HOKKAIDO UNIVERSITY

講義「人工知能」
第4回 ニューラルネットワーク2
ニューラルネットワークの学習

北海道大学大学院情報科学研究院
情報理工学部門 複合情報工学分野
調和系工学研究室 准教授 山下倫央
http://harmo-lab.jp
tomohisa@ist.hokudai.ac.jp
2024年4月18日(木)

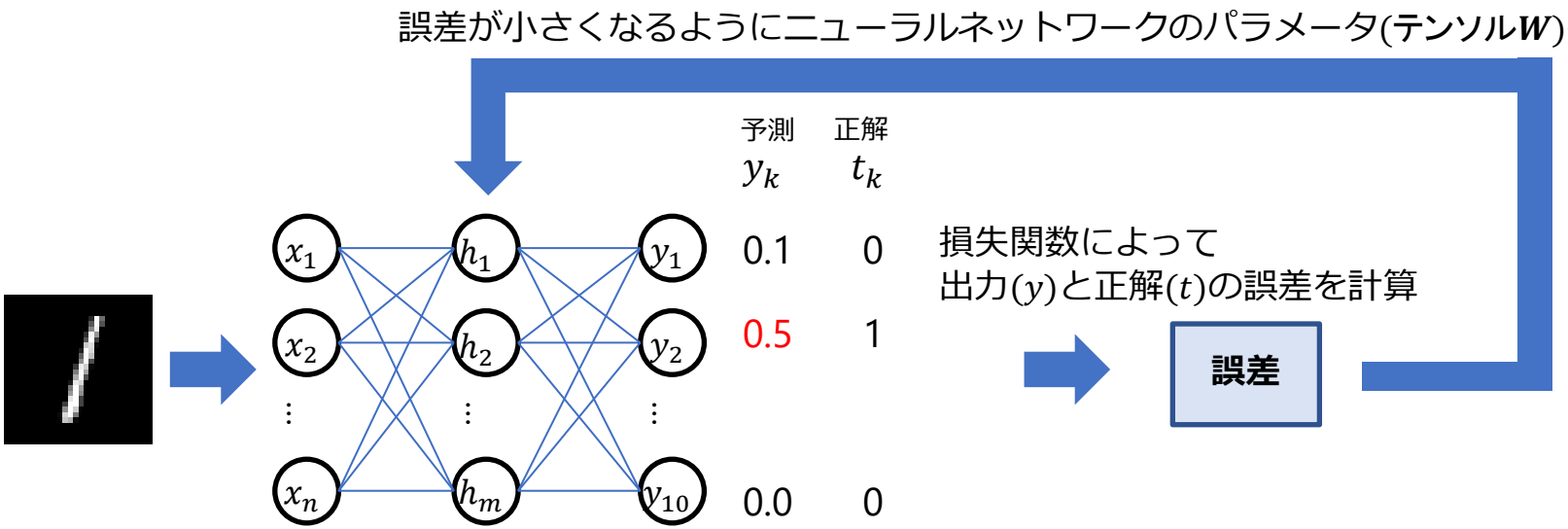
本日の内容

- ニューラルネットワークの学習
  - 重みの更新方法
  - 学習に用いるデータの整備
- ニューラルネットワークを試す
  - A Neural Network Playground
- 第1回課題の説明

ニューラルネットワークの教師あり学習

- ニューラルネットワークの学習とは
  - 損失関数を最小化するパラメータ(テンソル$\boldsymbol{W}$)を求めること

- 損失関数
  - 出力の値(y)と正解の値(t)の誤差を表す関数
    - 間違ってる場合には大きい値、正しい場合には小さい値をとる関数を設定

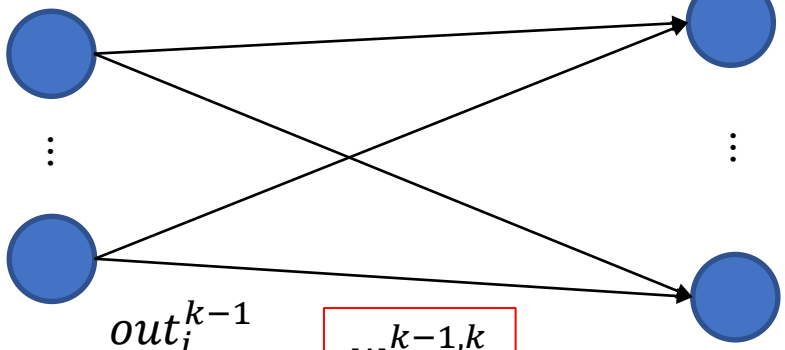


ニューラルネットワークの教師あり学習

1. 入力層に入力信号を与えて，順方向に中間層の出力を計算して，出力層からの出力を計算
2. 出力層からの出力と教師信号に損失関数を適用して誤差を計算
3. 誤差逆伝播法を用いて，出力層から入力層に向かう逆方向に各ユニット間の重みに対する損失関数の勾配を計算
4. この勾配を用いた誤差を最小化するために，最急降下法を適用して各ユニット間の重みを更新
5. 1-4 の繰り返し

記号の準備

ニューラルネットワーク



第 k–1層 i 番目の
ニューロンの出力

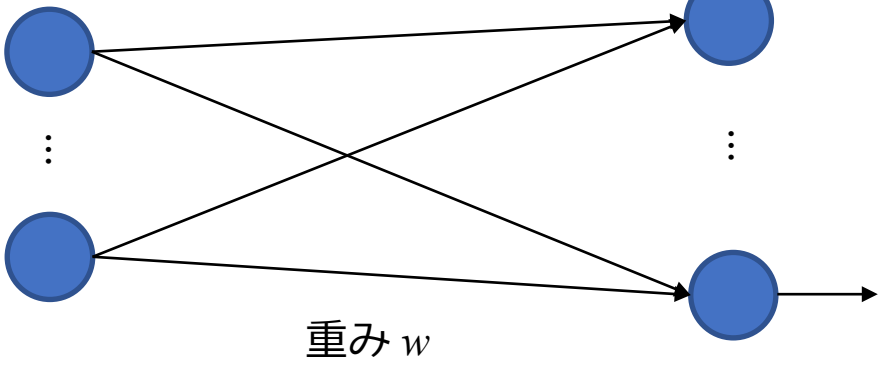
第 k–1層 i 番目ニューロンと
第 k 層 j 番目ニューロンの
間の重み

第 k 層 j 番目の
ニューロンの入力

$$in_i^k = \sum_{l=1}^{N_{k-1}} w_{lj}^{k-1,k} out_l^{k-1}$$

記号の準備

- ニューラルネットワーク



対応する教師信号

t_i

二乗誤差関数を用いた場合の誤差

$$E = \frac{1}{2}\sum_{l=1}^{N_M}(t_l - out_l^M)^2 \qquad 1 \le l \le N_M$$

誤差関数

- 誤差関数：モデルの出力と答えから計算される関数であり、学習がうまくいくほどゼロに収束する性質を持つ

- クラス分類
  - ソフトマックスクロスエントロピー関数

$$\mathrm{H}(p,q) = -\sum_{x} p(x)\log(q(x))$$

p：真の確率
q：推定した確率

- 回帰
  - 平均2乗誤差(MSE)

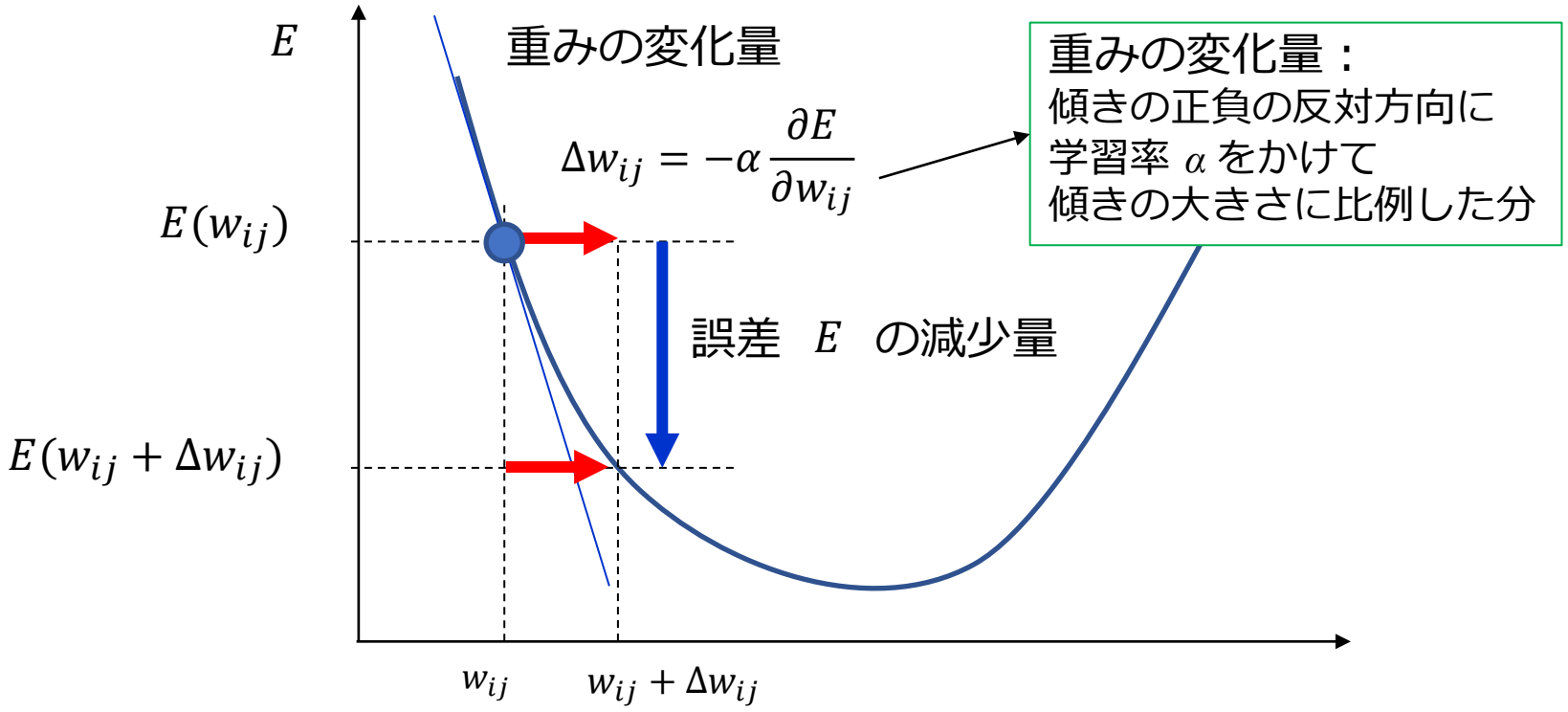
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$\hat{y}_i$ = 予測値
y_i = 正解値

ニューラルネットワークの学習

- 最急降下法
  - 重みを操作して、誤差 E を最小化

重み w_{ij} をどのように変更するか？



ニューラルネットワークの学習

- 重みの更新式

$$\Delta w_{ij}^{k-1,k} = -\alpha \frac{\partial E}{\partial w_{ij}^{k-1,k}}$$

連鎖律の適用

$$= -\alpha \frac{\partial E}{\partial in_j^k} \frac{\partial in_j^k}{\partial w_{ij}^{k-1,k}}$$

入力の定義

$$in_j^k = \sum_{l=1}^{N_{k-1}} w_{lj}^{k-1,k} out_l^{k-1} \qquad 1 \le l \le N_{k-1}$$

$$= -\alpha \frac{\partial E}{\partial in_j^k} \frac{\partial (\sum_{l=1}^{N_{k-1}} w_{lj}^{k-1,k} out_l^{k-1})}{\partial w_{ij}^{k-1,k}}$$

$$= -\alpha \frac{\partial E}{\partial in_j^k} out_i^{k-1}$$

$\sum_{l=1}^{N_{k-1}} w_{lj}^{k-1,k} out_l^{k-1}$ を $w_{ij}^{k-1,k}$ で微分

ニューラルネットワークの学習

- 重みの更新式

$$\Delta w_{ij}^{k-1,k} = -\alpha \frac{\partial E}{\partial in_j^k} out_i^{k-1}$$

$\delta_i^k = -\frac{\partial E}{\partial in_j^k}$ とする

$$= \alpha \delta_i^k \, out_i^{k-1}$$

ニューラルネットワークの学習

- 重みの更新式

$$\delta_i^k = -\frac{\partial E}{\partial in_j^k}$$

連鎖律の適用

$$= -\frac{\partial E}{\partial out_j^k}\frac{\partial out_j^k}{\partial in_j^k}$$

ニューロンの出力の定義

$$out_j^k = f\left(\sum_{l=1}^{N_k} w_{lj}^{k-1,k} in_l^k\right)$$

$$= -\frac{\partial E}{\partial out_j^k}\frac{\partial f(in_j^k)}{\partial in_j^k}$$

$$= -\frac{\partial E}{\partial out_j^k} f'(in_j^k)$$

ニューラルネットワークの学習

- 第 M 層(出力層) の場合

誤差の定義から　$E = \frac{1}{2}\sum_{l=1}^{N_M}(t_l - out_l^M)^2$　なので　$\frac{\partial E}{\partial out_i^M} = t_i - out_i^M$

$$\Delta w_{ij}^{k-1,k} = -\alpha \frac{\partial E}{\partial in_j^k} out_i^{k-1}$$

$$= \alpha \delta_i^k \, out_i^{k-1}$$

$$= \alpha \left(-\frac{\partial E}{\partial out_j^k} f'(in_j^k)\right) \, out_i^{k-1}$$

$$= \alpha \left(-1 \times (t_i - out_i^M) \, f'(in_j^k)\right) \, out_i^{k-1}$$

$$= -\alpha (t_i - out_i^M) \, f'(in_j^k) \, out_i^{k-1}$$

ニューラルネットワークの学習

- 中間層 の場合

$$\frac{\partial E}{\partial out_i^k} = \sum_{l=1}^{N_{k+1}} \left(\frac{\partial E}{\partial out_l^{k+1}} \right)$$

$$\frac{\partial E}{\partial out_i^k} = \sum_{l=1}^{N_{k+1}} \left(\frac{\partial E}{\partial in_j^{k+1}} \frac{\partial in_j^{k+1}}{\partial out_i^k} \right)$$

入力の定義

$$= \sum_{l=1}^{N_{k+1}} \left(\frac{\partial E}{\partial in_j^{k+1}} \frac{\partial (\sum_{l=1}^{N_{k+1}} w_{lj}^{k,k+1} out_l^k)}{\partial out_i^k} \right)$$

$$= \sum_{l=1}^{N_{k+1}} \left(\frac{\partial E}{\partial in_j^{k+1}} w_{lj}^{k,k+1} \right)$$

$$= -\sum_{l=1}^{N_{k+1}} \left(\delta_l^{k+1} w_{lj}^{k,k+1} \right)$$

ニューラルネットワークの学習

- 中間層 の場合

$$\Delta w_{ij}^{k-1,k} = -\alpha \frac{\partial E}{\partial in_j^k} out_i^{k-1}$$

$$= \alpha \delta_i^k \; out_i^{k-1}$$

$$= \alpha \left(-\frac{\partial E}{\partial out_j^k} f'(in_j^k)\right) \; out_i^{k-1}$$

$$\Delta w_{ij}^{k-1,k}$$

$$= \alpha \left(-\left(-\sum_{l=1}^{N_{k+1}} \left(\delta_l^{k+1} w_{lj}^{k,k+1}\right)\right) f'(in_j^k)\right) \; out_i^{k-1}$$

$$= \alpha \left(\sum_{l=1}^{N_{k+1}} \left(\delta_l^{k+1} w_{lj}^{k,k+1}\right)\right) f'(in_j^k) \; out_i^{k-1}$$

$\Delta w_{ij}^{k-1,k}$を計算するために δ_l^{k+1} が必要

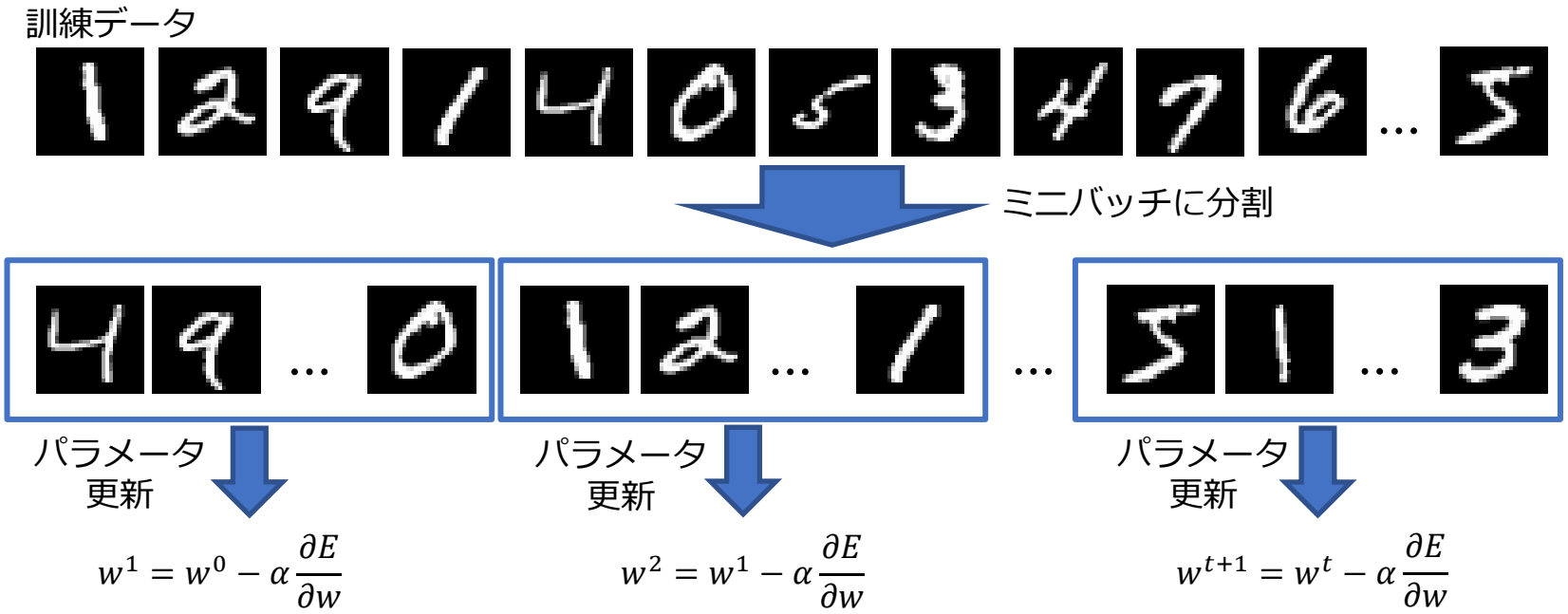
ある層の重みを更新する場合には、
1つ入力側の δ が必要 → 出力側から入力側に順々に重みを更新する

ニューラルネットワーク

- 学習には以下の3つのデータが必要なため、持っているデータを分割する必要がある
  - **訓練データ**：モデルのパラメータを見つけるためのデータ
    - 主に全データの6～8割が用いられる
  - **検証データ**：ハイパーパラメータ(人が決定すべき項目)を検証するためのデータ
    - 全データの1～2割程度
  - **テストデータ**：学習済みモデルの汎化性能を評価するためのデータ(実運用で想定される精度がここでわかる)
    - 全データの1～2割程度

ニューラルネットワーク

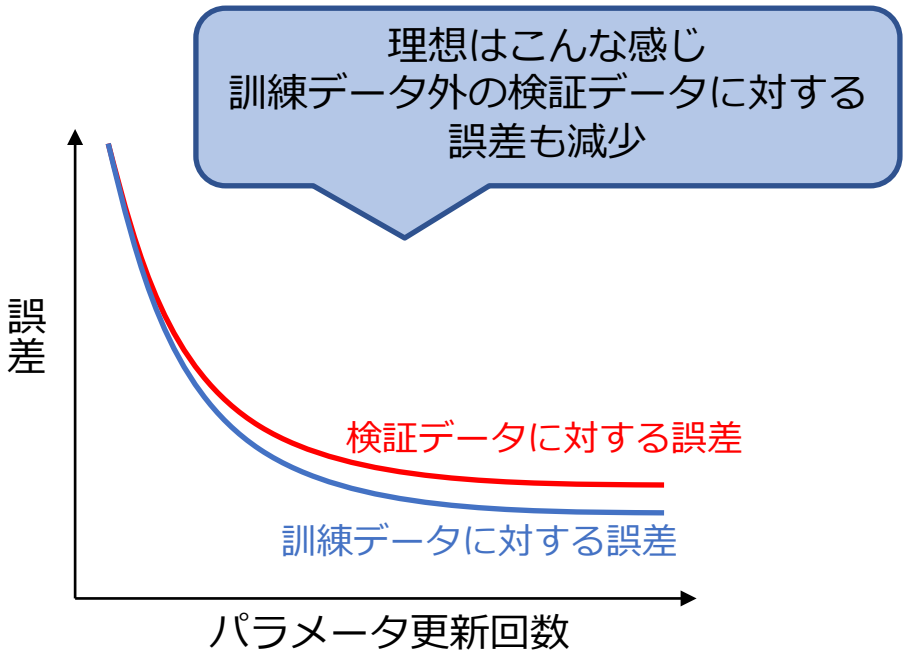
- 実際の学習の時には，訓練データをミニバッチに分割しミニバッチごとにパラメータの更新を行う
  - ミニバッチについて
    - 訓練データをn個のデータからなるグループに分割する
      - 各グループをミニバッチと呼ぶ
      - 分割はランダムに行う
    - 各ミニバッチのデータ数(n)のことをバッチサイズと呼ぶ

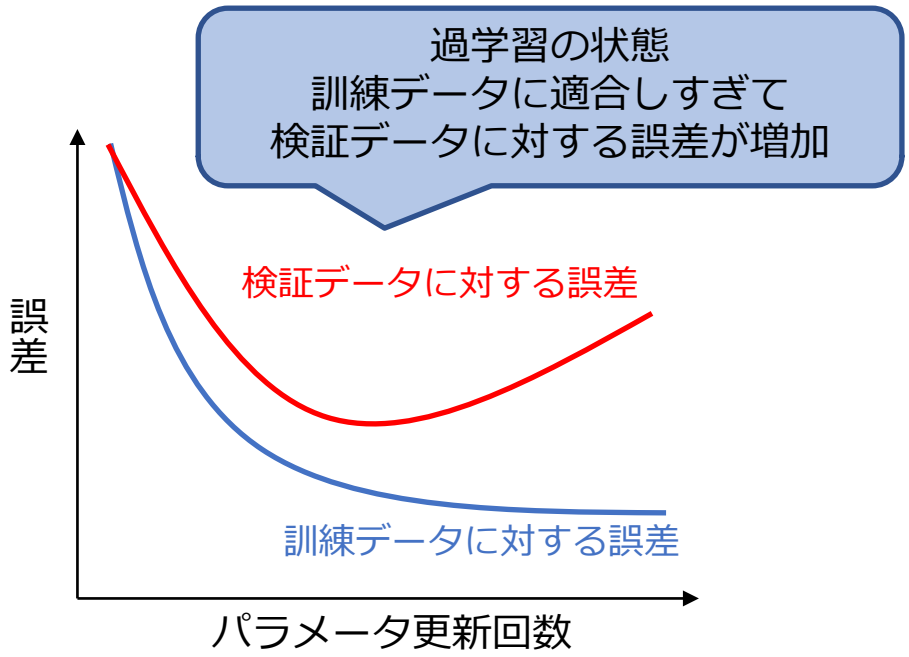


ニューラルネットワーク

- 学習の真の目的
  - 訓練データだけではなく，未知のデータに対しても正しく予測できるようにすること

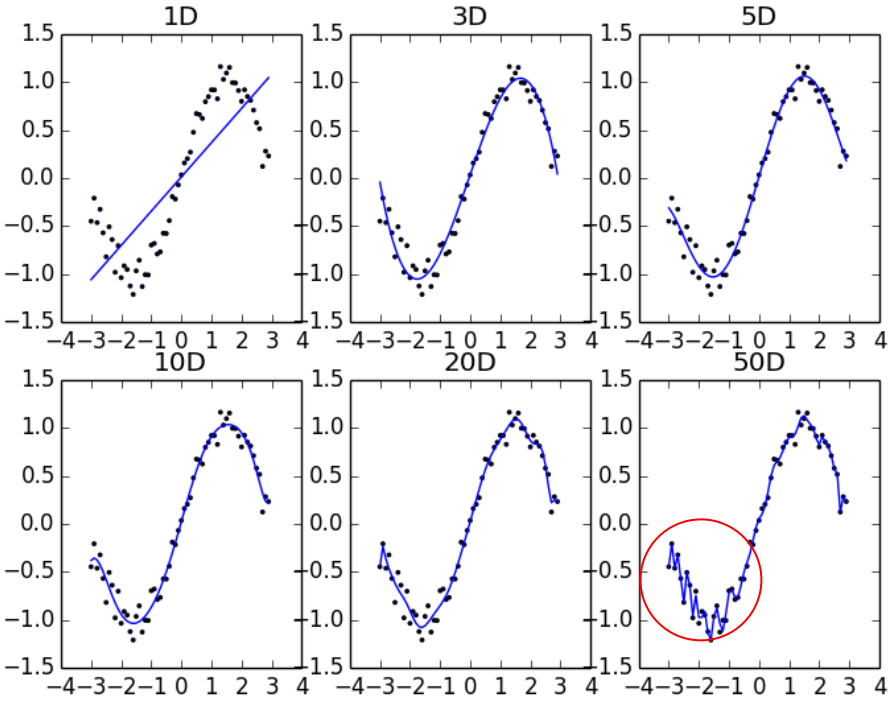
- 学習を行っていくと訓練データに対する誤差は減少していく
- 学習中は訓練データに対する誤差だけではなく，**検証データ**に対する誤差を見る必要ある





過学習

- 過学習(Overfitting=過適合)
  - 訓練データの一般的ではない、特定のランダムな（本来学習させたい特徴とは無関係な）特徴にまで学習してしまうこと



例）$y=\sin(x)$ の形状に分布するデータ列を表現するのに，
それぞれ1, 3, 5, 10, 20, 50次元の多項式でフィッティングした結果

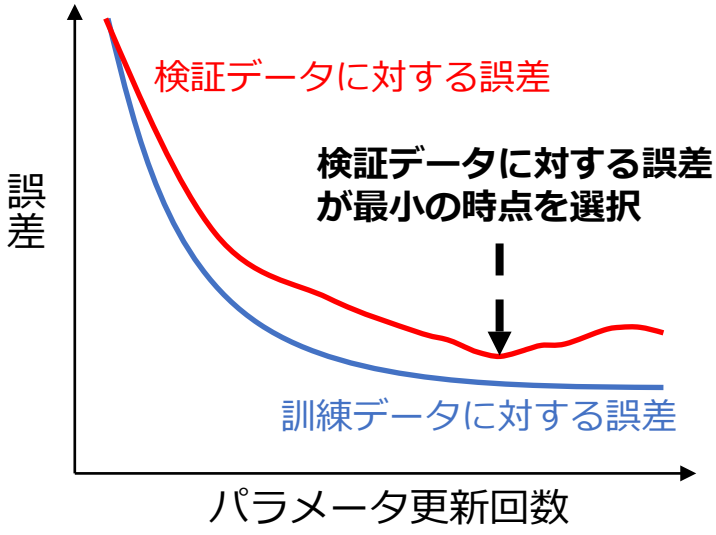
http://sig.tsg.ne.jp/ml2015/ml/2015/06/29/techniques-of-learning.html

ニューラルネットワーク

- 訓練データの一部を検証データとして使用
  - 検証データはモデルの訓練自体には使用しない
  - 検証データはモデルの評価(誤差、精度計算)に使用する
    - テストデータとは別

検証データの使用例

- ニューラルネットワークの学習推移を確認



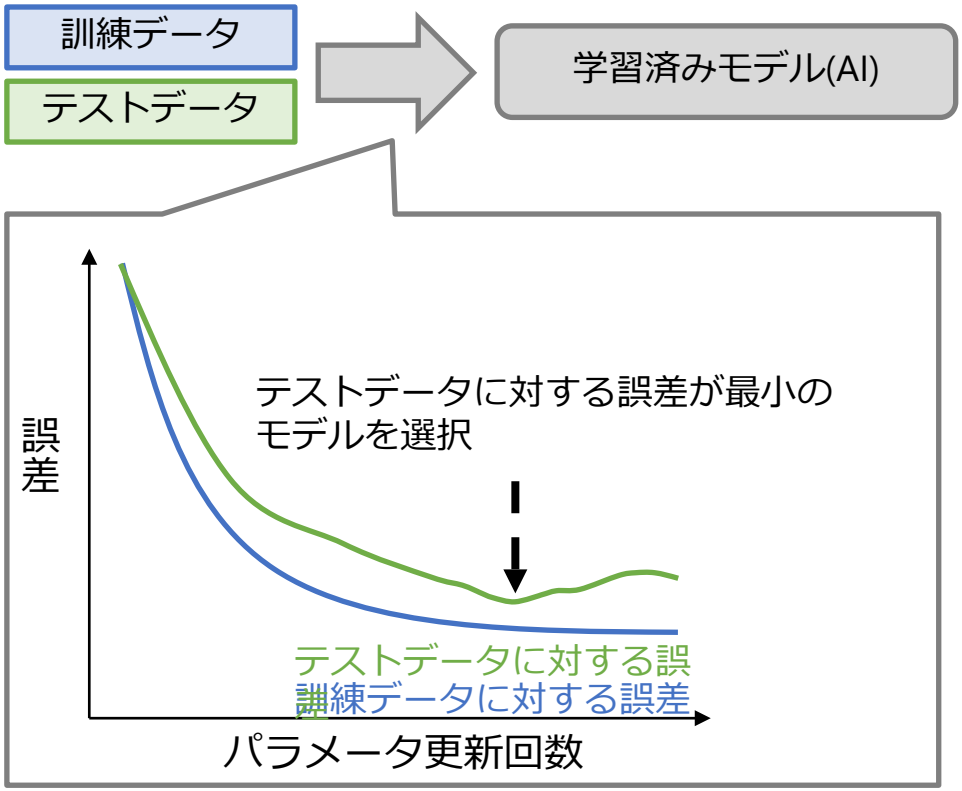
- 複数のモデルを比較し採用するモデルを選択

モデル	検証誤差
モデルA	0.2
モデルB	**0.1**
モデルC	0.3
モデルD	0.5

採用！

ニューラルネットワーク

- 学習の真の目的
  - 訓練データだけではなく，未知のデータに対しても正しく予測できるようにすること



テストデータに対する精度：90%

この学習済みモデルは未知のデータに対しても90%の精度を示すと言えるか？

テストデータに対して良い精度を示すモデルを選択してきている

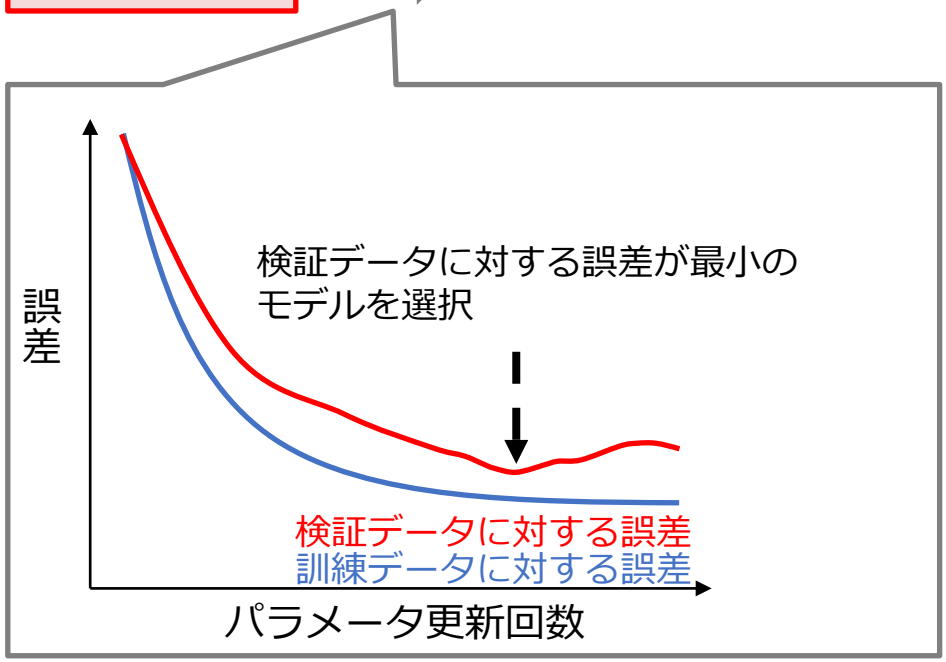
テストデータはモデルの学習、選択には使用してはいけない

- 学習の真の目的
  - 訓練データだけではなく，未知のデータに対しても正しく予測できるようにすること

訓練データ
検証データ

学習済みモデル(AI)

テストデータに対する精度：90%



モデルの学習に一切使用していない
テストデータで性能を評価することで
未知のデータに対する性能を評価する

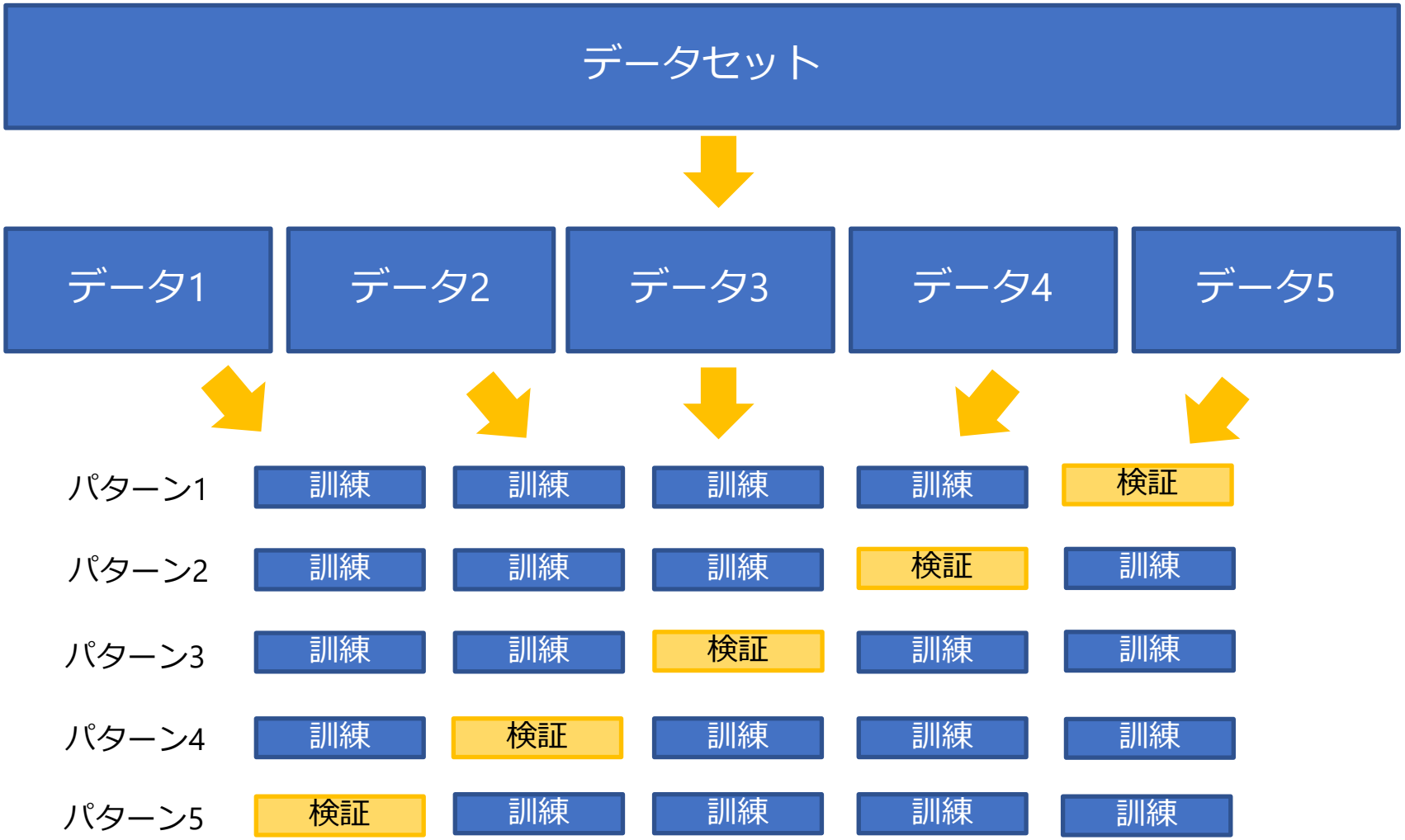
（学習中のモデル評価やモデルの選択には
検証データを使用する必要がある）

データセットの分割

- データは訓練・検証・テストに分ける必要がある
- ただし、**たまたま分類しやすいデータ**が検証データに集まってしまう場合、うまくモデルを評価できない(※逆のパターンも然り)
- そこで、訓練と検証を複数パターン試す“交差検証(クロスバリデーション)”を行うことでこの問題を解決する

5分割交差検証

- 分割されたデータは1度は必ず検証する



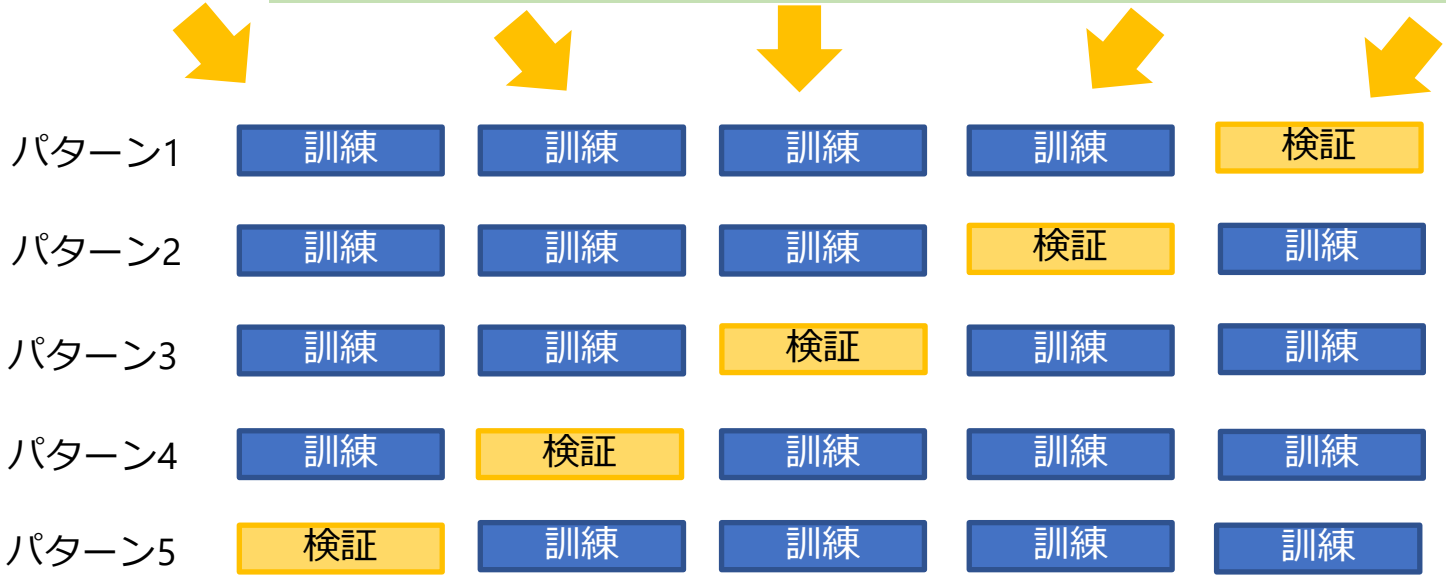
5分割交差検証

- 分割されたデータは1度は必ず検証する

ポイント

- 1パターンのみで実験して結果が良くても、たまたまデータの組み合わせが良いだけだったかも？
- 正しい汎化性能の検証は必ずたくさんのパターンを比較すること！

データ1



本日の内容

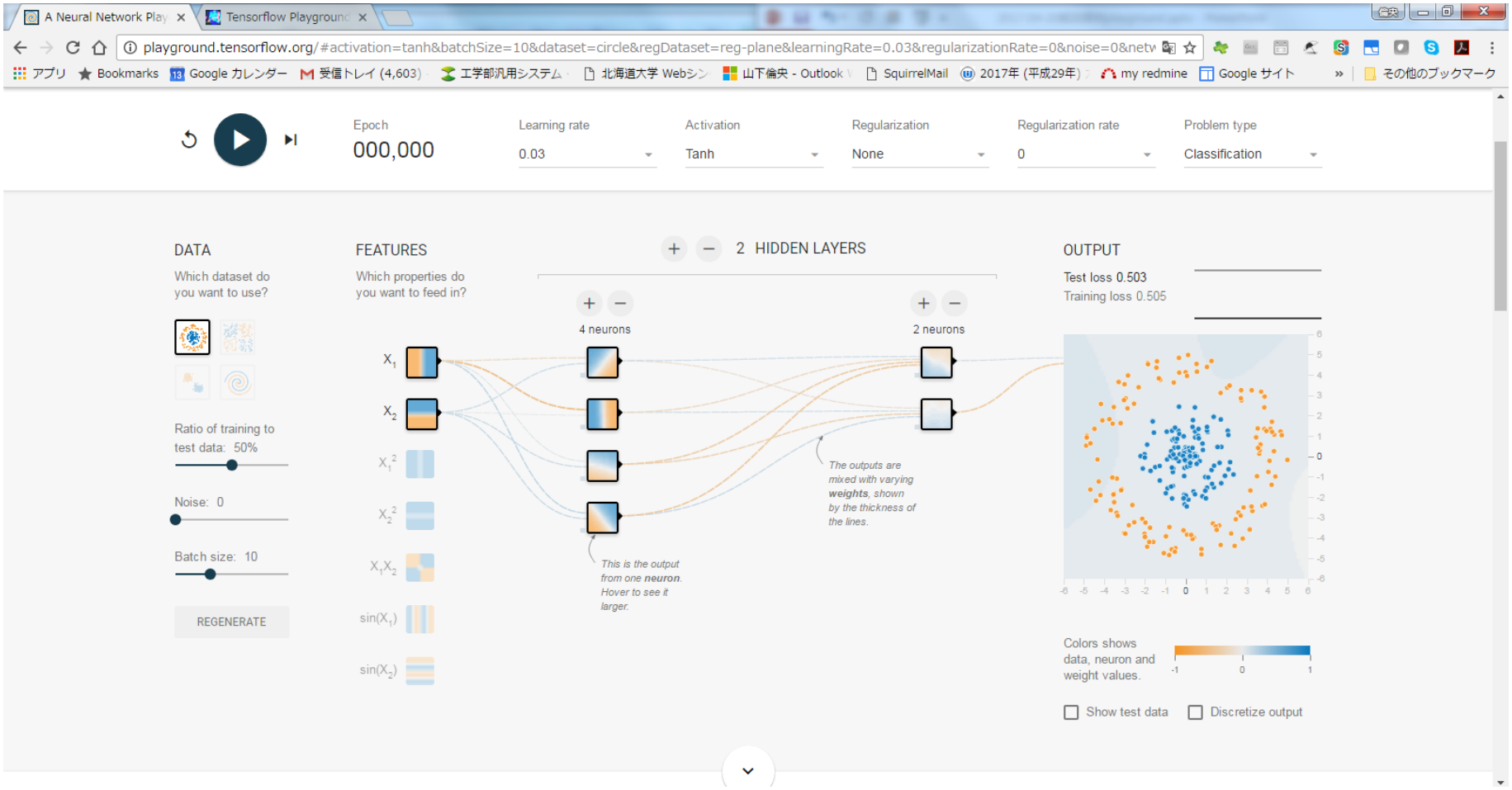
- ニューラルネットワークの学習
  - 重みの更新方法
  - 学習に用いるデータの整備
- ニューラルネットワークを試す
  - A Neural Network Playground
- 第1回課題の説明

本日の内容

- AI 関連ニュースの紹介
- 教師あり学習におけるモデル構築
- 教師あり学習の可視化
  - Neural Network Playground
- モデルの評価
  - 2クラス分類
  - 多クラス分類

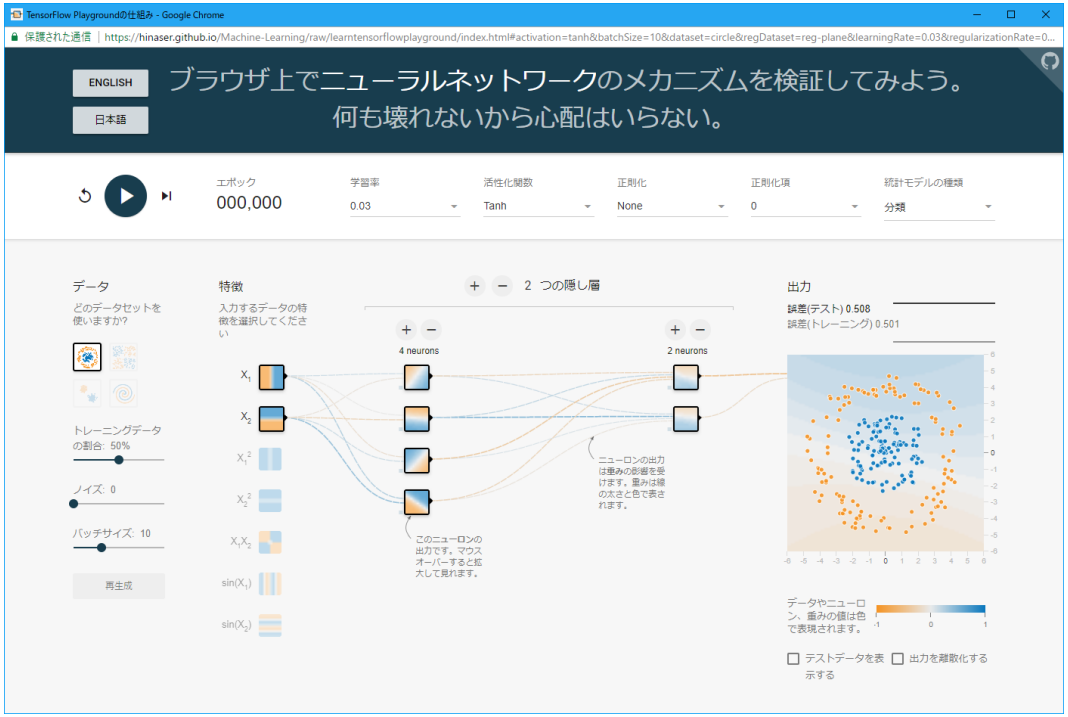
A Neural Network Playground

- http://playground.tensorflow.org/



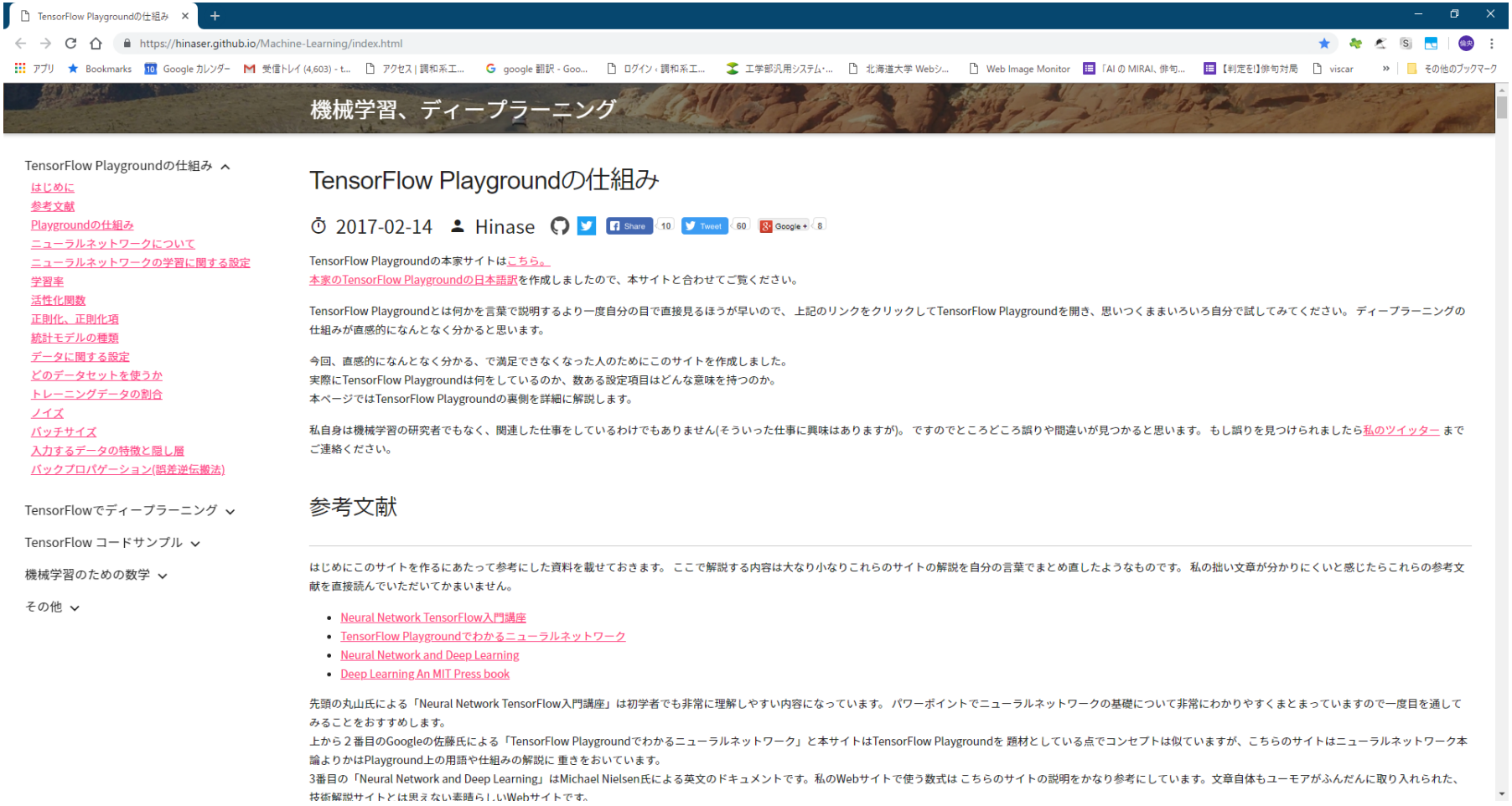
A Neural Network Playground日本語版

- 日本語版「TensorFlow Playgroundの仕組み」で検索
- https://hinaser.github.io/Machine-Learning/raw/learntensorflowplayground/index.html#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.73678&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false&lang=ja

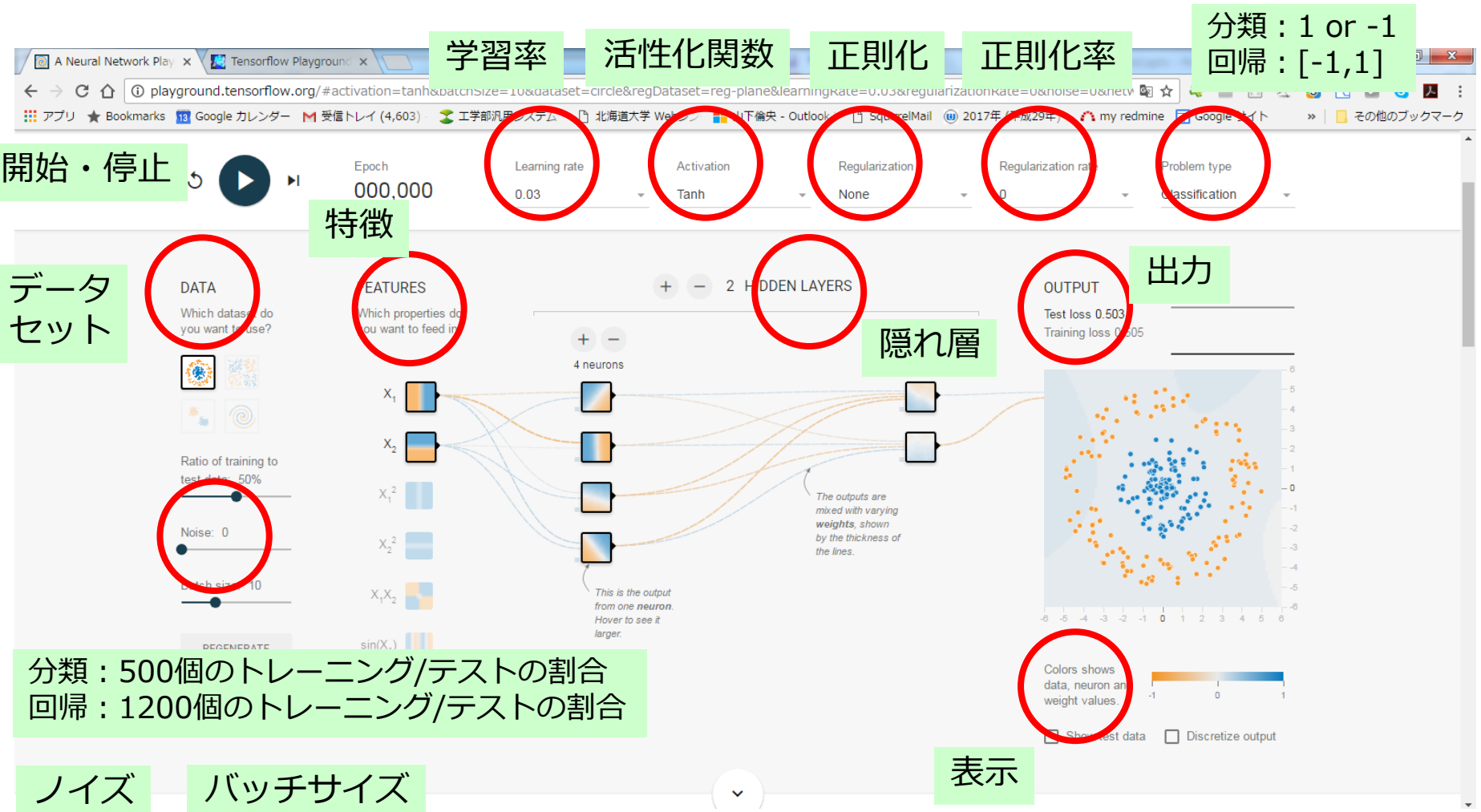


A Neural Network Playground 日本語版解説

- https://hinaser.github.io/Machine-Learning/index.html



A Neural Network Playground



統計モデルの種類

- 分類
  - ラベルは離散値
    - 0 or 1

    ※ 多クラス分類問題もある
  - 例）入力した画像が猫であるか猫でないかを0/1で判断
- 回帰
  - ラベルは連続値
  - 例）入力した画像が猫である確率

データの特徴

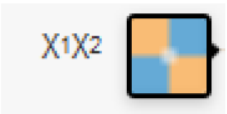
- どの特徴を選択するか？
  - 問題に適した入力データを選択することで、少ないデータでも高精度・高速度の学習が可能になる。

$i_x = x$

$i_y = y$

$i_{x^2} = x^2$

$i_{y^2} = y^2$



$i_{xy} = xy$

$i_{\sin x} = \sin x$

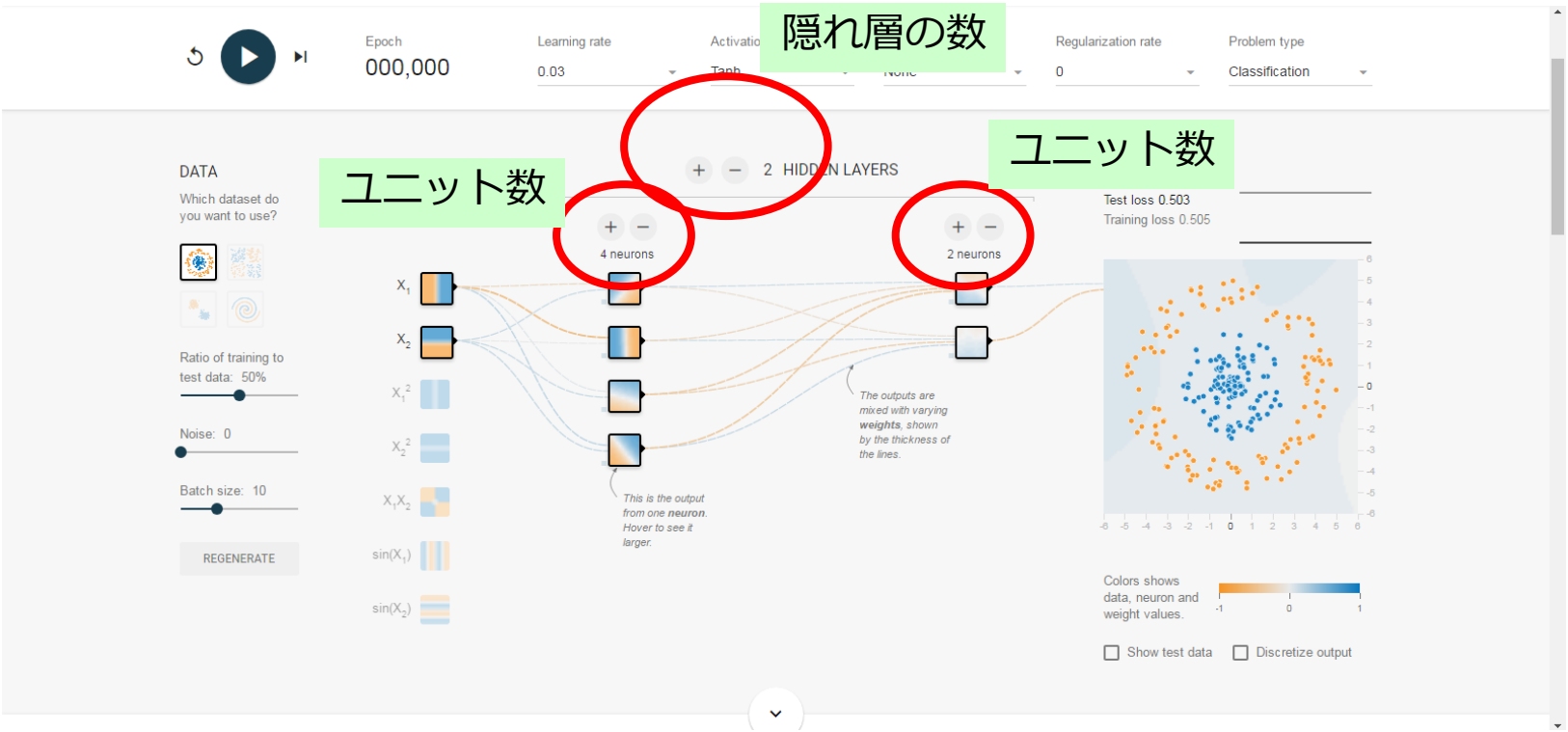
$i_{\sin y} = \sin y$

https://hinaser.github.io/Machine-Learning/index.html

隠れ層

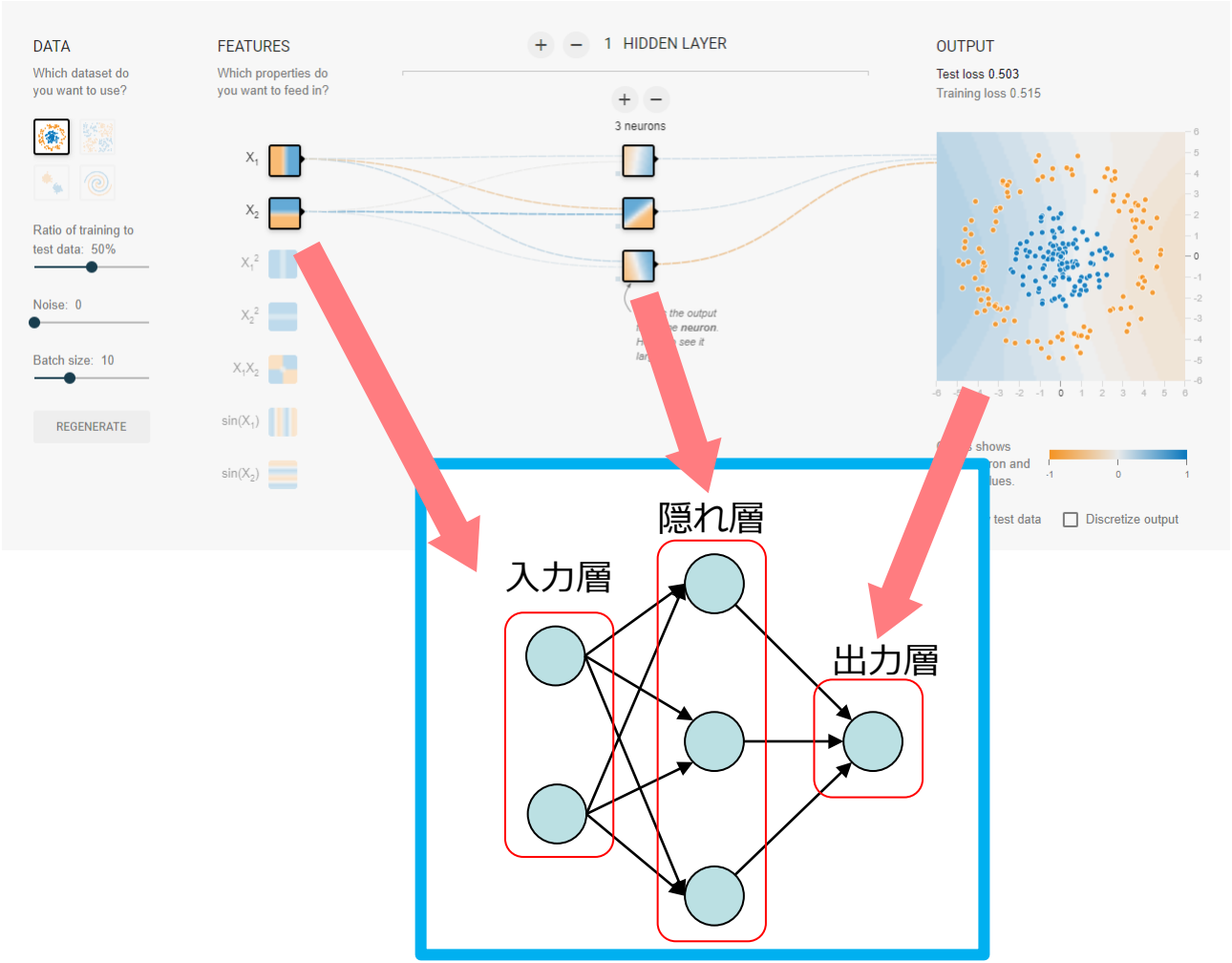
- 隠れ層の設定
  - 層の数
  - 各層のユニット数

「＋」「－」ボタンで調整



隠れ層

- 入力層・隠れ層・出力層の対応



隠れ層

- 隠れ層の設定の目安

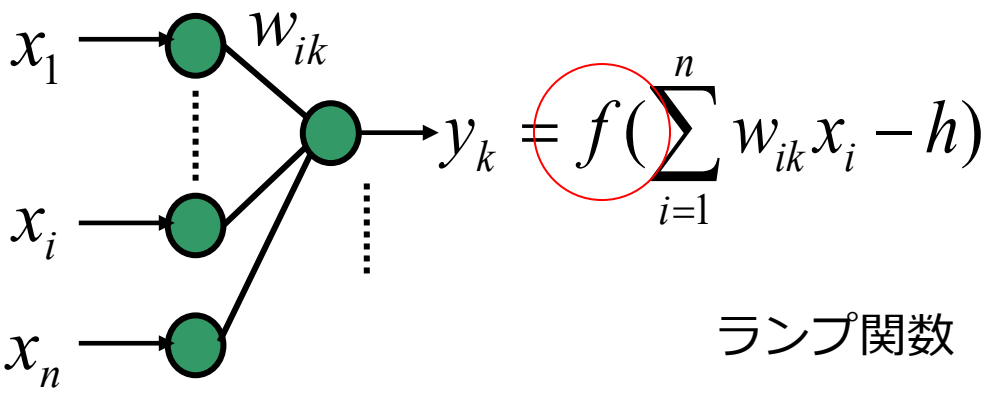
  ※playground ではなく一般的な設定

  - 全結合ネットワークでシンプルな問題を扱う場合
    - 層数
      - まずは 3層(中間層1層) で対応
    - ニューロン数
      - 入力層の次層：入力層のユニット数の2倍
      - 出力層の前層：出力層のユニット数の2倍

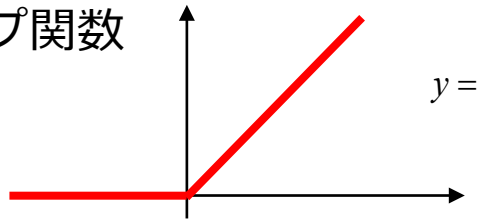
  - データが十分にある場合
    - 層数・ニューロン数を増やしても精度は上がる(下がらない)
  - データが十分にない場合
    - 層数・ニューロン数を増やし過ぎると、学習不足が発生

      → 精度の低下を招く
      - 設定する重みの数が増えてしまうため

活性化関数

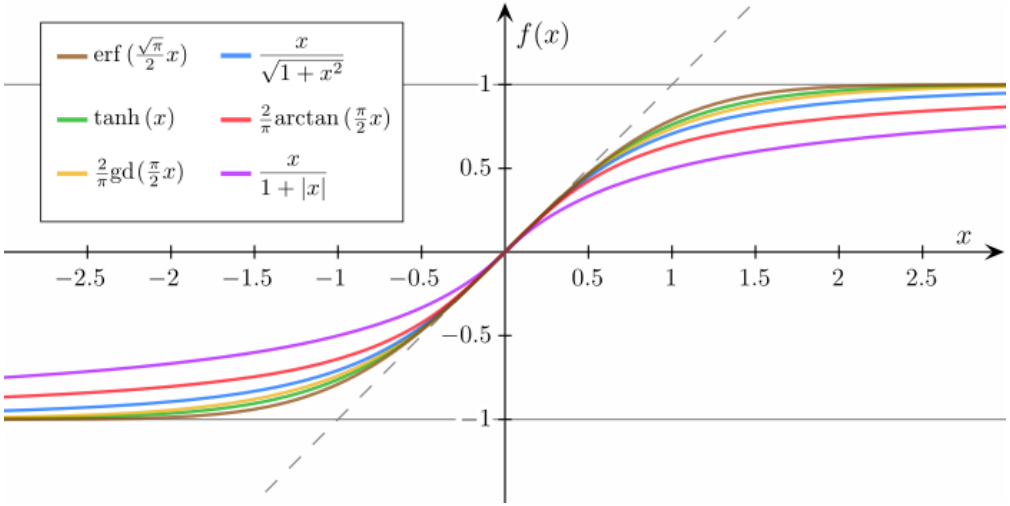


ランプ関数



$$y = \begin{cases} x & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

- ReLU（ランプ関数）
- tanh
- シグモイド関数
- 線形



https://upload.wikimedia.org/wikipedia/commons/6/6f/Gjl-t%28x%29.svg

正則化

- 正則化
  - 過学習を抑えるための手法
    - 重みに対して制約を加える
  - 正則化項 λR(w) を導入 → E(w)+λR(w)
    - λは正則化の強さを決める度合い

    - L1正則化
      - パラメータ w の成分をできるだけ 0とするための制約
      - E(w)+λ|w|　(|w| は重みの総和)
        - » λ は静的なパラメータ
    - L2正則化
      - パラメータ w の成分が大きくなることを防ぐ制約
      - $E(w)+\lambda|w|^2$

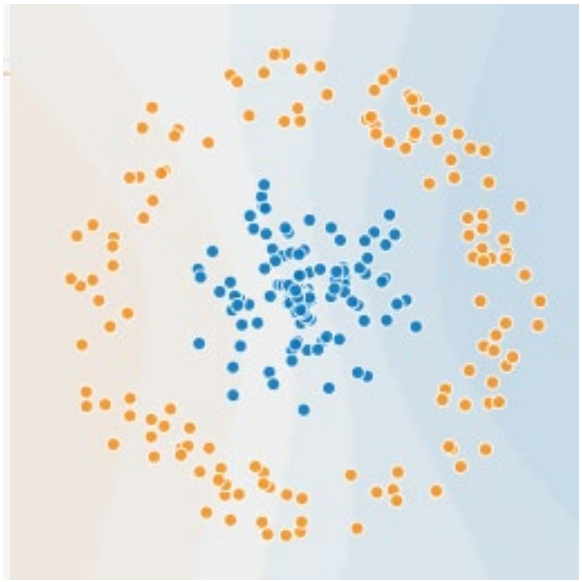
データサンプル

- データサンプル
  - トレーニングデータとテストデータ
    - 入力に対して答えの出力が分かっているデータ
  - サンプル数
    - 分類 500個
    - 回帰 1200個

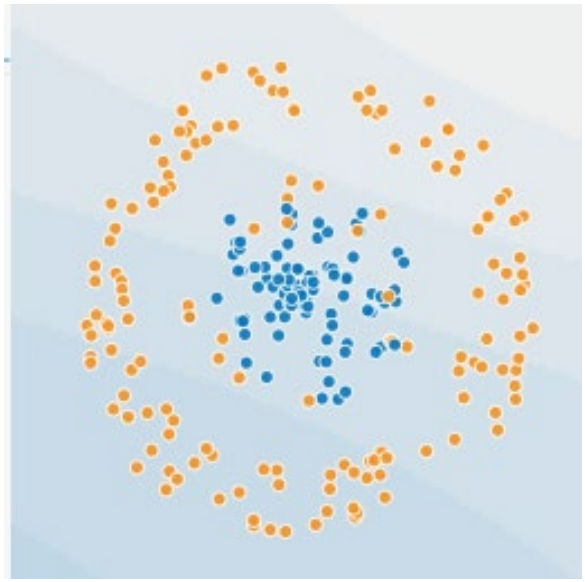
- トレーニングデータ
  - 重みやバイアスの修正に利用
- テストデータ
  - 誤差を計算に利用

ノイズ

- データセットに対してかけるノイズ



ノイズ： 0



ノイズ：20

ノイズ：40

ノイズの値の増加と共にデータのばらつきが大きくなる

バッチサイズ

- バッチサイズ
  - バッチサイズは1回の学習(重み、バイアスの更新)を何個のトレーニングデータで実施するかを決める数値
    - バッチサイズ 1の場合
      - トレーニングデータを1つニューラルネットワークに入れるたびに全ニューロンの重み、バイアス を更新
    - バッチサイズ 10の場合
      - 10個のトレーニングデータにつき1回パラメータを更新
      - 誤差には10回の平均誤差を利用
- エポック
  - 用意された全トレーニングデータをニューラルネットワークに投入しパラメータを更新・修正するサイクル(＝1エポック)

本日の内容

- ニューラルネットワークの学習
  - 重みの更新方法
  - 学習に用いるデータの整備
- ニューラルネットワークを試す
  - A Neural Network Playground
- 第1回課題の説明