

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

A/Prof Richard Yi Da Xu  
richardxu.com

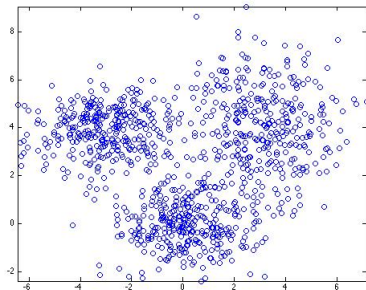
University of Technology Sydney (UTS)

August 1, 2018

- ▶ 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.,:

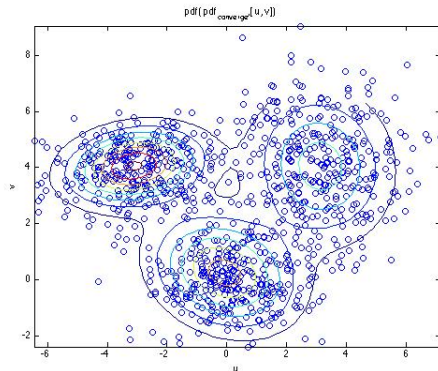
$$\begin{aligned} p(X) &= \mathcal{N}(X|\mu, \Sigma) \\ &= (2\pi)^{-k/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right) \end{aligned}$$

- ▶ Clearly NOT!

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

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

# Gaussian Mixture model result



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

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

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

- ▶ 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:

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

- ▶ **The trick** is to assume some “latent” variable  $Z$  to the model.
- ▶ such that we generate a series of  $\Theta = \{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(i)}\}$
- ▶ For each iteration of the E-M algorithm, we perform:

$$\Theta^{(g+1)} = \arg \max_{\theta} \left( \int_Z \log(p(X, Z|\theta)) p(Z|X, \Theta^{(g)}) \right) dz$$

- ▶ However, we must ensure convergence:

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

# Proof of convergence (1)

$$\begin{aligned}\mathcal{L}(\theta|X) &= \ln[p(X|\theta)] = \ln[p(Z, X, \theta)] - \ln[p(Z|X, \theta)] \\&\Rightarrow \int_{z \in \mathbb{S}} \ln[p(X|\theta)] p(z|X, \Theta^{(g)}) dz \\&= \int_{z \in \mathbb{S}} \ln[p(Z, X, \theta)] p(z|X, \Theta^{(g)}) dz - \int_{z \in \mathbb{S}} \ln[p(Z|X, \theta)] p(z|X, \Theta^{(g)}) dz \\&\Rightarrow \ln[p(X|\theta)] = \underbrace{\int_{z \in \mathbb{S}} \ln[p(Z, X, \theta)] p(z|X, \Theta^{(g)}) dz}_{Q(\theta, \Theta^{(g)})} - \underbrace{\int_{z \in \mathbb{S}} \ln[p(Z|X, \theta)] p(z|X, \Theta^{(g)}) dz}_{H(\theta, \Theta^{(g)})}\end{aligned}$$

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:

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

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)

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

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

$$\begin{aligned} H(\Theta^{(g)}, \Theta^{(g)}) - H(\theta, \Theta^{(g)}) &= \int_{z \in \mathbb{S}} \ln[p(Z|X, \Theta^{(g)})] p(z|X, \Theta^{(g)}) dz - \int_{z \in \mathbb{S}} \ln[p(Z|X, \theta)] p(z|X, \Theta^{(g)}) dz \\ &= \int_{z \in \mathbb{S}} \ln \left[ \frac{p(Z|X, \Theta^{(g)})}{p(Z|X, \theta)} \right] p(z|X, \Theta^{(g)}) dz = \int_{z \in \mathbb{S}} -\ln \left[ \frac{p(Z|X, \theta)}{p(Z|X, \Theta^{(g)})} \right] p(z|X, \Theta^{(g)}) dz \\ &\geq -\ln \left[ \int_{z \in \mathbb{S}} \frac{p(Z|X, \theta)}{p(Z|X, \Theta^{(g)})} p(z|X, \Theta^{(g)}) dz \right] = 0 \end{aligned}$$

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



# E-M Example: Gaussian Mixture Model

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

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

and  $\theta = \{\alpha_1, \dots, \alpha_k, \mu_1, \dots, \mu_k, \Sigma_1, \dots, \Sigma_k\}$

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

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

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

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

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

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

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

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

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

$$\begin{aligned} Q(\Theta, \Theta^{(g)}) &= \int_{\mathbf{Z}} \ln(p(\mathbf{X}, \mathbf{Z}|\Theta)) p(\mathbf{Z}|\mathbf{X}, \Theta^{(g)}) d\mathbf{z} \\ &= \int_{z_1} \cdots \int_{z_n} \left( \sum_{i=1}^n \ln p(z_i, x_i|\Theta) \prod_{i=1}^n p(z_i|x_i, \Theta^{(g)}) \right) dz_1, \dots, dz_n \end{aligned}$$

## The E-Step: (2)

Knowing,

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

$$\begin{aligned} Q(\Theta, \Theta^{(g)}) &= \int_{z_1} \cdots \int_{z_n} \left( \sum_{i=1}^n \ln p(z_i, x_i | \Theta) \prod_{i=1}^n p(z_i | x_i, \Theta^{(g)}) \right) dz_1, \dots, dz_n \\ &= \sum_{i=1}^n \left( \int_{z_i} \ln p(z_i, x_i | \Theta) p(z_i | x_i, \Theta^{(g)}) dz_i \right) \quad z_i \in \{1, \dots, k\} \\ &= \sum_{z_i=1}^k \sum_{i=1}^n \ln p(z_i, x_i | \Theta) p(z_i | x_i, \Theta^{(g)}) \quad \text{swap the summation terms} \\ &= \sum_{l=1}^k \sum_{i=1}^n \ln [\alpha_l \mathcal{N}(x_i | \mu_l, \Sigma_l)] p(l | x_i, \Theta^{(g)}) \quad \text{substitute Gaussian and replace } z_i \rightarrow l \end{aligned}$$

# The M-Step objective function

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

- ▶ The first term contains only  $\alpha$
- ▶ second term contains only  $\mu, \Sigma$ .
- ▶ So we can maximize both terms independently.

# The M-Step: maximizing $\alpha$

Maximizing  $\alpha$  means that:

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

This is to be solved using Lagrange Multiplier

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

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

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

# The M-Step: maximizing $\mu, \Sigma$

second part of  $Q(\Theta, \Theta^{(g)})$

$$\begin{aligned}\equiv \mathcal{S}(\mu_l, \Sigma_l) &= \sum_{l=1}^k \sum_{i=1}^n \ln[\mathcal{N}(x_i | \mu_l, \Sigma_l)] p(l | x_i, \Theta^{(g)}) \\ &= \sum_{i=1}^n \sum_{l=1}^k \ln \left( \frac{1}{\sqrt{(2\pi)^d |\Sigma_l|}} \exp \left( -\frac{1}{2} (x_i - \mu_l)^T \Sigma_l^{-1} (x_i - \mu_l) \right) \right) p(l | x_i, \Theta^{(g)}) \\ &= \sum_{i=1}^n -\frac{1}{2} \ln(|\Sigma_l|) p(l | x_i, \Theta^{(g)}) - \sum_{i=1}^n \frac{1}{2} (x_i - \mu_l)^T \Sigma_l^{-1} (x_i - \mu_l) p(l | x_i, \Theta^{(g)})\end{aligned}$$

how do we maximize  $\mu, \Sigma$ ?

# 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

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

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

- ▶ Derivatives of Traces of inverse, fact 1

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

- ▶ Derivatives of Traces of inverse, fact 2

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

- ▶ Derivatives of Traces of inverse, fact 3

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



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

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

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

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

# Maximization $\Sigma_I$

- ▶ 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  $x_i - \mu_I$
- ▶ let  $\mathbf{P}$  be diagonal matrix in which  $\mathbf{P}_{ii}$  correspond to  $p(l|x_i, \Theta^{(g)})$

$$\begin{aligned} \mathcal{S}(\mu_I, \Sigma_I) &= \sum_{i=1}^n -\frac{1}{2} \ln(|\Sigma_I|) p(l|x_i, \Theta^{(g)}) - \sum_{i=1}^n \frac{1}{2} (x_i - \mu_I)^T \Sigma_I^{-1} (x_i - \mu_I) p(l|x_i, \Theta^{(g)}) \\ &= -\frac{\text{tr}(\mathbf{P})}{2} \ln |\Sigma_I| - \frac{1}{2} \text{tr}(\Sigma_I^{-1} \mathcal{Y} \mathbf{P} \mathcal{Y}^T) \end{aligned}$$

- ▶ now taking the derivative:

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

# Summary of Gaussian Mixture Model

- Maximizing  $\mu, \Sigma, \alpha$  to update  $\Theta^{(g)} \rightarrow \Theta^{(g+1)}$ :

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

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

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

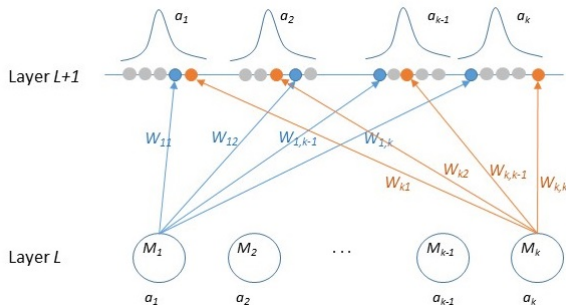
- responsibility probabilities:

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

# 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  $\{M_1, \dots, M_k\}$  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  $\Theta \equiv \{\mu_1, \dots, \mu_k, \sigma_1, \dots, \sigma_k\}$  at layer  $L + 1$  capsules.

# Apply EM to Matrix Capsule Routing algorithm

## ► M-STEP:

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

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

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

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

## ► E-STEP:

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

$$R_{ij} \leftarrow \frac{a_j p_j}{\sum_{k \in \Omega_{L+1}} a_k p_k} \quad \text{apply } a_j$$

# Apply EM to Matrix Capsule Routing algorithm

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

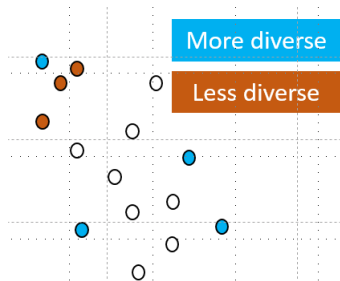
$$\begin{aligned}\mathcal{C}_j^h(\mu, \sigma) &= \sum_i R_{ij} \mathcal{C}_{ij}^h \\&= \sum_i -R_{ij} \ln(p_{i|j}^h) \\&= \sum_i R_{ij} \left( \frac{(\mathbf{v}_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + \ln(\sigma_j^h) + \frac{\ln(2\pi)}{2} \right) \\&= \sum_i R_{ij} \left( \frac{(\bar{\sigma}_j^h)^2}{2(\sigma_j^h)^2}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

### **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  $A$

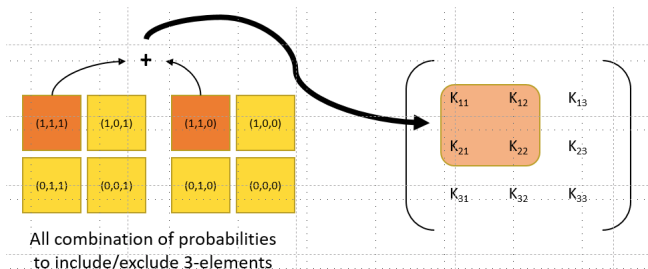
$$\Pr(A \subseteq \mathbf{Y}) = \det(K_A)$$

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

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

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

# 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**

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

- ▶ 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

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

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

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

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

However,

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

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:

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

- ▶  $\mathcal{Y} = \{1, 2\}$

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

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

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

- ▶ So where do rest of probabilities go?

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

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

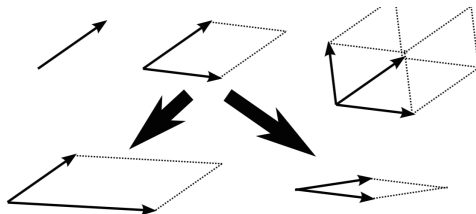
- ▶ Marginal distributions does **not** define probabilities in terms of a **particular** set
- ▶ i.e., instead of having  $\Pr(\mathbf{Y} \subseteq Y)$ , we want  $\Pr(\mathbf{Y} = Y)$

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

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

$$X = [x_1 \quad x_2 \quad \dots \quad x_n] \implies$$
$$L(x_1, \dots, x_n) = X^T X = \begin{pmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \dots & \langle x_n, x_n \rangle \end{pmatrix}$$

- ▶ 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
- ▶  $\Pr_L(Y) \propto \det(L_Y) = \text{Vol}^2(\{x_i\}_{i \in Y})$





# 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  $\text{span}(\mathbf{u}_1, \dots, \mathbf{u}_k)$ : imagine 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:

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

# Proof for the Geometry interpretation (2)

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

$$\begin{aligned}\mathbf{X} &= [\mathbf{Y} \quad \mathbf{u}_{k+1}] = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \dots \quad \mathbf{u}_k \quad \mathbf{u}_{k+1}] = [\mathbf{Y} \quad \mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1}] \\ \Rightarrow \mathbf{X}^\top \mathbf{X} &= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top (\mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1}) \\ (\mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1})^\top \mathbf{Y} & (\mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1})^\top (\mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Y}\mathbf{c} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{c} + \tilde{\mathbf{u}}_{k+1}^\top \tilde{\mathbf{u}}_{k+1} \end{bmatrix} \quad \text{since } \mathbf{Y}^\top \tilde{\mathbf{u}}_{k+1} = \mathbf{0} \\ &= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Y}\mathbf{c} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{c} + \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Y}\mathbf{c} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \left( \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{c} + \|\tilde{\mathbf{u}}_{k+1}\|^2 \right) \end{bmatrix}\end{aligned}$$

► **Multi-linearity** states:

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

$$\begin{aligned}\Rightarrow |\mathbf{X}^\top \mathbf{X}| &= \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{Y}^\top \mathbf{Y}\mathbf{c} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{c} \end{vmatrix} + \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{0} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{vmatrix} \\ &= \mathbf{0} + \begin{vmatrix} \mathbf{Y}^\top \mathbf{Y} & \mathbf{0} \\ \mathbf{c}^\top \mathbf{Y}^\top \mathbf{Y} & \|\tilde{\mathbf{u}}_{k+1}\|^2 \end{vmatrix} \\ &= |\mathbf{Y}^\top \mathbf{Y}| \underbrace{\|\tilde{\mathbf{u}}_{k+1}\|^2}_{\text{Vol}^2(\tilde{\mathbf{u}}_{k+1})}\end{aligned}$$

**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:

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

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

## Conversion to Marginal distribution (2)

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

# Conversion to Marginal distribution (3)

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

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

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

- Re-organise:

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

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

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

► **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  $K = I - (L + I)^{-1}$ :

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

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

► **Therefore,**

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

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

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



# 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, Kjellström, Mandt, Determinantal Point Processes for Mini-Batch Diversification”*

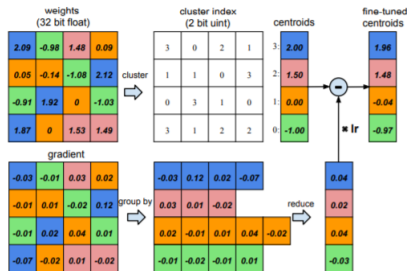
- ▶ as well as a lot of applications such as diverse subset of news collection portal
- ▶ today, we talk about an example of it it used in neural network compression

*“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  $\tau = \{I_1, \dots, I_J\}$  be the total number of  $J$  training data
- ▶  $a_{ij}$  is activation of neuron  $i$  on image  $j$ , then
- ▶  $\mathbf{v}_i$  contains a vector of all activation  $[a_{i,1}, a_{i,1}, \dots, a_{i,J}]^T$

# 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:

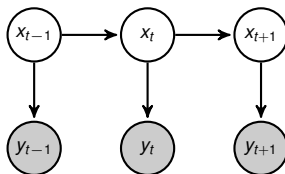
$$\mathbf{L}'_{ij} = \exp(-\beta \|\mathbf{v}_i - \mathbf{v}_j\|^2) \quad 1 \leq i, j \leq n_l$$

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

## LSTM to help Kalman Filter

- ▶ 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:  $y_1, y_2, \dots, y_t$
- ▶ **insert some pictures of application of Kalman Filter**

- ▶ graphical model for state space model:



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

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

- ▶ looks familiar to Recurrent Neural Networks?

# What do we want to compute?

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

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

This is because:

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



$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B + w_t \quad w_t \sim \mathcal{N}(0, Q_t)$$

$$\Rightarrow \text{Transition probability:} \quad p(\mathbf{x}_t | \mathbf{x}_{t-1}) \sim \mathcal{N}(A\mathbf{x}_{t-1} + B, Q_t)$$

$$y_t = H\mathbf{x}_t + v_t \quad v_t \sim \mathcal{N}(0, R_t)$$

$$\Rightarrow \text{Measurement probability:} \quad p(y_t | \mathbf{x}_t) \sim \mathcal{N}(H\mathbf{x}_t, R_t)$$

- ▶ 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.

- ▶ 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:

$$p(X_2) = \int_{x_1} p(X_2 | x_1) p(x_1) \sim \mathcal{N}(x_2 | A\mu + B, A\Sigma A^T + Q) dx_1$$

**Prediction :**

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

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

# Kalman Filter Update: $p(\mathbf{x}_t | \mathbf{y}_1, \dots, \mathbf{y}_t) = \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{y}_1, \dots, \mathbf{y}_t) &\sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t) \\ &\propto p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{y}_t | H\mathbf{x}_t, R_t) \mathcal{N}(\mathbf{x}_t | \bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t) \end{aligned}$$

- ▶ given **marginal**  $p(\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$        $p(\mathbf{y}_t) = \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy})$
- ▶ and **joint** density,  $\mathbf{x}_t \equiv x_t | y_1, \dots, y_{t-1}$  and  $\mathbf{y}_t \equiv y_t | y_1, \dots, y_{t-1}$

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{xy} & \boldsymbol{\Sigma}_{yy} \end{bmatrix} \right)$$

- ▶ then, **conditional** density is:

$$p(\mathbf{x}_t | \mathbf{y}_t) \sim \mathcal{N} \left( \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y}_t - \boldsymbol{\mu}_y), \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx} \right)$$

- ▶ In filtering:

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

- ▶ 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:  $(\mu_x, \mu_y, \Sigma_{xx}, \Sigma_{yy}, \Sigma_{xy}, \Sigma_{yx})$
- ▶ introduce a zero-mean variable:  $\Delta x_{t-1}$ :

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

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

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

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

The Independence assumptions:

- ▶  $\text{COV}(x_{t-1}, \mathbf{w}_t) = 0 \quad \text{COV}(x_{t-1}, v_t) = 0 \quad \text{COV}(\mathbf{w}_t, v_t) = 0$

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

$$\mu_x \equiv \mathbb{E}[x_t | y_{1:t-1}] = A\mathbb{E}[x_{t-1}] = A\hat{\mu}$$

$$\mu_y \equiv \mathbb{E}[y_t | y_{1:t-1}] = HA\mathbb{E}[x_{t-1}] = HA\hat{\mu}$$

$$\begin{aligned}\Sigma_{xx} &\equiv \mathbb{E}[\Delta x_t (\Delta x_t)^T | y_{1:t-1}] = \mathbb{E}[(A\Delta x_{t-1} + w_t)(A\Delta x_{t-1} + w_t)^T] \\ &= A\hat{\Sigma}_{t-1}A^T + Q_t = \bar{\Sigma}_t\end{aligned}$$

$$\begin{aligned}\Sigma_{yx} &\equiv \mathbb{E}[\Delta y_t (\Delta x_t)^T | y_{1:t-1}] = \mathbb{E}[(HA\Delta x_{t-1} + Hw_t + v_t)(A\Delta x_{t-1} + w_t)^T] \\ &= H(A\hat{\Sigma}_{t-1}A^T + Q_t) = H\bar{\Sigma}_t\end{aligned}$$

$$\begin{aligned}\Rightarrow \Sigma_{yx} &\equiv \mathbb{E}[\Delta x_t (\Delta y_t)^T | y_{1:t-1}] \\ &= (A\hat{\Sigma}_{t-1}A^T + Q_t)^T H^T = \bar{\Sigma}_t H^T\end{aligned}$$

$$\begin{aligned}\Sigma_{yy} &\equiv \mathbb{E}[\Delta y_t (\Delta y_t)^T | y_{1:t-1}] = \mathbb{E}[(HA\Delta x_{t-1} + Hw_t + v_t)(HA\Delta x_{t-1} + Hw_t + v_t)^T] \\ &= H(A\hat{\Sigma}_{t-1}A^T + Q_t)H^T + R_t \\ &= H(\bar{\Sigma}_t)H^T + R_t\end{aligned}$$

# Kalman Filter Update: $p(\mathbf{x}_t | \mathbf{y}_1, \dots, \mathbf{y}_t) = \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$

- ▶ finally we put all elements in:
- ▶ **mean:**  $\hat{\boldsymbol{\mu}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{y}_{1:t}]$ :

$$\begin{aligned} & \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y) \\ & \equiv \mathbb{E}[\mathbf{x}_t] + \mathbb{E}[\Delta \mathbf{x}_t (\Delta \mathbf{y}_t)^T] \mathbb{E}[\Delta \mathbf{y}_t (\Delta \mathbf{y}_t)^T]^{-1} (\mathbf{y}_t - \mathbb{E}[\mathbf{y}_t]) \\ & = A \hat{\boldsymbol{\mu}}_{t-1} + \underbrace{\bar{\boldsymbol{\Sigma}}_t^T H (H \bar{\boldsymbol{\Sigma}}_t H^T + R_t)^{-1}}_K (\mathbf{y}_t - H A \hat{\boldsymbol{\mu}}_{t-1}) \\ & = \bar{\boldsymbol{\mu}}_{t-1} + K (\mathbf{y}_t - H A \hat{\boldsymbol{\mu}}_{t-1}) \end{aligned}$$

**co-variance:**  $\hat{\boldsymbol{\Sigma}}_t = \text{COV}[\mathbf{x}_t | \mathbf{y}_{1:t}]$ :

$$\begin{aligned} & \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx} \\ & \equiv \mathbb{E}[\Delta \mathbf{x}_t (\Delta \mathbf{x}_t)^T] - \mathbb{E}[\Delta \mathbf{x}_t (\Delta \mathbf{y}_t)^T] \mathbb{E}[\Delta \mathbf{y}_t (\Delta \mathbf{y}_t)^T]^{-1} \mathbb{E}[\Delta \mathbf{y}_t (\Delta \mathbf{x}_t)^T] \\ & = \bar{\boldsymbol{\Sigma}}_t - \underbrace{\bar{\boldsymbol{\Sigma}}_t H^T (H \bar{\boldsymbol{\Sigma}}_t H^T + R_t)^{-1} H \bar{\boldsymbol{\Sigma}}_t}_K \\ & = (I - KH) \bar{\boldsymbol{\Sigma}}_t \end{aligned}$$

- ▶ 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*”

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

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

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

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

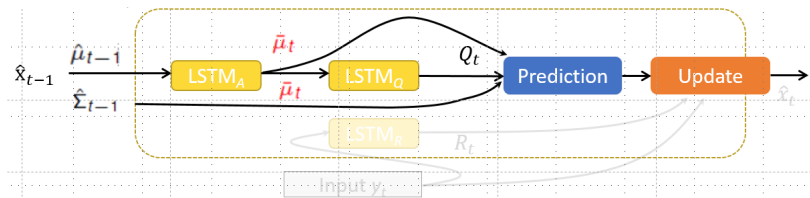
$$\begin{aligned} C_t &= f_t(C_{t-1}) \odot C_{t-1} + i_t(C_{t-1}) \odot \tilde{C}_t(C_{t-1}) \\ &= \color{red}{f_t} \odot C_{t-1} + i_t \odot \color{red}{\tanh(W_C[h_{t-1}, x_t] + b_C)} \\ &= \color{red}{f_t(C_{t-1})} C_{t-1} + \underbrace{i_t(h_{t-1}(C_{t-1})) \color{red}{\tanh(W_C[o_{t-1}(h_{t-1}(C_{t-1}))]}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

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

- ▶ 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



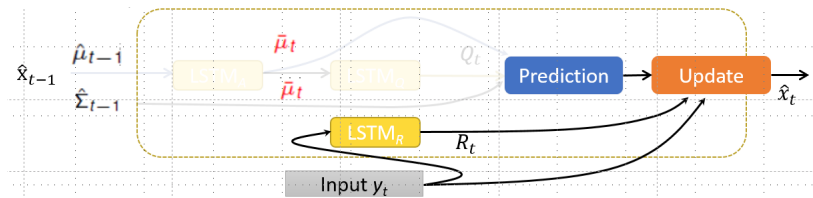
- the state-space model is changed into:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_t & \mathbf{w}_t &\sim \mathcal{N}(0, \mathbf{Q}_t) & \rightarrow & \mathbf{x}_t = \text{lstm}_A(\mathbf{x}_{t-1}) + \mathbf{w}_t & \mathbf{w}_t &\sim \mathcal{N}(0, \mathbf{Q}_t) \\ \mathbf{y}_t &= \mathbf{H}\mathbf{x}_t + \mathbf{v}_t & \mathbf{v}_t &\sim \mathcal{N}(0, \mathbf{R}_t) & \rightarrow & \text{unchanged} \end{aligned}$$

- prediction

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

# LSTM Kalman Filters: Update



- the state-space model is changed into:

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

- update:

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

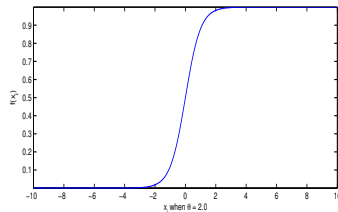
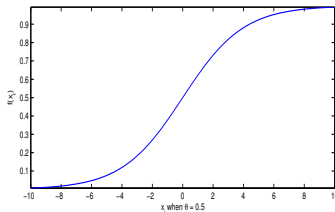
## Binary Classifier to help probability model estimation

- firstly, probability models and classification are closely related:

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

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

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



# Example: Bernoulli & Logistic regression

- Bernoulli distribution using Sigmoid function

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

- Logistic regression

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

# Example: Multinomial Distribution & Cross Entropy Loss

- Multinomial Distribution with softmax

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

- cross entropy loss with Softmax

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

# Example: Gaussian Distribution & Sum of Square Loss

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

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

- ▶ Gaussian distribution

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

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

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

# 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  $\mathbf{c} \equiv \mathbf{v}_c$

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

- ▶ 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  $q(\cdot)$  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?

- ▶ **training data generation:**  $(\mathbf{w}, \mathbf{c}, y)$ 
  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  $q(\cdot)$ , and label them as  $\mathcal{Y} = 0$
- ▶ can we instead, try to maximize the joint posterior Bernoulli distribution:

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

- ▶ or minimize the corresponding Logistic regression:

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

# 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  $P(\mathcal{Y}|\mathbf{c}, \mathbf{w})$ :

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

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

- ▶ 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:

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

- ▶ say  $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k\}$  are target words having high frequencies given  $\mathbf{c}$
- ▶  $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n\}$  are words having low frequency given  $\mathbf{c}$
- ▶ say we pick  $\mathbf{w}_i \in \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$  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:  $\{\{\mathbf{w}_j^{\top} \mathbf{c}\}_{j \neq i} \cup \{\mathbf{r}_j^{\top} \mathbf{c}\}\}$  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  $\mathbf{r}_j^{\top} \mathbf{c}$  will (hopefully) stay low
- ▶ **intuition:** in NCE, instead of using sum in the denominator, we “designed” a probability  $q(\cdot)$ , such that, while letting  $\mathbf{w}_j$  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** 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

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

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

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

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

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

- therefore the objective function is:

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

# NCE and Negative Sampling

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

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

- ▶ in Skip-gram:

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



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

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

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

where  $\Delta s_\theta(\mathbf{w}|\mathbf{c}) \equiv s_\theta(\mathbf{w}|\mathbf{c}) - \log(kq(\mathbf{w}))$

- we already know:

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

- in this case,

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

- 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  $\theta$ , 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  $m$ ):

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

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

► let,

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

► let  $n \rightarrow \infty$  and  $m \rightarrow \infty$ :  $C_n \rightarrow \mathcal{C}$ :

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

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

► take functional derivative:

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

## Probability density re-parameterization

- ▶ we **love** to have integral in a form:

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

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

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

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

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

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

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

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

$$p_{\theta}(\Pi) \equiv p_{\theta}(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t)$$
$$\implies \theta^* = \arg \max_{\theta} \left\{ \mathbb{E}_{\Pi \sim p_{\theta}(\Pi)} \left[ \underbrace{\sum_{t=1}^T R(s_t, a_t)}_{f(\mathbf{z})} \right] \right\}$$

# 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:

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

- ▶ can approximated by:

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

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



# Re-parameterization trick

- ▶ we let  $z = g(x)$ :

$$\mathbb{E}_{x \sim p(x)}[g(x)] = \mathbb{E}_{z \sim p(z)}[z]$$

$$\mathbb{E}_{x \sim p(x)}[g(x, \theta)] = \mathbb{E}_{z \sim p_{\theta}(z)}[z] \quad \text{parameterize the distribution with } \theta$$

$$\mathbb{E}_{x \sim p(x)}[f(g(x, \theta))] = \mathbb{E}_{z \sim p_{\theta}(z)}[f(z)] \quad \text{introduce function } f(\cdot)$$

$$\int_{x \in \Omega_x} f(g(x, \theta)) \color{red}{p(x) dx} = \int_{z \in \Omega_z} f(z) \color{blue}{p_{\theta}(z) dz}$$

- ▶ 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:

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

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

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

$$f_Y(y) = \frac{dF_Y(y)}{dy} = \frac{dF_X(x)}{dy} = \frac{dF_X(x)}{dx} \frac{dx}{dy} = f_X(x) \frac{dx}{dy}$$

- ▶ without change of limits

$$f_Y(y)|dy| = f_X(x)|dx|$$

- ▶ with change of limits

$$f_Y(y)dy = f_X(x)dx$$

- ▶ **main motivation**  $p(x)$  is **no longer** parameterized by  $\theta$ :

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

- ▶ during gradient decent,  $x$  are sampled independent of  $\theta$

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

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

- ▶ 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  $\mu$  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:

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

- ▶ very hard!

# solve it using **REINFORCE** trick

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

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

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

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

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

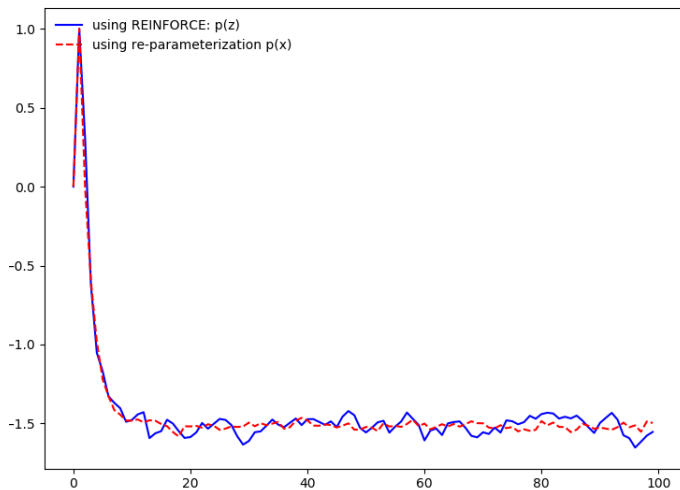
$$z \equiv g(x, \theta) = \mu(\theta) + x\sigma(\theta)$$

- ▶ the re-parameterised version is:

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

- ▶ 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  $p(Z)$

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



- ▶ when we have the following

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

- ▶ can we find their corresponding:

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



# Re-parameterization using Gumbel-max trick

- ▶ Gumbel-max trick also means:

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

- ▶ this is a form of re-parameterization:  
instead of sample  $\mathcal{K} \sim \text{softmax}(\mu_1(\theta), \dots, \mu_K(\theta))$ , 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  $\mu$ :

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

- ▶ CDF of Gumbel:

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

# 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  $z_k$ :

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

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

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

- keep on going:

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

## Gumbel-max trick and Softmax (2)

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

$$K \sim \left\{ \frac{\exp(\mu_1)}{\sum_i \exp(\mu_i)}, \dots, \frac{\exp(\mu_L)}{\sum_i \exp(\mu_i)} \right\}$$
$$\implies K = \arg \max_{i \in \{1, \dots, L\}} \{G_1, \dots, G_L\}$$

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

$$\implies K = \arg \max_{i \in \{1, \dots, L\}} \{\mu_1 + \mathcal{G}, \dots, \mu_L + \mathcal{G}\}$$

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

- ▶ what is  $\mu_i$ ? 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 :

$$\equiv \arg \max_{i \in \{1, \dots, L\}} \{\log(\mu_1) + \mathcal{G}, \dots, \log(\mu_L) + \mathcal{G}\}$$

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:

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

## Second problem with Softmax re-parameterisation

- ▶ the other remaining **problem**: sample  $\mathbf{v}$  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*”

## ► softmax map

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

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

- **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

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

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

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

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

- ▶ **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.
- ▶  $\pi^*$  its **probabilistic eigenvector** corresponding to the eigenvalue 1.

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

- ▶ 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  $\pi^*$

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

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

- ▶ in the **discrete** case:

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

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

$$\pi(x^{(n+1)}) = \int_{x^{(n)}} \pi(x^{(n)}) K(x^{(n)} \rightarrow x^{(n+1)})$$

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

$$\pi^{(2)}(x^{(2)}) = \int_{x^{(1)}} \pi^{(1)}(x^{(1)}) K(x^{(1)} \rightarrow x^{(2)}) dx^{(1)}$$

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

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

- ▶ This says,

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

- ▶ We know,

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

- ▶ But, in practice,

$$\lim_{t \rightarrow B} \pi^{(t)}(x^{(t)}) \rightarrow \pi(x^{(t)})$$

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



# What is MCMC research is all about

- **equilibrium equation:**

$$\pi(x^*) = \int_x \pi(x) K(x \rightarrow x^*) dx$$

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

- ▶ At equilibrium, that stationary distribution satisfies:

$$\pi(x^*) = \int_x \pi(x) K(x \rightarrow x^*) dx \quad \text{equilibrium equation}$$

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

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

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

$$\begin{aligned} \int_x \pi(x) K(x \rightarrow x^*) dx &= \int_x \pi(x^*) K(x^* \rightarrow x) dx \\ &= \pi(x^*) \int_x K(x^* \rightarrow x) dx \\ &= \pi(x^*) \quad \text{equilibrium equation} \end{aligned}$$

- ▶ the reverse is not always true.

# Extend target distribution with auxiliary variables

- ▶ At equilibrium, that stationary distribution satisfies:

$$\pi(x^*) = \int_x \pi(x) K(x \rightarrow x^*) dx$$

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

$$\pi(x|u)\pi(u)K(u, x \rightarrow u^*, x^*) = \pi(x^*|u^*)\pi(u^*)K(x^*, u^* \rightarrow x, u)$$

- ▶  $u$  is auxiliary variables help sampling
- ▶ one needs to ensure that:

$$\int_u \pi(x, u) du = \pi(x)$$

- ▶ 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  $\pi$  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  $i^{\text{th}}$  page (discrete state) to  $j^{\text{th}}$  page (discrete state)

- ▶ Page 1 links to pages  $\{2, 3\}$

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

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

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

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

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

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

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

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

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

- ▶ From Power Method Convergence Theorem, we know:

- ▶ sequence  $\{\pi^{(1)}, K\pi^{(1)}, K^2\pi^{(1)}, \dots, K^t\pi^{(1)}, \dots, K^\infty\pi^{(1)}\}$  converges to the vector  $\pi^*$

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

where  $\pi^*$  is a **probabilistic eigenvector** of  $K$  corresponding to the eigenvalue 1.

- ▶ **Exercise** What is the usefulness of  $\pi^*$  in the setting of web pages?

The **answer** to usefulness of  $\pi^*$  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

- What happens when you have the following  $K$ :

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

- Note that 4<sup>th</sup> 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:

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

**Exercise** give reason to why this is so?

- **Exercise** How can we solve this?



# Dangling nodes: what may be the solution?

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

- ▶ One simply solution is:

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

- ▶ 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.

- What happens when you have the following  $K$ :

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

- 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?

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

- ▶ 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),  $x$ :

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

- ▶ generate the sequence:  $\{x, Kx, K^2x \dots K^t x\}$  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** 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:

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

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

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

Then,

$$\begin{aligned} Kx &= K(c_1 v_1 + c_2 v_2 + \dots c_n v_n) \\ &= c_1 \underbrace{\lambda_1}_{=1} v_1 + c_2 \lambda_2 v_2 + \dots c_n \lambda_n v_n \quad \text{definition of eigen value/vector} \\ &= c_1 v_1 + c_2 \lambda_2 v_2 + \dots c_n \lambda_n v_n \\ \implies K^2 x &= c_1 v_1 + c_2 \lambda_2^2 v_2 + \dots c_n \lambda_n^2 v_n \\ \implies K^t x &= c_1 v_1 + c_t \lambda_2^t v_2 + \dots c_n \lambda_n^t v_n \\ \lambda_j^k &\rightarrow 0 \text{ when } j \geq 2 \implies K^t x \rightarrow c_1 v_1 \end{aligned}$$

# Thank you!

► Questions?