Scaling Hidden Markov Language Models

February 16, 2021

Abstract

Asdf

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. Rather than post-hoc analysis [**?**], we instead propose to explore the space of models that are defined with interpretability in mind. In particular, we focus on probabilistic graphical models [], which allow for the execution of arbitrary probabilistic queries, i.e. the calculation of (functions of) probability distributions via operations such as conditioning and marginalization [**?**] (simplify). We seek to answer the question of whether we can have 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. Additionally, language modeling captures import scientific and linguistic questions (more on this later). 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. For example, it is difficult to analyze the inner representations used in syntax-inspired neural language models [1] (this is my own take on that paper, need to find a better citation). 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 and constraints of HMMs, they are still computationally expensive to scale due to the cost of marginalization.

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 softmax kernel feature map approximation.

Before diving into these contributions, we first review HMMs.

2 Hidden Markov Models for Language Modeling

Hidden Markov models (HMMs) have a rich history in natural language processing. Speech recognition [3], part of speech tagging [2], and word alignment in machine translation [4]. Have seen some use in neural attention-based models (cite posterior attention). We focus on applying HMMs to the task of language modeling, where the goal is to model the tokens in a sentence $x = (x_1, \dots, x_T)$.

HMMs are cool because ?.

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 $z = (z_1, \dots, z_T)$ in order to obtain the evidence $p(x) = \sum_z p(x, z)$, accomplished 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 $t \in [2,\dots,T]$, 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 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.

3 Parameterization: Low Rank Decompositions

Linear

4 Kernels

Nonlinear Application to features, not datapoints.

Hilbert space, RKHS Feature mapping Empirical Mapping?

4.1 Softmax

Why is softmax / exponential kernel special? Maximum entropy distributions, exponential family. Slides on information projections Power series representation?

5 Softmax Approximations

Taylor approximation, generating functions

limitations?

6 Kernelized Inference

7 Generalization: Kernelized Belief Propagation

8 Spectral Methods

References

[1] Wenyu Du, Zhouhan Lin, Yikang Shen, Timothy J. O'Donnell, Yoshua Bengio, and Yue Zhang. Exploiting syntactic structure for better language modeling: A syntactic distance approach, 2020.

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

[3] 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.

[4] 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`.

A Gradient Estimator Implementation

In the log-semiring, addition is given by $\bigoplus = \text{L}\Sigma\text{E}$ and multiplication by $\bigotimes = +$. Consider the linear chain CRF, $\bigotimes_t \psi(x_{t-1}, x_t)$, with $\psi(x_{t-1}, t) = f(x_t) \oplus g(x_{t-1}, x_t)$. We would like to compute the gradient of the log partition function, $A = \bigoplus_x \bigotimes_t \psi(x_{t-1}, x_t)$. Recall the gradient identities

$$\begin{aligned} \nabla_a a \bigoplus b &= \frac{\exp(a)}{\exp(a \bigoplus b)} \\ \nabla_a a \bigotimes b &= 1. \end{aligned} \tag{3}$$

We then have

$$\begin{aligned} &\nabla_{\psi(x_a, x_b)} \bigoplus_t \psi(x_{t-1}, x_t) \\ &\quad = \nabla_{\psi(x_a, x_b)} \bigoplus_t \psi(x_{t-1}, x_t) \end{aligned} \tag{4}$$