(T|\mathcal{Z}|^2)$ for the forward algorithm. The quadratic dependence on the number of states precludes scaling to extremely large state spaces.

gradient computation requires $O(T|\mathcal{Z}|^2)$ space

In the following sections we will introduce techniques that aim to reduce the dependence of the time and space complexity of inference on the size of the state space.

3 Conditional Expectations with Generalized Softmax

Parameterization of the probability distributions in a model affects both computational performance as well as generalization. Recent work on efficient attention in neural models shows that careful choice of parameterization of the softmax kernel, the main component of attention, results in large computational gains at little cost in accuracy [1, 4]. In this section we will cover generalized softmax with kernels and the efficient computation of conditional expectations with generalized softmax.

Softmax is commonly used to parameterize conditional distributions in neural nets, and has origins in smooth approximations of argmax as well as the conditional max entropy problem []. In a conditional distribution $p(x \mid z)$, the softmax parameterization is as follows:

$$p(x \mid z) = \frac{\exp(\mathbf{u}_z^\top \mathbf{v}_x)}{\sum_{x'} \exp(\mathbf{u}_z^\top \mathbf{v}_{x'})},$$

with $\mathbf{u}_z, \mathbf{v}_x \in \mathbb{R}^d$ vector-space embeddings of values of x, z.[1] The numerator, $\exp(\mathbf{u}_z^\top \mathbf{v}_x)$, is the exponential kernel [6]. Although the parameterization of softmax relies on the exponential kernel, we can generalize softmax to arbitrary nonnegative kernels $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$. Replacing the exponential kernel in softmax with the nonnegative kernel K gives us generalized softmax,[2]

$$p(x \mid z) = \frac{K(\mathbf{u}_z, \mathbf{v}_x)}{\sum_{x'} K(\mathbf{u}_z, \mathbf{v}_{x'})},$$

where we again have embeddings $\mathbf{u}_z, \mathbf{v}_x \in \mathbb{R}^d$.

Kernels have a long history in machine learning, where they are used to extend linear classification models to nonlinear feature spaces, such as in support vector machines or regression. In those settings, kernels are intuitively used to measure the similarity between two data points, relying on the existence of a feature mapping in an associated reproducing kernel Hilbert space for kernel functions with the finitely positive semi-definite property []. In our generalized softmax setting, kernels instead parameterize conditional distributions by measuring the propensity of one random variable taking a particular value given the value of another random variable. However, the connection to feature

[1] Softmax may also have a temperature parameter that rescales the dot product, which we omit.

[2] The naming of generalized softmax is inspired by generalized linear models.

maps in reproducing kernel Hilbert spaces is still useful. In particular, the existence of a feature map $\phi : \mathbb{R}^d \to \mathbb{F}$, which maps from the input space $\mathbb{R}^d$ to the feature space $\mathbb{F}$, allows us to write generalized softmax in matrix form:

$$p(x \mid z) \propto [\phi(U)\phi(V)^\top]_{zx},$$

where $\phi(U) = [\phi(\mathbf{u}_1)^\top, \phi(\mathbf{u}_2)^\top, \ldots,] \in \mathbb{R}^{|\mathcal{Z}| \times \dim(\mathbb{F})}$ and $\phi(V) = [\phi(\mathbf{v}_1)^\top, \phi(\mathbf{v}_2)^\top, \ldots] \in \mathbb{R}^{|\mathcal{X}| \times \dim(\mathbb{F})}$. We will rely on the dimension of the feature space $\dim(\mathbb{F}) < \min(|\mathcal{Z}|, |\mathcal{X}|)$ in order to speed up and reduce the space complexity of computing conditional expectations.

Many probabilistic queries can be written as conditional expectations. In particular, we can write marginalization in an HMM as a conditional expectation:

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x \mid z) = \mathbb{E}_z \left[p(x \mid z)\right].$$

Decomposing the expectation over time and examining a single timestep, we have

$$p(x_t \mid x_{<t}) = \sum_{z_t} p(x_t \mid z_t)p(z_t \mid x_{<t}).$$

The second term, $p(z_t \mid x_{<t})$, can be further decomposed using the generative process of the HMM to

$$p(z_t \mid x_{<t}) = \sum_{z_{t-1}} p(z_t \mid z_{t-1})p(z_{t-1} \mid x_{<t}),$$

altogether yielding the equation

$$p(x_t \mid x_{<t}) = \sum_{z_t} p(x_t \mid z_t) \sum_{z_{t-1}} p(z_t \mid z_{t-1})p(z_{t-1} \mid x_{<t}) = \mathbb{E}_{z_{t-1}, z_t \mid z_{t-1}} \left[p(x_t \mid z_t)\right] p(z_{t-1} \mid x_{<t}).$$

We can use the matrix form of generalized softmax to speed up this computation. Let $[\mathbf{f}_t]_{z_t} = p(x_t \mid z_t), [\boldsymbol{\alpha}_t]_{z_{t-1}} = p(z_{t-1} \mid x_{<t})$ in

$$\mathbb{E}_{z_t \mid x_{<t}} \left[p(x_t \mid z_t)\right] = (\boldsymbol{\alpha}_t \circ \mathbf{d})^\top \phi(U)\phi(V)^\top \mathbf{f}_t,$$

where $\mathbf{d} = \frac{1}{\phi(U) \sum_z \phi(\mathbf{v}_z)} \in \mathbb{R}^{|\mathcal{Z}|}$.

Unfortunately, in the case of the exponential kernel, the feature space $\mathbb{F}_{\exp}$ has infinite dimension. This can be seen from the Taylor expansion:

$$\begin{aligned}
\exp(\mathbf{u}^\top \mathbf{v}) &= \sum_{n=0}^{\infty} \frac{(\mathbf{u}^\top \mathbf{v})^n}{n!} \\
&= \sum_{n=0}^{\infty} \frac{(\sum_{i=1}^d u_i v_i)^n}{n!} \\
&= \sum_{n=0}^{\infty} \frac{\sum_{\mathbf{j} \in [d]^n} (\prod_{i=1}^n u_{j_i})(\prod_{i=1}^n v_{j_i})}{n!},
\end{aligned}$$

where we have expanded the terms of the numerator by summing over all different orderings $\mathbf{j}$ of subsets of size n. This yields a feature map of

$$[\phi(\mathbf{u})]_{n,\mathbf{j}} = \frac{\prod_{i=1}^n u_{j_i}}{\sqrt{n!}}$$

for all $n \in \mathbb{N}$.[3] We can therefore express the exponential kernel as an inner product of infinite dimensional vectors [2],

$$\exp(\mathbf{u}^\top \mathbf{v}) = \sum_{n=0}^{\infty} \sum_{\mathbf{j} \in [d]^n} [\phi(\mathbf{u})]_{n,\mathbf{j}} [\phi(\mathbf{v})]_{n,\mathbf{j}} = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle,$$

where the dimension of the feature space $\dim(\mathbb{F}_{\exp}) = \dim(\phi(\mathbf{u})) = \infty$. Since the infinite dimensional feature space of the exponential kernel is clearly larger than $\min(|\mathcal{Z}|, |\mathcal{X}|)$, recent work has instead turned to approximations of the exponential kernel based on random Fourier features [1, 4, 6]. These approaches use a finite-dimensional unbiased estimator of the exponential kernel.

We propose to take a different strategy, where we directly learn a feature map while disregarding the explicit functional form of the kernel.

[3] Note that while this is one possible feature map, there may exist multiple valid feature maps.

4 Kernel Approximations

Move to appendix

Taylor approximation, generating functions

limitations?

4.1 Bochner Theorem

4.2 Nystrom

5 Kernelized Inference

6 Generalization: Kernelized Belief Propagation

7 Spectral Methods?

References

[1] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

[2] Andrew Cotter, Joseph Keshet, and Nathan Srebro. Explicit approximations of the gaussian kernel. *CoRR*, abs/1109.4603, 2011. URL `http://arxiv.org/abs/1109.4603`.

[3] Bernard Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994. URL `https://www.aclweb.org/anthology/J94-2001`.

[4] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=QtTKTdVrFBB`.

[5] Lawrence R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, page 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601244.

[6] Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13857–13867. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/e43739bba7cdb577e9e3e4e42447f5a5-Paper.pdf`.

[7] Shiv Shankar and Sunita Sarawagi. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BkltNhC9FX`.

[8] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Association for Computational Linguistics*, 2019. URL `https://arxiv.org/abs/1905.05950`.

[9] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, page 836–841, USA, 1996. Association for Computational Linguistics. doi: 10.3115/993268.993313. URL `https://doi.org/10.3115/993268.993313`.