Discrete Flows
Invertible Generative Models of Discrete Data

Dustin Tran Keyon Vafa Kumar Krishna Agrawal Laurent Dinh Ben Poole

1Google Brain 2Columbia 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.
\[ p(y) = p(x = f^{-1}(y)) \frac{\partial f^{-1}}{\partial y} \]

Autoregressive Flows.
\[ y_d = x_d \sigma_d(x < d) + \mu_d(x < d) \]
\[ \frac{\partial f}{\partial x} = \prod \sigma_d(x < d) \]

Bipartite Flows.
\[ y_{\geq d} = x_{\geq d} \sigma_d(x < d) + \mu(x < d) \]
\[ \frac{\partial f}{\partial x} = \prod \sigma(x < d) \]

Discrete Change of Variables

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:
\[ p(y = y) = \sum_{x \in f^{-1}(y)} p(x = x) \]

For an invertible function \( f \), this simplifies to
\[ p(y = y) = p(x = f^{-1}(y)) \]

Discrete Flows

Use location-scale transformation on the modulo integer space:
\[ y_d = (\mu_d + \sigma_d \cdot x_d) \mod K \]
\( \sigma_d \) and \( \mu_d \) are autoregressive or bipartite functions of \( y \) in 0,1,...,K-1 and 1,...,K-1.

For the flow to be invertible, \( \sigma_d \) 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
\[ \log p(y) = \log p(f^{-1}(y)) \]

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.

<table>
<thead>
<tr>
<th>Network</th>
<th>Test NLL (bpc)</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (Cooijmans+2016)</td>
<td>1.43</td>
<td>19.8s</td>
</tr>
<tr>
<td>64-layer Transformer (AI-Rfocs+2018)</td>
<td>1.13</td>
<td>35.5s</td>
</tr>
<tr>
<td>Bipartite flow (4 flows, w/o σ)</td>
<td>0.15s</td>
<td></td>
</tr>
<tr>
<td>Bipartite flow (8 flows, w/o σ)</td>
<td>0.16s</td>
<td></td>
</tr>
<tr>
<td>Bipartite flow (8 flows, w/ σ)</td>
<td>1.23</td>
<td>0.16s</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Autoregressive Base</th>
<th>Autoregressive Flow</th>
<th>Factorized Base</th>
<th>Bipartite Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>D = 2, K = 2</td>
<td>0.9</td>
<td>0.9</td>
<td>1.3</td>
<td>1.0</td>
</tr>
<tr>
<td>D = 5, K = 5</td>
<td>7.7</td>
<td>7.6</td>
<td>8.0</td>
<td>7.9</td>
</tr>
<tr>
<td>D = 5, K = 10</td>
<td>10.7</td>
<td>10.3</td>
<td>11.5</td>
<td>10.7</td>
</tr>
<tr>
<td>D = 10, K = 5</td>
<td>15.9</td>
<td>15.7</td>
<td>16.6</td>
<td>16.0</td>
</tr>
</tbody>
</table>

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