Improved Variational Inference

Abstract

- p(z|x)를 inference하는 normalizing flow의 일종
- 기존 NF와 다르게 latent의 high-dimension으로 scale up이 가능
- invertible transformation + autoregressive neural net

Introduction

- normalizing flow을 이용해 학습
- Gaussian autorefressive function을 이용
  - input : multidimensional tensor
  - output : 각 element에 해당되는 gaussian의 mean, var
- 참고
  - Inference : x → z
  - generative : z → x

Variational Inference and Learning

$logp(X) = \sum logp(x^{(i)})$

$logp(x) \geq E_{q(z|x)}[logp(x,z) - logq(z|x)] = L(x;\theta)$

- lower bound을 maximumize하면, p(x)을 증가시키고, q(z|x)가 p(z|x)와 같아짐
- q(z|x) 은 reparameterization 해서 표현
- q(z|x)은 factorize 가능
  - q(z1, z2|x) = q(z1|x)q(z2|z1, x)
- 요구 사항
  - q(z|x)가 계산이 가능하고, 미분이 가능
  - sampling이 가능

  → q(z|x)로 diagonal posterial 가능, q(z|x) = N(u(x), sigma(x))

Normalizing Flow

- svi을 이용해 posterior distribution을 계산

$z_0 \sim q(z_0|x), z_t = f_t(z_{t-1}, x)$

$logq(z_T|x) = logq(z_0|x) - \sum logdet|\frac{dz_t}{dz_{t-1}}|$

Inverse Autoregressive Transformation

- gaussian autoregressive autoencoders 사용
- sacle, shift을 이용해 새로운 y을 구함 (직접 계산이 불가능)
    - i≤j 일 때, u_i, sigma_i의 yj로 미분할 때, 0가 계산

$y_i = u_i(y_{1:i-1}) + \sigma(y_{1:i-1})\epsilon$

- inverse : inverse transformation (x→z)

$\epsilon = (y - u(y))/\sigma(y)$

- 단점
    - sampling시에 sequential함이 필요
- 장점
    - inverse transformation일 때, parallelize 사용
        - 학습이 쉬움
    - simple determinant
        - j > i 일 경우, ei의 determinant가 0임.
        - j = i 일 경우, 1

$logdet|\frac{d\epsilon}{dy}| = \sum -log\sigma_i(y)$

Inverse Autoregressive Flow(IAF)

- 알고리즘

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

Data:
- $\mathbf{x}$: a datapoint, and optionally other conditioning information
- $\boldsymbol{\theta}$: neural network parameters
- $\texttt{EncoderNN}(\mathbf{x};\boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
- $\texttt{AutoregressiveNN}[*](\mathbf{z},\mathbf{h};\boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
- $\texttt{sum}(.)$: sum over vector elements
- $\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:
- $\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
- l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample '$\mathbf{z}$'

$[\boldsymbol{\mu},\boldsymbol{\sigma},\mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x};\boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0,I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m},\mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z},\mathbf{h};\boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1-\boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

- q(z|x)을 최대화한다.

$$logq(z_T|x) = logq(z_0|x) - \sum logdet|\frac{dz_t}{dz_{t-1}}|$$

$$logp(x) = logp_z(f^{-1}(x)) - \sum logdet|\frac{df_m^{-1}}{dz_{m-1}}|$$

- 방법

> - 초기 encoder을 이용해, u, sigma, h을 추출
> - z을 계산
>
> $$z_0 = \sigma_0 * \epsilon + \mu_0$$
>
> - likelihood 계산
>
> $$-(\frac{1}{2}\epsilon^2 + \frac{1}{2}log(2\pi)) = log(\frac{1}{\sqrt{2\pi}}exp(-\frac{1}{2}\epsilon^2)) = q(z_0|x)$$
>
> - z와 h을 입력으로 받는 autoregressiveNN (LSTM)적용해 m, s 적용
>   - sigma는 sigmoid을 이용해서 계산
>
> $$z = \sigma * z + (1-\sigma) * m$$
>
> - likelihood 계산
>   - dzt/dzt-1은 sigma이기 때문에
>
> $$L = L - \sum log\sigma_t$$

- Autoregressive 팁
  - 다양한 non-linear transformation이 가능
    - maked autoregressive model

        - convolution autoregressive AE

Conclusion

- 새로운 타입의 normalizaing flow 방법
- 빠르게 샘플링이 가능