Invertible Normalizing Flows

ECE57000: Artificial Intelligence, Fall 2019
David I. Inouye
Announcements

- Quiz moved to Friday
  - Same content (i.e., up to DCGANs, not today)
GAN Limitation:
Cannot compute density values

▶ Evaluation of GANs is challenging
  ▶ (Explicit density models could use test log likelihood)
  ▶ “I think this looks better than that”

▶ Inception scores
  ▶ Train separate image classifier
  ▶ See if passing fakes to classifier produces a high confidence prediction

▶ Cannot use for classification or outlier detection
GAN Limitation: Challenging to train because of careful balance between discriminator and generator

1. Assumptions on possible $D$ and $G$
   1. Theory – All possible $D$ and $G$
   2. Reality – Only functions defined by a neural network

2. Assumptions on optimality
   1. Theory – Both optimizations are solved perfectly
   2. Reality – The inner maximization is only solved approximately, and this interacts with outer minimization

3. Assumption on expectations
   1. Theory – Expectations over true distribution
   2. Reality – Empirical expectations over finite sample; for images, much of the high-dimensional space does not have samples

GANs can be very difficult/finicky to train
GAN Limitation: Cannot go from observed to latent space, i.e. $x \rightarrow z$ not possible/easy

- Cannot manipulate an observed image in latent space
  - Cannot do the following, $x \rightarrow z$, $z' = z + 3$, $z' \rightarrow x'$
  - Rather, must start from fake image based on random $z$

All fake images->
Normalizing flows use invertible deep models for the generator which allow more capabilities

- Transforming between observed/input and latent space is easy
  - $x = G(z)$
  - $z = G^{-1}(x)$

- Simple sampling like GANs
  - $z \sim \text{SimpleDistribution}$
  - $x = G(z) \sim \hat{p}_g(x)$, which is estimated distribution

- **Exact density** is computable via change of variables
  - Standard maximum likelihood estimation can be used for training
Highly realistic random samples from powerful flow model (GLOW)

Figure 1: Synthetic celebrities sampled from our model; see Section 3 for architecture and method, and Section 5 for more results.
Interpolation between real images using GLOW

Figure 5: Linear interpolation in latent space between real images.

Transformations of real image along various features

Figure 6: Manipulation of attributes of a face. Each row is made by interpolating the latent code of an image along a vector corresponding to the attribute, with the middle image being the original image.
Back to maximum likelihood estimation (MLE): How can we compute the likelihood for normalizing flows?

▸ Suppose
  ▸ $z \sim \text{Uniform}([0,1])$, i.e., $p_z(z) = 1$ (latent space is uniform)
  ▸ $G(z) = 2z$
  ▸ Thus, $x = G(z) = 2z$.

▸ What is the density function of $x$, what is $p_x(x)$?
Change of variables formula gives $p_x$ in terms of the $p_z$ and the derivative of $G^{-1}$

- Key idea: Must conserve density *volume* (so that distribution sums to 1).
- $p_x(x)|dx| = p_z(z)|dz|$, this is like the preservation of volume/area/mass.
  - Intuition: We only have 1 unit of “dirt” to move around.
- Rearrange above equation to get formula
  \[
  p_x(x) = \left| \frac{dz}{dx} \right| p_z(z) = \left| \frac{dG^{-1}(x)}{dx} \right| p_z(G^{-1}(x))
  \]
Derivation of change of variables using CDF function (Increasing)

- Assume $x = G(z)$, where $G(z)$ is an increasing function
- $F_x(a) = \Pr(x \leq a) = \int_{-\infty}^{a} p_x(t)dt$
- $F_x(a) = \Pr(x \leq a) = \Pr(G(z) \leq a) = \Pr(z \leq G^{-1}(a)) = F_z(G^{-1}(a))$
- Now take the derivative of both sides with respect to $a$

\[
\frac{dF_x(a)}{da} = p_x(a) \\
\frac{dF_z(G^{-1}(a))}{da} = \frac{dF_z(G^{-1}(a))}{d(G^{-1}(a))} \left(\frac{dG^{-1}(a)}{da}\right) \\
= p_z(G^{-1}(a)) \left(\frac{dG^{-1}(a)}{da}\right)
\]
Derivation of change of variables using CDF function (Decreasing)

- Assume \( x = G(z) \), where \( G(z) \) is a decreasing function
- \( F_x(a) = \Pr(x \leq a) = \int_{-\infty}^{a} p_x(t)dt \)
- \( F_x(a) = \Pr(x \leq a) = \Pr(G(z) \leq a) \\
  = \Pr(z > G^{-1}(a)) = 1 - F_z(G^{-1}(a)) \)
- Now take the derivative of both sides with respect to \( a \)

\[
\frac{dF_x(a)}{da} = p_x(a) \\
- \frac{dF_z(G^{-1}(a))}{da} = - \frac{dF_z(G^{-1}(a))}{d(G^{-1}(a))} \left( \frac{dG^{-1}(a)}{da} \right) \\
= -p_z(G^{-1}(a)) \left( \frac{dG^{-1}(a)}{da} \right)
\]
Inverse transform sampling is based on change of variables

- \( z \sim \text{Uniform}([0,1]) \)
- \( \nu \sim \text{AnotherDistribution} \)
- \( x = F_{\nu}^{-1}(z) \), where \( F_{\nu}^{-1} \) is the inverse CDF for \( \nu \)
- What is the distribution of \( x \)?

\[
p_x(x) = p_z(F_{\nu}(x)) \left| \frac{dF_{\nu}(x)}{dx} \right|
\]

\[
p_x(x) = (1)|p_{\nu}(x)| = p_{\nu}(x)
\]
Announcements

▸ Moved office hours from Thursday to Wed, 3:30pm—4:30pm

▸ Project submission instructions

▸ Quiz 6
  ▸ Transposed convolution
What about change of variables in higher dimensions?

- Let’s again build a little intuition (see demo)
- Again, conservation of volume: Consider infinitesimal expansion or shrinkage of volume
  \[ p(x_1, x_2) |dx_1 \, dx_2| = p(z_1, z_2) |dz_1 \, dz_2| \]
- Given that Jacobian is all mixed derivatives we get generalization for vector to vector invertible functions:
  \[ p_x(x) = |\det J_{G^{-1}}(x)| \, p_z(G^{-1}(x)) \]
What is the Jacobian again?
The best linear approximation at a point

The Jacobian definition:

\[
\frac{dz}{dx} = J_z(x) = \begin{bmatrix}
\frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial z_d}{\partial x_1} & \cdots & \frac{\partial z_d}{\partial x_d}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial G^{-1}(x)_1}{\partial x_1} & \cdots & \frac{\partial G^{-1}(x)_1}{\partial x_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial G^{-1}(x)_d}{\partial x_1} & \cdots & \frac{\partial G^{-1}(x)_d}{\partial x_d}
\end{bmatrix}
\]

The determinant measures the local expansion or shrinkage around a point.
One useful identity for determinant of Jacobian of invertible function

- We can relate the Jacobian of a function to the Jacobian of the inverse in a natural way

\[ J_{G^{-1}}(x) = J_{G^{-1}}(G(z)) = [J_G(z)]^{-1} = J_G(z)^{-1} \]

- Using this we can come up with different ways of writing the change of variables formula

\[
\begin{align*}
    p_x(x) &= |\det J_{G^{-1}}(x)| \ p_z(G^{-1}(x)) \\
    p_x(x) &= |\det J_G(z)|^{-1} \ p_z(G^{-1}(x)) \\
    p_x(x) &= |\det J_G(z)|^{-1} \ p_z(z) \\
    p_z(z) &= |\det J_G(z)| \ p_x(x) \\
    p_z(z) &= |\det J_G(z)| \ p_x(G(z))
\end{align*}
\]
The determinant Jacobian of compositions of functions is the product of determinant Jacobians

- Suppose $F(x) = F_2(F_1(x))$
- The Jacobian expands like the chain rule
  $$J_F(x) = J_{F_2}(F_1(x))J_{F_1}(x) = J_{F_2}J_{F_1}$$
- If we take the determinant of the Jacobian, then it becomes a product of determinants
  $$\det J_F = \det J_{F_2}J_{F_1} = (\det J_{F_2})(\det J_{F_1})$$
- This will be useful since each layer of our flows will be invertible
Okay, now back to learning flows:
The log likelihood is the sum of determinant terms for each layer

- Simply optimize the maximize likelihood of model $F_\theta = G^{-1}$

$$
\arg \min_{F_\theta} - \log \prod_i \hat{p}_x(x_i; \theta) \\
\arg \min_{F_\theta} - \sum_i \left[ \log p_z(F_\theta(x_i)) + \log|\det J_{F_\theta}(x_i)| \right] \\
\arg \min_{F_\theta} - \sum_i \left[ \log p_z(F_\theta(x_i)) + \sum_{\ell} \log|\det J_{F_\theta}^{(\ell)}| \right]
$$
How do we create these invertible layers?

- Consider arbitrary invertible transformation $F_\theta$
  - How often would $|\det J_{F_\theta}|$ need to be computed?

- High computation costs
  - Determinant costs roughly $O(d^3)$ even if Jacobian is already computed!
  - Would need to be computed every stochastic gradient iteration
How do we create these invertible layers? Independent transformation on each dimension

- \( z_1 = F_1(x_1) \)
- \( z_2 = F_2(x_2) \)
- \( z_3 = F_3(x_3) \)

- What is the Jacobian?

\[
J_F = \begin{bmatrix}
\frac{dF_1(x_1)}{dx_1} & 0 & 0 \\
0 & \frac{dF_2(x_2)}{dx_2} & 0 \\
0 & 0 & \frac{dF_3(x_3)}{dx_3}
\end{bmatrix}
\]
How do we create these invertible layers? Autoregressive flows based on chain rule

- **Forward - Density estimation (in parallel)**
  - \( z_1 = F_1(x_1) \)
  - \( z_2 = F_2(x_2|x_1) \)
  - \( z_3 = F_3(x_3|x_1, x_2) \)

- **Inverse – Sampling (conditioned on \( x \) so must be sequential)**
  - \( x_1 = F_1^{-1}(z_1) \)
  - \( x_2 = F_2^{-1}(z_2|x_1) \)
  - \( x_3 = F_3^{-1}(z_3|x_1, x_2) \)

- **What is the Jacobian and determinant?**
  - Product of diagonal!

\[
J_F = \begin{bmatrix}
\frac{dF_1}{dx_1} & 0 & 0 \\
\frac{dF_2}{dx_1} & \frac{dF_2}{dx_2} & 0 \\
\frac{dF_3}{dx_1} & \frac{dF_3}{dx_2} & \frac{dF_3}{dx_3}
\end{bmatrix}
\]
How do we create these invertible layers?

**Autoregressive flows based on chain rule**

- **Forward - Density estimation (sequential)**
  - $z_1 = F_1(x_1)$
  - $z_2 = F_2(x_2|z_1)$
  - $z_3 = F_3(x_3|z_1, z_2)$

- **Inverse – Sampling (parallel)**
  - $x_1 = F_1^{-1}(z_1)$
  - $x_2 = F_2^{-1}(z_2|z_1)$
  - $x_3 = F_3^{-1}(z_3|z_1, z_2)$

- **What is the Jacobian and determinant?**
  - Product of diagonal!

$$J_F = \begin{bmatrix}
\frac{dF_1}{dx_1} & 0 & 0 \\
\frac{dF_2}{dx_1} & \frac{dF_2}{dx_2} & 0 \\
\frac{dF_3}{dx_1} & \frac{dF_3}{dx_2} & \frac{dF_3}{dx_3}
\end{bmatrix}$$

- **Forward – Density estimation** *(sequential)*
  - $z_1 = \exp(\alpha_1)x_1 + \mu_1$
  - $z_2 = \exp(\alpha_2)x_2 + \mu_2$, $\alpha_2 = f_2(x_1)$, $\mu_2 = g_2(x_1)$
  - $z_3 = \exp(\alpha_3)x_3 + \mu_3$, $\alpha_3 = f_3(x_1, x_2)$, $\mu_3 = g_3(x_1, x_2)$

- **What is the Jacobian and determinant?**

$$J_F = \begin{bmatrix}
\exp(\alpha_1) & 0 & 0 \\
\frac{dz_2}{dx_1} & \exp(\alpha_2) & 0 \\
\frac{dz_3}{dx_1} & \frac{dz_3}{dx_2} & \exp(\alpha_3)
\end{bmatrix}$$

> Keep some set of features **fixed** and transform others

- $z_{1:i-1} = x_{1:i-1}$
- $z_{i:d} = \exp(f(x_{1:i-1})) \odot x_{i:d} + g(x_{1:i-1})$

> Reverse or shuffle coordinates and repeat

> What is Jacobian?

$$J_F = \begin{bmatrix} I & 0 \\ J_{cross} & \text{diag}(\exp(f(x_{1:i-1}))) \end{bmatrix}$$
Checkboard or channel-wise masking can be used to separate fixed and non-fixed set of variables.
The squeeze operation trades off between spatial and channel dimensions.

**H x W x C**

**Squeeze**

**H/2 x W/2 x 4C**
GLOW: Convolutional flows
1 x 1 invertible convolutions are like fully connected layers for each pixel

▸ Suppose an image has dimension \( h \times w \times c \)
▸ Remember that a “1x1” convolution has kernel size of \( 1 \times 1 \times c \)
▸ Thus if we use \( c \) filters than we map from a \( h \times w \times c \) to another \( h \times w \times c \) image
▸ The number of parameters is a matrix \( K \in \mathbb{R}^{c \times c} \)
▸ Thus, a 1 x 1 convolution can also be seen as a linear transformation along the channel dimension
Highly realistic random samples from powerful flow model (GLOW)

Figure 1: Synthetic celebrities sampled from our model; see Section 3 for architecture and method, and Section 5 for more results.

Similar concepts can be used to generate realistic audio (WaveGlow)

- Listen to some examples
  https://nv-adlr.github.io/WaveGlow

- Very similar concepts for audio generation