Computational Statistics

Hyperspherical Variational Auto-Encoders

Victor Deng   Inès Vati

École Normale Supérieure Paris-Saclay, Master MVA

10^{th} JAN 2024

Table of contents

Table of contents

Introduction

2018 paper from Tim R. Davidson *et al.* [DFDC+]

- Replacing the Gaussian prior and approximate posterior distributions with a von Mises-Fisher distribution
- Goal: better model data with a hyperspherical latent structure
- Various experiments, where the $\mathcal{S}$-VAE (von Mises-Fisher distributions) often outperforms the $\mathcal{N}$-VAE (Gaussian distributions) in low dimensions

Table of contents

Sampling z' from vMF

Algorithm 1 Overview of the sampling method from $vMF(\mu, \kappa)$

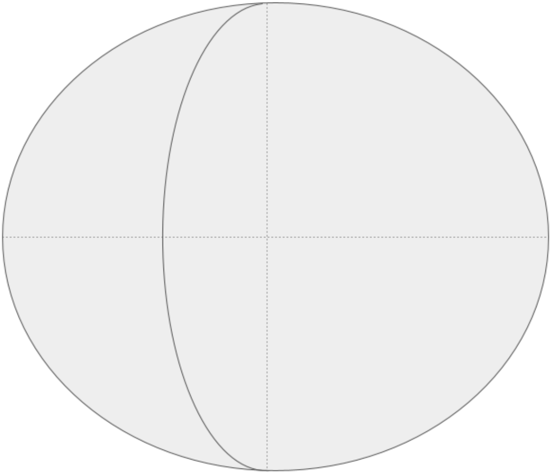
1: Sample $z \sim q(z|e_1, \kappa)$ where $e_1 = (1, 0, \ldots, 0)$
2: Compute Householder reflection $U(\mu)$ so that $U(\mu)e_1 = \mu$
3: **return** $z' = U(\mu)z$

Sampling z from vMF

Algorithm 2 Overview of the sampling method from $vMF(\mu, \kappa)$

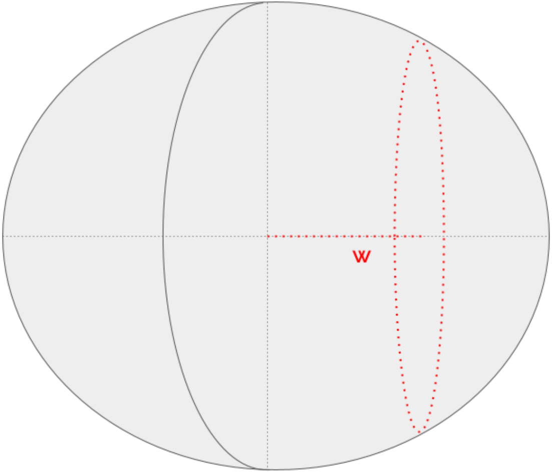
1: Sample $z \sim q(z|e_1, \kappa)$ where $e_1 = (1, 0, \dots, 0)$
2: Sample $w \in \mathbb{R} \sim g(w|\kappa)$ by acceptance rejection sampling
3: Sample $v \in \mathbb{R}^{d-1} \sim \mathcal{U}(S^{d-2})$ (uniform on the hypersphere S^{d-2} independent of w)
4: $z \leftarrow (w, \sqrt{1-w^2}v^T)^T$
5: Compute Householder reflection $U(\mu)$ so that $U(\mu)e_1 = \mu$
6: **return** $z' = U(\mu)z \sim q(z'|\mu, \kappa)$

Sampling w from $g(w|\kappa, \theta)$



S^2 : unit sphere in $\mathbb{R}^3$

Sampling w from $g(w|\kappa, \theta)$



Sample $w \in \mathbb{R} \sim g(w|\kappa, d)$ by acceptance rejection sampling

Sampling w from $g(w|\kappa)$

- Generale case
  - Sample target distribution $w \in \mathbb{R} \sim g(w|\kappa)$ by sampling a proposal w_{prop} of known density $r(w|\kappa)$
  - Perform backpropagation by reparameterizing $r(w|\kappa)$ so that the sampling is independent of the parameters
  - Note that r is not explicitly given in the article

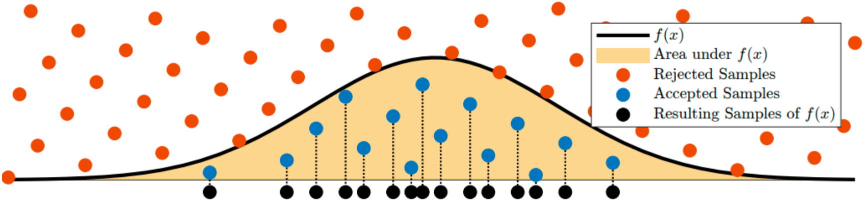


Illustration from [FH]

Sampling w from $g(w|\kappa)$

- Case $d = 3$: faster to use inverse transformation method
  - The vMF distribution explicity writes :

$$f_{vMF}(z) = \frac{\kappa}{4\pi \ sinh(\kappa)} exp(\kappa \mu^T z)$$

  - $(w, \sqrt{1-w^2} v^T)^T \sim vMF(e_1, \kappa)$ where $v \sim \mathcal{S}^2$ and $w \in [-1, 1]$ has density

$$f_W(w) = \frac{\kappa}{2 \ sinh(\kappa)} exp(\kappa w)$$

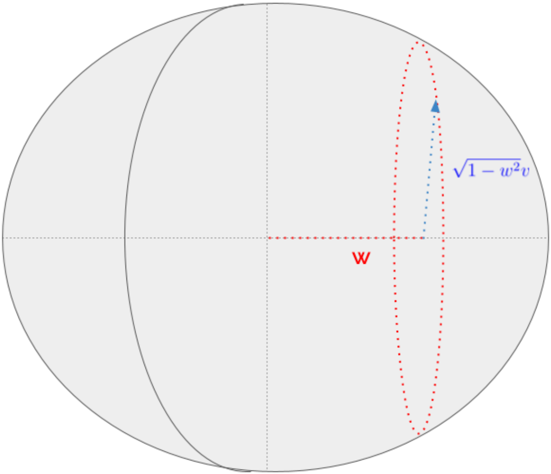
  - We compute its cumulative distribution function $F_W(w)$ and its inverse

$$F_W^{-1}(u) = \frac{1}{\kappa} ln(exp(-\kappa) + 2 \ sinh(\kappa) u)$$

  - As $sinh(\kappa)$ is numerically instable, we rewrites

$$F_W^{-1}(u) = 1 + \frac{1}{\kappa} ln(u + (1-u) exp(-2\kappa))$$

Sampling v from $\mathcal{U}(S^{d-2})$



Sample $v \in \mathbb{R}^{d-1} \sim \mathcal{U}(S^{d-2})$

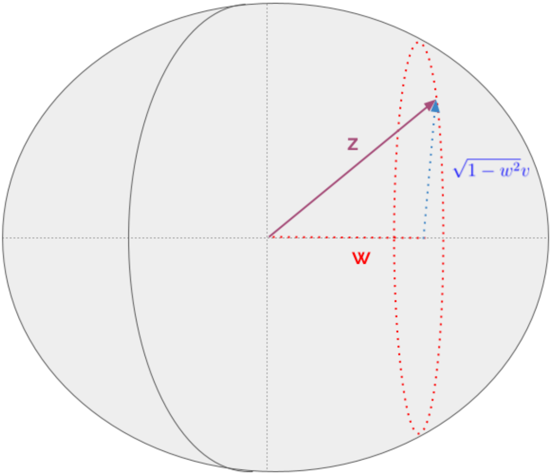
Sampling v from $\mathcal{U}(S^{d-2})$

- $\mathcal{N}(0, I_{d-1})$ is rotationally symmetric around the origin
- $f_{Y_1,\dots,Y_{d-1}} = \frac{1}{\sqrt{2\pi}^{d-1}} exp(-(Y_1^2 + \dots + Y_{d-1}^2)/2) = \frac{1}{\sqrt{2\pi}^{d-1}} exp(-1^2/2)$ which is constant in all of the angular variables.

Algorithm 3 Sampling v from $\mathcal{U}(S^{d-2})$

1: Generate $d-1$ *iid* variables (X_i) from $\mathcal{N}(0,1)$
2: $Y_i \leftarrow \frac{X_i}{\sqrt{X_1^1 + \dots + X_{d-1}^2}}$
3: **return** $(Y_i)_{i=1,\dots,d-1} \sim \mathcal{U}(S^{d-2})$

Sampling z from $q(z|e_1, \kappa)$



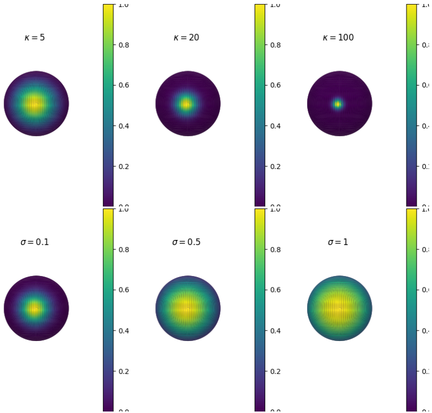
$$z = (w, \sqrt{1-w^2}v^T)^T$$

Algorithm 4 Overview of the sampling method from $vMF(\mu, \kappa)$

Require: $\mu \in \mathbb{R}^d$, $\kappa \in \mathbb{R}_+$

1: Sample $z \sim q(z|e_1, \kappa)$ where $e_1 = (1, 0, \ldots, 0)$
2: Compute Householder reflection $U(\mu)$ so that $U(\mu)e_1 = \mu$
3: $u \leftarrow Normalize(e_1 - \mu)$
4: $U \leftarrow I - 2uu^T$
5: **return** $z' = U(\mu)z$

Sampling results



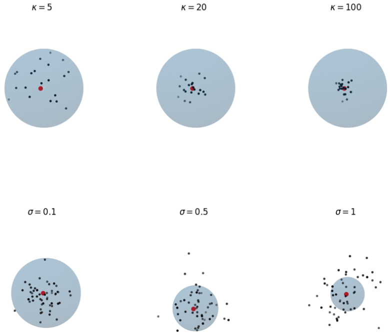


Table of contents

Reparameterization Trick

The authors use a reparameterization trick that has been extended to distributions that can be sampled using rejection sampling [NRLB20].

Algorithm 5 Reparameterized Rejection Sampling (from [NRLB20])

1: $i \leftarrow 0$
2: **repeat**
3: $\quad i \leftarrow i + 1$
4: $\quad$ Propose $\varepsilon_i \sim s(\varepsilon)$
5: $\quad$ Simulate $u_i \sim \mathcal{U}[0, 1]$
6: **until** $u_i < \frac{g(h(\varepsilon_i,\theta);\theta)}{r(h(\varepsilon_i,\theta);\theta)}$
7: **return** ε_i

Monte Carlo estimation

By noting $\pi(\varepsilon|\theta)$ the distribution of the resulting ε, we have (gradient of the expected log-likelihood)

$$\nabla_\theta \mathbb{E}_{g(\varepsilon|\theta)}[...] = \mathbb{E}_{\pi(\varepsilon|\theta)}[...] \text{ ''}= \mathbb{E}_{(\varepsilon_i, U_i)_i}[...]\text{''}$$

Problem: $(\varepsilon_i, U_i)_{i\in\mathbb{N}}$ is not a random variable (it is a stochastic process)
No reference to a convergence proof in [DFDC$^+$, NRLB20, PBJ12, MG14]

Table of contents

Experiments on link prediction

- Link prediction on a graph dataset: given a graph with some edges removed, predict the likelihood for each pair of nodes to be connected by an edge
- Cora dataset [MNRS00]: 2708 publications, 5429 links, 1433-dimensional feature vectors
- Using a Variational Graph Auto-Encoder [KW]: a variational encoder which uses a graph neural network (GNN) as encoder
- Reconstruction loss:

$$\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}(\log p(\mathbf{A}|\mathbf{Z})) \quad \text{where} \quad p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{i,j}|\mathbf{z}_i, \mathbf{z}_j)$$

- Negative sampling: in the sum $\sum_{i,j} \log p(A_{i,j}|\mathbf{z}_i, \mathbf{z}_j)$, keep all positive edges and one randomly sampled negative edge per positive edge

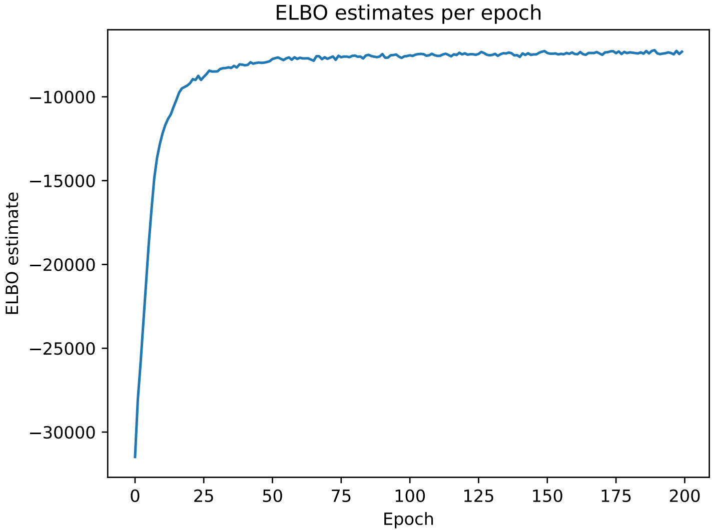
Loss computation

- Reconstruction loss
  - $$\mathcal{L}_{recon} = -\mathbb{E}_{q_\psi}(log\ \ p_\phi(x|z))$$
  - $$-\nabla_\kappa \mathcal{L}_{recon} \approx g_{recon} + g_{cor}$$
- KL Divergence
  - $$\mathcal{L}_{\mathcal{KL}} = \mathcal{KL}(q(z|\mu, \kappa)||p(z))$$
  - $$\nabla_\kappa \mathcal{L}_{\mathcal{KL}}$$

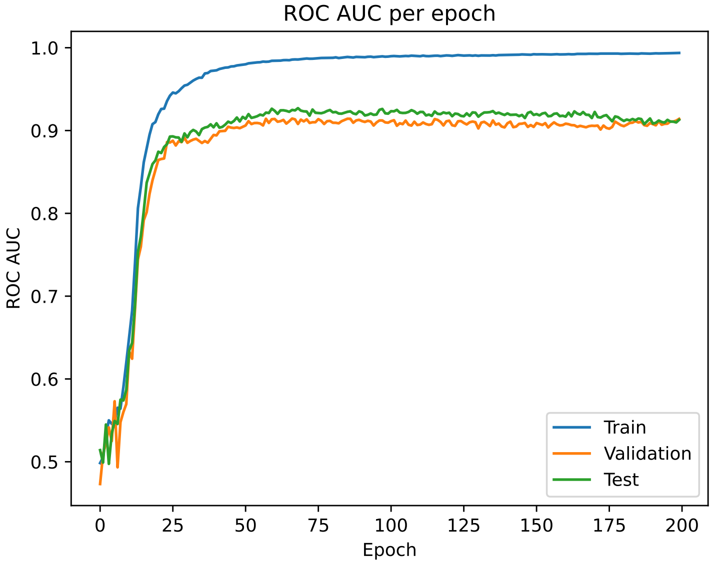
The formulas were explicity computed in the article.

Training curves for $\mathcal{N}$-VGAE

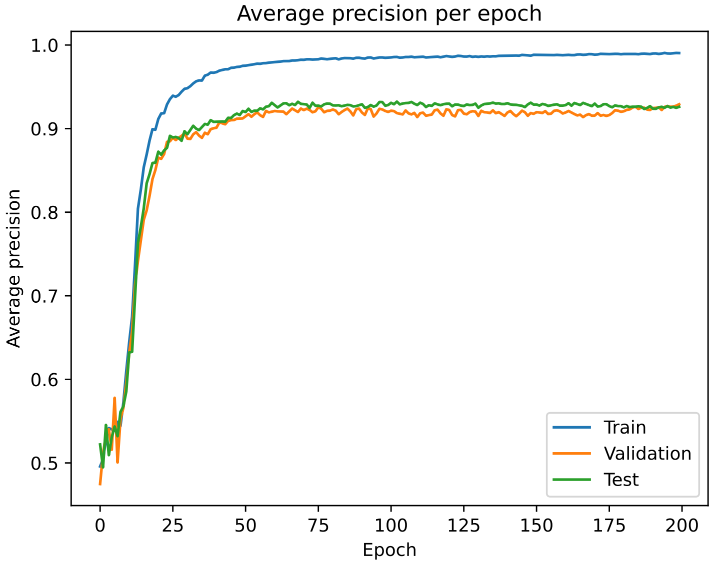
Latent dimension 16, learning rate 0.01. No KL divergence in loss.



Area under ROC curve for $\mathcal{N}$-VGAE

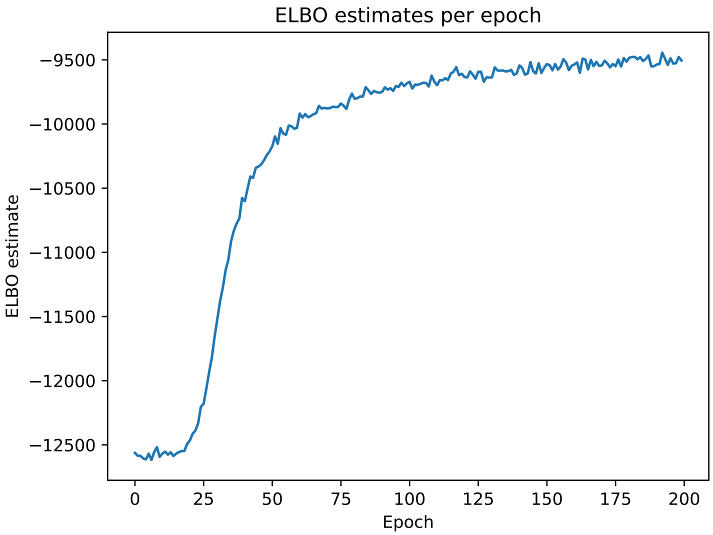


Average precision for $\mathcal{N}$-VGAE

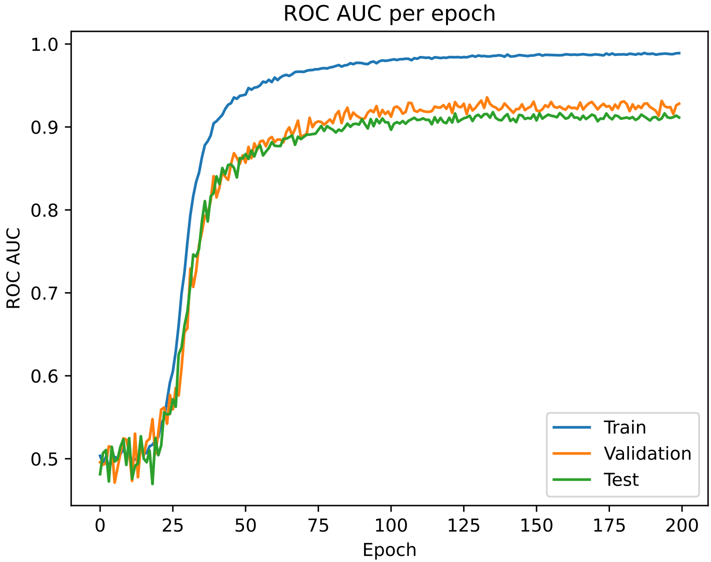


Training curves for $\mathcal{S}$-VGAE

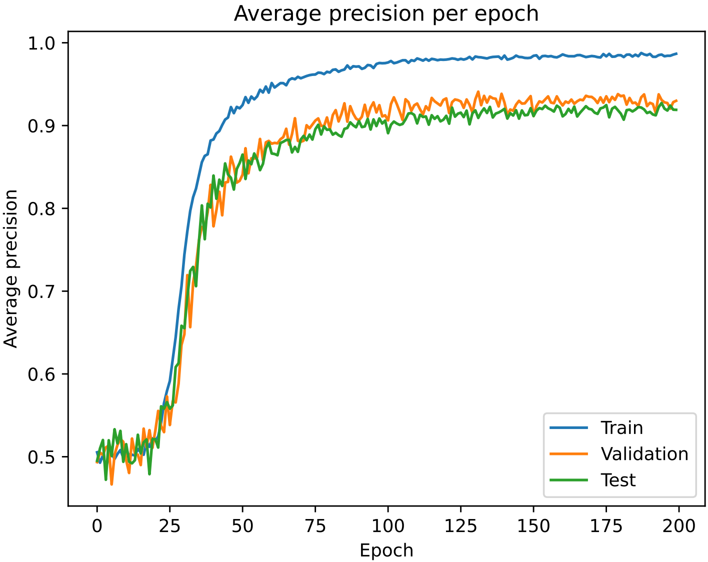
Latent dimension 16, learning rate 0.01. No KL divergence in loss.



Area under ROC curve for $\mathcal{S}$-VGAE



Average precision for $\mathcal{S}$-VGAE



Impact of dimension d on $\mathcal{S}$-VGAE

vmf d8 : mean auc : 0.90, 0.004638416275088545 vmd d8 : mean ap : 0.90, 0.006861910798200132 vmf d16 : mean auc : 0.91, 0.005106467210129149 vmd d16 : mean ap : 0.91, 0.002400819600884836 vmf d32 : mean auc : 0.92, 0.003988671804848861 vmd d32 : mean ap : 0.92, 0.006771540977322898 vmf d64 : mean auc : 0.92, 0.004086768956130232 vmd d64 : mean ap : 0.92, 0.006337561094050328

Table of contents

- Quite meaningful contribution in low dimensions
- Algorithm not really useful in high dimensions, due to vanishing surface problem and soap bubble effect of the $\mathcal{N}$-VAE
- Much less variance parameters (1 vs. d for $\mathcal{N}$-VAE), so possibly less expressivity

References I

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak.
Hyperspherical variational auto-encoders.

Daniel Frisch and Uwe Hanebeck.
Rejection sampling from arbitrary multivariate distributions using generalized fibonacci lattices.

Thomas N. Kipf and Max Welling.
Variational graph auto-encoders.

Andriy Mnih and Karol Gregor.
Neural variational inference and learning in belief networks, 2014.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore.
Automating the construction of internet portals with machine learning.
Information Retrieval, 3(2):127–163, 2000.

Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei.
Reparameterization gradients through acceptance-rejection sampling algorithms, 2020.

John Paisley, David Blei, and Michael Jordan.
Variational bayesian inference with stochastic search, 2012.