Particle MCMC for Latent LSTM Allocation

Xun Zheng, Manzil Zaheer
Carnegie Mellon University
xzheng1,manzil@cmu.edu

Amr Ahmed, Yuan Wang
Google Research
amar,yuanwang@google.com

Eric P. Xing
Carnegie Mellon University
epxing@cs.cmu.edu

Alexander J Smola
Carnegie Mellon University
alex@smola.org

Abstract

We present an efficient particle-based sampler for Latent LSTM Allocation [6]. Unlike the algorithm used in [6], we draw from the joint posterior directly without making any factorization assumptions. Experimental results confirms the superiority and stability of this algorithm on a variety of domains.

1 Introduction

Model Let $x_{1:T} = (x_1, \ldots, x_T)$ be a sequence of observations and $z_{1:T} = (z_1, \ldots, z_T)$ be the associated latent states. Latent LSTM Allocation (LLA) [6] is a model that jointly learns the transition dynamics of the latent states as well as the emission distribution of observations given latent states, by combining LSTM [5] with topic models [3]. The generative process of LLA for a single sequence is:

- For $t = 1, \ldots, T$:
  1. Perform LSTM transition: $s_t = \mathsf{LSTM}(s_{t-1}, z_{t-1})$
  2. Draw latent state: $z_t \sim \mathsf{Multinomial}(\mathsf{softmax}(Ws_t + b))$
  3. Draw observation: $x_t \sim \mathsf{Multinomial}(\phi_{z_t})$

where each ϕ_k is the parameter of the emission distribution, typically drawn from a Dirichlet prior. The generative process specifies the following joint likelihood:

$$p(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\omega(z_t|z_{1:t-1}) p_\phi(x_t|z_t), \tag{1}$$

where ω is the set of parameters of the LSTM and the linear transformation.

Parameter estimation We continue with a single sequence for the ease of notation. Given the joint likelihood, the stochastic EM algorithm leads to the following updates:

(E-step) Since it is difficult to compute or take expectation over the posterior due to the non-Markovian transition, collect posterior samples instead:

$$z^\star_{1:T} \sim p_\omega(z_{1:T}) p_\phi(x_{1:T}|z_{1:T}). \tag{2}$$

(M-step) Given the posterior samples $z^\star_{1:T}$, which can be seen as Monte Carlo estimate of the expectations, the subproblem for ω and ϕ are

$$\omega^\star = \operatorname*{argmax}_\omega \ \log p_\omega(z^\star_{1:T}), \quad \phi^\star = \operatorname*{argmax}_\phi \ \sum_t \log p_\phi(x_t|z^\star_t), \tag{3}$$

which is exactly the MLE of the LSTM with $z^\star_{1:T}$ serving as the input sequence, and the MLE of the given emission model.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

2 Inference with Particle Gibbs

In this section, we discuss how to draw samples from the posterior (2), corresponding to the E-step of the stochastic EM algorithm. Due to non-Markov state transition, only forward messages are available, for instance

$$\alpha_t \equiv p(x_t|z_{1:t-1}) \propto \int p_\omega(z_t|z_{1:t-1})p_\phi(x_t|z_t)\,\mathrm{d}z_t \tag{4}$$

$$\gamma_t \equiv p(z_t|z_{1:t-1}, x_t) \propto p_\omega(z_t|z_{1:t-1})p_\phi(x_t|z_t), \tag{5}$$

assuming the integration and normalization can be performed efficiently. The task is to draw from the joint posterior of $z_{1:T}$ only given access to these forward messages.

One way to circumvent the tight dependencies in $z_{1:T}$ is to make a factorization assumption, as in [6]. More concretely, the joint distribution is decomposed as

$$\text{(factorization assumption)} \qquad p(z_{1:T}|x_{1:T}) \propto \prod_t p_\omega(z_t|z_{1:t-1}^{\mathrm{prev}})p_\phi(x_t|z_t), \tag{6}$$

where $z_{1:t-1}^{\mathrm{prev}}$ is the assignments from the previous inference step. However, as we confirm in the experiments, this assumption can be restrictive since the flexibility of LSTM state transitions is offset by considering each time step independently.

An alternative is Sequential Monte Carlo (SMC) [4], which is a principled way of sampling from a sequence of distributions. The idea is to approximate the posterior with point masses, *i.e.*, weighted particles. The algorithm proceeds as follows, with P denoting the number of particles.

1. Let $z_0^p = z_0$ and weights $\alpha_0^p = 1/P$ for $p = 1, \ldots, P$.
2. For $t = 1, \ldots, T$:
   (a) Sample ancestors $a_{t-1}^p \sim \alpha_{t-1}$ for $p = 1, \ldots, P$.
   (b) Sample particles $z_t^p \sim \gamma_t^p = p(z_t|z_{1:t-1}^{a_{t-1}^p}, x_t)$ for $p = 1, \ldots, P$.
   (c) Set $z_{1:t}^p = (z_{1:t-1}^{a_{t-1}^p}, z_t^p)$ for $p = 1, \ldots, P$.
   (d) Compute the normalized weights for $p = 1, \ldots, P$:

$$\alpha_t^p \propto \tilde{\alpha}_t^p = \frac{p(z_{1:t}^p, x_{1:t})}{p(z_{1:t-1}^{a_{t-1}^p}, x_{1:t-1})p(z_t^p|z_{1:t-1}^{a_{t-1}^p}, x_t)} = \int p_\omega(z_t|z_{1:t-1}^{a_{t-1}^p})p_\phi(x_t|z_t)\,\mathrm{d}z_t. \tag{7}$$

After a full pass over the sequence, the algorithm produces Monte Carlo approximation of the posterior where samples can be easily drawn by sampling a path indexed by the last particle:

$$z_{1:T}^\star \sim \hat{p}(z_{1:T}|x_{1:T}) = \sum_p \alpha_T^p \delta_{z_{1:T}^p}(z_{1:T})\,. \tag{8}$$

Since SMC produces a Monte Carlo estimate, as the number of particles $P \to \infty$ the approximate posterior is guaranteed to converge to the true posterior for a fixed sequence. However, as T increases, the number of particles needed to provide a good approximation grows exponentially. To avoid this, we use Particle Gibbs [2] which treats $\hat{p}(\cdot)$ as a proposal and designs a Markov kernel that leaves the target distribution invariant. The resulting algorithm is a conditional SMC update in a sense that a reference path $z_{1:T}^{\mathrm{ref}}$ with its ancestral lineage is fixed throughout the particle propagation of SMC. It can be shown that this simple modification to SMC produces a transition kernel that is not only invariant, but also ergodic under mild assumptions. In practice, we use the assignments from previous step as the reference path.

3 Experiments

We present some empirical studies for the proposed inference method (denoted as SMC). For all experiments, we divide examples into 60% for training and 40% for testing. All the algorithms are implemented on TensorFlow [1]. We run our experiments on a commodity machine with Intel® Xeon® CPU E5-2630 v4 CPU, 256GB RAM, and 4 NVidia® Titan X (Pascal) GPU.

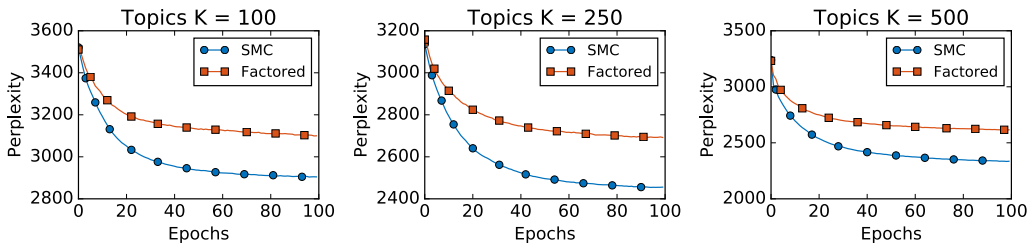

Figure 1: Test perplexity of SMC (proposed) and Factored (prior work) in different settings.

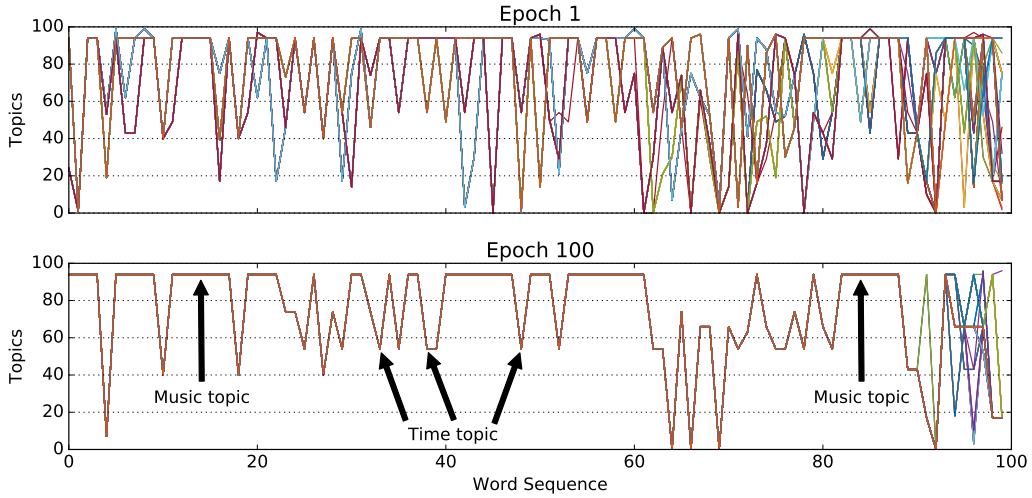

Figure 2: Particle paths of a document about a music album. Top row: at epoch 1. Bottom row: at epoch 100. After epoch 100 the document has converged to a sparse set of relevant topics.

3.1 Language Modeling

We used the publicly available Wikipedia dataset, excluding short documents that have less than 500 words. Figure 1 compares the test perplexity between the proposed method and the factored algorithm, with varying number of topics on a subset of 10k documents. We can see that the new algorithm uniformly outperforms the old one, suggesting the benefit of sampling from the full joint.

In Figure 2, we show the particle paths on a snippet of an article about a music album. At the top row, which plots the particle paths at the initial iteration, the model proposed a number of candidate topic sequences since it is uncertain about the latent semantics yet. However, after 100 epochs, as we can see from the bottom row, the model is much more confident about the underlying topical transition.

3.2 User Modeling

We use an anonymized sample of user search click history to measure the accuracy of different models on predicting users' future clicks. The dataset is anonymized by removing all items appearing less than a given threshold, this results in a dataset with 100K vocabulary and we vary the number of users from 500K to 1M. We fix the number of topics at 500 for all user experiments. The dataset is less structured than the language modeling task since users' click patterns are less predictable than the sequence of words which follow definite syntactic rules. As shown in table 1, the benefit of new inference method is highlighted as it yields much lower perplexity than the factored model.

Algorithm	# Users	
	500k	1M
Factored	2430	2254
SMC	1464	1447

Table 1: Comparison on user data

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and Others. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[5] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[6] M. Zaheer, A. Ahmed, and A. J. Smola. Latent LSTM Allocation: Joint Clustering and Non-Linear Dynamic Modeling of Sequence Data. In *ICML*, 2017.