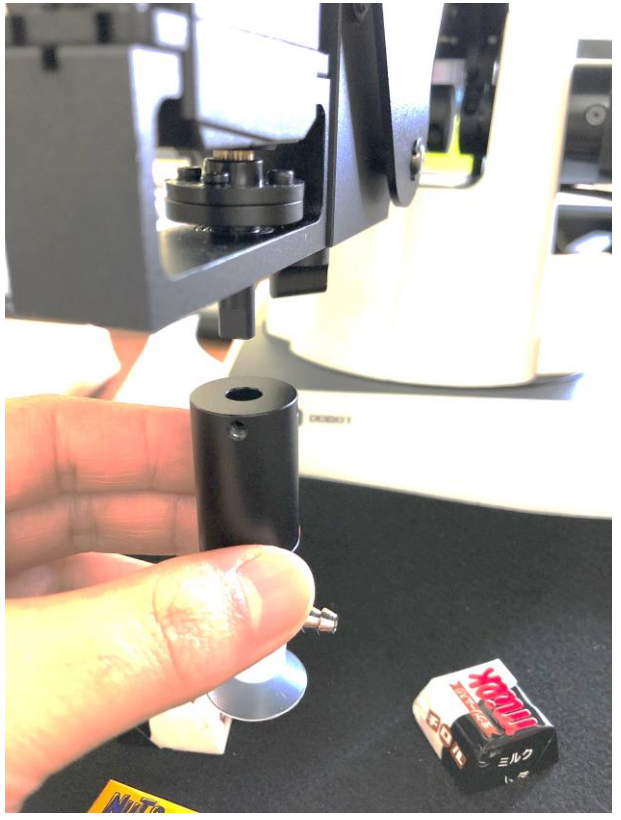
総合演習　７回目

豊橋技術科学大学　松田基

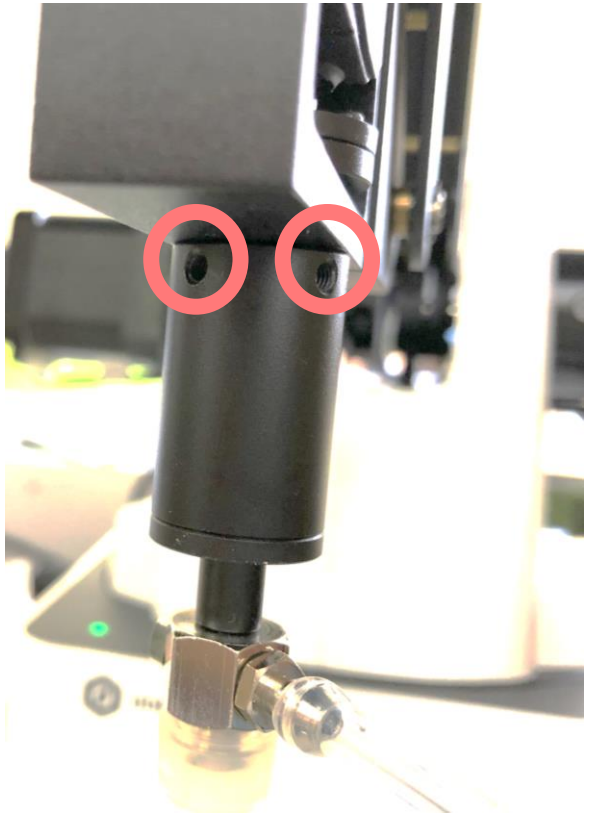
準備（カップへの付け替え）（15分）

1. グリッパのネジを緩める
<u>上のネジだけ</u>

2. 吸引カップを差し込む



3. 吸引カップのネジを緩める
ネジは２つ



6, 7回目の内容

- ロボットアーム（DOBOT）を**Pythonプログラムから制御**する。
- ロボットアームを使って、**指定位置にある対象物を運搬**する。

- AI技術の一つである「ディープラーニング」を使って**AI技術の基本的な考え方**を学ぶ。
- 「ディープラーニング」を使って**画像認識からロボットアームを制御**する。

今日の内容

- ロボットアーム（DOBOT）を**Pythonプログラムから制御**する。
- ロボットアームを使って、**指定位置にある対象物を運搬**する。

- AI技術の一つである「ディープラーニング」を使って**AI技術の基本的な考え方**を学ぶ。
- 「ディープラーニング」を使って**画像認識からロボットアームを制御**する。

前回のフィードバック（5分）

Q) アームの速さはどうやって変えるのか？

A) sample_00_change_velocity.py

```
# 初期設定
~省略~
dType.SetJOGJointParams(api, 50, 50, 50, 50, 50, 50, 50, 50, True)
dType.SetJOGCoordinateParams(api, 50, 50, 50, 50, 50, 50, 50, 50, True)
dType.SetJOGCommonParams(api, 100, 100, True)
dType.SetPTPJointParams(api, 100, 100, 100, 100, 100, 100, 100, 100, True)
dType.SetPTPCoordinateParams(api, 150, 150, 200, 200, True)
dType.SetPTPJumpParams(api, 20, 100, True)
dType.SetPTPCommonParams(api, 30, 30, True)
dType.SetHOMEParams(api, 200, 0, 0, 0, True)
dType.SetEndEffectorParams(api, 59.7 ,0, 0, 0)
```

XYZの速度　XYZの加速度　Rの速度　Rの加速度

「DobotDllType.py」に色んな関数が設定されている。（今回は特に見なくても大丈夫？）

前回のフィードバック（5分）

Q) 「06_pile_up_TIROL.py」で“JUMP”モーションを使ったけど思い通りにいかない。

A) 06_pile_up_TIROL_JUMP.py

```
＃ 1個目のブロックを移動する
dType.SetPTPJumpParams(api, 30, 100, True)
dType.SetPTPCmdEx(api, PTPMode.PTPJUMPXYZMode, 200, 100, -30, 0, True)
dType.SetEndEffectorGripperEx(api, True, True, True)
dType.dSleep(500)
dType.SetPTPCmdEx(api, PTPMode.PTPJUMPXYZMode, 250, 0, -30, 0, True)
dType.SetEndEffectorGripperEx(api, True, False, True)
dType.dSleep(500)
```

この宣言だけで
上への回避ができる！

注意！
PTP“JUMP”XYZMode を使うこと

今回は割愛

【AIなし分類】の説明（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

□手順

1. WEBカメラで複数のチロルチョコが写った画像を取得
2. 画像からチロルチョコの部分だけ切り取り、テンプレート画像として保存
3. WEBカメラから取得した画像内のチロルチョコの分類

今回は割愛

【AIなし分類】の説明（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

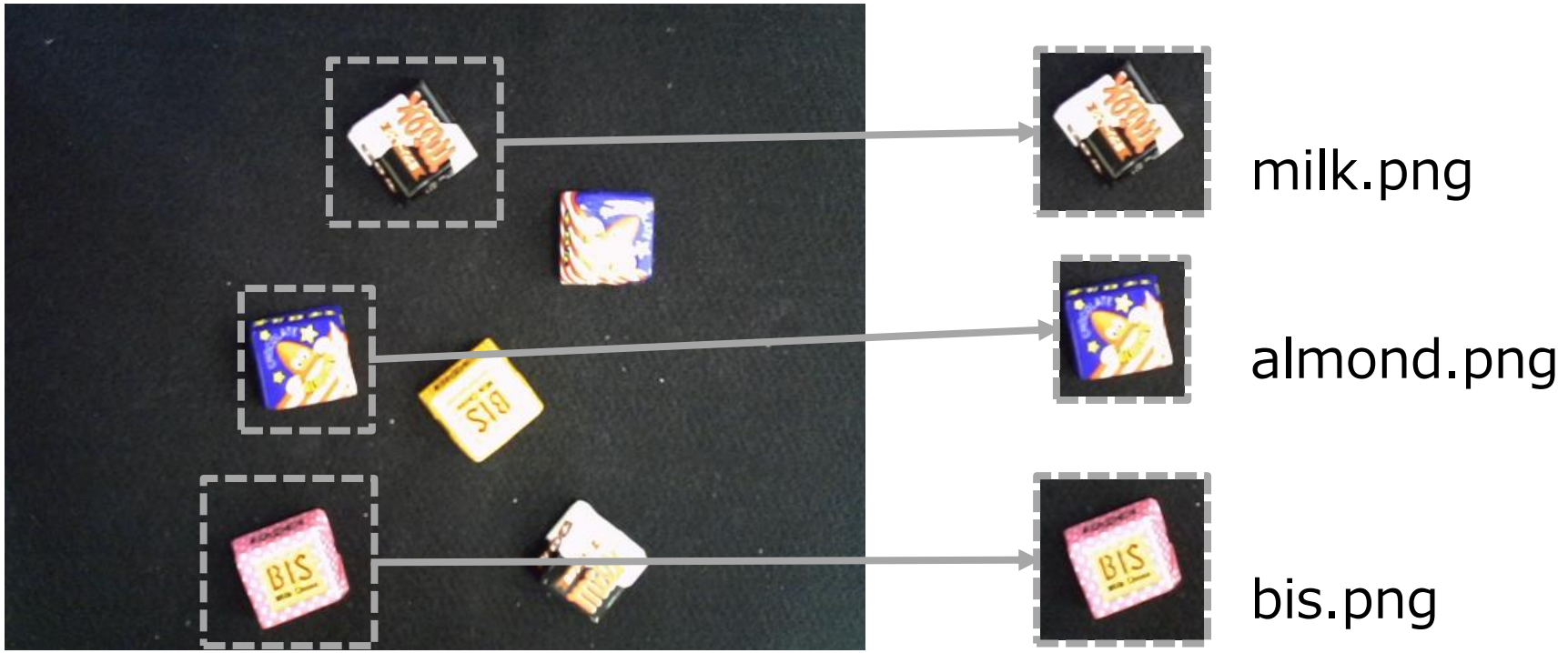
1. WEBカメラで複数のチロルチョコが写った画像を取得

今回は割愛

【AIなし分類】の説明（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

2. 画像からチロルチョコの部分だけ切り取り、テンプレート画像として保存

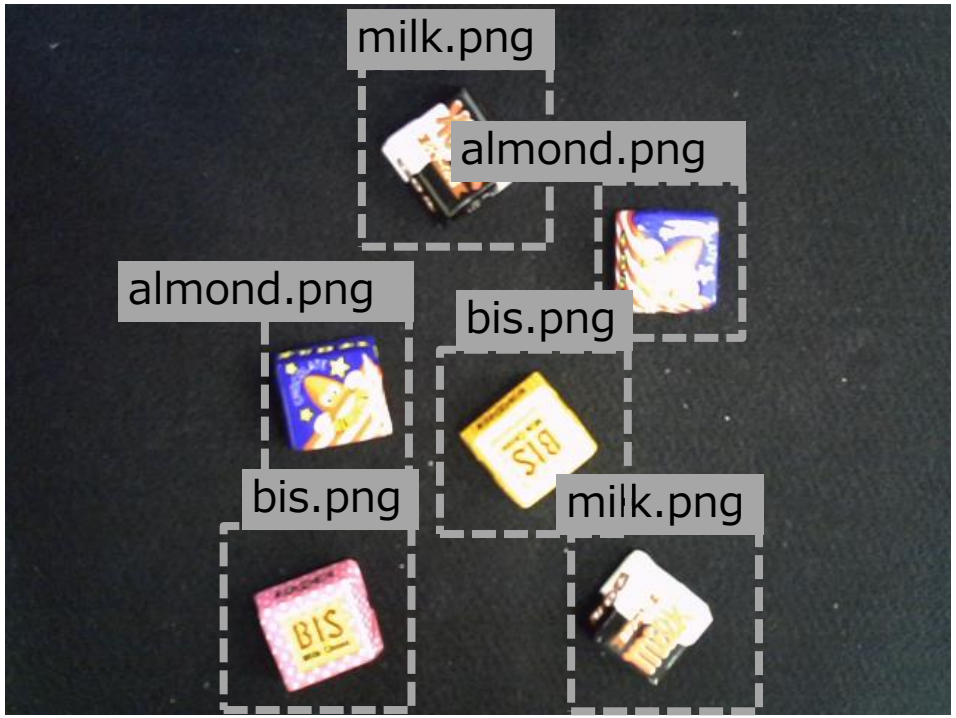


今回は割愛

【AIなし分類】の説明（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

3. WEBカメラから取得した画像内のチロルチョコの分類



今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
import os

import cv2
import numpy as np
from enum import IntEnum, auto

import cameraSetting as camset

# OpenCV画面出力用フォント設定
font = cv2.FONT_HERSHEY_DUPLEX
FONT_SIZE  = 0.42
FONT_WIDTH = 1
```

ライブラリのインポート

「cameraSetting.py」をインポート

カメラ画像の設定

```
# 描画用
draw_white  = (200, 200, 200)
draw_black  = ( 50,  50,  50)
draw_red    = ( 50,  50, 200)
draw_blue   = (200,  50,  50)
draw_green  = ( 50, 200,  50)
draw_yellow = ( 50, 200, 200)

# テンプレートマッチングの閾値 (1)
MATCH_THRESHOLD = 0.98
```

画像内に表示する色の定義

98%一致していれば同じとみなす

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
# テンプレートマッチングで検出する（2）
if __name__ == '__main__':
    # VideoCaptureのインスタンスを作成する（2-1）
    cap = cv2.VideoCapture(1)

    print("¥n - - - - - - - - - - ")
    # camset.camera_set(cv2, cap, gain = **調整した値**, exposure = **調整した値**.)
    camset.camera_get(cv2, cap)
    print(" - - - - - - - - - - ¥n")
```

「WEBカメラ1」を設定（PCによって変わる）

WEBカメラの設定情報を取得する

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
while True:
    # VideoCaptureから1フレーム読み込む（2-2）
    ret, frame = cap.read()
    ret, edframe = cap.read()
    # 加工なし画像を表示する
    cv2.imshow('Raw Frame', frame)

    for file in os.listdir("./template"):
        if file == "DS_Store":
            continue
        # テンプレート画像を読み込む（2-4）
        template = cv2.imread("./template/" + file)
        # 画像サイズを取得
        h, w = template.shape[:2]
```

プログラムが終了するまで
画像を更新し続ける
（動画のように見える）

WEBカメラの画像をそのまま表示

「template」フォルダにある
画像の数だけ繰り返す

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
# OpenCVで画像部分一致を検索（2-5）
result = cv2.matchTemplate(frame, template, cv2.TM_CCORR_NORMED)
# 検出結果から検出領域の位置を取得（2-6）
loc = np.where(result > MATCH_THRESHOLD)

# 検出結果の描画（7）
for cnt, top_left in enumerate(zip(*loc[::-1])):
    # 検出結果から左上座標を取り出し、テンプレート画像のサイズから右下座標を作成する
    bottom_right = (top_left[0] + w, top_left[1] + h)
    # 長方形を描画する。
    cv2.rectangle(edframe, top_left, bottom_right, draw_white, thickness = 1)
```

一致率が98％を超えたら同じと見なす

分類した数だけ繰り返す

チロルチョコを囲う長方形を画像内に描画

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
# ラベルを表示する
label = "%s - %.4f" % (file, result[loc][cnt])
cv2.rectangle(edframe, (top_left[0], top_left[1]-15),
    (top_left[0]+len(label)*8, top_left[1]), draw_white, -1, cv2.LINE_AA)
cv2.putText(edframe, label, (top_left[0], top_left[1]-4), font, FONT_SIZE,
    draw_black, FONT_WIDTH, cv2.LINE_AA)
```

ラベル（どの分類か）を表示する背景の長方形を描画

ラベル（どの分類か）を文字で表示

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
# 描画した画像を表示
cv2.imshow('Edited Frame', edframe)

# キー入力を1ms待つ
k = cv2.waitKey(1)

#「ESC（27）」キーを押す
# プログラムを終了する
if k == 27:
    break
```

分類した結果を画像内に表示

「ESC」キーを押すとプログラムが終了

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
# 「C」キーを押す
# WEBカメラのゲイン値、露出の値を調整する
elif k == ord('c'):
    g = input("gain     : ")
    e = input("exposure : ")
    print("¥n - - - - - - - - - - ")
    camset.camera_set(cv2, cap, gain = float(g), exposure = float(e))
    camset.camera_get(cv2, cap)
    print(" - - - - - - - - - - ¥n")
```

カメラ画像の明るさを変更

今回は割愛

【AIなし分類】の説明（15分）

opencv_templateMatch.py

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

```
        # 「S」キーを押す（2-3）
        # カメラ画像をそのまま保存する
        elif k == ord('s'):
            capstr = "frame.png"
            cv2.imwrite(capstr, frame)
            print(capstr)

    # キャプチャをリリースして、ウィンドウをすべて閉じる（2-8）
cap.release()
cv2.destroyAllWindows()
```

加工なしの（分類結果が描画されていない）画像を、「frame.png」として保存

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

1. WEBカメラで複数のチロルチョコが写った画像を取得

① 「opencv_templateMatch.py」を実行

```
> cd **配布資料のアドレス**¥プログラム¥dobot_templateMatch
> python opencv_templateMatch.py
```

② カメラ画像のウィンドウが出たら、ウィンドウをクリックしたのち「S」キーを押す
→「dobot_templateMatch」フォルダ内に「frame.png」が保存される

※ 画像が明るすぎる／暗すぎる場合は、「C」キーを押すとコマンドプロンプトから「gain」と「exposure」（明るさ）を変更できる。

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

2. 画像からチロルチョコの部分だけ切り取り、テンプレート画像として保存

① 「frame.png」をフォト、ペイント等で開きチロルチョコを切り取り、任意の名前でテンプレート画像として保存する（拡大・縮小は禁止）。

② 「template」フォルダにすべてのテンプレート画像を保存する。

※手元に合計6個、3種類のチロルチョコがあります。
1種類につき1つの画像を切り取ってください。
（計3つの画像）

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

3. WEBカメラから取得した画像内のチロルチョコの分類

① 再び「opencv_templateMatch.py」を実行

> *python opencv_templateMatch.py*

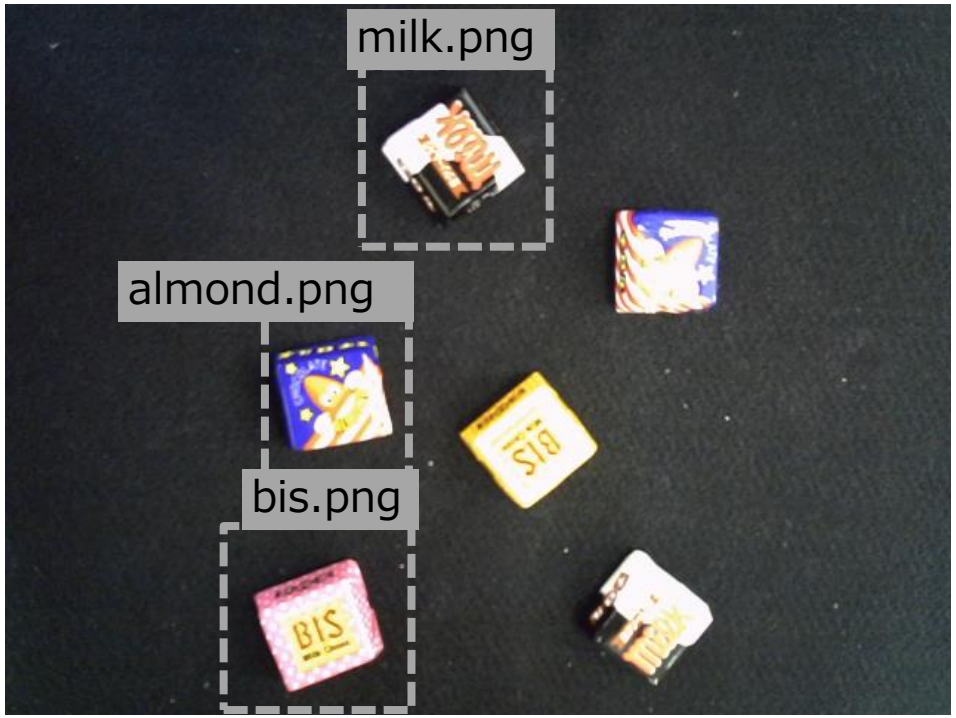
② テンプレート画像を用意した種類のチロルチョコが認識されているか確認する。

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

➢同じ種類でも、**テンプレート画像として用意したチロルチョコしか分類できない。**



milk.png

almond.png

bis.png

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

➢同じ種類でも、**テンプレート画像として用意したチロルチョコしか分類できない。**

□「単純なテンプレートマッチング」の欠点

- テンプレートとの適合率が低いと分類できない。
  回転、拡大縮小、ノイズ、環境による誤差を大きく受ける。
- 対象物が複雑な場合は識別が難しい。
  チロルチョコのような、混色や濃淡による差がある対象物には適さない。

今回は割愛

【AIなし分類】の実行（15分）

◆カメラ画像から、チロルチョコを分類する。（テンプレートマッチング）

➢同じ種類でも、**テンプレート画像として用意したチロルチョコしか分類できない。**

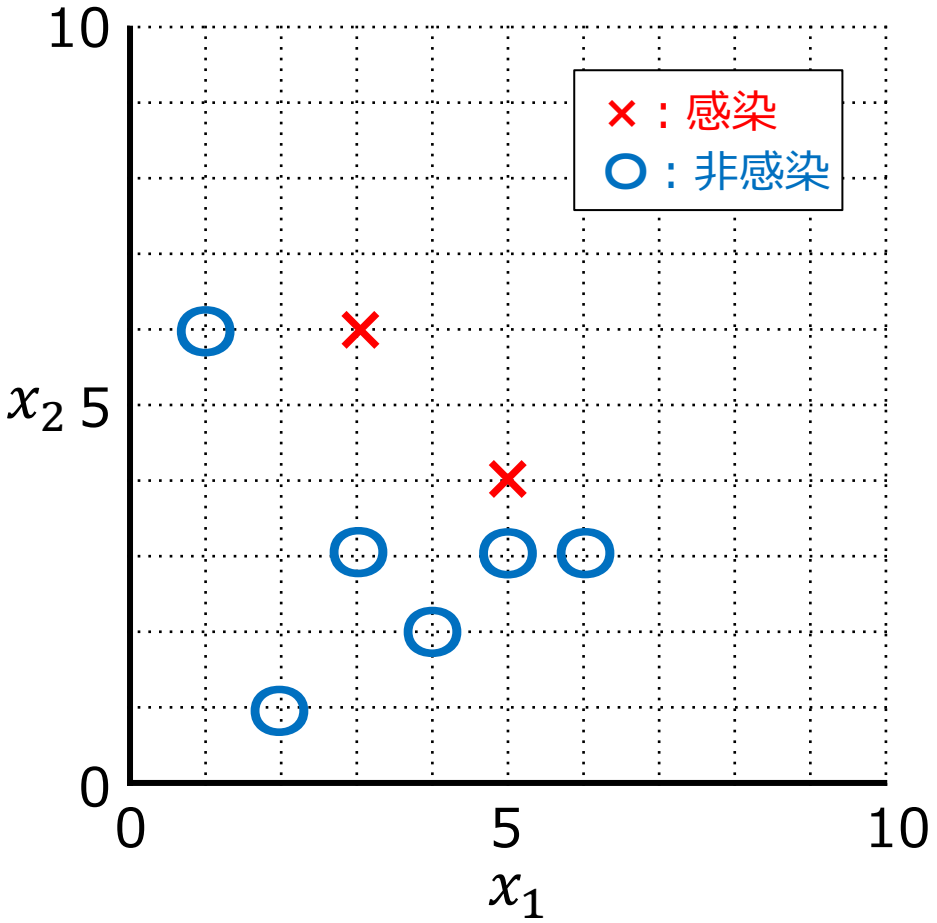
□「単純なテンプレートマッチング」の欠点

- テンプレートとの適合率が低いと分類できない。
  回転、拡大縮小、ノイズ、環境による誤差を大きく受ける。
- 対象物が複雑な場合は識別が難しい。
  チロルチョコのような、混色や濃淡による差がある対象物には適さない。

☞AIを使った分類（ニューラルネットワークを例に）

簡単な分類　課題1

- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



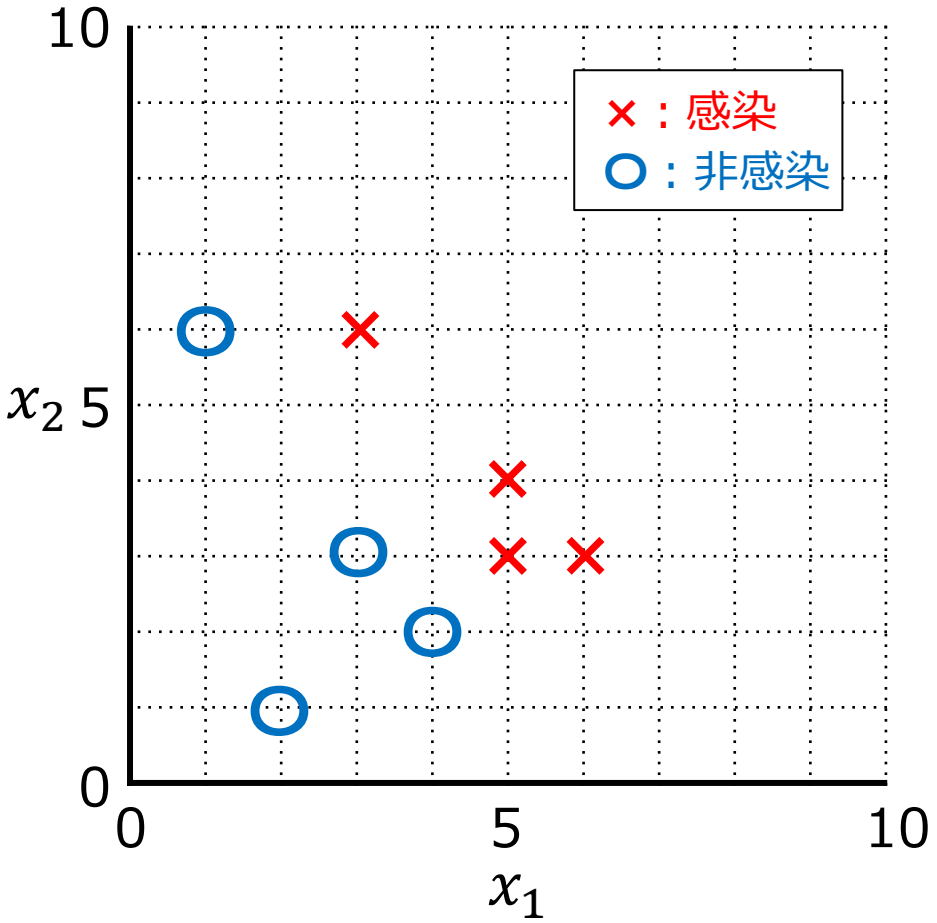
- (x_1, x_2)：検査結果を数値化したもの（例：体温）

- 直線だけを使って、感染・非感染を分類しましょう。
- 分類条件 を考えましょう。
- ニューラルネットワークの図で書いてみましょう。

ヒント 直線なら何本使っても良いです。

簡単な分類

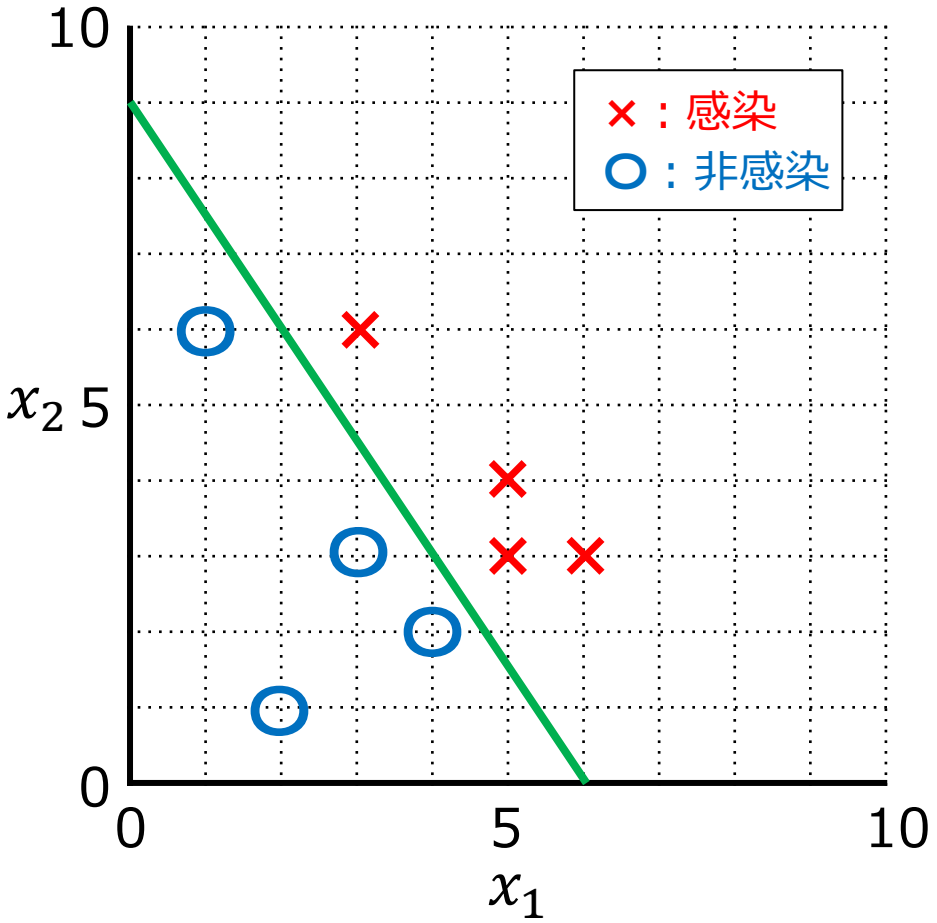
- 問題0：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



- (x_1, x_2)：検査結果を数値化したもの（例：体温）

簡単な分類

- 問題0：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



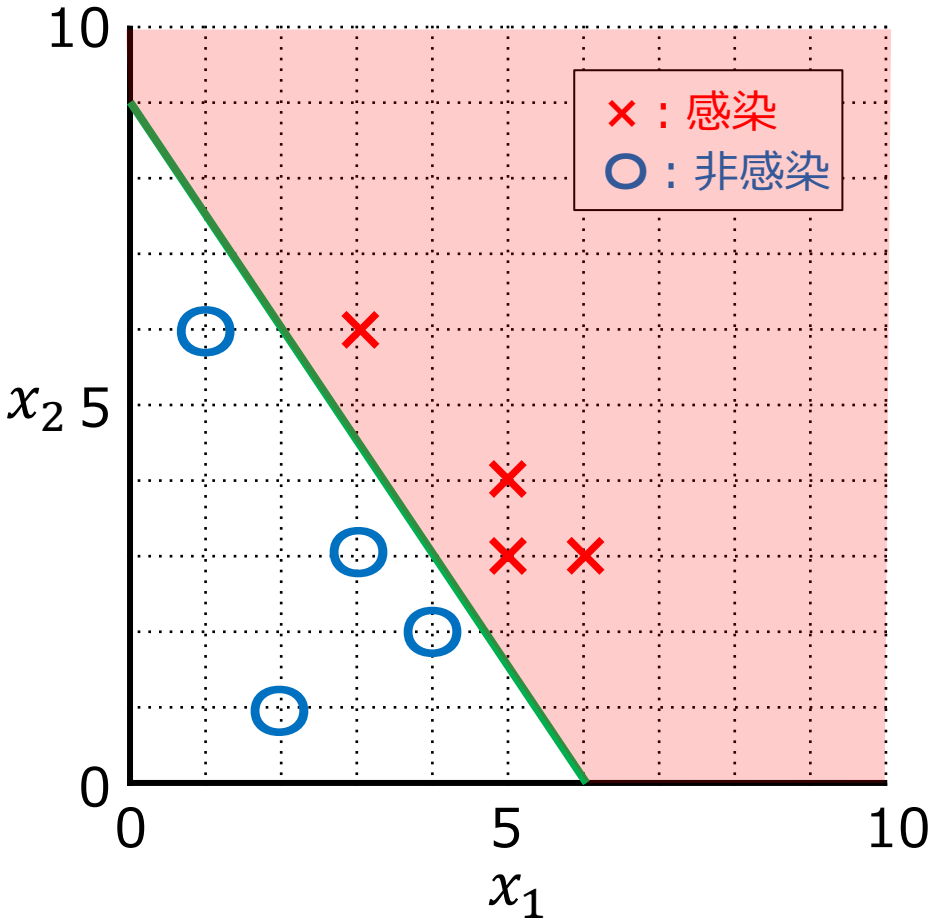
- (x_1, x_2)：検査結果を数値化したもの（例：体温）

$$x_2 = ax_1 + b \ \text{☞} \ a_1x_1 + a_2x_2 + b_1 = 0$$

$$x_2 = -\frac{3}{2}x_1 + 9 \ \text{☞} \ 3x_1 + 2x_2 - 18 = 0$$

簡単な分類

- 問題0：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



- (x_1, x_2)：検査結果を数値化したもの（例：体温）

$$x_2 = ax_1 + b \ \text{☞} \ a_1x_1 + a_2x_2 + b_1 = 0$$

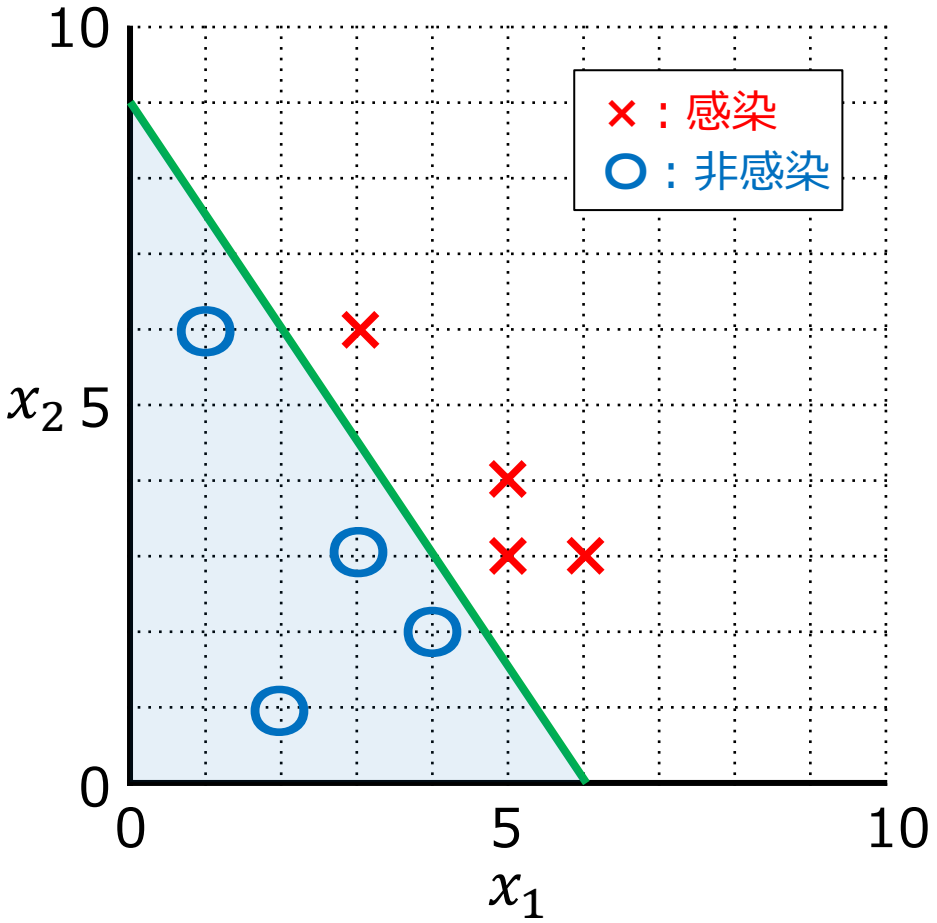
$$x_2 = -\frac{3}{2}x_1 + 9 \ \text{☞} \ 3x_1 + 2x_2 - 18 = 0$$

◆分類条件

- $3x_1 + 2x_2 - 18 > 0$の時：感染
- $3x_1 + 2x_2 - 18 \leqq 0$の時：非感染

簡単な分類

- 問題0：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



- (x_1, x_2)：検査結果を数値化したもの（例：体温）

$$x_2 = ax_1 + b \ \text{☞} \ a_1x_1 + a_2x_2 + b_1 = 0$$

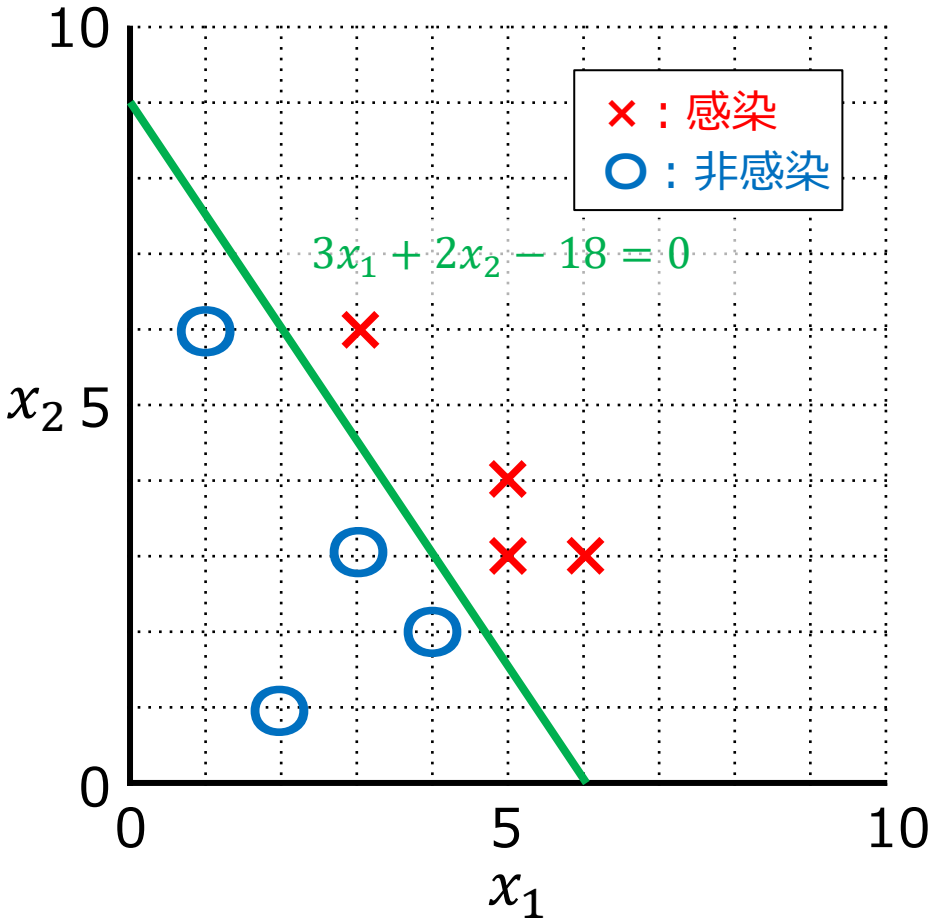
$$x_2 = -\frac{3}{2}x_1 + 9 \ \text{☞} \ 3x_1 + 2x_2 - 18 = 0$$

◆分類条件

- $3x_1 + 2x_2 - 18 > 0$の時：感染
- $3x_1 + 2x_2 - 18 \leqq 0$の時：非感染

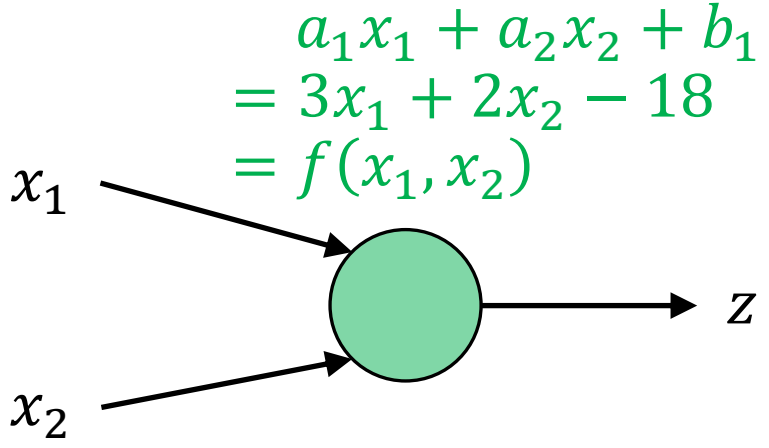
簡単な分類

- 問題0：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



- (x_1, x_2)：検査結果を数値化したもの（例：体温）

- ニューラルネットワーク風に書いてみる。

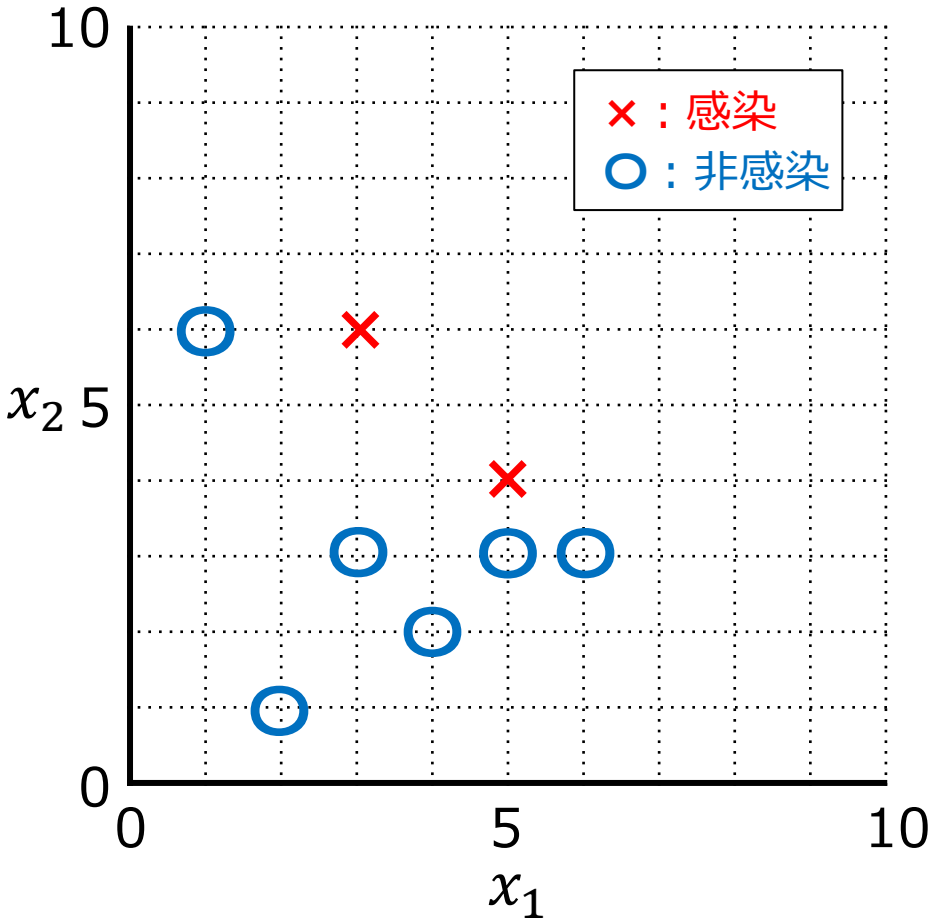


今回は
$f(x_1, x_2) > 0$の時$z = 1$
$f(x_1, x_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題1

- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。

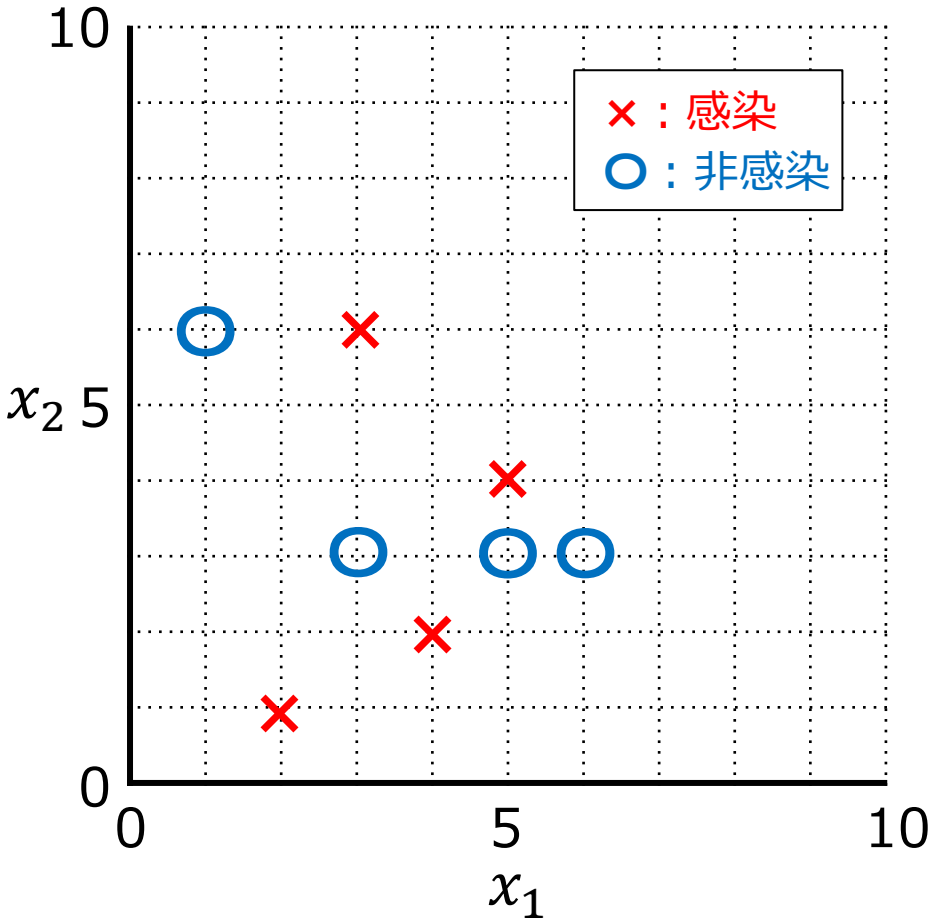


- (x_1, x_2)：検査結果を数値化したもの（例：体温）

- 直線だけを使って、感染・非感染を分類しましょう。
- 分類条件 を考えましょう。
- ニューラルネットワークの図で書いてみましょう。

簡単な分類　課題2

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。

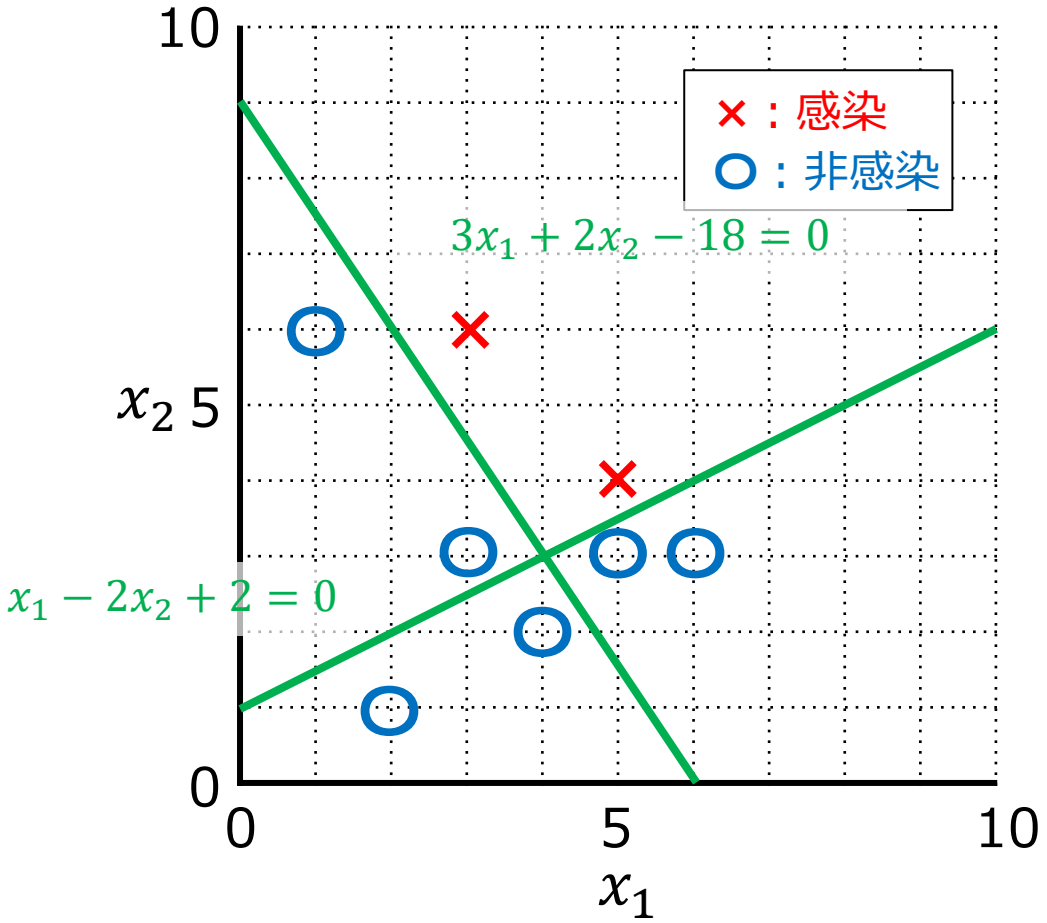


- (x_1, x_2)：検査結果を数値化したもの（例：体温）

- 直線だけを使って、感染・非感染を分類しましょう。
- 分類条件 を考えましょう。
- ニューラルネットワークの図で書いてみましょう。

簡単な分類　課題1の解答例

- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



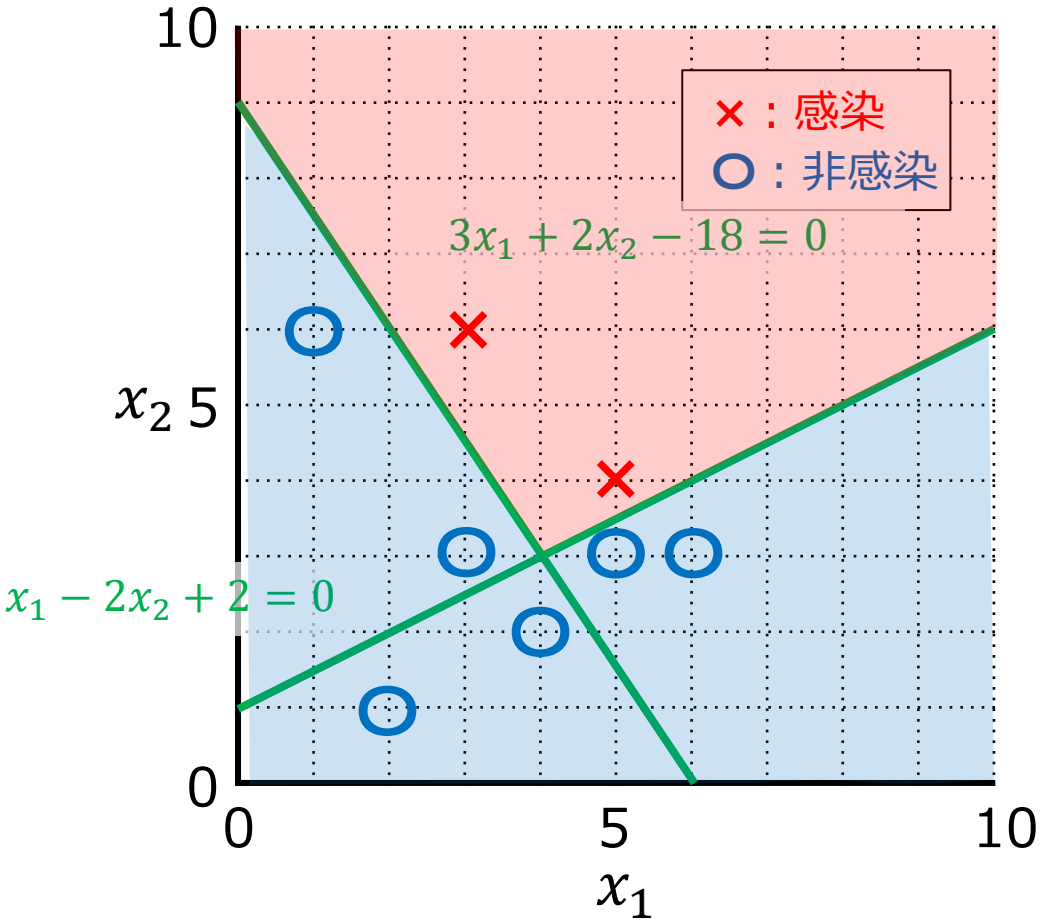
- 2本の直線を引くと、分類できる。

$$x_2 = -\frac{3}{2}x_1 + 9 \quad ☞ \quad 3x_1 + 2x_2 - 18 = 0$$

$$x_2 = \frac{1}{2}x_1 + 1 \quad ☞ \quad x_1 - 2x_2 + 2 = 0$$

簡単な分類　課題1の解答例

- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ 2本の直線を引くと、分類できる。

$$x_2 = -\frac{3}{2}x_1 + 9 \quad ☞ \quad 3x_1 + 2x_2 - 18 = 0$$

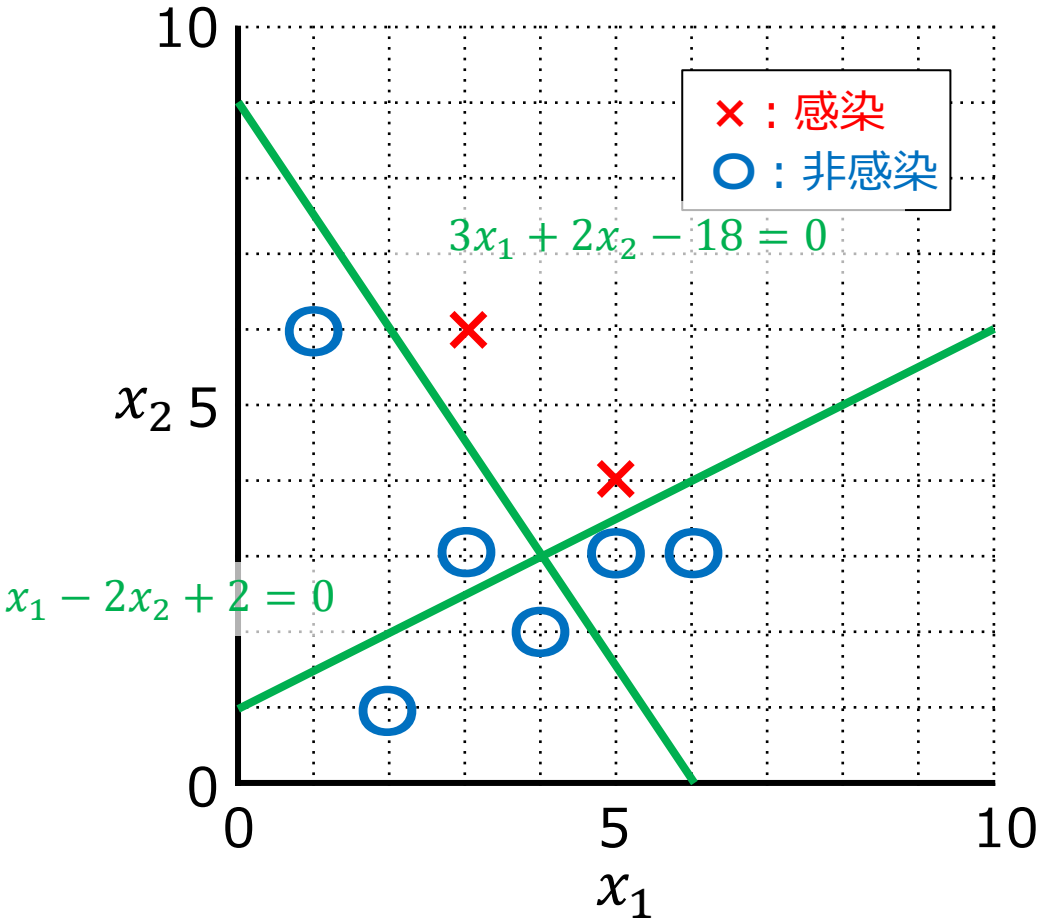
$$x_2 = \frac{1}{2}x_1 + 1 \quad ☞ \quad x_1 - 2x_2 + 2 = 0$$

◆ 分類条件

- $3x_1 + 2x_2 - 18 > 0$ かつ $x_1 - 2x_2 + 2 \leqq 0$ の時：感染
- $3x_1 + 2x_2 - 18 > 0$ かつ $x_1 - 2x_2 + 2 \leqq 0$ でない時：非感染

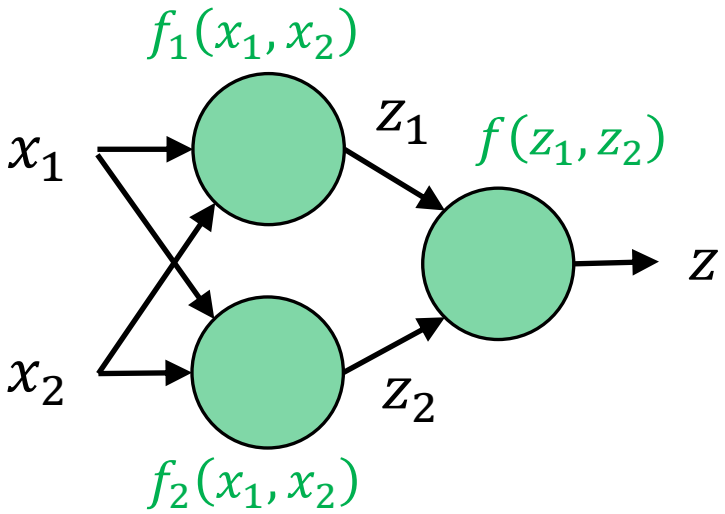
簡単な分類　課題1の解答例

- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ ニューラルネットワークの例

- $f_1(x_1, x_2) = 3x_1 + 2x_2 - 18$
- $f_2(x_1, x_2) = x_1 - 2x_2 + 2$
- $f(z_1, z_2) = z_1 - z_2$



$f_1(x_1, x_2) > 0$の時$z_1 = 1$
$f_1(x_1, x_2) \leqq 0$の時$z_1 = -1$

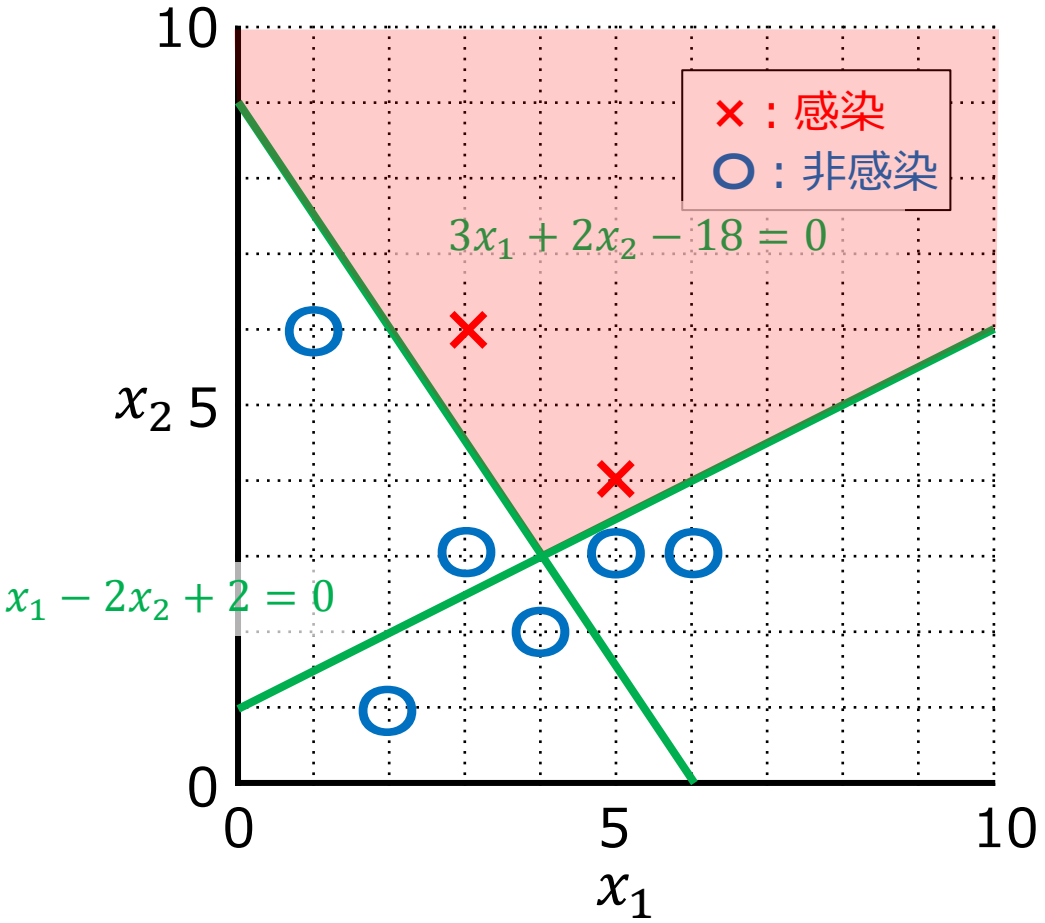
$f_2(x_1, x_2) > 0$の時$z_2 = 1$
$f_2(x_1, x_2) \leqq 0$の時$z_2 = -1$

$f(z_1, z_2) > 0$の時$z = 1$
$f(z_1, z_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題1の解答例

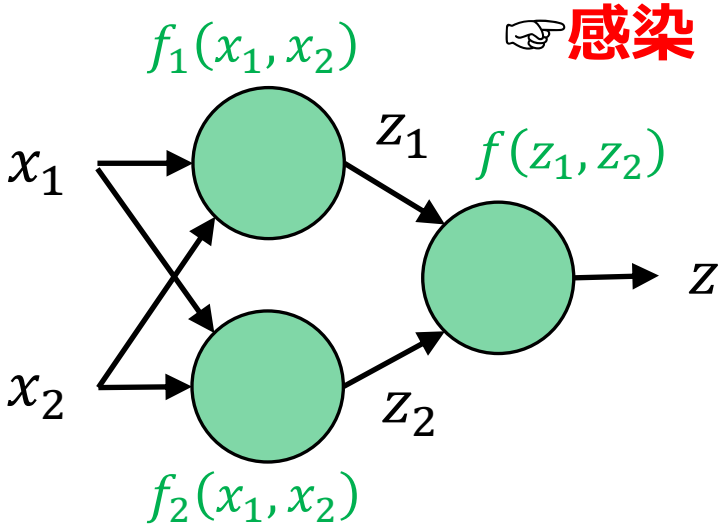
- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_1 = 1$ かつ $z_2 = -1$ の時

- $f(z_1, z_2) = z_1 - z_2$
  $= 1 + 1 = 2 > 0$
- $f(z_1, z_2) > 0$の時 $z = 1$

☞**感染**



$f_1(x_1, x_2) > 0$の時$z_1 = 1$
$f_1(x_1, x_2) \leqq 0$の時$z_1 = -1$

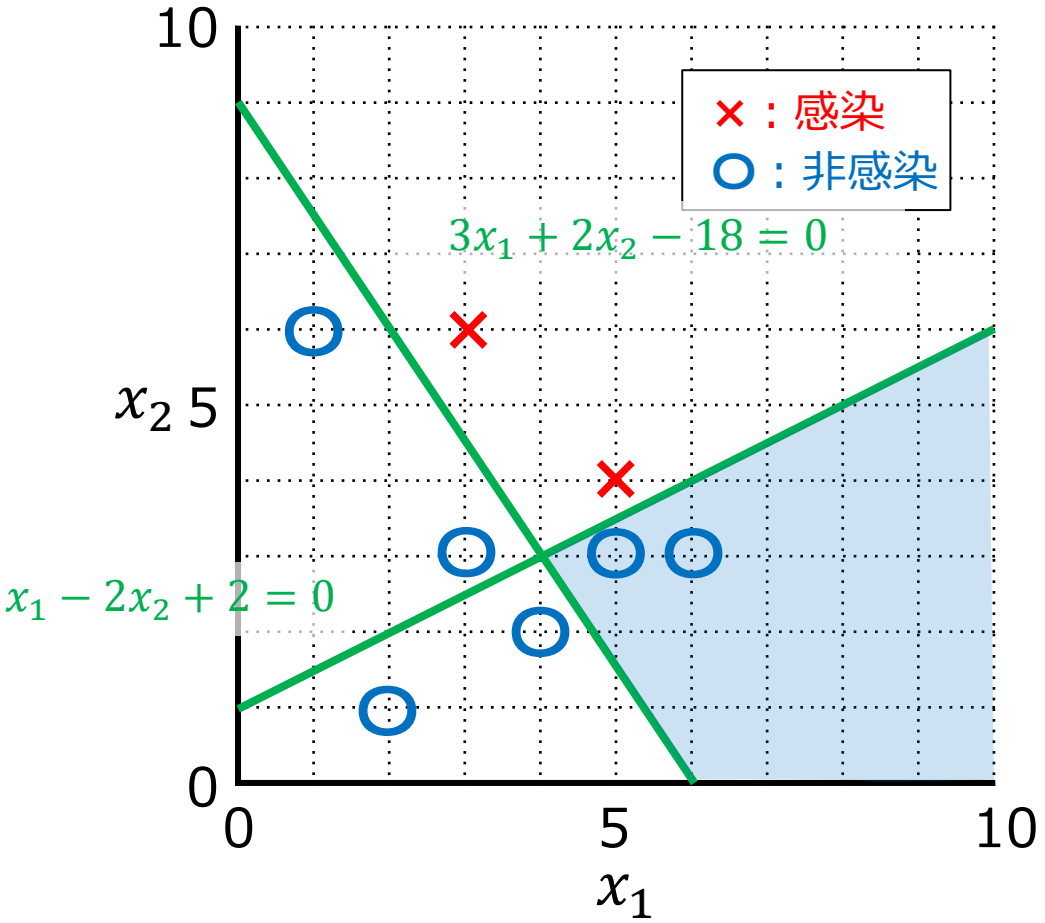
$f_2(x_1, x_2) > 0$の時$z_2 = 1$
$f_2(x_1, x_2) \leqq 0$の時$z_2 = -1$

$f(z_1, z_2) > 0$の時$z = 1$
$f(z_1, z_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題1の解答例

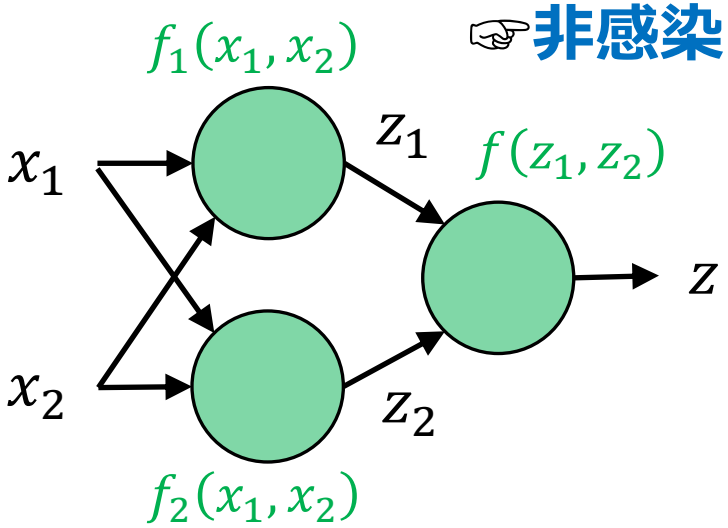
- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_1 = 1$ かつ $z_2 = 1$ の時

- $f(z_1, z_2) = z_1 - z_2$
  $= 1 - 1 = 0 \leqq 0$
- $f(z_1, z_2) \leqq 0$の時 $z = -1$

☞**非感染**



$f_1(x_1, x_2) > 0$の時$z_1 = 1$
$f_1(x_1, x_2) \leqq 0$の時$z_1 = -1$

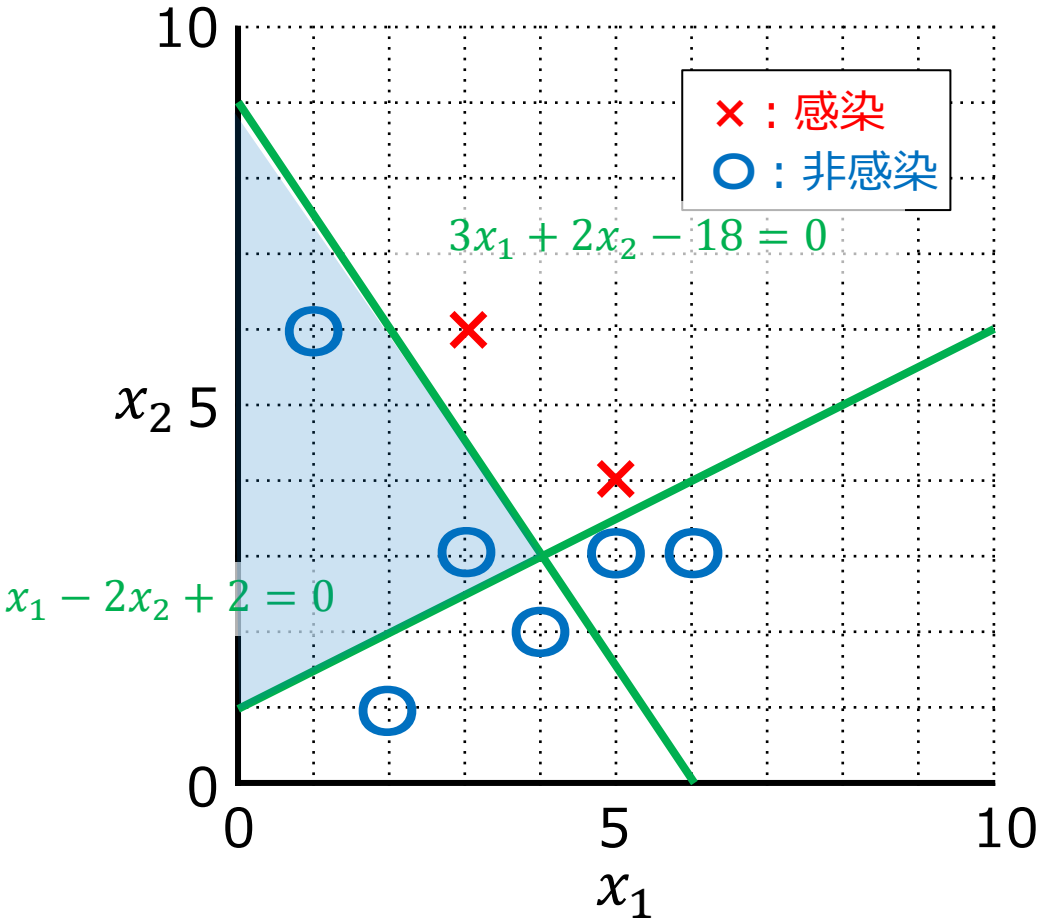
$f_2(x_1, x_2) > 0$の時$z_2 = 1$
$f_2(x_1, x_2) \leqq 0$の時$z_2 = -1$

$f(z_1, z_2) > 0$の時$z = 1$
$f(z_1, z_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題1の解答例

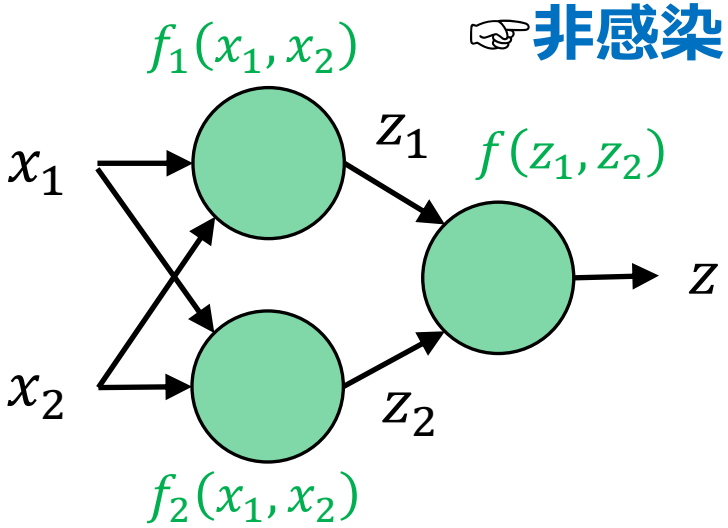
- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_1 = -1$ かつ $z_2 = -1$ の時

- $f(z_1, z_2) = z_1 - z_2$
  $= -1 + 1 = 0 \leqq 0$
- $f(z_1, z_2) \leqq 0$の時 $z = -1$

☞**非感染**



$f_1(x_1, x_2) > 0$の時$z_1 = 1$
$f_1(x_1, x_2) \leqq 0$の時$z_1 = -1$

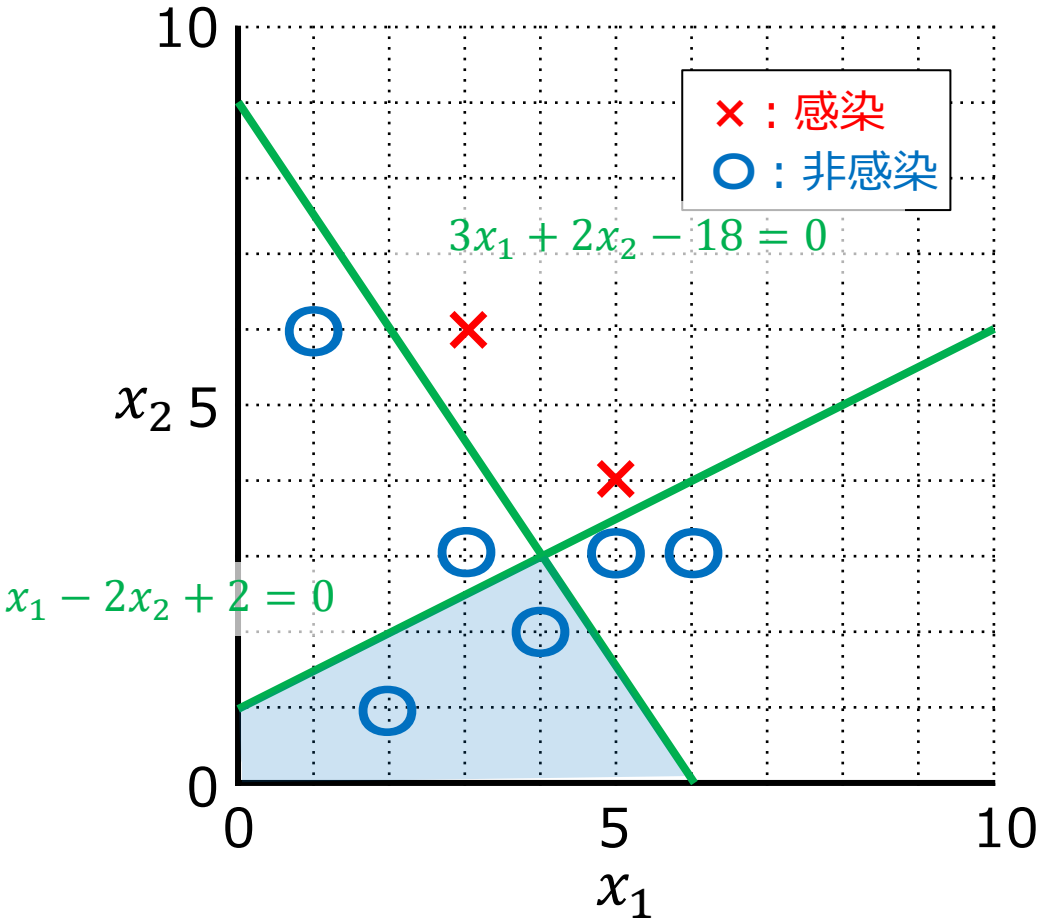
$f_2(x_1, x_2) > 0$の時$z_2 = 1$
$f_2(x_1, x_2) \leqq 0$の時$z_2 = -1$

$f(z_1, z_2) > 0$の時$z = 1$
$f(z_1, z_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題1の解答例

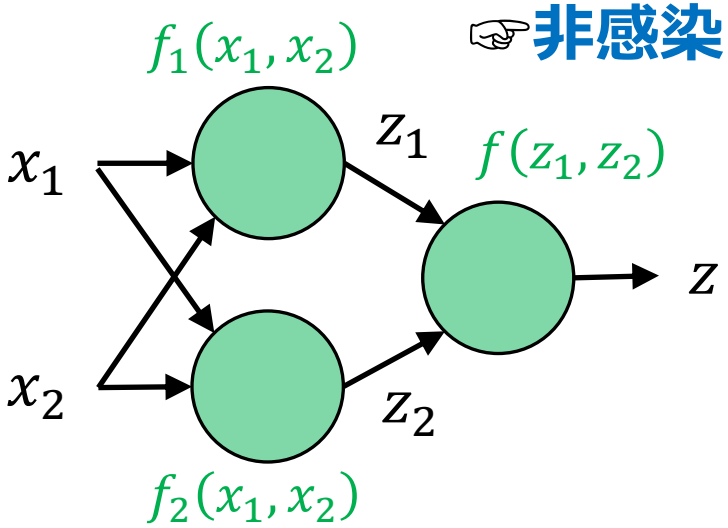
- 問題1：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_1 = -1$ かつ $z_2 = 1$ の時

- $f(z_1, z_2) = z_1 - z_2$
  $= -1 - 1 = -2 \leqq 0$
- $f(z_1, z_2) \leqq 0$の時 $z = -1$

☞**非感染**



$f_1(x_1, x_2) > 0$の時$z_1 = 1$
$f_1(x_1, x_2) \leqq 0$の時$z_1 = -1$

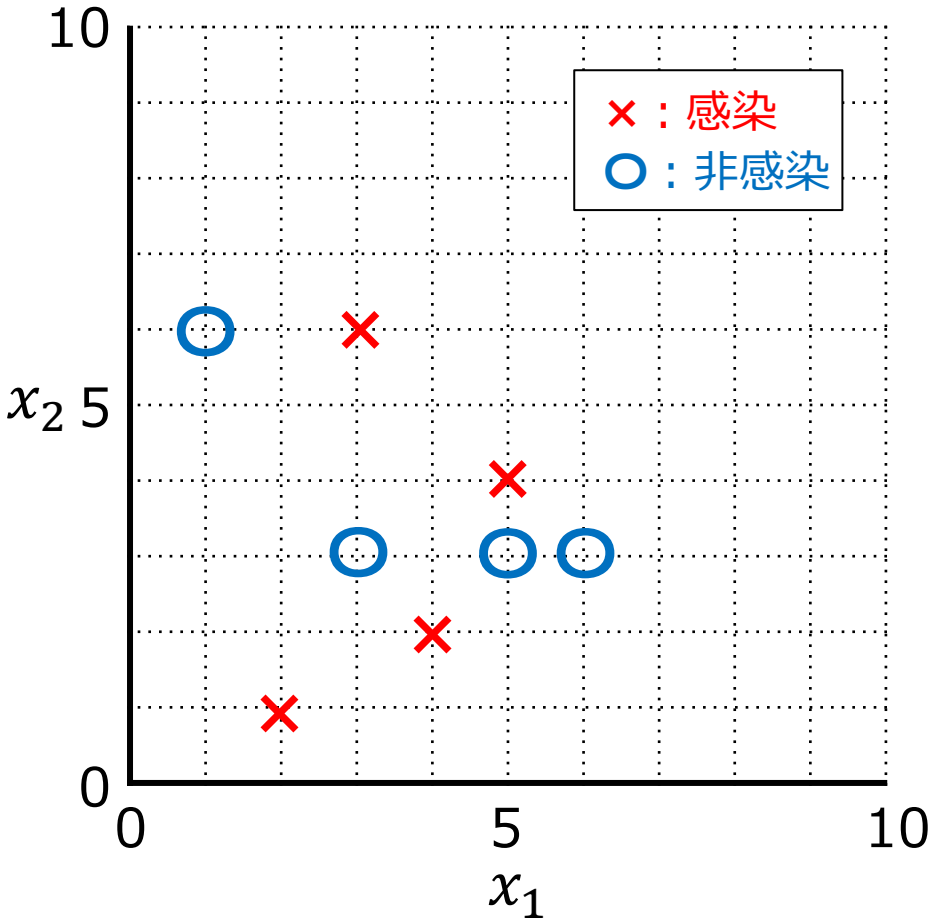
$f_2(x_1, x_2) > 0$の時$z_2 = 1$
$f_2(x_1, x_2) \leqq 0$の時$z_2 = -1$

$f(z_1, z_2) > 0$の時$z = 1$
$f(z_1, z_2) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題2

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



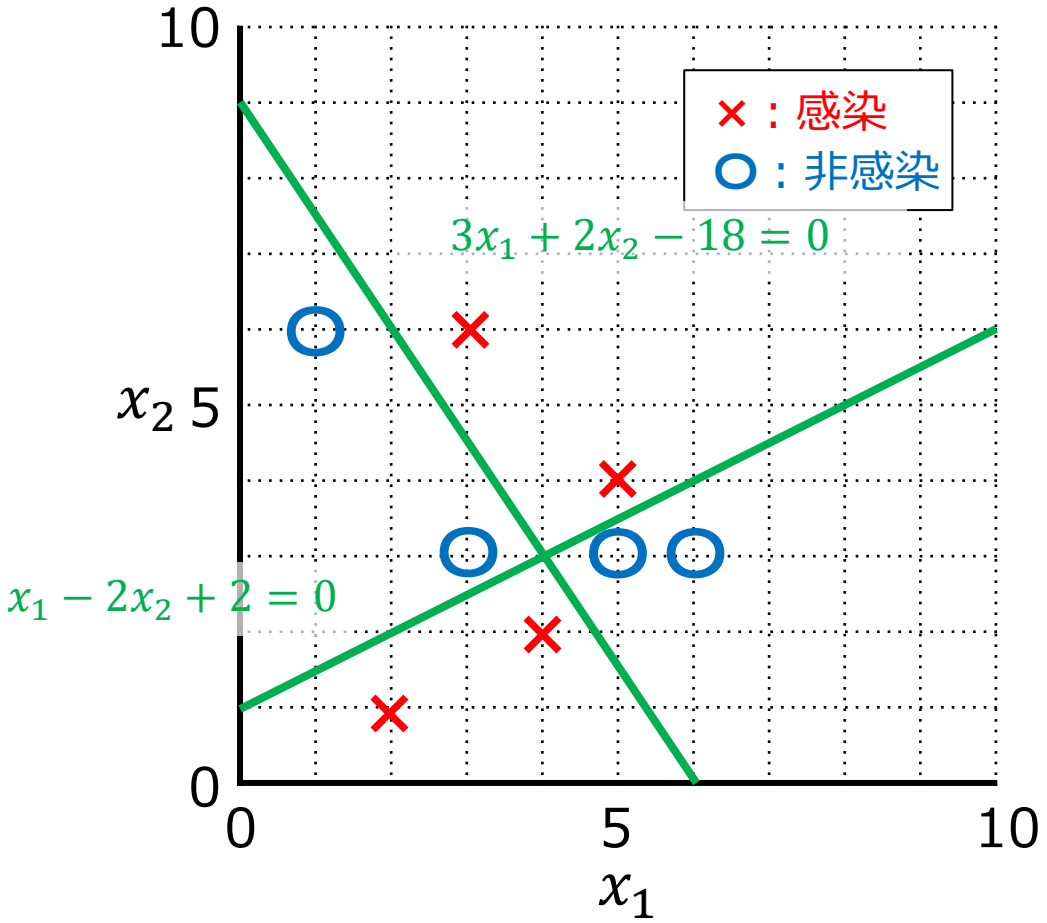
- (x_1, x_2)：検査結果を数値化したもの（例：体温）

- 直線だけを使って、感染・非感染を分類しましょう。
- 分類条件 を考えましょう。
- ニューラルネットワークの図で書いてみましょう。

ヒント 途中までは課題1と同じです。

簡単な分類　課題2の解答例

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



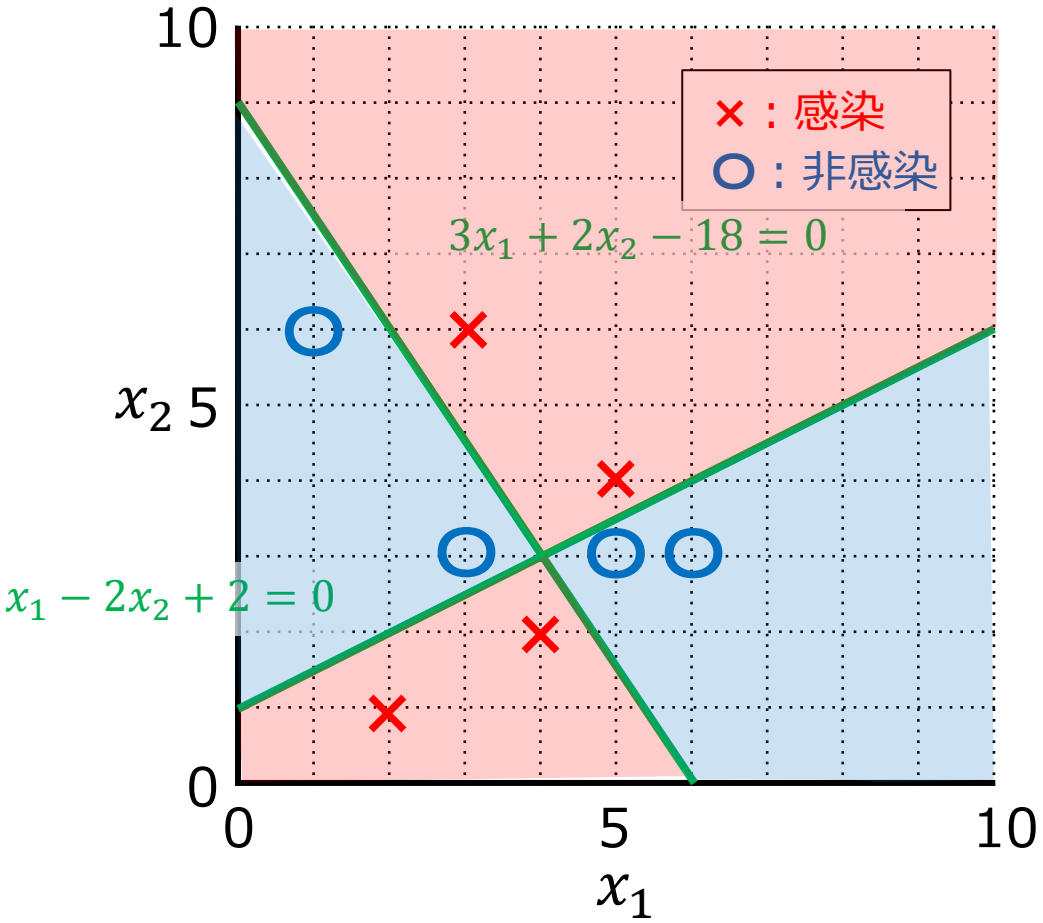
- 2本の直線を引くと、分類できる。

$$x_2 = -\frac{3}{2}x_1 + 9 \quad ☞ \quad 3x_1 + 2x_2 - 18 = 0$$

$$x_2 = \frac{1}{2}x_1 + 1 \quad ☞ \quad x_1 - 2x_2 + 2 = 0$$

簡単な分類　課題2の解答例

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ 2本の直線を引くと、分類できる。

$$x_2 = -\frac{3}{2}x_1 + 9 \quad ☞ \quad 3x_1 + 2x_2 - 18 = 0$$

$$x_2 = \frac{1}{2}x_1 + 1 \quad ☞ \quad x_1 - 2x_2 + 2 = 0$$

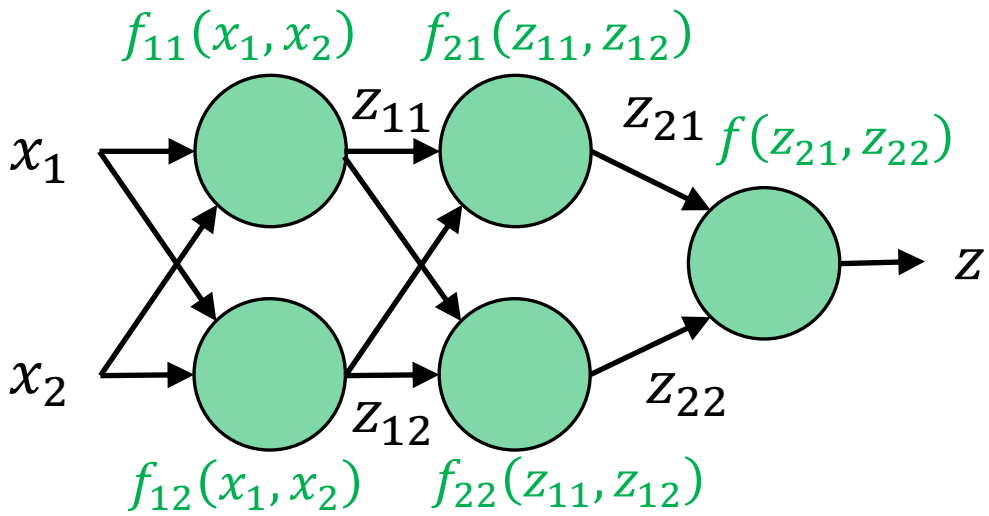
◆ 分類条件

- $3x_1 + 2x_2 - 18 > 0$ かつ $x_1 - 2x_2 + 2 \leqq 0$ の時：感染
- $3x_1 + 2x_2 - 18 > 0$ かつ $x_1 - 2x_2 + 2 > 0$ の時：非感染
- $3x_1 + 2x_2 - 18 \leqq 0$ かつ $x_1 - 2x_2 + 2 > 0$ の時：非感染
- $3x_1 + 2x_2 - 18 \leqq 0$ かつ $x_1 - 2x_2 + 2 \leqq 0$ の時：感染

簡単な分類　課題2の解答例

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。
  - ニューラルネットワークの例

- $f_{11}(x_1, x_2) = 3x_1 + 2x_2 - 18$
- $f_{12}(x_1, x_2) = x_1 - 2x_2 + 2$
- $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1$
- $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1$
- $f(z_{21}, z_{22}) = z_{21} + z_{22}$

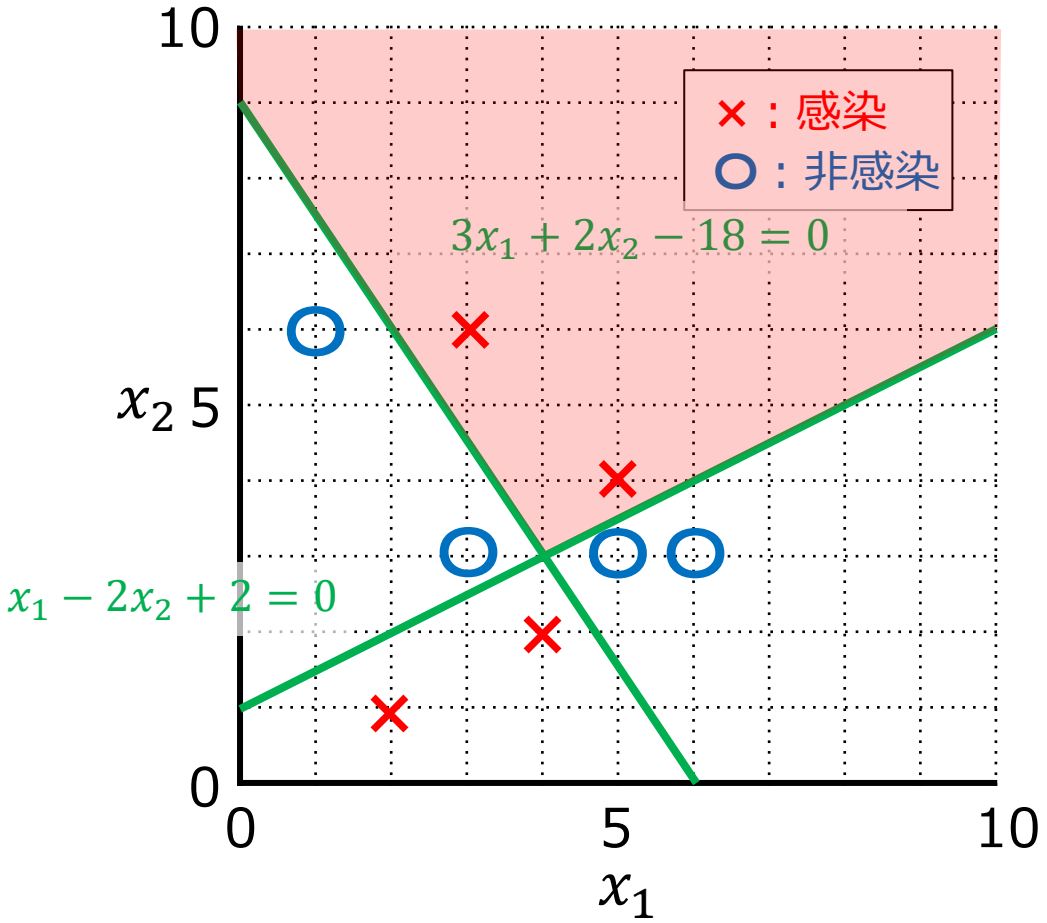


$f(z_{21}, z_{22}) > 0$の時$z = 1$
$f(z_{21}, z_{22}) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

簡単な分類　課題2の解答例

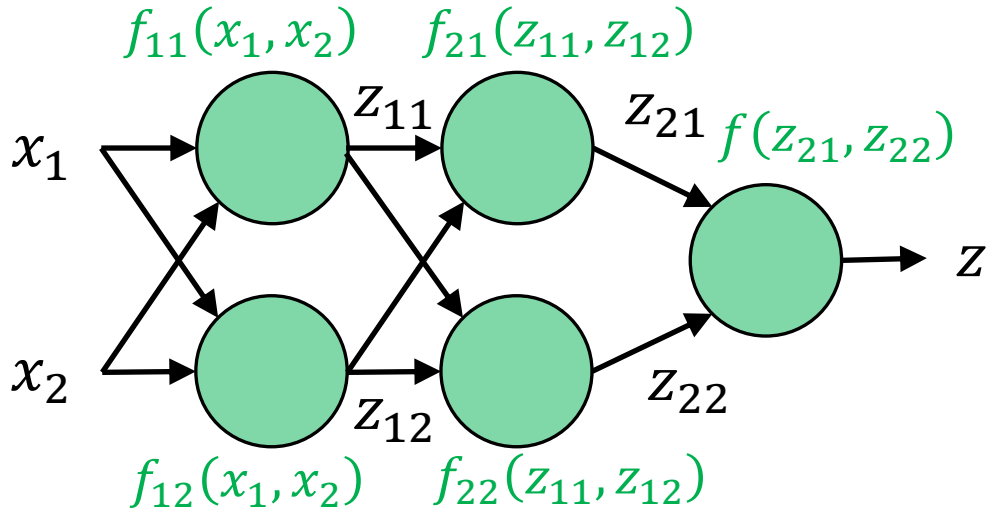
- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_{11} = 1$ かつ $z_{12} = -1$ の時

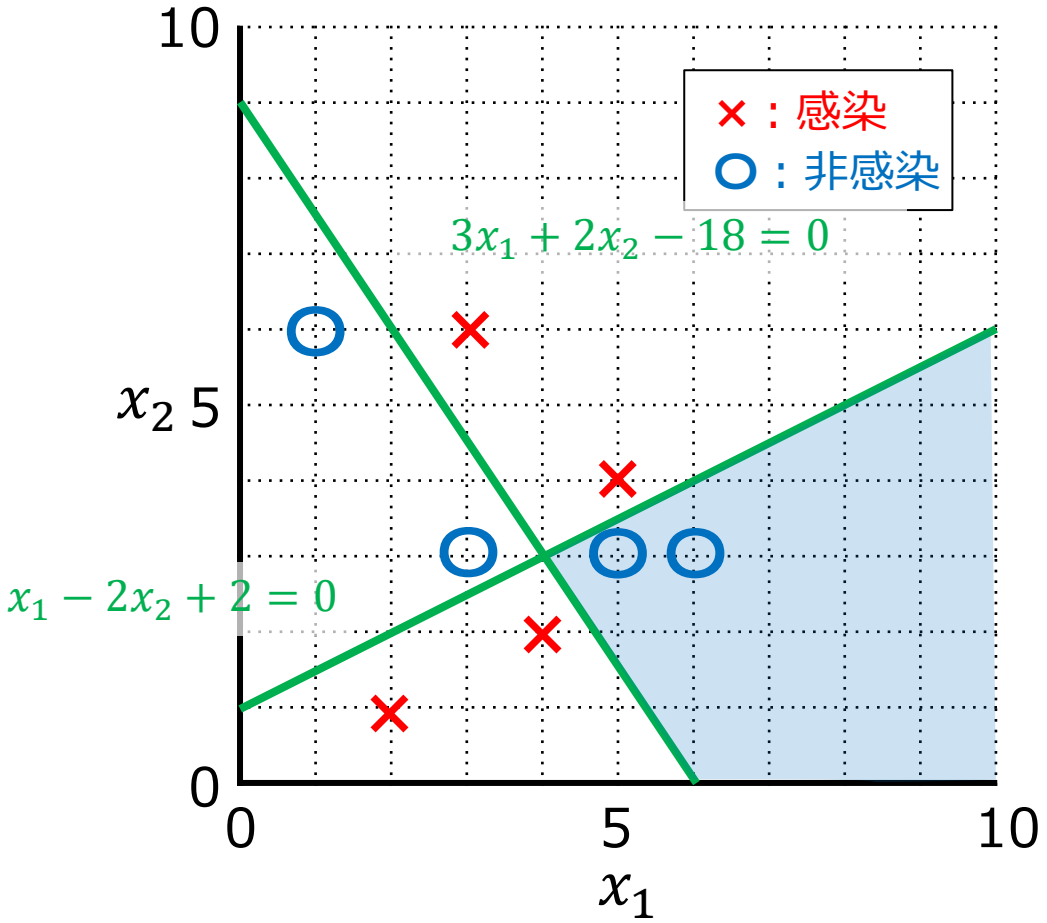
- $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1 = 1 > 0 \rightarrow z_{21} = 1$
- $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1 = 1 > 0 \rightarrow z_{22} = 1$
- $f(z_{21}, z_{22}) = z_{21} + z_{22} = 2 > 0 \rightarrow z = 1$

☞**感染**



簡単な分類　課題2の解答例

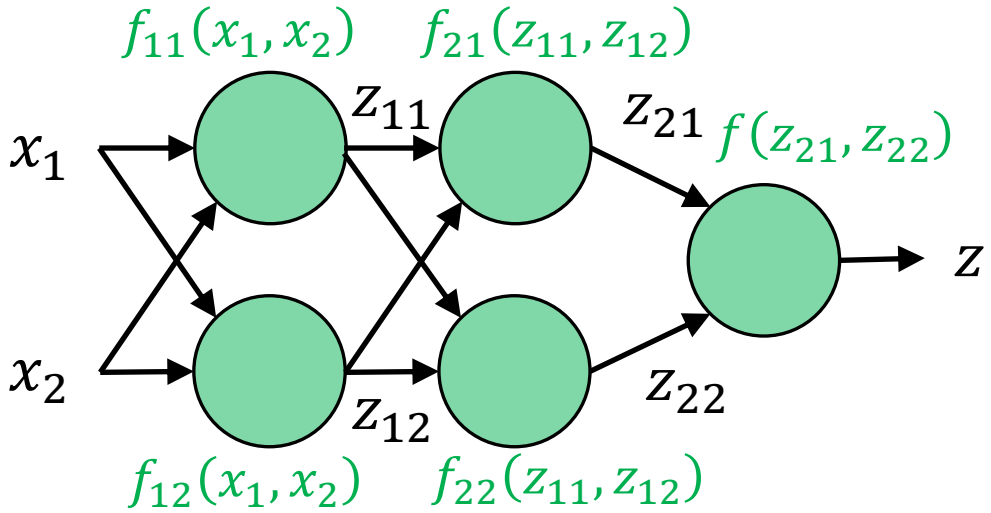
- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_{11} = 1$ かつ $z_{12} = 1$ の時

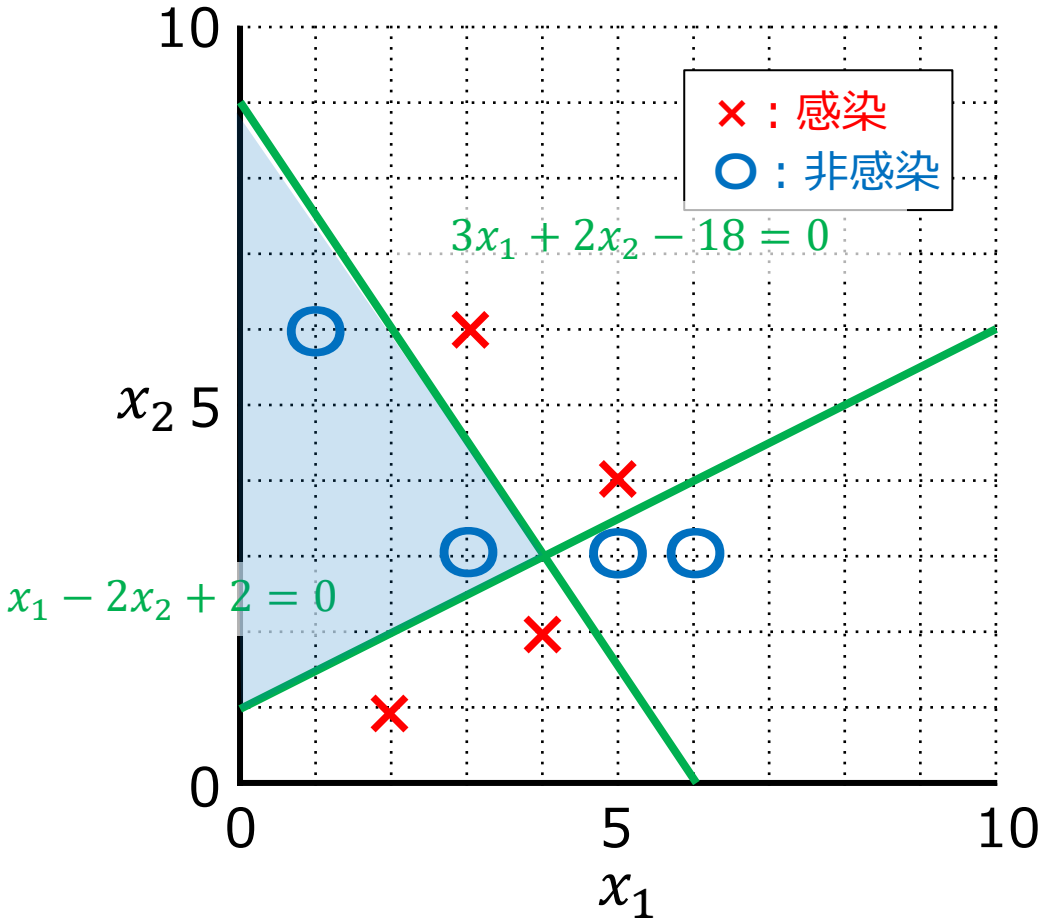
- $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1 = 3 > 0 \rightarrow z_{21} = 1$
- $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1 = -1 \leqq 0 \rightarrow z_{22} = -1$
- $f(z_{21}, z_{22}) = z_{21} + z_{22} = 0 \leqq 0 \rightarrow z = -1$

☞**非感染**



簡単な分類　課題2の解答例

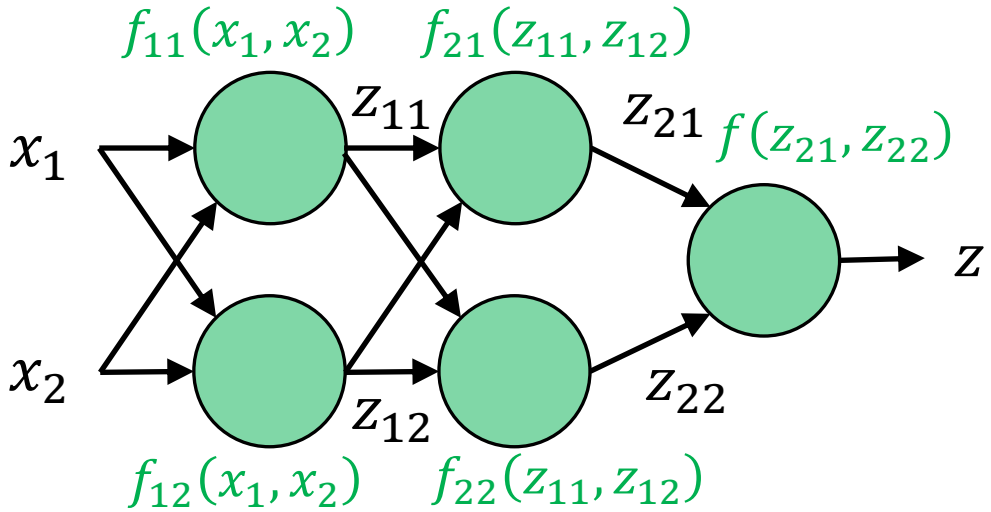
- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_{11} = -1$ かつ $z_{12} = -1$ の時

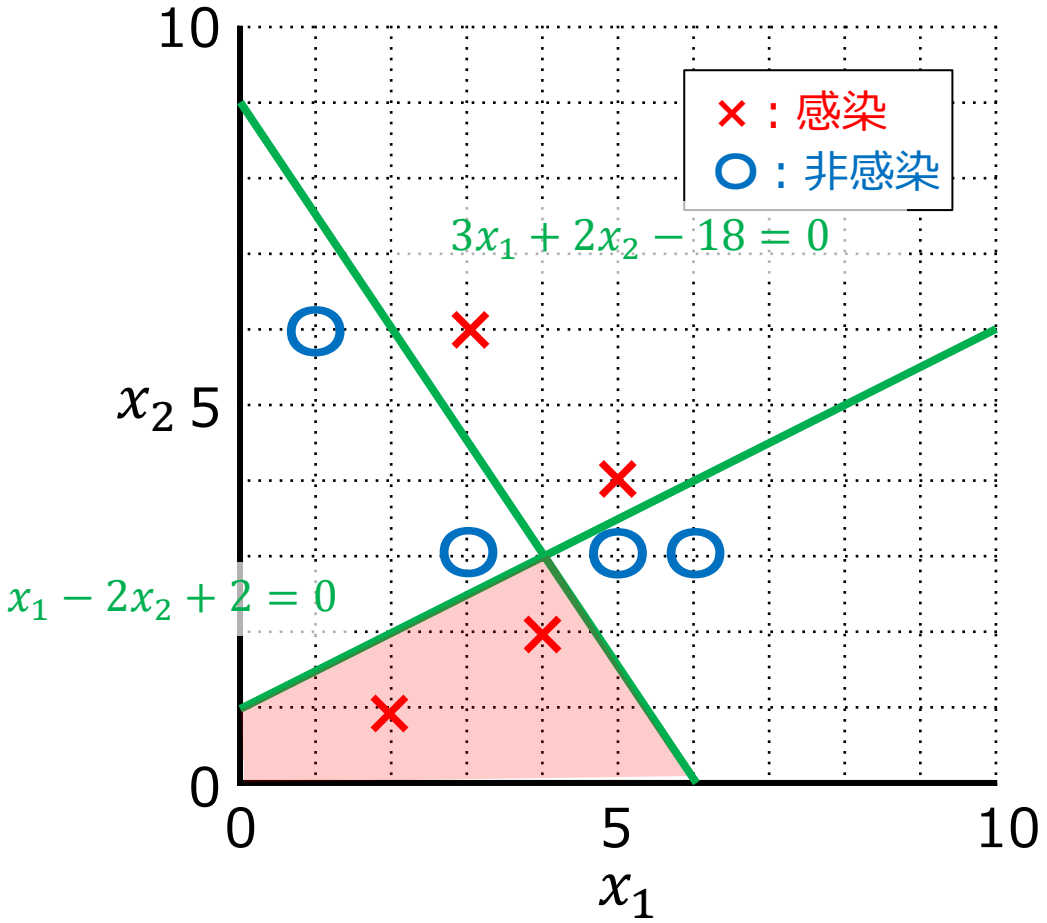
- $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1 = -1 \leqq 0 \rightarrow z_{21} = -1$
- $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1 = 3 > 0 \rightarrow z_{22} = 1$
- $f(z_{21}, z_{22}) = z_{21} + z_{22} = 0 \leqq 0 \rightarrow z = -1$

☞**非感染**



簡単な分類　課題2の解答例

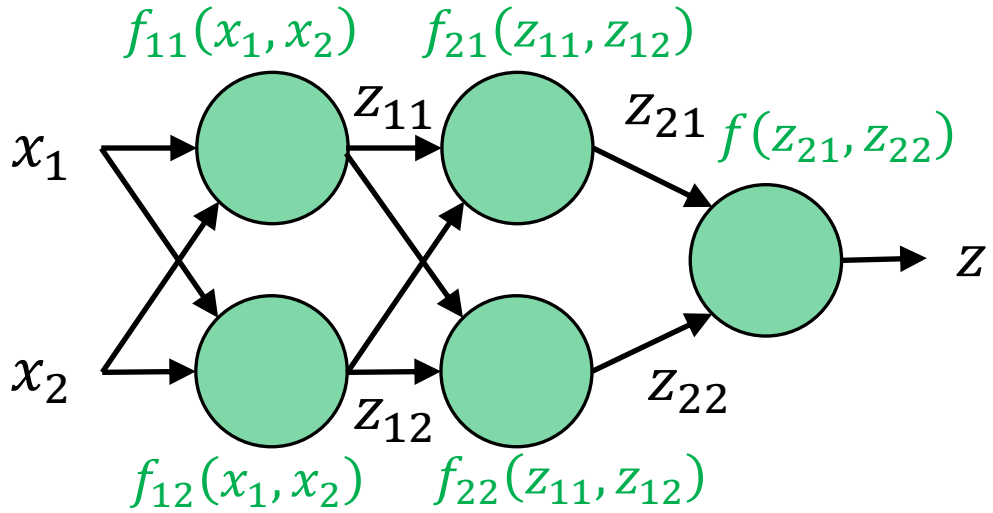
- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。



◆ $z_{11} = -1$ かつ $z_{12} = 1$ の時

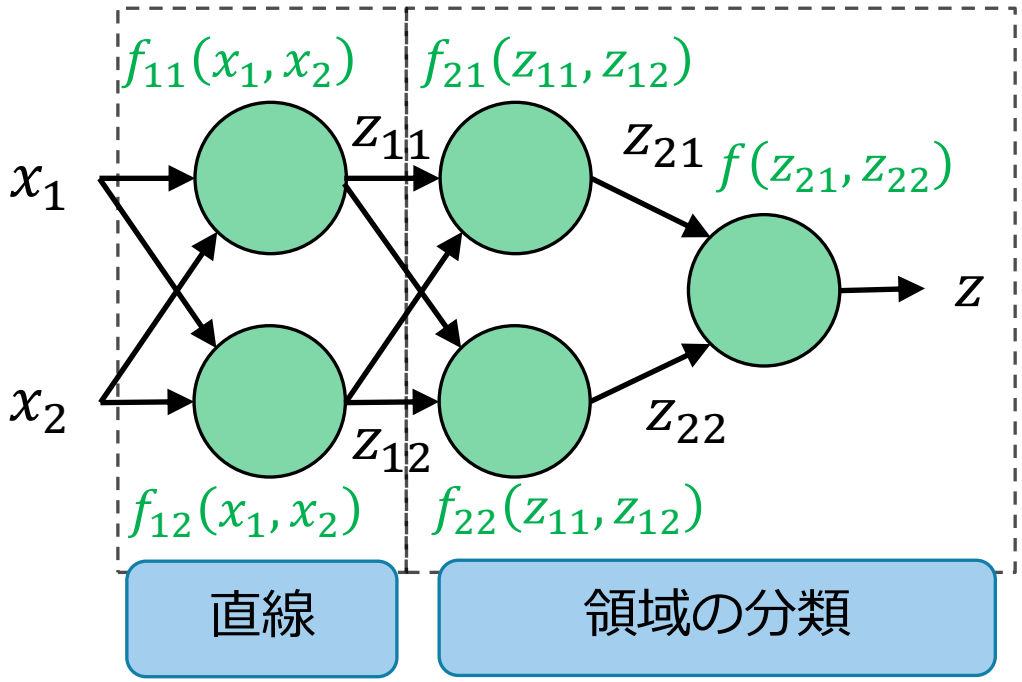
- $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1 = 1 > 0 \rightarrow z_{21} = 1$
- $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1 = 1 > 0 \rightarrow z_{22} = 1$
- $f(z_{21}, z_{22}) = z_{21} + z_{22} = 2 > 0 \rightarrow z = 1$

☞**感染**



簡単な分類　課題2の解答例

- 問題2：ウイルスに感染しているかどうか、2種類のデータ(x_1, x_2)から知りたい。
  - ニューラルネットワークの例
    - $f_{11}(x_1, x_2) = 3x_1 + 2x_2 - 18$
    - $f_{12}(x_1, x_2) = x_1 - 2x_2 + 2$
    - $f_{21}(z_{11}, z_{12}) = z_{11} + z_{12} + 1$
    - $f_{22}(z_{11}, z_{12}) = -z_{11} - z_{12} + 1$
    - $f(z_{21}, z_{22}) = z_{21} + z_{22}$



$f(z_{21}, z_{22}) > 0$の時$z = 1$
$f(z_{21}, z_{22}) \leqq 0$の時$z = -1$

$z = 1$なら、感染
$z = -1$なら、非感染

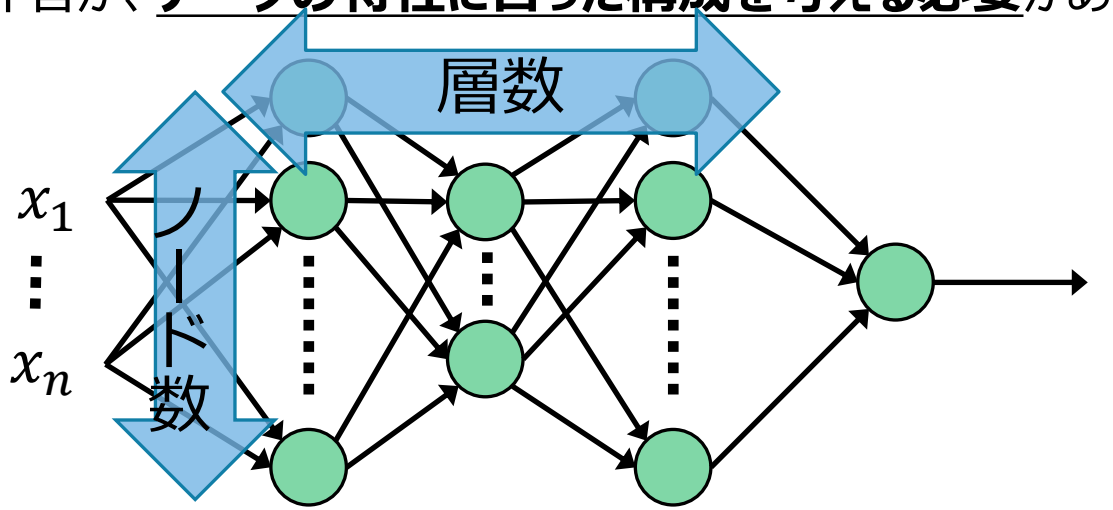
ニューラルネットワークの概要（レポート：10/100）

- 課題1，2を通して分かったこと
  - ニューラルネットワークのノード・層を増やすことで様々な分類ができる。

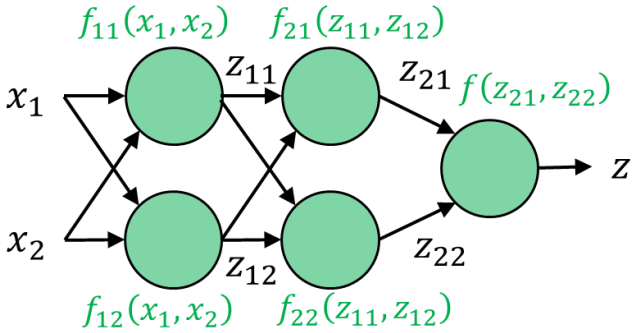
    ☞数学的にはノードを増やすことで、**どれだけ複雑な分類もできる**ことが知られている。

- AI（人工知能）と機械学習の仕組み
  - 傾きや切片などの**重みパラメータをコンピュータが自動で学習**してくれる仕組み。
  - ニューロン数（ノード・層）を**増やしすぎると計算しきれない**。

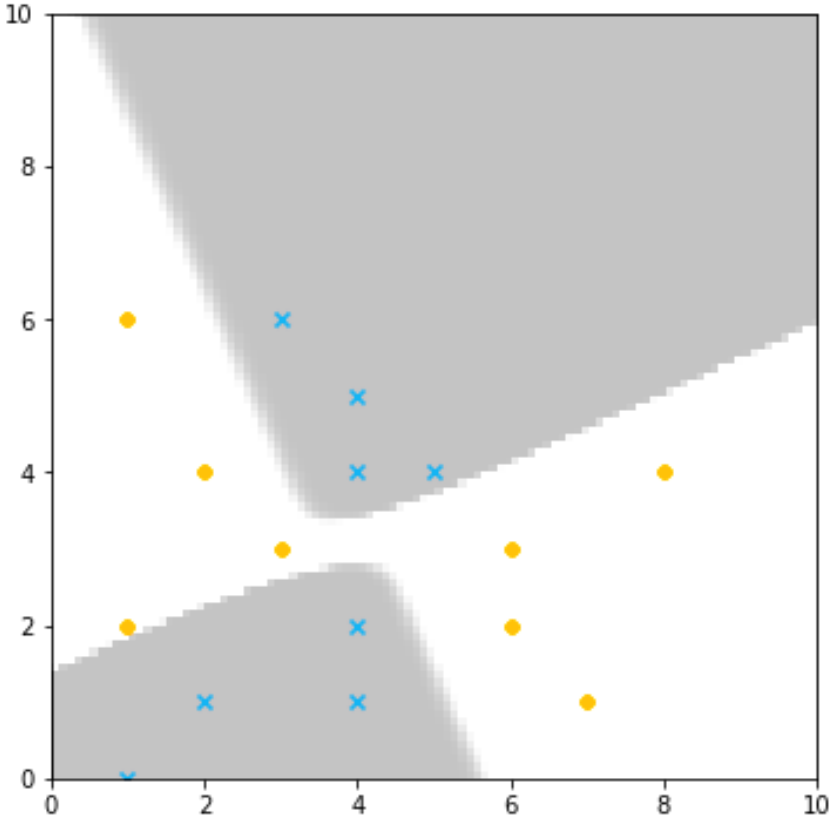
    ☞ネットワークの設計者が、**データの特性に合った構成を考える必要**がある。➡**ディープラーニング**



おまけ

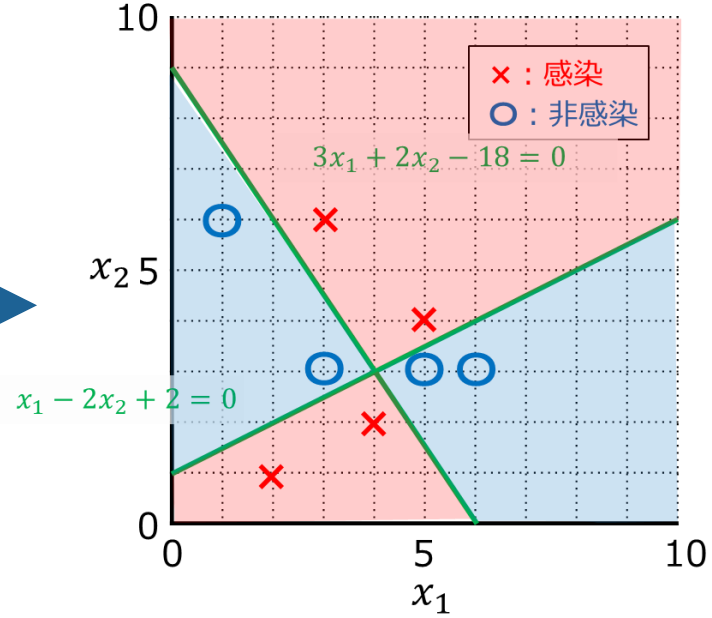


- 実際にニューラルネットワークでパラメータ（傾きと切片）を学習した結果



コンピュータが計算した結果

- 青×が感染、オレンジ〇が非感染
- 網掛けされた部分が感染予想



人間が計算した結果

機械学習の３ステップ（レポート：10/100）

① 与えられたデータを元にして、**未知のデータを予測する数式**を考える。
（今回の例）ニューラルネットワークの構成（ニューロンの層数・ノード数）を考える。

② 数式に含まれる**パラメータの良し悪しを判断する誤差関数**を用意する。
（今回の例）ニューラルネットワークの出力と感染・非感染の誤差を減らしたい。

③ 誤差関数を最小にするように**パラメータの値を決定**する。
（今回の例）重みパラメータ（傾き、切片）を更新し、分類精度を上げていく。

機械学習の３ステップ（レポート：10/100）

① 与えられたデータを元にして、**未知のデータを予測する数式**を考える。←人
（今回の例）ニューラルネットワークの構成（ニューロンの層数・ノード数）を考える。

② 数式に含まれる**パラメータの良し悪しを判断する誤差関数**を用意する。←人
（今回の例）ニューラルネットワークの出力と感染・非感染の誤差を減らしたい。

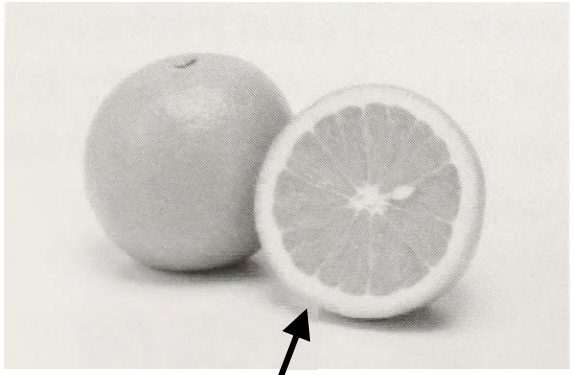
③ 誤差関数を最小にするように**パラメータの値を決定**する。←コンピュータ
（今回の例）重みパラメータ（傾き、切片）を更新し、分類精度を上げていく。

CNNの概要

- ディープラーニングとは？
  - 何らかの意図をもって組み上げられたニューラルネットワーク。
    ☞CNNは画像認識に有効なディープラーニングの手法。
- CNNとは？
  - Convolutional Neural Network（畳み込みニューラルネットワーク）の略。
  - 特徴抽出に「畳み込みフィルタ」を用いたニューラルネットワーク。
  - 「畳み込みフィルタ」は画像処理ソフトウェアとして利用されている機能（ディープラーニングに限らない）
    ☞画像を強調したり、ぼかしたり、色々な加工ができる。

畳み込みフィルター

▶画像をぼかすフィルター



$$91 \times 0.11 + 46 \times 0.11 + 35 \times 0.11$$
$$+141 \times 0.11 + 140 \times 0.12 + 135 \times 0.11$$
$$+156 \times 0.11 + 153 \times 0.11 + 153 \times 0.11 = 116.9$$

91	46	35
141	140	135
156	153	153

画像のピクセル値

×

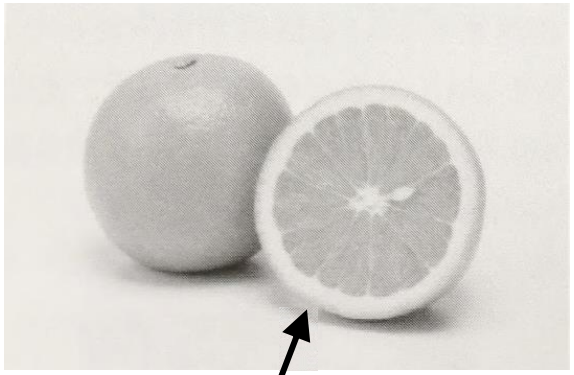
0.11	0.11	0.11
0.11	0.12	0.11
0.11	0.11	0.11

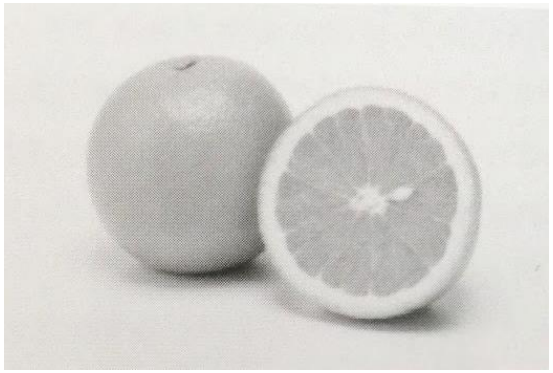
畳み込みフィルター

117

畳み込みフィルター

▶画像をぼかすフィルター





91	46	35
141	140	135
156	153	153

画像のピクセル値

×

0.11	0.11	0.11
0.11	0.12	0.11
0.11	0.11	0.11

畳み込みフィルター

117

畳み込みフィルター

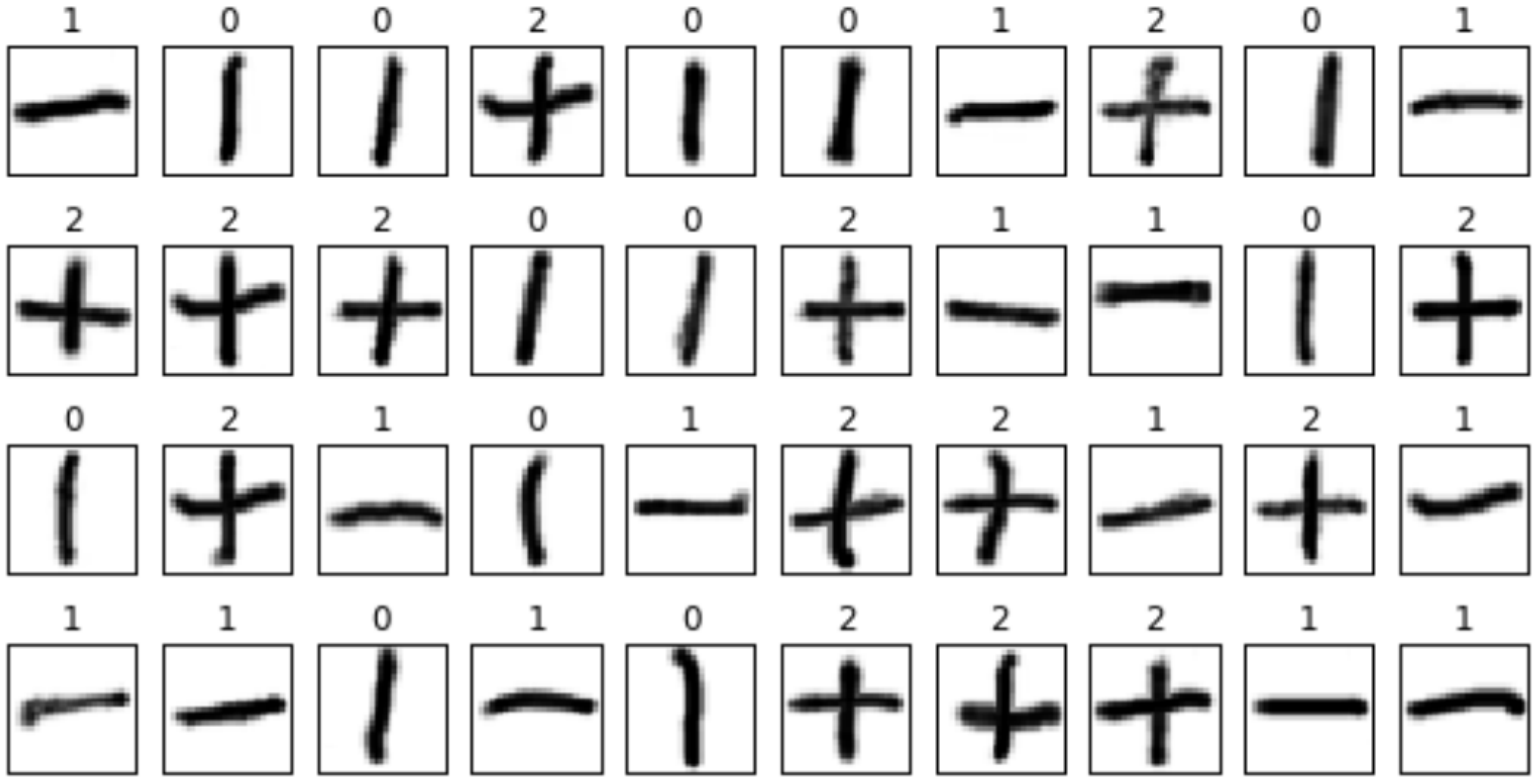
▶縦のエッジを抽出するフィルター

-1	0	1
-2	0	2
-1	0	1

畳み込みフィルター

簡単な画像に畳み込みフィルターを適応

▶縦、横線からなるグレースケール画像



簡単な画像に畳み込みフィルターを適応

▶縦のエッジと横のエッジを抽出するフィルター

2	1	0	-1	-2
3	2	0	-2	-3
4	3	0	-3	-4
3	2	0	-2	-3
2	1	0	-1	-2

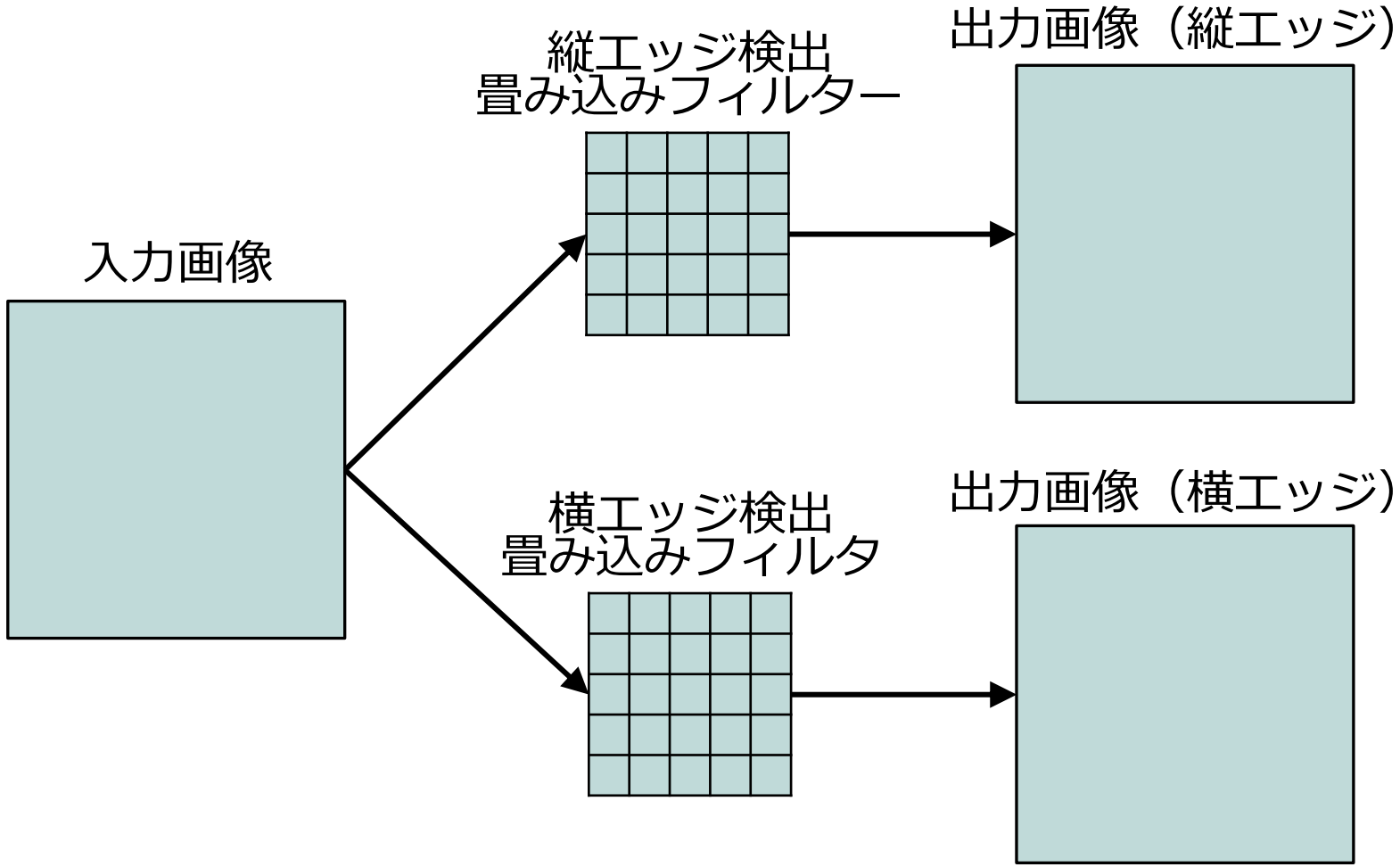
縦エッジ検出用
畳み込みフィルター

2	3	4	3	2
1	2	3	2	1
0	0	0	0	0
-1	-2	-3	-2	-1
-2	-3	-4	-3	-2

横エッジ検出用
畳み込みフィルター

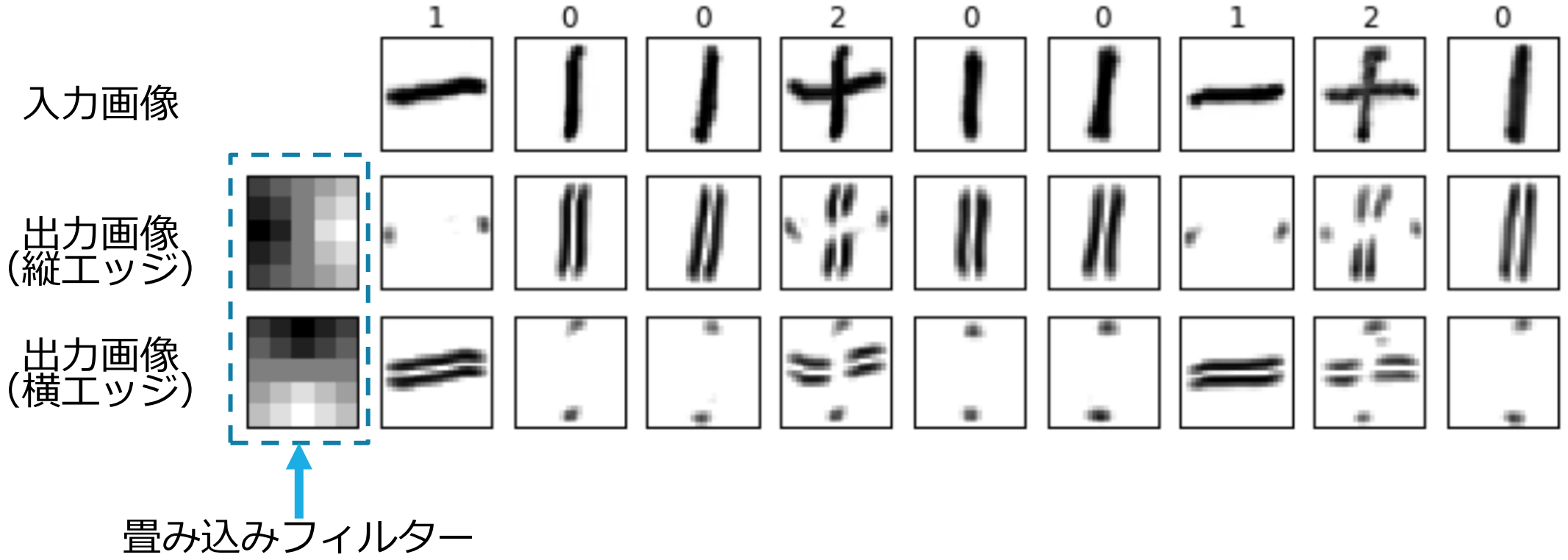
簡単な画像に畳み込みフィルターを適応

▶畳み込みフィルターの適応



簡単な画像に畳み込みフィルターを適応

▶入力画像と出力画像



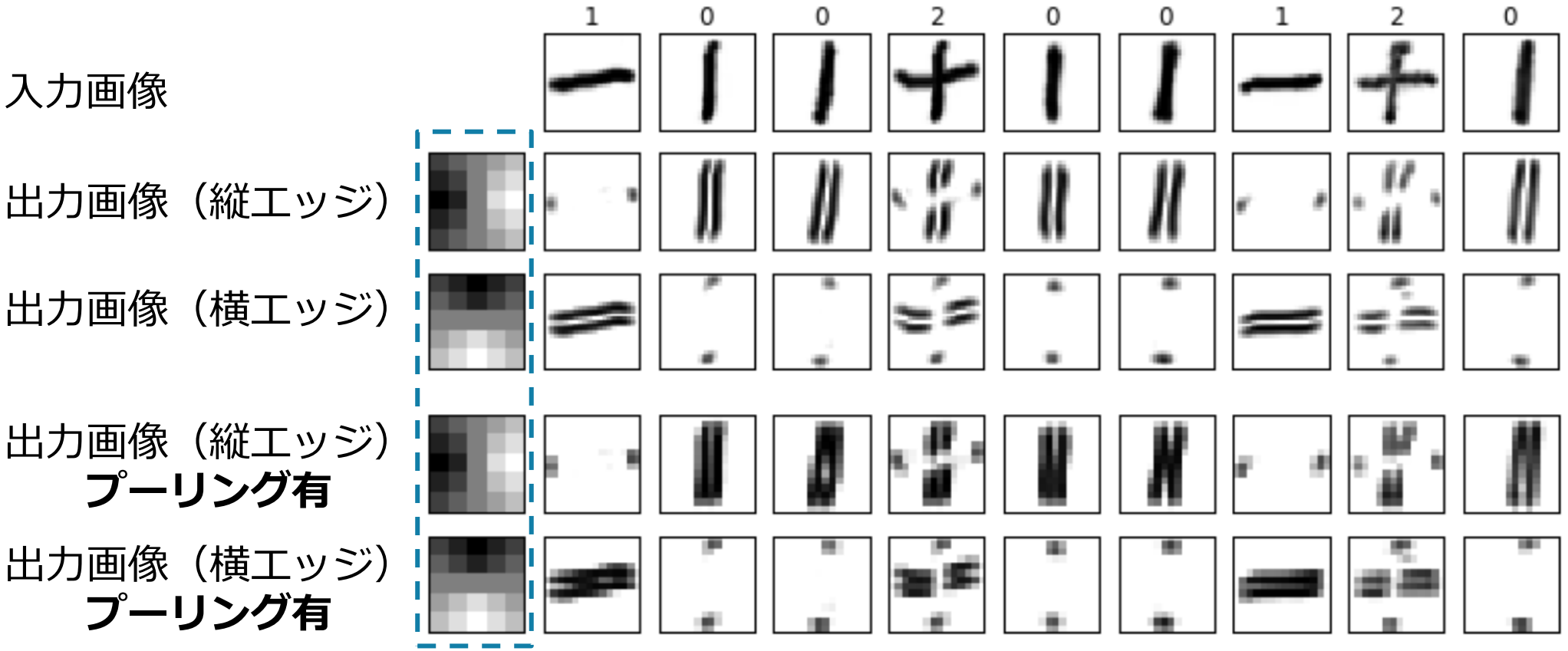
簡単な画像に畳み込みフィルターを適応

▶プーリング処理

- 出力結果の詳細が関係ない
  - ➢ 解像度を落として問題ない

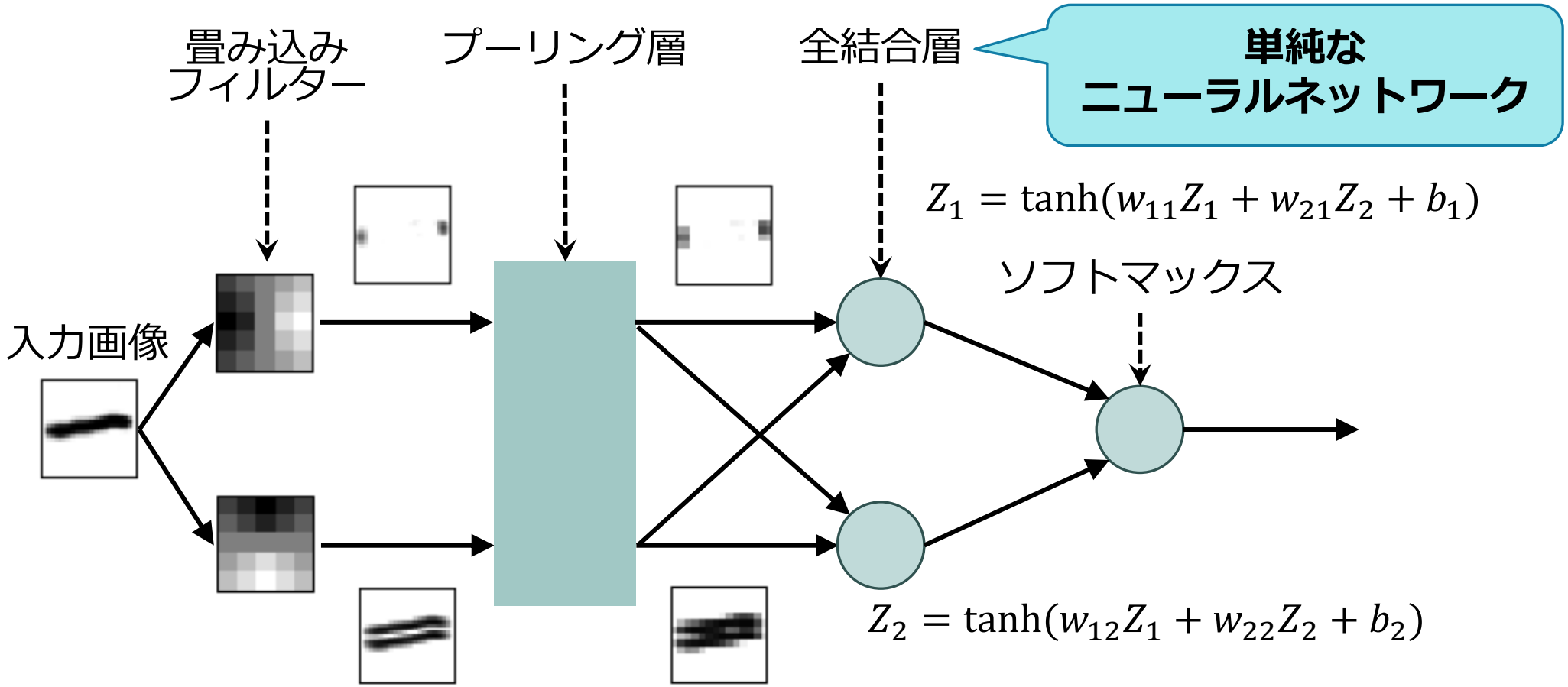
簡単な画像に畳み込みフィルターを適応

▶プーリング処理



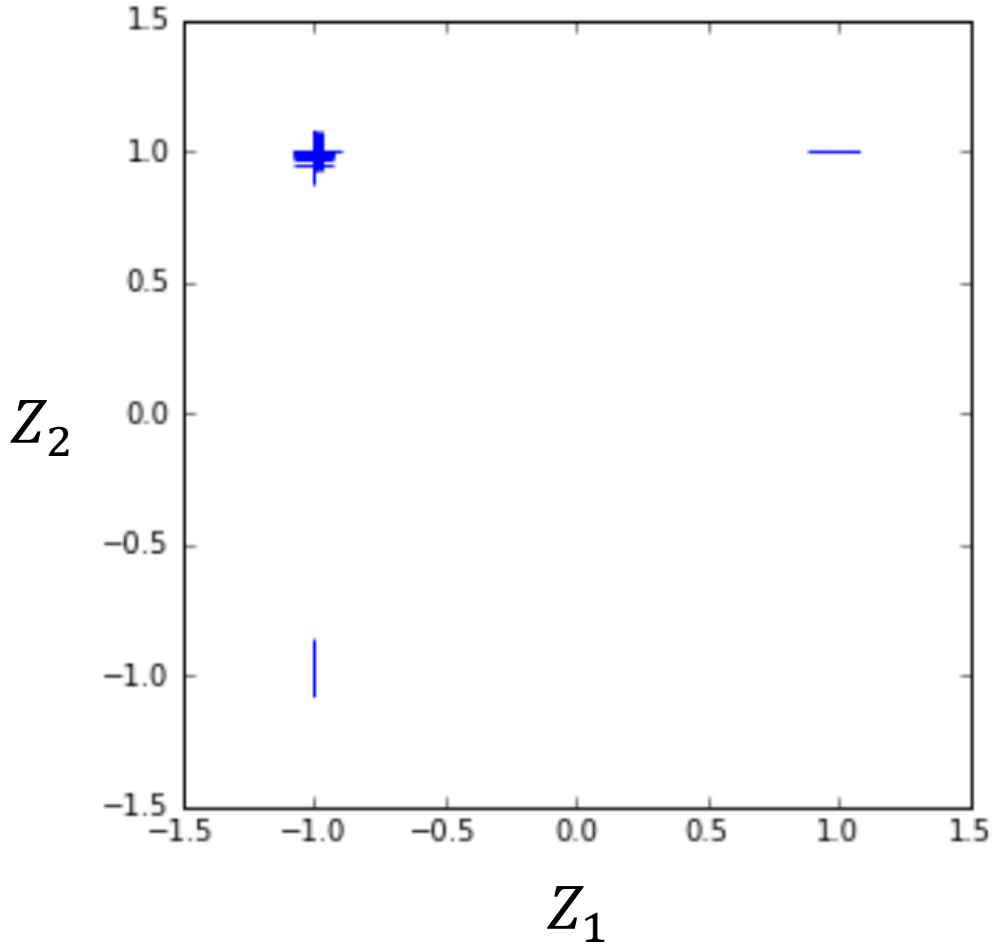
畳み込みフィルターを用いた簡単な画像分類

▶画像データを特徴変数に変換するニューラルネットワーク



畳み込みフィルターを用いた簡単な画像分類

▶学習結果

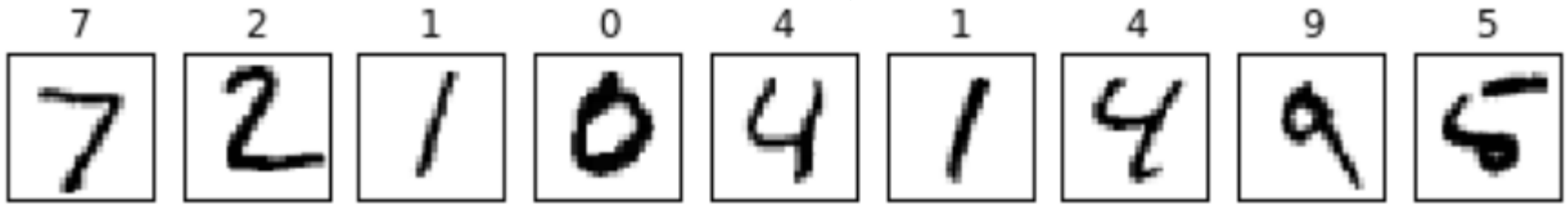


畳み込みフィルターの動的な学習

▶動的な学習

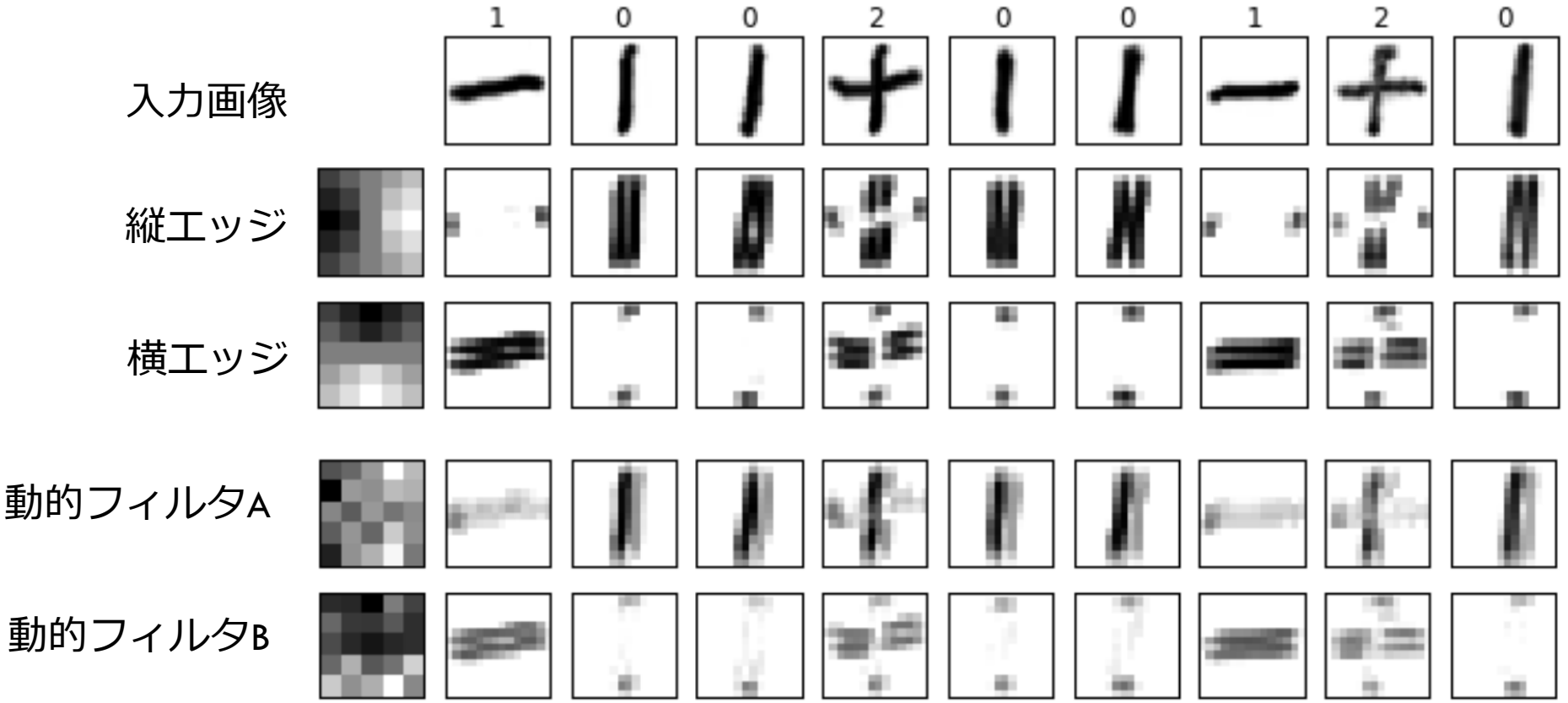
- 特徴抽出に有効な畳み込みフィルターを見つけるのが難しい
  - ➢ フィルタの最適化

「縦エッジ」と「横エッジ」だけでは
特徴抽出できなさそう



畳み込みフィルターの動的な学習

▶動的な学習

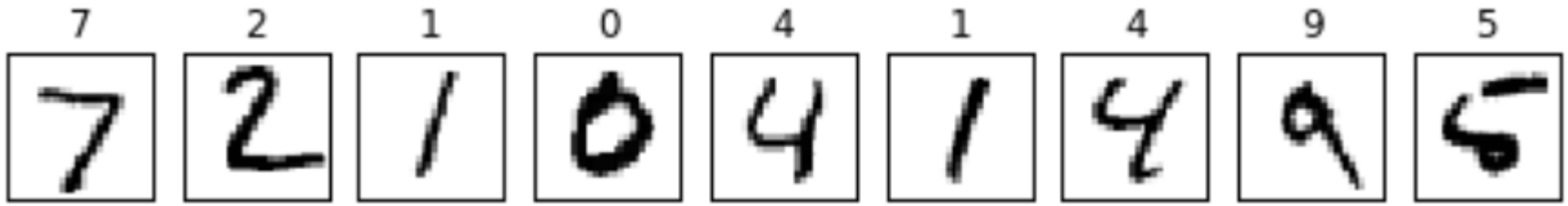


畳み込みフィルターを用いた手書き文字の分類

▶MNISTとは?

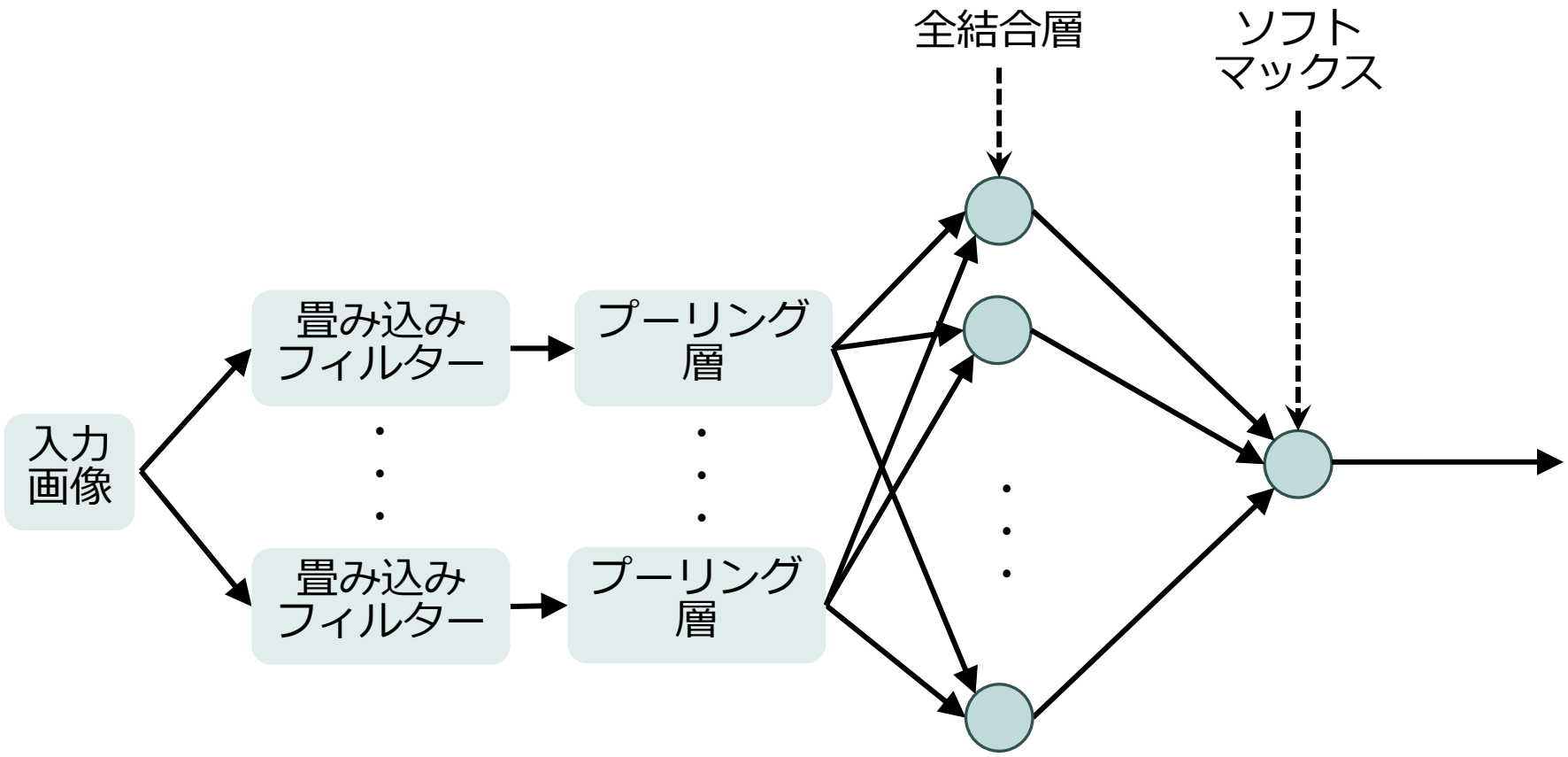
- MNISTは手書き文字画像のデータセット
- 学習データ6万とテストデータ１万が用意されている

(http://yann.lecun.com/exdb/mnist/)



畳み込みフィルターを用いた手書き文字の分類

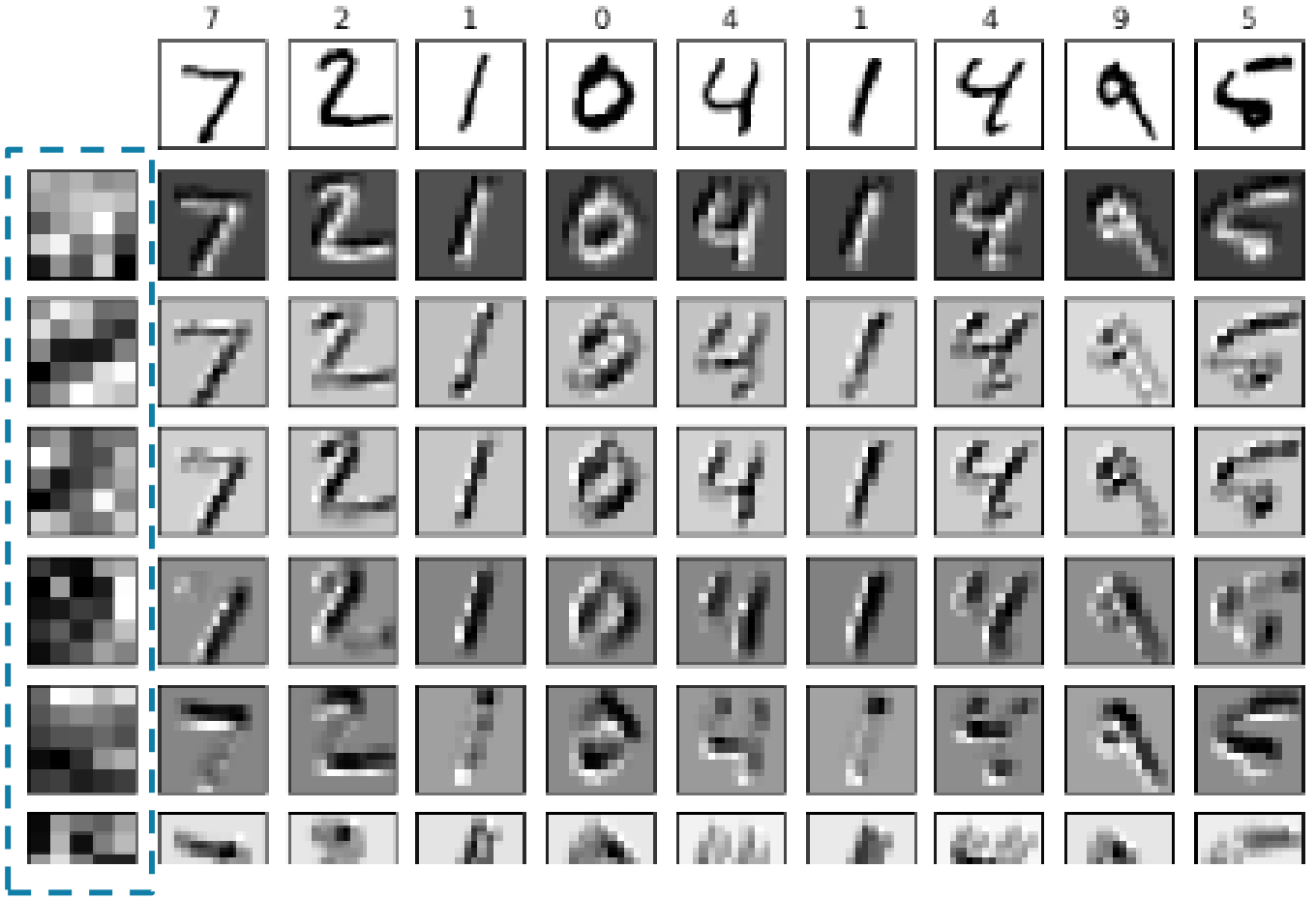
▶手書き文字分類用のニューラルネットワーク



畳み込みフィルターを用いた手書き文字の分類

▶学習結果

- 正答率: 98%



動的な畳み込みフィルター

AI・ディープラーニングのまとめ（レポート：10/100）

- ニューラルネットワークやAIは万能ではなく、**目的に応じて**人間が考えて**設計していく必要**がある。

- ディープラーニングは、何らかの意図をもって組み上げられたニューラルネットワークであり、**CNNは画像認識に有効なディープラーニングの手法**である。

- 活性化関数（Sigmoid関数、ReLU関数など）や誤差関数（最小二乗誤差など）も重要な手法なので、興味がある人は調べてみてください。

【AIあり仕分け】の説明（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

□手順

1. カメラ座標系とロボット座標系の変換行列の作成
2. 学習用画像の作成
3. 画像データを使用した教師あり学習
4. 学習モデルを使ったチロルチョコの仕分け

【AIあり仕分け】の説明（15分）

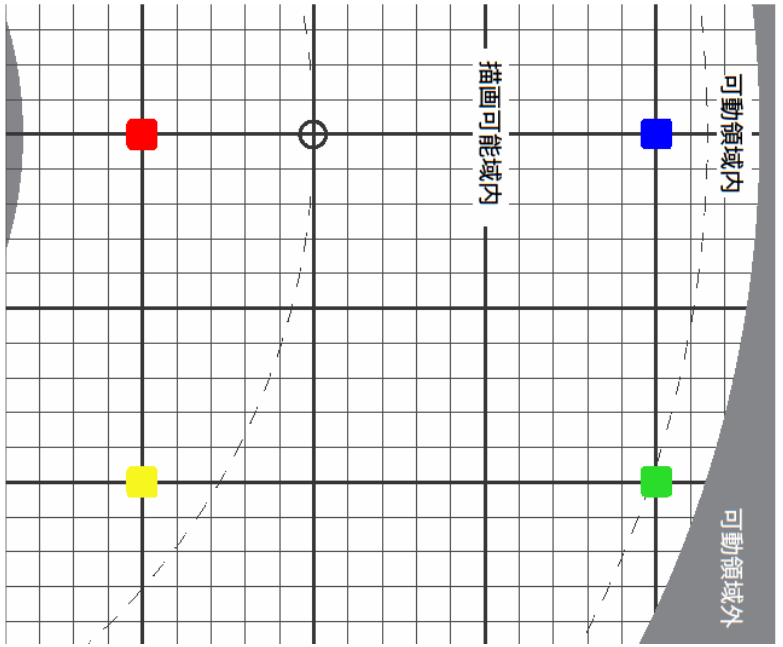
◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

1. カメラ座標系とロボット座標系の変換行列の作成

   各マーカーのDOBOTで制御する際の座標（ロボット座標系）4点と
   WEBカメラで見た時の座標（カメラ座標系）の4点を
   透視変換（ホモグラフィー変換）し、変換行列を作成

☞**WEBカメラの座標と**
DOBOTの座標を
対応付ける

測定マット上のマーカー
赤(150, 0)
青(300, 0)
黄(150, -100)
緑(300, -100)



【AIあり仕分け】の説明（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

2. 学習用画像の作成

学習には、ある程度の数のデータを準備する必要がある。

☞**学習データの水増し**

元画像

コントラスト調整
（高コントラスト）

ガンマ値調整
（$\gamma = 0.5$）

ガウス分布に
基づくノイズ処理

ゴマ塩ノイズ処理

【AIあり仕分け】の説明（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

3. 画像データを使用した教師あり学習

<u>畳み込みニューラルネットワーク(CNN)を用いたディープラーニング(DNN)</u>

●教師あり学習：予め用意した答え（教師信号）の分かっているデータ（学習データ）を与えて、その特徴や規則を記憶する

- 学習データ→チロルチョコのパッケージの画像
- 教師信号　→チロルチョコの種類

●学習後に、学習に使用していないデータ（テストデータ）で、正しく分類できるか検証する

【AIあり仕分け】の説明（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

4. 学習モデルを使ったチロルチョコの仕分け

DOBOTを実際に動かし、カメラ画像からチロルチョコの仕分けを行う。

【AIあり仕分け】の説明（15分）

opencv_setting.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 領域抽出用の最小／最大サイズ
MIN_AREA_SIZE = 200
MAX_AREA_SIZE = 600
```

WEBカメラの位置によって変わる

※　カメラ位置（DOBOTとカメラの位置関係）が変わるたびに、実行してください。

【AIあり仕分け】の説明（15分）

train.pyの一部抜粋

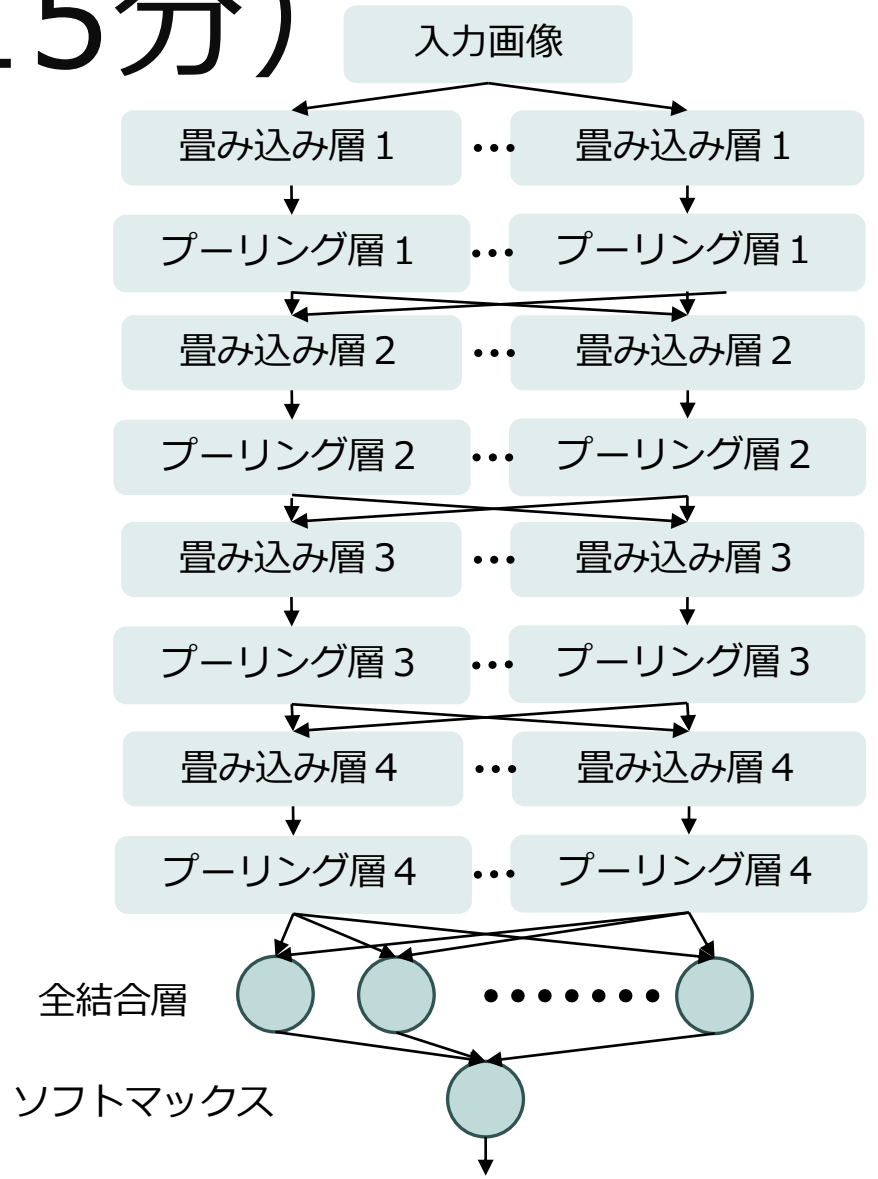
```
model.add(Conv2D(32, (3, 3), input_shape=input_shape, padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(32, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding=("same")))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))

model.add(Dense(output_classes))
model.add(Activation("softmax"))
```



【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
import os
import sys, time
from datetime import datetime
from argparse import ArgumentParser

from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing.image import img_to_array, load_img
import cv2
import numpy as np

import cameraSetting as camset
import dobotClassifier as dc

from common import *
from TransformationMatrix import MATRIX
MATRIX = np.array(MATRIX)
```

ライブラリのインポート

「tensorflow」「keras」は機械学習関連

「dobotClassifier.py」DOBOT関連の関数定義

「common.py」その他の定義

カメラ座標とロボットアーム座標の変換行列

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 学習済みモデルを使って仕分ける（3）
if __name__ == '__main__‘:
  ～省略（初期設定）～
    while True:
        # VideoCaptureから1フレーム読み込む（3-3）
        ret, frame = cap.read()
        ret, edframe = cap.read()
        # 加工なし画像を表示する
        cv2.imshow('Raw Frame', frame)

        # グレースケールに変換（3-4）
        gray = cv2.cvtColor(edframe, cv2.COLOR_BGR2GRAY)
        #  2値化（3-5）
        retval, bw = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        # 輪郭を抽出（3-6）
        contours, hierarchy = cv2.findContours(bw, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

画像加工

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
cutframe_array = []
result_array   = []
image_list     = []

# 各輪郭に対する処理
for i, contour in enumerate(contours):
    # ノイズを除去する（3-7）
    # 輪郭の領域を計算
    area = cv2.contourArea(contour)
    # ノイズ（小さすぎる領域）と全体の輪郭（大きすぎる領域）を除外
    if area < MIN_AREA_SIZE or MAX_AREA_SIZE < area:
        continue
```

予め空リストを作成

輪郭（チロルチョコ）の数だけ繰り返す

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# フレーム画像から対象物を切り出す（3-8）
# 回転を考慮した外接矩形を取得する
rect = cv2.minAreaRect(contour)
box = cv2.boxPoints(rect)
box = np.int0(box)
center, size, angle = rect
center = tuple(map(int, center))  # float -> int
size   = tuple(map(int, size))    # float -> int

# 回転行列を取得する
rot_mat = cv2.getRotationMatrix2D(center, angle, 1.0)
h, w = frame.shape[:2]

# 切り出す
rotated = cv2.warpAffine(frame, rot_mat, (w, h))
cropped = cv2.getRectSubPix(rotated, size, center)
img_src = rect_preprocess(cropped)
```

「center」「size」「angle」はそれぞれ
チロルチョコの「中心座標」「幅と高さ」「角度」

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# リサイズする
image = cv2.resize(img_src, dsize=(PIC_SIZE, PIC_SIZE))
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = np.asarray(image)
image_list.append(image)

# 画像の前処理（3-9）
image = img_to_array(image)
image = image.astype("float32") / 255.
image = image[None, ...]

# 分類する（3-10）
result = model.predict_classes(image)
proba  = model.predict_proba(image)
result_num = int(result[0])
```

学習済みのニューラルネットワークに処理した画像を入力

分類結果

分類の確率

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 輪郭に外接する長方形を取得する。（3-11）
x, y, width, height = cv2.boundingRect(contour)

# 長方形を描画する（3-12）
cv2.rectangle(edframe, (x, y), (x+width, y+height), draw_white, thickness=1)
# ラベルを表示する（3-13）
label = str(result_num) + " " + LabelName[result_num]
cv2.rectangle(edframe, (x, y-15), (x+len(label)*10, y), draw_white, -1, cv2.LINE_AA)
cv2.putText(edframe, label, (x, y-4), font, FONT_SIZE, draw_black, FONT_WIDTH, cv2.LINE_AA)
```

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 「P」キーが押されたときの処理（3-14）
if proba_flag == True:
    # 回転を考慮した外接矩形を表示
    cv2.drawContours(edframe, [box], 0, draw_red, 1)

    # 確率を表示
    for n in LabelName:
        cnt = getattr(LabelNumber, n)
        proba_str = "[{:<6}] {:>5.2f}%".format(n, proba[0, cnt] * 100.)
        if cnt == np.argmax(proba):
            cv2.putText(edframe, proba_str, (x+width+5, y+30+(20*cnt)), font,
                        FONT_SIZE, draw_red, FONT_WIDTH, cv2.LINE_AA)
        else:
            cv2.putText(edframe, proba_str, (x+width+5, y+30+(20*cnt)), font,
                        FONT_SIZE, draw_white, FONT_WIDTH, cv2.LINE_AA)
```

「p」キーが押されたとき
分類確率を画像上に表示

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
        # 外接矩形の中心点を描画（3-15）
        mp_x = int(center[0])
        mp_y = int(center[1])
        cv2.drawMarker(edframe, (mp_x, mp_y), draw_green,
            cv2.MARKER_TILTED_CROSS, thickness = 2)

        # 中心点の座標をカメラ座標系からロボット座標系へ変換（3-16）
        transform_pos = transform_coordinate(mp_x, mp_y)
        lavel = "DOBOT : " + str(transform_pos)
        cv2.putText(edframe, lavel, (x+width+5, y+10), font, FONT_SIZE,
            draw_green, FONT_WIDTH, cv2.LINE_AA)

    # 描画した画像を表示
    cv2.imshow('Edited Frame', edframe)
```

チロルチョコの(x, y)の
ロボット座標位置を決定

画像上に、分類結果や
座標位置等を表示

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# キー入力を1ms待つ
k = cv2.waitKey(1)

#「ESC（27）」キーを押す
# プログラムを終了する
if k == 27:
    break
```

「ESC」キーを押すと
プログラムが終了

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 「C」キーを押す
# WEBカメラのゲイン値、露出の値を調整する
elif k == ord('c'):
    g = input("gain     : ")
    e = input("exposure : ")
    print("¥n - - - - - - - - - - ")
    camset.camera_set(cv2, cap, gain = float(g), exposure = float(e))
    camset.camera_get(cv2, cap)
    print(" - - - - - - - - - - ¥n")
```

「c」キーを押すと
画像の明るさを調整

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
# 「P」キーを押す
# 各ラベルの確率を画面上に表示する／再度押すと消える
elif k == ord('p'):
    proba_flag = not(proba_flag)
    #im = cv2.imread('data/src/lena.jpg')
    cv2.imwrite('test.jpg', frame)
    transform_pos_all = np.array(transform_pos_all)
    np.savez('Pick_up', transform_pos_all, result_num_all)

# 「H」キーを押す
# DOBOTをホームポジションに移動させる（位置リセット）
elif k == ord('h'):
    dc.move_home()
```

「p」キーを押すと
分類確率を表示

「h」キーを押すと
ホームポジションへ移動
（オフセット）

【AIあり仕分け】の説明（15分）

Classifier.pyの一部抜粋

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

```
    # DOBOTで仕分け（3-17）
    # 「S」キーを押す
    # 最後に取得した矩形とその結果を元にDOBOTでピックアップする
    elif k == ord('s'):
        print(str(result_num) + " " + LabelName[result_num] + "  -  " + str(transform_pos))
        while dc.dobot_classifier(result_num, transform_pos[0], transform_pos[1]) != True:
            pass

# 終了処理（3-18）
# DOBOTの終了処理
dc.finalize()
# キャプチャをリリースして、ウィンドウをすべて閉じる
cap.release()
cv2.destroyAllWindows()
```

「s」を押すと一つずつ
チロルチョコを仕分け

「dobotClassifier.py」も
要確認

【AIあり仕分け】の実行（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

□手順

1. カメラ座標系とロボット座標系の変換行列の作成
2. 学習用画像の作成
3. 画像データを使用した教師あり学習
4. 学習モデルを使ったチロルチョコの仕分け

Colab組
「dobot_ai_GoogleColabratory」フォルダを
丸ごと「GoogleDrive」直下にアップロード

【AIあり仕分け】の実行（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

1. カメラ座標系とロボット座標系の変換行列の作成

① 「opencv_setting.py」を実行

```
> cd **配布資料のアドレス**¥プログラム¥dobot_ai
> python opencv_setting.py
```

② カメラ画像のウィンドウが出たら、ウィンドウをクリックしたのち「S」キーを押す
→「dobot_ai」フォルダ内に「TransformationMatrix.py」が保存される

注） ４つすべてのマーカーが認識されていることを確認。認識されていない場合、
①カメラ位置調整 ②明るさ調整 ③認識エリアの大きさ修正(プログラム内)

※ カメラ位置（DOBOTとカメラの位置関係）が変わるたびに、実行してください。

※ 画像が明るすぎる／暗すぎる場合は、「C」キーを押すとコマンドプロンプトから「gain」と「exposure」（明るさ）を変更できる。

Colab組
「TransformationMatrix.py」を
「GoogleDrive」にアップロード

【AIあり仕分け】の実行（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

2. 学習用画像の作成

① 黒いマットを引き、数個のチロルチョコを適当に並べる

② 「shoot_trainingData.py」を実行

```
> python shoot_trainingData.py
```

② カメラ画像のウィンドウが出たら、ウィンドウをクリックしたのち、
「S」キー(元画像)、「A」キー(加工画像)、「R」キー(回転した画像)を１回ずつ押す。
→「data」フォルダ内に画像が保存される。

③ 「data」フォルダ内に、チロルチョコの種類ごとのフォルダを作成し、
すべての画像をそれぞれのフォルダ内に移動させる。

※ 画像が明るすぎる／暗すぎる場合は、「C」キーを押すとコマンドプロンプトから「gain」と「exposure」（明るさ）を変更できる。

Colab組
「data」フォルダの中身を
「GoogleDrive」にアップロード

【AIあり仕分け】の実行（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

3. 画像データを使用した教師あり学習

① 「train.py」を実行

> *python train.py*

② 学習の様子が表示され、最後にテストデータ入力した際の「正解率」が表示される。
「誤差の減り方(学習の進み方)」と「正解率」のグラフが「history」フォルダに保存される。

③ 「model」フォルダ内に、新たに２つのファイル（学習したネットワーク）、
「data」フォルダ内に、新たに１つのファイル（学習データをまとめたもの）が確認できる。

注） 新しく学習する場合、これら３つを削除(または移動)してから「train.py」を実行。
上書きされない（前の学習の状態を保持する）ので注意。

Colab組
「GoogleDrive」で「classifier_ForGoogleColab.ipynb」をダブルクリックし
上から順番に再生マーク（▶）を押していく（「!python train.py」まで）

【AIあり仕分け】の実行（15分）

◆カメラ画像からチロルチョコを分類し、ロボットアームで種類ごとに仕分ける。

4. 学習モデルを使ったチロルチョコの仕分け

① 「classifier.py」を実行

```
> python classifier.py
```

「H」キー→ロボットアームのオフセット
「C」キー→画像の明るさを調整
「ESC」→プログラム終了

② カメラ画像のウィンドウが出たら、ウィンドウをクリックしたのち、「P」キーを押す。
→画像上に分類確率が表示される。

③ 「S」キーを押す。→チロルチョコを一つずつロボットアームで仕分けする。

Colab組

①PC上で「classifier_ShootImage.py」を実行し「P」キーを押すと「test.jpg」が生成される。
②Driveに「test.jpg」 をアップロードし、Colab上で「!python classifier_ForGoogleColab.py」を実行
③Driveから「Pick_up.npz」をダウンロードし、PC上の「dobot_ai_GoogleColabratory」フォルダに移動
④ PC上で「classifier_WithoutLearning.py」を実行し「P」キーを押すとロボットアームで仕分けする

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。（レポート：70/100）**

- チロルチョコ以外の対象物の仕分け（例えばペットボトルのキャップなど）
- エンドエフェクタの変更（吸引カップ→グリッパ）
- 認識しづらいチロルチョコ（コーヒー味）の認識
- より多い種類のチロルチョコの仕分け

など、独自の課題を設定してもらいます。（上記から選んでも問題ありません）

□望ましい対象物

吸引カップ☞**表面が平なもの**　グリッパ☞**挟む位置の幅が2.8[cm]以下のもの**

(+α)同じ対象物でも絵柄が若干異なるもの（例えばチロルチョコのミルク味は絵柄が10パターン）

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。（レポート：70/100）**

- 進め方
  1. 独自の課題を設定（ペアの人と相談しても良い(同じはダメ)）←**決まったら教えてください。**
  2. プログラムやデータセットの変更
  3. プログラムの実行
  4. 結果のまとめ、考察

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。（レポート：70/100）**

- 進め方

1. 独自の課題を設定（ペアの人と相談しても良い(同じはダメ)）←**決まったら教えてください。**
2. プログラムやデータセットの変更
3. プログラムの実行
4. 結果のまとめ、考察

5. ペアの人の課題設定を聞く
6. ペアの人に、どのような変更や工夫をしたか聞く
7. 結果を教えてもらい、**自分で**考察

授業中でも授業後でも可

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。（レポート：70/100）**

- 進め方
  1. 独自の課題を設定（ペアの人と相談しても良い(同じはダメ)）←**決まったら教えてください。**
  2. プログラムやデータセットの変更
  3. プログラムの実行
  4. 結果のまとめ、考察
  5. ペアの人の課題設定を聞く
  6. ペアの人に、どのような変更や工夫をしたか聞く
  7. 結果を教えてもらい、**自分で**考察

ペアの人だけではなく
他の人の課題も聞き、考察しても良いです。
レポートに２人以上の他者の課題について書いた場合
加点対象ですが、70/100は超えません。

授業中でも授業後でも可

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。（レポート：70/100）**

□ 取り組むときの注意点

- 変更点が複数ある場合、１つずつ問題を解決していく。

  例）①**エンドエフェクタをグリッパに変え**て、②**より多種のチロルチョコを仕分け**する。

  →①と②を同時に取り組むのではなく、①(②)を達成（①の部分だけ変更）してから②(①)に取り組むこと。

- 変更するファイルは「classifier.py」だけではないかもしれない。

  「classifier.py」にかかわるファイル

  「myDobotModule.py」：ロボットアーム操作にかかわる関数

  「dobotClassifier.py」　：ロボットアームの基本操作（初期設定も含む）

  etc…

レポートについて

- 【移動中の軌道】の違い（MOVJ、MOVL、JUMP）の長所・短所（6/100）と、それぞれどんな用途に適しているか（4/100）が書かれている。**（計10/100）**
- 【チロルチョコの積み上げ】のプログラムが完成している。**（10/100）**
- AI技術の一つであるニューラルネットワークおよびディープラーニングの基本的な考え方を理解できている。**（10/100）**
- 画像認識によるロボットアーム制御について、独自の課題を設定（10/100）し、プログラムやデータセット等を工夫した（10/100）うえで、課題を解決するプログラムが完成（10/100）しており、結果・考察（10/100）が記されている。
また、他の学習者（最低1人以上）の、課題設定（10/100）と、工夫した点（10/100）、結果からの考察（10/100）が示されている。**（計70/100）**

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。**

- チロルチョコ以外の対象物の仕分け（例えばペットボトルのキャップなど）
- より多い種類のチロルチョコの仕分け

ヒント

- 学習データセットを増やす
- 丸い対象物の場合、データセット作成でより細かく回転角を変化させる。→データの水増し

円検出に変更しても良い？
（試してないので注意）
cv2.HoughCircles()

shoot_trainingData.pyの一部抜粋(**430行目**付近)

チロルチョコは四角なので90度回転

```
        # 回転（0°, 90°, 180°, 270°）して、変換処理した画像を保存
        for j in range(0, 4):
            rot = 90 * j
            # 回転変換行列の算出
            rotation_matrix = cv2.getRotationMatrix2D(center, angle=rot, scale=1.0)
            # アフィン変換
            rot_img = cv2.warpAffine(img_src, rotation_matrix, size, flags=cv2.INTER_CUBIC)
```

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。**

- エンドエフェクタの変更（吸引カップ→グリッパ）

ヒント

- グリッパでの制御方式を定義する必要がある（dobotClassifier.pyの36~77行目参照）。
- グリッパにすることで、チロルチョコの回転角を考慮しなければならない。

Classifier.pyの一部抜粋(**291行目**付近)

```
# 最後に取得した矩形とその結果を元にDOBOTでピックアップする
elif k == ord('s'):
    print(str(result_num) + " " +
        LabelName[result_num] + "  -  " +
        str(transform_pos))
    while dc.dobot_classifier(result_num,
        transform_pos[0], transform_pos[1]) != True:
        pass
```

この関数は吸引カップ用

Classifier.pyの一部抜粋(**185行目**付近)

```
# フレーム画像から対象物を切り出す（3-8）
# 回転を考慮した外接矩形を取得する
rect = cv2.minAreaRect(contour)
box = cv2.boxPoints(rect)
box = np.int0(box)
center, size, angle = rect
```

チロルチョコの角度を取得

【AIを使った問題（自作）】（75分）

◆**最後に自身で課題を設定してもらい、その解決に取り組みます。**

- 認識しづらいチロルチョコ（コーヒー味）の認識

ヒント

- 認識が難しい原因は、背景の黒に隠れるため。→背景を白くすればよい。
  ↔関数cv2.findContours()は黒背景を前提としている。

shoot_trainingData.pyの一部抜粋(**162行目**付近)

```
contours, hierarchy = cv2.findContours(bw, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

入力画像

☞白い背景の画像を白黒反転するとどうなる？

備考）画像を白黒反転する関数：cv2.bitwise_not(**画像**)

片付け（グリッパへの付け替え）15分

1. 吸引カップのネジを緩める
   ネジは<u>完全には抜かない</u>

2. グリッパを差し込む

3. グリッパの**<u>上の</u>**ネジを締める
   <u>かなり強めに締める</u>

参考資料

- Afrel，【教材】DOBOT AI×画像認識×ロボットアーム制御，2022年(第3版)
- 中井悦司，TensorFlowで学ぶディープラーニング入門～畳み込みニューラルネットワーク徹底解説，2017年(第4刷)