Discrete Flows

Invertible Generative Models of Discrete Data

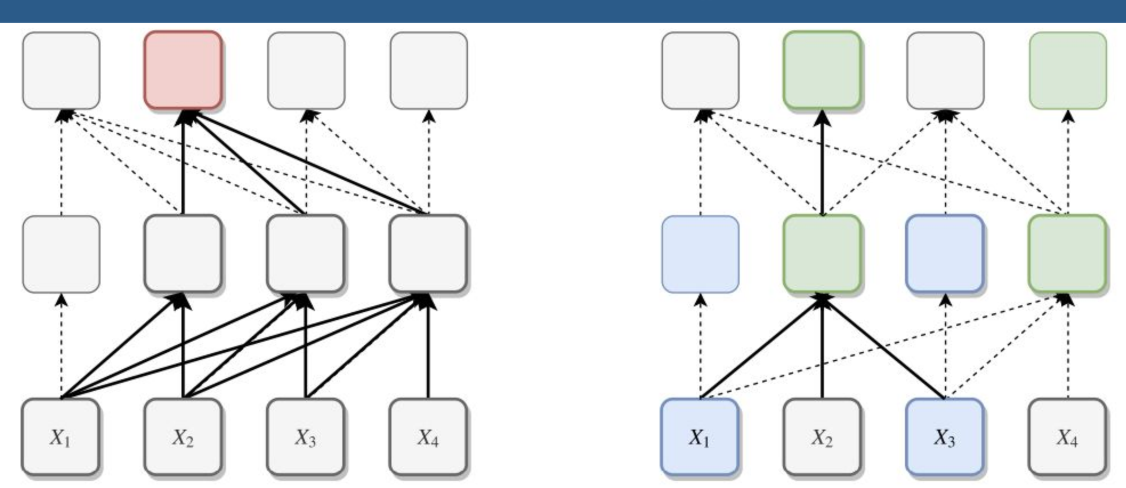
Dustin Tran[1] Keyon Vafa[1,2] Kumar Krishna Agrawal[1] Laurent Dinh[1] Ben Poole[1]
dustinvtran
[1]Google Brain [2] Columbia University

Summary

We develop flows for discrete distributions.

- **Discrete autoregressive flows** enable multiple levels of autoregressivity.
  Ex. Bidirectional language models that can generate data.
- **Discrete bipartite flows** enable flexible models with parallel generation.
  Ex. Non-autoregressive text models with an exact likelihood.

Normalizing Flows



(a) Autoregressive flows **(b)** Bipartite flows

Change of Variables.

$$p(y) = p\big(x = f^{-1}(y)\big)\left|\frac{\partial f^{-1}}{\partial y}\right|$$

Autoregressive Flows.

$$y_d = x_d \cdot \sigma_d(x_{<d}) + \mu_d(x_{<d})$$

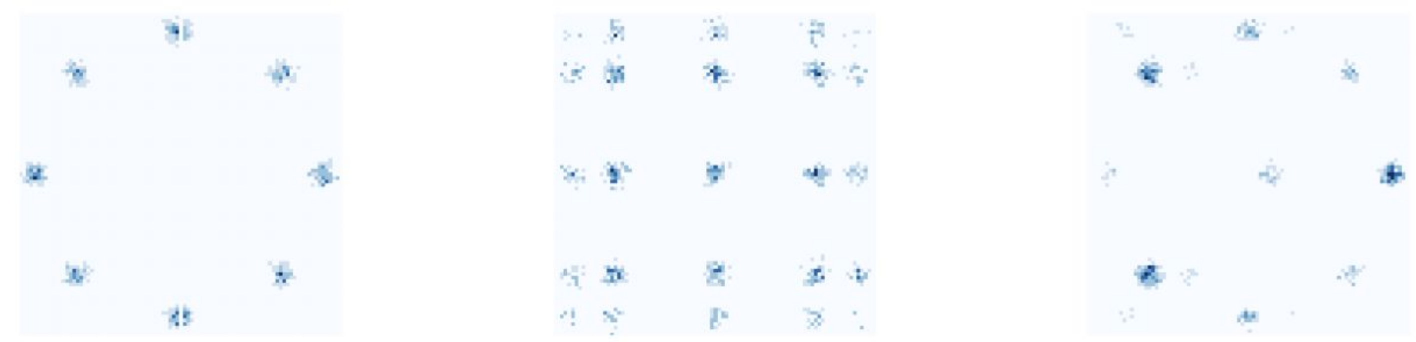
$$\left|\frac{\partial f}{\partial x}\right| = \prod \sigma_d(x_{<d})$$

Bipartite Flows.

$$y_{\geq d} = x_{\geq d} \cdot \sigma(x_{<d}) + \mu(x_{<q})$$

$$\left|\frac{\partial f}{\partial x}\right| = \prod \sigma(x_{<d})_i$$

Discrete Change of Variables



(a) Data (b) Factorized Base (c) 1 Flow

Let **x** be a discrete random variable and **y** = f(**x**) where f is some function of **x**. The induced probability mass function of **y** is:

$$p(\mathbf{y} = y) = \sum_{x \in f^{-1}(y)} p(\mathbf{x} = x)$$

For an invertible function f, this simplifies to

$$p(\mathbf{y} = y) = p(\mathbf{x} = f^{-1}(y)).$$

Discrete Flows

Use location-scale transformation on the modulo integer space:

$$\mathbf{y}_d = (\boldsymbol{\mu}_d + \boldsymbol{\sigma}_d \cdot \mathbf{x}_d) \bmod K.$$

σ_d and μ_d are autoregressive or bipartite functions of y in 0,1,...,K-1 and 1,...,K-1.

For the flow to be invertible, σ_d and K must be coprime (inverse uses Euclid's algorithm). For example: mask noninvertible values for a given K; or make K prime.

Training Discrete Flows

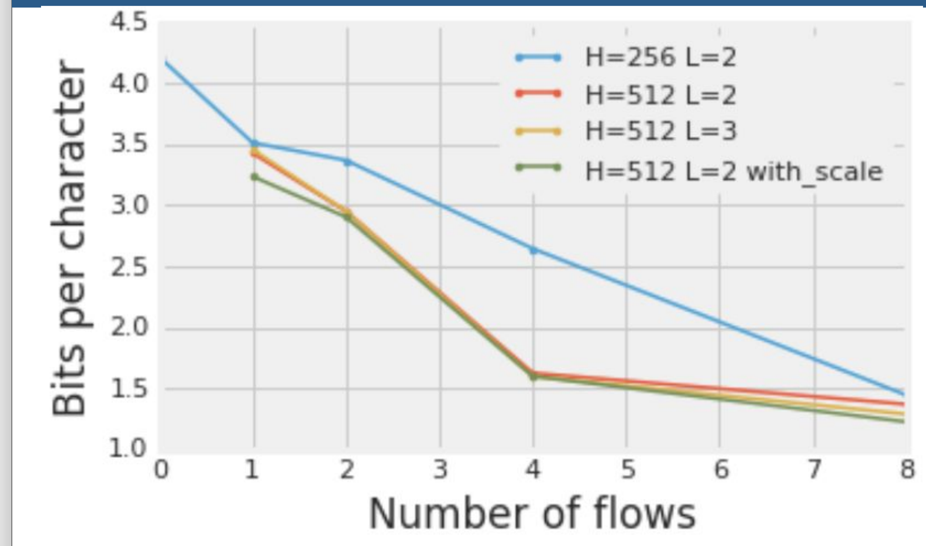
The maximum likelihood objective per datapoint is

$$\log p(\mathbf{y}) = \log p(f^{-1}(\mathbf{y})).$$

Gradient descent with respect to base distribution parameters is straightforward.

Gradient descent with respect to flow parameters requires backpropagation through the discrete-output function. We use the straight-through gradient estimator.

Experiments



Text8. LSTM per flow. Bipartite flows get best nonautoregressive results and 100x speedup.

	Test NLL (bpc)	Generation
3-layer LSTM (Merity et al., 2018)	**1.18**[1]	3.8 min
Ziegler and Rush (2019) (AF/SCF)	1.46	-
Ziegler and Rush (2019) (IAF/SCF)	1.63	-
Bipartite flow	1.38	**0.17 sec**

Penn Tree Bank. 2-3 layer Transformer per flow. Bipartite flows get best nonautoregressive results. 1000x speedup.

	bpc	Gen.
LSTM (Cooijmans+2016)	1.43	19.8s
64-layer Transformer (Al-Rfou+2018)	**1.13**	35.5s
Bipartite flow (4 flows, w/ σ)	1.60	**0.15s**
Bipartite flow (8 flows, w/o σ)	1.29	**0.16s**
Bipartite flow (8 flows, w/ σ)	1.23	**0.16s**

	Autoregressive Base	Autoregressive Flow	Factorized Base	Bipartite Flow
$D=2, K=2$	**0.9**	**0.9**	1.3	**1.0**
$D=5, K=5$	7.7	**7.6**	8.0	**7.9**
$D=5, K=10$	10.7	**10.3**	11.5	**10.7**
$D=10, K=5$	15.9	**15.7**	16.6	**16.0**

Full-Rank Discrete. Discrete autoregressive flows are best. Bipartite flows get similar performance as autoregressive base.

Limitations

- Expressivity: permutations only.
- Gradient bias over many flows and many classes.
- Devising more flexible transformations. RNG/compression?

References

"Normalizing Flows for Probabilistic Modeling and Inference." G. Papamakarios et al., 2019.

"Integer Discrete Flows and Lossless Compression." E. Hoogeboom et al., 2019.