ニューラルネットワークの仕組み

金子邦彦

アウトライン

1. 機械学習とは
2. ニューラルネットワーク
3. Python の配列（アレイ）と画像データセット
4. ニューラルネットを作るプログラム
5. ニューラルネットワークの学習を行うプログラム
6. ニューラルネットワークを使うプログラム

1 機械学習とは

機械学習

- **与えられたデータ（教師データ）**を使い，**未知のデータに対しても**当てはまるパターンや規則を，コンピュータが抽出すること

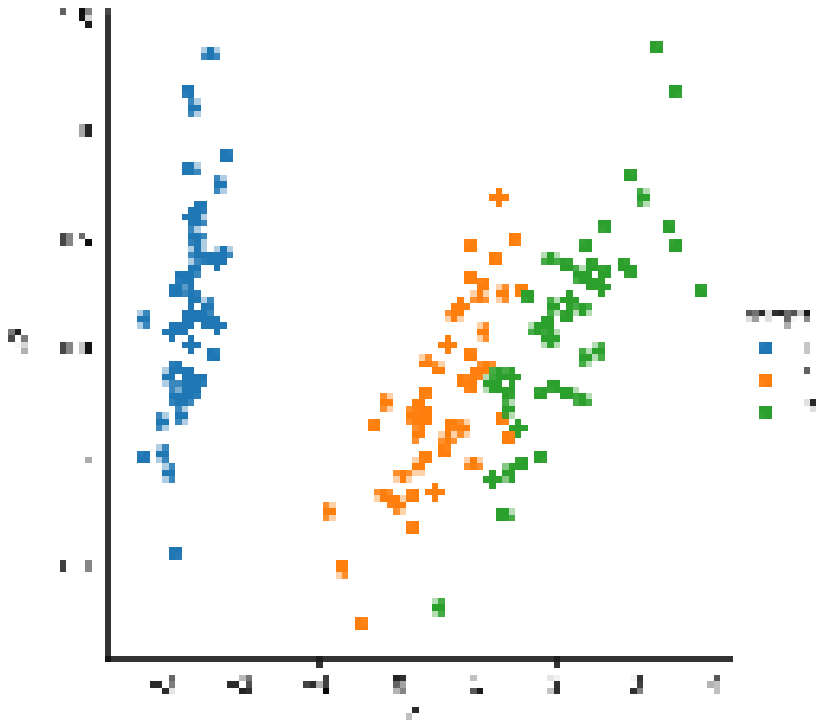
ニューラルネットワークなど，機械学習を可能にする，多数の技術がある

機械学習の用途

- **未知のデータ**の分類
- 予測

幅広い応用：画像認識，音声認識，自然言語処理，
データ分析

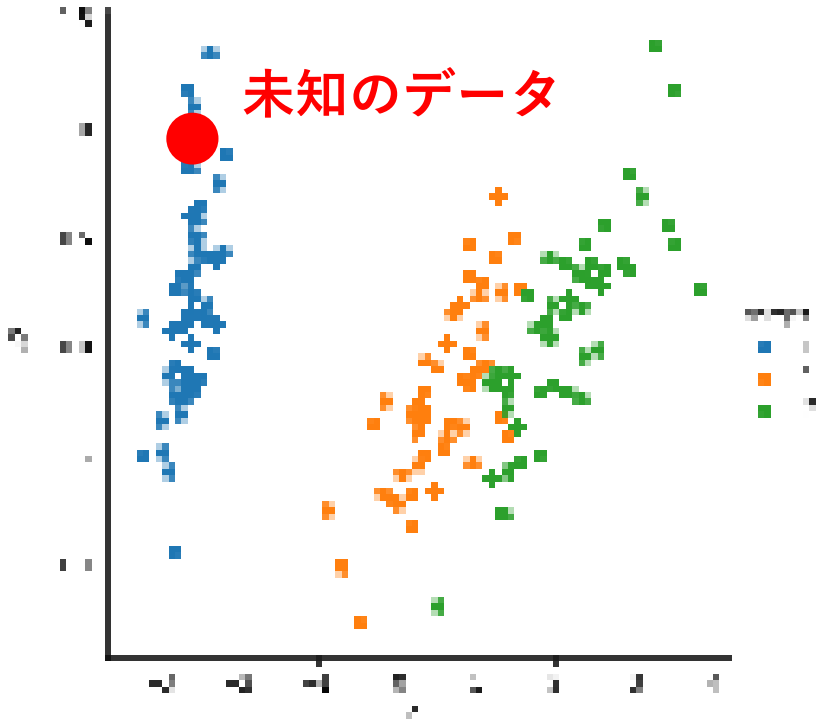
教師データの例



Iris データセット
・3種，150のアヤメの花びらのデータ
※ 右図は，主成分分析の結果のプロット

- **教師データ**は，**多数のデータの集まり**
- 上の図では，点１つで，１つのデータ

教師データによる予測



- 新しいデータ（**未知のデータ**）が
  あるとき，花の種類は何でありそうか

教師データの利用により，**未知のデータ**についても見通しを立てることが可能に

2. ニューラルネットワーク

ニューラルネットワークの進展を助けているもの

- **ニューラルネットワーク**の**技術革新**

  基盤技術: Heの初期化, Batch Normalization,

  Dropout, CNN, LTSM, GAN系列

  モデル: VGG16 など

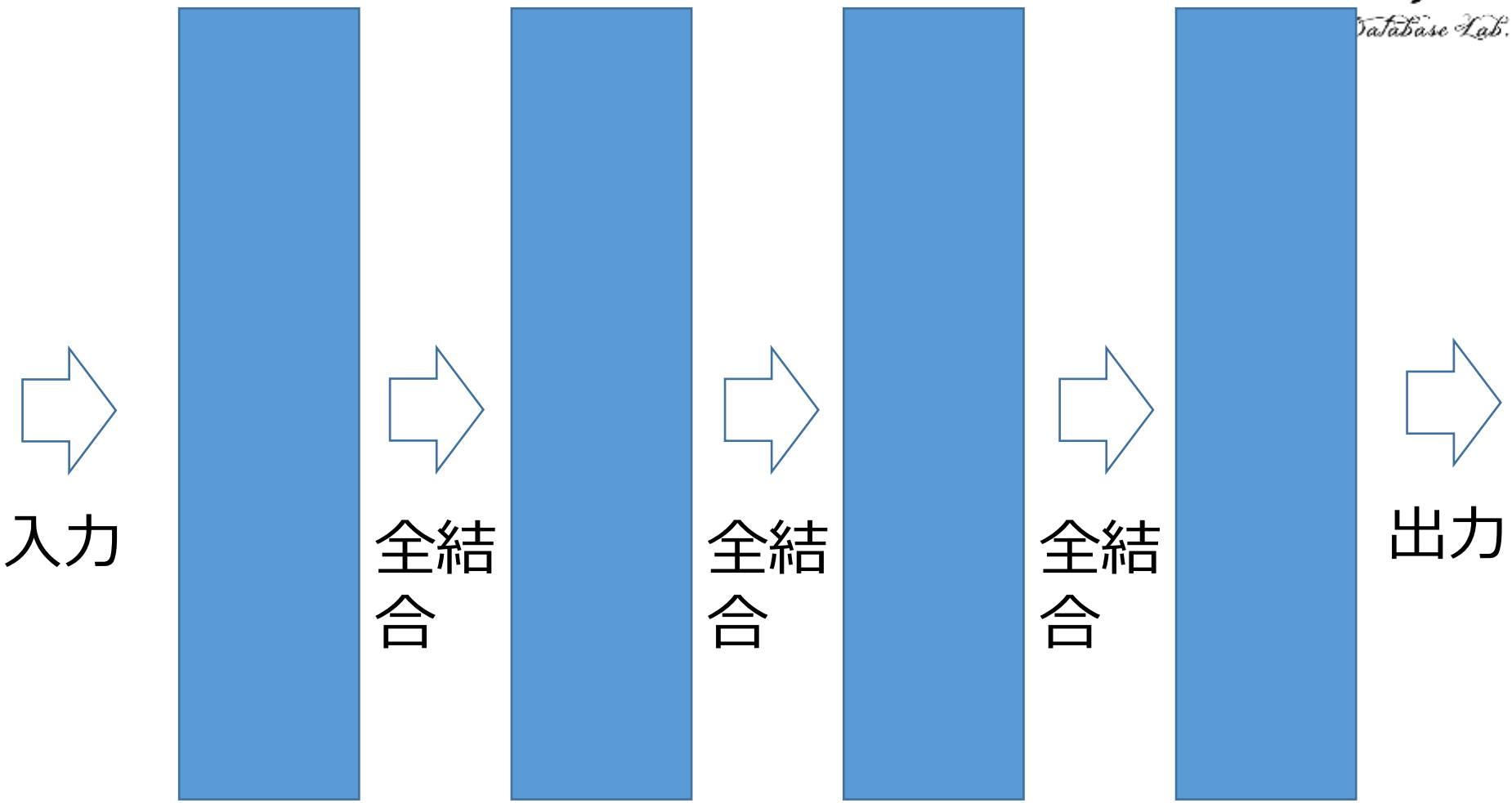
- **ニューラルネットワーク**を**高速にシミュレーション**できる高性能のコンピュータ

  高性能プロセッサ、GPU

- **ニューラルネットワーク**の学習に役立つ**大量のデータ**

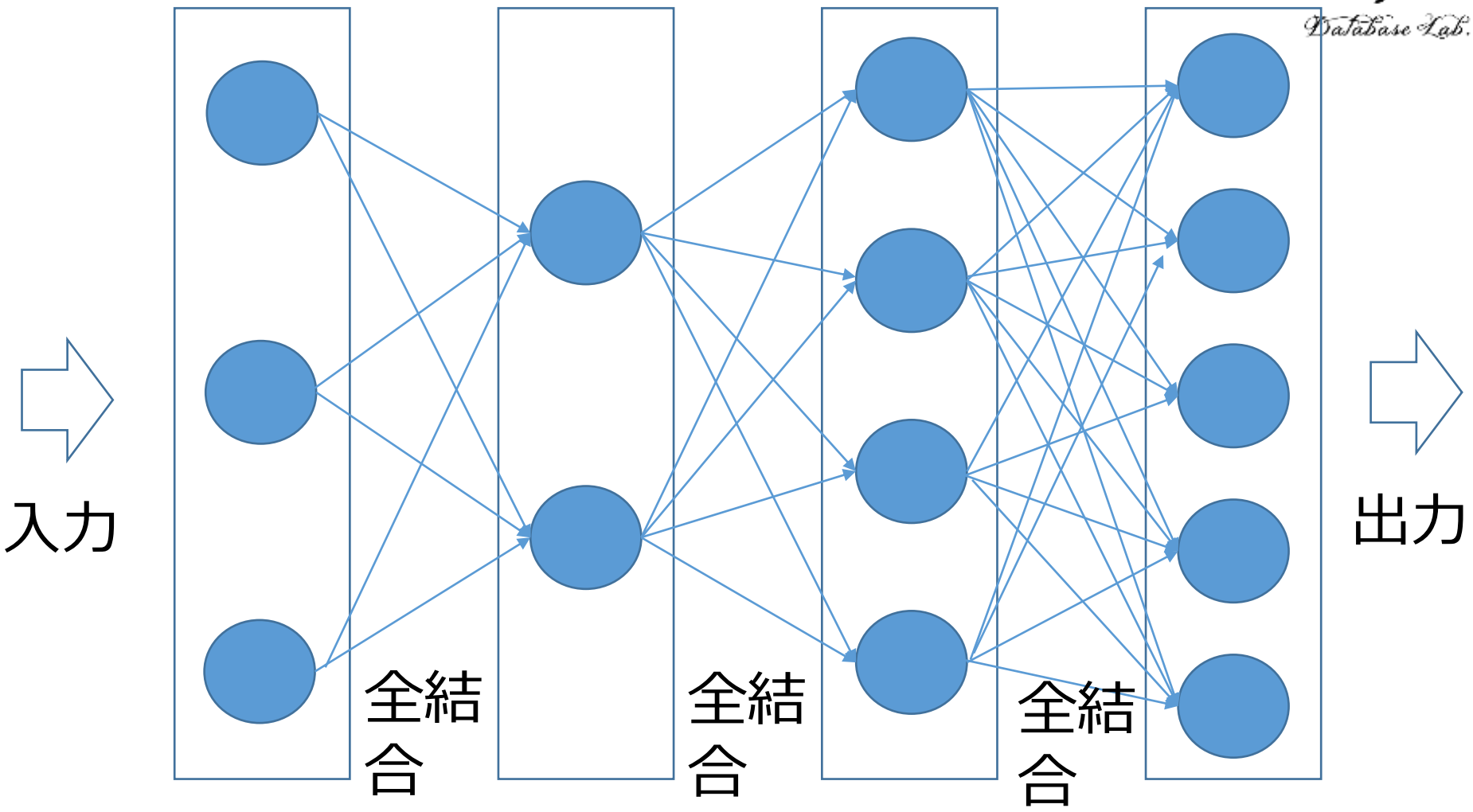
  データ計測、データ収集

層が直列になっているニューラルネットワーク



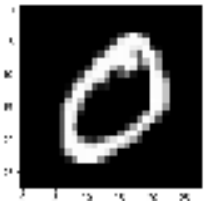
層数が４の場合（総数はいろいろ変わる）

ユニットと全結合



※　層の中には、ユニットが並ぶ

ニューラルネットワークによる予測



画素数
784 の
モノクロ画像
があったとする

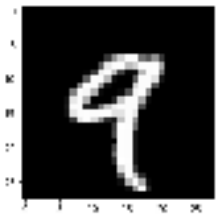
データは配列
（アレイ）

1層目

2層目

活性化

ニューラルネットワークによる予測



画素数
784 の
モノクロ画像
があったとする

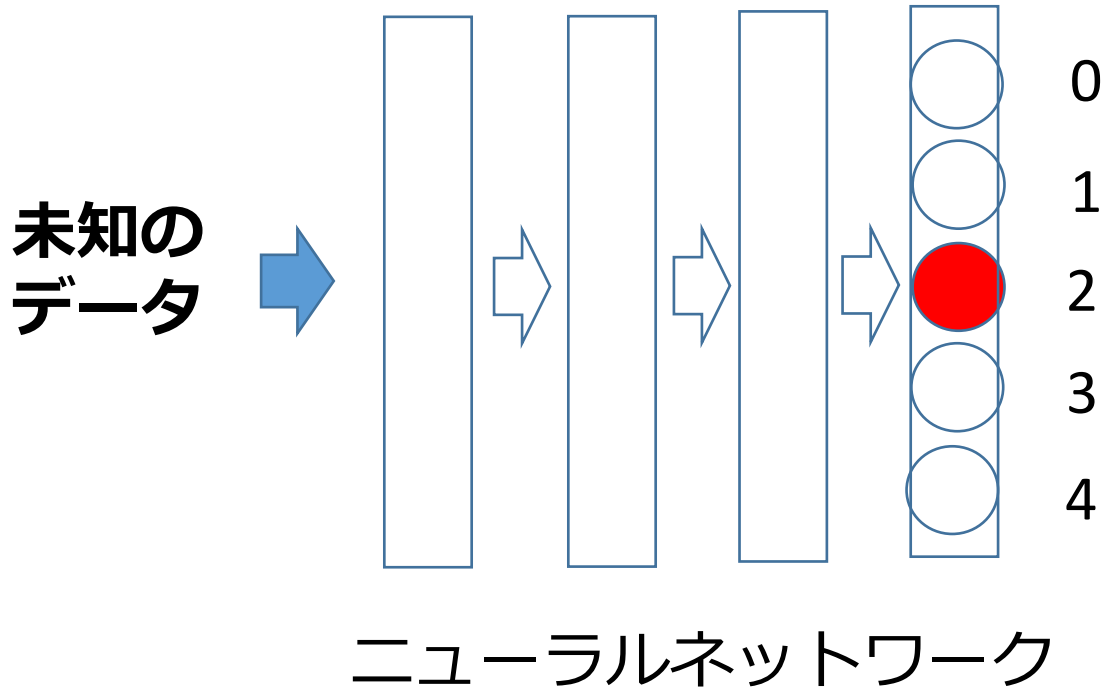
データは配列
（アレイ）

1層目

2層目

活性化

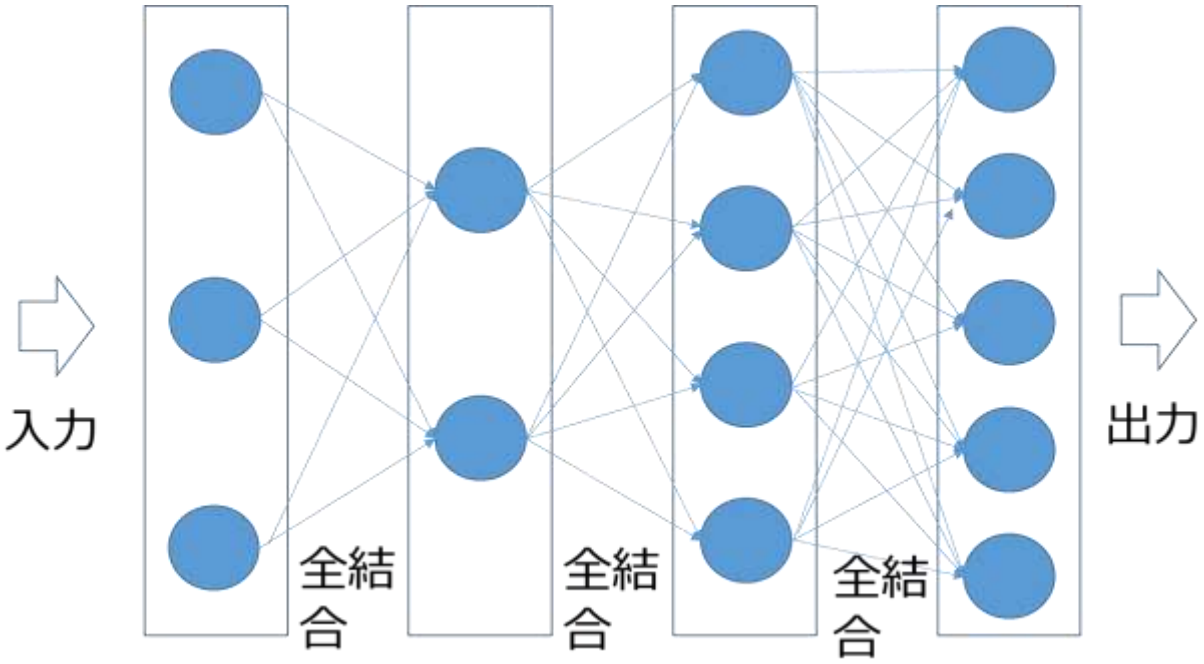
ニューラルネットワークによる予測



１つが活性化
↓
活性化したユニットが「２番」なら，
「**予測結果は２**」
の意味

・**ニューラルネットワーク**での予測は，
未知のデータを与えて，**最終層のユニットを活性化**
させることで行う

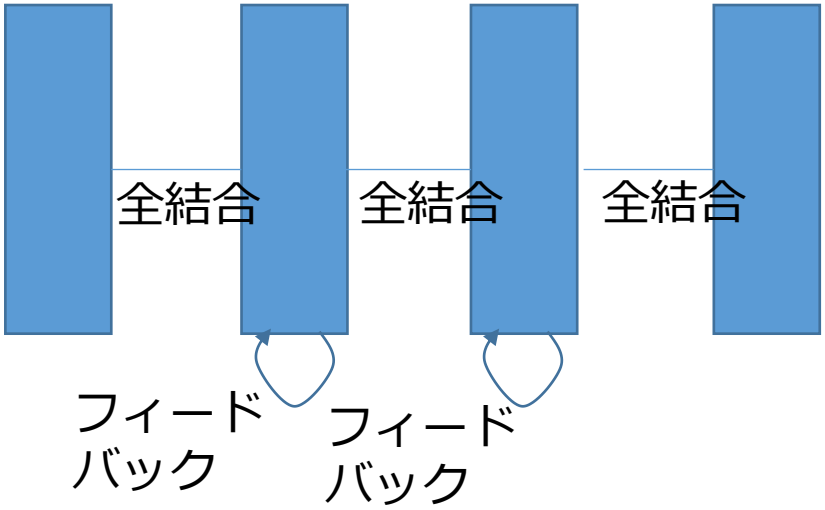
ニューラルネットワークとは



・**ニューラルネットワーク**は，**層**が積み重なっている
・**層**の中には，**ユニット**が並ぶ．
・**ユニット**は，**互いにつながり**，ときには**活性化**する
・**ニューラルネットワークで**の予測は，
未知のデータを与えて，最終層の**ユニット**を**活性化**させることで行う

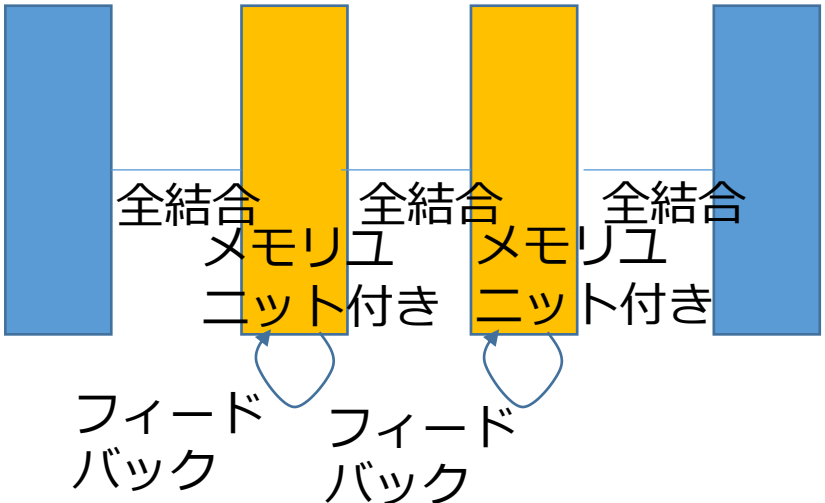
ニューラルネットワークのバリエーション

- Recurrent Network



フィードバック付き

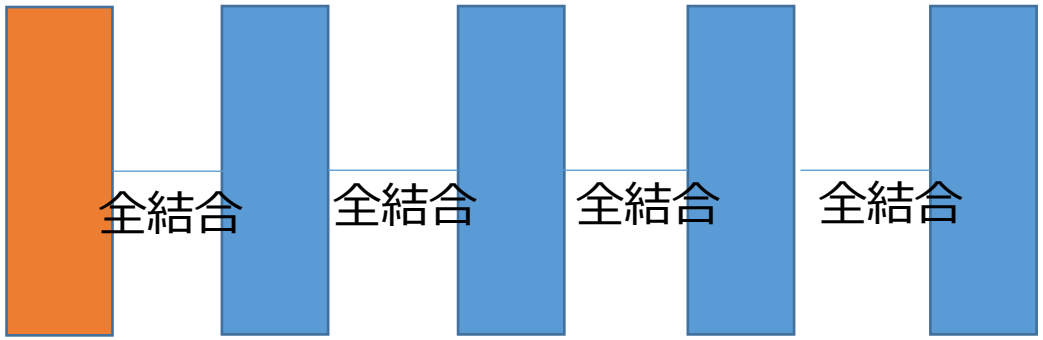
- LSTM Recurrent Neural Network



メモリユニット付き

ニューラルネットワークのバリエーション

- CNN



畳みこみ層

畳みこみ層付き

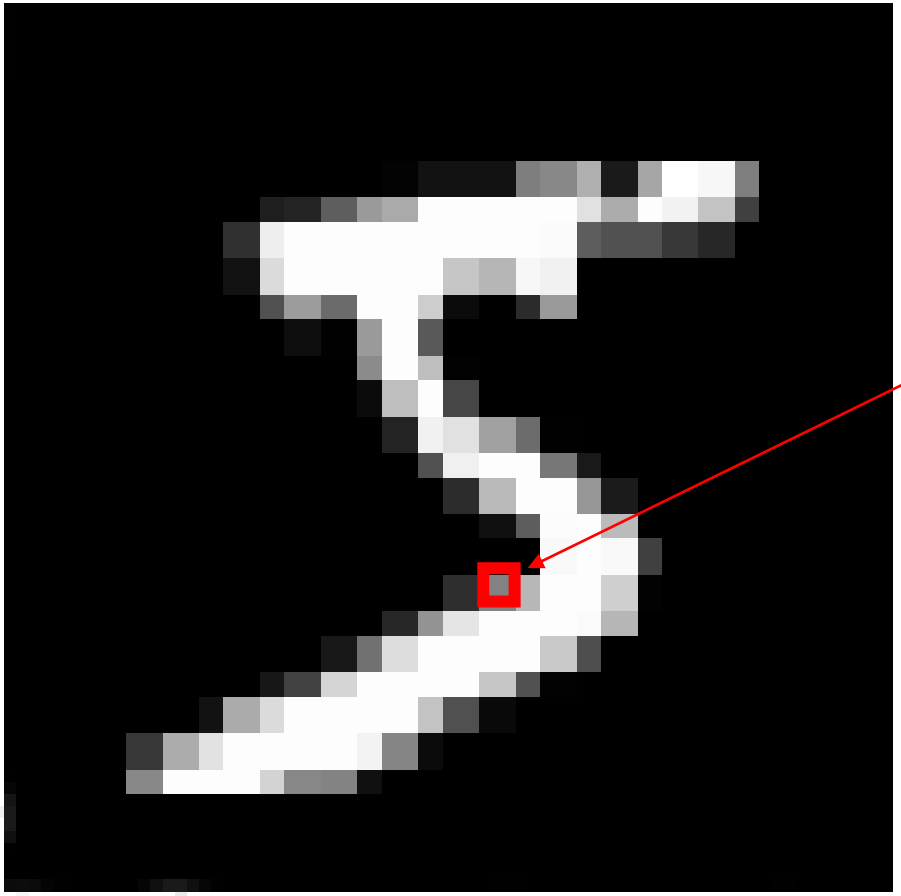
3. Python の配列（アレイ）と画像データセット

配列（アレイ）とは

0	1	2	3
180	**20**	**250**	**40**

配列（アレイ）とは，**データの並び**で，
０から始まる番号（添字）が付いている

画像と画素



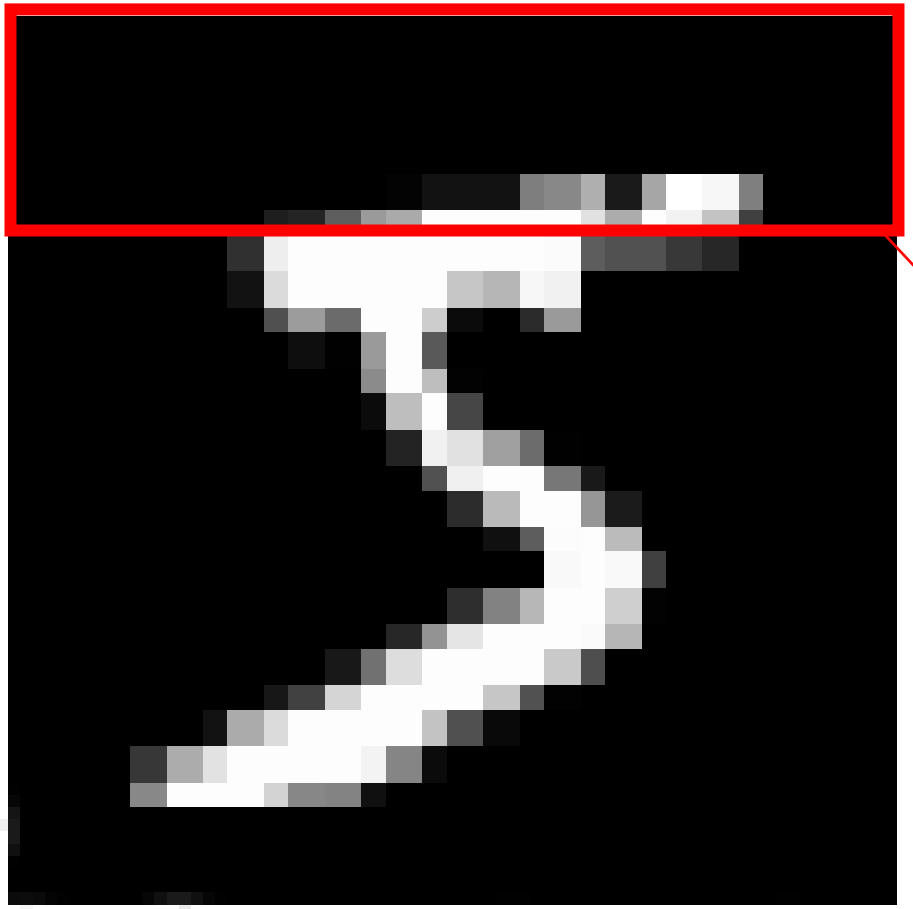
画素

	画素値
白	255
黒	0

画素値は，**画素**の明るさに応じた **0 から 255 の数値**

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



画像と配列（アレイ）

MNISTデータセット（手書き文字のデータセットで，濃淡画像）

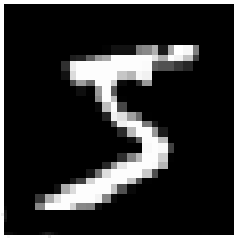
画像サイズ: 28 × 28

画像全体は，サイズ 28 ×28 の**配列**
（データの並び）

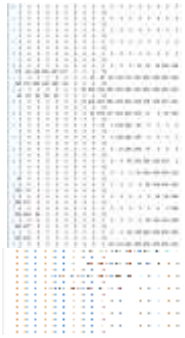
```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0]
 [   0    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   30   36   94  154  170  253  253  253  253  253
   225  172  253  242  195   64    0    0    0    0]
```

画像の**上 7行分**の画素値を表示したところ（28 ×7分）

配列（アレイ）の形と次元



1枚の画像



28 × 28

次元数は 2

60000枚の
画像データ
セット

60000 ×
28 × 28

次元数は 3

データ　　配列（アレイ）の形　　次元数

画像データセット MNIST

- 濃淡画像 **70000**枚．画像サイズは 28 × 28

  うち学習用 **60000**枚

  配列（アレイ）の形：60000 × 28 × 28

  テスト用 **10000**枚

  配列（アレイ）の形：10000 × 28 × 28

配列（アレイ）の形と次元

サイズ 28 × 28 の濃淡画像は

配列（アレイ）の形：**28 × 28**

次元: 2

サイズ 28 × 28 の濃淡画像 60000 枚の画像データセットは

配列（アレイ）の形：**60000 × 28 × 28**

次元: 3

4. ニューラルネットワークを作るプログラム

Python

- 数多くある**プログラミング言語**の１つ

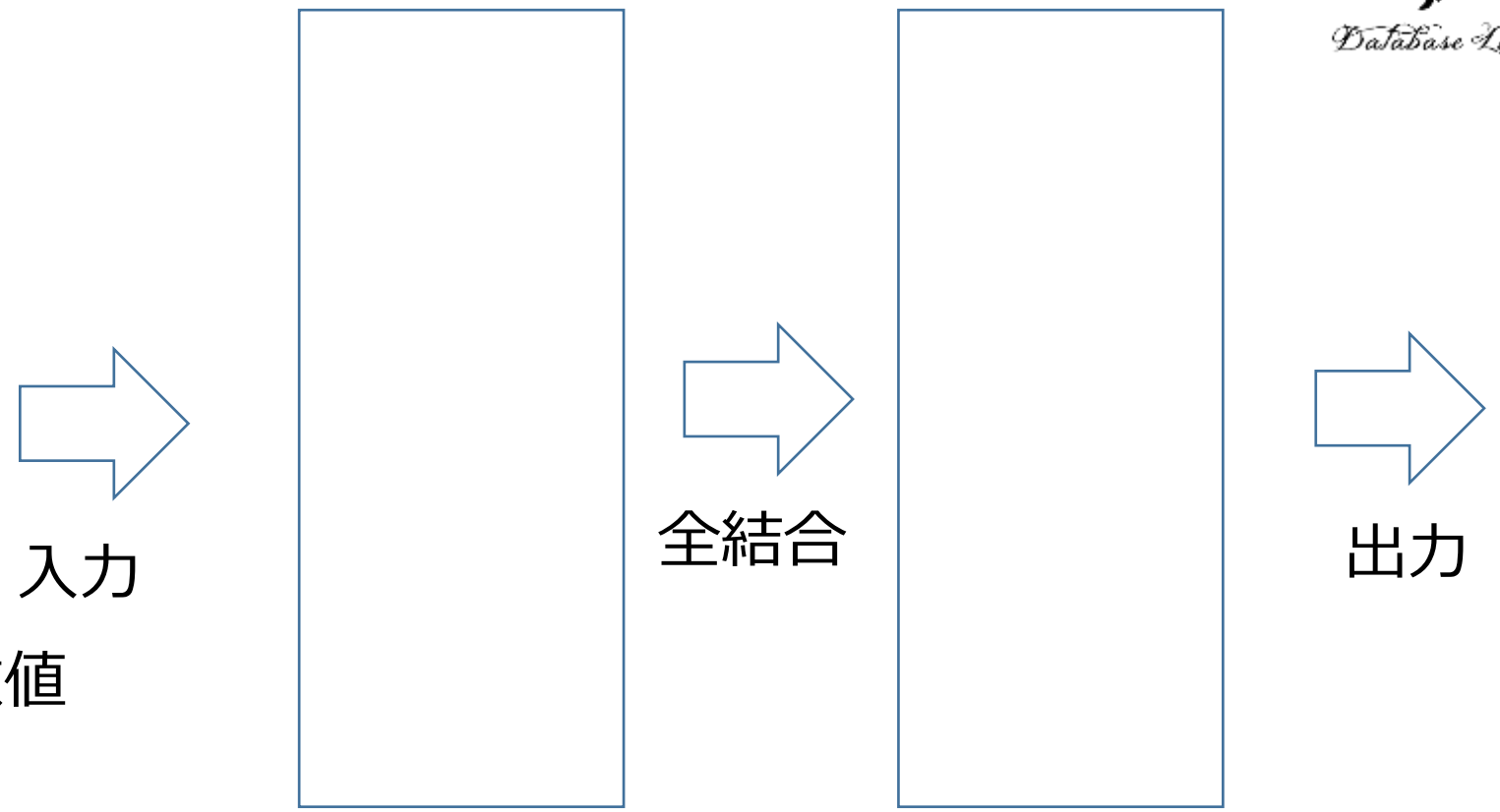
- さまざまな拡張機能（数百以上）がある

numpy	配列計算
tensorflow	ディープラーニング（深層学習）向けの配列計算
Pybrain	機械学習全般
Scipy	最適化，積分，信号，画像，統計，フーリエ変換
AIMA	

※ディープラーニング（深層学習）は，多層のニューラルネットワーク，その活用技術のこと

- アイデアをすぐに試したい（数十行以内のプログラム）場合にも向くとされる

ニューラルネットワークの例



4 個の数値

ユニット数 **64**

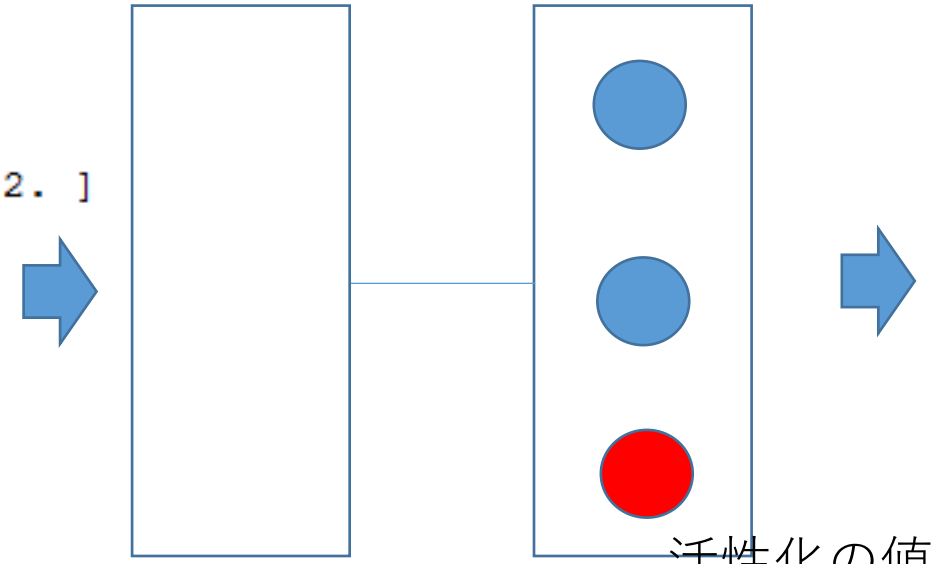
ユニット数 **3**
この層は，「**softmax**」になるように設定

層数： 2

ニューラルネットワークの動作イメージ

[7.7, 2.8, 6.7, 2.]

4 個の数値



0 である確率
0.0068
1 である確率
0.1592
2 である確率
0.8340

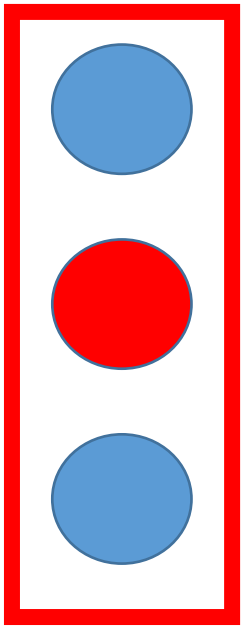
活性化の値が最大

ソフトマックス (softmax) に設定された層

ソフトマックス (softmax) に設定したとき

- **各ユニットの活性化**の度合いは，**0から１の数値**

- **最も値が高いもの**は「**活性化している**」，
  それ以外は「**活性化してない**」と考える



ニューラルネットワークを作るプログラム


```
import tensorflow as tf
from tensorflow.keras import layers

num_classes = 3
input_dim = 4

m = tf.keras.Sequential(
    [
        layers.Dense(units=64, input_dim=input_dim, activation='relu'),
        layers.Dense(units=num_classes, activation='softmax'),
    ]
)
```

3種類に分類

入力は **4**個の数値

1 層目のユニット数は **64**

「**softmax**」に設定

ニューラルネットワークの作成

プログラムで，次を指定

- 層の数
- 各層のユニットの数
- 各層のユニットの種類

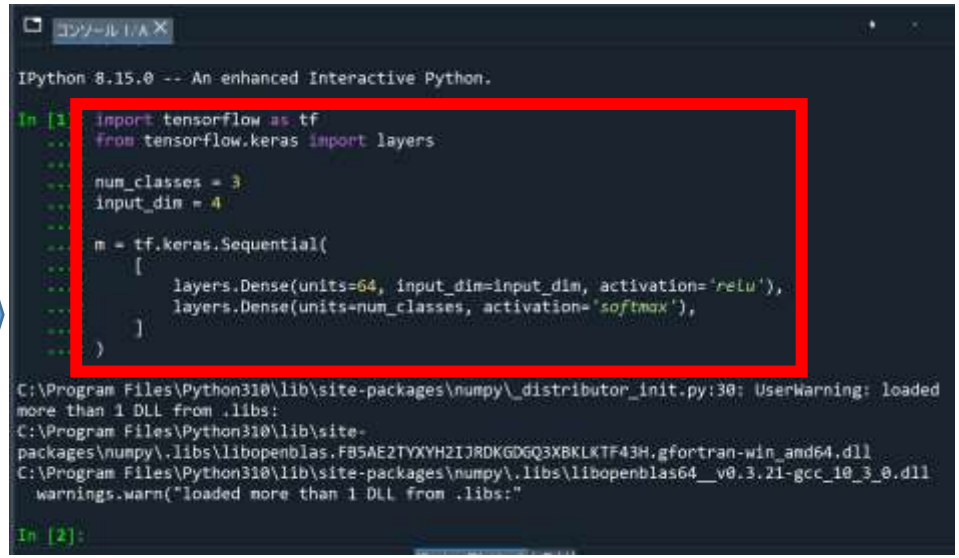
ノートページ

コンピュータ上の作成されるバーチャルなもの

■ 資料の Web ページを開く

https://www.kkaneko.jp/a/ai.html

■プログラムを，コピー，貼り付け



コピーしたい範囲をマウスで選び,
マウスの**右クリックメニュー**で「**コピー**」が便利

spyder の右下の　Python コンソールに張り付ける.
張り付けたあと **Enter キー**

マウスの**右クリックメニュー**で「Paste」が便利

■ ニューラルネットワークの確認表示

print(m.summary())


```
In [2]: print(m.summary())
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                320

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 515
Trainable params: 515
Non-trainable params: 0
_________________________________________________________________
None

In [3]:
```

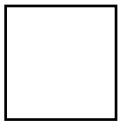
ニューラルネットワークのオブジェクト名は **m**

確認表示は print(**m**.summary())

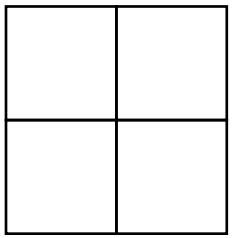
5. ニューラルネットワークの学習を行うプログラム

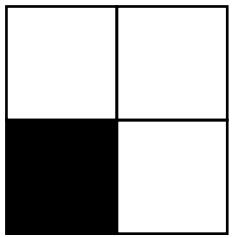
画像と画素

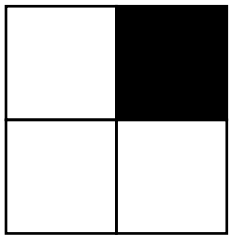
画素は，**白と黒の2種類**しかないとする

 **白は 0**　  **黒は 1,** とする

画像のサイズが 2 × 2のとき

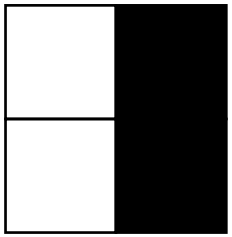
 **[0, 0, 0, 0]**

 **[0, 0, 1, 0]**

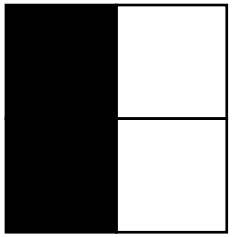
 **[0, 1, 0, 0]**

 **[1, 1, 1, 1]**

学習の例

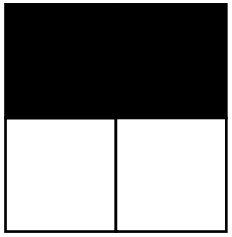


[0, 1, 0, 1]

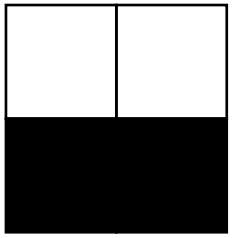


[1, 0, 1, 0]

1, を出力させる



[1, 1, 0, 0]



[0, 0, 1, 1]

2 を出力させる

その他

0, を出力させる

学習用データセットの例

[[0, 0, 0, 0],	[0,
[0, 0, 0, 1],	0,
[0, 0, 1, 0],	0,
[0, 0, 1, 1],	**2,**
[0, 1, 0, 0],	0,
[0, 1, 0, 1],	**1,**
[0, 1, 1, 0],	0,
[0, 1, 1, 1],	0,
[1, 0, 0, 0],	0,
[1, 0, 0, 1],	0,
[1, 0, 1, 0],	**1,**
[1, 0, 1, 1],	0,
[1, 1, 0, 0],	**2,**
[1, 1, 0, 1],	0,
[1, 1, 1, 0],	0,
[1, 1, 1, 1]]	0]

どういう**入力**のとき，どういう**出力**が出て欲しいかのデータ

ニューラルネットワークの学習

学習に使用されるデータセット

- どういう入力のとき，どういう出力が出て欲しいかのデータ
- 形は配列（アレイ）

パソコン実習

- さきほど作成したニューラルネットワークは**未学習**

ニューラルネットワークのオブジェクト名は **m**

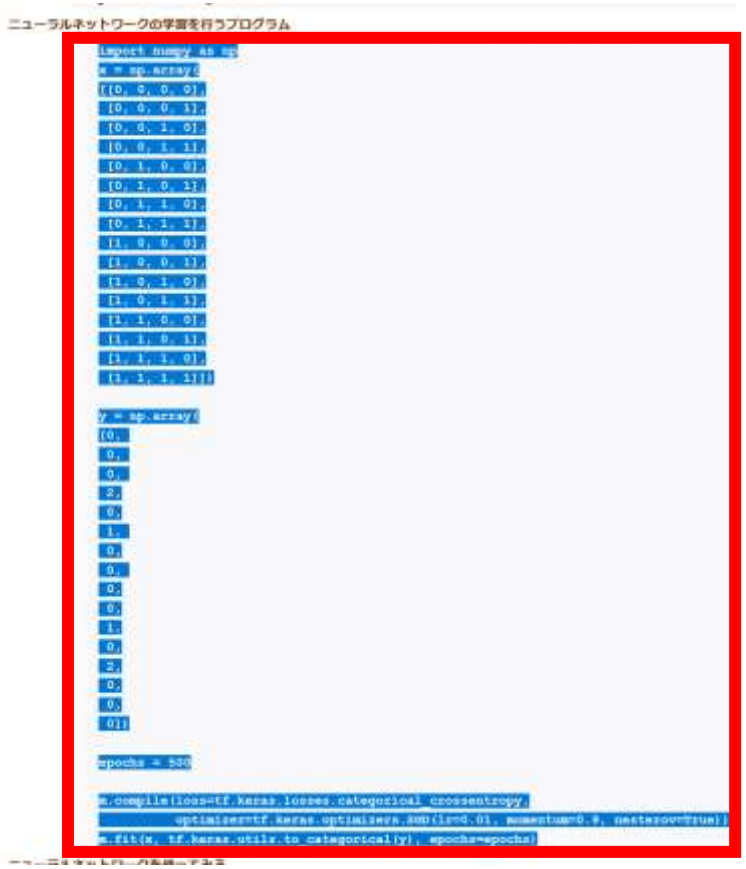
学習させるコマンド

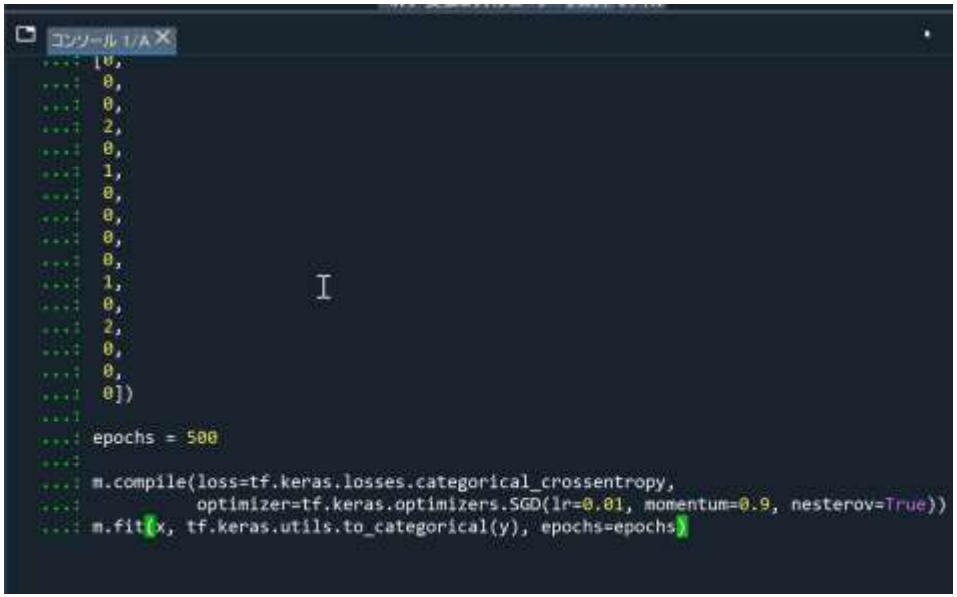
```
epochs = 500

m.compile(loss=tf.keras.losses.categorical_crossentropy,
        optimizer=tf.keras.optimizers.SGD(lr=0.01, momentum=0.9, nesterov=True))
m.fit(x, tf.keras.utils.to_categorical(y), epochs=epochs)
```

ニューラルネットワークの学習

■ **プログラムを，コピー，貼り付け**





コピーしたい範囲をマウスで選び,
マウスの**右クリックメニュー**で「**コピー**」が便利

spyder の右下の　Python コンソールに張り付ける.
張り付けたあと **Enter キー**

マウスの**右クリックメニュー**で「Paste」が便利

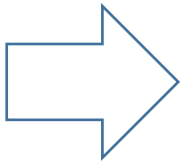
■ 学習の終了の確認


```
Epoch 493/500
1/1 [==============================] - 0s 2ms/step - loss: 0.0974
Epoch 494/500
1/1 [==============================] - 0s 9ms/step - loss: 0.0970
Epoch 495/500
1/1 [==============================] - 0s 1ms/step - loss: 0.0965
Epoch 496/500
1/1 [==============================] - 0s 2ms/step - loss: 0.0960
Epoch 497/500
1/1 [==============================] - 0s 2ms/step - loss: 0.0955
Epoch 498/500
1/1 [==============================] - 0s 2ms/step - loss: 0.0951
Epoch 499/500
1/1 [==============================] - 0s 3ms/step - loss: 0.0946
Epoch 500/500
1/1 [==============================] - 0s 2ms/step - loss: 0.0942
Out[3]: <keras.callbacks.History at 0x212441db700>

In [4]:
```

6. ニューラルネットワークを使うプログラム

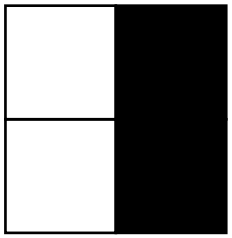
ニューラルネットの使用例

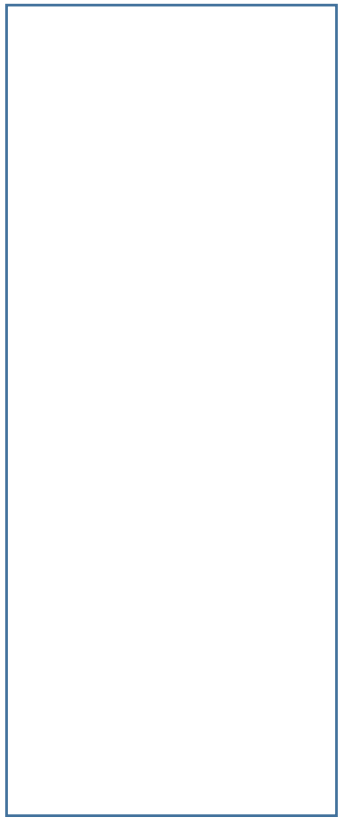


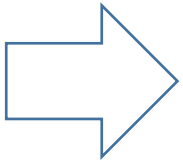
入力

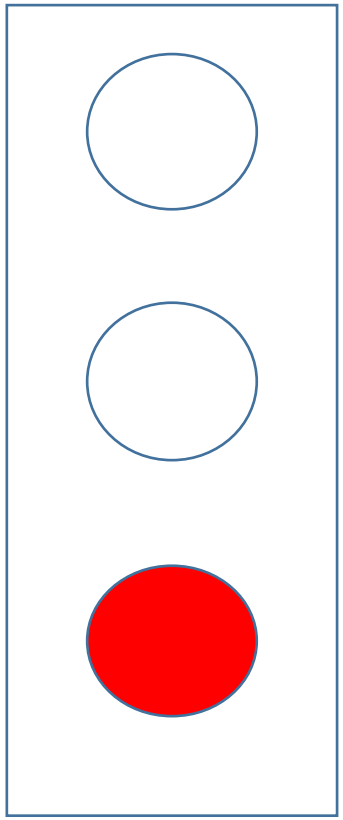
[0, 1, 0, 1]

4 個の数値









ユニット数 **3**

0.14816035

0.02995965

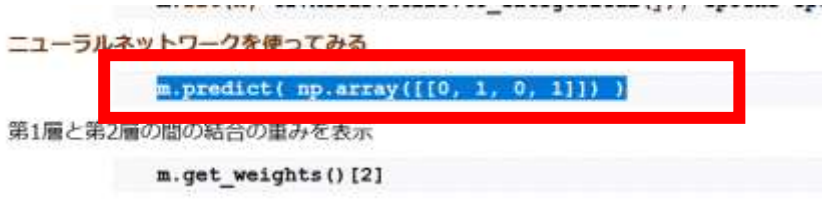
0.82188004

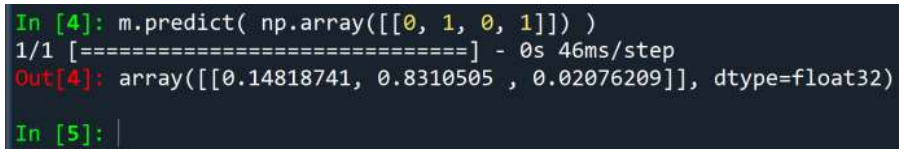
出力は，
各ユニットの
活性化の度合い

パソコン実習

■ プログラムを，コピー，貼り付け

学習では，乱数が使用されるので，各自，違った値が表示されることに注意.



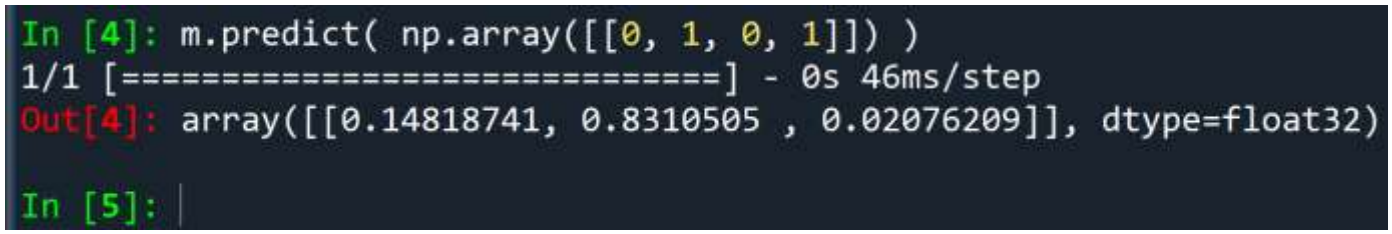


コピーしたい範囲をマウスで選び,
マウスの**右クリックメニュー**で「**コピー**」が便利

spyder の右下の　Python コンソールに張り付ける.
張り付けたあと **Enter キー**

マウスの**右クリックメニュー**で「Paste」が便利

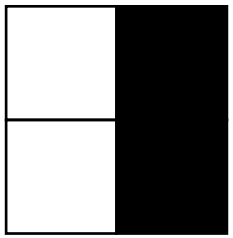
動作画面の例



入力

[0, 1, 0, 1]

4 個の数値



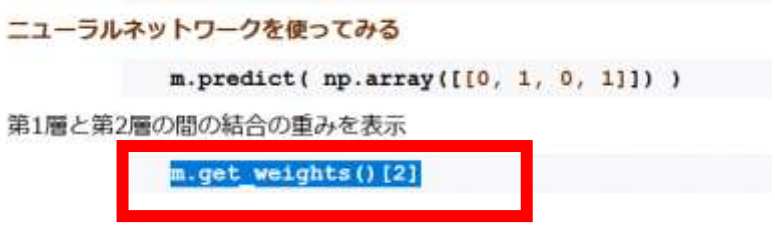
出力 [0.14818741, 0.8310505, 0.02076209]

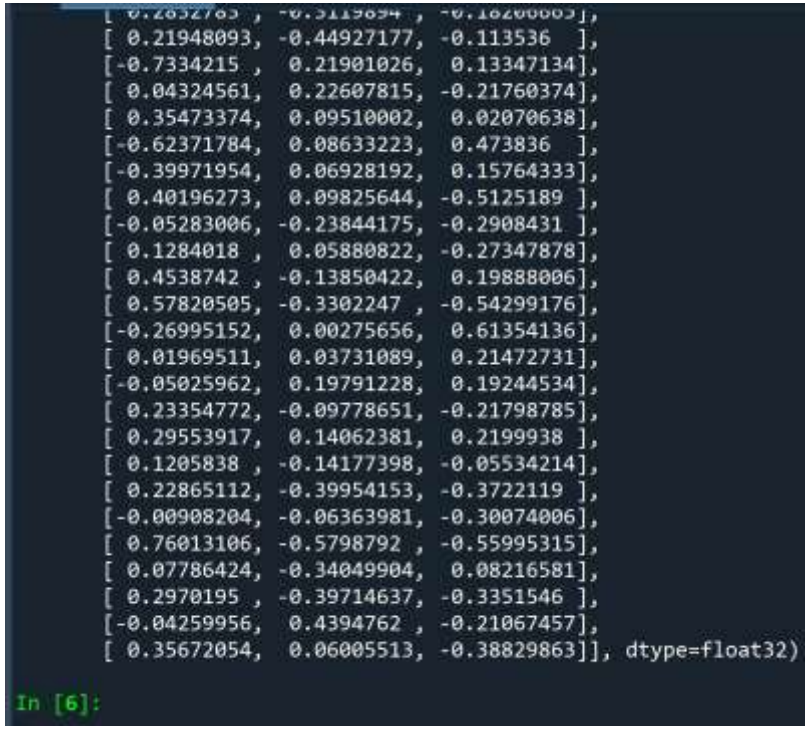
3 個の数値

0 である確率 0.14818741
1 である確率 0.8310505
2 である確率 0.02076209

結合の重みの表示

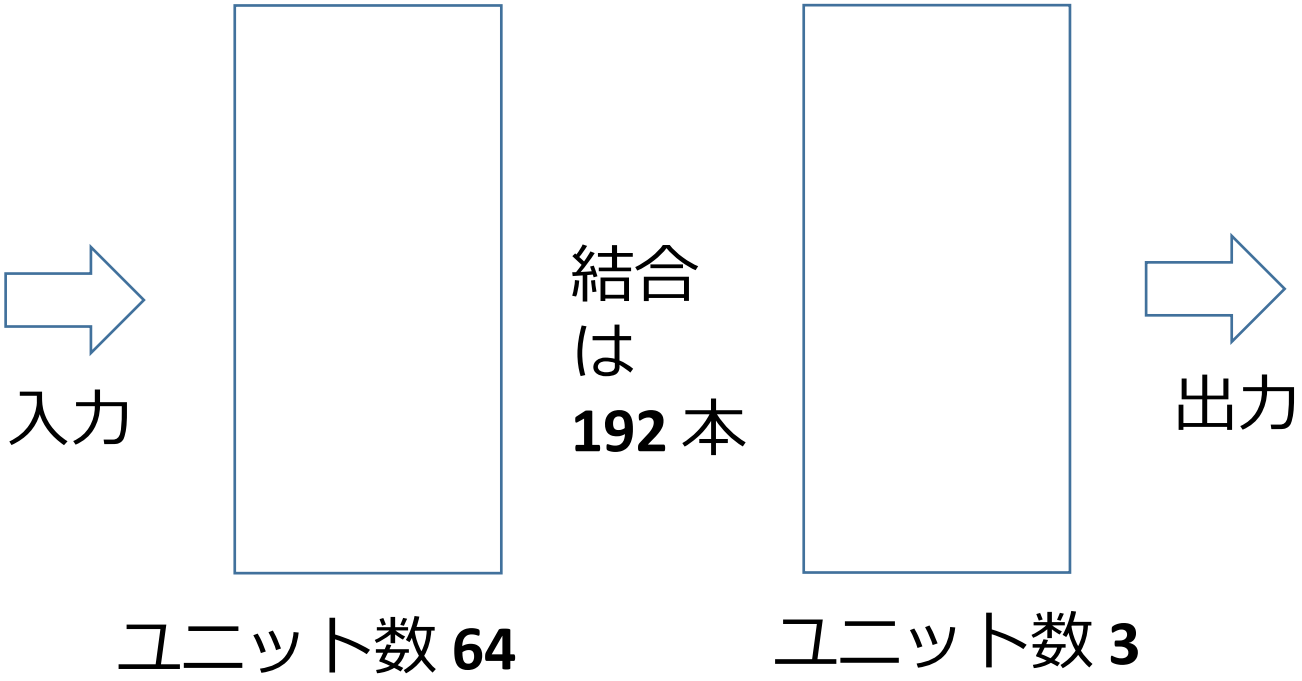
プログラムを，コピー，貼り付け





コピーしたい範囲をマウスで選び，
マウスの**右クリックメニュー**
で「**コピー**」が便利

ニューラルネットワークの例



64個のユニットと，3個のユニットとの結合は 192本

結合の重み

- ユニット間の結合の強さを「**結合の重み**」という.
- 学習では，**結合の重み**が変化する

ニューラルネットワークの使用

・学習により，ニューラルネットワークの結合の重みが変化

・学習が済んだ後，
新しい入力を与えて，出力を見る