AITCオープンラボ
身近になったAI開発シリーズ２

Neural Network Console 「初級編」

ジャパンシステム株式会社
チーフテクノロジーアドバイザー
吉田裕之

2021年4月15日
2021年6月11日改訂

本資料はジャパンシステム株式会社の社内向け教育資料をAITCオープンラボ向けに提供するものです。
本資料の全部または一部を許可なく複製・再配布・販売することはご遠慮ください。

目次

準備

NNCのセットアップ

- https://support.dl.sony.com/docs-ja/セットアップ/
- 「１セットアップ」「２アクティベート」を指示通り実施してください
- このページの説明は最新の2.00に未対応なので、表示画面が多少異なっています

- GPUをお使いの方は「３GPU環境のセットアップ」を実施してもかまいませんが、本勉強会ではGPU環境に関してのサポートはできません

注意事項

- NNCは、日本語を含む名前のフォルダでは動作しません
  - NNCをインストールするフォルダ
  - NNCのデータセットを置くフォルダ
  - NNCのプロジェクトを保存するフォルダ
- プロジェクト保存フォルダは書き込み可能である必要があります
  - サンプルプロジェクトはインストールフォルダの中にあるので、インストールフォルダが書き込み可能でないと、サンプルが実行できません
  - C:Program Filesは管理者でないと書き込めないことがあります
- おすすめの使い方
  - 解凍した「neural_network_console」フォルダをデスクトップに置く
  - その中に「myproject」フォルダを作り、データセットやプロジェクトはそこに保存

はじめに

勉強会の進め方

- 勉強会各回の最初の30分は前回の復習時間とします
（初回は進め方やNNCセットアップの確認）
各自で、前回に作成したプロジェクトを起動して動作を確認してください、質問があれば遠慮なくどうぞ
- オンラインハンズオンなので、皆さんはZoomとNNCの２つの画面を交互に見る必要があります。
２つの画面は「Alt+Tab」を使って交互に切り替えできます。
- 皆さんが順調なのか雰囲気を知りたいので、周囲の雑音が気にならない環境であれば、マイクONにして気軽につぶやいてください
「それどこ？」とか、「ちょっと待って」とか。
（レコーディングはしません）

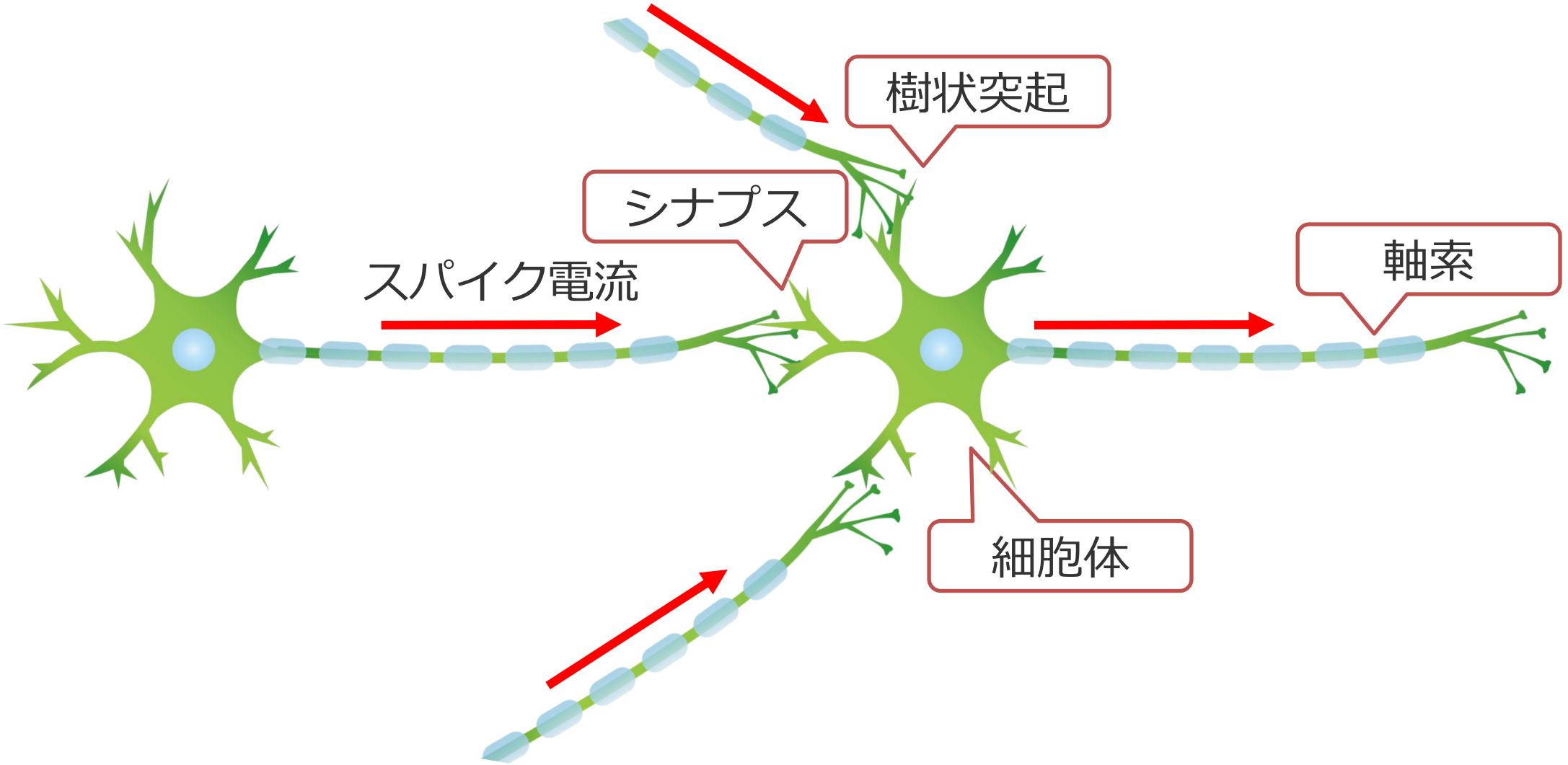
勉強会の進め方（つづき）

- 内容について質問があれば「手を挙げて」ください
  指名しますのでマイクONにして発言してください
  質疑の最中に別の方の手が挙がったら、早めに切り上げましょう
- 指示された作業がうまくできたら「親指を上げて」ください
  皆さんの親指を見て、先に進みます
- NNCの動作上の問題がある場合には、全員向けチャットに「Help」と書いてください。サポータが対応します
  画面共有が必要な場合には、ブレークアウトルームへ移動して個別に対応します

セットアップの確認

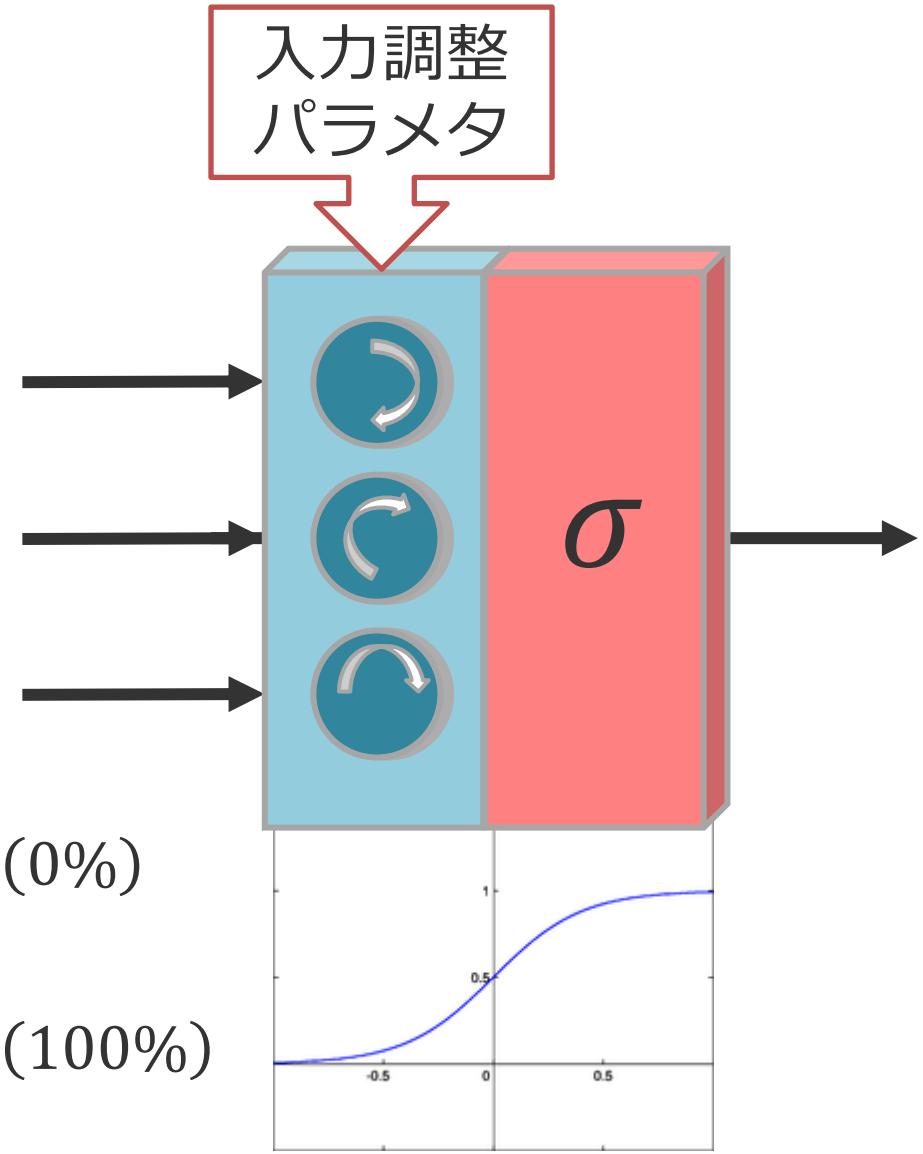
ニューラルネットワークとは？

神経回路網のしくみ



ニューロンのモデル

① 入力データを受ける
② 各入力データの大きさを
パラメタで調整
③ 調整したデータを足し算
④ シグモイド関数σで出力を決定



$$\sigma(x) = \begin{cases} x \text{ が大きな負の数の時は } 0.0\ (0\%) \\ x \text{ が } 0 \text{ の時は } 0.5\ (50\%) \\ x \text{ が大きな正の数の時は } 1.0\ (100\%) \end{cases}$$

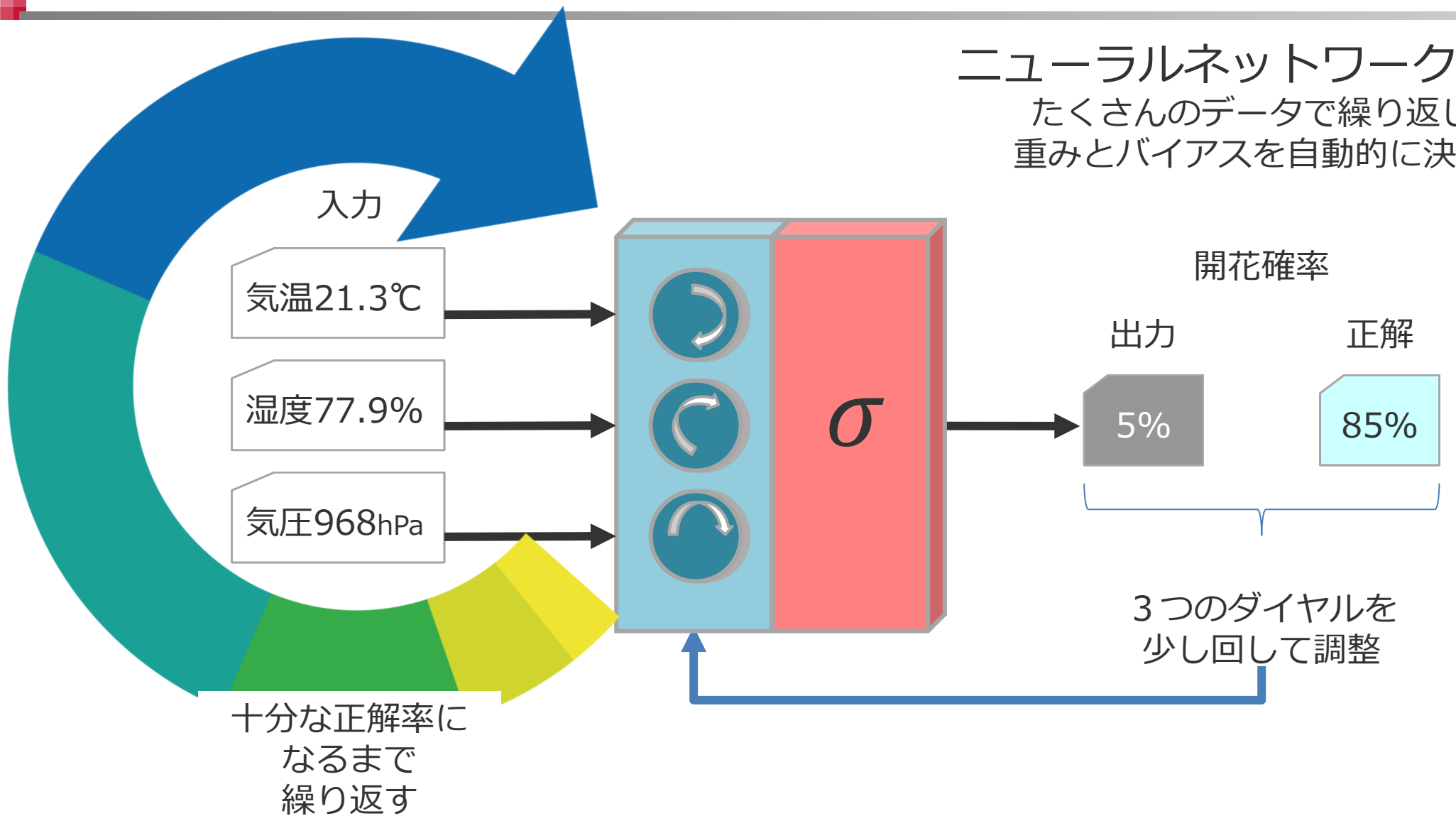
調整パラメタの例

		重みW		バイアスb		
気温：21.3	×	1.0	−	20.0	=	1.30
					+	
湿度：75.5	×	0.02	−	1.0	=	0.51
					+	
気圧：968	×	0.01	−	10.0	=	-0.32
					‖	
					1.49	

桜が咲く確率

$\sigma(1.49)$=81.61%

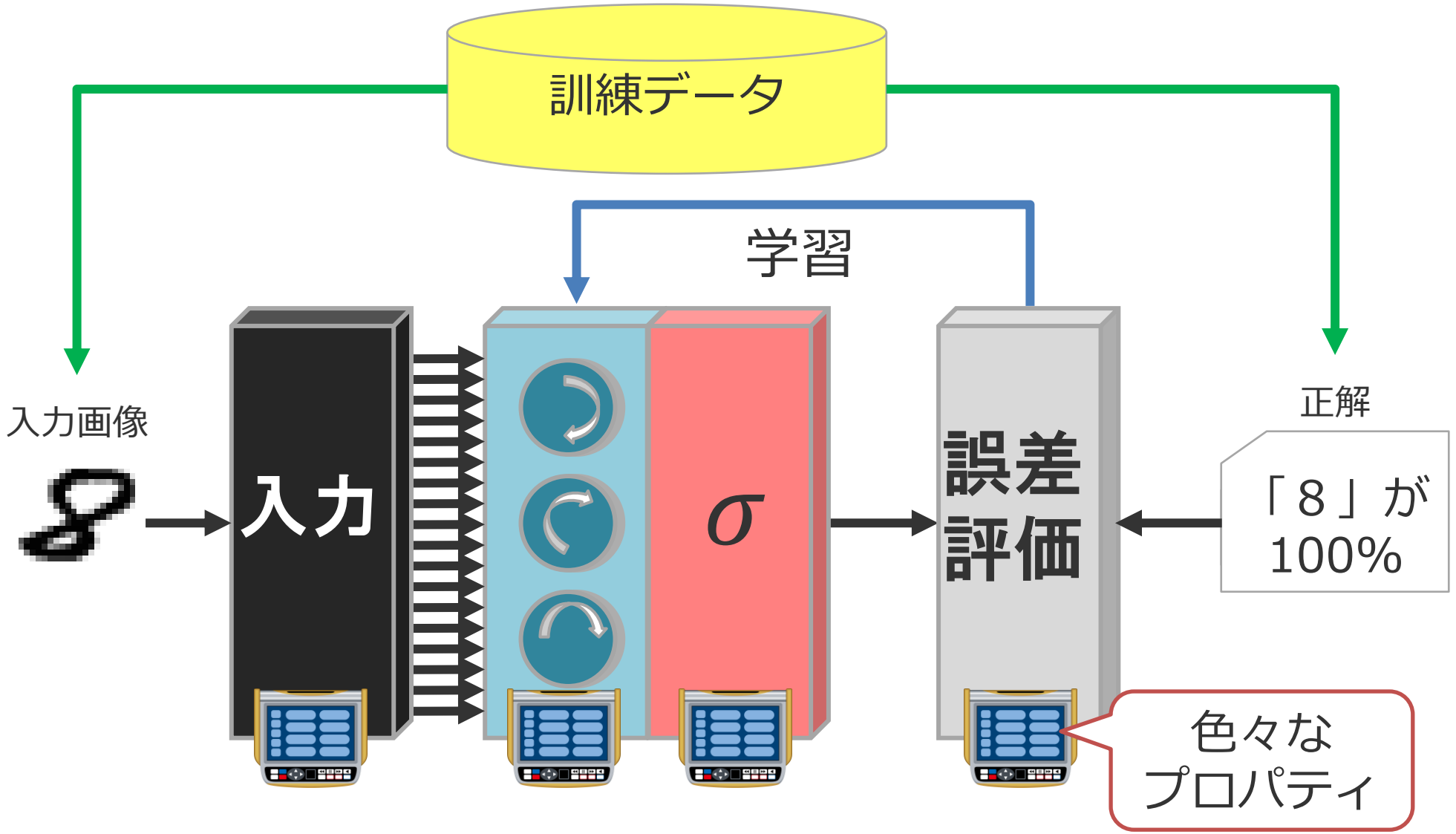
学習の原理



ニューラルネットワークの学習とは

たくさんのデータで繰り返し調整して
重みとバイアスを自動的に決定すること

レイヤー部品を使ったネットワーク設計（画像認識の例）

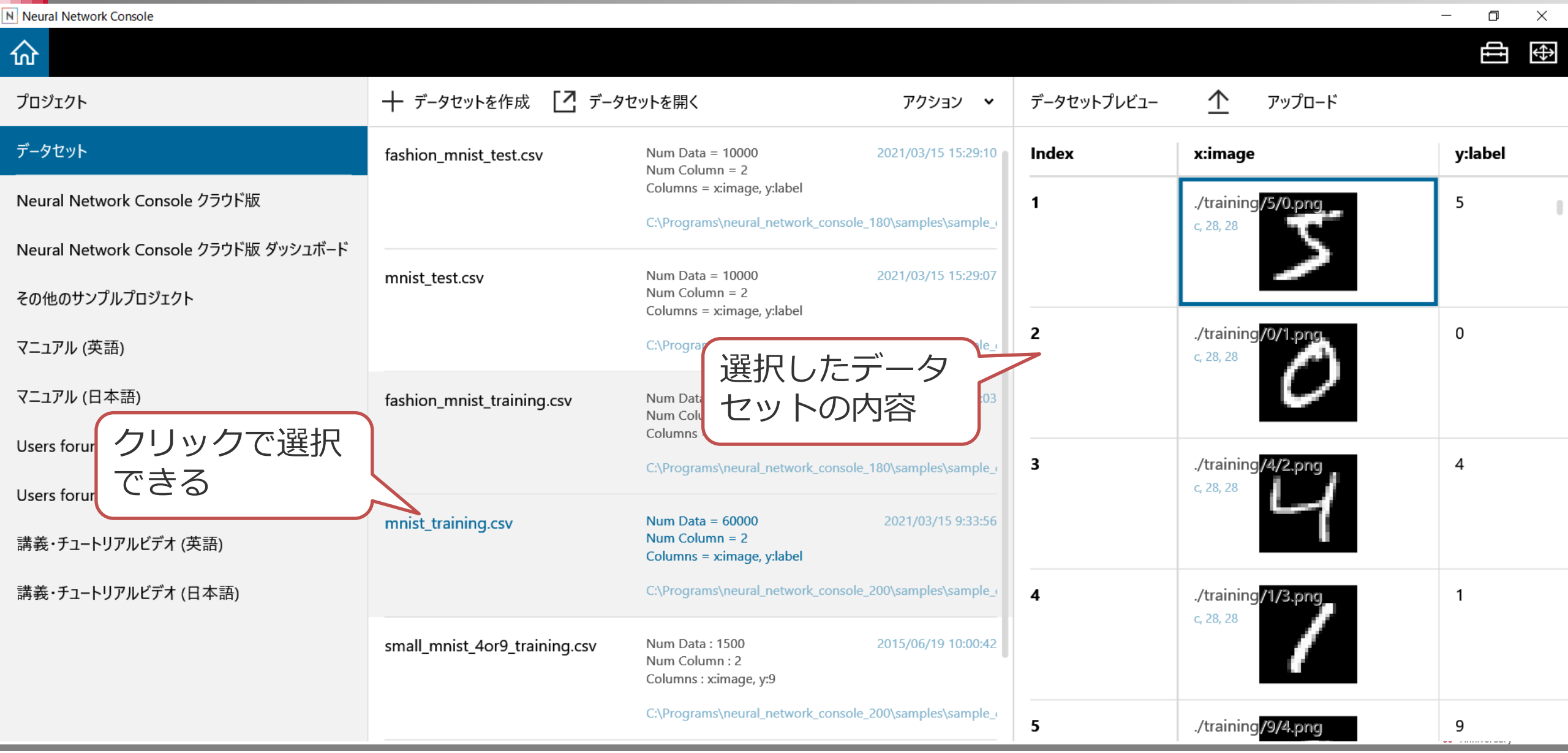


Sony Neural Network Console

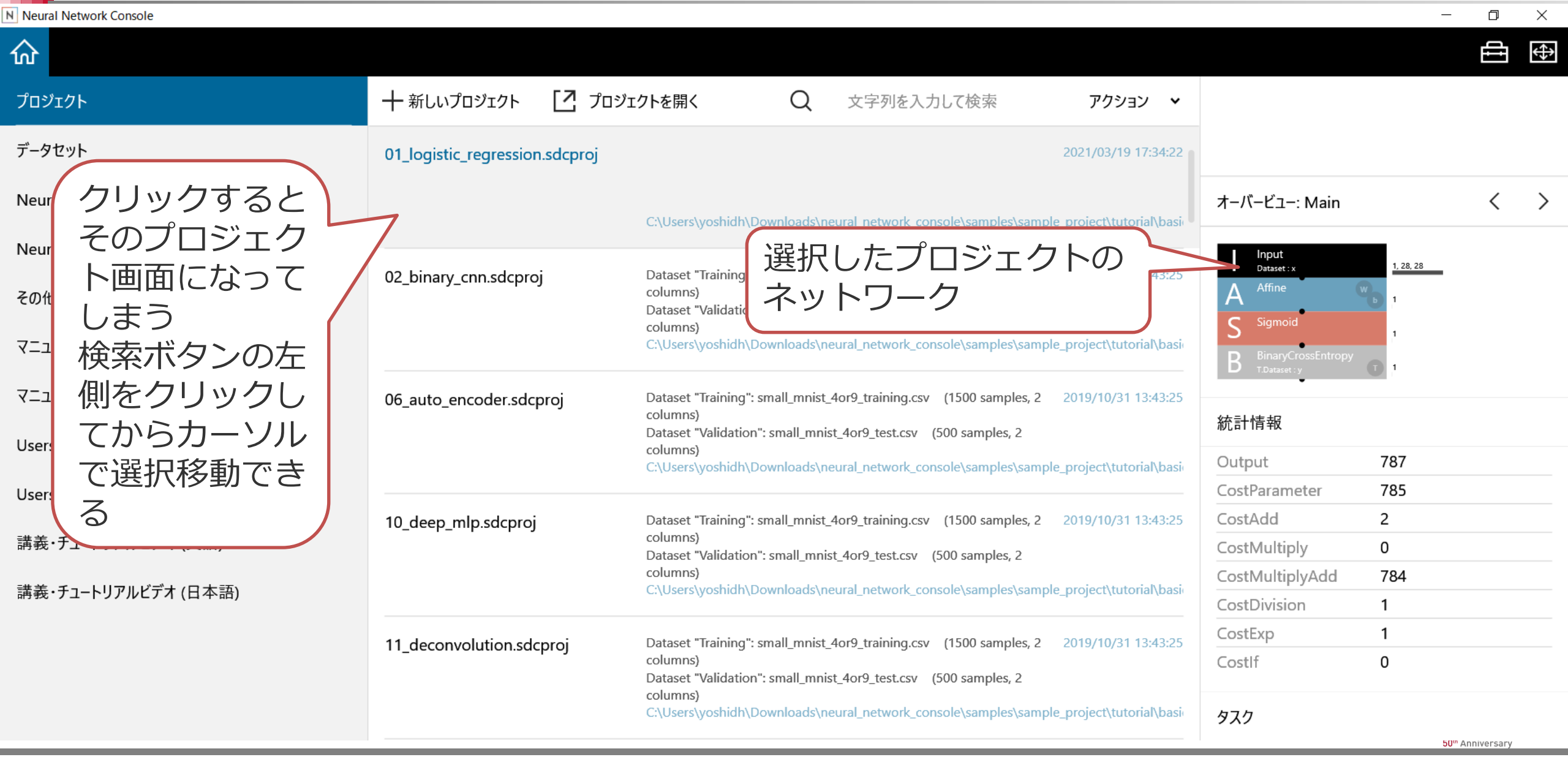
Sony Neural Network Console(SNNC)の一押しポイント

- レイヤー部品（コンポーネント）を使った設計を**GUIでできる**
- レイヤー部品（コンポーネント）がたくさんある
- Windowsで動く（GPUも利用可）
- Cloud版があるので、本気で使う時に便利、学習時のみ課金
- **学習経過がグラフ化されて楽しい**
- **学習履歴が管理できて色々試すのに便利**
- **学習結果をPowerPoint資料やhtmlページにしてくれる**
- 設計したネットワークをPythonコードに➡アプリに組み込める

ホーム画面（データセットタブ）



ホーム画面（プロジェクトタブ）



クリックするとそのプロジェクト画面になってしまう
検索ボタンの左側をクリックしてからカーソルで選択移動できる

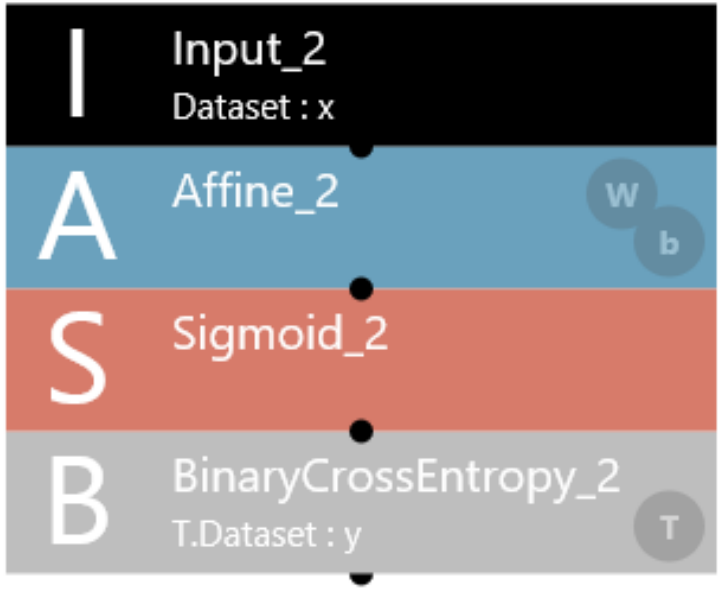
選択したプロジェクトのネットワーク

プロジェクト画面（編集タブ）01_logistic_regression

【ハンズオン】

同じネットワークを作ってみる：

1. 「コンポーネント」の「Input」をダブルクリック
2. 「文字列を入力して検索」に「af」と入力
3. 「Affine」をダブルクリック
4. 「文字列を入力して検索」に「sig」と入力
5. 「Sigmoid」をダブルクリック
6. 「文字列を入力して検索」に「cross」と入力
7. 「BinaryCrossEntropy」をダブルクリック
8. 「Affine_2」をクリックし
9. 「レイヤープロパティ」の「OutShape」を「１」に変更



【ハンズオン】

Affine_2とSigmoid_2の間にDropoutを入れてみる：

1. マウスの範囲選択で「Sigmoid_2」と「BinaryCrossEntropy_2」を選択
2. 「Sigmoid_2」を下にドラッグして十分に離す
3. 「Affine_2」と「Sigmoid_2」の間の接続線をクリック
4. Delキーで接続を消去
5. 「文字列を入力して検索」に「dr」と入力
6. 「Dropout」を一覧からドラッグして、「Affine_2」の下にくっつける（一度マウスを離す）
7. 「Affine_2」の下の「Dropuout」をドラッグして「Sigmoid_2」にくっつける

【ハンズオン】

接続線の結線方法：

1. 「Affine_2」と「Dropout」の間の接続線をいったん消去
2. 「Affine_2」の下に付いているピンを「Dropout」の箱の中までドラッグ

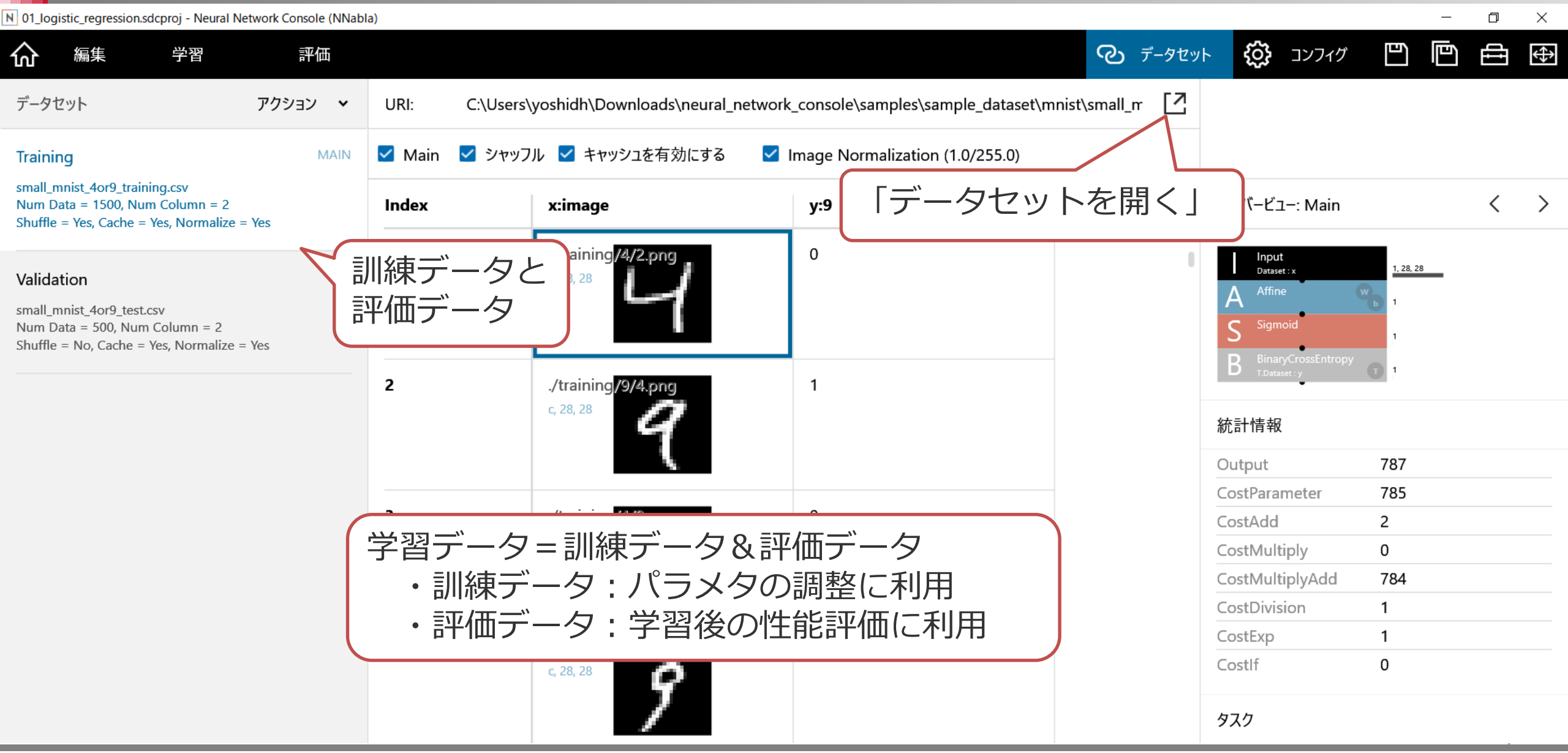
別の方法：

1. 「Affine_2」と「Dropout」の間の接続線をいったん消去
2. 「Affine_2」を選択した状態で
   Shiftキーを押しながら「Dropout」をクリック

自動整列：

1. ネットワークの近くの空白部分を右クリック「レイヤーの整列」

プロジェクト画面（データセットタブ）

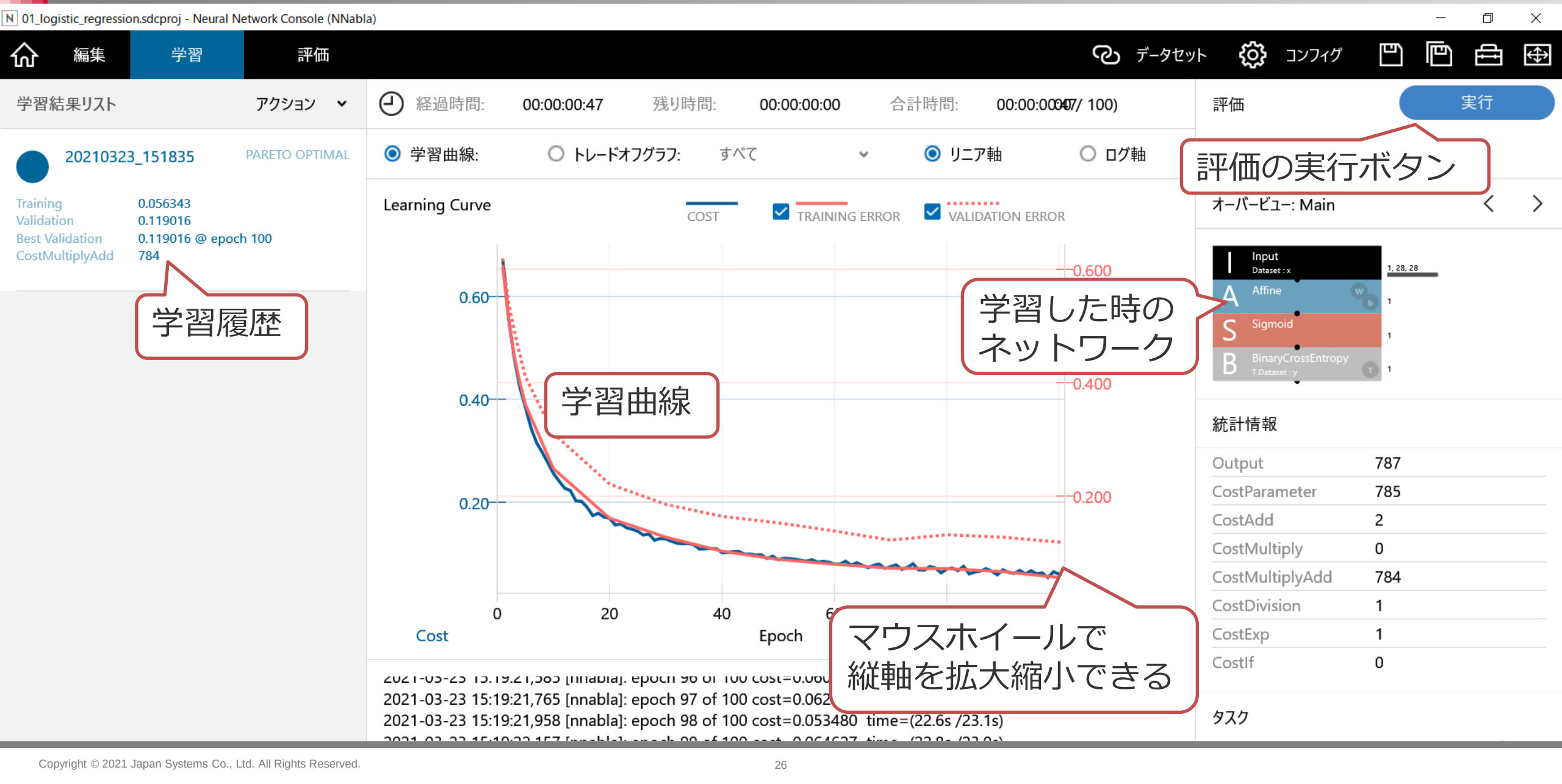


学習データ＝訓練データ＆評価データ
- 訓練データ：パラメタの調整に利用
- 評価データ：学習後の性能評価に利用

【ハンズオン】

1. 「編集」に戻り、追加したレイヤーを削除して元に戻す
   - 元のネットワークを壊してしまった場合は、
     保存しないでホーム画面に戻り、もう一度01_logistic_regressionを選択
2. 右上の青い「実行」ボタンの左が「学習」であることを確認
3. その下が「ローカルプロセッサ」であることを確認
   - 「ローカルプロセッサ」以外を選ぶと、クラウド上のGPUで実行（有料）
4. 「実行」ボタンをクリック

プロジェクト画面（学習タブ）



プロジェクト画面（評価タブ：出力結果）

01_logistic_regression.sdcproj - Neural Network Console (NNabla)

編集 学習 評価 データセット コンフィグ

学習結果リスト アクション

20210323_151835 PARETO OPTIMAL

Training	0.056343
Validation	0.119016
Best Validation	0.119016 @ epoch 100
CostMultiplyAdd	784

EVALUATED

評価履歴

経過時間: 00:00:00:06 残り時間: 00:00:00:00

出力結果 混同行列: y - y'

正解

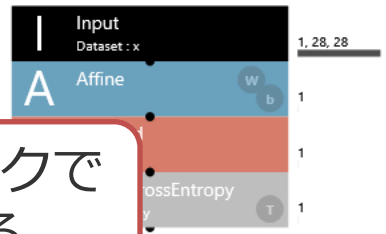
推論結果

Index	x:image	y:9	y'
1	... mnist\validation\4\4.png c, 28, 28 	0	0.045936447
2	... mnist\validation\4\6.png c, 28, 28 	0	
3	... mnist\validation\9\7.png c, 28, 28 	1	0.9954808
4	... mnist\validation\9\9.png	1	0.9806923

2021-03-23 15:27:19,154 [nnabla]: data 256 / 500
2021-03-23 15:27:19,242 [nnabla]: data 320 / 500
2021-03-23 15:27:19,322 [nnabla]: data 384 / 500

ダブルクリックでソートする

オーバービュー: Main



統計情報

Output	787
CostParameter	785
CostAdd	2
CostMultiply	0
CostMultiplyAdd	784
CostDivision	1
CostExp	1
CostIf	0

タスク

プロジェクト画面（評価タブ：混同行列）

01_logistic_regression.sdcproj - Neural Network Console (NNabla)

編集 | 学習 | 評価 | データセット | コンフィグ

学習結果リスト　アクション

20210323_151835　PARETO OPTIMAL

Training	0.056343
Validation	0.119016
Best Validation	0.119016 @ epoch 100
CostMultiplyAdd	784

EVALUATED

評価履歴

経過時間: 00:00:00:06　残り時間: 00:00:00:00　合計時間: 00:00:00:06/ 500)

出力結果　混同行列: y - y'　その他

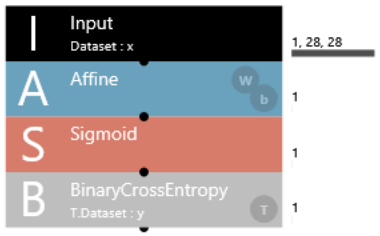
	y'=0	y'=1	Recall
y:9=0	238	12	0.952
y:9=1	12	238	0.952
Precision	0.952	0.952	
F-Measures	0.952	0.952	
Accuracy	0.952		
Avg.Precision	0.952		
Avg.Recall	0.952		
Avg.F-Measures	0.952		

混同行列

正解率

2021-03-23 15:27:19,154 [nnabla]: data 256 / 500
2021-03-23 15:27:19,242 [nnabla]: data 320 / 500
2021-03-23 15:27:19,322 [nnabla]: data 384 / 500

オーバービュー: Main

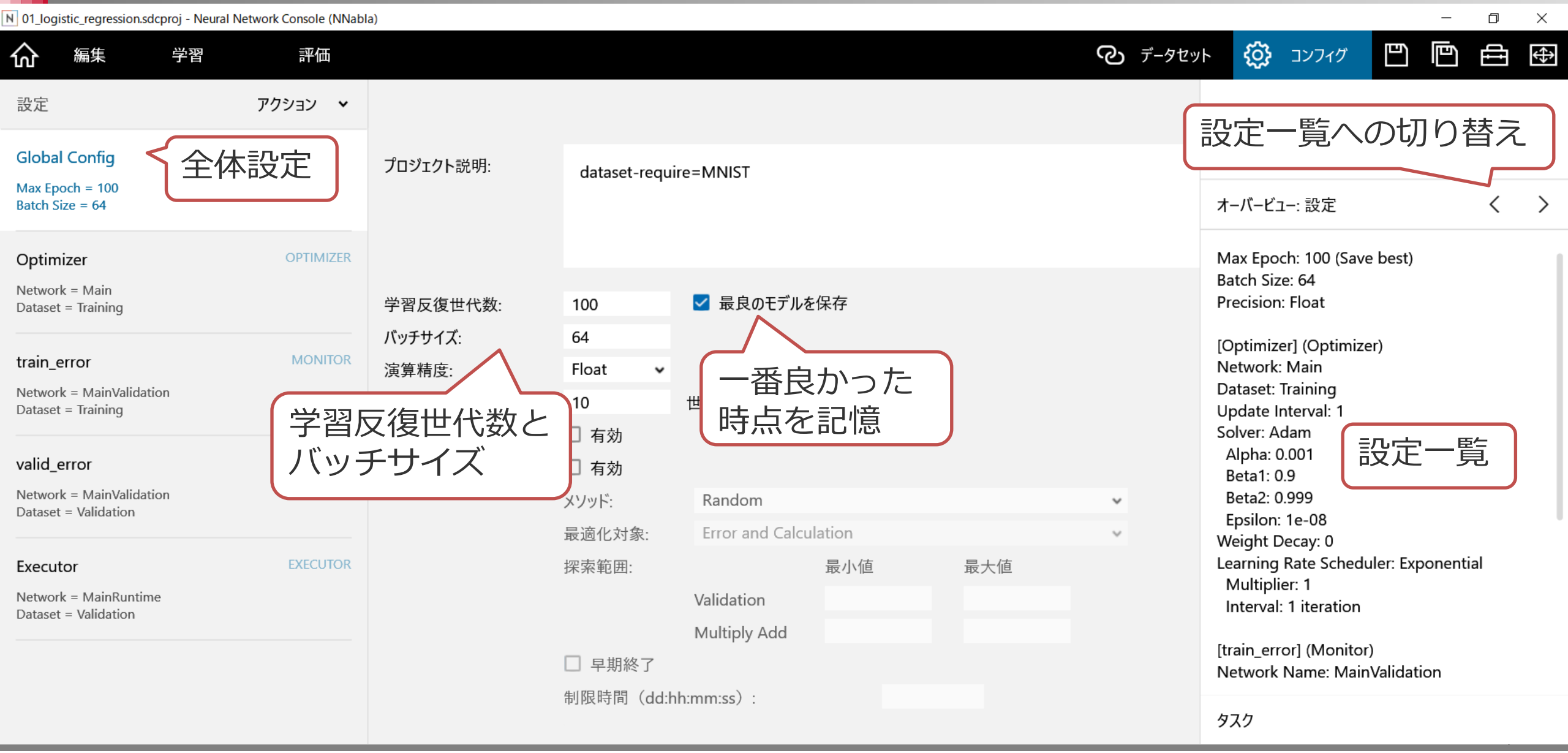


統計情報

Output	787
CostParameter	785
CostAdd	2
CostMultiply	0
CostMultiplyAdd	784
CostDivision	1
CostExp	1
CostIf	0

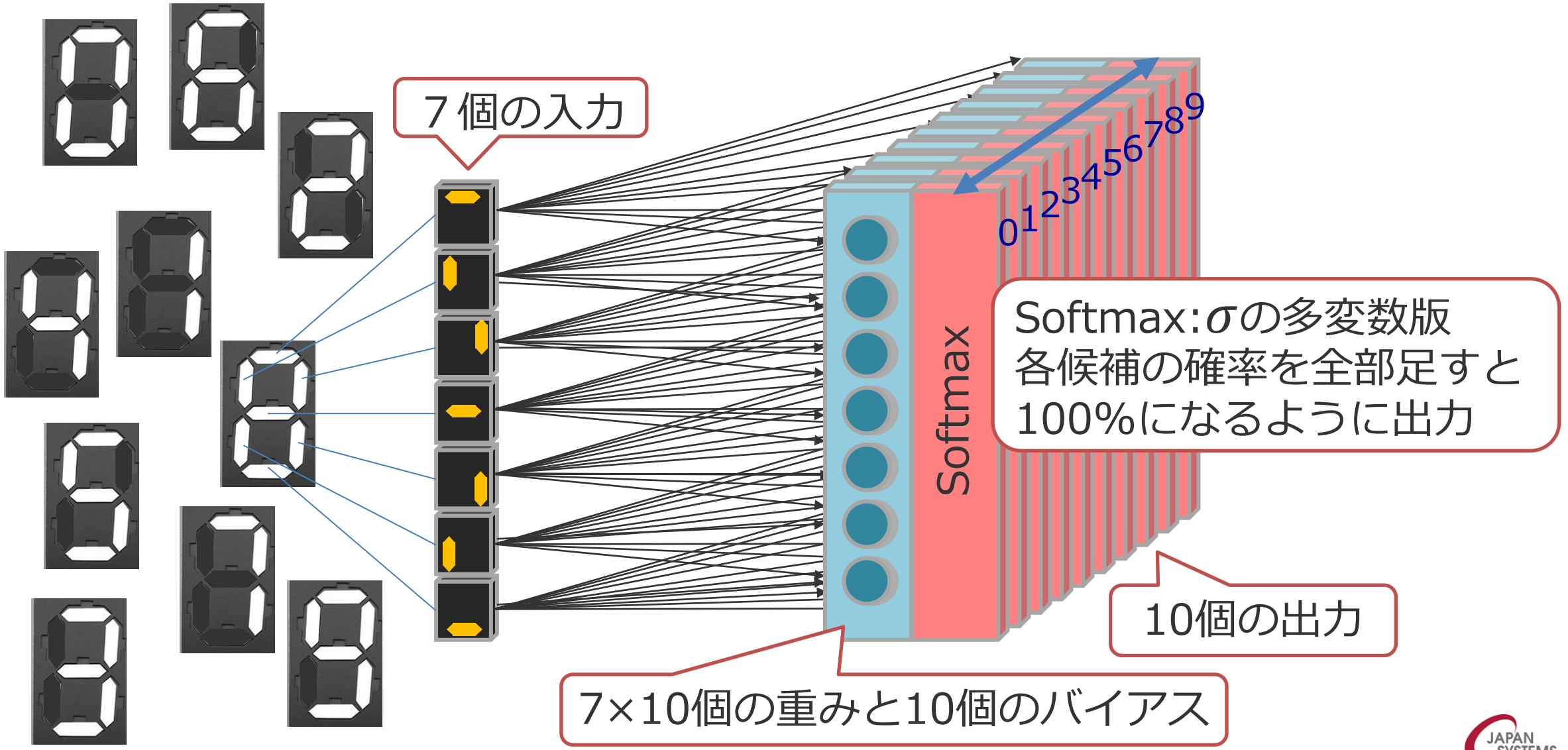
タスク

プロジェクト画面（コンフィグタブ：Global Config）



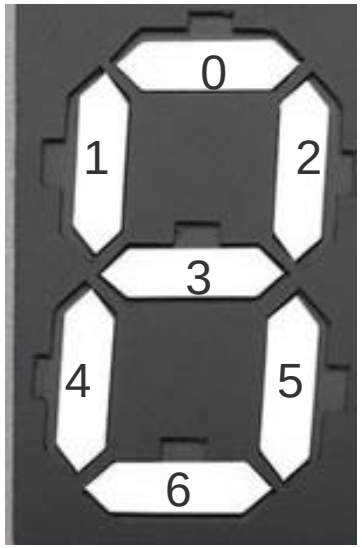
作ってみよう、LED数字の判定

適用例：LED数字の判定



【ハンズオン】LED数字判定のデータセットを作る

1. 配布したLEDDigit.csvの内容を確認




```
x__0,x__1,x__2,x__3,x__4,x__5,x__6,y
1,1,1,0,1,1,1,0
0,0,1,0,0,1,0,1
1,0,1,1,1,0,1,2
1,0,1,1,0,1,1,3
0,1,1,1,0,1,0,4
1,1,0,1,0,1,1,5
1,1,0,1,1,1,1,6
1,0,1,0,0,1,0,7
1,1,1,1,1,1,1,8
1,1,1,1,0,1,1,9
```

2. 「ホーム：データセット」の「データセットを開く」で LEDDigit.csvを開く

データセットLEDDigit.csvの内容

- Index：通し番号
- x__0, x__1, ..., x__6：入力配列xの７要素（「_」２個に注意）
  ※JavaやPython等で x[0] と書くものを、NNCでは x__0 と書く
- y：正解

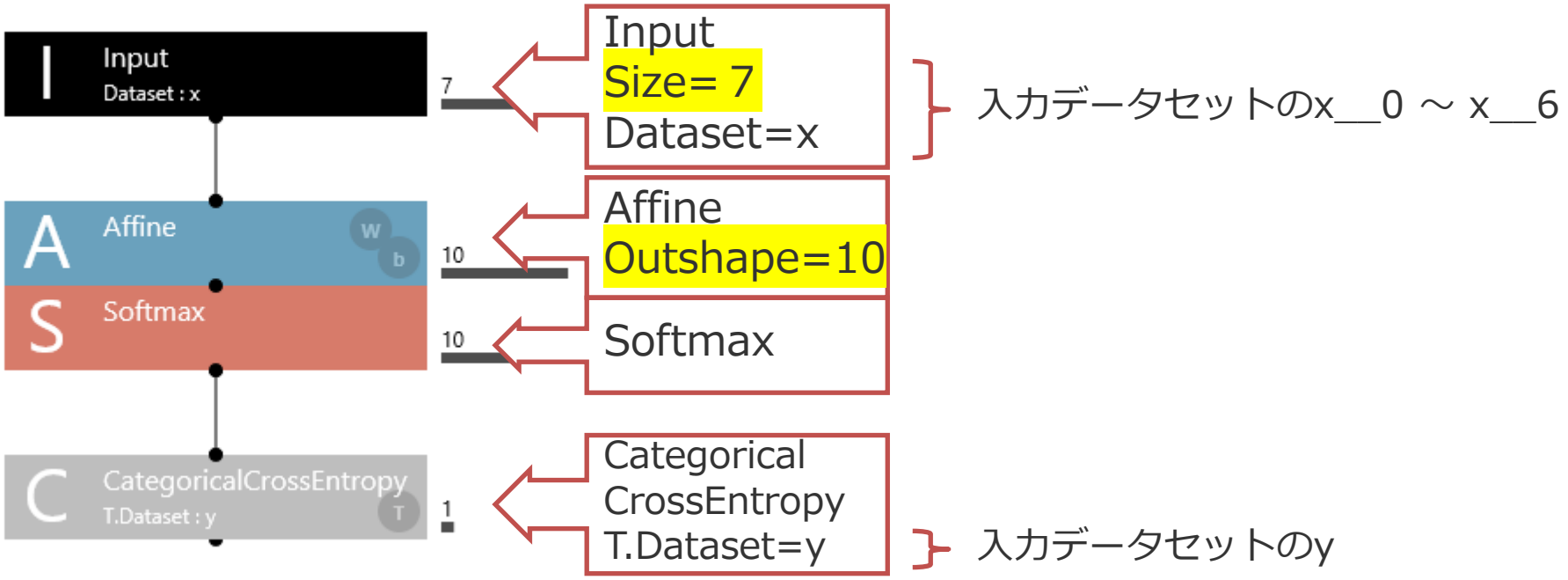
Index	x__0	x__1	…	x__6	y
1	1	1	…	1	0
2	0	0	…	0	1
…	…	…	…	…	…
10	1	1	1	1	9

【ハンズオン】LED数字判別の試作

1. 「ホーム：プロジェクト」で「新しいプロジェクト」
2. 「プロジェクト：データセットタブ」でTrainingを選択して
    でLEDDigit.csvを開く
3. Validationにも同様にLEDDigit.csvを設定
4. 「コンフィグ：Global Config」の「バッチサイズ」を10に設定

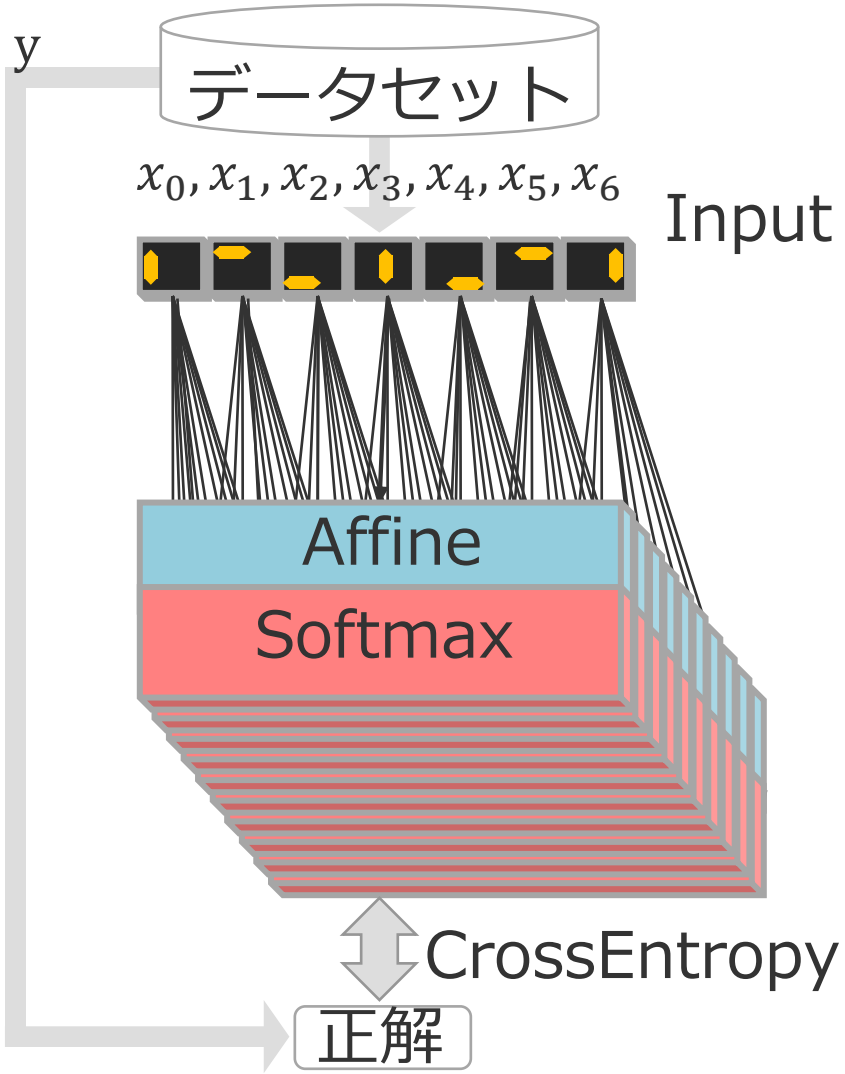
【ハンズオン】LED数字判別の試作

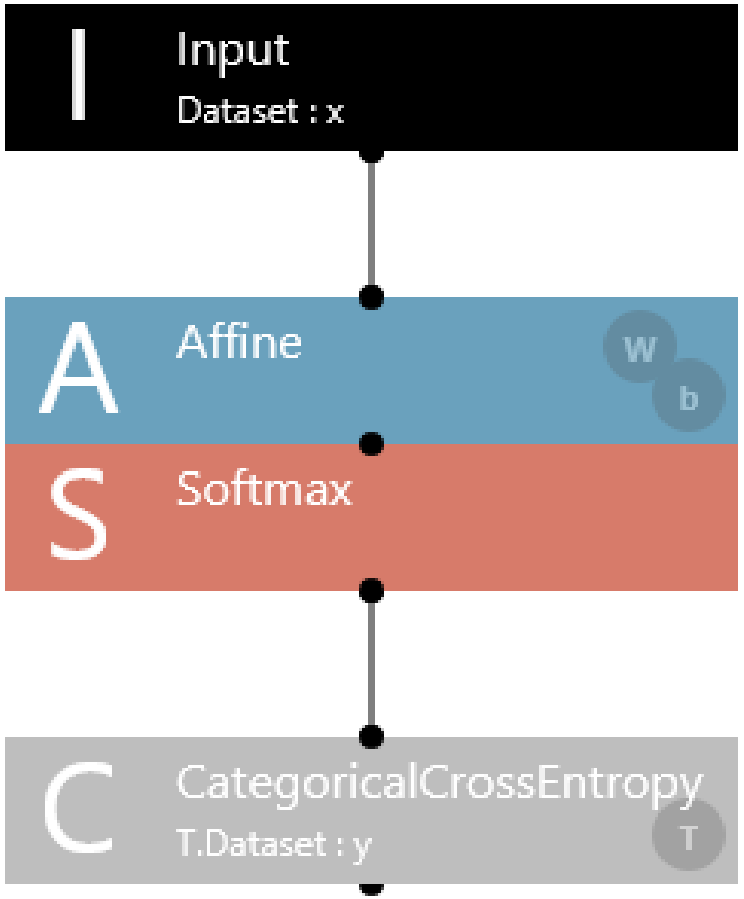
6. 「編集」で以下のネットワークを設計



7. 適当な場所（日本語パス不可！）に 💾「プロジェクトを保存」
8. 「ローカルプロセッサ」で学習を「実行」、評価を「実行」

ニューラルネットワークの構造





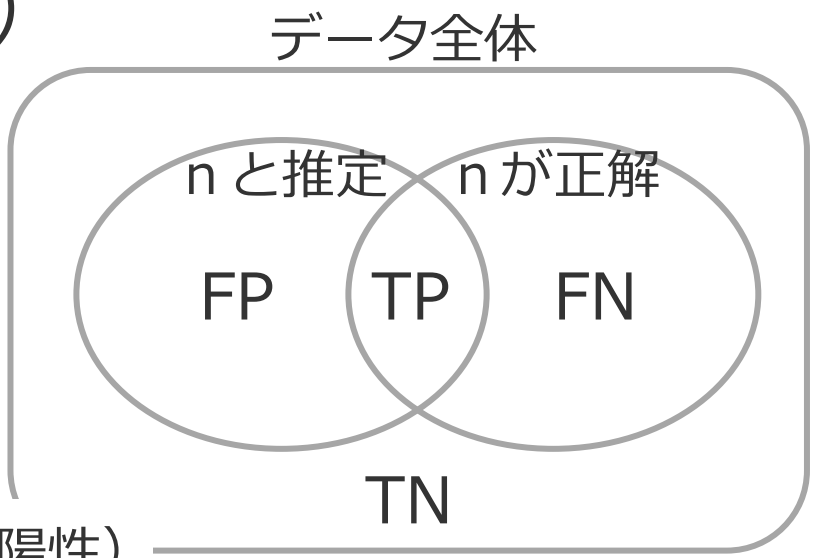
評価タブ：結果の見かた

- 出力結果
  - ✓ x__0～6、y：データセットと同じ
  - ✓ y’__0～9： n 番目の出力（10個）　一番大きなものが推定値
- 混同行列
  - ✓ 縦軸 y=0～9：nが正解
  - ✓ 横軸 y’__0～9：推定値が n の個数
  - ✓ Precision：推定値が正しい率
  - ✓ Recall：正解値を正しく推定した率
  - ✓ Accuracy：データセット全体で正解値を正しく推定した率
  - ✓ F-measure：PrecisionとRecallの調和平均

（参考）評価基準

- Accuracy（正解率） ＝ (TP+TN)÷データ総数
- Precision（適合率） ＝ TP÷推定数、　　推定数=TP+FP
- Recall（再現率）　 ＝ TP÷推定すべき数、推定すべき数=TP+FN
- F measure（F値）　＝ 2TP÷(FP+2TP+FN)

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{Precision} + \frac{1}{recall}\right)$$



TP(True Positive)：　正しく「nである」と推定した回数
TN(True Negative)：正しく「nではない」と推定した回数
FP(False Positive)：　間違えて「nである」と推定した回数（偽陽性）
FN(False Negative)：間違えて「nではない」と推定した回数（偽陰性）

学習タブ：Learning Curveの見かた

- 右縦軸Error（赤い曲線）
  - ✓ 正解値と推定値の違い（誤差評価値）
    CategoricalCrossEntropyでは、y=nの時、-log(y'__n)
  - ✓ 実線は訓練データセットに対する値、点線は評価データセットに対する値
    この例では２つのデータセットが同じなので、点線が見えない
  - ✓ 「コンフィグ：Global Config」の「モニター間隔」ごとに測定する
- 左縦軸Cost（青い曲線）
  - ✓ Error以外にさらに制約を付加した値
    この例では、Errorと同じなので青い曲線が見えない
- 横軸Epoch
  - ➢ 訓練データセットのデータ数だけ学習することを１Epochと言う
    この例では100Epochは1000回の学習に相当

【ハンズオン】報告書pptx/htmlの作成

- 「学習」で、報告したい履歴を右クリック、「エクスポート」、「pptx beta」→生成したreport.pptxが開く
- 同様に「html beta」 →生成したreport.htmlが開く

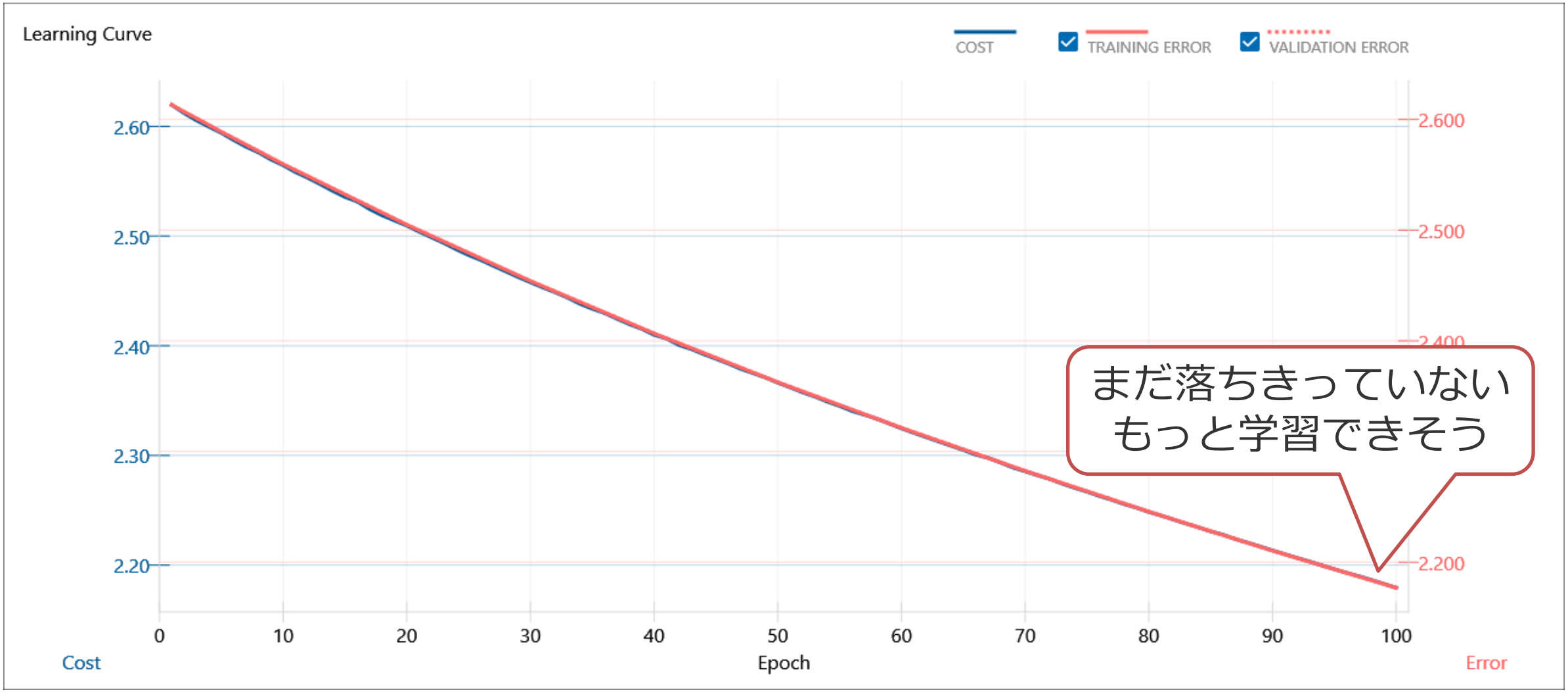
- pptx/htmlの作成場所は、作成した履歴を右クリック、「学習結果のフォルダを開く」→export_reportフォルダの中

- output_resultとconfusion_matrix_yも生成されている

この時点での評価結果

- Accuracyが0.1
- ６しか正しく推定できてない（y=6のRecallだけが1）
- 「学習」でLearning Curveを見ると、まだまだ行けそう
- 学習反復世代数（Max Epoch）が100では足りない！

アンダーフィット（学習不足）



【ハンズオン】

1. 「コンフィグ：Global Config」の「学習反復世代数」を500に
2. 「プロジェクトを保存」（コンフィグ変更時には必ず保存）
3. 学習を実行
4. 評価を実行
5. 「評価」の「出力結果」と「混同行列」を確認

- Accuracyはいくつになりましたか？

演習（次のページに解答あり）

- 正確に推定できない数字は何でしょう？

- それはどの数字と間違えていますか？
  - 混同行列で間違えている場所の「１」を
    ダブルクリックすると、間違えたデータのみ表示
- 正答と誤答の差はどれくらいでしょうか？

- 何Epochまでやればすべて正解できるようになるでしょう？

解答

- 正確に推定できない数字は何でしょう？：３と４
- それはどの数字と間違えていますか？：３→９，４→１
- 正答と誤答の差はどれくらいでしょうか？：
  3：11.70% vs 9：18.03%
  4：18.18% vs 1：18.59%
- 何Epochまでやればすべて正解できるようになるでしょう？：
  850Epoch程度

作ってみよう、開花予測

適用例その２：開花予測

- 温度と湿度から花が咲くかどうかを予測

列ラベルの「:」以降は無視

x__0:Temperature	x__1:Humidity	y:Bloom?
6.1	62	1
29.8	40	1
18.5	44	0

気温

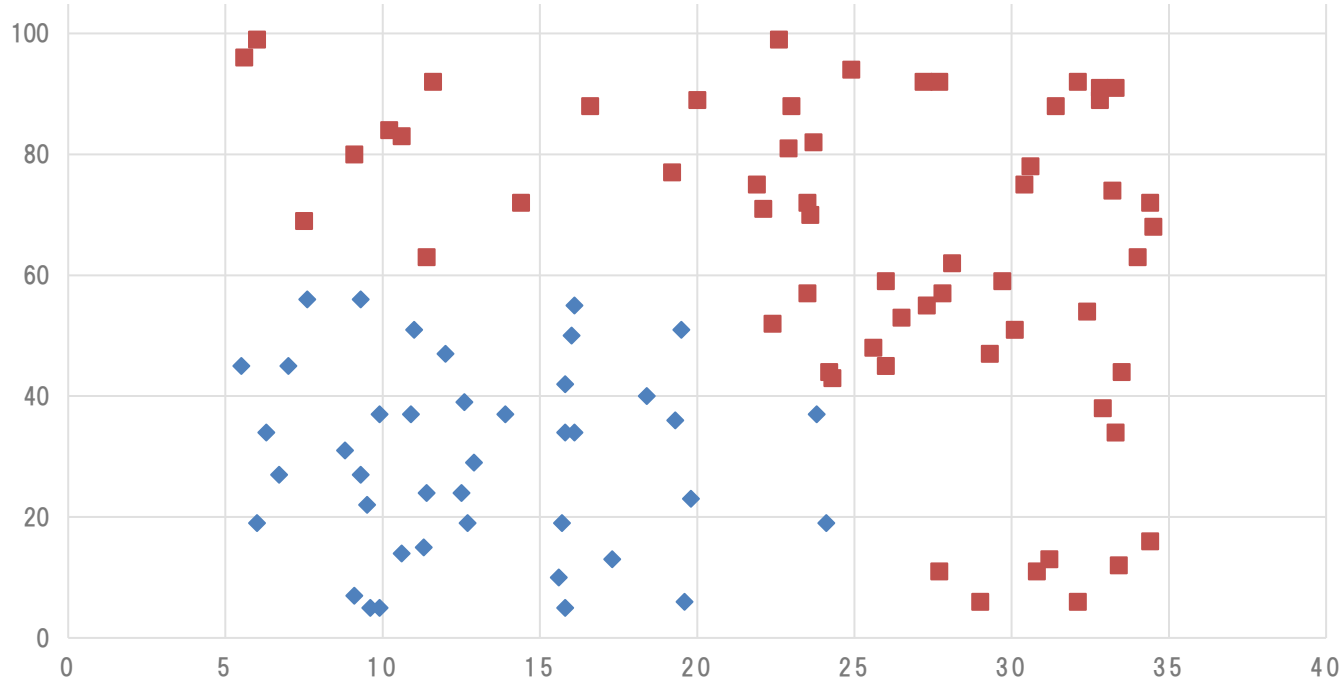
湿度

長さ２の配列

開花可能性
０％:咲かない
100%:咲く

【ハンズオン】開花予想のデータセットを作る

1. 配布したNonlinearBloom100.csvの内容を確認



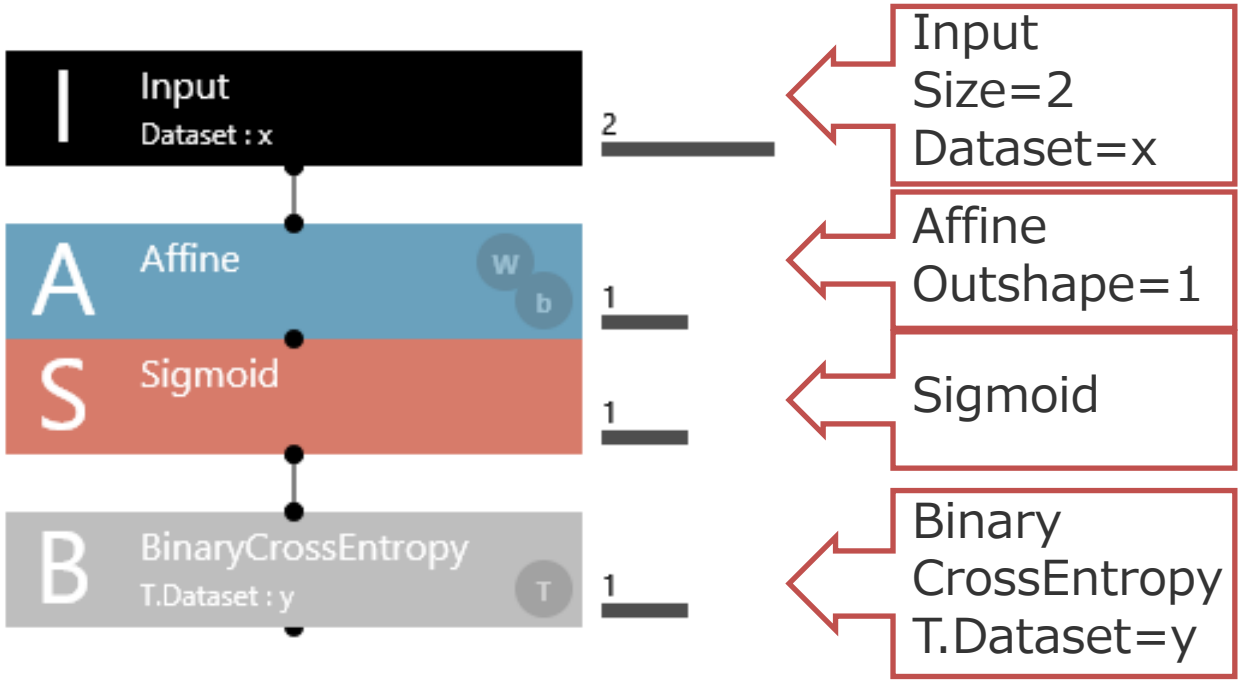
2. 「ホーム：データセット」の「データセットを開く」で
   NonlinearBloom100.csvとNonlinearBloom10.csvを開く

（参考）Excelで散布図の作り方

1. 配布したNonlinearBloom100.csvをExcelで開く
2. A~C列を選択し、「y:Bloom?」を「最優先されるキー」としてソート
3. C列の前に一列挿入
4. B列の値のうちD列（y:Bloom?)の値が０のものを、C列に移動
5. B列の１行目を「咲く」、C列の１行目を「咲かない」に変更
6. A~C列を選択し、「挿入」→「グラフ」→「散布図」

【ハンズオン】開花予想の試作(1)

1. 新しいプロジェクトを作成
2. 訓練データセットにNonlinearBloom100.csvを設定
3. 評価データセットにNonlinearBloom10.csvを設定
4. 右図のネットワークを設計
5. バッチサイズを10に設定
6. 学習 & 評価



解説：問題種類ごとの標準のレイヤー部品

- Activation関数(赤い部品)とLoss関数(グレーの部品)の標準

問題の種類	Activation関数	Loss関数
Yes/No問題	Sigmoid	BinaryCrossEntropy
３個以上の選択問題	Softmax	CatgoricalCrossEntropy
類似度や予測値等の 実数値の問題	不要	SquaredError
（２層以上の途中の層）	Sigmoid/Tanh	

  - Sigmoid：0.0~1.0の値
  - Tanh：-1.0~1.0の値 (Hyperbolic Tangent、ハイパータンジェント、タンエッチ)
  - Softmax：確率分布（0.0~1.0で合計が1.0）

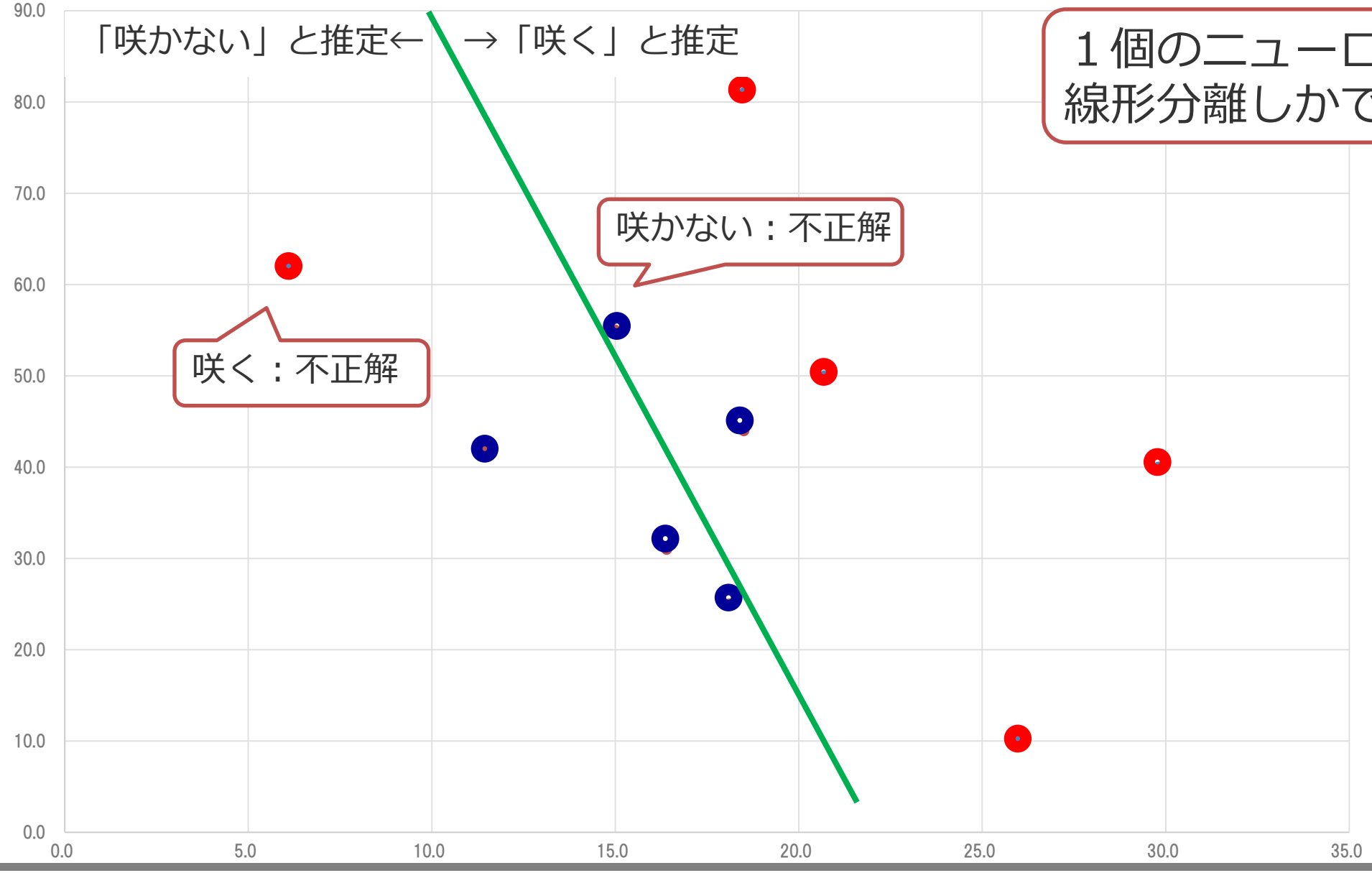
この時点での評価結果

- Accuracyが0.4
- 10個のうち７個で「咲く」と答えている
- 学習曲線を見ると、まだ少し行けそう

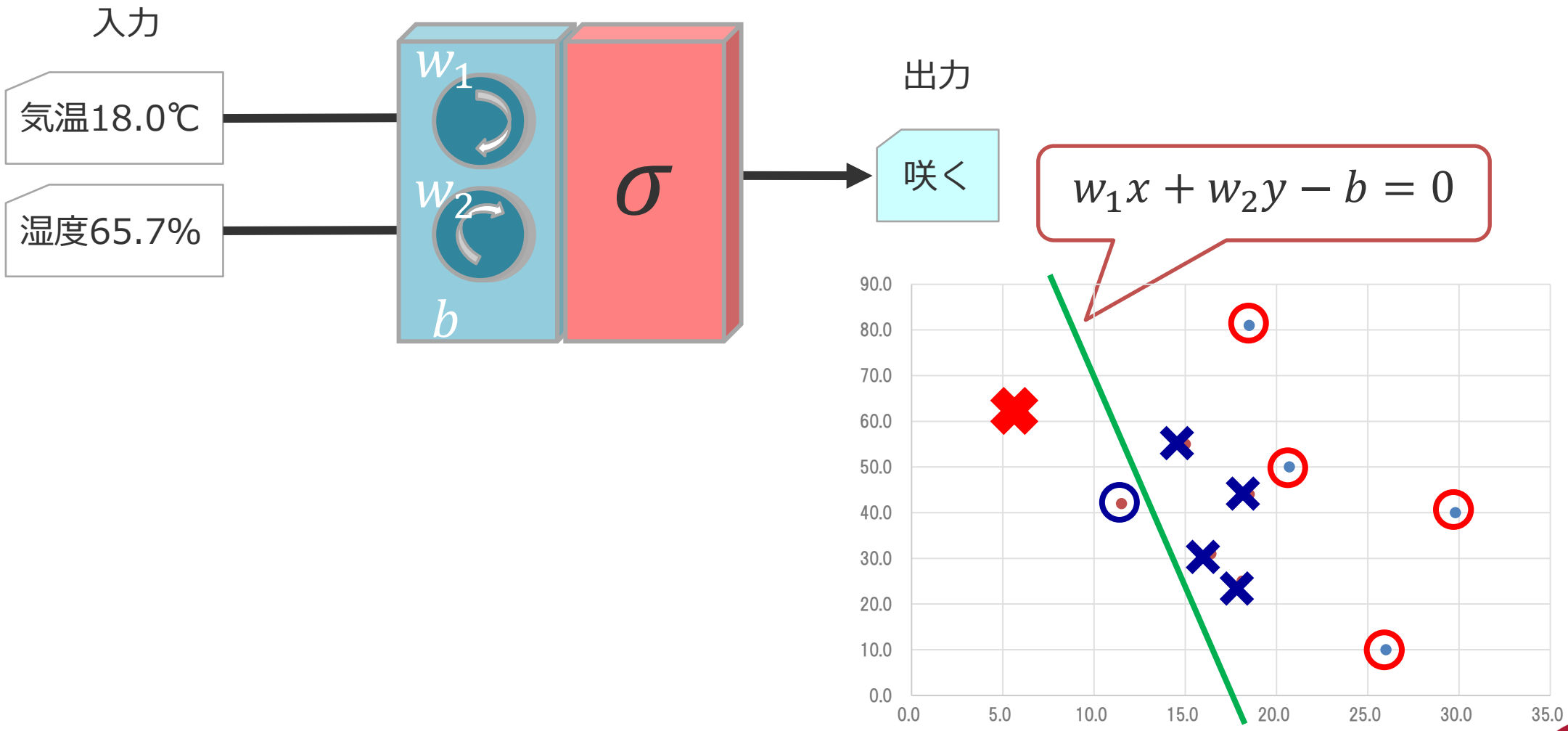
【ハンズオン】演習

- 何Epochまでやればすべて正解できるようになるでしょう？

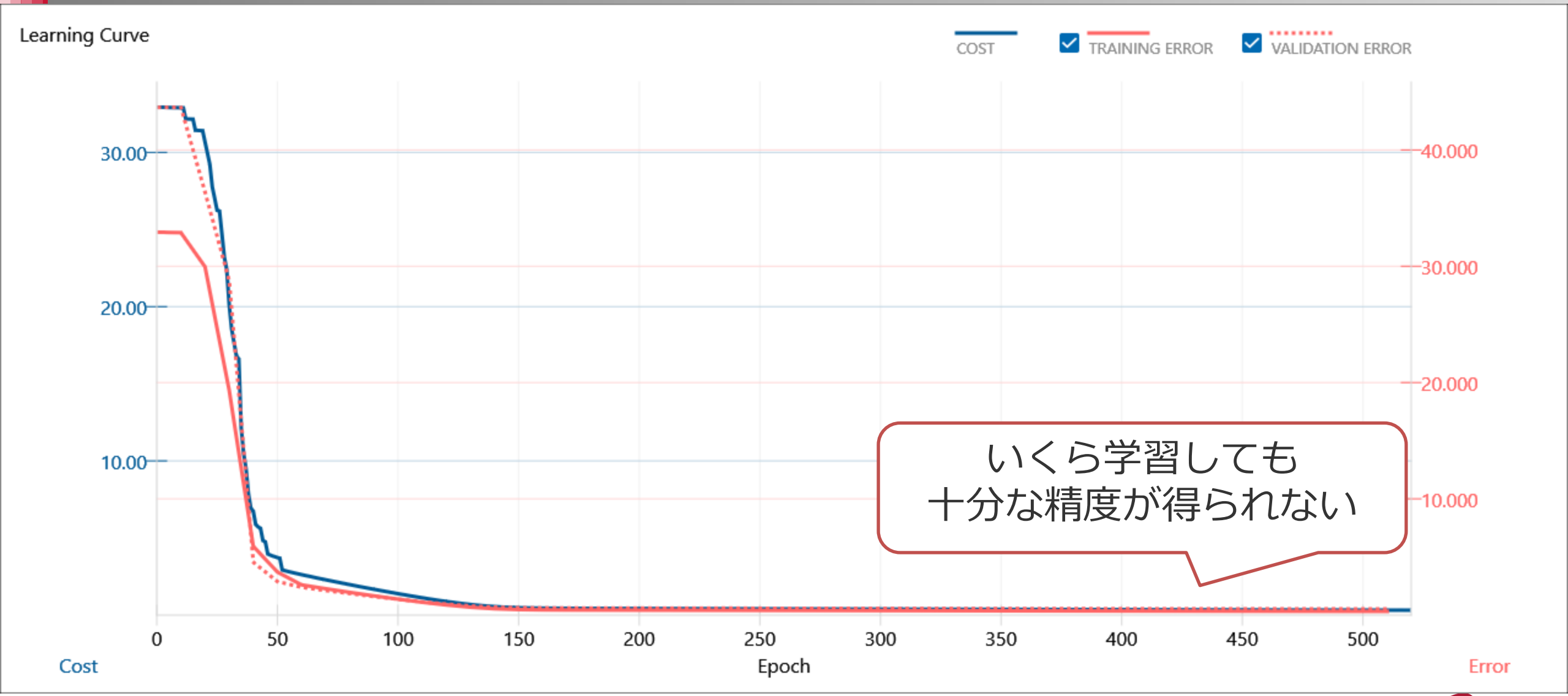
種明かし：このネットワークではできません



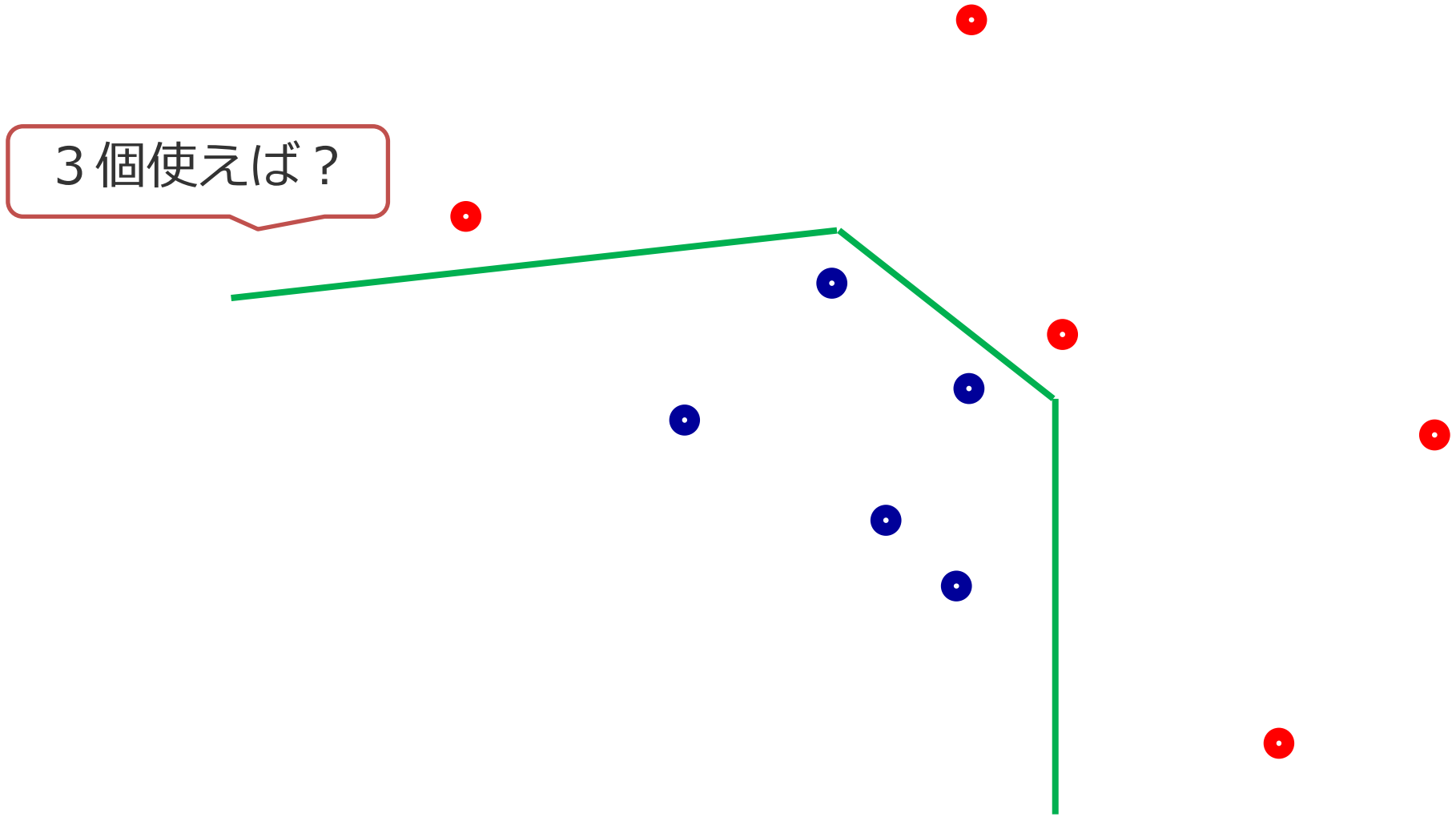
1個のニューロンでは線形分離しかできない



ネットワークのキャパシティ不足

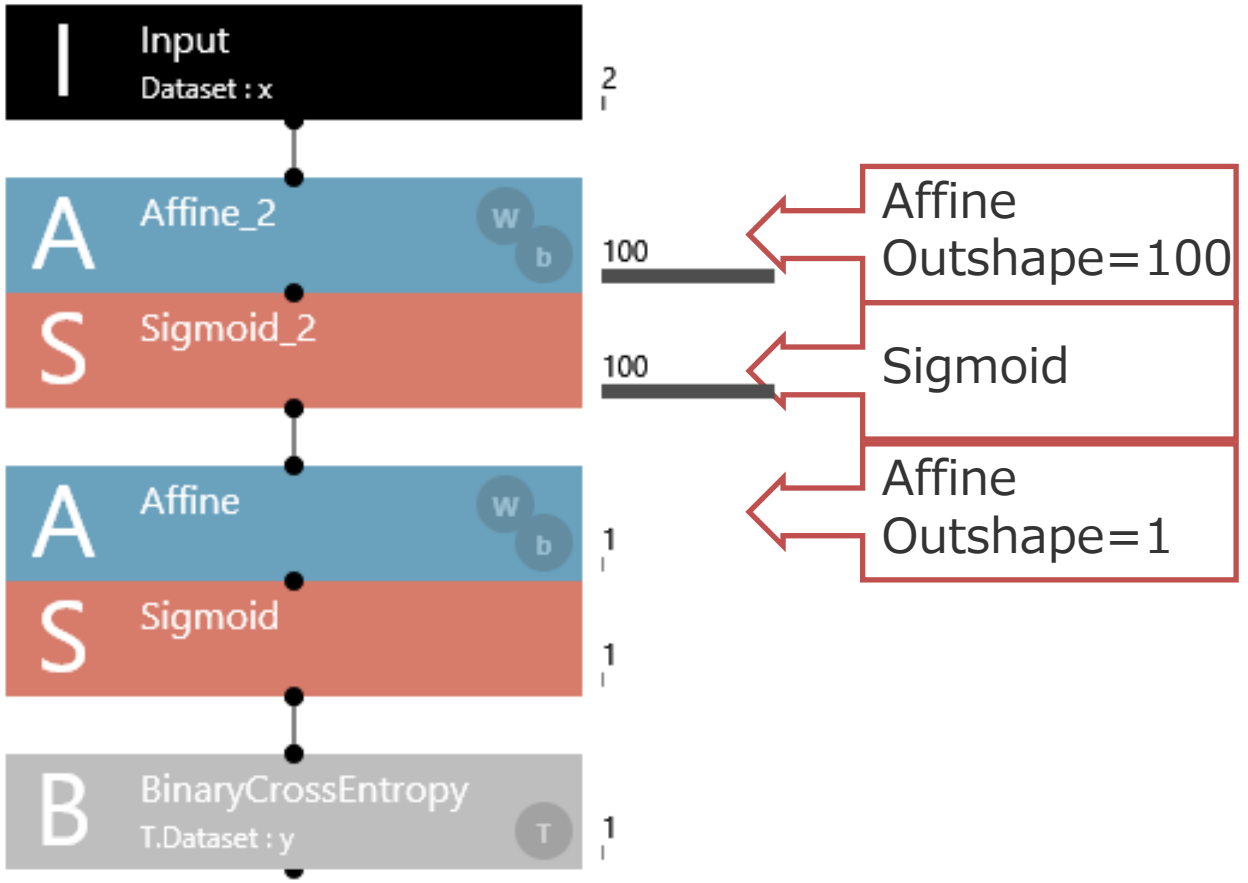


こう区切りたい



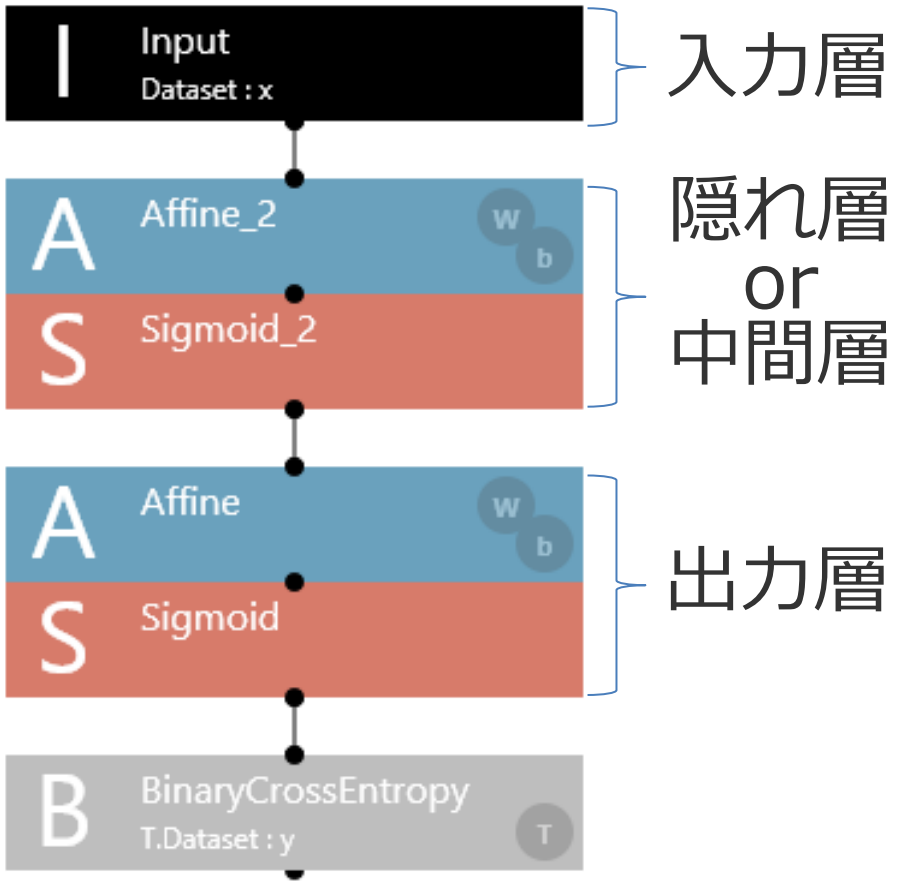
【ハンズオン】開花予想の試作(2)

1. 右図のネットワークに変更
2. 学習反復世代数500で学習



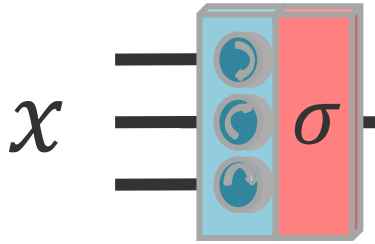
解説：レイヤーの呼び方

- NNCでは、すべての部品を「レイヤー」と呼ぶが、一般的には「層（layer)」とは右図の部分
- 「隠れ層(hidden layer)」は、外から見えない層の意味。中間層とも言う
- 右図全体は「２層ネットワーク」と言う（入力層にはパラメタが無い）



解説：ニューラルネットワークのキャパシティ

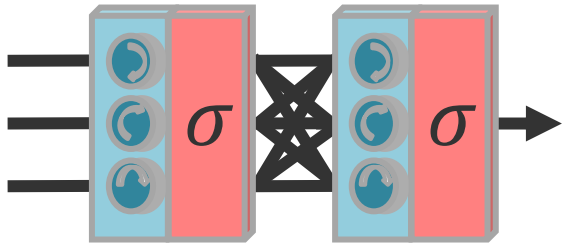
- ニューラルネットワークの学習とは、
  入力と出力の間に存在する「関数」をデータから見つける技術



$$y = f(x)$$

$$y = ax + b$$

- パラメタ１層のニューラルネットワークは「簡単な」関数しか表現できない
- パラメタ２層のニューラルネットワークは、
  中間に十分な数のパラメタがあれば、
  理論上、あらゆる関数を表現できる



ニューラルネットワークを使う時の注意事項

- 入力データと出力データの間に関数が存在するならば、
  ２層以上のニューラルネットワークはそれを表現できる
- しかし、
  - ➢ それを見つけられるかどうかはわからない
    - ✓ 100個のGPUを３ヶ月使わないと…かも
  - ➢ そもそも関数が存在しなければ見つからない
    - ✓ 宝くじが当たる売り場 $= f$(気象データ) ？？
  - ➢ 関数が存在しても、入力データを用意できるとは限らない
    - ✓ 「専門医の見立て」の入力データはカルテだけ？

【ハンズオン】演習

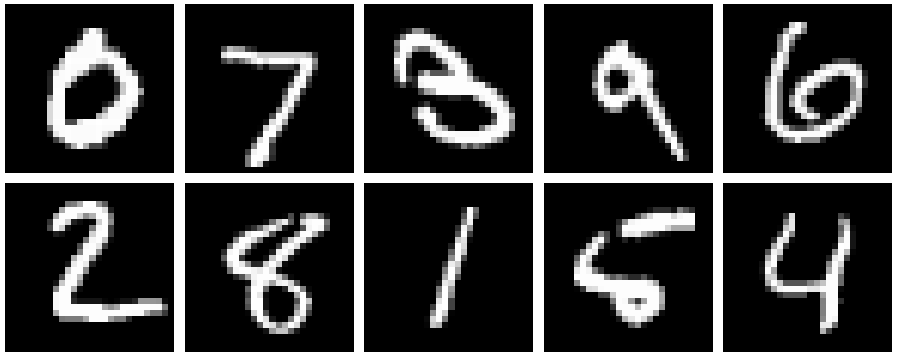
- Accuracyはいくつになりましたか？
- 精度を上げる２つの方法：
  - Epoch数を増やす
  - Sigmoid_2をTanhに変える

[余力のある人のみ]

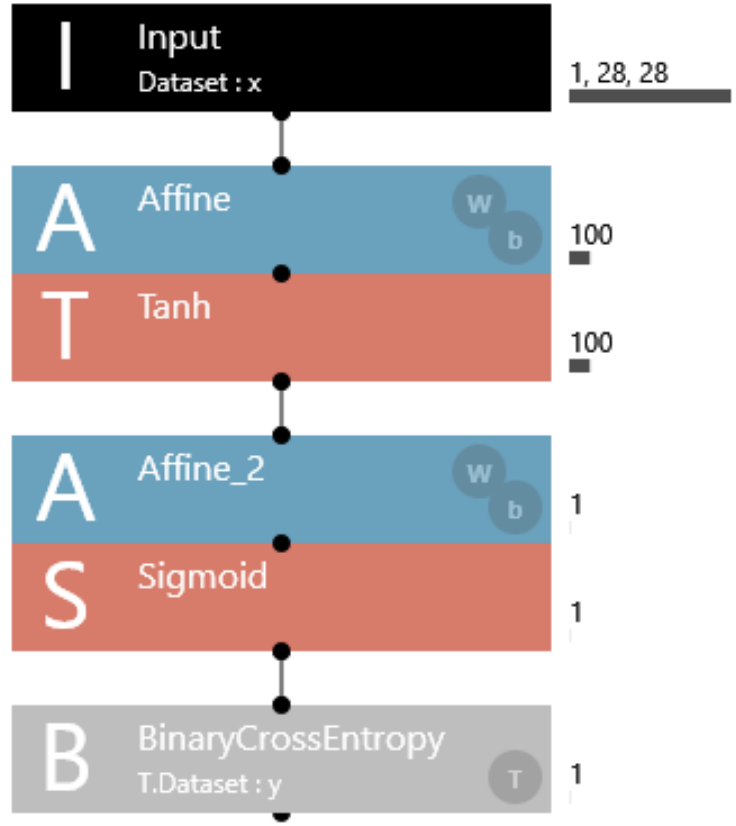
- Affine_2のOutShapeはいくつまで減らせるでしょう？
- LinearBloom100.csvとLinearBloom10.csvの方は
  パラメタ１層だけでできるはず、試してみて下さい

作ってみよう、手書き文字認識（余力のある人のみ）

適用例その３：手書き文字認識MNIST



- MNISTは6万＋1万文字あって数十分かかるので、まずはsmall_mnist_4or9.csvで、4 or 9のyes/no問題でやってみる
- yes/no問題なので、出力数は１、Activation部品はSigmoid、Loss部品はBinaryCrossEntropy



データセットsmall_mnist_4or9_test.csvの内容

- Index：通し番号
- x：画像ファイルのパス
- y：正解（９かどうか）

Index	x:image	y:9
1	./validation/4/4.png	0
2	./validation/9/7.png	1
…	…	…

画像は、１×縦×横（モノクロ）か３×縦×横（カラー）の３次元配列に変換

【ハンズオン】手書き文字認識MNIST

- 01_logistic_regressionで１層ネットワークでの正解率を確認
- ２層ネットワークでは、正解率何%までいけるでしょうか？
- 隠れ層のパラメタ数の最適値は？
  （オーバーフィットしているので100は多過ぎる）
- その他
  - 途中のActivation部品を変える：Sigmoid、Tanh、Relu、…
  - 隠れ層の後ろにDropoutを追加
  - 入力層の後ろにDropoutを追加

色々試す場合のノウハウ

- 訓練と評価データセットの「キャッシュを有効にする」にチェックを入れると既存のキャッシュを再利用するので2回目からの開始が早くなる
- オーバーフィット（後述）しだしたら、それ以上の学習は無駄なので「停止」ボタンで止める。そのままでは評価ができないので、右クリック「学習完了状態にする」
- 学習済みの状態から追加学習したい場合には、右クリック「編集タブで重み付きで開く」してから学習を実行
- バッチサイズ（後述）を小さくすると1epoch毎の学習時間は遅くなるが、精度はあがる
- 以前の学習曲線を「学習曲線を比較用に開く」で開き、他の学習履歴をクリックすると比較できる
- Activation部品の選択はけっこう効く、Dropoutを入れるとなかなか良い

付録

用語解説：ディープラーニングとバックプロパゲーション

- ディープラーニング（深層学習）とは、**隠れ層が２層以上あるネットワーク**を使った機械学習。なので、本日はディープラーニングをやっていない。
- 層の数を、パラメタを持たない入力層を数える流儀と数えない流儀とがあり、数える場合には４層以上、数えない場合には３層以上を、「深い」と言うので検定試験では上記の定義が出題される。
- 第２次AIブームの頃の計算機パワーでは隠れ層１つまでしか計算できなかったが、2010年代になって深いネットワークも扱えるようになり、第３次AIブームが始まった。
- バックプロパゲーション（誤差逆伝播法）とは、推定誤差を出力層からさらに隠れ層をさかのぼって伝播させ全層のパラメタを調整する計算方法。
1967年に東大・甘利教授が発見したが、1986年にラメルハート、ヒントンらが再発見してbackpropagationと命名した。

用語解説：ハイパーパラメタと学習率

- 学習によって自動的に調整さるニューロンの調整パラメタ（重みとバイアス）以外の種々の設定パラメタをハイパーパラメタと言う。
- ハイパーパラメタは、人間が試行錯誤によって適正値を発見する必要がある。
- NNCでは、各レイヤー部品のプロパティ、コンフィグタブでの各種の設定値がハイパーパラメタで、学習タブの右ペーンの「オーバービュー：設定」で一覧を確認できる。
- 学習率（learning rate）は誤差に対するパラメタの調整率であり、ハイパーパラメタの中で最も重要とされる。学習率が小さ過ぎると学習に時間がかかり、大き過ぎるとオーバーフィット（後述）し易い。
- NNCでは、学習率は「コンフィグ：Optimizer」で設定するが、「最適化アルゴリズム」の種類によって名前が異なる。「Adam」の場合は「Alpha」でデフォルトは0.001。
- 学習率の適正値は、ネットワークの構造やパラメタの数によって異なるので、これらを変更した場合には、そのつど探索する必要がある。例えば、学習反復世代数を100にしているのに10epochでオーバーフィットするのであれば、学習率を1/10にしてみる。逆に、100epochでもアンダーフィットしているならば、学習率を２倍にしてみる。

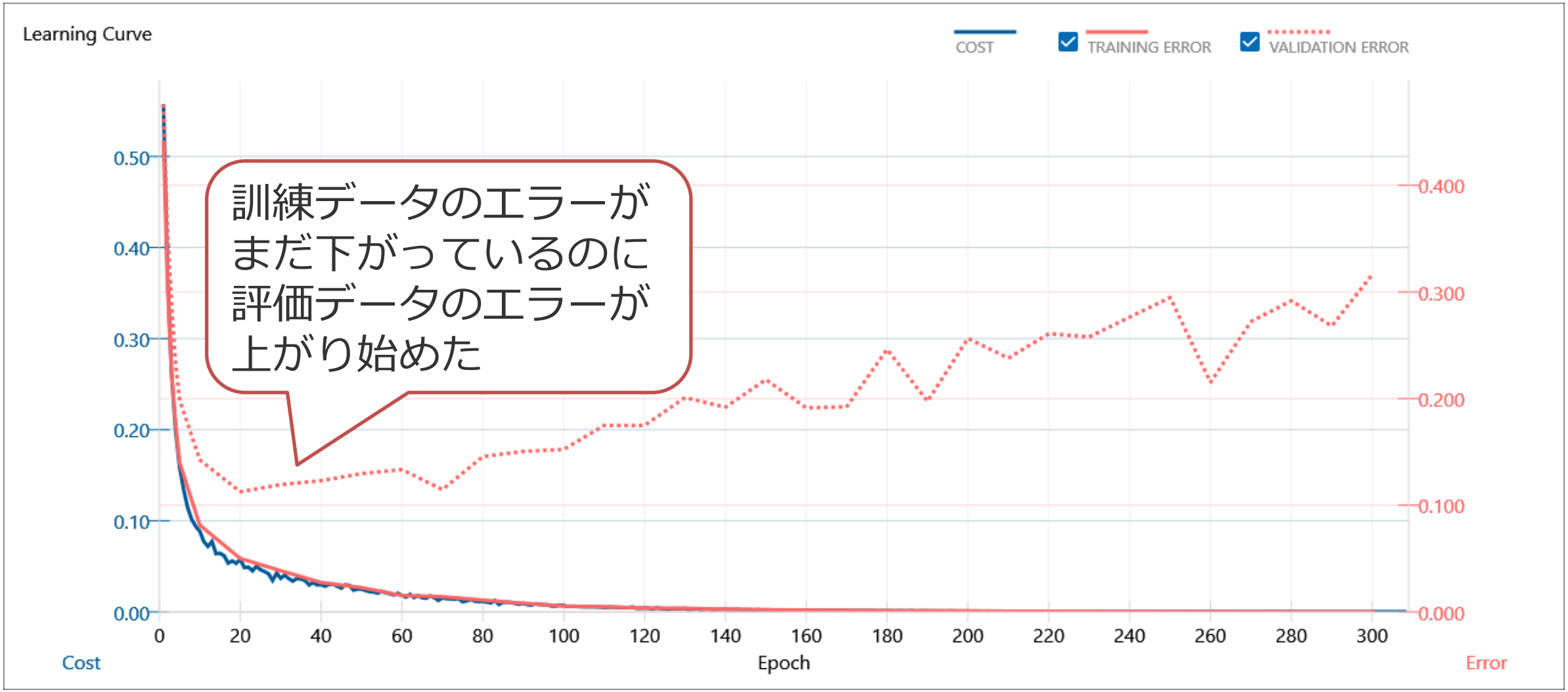
用語解説：オーバーフィットと分布の偏り

- 機械学習は訓練データの分布に偏りが無いことが前提
  これを「独立同分布（i.i.d.†）仮定」という
- 偏りの例
  - 訓練画像の中でソファーに必ず猫がいる　→ソファーがあると「猫」だと推定
  - 過去の白人男性のデータだけに基づいて保険査定　→黒人や女性の査定が正しくない

  注：「偏り」の英語は「バイアス」なので、パラメタの「バイアス」と混同しないように
- 実際の訓練データは、ほとんどは偏っているので
  訓練データとは別に取っておいたデータでの性能評価が必須
- オーバーフィット（過学習）は、偏りがある訓練データを学習し過ぎて、評価があがらないこと
- オーバーフィットを避ける技術を「正則化」という

† independent & identically distributed

オーバーフィット（過学習）



用語解説：Loss

- Loss関数（損失関数、Error関数、誤差関数）は、推定値と正解値の誤差を計算する関数。学習は、この誤差を小さくするようにパラメタを調整する。
- 明日の気温†など、値を推定する場合は二乗誤差(SqueredError)を使うのが一般的。30℃と推定して実際には33℃であったなら、$(30-33)^2=9$が誤差。
- 一方、明日の降雨確率†など、確率を推定する場合には交差エントロピー(CrossEntropy)を使うのが一般的。交差エントロピーを正確に書くと：

  −log(降る確率の推定値)×降る確率の正解値−log(降らない確率の推定値)×降らない確率の正解値

- しかし、翌日雨が降ったら、降る確率の正解値＝100%なので、例えば、推定値が60%ならば $-\log(0.6)=0.222$ が誤差。もし雨が降らなかったら、降らない確率の正解値＝100%なので、$-\log(0.4)=0.398$ が誤差。
  つまり交差エントロピー誤差は、**−log(正解の推定確率)** と考えておけばOK。

† 注意：気象業務法第17条により気象庁以外の者が予報業務を行うには気象庁長官の許可が必要です

用語解説：Cost

- Cost関数（コスト関数、目的関数）は、学習で小さくしたい目的値を計算する関数。
- Cost関数にはLoss関数が必ず含まれ、さらに、オーバーフィットを防ぐ（正則化）ために別の値もCost関数に含めて小さくすることがある。
- NNCでは、「コンフィグ：Optimizer」の「Weight Decay」がそれにあたる（詳細は第３回以降で解説予定）
- Weight Decayのデフォルトは0なので、通常はCostとLoss(Error)は等しい。
- ただし、学習曲線では左右の目盛りのスケールが異なることが多いので、Costの青い線とErrorの赤い線が重なり合わない。

用語解説：バッチサイズ

- 学習は、１つの訓練データ毎にパラメタ調整をすると効率が悪いので、複数個の訓練データによる推定結果の平均誤差を使ってパラメタ調整する。この一回分をバッチと言い、利用する訓練データ数をバッチサイズと言う。
- 訓練データ数が1000、バッチサイズが64ならば、1000÷64=15.6なので、1000個から選んだ64個（最後の1回は40個）のデータで学習するバッチを、16回行うのが1Epochになる。
- NNCでは、学習と評価を同じバッチサイズで実行するので、**バッチサイズは評価データの数以下を指定しなければならない。**
- GPUを利用する時にはバッチサイズ分だけ並列に計算するので、適切なバッチサイズ設定が学習速度に劇的に影響する。ただし、「学習するパラメタ数×バッチサイズ」分のメモリが必要。

用語解説：バッチサイズ（続き）

- 学習曲線に表示するTRAINING ERROR（赤い実線）は、各Epoch終了後に、上の例では16回のバッチで求めた平均誤差のさらなる平均値を取る。
- 一方、VALIDATION ERROR（赤い点線）は、最初の10Epochまでは毎回、それ以後は「コンフィグ：Global Config」の「モニター間隔」毎のEpoch終了後に評価データを使って求める。評価データ数が100ならば、64個と36個で2回評価して、それぞれの平均誤差の平均値を取る。
- したがって、厳密には1000個の訓練データ／100個の評価データの誤差の平均値にはならない。
- バッチサイズをデータの約数（例えば50）にすれば良いのだが、GPUの効率を考えて、２のべき乗(32や256)を選択するのが慣習になっている。
ただし、本当に２のべき乗が最も効率的なのかどうかは要確認。
- 一般に、バッチサイズが小さい方がオーバーフィットしにくいと言われている

用語解説：プロジェクト画面：データセットタブの機能

- Main：1Epochのバッチ数を計算するの使うデータセット。通常はTrainingをチェックする。複数のデータセットを学習に使う時に意味がある。
- シャッフル：チェックすると1Epochごとにデータセット全体をシャッフルする。Validationをシャッフルしてはいけない。
- キャッシュを有効にする：チェックすると、最初にデータセットのcsvを読み込んでキャッシュを作る。データセットcsvを変更した場合にはチェックしておく必要がある
- Image Normalization：画像は各ピクセルを1byte（0~255)で表現するが、これを0.0～1.0の実数値に自動変換する。画像以外のデーアに対しては無意味。

用語解説：Affine（アフィン変換）

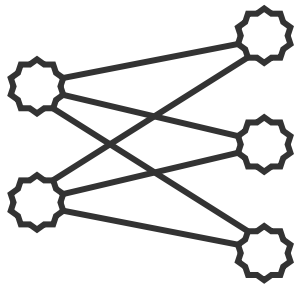
- n個の数値を掛け算と足し算・引き算だけでm個の数値に変換

  ※正確な定義はWikipediaで「アフィン写像」を検索

- n=2（v、w）、m=3（x、y、z）の例：

  x = 1.2v – 0.3w + 3.5
  y = 5.9v + 6.1w - 12.7
  z = -0.7v – 8.2w + 5.3

- n個とm個のニューロンをすべて結合することに相当



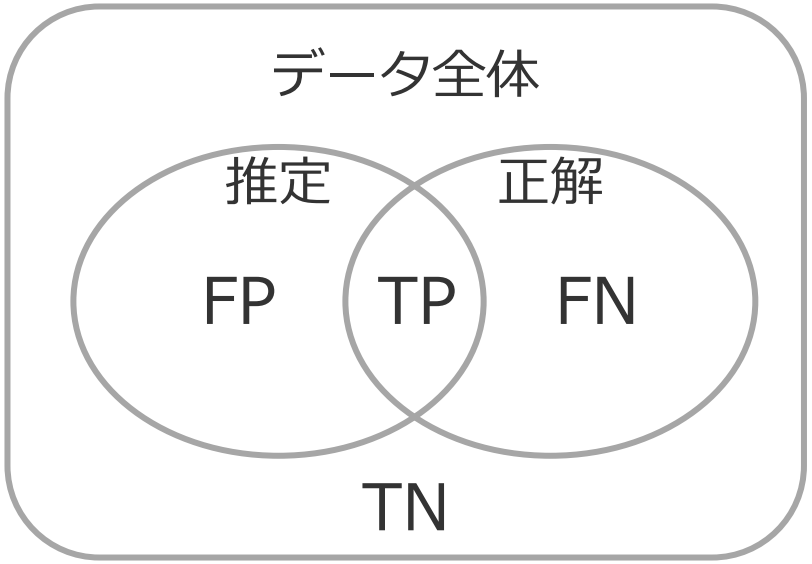
- 一般には「全結合」「Full Connection（FC）」と呼ぶことが多い

おまけの問題（次のページに解答あり）

- Precision（適合率） ＝ TP÷推定数、推定数=TP+FP
- Recall（再現率） ＝ TP÷正解数、正解数=TP+FN

問題：
東京都の新型コロナ感染率を１％とします。
PCR検査の精度（再現率）を99%とします。
（陽性の再現率も、陰性の再現率も99%）

ある人がPCR検査で陽性と言われてしまった時
その人が本当に陽性である確率（適合率）は？



例題の回答

- わかりやすくするために、東京の人口を1千万人とします。

  総数＝TP+TN+FP+FN=10,000,000

  真陽性数=TP+FN=総数×感染率(1%)=100,000

  真陰性数=TN＋FP=総数ー真陽性数=9,900,000

  TP=真陽性数×再現率(99%)=99,000

  TN＝真陰性数×再現率(99%)=9,801,000

  } 陽性と陰性の再現率を同じとしたのがミソ

  FP＝真陰性数ーTN=9,900,000ー9,801,000=99,000

  適合率＝TP÷(TP+FP)=50%

- 陰性の再現率が99.9%であれば、偽陽性FPが9,900、適合率は90.9%になります。真陰性数が圧倒的に多いので、検査は陰性の再現率を上げて、偽陽性を出さないようにすることが大事ですね。


JAPAN
SYSTEMS