Deep Learning

- what is a deep neural network?
- neural conditional models (supervised)
- deep generative models (unsupervised)
- deep exponential families

Neural network

parameterized function from features $x \in \mathbb{R}^p$ to output $w \in \mathbb{R}^d$

notation: $\Omega(x; \theta)$

"hidden unit"

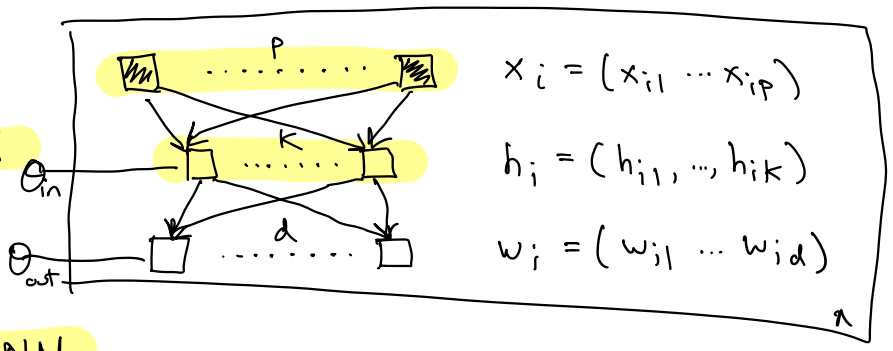
$x_i \in \mathbb{R}^p$

$h_i = g(\theta \cdot x_i)$ e.g., g is logistic

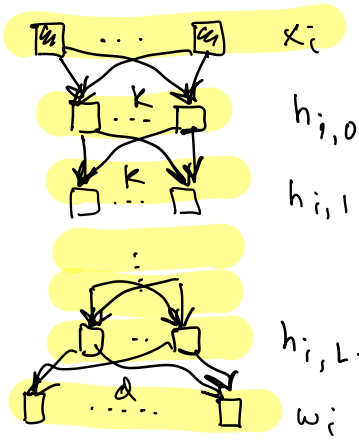
one-layer NN

$h_{ik} = g(\theta_{in,k} \cdot x_i)$ $k = 1 \dots K$

$w_{ij} = \theta_{out,j} \cdot h_i$ $j = 1 \dots d$

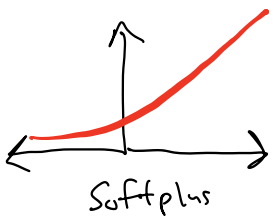


deep NN, aka feed-forward NN



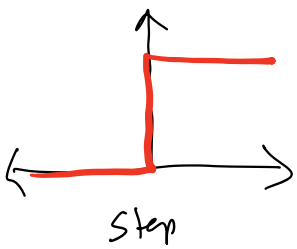
$\nabla_\theta \Omega(x; \theta)$

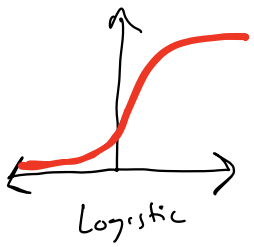
$\Omega(x; \theta)$

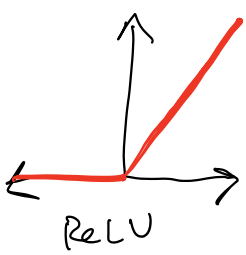


$\log(1 + e^x)$

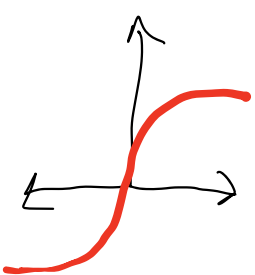
① activation function $g(\cdot)$







tanh.



Neural regression

$$\theta \sim N_B(0, \lambda^2)$$

$$y_i \mid x_i \sim N(\Omega(x_i; \theta), \sigma^2)$$

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$$

MAP estimation of θ:

$$\mathcal{L}(\theta) = -\frac{1}{2\lambda^2}\sum_{j=1}^{W}\theta_j^2 - \sum_{i=1}^{n}\frac{1}{2\sigma^2}(y_i - \Omega(x_i;\theta))^2$$

$$\nabla\mathcal{L}(\theta) = \underbrace{-\frac{1}{\lambda}\theta}_{W\text{-vector}} + \underbrace{\sum_{i=1}^{n}\frac{1}{\sigma^2}(y_i - \Omega(x_i;\theta))\nabla_\theta\Omega(x_i;\theta)}_{W\text{-vector}}$$

Generalized neural conditional model (GNCM)

$$\theta \sim N_W(0, \lambda^2)$$

$$y_i \mid x_i \sim \text{expfam}(\eta_i) \qquad \eta_i \triangleq \eta_\mu(f(\Omega(x_i;\theta)))$$

$$x_i \xrightarrow{\Omega} w_i \xrightarrow{f} \mu_i \xrightarrow{\eta_\mu} \eta_i \rightarrow y_i$$

Canonical link: $f = \eta_\mu^{-1} = \nabla a(\eta)$

in a GLM, $w_i = \theta \cdot x_i$

here, $w_i = \Omega(x_i;\theta)$

$$y_i \mid x_i \sim \text{expfam}(\Omega(x_i))$$

cross entropy loss : Bernoulli response w/ canonical link

soft max : Categorical response w/ " "

$$y_i \mid x_i \sim N(\Omega_\mu(x_i;\theta), \Omega_{\sigma^2}(x_i;\theta))$$

$$\mathcal{L}(\theta) = -\frac{1}{2\lambda^2}\sum_{j=1}^{W}\theta_j^2 + \sum_{i=1}^{n}\eta_\mu(\underbrace{f(\Omega(x_i;\theta))}_{\eta_i})\cdot y_i - a(\eta_\mu(f(\Omega(x;\theta))))$$

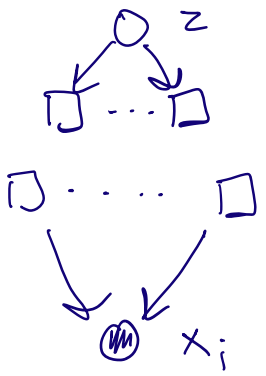
y_i is scalar

with canonical link:

$$\mathcal{L}(\theta) = -\frac{1}{2\lambda^2}\sum_{j=1}^{W}\theta_j^2 + \sum_{i=1}^{n}(\Omega(x_i;\theta)\cdot y_i - a(\Omega_\theta(x_i)))$$

$$\nabla \ell(\theta) = \lambda^2 \theta + \sum_{i=1}^{n} \left(y_i - \mathbb{E}[Y \mid X = x_i, \theta] \right) \nabla_\theta \Omega(x_i; \theta)$$

Deep generative models.



$\theta \sim p(\theta)$

$z_i \sim N_K(0, \lambda^2)$

$x_i \mid z_i \sim \text{expfam}(\Omega(z_i; \theta))$

deep generative model.

x_i : high dimensional

z_i : lower dimensional.

dimension reduction.

$$z_i \sim N_K(0, \lambda^2)$$

$$x_i \sim N_p(\Omega_\mu(z_i; \theta), \Omega_{\sigma^2}(z_i; \theta))$$

"decoder"

$p(z_i \mid x_i, \theta)$ — K-dim embedding of x_i

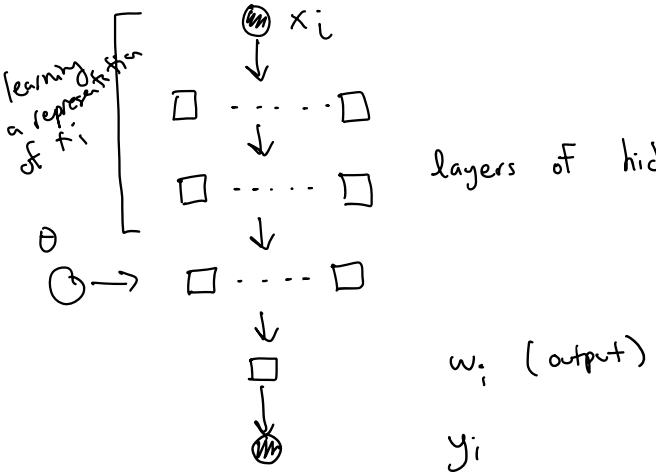
$$p(x \mid \theta) = \int p(x \mid z, \theta)\, p(z)\, dz$$

$$\begin{cases} z_i \sim N_K(0, \lambda^2) \\ x_i \mid z_i \sim N(\beta \cdot z_i, I\sigma^2) \end{cases} \qquad \beta : K \times p$$

prob. PCA.

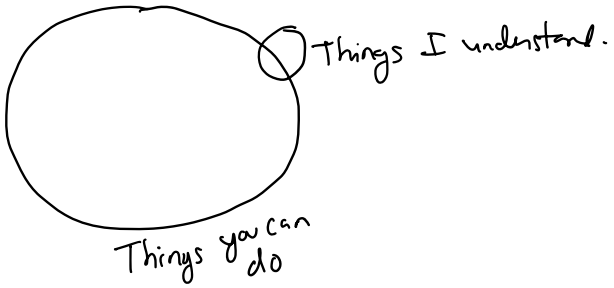
Today

- deep generative models
- deep exponential families
- architectural elements: residuals, attention, convolution
- priors for neural networks



$$y_i \mid x_i \sim \mathcal{N}(\theta_{out} \cdot h_L(x_i), \sigma^2)$$

Ben Recht's Venn Diagram

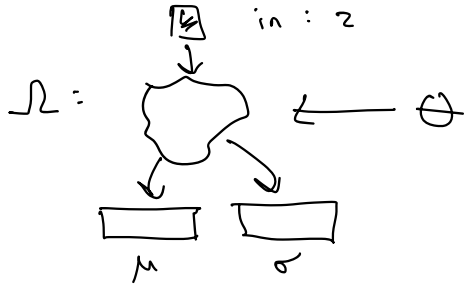


Deep Generative Model.

$$z_i \sim \mathcal{N}_K(0, 1) \qquad z_i \in \mathbb{R}^K$$

$$x_i \sim \text{expfam}(\Omega(z_i; \theta)) \qquad x_i \in \mathbb{R}^d$$

$$x_i \sim \mathcal{N}\left(\Omega_\mu(z_i; \theta), \Omega_\sigma(z_i; \theta)^2\right)$$



local posterior: $p(z_i \mid x_i, \theta)$

$\mathbb{E}[z \mid x_i, \theta] \triangleq \hat{z}_i$

dimension reduction.

aside: Prob. PCA (Tipping + 1999)

$$z_i \sim \mathcal{N}_k(0, 1)$$

$$x_i \mid z_i \sim \mathcal{N}(\beta \cdot z_i, \sigma^2)$$

β: $d \times K$

Idea: DGM can capture many complex distributions of x

$$p(x \mid \theta) = \int p(z, x \mid \theta)\, dz \qquad \text{marginal distribution of an obs.}$$

$$= \int p(z)\, p(x \mid z, \theta)\, dz$$

$$= \int \text{normal}_k(z; 0, 1)\ \text{normal}\left(x; \Omega_\mu(z), \Omega_\sigma(z)^2\right) dz$$

$\mathcal{D} = \{x_i\}_{i=1}^n$

$p(\theta \mid x_{1:n})$

$$p(\theta, z_{1:n}, x_{1:n}) = p(\theta) \prod_{i=1}^{n} p(z_i)\, p(x_i \mid z_i, \theta) \quad \leftarrow \text{"Factor model"}$$

z_i : local latent variables

θ : global " "

↙ scVI.

$x_i \sim \text{NegBin}(\Omega(z_i; \theta))$

RNA seq data : 10k–20k dim obs (counts), 20 dim z, goal: use $\hat{z}_i$ to predict sth else.

Robots : 15 dim obs, 5 dim $\mathbf{z}$, goal: sample from $p(x)$
↳ IR

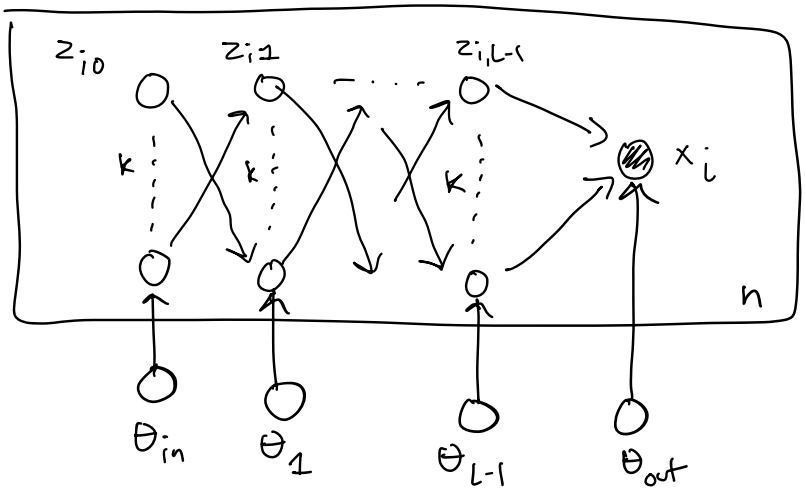
$$p(z_i \mid x_i, \hat{\theta}) = p(z_i)$$ POSTERIOR COLLAPSE

latent variable id. (Wang + 2020/1)

Deep Exponential Families

Ranganath + 2015

everything becomes a random variable.



$$\mathbb{E}\left[Z_{ilk} \mid \mathbf{z}_{i,l-1}, \theta_{lk}\right] = g\left(\theta_{lk} \cdot \mathbf{z}_{i,l-1}\right)$$

$z_{ilk} \mid \mathbf{z}_{i,l-1} \sim$ (η_z^{-1} : canonical link)

$$\text{expfam}\left(\eta_z\left(g\left(\theta_{lk} \cdot z_{i,l-1}\right)\right)\right)$$

$$x_i \mid \mathbf{z}_{i,L-1} \sim \text{expfam}\left(\eta_x\left(h\left(\theta_{out} \cdot \mathbf{z}_{i,L-1}\right)\right)\right)$$

$p(\mathbf{z}_{1:n}, \theta \mid \mathcal{D})$ — BBVI.

① how do DEFs stack against deep NNs.
② explore the "natural DEF"
③ what about conjugate pairs?

deep latent Gaussian model

$$z_{ikl} \mid z_{i,l-1} \sim \mathcal{N}\left(g(\theta_{lk} \cdot z_{i,l-1}), \sigma^2\right), \quad \mathbb{E}[z_{ikl} \mid z_{i,l-1}] = g(\theta \cdot z_{i,l-1})$$

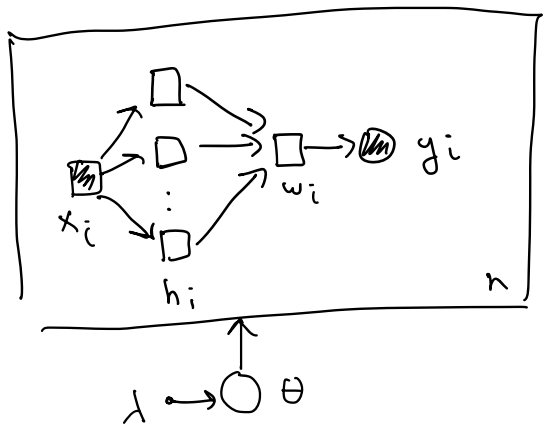
as $\sigma^2 \to 0$, DLGM approaches a NN.

Sigmoid belief network

$$z_{ikl} \mid z_{i,l-1} \sim \text{Bernoulli}\left(\sigma(\theta_{lk} \cdot z_{i,l-1})\right)$$

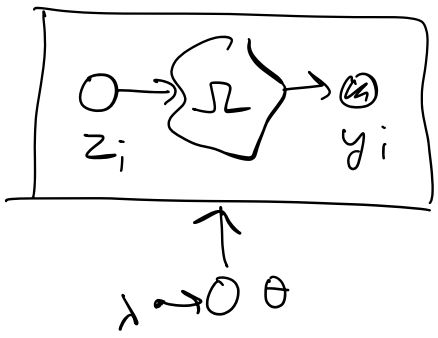
$$\mathbb{E}[z_{ikl} \mid z_{i,l-1}] = \sigma(\cdot)$$

- deep exponential families
- discussion of DL
- architectural elements
- priors for neural networks
- BBVI ...



$\Omega(x_i; \theta)$

$\nabla_\theta \Omega(x_i; \theta)$



Supervised Bayesian NN
MAP estimation. — Squared loss
x-entropy loss.

Deep generative model

- feature learners.
- Tensorflow, Pytorch*, Jax*.
- Automatic differentiation.

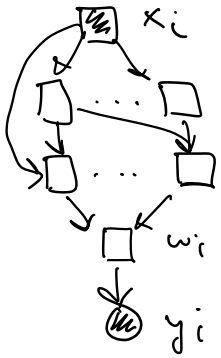
- activation function : ReLU
- intercept term : $h_{i,l,k} = g(\theta_{l,k} \cdot h_{i,l-1} + \alpha_{l,k})$

regularize less.

- residual connections :

or $h_{i,l,k} = h_{i,l-1,k} + g(\cdots)$

$$h_{i,l,k} = g(\theta_{l,k} \cdot h_{i,l-1} + \gamma_{l,k} \cdot x_i + \alpha_{l,k})$$

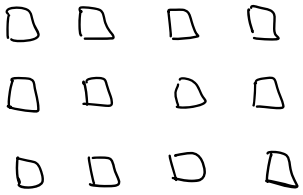


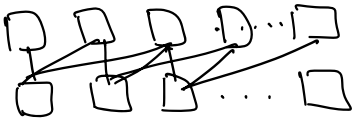
you can implement hidden → hidden residual connections too.

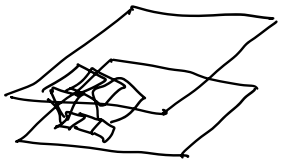
"skip connections"

- parameter sharing.

① convolution

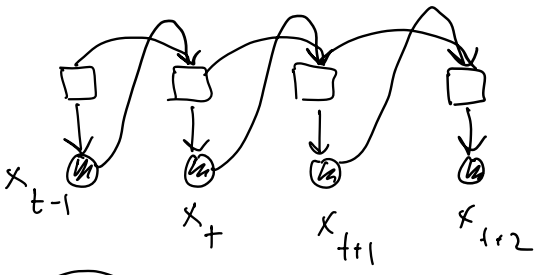






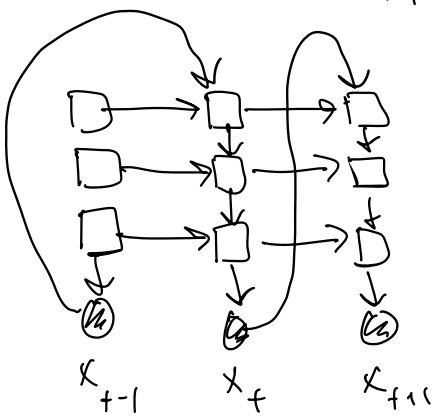
Yann Lecun.

- ② recurrence.



DNA data. methylation.

test data

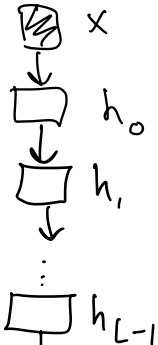


Sharing within each slice.

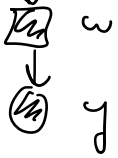
- Attention.

$$A(x_i; \alpha) : \mathbb{R}^p \to [0,1]^p$$

$$h_{i,0,k} = g(\theta_{in,k} \cdot [A(x_i; \alpha) \circ x_i])$$



$x_i \rightarrow$ 



$x_i \rightarrow$ 

The big dog ate my kid's gerbil and she was sad. "why is she sad?" I said, she just ate a gerbil.

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \dots \rightarrow x_T$$

$$y_i \mid x_i \sim N(\beta \cdot x_i, \sigma^2)$$

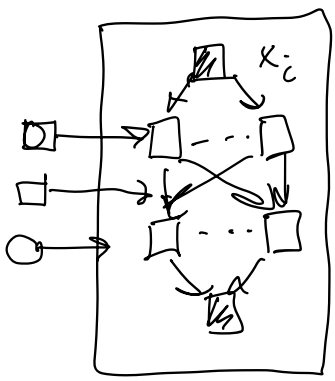
$$\sim N(\beta \cdot (\sigma(\alpha \cdot x_i) \circ x_i), \sigma^2)$$

$\sigma(\cdot)$: bank of logistic regressions.

soft $\rightarrow \Delta^{p-1}$

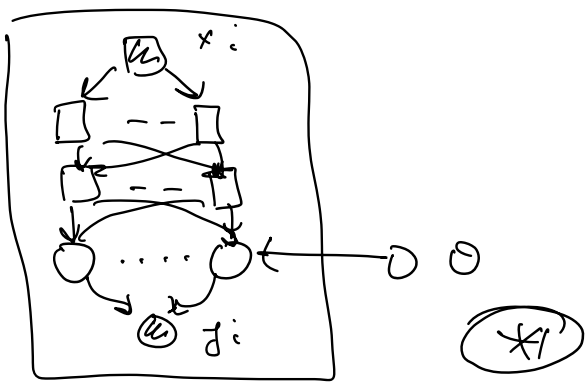
Bayesian N.N.

Automatic Relevance Determination.



feature learner

response model



Black Box Variational Inference

Complete conditionals → Gibbs, CAVI

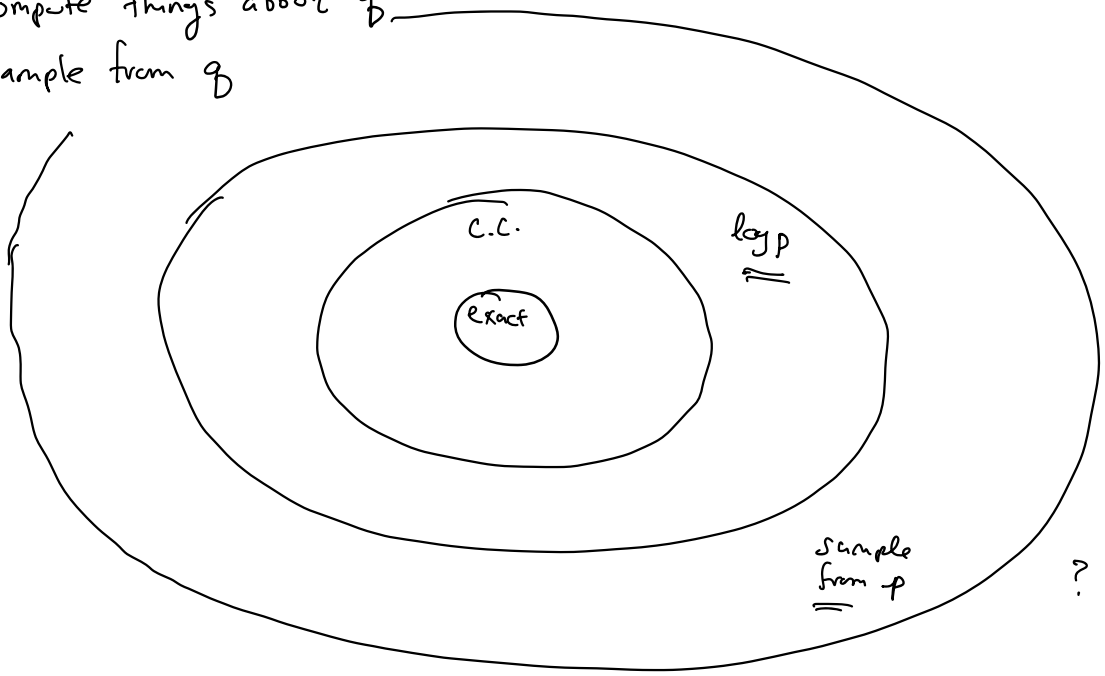
$p(\theta, z_{1:n}, x_{1:n})$ global θ local $z_{1:n}$

$$= p(\theta) \prod_{i=1}^{n} p(z_i \mid \theta)\, p(x_i \mid z_i, \theta)$$

$p(\theta, z_{1:n} \mid x_{1:n})$ - posterior. Goal: approximate it with $q(\theta, z_{1:n}; \nu)$

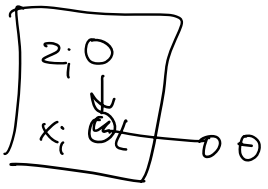
Black box criteria:

- compute the log joint (compute $\nabla_{\theta,z} \log p(z, \theta, x)$)
- compute things about q
- sample from q



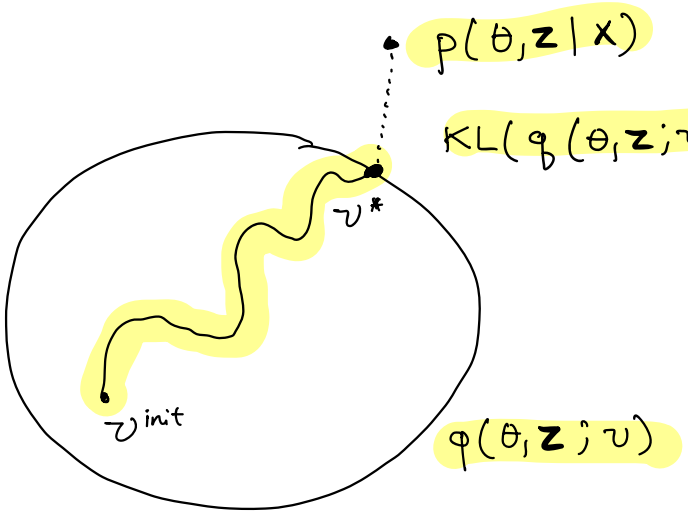
Recall: Deep generative model.

$$z_i \sim N_p(0, 1)$$
$$x_i \mid z_i, \theta \sim N(\Omega_\mu(z_i), \Omega^2_\sigma(z_i))$$



$$p(z_i \mid x_i, \theta) = \frac{p(z_i, x_i \mid \theta)}{p(x_i \mid \theta)}$$

$$p(z_i, x_i \mid \theta) = p(z_i)\, p(x_i \mid z_i, \theta)$$



Objective function

ELBO

EM: maximum likelihood estimation alg. when there are latent variables in the model.

$$\mathcal{L}(\nu) = \mathbb{E}_\nu[\log p(\theta, Z, x) - \log q(\theta, Z; \nu)]$$

Strategy:

1. Write $\nabla_\nu \mathcal{L}$ as an expectation $\mathbb{E}_q[\sim]$
2. Take a Monte Carlo approximation of $\nabla_\nu \mathcal{L}$ $\approx \frac{1}{B}\sum_b \sim_b$
3. Use stochastic optimization

Score gradient

$$\nabla_\nu \mathcal{L} = \mathbb{E}\left[\underbrace{\nabla_\nu \log q(\theta, Z; \nu)}_{\text{score function}} \underbrace{(\log p(\theta, Z, x) - \log q(\theta, Z; \nu))}_{\text{instantaneous ELBO}}\right]$$

$\nabla_\nu \log q(\theta, Z; \nu)$

$\nabla_{\theta, z} \log q(\theta, Z; \nu)$

$\nabla \log q(\theta, Z; \nu)$

] "score function"

Score VI

For iteration t:

$$\theta_b, z_b \sim q(\theta, z; v_t) \qquad b = 1 \dots B$$

$$g_t = \frac{1}{B} \sum_{b=1}^{B} \nabla_v \log q(\theta_b, z_b; v_t) \left(\log p(\theta_b, z_b, x) - \log q(\theta_b, z_b; v_t) \right)$$

$$v_{t+1} = v_t + \rho_t g_t$$

Example: DGM

$$\log p(\theta, z, x) = \log p(\theta) + \sum_{i=1}^{n} \log p(z_i) + \log p(x_i | z_i, \theta)$$

$$q(\theta, z; v) = q(\theta; v_\theta) \prod_{i=1}^{n} q(z_i; v_i)$$

($q(\theta; v_\theta)$: N; $q(z_i; v_i)$: N)

In practice:

Variance is high. — Andy Miller, Justin Domke, ...

control variates.

Let $g(\theta, z; v) \triangleq$ (stochastic) score gradient, $\mathbb{E}[g(\theta, z; v)] = \nabla_v \mathcal{L}(v)$

Choose $h(\theta, z; v)$ s/t $\mathbb{E}[h(\theta, z; v)]$

Define $\tilde{g}(\theta, z; v) = g(\theta, z; v) - \xi \left(h(\theta, z; v) - \mathbb{E}[h] \right)$ (ξ: scalar)

$$\mathrm{Var}(\tilde{g}) = \mathrm{Var}(g) + \xi^2 \mathrm{Var}(h) - 2\xi \, \mathrm{Cov}(g, h)$$

$$\xi^* = \frac{\mathrm{Cov}(g, h)}{\mathrm{Var}(h)}$$

$$\mathbb{E}\left[\nabla_v \log q(z;v)\right] = \int q(z;v)\, \nabla_v \log q(z;v)\, dz$$

$$= \int \nabla_v q(z;v)\, dz$$

$$= \nabla_v \int q(z;v)\, dz$$

$$= \nabla_v 1 = \emptyset$$

FACT:
$\nabla_v q(z;v) = q(z;v) \nabla_v \log q(z;v)$

Reparameterization gradient

transform the variables in q:

$\varepsilon \sim s(\varepsilon) \rightarrow z \sim q(z;v)$

$z = t(\varepsilon, v)$

e.g. for a normal:

$\varepsilon \sim N(0,1) \rightarrow z \sim N(v_\mu, v_\sigma^2)$

$z = \varepsilon v_\sigma + v_\mu$

Reparameterization gradient

$\nabla \mathcal{L}(v) =$

$$\mathcal{L}(v) = \mathbb{E}_{s(\varepsilon)}\left[\log p(\theta, z, x) - \log q(\theta, z; v)\right]$$

where: $\theta = t(\varepsilon_\theta, v_\theta)$, $z = t(\varepsilon_z, v_z)$

$$\nabla_v \mathcal{L}(v) = \mathbb{E}_{s(\varepsilon)}\left[\nabla_{\theta,z}\left[\log p(\theta,z,x) - \log q(\theta,z;v)\right]\nabla_v t(\varepsilon, v) - \nabla_v \log q(\theta, z; v)\right]$$

$$= \mathbb{E}_{s(\varepsilon)}\left[\nabla_{\theta,z}\left[\log p(\theta,z,x) - \log q(\theta,z;v)\right]\nabla_v t(\varepsilon, v)\right]$$