Selected topics in Probabilities and Neural Networks: (deeCamp 2018 & Peking University, China)

A/Prof Richard Yi Da Xu
richardxu.com

University of Technology Sydney (UTS)

July 24, 2018

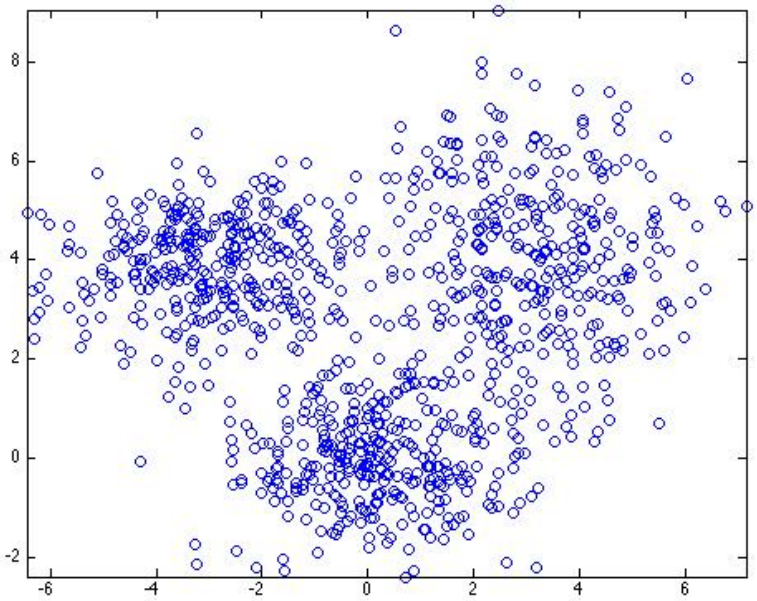
Motivations

- Probabilities and statistics are a significant part of Machine Learning
- in this class, we discuss about how probabilities and neural networks help each other, under the following four scenarios:
- This course assumes DeeCamp students are up-to-date with deep learning

1. Expectation-Maximization $\longrightarrow$ Matrix Capsule Networks
2. Determinantal Point Process $\longrightarrow$ Neural Networks compression
3. Kalman Filter $\longleftarrow$ LSTM
4. Probability model estimation $\longleftarrow$ Binary classifier
5. Probability density re-parameterization
6. Stochastic matrices and Monte Carlo Inference

Expectation-Maximization and Matrix Capsule Networks

A typical motivation of EM - Mixture Densities Model



- When you have data that looks like this figure
- Can you fit them using a single-mode Gaussian distribution, i.e.,:

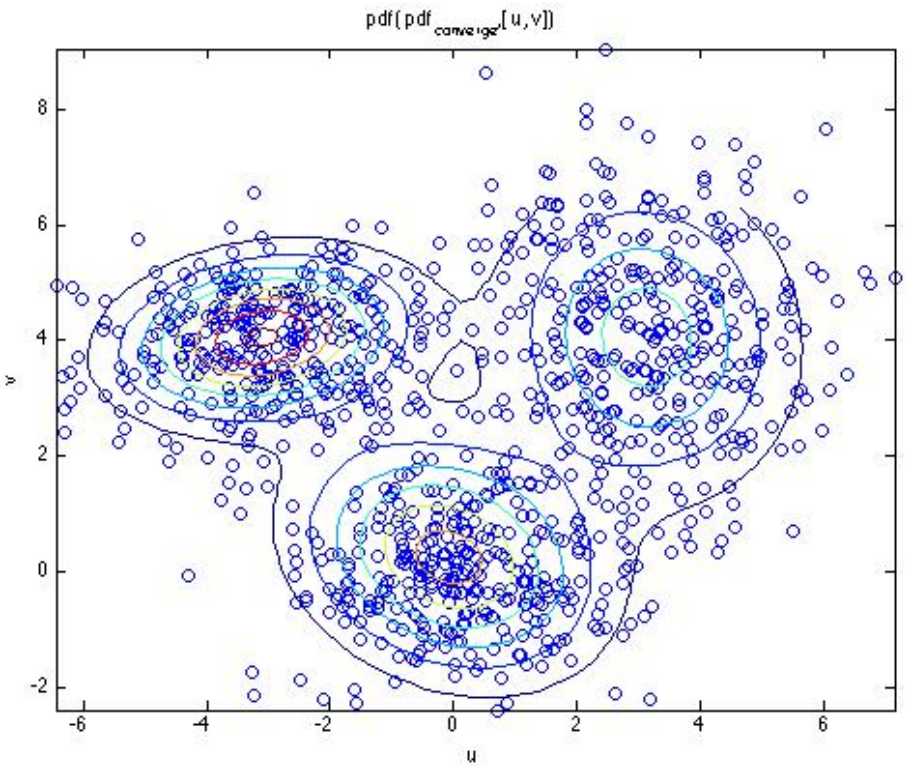
$$p(X) = \mathcal{N}(X|\mu, \Sigma)$$
$$= (2\pi)^{-k/2}|\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- Clearly NOT!

- This is typically modeled using **Mixture Densities**, in the case of Gaussian Mixture Model (k-mixture) (GMM):

$$p(X) = \sum_{l=1}^{k} \alpha_l \mathcal{N}(X|\mu_l, \Sigma_l) \qquad \sum_{l=1}^{k} \alpha_l = 1$$

Gaussian Mixture model result



Let $\Theta = \{\alpha_1, \ldots \alpha_k, \mu_1, \ldots \mu_k, \Sigma_1, \ldots \Sigma_k\}$

$$\Theta_{\text{MLE}} = \arg\max_{\Theta} \mathcal{L}(\Theta|X)$$

$$= \arg\max_{\Theta} \left(\sum_{i=1}^{n} \log \sum_{l=1}^{k} \alpha_l \mathcal{N}(x_i|\mu_l, \Sigma_l) \right)$$

- Unlike single mode Gaussian, solving the equation analytically is difficult
- this is where Expectation-Maximization is there to help

The Expectation-Maximization Algorithm

- Instead of perform:

$$\theta^{\text{MLE}} = \arg\max_{\theta}(\mathcal{L}(\theta)) = \arg\max_{\theta}\left(\log[p(X|\theta)]\right)$$

  - **The trick** is to assume some "latent" variable Z to the model.
  - such that we generate a series of $\Theta = \{\theta^{(1)}, \theta^{(2)}, \ldots \theta^{(t)}\}$

- For each iteration of the E-M algorithm, we perform:

$$\Theta^{(g+1)} = \arg\max_{\theta}\left(\int_{z} \log\left(p(X, Z|\theta)\right) p(Z|X, \Theta^{(g)})\right) \mathrm{d}z$$

- However, we must ensure convergence:

$$\log[p(X|\Theta^{(g+1)})] = \mathcal{L}(\Theta^{(g+1)}) \geq \mathcal{L}(\Theta^{(g)}) \qquad \forall i$$

Proof of convergence (1)

$$\mathcal{L}(\theta|X) = \ln[p(X|\theta)] = \ln[p(Z,X,\theta)] - \ln[p(Z|X,\theta)]$$

$$\implies \int_{z\in\mathbb{S}} \ln[p(X|\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z$$

$$= \int_{z\in\mathbb{S}} \ln[p(Z,X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z - \int_{z\in\mathbb{S}} \ln[p(Z|X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z$$

$$\implies \ln[p(X|\theta)] = \underbrace{\int_{z\in\mathbb{S}} \ln[p(Z,X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z}_{Q(\theta,\theta^{(g)})} - \underbrace{\int_{z\in\mathbb{S}} \ln[p(Z|X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z}_{H(\theta,\Theta^{(g)})}$$

In E-M, we only maximise, i.e., $\Theta^{(g+1)} = \arg\max_\theta Q(\theta,\Theta^{(g)})$. Why? **a trick** If we can prove:

$$\arg\max_\theta \left[\int_{z\in\mathbb{S}} \ln[p(Z|X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z\right] = \Theta^{(g)} \implies H(\Theta^{(g+1)},\Theta^{(g)}) \leq H(\Theta^{(g)},\Theta^{(g)})$$

Then

$$\mathcal{L}(\Theta^{(g+1)}) = \underbrace{Q(\Theta^{(g+1)},\Theta^{(g)})}_{\geq Q(\Theta^{(g)},\Theta^{(g)})} - \underbrace{H(\Theta^{(g+1)},\Theta^{(g)})}_{\leq H(\Theta^{(g)},\Theta^{(g)})} \geq Q(\Theta^{(g)},\Theta^{(g)}) - H(\Theta^{(g)},\Theta^{(g)}) = \mathcal{L}(\Theta^{(g)})$$

Proof of convergence (2)

$$\text{To prove} \quad \arg\max_{\theta}[H(\theta, \Theta^{(g)})] = \arg\max_{\theta}\left[\int_{z\in\mathbb{S}} \ln[p(Z|X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z\right] = \Theta^{(g)}$$

$$\implies \text{To prove} \quad H(\Theta^{(g)}, \Theta^{(g)}) - H(\theta, \Theta^{(g)}) \geq 0 \quad \forall\theta$$

$$H(\Theta^{(g)}, \Theta^{(g)}) - H(\theta, \Theta^{(g)}) = \int_{z\in\mathbb{S}} \ln[p(Z|X,\Theta^{(g)})]p(z|X,\Theta^{(g)})\mathrm{d}z - \int_{z\in\mathbb{S}} \ln[p(Z|X,\theta)]p(z|X,\Theta^{(g)})\mathrm{d}z$$

$$= \int_{z\in\mathbb{S}} \ln\left[\frac{p(Z|X,\Theta^{(g)})}{p(Z|X,\theta)}\right] p(z|X,\Theta^{(g)})\mathrm{d}z = \int_{z\in\mathbb{S}} -\ln\left[\frac{p(Z|X,\theta)}{p(Z|X,\Theta^{(g)})}\right] p(z|X,\Theta^{(g)})\mathrm{d}z$$

$$\geq -\ln\left[\int_{z\in\mathbb{S}} \frac{p(Z|X,\theta)}{p(Z|X,\Theta^{(g)})} p(z|X,\Theta^{(g)})\mathrm{d}z\right] = 0$$

Since $\Phi(.) = -\ln$ is a convex function:

E-M Example: Gaussian Mixture Model

- Gaussian Mixture Model (k-mixture) (GMM):

$$p(X|\Theta) = \sum_{l=1}^{k} \alpha_l \mathcal{N}(X|\mu_l, \Sigma_l) \qquad \sum_{l=1}^{k} \alpha_l = 1$$
$$\text{and } \theta = \{\alpha_1, \ldots \alpha_k, \mu_1, \ldots \mu_k, \Sigma_1, \ldots \Sigma_k\}$$

- For data $X = \{x_1, \ldots x_n\}$ we introduce "latent" variable $Z = \{z_1, \ldots z_n\}$, each z_i indicates which mixture component x_i belong to.
- Looking at the E-M algorithm:

$$\Theta^{(g+1)} = \arg\max_{\Theta} \left[Q(\Theta, \Theta^{(g)}) \right] = \arg\max_{\Theta} \left(\int_z \log \left(p(X, Z|\Theta) \right) p(Z|X, \Theta^{(g)}) \mathrm{d}z \right)$$

- We need to define both $p(X, Z|\Theta)$ and $p(Z|X, \Theta)$

Gaussian Mixture Model in action

$$p(X|\Theta) = \sum_{l=1}^{k} \alpha_l \mathcal{N}(X|\mu_l, \Sigma_l) = \prod_{i=1}^{n} \sum_{l=1}^{k} \alpha_l \mathcal{N}(X|\mu_l, \Sigma_l)$$

How to define $p(X, Z|\Theta)$

$$p(X, Z|\Theta) = \prod_{i=1}^{n} p(x_i, z_i|\Theta) = \prod_{i=1}^{n} \underbrace{p(x_i|z_i, \Theta)}_{\mathcal{N}(\mu_{z_i}, \Sigma_{z_i})} \underbrace{p(z_i|\Theta)}_{\alpha_{z_i}} = \prod_{i=1}^{n} \alpha_{z_i} \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Notice that $p(X, Z|\Theta)$ is actually simpler than $p(X|\Theta)$.

How to define $p(Z|X, \Theta)$

$$p(Z|X, \Theta) = \prod_{i=1}^{n} p(z_i|x_i, \Theta) = \prod_{i=1}^{n} \frac{\alpha_{z_i} \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})}{\sum_{l=1}^{k} \alpha_l \mathcal{N}(\mu_l, \Sigma_l)}$$

The E-Step: (1)

$$
\begin{aligned}
Q(\Theta, \Theta^{(g)}) &= \int_z \ln\left(p(X, Z|\Theta)\right) p(Z|X, \Theta^{(g)}) \mathrm{d}z \\
&= \int_{z_1} \cdots \int_{z_n} \left(\sum_{i=1}^{n} \ln p(z_i, x_i|\Theta) \prod_{i=1}^{n} p(z_i|x_i, \Theta^{(g)}) \right) \mathrm{d}z_1, \ldots \mathrm{d}z_n
\end{aligned}
$$

The E-Step: (2)

Knowing,

$$\int_{y_1} \cdots \int_{y_n} \left(\sum_{i=1}^{n} f_i(y_i) \right) P(Y) \mathrm{d}Y = \sum_{i}^{N} \left(\int_{y_i} f_i(y_i) P_i(y_i) \mathrm{d}y_i \right)$$

$$\begin{aligned}
Q(\Theta, \Theta^{(g)}) &= \int_{z_1} \cdots \int_{z_n} \left(\sum_{i=1}^{n} \ln p(z_i, x_i|\Theta) \prod_{i=1}^{n} p(z_i|x_i, \Theta^{(g)}) \right) \mathrm{d}z_1, \ldots \mathrm{d}z_n \\
&= \sum_{i=1}^{n} \left(\int_{z_i} \ln p(z_i, x_i|\Theta) p(z_i|x_i, \Theta^{(g)}) \mathrm{d}z_i \right) \qquad z_i \in \{1, \ldots, k\} \\
&= \sum_{z_i=1}^{k} \sum_{i=1}^{n} \ln p(z_i, x_i|\Theta) p(z_i|x_i, \Theta^{(g)}) \qquad \text{swap the summation terms} \\
&= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln[\alpha_l \mathcal{N}(x_i|\mu_l, \Sigma_l)] p(l|x_i, \Theta^{(g)}) \qquad \text{substitute Gaussian and replace } z_i \to l
\end{aligned}$$

The M-Step objective function

$$Q(\Theta, \Theta^{(g)}) = \sum_{l=1}^{k} \sum_{i=1}^{n} \ln[\alpha_l \mathcal{N}(x_i|\mu_l, \Sigma_l)] p(l|x_i, \Theta^{(g)})$$
$$= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln(\alpha_l) p(l|x_i, \Theta^{(g)}) + \sum_{l=1}^{k} \sum_{i=1}^{n} \ln[\mathcal{N}(x_i|\mu_l, \Sigma_l)] p(l|x_i, \Theta^{(g)})$$

- The first term contains only α
- second term contains only μ, Σ.
- So we can maximize both terms independantly.

The M-Step: maximizing α

Maximizing α means that:

$$\frac{\partial \sum_{l=1}^{k} \sum_{i=1}^{n} \ln(\alpha_l) p(l|x_i, \Theta^{(g)})}{\partial \alpha_1, \ldots, \partial \alpha_k} = [0 \ldots 0] \qquad \text{subject to } \sum_{l=1}^{k} \alpha_l = 1$$

This is to be solved using Lagrange Multiplier

$$\mathbb{LM}(\alpha_1, \ldots \alpha_k, \lambda) = \sum_{l=1}^{k} \ln(\alpha_l) \underbrace{\left(\sum_{i=1}^{n} p(l|x_i, \Theta^{(g)}) \right)}_{\text{contains no } \alpha} - \lambda \left(\sum_{l=1}^{k} \alpha_l - 1 \right)$$

$$\implies \frac{\partial \mathbb{LM}}{\partial \alpha_l} = \frac{1}{\alpha_l} \left(\sum_{i=1}^{n} p(l|x_i, \Theta^{(g)}) \right) - \lambda = 0$$

$$\implies \alpha_l = \frac{1}{N} \sum_{i=1}^{n} p(l|x_i, \Theta^{(g)})$$

The M-Step: maximizing μ, Σ

$$
\begin{aligned}
&\text{second part of } Q(\Theta, \Theta^{(g)}) \\
\equiv \mathcal{S}(\mu_l, \mathbf{\Sigma}_l) &= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln[\mathcal{N}(x_i|\mu_l, \mathbf{\Sigma}_l)] p(l|x_i, \Theta^{(g)}) \\
&= \sum_{i=1}^{n} \sum_{l=1}^{k} \ln \left(\frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}_l|}} \exp \left(-\frac{1}{2} (x_i - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1} (x_i - \boldsymbol{\mu}) \right) \right) p(l|x_i, \Theta^{(g)}) \\
&= \sum_{i=1}^{n} -\frac{1}{2} \ln(|\mathbf{\Sigma}_l|) p(l|x_i, \Theta^{(g)}) - \sum_{i=1}^{n} \frac{1}{2} (x_i - \boldsymbol{\mu}_l)^T \mathbf{\Sigma}^{-1} (x - \boldsymbol{\mu}_l) p(l|x_i, \Theta^{(g)})
\end{aligned}
$$

how do we maximize μ, Σ?

Some formulas to remember, from Matrix Cookbook

- derivatives of log of determinant (**with** determinant)

$$\frac{\partial \ln |\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^\top$$

- Derivatives of Traces

$$\frac{\partial \mathrm{tr}(F(\mathbf{X}))}{\partial \mathbf{X}} = (f(\mathbf{X}))^\top$$

where $f(\cdot)$ is the scalar derivative of $F(\cdot)$

- Derivatives of Traces of inverse, fact 1

$$\frac{\partial \mathrm{tr}(\mathbf{AXB})}{\partial \mathbf{X}} = \mathbf{A}^\top \mathbf{B}^\top$$

- Derivatives of Traces of inverse, fact 2

$$\frac{\partial \mathrm{tr}((\mathbf{X}+\mathbf{A})^{-1})}{\partial \mathbf{X}} = -((\mathbf{X}+\mathbf{A})^{-1}(\mathbf{X}+\mathbf{A})^{-1})^\top$$

- Derivatives of Traces of inverse, fact 3

$$\frac{\partial \mathrm{tr}(\mathbf{AX}^{-1}\mathbf{B})}{\partial \mathbf{X}} = -(\mathbf{X}^{-1}\mathbf{BAX}^{-1})^\top$$

Maximization μ_l

$$\mathcal{S}(\mu_l, \mathbf{\Sigma}_l) = \sum_{i=1}^{n} -\frac{1}{2} \ln(|\mathbf{\Sigma}_l|) p(l|x_i, \Theta^{(g)}) - \sum_{i=1}^{n} \frac{1}{2} (x_i - \boldsymbol{\mu}_l)^T \mathbf{\Sigma}^{-1} (x - \boldsymbol{\mu}_l) p(l|x_i, \Theta^{(g)})$$

$$= -\text{Tr}\left(\frac{\Sigma_l^{-1}}{2} \sum_{i=1}^{n} (x_i - \mu_l)(x - \mu_l)^T p(l|x_i, \Theta^{(g)}) \right) + \text{ Constant}$$

$$\implies \frac{\partial \mathcal{S}(\mu_l, \Sigma_l)}{\partial \mu_l} = \frac{\Sigma^{-1}}{2} \sum_{i=1}^{n} 2(x_i - \mu_l) p(l|x_i, \Theta^{(g)}) = 0$$

$$\implies \sum_{i=1}^{n} x_i p(l|x_i, \Theta^{(g)}) = \mu_l \sum_{i=1}^{n} p(l|x_i, \Theta^{(g)})$$

$$\implies \mu_l = \frac{\sum_{i=1}^{n} x_i p(l|x_i, \Theta^{(g)})}{\sum_{i=1}^{n} p(l|x_i, \Theta^{(g)})}$$

Maximization Σ_l

- let's try something **cool**, and try to obtain the answer using matrix form
- let $\mathcal{Y}$ be zero-meaned data matrix, where each column of $\mathcal{Y}$ is $x_i - \mu_l$
- let $\mathbf{P}$ be diagonal matrix in which $\mathbf{P}_{ii}$ correspond to $p(l|x_i, \Theta^{(g)})$

$$\begin{aligned}\mathcal{S}(\mu_l, \mathbf{\Sigma}_l) &= \sum_{i=1}^{n} -\frac{1}{2}\ln(|\mathbf{\Sigma}_l|)p(l|x_i, \Theta^{(g)}) - \sum_{i=1}^{n}\frac{1}{2}(x_i - \boldsymbol{\mu}_l)^T\mathbf{\Sigma}^{-1}(x - \boldsymbol{\mu}_l)p(l|x_i, \Theta^{(g)}) \\ &= -\frac{\text{tr}(\mathbf{P})}{2}\ln|\mathbf{\Sigma}_l| - \frac{1}{2}\text{tr}\left(\mathbf{\Sigma}_l^{-1}\mathcal{Y}\mathbf{P}\mathcal{Y}^\top\right)\end{aligned}$$

- now taking the derivative:

$$\begin{aligned}\frac{\partial\mathcal{S}(\mu_l, \mathbf{\Sigma}_l)}{\partial\mathbf{\Sigma}_l} &= \mathbf{\Sigma}_l{}^{-1}\mathcal{Y}\mathbf{P}\mathcal{Y}^\top\mathbf{\Sigma}_l^{-1} - \text{tr}(\mathbf{P})\mathbf{\Sigma}_l^{-1} = \mathbf{0} \\ \implies \mathbf{\Sigma}_l{}^{-1}\mathcal{Y}\mathbf{P}\mathcal{Y}^\top\mathbf{\Sigma}_l^{-1} &= \text{tr}(\mathbf{P})\mathbf{\Sigma}_l^{-1} \\ \implies \mathbf{\Sigma}_l^{-1}\mathcal{Y}\mathbf{P}\mathcal{Y}^\top &= \text{tr}(\mathbf{P}) \\ \implies \mathbf{\Sigma}_l^{-1} &= \text{tr}(\mathbf{P})(\mathcal{Y}\mathbf{P}\mathcal{Y}^\top)^{-1} \\ \implies \mathbf{\Sigma}_l &= \text{tr}(\mathbf{P})^{-1}(\mathcal{Y}\mathbf{P}\mathcal{Y}^\top) = \frac{(\mathcal{Y}\mathbf{P}\mathcal{Y}^\top)}{\text{tr}(\mathbf{P})} \\ \text{or, } \mathbf{\Sigma}_l &= \frac{\sum_{i=1}^{n}(x_i - \mu_l)(x - \mu_l)^T p(l|x_i, \Theta^{(g)})}{\sum_{i=1}^{n} p(l|x_i, \Theta^{(g)})}\end{aligned}$$

Summary of Gaussian Mixture Model

- Maximizing μ, Σ, α to update $\Theta^{(g)} \rightarrow \Theta^{(g+1)}$:

$$\alpha_l^{(g+1)} = \frac{1}{N} \sum_{i=1}^{N} p(l|x_i, \Theta^{(g)})$$

$$\mu_l^{(g+1)} = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^{(g)})}{\sum_{i=1}^{N} p(l|x_i, \Theta^{(g)})}$$

$$\Sigma_l^{(g+1)} = \frac{\sum_{i=1}^{N} [x_i - \mu_l^{(i+1)}][x_i - \mu_l^{(i+1)}]^T p(l|x_i, \Theta^{(g)})}{\sum_{i=1}^{N} p(l|x_i, \Theta^{(g)})}$$

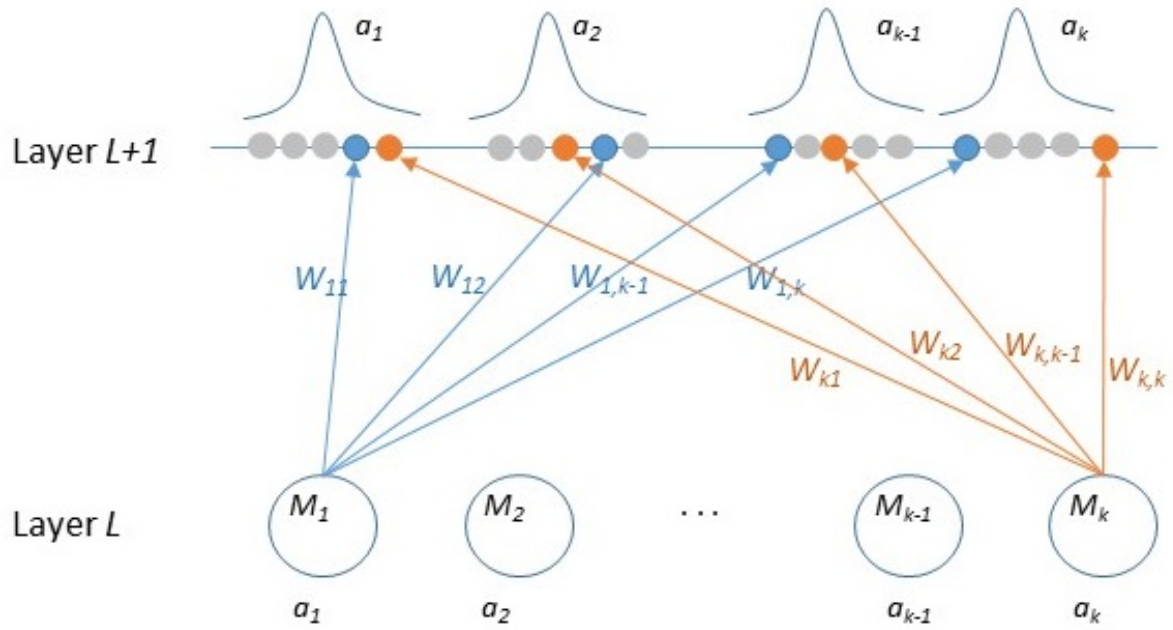
- responsibility probabilities:

$$p(l|x_i, \Theta^{(g)}) = \frac{\alpha_l \mathcal{N}(x_i|\mu_l, \Sigma_l)}{\sum_{s=1}^{k} \alpha_s \mathcal{N}(x_i|\mu_s, \Sigma_s)}$$

Apply EM to Matrix Capsule Routing algorithm

- Introduction of Capsule Networks, dynamic Routing and Matrix Capsule
- "*Hinton., et. al, (2018), Matrix capsules with EM routing*

Apply EM to Matrix Capsule Routing algorithm



assume both lower layer L and higher layer $L+1$ have K neurons

- lower layer $\{M_1, \ldots M_k\}$ acts like data source
- each M_i generates multiple "new data" V_{ij} through its transformation matrix W_{ij}, i.e.,

$$V_{ij} = M_i W_{ij}$$

- E-M then tries to fit $\Theta \equiv \{\mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k\}$ at layer $L+1$ capsules.

Apply EM to Matrix Capsule Routing algorithm

- M-STEP:

$$\forall i \in \Omega_L : R_{ij} \leftarrow R_{ij} \times a_i \qquad \text{apply } a_i$$

$$\forall h : \mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$$

$$\forall h : \left(\sigma_j^h\right)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$$

$$\text{cost}^h \leftarrow \left(\beta_u + \log(\sigma_j^h)\right) \sum_i R_{ij} \qquad a_j \leftarrow \text{sigmoid}\left(\lambda(\beta_a - \sum_h \text{cost}^h)\right) \qquad \text{update } a_j$$

- E-STEP:

$$\forall j \sim \Omega_{L+1} : \mathbf{p}_j \leftarrow \prod_{h=1}^{H} \mathcal{N}(V_{ij}^h; \mu_j^h, \sigma_j^h)$$

$$\mathbf{R}_{ij} \leftarrow \frac{a_j p_j}{\sum_{k \in \Omega_{L+1}} a_k p_k} \qquad \text{apply } a_j$$

Apply EM to Matrix Capsule Routing algorithm

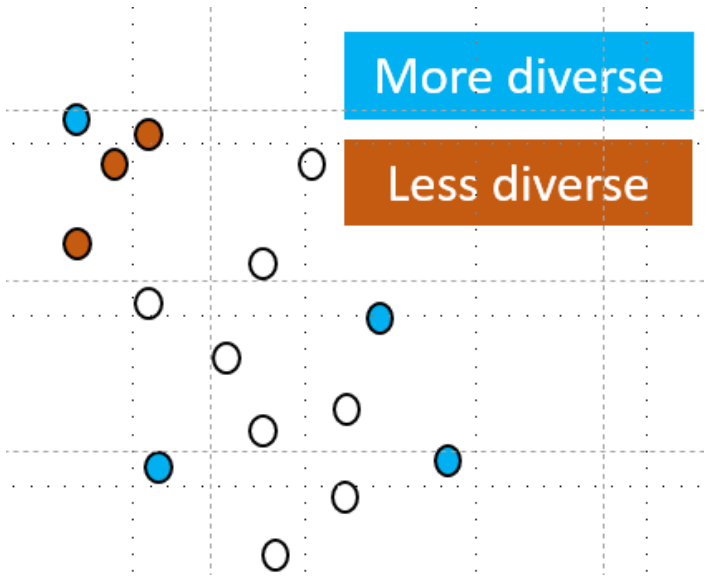
- cost $\mathcal{C}_j^h$ used as argument of sigmoid(.) and weighted from all lower capsules

$$
\begin{aligned}
\mathcal{C}_j^h(\mu, \sigma) &= \sum_i R_{ij} \mathcal{C}_{ij}^h \\
&= \sum_i -R_{ij} \ln(p_{i|j}^h) \\
&= \sum_i R_{ij} \left(\frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + \ln(\sigma_j^h) + \frac{\ln(2\pi)}{2} \right) \\
&= \sum_i R_{ij} \left(\frac{(\bar{\sigma}_j^h)^2}{2(\sigma_j^h)^2} + \ln(\sigma_j^h) + \frac{\ln(2\pi)}{2} \right) \\
&\approx \left(\ln(\sigma_j^h) + k \right) \sum_i R_{ij} \quad k \text{ is a constant}
\end{aligned}
$$

Determinantal Point Process and Neural Networks Compression

What is DPP?

- in **one sentence**: it's a probability defined on any subsets of N data points, such that a diverse subset attracts higher probability
- so, if it's a probability model, what is its parameter?
- we can either use a marginal kernel K, or to use an L-ensemble L
- let's look at marginal kernel first:



How do we define a DPP?

- Start with a **marginal** distribution of subset A

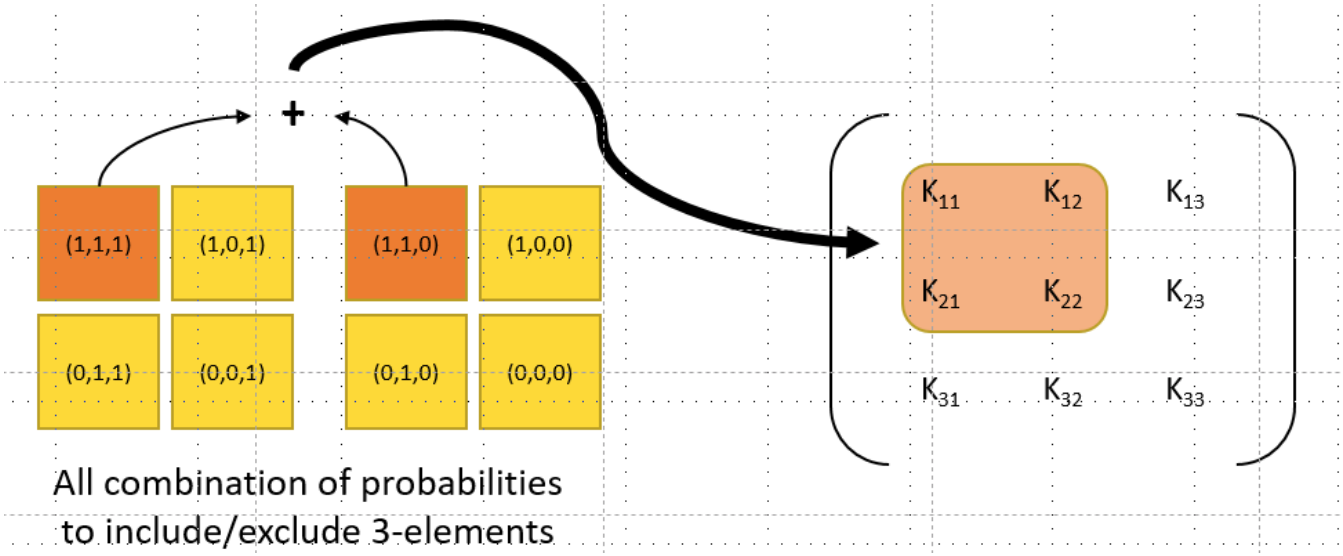
$$\Pr(A \subseteq \mathbf{Y}) = \det(K_A)$$

- An example: given $\mathcal{Y} = \{1, 2, 3, 4, 5\}$, $A = \{1, 2, 3\}$

$$\begin{aligned}\Pr(A \subseteq \mathbf{Y}) &\equiv \Pr(A \subseteq \mathbf{Y} \subseteq \mathcal{Y}) \equiv \Pr\left(\mathbf{Y} \in \{\{1,2,3\},\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,4,5\}\}\right) \\ &= \det(K_A)\end{aligned}$$

$$\begin{aligned}\Pr(A \subseteq \mathbf{Y}) &\equiv \Pr(A \subseteq \mathbf{Y} \subseteq \mathcal{Y}) \equiv \Pr\left(y_1 = 1, y_2 = 1, y_3 = 1\right) \\ &= \sum_{t_4=0}^{1}\sum_{t_5=0}^{1} \Pr\left(y_1 = 1, y_2 = 1, y_3 = 1, y_4 = t_4, y_5 = t_5\right) \\ &= \det(K_A)\end{aligned}$$

let's use a diagram!



- sum of probabilities to include 1st and 2nd elements
- This is defined by $\det(K_{\{1,2\}})$

Something about marginal distribution

- $\Pr(A \subseteq \mathbf{Y})$ is marginal, they don't need to add to 1
- it may be possible that, $\Pr(A_1 \subseteq \mathbf{Y}) + \Pr(A_2 \subseteq \mathbf{Y}) > 1$
- $\Pr(\emptyset \subseteq \mathbf{Y}) = \det(K_\emptyset) = 1$ This is obvious, as any $\mathbf{Y}$ is a superset of $\emptyset$.
- $\Pr(i \subseteq \mathbf{Y}) = \det(K_{ii}) = K_{ii}$
- Look at the two element case: **this is the key**

$$\begin{aligned}\Pr(i, j \in \mathbf{Y}) &= \begin{vmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{vmatrix} \\ &= K_{ii}K_{jj} - K_{ij}K_{ji} \\ &= \Pr(i \subseteq \mathbf{Y}) \Pr(j \subseteq \mathbf{Y}) - K_{ij}^2\end{aligned}$$

- Off-diagonal elements determine negative correlations between pairs.
- Large values of K_{ij} imply i and j tend **not** co-occur

Example of K does NOT define DPP

- Any $K, 0 \preceq K \preceq I$ defines a DPP.
- If $K \preceq K'$, that is, $K' - K$ is positive semidefinite.
  Therefore, $\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$ can NOT define DPP, as

$$\left| I - \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \right| = \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix} \implies \bar{\lambda} = [-0.5, 0.5]^\top$$

- Another way to see the above is incorrect:
  $\mathcal{Y} = \{1, 2\}$

$$\begin{aligned} \Pr(A = \{1\} \subseteq \mathbf{Y}) &\equiv \Pr\left(\mathbf{Y} \in \{\{1\}, \{1,2\}\}\right) \\ &= \det(K_1) = 1 \\ \Pr(A = \{2\} \subseteq \mathbf{Y}) &\equiv \Pr\left(\mathbf{Y} \in \{\{2\}, \{1,2\}\}\right) \\ &= \det(K_2) = 1 \end{aligned}$$

  However,

$$\begin{aligned} \Pr(A = \{1,2\} \subseteq \mathbf{Y}) &\equiv \Pr\left(\mathbf{Y} \in \{\{1,2\}\}\right) \\ &= \det(K_{\{1,2\}}) = 0.75 \end{aligned}$$

  The first two equation says $\{1\}$ and $\{2\}$ must be included; The third equation says both may NOT always be included.

Example of K define DPP

- Any $K, 0 \preceq K \preceq I$ defines a DPP.
- If $K \preceq K'$, that is, $K' - K$ is positive semidefinite.
  $\begin{bmatrix} 0.3 & -0.1 \\ -0.1 & 0.4 \end{bmatrix}$ **can** define DPP:

$$\left| I - \begin{pmatrix} 0.3 & -0.1 \\ -0.1 & 0.4 \end{pmatrix} \right| = \left| \begin{pmatrix} 0.7 & 0.1 \\ 0.1 & 0.6 \end{pmatrix} \right| \implies \bar{\lambda} = [0.5382, 0.7618]^\top$$

- $\mathcal{Y} = \{1, 2\}$

$$\begin{aligned} \Pr(A = \{1\} \subseteq \mathbf{Y}) &\equiv \Pr\left(Y \in \{\{1\}, \{1,2\}\}\right) \\ &= \det(K_1) = 0.3 \\ \Pr(A = \{2\} \subseteq \mathbf{Y}) &\equiv \Pr\left(Y \in \{\{2\}, \{1,2\}\}\right) \\ &= \det(K_2) = 0.4 \\ \Pr(A = \{1,2\} \subseteq \mathbf{Y}) &\equiv \Pr\left(Y \in \{\{1,2\}\}\right) \\ &= \det(K_{\{1,2\}}) = 0.11 \end{aligned}$$

- So where do rest of probabilities go?

$$\begin{aligned} \Pr(A = \emptyset \subseteq \mathbf{Y}) &\equiv \Pr\left(Y \in \{\emptyset, \{1\}, \{2\}, \{1,2\}\}\right) \\ &= \det(K_\emptyset) = 1 \end{aligned}$$

- Some probabilities mass is assigned to $\emptyset$.

L-Ensembles

- Marginal distributions does **not** define probabilities in terms of a **particular** set
- i.e., instead of having $\Pr(\mathbf{Y} \subseteq Y)$, we want $\Pr(\mathbf{Y} = Y)$

$$\Pr_L(\mathbf{Y} = Y) \propto \det(L_Y)$$

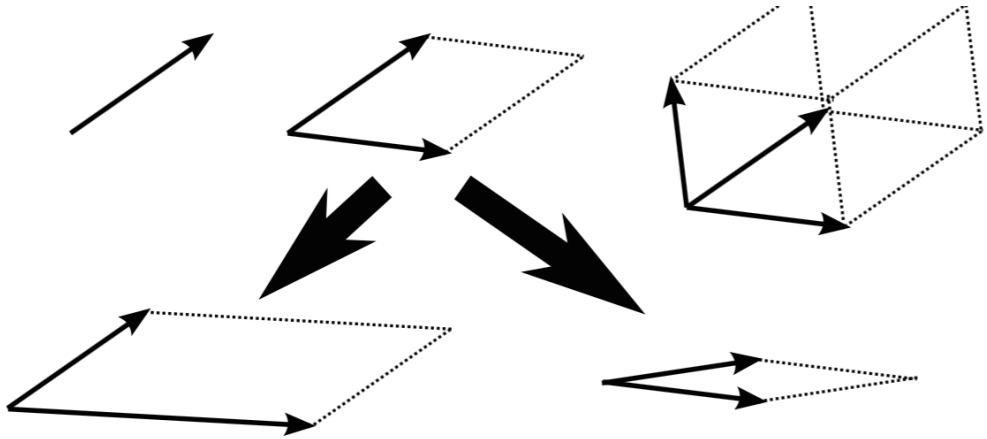
- L must be positive semidefinite.
- Only a statement of proportionality, eigenvalues of L need **not** < 1

Geometry interpretation

$$X = [x_1 \quad x_2 \quad \dots \quad x_n] \implies$$

$$L(x_1, \dots, x_n) = X^\top X = \begin{pmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \dots & \langle x_n, x_n \rangle \end{pmatrix}$$

- Gram determinant is the square of the volume of the parallelotope formed by the vectors
- vectors are linearly independent if and only if the Gram determinant is nonzero
- $\Pr_L(Y) \propto \det(L_Y) = \mathrm{Vol}^2\left(\{x_i\}_{i \in Y}\right)$



Proof for the Geometry interpretation (1)

- in **1-element** case: $\text{Vol}^2(\mathbf{u}_1) = \mathbf{u}_1^\top \mathbf{u}_1$, i.e., length square of a line
- in **k-element** case: $\text{Vol}^2(\mathbf{u}_1 \dots \mathbf{u}_k, \mathbf{u}_{k+1}) = \text{Vol}^2(\mathbf{u}_1, \dots, \mathbf{u}_k)\|\tilde{\mathbf{u}}_{k+1}\|^2$
- $\tilde{\mathbf{u}}_{k+1}$ is the orthogonal projection of $\mathbf{u}_{k+1}$ onto span $(\mathbf{u}_1, \dots, \mathbf{u}_k)$: imagine in in the **2-element** or **3-element** case.
- Let $(\mathbf{u}_1, \dots, \mathbf{u}_k)$ is an $n \times k$ matrix $\mathbf{Y}$:
- Then there exists a vector $\mathbf{c} \in \mathbb{R}^k$ such that:

$$\mathbf{u}_{k+1} = \mathbf{u}_{k+1} = \mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1}$$

$$= \underbrace{\begin{bmatrix} | & \vdots & | \\ \mathbf{u}_1 & \vdots & \mathbf{u}_k \\ | & \vdots & | \end{bmatrix}}_{\mathbf{Y}} \begin{bmatrix} c_1 \\ \cdot \\ c_k \end{bmatrix} + \tilde{\mathbf{u}}_{k+1} \qquad \text{or } \mathbf{u}_{k+1} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 \dots c_k\mathbf{u}_k + \tilde{\mathbf{u}}_{k+1}$$

Proof for the Geometry interpretation (2)

- extend $\mathbf{Y}$ to $\mathbf{X}$:

$$\mathbf{X} = [\mathbf{Y} \quad \mathbf{u}_{k+1}] = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \dots \quad \mathbf{u}_k \quad \mathbf{u}_{k+1}] = \begin{bmatrix} \mathbf{Y} & \mathbf{Yc} + \tilde{\mathbf{u}}_{k+1} \end{bmatrix}$$

$$\implies \mathbf{X}^\top \mathbf{X} = \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top (\mathbf{Yc} + \tilde{\mathbf{u}}_{k+1}) \\ (\mathbf{Yc} + \tilde{\mathbf{u}}_{k+1})^\top \mathbf{Y} & (\mathbf{Yc} + \tilde{\mathbf{u}}_{k+1})^\top (\mathbf{Yc} + \tilde{\mathbf{u}}_{k+1}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Yc} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Yc} + \tilde{\mathbf{u}}_{k+1}^\top \tilde{\mathbf{u}}_{k+1} \end{bmatrix} \quad \text{since } \mathbf{Y}^\top \tilde{\mathbf{u}}_{k+1} = \mathbf{0}$$

$$= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Yc} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Yc} + \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{bmatrix}$$

$$= \left[\begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} \end{bmatrix} \quad \left(\begin{bmatrix} \mathbf{Y}^\top \mathbf{Yc} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Yc} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{bmatrix} \right) \right]$$

- **Multi-linearity** states:

$$\det([a_1 + b_1, a_2, \dots, a_k]) = \det([a_1, a_2, \dots, a_k]) + \det([b_1, a_2, \dots, a_k])$$

$$\implies |\mathbf{X}^\top \mathbf{X}| = \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Yc} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Yc} \end{vmatrix} + \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{0} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{vmatrix}$$

$$= \mathbf{0} + \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{0} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{vmatrix}$$

$$= |\mathbf{Y}^\top \mathbf{Y}| \underbrace{\|\tilde{\mathbf{u}}_{k+1}\|^2}_{\text{Vol}^2(\tilde{\mathbf{u}}_{k+1})}$$

Normalization constant

Theorem says,

$$\sum_{A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I_{\bar{A}})$$

For example,

$$L = \begin{pmatrix} 2.8599 & -0.4936 & -1.8458 \\ -0.4936 & 2.6264 & -1.1437 \\ -1.8458 & -1.1437 & 2.0522 \end{pmatrix}$$

$$A = \{1, 2\} \implies \bar{A} = \{3\} \implies I_{\bar{A}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore, normalisation constant (or partition function) is:

$$\sum_{\emptyset \subseteq Y \subseteq \mathcal{Y}} \det(L_Y) = \sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I_{\bar{\emptyset}}) = \det(L + I)$$

Conversion to Marginal distribution (1)

$$\Pr_L(\mathbf{Y} = Y) \propto \det(L_Y) \implies \Pr_L(\mathbf{Y} = Y) = \frac{\det(L_Y)}{\det(L + I)}$$

An L-ensemble is a DPP, and its marginal kernel is:

$$K = L(L + I)^{-1} = I - (L + I)^{-1}$$

$L(L + I)^{-1} = I - (L + I)^{-1}$ is **true** for any L where $(L + I)^{-1}$ exist, think scalar case:

$$1 - \frac{1}{x + 1} = \frac{x + 1 - 1}{x + 1} = \frac{x}{x + 1}$$

Then,

$$\begin{aligned}
\Pr_L(A \subseteq \mathbf{Y}) &= \frac{\sum_{A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y)}{\sum_{Y \subseteq \mathcal{Y}} \det(L_Y)} \\
&= \frac{\det(L + I_{\bar{A}})}{\det(L + I)} \\
&= \det\left((L + I_{\bar{A}})(L + I)^{-1}\right)
\end{aligned}$$

Since, $\det(A^{-1}) = \frac{1}{\det(A)}$ $\quad$ $\det(AB) = \det(A)\det(B)$

Conversion to Marginal distribution (2)

$$
\begin{aligned}
\Pr_L(A \subseteq \mathbf{Y}) &= \det\left((L + I_{\bar{A}})(L + I)^{-1}\right) \\
&= \det\left(\underbrace{L(L + I)^{-1}}_{I-(L+I)^{-1}} + I_{\bar{A}}(L + I)^{-1}\right) \\
&= \det\left(I - (L + I)^{-1} + I_{\bar{A}}(L + I)^{-1}\right) \\
&= \det\left(I - (I - I_{\bar{A}})(L + I)^{-1}\right) \\
&= \det\left(I - I_A(L + I)^{-1}\right) \\
&= \det\left(\underbrace{I_A + I_{\bar{A}}}_{I} - I_A(L + I)^{-1}\right) \\
&= \det\left(I_{\bar{A}} + \underline{I_A - I_A(L + I)^{-1}}\right) \\
&= \det\left(I_{\bar{A}} + I_A\left(\underbrace{I - (L + I)^{-1}}_{K}\right)\right)
\end{aligned}
$$

Conversion to Marginal distribution (3)

$$\Pr_L(A \subseteq \mathbf{Y}) = \det\left(I_{\bar{A}} + I_A\left(\underbrace{I - (L+I)^{-1}}_{K}\right)\right)$$

- left multiplication by I_A **zeros out rows** of a matrix except those corresponding to A, $\implies$

$$K = \begin{pmatrix} K_{\bar{A}} & K_{\bar{A}A} \\ K_{A\bar{A}} & K_A \end{pmatrix} \implies I_A(K) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{|A|\times|A|} \end{pmatrix} \begin{pmatrix} K_{\bar{A}} & K_{\bar{A}A} \\ K_{A\bar{A}} & K_A \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ K_{A\bar{A}} & K_A \end{pmatrix}$$

- Re-organise:

$$\begin{aligned} \Pr_L(A \subseteq \mathbf{Y}) &= \det(I_{\bar{A}} + I_A K) \\ &= \begin{vmatrix} I_{|\bar{A}|\times|\bar{A}|} & \mathbf{0} \\ K_{A\bar{A}} & K_A \end{vmatrix} \\ &= \det(K_A) \end{aligned}$$

- $K = L(L+I)^{-1} = I - (L+I)^{-1}$ is the conversion formula!

Eigen-value conversion

$$K = L(L+I)^{-1} = I - (L+I)^{-1}$$

- **Properties**

$$\begin{aligned} \lambda_n \in \text{eig}(A) &\implies \lambda_n + 1 \in \text{eig}(A+I) \\ &\implies (\lambda_n)^{-1} \in \text{eig}(A^{-1}) \end{aligned}$$

- **Apply** it to $K = I - (L+I)^{-1}$:

$$\begin{aligned} (\lambda_n + 1) \in \text{eig}(L+I) &\implies \frac{1}{\lambda_n + 1} \in \text{eig}\left((L+I)^{-1}\right) \\ &\implies 1 - \frac{1}{\lambda_n + 1} \in \text{eig}\left(I - (L+I)^{-1}\right) \end{aligned}$$

$$1 - \frac{1}{\lambda_n + 1} = \frac{\lambda_n + 1 - 1}{\lambda_n + 1} = \frac{\lambda_n}{\lambda_n + 1}$$

- Therefore,

$$L = \sum_{n=1}^{N} \lambda_n v_n v_n^\top \implies K = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} v_n v_n^\top$$

Conversions from K to L

$$K = L(L+I)^{-1} = I - (L+I)^{-1}$$

$$\begin{aligned} K = I - (L+I)^{-1} &\implies I - K = (L+I)^{-1} \\ &\implies (L+I)(I-K) = I \\ &\implies L + I - LK - K = I \\ &\implies L(I-K) = K \\ &\implies L = K(I-K)^{-1} \end{aligned}$$

Many interesting things about DPP

- although it's a discrete distribution, but you can just sample it like softmax output, why?
- there is so much to study about this family of model and many interesting research, including neural networks, e.g,. for mini-batch diversification

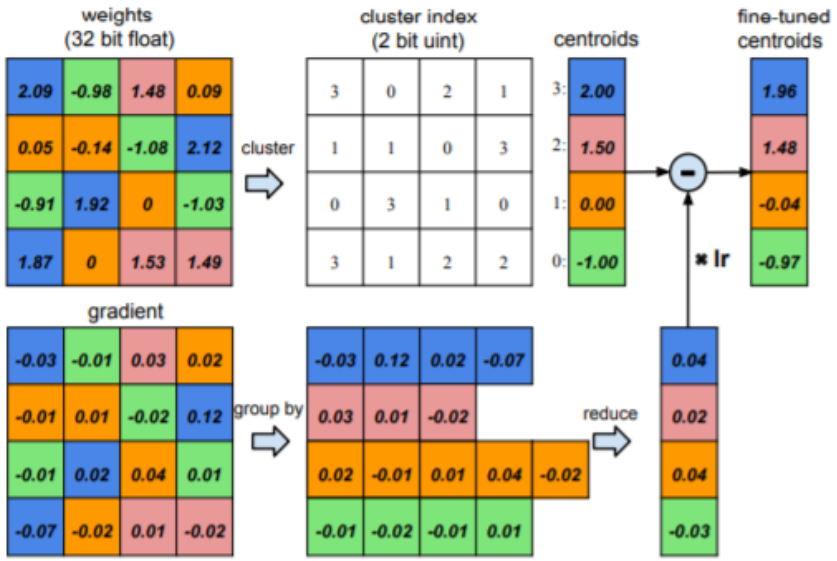
  "Zhang, Kjellstrm, Mandt, Determinantal Point Processes for Mini-Batch Diversification"
- as well as a lot of applications such as diverse subset of news collection portal
- today, we talk about an example of it it used in neural network compression

  "*Mariet et., al, Diversity Networks: Neural Network Compression Using Determinantal Point Processes*"

firstly, what is Neural Networks compression?

- very important topic, think about how we can put all VGG parameters in a mobile device!
- many works are started on neural network compression
- one seminal research:

  "*Han et. al, (2015) deep compression: Compression Deep Neural Networks with Pruning, Trained Quantization and Huffman coding*"



- examples as such performs compression on parameters
- question is, can we use DPP to remove redundantly-performing neurons?

Apply DPP to Neural Networks compression

- let $\tau = \{I_1, \ldots, I_J\}$ be the total number of J training data
- a_{il} is activation of neuron i on image j, then
- $\mathbf{v}_i$ contains a vector of all activation $[a_{i,1}, a_{i,1}, \ldots, a_{i,J}]^\top$

A possible solution is to use DPP

- it is logical to retain a set of most "diversely performed" neurons
- a particular neural layer has n_l neurons, each has some v_i responses to dataset,
- which we have a data matrix:

$$\mathbf{V} = \begin{bmatrix} | & \vdots & | \\ \mathbf{v}_1 & \vdots & \mathbf{v}_{n_l} \\ | & \vdots & | \end{bmatrix}$$

- to specify the L ensemble, we defined the following:

$$\mathbf{L}'_{ij} = \exp(-\beta \|\mathbf{v}_i - \mathbf{v}_j\|^2) \quad 1 \leq i, j \leq n_l$$

- then we can sample the most diverse subset of neurons using DPP

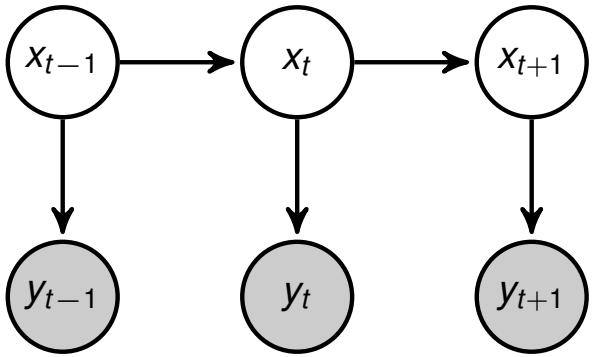
LSTM to help Kalman Filter

Kalman Filter

- why we need filtering?
- for example
- the objective is to estimate the probability of the current state x_t, given all the measurements so far: $y_1, y_2, \ldots, y_t$
- **insert some pictures of application of Kalman Filter**

State Space Models

- graphical model for state space model:



  using **markov property** of probabilistic graphical model:

$$p(x_t|x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1}) = p(x_t|x_{t-1})$$
$$p(y_t|x_1,\ldots,x_{t-1},x_t,y_1,\ldots,y_{t-1}) = p(y_t|x_t)$$

- looks familiar to Recurrent Neural Networks?

What do we want to compute?

$$\textbf{Prediction}: \quad p(x_t|\mathbf{y}_{1:t-1}) = \int_{x_{t-1}} p(x_t|x_{t-1})p(x_{t-1}|\mathbf{y}_{1:t-1})$$

$$\textbf{Update}: \quad p(x_t|\mathbf{y}_{1:t}) = \frac{p(y_t|x_t)p(x_t|\mathbf{y}_{1:t-1})}{\int_{s_t} p(y_t|s_t)p(ds_t|\mathbf{y}_{1:t-1})}$$

This is because:

$$\begin{aligned} p(x_t|\mathbf{y}_{1:t}) &\propto p(x_t, \mathbf{y}_{1:t}) \\ &\propto p(y_t|x_t)p(x_t|\mathbf{y}_{1:t-1}) \\ &= \frac{p(y_t|x_t)p(x_t|\mathbf{y}_{1:t-1})}{\int_{s_t}(y_t|s_t)p(ds_t|\mathbf{y}_{1:t-1})} \end{aligned}$$

Linear Gaussian Dynamic Model

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B + w_t \qquad w_t \sim \mathcal{N}(0, Q_t)$$
$$\implies \textbf{Transition probability:} \qquad p(\mathbf{x}_t|\mathbf{x}_{t-1}) \sim \mathcal{N}(A\mathbf{x}_{t-1} + B, Q_t)$$

$$y_t = H\mathbf{x}_t + v_t \qquad v_t \sim \mathcal{N}(0, R_t)$$
$$\implies \textbf{Measurement probability:} \qquad p(y_t|\mathbf{x}_t) \sim \mathcal{N}(H\mathbf{x}_t, R_t)$$

- Kalman Filter can be used to in this Gaussian, Linear case.
- In general, there are many other Dyanmic models which are non-Gaussian, non-Linear. They can NOT be solved using Kalman Filter.

Kalman Filter - Prediction

- following marginal distribution of linear Gaussian model, given:
- $p(X_1) \sim \mathcal{N}(x_1|\mu, \Sigma)$
- $p(X_2|x_1) \sim \mathcal{N}(x_2|Ax_1 + B, Q)$
- then, its **marginal** is:

$$p(X_2) = \int_{x_1} p(X_2|x_1)p(x_1) \sim \mathcal{N}\left(x_2|A\mu + B, A\Sigma A^T + Q\right) \mathrm{d}x_1$$

Prediction :
$$\begin{aligned} p(x_t|\mathbf{y}_{1:t-1}) \sim \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t) &= \int_{x_{t-1}} p(x_t|x_{t-1})p(dx_{t-1}|\mathbf{y}_{1:t-1}) \\ &= \int_{x_{t-1}} \mathcal{N}(x_t|Ax_{t-1} + B, Q_t)\mathcal{N}(x_{t-1}|\hat{\mu}_{t-1}, \hat{\Sigma}_{t-1}) \\ &= \mathcal{N}\left(x_t|A\hat{\mu}_{t-1} + B, A\hat{\Sigma}_{t-1}A^T + Q_t\right) \end{aligned}$$

Question How about **update**? Let's be "cool", and try an alternative method using Moment representation.

Kalman Filter Update: $p(x_t|y_1, \dots y_t) = \mathcal{N}(\hat{\mu}_t, \hat{\Sigma}_t)$

$$p(x_t|y_1, \dots y_t) \sim \mathcal{N}(\hat{\mu}_t, \hat{\Sigma}_t)$$
$$\propto p(y_t|x_t)p(x_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(y_t|Hx_t, R_t)\mathcal{N}(x_t|\bar{\mu}_t, \bar{\Sigma}_t)$$

- given **marginal** $p(\mathbf{x}_t) = \mathcal{N}(\mu_x, \Sigma_{xx})$ $\quad$ $p(\mathbf{y}_t) = \mathcal{N}(\mu_y, \Sigma_{yy})$
- and **joint** density, $\mathbf{x}_t \equiv x_t|y_1, \dots, y_{t-1}$ and $\mathbf{y}_t \equiv y_t|y_1, \dots, y_{t-1}$

$$\begin{bmatrix}\mathbf{x}_t \\ \mathbf{y}_t\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx}, \Sigma_{xy} \\ \Sigma_{xy}, \Sigma_{yy}\end{bmatrix}\right)$$

- then, conditional density is:

$$p(\mathbf{x}_t|\mathbf{y}_t) \sim \mathcal{N}\left(\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$

- In filtering:

$$p(\mathbf{x}_t) \equiv p(x_t|y_1, \dots, y_{t-1}) = \mathcal{N}(x_t|\bar{\mu}_t, \bar{\Sigma}_t)$$
$$p(\mathbf{y}_t) \equiv p(y_t|y_1, \dots y_{t-1})$$

- We are after conditional $p(\mathbf{x}_t|\mathbf{y}_t, y_1, \dots, y_{t-1})$

Kalman Filter Update

- in order to compute:

$$p(\mathbf{x}_t|\mathbf{y}_t) \sim \mathcal{N}\left(\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$

- we need to know: $\left(\mu_x, \mu_y, \Sigma_{xx}, \Sigma_{yy}, \Sigma_{xy}, \Sigma_{yx}\right)$
- introduce a zero-mean variable: $\triangle x_{t-1}$:

$$\triangle x_{t-1} \equiv x_{t-1} - \mathbb{E}[x_{t-1}] \sim \mathcal{N}(0, \hat{\Sigma}_{t-1}) \implies x_{t-1} = \triangle x_{t-1} + \mathbb{E}[x_{t-1}]$$

- attempt to write both $\triangle x_t$ and $\triangle y_t$ in terms of $\triangle x_{t-1}$:

$$\begin{aligned}
\mathbf{x}_t = A\mathbf{x}_{t-1} + w_t \qquad w_t \sim \mathcal{N}(0, Q_t) \implies \qquad x_t &= A(\triangle x_{t-1} + \mathbb{E}[x_{t-1}]) + w_t \\
&= A\mathbb{E}x_{t-1} + \underbrace{A\triangle x_{t-1} + w_t}_{\triangle x_t}
\end{aligned}$$

$$\begin{aligned}
y_t = H\mathbf{x}_t + v_t \qquad v_t \sim \mathcal{N}(0, R_t) \implies \qquad y_t &= Hx_t + v_t \\
&= H(A\mathbb{E}x_{t-1} + A\triangle x_{t-1} + w_t) + v_t \\
&= HA\mathbb{E}x_{t-1} + \underbrace{HA\triangle x_{t-1} + Hw_t + v_t}_{\triangle y_t}
\end{aligned}$$

The Independence assumptions:

- $\mathbb{COV}(x_{t-1}, w_t) = 0 \qquad \mathbb{COV}(x_{t-1}, v_t) = 0 \qquad \mathbb{COV}(w_t, v_t) = 0$

Moment Representation

- we need to know: $\left(\mu_x, \mu_y, \Sigma_{xx}, \Sigma_{yy}, \Sigma_{xy}, \Sigma_{yx}\right)$

$$
\begin{aligned}
\mu_x \equiv \quad & \mathbb{E}[x_t|y_{1:t-1}] = A\mathbb{E}[x_{t-1}] = A\hat{\mu} \\
\mu_y \equiv \quad & \mathbb{E}[y_t|y_{1:t-1}] = HA\mathbb{E}[x_{t-1}] = HA\hat{\mu} \\
\Sigma_{xx} \equiv \quad & \mathbb{E}[\triangle x_t(\triangle x_t)^T|y_{1:t-1}] = \mathbb{E}[(A\triangle x_{t-1} + w_t)(A\triangle x_{t-1} + w_t)^T] \\
& = A\hat{\Sigma}_{t-1}A^T + Q_t = \bar{\Sigma}_t \\
\Sigma_{yx} \equiv \quad & \mathbb{E}[\triangle y_t(\triangle x_t)^T|y_{1:t-1}] = \mathbb{E}\left[(HA\triangle x_{t-1} + Hw_t + v_t)(A\triangle x_{t-1} + w_t)^T\right] \\
& = H\left(A\hat{\Sigma}_{t-1}A^T + Q_t\right) = H\bar{\Sigma}_t \\
\implies \Sigma_{yx} \equiv \quad & \mathbb{E}[\triangle x_t(\triangle y_t)^\top|y_{1:t-1}] \\
& = \left(A\hat{\Sigma}_{t-1}A^T + Q_t\right)^\top H^\top = \bar{\Sigma}_t H^T \\
\Sigma_{yy} \equiv \quad & \mathbb{E}[\triangle y_t(\triangle y_t)^T|y_{1:t-1}] = \mathbb{E}\left[(HA\triangle x_{t-1} + Hw_t + v_t)(HA\triangle x_{t-1} + Hw_t + v_t)^T\right] \\
& = H\left(A\hat{\Sigma}_{t-1}A^T + Q_t\right)H^T + R_t \\
& = H(\bar{\Sigma}_t)H^T + R_t
\end{aligned}
$$

Kalman Filter Update: $p(x_t|y_1, \dots y_t) = \mathcal{N}(\hat{\mu}_t, \hat{\Sigma}_t)$

- finally we put all elements in:
- **mean:** $\hat{\mu}_t = \mathbb{E}[x_t|y_{1:t}]$:

$$
\begin{aligned}
&\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y) \\
&\equiv \mathbb{E}[\mathbf{x}_t] + \mathbb{E}[\triangle \mathbf{x}_t(\triangle \mathbf{y}_t)^T]\mathbb{E}[\triangle y_t(\triangle \mathbf{y}_t)^T]^{-1}(\mathbf{y}_t - \mathbb{E}[y_t]) \\
&= A\hat{\mu}_{t-1} + \underbrace{\bar{\Sigma}_t^T H(H\bar{\Sigma}_t H^T + R_t)^{-1}}_{K}(\mathbf{y}_t - HA\hat{\mu}_{t-1}) \\
&= \bar{\mu}_{t-1} + K(\mathbf{y}_t - HA\hat{\mu}_{t-1})
\end{aligned}
$$

co-variance: $\hat{\Sigma}_t = \mathbb{COV}[x_t|y_{1:t}]$:

$$
\begin{aligned}
&\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} \\
&\equiv \mathbb{E}[\triangle x_t(\triangle x_t)^T] - \mathbb{E}[\triangle x_t(\triangle y_t)^T]\mathbb{E}[\triangle y_t(\triangle y_t)^T]^{-1}\mathbb{E}[\triangle y_t(\triangle x_t)^T] \\
&= \bar{\Sigma}_t - \underbrace{\bar{\Sigma}_t H^T(H(\bar{\Sigma}_t)H^T + R_t)^{-1}}_{K} H\bar{\Sigma}_t \\
&= (I - KH)\bar{\Sigma}_t
\end{aligned}
$$

Kalman Filter and Neural Networks

- Kalman filters require a motion model and measurement model to be specified at priory
- it's hard!
- can be crude approximation of reality
- this is where LSTM can help out!
- "*Coskun., Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization*"

LSTM

- you see it everywhere, so I don't talk about it in detail:
- a compact form of representation:

$$\begin{bmatrix} i \\ f \\ o \\ \tilde{C} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + \mathbf{b} \right) \qquad C_t = f_t \odot C_{t-1} + i \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

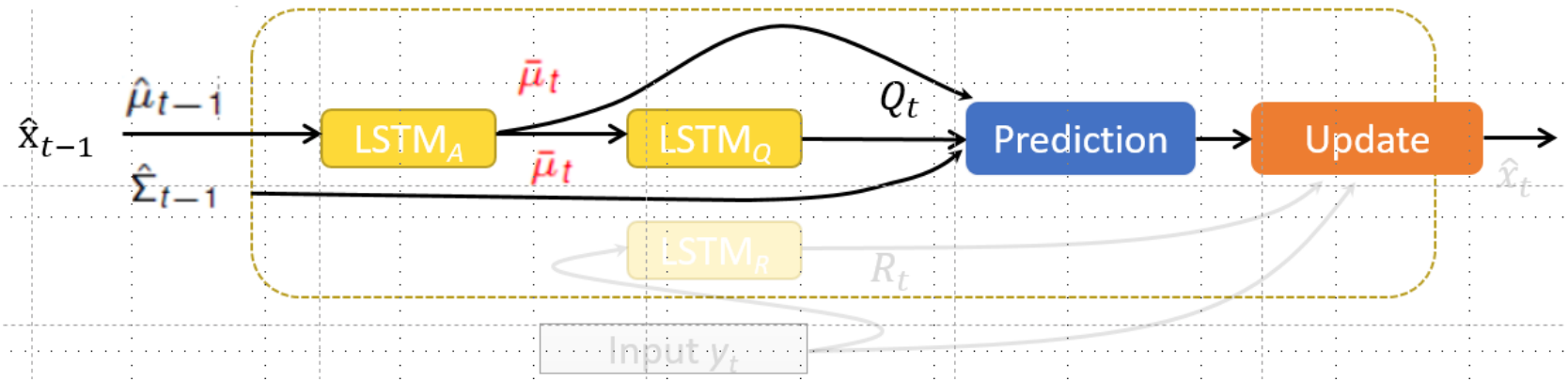
- in LSTMs, cell state C_t. The derivative of consecutive states is of the form:

$$\begin{aligned} C_t &= f_t(C_{t-1}) \odot C_{t-1} + i_t(C_{t-1}) \odot \tilde{C}_t(C_{t-1}) \\ &= f_t \odot C_{t-1} + i_t \odot \tanh(W_C[h_{t-1}, x_t] + b_C) \\ &= f_t(C_{t-1})C_{t-1} + \underbrace{i_t(h_{t-1}(C_{t-1}))\tanh(W_C[o_{t-1}(h_{t-1}(C_{t-1})) \odot \tanh(C_{t-1}), x_t] + b_C)}_{\xi(C_{t-1})} \end{aligned}$$

$$\frac{\partial C_t}{\partial C_{t-1}} = \underbrace{f_t}_{\text{gradient highway}} + \underbrace{\frac{\partial f_t}{\partial C_{t-1}} C_{t-1} + \frac{\partial \xi(C_{t-1})}{\partial C_{t-1}} C_{t-1}}_{\text{contains exponentially fast decay function}}$$

- of course, f_t may still close to zero
- **trick is to** initialize bias to be a large positive number, e.g., $f_t = \sigma(W_f[h_{t-1}, x_t] + \text{large positive number})$ so to make f_t closer to 1 initially

LSTM Kalman Filters: Prediction



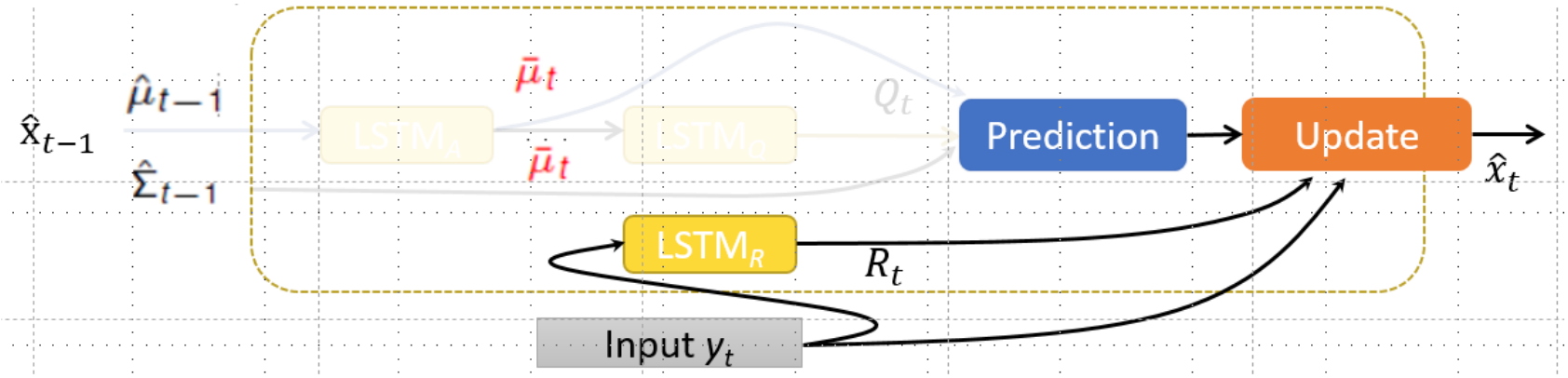
- the state-space model is changed into:

$$\begin{aligned} \mathbf{x}_t = A\mathbf{x}_{t-1} + w_t \quad w_t \sim \mathcal{N}(0, Q_t) \quad &\rightarrow \quad \mathbf{x}_t = \text{lstm}_A(\mathbf{x}_{t-1}) + w_t \quad w_t \sim \mathcal{N}(0, Q_t) \\ \mathbf{y}_t = H\mathbf{x}_t + v_t \quad v_t \sim \mathcal{N}(0, R_t) \quad &\rightarrow \quad \text{unchanged} \end{aligned}$$

- **prediction**

$$\begin{aligned} \bar{\mu}_t = A\hat{\mu}_{t-1} \quad &\rightarrow \quad \bar{\mu}_t = \text{lstm}_A(\hat{\mu}_{t-1}) \\ \bar{\Sigma}_t = A\hat{\Sigma}_{t-1}A^T + Q_t \quad &\rightarrow \quad \bar{\Sigma}_t = \mathcal{A}'\hat{\Sigma}_{t-1}\mathcal{A}'^{\top} + \text{lstm}_Q(\bar{\mu}_t), \text{ where } \mathcal{A}' = \nabla_{\mathbf{x}_{t-1}} \text{lstm}_A(\hat{\mu}_{t-1}) \end{aligned}$$

LSTM Kalman Filters: Update



- the state-space model is changed into:

$$\begin{aligned} \mathbf{x}_t = A\mathbf{x}_{t-1} + w_t \quad & w_t \sim \mathcal{N}(0, Q_t) \quad \rightarrow \quad \mathbf{x}_t = \text{lstm}_A(\mathbf{x}_{t-1}) + w_t \quad w_t \sim \mathcal{N}(0, Q_t) \\ \mathbf{y}_t = H\mathbf{x}_t + v_t \quad & v_t \sim \mathcal{N}(0, R_t) \quad \rightarrow \quad \text{unchanged} \end{aligned}$$

- **update**:

$$\begin{aligned} K_t = \bar{\Sigma}_t H^T (H(\bar{\Sigma}_t) H^T + R_t)^{-1} \quad & \rightarrow \quad K_t = \bar{\Sigma}_t H^T (H(\bar{\Sigma}_t) H^T + \text{lstm}_R(\mathbf{y}_t))^{-1} \\ \hat{\mu}_t = \bar{\mu}_t + K(\mathbf{y}_t - HA\hat{\mu}_{t-1}) \quad & \rightarrow \quad \text{unchanged} \\ \hat{\Sigma}_t = (I - KH)\bar{\Sigma}_t \quad & \rightarrow \quad \text{unchanged} \end{aligned}$$

Binary Classifier to help probability model estimation

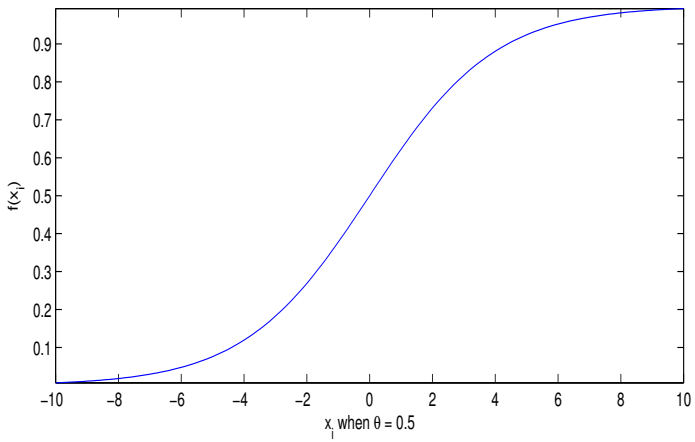
probability and classification

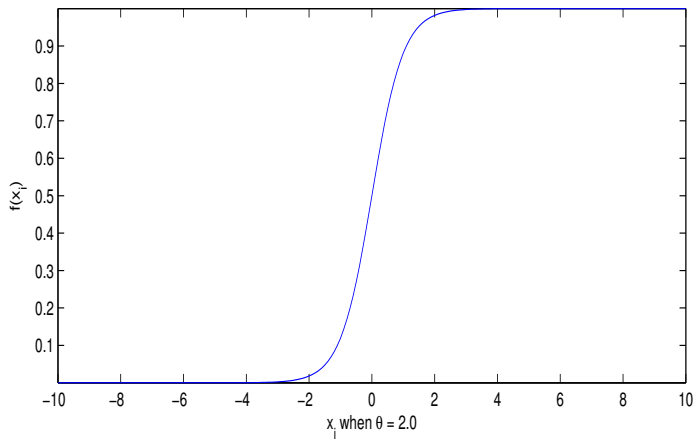
- firstly, probability models and classification are closely related:

$$\arg\max_{\theta} \left(p_\theta(\mathbf{Y}) \right) \implies \arg\min_{\theta} \left(-\log p_\theta(\mathbf{Y}) \right)$$

- in following example, let's show **classification models** incorporating our favorite sigmoid function:

$$\sigma(\mathbf{x}_i^\top \theta) = \frac{1}{1 + \exp(-\mathbf{x}_i^T \boldsymbol{\theta})}$$





Example: Bernoulli & Logistic regression

- Bernoulli distribution using Sigmoid function

$$p_{\theta}(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{n} \left[\frac{1}{1+\exp(-\mathbf{x}_i^T \theta)} \right]^{y_i} \left[1 - \frac{1}{1+\exp(-\mathbf{x}_i^T \theta)} \right]^{1-y_i}$$

- Logistic regression

$$\begin{aligned} \mathcal{C}(\theta) &= -\log[p_{\theta}(\mathbf{Y}|\mathbf{X})] \\ &= -\left(\sum_{i=1}^{n} y_i \log \left[\frac{1}{1+\exp(-\mathbf{x}_i^T \theta)} \right] + (1-y_i) \log \left[1 - \frac{1}{1+\exp(-\mathbf{x}_i^T \theta)} \right] \right) \end{aligned}$$

Example: Multinomial Distribution & Cross Entropy Loss

- Multinomial Distribution with softmax

$$p_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{n}\prod_{k=1}^{K}\left[\left(\frac{\exp(\mathbf{x}_i^T\boldsymbol{\theta}_k)}{\sum_{l=1}^{K}\exp(\mathbf{x}_i^T\boldsymbol{\theta}_l)}\right)\right]^{y_{i,k}}$$

- cross entropy loss with Softmax

$$\mathcal{C}(\boldsymbol{\theta}) = -\log[p_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{X})] = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\left[\log\left(\frac{\exp(\mathbf{x}_i^T\boldsymbol{\theta}_k)}{\sum_{l=1}^{K}\exp(\mathbf{x}_i^T\boldsymbol{\theta}_l)}\right)\right]$$

Example: Gaussian Distribution & Sum of Square Loss

- this time, let's go from $\mathcal{C}(\boldsymbol{\theta}) \rightarrow p_{\theta}(\mathbf{Y})$
- Sum of Square Loss

$$\mathcal{C}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\hat{y}_k(\boldsymbol{\theta}) - y_k\right)^2$$

- Gaussian distribution

$$p_{\theta}(\mathbf{Y}|\mathbf{X}) \propto \exp\left[-\mathcal{C}(\boldsymbol{\theta})\right] = \exp\left[-\sum_{k=1}^{K} \left(\hat{y}_k(\boldsymbol{\theta}) - y_k\right)^2\right]$$

- **question**: what if we use *Square* loss instead of *Cross Entropy* loss in Softmax, where:

$$\hat{y}_k(\boldsymbol{\theta}) = \frac{\exp(\mathbf{x}_i^T \boldsymbol{\theta}_k)}{\sum_{l=1}^{K} \exp(\mathbf{x}_i^T \boldsymbol{\theta}_l)}$$

Think about Classification's best friend, "Softmax" again!

- for example, in word embedding, we want to align a target word $\mathbf{u}_w$ with center word $\mathbf{v}_c$:
- for simplicity, for the rest of the article, we let $\mathbf{w} \equiv \mathbf{u}_w$ and $\mathbf{c} \equiv \mathbf{v}_c$

$$\Pr_\theta(\mathbf{w}|\mathbf{c}) = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{\sum_{w' \in \mathcal{V}} u_\theta(\mathbf{w}'|\mathbf{c})} = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{Z_c} \equiv \frac{\exp(\mathbf{w}^\top \mathbf{c})}{\sum_{\mathbf{w}' \in \mathcal{V}} \exp(\mathbf{w}'^\top \mathbf{c})}$$

- the denominator, i.e., the $\sum_{\mathbf{w}' \in \mathcal{V}} u(\mathbf{w}'|\mathbf{c})$ can be too computational

Turn the problem around!

- **data distribution**: we sample $\mathbf{w} \sim \bar{p}(\mathbf{w}|\mathbf{c})$ from its empirical (data) distribution, and give a label $\mathcal{Y} = 1$
- **noise distribution**: we can sample k $\bar{\mathbf{w}} \sim q(\mathbf{w})$, and give them labels $\mathcal{Y} = 0$ **importantly**, condition for $q(.)$ is: it does **not** assign zero probability to any data.
- Can we build a binary classifier to **classify** its label, i.e., which distribution has generated it?

Noise Contrastive Estimation (NCE)

- **training data generation**: (w, c, y)
  1. sample $(\mathbf{w}, \mathbf{c})$: using $\mathbf{c} \sim \tilde{p}(\mathbf{c}), \mathbf{w} \sim \tilde{p}(\mathbf{w}|\mathbf{c})$ and label them as $\mathcal{Y} = 1$
  2. k "noise" samples from $q(.)$, and label them as $\mathcal{Y} = 0$
- can we instead, try to maximize the joint posterior Bernoulli distribution:

$$\Pr_\theta(\mathcal{Y}|\mathbf{W}, \mathbf{c}) = \prod_{i=1}^{k+1} \left(\Pr(\mathcal{Y}_i|\mathbf{w}_i, \mathbf{c}) \right)^{y_i} \left(1 - \Pr(\mathcal{Y}_i|\mathbf{w}_i, \mathbf{c}) \right)^{1-y_i}$$

- or minimize the corresponding Logistic regression:

$$\begin{aligned} \mathcal{C} &= -\log[\Pr_\theta(\mathcal{Y}|\mathbf{W}, \mathbf{c})] \\ &= -\sum_{i=1}^{k+1} y_i \log\left[\Pr_\theta(\mathcal{Y}_i|\mathbf{w}_i, \mathbf{c})\right] + (1 - y_i) \log\left[1 - \Pr_\theta(\mathcal{Y}_i|\mathbf{w}_i, \mathbf{c})\right] \end{aligned}$$

Noise Contrastive Estimation (NCE)

- we assume there are k negative samples per positive sample, so the prior density is:

$$P(\mathcal{Y}=y)=\begin{cases}\frac{1}{k+1} & y=1\\ \frac{k}{k+1} & y=0\end{cases}$$

- then the posterior of $P(\mathcal{Y}|\mathbf{c},\mathbf{w})$:

$$\begin{aligned}P(\mathcal{Y}=1|\mathbf{c},\mathbf{w})&=\frac{\Pr(\mathcal{Y}=1,\mathbf{w}|\mathbf{c})}{\Pr(\mathbf{w}|\mathbf{c})}=\frac{\Pr(\mathbf{w}|\mathcal{Y}=1,\mathbf{c})P(\mathcal{Y}=1)}{\sum_{y\in\{0,1\}}p(w|\mathcal{Y}=y,\mathbf{c})P(\mathcal{Y}=y)}\\&=\frac{\tilde{p}(\mathbf{w})\times\frac{1}{1+k}}{\tilde{P}(\mathbf{w}|\mathbf{c})\times\frac{1}{k+1}+q(\mathbf{w})\times\frac{k}{1+k}}\\&=\frac{\tilde{P}(\mathbf{w}|\mathbf{c})}{\tilde{P}(\mathbf{w}|c)+kq(\mathbf{w})}\end{aligned}$$

$$\begin{aligned}\Pr(\mathcal{Y}=0|\mathbf{c},\mathbf{w})&=1-\Pr(\mathcal{Y}=1|\mathbf{c},\mathbf{w})\\&=1-\frac{\tilde{P}(\mathbf{w}|\mathbf{c})}{\tilde{P}(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})}\\&=\frac{kq(\mathbf{w})}{\tilde{P}(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})}\end{aligned}$$

Apply NCE to NLP problem

- in summary:

$$\Pr(\mathcal{Y} = y|\mathbf{c}, \mathbf{w}) = \begin{cases} \frac{\tilde{P}(\mathbf{w}|\mathbf{c})}{\tilde{P}(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})} & y = 1 \\ \frac{kq(\mathbf{w})}{\tilde{P}(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})} & y = 0 \end{cases}$$

- it can be replaced by un-normalized function:

$$\Pr(\mathcal{Y} = y|\mathbf{c}, \mathbf{w}) = \begin{cases} \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})} & y = 1 \\ \frac{kq(\mathbf{w})}{u_\theta(\mathbf{w}|\mathbf{c})+kq(\mathbf{w})} & y = 0 \end{cases}$$

- formal proof can be found "*Gutmann, 2012, Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics*"
- let's see an intuition through **softmax**

Intuition through Softmax

- think about Softmax in word embedding:

$$\Pr_\theta(\mathbf{w}|\mathbf{c}) = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{\sum_{w' \in \mathcal{V}} u_\theta(\mathbf{w}'|\mathbf{c})} = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{Z_c} \equiv \frac{\exp(\mathbf{w}^\top \mathbf{c})}{\sum_{\mathbf{w}' \in \mathcal{V}} \exp(\mathbf{w}'^\top \mathbf{c})}$$

- say $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$ are target words having high frequencies given $\mathbf{c}$
- $\{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_n\}$ are words having low frequency given $\mathbf{c}$
- say we pick $\mathbf{w}_i \in \{\mathbf{w}_1, \ldots w_k\}$ to optimize: at each round, we aim to increase $\mathbf{w}_i^\top \mathbf{c}$; at the same time, sum of rest of softmax weights: $\left\{\{\mathbf{w}_j^\top \mathbf{c}\}_{j \neq i} \cup \{\mathbf{r}_j^\top \mathbf{c}\}\right\}$ decrease
- in softmax, such decrease is guaranteed by the sum in denominator
- each $\mathbf{w}_i$ has a chance to increase $\mathbf{w}_i^\top \mathbf{c}$, but each $\mathbf{r}_i^\top \mathbf{c}$ will (hopefully) stay low

- **intuition**: in NCE, instead of using sum in the denominator, we "designed" a probability $q(.)$, such that, while letting $\mathbf{w}_i$ be a positive training sample, we also have chance to let $\mathbf{w}_{j \neq i}$ to be part of negative training sample, i.e., to reduce the value of $\mathbf{w}_j^\top \mathbf{c}$; it somewhat has a similar effect as **softmax**

NCE in a nutshell

NCE transforms:

- a problem of model estimation (computationally expensive) to:
- a problem of estimating parameters of probabilistic binary posterior classifier (computationally acceptable):
- main advantage: it allows us to fit models that are not explicitly normalized, making training time effectively independent of the vocabulary size

NCE objective function

- let $u_\theta(\mathbf{w}|\mathbf{c}) = \exp[s_\theta(\mathbf{w}|\mathbf{c})]$:

$$\Pr(\mathcal{Y}=1|\mathbf{c},\mathbf{w}) = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c}) + kq(\mathbf{w})} = \sigma\big(\triangle s_\theta(\mathbf{w}|\mathbf{c})\big)$$

$$\Pr(\mathcal{Y}=0|\mathbf{c},\mathbf{w}) = \frac{kq(\mathbf{w})}{u_\theta(\mathbf{w}|\mathbf{c}) + kq(\mathbf{w})} = 1 - \sigma\big(\triangle s_\theta(\mathbf{w}|\mathbf{c})\big)$$

where $\triangle s_\theta(\mathbf{w}|\mathbf{c}) \equiv s_\theta(\mathbf{w}|\mathbf{c}) - \log(kq(\mathbf{w}))$ let's see why

$$\begin{aligned}\sigma\big(\triangle s_\theta(\mathbf{w}|\mathbf{c})\big) &= \frac{1}{1+\exp\big[-s_\theta(\mathbf{w}|\mathbf{c}) + \log(kq(\mathbf{w}))\big]} \\ &= \frac{1}{1+\exp\big(-s_\theta(\mathbf{w}|\mathbf{c})\big) \times kq(\mathbf{w})} \\ &= \frac{\exp\big[s_\theta(\mathbf{w}|\mathbf{c})\big]}{\exp\big[s_\theta(\mathbf{w}|\mathbf{c})\big] + kq(\mathbf{w})} = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c}) + kq(\mathbf{w})}\end{aligned}$$

- therefore the objective function is:

$$\theta^* = \arg\max_\theta \sum_{(\mathbf{w},\mathbf{c})\in D} \sigma(\triangle s_\theta(\mathbf{w}|\mathbf{c})) + \sum_{(\bar{\mathbf{w}},c)\in \tilde{D}} \sigma(-\triangle s_\theta(\bar{\mathbf{w}}|\mathbf{c}))$$

NCE and Negative Sampling

- **negative sampling** is a special case of NCE
- we let $k = |\mathcal{V}|$ and $q(.)$ is uniform:

$$P(\mathcal{Y} = 1|\mathbf{c}, \mathbf{w}) = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c}) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c}) + 1}$$

$$P(\mathcal{Y} = 0|\mathbf{c}, \mathbf{w}) = \frac{|\mathcal{V}|\frac{1}{|\mathcal{V}|}}{u_\theta(\mathbf{w}|\mathbf{c}) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{1}{u_\theta(\mathbf{w}|\mathbf{c}) + 1}$$

- correspondingly, we have:

$$\triangle s_\theta(\mathbf{w}|\mathbf{c}) \equiv s_\theta(\mathbf{w}|\mathbf{c}) - \log\left(|\mathcal{V}|\frac{1}{|\mathcal{V}|}\right) = s_\theta(\mathbf{w}|\mathbf{c}) = \mathbf{w}^\top \mathbf{c}$$

- in Skip-gram:

$$\begin{aligned}\theta^* &= \arg\max_\theta \sum_{(\mathbf{w},\mathbf{c})\in D} \sigma(\mathbf{w}^\top \mathbf{c}) + \sum_{(\bar{\mathbf{w}},c)\in \tilde{D}} \sigma(-\bar{\mathbf{w}}^\top \mathbf{c}) \\ &= \arg\min_\theta \sum_{(w,c)\in D} \sigma(-\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \frac{1}{1 + \exp\left(-\bar{\mathbf{w}}^\top \mathbf{c}\right)}\end{aligned}$$

why un-normalised $u_\theta(\mathbf{w}, \mathbf{c})$ still works?

- talk a look at this again, let $u_\theta(\mathbf{w}|\mathbf{c}) = \exp[s_\theta(\mathbf{w}|\mathbf{c})]$:

$$\Pr(\mathcal{Y} = 1|\mathbf{c}, \mathbf{w}) = \frac{u_\theta(\mathbf{w}|\mathbf{c})}{u_\theta(\mathbf{w}|\mathbf{c}) + kq(\mathbf{w})} = \sigma\big(\triangle s_\theta(\mathbf{w}|\mathbf{c})\big)$$
$$\text{where } \triangle s_\theta(\mathbf{w}|\mathbf{c}) \equiv s_\theta(\mathbf{w}|\mathbf{c}) - \log(kq(\mathbf{w}))$$

- we already know:

$$= \sigma\big(\triangle s_\theta(\mathbf{w}|\mathbf{c})\big) = \frac{1}{1 + \underbrace{\exp\big(-s_\theta(\mathbf{w}|\mathbf{c})\big) \times kq(\mathbf{w})}_{G(\mathbf{w}, \theta)}}$$

- in this case,

$$G(\mathbf{w}, \theta) = \exp\big(-s_\theta(\mathbf{w}|\mathbf{c})\big) \times kq(\mathbf{w})$$
$$= \frac{kq(\mathbf{w})}{\exp(s_\theta(\mathbf{w}|\mathbf{c}))} = \frac{kq(\mathbf{w})}{u_\theta(\mathbf{w}|\mathbf{c})}$$

- or more generically:

$$G(\mathbf{w}, \theta) = \frac{m}{n} \frac{q(\mathbf{w})}{u_\theta(\mathbf{w}|\mathbf{c})}$$

what do we need to prove?

- look at $G(\mathbf{w}, \theta) = \frac{m}{n} \frac{q(\mathbf{w})}{u_\theta(\mathbf{w}|\mathbf{c})}$:
- $G(\mathbf{w}, \theta)$ is a function of θ, so this ratio changes; However, the **real trick** is if let:

$$\theta^* = \arg\max_\theta \frac{1}{n} \left(\sum_{i=1}^{n} \log \Pr(\mathcal{Y}_i = 1|\mathbf{w}_i, \theta) + \sum_{i=1}^{m} \log[\Pr(\mathcal{Y}_i = 0|\mathbf{w}_i, \theta)] \right)$$

- and we prove the following: (under large sample size n and m):

$$G(\mathbf{w}, \theta^*) \to \frac{m}{n} \frac{q(\mathbf{w})}{p(\mathbf{w})} \implies u_{\theta^*}(\mathbf{w}|\mathbf{c}) \to p(\mathbf{w}) \qquad \text{as } \theta \to \theta^*$$

so why does $G(\mathbf{w}, \theta^*) \to \frac{m}{n}\frac{q(\mathbf{w})}{\bar{p}(\mathbf{w})}$?

- let,

$$\begin{aligned}\mathcal{C}_n(\theta) &= \frac{1}{n}\left(\sum_{i=1}^{n}\log \Pr(\mathcal{Y}_i = 1|\mathbf{w}_i, \theta) + \sum_{i=1}^{m}\log[\Pr(\mathcal{Y}_i = 0|\mathbf{w}_i, \theta)]\right) \\ &= \frac{1}{n}\sum_{i=1}^{n}\log \Pr(\mathcal{Y}_i = 1|\mathbf{w}_i, \theta) + \underbrace{\frac{m}{n}}_{\nu}\frac{1}{m}\sum_{i=1}^{m}\log[\Pr(\mathcal{Y}_i = 0|\mathbf{w}_i, \theta)]\end{aligned}$$

- let $n \to \infty$ and $m \to \infty$: $\mathcal{C}_n \to \mathcal{C}$:

$$\begin{aligned}\mathcal{C} &= \mathbb{E}_{\mathbf{w}\sim p(\mathbf{w})}[\log \Pr(\mathcal{Y}_i = 1|\mathbf{w}_i, \theta)] + \nu\mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})}[\log[\Pr(\mathcal{Y}_i = 0|\mathbf{w}_i, \theta)] \\ &= \mathbb{E}_{\mathbf{w}\sim p(\mathbf{w})}\left[\log\frac{1}{1+G(\mathbf{w},\theta)}\right] + \nu\mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})}\left[\log\frac{G(\mathbf{w},\theta)}{1+G(\mathbf{w},\theta)}\right] \\ &= -\mathbb{E}_{\mathbf{w}\sim p(\mathbf{w})}\left[\log(1+G(\mathbf{w},\theta))\right] + \nu\mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})}\left[\log G(\mathbf{w},\theta) - \log(1+G(\mathbf{w},\theta))\right] \\ &= -\int \log\left(1+G(\mathbf{w},\theta)\right)p(\mathbf{w})\mathrm{d}\mathbf{w} + \nu\int\left(\log G(\mathbf{w},\theta) - \log(1+G(\mathbf{w},\theta))\right)q(\mathbf{w})\mathrm{d}\mathbf{w}\end{aligned}$$

using functional derivative

$$\mathcal{C} = -\int \log\big(1 + G(\mathbf{w}, \theta)\big) p(\mathbf{w}) \mathrm{d}\mathbf{w} + \nu \int \big(\log G(\mathbf{w}, \theta) - \log(1 + G(\mathbf{w}, \theta))\big) q(\mathbf{w}) \mathrm{d}\mathbf{w}$$

- take functional derivative:

$$\begin{aligned}
\frac{\delta \mathcal{C}(G)}{\delta G} &= -\frac{p(\mathbf{w})}{1 + G(\mathbf{w}, \theta)} + \nu q(\mathbf{w}) \left(\frac{1}{G(\mathbf{w})} - \frac{1}{1 + G(\mathbf{w})} \right) \\
&= -\frac{p(\mathbf{w})}{1 + G(\mathbf{w}, \theta)} + \frac{\nu q(\mathbf{w})}{G(\mathbf{w})(1 + G(\mathbf{w}))} = 0 \\
\implies \frac{\nu q(\mathbf{w})}{G(\mathbf{w})(1 + G(\mathbf{w}))} &= \frac{p(\mathbf{w})}{1 + G(\mathbf{w}, \theta)} \\
\implies \frac{\nu q(\mathbf{w})}{G(\mathbf{w})} &= p(\mathbf{w}) \\
\implies G(\mathbf{w}) &= \nu \frac{q(\mathbf{w})}{p(\mathbf{w})}
\end{aligned}$$

Probability density re-parameterization

Score Function Estimator

- we love to have integral in a form:

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

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

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

- we do **not** love $\int_x f(z)\nabla_\theta p(z|\theta)\mathrm{d}z$,
- in general, $\nabla_\theta p(z|\theta)$ 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:

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

- i.e, θ is the parameter of the distribution
- e.g., in **Reinforcement Learning**: let $\Pi \equiv \{s_1, a_1, \ldots, s_T, a_T\}$

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

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

Score Function Estimator

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

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

- looking at the original integral:

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

- can approximated by:

$$\frac{1}{N}\sum_{i=1}^{N} f(z^{(i)})\nabla_\theta[\log p(z^{(i)}|\theta)] \qquad z^{(i)} \sim p(z|\theta)$$

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

Re-parameterization trick

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

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

$$\int_{x\in\Omega_x} f(g(x,\theta))p(x)\mathrm{d}x = \int_{z\in\Omega_z} f(z)p_\theta(z)\mathrm{d}z$$

- 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: $z \sim \mathcal{N}(z;\mu(\theta),\sigma(\theta))$ can be rewritten as a function of a standard Gaussian variable:

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

revision on change of variable

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

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

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

- without change of limits

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

- with change of limits

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

re-parameterization trick (2)

- **main motivation** $p(x)$ is **no longer** parameterized by θ:

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

- during gradient decent, x are sampled independent of θ

example

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

$$\begin{aligned}\theta^* &= \arg\max_{\theta}[F(\theta)] \\ &= \arg\min \mathbb{E}_{z\sim\mathcal{N}(\mu(\theta),\sigma(\theta))}[z^2] \\ &= \arg\min_{\theta}\left[\int_z \underbrace{z^2}_{f(z)}\mathcal{N}\left(\underbrace{a\theta+b}_{\mu(\theta)},\underbrace{1}_{\sigma(\theta)}\right)\right]\end{aligned}$$

- we can solve it by imagine its diagram ...
- in words, it says: find mean of Gaussian, so that the "expected square of samples" from this Gaussian are minimized;
- it's obvious that you want to move μ to close to **zero** as possible
- which implies $\theta = -\frac{b}{a} \implies \mu(\theta) = 0$
- without using any tricks, the gradient is computed by:

$$\nabla_\theta F(\theta) = \int_z \underbrace{z^2}_{f(z)} \times \underbrace{\frac{2(z-\mu)}{\sigma^2}\frac{\exp^{-(z-\mu)^2/\sigma^2}}{\sqrt{2\pi}\sigma}}_{\frac{\partial\mathcal{N}(\mu,\sigma^2)}{\partial\mu}} \times \underbrace{a}_{\frac{\partial\mu}{\partial\theta}} \mathrm{d}z$$

- very hard!

solve it using **REINFORCE trick**

- let's solve it by gradient descend by **REINFORCE**:
- let $\mu(\theta) = a\theta + b$, and $\sigma(\theta) = 1$:

$$
\begin{aligned}
\int_z f(z)\nabla_\theta p(z|\theta)\mathrm{d}z &= \mathbb{E}_{z\sim p(z|\theta)}\left[f(z)\nabla_\theta[\log p(z|\theta)]\right] \\
&= \mathbb{E}_{z\sim p(z|\theta)}\left[z^2\nabla_\theta \log\left(\frac{1}{\sigma\sqrt{2\pi}}\exp^{-\frac{(z-\mu)^2}{2\sigma^2}}\right)\right] \\
&= \mathbb{E}_{z\sim p(z|\theta)}\left[z^2\nabla\mu\left[-\log(\sqrt{2\pi}\sigma) - \frac{(z-\mu)^2}{2\sigma^2}\right]\times\frac{\partial\mu(\theta)}{\theta}\right] \\
&= \mathbb{E}_{z\sim\mathcal{N}\left(z;a\theta+b,1\right)}\left[z^2(z-\mu(\theta))\times a\right] \qquad \text{let } \sigma = 1 \\
&= \mathbb{E}_{z\sim\mathcal{N}\left(z;a\theta+b,1\right)}\left[z^2 a(z-a\theta-b)\right]
\end{aligned}
$$

solve it using **re-parameterization trick:**

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

$$
\begin{aligned}
x &\sim \mathcal{N}(0, 1) \\
z &\equiv g(x, \theta) = \mu(\theta) + x\sigma(\theta)
\end{aligned}
$$

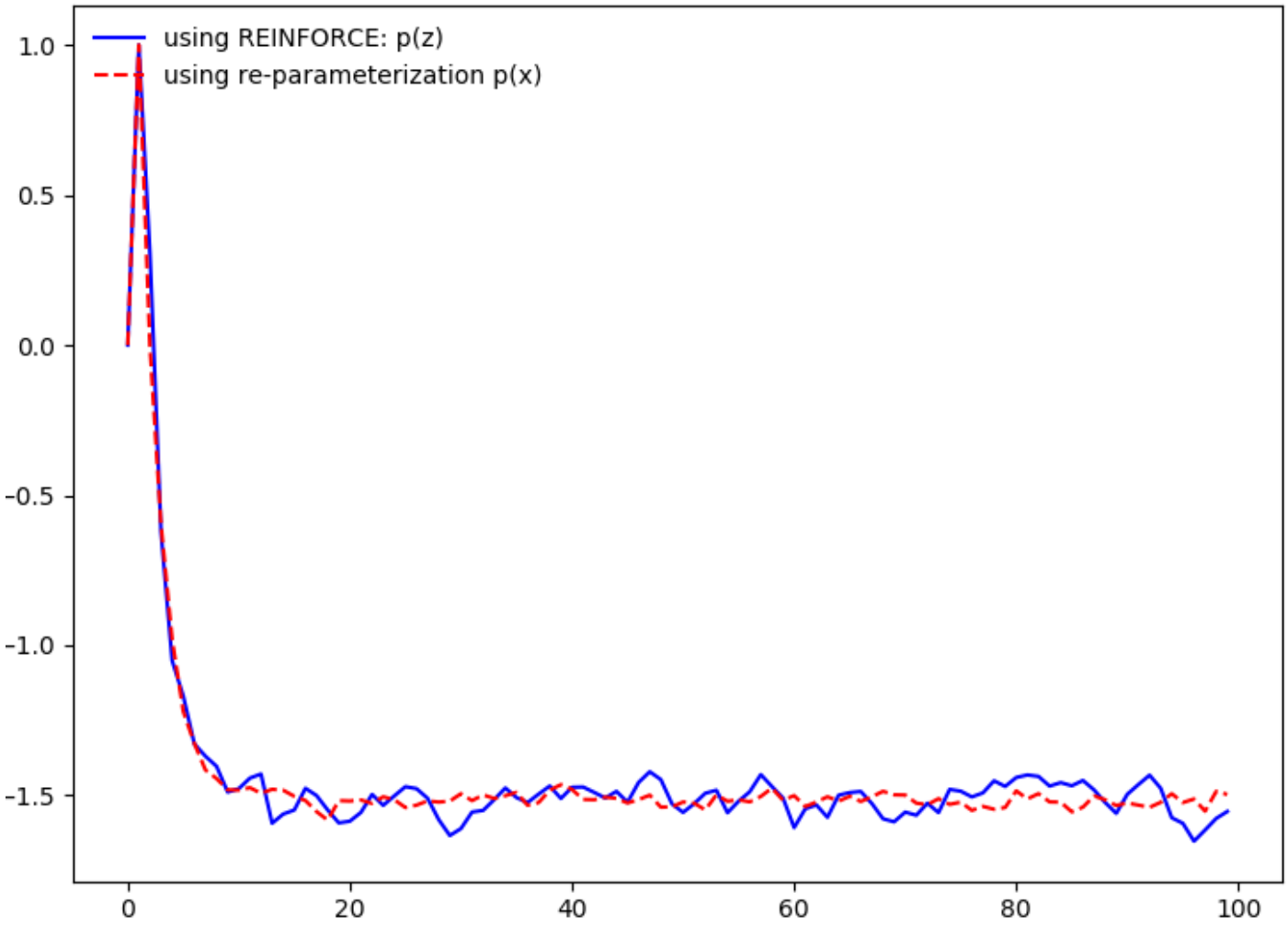
- the re-parameterised version is:

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

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

results

- compare both methods using $a = 2, b = 3$:



apply re-parameterization trick to softmax

- when we have the following

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

- can we find their corresponding:

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

Re-parameterization using Gumbel-max trick

- Gumbel-max trick also means:

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

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

- this is a form of re-paramterization:

  instead of sample $\mathcal{K} \sim \text{softmax}(\mu_1(\theta), \ldots, \mu_K(\theta))$, we i.i.d. sample $\mathcal{G}$ instead

- well, there is two problems, firstly **why is such true**?

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:

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

Gumbel-max trick and Softmax (1)

- 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:

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

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

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

Gumbel-max trick and Softmax (2)

- keep on going:

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

Gumbel-max trick and Softmax (2)

- moral of the story is, if one is to sample the largest element from **softmax**:

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

$$\implies K = \underset{i \in \{1,\ldots,L\}}{\arg\max} \{G_1, \ldots, G_L\}$$

$$\text{where } \underbrace{G_i \sim \text{gumbel}(z\,;\,\mu_i) \equiv \exp\left[-(z-\mu_i) - \exp\{-(z-\mu_i)\} \right]}$$

$$\implies K = \underset{i \in \{1,\ldots,L\}}{\arg\max} \{\mu_1 + \mathcal{G}, \ldots, \mu_L + \mathcal{G}\}$$

$$\text{where } \underbrace{\mathcal{G} \overset{\text{iid}}{\sim} \text{gumbel}(z\,;\,0) \equiv \exp\left[-(z) - \exp\{-(z)\} \right]}$$

- what is μ_i? for example,
  - $\mu_i \equiv \mathbf{x}^\top \theta_i$ in classification
  - $\mu_i \equiv \mathbf{u}_i^\top \mathbf{v}_c$ for word vectors
- some literature writes it as :

$$\equiv \underset{i \in \{1,\ldots,L\}}{\arg\max} \{\log(\mu_1) + \mathcal{G}, \ldots, \log(\mu_L) + \mathcal{G}\}$$

meaning, they let $\mu_i \equiv \exp(\mathbf{x}^\top \theta_i)$

how to sample a Gumbel?

- CDF of a Gumbel:

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

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

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

- therefore, sampling strategy:

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

Second problem with Softmax re-parameterisation

- the other 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**
- several solutions proposed, for example:
  "*Maddison, Mnih, and Teh (2017),The Concrete Distribution: a Continuous Relaxation of Discrete Random Variables*"

Relax the Softmax

- **softmax map**

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

$$\text{as} \quad \tau \to 0 \implies f_\tau(x) = \max\left(\left\{ \frac{\exp(\mu_k)}{\sum_{k=1}^K \exp(\mu_k)} \right\}_{k=1}^K \right)$$

- questions can you also think about the relationship between Gaussian Mixture Model and K-means?
- one can say $\tau = 1$ is softmax, and $\tau = 0$ is hard-max!
- then we can apply the same softmax map with added Gumbel variables:

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

Stochastic matrices and Monte Carlo Inference

Stochastic matrices

- **Right stochastic matrix** (or row stochastic matrix) is a real square matrix, with **each row** summing to 1.

$$\begin{bmatrix} K_{1\to 1} & \dots & K_{1\to n} \\ \dots & \dots & \dots \\ K_{d\to 1} & \dots & K_{d\to n} \\ \dots & \dots & \dots \\ K_{n\to 1} & \dots & K_{n\to n} \end{bmatrix}$$

- **Left stochastic matrix** (or column stochastic matrix) is a real square matrix, with **each column** summing to 1

$$\begin{bmatrix} K_{1\to 1} & \dots & K_{n\to 1} \\ \dots & \dots & \dots \\ K_{1\to d} & \dots & K_{n\to d} \\ \dots & \dots & \dots \\ K_{1\to n} & \dots & K_{n\to n} \end{bmatrix}$$

- **doubly stochastic matrices**: is a real square matrix, where both **each column** and **each row** summing to 1.

Product of two stochastic matrix is still stochastic

- Each entry in the product AB is a dot product of a row from A and a column from B.

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

- We need to prove, for a single row of product $(AB)_{i,:}$

$$\sum_{j=1}^{n} (AB)_{ij} = \sum_{j=1}^{n} \sum_{k=1}^{n} A_{ik} B_{kj} = \sum_{k=1}^{n} (A_{ik} \sum_{j=1}^{n} B_{kj})$$

- Because B is stochastic, $\sum_{j=1}^{n} B_{kj} = 1$
- Because A is stochastic, $\sum_{k=1}^{n} A_{ik} = 1$

Perron-Frobenius Theorem:

If K is a **positive**, left stochastic matrix, then:

- 1 is an eigenvalue of multiplicity one.
- 1 is the largest eigenvalue: all the other eigenvalues have absolute value smaller than 1.
- the eigenvectors corresponding to the eigenvalue 1 have either only positive entries or only negative entries.
- Note that K is a **positive** means, $K_{ij} \geq 0 \quad \forall i, j$. It's NOT **positive definite matrix**

Power Method Convergence Theorem

- Let K be a positive, left (i.e., column) stochastic $n \times n$ matrix.
- π^* its **probabilistic eigenvector** corresponding to the eigenvalue 1.

$$\begin{bmatrix} K_{1\to 1} & \dots & K_{n\to 1} \\ \dots & \dots & \dots \\ K_{1\to n} & \dots & K_{n\to n} \end{bmatrix} \begin{bmatrix} \pi_1^* \\ \dots \\ \pi_n^* \end{bmatrix} = \begin{bmatrix} \pi_1^* \\ \dots \\ \pi_n^* \end{bmatrix}$$

- Let $\pi^{(1)}$ be the column vector with all entries equal to some arbitrary stochastic vector.
- Then sequence $\{\pi^{(1)}, K\pi^{(1)}, K^2\pi^{(1)}, \dots, K^t\pi^{(1)} \dots, K^\infty\pi^{(1)}\}$ converges to the vector π^*

$$\lim_{t\to\infty} K^t = K^\infty \implies \lim_{t\to\infty} K^t\pi^{(1)} = \pi^*$$

- **Exercise** Generate some random matrix in MATLAB and to show an example of the above.
- **Exercise** observe what K^∞ looks like

Extend to continous case

- in the **discrete** case:

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & \dots & K_{n\to 1} \\ \dots & \dots & \dots & \dots \\ K_{1\to d} & K_{2\to d} & \dots & K_{n\to d} \\ \dots & \dots & \dots & \dots \\ K_{1\to n} & K_{2\to n} & \dots & K_{n\to n} \end{bmatrix} \begin{bmatrix} \pi_1^* \\ \dots \\ \pi_d^* \\ \dots \\ \pi_n^* \end{bmatrix} = \begin{bmatrix} \pi_1^* \\ \dots \\ \pi_d^* \\ \dots \\ \pi_n^* \end{bmatrix} \implies \pi_d^* = \sum_{i=1}^{n} \pi_i^* K_{i\to d}$$

- in the **continous** case, let $\pi(x)$ be the target distribution:

$$\pi(x^{(n+1)}) = \int_{x_n} \pi\left(x^{(n)}\right) K\left(x^{(n)} \to x^{(n+1)}\right)$$

- A transition kernel K contains element-wise entries: $\{K\left(x^{(n)} \to x^{(n+1)}\right)\} \quad \forall x^{(n)}, x^{(n+1)}$
- Sometimes we prefer to write $\left(x^{(n)} \text{ as } x\right)$ and $\left(x^{(n+1)} \text{ as } x^*\right)$.
- $K(x \to x^*)$ is the probability a process at state x moves to state x^* in a **one step**
- $K^n(x \to x^*)$ is the probability a process at state x moves to state x^* in **n steps**

Power Method Convergence in continuous case

- One may have first sample $x^{(1)}$ distributed from an arbitrary distribution:

$$x^{(1)} \sim \pi^{(1)}$$

- by applying K function, to obtain $x^{(2)}$ given $x^{(1)}$ with probability:

$$\pi^{(2)}(x^{(2)}) = \int_{x^{(1)}} \pi^{(1)}(x^{(1)}) K(x^{(1)} \to x^{(2)}) \mathrm{d}x^{(1)}$$

- by applying K function again, to obtain $x^{(3)}$ with probability:

$$\begin{aligned}
\pi^{(3)}(x^{(3)}) &= \int_{x^{(1)}} \int_{x^{(2)}} \pi^{(1)}(x^{(1)}) K(x^{(1)} \to x^{(2)}) K(x^{(2)} \to x^{(3)}) \mathrm{d}x^{(1)} \mathrm{d}x^{(2)} \\
&= \int_{x^{(1)}} \pi^{(1)}(x^{(1)}) \underbrace{\int_{x^{(2)}} K(x^{(1)} \to x^{(2)}) K(x^{(2)} \to x^{(3)}) \mathrm{d}x^{(2)}} \mathrm{d}x^{(1)} \\
&= \int_{x^{(1)}} \pi^{(1)}(x^{(1)}) \underbrace{K^2(x^{(1)} \to x^{(3)})} \mathrm{d}x^{(1)} \qquad \to \text{converge closer to } \pi(x^{(3)})
\end{aligned}$$

- This says,

$$\lim_{t \to \infty} \pi^{(t)}(x^{(t)}) \to \pi(x^{(t)})$$

Burn in samples

- We know,

$$\lim_{t \to \infty} \pi^{(t)}\left(x^{(t)}\right) \to \pi\left(x^{(t)}\right)$$

- But, in practice,

$$\lim_{t \to B} \pi^{(t)}\left(x^{(t)}\right) \to \pi\left(x^{(t)}\right)$$

- $\{x^{(1)}, \ldots, x^{(B)}\}$ are the **burn-in** samples, which we discard.

What is MCMC research is all about

- **equilibrium equation**:

$$\pi(x^*) = \int_x \pi(x) K\left(x \to x^*\right) \mathrm{d}x$$

- In machine learning, we always know the expression of stationary distribution $\pi(x)$,
- Our task is therefore, **find an appropriate** $K(x \to x^*)$ to generate samples in a Markov fashion.

Detailed Balance

- At equilibrium, that stationary distribution satisfies:

$$\pi(x^*) = \int_x \pi(x) K(x \to x^*) \mathrm{d}x \qquad \textbf{equilibrium equation}$$

- Proving **equilibrium equation** may be difficult in some cases, therefore, we instead prove detail balance:
- **detailed balance** condition holds when:

$$\pi(x) K(x \to x^*) = \pi(x^*) K(x^* \to x)$$

- **detailed balance** implies **equilibrium equation**:

$$\begin{aligned}
\int_x \pi(x) K(x \to x^*) \mathrm{d}x &= \int_x \pi(x^*) K(x^* \to x) \mathrm{d}x \\
&= \pi(x^*) \int_x K(x^* \to x) \mathrm{d}x \\
&= \pi(x^*) \qquad \textbf{equilibrium equation}
\end{aligned}$$

- the reverse is not always true.

Extend target distribution with auxiliary variables

- At equilibrium, that stationary distribution satisfies:

$$\pi(x^*) = \int_x \pi(x) K\left(x \to x^*\right) \mathrm{d}x$$

- under many scenarios, we may have an extended joint density (x, u):

$$\pi(x|u)\pi(u)K\left(u, x \to u^*, x^*\right) = \pi\left(x^*|u^*\right)\pi(u^*)K\left(x^*, u^* \to x, u\right)$$

- u is auxiliary variables help samping
- one needs to ensure that:

$$\int_u \pi(x, u) \mathrm{d}u = \pi(x)$$

Alternative Use of Stochastic Matrix

- Before dive deep into MCMC algorithms, let's have a look at alternative use of stochastic matrix
- PageRank algorithm is different to MCMC, in PageRank algorithm: K is known
- PageRank algorithm then computes π which is the **invariant distribution**, tells the importance of each web page.

PageRank algorithm

- Imagine we have the following four web pages and their links
- we can then compute the probability of navigating from i^{th} page (discrete state) to j^{th} page (discrete state)

- Page 1 links to pages $\{2, 3\}$

$$\implies K_{1\to 1} = 0, K_{1\to 2} = \frac{1}{2}, K_{1\to 3} = \frac{1}{2}, K_{1\to 4} = 0$$

- Page 2 has links to pages $\{1, 3, 4\}$

$$\implies K_{2\to 1} = \frac{1}{3}, K_{2\to 2} = 0, K_{2\to 3} = \frac{1}{3}, K_{2\to 4} = \frac{1}{3}$$

- Page 3 has links to pages $\{1, 3\}$

$$\implies K_{3\to 1} = \frac{1}{2}, K_{3\to 2} = 0, K_{3\to 3} = \frac{1}{2}, K_{3\to 4} = 0$$

- Page 4 has links to pages $\{2, 3\}$

$$\implies K_{4\to 1} = 0, K_{4\to 2} = \frac{1}{2}, K_{4\to 3} = \frac{1}{2}, K_{4\to 4} = 0$$

Stochastic matrix K

- From the preceding example, **Left stochastic matrix** is:

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & K_{3\to 1} & K_{4\to 1} \\ K_{1\to 2} & K_{2\to 2} & K_{3\to 2} & K_{4\to 2} \\ K_{1\to 3} & K_{2\to 3} & K_{3\to 3} & K_{4\to 3} \\ K_{1\to 4} & K_{2\to 4} & K_{3\to 4} & K_{4\to 4} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & 0 \end{bmatrix}$$

- From Power Method Convergence Theorem, we know:
  - sequence $\{\pi^{(1)}, K\pi^{(1)}, K^2\pi^{(1)}, \ldots, K^t\pi^{(1)} \ldots, K^\infty\pi^{(1)}\}$ converges to the vector π^*

$$\lim_{t\to\infty} K^t\pi^{(1)} = \pi^*$$

    where π^* is a **probabilistic eigenvector** of K corresponding to the eigenvalue 1.
- **Exercise** What is the usefulness of π^* in the setting of web pages?

Usefulness of π^* in the setting of web pages

The **answer** to usefulness of π^* in the setting of web pages is:

- Shows how **important** each webpage is
- i.e., regardless of the probabilities of the initial webpage visit: $\pi^{(1)}$,
- $\pi^{(1)} \rightarrow \pi^*$, where $\pi^*(i)$ is the target distribution i.e, the probability that the visit will end up at a web page i.
- Note that this is a **reverse problem** of MCMC

Dangling nodes

- What happens when you have the following K:

$$\begin{bmatrix} K_{1\to1} & K_{2\to1} & K_{3\to1} & K_{4\to1} \\ K_{1\to2} & K_{2\to2} & K_{3\to2} & K_{4\to2} \\ K_{1\to3} & K_{2\to3} & K_{3\to3} & K_{4\to3} \\ K_{1\to4} & K_{2\to4} & K_{3\to4} & K_{4\to4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

- Note that 4^{th} has no out-going node
- **Exercise** check eigenvector correspond to eigenvalue of 1
- What is the eigenvector correspond to eigenvalue of 1, if we change K into:

$$\begin{bmatrix} K_{1\to1} & K_{2\to1} & K_{3\to1} & K_{4\to1} \\ K_{1\to2} & K_{2\to2} & K_{3\to2} & K_{4\to2} \\ K_{1\to3} & K_{2\to3} & K_{3\to3} & K_{4\to3} \\ K_{1\to4} & K_{2\to4} & K_{3\to4} & K_{4\to4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & \mathbf{1} \end{bmatrix}$$

  Exercise give reason to why this is so?
- **Exercise** How can we solve this?

Dangling nodes: what may be the solution?

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & K_{3\to 1} & K_{4\to 1} \\ K_{1\to 2} & K_{2\to 2} & K_{3\to 2} & K_{4\to 2} \\ K_{1\to 3} & K_{2\to 3} & K_{3\to 3} & K_{4\to 3} \\ K_{1\to 4} & K_{2\to 4} & K_{3\to 4} & K_{4\to 4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

- One simply solution is:

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & K_{3\to 1} & K_{4\to 1} \\ K_{1\to 2} & K_{2\to 2} & K_{3\to 2} & K_{4\to 2} \\ K_{1\to 3} & K_{2\to 3} & K_{3\to 3} & K_{4\to 3} \\ K_{1\to 4} & K_{2\to 4} & K_{3\to 4} & K_{4\to 4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

- in words, it means any page doesn't have out-link, we assume it has equal probability of visiting entire web.
- Of course, **data mining** researchers may argue certain web page (having certain properties) may attract higher weights etc.

Disconnected sub-graphs

- What happens when you have the following K:

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & K_{3\to 1} & K_{4\to 1} \\ K_{1\to 2} & K_{2\to 2} & K_{3\to 2} & K_{4\to 2} \\ K_{1\to 3} & K_{2\to 3} & K_{3\to 3} & K_{4\to 3} \\ K_{1\to 4} & K_{2\to 4} & K_{3\to 4} & K_{4\to 4} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \end{bmatrix}$$

- node $\{1, 2\}$ and $\{3, 4\}$ each form a sub-graph.
- **Exercise** check eigenvector correspond to eigenvalue of 1, also multiplicity of eigenvalue 1
- **Exercise** How can we solve this?

Disconnected sub-graphs: what may be the solution?

$$\begin{bmatrix} K_{1\to 1} & K_{2\to 1} & K_{3\to 1} & K_{4\to 1} \\ K_{1\to 2} & K_{2\to 2} & K_{3\to 2} & K_{4\to 2} \\ K_{1\to 3} & K_{2\to 3} & K_{3\to 3} & K_{4\to 3} \\ K_{1\to 4} & K_{2\to 4} & K_{3\to 4} & K_{4\to 4} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \end{bmatrix}$$

- One solution is to use a convex combination between K and a square matrix having identical elements $\frac{1}{n}$:

$$K' = (1-p)K + p\left(\frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\right)$$

- in words, it means most of the time $1-p$, a surfer will follow links to navigate a page
- but with probability p, it will arbitrarily close the current page and go to the new one
- **Exercise** Prove K remains a left stochastic matrix

How to compute the **one hundred billion** dimension eigenvector?

- starting from the vector (not probabilistic eigenvector), x:

$$x = [1 \quad 1 \quad \dots \quad 1]^\top$$

- generate the sequence: $\{x, Kx, K^2x \dots K^tx\}$ until convergence then its is the eigenvectors of K correspond to eigenvalue of 1, up to a normalisation constant c
- This is solved using **power method**

Power method

- **power method** is used to finding an eigenvector of a square matrix corresponding to the **largest** eigenvalue (in terms of absolute value)
- for stochastic matrix K has eigenvalues:

$$1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$$

- the initial vector: x as a linear combination of eigenvectors of K:

$$x = c_1 v_1 + c_2 v_2 + \ldots c_n v_n$$

Then,

$$\begin{aligned}
Kx &= K(c_1 v_1 + c_2 v_2 + \ldots c_n v_n) \\
&= c_1 \underbrace{\lambda_1}_{=1} v_1 + c_2 \lambda_2 v_2 + \ldots c_n \lambda_n v_n \qquad \text{definition of eigven value/vector} \\
&= c_1 v_1 + c_2 \lambda_2 v_2 + \ldots c_n \lambda_n v_n \\
\implies K^2 x &= c_1 v_1 + c_2 \lambda_2^2 v_2 + \ldots c_n \lambda_n^2 v_n \\
\implies K^t x &= c_1 v_1 + c_t \lambda_2^t v_2 + \ldots c_n \lambda_n^t v_n
\end{aligned}$$

$$\lambda_j^k \to 0 \text{ when } j \geq 2 \implies K^t x \to c_1 v_1$$

Thank you!

- Questions?