Re-parameterization and Gumbel-max Trick

A/Prof Richard Yi Da Xu
http://richardxu.com

University of Technology Sydney (UTS)

July 6, 2018

a motivating example: Generative Adversarial Training

- **Generative Adversarial Training**

Algorithm 1 GAN with both G_θ and D_ϕ

Require: data: $x_1, \dots, x_n \sim p(x)$

while some termination condition is not satisfied **do**

sample mini-batch of inputs $\{x_1, \dots, x_m\}$

sample noise $\{z_1, \dots, z_n\}$

update **discriminator** parameter ϕ:

$$\phi = \arg\min_{\phi} - \left[\frac{1}{m} \sum_x \log D_\phi(x) - \frac{1}{n} \sum_z \log(1 - D_\phi(G_\theta(z))) \right]$$

update **generator** parameter θ:

$$\theta = \arg\min_{\theta} - \left[\frac{1}{n} \sum_z \log \frac{D_\phi(G_\theta(z))}{1 - D_\phi(G_\theta(z))} \right]$$

end while

---

consider $D_\phi(G_\theta(z))$

- if $G_\theta(z)$ generates contiguous samples, e.g., image pixels, everything is fine:
- when $G_\theta(z)$ generates discrete samples: in the case of LSTM:

$$
\begin{aligned}
&G_\theta(z) \begin{cases} \dots \\ K_t & \sim \text{softmax}\big(\mu_1(z_t,\theta),\dots,\mu_L(z_t,\theta)\big) \\ \mathbf{v}_t & = \text{one-hot}(K_t) \\ \dots \end{cases} \\
&D_\phi(\mathbf{v})
\end{aligned}
$$

- can **not** take derivative: $\frac{\partial D}{\partial K_t}$
- can **not** back-propagate gradients through **random nodes** in a computation graph
-
- so the solution is use **re-parameterization**

Gumbel-max trick and Softmax (1)

- pdf of Gumbel with **unit scale** and location parameter μ:

$$\text{gumbel}(Z = z\,;\ \mu) = \exp\Big[-(z-\mu) - \exp\{-(z-\mu)\}\Big]$$

- CDF of Gumbel:

$$\text{Gumbel}(Z \leq z\,;\ \mu) = \exp\Big[-\exp\{-(z-\mu)\}\Big]$$

- given a set of Gumbel random variables $\{Z_i\}$, each having own location parameters $\{\mu_i\}$, probability of all other $Z_{i\neq k}$ are less than a particular value of z_k:

$$p\left(\max\{Z_{i\neq k}\} = z_k\right) = \prod_{i\neq k} \exp\Big[-\exp\{-(z_k - \mu_i)\}\Big]$$

- obviously, $Z_k \sim \text{gumbel}(Z_k = z_k; \mu_k)$:

$$\begin{aligned}
\Pr(k \text{ is largest} \mid \{\mu_i\}) &= \int \exp\left\{-(z_k-\mu_k) - \exp\{-(z_k-\mu_k)\}\right\} \prod_{i\neq k} \exp\left\{-\exp\{-(z_k-\mu_i)\}\right\} \mathrm{d}z_k \\
&= \int \exp\Big[-z_k + \mu_k - \exp\{-(z_k-\mu_k)\}\Big] \exp\Big[-\sum_{i\neq k} \exp\{-(z_k-\mu_i)\}\Big] \mathrm{d}z_k \\
&= \int \exp\Big[-z_k + \mu_k - \exp\{-(z_k-\mu_k)\} - \sum_{i\neq k} \exp\{-(z_k-\mu_i)\}\Big] \mathrm{d}z_k \\
&= \int \exp\Big[-z_k + \mu_k - \sum_{i} \exp\{-(z_k-\mu_i)\}\Big] \mathrm{d}z_k \\
&= \int \exp\Big[-z_k + \mu_k - \sum_{i} \exp\{-z_k+\mu_i)\}\Big] \mathrm{d}z_k \\
&= \int \exp\Big[-z_k + \mu_k - \exp\{-z_k\} \sum_{i} \exp\{\mu_i)\}\Big] \mathrm{d}z_k
\end{aligned}$$

Gumbel-max trick and Softmax (2)

- keep on going:

$$\begin{aligned}
\Pr(k \text{ is largest} \mid \{\mu_i\}) &= \int \exp\left[- z_k + \mu_k - \exp\{-z_k\} \sum_i \exp\{\mu_i)\} \right] \mathrm{d}z_k \\
&= \exp^{\mu_k} \int \exp\left[- z_k - \exp\{-z_k\} C \right] \mathrm{d}z_k \\
&= \exp^{\mu_k} \left[\frac{\exp(-C\exp(-z_k))}{C} \Big|_{z_k=-\infty}^{\infty} \right] \\
&= \exp^{\mu_k} \left[\frac{1}{C} - 0 \right] = \frac{\exp^{\mu_k}}{\sum_i \exp\{\mu_i)\}}
\end{aligned}$$

- $\mu_i \equiv \mathbf{x}^\top \theta_i$ in classification
- $\mu_i \equiv \mathbf{u}_i^\top \mathbf{v}_c$ for word vectors
- moral of the story is, if one is to sample the largest element from **softmax**:

$$\begin{aligned}
K &\sim \left\{ \frac{\exp(\mu_1)}{\sum_i \exp(\mu_i)}, \dots, \frac{\exp(\mu_L)}{\sum_i \exp(\mu_i)} \right\} \\
&\equiv \arg\max_{i \in \{1,\dots,L\}} \{G_1, \dots, G_L\} \qquad \text{where } \underbrace{G_i \sim \text{gumbel}(z; \mu_i) \equiv \exp\left[- (z - \mu_i) - \exp\{-(z - \mu_i)\} \right]} \\
&\equiv \arg\max_{i \in \{1,\dots,L\}} \{\mu_1 + \mathcal{G}, \dots, \mu_L + \mathcal{G}\} \qquad \text{where } \underbrace{\mathcal{G} \overset{\text{iid}}{\sim} \text{gumbel}(z; 0) \equiv \exp\left[- (z) - \exp\{-(z)\} \right]}
\end{aligned}$$

sample a Gumbel

- so we know that:

$$K \sim \left\{ \frac{\exp(\mu_1)}{\sum_i \exp(\mu_i)}, \dots, \frac{\exp(\mu_L)}{\sum_i \exp(\mu_i)} \right\}$$

$$\Longrightarrow \; k = \arg\max_{i \in \{1,\dots,L\}} \{\mu_1 + \mathcal{G}, \dots, \mu_L + \mathcal{G}\} \qquad \text{where } \underbrace{\mathcal{G} \overset{\text{iid}}{\sim} \text{gumbel}(z\,;\,0) \equiv \exp\left[-(z) - \exp\{-(z)\}\right]}$$

- how are we going to sample Gumbel distribution?
- CDF of a Gumbel:

$$\begin{aligned} u &= \exp^{-\exp^{-(x-\mu)/\beta}} \\ \Longrightarrow \quad \log(u) &= -\exp^{-(x-\mu)/\beta} \\ \Longrightarrow \quad \log(-\log(u)) &= -(x-\mu)/\beta \\ \Longrightarrow \quad -\beta\log(-\log(u)) &= x - \mu \\ \Longrightarrow \quad x = \text{CDF}^{-1}(u) &\equiv \mu - \beta\log(-\log(u)) \end{aligned}$$

- for standard Gumbel, i.e., $\mu = 0,\ \beta = 1$:

$$x = \text{CDF}^{-1}(u) \equiv -\log(-\log(u))$$

- therefore, sampling strategy:

$$\begin{aligned} U &\sim \mathcal{U}(0,1) \\ \mathcal{G} &= -\log(-\log(U)) \\ K &= \arg\max_{i \in \{1,\dots,K\}} \{\mu_1 + \mathcal{G}, \dots, \mu_L + \mathcal{G}\} \\ \mathbf{v} &= \text{one-hot}(K) \end{aligned}$$

- let's hold that thought for now: take a look at **re-parameterization trick**

Score Function Estimator

- we love have integral of the form:

$$\int_z f(z)p(z)\mathrm{d}z \equiv \mathbb{E}_{z\sim p(z)}[f(z)]$$

as we can approximate the **expectation** with:

$$\sum_{i=1}^{N} f(z^{(i)}) \qquad\qquad z^{(i)} \sim p(z)$$

- we do **not** love $\int_x f(z)\triangledown_\theta p(z|\theta)\mathrm{d}z$,
- in general, $\triangledown_\theta p(z|\theta)$ is **not** a probability, e.g., look at derivative of a Gaussian distribution:

$$\frac{\partial}{\partial\mu}\left(\frac{\exp^{-(z-\mu)^2/\sigma^2}}{\sqrt{2\pi}\sigma}\right) = \frac{2(z-\mu)}{\sigma^2}\frac{\exp^{-(z-\mu)^2/\sigma^2}}{\sqrt{2\pi}\sigma}$$

Score Function Estimator

- however, in machine learning, we have to deal with:

$$\nabla_\theta \left[\int_z f(z)p(z|\theta)\mathrm{d}z \right] = \int_z \nabla_\theta \left[f(z)p(z|\theta) \right] \mathrm{d}z = \int_z f(z) \left[\nabla_\theta p(z|\theta) \right] \mathrm{d}z$$

- e.g., in **Reinforcement Learning**: let $\Pi \equiv \{s_1, a_1, \ldots, s_T, a_T\}$

$$p_\theta(\Pi) \equiv p_\theta(s_1, a_1, \ldots s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

$$\implies \theta^* = \arg\max_\theta \left\{ \mathbb{E}_{\Pi \sim p_\theta(\Pi)} \left[\underbrace{\sum_{t=1}^{T} R(s_t, a_t)}_{f(z)} \right] \right\}$$

- we use **REINFORCE trick**, with the follow property:

$$p(z|\theta)f(z)\nabla_\theta[\log p(z|\theta)] = p(z|\theta)f(z)\frac{\nabla_\theta p(z|\theta)}{p(z|\theta)} = f(z)\nabla_\theta \left[p(z|\theta) \right]$$

- looking at the original integral:

$$\begin{aligned} \int_z f(z)\nabla_\theta \left[p(z|\theta) \right] \mathrm{d}z &= \int_z p(z|\theta)f(z)\nabla_\theta[\log p(z|\theta)]\mathrm{d}z \\ &= \mathbb{E}_{z \sim p(z|\theta)} \left[f(z)\nabla_\theta[\log p(z|\theta)] \right] \end{aligned}$$

- suffers from **high variance** and is slow to converge

re-parameterization trick

- we let $z = g(x)$:

$$\begin{aligned} \mathbb{E}_{x \sim p(x)}[g(x)] &= \mathbb{E}_{z \sim p(z)}[z] \\ \mathbb{E}_{x \sim p(x)}[g(x, \theta)] &= \mathbb{E}_{z \sim p_\theta(z)}[z] \qquad \text{paramterize the distribution with } \theta \\ \mathbb{E}_{x \sim p(x)}[f(g(x, \theta))] &= \mathbb{E}_{z \sim p_\theta(z)}[f(z)] \qquad \text{introduce function } f(.) \\ \int_{x \in \Omega_x} f(g(x, \theta)) p(x) \mathrm{d}x &= \int_{z \in \Omega_z} f(z) p_\theta(z) \mathrm{d}z \end{aligned}$$

- only need to know deterministic function $z = g(x, \theta)$ and distribution $p(x)$
- does **not** need to explicitly know distribution of z
- e.g., Gaussian variable: $z \sim \mathcal{N}(z; \mu(\theta), \sigma^2(\theta))$ can be rewritten as a function of a standard Gaussian variable:

$$z = \underbrace{\mu(\theta) + x\sigma^2(\theta)}_{g(x,\theta)} \qquad \text{can be re-parameterised into} \qquad x \sim \underbrace{\mathcal{N}(0, 1)}_{p(x)}$$

revision on change of variable

- Let $y = T(x) \implies x = T^{-1}(y)$:

$$F_Y(y) = \Pr(T(X) \le y) = \Pr(X \le T^{-1}(y)) = F_X\left(T^{-1}(y)\right) = F_X(x)$$

$$f_Y(y) = \frac{\mathrm{d}F_Y(y)}{\mathrm{d}y} = \frac{\mathrm{d}F_X(x)}{\mathrm{d}y} = \frac{\mathrm{d}F_X(x)}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}y} = f_x\left(x\right)\frac{\mathrm{d}x}{\mathrm{d}y}$$

- without change of limits

$$f_Y(y)\left|\mathrm{d}y\right| = f_x\left(x\right)\left|\mathrm{d}x\right|$$

- with change of limits

$$f_Y(y)\mathrm{d}y = f_x\left(x\right)\mathrm{d}x$$

re-parameterization trick (2)

- **main motivation** $p(x)$ is **not** parameterized by θ:

$$
\begin{aligned}
\mathbb{E}_{x\sim p(x)}[f(g(x,\theta))] &= \int_x f(g(x,\theta))p(x)\mathrm{d}x \\
\Longrightarrow \frac{\partial}{\partial\theta}\mathbb{E}_{x\sim p(x)}[f(g(x,\theta))] &= \frac{\partial}{\partial\theta}\int_x f(g(x,\theta))p(x)\mathrm{d}x \\
&= \int_x \left[\frac{\partial}{\partial\theta} f(g(x,\theta))\right] p(x)\mathrm{d}x \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\theta} f(g(x^{(i)},\theta)) \qquad x\sim p(x) \\
&= \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta f(g(x^{(i)},\theta)) \qquad \text{use shorthand notation: } \nabla_\theta[\cdot] \equiv \frac{\partial}{\partial\theta}[\cdot]
\end{aligned}
$$

- imagine: $p(x)$ is also paramterized by θ

$$
\begin{aligned}
\mathbb{E}_{x\sim p(x|\theta)}[f(g(x,\theta))] &= \int_x f(g(x,\theta))p(x|\theta)\mathrm{d}x \\
\Longrightarrow \frac{\partial}{\partial\theta}\mathbb{E}_{x\sim p(x|\theta)}[f(g(x,\theta))] &= \frac{\partial}{\partial\theta}\int_x f(g(x,\theta))p(x|\theta)\mathrm{d}x \\
&= \int_x \nabla_\theta\left[f(g(x,\theta))p(x|\theta)\right]\mathrm{d}x
\end{aligned}
$$

re-parameterization trick: example (1)

- let $\mu(\theta) = a\theta + b$, and $\sigma^2(\theta) = 1$, and we would like to compute:

$$\theta^* = \arg\max_\theta \left[\int_z \underbrace{z^2}_{f(z)} \mathcal{N}\Big(\underbrace{a\theta + b}_{\mu(\theta)}, \underbrace{1}_{\sigma^2(\theta)} \Big) \right]$$

- in words, it says: find mean of Gaussian, so the "expected square of samples" from this Gaussian are minimized; it's obvious that you want μ to close to **zero**
- which implies $\theta = -\frac{b}{a}$
- to solve it by gradient decend in a **traditional way**, we need to let $\mathcal{L}_\theta(z) = \log \left[\frac{1}{\sigma\sqrt{2\pi}} \exp{-\frac{(z-\mu)^2}{2\sigma^2}} \right]$:

$$\begin{aligned} \nabla_\theta \mathcal{L}_\theta(z) \equiv \nabla_\mu \mathcal{L}_\mu(z) &= \frac{\partial}{\partial \mu} (\mathcal{L}_\theta(x)) \times \frac{\partial \mu(\theta)}{\theta} \\ &= (z - \mu(\theta)) \times a = a(z - a\theta - b) \end{aligned}$$

- to substitute it into derivative:

$$\begin{aligned} \nabla_\theta \mathbb{E}_{z \sim p_\theta(z)}[f(z)] &\equiv \mathbb{E}_{z \sim \mathcal{N}(z; a\theta + b\theta, 1)}[z^2 \nabla_\theta \mathcal{L}_\theta(z)] \\ &= \mathbb{E}_{z \sim \mathcal{N}(z; a\theta + b, 1)}[z^2 a(x - a\theta - b)] \end{aligned}$$

re-parameterization trick: example (2)

- $z \sim \mathcal{N}(z; \mu(\theta), \sigma^2(\theta))$ can be **re-parameterised** into:
- if we need to compute: $f(z) = z^2$

$$x \sim \mathcal{N}(0, 1)$$
$$z \equiv g(x, \theta) = \mu(\theta) + x\sigma^2(\theta)$$

- the re-parameterised version is:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x\sim p(x)}[f(g(x, \theta))] &\equiv \mathbb{E}_{x\sim\mathcal{N}(x;0,1)}\left[\nabla_\theta\left(z^2\right)\right] \\
&= \mathbb{E}_{x\sim\mathcal{N}(x;0,1)}\left[\nabla_\theta\left(\mu(\theta) + x\sigma^2(\theta)\right)^2\right] \\
&= \mathbb{E}_{x\sim\mathcal{N}(x;0,1)}\left[\nabla_\theta\left(a\theta + b + x\right)^2\right] \\
&= \mathbb{E}_{x\sim\mathcal{N}(x;0,1)}\left[2a(a\theta + b + x)\right]
\end{aligned}$$

- both of them must achieve the same result!
- knowing $p(X)$ and $g(x, \theta)$ is sufficient, we do **not** need to know explicitly $p(Z)$

another example of re-parameterisation: variational auto-encoder

- **encoder** $x \rightarrow z$
- **decoder** $z \rightarrow x'$, such you want x and x' to be as close as possible
- autoencoders generate things "as it is"

would be better, if we could feed z to **decoder** that **were not** encoded from the images in actual dataset

- then, we can create new, reasonable images
- an idea: when feed database of images $\{x\}$ to encoder, the corresponding $\{z\}$ are "forced into" to form a distribution, so that a **new** sample z' randomly drawn from this distribution creates a reasonable image

variational auto-encoder

- loss at a particular data point x_i:

$$l_i(\theta, \phi) = \underbrace{-\mathbb{E}_{z \sim Q_\theta(z|x_i)} \big[\log P_\phi(x_i|z) \big]}_{\text{reconstruction error}} + \underbrace{\text{KL}(Q_\theta(z|x_i)||p(z))}_{\text{regularizer}}$$

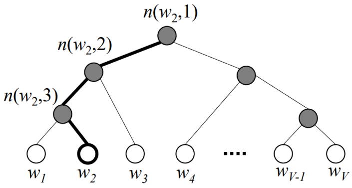
- we want $\mathbb{E}_{z \sim Q_\theta(z|x_i)} \big[\log P_\phi(x_i|z) \big]$ to be high, it needs for:
- $Q_\theta(z|x_i) \uparrow \implies P_\phi(x_i|z) \uparrow$ and $Q_\theta(z|x_i) \downarrow \implies P_\phi(x_i|z) \downarrow$
- therefore, the optimal solution may be for $Q_\theta(z|x_i)$ and $P_\phi(x_i|z)$ to be just a single delta function in a $x - z$ plane
- and all rest of $\{x, z\}$ are delta functions lies on a monotonic curve on the $x - z$ plane
- regularizer $\text{KL}(Q_\theta(z|x_i)||P(z))$ ensure $Q_\theta(z|x_i)$ doesn't behalf the above, i.e., $Q_\theta(z|x_i)$ are distributed as close to Gaussian distribution as possible
- $P_\phi(x_i|z)$ is just supervised learning: pixel value x_i is its label/value

objective function illustration

- loss at loss function again:

$$l_i(\theta, \phi) = \underbrace{-\mathbb{E}_{z \sim Q_\theta(z|x_i)}\Big[\log P_\phi(x_i|z)\Big]}_{\text{reconstruction error}} + \underbrace{\mathrm{KL}(Q_\theta(z|x_i)||p(z))}_{\text{regularizer}}$$

- without reconstruction loss, same numbers may not be close together, i.e., they spread across the entire multivariate normal distribution (so $P_\phi(Q_\theta(x))$ doesn't look like x
- without regularizer, same digits will stick together, but they don't form overall multivariate Gaussian distribution (so you can't sample)



- a bit of mathematics:

$$
\begin{aligned}
\mathrm{KL}\big(Q(z|X)\|P(z|X)\big) &= \mathbb{E}_{Z\sim Q(z|X)}\left[\log Q(z|X) - \log \frac{P(X|z)P(z)}{P(X)}\right] \\
&= \mathbb{E}_{Z\sim Q(z|X)}\big[\log Q(z|X) - (\log P(X|z) + \log P(z) - \log P(X))\big] \\
&= \mathbb{E}_{Z\sim Q(z|X)}\big[\log Q(z|X) - \log P(X|z) - \log P(z) + \log P(X)\big] \\
&= \mathbb{E}_{Z\sim Q(z|X)}\big[\log Q(z|X) - \log P(X|z) - \log P(z)\big] + \log P(X) \\
\mathrm{KL}\big(Q(z|X)\|P(z|X)\big) - \log P(X) &= \mathbb{E}_{Z\sim Q(z|X)}\big[\log Q(z|X) - \log P(X|z) - \log P(z)\big] \\
\log P(X) - \mathrm{KL}\big(Q(z|X)\|P(z|X)\big) &= \mathbb{E}_{Z\sim Q(z|X)}\big[\log P(X|z) - \log Q(z|X) + \log P(z)\big] \\
&= \mathbb{E}_{Z\sim Q(z|X)}\big[\log P(X|z)\big] - \mathbb{E}_{Z\sim Q(z|X)}\big[\log Q(z|X) - \log P(z)\big] \\
&= \mathbb{E}_{Z\sim Q(z|X)}\big[\log P(X|z)\big] - \mathrm{KL}\big(Q(z|X)\|P(z)\big)
\end{aligned}
$$

- it shows that:

$$
\begin{aligned}
\min \Big(l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + \mathrm{KL}(q_\theta(z|x_i)||p(z)) \Big) \\
\implies \text{minimize} \quad \mathrm{KL}\big(Q(z|X)\|P(z|X)\big)
\end{aligned}
$$

KL between two Gaussian distributions

- compute $\text{KL}\left(\mathcal{N}(\mu_1, \Sigma_1) \| \mathcal{N}(\mu_2, \Sigma_2)\right)$

$$
\begin{aligned}
\text{KL} &= \int_x \left[\frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2}(x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + \frac{1}{2}(x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2) \right] \times p(x) \mathrm{d}x \\
&= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} \text{tr} \left\{ \mathbb{E}[(x - \mu_1)(x - \mu_1)^T] \, \Sigma_1^{-1} \right\} + \frac{1}{2} \mathbb{E}[(x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2)] \\
&= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} \text{tr} \left\{ I_d \right\} + \frac{1}{2} (\mu_1 - \mu_2)^T \Sigma_2^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right]
\end{aligned}
$$

- substitute $\mu_2 = 1$ for each dimension, $\Sigma_2 = \text{I}$ is a Σ_2 is a diagonal matrix:

$$
\begin{aligned}
\text{KL}[N(\mu(X), \Sigma(X)) \| N(0, 1)] &= \frac{1}{2} \left(\text{tr}(\Sigma(X)) + \mu(X)^T \mu(X) - k - \log \det(\Sigma(X)) \right) \\
&= \frac{1}{2} \left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \log \prod_k \Sigma(X) \right) \\
&= \frac{1}{2} \left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \sum_k \log \Sigma(X) \right) \\
&= \frac{1}{2} \sum_k \left(\Sigma(X) + \mu^2(X) - 1 - \log \Sigma(X) \right)
\end{aligned}
$$

there is a simpler way to compute KL

- encoder net $Q(Z|x)$ takes input X and output two things: μ and Σ
-

where does neural network come in to play?

- to do Bayesian properly, we need:

$$P(z|x_i) \propto \underbrace{P_\theta(x_i|z)}_{\text{Encoder network}} \underbrace{P(z)}_{\mathcal{N}(0,I)}$$

- this is certainly not Gaussian! therefore, we need to use variational approach, and to define $Q_\theta(z|x_i) \equiv \mathcal{N}\big(\mu(x_i,\theta), \Sigma(x_i,\theta)\big)$
- we can choose any distribution, but having Normal distribution making KL computation a lot easier in objective function
- how do we obtain the parameter value of this Gaussian?
- of course a linear, or a kernel won't do its trick, we need a Neural Network for both $\mu(x_i,\theta), \Sigma(x_i,\theta)$

apply re-parameterization trick to softmax

- when we have the following

$$\mathbb{E}_{K\sim\text{softmax}(\mu_1(\theta),\dots,\mu_L(\theta))}[f(\mathbf{v}(K))] = \sum_{k=1}^{L} f(\mathbf{v}(k)) \Pr(k|\theta) \equiv \sum_{k=1}^{L} f(\mathbf{v}(k)) \big(\text{softmax}(\mu_1(\theta), \dots, \mu_L(\theta))\big)_k$$

- can we find their corresponding:

$$k = g(\mathcal{G}, \theta) \qquad \mathcal{G} \sim p(\mathcal{G})$$

Back to the Gumbel-max trick (1)

- Gumbel-max trick also means:

$$\underbrace{U \sim \mathcal{U}(0,1) \qquad \mathcal{G} = -\log(-\log(U))}_{p(\mathcal{G})}$$

$$k = \underbrace{\operatorname*{arg\,max}_{i \in \{1,\dots,K\}} \{\mu_1(\theta) + \mathcal{G}, \dots, \mu_K(\theta) + \mathcal{G}\}}_{g(\mathcal{G},\theta)} \qquad \mathbf{v} = \text{one-hot}(k)$$

- this is a form of re-paramterization:

  instead of sample $\mathbf{v} \sim \text{softmax}(\mu_1(\theta), \dots, \mu_K(\theta))$, we sample $\mathcal{G}$ instead

- the only remaining **problem**: sample **v** also has an $\arg\max$ operation, it's a discrete distribution!
- one can **relax** the softmax distribution, for example **softmax map**

Back to the Gumbel-max trick (2)

- **softmax map**

$$f_\tau(x)_k = \frac{\exp(\mu_k/\tau)}{\sum_{k=1}^K \exp(\mu_k/\tau)} \qquad \mu_k \equiv \mu_k(x_k)$$

- the new density:

$$p_{\mu,\tau}(x) = (n-1)!\tau^{n-1} \prod_{k=1}^K \left(\frac{\exp(\mu_k) x_k^{-\tau-1}}{\sum_{i=1}^K \exp(\mu_i) x_i^{-\tau}} \right), x \in \Delta^{K-1}$$

- check to see when $\tau = 1$:

$$\implies p_{\alpha,1}(x) = (n-1)! \prod_{k=1}^K \left(\frac{\exp(\mu_k)}{\sum_{i=1}^K \exp(\mu_i)} \right)$$

- then we can apply the same softmax map with added Gumbel variables:

$$(X_k^\tau)_k = f_\tau(\mu + G)_k = \left(\frac{\exp(\mu_k + G_k)/\tau)}{\sum_{i=1}^K \exp(\mu_i + G_i)/\tau)} \right)_k$$