Combinators for Modeling and Inference

Eli Sennesh*
Northeastern University
esennesh@ccis.neu.edu

Hao Wu*
Northeastern University
wu.hao10@husky.neu.edu

Jan-Willem van de Meent
Northeastern University
j.vandemeent@northeastern.edu

Abstract

We develop a *combinator* library for the Probabilistic Torch framework. Combinators are functions accept and return models. Combinators enable compositional interleaving of modeling and inference operations, which streamlines model design and enables model-specific inference optimizations. *Model combinators* define a static graph from (possibly dynamic) model components. Examples of patterns that can be expressed as combinators are mixtures, state-space models, and models with global and local variables. *Inference combinators* preserve model structure, but alter model evaluation. Operations that we can represent as combinators include enumeration, importance sampling, resampling, and Markov chain Monte Carlo transitions. We validate our approach on variational methods for hidden Markov models.

Introduction

Libraries such as Edward [13], Pyro [3], and Probabilistic Torch [12], extend deep learning frameworks with functionality for probabilistic modeling. This enables users to define *deep generative models* in which neural networks represent data such as images and text, while defining structured priors on latent variables. Neural networks can also serve as *inference models* mapping from data to latent variables. Both can be trained jointly via autoencoding variational methods [2, 9], which optimize a bound on the marginal likelihood via a Monte Carlo estimate of its gradient.

While autoencoding methods hold tremendous appeal, they are not a magic bullet. Models with local and global variables, such as hidden Markov models with unknown states and transitions, are particularly challenging. In autoencoders, the inference model predicts both levels of variable at once. In doing so, it effectively performs one-shot inference. This is feasible for certain models [4], but overburdens the inference network, which must "fully invert" the model before sampling anything.

Classical Bayesian inference strategies often rely on properties of the model structure, for example, by iterating between updates to blocks of variables. In the HMM, we can predict transition probabilities from the latent state sequence or vice versa. Research in probabilistic programming has traditionally emphasized the development of broadly applicable inference methods, which complicates model-specific optimizations. To address this need, we propose to develop abstractions to specify models and inference strategies in a modular and compositional manner.

Model and Inference Composition

In recent years, there have been a number of efforts to develop specialized inference methods for probabilistic programming. Venture [6] provides primitives for inference programming that can act on subsets of variables in an execution trace. There has also been work to formalize notions of valid inference compositionality. Hakaru [8] frames inference as program transformations, which can be composed so as to preserve a measure-theoretic denotation [14]. Work by Scibior et al. [11] defines measure-theoretic validity criteria for compositional inference transformations, usable for defining correct-by-construction algorithms.

Models in Probabilistic Torch are written in Python and can make use of if expressions, loops, and other control flow constructs. This means that models can dynamically instantiate random variables in a data-dependent manner, but also that it is more difficult to perform static analysis on the computation graph [10]. To reason about inference in the absence of static guarantees, we postulate the following requirements for model and inference composition:

1. Composition is static, evaluation dynamic. A model is statically composed from other models, but evaluates based on data. In the HMM example, we can compose a model that samples global parameters with a model for a sequence of variably many states and observations. While the resulting samples are dynamic, the model components always exist.

2. Inference composition preserves proper weights. A model defines a density $\gamma(x; y)$, which may be unnormalized, conditioned on some set of inputs y. Evaluation must yield a *properly weighted* [7] sample (X, W) such that, for all measurable functions f,

$$\mathbb{E}[f(X)W] = \int f(x)\gamma(x; y)\, dx.$$

We require that any inference operation must preserve this property. Operations that satisfy this requirement include importance sampling with a properly weighted proposal, resampling, and application of a Markov chain Monte Carlo (MCMC) operator.

Proper weighting implies that $\mathbb{E}[W] = \int \gamma(x; y)\, dx = Z(y)$, i.e. the weight W is an unbiased estimate of the normalizer. For a parameterized density $\gamma_\theta(x; y)$, we can approximate the gradient $\nabla_\theta \log Z_\theta(y)$ using the Monte Carlo estimator

$$\sum_k \frac{w_\theta^k}{\sum_l w_\theta^l} \nabla_\theta \log \gamma_\theta(x^k\, ;\, y),$$

where $x^k, w^k \backsim \gamma_\theta(x; y)$ are generated by evaluating the model.

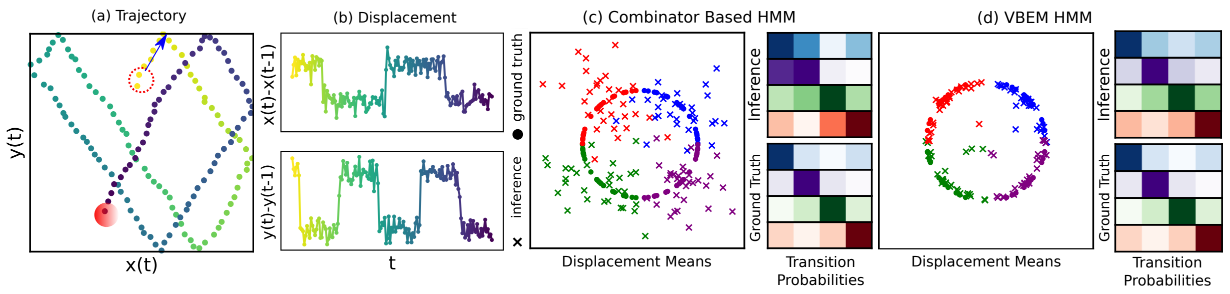

Figure 1. Combinator-based variational inference in hidden Markov models. a): a bouncing ball trajectory with initial velocity. b): and the displacement along x and y axis, respectively. c): Inferred travel directions and transition probabilities from combinator-based variational Sequential Monte Carlo with the inclusive Kullback-Leibler divergence. d): Inferred travel directions and transition probabilities from Variational Bayesian Expectation Maximization.

When we sample from an inference model $q_\phi(x \mid y)$, we obtain weights $w_{\theta,\phi} = \gamma_\theta(x\,;\,y)/q_\phi(x \mid y)$. Inspired by the reweighted wake-sleep algorithm [1, 5], we can minimize $\mathrm{KL}(\gamma_\theta(x\,;\,y)/Z_\theta(y)\,||\,q_\phi(x\,|\,y))$ using the estimator

$$-\sum_k \frac{w^k_{\theta,\phi}}{\sum_l w^l_{\theta,\phi}} \nabla_\phi \log q_\phi(x^k \mid y).$$

Note that the gradient w.r.t. θ computes $\nabla_\theta \log w_{\theta,\phi}$ whereas the gradient w.r.t. ϕ computes $-\nabla_\phi \log w_{\theta,\phi}$. In other words, we can perform variational inference in any properly weighted model by automatic differentiation on the importance weights.

Model Combinators

A model is a stochastic computation that returns a properly weighted sample. Model evaluation can be conditioned on a trace, which is an object that holds values for the set of random variables that were instantiated during a particular evaluation of the model. Combinators are functions that accept models as inputs and return a model.

Most higher-order functions commonly used in functional programming can be implemented as combinators. This includes, `partial`, `compose`, `map`, and `reduce`, which have the usual type signatures. We add a recurrent loop combinator `recur(step, init, n)`, in which `init` is a model returning an initial state, `step` accepts a state and returns another, and `n` defines a number of iterations.

In addition to combinators that correspond to functional programming constructs, we provide combinators for a number of model families, including `mix(like, latent)`, which accepts a likelihood `like` and a latent variable model `latent`. The `mix` and `recur` combinators can be combined to define `ssm(like, step, init, n)`, which returns a state space model based on the provided components.

Inference Combinators

By default, models have a likelihood weighting semantics: a $\gamma(x; y)$ generates samples from a $p(x; y)$ with weights $w = \gamma(x; y)/p(x; y)$. `importance(p, q, size=None)` accepts models `p` and `q`, possibly unnormalized, and returns a model targeting `p` using proposals from `q`. The optional argument `size` gives the shape of the sample set requested. Given a sample set, `resample(p)` returns a model that resamples at the end of evaluation, returning samples with the average weight of the sample set. `smc` performs stepwise importance resampling as part of models defined using `reduce`, or `recur`. Our framework will also support weight-preserving MCMC transition operators, such as single-site Metropolis-Hastings and Hamiltonian Monte Carlo (not yet implemented).

model	`compose`, `partial`, `map`, `reduce`, `recur`, `mix`, `ssm`
inference	`importance`, `resample`, `smc`, `imh`

Table 1. Combinators provided by Probabilistic Torch.

Evaluation

Figure1 shows inference results on simulated data. The data models a bouncing particle trajectory in a closed box (Fig. 1a). This trajectory has a piece-wise constant noisy velocity, which means that the displacements at each time step (Fig. 1b) can be described by an HMM with Gaussian observations, where each state's observation mean corresponds to the average velocity along one of four possible directions of motion. We compare variational SMC inference results for a combinator-based implementation (Fig. 1c) to those obtained using variational Bayesian expectation maximization (VBEM) (Fig. 1d), for a set of 30 time series that each contain 200 time points. Because VBEM optimizes the exclusive Kullback-Leibler (KL) divergence, $\mathcal{D}_{KL}(q\,||\,p)$, while in our combinator-based inference we optimized $\mathcal{D}_{KL}(p\,||\,q)$, the latter approximated the posterior with greater variance.

References

[1] Jörg Bornschein and Yoshua Bengio. Reweighted Wake-Sleep. *International Conference on Learning Representations*, 2015.

[2] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

[3] Uber AI Labs. Pyro: Deep Universal Probabilistic Programming. http://pyro.ai, 2017.

[4] Tuan Anh Le, Atılım Güneş Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

[5] Tuan Anh Le, Adam R. Kosiorek, N. Siddharth, Yee Whye Teh, and Frank Wood. Revisiting Reweighted Wake-Sleep. *arXiv:1805.10469 [cs, stat]*, May 2018.

[6] Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: A higher-order probabilistic programming platform with programmable inference. *arXiv*, pages 78–78, March 2014.

[7] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. Nested sequential monte carlo methods. In *International Conference on Machine Learning*, pages 1292–1301, 2015.

[8] Praveen Narayanan, Jacques Carette, Wren Romano, Chung-chieh Shan, and Robert Zinkov. Probabilistic inference by program transformation in Hakaru (system description). In *International Symposium on Functional and Logic Programming*, pages 62–79. Springer, 2016.

[9] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, June 2014. PMLR.

[10] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3528–3536. Curran Associates, Inc., 2015.

[11] Adam Ścibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, and Zoubin Ghahramani. Denotational Validation of Higher-order Bayesian Inference. *Proc. ACM Program. Lang.*, 2(POPL):60:1–60:29, December 2017.

[12] N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5927–5937, 2017.

[13] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv:1610.09787 [cs, stat]*, October 2016.

[14] Robert Zinkov and Chung-chieh Shan. Composing inference algorithms as program transformations. *Uncertainty in Artificial Intelligence*, 2017.