Bayesian Inference and Deep Learning

A/Prof Richard Yi Da Xu
Yida.Xu@uts.edu.au
Wechat: aubedata
`https://github.com/roboticcam/machine-learning-notes`

University of Technology Sydney (UTS)

March 3, 2021

before we begin

preambles

In a Deep Learning denominated world . . .

- Machine Learning has been dominated by Deep Learning since early 2010s
- many “hot” buzz-word Deep Learning frameworks has been over-studied:
  - various Neural Network architectures... Convolution, Recurrent, Residual, Self Attention, . . .
  - Generative Adversarial networks, Normalizing Flow, Noise Contrast Estimation
  - Reinforcement Learning, Meta Learning, Transfer Learning
  - . . .
- However, Deep Learning also has its drawbacks, they in general
  - require very large amount of data to perform better than other techniques
  - predominantly used for supervised learning, where correctness of labels are paramount
  - difficult to study the confidence of the result

In a Bayesian Framework

$$\underbrace{p(\theta|X)}_{\text{posterior}} = \frac{\underbrace{p(X|\theta)}_{\text{likelihood}}\,\underbrace{p(\theta)}_{\text{prior}}}{\underbrace{p(X)}_{\text{normalization constant}}}$$

$$= \frac{p(X|\theta)p(\theta)}{\int_\theta p(X|\theta)p(\theta)}$$

- **interpretability** can determine a % credible interval of true parameter
- **natural way to incorporate prior information** for example, in sequential modeling, posterior of state at $t-1$ becomes prior at time t
- **naturally applicable** to many unsupervised learning tasks, such as hierarchical models
- **readily available tools** many numerical methods such as MCMC available for inference
- ...

so the big question...

Can we bring the best of both worlds together?

Table of content (1)

Table of content (2)

- in this talk, I show a mixture of:
  1. work by other researchers
  2. our published and current work in this topic
  3. making references to my **detailed tutorials**
  4. no nice pictures, as I will hand-draw them using stylus

Part 1: Bayesian Inference Tools

1. Variational inference: one page
2. Monte-Carlo inference: discussed in detail
3. Laplace approximation: not discussed at all

Bayesian Inference Tools: Variational Inference

$$\begin{aligned}
\ln(p(\mathbf{x})) &= \log \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \\
&= \log \int \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \qquad \text{can also be just } q(\mathbf{z}) \quad \text{approximate distribution} \\
&= \log \int \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} p(\mathbf{x}|\mathbf{z})\right) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \quad \text{re-arrange} \\
&\geq \int \log \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} p(\mathbf{x}|\mathbf{z})\right) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \quad \text{Jensen's equality} \\
&= \mathcal{L}(q) \qquad \text{Evidence Lower BOund (ELBO)}
\end{aligned}$$

- ELBO split **one**

$$\begin{aligned}
&= \int \log p(\mathbf{x}|\mathbf{z}) q(\mathbf{z}|\mathbf{x})d\mathbf{z} + \int \log \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \int \log \left(\frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}\right) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \text{KL}\left[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \text{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right]
\end{aligned}$$

- ELBO split **two**

$$\begin{aligned}
&= \int \log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) q(\mathbf{z}|\mathbf{x})d\mathbf{z} + \int \log \left(\frac{1}{q(\mathbf{z}|\mathbf{x})}\right) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z})\right] - \int \log q(\mathbf{z}|\mathbf{x}) q(\mathbf{z}|\mathbf{x})d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]
\end{aligned}$$

my tutorial "Variational Inference" `https://github.com/roboticcam/machine-learning-notes/blob/master/files/variational.pdf`

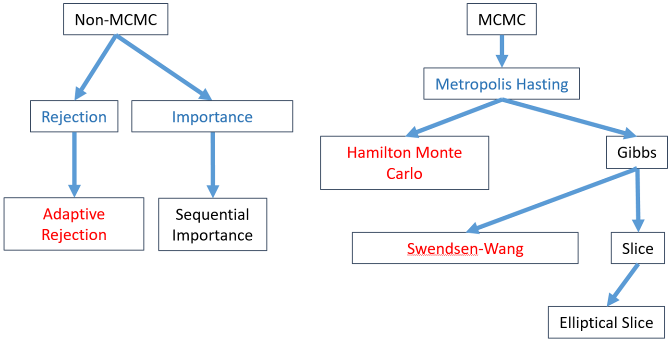
Bayesian Inference via Monte-Carlo

1. **Non-MCMC**:
   Rejection, adaptive rejection, importance, Sequential Monte Carlo

   my tutorial "Monte-Carlo method: an introduction"
   https://github.com/roboticcam/machine-learning-notes/blob/master/files/introduction_monte_carlo.pdf
2. **MCMC**:
   Metropolis-Hasting, Gibbs, Slice, Swendsen-Wang, Hamiltonian Monte-Carlo
   my tutorial "Stochastic Matrices"
   https://github.com/roboticcam/machine-learning-notes/blob/master/files/stochastic_matrices.pdf
   my tutorial "Markov Chain Monte Carlo"
   https://github.com/roboticcam/machine-learning-notes/blob/master/files/markov_chain_monte_carlo.pdf

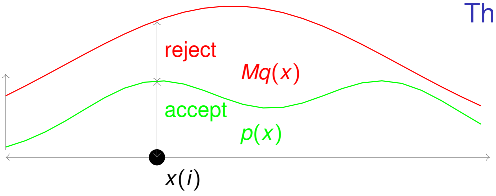
Hierarchies of Monte Carlo Inference methods

there were a **subset** of methods either I used, or I wrote into my tutorial:



let me draw the ones in blue and show Python demo to ones in red

Rejection Sampling



- Sampling is all about efficiency
- Rejection sampling can give you quite low acceptance ratio, should you choose a non-compatible $q(.)$
- let's take a look at **adaptive rejection sampling**!

The algorithm

```
$i = 0$
while $i \neq N$
    $x(i) \sim q(x)$ and $u \sim U(0,1)$
    if $u < \frac{p(x(i))}{Mq(x(i))}$ then
        accept $x(i)$
        $i = i + 1$
    else
        reject $x(i)$
    end
end
```

Simplest MCMC algorithm: Metropolis Hasting

```
1. initialize $x^{(0)}$
2. for i = 0 to $N-1$
       $u \sim \mathbf{U}(0,1)$
       $x^* \sim q(x^*|x^{(i)})$
       if $u < \min\left(1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right)$
           $x^{(i+1)} = x^*$
       else
           $x^{(i+1)} = x^{(i)}$
```

- The take-home message here, is that it does not “discard” samples like rejection sampling. It simply “repeats” samples.
- If the same sample repeats too many times, it has **bad mixing**
- the key: to make M-H more efficient algorithm, we need to:

$$\min\left(1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right) \to 1$$

Hamiltonian Metropolis Hasting (HMC): very brief introduction

- Hamiltonian = Potential + Kinetic

$$H(q, p) = U(q) + K(p)$$

- let q be **position** and p be **momentum**
  1. $p \sim \exp(K(p))$
  2. $(q_i, p_i) \to (q_i^*, p_i^*)$ using Hamiltonian dynamics

$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = \frac{\partial H}{\partial p_i} \qquad \frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial H}{\partial q_i}$$

  Hamiltonian motion's reversibility property $\implies \tilde{q}(p, q|p^*, q^*) = \tilde{q}(p^*, q^*|p, q)$

- conservation of Energy:

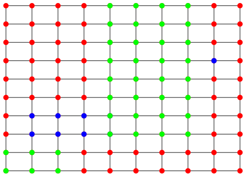
$$\begin{aligned} U(q) + K(p) &\approx U(q^*) + K(p^*) \\ \implies \Big(p(q, p) \propto \exp(U(q) + K(p))\Big) &\approx \Big(p(q^*, p^*) \propto \exp(U(q^*) + K(p^*))\Big) \\ \implies \min\left(1, \frac{\exp(U(q^*) + K(p^*))}{\exp(U(q) + K(p))} \frac{\cancel{\tilde{q}(p, q|p^*, q^*)}}{\cancel{\tilde{q}(p^*, q^*|p, q)}}\right) &\approx 1 \end{aligned}$$

- small rejection rate due to numerical errors in leap-frog

Swendesen-Wang

- Potts Model:

$$M(\Pi) \propto \exp\left(\sum_{i<j} \beta_{ij} \mathbf{1}_{z_i=z_j}\right)$$



- introduce auxiliary variables:

$$\Pr(r_{ij}=0|\Pi) = \exp(-\beta_{ij}\mathbf{1}_{z_i=z_j}) = q_{ij}$$
$$r_{ij} \sim \text{Bernoulli}\left(1-\exp(-\beta_{ij}\mathbf{1}_{z_i=z_j})\right)$$
$$\Pr(\mathbf{r}|\Pi) = \prod_{1\le i<j\le n} P(r_{ij}|\Pi)$$

  - Conventional Swendsen-Wang and adding data **y**: (see demo)

$$P(\Pi|\mathbf{r},\mathbf{y}) \propto p(\mathbf{y}|\Pi)P(\mathbf{r}|\Pi) = \prod_{j=1}^{k} p(\mathbf{y}_{A_j}) \prod_{1\le i<j\le n} \left[1-\exp(-\beta_{ij}\mathbf{1}_{z_i=z_j})\right]^{r_{ij}} \left[\exp(-\beta_{ij}\mathbf{1}_{z_i=z_j})\right]^{1-r_{ij}}$$

  - **key:** bond variables r_{ij} induce groups of sites which have same cluster label, a single $0^1 = 0$ makes the whole joint density become zero

Swendesen-Wang: our past and current work

- **our past work**
  combine S-W with Bayesian Non-Parametrics:

  Xu, R.Y.D. Caron, F., & Doucet, A (2016), Bayesian nonparametric image segmentation using a generalised Swendsen-Wang algorithm, arXiv:1602.03048

$$g(m_{-\ell,1},\dots,m_{-\ell,j}+|C_\ell|\dots,m_{-\ell,k_{-\ell}})\frac{p(\mathbf{y}_{C_\ell\cup A_{-\ell,j}})}{p(\mathbf{y}_{A_{-\ell,j}})}\prod_{\{(i,j)|i\in C_\ell,r_{ij}=0\}}\exp\left(\beta_{ij}(1-\delta_{ij})\mathbf{1}_{z_i=z_j}\right)$$

- **current exploratory work:**
  a neural network module to learn better mixing rate

Part 2.1: Bayesian framework to **explain** Deep Learning

2.1.1 **Explain** reason why infinite-width Neural Network is a Gaussian Process

Gaussian Process in one-minute

- $\mathcal{GP}$ is a (potentially infinite) collection of RVs, s.t., joint distribution of every finite subset of RVs is multivariate Gaussian:

$$f \sim \mathcal{GP}(\mu(x), \mathcal{K}(x, x')) \qquad \text{for any arbitary } x, x'$$

- **prior** defined over $p(f|\mathcal{X})$, instead of $p(x)$ over $\mathcal{X} \equiv \{x_1, \dots x_k\}$

$$p(f|\mathcal{X}) \equiv p\left(\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_k) \end{bmatrix}\right) = \mathcal{N}\Big(0, K\Big) = \mathcal{N}\left(0, \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_k) \\ \vdots & \ddots & \vdots \\ k(x_k, x_1) & \dots & k(x_k, x_k) \end{bmatrix}\right)$$

marginal and conditional marginal under **noisy output**

- in a regression setting:

$$y_i = f(x_i) + \epsilon_i \qquad \epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2)$$

- joint $[\mathcal{Y}, y^\star]^\top$, after integrate out f:

$$\begin{aligned} p\left(\begin{bmatrix}\mathcal{Y}\\ y^\star\end{bmatrix} \middle| \begin{bmatrix}\mathcal{X}\\ x^{\star\top}\end{bmatrix}, \sigma_\epsilon^2\right) &= \int p\left(\begin{bmatrix}\mathcal{Y}\\ y^\star\end{bmatrix} \middle| \begin{bmatrix}\mathcal{X}\\ x^{\star\top}\end{bmatrix}, f\right) p(f|\mathcal{X}, x^\star)\mathrm{d}f \\ &= \int \mathcal{N}\left(\begin{bmatrix}\mathcal{Y}\\ y^\star\end{bmatrix} \middle| \begin{bmatrix}f(\mathcal{X})\\ f(x^{\star\top})\end{bmatrix}, \sigma_\epsilon^2 I\right) p(f|\mathcal{X}, x^\star)\mathrm{d}f \\ &= \mathcal{N}\left(0, \begin{bmatrix} \underbrace{K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2 I}_{\Sigma_{1,1}} & \underbrace{K(\mathcal{X}, x^\star)}_{\Sigma_{1,2}} \\ \underbrace{K(x^\star, \mathcal{X})}_{\Sigma_{2,1}} & \underbrace{K(x^\star, x^\star) + \sigma_\epsilon^2}_{\Sigma_{2,2}} \end{bmatrix}\right) \end{aligned}$$

- **predictive distribution** of $y^\star|\mathcal{Y}$ using conditional formula of multivariate Gaussian:

$$\begin{aligned} & p\left(y^\star|\mathcal{Y}, \mathcal{X}, x^\star\right) \\ &= \mathcal{N}\Bigg(\underbrace{\mathbf{0}}_{\mu_2} + \underbrace{K(x^\star,\mathcal{X})}_{\Sigma_{2,1}} \underbrace{\left(K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2 I\right)^{-1}}_{\Sigma_{1,1}^{-1}} (\mathcal{Y} - \underbrace{0}_{\mu_1}), \\ & \qquad \underbrace{k(x^\star, x^\star) + \sigma_\epsilon^2}_{\Sigma_{2,2}} - \underbrace{K(x^\star,\mathcal{X})}_{\Sigma_{2,1}} \underbrace{\left(K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2 I\right)^{-1}}_{\Sigma_{1,1}^{-1}} \underbrace{K(\mathcal{X}, x^\star)}_{\Sigma_{1,2}}\Bigg) \end{aligned}$$

marginal and conditional marginal under **noiseless output**

- **posterior** of f given $\mathcal{Y}$ in regression:

$$p\left(\begin{bmatrix}\mathcal{Y}\\ f\end{bmatrix} \middle| \begin{bmatrix}\mathcal{X}\\ x\end{bmatrix}^\top\right) = p\left(\begin{bmatrix}f(\mathcal{X})\\ f(x)\end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix}K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2\mathbf{I} & K(\mathcal{X},x)\\ K(x,\mathcal{X}) & K(x,x)\end{bmatrix}\right)$$

for arbitrary variable x

conditional marginal of $y^\star|\mathcal{Y}$ using conditional formula of multivariate Gaussian:

$$p(f|\mathcal{X},\mathcal{Y}) = \mathcal{GP}\Big(K(x,\mathcal{X})(K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathcal{Y},$$
$$k(x,x') - K(x,\mathcal{X})\left(K(\mathcal{X},\mathcal{X}) + \sigma_\epsilon^2 I\right)^{-1} K(\mathcal{X},x')\Big)$$

- **deterministic function** $y_i = f(x_i)$ is used, e.g., neural network's read-out layer $f(x_i)$, data y_i
  $p([\mathcal{Y}, y^\star]^\top)$ no longer need to integrate f:

$$p\left(\begin{bmatrix}\mathcal{Y}\\ y^\star\end{bmatrix} \middle| \begin{bmatrix}\mathcal{X}\\ x^\star\end{bmatrix}^\top\right) = p\left(\begin{bmatrix}f(\mathcal{X})\\ f(x^\star)\end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix}K(\mathcal{X},\mathcal{X}) & K(\mathcal{X},x^\star)\\ K(x^\star,\mathcal{X}) & K(x^\star,x^\star)\end{bmatrix}\right)$$

predictive distribution $y^\star|\mathcal{Y}$ using conditional formula of multivariate Gaussian:

$$p\left(y^\star|\mathcal{Y},\mathcal{X},x^\star\right) = \mathcal{N}\Big(K(x^\star,\mathcal{X})K(\mathcal{X},\mathcal{X})^{-1}\mathcal{Y},$$
$$k(x^\star,x^\star) - K(x^\star,\mathcal{X})K(\mathcal{X},\mathcal{X})^{-1}K(\mathcal{X},x^\star)\Big)$$

$\mathcal{GP}$ and Neural Networks

- proven by
  "R. M. Neal. Bayesian Learning for Neural Networks. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996"
  and *"J. H. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. ICLR, 2018"*
- at initialization:

$$z_k^l(x) = b_k^l + \sum_{j=1}^{N_l} W_{k,j}^l \times \phi\left(z_j^{l-1}(x)\right) \qquad W_{k,j}^l \sim \mathcal{N}\left(0, \frac{1}{\sqrt{N_l}}\right) \quad b_k^l \sim \mathcal{N}(0, \sigma_b)$$

- interesting finding:

$$N_l \to \infty \implies z^l(\mathcal{X}) \sim \mathcal{GP}(0, \mathcal{K}^l(\mathcal{X}, \mathcal{X}))$$

with the recursive fomula: **NNGP**

$$K^l(x^{(p)}, x^{(q)}) = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{\left(z_j^{l-1}(x^{(p)}), z_j^{l-1}(x^{(q)})\right) \sim \mathcal{N}\left(0, K^{l-1}(x^{(p)}, x^{(q)})\right)} \left[\phi(z_j^{l-1}(x^{(p)}))\phi(z_j^{l-1}(x^{(q)}))\right]$$

$\mathcal{GP}$ and Neural Networks: our work

- **NNGP** was precursor towards **Neural Tangent Kernel**
  my tutorial "Infinite Width: Neural Networks as Gaussian Process and Neural Tangent Kernel (NTK)", `https://github.com/roboticcam/machine-learning-notes/blob/master/files/gp_nn.pdf`
- our work:
  Huang, W., ***Xu, R. Y. D****, Du, W., Zeng Y., and Zhao Y., (2020) Mean field theory for deep dropout networks: digging up gradient backpropagation deeply, the 24th European Conference on Artificial Intelligence (ECAI 2020)*

  Huang, W., Du, W., ***Xu, R. Y. D****, (2020) On the Neural Tangent Kernel of Deep Networks with Orthogonal Initialization, arXiv preprint arXiv:2004.05867*

Part 2.2: Bayesian framework to **assist** Deep Learning

2.2.1 Bayesian assisted **Generative Adversarial Network**

Generative Adversarial Networks

- GAN objective:

$$\min_G \max_D \left(\mathcal{L}(D, G) \equiv \mathbb{E}_{\mathbf{x} \sim p_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \right)$$
$$= \mathbb{E}_{\mathbf{x} \sim p_r(\mathbf{x})}[\log D(\mathbf{x})] \underbrace{+ \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(\mathbf{x})]}_{\text{alternative expression}}$$

- **my tutorial** on "Mathematics on Generative Adversarial Networks": https://github.com/roboticcam/machine-learning-notes/blob/master/files/GAN.pdf

Bayesian GAN

- **Generator** $p(\theta_g|\mathbf{z}, \theta_d) \propto \left(\prod_{i=1}^{n_g} D_{\theta_d}\left(G_{\theta_g}(\mathbf{z}^{(i)})\right)\right) p(\theta_g|\alpha_g)$
- **Discriminator** $p(\theta_d|\mathbf{z}, \mathbf{X}, \theta_g) \propto \prod_{i=1}^{n_d} D_{\theta_d}(\mathbf{x}^{(i)}) \times \prod_{i=1}^{n_g} \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}^{(i)})\right) \times p(\theta_d|\alpha_d)$
- **marginalization** $p(\theta_g|\theta_d)$

$$p(\theta_g|\theta_d) = \int p(\theta_g, \mathbf{z}|\theta_d) d\mathbf{z} = \int p(\theta_g|\mathbf{z}, \theta_d) \underbrace{p(\mathbf{z}|\theta_d)}_{\text{independent of } \theta_d} d\mathbf{z}$$

$$= \int p(\theta_g|\mathbf{z}, \theta_d) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{N}\sum_{i=1}^{N} p(\theta_g|\mathbf{z}^{(i)}, \theta_d) \qquad \mathbf{z}^{(i)} \sim p(\mathbf{z})$$

- **marginalization** $p(\theta_d|\theta_g)$

$$p(\theta_d|\theta_g) \equiv p(\theta_d|\mathbf{X}, \theta_g) = \int_{\mathbf{z}} p(\theta_d, \mathbf{z}|\mathbf{X}, \theta_g) d\mathbf{z} = \int p(\theta_d|\mathbf{z}, \mathbf{X}, \theta_g) \underbrace{p(\mathbf{z}|\mathbf{X}, \theta_g)} d\mathbf{z}$$

$$= \int_{\mathbf{z}} p(\theta_d|\mathbf{z}, \mathbf{X}, \theta_g) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{N}\sum_{i=1}^{N} p(\theta_g|\mathbf{z}^{(i)}, \mathbf{X}, \theta_g) \quad \mathbf{z}^{(i)} \sim p(\mathbf{z})$$

Saatchi, Y. and Wilson, A. G. Bayesian gan. In Advances in Neural Information Processing Systems, pp. 3625–3634, 2017

Our focus on Bayesian assisted GAN

- look at GAN again:

$$\min_G \max_D \left(\mathcal{L}(D, G) \equiv \mathbb{E}_{\mathbf{x} \sim p_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{z \sim \underbrace{p_z(z)}_{\text{simple distribution}}} [\log(1 - D(G(z)))] \right)$$

- researchers tend to use simple $p_z(z) \sim \mathcal{N}(0, \mathbf{I})$, but very complex G
- **question** is can we instead use more complex $p(z)$ to:
  - recover underlying data distribution $p_z(z)$
  - relief G, get $p_z(z)$ to do more work!

Our current and on-going work on sophisticated $p_z(z)$ in GAN

1. **work 1** replaced $p(z)$ in terms of Mixture densities:

   Huang, W., **Xu, R Y D***, Jiang, S., Liang, X.; Oppermann, I., GAN-Based Gaussian Mixture Model Responsibility Learning, accepted to International Conference on Pattern Recognition 2021*
2. **work 2** recent work replacing $p(z)$ with **Dirichlet Process** (under review)

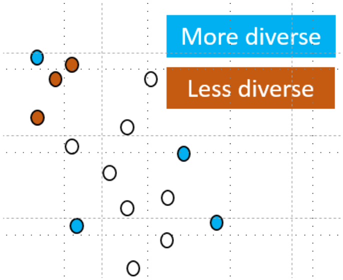
   Huang, W., **Xu, R Y D***, Jiang, S., Liang, X.; Oppermann, I., Dirichlet Process Mixture Model Learning via GAN*
3. **work 3** exploratory work
   use Hamiltonian Monte Carlo to sample complex $p_z(z)$

Part 2.2: Bayesian framework to **assist** Deep Learning

2.2.1 Determinantal Point Process (DPP) assisted Deep Learning

What is DPP?

- 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

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

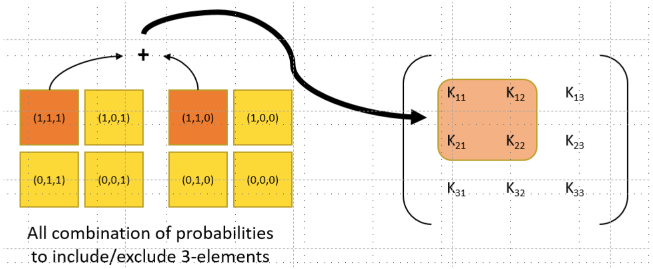
- An example: given $\mathbf{Y} = \{1, 2, 3, 4, 5\}$, $A = \{1, 2, 3\}$

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

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

- **my tutorial** on "Determinantal Point Process": https://github.com/roboticcam/machine-learning-notes/blob/master/files/dpp.pdf

let's use a diagram!



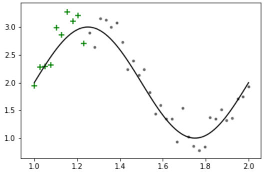
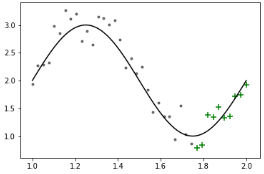
+
(1,1,1)
(1,0,1)
(1,1,0)
(1,0,0)
(0,1,1)
(0,0,1)
(0,1,0)
(0,0,0)
All combination of probabilities to include/exclude 3-elements
K_{11} K_{12} K_{13}
K_{21} K_{22} K_{23}
K_{31} K_{32} K_{33}

- sum of probabilities to include 1st and 2nd elements
- This is defined by $\det(K_{\{1,2\}})$
- our past work on DPP:
  - *Qiao, M.,* **Xu, R.Y.D.**, *Bian, W. & Tao, D. (2016), 'Fast sampling for time-varying Determinantal Point Processes', vol. 11, no. 1. ACM Transactions on Knowledge Discovery from Data*
  - *Qiao, M., Bian, W.,* **Xu, R.Y.D.** *& Tao, D. (2015), 'Diversified Hidden Markov Models for Sequential Labeling', vol. 27, no. 11, pp. 2947-2960. IEEE Transactions on Knowledge and Data Engineering*

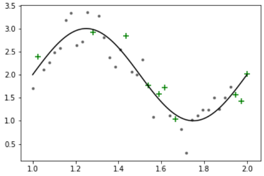
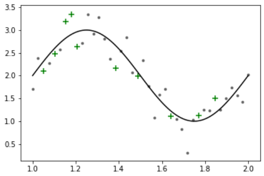
In Deep Learning: our work in Mini-batch selection via DPP

Huang, W., Xu, R.Y.D., I. Oppermann (2019), Efficient Diversified Mini-Batch Selection using Variable High-layer Features, Asian Conference on Machine Learning, (ACML 2019)

- Uniform sampling: less-diverse samples in one batch (lead to high variance in learning)

- ideally, each sample is a **good representation** of the entire data-set, higher diversities between selected samples (lead to low variance in learning)

Other example of Deep Learning assisted by DPP

- *Z. Mariet and S. Sra. Diversity networks: Neural network compression using Determinantal Point Processes. In International Conference on Learning Representations, 2016.*
- *Zelda Mariet, Yaniv Ovadia, and Jasper Snoek. Dppnet: Approximating determinantal point processes with deep networks. arXiv preprint arXiv:1901.02051, 2019*

Part 2.2: Bayesian framework to **assist** Deep Learning

2.2.3 probability distribution re-parameterization

Why Re-parameterization: **(1) otherwise infeasible**

- a **computation graph** of $L = f_3\Big(f_2\big(f_1(\theta)\big)\Big)$, chain rule gives:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial f_3} \times \frac{\partial f_3}{\partial f_2} \times \frac{\partial f_2(\theta)}{\partial f_1(\theta)} \times \frac{\partial f_1}{\partial \theta} \qquad f_2 = f_1(\theta)$$

- when f_2 is no longer deterministic:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial f_3} \times \frac{\partial f_3}{\partial f_2} \times \underbrace{\frac{\partial f_2(\theta)}{\partial f_1(\theta)}}_{\text{doesn't make sense!}} \times \frac{\partial f_1}{\partial \theta} \qquad f_2 \sim f_1(\theta)$$

- solution, **re-parameterization trick**, s.t., random variable $\epsilon \sim q$ drawn from a distribution that is θ-free!

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial f_3} \times \frac{\partial f_3}{\partial f_2} \times \underbrace{\frac{\partial f_2(\theta)}{\partial f_1(\theta)}}_{\text{OK again!}} \times \frac{\partial f_1}{\partial \theta} \qquad f_2 = g(f_1(\theta), \epsilon) \quad \epsilon \sim q(\epsilon)$$

Simple Re-parameterization trick example

example: instead of:

$$z \sim \mathcal{N}(z; \mu(\theta), \sigma)$$

instead it can be re-parameterised into as a function of a standard Gaussian variable:

$$z = g(\boldsymbol{\epsilon}, \theta) = \underbrace{\mu(\theta) + \boldsymbol{\epsilon}\sigma}_{g(\boldsymbol{\epsilon},\theta)} \qquad \boldsymbol{\epsilon} \sim \underbrace{\mathcal{N}(0, 1)}_{p(\boldsymbol{\epsilon})}$$

Other re-parameterizations available?

- many available!

name	$p(z;\theta)$	$p(\epsilon)$	$g(\epsilon,\theta)$
Exponential	$\exp(-x); x > 0$	$\epsilon \sim [0;1]$	$\ln(1/\epsilon)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$\epsilon \sim [0;1]$	$\tan(\pi\epsilon)$
Laplace	$\mathcal{L}(0;1) = \exp(-\lvert x\rvert)$	$\epsilon \sim [0;1]$	$\ln(\frac{\epsilon_1}{\epsilon_2})$
Laplace	$\mathcal{L}(\mu;b)$	$\epsilon \sim [0;1]$	$\mu - b\mathrm{sgn}(\epsilon)\ln(1-2\lvert\epsilon\rvert)$
Gaussian	$\mathcal{N}(0;1)$	$\epsilon \sim [0;1]$	$\sqrt{\ln(\frac{1}{\epsilon_1})}\cos(2\pi\epsilon_2)$
Gaussian	$\mathcal{N}(\mu;RR^\top)$	$\epsilon \sim \mathcal{N}(0;1)$	$\mu + R\epsilon$
Rademacher	$Rad(\frac{1}{2})$	$\epsilon \sim \mathrm{Bern}(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln\mathcal{N}(\mu;\sigma)$	$\epsilon \sim \mathcal{N}(\mu;\sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$\mathcal{IG}(Ik;\theta)$	$\epsilon \sim \mathcal{G}(k;\theta^{-1})$	$\frac{1}{\epsilon}$

- however, today we are interested only in **Softmax distribution** parameterizations!

Re-parameterization for variance reduction

- Re-parameterization is also used for variance reduction, discussing REBAR and RELAX algorithms:
- **my tutorial** on "Control Variate": `https://github.com/roboticcam/machine-learning-notes/blob/master/files/variance_reduction.pdf`
- will not discuss in this talk

re-parameterization to Softmax

- in deep learning, we need to perform:

$$k \sim \text{softmax}(\mu_1(\theta), \dots, \mu_L(\theta))$$

- μ_i defines **applications**:
  - $\mu_i \equiv \mathbf{x}^\top \theta_i$ in classification
  - $\mu_i \equiv \mathbf{u}_i^\top \mathbf{v}_c$ for word vectors
- but softmax function contains θ!, yes, in the chain rule path!

Re-parameterization using Gumbel-max trick

- instead of sample $k \sim \text{softmax}(\mu_1(\theta), \dots, \mu_K(\theta))$, we i.i.d. sample ga instead
- we can perform **Gumbel-max** trick:

$$
\begin{aligned}
z_i &\sim \text{ga}(z; 0, 1) \\
k &= \underset{i \in \{1,\dots,K\}}{\arg\max} \left\{ z_1 + \mu_1(\theta), \dots, z_K + \mu_K(\theta) \right\}
\end{aligned}
$$

- well, there is two problems, firstly **why are the two are equivalent**?

Gumbel distribution definitions

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

$$p(Z = z|\mu, 1) \equiv \text{ga}(z\,;\,\mu) = \exp\left[-(z-\mu) - \exp(-(z-\mu)) \right]$$

- CDF of Gumbel:

$$\Pr(Z \leq z|\mu, 1) \equiv \mathcal{G}(z\,;\,\mu) = \exp\left[-\exp(-(z-\mu)) \right]$$

- it is obvious that:

$$\text{ga}(z|\mu, 1) = \exp(-z+\mu)\mathcal{G}(z|\mu)$$

  which is a property you must know to work with Gumbels!
- some literature write location as $\log(\phi)$ instead of μ

$$\log(\phi) = \mu \qquad \Longrightarrow \qquad \phi = \exp(\mu)$$

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

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

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 summary

- moral of the story is, instead of sample from **softmax**

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

- one can instead perform:

$$k = \underset{i \in \{1,\dots,K\}}{\arg\max} \{z_1, \dots, z_K\}$$

$$\text{where } z_i \sim \text{ga}(z\,;\,\mu_i) \equiv \exp\Big[-(z-\mu_i) - \exp\{-(z-\mu_i)\}\Big]$$

  problem as μ still in Gumbel PDF, i.e., not "parameter-less" distribution
- using Gumbel's property of location:

$$k = \underset{i \in \{1,\dots,K\}}{\arg\max} \{\mu_1 + z_1, \dots, \mu_K + z_K\}$$

$$\text{where } z_i \overset{\text{iid}}{\sim} \text{ga}(z\,;\,0) \equiv \exp\Big[-(z) - \exp\{-(z)\}\Big]$$

how to sample a Gumbel?

- CDF of a Gumbel, easy to take inverse, just keep “log”:

$$
\begin{aligned}
u &= \exp^{-\exp^{-(z-\mu)/\beta}} \\
\implies \log(u) &= -\exp^{-(z-\mu)/\beta} \\
\implies \log(-\log(u)) &= -(z-\mu)/\beta \\
\implies -\beta\log(-\log(u)) &= z-\mu \\
\implies z &= \mathcal{G}^{-1}(u) \equiv \mu - \beta\log(-\log(u))
\end{aligned}
$$

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

$$z = \mathcal{G}^{-1}(u) \equiv -\log(-\log(u))$$

- therefore, sampling strategy:

$$
\begin{aligned}
u_i &\sim \mathcal{U}(0,1) \\
z_i &= -\log(-\log(u_i)) \\
k &= \underset{i\in\{1,\dots,K\}}{\arg\max}\ \{\mu_1 + z_1, \dots, \mu_L + z_K\}
\end{aligned}
$$

Second problem with Softmax re-parameterisation

- look at re-parameterization:

$$
\begin{aligned}
z_i &\sim \text{ga}(z; 0, 1) \\
k &= \operatorname*{arg\,max}_{i \in \{1,\dots,K\}} \{z_1 + \mu_1(\theta), \dots, z_K + \mu_K(\theta)\}
\end{aligned}
$$

- the other remaining **problem**: sample k 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

- use **softmax map** instead to propagate the gradient!

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

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

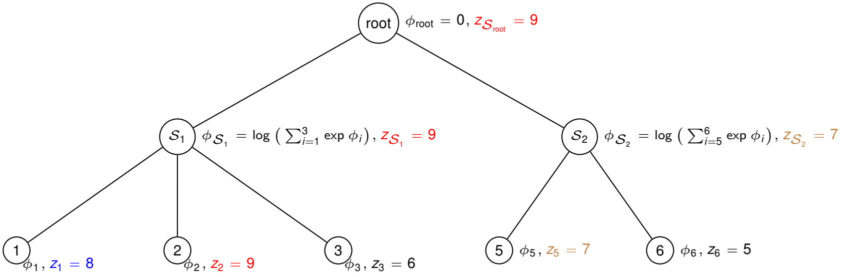
- questions can you also think about the relationship between Gaussian Mixture Model and K-means?
- one can say $\tau = 1$ is softmax, and $\tau = 0$ is hard-max!
- then we can apply the same softmax map with added Gumbel variables:

$$z_i \sim \text{ga}(0)$$

$$f_\tau(\boldsymbol{\mu} + \mathbf{z})_k = \frac{\exp(\mu_k + z_k)/\tau)}{\sum_{i=1}^{K}\exp(\mu_i + z_i)/\tau)}$$

- choose $\tau > 0$ for training, and $\tau = 0$ for inference

Sample Discrete Sequential use **Stochastic Beam Search**


root
$\phi_{\text{root}} = 0,\ z_{\mathcal{S}_{\text{root}}} = 9$
$\mathcal{S}_1$
$\phi_{\mathcal{S}_1} = \log\left(\sum_{i=1}^{3} \exp \phi_i\right),\ z_{\mathcal{S}_1} = 9$
$\mathcal{S}_2$
$\phi_{\mathcal{S}_2} = \log\left(\sum_{i=5}^{6} \exp \phi_i\right),\ z_{\mathcal{S}_2} = 7$
1
$\phi_1,\ z_1 = 8$
2
$\phi_2,\ z_2 = 9$
3
$\phi_3,\ z_3 = 6$
5
$\phi_5,\ z_5 = 7$
6
$\phi_6,\ z_6 = 5$

$$\text{softmax}(\theta) \equiv \left(\underbrace{\frac{\exp\left(\mu_\theta(y_1|Y_{\mathcal{S}_1})\right)}{\sum_{y'} \exp\left(\mu_\theta(y'_t|Y_{\mathcal{S}_1})\right)}}_{\exp(\phi_1)}, \underbrace{\frac{\exp\left(\mu_\theta(y_2|Y_{\mathcal{S}_1})\right)}{\sum_{y'} \exp\left(\mu_\theta(y'|Y_{\mathcal{S}_1})\right)}}_{\exp(\phi_2)}, \underbrace{\frac{\exp\left(\mu_\theta(y_3|Y_{\mathcal{S}_1})\right)}{\sum_{y'} \exp\left(\mu_\theta(y'|Y_{\mathcal{S}_1})\right)}}_{\exp(\phi_3)} \right)$$

$$\phi_i = \log\left(p(y_i|Y_{\mathcal{S}_1})\right) \equiv \mu_\theta(y_i|Y_{\mathcal{S}_1}) - \log\left(\sum_{y'} \exp\left(\mu_\theta(y'|Y_{\mathcal{S}_1})\right)\right)$$

Sample Discrete Sequential use **Stochastic Beam Search**

fact 1: $$p_\theta(Y_{\mathcal{S}}) = \underbrace{\exp(\phi_{\mathcal{S}})}_{\text{prob of node } \mathcal{S}} = \sum_{i \in S} \underbrace{\exp(\phi_i)}_{\text{prob of child}} \implies {\color{blue}\phi_{\mathcal{S}}} = \log p_\theta(Y_{\mathcal{S}}) = \log \sum_{i \in \mathcal{S}} \exp \phi_i$$

fact 2: let $z_i \sim \text{ga}(z; \phi_i)$

max value: $$\max_{i \in B}\{z_i\} \sim \text{ga}\left(\log \sum_{j \in B} \exp \phi_j\right)$$

max index: $$\arg\max_{i \in B}\{z_i\} \sim \text{Categorical}\left(\frac{\exp(\phi_i)}{\sum_{j \in B} \exp(\phi_j)}, i \in B\right)$$

these two operations are independent, when combine the two, we have:

$$z_{\mathcal{S}} = \max_{i \in \mathcal{S}}\{z_i\} \sim \text{ga}\underbrace{\left(\log \sum_{i \in S} \exp \phi_i\right)}_{{\color{blue}\phi_{\mathcal{S}}}} = \text{ga}\left({\color{blue}\phi_{\mathcal{S}}}\right)$$

fact 3: $$p\Big(\text{sample from softmax “without replacement” } m \text{ times}\Big) = p\Big(\text{choosing } m \text{ largest values from } \{z_i\}\Big)$$

Top-down Stochastic Beams search

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. arXiv preprint arXiv:1903.06059, 2019

- Given a partial tree $Y^{\mathcal{S}}$ (we know its value of $\phi_{\mathcal{S}}$ and $z_{\mathcal{S}}$):
  - For each $i \in \text{children}(S)$:

$$\phi_i \leftarrow \phi_{\mathcal{S}} + \log p_\theta(Y^i | Y^{\mathcal{S}}) : \qquad \text{i.e., } \exp(\phi_i) = \exp(\phi_{\mathcal{S}}) p_\theta(Y^i | Y^{\mathcal{S}})$$

S' extends partial tree length S by one token

$$z_i \sim \text{ga}(\phi_i)$$

  - $Z = \max\{z_i\}$
  - For each $i \in \text{children}(S)$:

$$\widetilde{z}_i \leftarrow -\log\big(\exp(-z_{\mathcal{S}}) - \exp(-Z) + \exp(-z_i)\big)$$

- BEAM $\leftarrow$ take top k of expansion according to $\{\widetilde{z}_i\}$ then expand to add $(Y^i, \phi_i, \widetilde{z}_i)$

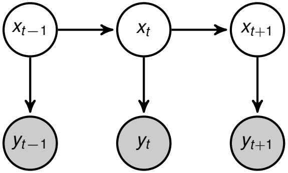
understand $\widetilde{z}_i \leftarrow -\log\big(\exp(-z_{\mathcal{S}}) - \exp(-Z) + \exp(-z_i)\big)$ read original paper, or:
my tutorial on "Some New Research in Softmax":
https://github.com/roboticcam/machine-learning-notes/blob/master/files/softmax.pdf

Part 2.3: Bayesian framework to **be assisted by** Deep Learning

2.3.1 Long Short Term Memory to assist Kalman Filter's parameter training

State Space Models

- 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?

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

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

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

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

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

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

$$\Longrightarrow \textbf{Measurement probability:} \qquad p(y_t|\mathbf{x}_t) \sim \mathcal{N}(H\mathbf{x}_t, R_t)$$

- 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 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!
- *H. Coskun, F. Achilles, R. DiPietro, N. Navab and F. Tombari, "Long short-term memory kalman filters: Recurrent neural estimators for pose regularization", Proc. IEEE Int. Conf. Comput. Vis., pp. 5525-5533, 2017*

LSTM

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

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

$$C_t = f_t \odot C_{t-1} + i \odot \tilde{C}_t$$

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

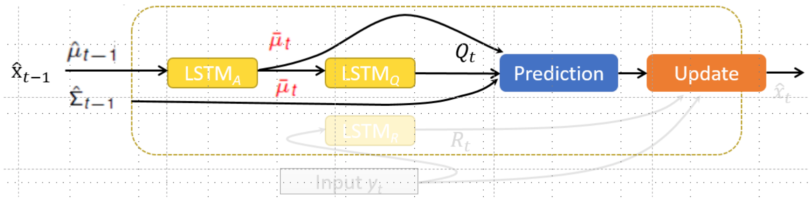
- in LSTMs, cell state C_t. The derivative of consecutive states is of the form:

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

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

- 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



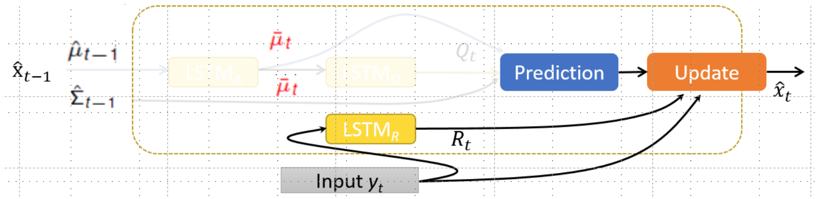
- the state-space model is changed into:

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

- **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:

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

- **update**:

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

Part 3: In Conclusion

- Bayesian inference as a **research topic** will continue to play a key role in machine learning and many applicable applications.
- Bayesian inference as a **tool** will be able to enhance the functions of many machine learning models.

Thank you!