AIの発展とディープラーニング

津田塾大学
学芸学部　情報科学科
新田善久
nitta@tsuda.ac.jp
https://nw.tsuda.ac.jp

機械学習とは (Machine Learning)

☞ 機械学習とはAIの分野のひとつであり、「データを扱う正しい処理方法をコンピュータ自身がデータから学習する」技術のことです

wikipediaより

経験からの学習により自動で改善するコンピュータアルゴリズム

「経験からの学習」→データ
(訓練データ、サンプルデータ)

「自動で改善」→ 間違いを減らす

人工知能とは (AI, Artificial Intelligence)

wikipediaより

言語の理解や推論、問題解決などの知的行動を人間に代わってコンピュータに行わせる技術

人間にしかできなかった知的な行為を、どのような手段とどのようなデータを準備すれば、それを機械的に実行できるか

AI ≒　機械学習

機械学習は高校数学で理解できます

微分　　☜ 数II

関数$f(x)$の導関数は $f'(x) = \lim_{h \to 0} \dfrac{f(x+h) - f(x)}{h}$

合成関数の微分　　☜ 数III

$y = f(u), u = g(x)$ がともに微分可能なとき、

$y = f(g(x))$ も微分可能で $\dfrac{dy}{dx} = \dfrac{dy}{du} \cdot \dfrac{du}{dx}$

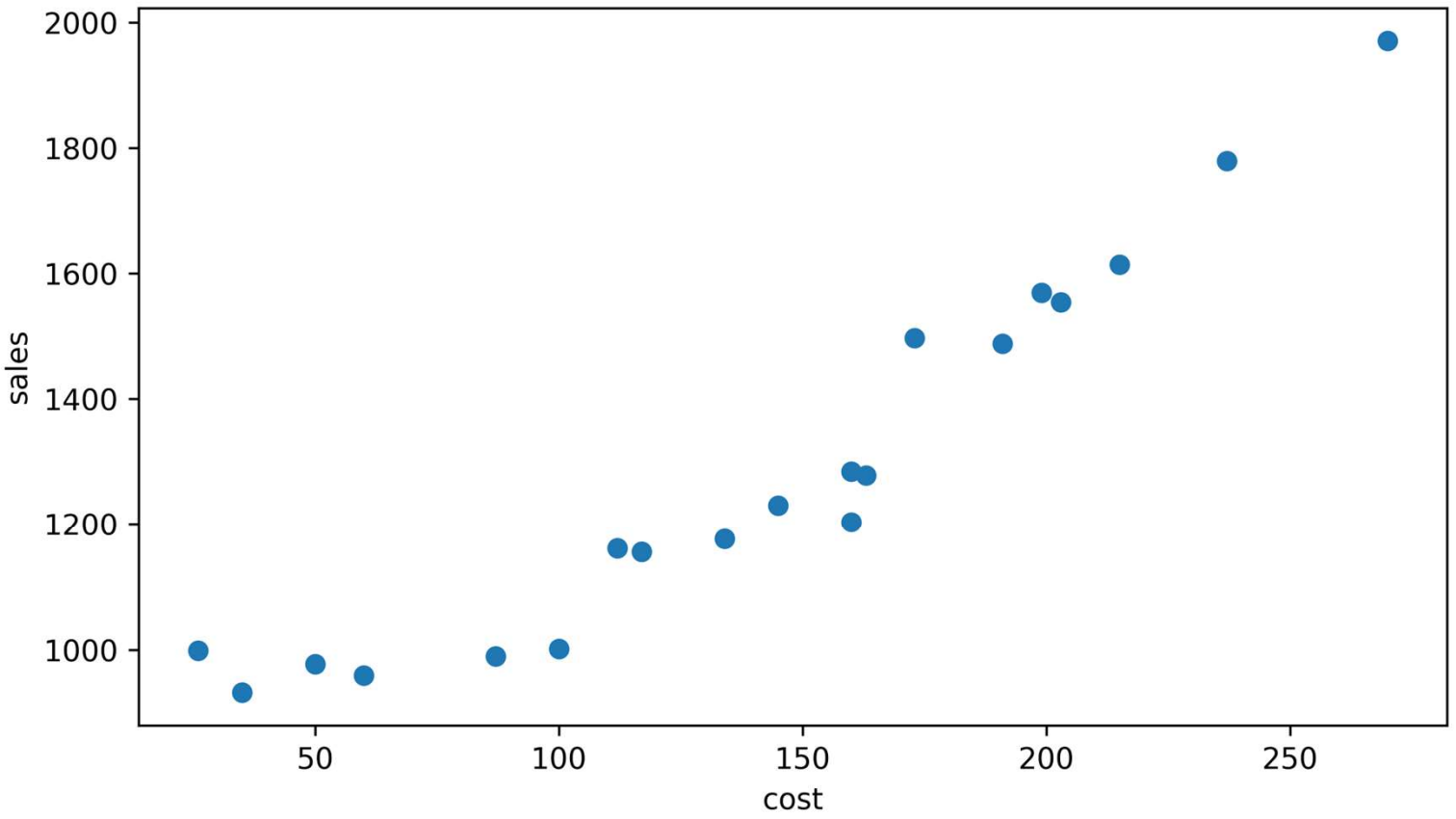
和の記号Σ　　☜ 数B

数列$\{a_n\}$の初項から第nまでの和 $a_1 + a_2 + \cdots + a_n$ を $\sum_{k=1}^{n} a_k$ と書く

【例題1】ネット広告費と売上のデータから、広告費によって売り上げを予測したい。

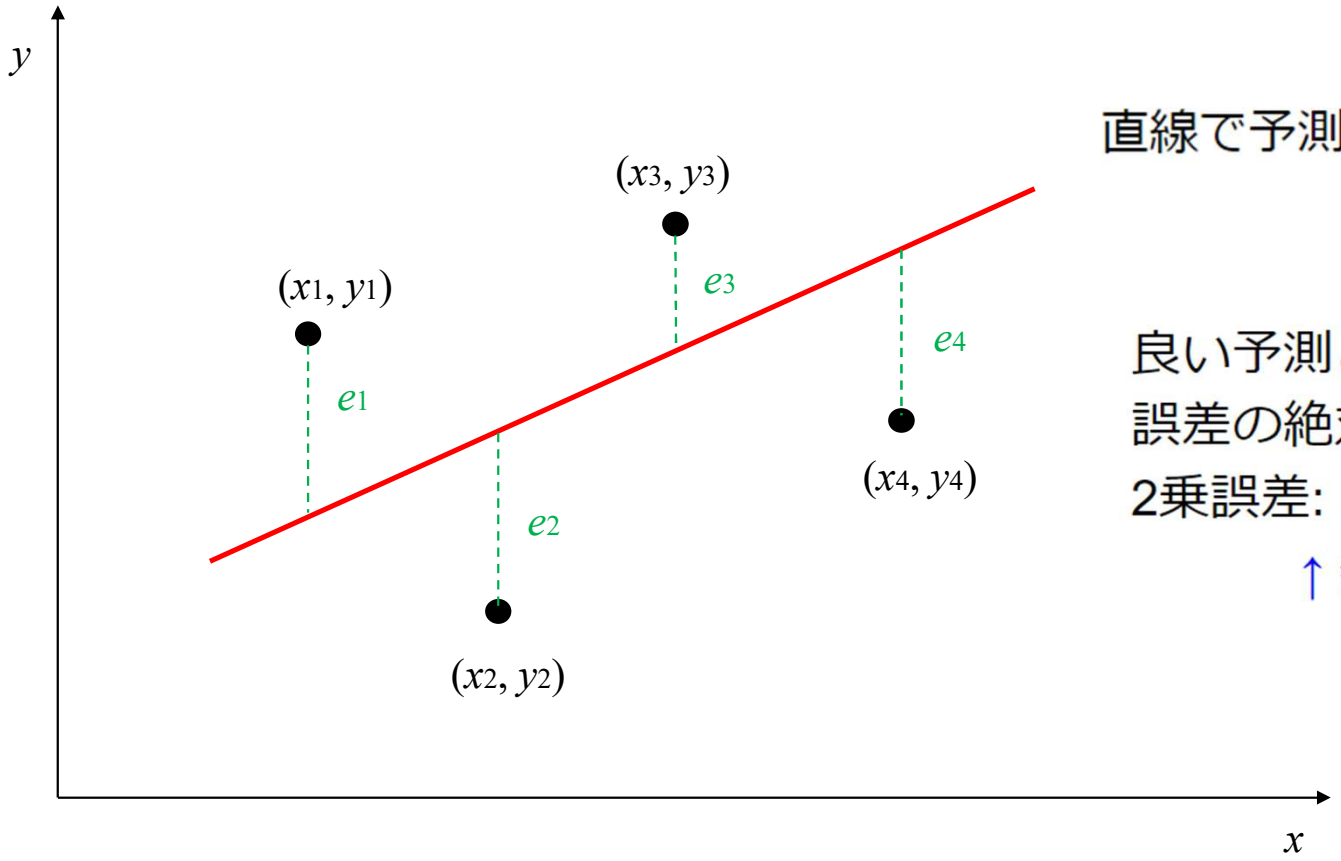
cost	173	191	134	100	117	112	35	87	203	50	26	163	270	160	160	60	199
sales	1497	1488	1176	1002	1155	1161	933	990	1554	978	999	1278	1971	1203	1284	960	1569

単位: 千円



【機械学習による解法1】直線で近似してみる

☜ 線形回帰 Linear Regression と言います



直線で予測すると $y = ax + b$

良い予測とは、誤差 J が小さいこと。

誤差の絶対値: $J = |e_1| + |e_2| + |e_3| + |e_4|$

2乗誤差: $J = e_1^2 + e_2^2 + e_3^2 + e_4^2$

↑ 数学的に扱いやすい

【機械学習による解法1】直線で近似してみる

誤差 J : 本来は、データの個数の影響を受けないようにJをデータ数4で割りますが

$$J = e_1^2 + e_2^2 + e_3^2 + e_4^2$$
$$= (y_1 - (ax_1 + b))^2 + (y_2 - (ax_2 + b))^2 + (y_3 - (ax_3 + b))^2 + (y_4 - (ax_4 + b))^2$$

誤差 J に対するaの責任 → aで微分すればわかる　数IIIっぽくdで書きます

$$\frac{d}{da}J = (-2x_1)(y_1 - (ax_1 + b)) + (-2x_2)(y_2 - (ax_2 + b)) + (-2x_3)(y_3 - (ax_3 + b)) + (-2x_4)(y_4 - (ax_4 + b))$$
$$= -2(x_1(y_1 - (ax_1 + b)) + x_2(y_2 - (ax_2 + b)) + x_3(y_3 - (ax_3 + b)) + x_4(y_4 - x_4(ax_4 + b)))$$
$$= -2\sum_{k=1}^{4} x_k(y_k - (ax_k + b))$$

誤差 J に対するbの責任 → bで微分すればわかる

$$\frac{d}{db}J = (-2)(y_1 - (ax_1 + b)) + (-2)(y_2 - (ax_2 + b)) + (-2)(y_3 - (ax_3 + b)) + (-2)(y_4 - (ax_4 + b))$$
$$= -2((y_1 - (ax_1 + b)) + (y_2 - (ax_2 + b)) + (y_3 - (ax_3 + b)) + (y_4 - x_4(ax_4 + b)))$$
$$= -2\sum_{k=1}^{4}(y_k - (ax_k + b))$$

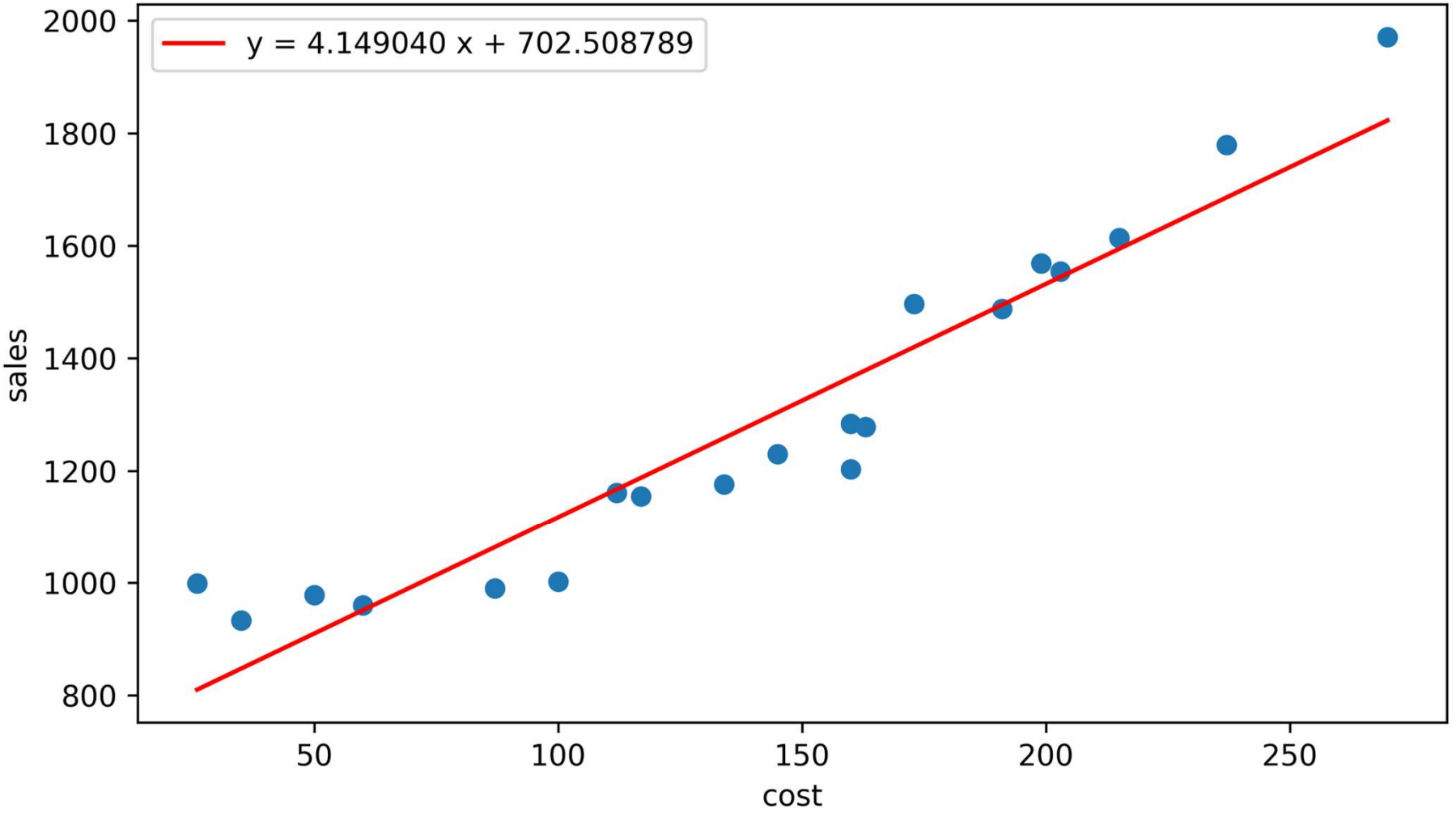
【機械学習による解法1】直線で近似してみる

適当なa, bの初期値から始めて、
繰り返しa, bを次のように変更すると
誤差Jが小さくなると期待できる

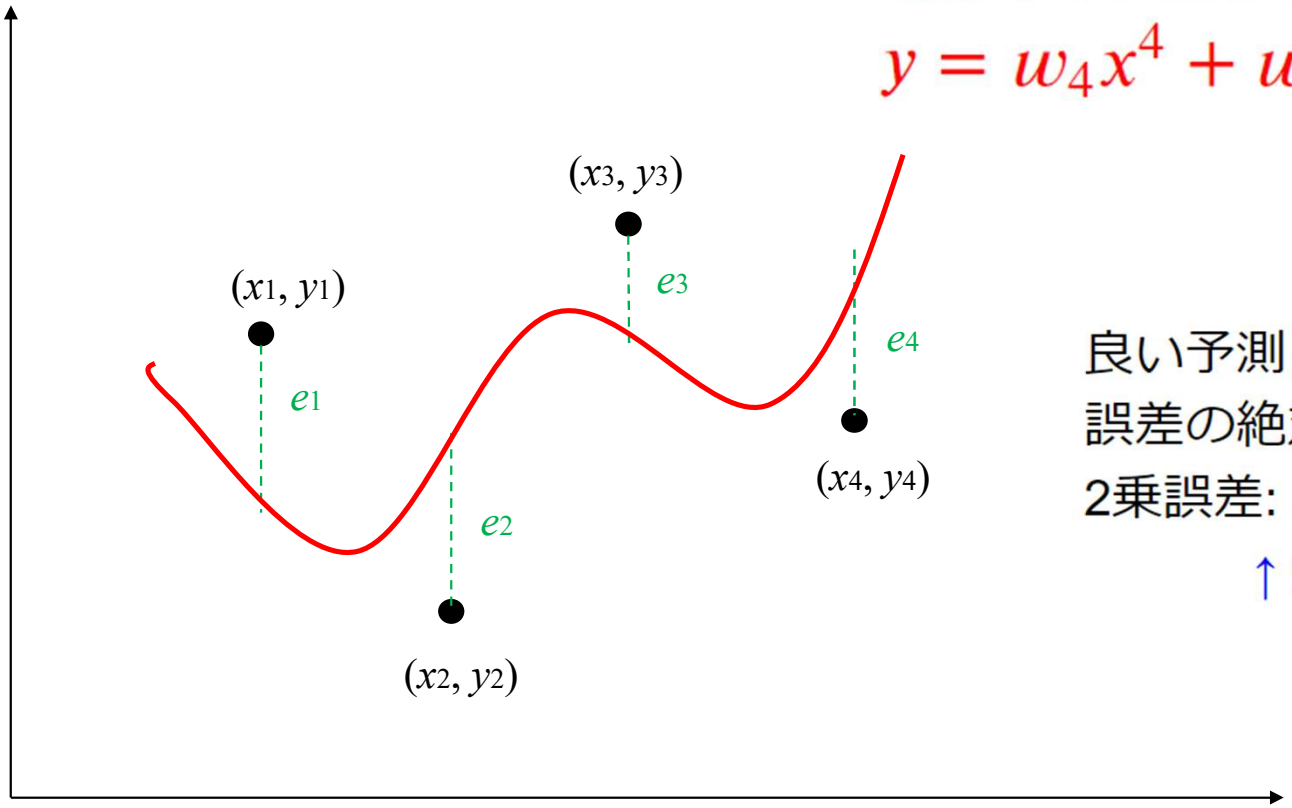
$$a := a - \text{小さい定数} \times \frac{d}{da} J$$

$$b := b - \text{小さい定数} \times \frac{d}{db} J$$

【機械学習による解法1】直線で近似してみる



【機械学習による解法2】4次式で近似してみる



4次式で予測すると

$$y = w_4x^4 + w_3x^3 + w_2x^2 + w_1x + b$$

良い予測とは、誤差 J が小さいこと。

誤差の絶対値: $J = |e_1| + |e_2| + |e_3| + |e_4|$

2乗誤差: $J = e_1^2 + e_2^2 + e_3^2 + e_4^2$

↑ 数学的に扱いやすい

【機械学習による解法2】4次式で近似してみる

誤差 J : 本来は、データの個数の影響を受けないようにJをデータ数4で割りますが

$$
\begin{aligned}
J &= e_1^2 + e_2^2 + e_3^2 + e_4^2 \\
&= (y_1 - (w_4 x_1^4 + w_3 x_1^3 + w_2 x_1^2 + w_1 x_1 + b))^2 + \cdots \\
&= \sum_{k=1}^{4} (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))^2
\end{aligned}
$$

誤差 J に対する各パラメータの責任 → そのパラメータで微分すればわかる　数IIIっぽくdで書きます

$$\frac{d}{dw_4} J = -2 \sum_{k=1}^{4} x_k^4 (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))$$

$$\frac{d}{dw_3} J = -2 \sum_{k=1}^{4} x_k^3 (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))$$

$$\frac{d}{dw_2} J = -2 \sum_{k=1}^{4} x_k^2 (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))$$

$$\frac{d}{dw_1} J = -2 \sum_{k=1}^{4} x_k (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))$$

$$\frac{d}{db} J = -2 \sum_{k=1}^{4} (y_k - (w_4 x_k^4 + w_3 x_k^3 + w_2 x_k^2 + w_1 x_k + b))$$

【機械学習による解法2】4次式で近似してみる

適当なw_4, w_3, w_2, w_1, bの初期値から始めて、
繰り返し次のように変更すると
誤差Jが小さくなると期待できる

$$w_4 := w_4 - \text{小さい定数} \times \frac{d}{dw_4} J$$

$$w_3 := w_3 - \text{小さい定数} \times \frac{d}{dw_3} J$$

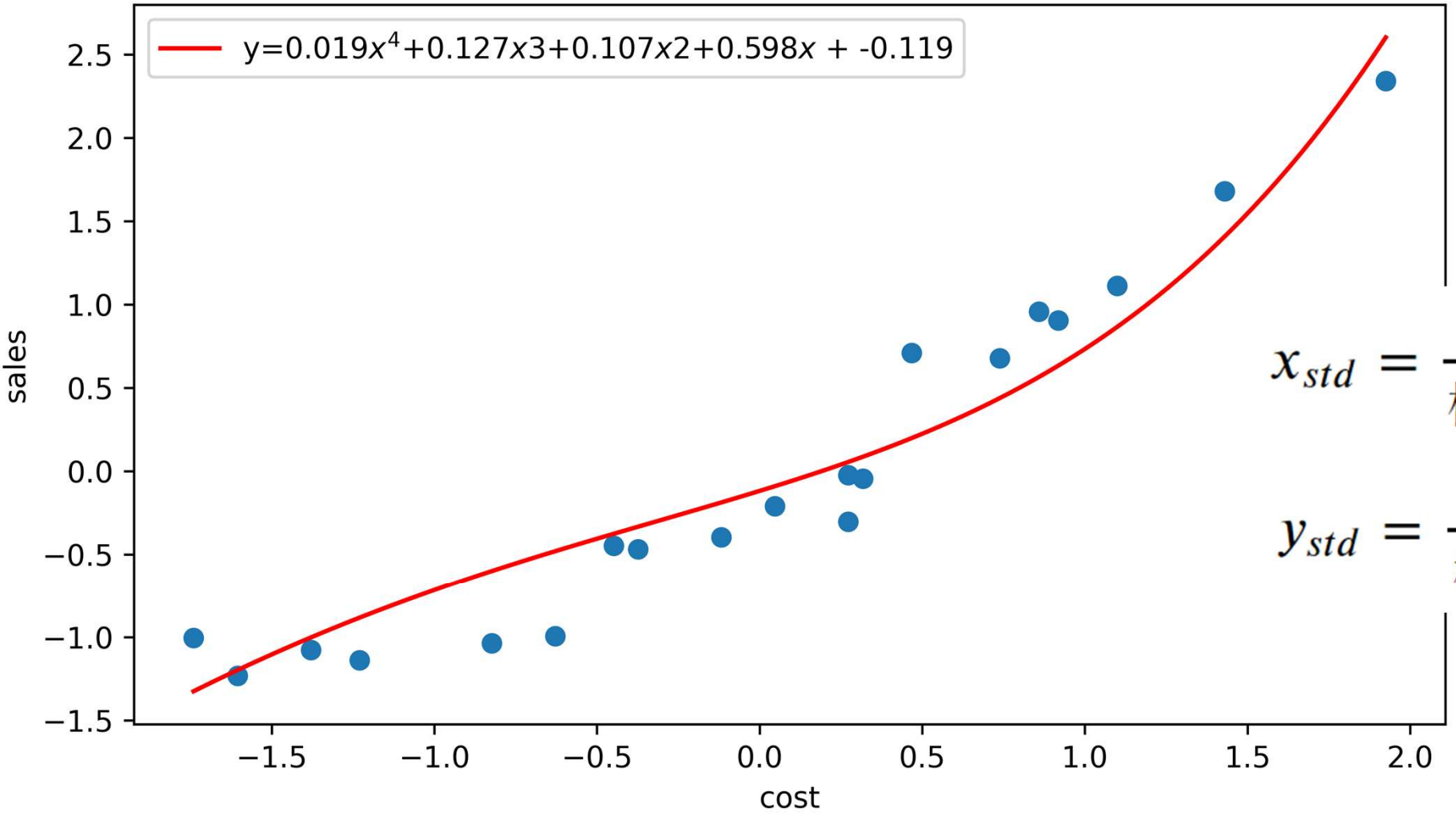
$$w_2 := w_2 - \text{小さい定数} \times \frac{d}{dw_2} J$$

$$w_1 := w_1 - \text{小さい定数} \times \frac{d}{dw_1} J$$

$$b := b - \text{小さい定数} \times \frac{d}{db} J$$

【機械学習による解法2】4次式で近似してみる

☜ 数値は標準化してから解いています

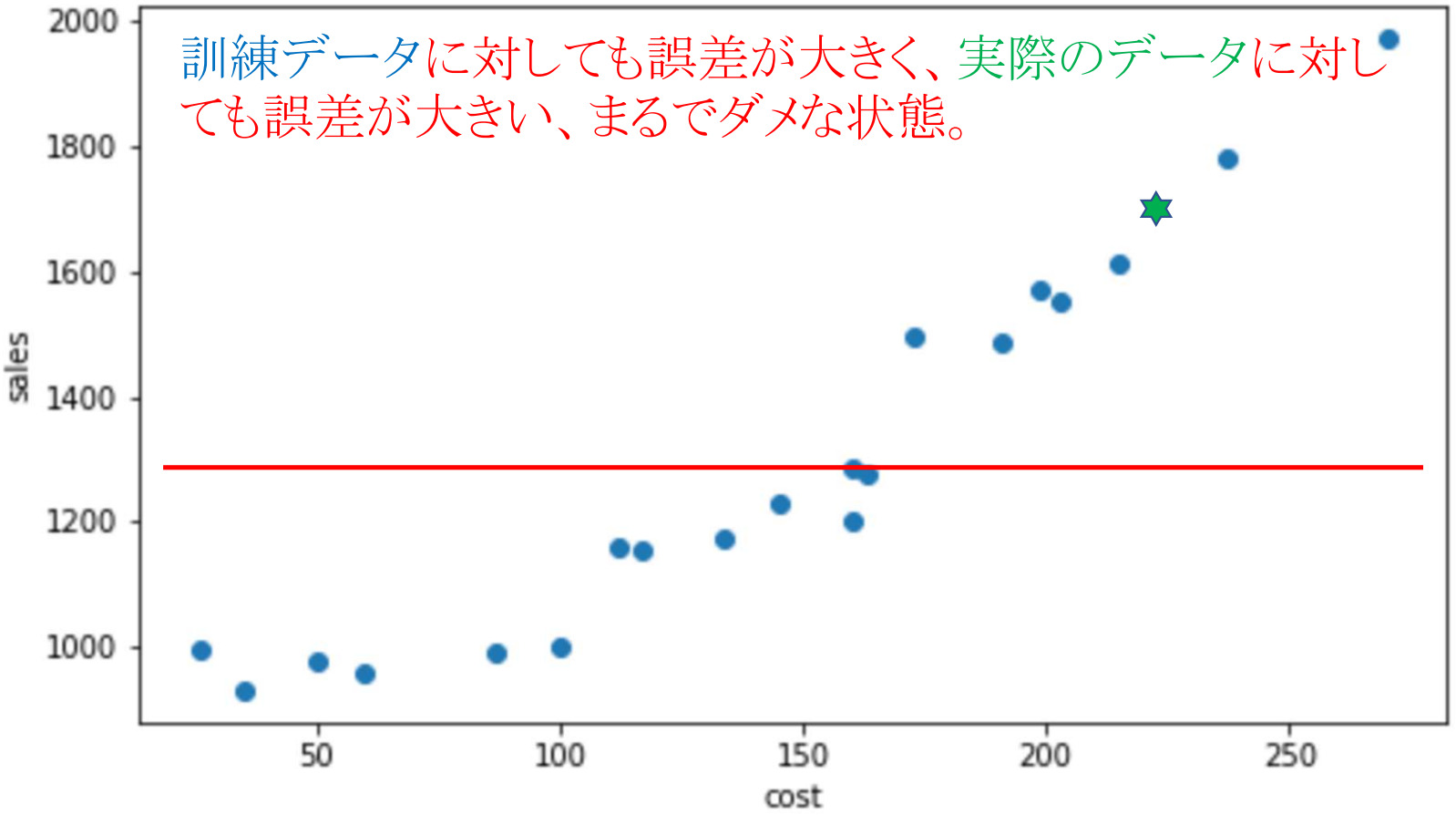


$$x_{std} = \frac{x - \text{平均}(x)}{\text{標準偏差}(x)}$$

$$y_{std} = \frac{y - \text{平均}(y)}{\text{標準偏差}(y)}$$

【考えてみよう】
【例題1】を近似するにはどんな式が適切なのだろうか？

単純すぎる式 → たとえば定数 → underfitting (過少適合、過少学習)

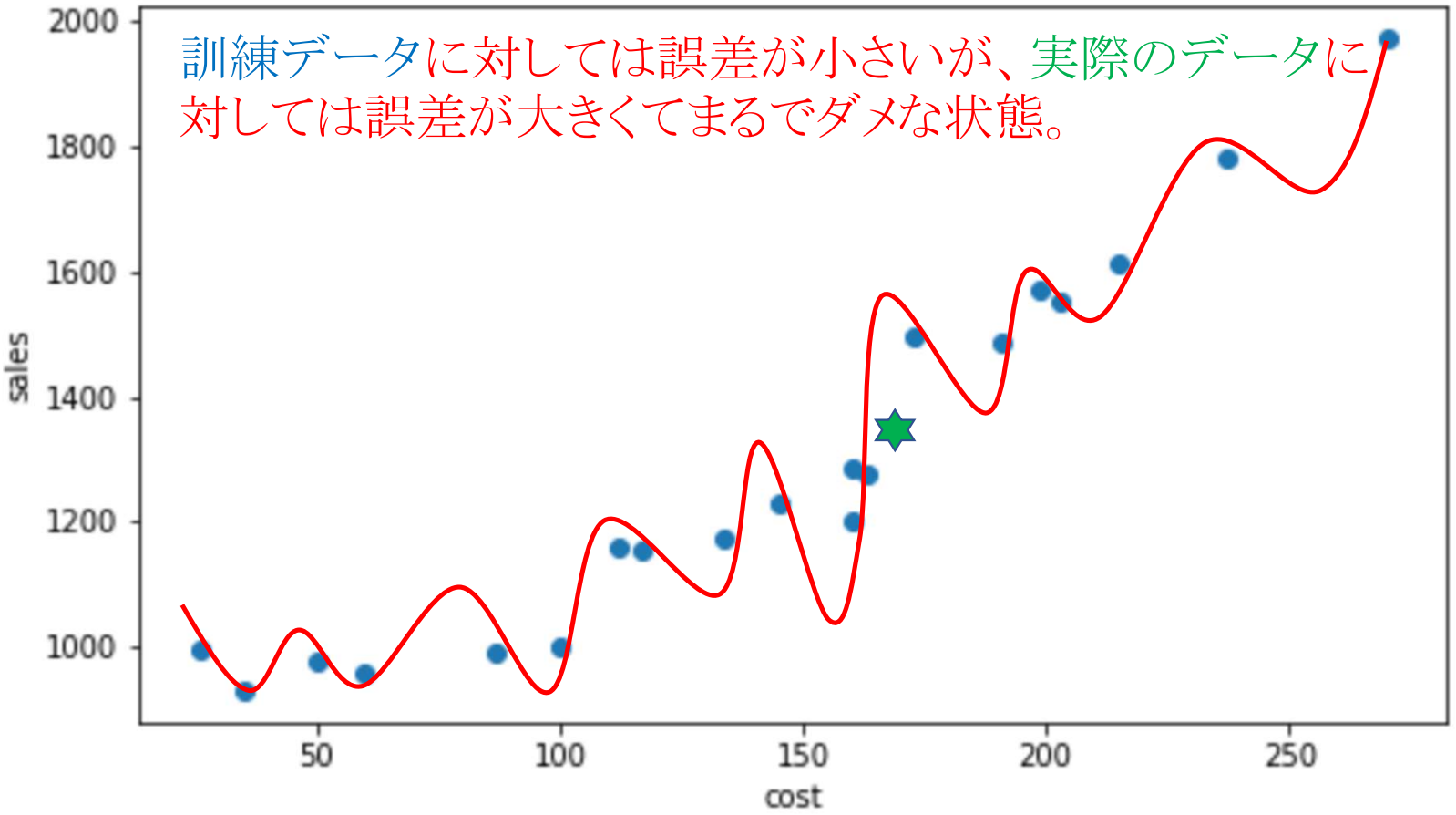


×駄目

【考えてみよう】【例題1】を近似するにはどんな式が適切なのだろうか？

複雑すぎる式 → overfitting (過学習、過適合)

×駄目



【新たな問題】【例題1】を近似するにはどんな式が適切なのだろうか？

単純すぎる式 → 訓練データにそもそも適合できない (underfitting, 過少適合、過少学習)

複雑すぎる式 → 訓練データだけに適合してしまう (overfitting , 過学習、過適合)

×駄目

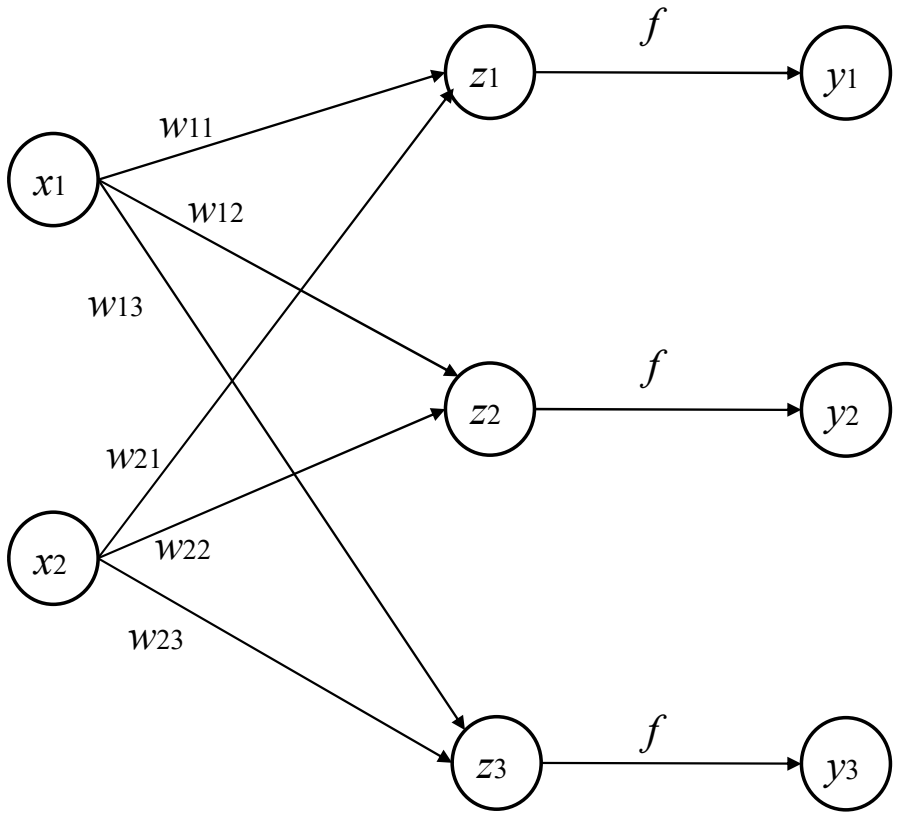
↓

問題に応じた適切な程度に複雑な式　○必要

↓

コンピュータが勝手に見つけてくれたらいいな

ニューラル・ネットワーク・モデル (Neural Network Model)



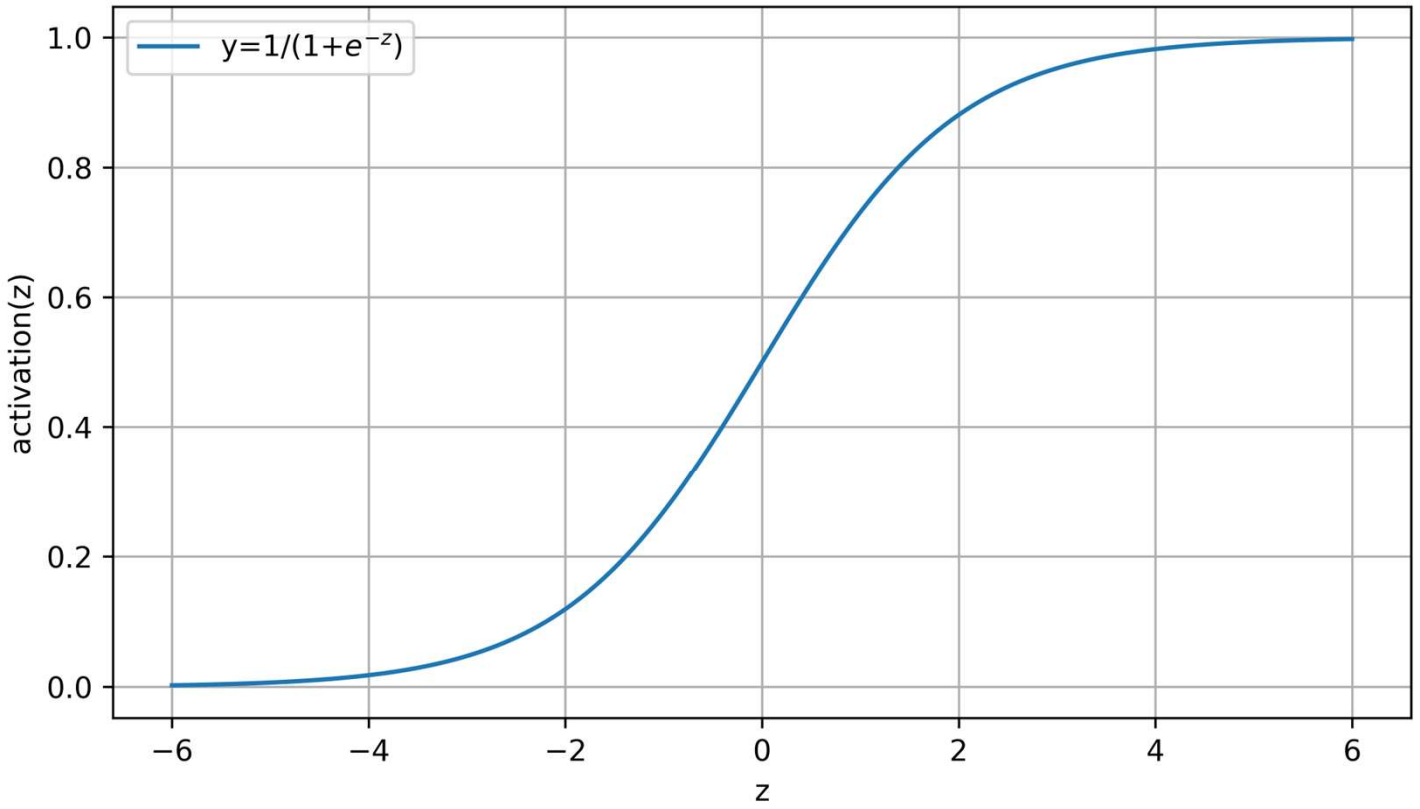
$$z_1 = w_{11}x_1 + w_{21}x_2$$
$$z_2 = w_{12}x_1 + w_{22}x_2$$
$$z_3 = w_{13}x_1 + w_{23}x_2$$

$$y_1 = f(z_1)$$
$$y_2 = f(z_2)$$
$$y_3 = f(z_3)$$

$f(z)$: 活性化関数
(activation function)

いろいろなアクティベーション関数: $y = f(z)$　非線形関数

sigmoid: $y = \dfrac{1}{1 + e^{-z}}$



sigmoid関数: $a = \dfrac{1}{1+e^{-z}}$

$$\frac{\partial a}{\partial z} = -\frac{-e^{-z}}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2}$$

また

$$(1-a)a = (1-\frac{1}{1+e^{-z}})\frac{1}{1+e^{-z}} = \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2}$$

であるから

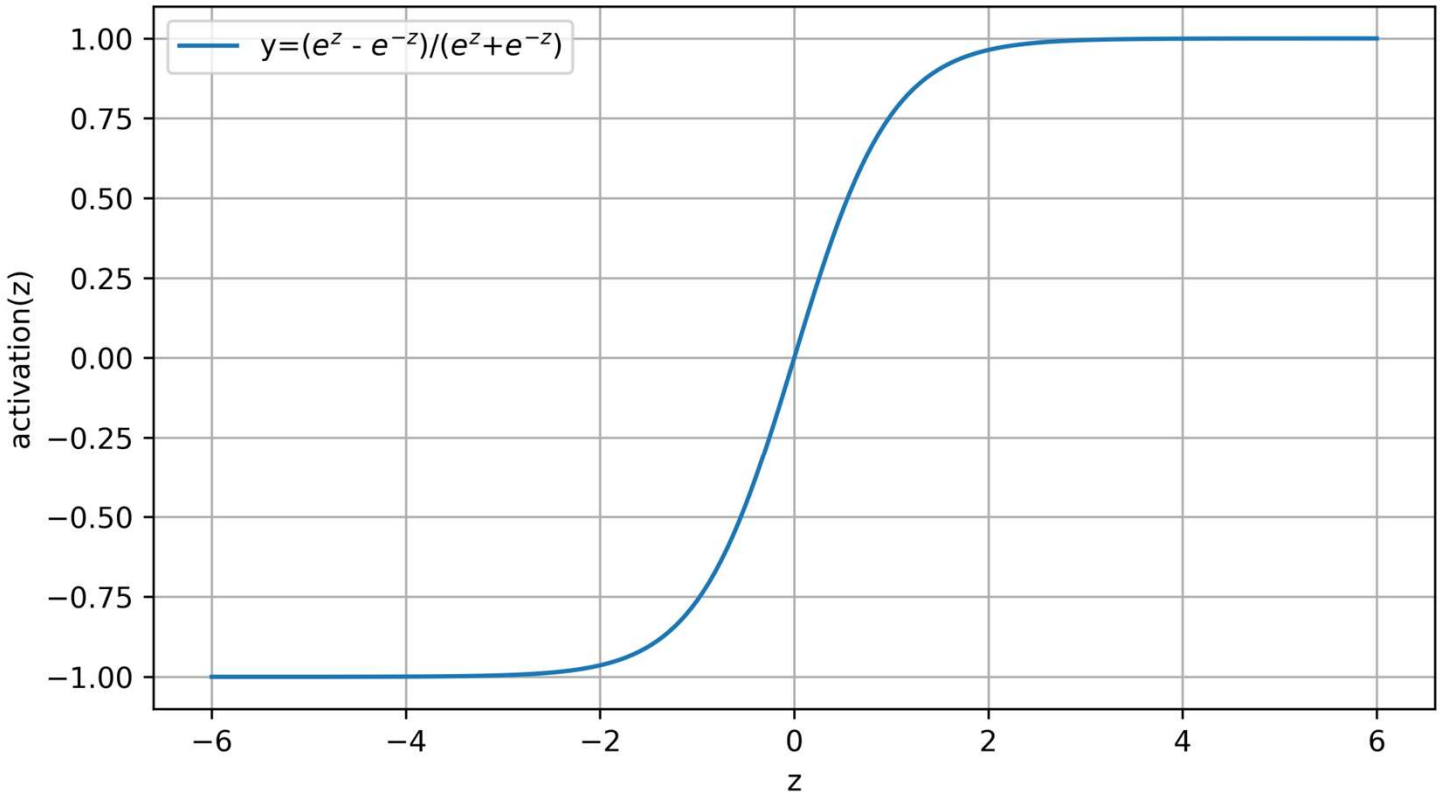
$$\frac{\partial a}{\partial z} = (1-a)a$$

がいえる。したがって、

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\cdot\frac{\partial a}{\partial z} = \frac{\partial L}{\partial z}\cdot(1-a)a$$

いろいろなアクティベーション関数: $y = f(z)$　非線形関数

$$\text{tanh: } y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



tanh関数: $a = \dfrac{e^z - e^{-z}}{e^z + e^{-z}} = (e^z - e^{-z})(e^z + e^{-z})^{-1}$

$$\frac{\partial a}{\partial z} = (e^z + e^{-z})(e^z + e^{-z})^{-1} - (e^z - e^{-z})(e^z - e^{-z})(e^z + e^{-z})^{-2}$$

$$= 1 - (\frac{e^z - e^{-z}}{e^z + e^{-z}})^2$$

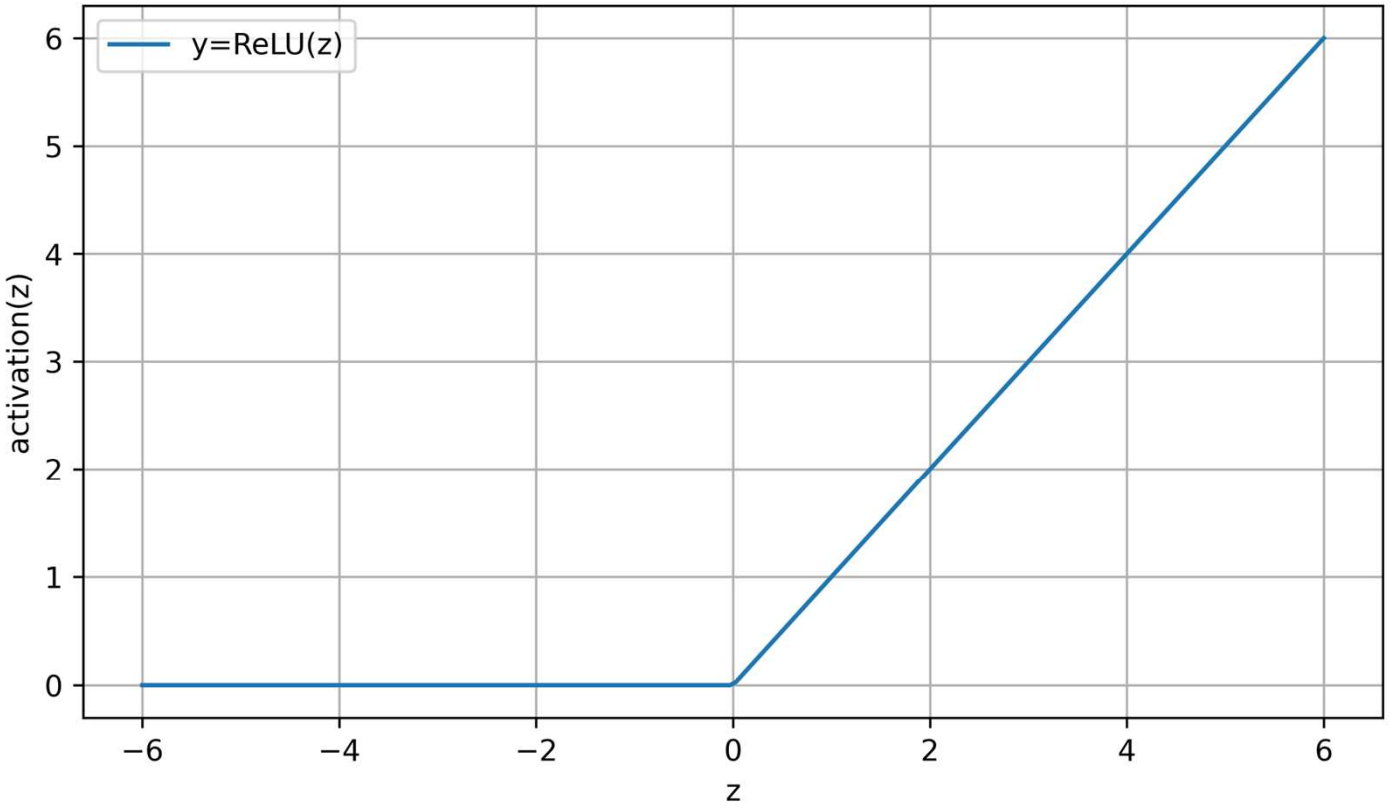
$$= 1 - a^2$$

なので

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} = \frac{\partial L}{\partial z} \cdot (1 - a^2)$$

いろいろなアクティベーション関数: $y = f(z)$　非線形関数

ReLU: $y = \begin{cases} x & if \;\; x > 0 \\ 0 & otherwise \end{cases}$



ReLU 関数 (Rectified Linear Unit)

$$a = \begin{cases} z & (z > 0) \\ 0 & (z \leqq 0) \end{cases}$$

なので

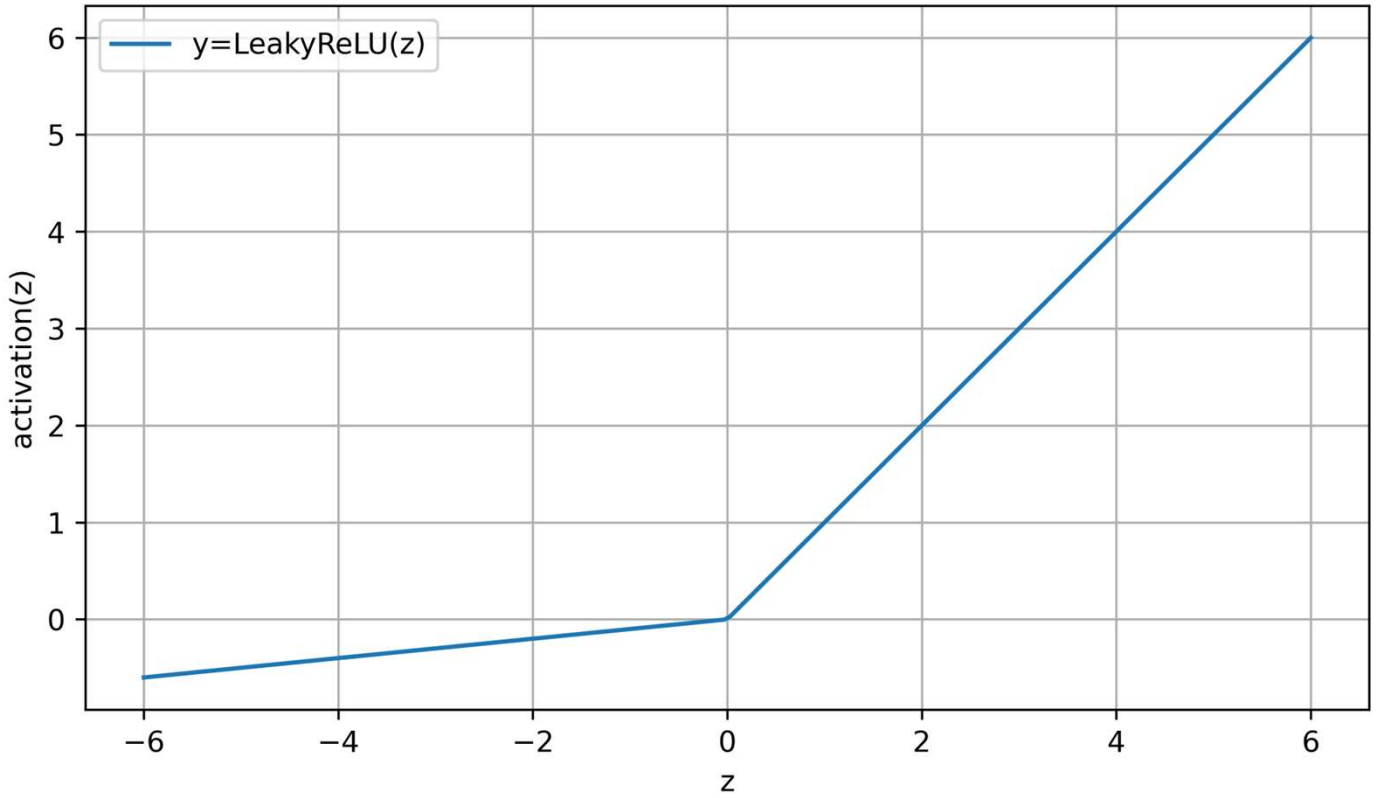
$$\frac{\partial a}{\partial z} = \begin{cases} 1 & (z > 0) \\ 0 & (z \leqq 0) \end{cases}$$

したがって

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} = \begin{cases} \frac{\partial L}{\partial a} & (z > 0) \\ 0 & (z \leqq 0) \end{cases}$$

いろいろなアクティベーション関数: $y = f(z)$　非線形関数

$$\text{LeakyReLU: } y = \begin{cases} x & if \;\; x > 0 \\ x \times 0.1 & otherwise \end{cases}$$



leaky ReLU 関数 (leaky Rectified Linear Unit)

ReLUでは、training中に一部のneuronが0だけを出力するようになり、実質的に死んでしまう dying ReLU とよばれる 問題が起きることがある。ReLU関数では、入力が負ならば gradient が0なのでneuron が復活する見込みはない。 そこで改良されたのが leaky ReLU 関数である。

$$a = \begin{cases} z & (z > 0) \\ \alpha z & (z \leqq 0) \end{cases} \qquad \alpha \simeq 0.01$$

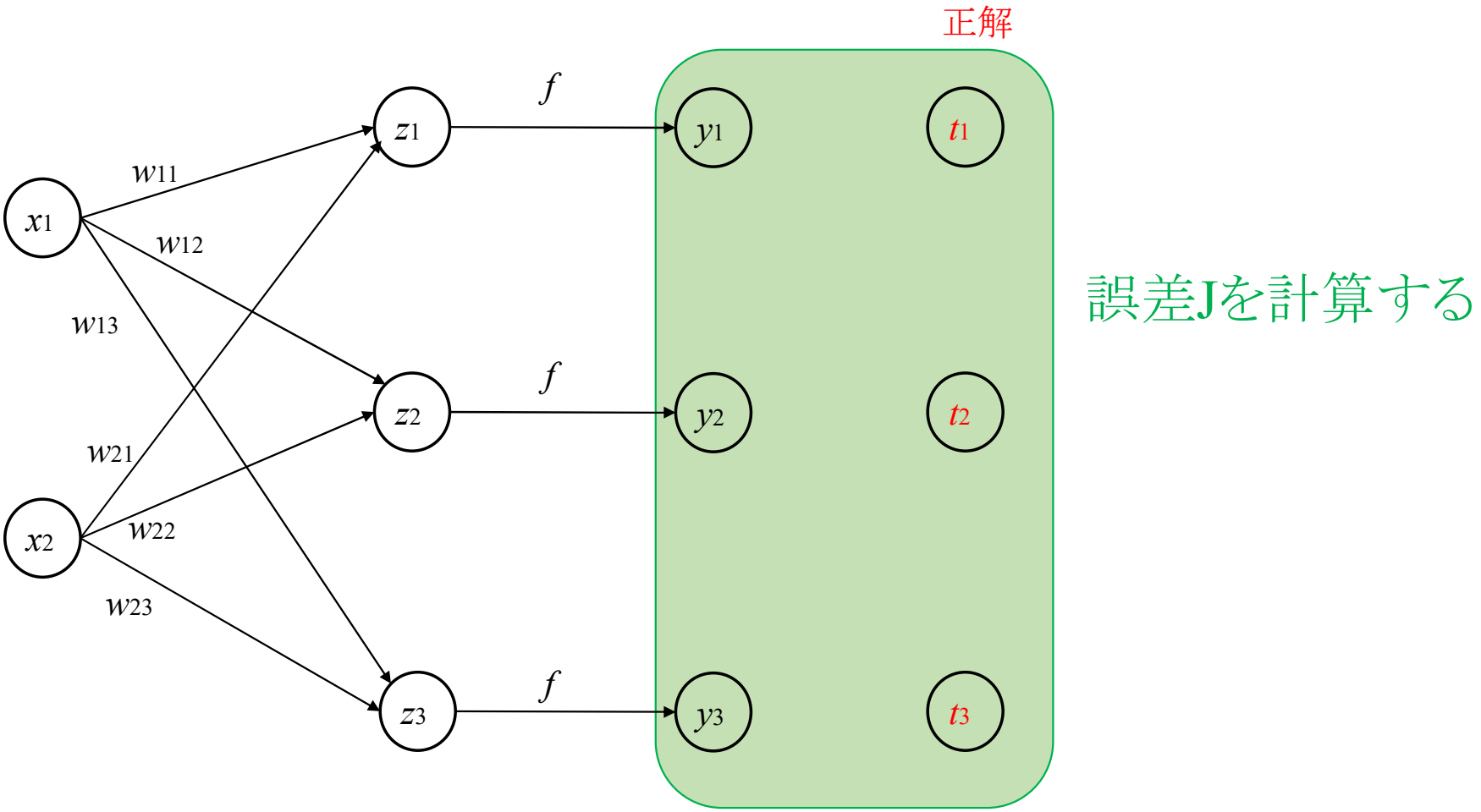
なので

$$\frac{\partial a}{\partial z} = \begin{cases} 1 & (z > 0) \\ \alpha & (z \leqq 0) \end{cases}$$

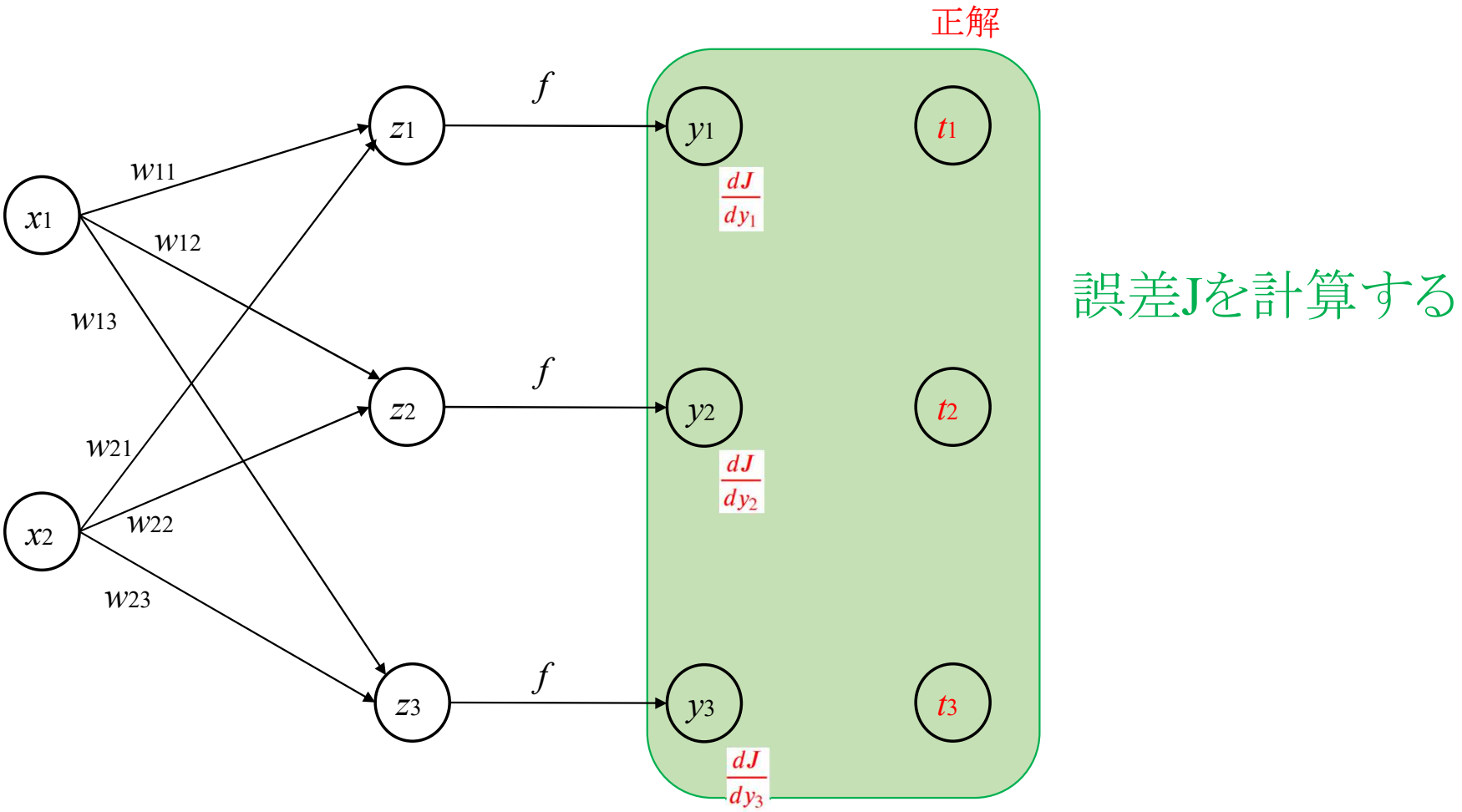
したがって

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} = \begin{cases} \frac{\partial L}{\partial a} & (z > 0) \\ \alpha \frac{\partial L}{\partial a} & (z \leqq 0) \end{cases}$$

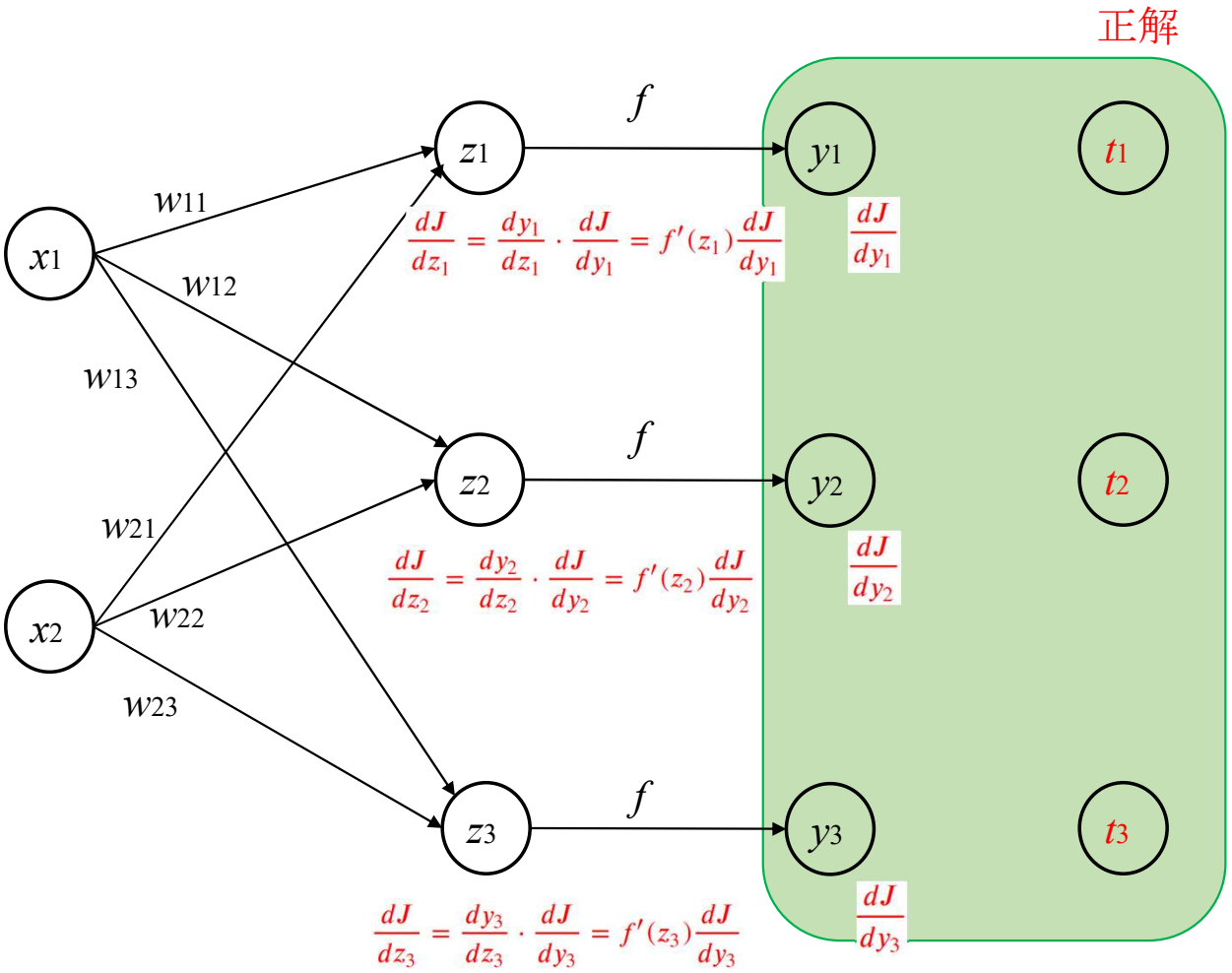
ニューラル・ネットワーク・モデル (Neural Network Model)



ニューラル・ネットワーク・モデル (Neural Network Model)

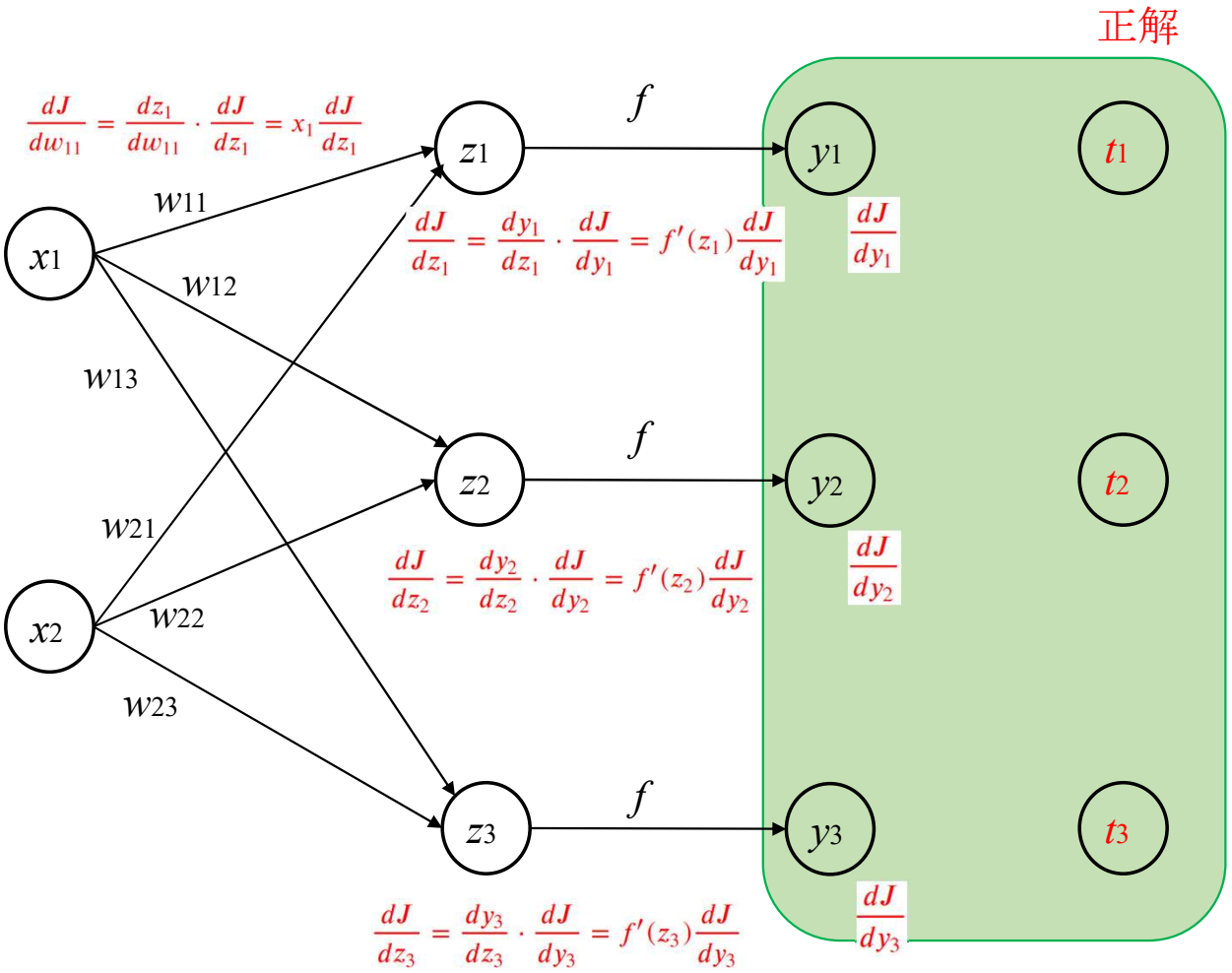


ニューラル・ネットワーク・モデル (Neural Network Model)



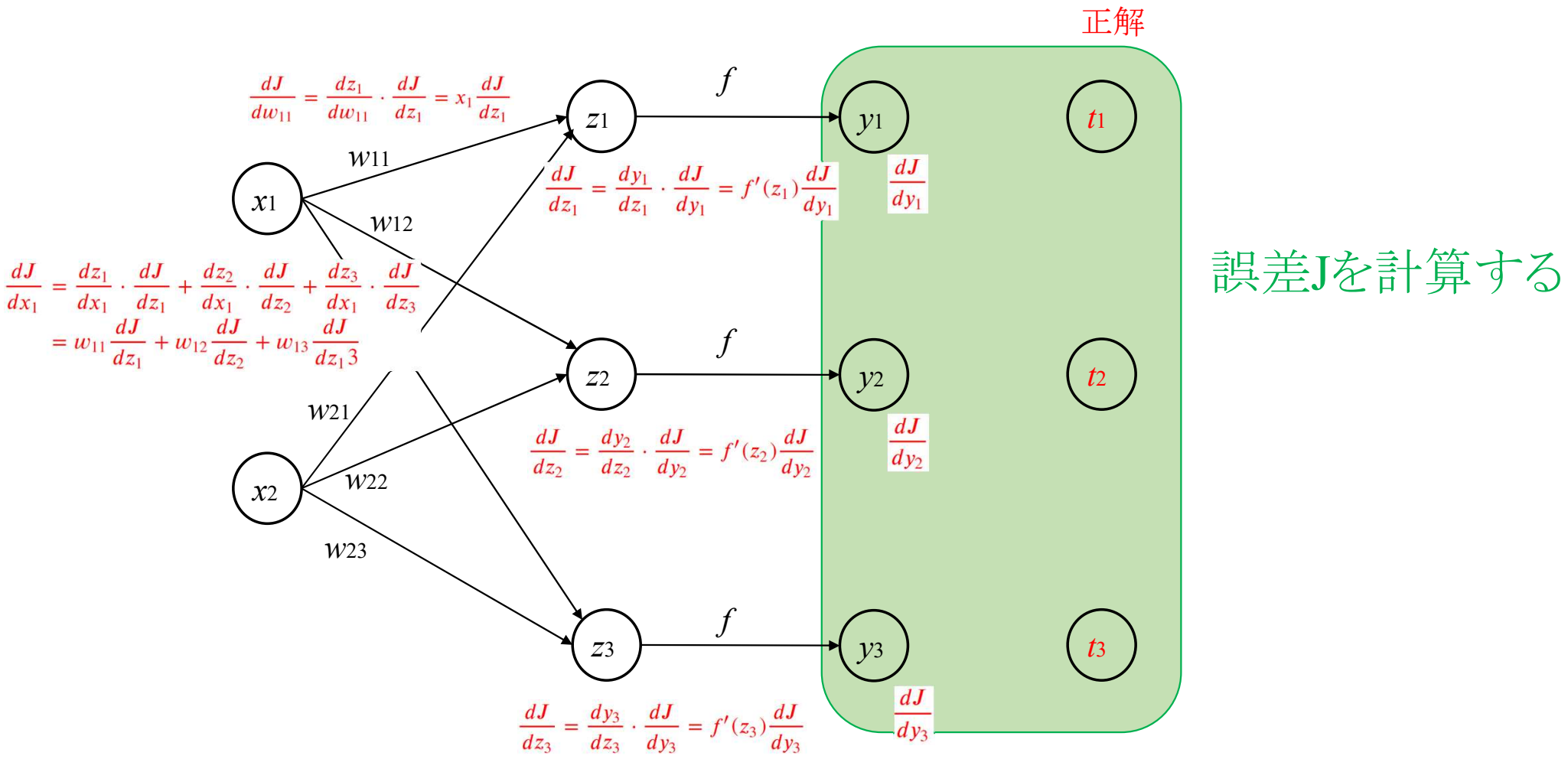
誤差Jを計算する

ニューラル・ネットワーク・モデル (Neural Network Model)



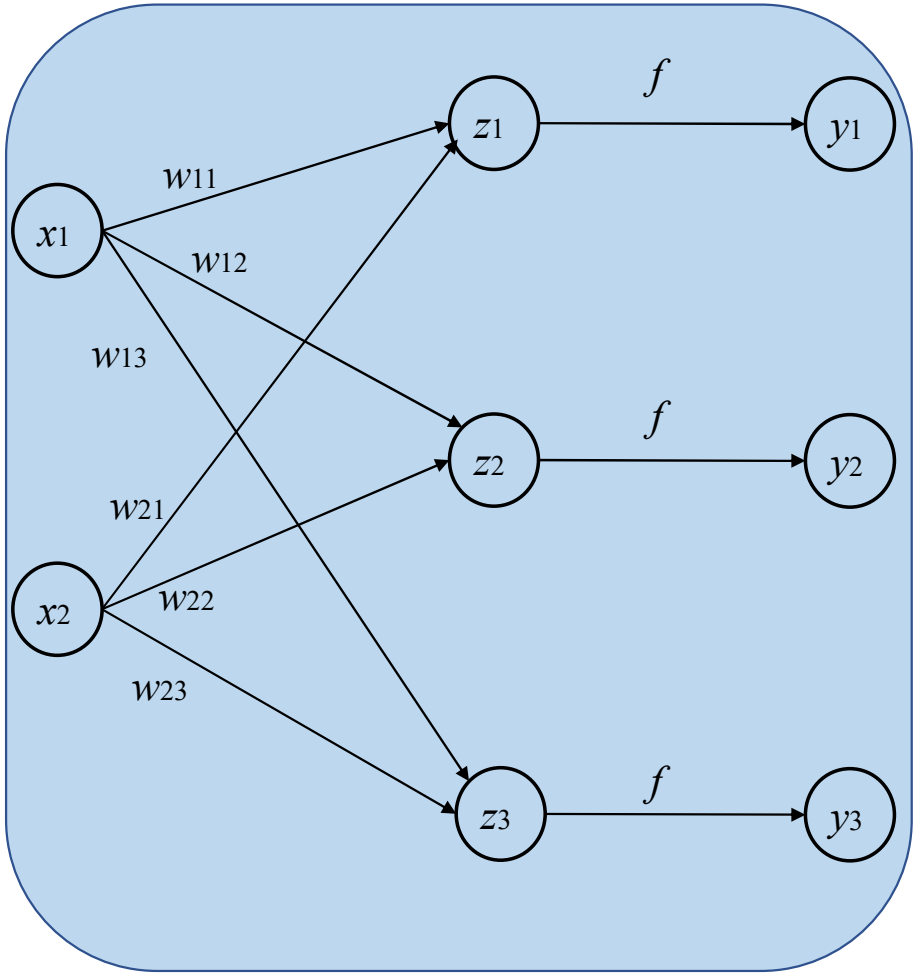
誤差Jを計算する

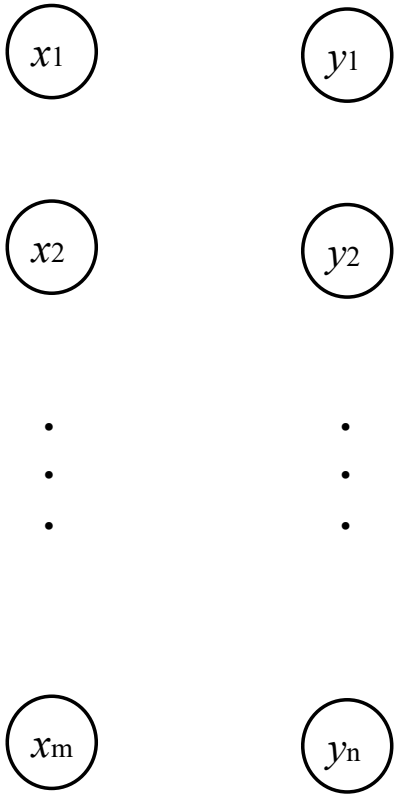
ニューラル・ネットワーク・モデル (Neural Network Model)



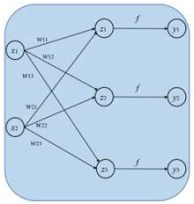
誤差Jを計算する

多層ニューラル・ネットワーク・モデル (Neural Network Model)





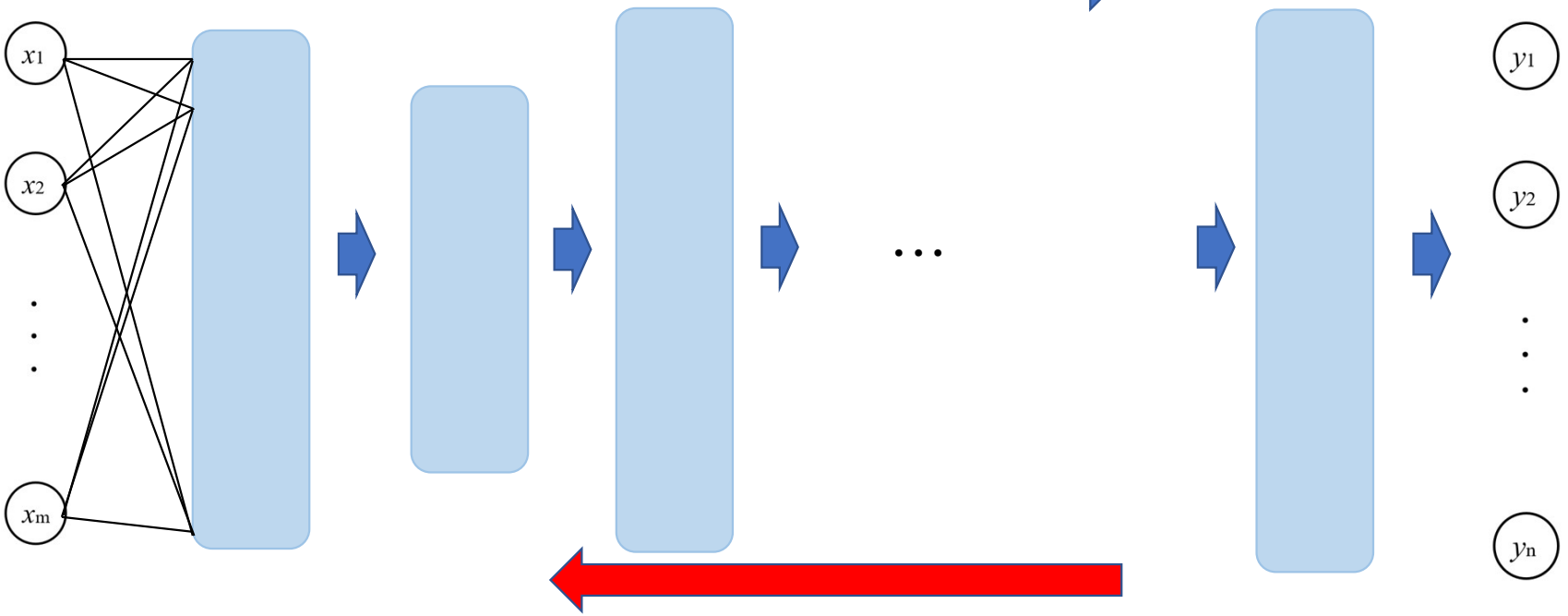
多層ニューラル・ネットワーク・モデル (Neural Network Model)



1層

ニューラルネットワークの層を何段も深くすることで、非常に複雑な関数を模倣することができる。

予測　順伝播 forward propagation



誤差

学習　逆伝播 backward propagation

Deep Learningがどうして流行ったか？

- ニューラルネットワークの理論は1943年に提唱されていて、1957年にはパーセプトロンが開発されている。
- 逆伝播の論文は1986年 by Rumelhart, Hinton, Williams
- 何度か流行と冬の時代を繰り返す
- 3層のネットワーク(入力28x28画像→500→500→2000→10種類のラベル)が学習できたのは2006年(Hinton, 事前学習による)。
- コンピュータの発達(CPUの速度向上、メモリの増大)
- GPU コンピューティングの発展
- 2012年の物体認識コンテストILSVRC ImageNet Large Scale Visual Recognition Challenge)でぶっちぎりの優勝
- 学習方式の改善: Stochastic GD, RMSprop, Adam, …
- モデルの改善: 畳み込み、リカレント(時間軸方向), Dropout, …
- ツールの充実: TensorFlow, PyTorch, …

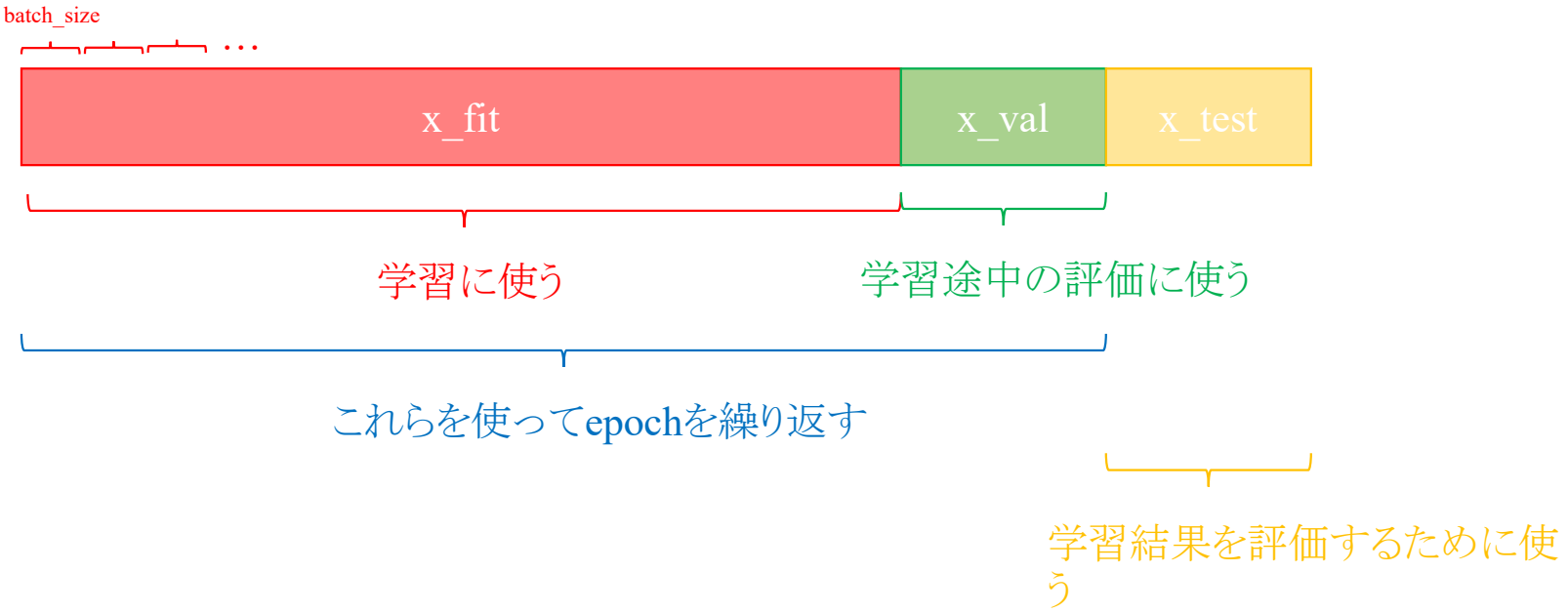
【例題2】Fashion-MNISTのデータを画像分類してみよう

- 10種類のファッションアイテムのモノクロ画像
- 画像の解像度は28x28

ファッションアイテム	ラベル
Tシャツ／トップス	0
ズボン	1
プルオーバー	2
ドレス	3
コート	4
サンダル	5
シャツ	6
スニーカー	7
バッグ	8
アンクルブーツ	9

【例題2】Fashion-MNISTのデータを画像分類してみよう

- 訓練データは60000個　☜ これを使って繰り返し学習する
  - 学習には80%の48000個使う
  - 検証用には20%の12000個使う
- テストデータは10000個　☜ 最後にこれで学習結果を確認する



【例題2】Fashion-MNISTのデータを画像分類してみよう

- 実際に機械学習の様子をデモします。
- TensorFlow + Keras を使います。
- Google Colab https://colab.research.google.com
- 手順は以下のURLを参考にしてください。
  - https://nw.tsuda.ac.jp/lec/TensorFlow/book4/html/tf2_book4_ch04_03.html

【例題3】YOLO-3

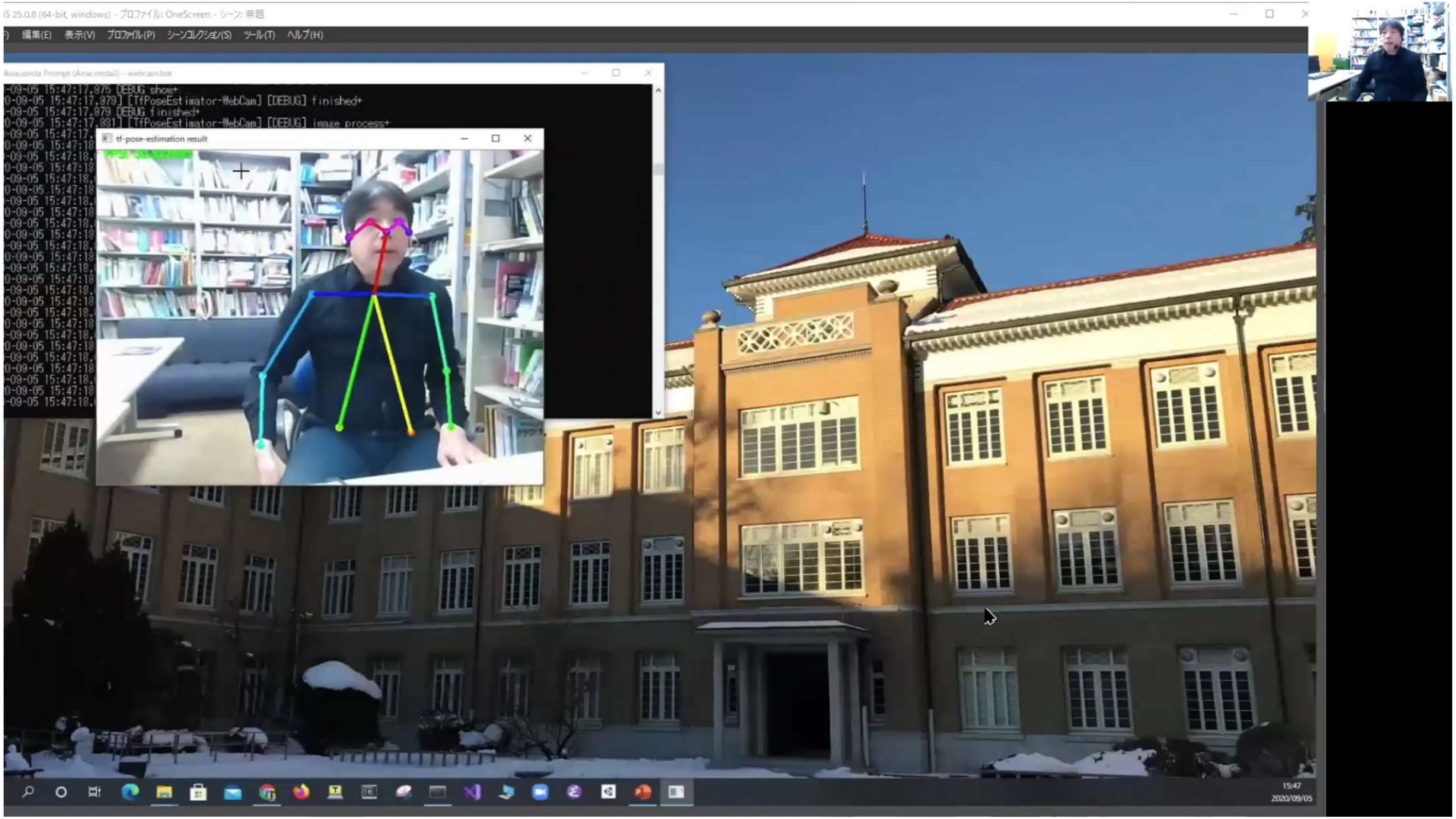
- You Only Look Once
  - 一度に複数の物体検出を行う
  - 検出された物体の場所も返す
- サンプル画像
- デモ　☜ リアルタイムのWeb Camの画像をYOLO-3で処理してみる

【例題3】YOLO-3

【例題4】OpenPose

- RGB画像から人間の骨格を検出する
- 何人でも、隠れた部分があってもOK
- CMU が開発
- デモ　☜ リアルタイムのWeb Camの画像をOpenPoseで処理してみる

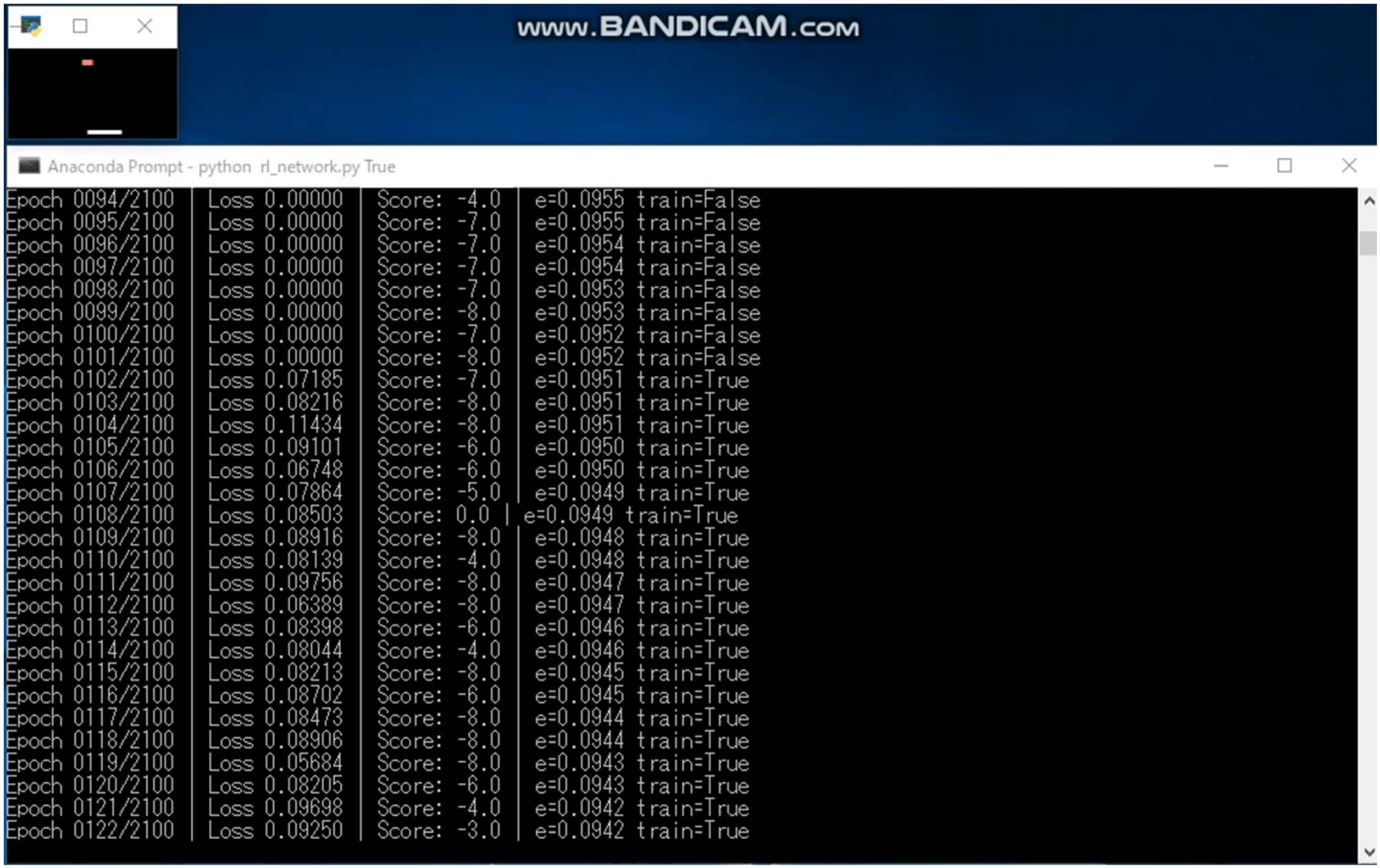
【例題4】OpenPose



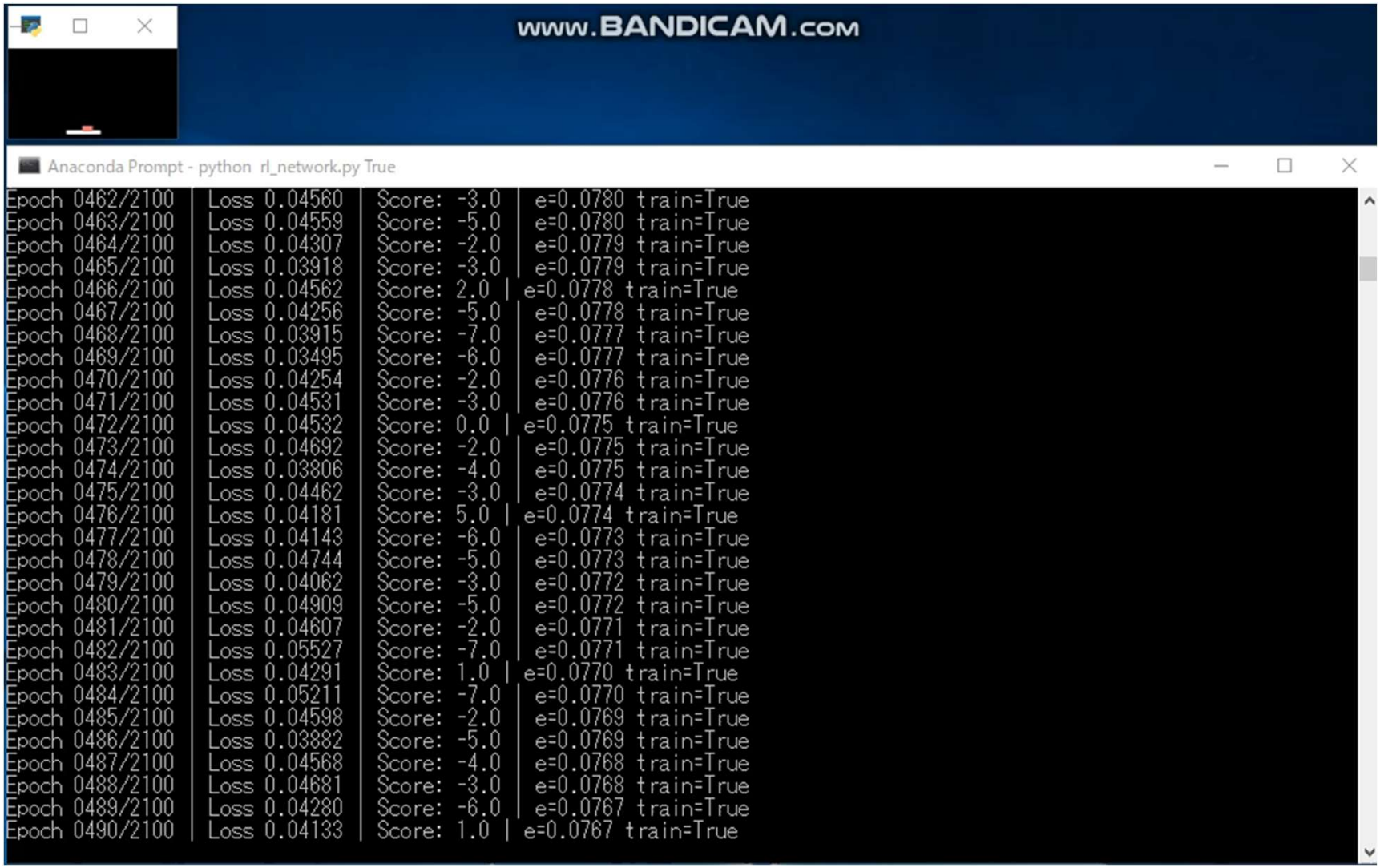
【例題5】強化学習 (Reinforced Training)

- 正しい行動には報酬を与え、間違った行動にはペナルティ(負の)報酬を与える。
- ボールをキャッチするゲームを強化学習によって、訓練する

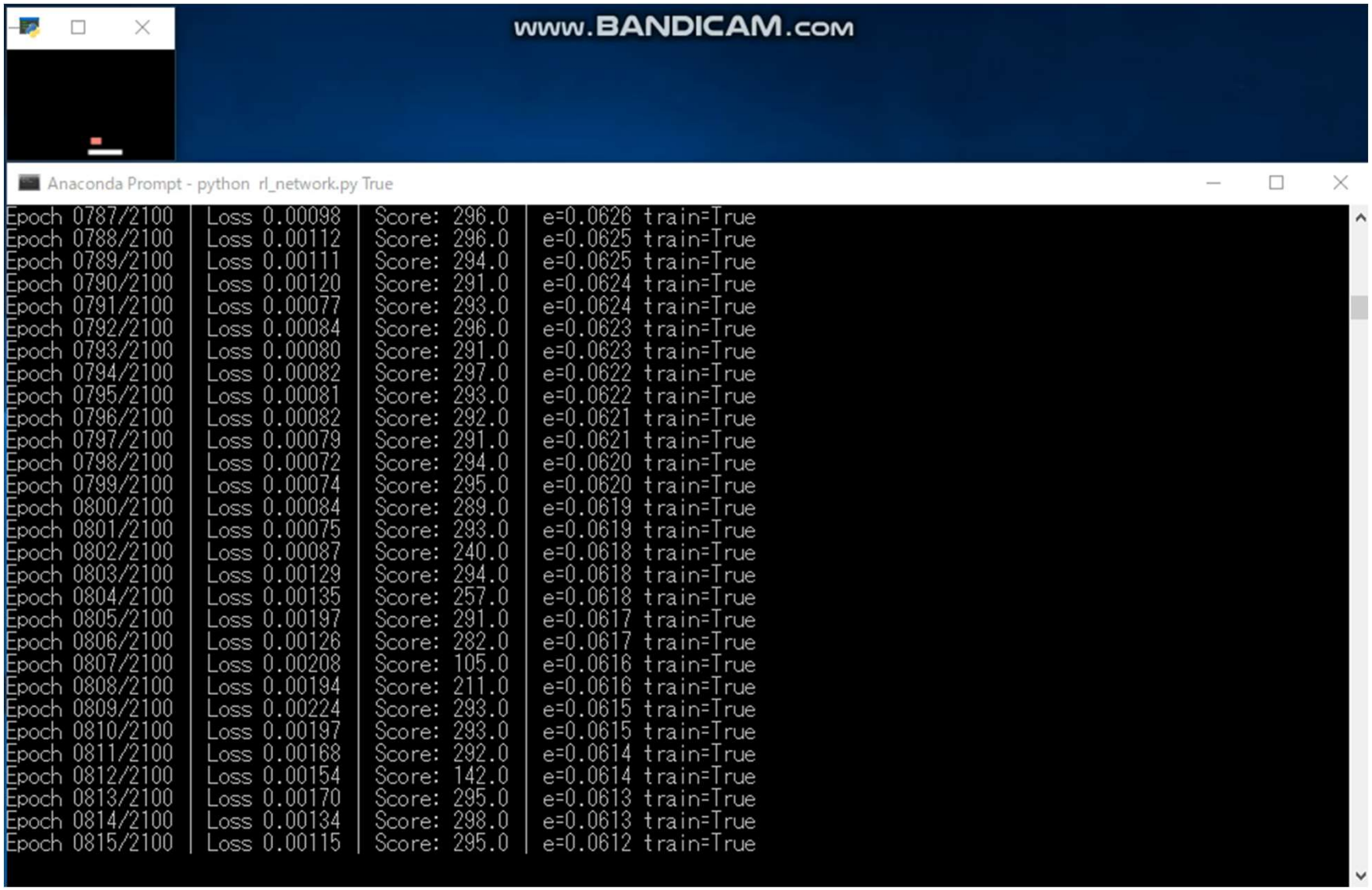
【例題5】強化学習 (Reinforced Training)ボールキャッチ epoch 0



【例題5】強化学習 (Reinforced Training)ボールキャッチ epoch 11



【例題5】強化学習 (Reinforced Training)ボールキャッチ epoch 18



【例題5】強化学習 (Reinforced Training)ボールキャッチ 学習終了

まとめ

- 機械学習、AI、ディープラーニングについて簡単に説明しました。
- これらの領域は最近ではデータサイエンスと呼ばれていて、現在も急速に発展しています。社会における重要性も飛躍的に増大しています。
- GAN (Generative Adversarial Network), Q-Learning, …
- 基礎から理解すれば、基本となる技術は決して難しくはないことが分かっていただけたと思います。
- 情報科学はいつも発展し続け、領域をどんどん広げていく学問です。
- 情報科学の面白さを、少しでも理解していただけたら、うれしく思います。