深層学習を用いた素数判定機

芝浦工業大学 数理科学研究会

平成 29 年 5 月 26 日

※何か不明な点や計算ミス等がありましたら加筆修正しますので指摘をお願いします

制作: BV15005 石川直幹

目 次

1　始めに

1.1　研究動機

以前から素数の分布に興味があり, 芝浦祭などのテーマとしてきた. しかし, 代数的構造や解析学の技術を用いる伝統的な研究法に難しさを感じた. そこで, 比較的新しく, 今ブームが来ている深層学習を用いることで, より効率的に素数判定ができるのではないかと考え, 研究するに至った.

1.2　機械学習について

学習は帰納的学習と演繹的学習の二つに分けられる.

演繹的学習とは, ある基礎的で抽象的な概念から具体的な知識を身につける物である. 例えば, 数学の定理を導出することなどが上げられる.

帰納的学習とは, 複数の具体的な結果を読み込むことで具体的な知識を身につけるものである. 例えば, スポーツの練習, Exsel のオートフィルなどが上げられる. 機械学習もその一つである.

定義 1.1 (機械学習)**.** 機械学習とは, データから反復的に学習し, そこに潜むパターンを見つけ出すことである. さらに, 学習した結果を新たなデータにあてはめることで, パターンにしたがって将来を予測することとする.

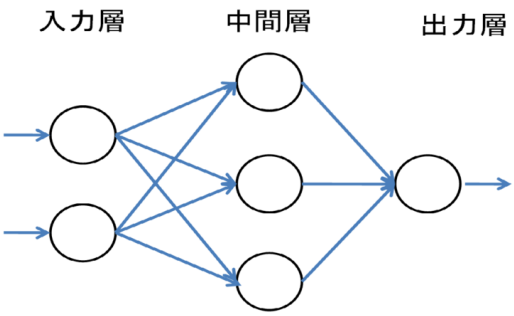
参考:機械学習とは—*SAS, www. sas. com/ ja_ jp/ insights/ analytics/ machine-learning. html*

(例) 強化学習, 遺伝的アルゴリズム, 深層学習

1.3　深層学習について

定義 1.2. 複数の信号を受け取り, 適当な計算をした上で信号を出力するものを**ニューロン**という. また, ニューロンを複数組み合わせたものをニューラルネットという.

ここでは, 深層学習を, 多層のニューラルネットを用いた機械学習のこととする. ニューラルネットとは, 人間の神経細胞を模した計算素子である人工ニューロンを組み合わせたものである. これは, 生物の神経細胞が他の複数の神経細胞から受け取り, 細胞内で処理を施したうえで, 出力信号を他の神経細胞に送ることで処理をする様子を参考にしている. 以下の図がその具体例である.



引用元:http://cdn-ak.f.st-hatena.com/images/fotolife/u/ura_ra/20111026/20111026235506.png

次に, 各ニューロンを数学的に定義する.

定義 1.3 (重み, バイアス, 活性化関数)**.** ニューロンが受け取る入力を $x_i \quad (i = 1, \ldots, n)$ とし, このニューロンが受け取る総出力を

$$u = w_1 x_1 + \cdots + w_n x_n$$

とする. このとき, w_i を**重み** *(***ウエイト***)*, b を**バイアス**という. さらに, このニューロンの出力を

$$y = h(u)$$

とする. この h を**活性化関数**という.

定義 1.4 (各ニューロンのモデル)**.** n 層から $n+1$ 層への伝播を考えるとする. このとき, n 層の i 番目のノードから $n+1$ 層の j 番目のノードへの重みを $w_{ji}^{(n+1)}$, バイアスを $b_j^{(n+1)}$ とし, これらの和をそれぞれ $u_j^{(n+1)}$ とする. また, $n+1$ 層の i 番目のノードからの出力を $z_i^{(n+1)}$, 活性化関数を $h^{(n+1)}$ とすると,

$$\begin{aligned} u_j^{(n+1)} &= \sum_i w_{ji}^{(n+1)} z_i^{(n)} + b_j^{(n+1)}, \\ z_j^{(n+1)} &= h^{(n+1)}(u_j^{(n+1)}). \end{aligned}$$

または, 行列を用いて,

$$\begin{aligned} \mathbf{u}^{(n+1)} &= \mathbf{w}^{(n+1)} \mathbf{z}^{(n)} + \mathbf{b}^{(n+1)}, \\ \mathbf{z}^{(n+1)} &= h^{(n+1)}(\mathbf{u}^{(n+1)}) \end{aligned}$$

とする. これをニューロンのモデルとする.

(例) 順伝播型ネットワークを考える. x_i から u_i へ伝播するときの重みを w_{ji}, u_j のバイアスを b_j とすると, u_j はそれぞれ,

$$\begin{aligned} u_1 &= w_{11} x_1 + w_{12} x_2 + w_{13} x_3 + w_{14} x_4 + b_1 \\ u_2 &= w_{21} x_1 + w_{22} x_2 + w_{23} x_3 + w_{24} x_4 + b_2 \\ u_3 &= w_{31} x_1 + w_{32} x_2 + w_{33} x_3 + w_{34} x_4 + b_3 \end{aligned}$$

となる. さらに, 2 層の出力は 2 層の活性化関数を h とすると,

$$z_j = h(u_j)$$

以下, ネットワークは順伝播型で多層ものを考える. 全部で l 個の層があるとし, 第 1 層を**入力層**, 第 l 層を**出力層**, それ以外の層を**中間層 (隠れ層)** とする.

ネットワークの最終的な出力 $\mathbf{y}$ は, 入力 $\mathbf{x}$, 重み $w^{(2)}, \ldots, w^{(L)}$, バイアス $b^{(2)}, \ldots, b^{(L)}$ で決まる. よって,

$$\mathbf{y} = N(\mathbf{x}, w^{(2)}, \ldots, w^{(L)}, b^{(2)}, \ldots, b^{(L)})$$

または, バイアスを入力 1 の重みと考えれば, $\mathbf{w} = (w^{(2)}, \ldots, w^{(L)}, b^{(2)}, \ldots, b^{(L)})$ とできるので,

$$\mathbf{y} = N(\mathbf{x}; \mathbf{w})$$

と表す. $N(\mathbf{x}, \mathbf{w})$ については, よりよい出力が得られるように $\mathbf{w}$ を選ぶことを考える. ここで, 1 つの入力 $\mathbf{x}$ に対しての望ましい出力 $\mathbf{d}$ のペアを訓練サンプルとする. さらに, 訓練サンプルが M 個与えられているとして,

$$D := \{(\mathbf{x}_1, \mathbf{d}_1), \ldots, (\mathbf{x}_M, \mathbf{d}_M)\}$$

と表す. これを訓練データという.

また, 活性化関数には, 単調増加する非線形関数が一般に用いられる. これは, 線形関数であるとすると任意の線形関数は, 線形関数で表されるので, 層を重ねる意味がなくなるためである. 主に中間層で使われるものとしては,

$$f(u) = \begin{cases} 0 & (-\infty < u \leq 0), \\ 1 & (0 < u < \infty) \quad (\text{ステップ関数}), \end{cases}$$

$$f(u) = \frac{1}{1+e^{-u}} \quad (\text{シグモイド関数}),$$

$$f(u) = \begin{cases} 0 & (-\infty < u \leq 0), \\ x & (0 < u < \infty) \quad (\text{Relu 関数}), \end{cases}$$

が上げられる.

2　誤差関数と出力関数

ネットワークが示す関数 $N(\mathbf{x}, \mathbf{w})$ と訓練データ $\mathbf{d}_m$ 近さの尺度を定める. これを誤差関数 $E(\mathbf{w})$ といい, 以下, 2 の関数を 0 に近づけていくこと考える.

(注) 認識精度を指標とすると, 微分がほとんどの場所で 0 となり, 後に導入する勾配降下法がうまくいかなくなってしまう.

ネットワークでは, 問題の種別に応じて, 誤差関数と出力層の設計が変わる. 下の表にその一覧を示す.

問題の種別	出力層の活性化関数	誤差関数
回帰	恒等関数	2 乗誤差 式 (1)
二値分類	シグモイド関数	式 (2)
他クラス分類	ソフトマックス関数	交差エントロピー 式 (3)

表 1: 問題の種別による出力関数と誤差関数の違い

2.1　回帰問題

回帰問題とは, ある入力データから連続的な数値の予測を行う問題のことである. 例えば, 人の画像から体重を予測するなどが上げられる. ここで, 目標となる関数の値域が $[-1, 1]$ の場合は, 出力層の活性化関数には双曲線正接関数が適しており, $(-\infty, \infty)$ の場合は恒等関数が適している.

回帰問題で用いる誤差関数は, 2 **乗誤差**

$$E(\mathbf{w}) = \sum_{m=1}^{M} (\mathbf{d}_m - N(\mathbf{x}, \mathbf{w}))^2 \tag{1}$$

とする.

2.2　分類問題

分類問題とは, 入力データがどのクラスに属するか予測する問題のことである. 例えば, 0 ～ 9 の手書き文字を判別するなどが上げられる. 分けるクラスが 2 つのとき, 二値分類. 2 個より多いとき, 多クラス分類という.

2.2.1 二値分類

二値分類を定式化する方法として, $d = 1$ となる事後確率 $p(d = 1|\mathbf{x})$ を考える. 与えれた $\mathbf{x}$ に対する d の推定は, このモデルを使って事後確率を計算し, その値が 0.5 を超えていれば $d = 1$, 下回れば, $d = 0$ と判断することとする.

この事後確率 $p(d = 1|\mathbf{x})$ をモデル化するためのネットワークは, 出力層のユニットを一つだけ持つとする. ネットワークのパラメータ $\mathbf{w}$ は, 訓練データ $\{(\mathbf{x}_m, \mathbf{d}_m)\}$ が、データが与える分布と最もよく整合するように定める. このネットワーク全体の入出力関係 $y(\mathbf{x}, \mathbf{w})$ を事後確率のモデル

$$p(d = 1|\mathbf{x}) \approx y(\mathbf{x}, \mathbf{w})$$

とする. これなら, ネットワークのパラメータ $\mathbf{w}$ を変えることで, さまざまな事後確率を表現することができる.

パラメータ $\mathbf{w}$ は, 訓練データ $\{(\mathbf{x}_n, \mathbf{d}_n)|n = 0, \ldots M\}$ を用いて, モデルが与える事後分布 $p(d|\mathbf{x}; \mathbf{w})$ を, 最尤推定を用いて定める. $d = \{0, 1\}$ なので,

$$p(d|\mathbf{x}) = p(d = 1|\mathbf{x})^d p(d = 0|\mathbf{x}\}^{1-d}$$

と表せる. また, $p(d|\mathbf{x}) = y(\mathbf{x}; \mathbf{w})$ としたので, $p(d = 0|\mathbf{x}) = 1 - y(\mathbf{x}; \mathbf{w})$ となる. 最尤推定は, このモデルの下で $\mathbf{w}$ に対する尤度を求め, それを最大化するような $\mathbf{w}$ を選ぶ. $\mathbf{w}$ の尤度は,

$$L(\mathbf{w}) \equiv \prod_{m=1}^{M} p(d_m|\mathbf{x}_m; \mathbf{w}) = \prod_{m=1}^{M} y(\mathbf{x}_m; \mathbf{w})^{d_m} \{1 - y(\mathbf{x}_m; \mathbf{w}_m)\}^{1-d_m}$$

で与えられる. 両辺に対数をとり, 符号を反転させたものを誤差関数とする. つまり,

$$E(\mathbf{w}) = -\sum_{m=1}^{M} [d_m \log y(\mathbf{x}_m; \mathbf{w}) + (1 - d_m) \log\{1 - y(\mathbf{x}_m; \mathbf{w})\}^{1-d_m}] \tag{2}$$

とする. ここで,

$$u \equiv \frac{p(\mathbf{x}, d = 1)}{p(\mathbf{x}, d = 0)}$$

とすると, 条件付き確率の定義より,

$$p(d = 1|\mathbf{x}) = \frac{p(\mathbf{x}, d = 1)}{p(\mathbf{x}, d = 0) + p(\mathbf{x}, d = 1)}$$

であるから, 事後確率 $p(d = 1|\mathbf{x})$ は出力層の活性化関数は, シグモイド関数とすればよいことがわかる.

以上の定式化の代わりに, 二値分類をこの後述べる他クラス分類の一種と見なし, そちらの方法論を採用する.

2.2.2 多クラス分類

多クラス分類を対象とする場合, ネットワークの出力関数 $l = L$ に分類したいクラス $C_1, \ldots, C_K$ それぞれのユニットをつくり, 並べる. このユニットの出力 y_k はクラス C_k に属する確率とする. つまり,

$$p(C_k|\mathbf{x}) = y_k = z_k^{(L)}$$

とし, 入力 $\mathbf{x}$ をこの確率が最大になるクラスに分類する.

多クラス分類でも, 二値分類同様に, ネットワークが実現する関数が各クラスの事後確率と見なし, そのようなモデルの下で, 訓練データに対するネットワークのパラメータの尤度を評価し, 最大化する. また,

$$\mathbf{d}_n = [d_{n1}, \ldots, d_{nK}]^T \quad (d_{nk} = \{0, 1\})$$

とし, 対応するクラスが真であるとき 1 で, それ以外のとき 0 をとるものとする. 例えば, 手書き文字の認識を行うことを考えると, 入力 $\mathbf{x}_m$ が"2"であり, 正解クラスを C_3 とすると, 目標の出力は,

$$\mathbf{d}_m = [0, 0, 1, 0, 0, 0, 0, 0, 0]^T$$

となる.

このように符号化を行うと, 事後分布は,

$$p(\mathbf{d}|\mathbf{x}) = \prod_{k=1}^{K} p(C_k|\mathbf{x})^{d_k}$$

と表せる. これより, 訓練データ $\{(\mathbf{x}_m, \mathbf{d}_m)\}$ に対する $\mathbf{w}$ の尤度を

$$L(\mathbf{w}) = \prod_{m=1}^{M} p(\mathbf{d}_m|\mathbf{x}_m; \mathbf{w}) = \prod_{m=1}^{M} \prod_{k=1}^{K} p(C_k|\mathbf{x}_m)\ d_{mk} = \prod_{m=1}^{M} \prod_{k=1}^{K} (y_k(\mathbf{x}; \mathbf{w}))^{d_{mk}}$$

と導ける. この尤度の対数をとり, 符号を反転させた値を誤差関数

$$E(\mathbf{w}) = -\sum_{m=1}^{M} \sum_{k=1}^{K} d_{mk} \log y_k(\mathbf{x}_m; \mathbf{w}) \tag{3}$$

とする. この関数を**交差エントロピー誤差**という.

出力層の活性化関数について考える. クラス C_k の事後確率は, 定義より,

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}, C_k)}{\sum_{j=1}^{K} p(\mathbf{x}, C_j)} = y_k$$

である. ここで, $u_k = \log p(\mathbf{x}, C_k)$ とおくと, この確率は,

$$p(C_k|\mathbf{x}) = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)}$$

となる. この関数を**ソフトマックス関数**といい, 出力層の活性化関数とすれば良いことがわかる. また, この関数の総和はいつでも 1 であり, 入力 $u_1^{(L)}, \ldots, u_K^{(L)}$ に定数 u_0 を足しても出力は変化しない.

3　学習の手法

順伝播型ネットワークの学習は, 与えられた学習データを元に計算される誤差関数を最小化することである. しかし, $E(\mathbf{w})$ は一般に凸関数ではなく大域的な最小解を直接求めることは困難である. そこで, 代わりに $E(\mathbf{w})$ の極小点を求めることを考える. $E(\mathbf{w})$ の極小点は一般に多数存在するので, 見つけた極小点は一般に $E(\mathbf{w})$ の大域的な最小解を与えるわけではない. しかし, その極小点 $\mathbf{w}$ で $E(\mathbf{w})$ の値がある程度小さければ, 目的の問題の良い解であることが期待できる. このような極小点を, 初期値から $\mathbf{w}$, を繰り返し更新することで求める. その一つの基本的な方法として, 勾配降下法がある. 以下では, 勾配降下法の変形である確率的勾配降下法を述べる.

3.1　確率的勾配降下法

勾配降下法とは, 誤差関数の勾配 (∇E) を求め, 負の方向に $\mathbf{w}$ を更新していく. つまり, 学習係数 ε を何らかの方法で定めて,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^t - \varepsilon \nabla E$$

とする方法のことである. この方法を勾配降下法といい, ランダムに選んだデータに対してこの方法を用いることを確率的勾配降下法という.

4　学習の結果

学習方法は, $\mathbf{x}$ を 1 ～ 12000 の数とし, $\mathbf{d}$ を素数なら 1 素数でないなら 0 とした. これは, 二値分類問題となる. ネットワークの中間層は 1 層のみとし, 活性化関数はシグモイド関数とした. 反復回数は 100000 回とし, 重みの初期値 0 ～ 1 の標準正規分布に従う乱数, バイアスの初期値は 0 とした. なお, プログラミングには Python3 を用いた.

4.1　学習その 1

まず, 1 ～ 10000 を学習データ, 10001 ～ 12000 をテストデータとした学習をおこなった. 以下は, 訓練データ, テストデータの認識精度と $W^{(1)}$ の値を 10000 ごとに表にしたものである. また, グラフは訓練データ, テストデータの認識精度を 10000 ごとにプロットし, 線でつないだものである.

表 2: 1 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8771	0.8955	−0.00091727	0.00999064	0.15907571
0.8771	0.8955	0.10057659	0.09887872	0.18051851
0.8771	0.8955	0.1080657	0.10665519	0.18466015
0.8771	0.8955	0.11097459	0.10966388	0.18643082
0.8771	0.8955	0.11363979	0.11241542	0.18790773
0.8771	0.8955	0.11599377	0.1148361	0.1894229
0.8771	0.8955	0.11744841	0.11633304	0.19034466
0.8771	0.8955	0.11867436	0.11759382	0.19114192
0.8771	0.8955	0.12029217	0.11925356	0.1922043
0.8771	0.8955	0.12175154	0.12074631	0.19331464

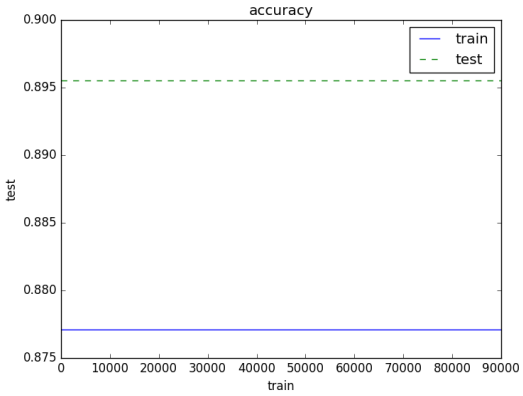
表 3: 2 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8771	0.8955	0.0115646	0.0005512	0.01035856
0.8771	0.8955	0.10456652	0.10290343	0.10070241
0.8771	0.8955	0.11414771	0.11228106	0.110682
0.8771	0.8955	0.11836094	0.1164302	0.11507788
0.8771	0.8955	0.12130393	0.11933224	0.11814115
0.8771	0.8955	0.12396341	0.12194714	0.12088747
0.8771	0.8955	0.12563434	0.12359076	0.12261078
0.8771	0.8955	0.12660055	0.12454497	0.12361156
0.8771	0.8955	0.12849383	0.12641595	0.12557158
0.8771	0.8955	0.12999612	0.12789311	0.12711388

5

表 4: 3 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8771	0.8955	−0.00970629	−0.00175097	−0.0040315
0.8771	0.8955	−0.02310494	−0.05001018	0.13409462
0.8771	0.8955	−0.02309829	−0.05014658	0.14298214
0.8771	0.8955	−0.02305733	−0.05021396	0.1498932
0.8771	0.8955	−0.02301883	−0.05025403	0.15253129
0.8771	0.8955	−0.02297426	−0.05028766	0.155922
0.8771	0.8955	−0.02292671	−0.0503191	0.1587535
0.8771	0.8955	−0.02288849	−0.05033958	0.16073094
0.8771	0.8955	−0.02285403	−0.05035519	0.16230821
0.8771	0.8955	−0.02281463	−0.05037377	0.16383588



学習その 1

以上の図とグラフから, 訓練データ, テストデータともに認識精度が一定である. さらに正解率から, すべての入力に対して”non-prime”と判定していることがわかる. また, $W^{(1)}$ については, ほとんどの値で増加している. この結果は, 学習が進んでいないことを意味する. よって, 有効な学習結果が得られなかった.

4.2 学習その 2

次に, 1 ~ 12000 のうち, ランダムの選んだ 10000 個を学習データ, 2000 個をテストデータとした (ただし, 素数の割合はそろえた.) 学習を行った. 以下は, 訓練データ, テストデータの認識精度と $W^{(1)}$ の値を 10000 ごとに表にしたものである. また, グラフは訓練データ, テストデータの認識精度を 10000 ごとにプロットし, 線でつないだものである.

表 5: 1 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8802	0.88	−0.01485245	0.00318149	0.00349706
0.8802	0.88	−0.01680389	0.1149435	0.11601032
0.8802	0.88	−0.01657757	0.12039401	0.12142266
0.8802	0.88	−0.0163962	0.1257257	0.12673309
0.8802	0.88	−0.01627137	0.12868517	0.12968375
0.8802	0.88	−0.01615666	0.13029208	0.13128669
0.8802	0.88	−0.01604868	0.13284149	0.13382918
0.8802	0.88	−0.01595806	0.13438035	0.13536133
0.8802	0.88	−0.01587439	0.1354013	0.13637766
0.8802	0.88	−0.01580518	0.13641956	0.13739149

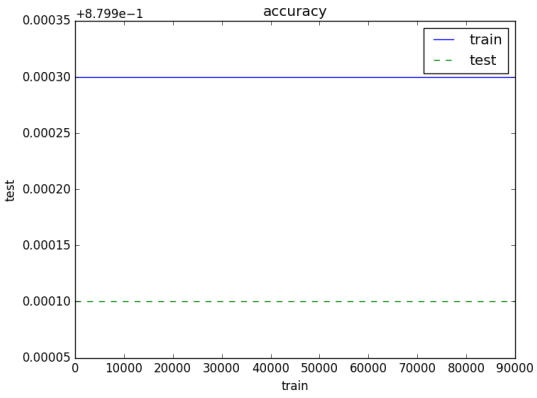
表 6: 2 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8802	0.88	0.0136283	0.00801849	−0.1664035
0.8802	0.88	0.1096312	0.11143688	−0.1662557
0.8802	0.88	0.11499672	0.116709	−0.16624038
0.8802	0.88	0.11985764	0.12151818	−0.16622376
0.8802	0.88	0.12258316	0.12420954	−0.16621461
0.8802	0.88	0.12471879	0.1263274	−0.16620672
0.8802	0.88	0.12620164	0.12778398	−0.16620239
0.8802	0.88	0.12767274	0.12923725	−0.16619751
0.8802	0.88	0.12938868	0.13094466	−0.16619086
0.8802	0.88	0.13120046	0.13274288	−0.16618423

以上の図とグラフから, 訓練データ, テストデータともに認識精度が一定である. さらに正解率から, すべての入力に対して”non-prime”と判定していることがわかる. また, $W^{(1)}$ については, ほとんどの値で増加している. この結果は, 学習が進んでいないことを意味する. よって, 有効な学習結果が得られなかった. つまり, 学習その 1 と同じ結果となった.

表 7: 3 回目

訓練データの認識精度	テストデータの認識精度	W_{11}	W_{12}	W_{13}
0.8802	0.88	0.01007679	0.01704089	−0.00308091
0.8802	0.88	0.12240685	0.12483788	−0.01330836
0.8802	0.88	0.12676277	0.12907985	−0.01290378
0.8802	0.88	0.13126434	0.13348154	−0.01258694
0.8802	0.88	0.13357271	0.13574469	−0.01234017
0.8802	0.88	0.13496128	0.13709747	−0.01213071
0.8802	0.88	0.13626672	0.13837905	−0.01192516
0.8802	0.88	0.13798484	0.1400699	−0.01175023
0.8802	0.88	0.13927062	0.14132954	−0.01157717
0.8802	0.88	0.14053134	0.14256884	−0.01140645



学習その 2

今後の課題

データの個数や特徴量を増やしたり. ミニバッチ学習をする. もっと層を深くしたり, 活性化関数の種類を増やす. また, 畳み込みニューラルネットワークや再帰型ネットワークを試すことが上げられる. さらに, これまでの判定機と早さと精度を比べることもしてみたい.

参考文献

[1] David M.Bressoud 著, 玉井浩 訳, 素因数分解と素数判定, エスアイビー・アクセス, 2004 年.

[2] 小高知宏, 機械学習と深層学習-C 言語によるシミュレーション-, オーム社, 2016 年.

[3] 斉藤康毅, ゼロから作る Deep Learning-Python で学ぶディープラーニングの理論と実装, オーム社, 2016 年.

[4] 柴田淳, みんなの Python 第 3 版, ソフトバンククリエイティブ株式会社, 2012 年.