

aa-6. 画像分類システム

(人工知能)

URL: <https://www.kkaneko.jp/ai/mi/index.html>

金子邦彦



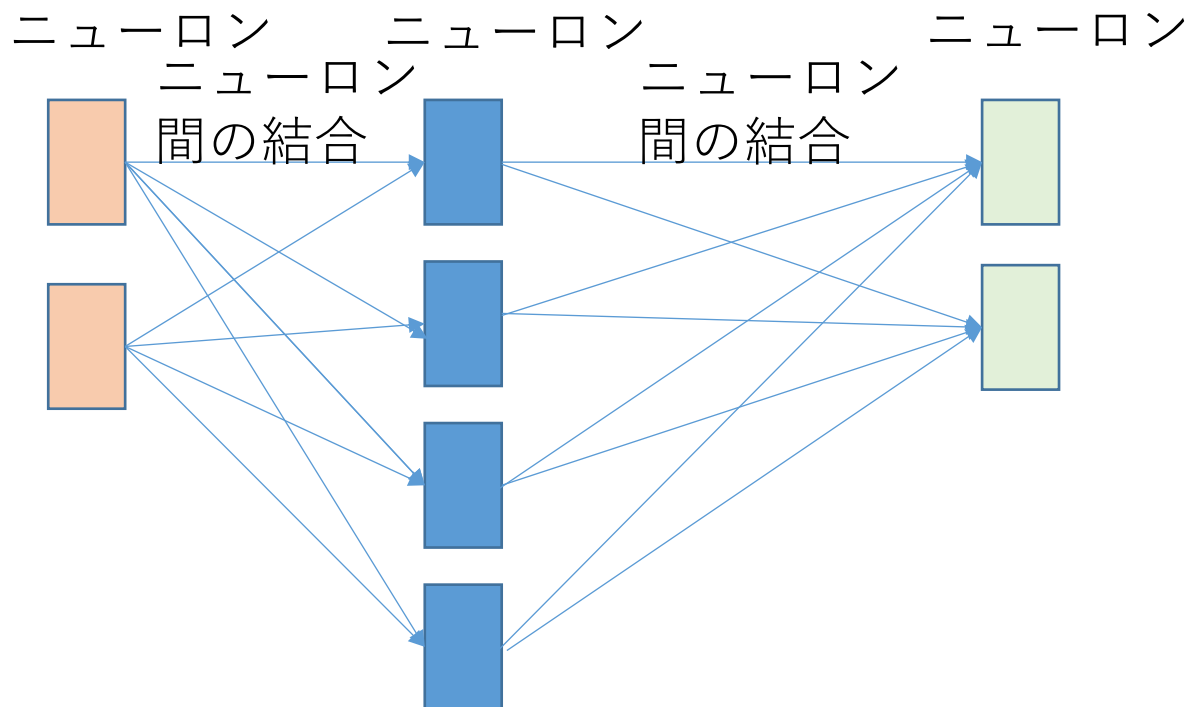
アウトライン

1. 画像と画素
2. 濃淡画像のデータ
3. ニューラルネットワークを用いた分類
4. 画像分類システム
5. ニューラルネットワークの作成
6. 学習
7. 画像分類

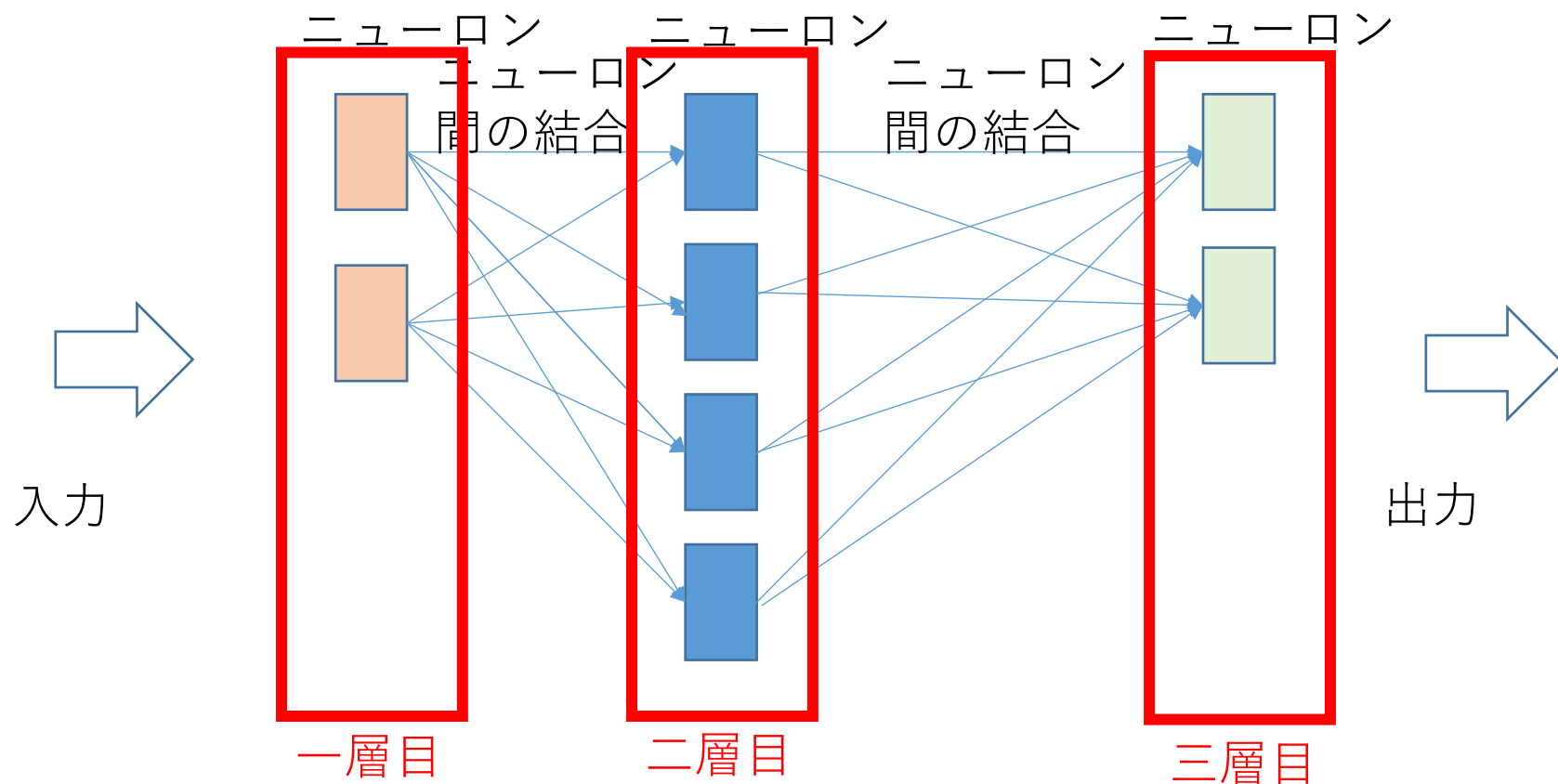
ニューラルネットワークの仕組み



入力の重みづけ, 合計とバイアス, 活性化関数の適用を行う
ニューロンがネットワークを形成



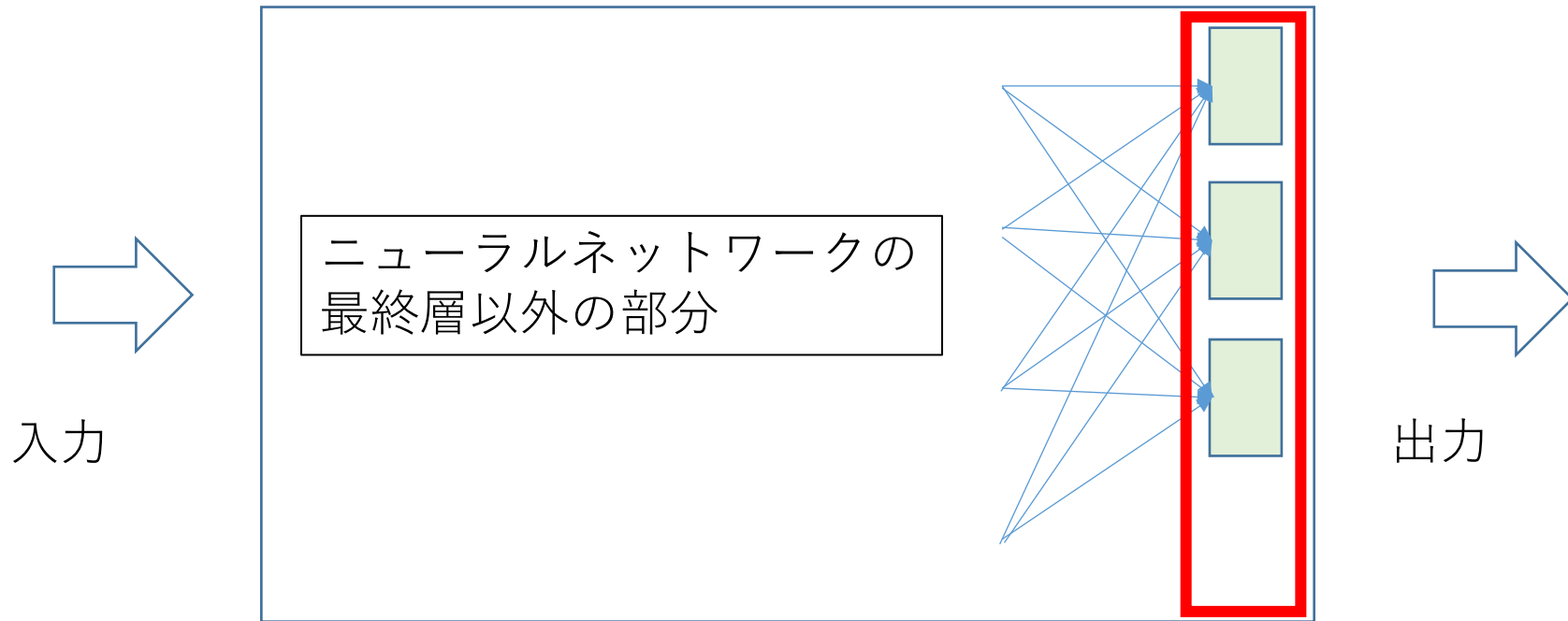
層構造とデータの流れ



- 層構造では、データは、入力から出力への一方向に流れる
- 各層は、同じ種類のニューロンで構成

3種類の中から1つに分類する場合

最終層のニューロン数：3 にする ニューロン



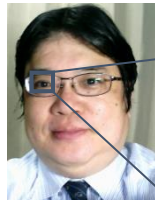
最終層

ニューロン数：3

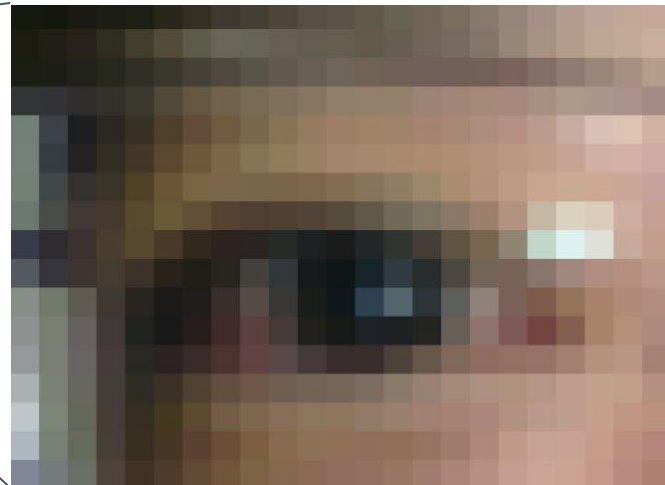
データは入力から出力の方向へ

6.1 画像と画素

画像と画素



画像



それぞれの格子が画素

画像の種類



カラー画像

輝度と色の情報



濃淡画像

輝度のみの情報

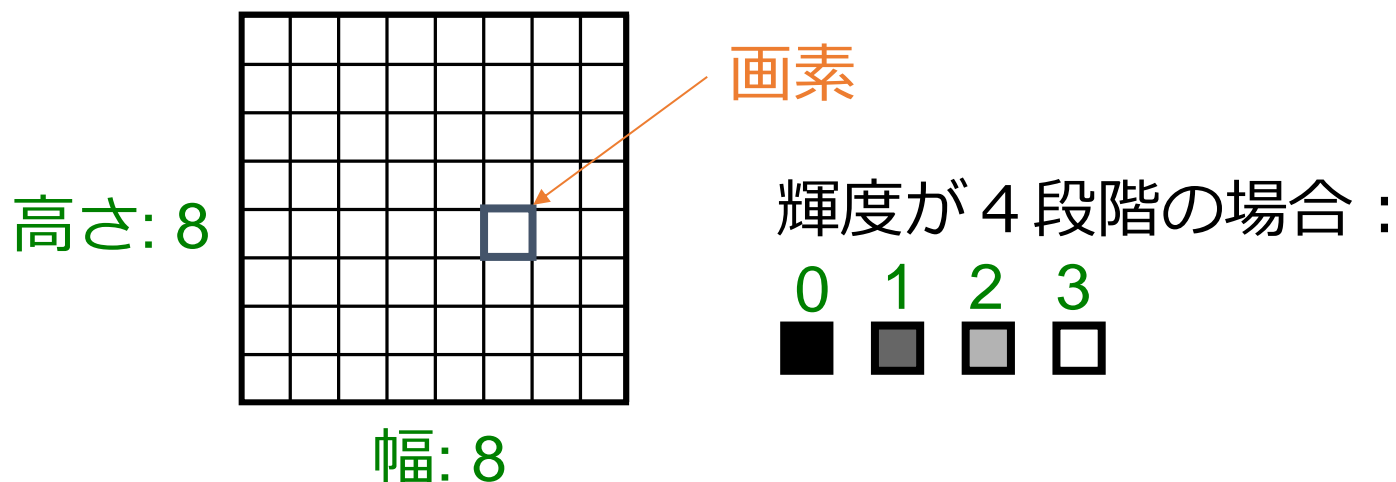
濃淡画像でのコード化



画像の輝度の情報

例えば： 黒 = 0 ,
暗い灰色 = 1 ,
明るい灰色 = 2 ,
白 = 3

のように**コード化**



カラー画像の成分

- R (赤) 成分, G (緑) 成分, B (青) 成分で考える場合



R (赤) 成分



G (緑) 成分



B (青) 成分

- 輝度成分, 色成分で考える場合



輝度成分

色成分

R (赤) 成分, G (緑), B (青) 成分で考える 場合



R (赤) 成分



G (緑) 成分



B (青) 成分



画素ごとに
1つの数値



画素ごとに
1つの数値



画素ごとに
1つの数値

すべてあわせて, 画素ごとに3つの数値

輝度成分，色成分で考える場合



輝度成分



画素ごとに
1つの数値



色成分



画素ごとに
2つの数値

すべてあわせて，画素ごとに3つの数値

画像と画素のまとめ

- **画像**：画素と呼ばれる単位で構成
- **画像の種類**：カラー画像、濃淡画像
- **カラー画像**：各画素において赤（R）、緑（G）、青（B）の3つの成分（輝度と色の情報）。画素ごとに**3つの数値**。
- **濃淡画像**：各画素において輝度のみの情報。例えば黒を0、暗い灰色を1、明るい灰色を2、白を3などのようにコード化。画素ごとに**1つの数値**



カラー画像

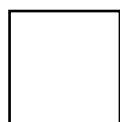


濃淡画像

6.2 濃淡画像のデータ

単純な白黒画像

画素：白と黒の2種類しかないとする

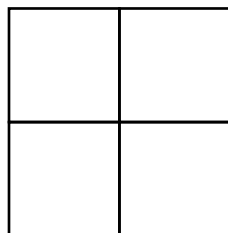


白は 0

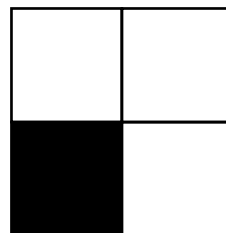


黒は 1 とする

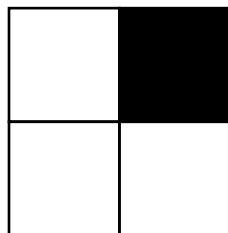
画像のサイズが 2×2 のとき



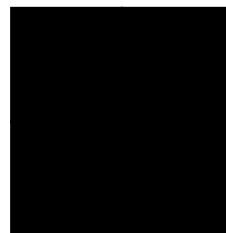
$[0, 0, 0, 0]$



$[0, 0, 1, 0]$

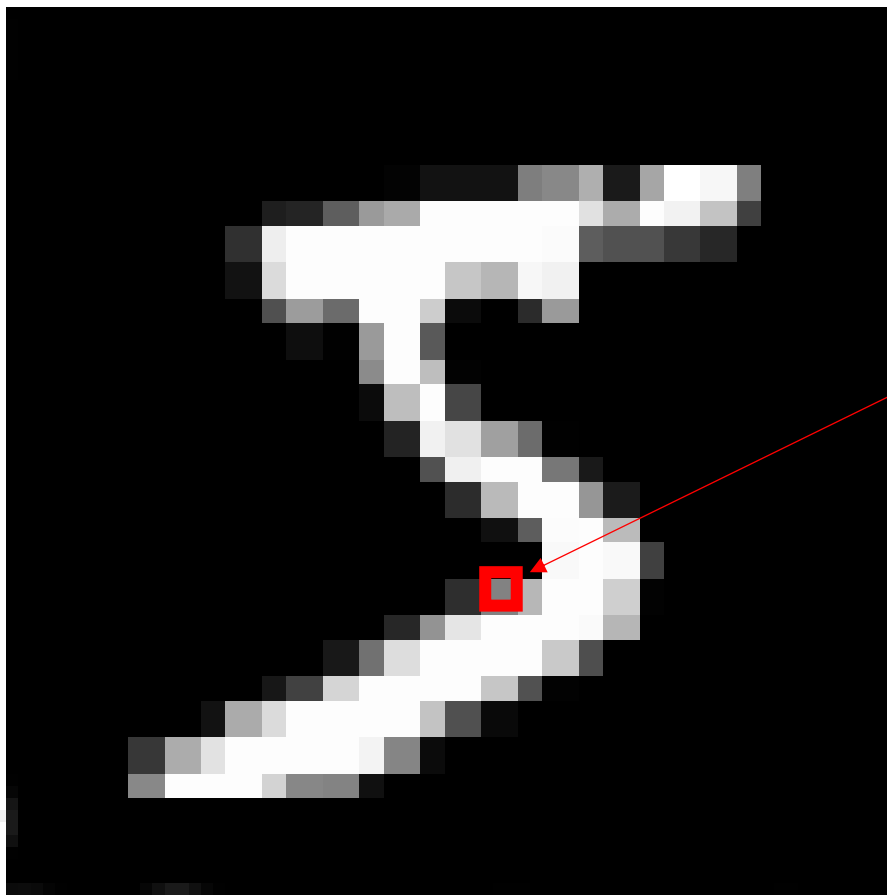


$[0, 1, 0, 0]$



$[1, 1, 1, 1]$

濃淡画像



MNISTデータセット（手書き文字のデータセットで、濃淡画像）

画像サイズ: **28 × 28**

画素

画素値



白

255



黒

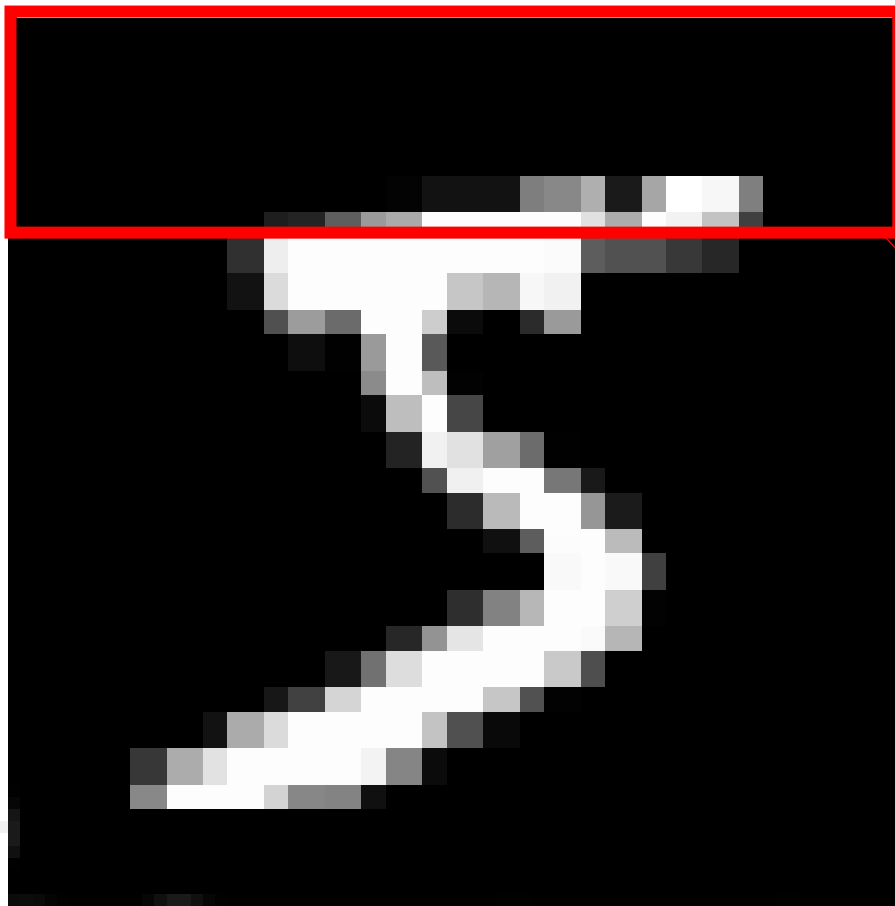
0

画素値は、**画素**の輝度に
応じた **0 から 255 の数値**

濃淡画像と画像データ



画像全体は784個の
数値



[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136
	175	26	166	255	247	127	0	0	0	0	0	0	0	0	0	0	0	0	0	
[0	0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	
	225	172	253	242	195	64	0	0	0	0	0	0	0	0	0	0	0	0	0	

MNISTデータセット（手書き文字のデータセットで，濃淡画像）

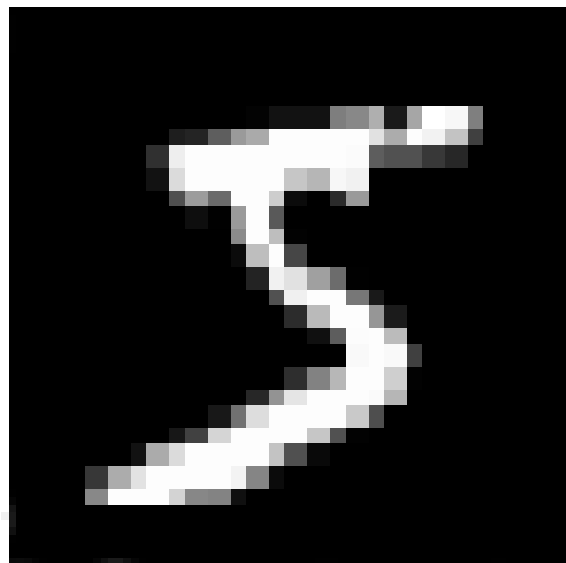
画像サイズ: 28 × 28

画像の上 7行分の画
素値を表示したとこ
ろ（28 × 7分）

濃淡画像のデータのまとめ



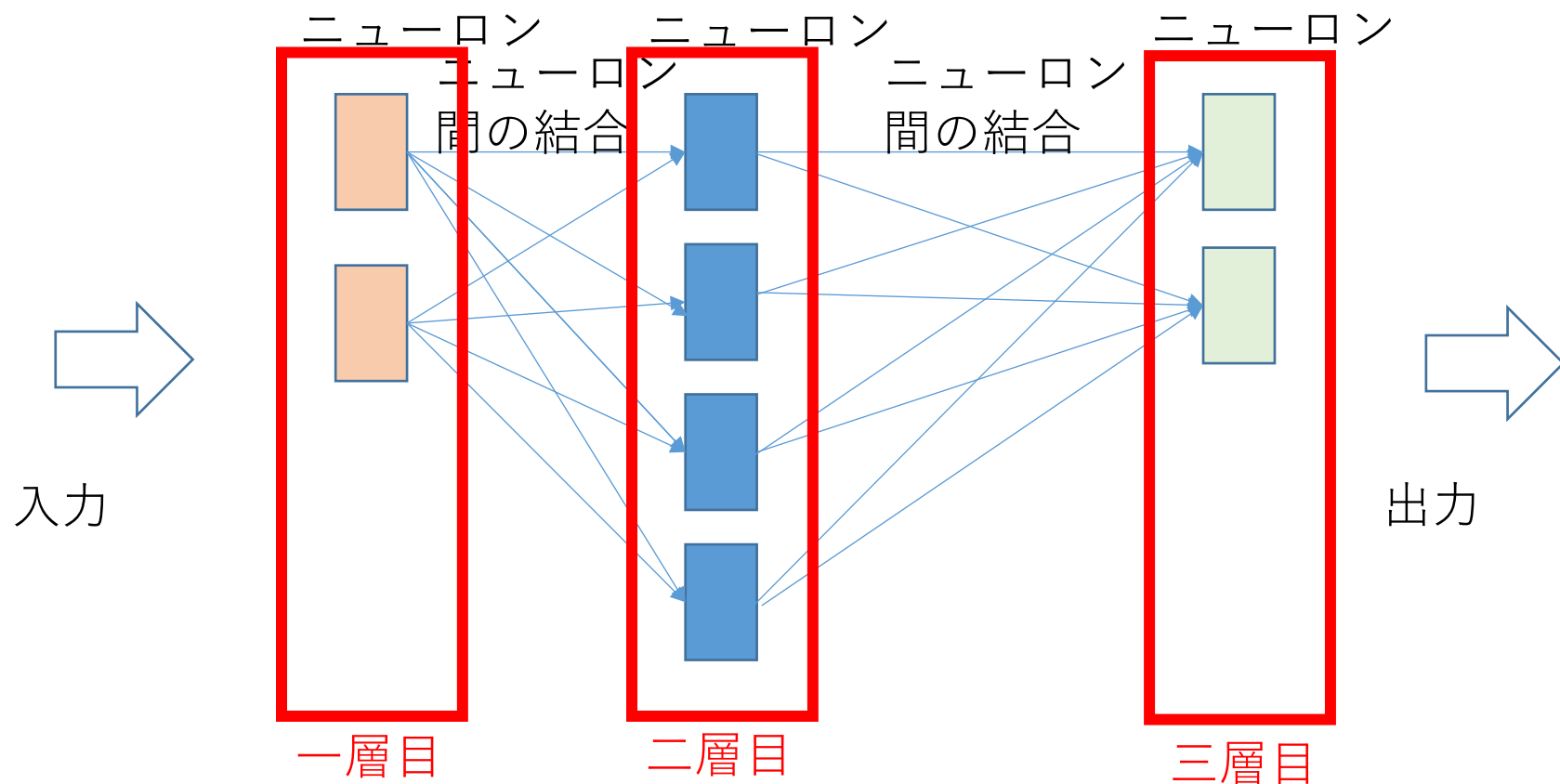
28×28の濃淡画像の場合



- 画素数は 784
- 画像全体は 784個の数値

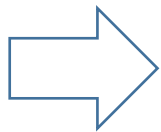
6.3 ニューラルネットワーク を用いた分類

層構造とデータの流れ

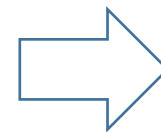
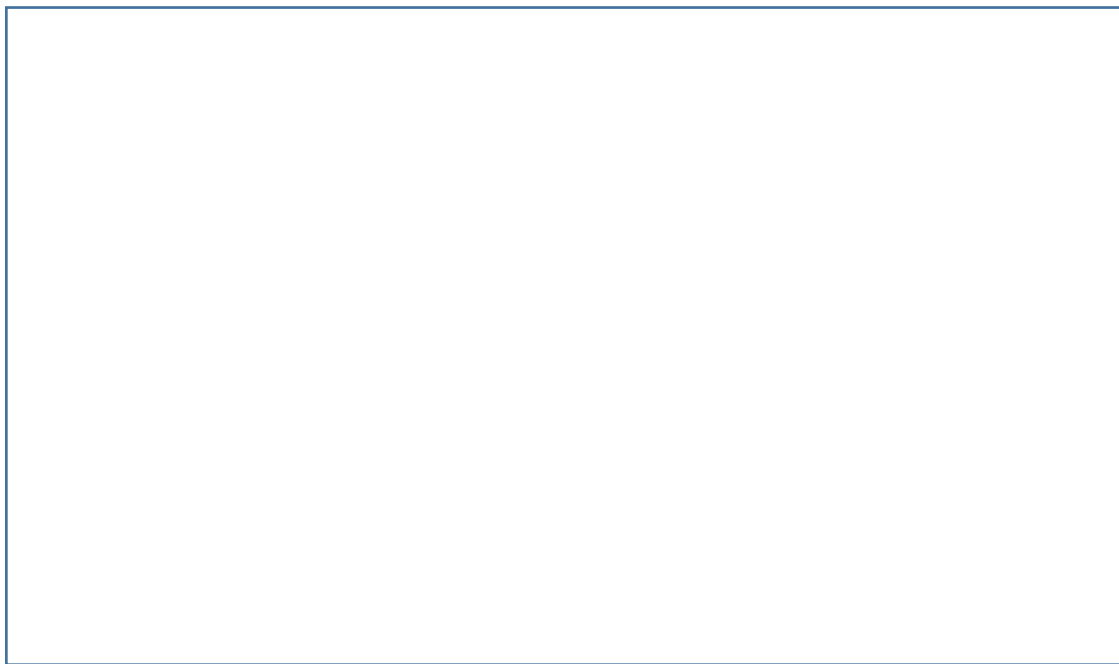


- 層構造では、データは、入力から出力への一方向に流れる
- 各層は、同じ種類のニューロンで構成

3 種類の中から 1 つに分類する場合



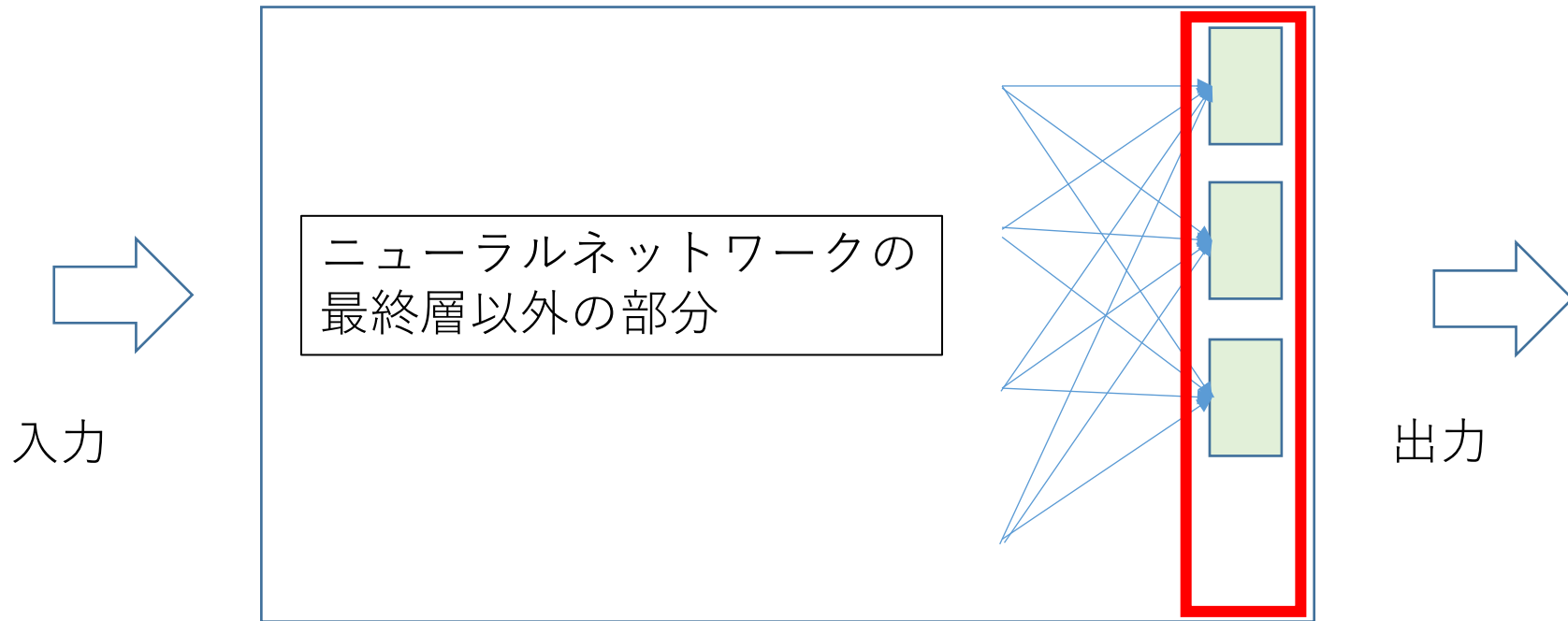
入力



分類結果
0 または 1
または 2

3 種類の中から 1 つに分類する場合

最終層のニューロン数 : **3** にする ニューロン

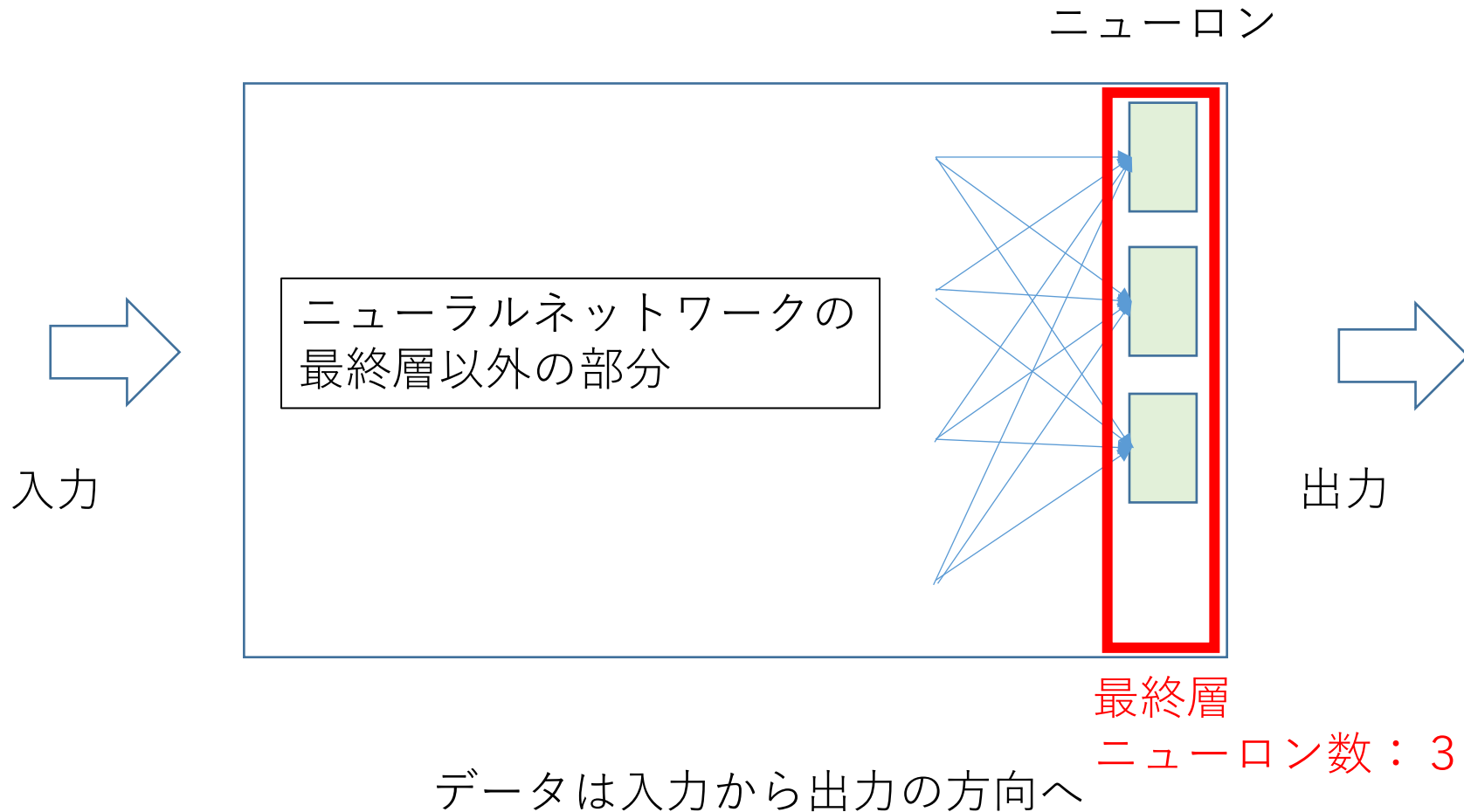


最終層

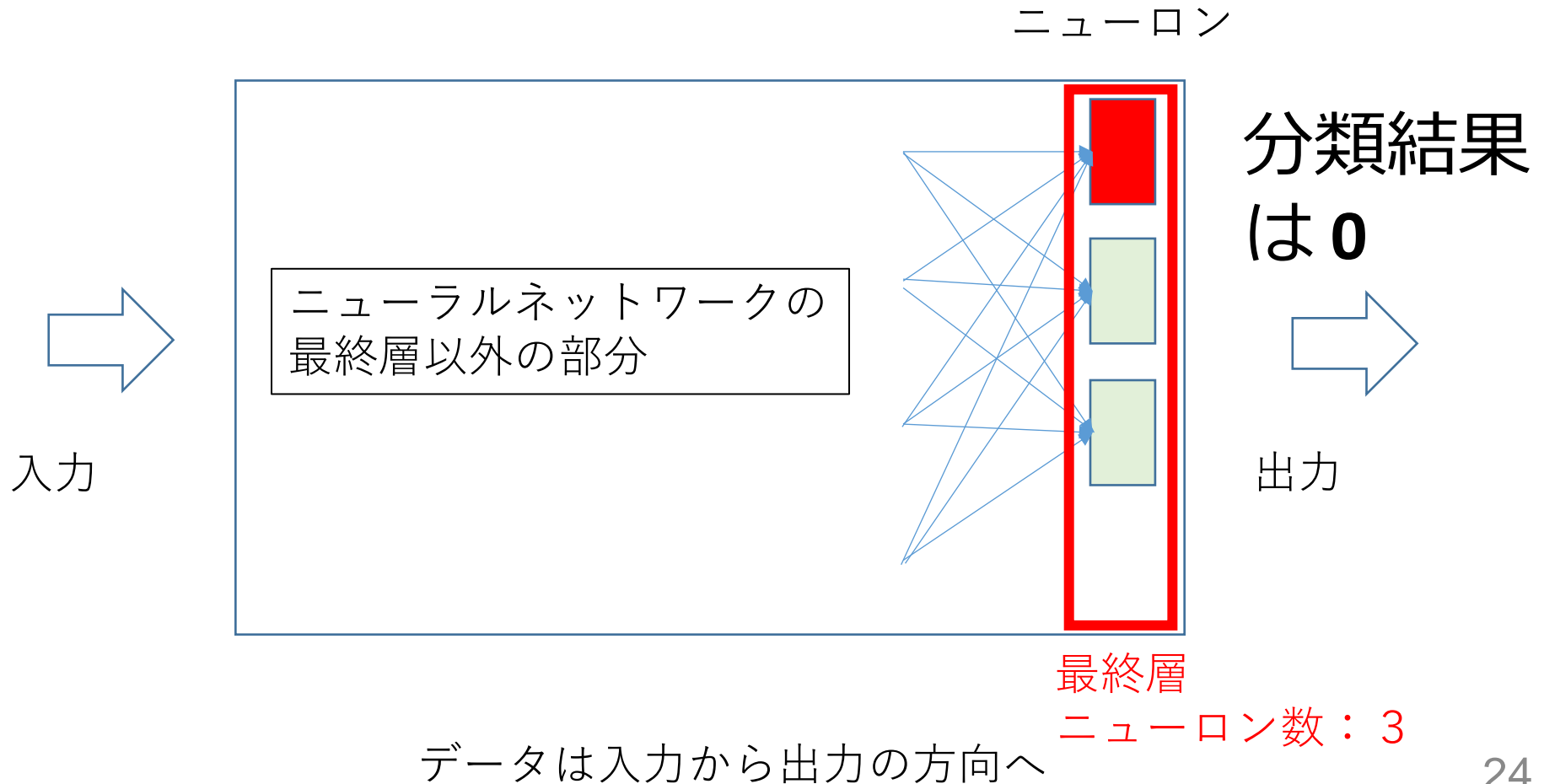
ニューロン数 : 3

データは入力から出力の方向へ

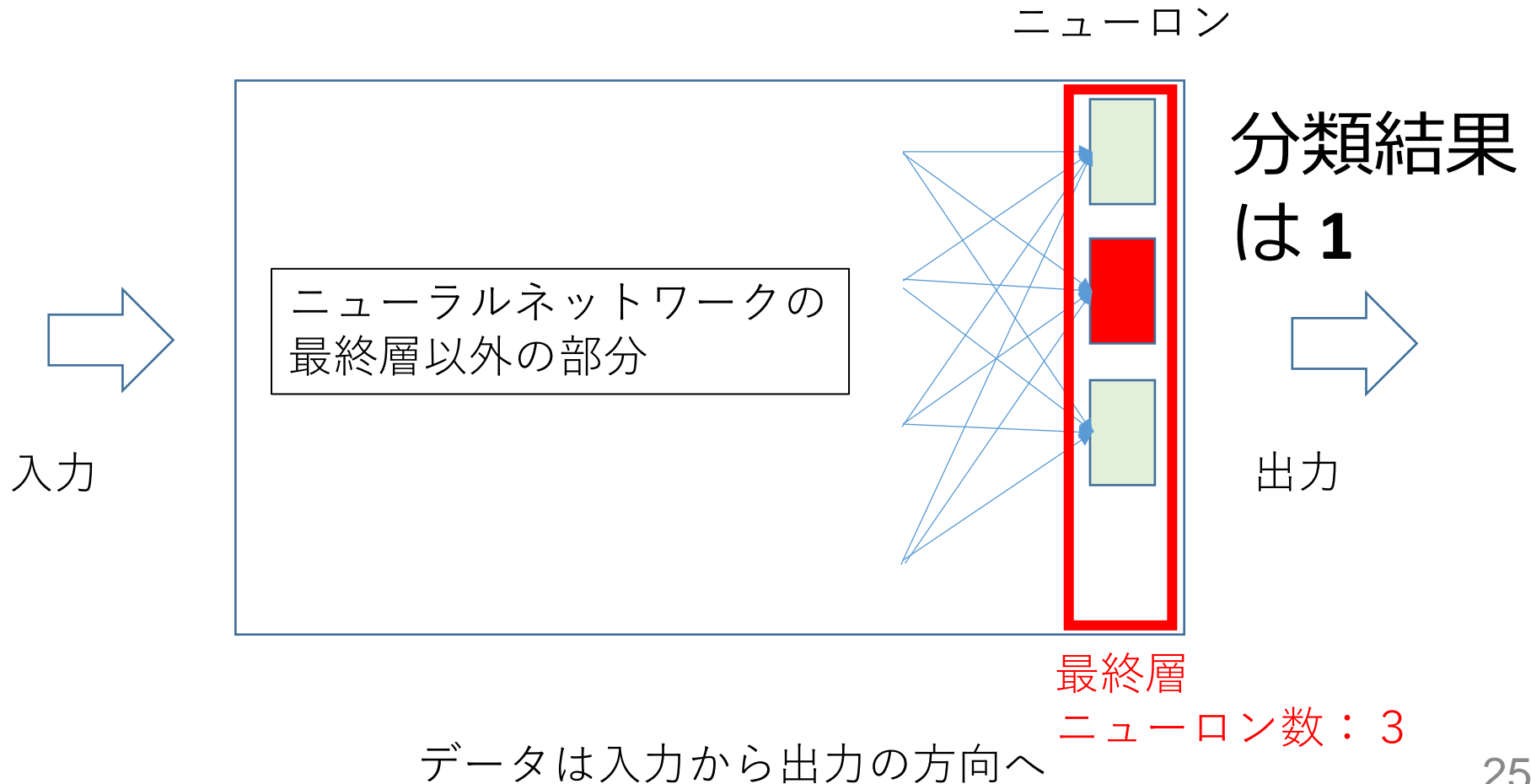
最終層について, 1つが強く 活性化するように調整



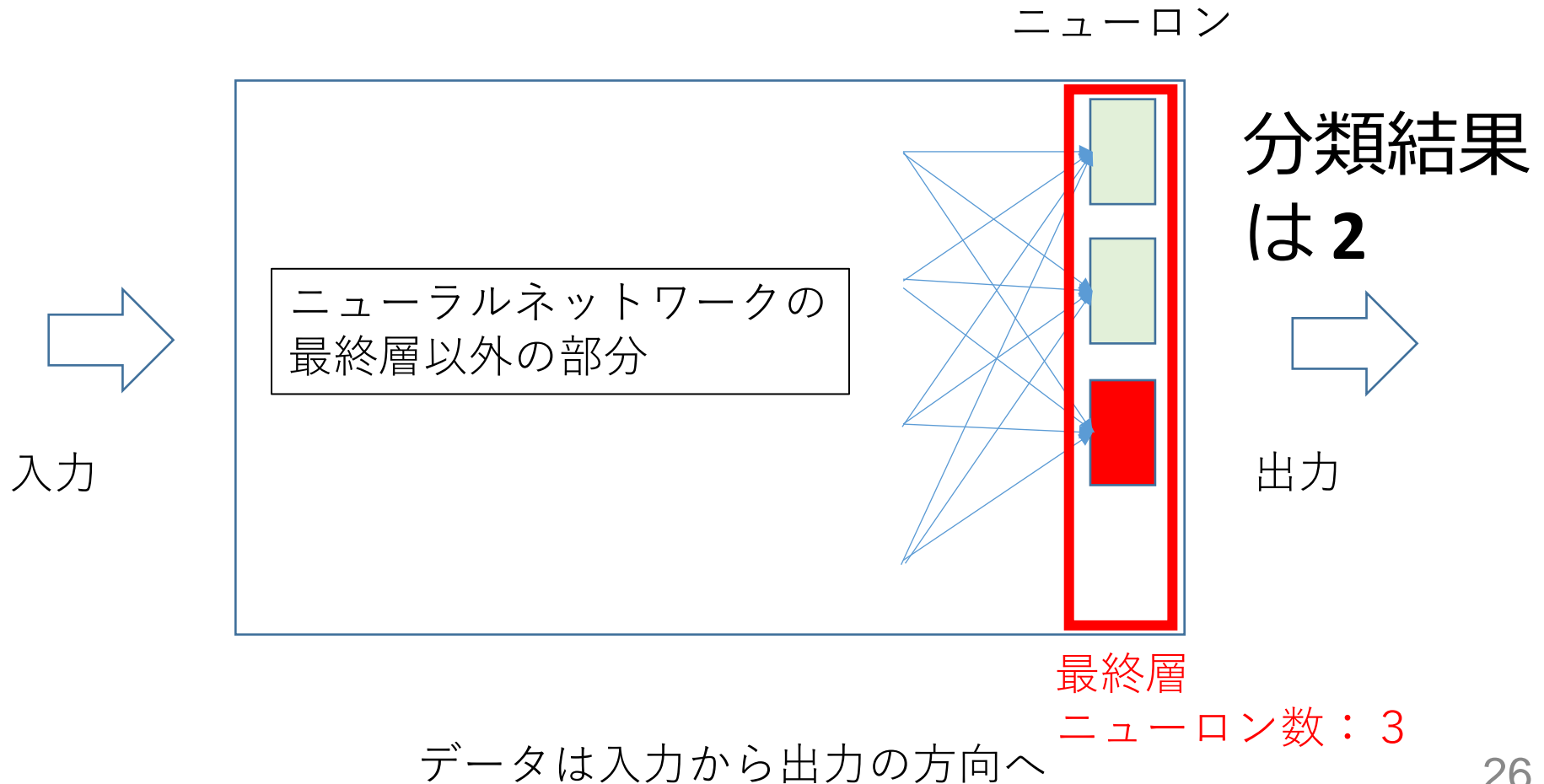
最終層について、1つが強く 活性化するように調整



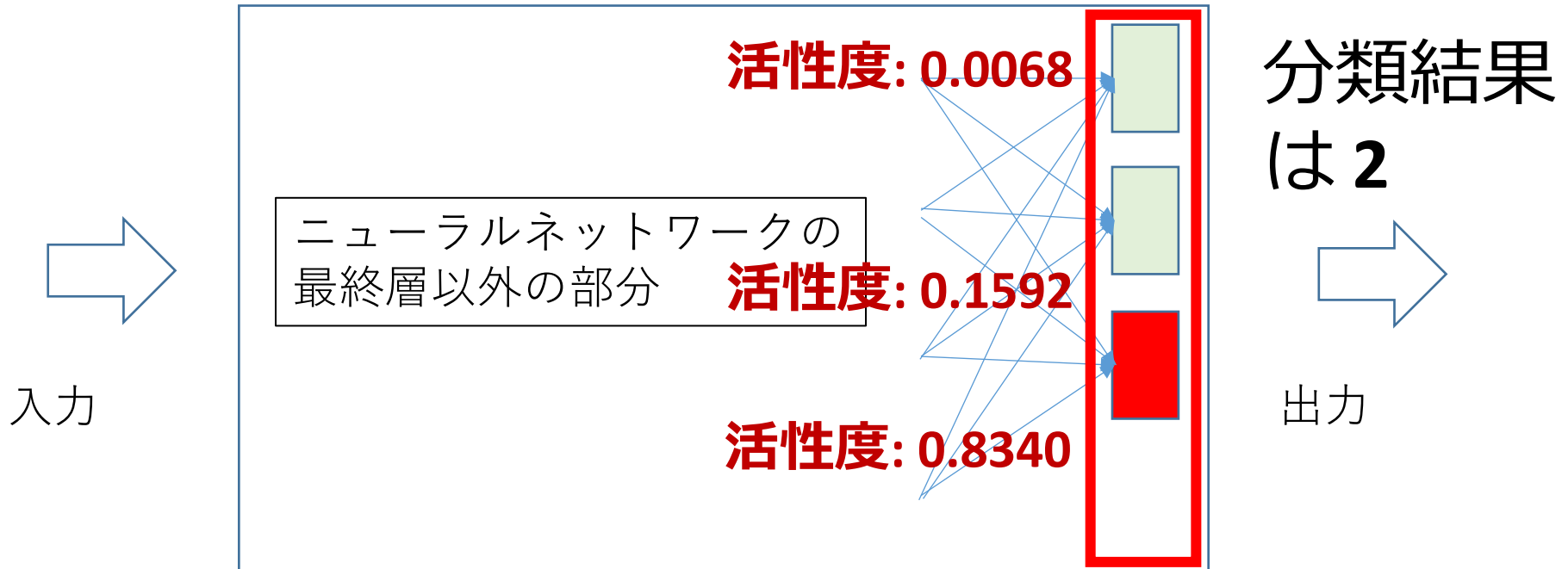
最終層について、1つが強く 活性化するように調整



最終層について、1つが強く 活性化するように調整



最終層について、1つが強く
活性化するように調整



実際には、**活性化度**は 0 から 1 のような数値である。
最も**活性化度**の値が高いものが選ばれて、分類結果となる

ここまでのまとめ



ニューラルネットワークを**分類**に使うことができる

- **最終層**のニューロンで、**最も活性度の値の高いもの**が選ばれて、分類結果となる
- **誤差**がある

6.4 画像分類システム

ここで行うこと



ここでは、画像を、0, 1, 2, 3, 4, 5, 6, 7, 8, 9の**10種類に分類**する



2



4



0



8

画像分類は、提供された画像に対して、各カテゴリ（ラベル）の確率を算出する

画像



画像分類
システム



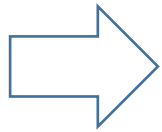
各カテゴリ（ラベル）の確率



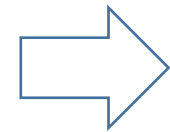
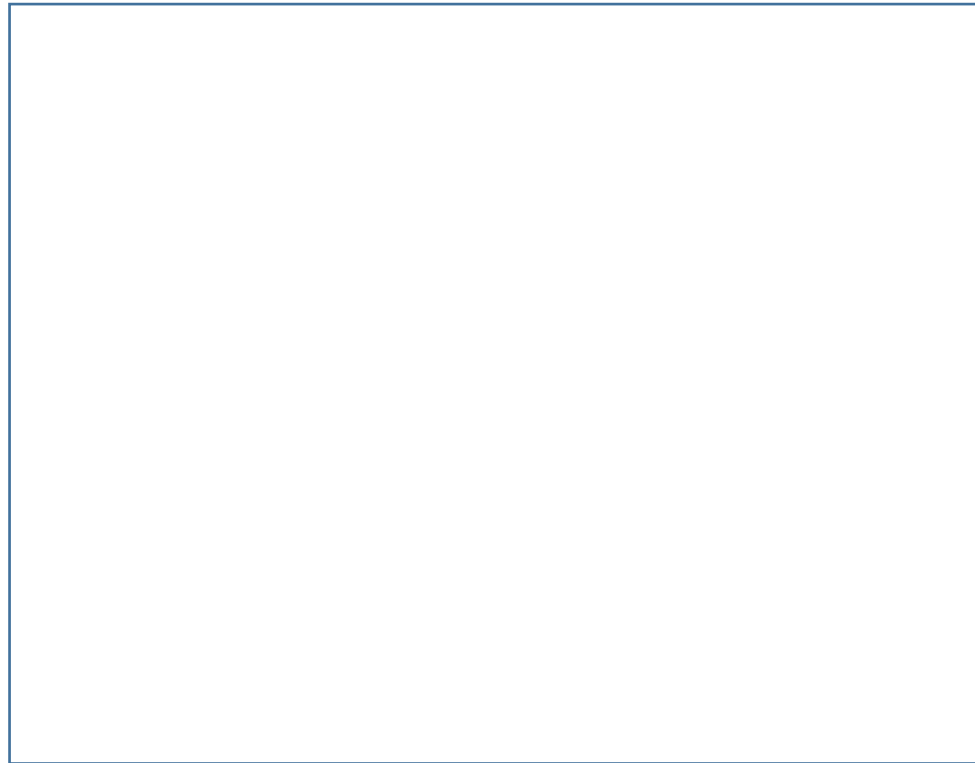
カテゴリ（ラベル）

0	確率	0.0000	...
1	確率	0.0000	...
2	確率	0.9999	...
3	確率	0.0000	...
4	確率	0.0000	...
5	確率	0.0000	...
6	確率	0.0000	...
7	確率	0.0000	...
8	確率	0.0000	...
9	確率	0.0000	...

10種類の中から1つに分類する場合



入力



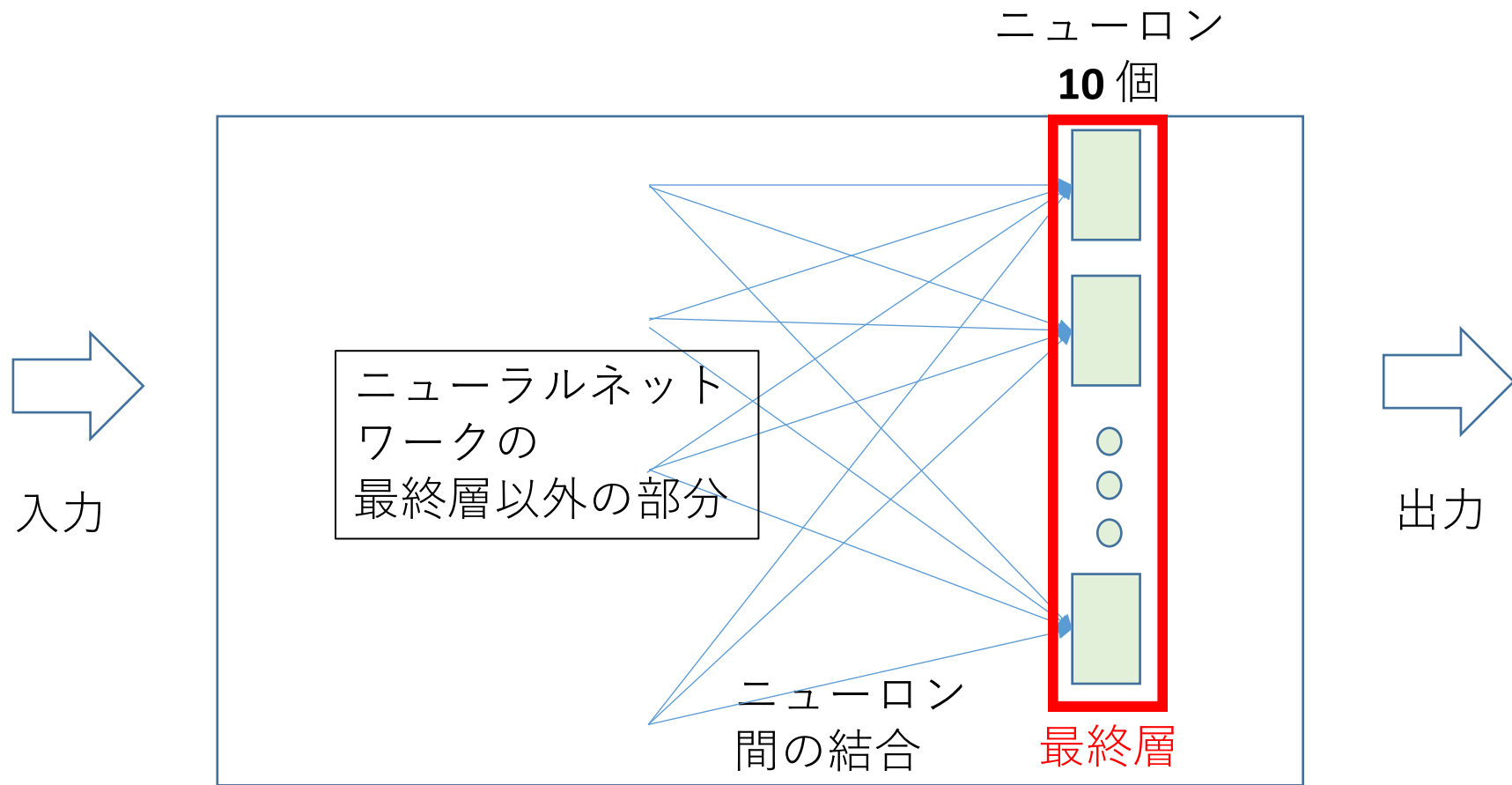
出力

0から9の
整数の
いずれか

10種類に分類するニューラルネットワーク



最終層のニューロン数：**10** にする。

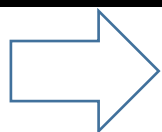
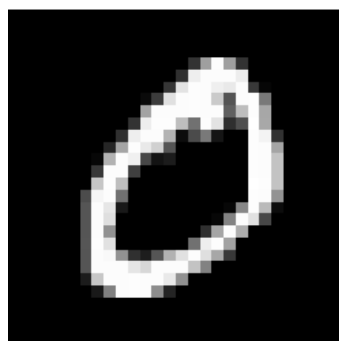


データは入力から出力の方向へ

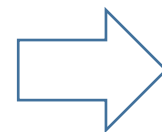
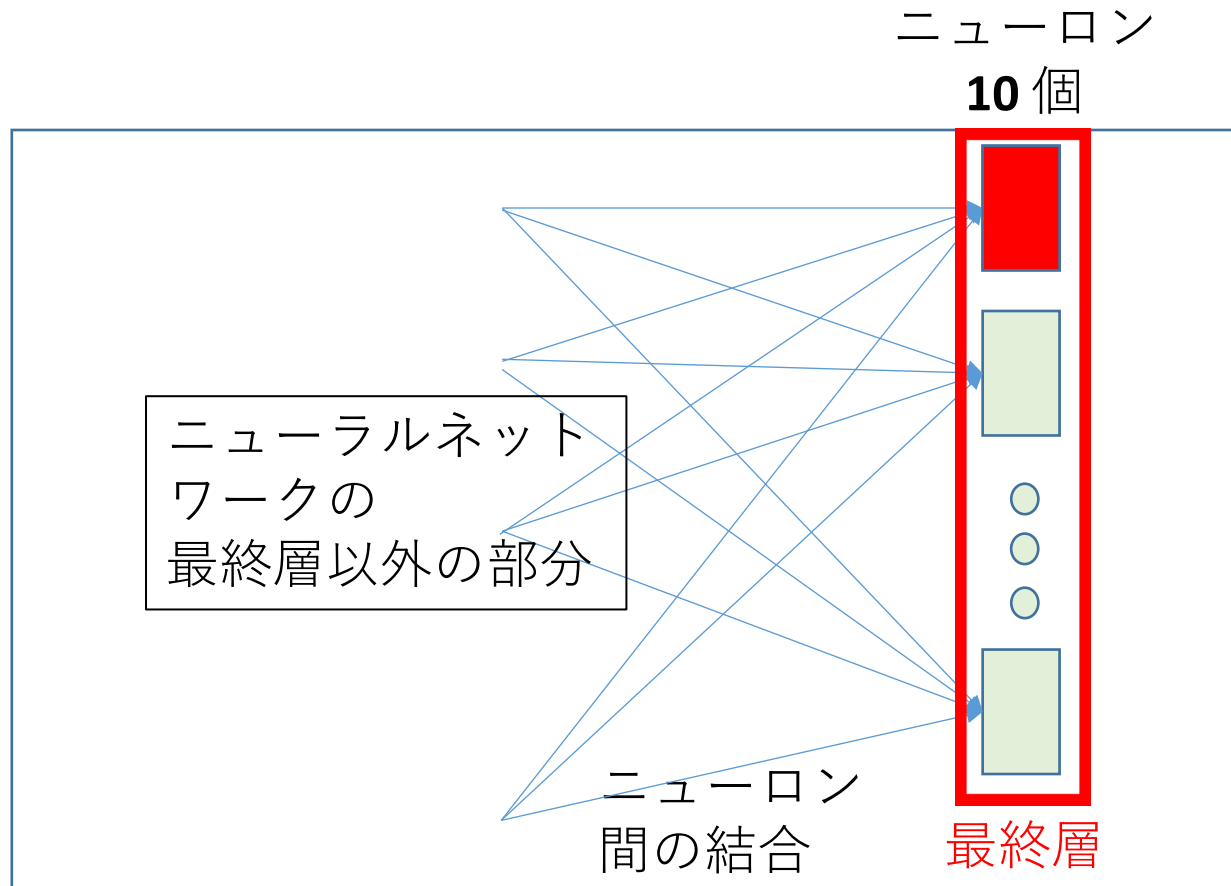
10種類に分類するニューラルネットワークの動作イメージ



最終層のニューロンは1つだけ強く活性化



入力



出力

データは入力から出力の方向へ

画像分類システムのまとめ



- **画像分類**：提供された**画像**を**0から9の10種類に分類**など
- **カテゴリ**：種類を表す名前のこと。「ラベル」ともいう
- **ニューラルネットワークを使用した画像分類**：**最終層のニューロン数を、分類したいカテゴリ数**（たとえば10）に設定。これにより、**各カテゴリに対する確率を出力**できる。
- **最終層のニューロン**の中で**1つだけが強く活性化**する仕組みを利用

画像分類システムは、**手書き文字認識**など、**画像認識**などで活用

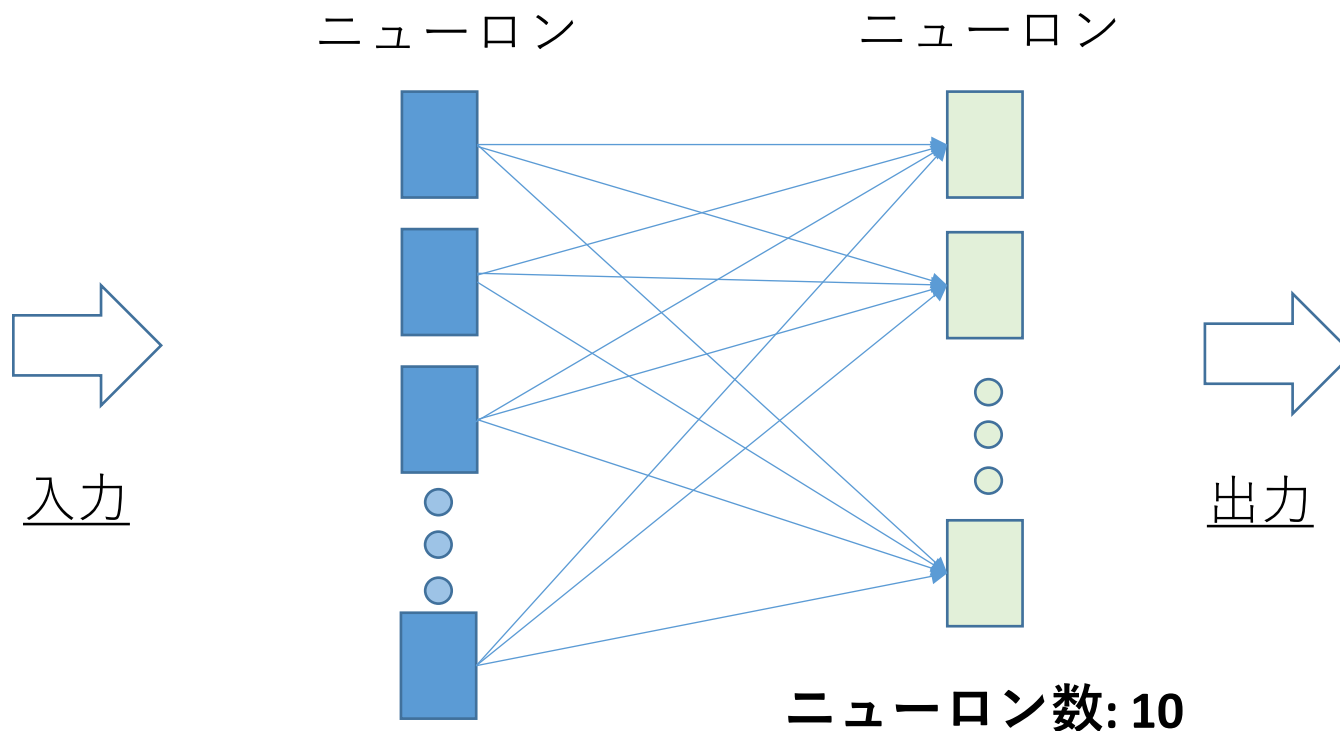
6.5 ニューラルネットワーク の作成

作成するニューラルネットワーク



最終層のニューロン数は 10、残りは自由に考える

- 1 層目：ニューロン数 128
- 2 層目：ニューロン数 10



ニューロン数: 128

ニューロン数: 10

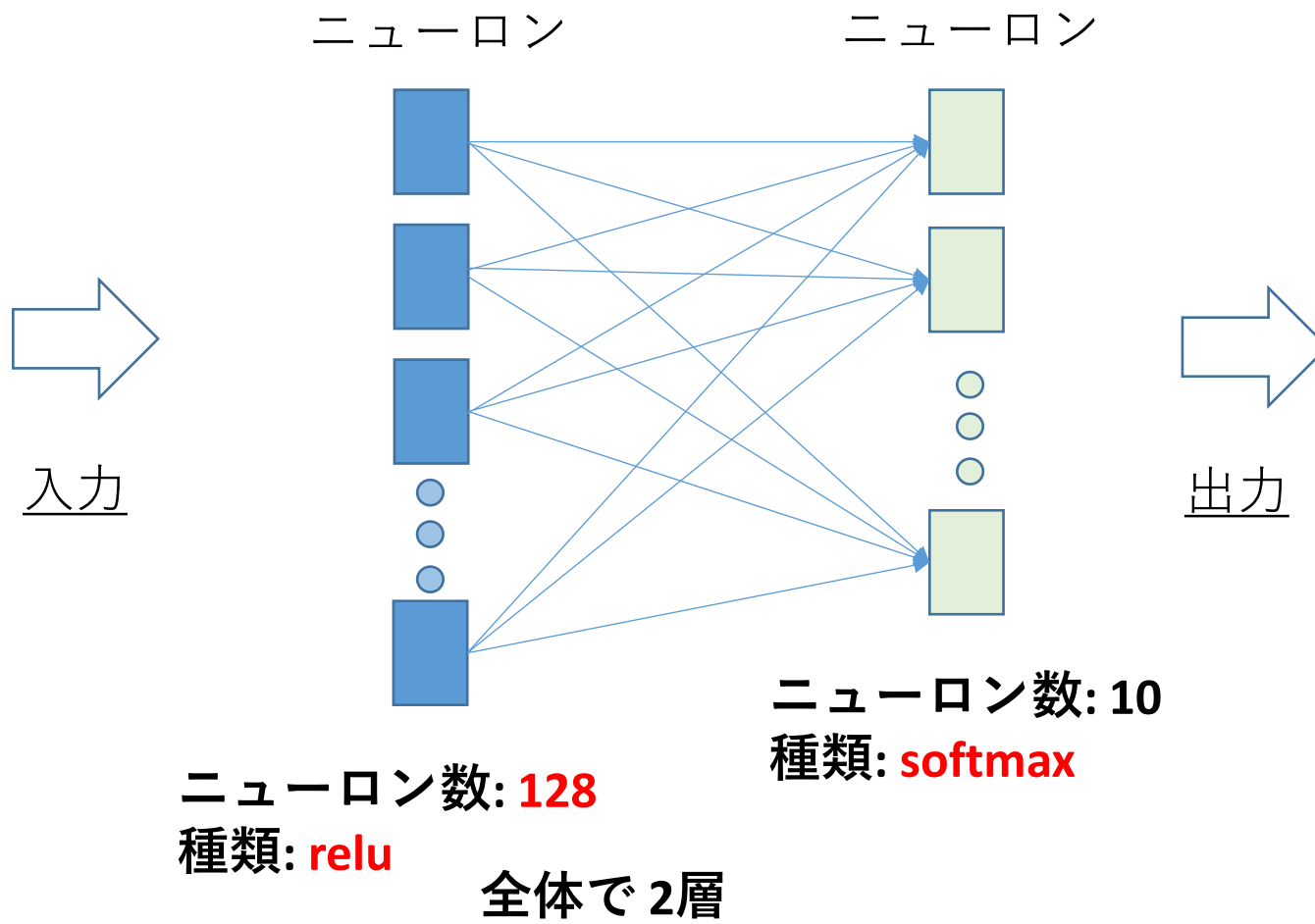
全体で 2 層

ニューロンの種類



ニューロンにはさまざまな種類がある

- relu: 活性化関数が relu
- softmax: 1 層のうち 1 つだけ強く活性化. 分類システムに向く



ニューラルネットワーク作成のプログラム例



入力は **28×28**個の数字

1 層目のニューロン数は **128**
種類は **relu**

```
import tensorflow as tf
m = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(units=128, activation='relu'),
    tf.keras.layers.Dense(units=10, activation='softmax')
])
```

Flatten は、2次元の配列
(アレイ) を、ニューラル
ネットワークの入力にでき
るようにするためのもの

2 層目のニューロン数は **10**
種類は **softmax**

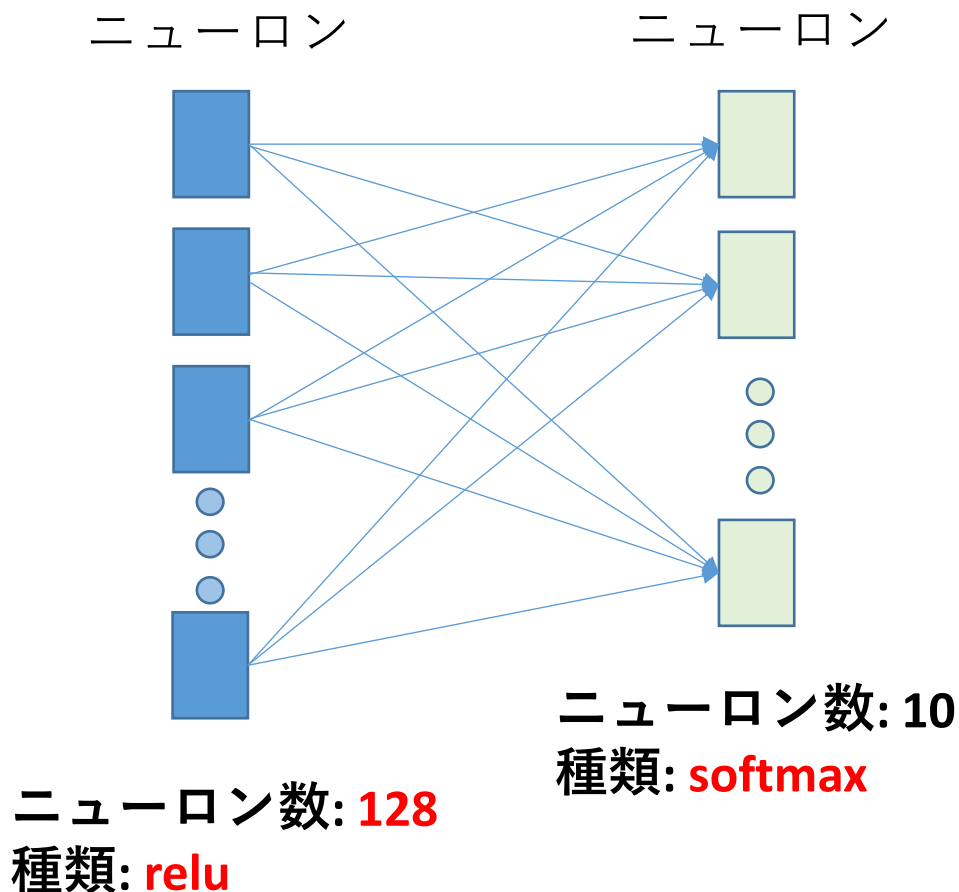
ニューラルネットワーク の作成では、次を設定する

- 入力での**数値の個数**（ここでは画素数）
- **ニューロン**の数（**層**ごと）
- **ニューロン**の種類（**層**ごと）

ニューラルネットワークの作成のまとめ



- ニューラルネットワークによる画像分類システム
- 入力での数値の数、ニューロンの数、ニューロンの種類を指定して、**ニューラルネットワークを作成**



高度なパターン認識やカラー画像の分類でも同様に可能

6.6 学习

ニューラルネットワークの学習のための準備



- 訓練データ
- 検証データ

訓練データと検証データ



訓練データ： **学習**に使用

訓練データによる**学習**により，**訓練データ**ではないデータでも分類できる能力（「汎化」という）を獲得

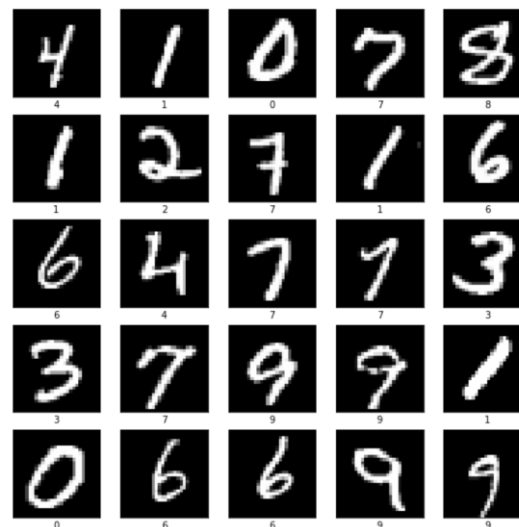
検証データ： **学習の結果を確認**するためのもの。
訓練データとは違うものを使用する。

学習のための準備



• 訓練データ

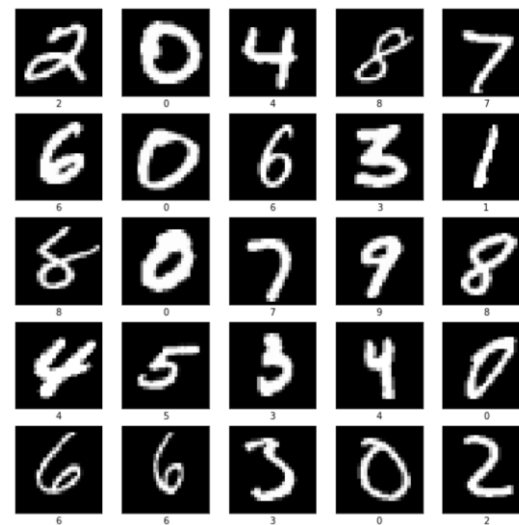
60000枚の画像と正解



抜粋

• 検証データ

10000枚の画像と正解



抜粋

ニューラルネットワークの学習

訓練データを使用

- ① 訓練データにより、ニューラルネットを動作させる
- ② ①の結果と、正解を照合し、誤差を得る
- ③ ニューロン間の結合の重みのバイアスの調整により、誤差を減らす

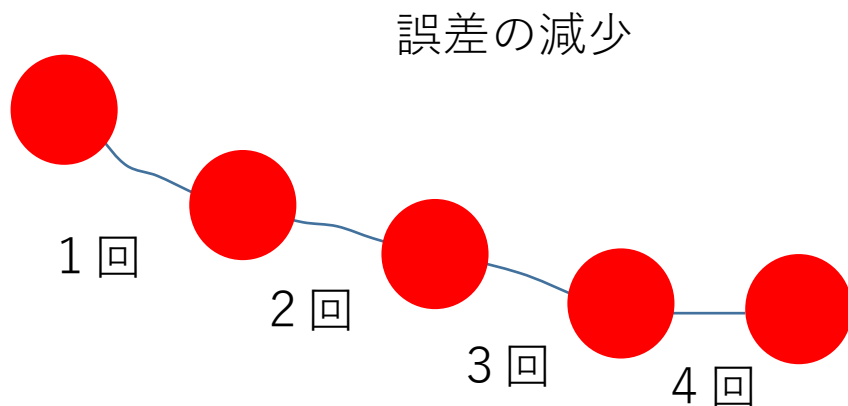
学習の繰り返し



同じ訓練データを繰り返し使用

訓練データを1回使っただけでは、**学習不足**の場合がある。

同じ訓練データを**繰り返し**使用することで、誤差をさらに減らす



学習の繰り返しを行うプログラム例



学習の繰り返し回数は **20**

```
EPOCHS=20  
history = m.fit(x=ds_train[0],  
                y=ds_train[1],  
                epochs=EPOCHS,  
                validation_data=(ds_test[0], ds_test[1]),  
                callbacks=[tensorboard_callback], verbose=2)
```

訓練データの指定

検証データの指定

学習の繰り返しを行うプログラムと実行結果



同じ訓練データを用いた学習を20回繰り返し。
そのとき、検証データで検証

```
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

EPOCHS=20
history = m.fit(x=ds_train[0],
                y=ds_train[1],
                epochs=EPOCHS,
                validation_data=(ds_test[0], ds_test[1]),
                callbacks=[tensorboard_callback], verbose=2)

m.evaluate(ds_test[0], ds_test[1], verbose=2)
```

```
Epoch 1/20
1875/1875 - 5s - loss: 0.2589 - accuracy: 0.9262 - val_loss: 0.1427 - val_accuracy: 0.9594 - 5s/epoch - 3ms/step
Epoch 2/20
1875/1875 - 4s - loss: 0.1151 - accuracy: 0.9668 - val_loss: 0.0944 - val_accuracy: 0.9713 - 4s/epoch - 2ms/step
Epoch 3/20
1875/1875 - 4s - loss: 0.0778 - accuracy: 0.9763 - val_loss: 0.0808 - val_accuracy: 0.9736 - 4s/epoch - 2ms/step
Epoch 4/20
1875/1875 - 4s - loss: 0.0573 - accuracy: 0.9828 - val_loss: 0.0786 - val_accuracy: 0.9755 - 4s/epoch - 2ms/step
Epoch 5/20
1875/1875 - 4s - loss: 0.0450 - accuracy: 0.9859 - val_loss: 0.0743 - val_accuracy: 0.9765 - 4s/epoch - 2ms/step
Epoch 6/20
1875/1875 - 4s - loss: 0.0343 - accuracy: 0.9892 - val_loss: 0.0762 - val_accuracy: 0.9756 - 4s/epoch - 2ms/step
Epoch 7/20
1875/1875 - 4s - loss: 0.0285 - accuracy: 0.9912 - val_loss: 0.0720 - val_accuracy: 0.9791 - 4s/epoch - 2ms/step
Epoch 8/20
1875/1875 - 4s - loss: 0.0226 - accuracy: 0.9931 - val_loss: 0.0770 - val_accuracy: 0.9785 - 4s/epoch - 2ms/step
Epoch 9/20
1875/1875 - 4s - loss: 0.0183 - accuracy: 0.9949 - val_loss: 0.0774 - val_accuracy: 0.9792 - 4s/epoch - 2ms/step
Epoch 10/20
1875/1875 - 4s - loss: 0.0150 - accuracy: 0.9955 - val_loss: 0.0797 - val_accuracy: 0.9784 - 4s/epoch - 2ms/step
Epoch 11/20
1875/1875 - 4s - loss: 0.0141 - accuracy: 0.9956 - val_loss: 0.0828 - val_accuracy: 0.9784 - 4s/epoch - 2ms/step
Epoch 12/20
1875/1875 - 4s - loss: 0.0107 - accuracy: 0.9965 - val_loss: 0.0821 - val_accuracy: 0.9786 - 4s/epoch - 2ms/step
Epoch 13/20
1875/1875 - 4s - loss: 0.0106 - accuracy: 0.9967 - val_loss: 0.0914 - val_accuracy: 0.9774 - 4s/epoch - 2ms/step
Epoch 14/20
1875/1875 - 4s - loss: 0.0085 - accuracy: 0.9974 - val_loss: 0.0931 - val_accuracy: 0.9766 - 4s/epoch - 2ms/step
Epoch 15/20
1875/1875 - 4s - loss: 0.0068 - accuracy: 0.9981 - val_loss: 0.0915 - val_accuracy: 0.9781 - 4s/epoch - 2ms/step
Epoch 16/20
1875/1875 - 4s - loss: 0.0076 - accuracy: 0.9977 - val_loss: 0.0885 - val_accuracy: 0.9798 - 4s/epoch - 2ms/step
Epoch 17/20
1875/1875 - 5s - loss: 0.0065 - accuracy: 0.9980 - val_loss: 0.0965 - val_accuracy: 0.9792 - 5s/epoch - 3ms/step
Epoch 18/20
1875/1875 - 5s - loss: 0.0060 - accuracy: 0.9981 - val_loss: 0.0936 - val_accuracy: 0.9785 - 5s/epoch - 3ms/step
Epoch 19/20
1875/1875 - 4s - loss: 0.0063 - accuracy: 0.9981 - val_loss: 0.1017 - val_accuracy: 0.9784 - 4s/epoch - 2ms/step
Epoch 20/20
1875/1875 - 4s - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.1105 - val_accuracy: 0.9761 - 4s/epoch - 2ms/step
313/313 - 0s - loss: 0.1105 - accuracy: 0.9761 - 394ms/epoch - 1ms/step
[0.1105121374190249, 0.9761000275611877]
```

プログラム

実行結果

学習の繰り返しにおける、
訓練データや検証データ
での**誤差**
の変化を確認

ニューラルネットワークの学習のまとめ



ニューラルネットワークの学習では、訓練データと検証データが必要

- **訓練データ**：学習に使用（訓練データ以外のデータでも正確に分類できる能力を獲得）
- **検証データ**：学習の結果を確認するために使用。訓練データとは異なるデータを使うこと。

ニューラルネットワークの学習の仕組み

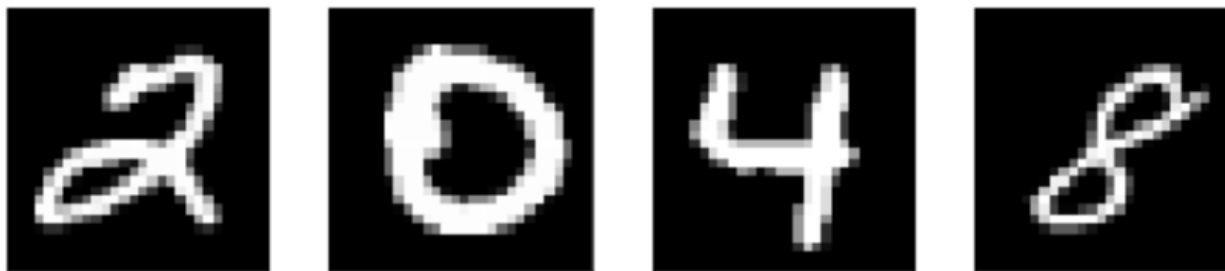
- ① 実データを使用して**ニューラルネットワークを動作**させる。
- ② **動作結果と正解を比較し、誤差を計算**
- ③ ニューロン間の**結合の重みとバイアス**を調整し、**誤差を減らす**
- ④ 同じ訓練データを**繰り返し使用**して、**誤差をさらに減らす**

6.7 画像分類

画像分類の結果



4 つの画像を分類



分類結果 10 個の数値が 4 つ

[2.82898581e-20 2.11713194e-20 1.00000000e+00 1.01542401e-08	2
3.07505820e-18 6.13309550e-19 1.73079867e-17 3.75218205e-16	
2.25546036e-12 1.27005634e-19]	
[1.00000000e+00 0.00000000e+00 7.43346550e-24 7.99614704e-31	0
1.60335865e-35 7.09844889e-24 1.09644683e-16 3.93163016e-20	
4.51085197e-28 9.15929917e-33]	
[4.02610817e-21 5.27186802e-21 1.43940942e-19 6.05407705e-22	
1.00000000e+00 1.87728168e-24 1.43710434e-20 5.10361284e-13	4
1.50390224e-20 5.03197449e-11]	
[1.59357307e-06 3.12464854e-09 4.91066432e-09 9.94732474e-09	
1.01848738e-11 1.25744277e-08 1.01212549e-08 3.50234806e-11	
9.99998331e-01 2.44763920e-09]	8

プログラムは、次で公開

<https://colab.research.google.com/drive/1IfArlvhh-FsvJIE9YTNO8T44Qhpi0rIJ?usp=sharing>

- 実行結果とプログラムと説明
- **（これは必須ではありません） Google アカウントがあれば、プログラムを変更し再実行できます。**

Google アカウントの取得法



- 次のページを使用

<https://accounts.google.com/SignUp>

- 次の情報を登録する

氏名

自分が希望するメールアドレス

<ユーザー名> [@gmail.com](mailto:kanekokunihiko12112@gmail.com)

パスワード

生年月日, 性別



Google アカウントの作成

姓	名
<input type="text" value="金子"/>	<input type="text" value="邦彦"/>

ユーザー名

半角英字、数字、ピリオドを使用できます。

選択可能なユーザー名:

[bangyanjinzi6](#) [jinzibangyan6](#) [kanekokunihiko72](#)

代わりに現在のメールアドレスを使用


パスワード	確認
<input type="password" value="....."/>	<input type="password" value="....."/>

半角英字、数字、記号を組み合わせ 8 文字以上で入力してください

[代わりにログイン](#)

[次へ](#)

- **機械学習**では、**データから自動的にパターンや関連性を見つけ出す**ため、**ルールや知識のプログラム化は不要**
- **ニューラルネットワーク**は**機械学習の一種**
- ニューラルネットワークは、**最終層のニューロン**で最も活性度の高いものを選ぶことで、**画像を分類**します
- **学習**を通じて**ニューラルネットワーク**は**知的能力を向上**
- ニューラルネットワークの学習と画像分類については、**TensorFlowの公式チュートリアル**などを利用して実践的な学びが可能
- **画像分類**は、**手書き文字認識**や**パターン認識**などで**広く活用**されている

- 
- ① AIの基礎を理解。「AIを使いこなせる」という**自信、満足感が向上**。これは、現代社会で欠かせないスキル。
 - ② 画像分類技術は、すでに**実社会の問題解決に応用されている**
 - ③ ニューラルネットワークの学習過程を理解。**AIの可能性と限界について理解が深まり、視野が広がる。**
 - ④ 習得した知識とスキルは、**画像類システムの作成や活用に直結。身の回りの問題解決に役立つ。**