# Deep Probabilistic Graphical Modeling

Adji Bousso Dieng

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

# Abstract

Deep Probabilistic Graphical Modeling

Adji Bousso Dieng

Probabilistic graphical modeling (PGM) provides a framework for formulating an interpretable generative process of data and expressing uncertainty about unknowns. This makes PGM very useful for understanding the phenomena underlying data and for decision making. PGM has been successfully used in domains where interpretable inferences are key, e.g. marketing, medicine, neuroscience, and the social sciences. However PGM tends to lack flexibility. This lack of flexibility makes PGM often inadequate for modeling large-scale high-dimensional complex data and performing tasks that do require flexibility, e.g. vision and language applications.

Deep learning (DL) is an alternative framework for modeling and learning from data that has seen great empirical success in recent years. DL is very powerful and offers great flexibility, but it lacks the interpretability and calibration of PGM.

This thesis develops *deep probabilistic graphical modeling* (DPGM). DPGM consists in leveraging DL to make PGM more flexible. DPGM brings about new methods for learning from data that exhibit the advantages of both PGM and DL.

We use DL within PGM to build flexible models endowed with an interpretable latent structure. One family of models we develop extends exponential family principal component analysis (EF-PCA) using neural networks to improve predictive performance while enforcing the interpretability of the

latent factors. Another model class we introduce enables accounting for long-term dependencies when modeling sequential data, which is a challenge when using purely DL or PGM approaches. This model class for sequential data was successfully applied to language modeling, unsupervised document representation learning for sentiment analysis, conversation modeling, and patient representation learning for hospital readmission prediction. Finally, DPGM successfully solves several outstanding problems of probabilistic topic models, a widely used family of probabilistic graphical models.

Leveraging DL within PGM also brings about new algorithms for learning with complex data. We develop *reweighted expectation maximization* (REM), an algorithm that unifies several existing maximum likelihood-based algorithms for learning models parameterized by deep neural networks. This unifying view is made possible using expectation maximization, a canonical inference algorithm in PGM. We also develop *entropy-regularized adversarial learning*, a learning paradigm that deviates from the traditional maximum likelihood approach used in PGM. From the DL perspective, entropy-regularized adversarial learning provides a solution to the long-standing mode collapse problem of generative adversarial networks (GANS), a widely used DL approach.

# Table of Contents

# List of Tables

# List of Figures

## Acknowledgements

I take this opportunity to thank people and organizations without whom this dissertation would not be possible.

I thank my advisors David Blei and John Paisley for their unwavering support and for offering me the flexibility to pursue my own research interests. I was lucky to have them as advisors. I also thank Tian Zheng, Kyunghyun Cho, and John Cunningham for taking time to be part of my thesis committee.

I thank Columbia and Google for awarding me fellowships to support my PhD. I thank Microsoft for granting me Azure cloud credits to facilitate my research.

I am very fortunate to have had the opportunity to collaborate with many wonderful people: Francisco Ruiz, Michalis Titsias, Chong Wang, Jianfeng Gao, Dustin Tran, Rajesh Ranganath, Jaan Altosaar, Yoon Kim, and Sasha Rush.

I am grateful to my mentors Yann LeCun and Kyunghyun Cho. I also thank all the people who hosted and mentored me during internships at Microsoft Research, Facebook AI Research, and DeepMind: Chong Wang, Jianfeng Gao, Yann LeCun, Kyunghyun Cho, Lei Yu, and Chris Dyer. I thank Alp Kucukelbir for helping me find my first PhD internship. I thank Philip Protter, Peter Orbanz, Kyle Cranmer, Hugo Larochelle, Danilo Rezende, Jasper Snoek, Scott Linderman, and Wesley Tansey for their support and/or advice.

I am thankful to several people I met in the course of my PhD whom I am fortunate to have as

friends: Makhtar Ba, Yassin Choye, Ibrahima Niang, Saliou Diallo, Nadia Raynes, Maimouna Diagne, Kashif Yusuf, Jing Wu, Emma Zhang, Peter Lee, Laurent Dinh, Zack Lipton, Cathy Seya, Yoon Kim, Siva Reddy, Maja Rudolph, Diana Cai, and Rajarshi Das.

I would like to thank people I met before starting my PhD and without whom I wouldn't be able to pursue a PhD in the U.S. I thank Dr. Cheick Modibo Diarra for awarding me a scholarship to study abroad through his Pathfinder Foundation. I thank Mary Levy-Bruhl, Remi Barbet-Massin, and Jean Pierre Foulon from Lycee Henri IV. I thank Eric Moulines and Francois Roueff whom I had the fortune to learn Statistics and Probability from while at Telecom ParisTech. I thank Martin Wells and David Lifka who supported me and offered me the opportunity to work on solving concrete problems using data and computing during the time I spent at Cornell.

Finally, I thank Patrick Guelah, Ousmane Kane, Mada Niang, Tening Diouf, Sarah Eugene, Arthur Bauer, Laetitia Gerin, and Cherif Gassama for their long-lasting friendship. I thank Jarra Jagne and Tonton Goumbala for their support. I owe a great deal to my family for providing me with plenty of love and moral support.

## Dedication

To my late father.

To my mother, who gave me the gift of education.

# Introduction

Probabilistic machine learning (PML) turns data into knowledge about the world. This involves collecting data, specifying a model, fitting this model to the data, and performing evaluation on some criterion of interest, e.g. predictive performance.

Probabilistic graphical modeling (PGM) is an approach to PML that specifies a model by specifying an interpretable generative process of data. This generative process often involves sampling a set of *latent variables* from some prior distribution and then conditioning on these latent variables to generate data. The latent variables carry meaning; they represent the hidden structure underlying the data. Learning with PGM involves estimating any parameters involved in specifying the model and discovering the hidden structure by performing posterior inference, i.e. learning the conditional distribution of the latent variables given the data. Often the posterior is intractable and we resort to variational inference, which uses optimization to find a tractable proxy for the true posterior. PGM has been widely applied, for example to discover themes underlying a corpus of documents (Blei et al., 2003), to model speech (Rabiner, 1989), to understand user preferences for recommendation (Wang & Blei, 2011), to learn interaction patterns between different countries (Schein et al., 2015), etc.

Despite its wide application, PGM may lack flexibility. This has prevented its use in applications that do require flexibility, for example in vision and language applications. This thesis develops *deep probabilistic graphical modeling* (DPGM), which consists in leveraging deep learning (DL) to bring flexibility to PGM. Leveraging DL often means using neural networks to parameterize conditional

distributions within a latent-variable model. DPGM is agnostic to the choice of the architecture of these underlying neural networks. Therefore, the methodologies we develop in this thesis are amenable to more recent neural network architectures developed by the DL community and any future innovations in the development of neural network architectures. The promise of DPGM is to birth methodologies that enjoy the interpretability and calibration of PGM, and the flexibility of DL. Interpretability, by means of composing latent variables and parameters using inductive biases from domain knowledge, offers the ability to control the behavior of artificial intelligence (AI) systems. This controllability is key to a safe application of AI to critical domains such as healthcare, autonomous and automated vision and language systems, and science.

The rest of the thesis is organized as follows.

In Chapter 1 we review the foundations for DPGM. We first review PGM, with a focus on the latent variable approach to PGM. We describe exponential families as a unifying framework for representing distributions over random variables, observed or latent. We then describe several examples of PGMs: exponential family principal component analysis (EF-PCA), latent Dirichlet allocation (LDA), and dynamic latent Dirichlet allocation (D-LDA). We then discuss variational inference, a framework for approximating posterior distributions over latent variables. The second part of this chapter is a review of DL. We first describe several neural network architectures and then review two DL methodologies that are key to DPGM, auto-encoding for dimensionality reduction and word embeddings. The final section of this chapter is a discussion of a line of work that combines neural networks with latent variables. In particular, we will describe variational autoencoders (VAEs) and discuss *latent variable collapse*, a phenomenon that arises when parameterizing conditional distributions of a latent variable model with deep neural networks.

In Chapter 2 we first describe three desiderata for DPGM and then introduce several instances of DPGM. One model class we introduce, called *deep generative skip models*, extends EF-PCA using neural networks. Deep generative skip models achieve superior predictive performance and learn interpretable latent factors (Dieng et al., 2019a). A second model class we introduce, called

TopicRNN, marries latent variables and neural networks to model sequential data, addressing the long-term dependency issue encountered by purely DL and PGM approaches such as recurrent neural networks (RNNS) and hidden Markov models (HMMS). The model class defined by TopicRNN encodes inductive biases that have been shown useful for language modeling (Dieng et al., 2016), conversation modeling (Wen & Luong, 2018), unsupervised document representation learning (Dieng et al., 2016), and patient representation learning for hospital readmission prediction (Xiao et al., 2018a). Finally, we describe how to leverage word embeddings, a successful DL approach that consists in representing words as continuous low-dimensional vectors, to solve several problems that pertain to probabilistic topic models, one of the most important PGM class of models in terms of domain application (Dieng et al., 2019c,b).

The models introduced in Chapter 2 have intractable likelihoods. They are fit by maximizing a lower bound of the log marginal likelihood of the data, called the evidence lower bound (ELBO), using variational inference (VI). This is the approach of VAEs (Gershman & Goodman, 2014; Kingma & Welling, 2014; Rezende et al., 2014). Since the lower bound is intractable, VAEs use a Monte Carlo approximation of it for learning. VAEs are prone to two main problems. First, the ELBO they optimize may be a lose lower bound to the log-marginal likelihood of the data, which may hurt generalization performance. Second, VAEs often suffer from *latent variable collapse*, a phenomenon in which the learned latent variables do not represent good summaries of the data. In Chapter 3 we propose an alternative approach for learning DPGMs called *reweighted expectation maximization* (REM). REM optimizes a better approximation of the log marginal likelihood of the data (Dieng & Paisley, 2019). It uses self-normalized importance sampling with moment matching to maximize the log marginal likelihood. REM generalizes several existing algorithms that are based on maximum likelihood, such as the importance weighted auto-encoder (IWAE) (Burda et al., 2015a) and reweighted wake-sleep (RWS) (Bornschein & Bengio, 2014). REM leads to better generalization performance and yields more interpretable latent variables.

Leveraging DL for PGM offers the opportunity to take advantage of algorithmic innovations in DL to

learn PGMS. In Chapter 4 we build on generative adversarial networks (GANS) and develop *entropy-regularized adversarial learning*. Entropy-regularized adversarial learning provides an alternative to maximum likelihood for fitting DPGMS. From the perspective of DL, entropy-regularized adversarial learning constitutes a solution to the *mode collapse* problem of GANS (Dieng et al., 2019d). Addressing this mode collapse problem is important because under mode collapse, GAN outputs lack diversity. This lack of diversity in outputs negatively affects the use of GANS for data augmentation, but also its application in healthcare and branches of machine learning (ML) such as *Fairness*.

We conclude with a discussion of the contributions of this thesis and possibilities for future work.

# Chapter 1: Foundations

In this chapter we lay the foundations for *deep probabilistic graphical modeling* (DPGM) by reviewing probabilistic graphical modeling (PGM) and deep learning (DL). We end the chapter with a discussion of probabilistic conditioning with neural networks and the latent variable collapse issue that might arise from it.

## 1.1 Probabilistic Graphical Modeling

PGM provides a useful framework for extracting knowledge from data. For example, a PGM fit on a corpus of documents can tell us about the thematic structure underlying the documents. The PGM approach to learning from data is to mimic the true process that generated the data. When specifying a generative process for data, to approximate the true data generating process, PGM offers the ability to incorporate our prior knowledge about the phenomenon under study. For example, when studying a corpus of documents, PGM allows us to integrate the knowledge that there is a set of topics discussed by all the documents in the corpus and that a given document expresses these topics at different lengths.

### 1.1.1 Latent Variables & Interpretability

Consider observed a set of $N$ i.i.d data points. Denote them by $\mathbf{x}_1, \ldots, \mathbf{x}_N$. The true phenomenon that generated these data is unknown and we want to learn about it. This will allow us to understand and make discoveries about the phenomenon underlying the data, perform prediction, and simulate new data. The PGM approach is to posit the existence of a set of *latent variables*, unobserved

random variables that represent the hidden structure underlying the observed data. These latent variables are composed with the observations to form an interpretable generative process of data that approximates the true underlying data generating process. Often there are two sets of latent variables: *global* latent variables and *local* latent variables. Global latent variables capture the stable aspects of the underlying data generating process; they are shared across all the observations. Local latent variables express the singularities of each observation. For example consider a dataset of images of human faces. Global latent variables may represent the features of a face, e.g. eyes, lips, nose, cheeks, hair. Local latent variables will capture instantiations of these features; for example one image might depict brown eyes and dark hair whereas another image may depict green eyes and red hair.

Denote by $\boldsymbol{\beta}$ the global latent variables and by $\mathbf{z}_1, \ldots, \mathbf{z}_N$ the local latent variables in a PGM. The generative process specified by the PGM implies a joint distribution over data and latent variables,

$$p(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \boldsymbol{\beta}) = p(\boldsymbol{\beta}) \cdot \prod_{i=1}^{N} p(\mathbf{z}_i|\boldsymbol{\beta}) \cdot p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\beta}). \tag{1.1}$$

The distributions $p(\boldsymbol{\beta})$ and $p(\mathbf{z}_i|\boldsymbol{\beta})$ are the priors over the global and $i^{th}$ local latent variable respectively. Their distributional forms can be chosen depending on the problem under study. The distribution $p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\beta})$ describes how to generate the $i^{th}$ observation $\mathbf{x}_i$ by conditioning on $\boldsymbol{\beta}$ and $\mathbf{z}_i$. Our knowledge about the phenomenon under study is also expressed in terms of conditional independencies between the different variables, observed and latent. For example, the conditional distribution of $\mathbf{x}_i$ may only depend on $\mathbf{z}_i$.

### 1.1.2 Exponential Families

The exponential family provides a unifying framework for specifying probability distributions over random variables. The distributions mentioned above can be chosen to be in the exponential family.

*Table 1.1.* Expressions of the sufficient statistic, the natural parameter, and the log normalizer for different members of the exponential family.

| Distribution | Parameter $\theta$ | $t(\mathbf{x})$ | $\eta(\theta)$ | $A(\eta(\theta))$ |
|---|---|---|---|---|
| Bernoulli | $p$ | $\mathbf{x}$ | $\log \frac{p}{1-p}$ | $\log(1+\exp(\eta(\theta)))$ |
| Gaussian | $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ | $\mathbf{x}, \mathbf{x}^2$ | $(\frac{\boldsymbol{\mu}}{\boldsymbol{\sigma}^2}, -\frac{1}{2\boldsymbol{\sigma}^2})$ | $-\frac{\eta(\theta)_1^2}{4\eta(\theta)_2} - \frac{1}{2}\log(-2\eta(\theta)_2)$ |
| Poisson | $\lambda$ | $\mathbf{x}$ | $\log(\lambda)$ | $\exp(\eta(\theta))$ |
| Categorical | $p_{1:K}$ | $(\mathbb{I}(\mathbf{x}=1), \ldots, \mathbb{I}(\mathbf{x}=K))$ | $\log(p_{1:K})$ | $0$ |
| Dirichlet | $\alpha_{1:K}$ | $\log(\mathbf{x}_{1:K})$ | $\alpha_{1:K}$ | $\sum_{k=1}^K \log \frac{\Gamma(\eta_k(\theta))}{\Gamma(\sum_{k=1}^K \eta_k(\theta))^{\frac{1}{K}}}$ |
| Gamma | $(\alpha, \beta)$ | $(\log(\mathbf{x}), \mathbf{x})$ | $(\alpha-1, -\beta)$ | $\log \frac{\Gamma(\eta_1(\theta)+1))}{-\eta_2(\theta)^{\eta_1(\theta)+1}}$ |

Almost all of the distributions used in practice are members of the exponential family, for example Gaussian, Gamma, Poisson, Bernoulli, Categorical, and Dirichlet. Below is the formal definition of an exponential family.

**Definition 1** *A family of probability density functions $\mathcal{P} = \{p_\theta : \theta \in \Theta\}$ on a measure space $(\mathcal{X}, \mathcal{B}, \nu)$ is said to form an exponential family if*

$$p_\theta(\mathbf{x}) = \exp\left(\eta(\theta)^T t(\mathbf{x}) - A(\eta(\theta))\right)$$
$$A(\eta(\theta)) = \log \int \exp\left(\eta(\theta)^T t(\mathbf{x})\right) \nu(d\mathbf{x})$$

*where $A(\eta(\theta))$ is called the log partition function (or log normalizer), $\eta(\theta)$ is called the natural parameter, and $t(\mathbf{x})$ denotes the vector of sufficient statistics.*

Table 1.1 provides the expressions of the sufficient statistics, the natural parameter, and the log normalizer for several members of the exponential family.

### 1.1.3 Example: Exponential Family PCA

One canonical example of a PGM is exponential family principal component analysis (EF-PCA) (Tipping & Bishop, 1999; Collins et al., 2002). Assume observed $N$ i.i.d data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where $\mathbf{x}_i \in \mathbb{R}^D$. EF-PCA posits the following data generative process:

1. Draw global latents $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$
2. For each data point $i = 1 \ldots N$:
   (a) Draw local latent variable $\mathbf{z}_i \sim p(\mathbf{z})$
   (b) Draw data point $\mathbf{x}_i \sim \text{EF}(\eta_i = f(\boldsymbol{\beta}^\top \mathbf{z}_i))$

Here $\text{EF}(\eta)$ stands for an exponential family distribution with natural parameter $\eta$ and $f(\cdot)$ is a deterministic function that maps the dot product $\boldsymbol{\beta}^\top \mathbf{z}_i$ to the right space for the natural parameter. Note the local latent variables and the global latent variables interact linearly. This is a simplifying assumption that we will relax in Chapter 2. When fit to data, EF-PCA learns interpretable low-dimensional representations of data $\mathbf{z}_{1:N}$.

### 1.1.4 Example: Latent Dirichlet Allocation

Another canonical PGM is latent Dirichlet allocation (LDA) (Blei et al., 2003). LDA is a probabilistic generative model of documents. It posits $K$ topics $\beta_{1:K}$, each of which is a distribution over a vocabulary (a predefined set of words). LDA assumes each document comes from a mixture of topics, where the topics are shared across the corpus (they are global latent variables) and the mixture proportions are unique to each document (they are local latent variables). The generative process for each document is the following:

1. Draw topic proportion $\theta_d \sim \text{Dirichlet}(\alpha_\theta)$.
2. For each word $n$ in the document:
   (a) Draw topic assignment $z_{dn} \sim \text{Cat}(\theta_d)$.
   (b) Draw word $w_{dn} \sim \text{Cat}(\beta_{z_{dn}})$.

Here, $\text{Cat}(\cdot)$ denotes the categorical distribution. LDA places a Dirichlet prior on the topics,

$$\beta_k \sim \text{Dirichlet}(\alpha_\beta) \text{ for } k = 1, \ldots, K.$$

The concentration parameters $\alpha_\beta$ and $\alpha_\theta$ of the Dirichlet distributions are fixed model hyperparameters often chosen to achieve a certain level of sparsity. Note all the distributions in LDA are members of the exponential family.

LDA is a powerful model for document corpora. It has been extended in many ways and applied to many fields, such as marketing, sociology, political science, and the digital humanities. Boyd-Graber et al. (2017) provide a review.

### 1.1.5 Example: Dynamic Latent Dirichlet Allocation

Dynamic latent Dirichlet allocation (D-LDA) is an extension of LDA that allows topics to vary over time in order to analyze time-series corpora (Blei & Lafferty, 2006). The generative model of D-LDA differs from LDA in that the topics are time-specific, i.e., they are $\beta_{1:K}^{(t)}$, where $t \in \{1, \ldots, T\}$ indexes time steps. Moreover, the prior over the topic proportions $\theta_d$ depends on the time stamp of document $d$, denoted $t_d \in \{1, \ldots, T\}$. The generative process for each document is:

1. Draw topic proportions $\theta_d \sim \mathcal{LN}(\eta_{t_d}, a^2 I)$.
2. For each word $n$ in the document:
   (a) Draw topic assignment $z_{dn} \sim \text{Cat}(\theta_d)$.
   (b) Draw word $w_{dn} \sim \text{Cat}(\beta_{z_{dn}}^{(t_d)})$.

Here, $a$ is a model hyperparameter and $\eta_t$ is a latent variable that controls the prior mean over the topic proportions at time $t$. To encourage smoothness over the topics and topic proportions, D-LDA places random walk priors over $\beta_{1:K}^{(t)}$ and $\eta_t$,

$$\widetilde{\beta}_k^{(t)} \mid \widetilde{\beta}_k^{(t-1)} \sim \mathcal{N}(\widetilde{\beta}_k^{(t-1)}, \sigma^2 I) \text{ and } \beta_k^{(t)} = \text{softmax}(\widetilde{\beta}_k^{(t)})$$
$$\eta_t \mid \eta_{t-1} \sim \mathcal{N}(\eta_{t-1}, \delta^2 I).$$

The variables $\widetilde{\beta}_k^{(t)} \in \mathbb{R}^V$ are the transformed topics; the topics $\beta_k^{(t)}$ are obtained after mapping $\widetilde{\beta}_k^{(t)}$ to the simplex, via the softmax$(\cdot)$ function. The hyperparameters $\sigma$ and $\delta$ control the smoothness

of the Markov chains.

### 1.1.6 Posterior Inference

To discover the structure specified by all the models described above, we need to revert the generative process of data and compute the conditional distribution of the latent variables given the data. This conditional distribution is called the *posterior distribution* of the latent variables. Consider the canonical EF-PCA described earlier. The posterior distribution is

$$p(\boldsymbol{\beta}, \mathbf{z}_{1:N}|\mathbf{x}_{1:N}) = \frac{p(\boldsymbol{\beta}) \cdot \prod_{i=1}^{N} p(\mathbf{z}_i|\boldsymbol{\beta}) \cdot p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\beta})}{\int p(\boldsymbol{\beta}) \cdot \left[\prod_{i=1}^{N} p(\mathbf{z}_i|\boldsymbol{\beta}) \cdot p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\beta}) d\mathbf{z}_i\right] d\boldsymbol{\beta}} \tag{1.2}$$

For simple models, the posterior has an analytical form. For many models, this is not the case and we must find a way to approximate the posterior.

### 1.1.7 Variational Inference

Variational inference (VI) approximates the posterior using optimization. The idea is to posit a family of approximating distributions and then to find the member of the family that is closest to the posterior. Typically, closeness is defined by the Kullback-Leibler (KL) divergence between the approximating distribution and the true posterior.

Concretely, consider the canonical running example in Eq. 1.1. Denote by $q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})$ the approximating family, also called the *variational family*; it is indexed by $\boldsymbol{\lambda}$, the *variational parameters*. VI solves the following optimization procedure:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \text{KL}\left(q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) || p(\boldsymbol{\beta}, \mathbf{z}_{1:N}|\mathbf{x}_{1:N})\right). \tag{1.3}$$

The KL above is intractable because the posterior is intractable. However, we can write the KL as

follows,

$$\text{KL}\left(q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) || p(\boldsymbol{\beta}, \mathbf{z}_{1:N} | \mathbf{x}_{1:N})\right) = \log p(\mathbf{x}_{1:N}) - \mathbb{E}_{q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})}\left[\log \frac{p(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \boldsymbol{\beta})}{q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})}\right] \tag{1.4}$$

This expression of the KL reveals two things. First, because $\log p(\mathbf{x}_{1:N})$ does not depend on the parameters $\boldsymbol{\lambda}$ we want to optimize over, minimizing the KL is equivalent to maximizing the second term on the right hand side of Eq. 1.4. Second, because KL is nonnegative, the second term on the right hand side of Eq. 1.4, called the evidence lower bound (ELBO), is a lower bound of the log marginal likelihood of the data $\log p(\mathbf{x}_{1:N})$. The ELBO is a function of the data and the variational parameters $\boldsymbol{\lambda}$,

$$\text{ELBO}(\mathbf{x}_{1:N}, \boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})}\left[\log \frac{p(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \boldsymbol{\beta})}{q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})}\right] \leq \log p(\mathbf{x}_{1:N}) \tag{1.5}$$

The ELBO is tractable, or can be tractably approximated, if we specify a tractable density for $q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})$. There are many ways to specify the family $q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})$. One way to specify $q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})$ is to use the *mean field* assumption.

**Mean-field VI.** The mean field assumption decomposes the variational distribution $q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda})$ into a product of factors,

$$q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}_\beta) \cdot \prod_{i=1}^{N} q(\mathbf{z}_i; \boldsymbol{\lambda}_i) \tag{1.6}$$

where $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_\beta, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N)$. Using this decomposition, mean-field VI then maximizes the ELBO,

$$\text{ELBO}(\mathbf{x}_{1:N}, \boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\beta}; \boldsymbol{\lambda}_\beta) \prod_{i=1}^{N} q(\mathbf{z}_i; \boldsymbol{\lambda}_i)}\left\{\log \frac{p(\boldsymbol{\beta})}{q(\boldsymbol{\beta})} + \sum_{i=1}^{N} \log \frac{p(\mathbf{z}_i | \boldsymbol{\beta})}{q(\mathbf{z}_i; \boldsymbol{\lambda}_i)} + \sum_{i=1}^{N} \log p(\mathbf{x}_i | \mathbf{z}_i, \boldsymbol{\beta})\right\} \tag{1.7}$$

For certain classes of models, e.g. conditionally conjugate models (Ghahramani & Beal, 2001), the ELBO can be optimized using coordinate ascent or stochastic optimization. Blei et al. (2017a) provide a review.

**Black-box variational inference (BBVI).** For a general class of models, the ELBO can be maximized using BBVI (Paisley et al., 2012; Ranganath et al., 2014). For simplicity, let's lump all latent variables into $\mathbf{z}$ and all the observations into $\mathbf{x}$. The ELBO is,

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})} \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})\right] \tag{1.8}$$

BBVI optimizes the ELBO with respect to $\boldsymbol{\lambda}$ using a Monte Carlo approximation of its gradients.

*Score gradients.* We can compute the gradient of the ELBO with respect to $\boldsymbol{\lambda}$ as follows.

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}}\text{ELBO} &= \nabla_{\boldsymbol{\lambda}} \int \left[q(\mathbf{z}; \boldsymbol{\lambda}) \log p(\mathbf{x}, \mathbf{z}) - q(\mathbf{z}; \boldsymbol{\lambda}) \log q(\mathbf{z}; \boldsymbol{\lambda})\right] d\mathbf{z} && (1.9)\\
&= \int \left[\nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) \log p(\mathbf{x}, \mathbf{z}) - \nabla_{\boldsymbol{\lambda}} \left(q(\mathbf{z}; \boldsymbol{\lambda}) \log q(\mathbf{z}; \boldsymbol{\lambda})\right)\right] d\mathbf{z} && (1.10)\\
&= \int \left[\log p(\mathbf{x}, \mathbf{z}) \nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) - \log q(\mathbf{z}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) - q(\mathbf{z}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda})\right] d\mathbf{z} && (1.11)\\
&= \int \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})\right] \nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z} - \int \nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z} && (1.12)\\
&= \int q(\mathbf{z}; \boldsymbol{\lambda}) \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})\right] \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z} - \nabla_{\boldsymbol{\lambda}} \int q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z} && (1.13)\\
&= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})} \left[\left(\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})\right) \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda})\right] && (1.14)
\end{aligned}$$

where we used the identities $\int q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z} = 1$ and $\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda}) = \frac{\nabla_{\boldsymbol{\lambda}} q(\mathbf{z};\boldsymbol{\lambda})}{q(\mathbf{z};\boldsymbol{\lambda})}$. The expectation in Eq. 1.14 can be approximated using Monte Carlo, by averaging the quantity inside the expectation evaluated at different samples $\mathbf{z}^{(1)} \ldots \mathbf{z}^{(S)}$ from $q(\mathbf{z}; \boldsymbol{\lambda})$,

$$\nabla_{\boldsymbol{\lambda}}\text{ELBO} \approx \frac{1}{S} \sum_{s=1}^{S} \left(\log p(\mathbf{x}, \mathbf{z}^{(s)}) - \log q(\mathbf{z}^{(s)}; \boldsymbol{\lambda})\right) \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}^{(s)}; \boldsymbol{\lambda}) \tag{1.15}$$

The estimator in Eq. 1.15 is an unbiased and consistent estimator of the true score gradient in Eq. 1.14. However, it has high variance, especially when the dimensionality of the latents is large enough. Researchers have developed other gradient approximation methods, for example using Rao-Blackwellization (Casella & Robert, 1996; Ranganath et al., 2014) or control variates (Givens

& Hoeting, 2012; Paisley et al., 2012).

Throughout this dissertation, we will rely on *reparameterization* (Givens & Hoeting, 2012; Kingma & Welling, 2014), a simple method to approximate gradients of Monte Carlo objectives, which we discuss next.

*Reparameterization gradients.* An alternative way to compute gradients of the ELBO in Eq. 1.19 is to introduce variables $\boldsymbol{\epsilon}$ whose distribution $q(\boldsymbol{\epsilon})$ is free from the variational parameters $\boldsymbol{\lambda}$ and such that

$$\mathbf{z} \sim q(\mathbf{z}; \boldsymbol{\lambda}) \iff \boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon}) \text{ and } \mathbf{z} = g(\boldsymbol{\epsilon}; \boldsymbol{\lambda}) \tag{1.16}$$

where $g(\cdot)$ is a function that composes $\boldsymbol{\epsilon}$ and $\boldsymbol{\lambda}$ into samples from the variational distribution. Under this reparameterization procedure, the ELBO takes the form

$$\text{ELBO} = \mathbb{E}_{q(\boldsymbol{\epsilon})} \left[ \log p(\mathbf{x}, g(\boldsymbol{\epsilon}; \boldsymbol{\lambda})) - \log q(g(\boldsymbol{\epsilon}; \boldsymbol{\lambda}); \boldsymbol{\lambda}) \right] \tag{1.17}$$

The gradient of the ELBO is therefore,

$$\nabla_{\boldsymbol{\lambda}} \text{ELBO} = \mathbb{E}_{q(\boldsymbol{\epsilon})} \nabla_{\boldsymbol{\lambda}} \left[ \log p(\mathbf{x}, g(\boldsymbol{\epsilon}; \boldsymbol{\lambda})) - \log q(g(\boldsymbol{\epsilon}; \boldsymbol{\lambda}); \boldsymbol{\lambda}) \right] \tag{1.18}$$

and can be simply approximated using Monte Carlo,

$$\nabla_{\boldsymbol{\lambda}} \text{ELBO} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\lambda}} \left[ \log p(\mathbf{x}, g(\boldsymbol{\epsilon}^{(s)}; \boldsymbol{\lambda})) - \log q(g(\boldsymbol{\epsilon}^{(s)}; \boldsymbol{\lambda}); \boldsymbol{\lambda}) \right] \tag{1.19}$$

where $\boldsymbol{\epsilon}^{(1)} \ldots \boldsymbol{\epsilon}^{(S)} \sim q(\boldsymbol{\epsilon})$. In all our experiments, we set $S = 1$, which has been shown enough for learning (Kingma & Welling, 2014).

## 1.2 Deep Learning

DL is a framework for learning from complex high-dimensional large-scale data. It has been very successful in the domain of supervised learning, where data are labeled. In particular, DL is very successful for vision and language applications, e.g. object detection, image captioning, machine translation, document classification, etc. While PGM specifies the structure underlying the data through a set of latent variables (Koller et al., 2009), DL uses neural networks to capture the structure in data.

### 1.2.1 Neural Networks & Flexibility

Neural networks are a hierarchy of nonlinear deterministic functions (LeCun et al., 2015; Goodfellow et al., 2016). Consider $N$ i.i.d pairs $(\mathbf{x}_i, y_i)$ for $i = 1 \dots N$. A neural network with $L$ layers maps a given input $\mathbf{x}_i$ to its output $y_i$ following a chain of transformations,

$$\mathbf{h}_0 = \mathbf{x}_i \tag{1.20}$$

$$\mathbf{h}_l = f_l(\mathbf{h}_{l-1}; W_l) \tag{1.21}$$

$$\mathbf{h}_L = f_L(\mathbf{h}_{L-1}; W_L) \tag{1.22}$$

$$y_i \sim \text{EF}(\eta_i = g(V^\top \mathbf{h}_L)) \tag{1.23}$$

Here $\mathbf{h}_{1:L}$ are called *hidden states*. The $l^{th}$ hidden state is computed by composing an *activation function* $f_l(\cdot)$ with some transformation of the output of the previous layer that uses the weights $W_l$. Different choices for the activation functions and the transformations yield different neural network architectures, we review some later. The weights represent the model parameters which we aim to learn. $\text{EF}(\eta)$ denotes an exponential family with natural parameter $\eta$. The function $g(\cdot)$ maps the dot product $V^\top \mathbf{h}_L$ to the natural parameter space. For example when $y_i$ is in the reals, $g(\cdot)$ is identity and if $y_i$ is categorical then $g(\cdot) = \text{softmax}(\cdot)$.

Neural networks have been shown to have the ability to represent any function (Hornik et al., 1989). They can therefore capture the complex dependencies in data without any feature engineering. It is this flexibility that has made DL very successful at modeling large-scale complex data.

However neural networks tend to overfit to data. DL approaches often require large datasets to achieve good performance. Researchers have developed several regularization methods for neural networks (Bishop, 1995; Maaten et al., 2013; Srivastava et al., 2014; Gal & Ghahramani, 2016; Dieng et al., 2018c).

Neural networks are fit using stochastic gradient descent with backpropagation (Rumelhart et al., 1986).

### 1.2.2 Example: Recurrent Neural Networks

Consider a sequence of observations, $\mathbf{x}_{1:T} = (\mathbf{x}_1, ..., \mathbf{x}_T)$. A recurrent neural network (RNN) factorizes its joint distribution according to the chain rule of probability,

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{1:t-1}). \tag{1.24}$$

To capture dependencies, the RNN expresses each conditional probability as a function of a low-dimensional recurrent hidden state,

$$\mathbf{h}_t = f_W(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) \text{ and } p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{h}_t).$$

The likelihood $p(\mathbf{x}_t|\mathbf{h}_t)$ can be of any form. We focus on the exponential family

$$p(\mathbf{x}_t|\mathbf{h}_t) = \nu(\mathbf{x}_t) \exp\left\{(V^\top \mathbf{h}_t)^\top \mathbf{x}_t - A(V^\top \mathbf{h}_t)\right\}, \tag{1.25}$$

where $\nu(\cdot)$ is the base measure, $V^\top \mathbf{h}_t$ is the natural parameter—a linear function of the hidden state $\mathbf{h}_t$—and $A(V^\top \mathbf{h}_t)$ is the log-normalizer. The matrix $V$ is called the *prediction* or *output* matrix of

the RNN.

The hidden state $\mathbf{h}_t$ at time $t$ is a parametric function $f_W(\mathbf{h}_{t-1}, \mathbf{x}_{t-1})$ of the previous hidden state $\mathbf{h}_{t-1}$ and the previous observation $\mathbf{x}_{t-1}$; the parameters $W$ are shared across all time steps. The function $f_W$ is the transition function of the RNN, it defines a recurrence relation for the hidden states and renders $\mathbf{h}_t$ a function of all the past observations $\mathbf{x}_{1:t-1}$; these properties match the chain rule decomposition in Eq. 1.24.

The particular form of $f_W$ determines the RNN. Researchers have designed many flavors, including the Elman recurrent neural network (ERNN) (Elman, 1990), the long-short term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and the gated recurrent unit (GRU) (Cho et al., 2014).

**Elman recurrent neural network.** The ERNN is the simplest RNN. In an ERNN, the transition function is

$$f_W(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) = s(W_x^\top \mathbf{x}_{t-1} + W_h^\top \mathbf{h}_{t-1}),$$

where we dropped an intercept term to avoid cluttered notation. Here, $W_h$ is called the *recurrent weight matrix* and $W_x$ is called the *embedding matrix* or *input matrix*. The function $s(\cdot)$ is called an *activation* or *squashing* function, which stabilizes the transition dynamics by bounding the hidden state. Typical choices for the squashing function include the sigmoid and the hyperbolic tangent.

**Long-short term memory.** The LSTM was designed to avoid optimization issues, such as vanishing (or exploding) gradients. Its transition function composes four ERNNs, three with sigmoid

activations and one with a tanh activation:

$$f_t = \sigma(W_{x1}^\top \mathbf{x}_{t-1} + W_{h1}^\top \mathbf{h}_{t-1}) \tag{1.26}$$

$$i_t = \sigma(W_{x2}^\top \mathbf{x}_{t-1} + W_{h2}^\top \mathbf{h}_{t-1}) \tag{1.27}$$

$$o_t = \sigma(W_{x4}^\top \mathbf{x}_{t-1} + W_{h4}^\top \mathbf{h}_{t-1}) \tag{1.28}$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tanh(W_{x3}^\top \mathbf{x}_{t-1} + W_{h3}^\top \mathbf{h}_{t-1}) \tag{1.29}$$

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t). \tag{1.30}$$

Here $f_t$, $i_t$, and $o_t$ are called *gates*. The LSTM state is the pair $(\mathbf{c}_t, \mathbf{h}_t)$. The state $\mathbf{c}_t$ is the *memory cell*; it is designed to capture long-term dependencies (Hochreiter & Schmidhuber, 1997). The gate $f_t$ is the *forget gate*; it determines which part of memory to discard. The gate $i_t$ controls the amount of new information to add to the memory. Finally, the gate $o_t$ determines how the output state $\mathbf{h}_t$ depends on the memory cell.

**Gated recurrent unit.** GRUS provide a simpler way to parameterize a RNN than the LSTM (Cho et al., 2014). The hidden state $\mathbf{h}_t$ of a GRU is computed as

$$u_t = \sigma(W_{x1}^\top \mathbf{x}_{t-1} + W_{h1}^\top \mathbf{h}_{t-1}) \tag{1.31}$$

$$r_t = \sigma(W_{x2}^\top \mathbf{x}_{t-1} + W_{h2}^\top \mathbf{h}_{t-1}) \tag{1.32}$$

$$\mathbf{h}_t = u_t \odot \mathbf{h}_{t-1} + (1 - u_t) \odot \tanh\left(W_{x3}^\top \mathbf{x}_{t-1} + W_{h3}^\top (r_t \odot \mathbf{h}_{t-1})\right). \tag{1.33}$$

Here $u_t$ is called an *update* gate, it decides whether to change the previous configuration of the hidden state or not. The variable $r_t$ is called a *reset gate* it indicates which coordinates of the previous hidden state are to be updated.

### 1.2.3 Example: Auto-Encoders

Auto-encoding is a successful DL technique for dimensionality reduction (Hinton & Salakhutdinov, 2006). The idea is to learn to reconstruct data. Consider a dataset of $N$ i.i.d observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$. An autoencoder (AE) minimizes reconstruction error,

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \|\mathbf{x}_i - f_\theta(g_\phi(\mathbf{x}_i))\|_2^2. \tag{1.34}$$

Here $g_\phi(\cdot)$ is called an *encoder*, it takes a data point $\mathbf{x}_i$ as input and outputs $\mathbf{h}_i = g_\phi(\mathbf{x}_i)$, which is a low-dimensional representation of $\mathbf{x}_i$ called a *code*. The function $f_\theta(\cdot)$ is called a *decoder*, it maps the code $\mathbf{h}_i$ to the observation space. Its output is $\tilde{\mathbf{x}}_i$ which is optimized to be close to $\mathbf{x}_i$. AEs have been shown to learn better low-dimensional representations of data than principal component analysis (PCA) (Hinton & Salakhutdinov, 2006). They have also been useful for other applications, e.g. image denoising (Vincent et al., 2008).

### 1.2.4 Example: Word Embeddings

Word embeddings provide models of language that use vector representations of words (Rumelhart & Abrahamson, 1973; Bengio et al., 2003). The word representations are fitted to relate to meaning, in that words with similar meanings will have representations that are close. (In embeddings, the "meaning" of a word comes from the contexts in which it is used (Harris, 1954).)

We focus on the continuous bag-of-words (CBOW) variant of word embeddings (Mikolov et al., 2013b). In CBOW, the likelihood of each word $w_{dn}$ is

$$w_{dn} \sim \text{softmax}(\rho^\top \alpha_{dn}). \tag{1.35}$$

The embedding matrix $\rho$ is a $L \times V$ matrix whose columns contain the embedding representations of the vocabulary, $\rho_v \in \mathbb{R}^L$. The vector $\alpha_{dn}$ is the *context embedding*. The context embedding is

the sum of the context embedding vectors ($\alpha_v$ for each word $v$) of the words surrounding $w_{dn}$.

## 1.3 Combining Neural Networks and Latent Variables

PGM and DL both aim to learn from data. While PGM uses latent variables to express the structure underlying data in an interpretable way, DL uses neural networks to express the structure underlying data in a flexible way. These two approaches of learning from data are complementary. Researchers have developed methods that combine neural networks and latent variables (Kingma & Welling, 2013; Rezende et al., 2014; Johnson et al., 2016; Gao et al., 2016; Krishnan et al., 2017).

### 1.3.1 Probabilistic Conditioning with Neural Networks

Probabilistic conditioning with neural networks is a way to combine neural networks and latent variables. There are two ways to do this:

- Use the output of a neural network that takes the latent variables as input to define the parameters of the conditional distribution of the data given the latent variables.
- Use the output of a neural network that takes data as input to define the parameters of the posterior distribution of the latent variables given the data.

More concretely, consider our running PGM example, the EF-PCA. It posits a shared global latent structure $\boldsymbol{\beta}$ and a per-observation local latent structure $\mathbf{z}_{1:N}$. A data point $\mathbf{x}$ is drawn by conditioning on both $\boldsymbol{\beta}$ and $\mathbf{z}$. In EF-PCA, the global and local structure interact linearly in the likelihood $p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\beta})$. (See Section 1.1.3.) Allowing non-linear interactions between global and local structure will make the model more flexible. This can be achieved using neural networks, the parameters of which will be shared across the observations to model the global structure. The conditional distribution of $\mathbf{x}$ given $\mathbf{z}$ is then

$$p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid f_{\boldsymbol{\beta}}(\mathbf{z})) \cdot p(\mathbf{z}) \tag{1.36}$$

where $f_{\beta}(\cdot)$ is the neural network that defines the likelihood.

On the other hand, the local latent variables $\mathbf{z}_{1:N}$ can be seen as low-dimensional representations of the data. Each observation has its own representation in the latent space. We saw in Section 1.2 that AES are good at finding low-dimensional representations of data. We can use auto-encoding to define the posterior distributions of the local latent variables,

$$q_{\phi}(\mathbf{z} \mid \mathbf{x}) = q_{\phi}(\mathbf{z} \mid g_{\phi}(\mathbf{x})) \tag{1.37}$$

where $g_{\phi}(\cdot)$ is the neural network that parameterizes the posterior, the encoder in an AE. Note $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ is not the true posterior distribution of the latent variables, but we can use it within the framework of VI to learn approximations of the true posterior.

### 1.3.2 Variational Auto-Encoders

Variational autoencoders (VAES) tie model design and posterior inference within one framework. They use Eq. 1.36 as a model for data and Eq. 1.37 as an approximate posterior over latent variables. More specifically, the likelihood is an exponential family whose natural parameter $\eta(\mathbf{z}; \boldsymbol{\beta})$ is computed as follows:

1. $\mathbf{h}^{(1)} = f_{\beta_0}(\mathbf{z})$
2. $\mathbf{h}^{(l+1)} = f_{\beta l}\left(\mathbf{h}^{(l)}\right) \quad l = 1 \ldots L-1$
3. $\eta(\mathbf{z}; \theta) = f_{\beta_L}\left(\mathbf{h}^{(L)}\right)$.

The parameter $\boldsymbol{\beta}$ is the collection $\{\beta_0, \ldots, \beta_L\}$. The output $\mathbf{h}^{(l+1)}$ of the $(l+1)^{th}$ layer is computed by composing the output $\mathbf{h}^{(1)}$ of the previous layer with layer-specific parameters $\beta l$.

VAES use stochastic gradient ascent to learn both sets of parameters $\boldsymbol{\beta}$ and $\phi$. The objective is the

ELBO,

$$\text{ELBO}(\boldsymbol{\beta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})} \left[ \log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \right]. \tag{1.38}$$

For a given setting of $\boldsymbol{\phi}$, maximizing the ELBO with respect to $\boldsymbol{\beta}$ corresponds to maximizing the likelihood of the observations. For a given setting of the model parameters $\boldsymbol{\beta}$, maximizing the ELBO with respect to $\boldsymbol{\phi}$ can be interpreted in two different ways.

**KL minimization perspective.** We can write the ELBO as

$$\text{ELBO}(\boldsymbol{\beta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})} \left[ \log p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x}) + \log p_{\boldsymbol{\beta}}(\mathbf{x}) - \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \right]. \tag{1.39}$$

Since $\log p_{\boldsymbol{\beta}}(\mathbf{x})$ does not depend on $\boldsymbol{\phi}$, maximizing the ELBO with respect to $\boldsymbol{\phi}$ is equivalent to minimizing the KL between $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and the true posterior $p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})$. Indeed,

$$\text{ELBO}(\boldsymbol{\beta}, \boldsymbol{\phi}) = -\text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) || p_{\boldsymbol{\beta}}(\mathbf{z} \mid \mathbf{x})) + cst. \tag{1.40}$$

**Regularized autoencoder perspective.** Maximizing the ELBO with respect to $\boldsymbol{\phi}$ can also be seen as regularizing an AE. Rewrite the ELBO as follows

$$\text{ELBO}(\boldsymbol{\beta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})} \left[ \log p_{\boldsymbol{\beta}}(\mathbf{x} \mid \mathbf{z}) \right] - \text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) || p(\mathbf{z})). \tag{1.41}$$

Assume without loss of generality that the likelihood is Gaussian with identity variance and that $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is also a Gaussian with identity variance. Assume we draw one sample $\mathbf{z}_{\boldsymbol{\phi}}(\mathbf{x}) = g_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\epsilon}$ from $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Assume the prior is standard Gaussian. The ELBO is

$$\text{ELBO}(\boldsymbol{\beta}, \boldsymbol{\phi}) = \|\mathbf{x} - f_{\boldsymbol{\beta}}(\mathbf{z}_{\boldsymbol{\phi}}(\mathbf{x}))\|_2^2 - \frac{1}{2} \|g_{\boldsymbol{\phi}}(\mathbf{x})\|_2^2. \tag{1.42}$$

When maximizing the ELBO with respect to $\boldsymbol{\phi}$, the first term is the objective of an AE, it forces

to learn settings of $\phi$ that are able to reconstruct the data well. The second term regularizes the parameters $\phi$ such that the output of the encoder have bounded $L_2$ norm.

In fact there is a second source of regularization in this particular case; noise $\boldsymbol{\epsilon}$ is first added to the code $g_\phi(\mathbf{x})$ to get the *latent* code $\mathbf{z}_\phi(\mathbf{x})$, which is then used as input to the decoder $f_{\boldsymbol{\beta}}(\cdot)$. This added noise trades off some reconstruction error with the ability to simulate new data from the fitted decoder.

**Amortized variational inference and mean field variational inference.** By using neural networks to define the approximate posterior over the latent variables, VAEs are doing amortized variational inference (AVI). The term comes from the fact that computing the approximate posterior boils down to passing data through a shared neural network, which *amortizes* the cost of inference for models with local latent variables when dealing with large datasets.

How does AVI relate to mean field VI? Consider our canonical EF-PCA example. Mean field VI uses the factorization

$$q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}_\beta) \cdot \prod_{i=1}^{N} q(\mathbf{z}_i; \boldsymbol{\lambda}_i) \tag{1.43}$$

Mean field assumes all latent variables are independent, both local and global. We can relax this a bit and assume the local latent variables $\mathbf{z}_{1:N}$ are conditionally independent given the global latent variables,

$$q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}_\beta) \cdot \prod_{i=1}^{N} q(\mathbf{z}_i \mid \boldsymbol{\beta}; \boldsymbol{\lambda}_i) \tag{1.44}$$

We let each factor explicitly condition on data,

$$q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}_\beta) \cdot \prod_{i=1}^{N} q(\mathbf{z}_i \mid \mathbf{x}_i, \boldsymbol{\beta}; \boldsymbol{\lambda}_i) \tag{1.45}$$

A way to get to AVI is to assume that the global structure $\boldsymbol{\beta}$ represents a posteriori a neural network

with parameters $\boldsymbol{\lambda}_\beta$ and let each latent $\mathbf{z}_i$ have its own distribution only through $\mathbf{x}_i$. Then we end up with

$$q(\boldsymbol{\beta}, \mathbf{z}_{1:N}; \boldsymbol{\lambda}) = \prod_{i=1}^{N} q(\mathbf{z}_i \mid \mathbf{x}_i, \boldsymbol{\lambda}_\beta) \tag{1.46}$$

where the conditioning on $\mathbf{x}_i$ and $\boldsymbol{\lambda}_\beta$ corresponds to passing $\mathbf{x}_i$ through the neural network $\boldsymbol{\lambda}_\beta$.

### 1.3.3 Challenges: Latent Variable Collapse

The ELBO in Eq. 1.38 is intractable because of the expectations. VAEs leverage BBVI (Paisley et al., 2012; Ranganath et al., 2014) and approximate the ELBO with Monte Carlo samples from the variational distribution. To reduce variance of the gradients of the ELBO, VAEs use reparameterization (Kingma & Welling, 2013; Rezende et al., 2014). This procedure often empirically leads to a degenerate solution where

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p(\mathbf{z}).$$

The variational "posterior" does not depend on the data; this is referred to as *latent variable collapse*. When the approximate posterior is close to the prior, posterior estimates of the latent variable $\mathbf{z}$ do not represent faithful summaries of their data $\mathbf{x}$—the VAE has not learned good representations. This issue is discussed in several papers (Bowman et al., 2015; Sønderby et al., 2016b; Kingma et al., 2016; Chen et al., 2016; Zhao et al., 2017c; Yeung et al., 2017). In Chapter 2 we provide a solution to this problem.

# Chapter 2: Deep Probabilistic Graphical Modeling

The previous chapter laid out the foundations for *deep probabilistic graphical modeling* (DPGM). We reviewed latent-variable graphical models and their inference. These models have an interpretable probabilistic structure and can be fit using variational inference (VI). However graphical models tend to lack flexibility, which hinders their use when it comes to modeling high-dimensional complex data and/or performing tasks that require flexibility (e.g. in vision and language applications.)

We reviewed deep learning (DL), a paradigm that offers flexibility both in terms of model specification and model fitting by leveraging neural networks and backpropagation. Although flexible, DL does not offer the same interpretability as probabilistic graphical modeling (PGM).

Finally, we described ways to combine neural networks and latent variables and the *latent variable collapse* issue that might arise from it leading to a non-interpretable latent structure.

In this chapter, we develop DPGM, a set of methodologies that leverage DL for PGM. DPGM benefits from the interpretability of PGM and the flexibility of DL. We first discuss three desiderata for DPGM before describing several instances of DPGM. One instance extends exponential family principal component analysis (EF-PCA) using deep neural networks while preserving the interpretability of the latent factors. Another instance corresponds to a model class for sequential data that allows to account for long-range dependencies. Finally, we show how DPGM solves several problems of probabilistic topic models.

## 2.1 Desiderata

The goal of DPGM is to make PGM more flexible. This leads to three desiderata for DPGM, which we discuss in more detail.

1. *Generalization*. Data are finite. We require systems built using DPGM to generalize beyond the observed data. Here, the term "generalization" encapsulates two things: (1) the capacity to assign high probability to unobserved data arising from the same distribution as the training data[1] and (2) the ability to yield good simulations of new data. The latter pertains to the diversity and visual quality of data generated from the fitted model. We will measure generalization using held-out predictive log-likelihood or measures of simulation quality.

2. *Interpretability*. PGM uncovers the hidden structure underlying data through a set of latent variables. These latent variables capture meaning and are interpretable. We require the same interpretability for DPGM. However, flexibility often comes in the way of interpretability. The *latent variable collapse* problem described in Chapter 1 is one manifestation of this. The DPGM instances we describe in Section 2.2, Section 2.3, and Section 2.4 will offer both flexibility and interpretability. We will measure interpretability of the learned latent variables using proxies of mutual information or performance on a downstream classification task.

3. *Scalability*. The data we deal with are large-scale and high-dimensional. We require DPGM methods to scale both in terms of the number of observations and the dimensionality of each observation.

## 2.2 From Exponential Family PCA to Deep Generative Skip Models

We introduce deep generative skip models (DGSMS), a class of models that extends EF-PCA using neural networks. DGSMS are efficiently fit using amortized variational inference (AVI) (Gershman & Goodman, 2014; Kingma & Welling, 2013; Rezende et al., 2014). When evaluated on image and

[1] this is the usual meaning of the word "generalization" in Machine Learning.

text data, they achieve higher predictive performance and learn more interpretable latent factors than several baselines.

### 2.2.1 Model Class

Assume observed $N$ i.i.d data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where $\mathbf{x}_i \in \mathbb{R}^D$. Consider the data generative process of EF-PCA:

1. Draw global latents $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$
2. For each data point $i = 1 \ldots N$:
   (a) Draw local latent variable $\mathbf{z}_i \sim p(\mathbf{z})$
   (b) Draw data point $\mathbf{x}_i \sim \text{EF}(\eta_i = f(\boldsymbol{\beta}^\top \mathbf{z}_i))$

As discussed in Chapter 1, the global latent variables $\boldsymbol{\beta}$ capture features shared across all the observations. We can infer them using variational inference. In this section, we set $\boldsymbol{\beta}$ to be deterministic parameters of a shared deep neural network. DGSMs define the following generative process for data,

1. For each data point $i = 1 \ldots N$:
   (a) Draw local latent variable $\mathbf{z}_i \sim p(\mathbf{z})$
   (b) Draw data point $\mathbf{x}_i \sim \text{EF}(\eta_i = f_{\boldsymbol{\beta}}(\mathbf{z}_i))$.

Here, the natural parameter is the output of a neural network $f_{\boldsymbol{\beta}}(\cdot)$ that takes $\mathbf{z}_i$ as input. It is computed through the following chain:

1. $\mathbf{h}_i^{(1)} = f_{\theta_0}(\mathbf{z}_i)$
2. $\mathbf{h}_i^{(l+1)} = g_{W_l}\left(f_{\theta_l}\left(\mathbf{h}_i^{(l)}\right), \mathbf{z}_i\right)$ for $l = 1 \ldots L-1$
3. $\eta_i = g_{W_L}\left(f_{\theta_L}\left(\mathbf{h}_i^{(L)}\right), \mathbf{z}_i\right)$.

Here $\boldsymbol{\beta} = (W_1, \ldots, W_L, \theta_0, \ldots, \theta_L)$ The functions $g_{W_1}(\cdot), \ldots, g_{W_L}(\cdot)$ and $f_{\theta_0}(\cdot), \ldots, f_{\theta_L}(\cdot)$ define the neural network $f_{\boldsymbol{\beta}}(\cdot)$. Their choices lead to different architectures. At a given layer $l$, the hidden state $\mathbf{h}_i^{(l)}$ of the neural network $f_{\boldsymbol{\beta}}(\cdot)$ is a function of both the latent variable $\mathbf{z}_i$ and the hidden state from the previous layer $\mathbf{h}_i^{(l-1)}$. The dependence on $\mathbf{z}_i$, via $g_{W_1}(\cdot), \ldots, g_{W_L}(\cdot)$, is called a *skip connection*. Skip connections are widely used in DL, for example, in designing residual, highway, and attention networks (Fukushima, 1988; He et al., 2016b; Srivastava et al., 2015; Bahdanau et al., 2014). Here we use them to define DPGMS to enforce a stronger dependence between the latent variables and the observations.

DGSMS are amenable to any type of skip functions; in this section we consider

$$g_{W_l}\left(f_{\theta_l}(\mathbf{h}_i^{(l)}), \mathbf{z}_i\right) = \sigma_l\left(W_l^{(h)} f_{\theta_l}(\mathbf{h}_i^{(l)}) + W_l^{(z)} \mathbf{z}_i\right)$$

where $\sigma_l$ is a typical nonlinear function such as sigmoid or ReLU. We set $\sigma_L(\cdot)$, the activation at the last layer, to identity. The weights $W_l^{(h)} \neq \mathbf{0}$ and $W_l^{(z)} \neq \mathbf{0}$ are parameters of the model. Similarly to other uses of skip connections (Fukushima, 1988; He et al., 2016b; Srivastava et al., 2015; Bahdanau et al., 2014) we do not need to explicitly enforce the constraints $W_l^{(h)} \neq \mathbf{0}$ and $W_l^{(z)} \neq \mathbf{0}$ in practice.

DGSMS are amenable to any neural network architecture. Defining $f_{\theta_0}(\cdot), \ldots, f_{\theta_L}(\cdot)$ corresponds to specifying a neural network architecture. In our empirical study we explore convolutional neural networks (CNNS) for image applications and long-short term memorys (LSTMS) for text applications.

### 2.2.2 Amortized Variational Inference

DGSMS are fit using AVI. For that we define a variational distribution over the latent variables,

$$q_\phi(\mathbf{z}_{1:N}|\mathbf{x}_{1:N}) = \prod_{i=1}^{N} q_\phi(\mathbf{z}_i|\mathbf{x}_i) \tag{2.1}$$

We set each factor $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$ as a Gaussian whose mean and Covariance are the given by the output of a neural network that takes $\mathbf{x}_i$ as input,

$$q_\phi(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_i;\phi), \Sigma_i(\mathbf{x}_i;\phi)) \tag{2.2}$$

Here $\boldsymbol{\mu}_i(\mathbf{x}_i;\phi_\mu)$ and $\Sigma_i(\mathbf{x}_i;\phi_\Sigma)$ are the neural networks for the mean and the covariance respectively and $\phi = (\phi_\mu, \phi_\Sigma)$. In practice we use one shared neural network whose output is mapped to one of two sets of weights to get the mean or the covariance. We use $\text{softplus}(a) = \log(1+\exp(a))$ as the final activation function when computing the covariance.

Note the variational distribution can also be parameterized using the same approach we used to define the model. More concretely, when computing the mean and the covariance of the variational distributions, we can add skip connections from an input $\mathbf{x}_i$ to each layer of the inference network.

We now can form the evidence lower bound (ELBO),

$$\text{ELBO} = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} \left[\log p_\beta(\mathbf{x}_i|\mathbf{z}_i) - \text{KL}(q_\phi(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i))\right] \tag{2.3}$$

The ELBO is intractable but we can estimate it using Monte Carlo with the reparameterization trick (Kingma & Welling, 2014). We can then optimize the ELBO with respect to both the model parameters $\beta$ and the variational parameters $\phi$.

### 2.2.3 Connections & Related Work

DGSM fit using AVI are related to VAEs. The difference between the approach of Kingma & Welling (2013) and the approach described above is the use of skip connections, when defining the model and/or the inference network. We verify empirically that these skip connections lead to more interpretable latent factors than the VAE. To make the connection more apparent, we call AVI in the context of DGSMs, SKIP-VAEs. Figure 2.1 highlights the methodological differences between VAEs

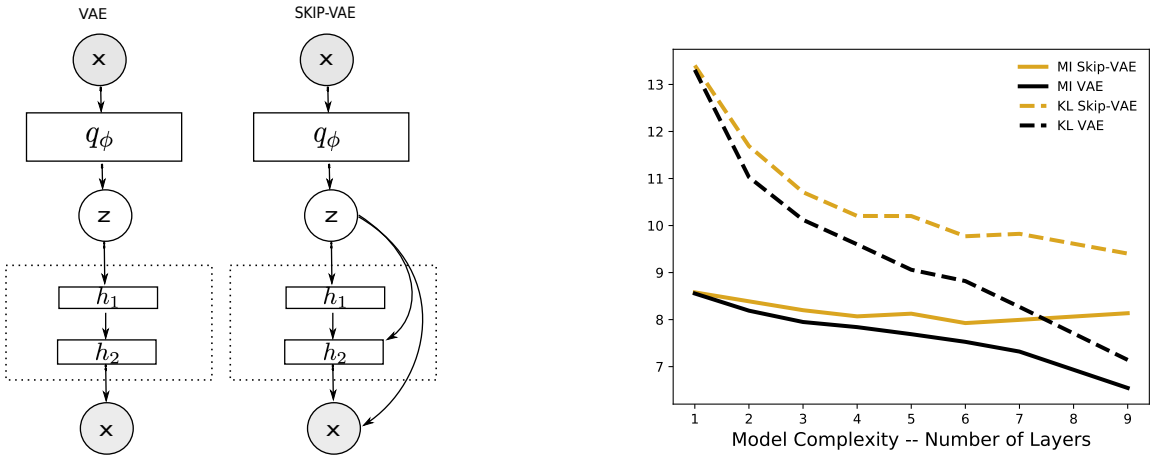



*Figure 2.1.* **Left:** The variational autoencoder (VAE) and Skip Variational Autoencoder (SKIP-VAE) with a two-layer generative model. The function $q_\phi$ denotes the variational neural network (here identical for VAE and SKIP-VAE). The difference is in the generative model class: the SKIP-VAE's generative model enforces residual paths to the latents at each layer. **Right:** The mutual information induced by the variational distribution and KL from the variational distribution to the prior for the VAE and the SKIP-VAE on MNIST as we vary the number of layers $L$. The SKIP-VAE leads to both higher KL and higher mutual information.

and SKIP-VAES and shows the benefit of SKIP-VAES over VAES in learning more interpretable latent variables. This figure is a visualization of two interpretability metrics as a function of the depth of the neural network used to define the generative model for data. The first interpretability metric is the mutual information between the data and the latent variables (MI) whereas the second metric is the Kullback-Leibler (KL) between the variational distribution and the prior. The higher these metrics the better; higher MI and KL signal stronger correlation between the data and the latent variables.

There have been many proposals for learning more interpretable latent variables with the VAE. These approaches tackle the latent variable collapse discussed in Chapter 1 from different angles. One approach is to handicap the training of the generative model (Bowman et al., 2015) or weaken its capacity (Gulrajani et al., 2016), effectively encouraging better representations by limiting the generative model. Another approach replaces the simple spherical Gaussian prior with more sophisticated priors. For example van den Oord et al. (2017) and Tomczak & Welling (2017) propose parametric priors, which are learned along with the generative model. Still another approach

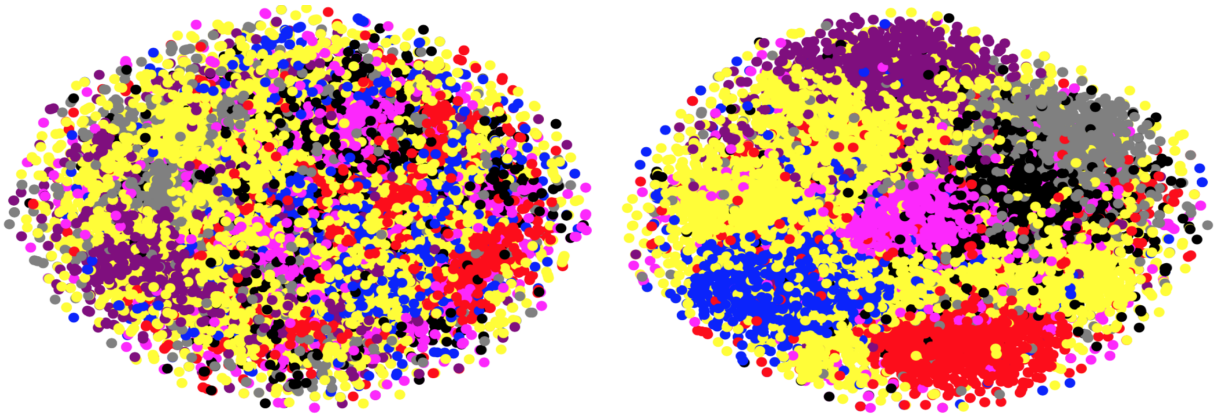


*Figure 2.2.* Clustering of the latent variables learned by fitting a VAE (left) and a SKIP-VAE (right) on MNIST and applying T-SNE on the test set. The model is a 9-layer PixelCNN and the variational neural network is a 3-layer ResNet. The colors represent digit labels. The SKIP-VAE clusters the latent variables better than the VAE; it discovers 7 digit classes. The remaining 3 classes are covered by the other classes. The latent variables learned by the VAE are not meaningful as they are spread out. The SKIP-VAE learns more useful latent representations.

uses richer variational distributions (Rezende & Mohamed, 2015). In another thread of research, Makhzani et al. (2015); Mescheder et al. (2017) replace the KL regularization term in the VAE objective with adversarial regularizers and Higgins et al. (2017) dampen the effect of the KL regularization term with Lagrange multipliers. Finally, one can appeal to new inference algorithms. For example Hoffman (2017) uses Markov chain Monte Carlo (MCMC) instead of variational inference and Kim et al. (2018) uses stochastic variational inference, initialized with the variational neural network parameters, to iteratively refine the variational distribution. A very recent approach against posterior collapse relies on ideas from directional statistics. More specifically it consists in using the Von Mises-Fisher distribution for both the prior and the variational posterior and fixing the dispersion parameter of the Von Mises-Fisher distribution to make the KL term in the ELBO constant (Guu et al., 2017; Xu & Durrett, 2018). However this practice might result in less expressive approximate posteriors.

### 2.2.4 Empirical Study

We first set some definitions

*Table 2.1.* Performance of SKIP-VAE vs VAE on MNIST as the dimensionality of the latent variable increases. SKIP-VAE outperforms VAE on all collapse metrics while achieving a similar log likelihood—as measured by ELBO.

| | ELBO | | KL | | MI | | AU | |
|---|---|---|---|---|---|---|---|---|
| Dim | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE |
| 2 | **-84.27** | −84.30 | 3.13 | **3.54** | 3.09 | **3.46** | 2 | 2 |
| 10 | −83.01 | **-82.87** | 8.29 | **9.41** | 7.35 | **7.81** | 9 | **10** |
| 20 | −83.06 | **-82.55** | 7.14 | **9.33** | 6.55 | **7.80** | 8 | **13** |
| 50 | −83.31 | **-82.58** | 6.22 | **8.67** | 5.81 | **7.49** | 8 | **12** |
| 100 | −83.41 | **-82.52** | 5.82 | **8.45** | 5.53 | **7.38** | 5 | **9** |

**Definition 2** *For any data* $\mathbf{x}$ *and variational posterior* $q_\phi(\mathbf{z}|\mathbf{x})$*, the variational joint* $q_\phi(\mathbf{x}, \mathbf{z})$ *is the joint distribution of* $\mathbf{x}$ *and* $\mathbf{z}$ *induced by* $q_\phi(\mathbf{z} \mid \mathbf{x})$*. It induces a marginal* $q_\phi(\mathbf{z})$ *called the aggregated posterior (Makhzani et al., 2015; Mescheder et al., 2017)*

$$q_\phi(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}) \cdot q_\phi(\mathbf{z}|\mathbf{x}) \quad \textit{and} \quad q_\phi(\mathbf{z}) = E_{p(\mathbf{x})} q_\phi(\mathbf{z}|\mathbf{x}).$$

*The mutual information* $\mathcal{I}_q(\mathbf{x}, \mathbf{z})$ *induced by the variational joint is*

$$\mathcal{I}_q(\mathbf{x}, \mathbf{z}) = E_{p(\mathbf{x})} E_{q_\phi(\mathbf{z} \mid \mathbf{x})} \log q_\phi(\mathbf{z} \mid \mathbf{x}) - E_{q_\phi(\mathbf{z})} \log q_\phi(\mathbf{z}).$$

We next assess the performance of SKIP-VAEs on learning latent representations of data by applying it to both a standard VAE (Kingma & Welling, 2013; Rezende et al., 2014) and to the recently introduced semi-amortized variational autoencoder (SA-VAE) (Kim et al., 2018). We use standard benchmark datasets for images and text: MNIST, Omniglot, and the Yahoo corpus. Text datasets have been shown to be particularly sensitive to latent variable collapse (Bowman et al., 2015).

The prior for all experiments is a spherical Gaussian, and the variational posterior is a diagonal Gaussian. Experiments compare SKIP-VAE to baselines when varying the dimensionality of the latent variable and the complexity of the generative model.

**Evaluation metrics** Our evaluation metrics assess both the performance—as given by some mea-

*Table 2.2.* Performance of SKIP-VAE vs VAE on MNIST (Top) and Omniglot (Bottom) as the complexity of the decoder increases. Skip-VAE outperforms VAE on all collapse metrics while achieving a similar log likelihood—as measured by ELBO. In particular, this advantage widens as the number of layers increases. The number of latent dimensions is 20—they are all active under SKIP-VAE on Omniglot.

| | ELBO | | KL | | MI | | AU | |
|---|---|---|---|---|---|---|---|---|
| Layers | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE |
| 1 | −89.64 | **-89.22** | 13.31 | **13.40** | 8.56 | 8.56 | 20 | 20 |
| 3 | −84.38 | **-84.03** | 10.12 | **10.71** | 7.95 | **8.20** | 16 | 16 |
| 6 | −83.19 | **-82.81** | 8.82 | **9.77** | 7.53 | **7.93** | 11 | **13** |
| 9 | −83.06 | **-82.55** | 7.14 | **9.34** | 6.55 | **7.80** | 8 | **13** |
| 1 | −97.69 | **-97.66** | **8.42** | 8.37 | **7.09** | 7.08 | 20 | 20 |
| 3 | −93.95 | **-93.75** | 6.43 | **6.58** | 5.88 | **5.97** | 20 | 20 |
| 6 | −93.23 | **-92.94** | 5.24 | **5.78** | 4.94 | **5.43** | 20 | 20 |
| 9 | −92.79 | **-92.61** | 4.41 | **6.12** | 4.24 | **5.65** | 11 | **20** |

*Table 2.3.* VAE and SkipVAE on MNIST using 50 latent dimensions. The encoder is a 2-layer MLP with 512 units in each layer. The decoder is also an MLP. The results below correspond to different number of layers for the decoder.

| | ELBO | | KL | | MI | | AU | |
|---|---|---|---|---|---|---|---|---|
| Layers | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE | VAE | SKIP-VAE |
| 2 | −94.88 | **-94.80** | 24.23 | **26.35** | **9.21** | 9.20 | 17 | **24** |
| 3 | −95.38 | **-94.17** | 21.87 | **26.15** | 9.20 | **9.21** | 13 | **21** |
| 4 | −97.09 | **-93.79** | 20.95 | **25.63** | **9.21** | **9.21** | 11 | **21** |

sure of log-likelihood—as well as latent variable collapse. Performance is measured using standard metrics: for image datasets we report the ELBO as a measure of log-likelihood, for text we report both the ELBO and perplexity estimated using importance sampling.

Quantitatively assessing latent variable collapse is more difficult. We employ three metrics KL, MI, and AU. The first metric is the KL regularization term of the ELBO as written in Eq. 2.3. The second measure of latent variable collapse is the mutual information induced by the variational joint $\mathcal{I}_q(\mathbf{x}, \mathbf{z})$. We follow Hoffman & Johnson (2016) and approximate this mutual information using Monte Carlo estimates of the two KL terms. In particular $\mathrm{KL}(q(\mathbf{z}; \phi) \,\|\, p(\mathbf{z}))$ is approximated

*Table 2.4.* SKIP-VAE and SKIP-SA-VAE perform better than their counterparts (VAE, SA-VAE) on the Yahoo corpus under all latent variable collapse metrics while achieving similar log-likelihoods. In particular, all latent dimensions are active when using SKIP-SA-VAE. Perplexity (PPL) for the variational models is estimated by calculating the log marginal likelihood with 200 samples from $q(\mathbf{z} \mid \mathbf{x}; \phi)$.

| Model | Dim | PPL | ELBO | KL | MI | AU |
|---|---|---|---|---|---|---|
| LANGUAGE MODEL | - | 61.60 | - | - | - | - |
| VAE | 32 | 62.38 | −330.1 | 0.005 | 0.002 | 0 |
| SKIP-VAE | 32 | 61.71 | −330.5 | **0.34** | **0.31** | **1** |
| SA-VAE | 32 | 59.85 | −327.5 | 5.47 | 4.98 | 14 |
| SKIP-SA-VAE | 32 | 60.87 | −330.3 | **15.05** | **7.47** | **32** |
| SA-VAE | 64 | 60.20 | −327.3 | 3.09 | 2.95 | 10 |
| SKIP-SA-VAE | 64 | 60.55 | −330.8 | **22.54** | **9.15** | **64** |

as

$$\begin{aligned}\mathrm{KL}(q(\mathbf{z};\phi) \,\|\, p(\mathbf{z})) &= \mathbb{E}_{q(\mathbf{z};\phi)}\left[\log q(\mathbf{z};\phi) - \log p(\mathbf{z})\right] \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \log q(\mathbf{z}^{(s)};\phi) - \log p(\mathbf{z}^{(s)})\end{aligned}$$

where each aggregated posterior $q(\mathbf{z}^{(s)};\phi)$ is also approximated by Monte Carlo.

The third measure of latent variable collapse is the number of "active" dimensions of the latent variable $\mathbf{z}$. This is defined in Burda et al. (2015a) as

$$\mathrm{AU} = \sum_{d=1}^{D} \mathbb{1}\left\{\mathrm{Cov}_{p(\mathbf{x})}\left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[z_d]\right) \geq \epsilon\right\},$$

where $z_d$ is the $d^{th}$ dimension of $\mathbf{z}$ and $\epsilon$ is a threshold. ($\mathbb{1}\{\cdot\}$ is an indicator giving 1 when its argument is true and 0 otherwise.) We follow Burda et al. (2015a) and use a threshold of $\epsilon = 0.01$. We observe the same phenomenon: the histogram of the number of active dimensions of $\mathbf{z}$ is bi-modal, which means that it is not highly sensitive to the chosen threshold.

**Images** We studied MNIST and Omniglot. We use a 3-layer ResNet (He et al., 2016a) (with $3 \times 3$ filters and 64 feature maps in each layer) as the variational neural network and a 9-layer Gated

PixelCNN (van den Oord et al., 2016) (with $3 \times 3$ filters and 32 feature maps) as the likelihood. For the baseline approach without skip connections we apply a linear map to the sample from the variational posterior (to project out to the image spatial resolution), concatenate this with the original image, and feed this to the PixelCNN. This set up reflects the set up of current state-of-the-art settings for modeling images with VAEs (Gulrajani et al., 2016; Chen et al., 2016). For the generative skip model, we apply a linear map to the sample and concatenate it with the output from each layer of the PixelCNN (before feeding it to the next layer). This results in more parameters for the SKIP-VAE model but we will see shortly that the baseline VAE's performance on the collapse metrics worsens more quickly than the SKIP-VAE as the capacity of the model increases.

Table 2.1 shows the results on MNIST when varying the size of the latent dimension. In all scenarios, the generative skip model yields higher KL between the variational posterior and the prior, higher mutual information, and uses more latent dimensions (as measured by AU).

Table 2.2 varies the generative model's complexity by increasing its depth. We used 20-dimensional latent variables. With the VAE, as the generative model becomes more expressive the model becomes less reliant on $\mathbf{z}$ as evidence by the poor performance on the collapse metrics. The generative skip model mitigates this issue and performs better on all latent-variable-collapse metrics. Note the ELBO is similar for both models. These results indicate that the family of generative skip models has a strong inductive bias to share more mutual information between the observation and the latent variable. Similar results are observed when using weaker models. For example in Table 2.3 we used feed forward neural networks (MLPS) for both the variational neural network and the generative model. We set the dimensionality of the latent variables to 50. Even with this weaker setting the SKIP-VAE leads to less collapse than the VAE.

**Quality of the learned latent variables.** To measure the quality of the learned latent variables for both the VAE and the SKIP-VAE we ran two sets of analyses: one quantitative and one qualitative. Qualitatively we cluster the latent variables using the learned variational neural network and the test set. Figure 2.2 illustrates this. It shows a clear clustering of the MNIST digits with the latent space

learned by the generative skip model. This is not the case for the latent variables learned by the VAE which are more spread out. Note we did not fit a VAE and a SKIP-VAE with 2-dimensional latents for the visualization as this is not a realistic setting in practice. Instead we fit the VAE and the SKIP-VAE on 50-dimensional latents—as is usual in state-of-the-art image modeling with VAEs—and used t-SNE to project the learned latents on a two-dimensional space.

Quantitatively we performed a classification experiment on MNIST using the latent variables learned by the variational neural networks of VAE and SKIP-VAE as features. This experiment uses 50 latent dimensions, a 9-layer PixelCNN as the generative model, a 3-layer ResNet as the variational neural network, and a simple 2-layer MLP over the posterior means as the classifier. The MLP has 1024 hidden units, ReLU activations, and a dropout rate of 0.5. The classification accuracy of the VAE is 97.19% which is lower than the accuracy of the SKIP-VAE which is 98.10%. We also studied this classification performance on a weaker model. We replaced the 9-layer PixelCNN and the 3-layer ResNet above by two MLPs. The VAE achieved an accuracy of 97.70% whereas the SKIP-VAE achieved an accuracy of 98.25%.

**Text** Next we analyze the Yahoo Answers dataset from Yang et al. (2017), a common benchmark for deep generative models of text. Successfully training VAEs for text with flexible autoregressive likelihoods such as LSTMs remains an important open issue in the field. In many cases the generative model learns to ignore the latent variable (setting $\mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}; \phi) \,\|\, p(\mathbf{z}))$ close to zero) and collapses to a deterministic language model Bowman et al. (2015).

We use the same training setup as the current best model from Kim et al. (2018). Concretely, the variational neural network is a 1-layer LSTM with 1024 hidden units, whose last hidden state is used to predict the mean vector and the (log) variance vector of the variational posterior. The generative model is also a 1-layer LSTM with 1024 hidden units. In the non-skip generative model the sample from the variational posterior is used to predict the initial hidden state of the decoder and also fed as input at each time step. In the generative skip model we also concatenate the sample with the decoder's hidden state before projecting out to the vocabulary space. In both cases we apply an

additional softmax layer to approximate the predictive distribution over the next word.

In addition to the vanilla VAE, we also study the SA-VAE (Kim et al., 2018), which proposes a different optimization-based strategy for targeting the latent variable collapse issue when training VAEs for text. SA-VAE combines stochastic variational inference (Hoffman et al., 2013) with amortized variational inference by first using an inference network over $\mathbf{x}$ to predict the initial variational parameters and then subsequently running iterative inference on the ELBO to refine the initial variational parameters. In our experiments we used 10 steps of iterative refinement for SA-VAE and SKIP-SA-VAE.

Table 2.4 shows the results. Here we compare performance of adding the skip connections to both VAE and SA-VAE. Table 2.4 shows that SKIP-VAE is better than VAE at avoiding latent variable collapse for similar log likelihoods. The same conclusion holds when comparing SA-VAE and Skip-SA-VAE. Without any skip connections the generative model learns to ignore the latent variable and the mutual information is lower. Adding skip connections increases the mutual information. All in all SKIP-SA-VAE outperforms all models and achieves perfect latent variable usage when the number of latent dimensions is either 32 or 64.

### 2.2.5 Conclusion

We proposed deep generative skip model a class of models that extend EF-PCA. When fit using AVI to scale inference, DGSMS lead to more interpretable latent variables. This is verified on both image and text datasets. The approach consists in using skip connections to promote a stronger dependence between the observations and their associated latent variables.

One interesting line of future work is to study the constraints that should be imposed on the skip model to achieve a good performance—as measured by predictive log-likelihood—while also yielding more expressive latent representations.

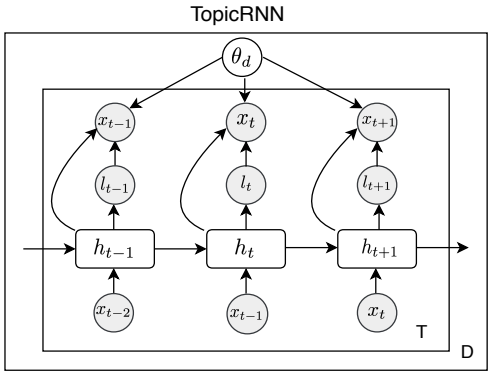



*Figure 2.3.* Graphical model of TopicRNN. There are $D$ documents. Each document has $T$ words $\mathbf{x}_{1:T}$. Observations are shaded in grey, deterministic variables are represented in squares, and latent variables are represented using unshaded circles. The unobserved $\mathbf{h}_{1:T}$ represent the hidden states of the RNN. The observed variables $l_{1:T}$ correspond to stop word indicators: $l_t = 1$ if $\mathbf{x}_t$ is a stop word and 0 otherwise. The latent variable $\boldsymbol{\theta}_d$ is shared by all the words in document $d$. The observation model over a word $\mathbf{x}_t$ is $p(\mathbf{x}_t \mid l_t, \boldsymbol{\theta}_d) = \text{softmax}(V^\top \mathbf{h}_t + (1 - l_t) B^\top \boldsymbol{\theta}_d)$.

## 2.3 Deep Sequential Models with Long-Range Latent Context

One challenge in modeling sequential data is the difficulty to capture long-term dependencies. PGM approaches, such as the hidden Markov model (HMM), make Markov assumptions that prune these long-term dependencies. DL approaches such as recurrent neural networks (RNNS) and their variants (LSTM, gated recurrent unit (GRU)) have unlimited memory, in theory, but face optimization challenges in practice that hinder their ability to capture long-term dependencies (Bengio et al., 1994; Pascanu et al., 2013).

In this section, we introduce a new class of model for sequential data, called TopicRNN, that allows to capture long-range dependencies. TopicRNN marries latent variables with neural networks. The latent variables model the structure shared by all the elements in a sequence whereas the neural network focuses on capturing local dependencies. This marriage has been shown useful for language modeling (Dieng et al., 2016), conversation modeling (Wen & Luong, 2018), patient representation learning for hospital readmission (Xiao et al., 2018a), and unsupervised document representation learning (Dieng et al., 2016).

### 2.3.1 Inductive Biases for Sequential Data Modeling

One advantage of PGM is that it makes it easy to include inductive biases when specifying a model. There are many known inductive biases for discrete sequences. To illustrate what these inductive biases are, let's consider the following excerpt from the CNN news network.

"*The U.S.presidential race isn't only drawing attention and controversy in the United States – it's being closely watched across the globe. But what does the rest of the world think about a campaign that has already thrown up one surprise after another? CNN asked 10 journalists for their take on the race so far, and what their country might be hoping for in America's next —*"

The missing word in this excerpt can be predicted with high accuracy by accounting for two types of contexts: *local* context and *global* context. Local context is defined as the set of few words preceding the word to be predicted. Order matters when defining local context, it defines *syntax* in language. Going back to our example above the phrase "America's next" represents local context for the word we want to predict. It already tells us what the function of the word we want to predict is. Here we know we have to predict a noun. Global context defines the semantics of the word of interest. Order does not matter. The semantic is defined by the semantics of the words that appear in the same paragraph. Here such words are "America", "United States", "race", "country", "campaign", and "presidential". Accounting for this global context narrows down the search for eligible words to "President" and its synonyms. A good language model, a model of sequences of words, should capture at least these two important properties of natural language: syntax (local context) and semantics (global context).

When should we account for which types of context? Local context should always be accounted for. An element of a sequence depends on the elements immediately preceding it. Global context is needed to predict certain elements in the sequence but not all elements in a sequence exhibit long-range dependencies. A priori we do not know which elements in a sequence do require long-term context and which elements do not. Ultimately, we want to discover those elements after we fit a

model to data. In what follows, we make the simple assumption that these elements correspond to *stop words* in language. We will justify this assumption later.

### 2.3.2 Model Class

We now describe TopicRNN. Consider $D$ i.i.d pairs $(\mathbf{x}^{(d)}, \mathbf{l}^{(d)})$ for $d = 1 \dots D$. Here $\mathbf{x}^{(d)} = \mathbf{x}^{(d)}_{1:T}$ is a document and $\mathbf{l}^{(d)} = \mathbf{l}^{(d)}_{1:T}$ is a vector indicating whether each word in document $d$ is a stop word or not. We determine this using a predefined stop word list. Under TopicRNN, a given pair is generated as follows:

1. Draw a global context vector $\theta_d \sim \mathcal{N}(0, I)$
2. For each word in the sequence, $t = 1 \dots T$:
   (a) Compute local context $\mathbf{h}^{(d)}_t = f_\eta\left(\mathbf{x}^{(d)}_{t-1}, \mathbf{h}^{(d)}_{t-1}\right)$
   (b) Draw stop word indicator $l^{(d)}_t \sim \text{Bernoulli}\left(\sigma\left(\Gamma^\top \mathbf{h}^{(d)}_t\right)\right)$
   (c) Draw word $\mathbf{x}^{(d)}_t \sim \text{Cat}\left(\mathbf{p}^d_t\right)$ where $\mathbf{p}^d_t = \text{softmax}\left(\mathbf{V}^\top \mathbf{h}^{(d)}_t + (1 - l^{(d)}_t) \cdot \beta^\top \theta_d\right)$

Figure 2.3 shows the graphical model corresponding to this generative process. Here $\theta_d$ represents global context, it is shared across all the words in the document. We chose its prior to be a standard Gaussian. The function $f_\eta(\cdot)$ is a neural network that takes as input the previous input $\mathbf{x}^{(d)}_{t-1}$ and its own previous output $\mathbf{h}^{(d)}_{t-1}$. It can be implemented using any of the RNN cells described in Chapter 1. The function $\sigma(\cdot)$ is the logistic function. The stop word indicator $l^{(d)}_t$ controls how the latent global context $\theta_d$ affects the output. If $l^{(d)}_t = 1$ (indicating $\mathbf{x}^{(d)}_t$ is a stop word), the global context $\theta_d$ has no contribution to the output. Otherwise, we add a bias to favor those words that are more likely to appear when mixing with $\theta_d$, as measured by the dot product between $\theta$ and the latent word vector $b_i$ for the $i$th vocabulary word.

To understand the bias term $\beta^\top \theta_d$ recall latent Dirichlet allocation (LDA), a probabilistic topic model described in Chapter 1. Marginalize out its per-word discrete topic assignments, the conditional

distribution of the words in the document given the topics and topic proportions under LDA is

$$p(\mathbf{x}_{1:T}^{(d)}|\theta_d, \boldsymbol{\beta}) = \prod_{t=1}^{T} \sum_{k=1}^{K} \theta_{dk} \boldsymbol{\beta}_{\mathbf{x}_t^{(d)}} = \prod_{t=1}^{T} \boldsymbol{\beta}^\top \theta_d \Big|_{\mathbf{x}_t^{(d)}} \tag{2.4}$$

This implies $\mathbf{x}_t^{(d)}|\theta_d, \boldsymbol{\beta} \sim \boldsymbol{\beta}^\top \theta_d$. Although this term has the same form as the bias term in TopicRNN the constraints put on the matrices $\boldsymbol{\beta}$ and $\theta_d$ are different in LDA and in TopicRNN. In LDA both $\boldsymbol{\beta}$ and $\theta_d$ are modeled using the Dirichlet distribution whereas in TopicRNN they are modeled as a deterministic model parameter and a standard Gaussian. Adding a simplex constraints to $\theta_d$ enforces interpretability for $\boldsymbol{\beta}$ (Donoho & Stodden, 2004) We can achieve this in TopicRNN by simply mapping the global context vector $\theta_d$ to softmax$(\cdot)$. In our empirical study we use a standard Gaussian for simplicity and found the matrix $\boldsymbol{\beta}$ still captures interpretable word clusters.

### 2.3.3 Amortized Variational Inference

We fit TopicRNN using AVI. For that we have to specify a variational family over the global context $\theta_d$. Denote by $q_\phi\left(\theta_{1:D}\middle|\mathbf{x}_{1:T}^{(1:D)}, \mathbf{l}_{1:T}^{(1:D)}\right)$ the variational family; it is indexed by $\phi$. We factorize it as

$$q_\phi\left(\theta_{1:D}\middle|\mathbf{x}_{1:T}^{(1:D)}, \mathbf{l}_{1:T}^{(1:D)}\right) = q_\phi\left(\theta_d\middle|\mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}\right)$$

Each factor $q_\phi\left(\theta_d\middle|\mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}\right)$ is a Gaussian whose mean and covariance are given by the outputs of neural networks,

$$q_\phi\left(\theta_d\middle|\mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}\right) = \mathcal{N}\left(\boldsymbol{\mu}^{(d)}\left(\mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}; \phi_\mu\right), \boldsymbol{\Sigma}^{(d)}\left(\mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}; \phi_\Sigma\right)\right)$$

Here $\phi = \left(\phi_\mu, \phi_\Sigma\right)$. In practice we use one shared neural network whose output we map to the mean and the covariance using two different sets of weight matrices. The input of this shared neural network is the bag-of-word representation of the document $\mathbf{x}_{1:T}^{(d)}$ multiplied by $1 - \mathbf{l}_{1:T}^{(d)}$. This

multiplication zeroes out the contributions of the stop words in the inference of $\theta_d$.

We can now form the ELBO,

$$\text{ELBO} = \sum_{d=1}^{D} \mathbb{E}_{q_\phi\left(\theta_d \middle| \mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}\right)} \left\{\log p\left(\mathbf{x}_t^{(d)} | \mathbf{l}_t^{(d)}, \theta_d\right)\right\} - \text{KL}\left(q_\phi\left(\theta_d \middle| \mathbf{x}_{1:T}^{(d)}, \mathbf{l}_{1:T}^{(d)}\right) \middle\| p(\theta_d)\right).$$

The ELBO is intractable. We approximate it using Monte Carlo with the reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014). Importantly, we apply truncated backpropagation through time, which unrolls the RNN $f_\eta(\cdot)$ a fixed number of time steps (instead of accounting for all the elements in the sequence of words when computing the states of the RNN, which would cause optimization issues (Bengio et al., 1994; Pascanu et al., 2013).)

### 2.3.4 Application to Language Modeling

We first tested TopicRNN on the word prediction task using the Penn Treebank (PTB) portion of the Wall Street Journal. We use the standard split, where sections 0-20 (930K tokens) are used for training, sections 21-22 (74K tokens) for validation, and sections 23-24 (82K tokens) for testing (Mikolov et al., 2010). We use a vocabulary of size $10K$ that includes the special token *unk* for rare words and *eos* that indicates the end of a sentence. TopicRNN takes documents as inputs. We split the PTB data into blocks of 10 sentences to constitute documents as done by (Mikolov & Zweig, 2012). The inference network takes as input the bag-of-words representation of the input document. For that reason, the vocabulary size of the inference network is reduced to 9551 after excluding 449 pre-defined stop words.

In order to compare with previous work on contextual RNNs (e.g. Mikolov & Zweig (2012)), we trained TopicRNN using different network sizes. We performed word prediction using a recurrent neural network with 10 neurons, 100 neurons and 300 neurons. For these experiments, we used a multilayer perceptron with 2 hidden layers and 200 hidden units per layer for the inference network. The dimensionality $K$ of $\theta_d$ was tuned depending on the size of the RNN. For 10 neurons we

*Table 2.5.* TopicRNN and its counterparts exhibit lower perplexity scores across different network sizes. These results prove TopicRNN has more generalization capabilities: for example we only need a TopicGRU with 100 neurons to achieve a better perplexity than stacking 2 LSTMs with 200 neurons each: 112.4 vs 115.9)

| | 10 Neurons | | 100 Neurons | | 300 Neurons | |
|---|---|---|---|---|---|---|
| Method | Val | Test | Val | Test | Val | Test |
| RNN | 239.2 | 225.0 | 150.1 | 142.1 | – | 124.7 |
| RNN + LDA | 197.3 | 187.4 | 132.3 | 126.4 | – | 113.7 |
| TopicRNN | 184.5 | 172.2 | 128.5 | 122.3 | 118.3 | 112.2 |
| TopicLSTM | 188.0 | 175.0 | 126.0 | 118.1 | 104.1 | 99.5 |
| TopicGRU | 178.3 | 166.7 | 118.3 | 112.4 | 99.6 | 97.3 |

used $K = 18$. For 100 and 300 neurons we chose $K = 50$. We used the validation set to tune the hyperparameters of the model (including $K$). We used a maximum of 15 epochs for the experiments and performed early stopping using the validation set. For comparison purposes we did not apply regularization and used 1 layer for the RNN and its counterparts in all the experiments.

Table 2.5 reports perplexity on the validation set and the test set for different network sizes. Perplexity can be thought of as a measure of surprise for a language model. It is defined as the exponential of the average negative log likelihood. We learn three things from Table 2.5. First, the perplexity is reduced the larger the network size. Second, RNNs with global context features perform better than RNNs without context features. Third, we see that TopicRNN gives better perplexity than the previous baseline result reported by Mikolov & Zweig (2012). Note we compute the perplexity scores for word prediction using a sliding window, to compute $\theta$ as we move along the sequences. The topic vector $\theta$ that is used from the current batch of words is estimated from the previous batch of words. This enables fair comparison to previously reported results (Mikolov & Zweig, 2012).

### 2.3.5 Unsupervised Feature Learning and Application to Sentiment Analysis

We performed sentiment analysis using TopicRNN as a feature extractor on the IMDB 100K dataset. This data consists of 100,000 movie reviews from the Internet Movie Database (IMDB) website. The data is split into 75% for training and 25% for testing. Among the 75K training reviews, 50K

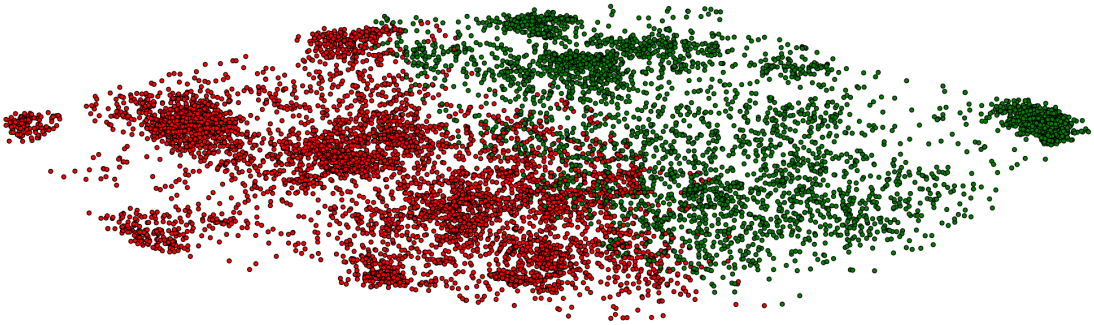


*Figure 2.4.* Clusters of a sample of 10000 movie reviews from the IMDB 100K dataset using TopicRNN as feature extractor. We used K-Means to cluster the feature vectors. We then used PCA to reduce the dimension to two for visualization purposes. red is a negative review and green is a positive review.

are unlabelled and 25K are labelled as carrying either a positive or a negative sentiment. All 25K test reviews are labelled. We trained TopicRNN on 65K random training reviews and used the remaining 10K reviews for validation. To learn a classifier, we passed the 25K labelled training reviews through the learned TopicRNN model. We then concatenated the output of the inference network and the last state of the RNN for each of these 25K reviews to compute the feature vectors. We then used these feature vectors to train a neural network with one hidden layer, 50 hidden units, and a sigmoid activation function to predict sentiment, exactly as done in Le & Mikolov (2014b).

To train the TopicRNN model, we used a vocabulary of size 5,000 and mapped all other words to the *unk* token. We took out 439 stop words to create the input of the inference network. We used 500 units and 2 layers for the inference network, and used 2 layers and 300 units per-layer for the RNN. We chose a step size of 5 and defined 200 topics. We did not use any regularization such as dropout. We trained the model for 13 epochs and used the validation set to tune the hyperparameters of the model and track perplexity for early stopping. This experiment took close to 78 hours on a MacBook pro quad-core with 16GHz of RAM.

Table 2.6 summarizes sentiment classification results from TopicRNN and other methods. Our

*Table 2.6.* Classification error rate on IMDB 100k dataset. TopicRNN achieves state of the art error rate amongst methods that first perform unsupervised feature extraction before doing sentiment classification.

| Model | Reported Error rate |
|---|---|
| BoW (bnc) (Maas et al., 2011) | 12.20% |
| BoW ($b\Delta$ tć) (Maas et al., 2011) | 11.77% |
| LDA (Maas et al., 2011) | 32.58% |
| Full + BoW (Maas et al., 2011) | 11.67% |
| Full + Unlabelled + BoW (Maas et al., 2011) | 11.11% |
| WRRBM (Dahl et al., 2012) | 12.58% |
| WRRBM + BoW (bnc) (Dahl et al., 2012) | 10.77% |
| MNB-uni (Wang & Manning, 2012) | 16.45% |
| MNB-bi (Wang & Manning, 2012) | 13.41% |
| SVM-uni (Wang & Manning, 2012) | 13.05% |
| SVM-bi (Wang & Manning, 2012) | 10.84% |
| NBSVM-uni (Wang & Manning, 2012) | 11.71% |
| seq2-bown-CNN (Johnson & Zhang, 2014) | 14.70% |
| NBSVM-bi (Wang & Manning, 2012) | 8.78% |
| Paragraph Vector (Le & Mikolov, 2014) | 7.42% |
| SA-LSTM with joint training (Dai & Le, 2015) | 14.70% |
| LSTM with tuning and dropout (Dai & Le, 2015) | 13.50% |
| LSTM initialized with word2vec embeddings (Dai & Le, 2015) | 10.00% |
| SA-LSTM with linear gain (Dai & Le, 2015) | 9.17% |
| LM-TM (Dai & Le, 2015) | 7.64% |
| SA-LSTM (Dai & Le, 2015) | 7.24% |
| **Virtual Adversarial (Miyato et al. 2016)** | **5.91%** |
| **TopicRNN** | **6.28%** |

error rate is 6.28%.[2] This is close to the state-of-the-art 5.91% (Miyato et al., 2016) despite that we do not use the labels and adversarial training in the feature extraction stage. Our approach is most similar to Le & Mikolov (2014b), where the features were extracted in a unsupervised way and then a one-layer neural net was trained for classification.

Figure 2.4 shows the ability of TopicRNN to cluster documents using the feature vectors as created during the sentiment analysis task. Reviews with positive sentiment are colored in green while reviews carrying negative sentiment are shown in red.

[2]The experiments were solely based on TopicRNN. Experiments using TopicGRU/TopicLSTM are being carried out and will be added as an extended version of this paper.

### 2.3.6 Conclusion

We introduced TopicRNN, a class of model for sequential data that marries latent variables and neural networks. The latent variables model the structure shared between all the elements of a sequence whereas the neural network models local dependencies. TopicRNN yields competitive per-word perplexity on the Penn Treebank benchmark dataset. It is effective at learning unsupervised document features for sentiment classification on the IMDB benchmark dataset. TopicRNN has also been applied to healthcare data where learning meaningful patient representations can help predict hospital readmission (Xiao et al., 2018a). Finally, it has been used for conversation modeling by Wen & Luong (2018). Future work can study the performance of TopicRNN when words that do not need global context are learned instead of chosen to be stop words of language.

## 2.4 Topic Modeling in Embedding Spaces

Topic models are statistical tools for discovering the hidden semantic structure in a collection of documents (Blei et al., 2003; Blei, 2012). Topic models and their extensions have been applied to many fields, such as marketing, sociology, political science, and the digital humanities. Boyd-Graber et al. (2017) provide a review.

Most topic models build on LDA (Blei et al., 2003), which we described in Chapter 1. LDA is a powerful model and it is widely used. However, it suffers from a pervasive technical problem—it fails in the face of large vocabularies. Practitioners must severely prune their vocabularies in order to fit good topic models, i.e., those that are both predictive and interpretable. This is typically done by removing the most and least frequent words (called *stop words* and *rare words* respectively.) On large collections, this pruning may remove important terms and limit the scope of the models. The problem of topic modeling with large vocabularies has yet to be addressed in the research literature.

In this section we describe how we leverage word embeddings, a successful advance of DL, to solve

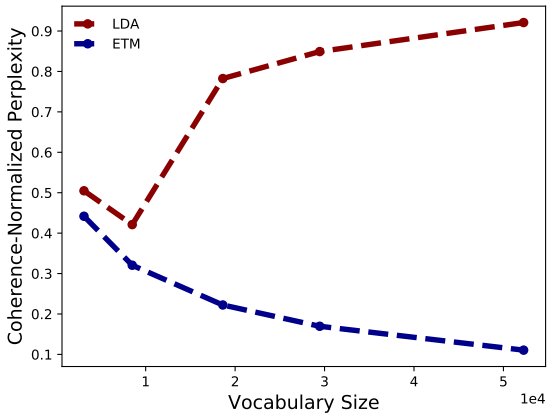



*Figure 2.5.* Ratio of the held-out perplexity on a document completion task and the topic coherence as a function of the vocabulary size for the ETM and LDA on the *20NewsGroup* corpus. The perplexity is normalized by the size of the vocabulary. While the performance of LDA deteriorates for large vocabularies, the ETM maintains good performance.

the problems described above. We develop the *embedded topic model* (ETM) (Dieng et al., 2019c). The ETM marries LDA and word embeddings to enable flexible topic modeling at large scale. The flexibility provided by the word embeddings will allow us to not have to prune stop words and rare words to learn interpretable topics. Furthermore, we devise an efficient AVI procedure to fit the ETM. The recognition network used for AVI will allow us to perform evaluation on new documents without running an optimization loop, as required in LDA. Other benefits brought in by the use of the word embeddings in the context of topic modeling is that novel unseen words can be assigned to a given topic.

Figure 2.5 illustrates one of the advantages of the ETM over LDA. This figure shows the ratio between the perplexity on held-out documents (a measure of predictive performance) and the topic coherence (a measure of the quality of the topics), as a function of the size of the vocabulary. (The perplexity has been normalized by the vocabulary size.) This is for a corpus of 11.2K articles from the *20NewsGroup* and for 100 topics. The red line is LDA; its performance deteriorates as the vocabulary size increases—the predictive performance and the quality of the topics get worse. The blue line is the ETM; it maintains good performance, even as the vocabulary size gets large.

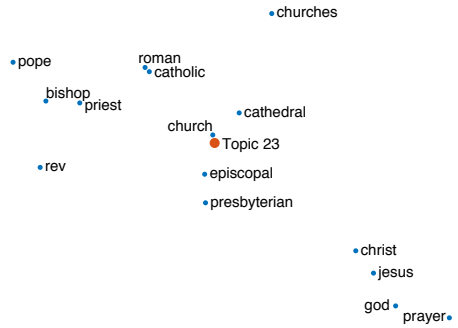



*Figure 2.6.* A topic about Christianity found by the ETM on *The New York Times*. The topic is a point in the word embedding space.

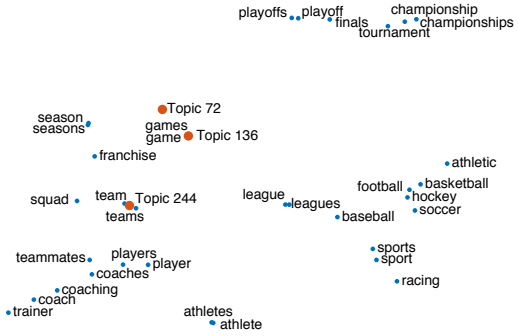



*Figure 2.7.* Topics about sports found by the ETM on *The New York Times*. Each topic is a point in the word embedding space.

Figures 2.6 and 2.7 show topics in the embedding space of words from a 300-topic ETM of *The New York Times*. These topics are about Christianity and sports.

We now describe the ETM in detail.

### 2.4.1 The Embedded Topic Model

The ETM is a topic model that uses embedding representations of both words and topics. It contains two notions of latent dimension. First, it embeds the vocabulary in an $L$-dimensional space. These embeddings are similar in spirit to classical word embeddings. Second, it represents each document in terms of $K$ latent topics.

In traditional topic modeling, each topic is a full distribution over the vocabulary. In the ETM, however, the $k^{\text{th}}$ topic is a vector $\alpha_k \in \mathbb{R}^L$ in the embedding space. We call $\alpha_k$ a *topic embedding*—it is a distributed representation of the $k^{\text{th}}$ topic in the semantic space of words.

In its generative process, the ETM uses the topic embedding to form a per-topic distribution over the vocabulary. Specifically, the ETM uses a log-linear model that takes the inner product of the word embedding matrix and the topic embedding. With this form, the ETM assigns high probability to a word $v$ in topic $k$ by measuring the agreement between the word's embedding and the topic's embedding.

Denote the $L \times V$ word embedding matrix by $\rho$; the column $\rho_v$ is the embedding of term $v$. Under the ETM, the generative process of the $d^{\text{th}}$ document is the following:

1. Draw topic proportions $\theta_d \sim \mathcal{LN}(0, I)$.
2. For each word $n$ in the document:
   a. Draw topic assignment $z_{dn} \sim \text{Cat}(\theta_d)$.
   b. Draw the word $w_{dn} \sim \text{softmax}(\rho^\top \alpha_{z_{dn}})$.

In Step 1, $\mathcal{LN}(\cdot)$ denotes the logistic-normal distribution (Aitchison & Shen, 1980; Blei & Lafferty, 2007); it transforms a standard Gaussian random variable to the simplex. A draw $\theta_d$ from this distribution is obtained as

$$\delta_d \sim \mathcal{N}(0, I); \quad \theta_d = \text{softmax}(\delta_d). \tag{2.5}$$

(We replaced the Dirichlet with the logistic normal to easily use reparameterization in the inference algorithm; see Section 2.4.2.)

Steps 1 and 2a are standard for topic modeling: they represent documents as distributions over topics and draw a topic assignment for each observed word. Step 2b is different; it uses the embeddings of the vocabulary $\rho$ and the assigned topic embedding $\alpha_{z_{dn}}$ to draw the observed word from the assigned topic, as given by $z_{dn}$.

The topic distribution in Step 2b mirrors the continuous bag-of-words (CBOW) likelihood in Eq. 1.35. Recall CBOW uses the surrounding words to form the context vector $\alpha_{dn}$. In contrast, the ETM uses the topic embedding $\alpha_{z_{dn}}$ as the context vector, where the assigned topic $z_{dn}$ is drawn from the per-document variable $\theta_d$. The ETM draws its words from a document context, rather than from a window of surrounding words.

The ETM likelihood uses a matrix of word embeddings $\rho$, a representation of the vocabulary in a lower dimensional space. In practice, it can either rely on previously fitted embeddings or learn them as part of its overall fitting procedure. When the ETM learns the embeddings as part of the

fitting procedure, it simultaneously finds topics and an embedding space.

When the ETM uses previously fitted embeddings, it learns the topics of a corpus in a particular embedding space. This strategy is particularly useful when there are words in the embedding that are not used in the corpus. The ETM can hypothesize how those words fit in to the topics because it can calculate $\rho_v^\top \alpha_k$, even for words $v$ that do not appear in the corpus.

### 2.4.2 Inference and Estimation

We are given a corpus of documents $\{\mathbf{w}_1, \ldots, \mathbf{w}_D\}$, where $\mathbf{w}_d$ is a collection of $N_d$ words. How do we fit the ETM?

**The marginal likelihood.** The parameters of the ETM are the embeddings $\rho_{1:V}$ and the topic embeddings $\alpha_{1:K}$; each $\alpha_k$ is a point in the embedding space. We maximize the marginal likelihood of the documents,

$$\mathcal{L}(\alpha, \rho) = \sum_{d=1}^{D} \log p(\mathbf{w}_d \mid \alpha, \rho). \tag{2.6}$$

The problem is that the marginal likelihood of each document is intractable to compute. It involves a difficult integral over the topic proportions, which we write in terms of the untransformed proportions $\delta_d$ in Eq. 2.5,

$$p(\mathbf{w}_d \mid \alpha, \rho) = \int p(\delta_d) \prod_{n=1}^{N_d} p(w_{dn} \mid \delta_d, \alpha, \rho) \, \mathrm{d}\delta_d. \tag{2.7}$$

The conditional distribution of each word marginalizes out the topic assignment $z_{dn}$,

$$p(w_{dn} \mid \delta_d, \alpha, \rho) = \sum_{k=1}^{K} \theta_{dk} \beta_{k,w_{dn}}. \tag{2.8}$$

Here, $\theta_{dk}$ denotes the (transformed) topic proportions (Eq. 2.5) and $\beta_{kv}$ denotes a traditional "topic,"

i.e., a distribution over words, induced by the word embeddings $\rho$ and the topic embedding $\alpha_k$,

$$\beta_{kv} = \text{softmax}(\rho^\top \alpha_k)|_v. \tag{2.9}$$

Eqs. 2.7 to 2.9 flesh out the likelihood in Eq. 2.6.

**Variational inference.** We sidestep the intractable integral with variational inference, which we reviewed in Chapter 1. Variational inference optimizes a sum of per-document bounds on the log of the marginal likelihood of Eq. 2.7. There are two sets of parameters to optimize: the model parameters, as described above, and the variational parameters, which tighten the bounds on the marginal likelihoods.

To begin, posit a family of distributions of the untransformed topic proportions $q(\delta_d\,;\,\mathbf{w}_d, \nu)$. We use AVI, where the variational distribution of $\delta_d$ depends on both the document $\mathbf{w}_d$ and shared variational parameters $\nu$. In particular $q(\delta_d\,;\,\mathbf{w}_d, \nu)$ is a Gaussian whose mean and variance come from an "inference network," a neural network parameterized by $\nu$ (Kingma & Welling, 2014). The inference network ingests the document $\mathbf{w}_d$ and outputs a mean and variance of $\delta_d$. (To accommodate documents of varying length, we form the input of the inference network by normalizing the bag-of-word representation of the document by the number of words $N_d$.)

We use this family of variational distributions to bound the log-marginal likelihood. The ELBO is a function of the model parameters and the variational parameters,

$$\mathcal{L}(\alpha, \rho, \nu) = \sum_{d=1}^{D}\sum_{n=1}^{N_d} \mathbb{E}_q\left[\log p(w_{nd} \mid \delta_d, \rho, \alpha)\right] - \sum_{d=1}^{D} \text{KL}(q(\delta_d; \mathbf{w}_d, \nu) \mid\mid p(\delta_d)). \tag{2.10}$$

The first term of the ELBO (Eq. 2.10) encourages variational distributions $q(\delta_d\,;\,\mathbf{w}_d, \nu)$ that place mass on unnormalized topic proportions $\delta_d$ that explain the observed words while the second term encourages $q(\delta_d\,;\,\mathbf{w}_d, \nu)$ to be close to the prior $p(\delta_d)$. Maximizing the ELBO with respect to the model parameters $(\alpha, \rho)$ is equivalent to maximizing the expected complete log-likelihood, $\sum_d \log p(\delta_d, \mathbf{w}_d \mid \alpha, \rho)$.

The ELBO in Eq. 2.10 is intractable because the expectation is intractable. However we can use Monte Carlo to approximate the ELBO,

$$\tilde{\mathcal{L}}(\alpha, \rho, \nu) = \frac{1}{S} \sum_{d=1}^{D} \sum_{n=1}^{N_d} \sum_{s=1}^{S} \log p(w_{nd} \mid \delta_d^{(s)}, \rho, \alpha) - \sum_{d=1}^{D} \mathrm{KL}(q(\delta_d; \mathbf{w}_d, \nu) \mid\mid p(\delta_d)) \tag{2.11}$$

where $\delta_d^{(s)} \sim q(\delta_d; \mathbf{w}_d, \nu)$ for $s = 1 \ldots S$. To reduce variance we use the reparameterization trick when sampling the unnormalized proportions $\delta_d^{(1)}, \ldots, \delta_d^{(S)}$(Kingma & Welling, 2014; Titsias & Lázaro-Gredilla, 2014; Rezende et al., 2014). That is, we sample $\delta_d^{(s)}$ from $q(\delta_d; \mathbf{w}_d, \nu)$ as

$$\epsilon_d^{(s)} \sim \mathcal{N}(0, I) \text{ and } \delta_d^{(s)} = \mu_d + \Sigma_d^{\frac{1}{2}} \odot \epsilon_d^{(s)} \tag{2.12}$$

where $\mu_d$ and $\Sigma_d$ are the mean and covariance of $q(\delta_d; \mathbf{w}_d, \nu)$ respectively.

We also use data subsampling to handle large collections of documents (Hoffman et al., 2013) and set $S = 1$. Denote by $\mathcal{B}$ a minibatch of documents. Then the approximation of the ELBO using data subsampling is

$$\tilde{\mathcal{L}}(\alpha, \rho, \nu) = \frac{D}{|\mathcal{B}|} \sum_{d \in \mathcal{B}} \sum_{n=1}^{N_d} \log p(w_{nd} \mid \delta_d, \rho, \alpha) - \frac{D}{|\mathcal{B}|} \sum_{d \in \mathcal{B}} \mathrm{KL}(q(\delta_d; \mathbf{w}_d, \nu) \mid\mid p(\delta_d)) \tag{2.13}$$

Finally, given the prior $p(\delta_d)$ and $q(\delta_d; \mathbf{w}_d, \nu)$ are both Gaussians, the KL is closed-form,

$$\mathrm{KL}(q(\delta_d; \mathbf{w}_d, \nu) \mid\mid p(\delta_d)) = \frac{1}{2} \left\{ \mathrm{tr}(\Sigma_d) + \mu_d^\top \mu_d - \log \det(\Sigma_d) - K \right\}. \tag{2.14}$$

Here both $\mu_d$ and $\Sigma_d$ depend implicitly on $\nu$ and $\mathbf{w}_d$ via the inference network.

We optimize the ELBO with respect to both the model parameters $(\alpha, \rho)$ and the variational parameters $\nu$. We set the learning rate with Adam (Kingma & Ba, 2015). The procedure is shown in Algorithm 1, where the notation $\mathrm{NN}(\mathbf{x}\,;\,\nu)$ represents a neural network with input $\mathbf{x}$ and parameters $\nu$.

**Algorithm 1:** Flexible topic modeling with the ETM

Initialize model and variational parameters
**for** iteration $i = 1, 2, \ldots$ **do**
  Compute $\beta_k = \text{softmax}(\rho^\top \alpha_k)$ for each topic $k$
  Choose a minibatch $\mathcal{B}$ of documents
  **for** each document $d$ in $\mathcal{B}$ **do**
    Get normalized bag-of-word representat. $\mathbf{x}_d$
    Compute $\mu_d = \text{NN}(\mathbf{x}_d\,;\, \nu_\mu)$
    Compute $\Sigma_d = \text{NN}(\mathbf{x}_d\,;\, \nu_\Sigma)$
    Sample $\theta_d \sim \mathcal{LN}(\mu_d, \Sigma_d)$
    **for** each word in the document **do**
      Compute $p(w_{dn} \mid \theta_d) = \theta_d^\top \beta_{\cdot, w_{dn}}$
    **end for**
  **end for**
  Estimate the ELBO and its gradient (backprop.)
  Update model parameters $\alpha_{1:K}$
  Update variational parameters $(\nu_\mu, \nu_\Sigma)$
**end for**

### 2.4.3 Related Work

One of the goals in developing the ETM is to incorporate word similarity into the topic model, and there is previous research that shares this goal. These methods either modify the topic priors (Petterson et al., 2010; Zhao et al., 2017b; Shi et al., 2017; Zhao et al., 2017a) or the topic assignment priors (Xie et al., 2015). For example Petterson et al. (2010) use a word similarity graph (as given by a thesaurus) to bias LDA towards assigning similar words to similar topics. As another example, Xie et al. (2015) model the per-word topic assignments of LDA using a Markov random field to account for both the topic proportions and the topic assignments of similar words. These methods use word similarity as a type of "side information" about language; in contrast, the ETM directly models the similarity (via embeddings) in its generative process of words.

Other work has extended LDA to directly involve word embeddings. One common strategy is to convert the discrete text into continuous observations of embeddings, and then adapt LDA to generate real-valued data (Das et al., 2015; Xun et al., 2016; Batmanghelich et al., 2016; Xun et al., 2017). With this strategy, topics are Gaussian distributions with latent means and covariances, and

the likelihood over the embeddings is modeled with a Gaussian (Das et al., 2015) or a Von-Mises Fisher distribution (Batmanghelich et al., 2016). The ETM differs from these approaches in that it is a model of categorical data, one that goes through the embeddings matrix. Thus it does not require pre-fitted embeddings and, indeed, can learn embeddings as part of its inference process.

There have been a few other ways of combining LDA and embeddings. Nguyen et al. (2015) mix the likelihood defined by LDA with a log-linear model that uses pre-fitted word embeddings; Bunk & Krestel (2018) randomly replace words drawn from a topic with their embeddings drawn from a Gaussian; and Xu et al. (2018) adopt a geometric perspective, using Wasserstein distances to learn topics and word embeddings jointly.

Another thread of recent research improves topic modeling inference through deep neural networks (Srivastava & Sutton, 2017; Card et al., 2017; Cong et al., 2017; Zhang et al., 2018). Specifically, these methods reduce the dimension of the text data through amortized inference and the variational auto-encoder (Kingma & Welling, 2014; Rezende et al., 2014). To perform inference in the ETM, we also avail ourselves of amortized inference methods (Gershman & Goodman, 2014).

Finally, as a document model, the ETM also relates to works that learn per-document representations as part of an embedding model (Le & Mikolov, 2014a; Moody, 2016; Miao et al., 2016). In contrast to these works, the document variables in the ETM are part of a larger probabilistic topic model.

### 2.4.4 Empirical Study

We study the performance of the ETM and compare it to other unsupervised document models. A good document model should provide both coherent patterns of language and an accurate distribution of words, so we measure performance in terms of both predictive accuracy and topic interpretability. We measure accuracy with log-likelihood on a document completion task (Rosen-Zvi et al., 2004; Wallach et al., 2009); we measure topic interpretability as a blend of topic coherence and diversity. We find that, of the interpretable models, the ETM is the one that provides better

*Table 2.7.* Word embeddings learned by all document models (and skip-gram) on the *New York Times* with vocabulary size 118,363.

Skip-gram embeddings

| **love** | **family** | **woman** | **politics** |
|---|---|---|---|
| loved | families | man | political |
| passion | grandparents | girl | religion |
| loves | mother | boy | politicking |
| affection | friends | teenager | ideology |
| adore | relatives | person | partisanship |

ETM embeddings

| **love** | **family** | **woman** | **politics** |
|---|---|---|---|
| joy | children | girl | political |
| loves | son | boy | politician |
| loved | mother | mother | ideology |
| passion | father | daughter | speeches |
| wonderful | wife | pregnant | ideological |

NVDM embeddings

| **love** | **family** | **woman** | **politics** |
|---|---|---|---|
| loves | sons | girl | political |
| passion | life | women | politician |
| wonderful | brother | man | politicians |
| joy | son | pregnant | politically |
| beautiful | lived | boyfriend | democratic |

$\Delta$-NVDM embeddings

| **love** | **family** | **woman** | **politics** |
|---|---|---|---|
| miss | home | life | political |
| young | father | marriage | faith |
| born | son | women | marriage |
| dream | day | read | politicians |
| younger | mrs | young | election |

PRODLDA embeddings

| **love** | **family** | **woman** | **politics** |
|---|---|---|---|
| loves | husband | girl | political |
| affection | wife | boyfriend | politician |
| sentimental | daughters | boy | liberal |
| dreams | sister | teenager | politicians |
| laugh | friends | ager | ideological |

predictions and topics.

In a separate analysis, we study the robustness of each method in the presence of stop words. Standard topic models fail in this regime—since stop words appear in many documents, every learned topic includes some stop words, leading to poor topic interpretability. In contrast, the ETM is able to use the information from the word embeddings to provide interpretable topics.

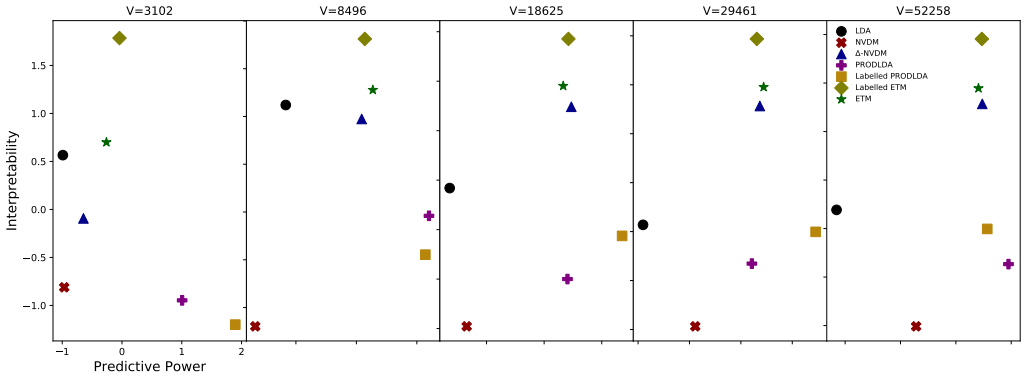
**Corpora.** We study the *20Newsgroups* corpus and the *New York Times* corpus.

The *20Newsgroup* corpus is a collection of newsgroup posts. We preprocess the corpus by filtering stop words, words with document frequency above 70%, and tokenizing. To form the vocabulary, we keep all words that appear in more than a certain number of documents, and we vary the thresh-

*Table 2.8.* Top five words of seven most used topics from different document models on 1.8M documents of the *New York Times* corpus with vocabulary size 212,237 and $K = 300$ topics.

| | | | | | | |
|---|---|---|---|---|---|---|
| LDA | | | | | | |
| time | year | officials | mr | city | percent | state |
| day | million | public | president | building | million | republican |
| back | money | department | bush | street | company | party |
| good | pay | report | white | park | year | bill |
| long | tax | state | clinton | house | billion | mr |
| NVDM | | | | | | |
| scholars | japan | gansler | spratt | assn | ridership | pryce |
| gingrich | tokyo | wellstone | tabitha | assoc | mtv | mickens |
| funds | pacific | mccain | mccorkle | qtr | straphangers | mckechnie |
| institutions | europe | shalikashvili | cheetos | yr | freierman | mfume |
| endowment | zealand | coached | vols | nyse | riders | filkins |
| $\Delta$-NVDM | | | | | | |
| concerto | servings | nato | innings | treas | patients | democrats |
| solos | tablespoons | soviet | scored | yr | doctors | republicans |
| sonata | tablespoon | iraqi | inning | qtr | medicare | republican |
| melodies | preheat | gorbachev | shutout | outst | dr | senate |
| soloist | minced | arab | scoreless | telerate | physicians | dole |
| PRODLDA | | | | | | |
| temptation | grasp | electron | played | amato | briefly | giant |
| repressed | unruly | nuclei | lou | model | precious | boarding |
| drowsy | choke | macal | greg | delaware | serving | bundle |
| addiction | drowsy | trained | bobby | morita | set | distance |
| conquering | drift | mediaone | steve | dual | virgin | foray |
| Labelled PRODLDA | | | | | | |
| mercies | cheesecloth | scoreless | chapels | distinguishable | floured | gillers |
| lockbox | overcook | floured | magnolias | cocktails | impartiality | lacerated |
| pharm | strainer | hitless | asea | punishable | knead | polshek |
| shims | kirberger | asterisk | bogeyed | checkpoints | refrigerate | decimated |
| cp | browned | knead | birdie | disobeying | tablespoons | inhuman |
| Labelled ETM | | | | | | |
| music | republican | yankees | game | wine | court | company |
| dance | bush | game | points | restaurant | judge | million |
| songs | campaign | baseball | season | food | case | stock |
| opera | senator | season | team | dishes | justice | shares |
| concert | democrats | mets | play | restaurants | trial | billion |
| ETM | | | | | | |
| game | music | united | wine | company | yankees | art |
| team | mr | israel | food | stock | game | museum |
| season | dance | government | sauce | million | baseball | show |
| coach | opera | israeli | minutes | companies | mets | work |
| play | band | mr | restaurant | billion | season | artist |

*Figure 2.8.* Interpretability as measured by the exponentiated product of topic coherence and topic diversity (the higher the better) vs. predictive performance as measured by log-likelihood on document completion (the higher the better) on the *20NewsGroup* dataset. Both interpretability and predictive power metrics are normalized by subtracting the mean and dividing by the standard deviation across models. Better models are on the top right corner. Overall, the ETM is a better topic model.



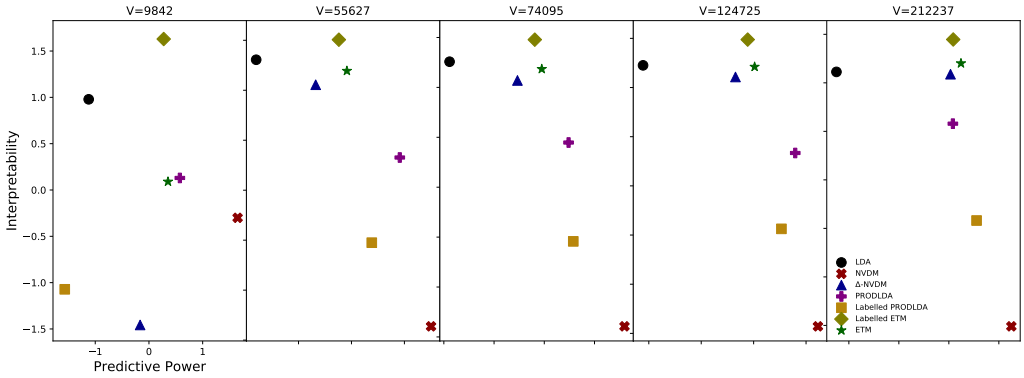

old from 100 (a smaller vocabulary, where $V = 3{,}102$) to 2 (a larger vocabulary, where $V = 52{,}258$). After preprocessing, we further remove one-word documents from the validation and test sets. We split the corpus into a training set of 11,260 documents, a test set of 7,532 documents, and a validation set of 100 documents.

The *New York Times* corpus is a larger collection of news articles. It contains more than 1.8 million articles, spanning the years 1987–2007. We follow the same preprocessing steps as for *20Newsgroups*. We form versions of this corpus with vocabularies ranging from $V = 5{,}921$ to $V = 212{,}237$. After preprocessing, we use 85% of the documents for training, 10% for testing, and 5% for validation.

**Models.** We compare the performance of the ETM against several document models. We briefly describe each below.

We consider latent Dirichlet allocation (LDA) (Blei et al., 2003), a standard topic model that posits Dirichlet priors for the topics $\beta_k$ and topic proportions $\theta_d$. (We set the prior hyperparameters to 1.) It is a conditionally conjugate model, amenable to variational inference with coordinate ascent. We consider LDA because it is the most commonly used topic model, and it has a similar generative

*Figure 2.9.* Interpretability as measured by the exponentiated product of topic coherence and topic diversity (the higher the better) vs. predictive performance as measured by log-likelihood on document completion (the higher the better) on the *New York Times* dataset. Both interpretability and predictive power metrics are normalized by subtracting the mean and dividing by the standard deviation across models. Better models are on the top right corner. Overall, the ETM is a better topic model.




process as the ETM.

We also consider the neural variational document model (NVDM) (Miao et al., 2016). The NVDM is a multinomial factor model of documents; it posits the likelihood $w_{dn} \sim \text{softmax}(\beta^\top \theta_d)$, where the $K$-dimensional vector $\theta_d \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_K)$ is a per-document variable, and $\beta$ is a real-valued matrix of size $K \times V$. The NVDM uses a per-document real-valued latent vector $\theta_d$ to average over the embedding matrix $\beta$ in the logit space. Like the ETM, the NVDM uses amortized variational inference to jointly learn the approximate posterior over the document representation $\theta_d$ and the model parameter $\beta$.

NVDM is not interpretable as a topic model; its latent variables are unconstrained. We study a more interpretable variant of the NVDM which constrains $\theta_d$ to lie in the simplex, replacing its Gaussian prior with a logistic normal (Aitchison & Shen, 1980). (This can be thought of as a semi-nonnegative matrix factorization.) We call this document model $\Delta$-NVDM.

We also consider PRODLDA (Srivastava & Sutton, 2017). It posits the likelihood $w_{dn} \sim \text{softmax}(\beta^\top \theta_d)$ where the topic proportions $\theta_d$ are from the simplex. Contrary to LDA, the topic-matrix $\beta$ is unconstrained. PRODLDA is fit using amortized variational inference with batch normalization (Ioffe

& Szegedy, 2015) and dropout (Srivastava et al., 2014).

Finally, we consider a document model that combines PRODLDA with pre-fitted word embeddings. We call this document model Labelled PRODLDA.

We study two variants of the ETM, one where the word embeddings are pre-fitted and one where they are learned jointly with the rest of the parameters. The variant with pre-fitted embeddings is called the "labelled ETM." We use skip-gram embeddings (Mikolov et al., 2013b).

**Algorithm settings.** Given a corpus, each model comes with an approximate posterior inference problem. We use variational inference for all of the models and employ stochastic variational inference (SVI) (Hoffman et al., 2013) to speed up the optimization. The minibatch size is 1,000 documents. For LDA, we set the learning rate as suggested by Hoffman et al. (2013): the delay is 10 and the forgetting factor is 0.85.

Within SVI, LDA enjoys coordinate ascent variational updates, with 5 inner steps to optimize the local variables. For the other models, we use amortized inference over the local variables $\theta_d$. We use 3-layer inference networks and we set the local learning rate to 0.002. We use $\ell_2$ regularization on the variational parameters (the weight decay parameter is $1.2 \times 10^{-6}$).

**Qualitative results.** We first examine the embeddings. The ETM, NVDM, $\Delta$-NVDM, and PRODLDA all involve a word embedding. We illustrate them by fixing a set of terms and calculating the words that occur in the neighborhood around them. For comparison, we also illustrate word embeddings learned by the skip-gram model.

Table 2.7 illustrates the embeddings of the different models. All the methods provide interpretable embeddings—words with related meanings are close to each other. The ETM, the NVDM, and PRODLDA learn embeddings that are similar to those from the skip-gram. The embeddings of $\Delta$-NVDM are different; the simplex constraint on the local variable changes the nature of the embeddings.

We next look at the learned topics. Table 2.8 displays the 7 most used topics for all methods,

as given by the average of the topic proportions $\theta_d$. LDA and the ETM both provide interpretable topics. The rest of the models do not provide interpretable topics; their model parameters $\beta$ are not interpretable as distributions over the vocabulary that mix to form documents.

**Quantitative results.** We next study the models quantitatively. We measure the quality of the topics and the predictive performance of the model. We found that among models with interpretable topics, the ETM provides the best predictions.

We measure topic quality by blending two metrics: topic coherence and topic diversity. Topic coherence is a quantitative measure of the interpretability of a topic (Mimno et al., 2011). It is the average pointwise mutual information of two words drawn randomly from the same document (Lau et al., 2014),

$$\text{TC} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{45}\sum_{i=1}^{10}\sum_{j=i+1}^{10} f(w_i^{(k)}, w_j^{(k)}),$$

where $\{w_1^{(k)}, \ldots, w_{10}^{(k)}\}$ denotes the top-10 most likely words in topic $k$. Here, $f(\cdot, \cdot)$ is the normalized pointwise mutual information,

$$f(w_i, w_j) = \frac{\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)}.$$

The quantity $P(w_i, w_j)$ is the probability of words $w_i$ and $w_j$ co-occurring in a document and $P(w_i)$ is the marginal probability of word $w_i$. We approximate these probabilities with empirical counts.

The idea behind topic coherence is that a coherent topic will display words that tend to occur in the same documents. In other words, the most likely words in a coherent topic should have high mutual information. Document models with higher topic coherence are more interpretable topic models.

We combine coherence with a second metric, topic diversity. We define topic diversity to be the percentage of unique words in the top 25 words of all topics. Diversity close to 0 indicates redundant

topics; diversity close to 1 indicates more varied topics.

We define the overall metric for the quality of a model's topics as the exponentiated product of its topic diversity and topic coherence.

A good topic model also provides a good distribution of language. To measure predictive quality, we calculate log likelihood on a document completion task (Rosen-Zvi et al., 2004; Wallach et al., 2009). We divide each test document into two sets of words. The first half is observed: it induces a distribution over topics which, in turn, induces a distribution over the next words in the document. We then evaluate the second half under this distribution. A good document model should provide higher log-likelihood on the second half. (For all methods, we approximate the likelihood by setting $\theta_d$ to the variational mean.)

We study both corpora and with different vocabularies. Figure 2.8 and Figure 2.9 show interpretability of the topics as a function of predictive power. (To ease visualization, we normalize both metrics by subtracting the mean and dividing by the standard deviation.) The best models are on the upper right corner.

LDA predicts worst in almost all settings. On the *20NewsGroups*, the NVDM's predictions are in general better than LDA but worse than for the other methods; on the *New York Times*, the NVDM gives the best predictions. However, topic quality for the NVDM is far below the other methods. (It does not provide "topics", so we assess the interpretability of its $\beta$ matrix.) In prediction, both versions of the ETM are at least as good as the simplex-constrained $\Delta$-NVDM. More importantly, both versions of the ETM outperform the Labelled PRODLDA; signaling the ETM provides a better way of integrating word embeddings into a topic model.

These figures show that, of the interpretable models, the ETM provides the best predictive performance while keeping interpretable topics. It is robust to large vocabularies.

**Stop words**

We now study a version of the *New York Times* corpus that includes all stop words. We remove


*Figure 2.10.* A topic containing stop words found by the ETM on *The New York Times*. The ETM is robust even in the presence of stop words.

infrequent words to form a vocabulary of size 10,283. Our goal is to show that the labeled ETM provides interpretable topics even in the presence of stop words, another regime where topic models typically fail. In particular, given that stop words appear in many documents, traditional topic models learn topics that contain stop words, regardless of the actual semantics of the topic. This leads to poor topic interpretability.

We fit LDA, the $\Delta$-NVDM, the labelled PRODLDA, and the labelled ETM with $K = 300$ topics. (We do not report the NVDM because it does not provide interpretable topics.) Table 2.9 shows the logarithm of the topic quality (the product of topic coherence and topic diversity). Overall, the labelled ETM gives the best performance in terms of topic quality.

While the ETM has a few "stop topics" that are specific for stop words (see, e.g., Figure 2.10), $\Delta$-NVDM and LDA have stop words in almost every topic. (The topics are not displayed here for space constraints.) The reason is that stop words co-occur in the same documents as every other word; therefore traditional topic models have difficulties telling apart content words and stop words. The labelled ETM recognizes the location of stop words in the embedding space; its sets them off on their own topic.

*Table 2.9.* Topic quality on the *New York Times* data in the presence of stop words. Topic quality here is given by the product of topic coherence and topic diversity (higher is better). The labeled ETM is robust to stop words; it achieves similar topic coherence than when there are no stop words.

| | TC | TD | Quality |
|---|---|---|---|
| LDA | 0.13 | 0.14 | 0.0182 |
| $\Delta$-NVDM | 0.17 | 0.11 | 0.0187 |
| Labelled PRODLDA | 0.03 | 0.53 | 0.0159 |
| Labeled ETM | **0.18** | **0.22** | **0.0396** |

### 2.4.5 Conclusion

We developed the ETM, a generative model of documents that marries LDA with word embeddings. The ETM assumes that topics and words live in the same embedding space, and that words are generated from a categorical distribution whose natural parameter is the inner product of the word embeddings and the embedding of the assigned topic.

The ETM learns interpretable word embeddings and topics, even in corpora with large vocabularies. We studied the performance of the ETM against several document models. The ETM learns both coherent patterns of language and an accurate distribution of words.

The construct used to define the ETM can be used to extend all versions of LDA, e.g. dynamic LDA (Blei & Lafferty, 2006), supervised LDA (Mcauliffe & Blei, 2008), and correlated LDA (Lafferty & Blei, 2005). In the next section we will apply the ETM technique for flexible dynamic topic modeling.

## 2.5 Dynamic Embedded Topic Modeling

Here we develop the *deep embedded topic model* (DETM), a model that combines the advantages of dynamic latent Dirichlet allocation (D-LDA) and the ETM. Like D-LDA, it allows the topics to vary smoothly over time to accommodate datasets that span a large period of time. Like the ETM, the DETM uses word embeddings, allowing it to generalize better than D-LDA and improving its topics.

We describe the model in Section 2.5.1 and then we develop an efficient structured variational inference algorithm in Section 2.5.2.

### 2.5.1 Model Description

The DETM is a dynamic topic model that uses embedding representations of words and topics. For each term $v$, it considers an $L$-dimensional embedding representation $\rho_v$. The DETM posits an embedding $\alpha_k^{(t)} \in \mathbb{R}^L$ for each topic $k$ at a given time stamp $t = 1, \ldots, T$. That is, the DETM represents each topic as a time-varying real-valued vector, unlike traditional topic models (where topics are distributions over the vocabulary). We refer to $\alpha_k^{(t)}$ as *topic embedding* (Dieng et al., 2019c); it is a distributed representation of the $k^{\text{th}}$ topic in the semantic space of words.

The DETM forms distributions over the vocabulary using the word and topic embeddings. Specifically, under the DETM, the probability of a word under a topic is given by the (normalized) exponentiated inner product between the embedding representation of the word and the topic's embedding at the corresponding time step,

$$p(w_{dn} = v \mid z_{dn} = k, \alpha_k^{(t_d)}) \propto \exp\{\rho_v^\top \alpha_k^{(t_d)}\}. \tag{2.15}$$

The probability of a particular term is higher when the term's embedding and the topic's embeddings are in agreement. Therefore, semantically similar words will be assigned to similar topics, since their representations are close in the embedding space.

The DETM enforces smooth variations of the topics by using a Markov chain over the topic embeddings $\alpha_k^{(t)}$. The topic representations evolve under Gaussian noise with variance $\gamma^2$,

$$p(\alpha_k^{(t)} \mid \alpha_k^{(t-1)}) = \mathcal{N}(\alpha_k^{(t-1)}, \gamma^2 I). \tag{2.16}$$

Similarly to D-LDA, the DETM considers time-varying priors over the topic proportions $\theta_d$. In addition to time-varying topics, this construction allows the model to capture how the general topic

usage evolves over time. The prior over $\theta_d$ depends on a latent variable $\eta_{t_d}$ (recall that $t_d$ is the time stamp of document $d$),

$$p(\theta_d \mid \eta_{t_d}) = \mathcal{LN}(\eta_{t_d}, a^2 I) \text{ where } p(\eta_t \mid \eta_{t-1}) = \mathcal{N}(\eta_{t-1}, \delta^2 I).$$

Figure 2.11 depicts the graphical model for the DETM. The full generative process is as follows:

1. Draw initial topic embedding $\alpha_k^{(1)} \sim \mathcal{N}(0, I)$.
2. Draw initial topic proportion mean $\eta_1 \sim \mathcal{N}(0, I)$.
3. For time step $t = 2, \ldots, T$:
   (a) Draw topic embeddings $\alpha_k^{(t)} \sim \mathcal{N}(\alpha_k^{(t-1)}, \gamma^2 I)$ for $k = 1, \ldots, K$.
   (b) Draw topic proportion means $\eta_t \sim \mathcal{N}(\eta_{t-1}, \delta^2 I)$.
4. For each document $d$:
   (a) Draw topic proportions $\theta_d \sim \mathcal{LN}(\eta_{t_d}, a^2 I)$.
   (b) For each word $n$ in the document:
      i. Draw topic assignment $z_{dn} \sim \text{Cat}(\theta_d)$.
      ii. Draw word $w_{dn} \sim \text{Cat}(\text{softmax}(\rho^\top \alpha_{z_{dn}}^{(t_d)}))$.

Steps 1 and 3a give the prior over the topic embeddings; they encourage smoothness on the resulting topics. Steps 2 and 3b are shared with D-LDA; they describe the evolution of the prior mean over the topic proportions. Steps 4a and 4b-i are standard for topic modeling; they represent documents as distributions over topics and draw a topic assignment for each word. Step 4b-ii is different—it uses the $L \times V$ word embedding matrix $\rho$ and the assigned topic embedding $\alpha_{z_{dn}}^{(t_d)}$ at time instant $t_d$ to form a categorical distribution over the vocabulary.

Since the DETM uses embedding representations of the words, it learns the topics in a particular embedding space. This aspect of the model is useful when the embedding of a new word is available, i.e., a word that does not appear in the corpus. Specifically, consider a term $v^\star$ that was not seen in the corpus. The DETM can assign it to topics by computing the inner products $\rho_{v^\star}^\top \alpha_k^{(t)}$, thus leveraging the semantic information of the word's embedding.

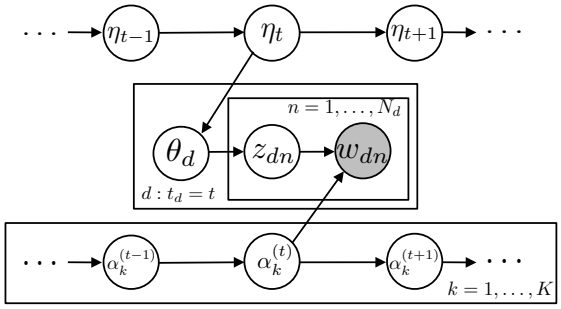



*Figure 2.11.* Graphical representation of DETM. The topic embeddings $\alpha_k^{(t)}$ and the latent means $\eta_t$ evolve over time. For each document at time step $t$, the prior over the topic proportions $\theta_d$ depends on $\eta_t$. The variables $z_{dn}$ denote the topic assignment; the variables $w_{dn}$ denote the words.

### 2.5.2 Structured Amortized Variational Inference

We observe a dataset $\mathcal{D}$ of documents $\{\mathbf{w}_1, \ldots, \mathbf{w}_D\}$ and their time stamps $\{t_1, \ldots, t_D\}$. Fitting a DETM involves finding the posterior distribution over the model's latent variables, $p(\theta, \eta, \alpha \mid \mathcal{D})$, where we have marginalized out the topic assignments $z$ from Eq. 2.15 for convenience,[3]

$$p(w_{dn} \mid \alpha_{1:K}^{(t_d)}) = \sum_{k=1}^{K} p(w_{dn} \mid z_{dn} = k, \alpha_k^{(t_d)}). \tag{2.17}$$

The posterior is intractable. We approximate it with variational inference (Jordan et al., 1999; Blei et al., 2017a).

Variational inference approximates the posterior using a family of distributions $q_\nu(\theta, \eta, \alpha)$. The parameters $\nu$ that index this family are called variational parameters, and are optimized to minimize the KL divergence between the approximation and the posterior. Solving this optimization problem is equivalent to maximizing the ELBO,

$$\mathcal{L}(\nu) = \mathbb{E}_q \left[\log p(\mathcal{D}, \theta, \eta, \alpha) - \log q_\nu(\theta, \eta, \alpha)\right]. \tag{2.18}$$

[3] Marginalizing $z_{dn}$ reduces the number of variational parameters and avoids discrete latent variables in the inference procedure, which is useful to form reparameterization gradients.

The model's log-joint distribution in Eq. 4.1 is

$$\log p(\mathcal{D}, \theta, \eta, \alpha) = \sum_{k=1}^{K} \sum_{t=1}^{T} \log p(\alpha_k^{(t)} \mid \alpha_k^{(t-1)}) + \sum_{t=1}^{T} \log p(\eta_t \mid \eta_{t-1}) + \sum_{d=1}^{D} \log p(\theta_d \mid \eta_{t_d}) + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log \left( \sum_{k=1}^{K} \theta_{dk} \beta_{k,w_{dn}}^{(t_d)} \right), \tag{2.19}$$

where $\beta_{k,w_{dn}}^{(t_d)} \triangleq \text{softmax}(\rho^\top \alpha_k^{(t_d)})|_{w_{dn}}$, $w_{dn}$ denotes the $n^{\text{th}}$ word in the $d^{\text{th}}$ document, and $N_d$ is the total number of words of the $d^{\text{th}}$ document.

To reduce the number of variational parameters and speed-up the inference algorithm, we use an amortized variational distribution, i.e., we let the parameters of the approximating distributions be functions of the data (Gershman & Goodman, 2014; Kingma & Welling, 2014). Additionally, we use a structured variational family to preserve some of the conditional dependencies of the graphical model (Saul & Jordan, 1996). The specific variational family in the DETM takes the form

$$q_\nu(\theta, \eta, \alpha) = \prod_d q(\theta_d \mid \eta_{t_d}, \mathbf{w}_d) \times \prod_t q(\eta_t \mid \eta_{1:t-1}, \widetilde{\mathbf{w}}_t) \times \prod_k \prod_t q(\alpha_k^{(t)} \mid \alpha_k^{(1:t-1)}, \widetilde{\mathbf{w}}_t). \tag{2.20}$$

(To avoid clutter, we suppress the notation for the variational parameters.)

The distribution over the topic proportions $q(\theta_d \mid \eta_{t_d}, \mathbf{w}_d)$ is a logistic-normal whose mean and covariance parameters are functions of both the latent mean $\eta_{t_d}$ and the bag-of-words representation of document $d$. In particular, these functions are parameterized by feed-forward neural networks that input both $\eta_{t_d}$ and the normalized bag-of-words representation. The distribution over the latent means $q(\eta_t \mid \eta_{1:t-1}, \widetilde{\mathbf{w}}_t)$ depends on all previous latent means $\eta_{1:t-1}$. We use an LSTM to capture this temporal dependency. We choose a Gaussian distribution $q(\eta_t \mid \eta_{1:t-1}, \widetilde{\mathbf{w}}_t)$ whose mean and covariance are given by the output of the LSTM. The input to the LSTM at time $t$ is formed by the concatenation of $\eta_{t-1}$ and the average of the bag-of-words representation of all documents whose time stamp is $t$. Here, $\widetilde{\mathbf{w}}_t$ denotes the normalized bag-of-words representation of all such documents. Finally, the distribution over the topic embeddings $q(\alpha_k^{(t)} \mid \alpha_k^{(1:t-1)}, \widetilde{\mathbf{w}}_t)$ is built analogously, using

**Algorithm 2:** Flexible dynamic topic modeling with the DETM

**Input:** Documents $\{\mathbf{w}_1, \ldots, \mathbf{w}_D\}$ and their time stamps $\{t_1, \ldots, t_D\}$
Initialize all variational parameters
**for** iteration $1, 2, 3, \ldots$ **do**
  Sample the latent means and the topic embeddings,
    $\eta \sim q(\eta \mid \widetilde{\mathbf{w}})$ and $\alpha \sim q(\alpha \mid \widetilde{\mathbf{w}})$
  Compute the topics $\beta_k^{(t)} = \text{softmax}(\rho^\top \alpha_k^{(t)})$ for
    $k = 1, \ldots, K$ and $t = 1, \ldots, T$
  Obtain a minibatch of documents
  **for** each document $d$ in the minibatch **do**
    Sample the topic proportions $\theta_d \sim q(\theta_d \mid \eta_{t_d}, \mathbf{w}_d)$
    **for** each word $n$ in the document **do**
      Compute $p(w_{dn} \mid \theta_d) = \sum_k \theta_{dk} \beta_{k,w_{dn}}^{(t_d)}$
    **end for**
  **end for**
  Estimate the ELBO in Eq. 2.21 and its gradient w.r.t.
    the variational parameters (backpropagation)
  Update the model and variational parameters (Adam)
**end for**

an LSTM to capture the temporal dependencies.

We optimize the ELBO with respect to the variational parameters. Because the expectations in Eq. 4.1 are intractable, we use black box variational inference, obtaining unbiased gradient estimators with a Monte Carlo method. In particular, we use one sample from the variational distribution to form reparameterization gradients (Kingma & Welling, 2014; Titsias & Lázaro-Gredilla, 2014; Rezende et al., 2014).

To sample from $q_\nu(\theta, \eta, \alpha)$ using reparameterization, we first sample a set of standard Gaussian auxiliary latent variables $\varepsilon \sim \mathcal{N}(0, I)$ and we then use a deterministic transformation $h_\nu(\varepsilon)$ that gives the samples $(\theta, \eta, \alpha)$. Therefore, the realized values of the latent variables are now functions of the variational parameters $\nu$, since $(\theta, \eta, \alpha) = h_\nu(\varepsilon)$. Given these samples, we estimate the ELBO

in Eq. 4.1 as

$$\mathcal{L}(\nu) \approx \sum_{d=1}^{D}\sum_{n=1}^{N_d} \log\left(\sum_{k=1}^{K} \theta_{dk}\beta_{k,w_{dn}}^{(t_d)}\right) - \sum_{k=1}^{K}\sum_{t=1}^{T} \text{KL}\left(q(\alpha_k^{(t)} \mid \alpha_k^{(1:t-1)}, \widetilde{\mathbf{w}}_t) \mid\mid p(\alpha_k^{(t)} \mid \alpha_k^{(t-1)})\right)$$
$$- \sum_{t=1}^{T} \text{KL}\left(q(\eta_t \mid \eta_{1:t-1}, \widetilde{\mathbf{w}}_t) \mid\mid p(\eta_t \mid \eta_{t-1})\right) - \sum_{d=1}^{D} \text{KL}\left(q(\theta_d \mid \eta_{t_d}, \mathbf{w}_d) \mid\mid p(\theta_d \mid \eta_{t_d})\right). \quad (2.21)$$

Here, each KL divergence corresponds to the KL between two Gaussian distributions whose parameters are functions of the latent variables in the conditioning set. Therefore, the KL terms can be obtained in closed form as a function of these latent variables.

The variational optimization problem reduces to a stochastic optimization method that approximates the gradients $\nabla_\nu \mathcal{L}(\nu)$ by differentiating through Eq. 2.21 w.r.t. $\nu$. To speed up the algorithm, we estimate the sum over documents by taking a minibatch of documents at each iteration; this allows to handle large collections of documents (Hoffman et al., 2013). We set the learning rate with Adam (Kingma & Ba, 2015). Algorithm 2 summarizes the procedure.

### 2.5.3 Related Work

The DETM builds on word embeddings, topic models, and dynamic topic models.

Word embeddings are low-dimensional continuous representations of words that capture their semantics (Rumelhart & Abrahamson, 1973; Bengio et al., 2003, 2006; Mikolov et al., 2013a,b; Pennington et al., 2014; Levy & Goldberg, 2014). Some recent work finds embedding representations that vary over time (Bamler & Mandt, 2017; Rudolph & Blei, 2018). Despite incorporating a time-varying component, these works have a different goal than the DETM. Rather than modeling the temporal evolution of documents, they model how the meaning of words shifts over time. (In future research, the DETM developed here could be used in concert with these methods.)

There has been a surge of methods that combine word embeddings and probabilistic topic models. Some methods modify the prior distributions over topics in LDA (Petterson et al., 2010; Xie et al.,

2015; Shi et al., 2017; Zhao et al., 2017a,b). These methods use word embeddings as a type of "side information." There are also methods that combine LDA with word embeddings by first converting the discrete text into continuous observations of embeddings (Das et al., 2015; Xun et al., 2016; Batmanghelich et al., 2016; Xun et al., 2017). These works adapt LDA for real-valued observations, for example using a Gaussian likelihood. Still other ways of combining LDA and word embeddings modify the likelihood (Nguyen et al., 2015), randomly replace words drawn from a topic with the embeddings drawn from a Gaussian (Bunk & Krestel, 2018), or use Wasserstein distances to learn topics and embeddings jointly (Xu et al., 2018). In contrast to all these methods, the DETM uses sequential priors and is a probabilistic model of discrete data that directly models the words.

Another line of research improves topic modeling inference through deep neural networks; these are called neural topic models (Miao et al., 2016; Srivastava & Sutton, 2017; Card et al., 2017; Cong et al., 2017; Zhang et al., 2018). Most of these works are based on the variational autoencoder (Kingma & Welling, 2014) and use amortized inference (Gershman & Goodman, 2014). Finally, the ETM (Dieng et al., 2019c) is a probabilistic topic model that also makes use of word embeddings and uses amortization in its inference procedure.

The first and most common dynamic topic model is D-LDA (Blei & Lafferty, 2006). Bhadury et al. (2016) scale up the inference method of D-LDA using a sampling procedure. Other extensions of D-LDA use stochastic processes to introduce stronger correlations in the topic dynamics (Wang & McCallum, 2006; Wang et al., 2008; Jähnichen et al., 2018). The DETM is also an extension of D-LDA, but developed for a different purpose. The DETM better fits the distribution of words via the use of distributed representations for both the words and the topics.

### 2.5.4 Empirical Study

We use the DETM to analyze the transcriptions of the United Nations (UN) general debates from 1970 to 2015, a corpus of ACL abstracts from 1973 to 2006, and a set of articles from Science Magazine from 1990 to 1999. We found the DETM provides better predictive power and higher

*Table 2.10.* Summary statistics of the UN, SCIENCE, and ACL datasets.

| Dataset | # Docs Train | # Docs Val | # Docs Test | # Timestamps | Vocabulary |
|---|---|---|---|---|---|
| UN | 196,290 | 11,563 | 23,097 | 46 | 12,466 |
| SCIENCE | 13,894 | 819 | 1,634 | 10 | 25,987 |
| ACL | 8,936 | 527 | 1,051 | 31 | 35,108 |

*Table 2.11.* Performance as measured by perplexity (PPL), topic coherence (TC), topic diversity (TD), topic quality (TQ), and runtime (in minutes per epoch) on the UN dataset. The DETM achieves better predictive and qualitative performance than D-LDA and D-LDA-REP and runs significantly faster than D-LDA.

| METHOD | PPL | TC | TD | TQ | RUNTIME |
|---|---|---|---|---|---|
| D-LDA (Blei & Lafferty, 2006) | 2393.5 | **0.1317** | 0.6065 | 0.0799 | 28.70 |
| D-LDA-REP | 2931.3 | 0.1180 | 0.2691 | 0.0318 | 6.00 |
| DETM | **1970.7** | 0.1206 | **0.6703** | **0.0809** | 3.70 |

topic quality in general on these datasets when compared to D-LDA.

On the transcriptions of the UN general debates, we additionally carried out a qualitative analysis of the results. We found that the DETM reveals the temporal evolution of the topics discussed in the debates (such as climate change, war, poverty, or human rights).

We compared the DETM against two versions of D-LDA, labeled as D-LDA and D-LDA-REP, which differ only in the inference method (the details are below). The comparison of the DETM against D-LDA-REP reveals that the key to the DETM's performance is the model and not simply the scalable inference procedure.

**Datasets.** We study the DETM on three datasets. The UN debates corpus[4] spans 46 years (Baturo et al., 2017). Each year, leaders and other senior officials deliver statements that present their government's perspective on the major issues in world politics. The corpus contains the transcriptions of each country's statement at the UN General Assembly. We follow Lefebure (2018) and split the speeches into paragraphs, treating each paragraph as a separate document.

The second dataset contains ten years of SCIENCE articles, from 1990 to 1999. The articles are from

[4]See https://www.kaggle.com/unitednations/un-general-debates.

*Table 2.12.* Performance as measured by perplexity (PPL), topic coherence (TC), topic diversity (TD), topic quality (TQ), and runtime (in minutes per epoch) on the SCIENCE dataset. The DETM achieves better predictive and qualitative performance than D-LDA and D-LDA-REP and runs significantly faster than D-LDA.

| METHOD | PPL | TC | TD | TQ | RUNTIME |
|---|---|---|---|---|---|
| D-LDA (Blei & Lafferty, 2006) | **3600.7** | **0.2392** | 0.6502 | 0.1556 | 15.03 |
| D-LDA-REP | 8377.4 | 0.0611 | 0.2290 | 0.0140 | 0.36 |
| DETM | 4206.1 | 0.2298 | **0.8215** | **0.1888** | 0.47 |

*Table 2.13.* Performance as measured by perplexity (PPL), topic coherence (TC), topic diversity (TD), topic quality (TQ), and runtime (in minutes per epoch) on the ACL dataset. The DETM achieves better predictive and qualitative performance than D-LDA and D-LDA-REP and runs significantly faster than D-LDA.

| METHOD | PPL | TC | TD | TQ | RUNTIME |
|---|---|---|---|---|---|
| D-LDA (Blei & Lafferty, 2006) | 4324.2 | 0.1429 | 0.5904 | 0.0844 | 26.30 |
| D-LDA-REP | 5836.7 | 0.1011 | 0.2589 | 0.0262 | 1.60 |
| DETM | **4120.6** | **0.1630** | **0.8286** | **0.1351** | 0.75 |

JSTOR, an on-line archive of scholarly journals that scans bound volumes and runs optical character recognition algorithms on the scans. This data was used by Blei & Lafferty (2007).

The third dataset is a collection of articles from 1973 to 2006 from the ACL Anthology (Bird et al., 2008). This anthology is a repository of computational linguistics and natural language processing papers.

For each dataset, we apply standard preprocessing techniques, such as tokenization and removal of numbers and punctuation marks. We also filter out stop words, i.e., words with document frequency above 70%, as well as standard stop words from a list. Additionally, we remove low-frequency words, i.e., words that appear in less than a certain number of documents (30 documents for UN debates, 100 for the SCIENCE corpus, and 10 for the ACL dataset). We use 85% randomly chosen documents for training, 10% for testing, and 5% for validation, and we remove one-word documents from the validation and test sets. Table 2.10 summarizes the characteristics of each dataset.

**Methods.** We compare the DETM against two variants of D-LDA. One variant is the original

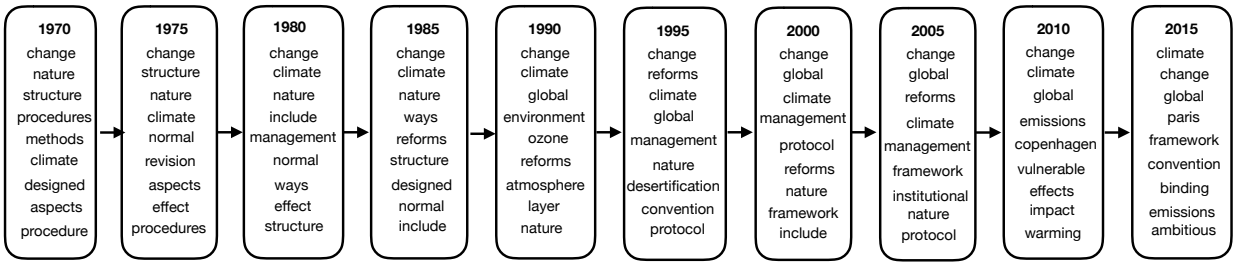



*Figure 2.12.* Temporal evolution of the top-10 words from a topic about climate change learned by the DETM. This topic is in agreement with historical events. In the 1990s the destruction of the ozone layer was of major concern. More recently the concern is about global warming. Events such as the Kyoto protocol and the Paris convention are also reflected in this topic's evolution.

model and algorithm of Blei & Lafferty (2006). The other variant, which we call D-LDA-REP, is the D-LDA model fitted using mean-field variational inference with the reparameterization trick. The comparison against D-LDA-REP helps us delineate between performance due to the model and performance due to the inference algorithm.

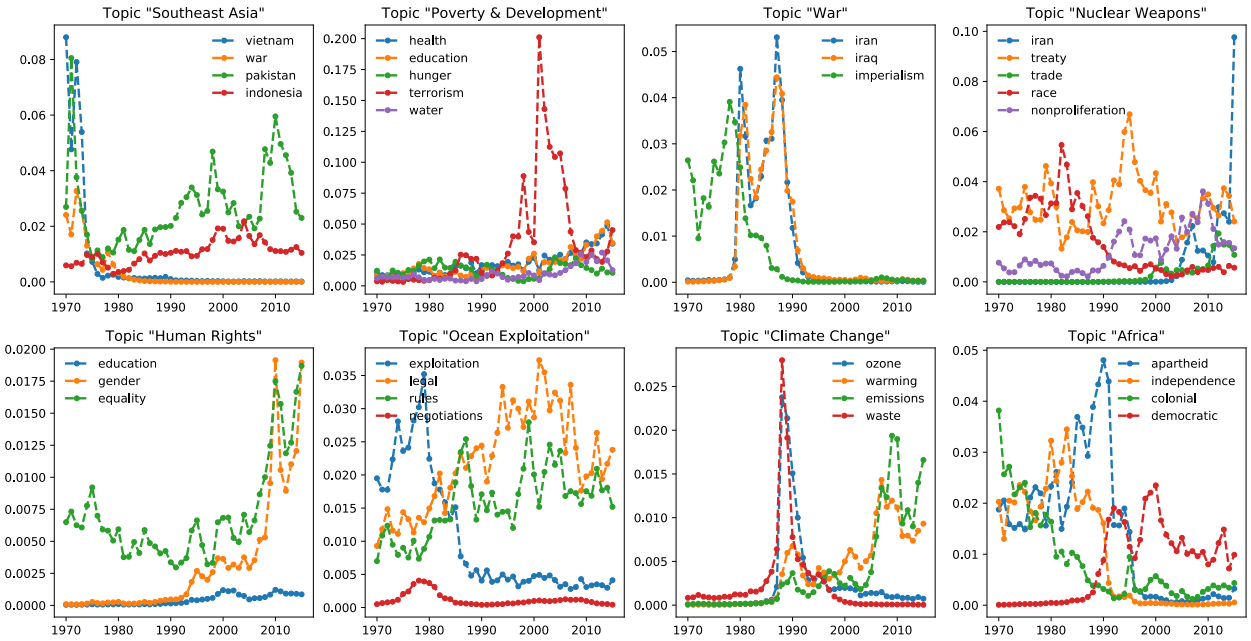



*Figure 2.13.* Evolution of word probability across time for eight different topics learned by the DETM. For each topic, we choose a set of words whose probability shift aligns with historical events (these are not the words with the highest probability in each topic). For example, one interesting finding is the increased relevance of the words "gender" and "equality" in a topic about human rights.

**Settings.** We use 50 topics for all the experiments and follow Blei & Lafferty (2006) to set the variances of the different priors as $\delta^2 = \sigma^2 = \gamma^2 = 0.005$ and $a^2 = 1$.

For the DETM, we first fit 300-dimensional word embeddings using skip-gram (Mikolov et al., 2013b).[5] We apply the algorithm in Section 2.5.2 using a batch size of 200 documents for all datasets except for ACL, for which we used 100. To parameterize the variational distribution, we use a fully connected feed-forward inference network for the topic proportions $\theta_d$. The network has ReLU activations and 2 layers of 800 hidden units each. We set the mean and log-variance for $\theta_d$ as linear maps of the output. We applied a small dropout rate of 0.1 to the output of this network before using it to compute the mean and the log-variance. For the latent means $\eta_{1:T}$, each bag-of-word representation $\widetilde{\mathbf{w}}_t$ is first linearly mapped to a low-dimensional space of dimensionality 400. This conforms the input of an LSTM that has 4 layers of 400 hidden units each. The LSTM output is then concatenated with the previous latent mean $\eta_{t-1}$, and the result is linearly mapped to a $K$-dimensional space to get the mean and log-variance for $\eta_t$. We apply a weight decay of $1.2 \cdot 10^{-6}$ on all network parameters. We run Algorithm 1 for a maximum of 1000 epochs on SCIENCE and ACL and for 400 epochs on the UN dataset; the stopping criterion is based on the held-out log-likelihood on the validation set. The learning rate is set to 0.001 for the UN and SCIENCE datasets and to 0.0008 on the ACL corpus. We fixed the learning rate throughout training. We clip the norm of the gradients of the ELBO to 2.0 to stabilize training.

We fit D-LDA using the published code of Blei & Lafferty (2006).[6] To fit D-LDA, Blei & Lafferty (2006) derived a bound of the ELBO to enable a coordinate-ascent inference algorithm that also uses Kalman filtering and smoothing as a subroutine. Besides loosening the variational bound on the log-marginal likelihood of the data, this algorithm presents scalability issues both in terms of the number of topics and in terms of the vocabulary size. (See Table 2.13 for a comparison of the runtime across methods.) To fit D-LDA, we follow Blei & Lafferty (2006) and initialize the algorithm with LDA. In particular, we run 25 epochs of LDA followed by 100 epochs of D-LDA.

[5] More advanced methods can be used to learn word embeddings. We used skip-gram for simplicity and found it leads to good performance.

[6] See https://github.com/blei-lab/dtm.

We also fit D-LDA-REP to overcome the scalability issues of D-LDA by leveraging recent advances in variational inference. We use stochastic optimization based on reparameterization gradients and we draw batches of 1,000 documents at each iteration. We collapse the discrete latent topic indicators $z_{dn}$ to enable the reparameterization gradients, and we use a fully factorized Gaussian approximation for the rest of the latent variables, except for $\eta_{1:T}$, for which we use a full-covariance Gaussian for each of its dimensions. We run 5 epochs of LDA to initialize D-LDA-REP and then run 120 epochs of the D-LDA-REP inference algorithm. For D-LDA-REP, we use RMSProp (Tieleman & Hinton, 2012) to set the step size, setting the learning rate to 0.05 for the mean parameters and to 0.005 for the variance parameters.

**Quantitative results.** We compare the DETM, D-LDA, and D-LDA-REP according to two metrics: perplexity on a document completion task and topic quality. The perplexity is obtained by computing the probability of each word in the second half of a test document, conditioned on the first half (Rosen-Zvi et al., 2004; Wallach et al., 2009). To obtain the topic quality, we combine two metrics. The first metric is topic coherence; it provides a quantitative measure of the interpretability of a topic (Mimno et al., 2011). We obtain the topic coherence by taking the average pointwise mutual information of two words drawn randomly from the same document (Lau et al., 2014); this requires to approximate word probabilities with empirical counts. The second metric is topic diversity; it is the percentage of unique words in the top 25 words of all topics (Dieng et al., 2019c). Diversity close to 0 indicates redundant topics. We obtain both topic coherence and topic diversity by averaging over time. Finally, topic quality is defined as the product between topic coherence and diversity (Dieng et al., 2019c).

Table 2.13 shows that the DETM outperforms both D-LDA and D-LDA-REP according to both perplexity and topic quality on almost all datasets. In particular, the DETM finds more diverse and coherent topics. We posit this is due to its use of embeddings.

**Qualitative results.** The DETM finds that the topics' evolution over time are in agreement with historical events. As an example, Figure 2.12 shows the trajectory of a topic on climate change (a

topic that D-LDA-REP did not discover). In the 1990s, protecting the ozone layer was the primary concern; more recently the topic has shifted towards global warming and reducing the greenhouse gas emissions. Some events on climate change, such as the Kyoto protocol (1997) or the Paris convention (2016), are also reflected in the topic's evolution.

We now examine the evolution of the probability of individual words. Figure 2.13 shows these probabilities for a variety of words and topics. For example, the probability of the word "Vietnam" in a topic on Southeast Asia decays after the end of the war in 1975. In a topic about nuclear weapons, the concern about the arms "race" between the USA and the Soviet Union eventually decays, and "Iran" becomes more relevant in recent years. Similarly, words like "equality" and "gender" become more important in recent years within a topic about human rights. Note that the names of the topics are subjective; we assigned the names inspired by the top words in each topic (the words in Figure 2.13 are not necessarily the most likely words within each topic). One example is the topic on climate change, whose top words are shown in Figure 2.12. Another example is the topic on human rights, which exhibits the words "human" and "rights" consistently at the top across all time steps.

### 2.5.5 Conclusion

We developed the DETM, a probabilistic model of documents that combines word embeddings and dynamic latent Dirichlet allocation (D-LDA). The DETM models each word with a categorical distribution parameterized by the dot product between the embedding of the word and an embedding representation of its assigned topic. Each topic embedding is a time-varying vector in the embedding space of words. Using a random walk prior over these topic embeddings, the DETM uncovers smooth topic trajectories. We applied the DETM to analyze three different corpora and found that the DETM outperforms D-LDA both in terms of predictive performance and topic quality while requiring significantly less time to fit.

# Chapter 3: Learning via Reweighted Expectation Maximization

The models described in Chapter 2 were fitted using AVI. AVI scales learning by using recognition networks to define the variational family. It then maximizes the ELBO, a lower bound of the log marginal likelihood of the data. Because the ELBO is often intractable, AVI uses Monte Carlo to approximate it. Monte Carlo estimates of the ELBO are biased and lead to a loose bound of the log marginal likelihood. To address this, several other learning algorithms have been proposed that maximize a tighter lower bound than the ELBO (e.g. Bornschein & Bengio (2014); Burda et al. (2015b).)

In this chapter, we develop an algorithm for fitting DPGMs called *reweighted expectation maximization* (REM). REM optimizes an asymptotically unbiased approximation of the log marginal likelihood of the data. This procedure involves learning a proposal distribution over the latent variables. We propose to leverage moment matching to learn expressive proposals. Because REM optimizes a better approximation to the log marginal likelihood of the data, it generalizes better to unseen data than approaches such as the VAE.

## 3.1 Rethinking ELBO Maximization for Fitting DPGMs

For simplicity, we focus on the simplest DPGM. We consider a set of $N$ i.i.d datapoints $\mathbf{x}_1, \ldots, \mathbf{x}_N$. We posit each observation $\mathbf{x}_i$ is drawn by first sampling a latent variable $\mathbf{z}_i$ from some fixed prior $p(\mathbf{z})$ and then sampling $\mathbf{x}_i$ from $p_\theta(\mathbf{x}_i \mid \mathbf{z}_i)$—the conditional distribution of $\mathbf{x}_i$ given $\mathbf{z}_i$. We define the conditional $p_\theta(\mathbf{x}_i \mid \mathbf{z}_i)$ using a deep neural network with parameters $\theta$. Our goal is to learn the parameters $\theta$ and perform posterior inference over the latent variables.

One way to achieve this goal is to use VI and maximize the ELBO,

$$\text{ELBO} = \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\right]. \tag{3.1}$$

This is the approach of the VAE (Kingma & Welling, 2013; Rezende et al., 2014), which maximizes the ELBO with respect to both $\phi$ and $\theta$. To better understand what this maximization procedure corresponds to, consider the expression of the log marginal likelihood of the data in terms of the ELBO

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\right] + \text{KL}\left(q_\phi(\mathbf{z}) || p_\theta(\mathbf{z} \mid \mathbf{x})\right). \tag{3.2}$$

The log marginal likelihood $\log p_\theta(\mathbf{x})$ does not depend on the variational parameters $\phi$. Therefore performing posterior inference by minimizing the KL term $\text{KL}\left(q_\phi(\mathbf{z}) || p_\theta(\mathbf{z} \mid \mathbf{x})\right)$ is equivalent to maximizing the ELBO, for a fixed $\theta$. However $\log p_\theta(\mathbf{x})$ depends on the parameters $\theta$, which makes the KL minimization over a "moving target"—the true posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$ changes with $\theta$. As a result, there is possibility of running into a bad local optimum in which the KL is rendered small but the neural network parameterizing the model is useless.

In what follows, we first review expectation maximization (EM) and propose an algorithm that leverages EM to fit the model parameters $\theta$. Posterior inference can be done, once the model is fit, by minimizing KL using amortized VI.

### 3.2 Expectation Maximization

EM was first introduced in the statistics literature, where it was used to solve problems involving missing data (Dempster et al., 1977). One classic application of EM is to fit mixtures of Gaussians, where the cluster assignments are considered *unobserved* data (Murphy, 2012). Another use of EM is for probabilistic PCA (Tipping & Bishop, 1999). EM is a maximum likelihood iterative optimization technique that directly targets the log marginal likelihood and served as the departure

point for the development of variational inference methods.

The EM objective is the log marginal likelihood of the data in Eq. 3.2. EM alternates between an *E-step*, which sets the KL term in Eq. 3.2 to zero, and an *M-step*, which fits the model parameters $\theta$ by maximizing ELBO using the proposal learned in the E-step. Note that after the E-step, the objective in Eq. 3.2 says the log marginal is exactly equal to the ELBO which is a tractable objective for fitting the model parameters. EM alternates these two steps until convergence to an approximate maximum likelihood solution for $p_\theta(\mathbf{x})$.

Contrast this with VI. The true objective for VI is the KL term in Eq. 3.2, $\text{KL}\left(q_\phi(\mathbf{z})||p_\theta(\mathbf{z}\,|\,\mathbf{x})\right)$, which is intractable. The argument in VI is that minimizing this KL is equivalent to maximizing the ELBO, the first term in Eq. 3.2. This argument only holds when the log marginal likelihood $\log p_\theta(\mathbf{x})$ has no free parameters, in which case it is called the *model evidence*. Importantly, VI does not necessarily maximize $\log p_\theta(\mathbf{x})$ because it chooses approximate posteriors $q_\phi(\mathbf{z})$ that may be far from the exact conditional posterior.

In contrast EM effectively maximizes $\log p_\theta(\mathbf{x})$ after each iteration. Consider given $\theta_t$, the state of the model parameters after the $t^{th}$ iteration of EM. EM learns $\theta_{t+1}$ through two steps, which we briefly review:

$$\text{E-step: set } q_\phi(\mathbf{z}) = p_{\theta_t}(\mathbf{z}\,|\,\mathbf{x}) \tag{3.3}$$

$$\begin{aligned}\text{M-step: define } \theta_{t+1} &= \arg\max_\theta \mathcal{L}(\theta) \\ &= \arg\max_\theta \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z})\right] \\ &= \arg\max_\theta \mathbb{E}_{p_{\theta_t}(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log p_{\theta_t}(\mathbf{z}\,|\,\mathbf{x})\right] \\ &= \arg\max_\theta \mathbb{E}_{p_{\theta_t}(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z})\right] \end{aligned} \tag{3.4}$$

The value of the log marginal likelihood for $\theta_{t+1}$ is greater than for $\theta_t$. To see this, write

$$\begin{aligned}\log p_{\theta_t}(\mathbf{x}) &= \mathcal{L}(\theta_t) + \textsc{kl}\left(q_\phi(\mathbf{z}) || p_{\theta_t}(\mathbf{z} \mid \mathbf{x})\right) = \mathcal{L}(\theta_t) \\ &\leq \mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_{t+1}) + \textsc{kl}\left(q_\phi(\mathbf{z}) || p_{\theta_{t+1}}(\mathbf{z} \mid \mathbf{x})\right) \\ &= \log p_{\theta_{t+1}}(\mathbf{x})\end{aligned}$$

where the second equality is due to the E-step, the first inequality is due to the M-step, and the second inequality is due to the nonnegativity of KL.

We next propose an algorithm that leverages EM to fit the model parameters $\theta$.

### 3.3 Reweighted Expectation Maximization

We develop REM, an algorithm that leverages EM to fit the model parameters $\theta$. Assume given $\theta_t$ from the previous iteration of EM. We want to find the next settings of the parameters $\theta_{t+1}$ that maximize the objective in the M-step in Eq. 3.4,

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{p_{\theta_t}(\mathbf{z}_i \mid \mathbf{x}_i)} \left[\log p_\theta(\mathbf{x}_i, \mathbf{z}_i)\right] \tag{3.5}$$

$$= \sum_{i=1}^{N} \int \frac{p_{\theta_t}(\mathbf{z}_i, \mathbf{x}_i)}{p_{\theta_t}(\mathbf{x}_i)} \log p_\theta(\mathbf{x}_i, \mathbf{z}_i)\, d\mathbf{z}_i. \tag{3.6}$$

This objective is intractable because it involves the marginal $p_{\theta_t}(\mathbf{x}_i)$[1]. However we can make it tractable using self-normalized importance sampling (Owen, 2013),

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)} \left[\frac{\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t) \log p_\theta(\mathbf{x}_i, \mathbf{z}_i)}{\mathbb{E}_{r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)} \left(\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t)\right)}\right]. \tag{3.7}$$

[1] Although the marginal here does not depend on $\theta$, it cannot be ignored because it depends on the $i^{th}$ datapoint. Therefore it cannot be pulled outside the summation.

where $\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t) = \frac{p_{\theta_t}(\mathbf{z}_i, \mathbf{x}_i)}{r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)}$. Here $r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$ is a proposal distribution. Its parameters $\eta_t$ were fitted in the previous iteration (the $t^{th}$ iteration.) We now approximate the expectations in Eq. 3.7 using Monte Carlo by drawing $K$ samples $\mathbf{z}_i^{(1)}, \ldots, \mathbf{z}_i^{(K)}$ from the proposal,

$$\begin{aligned} \boldsymbol{\alpha}_{it}^k &= \frac{\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i^{(k)}; \theta_t, \eta_t)}{\sum_{k=1}^K \mathbf{w}(\mathbf{x}_i, \mathbf{z}_i^{(k)}; \theta_t, \eta_t)} \\ \mathcal{L}(\theta) &= \sum_{i=1}^N \sum_{k=1}^K \boldsymbol{\alpha}_{it}^k \cdot \log p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)}) \end{aligned} \tag{3.8}$$

Note the approximation in Eq. 3.8 is biased but asymptotically unbiased. More specifically, the approximation improves as the number of particles $K$ increases.

We use gradient-based learning which requires to compute the gradient of $\mathcal{L}(\theta)$ with respect to the model parameters $\theta$, this is

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{i=1}^N \sum_{k=1}^K \boldsymbol{\alpha}_{it}^k \cdot \nabla_\theta \log p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)}). \tag{3.9}$$

We now describe how to learn expressive proposals by leveraging moment matching.

### 3.3.1 Learning Expressive Proposals via Moment Matching

Denote by $\eta_t$ the proposal parameters at the previous iteration. We learn $\eta_{t+1}$ by targeting the true posterior $p_{\theta_t}(\mathbf{z} \mid \mathbf{x})$,

$$\eta_{t+1} = \arg\min_\eta \mathcal{L}_{\text{REM}}(\eta) = \text{KL}(p_{\theta_t}(\mathbf{z} \mid \mathbf{x}) || r_\eta(\mathbf{z} \mid \mathbf{x})). \tag{3.10}$$

Unlike the importance weighted auto-encoder (IWAE), the proposal here targets the true posterior using a well defined objective—the inclusive KL divergence. The inclusive KL induces overdispersed proposals which are beneficial in importance sampling (Minka et al., 2005).

**Algorithm 3:** Learning with reweighted expectation maximization (REM (v1))

**Input:** Data $\mathbf{x}$
Initialize model and proposal parameters $\theta, \eta$
**for** *iteration* $t = 1, 2, \ldots$ **do**
  Draw minibatch of observations $\{\mathbf{x}_n\}_{n=1}^B$
  **for** *observation* $n = 1, 2, \ldots, B$ **do**
    Draw $\mathbf{z}_n^{(1)}, \ldots, \mathbf{z}_n^{(K)} \sim r_{\eta_t}(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)$
    Compute importance weights $\mathbf{w}^{(k)} = \frac{p_{\theta_t}(\mathbf{z}_n^{(k)}, \mathbf{x}_n)}{r_{\eta_t}(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)}$
    Compute $\boldsymbol{\mu}_{nt}$ and $\boldsymbol{\Sigma}_{nt}$ using Eq. 3.16 and Eq. 3.17
    Set proposal $s(\mathbf{z}_n^{(t)}) = \mathcal{N}(\boldsymbol{\mu}_{nt}, \boldsymbol{\Sigma}_{nt})$
  **end for**
  Compute $\nabla_\eta \mathcal{L}(\eta)$ as:
  $\nabla_\eta \mathcal{L}(\eta) = \frac{1}{|B|} \sum_{n \in B} \sum_{k=1}^K \frac{\mathbf{v}^{(k)}}{\sum_{k=1}^K \mathbf{v}^{(k)}} \nabla_\eta \log r_\eta(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)$
  Update $\eta$ using Adam
  Compute $\nabla_\theta \mathcal{L}(\boldsymbol{\theta})$ as
  $\nabla_\theta \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|B|} \sum_{n \in B} \sum_{k=1}^K \frac{\mathbf{w}^{(k)}}{\sum_{k=1}^K \mathbf{w}^{(k)}} \nabla_\theta \log p_\theta(\mathbf{x}_n, \mathbf{z}_n^{(k)})$
  Update $\theta$ using Adam
**end for**

The objective in Eq. 3.28 is still intractable as it involves the true posterior $p_{\theta_t}(\mathbf{z} \mid \mathbf{x})$,

$$\mathcal{L}_{\text{REM}}(\eta) = -\sum_{i=1}^N \mathbb{E}_{p_{\theta_t}(\mathbf{z}_i \mid \mathbf{x}_i)} \left[ \log r_\eta(\mathbf{z}_i \mid \mathbf{x}_i) \right] + \text{const.}, \tag{3.11}$$

where const. is a constant with respect to $\eta$ that we can ignore. We use the same approach as for fitting the model parameters $\theta$. That is, we write

$$\mathcal{L}_{\text{REM}}(\eta) = -\sum_{i=1}^N \mathbb{E}_{s(\mathbf{z}_i)} \left[ \frac{\mathbf{v}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t) \log r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)}{\mathbb{E}_{s(\mathbf{z}_i)} \left( \mathbf{v}\left(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t\right) \right)} \right]. \tag{3.12}$$

where $\mathbf{v}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t) = \frac{p_{\theta_t}(\mathbf{z}_i, \mathbf{x}_i)}{s(\mathbf{z}_i)}$. Here $s(\mathbf{z}_i)$ is a hyperproposal that has no free parameters. (We will describe it shortly.) The hyperobjective in Eq. 3.12 is still intractable due to the expectations.

We approximate it using Monte Carlo by drawing $K$ samples $\mathbf{z}_i^{(1)}, \ldots, \mathbf{z}_i^{(K)}$ from $s(\mathbf{z}_i)$. Then

$$\begin{aligned}
\beta_{it}^k &= \frac{\mathbf{v}\left(\mathbf{x}_i, \mathbf{z}_i^{(k)}; \theta_t, \eta_t\right)}{\sum_{k'=1}^K \mathbf{v}\left(\mathbf{x}_i, \mathbf{z}_i^{(k')}; \theta_t, \eta_t\right)} \\
\mathcal{L}_{\text{REM}}(\eta) &= -\sum_{i=1}^N \sum_{k=1}^K \beta_{it}^k \cdot \log r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i),
\end{aligned} \tag{3.13}$$

We choose the proposal $s(\mathbf{z}_i)$ to be a full Gaussian whose parameters are found by matching the moments of the true posterior $p_{\theta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$. More specifically, $s(\mathbf{z}_i) = \mathcal{N}(\boldsymbol{\mu}_{it}, \Sigma_{it})$ where

$$\begin{aligned}
\boldsymbol{\mu}_{it} &= \mathbb{E}_{p_{\theta_t}(\mathbf{z}_i \mid \mathbf{x}_i)}[\mathbf{z}_i] \\
\Sigma_{it} &= \mathbb{E}_{p_{\theta_t}(\mathbf{z}_i \mid \mathbf{x}_i)}\left[\left(\mathbf{z}_i - \boldsymbol{\mu}_i^{(t)}\right)\left(\mathbf{z}_i - \boldsymbol{\mu}_i^{(t)}\right)^\top\right].
\end{aligned} \tag{3.14}$$

The expressions for the mean and covariance matrix are still intractable. We estimate them using self-normalized importance sampling, with proposal $r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$, and Monte Carlo. We first write

$$\boldsymbol{\mu}_{it} = \mathbb{E}_{r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)}\left(\frac{\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t)}{\mathbb{E}_{r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)}\left(\mathbf{w}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t)\right)}\mathbf{z}_i\right), \tag{3.15}$$

(the covariance $\boldsymbol{\Sigma}_{it}$ is analogous), and then estimate the expectations using Monte Carlo,

$$\begin{aligned}
\boldsymbol{\mu}_{it} &\approx \sum_{k=1}^K \boldsymbol{\alpha}_{it}^k \cdot \mathbf{z}_i^{(k)} \\
\boldsymbol{\Sigma}_{it} &\approx \sum_{k=1}^K \boldsymbol{\alpha}_{it}^k \left[(\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_{it})(\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_{it})^\top\right].
\end{aligned} \tag{3.16}$$

Note Eq. 3.16 imposes the implicit constraint that the number of particles $K$ be greater than the square of the dimensionality of the latents for the covariance matrix $\boldsymbol{\Sigma}_{it}$ to have full rank. We lift

**Algorithm 4:** Learning with reweighted expectation maximization (REM (v2))

**Input:** Data $\mathbf{x}$

Initialize model and proposal parameters $\theta, \eta$

**for** *iteration* $t = 1, 2, \ldots$ **do**

  Draw minibatch of observations $\{\mathbf{x}_n\}_{n=1}^B$

  **for** *observation* $n = 1, 2, \ldots, B$ **do**

    Draw $\mathbf{z}_n^{(1)}, \ldots, \mathbf{z}_n^{(K)} \sim r_{\eta_t}(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)$

    Compute importance weights $\mathbf{w}^{(k)} = \frac{p_{\theta_t}(\mathbf{z}_n^{(k)}, \mathbf{x}_n)}{r_{\eta_t}(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)}$

    Compute $\boldsymbol{\mu}_{nt} = \sum_{k=1}^K \frac{\mathbf{w}^{(k)}}{\sum_{k=1}^K \mathbf{w}^{(k)}} \mathbf{z}_n^{(k)}$ and $\boldsymbol{\Sigma}_{nt} = \sum_{k=1}^K \frac{\mathbf{w}^{(k)}}{\sum_{k=1}^K \mathbf{w}^{(k)}} (\mathbf{z}_n^{(k)} - \boldsymbol{\mu}_{nt})(\mathbf{z}_n^{(k)} - \boldsymbol{\mu}_n)^\top$

    Set proposal $s(\mathbf{z}_n^{(t)}) = \mathcal{N}(\boldsymbol{\mu}_{nt}, \boldsymbol{\Sigma}_{nt})$

  **end for**

  Compute $\nabla_\eta \mathcal{L}(\eta) = \frac{1}{|B|} \sum_{n \in B} \sum_{k=1}^K \frac{\mathbf{v}^{(k)}}{\sum_{k=1}^K \mathbf{v}^{(k)}} \nabla_\eta \log r_\eta(\mathbf{z}_n^{(k)} \mid \mathbf{x}_n)$ and update $\eta$ using Adam

  Compute $\nabla_\theta \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|B|} \sum_{n \in B} \sum_{k=1}^K \frac{\mathbf{v}^{(k)}}{\sum_{k=1}^K \mathbf{v}^{(k)}} \nabla_\theta \log p_\theta(\mathbf{x}_n, \mathbf{z}_n^{(k)})$ and update $\theta$ using Adam

**end for**

this constraint by adding a constant $\epsilon$ to the diagonal of $\boldsymbol{\Sigma}_{it}$ and setting

$$\boldsymbol{\Sigma}_{it} \approx \sum_{k=1}^K (\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_{it})(\mathbf{z}_i^{(k)} - \boldsymbol{\mu}_{it})^\top. \tag{3.17}$$

Algorithm 3 summarizes the procedure for fitting deep generative models with REM where $\mathbf{v}^{(k)}$ is computed the same way as $\mathbf{v}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t)$. We call this algorithm REM (v1).

We can also consider using the rich moment matched distribution $s(\mathbf{z})$ to directly update the generative model. This changes the objective $\mathcal{L}(\theta)$ in Eq. 3.8 to

$$\mathcal{L}(\theta) = \sum_{i=1}^N \sum_{k=1}^K \boldsymbol{\beta}_{it}^k \cdot \log p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)}) \tag{3.18}$$

where $\mathbf{z}_i^{(1)}, \ldots, \mathbf{z}_i^{(K)} \sim s(\mathbf{z}_i)$ and $\boldsymbol{\beta}_{it}^k$ is as defined in Eq. 3.13. We let the recognition network $r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$ be learned the same way as done for REM (v1). Algorithm 4 summarizes the procedure for fitting DPGMS with REM (v2).

*Table 3.1.* Comparing REM against the VAE, the IWAE, and reweighted wake-sleep (RWS). REM uses a rich distribution $s(\mathbf{z})$ found by moment matching to learn the generative model and/or the recognition network $r_\eta(\mathbf{z} \mid \mathbf{x})$.

| Method | Objective | Proposal | Hyperobjective | Hyperproposal |
|---|---|---|---|---|
| VAE | VI | $r_\eta(\mathbf{z} \mid \mathbf{x})$ | $\mathrm{KL}(r_\eta(\mathbf{z} \mid \mathbf{x}) \| \| p_\theta(\mathbf{z} \mid \mathbf{x}))$ | $r_\eta(\mathbf{z} \mid \mathbf{x})$ |
| IWAE | EM | $r_\eta(\mathbf{z} \mid \mathbf{x})$ | $\mathcal{L}_{\text{IWAE}}(\eta)$ | $r_\eta(\mathbf{z} \mid \mathbf{x})$ |
| RWS | EM | $r_\eta(\mathbf{z} \mid \mathbf{x})$ | $\mathrm{KL}(p_\theta(\mathbf{z} \mid \mathbf{x}) \| \| r_\eta(\mathbf{z} \mid \mathbf{x}))$ | $r_\eta(\mathbf{z} \mid \mathbf{x})$ |
| REM(v1) | EM | $r_\eta(\mathbf{z} \mid \mathbf{x})$ | $\mathrm{KL}(p_\theta(\mathbf{z} \mid \mathbf{x}) \| \| r_\eta(\mathbf{z} \mid \mathbf{x}))$ | $s(\mathbf{z})$ |
| REM (v2) | EM | $s(\mathbf{z})$ | $\mathrm{KL}(p_\theta(\mathbf{z} \mid \mathbf{x}) \| \| r_\eta(\mathbf{z} \mid \mathbf{x}))$ | $r_\eta(\mathbf{z} \mid \mathbf{x})$ |

## 3.4 Connections

REM generalizes and connects algorithms that rely on importance sampling to optimize a tighter approximation of the log marginal likelihood (e.g. IWAE (Burda et al., 2015b) and RWS (Bornschein & Bengio, 2014).) We discuss this next.

### 3.4.1 Importance-Weighted Auto-Encoders

IWAE was introduced to learn better generative models (Burda et al., 2015b). It relies on importance sampling to optimize both the model parameters and the recognition network. IWAE maximizes

$$\mathcal{L}_{\text{IWAE}}(\theta, \eta) = \sum_{i=1}^{N} \log \left( \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)} \right) \tag{3.19}$$

where $r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)$ is an importance sampling proposal and $\mathbf{z}_1^{(k)}, \ldots, \mathbf{z}_i^{(K)} \sim r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)$. This objective is simply a biased Monte Carlo approximation of the log marginal likelihood using importance

sampling. To confirm this, write

$$\log p_\theta(\mathbf{x}_{1:N}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \tag{3.20}$$

$$= \sum_{i=1}^{N} \log \int \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i) \cdot r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)}{r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)} \, d\mathbf{z}_i \tag{3.21}$$

$$= \sum_{i=1}^{N} \log \mathbb{E}_{r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)} \left( \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i)}{r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)} \right) \tag{3.22}$$

$$\approx \sum_{i=1}^{N} \log \left( \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)} \right) \tag{3.23}$$

where $\mathbf{z}_1^{(k)}, \ldots, \mathbf{z}_i^{(K)} \sim r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)$. Note the VAE lower bounds Eq. 3.22 using concavity of logarithm, which leads to the ELBO objective.

The IWAE objective is shown to be a tighter approximation to the log marginal likelihood of the data than the ELBO (Burda et al., 2015b); this tightness is determined by the number of particles $K$ used for importance sampling.

Consider taking gradients of $\mathcal{L}_{\text{IWAE}}(\theta, \eta)$ with respect to the model parameters $\theta$,

$$\nabla_\theta \mathcal{L}_{\text{IWAE}}(\theta, \eta) = \sum_{i=1}^{N} \nabla_\theta \log \left( \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)} \right) \tag{3.24}$$

$$= \sum_{i=1}^{N} \frac{\nabla_\theta \left( \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)} \right)}{\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)}} \tag{3.25}$$

$$= \sum_{i=1}^{N} \frac{\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)} \nabla_\theta \log p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)})}{r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)}} \tag{3.26}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{\alpha}_{it}^k \cdot \nabla_\theta \log p_\theta(\mathbf{x}_i, \mathbf{z}_i^{(k)}) \tag{3.27}$$

where $\boldsymbol{\alpha}_{it}^k$ was previously defined in Eq. 3.8. Note Eq. 3.27 is the expression of the REM gradient with respect to the model parameters $\theta$ (Eq. 4.17.)

IWAE updates the proposal by taking gradients of $\mathcal{L}_{\text{IWAE}}(\theta, \eta)$ with respect to $\eta$. As pointed out in Le et al. (2017) this objective does not correspond to minimizing any divergence between the IWAE's proposal $r_\eta(\mathbf{z}_i^{(k)} \mid \mathbf{x}_i)$ and the true posterior. However $\mathcal{L}_{\text{IWAE}}(\theta, \eta)$ can be viewed as a divergence between an *importance weighted distribution* and the true posterior. We refer the reader to Cremer et al. (2017) for a detailed exposition.

To illustrate how REM (v1) improves upon the IWAE, consider replacing $s(\mathbf{z}_i)$ in the definition of $\mathbf{v}(\mathbf{x}_i, \mathbf{z}_i; \theta_t, \eta_t)$ with $r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$. Then taking gradients of Eq. 3.13 with respect to $\eta$ reduces to the IWAE gradient for updating the recognition network $r_\eta(\mathbf{z}_i \mid \mathbf{x}_i)$. Instead of using $r_{\eta_t}(\mathbf{z}_i \mid \mathbf{x}_i)$, REM (v1) uses a more expressive distribution found via moment matching to update the recognition network. This further has the advantage of decoupling the generative model and the recognition network as they do not use the same objective for learning.

### 3.4.2 Reweighted Wake-Sleep

The RWS algorithm extends the wake-sleep (WS) algorithm of Hinton et al. (1995) to importance sampling the same way the IWAE algorithm extends the VAE to importance sampling. It uses the same importance sampling approximation of the log marginal likelihood as IWAE (Eq. 3.23.) Therefore RWS leads to the same gradients with respect to the model parameters than IWAE. The two approaches differ in how they learn the proposal.

The RWS proposal minimizes the inclusive KL, similarly to REM,

$$\eta_{t+1} = \arg\min_{\eta} \mathcal{L}_{\text{RWS}}(\eta) = \text{KL}(p_{\theta_t}(\mathbf{z} \mid \mathbf{x}) || r_\eta(\mathbf{z} \mid \mathbf{x})). \tag{3.28}$$

However, unlike REM which leverages the fact that the inclusive KL admits moment matching as a

solution, RWS minimizes an approximation of the KL,

$$\mathcal{L}_{\text{RWS}}(\eta) = -\mathbb{E}_{p_\theta(\mathbf{z}\,|\,\mathbf{z})}\left(\log r_\eta(\mathbf{z}\,|\,\mathbf{x})\right) + \text{cst} \tag{3.29}$$

$$= -\int \frac{p_\theta(\mathbf{x},\mathbf{z})}{\int p_\theta(\mathbf{x},\mathbf{z})\, d\mathbf{z}} \log r_\eta(\mathbf{z}\,|\,\mathbf{x}) d\mathbf{z} + \text{cst} \tag{3.30}$$

$$= -\mathbb{E}_{r_\eta(\mathbf{z}\,|\,\mathbf{x})}\left(\frac{\frac{p_\theta(\mathbf{x},\mathbf{z})}{r_\eta(\mathbf{z}\,|\,\mathbf{x})}}{\mathbb{E}_{r_\eta(\mathbf{z}\,|\,\mathbf{x})}\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{r_\eta(\mathbf{z}\,|\,\mathbf{x})}\right)} \log r_\eta(\mathbf{z}\,|\,\mathbf{x})\right) + \text{cst} \tag{3.31}$$

$$\approx -\sum_{k=1}^{K} \mathbf{w}_k \log r_\eta(\mathbf{z}^{(k)}\,|\,\mathbf{x}) + \text{cst} \tag{3.32}$$

where $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(K)} \sim r_\eta(\mathbf{z}\,|\,\mathbf{x})$ and $\mathbf{w}_k = \frac{p_\theta(\mathbf{z}^{(k)},\mathbf{x})}{r_\eta(\mathbf{z}^{(k)}\,|\,\mathbf{x})}$. RWS updates its proposal by taking gradients of $\mathcal{L}_{\text{RWS}}(\eta)$ with respect to $\eta$,

$$\nabla_\eta \mathcal{L}_{\text{RWS}}(\eta) = -\sum_{k=1}^{K} \mathbf{w}_k \nabla_\eta \log r_\eta(\mathbf{z}^{(k)}\,|\,\mathbf{x}) \tag{3.33}$$

REM improves upon REM by using a richer hyperproposal than $r_\eta(\mathbf{z}^{(k)}\,|\,\mathbf{x})$ to update its proposal. To see this, replace $s(\mathbf{z})$ used to compute the gradients of the REM objective with respect to $\eta$ with $r_\eta(\mathbf{z}^{(k)}\,|\,\mathbf{x})$ to recover the RWS gradients.

Table 3.1 highlights the differences between the VAE, the IWAE, RWS, REM(v1), and REM(v2).

### 3.5 Empirical Study

In this section, we showcase the benefits of using REM over the VAE and the IWAE. We assess generalization using predictive log-likelihood on held-out data.

Note RWS requires specific architectures, e.g. NADE (Uria et al., 2016) or SBN (Saul et al., 1996), to achieve good results. In our empirical studies we focus on the controlled setting of Burda et al. (2015b), which uses simple MLPs for density estimation. RWS achieves significantly worse results than the VAE when using MLPs and we don't report those results.

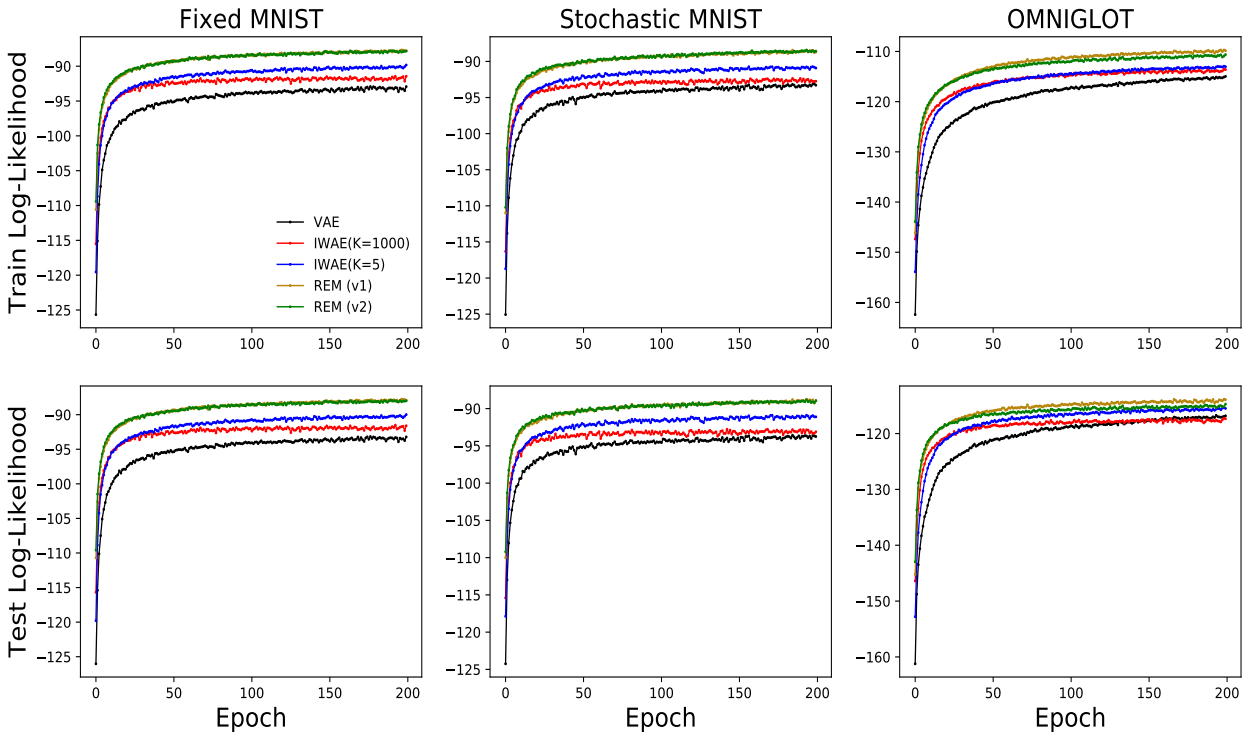



*Figure 3.1.* REM achieves significantly better performance than the VAE and the IWAE on three benchmark datasets in terms of log-likelihood (the higher the better).

We consider several benchmark datasets, which we describe next.

### 3.5.1 Datasets

We evaluated all methods on the OMNIGLOT dataset and two versions of MNIST. The OMNIGLOT is a dataset of handwritten characters in a total of 50 different alphabets (Lake et al., 2013). Each of the characters is a single-channel image with dimension $28 \times 28$. There are in total 24,345 images in the training set and 8,070 images in the test set. MNIST is a dataset of images of handwritten digits introduced by LeCun et al. (1998). The first version of MNIST we consider is the fixed binarization of the MNIST dataset used by Larochelle & Murray (2011). The second version of MNIST corresponds to random binarization; a random binary sample of digits is newly created during optimization to get a minibatch of data. In both cases the images are single-channel and have dimension $28 \times 28$.

There are 60,000 images in the training set and 10,000 images in the test set. All these datasets are available online at https://github.com/yburda/iwae.

### 3.5.2 Settings

We used the same network architecture for all methods. We followed Burda et al. (2015b) and set the generative model, also called a *decoder*, to be a fully connected feed-forward neural network with two layers where each layer has 200 hidden units. We set the recognition network, also called an *encoder*, to be a fully connected feed-forward neural network with two layers and 200 hidden units in each layer. We use two additional linear maps to get the mean and the log-variance for the distribution $r_\eta(\mathbf{z} \mid \mathbf{x})$. The actual variance is obtained by exponentiating the log-variance.

We used a minibatch size of 20 and set the learning rate following the schedule describes in Burda et al. (2015b) with an initial learning rate of $10^{-3}$. We use this same learning rate schedule for both the learning of the generative model and the recognition network. We set the dimension of the latents used as input to the generative model to 20. We set the seed to 2019 for reproducibility. We set the number of particles $K$ to 1,000 for both training and testing. We ran all methods for 200 epochs. We used Amazon EC-2 P3 GPUs for all our experiments.

### 3.5.3 Results

We now describe the results in terms of quality of the learned generative model and proposal.

**EM-based methods learn better generative models.** We assess the quality of the fitted generative model for each method using log-likelihood. We report log-likelihood on both the training set and the test set. Figure 3.1 illustrates the results. The VAE performs the worse on all datasets and on both the training and the test set. The IWAE performs better than the VAE as it optimizes a better objective function to train its generative model. Finally, both versions of REM significantly outperform the IWAE on all cases. This is evidence of the effectiveness of EM as a good alternative for learning

*Table 3.2.* REM (v1) outperforms REM (v2) on all but one dataset. This suggests that recognition networks are effective proposals for the purpose of learning the generative model.

| REM | | Fixed MNIST | | Stochastic MNIST | | Omniglot | |
|---|---|---|---|---|---|---|---|
| Proposal | Hyperproposal | Train | Test | Train | Test | Train | Test |
| $r_\eta(\mathbf{z} \mid \mathbf{x})$ | $s(\mathbf{z})$ | **87.77** | **87.91** | 88.68 | 88.95 | **109.84** | **113.94** |
| $s(\mathbf{z})$ | $r_\eta(\mathbf{z} \mid \mathbf{x})$ | 87.84 | 87.99 | **88.58** | **88.92** | 110.63 | 114.73 |

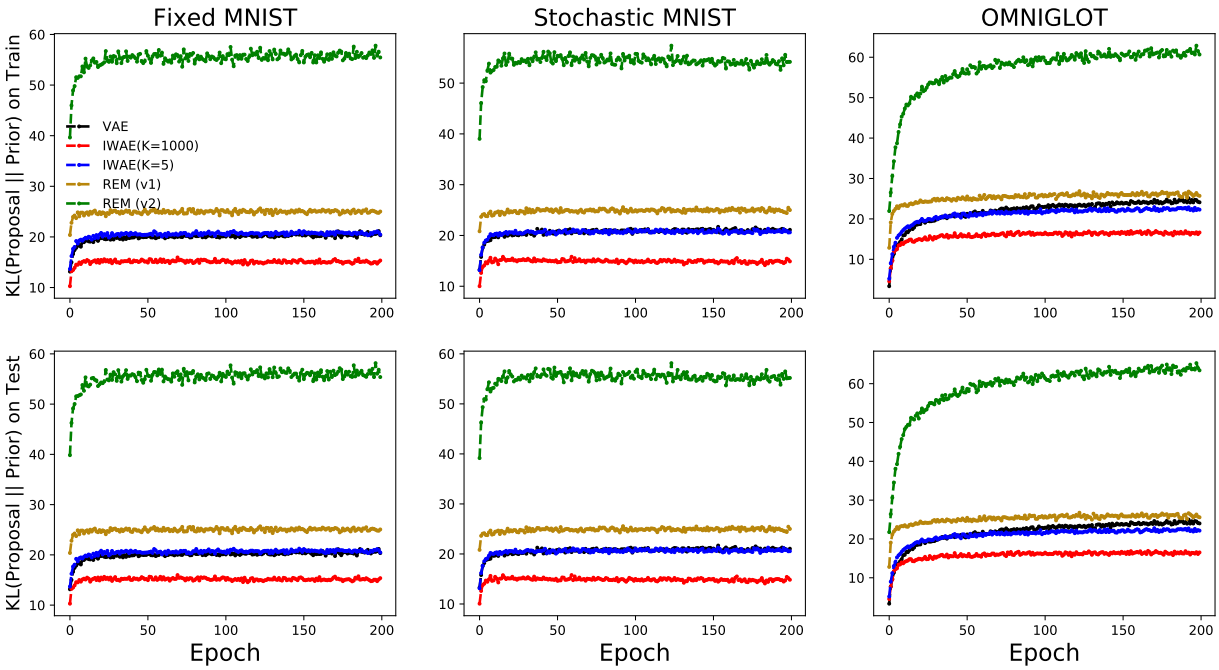



*Figure 3.2.* REM learns a better proposal than the VAE and the IWAE. This figure also shows that the quality of the IWAE's fitted posterior deteriorates as $K$ increases.

deep generative models.

**Recognition networks are good proposals.** Here we study the effect of the proposal on the performance of REM. We report the log-likelihood on both the train and the test set in Table 3.2. As shown in Table 3.2, using the richer distribution $s(\mathbf{z})$ does not always lead to improved performance. These results suggest that recognition networks are good proposals for updating model parameters in deep generative models.

**The inclusive KL is a better hyperobjective.** We also assessed the quality of the learned proposal for each method. We use the KL from the fitted proposal to the prior as a quality measure. This

form of KL is often used to assess latent variable collapse. Figure 3.2 shows REM learns better proposals than both the IWAE and the VAE. It also confirms the quality of the IWAE degrades when the number of particles $K$ increases.

# Chapter 4: Entropy-Regularized Adversarial Learning

Maximum likelihood is the de-facto approach for fitting PGMs to data. The models in Chapter 2 were fit by maximizing likelihood using AVI. Because the likelihood is intractable for DPGMs, we relied on amortized VI and maximized the ELBO

$$\text{ELBO} = E_{p_d(\mathbf{x})} E_{q_\phi(\mathbf{z}\,|\,\mathbf{x})} \left[ \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})} \right] = -\text{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x}) p_d(\mathbf{x}) || p_\theta(\mathbf{x}, \mathbf{z})). \tag{4.1}$$

Maximizing the ELBO is equivalent to minimizing the KL between the model joint and the variational joint, which leads to issues such as latent variable collapse (Bowman et al., 2015; Dieng et al., 2018b). Furthermore, optimizing Eq. 4.1 may lead to blurriness in the generated samples because of a property of the reverse KL known as *zero-forcing* (Minka et al., 2005).

In Chapter 3, we proposed REM, an algorithm that optimizes a better approximation to the log marginal likelihood of the data using EM and moment matching.

In this chapter, we develop entropy-regularized adversarial learning as an alternative to maximum likelihood for fitting DPGMs. From the perspective of PGM, entropy-regularized adversarial learning opens the door for using PGM in tasks where high simulation quality matters (e.g. image generation, image superresolution, data augmentation, and model-based reinforcement learning.) From the DL perspective, entropy-regularized adversarial learning provides a solution to the long-standing mode collapse problem of generative adversarial networks (GANs).

## 4.1 Generative Adversarial Networks

The GAN of Goodfellow et al. (2014) is a DL technique for simulating high-quality data. The GAN and its extensions have achieve state-of-the-art performance in the image domain; for example in image generation (Karras et al., 2019; Brock et al., 2018), image super-resolution (Ledig et al., 2017), and image translation (Isola et al., 2017).

The algorithmic idea behind GANS is to learn to sample high-quality data from a *generator* by following feedback from a *critic* (also called a *discriminator*.) Both the generator and the discriminator are deep neural networks.

A GAN samples data by sampling noise $\boldsymbol{\delta}$ from a fixed distribution $p(\boldsymbol{\delta})$ and then using this noise as input to the generator, the output of which is the sample from the GAN. Denote by $\theta$ the parameters of the generator. Denote by $\tilde{\mathbf{x}}(\boldsymbol{\delta}; \theta)$ the GAN sample. The generative process for data defined by the GAN implies a density $p_\theta(\mathbf{x})$. However this density is undefined (Mohamed & Lakshminarayanan, 2016). Although GANS do not define a tractable density over the generated samples, they can fit their parameters $\theta$ by leveraging feedback from the discriminator. Denote by $D_\phi$ the discriminator; it is a deep neural network with parameters $\phi$ that takes a sample and outputs the probability that the input sample is from the true data generating distribution or from the generator. The parameters $\theta$ and $\phi$ are learned jointly by optimizing the GAN objective,

$$\mathcal{L}_{\text{GAN}}(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} \left[ \log D_\phi(\mathbf{x}) \right] + \mathbb{E}_{\boldsymbol{\delta} \sim p(\boldsymbol{\delta})} \left[ \log \left( 1 - D_\phi(\tilde{\mathbf{x}}(\boldsymbol{\delta}; \theta)) \right) \right], \tag{4.2}$$

where $p_d(\mathbf{x})$ is the empirical data distribution. GANS iteratively maximize the loss in Eq. 4.2 with respect to $\phi$ and minimize it with respect to $\theta$. Maximizing the loss $\mathcal{L}_{\text{GAN}}(\theta, \phi)$ with respect to $\phi$ forces the discriminator to assign high probability to the real data and low probability to samples from the generator. On the other hand, minimizing the loss $\mathcal{L}_{\text{GAN}}(\theta, \phi)$ with respect to $\theta$ forces the discriminator to assign high probability to samples from the generator. These two iterative optimization loops are at odds with each other, hence the word “adversarial" in the name of the

approach.

In practice, the minimax procedure described above is stopped when the generator produces realistic data. This is problematic because producing realistic data does not necessarily correlate with achieving goodness of fit to the true data generating distribution. For example, memorizing the training data is a trivial solution to producing realistic data. Fortunately, GANS do not merely memorize the training data (Zhang et al., 2017; Arora et al., 2017).

However GANS are able to produce samples indistinguishable from real data while still failing to fully capture the data generating distribution (Brock et al., 2018; Karras et al., 2019). Indeed GANS suffer from an issue known as *mode collapse*. When mode collapse happens, the generative distribution $p_\theta(\mathbf{x})$ implied by the GAN sampling procedure is degenerate and has low support (Arora et al., 2017, 2018). Mode collapse causes GANS, to fail both qualitatively and quantitatively. Qualitatively, mode collapse causes lack of diversity in the generated samples. This is problematic for certain applications of GANS, e.g. data augmentation. Quantitatively, mode collapse causes poor generalization to new data. This is because when mode collapse happens, there is a (support) mismatch between the learned distribution $p_\theta(\mathbf{x})$ and the data distribution. Using annealed importance sampling with a kernel density estimate of the likelihood, Wu et al. (2016) report significantly worse log-likelihood scores for GANS when compared to VAES. Similarly poor generalization performance was reported by Grover et al. (2018).

### 4.1.1 Why does mode collapse happen?

For simplicity, and only for the rest of this section, let's denote by $t(\mathbf{x})$ a target distribution of interest. Assume we are using the GAN minimax framework to approximate $t(\mathbf{x})$ with $f(\mathbf{x})$. Denote by $D(\mathbf{x})$ the discriminator. The loss is,

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\mathbf{x}\sim t(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim f(\mathbf{x})}\left[\log(1 - D(\mathbf{x}))\right]. \tag{4.3}$$

The loss $\mathcal{L}_{\text{GAN}}$ in Eq. 4.3 is a concave function of $D(\mathbf{x})$. Taking the gradient of $\mathcal{L}_{\text{GAN}}$ in Eq. 4.3 with respect to $D(\mathbf{x})$ and setting it to zero yields the optimal discriminator (Goodfellow et al., 2014),

$$D^*(\mathbf{x}) = \frac{t(\mathbf{x})}{t(\mathbf{x}) + f(\mathbf{x})} \tag{4.4}$$

Replacing this optimal discriminator in Eq. 4.3 and rearranging terms leads to the following objective for learning $f(\mathbf{x})$:

$$\text{JS}(t(\mathbf{x})||f(\mathbf{x})) = \frac{1}{2}\text{KL}(t(\mathbf{x})||r(\mathbf{x})) + \frac{1}{2}\text{KL}(f(\mathbf{x})||r(\mathbf{x})), \tag{4.5}$$

where $r(\mathbf{x}) = \frac{t(\mathbf{x})+f(\mathbf{x})}{2}$.

Let's look more closely at the objective function $\text{JS}(t(\mathbf{x})||f(\mathbf{x}))$, which we minimize to find a good approximation $f(\mathbf{x})$ for the target distribution $t(\mathbf{x})$. The objective $\text{JS}(t(\mathbf{x})||f(\mathbf{x}))$ is the sum of two KL divergences. The first KL, $\text{KL}(t(\mathbf{x})||r(\mathbf{x}))$ has a *zero-avoiding* behavior (Minka et al., 2005; Dieng et al., 2017), minimizing it yields a distribution $r(\mathbf{x})$ that overgeneralizes $t(\mathbf{x})$. This can be achieved without requiring $f(\mathbf{x})$ to cover all the modes of $t(\mathbf{x})$. Furthermore, the second KL term, $\text{KL}(f(\mathbf{x})||r(\mathbf{x}))$, has a *zero-forcing* behavior (Minka et al., 2005; Dieng et al., 2017), minimizing it yields a distribution $f(\mathbf{x})$ that undergeneralizes $r(\mathbf{x})$. As a consequence, minimizing $\text{JS}(t(\mathbf{x})||f(\mathbf{x}))$ tends to lead to a distribution $f(\mathbf{x})$ that does not cover all the modes of the target distribution $t(\mathbf{x})$.

### 4.1.2 Motivating entropy regularization

In light of the analysis in Section 4.1.1, a natural way to prevent mode collapse in GANS is to maximize entropy (Belghazi et al., 2018). Indeed, adding the entropy of $f(\mathbf{x})$ to the objective

$\mathrm{JS}(t(\mathbf{x})||f(\mathbf{x}))$ leads to an entropy-regularized objective,

$$\mathcal{L}(f(\mathbf{x})) = \frac{1}{2}\mathrm{KL}(t(\mathbf{x})||r(\mathbf{x})) + \frac{1}{2}\mathrm{KL}(f(\mathbf{x})||r(\mathbf{x})) - \lambda\mathbb{E}_{\mathbf{x}\sim f(\mathbf{x})}[\log f(\mathbf{x})] \tag{4.6}$$

$$= \frac{1}{2}\mathrm{KL}(t(\mathbf{x})||r(\mathbf{x})) + \left(\frac{1}{2} - \lambda\right)\mathrm{KL}(f(\mathbf{x})||r(\mathbf{x})) - \lambda\mathbb{E}_{\mathbf{x}\sim f(\mathbf{x})}[\log r(\mathbf{x})]. \tag{4.7}$$

Let's now look closely at Eq. 4.7. The first KL, $\mathrm{KL}(t(\mathbf{x})||r(\mathbf{x}))$ has the same weight as in Eq. 4.5, it yields $r(\mathbf{x})$ that overgeneralizes $t(\mathbf{x})$. The second KL, which leads to a distribution $f(\mathbf{x})$ that undergeneralizes $r(\mathbf{x})$, has reduced effect. There is a new term, $-\mathbb{E}_{\mathbf{x}\sim f(\mathbf{x})}[\log r(\mathbf{x})]$, whose minimization enforces high cross-entropy between $f(\mathbf{x})$ and $r(\mathbf{x})$. This in turn forces $f(\mathbf{x})$ to cover the modes of the target distribution $t(\mathbf{x})$.

Unfortunately maximizing entropy is impossible for GANS, because their entropy is not well-defined.

GAN researchers have looked at indirect ways to alleviate mode collapse. For example, Srivastava et al. (2017) use a reconstructor network that reverses the action of the generator. Lin et al. (2018) use multiple observations (either real or generated) as an input to the discriminator to prevent mode collapse. Azadi et al. (2018) and Turner et al. (2018) use sampling mechanisms to correct errors of the generative distribution. Xiao et al. (2018b) relies on identifying the geometric structure of the data embodied under a specific distance metric. Other works have combined adversarial learning with maximum likelihood (Grover et al., 2018; Yin & Zhou, 2019); however, the low sample quality induced by maximum likelihood still occurs. Finally, Cao et al. (2018) introduce a regularizer for the discriminator to encourage diverse activation patterns in the discriminator across different samples.

## 4.2 Prescribed Generative Adversarial Networks

In this section we leverage the minimax procedure used when fitting GANS, called adversarial learning, within the context of DPGMS. We build on adversarial learning in such a way that our desiderata for DPGM are met. In particular, we maximize entropy to enforce diversity in the data generating

process. We call the resulting learning algorithm *entropy-regularized adversarial learning*. We call a DPGM fit using entropy-regularized adversarial learning a prescribed GAN (PresGAN.)

In this section, we focus on a simple DPGM where the generative process is to sample from the prior $p(\mathbf{z})$ and then condition on the sample to draw data from $p_\theta(\mathbf{x} \mid \mathbf{z})$, an exponential family distribution parameterized by a deep neural network. This generative process implies a well-defined density over $\mathbf{x}$,

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) \cdot p(\mathbf{z}) d\mathbf{z}. \tag{4.8}$$

For simplicity we define the prior $p(\mathbf{z})$ and the likelihood $p_\theta(\mathbf{x} \mid \mathbf{z})$ to be Gaussians,

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \boldsymbol{I}) \quad \text{and} \quad p_\theta(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z})\right). \tag{4.9}$$

The mean $\boldsymbol{\mu}_\theta(\mathbf{z})$ and covariance $\boldsymbol{\Sigma}_\theta(\mathbf{z})$ of the conditional $p_\theta(\mathbf{x} \mid \mathbf{z})$ are given by a neural network that takes $\mathbf{z}$ as input. In general, both the mean $\boldsymbol{\mu}_\theta(\mathbf{z})$ and the covariance $\boldsymbol{\Sigma}_\theta(\mathbf{z})$ can be functions of $\mathbf{z}$. For simplicity, in order to speed up the learning procedure, we set the covariance matrix to be diagonal with elements independent from $\mathbf{z}$, i.e., $\boldsymbol{\Sigma}_\theta(\mathbf{z}) = \text{diag}\left(\boldsymbol{\sigma}^2\right)$, and we learn the vector $\boldsymbol{\sigma}$ together with $\theta$. From now on, we parameterize the mean with $\eta$, write $\boldsymbol{\mu}_\eta(\mathbf{z})$, and define $\theta = (\eta, \boldsymbol{\sigma})$ as the parameters of the generative distribution.

To fit the model parameters $\theta$, we optimize an adversarial loss similarly to GANS. Unlike GANS, the entropy of the generative distribution of a PresGAN is well-defined, and therefore we can prevent mode collapse by adding an entropy regularizer to Eq. 4.2. The idea of entropy regularization has been widely applied in many problems that involve estimation of unknown probability distributions. Examples include approximate Bayesian inference, where the variational objective contains an entropy penalty (Jordan, 1998; Bishop, 2006; Wainwright et al., 2008; Blei et al., 2017b); reinforcement learning, where the entropy regularization allows to estimate more uncertain and explorative policies (Schulman et al., 2015; Mnih et al., 2016); statistical learning, where entropy regularization allows an inferred probability distribution to avoid collapsing to a deterministic solution (Freund &

Schapire, 1997; Soofi, 2000; Jaynes, 2003); or optimal transport (Rigollet & Weed, 2018). More recently, Kumar et al. (2019) have developed maximum-entropy generators for energy-based models using mutual information as a proxy for entropy.

Entropy regularized adversarial learning keeps GANs' ability to generate samples with high perceptual quality while enforcing diversity in the data generation process. The loss is

$$\mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \mathcal{L}_{\text{GAN}}(\theta, \phi) - \lambda \mathcal{H}\left(p_\theta(\mathbf{x})\right). \tag{4.10}$$

Here $\mathcal{H}\left(p_\theta(\mathbf{x})\right)$ denotes the entropy of the generative distribution. It is defined as

$$\mathcal{H}\left(p_\theta(\mathbf{x})\right) = -\mathbb{E}_{p_\theta(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right]. \tag{4.11}$$

The loss $\mathcal{L}_{\text{GAN}}(\theta, \phi)$ in Eq. 4.10 can be that of any of the existing GAN variants. In our empirical study we explore the standard deep convolutional generative adversarial network (DCGAN) (Radford et al., 2015) and the more recent StyleGAN (Karras et al., 2019).

The constant $\lambda$ in Eq. 4.10 is a hyperparameter that controls the strength of the entropy regularization. In the extreme case when $\lambda = 0$, the loss function $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ coincides with the loss of a GAN, where we replaced its ill-defined generative distribution with that in Eq. 4.8. In the other extreme when $\lambda = \infty$, optimizing $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ corresponds to fitting a maximum entropy generator that ignores the data. For any intermediate values of $\lambda$, the first term of $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ encourages the generator to fit the data distribution, whereas the second term encourages diversity.

The entropy $\mathcal{H}\left(p_\theta(\mathbf{x})\right)$ is intractable because the integral in Eq. 4.11 cannot be computed. However, fitting the parameters $\theta$ of PresGANs only requires the gradients of the entropy.

We fit PresGANs following the same adversarial procedure used in GANs. That is, we alternate between updating the parameters of the generative distribution $\theta$ and the parameters of the discriminator $\phi$. The full procedure is given in Algorithm 5. We now describe each part in detail.

### 4.2.1 Fitting the discriminator

Since the entropy term in Eq. 4.10 does not depend on $\phi$, optimizing the discriminator is analogous to optimizing the discriminator of a GAN,

$$\nabla_\phi \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\phi \mathcal{L}_{\text{GAN}}(\theta, \phi). \tag{4.12}$$

To prevent the discriminator from getting stuck in a bad local optimum where it can perfectly distinguish between real and generated data by relying on the added noise, we apply the same amount of noise to the real data $\mathbf{x}$ as the noise added to the generated data. That is, when we train the discriminator we corrupt the real data according to

$$\widehat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \tag{4.13}$$

where $\boldsymbol{\sigma}$ is the standard deviation of the generative distribution and $\mathbf{x}$ denotes the real data. We then let the discriminator distinguish between $\widehat{\mathbf{x}}$ and $\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta)$ from Eq. 4.18.

Using the same noise has a theoretical motivation. Let $p_d(\mathbf{x})$ denote the data distribution and $p_g(\mathbf{x})$ the distribution implied by the sampling procedure:

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = \boldsymbol{\mu}_\eta(\mathbf{z}) \tag{4.14}$$

where $\boldsymbol{\mu}_\eta(\cdot)$ is the output of the generator. Adding noise with the same variance $\boldsymbol{\sigma}$ to a sample from $p_d(\mathbf{x})$ and to a sample from $p_g(\mathbf{x})$ is equivalent to convolving both distributions with the same Gaussian $\mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \boldsymbol{\sigma}^2)$:

$$p_{d,\sigma}(\tilde{\mathbf{x}}) = \int p_d(\mathbf{x}) \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \boldsymbol{\sigma}^2) d\mathbf{x} \tag{4.15}$$

$$p_{g,\sigma}(\tilde{\mathbf{x}}) = \int p_g(\mathbf{x}) \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \boldsymbol{\sigma}^2) d\mathbf{x} \tag{4.16}$$

Now observe that if $p_d = p_g$ then $p_{d,\sigma} = p_{g,\sigma}$ for any value of $\boldsymbol{\sigma}$. This property holds only when using the same noise variance $\sigma$.

The data noising procedure described above is a form of *instance noise* (Sønderby et al., 2016a). However, instead of using a fixed annealing schedule for the noise variance as Sønderby et al. (2016a), we let $\boldsymbol{\sigma}$ be part of the parameters of the generative distribution and fit it using gradient descent according to Eq. 4.25.

### 4.2.2 Fitting the generator

We fit the generator using stochastic gradient descent. This requires computing the gradients of $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ with respect to $\theta$,

$$\nabla_\theta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\theta \mathcal{L}_{\text{GAN}}(\theta, \phi) - \lambda \nabla_\theta \mathcal{H}\left(p_\theta(\mathbf{x})\right). \tag{4.17}$$

We form stochastic estimates of $\nabla_\theta \mathcal{L}_{\text{GAN}}(\theta, \phi)$ based on reparameterization (Kingma & Welling, 2013; Rezende et al., 2014; Titsias & Lázaro-Gredilla, 2014); this requires differentiating Eq. 4.2. Specifically, we introduce a noise variable $\boldsymbol{\epsilon}$ to reparameterize the conditional from Eq. 4.9,[1]

$$\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta) = \boldsymbol{\mu}_\eta(\mathbf{z}) + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \tag{4.18}$$

where $\theta = (\eta, \boldsymbol{\sigma})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Here $\boldsymbol{\mu}_\eta(\mathbf{z})$ and $\boldsymbol{\sigma}$ denote the mean and standard deviation of the conditional $p_\theta(\mathbf{x} \mid \mathbf{z})$, respectively. We now write the first term of Eq. 4.17 as an expectation with respect to the latent variable $\mathbf{z}$ and the noise variable $\boldsymbol{\epsilon}$ and push the gradient into the expectation,

$$\nabla_\theta \mathcal{L}_{\text{GAN}}(\theta, \phi) = \mathbb{E}_{p(\mathbf{z})p(\boldsymbol{\epsilon})}\left[\nabla_\theta \log\left(1 - D_\phi(\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta))\right)\right]. \tag{4.19}$$

[1] With this reparameterization we use the notation $\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta)$ instead of $\tilde{\mathbf{x}}(\mathbf{z}; \theta)$ to denote a sample from the generative distribution.

In practice we use an estimate of Eq. 4.19 using one sample from $p(\mathbf{z})$ and one sample from $p(\boldsymbol{\epsilon})$,

$$\widehat{\nabla}_\theta \mathcal{L}_{\text{GAN}}(\theta, \phi) = \nabla_\theta \log \Big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta))\Big) \tag{4.20}$$

The second term in Eq. 4.17, corresponding to the gradient of the entropy, is intractable. We estimate it using the same approach as Titsias & Ruiz (2018). We first use the reparameterization in Eq. 4.18 to express the gradient of the entropy as an expectation,

$$\begin{aligned}
\nabla_\theta \mathcal{H}\left(p_\theta(\mathbf{x})\right) &= -\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= -\nabla_\theta \mathbb{E}_{p(\boldsymbol{\epsilon})p(\mathbf{z})}\left[\log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\boldsymbol{\epsilon};\theta)}\right] \\
&= -\mathbb{E}_{p(\boldsymbol{\epsilon})p(\mathbf{z})}\left[\nabla_\theta \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\boldsymbol{\epsilon};\theta)}\right] \\
&= -\mathbb{E}_{p(\boldsymbol{\epsilon})p(\mathbf{z})}\left[\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\boldsymbol{\epsilon};\theta)} \nabla_\theta \mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta)\right],
\end{aligned}$$

where we have used the score function identity $\mathbb{E}_{p_\theta(\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x})\right] = 0$ on the second line. We form a one-sample estimator of the gradient of the entropy as

$$\widehat{\nabla}_\theta \mathcal{H}\left(p_\theta(\mathbf{x})\right) = -\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\boldsymbol{\epsilon};\theta)} \times \nabla_\theta \mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta). \tag{4.21}$$

In Eq. 4.21, the gradient with respect to the reparameterization transformation $\nabla_\theta \mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta)$ is tractable and can be obtained via back-propagation. We now derive $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$,

$$\begin{aligned}
\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) &= \frac{\nabla_{\mathbf{x}} p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})} \\
&= \frac{\int \nabla_{\mathbf{x}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})} \\
&= \int \frac{\frac{\nabla_{\mathbf{x}} p_\theta(\mathbf{x} \mid \mathbf{z})}{p_\theta(\mathbf{x} \mid \mathbf{z})} p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} d\mathbf{z} \\
&= \int \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} \\
&= \mathbb{E}_{p_\theta(\mathbf{z} \mid \mathbf{x})} \left[ \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}) \right].
\end{aligned}$$

While this expression is still intractable, we can estimate it. One way is to use self-normalized importance sampling with a proposal learned using moment matching with an encoder as we did in Chapter 3 (Dieng & Paisley, 2019). However, this would lead to a biased (albeit asymptotically unbiased) estimate of the entropy. In this paper, we form an unbiased estimate of $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ using samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}$ from the posterior,

$$\widehat{\nabla}_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(m)}), \quad \text{where} \quad \mathbf{z}^{(m)} \sim p_\theta(\mathbf{z} \mid \mathbf{x}). \tag{4.22}$$

We obtain these samples using Hamiltonian Monte Carlo (HMC) (Neal et al., 2011). Crucially, in order to speed up the algorithm, we initialize the HMC sampler at stationarity. That is, we initialize the HMC sampler with the sample $\mathbf{z}$ that was used to produce the generated sample $\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta)$ in Eq. 4.18, which by construction is an exact sample from $p_\theta(\mathbf{z} \mid \mathbf{x})$. This implies that only a few HMC iterations suffice to get good estimates of the gradient (Titsias & Ruiz, 2018). We also found this holds empirically; for example in the empirical study, we use 2 burn-in iterations and $M = 2$ HMC samples to form the Monte Carlo estimate in Eq. 4.22.

Finally, using Eqs. 4.17 and 4.20 to 4.22 we can approximate the gradient of the entropy-regularized

adversarial loss with respect to the model parameters $\theta$,

$$\widehat{\nabla}_\theta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\theta \log \left(1 - D_\phi(\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta))\right) + \frac{\lambda}{M} \sum_{m=1}^{M} \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(m)})|_{\mathbf{x}=\mathbf{x}(\mathbf{z}^{(m)}, \boldsymbol{\epsilon}; \theta)} \times \nabla_\theta \mathbf{x} \left(\mathbf{z}^{(m)}, \boldsymbol{\epsilon}; \theta\right). \tag{4.23}$$

In particular, the gradient with respect to the generator's parameters $\eta$ is unbiasedly approximated by

$$\widehat{\nabla}_\eta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\eta \log \left(1 - D_\phi(\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta))\right) - \frac{\lambda}{M} \sum_{m=1}^{M} \frac{\mathbf{x}(\mathbf{z}^{(m)}, \boldsymbol{\epsilon}; \theta) - \boldsymbol{\mu}_\eta \left(\mathbf{z}^{(m)}\right)}{\boldsymbol{\sigma}^2} \nabla_\eta \boldsymbol{\mu}_\eta(\mathbf{z}^{(m)}), \tag{4.24}$$

and the gradient estimator with respect to the standard deviation $\boldsymbol{\sigma}$ is

$$\widehat{\nabla}_{\boldsymbol{\sigma}} \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_{\boldsymbol{\sigma}} \log \left(1 - D_\phi(\mathbf{x}(\mathbf{z}, \boldsymbol{\epsilon}; \theta))\right) - \frac{\lambda}{M} \sum_{m=1}^{M} \frac{\mathbf{x}(\mathbf{z}^{(m)}, \boldsymbol{\epsilon}; \theta) - \boldsymbol{\mu}_\eta \left(\mathbf{z}^{(m)}\right)}{\boldsymbol{\sigma}^2} \cdot \boldsymbol{\epsilon}. \tag{4.25}$$

These gradients are used in a stochastic optimization algorithm to fit the generative distribution of PresGAN.

Note there are two failure cases brought in by learning the variance $\boldsymbol{\sigma}^2$ using gradient descent.

The first failure mode is when $\boldsymbol{\sigma}^2$ gets very small, which makes the gradient of the entropy in Eq. 4.24 dominate the overall gradient of the generator. This is problematic because the learning signal from the discriminator is lost.

The second failure mode is when the variance gets very large. Consider the adversarial loss with data noising,

$$\mathcal{L}(\eta, \boldsymbol{\sigma}, \phi) = \mathbb{E}_{p_d(\mathbf{x})p(\boldsymbol{\epsilon})} \left[\log D_\phi(\mathbf{x} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon})\right] + \mathbb{E}_{p(\mathbf{z})p(\boldsymbol{\epsilon})} \left[\log \left(1 - D_\phi \left(\boldsymbol{\mu}_\eta(\mathbf{z}) + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}\right)\right)\right] \tag{4.26}$$

When $p_d = p_g$, then the adversarial loss function $\mathcal{L}(\eta, \boldsymbol{\sigma}, \phi)$ is constant with respect to $\boldsymbol{\sigma}$ and as

**Algorithm 5:** Entropy-Regularized Adversarial Learning

Initialize parameters $\eta, \boldsymbol{\sigma}, \phi$
**for** iteration $t = 1, 2, \ldots$ **do**
  Draw minibatch of observations $\mathbf{x}_1, \ldots, \mathbf{x}_b, \ldots, \mathbf{x}_B$
  **for** $b = 1, 2, \ldots, B$ **do**
    Get noised data: $\boldsymbol{\epsilon}_b \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\widehat{\mathbf{x}}_b = \mathbf{x}_b + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_b$
    Draw latent variable $\mathbf{z}_b \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    Generate data: $\mathbf{s}_b \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\tilde{\mathbf{x}}_b = \tilde{\mathbf{x}}_b(\mathbf{z}_b, \mathbf{s}_b; \theta) = \boldsymbol{\mu}_\eta(\mathbf{z}_b) + \boldsymbol{\sigma} \odot \mathbf{s}_b$
  **end for**
  Compute $\nabla_\phi \mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ (Eq. 4.12) and take a gradient step for $\phi$
  Initialize an HMC sampler using $\mathbf{z}_b$
  Draw $\tilde{\mathbf{z}}_b^{(m)} \sim p_\theta(\mathbf{z} \mid \tilde{\mathbf{x}}_b)$ for $m = 1, \ldots, M$ and $b = 1, \ldots, B$ using that sampler
  Compute $\widehat{\nabla}_\eta \mathcal{L}_{\text{PresGAN}}((\eta, \boldsymbol{\sigma}), \phi)$ (Eq. 4.24) and take a gradient step for $\eta$
  Compute $\widehat{\nabla}_{\boldsymbol{\sigma}} \mathcal{L}_{\text{PresGAN}}((\eta, \boldsymbol{\sigma}), \phi)$ (Eq. 4.25) and take a gradient step for $\boldsymbol{\sigma}$
  Truncate $\boldsymbol{\sigma}$ in the range $[\boldsymbol{\sigma}_{\text{low}}, \boldsymbol{\sigma}_{\text{high}}]$
**end for**

a result, the gradient of $\mathcal{L}(\eta, \boldsymbol{\sigma}, \phi)$ with respect to $\boldsymbol{\sigma}$ is zero. However, during training $p_d \neq p_g$. This can lead to large values for $\boldsymbol{\sigma}$ because the generator can completely fool the discriminator so that $D_\phi(\tilde{\mathbf{x}}) = \frac{1}{2}$, its optimal value, by letting $\boldsymbol{\sigma} \to \infty$. However setting $\boldsymbol{\sigma}$ very large is undesirable since it corresponds to the bad equilibrium point where the samples from the data distribution and from the generative distribution are indistinguishable from one another simply because they are both buried in noise.

### 4.2.3 Variance Regularization

We propose to alleviate the two failure modes discussed above by regularizing the variance to prevent it from reaching very low or very large values.

**Truncation.** One way to regularize the variance $\boldsymbol{\sigma}$ is to simply bound it during optimization, $\boldsymbol{\sigma}_{\text{low}} \leq \boldsymbol{\sigma} \leq \boldsymbol{\sigma}_{\text{high}}$. Note this is applied element-wise. The limits $\boldsymbol{\sigma}_{\text{low}}$ and $\boldsymbol{\sigma}_{\text{high}}$ are hyperparameters.

**Entropy minimization.** To avoid large values of $\boldsymbol{\sigma}$, we can minimize the entropy of the noise

process $\mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \boldsymbol{\sigma}^2)$. The regularized objective for PresGAN becomes

$$\mathcal{L}_{\text{PresGAN}}(\eta, \boldsymbol{\sigma}, \phi) = \mathcal{L}_{\text{GAN}}(\eta, \boldsymbol{\sigma}, \phi) - \lambda \mathcal{H}(p_{\eta,\sigma}(\mathbf{x})) + \tilde{\lambda} \sum_{d=1}^{D} \log \boldsymbol{\sigma}_d^2 \tag{4.27}$$

where $\tilde{\lambda} > 0$ is a hyperparameter that determines the strength of the regularization of the entropy of the noise process. The hyperparameter $\lambda$ controls the entropy regularization of the generative distribution, as described earlier.

Note making $\boldsymbol{\sigma}$ arbitrarily large increases the entropy of the generative distribution $p_{\eta,\sigma}(\mathbf{x})$. However, the term $\tilde{\lambda} \sum_{d=1}^{D} \log \boldsymbol{\sigma}_d^2$ in Eq. 4.27 will prevent that behavior and ensures the entropy of the generative distribution is maximized by means of the latent variables $\mathbf{z}$ and not the noise variance $\boldsymbol{\sigma}$.

Regularizing the variance of the noise process as described above yields an interesting result we summarize in the following proposition.

***Proposition.*** *Consider the generative distribution of Pres*GAN *under a Gaussian likelihood*

$$p_{\eta,\sigma}(\mathbf{x}) = \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\eta(\mathbf{z}), \boldsymbol{\sigma}^2) p(\mathbf{z}) d\mathbf{z}.$$

*Then when $\tilde{\lambda} = \lambda$ in Eq. 4.27,*

$$\mathcal{I}(\mathbf{x}, \mathbf{z}) = \mathcal{H}(p_{\eta,\sigma}(\mathbf{x})) - \sum_{d=1}^{D} \log \boldsymbol{\sigma}_d^2$$

*where $\mathcal{I}(\mathbf{x}, \mathbf{z})$ denotes the mutual information between $\mathbf{x}$ and $\mathbf{z}$ under the generative model.*

The proposition above means that under Gaussian likelihood and Gaussian noise process, optimizing Eq. 4.27 is equivalent to adversarial learning with a mutual information regularizer.

***Proof.*** Denote by $\mathcal{I}(\mathbf{x}, \mathbf{z})$ the mutual information between $\mathbf{x}$ and $\mathbf{z}$ under the PresGAN generative

distribution. Then,

$$\mathcal{I}(\mathbf{x}, \mathbf{z}) = \int p_{\eta,\sigma}(\mathbf{x}, \mathbf{z}) \log \frac{p_{\eta,\sigma}(\mathbf{x}, \mathbf{z})}{p_{\eta,\sigma}(\mathbf{x}) p(\mathbf{z})} d\mathbf{x} d\mathbf{z} \tag{4.28}$$

$$= \int p_{\eta,\sigma}(\mathbf{x}, \mathbf{z}) \log \frac{p_{\eta,\sigma}(\mathbf{x}|\mathbf{z})}{p_{\eta,\sigma}(\mathbf{x})} d\mathbf{x} d\mathbf{z} \tag{4.29}$$

$$= -\int p_{\eta,\sigma}(\mathbf{x}) \log p_{\eta,\sigma}(\mathbf{x}) d\mathbf{x} + \int p(\mathbf{z}) \left( \int p_{\eta,\sigma}(\mathbf{x}|\mathbf{z}) \log p_{\eta,\sigma}(\mathbf{x}|\mathbf{z}) d\mathbf{x} \right) d\mathbf{z} \tag{4.30}$$

$$= \mathcal{H}(p_{\eta,\sigma}(\mathbf{x})) - \sum_{d=1}^{D} \log \boldsymbol{\sigma}_d^2 + \text{cst} \tag{4.31}$$

where we used the Gaussian assumption on the likelihood to replace $\int p_{\eta,\sigma}(\mathbf{x}|\mathbf{z}) \log p_{\eta,\sigma}(\mathbf{x}|\mathbf{z}) d\mathbf{x}$, the negative entropy of a Gaussian, with $-\sum_{d=1}^{D} \log \boldsymbol{\sigma}_d^2 + \text{cst}$.

## 4.3 Empirical Study

Here we demonstrate PresGANs' ability to prevent mode collapse and generate high-quality samples. We also evaluate its predictive performance as measured by log-likelihood.

### 4.3.1 Simulation Study

In this section, we fit a GAN to a toy synthetic dataset of 10 modes. We choose the hyperparameters such that the GAN collapses. We then apply these same hyperparameters to fit a PresGAN on the same synthetic dataset. This experiment demonstrates the PresGAN's ability to correct the mode collapse problem of a GAN.

We form the target distribution by organizing a uniform mixture of $K = 10$ two-dimensional Gaussians on a ring. The radius of the ring is $r = 3$ and each Gaussian has standard deviation 0.05. We then slice the circle into $K$ parts. The location of the centers of the mixture components are deter-

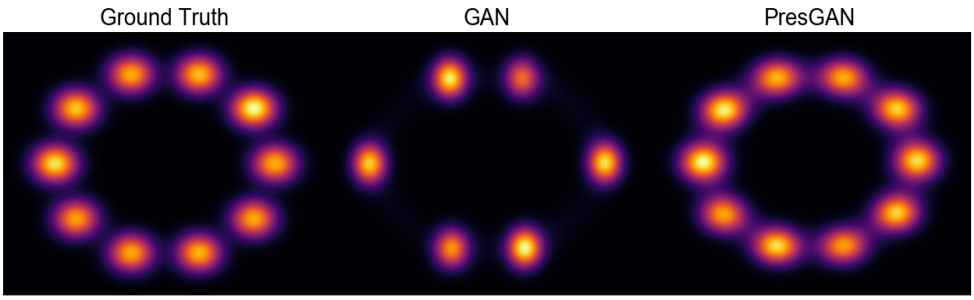



*Figure 4.1.* Density estimation with GAN and PresGAN on a toy two-dimensional experiment. The ground truth is a uniform mixture of 10 Gaussians organized on a ring. Given the right set of hyperparameters, a GAN could perfectly fit this target distribution. In this example we chose the GAN hyperparameters such that it collapses—here 4 out of 10 modes are missing. We then fit the PresGAN using the same hyperparameters as the collapsing GAN. The PresGAN is able to correct the collapsing behavior of the GAN and learns a good fit for the target distribution.

mined as follows. Consider the $k^{\text{th}}$ mixture component. Its coordinates in the 2D space are

$$\text{center}_x = r \cdot \cos\Big(k \cdot \frac{2\pi}{K}\Big) \quad \text{and} \quad \text{center}_y = r \cdot \sin\Big(k \cdot \frac{2\pi}{K}\Big).$$

We draw 5,000 samples from the target distribution and fit a GAN and a PresGAN.

We set the dimension of the latent variables $\mathbf{z}$ used as the input to the generators to 10. We let both the generators and the discriminators have three fully connected layers with tanh activations and 128 hidden units in each layer. We set the minibatch size to 100 and use Adam for optimization (Kingma & Ba, 2014), with a learning rate of $10^{-3}$ and $10^{-4}$ for the discriminator and the generator respectively. The Adam hyperparameters are $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We take one step to optimize the generator for each step of the discriminator. We pick a random minibatch at each iteration and run both the GAN and the PresGAN for 500 epochs.

For PresGAN we set the burn-in and the number of HMC samples to 2. We choose a standard number of 5 leapfrog steps and set the HMC learning rate to 0.02. The acceptance rate is fixed at 0.67. The log-variance of the noise of the generative distribution of PresGAN is initialized at 0.0. We put a threshold on the variance to a minimum value of $\boldsymbol{\sigma}_{\text{low}} = 10^{-2}$ and a maximum value of $\boldsymbol{\sigma}_{\text{high}} = 0.3$. The regularization parameter $\lambda$ is 0.1. We fit the log-variance using Adam with a learning rate of

$10^{-4}$.

Figure 4.1 demonstrates how the PresGAN alleviates mode collapse. The distribution learned by the regular GAN misses 4 modes of the target distribution. The PresGAN is able to recover all the modes of the target distribution.

### 4.3.2 Assessing mode collapse

In this section we evaluate PresGANs' ability to mitigate mode collapse on real datasets. We run two sets of experiments. In the first set of experiments we adopt the current experimental protocol for assessing mode collapse in the GAN literature. That is, we use the MNIST and STACKEDMNIST datasets, for which we know the true number of modes, and report two metrics: the number of modes recovered by the PresGAN and the KL divergence between the label distribution induced by the PresGAN and the true label distribution. In the second set of experiments we demonstrate that mode collapse can happen in GANs even when the number of modes is as low as 10 but the data is imbalanced.

**Increased number of modes.** We consider the MNIST and STACKEDMNIST datasets. MNIST is a dataset of hand-written digits,[2] in which each $28 \times 28 \times 1$ image corresponds to a digit. There are 60,000 training digits and 10,000 digits in the test set. MNIST has 10 modes, one for each digit. STACKEDMNIST is formed by concatenating triplets of randomly chosen MNIST digits along the color channel to form images of size $28 \times 28 \times 3$ (Metz et al., 2017). We keep the same size as the original MNIST, 60,000 training digits for 10,000 test digits. The total number of modes in STACKEDMNIST is 1,000, corresponding to the number of possible triplets.

We consider DCGAN as the base architecture and, following Radford et al. (2015), we resize the spatial resolution of images to $64 \times 64$ pixels.

To measure the degree of mode collapse we form two diversity metrics, following Srivastava et al.

[2] See http://yann.lecun.com/exdb/mnist.

*Table 4.1.* Assessing mode collapse on MNIST. The true total number of modes is 10. All methods capture all the 10 modes. The KL captures a notion of discrepancy between the labels of real versus generated images. PresGAN generates images whose distribution of labels is closer to the data distribution, as evidenced by lower KL scores.

| Method | Modes | KL |
|---|---|---|
| DCGAN (Radford et al., 2015) | $10 \pm 0.0$ | $0.902 \pm 0.036$ |
| VEEGAN (Srivastava et al., 2017) | $10 \pm 0.0$ | $0.523 \pm 0.008$ |
| PACGAN (Lin et al., 2018) | $10 \pm 0.0$ | $0.441 \pm 0.009$ |
| PresGAN (this paper) | $\mathbf{10 \pm 0.0}$ | $\mathbf{0.003 \pm 0.001}$ |

*Table 4.2.* Assessing mode collapse on STACKEDMNIST. The true total number of modes is 1,000. All methods suffer from collapse except PresGAN, which captures nearly all the modes of the data distribution. Furthermore, PresGAN generates images whose distribution of labels is closer to the data distribution, as evidenced by lower KL scores.

| Method | Modes | KL |
|---|---|---|
| DCGAN (Radford et al., 2015) | $392.0 \pm 7.376$ | $8.012 \pm 0.056$ |
| VEEGAN (Srivastava et al., 2017) | $761.8 \pm 5.741$ | $2.173 \pm 0.045$ |
| PACGAN (Lin et al., 2018) | $992.0 \pm 1.673$ | $0.277 \pm 0.005$ |
| PresGAN (this paper) | $\mathbf{999.6 \pm 0.489}$ | $\mathbf{0.115 \pm 0.007}$ |

(2017). Both of these metrics require to fit a classifier to the training data. Once the classifier has been fit, we sample $S$ images from the generator. The first diversity metric is the *number of modes captured*, measured by the number of classes that are captured by the classifier. We say that a class $k$ has been captured if there is at least one generated sample for which the probability of being assigned to class $k$ is the largest. The second diversity metric is the KL *divergence* between two discrete distributions: the empirical average of the (soft) output of the classifier on generated images, and the empirical average of the (soft) output of the classifier on real images from the test set. We choose the number of generated images $S$ to match the number of test samples on each dataset. That is, $S = 10{,}000$ for both MNIST and STACKEDMNIST. We expect the KL divergence to be zero if the distribution of the generated samples is indistinguishable from that of the test samples.

We measure the two mode collapse metrics described above against DCGAN (Radford et al., 2015) (the base architecture of PresGAN for this experiment). We also compare against other methods that

*Table 4.3.* Assessing the impact of the entropy regularization parameter $\lambda$ on mode collapse on MNIST and STACKEDMNIST. When $\lambda = 0$ (i.e., no entropy regularization is applied to the generator), then mode collapse occurs as expected. When entropy regularization is applied but the value of $\lambda$ is very small ($\lambda = 10^{-6}$) then mode collapse can still occur as the level of regularization is not enough. When the value of $\lambda$ is appropriate for the data then mode collapse does not occur. Finally, when $\lambda$ is too high then mode collapse can occur because the entropy maximization term dominates and the data is poorly fit.

| | MNIST | | STACKEDMNIST | |
|---|---|---|---|---|
| $\lambda$ | Modes | KL | Modes | KL |
| 0 | $10 \pm 0.0$ | $0.050 \pm 0.0035$ | $418.2 \pm 7.68$ | $4.151 \pm 0.0296$ |
| $10^{-6}$ | $10 \pm 0.0$ | $0.005 \pm 0.0008$ | $989.8 \pm 1.72$ | $0.239 \pm 0.0059$ |
| $10^{-2}$ | $\mathbf{10 \pm 0.0}$ | $\mathbf{0.003 \pm 0.0006}$ | $\mathbf{999.6 \pm 0.49}$ | $0.115 \pm 0.0074$ |
| $5 \times 10^{-2}$ | $10 \pm 0.0$ | $0.004 \pm 0.0008$ | $999.4 \pm 0.49$ | $\mathbf{0.099 \pm 0.0047}$ |
| $10^{-1}$ | $10 \pm 0.0$ | $0.005 \pm 0.0004$ | $999.4 \pm 0.80$ | $0.102 \pm 0.0032$ |
| $5 \times 10^{-1}$ | $10 \pm 0.0$ | $0.006 \pm 0.0011$ | $907.0 \pm 9.27$ | $0.831 \pm 0.0209$ |

aim at alleviating mode collapse in GANS, namely, VEEGAN (Srivastava et al., 2017) and PACGAN (Lin et al., 2018). For PresGAN we set the entropy regularization parameter $\lambda$ to 0.01. We chose the variance thresholds to be $\boldsymbol{\sigma}_{\text{low}} = 0.001$ and $\boldsymbol{\sigma}_{\text{high}} = 0.3$.

Tables 4.1 and 4.2 show the number of captured modes and the KL for each method. The results are averaged across 5 runs. All methods capture all the modes of MNIST. This is not the case on STACKEDMNIST, where the PresGAN is the only method that can capture all the modes. Finally, the proportion of observations in each mode of PresGAN is closer to the true proportion in the data, as evidenced by lower KL divergence scores.

We also study the impact of the entropy regularization by varying the hyperparameter $\lambda$ from 0 to 0.5. Table 4.3 illustrates the results. Unsurprisingly, when there is no entropy regularization, i.e., when $\lambda = 0$, then mode collapse occurs. This is also the case when the level of regularization is not enough ($\lambda = 10^{-6}$). There is a whole range of values for $\lambda$ such that mode collapse does not occur ($\lambda \in \{0.01, 0.05, 0.1\}$). Finally, when $\lambda$ is too high for the data and architecture under study, mode collapse can still occur. This is because when $\lambda$ is too high, the entropy regularization term dominates the loss in Eq. 4.10 and in turn the generator does not fit the data as well. This is also evidenced by the higher KL divergence score when $\lambda = 0.5$ vs. when $0 < \lambda < 0.5$.

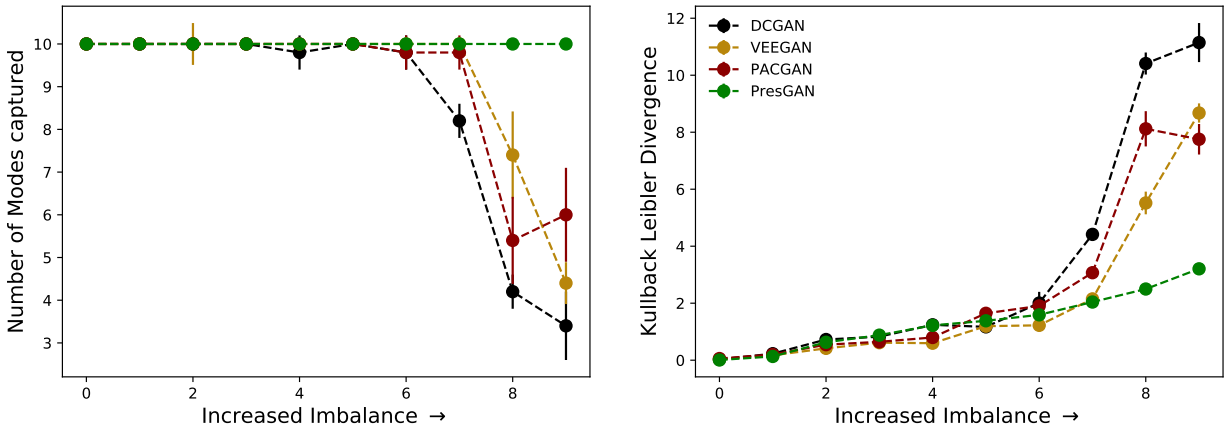



*Figure 4.2.* Assessing mode collapse under increased data imbalance on MNIST. The figures show the number of modes captured (higher is better) and the KL divergence (lower is better) under increasingly imbalanced settings. The maximum number of modes in each case is 10. All methods suffer from mode collapse as the level of imbalance increases except for the PresGAN which is robust to data imbalance.

**Increased data imbalance.** We now show that mode collapse can occur in GANS when the data is imbalanced, even when the number of modes of the data distribution is small. We follow Dieng et al. (2018a) and consider a perfectly balanced version of MNIST as well as nine imbalanced versions. To construct the balanced dataset we used 5,000 training examples per class, totaling 50,000 training examples. We refer to this original balanced dataset as $D_0$. Each additional training set $D_k$ leaves only 5 training examples for each class $j \leq k$, and 5,000 for the rest. (See the Appendix for all the class distributions.)

We used the same classifier trained on the unmodified MNIST but fit each method on each of the 9 new MNIST distributions. We chose $\lambda = 0.1$ for PresGAN. Figure 4.2 illustrates the results in terms of both metrics—number of modes and KL divergence. DCGAN, VEEGAN, and PACGAN face mode collapse as the level of imbalance increases. This is not the case for PresGAN, which is robust to imbalance and captures all the 10 modes.

*Figure 4.3.* Generated images on FFHQ for StyleGAN (left) and PresGAN (right). The PresGAN maintains the high perceptual quality of the StyleGAN.

*Table 4.4.* Fréchet Inception distance (FID) (lower is better). PresGAN has lower FID scores than DCGAN, VEEGAN, and PACGAN. This is because PresGAN mitigates mode collapse while preserving sample quality.

| Method | Dataset | FID |
|---|---|---|
| DCGAN (Radford et al., 2015) | MNIST | 113.129 ± 0.490 |
| VEEGAN (Srivastava et al., 2017) | MNIST | 68.749 ± 0.428 |
| PACGAN (Lin et al., 2018) | MNIST | 58.535 ± 0.135 |
| PresGAN (this paper) | MNIST | **42.019** ± **0.244** |
| DCGAN | STACKEDMNIST | 97.788 ± 0.199 |
| VEEGAN | STACKEDMNIST | 86.689 ± 0.194 |
| PACGAN | STACKEDMNIST | 117.128 ± 0.172 |
| PresGAN | STACKEDMNIST | **23.965** ± **0.134** |
| DCGAN | CIFAR-10 | 103.049 ± 0.195 |
| VEEGAN | CIFAR-10 | 95.181 ± 0.416 |
| PACGAN | CIFAR-10 | 54.498 ± 0.337 |
| PresGAN | CIFAR-10 | **52.202** ± **0.124** |
| DCGAN | CELEBA | 39.001 ± 0.243 |
| VEEGAN | CELEBA | 46.188 ± 0.229 |
| PACGAN | CELEBA | 36.058 ± 0.212 |
| PresGAN | CELEBA | **29.115** ± **0.218** |

### 4.3.3 Assessing sample quality

In this section we assess PresGANs' ability to generate samples of high perceptual quality. We rely on perceptual quality of generated samples and on Fréchet Inception distance (FID) scores (Heusel et al., 2017). We also consider two different GAN architectures, the standard DCGAN and the more recent StyleGAN, to show robustness of PresGANs vis-a-vis the underlying GAN architecture.

**DCGAN.** We use DCGAN (Radford et al., 2015) as the base architecture and build PresGAN on top of it. We consider four datasets: MNIST, STACKEDMNIST, CIFAR-10, and CelebA. CIFAR-10 (Krizhevsky et al., 2009) is a well-studied dataset of $32 \times 32$ images that are classified into one of the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CelebA (Liu et al., 2015) is a large-scale face attributes dataset. Following Radford et al. (2015), we resize all images to $64 \times 64$ pixels. We use the default DCGAN settings. We refer the reader to the code we used for DCGAN, which was taken from `https://github.com/pytorch/examples/tree/master/dcgan`. We set the seed to 2019 for reproducibility.

There are hyperparameters specific to PresGAN. These are the noise and HMC hyperparameters. We set the learning rate for the noise parameters $\boldsymbol{\sigma}$ to $10^{-3}$ and constrain its values to be between $10^{-3}$ and 0.3 for all datasets. We initialize $\log \boldsymbol{\sigma}$ to $-0.5$. We set the burn-in and the number of HMC samples to 2. We choose a standard number of 5 leapfrog steps and set the HMC learning rate to 0.02. The acceptance rate is fixed at 0.67. We found that different $\lambda$ values worked better for different datasets. We used $\lambda = 5 \times 10^{-4}$ for CIFAR-10 and CELEBA $\lambda = 0.01$ for MNIST and STACKEDMNIST.

We found the PresGAN's performance to be robust to the default settings for most of these hyperparameters. However we found the initialization for $\boldsymbol{\sigma}$ and its learning rate to play a role in the quality of the generated samples. The hyperparameters mentioned above for $\boldsymbol{\sigma}$ worked well for all datasets.

Table 4.4 shows the FID scores for DCGAN and PresGAN across the four datasets. We can conclude that PresGAN generates images of high visual quality. In addition, the FID scores are lower because PresGAN explores more modes than DCGAN. Indeed, when the generated images account for more modes, the FID sufficient statistics (the mean and covariance of the Inception-v3 pool3 layer) of the generated data get closer to the sufficient statistics of the empirical data distribution.

We also report the FID for VEEGAN and PACGAN in Table 4.4. VEEGAN achieves better FID scores than DCGAN on all datasets but CELEBA. This is because VEEGAN collapses less than DCGAN as evidenced by Table 4.1 and Table 4.2. PACGAN achieves better FID scores than both DCGAN and VEEGAN on all datasets but on STACKEDMNIST where it achieves a significantly worse FID score. Finally, PresGAN outperforms all of these methods on the FID metric on all datasets signaling its ability to mitigate mode collapse while preserving sample quality.

Besides the FID scores, we also assess the visual quality of the generated images. In **??** of the appendix, we show randomly generated (not cherry-picked) images from DCGAN, VEEGAN, PACGAN, and PresGAN. For PresGAN, we show the mean of the conditional distribution of $\mathbf{x}$ given $\mathbf{z}$. The samples generated by PresGAN have high visual quality; in fact their quality is comparable to or

better than the DCGAN samples.

**StyleGAN.** We now consider a more recent GAN architecture (StyleGAN) (Karras et al., 2019) and a higher resolution image dataset (FFHQ). FFHQ is a diverse dataset of faces from Flickr[3] introduced by Karras et al. (2019). The dataset contains 70,000 high-quality PNG images with considerable variation in terms of age, ethnicity, and image background. We use a resolution of $128 \times 128$ pixels.

StyleGAN feeds multiple sources of noise $\mathbf{z}$ to the generator. In particular, it adds Gaussian noise after each convolutional layer before evaluating the nonlinearity. Building PresGAN on top of StyleGAN therefore requires to sample all noise variables $\mathbf{z}$ through HMC at each training step. To speed up the training procedure, we only sample the noise variables corresponding to the input latent code and condition on all the other Gaussian noise variables. In addition, we do not follow the progressive growing of the networks of Karras et al. (2019) for simplicity.

For this experiment, we choose the same HMC hyperparameters as for the previous experiments but restrict the variance of the generative distribution to be $\boldsymbol{\sigma}_{\text{high}} = 0.2$. We set $\lambda = 0.001$ for this experiment.

Figure 4.3 shows cherry-picked images generated from StyleGAN and PresGAN. We can observe that the PresGAN maintains as good perceptual quality as the base architecture. In addition, we also observed that the StyleGAN tends to produce some redundant images (these are not shown in Figure 4.3), something that we did not observe with the PresGAN. This lack of diversity was also reflected in the FID scores which were $14.72 \pm 0.09$ for StyleGAN and $12.15 \pm 0.09$ for PresGAN. These results suggest that entropy regularization effectively reduces mode collapse while preserving sample quality.

[3]See https://github.com/NVlabs/ffhq-dataset.

### 4.3.4 Assessing held-out predictive log-likelihood

In this section we evaluate PresGANs for generalization using predictive log-likelihood. We use the DCGAN architecture to build PresGAN and evaluate the log-likelihood on two benchmark datasets, MNIST and CIFAR-10. We use images of size $32 \times 32$.

We compare the generalization performance of the PresGAN against the VAE (Kingma & Welling, 2013; Rezende et al., 2014) by controlling for the architecture and the evaluation procedure. In particular, we fit a VAE that has the same decoder architecture as the PresGAN. We form the VAE encoder by using the same architecture as the DCGAN discriminator and getting rid of the output layer. We used linear maps to get the mean and the log-variance of the approximate posterior.

To measure how PresGANs compare to traditional GANs in terms of log-likelihood, we also fit a PresGAN with $\lambda = 0$.

Consider an unseen datapoint $\mathbf{x}^*$. We estimate its log marginal likelihood $\log p_\theta(\mathbf{x}^*)$ using importance sampling,

$$\log p_\theta(\mathbf{x}^*) \approx \log\left(\frac{1}{S}\sum_{s=1}^{S}\frac{p_\theta\left(\mathbf{x}^* \mid \mathbf{z}^{(s)}\right)\cdot p\left(\mathbf{z}^{(s)}\right)}{r\left(\mathbf{z}^{(s)} \mid \mathbf{x}^*\right)}\right), \tag{4.32}$$

where we draw $S$ samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)}$ from a proposal distribution $r(\mathbf{z} \mid \mathbf{x}^*)$.

There are different ways to form a good proposal $r(\mathbf{z} \mid \mathbf{x}^*)$, and we discuss several alternatives in Section 4.5.1 of the appendix. In this paper, we take the following approach. We define the proposal as a Gaussian distribution,

$$r(\mathbf{z} \mid \mathbf{x}^*) = \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r). \tag{4.33}$$

We set the mean parameter $\boldsymbol{\mu}_r$ to the *maximum a posteriori* solution, i.e.,

$$\boldsymbol{\mu}_r = \arg\max_z \left(\log p_\theta\left(\mathbf{x}^* \mid \mathbf{z}\right) + \log p\left(\mathbf{z}\right)\right).$$

We initialize this maximization algorithm using the mean of a pre-fitted encoder, $q_\gamma(\mathbf{z} \mid \mathbf{x}^*)$. The encoder is fitted by minimizing the reverse KL divergence between $q_\gamma(\mathbf{z} \mid \mathbf{x})$ and the true posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$ using the training data. This KL is

$$\text{KL}\left(q_\gamma(\mathbf{z} \mid \mathbf{x}) || p_\theta(\mathbf{z} \mid \mathbf{x})\right) = \log p_\theta(\mathbf{x}) - \mathbb{E}_{q_\gamma(\mathbf{z} \mid \mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) - \log q_\gamma(\mathbf{z} \mid \mathbf{x})\right]. \tag{4.34}$$

Because the generative distribution is fixed at test time, minimizing the KL here is equivalent to maximizing the second term in Eq. 4.34, which is the ELBO objective of VAEs.

We set the proposal covariance $\boldsymbol{\Sigma}_r$ as an overdispersed version[4] of the encoder's covariance matrix, which is diagonal. In particular, to obtain $\boldsymbol{\Sigma}_r$ we multiply the elements of the encoder's covariance by a factor $\gamma$. In our experiments we set $\gamma$ to 1.2.

We use $S = 2{,}000$ samples to form the importance sampling estimator. Since the pixel values are normalized in $[-1, +1]$, we use a truncated Gaussian likelihood for evaluation. Specifically, for each pixel of the test image, we divide the Gaussian likelihood by the probability (under the generative model) that the pixel is within the interval $[-1, +1]$. We use the truncated Gaussian likelihood at test time only.

**Settings.** For the PresGAN, we use the same HMC hyperparameters as for the previous experiments. We constrain the variance of the generative distribution using $\boldsymbol{\sigma}_{\text{low}} = 0.001$ and $\boldsymbol{\sigma}_{\text{high}} = 0.2$. We use the default DCGAN values for the remaining hyperparameters, including the optimization settings. For the CIFAR-10 experiment, we choose $\lambda = 0.001$. We set all learning rates to 0.0002. We set the dimension of the latent variables to 100. We ran both the VAE and the PresGAN for a maximum of 200 epochs. For MNIST, we use the same settings as for CIFAR-10 but use $\lambda = 0.0001$ and ran all methods for a maximum of 50 epochs.

**Results.** Table 4.5 summarizes the results. Here GAN denotes the PresGAN fitted using $\lambda = 0$. The VAE outperforms both the GAN and the PresGAN on both MNIST and CIFAR-10. This is

[4] In general, overdispersed proposals lead to better importance sampling estimates.

*Table 4.5.* Generalization performance as measured by negative log-likelihood (lower is better) on MNIST and CIFAR-10. Here the GAN denotes a PresGAN fitted without entropy regularization ($\lambda = 0$). The PresGAN reduces the gap in performance between the GAN and the VAE on both datasets.

| | MNIST | | CIFAR-10 | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| VAE | −3483.94 | −3408.16 | −1978.91 | −1665.84 |
| GAN | −1410.78 | −1423.39 | −572.25 | −569.17 |
| PresGAN | −1418.91 | −1432.50 | −1050.16 | −1031.70 |

unsurprising given VAEs are fitted to maximize log-likelihood. The GAN's performance on CIFAR-10 is particularly bad, suggesting it suffered from mode collapse. The PresGAN, which mitigates mode collapse achieves significantly better performance than the GAN on CIFAR-10. To further analyze the generalization performance, we also report the log-likelihood on the training set in Table 4.5. We can observe that the difference between the training log-likelihood and the test log-likelihood is very small for all methods.

## 4.4 Appendix

### 4.5.1 Other ways to compute predictive log-likelihood

Here we discuss different ways to obtain a proposal in order to approximate the predictive log-likelihood. For a test instance $\mathbf{x}^*$, we estimate the marginal log-likelihood $\log p_\theta(\mathbf{x}^*)$ using importance sampling,

$$\log p_\theta(\mathbf{x}^*) \approx \log\left(\frac{1}{S}\sum_{s=1}^{S}\frac{p_\theta\left(\mathbf{x}^* \mid \mathbf{z}^{(s)}\right)\; p\left(\mathbf{z}^{(s)}\right)}{r\left(\mathbf{z}^{(s)} \mid \mathbf{x}^*\right)}\right), \tag{4.35}$$

where we draw the $S$ samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)}$ from a proposal distribution $r(\mathbf{z} \mid \mathbf{x}^*)$. We next discuss different ways to form the proposal $r(\mathbf{z} \mid \mathbf{x}^*)$.

One way to obtain the proposal is to set $r(\mathbf{z} \mid \mathbf{x}^*)$ as a Gaussian distribution whose mean and variance are computed using samples from an HMC algorithm with stationary distribution $p_\theta(\mathbf{z} \mid \mathbf{x}^*) \propto p_\theta(\mathbf{x}^* \mid \mathbf{z})p(\mathbf{z})$. That is, the mean and variance of $r(\mathbf{z} \mid \mathbf{x}^*)$ are set to the empirical mean and variance

of the HMC samples.

The procedure above requires to run an HMC sampler, and thus it may be slow. We can accelerate the procedure with a better initialization of the HMC chain. Indeed, the second way to evaluate the log-likelihood also requires the HMC sampler, but it is initialized using a mapping $\mathbf{z} = g_\eta(\mathbf{x}^\star)$. The mapping $g_\eta(\mathbf{x}^\star)$ is a network that maps from observed space $\mathbf{x}$ to latent space $\mathbf{z}$. The parameters $\eta$ of the network can be learned at test time using generated data. In particular, $\eta$ can be obtained by generating data from the fitted generator of PresGAN and then fitting $g_\eta(\mathbf{x}^\star)$ to map $\mathbf{x}$ to $\mathbf{z}$ by maximum likelihood. This is, we first sample $M$ pairs $(\mathbf{z}_m, \mathbf{x}_m)_{m=1}^M$ from the learned generative distribution and then we obtain $\eta$ by minimizing $\sum_{m=1}^M ||\mathbf{z}_m - g_\eta(\mathbf{x}_m)||_2^2$. Once the mapping is fitted, we use it to initialize the HMC chain.

A third way to obtain the proposal is to learn an encoder network $q_\eta(\mathbf{z} \mid \mathbf{x})$ jointly with the rest of the PresGAN parameters. This is effectively done by letting the discriminator distinguish between pairs $(\mathbf{x}, \mathbf{z}) \sim p_d(\mathbf{x}) \cdot q_\eta(\mathbf{z} \mid \mathbf{x})$ and $(\mathbf{x}, \mathbf{z}) \sim p_\theta(\mathbf{x}, \mathbf{z})$ rather than discriminate $\mathbf{x}$ against samples from the generative distribution. These types of discriminator networks have been used to learn a richer latent space for GAN (Donahue et al., 2016; Dumoulin et al., 2016). In such cases, we can use the encoder network $q_\eta(\mathbf{z} \mid \mathbf{x})$ to define the proposal, either by setting $r(\mathbf{z} \mid \mathbf{x}^*) = q_\eta(\mathbf{z} \mid \mathbf{x}^*)$ or by initializing the HMC sampler at the encoder mean.

The use of an encoder network is appealing but it requires a discriminator that takes pairs $(\mathbf{x}, \mathbf{z})$. The approach that we follow in the paper also uses an encoder network but keeps the discriminator the same as for the base DCGAN. We found this approach to work better in practice. More in detail, we use an encoder network $q_\eta(\mathbf{z} \mid \mathbf{x})$; however the encoder is fitted at test time by maximizing the variational ELBO, given by $\sum_n \mathbb{E}_{q_\eta(\mathbf{z}_n \mid \mathbf{x}_n)} \Big[\log p_\theta(\mathbf{x}_n, \mathbf{z}_n) - \log q_\eta(\mathbf{z}_n \mid \mathbf{x}_n)\Big]$. We set the proposal $r(\mathbf{z} \mid \mathbf{x}^*) = q_\eta(\mathbf{z} \mid \mathbf{x}^*)$. (Alternatively, the encoder can be used to initialize a sampler.)

### 4.5.2 Assessing mode collapse under increased data imbalance

In the main paper we show that mode collapse can happen not only when there are increasing number of modes, as done in the GAN literature, but also when the data is imbalanced. We consider a perfectly balanced version of MNIST by using 5,000 training examples per class, totalling 50,000 training examples. We refer to this original balanced dataset as **D**1. We build nine additional training sets from this balanced dataset. Each additional training set **D**$k$ leaves only 5 training examples for each class $j < k$. See Table 4.6 for all the class distributions.

*Table 4.6.* Class distributions using the MNIST dataset. There are 10 class—one class for each of the 10 digits in MNIST. The distribution D1 is uniform and the other distributions correspond to different imbalance settings as given by the proportions in the table. Note these proportions might not sum to one exactly because of rounding.

| Dist | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| D1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| D2 | $10^{-3}$ | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 |
| D3 | $10^{-3}$ | $10^{-3}$ | 0.12 | 0.12 | 0.12 | 0.12 | 0.12 | 0.12 | 0.12 | 0.12 |
| D4 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.14 | 0.14 | 0.14 | 0.14 | 0.14 | 0.14 | 0.14 |
| D5 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 |
| D6 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.20 | 0.20 | 0.20 | 0.20 | 0.20 |
| D7 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.25 | 0.25 | 0.25 | 0.25 |
| D8 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.33 | 0.33 | 0.33 |
| D9 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.49 | 0.49 |
| D10 | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | 0.99 |

# Conclusion

Probabilistic graphical modeling with latent variables provides a useful framework for learning from data. It enables accounting for uncertainty, learning the latent structure underlying data in an interpretable way, and incorporating prior knowledge. However probabilistic graphical modeling might lack flexibility for the purpose of learning from the types of high-dimensional complex data we currently encounter in practice. This thesis developed deep probabilistic graphical modeling, which leverages deep learning to bring flexibility to probabilistic graphical modeling. We used neural networks to extend the canonical EF-PCA to model and learn interpretable quantities from image and text data. We leveraged recurrent neural networks to build a class of models for sequential data where long-term dependencies are accounted for using latent variables. We solved several problems that probabilistic topic models suffer from using distributed representations of words for model specification and neural networks for inference. This thesis also made contributions on the algorithmic front. We developed reweighted expectation maximization (REM), an algorithm that unifies several existing maximum likelihood-based algorithms for learning models parameterized by deep neural networks. This unifying view is made possible using expectation maximization, a canonical inference algorithm for probabilistic graphical models. REM leads to better generalization to unseen data. Finally, we showed how to leverage the learning procedure behind generative adversarial networks to fit probabilistic latent-variable models. This new algorithm, called entropy-regularized adversarial learning, constitutes a solution to the mode collapse

problem that is pervasive in generative adversarial networks.

There are several choice points for deep probabilistic graphical modeling, each of which can be explored for future work.

1. **Prior.** Choosing a prior pertains to specifying our a priori knowledge of the latent structure. Several of the model classes we developed above used simple priors. Future work will explore how to devise richer priors for deep probabilistic graphical modeling. We will also explore how to translate domain knowledge into prior specification to apply deep probabilistic graphical modeling to new domains (e.g. science.)

2. **Likelihood.** We leveraged neural networks or word embeddings to define the conditional distribution of the data given the latent variables as an exponential family. The exponential family provides an umbrella distribution for the types of data we encounter in practice (e.g. real-valued, categorical, and binary.) Future work can explore other distributional forms for the likelihood (e.g. distributions specified via a sampling procedure) or use constrained neural networks to parameterize the likelihood (e.g. invertible neural networks.)

3. **Posterior.** We used variational inference as a framework for inferring the posterior distribution of the latent variables. In particular, we used distributions that are amenable to reparameterization such as the Gaussian, and parameterized them using neural networks. Future work can explore other choices of approximate posterior distributions, especially for discrete latent variables often used in probabilistic graphical modeling.

4. **Algorithm.** We explored both maximum likelihood and adversarial learning for model fitting. These two paradigms are complementary. Adversarial learning favors high quality of simulation of new data while maximum likelihood favors high held-out likelihood on unseen data. In future work we will explore how we can combine the strength of these two approaches to achieve all aspects of generalization as described in our desiderata. We will also explore how to make entropy-regularized adversarial learning amenable to discrete data and discrete latent variables.

# Bibliography


Aitchison, J. and Shen, S. Logistic normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, August 1980.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 224–232. JMLR. org, 2017.

Arora, S., Risteski, A., and Zhang, Y. Do gans learn the distribution? some theory and empirics. 2018.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, 2018.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bamler, R. and Mandt, S. Dynamic word embeddings. In *International Conference on Machine Learning*, 2017.

Batmanghelich, K., Saeedi, A., Narasimhan, K., and Gershman, S. Nonparametric spherical topic modeling with word embeddings. In *Association for Computational Linguistics*, 2016.

Baturo, A., Dasandi, N., and Mikhaylov, S. Understanding state preferences with text as data: introducing the UN general debate corpus. *Research & Politics*, 4:1–9, jun 2017.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. Neural probabilistic language models. In *Innovations in Machine Learning*, pp. 137–186. Springer, 2006.

Bhadury, A., Chen, J., Zhu, J., and Liu, S. Scaling up dynamic topic models. In *International World Wide Web Conference*, 2016.

Bird, S., Dale, R., Dorr, B., Gibson, B., Joseph, M., Kan, M.-Y., Lee, D., Powley, B., Radev, D., and Tan, Y. F. The ACL anthology reference corpus: a reference dataset for bibliographic research in computational linguistics. In *International Conference on Language Resources and Evaluation*, 2008.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7 (1):108–116, 1995.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Blei, D. M. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

Blei, D. M. and Lafferty, J. D. Dynamic topic models. In *International Conference on Machine Learning*, 2006.

Blei, D. M. and Lafferty, J. D. A correlated topic model of Science. *The Annals of Applied Statistics*, 1(1):17–35, 2007.

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

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017a.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017b.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Boyd-Graber, J., Hu, Y., and Mimno, D. Applications of topic models. *Foundations and Trends in Information Retrieval*, 11(2–3):143–296, 2017.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Bunk, S. and Krestel, R. WELDA: enhancing topic models by incorporating local word context. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, pp. 293–302. ACM, 2018.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *Proceedings of International Conference on Learning Representations*, 2015a.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015b.

Cao, Y., Ding, G. W., Lui, K. Y.-C., and Huang, R. Improving gan training via binarized representation entropy (bre) regularization. *arXiv preprint arXiv:1805.03644*, 2018.

Card, D., Tan, C., and Smith, N. A. A neural framework for generalized topic models. In *arXiv:1705.09296*, 2017.

Casella, G. and Robert, C. P. Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Collins, M., Dasgupta, S., and Schapire, R. E. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pp. 617–624, 2002.

Cong, Y., Chen, B., Liu, H., and Zhou, M. Deep latent Dirichlet allocation with topic-layer-adaptive stochastic gradient Riemannian MCMC. In *International Conference on Machine Learning*, 2017.

Cremer, C., Morris, Q., and Duvenaud, D. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

Das, R., Zaheer, M., and Dyer, C. Gaussian LDA for topic models with word embeddings. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Dieng, A. B. and Paisley, J. Reweighted expectation maximization. *arXiv preprint arXiv:1906.05850*, 2019.

Dieng, A. B., Wang, C., Gao, J., and Paisley, J. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

Dieng, A. B., Tran, D., Ranganath, R., Paisley, J., and Blei, D. Variational inference via chi upper bound minimization. In *Advances in Neural Information Processing Systems*, pp. 2732–2741, 2017.

Dieng, A. B., Cho, K., Blei, D. M., and LeCun, Y. Learning with reflective likelihoods. 2018a.

Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018b.

Dieng, A. B., Ranganath, R., Altosaar, J., and Blei, D. M. Noisin: Unbiased regularization for recurrent neural networks. *arXiv preprint arXiv:1805.01500*, 2018c.

Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. Avoiding latent variable collapse with generative skip models. *Artificial Intelligence and Statistics*, 2019a.

Dieng, A. B., Ruiz, F. J., and Blei, D. M. The dynamic embedded topic model. *arXiv preprint arXiv:1907.05545*, 2019b.

Dieng, A. B., Ruiz, F. J., and Blei, D. M. Topic modeling in embedding spaces. *arXiv preprint arXiv:1907.04907*, 2019c.

Dieng, A. B., Ruiz, F. J., Blei, D. M., and Titsias, M. K. Prescribed generative adversarial networks. *arXiv preprint arXiv:1910.04302*, 2019d.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Donoho, D. and Stodden, V. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, pp. 1141–1148, 2004.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Gao, Y., Archer, E. W., Paninski, L., and Cunningham, J. P. Linear dynamical neural population models through nonlinear embeddings. In *Advances in neural information processing systems*, pp. 163–171, 2016.

Gershman, S. J. and Goodman, N. D. Amortized inference in probabilistic reasoning. In *Annual Meeting of the Cognitive Science Society*, 2014.

Ghahramani, Z. and Beal, M. J. Propagation algorithms for variational bayesian learning. In *Advances in neural information processing systems*, pp. 507–513, 2001.

Givens, G. H. and Hoeting, J. A. *Computational statistics*, volume 703. John Wiley & Sons, 2012.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Grover, A., Dhar, M., and Ermon, S. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Guu, K., Hashimoto, T. B., Oren, Y., and Liang, P. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878*, 2017.

Harris, Z. S. Distributional structure. *Word*, 10(2-3):146–162, 1954.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016b.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *Proceedings of International Conference on Learning Representations*, 2017.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoffman, M. D. Learning deep latent gaussian models with markov chain monte carlo. In *Proceedings of International Conference on Machine Learning*, pp. 1510–1519, 2017.

Hoffman, M. D. and Johnson, M. J. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, May 2013.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Jähnichen, P., Wenzel, F., Kloft, M., and Mandt, S. Scalable generalized dynamic topic models. In *Artificial Intelligence and Statistics*, 2018.

Jaynes, E. T. *Probability theory: The logic of science*. Cambridge university press, 2003.

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R., and Adams, R. P. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, 2016.

Jordan, M. I. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Ba, J. L. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Koller, D., Friedman, N., and Bach, F. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Kumar, R., Goyal, A., Courville, A., and Bengio, Y. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

Lafferty, J. D. and Blei, D. M. Correlated topic models. In *Advances in Neural Information Processing Systems*, 2005.

Lake, B. M., Salakhutdinov, R. R., and Tenenbaum, J. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pp. 2526–2534, 2013.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

Lau, J. H., Newman, D., and Baldwin, T. Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Le, Q. and Mikolov, T. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 2014a.

Le, Q. and Mikolov, T. Distributed representations of sentences and documents. In *International conference on machine learning*, pp. 1188–1196, 2014b.

Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*, 2017.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

Lefebure, L. Exploring the UN general debates with dynamic topic models. Available online at https://towardsdatascience.com, 2018.

Levy, O. and Goldberg, Y. Neural word embedding as implicit matrix factorization. In *Neural Information Processing Systems*, pp. 2177–2185, 2014.

Lin, Z., Khetan, A., Fanti, G., and Oh, S. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 1498–1507, 2018.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Maaten, L., Chen, M., Tyree, S., and Weinberger, K. Learning with marginalized corrupted features. In *International Conference on Machine Learning*, pp. 410–418, 2013.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Mcauliffe, J. D. and Blei, D. M. Supervised topic models. In *Advances in neural information processing systems*, pp. 121–128, 2008.

Mescheder, L., Nowozin, S., and Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2391–2400. JMLR. org, 2017.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Miao, Y., Yu, L., and Blunsom, P. Neural variational inference for text processing. In *International conference on machine learning*, pp. 1727–1736, 2016.

Mikolov, T. and Zweig, G. Context dependent recurrent neural network language model. *SLT*, 12: 234–239, 2012.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 3, 2010.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *ICLR Workshop Proceedings. arXiv:1301.3781*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*, 2013b.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. Optimizing semantic coherence in topic models. In *Conference on Empirical Methods in Natural Language Processing*, 2011.

Minka, T. et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

Miyato, T., Dai, A. M., and Goodfellow, I. Adversarial training methods for semi-supervised text classification. *stat*, 1050:7, 2016.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv:1610.03483*, 2016.

Moody, C. E. Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv:1605.02019*, 2016.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2 (11):2, 2011.

Nguyen, D. Q., Billingsley, R., Du, L., and Johnson, M. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313, 2015.

Owen, A. B. Monte Carlo theory, methods and examples. Book in preparation, 2013.

Paisley, J. W., Blei, D. M., and Jordan, M. I. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, 2012.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing*, volume 14, pp. 1532–1543, 2014.

Petterson, J., Buntine, W., Narayanamurthy, S. M., Caetano, T. S., and Smola, A. J. Word features for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pp. 1921–1929, 2010.

Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ranganath, R., Gerrish, S., and Blei, D. M. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rigollet, P. and Weed, J. Entropic optimal transport is maximum-likelihood deconvolution. *Comptes Rendus Mathématique*, 356(11-12):1228–1235, 2018.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. The author-topic model for authors and documents. In *Uncertainty in Artificial Intelligence*, 2004.

Rudolph, M. and Blei, D. M. Dynamic embeddings for language evolution. In *International World Wide Web Conference*, 2018.

Rumelhart, D. and Abrahamson, A. A model for analogical reasoning. *Cognitive Psychology*, 5 (1):1–28, 1973.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Saul, L. K. and Jordan, M. I. Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems*, 1996.

Saul, L. K., Jaakkola, T., and Jordan, M. I. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76, 1996.

Schein, A., Paisley, J., Blei, D. M., and Wallach, H. Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1045–1054, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Shi, B., Lam, W., Jameel, S., Schockaert, S., and Lai, K. P. Jointly learning word embeddings and latent topics. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016a.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 2016b.

Soofi, E. S. Principal information theoretic approaches. *Journal of the American Statistical Association*, 95(452):1349–1353, 2000.

Srivastava, A. and Sutton, C. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*, 2017.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Tieleman, T. and Hinton, G. Lecture 6.5-RMSPROP: divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 4, 2012.

Tipping, M. E. and Bishop, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational bayes for non-conjugate inference. In *International conference on machine learning*, pp. 1971–1979, 2014.

Titsias, M. K. and Lázaro-Gredilla, M. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

Titsias, M. K. and Ruiz, F. J. Unbiased implicit variational inference. *arXiv preprint arXiv:1808.02078*, 2018.

Tomczak, J. M. and Welling, M. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

Turner, R., Hung, J., Saatci, Y., and Yosinski, J. Metropolis-hastings generative adversarial networks. *arXiv preprint arXiv:1811.11357*, 2018.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, 2016.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6309–6318, 2017.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Wainwright, M. J., Jordan, M. I., et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. Evaluation methods for topic models. In *International Conference on Machine Learning*, 2009.

Wang, C. and Blei, D. M. Collaborative topic modeling for recommending scientific articles. In *ACM SIGKDD*, 2011.

Wang, C., Blei, D. M., and Heckerman, D. Continuous time dynamic topic models. In *Uncertainty in Artificial Intelligence*, 2008.

Wang, X. and McCallum, A. Topics over time: a non-Markov continuous-time model of topical trends. In *ACM SIGKDD*, 2006.

Wen, T.-H. and Luong, M.-T. Latent topic conversational models. *arXiv preprint arXiv:1809.07070*, 2018.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

Xiao, C., Ma, T., Dieng, A. B., Blei, D. M., and Wang, F. Readmission prediction via deep contextual embedding of clinical concepts. *PloS one*, 13(4), 2018a.

Xiao, C., Zhong, P., and Zheng, C. BourGAN: Generative networks with metric embeddings. In *Advances in Neural Information Processing Systems*, 2018b.

Xie, P., Yang, D., and Xing, E. Incorporating word correlation knowledge into topic modeling. In *Conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

Xu, H., Wang, W., Liu, W., and Carin, L. Distilled Wasserstein learning for word embedding and topic modeling. In *Advances in Neural Information Processing Systems*, 2018.

Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*, 2018.

Xun, G., Gopalakrishnan, V., Ma, F., Li, Y., Gao, J., and Zhang, A. Topic discovery for short texts using word embeddings. In *International Conference on Data Mining*, 2016.

Xun, G., Li, Y., Zhao, W. X., Gao, J., and Zhang, A. A correlated topic model using word embeddings. In *IJCAI*, pp. 4207–4213, 2017.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of International Conference on Machine Learning*, 2017.

Yeung, S., Kannan, A., Dauphin, Y., and Fei-Fei, L. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

Yin, M. and Zhou, M. Semi-implicit generative model. *arXiv preprint arXiv:1905.12659*, 2019.

Zhang, H., Chen, B., Guo, D., and Zhou, M. WHAI: Weibull hybrid autoencoding inference for deep topic modeling. In *International Conference on Learning Representations*, 2018.

Zhang, P., Liu, Q., Zhou, D., Xu, T., and He, X. On the discrimination-generalization tradeoff in gans. *arXiv preprint arXiv:1711.02771*, 2017.

Zhao, H., Du, L., and Buntine, W. A word embeddings informed focused topic model. In *Asian Conference on Machine Learning*, 2017a.

Zhao, H., Du, L., Buntine, W., and Liu, G. MetaLDA: A topic model that efficiently incorporates meta information. In *International Conference on Data Mining*, 2017b.

Zhao, S., Song, J., and Ermon, S. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017c.