

Generative Adversarial Network (と強化学習との関係を少し)

鈴木 雅大

2017/12/12

自己紹介

鈴木雅大（博士課程3年）

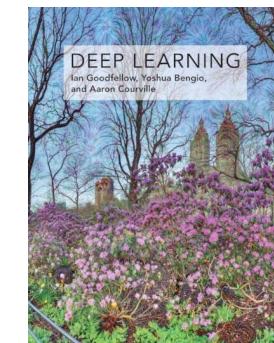
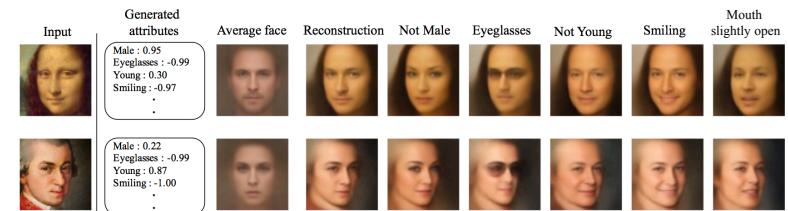
□ 経歴

- 学部～修士：北海道大学情報科学研究科
- 博士～：東京大学工学系研究科 松尾研究室

□ 専門分野：

- 機械学習， 転移学習
- 深層生成モデル (VAE)

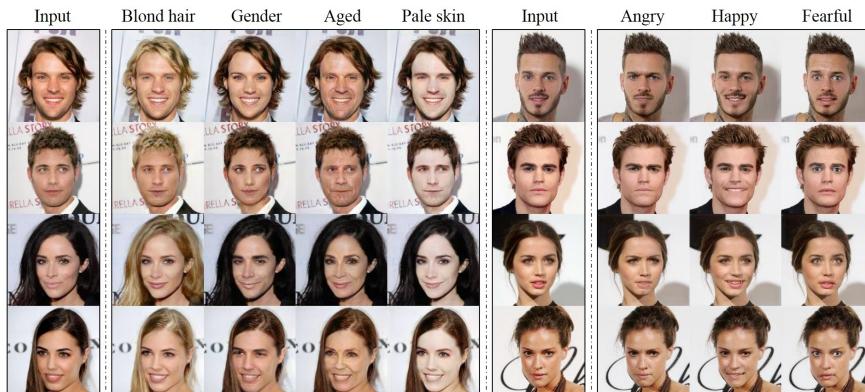
- Deep Learning基礎講座，先端人工知能論などの講義・演習担当
- Goodfellow著「Deep Learning」の監修・翻訳



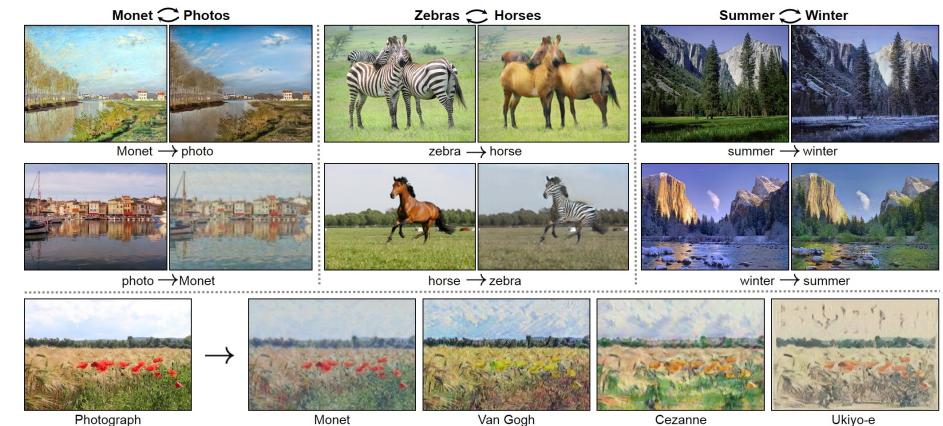
今日の内容

□ GANの説明と強化学習との関係について

- 前半：GANの説明（※先端人工知能論 II／Deep Learning 応用講座で話した内容とほぼ同じです）
- 後半：GANと強化学習の関係



[Choi+ 2017]



[Zhu+ 2017]

目次

□ 前半 :

- 生成モデルとニューラルネットワーク
- Generative Adversarial Network
- GANの学習の難しさ
- GANの種類
- GANの評価
- GANの応用

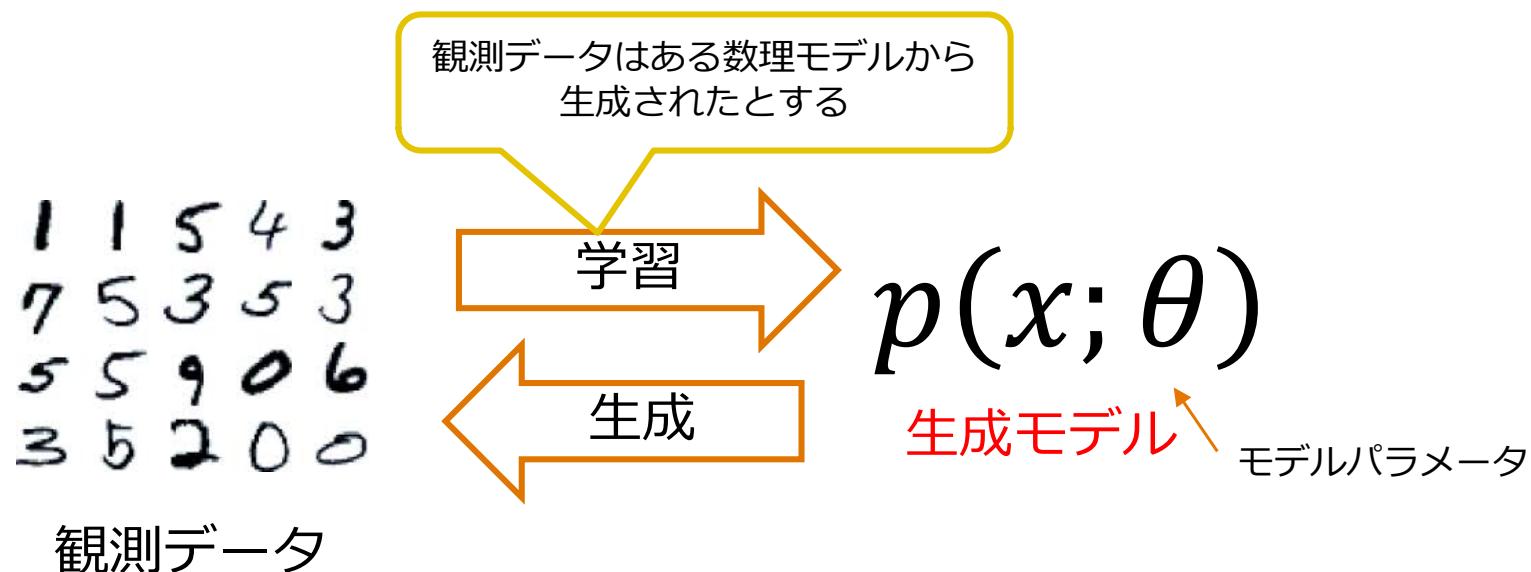
□ 後半 :

- Actor-criticとGAN
- 逆強化学習とGAN
- 深層生成モデルと強化学習

生成モデルとニューラルネットワーク

生成モデル

- データの生成過程を数理的にモデル化したもの.
- 数理モデルは確率分布によって表される



- x_i が確率 $p(x; \theta)$ から生成されるときは $x_i \sim p(x; \theta)$ と表記する.
- $p(x; \theta)$ を $p_\theta(x)$ とも書く

生成モデルだとできること

□ サンプリング :

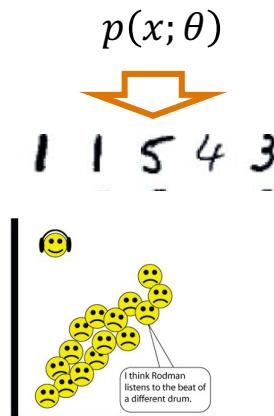
- 確率モデルがあるので、未知のデータを生成できる
- 「生成」モデルと呼ばれるのはここから

□ 密度推定 :

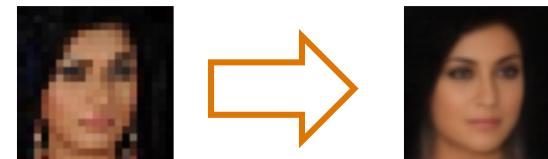
- データ x を入力すると、密度 $p(x)$ が得られる。
- 外れ値検出や異常検知に用いられる。

□ 欠損値補完、ノイズ除去 :

- 欠損やノイズのある \tilde{x} を入力すると、真の x の推定値が得られる。

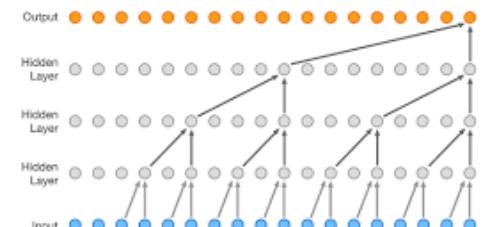
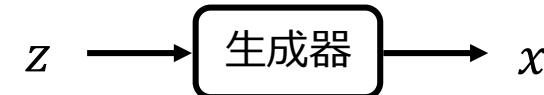


<http://jblomo.github.io/datamining290/slides/2013-04-26-Outliers.html>



ニューラルネットワークによる生成モデルの表現

- 生成モデル $p(x; \theta)$ を **深層ニューラルネットワーク** によって表現する (**深層生成モデル**) .
 - 従来の確率分布だと、複雑な入力を直接扱えない.
 - ニューラルネットワークで表現することで、より複雑な入力 x を扱えるようになる.
 - 一般物体画像のような複雑な画像も生成できる可能性がある !
- 大きく分けて2つのアプローチがある (有向モデルの場合) .
 - $p(x|z)$ (生成器, generator) をモデル化する.
 - z から x へのニューラルネットワーク $x = f(z)$ によって表現.
 - VAE, GANなど.
 - $p(x)$ を直接モデル化する (自己回帰モデル) .
 - $p(x) = \prod_i p(x_i|x_1, \dots, x_{i-1})$ として、各条件付き分布をモデル化.
 - NADE[Larochelle+ 11], PixelRNN (CNN) [Oord+ 16], WaveNet[Oord+ 16]など.



<https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

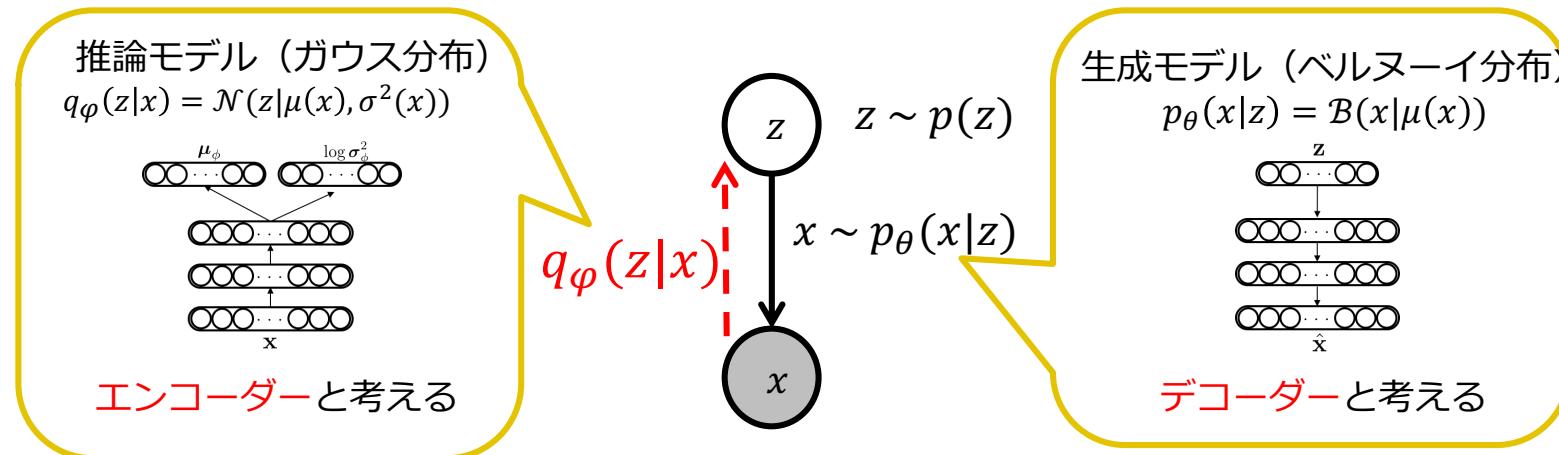
深層生成モデルの分類

	学習方法	尤度の計算	サンプリング	潜在変数への推論	学習するモデル (NNでモデル化)
VAE	変分下界の 最大化	厳密にはできない (変分下界を計算)	低成本	近似分布によって可能	生成モデル $p(x z)$ 推論モデル $q(z x)$
GAN	敵対的学習	できない (尤度比のみ)	低成本	推論はモデル化されてい ない (モデルによる)	生成器 $G(z)$ 識別器 $D(x)$
自己回帰 モデル	対数尤度の 最大化	できる	高コスト	潜在変数自体がない	複数の条件付き分布 $\prod_i p(x_i x_1, \dots, x_{i-1})$

- それぞれ利点と欠点がある
 - VAEとGANはサンプリングが早いが、尤度を直接求められない。
 - 自己回帰モデルは尤度を正確に求められるが、モデルが多くサンプリングや学習の速度が遅い。

Variational Autoencoder

- Variational autoencoder (VAE) [Kingma+ 13; ICLR 2014] [Rezende+ 14; ICML 2014]



VAEからの画像生成

[Kingma+ 13]より

- $p(x|z)$ だけでなく $p(z)$ ($= \int q(z|x)p(x)dx$) も学習している
 - データの多様体が z で獲得されている。

生成画像

- ランダムな z から画像をサンプリング
- 輪郭等がぼやける傾向がある。

2 8 3 8 3 8 5 7 3 8
8 3 8 2 7 9 3 5 3 8
8 5 5 9 4 3 9 5 1 6
1 9 1 8 3 8 3 4 9 7
2 7 3 6 4 3 0 2 0 3
5 9 7 0 5 9 3 8 4 5
6 9 4 3 6 2 8 5 5 2
8 4 9 0 8 0 7 9 6 6
7 4 3 6 3 0 3 6 0 1
2 1 8 0 9 7 1 8 0 0

[Kingma+ 13]より

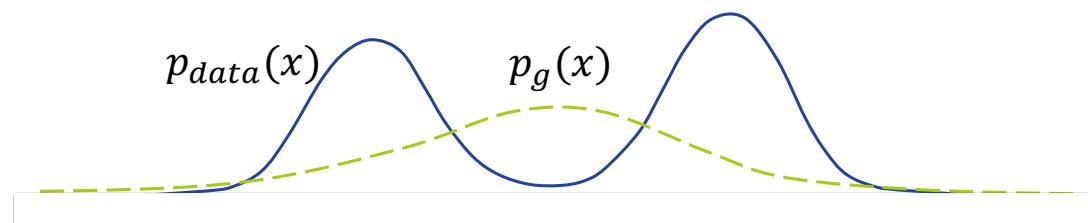


@AlecRad

Generative Adversarial Network

なぜ生成画像がぼやけてしまうのか？

- 理由の一つは、**確率分布を明示的にモデル化して最尤学習を行っているため**。
 - VAEでは $p(x|z)$ をガウス分布としてモデル化しているため、再構成誤差が（分散を考慮しなければ）二乗誤差になる。
 - 画素をはっきりさせるより曖昧にした方が、誤差は小さくなる。
 - ガウス分布の分散を小さくすることははっきり再構成できるようになるが、逆に未知のデータをうまく生成できなくなる。
- また最尤学習では、訓練データにない部分についても高い確率を置くように訓練しがち。
 - 最尤学習（尤度最大化） = KLダイバージェンスの最小化
 - 訓練データにない部分がぼやけた画像になる。



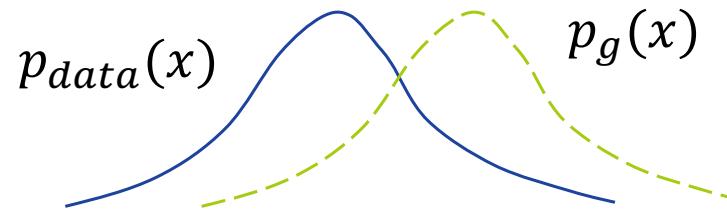
$$\begin{aligned}KL[p_{data}||p_g] &= \int p_{data} \log \frac{p_{data}}{p_g} dx \\&= E_{p_{data}}[\log p_{data}] - E_{p_{data}}[\log p_g]\end{aligned}$$

対数尤度

-> 確率分布を**暗黙的に**モデル化し、**別の学習基準**を用いる必要がある！

暗黙的な生成モデルの学習

- 確率分布 $p_g(x)$ を定義しないで、暗黙的な生成モデルとして考える。
 - 尤度を測ることができない！
- 目標：真の分布 $p_{data}(x)$ と近いモデル分布 $p_g(x)$ を求める.



- 直接尤度を評価できないので、モデル分布と真の分布の密度比を考える[Uehara+ 17, Mohamed+ 16].

$$r(x) = \frac{p_{data}(x)}{p_g(x)}$$

密度比の解釈

- データ集合 $\mathcal{X} = \{x_1, \dots, x_N\}$ のうち、半分のデータがモデル分布から生成され、もう半分が真の分布から生成されるとする。
 - 真の分布からの方を $y = 1$ 、モデル分布からの方を $y = 0$ とラベル付けする。
 - するとデータ集合は次のようになる。

$$\{(x_1, 1), \dots, \left(x_{\frac{N}{2}}, 1\right), \left(x_{\frac{N}{2}+1}, 0\right), \dots, (x_N, 0)\}$$

- このことから、モデル分布と真の分布は、ラベルが与えられた下での条件付き分布となる [Sugiyama+ 2012].

$$p_{data}(x) = p(x|y=1)$$

$$p_g(x) = p(x|y=0)$$

密度比の解釈

- したがって、密度比は次のようになる。

$$\begin{aligned}\frac{p_{data}(x)}{p_g(x)} &= \frac{p(x|y=1)}{p(x|y=0)} \\ &= \frac{\frac{p(y=1|x)p(x)}{p(y=1)}}{\frac{p(y=0|x)p(x)}{p(y=0)}} = \frac{p(y=1|x)}{p(y=0|x)} \cdot \frac{1-\pi}{\pi} \quad (\text{ただし, } \pi = p(y=1))\end{aligned}$$

->すなわち、 $p(y=1|x)$ を推定できればよい。

- $p(y=1|x)$ を推定する分布を $q_\varphi(y=1|x)$ とする。

- この分布をニューラルネットワークでパラメータ化する。

$$q_\varphi(y=1|x) = D(x; \varphi) \quad x \rightarrow D(x; \varphi) \rightarrow y$$

- D を識別器(discriminator)と呼ぶ。
- 識別器によって、密度推定をNNが得意な分類問題に置き換えることができる。

識別器の目的関数

- データ集合から識別器を学習するため、次の負の交差エントロピー損失を最大化する。

$$E_{p(x,y)}[y \log D(x; \varphi) + (1 - y) \log(1 - D(x; \varphi))]$$

- 式変形して、

$$\begin{aligned} & E_{p(x|y)p(y)}[y \log D(x; \varphi) + (1 - y) \log(1 - D(x; \varphi))] \\ &= E_{p(x|y=1)p(y=1)}[\log D(x; \varphi)] + E_{p(x|y=0)p(y=0)}[\log(1 - D(x; \varphi))] \\ &= \pi E_{p_{data}(x)}[\log D(x; \varphi)] + (1 - \pi) E_{p_g(x)}[\log(1 - D(x; \varphi))] \end{aligned}$$

- $\pi = \frac{1}{2}$ とすると、目的関数 $V(D)$ は、

$$V(D) = E_{p_{data}(x)}[\log D(x; \varphi)] + E_{p_g(x)}[\log(1 - D(x; \varphi))]$$

目的関数の意味

- ▣ 本来の目的は、 $p_g(x)$ を $p_{data}(x)$ に近づけること！
 - ▣ 学習した D を使って $p_g(x)$ と $p_{data}(x)$ の距離を求めたい。
 - ▣ 目的関数 $V(D)$ にはどのような意味がある？
- ▣ もし識別関数が適切に推定できたら（すなわち $D^*(x) = p(y=1|x)$ ）,

$$D^*(x) = \frac{r}{r+1} = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)} \text{ に収束する。}$$

- ▣ このとき、 $V(D^*) = 2 \cdot JSD(p_{data}||p_g) - 2 \log 2$ となる。
 - ▣ JSD はJensen-Shannonダイバージェンス。
 - ▣ $JSD(p_{data}||p_g) = \frac{1}{2}KL(p_{data}||\frac{p_{data}+p_g}{2}) + \frac{1}{2}KL(p_g||\frac{p_{data}+p_g}{2})$ (KLと違って対称的)

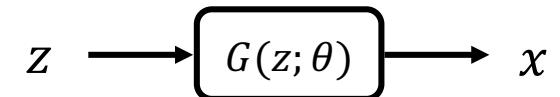
->つまり、 $V(D^*)$ は p_{data} と p_g のJensen-Shannonダイバージェンスに対応している！

生成モデルの学習

- $p_g(x) = \int p(x|z)p(z)dz$ と考えて, $p(x|z)$ を推定する分布を $q_\theta(x|z)$ とする.

- この分布をニューラルネットワークでパラメータ化する.

$$q_\theta(x|z) = G(z; \theta)$$



- G を**生成器(generator)**と呼ぶ.

- すると, G を学習するための目的関数は,

$$V(D^*, G) = E_{p_{data}(x)}[\log D^*(x)] + E_{p(z)}[\log(1 - D^*(G(z; \theta)))]$$

- G は, この目的関数を**最小化**するように学習する.

- p_{data} と p_g のJS距離を近づけるため.

生成器と識別器の学習

- 実際には、適切な G が得られないと、最適な D は学習できない。
 - 逆も同じ。
- したがって、目的関数を交互に最適化することを考える。

- ・識別器の学習 (G は固定)

$$\max_{\varphi} E_{p_{data}(x)}[\log D(x; \varphi)] + E_{p(z)}[\log(1 - D(G(z; \theta); \varphi))]$$

- ・生成器の学習 (D は固定)

$$\min_{\theta} E_{p(z)}[\log(1 - D(G(z; \theta); \varphi))]$$

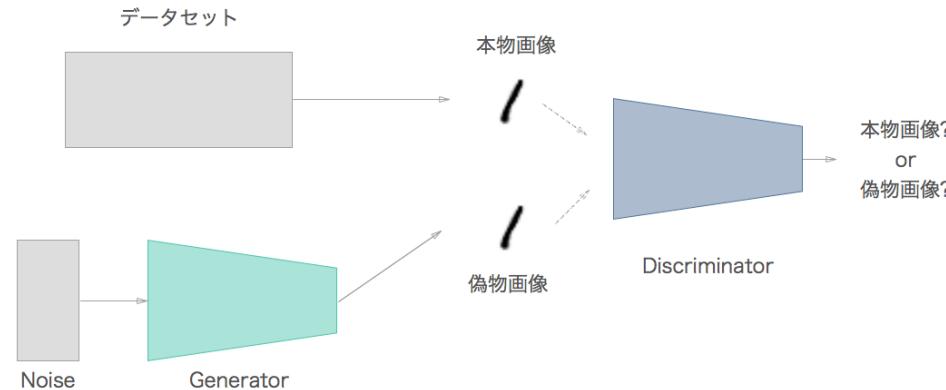
- このような枠組みで学習する生成モデルを、

generative adversarial networks (GANs) [Goodfellow+ 14]

と呼ぶ。

Generative adversarial nets

□ 全体の構造



□ 直感的には、 G と D で次のゲーム（ミニマックスゲーム）をする（敵対的学習）。

- G はなるべく D を騙すように x を生成する。
- D はなるべく G に騙されないように識別する。

$$\min_G \max_D V(D, G)$$

->最終的には、 D が本物と区別できないような x が G から生成される。

アルゴリズム

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

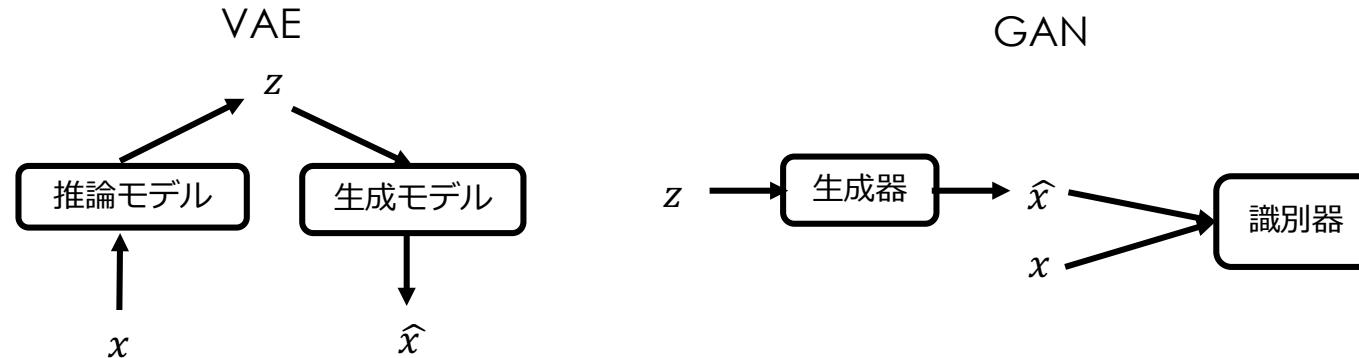
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

end for

□ D を k ステップ更新してから、 G を更新する。

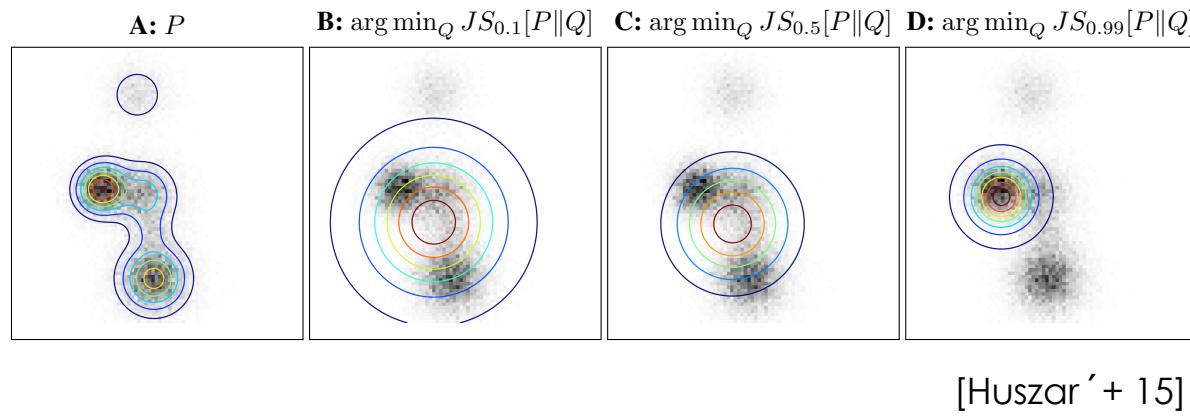
□ ただし、 $k = 1$ とする場合が多い。

VAEとGANの違い



- VAEは生成モデルの形を仮定しているが、GANは分布を仮定しない
 - GANは尤度ではなく、識別器によって、生成器のよさを評価する。
- 測る距離が異なる。
 - VAEは対数尤度 (=KLダイバージェンス) , GANはJSダイバージェンス。
- VAEは推論分布（エンコーダー）を考えるが、GANは考えない。
 - ただし、モデルによる。
- どちらもDNNの得意な識別問題に持つていているところがポイント。

KLダイバージェンスとJSダイバージェンス



- 尤度最大化 (KLダイバージェンス最小化) はデータのないところも覆ってしまう (B) .
- KLを逆にすると、一つの峰だけにfitするようになる (C) .
- JSダイバージェンスはちょうど中間あたりで学習 (D) .

GANの生成画像

- ランダムな z から画像をサンプリング[Goodfellow+ 14]

7	3	9	3	9	9
1	1	0	6	0	0
0	1	9	1	2	2
6	3	2	0	8	8

a)



b)



c)



d)

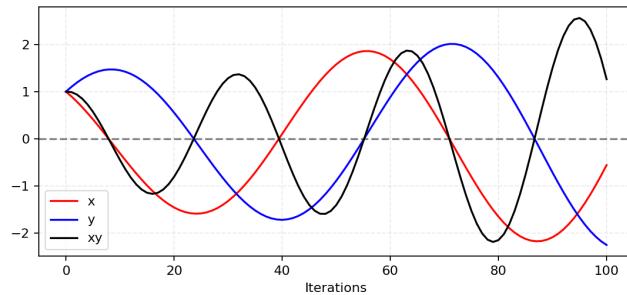
GANの学習の難しさ

GANの問題点

- GANにはいくつかの困難な点がある。
 - 収束性
 - Mode collapse問題
 - 勾配消失
- これらの困難は互いに関係している。

GANの収束性について

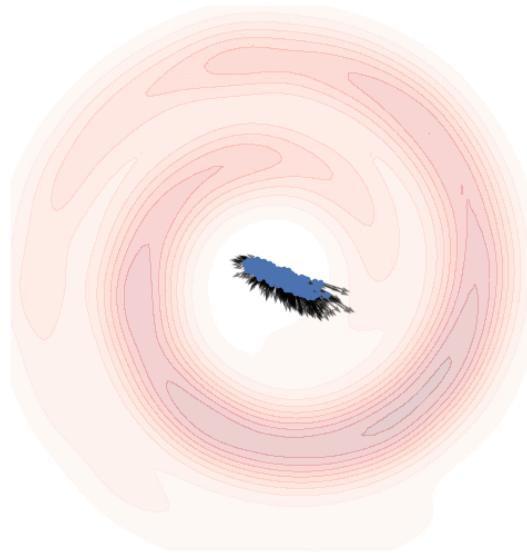
- $V(D^*, G)$ が θ について凸関数ならば, G は収束することが保証されている.
 - しかし, 非凸の場合は保証されない (NNは当然非凸) .
- 実際の学習では, SGDで交互に最適化している.
 - ミニマックスゲームの平衡点は, 両方のプレイヤーが同時に最小になる点 (ナッシュ均衡) .
 - ゼロサムゲームなので, お互いが V を最適化しても, 平衡点に行かずに**振動する可能性がある**.
- 例 : 目的関数を $v(a, b) = ab$ とし, G と D で交互に最小化と最大化を繰り返すと, 次のようになる.



<https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html>

※最近では, 最適化によって (局所) ナッシュ均衡に到達するよう保証する研究が行われている ([Nagarajan+ 17] [Heusel+ 17] [Kodali+ 17]など) .

GANにおけるGとDの学習の様子

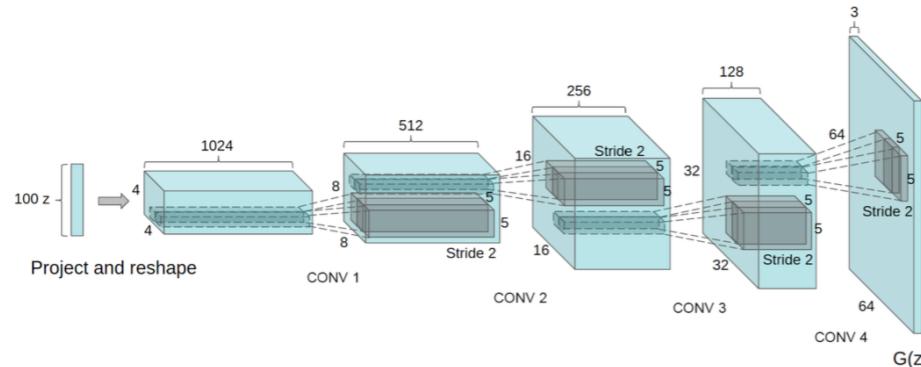


<http://www.inference.vc/an-alternative-update-rule-for-generative-adversarial-networks/>

- D は、 G が最適化する方向を示している。
- 敵対的というよりは、協力して学習しているイメージ。

DCGAN

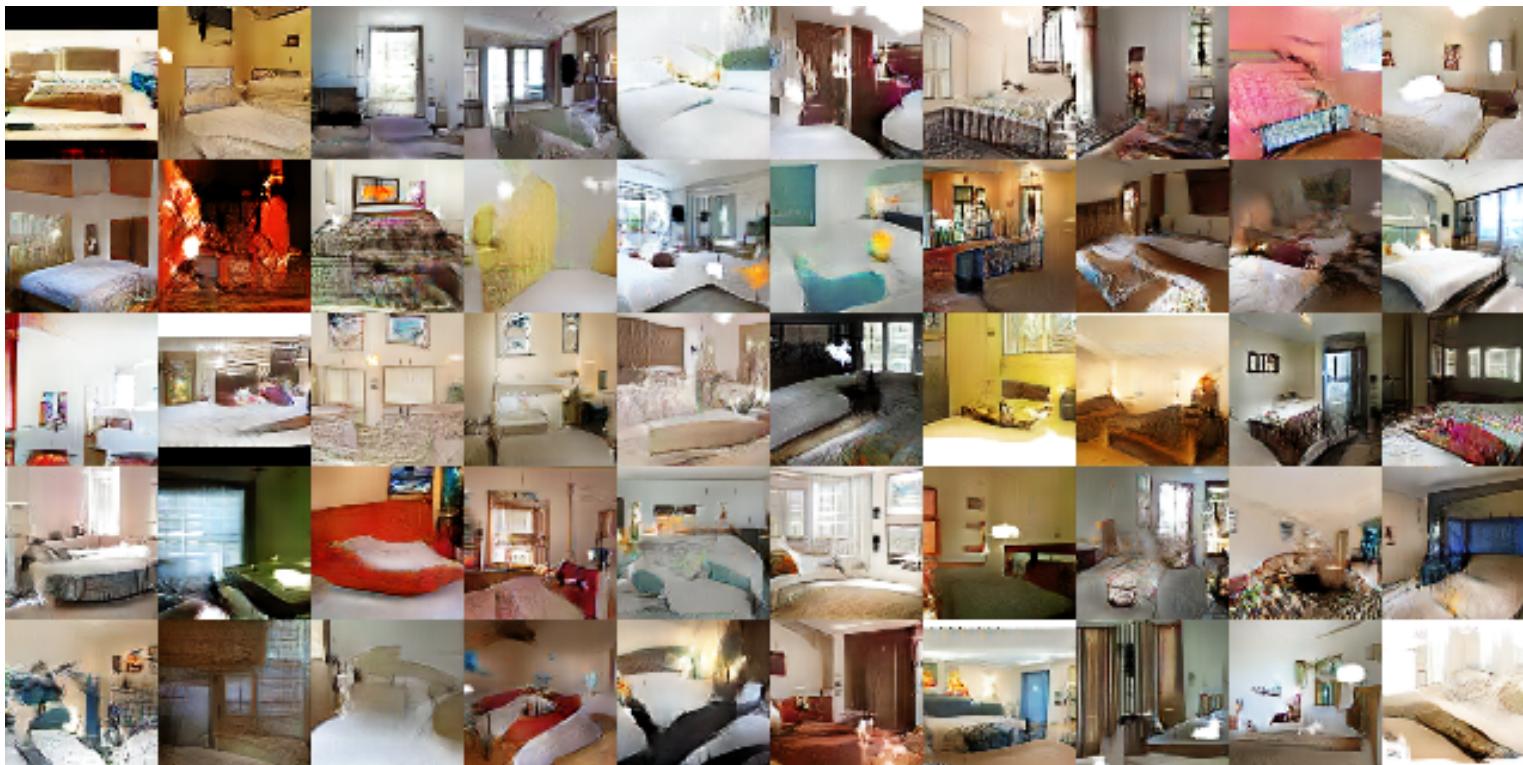
- Deep Convolutional Generative Adversarial Network[Alec+ 15]
 - 置込みNNでモデルを設計



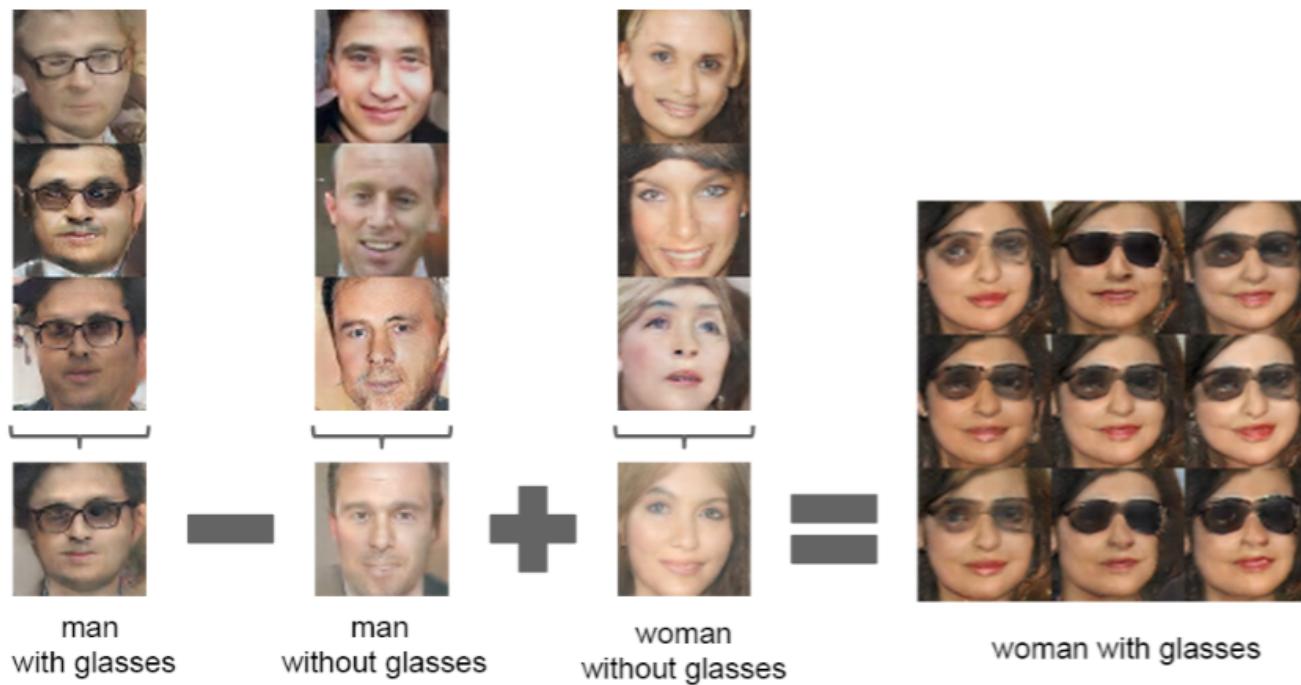
- 学習の安定性のために、いくつかのトリックを利用。
 - Batch Normalizationを使う（学習の安定のために重要）。
 - G にはReLUを使い、出力はtanh（必然的にデータは事前に標準化しておく）。
 - D にはleaky ReLUを使う。

DCGANの生成画像

- 従来と比較して、はるかに綺麗な画像が生成できるようになった.
- DCGANはGANのブレイクスルー的研究



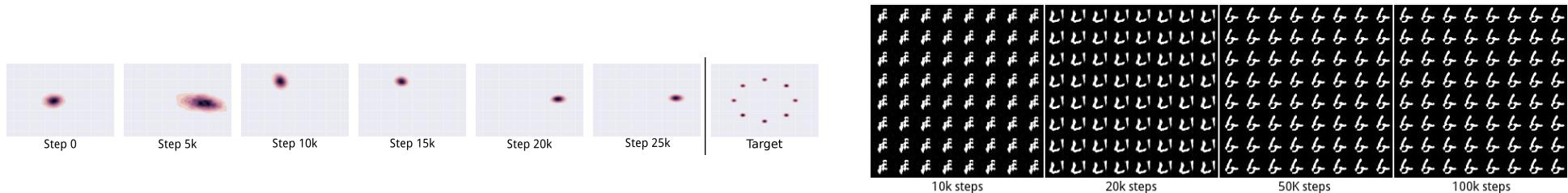
潜在空間でのベクトル演算



- ただし、GANにしかできない訳ではないことに注意。
- VAEでも可能（ただしGANの方が綺麗な画像が生成できる）。

Mode collapse問題

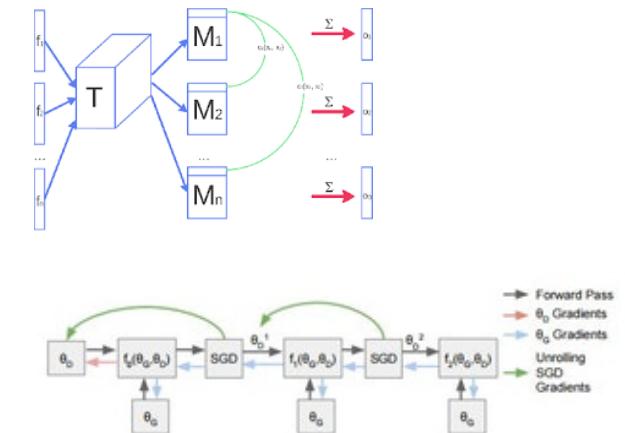
- 元々の定式化は、 G を固定して D を最適化するというものだった。
 - G は最適化した D を用いて学習する(minmax).
- (学習が不十分な) D を固定して、 G を最適化した場合はどうなるか？
-> G の生成データが全て、ある峰 (peaks) に対応するように学習してしまう (mode collapse) .



[Metz+ 17]

解決方法

- Minibatch discrimination[Salimans+ 17]
 - Minibatch内の多様性（サンプル間のノルムの距離）を考慮した discriminatorを設計。多様性が大きくなる方向にDを導くようにする。
- Unrolled GANs[Metz+ 17]
 - D のパラメータをK回更新し（unrolling），その時点でのパラメータで G を更新。 D は1回更新したパラメータに戻す。
- AdaGAN[Tolstikhin+ 17]
 - GANにブースティングの手法を用いる。
- Wasserstein GANs[Arjovsky+ 17]
 - 後述。



GANの勾配消失問題

- では、 D を完全に最適化させればいい？

- $p_{data}(x)$ について1, $p_g(x)$ について0を出力する.

-> 勾配が消えてしまうため、 G の方向がわからなくなる！！

- GANのジレンマ：

- D が十分学習できないうちに G を最適化すると、 G が同じようなサンプルしか生成しなくなる（mode collapse）.
- ある G のときに D を完全に学習すると、勾配が消失してしまう（勾配消失問題）.

勾配消失を防ぐトリック

- 勾配が消失しないようにするため, G の学習を次のように変更する場合が多い.

$$\min_{\theta} E_{p(z)}[-\log(D(G(z; \theta); \varphi))]$$

- D がほぼ偽物 ($D = 0$) と判断したときでも, 勾配が消失することを防ぐ.

元の目的関数の勾配

$$\nabla_{\theta} \log(1 - D(G(z; \theta))) = -\frac{\nabla_{\theta} D(G(z; \theta))}{1 - D(G(z; \theta))}$$

変更した目的関数の勾配

$$-\nabla_{\theta} \log(D(G(z; \theta))) = -\frac{\nabla_{\theta} D(G(z; \theta))}{D(G(z; \theta))}$$

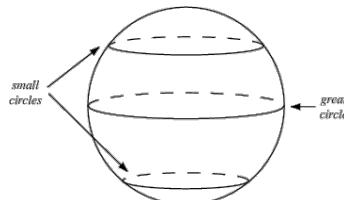
- ただし, もはやJSダイバージェンスの最小化ではない!

- 逆方向のKLと負のJSを最小化することに対応[Arjovsky+ 17]
- (ただし[Zhou+ 17]はこの証明は誤りと指摘している)

その他のトリック

- <https://github.com/soumith/ganhacks> に色々経験則が書いてある.

- Normalize the inputs
 - Use a spherical Z
 - Use Soft and Noisy Labels
- などなど.

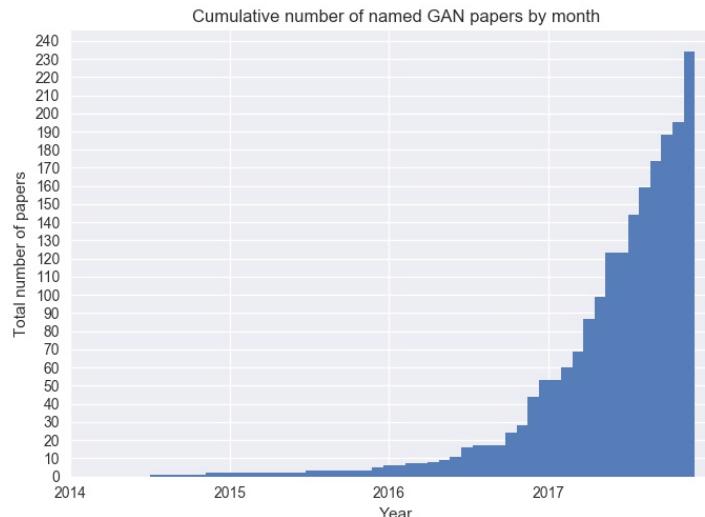


- 基本的には、実装例があればそれに従うこと.

GANの種類

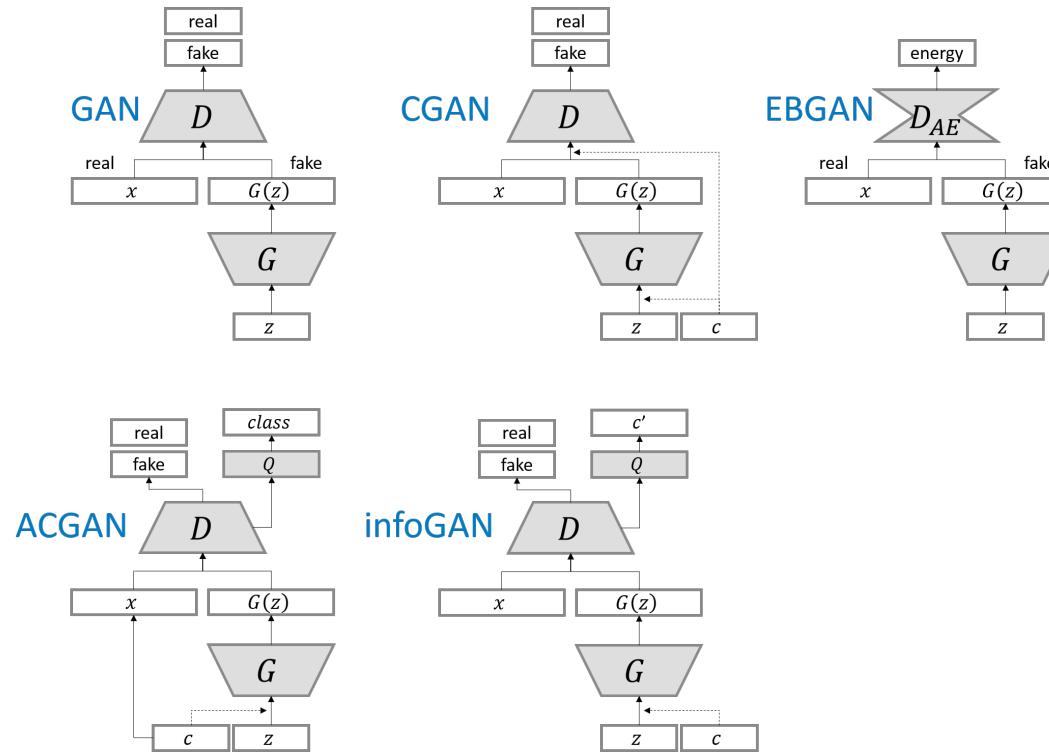
GAN Zoo

- ▣ GANの論文一覧
 - ▣ <https://github.com/hindupuravinash/the-gan-zoo>



- ▣ 指数関数的に増加.
 - ▣ 理論研究など、ここには入っていない研究も結構ある.
- ▣ 12/11現在234. 12月分はまだ.

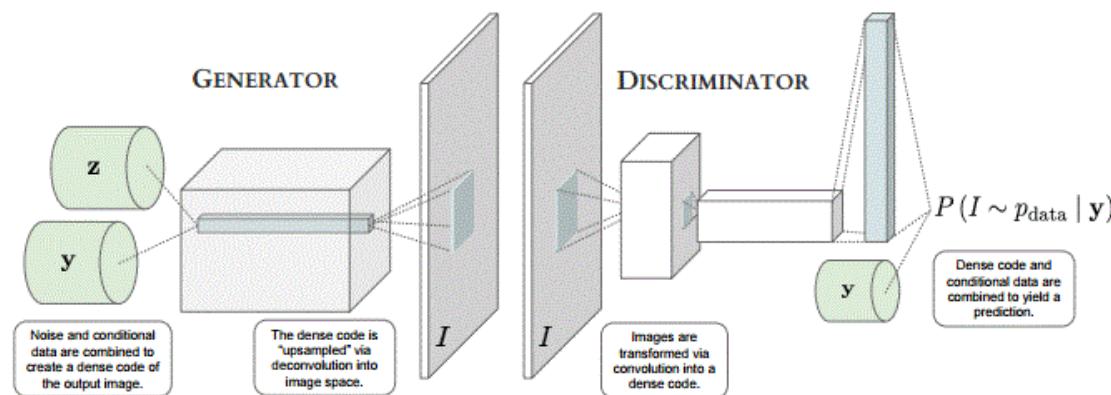
代表的なGANs



<https://github.com/hwalsuklee/tensorflow-generative-model-collections>

Conditional GAN

- Conditional Generative Adversarial Nets[Mirza+ 14]
 - c で条件づけたGAN.
 - GとDの両方に c を加える（下の図では y ）.



$$V_D(D, G) = E_{p_{\text{data}}(x)}[\log D(x, \textcolor{red}{c}; \varphi)] + E_{p(z)}[\log(1 - D(G(z; \theta), \textcolor{red}{c}; \varphi))]$$
$$V_G(D, G) = -E_{p(z)}[\log(D(G(z; \theta), \textcolor{red}{c}; \varphi))]$$

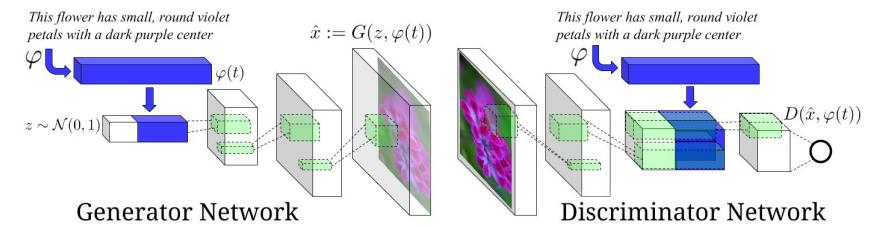
cGANで生成した画像

□ 文章情報で条件づけた例[Reed+ 16].

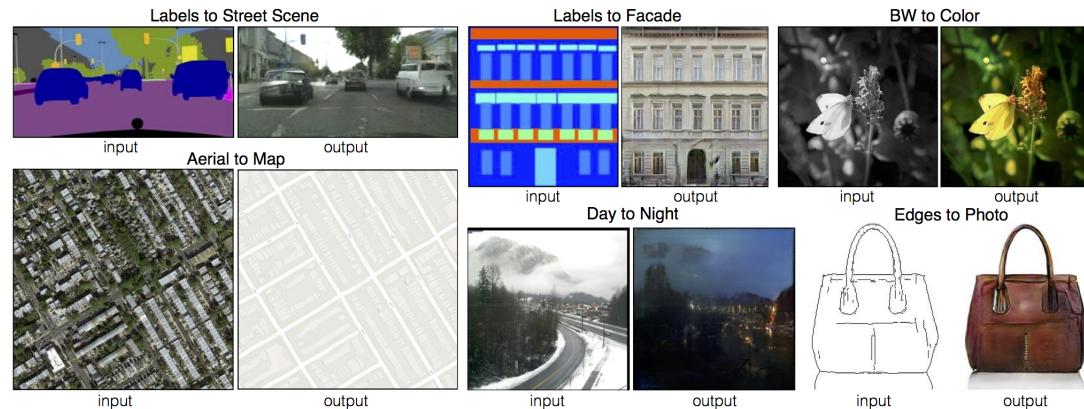
this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



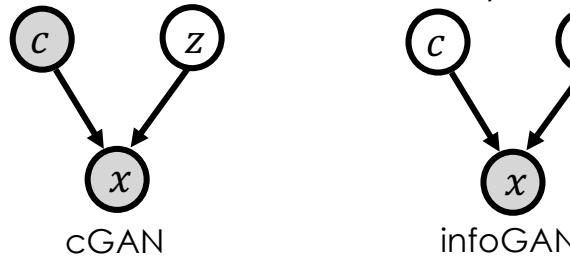
□ 画像で条件づけた例[Isola+ 16] (pix2pix, 後ほど改めて紹介) .



infoGAN

- Information Maximizing Generative Adversarial Networks[Chen+ 16]

- cGANでは明示的に c と x のペアを与えていたが、暗黙的に x と対応する c を獲得したい。



- そのため、相互情報量 $I(c; x = G(c, z))$ が高くなるように学習する。

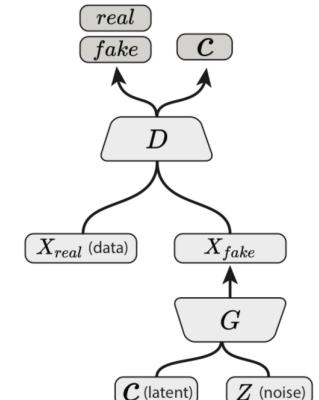
- 実際はその下界を正則化項として加える。

$$L_I(G, Q) = E_{c \sim p(c), x \sim G(z, c)} [\log Q(c|x)] \leq I(c; G(c, z))$$

- $Q(c|x)$ はニューラルネットワークとする (D と重みを共有) 。

$$V_{D,Q}(D, G, Q) = E_{p_{data}(x)} [\log D(x)] + E_{p(z)} [\log (1 - D(G(z)))] - \lambda L_I(G, Q)$$

$$V_G(D, G, Q) = -E_{p(z)} [\log (D(G(z)))] + \lambda L_I(G, Q)$$



InfoGAN
(Chen, et al., 2016)

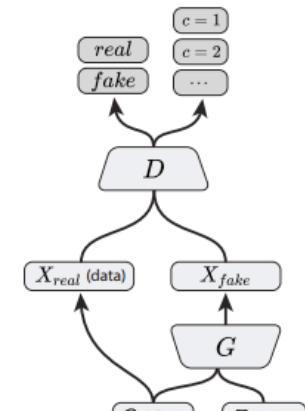
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	7
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

(a) Varying c_1 on InfoGAN (Digit type)

ACGAN

■ Conditional Image Synthesis With Auxiliary Classifier GANs[Odena+ 16]

- 綺麗な画像 x は、クラス c の識別性も高いはず。
 - 多様性と識別性を高めたい。
- G を c で条件づけて、 D で c を識別する補助分類器 $Q(C = c|x)$ を追加。
 - cGANの G とinfoGANの D 。
 - G を c で条件づけることで、学習する分布の峰を少なくできる。
- 128×128の画像を生成。



AC-GAN
(Present Work)

$$V_{D,Q}(D, G, \textcolor{red}{Q}) = E_{p_{data}(x)}[\log D(x)] + E_{p(z)}\left[\log(1 - D(G(z)))\right] + \textcolor{red}{E}[Q(C = c|x)] + E[Q(C = c|G(z))]$$

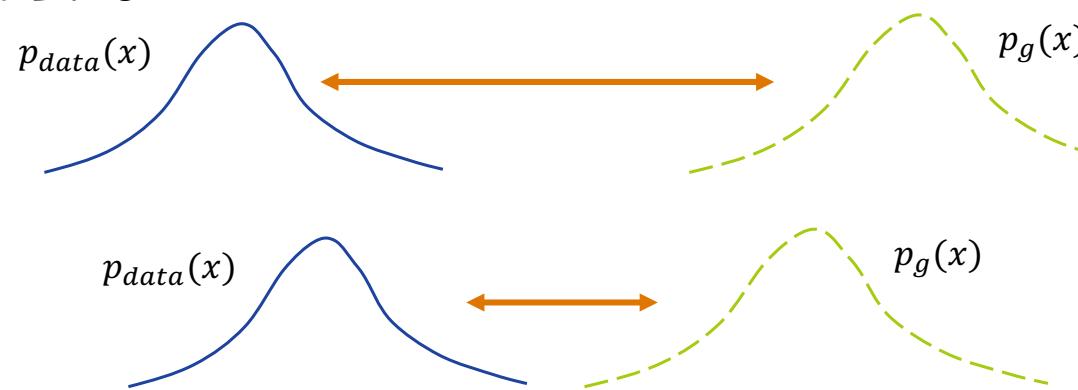
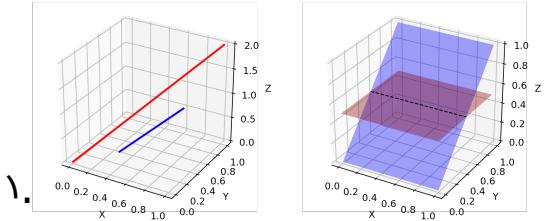
$$V_G(D, G, \textcolor{red}{Q}) = -E_{p(z)}\left[\log(D(G(z)))\right] - \textcolor{red}{E}[Q(C = c|G(z))]$$

GANのダイバージェンスの変更

- GANはJSダイバージェンスを最小化している。
 - 通常の生成モデルの最尤推定（KL最小化）から解放されている。
->別のダイバージェンスで最適化できないか？
- これまでに様々なダイバージェンス（距離）のGANが提案されている。
 - ピアソンの χ^2 ダイバージェンス : LSGAN
 - f ダイバージェンス : f-GAN
 - Bregmanダイバージェンス（真の密度比との距離） : b-GAN[Uehara+ 17]
 - **Wasserstein距離** : WGAN[Arjovsky+ 17], WGAN-GP[Gulrajani+ 17], BEGAN[Berthelot+ 17]
 - MMD : MMD GAN [Li+ 17]
 - Cramér距離 : Cramér GAN[Bellemare+ 17]
 - その他のIntegral Probability Metrics : McGAN[Mroueh+ 17], Fisher GAN[Mroueh+ 17]
- などなど。

JSダイバージェンスの限界

- データは次元のあらゆるところに存在する訳ではない.
 - 実際は低次元な多様体として存在する.
 - 高次元空間の場合、2つの分布の低次元多様体が交わらない可能性が高い.
- 分布の台が交わらない場合、 D は完全に $p_{data}(x)$ と $p_g(x)$ が分離できてしまう.
 - JSダイバージェンスには交わらない分布間の距離の違いがわからない（定数になる）.
 - G の方向が決められない！！



Wasserstein距離

■ Earth Mover距離 (Wasserstein-1距離)

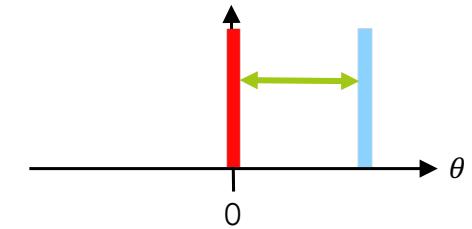
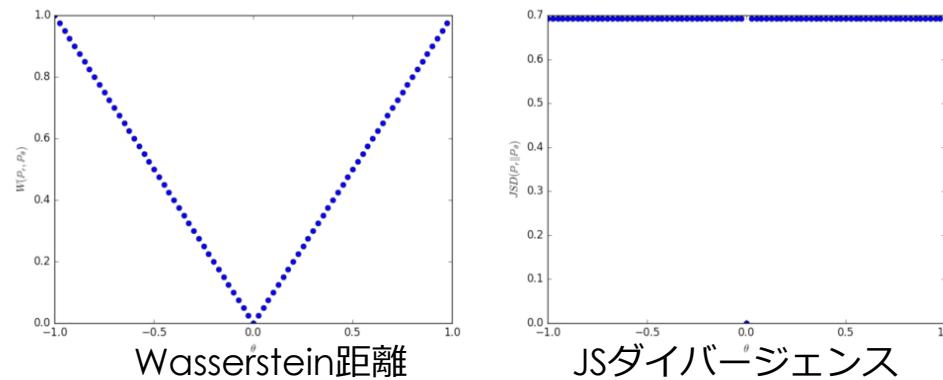
$$\inf_{\gamma \in (p_{data}, p_g)} E_{(x,y) \sim \gamma} [|x - y|]$$

□ 直感的には、 p_{data} から p_g に確率密度を移すときの最小コスト.

□ γ が輸送量、 $||x - y||$ が移動距離に対応.

■ JSダイバージェンスとの比較

□ 直線の確率密度分布を移動した時の距離 (右図)



Wasserstein GAN

- Wasserstein距離は双対表現として、次のように書ける。

$$W(p_g, p_{data}) = \max_{\varphi} \{ E_{p_{data}}[D(x; \varphi)] - E_{p(z)}[D(G(z); \varphi)] \}$$

ただし $\|D(x) - D(y)\| \leq \|x - y\|$

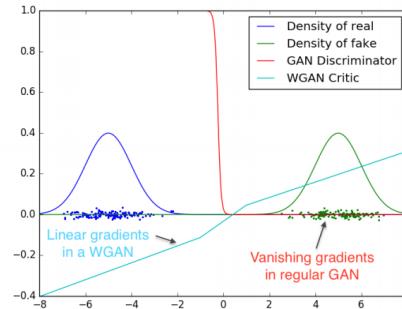
- Wasserstein GAN (WGAN) では、これを G の目的関数とする。 1-リップシツツ連續

- 通常のGANとほぼ同じ！ただし・・・
 - D の出力はそのまま（非線形関数を加えない）。
 - D がリップシツツ性を満たすように学習する。
 - 毎回 φ を複数回更新して、Wasserstein距離を正確に近似するようにする。

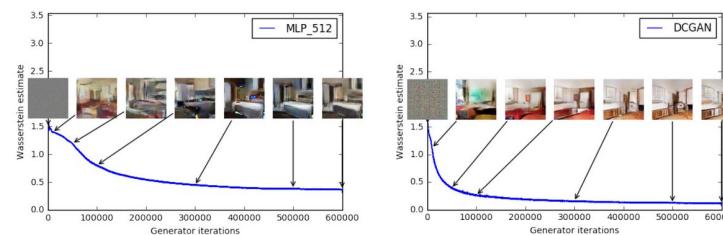
$$\begin{aligned} V_D(D, G) &= E_{p_{data}(x)}[D(x; \varphi)] - E_{p(z)}[D(G(z; \theta); \varphi)] \\ V_G(D, G) &= -E_{p(z)}[D(G(z; \theta); \varphi)] \\ &\quad \text{s.t. } \|D(x) - D(y)\| \leq \|x - y\| \end{aligned}$$

WGANの利点

- 従来のダイバージェンスに起因するmode collapseなどの問題が発生しない.
- 台が交わらない分布でも、勾配が得られる.



- 目的関数が意味を持つようになる.
- 従来のGANは、どれくらい画像が学習できているか、客観的に計測できなかった.



課題：リップシツ性の担保

- Dのリップシツ性を保つために、パラメータを制約する必要がある。
 - 方法1：パラメータをclippingする。
 - φ を $[-0.01, 0.01]$ の範囲に収める。
 - ただし、勾配爆発や消失の問題が生じる可能性がある。
 - 方法2：gradient penaltyの追加[Gulrajani+ 17]
 - Dの勾配ノルムが1になるような制約項を追加して学習。
 - ただし、制約が厳しすぎるという指摘もあり[Kodali+ 17]。
- その後も、様々なintegral probability metricsによるGANが提案されている。
 - Dの制約を変更することで、目的関数が様々な距離に対応するようになる。
 - MMD[Li+ 17], Cramér距離[Bellemare+ 17]などなど。

GANの評価

生成モデルの評価とGAN

- 生成モデルの場合、学習したモデルを評価するのが非常に困難。
 - 教師あり学習では、テストデータ(x, y)でうまく予測できるかを評価すればいい。
 - 教師なし学習では、 $x \rightarrow z$ などは自明ではない。
- よくある方法は、テストデータに対する（対数）尤度の値を調べる。
 - VAEでは下界（正確にはimportance weighted AE[Burda+ 15]の下界）を計算する。
- GANは明示的に分布を持っていないので、尤度の評価が困難。
 - 初期はパリエツエン窓密度推定などが使われていた。
 - しかし、評価としては役に立たない場合が多いことが示された[Theis+ 15]
- しかし、そもそも対数尤度が高ければ良い生成モデルといえるのか？？
 - 最初に述べたように、KLはぼやけた画像を高く評価する。
 - GANは、尤度の代わりにモデルに応じて独自の評価指標（識別器）で学習している。

-> 客観的にGANの評価は可能なのか？？

Inception Score

- ▣ 良い生成画像とは何か ? [Salimans+ 16]
 1. 高い信念でクラス分類ができる->エントロピー $p(y|x)$ が小さい.
 2. サンプルのバリエーションが大きい->エントロピー $p(y)$ が大きい.
- ▣ よって次のスコアを調べればいい (inception score) .

$$IS(G) = \exp(E_{x \sim G} [p(y|x) || p(y)])$$

- ▣ G からのサンプル x を使って評価
- ▣ スコアが小さければ良いモデル.
- ▣ $p(y|x)$ は事前学習済みのinceptionモデル.
- ▣ 経験的に, 人間の評価と相関することがわかった.
- ▣ 現在GANで最もよく使われている評価指標.

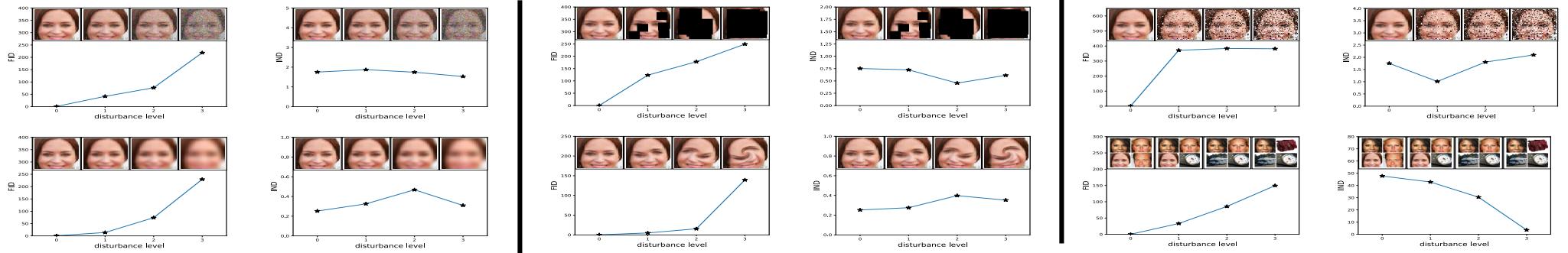
Fréchet Inception Distance

- ISとは異なる評価指標[Heusel+ 17].

- Inceptionモデルの任意の層（pool 3層）に， $p_{data}(x)$ と $p_g(x)$ からのサンプルを写像する.
- 埋め込んだ層を連続多変量ガウス分布と考えて，平均と共に分散を計算する.
- それらを用いてFréchet距離（Wasserstein-2距離）を計算する（Fréchet inception distance, FID）.

$$FID(data, g) = \|\mu_{data} - \mu_g\|_2^2 + Tr(\Sigma_{data} + \Sigma_g - 2(\Sigma_{data}\Sigma_g)^{\frac{1}{2}})$$

- IS（ここではinception distance）と比べて，適切な評価指標になっている.



結局どのGANがいい？

- ▣ 安定性や収束性の問題はいまだに解決されていない。
 - ▣ GANはVAEに比べて不安定で、ハイパーパラメータ等に対する分散が非常に大きい。
 - ▣ 計算コストをかけて適切にパラメータチューニングすれば、GANの種類であまり差はない[Lucic+ 17]。
- ▣ 自分で動かして確認するのが重要。
 - ▣ ほとんどの場合、目的関数の変更と多少NNアーキテクチャを変更すればよい。
 - ▣ 最近はまとまった実装やライブラリが公開されている。
 - ▣ <https://github.com/hwalsuklee/tensorflow-generative-model-collections>
 - ▣ <https://github.com/pfnet-research/chainer-gan-lib>
 - ▣ <https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/gan>
- ▣ 自分の目的にあったGANを選ぶ。
 - ▣ 具体的な目的ベースで提案されたGANも多い（GAN Zooやこの後のスライドを参照）。

GANの応用

高解像度の画像生成

- ▣ 段階的に解像度を上げるように学習することで、高解像度の画像生成が可能になっている。
 - ▣ LAPGAN[Denton+ 15], StackGAN[Zhang+ 16], StackGAN++[Zhang+ 17]
- ▣ Progressive GAN[Karras+ 17]
 - ▣ 低解像度の生成から学習していく（最初はモードが少ないので楽）。
 - ▣ 安定して高解像度（1024x1024！）の画像を生成できる。

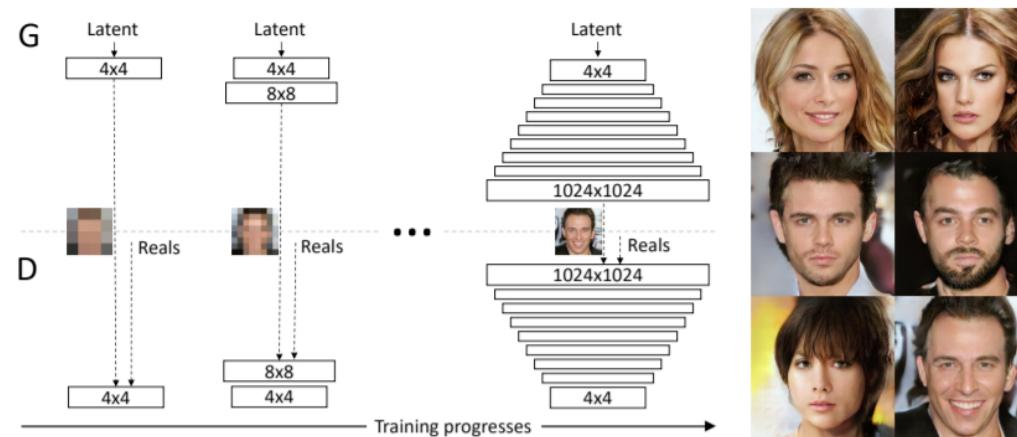
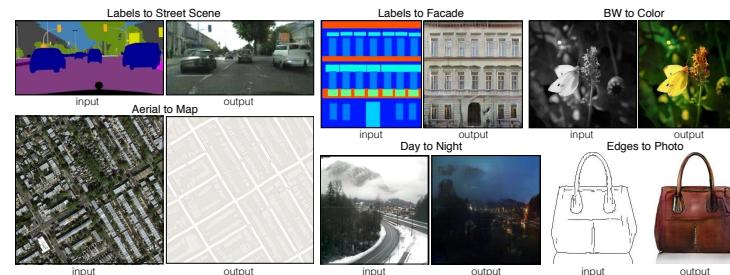


Image-to-image translation

- 画像から対応する画像を生成する。

- Pix2pix[Isola+ 16]

- Conditional GANの条件付けを変換前の画像とする。
 - Gをautoencoderにする。



- BicycleGAN[Zhu+ 17]

- 決定論的な対応を学習するのではなく、潜在変数の分散を考慮したモデル。

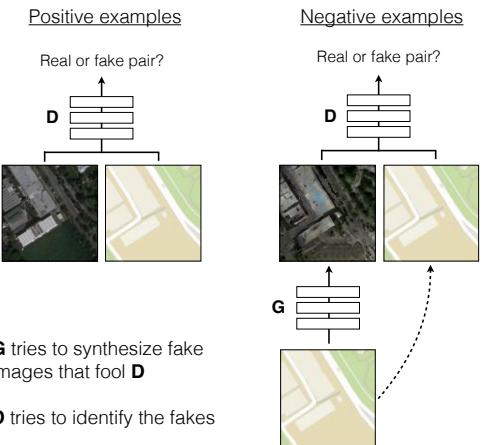
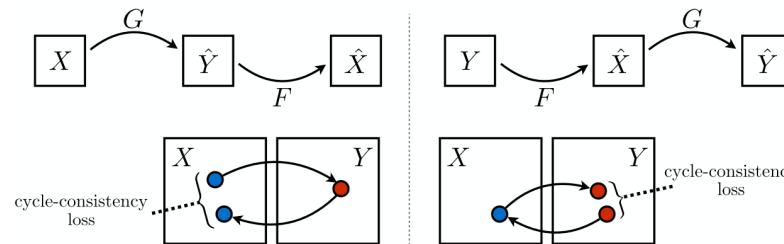


Image-to-image translation

- ペアになっていないデータで双方向に変換

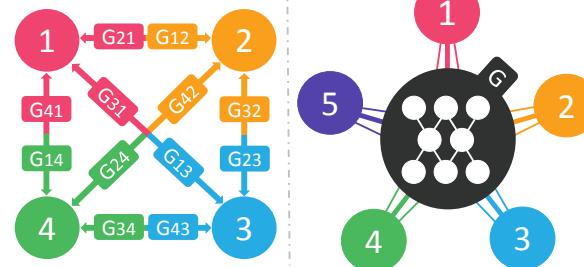
- CycleGAN[Zhu+ 17]



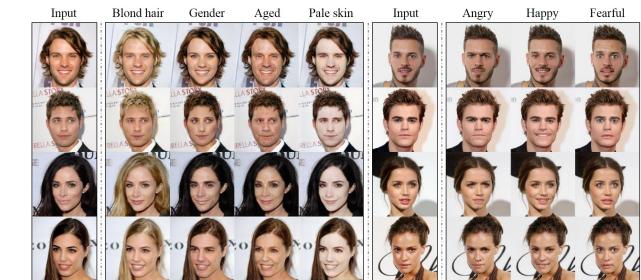
- StarGAN[Choi+ 17]

- 2つのドメインではなく、複数のドメイン間で変換.
- GとDは1つだけ.

(a) Cross-domain models



(b) StarGAN



Actor-criticとGAN

Actor-critic

- 方策 (actor) と価値関数 (critic) を同時に学習する手法.

- MDPの下で (記号の説明を省略) . . .

- 行動価値関数は, $Q^\pi(s, a) = \mathbb{E}_{s_{t+k} \sim \mathcal{P}, r_{t+k} \sim \mathcal{R}, a_{t+k} \sim \pi} \left[\sum_{k=1}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a \right]$

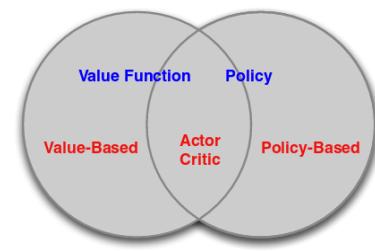
- 方策の更新 : $\pi^* = \arg \max_{\pi} \mathbb{E}_{s_0 \sim p_0, a_0 \sim \pi} [Q^\pi(s_0, a_0)]$

- 行動価値関数の更新 : $Q^\pi = \arg \min_Q \mathbb{E}_{s_t, a_t \sim \pi} [\mathcal{D}(\mathbb{E}_{s_{t+1}, r_t, a_{t+1}} [r_t + \gamma Q(s_{t+1}, a_{t+1})] || Q(s_t, a_t))]$

- よって, 2段階の最適化アルゴリズムになる.

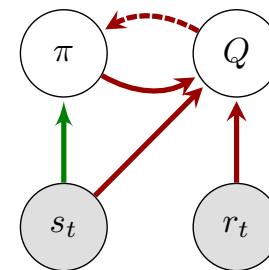
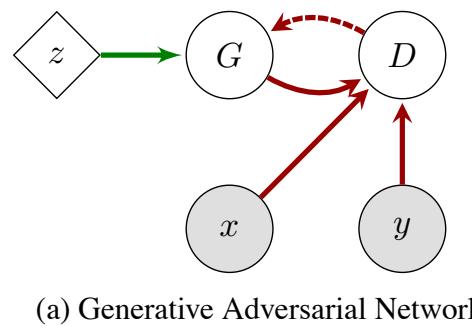
$$\begin{aligned} F(Q, \pi) &= \mathbb{E}_{s_t, a_t \sim \pi} [\mathcal{D}(\mathbb{E}_{s_{t+1}, r_t, a_{t+1}} [r_t + \gamma Q(s_{t+1}, a_{t+1})] || Q(s_t, a_t))] \\ f(Q, \pi) &= -\mathbb{E}_{s_0 \sim p_0, a_0 \sim \pi} [Q^\pi(s_0, a_0)] \end{aligned}$$

- なんかGANと似ている？？



Actor-criticとGAN

- Actor-criticとGANを同じ図式で並べてみる[Pfau+ 17].



(b) Deterministic Policy Gradient [7] /
SVG(0) [8] / Neurally-Fitted Q-learning
with Continuous Actions [9]

- G =方策, D =価値関数とみなせる.
- ただし, GANでは G が x （現在の状態）を受け取らない.
 - z からのランダム.

双方のテクニックを利用できないか？

Method	GANs	AC
Freezing learning	yes	yes
Label smoothing	yes	no
Historical averaging	yes	no
Minibatch discrimination	yes	no
Batch normalization	yes	yes
Target networks	n/a	yes
Replay buffers	no	yes
Entropy regularization	no	yes
Compatibility	no	yes

- 著者のPfau氏はUnrolled GAN[Metz+ 17]を推している。
 - というかUnrolled GAN提案者の一人。

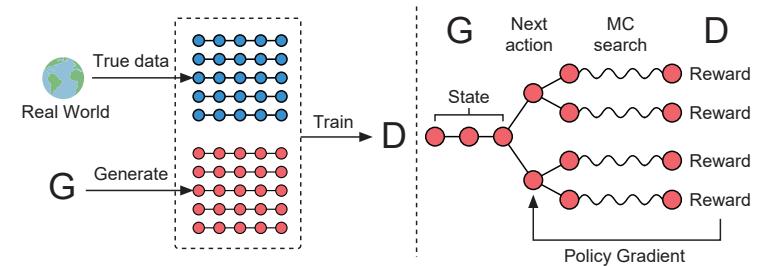
SeqGAN

- SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient[Yu+ 17]
 - 系列情報を生成するGAN
- 系列情報を学習するのは困難
 - 系列情報は離散.
 - 識別器は完全な系列しか評価できないため，部分的に生成された系列の評価が困難（最終的なスコアが最大になるようにGを誘導したい）.

■ 解決策：強化学習の利用

- Gを方策，Dを報酬とする.
- Dから価値関数を求める.

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N} \sum_{n=1}^N D_\phi(Y_{1:T}^n), \quad Y_{1:T}^n \in \text{MC}^{G_\beta}(Y_{1:t}; N) & \text{for } t < T \\ D_\phi(Y_{1:t}) & \text{for } t = T, \end{cases}$$



SeqGAN : 擬似コード

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ ; roll-out policy G_β ; discriminator D_ϕ ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

- 1: Initialize G_θ , D_ϕ with random weights θ, ϕ .
- 2: Pre-train G_θ using MLE on \mathcal{S}
- 3: $\beta \leftarrow \theta$
- 4: Generate negative samples using G_θ for training D_ϕ
- 5: Pre-train D_ϕ via minimizing the cross entropy
- 6: **repeat**
- 7: **for** g-steps **do**
- 8: Generate a sequence $Y_{1:T} = (y_1, \dots, y_T) \sim G_\theta$
- 9: **for** t in $1 : T$ **do**
- 10: Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
- 11: **end for**
- 12: Update generator parameters via policy gradient Eq. (8)
- 13: **end for**
- 14: **for** d-steps **do**
- 15: Use current G_θ to generate negative examples and combine with given positive examples \mathcal{S}
- 16: Train discriminator D_ϕ for k epochs by Eq. (5)
- 17: **end for**
- 18: $\beta \leftarrow \theta$
- 19: **until** SeqGAN converges

IRLとGAN

逆強化学習

- Inverse reinforcement learning (IRL)
 - 目標の行動（最適戦略）から報酬関数を推定する手法.



- Maximum entropy IRL (MaxEnt IRL) [Ng+ 00]
 - 軌道 $\tau = \{\mathbf{x}_1, \mathbf{u}_1, \dots, \mathbf{x}_T, \mathbf{u}_T\}$ がボルツマン分布に従うと考える.

$$p_{\theta}(\tau) = \frac{1}{Z} \exp(-c_{\theta}(\tau)) \quad c_{\theta}(\tau) = \sum_t c_{\theta}(\mathbf{x}_t, \mathbf{u}_t)$$

コスト関数（報酬関数）

- 最適な軌道で尤度が最大になると考える -> 尤度最大化
- しかし、分配関数Zをどうする・・・？

逆強化学習

■ Guided cost learning (GCL) [Finn+ 16]

- 分配関数を推定するために、提案分布 $q(\tau)$ を考えて重点サンプリングによって求める。

$$\begin{aligned}\mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau \sim p}[-\log p_{\theta}(\tau)] = \mathbb{E}_{\tau \sim p}[c_{\theta}(\tau)] + \log Z \\ &= \mathbb{E}_{\tau \sim p}[c_{\theta}(\tau)] + \log \left(\mathbb{E}_{\tau \sim q} \left[\frac{\exp(-c_{\theta}(\tau))}{q(\tau)} \right] \right)\end{aligned}$$

- 提案分布が高い尤度で軌道をカバーできないと重点サンプリングが高バリアンスになる可能性があるので、デモンストレーション \tilde{p} も利用する。

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau \sim p}[c_{\theta}(\tau)] + \log \left(\mathbb{E}_{\tau \sim \mu} \left[\frac{\exp(-c_{\theta}(\tau))}{\frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)} \right] \right) \quad \mu = \frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)$$

- q は次のコストを最小化して求める（詳細は省略）。

$$\mathcal{L}_{\text{sampler}}(q) = \mathbb{E}_{\tau \sim q}[c_{\theta}(\tau)] + \mathbb{E}_{\tau \sim q}[\log q(\tau)]$$

-> θ と q について交互に最適化すればいい。

GANの再解釈

- GANの最適な識別器は $D^*(\tau) = \frac{p(\tau)}{p(\tau) + q(\tau)}$
- ここでモデル分布 q の確率密度は推定できるとする（GANの前提を変える）。
- さらに真の分布 p をコスト関数でパラメータ化したものに置き換える。

$$D_\theta(\tau) = \frac{\tilde{p}_\theta(\tau)}{\tilde{p}_\theta(\tau) + q(\tau)} = \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)}$$

- するとGANの識別器の損失関数は

$$\mathcal{L}_{\text{discriminator}}(D_\theta) = \mathbb{E}_{\tau \sim p} \left[-\log \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)} \right] + \mathbb{E}_{\tau \sim q} \left[-\log \frac{q(\tau)}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)} \right]$$

- Dのパラメータ=コスト関数のパラメータ
- $\frac{1}{Z} \exp(-c_\theta(\tau)) = p(\tau)$ のとき最適。
- [再掲]MaxEnt IRLのコストは ($\tilde{p}(\tau) = p_\theta(\tau) = \frac{1}{Z} \exp(-c_\theta(\tau))$ とすると)

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau \sim p} [c_\theta(\tau)] + \log \left(\mathbb{E}_{\tau \sim \mu} \left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2Z} \exp(-c_\theta(\tau)) + \frac{1}{2} q(\tau)} \right] \right)$$

GANとIRLの関係

- 次のことが証明されている[Finn+ 17].
 - 識別器の目的関数を最小化するZは、分配関数の重点サンプリングに対応する.

$$\mathcal{L}_{\text{discriminator}}(D_\theta) = \mathbb{E}_{\tau \sim p} \left[-\log \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)} \right] + \mathbb{E}_{\tau \sim q} \left[-\log \frac{q(\tau)}{\tilde{\mu}(\tau)} \right] \quad \Rightarrow \quad Z = \mathbb{E}_{\tau \sim \mu} \left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)} \right]$$

- このZで、識別器の目的関数の勾配 = MaxEnt IRLの目的関数の勾配となる.

$$\begin{aligned} \partial_\theta \mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau \sim p} [\partial_\theta c_\theta(\tau)] + \partial_\theta \log \left(\mathbb{E}_{\tau \sim \mu} \left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)} \right] \right) \\ &= \mathbb{E}_{\tau \sim p} [\partial_\theta c_\theta(\tau)] + \left(\mathbb{E}_{\tau \sim \mu} \left[\frac{-\exp(-c_\theta(\tau)) \partial_\theta c_\theta(\tau)}{\tilde{\mu}(\tau)} \right] \right) / \mathbb{E}_{\tau \sim \mu} \left[\frac{\exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)} \right] \\ &= \mathbb{E}_{\tau \sim p} [\partial_\theta c_\theta(\tau)] - \mathbb{E}_{\tau \sim \mu} \left[\frac{\frac{1}{Z} \exp(-c_\theta(\tau)) \partial_\theta c_\theta(\tau)}{\tilde{\mu}(\tau)} \right] \\ &= \partial_\theta \mathcal{L}_{\text{discriminator}}(D_\theta). \end{aligned}$$

- 生成器の目的関数 = MaxEnt IRLの提案分布の目的関数

$$\begin{aligned} \mathcal{L}_{\text{generator}}(q) &= \mathbb{E}_{\tau \sim q} [\log(1 - D(\tau)) - \log(D(\tau))] \\ &= \mathbb{E}_{\tau \sim q} \left[\log \frac{q(\tau)}{\tilde{\mu}(\tau)} - \log \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\tilde{\mu}(\tau)} \right] \\ &= \mathbb{E}_{\tau \sim q} [\log q(\tau) + \log Z + c_\theta(\tau)] \\ &= \log Z + \mathbb{E}_{\tau \sim q} [c_\theta(\tau)] + \mathbb{E}_{\tau \sim q} [\log q(\tau)] = \log Z + \mathcal{L}_{\text{sampler}}(q). \end{aligned}$$

GANとIRLの関係

- つまり,
 - コスト関数（報酬関数）の最小化は D の最適化とみなせる.
 - 提案分布 q は G とみなせる.
- GAN = IRLを示すと何がうれしい?
 - q は密度推定できるので、直接最大化できるのでは？
 - q を最尤推定 (=KL最小化) で求めたくない（peakがないところにも分布を置くので）.
 - GANとみなすと、様々なダイバージェンスで考えることができそう.
- その他の研究
 - 模倣学習とGAN (GAIL[Ho+ 17])
 - OptionGAN[Henderson+ 17]
 - InfoGAIL[Li+ 17]

深層生成モデルと強化学習

LeCun先生曰<



■ "Pure" Reinforcement Learning (cherry)

- ▶ The machine predicts a scalar reward given once in a while.
- ▶ **A few bits for some samples**

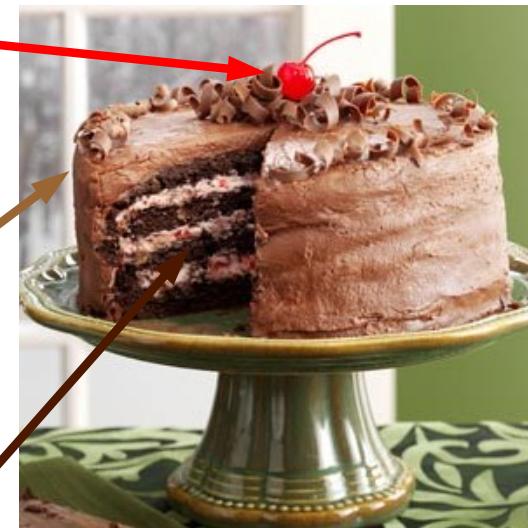
■ Supervised Learning (icing)

- ▶ The machine predicts a category or a few numbers for each input
- ▶ Predicting human-supplied data
- ▶ **10→10,000 bits per sample**

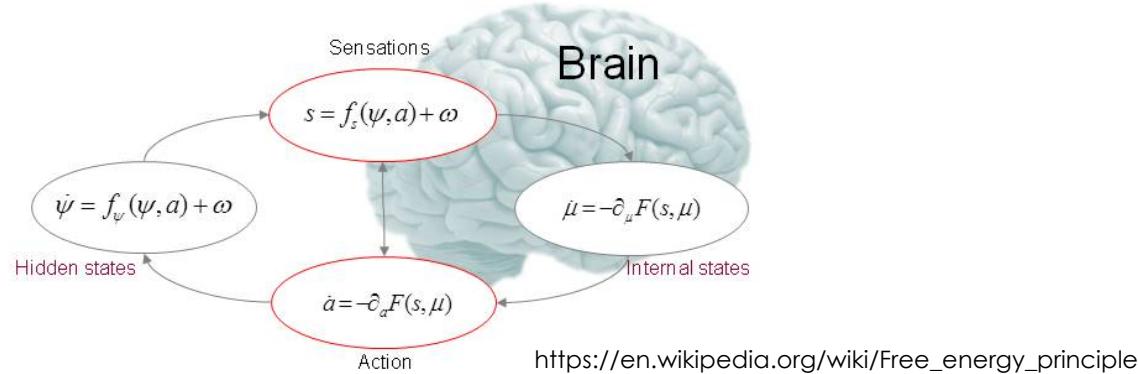
■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.
- ▶ Predicts future frames in videos
- ▶ **Millions of bits per sample**

■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)



Fristonの自由エネルギー原理



- 簡単に言うと
 - 状態を入力として生成モデルを変分推論で学習.
 - 学習した生成モデルの尤度が最大になるように行動を選択.
- ただし、生成モデルが全ての状態で学習しきっていることを前提としている.
 - そうしないと、エージェントが身動き取れなくなる.
 - 最新のサーベイでは触れているかもしれません（まだ読んでないです・・・）.
- このような枠組みは、単純なタスクでは不利.
 - 長期的に様々なタスクを解くような問題設定を考える必要性.

強化学習と生成モデルの共通の問題

□ 离散分布の期待値の勾配

- REINFORCEは不偏推定量だがバリアンスが大きい
- VAEでは、不偏推定量でバリアンスが小さい再パラメータ化トリック[Kingma+ 14]を使う.
- 离散用には、Gumbel-softmax[Jang+ 16]が提案されている（ただし温度パラメータ次第で不偏推定量ではないこともある）.
- 最近では、REINFORCE + Gumbel-softmax的なRELAX[Grathwohl+ 17]が提案されている（不偏推定量かつ低バリアンス）.

□ 分配関数

- 従来の生成モデル（無向モデル）は分配関数の計算が必要だった.
 - 推定のためのコストがかかる.
 - 例：RBMのnegative phase
- 方策勾配法では、方策の定義に分配関数が入ってくる.
- GANのように自身の評価を他に任せれば、分布の定義が必要なくなる = 分配関数の計算が不要.

まとめ

- GANの説明と強化学習との関係を説明した.
- 今回話せなかったGANの重要トピック
 - GANとVAEの関係・拡張
 - GANとドメイン適応
 - GANと半教師あり学習
- 深層生成モデル + 深層強化学習が今後重要なのは明らか.
->双方の知見を生かした研究を進めていくべき