



第2回  
 $y=ax+b$ から始める  
初心者向けML講座

# 自己紹介

名前：Earl Klutz（読み方：アール・クルツ）

- 元はPC・サーバ管理運用、ネットワークインフラ方面生まれ
  - 十数年ほど大手企業3社+a、放浪しつつITと名の付くもの一通りやる。
  - PC5000台管理、サーバ100台管理、ビル1棟のネットワーク配線、プログラミング開発(バック～フロントエンド全部)など色々やる。
  - 何故かAndroidアプリも開発する羽目になった事も…
  - F社方面で仮想化技術（今で言うクラウド）の研究などもしていました。
- 一時期は会社を立てたりもしましたが、現在はフリーで運用・設計・開発など何でもやる便利屋さんです。
- 週5で働くのは微妙 & 請負開発は忙しい時と暇な時の差が激しいので企業向けのPG講師も始める。
  - 気が付いたらJava、PHP、Python、ABAP、SQL、Linux、AWS、Ciscoルータなど色々教える羽目に。
- AIも教えてくれと言われJDLA（日本ディープラーニング協会）のE資格を取得。
- 2020年ごろからとある大学教授に作ってもらった資料を基にE資格講座を全部作って教えることに。

教えてくれ。俺たちははあと何個のプログラム言語を教えればいい？

# 前回のおさらい

# AIの技術分類

- レベル1

シンプルな制御プログラム（センサーなどを用いた単純な動きをするもの）

例：エアコン、洗濯機など

- レベル2

古典的な人工知能（学習はしないがある程度複雑な動きをするもの）

例：ゲームのAI、お掃除ロボットなど

- レベル3

機械学習を用いた人工知能（処理の中で学習機能を持つもの？）

例：通販サイトのおすすめ商品、迷惑メール判定など

- レベル4

ディープラーニングを用いた人工知能

例：Siri、ChatGTP、stable diffusionなど

レベル4の前提  
おろそかにできない！

いっぱいあるがこれが主流

# AIの技術分類

- レベル1

シンプルな制御プログラム（センサーなどを用いた単純な動きをするもの）

例：エアコン、洗濯機など

- レベル2

古典的な人工知能（学習はしないがある程度複雑な動きをするもの）

例：ゲームのAI、お掃除ロボットなど

- レベル3

機械学習を用いた人工知能（処理の中で学習機能を持つもの？）

例：通販サイトのおすすめ商品、迷惑メール判定など

- レベル4

ディープラーニングを用いた人工知能

例：Siri、ChatGTP、stable diffusionなど

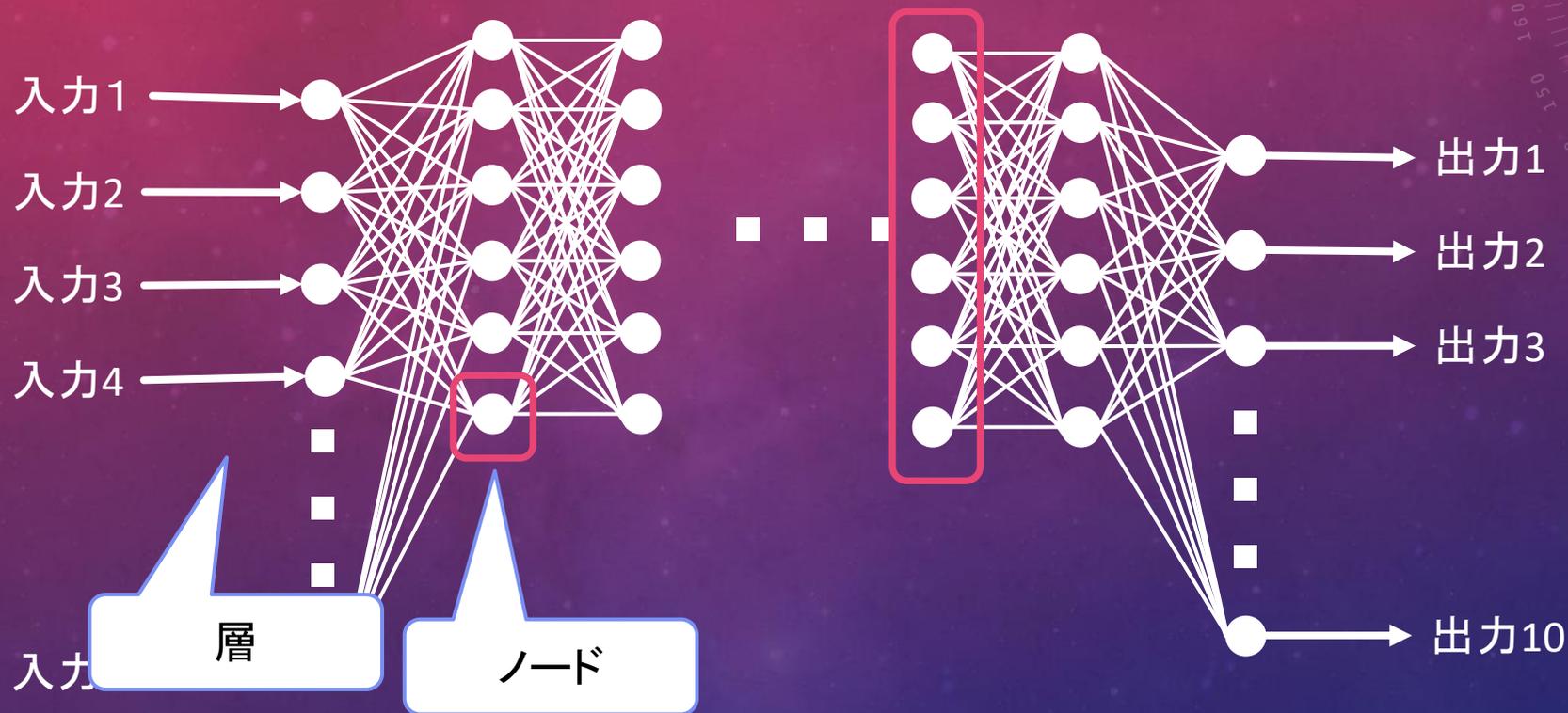
やることが..  
やることが多い...!!

レベル4の前提  
おろそかにできない！

いっぱいあるがこれが主流

# ディープなニューラルネットワーク(NN)が主流

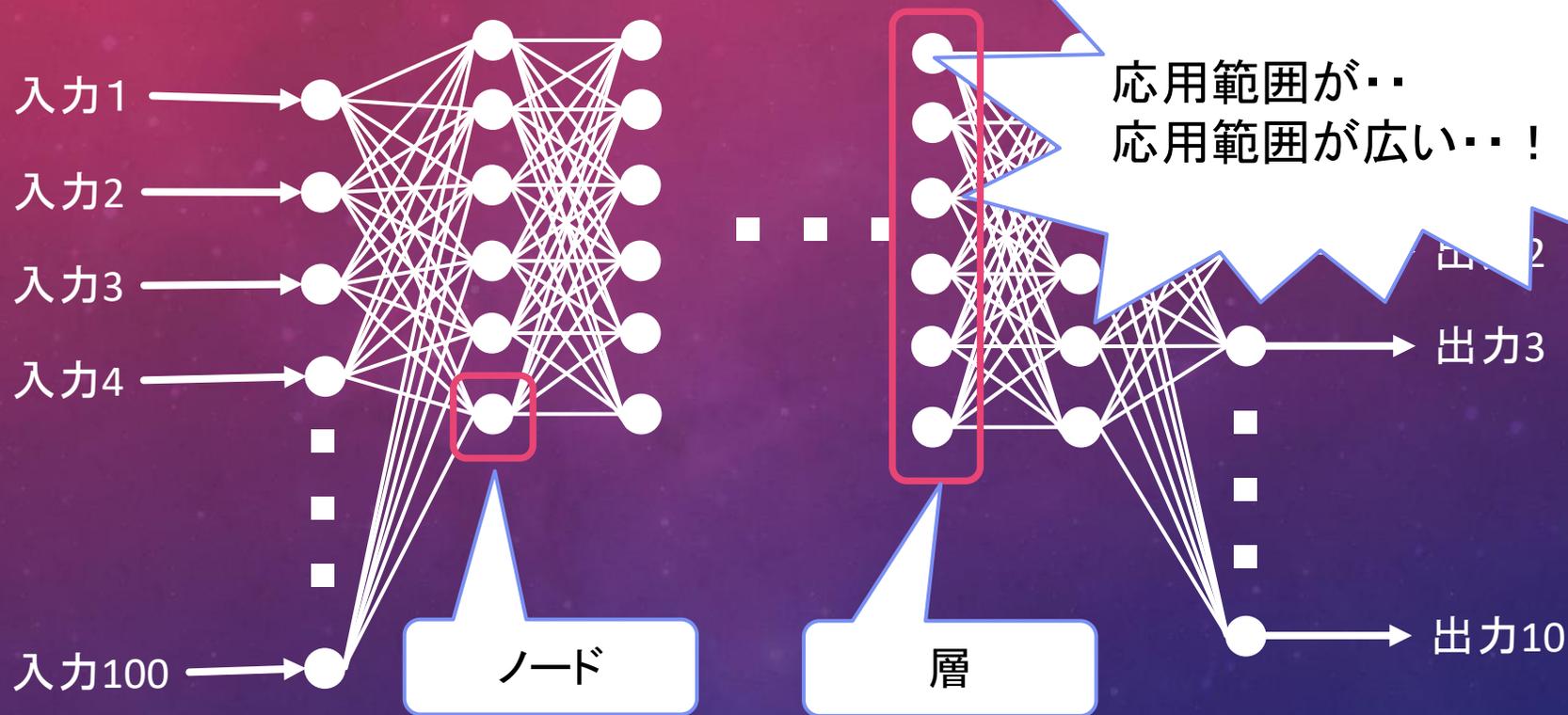
- 「入力」や「出力」「各層のノードの数」や「各層での処理」や「層の数」などを変更していけば様々な応用が可能！



様々な応用的な事を組み合わせたり、複数のモデルを組み合わせるなどしていくと stable diffusion や ChatGTP にもなっていきます。

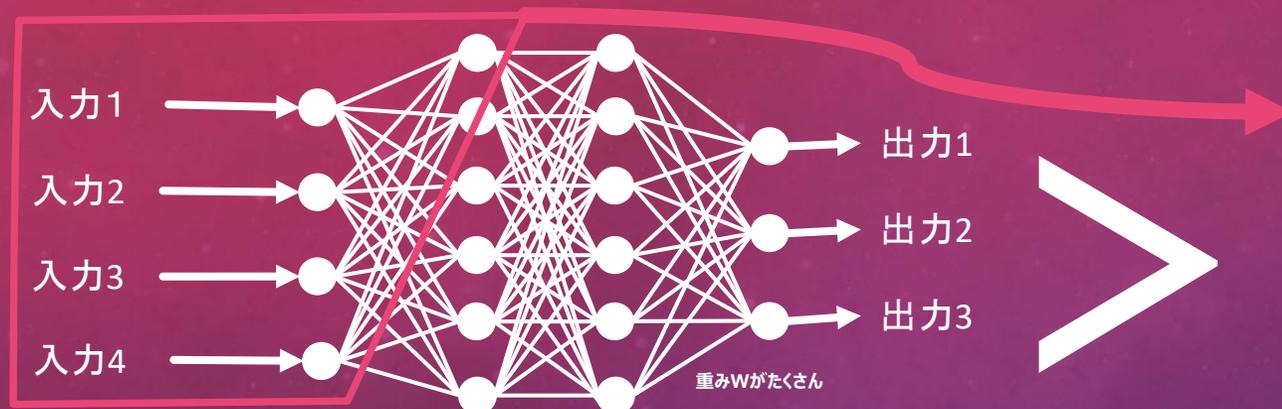
# ディープなニューラルネットワーク(NN)が主流

- 「入力」や「出力」「各層のノードの数」や「各層での処理」や「層の数」などを変更が可能！

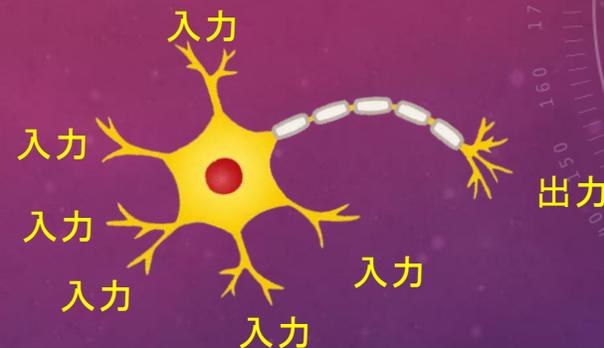


様々な応用的な事を組み合わせたり、複数のモデルを組み合わせるなどしていくと stable diffusion や ChatGTP にもなっていきます。

# NNの一部を抜き出しても学習原理は同じ



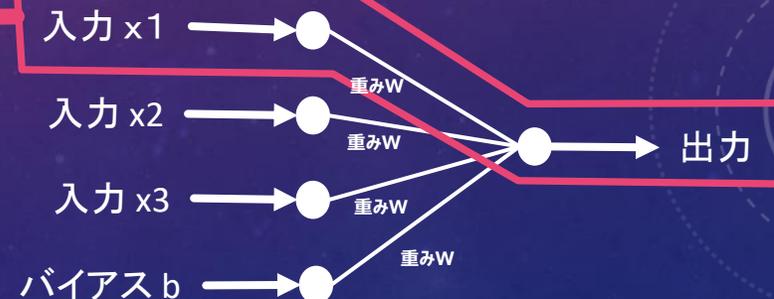
ニューラルネットワーク(NN)



ニューロン

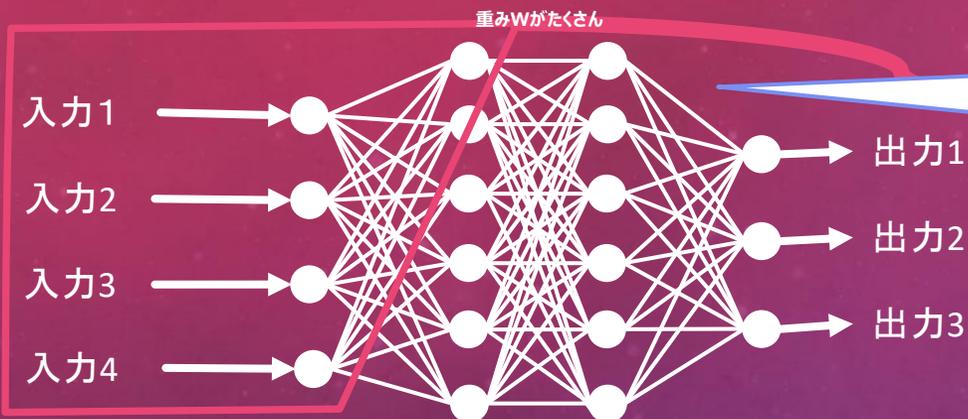


$$y = ax + b$$



パーセプトロン

# NNの一部を抜き出しても学習原理は同じ



ニューラルネットワーク(NN)

線1本1本のところにある【重みW】を【予測値-正解値】から別々に学習する！

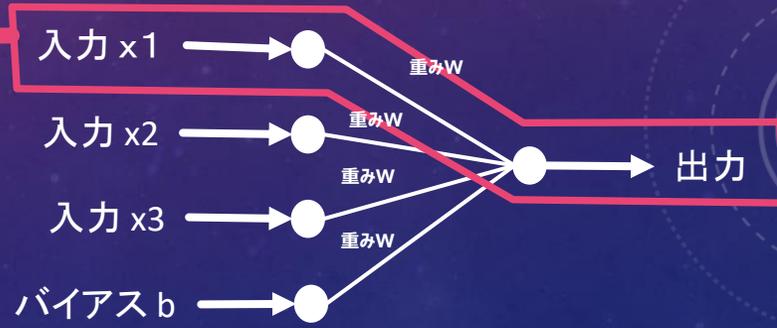


ニューロン

||



$$y = ax + b$$



パーセプトロン

# 機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の計算
  - ②予測値と正解の間の誤差についての学習パラメータに関する各微分値を計算
  - ③微分値から勾配法を用いてパラメータの更新
- 
- どのようなモデルで有ろうとも、これを繰り返すのが機械学習の学習手順
- 
- つまり

$y=ax+b$ の機械学習がニューラルネットワークの基本

# 機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の算出
- ②予測値と正解の間の誤差についての学習パラメータに関する計算
- ③微分値から勾配法を用いてパラメータの更新

力こそパワー！  
計算量こそ全てを  
解決する！

• どのようなモデルで有ろうとも、これを繰り返すのが機械学習の学習手順

• つまり

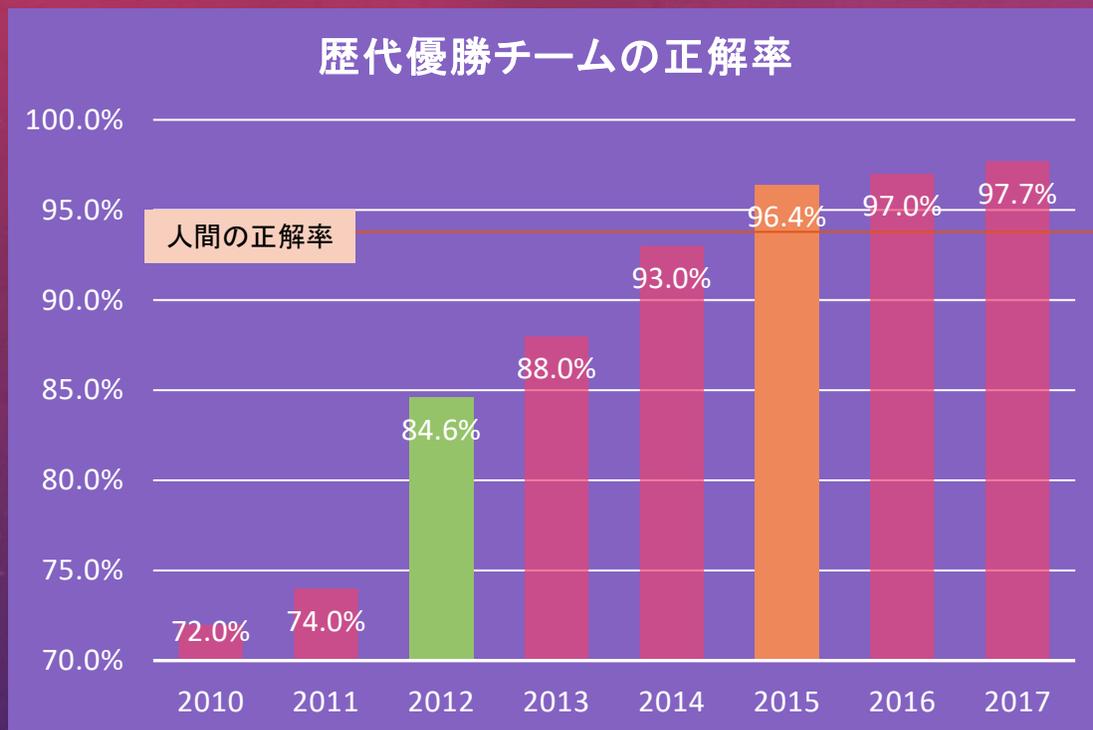
$y=ax+b$ の機械学習がニューラルネットワークの基本

# 何故ニューラルネットワーク(NN)を使うのか？

ニューラルネットワーク(NN)とパーセプトロンの違い

# 何故NNを使うのか？

- LSVRC(画像コンペ)でディープラーニングのAIが優勝したから！



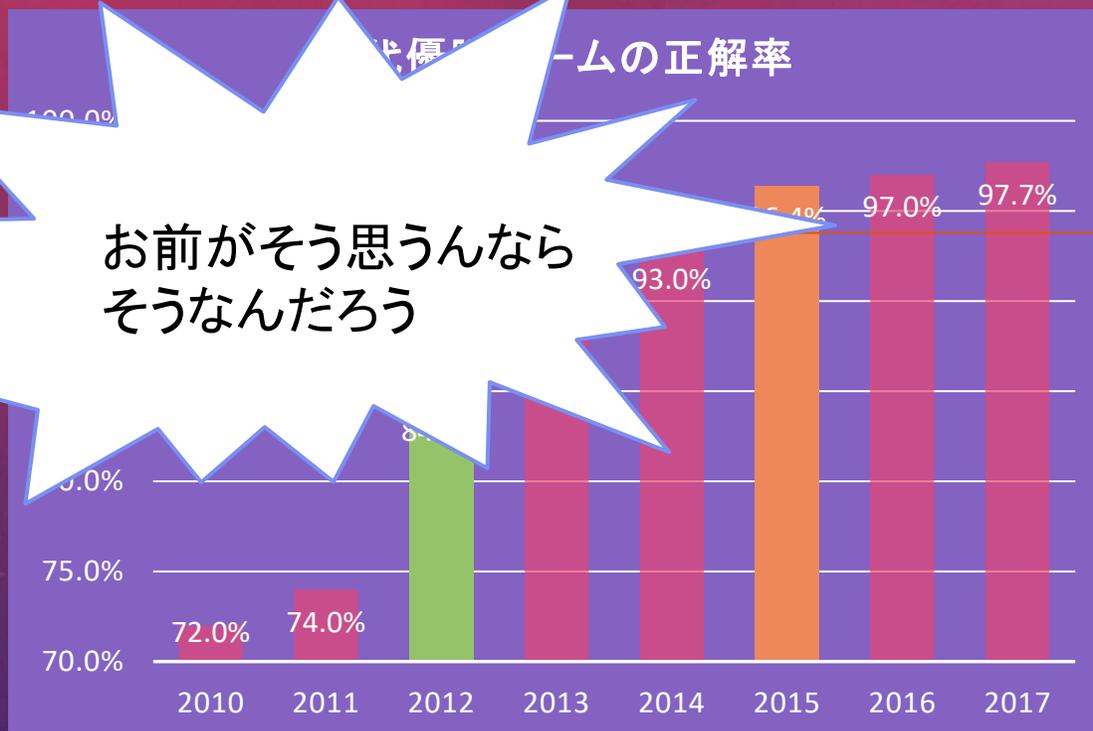
1000万枚の画像で学習後、15万枚の画像でテスト  
2012年に「SuperVision」が84.685%と  
2位と10%以上の大差を付け勝利し、  
ディープラーニングの潜在能力が注目され、  
研究者だけでなく一般にも広がった。

引用元：<https://www.image-net.org/challenges/LSVRC/2017/index.php>

# 何故NNを使うのか？

- LSVRC(画像コンペ)でディープラーニングのAIが優勝したから！

お前がそう思うんなら  
そうなんだろう



引用元：<https://www.image-net.org/challenges/LSVRC/2017/index.php>

1000万枚の画像で学習後、15万枚の画像でテスト  
2012年に「SuperVision」が84.685%と  
2位と10%以上の大差を付け勝利し、  
ディープラーニングの潜在能力が注目され、  
研究者だけでなく一般にも広がった

お前ん中ではな

# 何故NNを使うのか？ 実際は？

Q.何でパーセプトロンは複雑な判断ができないんだ！？

A.パーセプトロンだと直線的な判断しか出来ない！

NNは複雑な判断も出来る！

# 何故NNを使うのか？ 実際は？

Q.何でパーセプトロンは複雑な判断ができないんだ！？

A.パーセプトロンだと直線的な判断しか出来ない！

NNは複雑な判断も出来る！

Q.何故、NNは複雑な判断が出来るんだ！？

A.NNの方が誤差が少なくなるから！

# 何故NNを使うのか？ 実際は？

Q.何でパーセプトロンは複雑な判断ができないんだ！？

A.パーセプトロンだと直線的な判断しか出来ない！

NNは複雑な判断も出来る！

Q.何故、NNは複雑な判断が出来るんだ！？

A.NNの方が誤差が少なくなるから！

Q.だから何でだって言ってるんだろ！

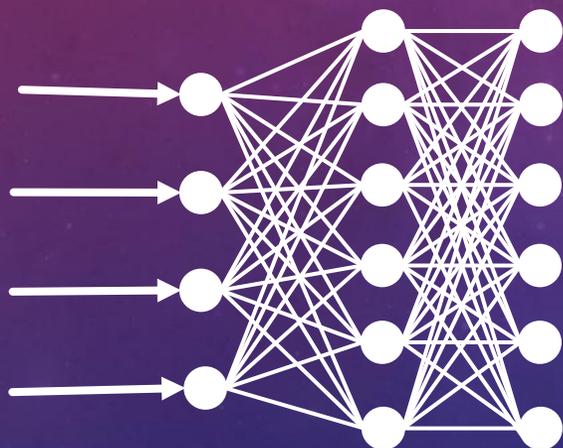
A.何故かそうなるんだよ！知らねーよ！



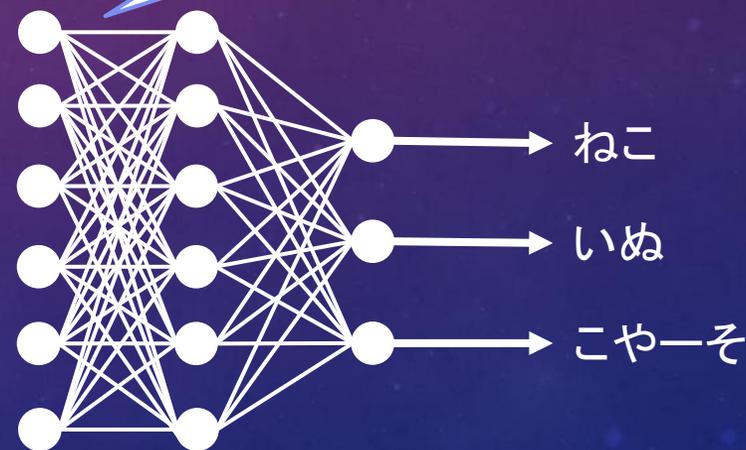
# ニューラルネットワークは事実上のブラックボックス

- ディープラーニングは、機械が自ら膨大なデータを学習し自律的に答えを導き出します。
- このため、その「**思考のプロセスが人間には分からない（ブラックボックス）**」という特性があります。
- もちろんそれを解き明かすXAIという研究もされていますが…

数百次元の重みのパラメータから  
AIの思考を解き明かすのは  
事実上不可能…っ！



■ ■ ■



# ニューラルネットワークは事実上のブラックボックス

- ディープラーニングは、機械が自ら膨大なデータを学習し自律的に答えを導き出します。
- このため、その「**思考のプロセスが人間には分からない（ブラックボックス）**」



う研究もされていますが…

数百次元の重みのパラメータから  
AIの思考を解き明かすのは  
事実上不可能…っ！

まさに宇宙猫！

→ ねこ

→ いぬ

→ こやーそ

ラチがあかないので実験から体感してみる

## • 便利なツール

# 「A Neural Network Playground」

Googleが提供している学習用ツール



日本語版はDeepinsiderが提供  
(初・中級者のためのAI、機械学習の技術フォーラム)  
<https://deepinsider.github.io/playground/>

ラチがあかないので実験から体感してみる

• 便利

Deepinsider版はバグがあるので慣れたら英語版を使おう



TensorFlow Playground  
Googleが提供している学習用ツール



英語版の本家

<https://playground.tensorflow.org/>

日本語版の本家  
(初・中級者のためのAI、機械学習の技術フォーラム)  
<https://deepinsider.github.io/playground/>

# 「A Neural Network Playground」

ニューラルネットワークに、いますぐブラウザ上で触れてみよう！  
大丈夫、壊れたりしないから。好きにいじってみてね。

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層) Tanh  
活性化関数 (出力層) Tanh (分類)

正則化 なし  
正則化率 0

**5** “学習方法” の設計：モデルの生成

損失関数 平均二乗誤差 (主に回帰)  
最適化 SGD

学習率 0.03  
バッチサイズ: 10個  全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

データの何%を訓練  
【Training】用に？  
(残りは精度検証  
【Validation】用)  
: 50%

ノイズ: 0%

データの再生成

入力層

入力データとして使  
いたい特徴を選択し  
てください。

4 neurons

2 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.574  
Validation loss 0.569

出力は、変化する  
重みと合成されま  
す。重み (weights)  
は、線の太さで示  
されています。

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

Train acc (正解率) 0.460  
Validation acc 0.492

データ / ニューロ  
ン (neurons) / 重み  
の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

**8** テスト ⇒ 完成

1つの点を自動生成してテスト



# 「A Neural Network Playground」使い方

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層) Tanh 正則化 なし

活性化関数 (出力層) Tanh (分類) 正則化率 0

**5** "学習方法" の設計 : モデルの生成

損失関数 平均二乗誤差 (主に回帰)

最適化 SGD

学習率 0.03

バッチサイズ: 10個

**6** 学習 : トレーニング

Train loss (平均) 0.669

Validation loss 0.688

Train acc (正解率) 0.556

Validation acc 0.504

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを使いますか?

**3** 前処理

データの何%を訓練【Training】用に?  
(残りは精度検証)

入力層

入力データとして使いたい特徴を選択してください。

$X_1$

$X_2$

4 neurons

2 neurons

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

2個の入力値からオレンジと青の点に2分類するタスク

入力はグラフ化出来るよう  $X_1$ と $X_2$ の2個

The screenshot displays the A Neural Network Playground interface. At the top, there are three main sections: 4. Model Definition, 5. Model Generation, and 6. Training. Section 4 shows the selection of activation functions (Tanh for hidden and output layers) and regularization (none for hidden, 0 for output). Section 5 shows the loss function (Mean Squared Error) and optimizer (SGD) with a learning rate of 0.03 and a batch size of 10. Section 6 shows the training progress with Train loss of 0.669 and Validation loss of 0.688. Below these are performance metrics: Train accuracy of 0.556 and Validation accuracy of 0.504. The central part of the interface shows the neural network architecture with an input layer of 2 neurons ( $X_1$  and  $X_2$ ), a hidden layer of 4 neurons, and an output layer of 2 neurons. A callout box highlights the input layer, stating '入力はグラフ化出来るよう  $X_1$ と $X_2$ の2個' (Input can be graphed with 2 items  $X_1$  and  $X_2$ ). Another callout box points to the training results, stating '2個の入力値からオレンジと青の点に2分類するタスク' (Task of 2-class classification from 2 input values: orange and blue points). A scatter plot on the right shows the data points and the decision boundary. A text box explains that the output is a combination of weights, with line thickness representing the weights.

# 「A Neural Network Playground」使い方

The screenshot shows the A Neural Network Playground interface. At the top, there are three main sections: 4 "手法" の選択: モデルの定義, 5 学習方法の設計: モデルの生成, and 6 学習: トレーニング. The interface includes various controls for model configuration, training, and visualization.

**4 "手法" の選択: モデルの定義**  
活性化関数 (隠れ層): Tanh  
活性化関数 (出力層): Tanh (分類)  
正則化: なし  
正則化率: 0

**5 学習方法の設計: モデルの生成**  
学習率: 0.03  
バッチサイズ: 10個

**6 学習: トレーニング**  
Train loss: 669  
Validation loss: 688  
Train acc (正解率): 0.556  
Validation acc: 0.504

**1 データ準備**  
座標点

**2 問題種別**  
分類  
どのデータセットを  
使いますか?

**3 前処理**  
データの何%を訓練  
【Training】用に?  
(残りは精度検証)

**層の数と各層のノード数を変更可能**

**現状の予測が青とオレンジの背景で可視化されている**

入力層  
入力データとして使  
いたい特徴を選択し  
てください。

2 隠れ層

4 neurons

2 neurons

出力は、変化する  
重みと合成されま  
す。重み (weights)  
は、線の太さで示  
されています。

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

Train loss: 669  
Validation loss: 688

Train acc (正解率) 0.556  
Validation acc: 0.504

# 「A Neural Network Playground」使い方

The screenshot shows the A Neural Network Playground interface with several key components and annotations:

- 4 "手法" (Method):** Includes options for activation functions like Tanh.
- 5 "学習方法" の設計: モデルの生成 (Learning Method Design: Model Generation):** Includes dropdowns for loss function (平均二乗誤差 (主に回帰) - Mean Squared Error (mainly regression)), optimizer (SGD), and learning rate (0.03).
- 6 学習: トレーニング (Learning: Training):** Includes a play button, a refresh button, and a progress indicator for Epochs (000,000).
- 1 データ準備 (Data Preparation):** Includes a dropdown for coordinates (座標点).
- 2 問題種別 (Problem Type):** Includes a dropdown for classification (分類) and a question about which dataset to use.
- 3 前処理 (Preprocessing):** Includes a dropdown for training percentage (データの何%を訓練【Training】用に?) and a note that the remainder is for accuracy verification (残りは精度検証).
- Network Diagram:** Shows input nodes ( $X_1, X_2, X_1^2, X_2^2, X_1X_2, \sin(X_1)$ ), hidden nodes (隠れ層), and output nodes. A callout explains that the output is composed of weights and that the thickness of the lines represents the weights.
- Scatter Plot:** Shows a 2D scatter plot of data points with a decision boundary. Below it, the training accuracy (Train acc (正解率) 0.556) and validation accuracy (Validation acc 0.504) are displayed.

**Annotations:**

- Blue callout: 前回見た誤差の求め方 (How to find the error we saw last time).
- Red callout: SGD: 確率的勾配降下法 前回やった方法 (SGD: Stochastic Gradient Descent, the method we did last time).
- Blue callout: 前回やった学習係数  $\lambda$  の重みの更新式の「 $\lambda$ 」 (The  $\lambda$  in the weight update formula from the last time we did learning). The formula shown is 
$$a \leftarrow a - \lambda \frac{\partial E}{\partial a}$$

# 「A Neural Network Playground」使い方

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層) Tanh

活性化関数 (出力層) Tanh (分類)

正則化 なし

正則化率 0

**5** “学習方法” の設計：モデルの生成

損失関数

学習率 0.03

1回の学習で使うサンプル数

バッチサイズ: 10個

全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

データの何%を訓練  
【Training】用に？  
(残りは精度検証)

入力層

入力データとして使  
いたい特徴を選択し  
てください。

+ - 2 隠れ層

4 neurons

$X_1$

$X_2$

$X_1^2$

$X_2^2$

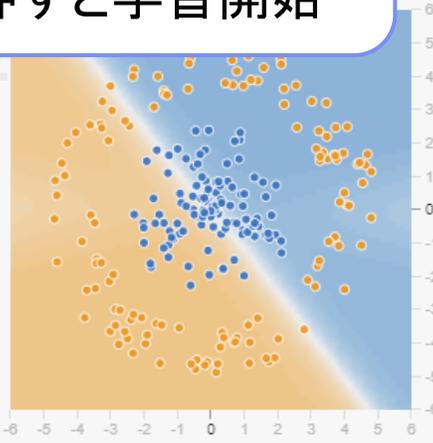
$X_1X_2$

$\sin(X_1)$

出力は、変化する  
重みと合成されま  
す。重み (weights)  
は、線の太さで示  
されています。

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

学習中に学習データを  
全部使い切ると1エポック  
「▶」を押すと学習開始



Train acc (正解率) 0.556

Validation acc 0.504

# 隠れ層0、ノード1個でパーセプトロン：2分類問題

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

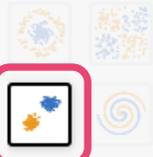
**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？



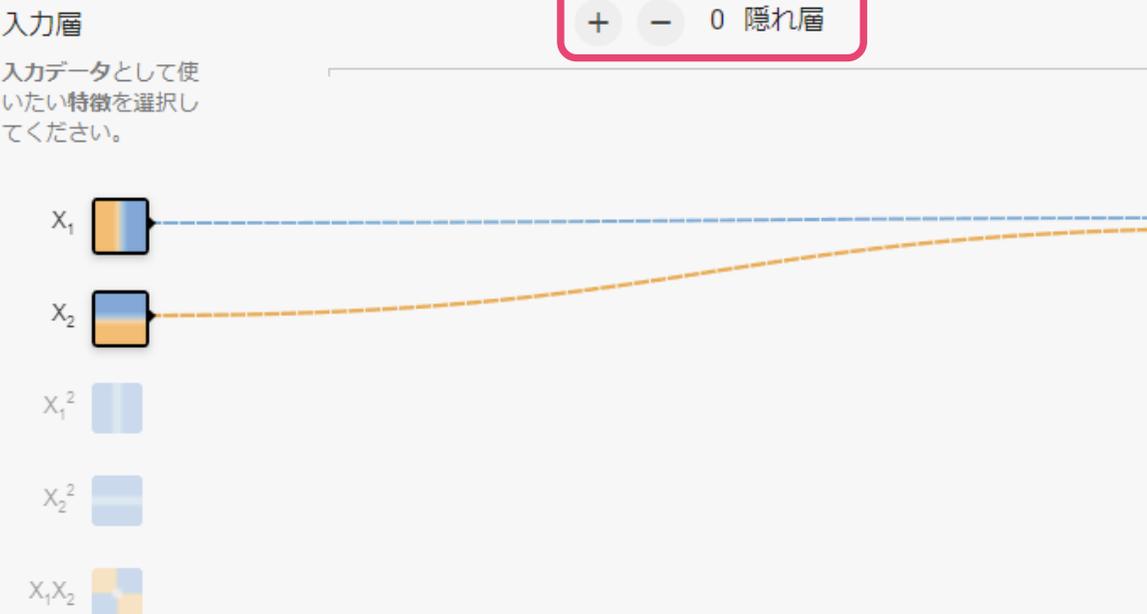
**3** 前処理

**4**

入力層

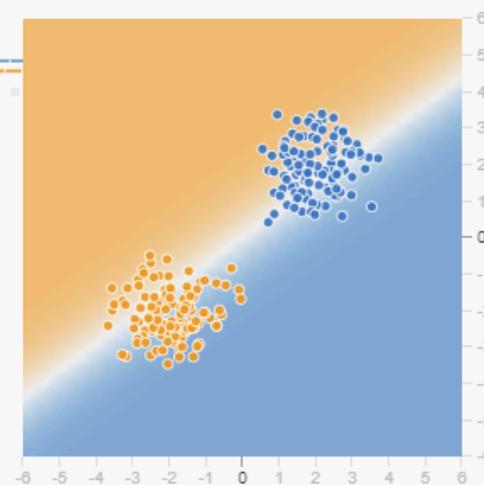
入力データとして使  
いたい特徴を選択し  
てください。

+ - 0 隠れ層



出力層 ⇒ **7** 評価

Train loss (損失) 0.963  
Validation loss 0.904



# 隠れ層0、ノード1個でパーセプトロン：2分類問題

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	Tanh	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)
最適化	SGD

学習率: 0.01  
バッチサイズ: 10個  全部

エポック (Epoch) 000,010

学習データを10周した

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理

入力層  
入力データとして使いたい特徴を選択してください。

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1 X_2$

誤差が小さくなっている  
グラフでも見れる

出力層 ⇒ **7** 評価  
Train loss (損失) 0.006  
Validation loss 0.007

結果は正しく青とオレンジの背景で  
2分割出来ていることが分かる

The image shows a software interface for training a neural network. At the top, there are two main configuration sections: '4 "手法" の選択：モデルの定義' and '5 "学習方法" の設計：モデルの生成'. Section 4 includes dropdowns for activation functions (Tanh for hidden and output layers) and regularization (none and 0). Section 5 includes loss function (mean squared error), optimizer (SGD), learning rate (0.01), and batch size (10). A progress indicator shows 'エポック (Epoch) 000,010' and a callout says '学習データを10周した'. Below the configuration is a diagram of the neural network with an input layer and an output layer. The input layer has nodes for  $X_1$ ,  $X_2$ ,  $X_1^2$ ,  $X_2^2$ , and  $X_1 X_2$ . A callout points to the diagram saying '誤差が小さくなっている グラフでも見れる'. On the right, there is a scatter plot showing two clusters of data points (orange and blue) separated by a vertical decision boundary. A callout points to the plot saying '結果は正しく青とオレンジの背景で 2分割出来ていることが分かる'. Above the plot, a callout says '出力層 ⇒ 7 評価' and shows 'Train loss (損失) 0.006' and 'Validation loss 0.007'.

# 隠れ層0、ノード1個でパーセプトロン：2分類問題

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,010

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理

入力層  
入力データとして使いたい特徴を選択してください。

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$

線にカーソルを合わせると  
重みの値が見れる

クリックすると、値を手動で編集  
できます。  
重みは 1.8。

出力層 ⇒ **7** 評価  
Train loss (損失) 0.006  
Validation loss 0.007

The screenshot shows a web-based neural network training interface. At the top, there are three main sections: 4. Model Definition, 5. Model Generation, and 6. Training. Section 4 shows activation functions for hidden and output layers, and regularization settings. Section 5 shows loss function, learning rate, optimizer, and batch size. Section 6 shows the training progress, including the number of epochs (000,010) and loss values (Train loss: 0.006, Validation loss: 0.007). Below these are three steps: 1. Data Preparation (座標点), 2. Problem Type (分類), and 3. Preprocessing (前処理). The main area displays the input layer with feature selection options ( $X_1$ ,  $X_2$ ,  $X_1^2$ ,  $X_2^2$ ,  $X_1X_2$ ). A blue dashed line connects the  $X_1$  input to the output layer. A callout box points to this line, stating that hovering over it shows the weight value (1.8). Another callout box points to the output layer, stating that clicking it allows manual editing of the weight. On the right, a scatter plot shows two classes of data points (orange and blue) separated by a vertical decision boundary at  $X_1 = 0$ . The plot axes range from -6 to 6.

# 隠れ層0、ノード1個でパーセプトロン：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？



**3** 前処理

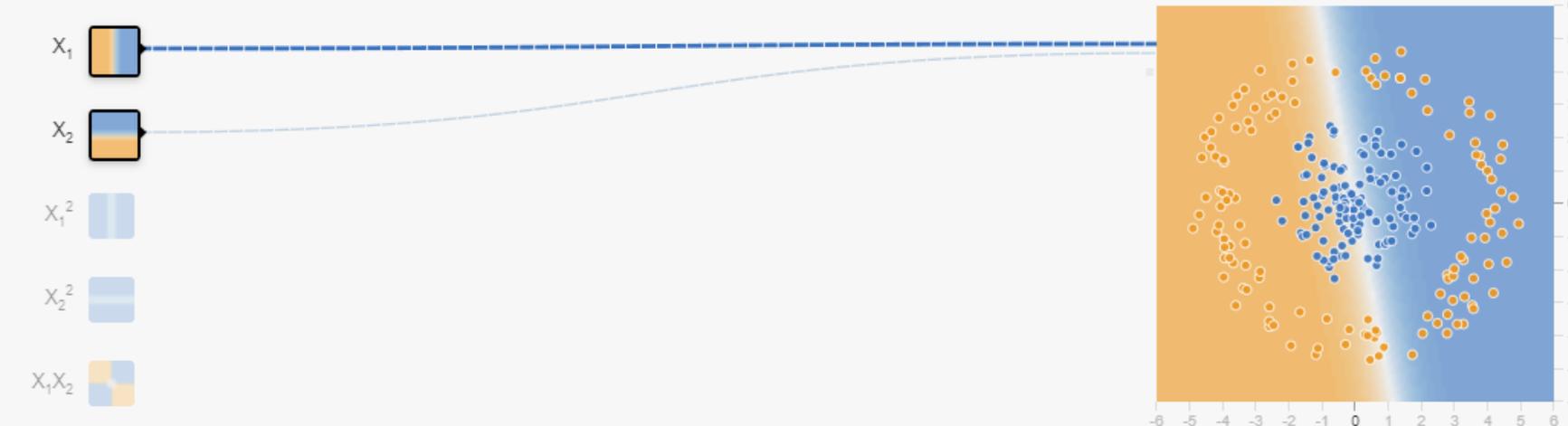
入力層 **4**

入力データとして使  
いたい特徴を選択し  
てください。

+ - 0 隠れ層

出力層 ⇒ **7** 評価

Train loss (損失) 0.819  
Validation loss 0.816



# 隠れ層0、ノード1個でパーセプトロン：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03

**6** 学習：トレーニング

エポック (Epoch) 000,019

Train loss (損失) 0.495  
Validation loss 0.508

出力層 ⇒ **7** 評価

色が薄くなり  
判断は不能になっている

誤差は小さくなっているが

The image shows a software interface for training a neural network. At the top, there are three main sections: 4. Model Definition, 5. Model Generation, and 6. Training. Section 4 shows the activation function for the hidden layer is Tanh and for the output layer is Tanh (Classification). The regularization rate is 0. Section 5 shows the loss function is Mean Squared Error (Mainly Regression) and the learning rate is 0.03. Section 6 shows the training progress with 19 epochs completed. A scatter plot on the right shows the output layer evaluation, with a decision boundary separating blue and orange data points. A callout box points to the plot, stating '色が薄くなり判断は不能になっている' (Colors are fading and judgment is becoming impossible). Another callout box points to the training progress, stating '誤差は小さくなっているが' (Error is getting smaller, but...). The interface also includes sections for data preparation, problem type (Classification), and preprocessing.

# 隠れ層0、ノード1個でパーセプトロン：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03

**6** 学習：トレーニング

エポック (Epoch) 000,019

Train loss (損失) 0.495  
Validation loss 0.508

出力層 ⇒ **7** 評価

色が薄くなり  
判断は不能になっている

誤差は小さくなっているが

The image shows a software interface for training a neural network. It is divided into several sections: 1. Data preparation (座標点), 2. Problem type (分類), 3. Preprocessing (前処理), 4. Model definition (手法), 5. Model generation (学習方法), and 6. Training (学習). The training progress shows 000,019 epochs. A scatter plot shows data points (blue and orange) with a decision boundary. A callout box points to the plot, stating '色が薄くなり判断は不能になっている' (Colors are fading and judgment is becoming impossible). Another callout box points to the training progress, stating '誤差は小さくなっているが' (Error is getting smaller but...). The loss values are Train loss 0.495 and Validation loss 0.508.

# 隠れ層1個、ノード1個の場合：2分類問題その2

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** “学習方法” の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

**4**

+ - 1 隠れ層

+ -  
1 neuron

これは、このニュー  
ロン (neuron)  
の出力  
マウス  
と、右  
側に

出力層 ⇒ **7** 評価

Train loss (損失) 0.811  
Validation loss 0.797

パーセプトロンを2個  
連結させたような感じ

33

# 隠れ層1個、ノード1個の場合：2分類問題その2

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** “学習方法” の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,173

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1X_2$

+ - 1 隠れ層

+ -

1 neuron

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.405

Validation loss 0.431

# 隠れ層1個、ノード1個の場合：2分類問題その2

**4** “手法” の選択：モデルの定義  
活性化関数 (隠れ層) Tanh  
活性化関数 (出力層) Tanh (分類)  
正則化 なし  
正則化率 0

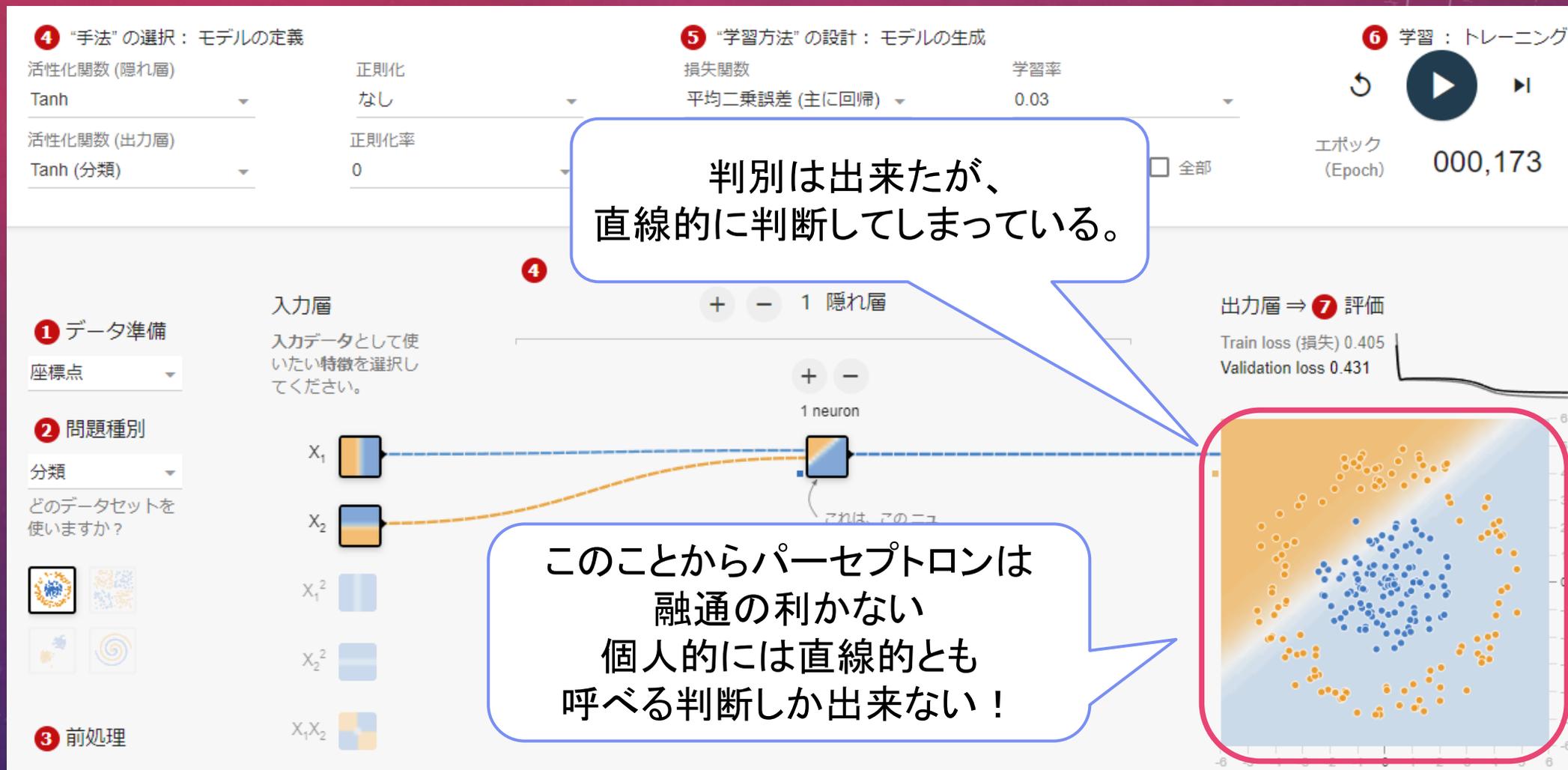
**5** “学習方法” の設計：モデルの生成  
損失関数 平均二乗誤差 (主に回帰)  
学習率 0.03

**6** 学習：トレーニング  
エポック (Epoch) 000,173

Train loss (損失) 0.405  
Validation loss 0.431

判別は出来たが、直線的に判断してしまっている。

このことからパーセプトロンは融通の利かない個人的には直線的とも呼べる判断しか出来ない！



# 隠れ層1個、ノード1個の場合：2分類問題その2

入力値が複数個でも…っ！

判別は出来たが、  
目的に判断してしまっ

これが…っ！  
パーセプトロンの限界っ！

複雑な判断は  
不可能…っ！！

このことからパーセプトロンは  
融通の利かない  
個人的には直線的とも  
呼べる判断しか出来ない！

The screenshot shows a web-based neural network training tool. It is divided into several sections:

- 5 "学習方法" の設計：モデルの生成**: Shows the loss function set to "平均二乗誤差 (主に回帰)" and the learning rate set to "0.03".
- 6 学習：トレーニング**: Shows a play button and a progress indicator at "0,173".
- 1 データ準備**: "座標点" is selected.
- 2 問題種別**: "分類" is selected, and "どのデータセットを 使いますか?" is asked.
- 3 入力層**: "入力データとして使 いたい特徴を選択し てください。" Two input nodes,  $X_1$  and  $X_2$ , are shown.
- 4 隠れ層**: "1 neuron" is shown.

Annotations include:

- A blue starburst pointing to the input layer with the text "入力値が複数個でも…っ！".
- A blue starburst pointing to the output layer with the text "判別は出来たが、目的に判断してしまっ".
- A blue starburst pointing to the training progress with the text "これが…っ！ パーセプトロンの限界っ！".
- A blue starburst pointing to the input layer with the text "複雑な判断は 不可能…っ！！".
- A blue starburst pointing to the single neuron with the text "このことからパーセプトロンは 融通の利かない 個人的には直線的とも 呼べる判断しか出来ない！".

On the right side, a scatter plot shows two classes of data points (blue and orange) separated by a linear decision boundary. A red box highlights this plot.

# 隠れ層1、ノード2個では？ : 2分類問題その2

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	Tanh	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計 : モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか?

**3** 前処理

**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層

2 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.757  
Validation loss 0.744

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

# 隠れ層1、ノード2個では？ : 2分類問題その2

**4** "手法" の選択 : モデルの定義  
活性化関数 (隠れ層) Tanh  
活性化関数 (出力層) Tanh (分類)  
正則化 なし  
正則化率 0

**5** "学習方法" の設計 : モデルの生成  
損失関数 平均二乗誤差 (主に回帰)  
最適化 SGD  
学習率 0.03  
バッチサイズ: 10個  全部

**6** 学習 : トレーニング  
エポック (Epoch) 000,000

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを  
使いますか?

**3** 前処理

入力層  
入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層  
+ -  
2 neurons

出力層 ⇒ **7** 評価  
Train loss (損失) 0.757  
Validation loss 0.744

これはこのニュー  
ロン (neuron)  
の出力内部  
マウスを  
と、右の出  
所に拡大し

さらに1個ノードを増やしてみる

# 隠れ層1、ノード2個では？ : 2分類問題その2

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,078

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか?

**3** 前処理

**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層

2 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.176  
Validation loss 0.266

39

# 隠れ層1、ノード2個では？ : 2分類問題その2

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,078

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層

2 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.176  
Validation loss 0.266

ちよつと曲線的な判断が出来ているが  
まだ完全には分類出来ていない

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1X_2$

4

+ - 1 隠れ層

+ -

3 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.601

Validation loss 0.642

これは、このニュー  
ーロン (neuron)  
の出力内容です。  
マウスを載せると、右の出力の場  
所に拡大します。

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	Tanh	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理

入力層  
入力データとして使いたい特徴を選択してください。

1 隠れ層  
3 neurons

出力層 ⇒ **7** 評価  
Train loss (損失) 0.601  
Validation loss 0.642

これは、ニューロン (ノード) の出力内容を確認するためにマウスを置きます

今度はノードを3個だって？  
このいやしんぼめ！

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** “学習方法” の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,007

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1X_2$

+ - 1 隠れ層

+ -  
3 neurons

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.303

Validation loss 0.325

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,026

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1 X_2$

+ - 1 隠れ層

3 neurons

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.190

Validation loss 0.263

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,045

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1X_2$

1 隠れ層

3 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.163

Validation loss 0.235

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、右の出力の場  
所に拡大します。

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** “学習方法” の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,068

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1X_2$

1 隠れ層

3 neurons

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せる  
と、右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.088

Validation loss 0.107

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	Tanh	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,095

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

$X_1$

$X_2$

$X_1^2$

$X_2^2$

$X_1 X_2$

4

+ - 1 隠れ層

+ -

3 neurons

これは、このニュー  
ーロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.041

Validation loss 0.046

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	Tanh	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	
最適化	SGD		

**6** 学習：トレーニング

エポック (Epoch) 000,095

100エポック前後  
でうまく収束

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層

入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層

3 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.041  
Validation loss 0.046

誤差が小さくなり  
分類が出来た！

The screenshot displays a neural network training interface. At the top, there are three main sections: 4. Model Definition, 5. Model Generation, and 6. Training. Section 4 shows the selection of Tanh for both hidden and output layers, and no regularization. Section 5 shows Mean Squared Error as the loss function and SGD as the optimizer. Section 6 shows the training progress at 95 epochs with a loss of 0.041. Below this, there are three numbered steps: 1. Data Preparation (Coordinate points), 2. Problem Type (Classification), and 3. Preprocessing. The main part of the interface shows a neural network diagram with an input layer, one hidden layer with 3 neurons, and an output layer. To the right, there is a scatter plot of data points, with a blue cluster in the center and orange points around it. A red box highlights the scatter plot and the training progress section. A blue speech bubble points to the training progress section, and another blue speech bubble points to the scatter plot.

# 隠れ層1、ノード3個にしてみる：2分類問題その2

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層) Tanh

活性化関数 (出力層) Tanh (分類)

正則化 なし

正則化率 0

**5** "学習方法" の設計：モデルの生成

損失関数 平均二乗誤差 (主に回帰)

最適化 SGD

学習率 0.03

バッチサイズ: 10個

エポック 000,095

**6** 学習：トレーニング

Train loss (損失) 0.040

Validation loss 0.040

**1** データ準備

座標点

**2** 問題種別

分類

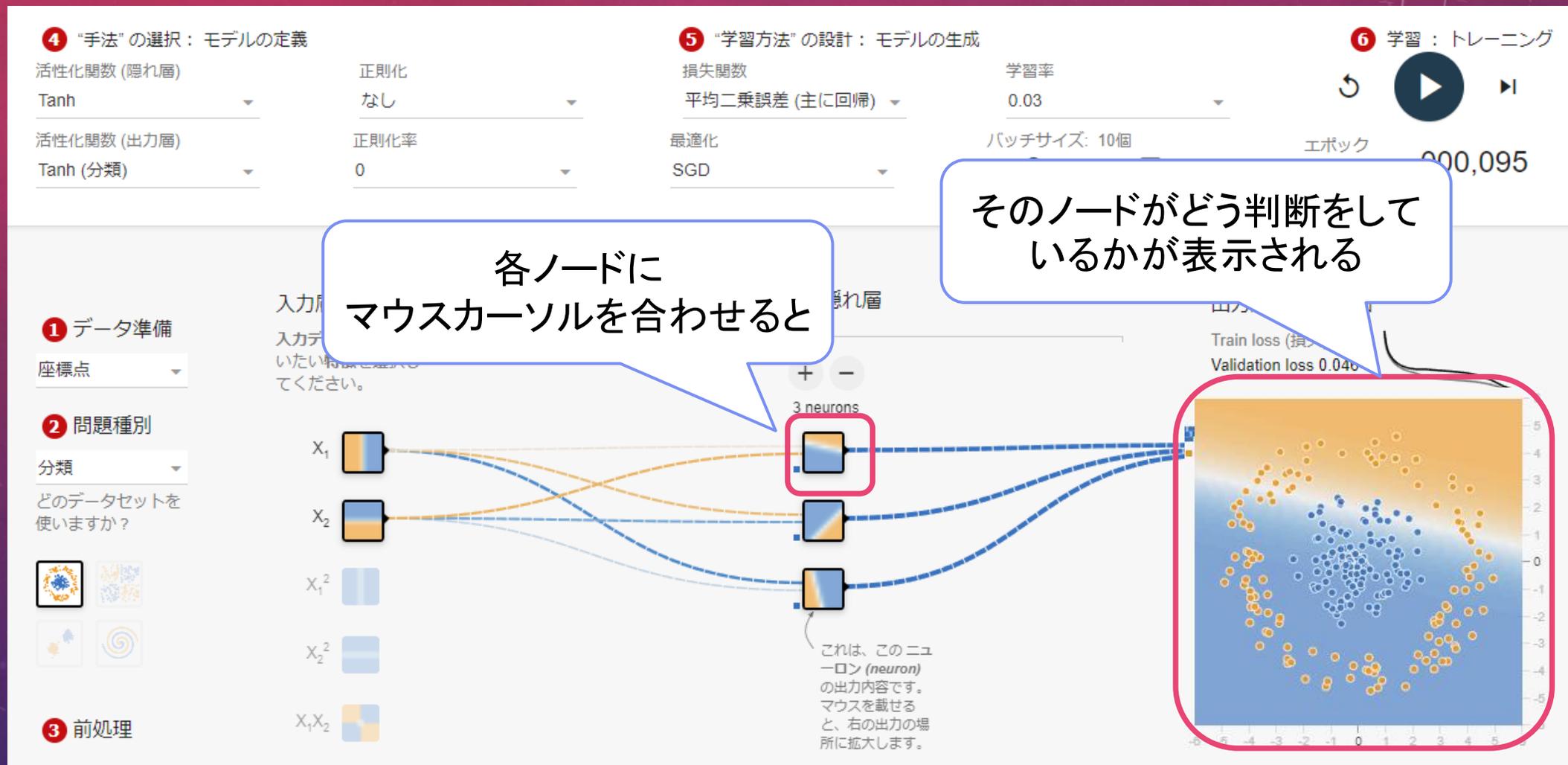
どのデータセットを  
使いますか？

**3** 前処理

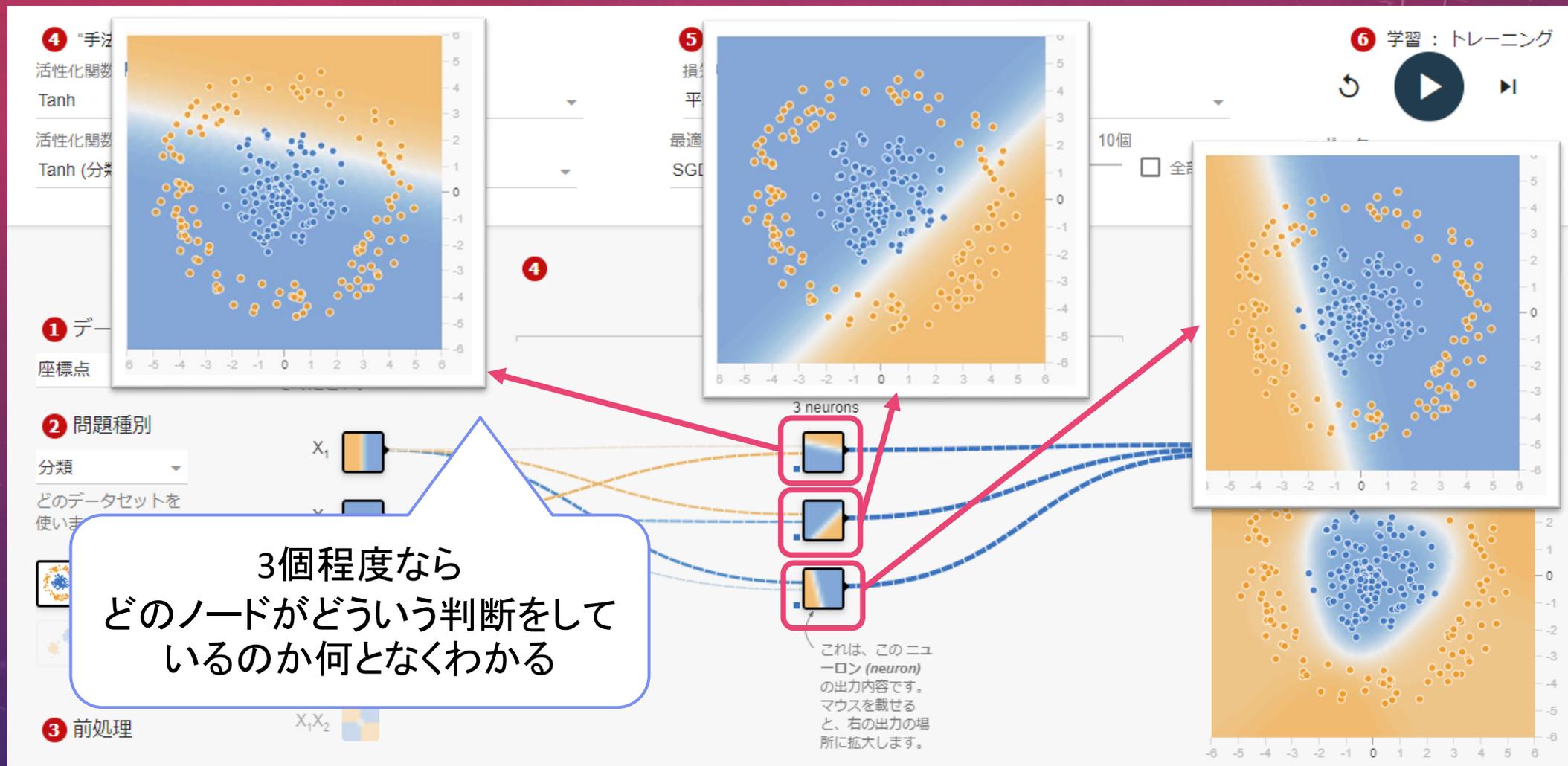
各ノードに  
マウスカーソルを合わせると

そのノードがどう判断をして  
いるかが表示される

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。



# 隠れ層1、ノード3個にしてみる：2分類問題その2



# 渦を巻いたようなデータだと？ : 2分類問題その3

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを使いますか?

**3** 前処理

**4**

入力層

入力データとして使いたい特徴を選択してください。

1 隠れ層

3 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.665  
Validation loss 0.690

# 渦を巻いたようなデータだと？ : 2分類問題その3

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,000

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか?

**3** 前処理

入力層  
入力データとして使いたい特徴を選択してください。

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$

出力層 ⇒ **7** 評価  
Train loss (損失) 0.665  
Validation loss 0.690

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

グラフにプロットすればわかるが、数値データだけだとかなり複雑なデータの例

# 渦を巻いたようなデータだと？：2分類問題その3

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計：モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを  
使いますか？

**3** 前処理

入力層  
入力データとして使  
いたい特徴を選択し  
てください。

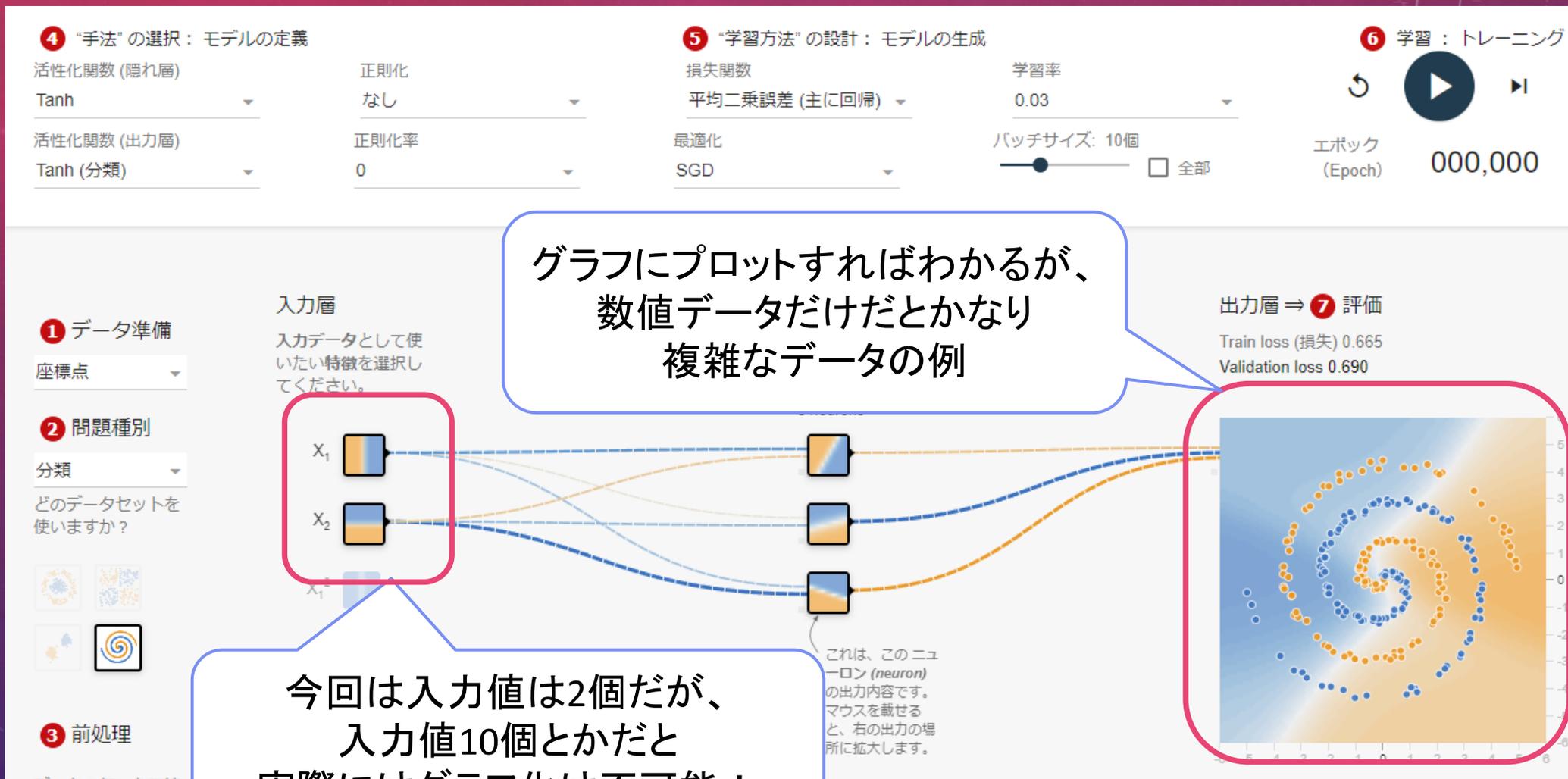
$x_1$   
 $x_2$

出力層 ⇒ **7** 評価  
Train loss (損失) 0.665  
Validation loss 0.690

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

グラフにプロットすればわかるが、  
数値データだけだとかなり  
複雑なデータの例

今回は入力値は2個だが、  
入力値10個とかだと  
実際にはグラフ化は不可能！



# 渦を巻いたようなデータだと？ : 2分類問題その3

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,148

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？



**3** 前処理

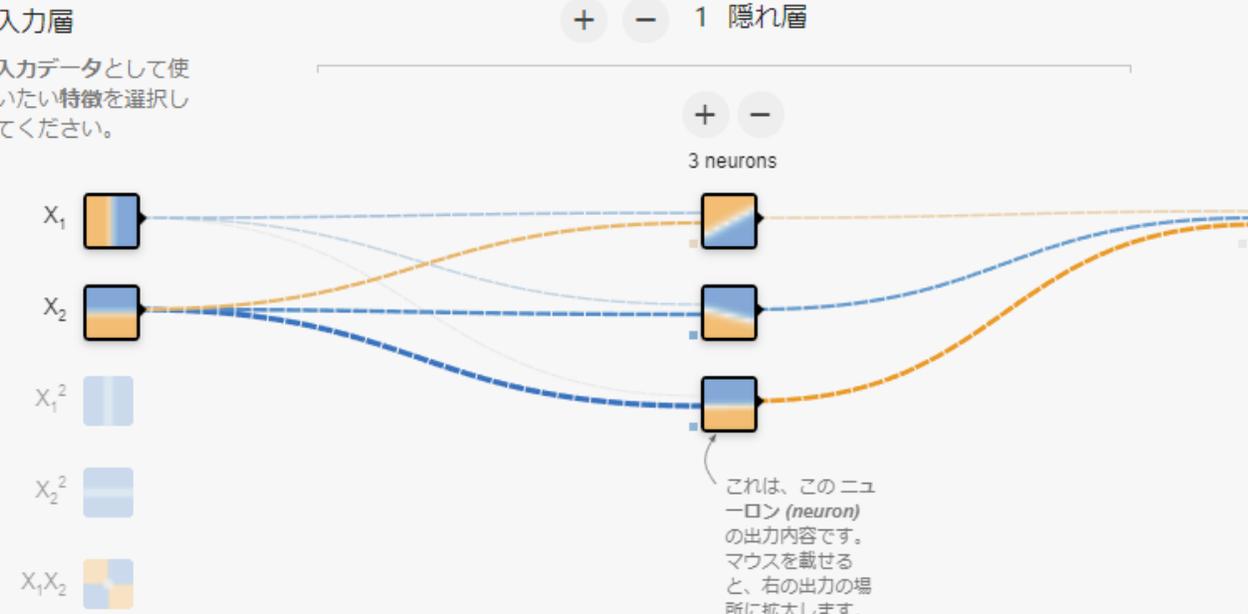
**4**

入力層

入力データとして使  
いたい特徴を選択し  
てください。

1 隠れ層

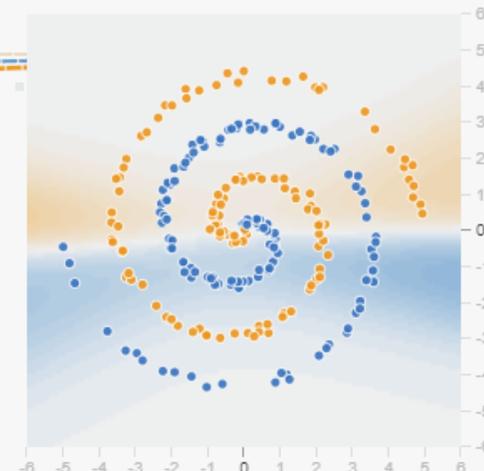
3 neurons



これは、このニュー  
ーロン (neuron)  
の出力内容です。  
マウスを載せる  
と、右の出力の場  
所に拡大します。

出力層 ⇒ **7** 評価

Train loss (損失) 0.478  
Validation loss 0.483



# 渦を巻いたようなデータだと？ : 2分類問題その3

**4** "手法" の選択 : モデルの定義

活性化関数 (隠れ層)	正則化
Tanh	なし
活性化関数 (出力層)	正則化率
Tanh (分類)	0

**5** "学習方法" の設計 : モデルの生成

損失関数	学習率
平均二乗誤差 (主に回帰)	0.03
最適化	バッチサイズ: 10個
SGD	<input type="checkbox"/> 全部

**6** 学習 : トレーニング

エポック (Epoch) 000,148

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

入力層  
入力データとして使  
いたい特徴を選択し  
てください。

3 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$

誤差は横ばいとなり  
学習は進んでいない

出力層 ⇒ **7** 評価

Train loss (損失) 0.478  
Validation loss 0.483

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せると、  
右の出力の場  
所に拡大します。

# 渦を巻いたようなデータだと？ : 2分類問題その3

**4 "手法" の選択 : モデルの定義**  
活性化関数 (隠れ層) Tanh  
正則化 なし  
活性化関数 (出力層) Tanh (分類)

**5 "学習方法" の設計 : モデルの生成**  
損失関数 平均二乗誤差 (主に回帰)  
学習率 0.03  
最適化 SGD  
バッチサイズ: 10個  
全部

**6 学習 : トレーニング**  
エポック (Epoch) 000,148

**1 データ準備**  
座標点

**2 問題種別**  
分類  
どのデータセットを  
使いますか?

**3 前処理**

入力データとして使  
いたい特徴を選択し  
てください。

3 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$

この二  
ーロン (neuron)  
の出力内容です。  
マウスを載せる  
と、右の出力の場  
所に拡大します。

出力層 ⇒ **7 評価**  
Train loss (損失) 0.478  
Validation loss 0.483

**当然っ!**

**誤差は横ばいとなり  
学習は進んでいない**

**駄目っ  
.....!**

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** “学習方法” の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを使いますか?

**3** 前処理

データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用) : 50%

ノイズ: 15%

入力層

入力データとして使いたい特徴を選択してください。

6 neurons    6 neurons    8 neurons

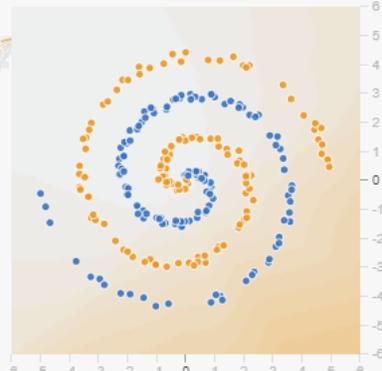
$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

出力層 ⇒ **7** 評価

Train loss (損失) 0.522  
Validation loss 0.531



Train acc (正解率) 0.480  
Validation acc 0.520

データ / ニューロン (neurons) / 重みの値を色で表現。

訓練データ表示     精度検証データ表示  
 テストデータ表示     出力の離散化

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

一気に増やしてみる

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** “学習方法” の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,000

出力層 ⇒ **7** 評価

Train loss (損失) 0.522  
Validation loss 0.531

Train acc (正解率) 0.480  
Validation acc 0.520

データ / ニューロン (neurons) / 重み の値を色で表現。 ■ -1 ■ 0 ■ 1

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

入力層

3 隠れ層

6 neurons 6 neurons 8 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1 X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4 "手法" の選択：モデルの定義**

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5 "学習方法" の設計：モデルの生成**

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6 学習：トレーニング**

エポック (Epoch) 000,072

**1 データ準備**

座標点

**2 問題種別**

分類

どのデータセットを使いますか？



**3 前処理**

データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)

50%

ノイズ: 15%

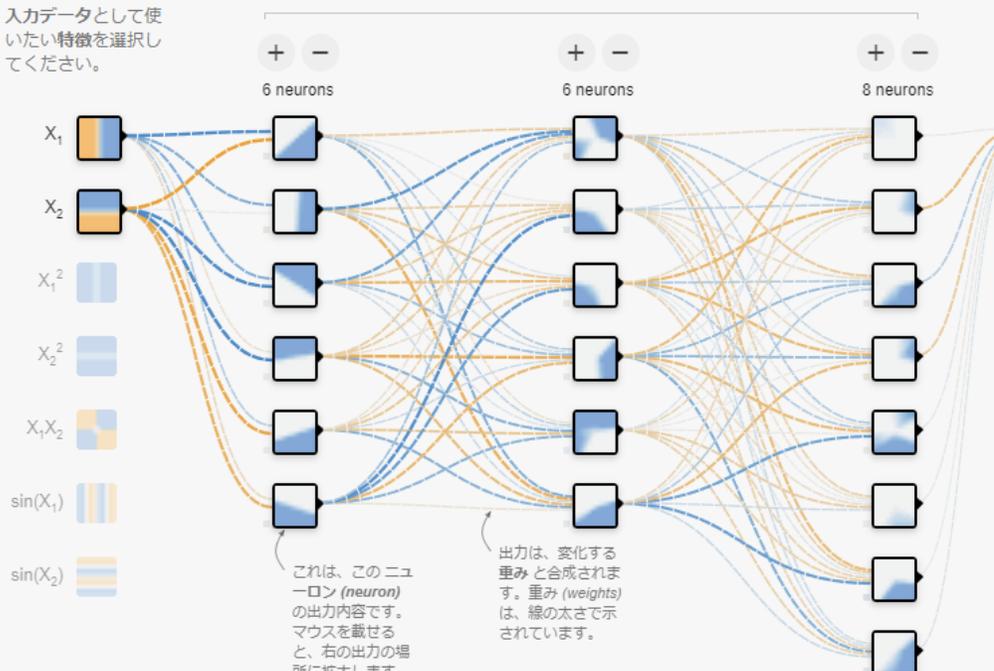
**4**

入力層

入力データとして使いたい特徴を選択してください。

3 隠れ層

6 neurons    6 neurons    8 neurons

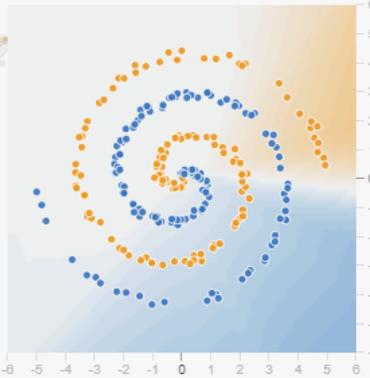


これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

出力層 ⇒ **7** 評価

Train loss (損失) 0.480  
Validation loss 0.483



Train acc (正解率) 0.576  
Validation acc 0.524

データ / ニューロン (neurons) / 重みの値を色で表現。

訓練データ表示     精度検証データ表示  
 テストデータ表示     出力の離散化

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4 "手法" の選択：モデルの定義**

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5 "学習方法" の設計：モデルの生成**

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6 学習：トレーニング**

エポック (Epoch) 000,163

**1 データ準備**

座標点

**2 問題種別**

分類

どのデータセットを使いますか？

**3 前処理**

データの何%を訓練【Training】用に？  
【Validation】用に？  
(残りは精度検証【Validation】用)  
: 50%

ノイズ: 15%

**4**

入力層

入力データとして使いたい特徴を選択してください。

3 隠れ層

6 neurons    6 neurons    8 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1 X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

出力層 ⇒ **7** 評価

Train loss (損失) 0.454  
Validation loss 0.448

Train acc (正解率) 0.632  
Validation acc 0.704

データ / ニューロン (neurons) / 重みの値を色で表現。

訓練データ表示     精度検証データ表示  
 ニューロン表示     出力の離散化

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** "手法" の選択：モデルの定義

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** "学習方法" の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,289

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

データの何%を訓練  
【Training】用に？  
(残りは精度検証  
【Validation】用)

: 50%

レイズ: 15%

入力層

入力データとして使  
いたい特徴を選択し  
てください。

6 neurons

6 neurons

8 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.333  
Validation loss 0.344

Train acc (正解率) 0.776  
Validation acc 0.788

データ / ニュー  
ロン (neurons) / 重み  
の値を色で表現。

訓練データ表示  精度検証データ表示

The screenshot displays a neural network training interface. At the top, there are three main sections: 4. Model Definition, 5. Model Generation, and 6. Training. Section 4 shows the selection of ReLU for hidden layers and Tanh for the output layer, with no regularization. Section 5 shows the selection of Mean Squared Error (MSE) as the loss function and SGD as the optimizer, with a learning rate of 0.03 and a batch size of 10. Section 6 shows the training progress at 289 epochs, with a training loss of 0.333 and a validation loss of 0.344. Below these sections, there are three sub-sections: 1. Data Preparation, 2. Problem Type, and 3. Preprocessing. The main part of the interface shows a neural network diagram with an input layer of 2 nodes, two hidden layers of 6 nodes each, and an output layer of 8 nodes. The nodes are connected by lines representing weights. To the right of the diagram is a visualization of the training data, which is a spiral pattern of points. The training accuracy is 0.776 and the validation accuracy is 0.788. A color scale at the bottom right indicates the values of the data and weights, ranging from -1 to 1.

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4 "手法" の選択：モデルの定義**

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5 "学習方法" の設計：モデルの生成**

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6 学習：トレーニング**

エポック (Epoch) 000,351

**1 データ準備**

座標点

**2 問題種別**

分類

どのデータセットを使いますか？

**3 前処理**

データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)

: 50%

ノイズ: 15%

**4**

入力層

入力データとして使いたい特徴を選択してください。

6 neurons

6 neurons

8 neurons

出力層 ⇒ **7** 評価

Train loss (損失) 0.087

Validation loss 0.112

Train acc (正解率) 0.980

Validation acc 0.968

データ/ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** “学習方法” の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,426

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

データの何%を訓練  
【Training】用に？  
(残りは精度検証  
【Validation】用)

: 50%

ノイズ: 15%

入力層

入力データとして使  
いたい特徴を選択し  
てください。

3 隠れ層

6 neurons    6 neurons    8 neurons

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

出力層 ⇒ **7** 評価

Train loss (損失) 0.023  
Validation loss 0.043

Train acc (正解率) 0.996  
Validation acc 0.980

データ / ニューロン (neurons) / 重みの値を色で表現。

訓練データ表示     精度検証データ表示

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** “手法” の選択：モデルの定義

活性化関数 (隠れ層)	ReLU	正則化	なし
活性化関数 (出力層)	Tanh (分類)	正則化率	0

**5** “学習方法” の設計：モデルの生成

損失関数	平均二乗誤差 (主に回帰)	学習率	0.03
最適化	SGD	バッチサイズ: 10個	<input type="checkbox"/> 全部

**6** 学習：トレーニング

エポック (Epoch) 000,873

**1** データ準備

座標点

**2** 問題種別

分類

どのデータセットを  
使いますか？

**3** 前処理

データの何%を訓練  
【Training】用に？  
(残りは精度検証  
【Validation】用)

: 50%

ノイズ: 15%

入力層

入力データとして使  
いたい特徴を選択し  
てください。

3 隠れ層

6 neurons    6 neurons    8 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1 X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

これは、このニュー  
ロン (neuron)  
の出力内容です。  
マウスを載せる  
と、右の出力の場  
所に拡大します。

出力は、変化する  
重みと合成されま  
す。重み (weights)  
は、線の太さで示  
されています。

出力層 ⇒ **7** 評価

Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロ  
ン (neurons) / 重み  
の値を色で表現。

訓練データ表示     精度検証データ表示  
 テストデータ表示     出力の離散化

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** “手法” の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)  
正則化 なし  
正則化率 0

**5** “学習方法” の設計：モデルの生成  
損失関数 平均二乗誤差 (主に回帰)  
学習率 0.03

**6** 学習：トレーニング  
エポック (Epoch) 000,873

900エポック近く  
学習してみた

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理  
データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)  
: 50%  
ノイズ: 15%

入力層  
入力データとして使いたい特徴を選択してください。

出力層 ⇒ **7** 評価  
Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

**4** “手法” の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)  
正則化 なし  
正則化率 0

**5** “学習方法” の設計：モデルの生成  
損失関数 平均二乗誤差 (主に回帰)  
学習率 0.03

**6** 学習：トレーニング  
エポック (Epoch) 000,873

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理  
データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)  
: 50%  
ノイズ: 15%

入力層  
入力データとして使いたい特徴を選択してください。

出力層 ⇒ **7** 評価  
Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

ある程度綺麗に分類が出来ている

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

The image shows a neural network training interface. The main part of the interface displays a neural network diagram with three layers: an input layer with 6 neurons, two hidden layers with 6 neurons each, and an output layer with 8 neurons. The connections between neurons are shown as lines of varying thickness, representing weights. To the right of the diagram is a scatter plot showing the training data points (blue and orange) and a decision boundary (a spiral shape) that separates the two classes. The plot is titled '出力層 ⇒ 7 評価' and shows training and validation loss and accuracy. The training loss is 0.005 and the validation loss is 0.028. The training accuracy is 0.996 and the validation accuracy is 0.984. A blue callout box points to the decision boundary with the text 'ある程度綺麗に分類が出来ている'. A red box highlights the scatter plot. The interface also includes settings for activation functions, regularization, learning rate, and data preparation options.

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

4 “手法”の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)

5 “学習方法”の設計：モデルの生成  
学習率 0.03  
バッチサイズ: 10個  
エポック (Epoch) 000,873

6 学習：トレーニング

1 データ準備  
座標点

2 問題種別  
分類  
どのデータセットを使いますか？

3 前処理  
データの何%を訓練【Training】用に？  
【Validation】用)  
: 50%  
ノイズ: 15%

入力層  
入力データとして使いたい特徴を選択してください。

3 隠れ層

6 neurons 6 neurons 8 neurons

出力層 ⇒ 7 評価  
Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

☑ 訓練データ表示 ☐ 精度検証データ表示  
☑ テストデータ表示 ☐ 出力の離散化

何故この判断をしたのかは分からないが複雑な判断が下せている

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。



# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

4 "手法" の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)

5 "学習方法" の設計：モデルの生成  
学習率 0.03  
バッチサイズ: 10個  
エポック (Epoch) 000,873

6 学習：トレーニング

出力層 ⇒ 7 評価  
Train loss (損失) 0.005  
Validation loss 0.028

3 前処理  
データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)  
: 50%  
ノイズ: 15%

これは、このニューロン (neuron) の出力内容です。マウスを移動すると、右の出力の値が拡大されます。

計算量こそすべてを解決する...!!

何故この判断をしたのかは分からないが複雑な判断が下せている

がっ.....!  
中身はブラックボックスっ!

# 3層でそれぞれ6、6、8ノードだと：2分類問題その3

The screenshot shows a neural network training interface with a central image of a cat looking at a black hole. The interface is divided into several sections:

- 4 "手法" の選択：モデルの定義**
  - 活性化関数 (隠れ層): ReLU
  - 活性化関数 (出力層): Tanh (分類)
- 6 学習：トレーニング**
  - エポック (Epoch): 000,873
- 7 評価**
  - 損失 (損失) 0.005
  - 精度 0.028
- 3 前処理**
  - データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用) : 50%
  - ノイズ: 15%

Additional text and elements:

- 何故この判断から複雑な判断 (Why this judgment from a complex judgment)
- 計算量こそすべてを解決する...!! (Computational power is what solves everything...!!)
- まさに宇宙猫! (It's really a space cat!)
- 中身はブラックホックスっ! (The inside is a black hole!)

# 詳細を見てみると

**4** “手法” の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)  
正則化 なし  
正則化率  
最適化

**5** “学習方法” の設計：モデルの生成  
損失関数 平均二乗誤差 (主に回帰)  
学習率 0.03  
バッチサイズ: 10個  
エポック (Epoch) 000,873

**6** 学習：トレーニング

**1** データ準備  
座標点

**2** 問題種別  
分類  
どのデータセットを使いますか？

**3** 前処理  
データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)  
: 50%  
ノイズ: 15%

1ノードで1個のパーセプトロンを表している。

出力層 ⇒ **7** 評価  
Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。  
-1 0 1

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

# 詳細を見てみると

**4** “手法” の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)

**5** “学習方法” の設計：モデルの生成  
学習率 0.03  
バッチサイズ: 10個  
エポック (Epoch) 000,873

**6** 学習：トレーニング

出力層 ⇒ **7** 評価  
Train loss (損失) 0.005  
Validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。  
-1 0 1

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

**この場合**  
 $X_1 * W_1 + X_2 * W_2 + b$   
を【活性化関数】に通したものが  
出力されている

入力データとして使いたい特徴を選択してください。

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

The image shows a screenshot of a neural network training interface. On the left, there are settings for data preparation, problem type, and preprocessing. The main area displays a neural network diagram with three layers: an input layer with 6 neurons, a hidden layer with 6 neurons, and an output layer with 8 neurons. A callout box highlights the input layer and the equation  $X_1 * W_1 + X_2 * W_2 + b$ . On the right, there are plots for training and validation loss, and training and validation accuracy. A scatter plot shows the training and validation data points. The interface also includes a play button to start training and a progress indicator for the number of epochs.

# 詳細を見てみると

4 “手法” の選択：モデルの定義  
活性化関数 (隠れ層) ReLU  
活性化関数 (出力層) Tanh (分類)

5 “学習方法” の設計：モデルの生成

6 学習：トレーニング 0,873

例えばここ場所でも入力をxとすると  
 $X_1 * W_1 + X_2 * W_2 + X_3 * W_3 + X_4 * W_4 + X_5 * W_5 + X_6 * W_6 + b$   
を【活性化関数】に通したものが  
右から出力されている

1 データ準備  
座標点

2 問題種別  
分類  
どのデータセットを使いますか？

3 前処理  
データの何%を訓練【Training】用に？ (残りは精度検証【Validation】用)  
: 50%  
ノイズ: 15%

6 neurons 6 neurons

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1 X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

# 詳細を見てみると

この場所でも

$$X_1 * W_1 + X_2 * W_2 + X_3 * W_3 + X_4 * W_4 + X_5 * W_5 + X_6 * W_6 + b$$

を活性化関数を通したものが出力となるように必ず

**【入力値 × 重み + b (バイアス)】に  
【活性化関数】を通したものが出力となる。**

The screenshot shows a neural network simulator interface. On the left, there are settings for '手法' (Method) with 'モデル' (Model) selected, '活性化関数 (隠れ層)' (Activation function (Hidden layer)) set to 'ReLU', and '活性化関数 (出力層)' (Activation function (Output layer)) set to 'Tanh (分類)' (Tanh (Classification)). Below are sections for 'データ準備' (Data preparation), '問題種別' (Problem type) set to '分類' (Classification), and '前処理' (Preprocessing) with 'データの何%を訓練' (What percentage of data for training) set to 50% and 'ノイズ' (Noise) at 15%. The main area displays a neural network diagram with 6 input neurons, 6 hidden neurons, and 6 output neurons. A red box highlights the hidden layer, and a red arrow points from the text above to the output layer. A legend indicates that the thickness of the lines represents weights. To the right, a plot shows a spiral pattern of data points with a 'validation loss 0.028'. Training metrics are shown as 'Train acc (正解率) 0.996' and 'Validation acc 0.984'. A color scale for 'データ / ニューロン (neurons) / 重み (weights) の値を色で表現' (Representing the value of data/neurons/weights by color) ranges from -1 to 1. Checkboxes for '訓練データ表示' (Training data display), '精度検証データ表示' (Validation data display), 'テストデータ表示' (Test data display), and '出力の離散化' (Discretization of output) are visible at the bottom.

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、このニューロンの重みと合成されます。重み (weights) は、線の太さで示されています。

validation loss 0.028

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

# まとめ

- 経験則としてNNの層やノードが深ければ深いほど複雑な判断が対応できる。
- もちろん、その分学習コストは重くなる。
- NNは

【 $y=ax+b$ 】と【活性化関数】をたくさん繋げている！

と言える！

# まとめ

- 経験則としてNNの層やノードが深ければ深いほど複雑な判断が対応できる。
- もちろん、その分学習コストは重くなる。
- NNは

【 $y=ax+b$ 】と【活性化関数】をたくさん繋げている！

と言える！

強引すぎるまとめ方っ！

だが、それがいい！！

# おまけ：今回の活性化関数

The screenshot shows a neural network training interface with the following components:

- 4 "手法" の選択：モデルの定義**
  - 活性化関数 (隠れ層): ReLU (highlighted with a red box)
  - 活性化関数 (出力層): Tanh (分類)
- 5 "学習方法" の設計：モデルの生成**
  - 損失関数: 平均二乗誤差 (主に回帰)
  - 学習率: 0.03
  - エポック (Epoch): 000,873
- 6 学習：トレーニング**
  - Train loss (損失): 0.005
  - Validation loss: 0.028
  - Train acc (正解率): 0.996
  - Validation acc: 0.984

The central part of the interface displays a neural network diagram with 6 input neurons, 3 hidden layers of 6 neurons each, and 8 output neurons. A callout box points to the hidden layer activation function selection:

今回は最後の奴は  
隠れ層にReLU関数を使用

Additional text in the diagram explains that the output of a neuron is a combination of weights, shown by the thickness of the lines connecting neurons.

出力層 ⇒ 7 評価

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

Train acc (正解率) 0.996  
Validation acc 0.984

Train loss (損失) 0.005  
Validation loss 0.028

エポック (Epoch) 000,873

学習率 0.03

損失関数 平均二乗誤差 (主に回帰)

活性化関数 (隠れ層) ReLU

活性化関数 (出力層) Tanh (分類)

正則化 なし

最適化 勾配降下法

ニューロンサイズ: 10個

全部

1 データ準備  
座標点

2 問題種別  
分類  
どのデータセットを使いますか?

3 前処理  
データの何%を訓練【Training】用に?  
【Validation】用)  
(残りは精度検証【Validation】用)  
: 50%

ノイズ: 15%

これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。

出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

Train acc (正解率) 0.996  
Validation acc 0.984

データ / ニューロン (neurons) / 重み (weights) の値を色で表現。

訓練データ表示  精度検証データ表示  
 テストデータ表示  出力の離散化

# おまけ：今回の活性化関数

The screenshot shows a neural network training interface with several key elements and annotations:

- 4 "手法" の選択：モデルの定義**
  - 活性化関数 (隠れ層): ReLU
  - 活性化関数 (出力層): **Tanh (分類)** (highlighted with a red box)
- 5 "学習方法" の設計：モデルの生成**
- 6 学習：トレーニング**
  - エポック (Epoch): 000,873
- 1 データ準備**
  - 座標点
- 2 問題種別**
  - 分類
  - どのデータセットを使いますか?
- 3 前処理**
  - データの何%を訓練【Training】用に? (残りは精度検証【Validation】用): 50%
  - ノイズ: 15%
- 入力層**
  - 入力データとして使いたい特徴を選択してください。
  - 特徴:  $X_1$ ,  $X_2$ ,  $X_1^2$ ,  $X_2^2$ ,  $X_1 X_2$ ,  $\sin(X_1)$ ,  $\sin(X_2)$
- 隠れ層**
  - 6 neurons
  - 6 neurons
  - 8 neurons
- 出力層**
  - 1 neuron (highlighted with a red box)
- 出力値**: 0.028
- 可視化**
  - Scatter plot of training data (orange and blue points).
  - Heatmap of the output layer.
  - Train acc (正解率) 0.996
  - Validation acc 0.984
  - データ / ニューロン (neurons) / 重み (weights) の値を色で表現。
  - Legend: 訓練データ表示, 精度検証データ表示, テストデータ表示, 出力の離散化
- Annotations**
  - 出力層にTanh関数を使用 (Callout pointing to the Tanh selection in the top menu).
  - 見えていないがここにも1個出力用のノードがある (Callout pointing to a hidden node in the 8-neuron layer).
  - これは、このニューロン (neuron) の出力内容です。マウスを載せると、右の出力の場所に拡大します。
  - 出力は、変化する重みと合成されます。重み (weights) は、線の太さで示されています。

# おわり

## ありがとうございました

第2回  $y=ax+b$ から始める初心者向けML講座

次回はタスク別の活性化関数と損失関数ニューラルネットワークの組み合わせ方法

(多分)

他にLTやる奴が  
いないだって!?

おわり

ヒヤア!!!  
我慢できねえ!!

第...  $y=a+bx$  から始める初心者向けML講座

次回はタスク別の活性化関数と損失関数ニューラルネットワークの組み合わせ方法

(多分)

## 代表的な活性化関数

- ReLU関数
- Leaky ReLU関数
- シグモイド関数
- ソフトマックス関数
- Tanh（ハイパボリックタンジェント）関数

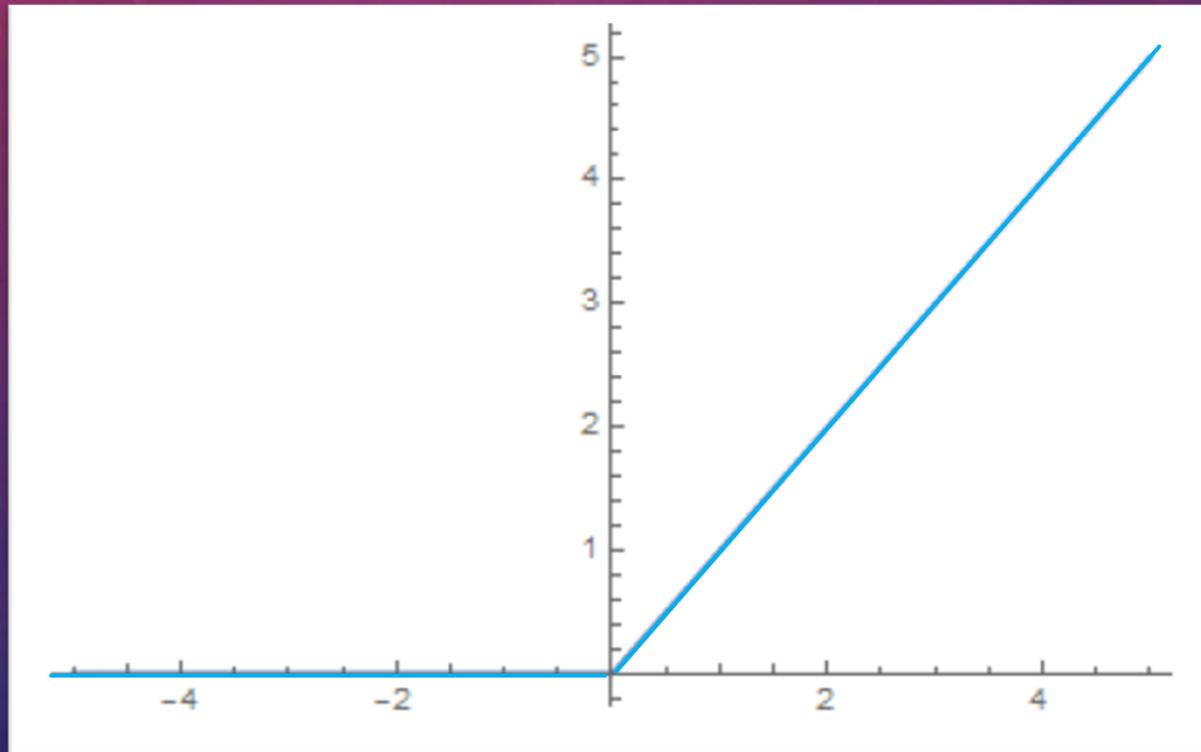
などがあります。

# ReLU関数

- $F(x) = \max\{0, x\}$
- 0かX大きい方を出力 (0以上ならそのまま出力)



ここで通す関数だよ!



# ReLU関数

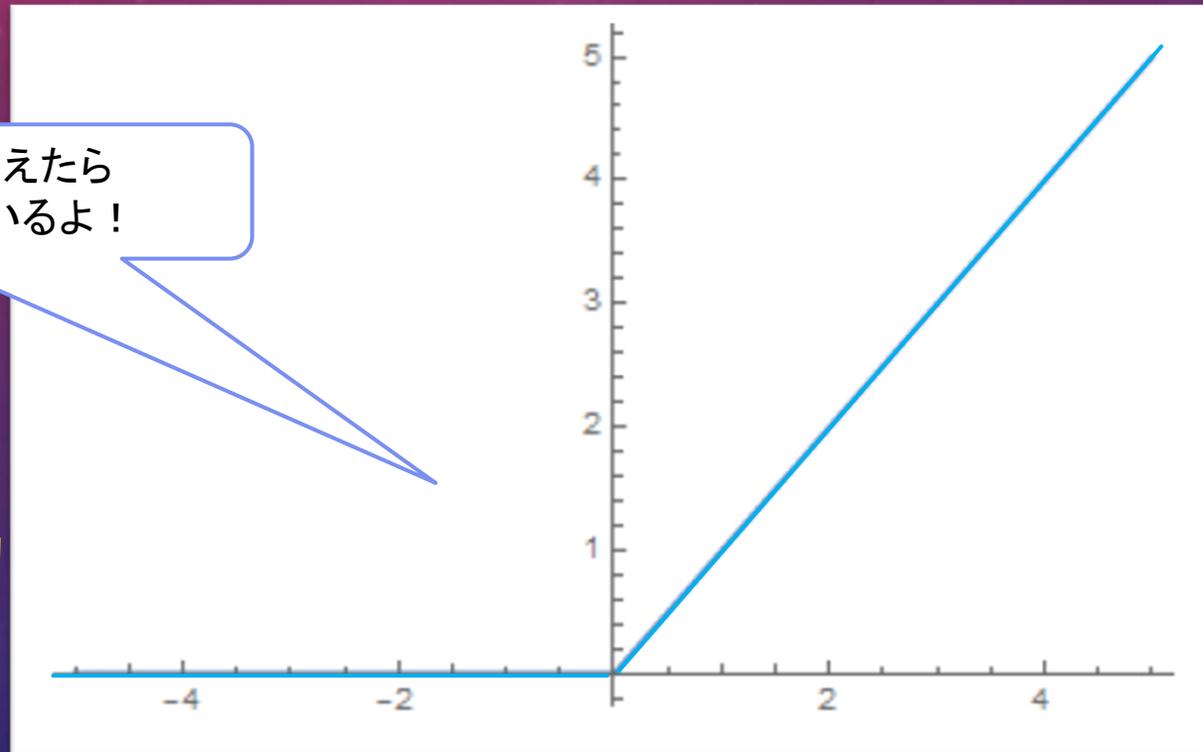
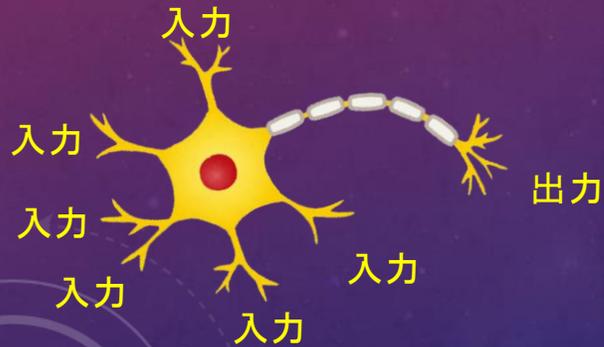
- $F(x) = \max\{0, x\}$
- 0かX大きい方を出力 (0以上ならそのまま出力)



隠れ層はこれを使うのが一般的だよ！

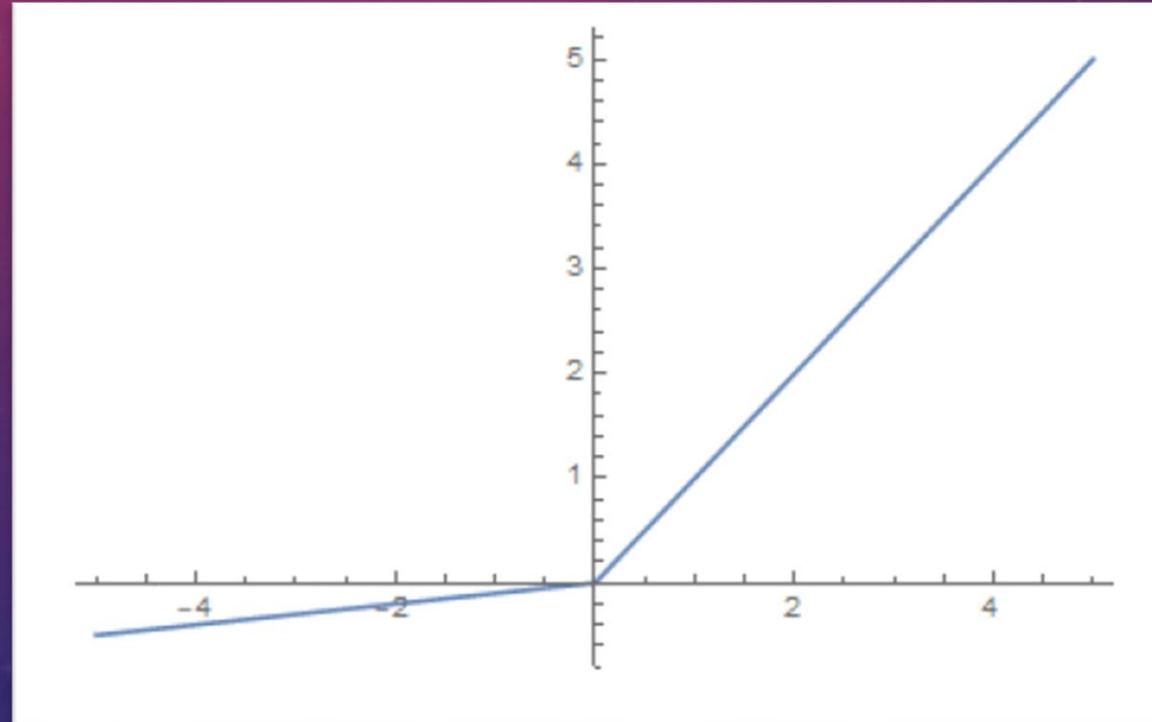
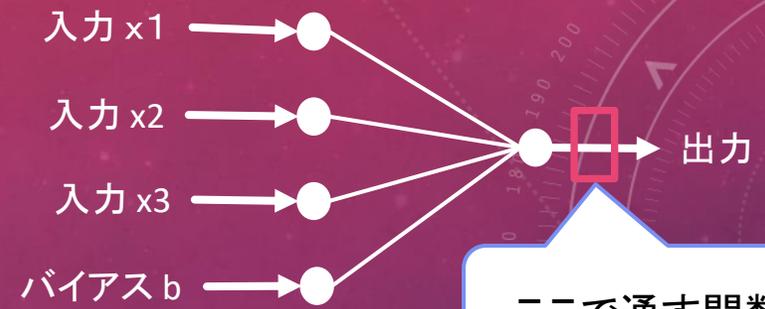
ここで通す関数だよ！

ニューロンが閾値を越えたら  
発火するのを意識しているよ！



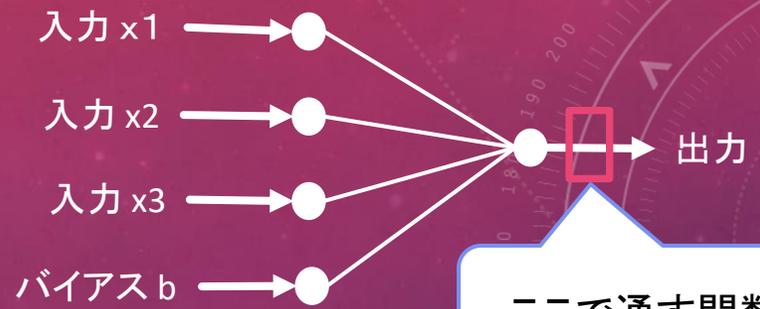
# Leaky ReLU関数

•  $F(x) = \begin{cases} x & (x > 0) \\ 0.1x & (x \leq 0) \end{cases}$  を通したものを出力



# Leaky ReLU関数

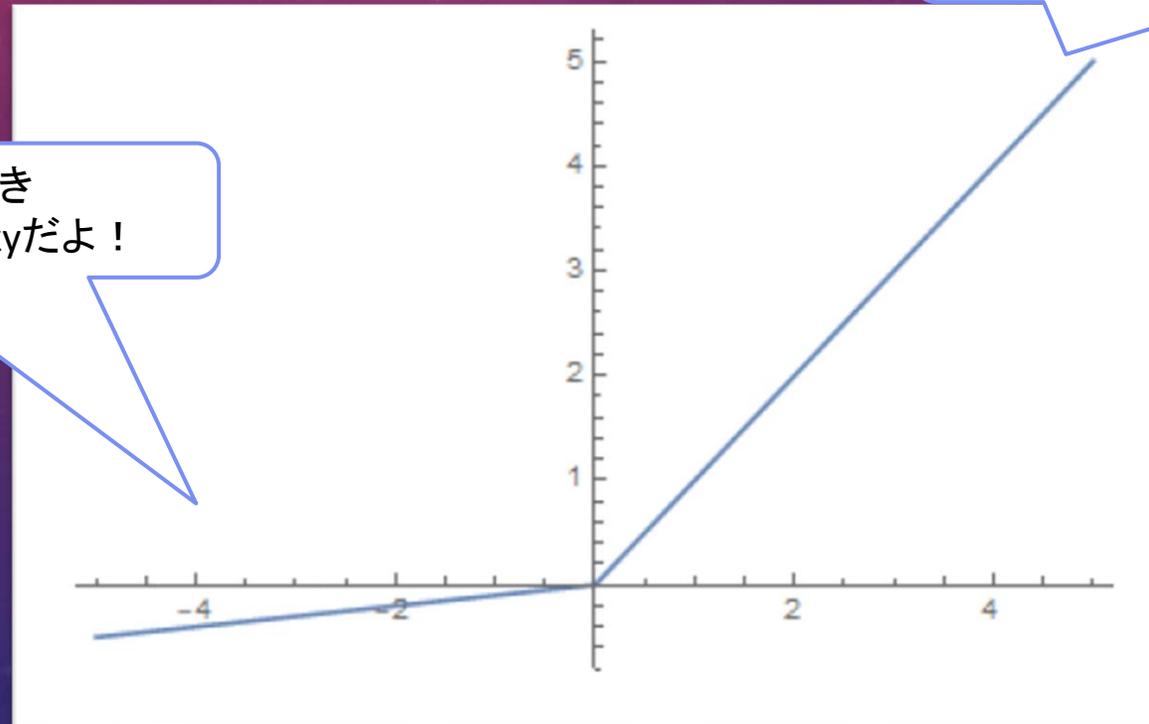
•  $F(x) = \begin{cases} x & (x > 0) \\ 0.1x & (x \leq 0) \end{cases}$  を通したものを出力



ここで通す関数だよ！

ReLU関数の代わりに使われるよ

入力xがマイナスのとき  
ちよつとだけ漏らすからLeakyだよ！

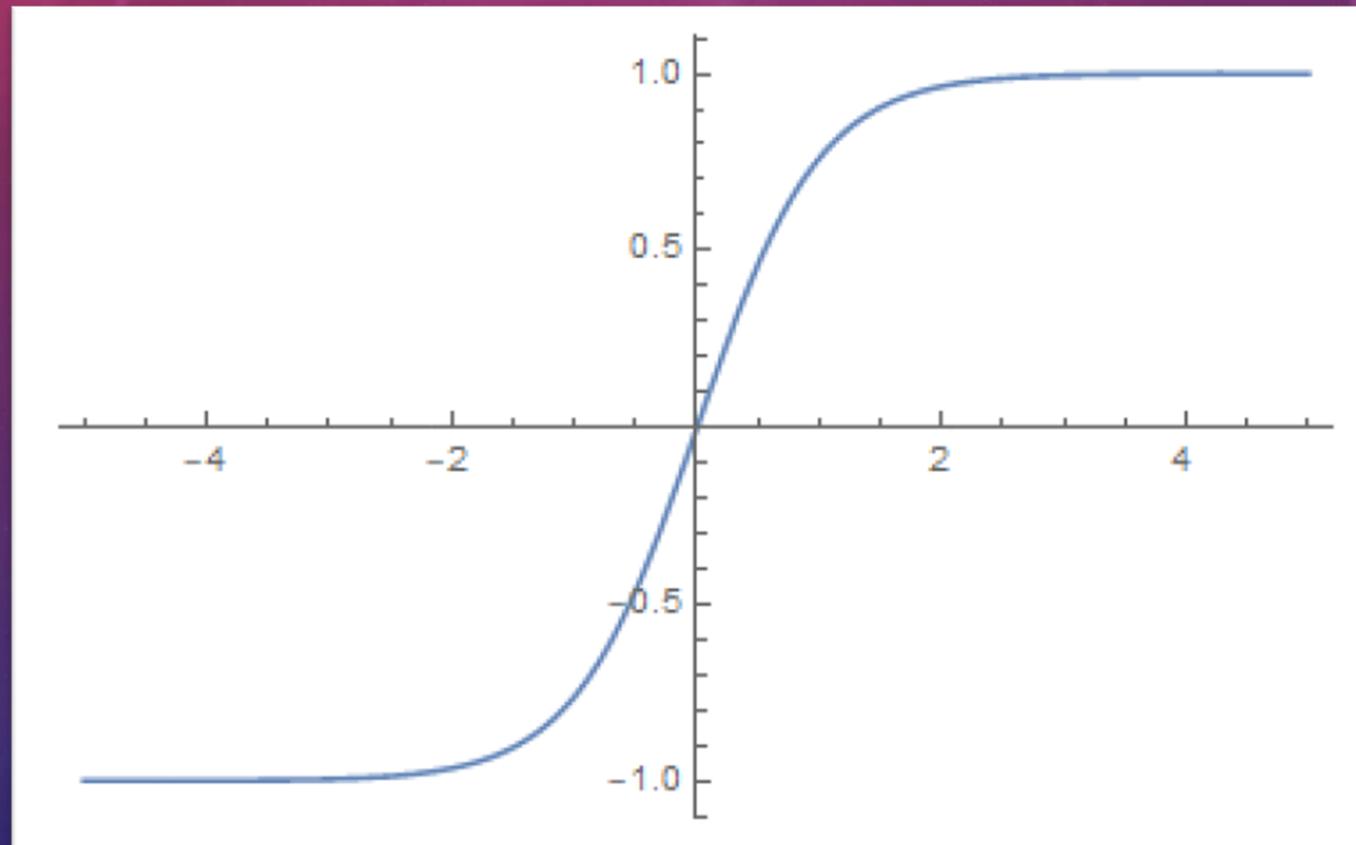


# Tanh (ハイパボリックタンジェント) 関数

- $F(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$  を通したものを出力



ここで通す関数だよ!



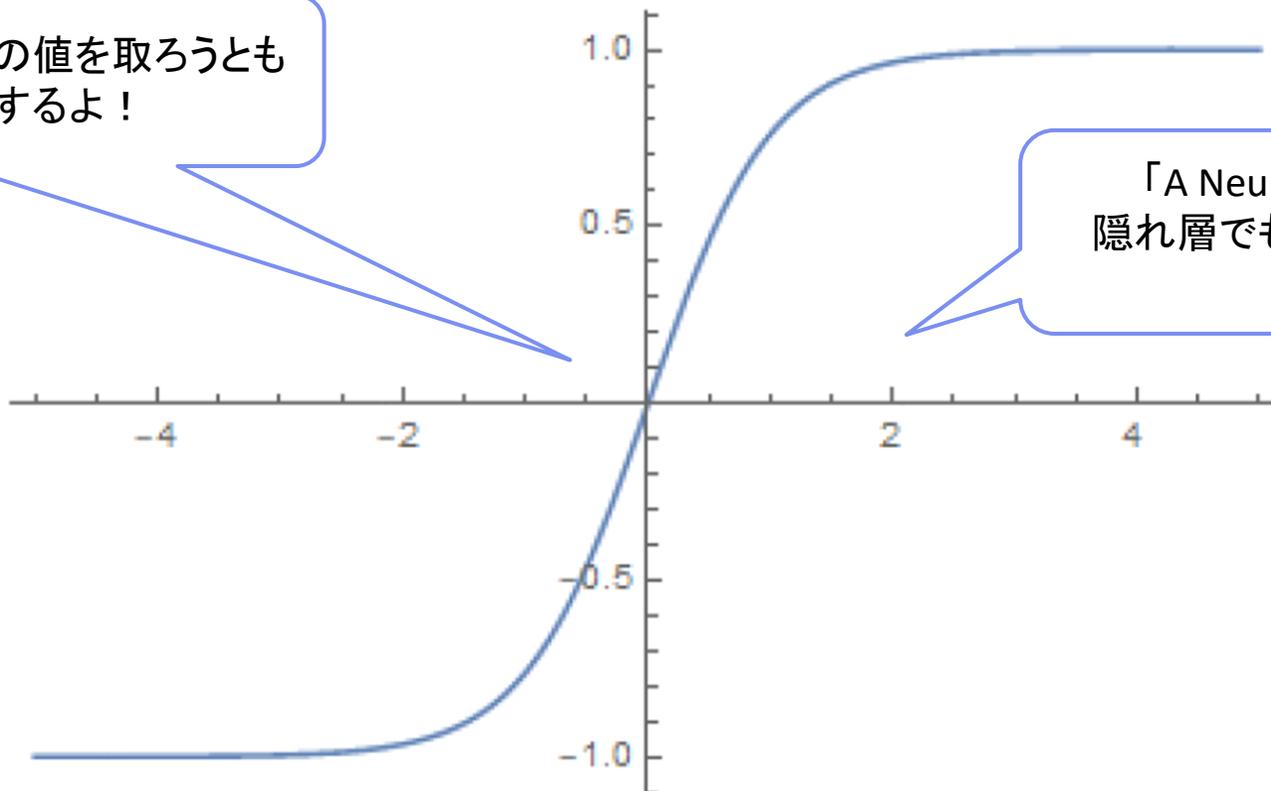
# Tanh (ハイボリックタンジェント) 関数

•  $F(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$  を通したものを出力



ここで通す関数だよ!

出力は入力(x)がどれぐらいの値を取ろうとも  
-1~+1の範囲で出力するよ!



「A Neural Network Playground」では  
隠れ層でも出力層でもデフォルトでこれが  
使われているよ

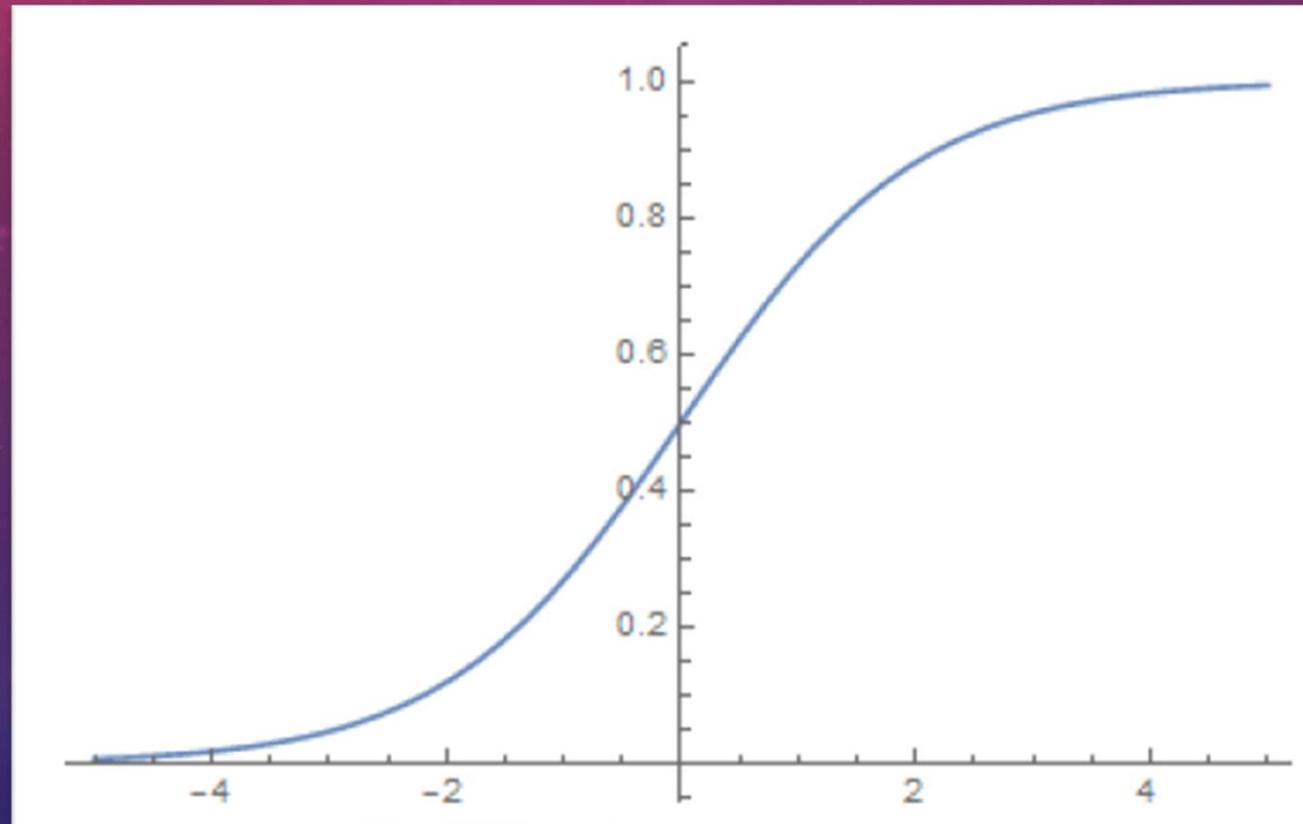
何故かは知らないよ!

# シグモイド関数

•  $F(x) = \frac{1}{1+e^{-x}}$  を通したものを出力



ここで通す関数だよ!



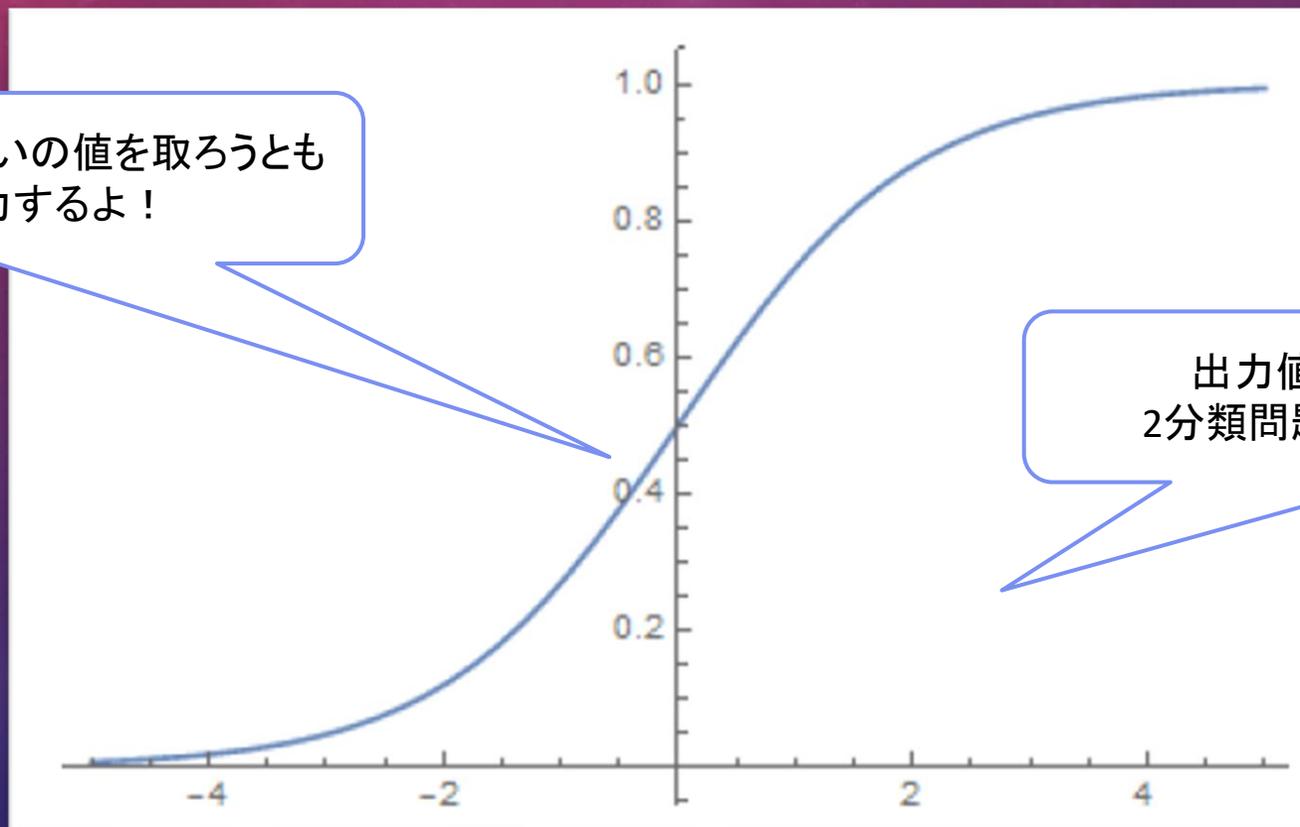
# シグモイド関数

•  $F(x) = \frac{1}{1+e^{-x}}$  を通したものを出力



ここで通す関数だよ!

出力は入力(x)がどれぐらいの値を取ろうとも  
0~1の範囲で出力するよ!

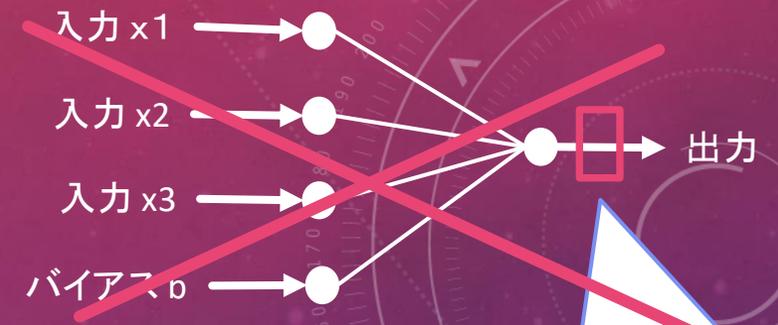


出力値を確率として扱えるから  
2分類問題の出力層に使われるよ!

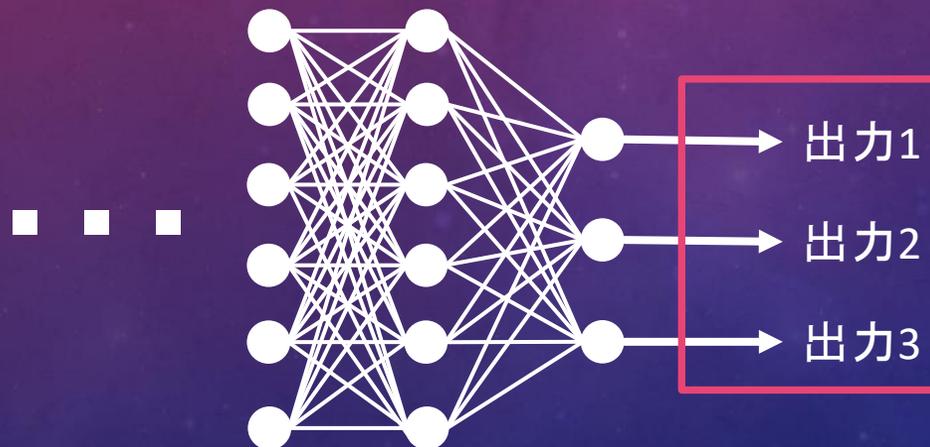
# ソフトマックス関数

- 複数分類のとき出力層で使われる関数
- 出力の合計値が1になる

- $[Y_0, Y_1, \dots, Y_{k-1}] = \frac{[e^{x_0}, e^{x_1}, \dots, e^{x_{k-1}}]}{e^{x_0} + e^{x_1} + \dots + e^{x_{k-1}}}$



ノード1個では意味ないよ！

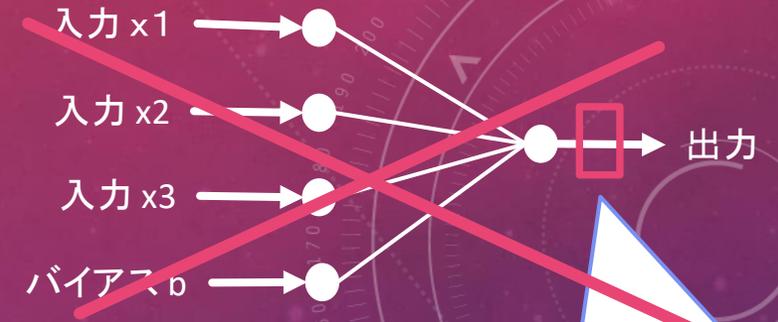


出力の合計が1、つまり合計100%の確率として扱えるよ！

出力層が2個以上  
つまり、複数分類問題  
の出力層で使われるよ！

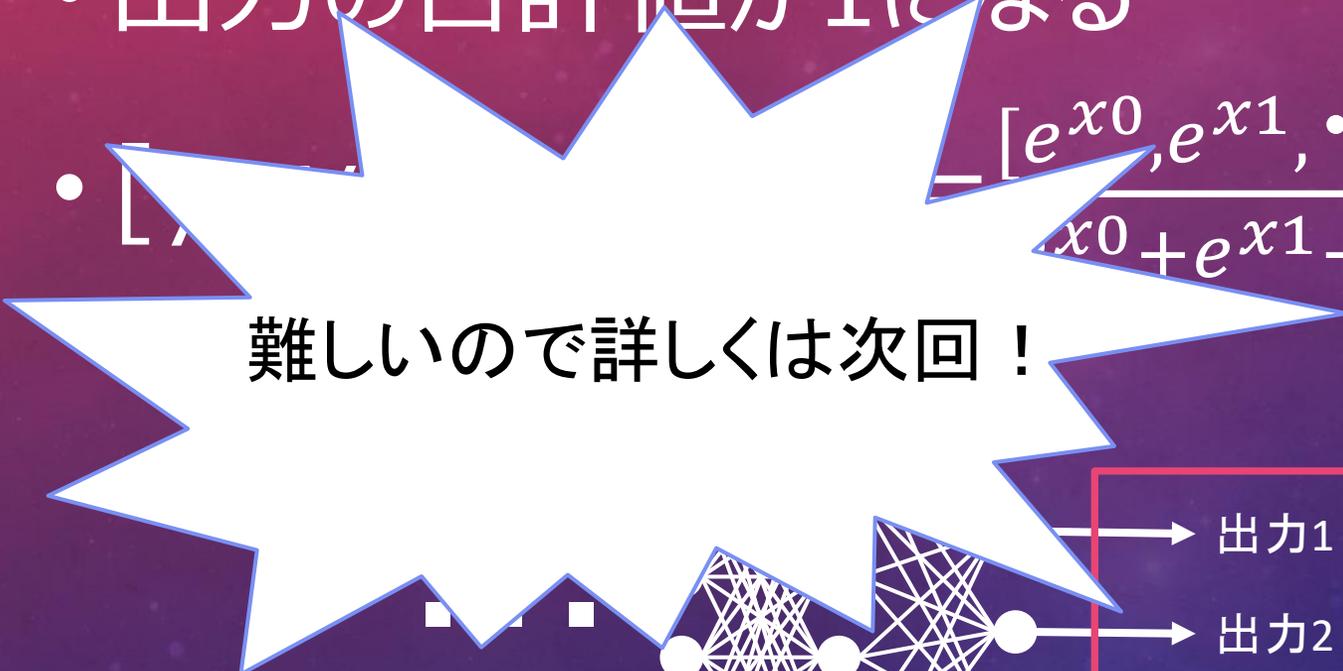
# ソフトマックス関数

- 複数分類のとき出力層で使われる関数
- 出力の合計値が1になる

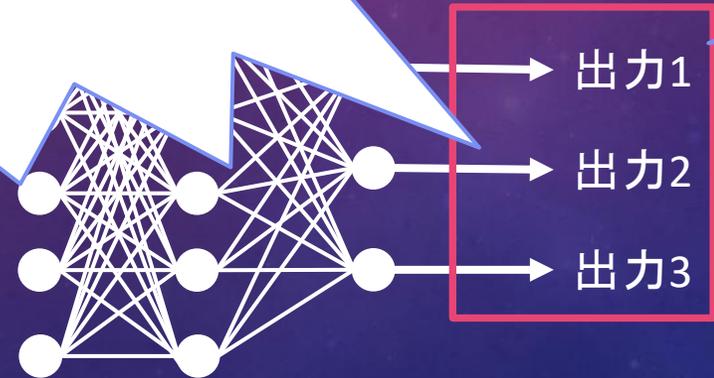


ノード1個では意味ないよ！

- $$\frac{[e^{x_0}, e^{x_1}, \dots, e^{x_{k-1}}]}{e^{x_0} + e^{x_1} + \dots + e^{x_{k-1}}}$$



難しいので詳しくは次回！

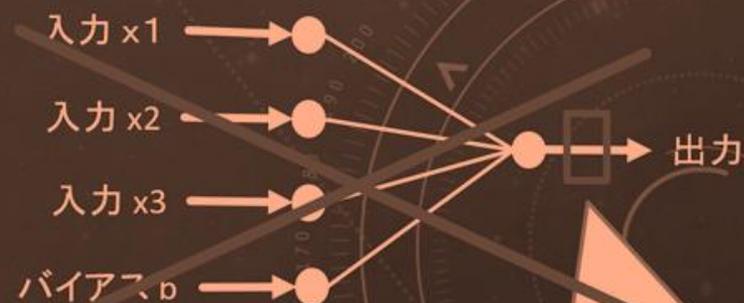


出力の合計が1、つまり合計100%の確率として扱えるよ！

出力層が2個以上  
つまり、複数分類問題  
の出力層で使われるよ！

# ソフトマックス関数

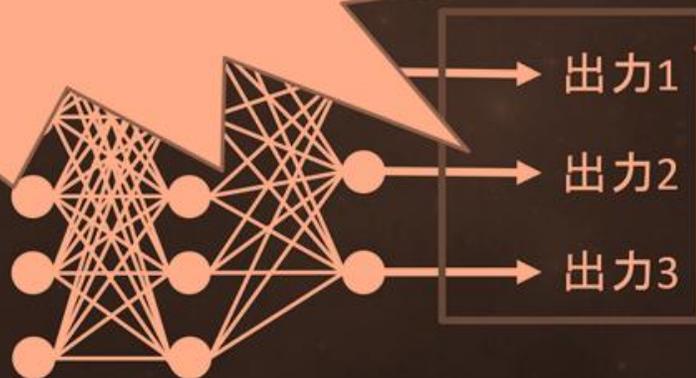
- 複数分類のとき出力層で使われる関数
- 出力の合計値が1になる



ノード1個では意味ないよ！

$$\frac{[e^{x_0}, e^{x_1}, \dots, e^{x_{k-1}}]}{e^{x_0} + e^{x_1} + \dots + e^{x_{k-1}}}$$

難しいので詳しくは次回！

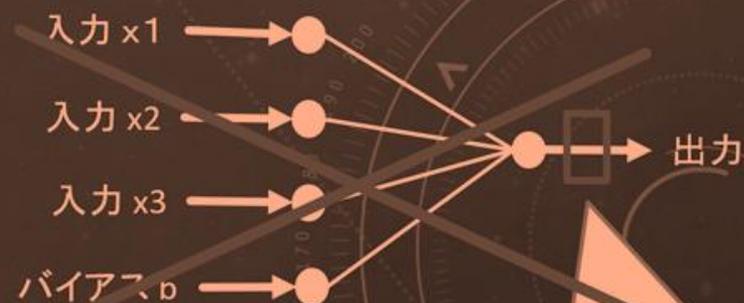


出力の合計が1、つまり合計100%の確率として扱えるよ！

出力層が2個以上  
つまり、複数分類問題  
の出力層で使われるよ！

# ソフトマックス関数

- 複数分類のとき出力層で使われる関数
- 出力の合計値が1になる

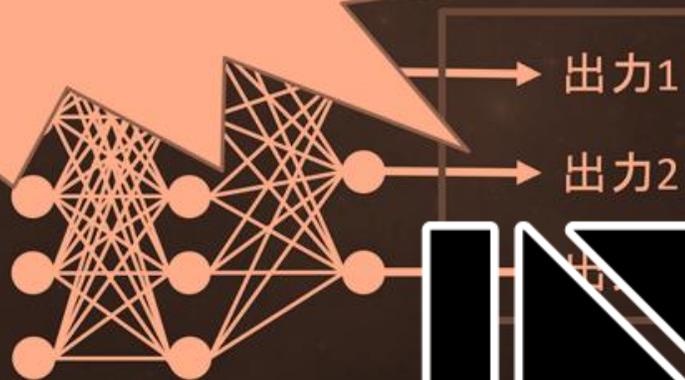


ノード1個では意味ないよ！

$$\frac{[e^{x_0}, e^{x_1}, \dots, e^{x_{k-1}}]}{e^{x_0} + e^{x_1} + \dots + e^{x_{k-1}}}$$

難しいので詳しくは次回！

出力の合計が1、つまり合計100%の確率として扱えるよ！



To Be Continued

# 次回予告

## タスク別の 活性化関数と損失関数 ニューラルネットワークの組み合わせ方法

# こんどこそ おわり

## ありがとうございました

第2回  $y=ax+b$ から始める初心者向けML講座

次回はタスク別の活性化関数と損失関数ニューラルネットワークの組み合わせ方法

(多分)