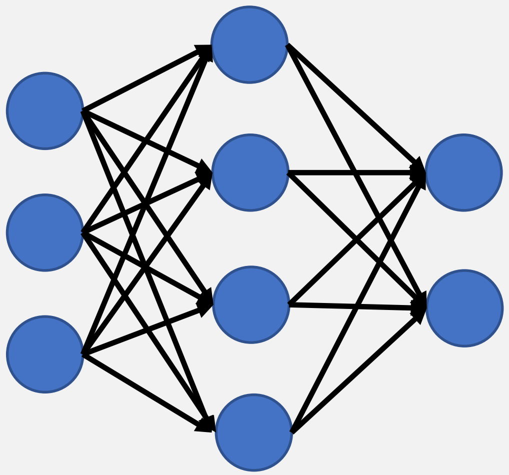
セミナー
ニューラルネットワーク

M1　町田 龍昭

ニューラルネットワーク（多層パーセプトロン）

- 生物の神経から着想を得た構造の関数
- 非線形関数
- 様々な種類がある
  Ex.) CNN, RNN, …
  - 最もわかりやすい多層パーセプトロンで説明する



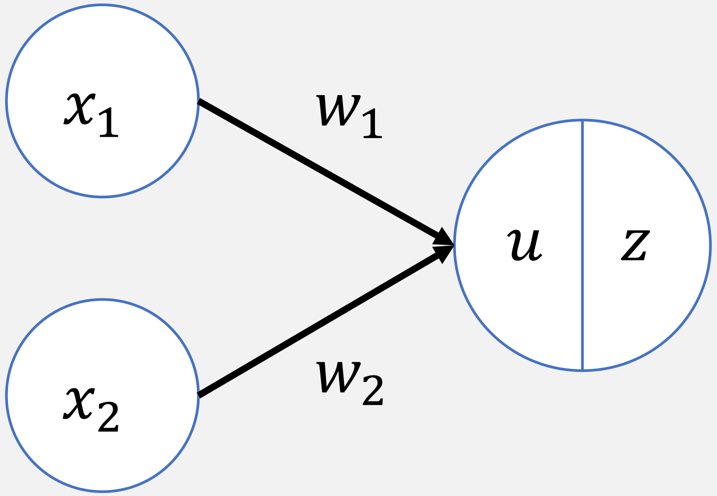
入力層　中間層　出力層

：ニューロン　前層から受け取る値を元に何らかの値を出力する

：結合　左側のニューロンの出力に係数をかけて右側のニューロンに入力

ニューラルネットワーク

- 計算方法



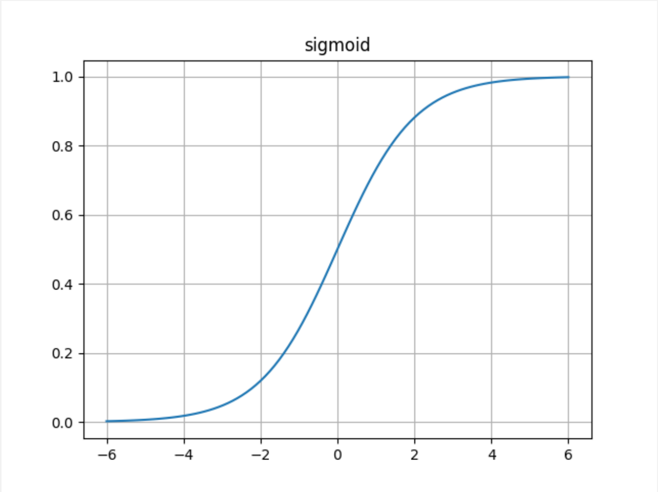
x_1, x_2を入力としてzを出力する

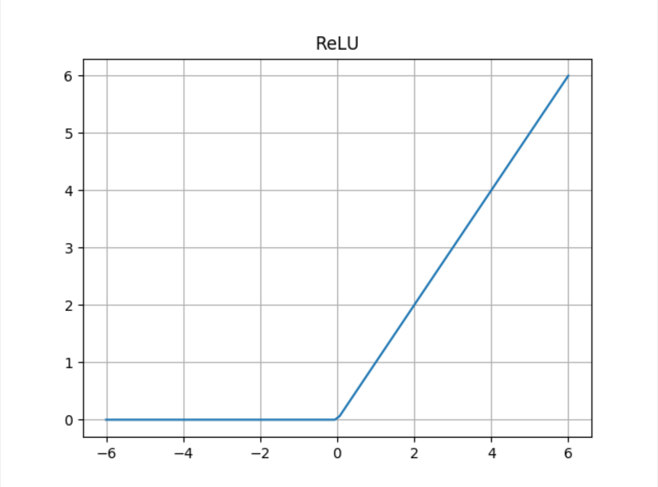
$$u = w_1 x_1 + w_2 x_2 (= Wx)$$

$$z = h(u)$$

前層の線形結合に非線形関数をかけたものが出力となる.

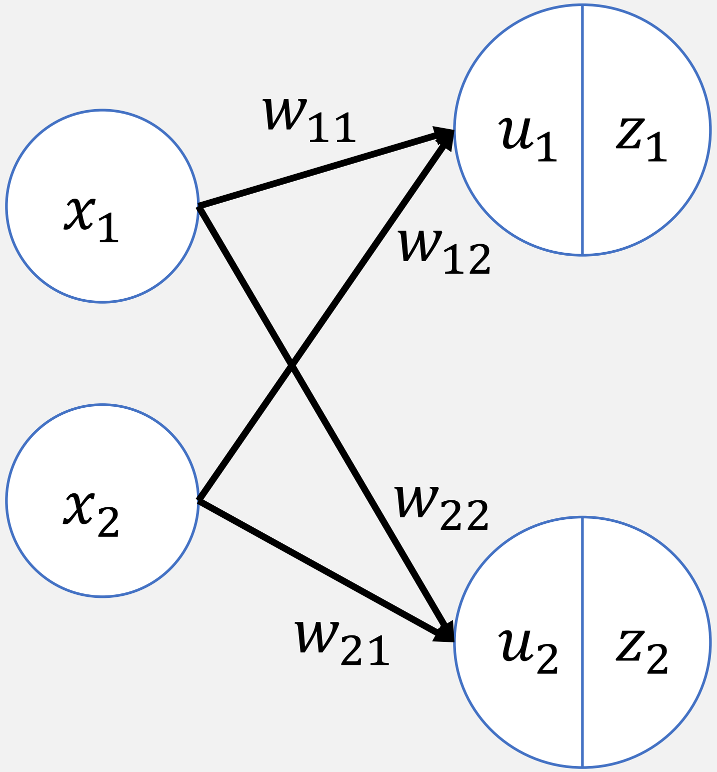
hは非線形関数で，sigmoid，tanh，ReLUなどが使われる





ニューラルネットワーク

- 複数の出力はそれぞれ独立に計算される



$$z_1 = h(w_{11}x_1 + w_{12}x_2)$$
$$z_2 = h(w_{21}x_1 + w_{22}x_2)$$

$$\boldsymbol{x} = [x_1 \ \ x_2]^T, \boldsymbol{u} = [u_1 \ \ u_2]^T$$
$$\boldsymbol{z} = [z_1 \ \ z_2]^T W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

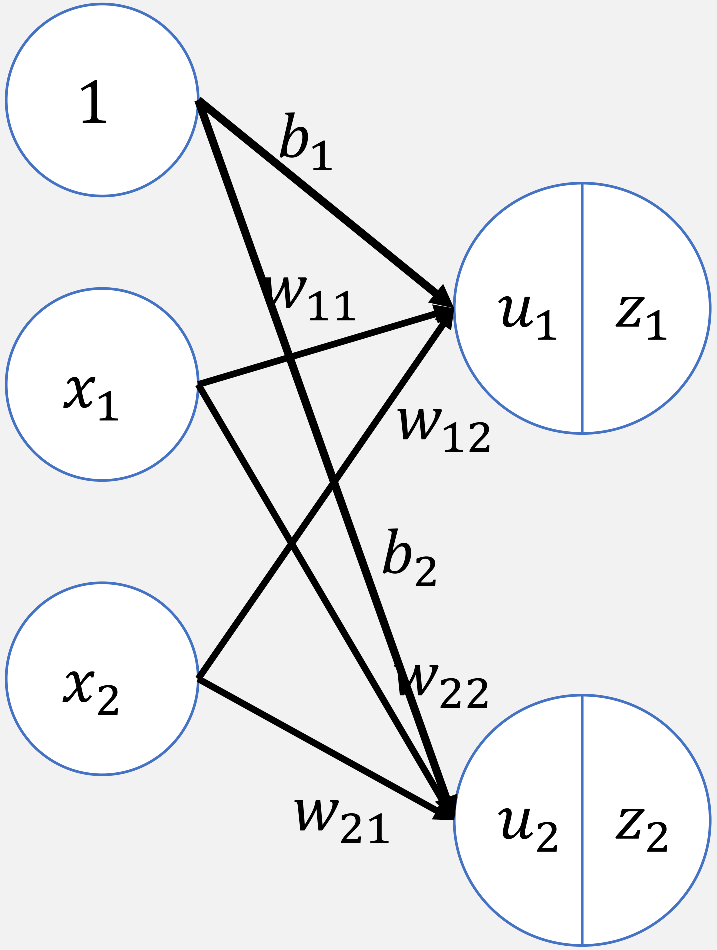
と表すと,

$$\boldsymbol{z} = h(W\boldsymbol{x})$$

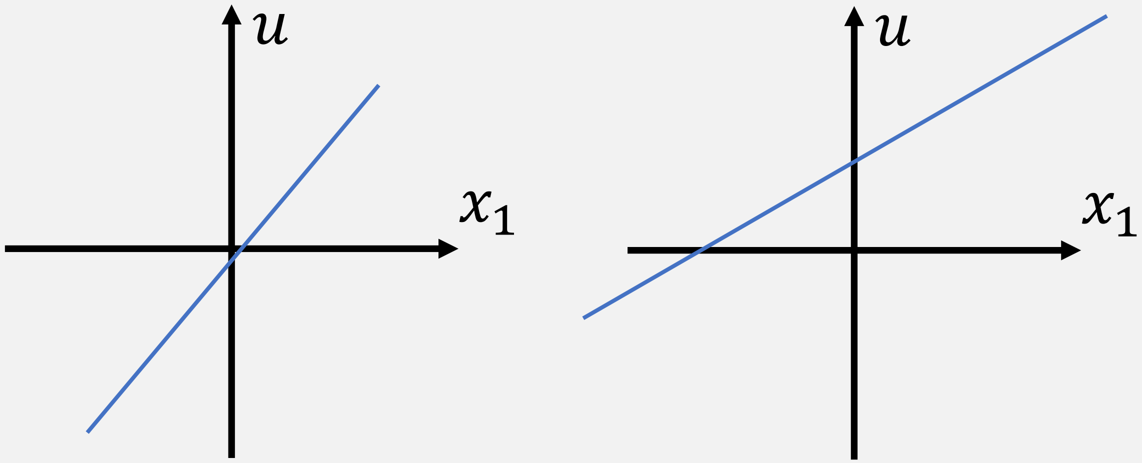
と書ける.

ニューラルネットワーク

- 前層の線形結合に定数（バイアス）を加えることが多い



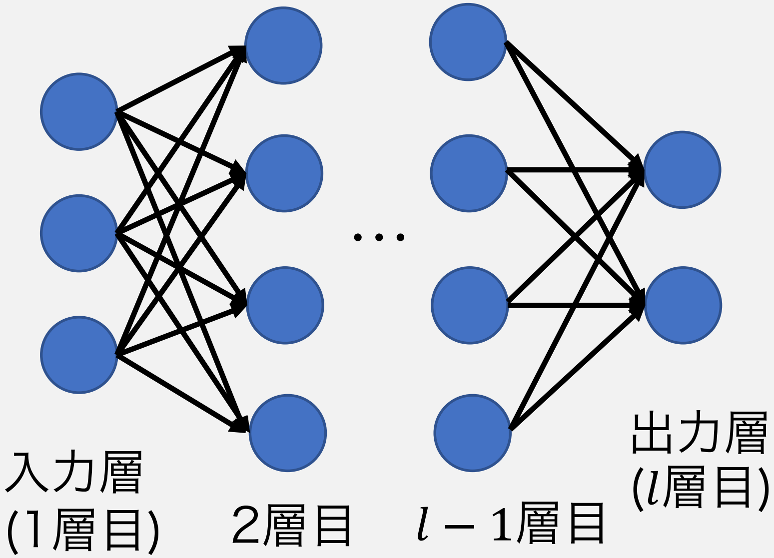
$$\boldsymbol{z} = h(W\boldsymbol{x} + \boldsymbol{b})$$



なぜ非線形関数をかけるのか？

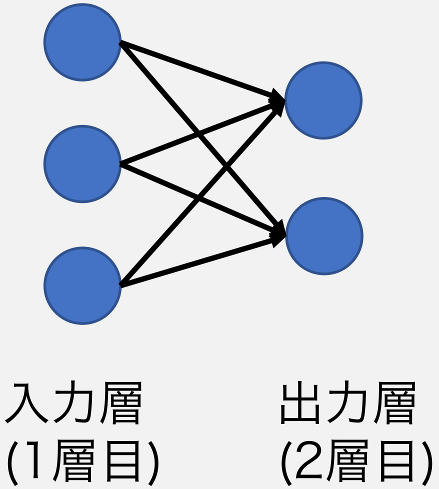
- 広い表現力を持たせるため
  - もし非線形関数がなければ，いくら層を重ねても2層と同じ！

$$\boldsymbol{z} = W^{(l)}W^{(l-1)} \dots W^{(2)}\boldsymbol{x}$$



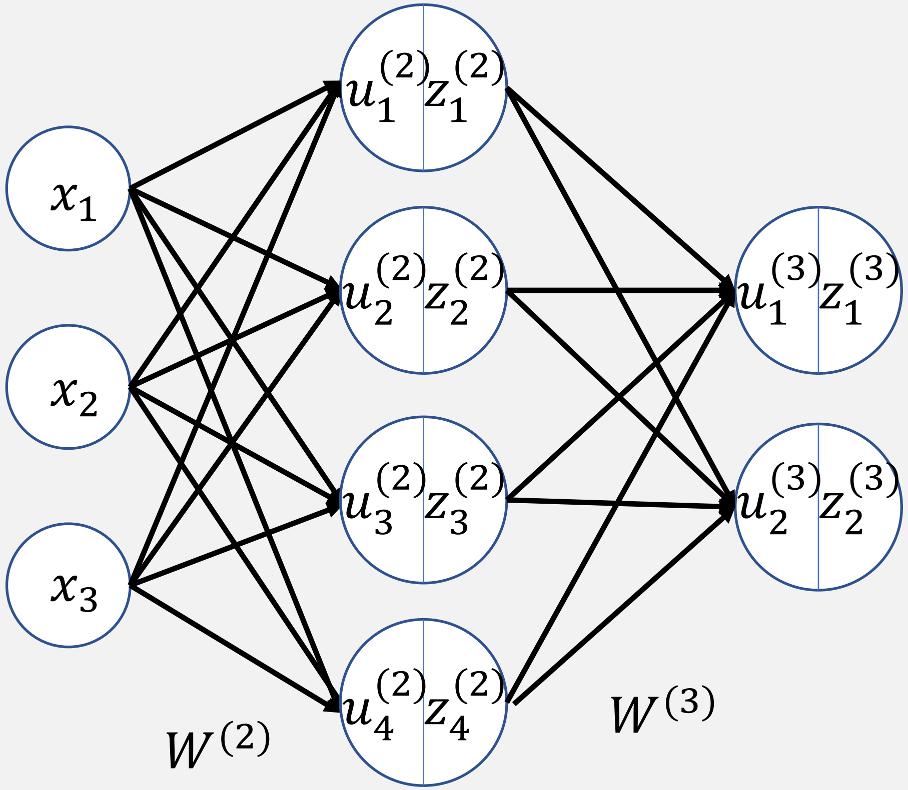
$$=$$

$$\boldsymbol{z} = \widetilde{W}\boldsymbol{x}$$
$$(\widetilde{W} = W^{(l)}W^{(l-1)} \dots W^{(2)})$$



NNでの多クラス分類

- 出力層のニューロンの中で最大値を示したニューロンに対応したクラスを分類結果と考える



りんごの確率

バナナの確率

Ex.)

$$\boldsymbol{z}^{(3)} = \begin{pmatrix} 0.95 \\ 0.05 \end{pmatrix}$$

りんご

softmax関数

- 多クラス分類では，最終層の活性化関数にsoftmax関数というものを使う
- 各出力の総和が1になるので，確率のようなものとみなせる

$$softmax(u_k) = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)} \quad (\boldsymbol{u} \in \mathbb{R}^K)$$

Ex.) $\boldsymbol{u} = [1, 2, 3]^T$ のとき，

$$softmax(\boldsymbol{u}) = [0.090 \quad 0.245 \quad 0.665]^T$$

➡ 総和を取ると1に！

多クラス分類での損失関数

- 多クラス分類では交差エントロピー (cross entropy)という損失関数を使う
- 2つの確率分布間の「近さ」を表す
  - [0, 1]の範囲．小さければ小さいほど近い
- softmax関数と相性がいい（教科書参照）

$$E(\boldsymbol{d}, \boldsymbol{y}; \boldsymbol{w}) = -\sum_{k=1}^{K} d_k \log y_k(\boldsymbol{x}; \boldsymbol{w})$$

K: クラス数

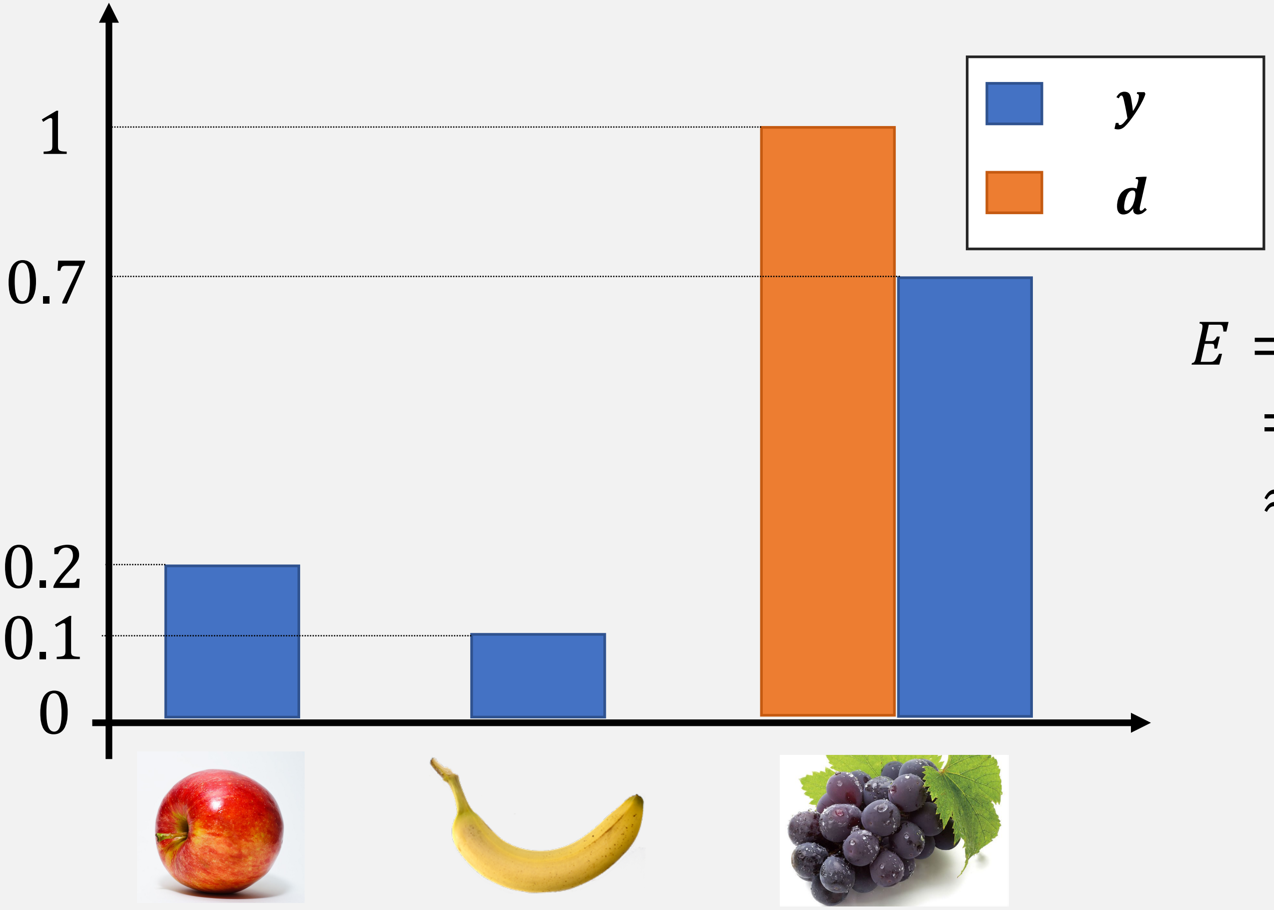
$\boldsymbol{d} = [0, \dots, 0,1,0, \dots, 0]^T$: 正解ラベル

$\boldsymbol{y} = \boldsymbol{z}^{(L)}$: 最終層の出力

$\boldsymbol{w} = \{W^{(2)}, \dots, W^{(L)}, \boldsymbol{b}^{(2)}, \dots, \boldsymbol{b}^{(L)}\}$: ネットワークの全パラメータ

多クラス分類での損失関数

Ex.) りんご，バナナ，ぶどう識別器



$$E = -(0 \times \log 0.1 + 0 \times \log 0.2 + 1 \times \log 0.7)$$
$$= -\log 0.7$$
$$\approx 0.357$$

$$E(\boldsymbol{w}) = -\sum_{k=1}^{K} d_k \log y_k(\boldsymbol{x}; \boldsymbol{w})$$

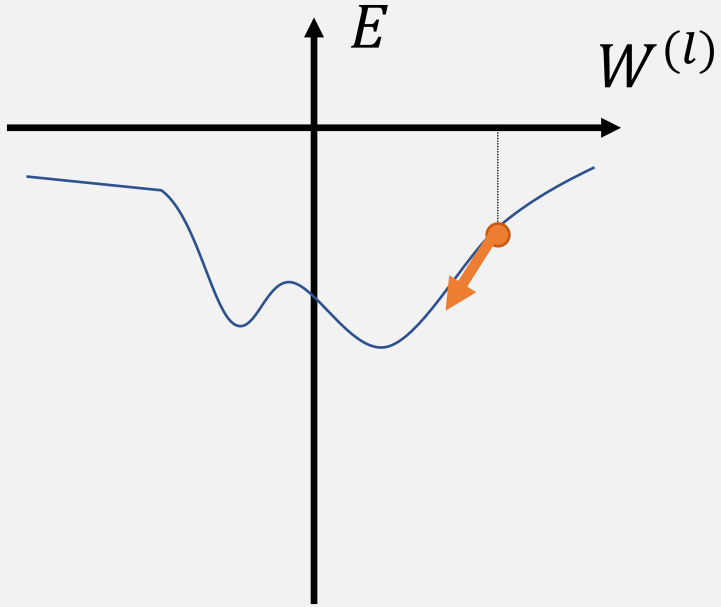
学習方法

- SGDで学習する
  - 損失関数に対してパラメータの勾配を求め，損失関数の値を小さくする方に更新
  - 最適な値が求まるとは限らないが，かなり良い局所解が求まることが知られている

$$W^{(l)} \leftarrow W^{(l)} - \alpha \frac{\partial E}{\partial W^{(l)}}$$

$$\boldsymbol{b}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \alpha \frac{\partial E}{\partial \boldsymbol{b}^{(l)}}$$

$$(l = L, L-1, \dots, 1)$$



勾配の求め方

- 単純に: 数値微分
  - $\frac{\partial E(\boldsymbol{w})}{\partial W^{(l)}} = \frac{E\left(W^{(2)},\ldots,W^{(l)}+\varepsilon,\ldots W^{(L)},\boldsymbol{b}^{(2)},\ldots,\boldsymbol{b}^{(L)}\right) - E\left(W^{(2)},\ldots,W^{(l)},\ldots W^{(L)},\boldsymbol{b}^{(2)},\ldots,\boldsymbol{b}^{(L)}\right)}{\varepsilon}$
  - $E = E\left(\boldsymbol{d}, \boldsymbol{y}\left(W^{(l)} \to W^{(l)} + \varepsilon\right)\right)$
  - $\boldsymbol{y}$
$$= softmax\left(W^{(L)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L)}\right)$$
$$= softmax\left(W^{(L)}h\left(W^{(L-1)}\boldsymbol{z}^{(L-2)} + \boldsymbol{b}^{(L-1)}\right) + \boldsymbol{b}^{(L)}\right)$$
$$= softmax\left(W^{(L)}h\left(W^{(L-1)}h(\boldsymbol{z}^{(L-3)}, \ldots h\left((W^{(l)}+\varepsilon)\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}\right) + \boldsymbol{b}^{(l+1)}\right) + \cdots) + \boldsymbol{b}^{(L)}\right)$$

勾配の求め方

- 単純に: 数値微分
  - $\frac{\partial E(\boldsymbol{w})}{\partial W^{(l)}} = \frac{E\left(W^{(2)},\ldots,W^{(l)} \cdots W^{(L)},\boldsymbol{b}^{(2)},\ldots,\boldsymbol{b}^{(L)}\right) - E\left(W^{(2)} \cdots \ldots W^{(L)},\boldsymbol{b}^{(2)},\ldots,\boldsymbol{b}^{(L)}\right)}{\varepsilon}$
  - $E = E(\boldsymbol{d}, \boldsymbol{y})$
  - $\boldsymbol{y}$

$$
\begin{aligned}
&= softmax\left(W^{(L)}\boldsymbol{z}^{(L-1)} \cdots\right) \\
&= softmax\left(W^{(L)}h\left(W^{(L-1)} \cdots\right) + \boldsymbol{b}^{(L-1)}\right) + \boldsymbol{b}^{(L)}\right)
\end{aligned}
$$

$= softmax\left(W^{(L)}h\left(W^{(L-1)} \cdots 3) \cdots W^{(l)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}\right) + \boldsymbol{b}^{(l+1)}\right) + \cdots) + \boldsymbol{b}^{(L)}\right)$

勾配の求め方

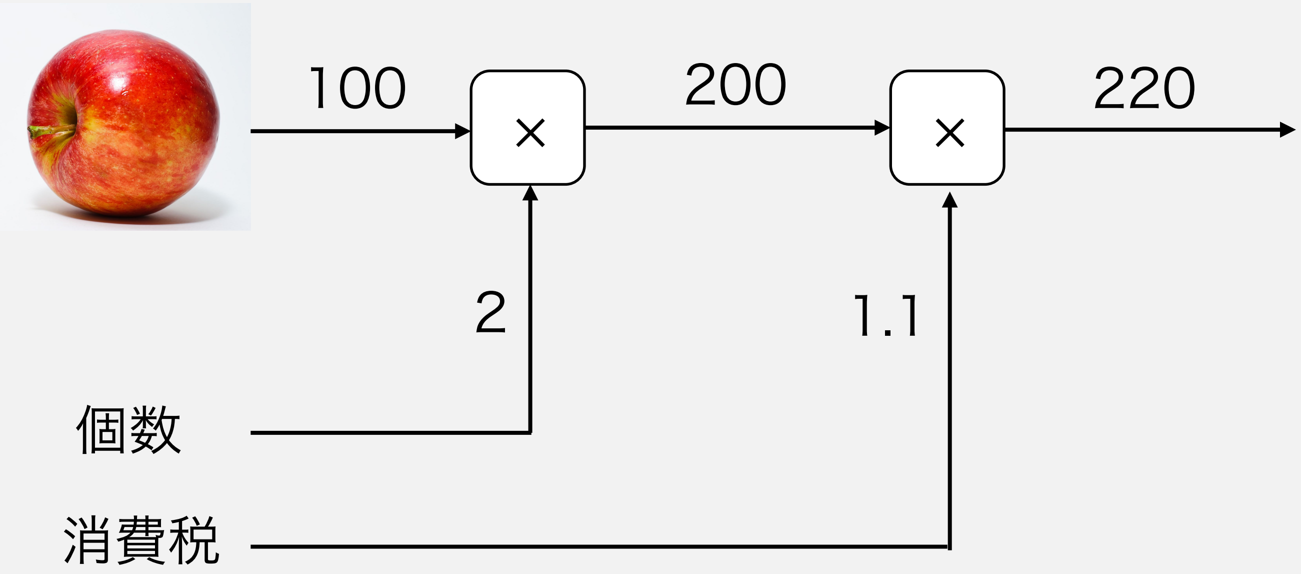
- 数値微分
  - 実装は単純になるが，計算量が現実的ではない

- 損失関数の値自体は計算している
- chain-ruleをうまく使って各層の微分を計算できる

誤差逆伝播法（バックプロパゲーション）

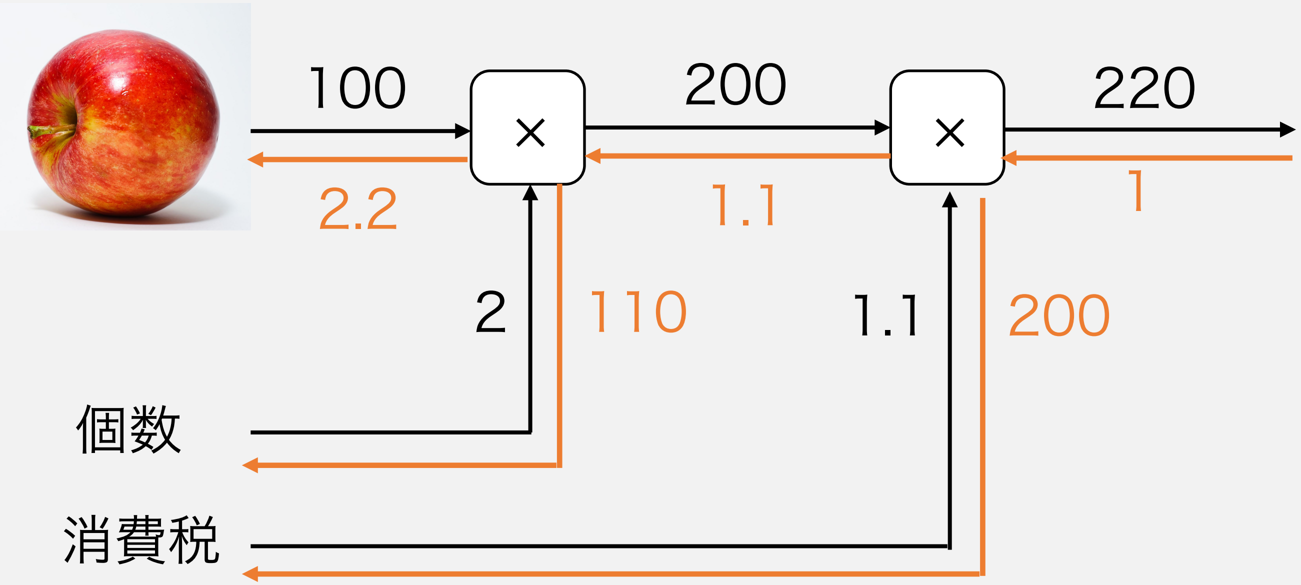
誤差逆伝播法

- 計算グラフ
  - □ が1つの演算操作を表す
- Ex.) りんごの値段の計算



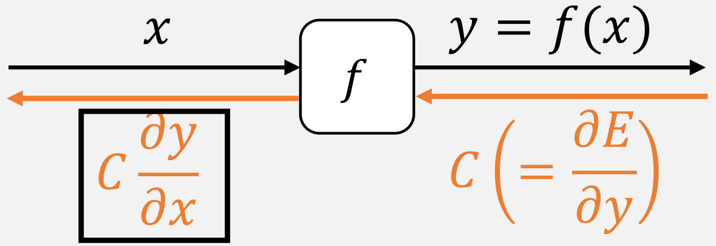
誤差逆伝播法

- 計算グラフ
  - □ が1つの演算操作を表す
- →を逆にたどることで，微分を求めることができる
- Ex.) りんごの値段の計算



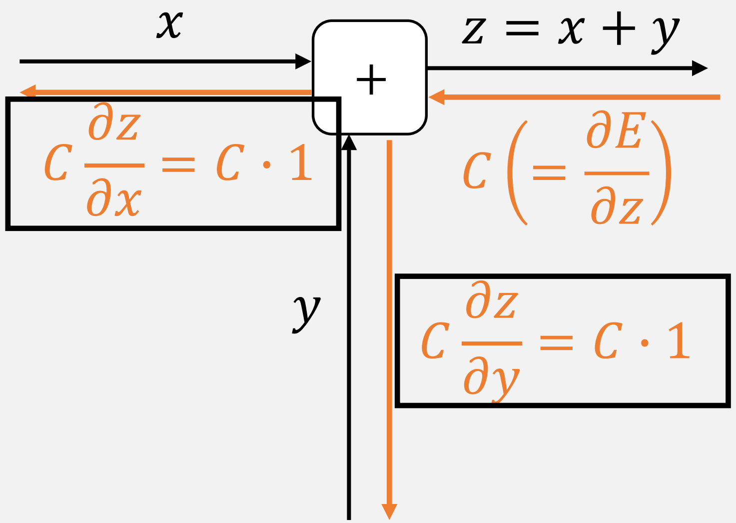
chain-rule（連鎖律）

- $\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial x}$
- $\frac{\partial E}{\partial y} = C$とわかっているとき，　$\frac{\partial E}{\partial x} = C\frac{\partial y}{\partial x}$と求められる



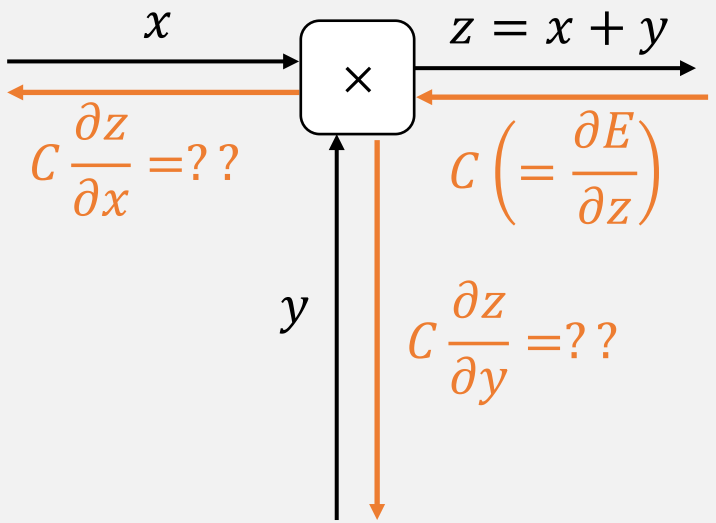
和のchain-rule

- 足し算のとき，微分はどう伝播するか
- $z = x + y \Longrightarrow \dfrac{\partial z}{\partial x} = 1,\ \dfrac{\partial E}{\partial y} = 1$
- $\dfrac{\partial E}{\partial y} = C$とわかっているので，$\dfrac{\partial E}{\partial x}, \dfrac{\partial E}{\partial y}$は以下のようになる



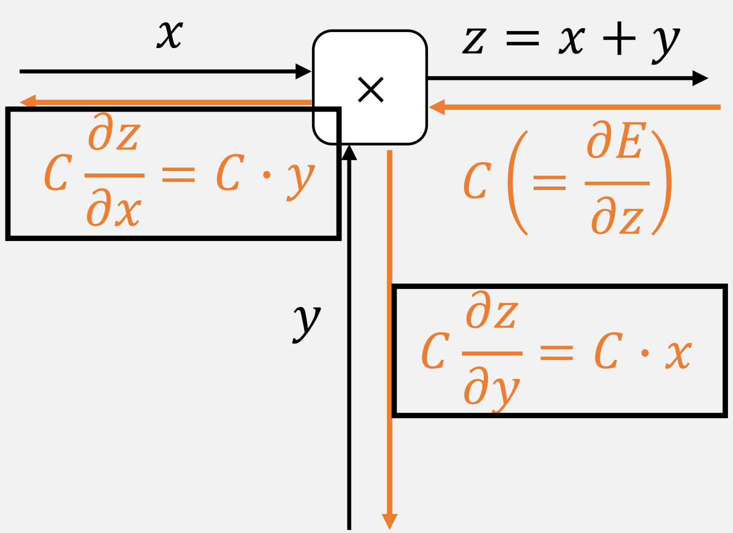
積のchain-rule

- 掛け算のとき，微分はどう伝播するか
- $z = xy \Longrightarrow \dfrac{\partial z}{\partial x} = y,\ \dfrac{\partial E}{\partial y} = x$
- $\dfrac{\partial E}{\partial y} = C$とわかっているので，$\dfrac{\partial E}{\partial x}, \dfrac{\partial E}{\partial y}$は以下のようになる



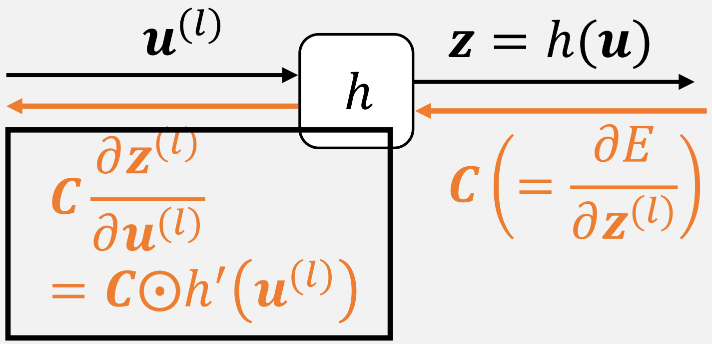
積のchain-rule

- 掛け算のとき，微分はどう伝播するか
- $z = xy \Longrightarrow \frac{\partial z}{\partial x} = y,\ \frac{\partial E}{\partial y} = x$
- $\frac{\partial E}{\partial y} = C$とわかっているので，$\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}$は以下のようになる



NN①: 多次元関数のchain-rule

- ベクトルの各要素に関数を作用させたとき，微分はどう伝播するか
- $\boldsymbol{z}^{(l)} = h\left(\boldsymbol{u}^{(l)}\right) \Rightarrow \dfrac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{u}^{(l)}} = h'\left(\boldsymbol{u}^{(l)}\right)$
  (本当は$\mathrm{diag}\left(h'\left(u_1^{(l)}\right), \cdots, h'\left(u_1^{(l)}\right)\right)$)
- $\dfrac{\partial E}{\partial \boldsymbol{z}^{(l)}} = \boldsymbol{C}$とわかっているので，$\dfrac{\partial E}{\partial \boldsymbol{u}^{(l)}}$は以下のようになる



$$\boldsymbol{C} = \frac{\partial E}{\partial \boldsymbol{z}^{(l)}} = \left[\frac{\partial E}{\partial z_n^{(l)}}, \dots, \frac{\partial E}{\partial z_n^{(l)}}\right] \in \mathbb{R}^n$$

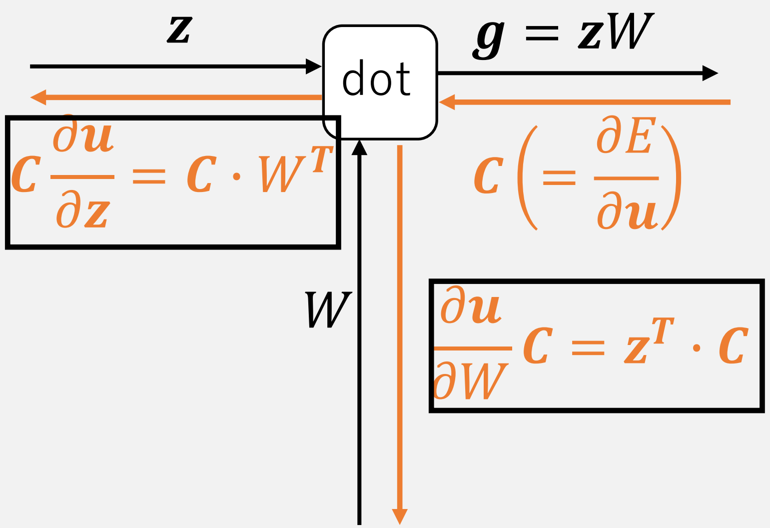
$$\frac{\partial E}{\partial \boldsymbol{u}^{(l)}} = \left[\frac{\partial E}{\partial u_1^{(l)}}, \dots, \frac{\partial E}{\partial u_n^{(l)}}\right] \in \mathbb{R}^n$$

⊙: 要素ごとの積
（アダマール積）

$$\boldsymbol{C} \odot h'(\boldsymbol{u}) = \begin{bmatrix} \dfrac{\partial E}{\partial z_1} h'(u_1) \\ \vdots \\ \dfrac{\partial E}{\partial z_n} h'(u_n) \end{bmatrix}$$

行列積のchain-rule

- 行列と (行) ベクトルの積のとき，微分はどう伝播するか
- $\boldsymbol{g} = \boldsymbol{z}W$
- $\frac{\partial E}{\partial \boldsymbol{g}} = \boldsymbol{C}$とわかっているので，$\frac{\partial E}{\partial \boldsymbol{z}}, \frac{\partial E}{\partial W}$は以下のようになる



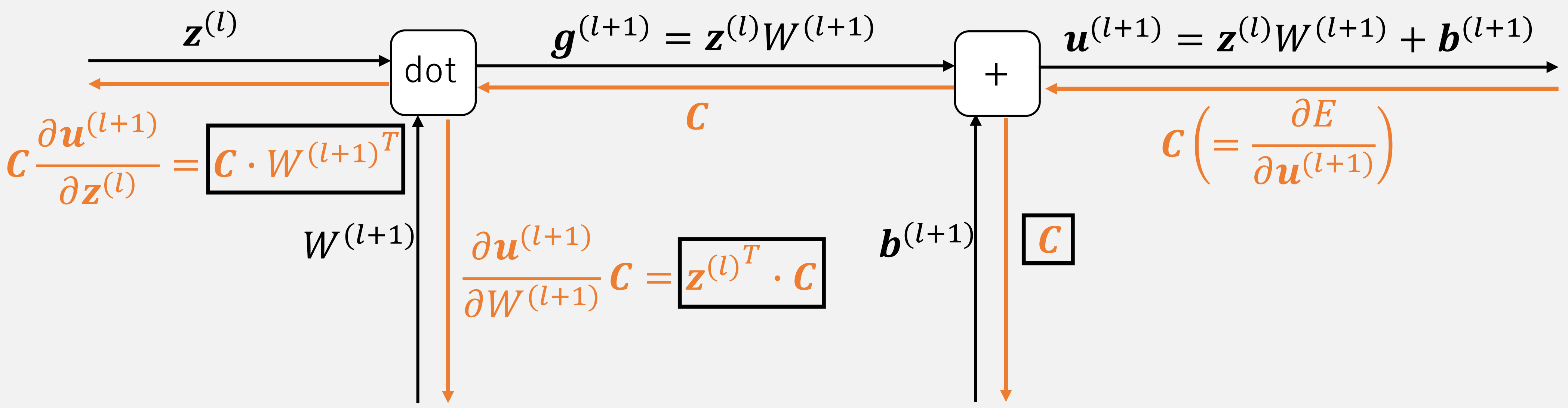
$$\frac{\partial E}{\partial \boldsymbol{g}} = \left[\frac{\partial E}{\partial g_1}, \dots, \frac{\partial E}{\partial g_m}\right] \in \mathbb{R}^m$$

$$\frac{\partial E}{\partial \boldsymbol{z}} = \left[\frac{\partial E}{\partial z_1}, \dots, \frac{\partial E}{\partial z_n}\right] \in \mathbb{R}^n$$

$$\frac{\partial E}{\partial W} = \begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \cdots & \frac{\partial E}{\partial w_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{m1}} & \cdots & \frac{\partial E}{\partial w_{mn}} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

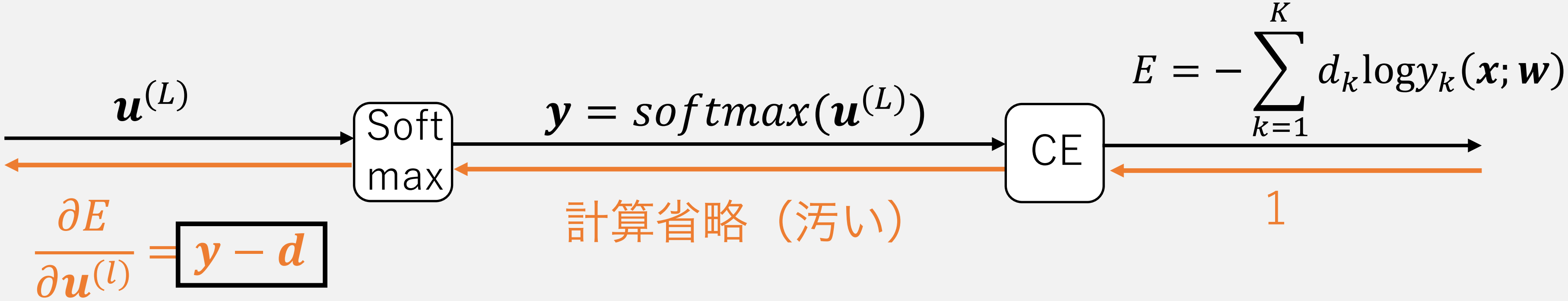
NN②: アフィン変換のchain-rule

- 行列と(行) ベクトルの積にバイアス項を足したとき，微分はどう伝播するか
- $\boldsymbol{u}^{(l+1)} = \boldsymbol{z}^{(l)} W^{(l+1)} + \boldsymbol{b}^{(l+1)}$
- $\dfrac{\partial E}{\partial \boldsymbol{u}^{(l+1)}} = \boldsymbol{C}$とわかっているので，$\dfrac{\partial E}{\partial \boldsymbol{z}^{(l)}}, \dfrac{\partial E}{\partial W^{(l+1)}}$は以下のようになる



NN③: softmax + cross entropyの chain-rule

- 行列と(行) ベクトルの積にバイアス項を足したとき，微分はどう伝播するか
- $E = CrossEntropy(\boldsymbol{t}, \boldsymbol{y}),\quad \boldsymbol{y} = Softmax(\boldsymbol{u}^{(L)})$
- $\frac{\partial E}{\partial E} = 1$とわかっているので，$\frac{\partial E}{\partial \boldsymbol{u}^{(L)}}$, は以下のようになる

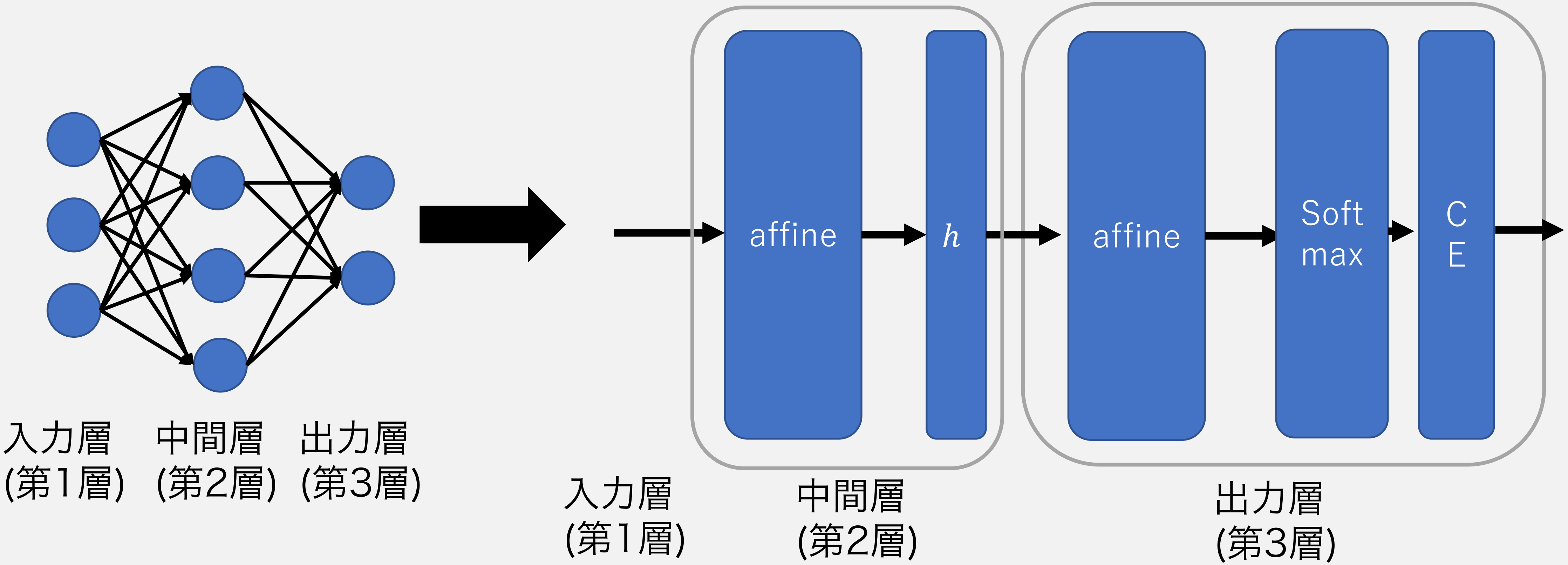


計算の補足

- あとで作る

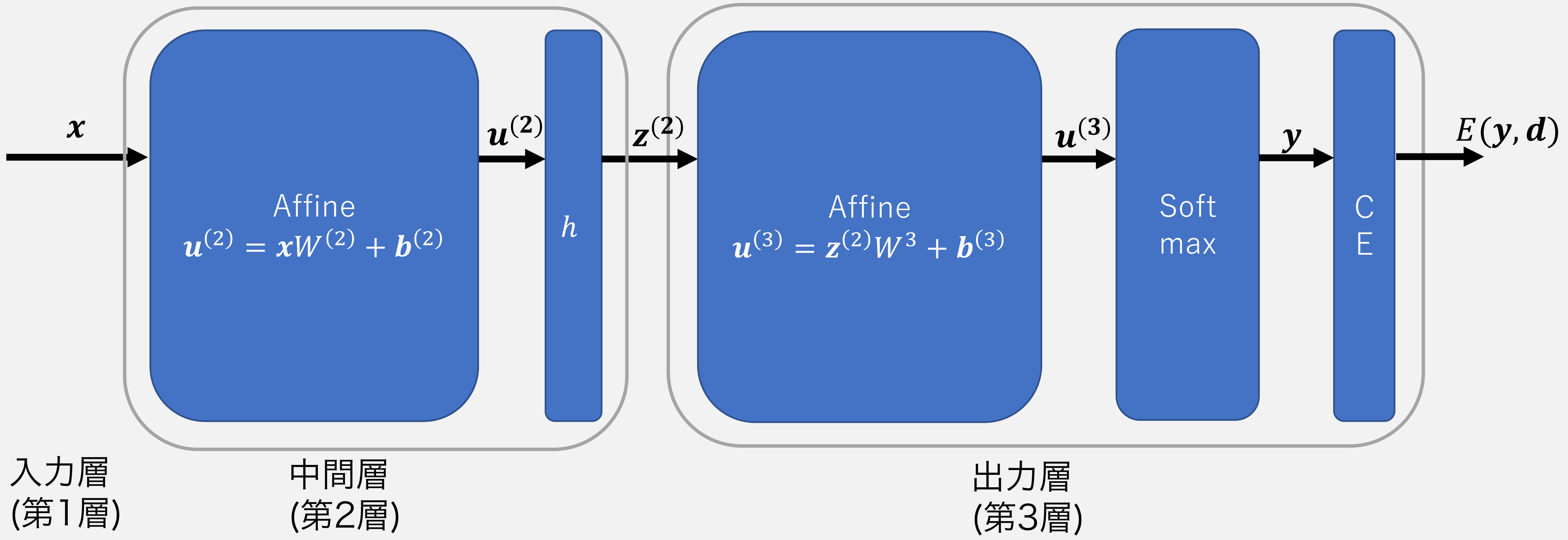
NNを計算グラフで表す

- NNも計算グラフで表せる
- 3層のNNについて，誤差逆伝播法を使い微分を求めてみる

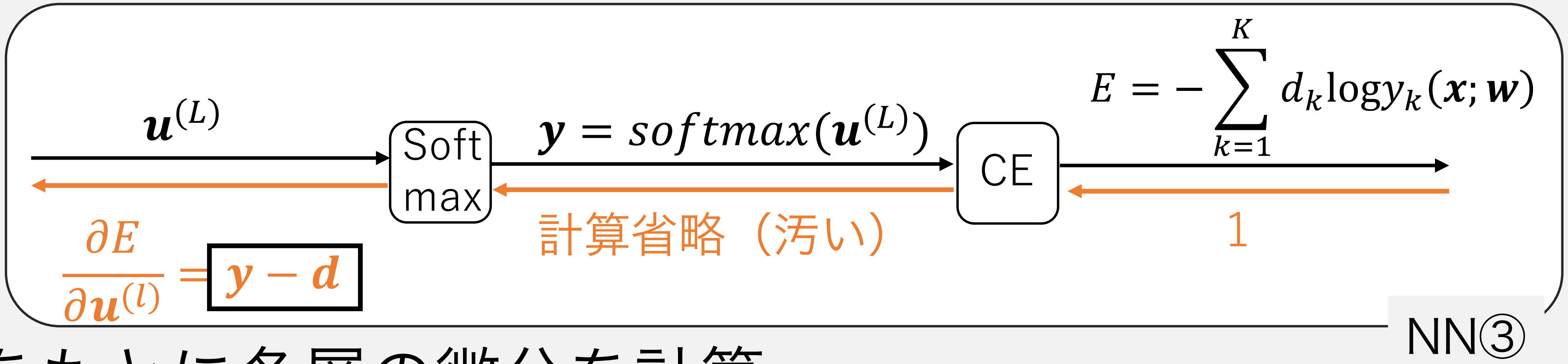


NNの学習

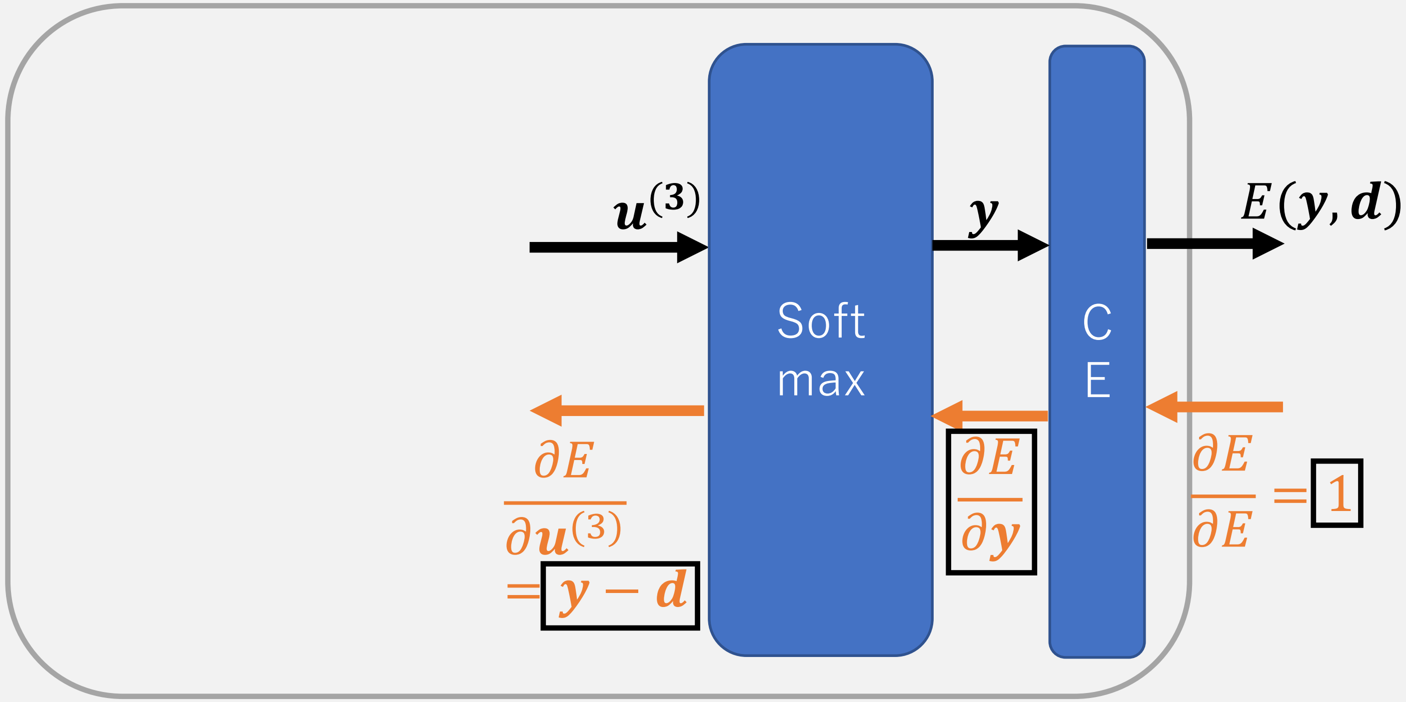
1. xを入力して損失関数の値を計算



NNの学習

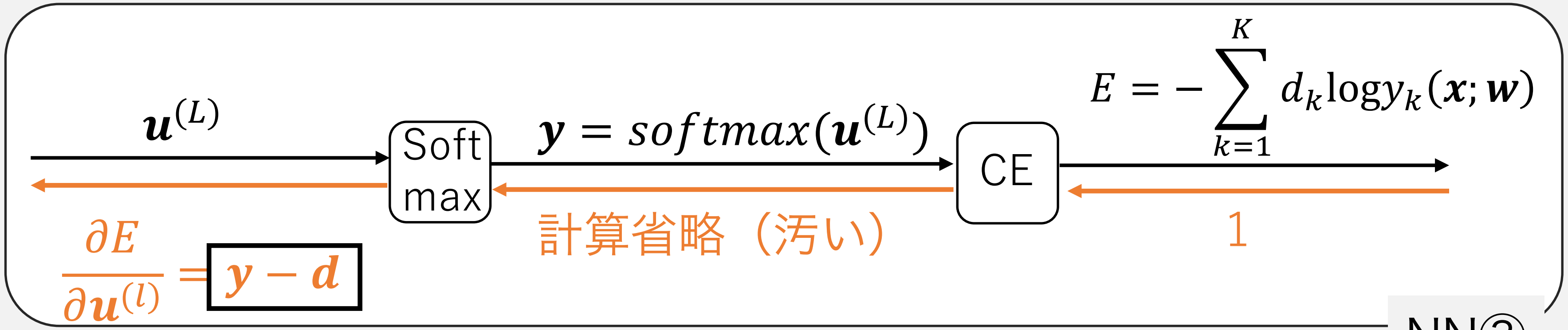


2. 損失関数の値をもとに各層の微分を計算



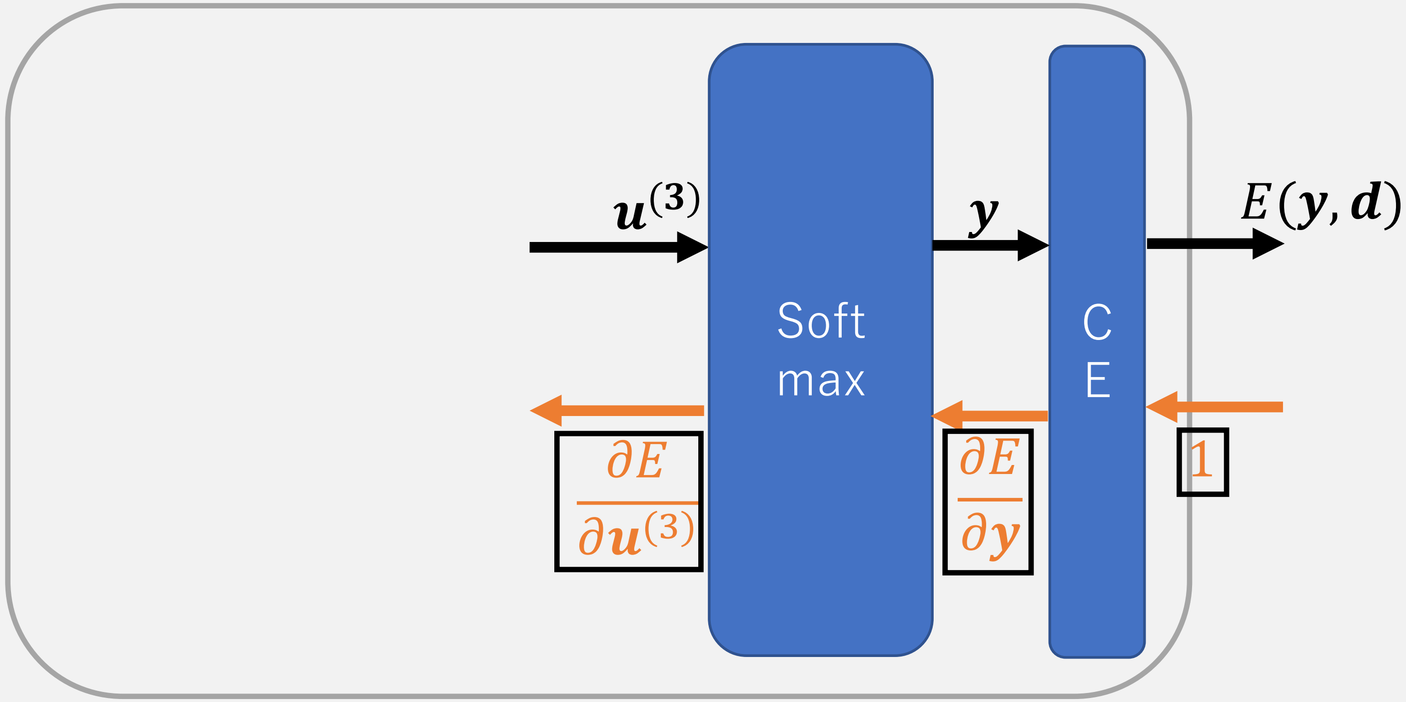
出力層
(第3層)

NNの学習



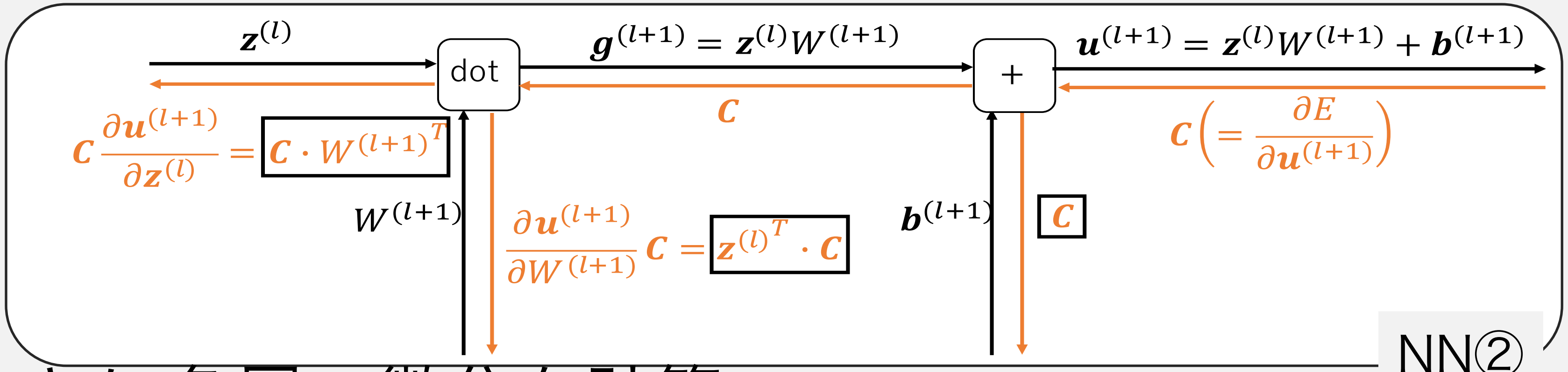
NN③

2. 損失関数の値をもとに各層の微分を計算

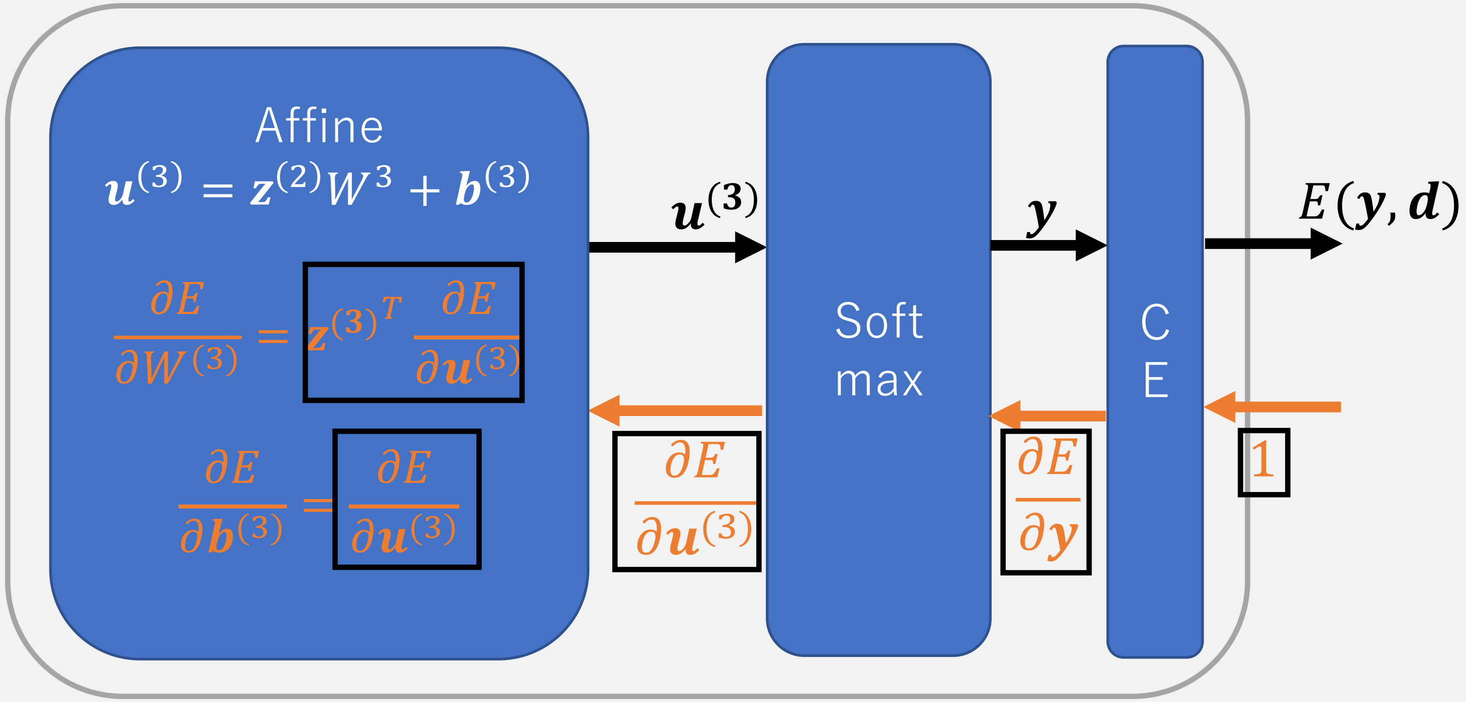


出力層
(第3層)

NNの学習



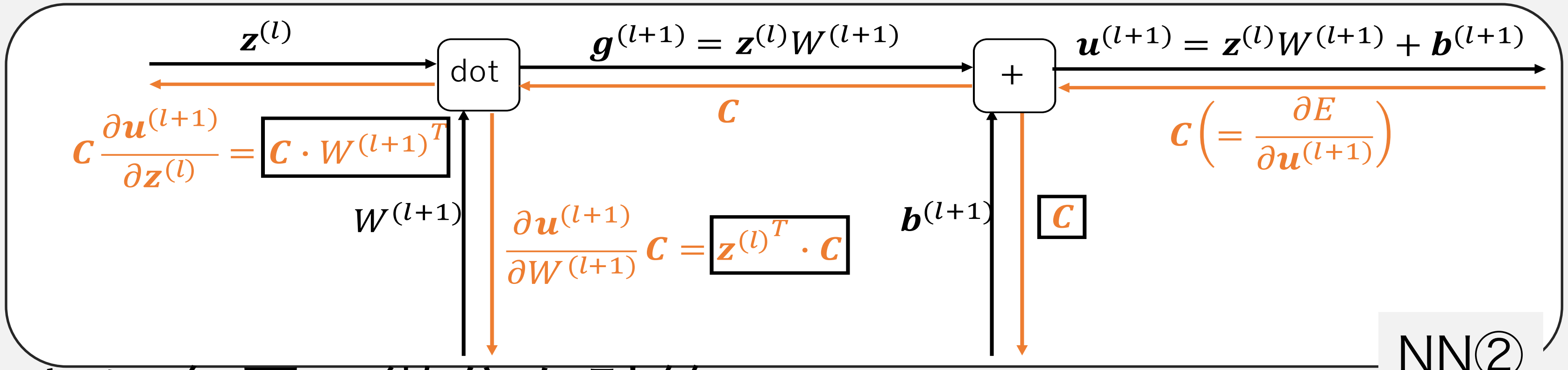
2. 損失関数の値をもとに各層の微分を計算



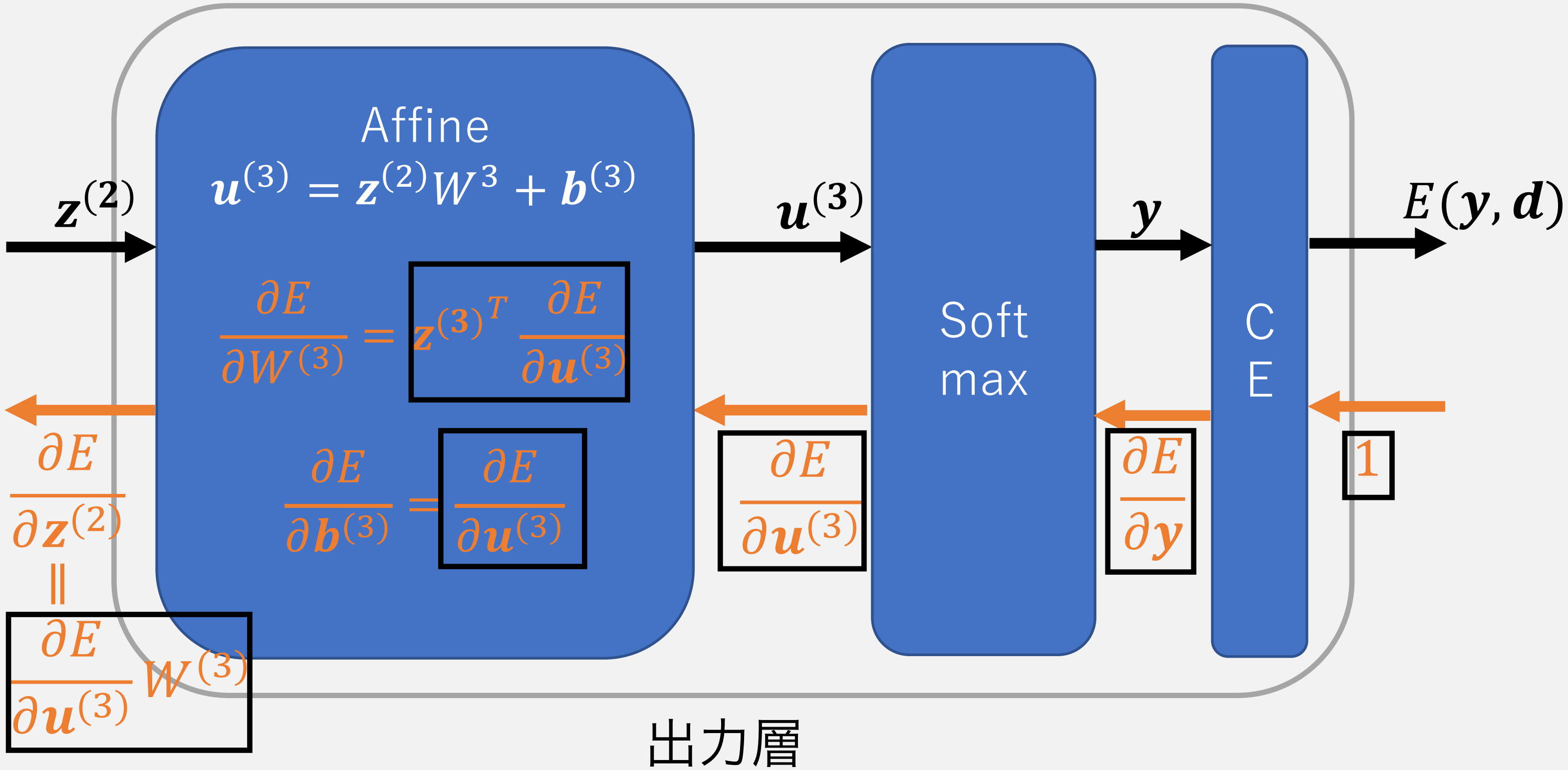
出力層
(第3層)

NNの学習

2. 損失関数の値をもとに各層の微分を計算

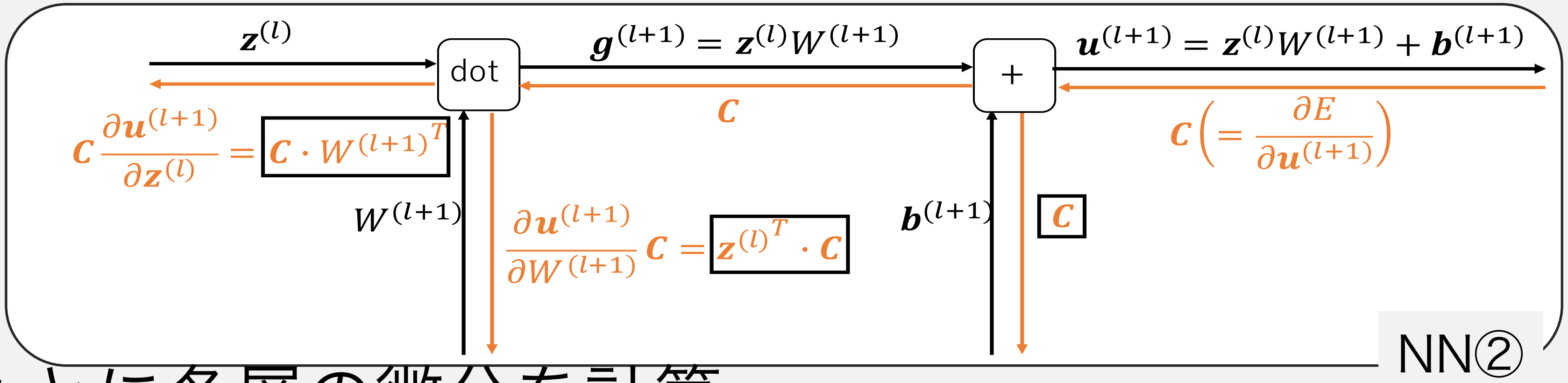


NN②

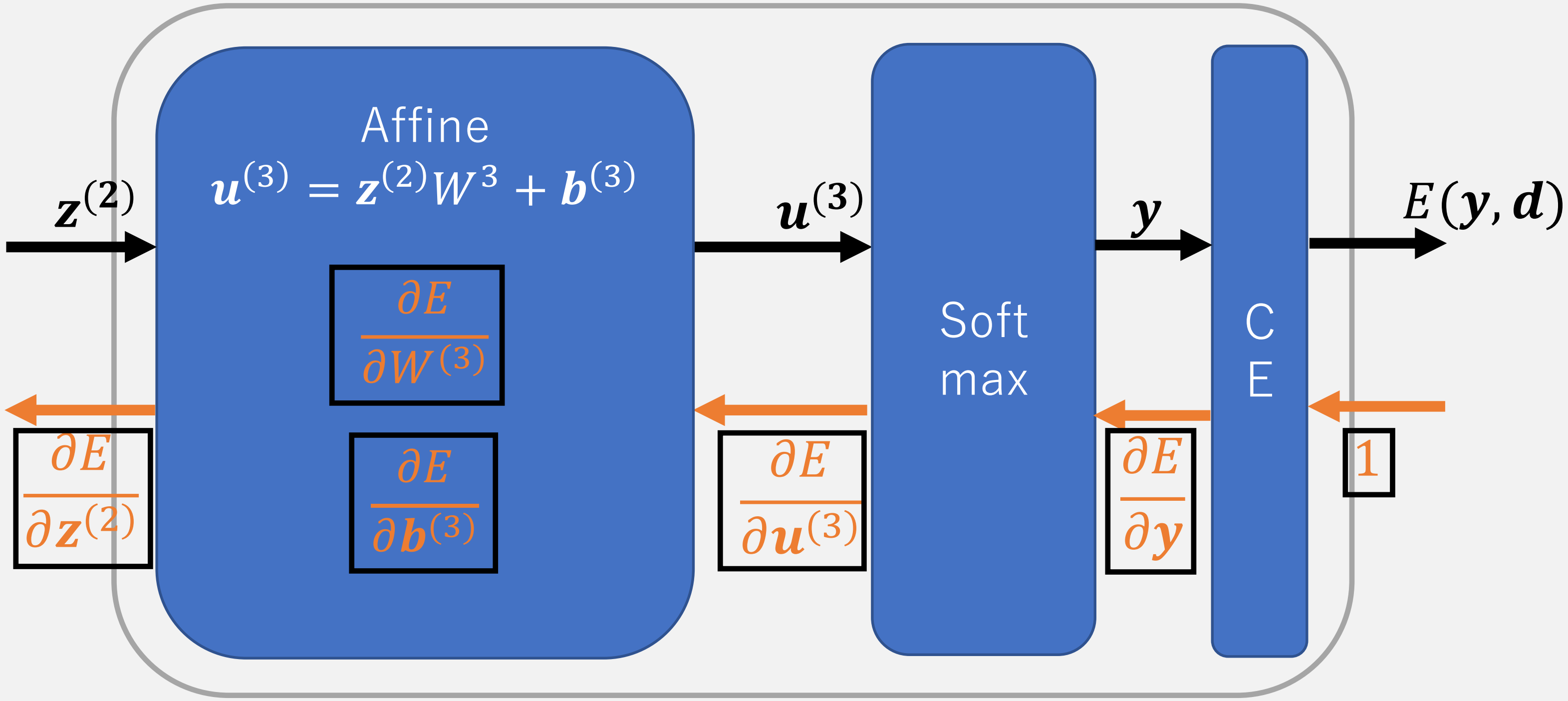


出力層
(第3層)

NNの学習



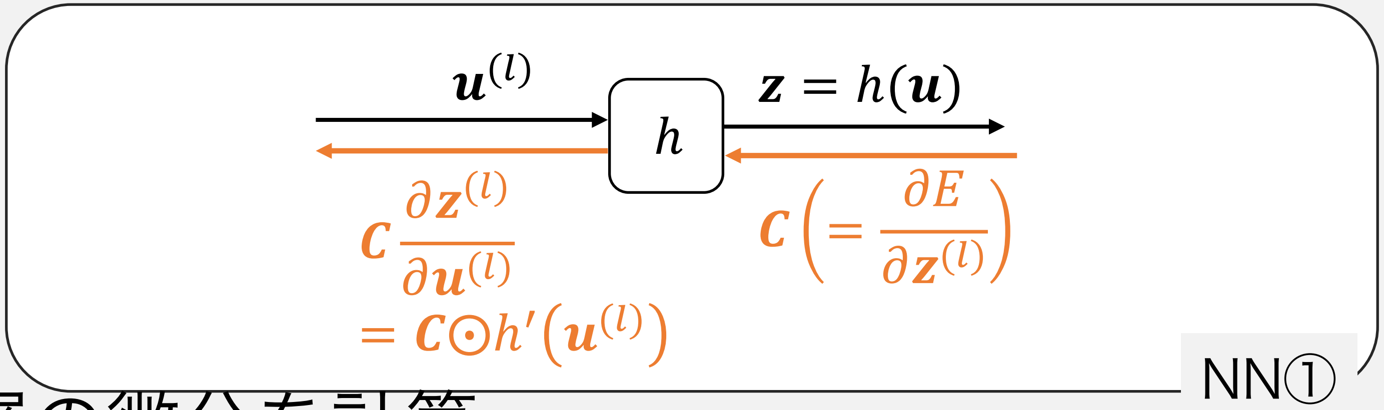
2. 損失関数の値をもとに各層の微分を計算

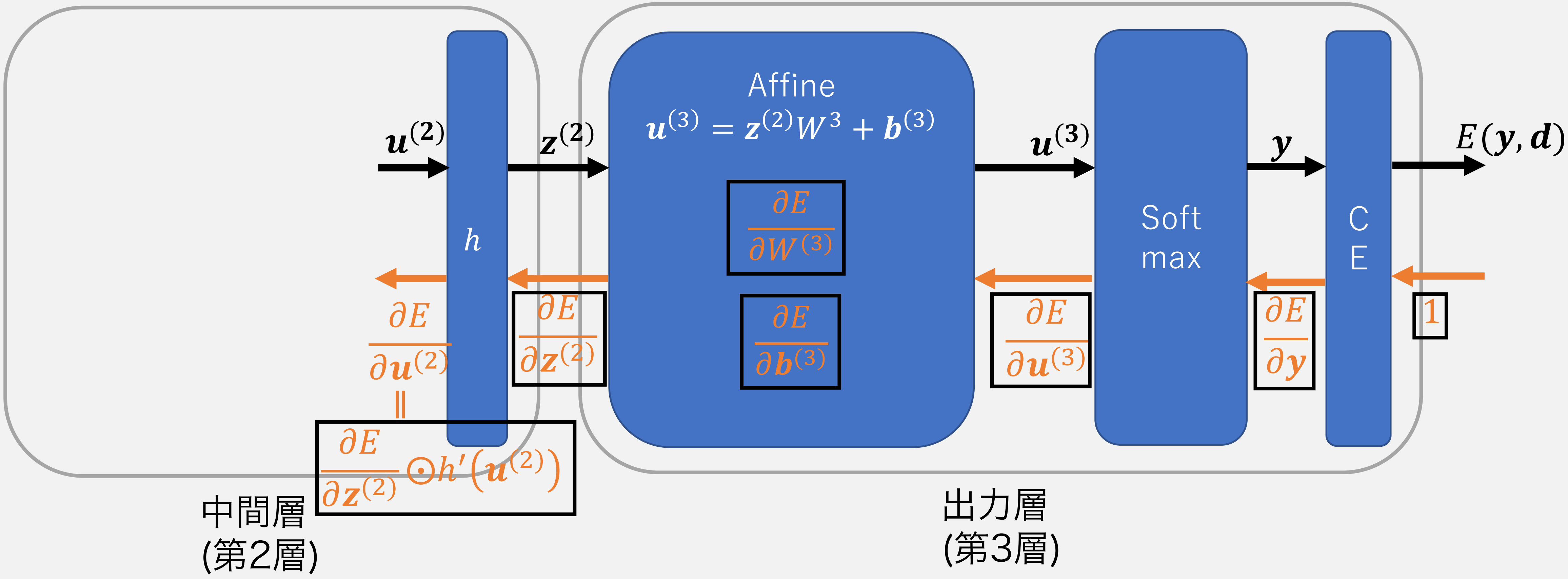


出力層
(第3層)

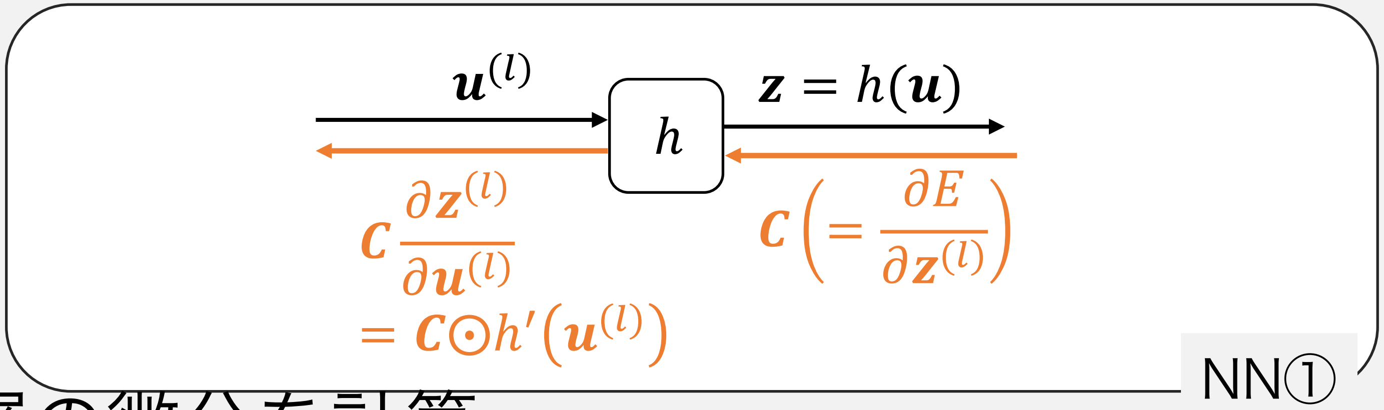
NNの学習

2. 損失関数の値をもとに各層の微分を計算

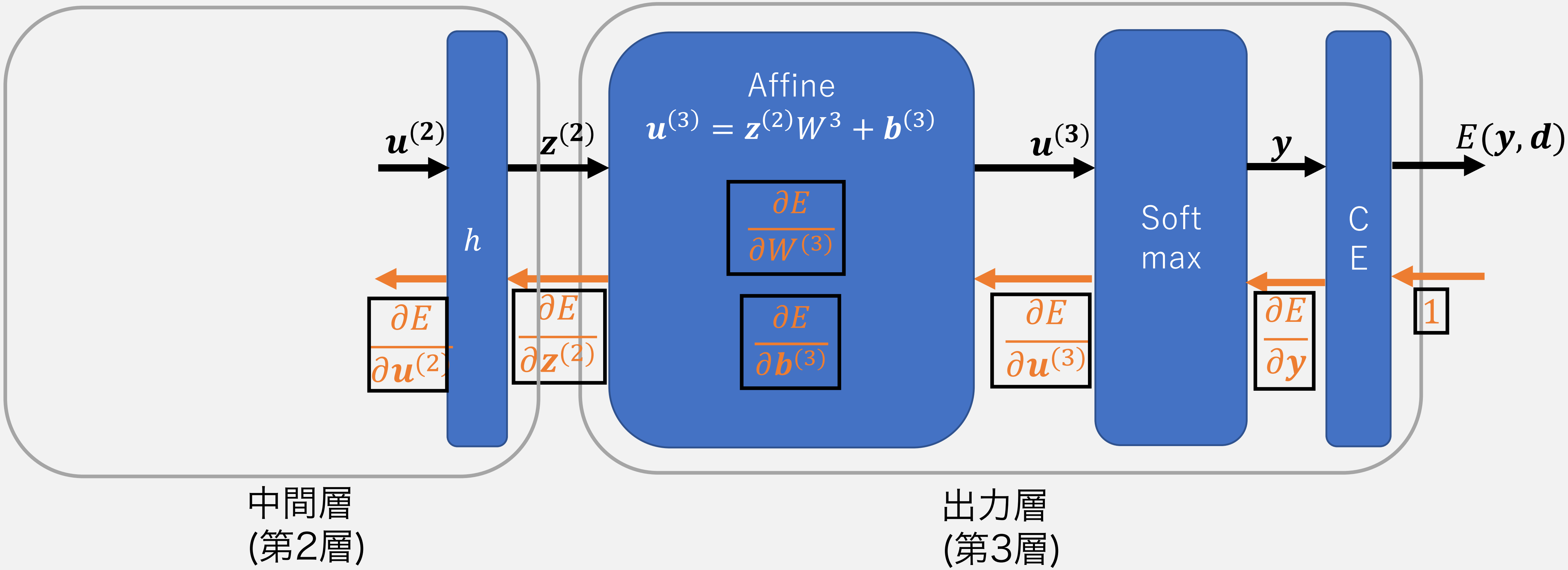




NNの学習

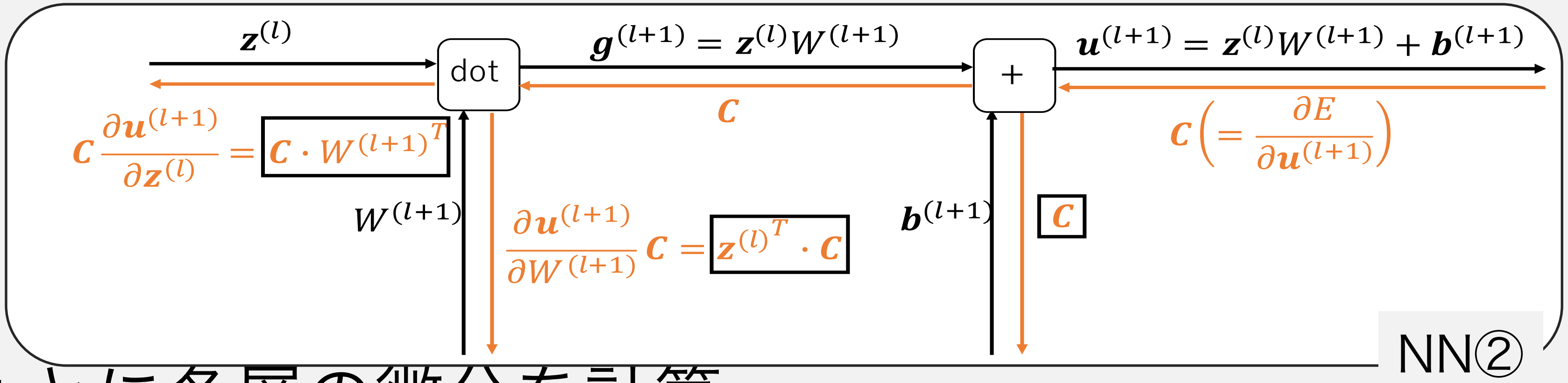


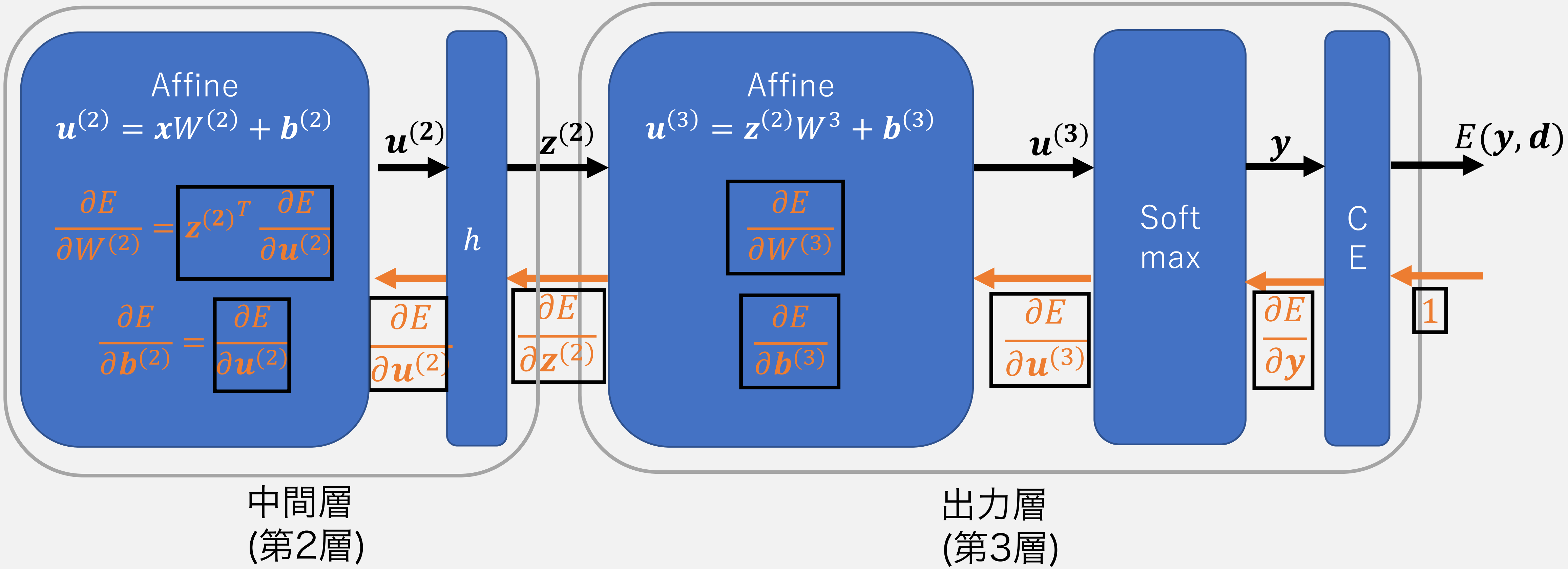
2. 損失関数の値をもとに各層の微分を計算



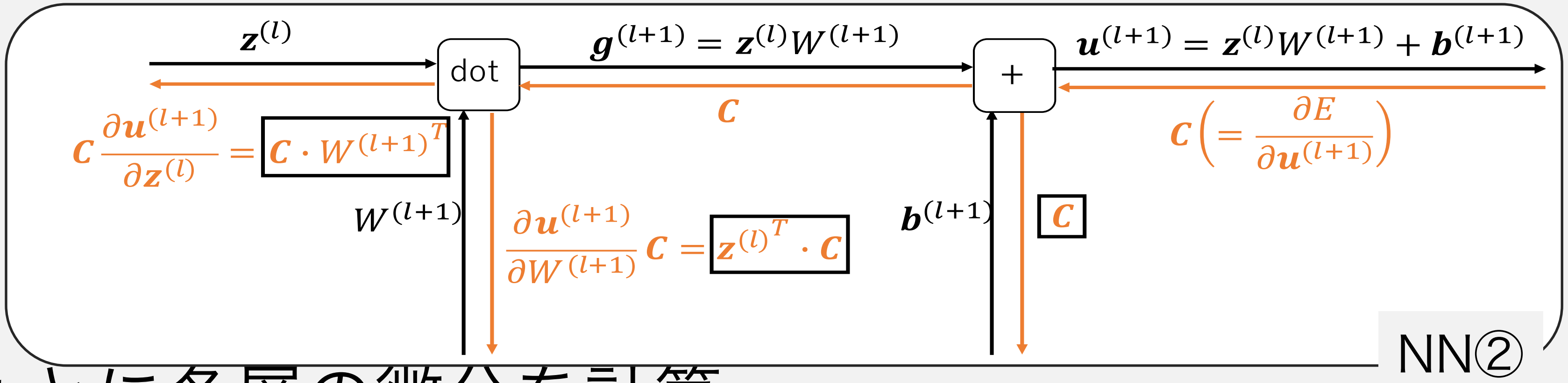
NNの学習

2. 損失関数の値をもとに各層の微分を計算

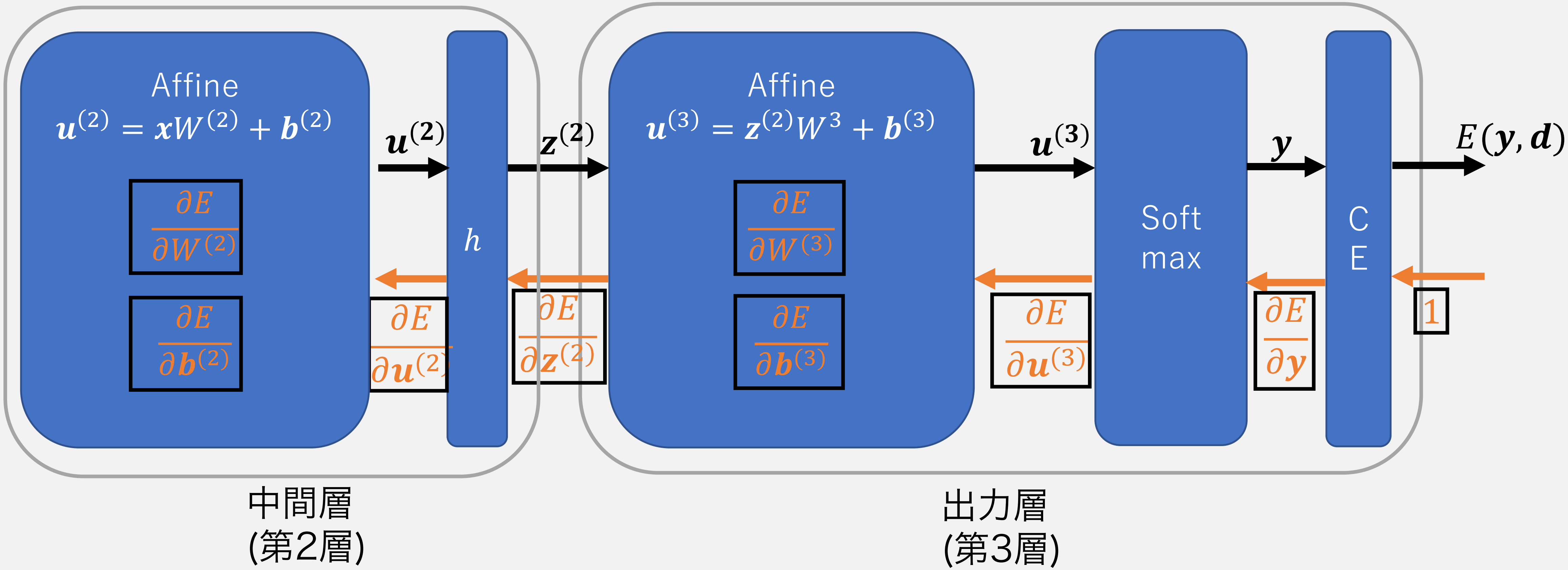




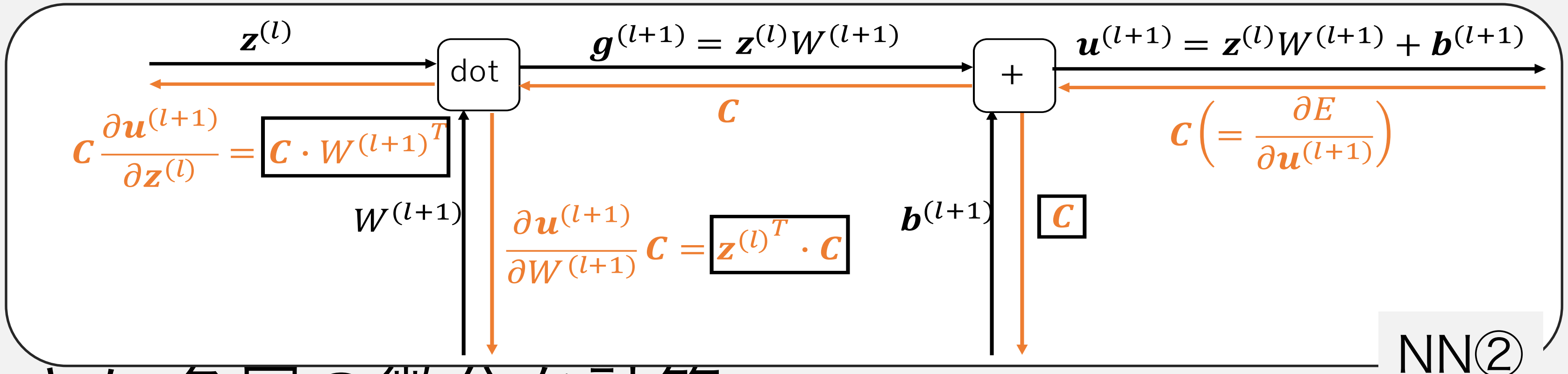
NNの学習



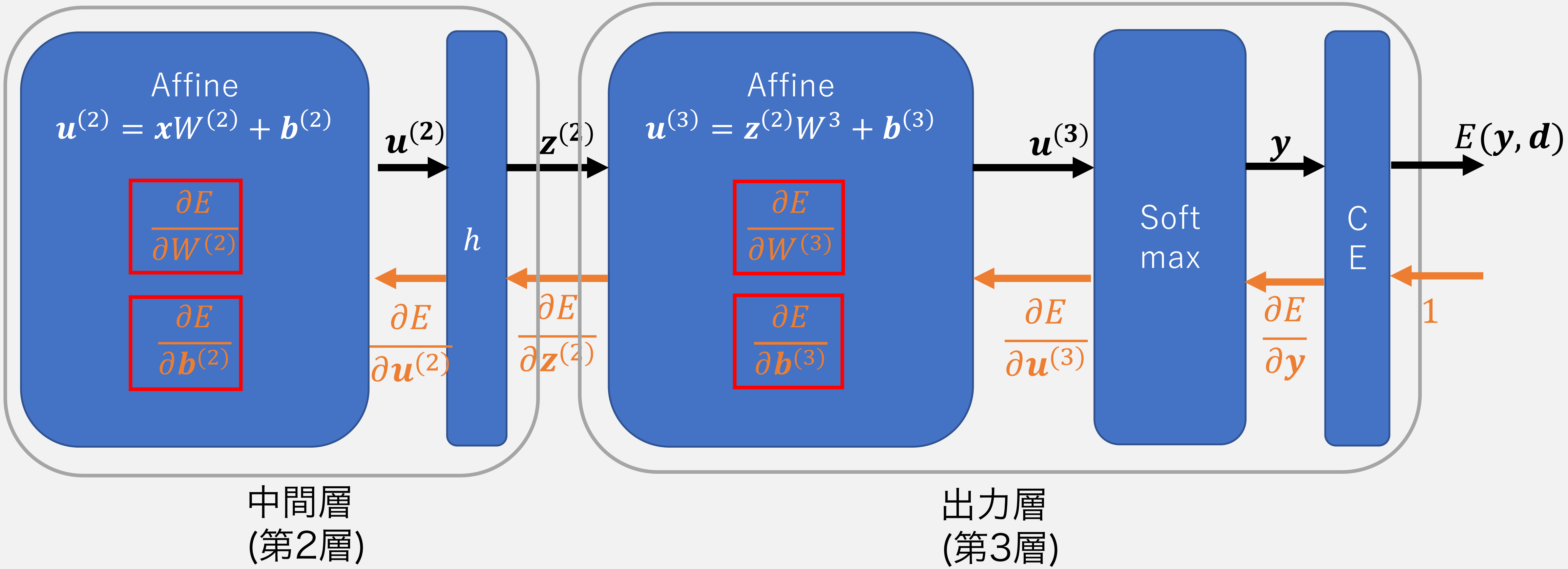
2. 損失関数の値をもとに各層の微分を計算



NNの学習



2. 損失関数の値をもとに各層の微分を計算



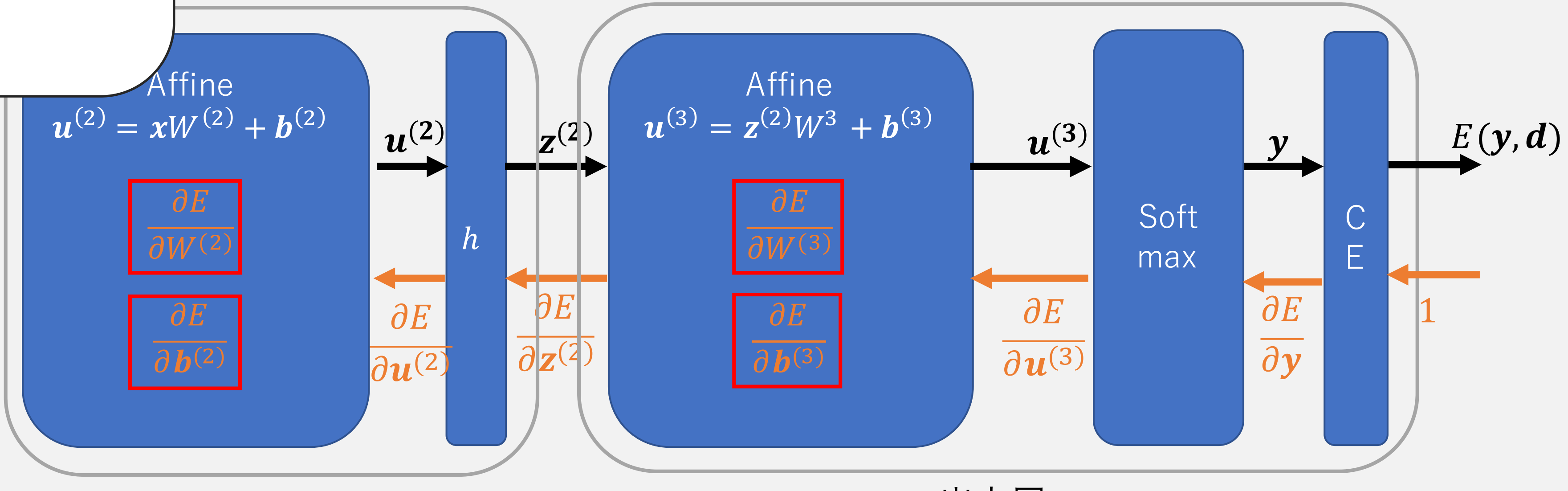
NNの学習

3. 求めた微分をもとに，損失関数が小さくなる方向に値を更新

$$W^{(l)} \leftarrow W^{(l)} - \alpha \frac{\partial E}{\partial W^{(l)}}$$

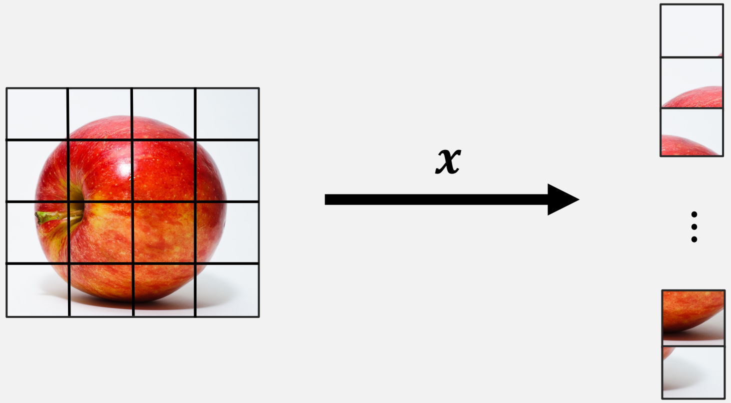
$$\boldsymbol{b}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \alpha \frac{\partial E}{\partial \boldsymbol{b}^{(l)}}$$

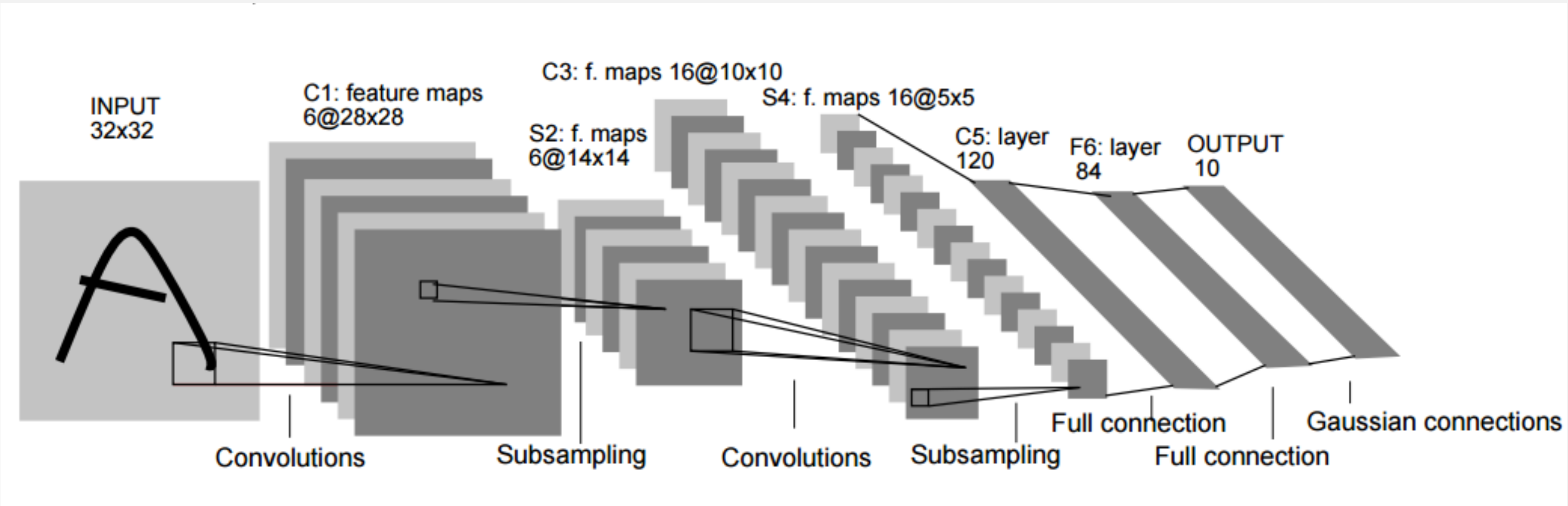
$$(l = 2,3)$$



畳み込みニューラルネットワーク (CNN)

- 今回は最も基本的なNNである多層パーセプトロンを説明した.
- 画像識別にももちろん使えるが，入力する際に画像をバラバラにしてしまっている.　→ CNN





LeNet[LeCun et.al, 1988]

まとめ

- 非線形な識別方法として，ニューラルネットワークというものがある
- ニューラルネットワークはSGDで学習するが，数値微分で微分を求めると計算量が膨大
  - 誤差逆伝播法（バックプロパゲーション）で求める
- 画像認識においてはCNNがよく使われる（次回詳細）

今日話していないが重要なこと

- 正則化
  - 過学習を防ぐための一手法
- ハイパーパラメータの調整
  - ハイパーパラメータとはNNであれば学習率α，中間層の数など，事前に定める必要のあるパラメータのこと
  - これらの良い値を求める手法として交差検証法（クロスバリデーション）というものがある
- どちらも教科書に載っているので読んでみてください

演習

- ニューラルネットワーク（多層パーセプトロン）の実装
  - 1101/main.ipynbを開いてください