AI Learning Team

AI概要と微分

Agenda

1. AI概要
2. 機械学習とは
3. ニューラルネットワーク
4. 関数
5. 微分

1.AI概要

最近のすげえAI

1. 自然言語処理
2. 画像認識(物体検知)
3. 生成モデル

最近のすげえAI

自然言語処理

最近のすげえAI

画像認識（物体検知）

最近のすげえAI

生成モデル

AI概要

Artificial Intelligenceの略

日本語で人工知能

人間の知的活動をソフトウェアで再現したもの

AI概要

第一次AIブーム - 探索と推論の時代（1950年代）

　トイプロブレム

第二次AIブーム - 知識の時代（1980年代）

　エキスパートシステム

第三次AIブーム - 機械学習の時代（2010年～）

　ニューラルネットワーク

　Deep Learning

2.機械学習とは

機械学習とは

コンピュータを使って、大量のデータから自ら法則を学習する手法

機械学習とは

エキスパートシステムと何が違うのか

- エキスパートシステム
  - **形式知**のプログラミング
- 機械学習
  - **暗黙知**のプログラミング

機械学習とは

暗黙知とは「人間が言葉で説明できない知識」

この花は美しいか

この花はいつ枯れるか

この花は朝顔か

機械学習とは

「この花が美しいか」どうやって判断するのか

=> 正解がない

=> **統計をとる**

具体的には

「美しい花」と「そうではない」画像を読み込んで、

人間が美しいと認識するパターンを見つけ出す

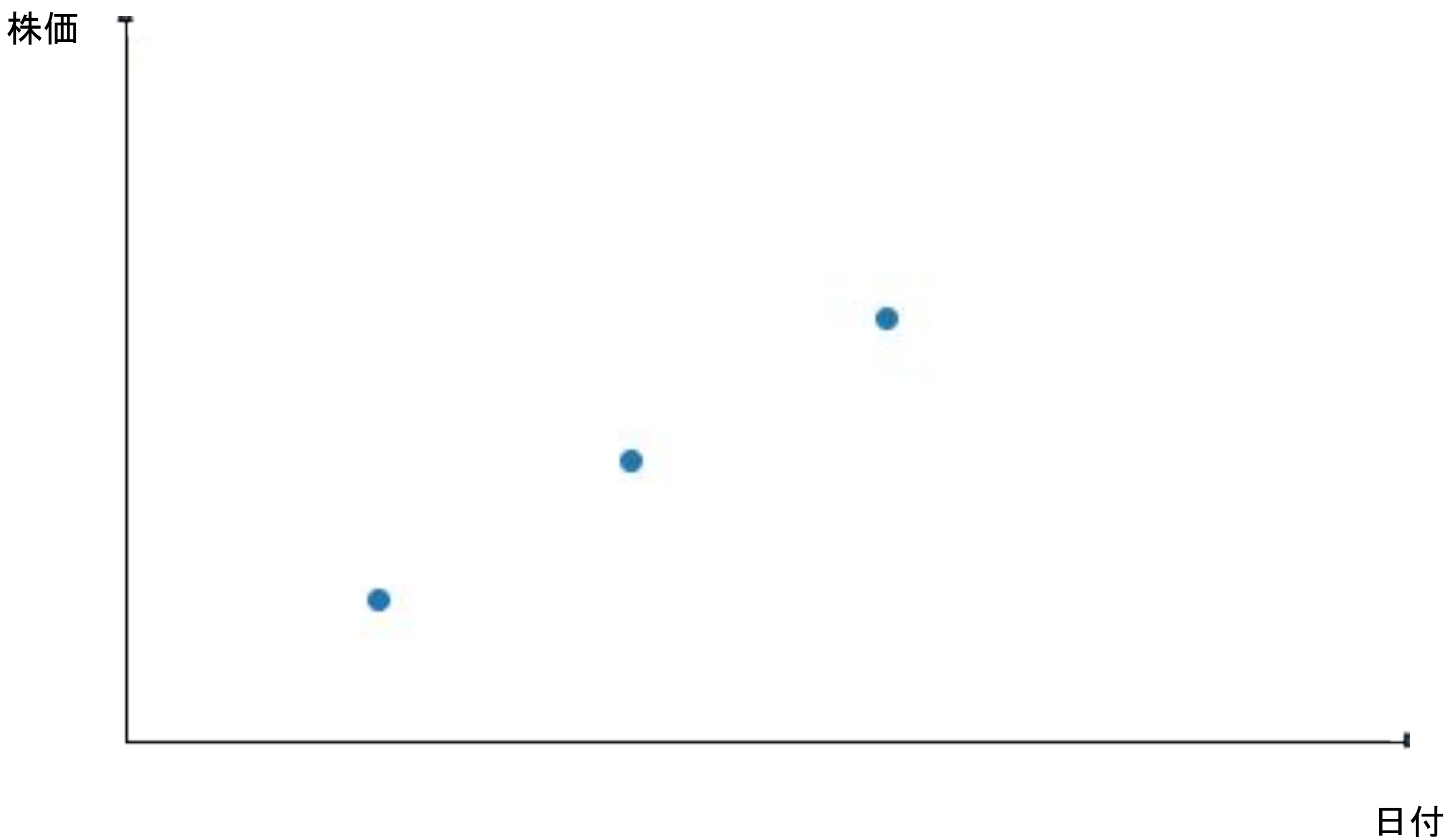
機械学習とは

統計には

- 回帰
- 分類

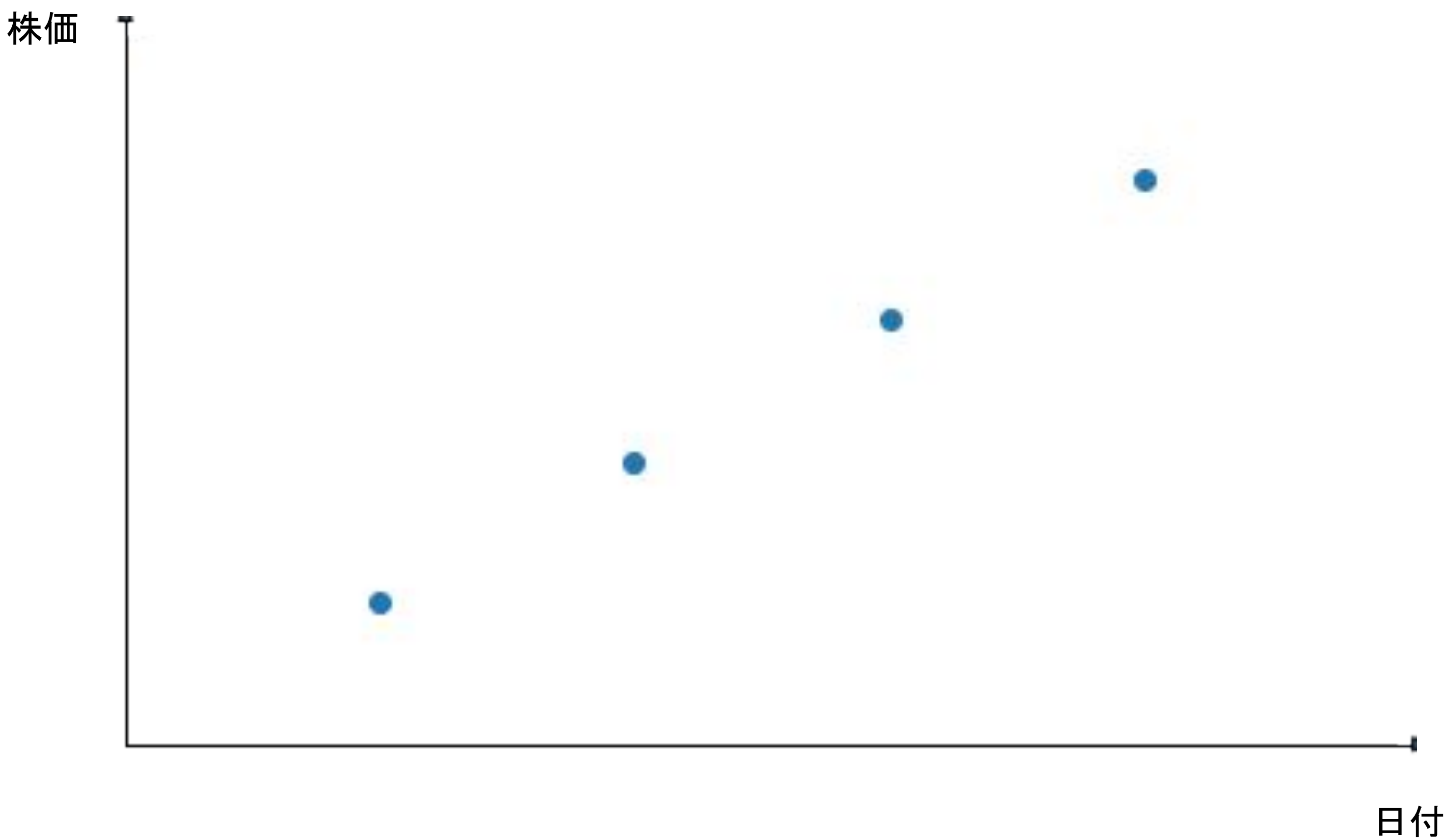
機械学習とは

回帰



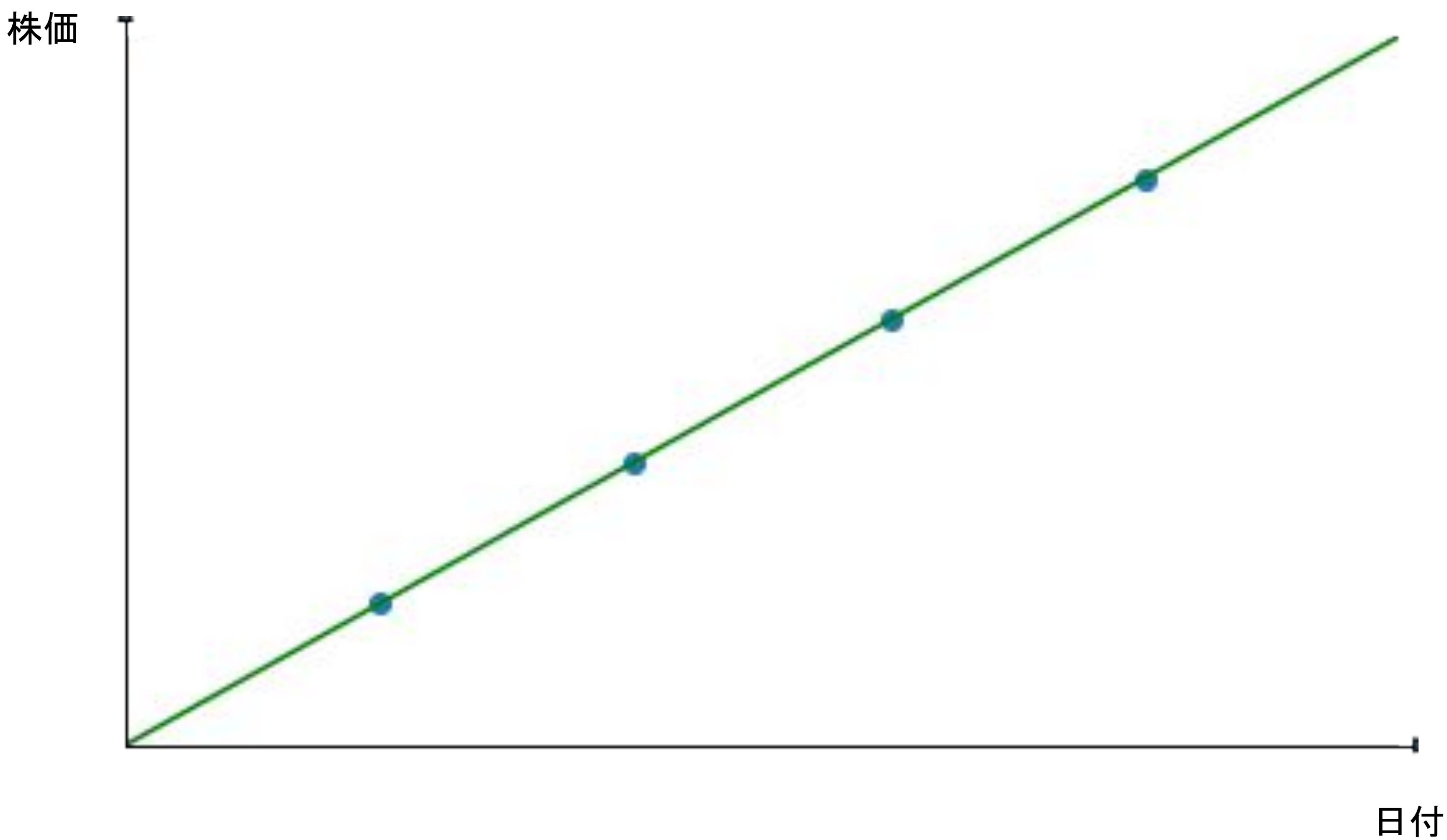
機械学習とは

回帰



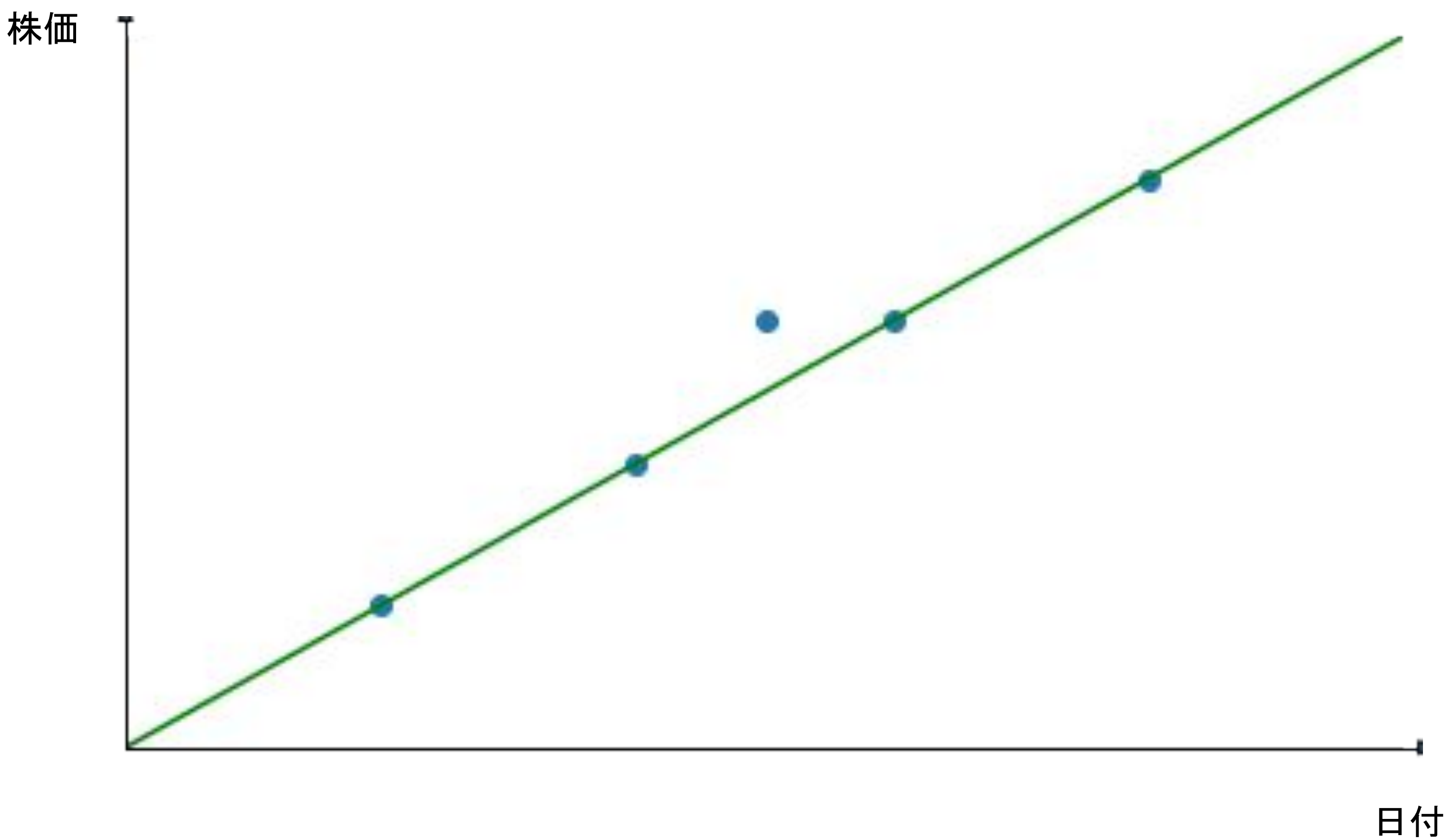
機械学習とは

回帰



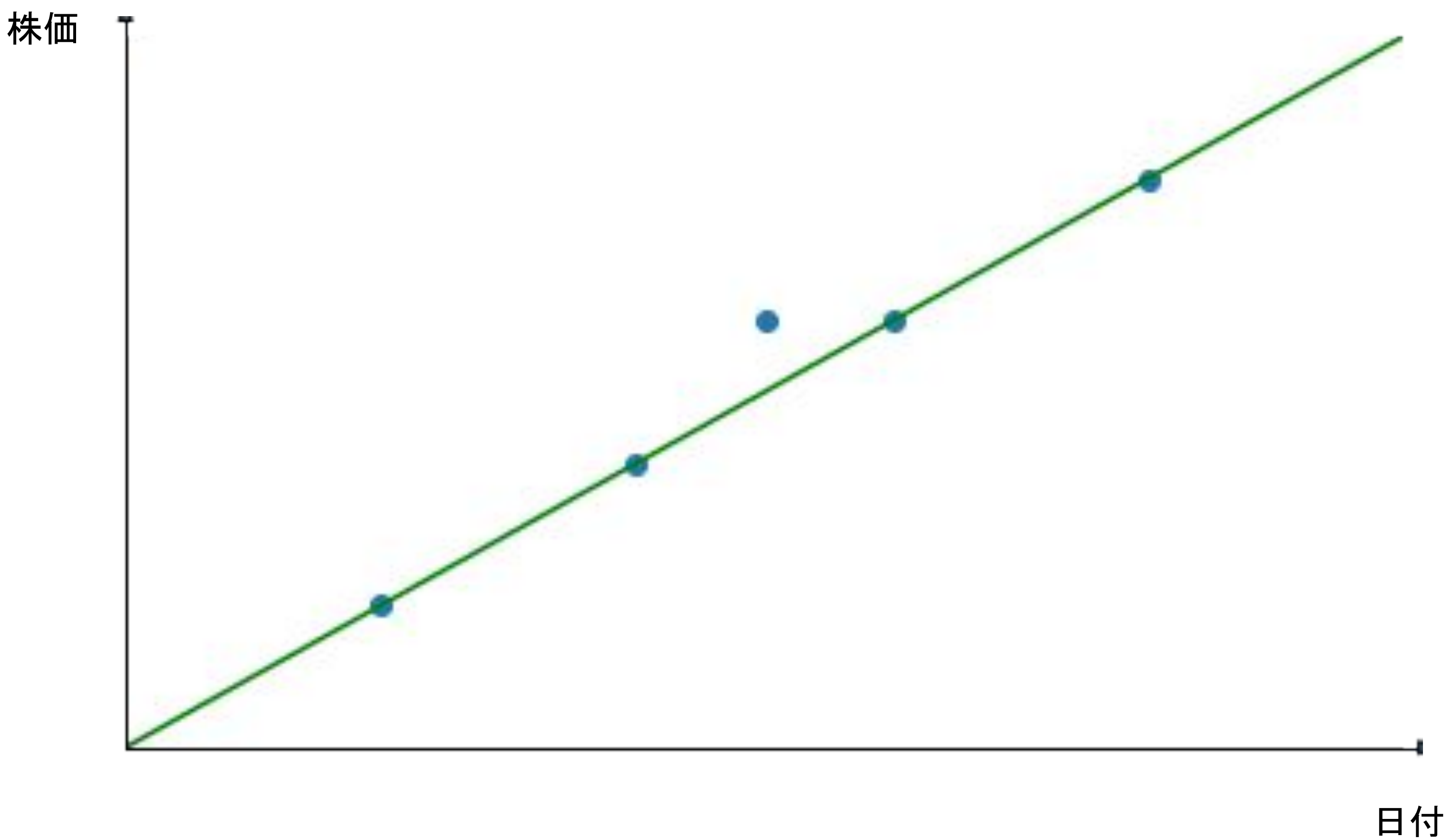
機械学習とは

回帰



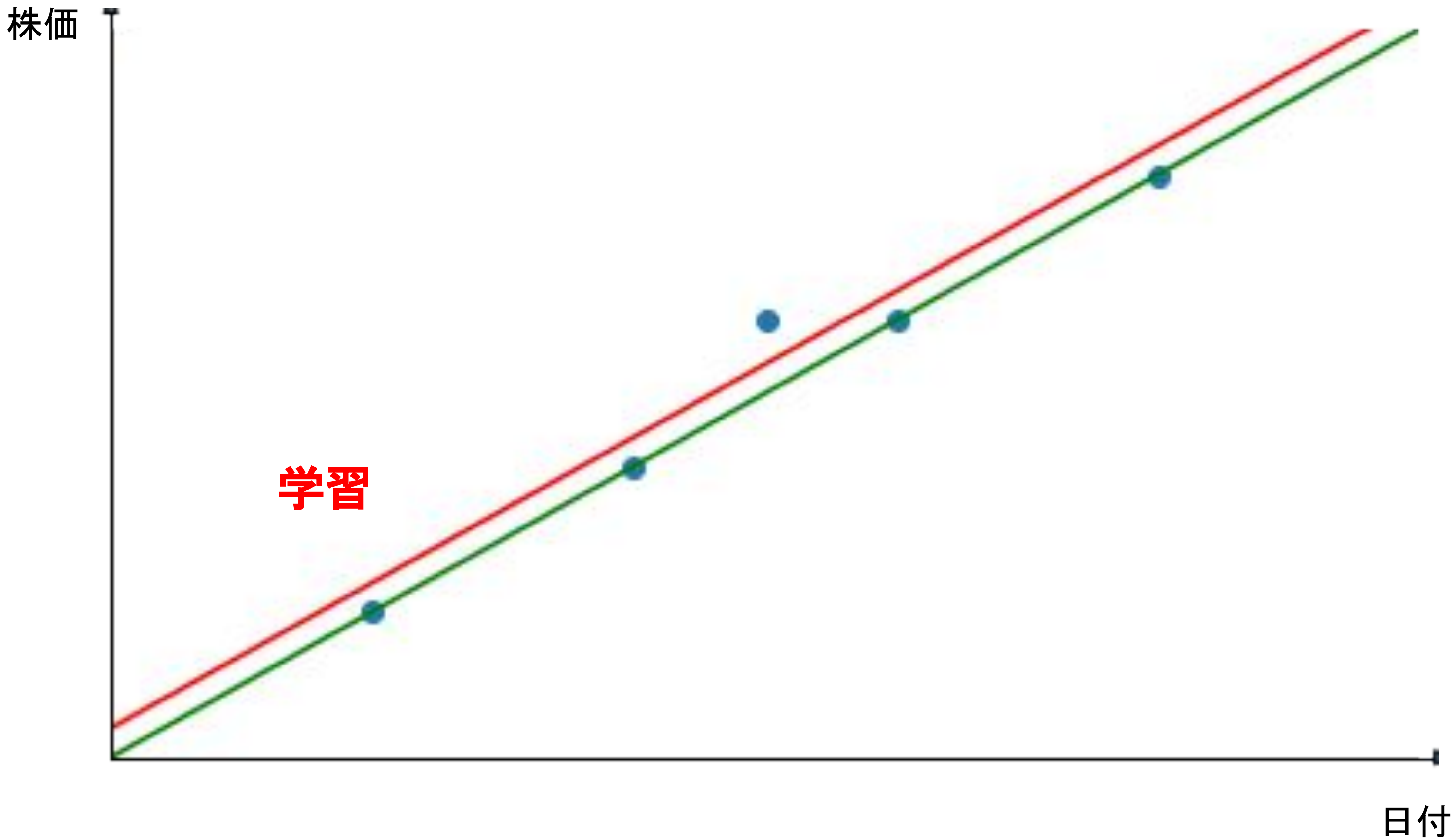
機械学習とは

回帰



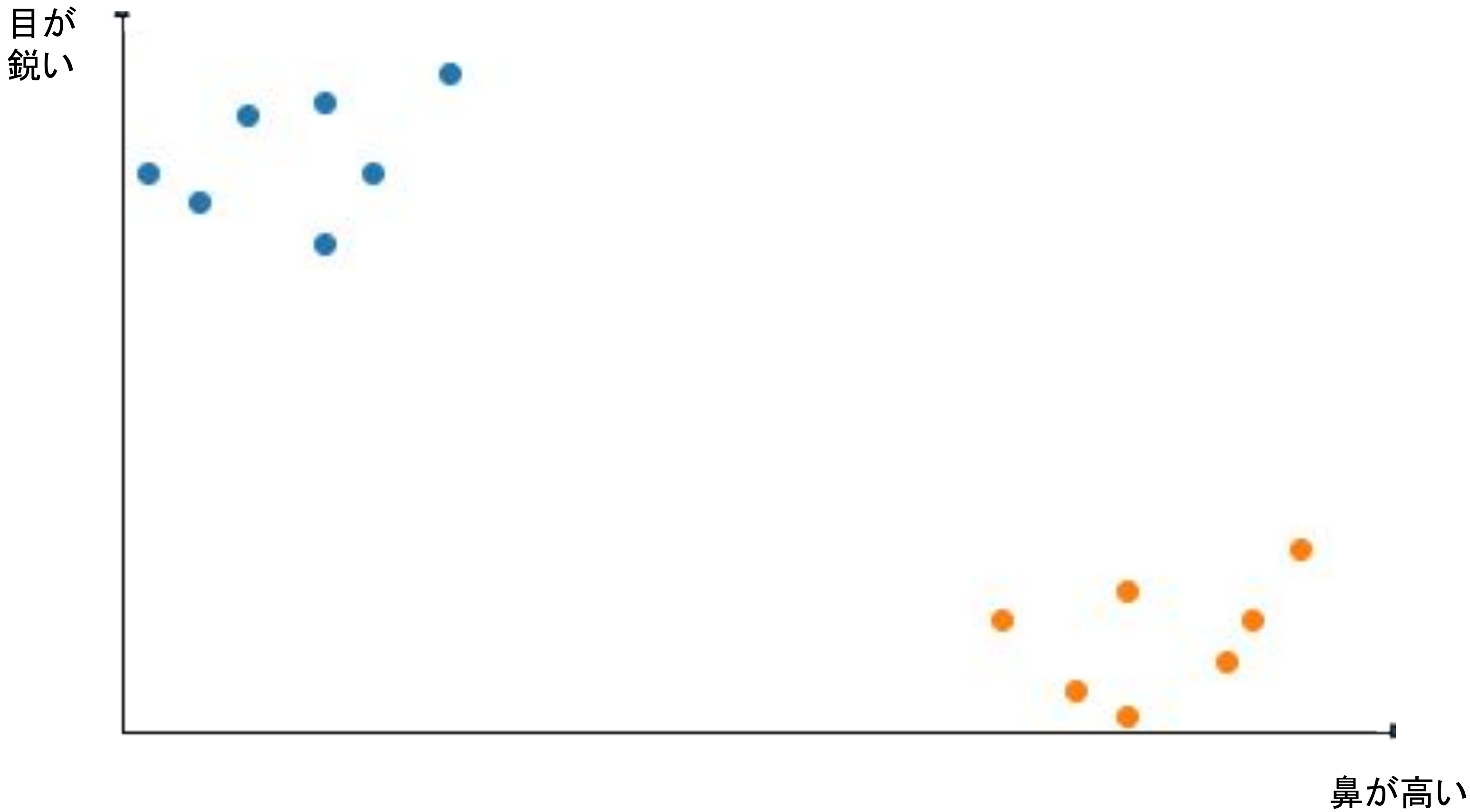
機械学習とは

回帰



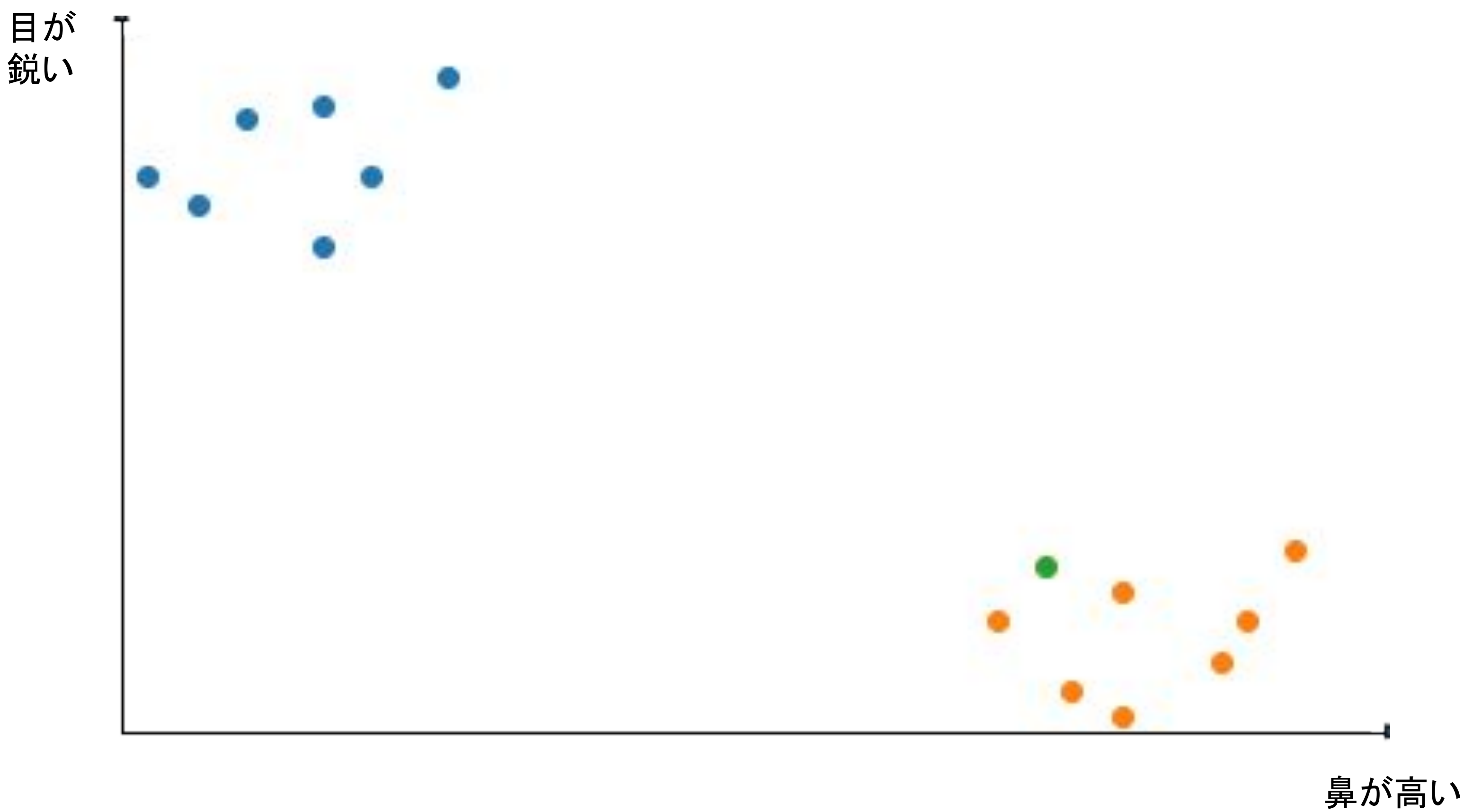
機械学習とは

分類



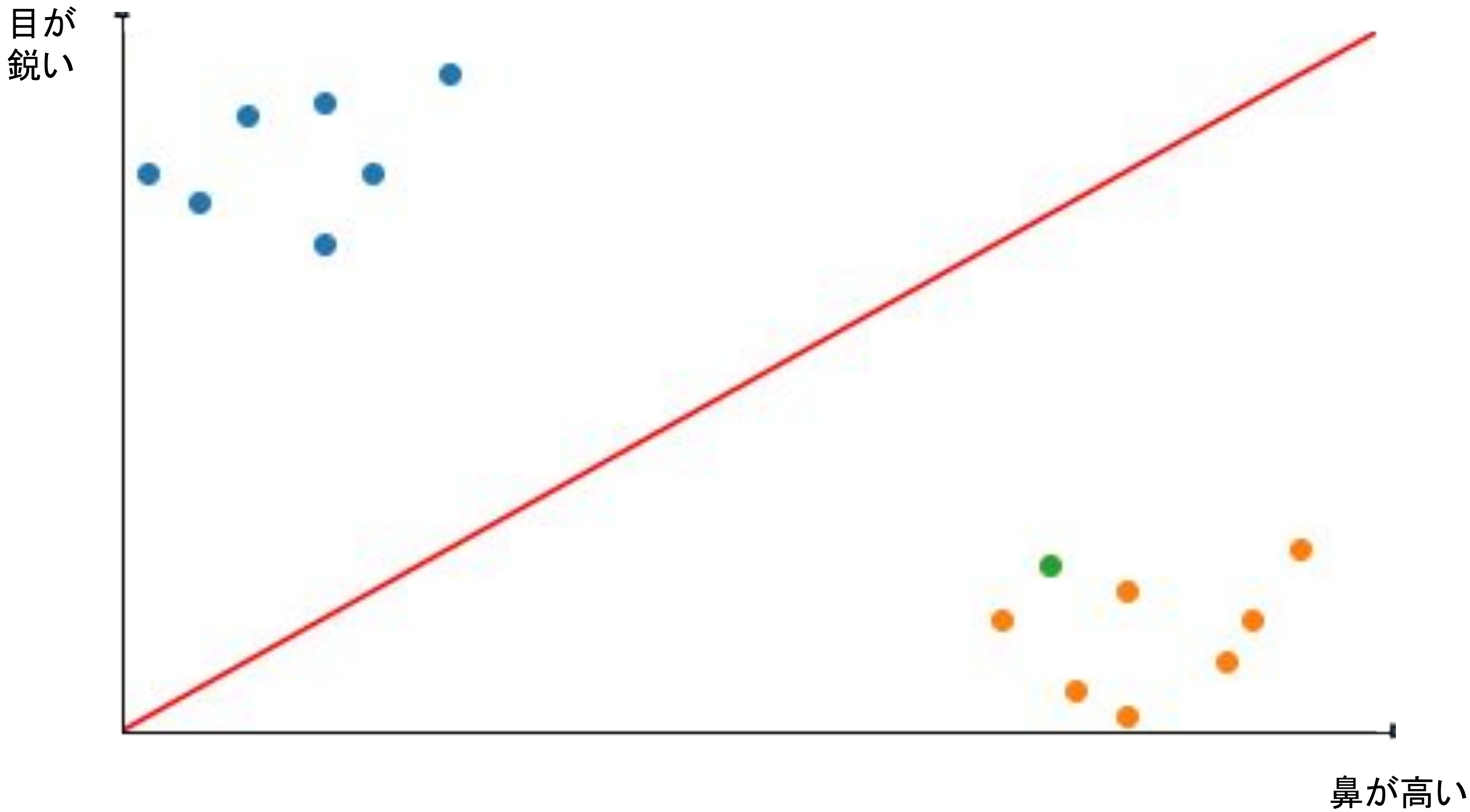
機械学習とは

分類



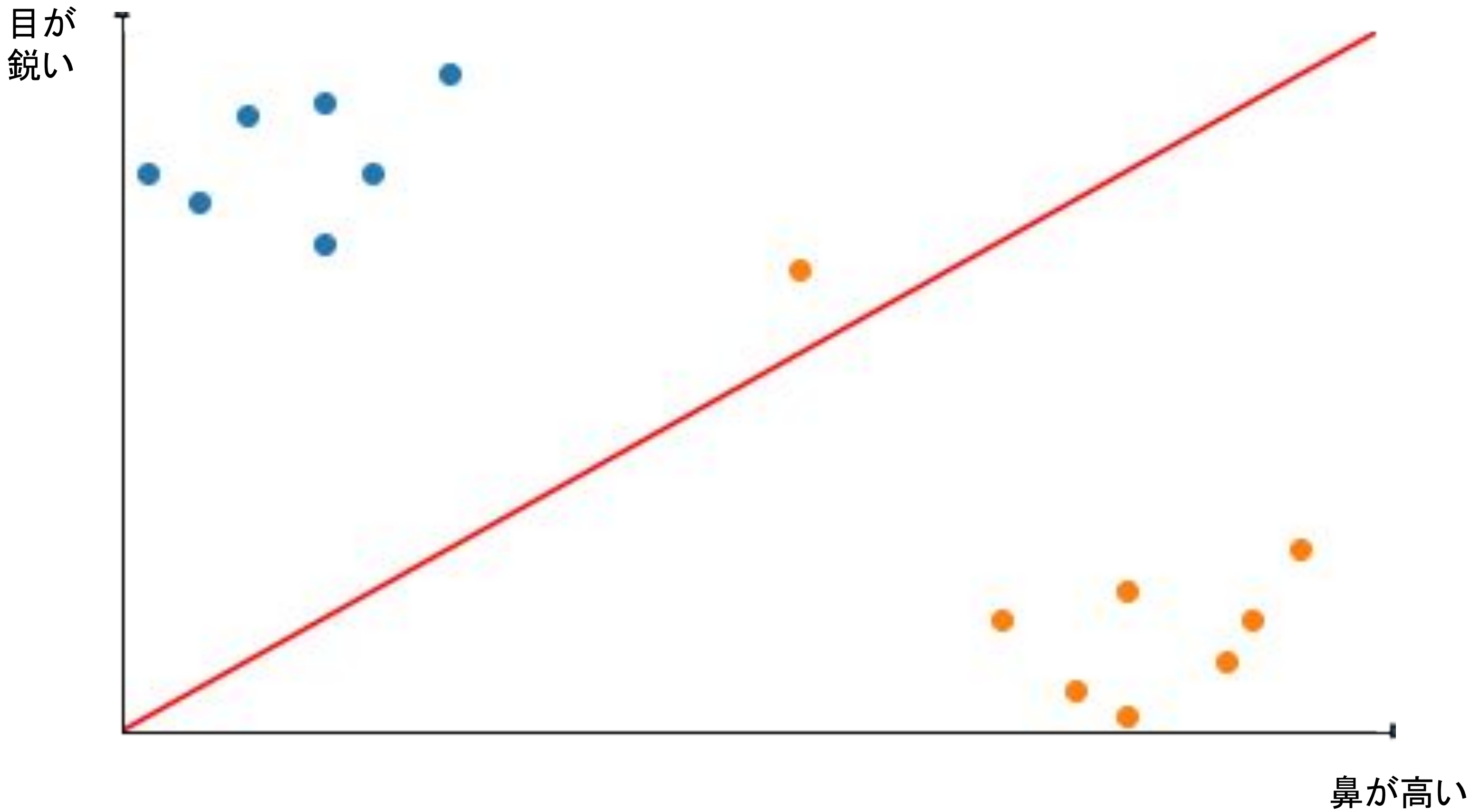
機械学習とは

分類



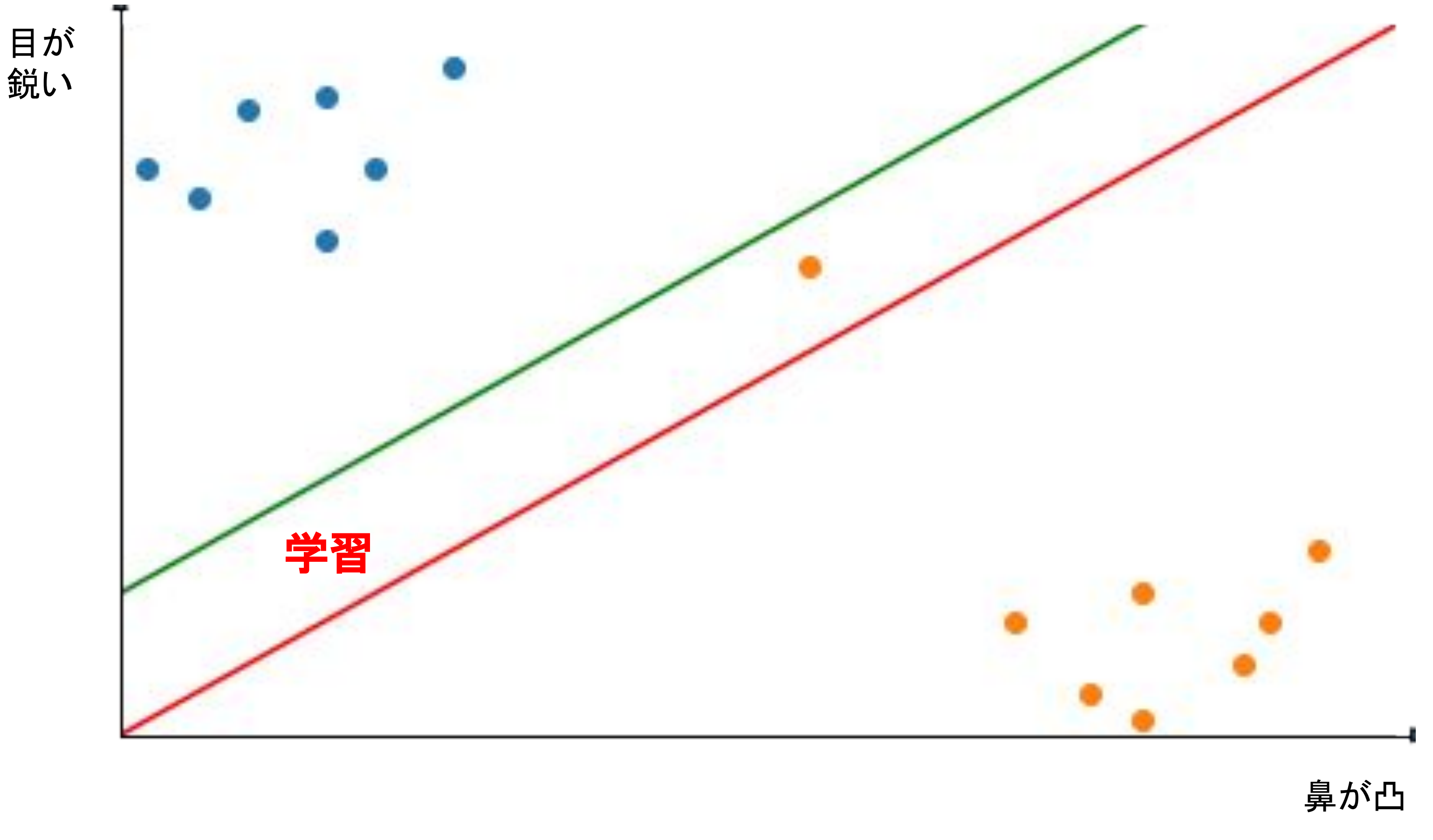
機械学習とは

分類



機械学習とは

分類



機械学習とは

コンピュータで、大量データから統計的に学習を行うこと

回帰と分類ができる

最近のすげえAI

自然言語処理

最近のすげえAI

画像認識（物体検知）

最近のすげえAI

生成モデル

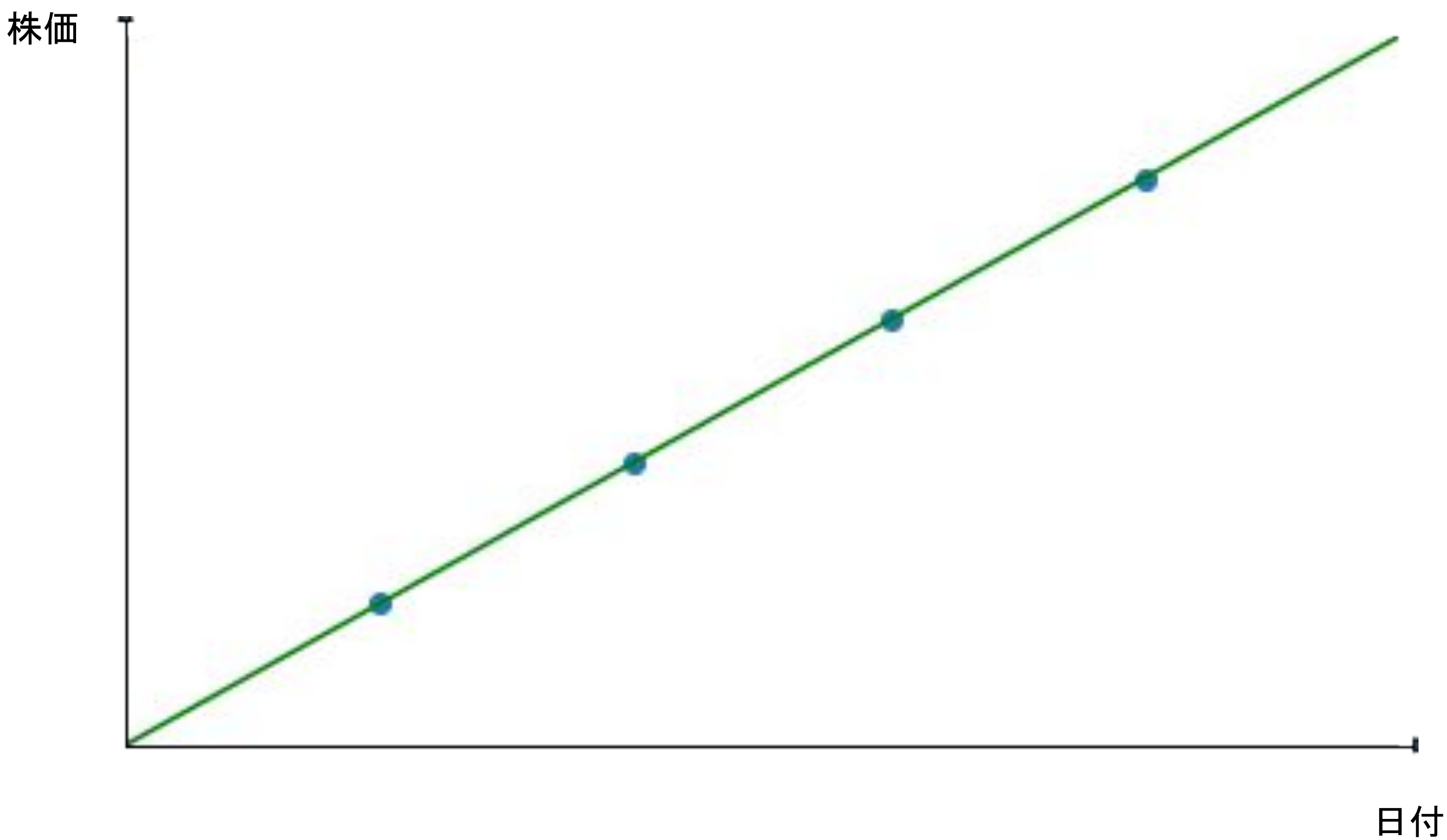
機械学習とは

統計手法

- 線形回帰
- SVM
- 決定木
- **ニューラルネットワーク**

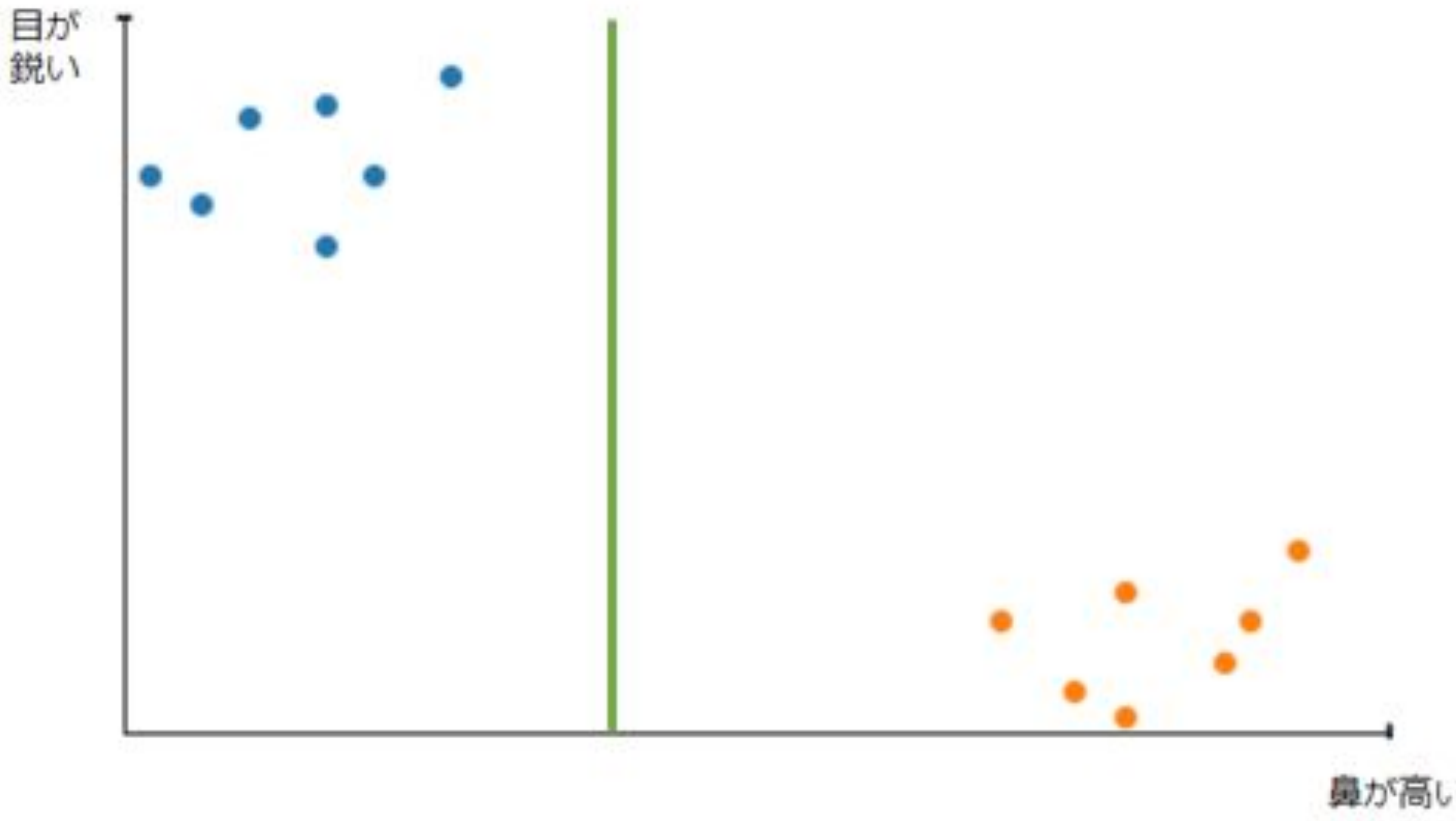
機械学習とは

線形回帰



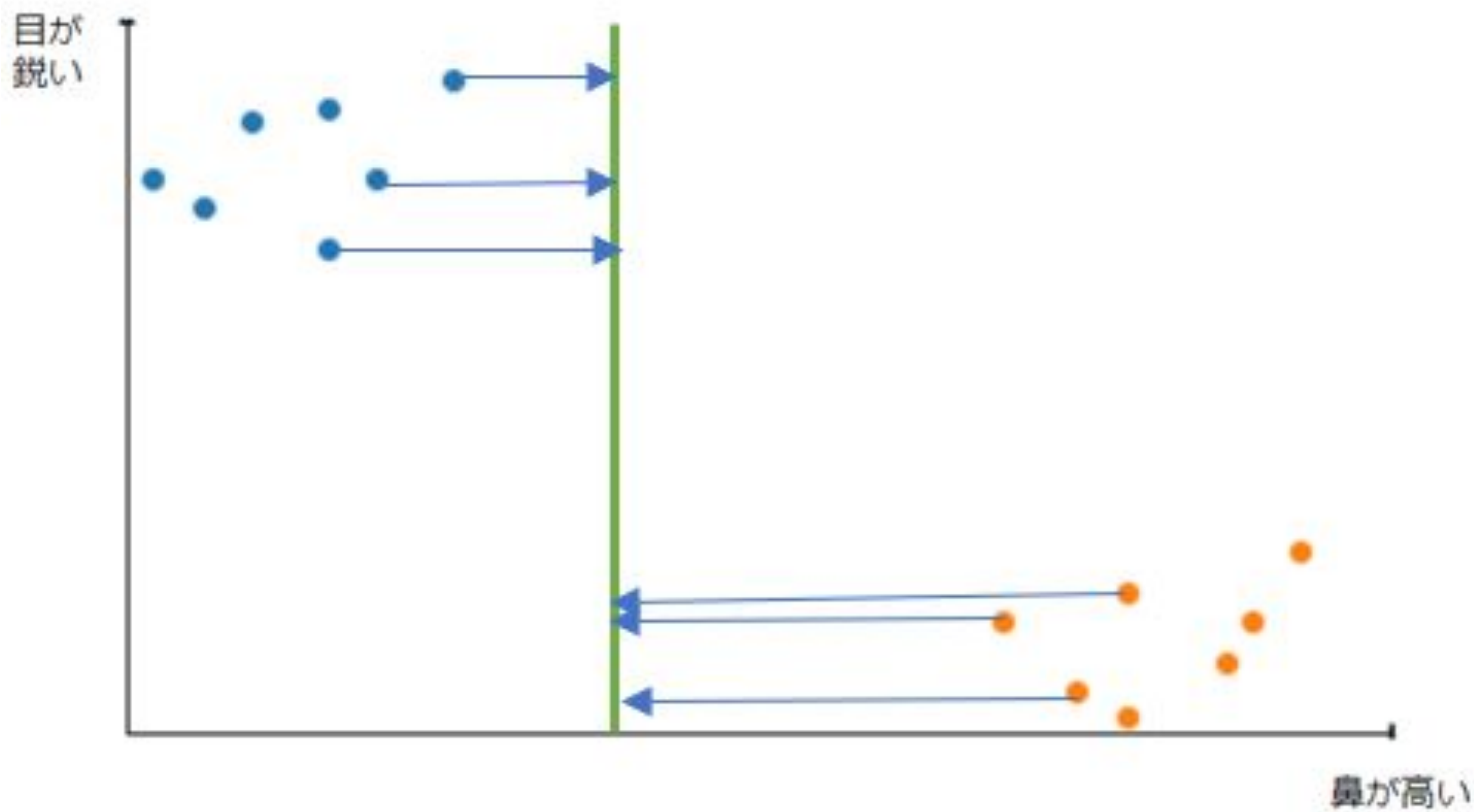
機械学習とは

SVM



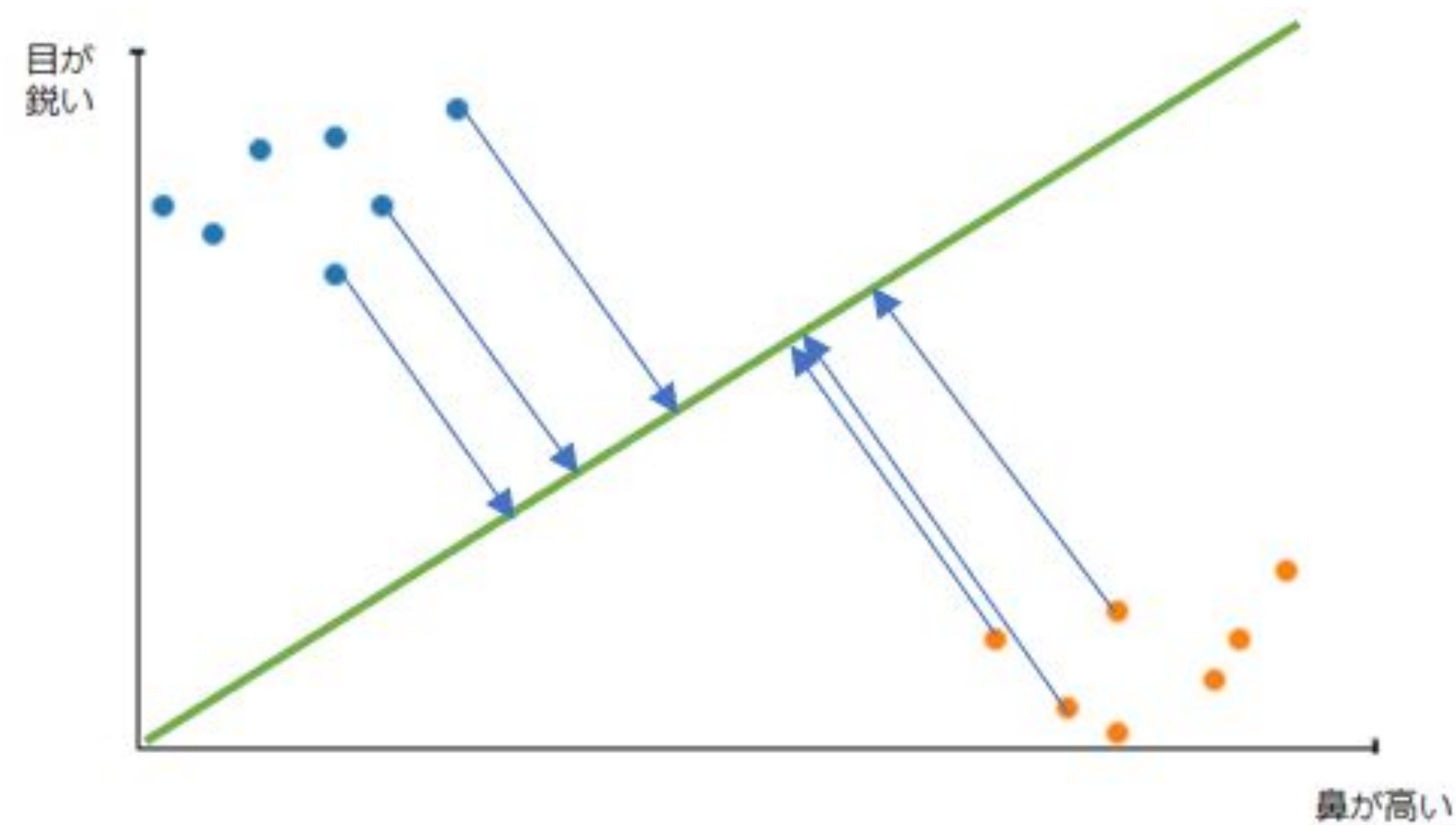
機械学習とは

SVM



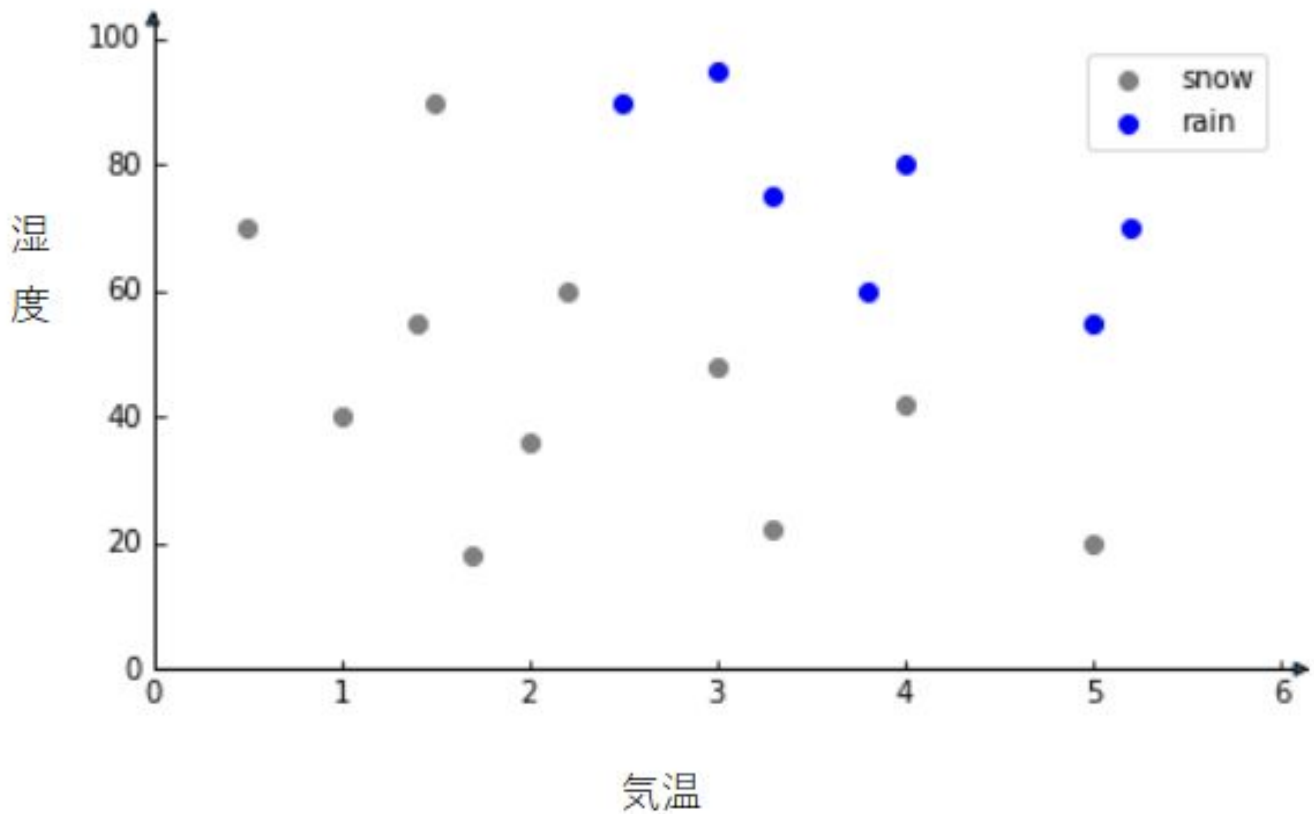
機械学習とは

SVM



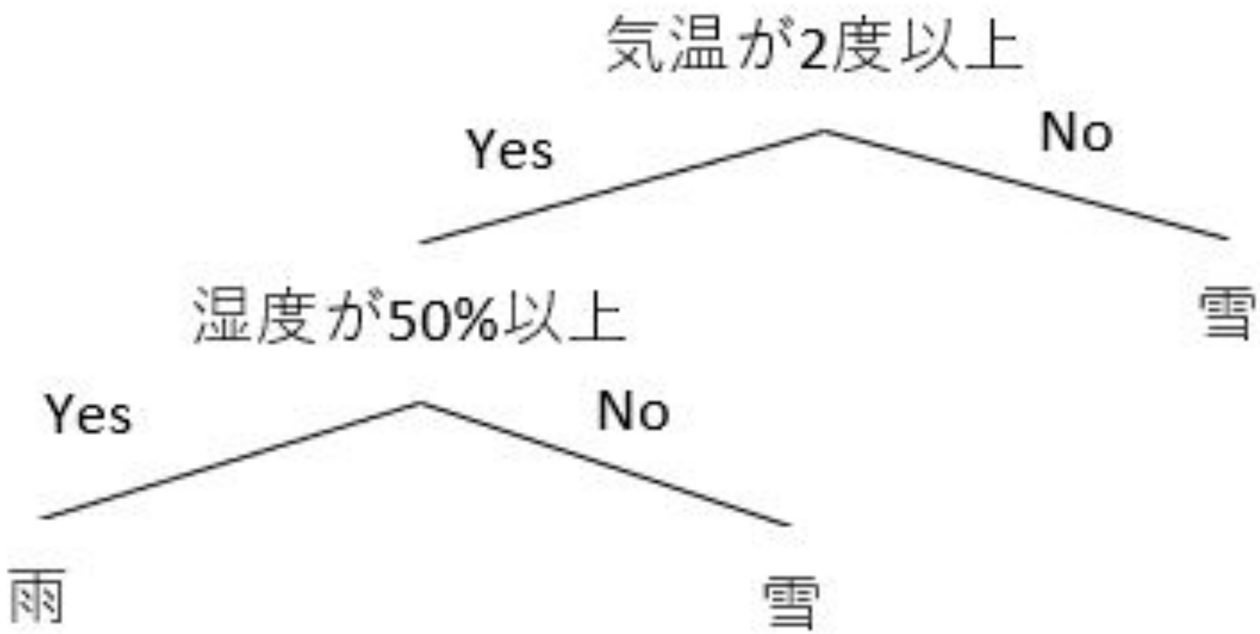
機械学習とは

決定木



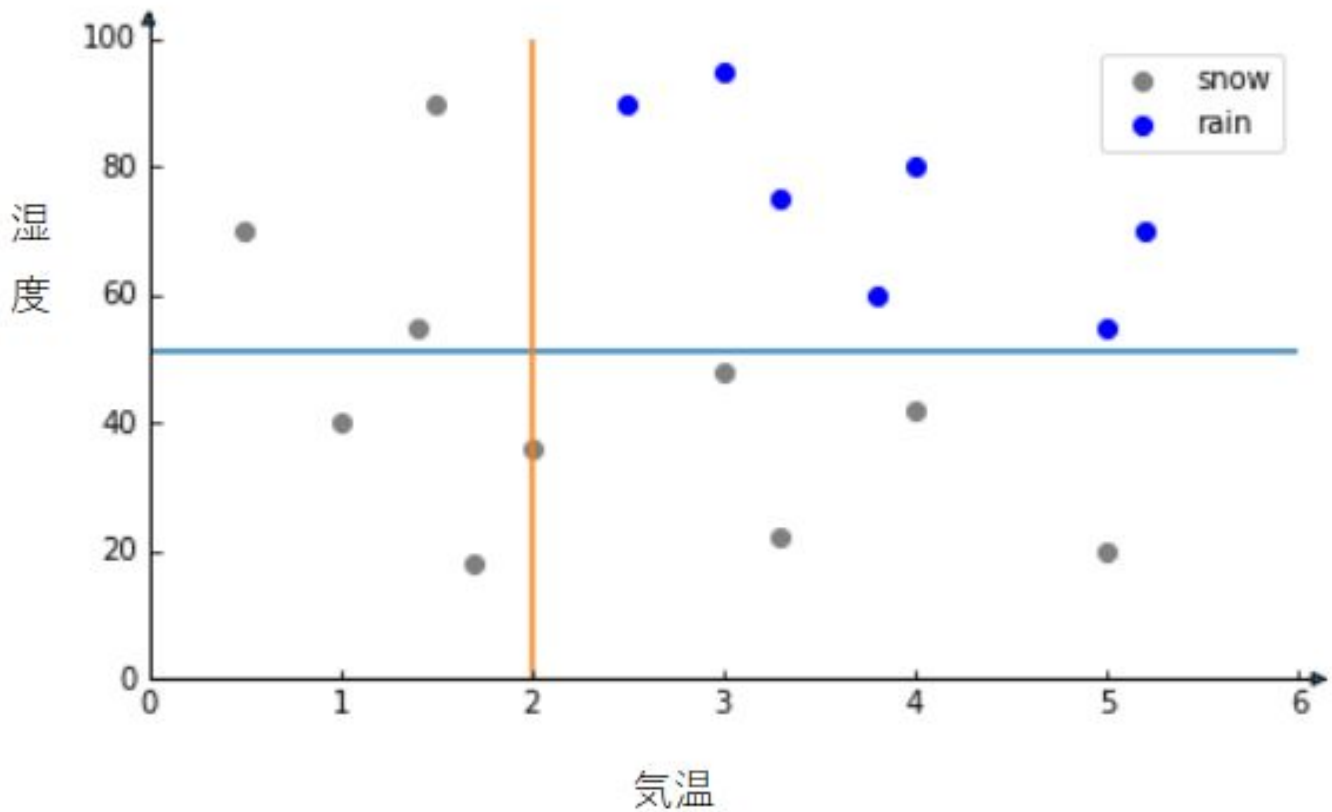
機械学習とは

決定木



機械学習とは

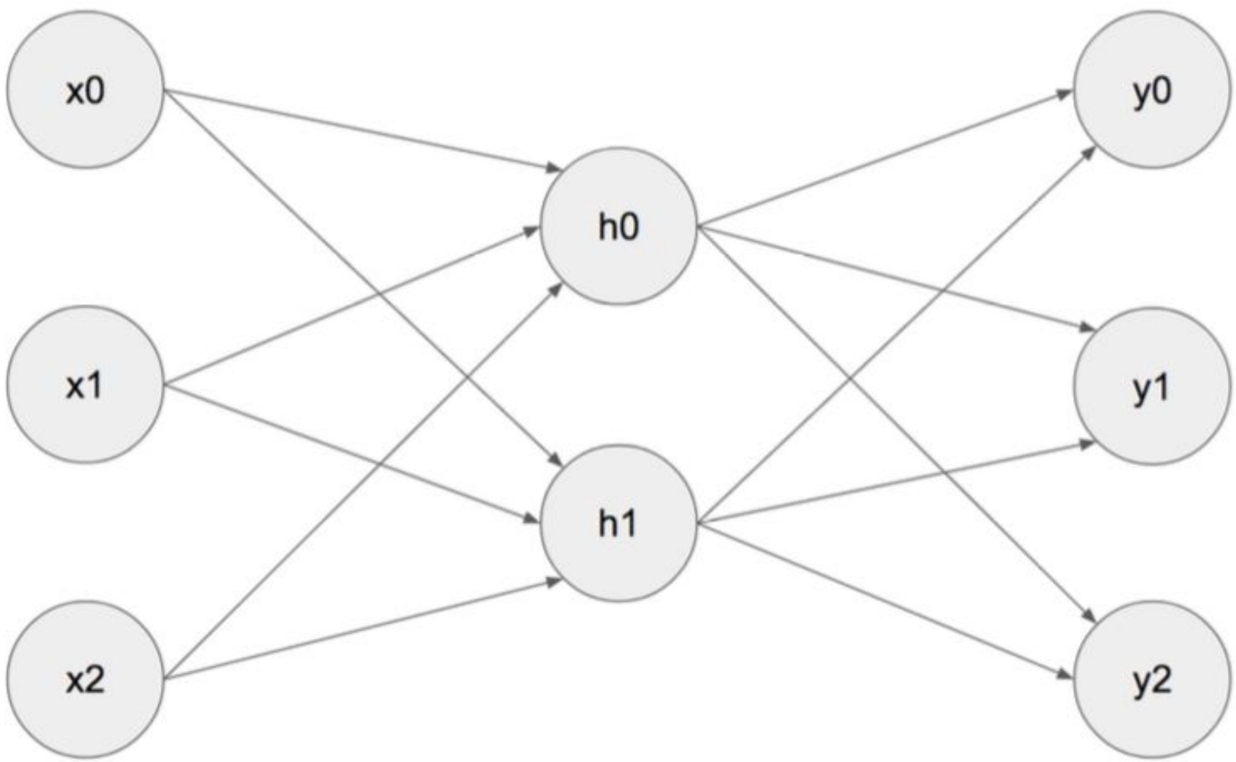
決定木



3.ニューラルネットワーク

3.ニューラルネットワーク

人間の脳が学習する過程をソフトウェアで再現したもの

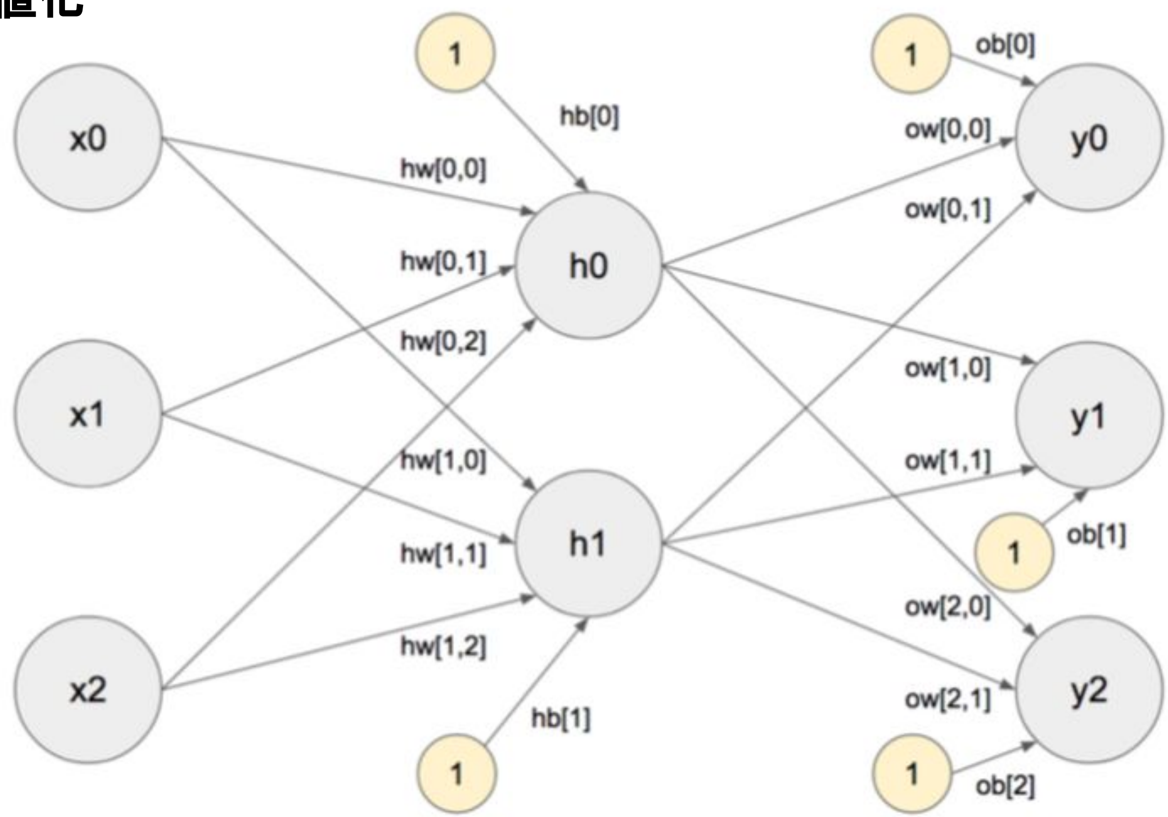


入力層 **隠れ層** **出力層**

3.ニューラルネットワーク

どう使うのか

画像を数値化して流す



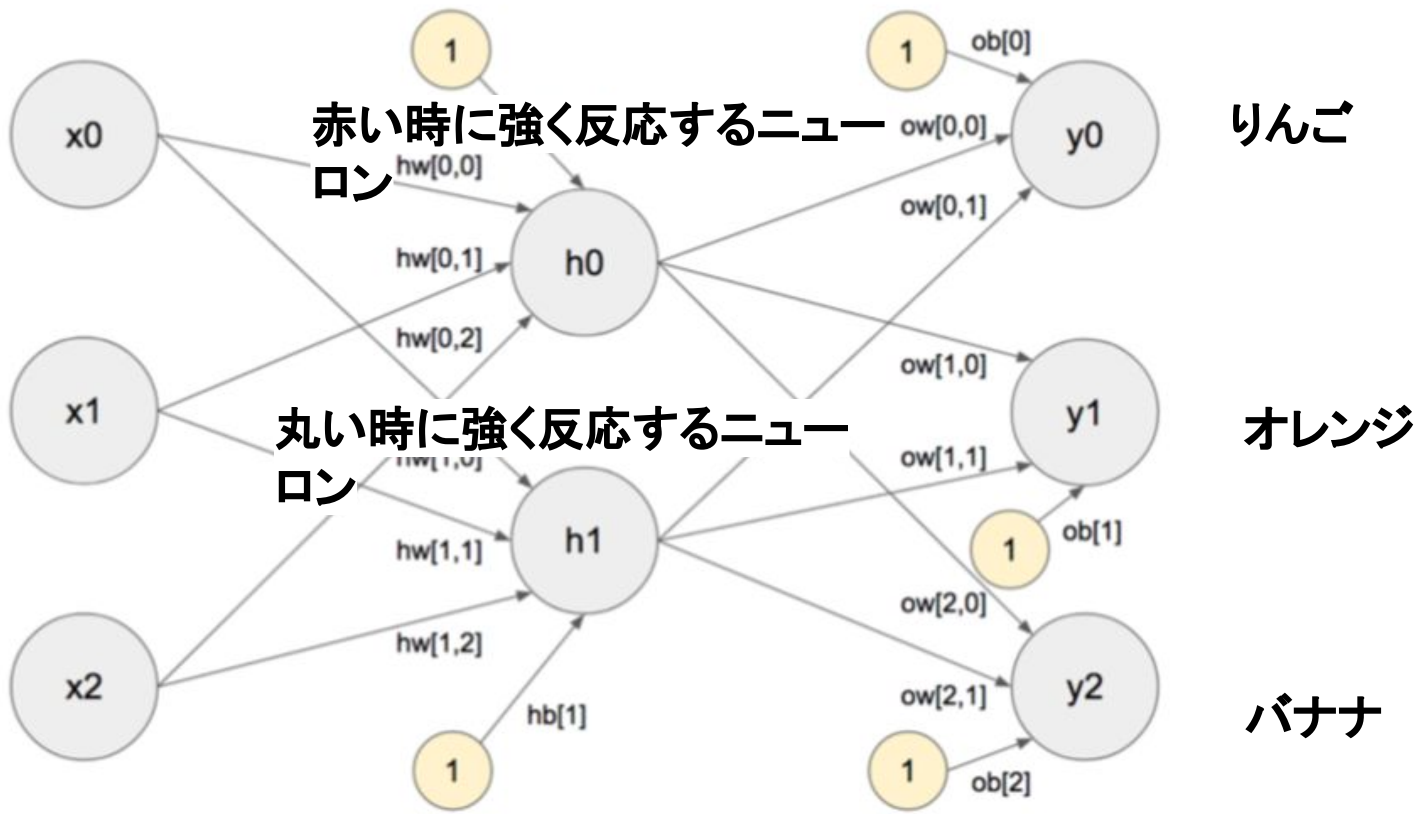
りんご

オレンジ

バナナ

3.ニューラルネットワーク

ニューロンにはそれぞれ役割があり重みが違う



3.ニューラルネットワーク

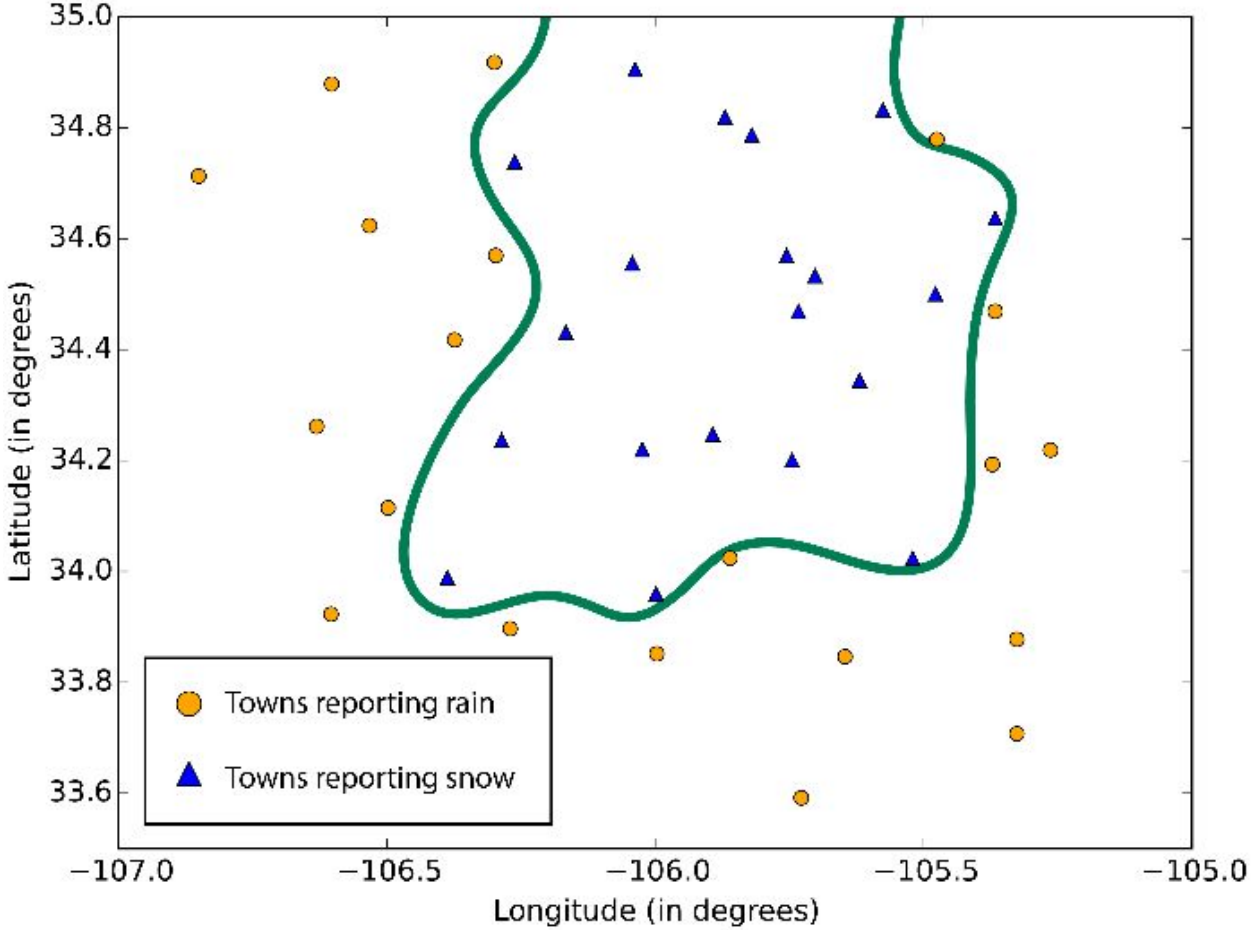
ニューラルネットワークの強み

１. 表現力が高い

２. 特徴量設計が不要

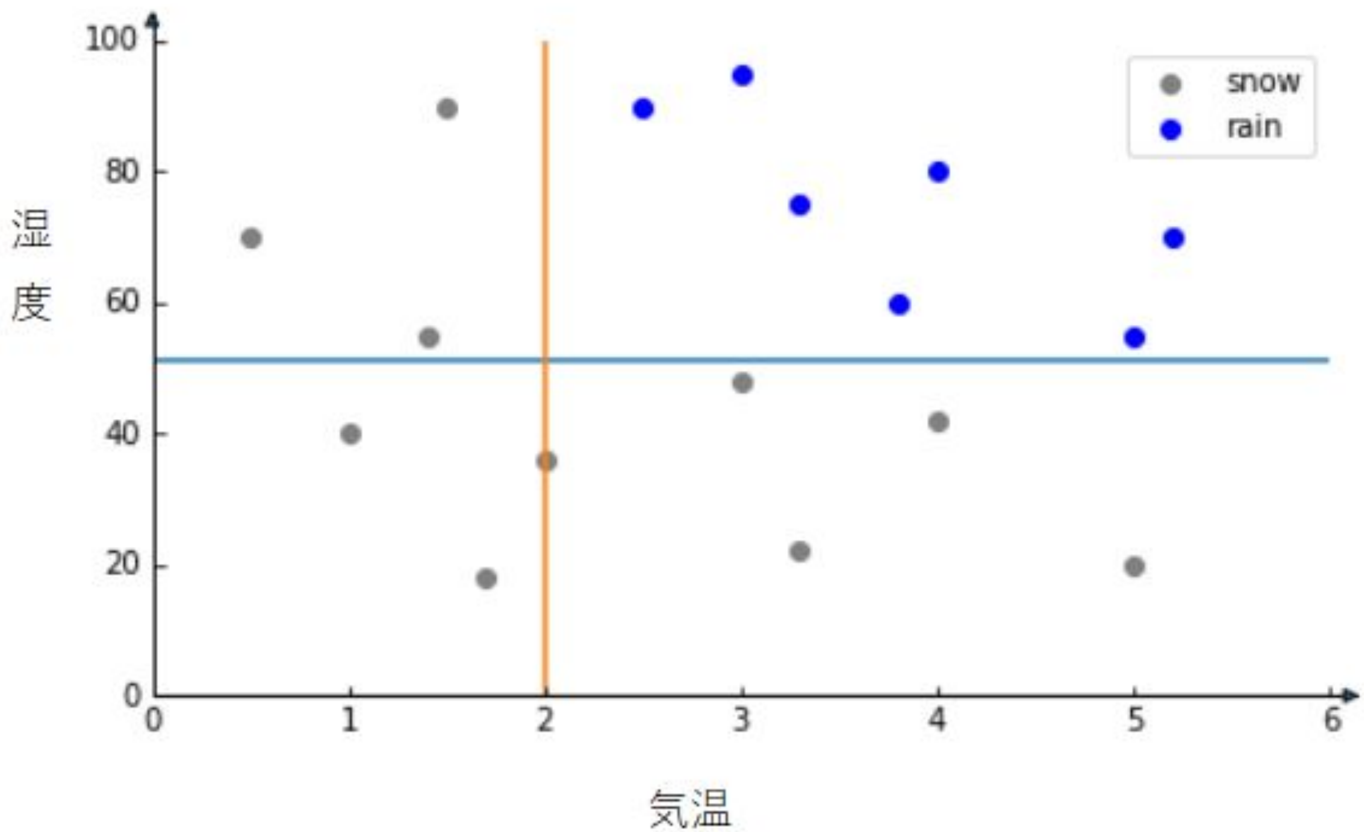
3.ニューラルネットワーク

表現力が高い



3.ニューラルネットワーク

特徴量設計が不要



3.ニューラルネットワーク

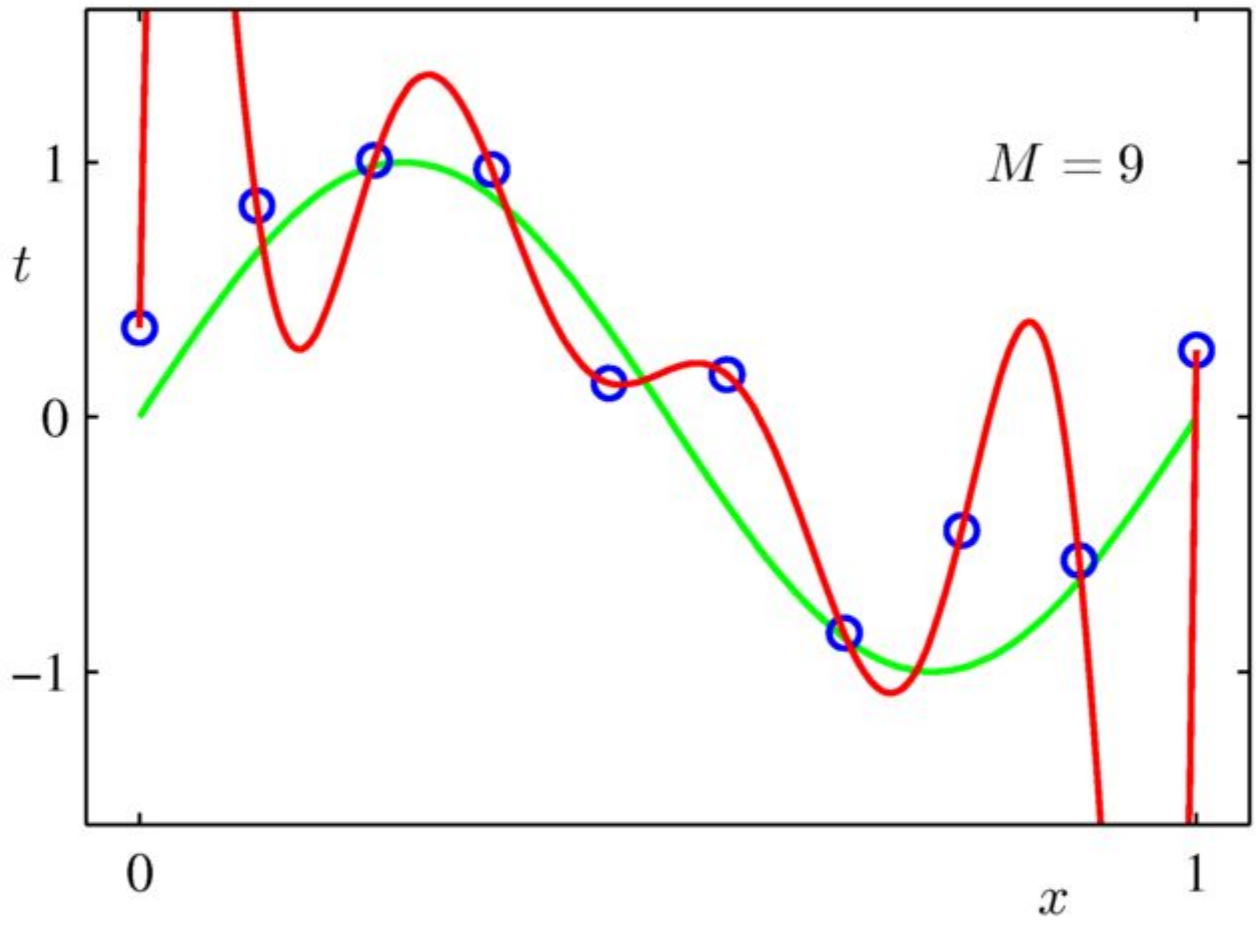
ニューラルネットワークの弱み

1．過学習しやすい

2．ブラックボックス

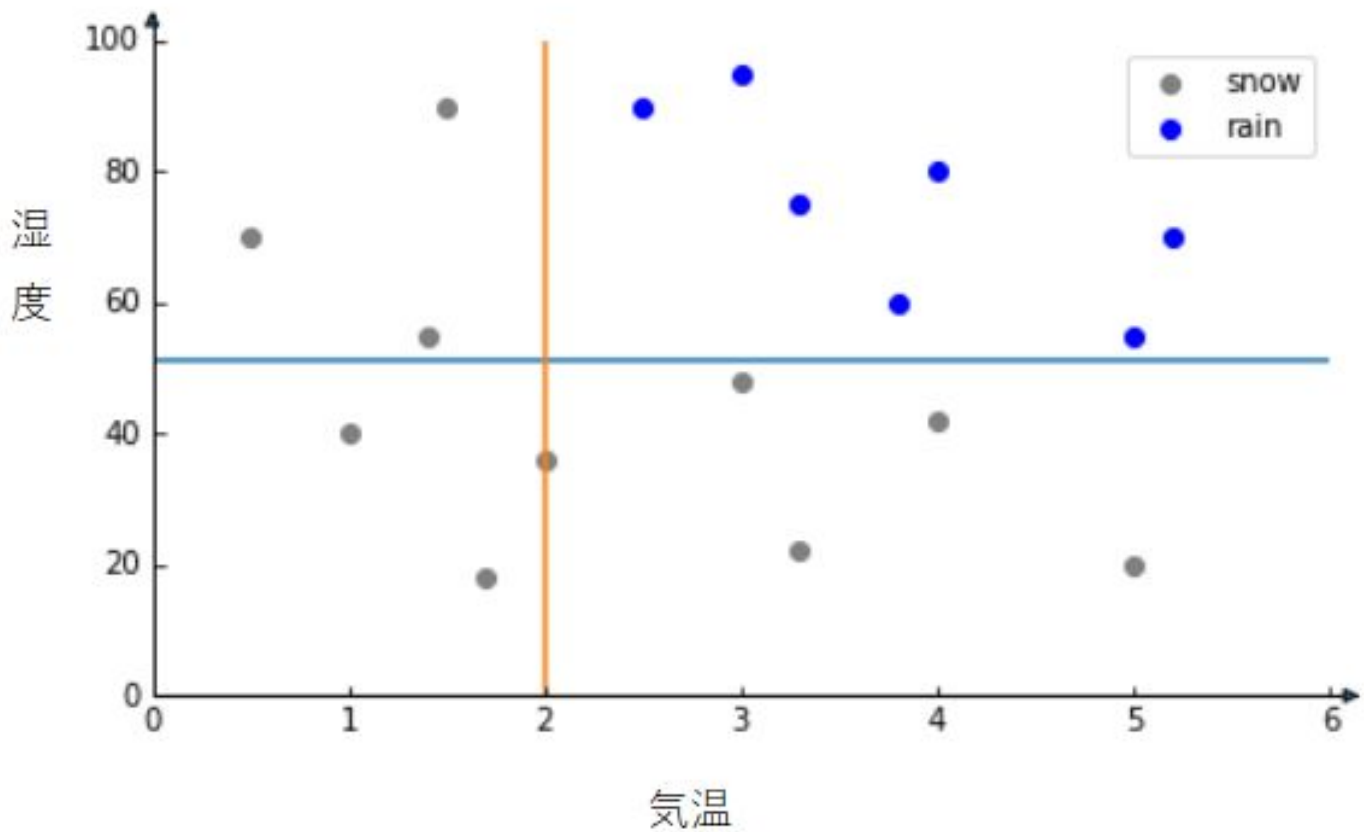
3.ニューラルネットワーク

過学習しやすい



3.ニューラルネットワーク

ブラックボックス



4.関数

関数

```
function double(x) {
    return x * 2;
}
```

関数

関数とは何かを渡したら、何かを返すもの

プログラムと違って数学の関数は

・数値しか渡せない

・数値しか返せない

関数

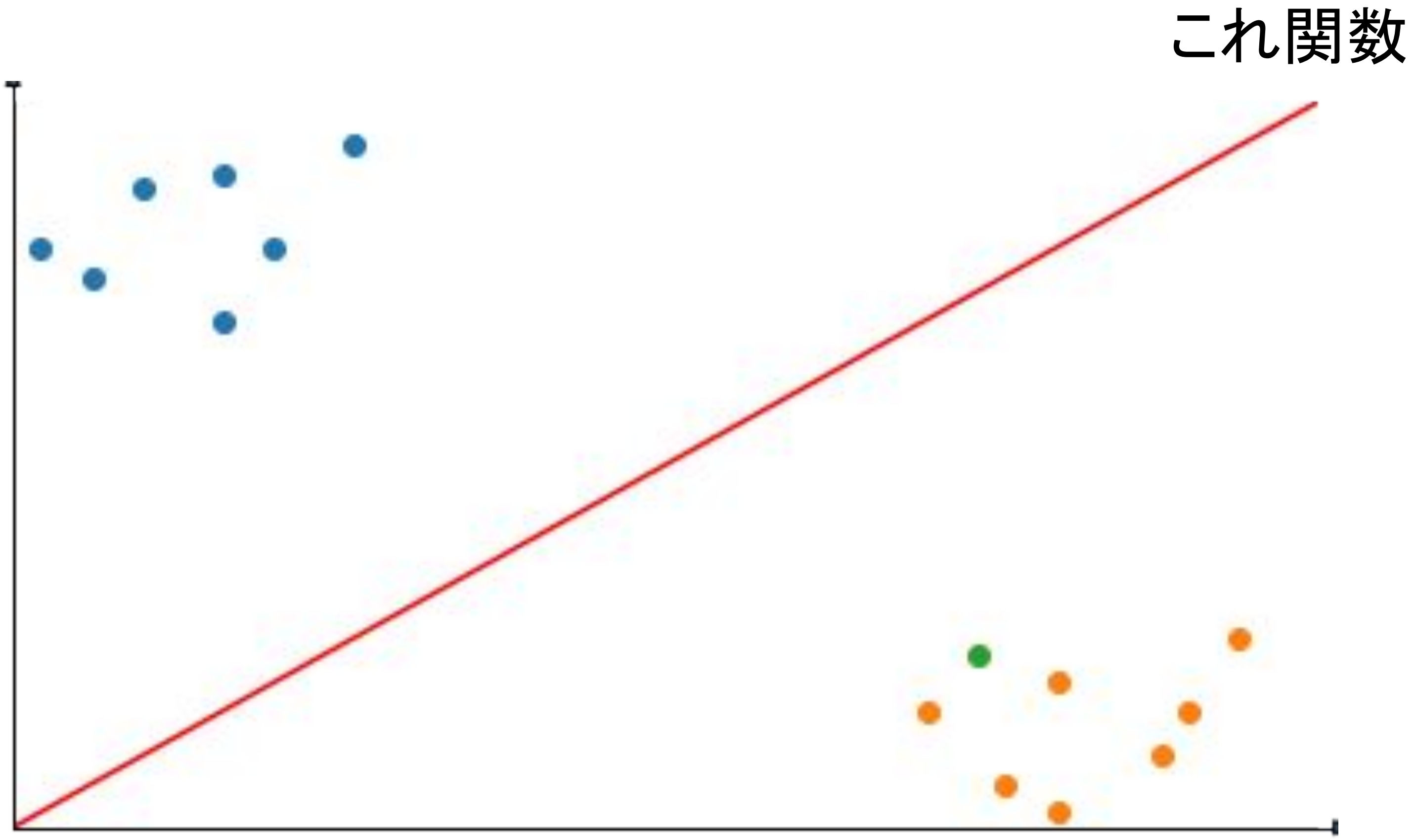
$$f(x) = x^2 + 2$$

関数

$$f(x) = x^2 + 2$$

```
function noname(x) {
    return x * x + 2;
}
```

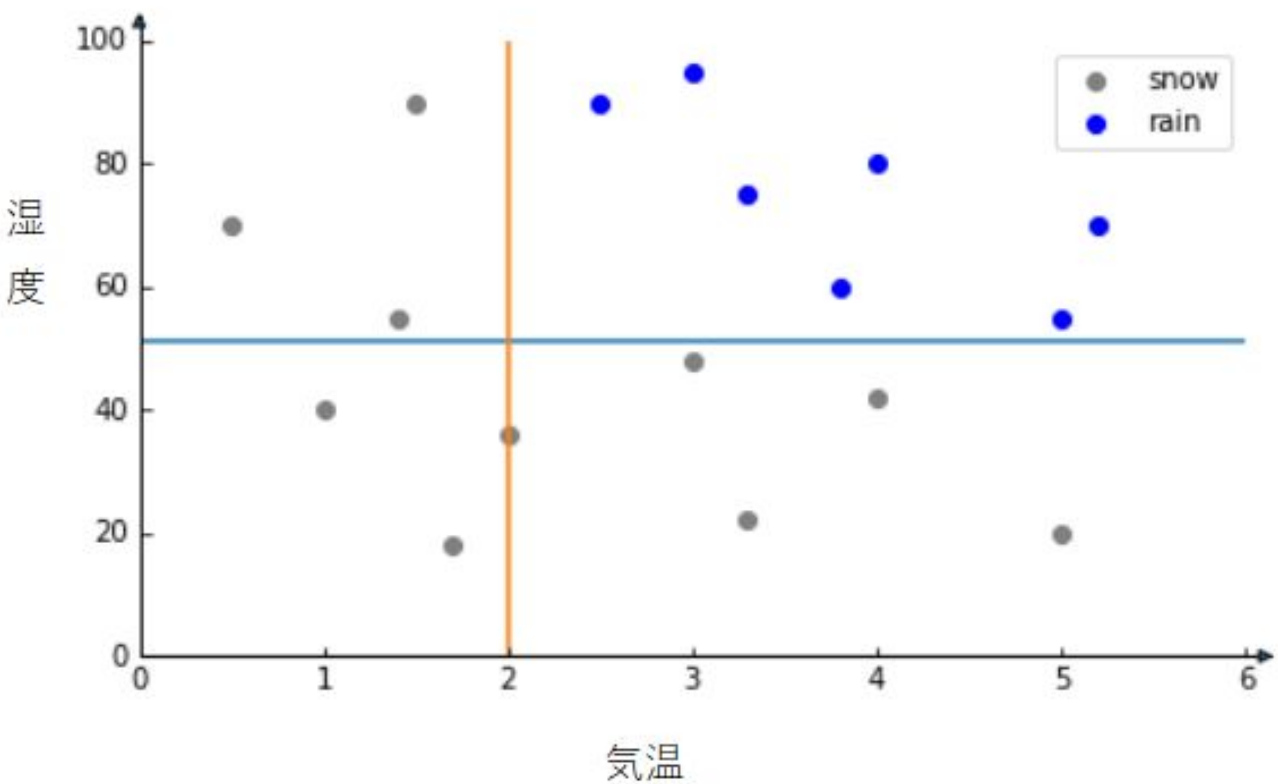
関数



関数

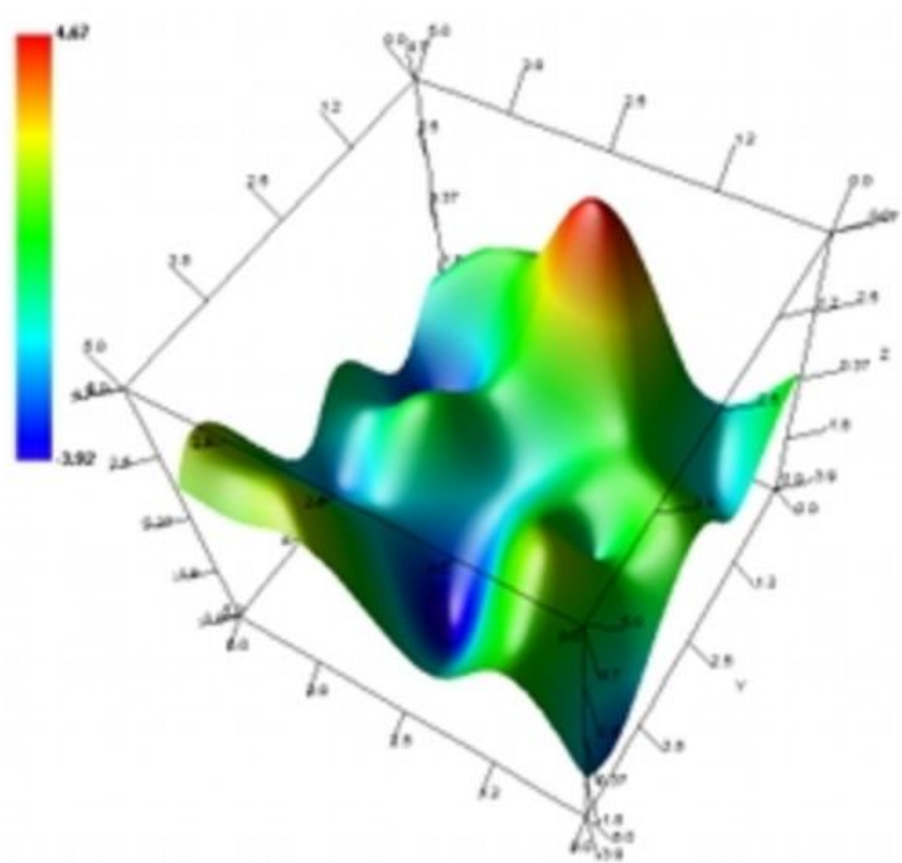
グラフとは関数を可視化したもの

可視化しないと回帰分類が何してるかわからん



関数

ちゃんと可視化できるのは3次元まで



関数

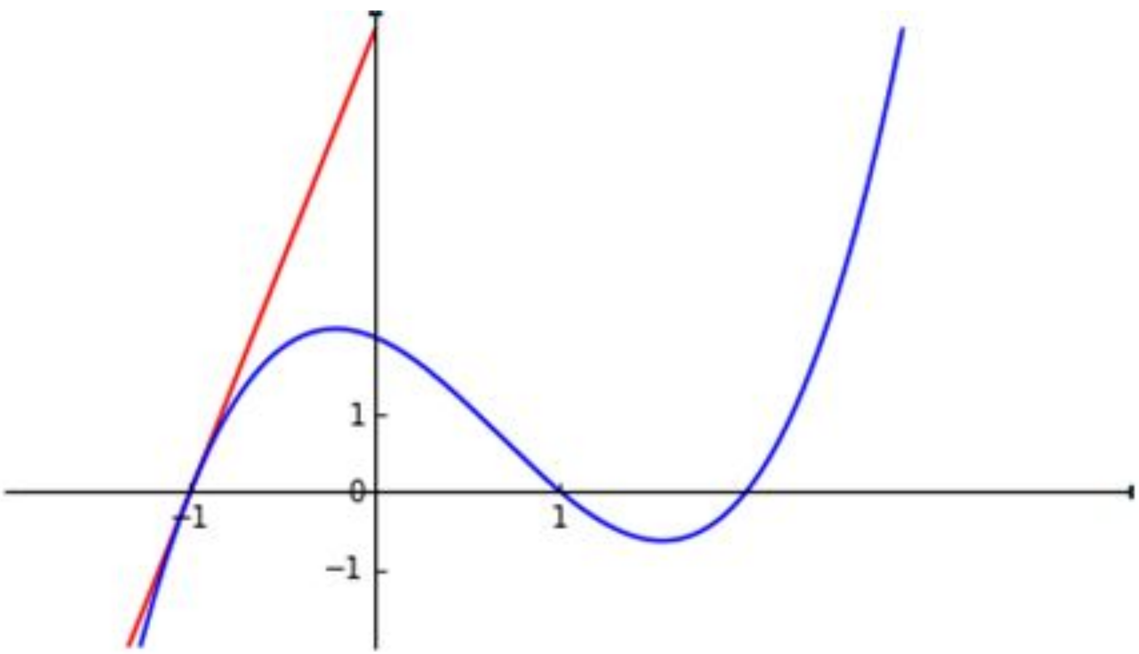
微分とは可視化できないグラフのごく狭い範囲でなんとなくどっちに傾いてるか求めるもの

よくわからん関数は微分して少しずつ解明する

5.微分

微分

ある曲線上の1点における接線の傾きを求める計算

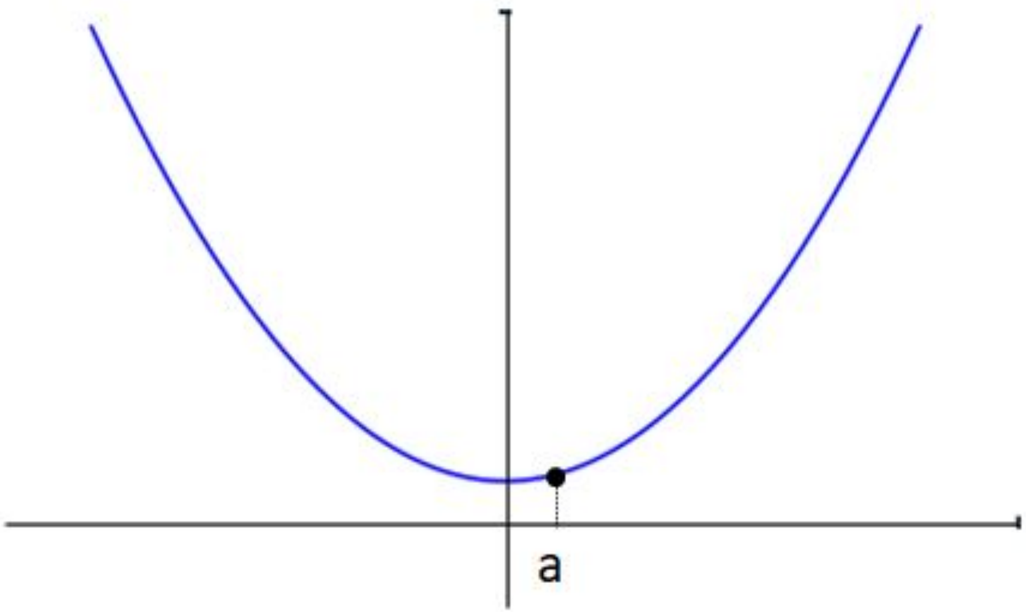


※接線の傾きは

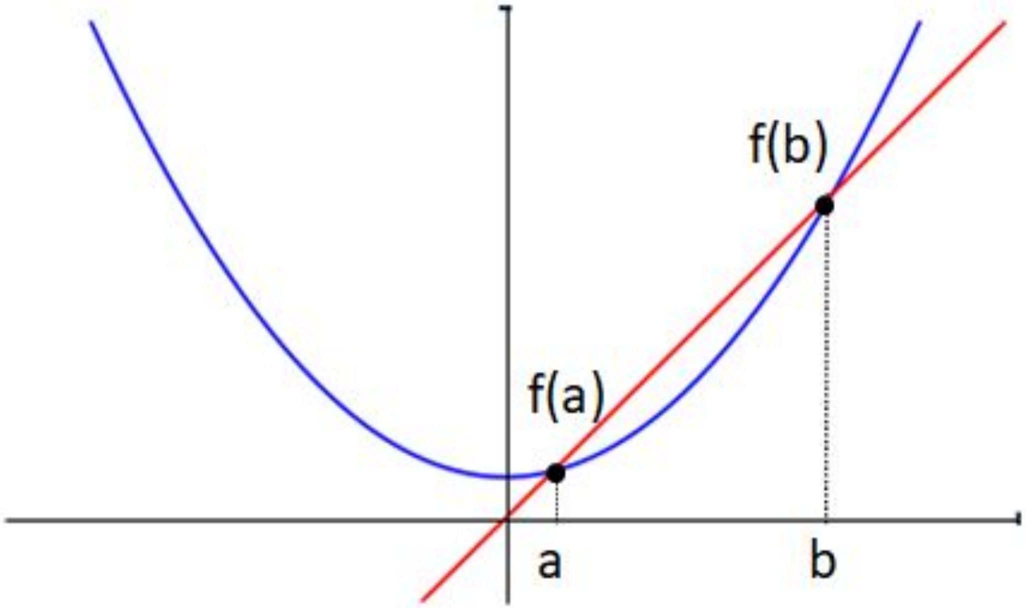
$$\frac{yの変化量}{xの変化量}$$

微分

次の関数$y = f(x)$において、$x = a$の接線の傾きを考える

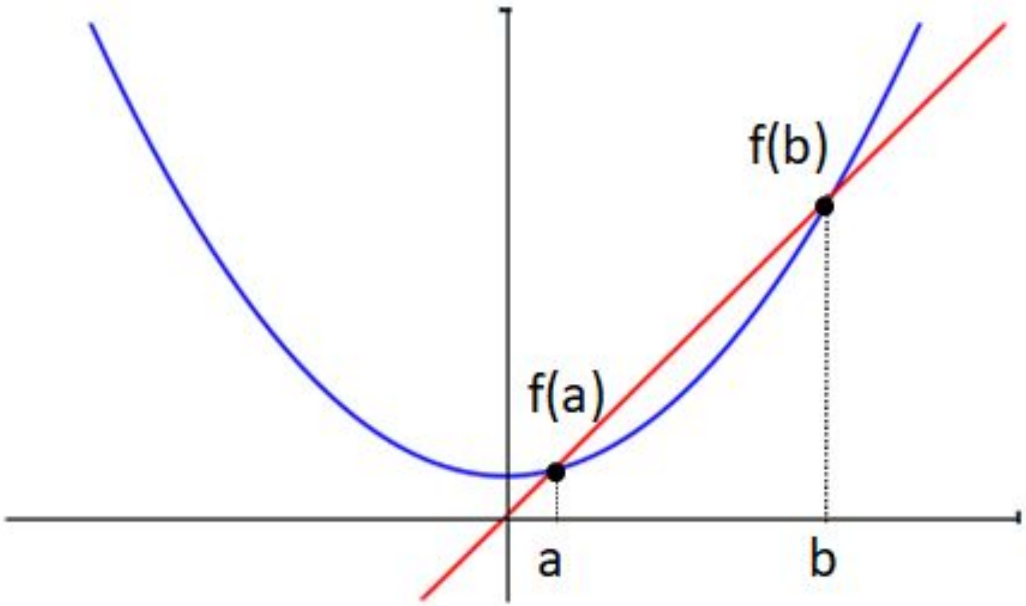


微分



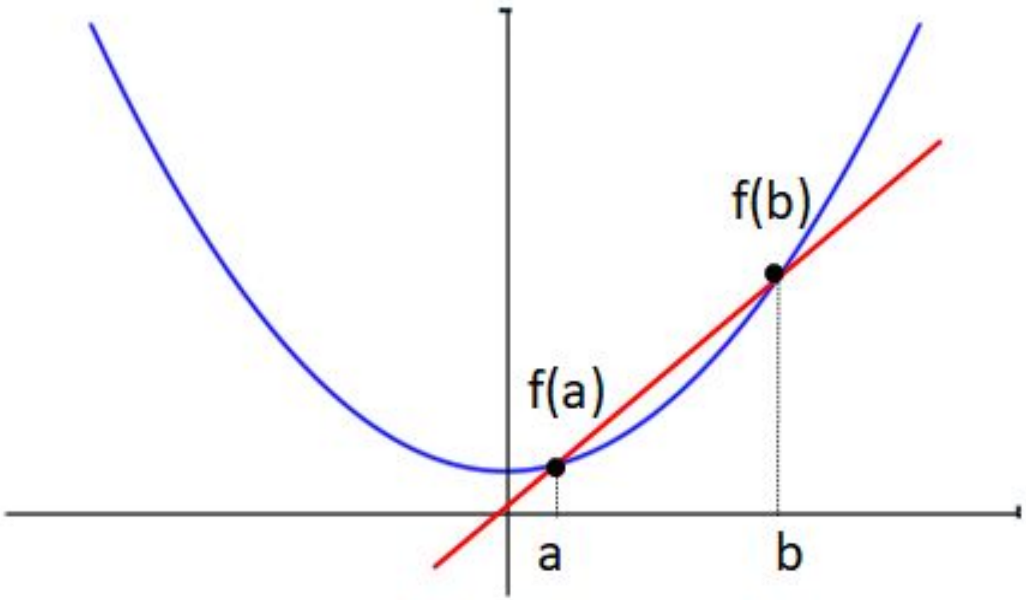
微分

この直線の傾きは $\dfrac{f(b)-f(a)}{b-a}$



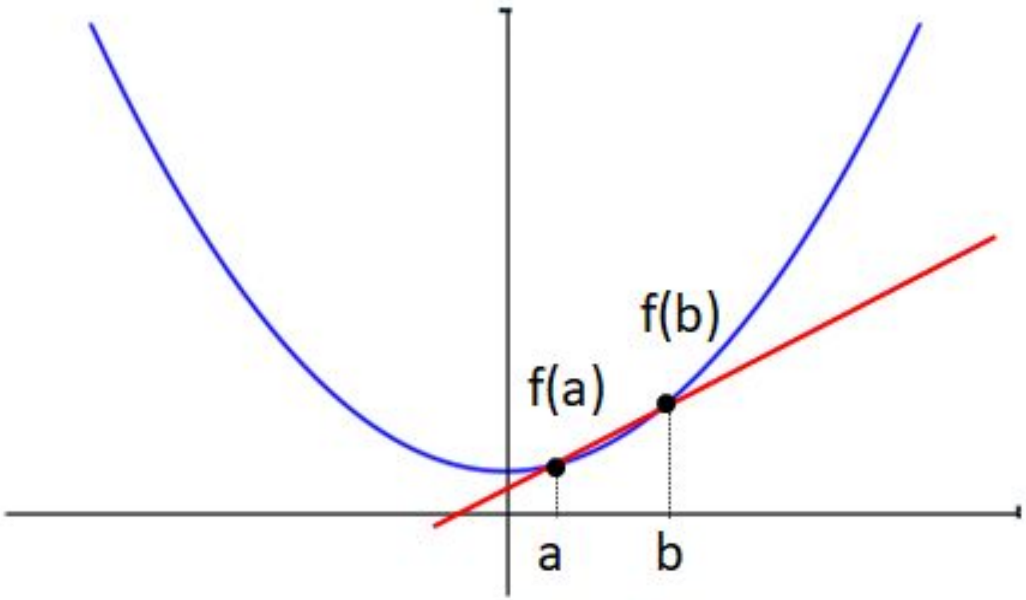
微分

bをaに近づける



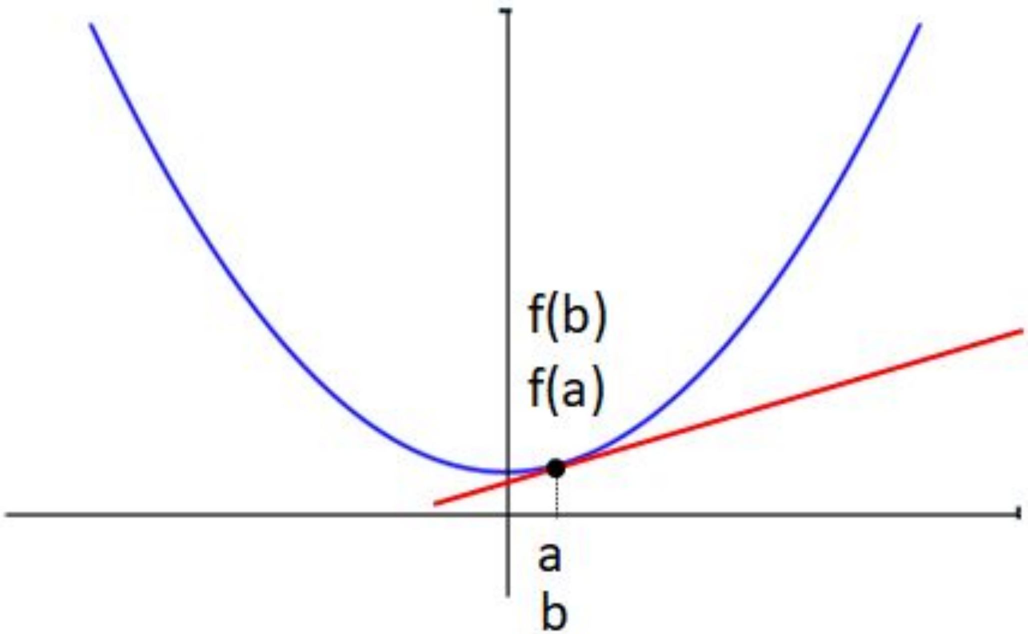
微分

bをaに近づける



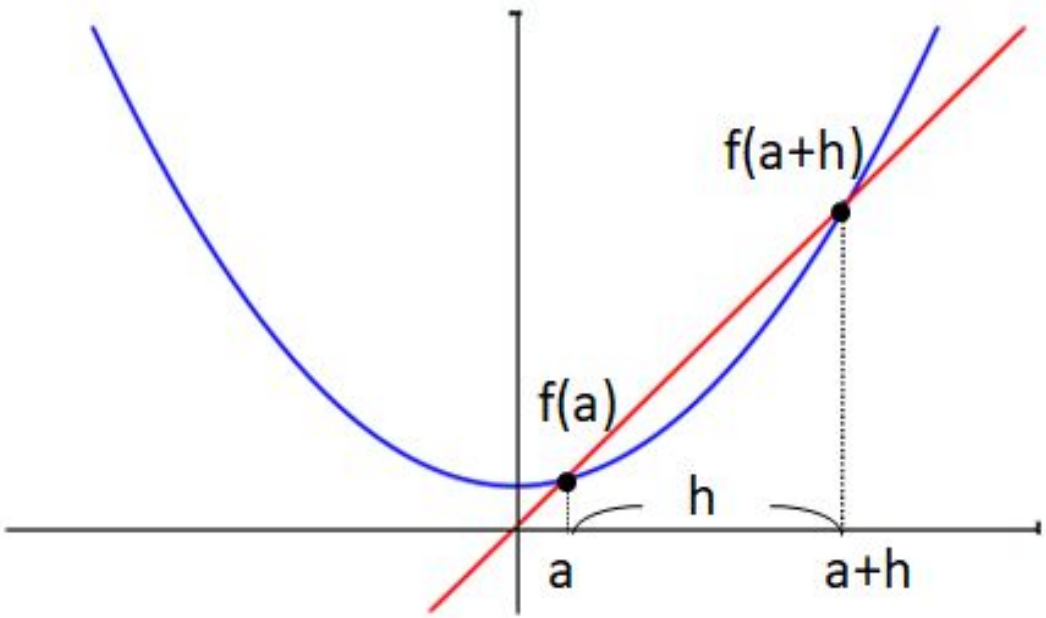
微分

bをaに近づける



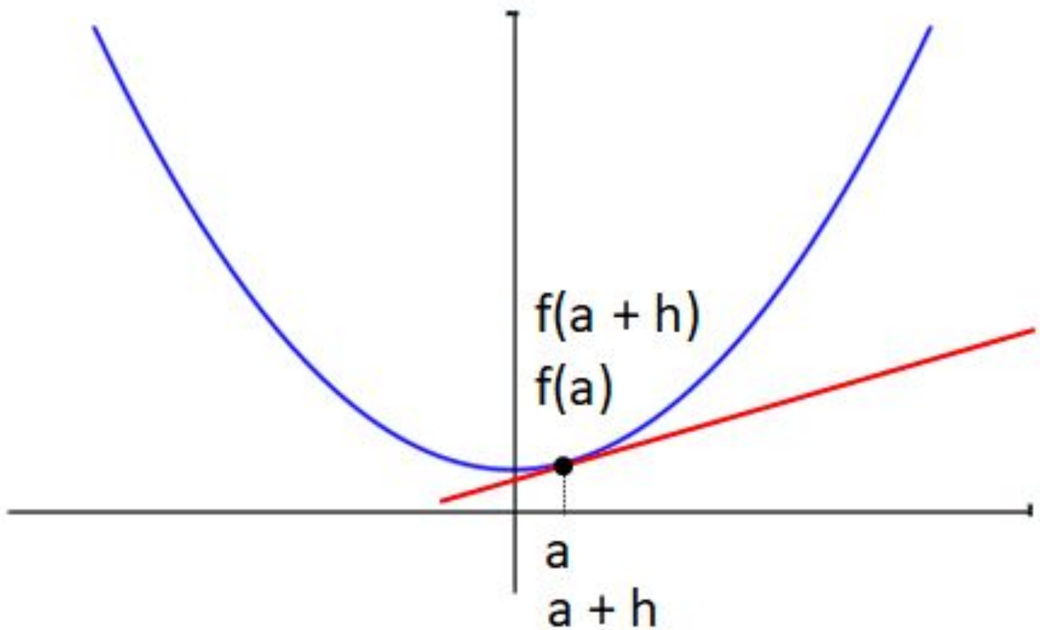
微分

$\dfrac{f(b)-f(a)}{b-a}$ のb-aをhと置くと、直線の傾きは $\dfrac{f(a+h)-f(a)}{h}$



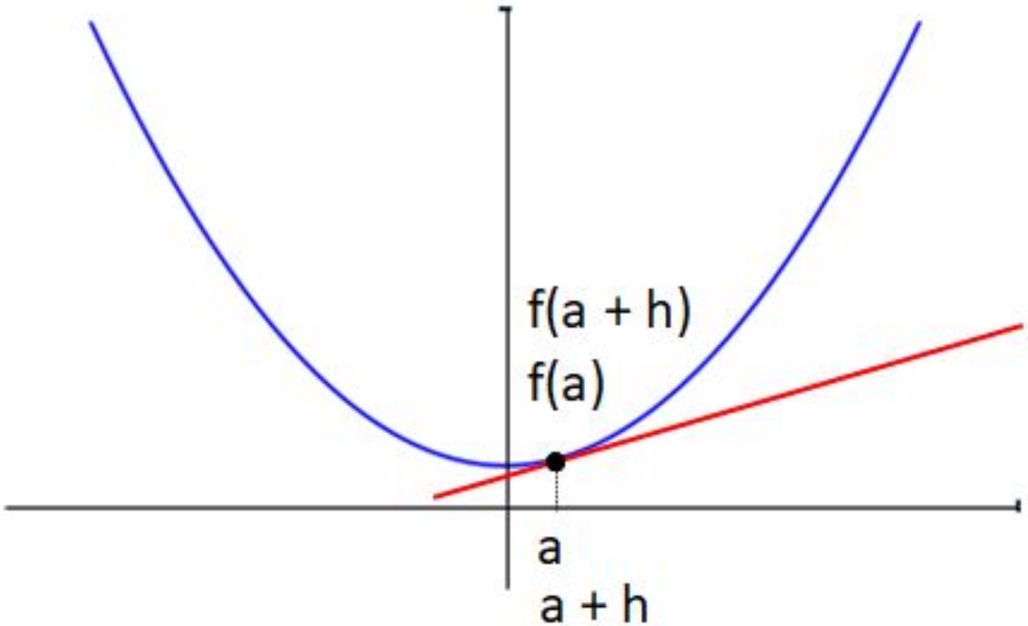
微分

$$\frac{f(a+h)-f(a)}{h}$$ のhを極限まで0に近づける



微分

微分係数： $f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$



微分

limitがよくわからんかったら

hを0.000000001（無視できるぐらい小さい数字）

にすると思えばOK

とにかくhを小さくしたいだけ

微分

$f(x) = x^2 + x + 1$ をx = 1において微分する

$$\frac{df(a)}{dx} = \frac{f(a+h) - f(a)}{h}$$ より

（本当はlimitつけないといけないです）

$$\frac{df(1)}{dx} = \frac{f(1+h) - f(1)}{h}$$

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

$$= \frac{h^2 + 3h}{h} = h + 3 = 3$$

微分

x2乗の2を前に出してマイナス1とか覚えなくてOK

あれは微分の計算を簡単にするためのもの

私たちはプログラマーなので難しい計算は

プログラムにさせる

微分

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$ を実装

```
def numerical_diff(f, x):
    h = 0.000000001
    return (f(x + h) - f(x)) / h

def func(x):
    return x**2 + x + 1

result = numerical_diff(func, 1)
print(result) # 3.000000248221113
```

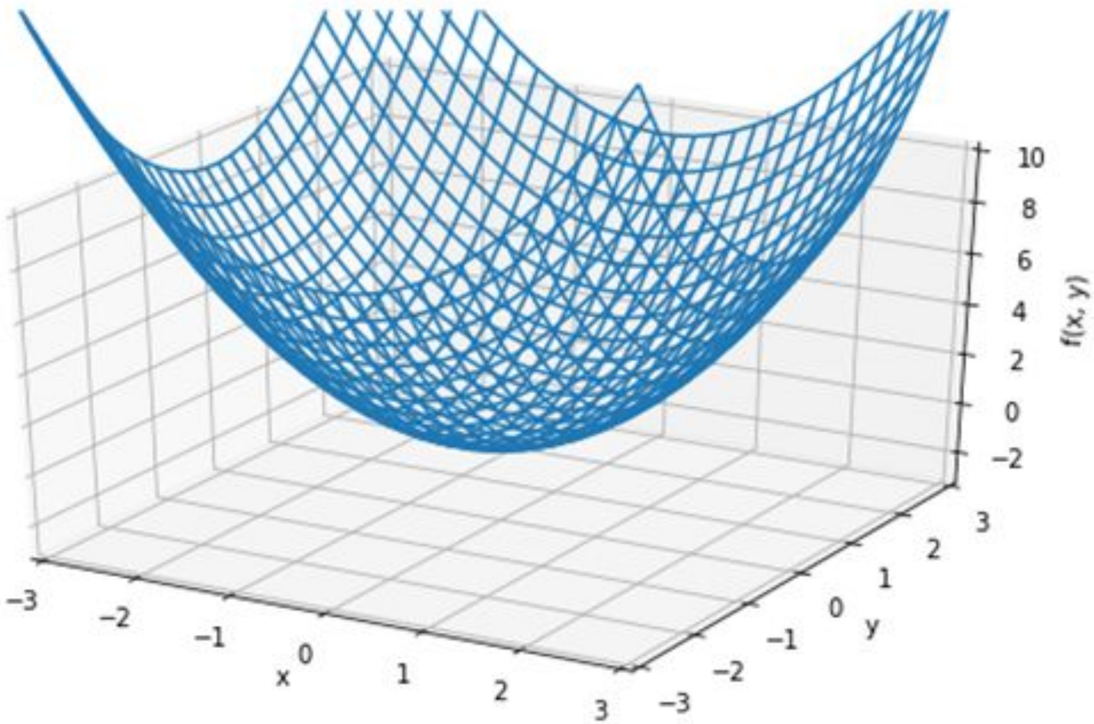
6.偏微分

偏微分

変数が複数ある場合の微分

一つの変数ずつ微分する

$f(x, y) = x^2 + y^2$



偏微分

x軸方向への微分

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h \to 0} \frac{f(x+h,y) - f(x,y)}{h}$$

y軸方向への微分

$$\frac{\partial f(x,y)}{\partial y} = \lim_{h \to 0} \frac{f(x,y+h) - f(x,y)}{h}$$

偏微分

$f(x, y) = x^2 + y^2$ の（x,y）＝（1,2）における勾配を求める

```
def numerical_grad(f, x, y):
    h = 0.000000001
    grad1 = (f(x + h, y) - f(x, y)) / h
    grad2 = (f(x, y + h) - f(x, y)) / h
    return (grad1, grad2)

def func(x, y):
    return x**2 + y**2

result = numerical_grad(func, 1, 2)
print(result) # (2.000000165480742, 4.000000330961484)
```