9-1 第９回の内容

（人工知能の基本）

URL: https://www.kkaneko.jp/db/mi/index.html

金子邦彦

第９回の内容

- **ニューラルネットワーク**の**学習**について，その基礎を知る
- **ニューラルネットワーク**の**学習**について，**学習不足**や**過学習**を知る

【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．前回の資料，今回の資料を復習しておく．

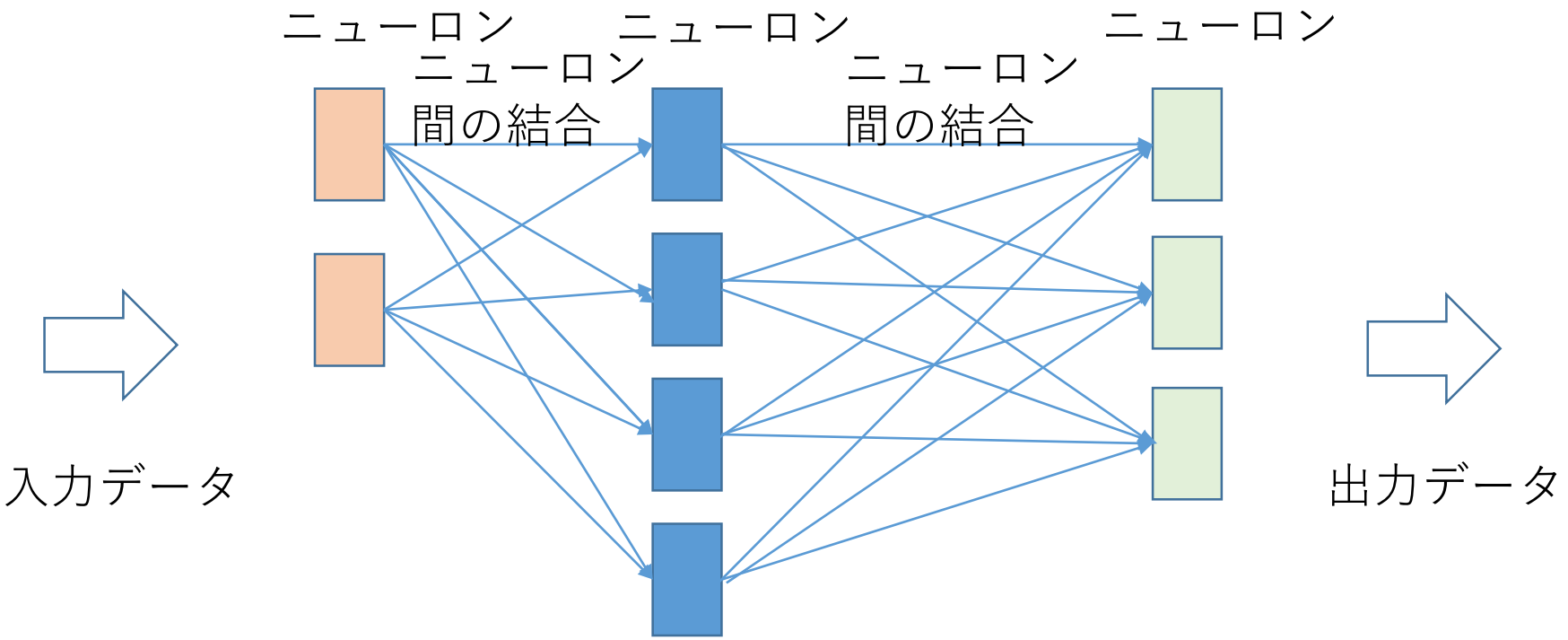
9-2 ニューラルネットワークを用いた分類

（人工知能の基本）

URL: https://www.kkaneko.jp/db/mi/index.html

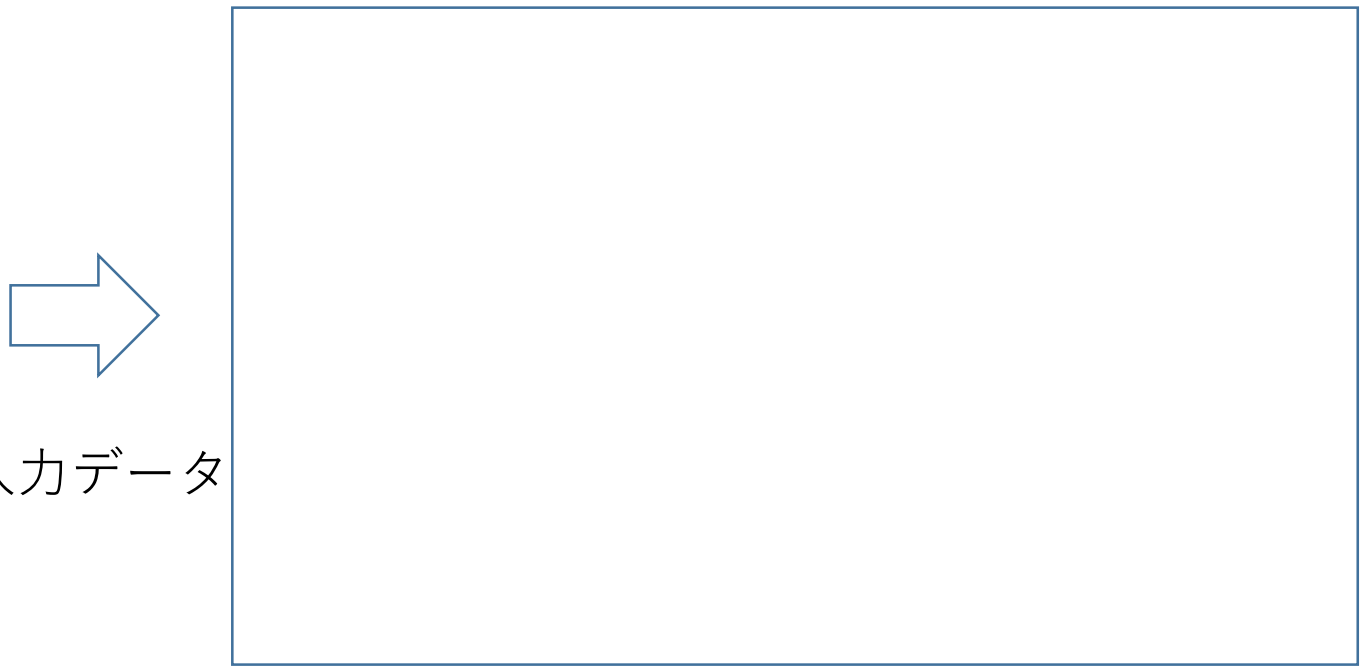
金子邦彦

層が直列になっているニューラルネットワーク

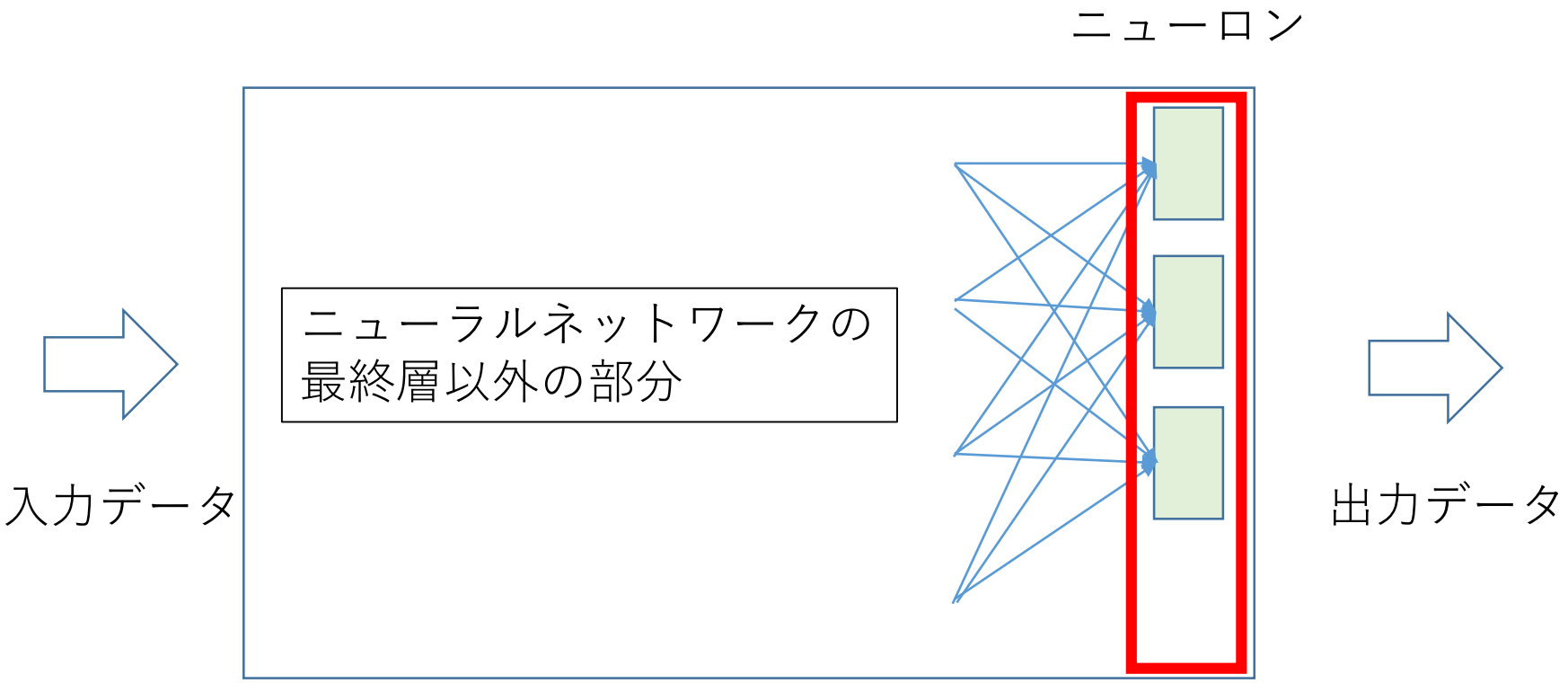


データは入力から出力の方向へ

３種類の中から１つに分類する場合

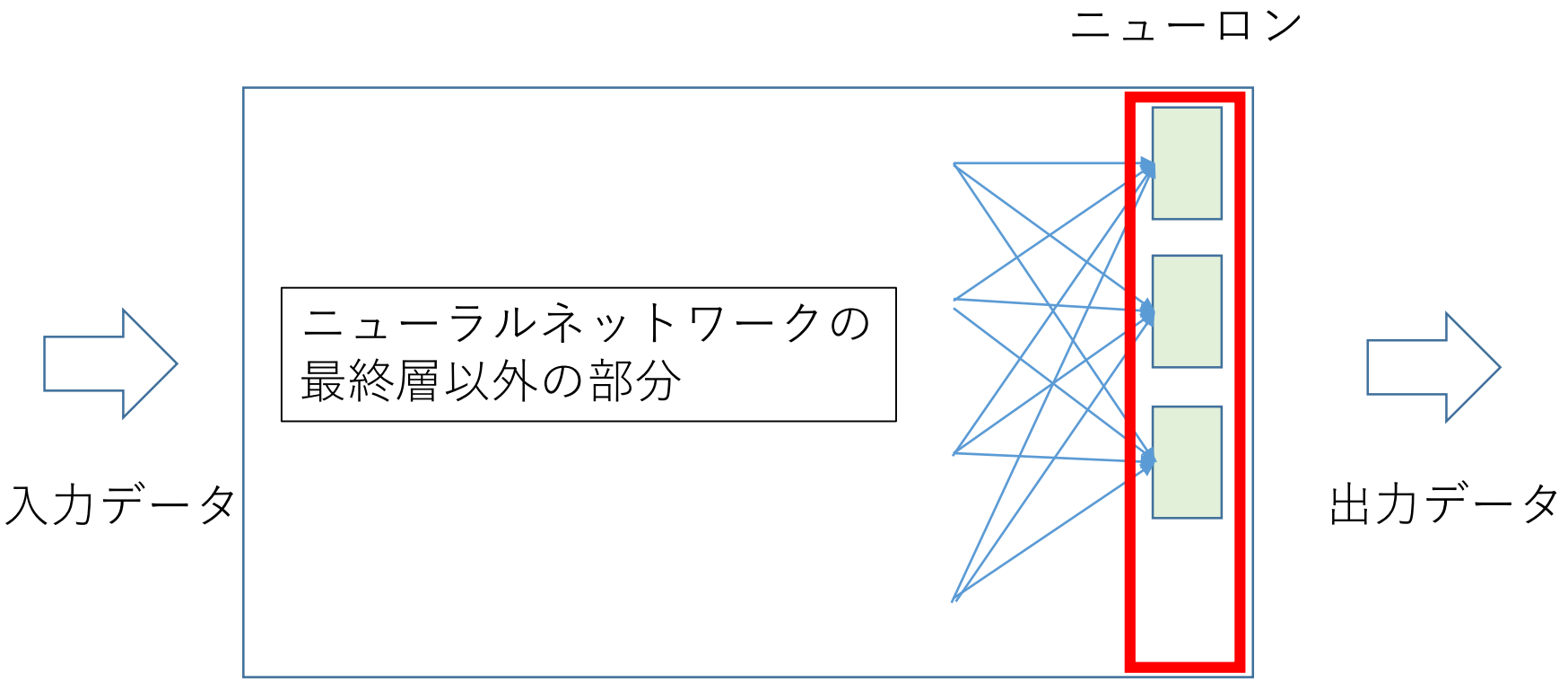


３種類の中から１つに分類する場合

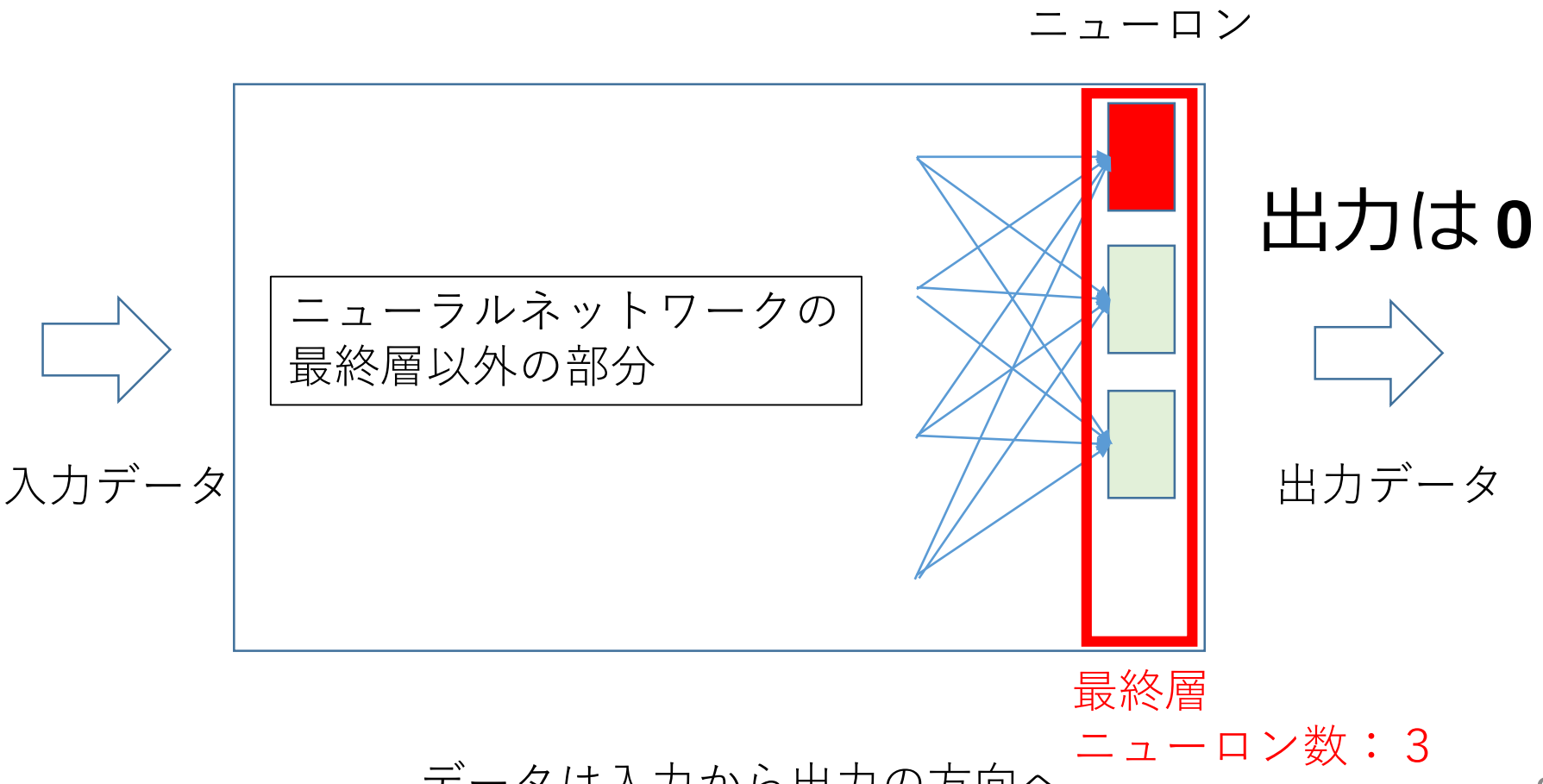


データは入力から出力の方向へ

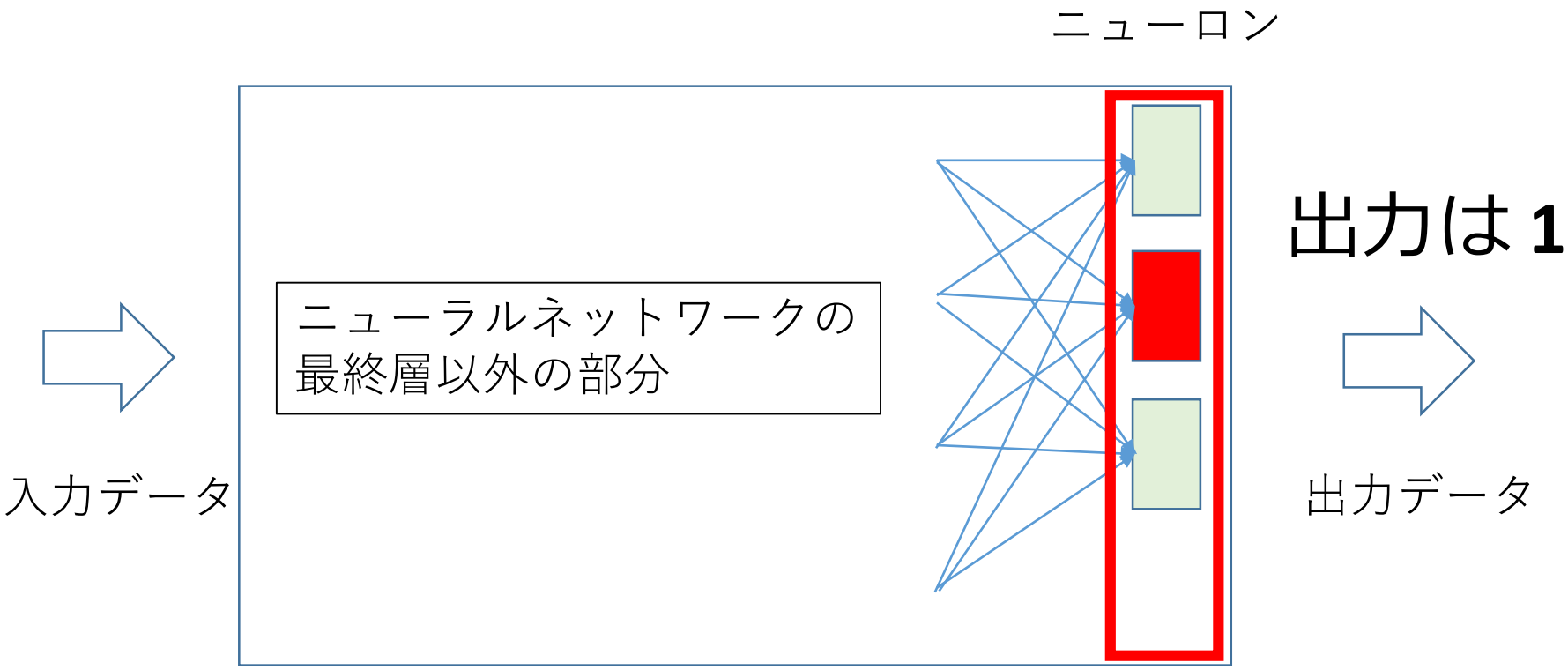
最終層について，１つが強く
活性化するように調整



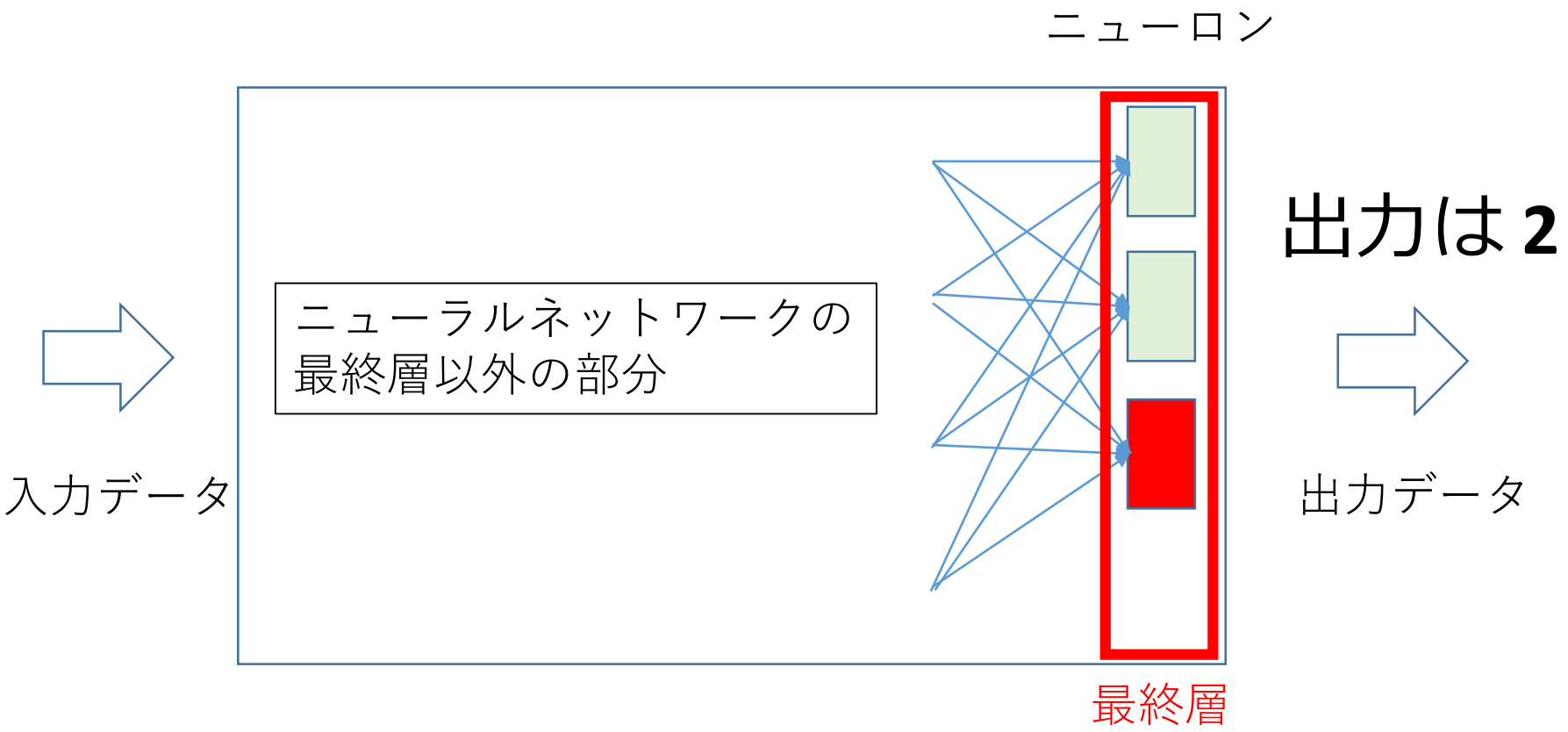
最終層について，１つが強く
活性化するように調整



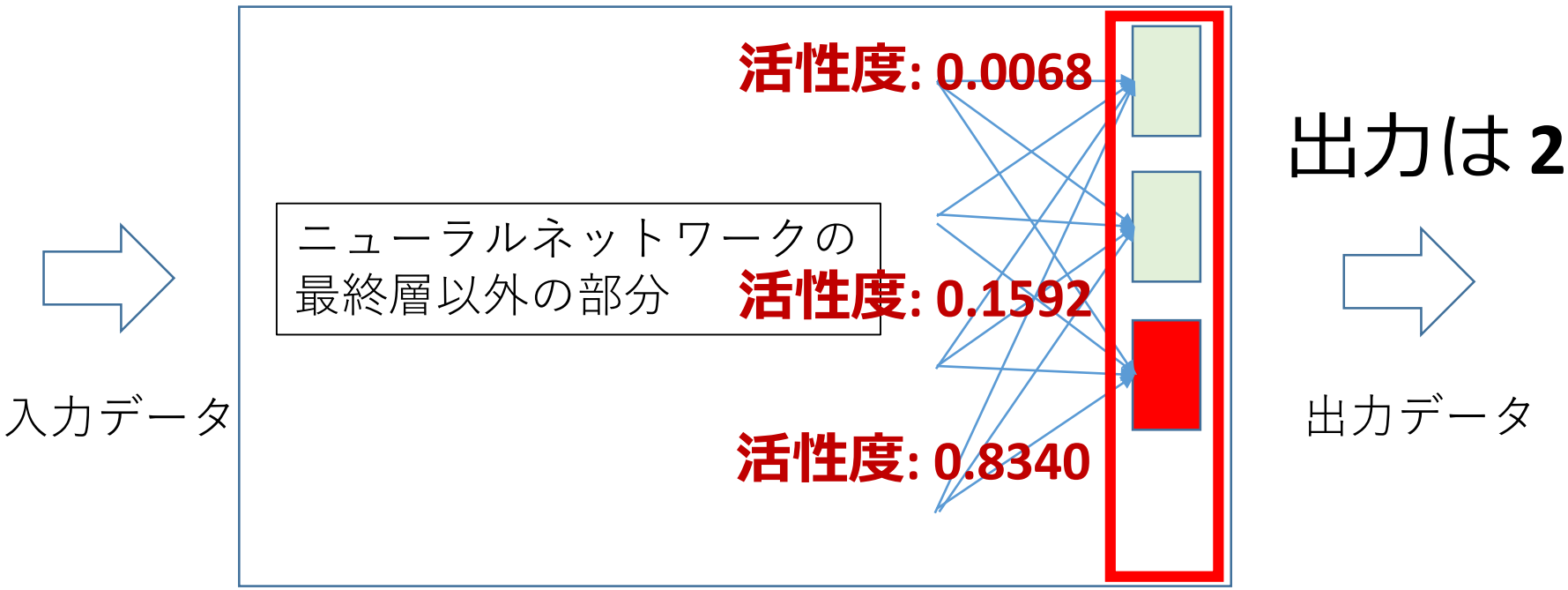
最終層について，１つが強く
活性化するように調整



最終層について，１つが強く
活性化するように調整



最終層について，１つが強く
活性化するように調整



実際には，活性度は**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

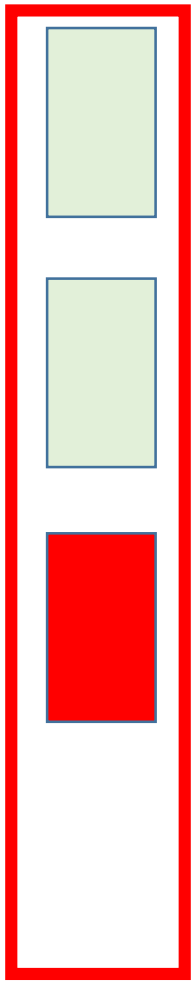
正解と誤差

正解は 2
であるとする

活性度: 0.0068

活性度: 0.1592

活性度: 0.8340



誤差: 0.0068 ⟷ **あるべき値: 0**

誤差: 0.1592 ⟷ **あるべき値: 0**

誤差: - 0.1760 ⟷ **あるべき値: 1**

誤差をもとに、結合の重みを自動調節

ニューラルネットワークを用いた分類

ニューラルネットワークを分類に使うとき

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

- そこには**誤差**がある

9-3 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/db/mi/index.html

金子邦彦

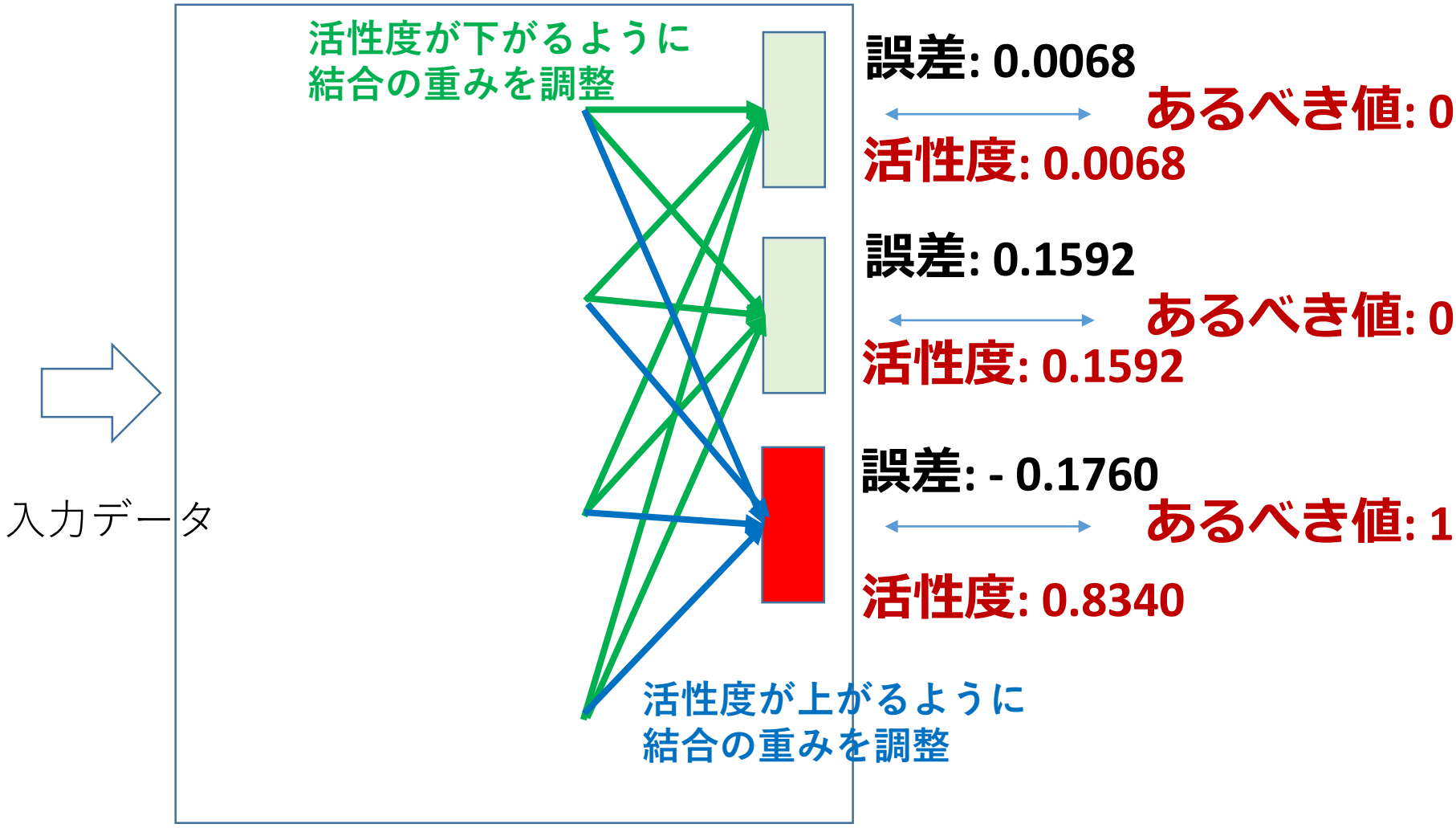
機械学習

- **機械学習**では，データによる**学習**を行う
- **学習**に用いるデータのことを，**教師データ**などという
- **学習を重ねることで上達する**

- 「**学習**によって，**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している」という考え方もある

ニューラルネットワークの学習

正解は 2
であるとする



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

9-4 最適化

（人工知能の基本）

URL: https://www.kkaneko.jp/db/mi/index.html

金子邦彦

最適化

- **最適化**は,

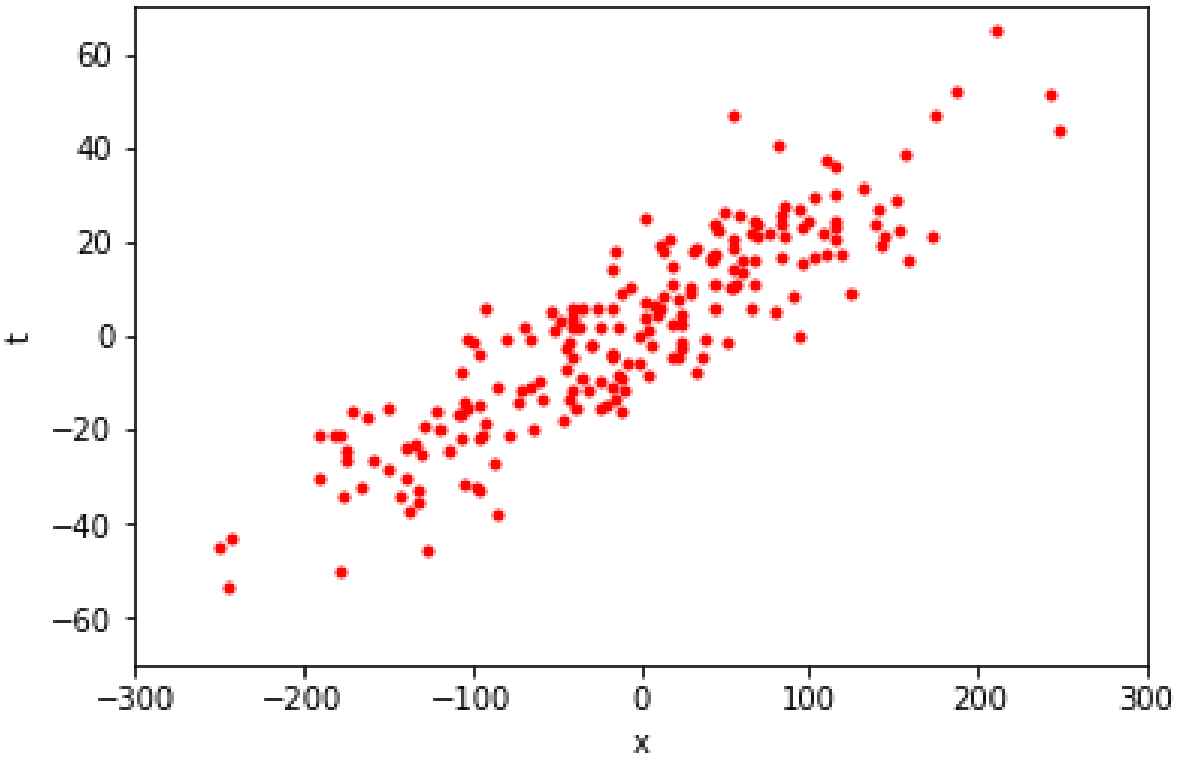
パラメータを調整して、

ある尺度での**値**を**最適**にするように,

調整を行うこと

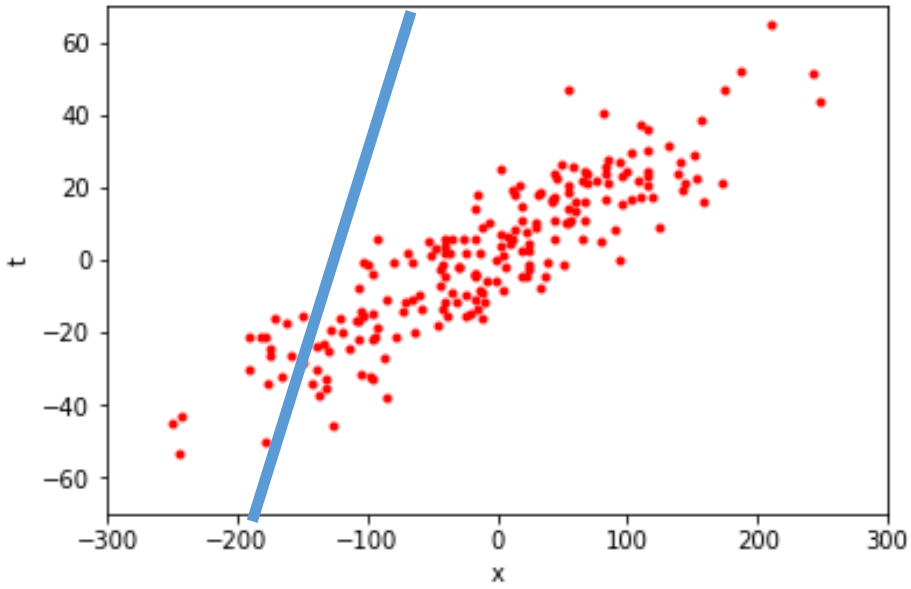
- **誤差を自動で最小化したい**ときに有効な技術

データの例

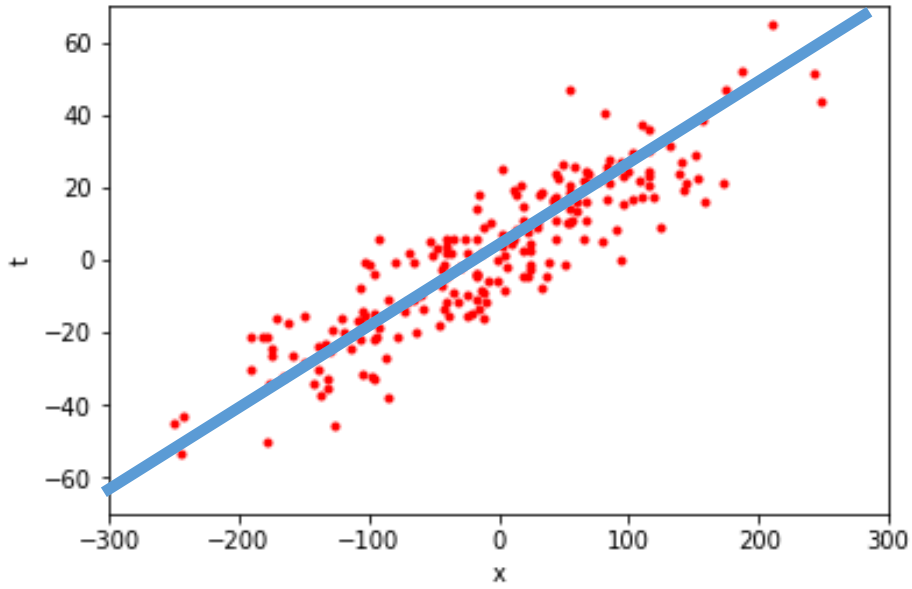


- **多数のデータの集まり**
- 上の図では，点１つで，１つのデータ

直線による近似の例

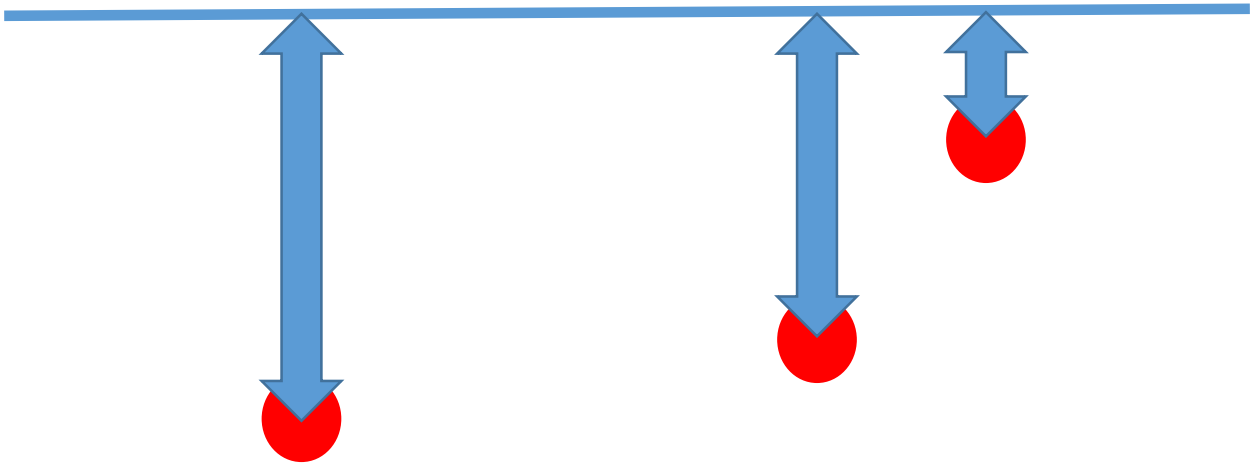


最適でない近似
（誤差大）



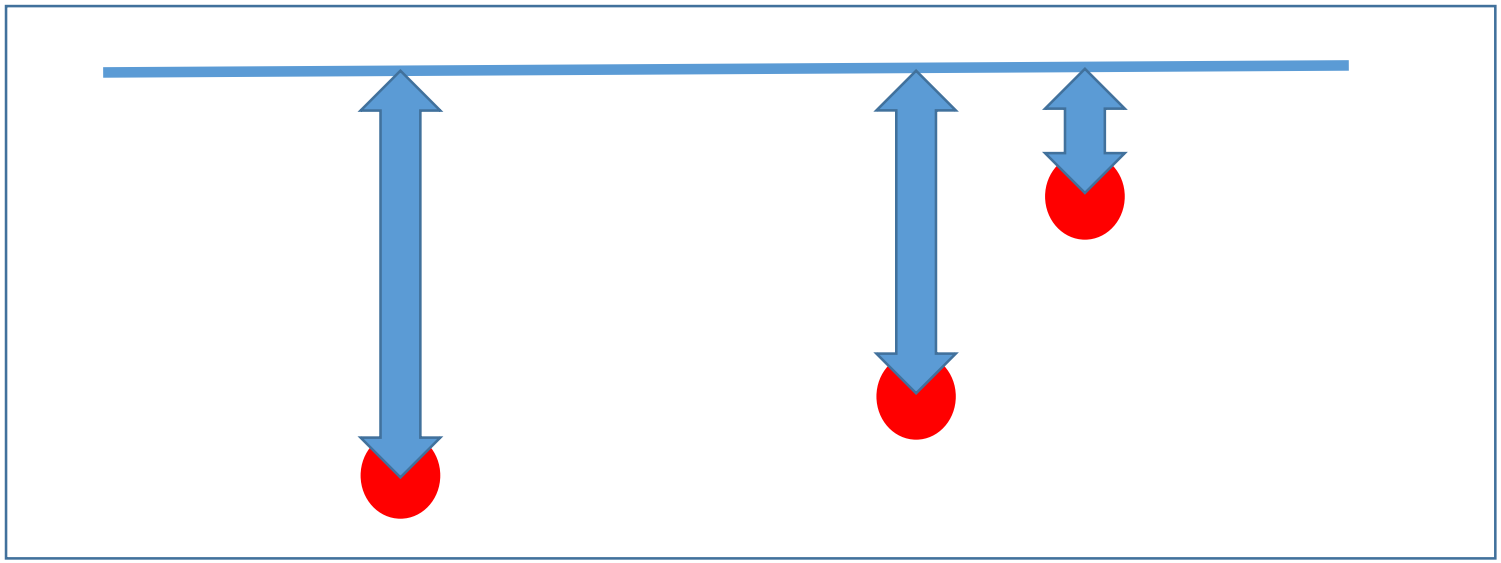
最適な近似
（誤差小）

誤差

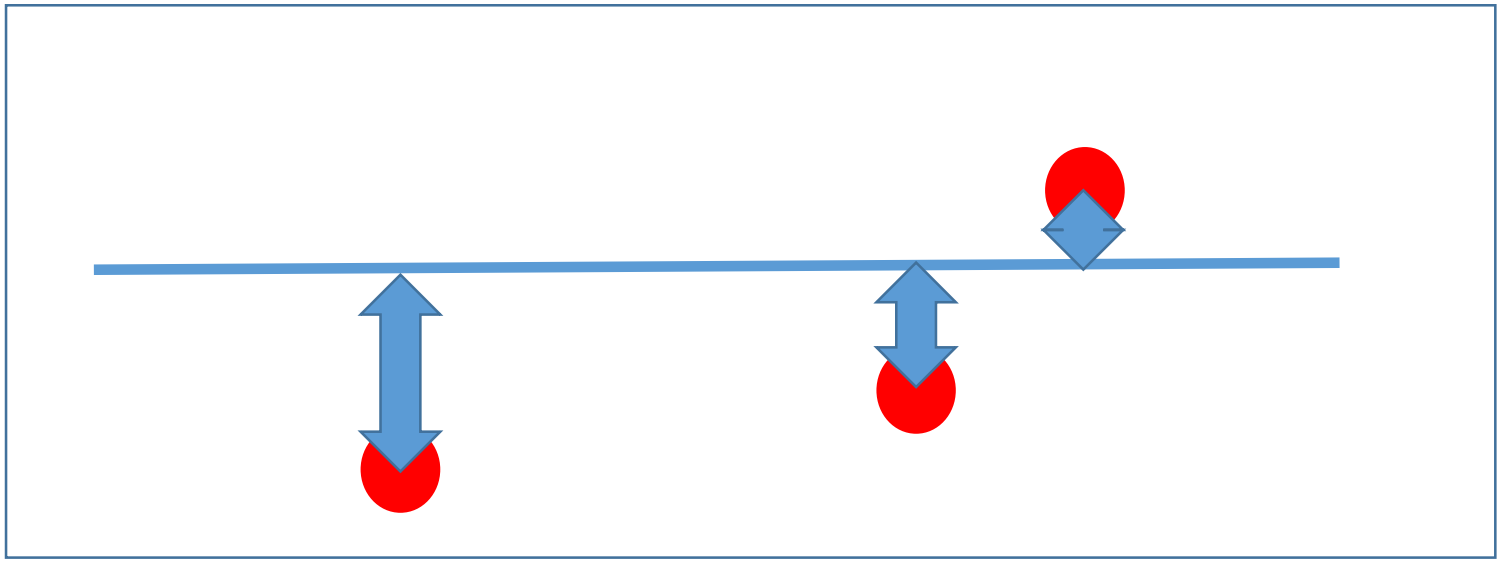


赤点：元データ

直線の上下移動による誤算の変化



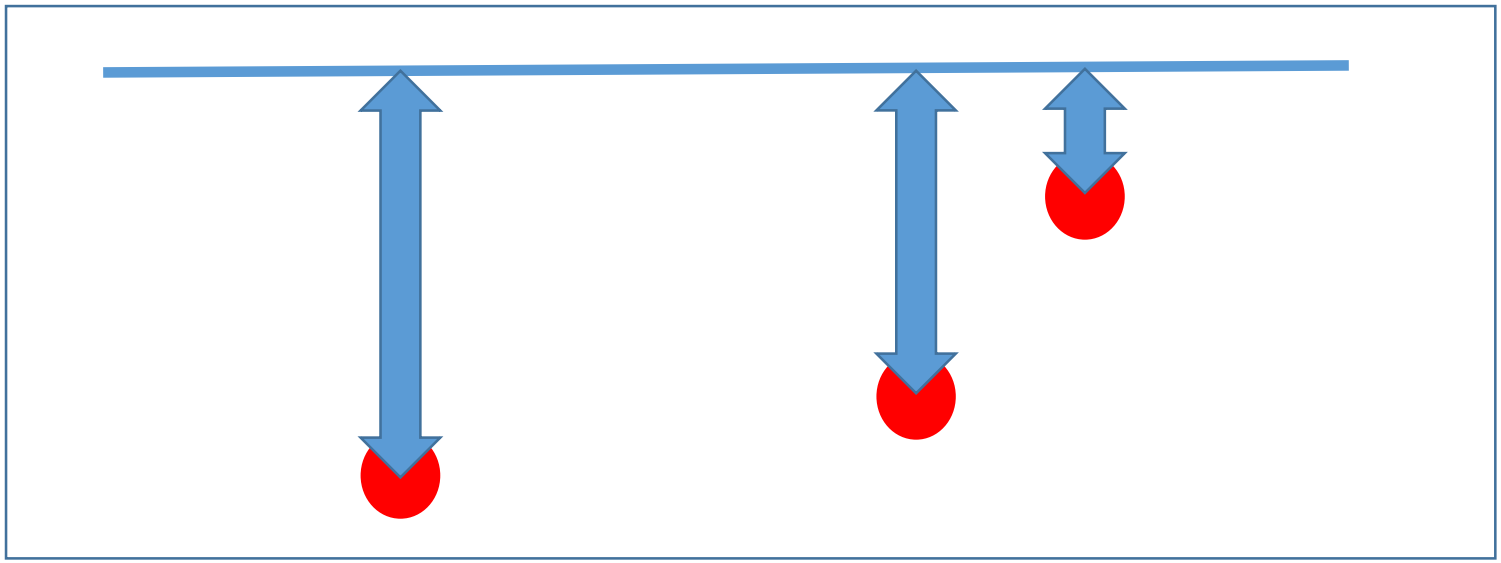
誤差大



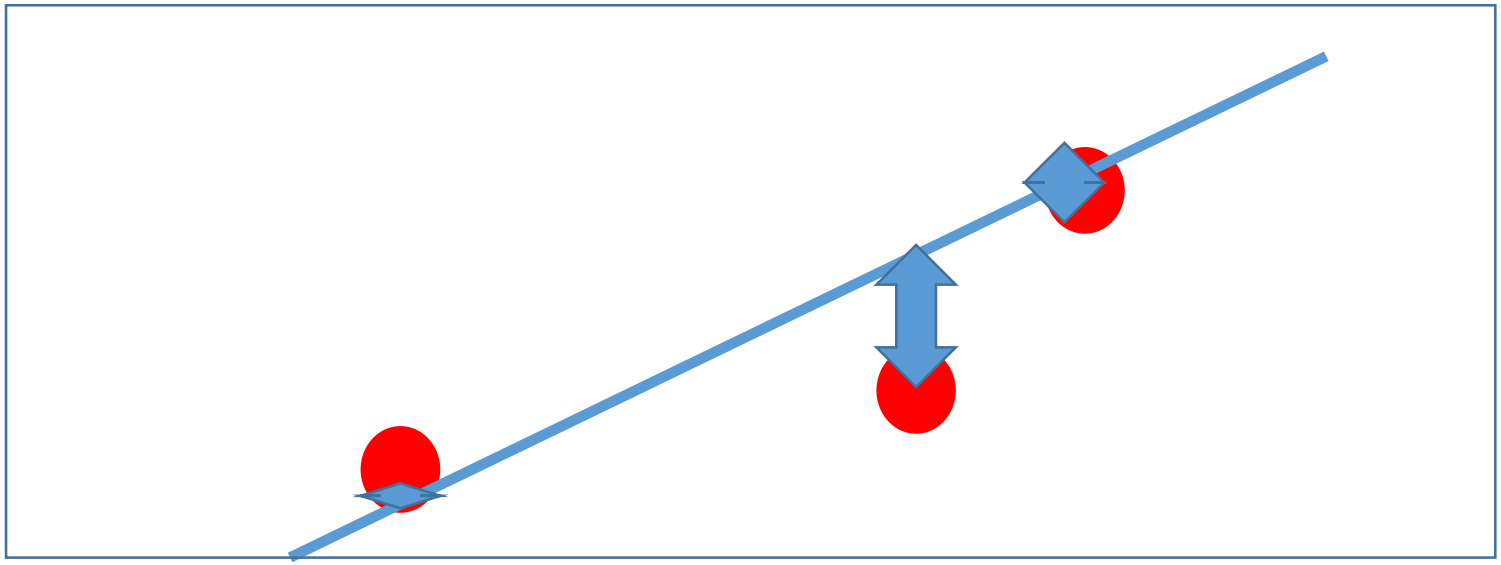
誤差小

赤点：元データ

直線の傾きの変化による誤算の変化



誤差大

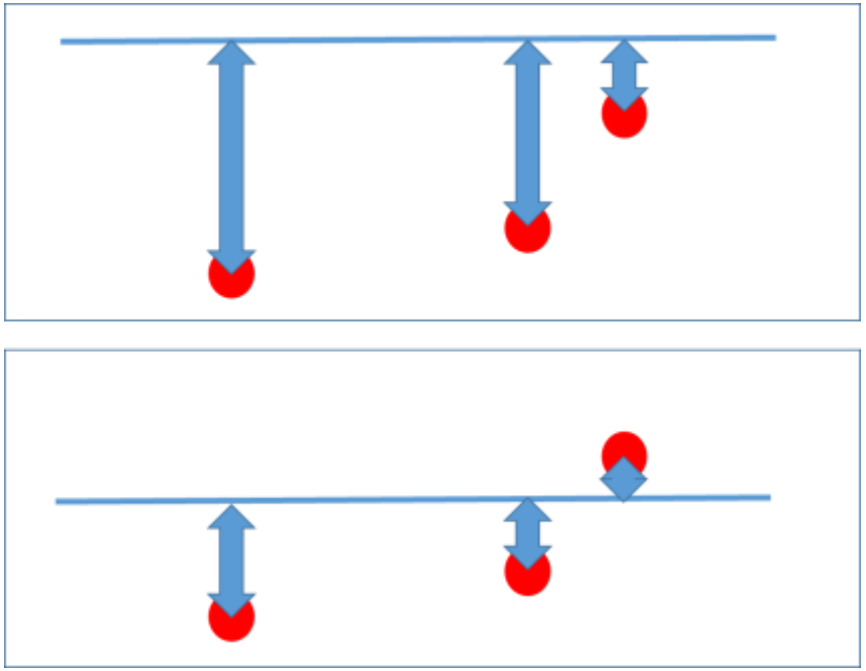


誤差小

赤点：元データ

誤差の変化

- 直線の上下移動や，傾きの変化により，**誤差**が変化



最適化

- **最適化**は,

パラメータを調整して、
ある尺度での**値**を**最適**にするように,
調整を行うこと

ゴール：　誤差の最小化
パラメータ：　直線の上下の位置と,
直線の傾き

誤差を自動で最小化したいときに有効な技術

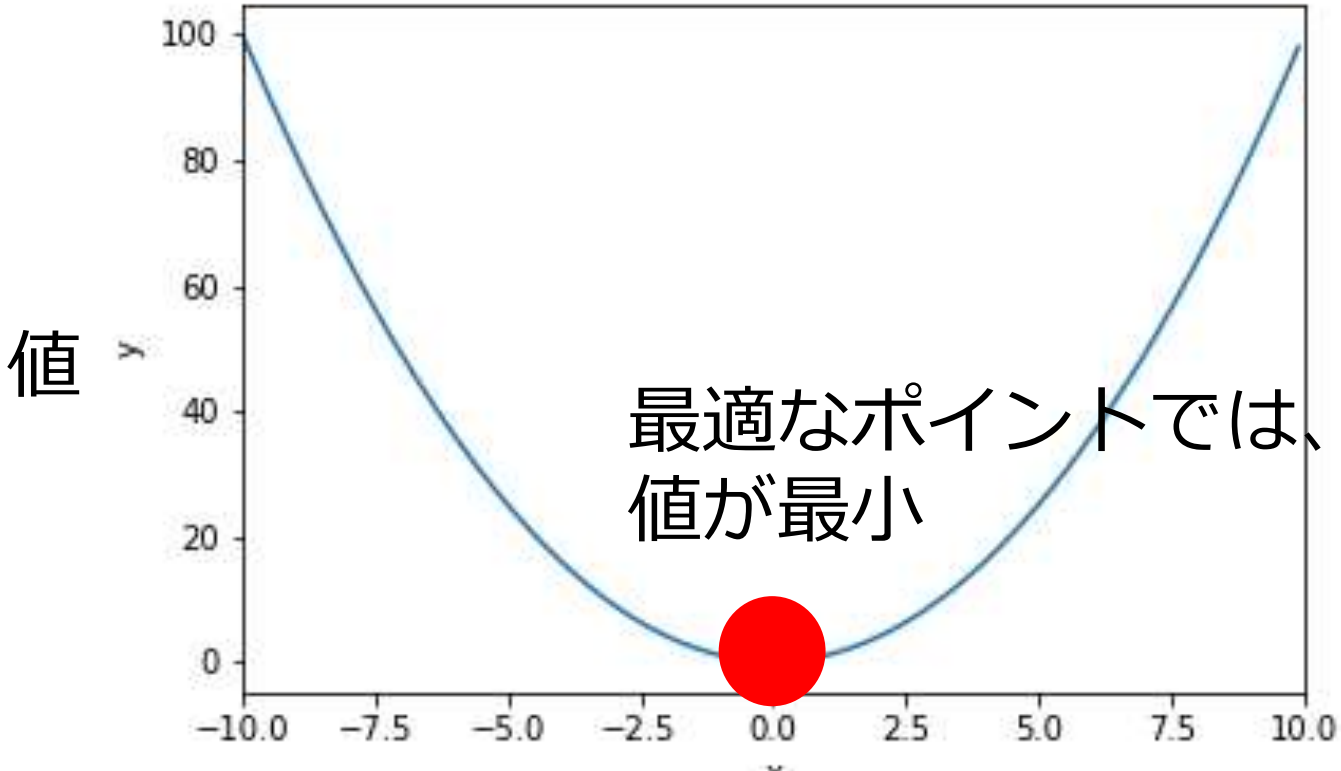
9-5 最適化の仕組み

（人工知能の基本）

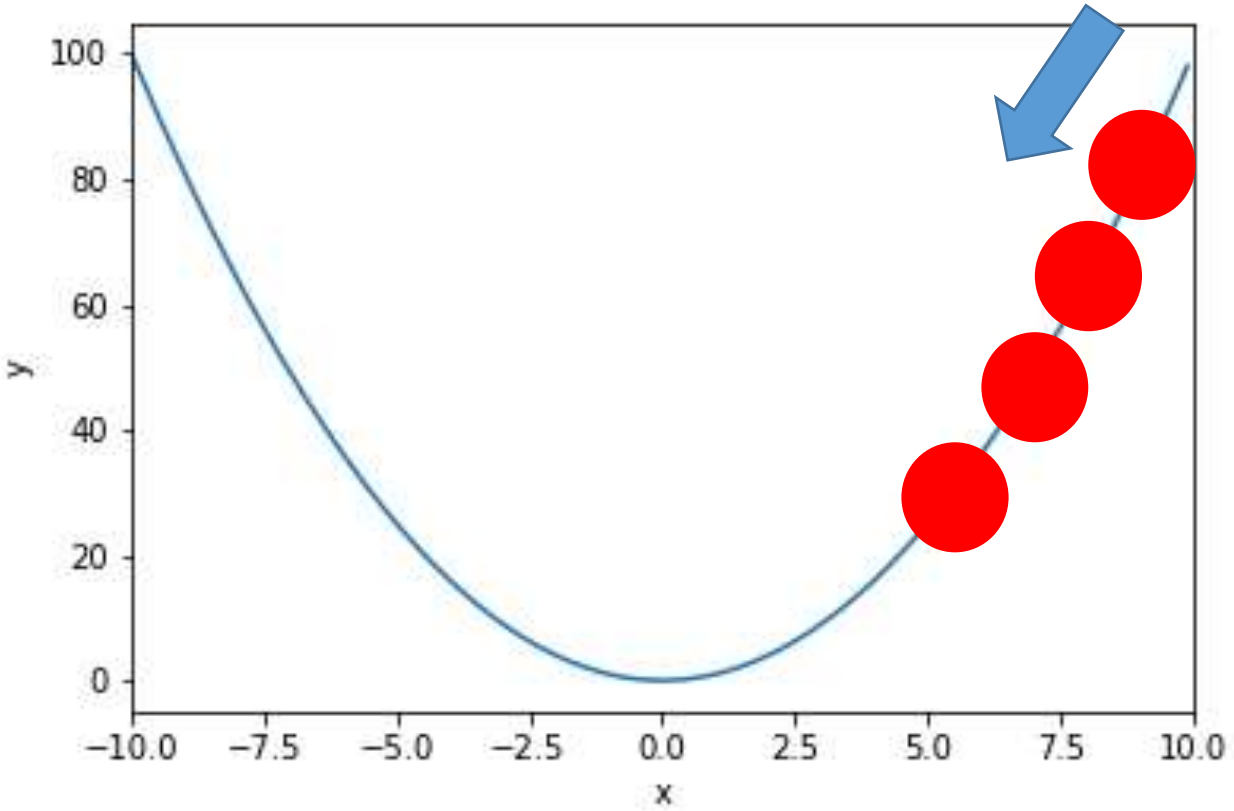
URL: https://www.kkaneko.jp/db/mi/index.html

金子邦彦

- **パラメータの変化**により，**値**が**変化**
- ここでは**「値が最小」になるのが最適**であるとする

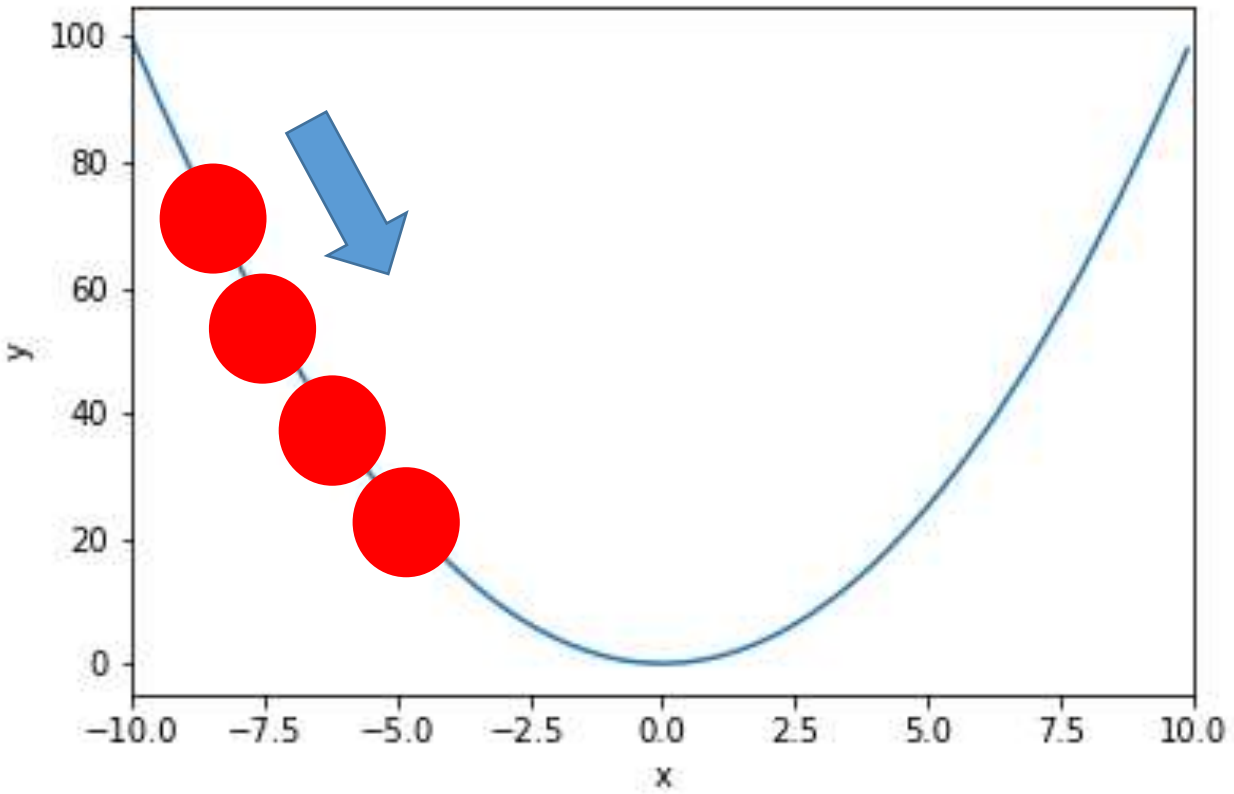


傾きが右**上がり**のとき：
パラメータ値を**減らす**と、
最適に近づく

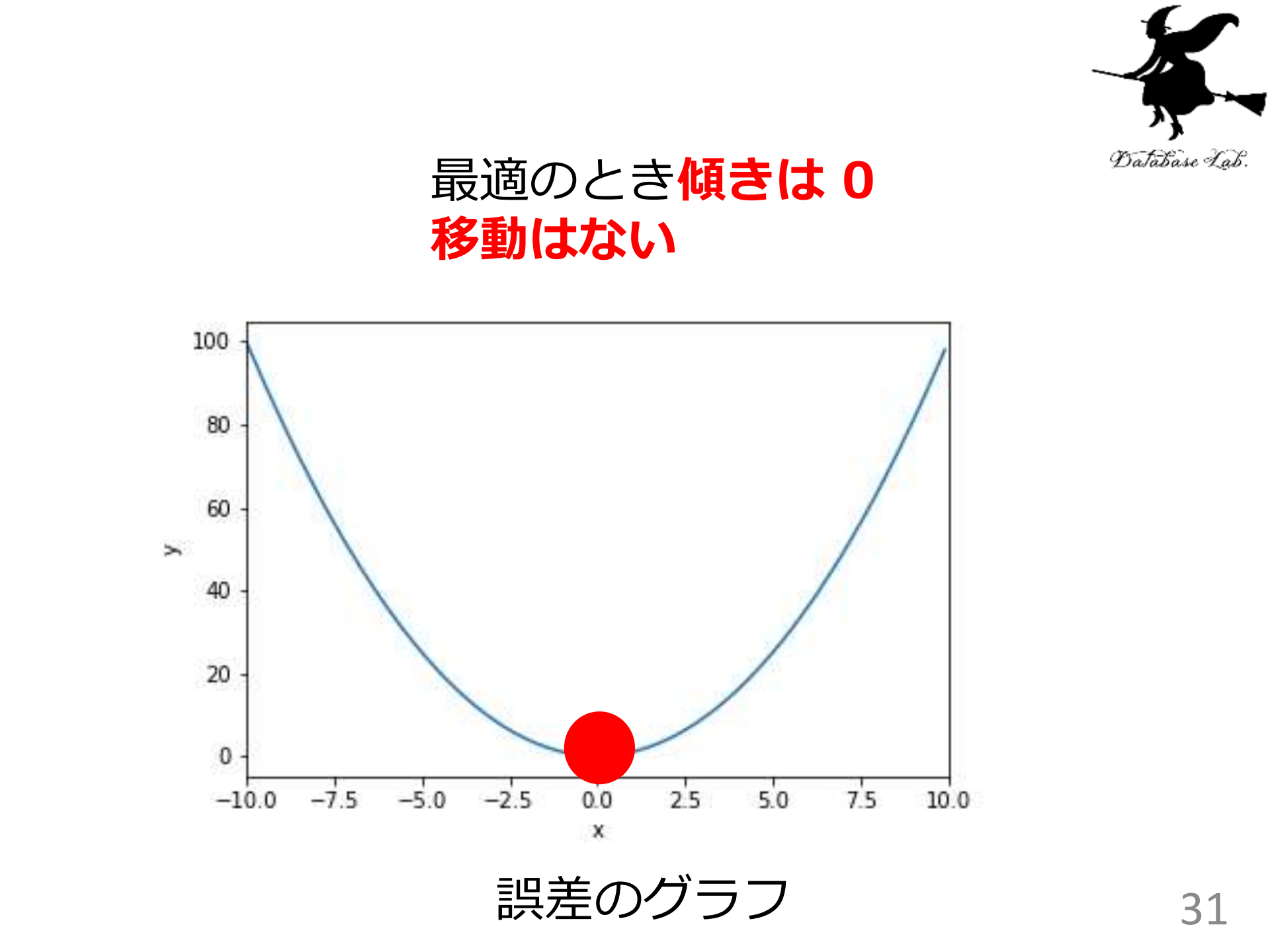


１回では最適にならない。
移動を繰り返す

傾きが右**下がり**のとき：
パラメータ値を**増やす**と、
最適に近づく



１回では最適にならない。
移動を繰り返す



最適のとき**傾きは 0**
移動はない

誤差のグラフ

傾きと最適化

パラメータの変化により、値が増減するとき

- **値の変化（傾き）**による**パラメータ調整**により、**最適ポイントを得る**

9-6 最適化が役に立つ例

（人工知能の基本）

URL: https://www.kkaneko.jp/db/mi/index.html

金子邦彦

最適化の例

- **次の式の値が最小**になるように，x **の値を定めたい**．但し，$N = 5$ とする．

$$f(\mathbf{x}) = \sum_{i=2}^{N} 100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2.$$

- https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

最適化を行う Python プログラム


```
import numpy as np
from scipy.optimize import minimize
def rosen(x):
    """The Rosenbrock function"""
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)

x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
res = minimize(rosen, x0, method='nelder-mead',
    options={'xtol': 1e-8, 'disp': True})
print(res.x)
```

最適化を行う Python プログラム


```
import numpy as np
from scipy.optimize import minimize
def rosen(x):
    """The Rosenbrock function"""
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)

x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
res = minimize(rosen, x0, method='nelder-mead',
    options={'xtol': 1e-8, 'disp': True})
print(res.x)
```

```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 339
         Function evaluations: 571
[1. 1. 1. 1. 1.]
```

x = [1 1 1 1 1] のとき（すべての値が 1のとき）
最適であると求まった.

9-7 学習不足，過学習

（人工知能）

金子邦彦

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）

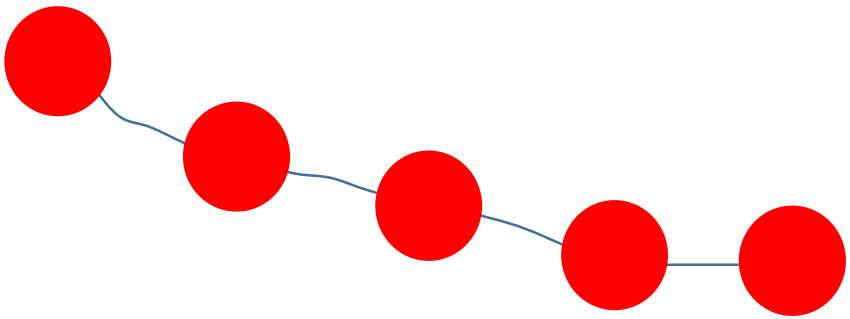
- ニューロンの数が増えたり減ったりなどではない
- **誤差**が**減らなくなったら**、**最適**になったとみなす

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

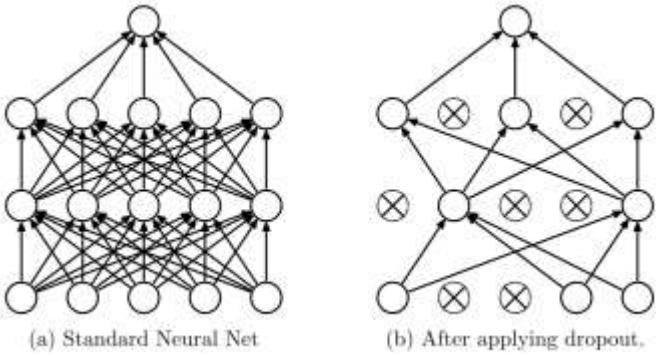
- **教師データ**での学習を終了したとき

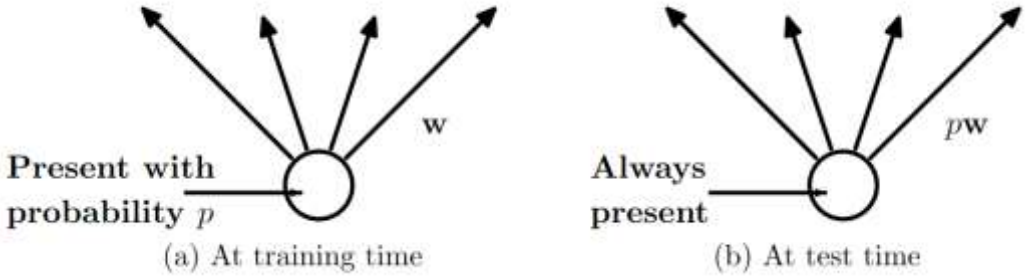
- 他のデータ（教師以外のデータ）で**検証**すると，
学習がうまくいっていないことが分かること

・解決策

教師データの拡張（増量）と再学習，
ドロップアウト，重みの正則化

ドロップアウト





ドロップアウト：学習時に，
ニューロンをランダムに選び，
存在しないことにする．
（学習を繰り返すたびに選びなおす）

存在確率を p とする．
学習時には，確率 p で存在する．
検証時には，重みに p をかける（掛け算）

TensorFlow でのプログラム例
p = 0.5
tf.keras.layers.Dropout(rate = 1 - p)

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. ***Dropout: A Simple Way to Prevent Neural Networks from Overfitting.*** *The Journal of Machine Learning Research, Volume 15 Issue 1, January 2014 Pages 1929-1958*
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

重みの正則化

- 学習時に，ニューロン間の結合の重みを**正則化**する
- 正則化には，L1，L2，Elastic Net などの種類がある
- L2 正則化では，「結合の重みが多いほど，誤差を増やす」という考え方が導入される

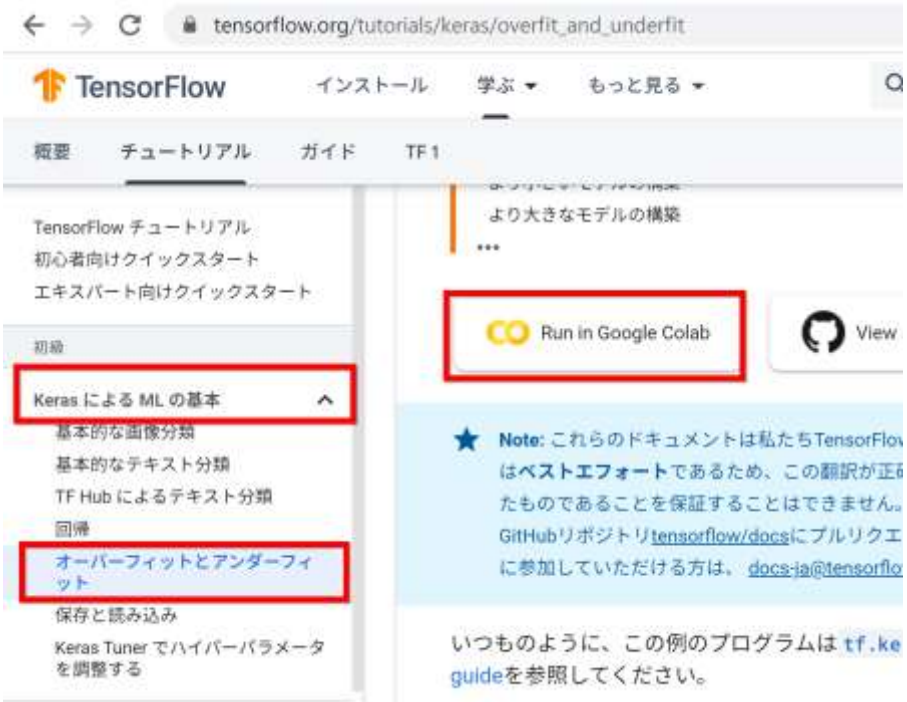
TensorFlow でのプログラム例
tf.kernel_regularizer=keras.regularizers.l2(0.001)

手順

① パソコンの Web ブラウザで，次のページを開く

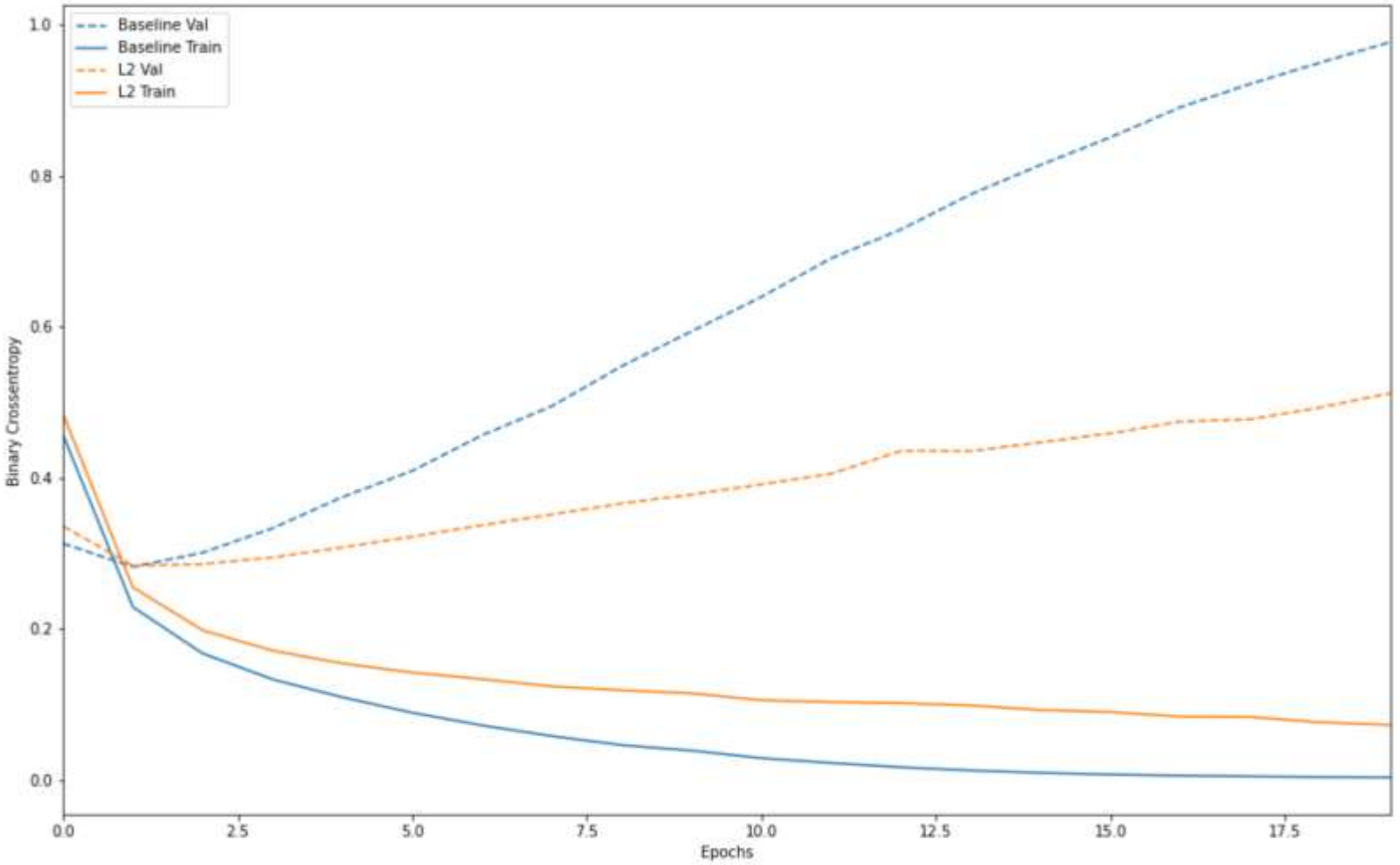
https://www.tensorflow.org/tutorials

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



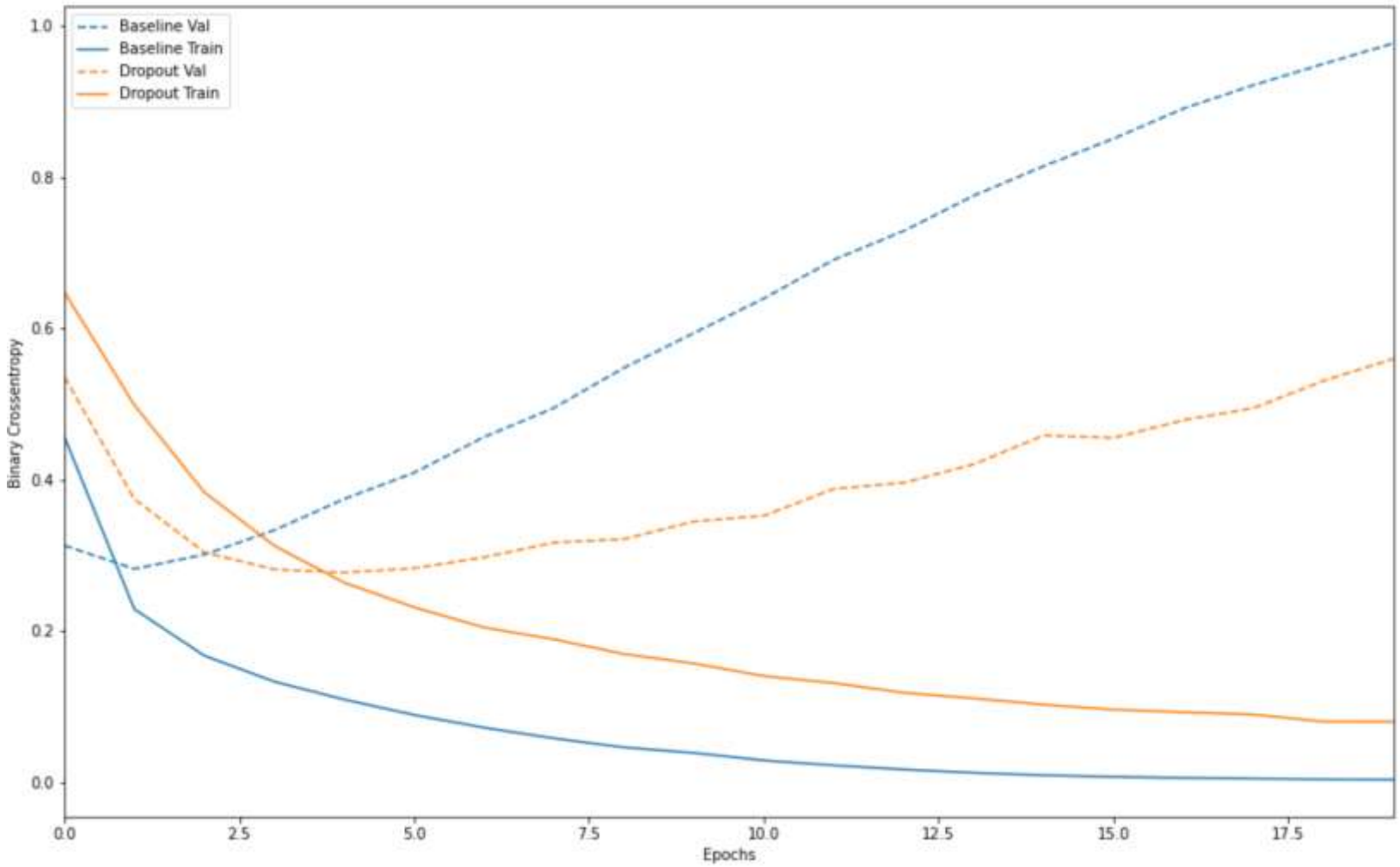
L2正則化の効果

- 検証により確認



ドロップアウトの効果

- 検証により確認



データの拡張の例

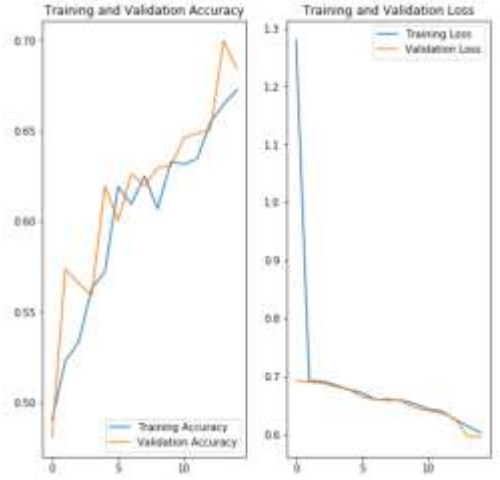
データの拡張：**教師データ**の**拡張（増量）**による**過学習**の**解決**

- 次のページで公開されているデモを**実行**してみる

https://www.tensorflow.org/tutorials/images/classification

データの拡張前
過学習あり

画像データの拡張
・拡大、縮小、反転、
移動により、内容を変えずに
データを拡張（増量）



データの拡張後
過学習の解決

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