Rapid Model Comparison by Amortizing Across Models

Lily Zhang
Indico Data Solutions

Michael C. Hughes
Tufts University

Abstract
Comparing the inferences of diverse candidate models is an essential part of model checking and escaping local optima. We introduce a novel variational inference method which can perform fast and reliable approximate inference across models of the same architecture. Our Any-Parameter Encoder (APE) modifies the encoder usually used in amortized inference to take in both the data and the model as input, allowing posterior inference across data and models to be performed in a single forward pass. In experiments comparing candidate topic models for synthetic data and product reviews, we show that our Any-Parameter Encoders yield posteriors comparable to more expensive methods in a fraction of the time.

Keywords: Variational inference, amortized inference, model comparison, topic models

1. Introduction
An important step in extracting insight from data using Bayesian hierarchical models is model comparison. While many types of comparison are possible (Gelman et al., 2013)\(^1\), we focus on within-model parameter comparison: Given possible candidate parameters, \(\theta_1, \theta_2, \ldots, \theta_m, \ldots\), all in the same vector space \(\Theta \subseteq \mathbb{R}^D\), which is best at explaining a given dataset of \(N\) examples \(\{x_n\}_{n=1}^N\). This evaluation requires estimating the posterior \(p(h_n|x_n, \theta_m)\) over hidden variables \(h_n\) at each example \(n\) and model \(m\), which can be prohibitively expensive for enormous datasets. In this paper, we develop new variational inference tools that enable rapid-yet-effective within-model comparisons.

Within-model comparison is important in many practical tasks. Our running example will be topic models of text data (Blei, 2012). To improve interpretability, an analyst may inspect an estimated \(\theta\) and then suggest parameters \(\theta'\), e.g. removing “intruders” (Chang et al., 2009). To perform robust learning of parameters \(\theta\) and escape the local optima common in non-convex objectives for latent variable models (Roberts et al., 2016), many algorithms propose data-driven transformations of the current solution \(\theta\) into a new candidate \(\theta'\). Examples include split-merge proposal moves (Ueda and Ghahramani, 2002; Jain and Neal, 2004) or evolutionary algorithms (Sundararajan and Mengshoel, 2016). Across these scenarios, new candidates \(\theta'\) arise repeatedly over time, and inference of hidden variables for each is essential to assess fitness yet expensive to perform for large datasets.

Our contribution is the Any-Parameter Encoder (APE), which amortizes per-example posterior inference across both models \(\theta_m\) and data \(x_n\). We are inspired by efforts to scale a single model to large datasets by using an encoder neural network (NN) to amortize

\(^1\) Between-model comparisons (e.g. mixtures vs. factor analysis) are challenging and left to future work.

© L. Zhang & M.C. Hughes.
posterior inference (Rezende et al., 2014; Kingma and Welling, 2014). Our key idea is that to generalize across models, we feed both global parameter vector $\theta_m$ and data $x_n$ as input to the encoder. APE is applicable to any continuous hidden variable model amenable to variational autoencoders (VAEs) (Kingma and Welling, 2014).

2. Methods

Generative Model. We consider a general family of probabilistic models that use global parameter vector $\theta_m$ to generate continuous hidden variables $h_n$ and observations $x_n$ via a factorized distribution: $\prod_{n=1}^{N} p(h_n|\theta_m)p(x_n|h_n, \theta_m)$. Our goal is to estimate each example’s local posterior $p(h_n|x_n, \theta_m)$ accurately, for a range of model parameters $\theta_1, \theta_2, \ldots \in \Theta$.

Topic Models. As a sample application of APE, we focus on the Logistic Normal topic model from Srivastava and Sutton (2017). Given known vocabulary size $V$, we observe $N$ documents represented by count vectors $x_n$ (vector of size $V$ counting the types of all $T_n$ words in document $n$). We model each $x_n$ as a mixture of $K$ possible topics. Let hidden variable $h_{nk}$ be the probability that any word in document $n$ is produced by topic $k$. We draw $h_n \in \Delta^K$ from a Logistic Normal prior, with mean and covariance chosen to match a Dir(0.01) prior, as in Hennig et al. (2012). We draw the word-count vector $x_n \sim \text{Mult}(T_n, \sum_{k=1}^{K} h_{nk}\lambda_k)$; this is a document-specific mixture of the each topic’s word distribution $\lambda_k \in \Delta^V$, weighted by $h_n$. Our global parameter is the topic-word probabilities $\theta = \{\lambda_k\}_{k=1}^{K}$.

VI Approximations for the Single Example Posterior. While the true posterior $p(h_n|x_n, \theta_m)$ is usually intractable, we use variational inference (VI) (Wainwright and Jordan, 2008) to approximate it. We choose a simpler density $q(h_n|\lambda_n)$, and optimize parameter $\lambda_n$ to minimize KL divergence from the true posterior. Inference reduces to the well-known evidence lower bound (ELBO) optimization problem given example $x_n$ and model $\theta_m$:

$$\text{Inference: } \lambda_n \leftarrow \arg \max_{\lambda_n} \mathbb{E}_q [\log p(x_n, h_n|\theta_m) - \log q(h_n|\lambda_n)],$$

(1)

Given several parameters of interest, we can perform model comparison by solving the above optimization problem separately for each $\theta_m$. However, this is expensive. Solving Eq. (1) for a model $\theta_m$ requires dozens of iterative updates of gradient ascent for each example.

VI Amortized across Data Examples. Previously, Rezende et al. (2014) and Kingma and Welling (2014) have sped up inference by setting local, per-example $q$-parameters $\lambda_n$ to the output of a global neural network instead of an iterative optimization procedure. The “Standard” encoder network, with global parameter vector $\phi$, takes input data $x_n$ and produces a valid $q$-parameter $\lambda^{\text{NN}}(x_n)$. Inference for example $n$ reduces to one forward pass: $\lambda_n \leftarrow \lambda^{\text{NN}}(x_n)$. This approach is faster than solving Eq. (1) but is expensive for model comparison, because for each parameter $\theta_m$ of interest we must train separate specialized NN-parameters $\phi_m$. In terms of quality, the best case ELBO score of $\lambda_{\phi}^{\text{NN}}$ will be the same as optimal solutions to Eq. (1). Typically the encoder will be worse (Krishnan et al., 2018).

Contribution: VI Amortized over Model Parameters. Our goal is to enable rapid estimation of posteriors $p(h_n|x_n, \theta_m)$ for many possible parameters $\theta_1, \theta_2, \ldots \in \Theta$ (not all known in advance). We thus consider a family of approximating densities $q$ that explicitly conditions on both a given data vector $x_n$ and the query parameter vector $\theta_m$. 
Table 1: Comparison of per-word posterior predictive log likelihood (higher is better) and inference time (lower is better) for heldout test sets of 300 document-model combinations. 'Agreement' measures how often a method’s ELBO ranking of pairs $\theta, \theta'$ matches VI’s ranking.

Again, we use a neural network to transform these inputs into the variational parameters, $\lambda_n \leftarrow \lambda_{NN}^N(x_n, \theta_m)$. We call this the Any-Parameter Encoder. Unlike the earlier Standard Encoder, which trains $\phi$ for one specific $\theta$, our approach can directly generalize to many $\theta$.

Designing Encoder Architectures. Given the difficulty of posterior inference even for a single-parameter setting, developing an effective Any-Parameter Encoder requires careful selection of a specific NN architecture $A$ that can combine its two inputs, data $x_n$ and model $\theta$, to produce high-ELBO posteriors. We found that naively concatenating the data vector and model vector as the input to a NN was difficult to train effectively. We reason that such an architecture is ill-suited to the task given that the primary reasoning that needs to take place is that of the interaction between the document and relevant parts of the model, a logic that is difficult to learn without additional inductive biases incorporated into the network. Thus, we took inspiration from the specialized structure of the topic model. Given $V$-vector $x_n$ and arranging $\theta$ as a $K \times V$ topic-word matrix, we compute the product $\theta x_n$, a $K$-length vector where entry $k$ will be large if topic $\theta_k$ has high overlap with words in $x_n$. We additionally concatenate the model vector $\theta$ as input, allowing potential interactions between topics. Our network consists of 2 feedforward layers as in Srivastava and Sutton (2017).

Training the Encoder. Training our encoder parameters $\phi$ requires an available set of $M$ parameter vectors $\{\theta_m\}_{m=1}^M$ of interest. We choose these to be representative of the subset of $\Theta$ we wish to generalize well to. We then maximize ELBO across all $M$ models:

$$\text{Enc. Training: max}_{\phi \in A} \mathbb{E}_q \left[ \sum_{m=1}^M \sum_{n=1}^N \log p(x_n, h_n | \theta_m) - \log q(h_n | \lambda_{NN}^N(x_n, \theta_m)) \right]$$

We use stochastic gradient ascent to solve for $\phi$, using the reparameterization trick to estimate gradients for a minibatch of examples and models at each step.

Related Work. Recent efforts on meta-learning VAEs (Wu et al., 2019; Gordon et al., 2019) also try to generalize across models, but their encoders take sampled datasets from a model as input, not model parameters $\theta$. Our approach offers a simpler, faster, more direct approach to model comparison. Amortization for topic model inference goes back to Yao et al. (2009), though Srivastava and Sutton (2017) first used VAE-inspired approaches.

3. Experiments: Topic Models for Synthetic Data and Product Reviews

Baselines. We compare our Any-Parameter Encoder to several baselines for inference, all implemented for Logistic Normal topic models in Pyro (Bingham et al., 2018) and PyTorch (Paszke et al., 2017). First, Variational Inference (VI) indicates a gradient ascent procedure to optimize Eq. (1). Second, we use Standard encoder VAEs, as formulated

\[
\text{standard encoder: } \log p(x_n | \theta_m) - \log q(h_n | \lambda_{NN}^N(x_n, \theta_m))
\]
for topic models in Srivastava and Sutton (2017). This encoder is specialized to a single parameter $\theta$. All variational methods choose $q$ to be a Logistic Normal with diagonal covariance. Finally, we run Pyro’s off-the-shelf implementation of Hamiltonian Monte Carlo with the No U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014), though we expect specialized implementations to be more performant.

**Synthetic Data Experiments.** We consider a $V = 100$ vocabulary dataset inspired by the “toy bars” of Griffiths and Steyvers (2004). Using $K = 20$ true topics $\theta^*$, we sample a 500-document dataset. We consider $M = 50,000$ possible model parameters $\{\theta_m\}_{m=1}^{M}$, sampled from a symmetric, sparse Dirichlet prior over the vocabulary. Typical $\theta_m$ look unlike the true topics $\theta^*$, as shown in the supplement, so inference must handle diversity well. We train our APE on 25 million possible document-$\theta$ pairs in under 6 hours, then evaluate on unseen document-$\theta$ pairs drawn from the same generative process.

**Product Reviews Experiments.** Next, we model online product reviews (Blitzer et al., 2007). We set vocabulary to the $V = 3000$ most frequent words. We subsample 6,343 documents (80% of the corpus). We use models with $K = 30$ topics. We generate training topics in the same way as in the synthetic data experiments, and we evaluate on test topics found via Gibbs sampling over different runs.

**Results: Quality-vs-Time Tradeoff.** Results for both datasets are in Table 1 and Fig. 1. While the Standard Encoder understandably fails to generalize across models, our Any-Parameter Encoder achieves quality close to VI (likelihood -4.24 vs. VI’s -3.90 on synthetic data, -5.968 vs. VI’s -5.586 on product reviews), while $>1000x$ faster than NUTS on the synthetic data and over $100x$ faster than VI on product reviews.

**Results: Agreement in model comparison.** Motivated by the need to rapidly assess proposal moves that escape local optima, we gather 10 different models and measure whether each encoder’s ranking of a pair $\theta, \theta'$ on the test set agrees with VI’s ranking. Table 1 shows that APE agrees with VI in 87% of 45 cases, while Standard Encoder agrees just 7% of the time. This suggests APE could be more trustworthy for accept/reject decisions.
4. Conclusion

Across all datasets, our Any-Parameter Encoder produces posterior approximations that are nearly as good as expensive VI, but up to 100x faster. Future work opportunities include trying more flexible models, exploring better encoder architectures, and handling Bayesian nonparametric models where the size of $\theta$ changes during training (Hughes et al., 2015).

References


Appendix A. Experiment Details

A.1. Synthetic Data Generation

We generate a set of different models \( \{\theta_0, \theta_1, ..., \theta_M\} \) from a symmetric Dirichlet prior with \( \alpha = 0.1 \). We train our Any-Parameter Encoder in random batches of document-topic combinations. With 500 documents and 50,000 topics (i.e. \( D = 500, M = 50,000 \)), we have 25 million combinations in total.

The topics used to generate the synthetic data represent “toy bars”, inspired by (Griffiths and Steyvers, 2004). See Figure 2 for a visualization. We use this same toy bars-biased prior to generate all our topics in the holdout set, though the order of the topics is random. See Figures 2(a) and 2(b) for a visualization of the true topics \( \theta^* \) and a representative model \( \theta \) in the training set, respectively. Figure 2(c) shows sample reconstructions from each inference technique (rows) on different document-topic combinations (columns). The reconstructions are a qualitative assessment of the ability of the inference method to generate good posteriors and demonstrate the ability of APE to reach close-to comparable results with more expensive methods.

A.2. Topic Models for Bag-of-Words Data

We focus on experimental implementation of the logistic Normal topic model studied by Srivastava and Sutton (2017). Given a bag-of-words dataset with \( N \) documents and \( V \) possible vocabulary terms, at each document (indexed by \( n \)) we observe the count vector \( x_n \) (vector of size \( V \) with non-negative entries). The model assumes that this data is generated as a mixture of \( K \) possible topics. We have global parameters: \( \theta = \{\alpha, \beta\} \), where \( \alpha > 0 \) defines a prior on topic appearance and \( \beta \) define the topic-word probabilities. We’ll assume \( \alpha \) is known, so the effective global parameter vector is \( \theta = \beta \).

Hidden variable \( h_{nk} \) defines the probability that any word in document \( n \) is produced by topic \( k \). Thus, \( h_n \) is a vector of size \( K \) that sums to one. We draw \( h_n \) from a Logistic Normal prior (with mean and covariance chosen to match a Dir(\( \alpha \)) prior, as in (Hennig et al., 2012)). The choice of Logistic Normal prior enables the use of the reparametrization trick to reduce the variance of the gradient estimate one would otherwise see with alternative estimators. We then draw the word-count vector \( x_n \sim \text{Mult}(T_n, \sum_{k=1}^K h_{nk}\beta_k) \), which is a document-specific mixture of the each topic’s word distribution \( \beta_k \in \Delta^V \). The number of tokens \( T_n \) is known in advance.

\[
\begin{align*}
    h_n &\sim \text{LogisticNormal} ([\mu_1(\alpha), \ldots, \mu_K(\alpha)], \text{diag}([v_1(\alpha), \ldots, v_K(\alpha)])) \\
    \mu_k(\alpha) &= \log \alpha_k - \frac{1}{K} \log \alpha_k, \quad v_k(\alpha) = \frac{1}{\alpha_k} \left(1 - \frac{2}{K}\right) + \frac{1}{K^2} \sum_k \frac{1}{\alpha_k}.
\end{align*}
\]

A.3. Encoder Training Details

We train for 2 epochs with a batch size of 100.

For training both APE and the Standard encoder on the synthetic data, we use Adam with an exponential decay learning schedule, a starting learning rate of 0.01, and a decay rate of .8 every 50,000 steps. We find that this schedule tends to be fairly robust; these
Rapid Model Comparison by Amortizing Across Models

Figure 2: Toy Bars Qualitative Visualizations. We represent document and the individual topics of a model by a 10x10, where each pixel in the plot represents a word in the vocabulary. The darkness of each pixel represents the density of the word (for topics) or the relative occurrence of the word (for documents). (a) and (b) show topics, and (c) shows documents and corresponding reconstructions from posteriors obtained through the various methods.

Appendix B. Evaluation Details

We compare our Any-Parameter Encoder (APE) inference method to several baseline methods, all implemented for Logistic Normal topic models using Pyro (Bingham et al., 2018)
with PyTorch backend (Paszke et al., 2017). We use CPU for all methods during evaluation for a consistent time comparison across methods.

B.1. Inference Implementation Details.

**Approximate Posterior Family.** For all variational methods, we chose approximate posterior $q$ to belong to the Logistic Normal family parameterized by a mean vector $\mu_n \in \mathbb{R}^K$ and a diagonal covariance matrix with diagonal $\sigma^2_n \in \mathbb{R}_+^K$.

$$
q(h_n | \lambda_n = \{\mu_n, \sigma_n\}) = \text{LogisticNormal}_K(\mu_n, \text{diag}(\sigma^2_n))
$$

For encoder methods, the parameters $\{\mu_n, \log \sigma^2_n\}$ are the output of a shared encoder NN. For VI, these are free parameters of the optimization problem.

**Variational Inference (VI).** We perform using gradient ascent to maximize the objective in Eq. (1), learning a per-example mean and variance variational parameter. We run gradient updates until our moving average loss (window of 10 steps) has improved by less than 0.001% of its previous value. For our VI runs from random initializations, we use the Adam optimizer with an initial learning rate of .01, decaying the rate by 50% every 5000 steps. For our warm-started runs, we use an initial learning rate of 0.0005. In practice, we ran VI multiple times with different learning rate parameters and took the best one. Table 1 only reports the time to run the best setting, not the total time which includes various restarts.

**Standard encoder.** We use a standard encoder that closely matches the VAE for topic models in Srivastava and Sutton (2017). The only architectural difference is the addition of a temperature parameter on the $\mu_n$ vector before applying the softmax to ensure the means lie on the simplex. We found that the additional parameter sped up training by allowing the peakiness of the posterior to be directly tuned by a single parameter. We use a feedforward encoder with two hidden layers, each 100 units. (We found that with more layers and hidden units, the encoder unsurprisingly was even more prone to overfitting.)

The total number of trainable parameters in the model is 24,721 on the synthetic data and 316,781 on the real data; this is compared to 316,781 and 9,019,781 parameters for APE.

**NUTS.** The Hamiltonian Monte Carlo (HMC) with the No U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014) is considered a gold standard method for delivering high-quality posterior estimates. Regarding the quality of the method, we find that its slightly lower posterior predictive log likelihood relative to VI is due to its wider posteriors. Regarding speed, we find it is quite slow and consequently warm-start the NUTS sampler using VI to encourage rapid mixing. We are aware that there exist faster, more specialized implementations, but we decided to keep our tooling consistent for scientific purposes. We use a step size of 1 adapted during the warmup phase using Dual Averaging scheme.

B.2. TLDR

We develop VAEs where the encoder takes a model parameter vector as additional input, so we can do rapid inference for many models.