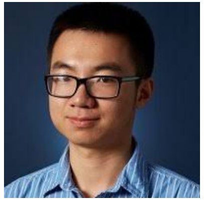
CS294-158 Deep Unsupervised Learning

Lecture 2c: Likelihood Models II: Flow-based Models

Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas

UC Berkeley

Pros and cons of models so far

- The ultimate goal: a likelihood-based model with…
  - Fast training and sampling
  - Good samples and good compression performance
- So far: discrete autoregressive models
  - Pros: fast evaluation of p(x), great compression performance, good samples with carefully designed dependence structure
  - Cons: Slow sampling (at least without significant engineering), discrete data only

The framework of flows will let us design models that trade off between these pros and cons.

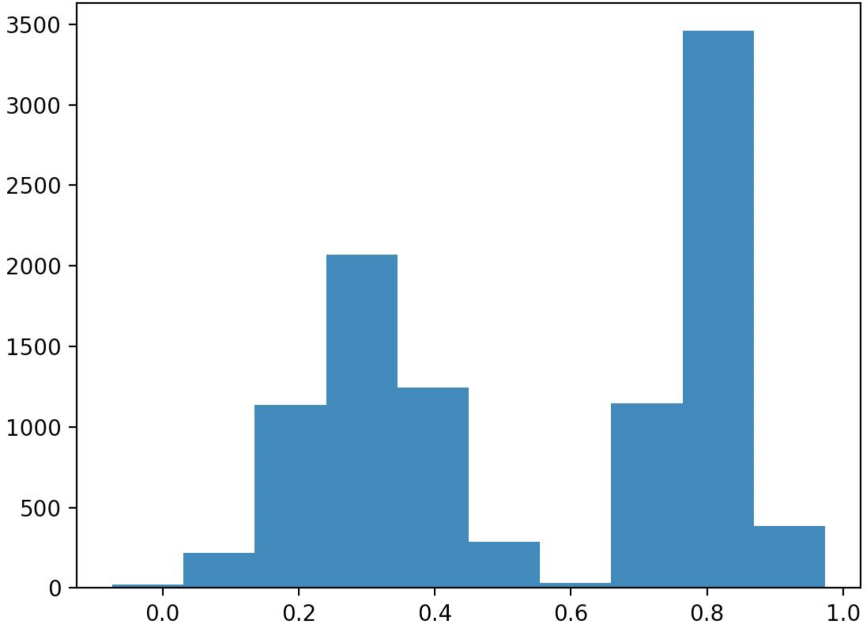
Outline for flows

- ***Definition, change of variables formula, training***
- Constructions of flows
  - Density models and flows in 1D
  - Flows in high dimensions
    - Affine flows, elementwise flows
    - Coupling layers. NICE/RealNVP
    - Constructions based on directed graphical models. Autoregressive flows and inverse autoregressive flows
    - More types of flows: invertible 1x1 convs, FFJORD, etc
- Dequantization, compression

Density models

Continuous data

```
0.22159854, 0.84525919, 0.09121633, 0.364252  , 0.30738086,
0.32240615, 0.24371194, 0.22400792, 0.39181847, 0.16407012,
0.84685229, 0.15944969, 0.79142357, 0.6505366 , 0.33123603,
0.81409325, 0.74042126, 0.67950372, 0.74073271, 0.37091554,
0.83476616, 0.38346571, 0.33561352, 0.74100048, 0.32061713,
0.09172335, 0.39037131, 0.80496586, 0.80301971, 0.32048452,
0.79428266, 0.6961708 , 0.20183965, 0.82621227, 0.367292  ,
0.76095756, 0.10125199, 0.41495427, 0.85999877, 0.23004346,
0.28881973, 0.41211802, 0.24764836, 0.72743029, 0.20749136,
0.29877091, 0.75781455, 0.29219608, 0.79681589, 0.86823823,
0.29936483, 0.02948181, 0.78528968, 0.84015573, 0.40391632,
0.77816356, 0.75039186, 0.84709016, 0.76950307, 0.29772759,
0.41163966, 0.24862007, 0.34249207, 0.74363912, 0.38303383, ...
```

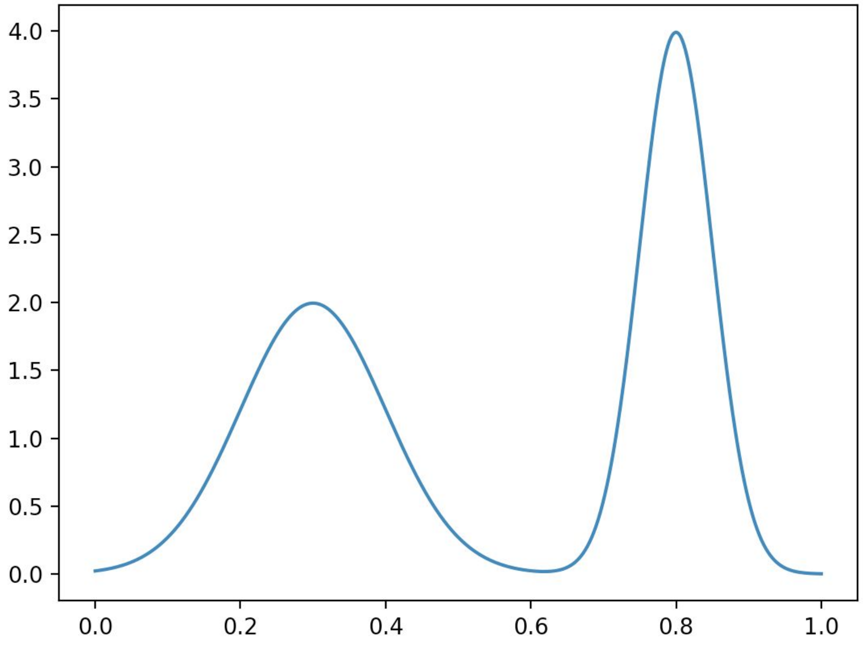


One modeling approach: quantize and fit a discrete model

Density models

Continuous data

```
0.22159854, 0.84525919, 0.09121633, 0.364252  , 0.30738086,
0.32240615, 0.24371194, 0.22400792, 0.39181847, 0.16407012,
0.84685229, 0.15944969, 0.79142357, 0.6505366 , 0.33123603,
0.81409325, 0.74042126, 0.67950372, 0.74073271, 0.37091554,
0.83476616, 0.38346571, 0.33561352, 0.74100048, 0.32061713,
0.09172335, 0.39037131, 0.80496586, 0.80301971, 0.32048452,
0.79428266, 0.6961708 , 0.20183965, 0.82621227, 0.367292  ,
0.76095756, 0.10125199, 0.41495427, 0.85999877, 0.23004346,
0.28881973, 0.41211802, 0.24764836, 0.72743029, 0.20749136,
0.29877091, 0.75781455, 0.29219608, 0.79681589, 0.86823823,
0.29936483, 0.02948181, 0.78528968, 0.84015573, 0.40391632,
0.77816356, 0.75039186, 0.84709016, 0.76950307, 0.29772759,
0.41163966, 0.24862007, 0.34249207, 0.74363912, 0.38303383, ...
```



This lecture: define and fit a **density model**

Mixtures of Gaussians

A density model is a parameterized **probability density function**.

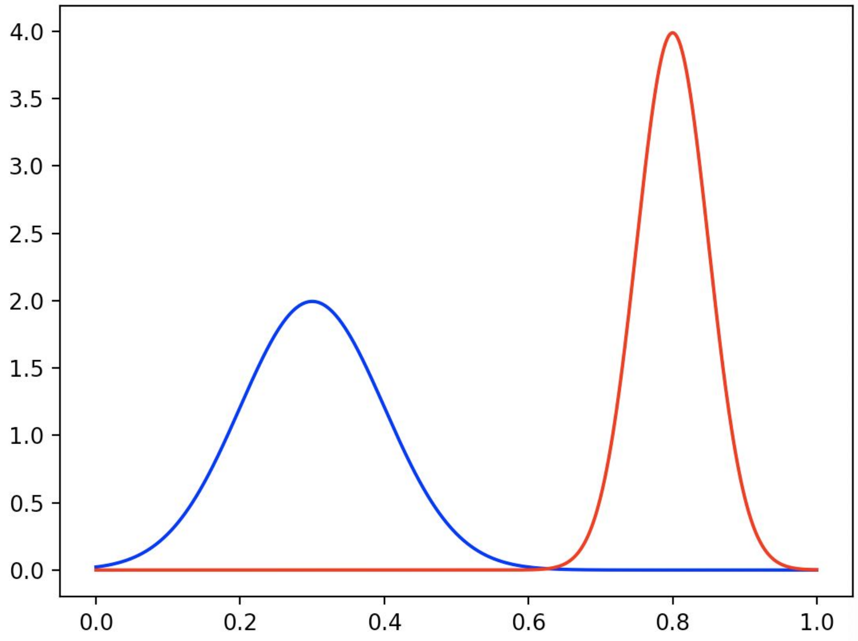
Example: **Mixture of Gaussians**

- Parameters: means and variances of components, mixture weights

$$p_\theta(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x; \mu_i, \sigma_i^2)$$

- Fit with maximum likelihood

$$\arg\min_\theta \mathbb{E}_x[-\log p_\theta(x)]$$



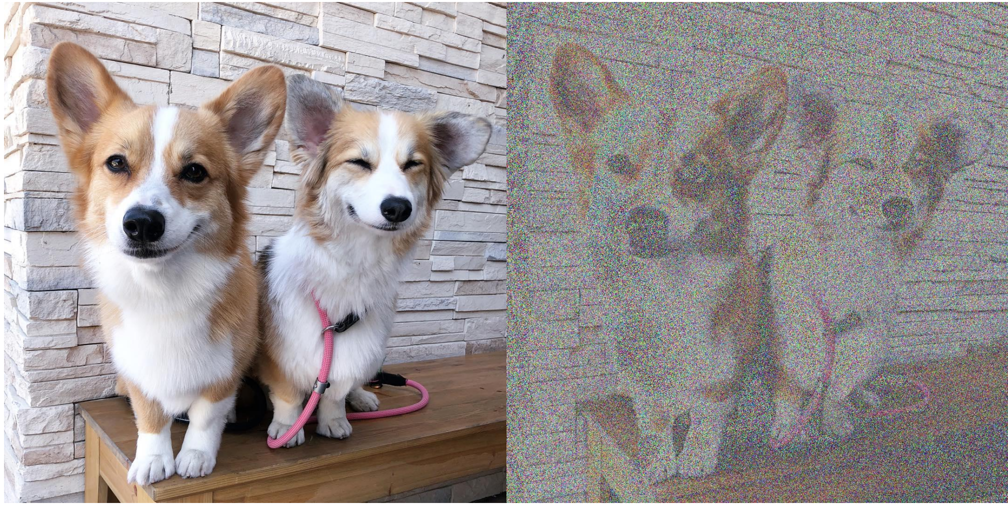
Mixtures of Gaussians

Do mixtures of Gaussians work for high-dimensional data?

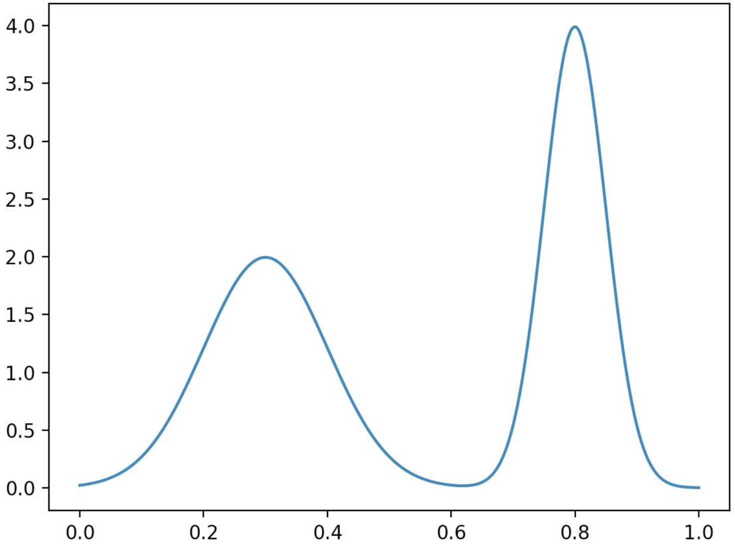
Not really. The sampling process is:

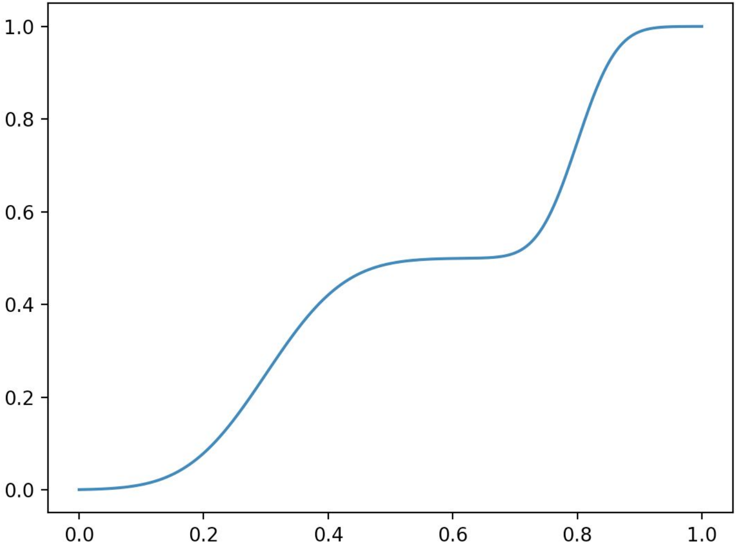
1. Pick a cluster center
2. Add Gaussian noise

Imagine this for modeling natural images! The only way a realistic image can be generated is if it is a cluster center, i.e. if it is already stored directly in the parameters.



Shifting perspective to the CDF





$$p_\theta(x)$$

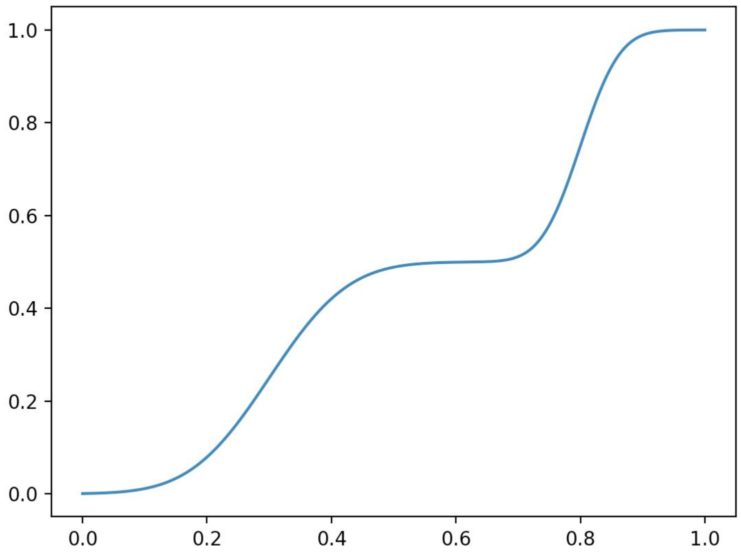
$$f_\theta(x) = \int_{-\infty}^{x} p_\theta(t)\, dt$$

Sampling via inverse CDF

Sampling from the model:

$$z \sim \text{Uniform}([0,1])$$
$$x = f_\theta^{-1}(z)$$

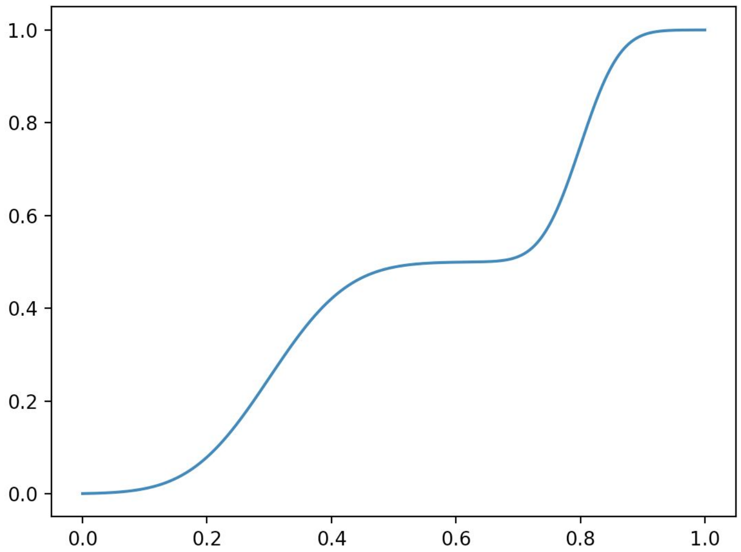
The CDF is an invertible, differentiable map from data to [0, 1]



$$f_\theta(x) = \int_{-\infty}^{x} p_\theta(t)\,dt$$

CDFs in 1D

- **Flow from x to z**: an invertible differentiable function from x to z
- Training the PDF is the same as **training the CDF to map Uniform([0, 1]) to the data distribution**
- Equivalently, it trains the CDF so that **mapping Uniform([0, 1]) through the inverse CDF yields the data distribution**
- Let's **work with flows directly**, instead of treating them as derived objects.

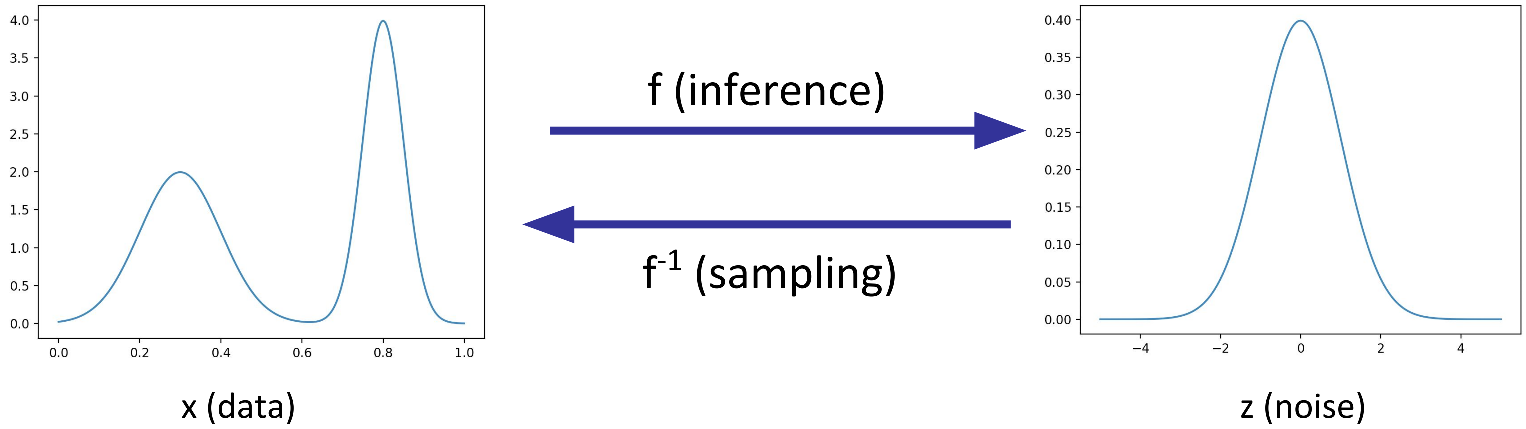


$$f_\theta(x) = \int_{-\infty}^{x} p_\theta(t)\, dt$$

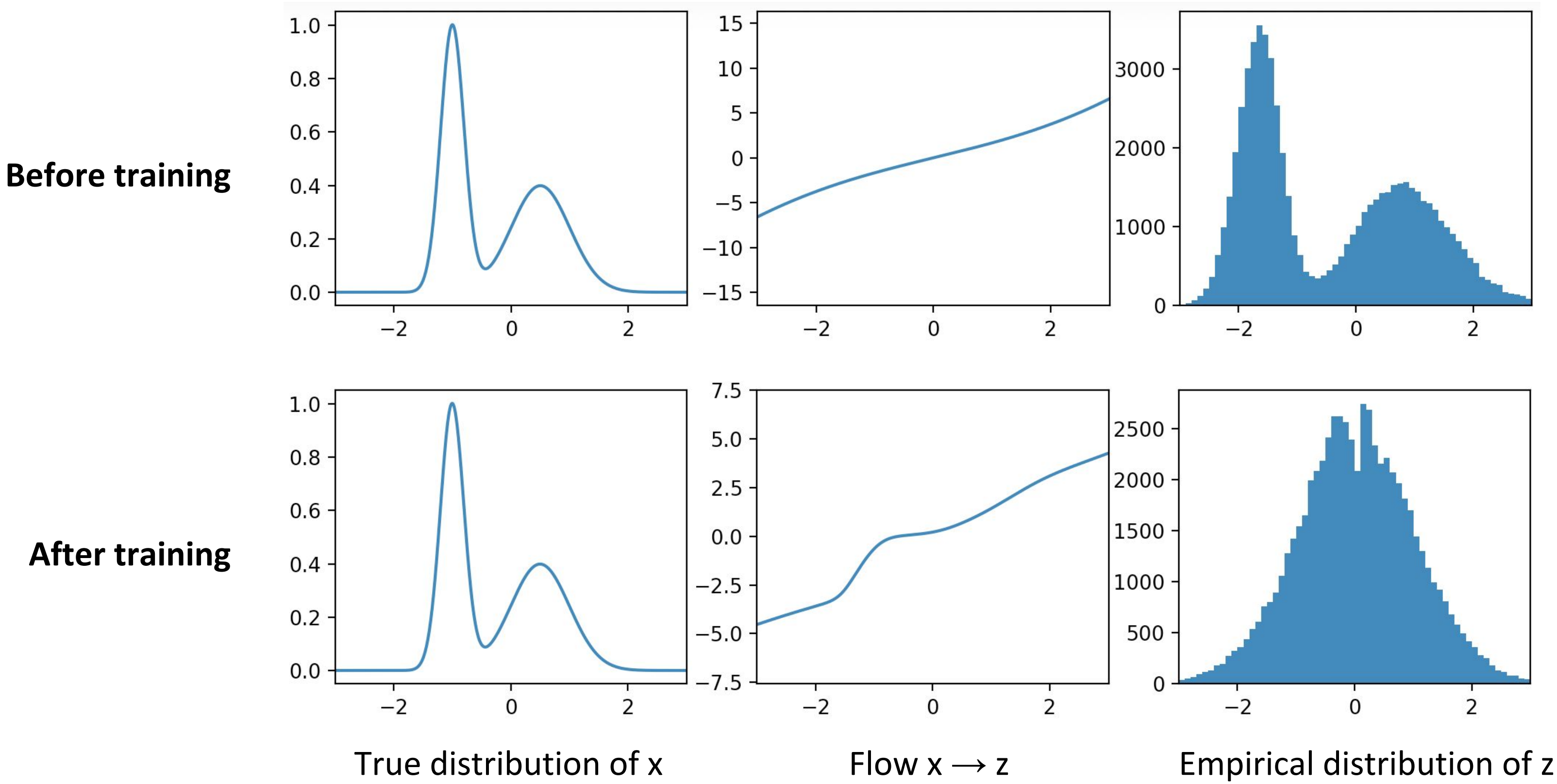
Flows in 1D

Flow: a differentiable, invertible mapping from x (data) to z (noise)

- Train so that it turns the data distribution into a **base distribution p(z)**
  - Common choices: uniform, standard normal
- This way, mapping **z ~ p(z)** through the flow's **inverse** will yield good samples



Example: Flow to Gaussian



Fitting flows

We will fit flows with maximum likelihood

$$\arg\min_{\theta} \mathbb{E}_x[-\log p_\theta(x)]$$

Need $p_\theta(x)$. A flow f_θ defines a distribution over x via sampling:

$$\mathbf{z} \sim p(\mathbf{z}) \qquad \mathbf{x} = f_\theta^{-1}(\mathbf{z})$$

So $p_\theta(x)$ is the density of x under this sampling process. But how do we calculate it?

Change of variables: intuition

Change of variables: 1D

$$z = f_\theta(x)$$

$$p_\theta(x)\,dx = p(z)\,dz$$

$$p_\theta(x) = p(f_\theta(x))\left|\frac{\partial f_\theta(x)}{\partial x}\right|$$

CDF Flows

If we use a flow defined to be a CDF, we recover the original objective for fitting the corresponding PDF.

Of course, flows can be more general than CDFs.

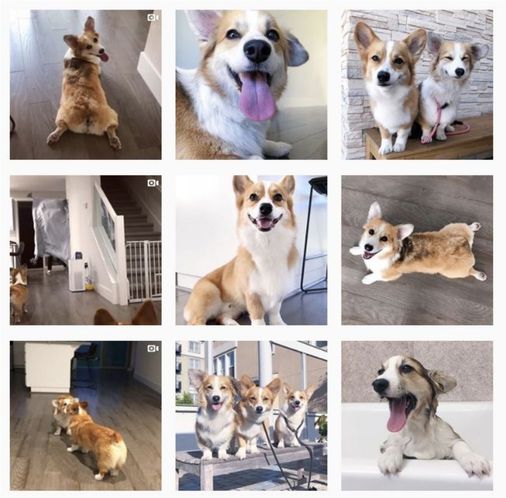
Recap so far

- Instead of working with a probability density function over x, work with a flow from x to z
  - Transform data distribution to a base distribution p(z)
  - Sampling via mapping from z to x through the inverse flow
  - The CDF is a flow from data to Uniform([0, 1])
- Benefit of the flow viewpoint
  - Generalize to arbitrary base distributions p(z)
  - Gateway to architectures for high-dimensional data

Outline for flows

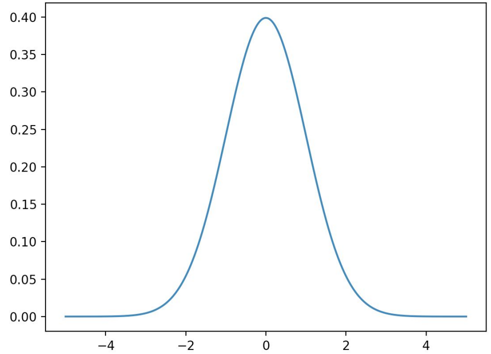
- Definition, change of variables formula, training
- Constructions of flows
  - Density models and flows in 1D
  - ***Flows in high dimensions***
    - Affine flows, elementwise flows
    - Coupling layers. NICE/RealNVP
    - Constructions based on directed graphical models. Autoregressive flows and inverse autoregressive flows
    - More types of flows: invertible 1x1 convs, FFJORD, etc
- Dequantization, compression

High-dimensional data



f (inference)

f^{-1} (sampling)



x and z must have the same dimension

Change of variables

For $z \sim p(z)$, sampling process f^{-1} linearly transforms a small cube dz to a small parallelepiped dx. Probability is conserved:

$$p(x) = p(z)\frac{\mathrm{vol}(dz)}{\mathrm{vol}(dx)} = p(z)\left|\det\frac{dz}{dx}\right|$$

Intuition: x is likely if it maps to a “large” region in z space

Flow models: training

Change-of-variables formula lets us compute the density over x:

$$p_\theta(\mathbf{x}) = p(f_\theta(\mathbf{x})) \left| \det \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right|$$

Train with maximum likelihood:

$$\arg\min_\theta \mathbb{E}_{\mathbf{x}} \left[-\log p_\theta(\mathbf{x}) \right] = \mathbb{E}_{\mathbf{x}} \left[-\log p(f_\theta(\mathbf{x})) - \log \det \left| \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right| \right]$$

Key requirement: the Jacobian determinant must be easy to calculate and differentiate!

Constructing flows: composition

- **Flows can be composed**

$$x \rightarrow f_1 \rightarrow f_2 \rightarrow \ldots f_k \rightarrow z$$

$$z = f_k \circ \cdots \circ f_1(x)$$

$$x = f_1^{-1} \circ \cdots \circ f_k^{-1}(z)$$

$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^{k} \log \left| \det \frac{\partial f_i}{\partial f_{i-1}} \right|$$

- Easy way to increase expressiveness

Affine flows

- Another name for affine flow: multivariate Gaussian.
  - Parameters: an invertible matrix A and a vector b
  - $f(x) = A^{-1}(x - b)$
- Sampling: $x = Az + b$, where $z \sim N(0, I)$
- Log likelihood is expensive when dimension is large.
  - The Jacobian of f is A^{-1}
  - Log likelihood involves calculating det(A)

Elementwise flows

$$f_\theta((x_1, \ldots, x_d)) = (f_\theta(x_1), \ldots, f_\theta(x_d))$$

- Lots of freedom in elementwise flow
  - Can use elementwise affine functions or CDF flows.
- The Jacobian is diagonal, so the determinant is easy to evaluate.

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathrm{diag}(f'_\theta(x_1), \ldots, f'_\theta(x_d))$$

$$\det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \prod_{i=1}^{d} f'_\theta(x_i)$$

NICE/RealNVP

Affine coupling layer

- Split variables in half: $x_{1:d/2}$, $x_{d/2+1:d}$

$$\mathbf{z}_{1:d/2} = \mathbf{x}_{1:d/2}$$
$$\mathbf{z}_{d/2:d} = \mathbf{x}_{d/2:d} \cdot s_\theta(\mathbf{x}_{1:d/2}) + t_\theta(\mathbf{x}_{1:d/2})$$

- Invertible! Note that s_θ and t_θ can be arbitrary neural nets with **no restrictions**.
  - Think of them as **data-parameterized elementwise flows**.

NICE/RealNVP

- It also has a tractable Jacobian determinant

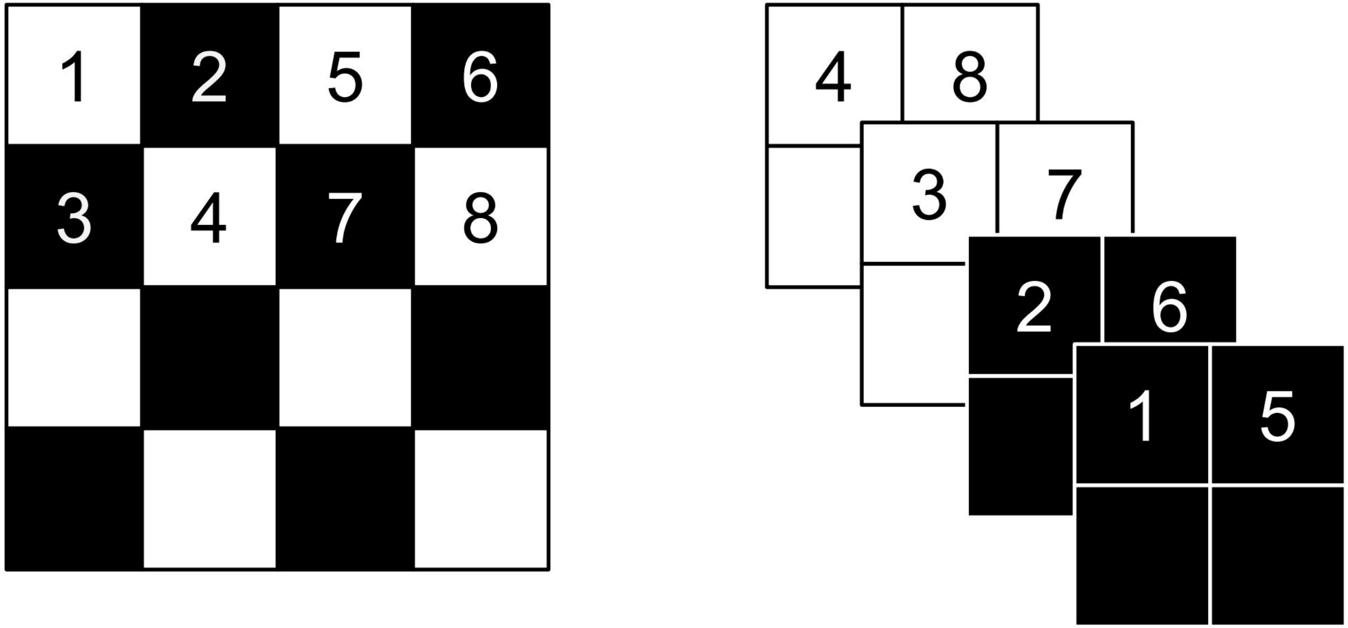
$$\mathbf{z}_{1:d/2} = \mathbf{x}_{1:d/2}$$
$$\mathbf{z}_{d/2:d} = \mathbf{x}_{d/2:d} \cdot s_\theta(\mathbf{x}_{1:d/2}) + t_\theta(\mathbf{x}_{1:d/2})$$

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} I & 0 \\ \frac{\partial \mathbf{z}_{d/2:d}}{\partial \mathbf{x}_{1:d/2}} & \operatorname{diag}(s_\theta(\mathbf{x}_{1:d/2})) \end{bmatrix}$$

- The Jacobian is triangular, so its determinant is the product of diagonal entries.

$$\det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \prod_{k=1}^{d} s_\theta(\mathbf{x}_{1:d/2})_k$$

RealNVP: How to partition variables?



RealNVP

- Takeaway: coupling layers allow unrestricted neural nets to be used in flows, while preserving invertibility and tractability

[Dinh et al. Density estimation using Real NVP. ICLR 2017]

Constructing flows: directed graphical models

- RealNVP may seem magical. Where does it come from?

- Inspired by Bayes nets, we can construct flows from directed acyclic graphs.

Autoregressive flows

- The sampling process of a Bayes net is a flow
  - If autoregressive, this flow is called an **autoregressive flow**

$$
\begin{aligned}
x_1 &\sim p_\theta(x_1) & \qquad x_1 &= f_\theta^{-1}(z_1) \\
x_2 &\sim p_\theta(x_2 | x_1) & \qquad x_2 &= f_\theta^{-1}(z_2; x_1) \\
x_3 &\sim p_\theta(x_3 | x_1, x_2) & \qquad x_3 &= f_\theta^{-1}(z_3; x_1, x_2)
\end{aligned}
$$

- Sampling is an **invertible** mapping from z to x
- The DAG structure causes the Jacobian to be triangular, when variables are ordered by topological sort

Autoregressive flows

- How to fit autoregressive flows?
  - Map **x** to **z**
  - Fully parallelizable

$$p_\theta(\mathbf{x}) = p(f_\theta(\mathbf{x})) \left| \det \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right|$$

- Notice
  - **x** $\rightarrow$ **z** has the same structure as the **log likelihood** computation of an autoregressive model
  - **z** $\rightarrow$ **x** has the same structure as the **sampling** procedure of an autoregressive model

$$z_1 = f_\theta(x_1) \qquad\qquad x_1 = f_\theta^{-1}(z_1)$$

$$z_2 = f_\theta(x_2; x_1) \qquad\qquad x_2 = f_\theta^{-1}(z_2; x_1)$$

$$z_3 = f_\theta(x_3; x_1, x_2) \qquad\qquad x_3 = f_\theta^{-1}(z_3; x_1, x_2)$$

Inverse autoregressive flows

- The inverse of an autoregressive flow is also a flow, called the **inverse autoregressive flow (IAF)**
  - $\mathbf{x} \longrightarrow \mathbf{z}$ has the same structure as the **sampling** in an autoregressive model
  - $\mathbf{z} \longrightarrow \mathbf{x}$ has the same structure as **log likelihood** computation of an autoregressive model. So, **IAF sampling is fast**

$$\begin{aligned} z_1 &= f_\theta^{-1}(x_1) \\ z_2 &= f_\theta^{-1}(x_2; z_1) \\ z_3 &= f_\theta^{-1}(x_3; z_1, z_2) \end{aligned}$$

$$\begin{aligned} x_1 &= f_\theta(z_1) \\ x_2 &= f_\theta(z_2; z_1) \\ x_3 &= f_\theta(z_3; z_1, z_2) \end{aligned}$$

AF vs IAF

- Autoregressive flow
  - **Fast** evaluation of p(x) for arbitrary x
  - **Slow** sampling
- Inverse autoregressive flow
  - **Slow** evaluation of p(x) for arbitrary x, so training directly by maximum likelihood is slow.
  - **Fast** sampling
  - **Fast** evaluation of p(x) if x is a sample
- There are models (Parallel WaveNet, IAF-VAE) that exploit IAF’s fast sampling

Back to RealNVP

- These constructions work for all Bayes net structures, not just autoregressive structures
- A RealNVP coupling layer corresponds to a certain Bayes net
  - Half of the variables are sampled independently
  - The other half are conditionally independent given the first half

$$\mathbf{x}_i = \mathbf{z}_i \cdot \mathbf{a}_\theta(\text{parent}(\mathbf{x}_i)) + \mathbf{b}_\theta(\text{parent}(\mathbf{x}_i))$$

Choice of coupling transformation

- A Bayes net defines coupling dependency, but what invertible transformation f to use is a design question

$$\mathbf{x}_i = f_\theta(\mathbf{z}_i; \text{parent}(\mathbf{x}_i))$$

- Affine transformation is the most commonly used one (NICE, RealNVP, IAF-VAE, …)

$$\mathbf{x}_i = \mathbf{z}_i \cdot \mathbf{a}_\theta(\text{parent}(\mathbf{x}_i)) + \mathbf{b}_\theta(\text{parent}(\mathbf{x}_i))$$

- More complex, nonlinear transformations -> better performance
  - CDFs and inverse CDFs for Mixture of Gaussians or Logistics (Flow++)
  - Piecewise linear/quadratic functions (Neural Importance Sampling)

NN architecture also matters

- Flow++ = MoL transformation + self-attention in NN
  - Bayes net (coupling dependency), transformation function class, NN architecture all play a role in a flow's performance. Still an

Table 2. CIFAR10 ablation results after 400 epochs of training. Models not converged for the purposes of ablation study.

Ablation	bits/dim	parameters
uniform dequantization	3.292	32.3M
affine coupling	3.200	32.0M
no self-attention	3.193	31.4M
Flow++ (not converged for ablation)	**3.165**	**31.4M**

Other classes of flows

- Glow (link)
  - Invertible 1x1 convolutions
  - Large-scale training

- Continuous time flows (FFJORD)
  - Allows for unrestricted architectures. Invertibility and fast log probability computation guaranteed.

Unifying discrete autoregressive models

- Discrete models (incl discrete AR models) are also invertible transformations between discrete data and the noise space
- Discrete autoregressive models is essentially transforming data back to noise space (the index of inverse cdf of each conditional)

Outline for flows

- Definition, change of variables formula, training
- Constructions of flows
  - Density models and flows in 1D
  - Flows in high dimensions
    - Affine flows, elementwise flows
    - Coupling layers. NICE/RealNVP
    - Constructions based on directed graphical models. Autoregressive flows and inverse autoregressive flows
    - More types of flows: invertible 1x1 convs, FFJORD, etc
- ***Dequantization***

Continuous flows for discrete data

- A problem arises when fitting continuous density models to discrete data: degeneracy
  - When the data are 3-bit pixel values, $\mathbf{x} \in \{0, 1, 2, \ldots, 255\}$
  - What density does a model assign to values between bins like 0.4, 0.42…?
- Correct semantics: we want the integral of probability density within a discrete interval to approximate discrete probability mass

$$P_{\text{model}}(\mathbf{x}) := \int_{[0,1)^D} p_{\text{model}}(\mathbf{x} + \mathbf{u})\, d\mathbf{u}$$

Continuous flows for discrete data

- Solution: **Dequantization**. Add noise to data.
  - $\mathbf{x} \in \{0, 1, 2, \ldots, 255\}$
  - We draw noise u uniformly from $[0, 1)^D$

$$\begin{aligned}
\mathbb{E}_{\mathbf{y}\sim p_{\text{data}}}\left[\log p_{\text{model}}(\mathbf{y})\right] &= \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \int_{[0,1)^D} \log p_{\text{model}}(\mathbf{x}+\mathbf{u})\, d\mathbf{u} \\
&\leq \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \log \int_{[0,1)^D} p_{\text{model}}(\mathbf{x}+\mathbf{u})\, d\mathbf{u} \\
&= \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_{\text{model}}(\mathbf{x})\right]
\end{aligned}$$

[Theis, Oord, Bethge, 2016]

Relationship to compression

- Maximum likelihood for densities: same as coding for a very fine discretization
- What about non-infinitesimal discretization? See how later.

Future directions

- The ultimate goal: a likelihood-based model with
  - fast sampling
  - fast inference
  - fast training
  - good samples
  - good compression
- Flows seem to let us achieve some of these criteria.
- But how exactly do we design and compose flows for great performance? That's an open question.

Bibliography

NICE: Dinh, Laurent, David Krueger, and Yoshua Bengio. "NICE: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

RealNVP: Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP." *arXiv preprint arXiv:1605.08803* (2016).

AF:Chen, Xi, et al. "Variational lossy autoencoder." *arXiv preprint arXiv:1611.02731* (2016).; Papamakarios, George, Theo Pavlakou, and Iain Murray. "Masked autoregressive flow for density estimation." *Advances in Neural Information Processing Systems*. 2017.

IAF: Kingma, Durk P., et al. "Improved variational inference with inverse autoregressive flow." *Advances in neural information processing systems*. 2016.

Flow++: Ho, Jonathan, et al. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design." *arXiv preprint arXiv:1902.00275* (2019).

Neural Importance Sampling: Müller, Thomas, et al. "Neural importance sampling." *arXiv preprint arXiv:1808.03*

Glow: Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." *Advances in Neural Information Processing Systems*. 2018.

FFJORD: Grathwohl, Will, et al. "Ffjord: Free-form continuous dynamics for scalable reversible generative models." *arXiv preprint arXiv:1810.01367* (2018).

Neural Autoregressive Flow: Huang, Chin-Wei, et al. "Neural autoregressive flows." arXiv preprint arXiv:1804.00779 (2018).