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 D1）

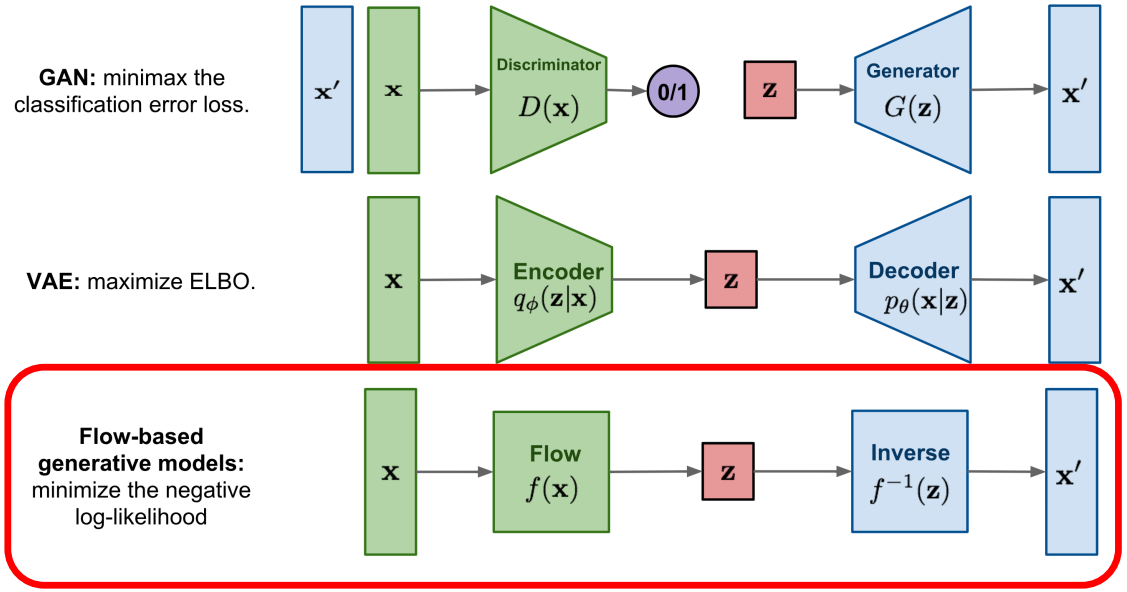
目次

- 事前知識：Normalizing Flow
- 論文紹介

事前知識：Normalizing Flow

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



出典：[1]

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

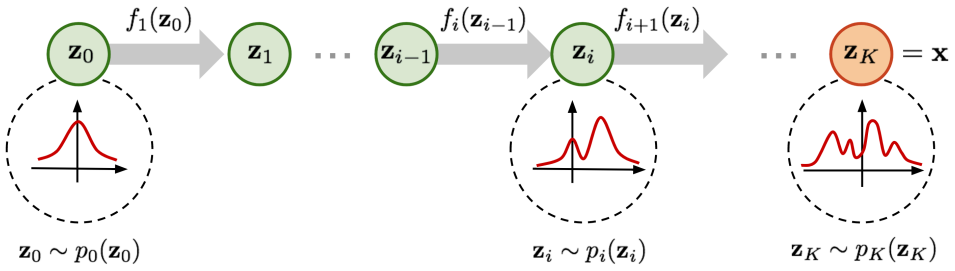
Normalizing Flow for NLP

- Unsupervised Learning of Syntactic Structure with Invertible Neural Projections (EMNLP2018)
- Riemannian Normalizing Flow on Variational Wasserstein Autoencoder for Text Modeling (NAACL2019)
- Unsupervised Learning of PCFGs with Normalizing Flow (ACL2019)
- FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow (EMNLP2019)
- 本論文
- …?

(acl anthologyで検索)

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



出典：[1]

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

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

$z,\ \epsilon$	確率変数
$p_z,\ p_\epsilon$	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

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

と呼ぶ

NFの訓練・生成

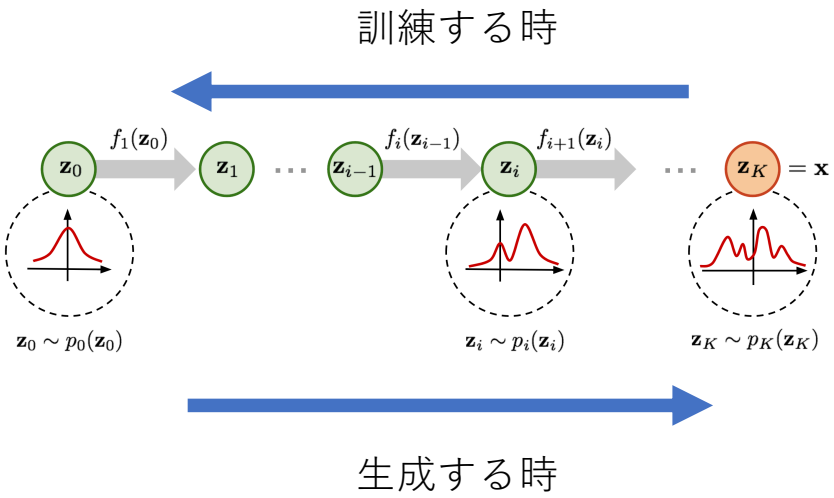
- 訓練：最尤推定（変数変換の公式を繰り返し適用）

$$\log p_K(z_K) = \log p_{K-1}(z_{K-1}) - \log \left|\det \frac{\partial f_K}{\partial z_{K-1}}\right|$$

$$= \log p_{K-2}(z_{K-2}) - \log \left|\det \frac{\partial f_{K-1}}{\partial z_{K-2}}\right| - \log \left|\det \frac{\partial f_K}{\partial z_{K-1}}\right|$$

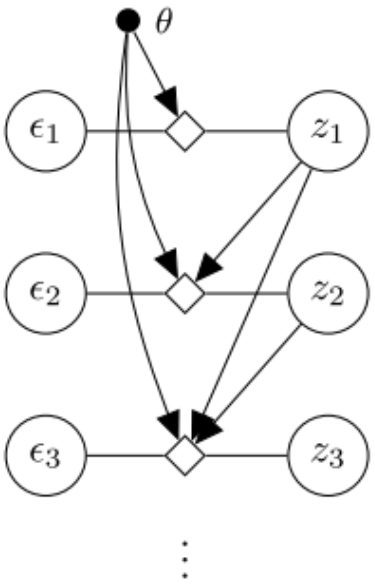
$$= \log p_0(z_0) - \sum_i \log \left|\det \frac{\partial f_i}{\partial z_{i-1}}\right|$$

可逆変換のおかげで対数尤度を直接計算できる！



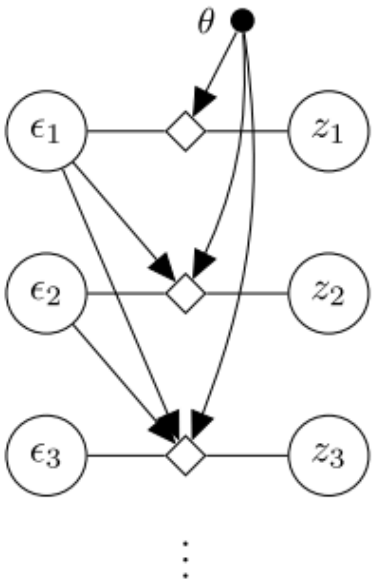
- 生成：z_0をサンプリング
  → $f_K \circ f_{K-1} \circ \cdots \circ f_1$で変換して$x$を得る

色々なFlowがある（各ノードは実数）



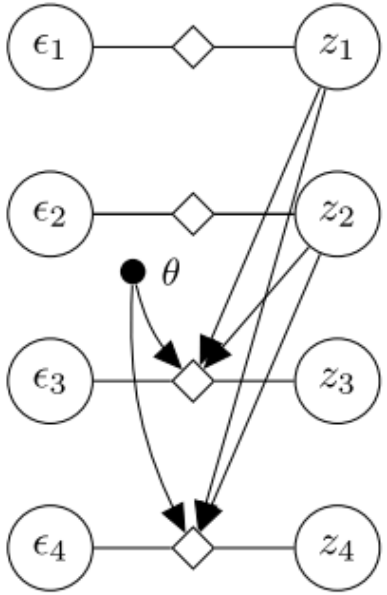
(a) AF (←)

Autoregressive Flow
(Papamakarios et al., 2017)



(b) IAF (→)

Inverse Autoregressive Flow
(Kingma et al., 2016)



(c) SCF (↔)

Split Coupling Flow
(Dinh et al., 2017)

菱形◇は
アフィン変換
（可逆）

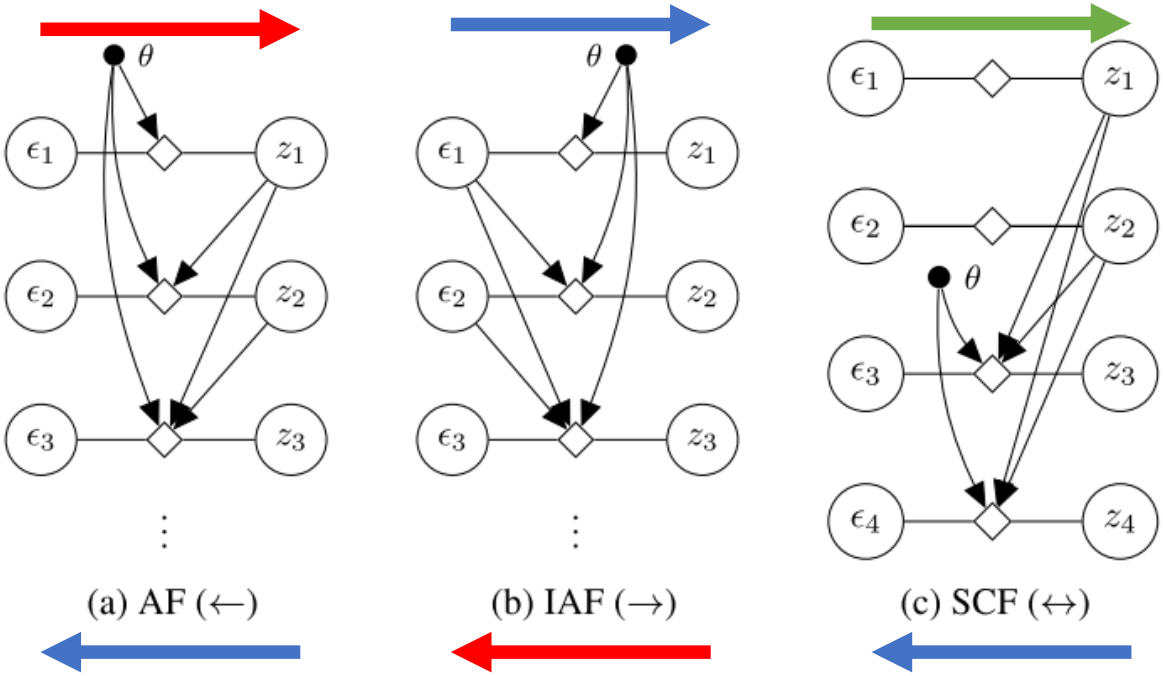
ヤコビアンが計算
しやすいような設計

Trade-off

自己回帰・遅い

非・速

自？



(a) AF (←)

(b) IAF (→)

(c) SCF (↔)

非自己回帰・速い

自・遅

非・速

SCF:

$$\epsilon_1 = z_1$$
$$\epsilon_2 = z_2$$

とすることでAFに比べて
生成の計算効率を向上させた
が、modeling flexibilityは悪化

論文紹介：Latent Normalizing Flows for Discrete Sequences

Introduction

- Normalizing flow (NF)は**複雑な分布をモデル化**することができ、連続変数（画像, 音声）の生成モデルとして強力

- 生成を非自己回帰にして**速度を向上させることも可能**
  - van den Oord et al., “Parallel WaveNet: Fast High-Fidelity Speech Synthesis”
  - Kingma et al., “Glow : Generative Flow with Invertible 1x1 Convolutions”

- 離散系列にもNFを適用したい
- しかし、NFだけを使って離散変数系列への適用は難しい

- VAEとNFを使うことで離散変数の系列データの非自己回帰な生成モデルを提案

(Supplementary Materials)

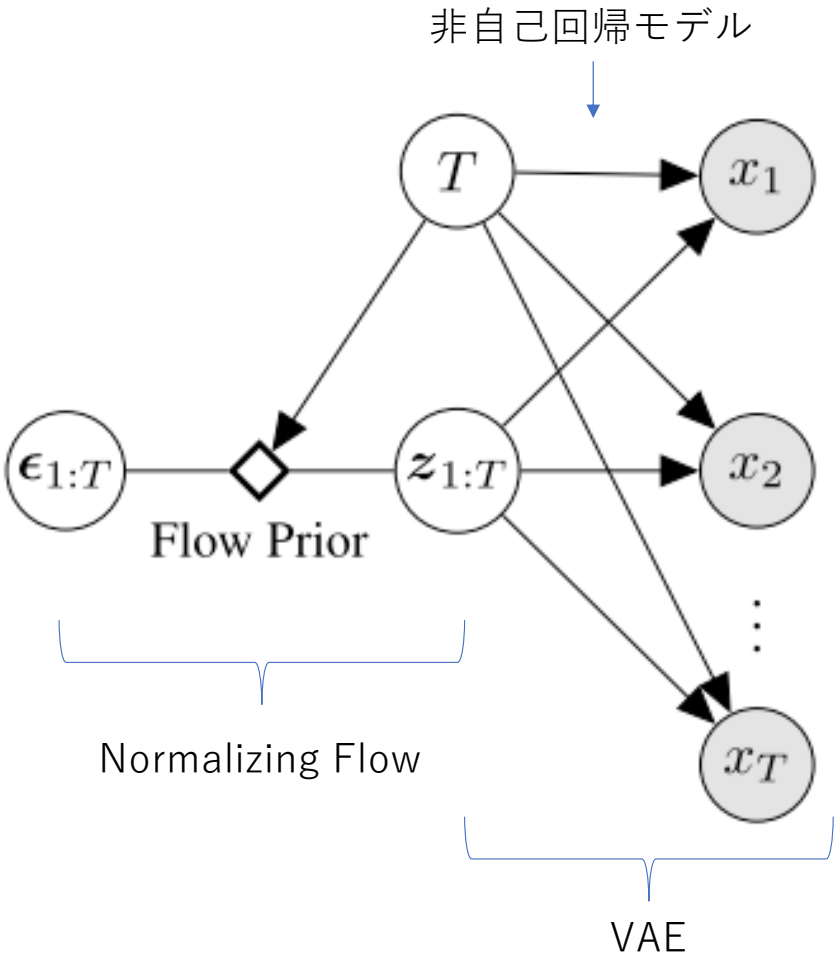
- 一次元の離散変数同士の可逆変換はpermutationのみなので、NFのように複雑な分布を表現できない（と主張）

- 多次元の場合、permutation以外にも可逆変換は存在
- 例えばXOR（=Future work）

a=0011　　a=0011
b=0110　　c=0101
↓　　↓
c=0101　　b=0110

連続変数を介することなく変換できるかも・・・？

Method



生成手順

1. 系列長Tをサンプリング※

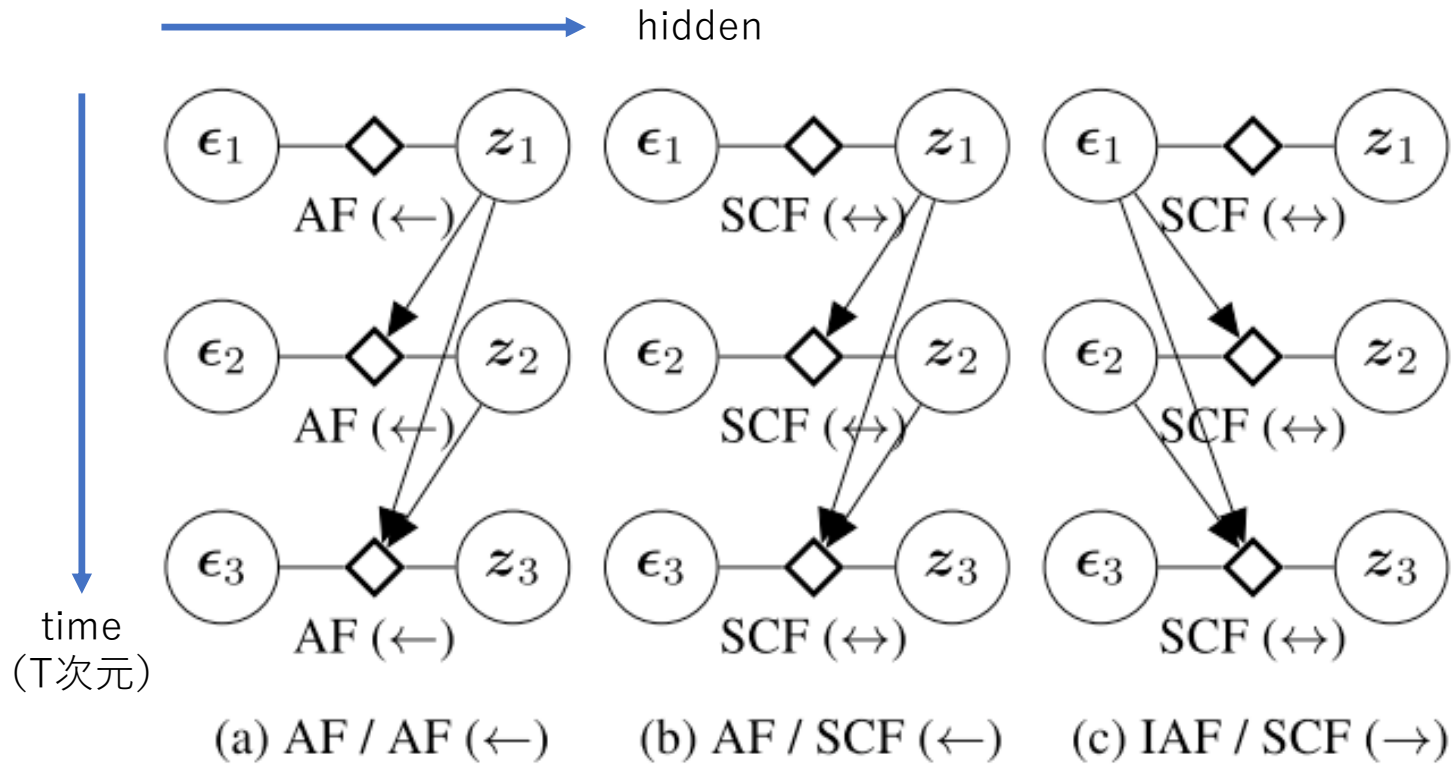
2. ϵをサンプリング

3. zをサンプリング

4. x_tを個別にサンプリング

※系列長Tは経験に基づいて決める
“For unconditional sequence modeling we can use the empirical likelihood for p(T)”

色々なflowの組み合わせを試した（各ノードはH次元ベクトル）



（ time方向 / hidden方向 ）

Extension: Non-Linear Squared Flow

- Huang et al. (2018) に倣って、モデルのflexibilityをさらに向上させるために、flowの最小単位をアフィン変換からNon-Linear Squared Flow (NLSq) に変える

アフィン変換　$f(\epsilon) = z = a + b\epsilon$

NLSq　$f(\epsilon) = z = a + b\epsilon + \dfrac{c}{1 + (d\epsilon + g)^2}$

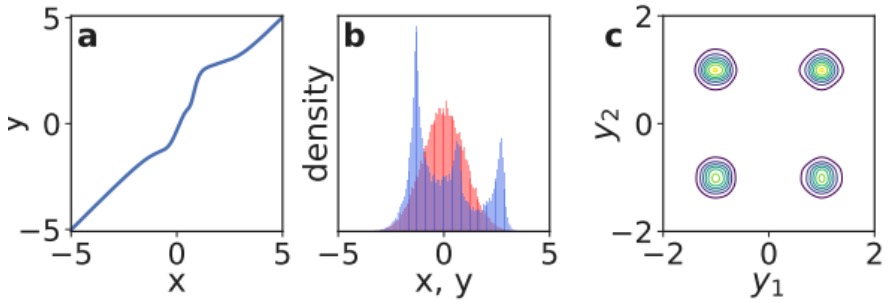


Figure 5. Non-Linear Squared (NLSq) flow for multimodal distribution modeling. **(a, b)** NLSq transformation defined by hand-selecting 4 layers of flow parameters, **(a)** composed transformation, **(b)** base density (red), final density (blue). **(c)** Resulting density for learned 2D transformation via 5 layer AF-like using the NLSq flow from a standard Gaussian to a Gaussian mixture distribution.

Experiment

- 1. Character-level language modeling
  - Dataset: Penn Treebank
  - Vocab size: 51
  - Dataset size: 5M characters
  - カテゴリカル分布

- 2. Polyphonic music modeling
  - Vocab size: 2^{88}
    - 88次元のbinary vectorで表現
  - ベルヌーイ分布

（88はピアノの鍵盤の数
ある時刻に複数の鍵盤が押されることがある）

Exp 1: Character-level language modeling

Table 1. Character-level language modeling results on PTB. NLL for generative models is estimated with importance sampling using 50 samples[2], the reconstruction term and KL term refer to the two components of the ELBO. The LSTM from Cooijmans et al. (2017) and AWD-LSTM from Merity et al. (2018) use the standard character-setup which crosses sentence boundaries[3].

Model	Test NLL (bpc)	Reconst. (bpc)	KL (bpc)
LSTM	1.38	-	-
AWD-LSTM	1.18	-	-
LSTM (sentence-wise)	1.41	-	-
AF-only	2.90	0.15	2.77
AF / AF	1.42	0.10	1.37
AF / SCF	1.46	0.10	1.43
IAF / SCF	1.63	0.21	1.55

(time / hidden)

ここから言えること
・AF / AFはLSTM (baseline)とほぼ同じNLL

・AF is better than IAFは画像分野でも同じ傾向
　・非自己回帰サンプリングは精度に難あり

・KL項がNLLのほとんどを占めている(90%)
　・flowを使わないモデルでは5~30%

Exp 1: Character-level language modeling

Table 2. Ablation experiments. AF / AF is the same result as in Table 1. -NLSq indicates the affine transformation is used instead of NLSq. -AF hidden indicates no dependencies across hidden (an independent vector affine transformation is used instead).

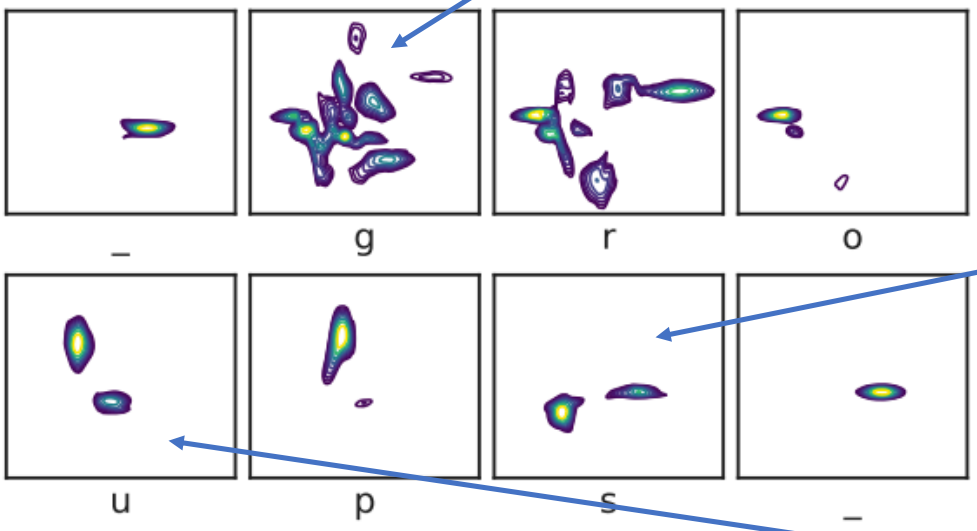
Model	Test NLL (bpc)	Reconst. (bpc)	KL (bpc)
AF / AF	1.42	0.10	1.37
- NLSq	1.50	0.11	1.51
- AF hidden	1.57	0.14	1.57
- AF hidden and NLSq	1.56	0.29	1.56

ここから言えること
・hidden方向のdependencyは大事

・アフィン変換よりも非線形なNLSqが良い

Exp 1: Character-level language modeling

区切り文字”_”の後は色々な文字が来る可能性がある



“group”の後は”_”か”s”の２通り
(単数形or複数形)

時刻とともにmodeが減っている

“gro”の後は”u”か”w”の２通り？

Figure 6. Conditional prior densities corresponding to characters in the string ‘_groups_’ (_ indicates a space), from top left to bottom right. Each figure shows $p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ for increasing t, where $\boldsymbol{z}_{1:T}$ is sampled from $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$ and $\boldsymbol{x}_{1:T}$ comes from validation.

Exp 2: Polyphonic music modeling

Model	Nottingham	Piano	Musedata	JSB
RNN-NADE	2.31	7.05	5.6	5.19
TSBN	3.67	7.89	6.81	7.48
STORN	2.85	**7.13**	**6.16**	6.91
NASMC	2.72	7.61	6.89	**3.99**
SRNN	2.94	8.2	6.28	4.74
DMM	2.77	7.83	6.83	6.39
LSTM	3.43	7.77	7.23	8.17
AF / AF	**2.39**	8.19	6.92	6.53
AF / SCF	2.56	8.26	6.95	6.64
IAF / SCF	2.54	8.25	7.06	6.59

時間方向にも88 notes間にもdependencyを仮定

潜在変数を使うモデル
88 notesは条件付き独立と仮定

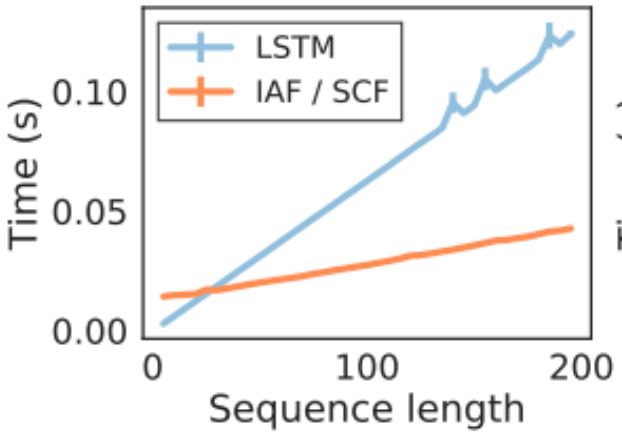
Piano データセットは最も系列が長い
提案手法のinductive biasには
matchしなかった

潜在変数を使うモデルの中では一番良い

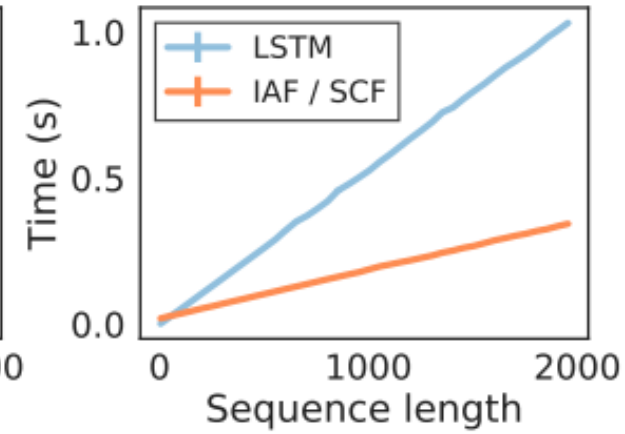
総括すると、提案手法は少なくとも
既存の潜在変数を使うモデル（２段目）
と同じくらい良い

Generation speed

1. Character-level language modeling



2. Polyphonic music modeling



生成の速度は、想定通り
自己回帰モデル＜非自己回帰モデル

系列長が大きくなると差が顕著

Conclusion, Discussion

- normalizing flowの導入により非自己回帰モデルで精度を大きく落とさずに生成速度を向上

- flowの種類よって結果は違う
  - 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はさらに生成速度を向上させたが、精度は悪くなる

参考

- [1] Flow-based Deep Generative Models
  - https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html
- [2] Normalizing Flows Tutorial
  - https://blog.evjang.com/2018/01/nf1.html

以下、参考スライド

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)

要は

$$x \sim p(x)$$

$$x \sim p(x|z)$$

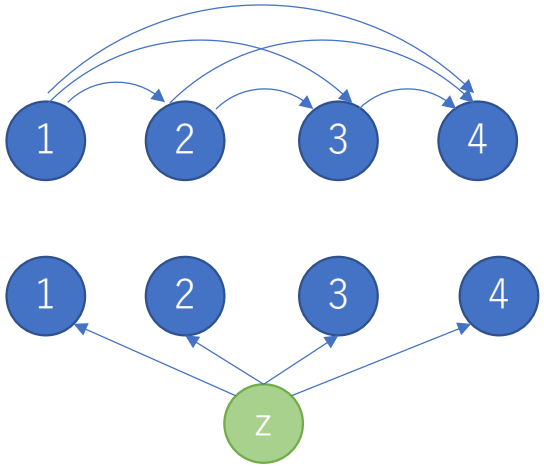
$$x \sim p(x|\theta)$$

生成モデル

x: 観測変数
（画像、言語、…）

準備：系列データの生成モデル

- 特に系列データ$x_{1:T} = \{x_1, x_2, \ldots, x_T\}$の生成モデルは大きく分けて
  - 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

文字は画像よりも分布が複雑

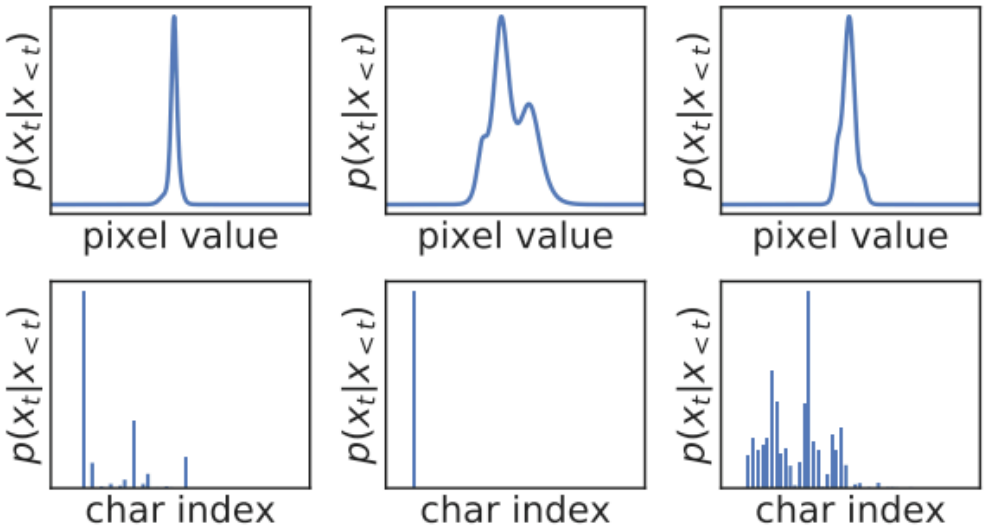


Figure 3. Example conditional distributions $p(x_t|\boldsymbol{x}_{<t})$ from continuous (PixelCNN++, 10 mixture components, trained on CIFAR-10, top) and discrete (LSTM char-level LM trained on PTB, bottom) autoregressive models.

自己回帰モデルで計算した条件確率分布は
画像は単峰(unimodal)だが、
文字は多峰(multi-modal)

これを非自己回帰モデルで表現するためには、
この複雑さを潜在空間内で表現する必要がある