Generative Adversarial Networks and their theoretical Analysis

理工学部 数理科学科 小泉 孝弥

1 Introduction

皆さんは, GAN というものを知っているだろうか. GAN とは敵対的生成ネットワーク (Generative Adversarial Networks) のことであり, 2014 年に登場した人工知能の技術である. 「敵対的」という単語がついている通り, 2 つのニューラルネットワークという関数を競わせることで, 学習させていく手法である. なお, 前提知識は数学科学部 3 回生程度の知識である.

2 Fundermetal concepts of Machine Learning

この節では機械学習の数学的な定式化を行う. d, m を自然数とする.

Definition 2.1 (仮説空間, 仮説)

$\mathcal{X} \subset \mathbb{R}^d$ から $\mathcal{Y} \subset \mathbb{R}^m$ へのなんらかの条件を満たす写像の集まりのことを仮説空間といい $\mathcal{H}$ と表記する. すなわち,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f \text{ が満たす条件 }\}$$

である. *(条件の具体例は後に述べる)*. 仮設空間 $\mathcal{H}$ の元のことを仮説 *(またはモデル)* と呼ぶ. また, この時の $\mathcal{X}$ を入力空間, $\mathcal{Y}$ を出力空間と呼ぶ.

Definition 2.2 (データ)

$(\Omega, \mathcal{F}, \mathbb{P})$ を確率空間, ρ を $\mathcal{X} \times \mathcal{Y}$ 上の確率分布とする. $\{(X_n, Y_n)\}_{n=1}^N$ を ρ に従う独立な確率変数列とした時, $\{(X_n, Y_n)\}_{n=1}^N$ の観測値 $\{(X_n(\omega), Y_n(\omega))\}_{n=1}^N$ のことを教師ありデータと呼び, $\{(x_n, y_n)\}_{n=1}^N$ と表記する. 同様に, ρ を $\mathcal{X}$ 上の確率分布とし, $\{X_n\}_{n=1}^N$ を ρ に従う独立な確率変数列とした時, $\{X_n\}_{n=1}^N$ の観測値 $\{X_n(\omega)\}_{n=1}^N$ のことを教師なしデータと呼び $\{x_n\}_{n=1}^N$ と表記する. また, 両者をまとめてデータと呼び D で表す.

Definition 2.3 (汎化誤差, 損失関数)

$\mathcal{H}$ を仮説空間, $(\Omega, \mathcal{F}, \mathbb{P})$ を確率空間, ρ をデータの確率分布とする. この時, 汎化誤差 $\ell : \mathcal{H} \to \mathbb{R}$

を,

$$l(f_w) = \mathbb{E}_{(X,Y)\sim\rho}[\ell(f_w(X), Y)]$$

と定義する. ここで, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ は損失関数と呼ばれる凸関数である.

Example 2.4 (損失関数)

ここで, よく使われる損失関数の例をいくつか述べておく.

1. *2* 乗損失関数 $\ell(y_1, y_2) = (y_1 - y_2)^2$
2. 交差エントロピー誤差 $\ell(y_1, y_2) = -y_2 \log y_1$

汎化誤差が最小となるような $f \in \mathcal{H}$ を求めることが機械学習の目標である. しかし, 一般にデータの分布は未知であるためこの式を解くことができない. そこで、損失関数というものを定義し, それを最適化することを考える.

Definition 2.5 (経験損失関数)

$\mathcal{H}$ を仮説空間とする. $\mathcal{H}$ から $\mathbb{R}$ への写像 $\mathcal{L}_D : \mathcal{H} \to \mathbb{R}$ を経験損失関数 *(Loss function)* と呼ぶ.

Definition 2.6 (機械学習空間 *(ML* 空間*))*

D をデータ, $\mathcal{H}$ を仮説空間, $\mathcal{L}_D$ を経験損失関数とする. この時 *5* つ組 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を機械学習空間 *(Machine Learning space)* あるいは, *ML* 空間という.

Definition 2.7 (学習, 最適仮説)

$\mathcal{H}$ を仮設空間, $\mathcal{L}_D : \mathcal{H} \to \mathbb{R}$ を経験損失関数とする. 経験損失関数が最大または, 最小となるような $f^* \in \mathcal{H}$ を求めること[*1]を学習といい, $f^* \in \mathcal{H}$ を最適仮説と呼ぶ.

Definition 2.8 (機械学習)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上での学習を機械学習という. 特に, D が教師ありデータの時, 教師あり機械学習と呼び, 教師なしデータの時, 教師なし機械学習と呼ぶ.

Remark 2.9 (訓練データとテストデータ)

機械学習を行う際は, データ D を全て使って学習させるのではなく, データを訓練データ D_{Train} とテストデータ D_{Test} に分ける. 訓練データはモデルの学習に用いて, テストデータはモデルの精度の確認のために用いる.

3 Some Examples

前節では, 機械学習の抽象的な枠組みを紹介したが, この説では機械学習空間の具体例を述べる. $D = \{(x_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \mathcal{Y}$ を教師ありデータとする.

3.1 単回帰分析・重回帰分析

最初に機械学習の最も基本的なモデルである重回帰分析を紹介する.

Example 3.1 (重回帰分析)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$\mathcal{X} = \mathbb{R}^d (N \geq d)$, $\mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = W^\top x + b, W \in \mathbb{R}^d, b \in \mathbb{R}\},$$
$$\mathcal{L}_D(f) = \sum_{i=1}^{N} |f(x_i) - y_i|^2.$$

この機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を重回帰分析という. $X \in \mathbb{R}^{N \times (d+1)}$ を

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nd} \end{pmatrix}$$

と定義し, $\mathbf{y} = (y_1, y_2, \cdots, y_N)^\top \in \mathbb{R}^N$ とする. この時, $\mathbf{rank}(X) = d + 1$ ならば, 最適仮説 $f^* \in \mathcal{H}$ は $[b\ W] = (X^\top X)^{-1} X^\top \mathbf{y}$ とすれば

$$f^*(x) = [b\ W]^\top x$$

である. なお, $d = 1$ の時, 重回帰分析を単回帰分析という.

3.2 線形回帰

前小節で定義した重回帰分析は線形的なデータにしか適応できなかった. それを基底関数と呼ばれる関数列を定義する事で, 非線形データにも対応することができる.

Definition 3.2 (基底関数)

$\mathcal{X}$ を入力空間とし, 任意に $d \in \mathbb{N}$ をとる. $\mathcal{X}$ 上の連続関数列 $\{\phi_n\}_{n=1}^d \subset C(\mathcal{X}, \mathcal{X})$ が一次独立であるとき, $\Phi = (\phi_1, \phi_2, \cdots, \phi_d)^\top : \mathcal{X} \to \mathcal{X}^d$ を基底関数 *(basis function)* と呼ぶ.

Example 3.3 (多項式回帰)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.

$\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = W^\top \Phi(x), W \in \mathbb{R}^{d+1}\},$$
$$\mathcal{L}_D(f) = \sum_{i=1}^{N} |f(x_i) - y_i|^2.$$

ここで, $\phi_n(x) = x^n$, $n \in \{0, 1, \cdots, d\}$ とする.
この機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を多項式回帰という. $X = [\Phi(x_1), \Phi(x_2), \cdots, \Phi(x_N)]^T \in \mathbb{R}^{N \times d}$ とし, $\mathbf{y} = (y_1, y_2, \cdots, y_N)^T \in \mathbb{R}^N$ とする. この時, 最適仮説 f^* は重回帰分析と同様に $W = (X^\top X)^{-1} X^\top \mathbf{y}$ とすれば

$$f^*(x) = W^\top x$$

である.

Remark 3.4 (ハイパーパラメータと学習パラメータ)

多項式回帰の多項式の次数 $d \in \mathbb{N}$ や基底関数 $\{\phi_n\}_{n=1}^d$ のようにコンピュータに学習させるのではなく, 機械学習モデルの設計者が設定するパラメータのことをハイパーパラメータと呼ぶ. 一方, $W \in \mathbb{R}^{d+1}$ のように, データからコンピュータが自動で学習するパラメータのことを学習パラメータと呼び, Θ と表す.

3.3 過学習と正則化

前節の多項式回帰で学習を行うと以下のグラフのように, 訓練データに過剰適合してしまい, 未知のデータに関しての汎化性能が小さくなってしまうことを過学習 (overfitting) という.

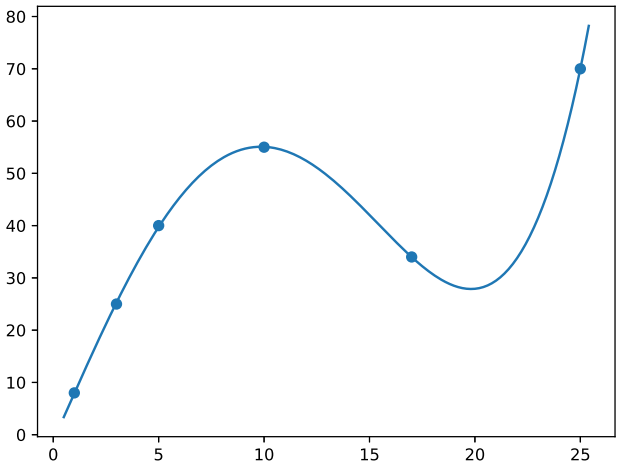

図 1　$d = 4$ の時の多項式回帰の過学習

過学習を防ぐための手法が正則化である.

Definition 3.5 (正則化)

$\mathcal{L}_D : \mathcal{H} \to \mathbb{R}$ を損失関数とする. $\mathcal{L}_D^R := \mathcal{L}_D + L^R$ とする. この時, $\mathcal{L}_D^R$ を $\mathcal{L}_D$ の正則化と呼び, $L^R : \Theta \to \mathbb{R}$ を正則化項と呼ぶ.

Example 3.6 (***Ridge*** 正則化多項式回帰)

$\lambda \in \mathbb{R}^+ := \{x \in \mathbb{R} \mid x \geq 0\}$, $d \in \mathbb{N}$ を任意にとる. 機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.

$\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = W^\top \Phi(x), W \in \mathbb{R}^{d+1}\},$$

$$\mathcal{L}_D(f) = \sum_{i=1}^{N} |f(x_i) - y_i|^2 + \lambda W^\top W. \qquad (W \text{ は } f \text{ のパラメータ})$$

ここで, $\phi_n(x) = x^n$, $n \in \{0, 1, \cdots, d\}$ とする ($\lambda W^\top W$ が正則化項である).

この機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を *Ridge* 正則化多項式回帰という. 通常の多項式回帰の時と同様に, $X = [\phi_1(x_1), \phi_2(x_2), \cdots, \phi_N(x_N)]^\top \in \mathbb{R}^{N \times d}$ とし, $\mathbf{y} = (y_1, y_2, \cdots, y_N)^\top \in \mathbb{R}^N$ とする. この時, 最適仮説 f^* は $W = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$ とすれば

$$f^*(x) = W^\top x$$

である.

正則化の結果が以下のグラフである.

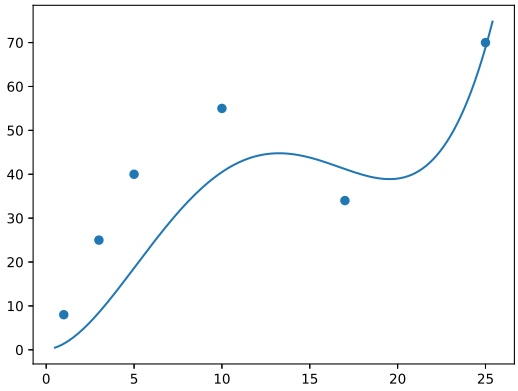

図 2　$\lambda = 700$, $d = 4$ の時の Ridge 正則化多項式回帰

このように正則化をすることで, パラメータが大きくなりすぎることを防ぐことで過学習を防ぐことができる.

Remark 3.7 (未学習について)

正則化を行う際にパラメータ[*2]$\lambda \in \mathbb{R}^+$ は自分で決める必要がある. その際に λ の値を大きくしすぎると未学習 *(underfitting)* という問題が発生する. 未学習とは訓練データに対してモデルが十分に適合できていないことをいう.

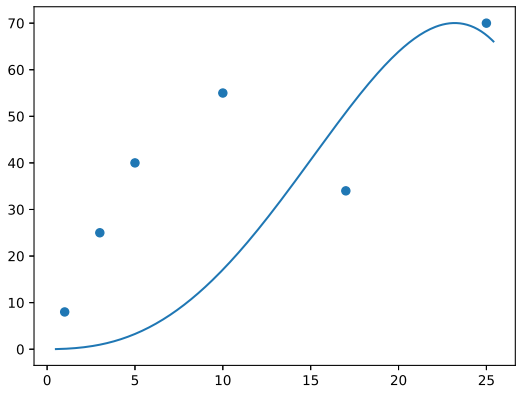

図 3　$\lambda = 30000$, $d = 4$ の時の Ridge 正則化多項式回帰の未学習

このように, 正則化パラメータは適切に選ぶことが大切である.

3.4 ロジスティック回帰と勾配降下法

最後の具体例として, ロジスティック回帰を紹介する. ロジスティック回帰は分類問題を解くための手法の 1 つである. まず, 実数値ベクトルを確率に変換するソフトマックス関数および, one-hot ベクトルというものを導入する.

Definition 3.8 (ソフトマックス関数)

$\psi : \mathbb{R}^d \to \mathbb{R}^d$ を以下で定義する.

$$\psi(x)_i = \frac{\exp(x_i)}{\sum_{k=1}^{d} \exp(x_k)}$$

この時, ψ をソフトマックス関数と呼ぶ.

Definition 3.9 (*one-hot* ベクトル)

$y \in \mathbb{R}^m$ が第 $c \in \{1, 2, \cdots, m\}$ 成分が *1* であり, 残りの成分が *0* である時, y を *one-hot* ベクトルという.

Example 3.10 (m 値分類ロジスティック回帰)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^m$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = \psi(Wx + b), W \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m\},$$

$$\mathcal{L}_D(f) = -\sum_{n=1}^{N} \sum_{k=1}^{m} y_{nk} \log f(x_n)_k,$$

ここで $\psi : \mathbb{R}^m \to \mathbb{R}^m$ はソフトマックス関数であり, y_n は *one-hot* ベクトルである.
この機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を m 値分類ロジスティック回帰という.

先ほどまでは, 損失関数の最小値を解析的に求めていたが, 今回の損失関数の最小値を解析的に求めることはせず, 勾配降下法という連続最適化アルゴリズムを用いて求めることにする. 勾配降下法とは C^1 関数 F の最小値を求めるためのアルゴリズムである.

Algorithm 1 Gradient Decent

Require: F: C^1 function on $\mathbb{R}^d$

Require: $0 < \alpha < 1$: learning rate

Require: θ: Initial parameter vector

$\theta \leftarrow \theta_0$

while θ not converged **do**

$\theta \leftarrow \theta - \alpha \nabla F(\theta)$

end while

return θ

Example 3.11 (*2* 次関数における勾配降下法)

$y = x^2$ において勾配降下法を適応した結果を以下の図と表に示す．本来の最適値である $x = 0$ に近づいているのが分かる．

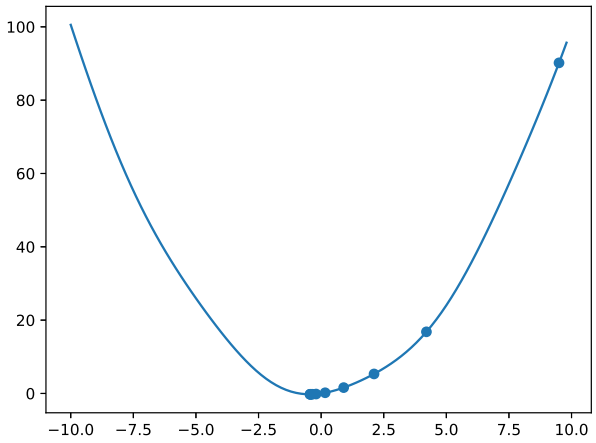

図 4　$\alpha = 0.001$ とした時の勾配降下法の結果.

イテレーション数	0	400	800	1200	1600	2000	2400	2800	3200	3600
θ の値	9.50	4.27	1.91	0.86	0.39	0.17	0.08	0.03	0.02	0.01

表 1　値の詳細

3.5　MNIST での実験

MNIST とは下記のような 0 から 9 までの自然数が書かれた 28×28 ピクセル画像のデータセットの名前である．今回は, m 値分類のロジスティック回帰を用いて、画像に書かれている文字を予

測する問題を考える．ロジスティック回帰の損失関数 $\mathcal{L}_D$ を勾配降下法を用いて最小化するため，まずは損失関数の勾配を計算する．ここで，$\mathcal{H} \simeq \mathbb{R}^{m\times d+1}$ だから，$\mathcal{L}_D$ を $\mathbb{R}^{m\times d+1}$ 上の関数と考える．

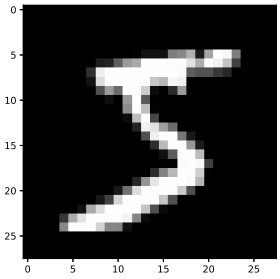

図 5　数字の 5

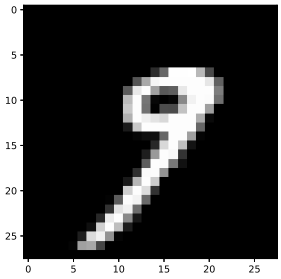

図 6　数字の 9

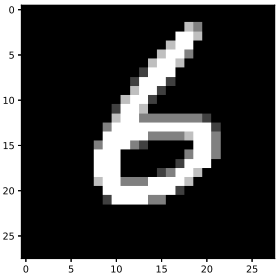

図 7　数字の 6

Lemma 3.12 ([*1*])

任意の $i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,d\}$ に対して以下が成立する．

$$\frac{\partial \mathcal{L}_D}{\partial w_{ij}} = -\sum_{n=1}^{N} (y_{ni} - f(x_n)_i)\, x_{nj}$$
$$\frac{\partial \mathcal{L}_D}{\partial b_i} = -\sum_{n=1}^{N} (y_{ni} - f(x_n)_i)$$

この補題 3.12 より，

$$\begin{aligned}
\frac{\partial \mathcal{L}_D}{\partial W} &= -\begin{pmatrix} \sum_{n=1}^{N} (y_{n1} - f(x_n)_1)\, x_{n1} & \cdots & \sum_{n=1}^{N} (y_{n1} - f(x_n)_1)\, x_{nd} \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^{N} (y_{nm} - f(x_n)_m)\, x_{n1} & \cdots & \sum_{n=1}^{N} (y_{nm} - f(x_n)_m)\, x_{nd} \end{pmatrix} \\
&= -\begin{pmatrix} y_{11} - f(x_1)_1 & \cdots & y_{N1} - f(x_N)_m \\ \vdots & \ddots & \vdots \\ y_{1m} - f(x_1)_m & \cdots & y_{Nm} - f(x_N)_m \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nd} \end{pmatrix} \\
&= -(Y - F)X^{\top},
\end{aligned}$$

ここで，$X = [x_1, x_2, \cdots, x_N]$, $Y = [y_1, y_2, \cdots, y_N]$, $F = [f(x_1), f(x_2), \cdots, f(x_N)]$ である．また，この計算より $\mathbf{1} = [1, 1, \cdots, 1]^{\top} \in \mathbb{R}^{N\times 1}$ とすれば，

$$\frac{\partial \mathcal{L}_D}{\partial b} = -(Y - F)\mathbf{1}$$

もいえる．これらを用いて学習した結果を以下に示す．また，この例からもわかるように機械学習においてパラメータの次元は非常に高次元なものとなる．(今回は $10 \times 784 + 10 = 7850$ 次元).

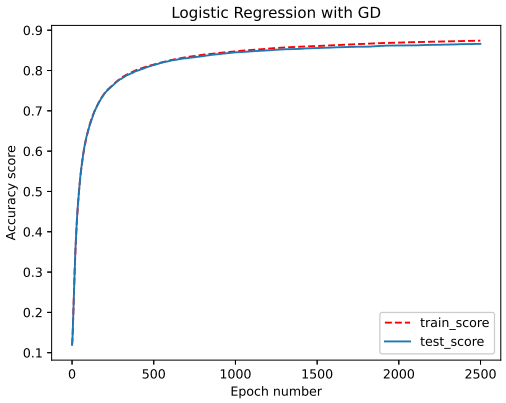

図 8　$\alpha = 0.0001$ とした時の, MNSIT 実験の結果

この例からもわかる通り, 機械学習において $\mathbb{R}^d$ 上の関数を学習するのではなく意味のある $\mathbb{R}^d$ の部分集合上の関数 (今回なら $[0,1]^d$) を学習していることがほとんどである.

4　Deep Learning

ここから、高い表現能力を持つとされるニューラルネットワークについて説明する. ニューラルネットワークとはアフィン変換と活性化関数と呼ばれる関数の合成関数である. 前節と同様に $D = \{(x_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \mathcal{Y}$ を教師ありデータとする.

4.1　Neural Network

Definition 4.1 (活性化関数)

$\sigma : \mathbb{R}^d \to \mathbb{R}^d$ を非線形かつ, *Lipschitz* 連続な関数とする. この時, σ を活性化関数 *(activation function)* という.

Example 4.2 (活性化関数の具体例)

活性化関数の具体例をいくつか述べる.

$$\sigma(x)_i = \frac{1}{1+\exp(-x_i)},$$
$$ReLU(x)_i = \max(x_i, 0).$$

σ をシグモイド関数 *(sigmoid function)*, $ReLU$ を正規化線形関数 *(Rectified Linear Unit)* という.

Definition 4.3 (回帰型ニューラルネットワーク)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する. $L = \{L_1, L_2, \cdots, L_K\}$ を自然数列と

する. $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathcal{Y} \;\middle|\; \begin{array}{l} f(x) = W^\top g_K \circ g_{K-1} \circ \cdots \circ g_1(x) + b, g_i = \eta(W_i x + b_i) \\ W_i \in \mathbb{R}^{L_i \times L_{i-1}}, b_i \in \mathbb{R}^{L_i}, W \in \mathbb{R}^{L_K}, b \in \mathbb{R} \end{array} \right\},$$

$$\mathcal{L}_D(f) = \frac{1}{N} \sum_{i=1}^{N} |f(x_i) - y_i|^2,$$

ここで η は活性化関数である. $f \in \mathcal{H}$ を回帰型 *(*全結合*)* ニューラルネットワーク *(Full-connected Neural Network for Regression)* と呼び, 各 g_i を f の第 i 層 *(layer)* と呼ぶ.

Definition 4.4 (分類型ニューラルネットワーク)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する. $L = \{L_1, L_2, \cdots, L_K\}$ を自然数列とする. $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^m$,

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathcal{Y} \;\middle|\; \begin{array}{l} f(x) = \psi(W g_K \circ g_{K-1} \circ \cdots \circ g_1(x) + b), g_i = \eta(W_i x + b_i) \\ W_i \in \mathbb{R}^{L_i \times L_{i-1}}, b_i \in \mathbb{R}^{L_i}, W \in \mathbb{R}^{m \times L_K}, b \in \mathbb{R}^m \end{array} \right\},$$

$$\mathcal{L}_D(f) = -\sum_{n=1}^{N} \sum_{k=1}^{m} y_{nk} \log f(x_n)_k,$$

ここで η は活性化関数, y_i は *one-hot* ベクトル, ψ はソフトマックス関数である. $f \in \mathcal{H}$ を分類型 *(*全結合*)* ニューラルネットワーク *(Full-connected Neural Network for Classification)* と呼び, 回帰型の時と同様に各 g_i を f の第 i 層と呼ぶ.

以下, ニューラルネットと書けば, 分類, 回帰ニューラルネットワークのいずれかを表すものとする.

Definition 4.5 (深層学習)

ニューラルネットワークが $K \geq 3$ の時に

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を深層学習 *(Deep Learning)* と呼ぶ.

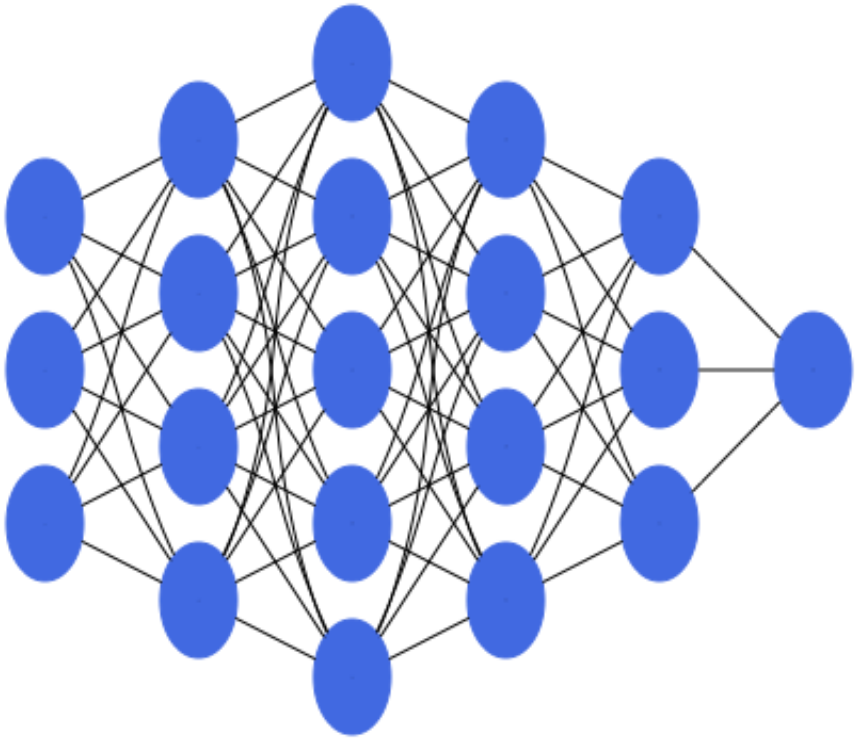

図 9　ニューラルネットワークの図

Remark 4.6 (フレームワーク)

現在はニューラルネットワークを簡単に実装するための深層学習フレームワークというものがいくつか存在する．代表なものとして，*Facebook* の *Pytorch* や，*Google* の *Tensorflow* などがある．

4.2　The Universal Theorem of Neural Network

冒頭でも述べたが，深層学習は前説まで紹介してきた機械学習の手法と比べて，精度がよくなることが知られている．しかしながら，その数学的な理由については現在も研究途中である．一方で，ニューラルネットの表現能力についてはいくつかの成果が出ており，その一つである普遍性定理を紹介する．証明については [3] や [9] を見て欲しい．

Definition 4.7 (符号付き測度)

$(X, \mathcal{F})$ を可測空間とする．$\mu : \mathcal{F} \to \mathbb{R}$ が，任意の互いに素な $\{A_n\}_{n\in\mathbb{N}} \subset \mathcal{F}$ に対して

$$\mu\left(\bigcup_{n\in\mathbb{N}} A_n\right) = \sum_{n\in\mathbb{N}} \mu(A_n)$$

を満たすとき，μ を符号付き測度という．

Definition 4.8 (正則 *Borel* 測度)

$(X, \mathcal{O})$ を位相空間とする. *Borel* 集合族 $\mathcal{B}(X)$ 上の *Borel* 測度 μ が

$$\begin{aligned}\forall A \in \mathcal{B}(X), \mu(A) &= \inf\{\mu(U) \mid U \supset A, U \in \mathcal{O}\} \\ &= \sup\{\mu(K) \mid K \subset A, K \textit{ is compact and closed}\}\end{aligned}$$

を満たす時, 正則 *Borel* 測度という.

Definition 4.9 (正則符号付き *Borel* 測度)

$(X, \mathcal{O})$ を位相空間とする. $(X, \mathcal{B}(X))$ 上の符号付き測度 μ が正則符号付き *Borel* 測度であるとは, 以下で定義される $\mathcal{B}(X)$ 上の有限測度 μ^+, μ^- が共に正則 *Borel* 測度になることである.

$$\begin{aligned}\mu^+(A) &= \sup\{\mu(B) \mid B \in \mathcal{B}(X), B \subset A\}, \\ \mu^-(A) &= -\inf\{\mu(B) \mid B \in \mathcal{B}(X), B \subset A\}.\end{aligned}$$

また, X 上の正則符号付き測度全体の集合を $M(X)$ と表記する.

Definition 4.10 (*Sigmoidal*)

関数 $\sigma : \mathbb{R} \to \mathbb{R}$ がシグモイド的関数であるとは

$$\lim_{t \to +\infty} \sigma(t) = 1 \quad \lim_{t \to -\infty} \sigma(t) = 0$$

を満たすことである.

Theorem 4.11 (普遍性定理)

$X \subset \mathbb{R}^n$ をコンパクト連結部分空間とし, $\mathcal{H}$ を $K = 1$ の時の回帰型ニューラルネットの仮説空間とする. この時, 活性化関数 η がシグモイド的関数ならば,

$$\forall f \in C(X), \forall \varepsilon > 0, \exists G \in \mathcal{H} \ s.t. \ \sup_{x \in X} |f(x) - G(x)| < \varepsilon$$

が成立する.

この定理からもわかる通り, ニューラルネットは 1 つの層でどんな連続関数でも近似できるほどの高い近似能力を持つ. しかしながら実用上は, 実際に構成するのは非常に困難であるため, 層を増やすことで表現能力を向上させられることが知られているため, 層の数を多くすることが多い.

4.3 確率的最適化

前節でも述べてきたが, 機械学習において 10000 を超えるパラメータの勾配を求めなければならないことも珍しくはない. そこで, 確率的最適化というものを用いて計算量を減らすことを考える.

確率的最適化手法には様々なものが存在するが, ここでは SGD や Adam[2] について紹介する.

Definition 4.12 (ミニバッチ)

データ D に対して, 分割部分集合列 $\{D_k\}_{k=1}^K$ のことを D のミニバッチと呼び, K をバッチサイ

ズ *(batch size)* とよぶ.

以下, $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を機械学習空間とする.

Algorithm 2 Stchastic Gradient Decent

Require: $0 < \alpha \leq 1$: learning rate
Require: θ_0: Initial parameter vector
  $\theta \leftarrow \theta_0$
  while θ not converged **do**
    decompose D to mini batch D_{batch} randomly.
    for D_{batch} in D **do**
      $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{D_{batch}}$
    end for
  end while
  return θ

Algorithm 3 Adaptive Moment Estimation

Require: $0 < \alpha \leq 1$: learning rate
Require: $\beta_1, \beta_2 \in [0, 1)$
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$
  $v_0 \leftarrow 0$
  $t \leftarrow 0$
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    decompose D to mini batch D_{batch} randomly.
    for D_{batch} in D **do**
      $g_t \leftarrow \nabla_\theta \mathcal{L}_{D_{batch}}$
      $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
      $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
      $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
      $\hat{v_t} \leftarrow v_t / (1 - \beta_2^t)$
      $\theta_t \leftarrow \theta_{t-1} \hat{m}_t / (\sqrt{\hat{v_t}} + \varepsilon)$
    end for
    $\theta \leftarrow \theta_t$
  end while
  return θ

4.4　ニューラルネットでの学習

この節の最後に, ニューラルネットを SGD で学習した結果を紹介する.

Example 4.13 (det の学習)

回帰型ニューラルネットの具体例として行列式 $\det : [-5,5]^{3\times 3} \to \mathbb{R}$ をを *100000* 個の学習データ $D = \{(x_n, \det(x_n))\}$ を用いて学習した. 以下にその結果を示す.

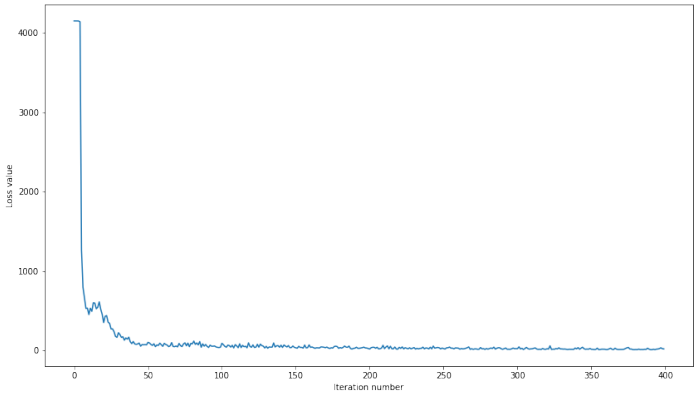

図 10　SGD を用いた det の学習結果

5　Generative Adversarial Networks

GAN[4] は 2014 年に Goodfellow らによって提案された手法である. その後, WGAN, DCGAN といった様々な亜種が登場し, 現在も様々な手法が提案されている. 冒頭でも述べたが, GAN は警察と泥棒の偽札造りに例えられる. G を泥棒, D を警察とした時, 以下の作業を繰り返すことで, 偽札の本物具合をあげていくといったものである.

1. 泥棒 G が偽札を作る.
2. 警察 D が偽札であると見破る.
3. 泥棒 G はなぜ, 警察 D に偽札であるとバレたのかを考え, より本物に近い偽札を作れるようになる.

これらは以下のように min-max ゲームとして定式化される.

5.1　GAN の定式化

この節では D は教師なしデータとする. また, データの確率分布を $\mathbb{P}_{data}$ とする.

Definition 5.1 (*Generative Adversarial Networks*)

機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する [8].
$\mathcal{X}$ は $\mathbb{R}^d$のコンパクト部分集合, $\mathcal{Y} = [0, 1]$,

$$\begin{aligned}\mathcal{H}_G &= \{G : \mathcal{Z} \to \mathcal{X} \mid G \text{ はニューラルネット }\},\\ \mathcal{H}_D &= \{D : \mathcal{X} \to \mathcal{Y} \mid D \text{ はニューラルネット }\},\\ \mathcal{L}_D(G, D) &= \mathbb{E}_{x\sim\mathbb{P}_{data}}[\log D(x)] + \mathbb{E}_{x'\sim\mathbb{P}_G}[\log(1 - D(x'))],\end{aligned}$$

ここで, $\mathcal{Z}$ は潜在空間と呼ばれる $\mathbb{R}^d$ の部分空間である. また, $G \in \mathcal{H}_1$ と確率分布 $\mathbb{P}_Z$(一様分布や正規分布) に従う確率変数 $Z : \Omega \to \mathcal{Z}$ に対し, $G(Z)$ が従う確率分布を $\mathbb{P}_G$ とする. $G \in \mathcal{H}_G$ を *Generator* と呼び, $D \in \mathcal{H}_D$ を *Discriminator* と呼ぶ. $\mathcal{H} = \mathcal{H}_G \times \mathcal{H}_D$ とした時, 機械学習空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を敵対的生成ネットワークという.

GAN の目的は, $\mathbb{P}_G$ を $\mathbb{P}_{data}$ に近づけることである. そのために, 以下の min-max ゲームを解くことを考える.

$$\arg\min_{G\in\mathcal{H}_G} \arg\max_{D\in\mathcal{H}_D} \mathcal{L}_D(G, D).$$

Remark 5.2 (*Generator* と *Discriminator* の役割について)

Generator $G \in \mathcal{H}_G$ の役割は, $z \in \mathcal{Z}$ からデータとよく似た偽物 $G(z) \in \mathcal{X}$ を作成することである. 一方, *Discriminator* $D \in \mathcal{H}_D$ の役割は, $x \in \mathcal{X}$ が本物のデータなのか *Generator* が作った偽物であるのかを判断することである. そのために, $X \sim \mathbb{P}_{\mathcal{X}}$ が本物である確率を学習する.

Definition 5.3 (最適 *Discriminator*)

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* とし, $G \in \mathcal{H}_G$ とする. $D^*_G \in \mathcal{H}_D$ が

$$\forall D \in \mathcal{H}_D, \mathcal{L}_D(D^*_G, G) \geq \mathcal{L}_D(D, G)$$

を満たす時, G に関しての最適 *Discriminator* であるという.

Proposition 5.4 ([4])

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* とし, $G \in \mathcal{H}_G$ とする. この時, 最適 *Discriminator* は以下で与えられる.

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)},$$

ここで, p_{data} は $\mathbb{P}_{data}$ の確率密度関数であり, p_G は $\mathbb{P}_G$ の確率密度関数である.

Proof: 期待値の計算から

$$\begin{aligned}\mathcal{L}_D(G, D) &= \int_{\mathcal{X}} p_{data}(x) \log(D(x))dx + \int_{\mathcal{X}} p_G(x) \log(1 - D(x))dx\\ &= \int_{\mathcal{X}} p_{data}(x) \log(D(x)) + p_G(x) \log(1 - D(x))dx\end{aligned}$$

を得る. ここで, 関数 $F(a,b)(y) = a\log(y) + b\log(1-y)$ について,

$$\underset{(a,b)\in\mathbb{R}^2-\{(0,0)\}}{\arg\max} F(a,b) = \frac{a}{a+b}$$

であることを用いれば主張が従う. ■

この命題 5.4 より $\mathcal{H}_G$ 上の関数 $C : \mathcal{H}_G \to \mathbb{R}$ が

$$C(G) = \mathbb{E}_{x\sim\mathbb{P}_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + \mathbb{E}_{x'\sim\mathbb{P}_G}\left[\log\frac{p_G(x')}{p_{data}(x')+p_G(x')}\right]$$

で定まる. 関数 C を仮想訓練基準 (virtual training criterion) と呼ぶ.

Theorem 5.5 (*GAN* の最小性 [*4*])

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN*, C を仮想訓練基準とする. C が最小値 $-\log 4$ を取るための必要十分条件は $\mathbb{P}_{data} = \mathbb{P}_G$ となることである.

Proof: $\mathbb{P}_{data} = \mathbb{P}_G$ とする. この時,

$$\begin{aligned} C(G) &= \mathbb{E}_{x\sim\mathbb{P}_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + \mathbb{E}_{x'\sim\mathbb{P}_G}\left[\log\frac{p_G(x')}{p_{data}(x')+p_G(x')}\right] \\ &= \mathbb{E}_{x\sim\mathbb{P}_{data}}\left[\log\frac{1}{2}\right] + \mathbb{E}_{x'\sim\mathbb{P}_G}\left[\log\frac{1}{2}\right] \\ &= -\log 2 - \log 2 = -\log 4 \end{aligned}$$

であるから, C は常に最小値 $-\log 4$ をとる. 逆に C の最小値が $-\log 4$ であるとする. *JSD* を *Jensen-Shannon Divergence* とし, C を変形すると

$$C(G) = -\log 4 + JSD\left(\mathbb{P}_{data}\|\mathbb{P}_G\right)$$

となる. ここで $JSD\left(\mathbb{P}_{data}\|\mathbb{P}_G\right) = 0$ と $\mathbb{P}_{data} = \mathbb{P}_G$ は同値だから $\mathbb{P}_{data} = \mathbb{P}_G$ である. ■

これらからわかる通り, GAN とはデータの確率密度関数をニューラルネットを用いて近似する機構である.

5.2 Instability of training GAN

前節では GAN の定式化及び, いくつかの命題を証明した. ここでは GAN の学習の不安定性についてのべる. 詳しくは [6] を参照してほしい.

Theorem 5.6 (*The Perfect Discriminator Theorem*)

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* とし, p_{data} を $\mathbb{P}_{data}$ の確率密度関数, p_G は $\mathbb{P}_G$ の確率密度関数とする. この時,

$$\exists M, P \subset \mathcal{X} : compact\ s.t.\ M \cap P = \emptyset, supp(p_{data}) \subset M \text{ かつ } supp(p_G) \subset P$$

が成立するならば, 以下の性質を満たす *smooth* な最適 *Discriminator* D_G^* が存在する.

1. $\mathbb{P}_{data}(D=1)=1$ かつ $\mathbb{P}_G(D=0)=1$
2. $\forall x \in M \cup P, \nabla_x D_G^*(x)=0.$

Proof: $P \cap M = \emptyset$ だから, $\delta = d(P,M)$ とすれば $\delta > 0$ である. ここで,

$$\hat{M} = \{x \in \mathcal{X} \mid d(x,M) \leq \delta/3\},$$
$$\hat{P} = \{x \in \mathcal{X} \mid d(x,P) \leq \delta/3\}$$

と定義する. M, P がコンパクトであることと, δ の定義より $\hat{M}$, $\hat{P}$ は共にコンパクトであり, $\hat{M} \cap \hat{P} = \emptyset$ である. したがって, *Urysohn's smooth lemma* より

$$\exists D_G^* \in \mathcal{H}_D : \textit{smooth s.t. } D_G^*|_{\hat{M}} = 1 \text{ かつ } D_G^*|_{\hat{P}} = 0$$

が成立する. 任意の $x \in supp(p_{data})$ に対して, $D_G^*(x)=1$ だから, $\log D_G^*(x) = 0$ である. また, 任意の $x \in supp(p_G)$ に対して, $D_G^*(x)=0$ だから, $\log(1-D_G^*(x))=0$ である. これより D_G^* が最適 *Discriminator* であること, 及び *1.* が従う. また D_G^* は $M \cup P$ 上で定値写像だから *2.* が成立する. ■

Definition 5.7 (*Discriminator* ノルム)

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* とする. $D \in \mathcal{H}_D \subset C^1(\mathcal{X})$ に対して,

$$\|D\| = \sup_{x \in \mathcal{X}} |D(x)| + \|\nabla_x D(x)\|_2$$

とする. この時, $\|\cdot\|$ の $\mathcal{H}_D$ に制限したノルム $\|\cdot\|_{\mathcal{H}_D}$ を *Discriminator* ノルムという.

Theorem 5.8 (*Vanishing gradient on the Generator*)

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* とする. $G_\theta \in \mathcal{H}_G$ とし, p_{data} を $\mathbb{P}_{data}$ の確率密度関数, p_G は $\mathbb{P}_G$ の確率密度関数とする. この時,

$$\exists M, P \subset \mathcal{X} : \textit{compact s.t. } M \cap P = \emptyset, supp(p_{data}) \subset M \text{ かつ } supp(p_G) \subset P$$

が成立し,

$$\exists M \in \mathbb{R}^+ \textit{ s.t. } \forall \mathbb{E}_{z \sim \mathbb{P}_Z}[\|J_\theta G_\theta(z)\|_2^2] \leq M^2$$

が成立するとする. この時, 任意の $\delta \in \mathbb{R}^+ - \{1\}$ について,

$$\forall D \in \mathcal{H}_D, \|D_G - D_G^*\|_{\mathcal{H}_D} < \delta \Longrightarrow \|\nabla_\theta \mathbb{E}_{z \sim \mathbb{P}_Z}[\log(1 - D(G_\theta(z)))]\|_2 < M\frac{\delta}{1-\delta}$$

が成立する.

Proof: $\|\cdot\|^2$ が凸であることと, *Jensen* の不等式より

$$\begin{aligned}
\|\nabla_\theta \mathbb{E}_{z\sim\mathbb{P}_Z}[\log(1-D(G_\theta(z)))]\|_2^2 &\leq \mathbb{E}_{z\sim\mathbb{P}_Z}\left[\frac{\|\nabla_\theta D(G_\theta(z))\|_2^2}{|1-D(G_\theta(z))|^2}\right] \\
&\leq \mathbb{E}_{z\sim\mathbb{P}_Z}\left[\frac{\|\nabla_x D(G_\theta(z))\|_2^2\|J_\theta G_\theta(z)\|_2^2}{|1-D(G_\theta(z))|^2}\right] \\
&\leq \mathbb{E}_{z\sim\mathbb{P}_Z}\left[\frac{(\|\nabla_x D^*(G_\theta(z))\|_2+\delta)^2\|J_\theta G_\theta(z)\|_2^2}{(|1-D^*(G_\theta(z))|-\delta)^2}\right] \\
&\leq \mathbb{E}_{z\sim\mathbb{P}_Z}\left[\frac{\delta^2\|J_\theta G_\theta(z)\|_2^2}{(1-\delta)^2}\right] \\
&\leq M^2\frac{\delta^2}{(1-\delta)^2}
\end{aligned}$$

となる. 両辺の平方をとれば題意が従う. ■

Corollary 5.9

$(\mathcal{X},\mathcal{Y},D,\mathcal{H},\mathcal{L}_D)$ を *GAN* とする. $G_\theta\in\mathcal{H}_G$ とし, p_{data} を $\mathbb{P}_{data}$ の確率密度関数, p_G は $\mathbb{P}_G$ の確率密度関数とする. 先の定義の仮定条件の元で,

$$\lim_{\|D-D^*\|_{\mathcal{H}_D}\to 0}\nabla_\theta\mathbb{E}_{z\sim\mathbb{P}_Z}[\log(1-D(G_\theta(z)))]=0$$

が成立する.

これらの定理・系から Discriminator が最適解に近づけば近づくほど, Generator の勾配が 0 に近づき学習が安定しないことがわかる.

5.3 Stabilization of training GAN

GAN の学習を安定化させるために, Gradient Penalty[7] や Spectral Normalization[8] などの手法が提案されている. 今回は, パラメータ行列の特異値を利用する Spectral Normalization を紹介する. $(\mathcal{X},\mathcal{Y},D,\mathcal{H},\mathcal{L}_D)$ を GAN とする. f を D から最終層の活性化関数 $\mathcal{A}$ を省いたものとする (すなわち $D=\mathcal{A}\circ f$.)

Proposition 5.10 (*[8]*)

f の各層の活性化関数の *Lipschitz* ノルムが *1* であるとする. この時,

$$\|f\|_{Lip}\leq\prod_{k=1}^{K+1}\|W_k\|_{op}$$

が成立する. ここで $g_k=\eta(W_kx+b_k)$ とした時, $f(x)=W_{K+1}^\top g_K\circ\cdots\circ g_1(x)+b$ とする. ここで, $\|f\|_{Lip}$ は *Lipschitz* ノルム, $\|\cdot\|_{op}$ で行列の作用素ノルムを表す.

Proof: *Lipschitz* ノルムの性質と $\|g_k\|_{Lip} = \|W_k\|_{op}$ より,

$$\begin{aligned}\|f\|_{Lip} &\leq \prod_{k=1}^{K} \|g_k\|_{Lip} \\ &\leq \prod_{k=1}^{K} \|\eta_k\|_{Lip}\|W_k\|_{op} \\ &= \prod_{k=1}^{K} \|W_k\|_{op}.\end{aligned}$$

■

したがって, 各層のパラメータ W_K の各成分を $\|W_k\|_{op}$ で割れば, $\|f\|_{Lip} \leq 1$ とすることができる. この手法を, Spectral Normalization と呼ぶ.

Definition 5.11 (*SNGAN[8]*)

$(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を *GAN* の *ML* 空間とする. この時, *Spectral Normalization* を用いて

$$\arg\min_{G\in\mathcal{H}_G} \quad \arg\max_{D\in\mathcal{H}_D, \|f\|_{Lip}\leq 1} \quad \mathcal{L}_D(G, D).$$

を解く問題を *SNGAN* という. ここで, f は D から最終層の活性化関数 $\mathcal{A}$ を省いたものである *(*すなわち $D = \mathcal{A} \circ f$*.)*

5.4 GAN の応用

最後に言葉のみではあるが GAN の応用である Cycle-GAN[5] や GameGAN[11] について紹介する. Cycle-GAN は 2 つの GAN を用意することにより, 画像の変換をすることができるモデルである. 例えば, 夏の画像を冬の画像に変えたりすることができる. GameGAN はゲーム環境のようにリアルで複雑な環境の生成を目指した GAN であり, 論文中では Pac-Man の生成に成功している,

6 Conclusion

今回は, 近年の深層学習の大きな結果である敵対的生成ネットワークについてのべた. しかしながら, 本稿は 2017 年までの理論解析部分しか追えておらず, 2020, 2021 年の話題についておまり触れられなかったことが心残りである. しかし, この記事を書くにあたり, もう一度文献を読み直し, 機械学習への理解を深めることができたことはとてもよかった.

最後にはなるが, 学部 4 年間の間で支えていただいた, 友人, 先生方の皆様にはこの場で感謝を申し上げる. 引き続き大学院でも頑張っていきたい.

参考文献

[1] darden. 多クラス分類ロジスティック回帰.
http://darden.hatenablog.com/entry/2018/01/25/222201. Python と機械学習, 2018

[2] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 2015.

[3] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. Mathmatics of control, signal and systems. vol. 2, no. 4, 1989

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, 2014.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. International Conference on Computer Vision. 2017.

[6] Martin Arjovsky and Leon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. International Conference on Learning Representations, 2017.

[7] Martin Arjovsky, Soumith Chintala and Leon Bottou. Wasserstein GAN. International Conference on Learning Representations, 2017.

[8] Takeru Miyato, Toshiki Kataoka, Masanori Koyama and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. International Conference on Learning Representations, 2018.

[9] mochimochidog. ニューラルネットワークの普遍性定理.
https://qiita.com/mochimochidog/items/ca04bf3df7071041561a. Qiita, 2020.

[10] Preferred Networks. ディープラーニング入門 Chainer チュートリアル.
https://tutorials.chainer.org/ja/index.html, 2019.

[11] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba and Sanja Fidler. Learning to Simulate Dynamic Environments With GameGAN. Conference on Computer Vision and Pattern Recognition. 2020.