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＋α、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

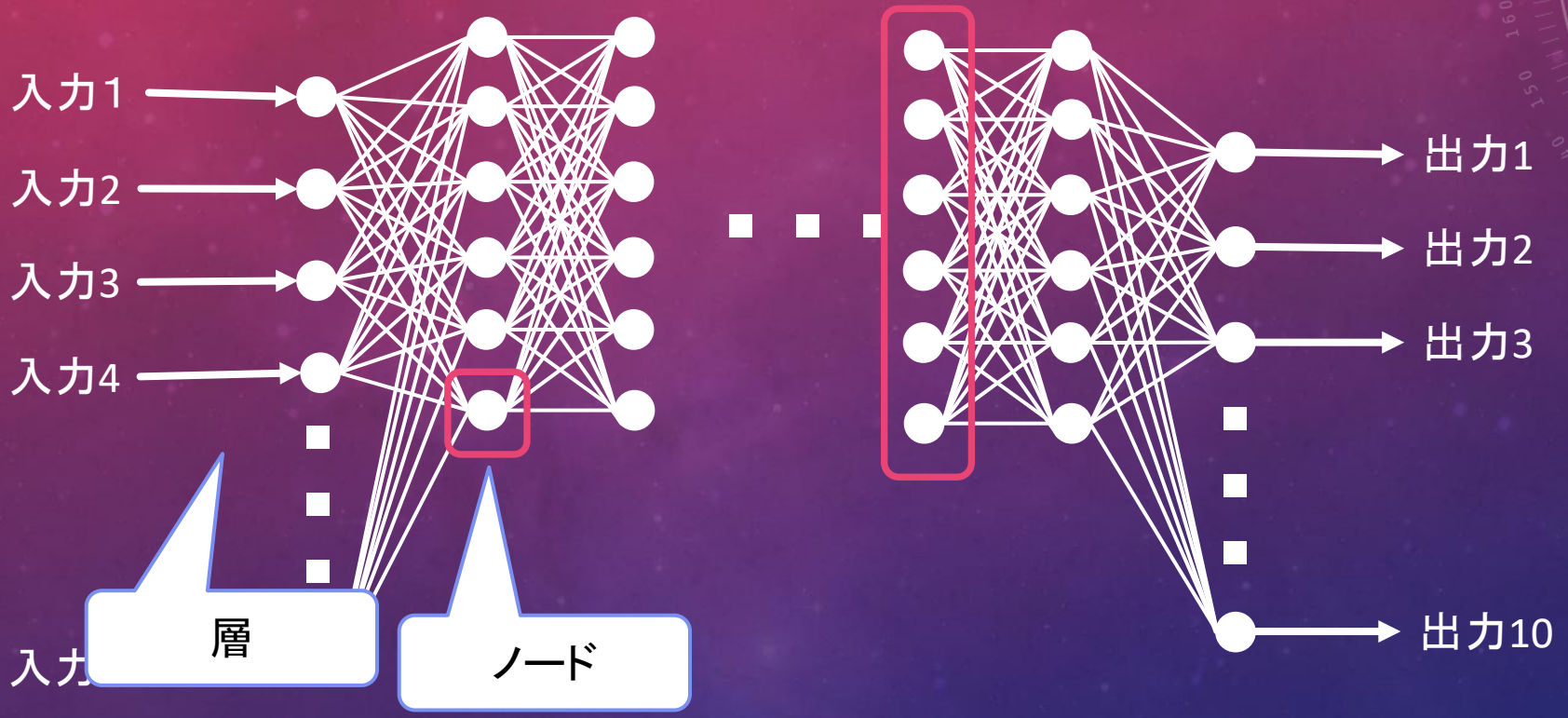
やることが‥
やることが多い‥！！

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

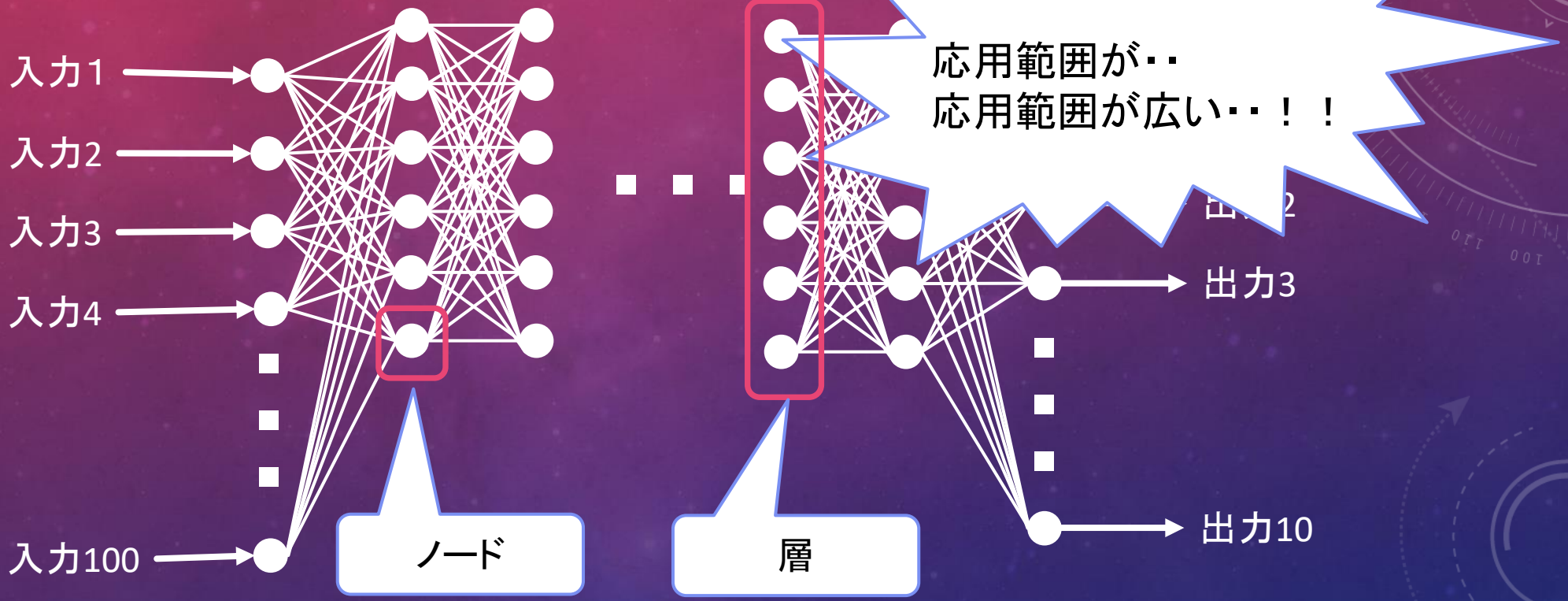
- 「入力」や「出力」「各層のノードの数」や「各層での処理」や「層の数」などを変更していけば様々な応用が可能！



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て使うなどしていくと
stable diffusionやChatGTPにもなっていきます。

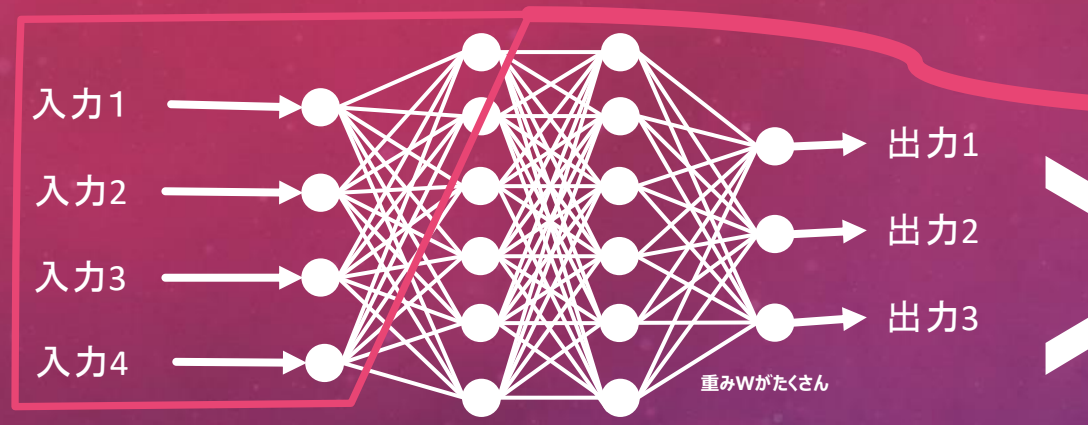
ディープなニューラルネットワーク(NN)が主流

- 「入力」や「出力」「各層のノードの数」や「各層での処理」や「層の数」など変更[illegible]が可能！

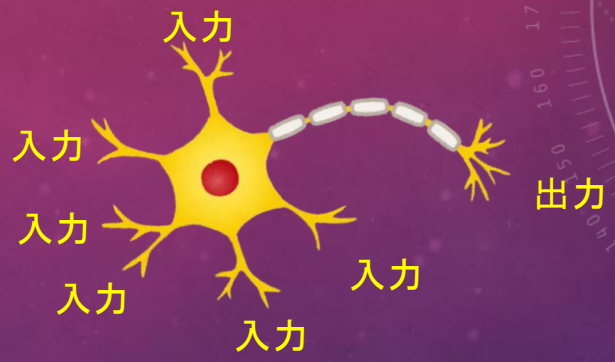


様々な応用的な事を組み合わせたり、複数のモデルを組み合わせて使うなどしていくと
stable diffusionやChatGTPにもなっていきます。

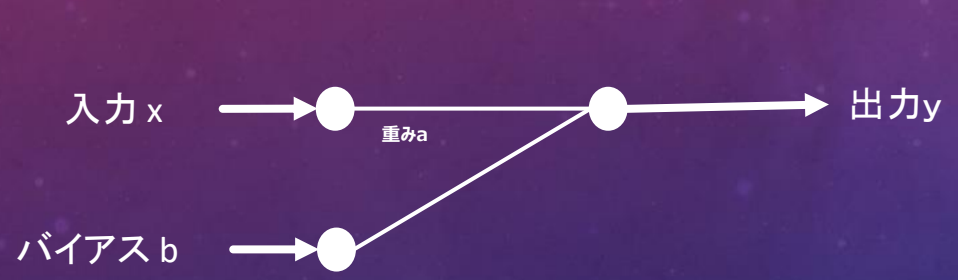
NNの一部を抜き出しても学習原理は同じ



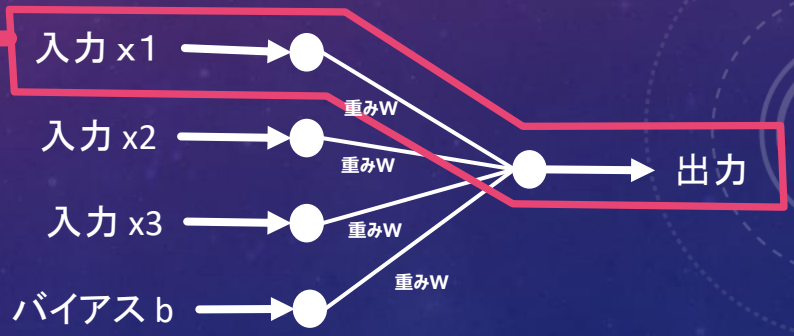
ニューラルネットワーク(NN)



ニューロン

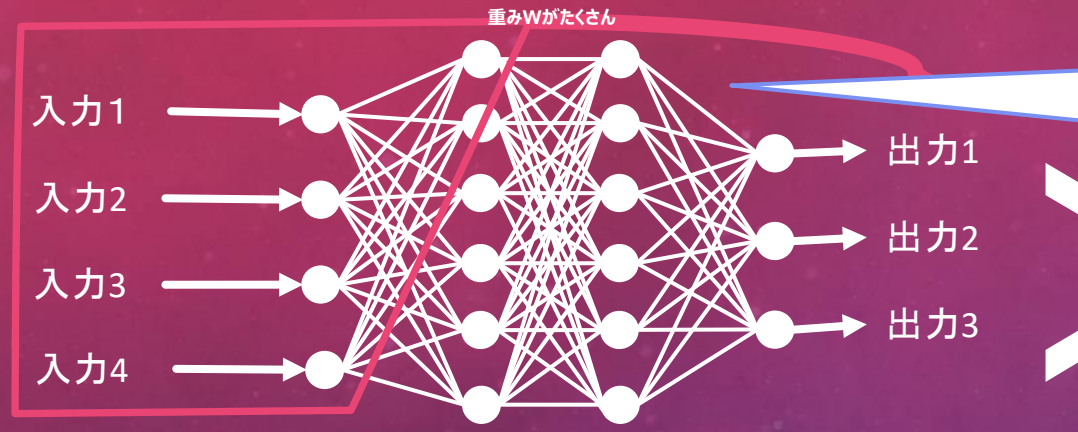


y=ax+b



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

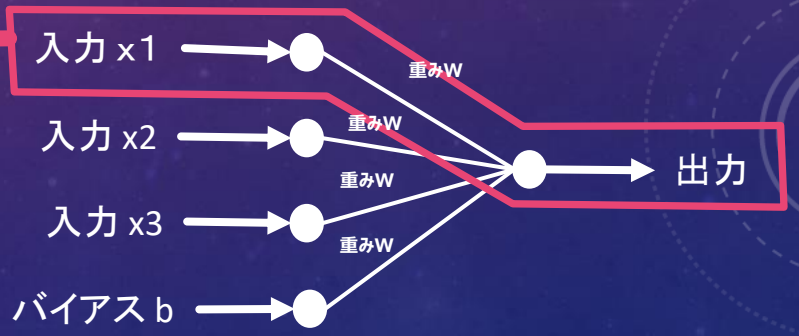


ニューラルネットワーク(NN)

線1本1本のところにある
【重みW】を【予測値ー正解値】
から別々に学習する！

ニューロン

=



パーセプトロン

<

y=ax+b

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導
- 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
- 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

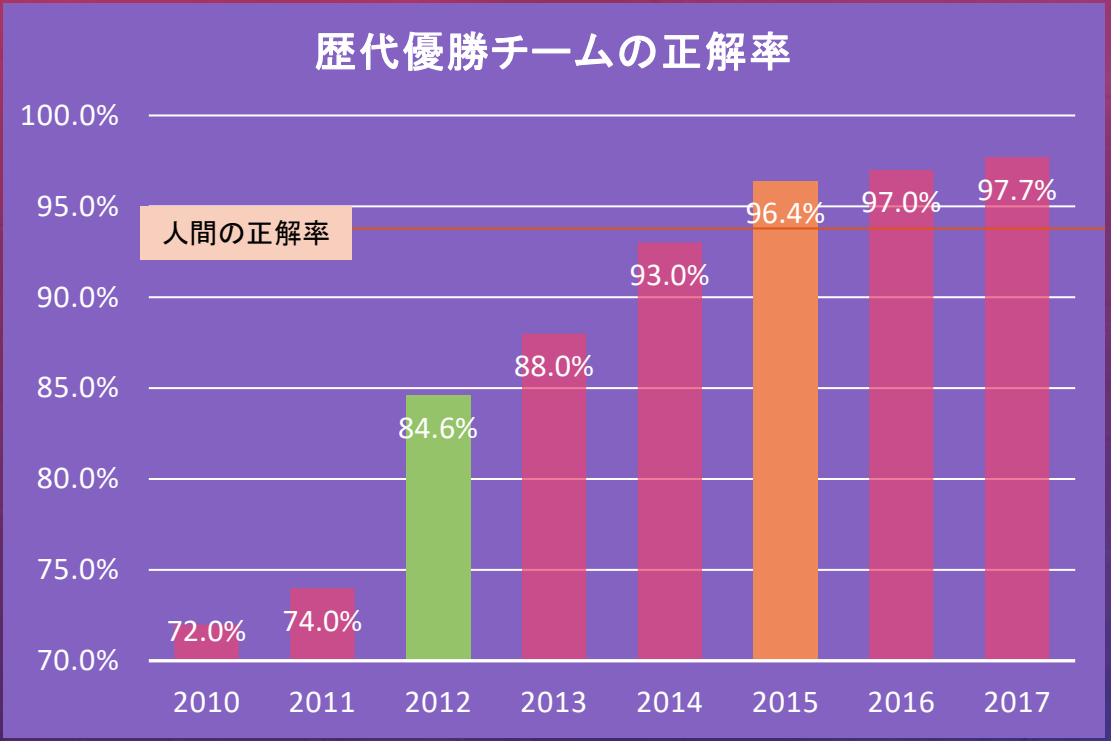
  y＝ax＋bの機械学習がニューラルネットワークの基本

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！



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

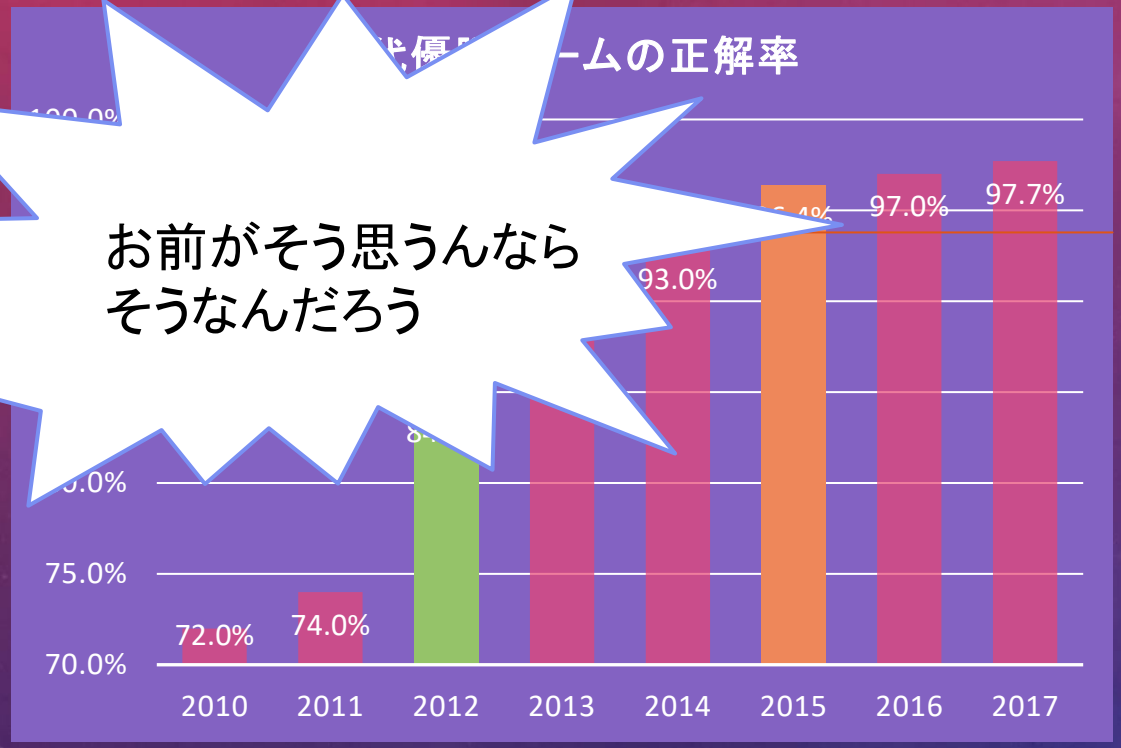
2位と10%以上の大差を付け勝利し、

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。

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！

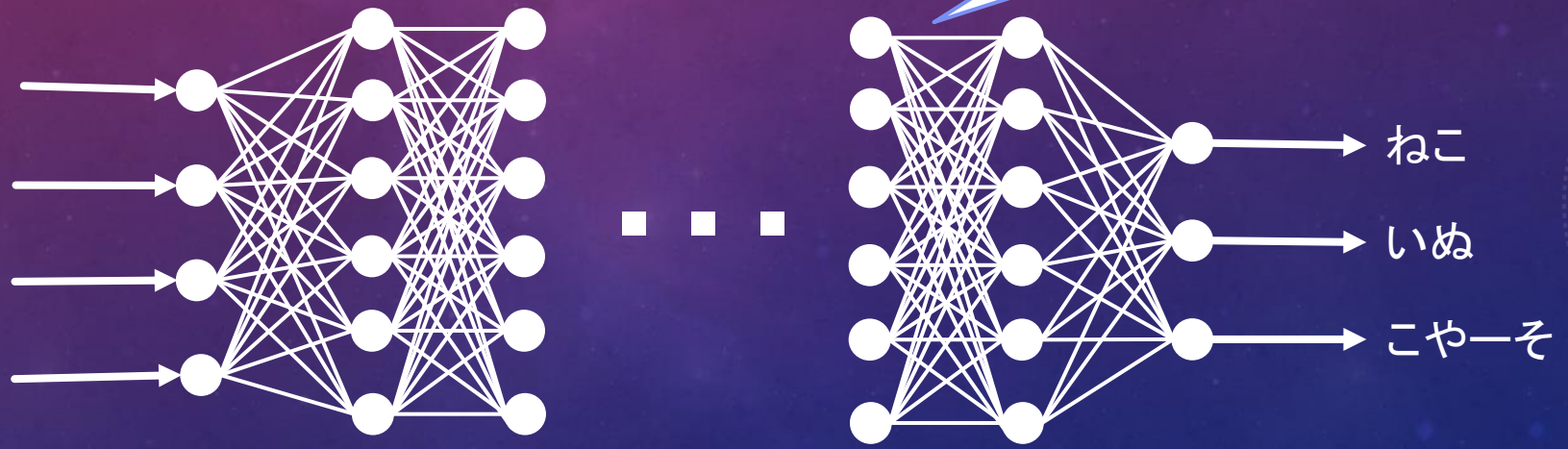
Q.だから何でだって言ってんだろ！

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

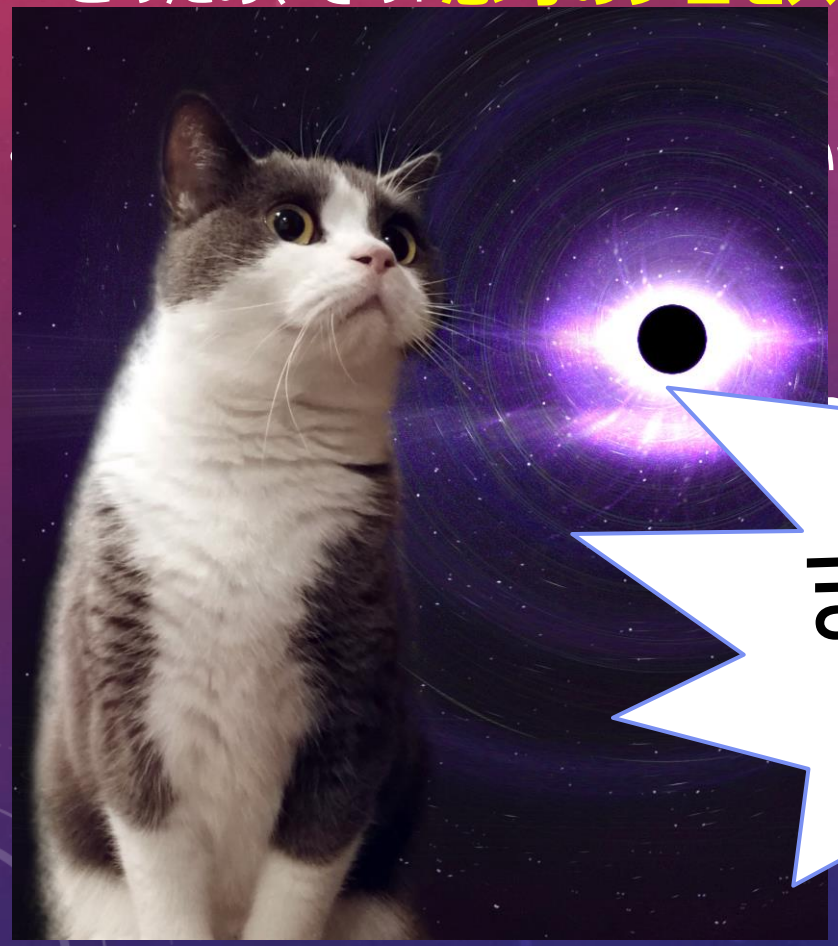
- ディープラーニングは、機械が自ら膨大なデータを学習し自律的に答えを導き出します。
- このため、その「**思考のプロセスが人間には分からない（ブラックボックス）**」という特性があります。
- もちろんそれを解き明かすXAIという研究もされていますが…

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

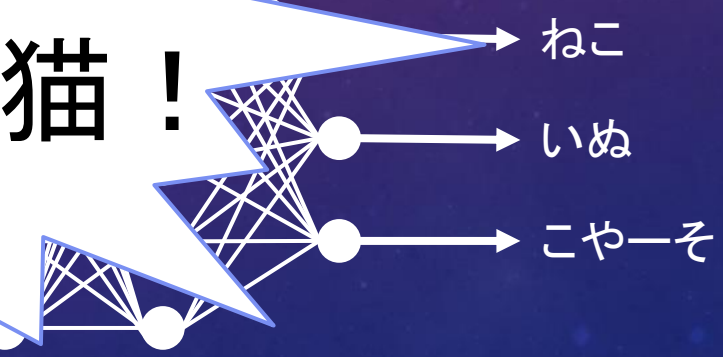
- ディープラーニングは、機械が自ら膨大なデータを学習し自律的に答えを導き出します。
- このため、その「**思考のプロセスが人間には分からない（ブラックボックス）**」



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！



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

- 便利なWebツール

TensorFlow Neural Network Playg

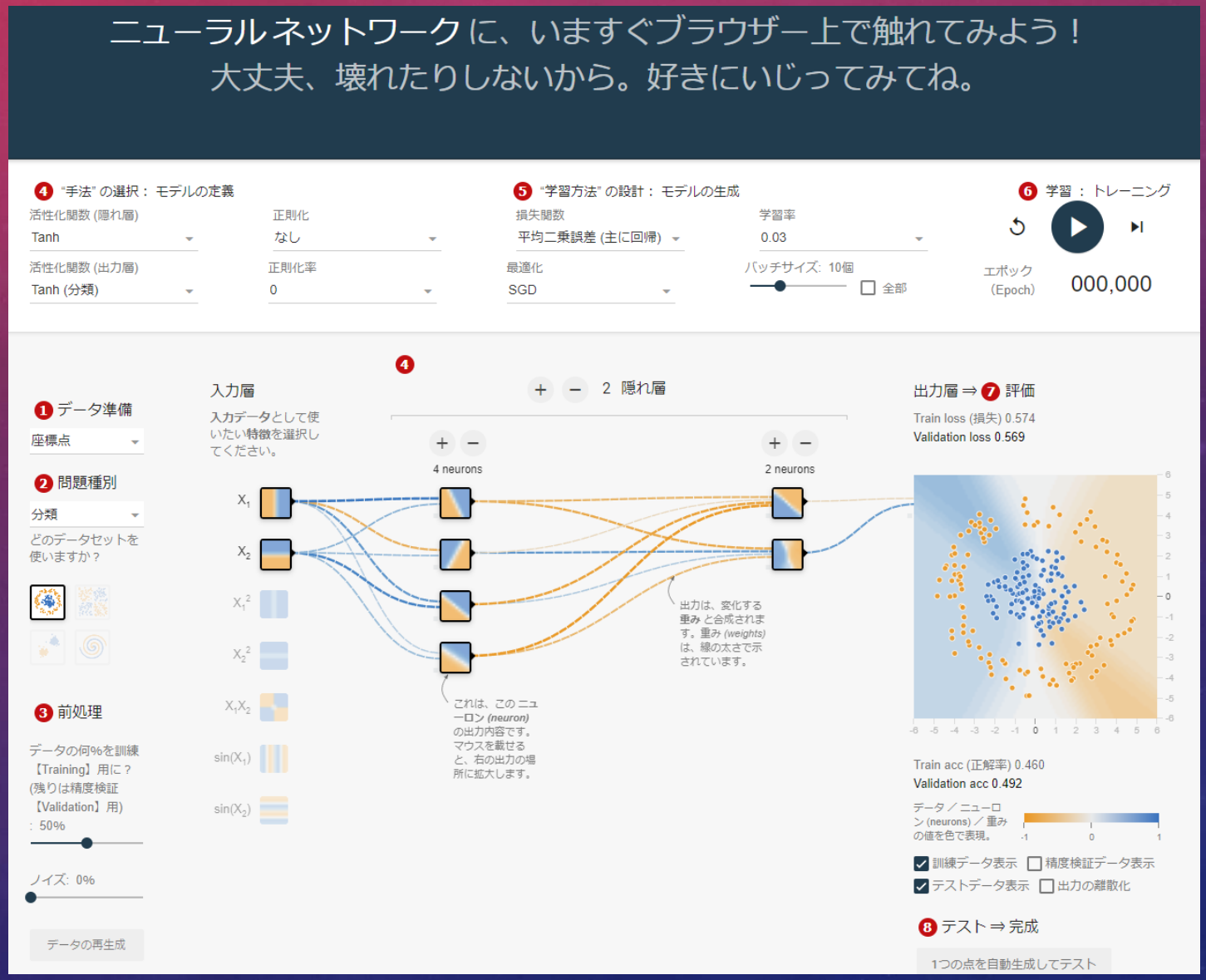
Googleが提供している学習用ツール

Deepinsider版はバグがあるので慣れたら英語版を使おう

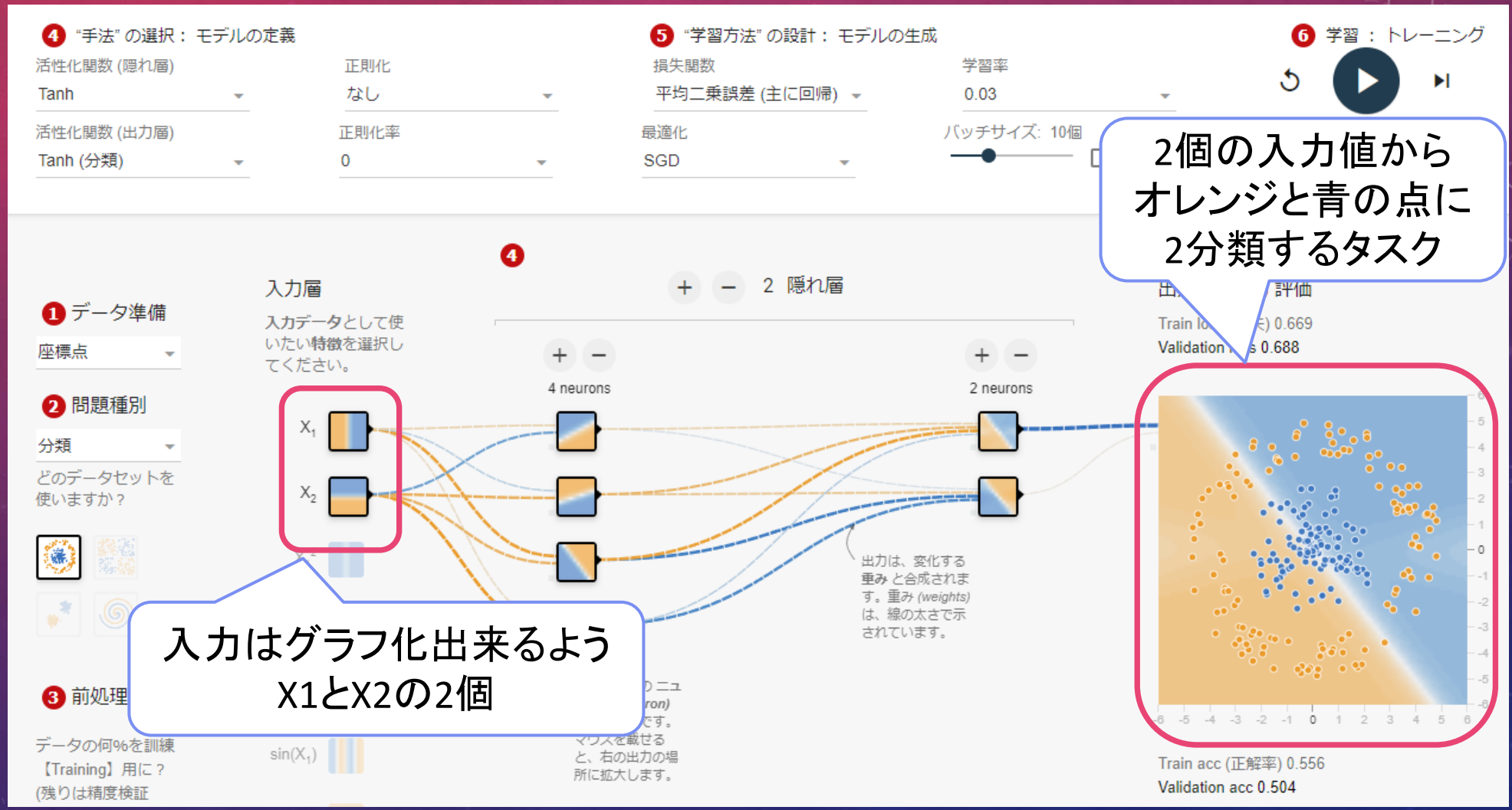
英語版の本家
https://playground.tensorflow.org/

日本語版
（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/

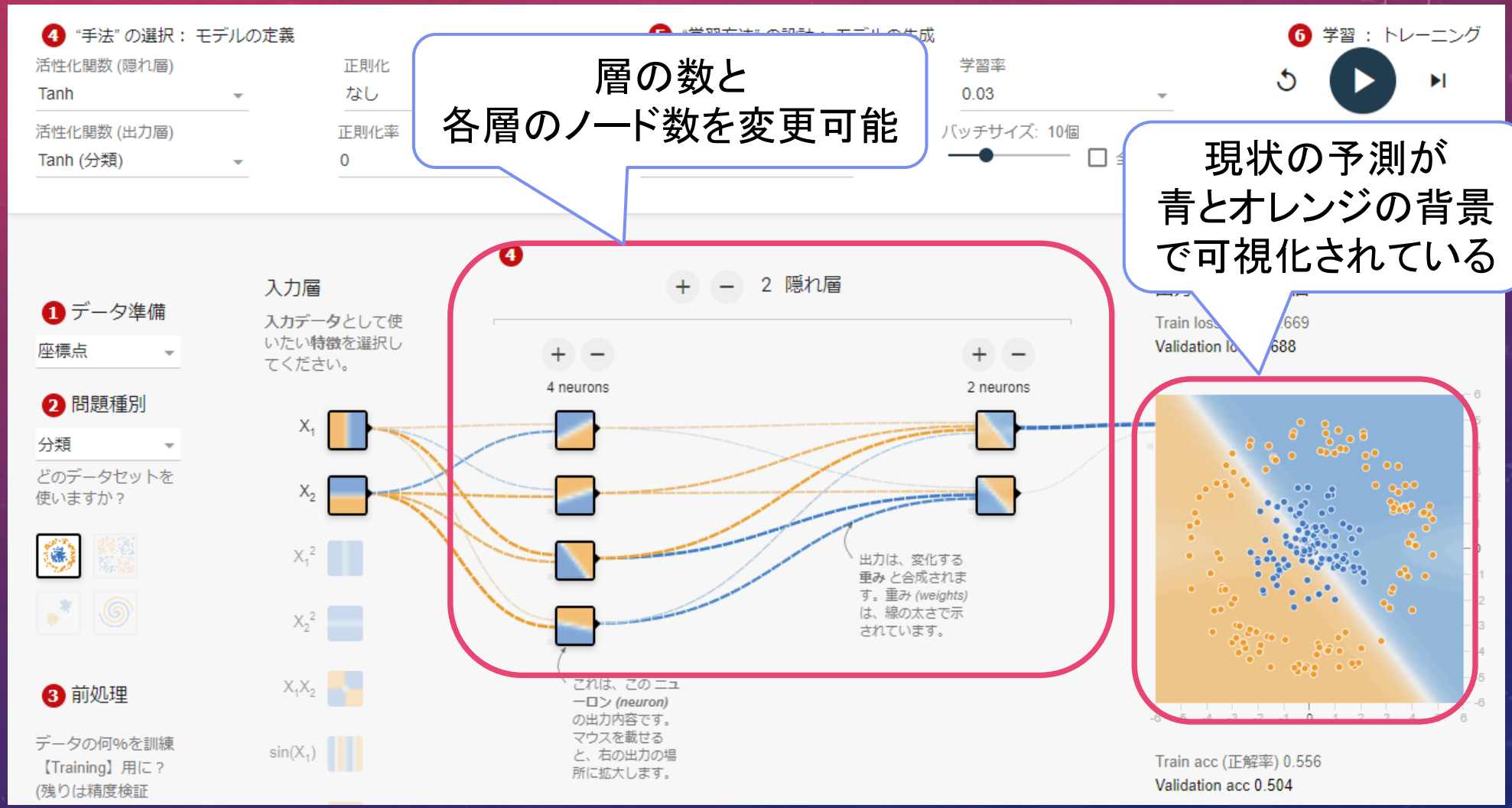
「A Neural Network Playground」



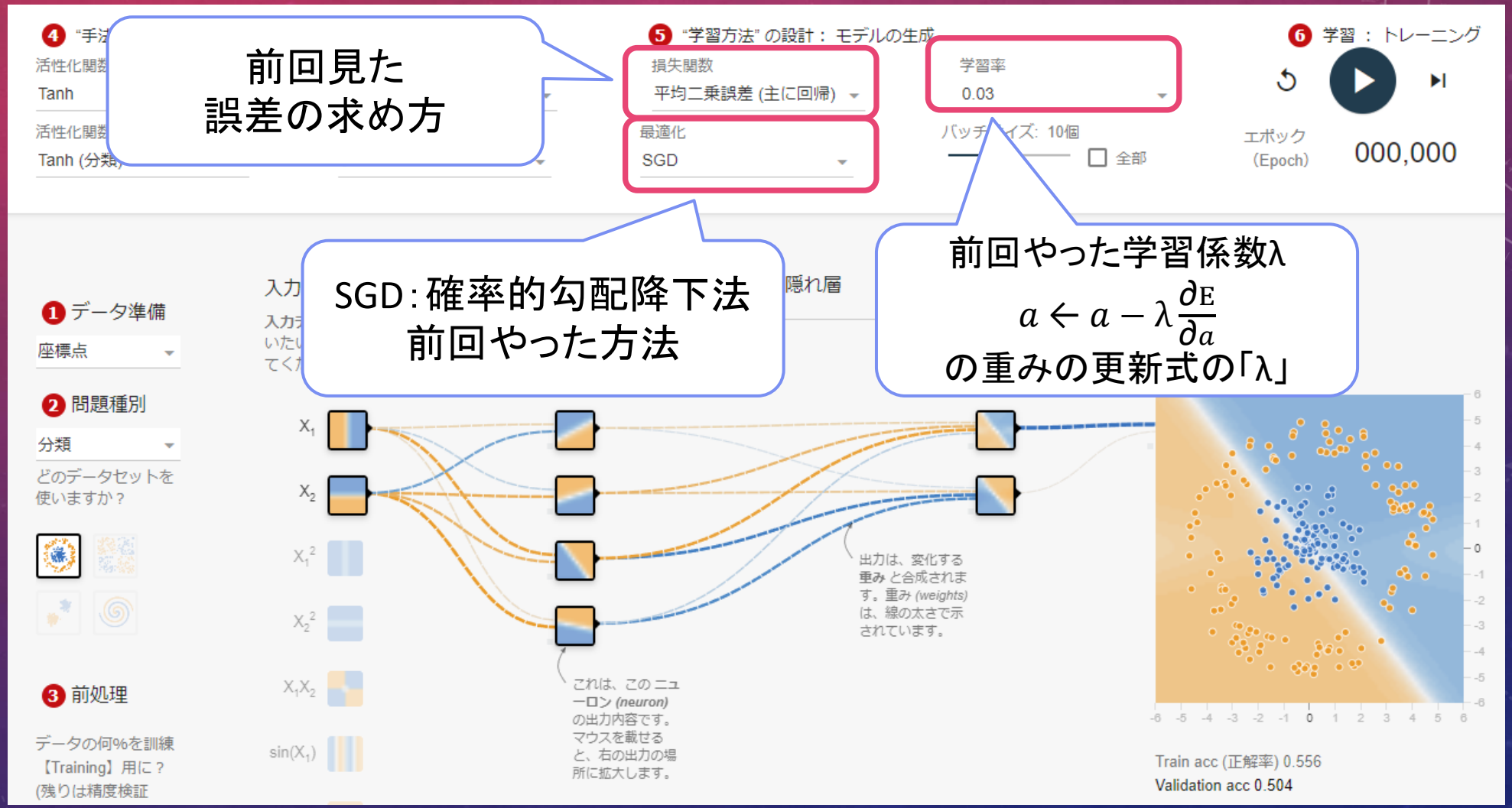
「A Neural Network Playground」使い方



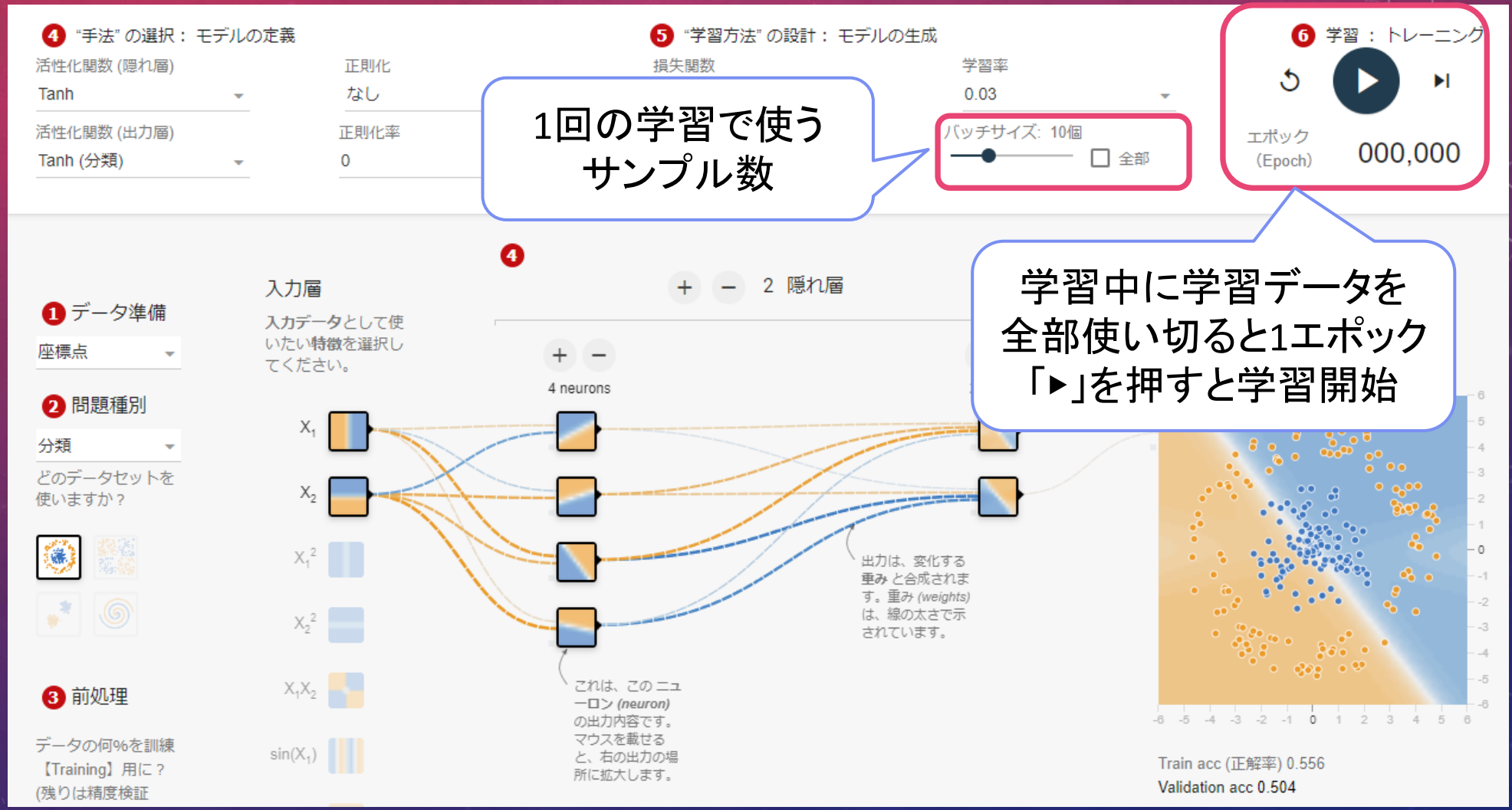
「A Neural Network Playground」使い方



「A Neural Network Playground」使い方

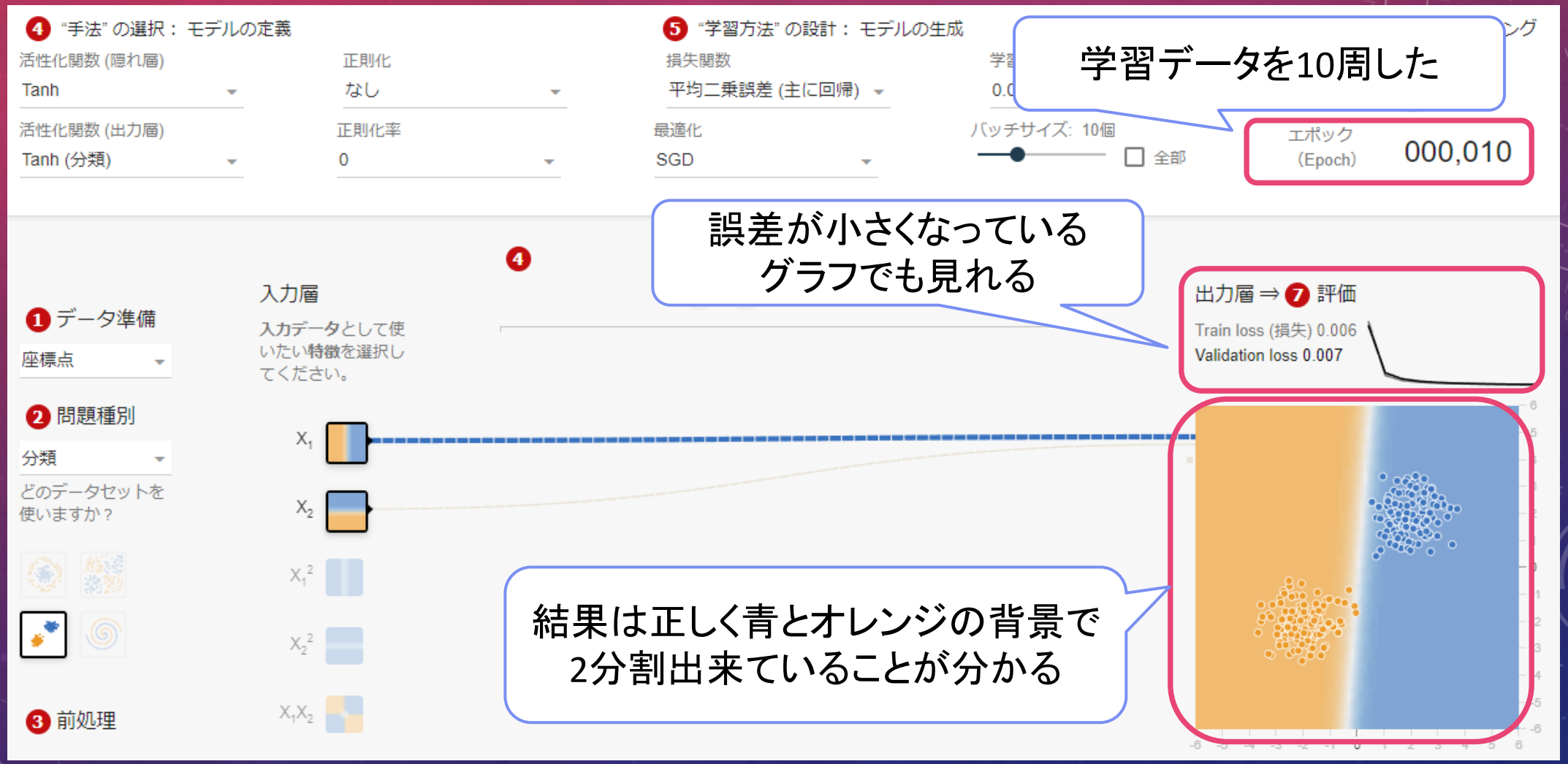


「A Neural Network Playground」使い方

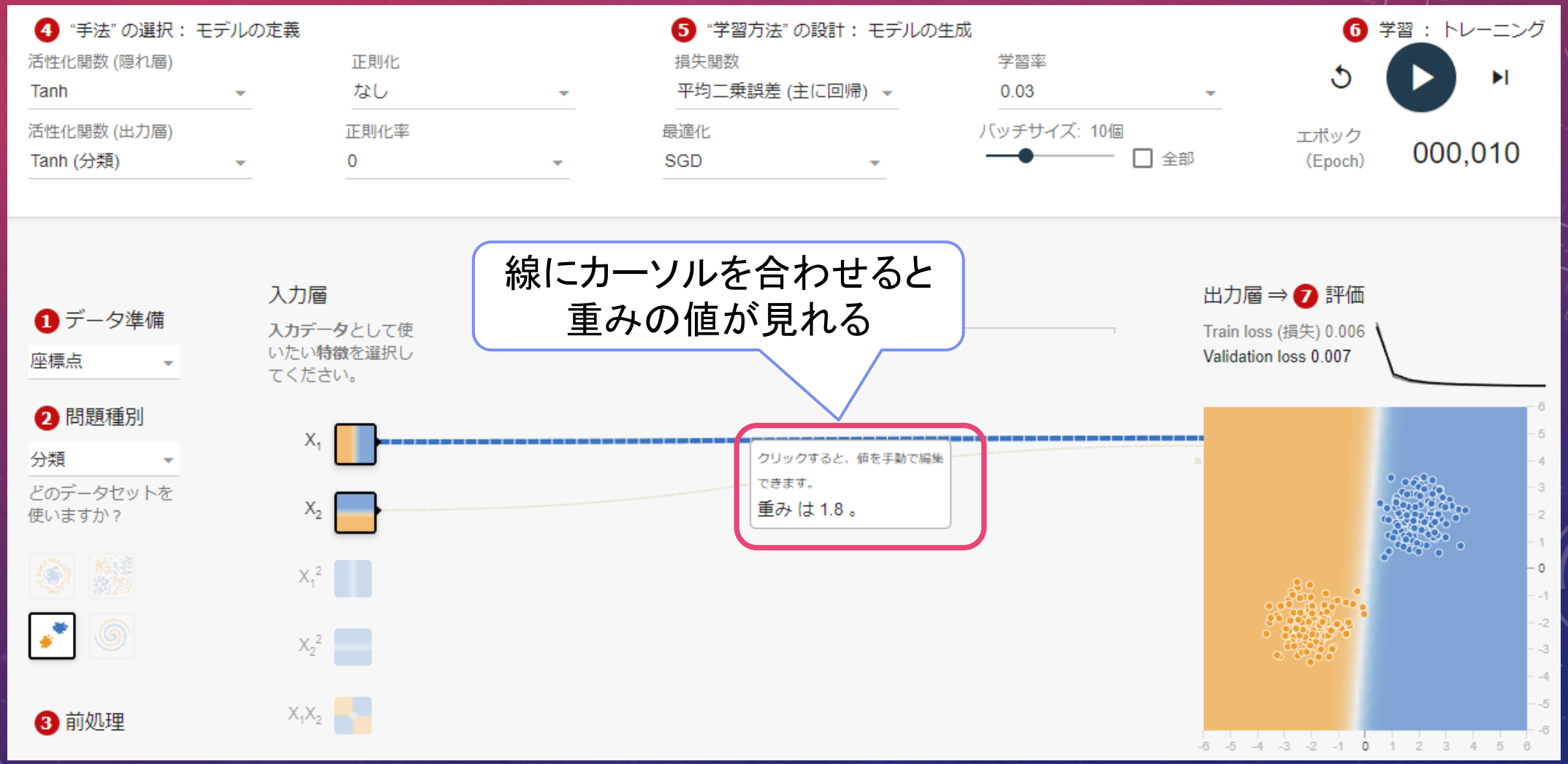


隠れ層0、ノード1個でパーセプトロン：2分類問題

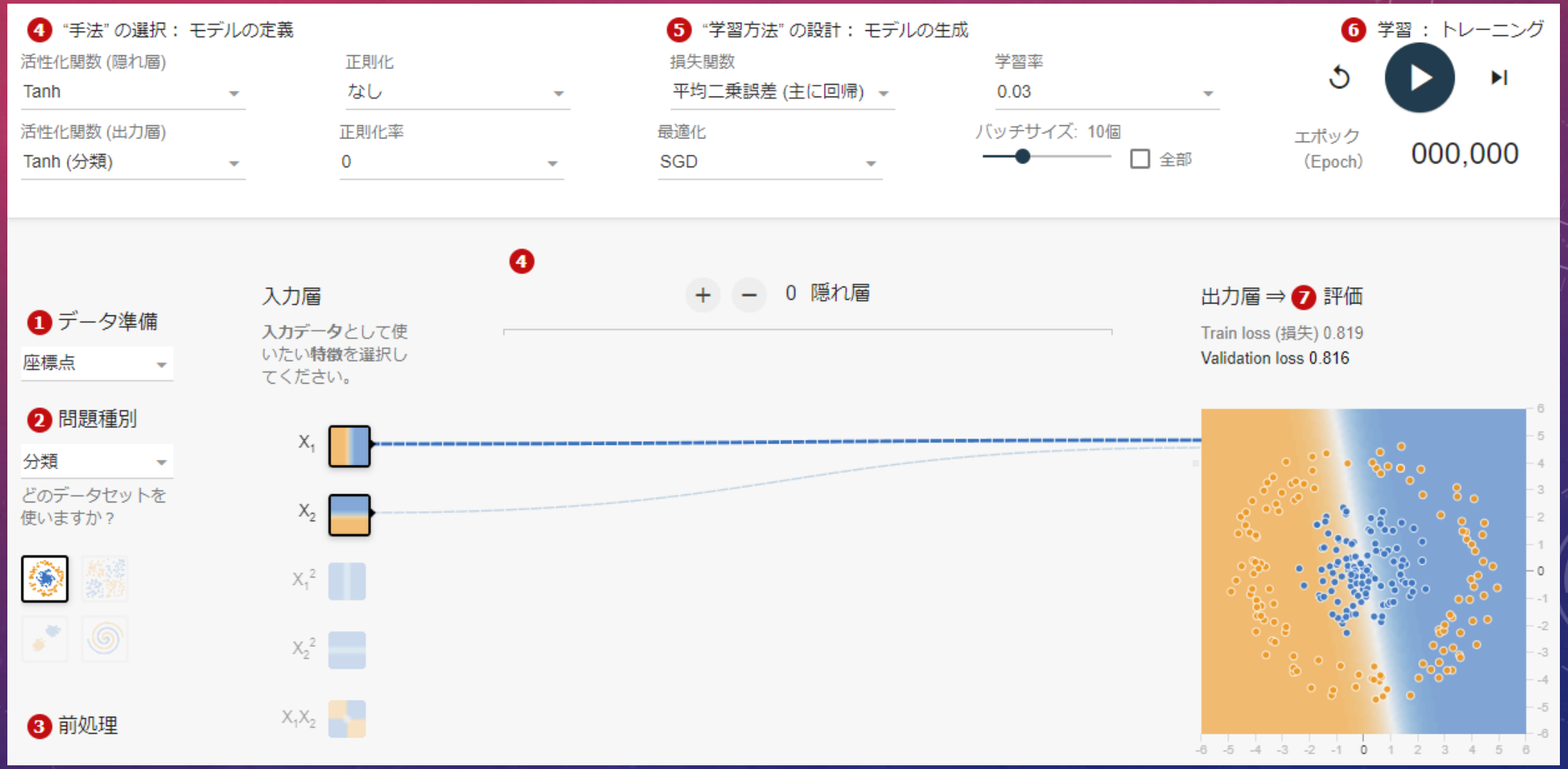
隠れ層0、ノード1個でパーセプトロン：2分類問題



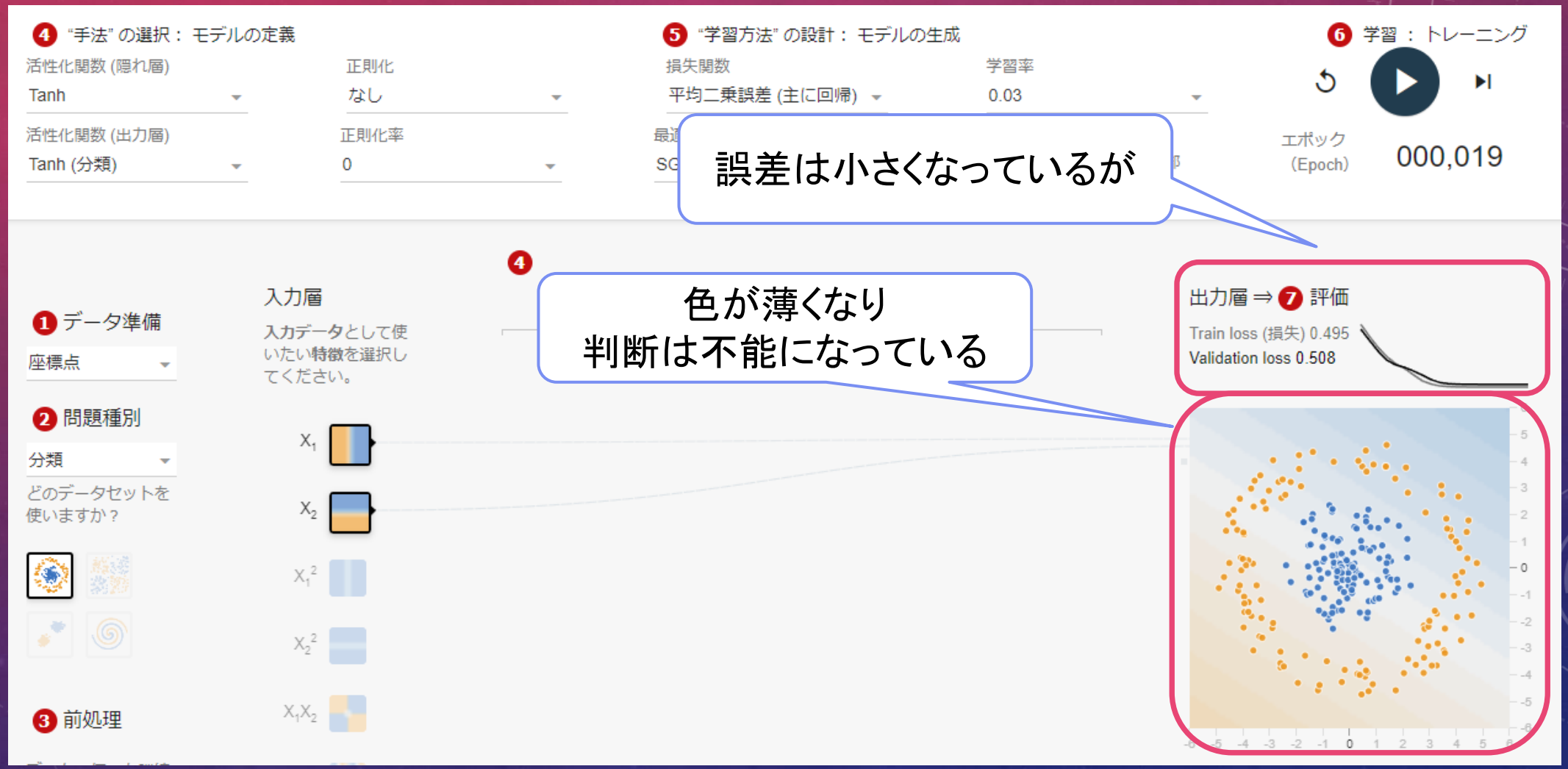
隠れ層0、ノード1個でパーセプトロン：2分類問題



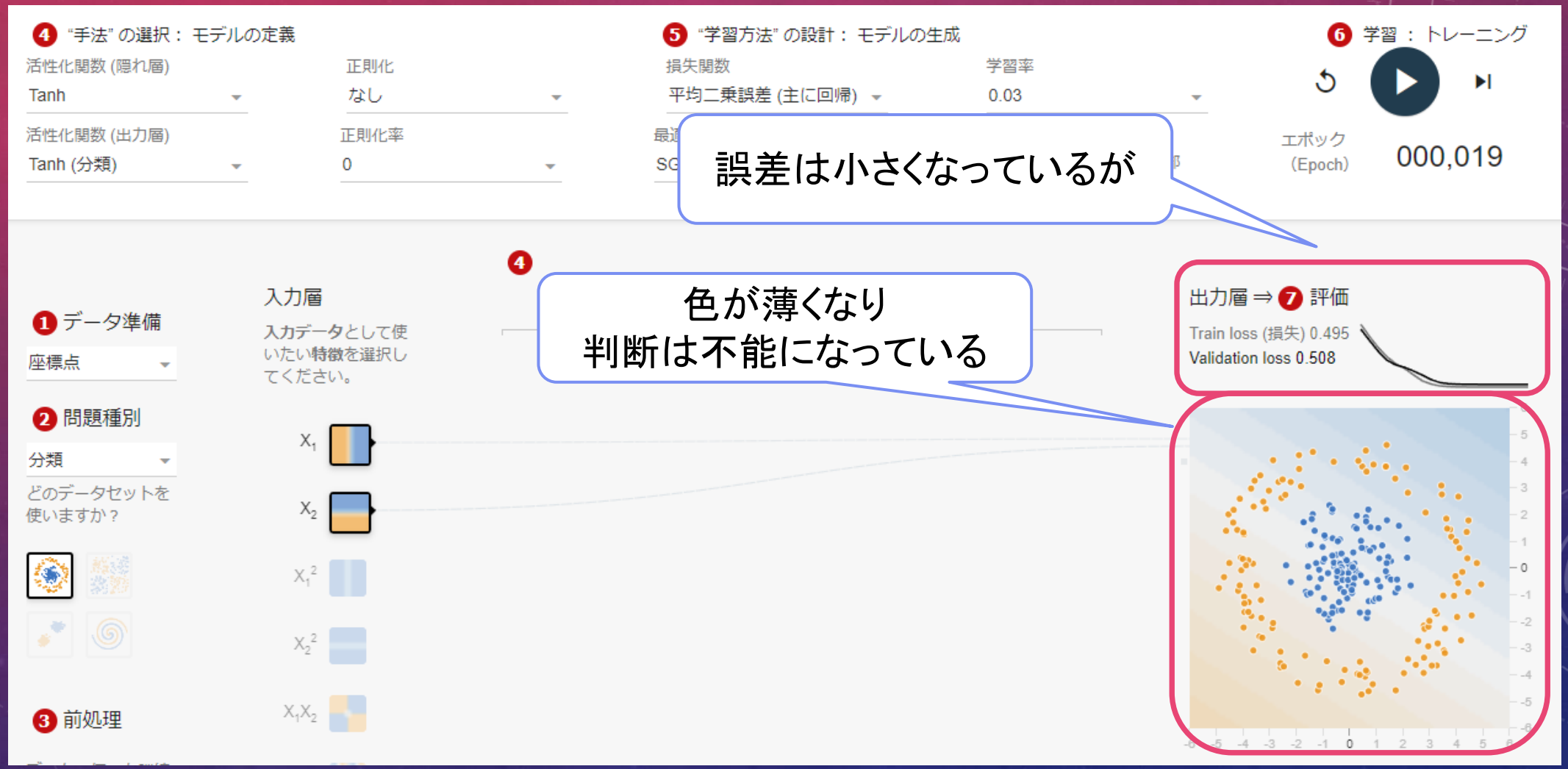
隠れ層0、ノード1個でパーセプトロン：2分類問題その2



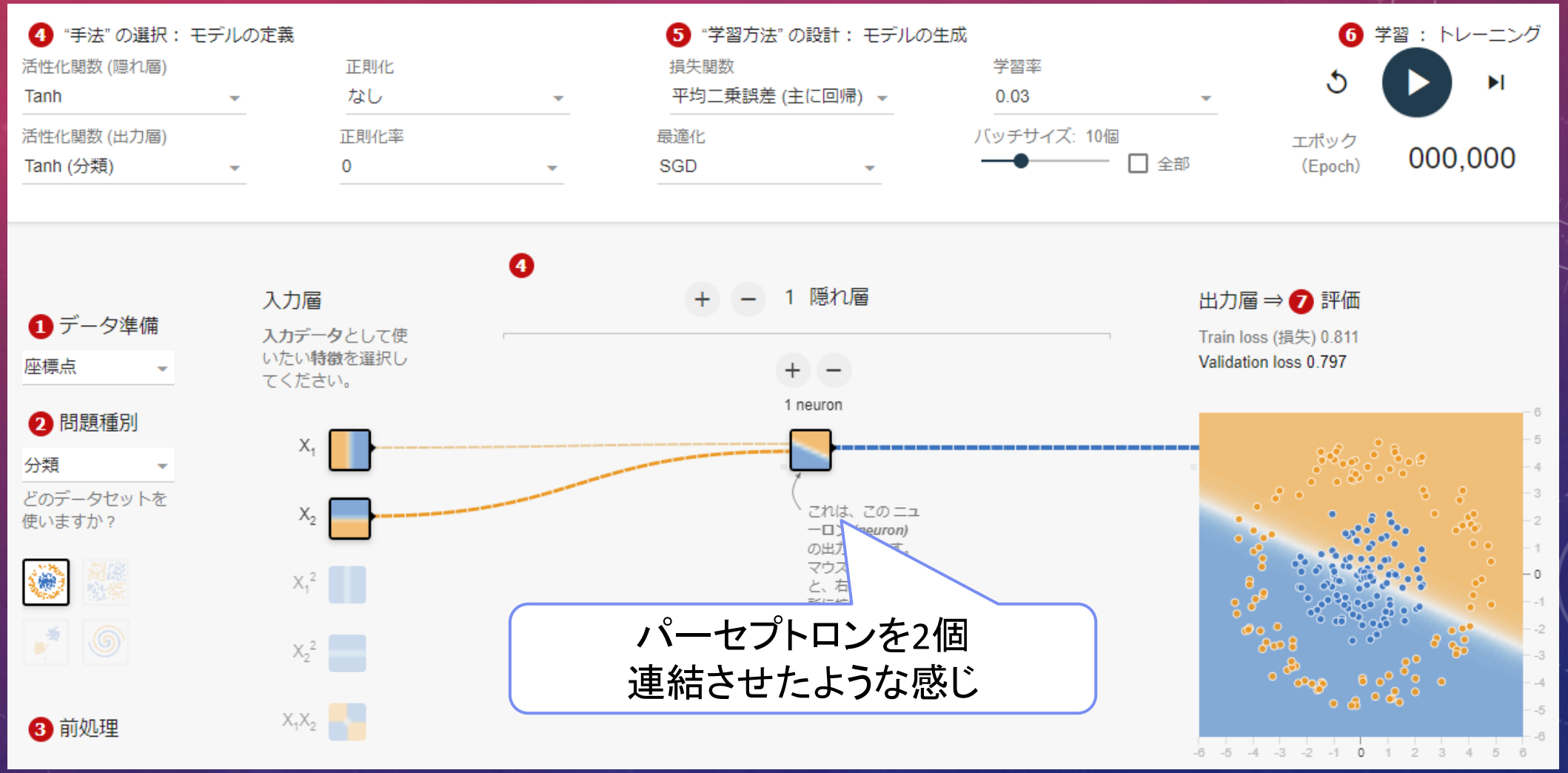
隠れ層0、ノード1個でパーセプトロン：2分類問題その2



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

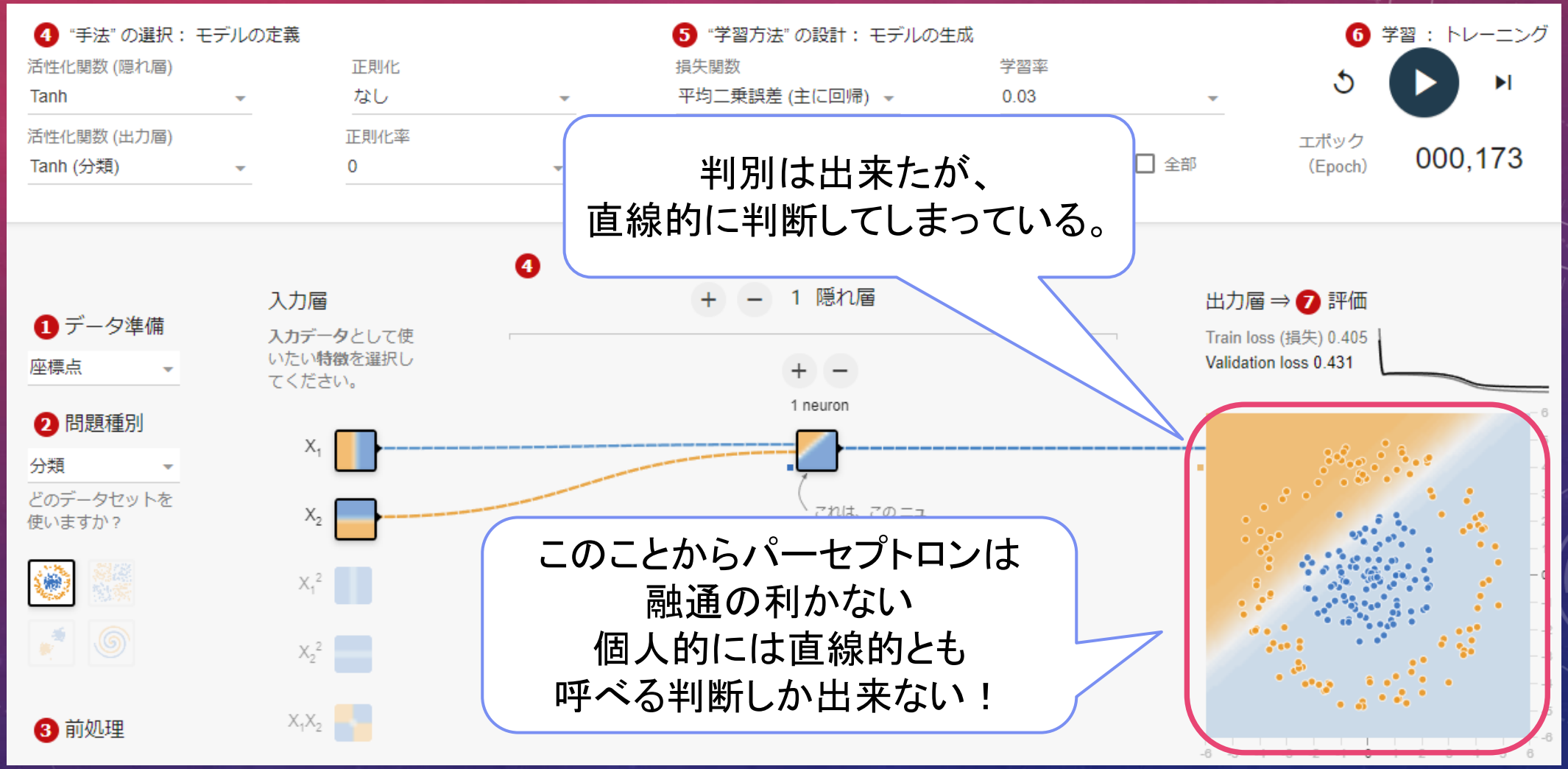


隠れ層1個、ノード1個の場合：2分類問題その2

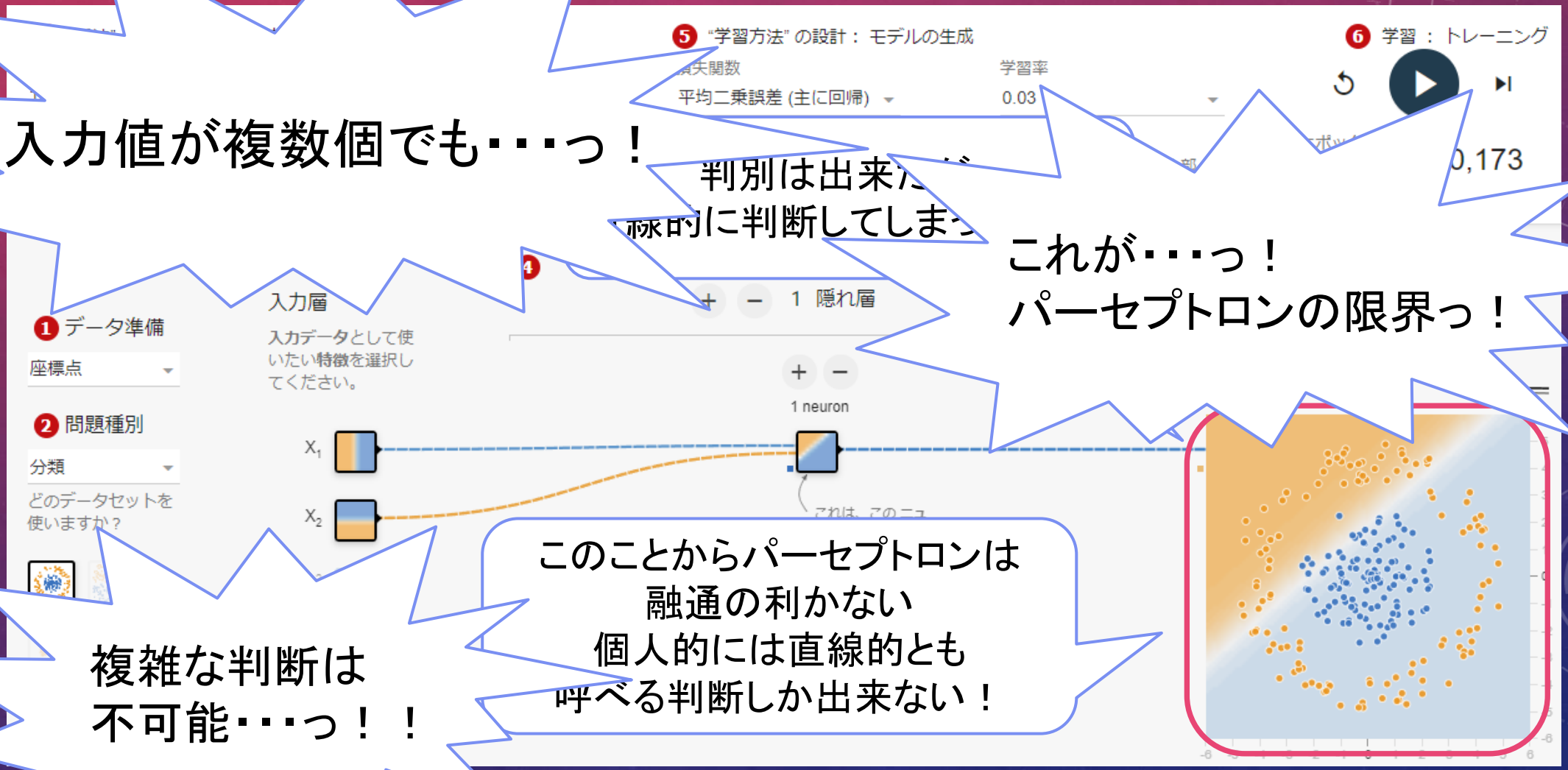


隠れ層1個、ノード1個の場合：2分類問題その2

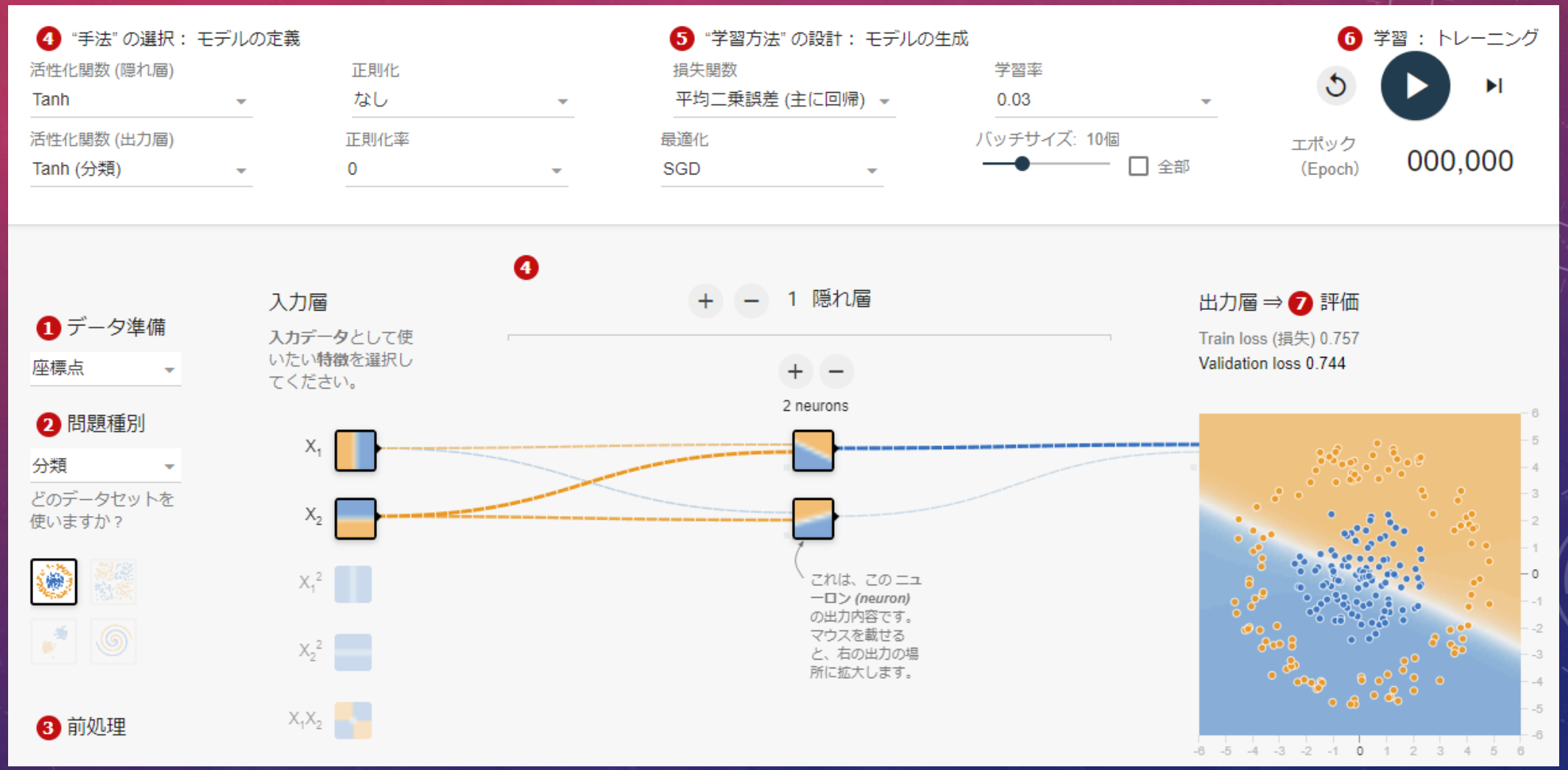
隠れ層1個、ノード1個の場合：2分類問題その2



隠れ層1個、ノード1個の場合：2分類問題その2

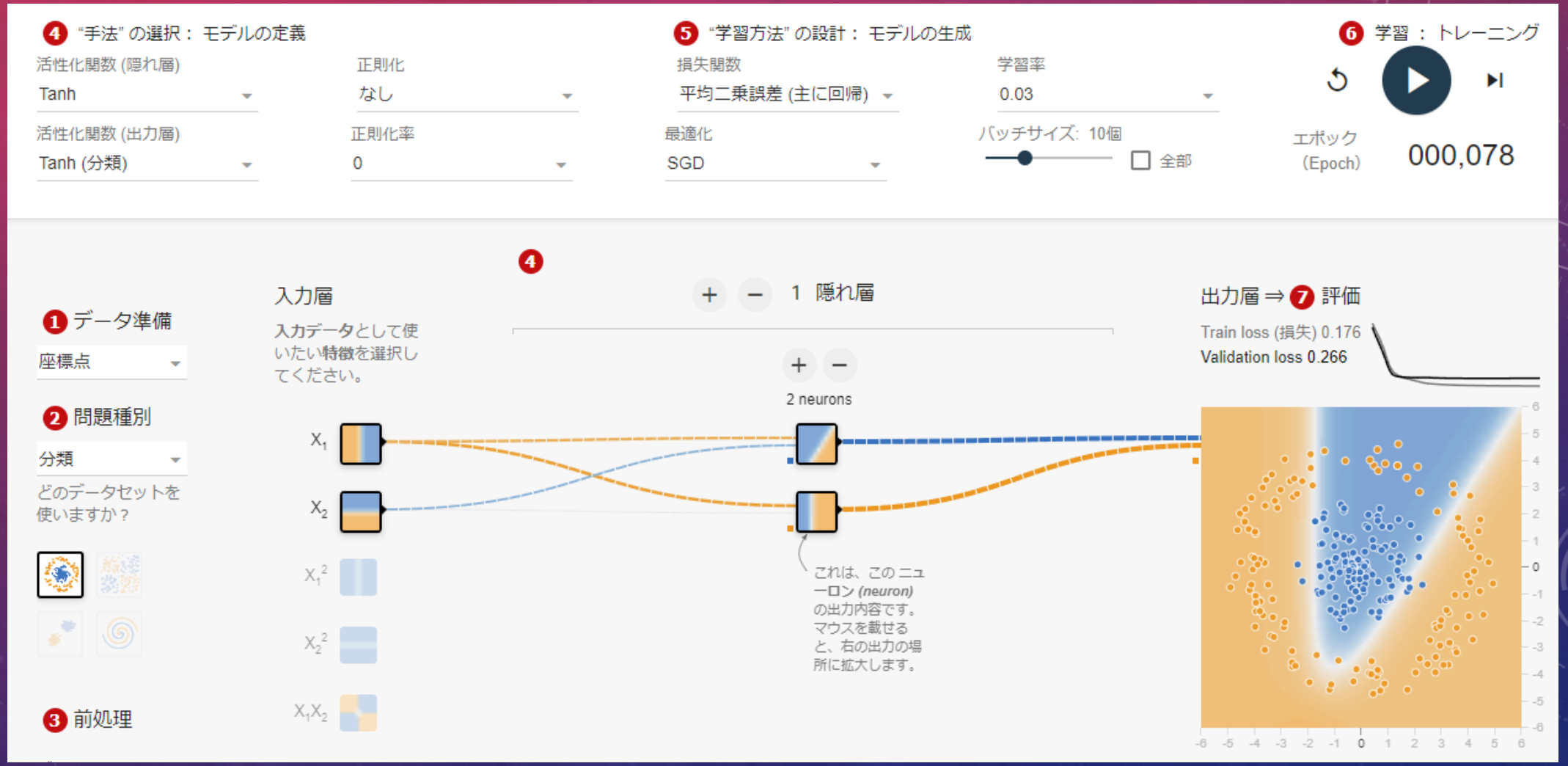


隠れ層1、ノード2個では？　：2分類問題その2

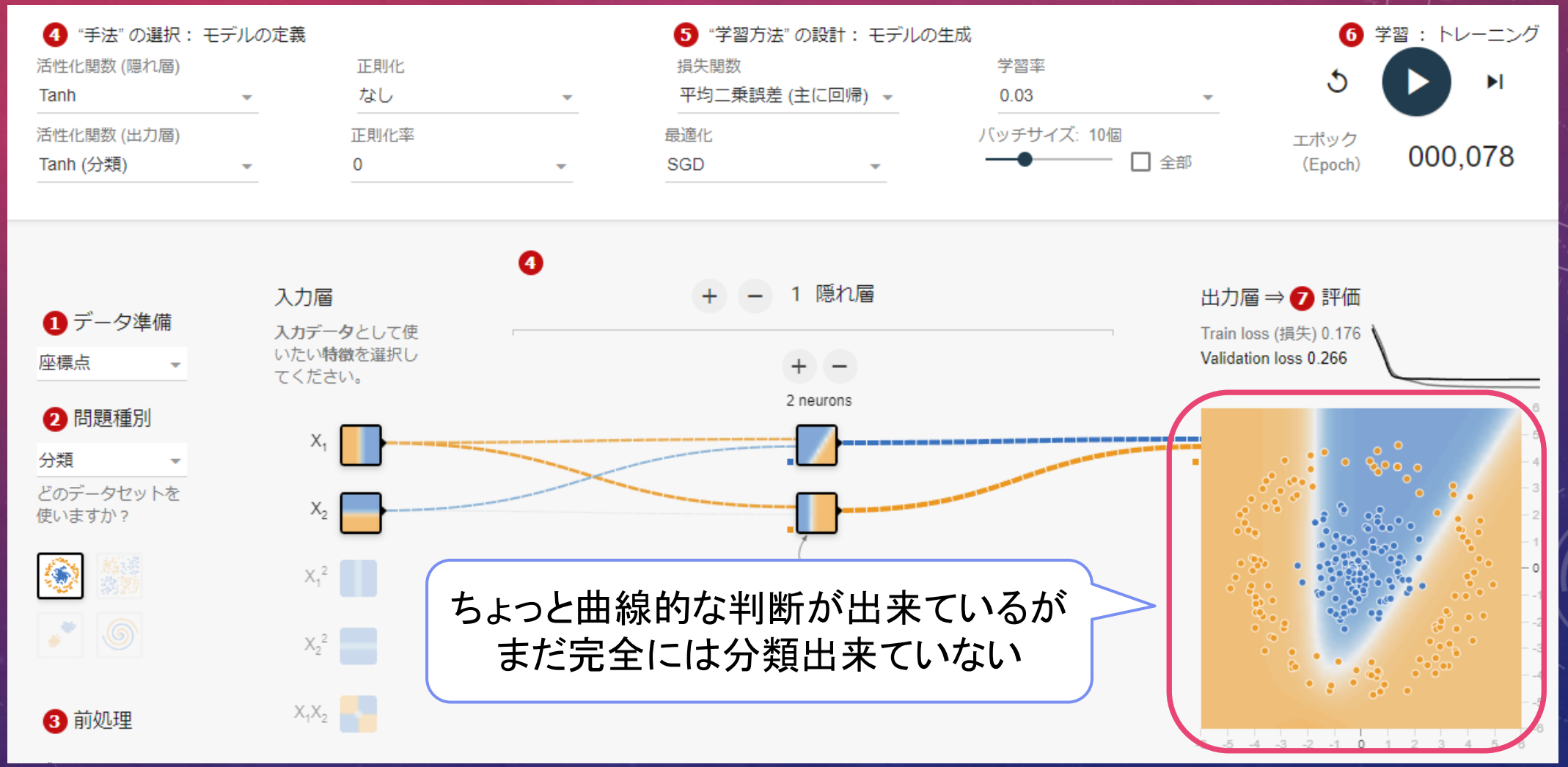


隠れ層1、ノード2個では？　：2分類問題その2

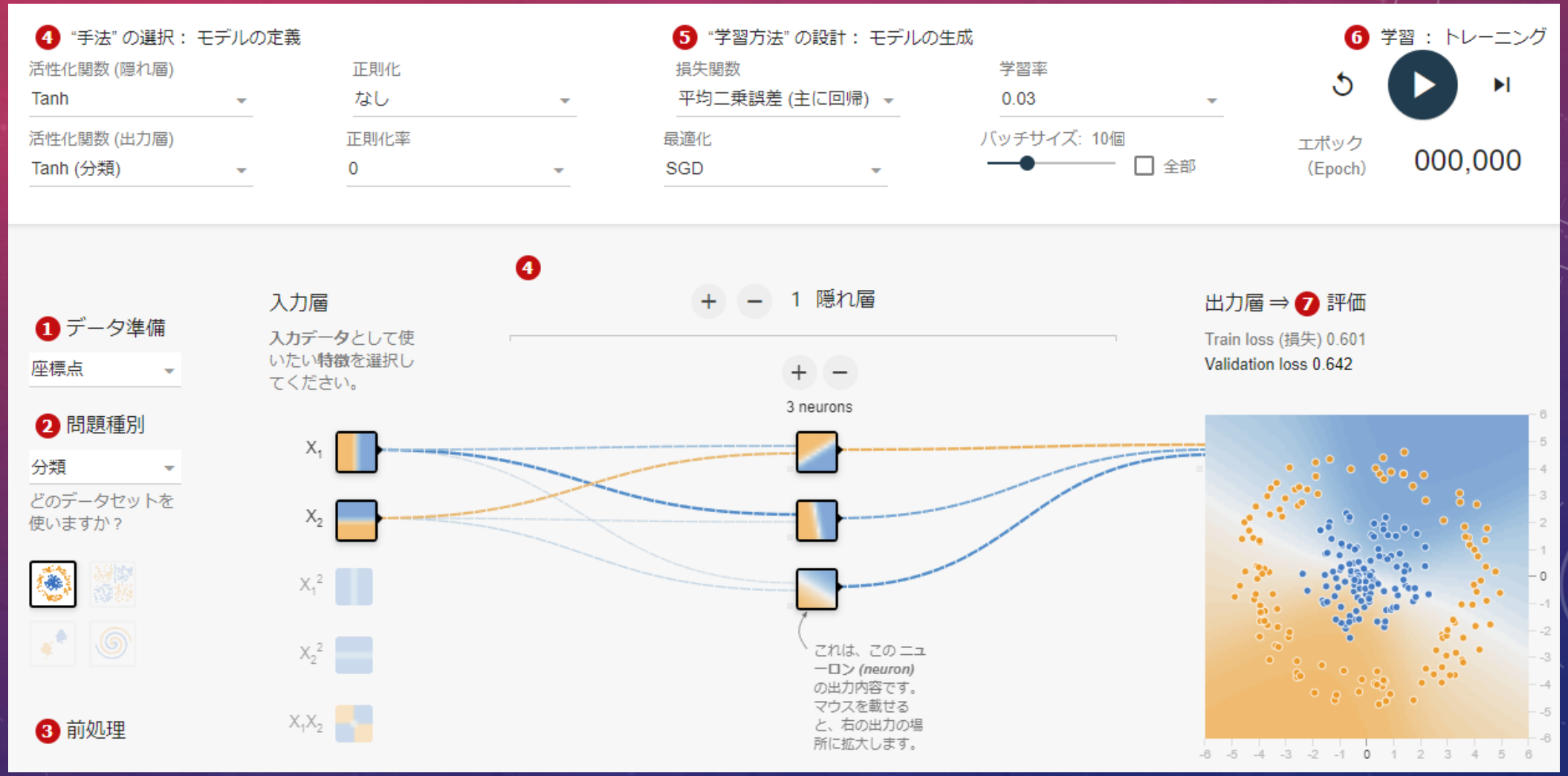
隠れ層1、ノード2個では？　：2分類問題その2



隠れ層1、ノード2個では？　：2分類問題その2



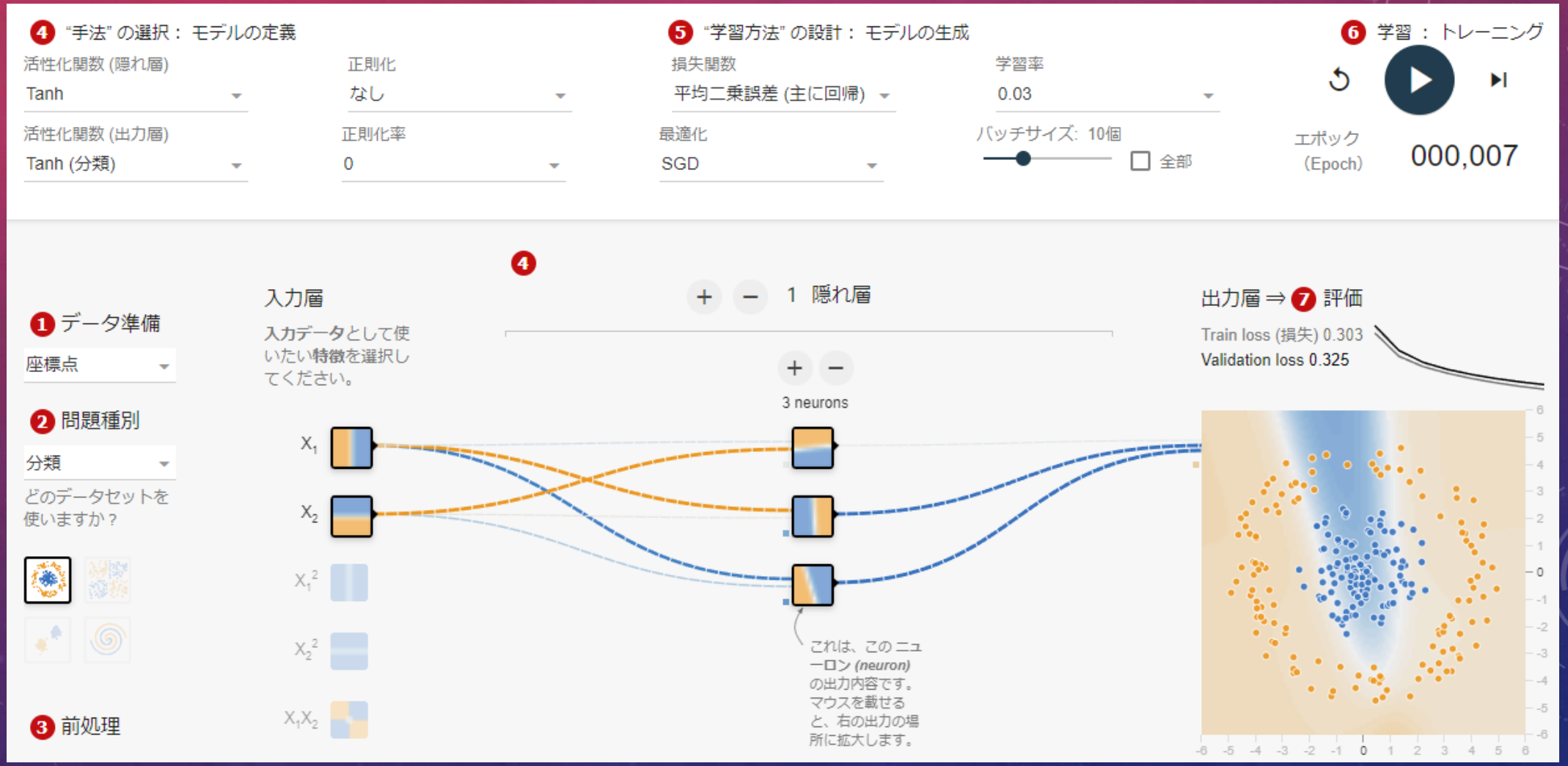
隠れ層1、ノード3個にしてみる：2分類問題その2



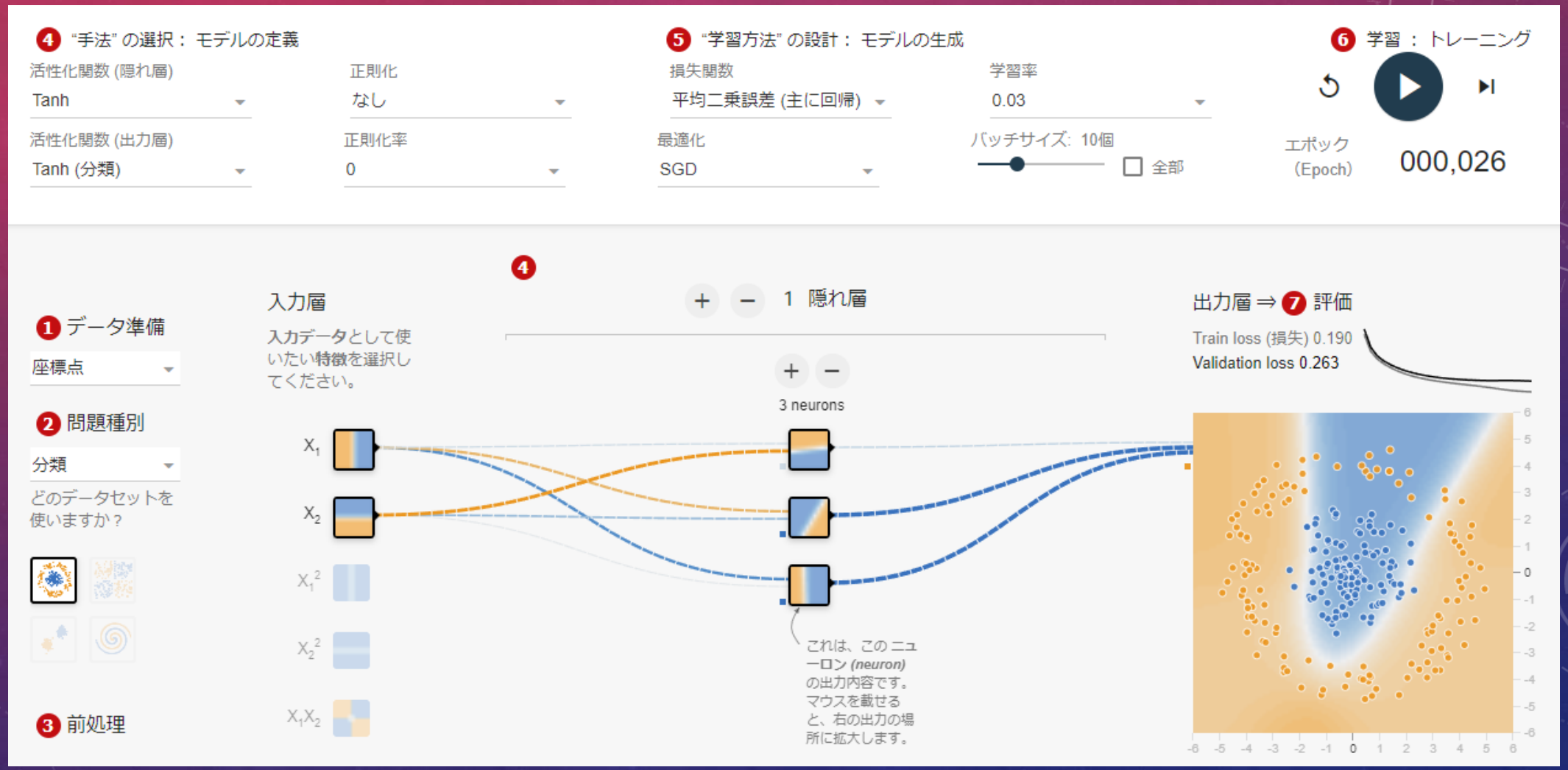
隠れ層1、ノード3個にしてみる：2分類問題その2

今度はノードを3個だって？
このいやしんぼめ！

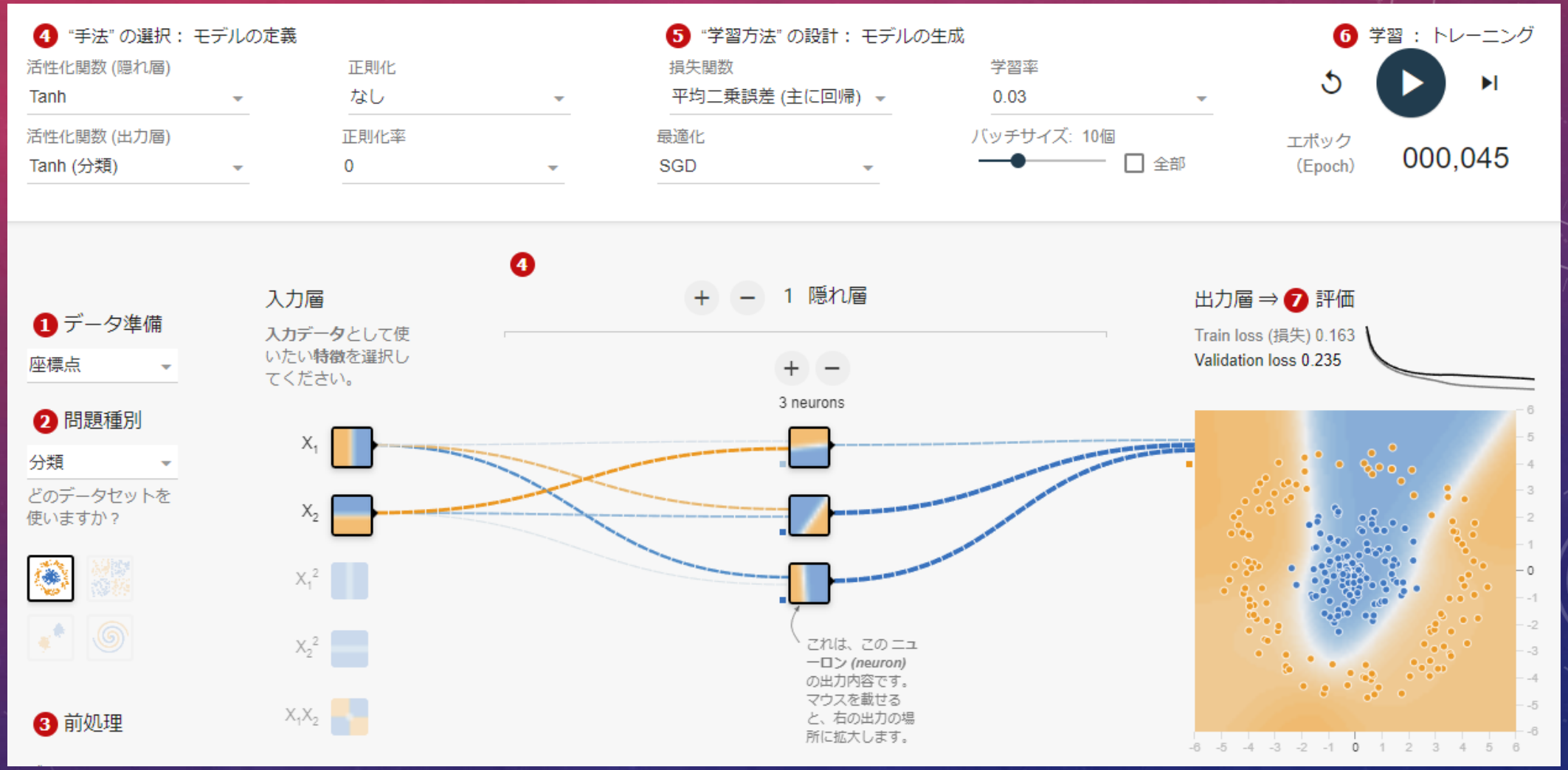
隠れ層1、ノード3個にしてみる：2分類問題その2



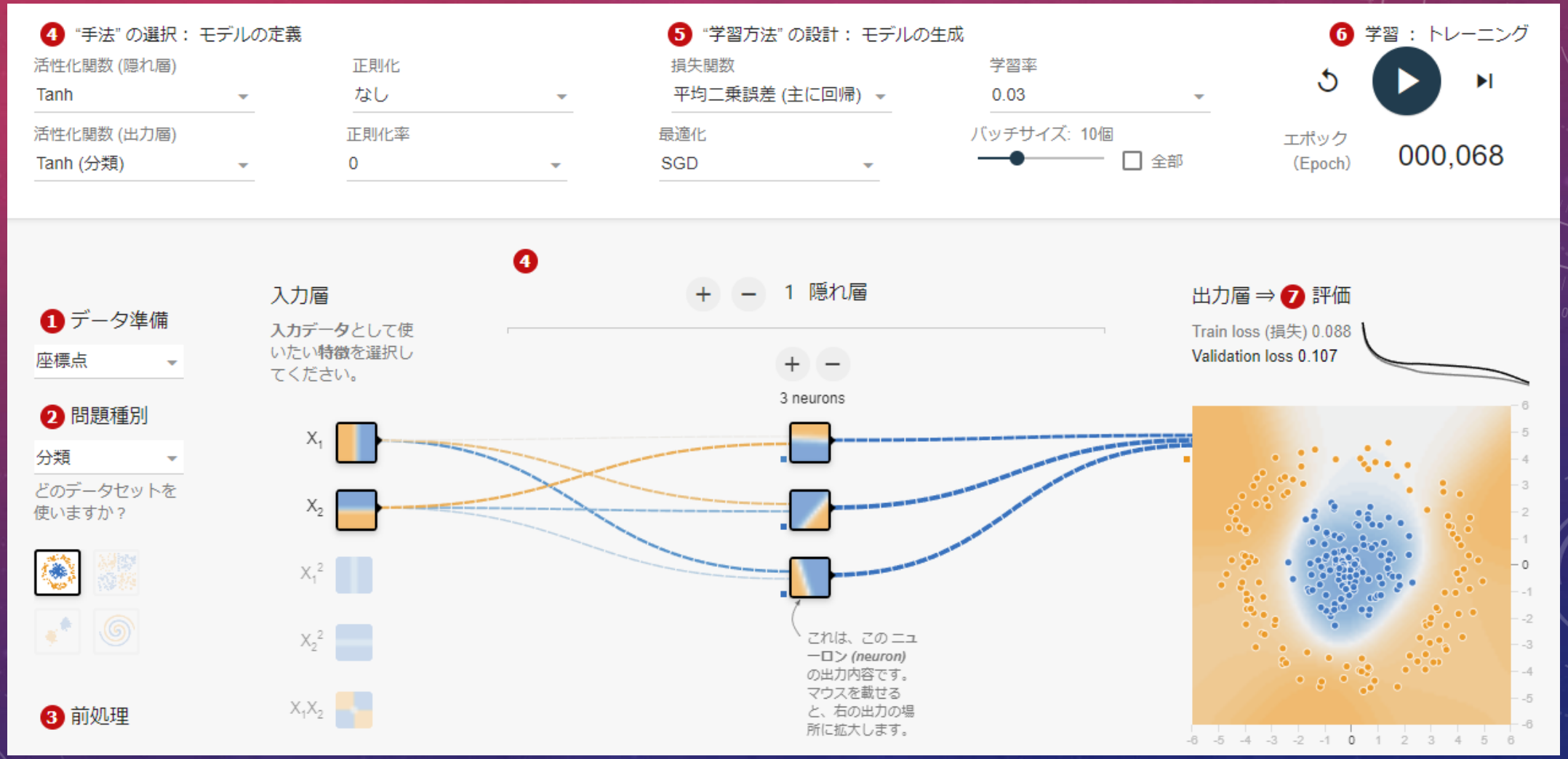
隠れ層1、ノード3個にしてみる：2分類問題その2



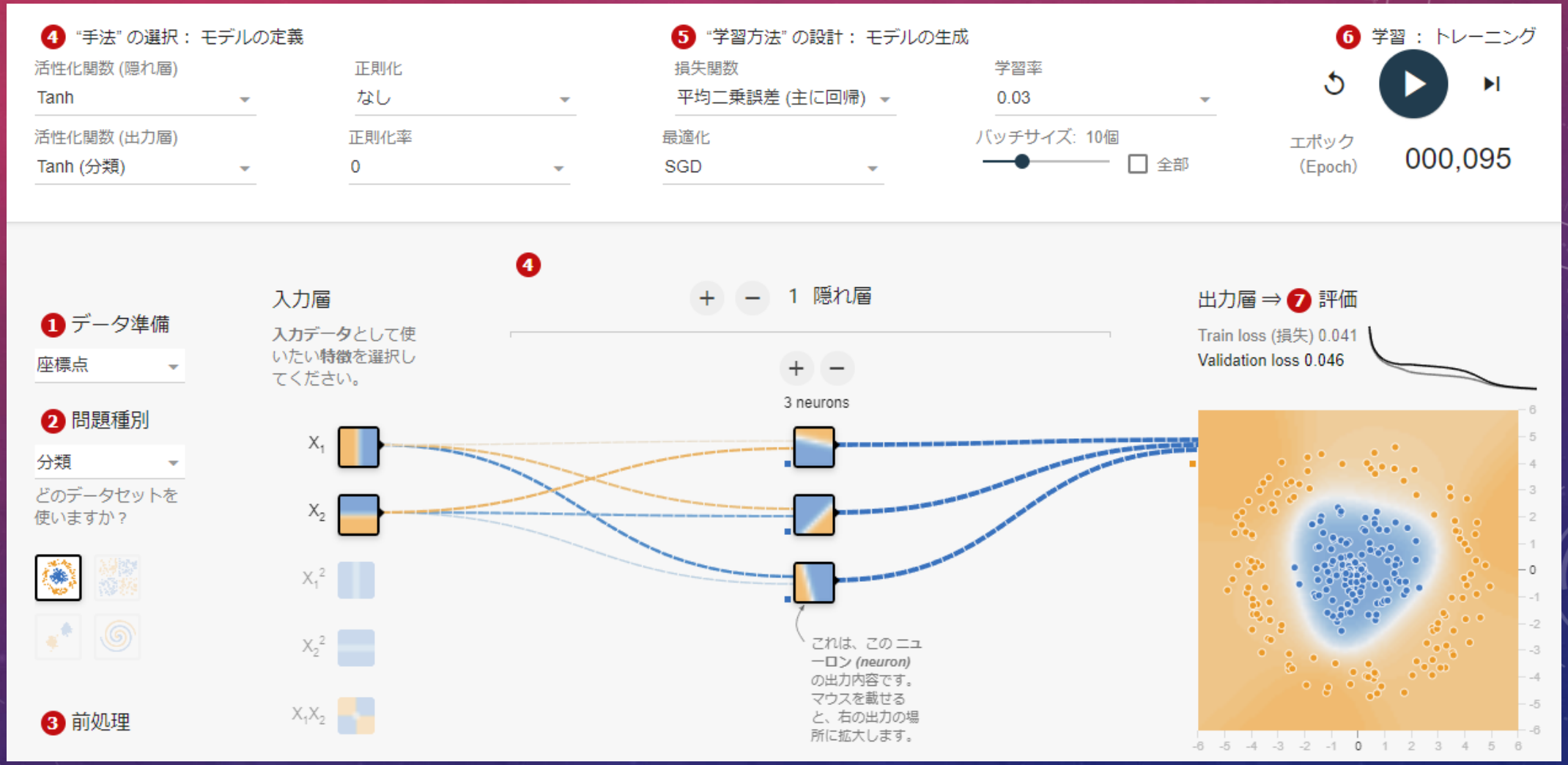
隠れ層1、ノード3個にしてみる：2分類問題その2



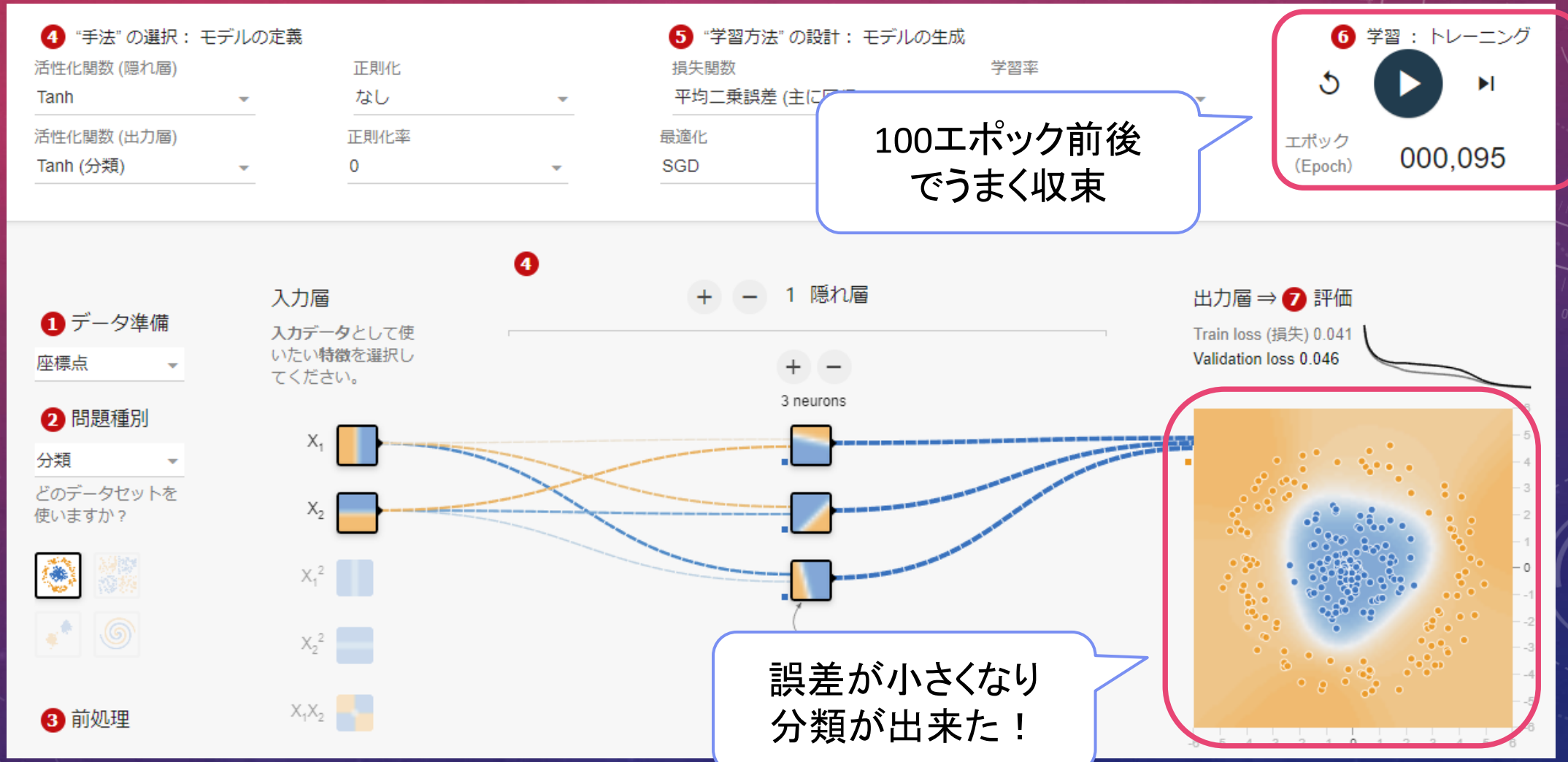
隠れ層1、ノード3個にしてみる：2分類問題その2



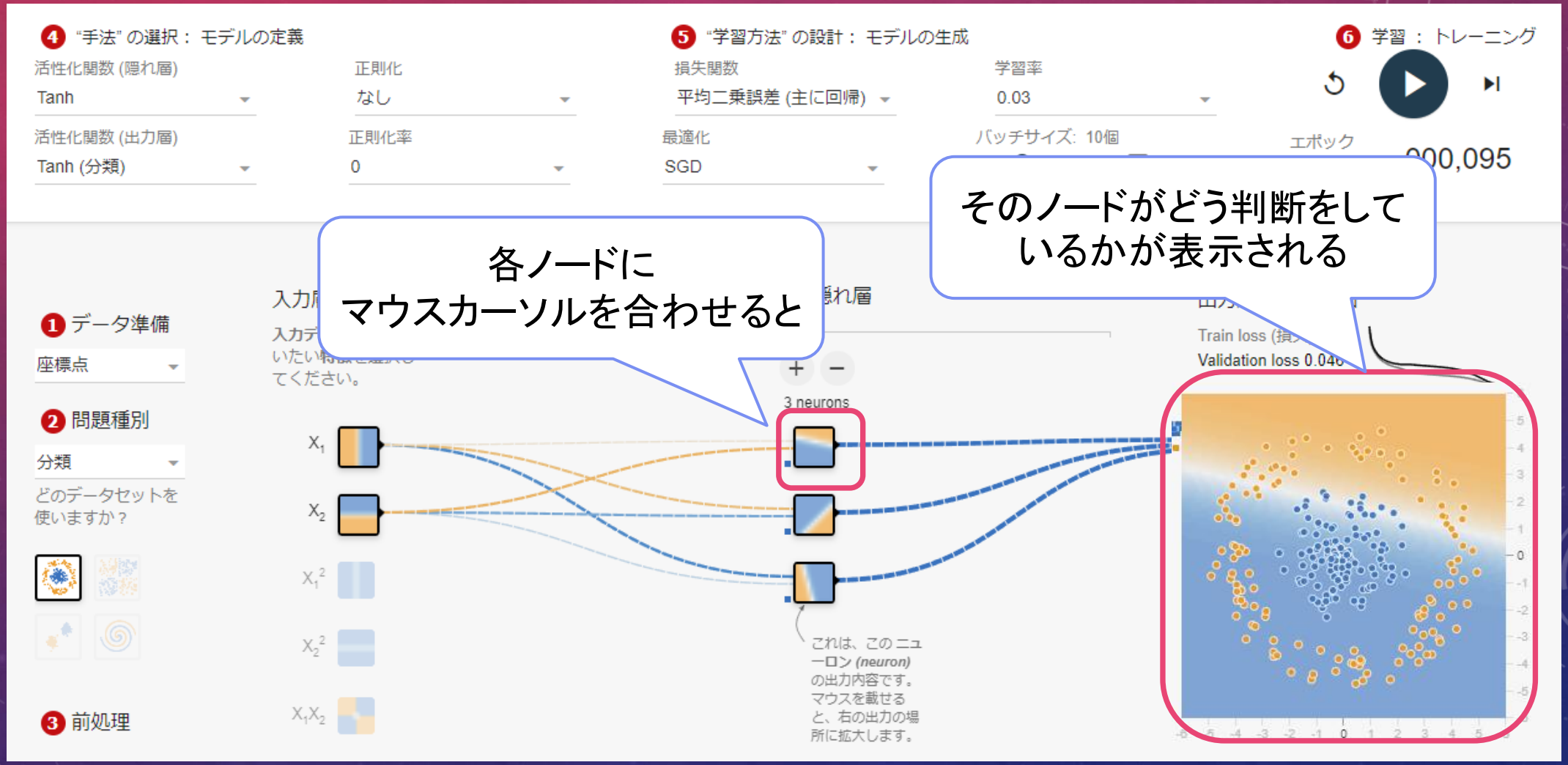
隠れ層1、ノード3個にしてみる：2分類問題その2



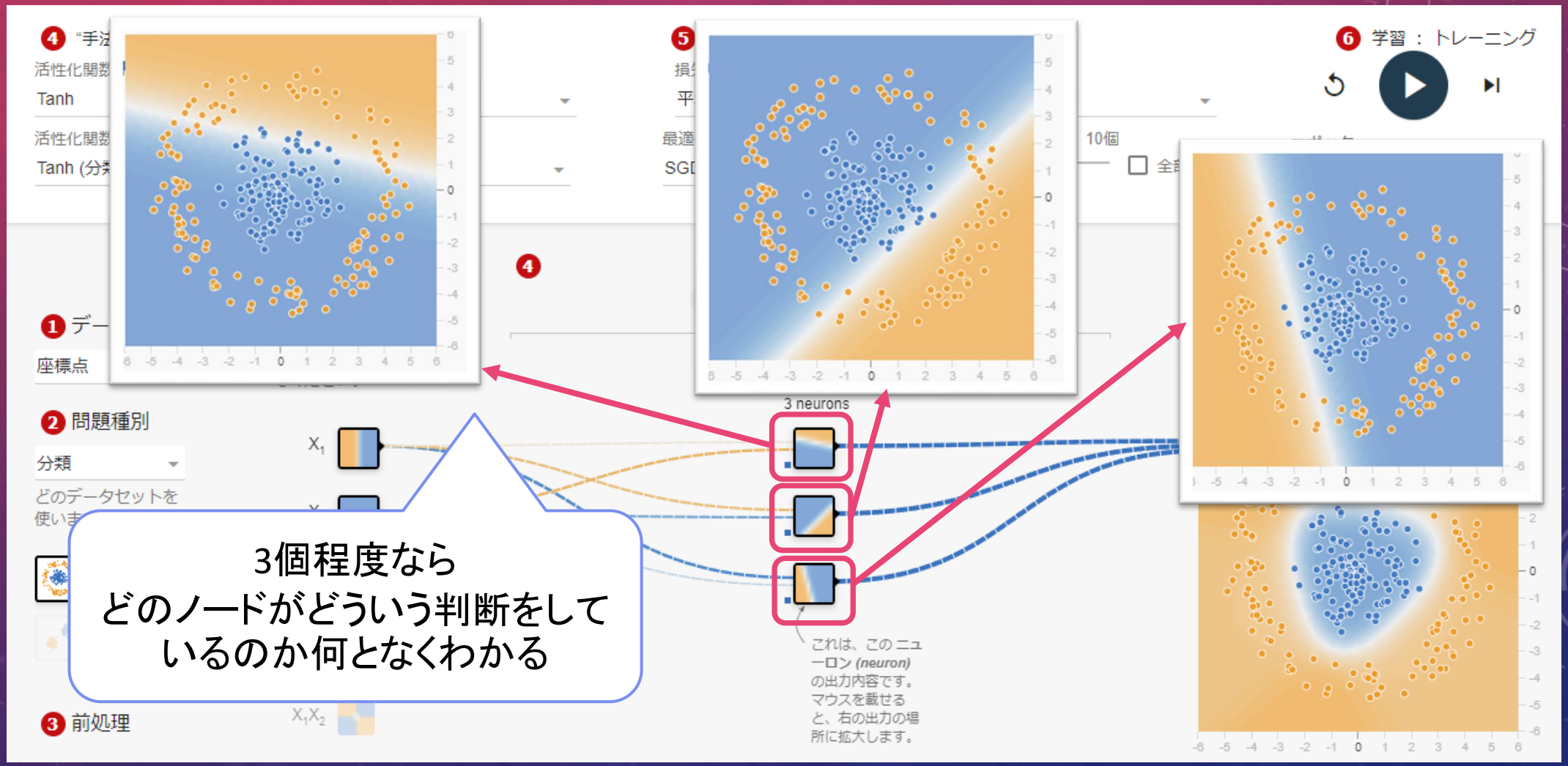
隠れ層1、ノード3個にしてみる：2分類問題その2



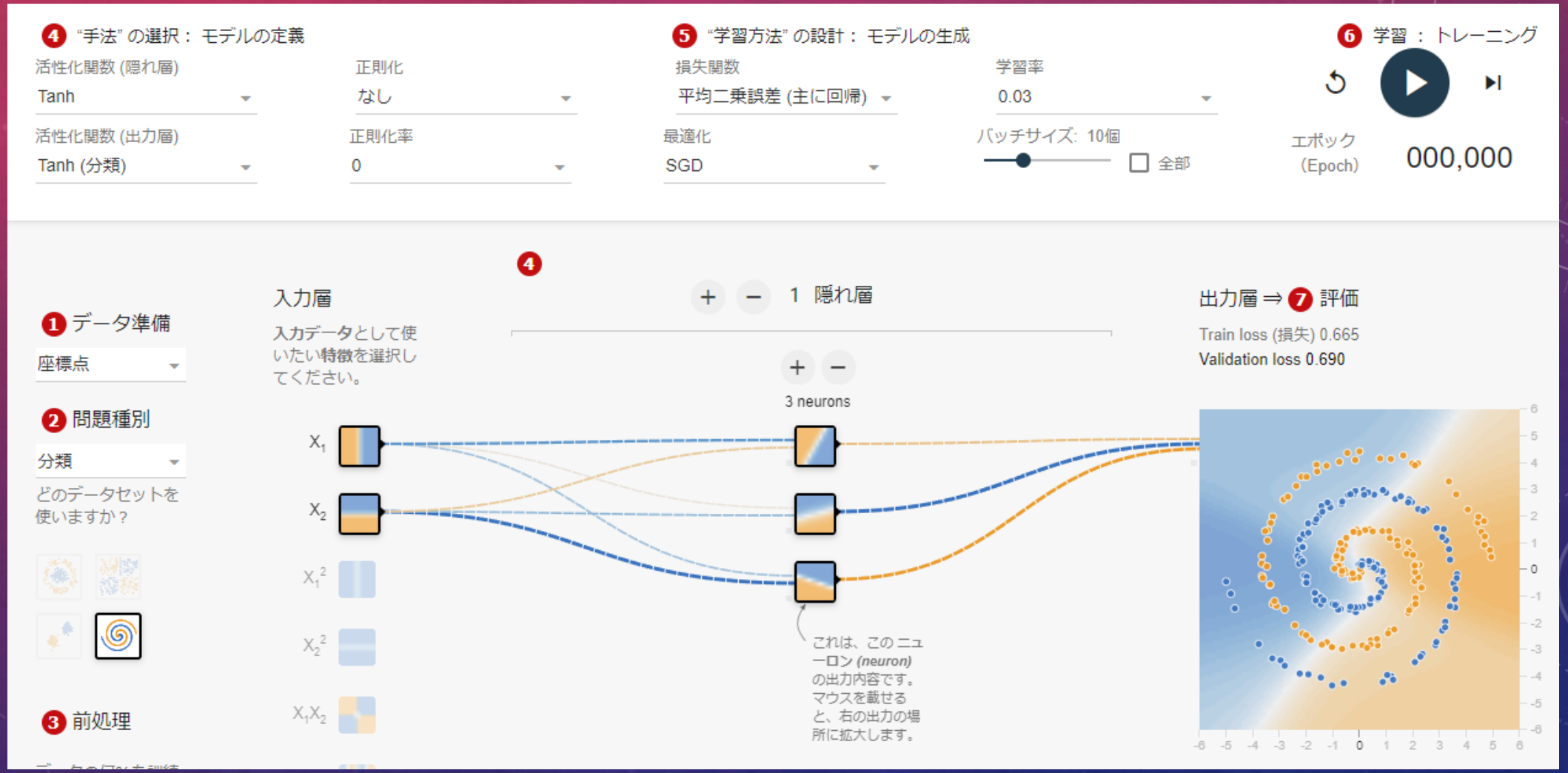
隠れ層1、ノード3個にしてみる：2分類問題その2



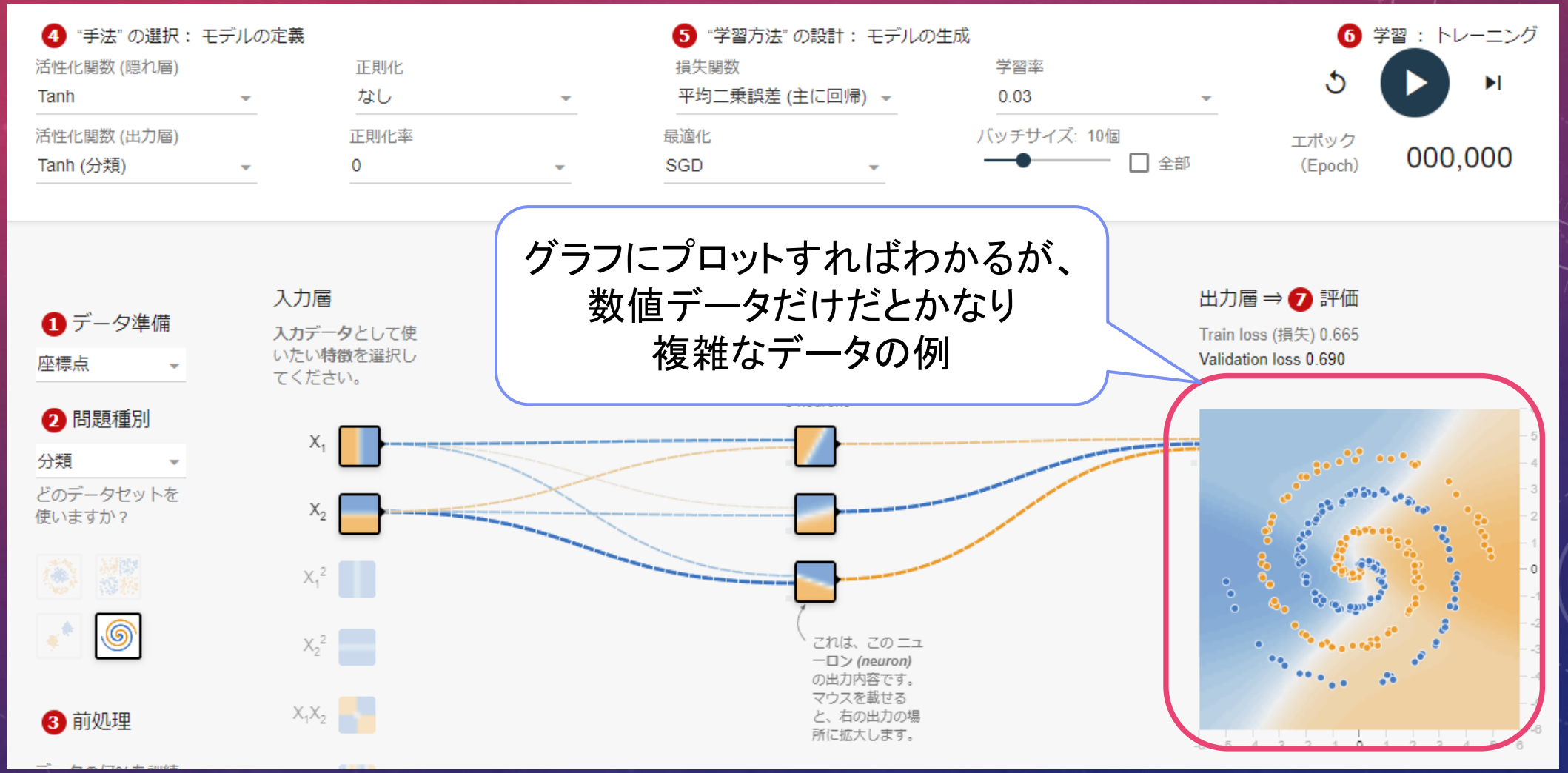
隠れ層1、ノード3個にしてみる：2分類問題その2



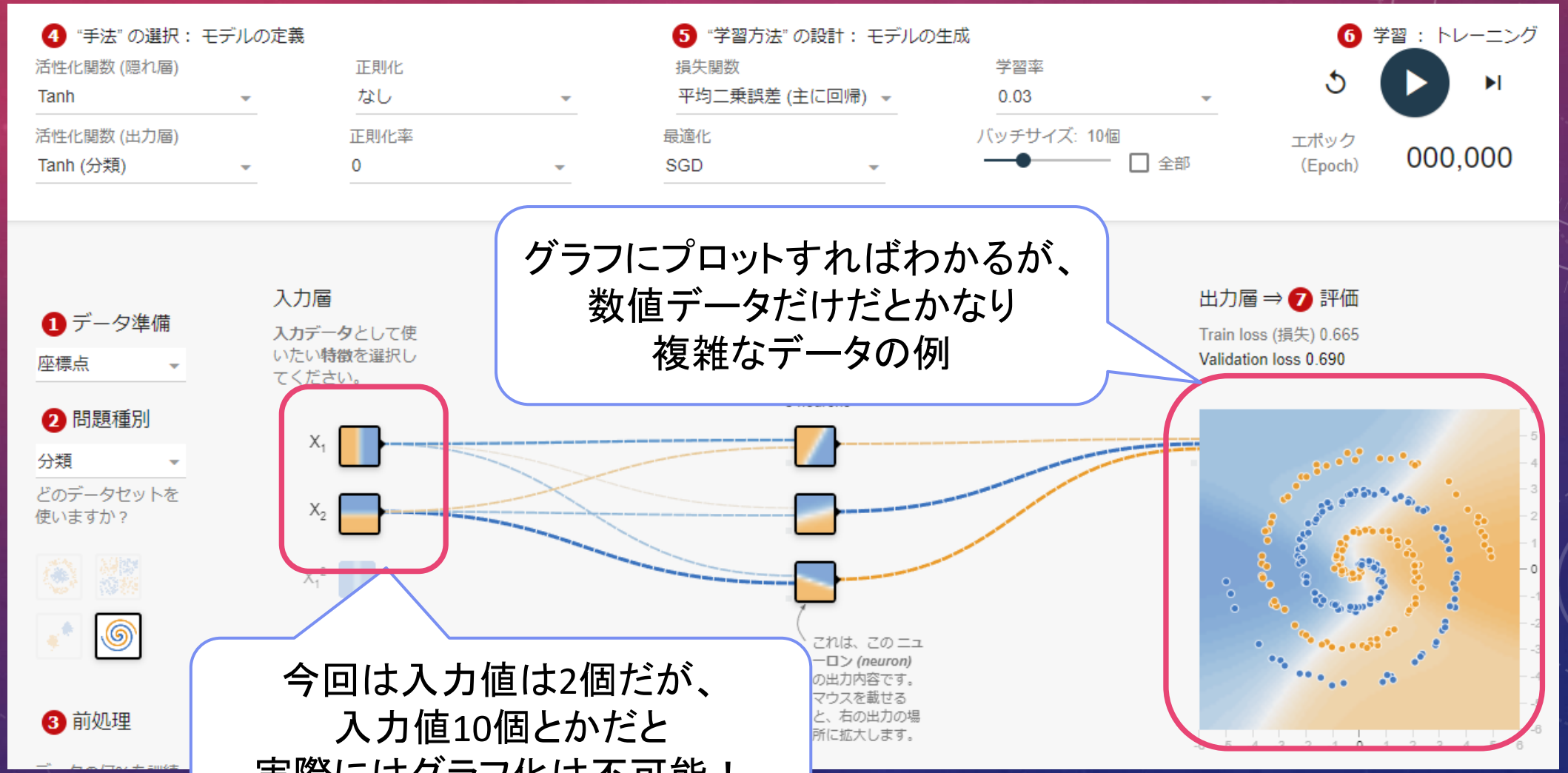
渦を巻いたようなデータだと？：2分類問題その 3



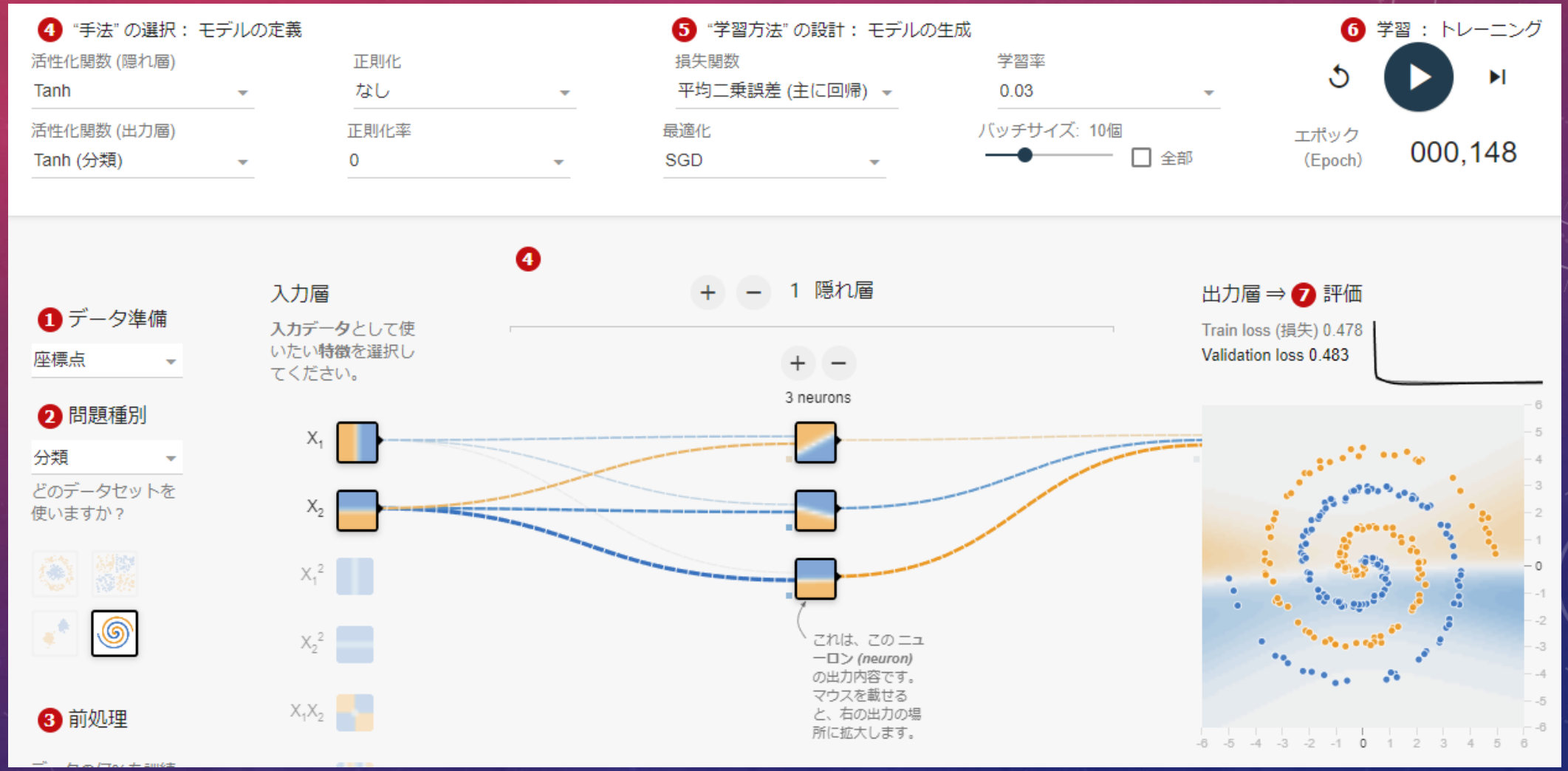
渦を巻いたようなデータだと？：2分類問題その３



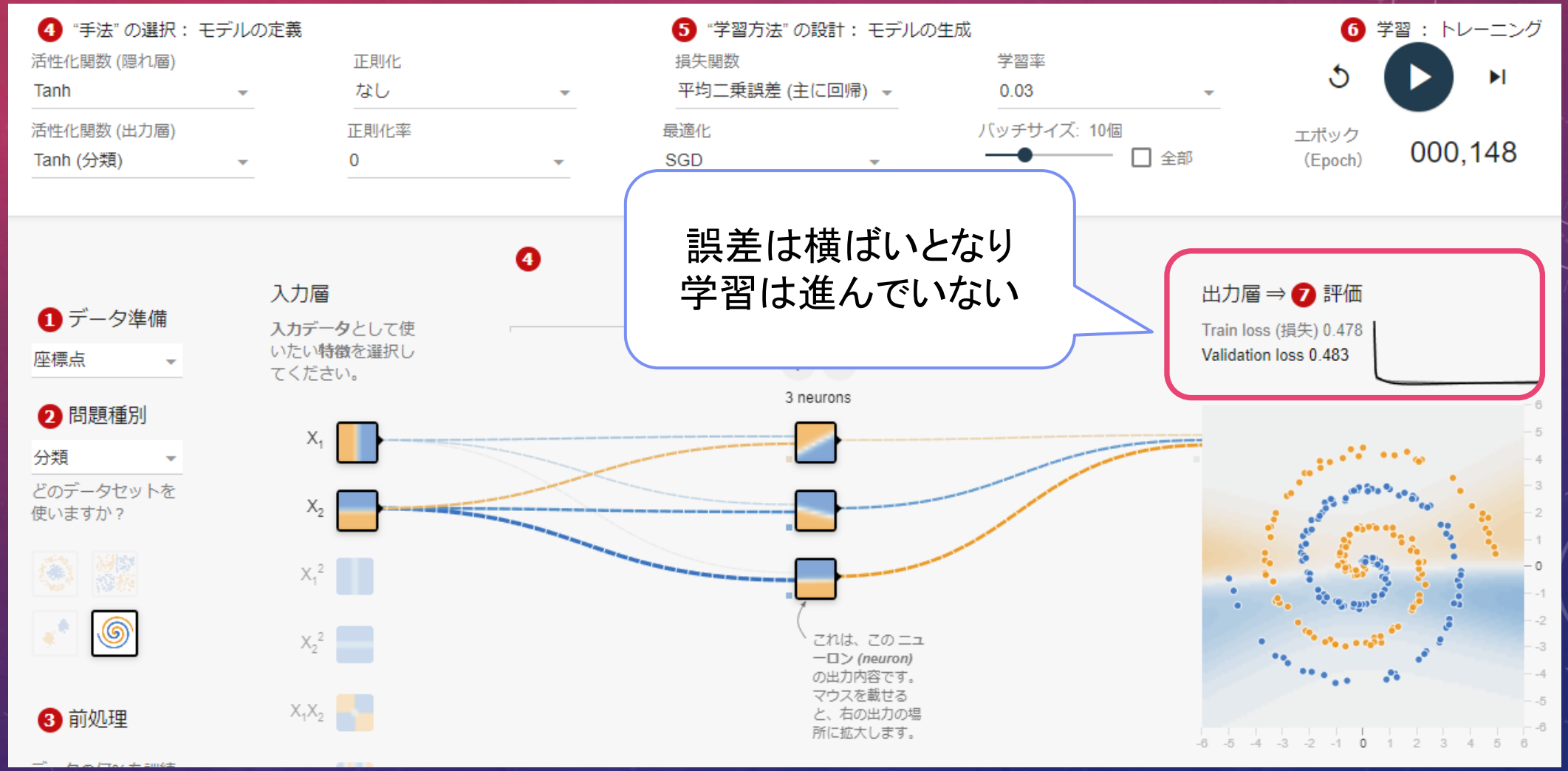
渦を巻いたようなデータだと？：2分類問題その３



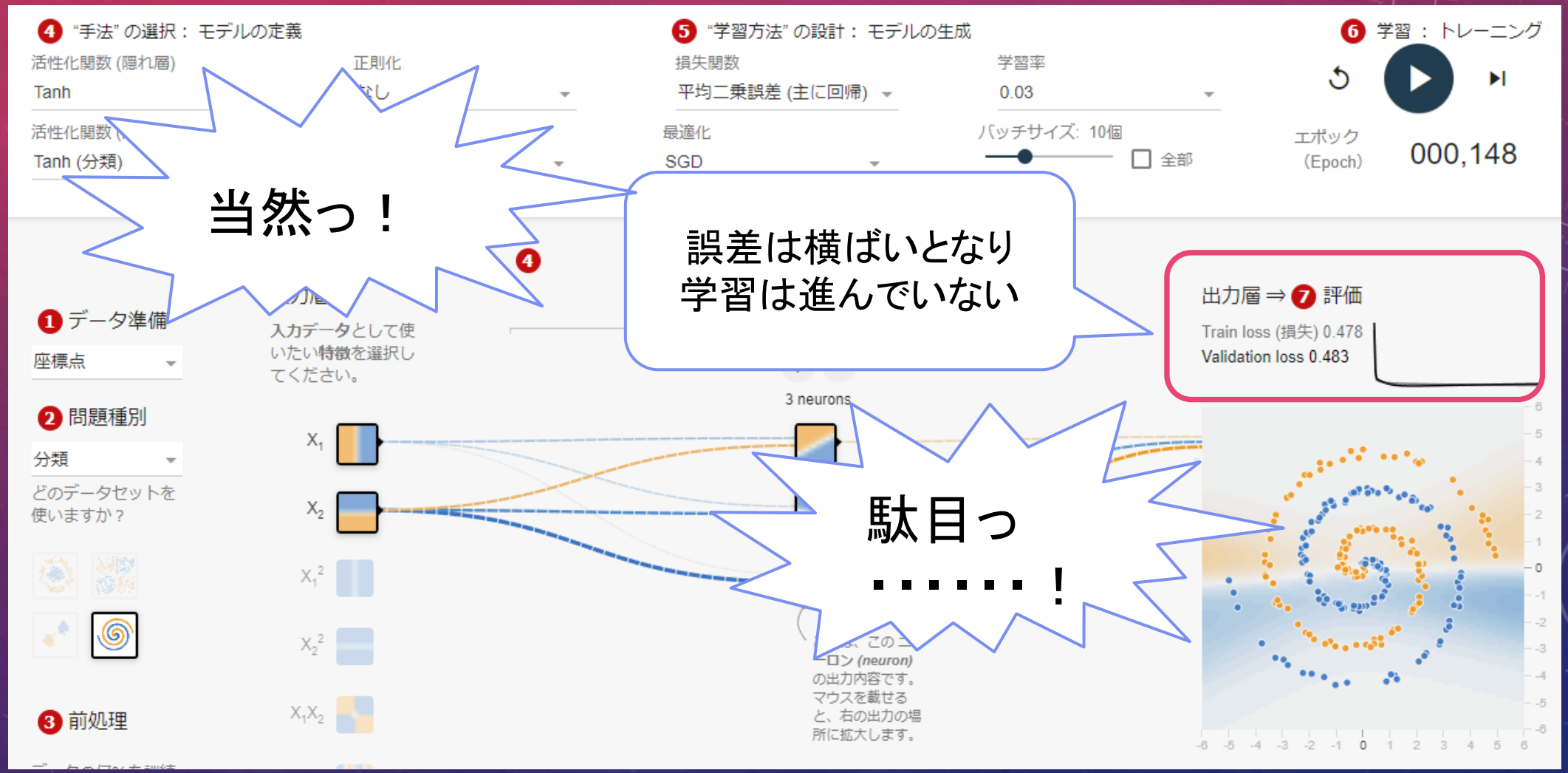
渦を巻いたようなデータだと？：2分類問題その 3



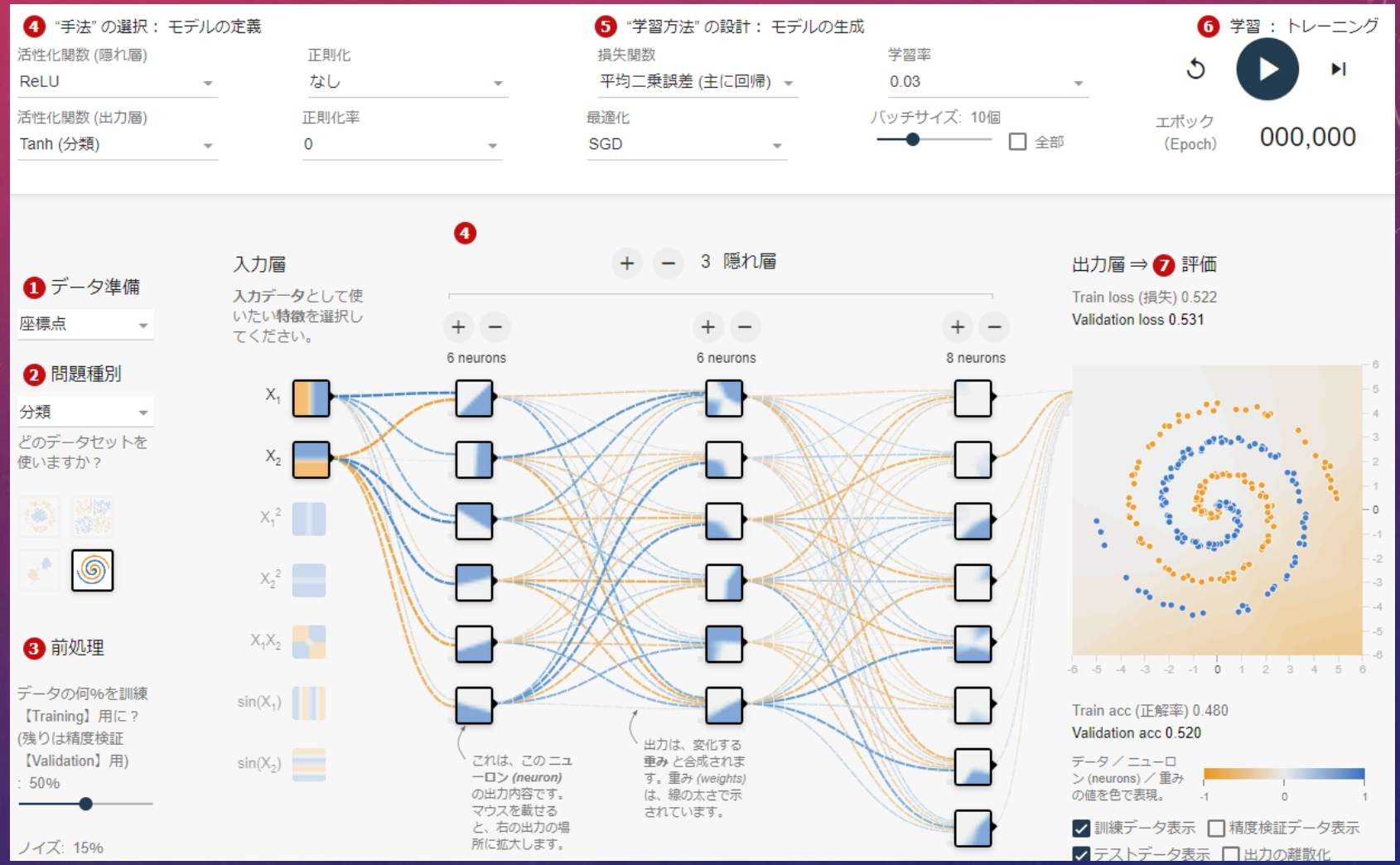
渦を巻いたようなデータだと？：2分類問題その 3



渦を巻いたようなデータだと？：2分類問題その 3

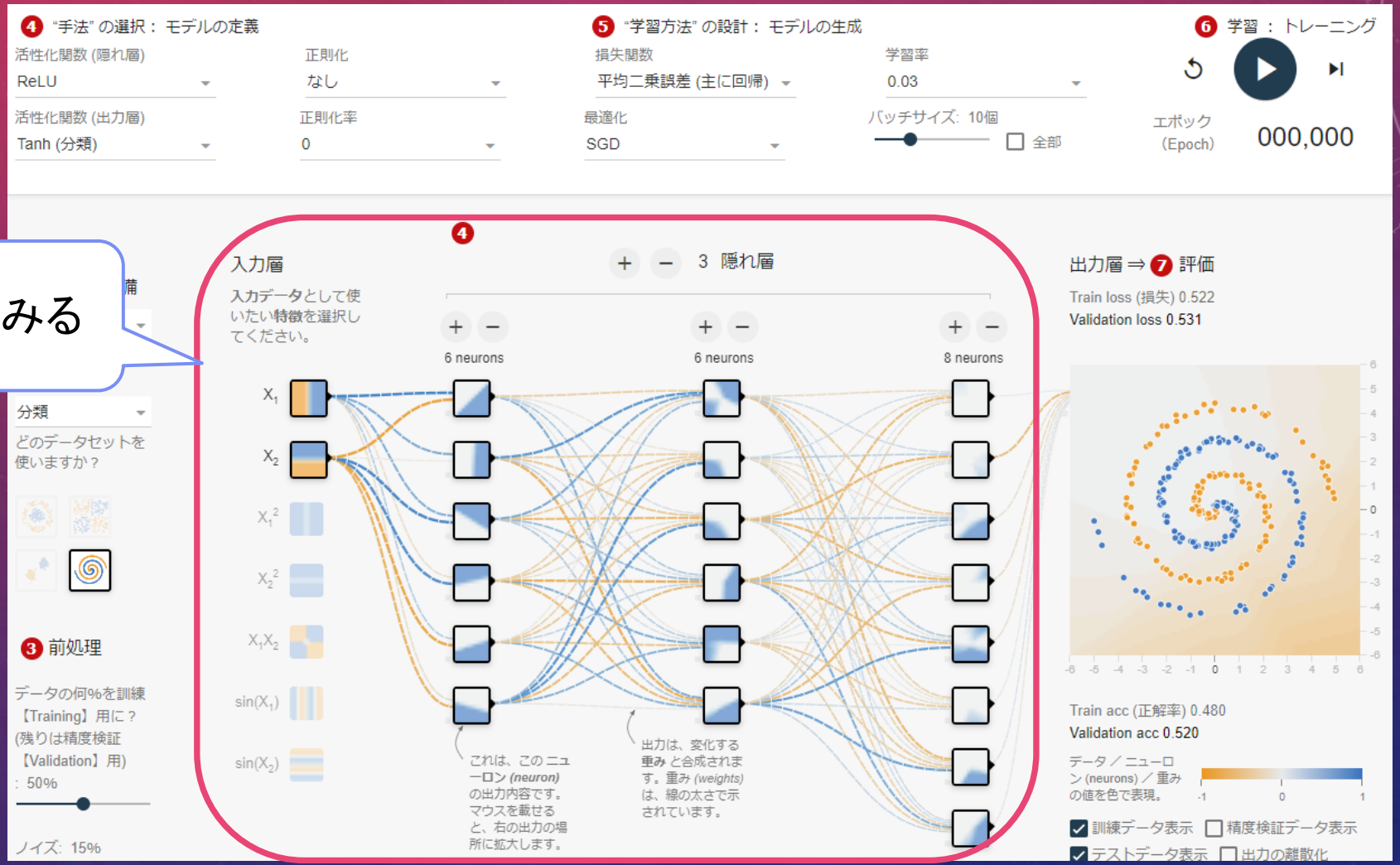


3層でそれぞれ6、6、8ノードだと：2分類問題その 3

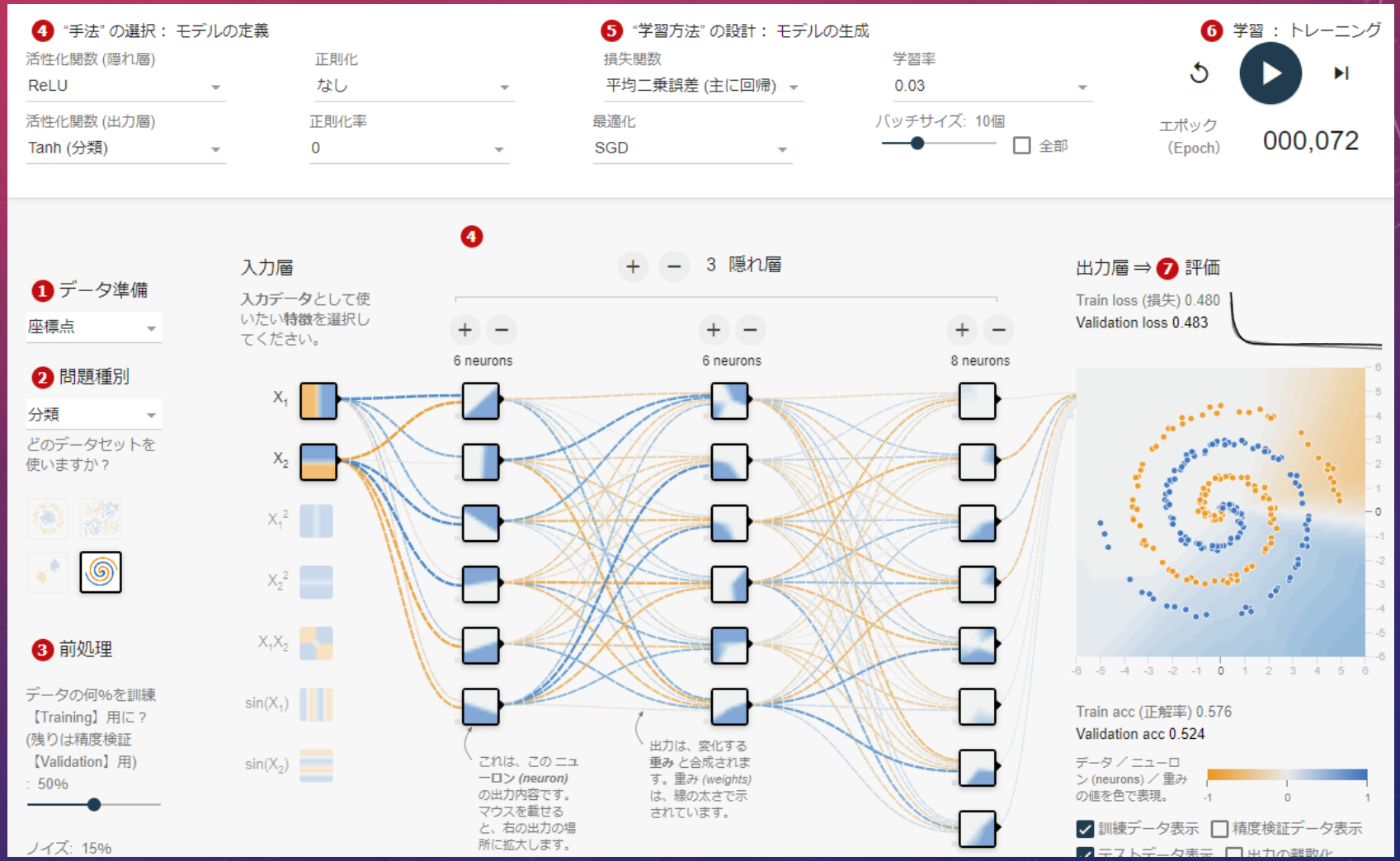


3層でそれぞれ6、6、8ノードだと：2分類問題その３

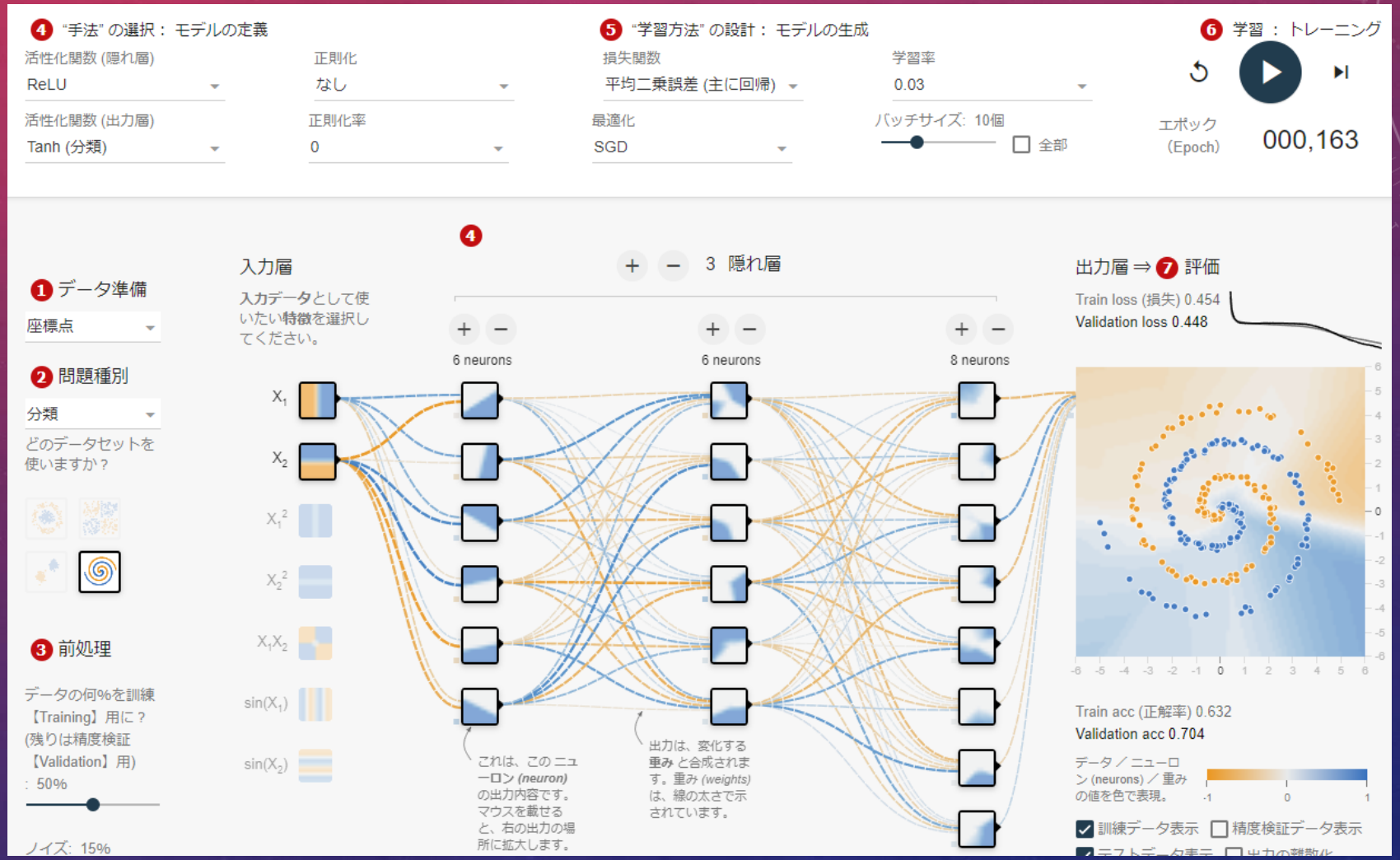
一気に増やしてみる



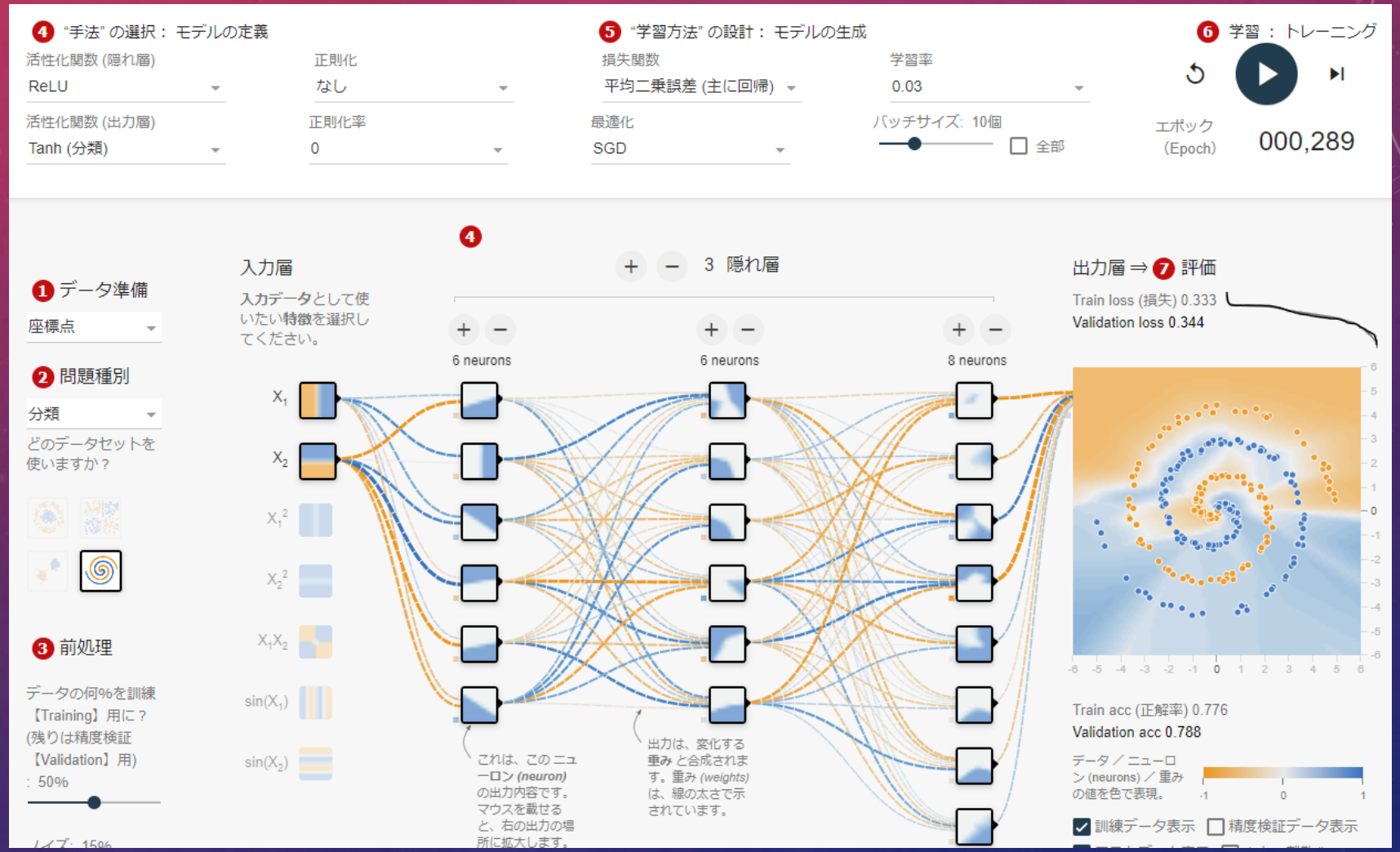
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



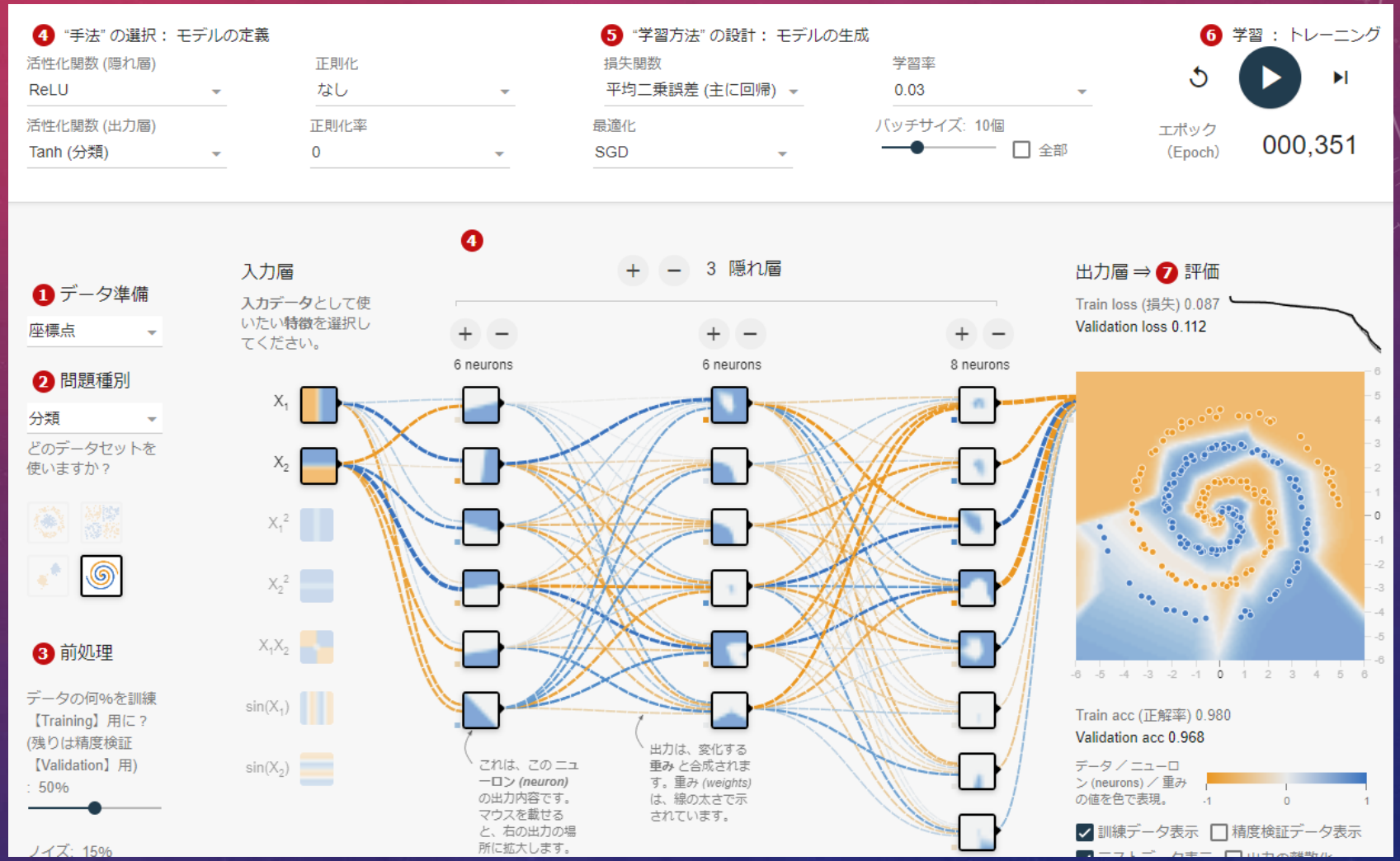
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



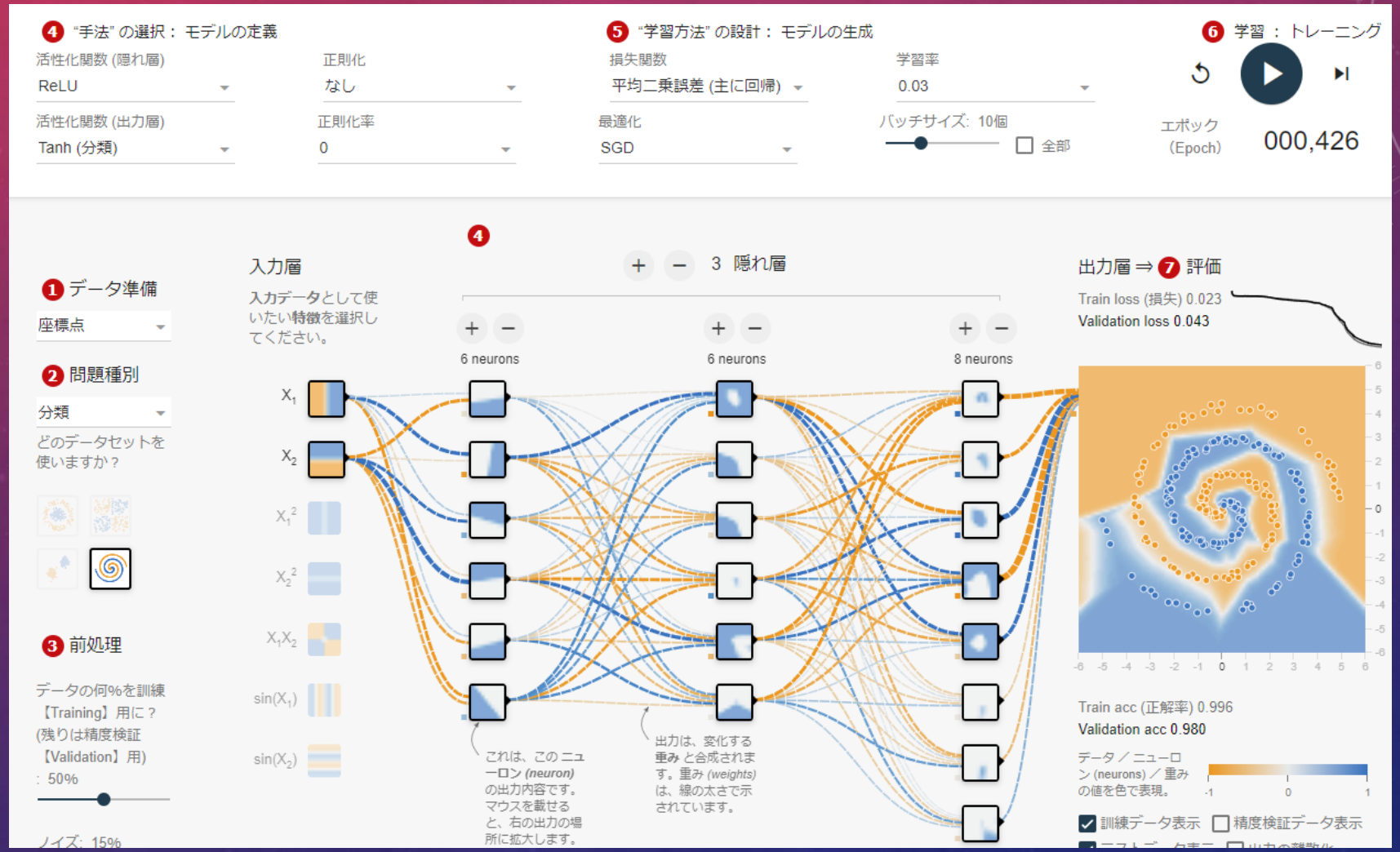
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



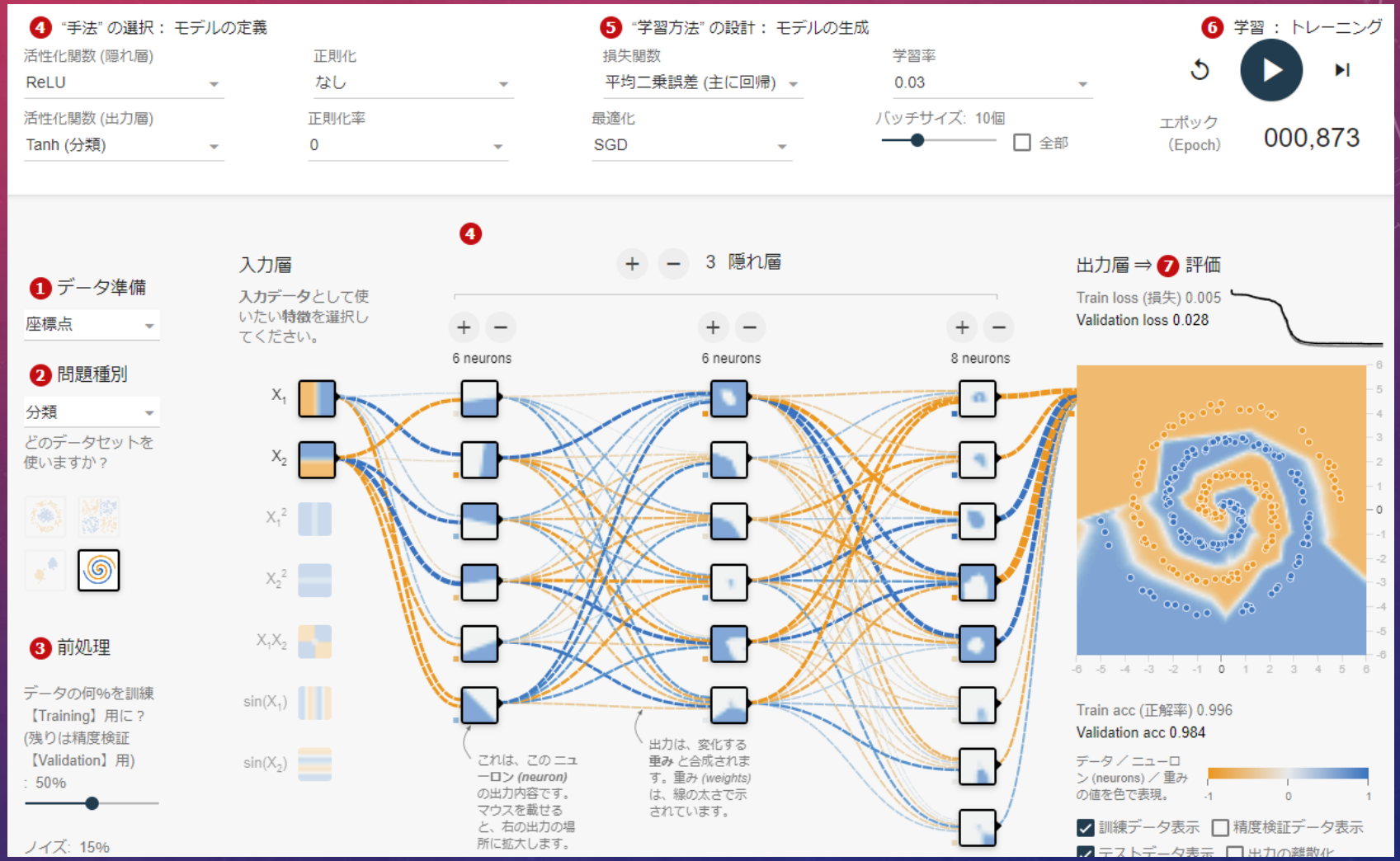
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



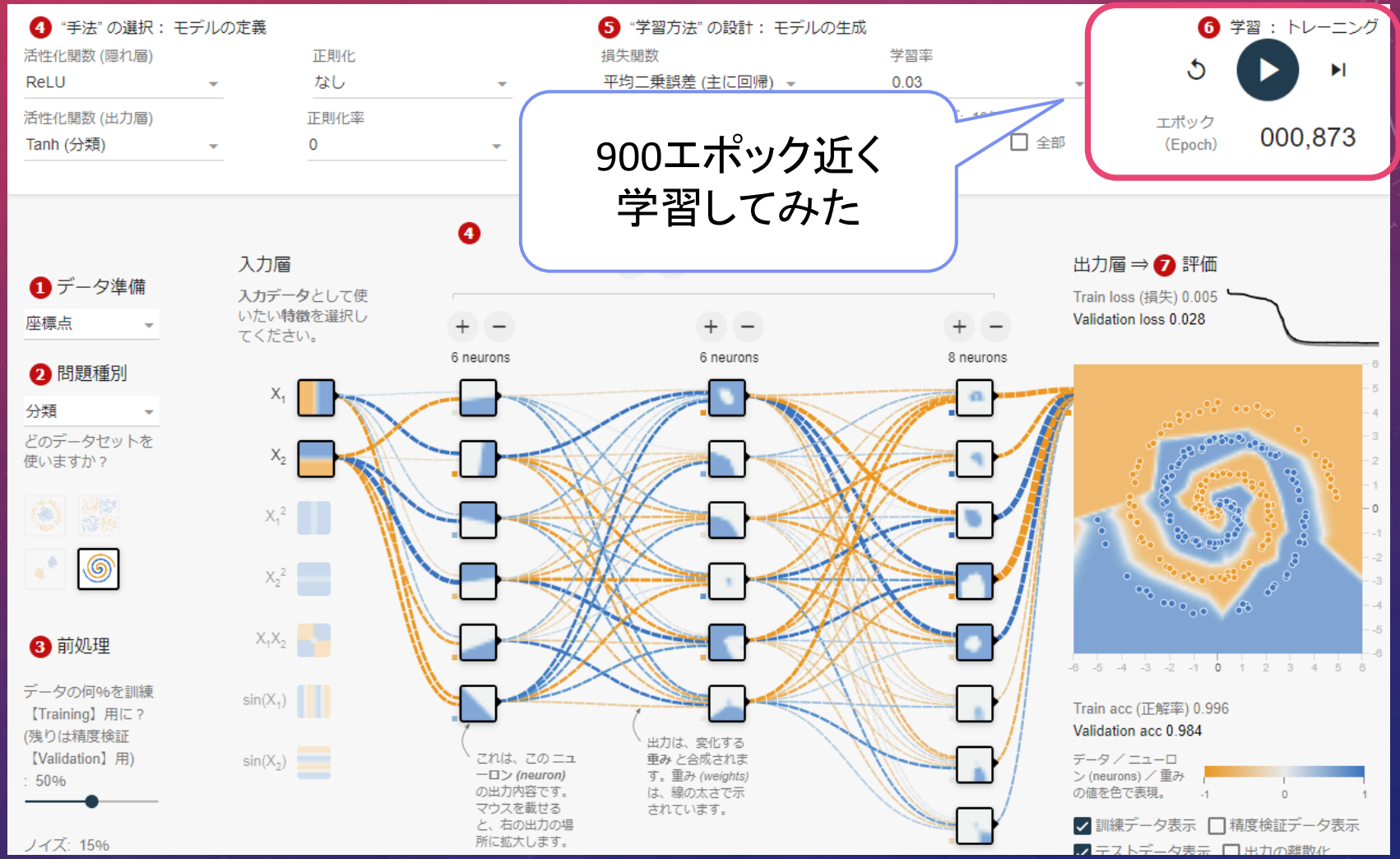
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



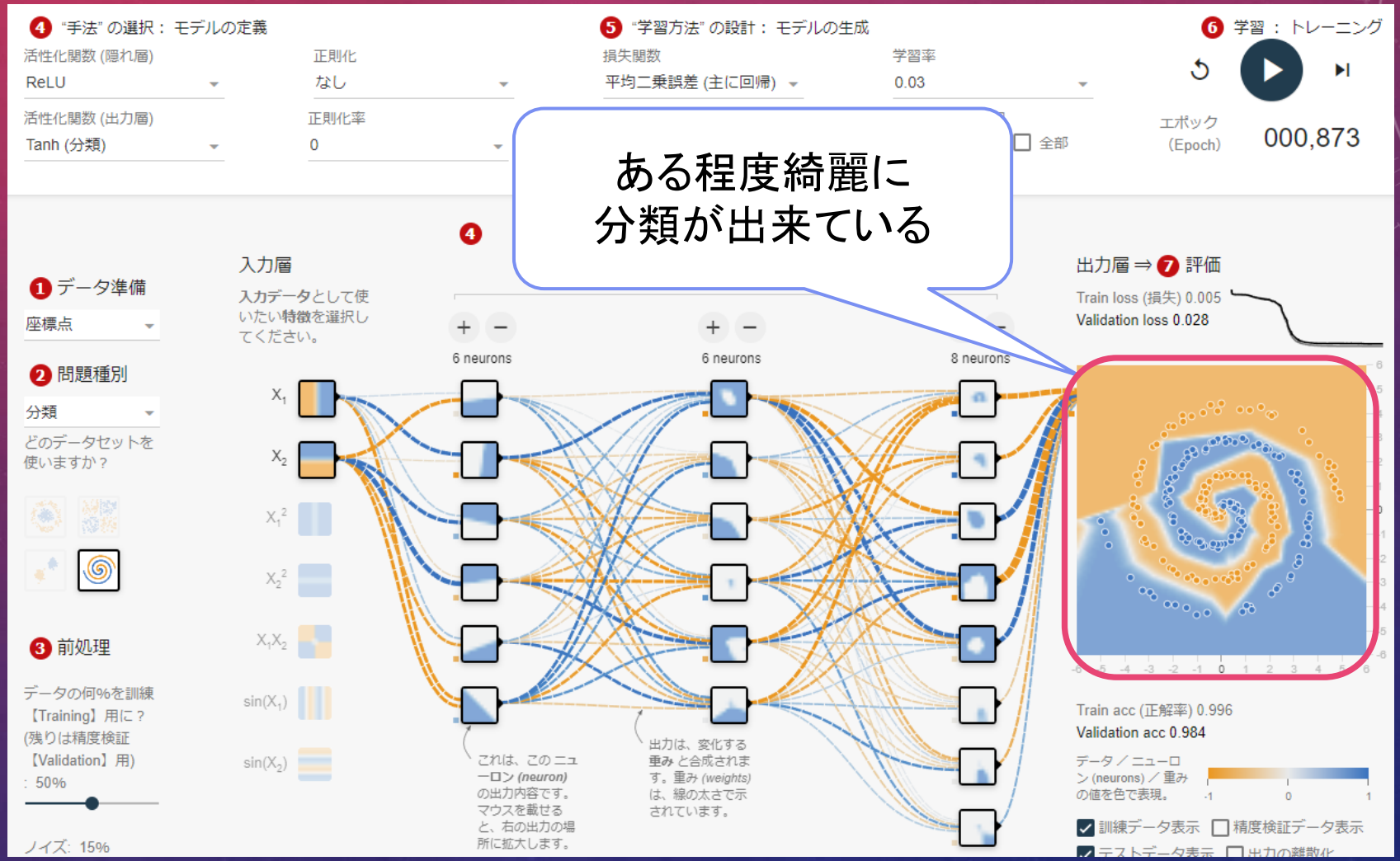
3層でそれぞれ6、6、8ノードだと：2分類問題その 3



3層でそれぞれ6、6、8ノードだと：2分類問題その３

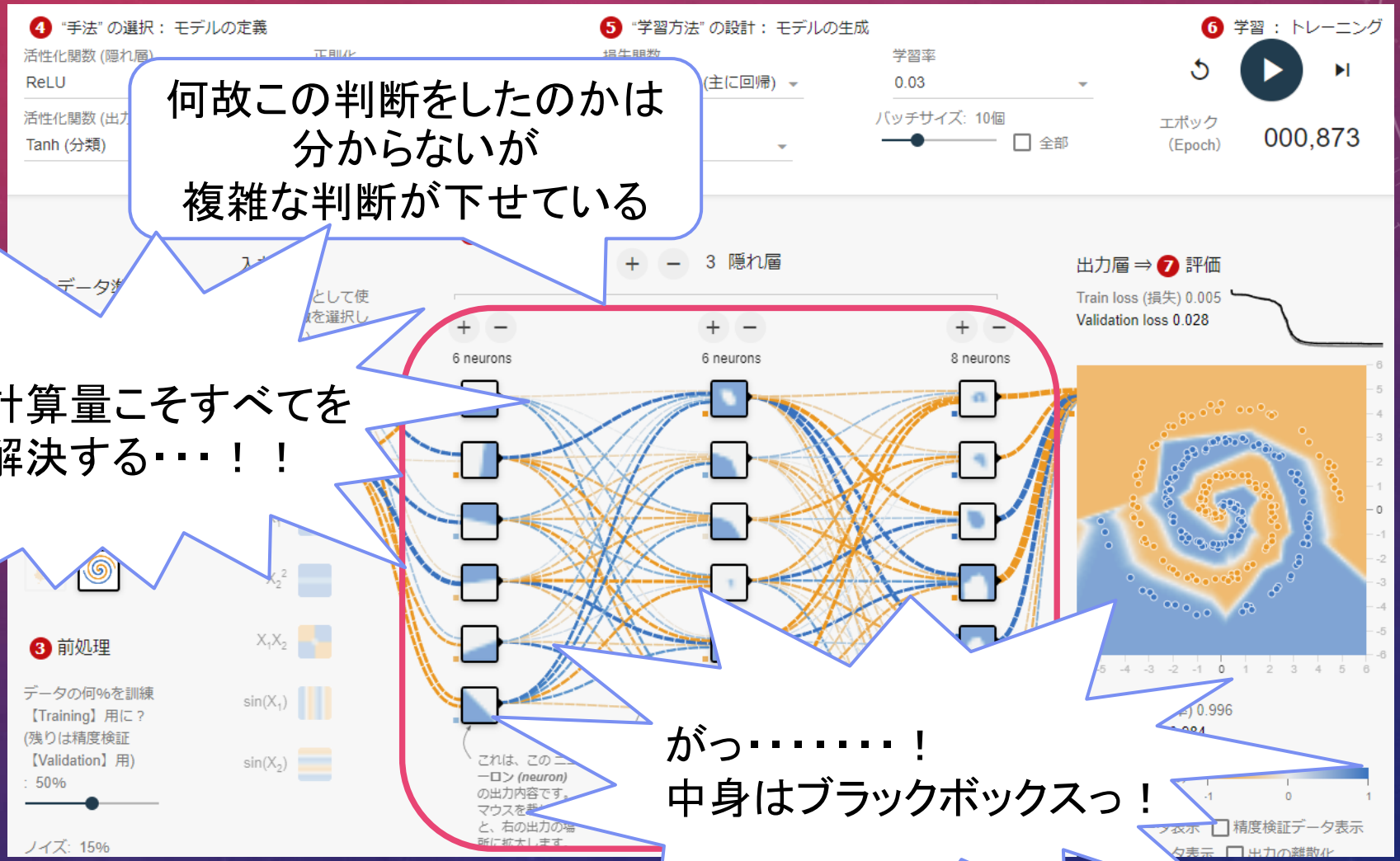


3層でそれぞれ6、6、8ノードだと：2分類問題その３



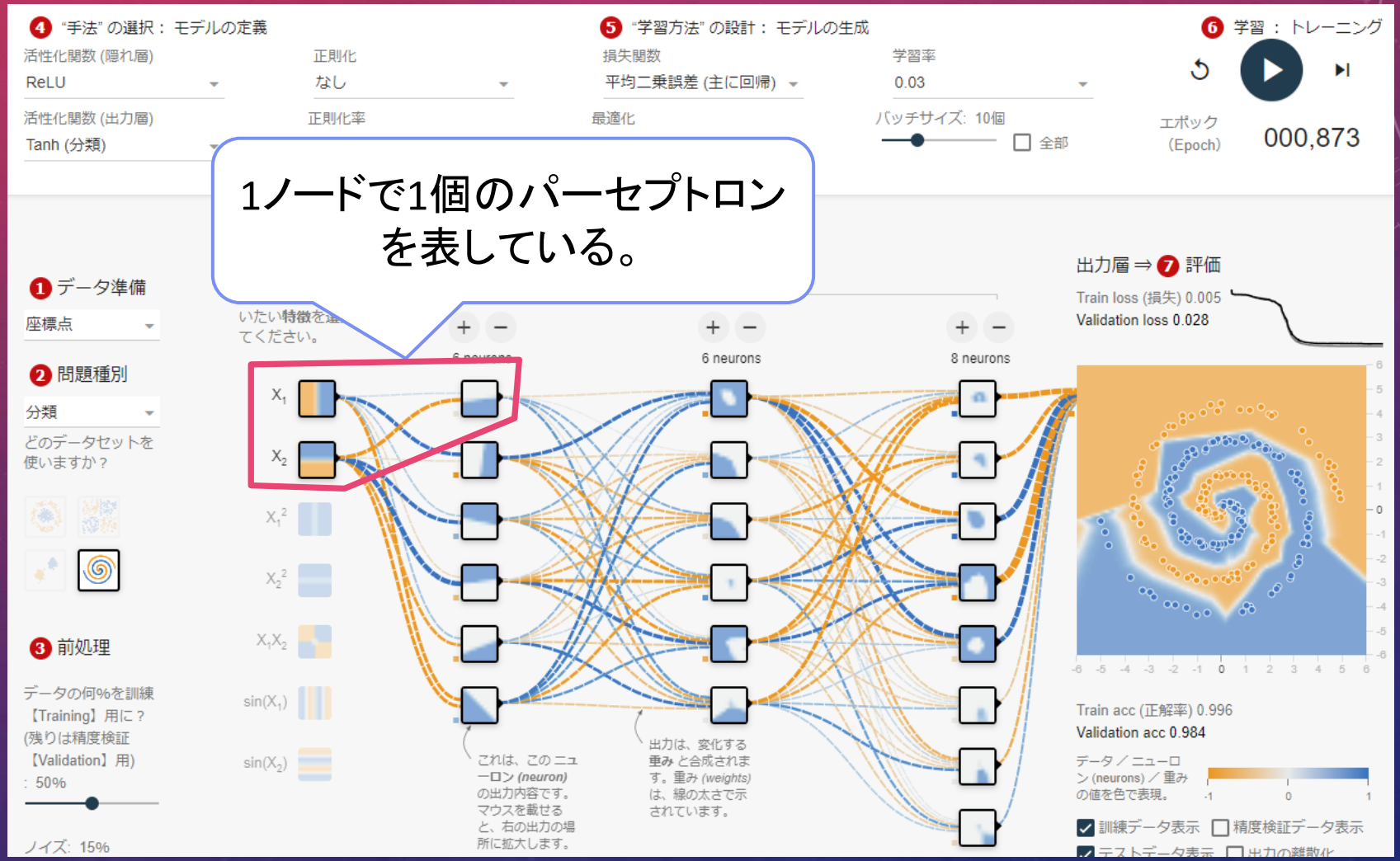
3層でそれぞれ6、6、8ノードだと：2分類問題その３

3層でそれぞれ6、6、8ノードだと：2分類問題その３

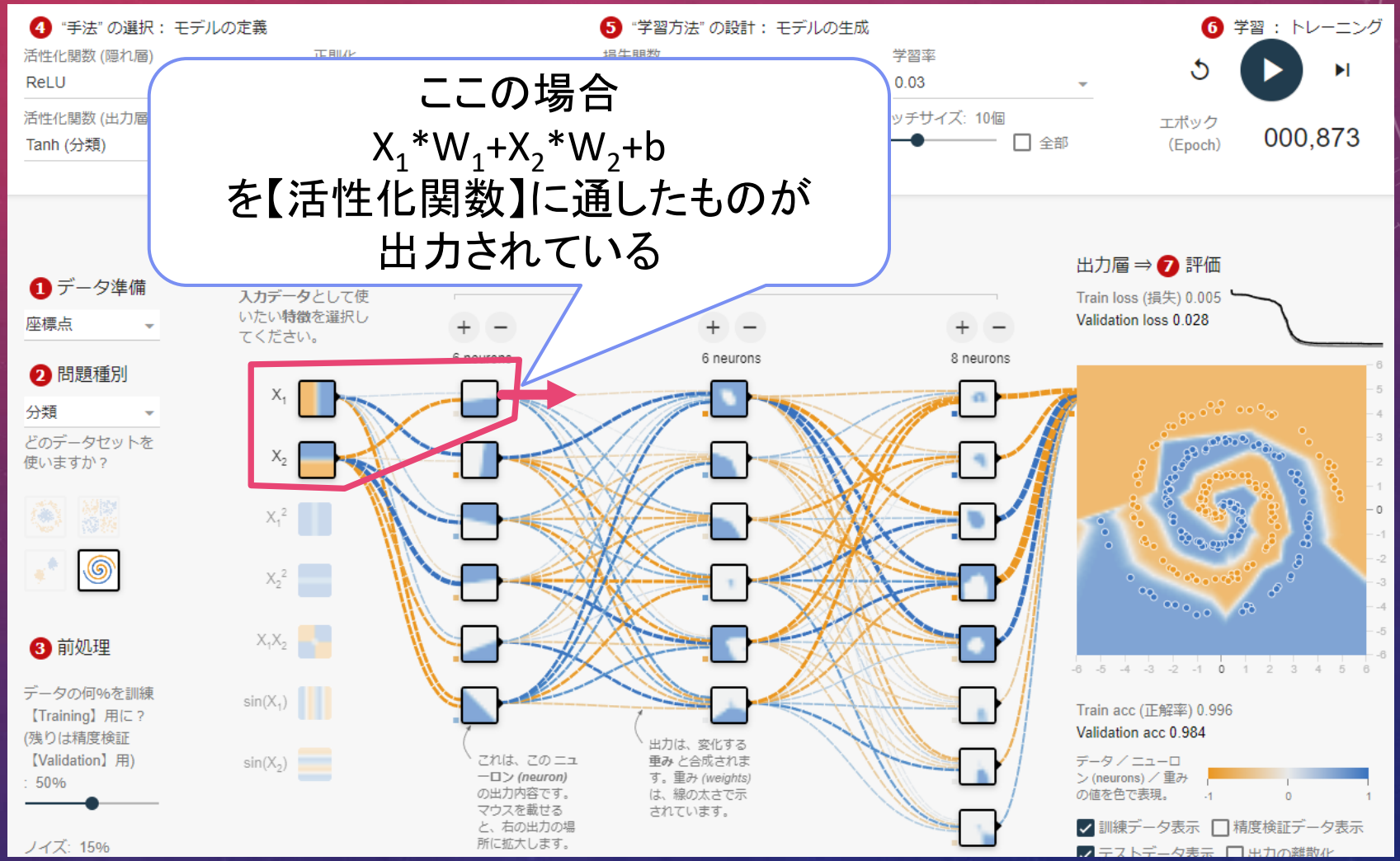


3層でそれぞれ6、6、8ノードだと：2分類問題その３

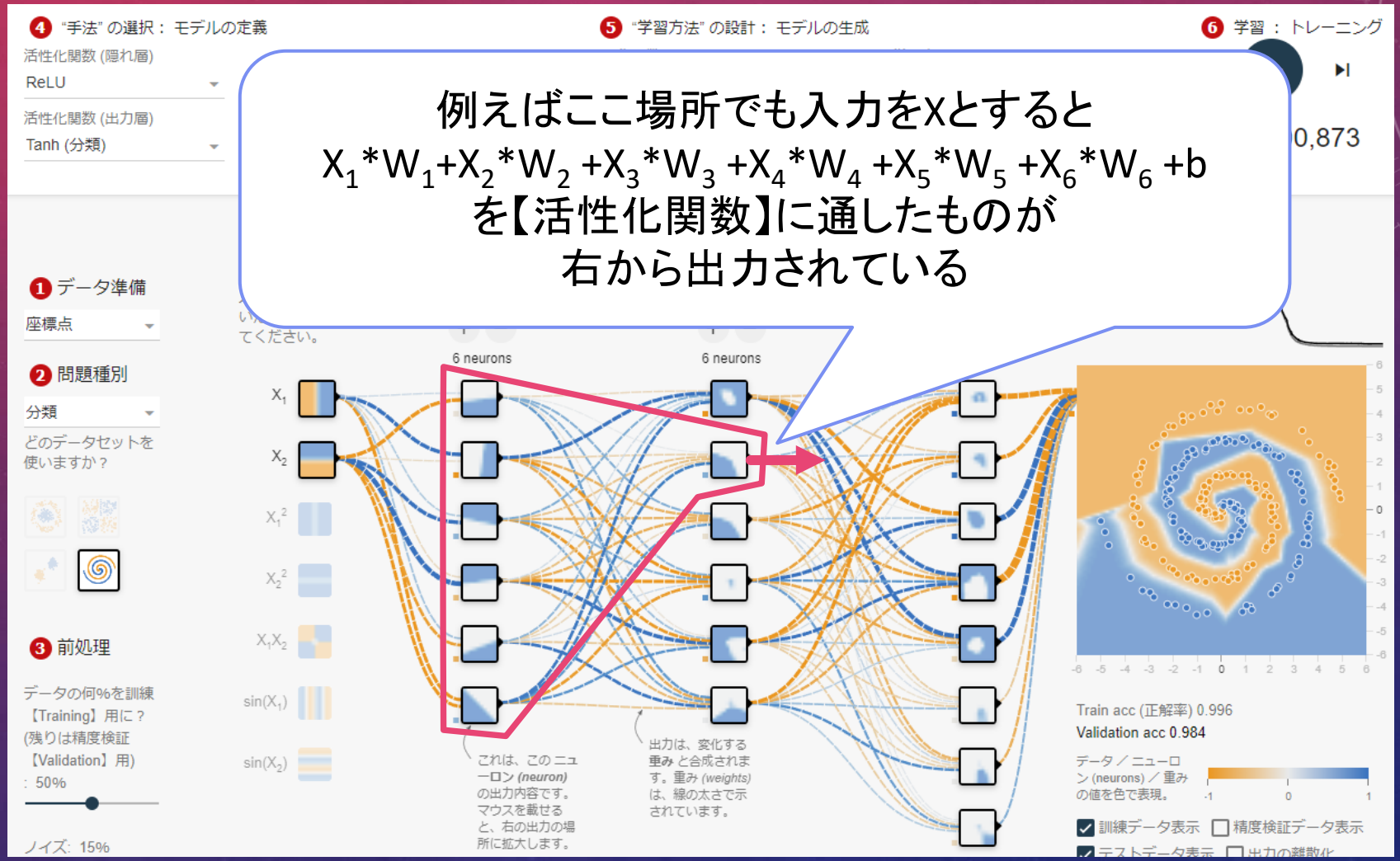
詳細を見てみると


4 “手法” の選択： モデルの定義
活性化関数 (隠れ層)
ReLU
正則化
なし
活性化関数 (出力層)
Tanh (分類)
正則化率
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
バッチサイズ: 10個
全部
6 学習 ： トレーニング
エポック (Epoch)
000,873
1ノードで1個のパーセプトロンを表している。
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%
いたい特徴を選
てください。
6 neurons
8 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
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

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
ッチサイズ: 10個
全部
6 学習 ： トレーニング
エポック
(Epoch)
000,873
ここの場合
$X_1*W_1+X_2*W_2+b$
を【活性化関数】に通したものが
出力されている
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%
入力データとして使
いたい特徴を選択し
てください。
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) 0.005
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

詳細を見てみると


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
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
6 学習 ： トレーニング
活性化関数 (隠れ層)
ReLU
活性化関数 (出力層)
Tanh (分類)
0,873
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%
6 neurons
6 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。
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

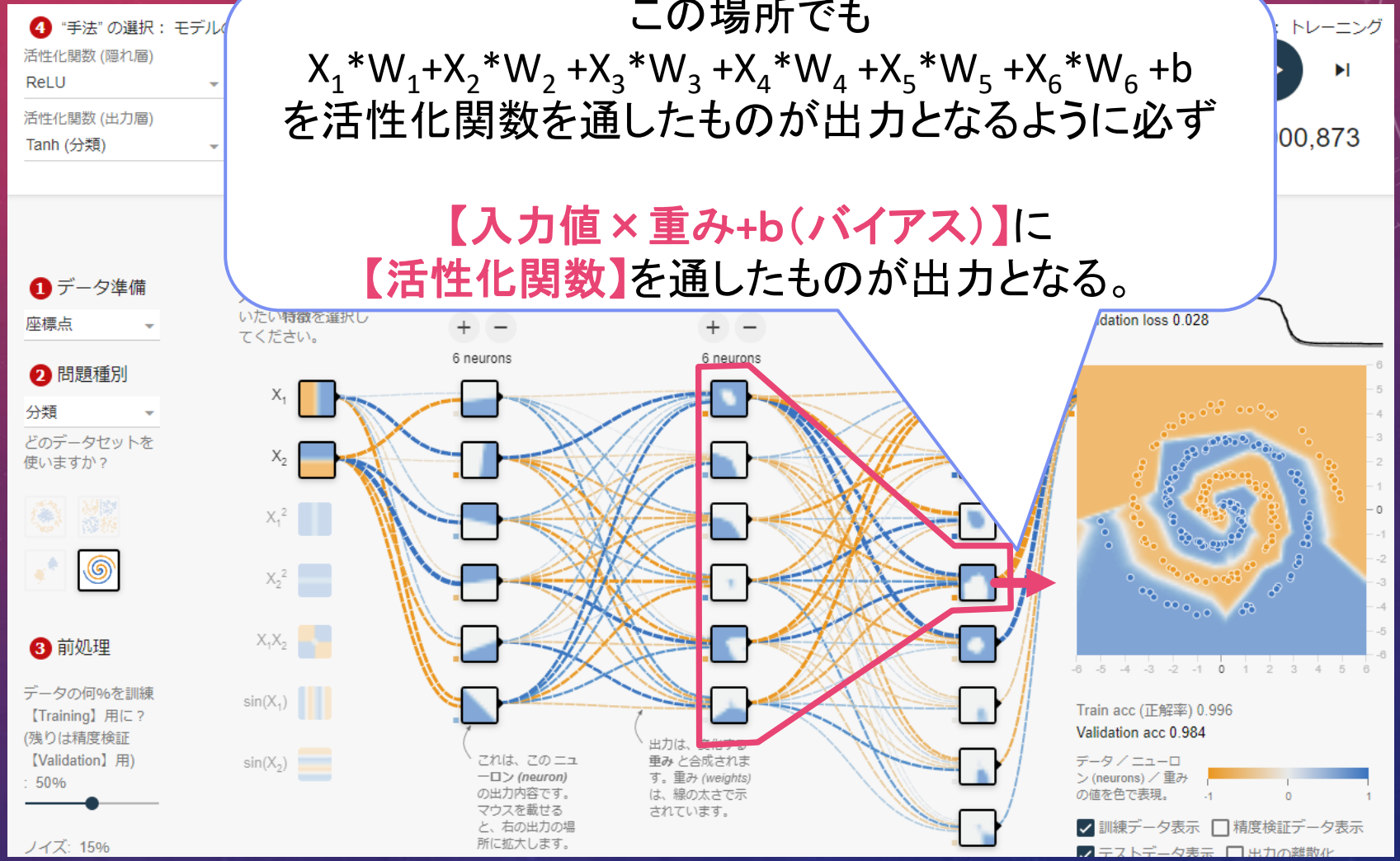
詳細を見てみると

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。



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

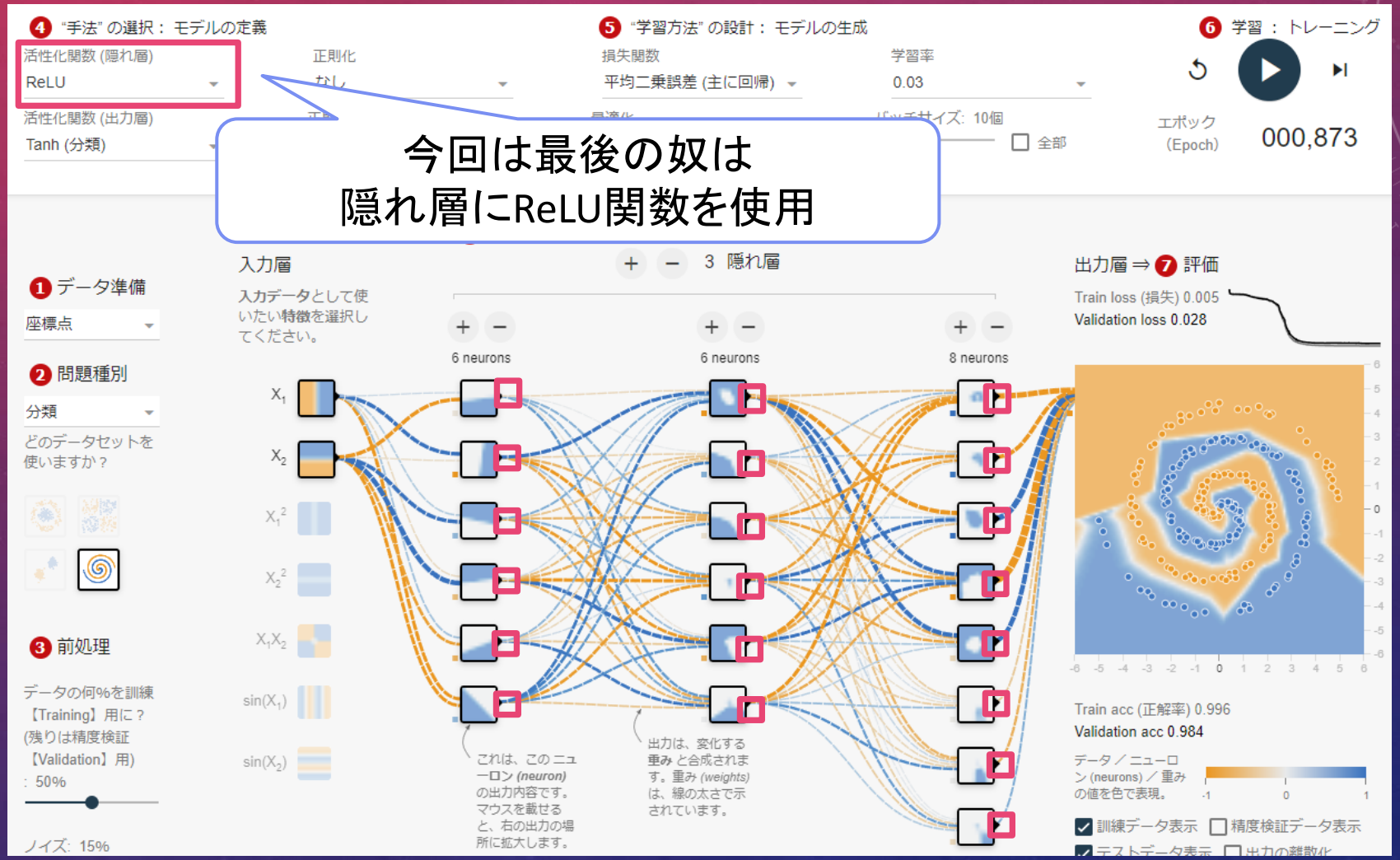
【y=ax+b】と【活性化関数】をたくさん繋げている！

と言える！

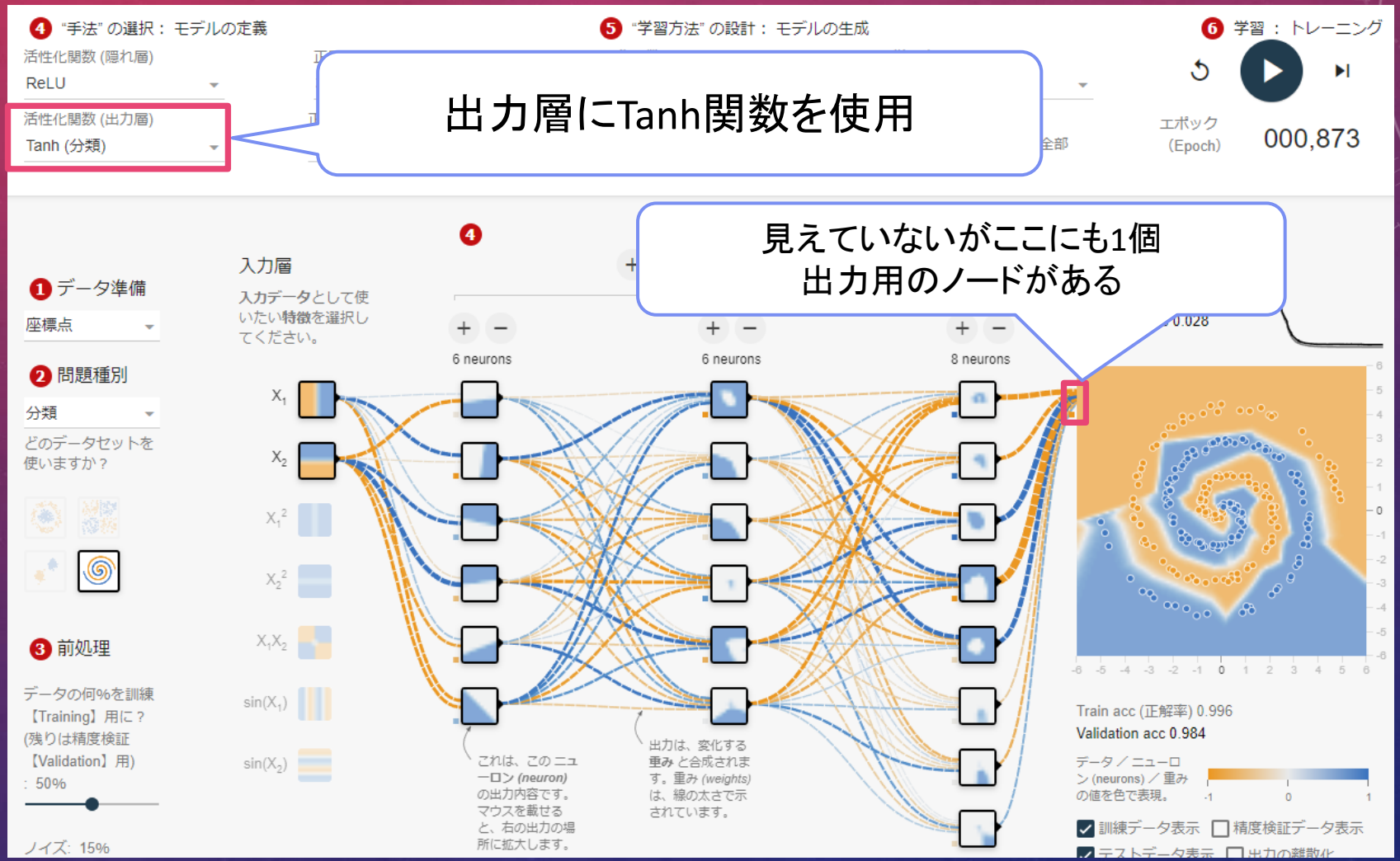
強引すぎるまとめ方っ！

だが、それがいい！！

おまけ：今回の活性化関数



おまけ：今回の活性化関数



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

ヒャア！！！
我慢できねぇ！！

あ[illegible]た

第[illegible] y=a[illegible] bから[illegible]める初心者向けML講座

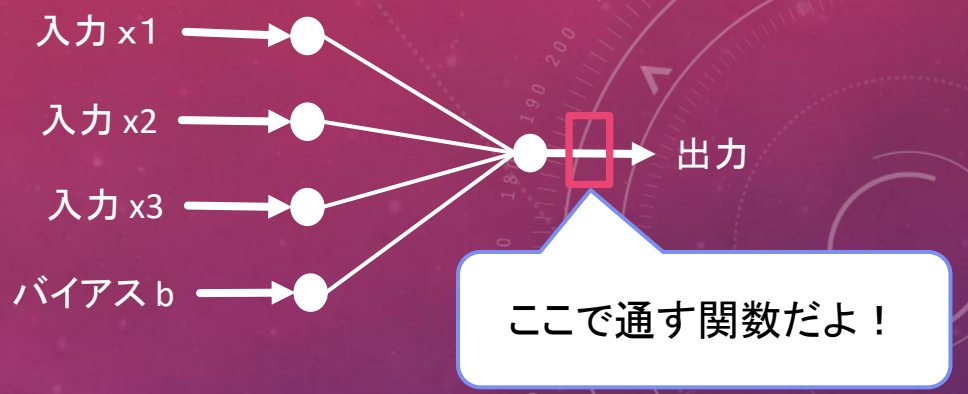
次回はタスク別の活性化関数と損失関数ニューラルネットワークの組み合わせ方法
（多分）

代表的な活性化関数

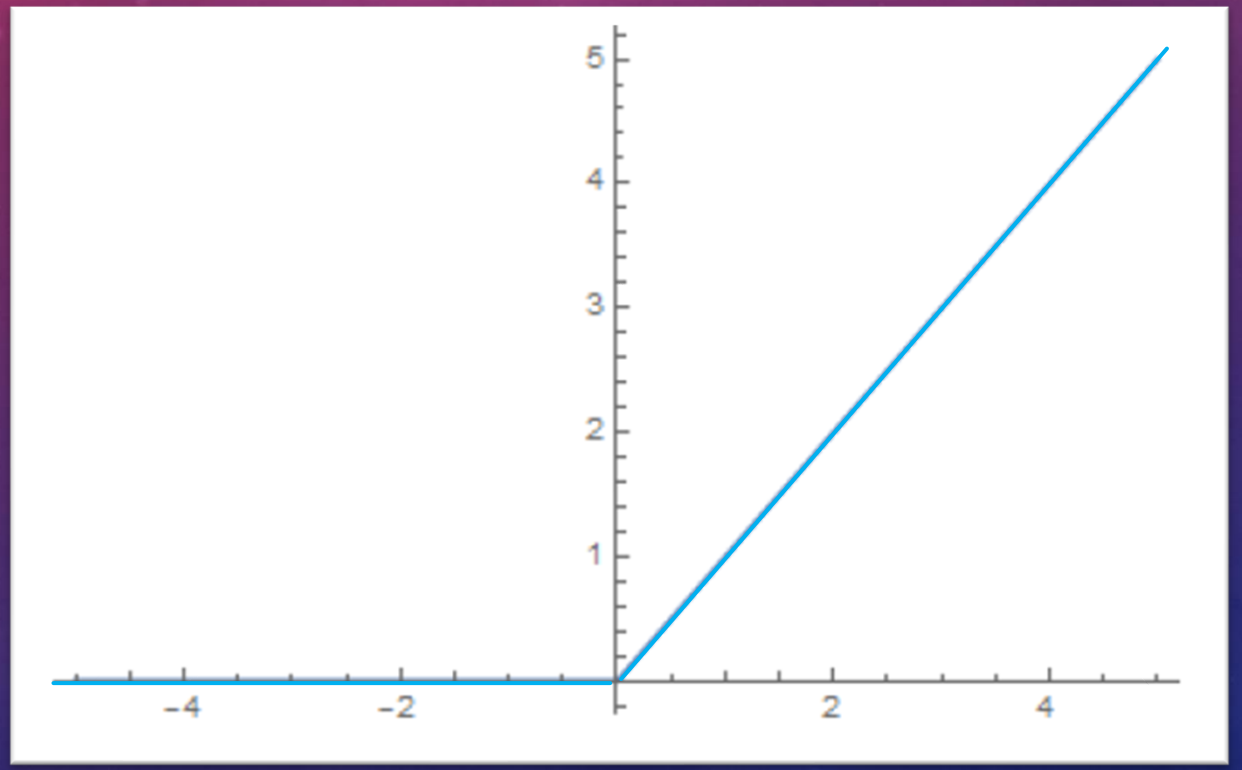
- ReLU関数
- Leaky ReLU関数
- シグモイド関数
- ソフトマックス関数
- Tanh（ハイパボリックタンジェント）関数

などがあります。

ReLU関数

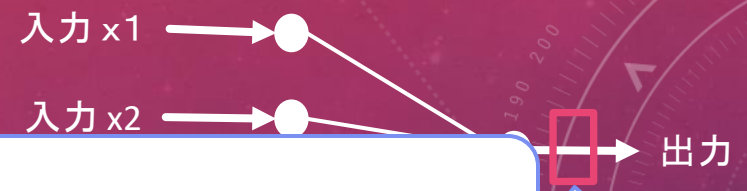


- F(x)= max{0,x}
- 0かX大きい方を出力（0以上ならそのまま出力）



ReLU関数

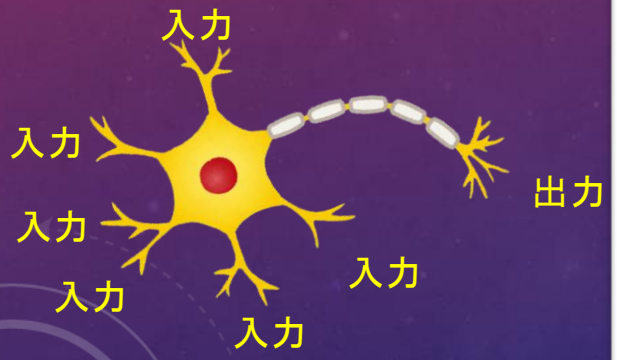
- F(x)= max{0,x}
- 0かX大きい方を出力（0以上ならそのまま出力）

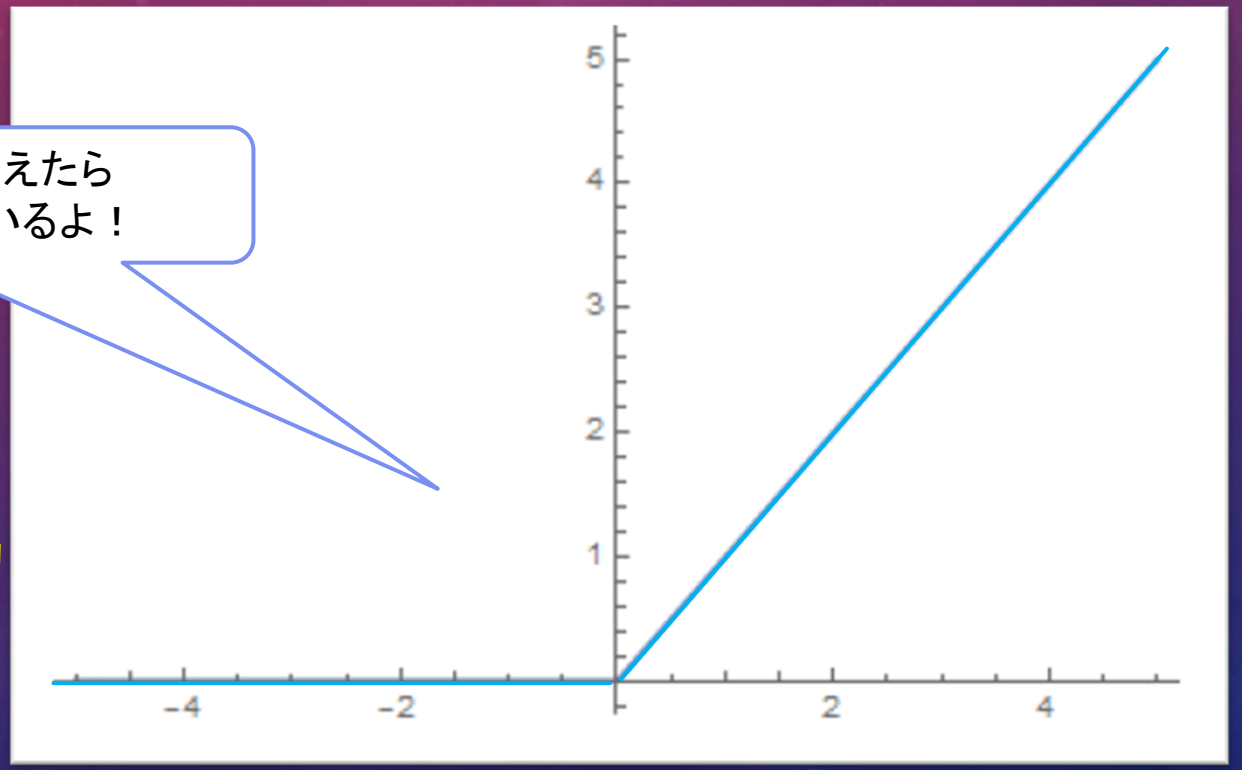


隠れ層はこれを使うのが一般的だよ！

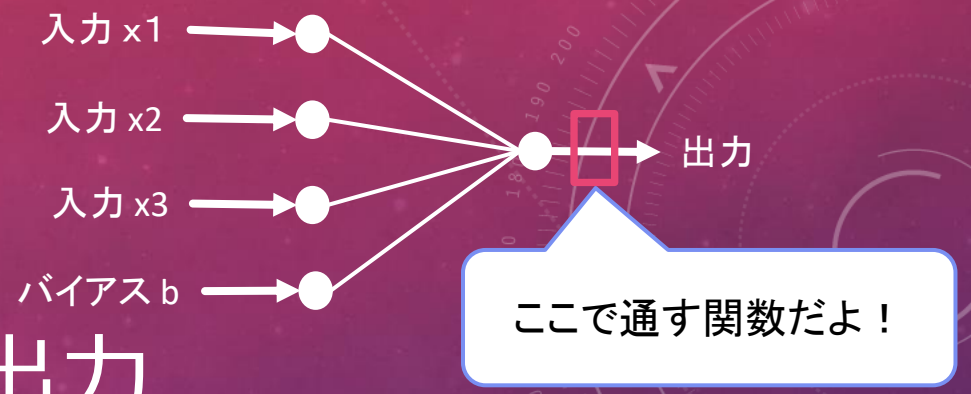
ここで通す関数だよ！

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！

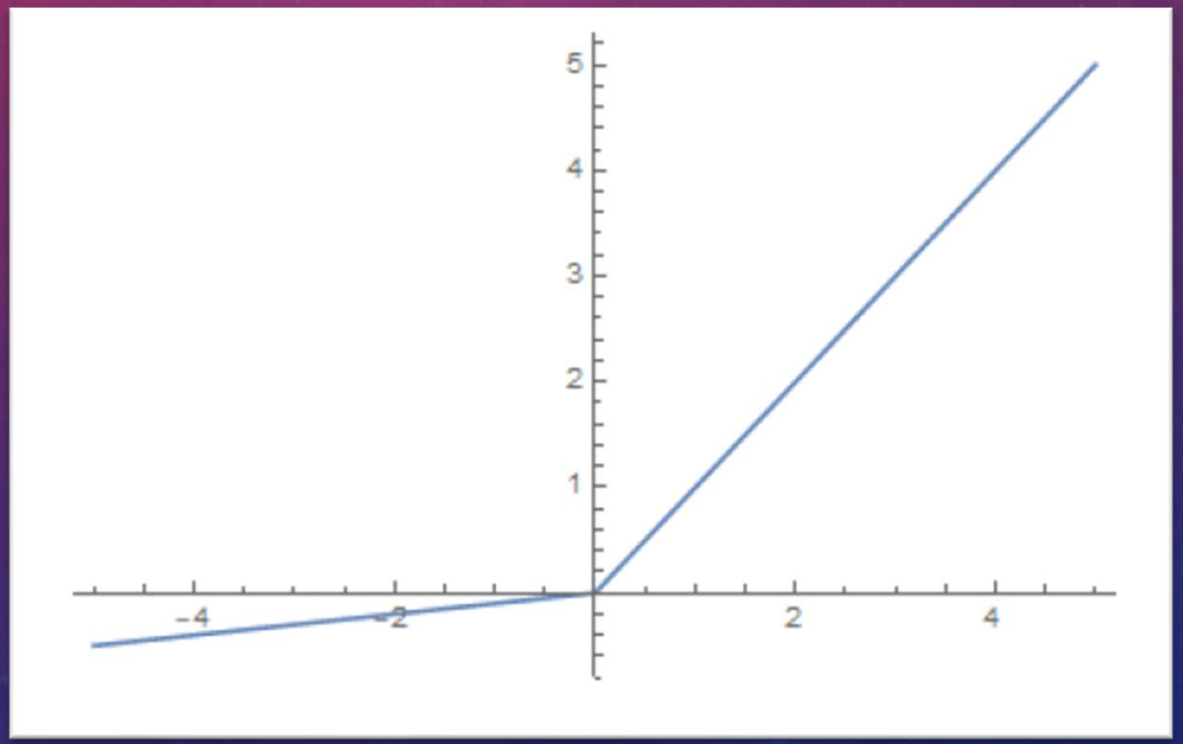




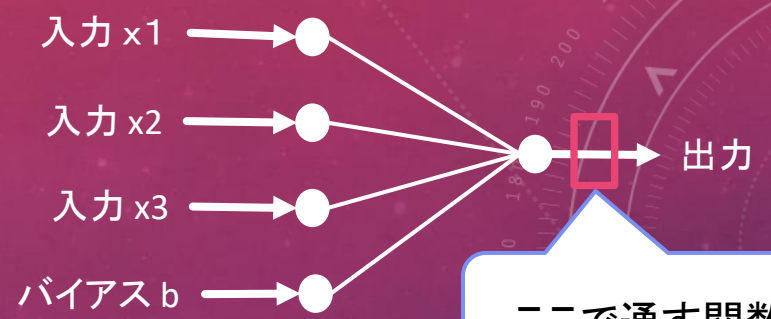
Leaky ReLU関数



- 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

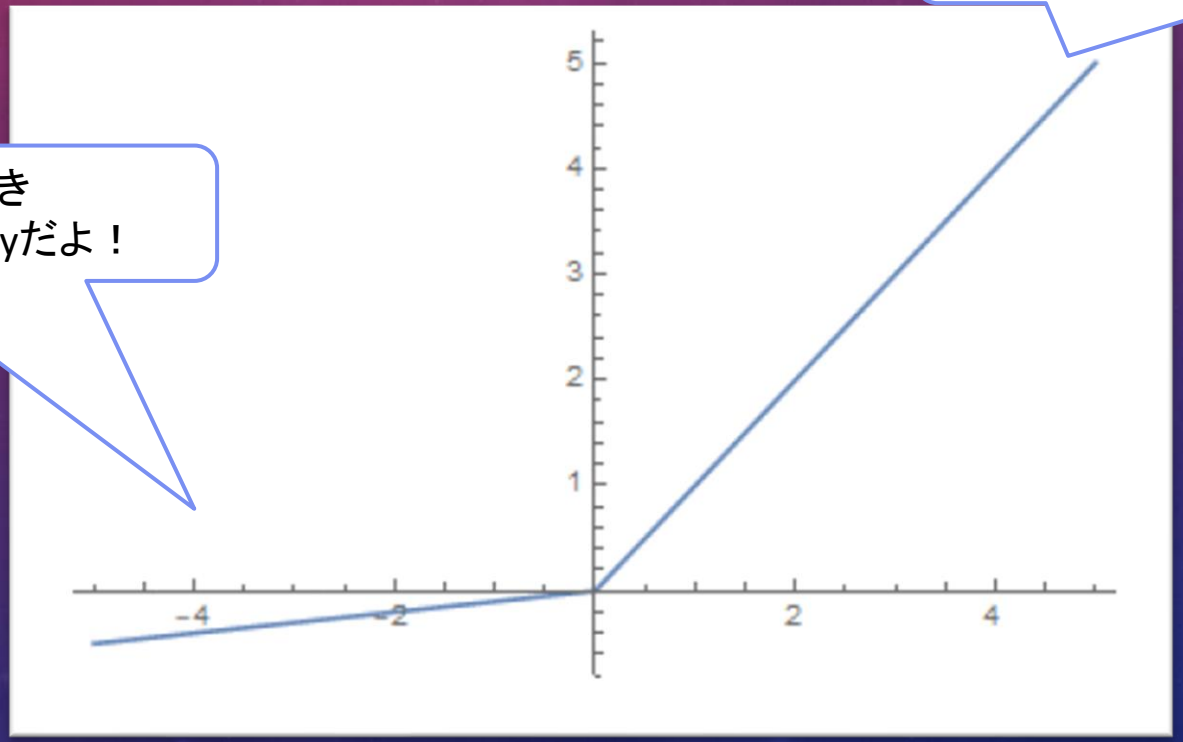


ここで通す関数だよ！

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

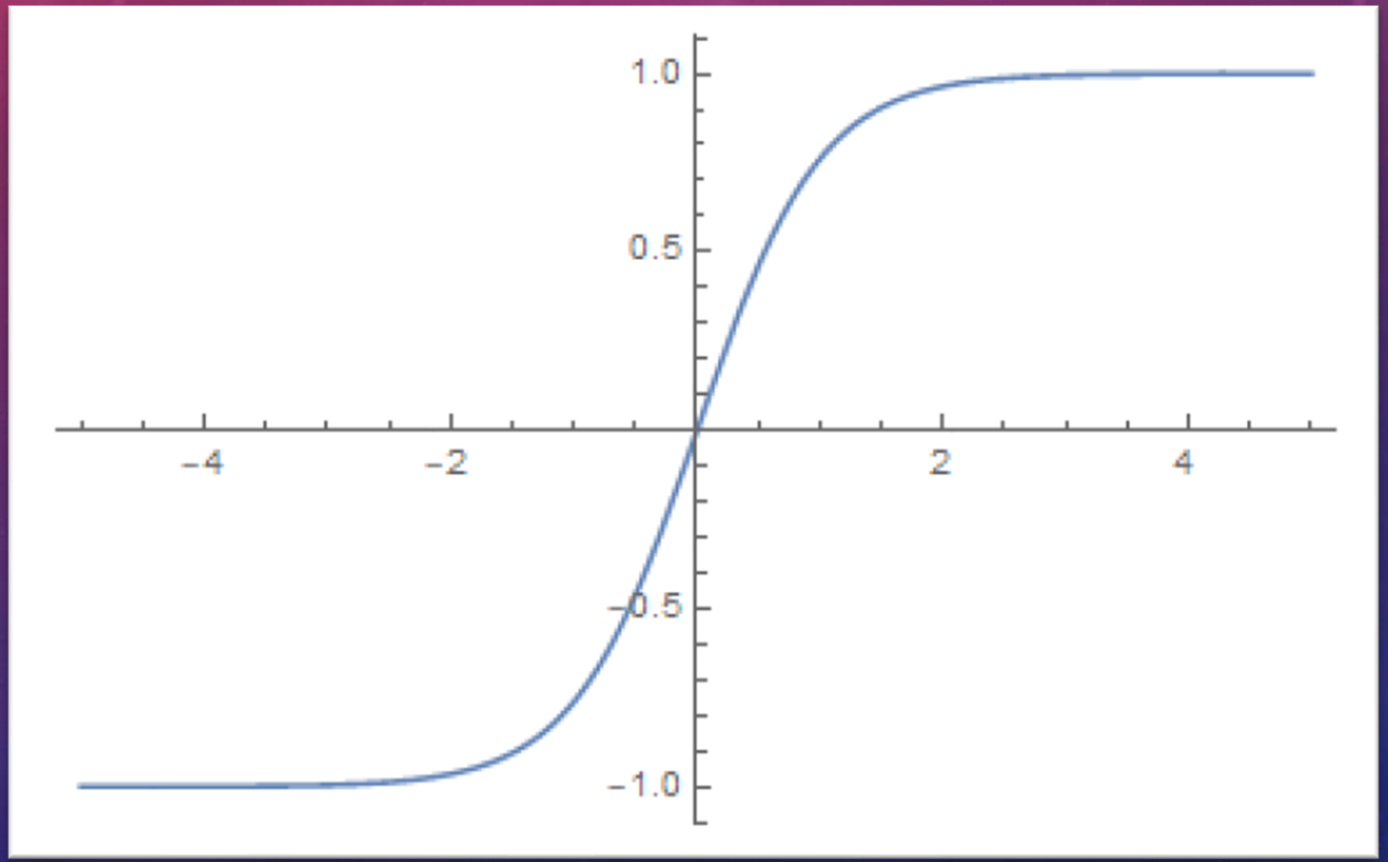
ReLU関数の代わりに使われるよ

入力Xがマイナスのとき
ちょっとだけ漏らすから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

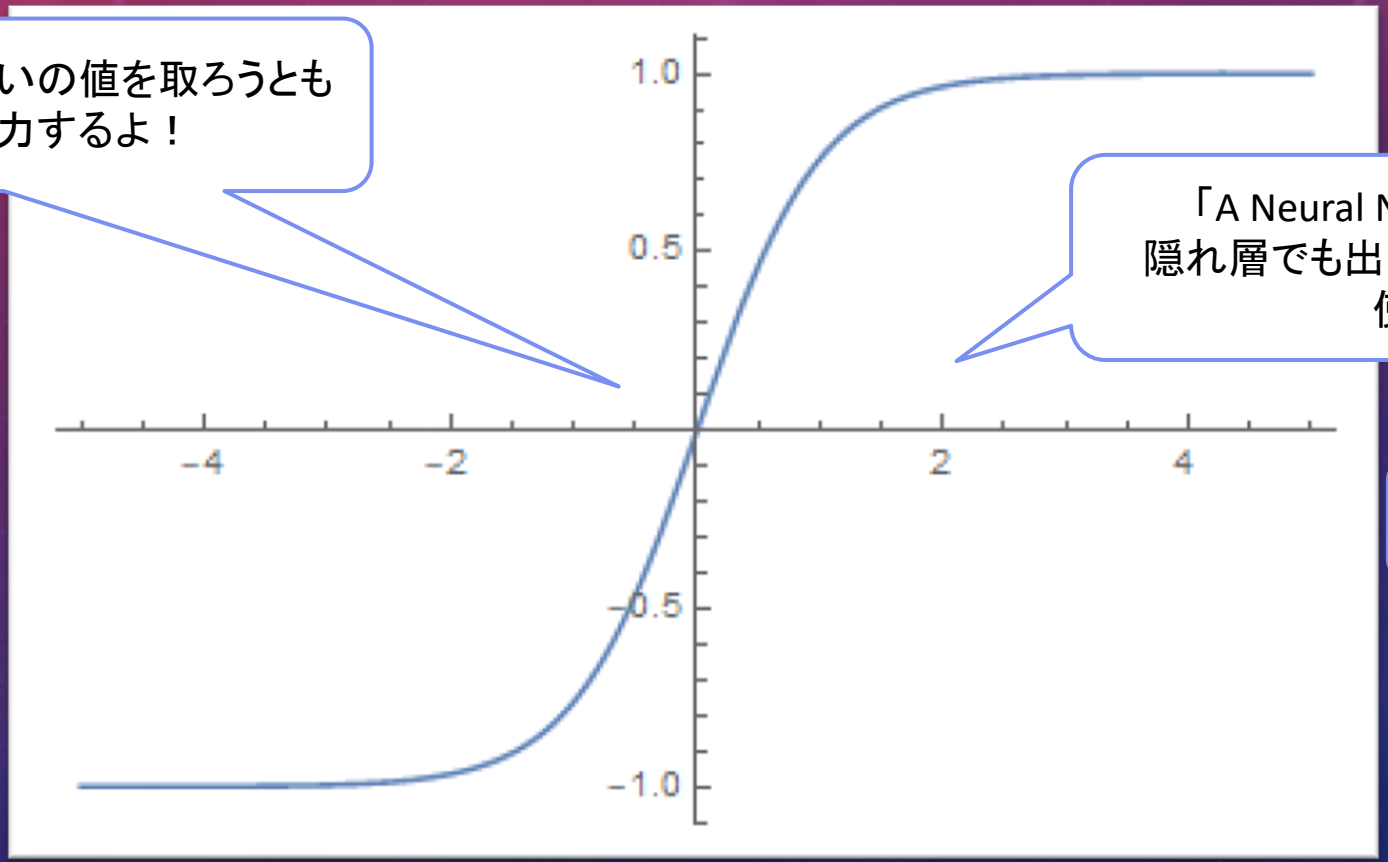


Tanh（ハイパボリックタンジェント）関数

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

ここで通す関数だよ！

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！

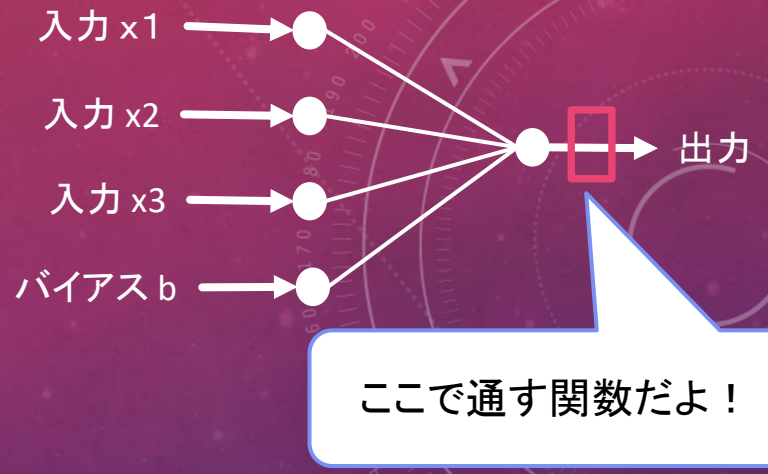


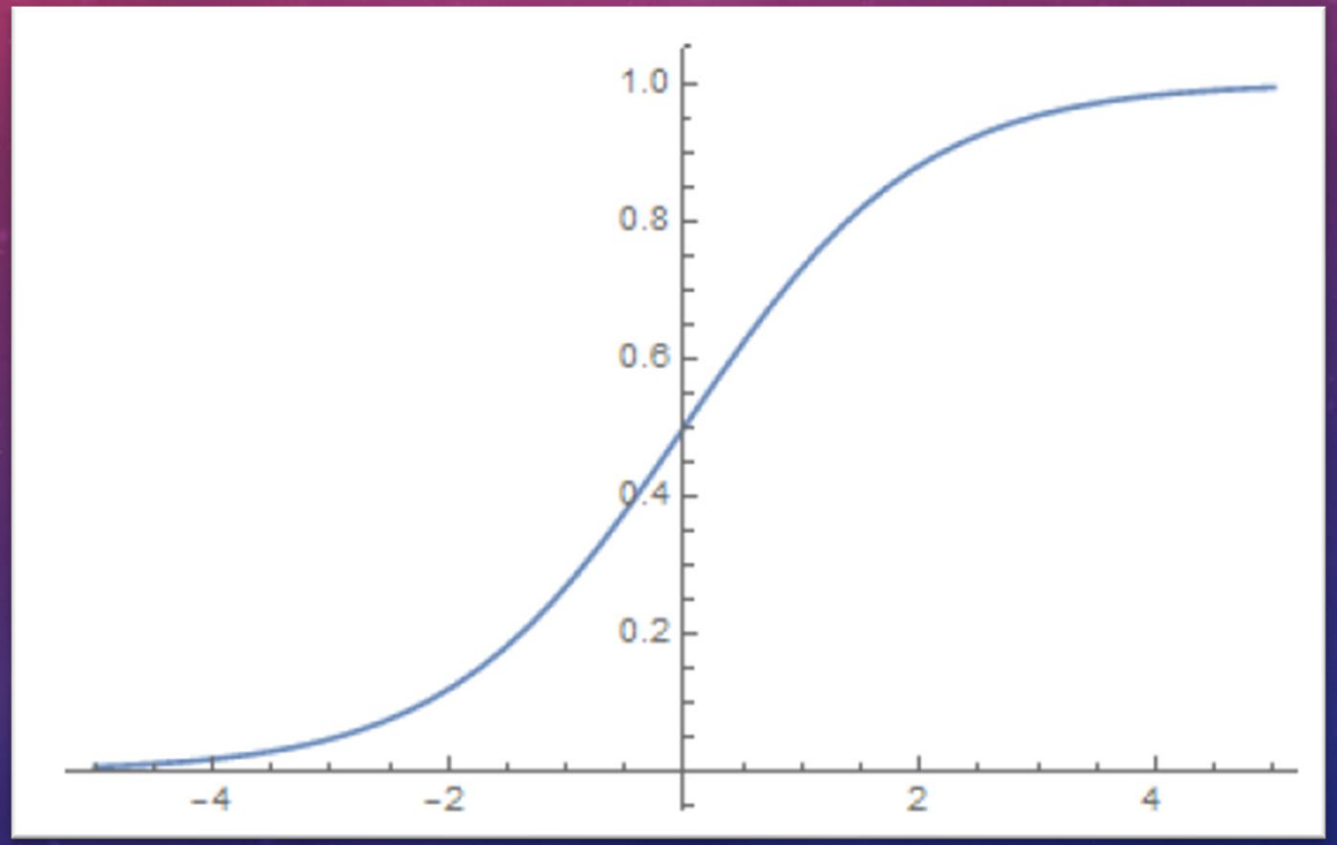
「A Neural Network Playground」では
隠れ層でも出力層でもデフォルトでこれが
使われているよ

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

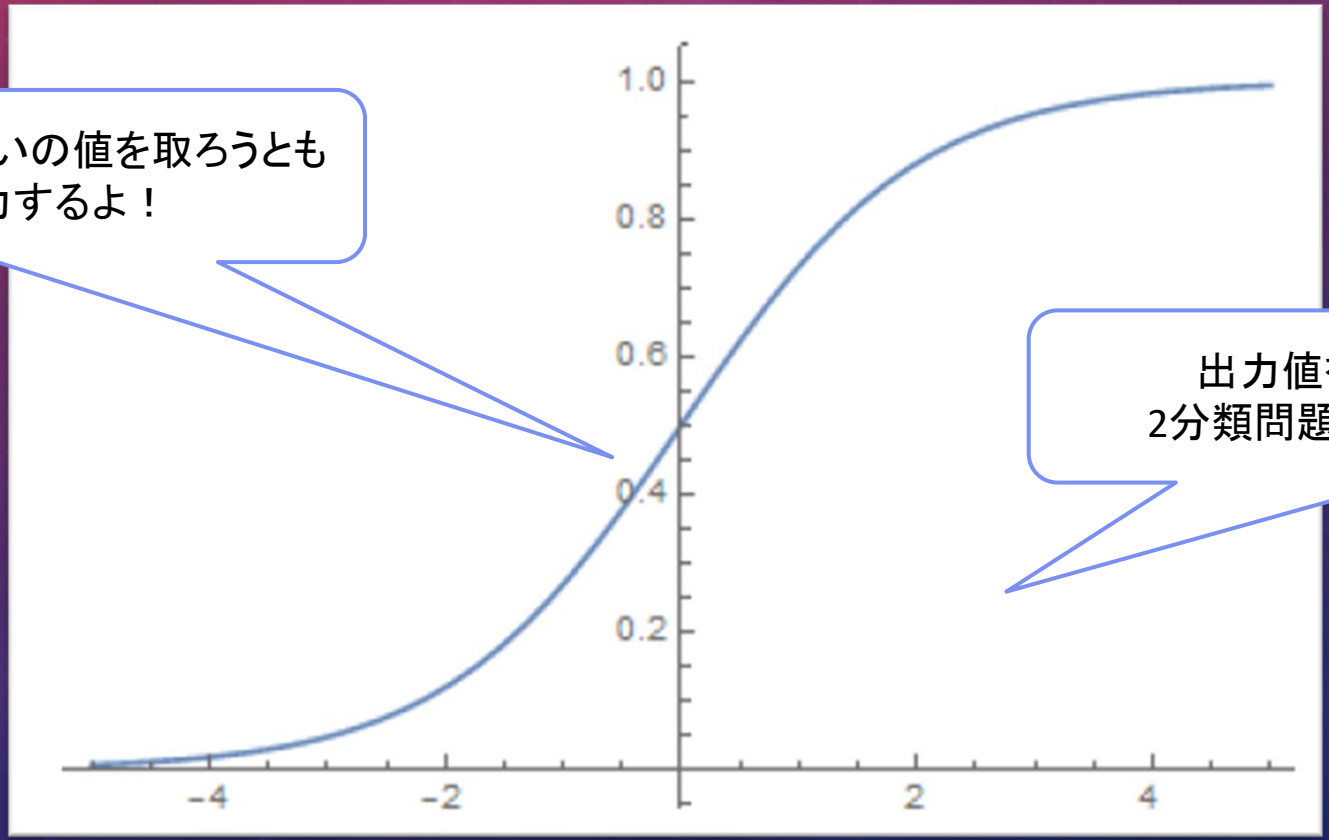




シグモイド関数

- $F(x)=\frac{1}{1+e^{-x}}$ を通したものを出力

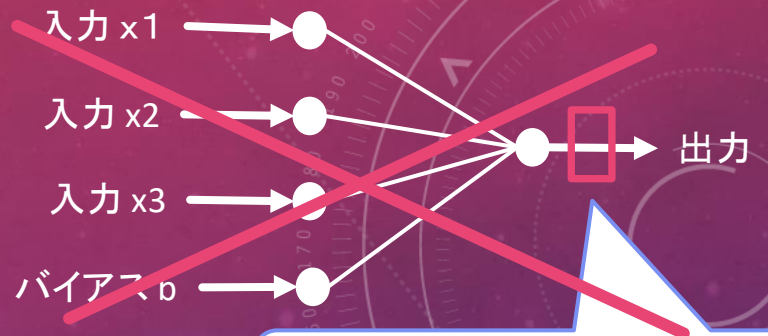
出力は入力（x）がどれぐらいの値を取ろうとも
0～1の範囲で出力するよ！



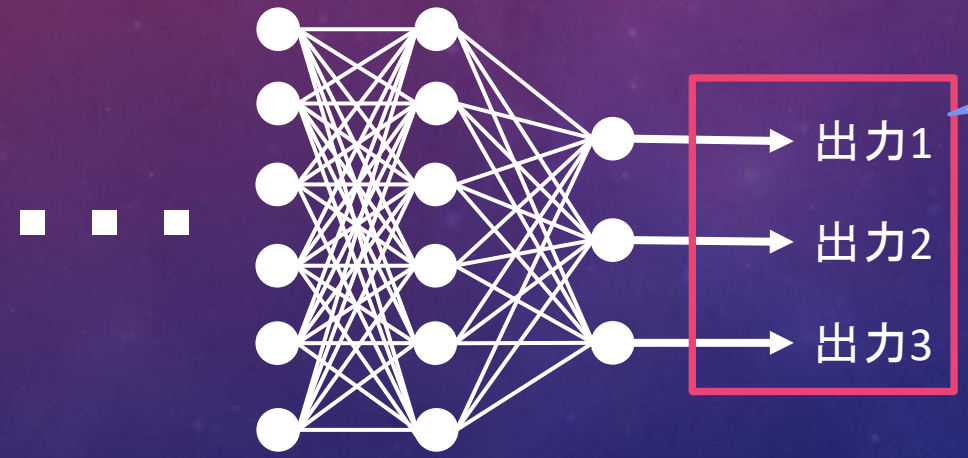
出力値を確率として扱えるから
2分類問題の出力層に使われるよ！

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
- $[y_0, y_1, \cdots, y_{k-1}] = \frac{[e^{x0}, e^{x1}, \cdots, e^{xk-1}]}{e^{x0} + e^{x1} + \cdots + e^{xk-1}}$



ノード1個では意味ないよ！

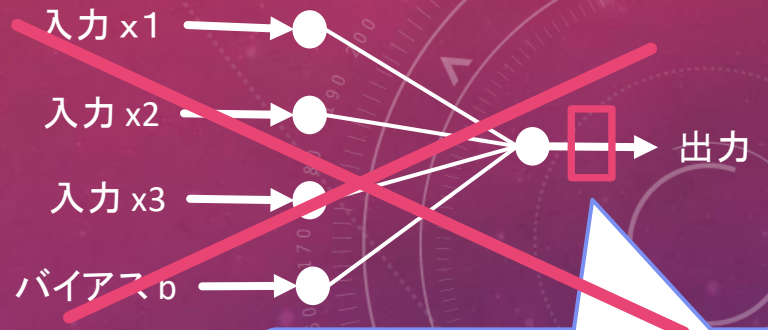


出力の合計が1、つまり合計100%の確率として扱えるよ！

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！

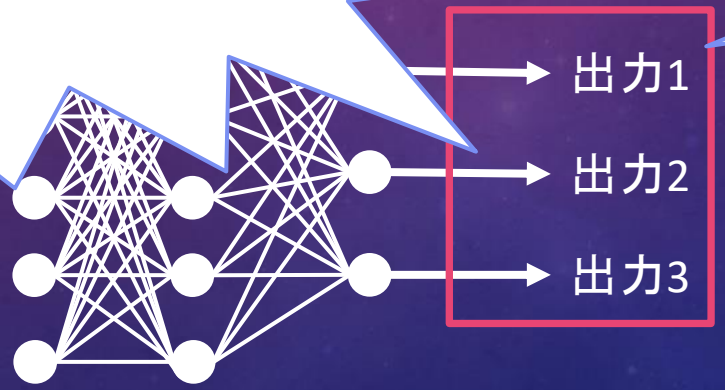
ソフトマックス関数

- 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
- $\left[\frac{e^{x0}}{e^{x0}+e^{x1}+\cdots+e^{xk-1}}, \frac{e^{x1}}{e^{x0}+e^{x1}+\cdots+e^{xk-1}}, \cdots, \frac{e^{xk-1}}{e^{x0}+e^{x1}+\cdots+e^{xk-1}}\right]$



ノード1個では意味ないよ！

難しいので詳しくは次回！

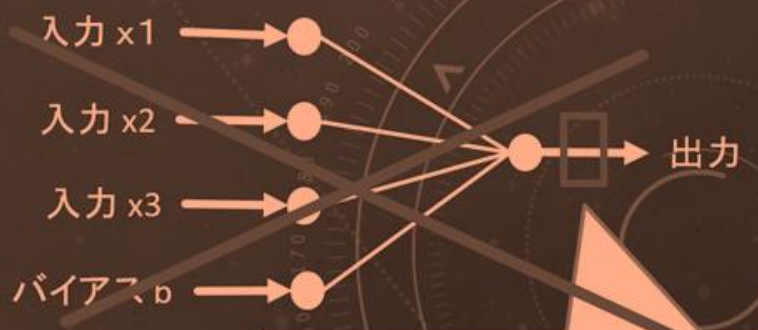


出力の合計が1、つまり合計100%の確率として扱えるよ！

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！

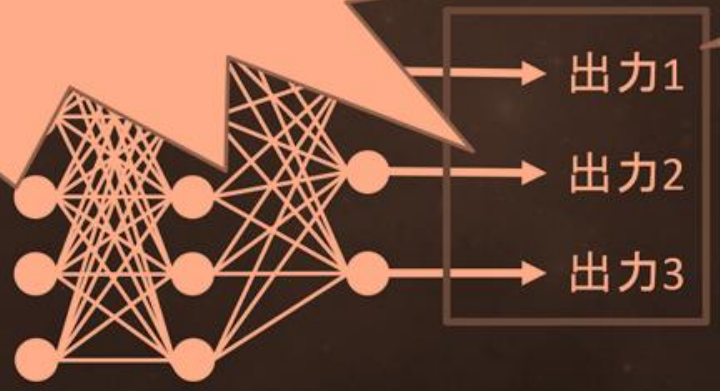
ソフトマックス関数

- 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
- [ ] $\frac{[e^{x0}, e^{x1}, \cdots, e^{xk-1}]}{e^{x0}+e^{x1}+\cdots+e^{xk-1}}$



ノード1個では意味ないよ！

難しいので詳しくは次回！

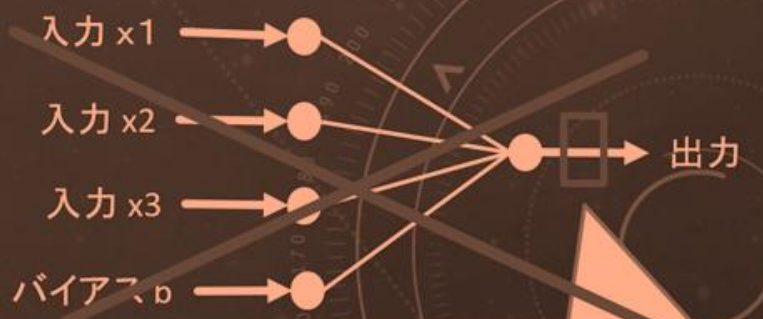


出力の合計が1、つまり合計100%の確率として扱えるよ！

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

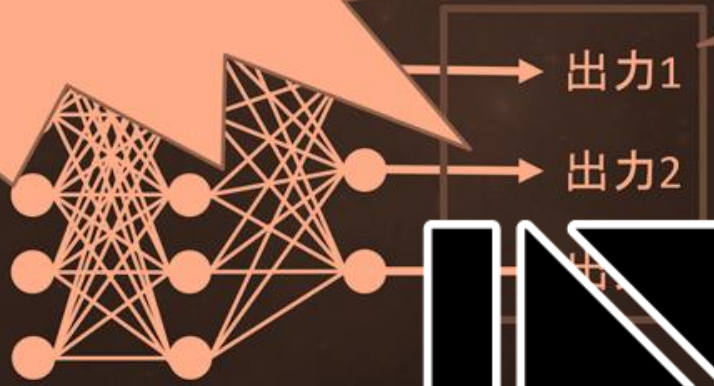
- 複数分類のとき出力層で使われる関数
- 出力の合計値が1になる
- $[\quad] \rightarrow \frac{[e^{x0}, e^{x1}, \cdots, e^{xk-1}]}{e^{x0}+e^{x1}+\cdots+e^{xk-1}}$



ノード1個では意味ないよ！

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）