

# 知能工学特別講義 第1講

担当：和田山 正

wadayama@nitech.ac.jp

名古屋工業大学

# 本講義の学習目標

- ・深層学習とスパースモデリングに関する基礎と実践を学ぶ
- ・全8コマ
- ・オンライン講義 + Google Colab(演習課題)
- ・Pytorch のコードを動かして理解を深める
- ・キーワード: ニューラルネットワーク, 勾配法, 誤差逆伝播法, クラス分類問題, 回帰問題, スパースモデリング, 圧縮センシング

# 講義の進め方(1)

演習の進度などにより柔軟に予定を変えますので、当初案ということで

9/20 4コマ

1 深層学習入門

9/20 5コマ

2 ニューラルネットワーク学習

9/21 3コマ

3 パターン認識問題

9/21 3コマ

4 回帰問題

9/21 5コマ

5 予備時間

9/22 3コマ

6 深層展開とモデルベース深層学習

9/22 4コマ

7 深層生成モデル

9/22 5コマ

8 予備時間(演習・レポート作成)

# 講義の進め方(2)

60 ~ 70分

20 ~ 30分

スライドによる内容の説明

Google colabによる演習

- ・演習課題で時間中出来なかつたものは、可能であれば23日、または集中講義後に自分でやってもらえると理解が進みます。
- ・演習があるので、演習の進め方に従って柔軟に予定を修正しながら進めます。

# 成績評価

- ・演習課題(Google colab)に演習課題がついていますので、結果・考察をPDFファイルにまとめて提出していただきます。
- ・そのレポートに基づき成績評価を行います。
- ・提出方法・締め切りは最後の講義に連絡します。

# 本講義の内容

- 深層学習の概要
- 勾配法による最小化
- Google Colabを使ってみる

# 深層學習技術の進展

- ・画像認識

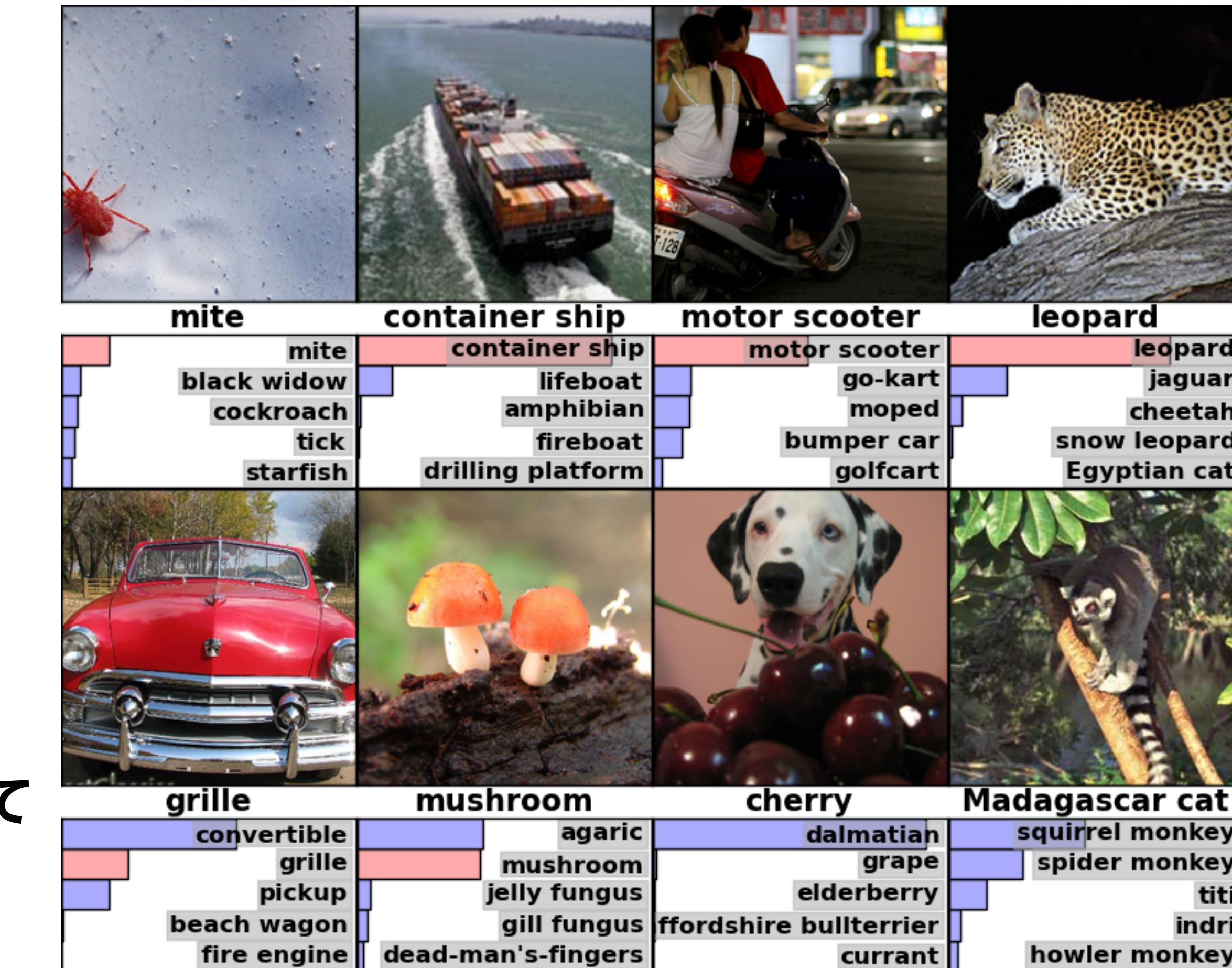
- ・音声認識

- ・自然言語処理

- ・機械翻訳

深層學習技術は、これらの分野において特に圧倒的な強みを見せている

ImageNet Classification



cited from: ``ImageNet Classification with Deep Convolutional Neural Networks'', Alex Krizhevsky et al.

<https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf>

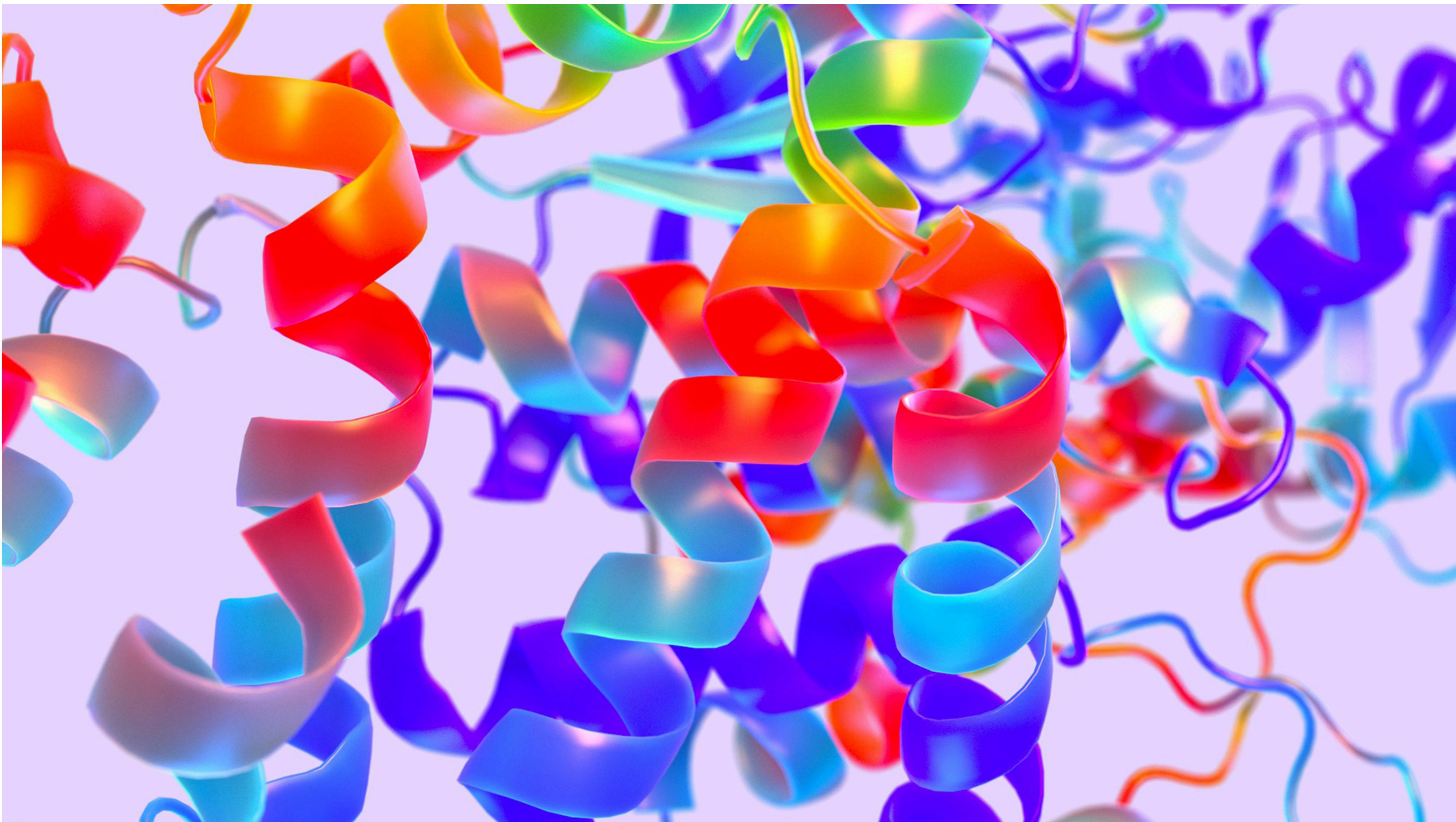
# AlphaGOの登場(2016)



(左)2016年 AlphaGo と対戦したイ・セドル  
(右)DeepMind のデミス・ハサビス

- ・モンテカルロ木探索 + 深層畳み込みネットワーク(ポリシーネットワーク)
- ・3週間50GPUを利用し、3億4000万回のトレーニング

# AlphaFold2(2021)



<https://deepmind.com/blog/article/AlphaFold-Using-AI-for-scientific-discovery>

# 深層生成モデル (2022)

## Stable Diffusion

<https://huggingface.co/spaces/stabilityai/stable-diffusion>

prompt:

mecha robot hamster

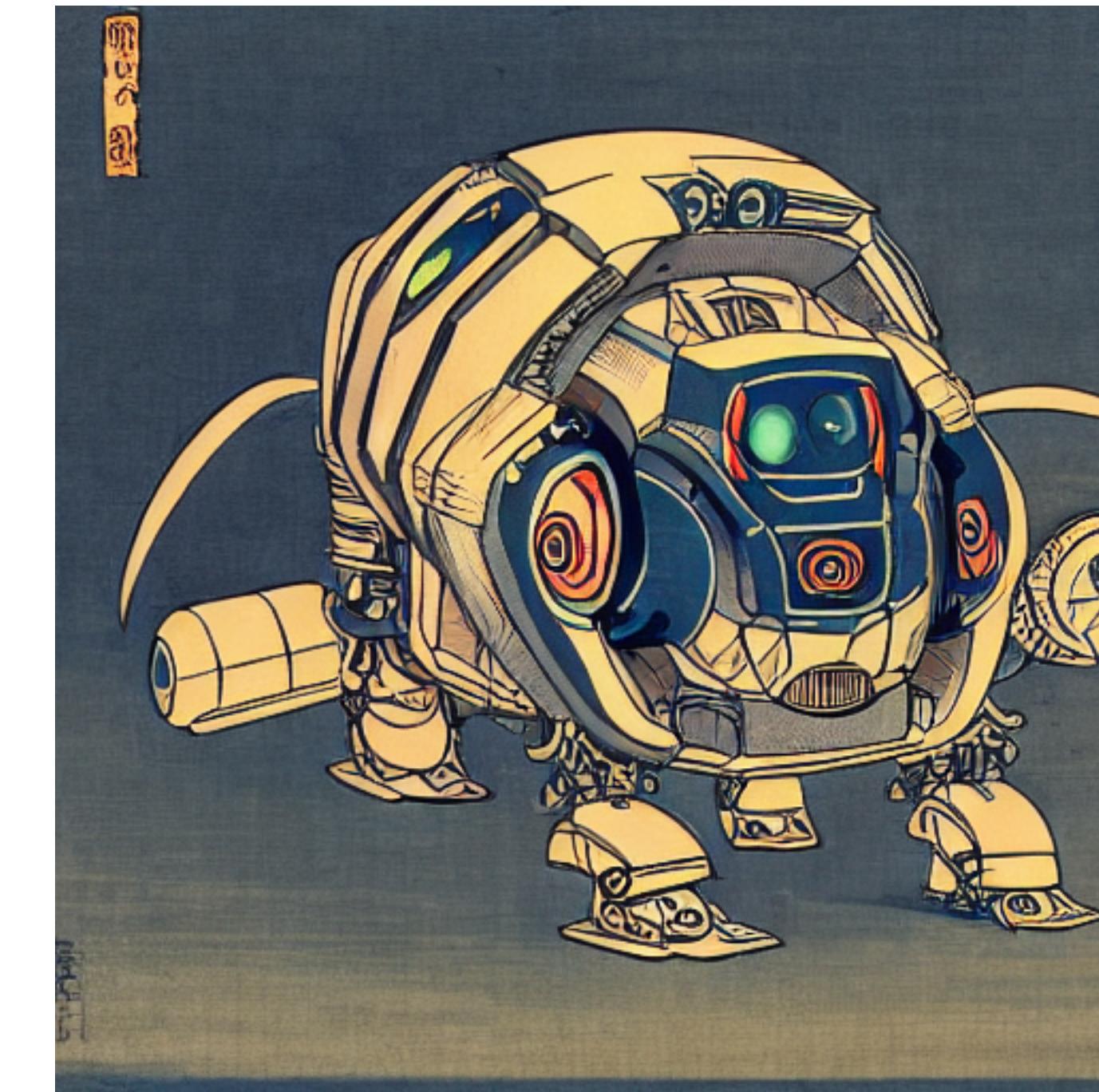
in photo style



prompt:

mecha robot hamster

in Hokusai style



prompt:

hamster

in Claude Monet style



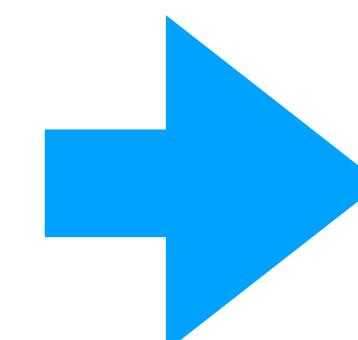
# 深層学習の概要



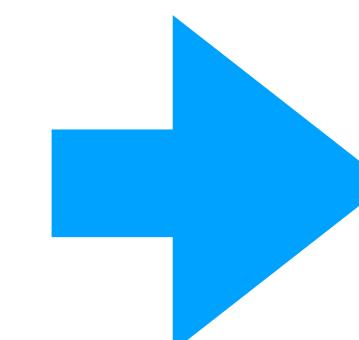
# イヌ・ネコ分類問題

こんな関数を作りたい！

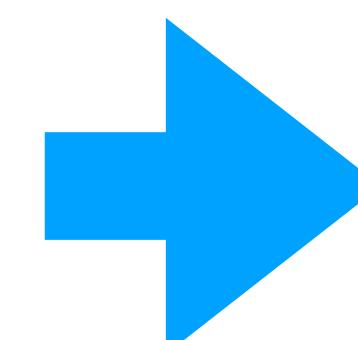
- ・入力された画像について、写っているのがイヌなのかネコなのか  
判別する関数を構成したい（クラス分類問題）



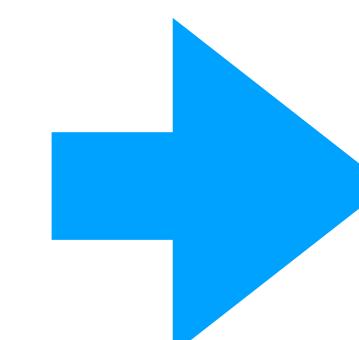
$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



0: 猫



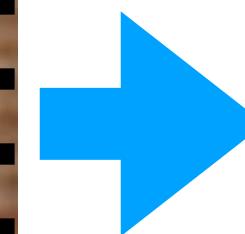
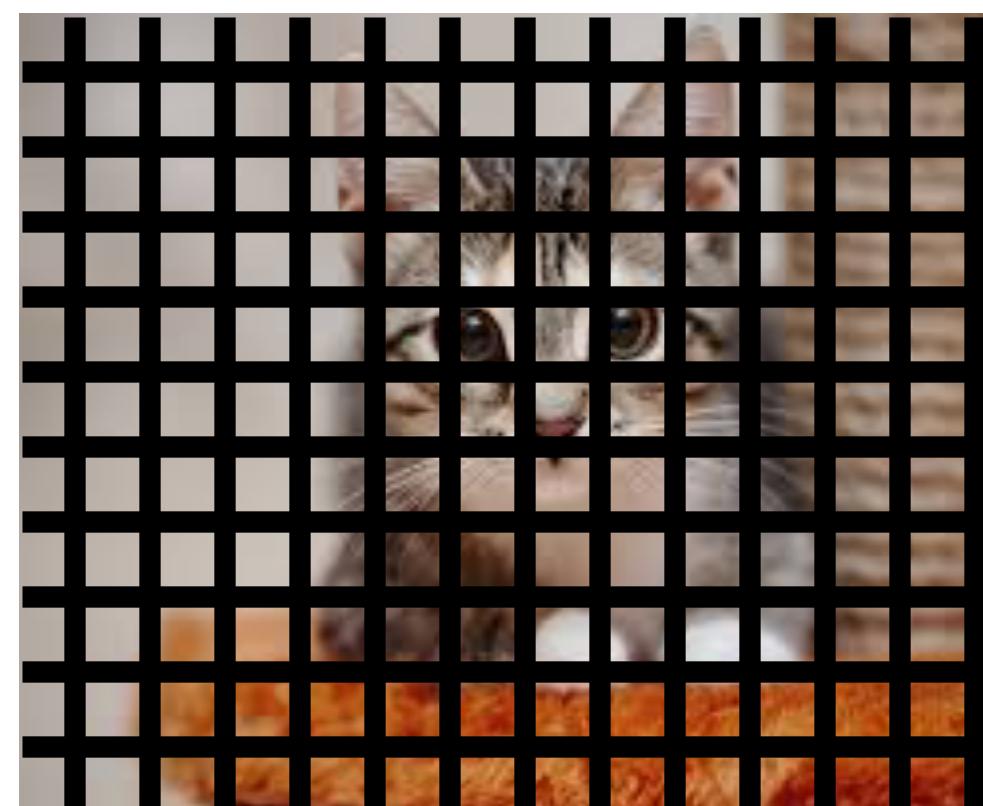
$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



1: 犬

# イヌ・ネコ分類問題

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



- 一次元に並べ替える
- $n$ 個の実数の組として、画像が表現される

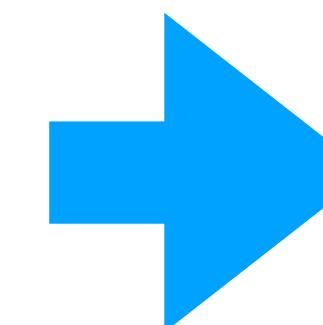
$$(x_1, x_2, \dots, x_n) \rightarrow f(x_1, x_2, \dots, x_n)$$

- 関数 $f$ を適用する
- 0 または 1 の値が返る

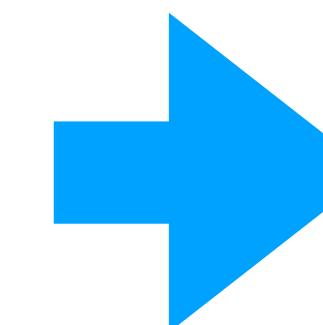
- ピクセルに分割
- 例ではカラーだけど白黒  
各ピクセルは実数だと思う

# イヌ・ネコ分類問題

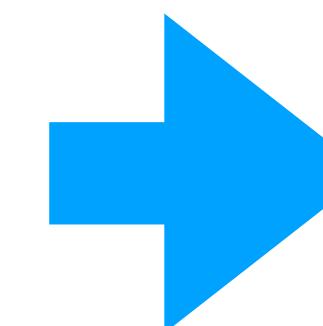
こんな関数を作りたい！



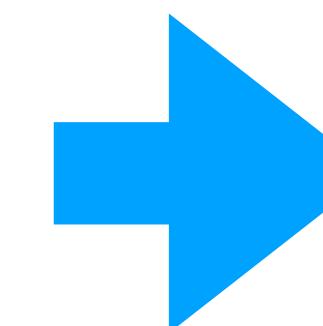
$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



0: 猫



$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



1: 犬

# 関数の形状を制御するパラメータを導入する

パラメトリックモデル

$$f_{\Theta} : \mathbb{R}^n \rightarrow \mathbb{R}$$

$\Theta$ : 関数の形状をコントロールするパラメータ群

---

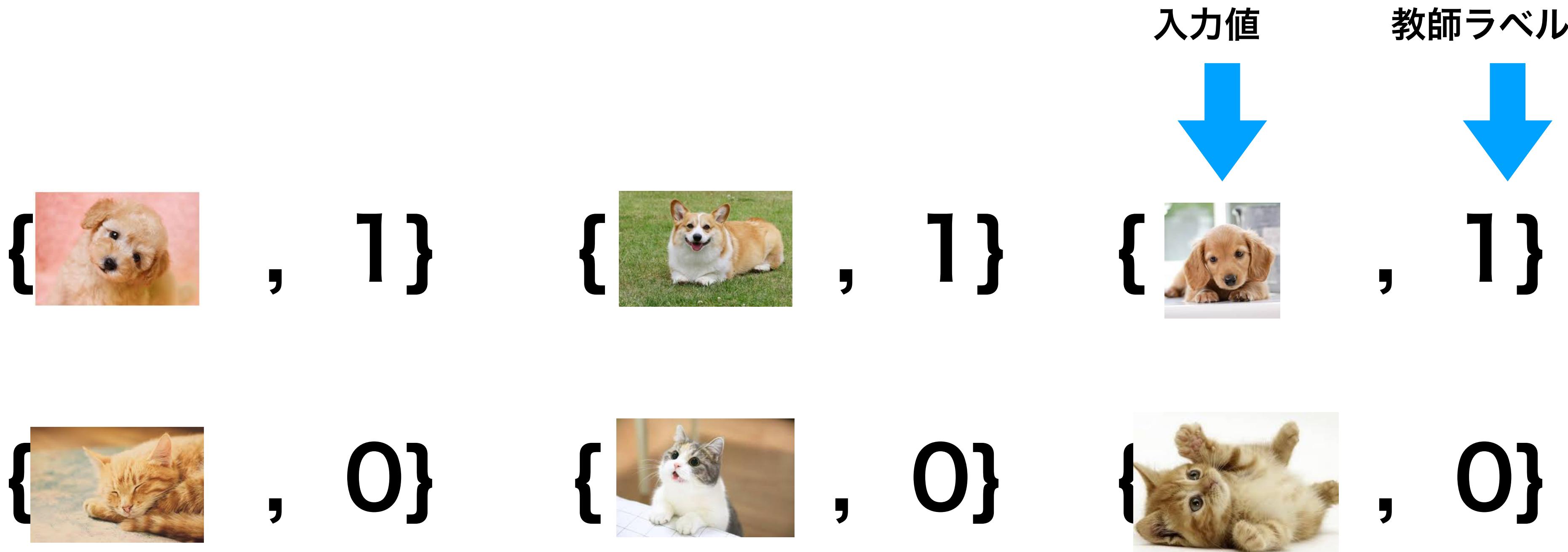
例：2次関数に基づくパラメトリックモデル

$$f_{\Theta}(x) = ax^2 + bx + c$$

$$\Theta = \{a, b, c\}$$

# 訓練データを準備する

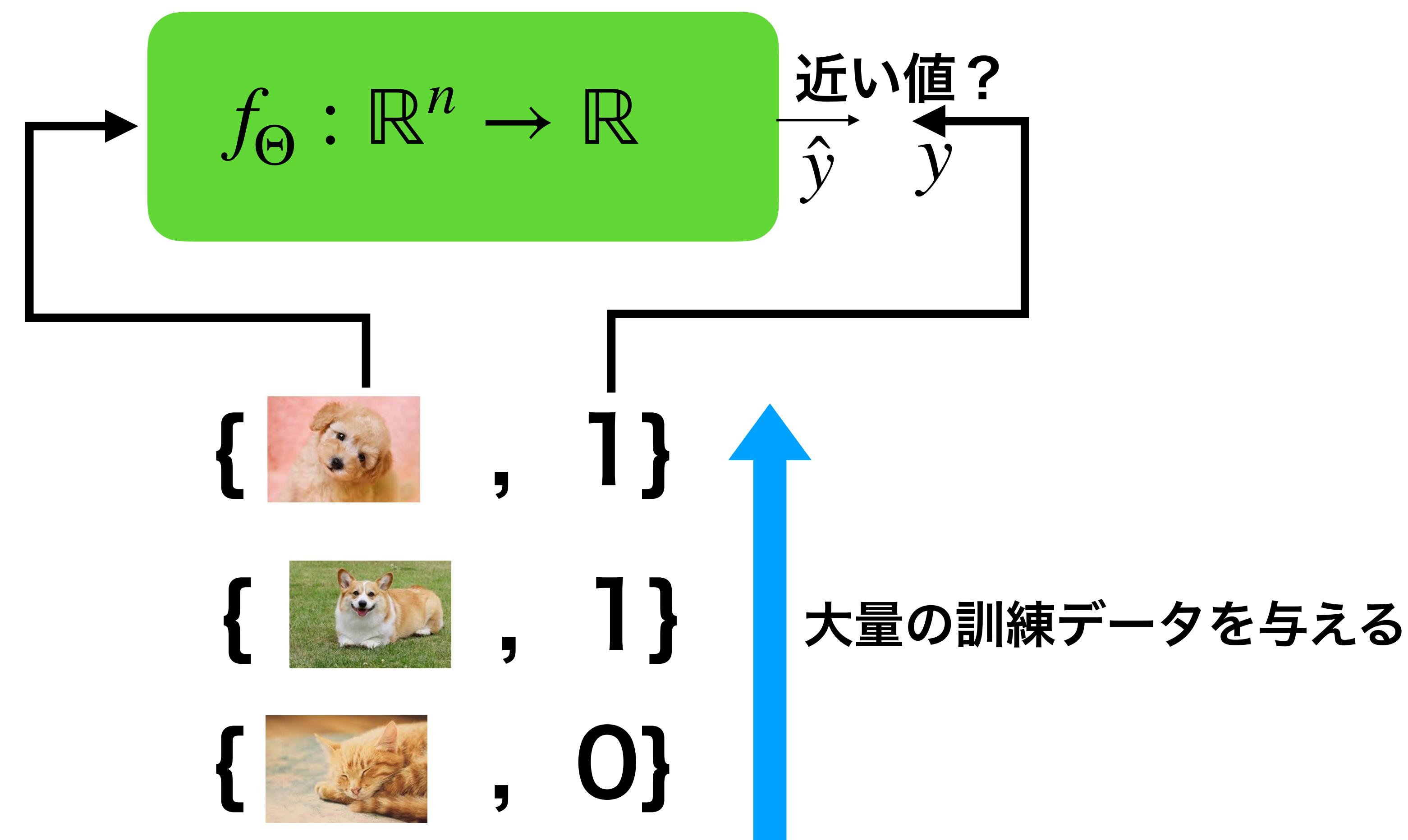
学習パラメータをチューニング（調整）するために訓練データを準備する



- ・画像はn次元のベクトルとして表現しておく
- ・良い認識結果を得るには、一般に多数の訓練データが必要

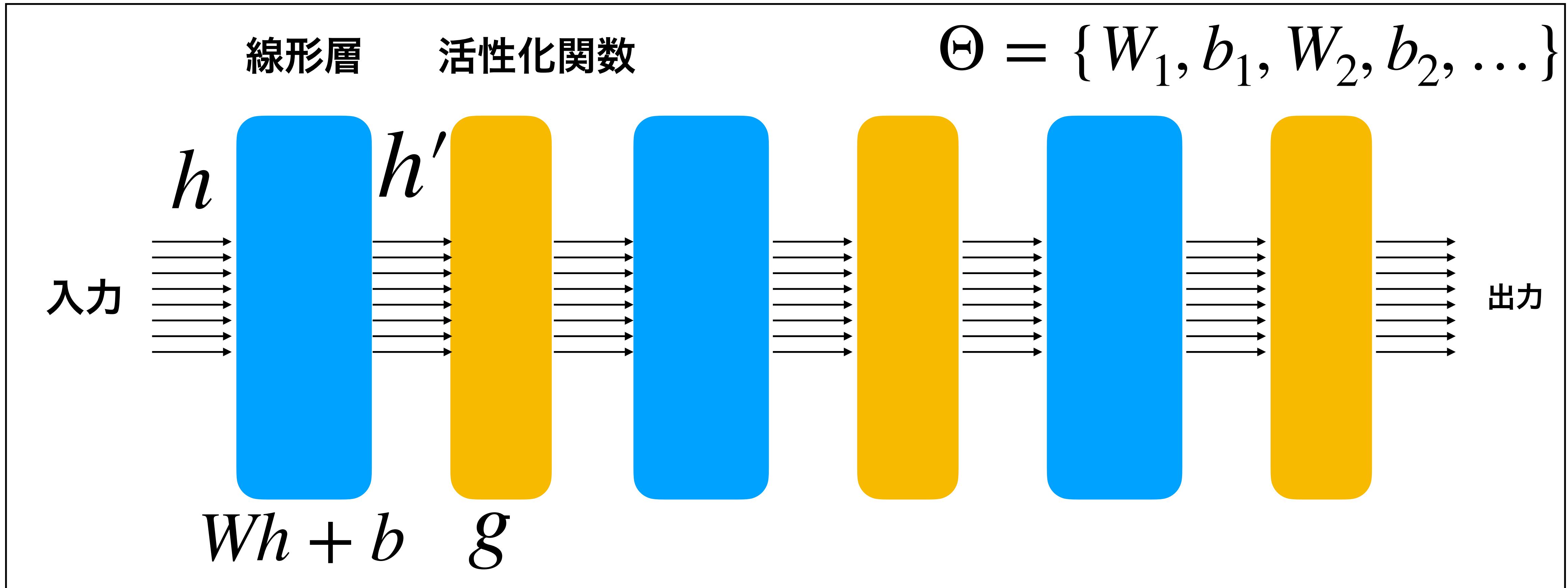
# パラメータを更新する(訓練・学習)

出力と教師ラベルとの間の**食い違い**がなるべく小さくなるように学習パラメータを更新する



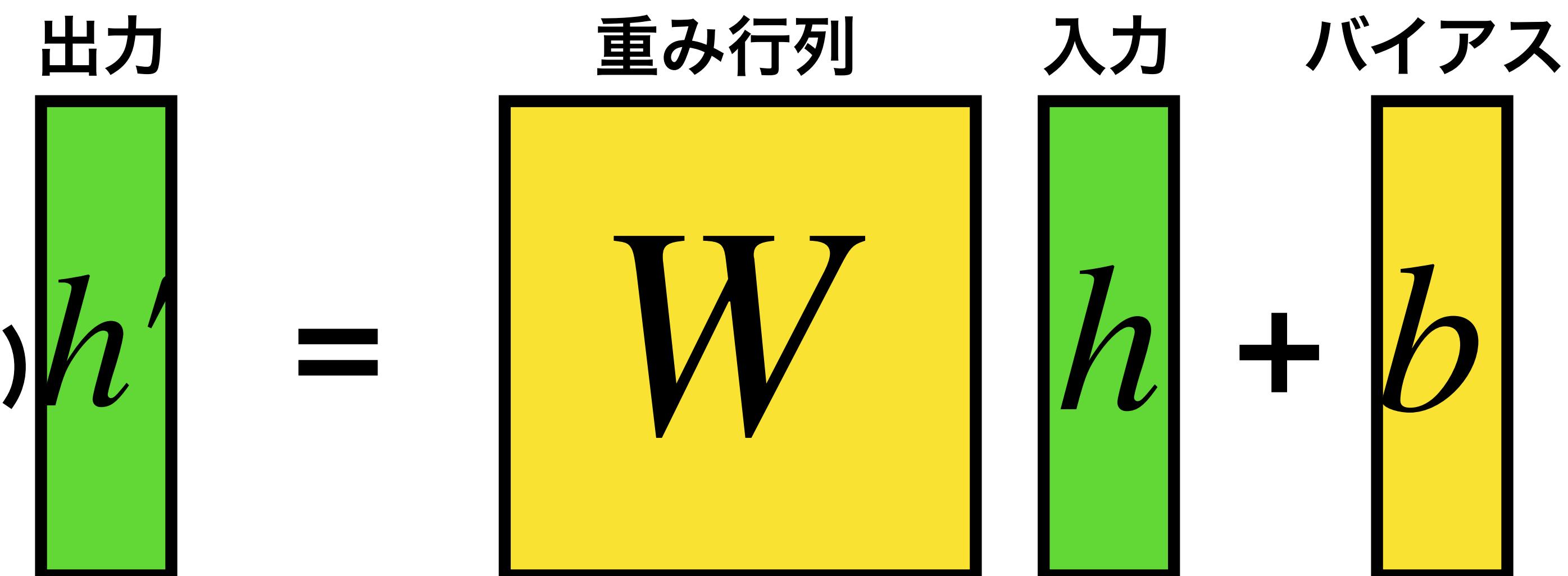
# 深層ニューラルネットワークモデル

$$f_{\Theta} : \mathbb{R}^n \rightarrow \mathbb{R}^m$$



# アフィン変換と活性化関数

アフィン変換  
( $W, b$ : 学習パラメータ)



## 活性化関数

### シグモイド関数

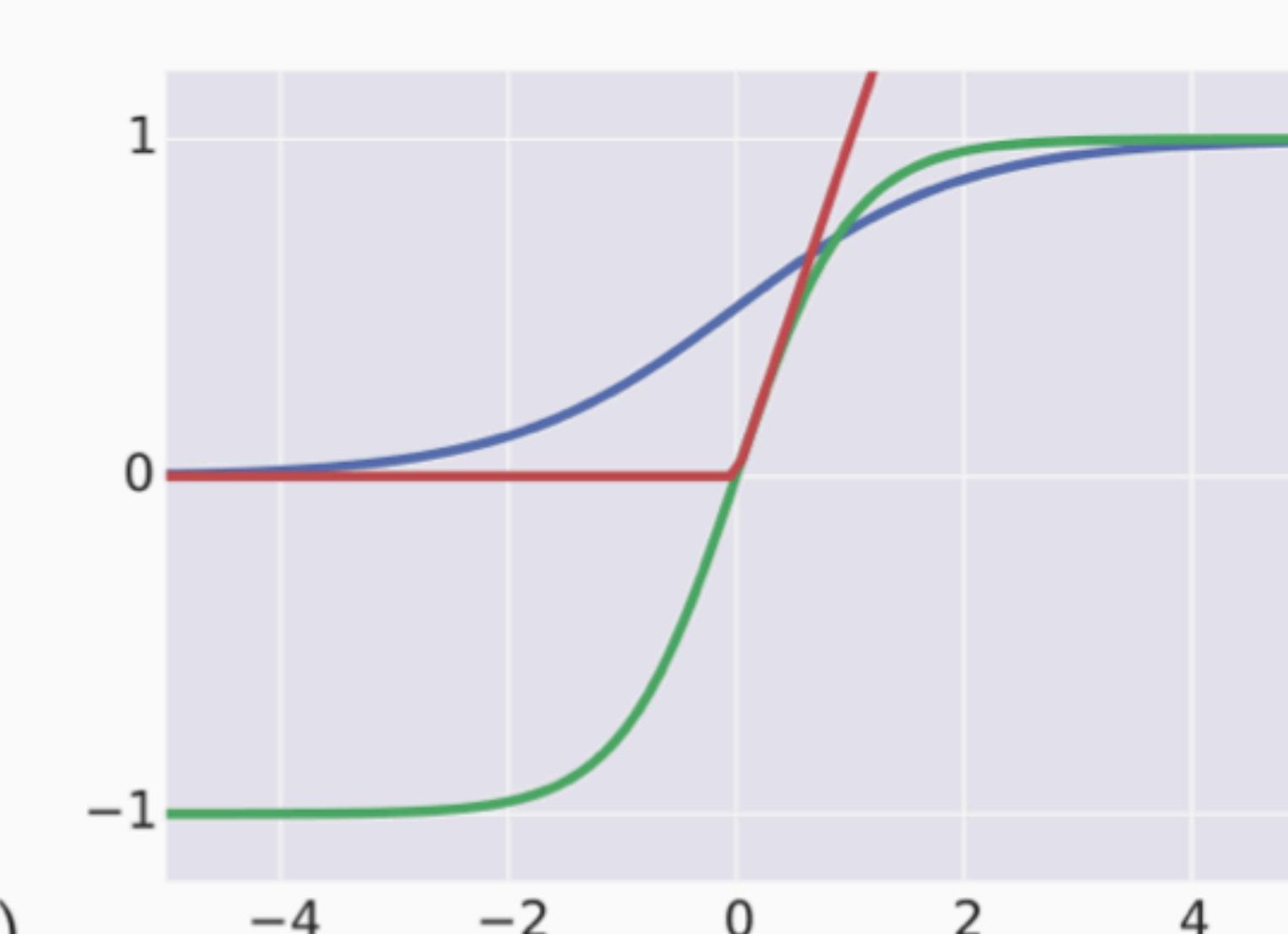
$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

### tanh関数

$$\tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

### ReLU関数

$$\text{ReLU}(u) = \max(0, u)$$



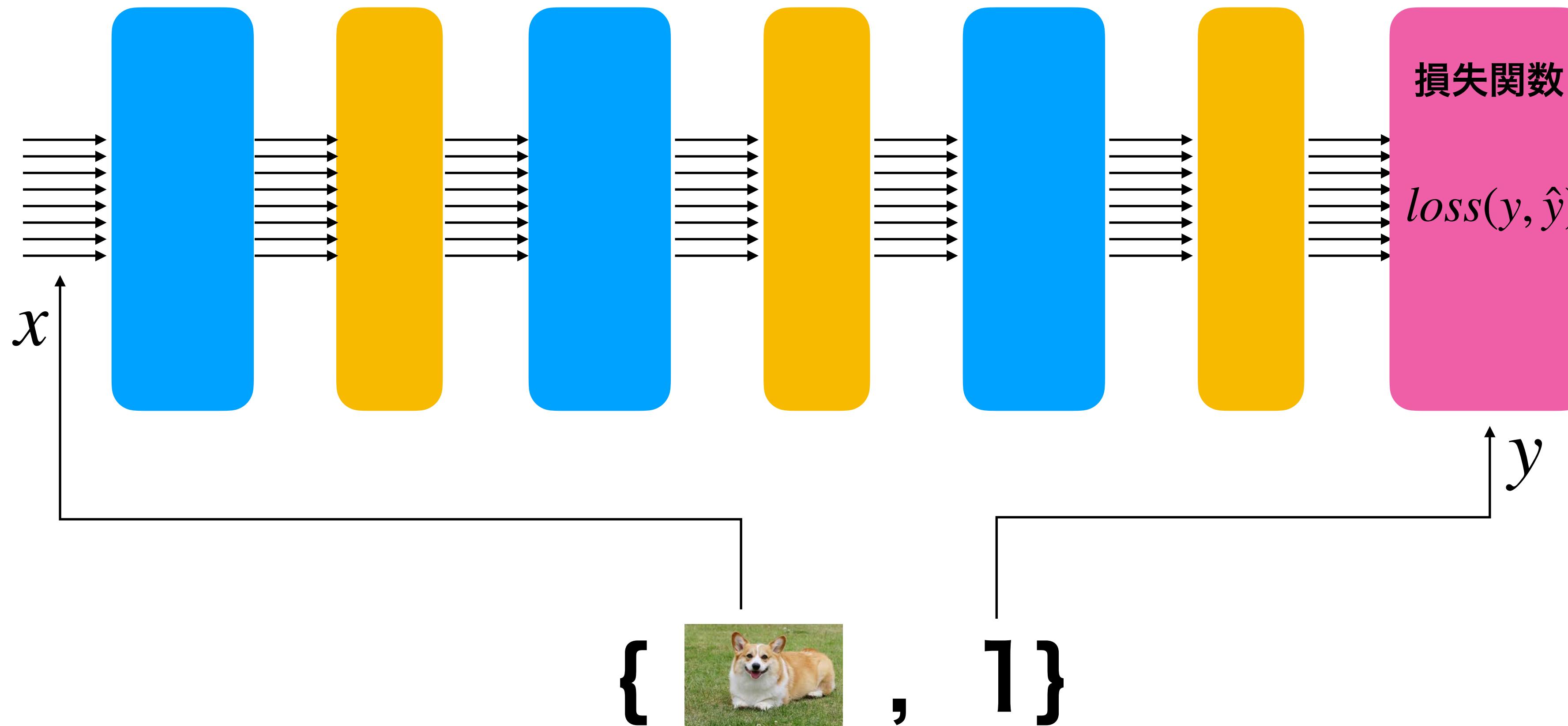
# 学習プロセス

- ・ニューラルネットワークなどのモデルの学習 = 学習パラメータの調整(最適化)
- ・モデル出力と教師信号の食い違い = 損失関数 (二乗誤差関数, クロスエントロピーなど)
- ・学習プロセス = 多数の訓練データに基づく確率的勾配法による損失関数值の最小化

# 深層ネットワークの訓練

$$\Theta = \{W_1, b_1, W_2, b_2, \dots\}$$

$$\hat{y} = f_{\Theta}(x)$$



訓練・学習過程では、損失関数值を最小化するように  
学習パラメータを最適化（調整）する。

# 損失関数

入力信号  $x$

推定出力  $\hat{y} = f_{\Theta}(x)$

教師信号  $y$

二乗誤差関数(典型的な損失関数)

$$loss(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^n |y_i - \hat{y}_i|^2$$

# 学習プロセス

$$h' = W h + b$$

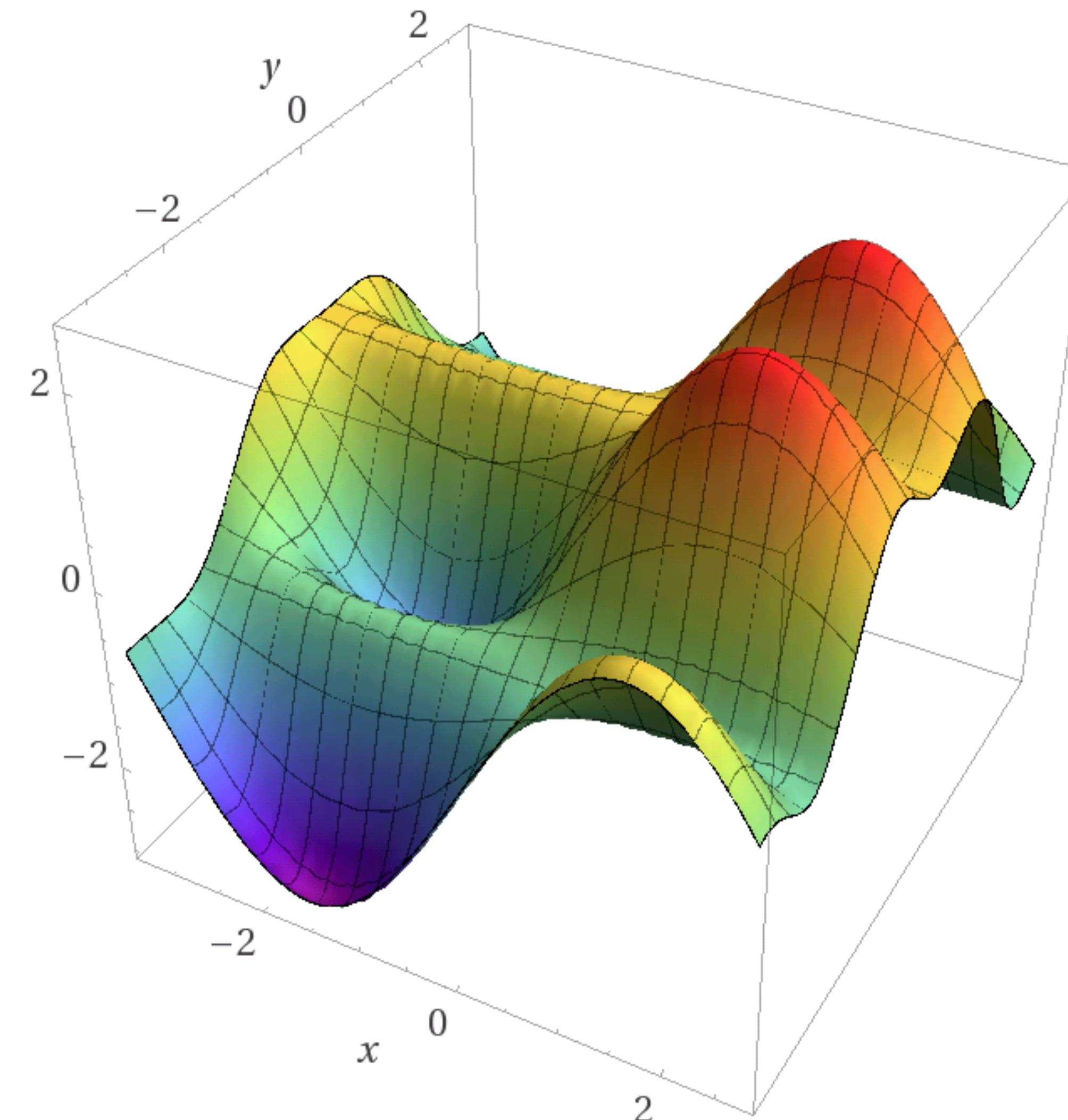
中身は動かしてよい・うまく  
チューンアップしたい！

二乗誤差関数を最小化するように学習パラメータを動かせばよい

$$loss(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^n |y_i - \hat{y}_i|^2$$

# 勾配法による数値最小化

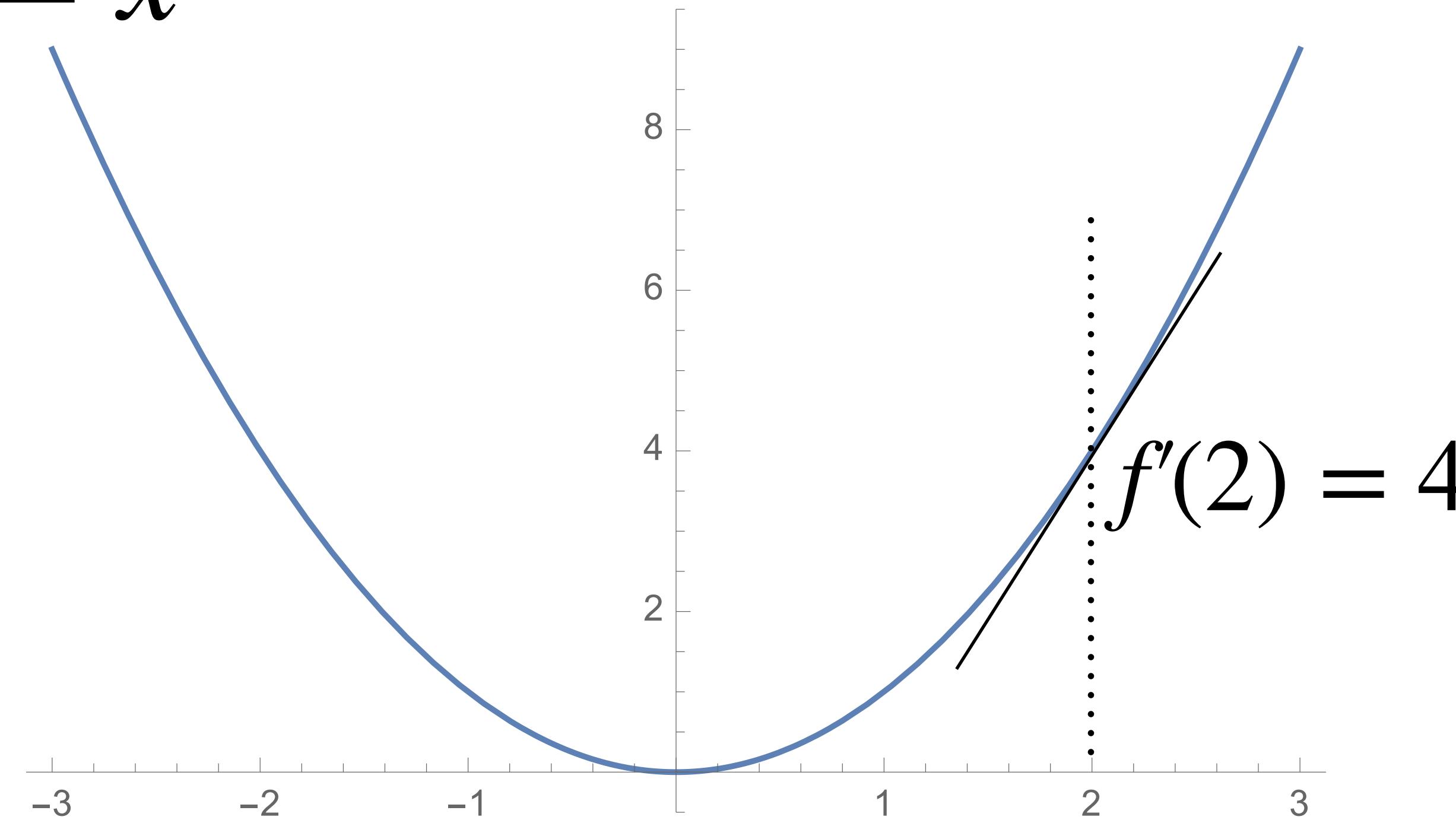
# 学習プロセス=学習パラメータの調整



どうやって誤差を最小にすればよいの？

# 微積の力を借りる！

$$y = x^2$$



導関数  $\frac{dy}{dx} = f'(x) = 2x$

# 勾配法(数値的最小化法)

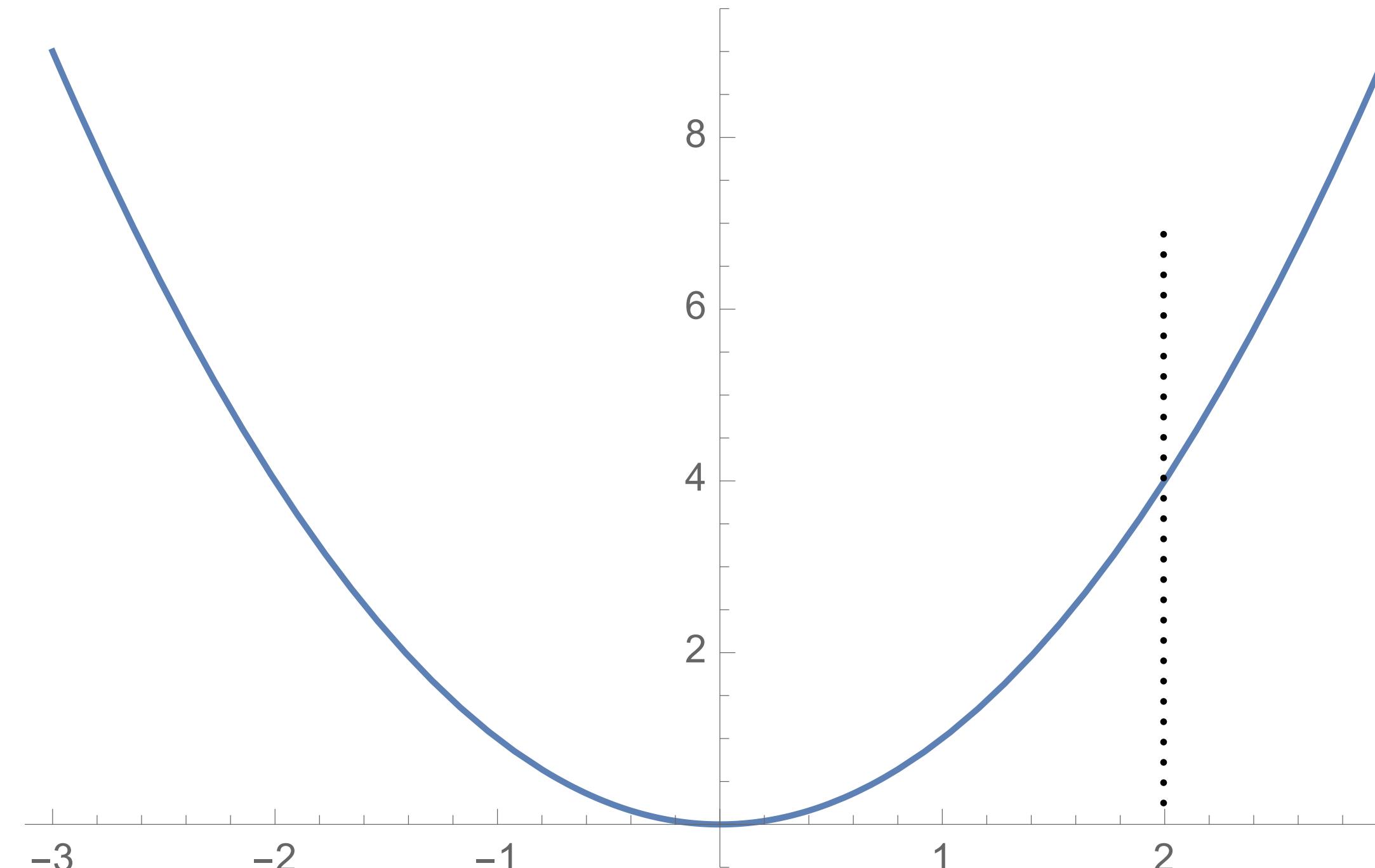
ステップ1 :  $s$  を初期値にセットする

ステップ2 :  $g := f'(s)$  とする

ステップ3 :  $s := s - a g$  とする ( $a$  は学習係数と呼ばれる)

ステップ4 : ステップ2に戻る

[練習問題] 初期値  $s = 2$ ,  $a = 0.2$  として上の勾配法のステップを実行せよ。



# 勾配法(数値的最小化法)

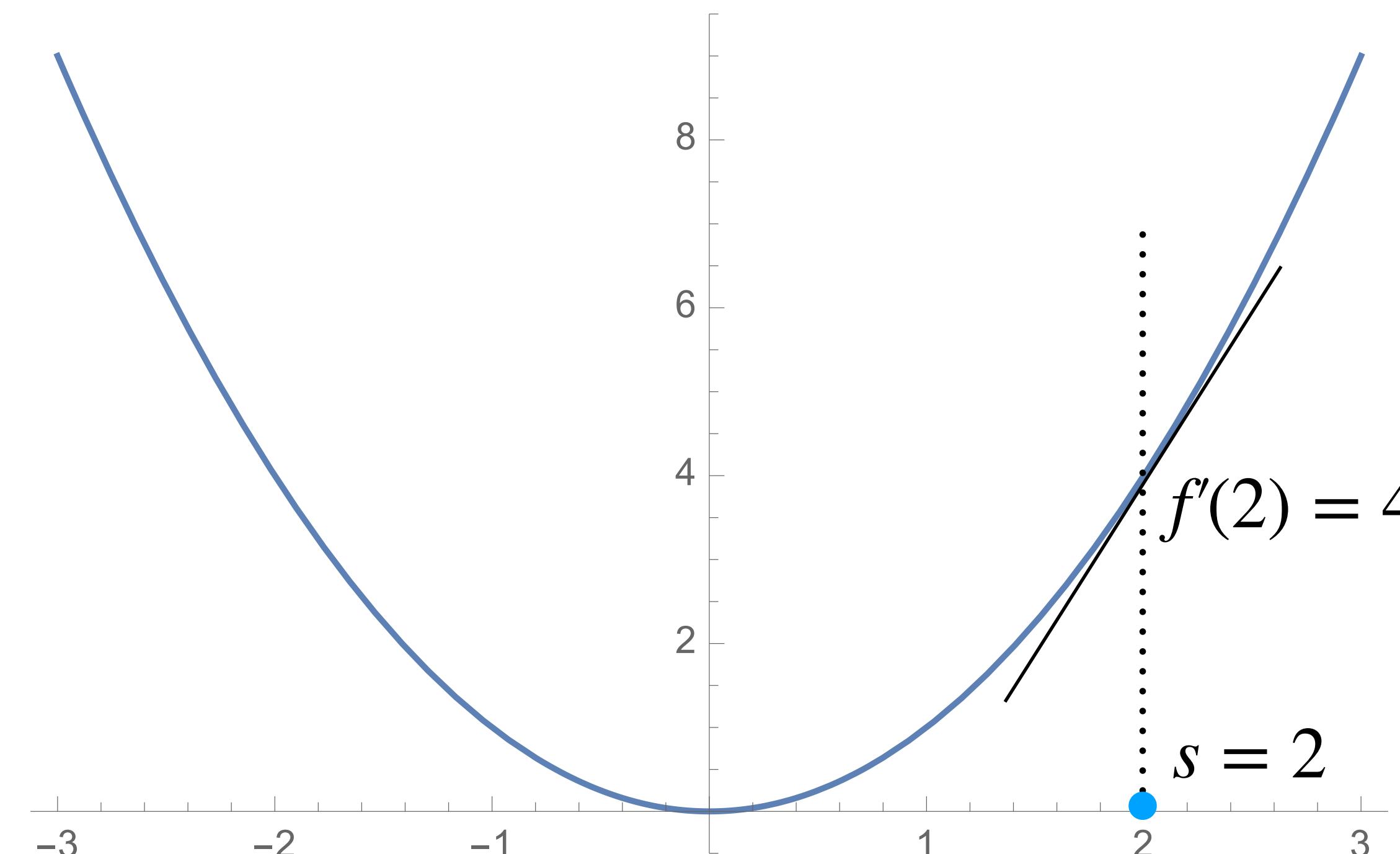
ステップ1 :  $s$  を初期値にセットする

ステップ2 :  $g := f'(s)$  とする

ステップ3 :  $s := s - a g$  とする ( $a$  は学習係数と呼ばれる)

ステップ4 : ステップ2に戻る

[練習問題] 初期値  $s = 2$ ,  $a = 0.2$  として上の勾配法のステップを実行せよ。



# 勾配法(数値的最小化法)

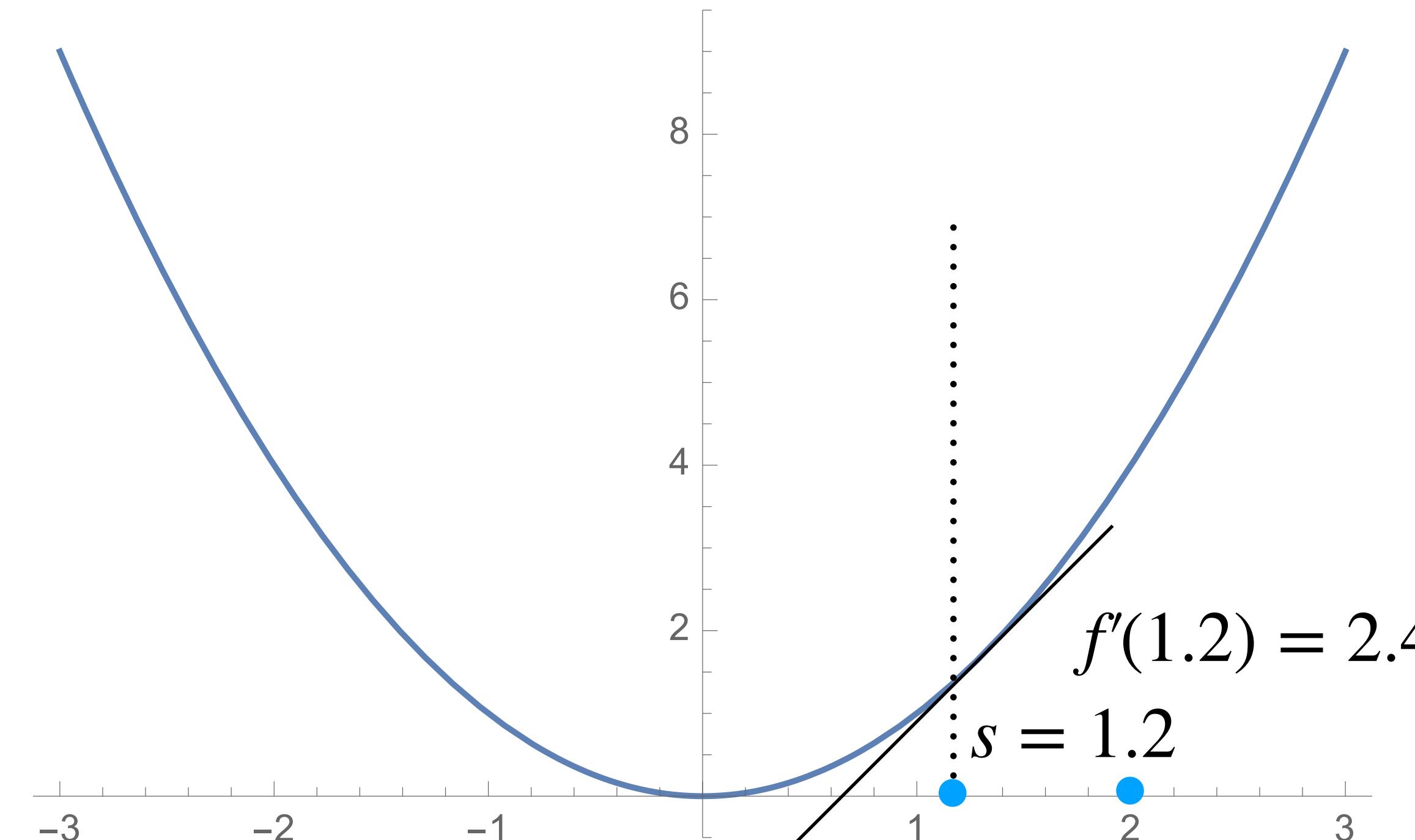
ステップ1 :  $s$  を初期値にセットする

ステップ2 :  $g := f'(s)$  とする

ステップ3 :  $s := s - a g$  とする ( $a$  は学習係数と呼ばれる)

ステップ4 : ステップ2に戻る

[練習問題] 初期値  $s = 2$ ,  $a = 0.2$  として上の勾配法のステップを実行せよ。



# 勾配法(数値的最小化法)

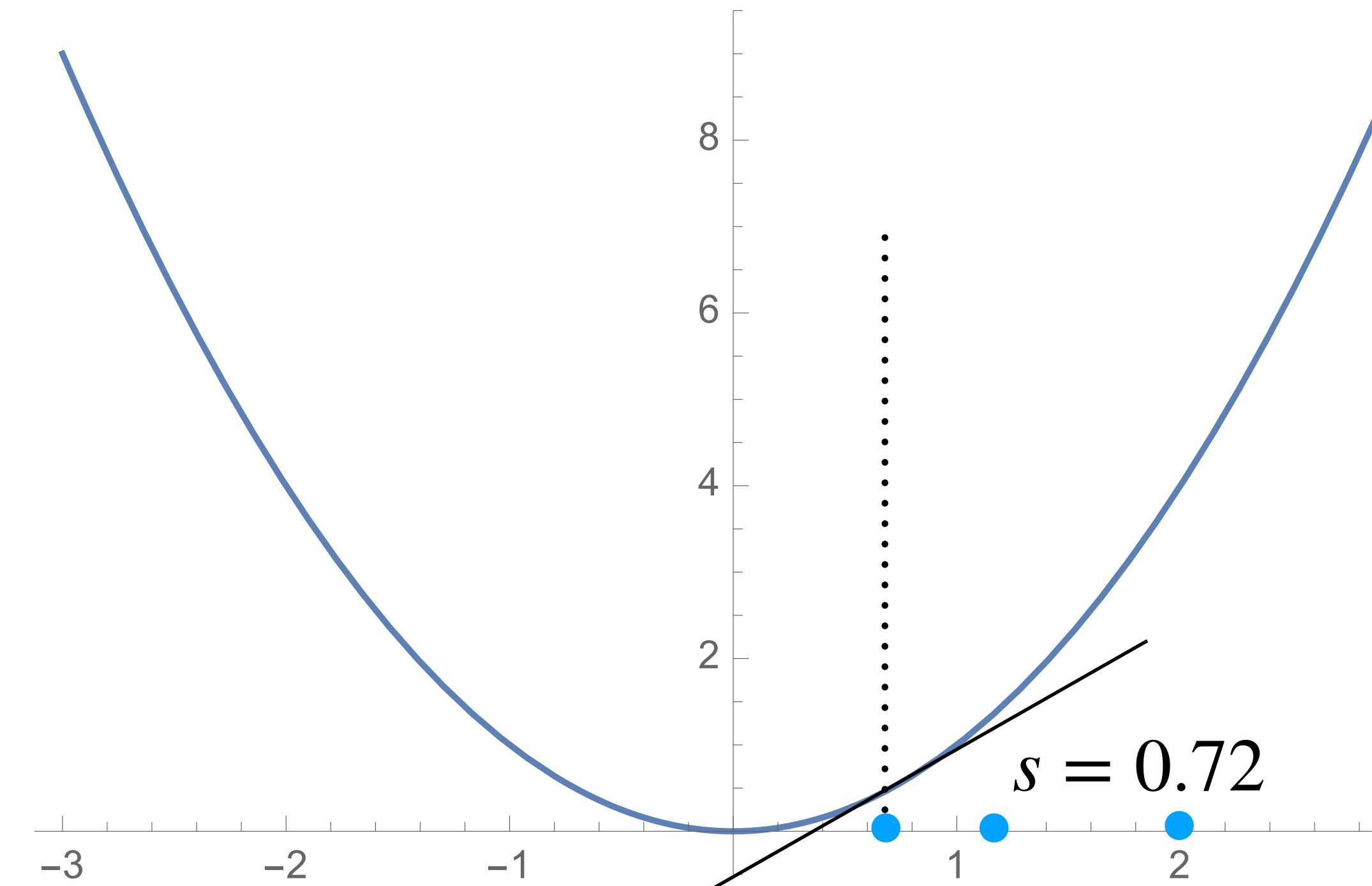
ステップ1 :  $s$  を初期値にセットする

ステップ2 :  $g := f'(s)$  とする

ステップ3 :  $s := s - a g$  とする ( $a$  は学習係数と呼ばれる)

ステップ4 : ステップ2に戻る

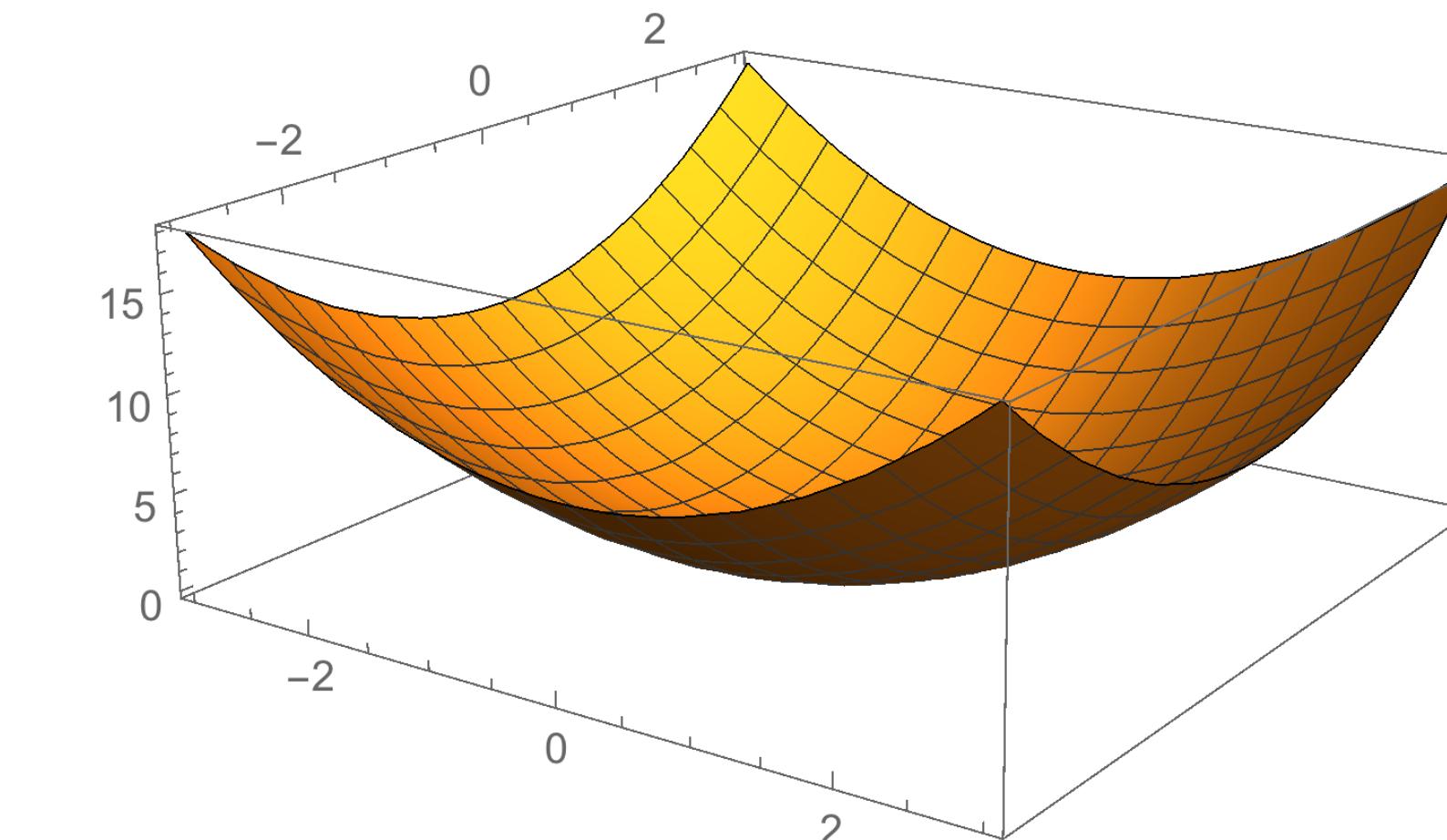
[練習問題] 初期値  $s = 2$ ,  $a = 0.2$  として上の勾配法のステップを実行せよ。



# 多次元の関数の場合には？

深層学習の場合、パラメータは複数ある  
(場合によっては数万パラメータにも及ぶ)

$$f(x_1, x_2) = x_1^2 + x_2^2$$

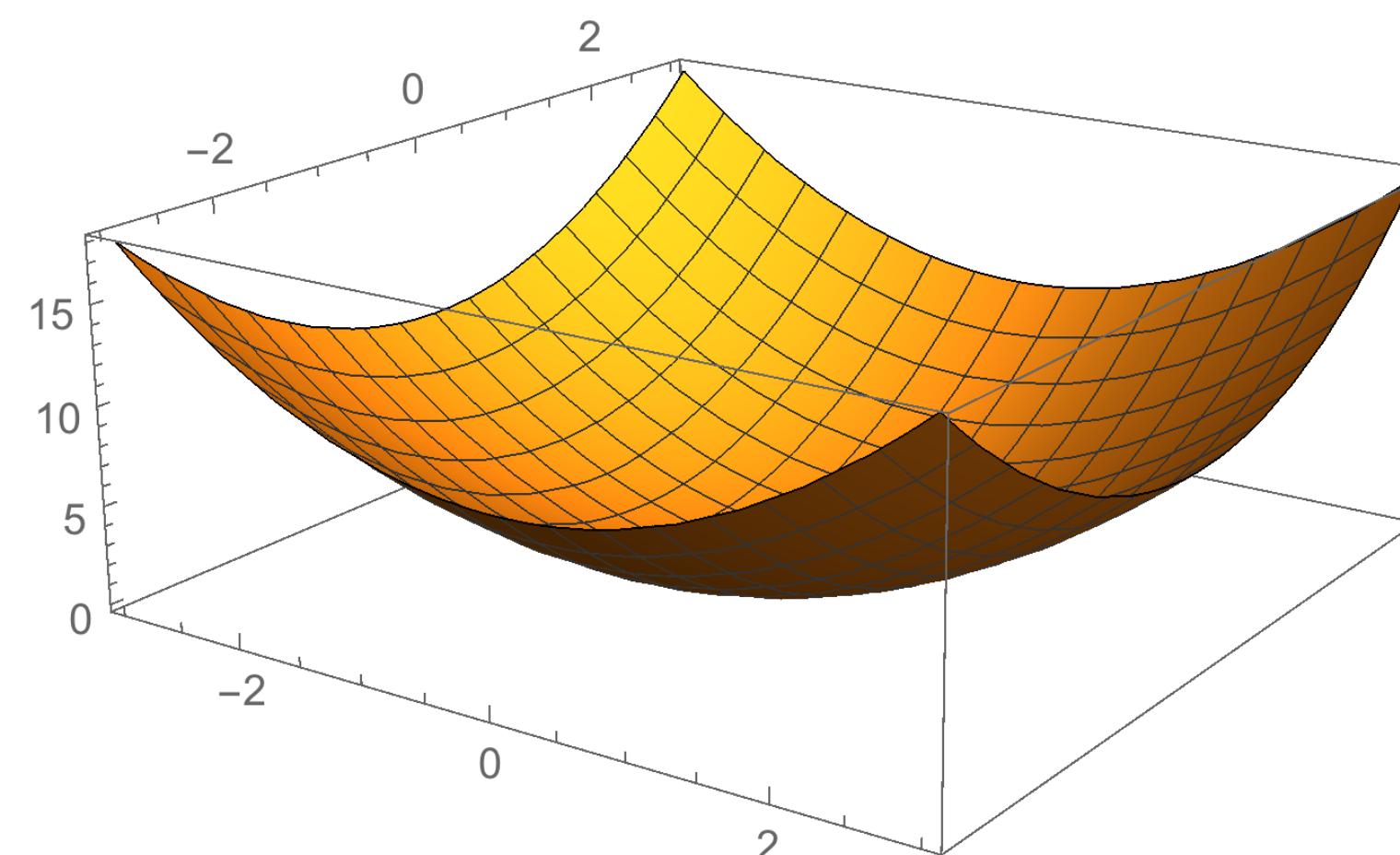


勾配ベクトル

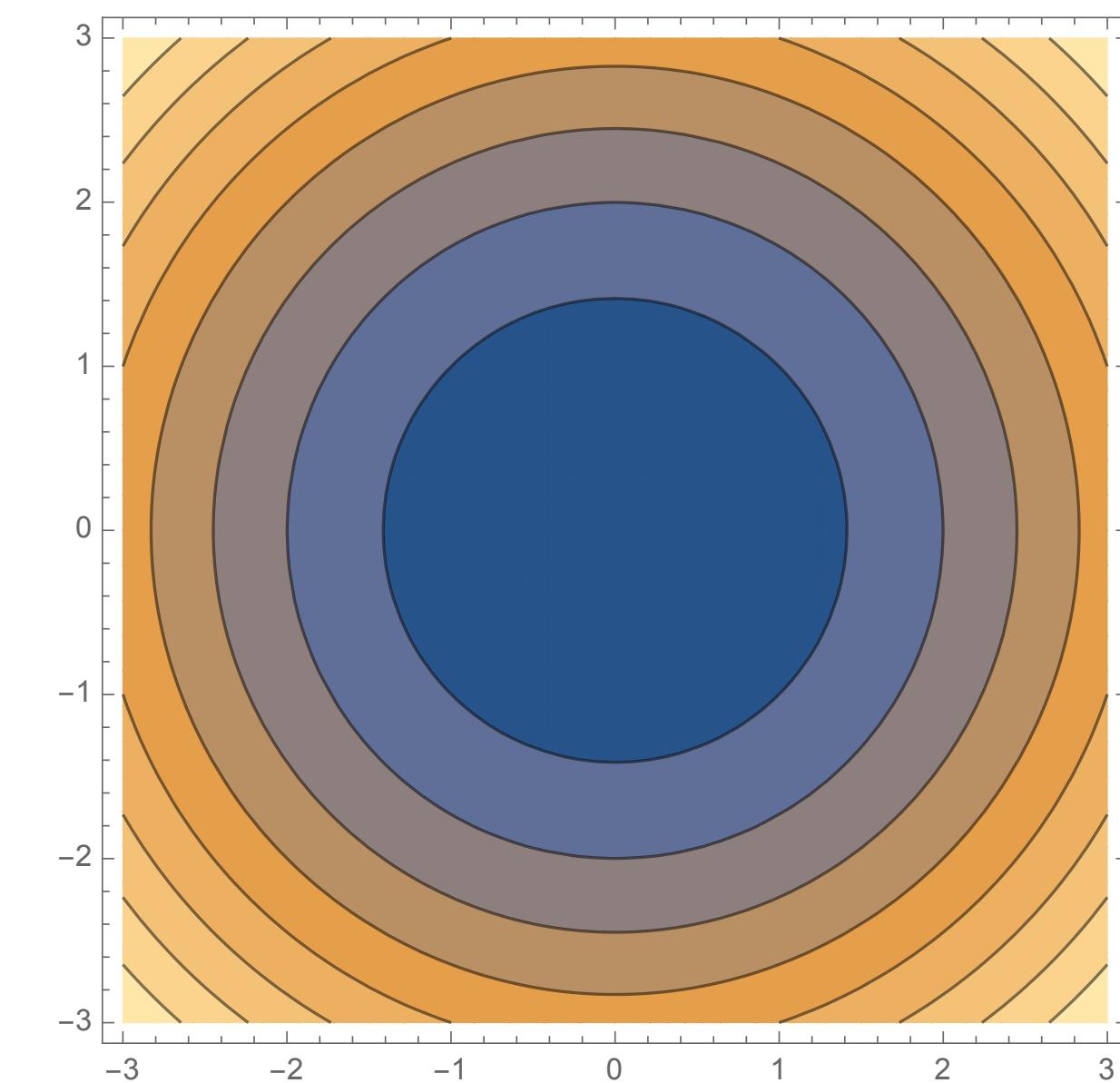
$$\nabla f(x) = \left( \frac{\partial f}{\partial x_1} = 2x_1, \frac{\partial f}{\partial x_2} = 2x_2 \right)^T$$

# 勾配ベクトルと等高線

$$f(x_1, x_2) = x_1^2 + x_2^2$$



3D表示



等高線表示

勾配ベクトル

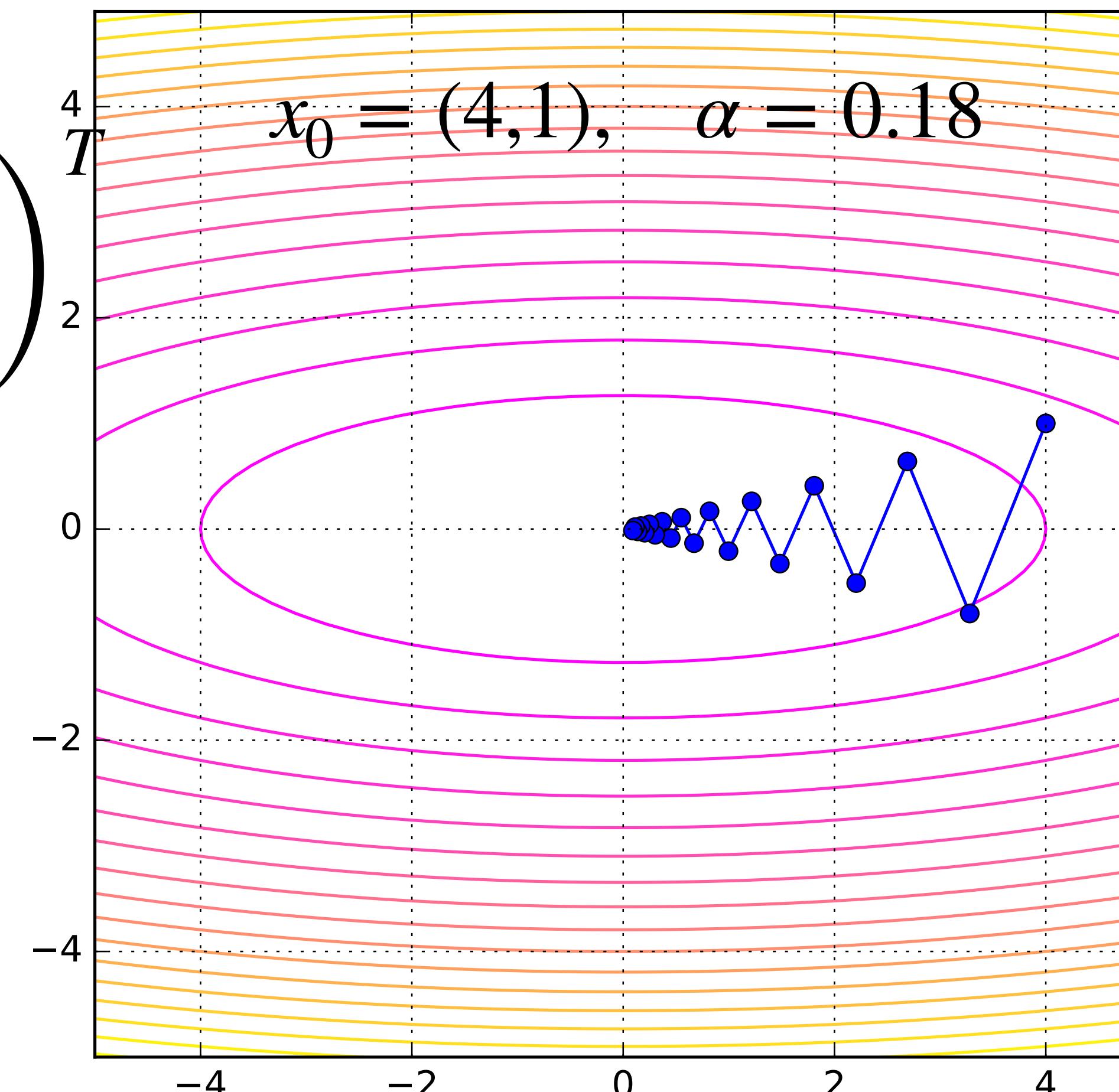
$$\nabla f(x) = \left( \frac{\partial f}{\partial x_1} = 2x_1, \frac{\partial f}{\partial x_2} = 2x_2 \right)^T$$

# 2次元関数における勾配法

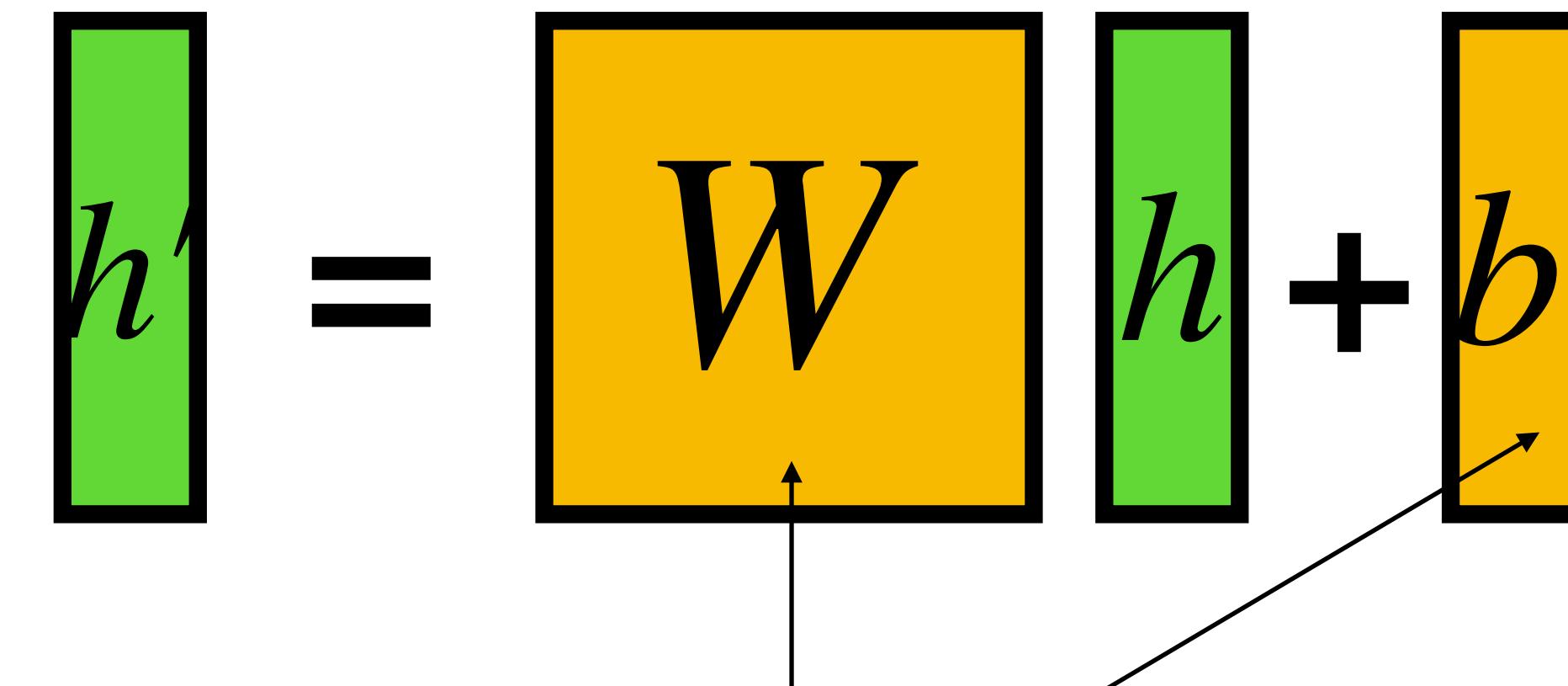
$$f(x_1, x_2) = \frac{1}{2}x_1^2 + 5x_2^2$$

$$\nabla f(x_1, x_2) = \left( \frac{\partial f}{\partial x_1} = x_1, \frac{\partial f}{\partial x_2} = 10x_2 \right)$$

勾配法：  $x_t = x_{t-1} - \alpha \nabla f(x_{t-1})$



# ふたたび：学習プロセス

$$h' = W h + b$$


中身は動かしてよい・うまく  
チューンアップしたい！

二乗誤差関数を最小化するように学習パラメータを動かせばよい

→ $W$ ,  $b$ の各要素の偏微分(勾配ベクトル)を求めて、勾配法  
を使えばよい！

# 中間まとめ

- 深層学習では、**層構造を持つパラメトリック関数モデル**が利用されている
- 各層は、**行列ベクトル積計算（アフィン変換）**と**非線形関数の要素ごとの適用**からなる
- 所望の性能を出すためには、**学習（訓練）プロセス**が必要
- 学習プロセスでは、**大量の訓練データ**が必要
- 学習プロセスでは勾配法(本当は確率的勾配法)を利用して学習パラメータを調節する

# Google Colaboratoryを 使ってみる



# Google Colabratoty

- 機械学習の教育と研究を促進するために  
Google が開発したツール
- 無償でクラウド上のマシンを利用可能
- ブラウザベース
- 環境構築不要でNNフレームワーク(TensorFlow,  
PyTorchが使える)
- GPUが利用できる

# Google Colabを使ってみよう

ブラウザで下記のURLにアクセスする

<https://colab.research.google.com/>

windowsユーザはchromeを使ってください。  
macユーザはsafariでOK

# Google Colabを使ってみよう



# 演習ノートブックと演習課題(1)

本講義の演習課題はipython notebook形式(拡張子 .ipynb )で配布します。下記のレポジトリが入手してください。

- <https://github.com/wadayama/HCU2022>

今回、この講義で利用するのは、1-Colab\_intro.ipynb

# 演習用ノートブックの入手

The screenshot shows a GitHub repository page for 'wadayama / MIKA2019'. The repository contains several files and folders related to machine learning, such as 'wadayama slides', '.ipynb\_checkpoints', 'ANDfunction.ipynb', 'Egg.ipynb', 'ISTA.ipynb', 'MIKA-tutorial\_open.pdf', 'MIKA2019.pdf', 'MIMO.ipynb', 'NoisyGDBF.ipynb', and 'PFGRec504x1008 dec'. A context menu is open over the repository name, with two specific options highlighted by red circles:

- Code** (green button) - This option has a red circle around it.
- Download ZIP** (option circled with a red circle)

The 'Code' button likely provides a direct link to the repository's code view or a specific file. The 'Download ZIP' option allows for a quick download of the entire repository's contents as a compressed archive.

# 演習ノートブックと演習課題(2)

本講義の演習課題はipython notebook形式(拡張子 .ipynb )で配布します。Google Colabにアップロードして使用してください。

- Google Colabに○○.ipynbをアップロード
- 演習の実施(レポート作成のために実行結果や編集結果を残すこと)
- 演習課題については指示に従い、Word, LaTeXなどを利用して作成する(手書き不可)。集中講義後にPDFを提出すること。

# 演習ノートブックのアップロード



# 本講義のまとめ

- 深層学習の概要
- 勾配法による最小化
- Google Colabの基本的な利用法