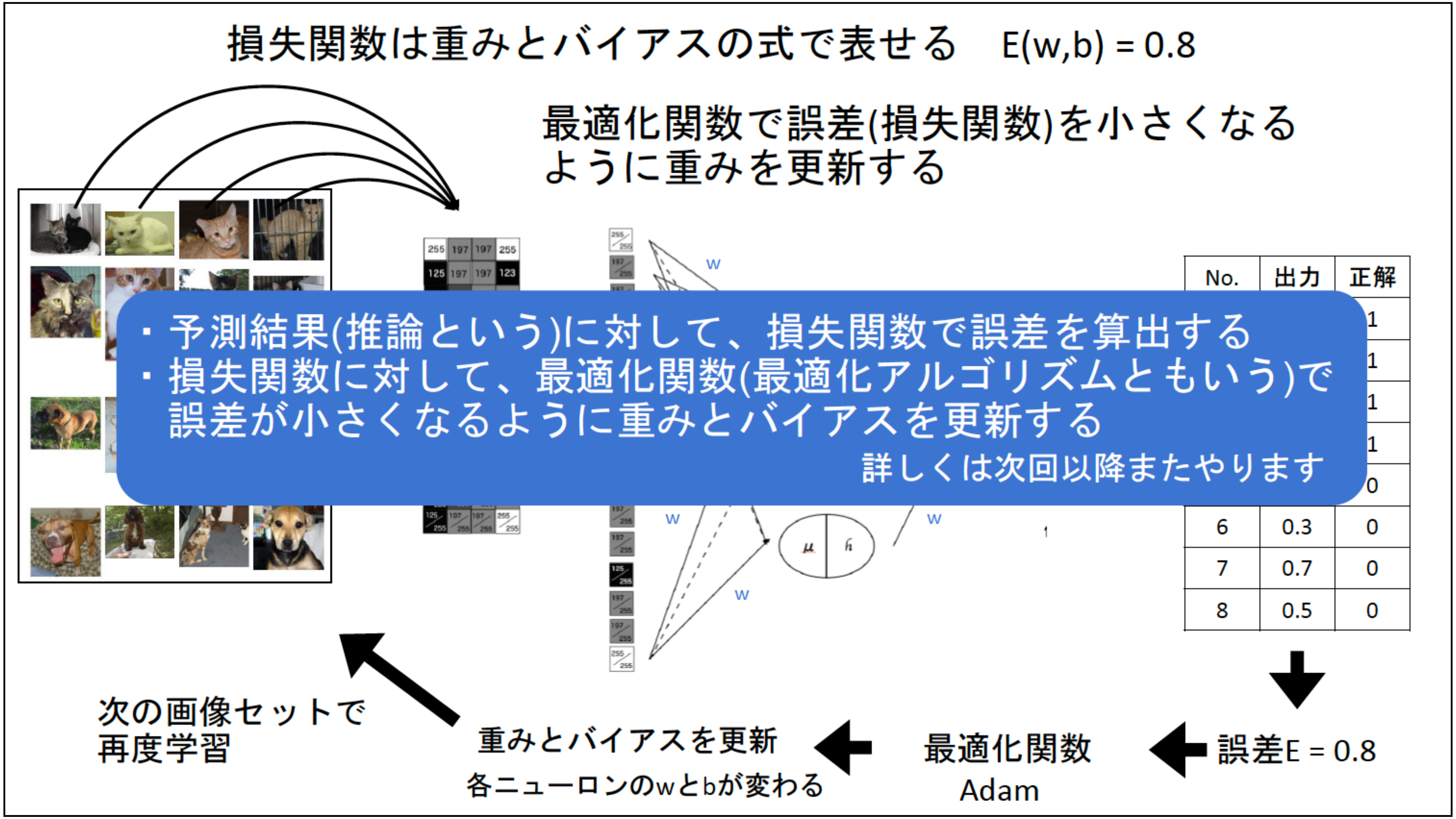
医療とAI・ビッグデータ応用
MLP　学習の仕組み

本スライドは、自由にお使いください。
使用した場合は、このQRコードからアンケートに回答をお願いします。

統合教育機構
須藤毅顕

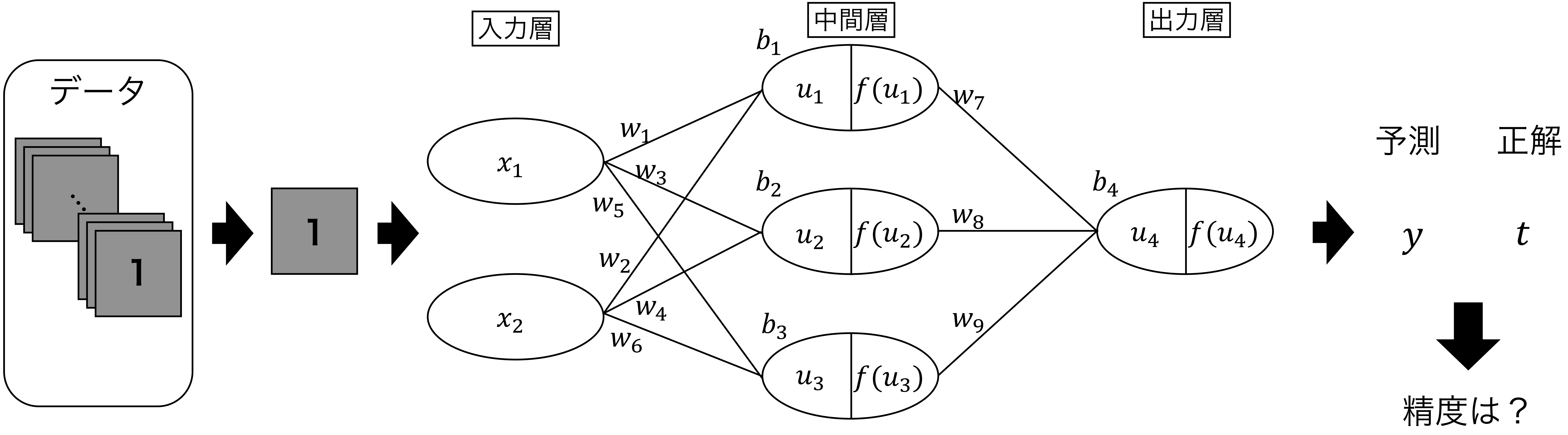
第3回のスライド



- 予測結果(推論という)に対して、損失関数で誤差を算出する
- 損失関数に対して、最適化関数(最適化アルゴリズムともいう)で誤差が小さくなるように重みとバイアスを更新する

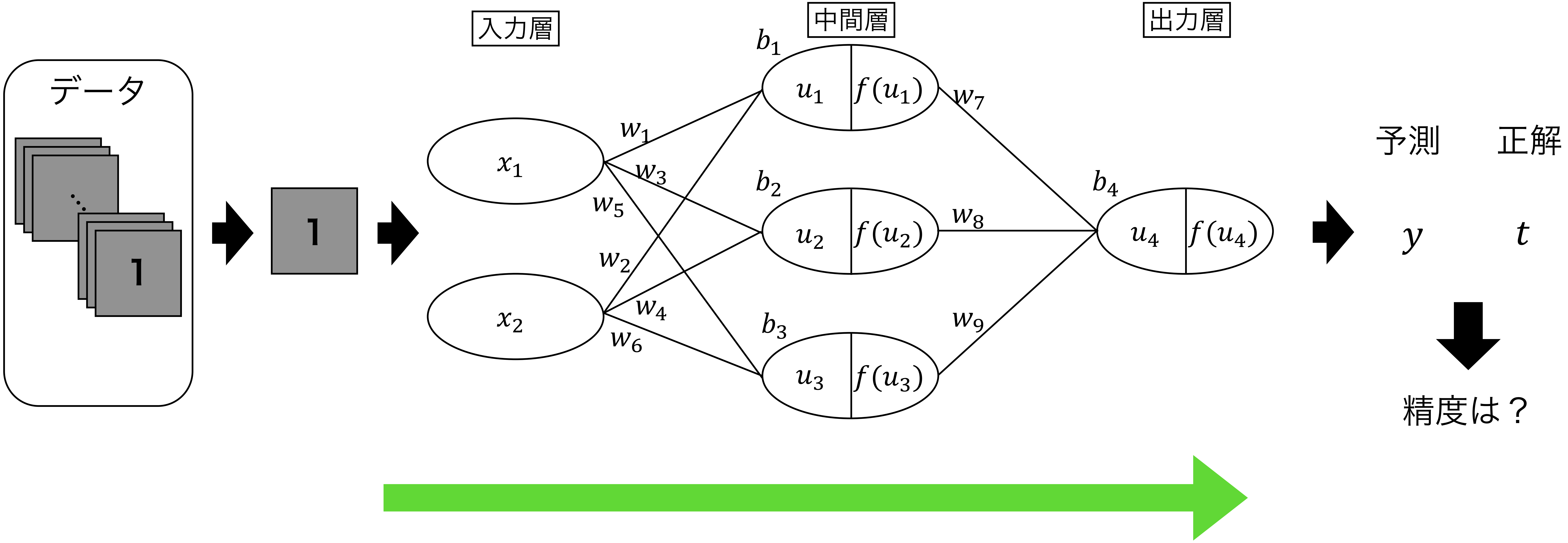
詳しくは次回以降またやります

順伝搬と逆伝播



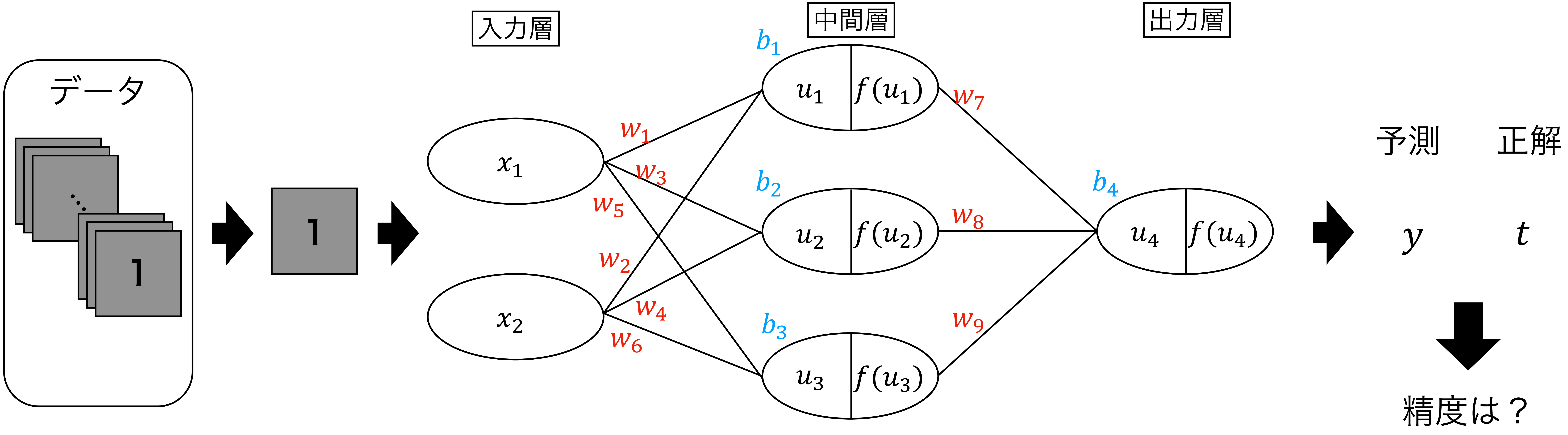
深層学習の推論(順伝播)と学習(逆伝播)の仕組みをもう少し勉強してみましょう

推論



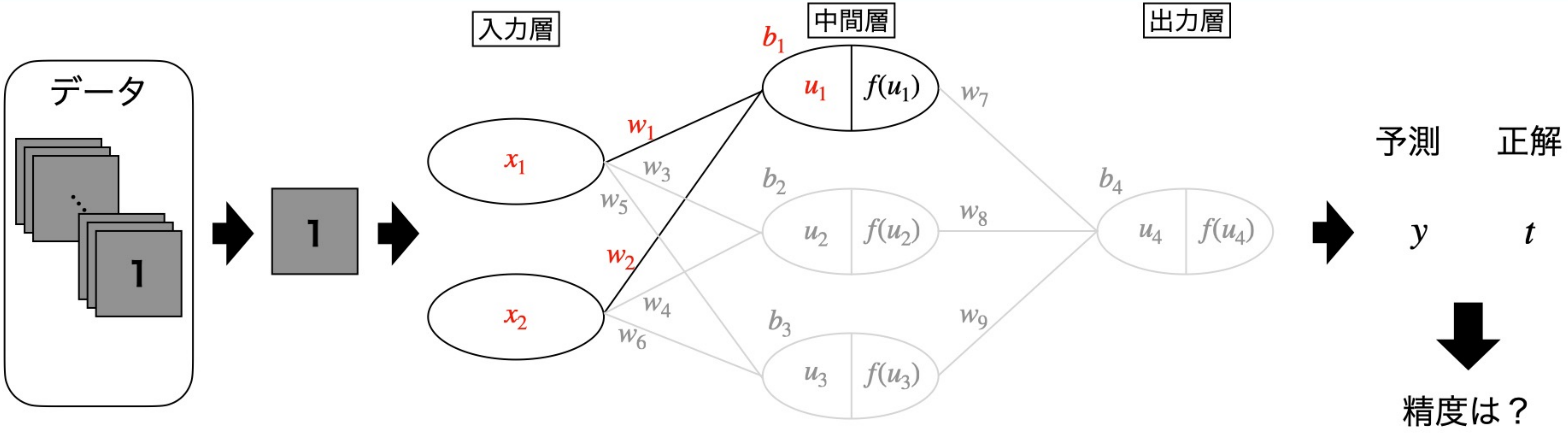
データを入力して結果を予測する流れを推論と言います
この図で言うと左から右へ流れていくので順伝播という言い方をします
(ここでは上の図のような簡単なモデルを想定してみます)

推論



それぞれのニューロンには前の層から繋がる重みwとバイアスbが存在します

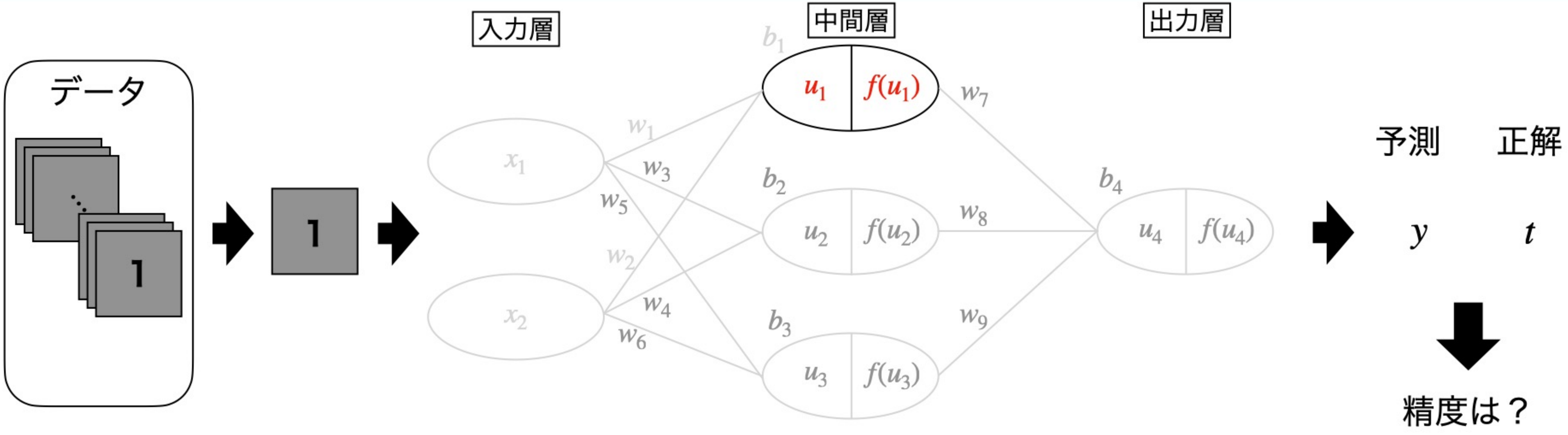
推論



中間層の1つ目のニューロンのu_1は入力層の各ニューロンの重み付き和とバイアスが足されたものです

$$u_1 = w_1x_1 + w_2x_2 + b_1$$

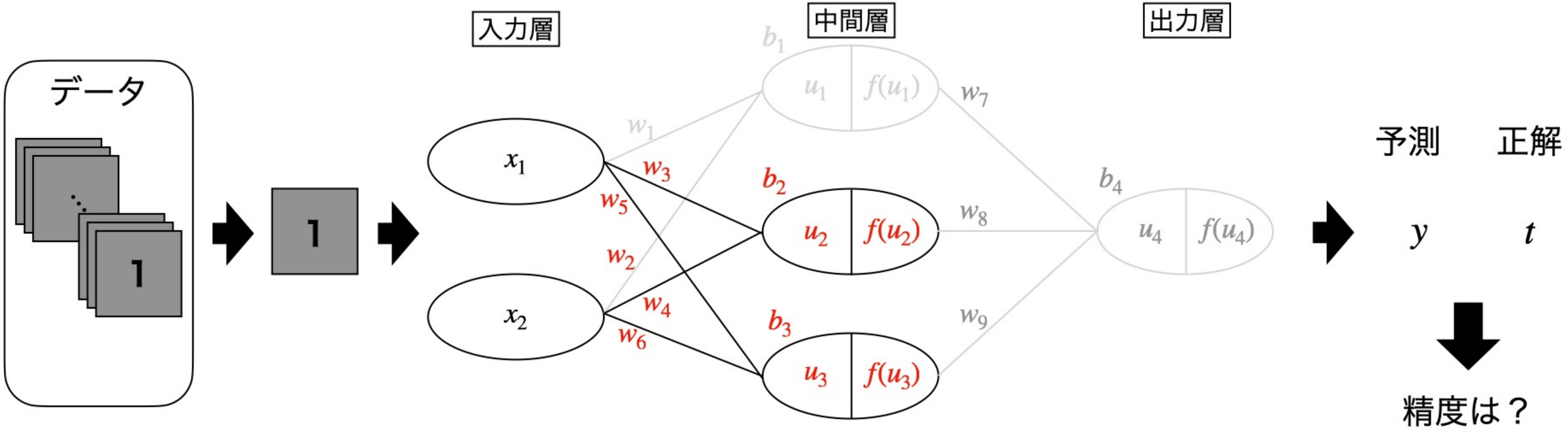
推論



中間層の1つ目のニューロンの $f(u_1)$ は活性化関数 $f(x)$ に u_1 を代入したものです
(活性化関数はRelu関数など)

$$f(u_1) = f(w_1x_1 + w_2x_2 + b_1)$$

推論



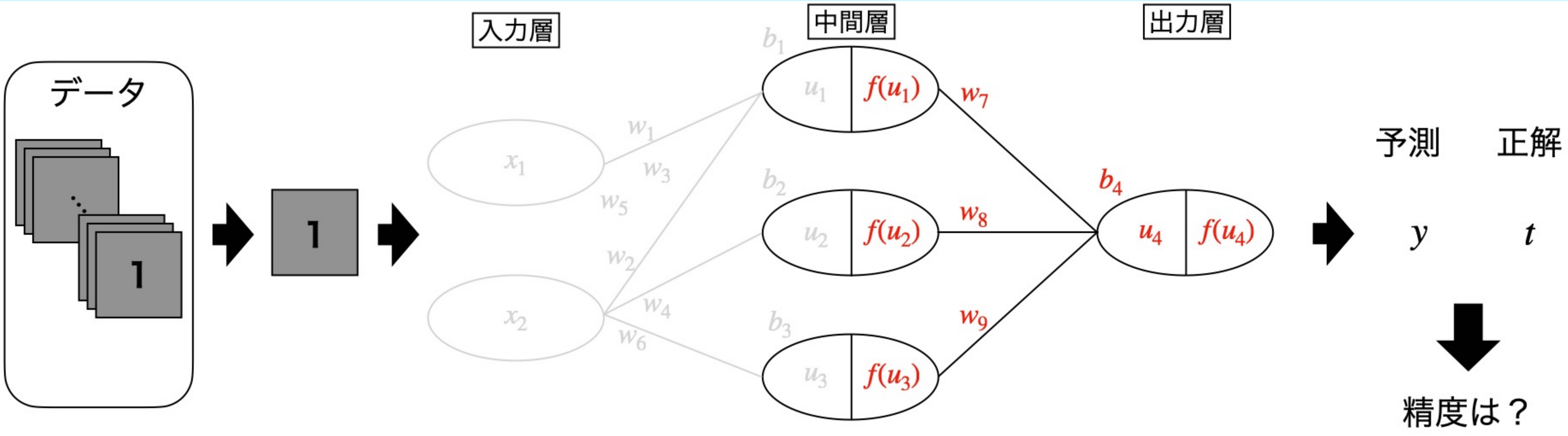
u_2、u_3、$f(u_2)$、$f(u_3)$、も同様です

$$u_1 = w_1x_1 + w_2x_2 + b_1 \qquad f(u_1) = f(w_1x_1 + w_2x_2 + b_1)$$

$$u_2 = w_3x_1 + w_4x_2 + b_2 \qquad f(u_2) = f(w_3x_1 + w_4x_2 + b_2)$$

$$u_3 = w_5x_1 + w_6x_2 + b_3 \qquad f(u_3) = f(w_5x_1 + w_6x_2 + b_3)$$

推論

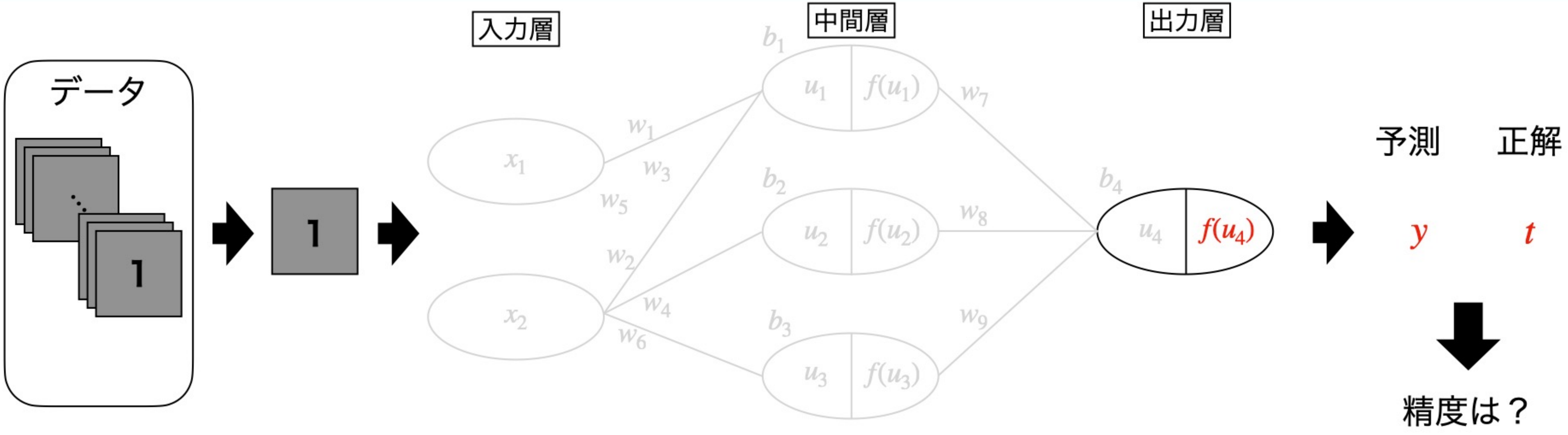


出力層のニューロンの u_4 は中間層の各ニューロンの重み付き和とバイアスが足されたものです
(活性化関数はシグモイド関数やソフトマックス関数など)

$$u_4 = w_7 f(u_1) + w_8 f(u_2) + w_9 f(u_3) + b_4$$

$$f(u_4) = f(w_7 f(u_1) + w_8 f(u_2) + w_9 f(u_3) + b_4)$$

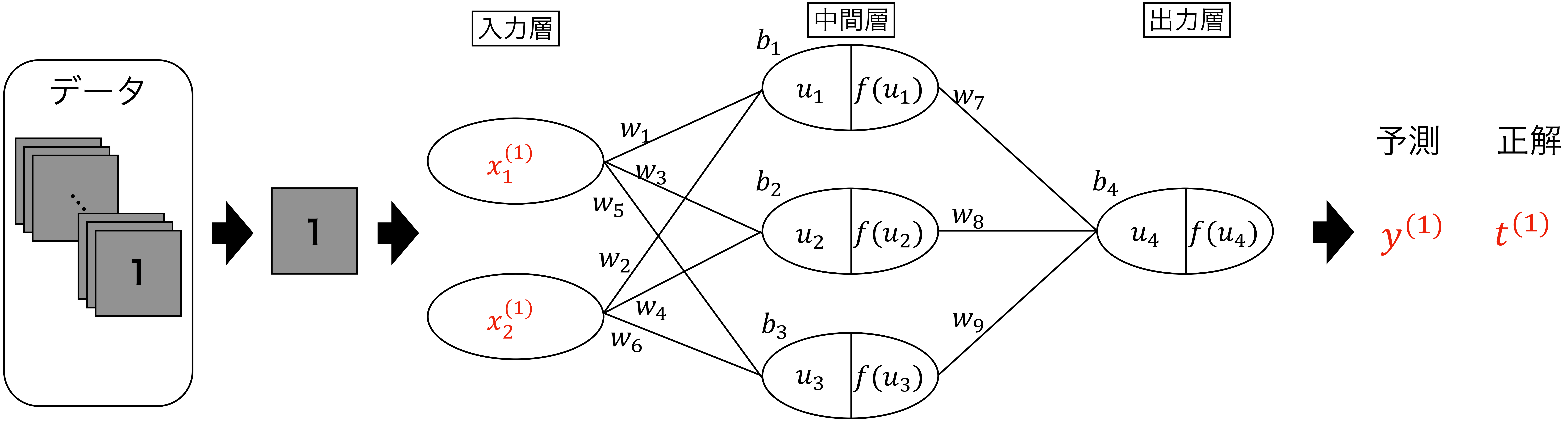
推論



出力層の$f(u_4)$ が予測結果 y になります。
また犬=1、猫=0など事前に正解 t は分かっています。

$$y = f(u_4)$$

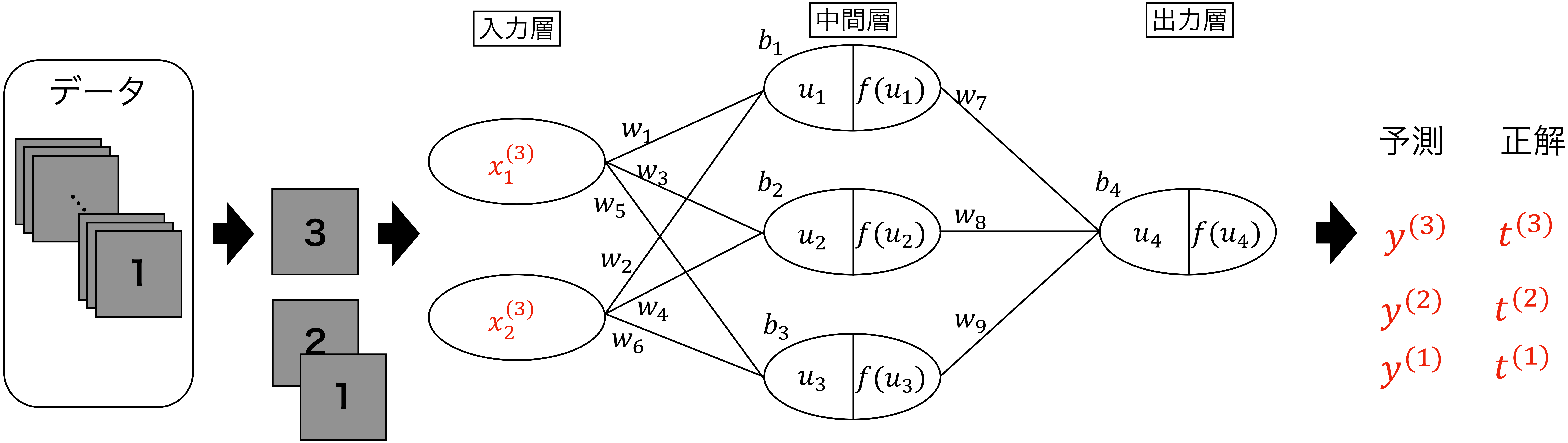
推論



これで一枚目の予測結果 y が得られました（正解 t は最初から既知）

二枚目と区別するために一枚目を $y^{(1)}$ のように書くことにします

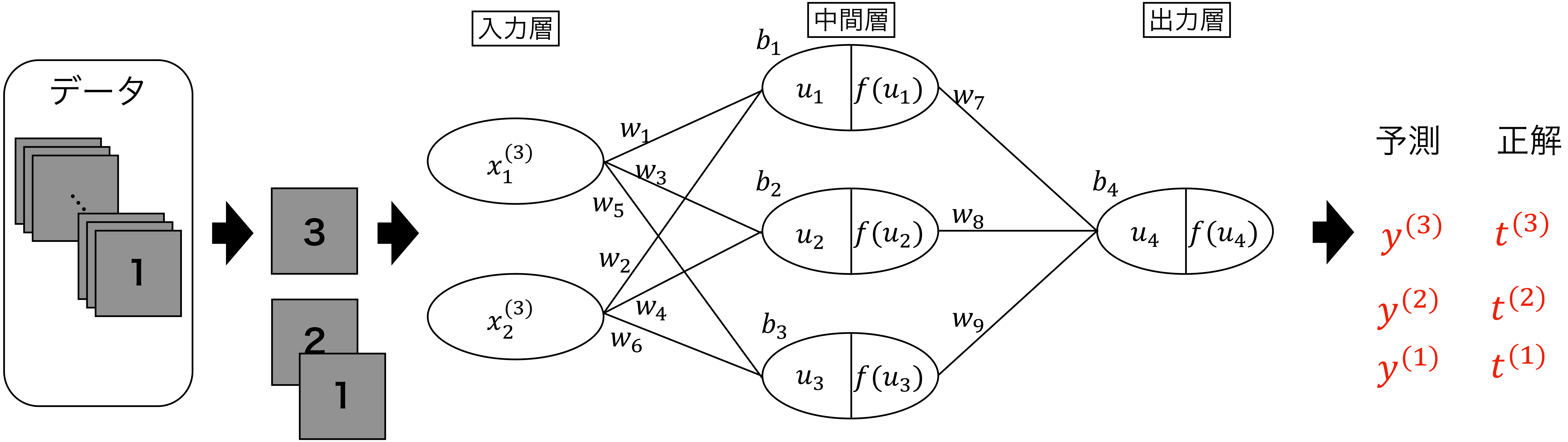
推論



二枚目、三枚目も同様に計算できます

ここまでの流れが推論です。ここから精度を調べて学習をします。

学習



最初は重み$w_i(i = 1,2, \dots 9)$ とバイアス$b_i(i = 1, \dots, 4)$ はランダムな数字なので
精度は低いものになります。
精度が高くなるように重みとバイアスの値を変えていく作業が学習です。

損失関数

予測	正解
$y^{(3)}$	$t^{(3)}$
$y^{(2)}$	$t^{(2)}$
$y^{(1)}$	$t^{(1)}$

精度を調べるために予測と正解の誤差を求めます
この誤差を計算するための関数を損失関数 E と言います

下の式は損失関数の1つである二乗和誤差の式になります

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

損失関数

誤差について

①完璧

出力	正解	差
1	1	0
1	1	0
0	0	0
0	0	0

②誤差小

出力	正解	差
0.7	1	0.3
0.8	1	0.2
0.2	0	0.2
0.1	0	0.1

③誤差大

出力	正解	差
0.5	1	0.5
0.3	1	0.7
0.6	0	0.4
0.7	0	0.7

この誤差をどのように計算するか？

損失関数

（正解 – 出力）の和？

①完璧

出力	正解	正解 – 出力
1	1	0
1	1	0
0	0	0
0	0	0

②誤差小

出力	正解	正解 – 出力
0.7	1	0.3
0.8	1	0.2
0.2	0	-0.2
0.1	0	-0.1

③誤差大

出力	正解	正解 – 出力
0.4	1	0.6
0.3	1	0.7
0.6	0	-0.4
0.7	0	-0.7

この4回の誤差を計算する場合、（正解 – 出力）の和にすると

①： $0 + 0 + 0 + 0 = 0$
②：$0.3 + 0.2 + (-0.2) + (-0.1) = 0.2$
③：$0.6 + 0.7 + (-0.4) + (-0.7) = 0.2$

となり、②と③の誤差が等しくなってしまう（ので不適当）

損失関数

２乗すると誤差を正しく評価出来る

①完璧

出力	正解	正解 – 出力
1	1	0
1	1	0
0	0	0
0	0	0

②誤差小

出力	正解	正解 – 出力
0.7	1	0.3
0.8	1	0.2
0.2	0	-0.2
0.1	0	-0.1

③誤差大

出力	正解	正解 – 出力
0.4	1	0.6
0.3	1	0.7
0.6	0	-0.4
0.7	0	-0.7

(正解 – 出力)^{2}の和にすると

①： $0^2 + 0^2 + 0^2 + 0^2 = 0$
②：$0.3^2 + 0.2^2 + (-0.2)^2 + (-0.1)^2 = 0.09 + 0.04 + 0.04 + 0.01 = 0.18$
③：$0.6^2 + 0.7^2 + (-0.4)^2 + (-0.7)^2 = 0.36 + 0.49 + 0.16 + 0.49 = 1.50$

となり、正しく誤差を① < ② < ③と評価できます。

損失関数

予測 正解

$y^{(3)}$ $t^{(3)}$

$y^{(2)}$ $t^{(2)}$

$y^{(1)}$ $t^{(1)}$

予測 y	正解 t
0.3	0
0.8	1
0.2	0

精度を調べるために予測と正解の誤差を求めます
この誤差を計算するための関数を損失関数 E と言います

下の式は損失関数の1つである二乗和誤差の式になります

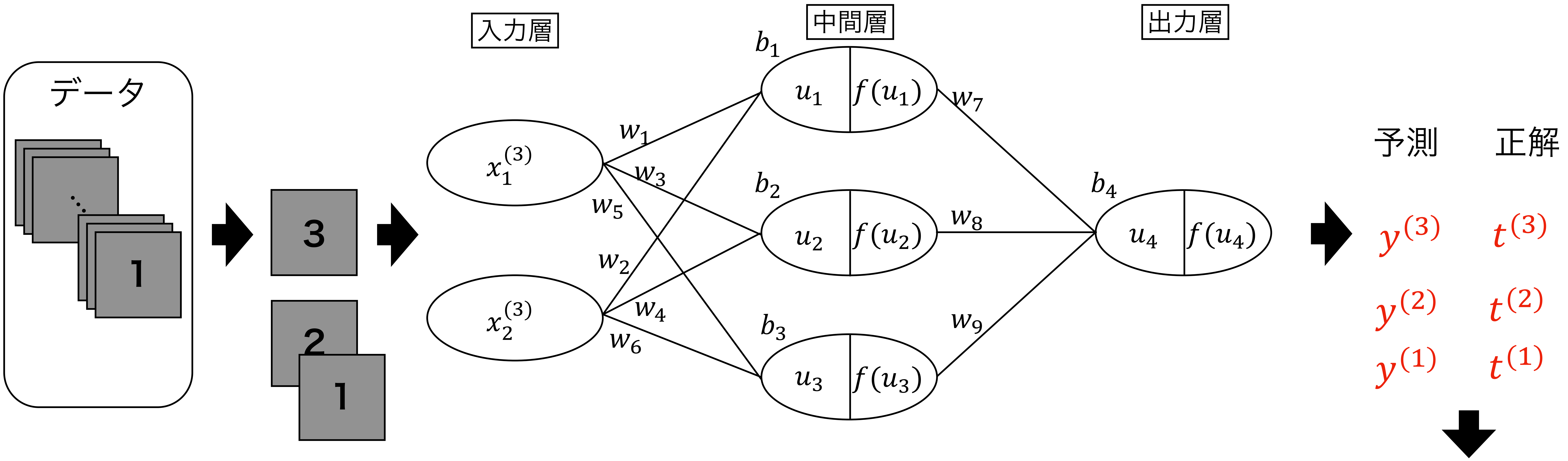
$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

仮に左のような3枚の画像の予測と正解が得られたとすると、

$$\begin{aligned} E &= \frac{1}{2}((0.3-0)^2 + (0.8-1)^2 + (0.2-0)^2) \\ &= \frac{1}{2}(0.09 + 0.04 + 0.04) = 0.085 \end{aligned}$$

となり、誤差は0.085となります

損失関数



$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})$$

つまり、学習して精度をよくするには、

E が小さくなるように重みとバイアスを変える作業になります。

損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

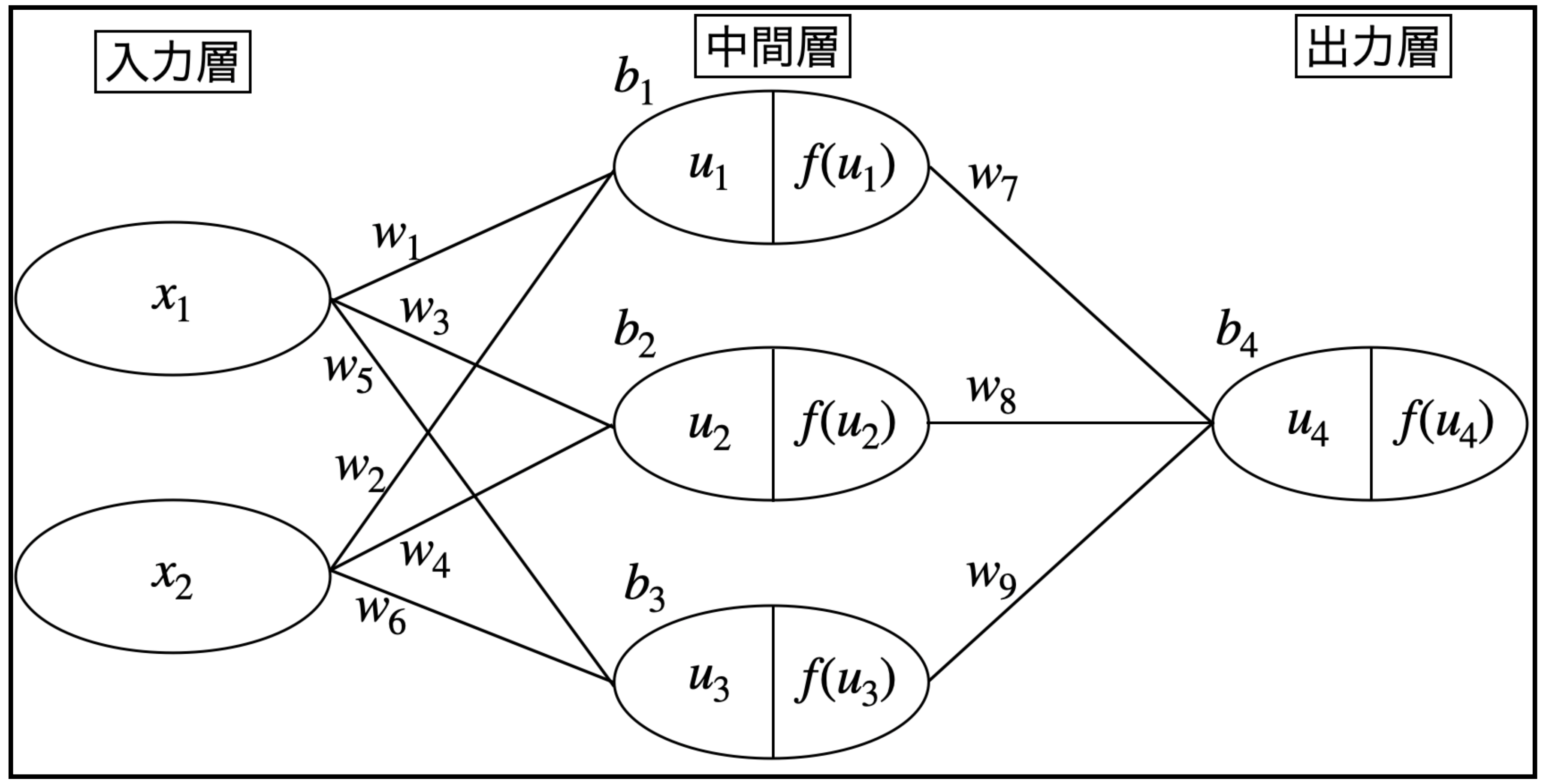
損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

推論でやったように、$y^{(k)}$はモデル内の全ての重みとバイアスで表せます



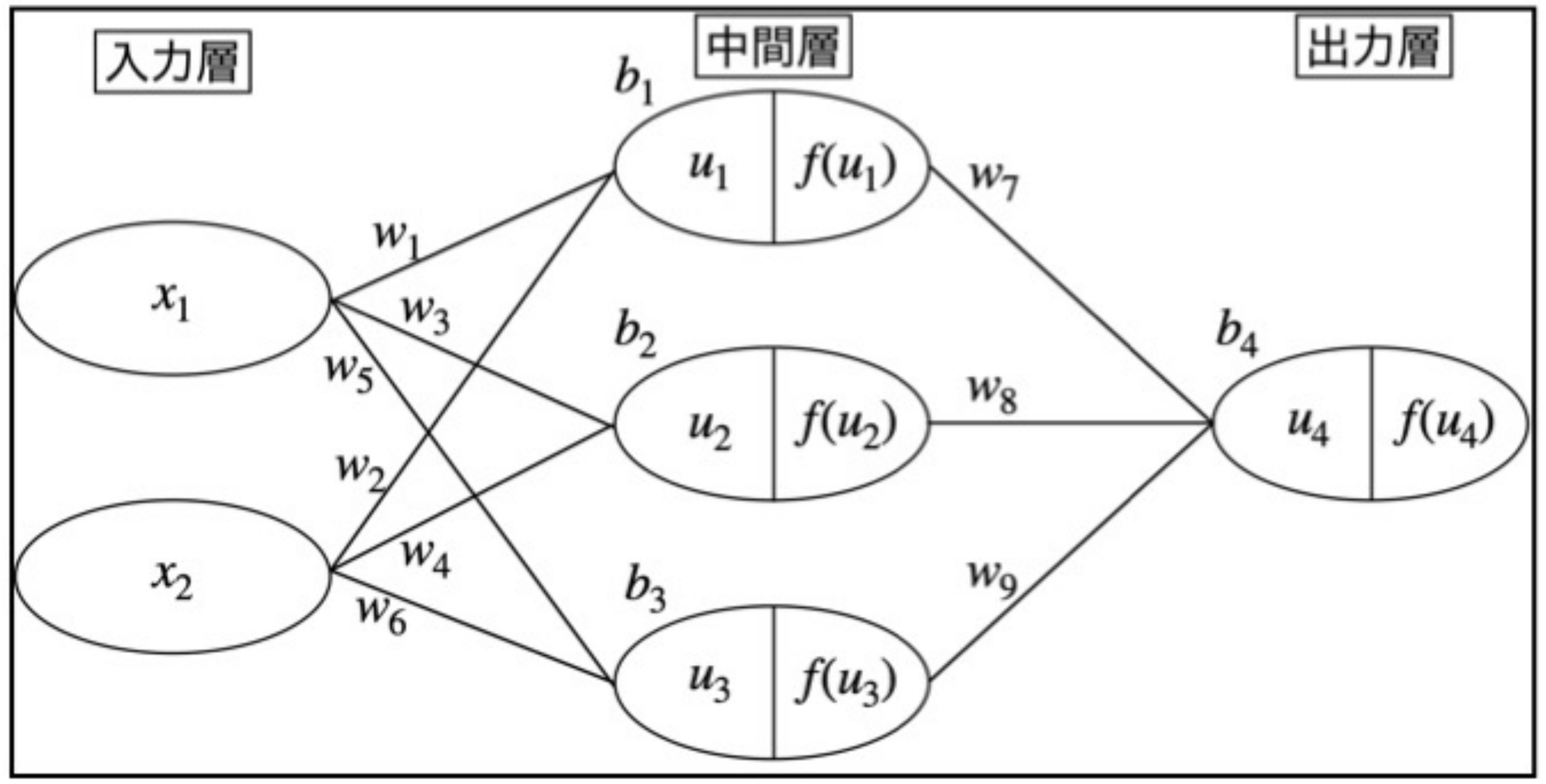
損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$ は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

推論でやったように、$y^{(k)}$ はモデル内の全ての重みとバイアスで表せます



$$y = f(u_4)$$

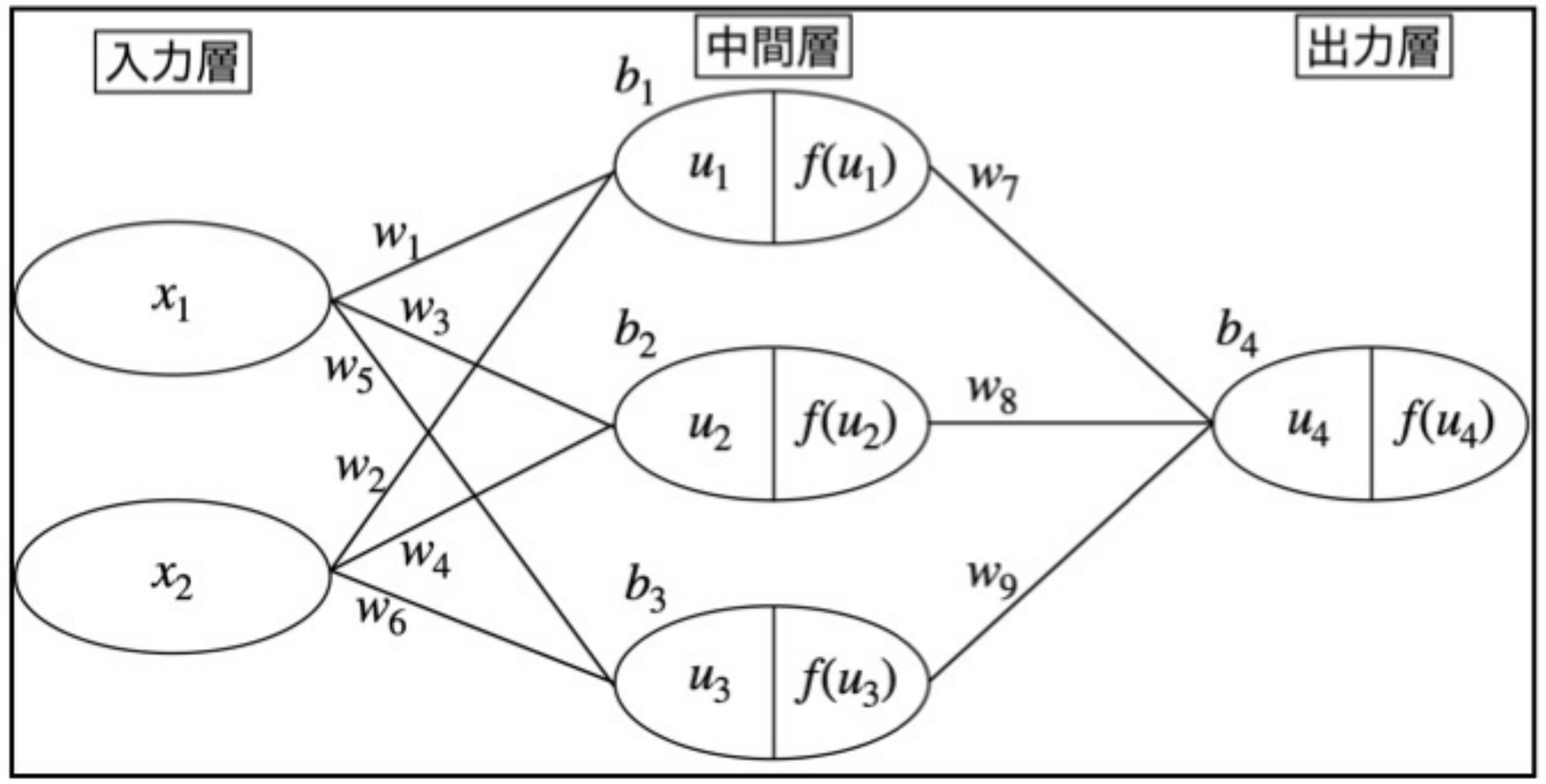
損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$ は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

推論でやったように、$y^{(k)}$ はモデル内の全ての重みとバイアスで表せます



$$y = f(u_4)$$

$$u_4 = w_7 f(u_1) + w_8 f(u_2) + w_9 f(u_3) + b_4$$

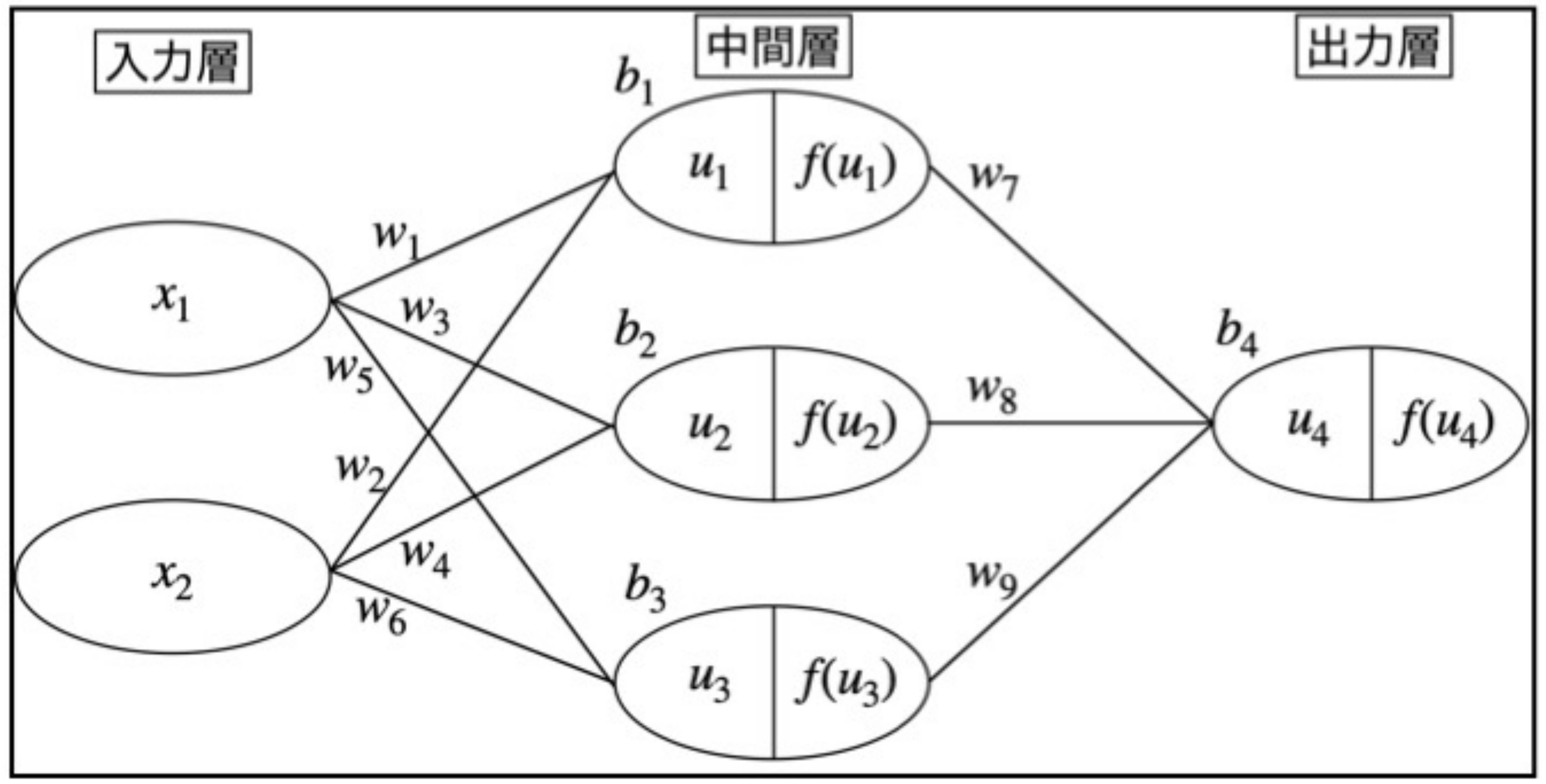
損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$ は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

推論でやったように、$y^{(k)}$ はモデル内の全ての重みとバイアスで表せます



$$y = f(u_4)$$

$$u_4 = w_7 f(u_1) + w_8 f(u_2) + w_9 f(u_3) + b_4$$

$$u_1 = w_1x_1 + w_2x_2 + b_1 \qquad f(u_1) = f(w_1x_1 + w_2x_2 + b_1)$$

$$u_2 = w_3x_1 + w_4x_2 + b_2 \qquad f(u_2) = f(w_3x_1 + w_4x_2 + b_2)$$

$$u_3 = w_5x_1 + w_6x_2 + b_3 \qquad f(u_3) = f(w_5x_1 + w_6x_2 + b_3)$$

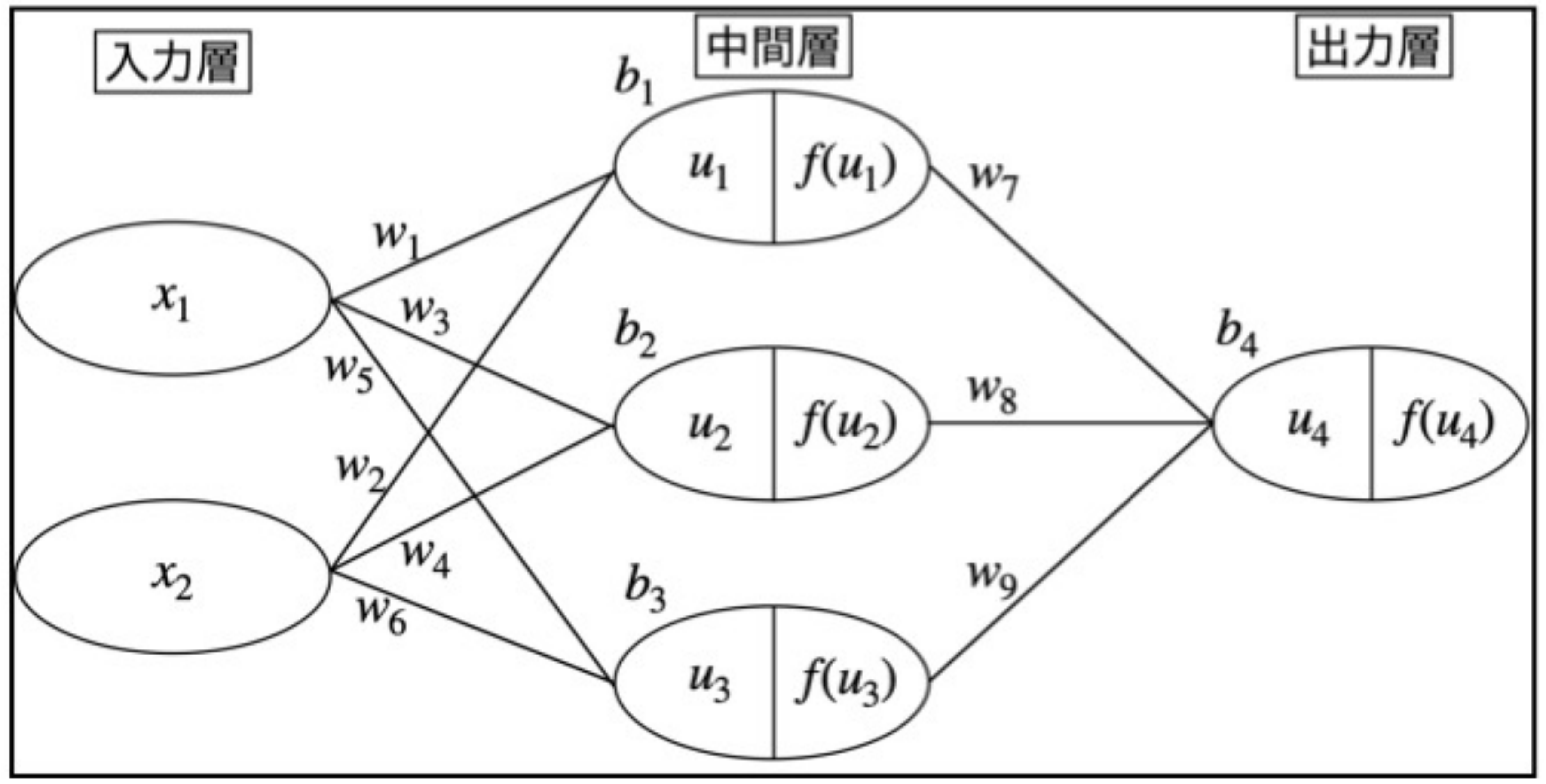
損失関数

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

引き続き二乗和誤差で考えてみます。

$t^{(k)}$ は定数(0,1など)なので、変数は予測結果 $y^{(k)}$ のみです

推論でやったように、$y^{(k)}$ はモデル内の全ての重みとバイアスで表せます



$$y = f(u_4)$$

$$u_4 = w_7 f(u_1) + w_8 f(u_2) + w_9 f(u_3) + b_4$$

$$u_1 = w_1 x_1 + w_2 x_2 + b_1 \qquad f(u_1) = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$u_2 = w_3 x_1 + w_4 x_2 + b_2 \qquad f(u_2) = f(w_3 x_1 + w_4 x_2 + b_2)$$

$$u_3 = w_5 x_1 + w_6 x_2 + b_3 \qquad f(u_3) = f(w_5 x_1 + w_6 x_2 + b_3)$$

つまり、E はモデル内の全ての重みとバイアスで表すことができます。

(= E は、重みとバイアスの関数 $E(w, b)$ である)

学習

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

E は、重みとバイアスの関数 $E(w, b)$ になる

学習

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

E は、重みとバイアスの関数 $E(w, b)$になる

重みとバイアスの値を更新して $E(w, b)$ が小さくなればよい

学習

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

E は、重みとバイアスの関数 $E(w,b)$ になる

重みとバイアスの値を更新して $E(w,b)$ が小さくなればよい

勾配降下法という手法を用います

(これまで出てきたAdamはこれを改良したものです)

学習

$$E = \frac{1}{2}\sum_{k=1}^{n}(y^{(k)} - t^{(k)})^2$$

E は、重みとバイアスの関数 $E(w, b)$になる

重みとバイアスの値を更新して $E(w, b)$ が小さくなればよい

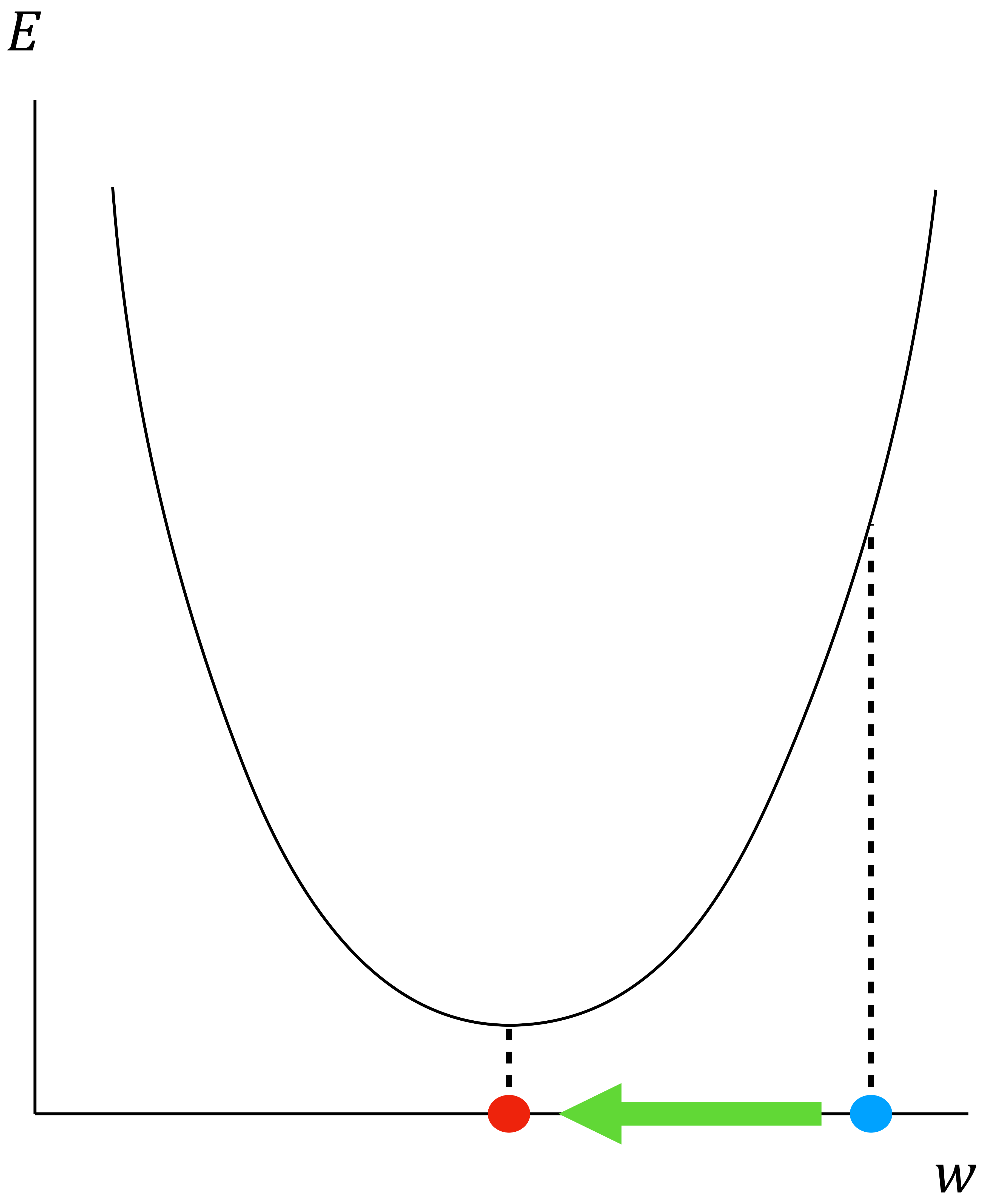
勾配降下法という手法を用います

(これまで出てきたAdamはこれを改良したものです)

$$w \leftarrow w - \alpha\frac{\partial E}{\partial w} \qquad b \leftarrow b - \alpha\frac{\partial E}{\partial b}$$

勾配降下法は上のような式によって重みとバイアスを更新します

勾配降下法

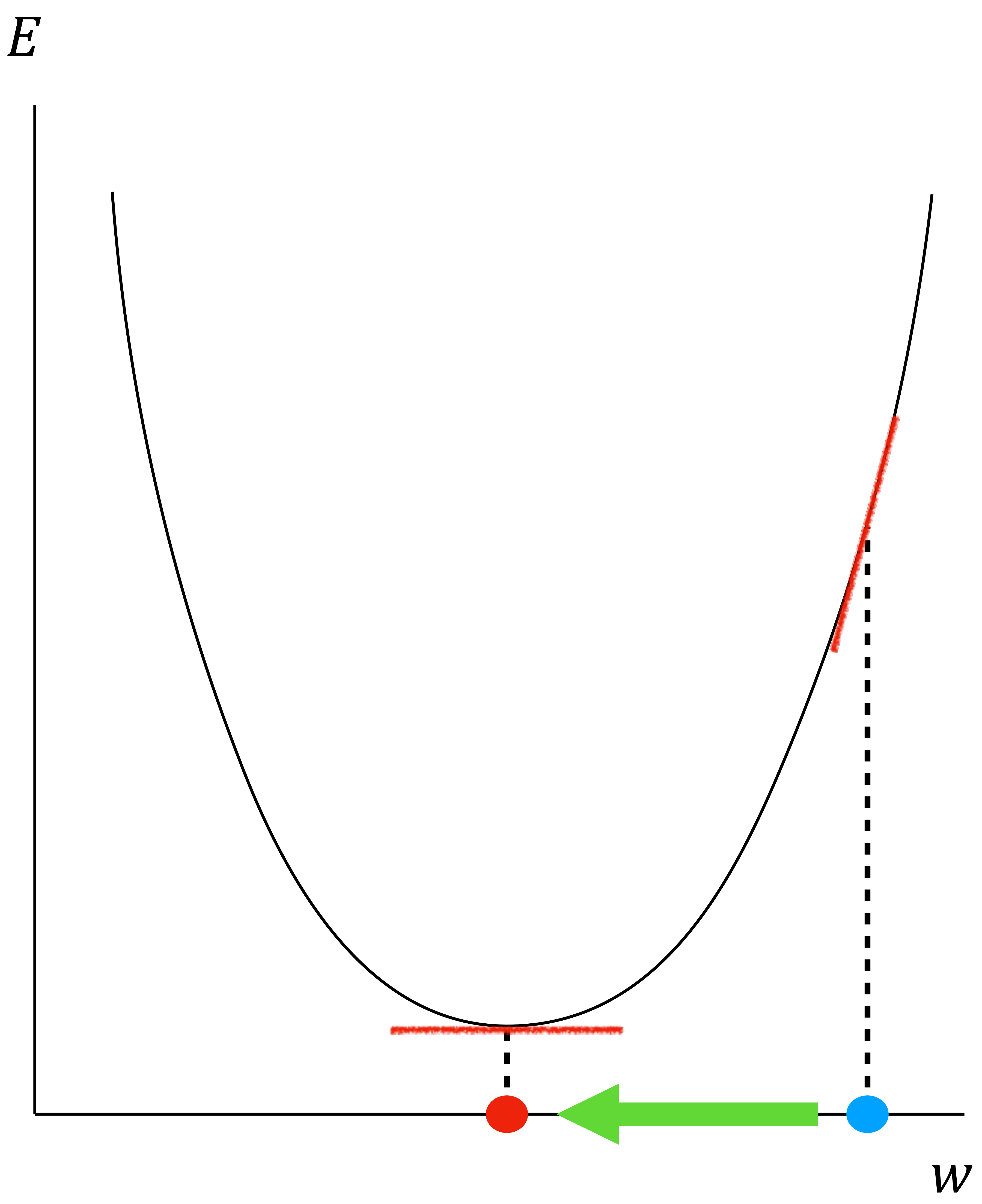


wを変えてEを小さくしたい

左の図のように縦軸Eと横軸に1つのwを考えたとき、

wを ● から ● に移動させるとEは最小になる

勾配降下法



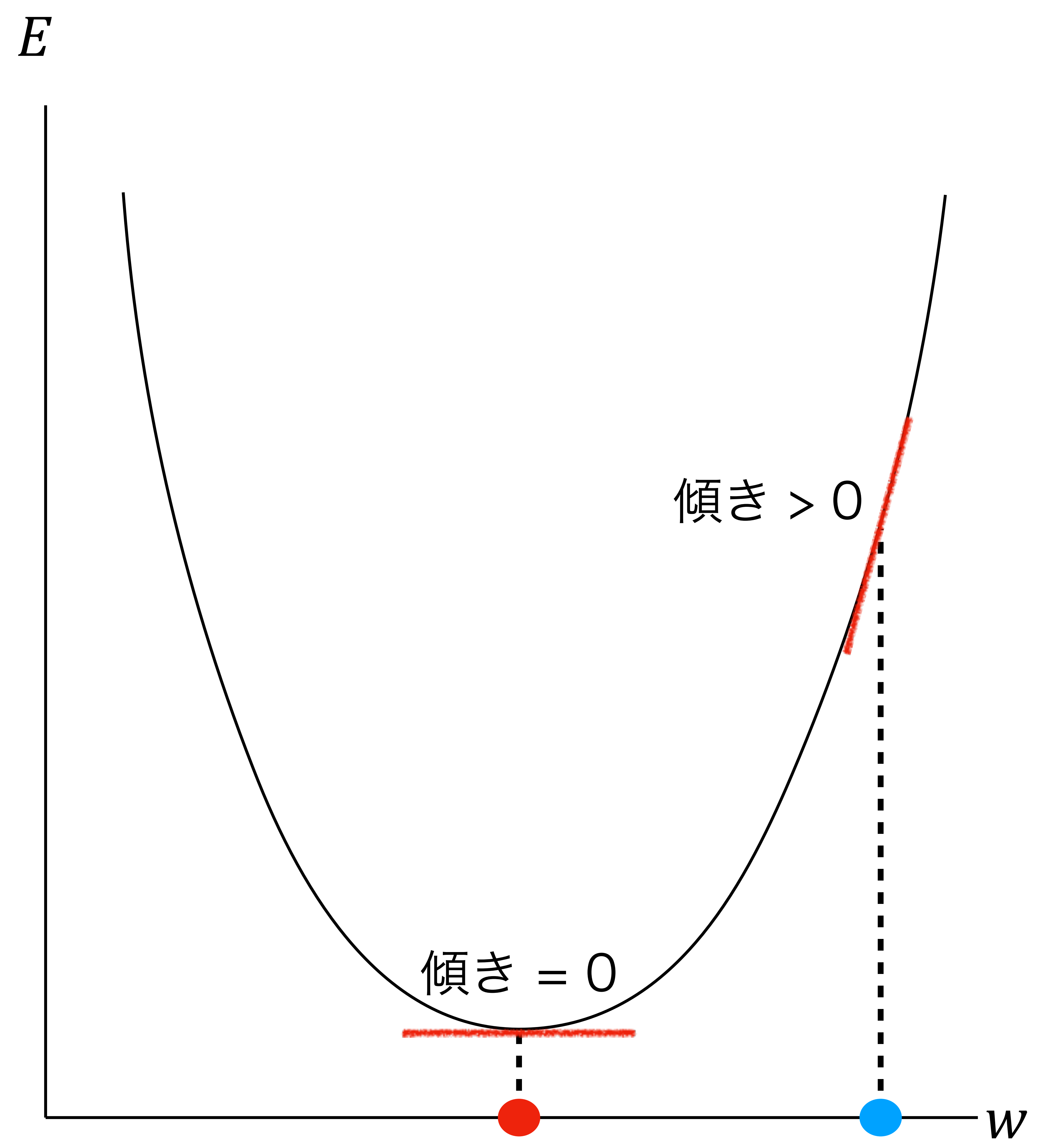
wを変えてEを小さくしたい

左の図のように縦軸Eと横軸に1つのwを考えたとき、

wを ● から ● に移動させるとEは最小になる

曲線の傾き(勾配)に着目する

勾配降下法



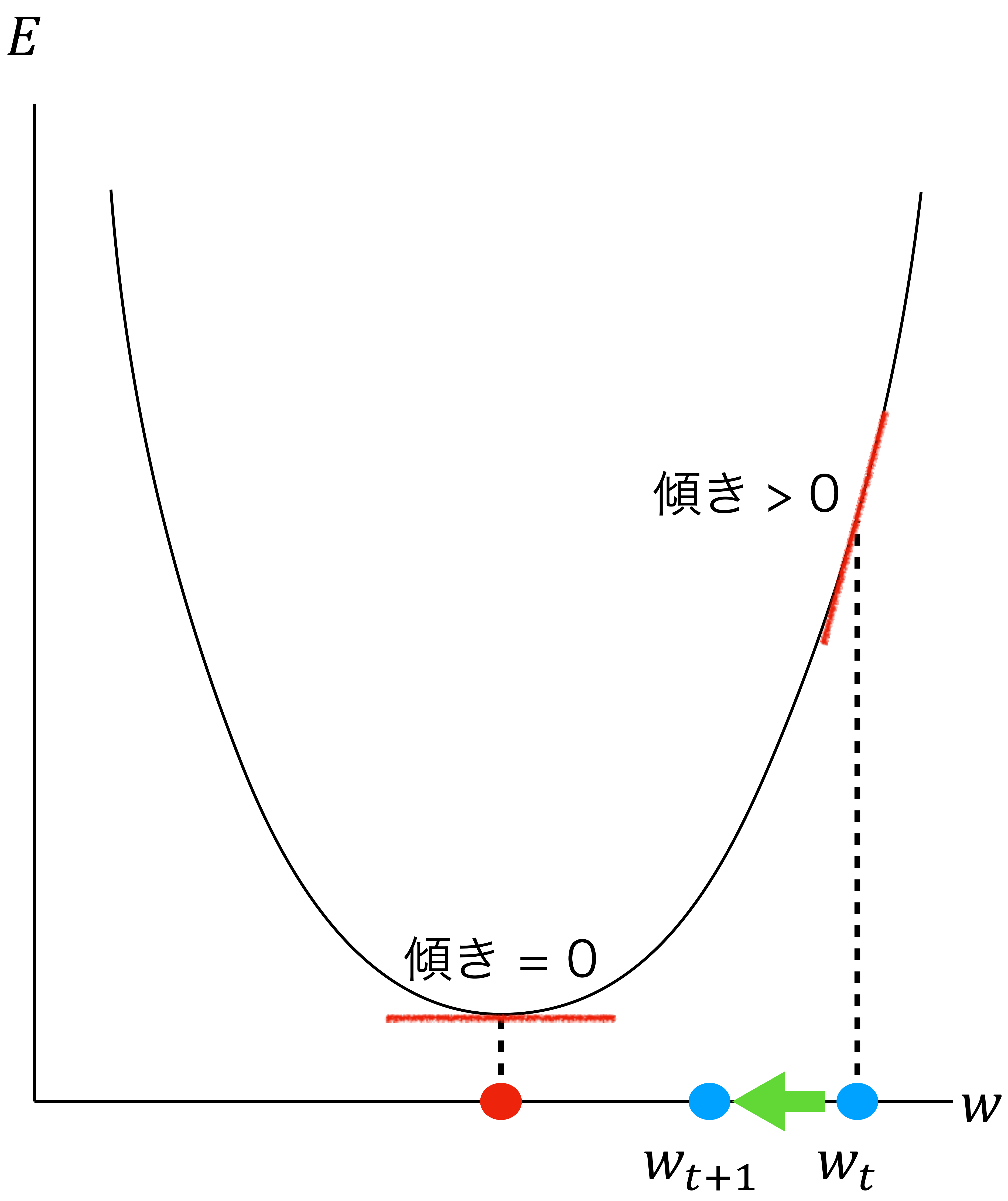
wを変えてEを小さくしたい

左の図のように縦軸Eと横軸に1つのwを考えたとき、

wを ● から ● に移動させるとEは最小になる

曲線の傾き(勾配)に着目する

勾配降下法



wを変えてEを小さくしたい

左の図のように縦軸Eと横軸に1つのwを考えたとき、

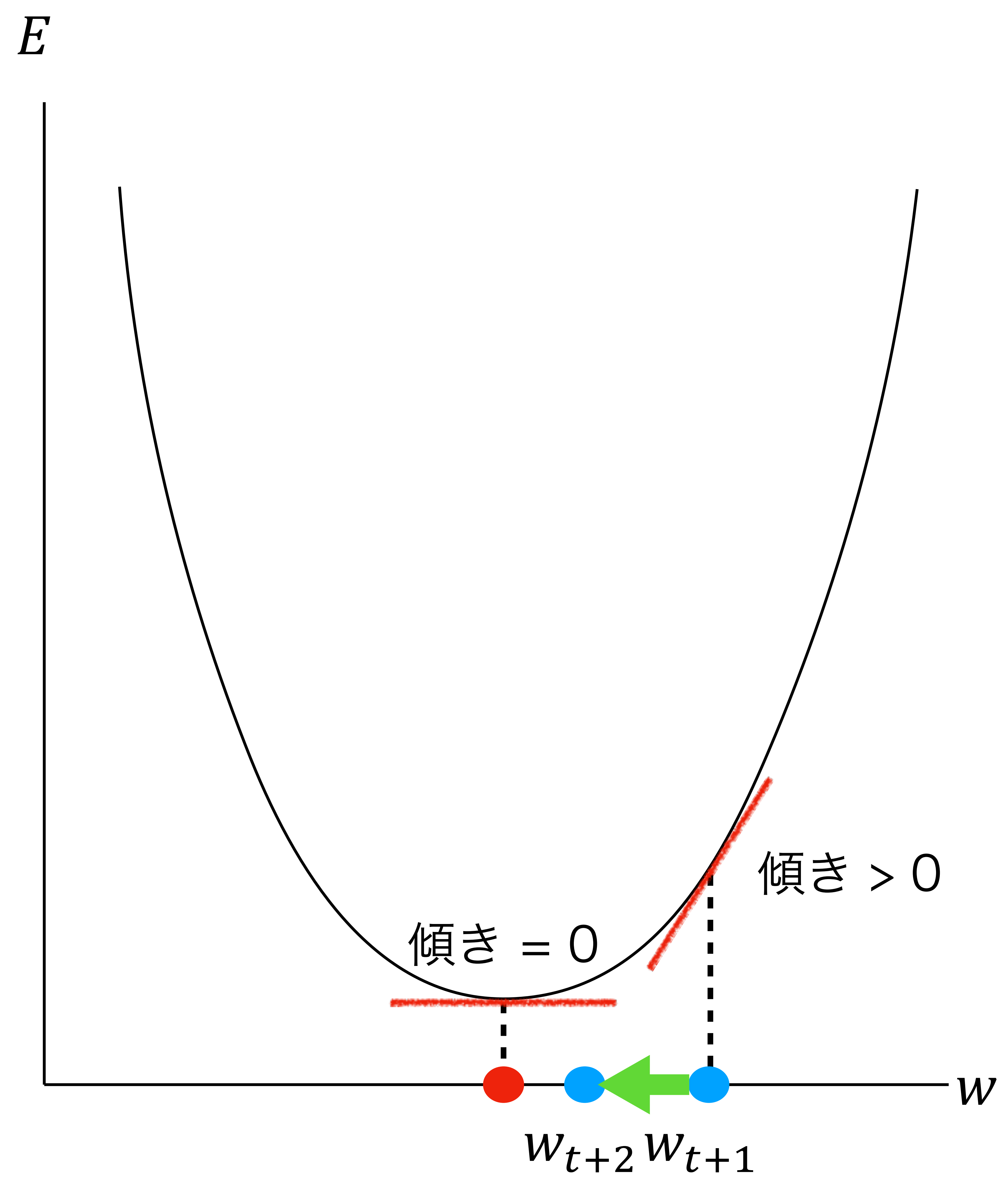
wを 🔵 から 🔴 に移動させるとEは最小になる

$\boxed{\textbf{曲線の傾き(勾配)に着目する}}$

E **の**w_t **における傾きは** $\dfrac{dE}{dw_t}$ **(偏微分で書くと**$\dfrac{\partial E}{\partial w_t}$**)**

$$w_{t+1} = w_t - \alpha \frac{\partial E}{\partial w_t}$$ （αは定数 $\alpha = 0.001$など）

勾配降下法



wを変えてEを小さくしたい

左の図のように縦軸Eと横軸に1つのwを考えたとき、

wを ● から ● に移動させるとEは最小になる

$\boxed{\text{曲線の傾き(勾配)に着目する}}$

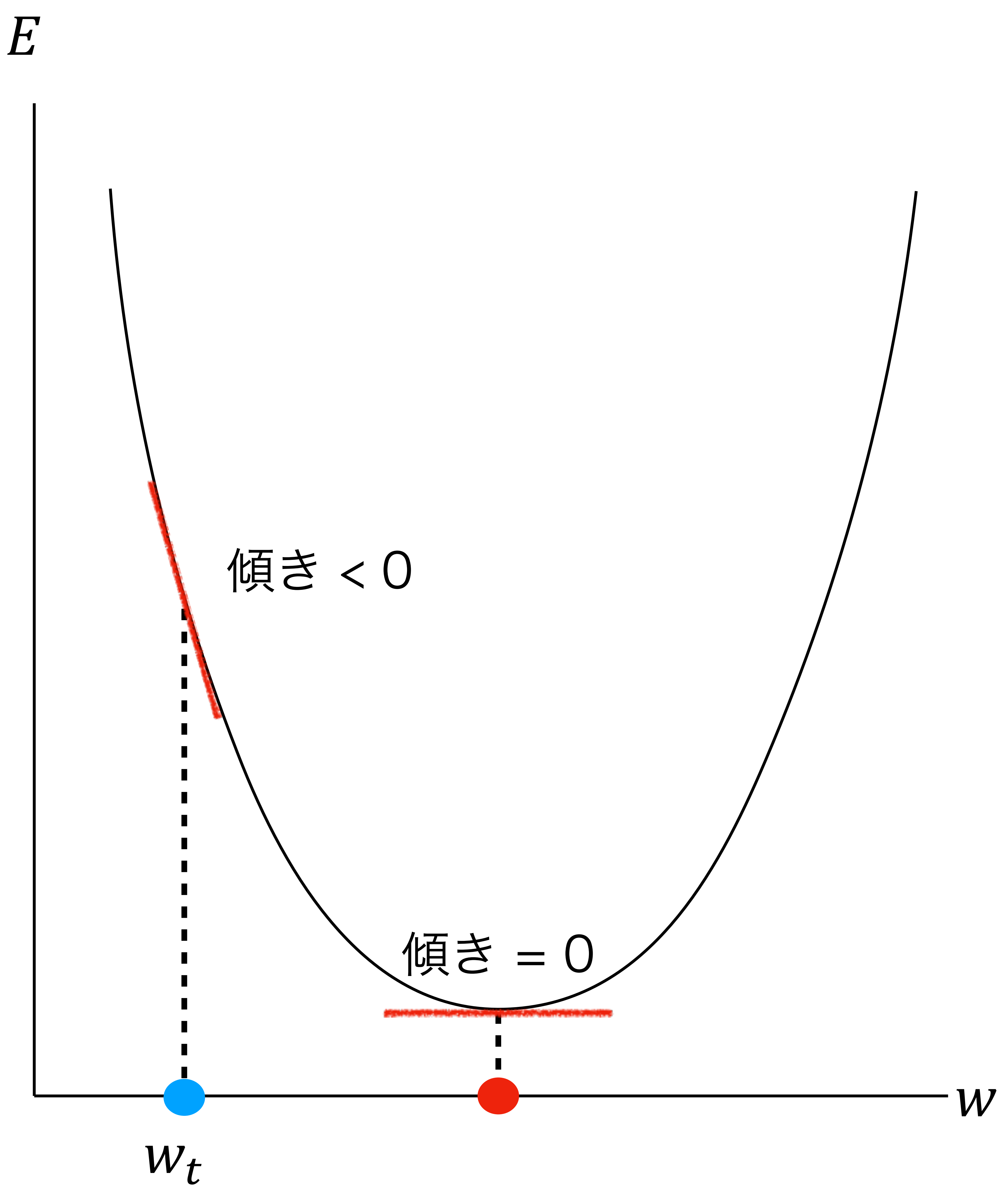
E のw_t における傾きは $\frac{dE}{dw_t}$ (偏微分で書くと$\frac{\partial E}{\partial w_t}$)

$$w_{t+1} = w_t - \alpha \frac{\partial E}{\partial w_t}$$ （αは定数 α＝0.001など）

$$w_{t+2} = w_{t+1} - \alpha \frac{\partial E}{\partial w_{t+1}}$$

w はEが小さくなる方向に更新される

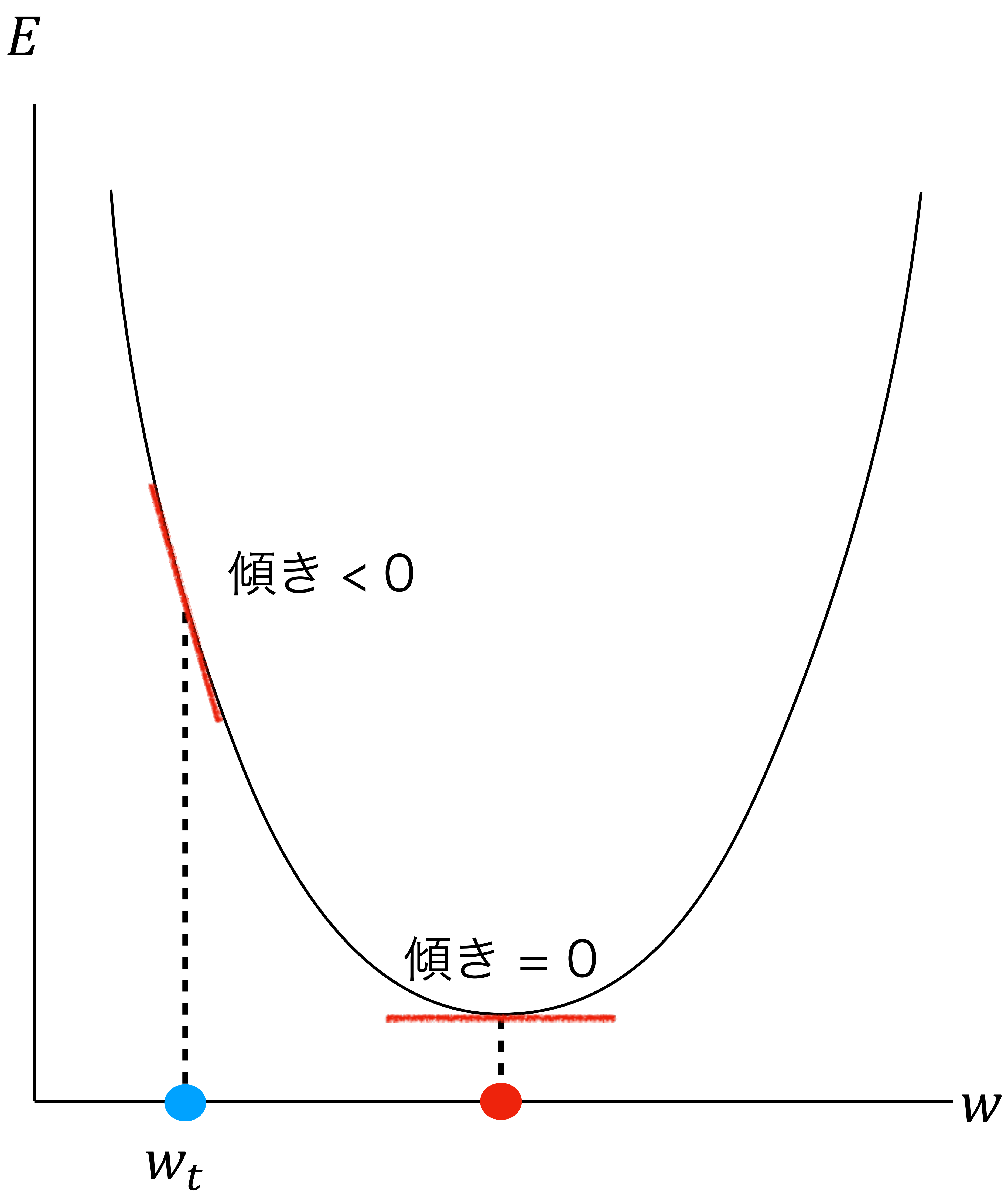
勾配降下法



傾きが負でも同様

$$w_{t+1} = w_t - \alpha \frac{\partial E}{\partial w_t}$$ （αは定数 $\alpha = 0.001$など）

勾配降下法



傾きが負でも同様

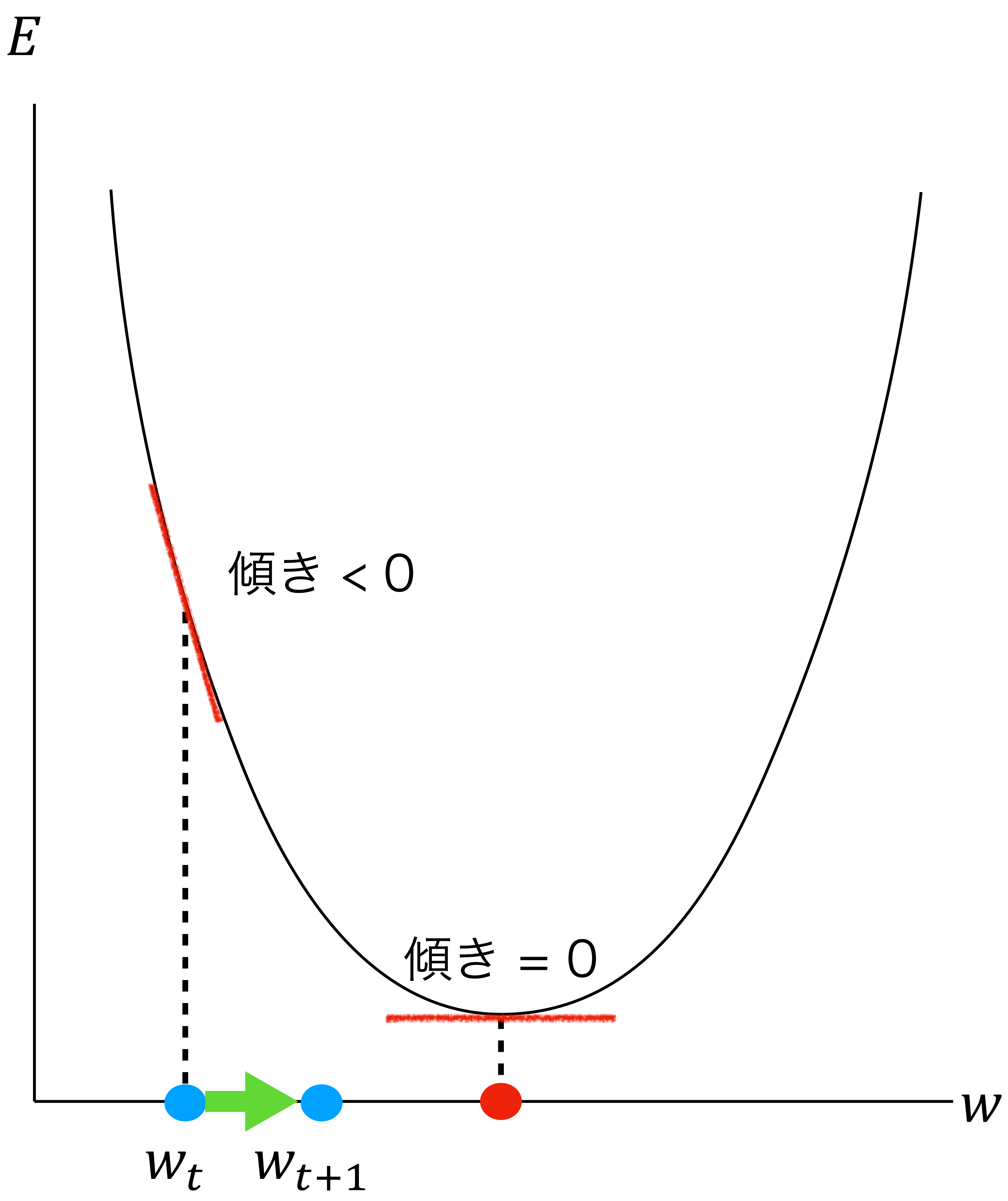
$$w_{t+1} = w_t - \alpha \frac{\partial E}{\partial w_t}$$

（αは定数 α＝0.001など）

負

正

勾配降下法



傾きが負でも同様

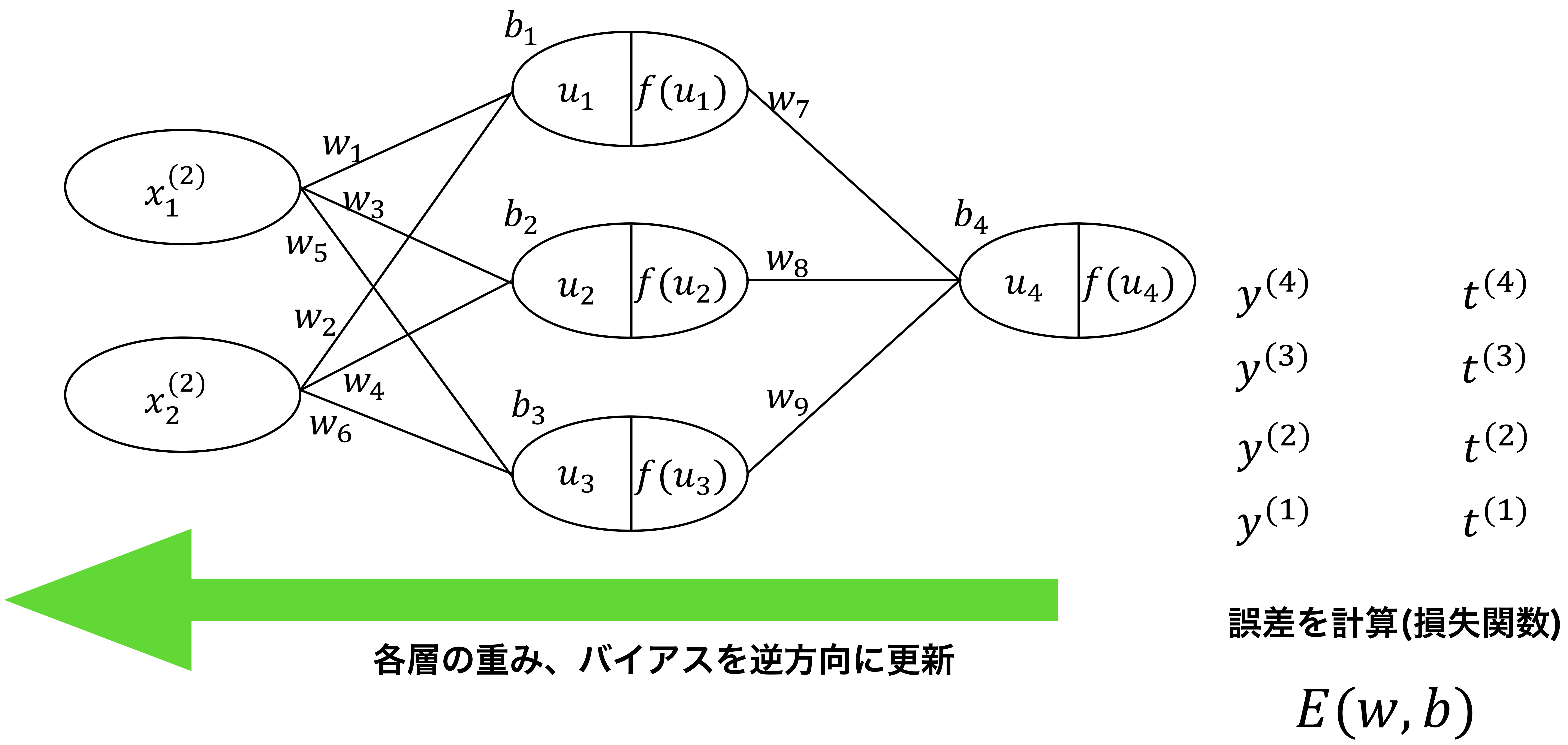
$$w_{t+1} = w_t - \alpha \frac{\partial E}{\partial w_t}$$ （αは定数 α＝0.001など）

負

正

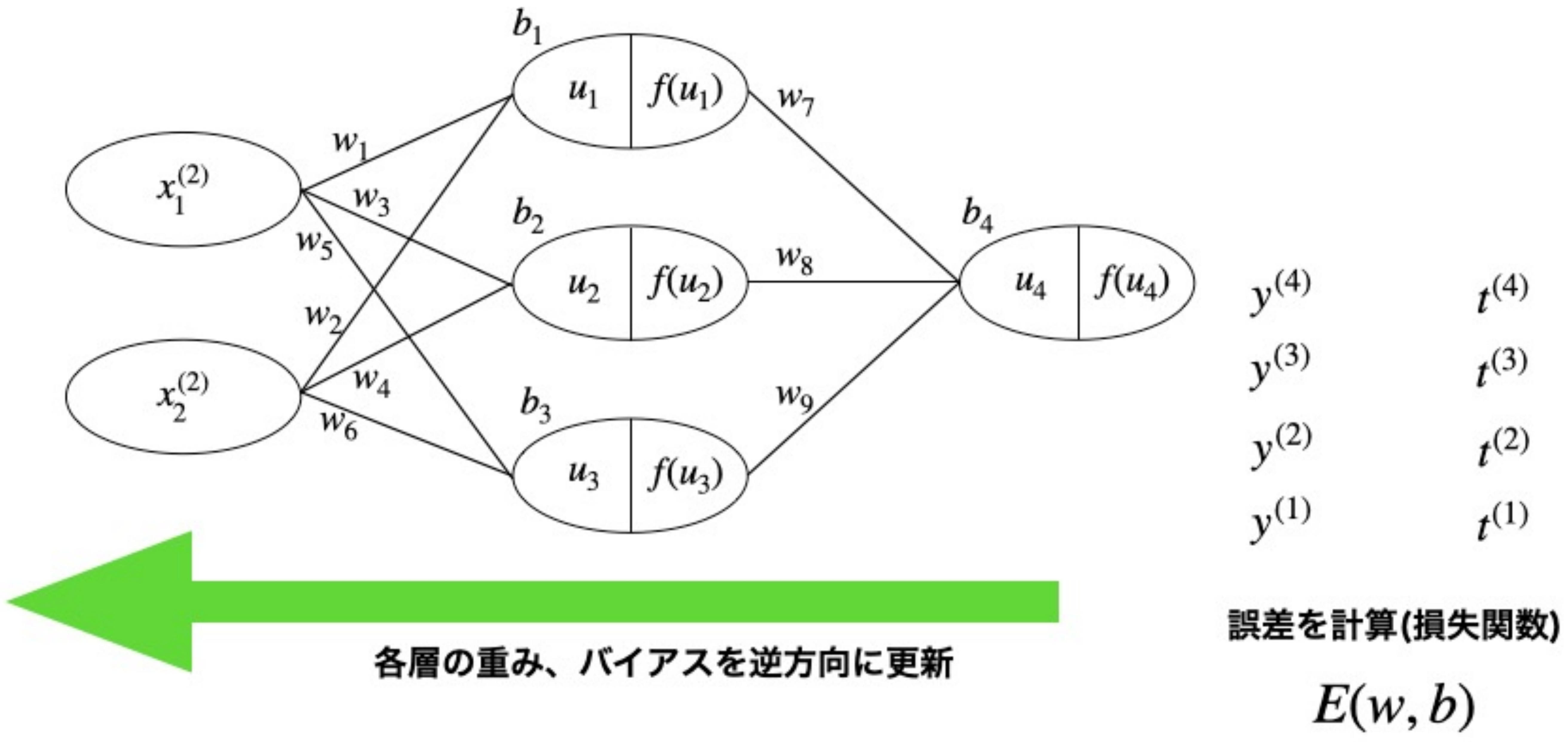
w_{t+1} **はEが小さくなる方向に更新される**

学習 ≒ 誤差逆伝播法



実際に各重みとバイアスを更新する際は、
出力層→中間層→入力層の順に計算することで求めることができます
推論と異なり逆方法に値を求めていくため、この学習の過程を誤差逆伝播法と言います。

学習 ≒ 誤差逆伝播法




```
model.compile(loss=‘categorical_crossentropy’, optimizer=‘Adam’, metrics=[‘accuracy’]
```

前回の演習ではこの損失関数にカテゴリカルクロスエントロピー関数、最適化関数に勾配降下法の改良版であるAdamを使っていました。
最適化関数は、他にもSGD(確率的勾配降下法)やRMSpropなど多くのアルゴリズムが存在します。

終わり

次回はMLPではない深層学習であるCNNに取り組みます
今回は課題はありません(出席(視聴)を確認します)