生成モデルの Deep Learning

PFI セミナー　得居 誠也

Preferred Networks, Inc.

2016/02/25

自己紹介

- 得居 誠也 (Seiya Tokui)
- beam2d (Twitter, GitHub)

- 略歴
  - PFIアルバイト(2010.10)→PFI(2012.04)→PFN(2014.10)
  - 東大情報理工数理 (2010.04-2012.03)：近傍探索の機械学習
  - 4 月から社会人博士課程に入る予定
- 主な仕事
  - Chainer
  - 深層学習まわりをやっています

生成モデル

- 訓練データを生成する確率分布 $p(\mathbf{x})$ を推定したい

- 主な目的
  - 新しいデータを生成したい ← 今日はこちら
  - 半教師あり学習につかいたい
  - 事前学習につかいたい（最近は下火）

データの生成器

- 目標：自然なデータを生成したい
  - 画像：写真、絵
    （今日はおもに画像生成の話をします）
  - 自然言語：文章、会話文

- モチベーション
  - 創作活動の補助
  - 人に対するインターフェイスの提供（とくに自然文生成）
  - データ作成コストの削減（「シミュレータ」としての使用）

問題設定

- サンプリング可能なモデル $p_\theta(\mathbf{x})$ (generator)
- この θ を訓練データから学習して、 $p_\theta(\mathbf{x})$ がデータの生成分布 $p^\star(\mathbf{x})$ に合うようにしたい

- 難しいポイント
  - $p_\theta(\mathbf{x})$ はサンプリングできる形になっていないといけない。
    このとき、普通は密度 $p_\theta(\mathbf{x})$ の値は計算できない。
  - 訓練データは一般に有限個しかない。 $p^\star(\mathbf{x})$ の推定には自由度がある。
  - 多くの応用で、サンプリングは低コストな方が望ましい。
    表現力の高いモデル $p_\theta(\mathbf{x})$ は、サンプリングの計算コストが高くなりがち。
    例：MCMC vs. 伝承サンプリング
    - 今日は主に伝承サンプリングで済む手法のみを紹介

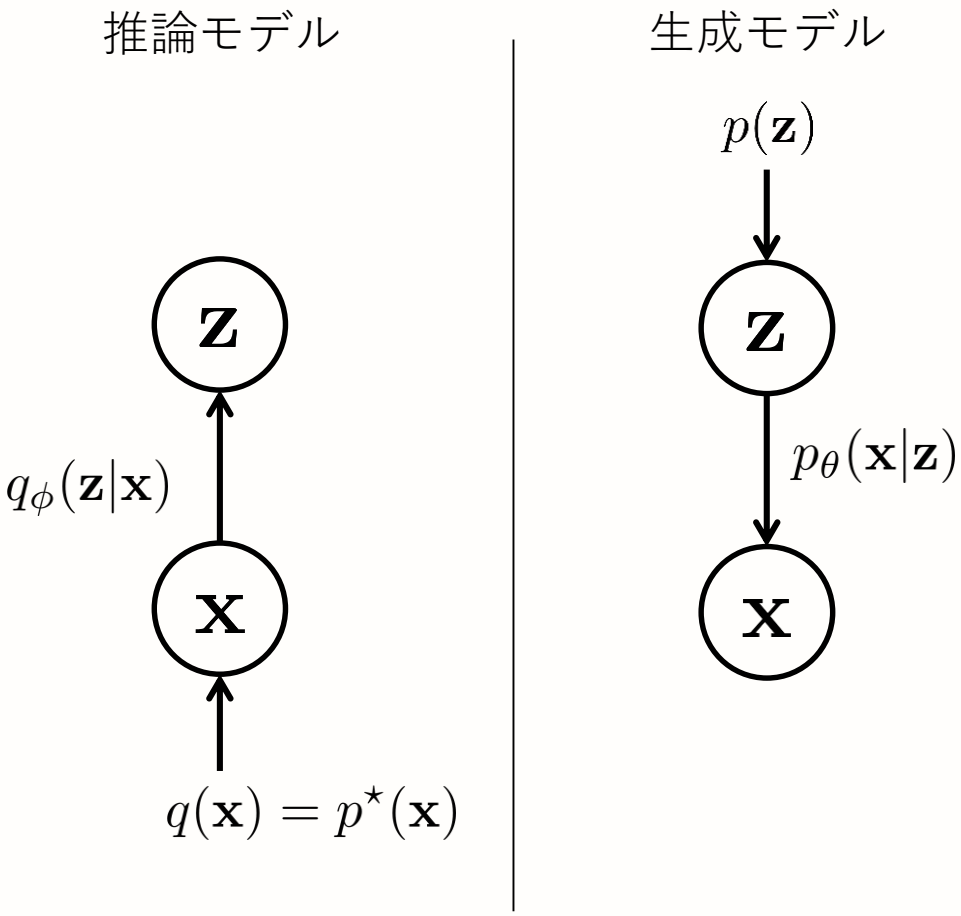
今日のおはなし

- データ生成の問題に対する深層学習でのアプローチを紹介します
- 大きく３つのトピックからなります
  - Helmholtz Machine
  - Generative Adversarial Nets
  - 生成モデルと意思決定

Helmholtz Machine

Helmholtz Machine [Dayan+, ’95]

- Generator：潜在変数 $\mathbf{z}$ から伝承サンプリングで $\mathbf{x}$ を生成
- 逆向きの推論をする**推論モデル**を一緒に学習する



Helmholtz Machine の目的関数：変分下界 $\mathcal{L}$

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \\
\text{（イェンセンの不等式）} \quad &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} =: \mathcal{L}
\end{aligned}
$$

不等号の両辺の差は $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))$ で、 $\mathcal{L}$ を最大化すればこの差は最小化される

変分下界最大化 1：Wake-Sleep [Hinton+, ’95]

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}, \mathbf{z}) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log q_\phi(\mathbf{z}|\mathbf{x})$$

ϕ についてのこの量の最大化は、$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))$ を最小化するのと等価。

これを最小化する代わりに、分布をひっくり返した $D_{\mathrm{KL}}(p_\theta(\mathbf{z}|\mathbf{x}) \| q_\phi(\mathbf{z}|\mathbf{x}))$ を最小化する。

これは ϕ について $\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})} \log q_\phi(\mathbf{z}|\mathbf{x})$ を最大化するのと等価。

$$\tilde{\mathcal{L}} = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}, \mathbf{z})}_{\substack{\text{Wake phase} \\ \mathbf{x}\text{ はデータセット} \\ \text{からサンプリング} \\ \text{(real data)}}} + \underbrace{\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})} \log q_\phi(\mathbf{z}|\mathbf{x})}_{\substack{\text{Sleep phase} \\ \mathbf{x}\text{ はモデルから} \\ \text{サンプリング} \\ \text{(fantasy)}}}$$

変分下界最大化 2：Reweighted Wake-Sleep [Bornschein+, '15]

$q_\phi(\mathbf{z}|\mathbf{x})$ を $p_\theta(\mathbf{z}|\mathbf{x})$ により正しく近づけたい。

$$p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \approx \frac{1}{K}\sum_{k=1}^{K}\underbrace{\frac{p_\theta(\mathbf{x},\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x})}}_{=:\,\omega_k} \quad (\mathbf{z}^{(k)} \sim q_\phi(\mathbf{z}|\mathbf{x}), \forall k)$$

これを使って勾配を近似。

Wake phase
$\mathbf{x}$ はデータセットからサンプリング

$$\nabla_\theta \log p_\theta(\mathbf{x}) = \frac{1}{p_\theta(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})\right]$$
$$\approx \frac{1}{\frac{1}{K}\sum_k \omega_k}\frac{1}{K}\sum_k \omega_k \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}^{(k)})$$
$$= \sum_k \tilde{\omega}_k \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}^{(k)}) \quad \left(\tilde{\omega}_k = \frac{\omega_k}{\sum_k \omega_k}\right)$$

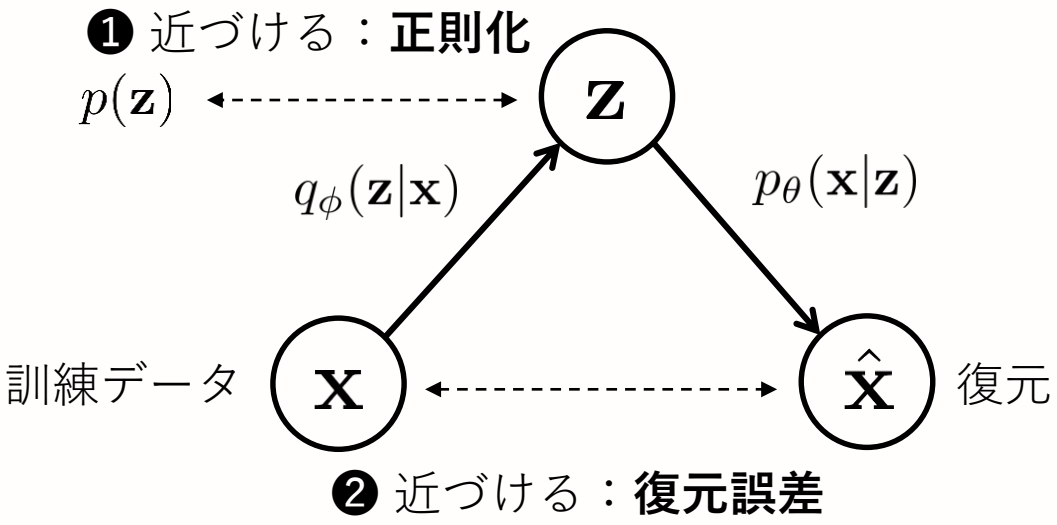
$$-\nabla_\phi D_{\mathrm{KL}}(p_\theta(\mathbf{z}|\mathbf{x})\|q_\phi(\mathbf{z}|\mathbf{x})) = \frac{1}{p_\theta(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x})\right]$$
$$\approx \sum_k \tilde{\omega}_k \nabla_\theta \log q_\phi(\mathbf{z}^{(k)}|\mathbf{x})$$

Sleep phase は Wake-Sleep と同じ

変分下界最大化 3：変分 AutoEncoder [Kingma+, ‘14][Rezende+, ‘14]

変分下界を違う形に変形

$$
\begin{aligned}
-\mathcal{L} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{z})} \\
&= \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{❶}} - \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{❷}}
\end{aligned}
$$

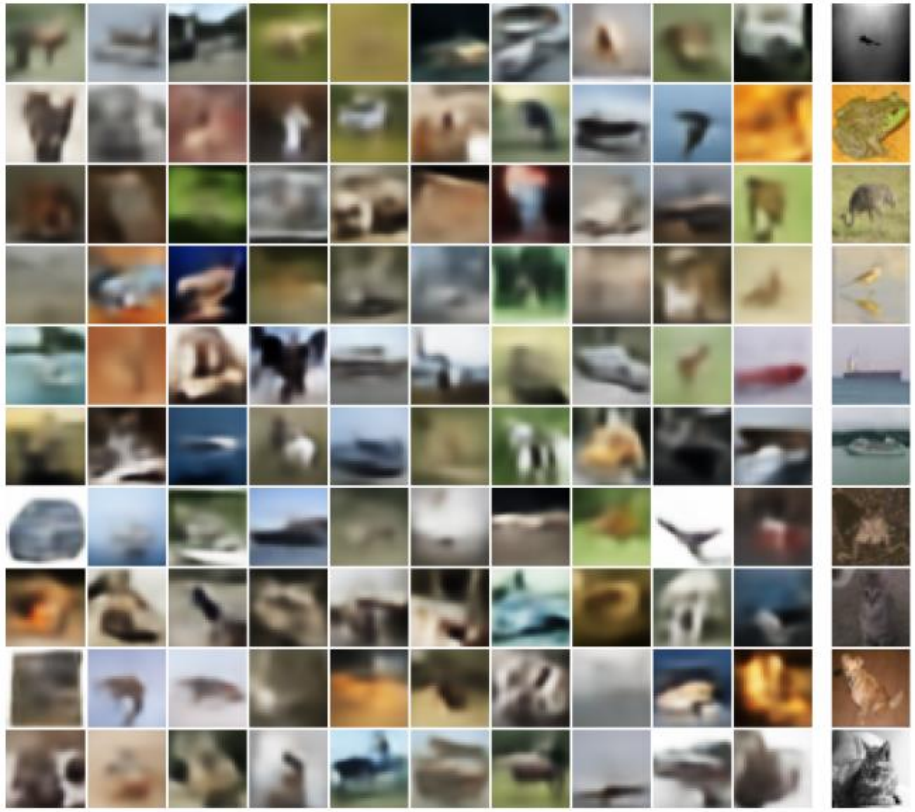


各手法のいいところ・わるいところ

- Wake-Sleep, Reweighted Wake-Sleep
  - 潜在変数 $\mathbf{z}$ が多層になっていても・離散変数でも学習できる
  - 勾配は不偏推定でないので、学習がうまくいくかは問題による
    （Reweighted WS に関してはサンプル数 K を大きくすれば正しい勾配に近づく）
- 変分 AutoEncoder
  - 潜在変数 $\mathbf{z}$ がガウシアンの場合、復元誤差の ϕ に関する勾配が低バリアンスで推定できる（Reparameterization Trick）
    - $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \quad (\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}))$ という変数変換により、$\mathbf{z}$ に関する勾配を直接 $\boldsymbol{\mu}, \boldsymbol{\sigma}$ に逆伝播できる
  - $\mathbf{z}$ が高次元ガウシアンの場合に、多くの次元が正則化で「潰れて」しまうことが多い
    （$q_\phi(\mathbf{z}|\mathbf{x})$ が $p(\mathbf{z})$ に縮退してしまう）
- どちらの手法も、伝承サンプリングで現れる条件付き分布の密度とその勾配が計算できないといけない

条件付き尤度に勾配が計算可能なベルヌーイ分布やガウス分布を使う場合の限界

- データが低次元多様体に密集しているという仮説がある
- この仮説が正しい場合、多様体に沿うずれと直交するずれは区別して評価できないといけない
- $p_\theta(\mathbf{x}|\mathbf{z})$ がベルヌーイやガウスなど単純な分布だと、これらを区別できない
- 例：画像の場合、平行移動は多様体に沿う（ユークリッド距離では大きな）ずれ、ガウスぼかしは多様体に直交する（ユークリッド距離では小さな）ずれ
- → ぼけた画像を生成しやすくなる



K. Gregor, I. Danihelka, A. Graves, D. Wierstra.
DRAW: A Recurrent Neural Network For Image Generation.
ICML 2015.

Helmholtz Machine のまとめ

- Helmoholtz Machine は、生成モデルと推論モデルを一緒に学習する枠組み
- 変分下界を最大化するように各モデルを学習する
- Wake-Sleep では、推論モデルで実データから推論した潜在変数に生成モデルをマッチさせ (wake phase)、生成モデルがつくった仮想データに推論モデルをマッチさせる (sleep phase)
- Wake-Sleep は推論モデルの学習におけるバイアスが大きいが、Reweighted Wake-Sleep はモンテカルロ近似によってこれを改善する
- 変分 AutoEncoder はガウス潜在変数をつかった場合に勾配を低バリアンスで推定でき、効率的に学習できる

Generative Adversarial Nets

尤度関数を陽にモデル化せずにデータを生成したい

- いままでは、$p_\theta(\mathbf{x}|\mathbf{z})$（の勾配）が計算できないと学習できなかった
- そのためには $p_\theta(\mathbf{x}|\mathbf{z})$ としてベルヌーイ分布やガウス分布など、単純な分布を使う必要がある
  - 平均値や標準偏差をニューラルネットなど複雑なモデルで推定することはできる
  - つまり、条件付けの部分のモデル化は自由
  - しかし、最終的に予測できる分布は計算可能な単純なものでないといけなかった

- 分布の形状自体をニューラルネットで推定できないか？
  - $p_\theta(\mathbf{x}|\mathbf{z})$ 自体をニューラルネットで推定しようとすると、正規化が困難という問題が出てくる（分配関数が計算できない）
  - 正規化されていない量をつかって分布をマッチングできないか

Discriminator による密度比推定

- 「どれくらい真のデータっぽいか」をニューラルネットで測る
- 確率変数 $y \in \{0, 1\}$ を用意して、$\mathbf{x}$ の条件付き分布を次のように定める

$$p(\mathbf{x}|y) = \begin{cases} p_\theta(\mathbf{x}) & \text{if} \quad y = 0 \\ p^\star(\mathbf{x}) & \text{if} \quad y = 1 \end{cases}$$

- 簡単のため $p(y = 0) = p(y = 1) = 1/2$ とする（つまり $p^\star(\mathbf{x})$ と $p_\theta(\mathbf{x})$ のどちらからサンプリングするかは等確率に決める）
- このとき、$p^\star(\mathbf{x})$ と $p_\theta(\mathbf{x})$ の密度比は、ベイズの定理より

$$\frac{p^\star(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(y=0)}{p(y=0|\mathbf{x})p(y=1)} = \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})}$$

- つまり、データが $p^\star(\mathbf{x})$ と $p_\theta(\mathbf{x})$ のどちらから来たかを予測する識別モデル $\underbrace{D(\mathbf{x}) = p(y=1|\mathbf{x})}_{\text{Discriminator}}$ を学習できれば、密度比が推定できる

Discriminator によるダイバージェンス推定

- Generative Adversarial Nets では Discriminator を用いて Jensen-Shannon ダイバージェンスを推定する。 $\bar{p}(\mathbf{x}) = (p^\star(\mathbf{x}) + p_\theta(\mathbf{x}))/2$ とおいて、

$$
\begin{aligned}
2D_{\mathrm{JS}} &= D_{\mathrm{KL}}(p^\star(\mathbf{x}) \| \bar{p}(\mathbf{x})) + D_{\mathrm{KL}}(p_\theta(\mathbf{x}) \| \bar{p}(\mathbf{x})) \\
&= \mathbb{E}_{p^\star(\mathbf{x})} \log \frac{2p^\star(\mathbf{x})}{p^\star(\mathbf{x}) + p_\theta(\mathbf{x})} + \mathbb{E}_{p_\theta(\mathbf{x})} \log \frac{2p_\theta(\mathbf{x})}{p^\star(\mathbf{x}) + p_\theta(\mathbf{x})} \\
&= \underbrace{\mathbb{E}_{p^\star(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x})} \log(1 - D(\mathbf{x}))}_{\text{Discriminator の目的関数そのもの}} + \log 4
\end{aligned}
$$

- この JS ダイバージェンスを最小化すれば、生成モデルを真の分布にマッチさせることができる

$$
\min_\theta \max_D \ \mathbb{E}_{p^\star(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x})} \log(1 - D(\mathbf{x}))
$$

GAN: Discriminator と Generator を交互に更新 [Goodfellow+, '14]

- 生成モデルを $\mathbf{x} \sim p_\theta(\mathbf{x}) \Leftrightarrow \mathbf{x} = G_\theta(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})$ と定義
  （$p(\mathbf{z})$ は任意の分布）
- 識別モデルを $D_\phi(\mathbf{x}) \approx D(\mathbf{x})$ とおく
- GAN: さきほどの min-max 問題を、これらを交互に更新して解く

$$\min_\theta \max_\phi \ \mathbb{E}_{p^\star(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D_\phi(G_\theta(\mathbf{z})))$$

GAN: Tips

- Discriminator を $D_\phi(\mathbf{x}) = \mathrm{sigmoid}(NN_\phi(\mathbf{x}))$ のような式でモデル化する場合、その対数は

$$\log D_\phi(\mathbf{x}) = -\mathrm{softplus}(-NN_\phi(\mathbf{x}))$$
$$\log(1 - D_\phi(\mathbf{x})) = -\mathrm{softplus}(NN_\phi(\mathbf{x}))$$

  ただし $\mathrm{softplus}(x) = \log(1 + \exp(x))$

- Discriminator はチャンスレートよりよい識別をする。このとき $\log(1 - D_\phi(\mathbf{x}))$ より $\log D_\phi(\mathbf{x})$ の方が $\mathbf{x}$ に関する勾配が大きくて、学習が進みやすい。そこで実際には

$$\min_\theta \mathbb{E}_{p(\mathbf{z})} \log D_\phi(G_\theta(\mathbf{z}))$$
$$\text{s.t. } \phi = \arg\min_\phi \mathbb{E}_{p^\star(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x})} \log(1 - D_\phi(\mathbf{x}))$$

  という問題を代わりに解くようにすると学習しやすい
  （というかそうしないと学習が進まない）

ぼやけない画像生成: Deep Convolutional GAN [Radford+, '16]

- LSUN bedroom データセットに対し、Generator に Deconv Net、Discriminator に Conv Net を使って GAN を学習した例
- Pooling を使わない (strided conv)、BatchNormalization を使うなどの工夫

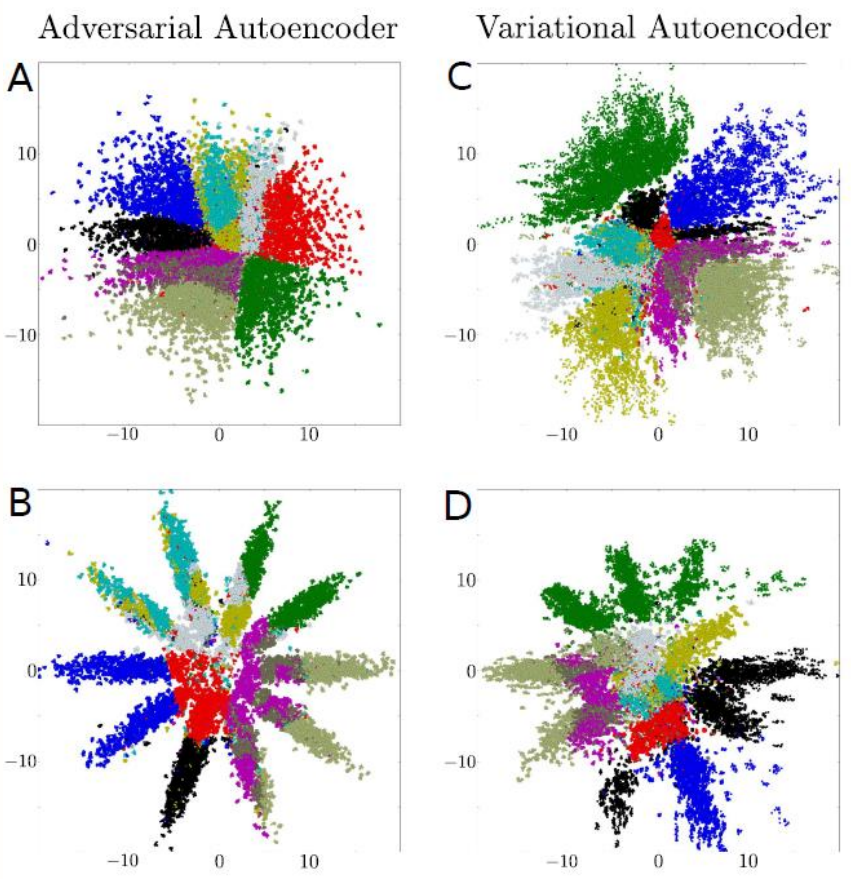
GAN の問題点：最適化が難しい、定量的な指標がない

- Discriminator の学習が足りないと、Generator が正しい方向に導かれない
- 一方、Discriminator の学習が進み過ぎると、$D_\phi(\mathbf{x})$ の $\mathbf{x}$ に関する勾配が小さくなってしまい、Generator の学習が進まなくなる

- 単一の目的関数を最適化する問題になっていないので、Discriminator と Generator の各目的関数の値をみても、学習がうまくいっているのかわからない（学習曲線が描けない）
  - Discriminator が圧勝するのは見てわかるので、その場合学習がうまくいかなさそうと推測できる
  - そうでない場合、うまく進んでいるのかどうかロス値だけではわからない
  - Parzen 窓による密度推定などで定量評価するのが一般的だが、この方法は GAN とあまり相性が良くない（VAE のようにガウス分布をつかったモデルの方が有利になってしまう）

Adversarial AutoEncoder [Makhzani+, ‘16 (arXiv)]

- GAN は $\mathbf{x}$ 以外の確率変数をマッチングするのにも使える
- Adversarial AutoEncoder
  - AutoEncoder の $\mathbf{z}$ を所望の分布 $p(\mathbf{z})$ にマッチさせるのに使う
  - $p(\mathbf{z})$ が単純な標準ガウス分布であっても、KL ダイバージェンスによるマッチングに比べて分布がきれいに合う



Generative Adversarial Nets まとめ

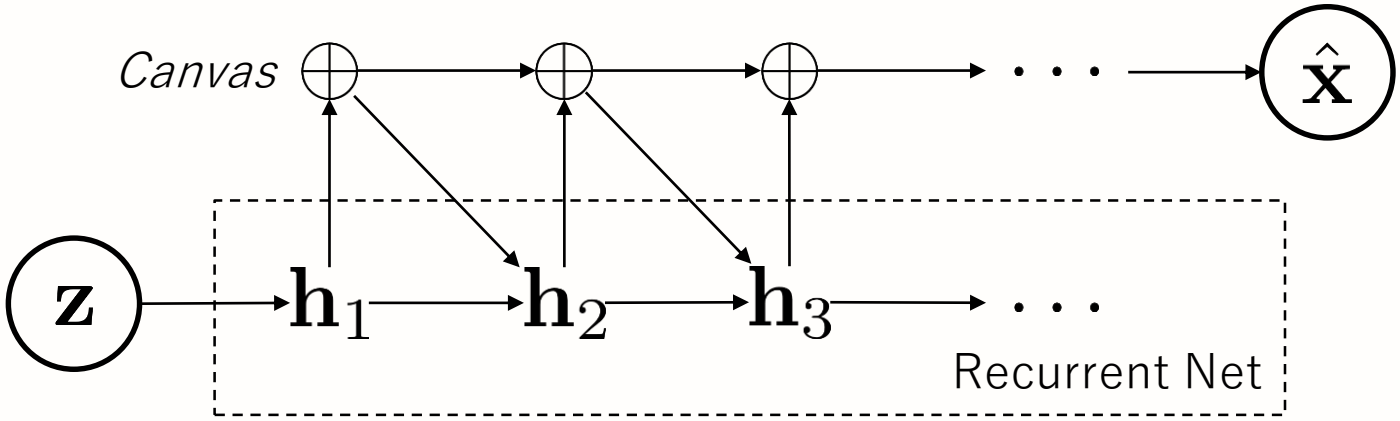
- GAN は条件付き尤度 $p_\theta(\mathbf{x}|\mathbf{z})$ やその勾配が計算できなくても学習できる生成モデル
- 識別モデルによる JS ダイバージェンス推定にもとづいて、Generator と Discriminator の間の min-max 最適化を行う
- Generator と Discriminator を交互に更新して学習する
- Deconv Net / Conv Net と組み合わせることで、ぼけないキレイな画像が生成できる
  - まだうまく生成できないデータセットはたくさんある（特にバリエーションが多く解像度の高い画像）
- 一方、GAN の最適化は難しい

生成モデルと意思決定

複雑なデータを少しずつ生成する

- 複雑なデータ分布を学習したい
  - たくさんの層が必要
  - でもパラメータ数は抑えたい
  - つまり、同じパラメータを複数回使いたい → ConvNet, **Recurrent Net**
- 層を増やしたときに、途中状態に何かしらの構造を入れないと学習が難しい
  - 一番単純で意味がありそうなのは、データと同じ構造を持った変数を入れること
  - たとえば、各ステップでの出力の総和が結果になるような Recurrent Net
  - → データを少しずつ生成する

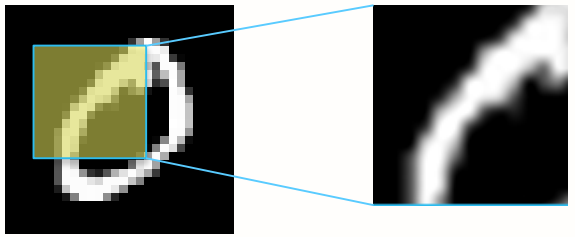
複雑なデータを少しずつ生成する

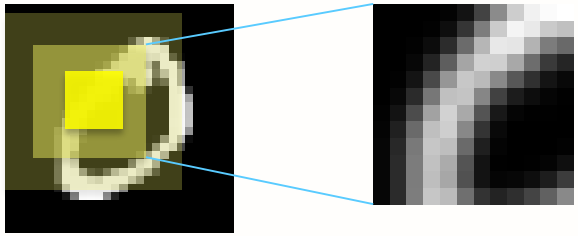


- 「キャンバス」に出力を足しこんでいく
- 現在のキャンバスの状態を Recurrent Net に再入力する
- 「キャンバス」が大きい場合、どちらも **Attention** を入れることでネットワークの複雑度を下げられる
  - 一度にキャンバスの一部にだけ書き込む
  - 一度にキャンバスの一部だけを読み込む

構造を持ったデータに対する Attention

- 長さや縦横などの構造を持ったデータについて、一部だけを読みだす・書き込む
- **Hard Attention**
  - 中心座標とサイズを指定して、そこから正確に矩形を切り抜く／その矩形に足し込む
  - Attention のサイズだけに依存した計算量だが、微分できない
- **Soft Attention**
  - 矩形に加えて「ぼかし幅」を指定して、周辺の要素との重み付き和を読みだす／重み付きで足し込む
  - Hard Attention を 1-hot な action とみなして、softmax を使うことが多い
  - 微分できるが、周辺要素も計算に絡む（softmaxの場合、全体を見る必要）





Soft attention: DRAW [Gregor+, ‘15]

- Soft Attention による reader / writer を変分 AutoEncoder と組み合わせる

（追試例）

Hard attention: Recurrent Attention Model [Mnih+, '14]

- 最近の attention + Recurrent Net のはしり
- データ生成ではなく、分類問題に使っていた

- Hard attention は微分できないので、単純な誤差逆伝播では attention の当て方を学習できない
- Attention の当て方を $\mathbf{a}$, 目的関数を R と書く
- Attention を確率的に決める： $p_\theta(\mathbf{a}|\mathbf{x})$ （θ は attention を出力するのに使うパラメータ)
- すると、目的関数の θ に関する勾配は

$$\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{a})} R = \int R \nabla_\theta p_\theta(\mathbf{a}) d\mathbf{a} = \int R\, p_\theta(\mathbf{a}) \frac{\nabla_\theta p_\theta(\mathbf{a})}{p_\theta(\mathbf{a})} d\mathbf{a}$$
$$= \mathbb{E}_{p_\theta(\mathbf{a})}[R \nabla_\theta \log p_\theta(\mathbf{a})]$$

微分できない確率変数の入ったモデルの最適化：尤度比法

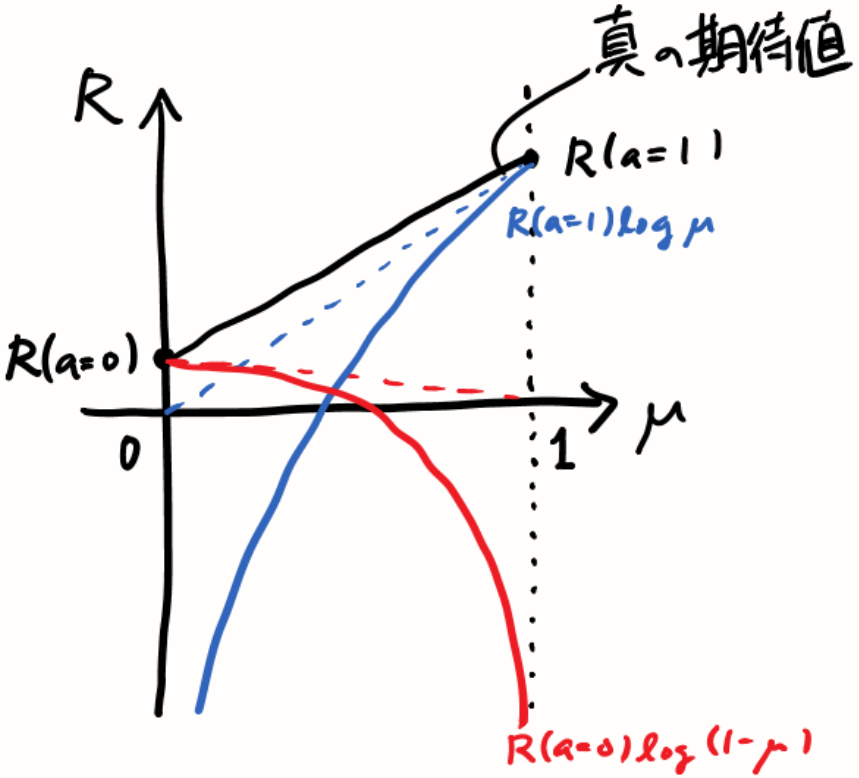
$$\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{a})} R = \int R \nabla_\theta p_\theta(\mathbf{a}) d\mathbf{a} = \int R\, p_\theta(\mathbf{a}) \frac{\nabla_\theta p_\theta(\mathbf{a})}{p_\theta(\mathbf{a})} d\mathbf{a}$$
$$= \mathbb{E}_{p_\theta(\mathbf{a})}[R \nabla_\theta \log p_\theta(\mathbf{a})]$$

- このように、期待値を取る分布に関する勾配を対数勾配の期待値で近似する方法を**尤度比法 (Likelihood Ratio; LR)** という
- 強化学習においては **REINFORCE** と呼ばれる

- REINFORCE
  - 行動にたいする環境の状態変化の微分がわからないときに、報酬の方策にたいする勾配を尤度比法とおなじように求める手法 (policy gradient)
  - Hard attention が行動に、その結果得られる入力が状態に、最終的な生成データから得られる目的関数が報酬（コスト）に対応する

尤度比法は勾配推定の分散が大きい

- 例：単純な 0/1 値変数（ベルヌーイ分布の平均 μ で指定する）



- 確率 μ で青い線の勾配が、確率 $1-\mu$ で赤い線の勾配が選ばれる
- （青赤の線の絶対的な値には意味がなく、勾配だけを見る）
- 選ばれる線によって、勾配の符号が逆であることに注目
- 特に μ が小さいとき、ほぼ確実に正しくない符号の勾配を推定してしまう
- 同様の事象は、reparameterization trick が可能なガウス分布でも観察できる

勾配推定の分散を下げる：Control Variate（ベースライン）

- 尤度比法において勾配推定を安定させる手法
- 期待値が解析的に求まる変数 B（ベースライン）をつかって

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{a})} R &= \nabla_\theta \mathbb{E}_{p_\theta(\mathbf{a})}[R - B] + \nabla_\theta \bar{B} \\ &= \mathbb{E}_{p_\theta(\mathbf{a})}[(R - B)\nabla_\theta \log p_\theta(\mathbf{a})] + \nabla_\theta \bar{B}\end{aligned}$$

  のように変形（$\bar{B} = \mathbb{E}_{p_\theta(\mathbf{a})} B$）して、サンプリングされる勾配の分散を下げる

  - たとえば、報酬の期待値を推定してそれをつかう、というのがよくやられる
  - ほかに期待値を action として使える環境で線形近似でベースラインを求める手法も出てきている (MuProp)
- Control Variate を使っても、分散はある程度のこる
- サンプリングに使う分布が単純な場合（基本的にそう）、Reparameterization Trick より確実に分散が大きい
  - ただ、Reparameterization Trick は微分可能な変数変換がないと使えないので、hard attention にはつかえない

データ生成は意思決定のくりかえし [Bachman+, '15]

- "Data Generation as Sequential Decision Making"
- モデル中の中間変数を
  その分布パラメータを出力する action
  分布パラメータ (action) をもとにサンプリングを行う環境
  とみなすと、マルコフ決定過程における報酬最大化問題になる
- この場合、学習したいのは action を予測する方策 (policy)
- Helmholtz Machine のように変分ベイズにもとづく手法は、policy に対して「データを知っている」推論モデルの予測に policy を近づける
  - これは、強化学習における Guided Policy Search と等価

生成モデルと意思決定のまとめ

- 複雑なデータ分布を生成するには、少しずつデータを生成するのが有効と考えられている
- その際、attention など、何かしらの action によってデータを読み書きする
- Soft attention は微分でき、全体を誤差逆伝播法で学習できるが、計算コストが高い
- Hard attention は微分できないので、学習に尤度比法などを用いるが、勾配推定の分散が大きく学習が遅い
- 尤度比法は REINFORCE とも呼ばれ、Policy Gradient 法と関連が深い
- さらにデータ生成の確率モデルはマルコフ決定過程としても定式化でき、変分ベイズ法は Guided Policy Search と対応づけられる

まとめ

- 生成モデルの Deep Learning について、とくに画像生成の文脈での研究を紹介
- Helmholtz Machine は問題がキレイだが、尤度をモデル化する必要があって画像など低次元多様体を意識するべきデータとは相性が悪い
- Generative Adversarial Nets はそれを解決するが、学習・評価が難しい
- 一方で、微分できないモデルの研究も進んでいて、とくに強化学習との融合が進んでいる

- Recurrent Net を組み込んだモデルは盛んに研究されていますが、今回は省略
- 言語モデル系の話も省略しました

Reference

P. Bachman, D. Precup. Data Generation as Sequential Decision Making. NIPS, 2015.

J. Bornschein, Y. Bengio. Reweighted Wake-Sleep. ICLR, 2015.

P. Dayan, G. E. Hinton, R. M. Neal, R. S. Zemel. The Helmholtz Machine. Neural Computation 7, 889-904, 1995.

I. J. Goodfellow, J. P.-Abadie, M. Mirza, B. Xu, D. W.-Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Nets. NIPS, 2014.

K. Gregor, I. Danihelka, A. Graves, D. Wierstra. DRAW: A Recurrent Neural Networks For Image Generation. ICML, 2015.

G. E. Hinton, P. Dayan, B. J. Frey, R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. Science, vol. 268, pp. 1158-1161, 1995.

D. P. Kingma, M. Welling. Auto-Encoding Variational Bayes. ICLR, 2014.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow. Adversarial Autoencoders. ArXiv:1511.05644, 2015.

V. Mnih, N. Hees, A. Graves, K. Kavukcuoglu. Recurrent Models of Visual Attention. NIPS, 2014.

A. Radford, L. Metz, S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR, 2016.

D. J. Rezende, S. Mohamed, D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. ICML, 2014.