Normalizing Flow Models (Part 2)

Hao Dong

Peking University
So far

- Learning via **maximum likelihood** over the dataset $D$

$$\max_{\theta} \log p(D; \theta) = \sum_{x \in D} \log \pi (G^{-1}_\theta(x)) + \log \left| \det \left( \frac{\partial G^{-1}_\theta(x)}{\partial x} \right) \right|$$

- What we need?
  - prior $z \sim \pi(z)$ easy to sample
  - Invertible transformations
  - Determinants of Jacobian Efficient to compute
Reference slides

- Hung-yi Li. Flow-based Generative Model
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow
• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
• Coupling layer based normalizing flow models
  • Coupling layer
    • NICE
    • Real NVP
    • Glow
  • Autoregressive models as flow models
    • MAF
    • IAF
    • Parallel Wavenet
Coupling Layer

\[ i \leq d \quad \begin{cases} z_1 \\ \vdots \\ z_d \end{cases} \]

\[ F \]

\[ \beta_{d+1} \\ \vdots \\ \gamma_d \]

\[ H \]

\[ x_1 \\ \vdots \\ x_d \]

\[ x_{i \leq d} = z_{i \leq d} \]

\[ x_{i > d} = \beta z_{i > d} + \gamma \]

No matter how complicated it is

\[ x_{d+1} \\ \vdots \\ x_D \]

\[ \text{elementwise multiplication} \quad \text{elementwise add} \]
Coupling Layer

\[
\begin{align*}
\text{if } i \leq d & \quad z_i = x_i \\
\text{if } i > d & \quad z_i = \frac{x_i - \gamma}{\beta}
\end{align*}
\]

NICE

Real NVP
Learning via maximum likelihood over the dataset $\mathcal{D}$

$$\max_{\theta} \log p(D; \theta) = \sum_{x \in \mathcal{D}} \log \pi \left( G_{\theta}^{-1}(x) \right) + \log \left| \det \left( \frac{\partial G_{\theta}^{-1}(x)}{\partial x} \right) \right|$$

Coupling Layer

- **Jacobian**
Coupling Layer

\[ J_G \]

\[
\begin{array}{cccc}
 z_1 & \ldots & z_d & \ldots & z_D \\
 I & \text{(Identity)} & O & \text{(zero)} & \\
 x_1 & \ldots & x_d & \ldots & x_D \\
 & x_{d+1} & \ldots & x_D & \\
 & x_D & \ldots & x_D & \\
\end{array}
\]

I don’t care. Diagonal

\[
\text{det}(J_G) = \frac{\partial x_{d+1}}{\partial z_{d+1}} \frac{\partial x_{d+2}}{\partial z_{d+2}} \cdots \frac{\partial x_D}{\partial z_D}
\]

\[
= \beta_{d+1} \beta_{d+2} \cdots \beta_D
\]

\[
x_{i>d} = \beta z_{i>d} + \gamma
\]
Coupling Layer

• We can use coupling layer to design invertible function and calculate the determinant of Jacobian efficiently!
Coupling Layer - Stacking
Coupling Layer

\[
\begin{align*}
&F_1 & H_1 \\
&F_2 & H_2 \\
&F_3 & H_3
\end{align*}
\]
- **Coupling layer based normalizing flow models**
  - Coupling layer
  - NICE
  - Real NVP
  - Glow
- **Autoregressive models as flow models**
  - MAF
  - IAF
  - Parallel Wavenet
NICE: Nonlinear Independent Components Estimation

- **Additive** coupling layers
  - Partition the variables $z$ into two disjoint subsets
  - $x_{1:d} = z_{1:d}$
  - $x_{d+1:n} = z_{d+1:n} + H(z_{1:d})$
  - **Volume preserving transformation** since determinant is 1.

- Additive coupling layers are composed together (with arbitrary partitions of variables in each layer)

- Final layer of NICE applies a rescaling transformation

Dinh et al., 2014. Nonlinear Independent Components Estimation
NICE - Rescaling layers

- **Rescaling layers**
  - **Forward:**
    - \( x_i = s_i z_i \), where \( s_i > 0 \) is the scaling factor for the i-th dimension.
  - **Inverse:**
    - \( z_i = x_i / s_i \)
  - **Jacobian:**
    - \( J = \text{diag}(s) \)
Samples generated via NICE

(a) Model trained on MNIST

(b) Model trained on TFD
Samples generated via NICE

(c) Model trained on SVHN

(d) Model trained on CIFAR-10
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow
• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
Real NVP

• Coupling layers
  • Partition the variables $z$ into two disjoint subsets
  • $x_{1:d} = z_{1:d}$
  • $x_{d+1:n} = z_{d+1:n} \odot F(z_{1:d}) + H(z_{1:d})$
  • **Non-volume preserving transformation** in general since determinant can be less than or greater than 1

• Coupling layers are composed together (with arbitrary partitions of variables in each layer)

Dinh et al., 2017. Density estimation using Real NVP
Samples generated via Real-NVP
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow
• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
Glow: Generative Flow with Invertible 1×1 Convolutions

Kingma et al. Glow: Generative Flow with Invertible 1x1 Convolutions
1x1 Convolution

If $W$ is invertible, it is easy to compute $W^{-1}$.

$W$ can shuffle the channels.

$W = \begin{pmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
1x1 Convolution

\[ x = f(z) = Wz \]

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 
\end{bmatrix} =
\begin{bmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33}
\end{bmatrix}
\begin{bmatrix}
  z_1 \\
  z_2 \\
  z_3 
\end{bmatrix}
\]
1x1 Convolution

\[(\text{det}(W))^{d\times d}\]

If \(W\) is 3x3, computing \(\text{det}(W)\) is easy.

d\times d positions (pixels)
Image results: Glow

Figure 5: Linear interpolation in latent space between real images

Coupling Layer

Invertible 1x1 conv

Coupling Layer
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow
• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
Autoregressive models as flow models

• Consider a Gaussian autoregressive model:
  • \( p(x) = \prod_{i=1}^{n} p(x_i | x_{<i}) \)
  • Such that \( p(x_i | x_{<i}) = N(\mu_i(x_1, ..., x_{i-1}), \exp(\alpha_i(x_1, ..., x_{i-1}))^2) \), \( \mu_i, \alpha_i \) are neural networks.

• Sampler for this model:
  • Sample \( z_i \sim N(0,1) \)
  • Let \( x_i = \exp(\alpha_i) z_i + \mu_i \) \( \leftarrow \) look like coupling layer ~~

• **Flow interpretation:** transform \( z \) to \( x \) via invertible transformation (parameterized by \( \mu_i, \alpha_i \))
Masked Autoregressive Flow (MAF)

- **Forward: (z to x)**
  - $x_i = z_i \exp(\alpha_i) + \mu_i \; \forall \; i \in \{1 \ldots n\}$
  - Then calculate $\alpha_{i+1}, \mu_{i+1}$
- Sampling is sequential and slow (like autoregressive)

Figure adapted from Eric Jang’s blog

Papamakarios et al. Masked Autoregressive Flow for Density Estimation
Masked Autoregressive Flow (MAF)

- Inverse \((x \text{ to } z)\)
  - \(z_i = (x_i - \mu_i) \exp(-\alpha_i)\)
  - can be done in parallel.

- Jacobian is lower diagonal, hence determinant can be computed efficiently

- Likelihood evaluation is easy and parallelizable

Figure adapted from Eric Jang’s blog
\[
\max_\theta \log p(D; \theta) = \sum_{x \in D} \log \pi \left( G^{-1}_\theta(x) \right) + \log \left| \det \left( \frac{\partial G^{-1}_\theta(x)}{\partial x} \right) \right|
\]

- MAF can calculate \( G^{-1}_\theta(x) \) parallel.

- MAF: Fast likelihood evaluation (parallel), slow sampling (sequential)
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow
• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
Inverse Autoregressive Flow (IAF)

- **Forward: (z to x)**
  - \( x_i = z_i \exp(\alpha_i) + \mu_i \) \( \forall i \in \{1 \ldots n\} \)
  - parallel

- **Inverse (x to z)**
  - \( z_i = (x_i - \mu_i) \exp(-\alpha_i) \)
  - Then compute \( \mu_i, \alpha_i \)
  - sequential

Figure adapted from Eric Jang’s blog

Kingma et al. Improving Variational Inference with Inverse Autoregressive Flow
Inverse Autoregressive Flow (IAF)

• Fast to sample (parallel)

• Slow to evaluate likelihoods of data points during training (sequential)

• Fast to evaluate likelihoods of a generated point (we only need to cache $z_1, z_2, \ldots, z_n$)
IAF is inverse of MAF

**Figure:** Inverse pass of MAF (left) vs. Forward pass of IAF (right)
IAF vs. MAF

• Computational tradeoffs
  • MAF: Fast likelihood evaluation, slow sampling
  • IAF: Fast sampling, slow likelihood evaluation
• MAF more suited for training based on MLE, density estimation
• IAF more suited for real-time generation
• Can we get the best of both worlds?
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE
  • Real NVP
  • Glow

• Autoregressive models as flow models
  • MAF
  • IAF
  • Parallel Wavenet
Parallel Wavenet

MAF: $x \mapsto z$ parallel
IAF: $z \mapsto x$ parallel

• Two part training with a teacher (MAF) and student model (IAF)
• Teacher can be efficiently trained via MLE.
• Once teacher is trained, initialize a student model parameterized by IAF. Student model cannot efficiently evaluate density for external data points but allows for efficient sampling
• **Key observation:** IAF can also efficiently evaluate densities of its own generations (via caching the noise variates $z_1, z_2, \ldots, z_n$)

Parallel Wavenet

MAF: $x \mapsto z$ parallel
IAF: $z \mapsto x$ parallel

Figure 2: Overview of Probability Density Distillation. A pre-trained WaveNet teacher is used to score the samples $x$ output by the student. The student is trained to minimise the KL-divergence between its distribution and that of the teacher by maximising the log-likelihood of its samples under the teacher and maximising its own entropy at the same time.

Parallel Wavenet

MAF: $x \mapsto z$ parallel
IAF: $z \mapsto x$ parallel

- **Probability density distillation**: Student distribution is trained to minimize the KL divergence between student ($s$) and teacher ($t$)
  \[
  D_{KL}(s, t) = E_{x \sim s} [\log(s(x)) - \log(t(x))]
  \]
- Evaluating and optimizing Monte Carlo estimates of this objective requires:
  - Samples $x$ from student model (IAF)
  - Density of $x$ assigned by student model (IAF)
  - Density of $x$ assigned by teacher model (MAF)
- All operations above can be implemented efficiently!
Parallel Wavenet: Overall algorithm

• Training
  • Step 1: Train teacher model (MAF) via MLE
  • Step 2: Train student model (IAF) to minimize KL divergence with teacher

• Test-time: Use student model for testing

• Improves sampling efficiency over original Wavenet (vanilla autoregressive model) by 1000x!

• Useful in speech synthesis
• Coupling layer based normalizing flow models
  • Coupling layer
  • NICE add only
  • Real NVP add+mul
  • Glow conv 1x1
• Autoregressive models as flow models
  • MAF fast train, slow test
  • IAF fast test, slow train
  • Parallel Wavenet fast train, fast test
Summary of Normalizing Flow Models

• Transform simple distributions into more complex distributions via change of variables

• Jacobian of transformations should have tractable determinant for efficient learning and density estimation

• Computational tradeoffs in evaluating forward and inverse transformations
Thanks