

S P R I N T 2 6

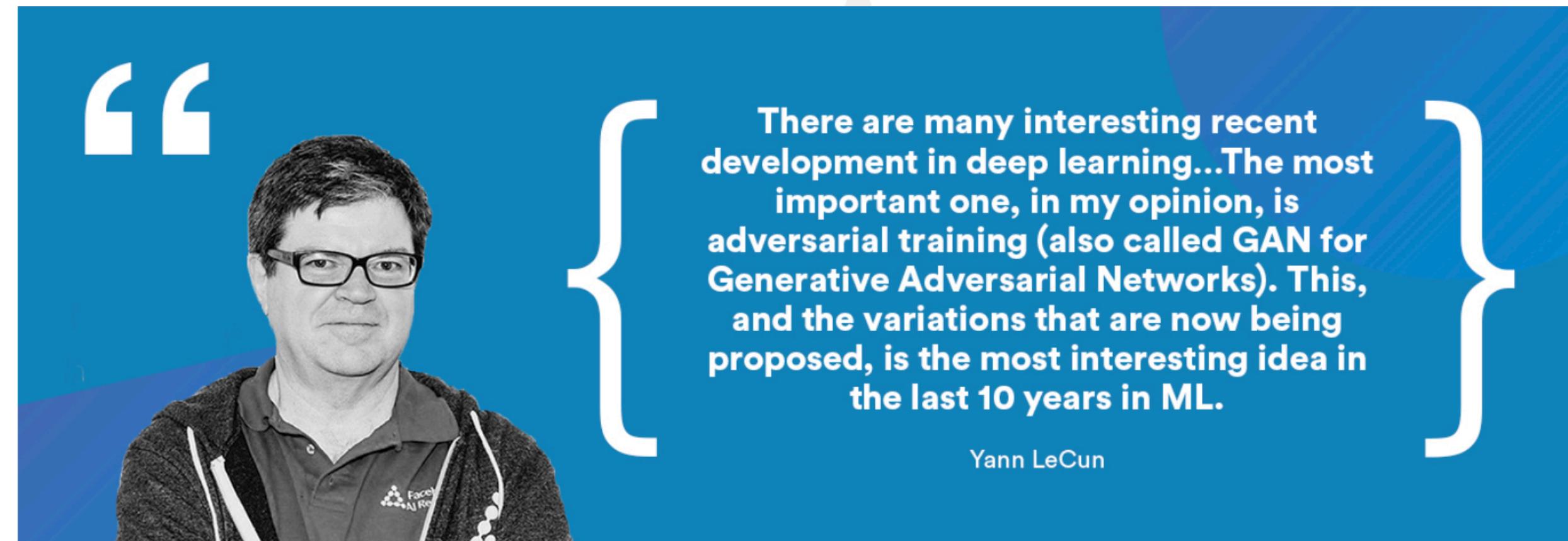


Ian Goodfellow: Adversarial Machine Learning (ICLR 2019 invited talk)

<https://www.youtube.com/watch?v=sucqskXRkss>

Generative Adversarial Networks (GAN)は、2014年にIan Goodfellowによって考案された**生成モデル**の一種。

<https://arxiv.org/pdf/1406.2661.pdf>



A History of Machine Learning and Deep Learning

<https://www.import.io/post/history-of-deep-learning/>

Yann LeCun⁽¹⁾氏による**GAN**の評価：

'the most interesting idea in the last ten years in machine learning'

(1) Yann LeCun氏（ニューヨーク大学教授）は、Geoffrey Hinton氏（トロント大学教授およびGoogle Brainの研究者）とYoshua Bengio氏（Element AIの創設者およびモントリオール大学教授）とともに、チューリング賞のAssociation for Computing Machinery (ACM)を受賞した。Ian Goodfellow氏はYoshua Bengio氏の弟子である。

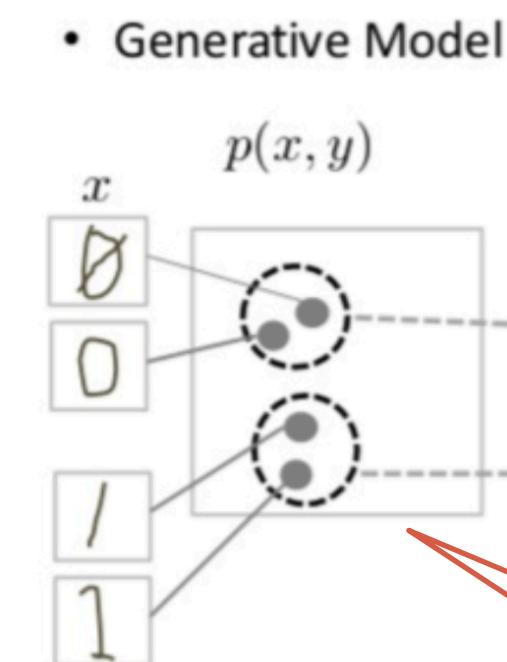
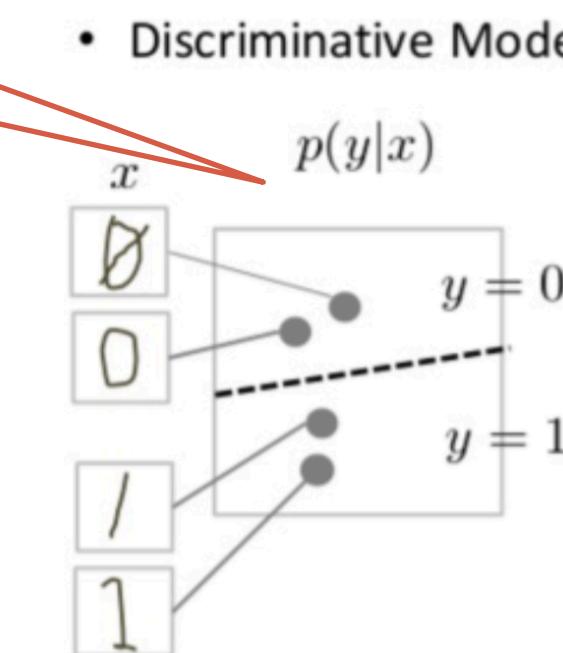
生成モデルとは？

識別モデルとの比較から捉えてみる。

生成モデルは、各クラスのデータの分布（データが空間全体にどのように配置されるか）を明示的にモデル化する。
他方で、識別モデルはクラス間の決定境界をモデル化する。

生成モデルとは？

クラスごとのパ
ターンの違いから境界
を得る



<https://developers.google.com/machine-learning/gan/generative>

いずれも条件付き確率⁽¹⁾ $p(Y|X)$ を予測しているが、それぞれ異なる確率を学習する。

生成モデルは、まず、 $p(Y)$ と $p(X|Y)$ をモデル化しパラメータを学習する。ここでベイズの定理を使用すると、条件付き確率 $p(Y|X)$ から Y を予測する⁽²⁾。この過程において、モデル化した $p(X|Y)$ からデータを生成することが可能。

識別モデルは、条件付き確率 $p(Y|X)$ をモデル化してパラメータを学習し、 Y を予測する。

(1) 条件付き確率：<http://arduino.pid.web.fc2.com/P9.html>

(2) ベイズの定理を用いた式変形：https://ahcweb01.naist.jp/lecture/2015/bda/3rd_oct23/bda2015_3rd.pdf p10

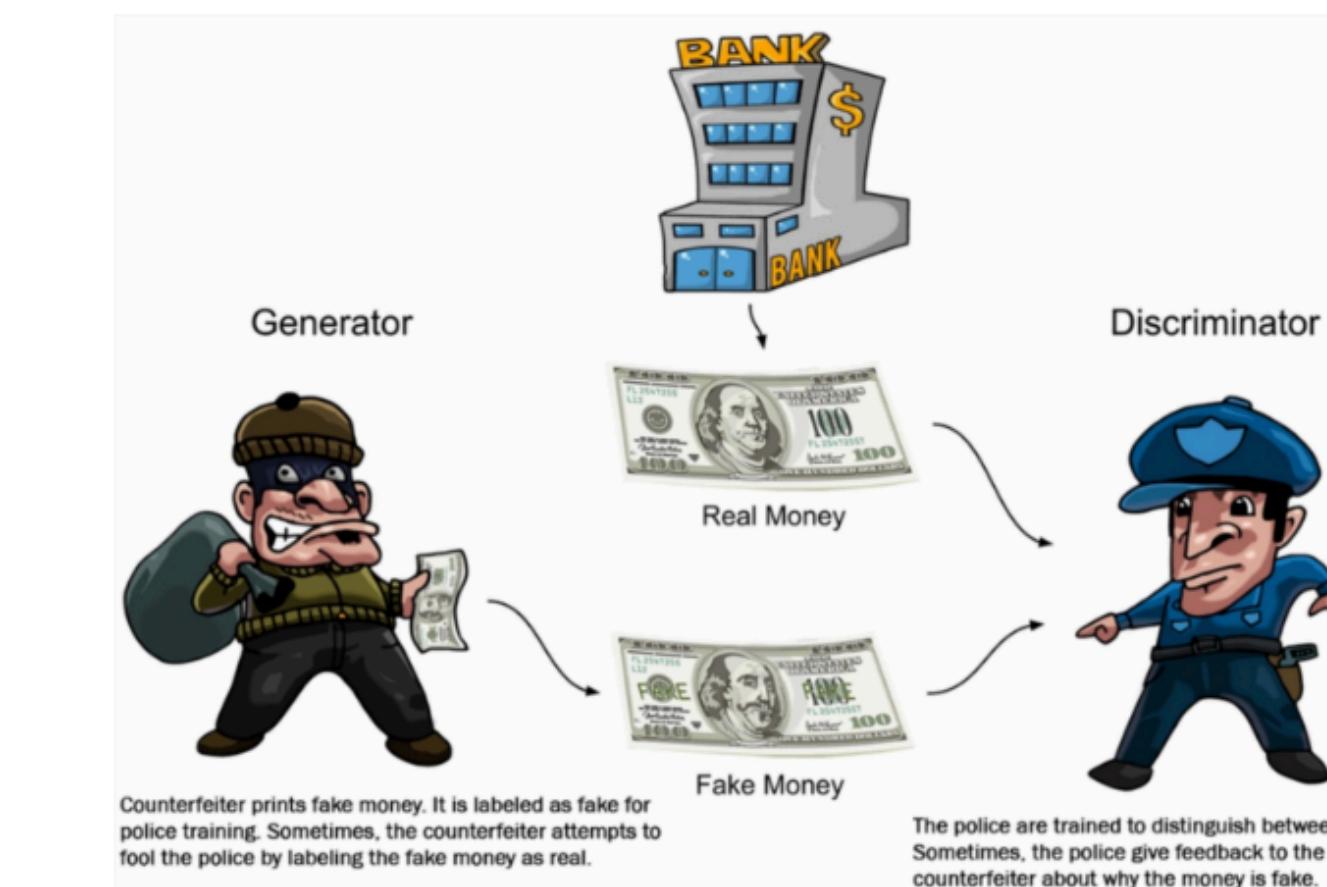
* $p(\cdot)$ はprobabilityの頭文字から「(事象)の起きる確率」を返す確率関数を表す

$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)} \propto P(X | Y)P(Y)$$

生成的敵対ネットワークとは？

Generative Adversarial Networks (GAN)は、「生成的敵対ネットワーク」と訳されるが、どんなアイデアか。

- 「生成的」のアイデア
- 「敵対」のアイデア



「生成的」のアイデア

「生成的」な部分の最初のアイデアは、ランダムなノイズベクトルを用いて**データの確率分布**を求める点である。

はじめはただのノイズであり、そこから生成されただけの画像は砂嵐のような使い物にならない画像である。

そこから、（例えば顔の画像を生成したい場合は）どの画像が顔であり、どの画像が顔でないのかについて学習する。

*深層生成モデル（ニューラルネットワークを採用した生成モデル）は、「額には目が表示されそうにない」「口の下に鼻はない」といった、画素間の強い**相関関係**を学習している。

「生成的」のアイデア

なぜデータの確率分布を求めるのか？

データの確率分布が求まれば、そこからデータを生成（サンプリング）できるから⁽¹⁾。

この背後には、データを高次元空間中の1点と考え、データが空間中でどのように分布しているのか（確率分布）を求めてることで、データ分布をモデル化することができる、という発想がある。

深層生成モデルでは、高次元空間上におけるデータ分布をモデル化するために、ニューラルネットワークでの実装が採用される。

Q：高次元空間中に分布するデータをモデル化する場合、その次元のべき乗に比例したサンプルがなければ推定が不安定になるのではないだろうか？

A：ノイズの次元数がデータの次元数より小さく、生成的ネットワークがニューラルネットワークの場合、生成されたデータは高次元空間中に低次元の薄い膜として分布すると考えられている。このようなデータ分布は低次元多様体と呼ばれる。深層生成モデルは、この低次元多様体を潜在表現として（Generatorの出力として）獲得できることがわかっている。

(1) GANの生成モデルは、実際は、画像データの確率分布ではなく、画像データそのものを出力する関数として実装される。理想的には、生成モデルは確率分布をニューラルネットワークで近似するという戦略のもと、その分布に従った揺らぎを持って確率論的に値を生成したいところだが、そのようには実装されない。GANの生成モデルの場合、同じノイズベクトルを渡せば、決定論的に同じ値が出力される。

「生成的」のアイデア

低次元多様体とは？

低次元多様体の説明：

「各点の周りがn次元的に拡がっているような空間を多様体と呼ぶ。…例えば、n次元ユークリッド空間自身はどの点でもn次元の拡がりを持った多様体である。また、球面というのは3次元空間中の2次元的な広がりを持つ多様体である。**球面全体は2次元と同相ではないが**（例えば世界地図では北極、南極で非連続になっている）、**各点の周りだけをみれば2次元とみなすことができる。**このことは私達が地球上でほとんど2次元の平面上にいるように考えていることからもいえる。…現実世界

で観測される多くのデータの分布がこのような低次元多様体として捉えられるという仮説を多様体仮説と呼ぶ。」

「画像や音声など自然界にみられるデータの多くは、可能な値の組み合わせのうちのほんの一部しか取らない。これらのデータはデータの見かけ上の次元数よりずっと少ないパラメータによって支配される空間に分布していると考えられている。
…多くのデータも高次元空間中に低次元の薄い膜として分布していると考えられている。」

<https://tech.nikkeibp.co.jp/dm/atcl/mag/15/00144/00031/>

「敵対」のアイデア

「敵対」の部分が由来するアイデアは、モデルを一種のコンテストとして設定したところである。

重要なのは、1つではなく2つの競合するネットワーク（**生成モデル**と**識別モデル**）を構築することだった。

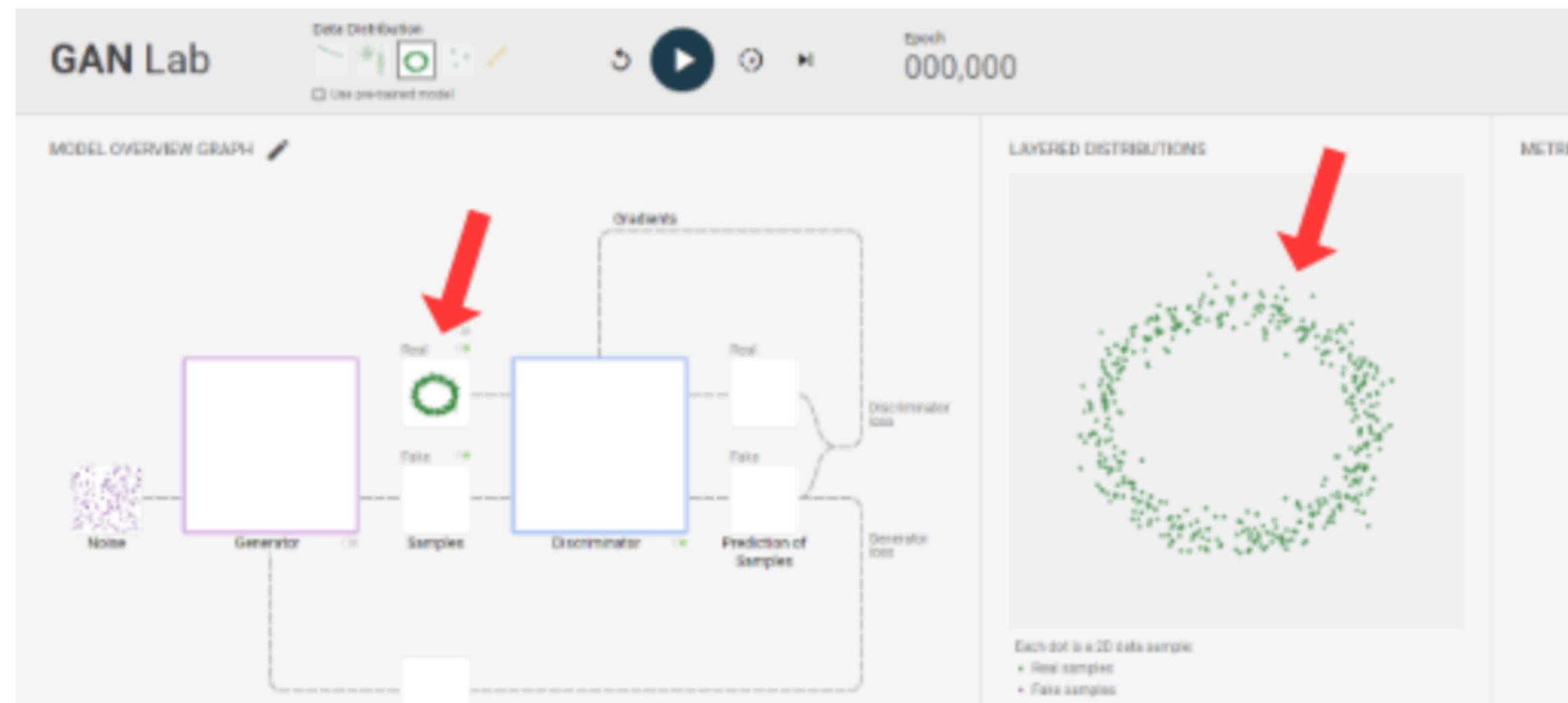
生成モデルはランダムな合成画像（たとえば、顔の画像）を作成しようとする。

識別モデルはこの合成画像と正解画像の入力（たとえば、有名人のデータベース）に対し、正解である確率を出力することによって、偽物と本物を見分けようとする。

「生成的敵対ネットワーク」の全体像を概観しよう

生成モデルは**Generator**、識別モデルは**Discriminator**と表記されている。

<https://poloclub.github.io/ganlab/>

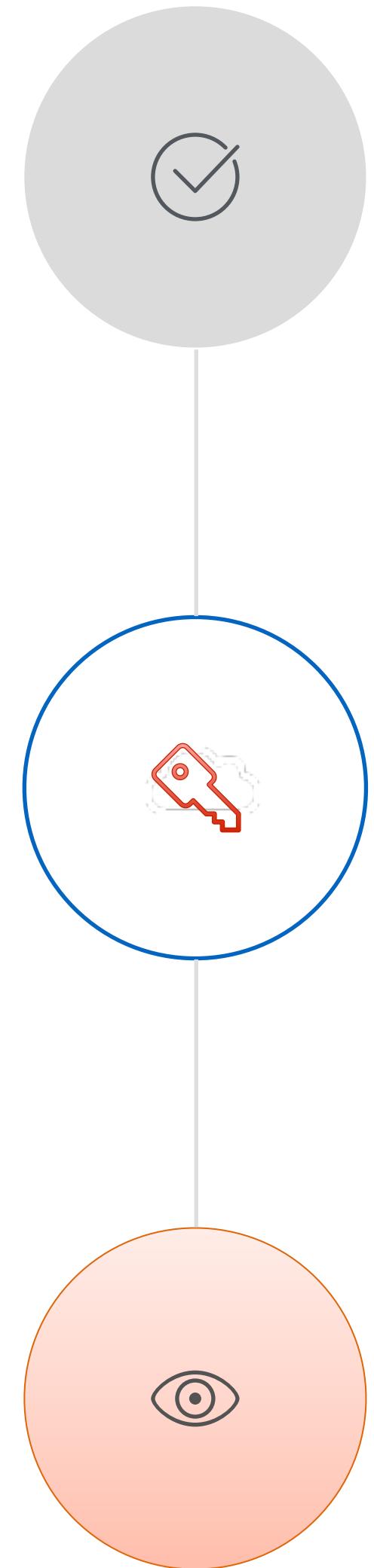


Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio‡
Departement d'informatique et de recherche opérationnelle ‐ Université de Montréal ‐ Montreal, QC H3C 3J7

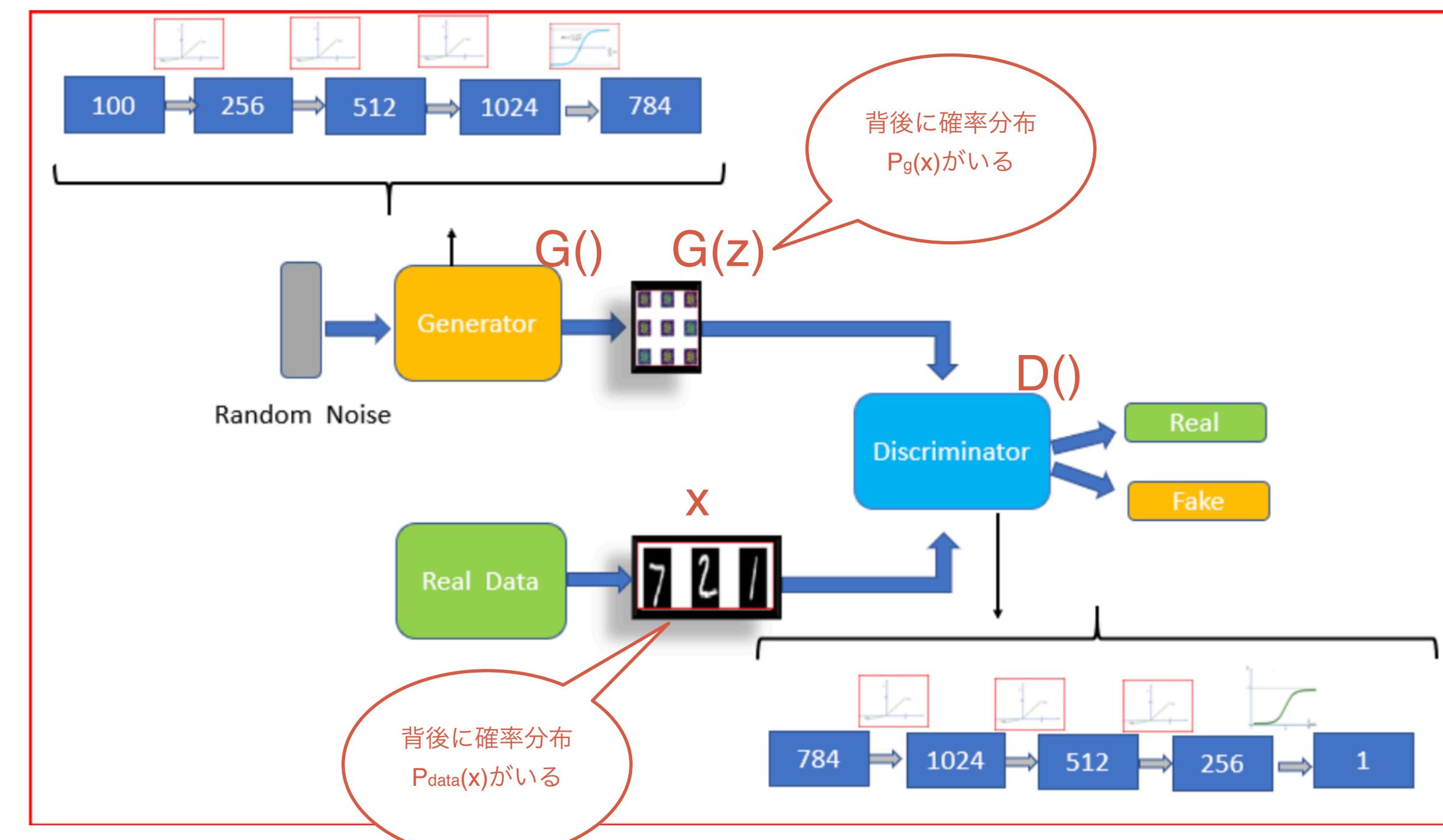
論文リンク:

<https://arxiv.org/pdf/1406.2661.pdf>



モデルの概念図

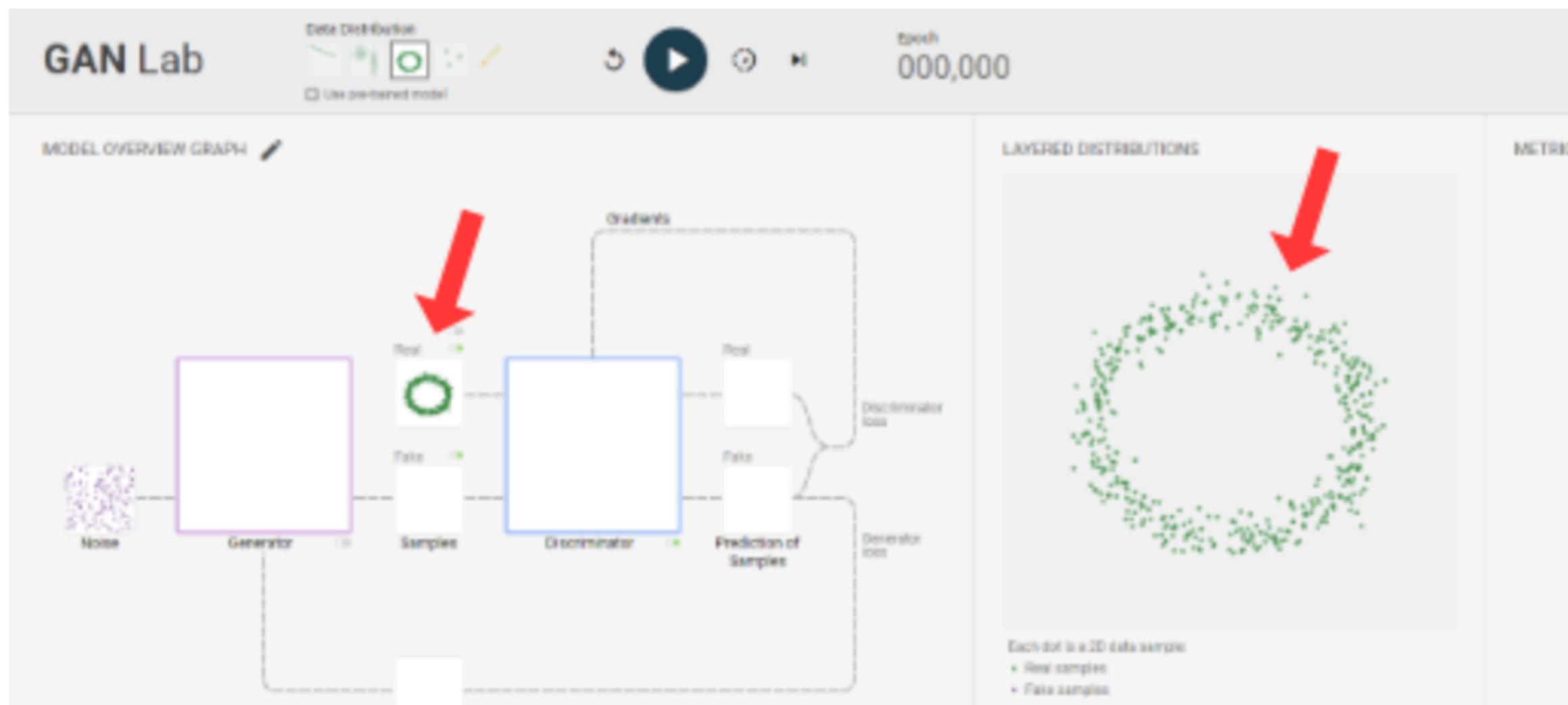
まず、一様分布から100次元のランダムベクトル（潜在変数） z を m 個（=（ミニバッチ数））サンプリングし⁽⁴⁾、**関数G()**によって正解データと同じドメインに写像する（ $G(z)$ ）。確率分布 $P_{\text{data}}(x)$ に従う m 個の画像データを x で表す。このとき、 $G(z)$ は x と同一の空間上で表現される合成画像で、これは背後に $P_g(G(z))$ という確率分布の存在を仮定している。この x と $G(z)$ に対し**関数D()**は本物の画像である確率を出力する。最終的には学習を通して、 $P_g = P_{\text{data}}$ となるような確率分布 $P_g(G(z))$ を求めればよい。⁽⁴⁾サンプリングとは、実装上では確率分布を用いて乱数を発生させることを指す。



識別モデルと生成分布の位置関係を見てみる。

再びGAN Labへ

<https://poloclub.github.io/ganlab/>



そもそも**確率分布**とは？

まず、ある確率を伴って生じる事象に数値（実数）を割り当てるものを**確率変数**という。

例えば、コインを1000回投げるとして、表が出た回数に注目し、その回数を X とおいた場合、 X の取りうる値の集合は、 $\{0, 1, \dots, 1000\}$ （要素数1001個）となる。この X は、コインの表が出ること（事象）に対し、実数值（0~1000のいずれか）を割り当てる関数とみなせる。このような X を確率変数と呼ぶ。

確率変数は、それぞれの実数值に対して、そのような事象が生じる確率を対応させるような**規則性**を伴っている。

この規則性を、表やヒストグラムで表現したものを**確率分布**（各数値の確率の分布）という。

[コインの確率分布の例] <https://math-fun.net/20190924/3017/>

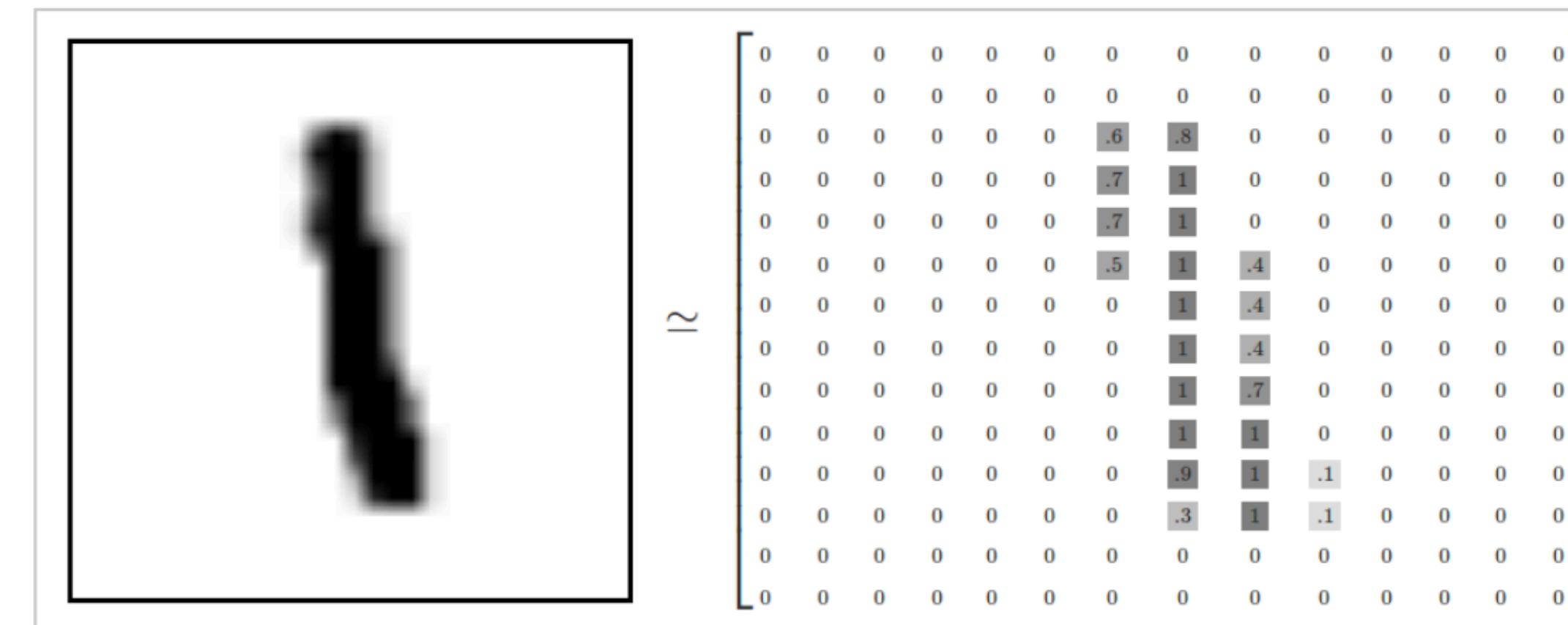
[確率分布曼荼羅] <http://www.math.wm.edu/~leemis/chart/UDR/UDR.html>

事例：MNISTデータの背後にある確率分布

ここで、MNISTデータを眺めてみる（上の図は0～255を0～1に正規化済み）。

1つのデータは28*28の784次元（784個の特徴量）からなる画像データで、0から9のいずれかの数字を表し、各ピクセルは0~1の間の値を持つ。

この値は白黒の濃淡を表し、0が黒、1が白を表す（上の図は白黒反転している）。



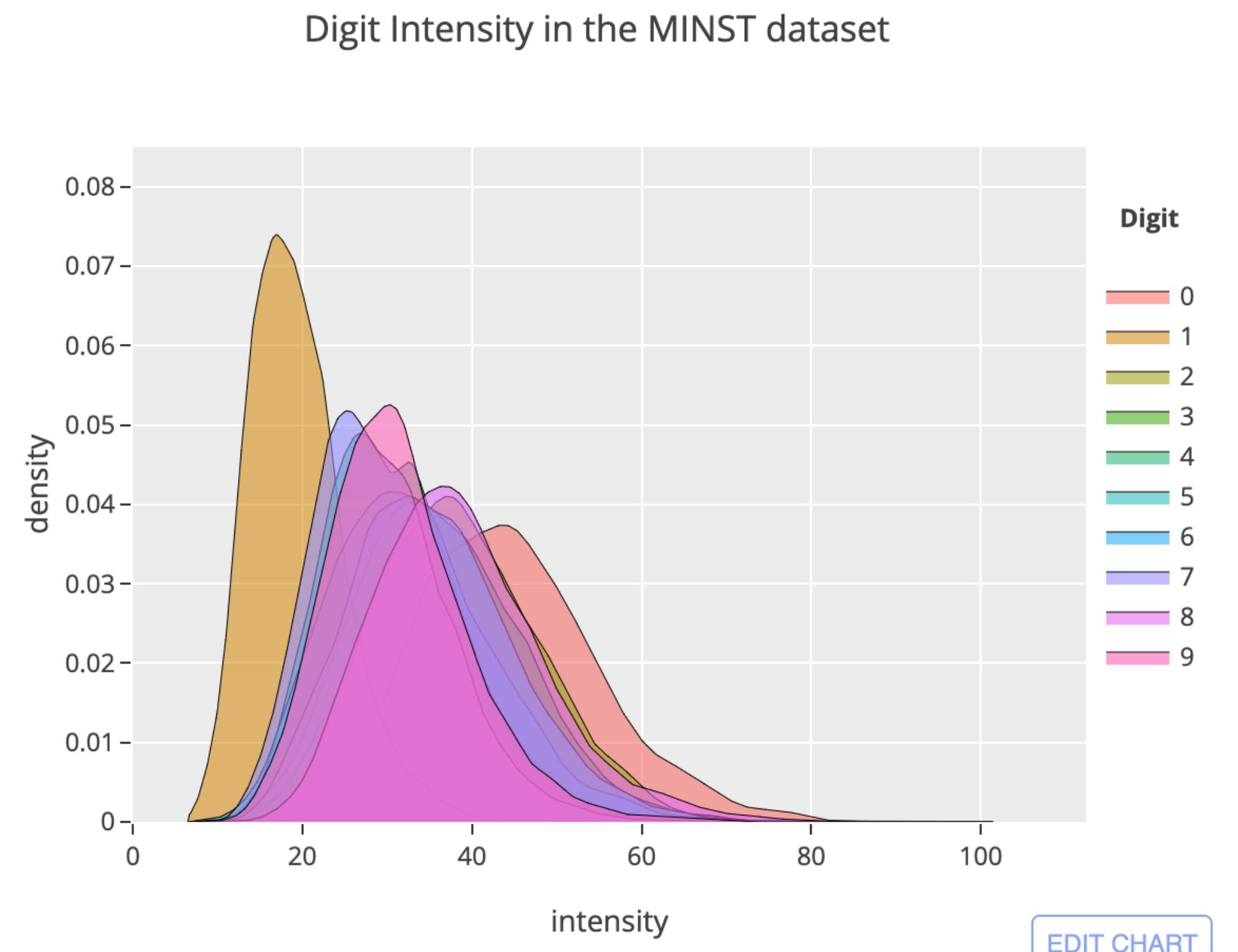
MNISTデータの背後にある 確率分布

1～9の数字の各クラスをそれぞれ確率密度関数⁽¹⁾で表した図を見てみよう⁽²⁾。

ただし、この図は可視化のために784次元ではなく、各ピクセルの平均値をとることで1次元の関数としている。

横軸は、各画像ごとのピクセル値の平均（ピクセルの平均強度）で、縦軸は確率密度（起こりやすさ）を示している。

横軸のある区間のピクセル平均強度をもつ手書き数字がサンプリングされる確率は、その区間で確率密度関数を積分した面積によって示される。



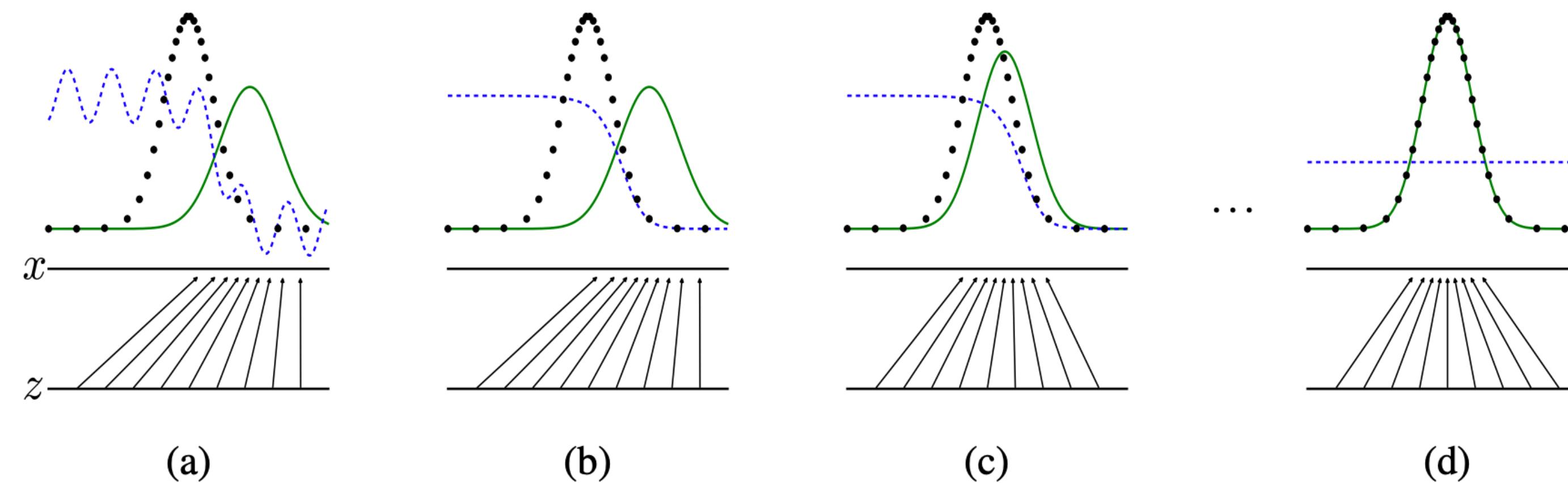
(1) <https://ja.wikipedia.org/wiki/>

%E7%A2%BA%E7%8E%87%E5%AF%86%E5%BA%A6%E9%96%A2%E6%95%B0

(2) <http://apapiu.github.io/2016-01-02-minst/>

分布がどうなると嬉しいの？

データの背後にある確率分布（緑と黒）が似かよるようにGを育てたい

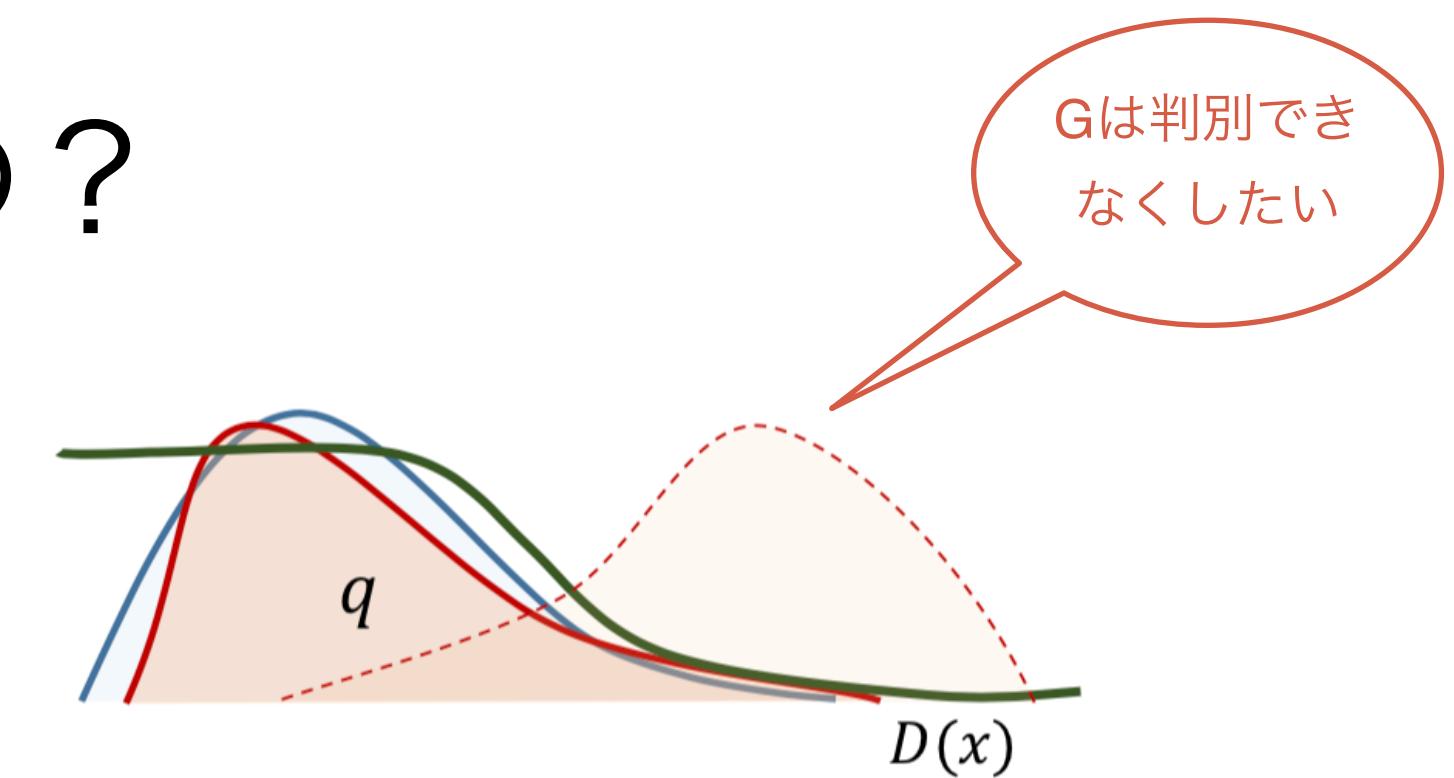
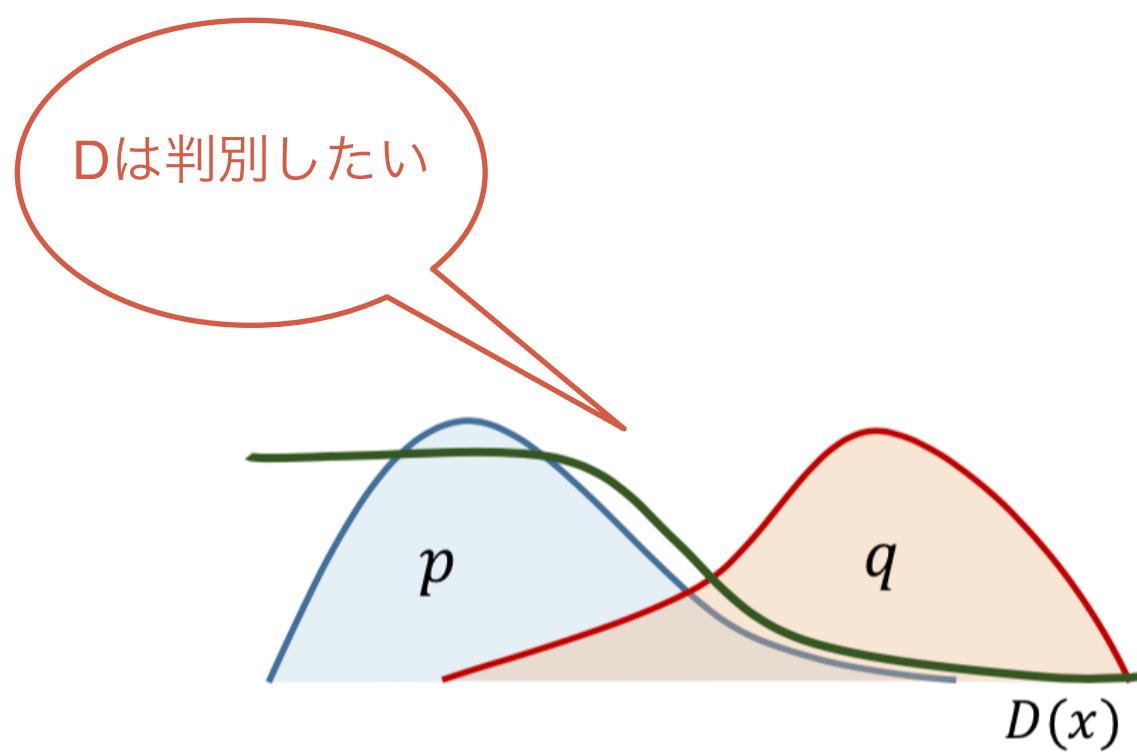


緑の線は $P_g(G(z))$: サンプリングされた $G(z)$ の背後にある確率分布。

黒い線は $P_{data}(x)$: サンプリングされた x (本物の画像) の背後にある確率分布。

青い線は D : 識別関数。Sigmoidの出力 $[0,1]$ の間で本物のデータである確率を返す。

どんな問題を解けばいいの？



$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))].$$

期待値 = 情報量の平均

Dは情報量を0にしたい

確率分布 $p_{\text{data}}(x)$ からサンプリングした $x=1$ にしたい

Dは情報量を0にしたい / Gは情報量を $-\infty$ にしたい

Dは1にしたい / Gは0にしたい

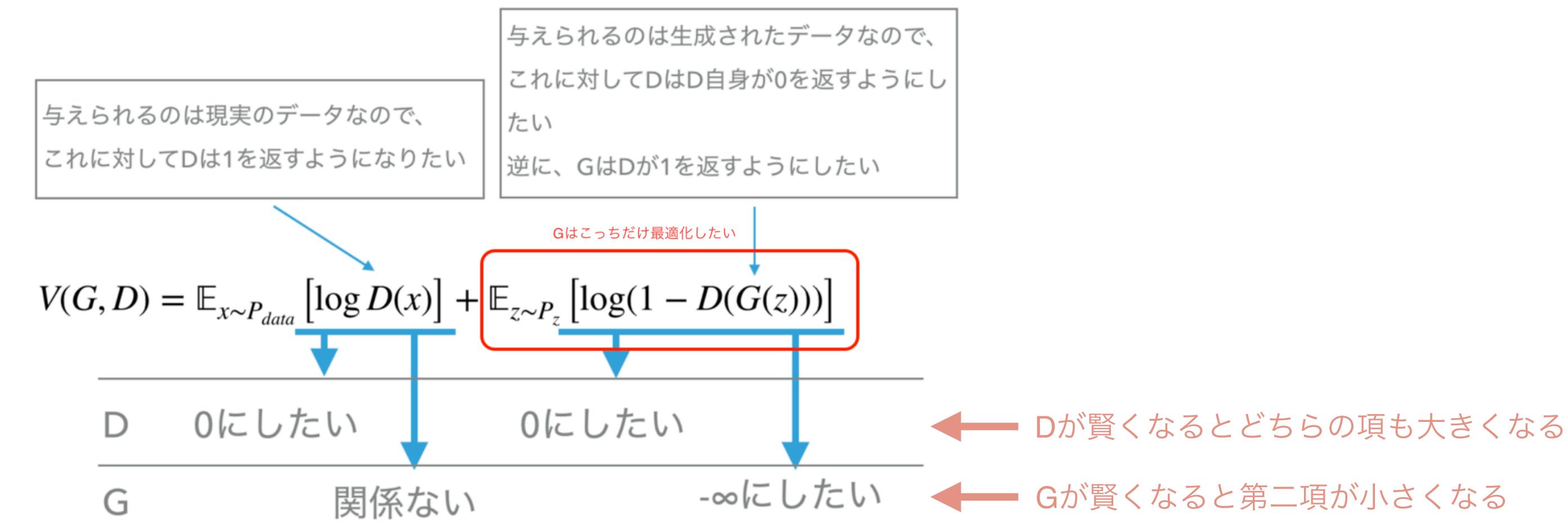
生成モデルと識別モデルを敵対的に訓練させることで、互いに競い合って精度を向上させていくミニマックス問題を解くように設定されている。

評価関数 $V(D, G)$ は右辺の期待値の和である。 D の値域は $[0, 1]$ なので \log はどちらも $[-\infty, 0]$ 間の値を返す。

最適な識別モデルは、生成モデルを固定した状態で、識別モデルに関する関数を **最大化** する。

最適な生成モデルは、識別モデルを D^* に固定した状態で、生成モデルに関する関数を **最小化** する。

ミニマックス問題を解く



Dとしては本物のデータ(x)に対しては1を返し、偽物のデータ($G(z)$)に対しては0を返したい。

Dが賢くなると第1項の**logD(x)**が大きくなる。第2項も、偽物を見抜くので**大きくなる**。

Gとしては偽物のデータ($G(z)$)に対して**Dから1を返して欲しい**。

Gがうまく生成できるようになると、**log(1-D(G(z)))**は小さくなる。

モデルの訓練フェーズ

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```

for number of training iterations do
    for  $k$  steps do ミニバッチ分(m個)のノイズベクトル
        • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
        • Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{\text{data}}(x)$ .
        • Update the discriminator by ascending its stochastic gradient:
```

識別モデルは
右の勾配でパ
ラメータを大
きくしていく

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

end for

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

生成モデルは
右の勾配でパ
ラメータを小
さくしていく

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

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

モデルの訓練フェーズ

識別モデル(D)と生成モデル(G)を**交互に**学習

以下のフローをkステップ分繰り返して、**識別モデルを訓練します。**

- 1.生成モデルの事前分布 $Pz(z)$ を使ってサンプリングをm回実行し、潜在変数 $\{z(1), \dots, z(m)\}$ を得る。
- 2.訓練途中の生成モデルと潜在変数 $\{z(1), \dots, z(m)\}$ から、生成データ $\{G(z(1)), \dots, G(z(m))\}$ を得る
- 3.訓練データセットから、m個の観測データ $\{x(1), \dots, x(m)\}$ を抽出する。
上方の勾配でパラメータ θ_d を上昇させ(ascending)、更新する。

以下のフローで**生成モデルを訓練します。**

- 1.生成モデルの事前分布 $Pz(z)$ を使ってサンプリングをm回実行し、潜在変数 $\{z(1), \dots, z(m)\}$ を得る。
- 2.訓練途中の生成モデルと潜在変数 $\{z(1), \dots, z(m)\}$ から、生成データ $\{G(z(1)), \dots, G(z(m))\}$ を得る

下方の勾配でパラメータ θ_d を下降させ(descending)、更新する。