応用数学

線形代数学

1. 行列

・行列の積：「行（行列の横方向）」×「列（行列の縦方向）」で行列の成分を求める。

・行基本変形：各行を○倍、行同士を加算・入れ替えして、連立方程式を解く。

・逆行列：行列の積が単位行列になる行列。

・逆行列の求め方：掃き出し法で求めることができる。

・逆行列が存在しない条件：行列式がゼロ。

・行列式の求め方：複雑な場合、行列式は展開できる。行を入れ替えると符号が変わる。

1. 固有値・固有ベクトル

・固有値：m×nの行列の場合、固有値はn個存在する。

・固有ベクトル：1つの固有値に対する固有ベクトルは複数存在する。（比率が重要）

・求め方：|A-λI|=0として、行列式を解く。

1. 固有値分解

・固有値分解：固有値と固有ベクトルを求め、A=VΛV-1に変換する。

1. 特異値分解

・特異値分解：正方行列以外の固有値分解で、M=USVTを満たす単位ベクトル。

・求め方：MMTを固有値分解するとMの特異値と特異ベクトルの2乗が求まる。

・利用例：特異値を減らすことで、画像のデータ量を効果的に減らせる。

統計学

1. 集合

・集合：集合は要素の集まりで、ある集合との関係（含まれるか）を区別できる。

・和集合：AまたはBに含まれる部分。

・共通部分：AとBの両方に含まれる部分。

・絶対補：Aでない部分。

・総体補：BのうちAに含まれない部分。

1. 確率

・確率：頻度確立（発生頻度）と、ベイズ確率（信念の度合い）の2つの考え方がある。

・確率の定義：P(A)=事象Aの起こる数/すべての事象が起こる数

・条件付き確率：P(A|B)=AとBの和集合/事象Bの確率

・独立事象の同時確率：因果関係のない事象の同時発生確率は確率の乗算で求まる。

・ベイズ則：AとBの和集合は、事象A条件下の場合と事象B条件下の場合は等価。

1. 統計

・期待値：ある確率分布の確率変数の平均値。

・標準偏差：期待値からのズレを大きさ（正の値）で示す。

・ベルヌーイ分布：コイントスのような表・裏の発生割合を示す。

・推定値：^（ハット）を付与して表す。

・不偏分散：標本分散のデータ数（サンプル）が少ないことによる影響を修正。

情報科学

1. 情報科学

・人は情報の増え方（わかりやすさ）を比率でとらえている。

・元の状況からの変化を考えた場合、増加比率の違いを考えるようなこと。

1. 自己情報量

・情報の変化を比率でとらえたもの(1/W)を積分するとlogになる。

・確率Pで表すとWの逆数なので、情報量は-log(P(x))になる。

1. シャノンエントロピ

・自己情報量の期待値

・例：コイントスすることで、どれだけ新しく情報を得ることができるか。

・シャノンエントロピが最大になる部分を求めることで予測できる。

1. ダイバージェンス

・カルバック・ライブラー　ダイバージェンス

・同じ事象・確率変数における異なる確率分布P,Qの違いを表す。

・ダイバージェンスは情報量よりも距離の概念に近い（ようなもの）。

・Qの情報量とPの情報量の差をもとに表現されている。

1. 交差エントロピ

・想定情報（暗号）による信号圧縮のようなもの

・Qについての自己情報量をPの分布で平均する。

機械学習

線形回帰モデル

1. 線形回帰モデル

・回帰問題を直線で予測する。

・教師あり学習で、教師データに正解データを含む。

・予測値は^（ハット）をつけ、正解データと異なることを明示する。

・線形結合：入力とパラメータの内積に切片を加える。

・モデルのパラメータ：予測値（出力）に対して特徴量の影響を決定する重みの集合。

・最小二乗法(MSE)によりパラメータを推定する。

1. データ分割/学習

・データは、学習用データ(train)と検証用データ(test)の２つに分割する必要がある。

・未学習のデータでモデルの汎化性能を測定するため。

1. ハンズオン

・E資格では、Scikit-learnやNumpyで実装を問われることがあるので要注意。

・fitは平均2乗誤差を最小にする点を探している。

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

考察

サンプルコードに一部誤りがあり、エラーとなったので、修正して再実行した。

Scikit-learnやNumpy、Pandasはデータ分析で、これまで使っていたので、難なく進めることができた。

非線形回帰モデル

1. 非線形回帰モデル

・複雑な非線形構造を内在する現象に対応する。

・線形結合：基底関数（非線形関数）とパラメータベクトルを使用。

・最小二乗法や最尤法によりパラメータを推定する。

・未学習：学習用データに対するモデルの表現力が低い。

・過学習：検証用データに対する汎化性能が低い。差が大きい。

1. 正則化法

・モデルの複雑さにより値が大きくなる正則化項をつける。

・Ridge推定量：制約を与えることで、パラメータを0に近づけるMSEを求める。

・Lasso推定量：ダイヤモンド型の制約で、いくつかのパラメータを正確に0にする。

・Lasso推定量（スパース推定）は計算コストを低減することができるメリットもある。

1. モデル選択

・ホールドアウト法：大量のデータがないと良い性能評価を与えない欠点がある。

・クロスバリデーション（交差検証）：学習用データと検証用データの組合せを変えた複数のイテレータでCV値（精度の平均）を計測する。

・汎用性能を測る場合、クロスバリデーションのほうが優秀な方法。

ロジスティック回帰モデル

1. ロジスティック回帰モデル

・数値入力からクラス分類する。

・特徴量を入力し、0または1の出力をデータとして扱う。

・線形結合：入力とパラメータの内積に切片を加える。

・シグモイド関数に線形結合を入力。

・シグモイド関数の微分は自身で表現可能。（予習教材で学習済み。）

1. 最尤推定

・尤度関数を最大化するパラメータを求める。

・対数尤度関数と尤度関数の最大点は同一なので、計算優位性のある対数を使用する。

1. 勾配降下法

・ある学習率で反復学習によりパラメータを更新する。

・対数尤度関数を係数とバイアスに関して偏微分する。

・データ容量や計算時間の問題を持っている。

1. 確率的勾配降下法

・データを確立的に選んでパラメータを更新する。

・データ容量や計算時間が少ないので、効率的に計算コストが使える。

1. モデルの評価

・混同行列：検証用データの結果とモデルの予測結果を表で分類し、各個数を表記。

・再現率：Positiveな検証用データをPositiveと予測できる割合。

・適合率：Positiveの予測結果の内、検証用データもPositiveな割合。

・F値：適合率と再現率の調和平均。値が高いと、適合率及び再現率がともに高い。

1. ハンズオン

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

地図, テキスト, スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

考察

データの中身を確認しながら実行することで、平均値で補完されている状況が理解できた。

入力値を変更することにより、予測結果が変わることも確認した。

可視化はこれまでもいくつか試したことはあったが、改めて色々な表現があり、もっと調べて試してみたいと感じた。

主成分分析

1. 主成分分析

・次元圧縮により分析や可視化を実現する。

・線形変換後の変数分散が最大となる射影軸を求める。

・無限に解があるので、ノルム1の制約を入れる。

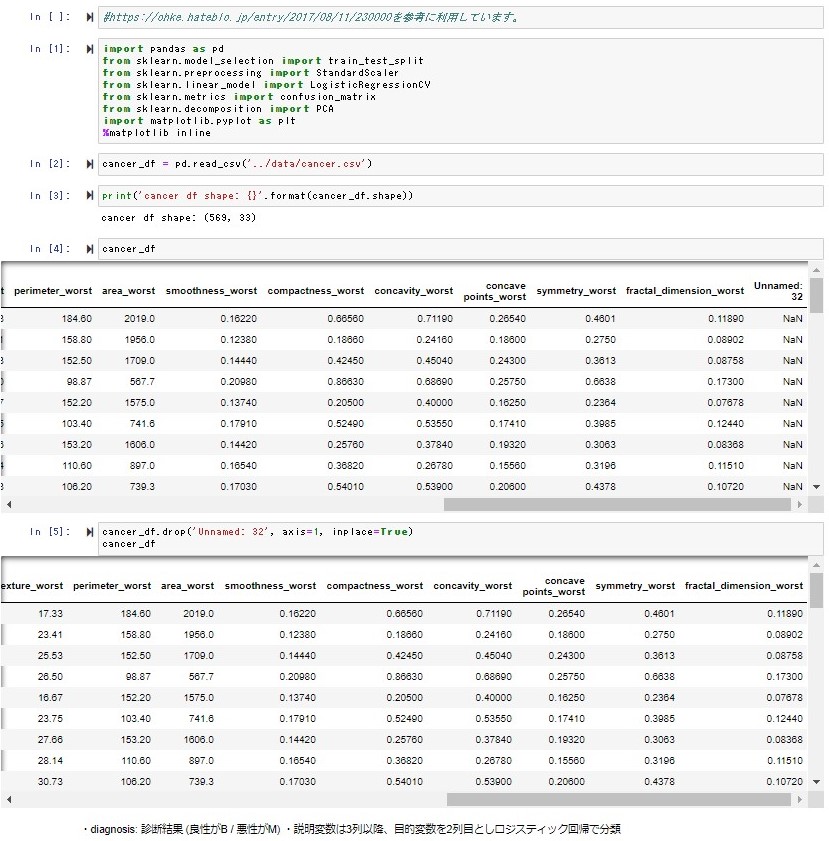
・ラグランジュ関数を微分して最適解を求めることは、分散共分散行列の固有値と固有ベクトルを求めることになる。

・第1主成分と第2主成分は直交する。

・寄与率：ある主成分分散の全分散に対する割合は、ある主成分の情報量の割合となる。

・累積寄与率を求めることで、圧縮による情報損失量の割合を示す。

1. ハンズオン



スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

考察

30次元のデータで97%の分類精度があることが理解できた。

寄与率をグラフ表示することで、どのくらいの情報損失量かを可視化できる。

2次元に落とすと65%の情報量なので、結果があいまいになることがわかった。

アルゴリズム

1. k近傍法

・最近傍のデータk個が最も多く所属するクラスに分類する。

・kの数を変化させると分類結果も変化する。決定境界が滑らかになっていく。

・ハンズオン

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

1. k平均法(k-means)

・与えられたデータをk個のクラスタに分類する。

・クラスタリング：特徴の似ているもの同士をグループ化する。

・各クラスタの中心と全データの距離を計算し、最適な各クラスタの中心を求める。

・各クラスタの中心の初期値が重要。ランダムで与えると結果がよい。

・教師なし学習の１つ。

・ハンズオン

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

サポートベクターマシン

・線形モデルの正負で2値分類する。

・線形結合：入力とパラメータの内積に切片を加える。

・符号関数に線形結合を入力。

・マージン：線形判別関数と最近傍点の距離。

・マージンが最大となる線形判別関数を求める。

・分離超平面を構成する学習データは、サポートベクターだけで残りのデータは不要。

・ソフトマージンSVM：線形分離できない場合、誤差を許容しペナルティを与える。

・トレードオフパラメータの大小で決定境界が変化する。

・非線形カーネル（ガウシアン）を用いて分離を可視化できる。

・特徴空間に写像することで、線形分離する。

・ハンズオン

スクリーンショット が含まれている画像

自動的に生成された説明

テキスト, スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

深層学習：前編１

Section1：入力層～中間層

・入力毎に重みがかけられ、次の中間層に情報を伝播する。

・入力毎にインデックスとしてjが与えられる。

・各層にバイアス項が加えられる。

Section2：活性化関数

・次の層への出力の大きさを決める非線形関数。

・次の層への信号伝播のON/OFFや強弱を決める。関数を変えると伝播が変わる。

・ステップ関数：0または１の結果を与える。線形分離可能でないと使えない。

・シグモイド関数：0～1を緩やかに変化する結果を与える。勾配消失問題の課題あり。

・RELU関数：0以上で入力値をそのまま結果として与える。現在の主流。

Section3：出力層

1. 誤差関数

・出力層はある関数で処理することで、人間が知覚できる理論値として算出できる。

・出力結果の数値のばらつきが少ないと学習データが少ないということがある。

・出力（予測値）と訓練データ（正解値）の誤差について誤差関数を使用する。

・誤差を最小化していくことが学習の目標で、誤差関数はツール。

1. 出力層の活性化関数

・出力層と中間層では活性化関数が異なる。

・信号の大きさ（比率）をそのままに変換。（価値はそのままに。）

・分類問題の場合、出力層の出力の総和を1にする必要がある。

・恒等写像：回帰の場合で、誤差関数は二乗誤差を使用。

・ソフトマックス関数：多クラス分類で誤差関数は交差エントロピを使用。

・シグモイド関数：二乗分類で誤差関数は交差エントロピを使用。

・上記、活性化関数と誤差関数の組合せで計算の相性がよい。

Section4：勾配降下法

1. 勾配降下法

・学習を繰り返すことで、誤差を最小にするパラメータを求める。（最適化する。）

・学習率が大きすぎると最小値にたどり着かず発散する。

・学習率が小さすぎると収束まで時間がかかる。

・大域的極小解に収束させたい。

・学習率の決定も考慮が必要。

・Momentum、AdaGrad、Adadelta、Adamの向上アルゴリズムがある。

・エポック：学習回数

1. 確率的勾配降下法

・ランダムに抽出したサンプルの誤差からパラメータを最適化する。

・計算コスト削減と局所極小解への収束リスク軽減のメリットがある。

・オンライン学習が可能になる。

1. ミニバッチ勾配降下法

・ランダムに分割したミニバッチ（データ集合）の平均誤差から最適化する。

・確率的勾配降下法のメリットに加え、計算資源を有効活用できるメリットがある。

・CPUのスレッド並列化も活用できる。

Section5：誤差逆伝播法

・誤差を出力層から微分し、入力層まで逆伝播する。

・再帰的計算を避け、最小限の計算で微分値を計算する。

・順番に偏微分を繰り返す。

確認テスト

全体像

Q：ディープラーニングは、結局何をやろうとしているのか2行以内で述べよ。

また、次の中のどの値を最適化が最終目標か。全て選べ。（1分）

①入力値[X] ②出力値[Y] ③重み[W] ④バイアス[b]

⑤総入力[u] ⑥中間層入力[z] ⑦学習率[p]

A：誤差を最小化するパラメータを求めること。

③重み[w]と④バイアス[b]

Q：次のネットワークを紙にかけ。

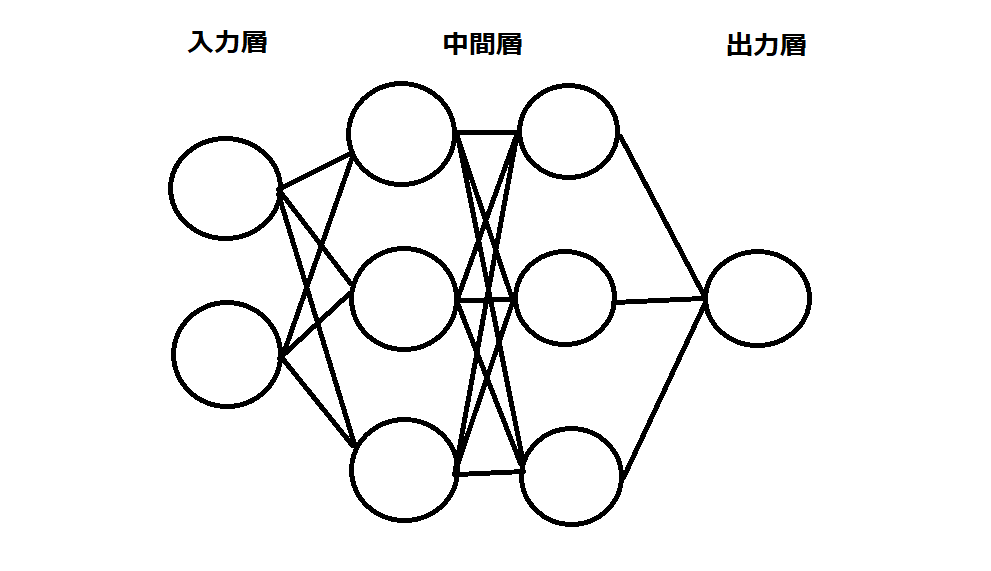
入力層：2ノード1層

中間層：3ノード2層

出力層：1ノード1層

（5分）

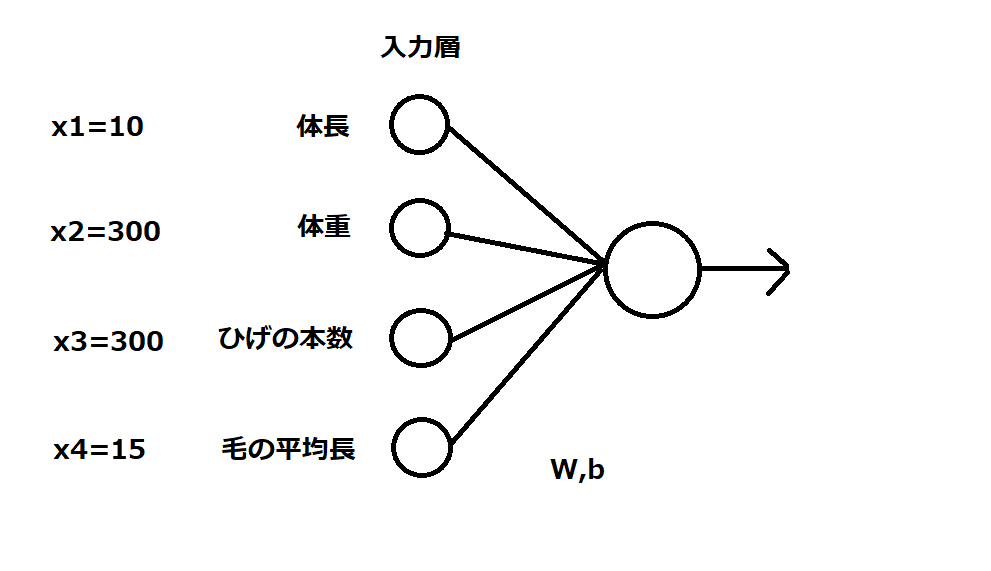
A：



Section1：入力層～中間層

Q：この図式に動物分類の実例を入れてみよう。（3分）

A：



Q：この数式をPythonで書け。（3分）NumPyのライブラリを使う。

A：u = np.dot(x, W) + b

次元が合わないとエラーになることがあるので要注意。

Q：1-1のファイルから中間層の出力を定義しているソースを抜き出せ。（3分）

A：

# 中間層出力

z = functions.relu(u) <<--

print\_vec("中間層出力", z)

Section2：活性化関数

Q：線形と非線形の違いを図にかいて簡単に説明せよ。（3分）

A：データ点の分布の特徴に近い線を引くとき、直線で表せるのが線形、曲線で表せるのが非線形となる。

空, 写真 が含まれている画像

自動的に生成された説明

Q：配布されたソースコードより該当する箇所を抜き出せ。（3分）

A：

# 中間層出力

z = functions.sigmoid(u) <<--

print\_vec("中間層出力", z)

Section3：出力層

Q：なぜ、引き算ではなく二乗するのか述べよ。

下式の1/2はどういう意味を持つか述べよ。（2分）

A：大きさ（正符号）で結果を求めたいから。

1/2は二乗を微分した時に最終的な計算を簡単にするため。

Q：①～③の数式に該当するソースコードを示し、一行づつ処理の説明をせよ。（5分）

A：①def softmax(x)

②np.exp(x)

③np.sum(np.exp(x), axis=0)

xが2次元の場合、xを転置して扱い、xの最大値を減算した値をxとして、yを計算し、yを転置して戻り値を返す。

Q：①～②の数式に該当するソースコードを示し、一行づつ処理の説明をせよ。（5分）

A：①def cross\_entropy\_error(d,y)

②-np.sum(np.log(y[np.arange(batch\_size),d]+1e-7))/batch\_size

yが1次元の場合、dとyの配列形状を変換し、dとyのサイズが同じなら、dを最大値にする。yのサイズをbatch sizeとして、交差エントロピを計算する。

※1e-7は微小な値を与えて0にならない実装上の工夫がされている。

Section4：勾配降下法

Q：該当するソースコードを探してみよう。（1分）

A：

grad = backward(x, d, z1, y) <<--∇Eの式

for key in ('W1', 'W2', 'b1', 'b2'):

network[key] -= learning\_rate \* grad[key] <<--wの式

Q：オンライン学習とは何か。2行でまとめよ。（2分）

A：既存の学習済みモデルに対して、利用しているユーザーからの情報のみで再学習することことができる。

Q：この数式の意味を図に書いて説明せよ。（5分）

A：

テキスト が含まれている画像

自動的に生成された説明

Section5：誤差逆伝播法

Q：誤差逆伝播法では不要な再帰的処理を避ける事が出来る。

既に行った計算結果を保持しているソースコードを抽出せよ。（3分）

A：

# 誤差逆伝播

def backward(x, d, z1, y):

print("\n##### 誤差逆伝播開始 #####")

grad = {}

W1, W2 = network['W1'], network['W2']

b1, b2 = network['b1'], network['b2']

# 出力層でのデルタ

delta2 = functions.d\_sigmoid\_with\_loss(d, y)

# b2の勾配

grad['b2'] = np.sum(delta2, axis=0)

# W2の勾配

grad['W2'] = np.dot(z1.T, delta2)

# 中間層でのデルタ

delta1 = np.dot(delta2, W2.T) \* functions.d\_relu(z1)

# b1の勾配

grad['b1'] = np.sum(delta1, axis=0)

# W1の勾配

grad['W1'] = np.dot(x.T, delta1)

print\_vec("偏微分\_dE/du2", delta2)

print\_vec("偏微分\_dE/du2", delta1)

print\_vec("偏微分\_重み1", grad["W1"])

print\_vec("偏微分\_重み2", grad["W2"])

print\_vec("偏微分\_バイアス1", grad["b1"])

print\_vec("偏微分\_バイアス2", grad["b2"])

return grad

Q：2つの空欄に該当するソースコードを探せ。（3分）

A：

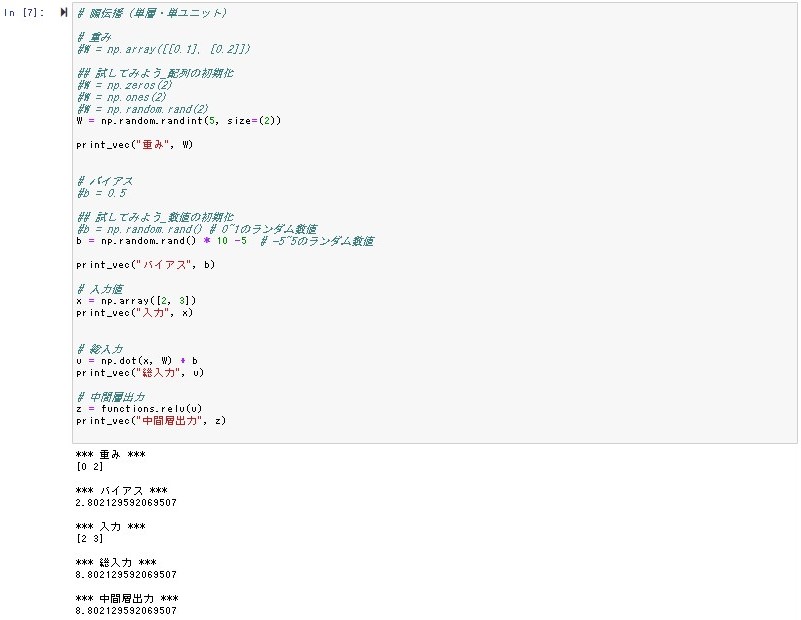
uでの微分：delta2 = functions.d\_mean\_squared\_error(d, y)

w2での微分：grad['W2'] = np.dot(z1.T, delta2)

演習問題

Section1：入力層～中間層

重み項での配列初期化とバイアス項での数値の初期化



考察

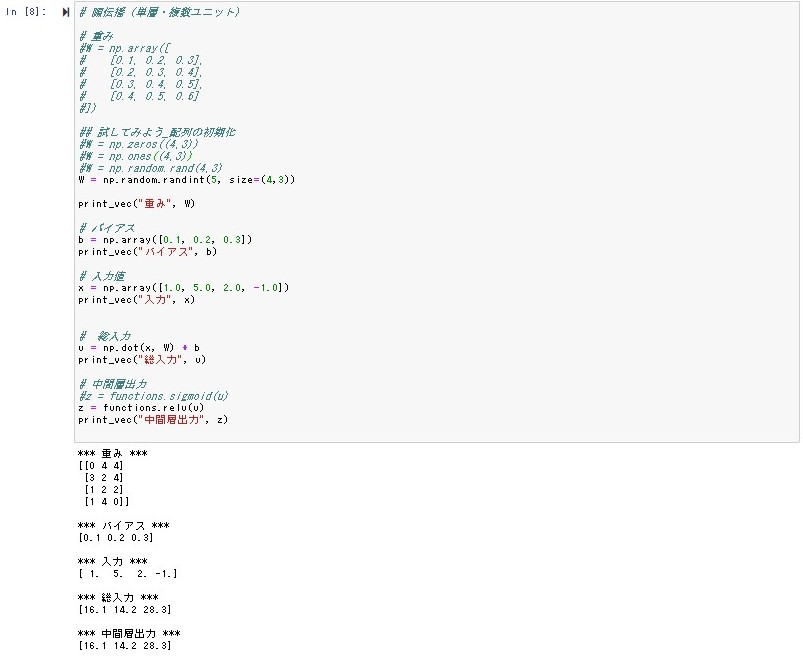
・NumPyを活用した初期値の与え方を試して学んだ。

・NumPyがランダムに初期値を生成してくれるのは、記述も簡単で非常に使い易い。

・重みの配列を増やし、次元数の矛盾によるエラー内容の確認と、入力の次元数を増やし対処することも試し確認した。

Section2：活性化関数

重み項での配列初期化と活性化関数の変更



考察

・単層・複数ノードでNumPyを活用した初期値の与え方を試して学んだ。

・活性化関数をシグモイド関数からRELU関数に変更し、変化を確認した。

・from common import functionsで、functions.pyを読み込んでいることも理解した。

Section5：誤差逆伝播法

RELU関数からシグモイド関数に変更する。入力値の設定を変更する。

テキスト, スクリーンショット が含まれている画像

自動的に生成された説明

テキスト が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

考察

・サンプルコードがRELU関数なのに、中間層でのデルタ計算にシグモイド関数の導関数を使用していたので、修正し実行した。

・RELU関数からシグモイド関数に変更するとグラフのばらつきが増えることを確認した。

・入力値の設定を変更すると更にグラフのばらつきが増えることを確認した。

修了課題

Q1.課題の目的とは？　どのような工夫ができそうか。

A1.構造図（設計書）どおりの実装ができるか。

設定値を変更できるようにする。

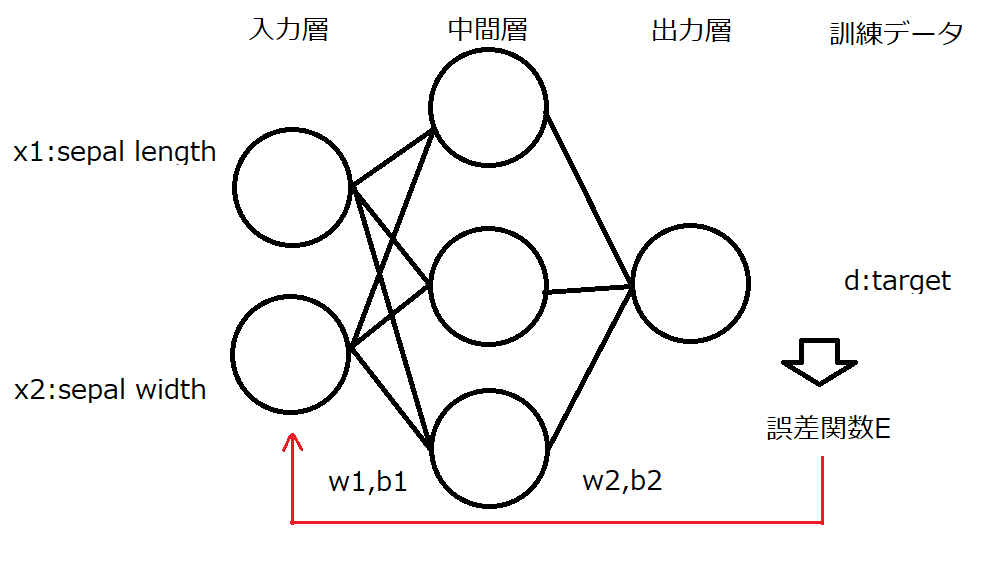
実データを加工して、入力値を変更できるようにする。

Q2.課題を分類タスクで解く場合の意味は何か。

A2.irisデータが判別分析用データセットであることと、予測結果との誤差がわかりやすい。

Q3.irisデータとは何か2行で述べよ。

A3.3種類のあやめに関する 4個の計測値からなる判別分析やクラスタ分析などの研究に使われるテストデータ。scikit-learnのデータセットとして準備されている。



スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

スクリーンショット が含まれている画像

自動的に生成された説明

参照ソースコード　－　iris-practice.ipynb