Latent Normalizing Flows for Discrete Sequences

Zachary M. Ziegler and Alexander M. Rush (Harvard University), ICML2019

2019年9月27-28日 最先端NLP勉強会

読んだ人：篠田 一聡 （東大相澤研）

目次

- 事前知識 3m
- 論文紹介
  - 導入 1m
  - 手法 2.5m
  - 実験・結果 2.5m
  - 結論 1m

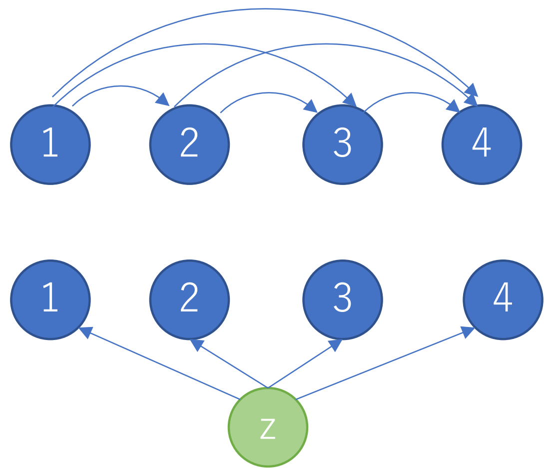
事前知識：Normalizing Flow

準備：生成モデル

- 生成モデルの定義
  - “A model is generative if it places a joint distribution over all observed dimensions of the data”
    - （IJCAI2018: A Tutorial on Deep Probabilistic Generative Modelsより）

準備：系列データの生成モデル

- 特に系列データ$\boldsymbol{x} = \boldsymbol{x}_{1:T}$の生成モデルには
  - 自己回帰(autoregressive)モデル
    - $p(x_{1:T}) = \prod_t p(x_t | x_{1:t-1})$
  - 非自己回帰(non-autoregressive)モデル
    - $p(x_{1:T}, z) = p(z) \prod_t p(x_t | z)$

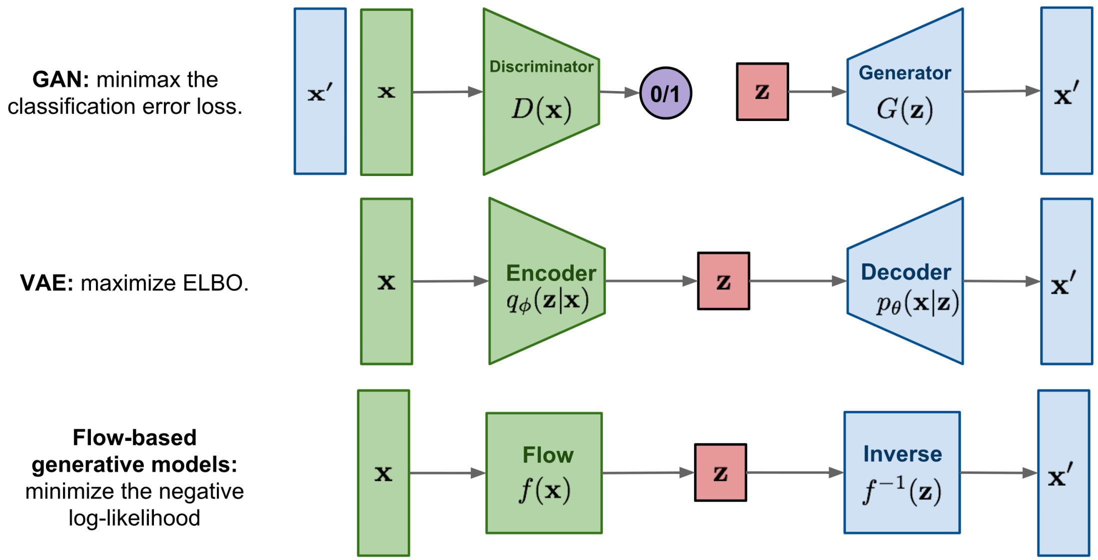


- Trade-off

	Sample quality	Generation speed
自己回帰モデル	良い	遅い
非自己回帰モデル	悪い	速い

準備：深層生成モデル

・潜在変数zから観測変数xへのマッピングの学習方法

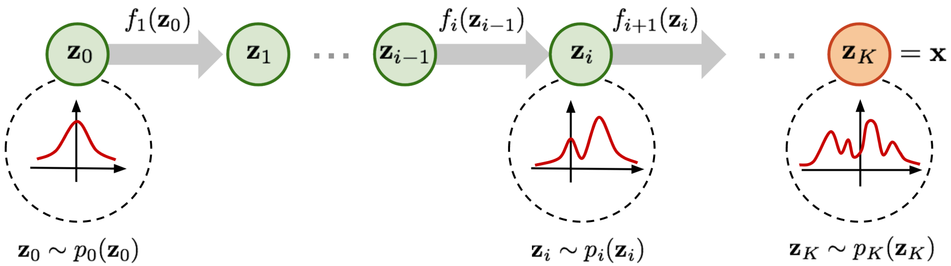


https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

・trade-offs
  ・sample quality
  ・generation speed
  ・availability of comparative metrics
  ・interpretability

Normalizing flow [Rezende & Mohamed, 2015; Kingma et al., 2016]

- **可逆**写像を複数回施して、単純な分布(p_0, base density)から複雑な分布(p_K)を得る



- 変数変換の公式（Change-of-variables formula）

$$p_z(z) = p_\epsilon(f_\theta^{-1}(z)) \left|\det \frac{\partial f_\theta^{-1}(z)}{\partial z}\right| = p_\epsilon(\epsilon) \left|\det \frac{\partial \epsilon}{\partial z}\right|$$

z, ϵ	確率変数
p_z, p_ϵ	確率密度関数
$f_\theta: \epsilon \to z$	可逆写像（f_θ^{-1}が存在する）

Normalizing Flow

- $z_i = f_i(z_{i-1})$
- $z_K(= x) = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$

- f_i をflow
- $f_K \circ f_{K-1} \circ \cdots \circ f_1$をnormalizing flow
- と呼ぶ

訓練・推論

- 訓練：最尤推定（変数変換の公式を繰り返し適用）

$$
\begin{aligned}
\log p_K(z_K) &= \log p_{K-1}(z_{K-1}) - \log \left|\det \frac{\partial f_K}{\partial z_{K-1}}\right| \\
&= \log p_{K-2}(z_{K-2}) - \log \left|\det \frac{\partial f_{K-1}}{\partial z_{K-2}}\right| - \log \left|\det \frac{\partial f_K}{\partial z_{K-1}}\right| \\
&= \log p_0(z_0)
\end{aligned}
$$

- 推

$$
\begin{aligned}
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log\left|\det \frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right| - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|
\end{aligned}
$$

論文紹介

Introduction

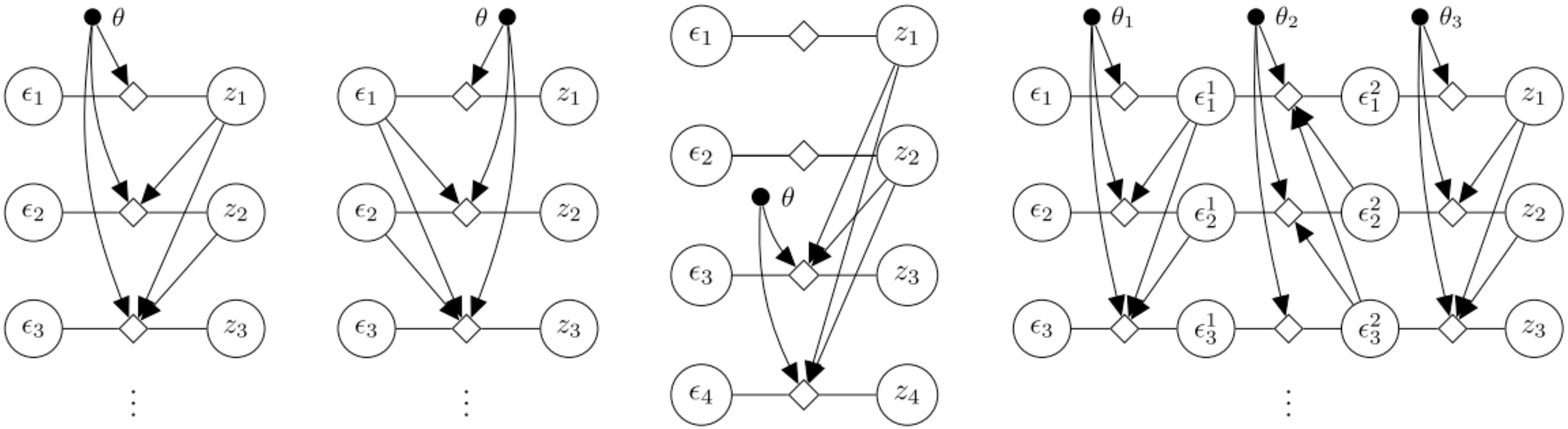
- Normalizing flowは連続変数（画像, 音声）の生成モデルとして強力、非自己回帰的な生成も可能
- しかし、flowの離散変数（文）への適用は難しいので
- 離散変数と潜在変数間のマッピングにVAEベースの生成モデルを適用
- 潜在変数の分布の学習にflowを利用

Introduction

- Normalizing flowは連続変数の同時分布を表現する
- 1) 直接観測変数の生成モデルとして使われる
- 2) VAEにおいて潜在変数の事後分布

- 利点：
  - Model flexibility
  - Generation speed
  - 非自己回帰的にもできる

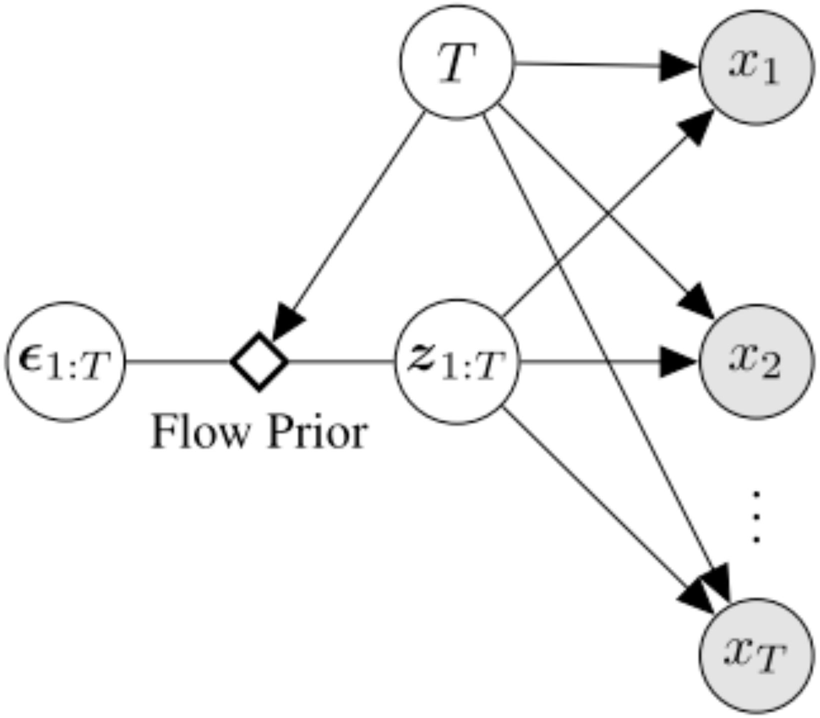
Flow



(a) AF ($\leftarrow$) (b) IAF ($\rightarrow$) (c) SCF ($\leftrightarrow$) (d) 3-layer AF ($\leftarrow$)

各ノードは実数

Method



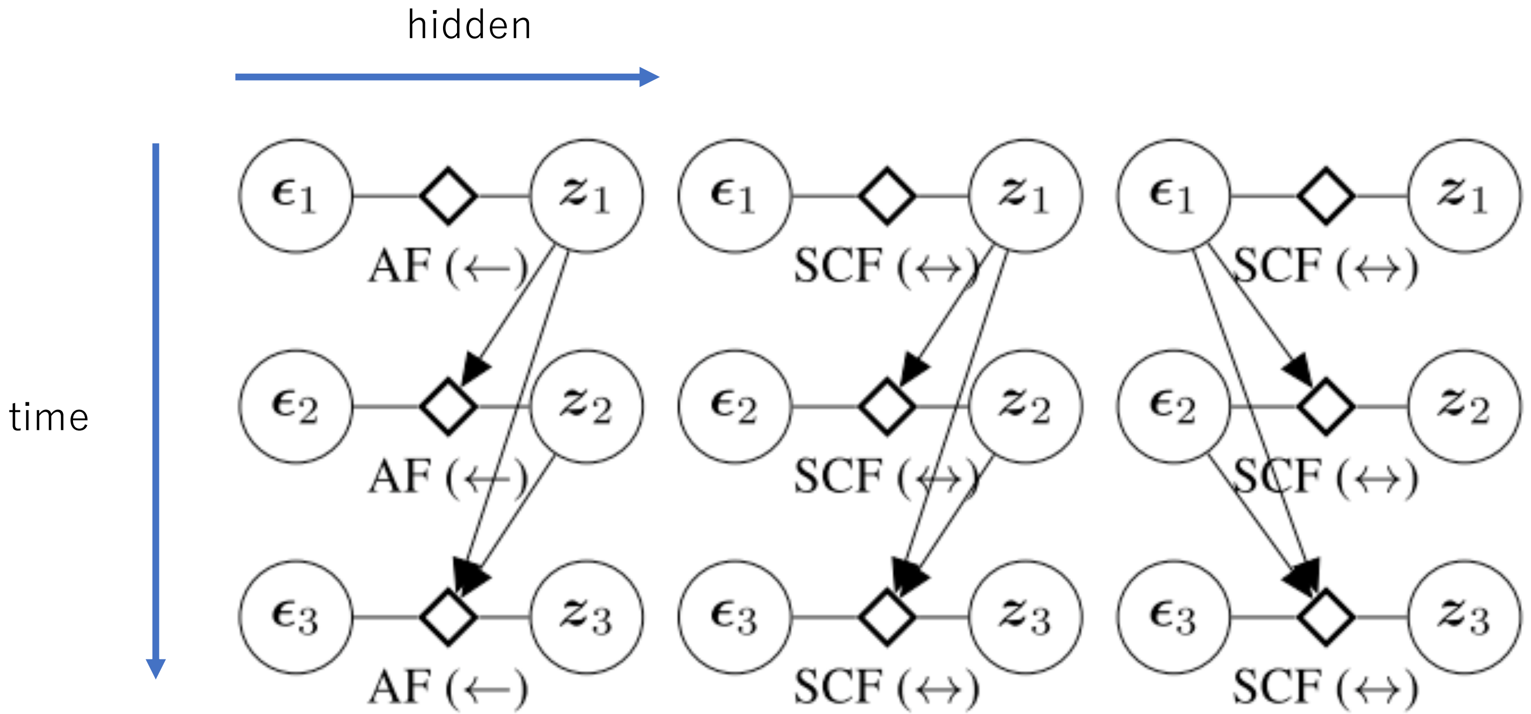
1. 系列長Tをサンプリング

2. ϵをサンプリング

3. zをサンプリング

4. x_tを個別にサンプリング

Method



(a) AF / AF (←) (b) AF / SCF (←) (c) IAF / SCF (→)

(time / hidden)

Experiment 1:

Result & Discussion

Conclusion

- 自己回帰的なflow-based modelの精度は、baselineに匹敵するものだった
- 非自己回帰的なflow-based modelは生成速度を向上させたが、精度は悪くなる

参考リンク

- Flow-based Deep Generative Models
  - https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html
- Normalizing Flows Tutorial
  - https://blog.evjang.com/2018/01/nf1.html
- [DL輪読会]Flow-based Deep Generative Models
  - https://www.slideshare.net/DeepLearningJP2016/dlflowbased-deep-generative-models

以下、参考スライド

文字レベルで見ると、分布が複雑

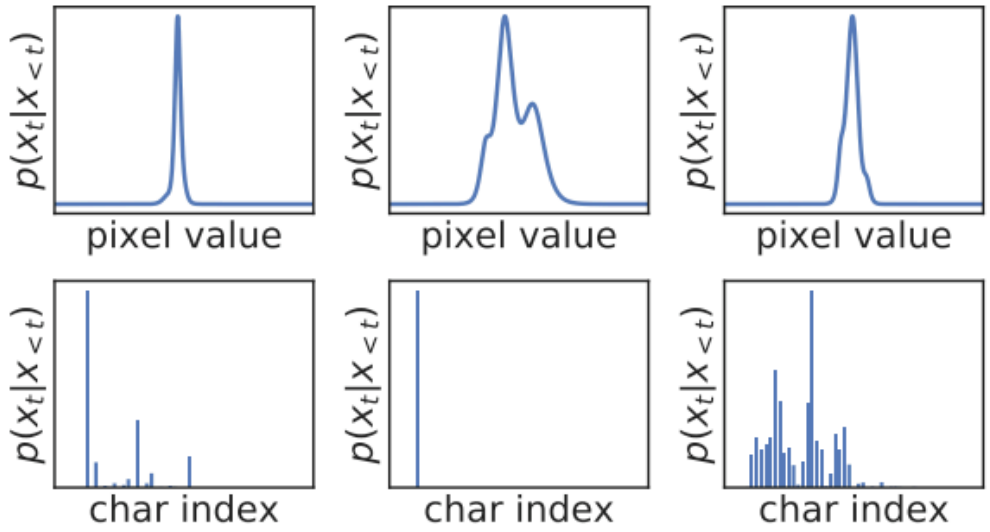


Figure 3. Example conditional distributions $p(x_t|\boldsymbol{x}_{<t})$ from continuous (PixelCNN++, 10 mixture components, trained on CIFAR-10, top) and discrete (LSTM char-level LM trained on PTB, bottom) autoregressive models.

- あ
- い