12-1 第１２回の内容

（情報システム工学特論）

URL: https://www.kkaneko.jp/a/cs/index.html

金子邦彦

第１２回の内容

- **ニューラルネットワーク**の**基礎**について，理解を深める

  学習，教師データ，検証データ，学習不足，過学習

- オンライン（Web ブラウザを使用）の実行.（各自で実行することも可能）

【次回に向けての準備学習】
次回はニューラルネットワークについて，さらに詳しく学ぶ．今回の資料を復習しておく．

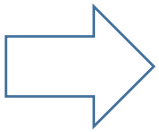
12-2 ニューラルネットワークを用いた分類

（人工知能の基本）

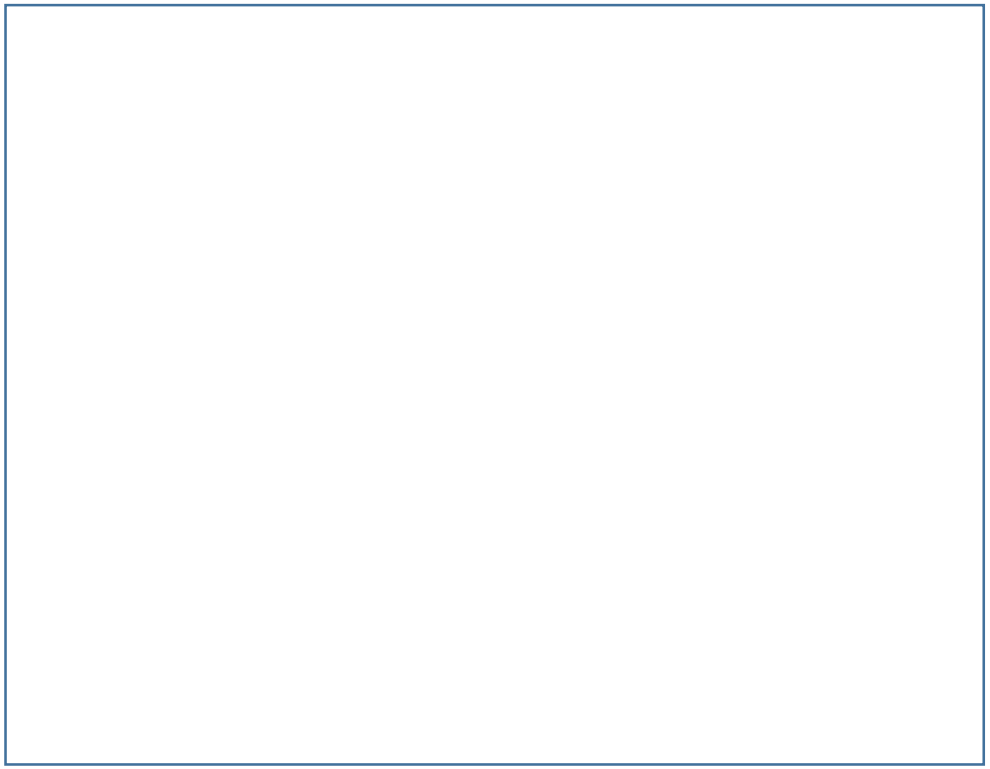
URL: https://www.kkaneko.jp/a/cs/index.html

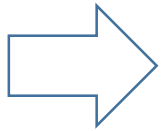
金子邦彦

１０種類の中から１つに分類する場合



入力データ



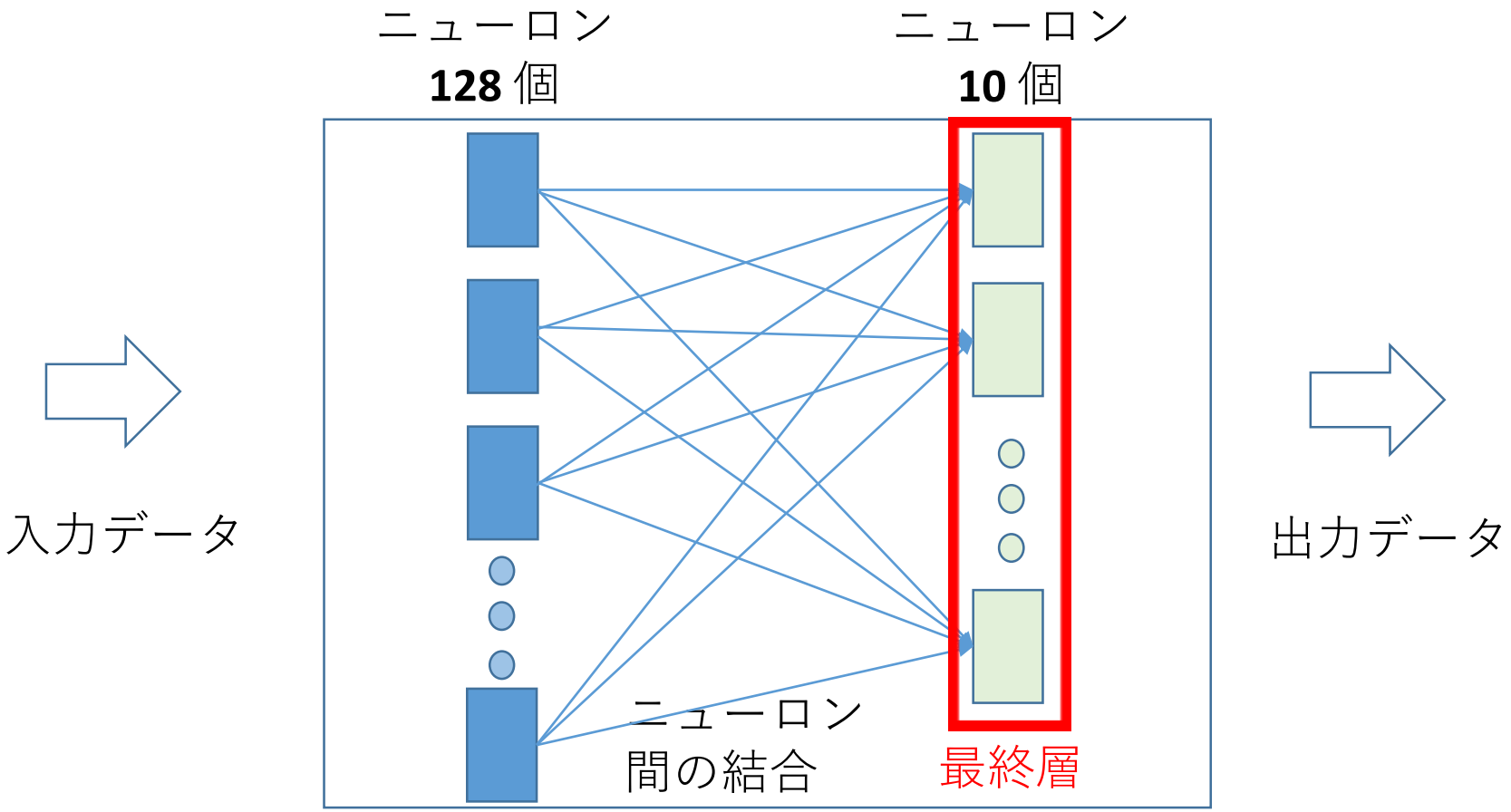


出力データ

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

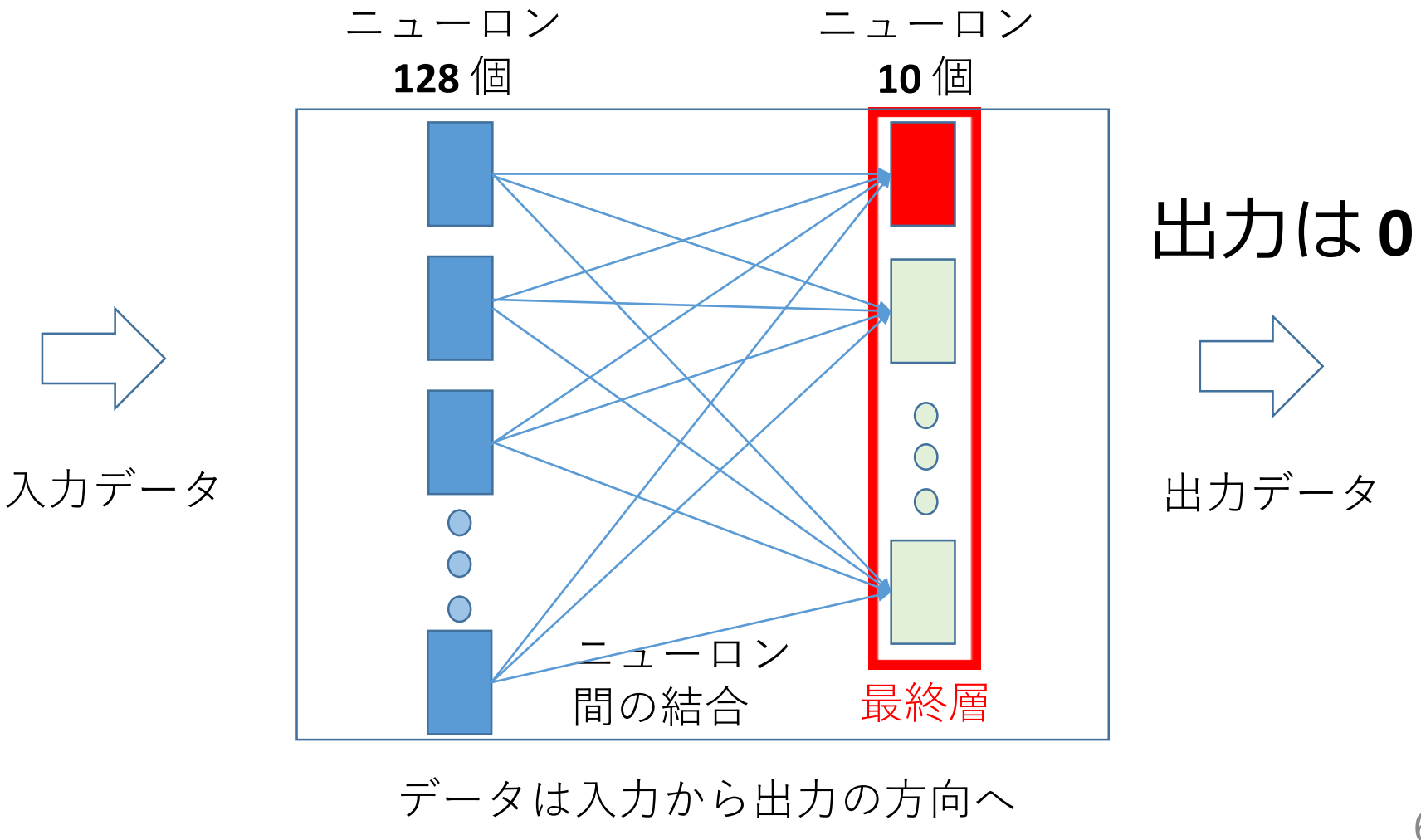
最終層について，１つが強く
活性化するように調整



データは入力から出力の方向へ

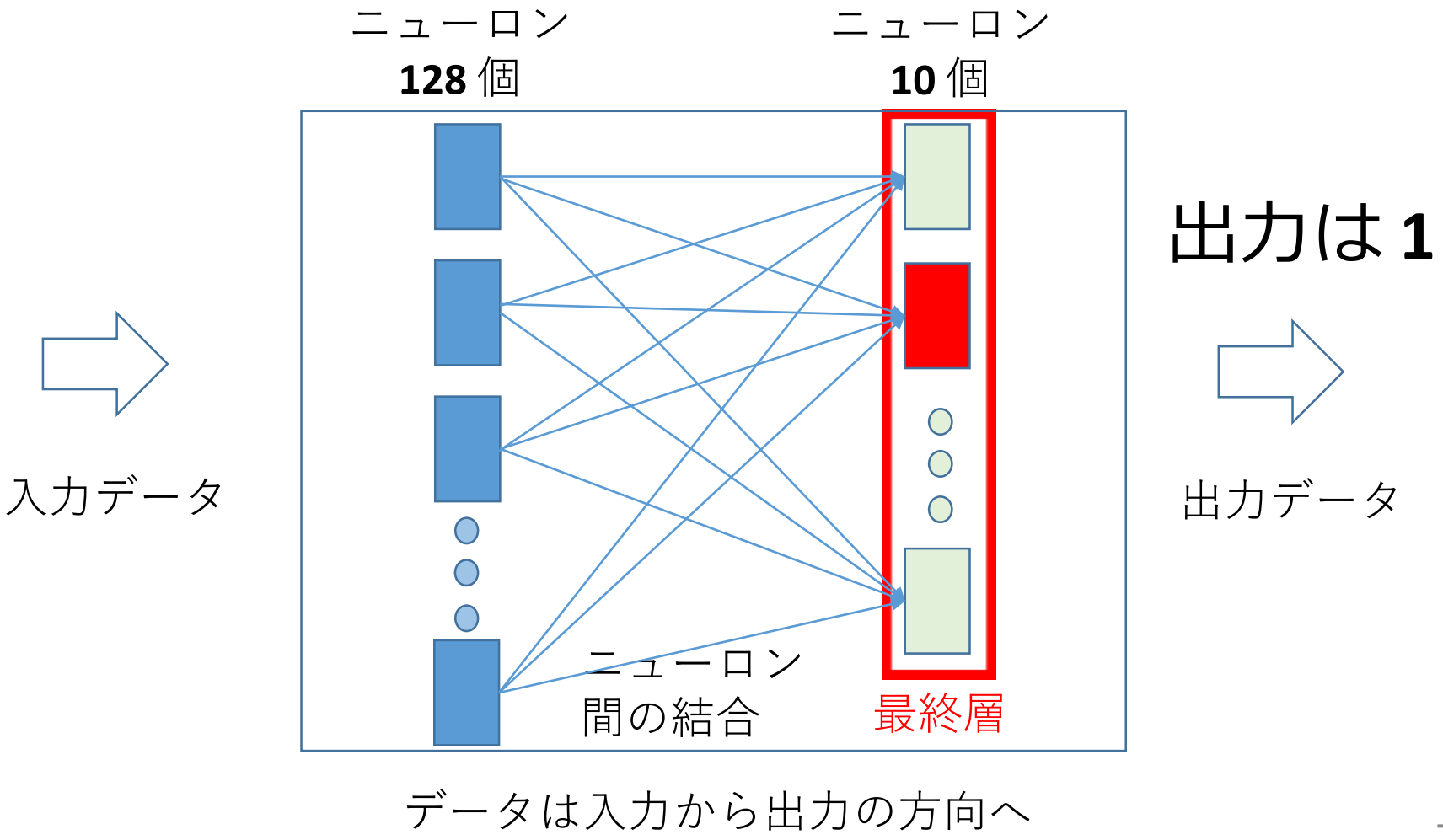
１０種類に分類するニューラルネットワーク

最終層について，１つが強く
活性化するように調整



１０種類に分類するニューラルネットワーク

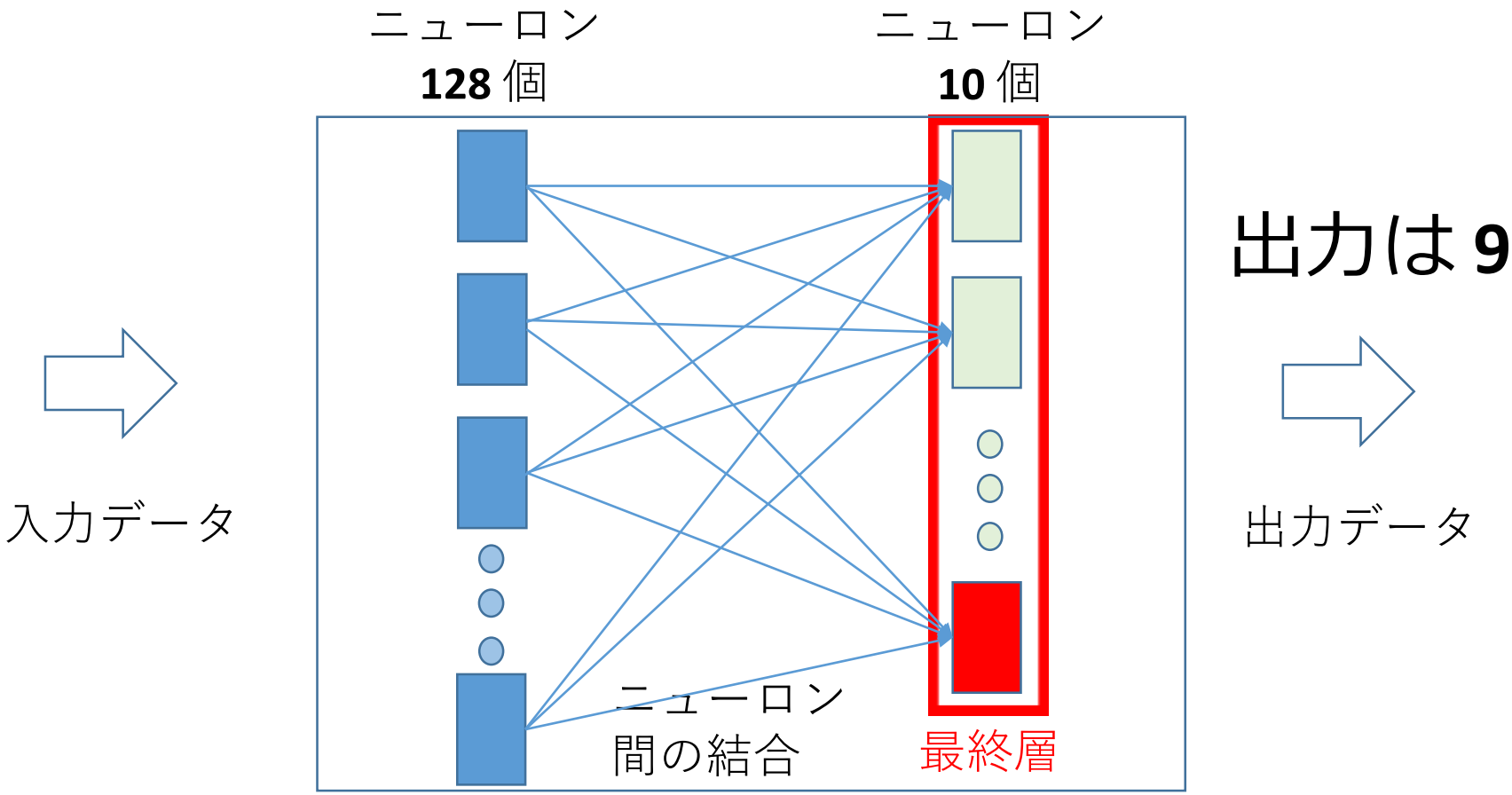
最終層について，１つが強く
活性化するように調整



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

最終層について，１つが強く
活性化するように調整



データは入力から出力の方向へ

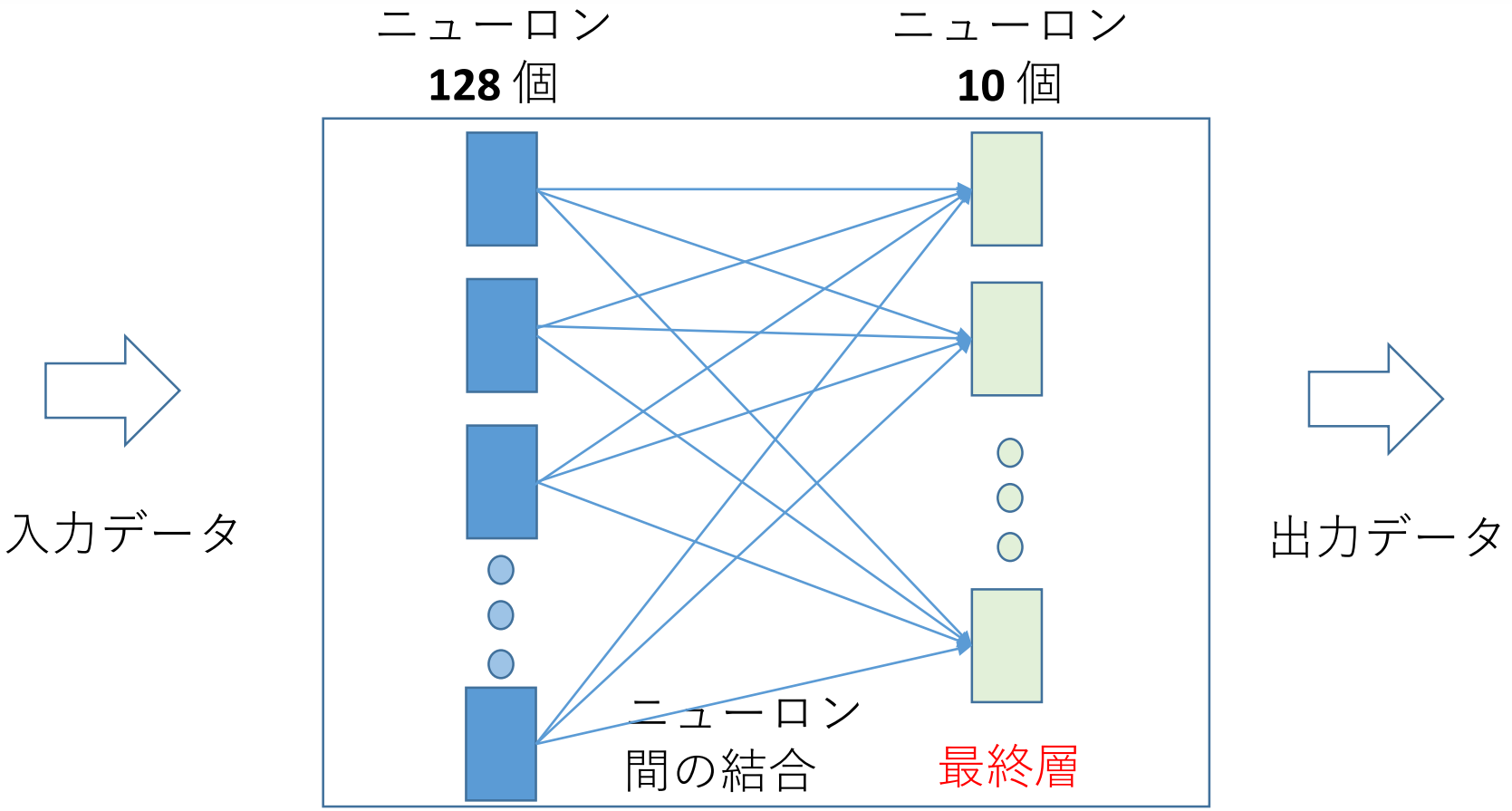
12-3 ニューラルネットワークの学習

（人工知能の基本）

URL: https://www.kkaneko.jp/a/cs/index.html

金子邦彦

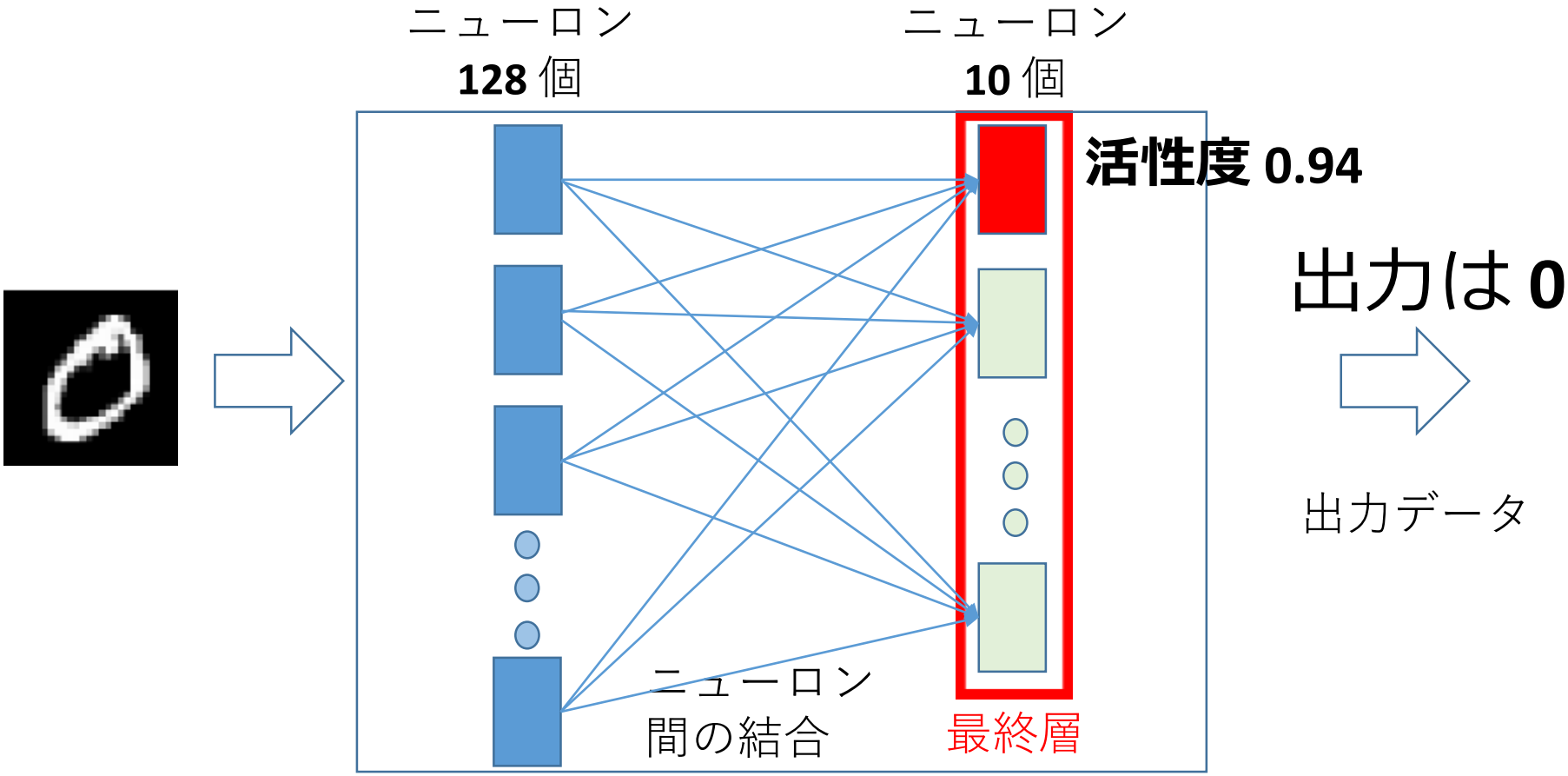
層が直列になっているニューラルネットワーク



データは入力から出力の方向へ

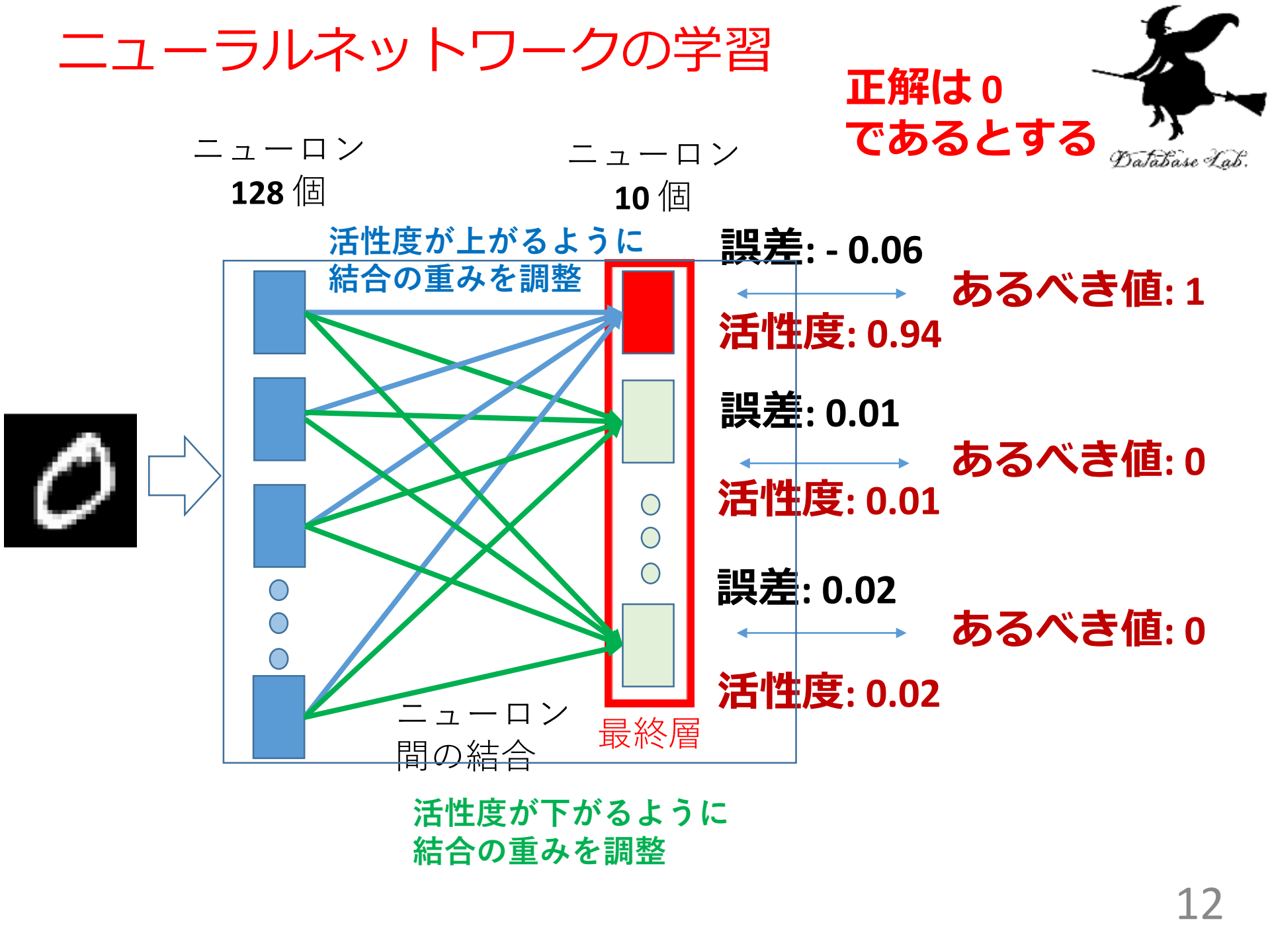
１０種類に分類するニューラルネットワーク

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，活性度は０から１のような数値である．
最も**活性度**の**値が高いもの**が選ばれて，分類結果となる

ニューラルネットワークの学習



ニューラルネットワークの学習

- **教師データ**（学習のためのデータ）を使用
- **学習**は**自動**で行われる

① **教師データ**により，**ニューラルネット**を動かし，誤差を得る

② **ニューロン間の結合の重み**の**上げ下げ**により、**誤差**を減らす （最終層の結果が，手前の層の結合の重みに伝搬することから，フィードバックともいわれる）

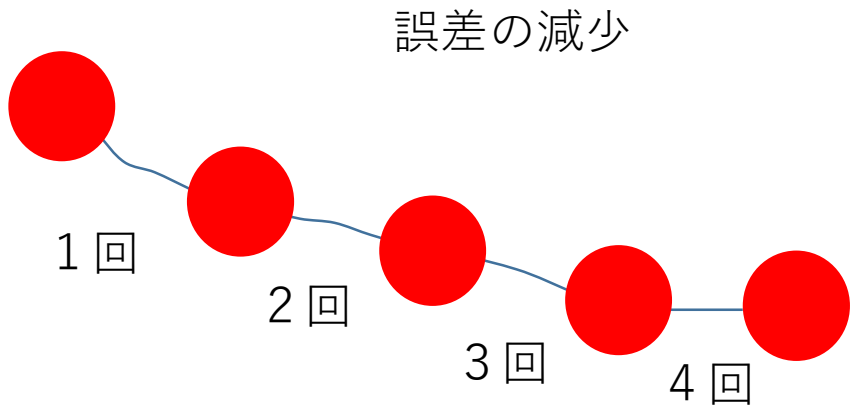
- ニューロンの数が増えたり減ったりなどではない
- **誤差**が**減らなくなったら**、**最適**になったとみなす

学習不足

- **ニューラルネットワーク**の**学習**では，学習のためのデータ（**教師データ**）を使う

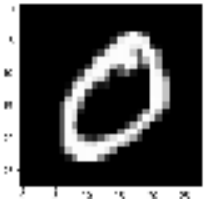
- **教師データ**を１回使っただけでは，**学習不足**の場合がある

→　同じ**教師データ**を繰り返し使って学習を行う．
　　繰り返しながら，誤差の減少を確認



学習と検証

教師データ：　**学習**に使用



ニューラル
ネットワーク

検証データ：　**学習の結果を確認するためのもの.**
教師データとは別に準備

ニューラル
ネットワーク

前準備

Google アカウントの取得が必要

- 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

パスワード

生年月日，性別

手順

① パソコンの Web ブラウザで，次のページを開く

https://www.tensorflow.org/tutorials

② 左側のメニューの「**初心者向けクイックスタート**」をクリック

手順

③「Run in Google Colab」をクリック

④ セルを**上から順に実行**する.

セルの実行の終了を確認してから、次のセルに移ること

```
[ ] import tensorflow as tf
```

MNIST データセットをロードして準備します。サンプルを整数から浮動小数点数に変換します。

```
[ ] mnist = tf.keras.datasets.mnist

    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train, x_test = x_train / 255.0, x_test / 255.0
```

層を積み重ねて tf.keras.Sequential モデルを構築します。訓練のためにオプティマイザと損失関数を選びます

```
[ ] model = tf.keras.models.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Dense(10)
    ])
```

モデルはそれぞれの標本について、クラスごとに"ロジット"や"対数オッズ比"と呼ばれるスコアを算出します。

```
[ ] predictions = model(x_train[:1]).numpy()
    predictions
```

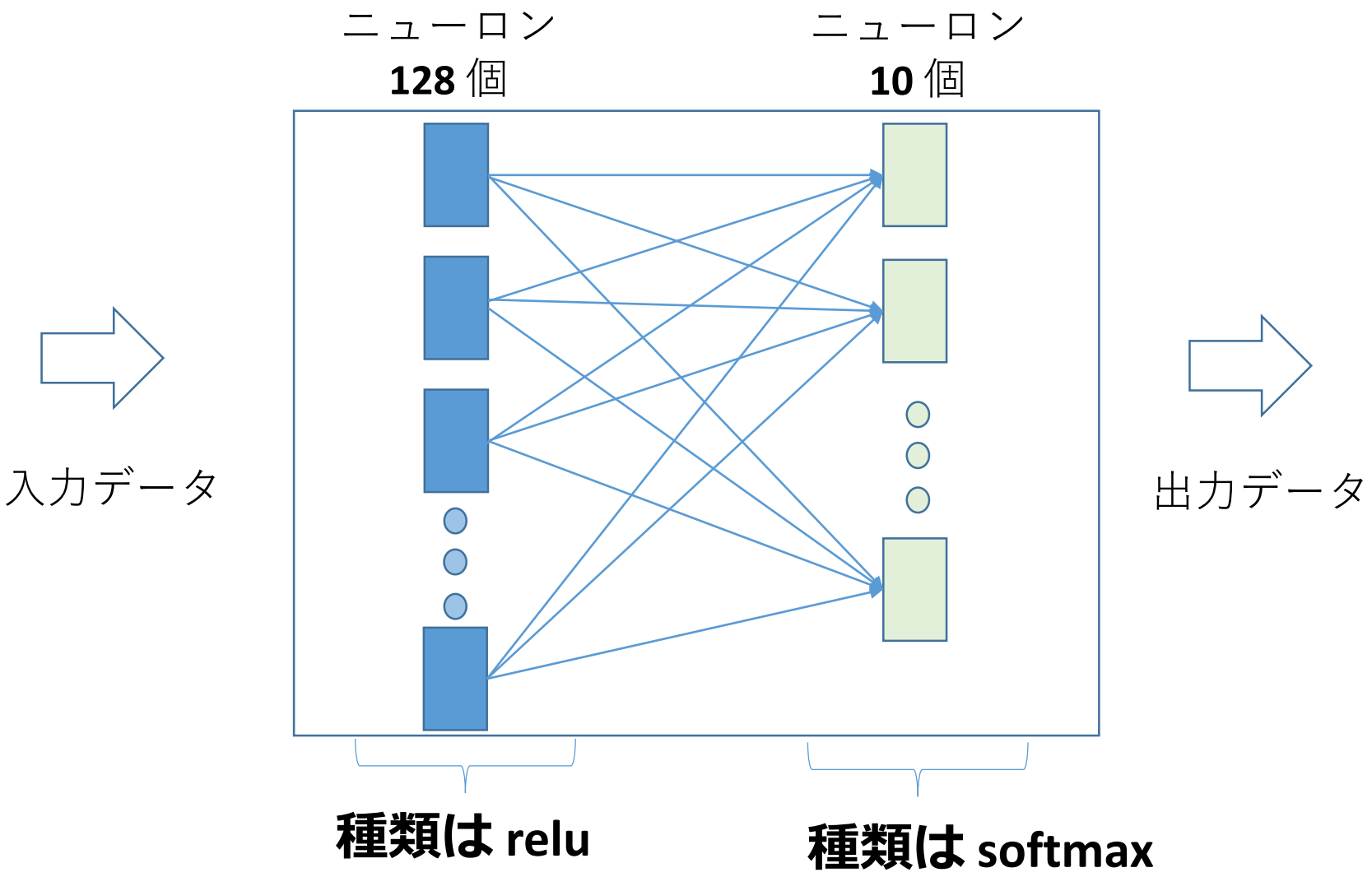
tf.nn.softmax 関数はクラスごとにこれらのロジットを "確率" に変換します。

```
[ ] tf.nn.softmax(predictions).numpy()
```

Google
アカウント
が必要

最後まで続ける

ニューラルネットワークの例



ニューラルネットワークを作成するプログラム


```
[ ]  model = tf.keras.models.Sequential([
       tf.keras.layers.Flatten(input_shape=(28, 28)),
       tf.keras.layers.Dense(128, activation='relu'),
       tf.keras.layers.Dropout(0.2),
       tf.keras.layers.Dense(10)
     ])
```

ニューラルネットワークの学習の様子

同じ**教師データ**を繰り返し使って学習を行う

```
model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3021 - accuracy: 0.9130
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1451 - accuracy: 0.9564
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1076 - accuracy: 0.9677
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0891 - accuracy: 0.9726
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0765 - accuracy: 0.9768
<tensorflow.python.keras.callbacks.History at 0x7f5813df5828>
```

繰り返し回数：**5回**

繰り返しのたびに**誤差が減少**（loss の右横の数値）

検証

Model.evaluate メソッドはモデルの性能を検査します。これには通常 "検証用データセット" または "テストデータセット" を用います。

```
[9]  model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.0707 - accuracy: 0.9775
[0.07074886560440063, 0.9775000214576721]
```

- 検証結果として「loss: 0.0707」のように表示
  この値は**（正解との差）**
  0：良い、　1：良くない

学習では、乱数が使用されるので、学習ごとに loss の値は変わる

ニューラルネットワークによる分類の様子

- **５つの画像**、それぞれについて文字認識

０，１，２，３，４，５，６，７，８，９で

ある**確率**がいくらかが得られる

```
probability_model(x_test[:5])
```

```
<tf.Tensor: shape=(5, 10), dtype=float32, numpy=
array([[1.91370447e-07, 1.24967245e-08, 1.01994574e-05, 2.02186013e-04,
        5.23260740e-11, 1.21992900e-06, 7.21373915e-14, 9.99775350e-01,
        7.67971144e-08, 1.09266639e-05],
       [2.31039738e-08, 7.11840185e-05, 9.99922037e-01, 6.47704428e-06,
        7.76041930e-15, 7.25283777e-10, 7.21126625e-09, 1.27973126e-13,
        1.91307279e-07, 2.46916723e-12],
       [1.71784973e-06, 9.98358786e-01, 3.25728615e-04, 3.66985405e-05,
        3.45551729e-04, 3.43645456e-06, 4.32516754e-05, 4.42476623e-04,
        4.21421661e-04, 2.09370755e-05],
       [9.99385834e-01, 2.56485464e-08, 3.70431371e-04, 8.79353706e-07,
        3.17038212e-06, 3.16674268e-05, 1.02956597e-04, 8.99282895e-05,
        7.86026746e-08, 1.49787465e-05],
       [2.92679069e-05, 1.79625748e-09, 1.50865104e-04, 2.45463582e-07,
        9.77253616e-01, 6.81265146e-06, 1.93367887e-05, 2.10188431e-04,
        4.60406409e-06, 2.23250519e-02]], dtype=float32)>
```

実験１

（仮説）0学習の繰り返し回数は、

いまは　**５.**

これを**増やすと**、**誤差**は下がるかも。

② 実行

```
model.fit(x_train, y_train, epochs=20)
```

① まず、20に書き換え

```
Epoch 1/20
1875/1875 [==============================] - 3s 2ms/step - loss
```

④ loss の右横の数値を確認

③ 実行

```
model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.0922 - accuracy: 0.9798
[0.09222090244293213, 0.9797999858856201]
```

実験２

（仮説）**ニューロン数**を 128 から 1000 に変えると、**誤差**は変化するかも。

② 実行

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(1000, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

① まず、1000に書き換え

③ その下のセルをすべて実行

```
predictions = model(x_train[:1]).numpy()
predictions
```

```
tf.nn.softmax(predictions).numpy()
```

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
loss_fn(y_train[:1], predictions).numpy()
```

```
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

④ loss の右横の数値を確認

```
model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.1055 - accuracy: 0.9825
[0.10547599196434021, 0.9825000166893005]
```

12-4 ニューラルネットワークを用いた画像分類

（人工知能の基本）

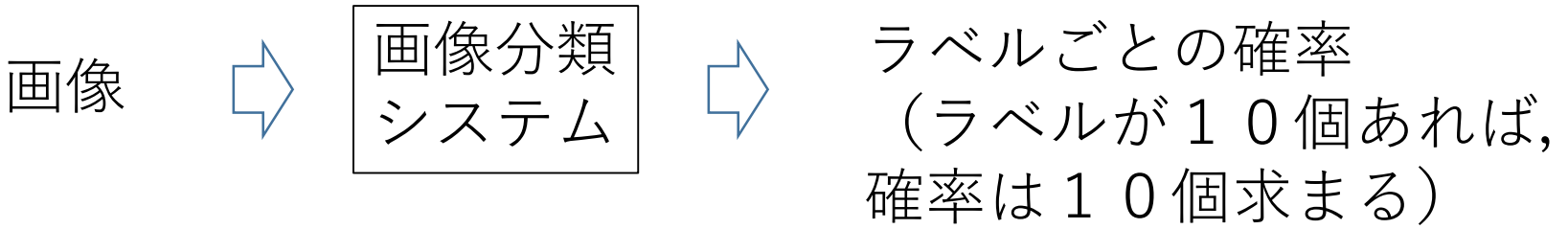
URL: https://www.kkaneko.jp/a/cs/index.html

金子邦彦

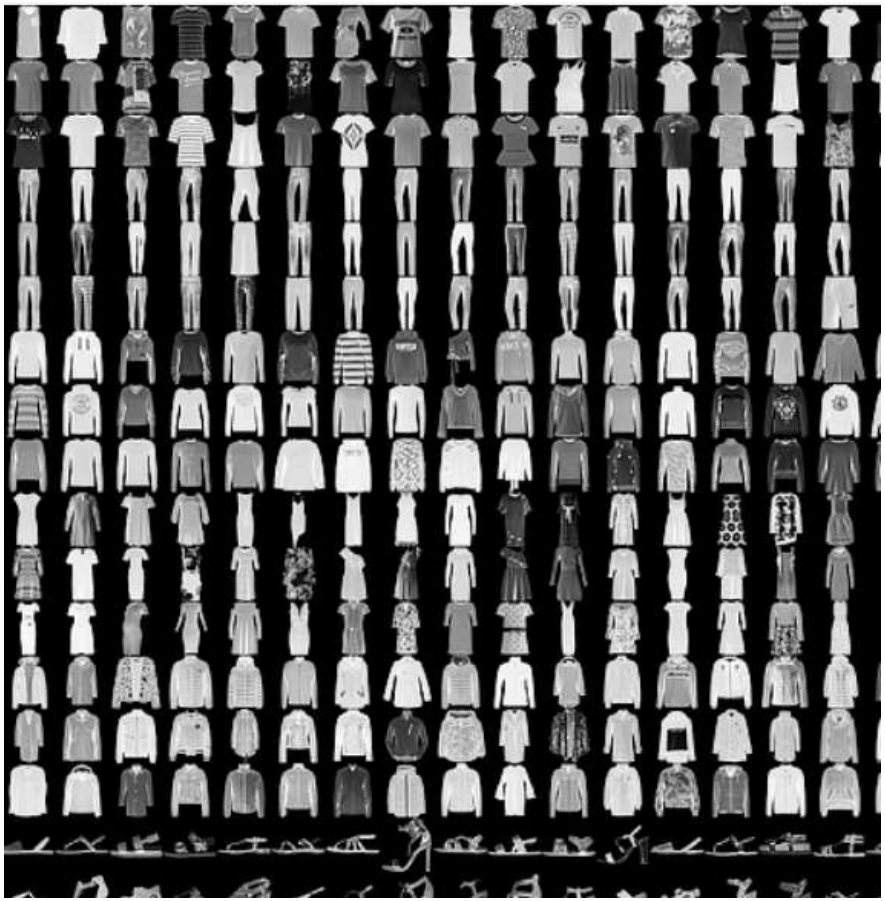
画像分類

- 画像分類は，**与えられた画像**に対して，次を得ること

  ラベルごとの確率



ここで行う画像の分類



たくさんの画像

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

- 画像を 10 種類に自動分類。

前準備

Google アカウントの取得が必要

- 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

パスワード

生年月日，性別

手順

① パソコンの Web ブラウザで，次のページを開く

https://www.tensorflow.org/tutorials

② 左側のメニューの「**Keras による ML の基本**」を**展開**，「**基本的な画像分類**」をクリック，「**Run in Google Colab**」をクリック

③ セルを**上から順に実行**する.

セルの実行の終了を確認してから、次のセルに移ること

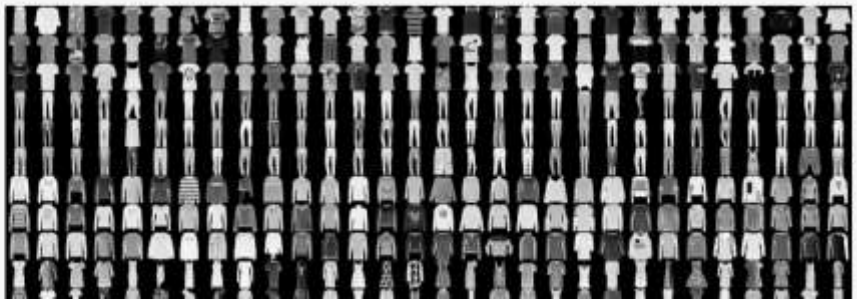
```
[ ] # TensorFlow と tf.keras のインポート
    import tensorflow as tf
    from tensorflow import keras

    # ヘルパーライブラリのインポート
    import numpy as np
    import matplotlib.pyplot as plt

    print(tf.__version__)
```

▾ ファッションMNISTデータセットのロード

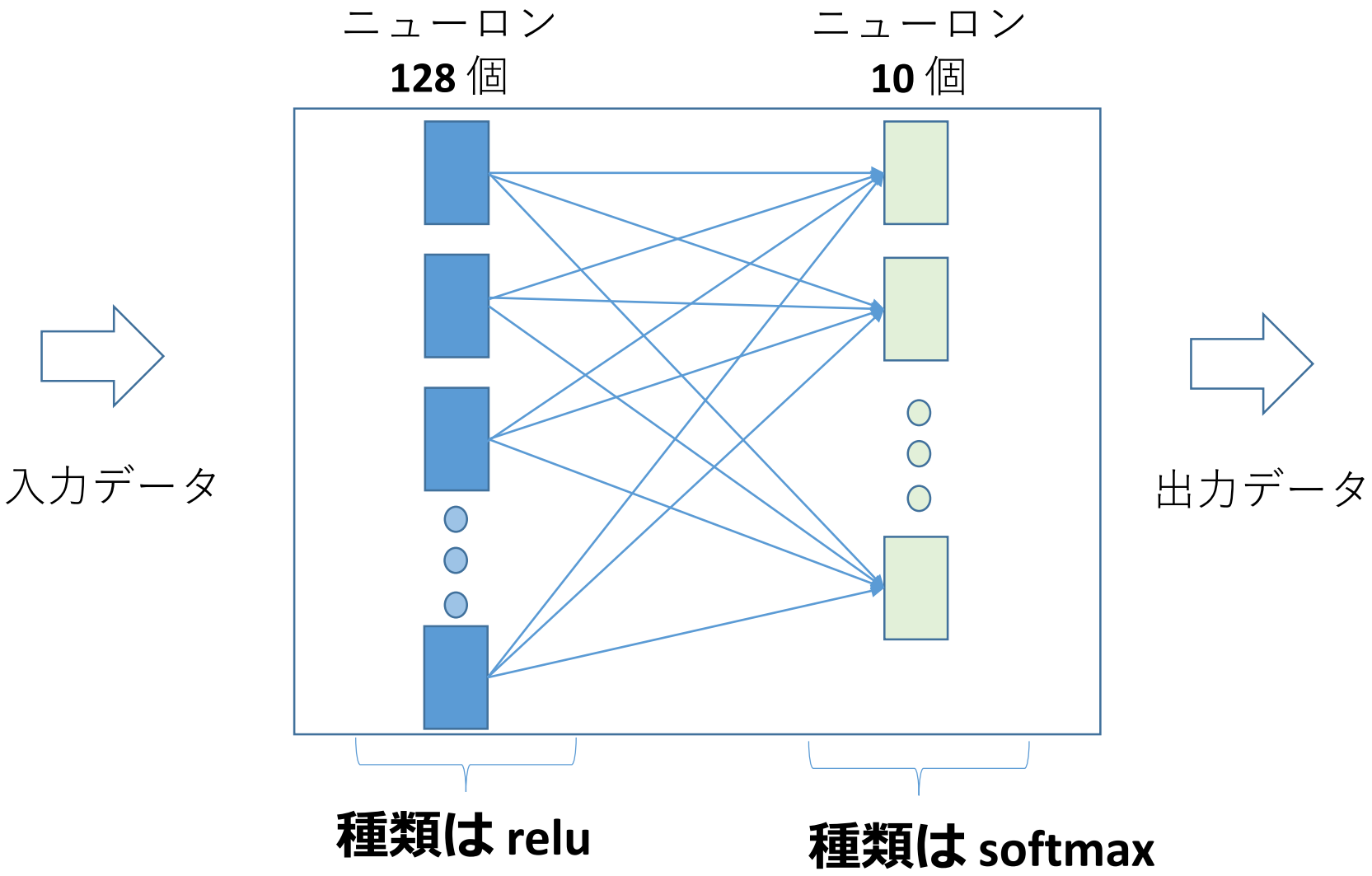
このガイドでは、Fashion MNISTを使用します。Fashion MNISTには10カテゴリーの白黒画像70,000枚が含ま
それは下図のような1枚に付き1種類の衣料品が写っている低解像度（28×28ピクセル）の画像です。



Google
アカウント
が必要

最後まで続ける

ニューラルネットワークの例



ニューラルネットワークを作成するプログラム

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

ニューラルネットワークの学習の様子

同じ**教師データ**を繰り返し使って学習を行う

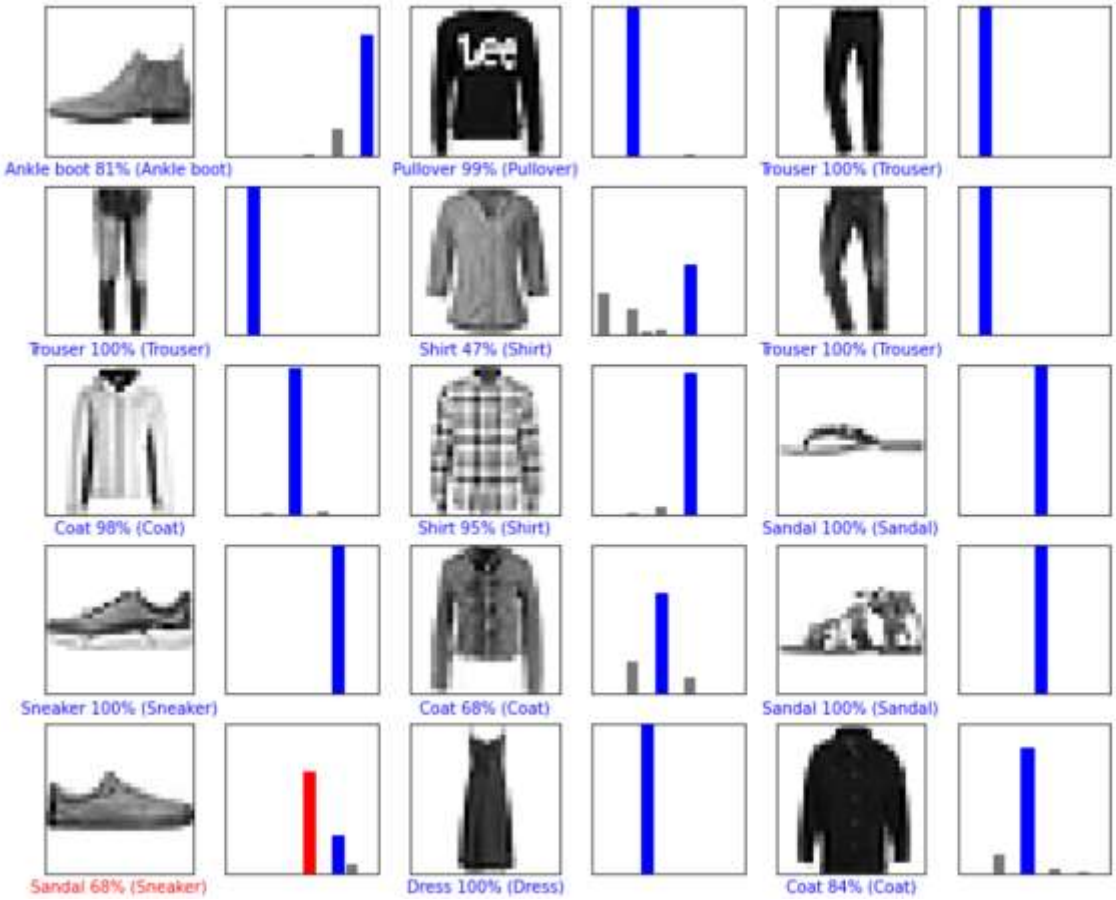
```
[14] model.fit(train_images, train_labels, epochs=5)

Epoch 1/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.5026 - accuracy: 0.8225
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3774 - accuracy: 0.8638
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3413 - accuracy: 0.8754
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3162 - accuracy: 0.8845
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2960 - accuracy: 0.8907
<tensorflow.python.keras.callbacks.History at 0x7f43d53c1e48>
```

繰り返し回数：**5回**

繰り返しのたびに**誤差が減少**（loss の右横の数値）

ニューラルネットワークによる予測の様子



１０種類のどれに分類されたかを棒グラフで表示
（ラベルが１０個あるので，**確率は１０個求まる**）
青：正解、　　赤や黒：不正解

12-5 学習不足と過学習

（人工知能の基本）

URL: https://www.kkaneko.jp/a/cs/index.html

金子邦彦

ニューラルネットワークの学習で気を付けること

- **学習**には**大量のデータ**が必要

  学習の成功のため

- **同じ教師データ**を使って**学習**を**繰り返す**

  学習不足の解消

- **学習**の**検証**が必要

  過学習が無いことの確認

ニューラルネットワークの学習

- **教師データ**（学習のためのデータ）を使用
- **学習**は**自動**で行われる

① **教師データ**により，**ニューラルネット**を動かし，誤差を得る

② **ニューロン間の結合の重み**の**上げ下げ**により、**誤差**を減らす （最終層の結果が，手前の層の結合の重みに伝搬することから，フィードバックともいわれる）

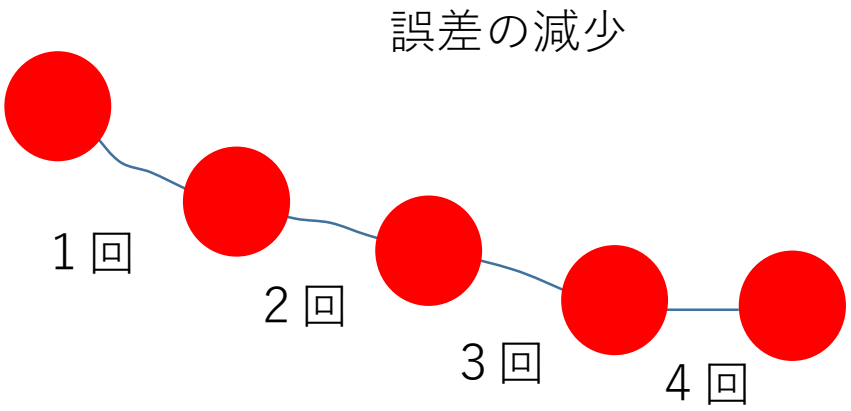
- ニューロンの数が増えたり減ったりなどではない
- **誤差**が**減らなくなったら**、**最適**になったとみなす

学習不足

- **ニューラルネットワーク**の**学習**では，学習のためのデータ（**教師データ**）を使う

- **教師データ**を１回使っただけでは，**学習不足**の場合がある

→　同じ**教師データ**を繰り返し使って学習を行う．
　　繰り返しながら，誤差の減少を確認



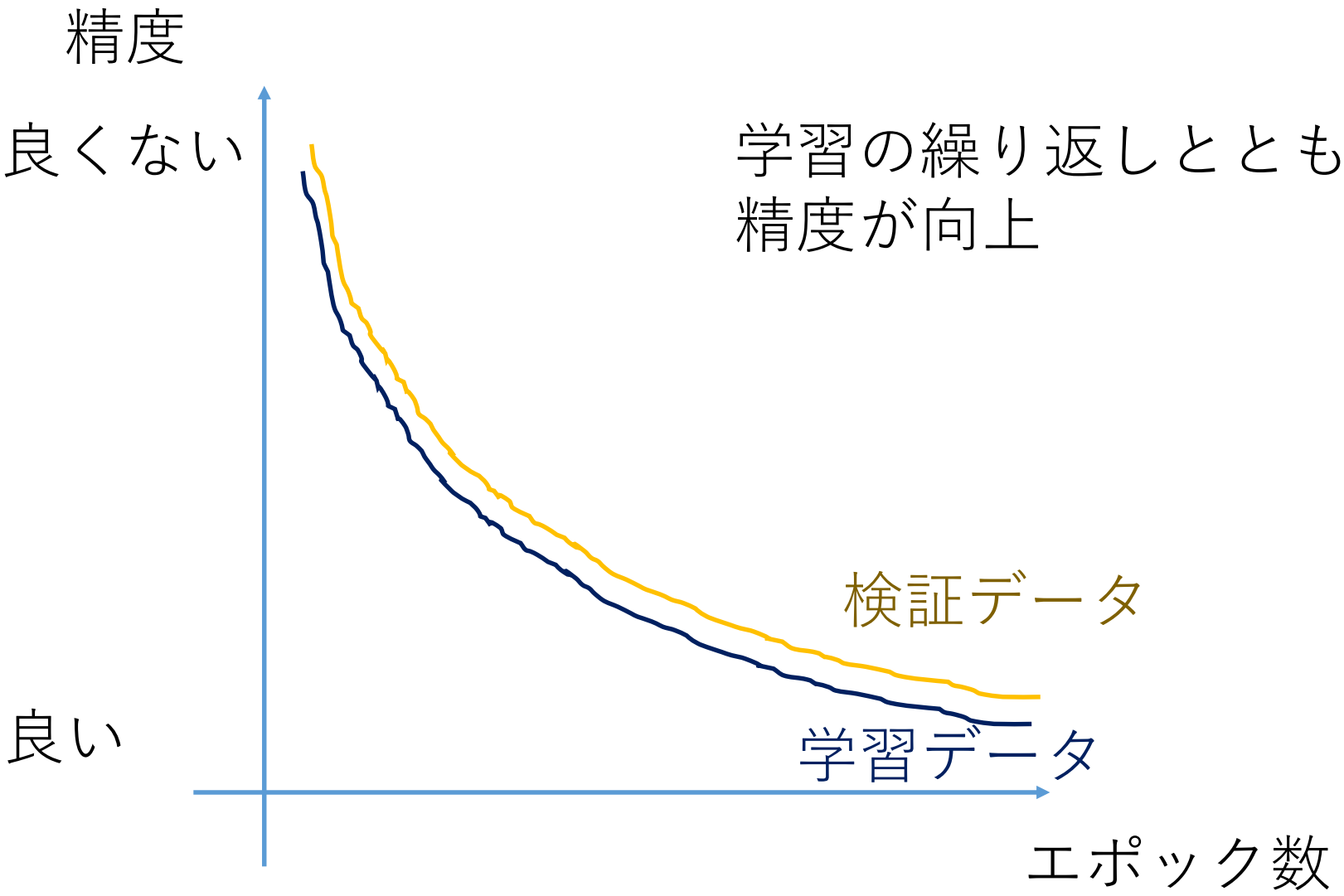
過学習

- **教師データ**での学習を終了したとき

- 検証データで**検証**すると，

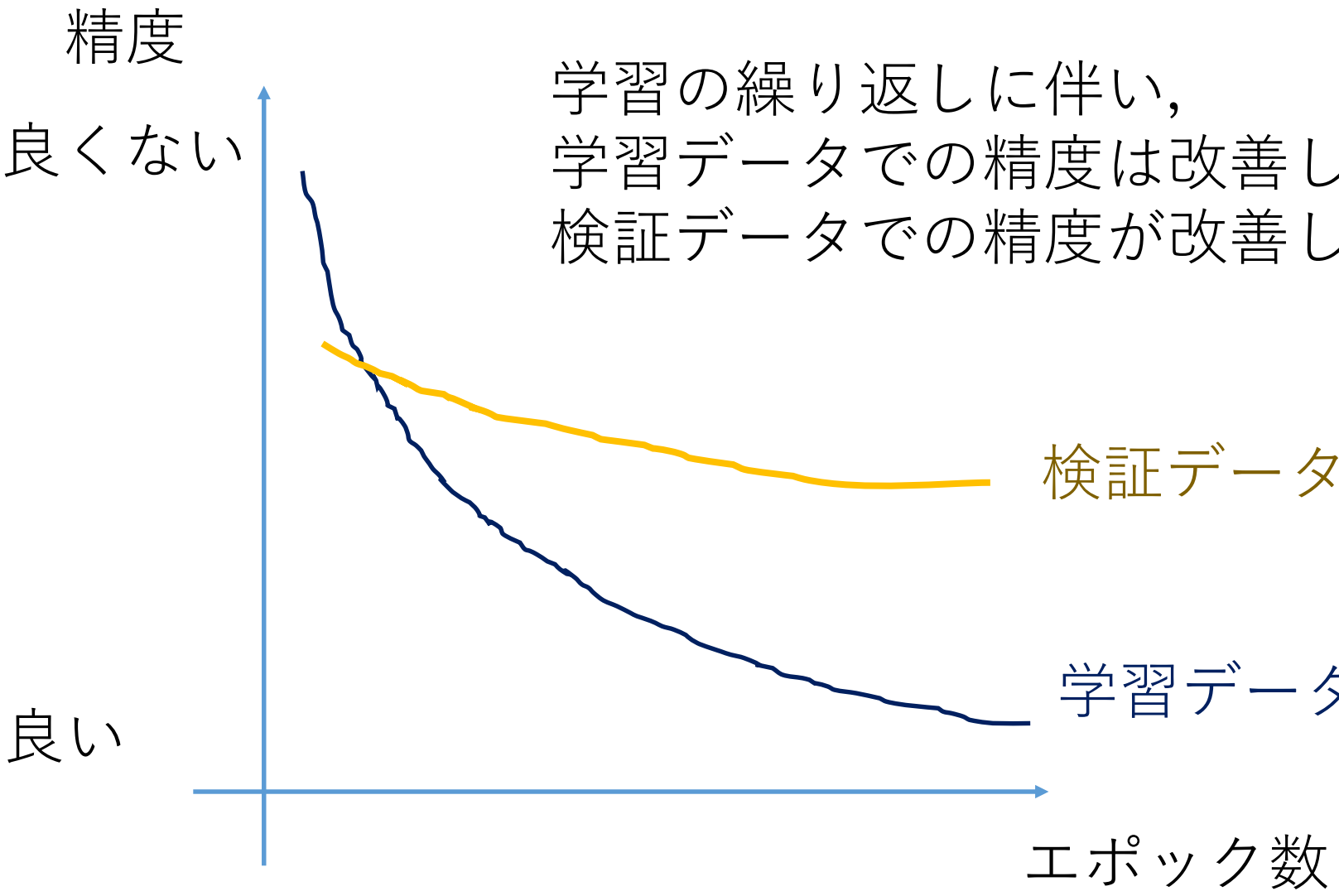
学習がうまくいっていないことが分かる場合がある

過学習なし



学習の繰り返しとともに
精度が向上

過学習あり



学習の繰り返しに伴い，
学習データでの精度は改善しても，
検証データでの精度が改善しない

ニューラルネットワークの学習では,

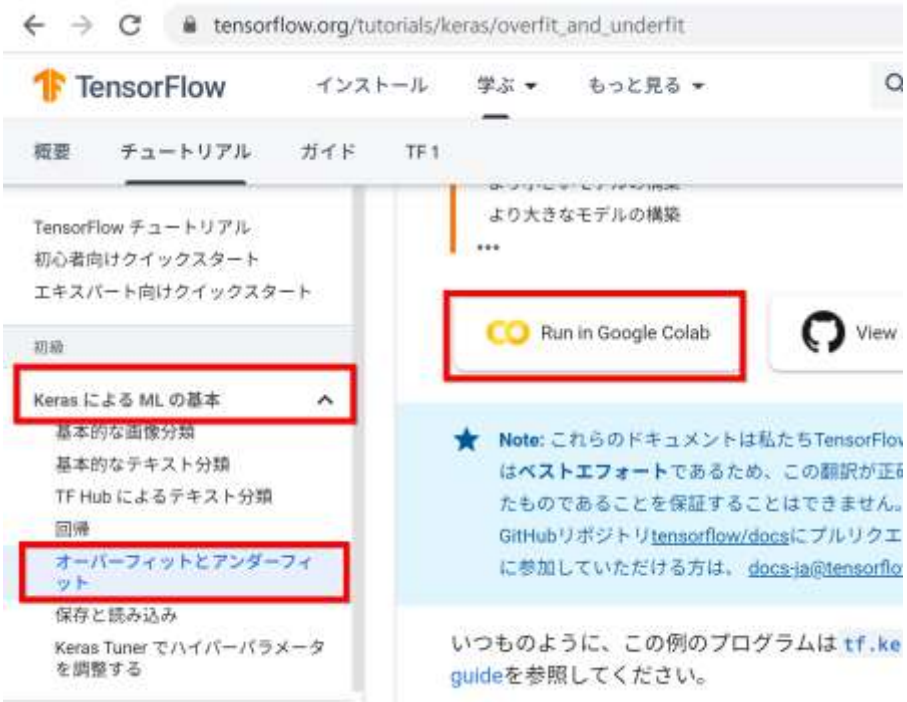
過学習が発生していないことを確認すること

手順

① パソコンの Web ブラウザで，次のページを開く

https://www.tensorflow.org/tutorials

② 左側のメニューの「**Keras による ML の基本**」を**展開**，「**オーバーフィットとアンダーフィット**」をクリック，「**Run in Google Colab**」をクリック



- ３つのニューラルネットワークの学習曲線

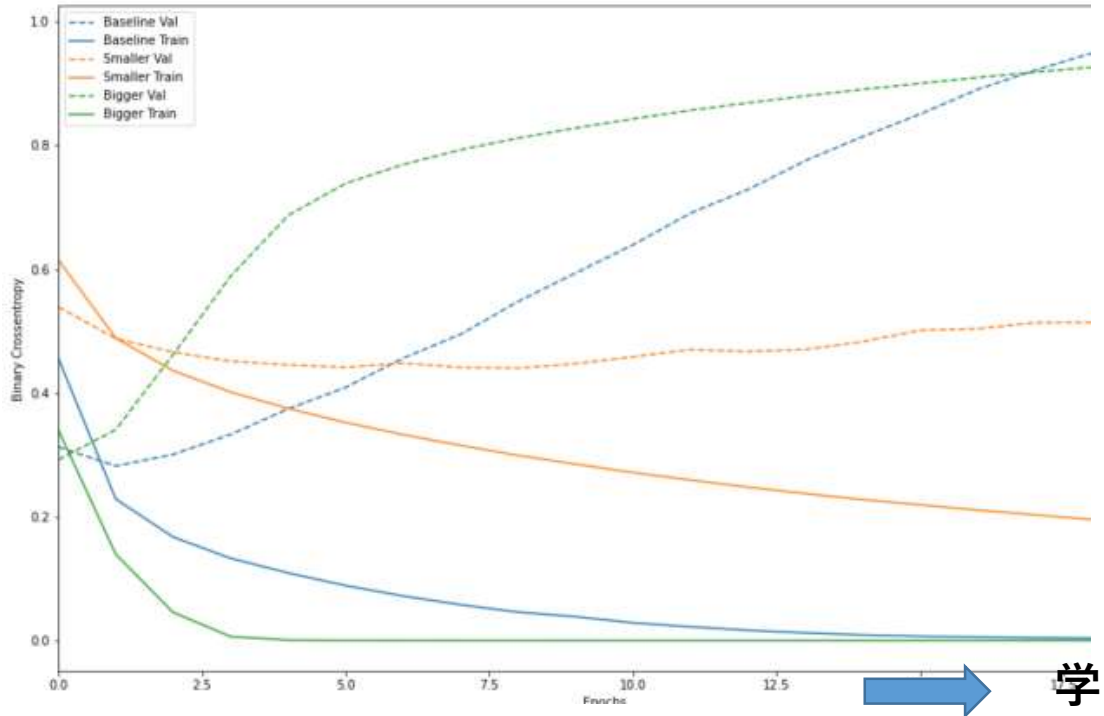
ニューロン数

Baseline:　1層目は**16**個，　2層目は**16**個

Smaller:　1層目は**4**個，　2層目は**4**個

Bigger:　1層目は**512**個，　2層目は**512**個

ニューロン数が多いと**過学習**が**起きやすい**



点線は**検証データ**

実線は**学習データ**

学習の繰り返し

考察の例

次のグラフから次を読み取る

- **学習の繰り返し回数**はいくつがよさそうか？
- **過学習**は発生しているか、していないか？

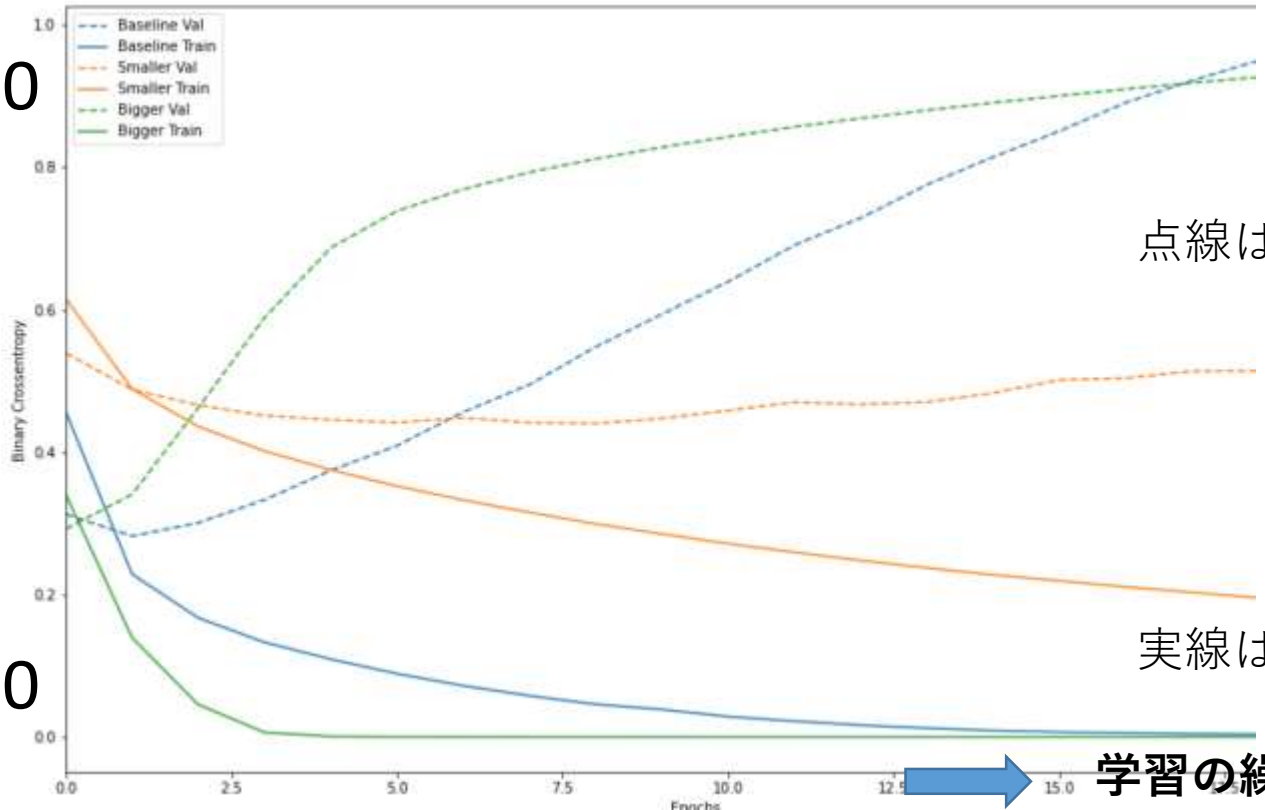
良くない

1.0

精度

0.0

良い



点線は**検証データ**

実線は**学習データ**

学習の繰り返し

過学習の防止に役立つ技術

- データの拡張

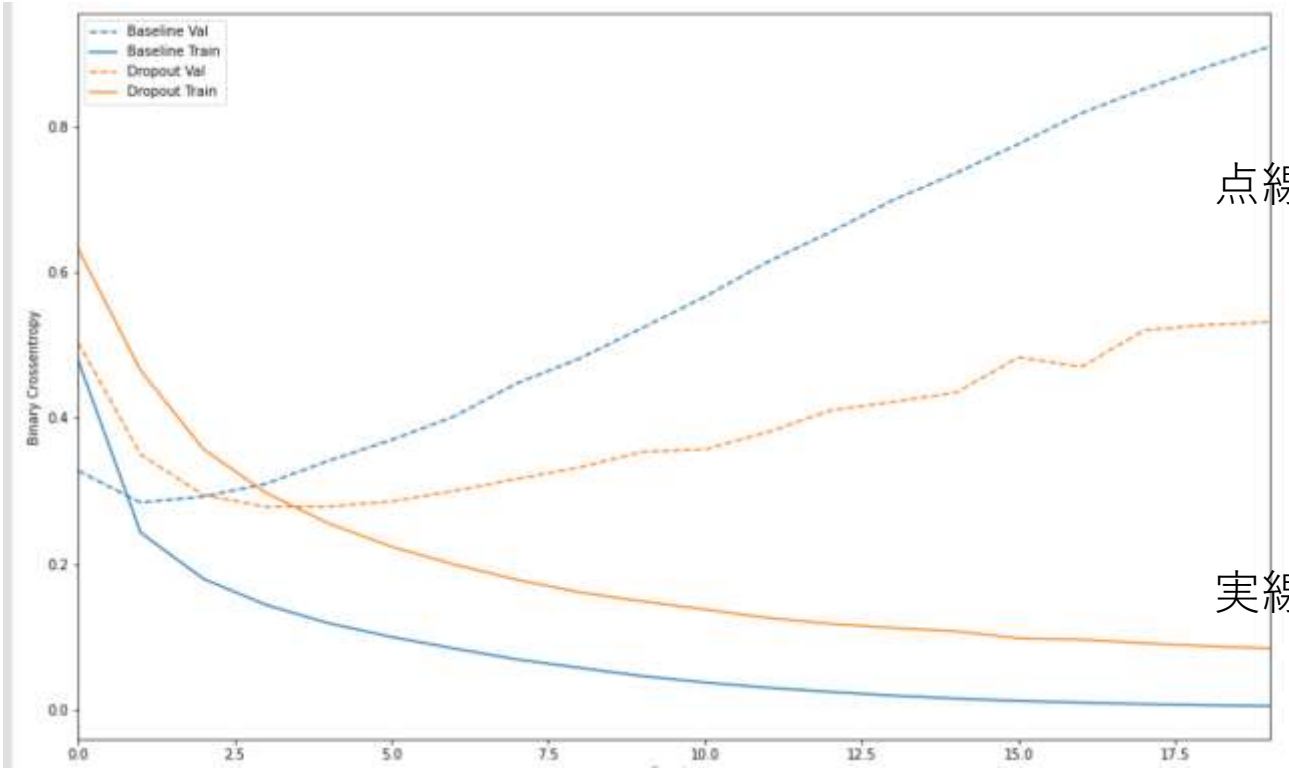
  教師データの拡張（増量）と再学習による解決

- ドロップアウト

  学習の途中で，**ニューロン間の結合をランダムに無効化する**ことで解決

- その他（正則化など）

- **ドロップアウト**等の技術により，過学習を緩和



点線は**検証データ**

実線は**学習データ**

学習の繰り返し

青：ドロップアウト等なし　オレンジ：ドロップアウト等あり