山田　克拡 提出レポート　深層学習前半［Day1とDay2］（講義動画と実装演習）

**深層学習: 前半Day1**

**各章の要点**

**Section1：入力層～中間層**■入力層のそれぞれのノードの入力値xに対して、重みｗを掛けてバイアスｂを足した値が第１の中間層のノードに入力される。

■重みｗijについて、入力層のノードiと第一の中間層のノードjの組み合わせごとに異なる値を取る。

**Section2：活性化関数**

■活性化関数とは、ニューラルネットワークにおいて、次の層への出力の大きさを決める非線形関数である。

■中間層用の活性化関数としては、シグモイド関数やReLU関数、ステップ関数が使用され、出力層の活性化関数としては、恒等関数やソフトマックス関数、シグモイド関数が使用される。

■シグモイド関数は勾配消失問題を引き起こすため、代わりにReLU関数が使用されることが多い。  
■RNNではハイパボリックタンジェント関数が使われる。

Section3：出力層

■回帰の場合、出力層の活性化関数は恒等関数を使用する。２値分類の場合はシグモイド関数、多クラス分類の場合はソフトマックス関数を使用する。

■出力層からの出力データと正解値の誤差を評価するため誤差関数を用いる。誤差関数が小さくなれば、予測精度が上がる。回帰の場合は二乗誤差関数、分類の場合は交差エントロピー誤差関数を使用する。  
Section4：勾配降下法

■勾配降下法は誤差関数の値が最小になる重みｗを見つける方法である。

　学習率と誤差関数を重みｗで偏微分した値を使って重みｗを更新する。

■勾配降下法の学習率が大きいと、移動量が大きすぎて最小値にたどり

つかず発散してしまう。学習率が小さいと最小値にたどりつくのに

時間がかかる。

■バッチ学習用の勾配降下法や、ミニバッチ学習用のミニバッチ勾配降下法、

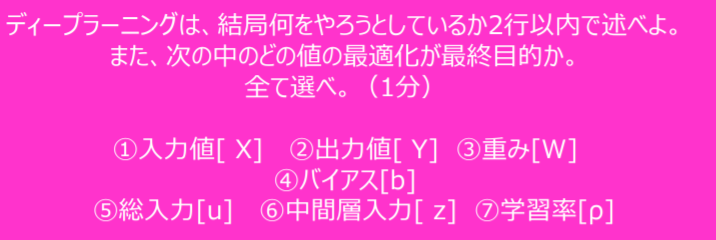
オンライン学習用の確率的勾配降下法がある。

Section5：誤差逆伝播法

■重みやバイアスを更新するために、誤差関数を各層各ノードの重み、バイアスで微分する必要がある。微分の計算を準伝播で計算すると計算コストが高くなる課題がある。誤差逆伝播法では、不要な再帰的処理を行う必要が無いため、最小限の計算で微分値を計算し、重みの計算ができる。

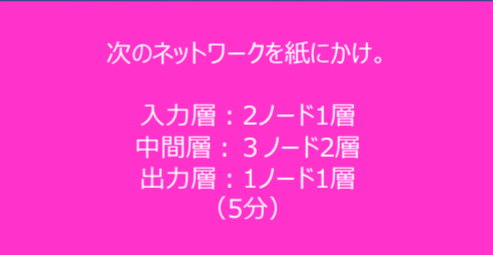
確認テストの考察

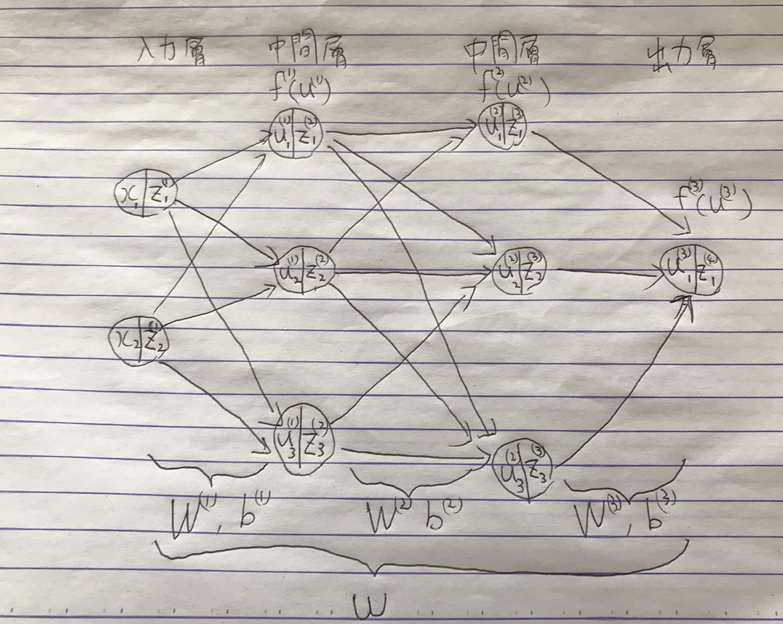
Day1 【ｐ10】



②出力値Yを最適化するのが最終目的である。誤差関数が小さくなるように重みを最適化したモデルを作成し、未知/既知の入力値Xに対して、誤差が小さい出力値Yを予測する。

【Page 12】

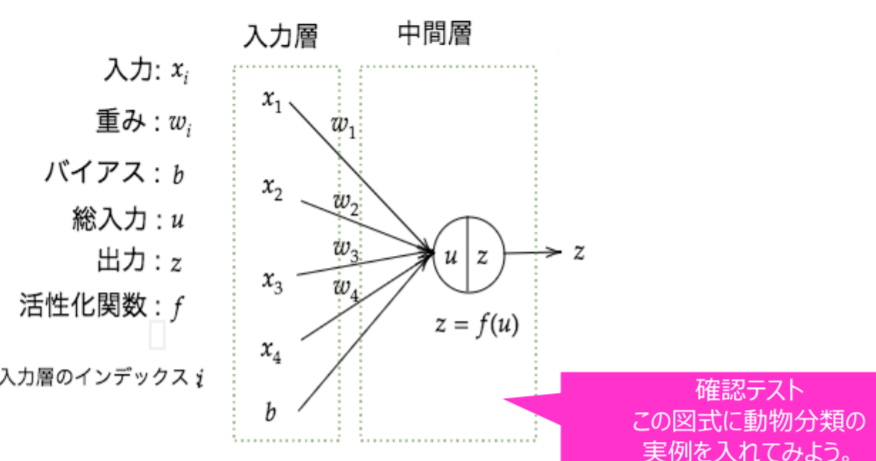




【Page 19】

■考察

赤字部のとおり。



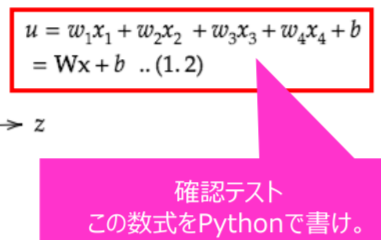
**体長**

**足の長さ**

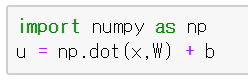
**体重**

**手の長さ**

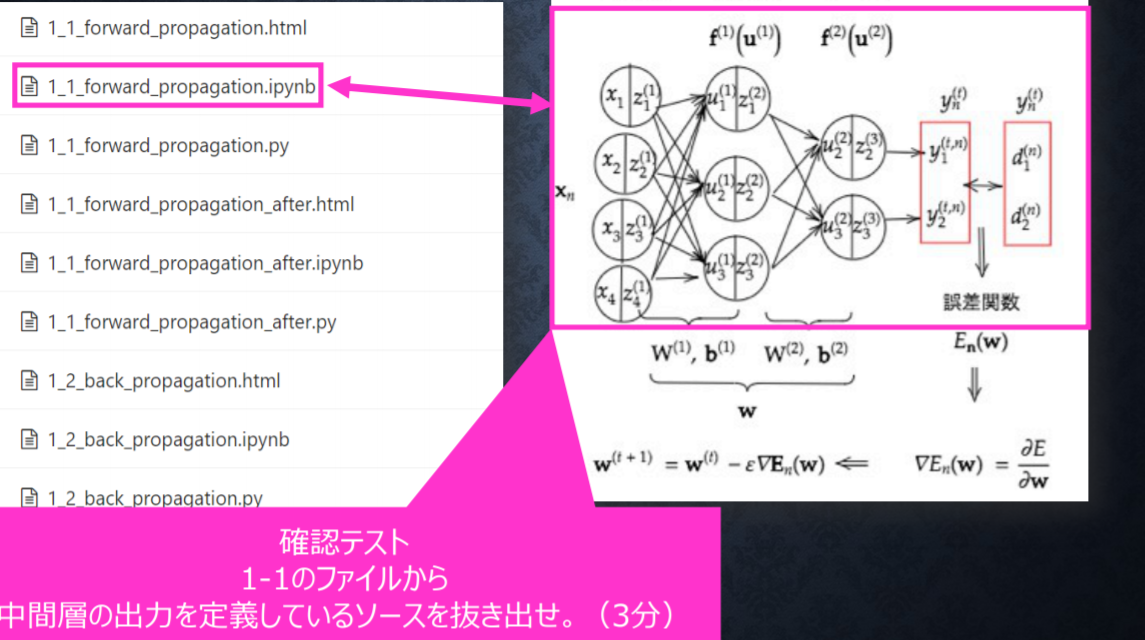
【Page 21】



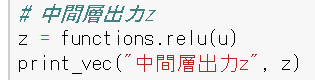
■考察



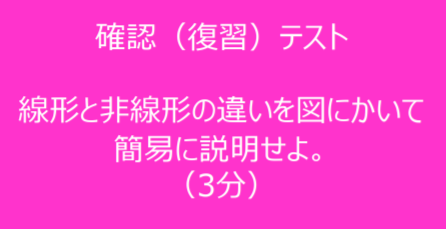
【Page 23】



■考察

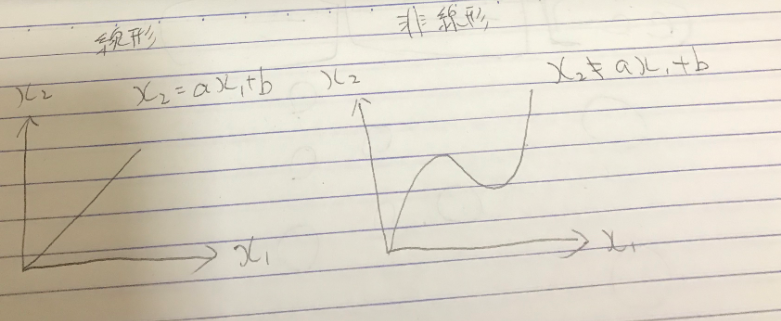


【ｐ26】

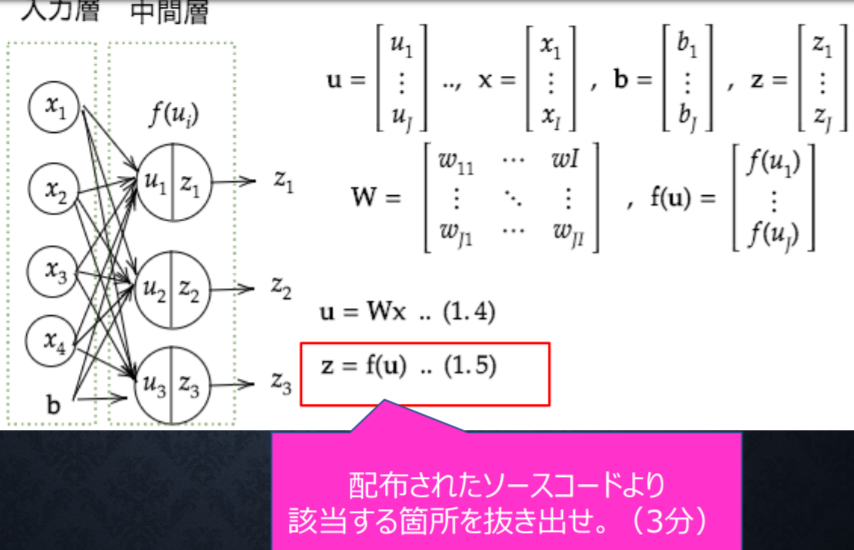


■考察

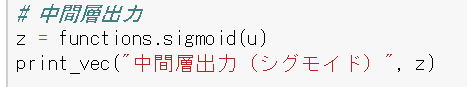
線形は変数間の関係を1次式で表すことができ、非線形はできない。



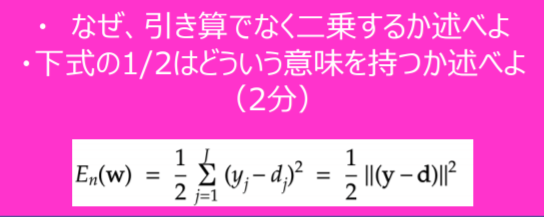
【ｐ３３】



■考察



【ｐ４４】

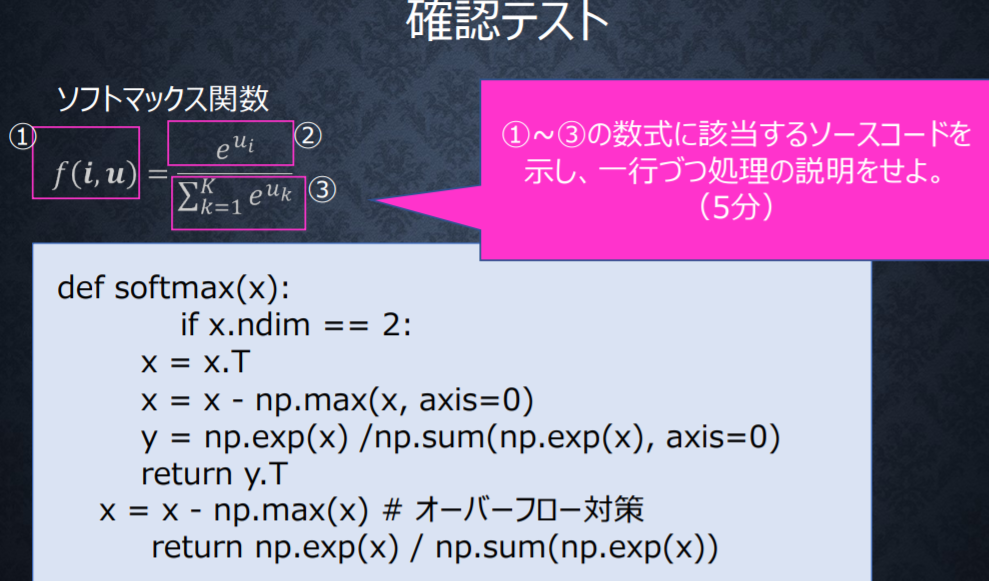


■考察

誤差を二乗しない場合、一つ目の誤差が＋１で、２つ目の誤差がー１だとすると、誤差を足し合わせると、合計が０になり、誤差が打ち消しあうため正しく評価できない。

誤差を二乗すれば、誤差の合計は（１）２＋（－１）２＝２となり、誤差を正しく評価できる。1/２は、誤差関数の微分を計算しやすくするためである。

【ｐ５１】



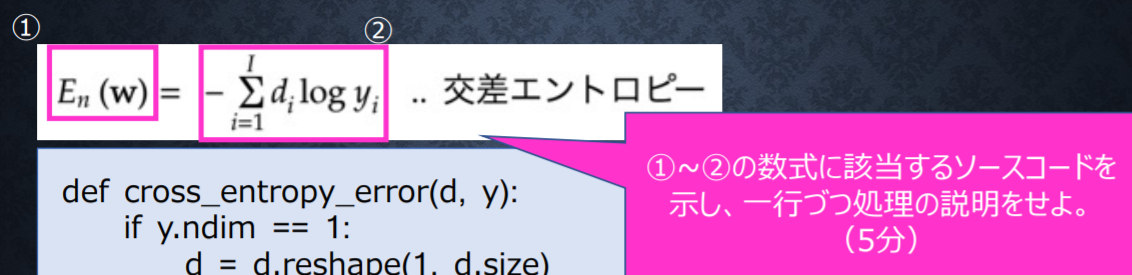
■考察

①がdef softmax(x): で、配列ｘに対して配列ｙを返す関数である。

②がnp.exp(x)で、Ｎｕｍｐｙのexp関数を使用してexの計算をしている。

③がnp.sum(np.exp(x),axis=0)で、Ｎｕｍｐｙのexp関数とsum関数を使用してexの合計値を計算している。

【p53】



■考察

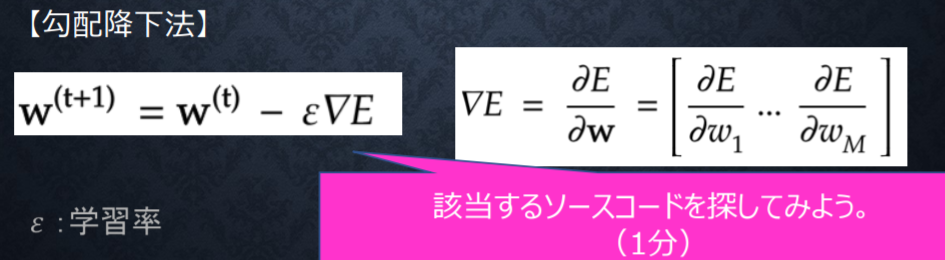
①def cross\_entropy\_error(d,y):

公差エントロピー誤差を求める関数である。

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

1e-7を加算しているのは発散対策である。

【ｐ５６】



■考察



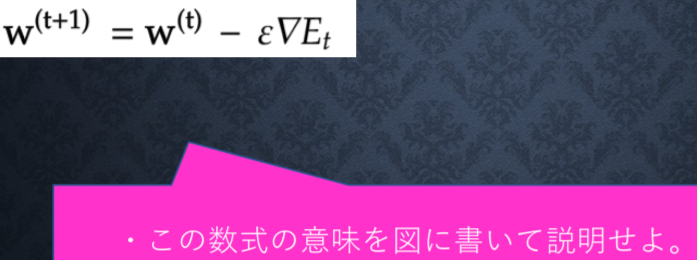
【ｐ６５】



■考察

オンライン学習とは、毎回１つのデータをランサムサンプリングして、重みｗを更新する方法である。

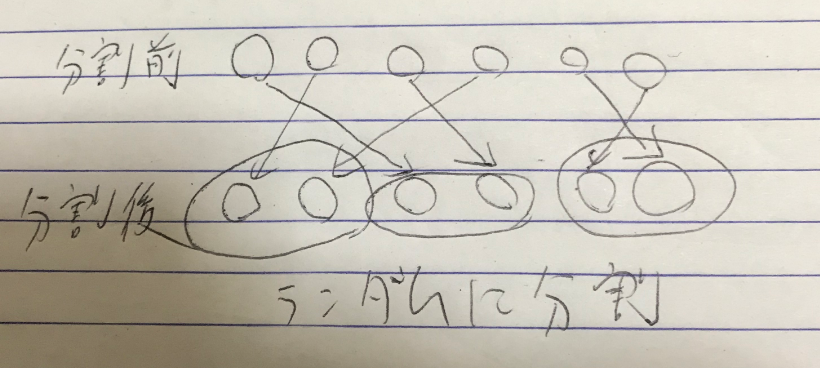
【ｐ68】



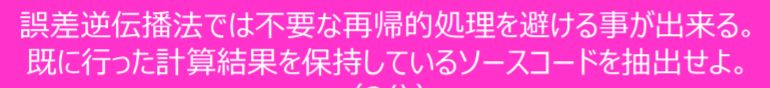
■考察

データをランダムに分割し、分割したデータに対して重みｗの更新量を計算し、

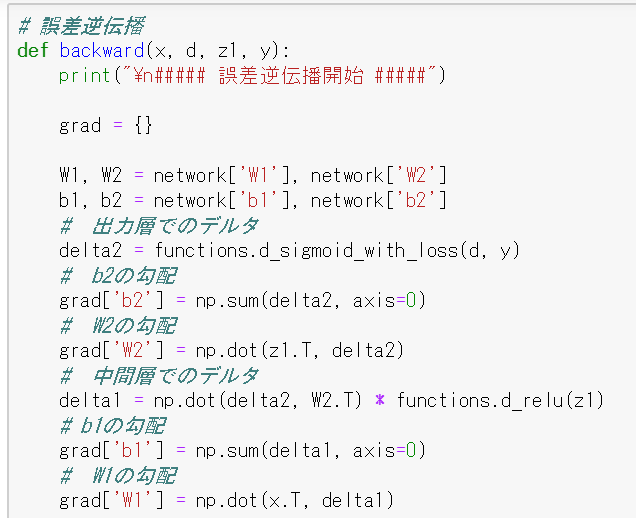
重みを計算している。



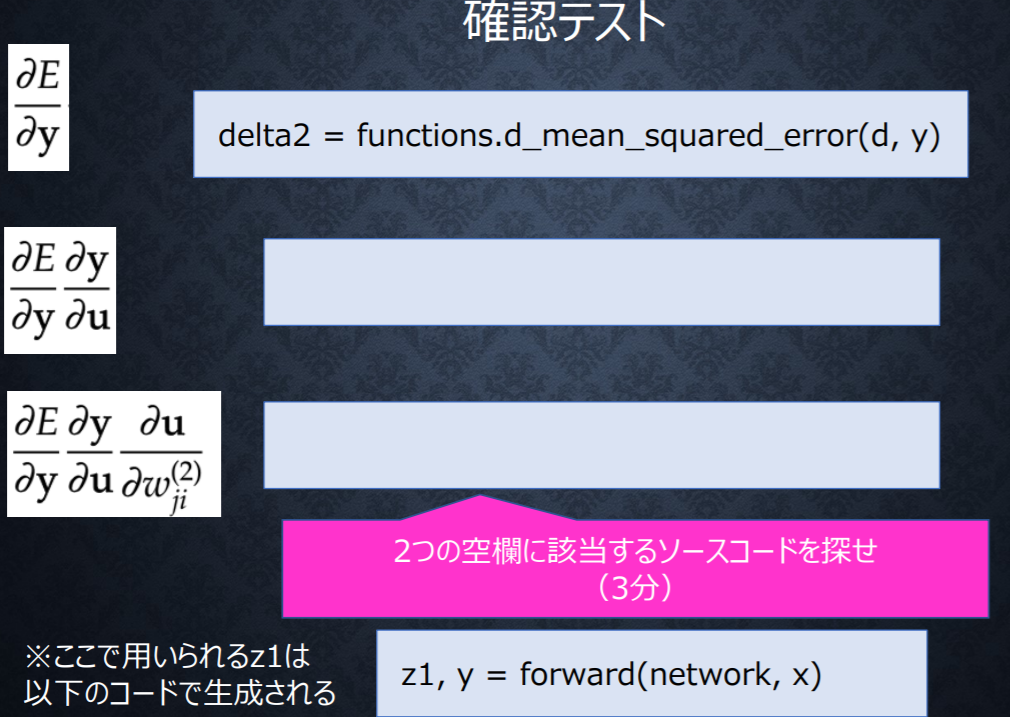
【ｐ７８】



■考察



【ｐ８３】

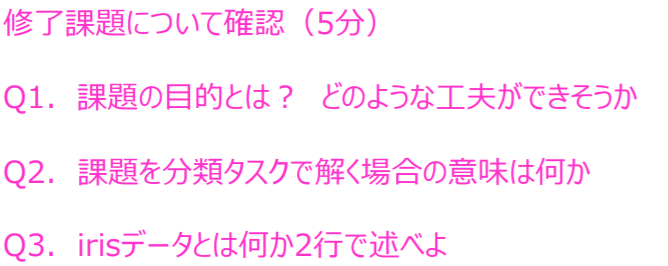


■考察





【ｐ８７】



■考察

Q1　修了課題の目的は、DNNのアルゴリズムを使い、３つの説明変数からあやめ（アイリス）の花びらの長さを予測する、汎化性能の良い学習済みモデルを作成することである。

工夫点としては、勾配消失問題を起こさないようにReLU関数を使用することである。

Q2　あやめの花の種類が何種類かある場合、分類タスクで解くとあやめの花の種類を予測する学習済みモデルを作成することも可能である。

Q3　irisデータとはscikit-learnのライブラリ内にあるデータセットで、がく片の長さ , がく片の幅 , 花びらの長さ , 花びらの幅などのデータである。Irisはあやめを示す。

**演習考察**

[**DN06\_Jupyter演習**](https://vimeo.com/showcase/5575638/video/302234447)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/1_1_forward_propagation-ensyu.pdf>

■考察

・重みとバイアスの値の設定には、すべて０or1にする方法や乱数を取る方法がある。

[**DN15\_Jupyter演習２**](https://vimeo.com/showcase/5575638/video/302231315)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/1_3_stochastic_gradient_descent-ensyu.pdf>

■考察

・訓練データのｘ値を変更しただけで、モデルの収束に影響があることがわかった。

　これはｘ値を大きくするとｙ値が大きくなりやすいので誤差が大きくなり収束しにくく

なっていると考える。

**深層学習: 前半Day2**

**各章の要点**

Section1：勾配消失問題

シグモイド関数を微分したときの最大値が0.25なので、中間層が2個だと最大0.0625、3個だと最大0.015625といった感じで、シグモイド関数を使うと誤差逆伝播法で勾配消失が起きる。対策として、微分値が１であるReLU関数を使うとよい。重みの初期値設定はXavierかHeを使うとさらに勾配消失が起きにくい。  
Section2：学習率最適化手法

学習率が大きすぎると最適値にたどり着かず発散し、小さすぎると収束するまでに時間がかかる。学習率最適化手法とは、誤差関数が小さくなるように学習率を最適化する手法である。SGD、モメンタム、AdaGrad、RMSprop、Adamなどがある。

Section3：過学習  
過学習とは、特定の訓練データにフィットしすぎるように学習したために、訓練誤差とテスト誤差が乖離することを示す。パラメータやノードの数や層数が大きすぎたり、パラメータの値が大きいと過学習になりやすくなる。L1/L2正則化やDROPOUTなどの正則化手法を用いることで過学習を抑制できる。

Section4：畳み込みニューラルネットワークの概念  
畳み込みニューラルネットワークの構造は、入力層、畳み込み層、プーリング層、全結合層、

出力層からなる。入力層では画像データや音声データを入力し、畳み込み層では入力データにフィルターの値を掛けわせてバイアスを加算し、プーリング層では最大プーリングもしくは平均プーリングで画像サイズを圧縮し、全結合層では分類を行う。

Section5：最新のCNN

最新のCNNとして、2012年の画像コンペで優勝したCNNモデルのAlexNetがある。

AlexNetは５層の畳み込み層とプーリング層、3層の全結合層から構成されている。

過学習を防ぐために、全結合層でドロップアウトを使用する工夫も行っている。

確認テストの考察

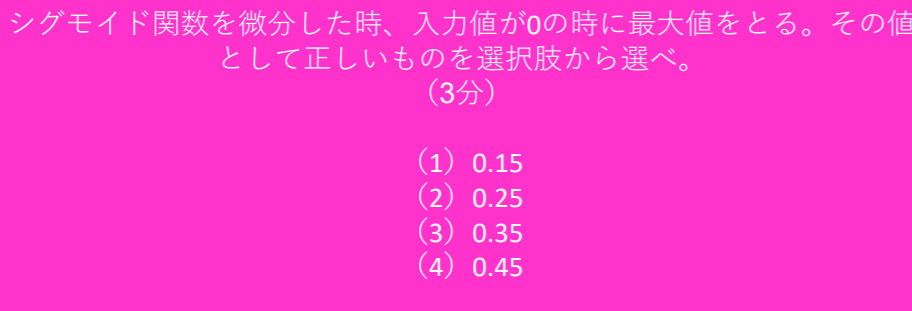
【ｐ１２】



■考察

dz/dx=(dz/dt)\*(dt/dx)=(2t)\*(1)=2(x+y)

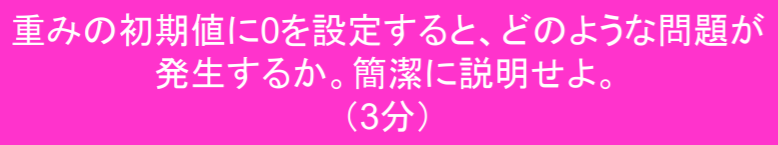
【ｐ２０】



■考察

（１－sigmoid(0)）\*sigmoid(0)=(1-0.5)\*0.5=0.25なので、（２）である。

【ｐ28】



■考察

重みの初期値を０にすると、誤差逆伝播法の誤差関数を重みで微分した値が０になり、重みの更新量が０になり、重みを更新しても値が変わらない。

【ｐ31】



■考察

①学習が安定し高速化できる。

②勾配消失が起きづらくなる。

【ｐ３５】



■考察

（き）に当てはまるのは、（１）である。

バッチサイズ分だけデータを取り出している。

【ｐ４７】



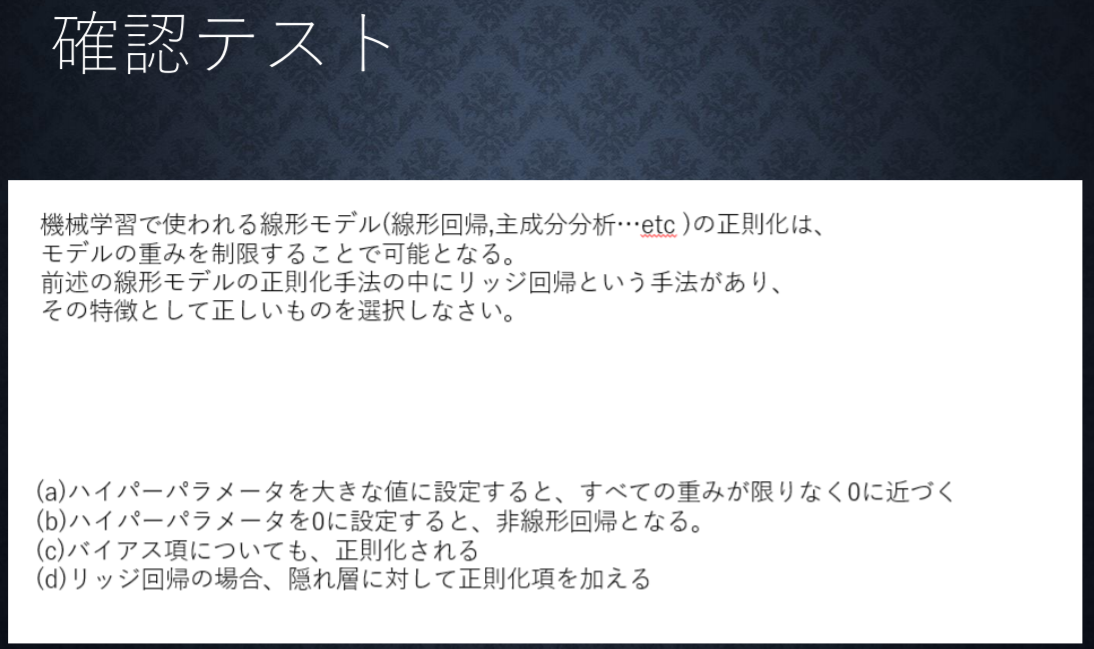
■考察

モーメンタムは谷に落ちてからの収束が早い。

AdaGradは勾配な緩やかな斜面でも最適値に近づきやすい。

RMSPropはパラメータの調整が少なくてすむ。

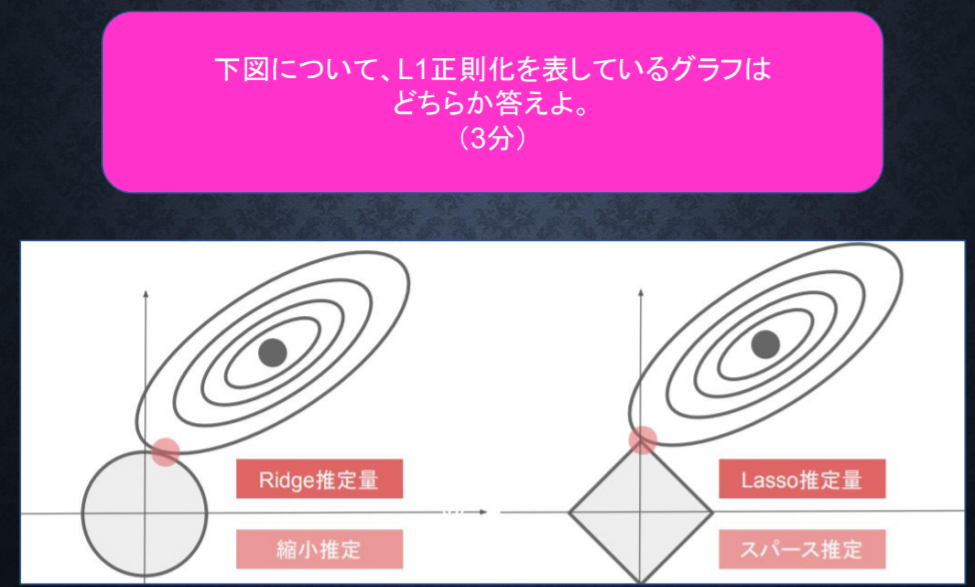
　【ｐ63】



■考察

正解は（a）である。

【ｐ６８】



■考察

L1正則化を表すグラフは、右のグラフである。

【ｐ69】

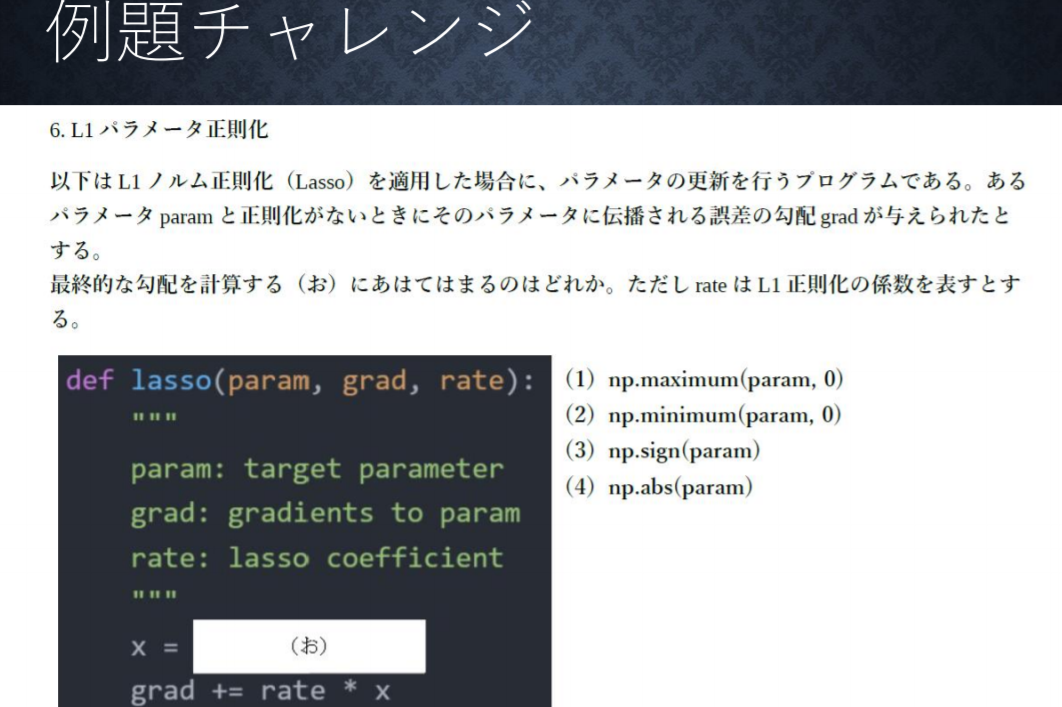


■考察

（え）にあてはまるのは、（４）である。

L2正則化項を微分するとparamになる。

【ｐ７１】



■考察

（お）にあてはまるのは、（３）である。

L1正則化項を微分すると、符号だけ残る。

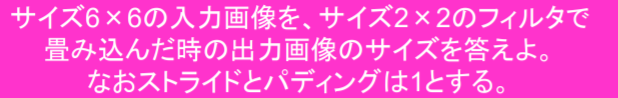
【ｐ７３】



■考察

（か）にあてはまるのは、（４）である。

【ｐ100】



■考察

　出力画像サイズは、７×７である。

（６＋２×１―２）/１＋１＝７

**演習考察**

[**DN23\_Jupyter演習**](https://vimeo.com/showcase/5577593/video/302561669)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/2_2_1_vanishing_gradient-ensyu.pdf>■考察

①シグモイド関数とガウス関数の組み合わせだと、勾配消失が起きている。

②①において、シグモイド関数をReLU関数にすると勾配消失が改善する。

③①において、ガウス関数をXavier関数にすると勾配消失が改善する。

④①において、シグモイド関数とガウス関数をReLU関数とHe関数にすると勾配消失

問題の改善効果が大きい。

⑤シグモイド関数にHe関数を組み合わせた場合、ガウス関数やXavier関数を

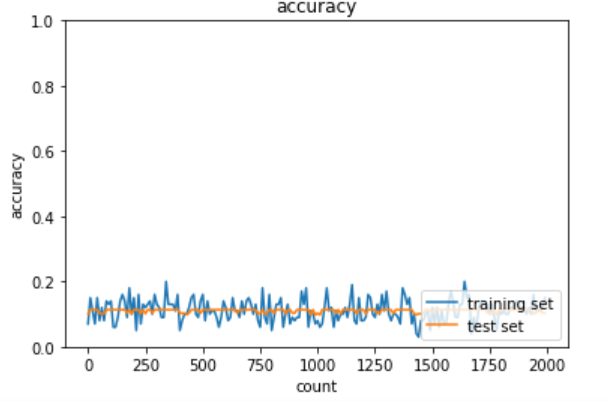
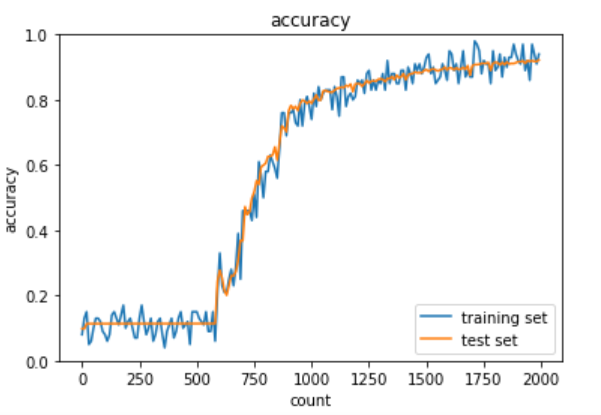
組み合わせた場合よりも、汎化性能が高い。

⑥ReLu関数にHe関数を組み合わせた場合、ガウス関数やXavier関数を組み合わせた

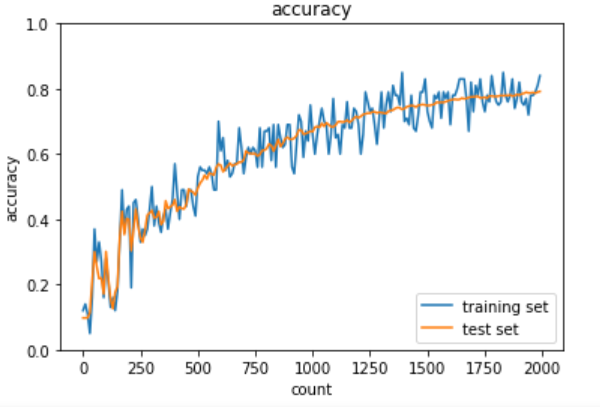
場合よりも、汎化性能が高い。

⑦隠れ層のノードを増やすと、汎化性能が高くなった。

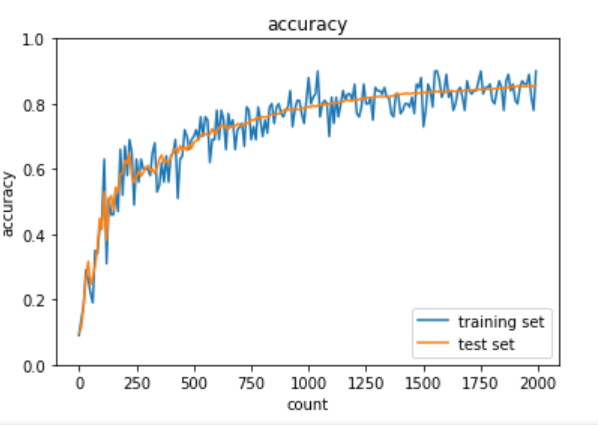
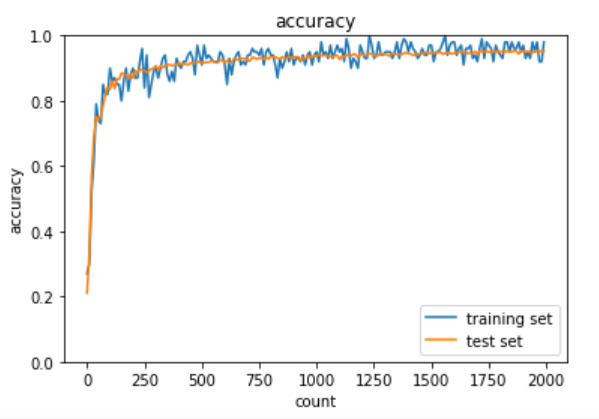
Sigmoid-Gauss　　　　　　　　　　　　　　　　ReLU-Gauss

　Sigmoid-Xavier　　　　　　　　　　　　　　　ReLU-He

Sigmoid-He　　　　　　　　　　　　　　　　　　ReLU-Xavier

### [**DN26\_Jupyter演習(2)**](https://vimeo.com/showcase/5577593/video/302389021)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/2_4_optimizer-ensyu.pdf>

■考察

・SGDで学習率を上げると正答率が上がる。

　その理由は、０から大幅に離れた入力値だとシグモイド関数の微分値はほぼ０

になるため、勾配消失が起きやすいが、学習率を上げると誤差関数が小さくなり

やすいためである。

・SGDで活性関数をシグモイド関数からReLU関数に変更し、重みの初期値をHe関数

　で決めると、正答率が上がる。

　その理由は、０から大幅に離れた入力値だとシグモイド関数の微分値はほぼ０

になるため、勾配消失が起きやすいが、ReLU関数だと勾配消失しにくいため。

・SGDやモメンタムをバッチ正規化すると正答率が上がる。

　その理由は、０から大幅に離れた入力値だとシグモイド関数の微分値はほぼ０

になるため、勾配消失が起きやすいが、正規化により入力値が０付近になるため、

勾配消失が起きにくくなる。

・モメンタムの慣性を小さくすると、誤差関数が最小値にたどりつくのが遅くなり、

　予測精度が下がる。

・バッチ正規化すると、学習速度が上がり、予測精度が上がる。

### [**DN32\_Jupyter演習(Dropout)**](https://vimeo.com/showcase/5577593/video/302806130)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/2_5_overfiting-ensyu.pdf>

■考察

・訓練データが少ないと、過学習を起こしやすくなる。

・Ｌ２正則化項を追加すると、過学習の度合いが減るが、対策がまだ不十分である。

・Ｌ１正則化項を追加すると、過学習の度合いが減るが、予測精度が安定しない。

・weight\_decay\_lambdaを大きくすると過学習を抑えることができるが、大きすぎる

と学習不足になる。weight\_decay\_lambdaを小さくしていくと正則化の効果がなく

なっていく。

### [**DN35\_Jupyter演習(im2col)**](https://vimeo.com/showcase/5577593/video/302843178)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/2_6_simple_convolution_network-ensyu.pdf>

■考察

・im2colを使って画像の多次元配列を2次元配列に変換し、行列計算できるようして

いる。

　・im2colで画像の多次元配列を2次元配列にし、col2imで多次元配列に変換しても、

元の配列にもどるというわけではない。

[**DN37\_Jupyter演習(3)**](https://vimeo.com/showcase/5577593/video/302805300)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/2_6_simple_convolution_network-ensyu.pdf>

■考察

・CNNで学習する際、計算が重い場合、GPUを使うか、ハイスペックのPCを使うこと

　で、計算が早くなる。

・optimizerについて、CNNにおいてもSGDよりAdamの方が学習速度がはやい。