When are equilibrium networks scoring algorithms?

Russell Tsuchida
Data61, CSIRO, Australia
russell.tsuchida@data61.csiro.au

Cheng Soon Ong
Data61, CSIRO
and Australian National University
cheng-soon.ong@data61.csiro.au

Abstract

Principal Component Analysis (PCA) and its exponential family extensions have three components: observed variables, latent variables and parameters of a linear transformation. The likelihood of the observation is an exponential family with canonical parameters that are a linear transformation of the latent variables. We show how joint maximum a-posteriori (MAP) estimates can be computed using a deep equilibrium model that computes roots of the score function. Our analysis provides a systematic way to relate neural network activation functions back to statistical assumptions about the observations. Our layers are implicitly differentiable, and can be fine-tuned in downstream tasks, as demonstrated on a synthetic task.

1 Introduction and background

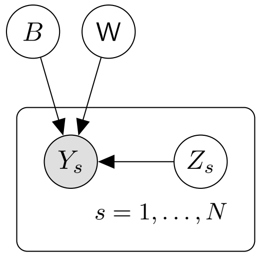
Deep learning provides a means of fitting flexible function classes to data. Parameters of layers are typically jointly adjusted by applying first-order optimisation methods to an objective involving the data and the parameters [Deisenroth et al., 2020, §5.6.1]. Layers may provide an explicit description of a mapping. For example, given an input $x \in \mathbb{R}^D$, the L-dimensional output of a layer with parameters $\theta \in \mathbb{R}^{L\times D}$ might be defined by $\sigma\big(\theta x\big)$ for some function σ applied element-wise.

Alternatively, layers may be defined implicitly as solutions to certain problems. Given an input $x \in \mathbb{R}^D$, a Deep Equilibrium Model (DEQ) [Bai et al., 2019] outputs a solution to the fixed point equation $z = \sigma\big(\gamma z + \theta x\big)$, where $\theta \in \mathbb{R}^{L\times D}$ and $\gamma \in \mathbb{R}^{L\times L}$ are parameters and $z \in \mathbb{R}^L$ is the implicitly defined output. Implicit layers include DEQs which solve fixed point equations, Neural ODEs [Chen et al., 2018] which solve ODEs, and Deep Declarative Networks (DDNs) [Gould et al., 2021] which solve optimisation problems.

Computing roots of the score function — the derivative of the log density/mass function with respect to the parameter — is one of the most classical and generally applicable methods for finding point estimates of parameters [Stigler, 2007]. Fixed points $z^* = f(z^*)$ are equivalent to roots $f(z^*) - z^* = 0$, so roots of score functions can be computed using DEQ layers. An interesting question to consider is: *When are equilibrium networks scoring algorithms?*

Contributions. We present a DEQ that solves the problem of joint MAP estimation in a graphical model representing exponential family PCA (Figure 1). Conveniently, we recover neural network architectures with commonly used activations such as $\tanh$, logistic sigmoid and softmax. See Table 1. This enables us to relate statistical assumptions about the data to a choice of activation function. Our analysis provides a bottom-up justification for the use of particular DEQ layers in unsupervised learning settings. Our DEQ layers compute solutions to strongly convex optimisation problems. Hence they are guaranteed to admit unique fixed points. See Theorem 1. We provide an implementation [1] and compare it to PCA, tSNE and UMAP on some synthetic datasets. See Figure 3.

[1]https://github.com/RussellTsuchida/ped/



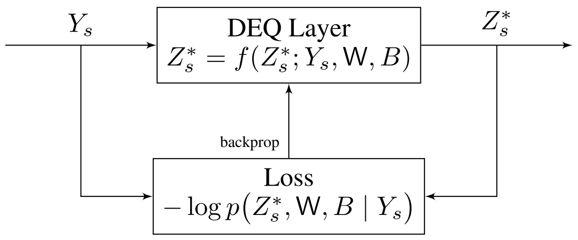

Y_s
DEQ Layer
$Z_s^* = f(Z_s^*; Y_s, \mathsf{W}, B)$
Z_s^*
backprop
Loss
$-\log p\big(Z_s^*, \mathsf{W}, B \mid Y_s\big)$

Figure 1: (Left) Graphical model for exponential family PCA with observations Y_s, latents Z_s, and parameters W, B. The likelihood of the observation Y_s is an exponential family with a canonical parameter $\mathsf{W}Z_s + B$. (Right) MAP estimates can be found using a DEQ. The latents are estimated by the prediction of the DEQ layer given parameters of the DEQ layer, adjusted through backpropagation.

Exponential families. Recall the exponential family (see Appendix A.1), which is a class of probability distributions whose sufficient statistics admit a moment generating function defined on an open set, which we take to be $\mathbb{R}$. The probability density (mass) function

$$p(y \mid v) = h(y) \exp\big(vT(y) - A(v)\big) \tag{1}$$

belongs to a minimal regular exponential family with canonical parameter v, log-partition function A, sufficient statistic T and base density (mass function) h. A is both infinitely differentiable and strictly convex [Wainwright and Jordan, 2008, Proposition 3.1]. We write $y \sim \mathcal{D}_{A,\chi}(v)$ if a scalar-valued random variable y follows an exponential family with log partition function A, canonical parameter v and additional parameter χ. If $Y = (y_1, \ldots, y_d)$ and $V = (v_1, \ldots, v_d)$, we write $Y \sim \mathcal{D}_{A,\chi}(V)$ to mean $y_i \sim \mathcal{D}_{A,\chi}(v_i)$ for all $1 \leq i \leq d$, with each y_i mutually conditionally independent given V.

Matrices and data. Let $Y_s = (y_{s1}; \ldots; y_{sd}) \sim \mathcal{D}_{A,\chi}(V_s)$, where for each s, $V_s \in \mathbb{R}^d$ is a vector consisting of canonical parameters. Let $\mathsf{Y} \in \mathbb{R}^{d \times N}$ denote a matrix with N such vectors Y_s, $1 \leq s \leq N$, each sampled conditionally independently given the canonical parameters. Each of the Y_s are associated with a latent representation $Z_s \in \mathbb{R}^l$. Write $\mathsf{Z} \in \mathbb{R}^{l \times N}$ for the matrix consisting of such latent representations. Scalars are lower-case, vectors are upper-case and matrices are upper-case sans-serif. Functions that accept scalars extend to vector and matrix inputs applied elementwise.

Scoring. The score S — $\frac{\partial \log p(Y|V)}{\partial V}$ or $\frac{\partial \log p(V|Y)}{\partial V}$ — is necessarily zero when evaluated at an MLE or MAP estimate V^*, for sufficiently smooth likelihoods. When the log density is convex, the score S being the zero vector is also sufficient. The Jacobian of the score S, called the observed information matrix. When using density (1), the Jacobian is independent of the data Y and coincides with $\mathbb{E}[SS^\top]$— the Fisher information matrix [Stigler, 2007, §7]. Fisher's scoring method is essentially Newton's method for the special case of finding the root of S using the Fisher information matrix as the Hessian. Fisher's scoring method using exponential families appears in supervised and unsupervised contexts as generalised linear models [McCullagh and Nelder, 1989] and exponential family PCA [Collins et al., 2001] respectively. The latter problem is only convex in one of the latents or parameters but not both.

Implicit neural networks. We work with a prototype problem to represent both DEQs and DDNs

$$\min_{\theta,\gamma} \sum_{s=1}^{N} \mathcal{L}\Big(\mathcal{F}_\gamma\big(Z_s^*, Y_s\big), Y_s\Big) \qquad \text{subject to } Z_s^* = \text{SolveProblem}(\theta, Y_s),\ s = 1, \ldots, N. \tag{2}$$

Here θ are parameters of the implicit layer, Z_s^* are outputs of the implicit layer, $\mathcal{F}_\gamma$ is a neural network (that may also contain implicit layers) parameterised by γ, and $\mathcal{L}$ is a loss function. SolveProblem defines and solves an optimisation or fixed point problem depending on parameters θ and input data Y_s. Connections between DDNs and DEQs have been explored [Tsuchida et al., 2022, Li et al., 2022]. The outer problem learns the parameters of the network, and the inner constraint outputs predictions of the implicit layer. The inner constraint is subject to questions of well-posedness, that is, whether solutions to the problem exist and are unique. Following usual deep learning philosophy, we ignore such questions for the outer parameter learning problem, which is likely to be nonconvex. If a problem can be cast in the form of (2), the machinery of implicit deep learning can be used to solve the problem. Our aim is to cast problems in the form (2) with well-posed inner constraints.

$\mathbb{Y}$	$T(y)$	$A(\eta)$	$h(y)$	$\sigma(\eta) := A'(\eta)$
$\mathbb{R}$	y/b	$\eta^2/2$	$(2\pi)^{-1/2}$	η
$\{0,1\}$	y	$\log\big(1+\exp(\eta)\big)$	1	$\frac{\exp(\eta)}{\exp(\eta)+1}$
$\{-1,1\}$	y	$\log\cosh\eta$	$1/2$	$\tanh(\eta)$
$[0,1]$	y	$\log\frac{e^\eta-1}{\eta}$	1	$\frac{e^\eta(\eta-1)+1}{(e^\eta-1)\eta}$
$\{0,1,\ldots\}$	y	e^η	$(y!)^{-1}$	e^η

Table 1: Different exponential families induce different inverse link functions (which we call activation functions) σ. In order, each row represents a different exponential family: Gaussian, Bernoulli, non-interacting Ising, continuous Bernoulli [Loaiza-Ganem and Cunningham, 2019] and Poisson likelihoods respectively. See Appendix A and also Nielsen and Garcia [2009].

2 A DEQ layer that solves exponential family PCA

Setup. We consider the graphical model in Figure 1, which mirrors various versions of PCA [Bishop, 2006, Collins et al., 2001, Mohamed et al., 2008]. Let $\mathsf{H} = \mathsf{W}\mathsf{Z} + B1_{1\times N}$, where $\mathsf{W} \in \mathbb{R}^{d\times l}$, $B \in \mathbb{R}^d$ and $\mathsf{Z} \in \mathbb{R}^{l\times N}$. Equivalently $H_s = \mathsf{W}Z_s + B$ where $Z_s \in \mathbb{R}^{l\times 1}$ are the latent variables for data $Y_s = (y_{s1}, \ldots y_{sd})$. Typically $l < d$, although this need not be the case. The canonical parameters of the distribution of the observed variables Y_s are $H_s \in \mathbb{R}^d$. In other words, we assume that the observed vector of data Y_s is drawn from an exponential family $\mathcal{D}_{A,\mathcal{X}}(H_s)$. Following (1),

$$p(\mathsf{Y} \mid \mathsf{Z}, \mathsf{W}, B) = \prod_{s=1}^{N}\prod_{i=1}^{d} h\big(y_{si}\big)\exp\Big((\mathsf{W}Z_s + B)_i T(y_{si}) - A\big((\mathsf{W}Z_s + B)_i\big)\Big). \tag{3}$$

For each s, place independent zero mean iid standard Gaussian priors over Z_s. By Bayes' theorem, the posterior and a (not necessarily unique) MAP estimate are respectively

$$p(\mathsf{Z}, \mathsf{W}, B \mid \mathsf{Y}) = p(\mathsf{Y} \mid \mathsf{Z}, \mathsf{W}, B)p(\mathsf{Z})p(\mathsf{W}, B)/p(\mathsf{Y}), \quad \text{and} \tag{4}$$

$$\mathsf{Z}^*, \mathsf{W}^*, b^* \in \operatorname*{argmin}_{\mathsf{Z},\mathsf{W},B} p(\mathsf{Z}, \mathsf{W}, B \mid \mathsf{Y}). \tag{5}$$

Results. Since the log-posterior (4) is strongly convex in Z, first order optimality conditions are sufficient to characterise the unique global optima of the latents given a fixed set of parameters W, B. This yields a fixed point equation, which is a DEQ of the form (2) — see Appendix B for proof.

Theorem 1. *Any MAP estimate* (5) *of the graphical model in Figure 1 is a DEQ*

$$\mathsf{W}^*, B^* \in \operatorname*{argmin}_{\mathsf{W},B} -\log p(\mathsf{W}, B) - \log p(\mathsf{Z}^*) + \sum_{s=1}^{N} 1_{d\times 1}^\top A\left(\mathsf{W}Z_s^* + B\right) - T(Y_s)^\top(\mathsf{W}Z_s^* + B)$$

$$\textit{subject to } Z_s^* = \frac{1}{\lambda}\mathsf{W}^\top\big(T(Y_s) - \sigma(\mathsf{W}Z_s^* + B)\big), \quad \forall 1 \le s \le N. \tag{6}$$

Solutions to the inner constraint (6) *are guaranteed to exist and be unique.*

Constraint (6) is a fully connected (FC) DEQ layer with activation σ, shared weights W and biases B. While we are not aware of any published usage of DEQs applied to dimensionality reduction, (6) resembles what practitioners might use as a generic baseline in applying DEQs as black-box predictors, without any explicit statistical model or intention to compute with score functions. The outer optimisation is a non-Gaussian exponential family generalisation of a squared reprojection error.

3 Illustration on a low dimensional latent space

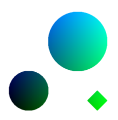


Figure 2: Z ground truth.

We compare performance and features of PCA, tSNE [Van der Maaten and Hinton, 2008], UMAP [McInnes et al., 2018] and DEQ. We focus on two issues.

Ground truth latents. We generate datasets through graphical model in Figure 1, except that Z is fixed to be as shown in Figure 2. For each distribution shown in the left column of Figure 3, we generate 100 parameters W according to the graphical model and them sample the corresponding Y. This results in 100 different datasets for each distribution, each with known ground truth Z.

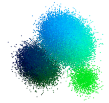
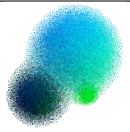
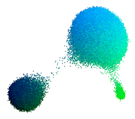
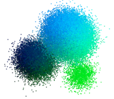
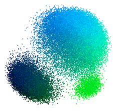
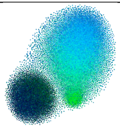
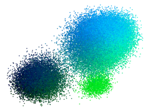
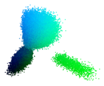
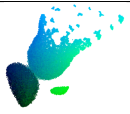
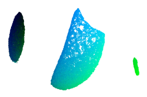
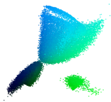
	PCA (sklearn)	tSNE	UMAP	Our DEQ
Gaussian $A(r) = r^2/2$ $R(\eta) = \eta$	 5	 -	 0	 95
Bernoulli $A(r) = \log\left(1 + e^r\right)$ $R(\eta) = \eta$	 6	 -	0	 94
Poisson $A(r) = \exp(r)$ $R(\eta) = \eta$	 0	 -	 16	 84

Figure 3: The number in each cell represents the number of times that model performed the best compared with others when used as the input for a downstream task over 100 random trials. Interestingly, PCA and DEQ produce visually similar results. However, DEQ is able to obtain better results on downstream tasks on account of its ability to be fine-tuned. While UMAP's results are visually pleasing, embeddings do not bear similarity with the ground truth, except in the Gaussian case. For non-Gaussian cases, UMAP places latents in good clusters, but fails to globally position the clusters accurately relative to one another. tSNE is incompatible in a pipeline with a downstream task. tSNE produces globally accurate positioning, but sometimes adds artefacts into the visualisation. Embeddings are an equivalence class up to a rotation and scaling, since W and Z are non-identifiable; we rotated images by hand. For DEQ, we visualise the latents in an orthogonal basis, i.e. shown are RZ, where $\mathsf{W} = \mathsf{QR}$ is a QR decomposition.

Downstream performance and deep learning compatibility. We can fine-tune backbone DEQ layers that are pretrained in an unsupervised manner on a supervised task with a head network. In contrast, since PCA is a linear mapping, composing it with a head network and optimising the result is equivalent to just using a head network. Other dimensionality reduction techniques that do not posses neural network parameters such as UMAP cannot benefit from fine-tuning. Since tSNE's mapping does not transfer from training to testing data it is incompatible in a pipeline with downstream tasks. We evaluate the performance of each dimensionality reduction algorithm by passing its output to a downstream task. Figure 3 shows the number of times each method performed the best over 100 random trials. See Appendix C for more details.

4 Discussion, conclusion and future work

We derived DEQ architectures that solve the problem of MAP estimation of exponential family PCA. Our DEQ layers compute unique roots of the score function (given a set of parameters W, B), and may be computed using Fisher scoring or otherwise. Our analysis grounds DEQ architectures in a probabilistic framework, and shows how certain architectural choices are implied by statistical assumptions on observations. As noted by Bai et al. [2019], DEQs built from contraction mappings may be understood as infinitely deep neural networks with shared weights. In this sense, our analysis motivates the use of such infinitely deep networks. In future, it might be interesting to extend our implementation to a Bayesian setting by taking the Hessian of the negative log likelihood evaluated at the MAP. Surprisingly, the Laplace approximation has recently been shown to be competitive with other Bayesian approximations in deep learning [Daxberger et al., 2021]. We found expressions for the Laplace approximation (see Appendix D), which may be useful for future investigations. Our canonical parameters $\mathsf{W}Z_s + B$ are linear transformations of latent variables, which recover fully connected and convolutional DEQ architectures. Empirical success stories of DEQs applied to language prediction and computer vision usually other deep learning elements such as activations that are not canonical inverse link functions (such as the ReLU), attention and/or multi-scale residual blocks [Bai et al., 2020]. In future, we hope to cast these elements in light of scoring methods.

References

Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Stephen Gould, Richard Hartley, and Dylan John Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Stephen M Stigler. The epic story of maximum likelihood. *Statistical Science*, pages 598–620, 2007.

M.J. Wainwright and M.I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1989. ISBN 9780412317606.

Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. A generalization of principal components analysis to the exponential family. *Advances in neural information processing systems*, 14, 2001.

Russell Tsuchida, Suk Yee Yong, Mohammad Ali Armin, Lars Petersson, and Cheng Soon Ong. Declarative nets that are equilibrium models. In *International Conference on Learning Representations*, 2022.

Mingjie Li, Yisen Wang, Xingyu Xie, and Zhouchen Lin. Optimization inspired multi-branch equilibrium models. In *International Conference on Learning Representations*, 2022.

Gabriel Loaiza-Ganem and John P Cunningham. The continuous bernoulli: fixing a pervasive error in variational autoencoders. *Advances in Neural Information Processing Systems*, 32, 2019.

Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Shakir Mohamed, Zoubin Ghahramani, and Katherine A Heller. Bayesian exponential family pca. *Advances in neural information processing systems*, 21, 2008.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29), 2018.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020.

RB Dingle. The Bose-Einstein integrals. *Applied Scientific Research, Section A*, 6(1):240–244, 1957.

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, Joydeep Ghosh, and John Lafferty. Clustering with bregman divergences. *Journal of machine learning research*, 6(10), 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

A Exponential family calculations

A.1 Construction of exponential families

Let $\mathbb{Y} \subseteq \mathbb{R}$ and form a measurable space $(\mathbb{Y}, \mathcal{B})$ for some sigma algebra $\mathcal{B}$. Let ω be some reference measure (nominally Lebesgue or counting) and let $h : \mathbb{Y} \to \mathbb{R}_{\geq 0}$ be some ω-integrable function, $\int_{\mathbb{Y}} h(y) d\omega(y) < \infty$. Let T be some measurable function and let $\mathbb{F} \subseteq \mathbb{R}$ be the set of all canonical parameters r such that $\int_{\mathbb{Y}} \exp\big(T(y)r\big)h(y)d\omega(y) < \infty$. Assume that $\mathbb{F}$ is open. Define the *log partition function* $A : \mathbb{F} \to \mathbb{R}$ by $A(r) = \log \int_{\mathbb{Y}} \exp\big(T(y)r\big)h(y)d\omega(y)$. Call

$$p(y \mid r) = h(y) \exp\big(rT(y) - A(r)\big) \tag{7}$$

the PDF of a minimal regular exponential family. For such a family, A is both infinitely differentiable and strictly convex [Wainwright and Jordan, 2008, Proposition 3.1]. Strictly convex and infinitely differentiable functions A are admissible if and only if $\exp\big(A(ir)\big)$ is a positive definite function. See Lemma 1, Appendix A.2. A acts as a cumulant generating function, so that in particular the expected value under $p(y \mid r)$ is $\mu = A'(r)$. The parameter μ is called the expectation parameter. Following common convention, we will use $p(\cdot \mid \cdot)$ to denote all (conditional) probability density functions, with a meaning clear from the arguments. We write $y \sim \mathcal{D}_{A,\chi}(r)$ to mean that a scalar-valued random variable y follows an exponential family with log partition function A, scalar canonical parameter r and additional parameter ν. If $Y = (y_1, \dots, y_d)$ and $V = (r_1, \dots, r_d)$, we write $Y \sim \mathcal{D}_{A,\chi}(V)$ to mean $y_i \sim \mathcal{D}_{A,\chi}(r_i)$ for all $1 \leq i \leq d$, with each y_i mutually conditionally independent given H.

A.2 How to check whether a log partition function is admissible

Lemma 1. *An infinitely differentiable and strictly convex A is an admissible log partition function for a minimal, regular, one-dimensional exponential family if and only if*

- *there exists some c such that $c \exp\big(A(r)\big)$ is the moment generating function of some random variable evaluated at $r \in \mathbb{F} \subseteq \mathbb{R}$.*
- *$\exp\big(A(ir)\big)$ is a positive definite function evaluated at $r \in \mathbb{R}$ and is continuous at $r = 0$.*

Proof. Define

$$q(y \mid r) = h(y) \exp\big(T(y)r - A(r)\big)$$

for some nonnegative h and $A : \mathbb{F} \to \mathbb{R}$ with $\mathbb{F} \subseteq \mathbb{R}$. For a given A, we would like to determine whether there exist an h such that $q(\cdot \mid r)$ defines a valid probability density function (or mass function). This is true if and only if $ch(\cdot)$ for some $c > 0$ is the probability density function (or probability mass function) of a random variable y satisfying

$$\begin{aligned} \frac{1}{c}\mathbb{E}_{y \sim ch(\cdot)}\big[\exp\big(T(y)r - A(r)\big)\big] &= 1 \\ \mathbb{E}_{y \sim ch(\cdot)}\big[\exp\big(T(y)r\big)\big] &= c\exp\big(A(r)\big), \end{aligned} \tag{8}$$

which is the moment generating function of a random variable $T(y)$. Alternatively, we may view $\phi(r) := c \exp\big(A(ir)\big)$ as the characteristic function of a random variable $T(y)$ evaluated at real value r. Choose $c = \exp\big(-A(0)\big)$. By Bochner's theorem, ϕ is a characteristic function if and only if is continuous at 0 and ϕ is positive definite. □

A.3 Antiderivative of hyperbolic tangent

Choose $A(r) = \int \tanh(r)\, dr = \log\cosh(r)$. One readily observes that A is admissible since $\exp\big(A(ir)\big) = \cos(r)$, which is positive definite.

Since the characteristic function of the base pdf is 2π-periodic, it coincides with a discrete integer-valued random variable. In fact, we can construct an exponential family supported on $\{-1, 1\}$ with this choice of A, $T(y) = y$ and uniform $h(y) = 1/2$ by verifying that (8) holds with $c = 1$. This is a non-interacting Ising model.

A.4 The Bose-Einstein integral

Let

$$B_j(r) = \frac{1}{\Gamma(j+1)} \int_0^\infty \frac{y^j}{e^{y-r}-1}\, dy, \quad j > -1, \quad r < 0$$

denote the complete Bose-Einstein integral of order j. It is known [Dingle, 1957] that $B_j(r) = \mathrm{Li}_{j+1}\big(e^r\big)$, where Li_{j+1} denotes the polylogarithm of order $j+1$. The Bose-Einstein integral satisfies a recursive relationship

$$\frac{d}{dr} B_j(r) = B_{j-1}(r),$$

with a closed-form initial value $B_1(r) = -\log(1-e^r)$.

Define

$$p(y \mid r) = h(y) \exp\big(yr - B_j(\beta r)\beta^{-(j+1)}\big),$$

where $h(y)$ is a nonnegative function and $\beta > 0$ is some fixed value. We would like to determine whether this choice of log partition function is valid.

Since the composition of the exponential function with a positive definite function is positive definite, it suffices to show that the Bose-Einstein integral $\mathrm{Li}_{j+1}\big(e^{ir}\big)$ is positive definite. This is observed by a comparison to the characteristic function of the Geometric distribution. We have that

$$\begin{aligned}
\int_0^\infty \frac{y^j}{e^{y-ir}-1}\, dy &= \int_0^\infty \frac{y^j e^{-y} e^{ir}}{1-e^{-y}e^{ir}}\, dy \\
&= \int_2^1 \frac{(p-1)\big(-\log(p-1)\big)^j e^{ir}}{1-(p-1)e^{ir}} \frac{-1}{p-1}\, dp, \qquad p = e^{-y}+1 \\
&= \int_1^2 \Big(\frac{(-1)^j \log(p-1)^j}{p}\Big) \frac{pe^{ir}}{1-(p-1)e^{ir}}\, dp.
\end{aligned}$$

Now observe that $\Big(\frac{(-1)^j \log(p-1)^j}{p}\Big)$ is positive in $(1,2)$, and $\frac{1}{1-(1-p)e^{ir}}$ admits a Fourier series (via the geometric series) $\sum_{k=0}^\infty (p-1)^k e^{irk}$. Since the Fourier series coefficients are positive, $\frac{1}{1-(p-1)e^{ir}}$ is positive definite, as is the product of positive definite functions $\frac{pe^{ir}}{1-(p-1)e^{ir}}$. The integrand is therefore positive definite, and by Lévy's continuity theorem the integral is a positive definite function.

Since ϕ is 2π-periodic, it is the characteristic function of a discrete random variable taking integer values.

A.5 A non-example in the Fermi-Dirac integral

The Fermi-Dirac integral is closely related to the Bose-Einstein integral. Let

$$B_j(r) = \frac{1}{\Gamma(j+1)} \int_0^\infty \frac{y^j}{e^{y-r}+1}\, dy, \quad j > -1, \quad r < 0$$

denote the complete Fermi-Einstein integral of order j. It is known [Dingle, 1957] that $B_j(r) = -\mathrm{Li}_{j+1}\big(-e^r\big)$, where Li_{j+1} denotes the polylogarithm of order $j+1$. The Fermi-Dirac integral satisfies a recursive relationship

$$\frac{d}{dr} B_j(r) = B_{j-1}(r),$$

with a closed-form initial value $B_1(r) = \log(1+e^r)$.

At first, this might appear to be an attractive way to construct softplus activations and their zero-temperature limits. If $B_2(r)$ were an admissible log partition function, $\sigma(r) = A'(r) = \log(1+e^r)$.

However, the Fermi-Dirac integral is not necessarily positive definite. Following the same method as the Bose-Einstein integral, we have

$$\begin{aligned}\int_0^\infty \frac{y^j}{e^{y-ir}+1}\,dy &= \int_0^\infty \frac{y^j e^{-y} e^{ir}}{1+e^{-y}e^{ir}}\,dy \\ &= \int_2^1 \frac{(p-1)\big(-\log(p-1)\big)^j e^{ir}}{1+(p-1)e^{ir}} \frac{-1}{p-1}\,dp, \qquad p = e^{-y}+1 \\ &= \int_1^2 \Big(\frac{(-1)^j \log(p-1)^j}{p}\Big) \frac{pe^{ir}}{1+(p-1)e^{ir}}\,dp.\end{aligned}$$

As before, observe that $\Big(\frac{(-1)^j \log(p-1)^j}{p}\Big)$ is positive in $(1,2)$. However, $\frac{1}{1+(1-p)e^{ir}}$ admits a Fourier series (via the geometric series) $\sum_{k=0}^\infty (1-p)^k e^{irk}$. Since the Fourier series coefficients are **not always positive**, the Fermi-Dirac integral is indefinite. Hence we cannot conclude that its exponential is a positive definite function.

A.6 Bregman divergences

We need to recall some machinery from Banerjee et al. [2005]. In this subsection, without loss of generality, suppose T is the identity. We work with the exponential family form

$$p(y \mid r) = h(y) \exp\big(yr - A(r)\big),$$

which is a probability density function for a scalar-valued random variable. Connections extend to the vector-valued case, but since the setting in our text is restricted to factorised (conditionally independent) scalar-valued densities, we write the connection in terms of scalar-valued objects.

Let ϕ be a strictly convex and differentiable function mapping from a convex subset of $\mathbb{R}$ to $\mathbb{R}$. The Bregman divergence D_ϕ is defined as

$$D_\phi(\upsilon_1, \upsilon_2) = \phi(\upsilon_1) - \phi(\upsilon_2) - (\upsilon_1 - \upsilon_2)\frac{\partial}{\partial \upsilon_2}\phi(\upsilon_2).$$

Every Bregman divergence corresponds with exactly one exponential family through a convex conjugate. Concretely, the function ϕ that generates the Bregman divergence may be thought of as the convex conjugate of a log partition function A,

$$\phi(\mu) = \sup_{r\in\mathbb{F}}\{\mu r - A(r)\}.$$

We have that [Banerjee et al., 2005, Theorem 4]

$$p(y \mid r) = \exp\big(-D_\phi(y,\mu)\big) b_\phi(y),$$

where b_ϕ is a uniquely determined function that does not depend on the expectation parameter μ (or the canonical parameter r). The expectation parameter is related to the canonical parameter through $\mu = A'(r)$ and $r = \phi'(\mu)$.

B Proof

Theorem 1. *Any MAP estimate* (5) *of the graphical model in Figure 1 is a DEQ*

$$\mathsf{W}^*, B^* \in \operatorname*{argmin}_{\mathsf{W},B} -\log p(\mathsf{W}, B) - \log p(\mathsf{Z}^*) + \sum_{s=1}^{N} 1_{d\times 1}^\top A\left(\mathsf{W}Z_s^* + B\right) - T(Y_s)^\top (\mathsf{W}Z_s^* + B)$$

$$\textit{subject to } Z_s^* = \frac{1}{\lambda}\mathsf{W}^\top \big(T(Y_s) - \sigma(\mathsf{W}Z_s^* + B)\big), \quad \forall 1 \le s \le N. \tag{6}$$

Solutions to the inner constraint (6) *are guaranteed to exist and be unique.*

Proof. The log posterior over Z and W, B is

$$\log p(\mathsf{Z}, \mathsf{W}, B \mid \mathsf{Y}) = \log p(\mathsf{Y} \mid \mathsf{W}, \mathsf{Z}, B) + \log p(\mathsf{Z}) + \log p(\mathsf{W}, B) - \log p(\mathsf{Y}).$$

The joint MAP estimate $(\mathsf{Z}^*, \mathsf{W}^*, B^*)$ solves the constrained optimisation problem

$$\mathsf{W}^*, B^* = \operatorname*{argmin}_{W,B\in\mathbb{W}} -\log p\big(\mathsf{Y} \mid \mathsf{W}\mathsf{Z}^* + B\big) - \log p(\mathsf{W}, B) - \log p(\mathsf{Z}^*)$$

$$\text{subject to } Z_s^* = \operatorname*{argmin}_{Z_s} \frac{\lambda}{2}\|Z_s\|_2^2 - Z_s^\top \mathsf{W}^\top + B^\top T(Y_s) + 1_{d\times 1}^\top A\big(\mathsf{W}Z_s + B\big).$$

The outer problem is converted into a sum by replacing $\log p\big(\mathsf{Y} \mid \mathsf{W}\mathsf{Z}^* + B\big)$ with the sum of likelihoods (1) evaluated at the datapoints $Y_s = (y_{si})_{i=1}^d$ and canonical parameters $H_s = \mathsf{W}Z_s^* + B$.

We convert the inner DDN layer into a DEQ layer. We consider the inner problem for fixed s, which finds coefficients given W and b. The stationary points of the inner constraint satisfy

$$Z^* = \operatorname*{argmin}_{z\in\mathbb{R}^l} \frac{\lambda}{2}\|z\|_2^2 - R(Z^\top \mathsf{W}^\top + B^\top)T(Y) + 1_{d\times 1}^\top A(\mathsf{W}z + B)$$

$$0 = \lambda Z^* - \mathsf{W}^\top \big(T(Y) + \mathsf{W}^\top \sigma(\mathsf{W}Z^* + B) \tag{9}$$

$$Z^* = \frac{1}{\lambda}\mathsf{W}^\top \big(T(Y) - \sigma(\mathsf{W}Z^* + B)\big). \tag{10}$$

The Hessian H of the constraint is strictly positive definite since the objective is strongly convex. Concretely, we have

$$H = \lambda I + \mathsf{W}^\top \Big(\operatorname{diag}\big(\sigma'(\mathsf{W}Z^* + B)\big)\Big)\mathsf{W},$$

where $\sigma' = A''$ is nonnegative by strict convexity of A. $\square$

C Example details

PCA. We use the scikit learn implementation of PCA with default hyperparameters. We scale the input data into PCA with scikit learn's StandardScaler.

UMAP. We use the publically available UMAP implementation [McInnes et al., 2018, BSD 3-Clause License] with default hyperparameters.

Data-generating process. We generate a 2D grid of size $10^5 \times 10^5$ with corners $(\pm 5, \pm 5)$. We then remove all points that are not inside the circle of radius 3 centered at $(2, 2)$, the circle of radius 2 centered at $(-3, -3)$ or the diamond of side length 1 centered at $(4, -4)$. These are the ground truth Z, consisting of 42453 points in the 2D plane. We choose an A, then repeat the following steps 100 times to obtain 100 random datasets. We generate $\mathsf{W} \in \mathbb{R}^{50\times 2}$ by drawing elements iid from a Gaussian with zero mean and variance $\frac{1}{2}$. We then sample Y from an exponential family with canonical parameter WZ, log partition function A and sufficient statistic T.

DEQ. We use a DEQ with the same choice of A and T that matches the data generating process. This automatically defines the dropout functions (if any) and activation functions in the network. We use a value of $\lambda = 0.1$. We found it important to initialise weights with a small variance, especially for Poisson distributions, where the exponential inverse link function can either overflow or become very large. We use a zero mean iid Gaussian prior over W, B, which is implemented by applying weight decay to the neural network optimiser. We use Adam [Kingma and Ba, 2015] to optimise parameters W and B using default hyperparameters using a batch size of 500 and weight decay varying with layer (note this is not the same as using weight decay with AdamW, which would not be equivalent to L2 regularisation or Gaussian prior regularisation [Loshchilov and Hutter, 2019]). Weight decay is set to 10×50. We train the network for 30 epochs. We use a publically available DEQ repository [Bai et al., 2019, MIT License] to implement our DEQs, with a Anderson acceleration method used as the solver with default hyperparameters.

Downstream task. We use a fully connected head with widths $2, 100, 1$ and ReLU activations. Together with our DEQ or (untrainable) PCA/UMAP backbone, we train the full network to predict $g(Z_s) \in \mathbb{R}$ given an input $Y_s \in \mathbb{R}^{50}$ by minimising the sum of squared errors. This is done for each of the 100 randomly generated datasets. The corresponding $Z_s \in \mathbb{R}^2$ are not known to the network. This is an easy to visualise but deceptively difficult task that relies heavily on the quality of the latent embedding. A limitation of our evaluation is that we did not tune hyperparameters. However, if latents were to be used in a downstream task not known *a priori*, there would be no way of obviously tuning hyperparameters.

We use a small head network Linear(100) -> ReLU -> Linear(1) using default initialisation in Pytorch. The headnet is appended to our pretrained backbone of each of PCA, DEQ and UMAP and fine-tuned using Adam with default hyperparameters. For PCA and UMAP, which do not contain learnable parameters, the backbone network is fixed. The full network is trained to minimise the sum of squared errors between the output of the network and the function $g(z) = Z_1 + Z_2$, where Z_1 and Z_2 are respectively the first and second coordinate of z. We use a batch size of 500 and train the network for 200 epochs. In order to evaluate our model on the downstream task, we turn dropout off. We randomly partition each of the datasets into $80\% - 20\%$ training-testing split and evaluate our network on the test set. We report the number of times the network performed the best out of the 100 random runs for each of PCA, UMAP and DEQ.

Hardware, software and computational cost. We parallelised runs over multiple nodes, each consisting of a single Tesla P100-SXM2-16GB GPU. We used CUDA 11.4 and Pytorch 1.11.0. Only DEQ is GPU-accelerated. We found that typical run times for the dimensionality reduction task were task dependent. Table 2 shows typical run times for each task.

	PCA	tSNE	UMAP	DEQ
Gaussian	0.09	901.94	180.46	294.09
Bernoulli	0.11	923.85	174.71	267.81
Poisson	0.10	698.34	176.33	189.87

Table 2: Measured run times (seconds) for each dataset and model.

D Approximation to the posterior

The Hessian of the Laplace approximation is found by taking the derivative of the right hand side of (9) with respect to Z^*. We find

$$H(Z^*) := -\log p(Z^* \mid \mathsf{Y}, \mathsf{W}, B) = \lambda \mathsf{I} + \mathsf{W}^\top \mathrm{diag}\big(\sigma'(\mathsf{W}Z^* + B)\big)\mathsf{W}.$$

This leads to the Laplace approximation to the posterior,

$$q(z \mid \mathsf{Y}, \mathsf{W}, B) = \mathcal{N}\Bigg(z \mid Z^*, H(Z^*)^{-1}\Bigg).$$

Note that the Hessian is positive definite at the local minima that we found, so that inversion of the Hessian is well-defined.

If the likelihood were linearly parameterised Gaussian, σ' would be 1 and we would recover a familiar Gaussian case,

$$H(Z^*) = \lambda I + \mathsf{W}^\top \mathsf{W},$$

which is independent of the MAP Z^*.