慶應義塾大学理工学部

機械学習基礎

第６回　誤差逆伝播法

情報工学科　教授　杉浦孔明
komei.sugiura@keio.jp

本講義の到達目標と今回の授業の狙い

本講義の到達目標

- DNNの基礎理論と実装の関係を理解する
- 種々のDNNをコーディングできる

今回の授業の狙い

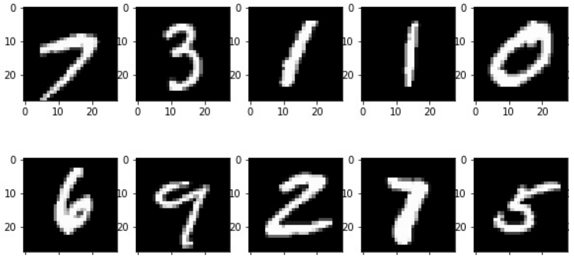
- 誤差逆伝播法を習得する

- 出席確認： K-LMS上の機械学習基礎のMainページへアクセス

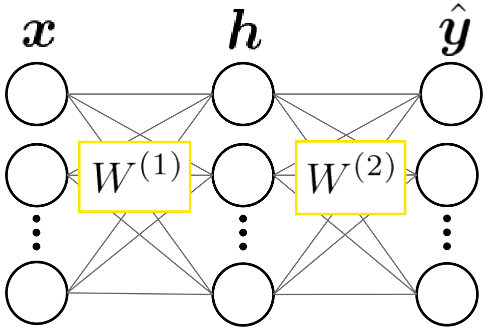
これまでの復習

機械学習の主要要素：
データ・モデル・目的関数を定めたうえでの最適化問題

学習に使用される**データ**



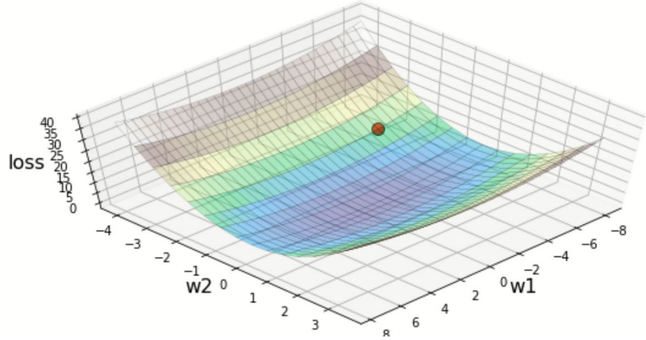
ニューラルネット等の**モデル**



モデルの良さを定量化する**目的関数**

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \log P(\hat{y}_{nk})$$

目的関数を最大化/最小化するために、モデルのパラメータを調整する**最適化**



【第３回】用語定義

- **訓練集合**（training set）
  $X_{\text{train}} = \{(x_n, y_n) | n = 1, ..., 1000\}$
  1個分を訓練サンプルと呼ぶ

- X_{train} から写像 $f(x)$ を求める
  ＝**訓練**（training）または**学習**（learning）

- **テスト集合**（test set）
  $X_{\text{test}} = \{(x_n, y_n) | n = 1001, 1002\}$
  を用いて真値と予測値の誤差を評価

$$E(w, b) = \frac{1}{2} \sum_{n=1}^{1000} (y_n - \hat{y}_n)^2$$

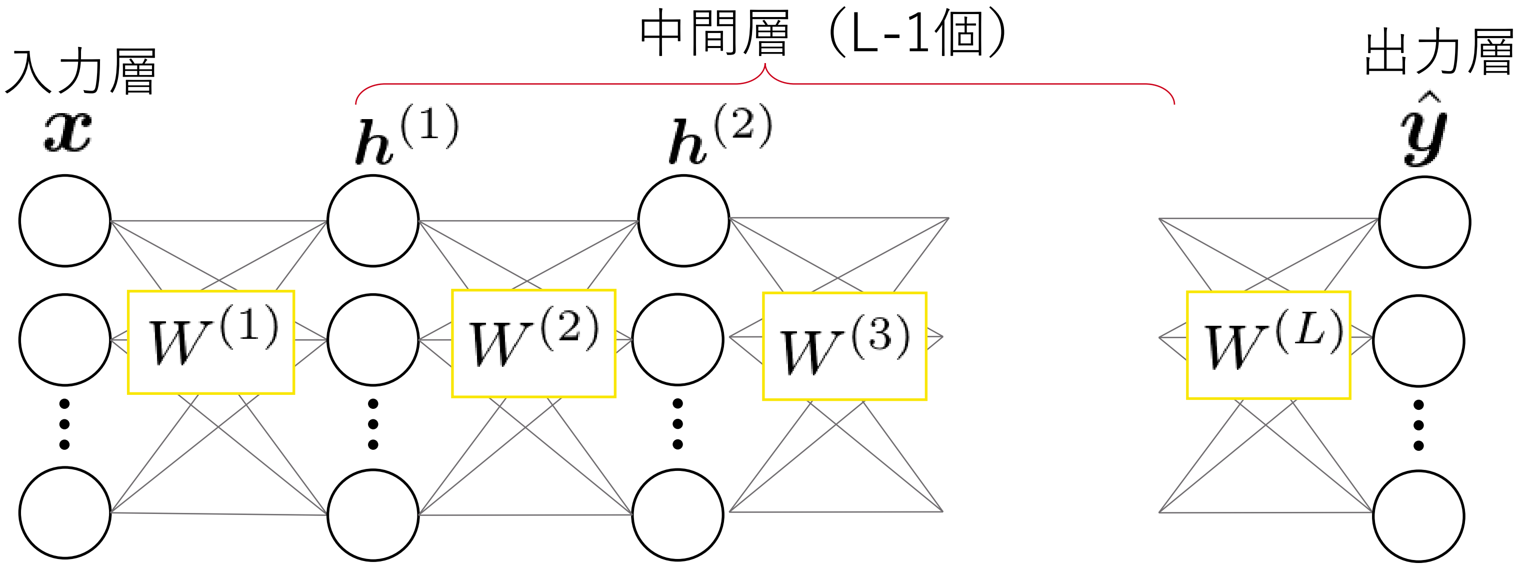
$$= \frac{1}{2} \sum_{n=1}^{1000} (y_n - w x_n - b)^2$$

損失関数（loss function）　or
誤差関数（error function）　or
コスト関数（cost function）　or
目的関数（objective function）
と呼ばれる

【第４回】順伝播型ニューラルネット

■ **順伝播型ニューラルネット**（feed-forward neural network; FFNN）

$$\boldsymbol{h}^{(1)} = f^{(1)}(W^{(1)}\boldsymbol{x})$$
$$\boldsymbol{h}^{(2)} = f^{(2)}(W^{(2)}\boldsymbol{h}^{(1)}) \xrightarrow{\text{一般化すると}} \boldsymbol{h}^{(l)} = f^{(l)}(W^{(l)}\boldsymbol{h}^{(l-1)})$$
$$\hat{\boldsymbol{y}} = f^{(L)}(W^{(L)}\boldsymbol{h}^{(L-1)})$$



【第５回】 Adam [Kingma+ 2014]の更新則

■ **Adam**

$$(m_{i,0}, v_{i,0}) = (0, 0)$$

$$m_{i,t} = \rho_1 m_{i,t-1} + (1-\rho_1)\nabla E(\boldsymbol{w}^{(t)})_i$$

$$v_{i,t} = \rho_2 v_{i,t-1} + (1-\rho_2)(\nabla E(\boldsymbol{w}^{(t)})_i)^2$$

■ 勾配の指数移動平均を使う点がAdaDeltaと異なる

■ Adamの更新則

真の値より0側に偏るので以下のように補正

$$\hat{m}_{i,t} = \frac{m_{i,t}}{1-(\rho_1)^t} \qquad \hat{v}_{i,t} = \frac{v_{i,t}}{1-(\rho_2)^t}$$

↑ ρ1のt乗

$$\Delta \boldsymbol{w}_i^{(t)} = -\eta \frac{\hat{m}_{i,t}}{\sqrt{\hat{v}_{i,t} + \epsilon}}$$

【第５回】バッチ正規化：ユニットが１つの場合

①活性値uを**標準化**（＝平均０、分散１になるように変換）

ミニバッチ内のサンプルに対するuの平均

$$\hat{u} = \frac{u - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

ミニバッチ内のサンプルに対するuの分散

ゼロ除算を避けるための微小な正数

②活性値uに対するバッチ正規化の定義

$$f_{\mathrm{BN}}(u) = \underline{\gamma}\hat{u} + \underline{\beta}$$

学習パラメータ

誤差逆伝播法

連鎖律（chain rule）の復習

- $y = \log(\cos x)$ を x について微分せよ

連鎖律（chain rule）の復習

- $y = \log(\cos x)$ を x について微分せよ

$$\begin{aligned} y &= \log u \\ u &= \cos x \end{aligned} \quad \rightarrow \quad \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{1}{u}(-\sin x)$$

高校の復習

$$\{f(g(x))\}' = f'(g(x))g'(x)$$

$$\frac{dy}{dx} = \frac{1}{\cos x}(-\sin x)$$

連鎖律（chain rule）の復習

- $y = \log(\cos x)$ を x について微分せよ

$$y = \log u \qquad u = \cos x \quad \rightarrow \quad \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{1}{u}(-\sin x)$$

高校の復習

$$\{f(g(x))\}' = f'(g(x))g'(x)$$

$$= \frac{dy}{dx} = \frac{1}{\cos x}(-\sin x)$$

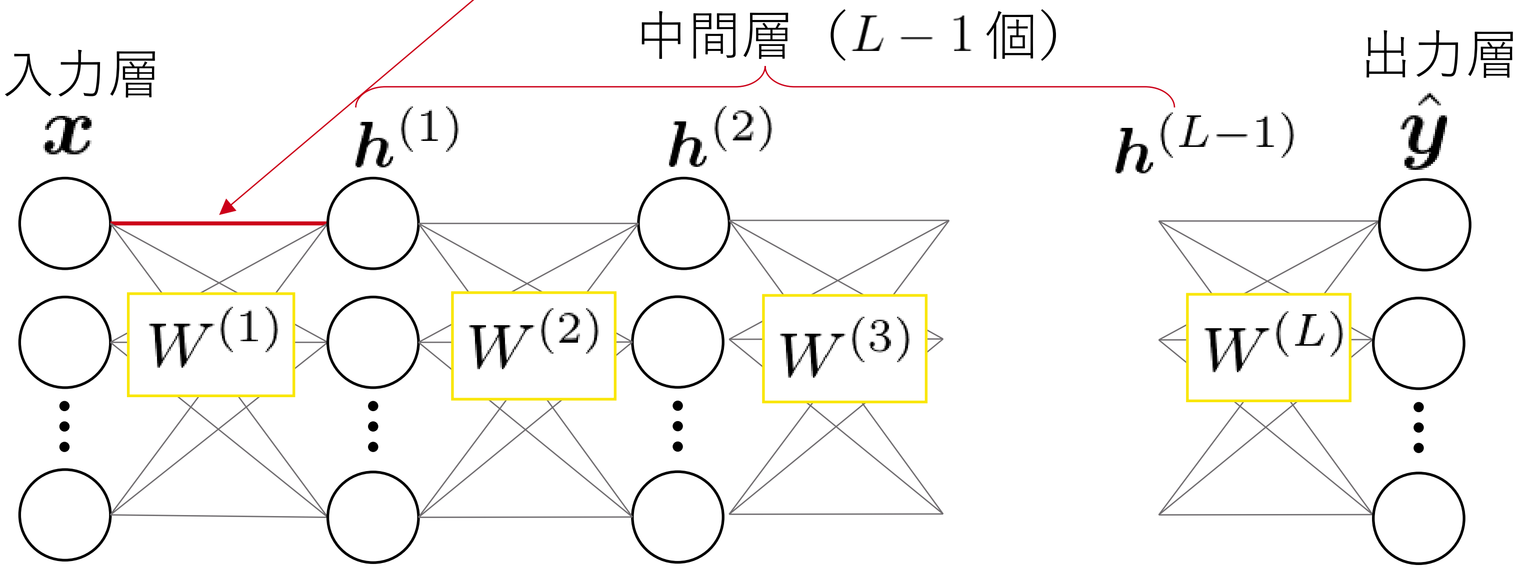
- 偏微分の場合の連鎖律の例

$$u = f(x) \qquad v = g(x) \qquad z = h(u, v) \quad \rightarrow \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial z}{\partial v}\frac{\partial v}{\partial x}$$

誤差逆伝播（backpropagation）法の背景

- 巨大な合成関数を全パラメータ（例：$w_{11}^{(1)}$）について安直に偏微分するのは計算量・精度面で問題☹
  →**誤差逆伝播法**で効率的に計算☺

$$\hat{\boldsymbol{y}} = f^{(L)}(W^{(L)}\boldsymbol{h}^{(L-1)}) = f^{(L)}(f^{(L-1)}(\cdots f^{(1)}(W^{(1)}\boldsymbol{x})))$$

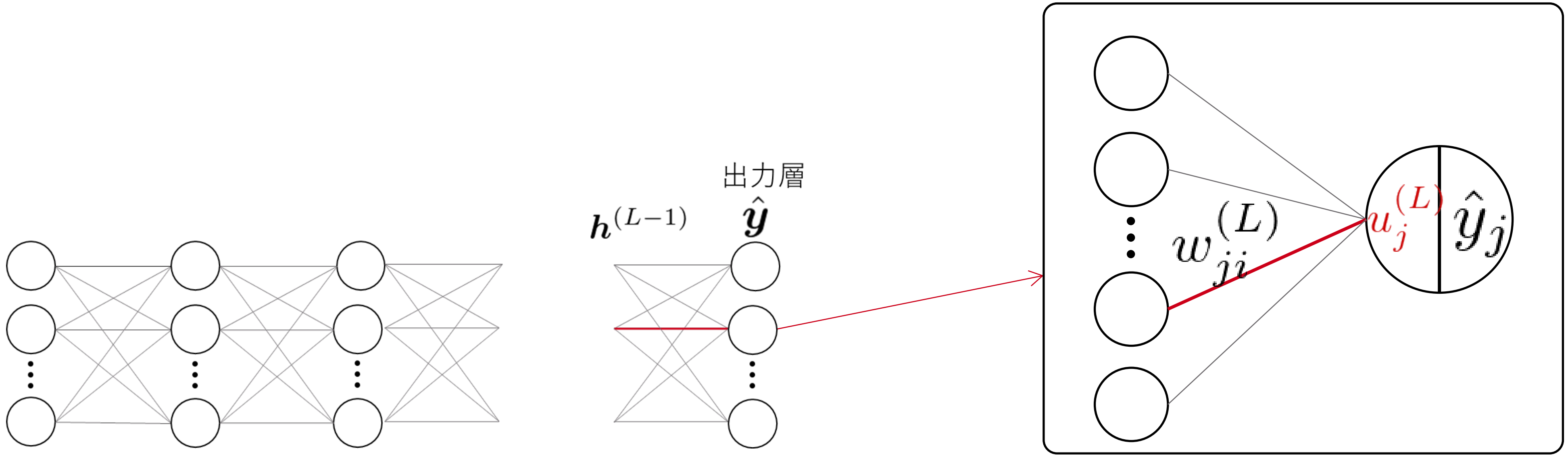


出力層への重みに関する偏微分を求めよう

- 出力層への重みに関する偏微分は、連鎖律より以下で求まる

$$\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}}$$

jが繋がり先、iが繋がる元



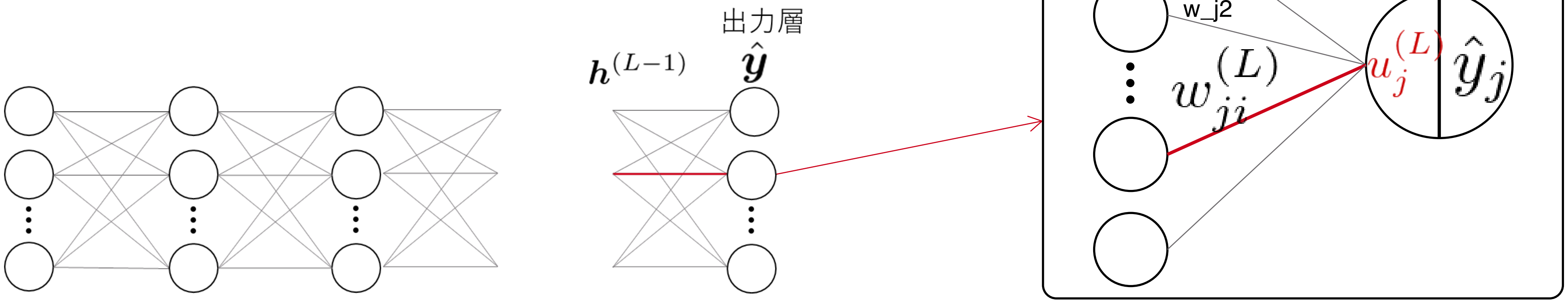
出力層への重みに関する偏微分を求めよう

- 出力層への重みに関する偏微分は、連鎖律より以下で求まる

$$
\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}}
$$

$$
= \delta_j^{(L)} \frac{\partial}{\partial w_{ji}^{(L)}} \{ w_{j1}^{(L)} h_1^{(L-1)} + \cdots + w_{ji}^{(L)} h_i^{(L-1)} + \cdots + w_{jI}^{(L)} h_I^{(L-1)} \}
$$

デルタと定義



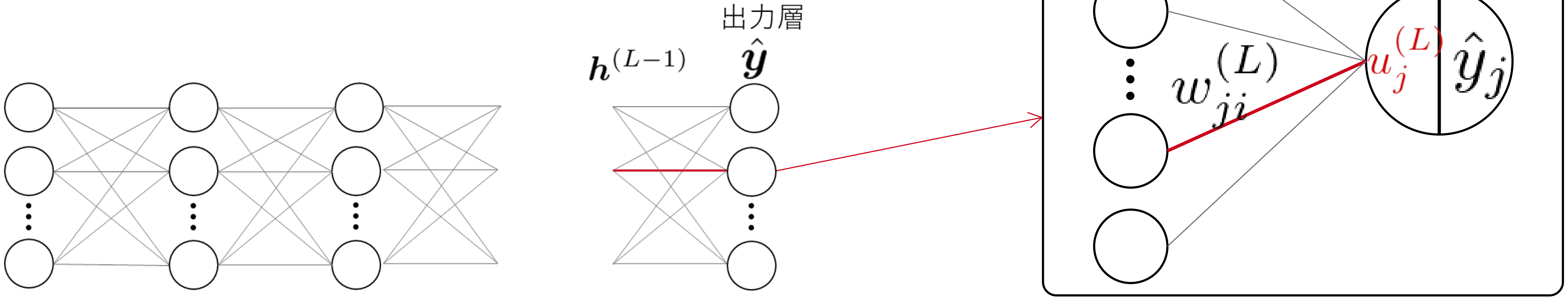
出力層への重みに関する偏微分を求めよう

- 出力層への重みに関する偏微分は、連鎖律より以下で求まる

$$
\begin{aligned}
\frac{\partial E}{\partial w_{ji}^{(L)}} &= \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}} \\
&= \delta_j^{(L)} \frac{\partial}{\partial w_{ji}^{(L)}} \{ \cancel{w_{j1}^{(L)} h_1^{(L-1)}} + \cancel{\cdots} + w_{ji}^{(L)} h_i^{(L-1)} + \cancel{\cdots} + \cancel{w_{jI}^{(L)} h_I^{(L-1)}} \} \\
&= \delta_j^{(L)} h_i^{(L-1)}
\end{aligned}
$$

デルタと定義

$\partial w_{ji}^{(L)}$には関係ないため



誤差逆伝播法：デルタを求める

■ 出力層への重みに関する偏微分

$$\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}}$$

$$= \delta_j^{(L)} \underbrace{h_i^{(L-1)}}_{\text{順伝播時に計算済}}$$

誤差逆伝播法：デルタを求める

- 出力層への重みに関する偏微分

$$\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}} = \delta_j^{(L)} \underline{h_i^{(L-1)}}$$

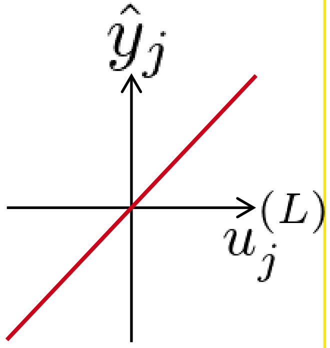
順伝播時に計算済

- 例：簡単のため、回帰問題かつ1サンプル分の誤差を考える

$$E = \frac{1}{2}||\hat{\boldsymbol{y}} - \boldsymbol{y}||^2$$

$$\delta_j^{(L)} \triangleq \frac{\partial E}{\partial u_j^{(L)}} = \frac{\partial E}{\partial \hat{y}_j} \boxed{\frac{\partial \hat{y}_j}{\partial u_j^{(L)}}} = \hat{y}_j - y_j$$

活性化関数が恒等写像なら１



誤差逆伝播法：デルタを求める

- 出力層への重みに関する偏微分

$$\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}}$$

$$= \underline{\delta_j^{(L)}} h_i^{(L-1)}$$

同様に考えれば、

- $\delta_j^{(L-1)}, \delta_j^{(L-2)}, \cdots$ が求まれば、

$$\frac{\partial E}{\partial w_{ji}^{(L-1)}}, \cdots, \frac{\partial E}{\partial w_{ji}^{(1)}}$$ が得られる

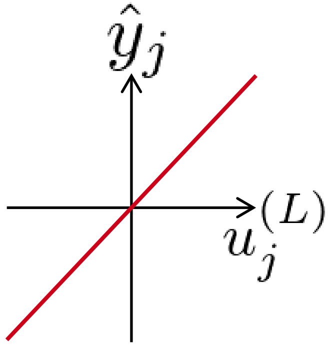
- 例：簡単のため、回帰問題かつ1サンプル分の誤差を考える

$$E = \frac{1}{2} ||\hat{\boldsymbol{y}} - \boldsymbol{y}||^2$$

$$\delta_j^{(L)} \triangleq \frac{\partial E}{\partial u_j^{(L)}}$$

$$= \frac{\partial E}{\partial \hat{y}_j} \boxed{\frac{\partial \hat{y}_j}{\partial u_j^{(L)}}}$$

$$= \hat{y}_j - y_j$$



活性化関数が恒等写像なら１

誤差逆伝播法：デルタの漸化式を求める

- 出力層への重みに関する偏微分

$$\frac{\partial E}{\partial w_{ji}^{(L)}} = \frac{\partial E}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ji}^{(L)}}$$

$$= \underline{\delta_j^{(L)}} h_i^{(L-1)}$$

同様に考えれば、

- $\delta_j^{(L-1)}, \delta_j^{(L-2)}, \cdots$ が求まれば、

$$\frac{\partial E}{\partial w_{ji}^{(L-1)}}, \cdots, \frac{\partial E}{\partial w_{ji}^{(1)}}$$ が得られる

- デルタの漸化式を求めたい

$$\delta_j^{(L-1)} \triangleq \frac{\partial E}{\partial u_j^{(L-1)}}$$

$$= \sum_{k=1}^{K} \boxed{\frac{\partial E}{\partial u_k^{(L)}}} \frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}}$$

偏微分の連鎖律

L層のデルタ（計算済）

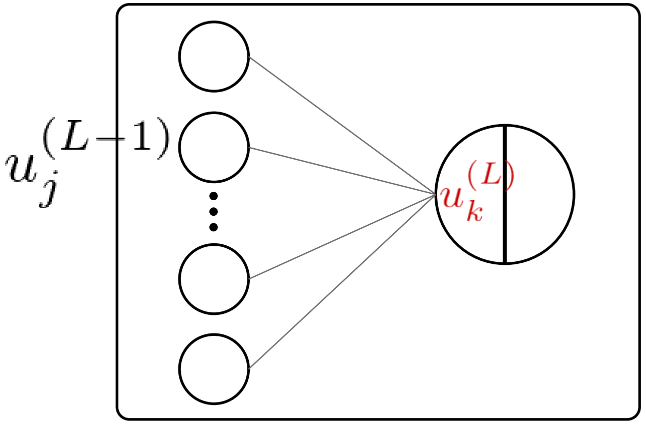
誤差逆伝播法：デルタの漸化式を求める

$$\frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}}$$

$$= \frac{\partial}{\partial u_j^{(L-1)}} \left[\sum_{j'} w_{kj'}^{(L)} f^{(L-1)}(u_{j'}^{(L-1)}) \right]$$

$$= w_{kj}^{(L)} f'^{(L-1)}(u_j^{(L-1)})$$

$f^{(L-1)}$の微分



- デルタの漸化式を求めたい

$$\delta_j^{(L-1)} \triangleq \frac{\partial E}{\partial u_j^{(L-1)}}$$

$$= \sum_{k=1}^{K} \frac{\partial E}{\partial u_k^{(L)}} \frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}}$$

偏微分の連鎖律

L層のデルタ
（計算済）

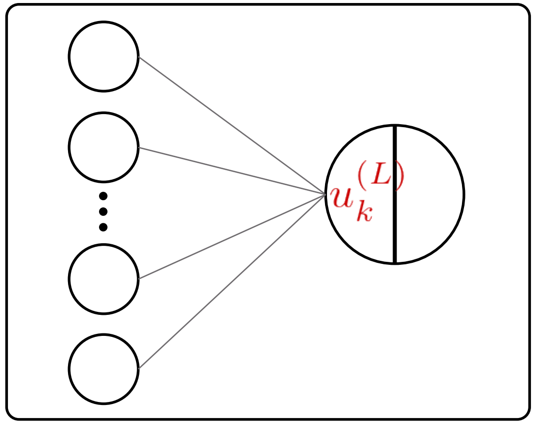
誤差逆伝播法：デルタの漸化式を求める

$$\frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}}$$

$$= \frac{\partial}{\partial u_j^{(L-1)}} \left[\sum_{j'} w_{kj'}^{(L)} f^{(L-1)}(u_{j'}^{(L-1)}) \right]$$

$$= w_{kj}^{(L)} f'^{(L-1)}(u_j^{(L-1)})$$

$f^{(L-1)}$の微分

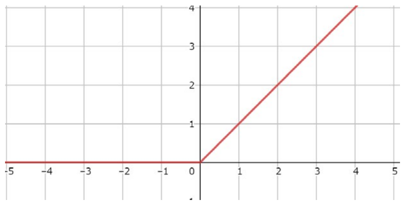


- デルタの漸化式が求まった

$$\delta_j^{(L-1)} \triangleq \frac{\partial E}{\partial u_j^{(L-1)}}$$

$$= \sum_{k=1}^{K} \frac{\partial E}{\partial u_k^{(L)}} \frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}}$$

$$= \sum_{k=1}^{K} \delta_j^{(L)} w_{kj}^{(L)} f'^{(L-1)}(u_j^{(L-1)})$$



ReLUなら0または1

誤差逆伝播法：
順伝播と逆伝播

- まとめると、

$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} h_i^{(l-1)}$$

- 順伝播

$$\boldsymbol{x} \to \boldsymbol{h}^{(1)} \to \dots \to \boldsymbol{h}^{(L-1)} \to \hat{\boldsymbol{y}}$$

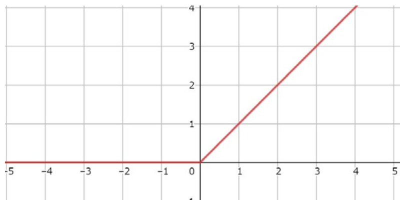
- 逆伝播

$$\boldsymbol{\delta}^{(1)} \leftarrow \dots \leftarrow \boldsymbol{\delta}^{(L-1)} \leftarrow \boldsymbol{\delta}^{(L)}$$

- デルタの漸化式が求まった

$$\begin{aligned}
\delta_j^{(L-1)} &\triangleq \frac{\partial E}{\partial u_j^{(L-1)}} \\
&= \sum_{k=1}^{K} \frac{\partial E}{\partial u_k^{(L)}} \frac{\partial u_k^{(L)}}{\partial u_j^{(L-1)}} \\
&= \sum_{k=1}^{K} \delta_j^{(L)} w_{kj}^{(L)} \underline{f'^{(L-1)}(u_j^{(L-1)})}
\end{aligned}$$

ReLUなら0または1



理解度確認

理解度確認
以下について周りと相談して１分以内に答えよ

1. ３層ニューラルネットトを構成する層は何と何と何と呼ばれるか？
2. 尤度とは何か？
3. ベルヌーイ分布の確率質量関数をμとxを使って表せ
4. Kクラス分類問題に対する交差エントロピー誤差関数をy_{nk}を使って表せ
5. 1-of-K表現とは何か？

理解度確認
以下について周りと相談して１分以内に答えよ

1. 最適化の定義
2. 汎化誤差の定義
3. 勾配降下法とモーメンタム法の違い
4. 指数移動平均の式

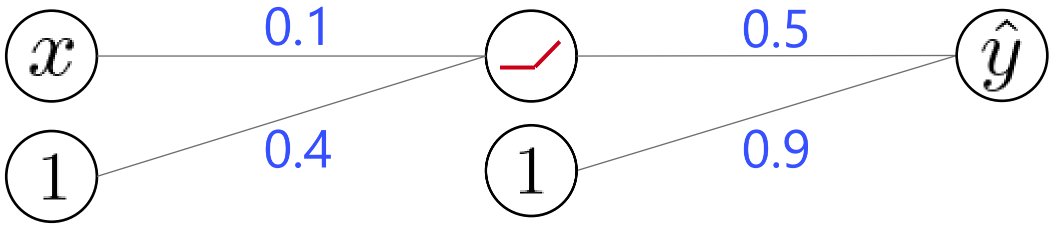
コードとの対応例（1）：インポートおよび構造定義

```python
import torch
import torch.nn as nn

net = nn.Sequential(
      nn.Linear(1, 1),    ←bias=True
      nn.ReLU(),
      nn.Linear(1, 1)
)
x = torch.tensor([[3.0]])
y = torch.tensor([[4.0]])
y_hat = net(x)  # 順伝播
E = nn.MSELoss()(y_hat, y) # 平均二乗誤差

net.zero_grad() # 勾配をゼロにリセット
E.backward() # 逆伝播による勾配計算
```

■ ニューラルネットの構造



パラメータの初期値は
ランダムに設定される

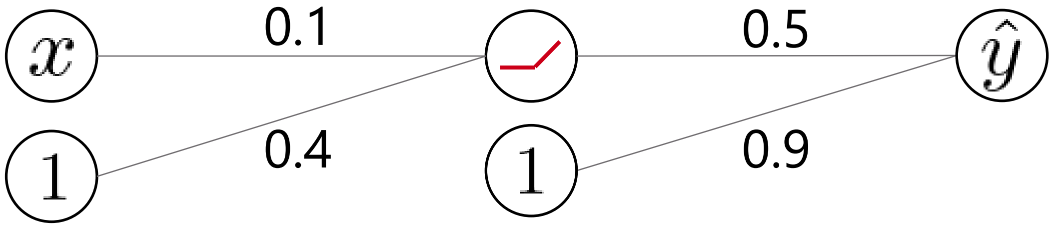
コードとの対応例（２）：順伝播

```python
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 1),
    nn.ReLU(),
    nn.Linear(1, 1)
)
x = torch.tensor([[3.0]])
y = torch.tensor([[4.0]])
y_hat = net(x)  # 順伝播
E = nn.MSELoss()(y_hat, y) # 平均二乗誤差

net.zero_grad() # 勾配をゼロにリセット
E.backward() # 逆伝播による勾配計算
```

- ニューラルネットの構造



- 訓練サンプル・順伝播・損失関数

$$(x, y) = (3, 4)$$

$$x \to \hat{y}$$

$$E = \frac{1}{2}||\hat{y} - y||^2$$

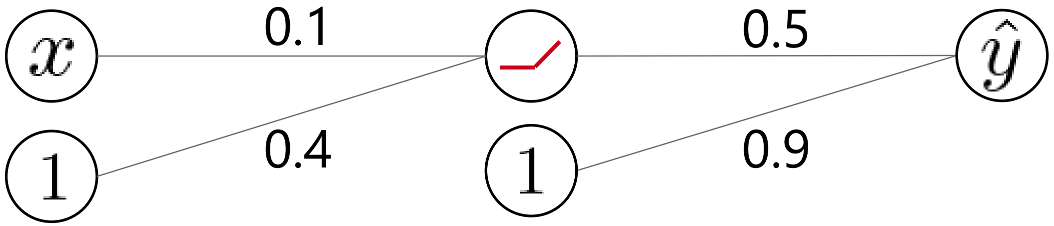
コードとの対応例（３）：逆伝播

```python
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 1),
    nn.ReLU(),
    nn.Linear(1, 1)
)
x = torch.tensor([[3.0]])
y = torch.tensor([[4.0]])
y_hat = net(x)  # 順伝播
E = nn.MSELoss()(y_hat, y) # 平均二乗誤差

net.zero_grad() # 勾配をゼロにリセット
E.backward() # 逆伝播による勾配計算
```

- ニューラルネットの構造



- E.backward() 以前→以後

```
0.weight.data = tensor([[0.1]])
0.weight.grad = None
0.bias.data = tensor([0.4])
0.bias.grad = None
2.weight.data = tensor([[0.5]])
2.weight.grad = None
2.bias.data = tensor([0.9])
2.bias.grad = None
```

→

```
0.weight.data = tensor([[0.1]])
0.weight.grad = tensor([[-8.25]])
0.bias.data = tensor([[0.4]])
0.bias.grad = tensor([[-2.75]])
2.weight.data = tensor([[0.5]])
2.weight.grad = tensor([[-3.85]])
2.bias.data = tensor([[0.9]])
2.bias.grad = tensor([[-5.5]])
```

勾配消失・勾配爆発

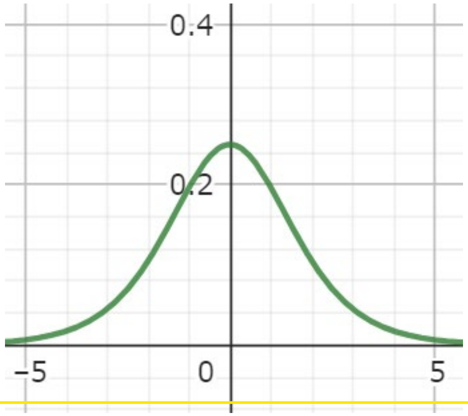
- **勾配消失問題**（vanishing gradient problem）
  - 逆伝播計算では出力から入力まで何度も線形変換を行うため、勾配が急速に小さくなることがある
  - 逆に勾配が発散する場合を**勾配爆発問題**と呼ぶ
- 1980-2000年代まで、ニューラルネットの深層化を阻む問題であった

勾配消失・勾配爆発の緩和

- **勾配消失問題**（vanishing gradient problem）
  - 逆伝播計算では出力から入力まで何度も線形変換を行うため、勾配が急速に小さくなることがある
  - 逆に勾配が発散する場合を**勾配爆発問題**と呼ぶ
- 1980-2000年代まで、ニューラルネットの深層化を阻む問題であった

- 現代では種々の技術により緩和
  - **バッチ正規化**
  - **残差接続**
  - **ReLU**（微分が１）
    - ⇔☹シグモイド関数の場合、最大0.25 → 0.0625 → 0.015.. → 0.0039.. → …

シグモイド関数の微分



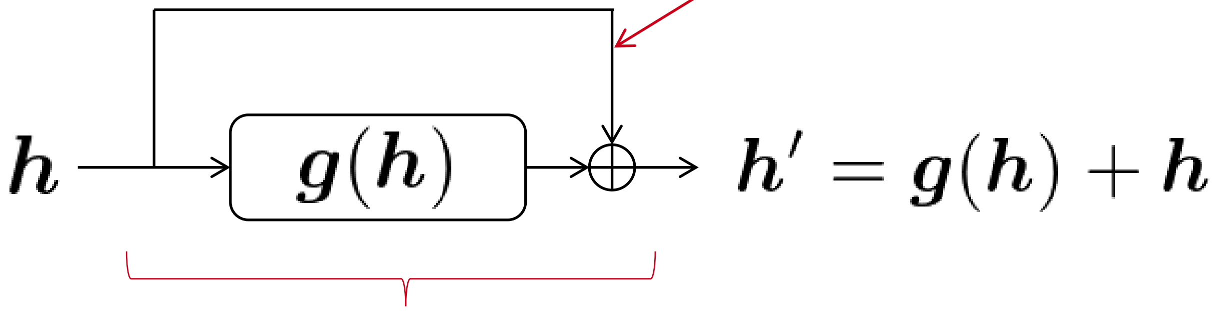
★残差接続（residual connection）[He+ 2016]

- 層を迂回する近道を設ける接続方法
- 効果：　深層化を可能にした

　　　↑迂回された層が不要なら、ゼロになるよう学習されれば良い

- 迂回された層の役割：　残差 $\boldsymbol{g}(\boldsymbol{h}) = \boldsymbol{h}' - \boldsymbol{h}$ の予測

スキップ接続（skip connection）



残差ブロック（residual block）

本講義全体の参考図書

- ★機械学習スタートアップシリーズ これならわかる深層学習入門　瀧雅人著　講談社（本講義では、異なる表記を用いることがあるので注意）
- ★Dive into Deep Learning (https://d2l.ai/)
- 深層学習 改訂第2版 (機械学習プロフェッショナルシリーズ)　岡谷貴之著　講談社
- ディープラーニングを支える技術　岡野原大輔著　技術評論社
- 画像認識 (機械学習プロフェッショナルシリーズ)　原田達也著　講談社
- 深層学習による自然言語処理 (機械学習プロフェッショナルシリーズ)　坪井祐太、海野裕也、鈴木潤 著、講談社
- 東京大学工学教程 情報工学 機械学習　中川 裕志著、東京大学工学教程編纂委員会編　丸善出版
- パターン認識と機械学習 上・下　C.M. ビショップ著　丸善出版

参考文献

1. Sietsma, J., & Dow, R. J. (1991). Creating artificial neural networks that generalize. Neural networks, 4(1), 67-79.
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.
3. Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

TA発表

実装課題①

実装課題①

- **提出方法・仕様：　以下のリンク（keio.jp認証あり）に記載**
https://forms.gle/9uW4tuRTyQ5b6ikg6

- **提出期限：**　2025/11/24 23:59

- Kaggleの使い方がわからない場合はLLMに問い合わせましょう

実習

実習

実習の目的

- コーディングと基礎理論の関係を学ぶ

実習課題の場所

- K-LMSから辿る

実習に関する質問

- ChatGPTに説明させる
- 教科書で調べる・検索・周囲と相談（私語禁止ではありません）
- 上記で解消しなければ挙手→TAが対応

付録

DNNの学習におけるGPUの利用

- 背景
  - CPUによる計算ではDNNの学習に時間がかかりすぎる
- GPUで学習を行うには、入力x、ラベルy、モデルの３点をGPUに送る
  - x = x.to(device)
  - y = y.to(device)
  - model = MyMnistNet().to(device)

DNNの学習におけるGPUの利用

- 背景
  - CPUによる計算ではDNNの学習に時間がかかりすぎる
- GPUで学習を行うには、入力x、ラベルy、モデルの３点をGPUに送る
  - x = x.to(device)
  - y = y.to(device)
  - model = MyMnistNet().to(device)

- device = torch.device("cuda")
  - torch.device("cuda")
    - 複数あるGPUのうちデフォルトのGPUが使用される
  - torch.device("cpu")
    - CPUが使用される