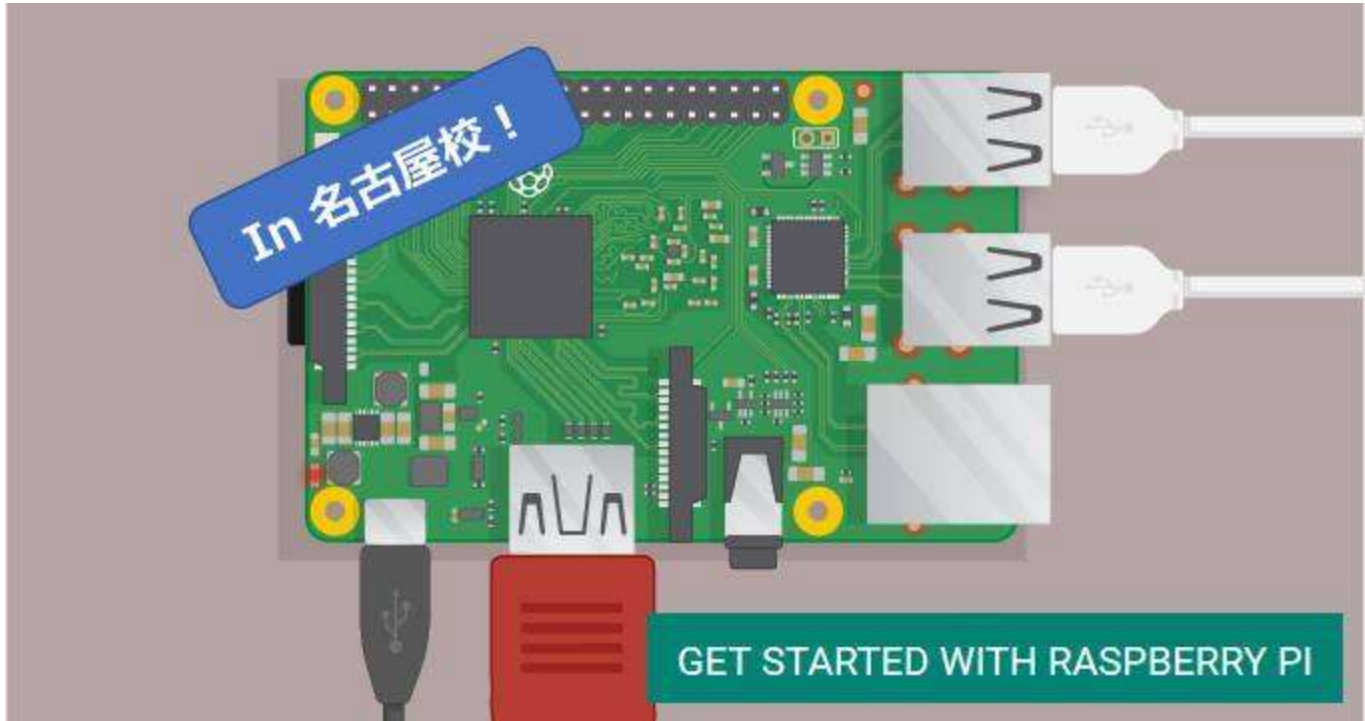
ゼロから学ぶ、ラズパイAI実装ハンズオンセミナー
～セットアップから画像認識AI実装まで～

2018年2月24日

名古屋校 2011期 越智 由浩



今日の心がまえ、スタンス

- 他人の知見(ブログや記事)を参考に“真似ながら動かしてみる”を実践できるようになることを目指します

- コマンドや各種ツールなど次々に新しいことが出てきます。手順はGitHubに掲載しますので復習などにお使いください

- AIを実装して動かすまでの一通りを体感してもらうことで、どのような要素で成り立っているのか、これからどういったことを深掘りして学べばよいかを考えるきっかけになればと思います

応用

- ｹﾞｰﾑｴｰｼﾞｪﾝﾄ
- 自然言語処理
- 画像処理
- 音声認識

人工知能

機械学習

- 教師あり学習
  - 単回帰/重回帰
  - 過学習/正則化
  - クロスバリデーション
  - 決定木/ランダムフォレスト
  - SVM
  - ナイーブベイズ
- 教師なし学習
  - クラスタリング
  - Feature Scaling
  - Feature Selection
  - Feature Extraction

ディープラーニング

- 基礎
  - ニューラルネットワーク
  - 確率的勾配法
  - バックプロパゲーション
- 応用
  - 畳み込みニューラルネットワーク
  - 再帰型ニューラルネットワーク
- フレームワーク
  - Tensorflow
  - Keras
  - Chainer

座学でカバー

基盤知識

IT

- インフラ関連
  - サーバー、ネットワーク
  - Linux
- プログラミング関連
  - Python言語
  - Git/Github
  - Anaconda
  - アルゴリズム・データ構造
  - スクレイピング/クローリング

数学

- 微分積分
- 線形代数
- 確率統計学

一気通貫にやっちまう欲張りなハンズオン

今日の時間配分

0:00 – 2:00	• ラズパイ基本セットアップ • カメラを使った画像配信

2:00 – 3:30	• 座学
	  • ニューラルネットワーク
	  • 手書き文字認識
	  • 畳み込みニューラルネットワーク

3:30 – 4:30	• 手書き文字認識システムの実装とテスト • 物体識別システムの実装とテスト
4:30 – 5:00	• クロージング・振り返り

GitHubのガイドに沿って進んで行きましょう

https://github.com/yoshihiroo/programming-workshop/tree/master/rpi_ai_handson

今日の時間配分

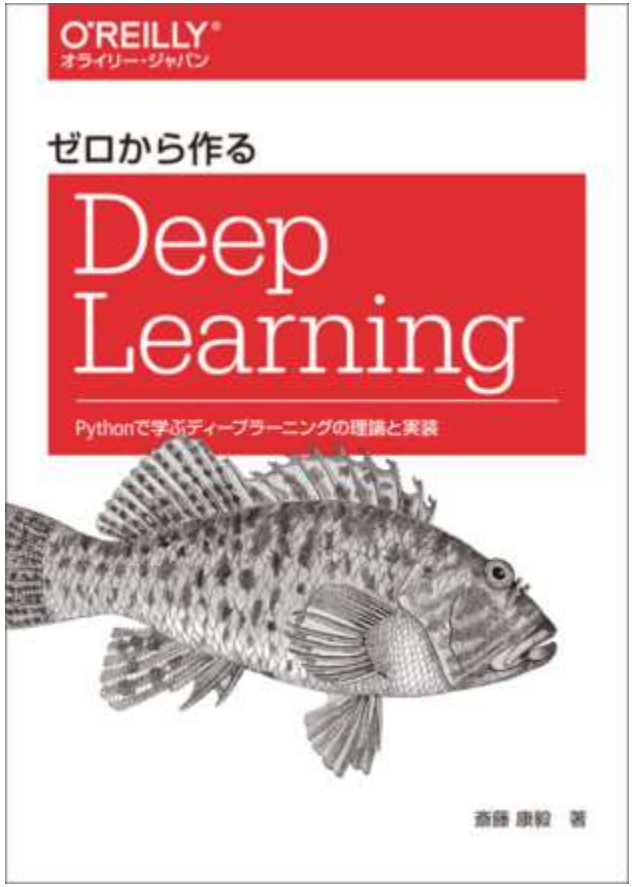
0:00 – 2:00	• ラズパイ基本セットアップ • カメラを使った画像配信

2:00 – 3:30	• 座学
	  • ニューラルネットワーク
	  • 手書き文字認識
	  • 畳み込みニューラルネットワーク

3:30 – 4:30	• 手書き文字認識システムの実装とテスト • 物体識別システムの実装とテスト
4:30 – 5:00	• クロージング・振り返り

今日の内容のベースとなる本

- 内容、ソースコード、図など、こちらの本から引用しております
- Amazonでディープラーニングで検索するとベストセラー本として簡単に見つかります
- 復習兼ねてぜひご購読をお勧めします

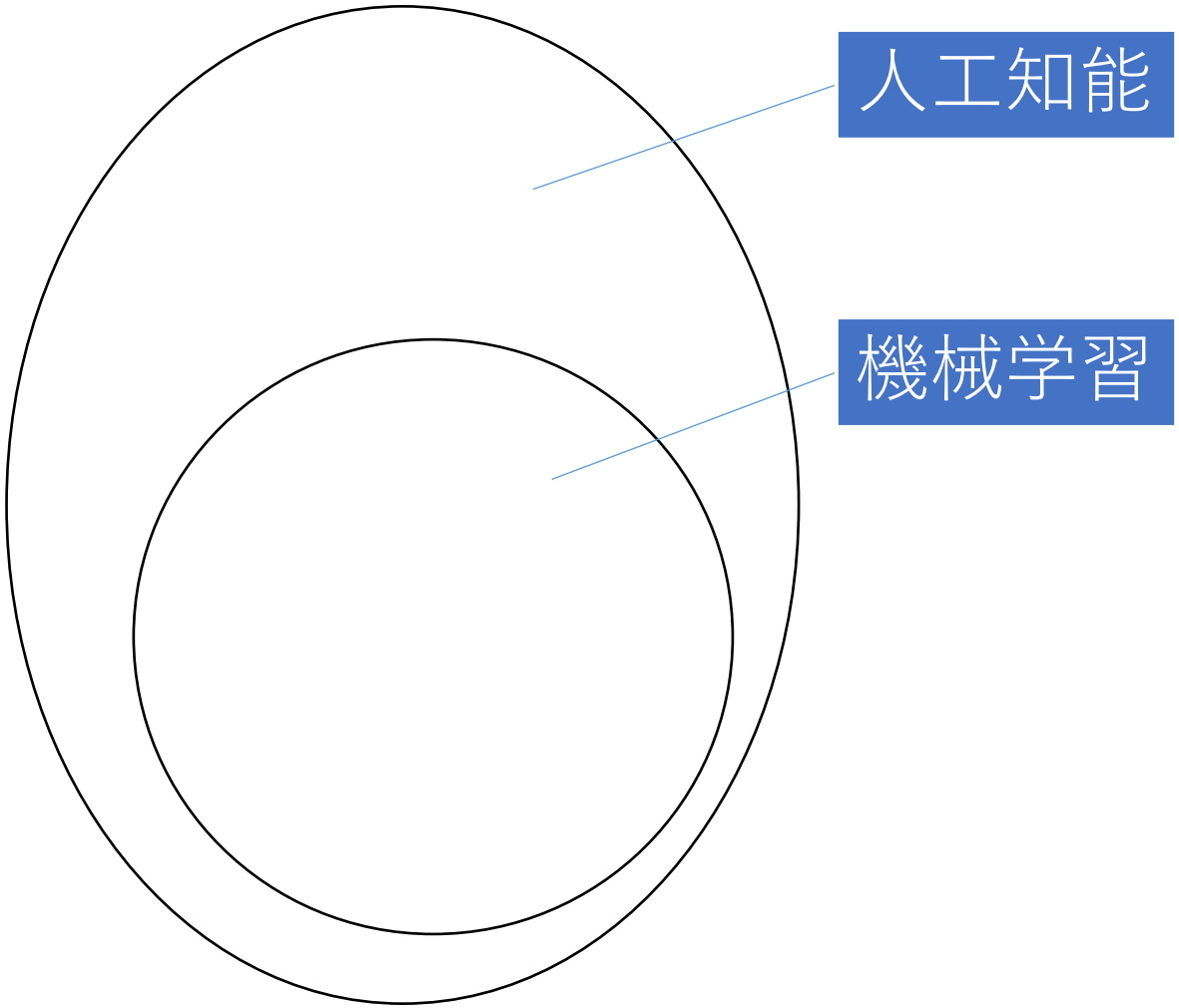


まず、言葉の整理

人工知能

「お、こいつ賢いな!」と思わせるもの、ふるまい。
またそれを探求する学問領域。

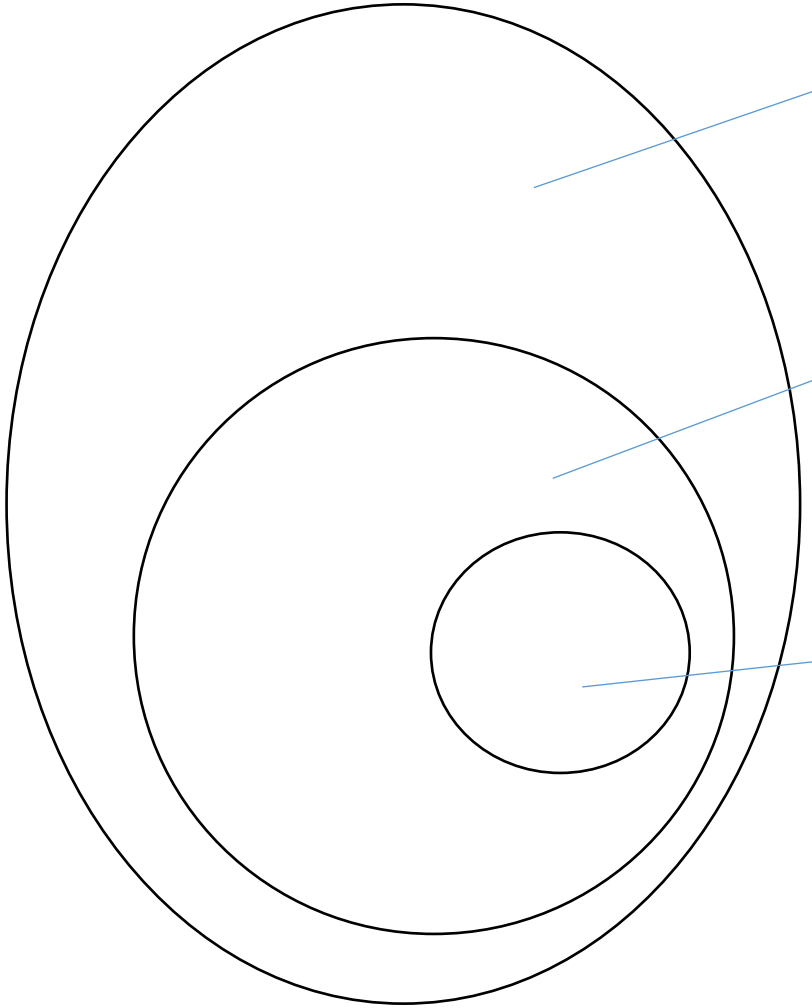
まず、言葉の整理



人工知能を実現する手段の一つ。
過去のデータ(知見/経験)に基づいて：
- 将来を予測する
- 未知のものを分類する

例)
eコマースサイトのリコメンド
迷惑メールの自動振り分け

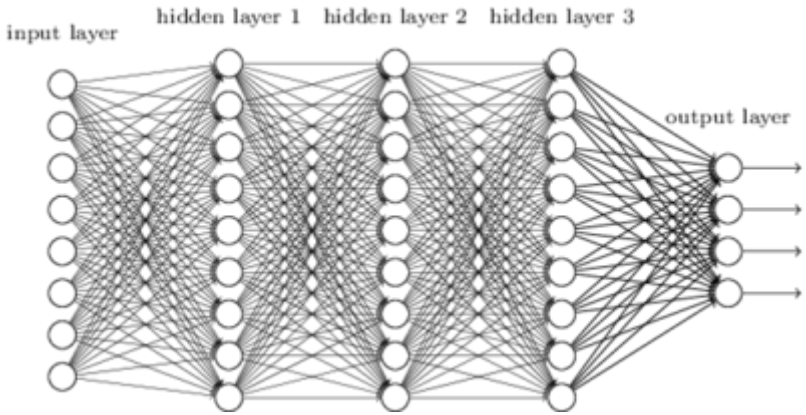
まず、言葉の整理



人工知能

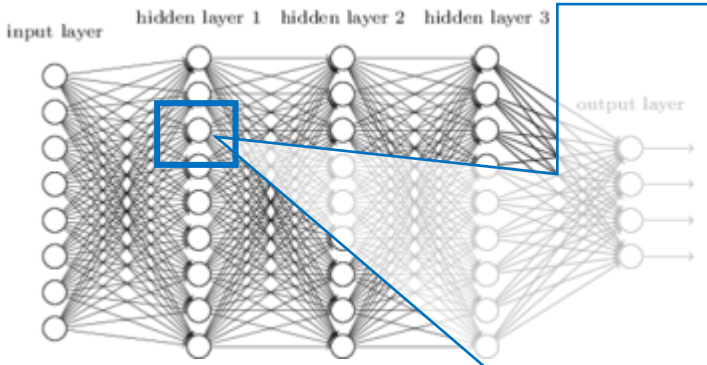
機械学習

ディープラーニング



多階層(ディープな)ニューラルネットワークを用いた機械学習(ラーニング)

ニューラルネットワークの計算の考え方



左から右に順に数字を計算して伝えていく

重み(信号の伝わりやすさ)

バイアス(活性のし易さ)

1.0

0.5

0.1

0.2

+0.3

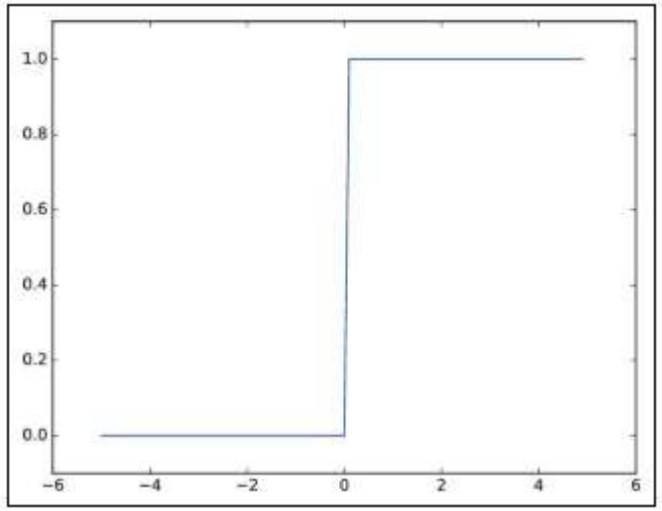
入力元の
信号の強さ

次のノードへ

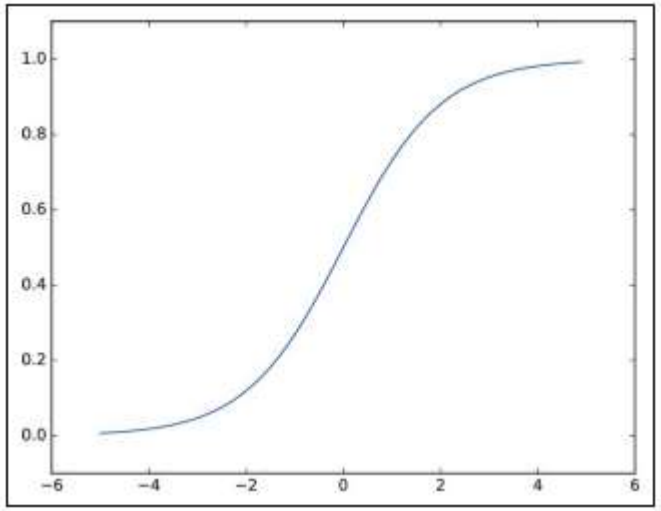
活性化関数
(いくつか種類がある)

活性化関数

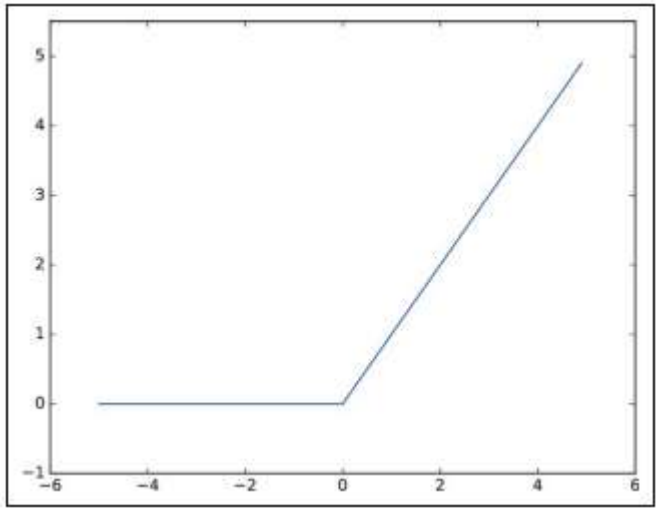
ステップ関数



シグモイド関数



ReLU関数
(Rectified Linear Unit)

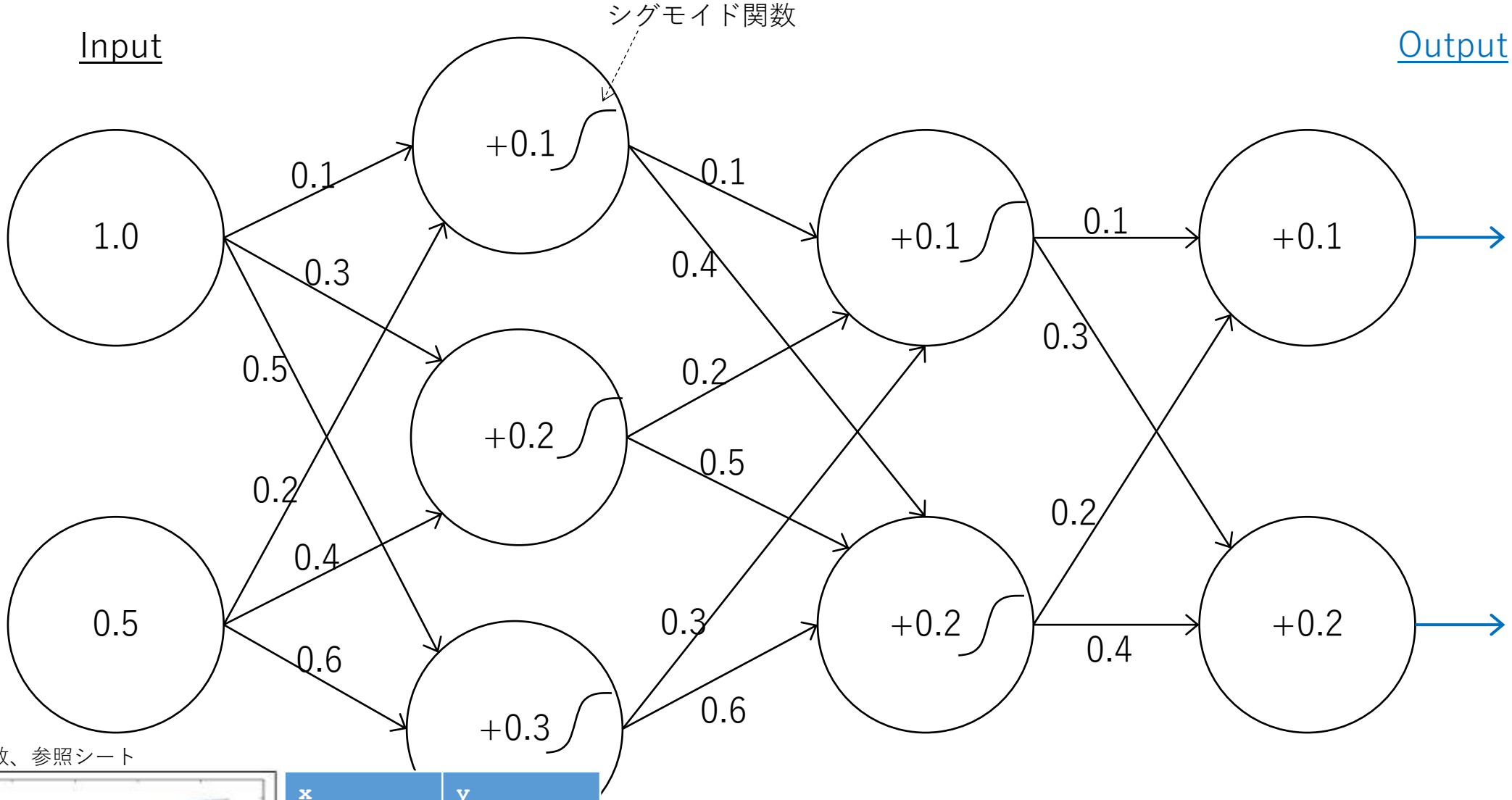


$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

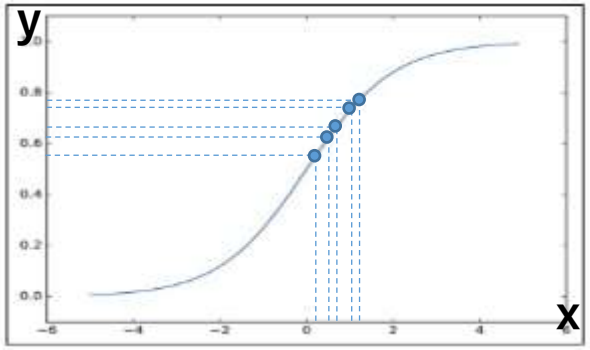
$$h(x) = \frac{1}{1 + \exp(-x)}$$

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

手触り感持って理解するために、
ニューラルネットワークを手計算！

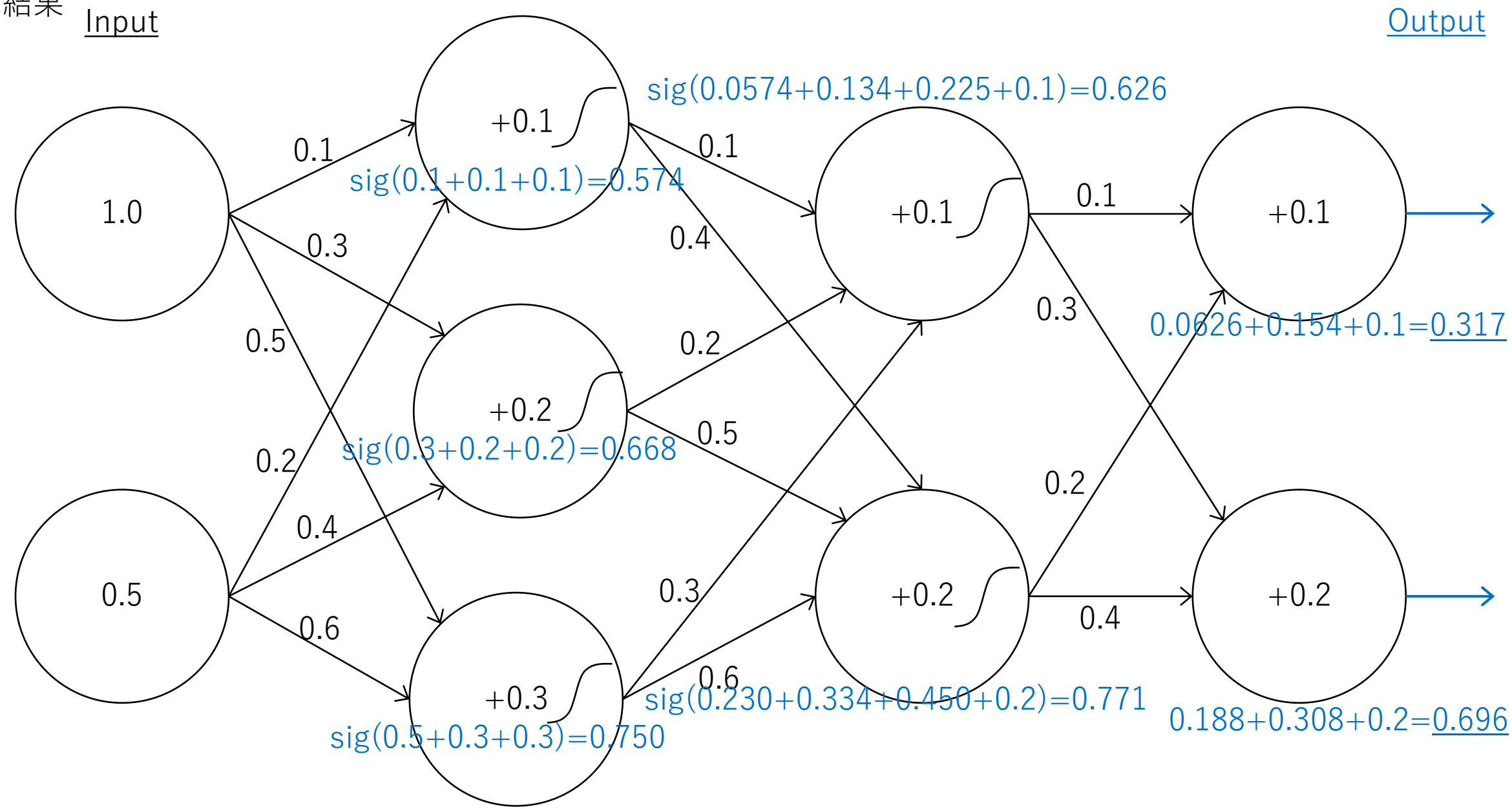


シグモイド関数、参照シート

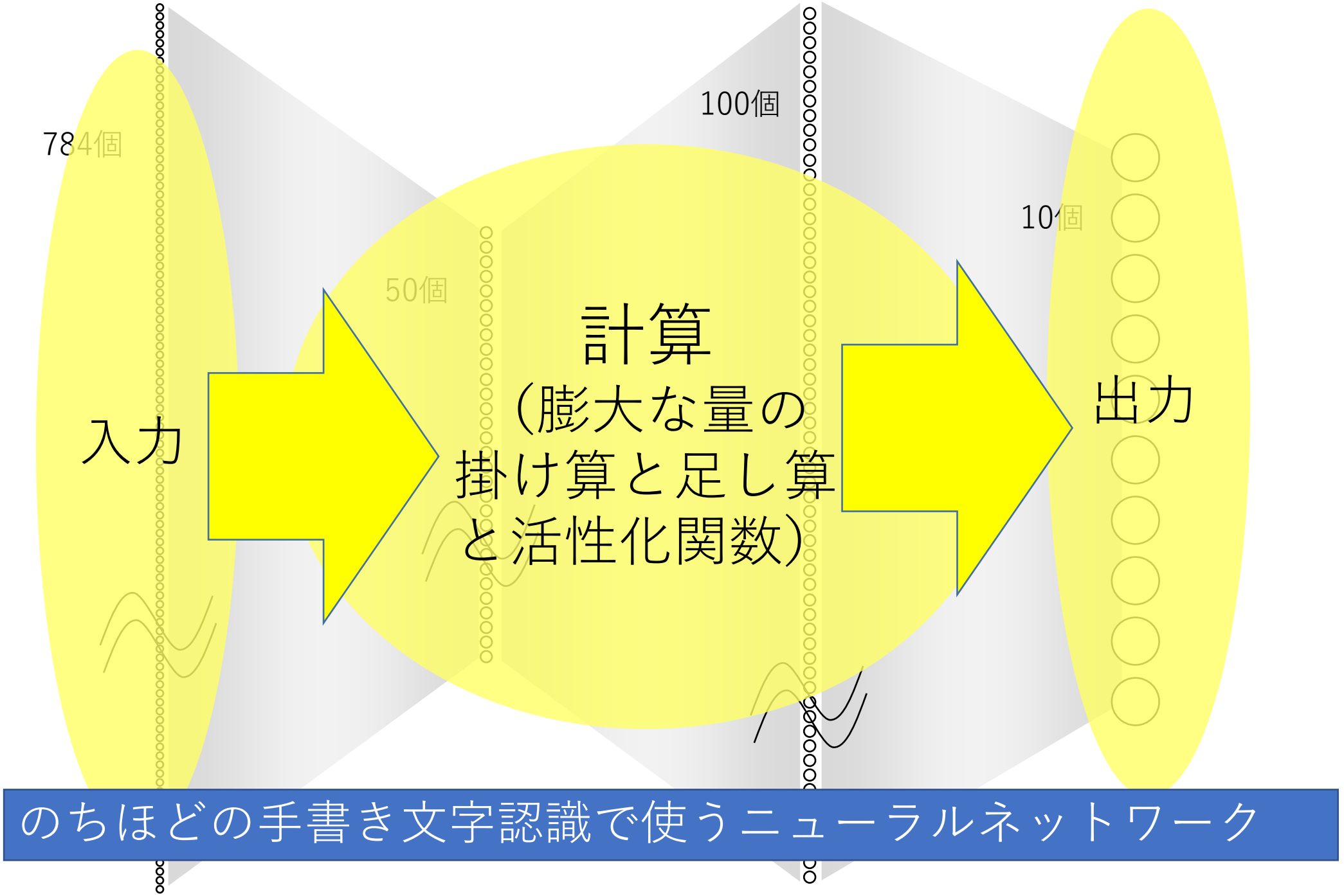


x	y
0.3	0.574
0.516	0.626
0.7	0.668
1.1	0.750
1.214	0.771

手計算結果



sig()：シグモイド関数



のちほどの手書き文字認識で使うニューラルネットワーク

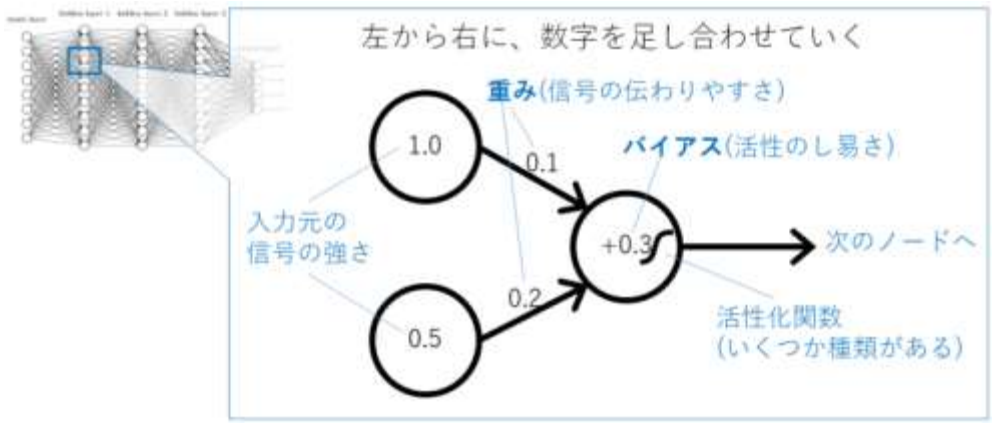
手計算じゃムリ！

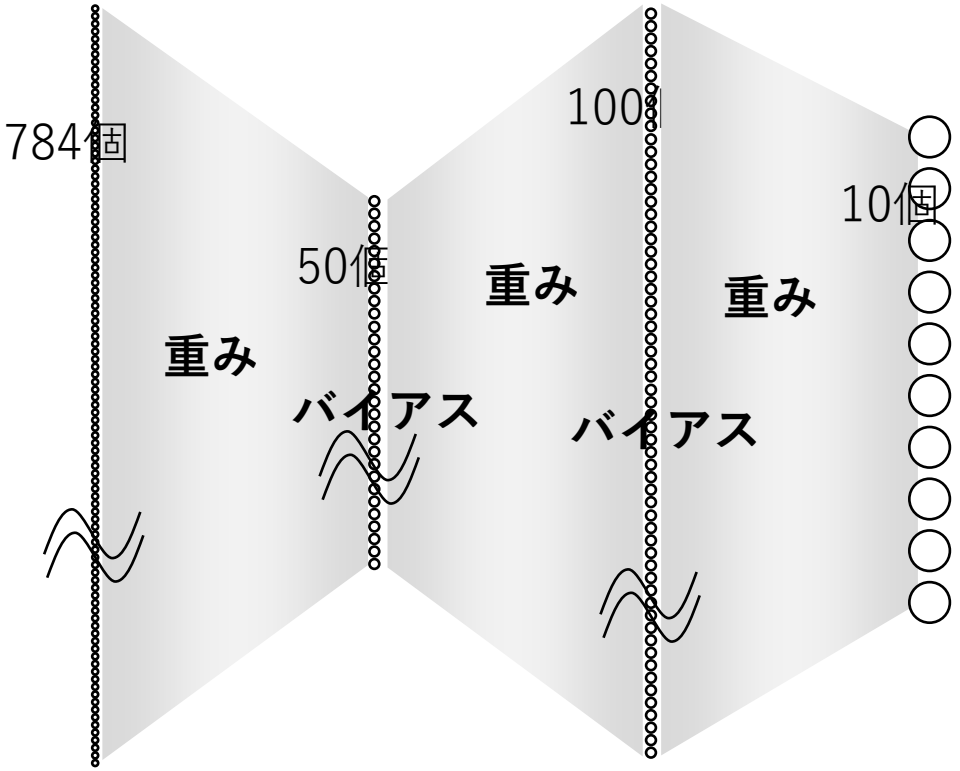
計算
（膨大な量の
掛け算と足し算
と活性化関数）

単調な計算の繰り返しは
コンピューターの得意技。
プログラムを作って計算
させちゃえばいい！

パラメータとハイパーパラメータ

先ほど見せた図





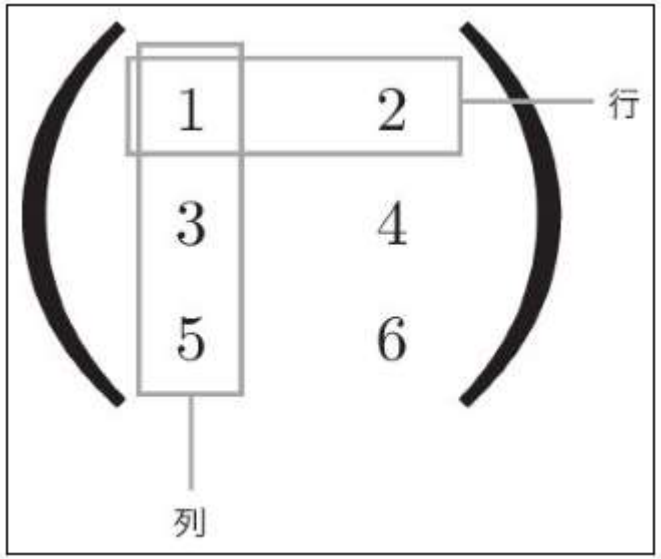
パラメータとは：　右図でいう重みとバイアス。
ニューラルネットワークの振る舞いを調整する値。

ハイパーパラメータとは：　ネットワークを何層にするか、層あたりのノードの数をいくつにするかなど、ネットワークの設計を決める値。

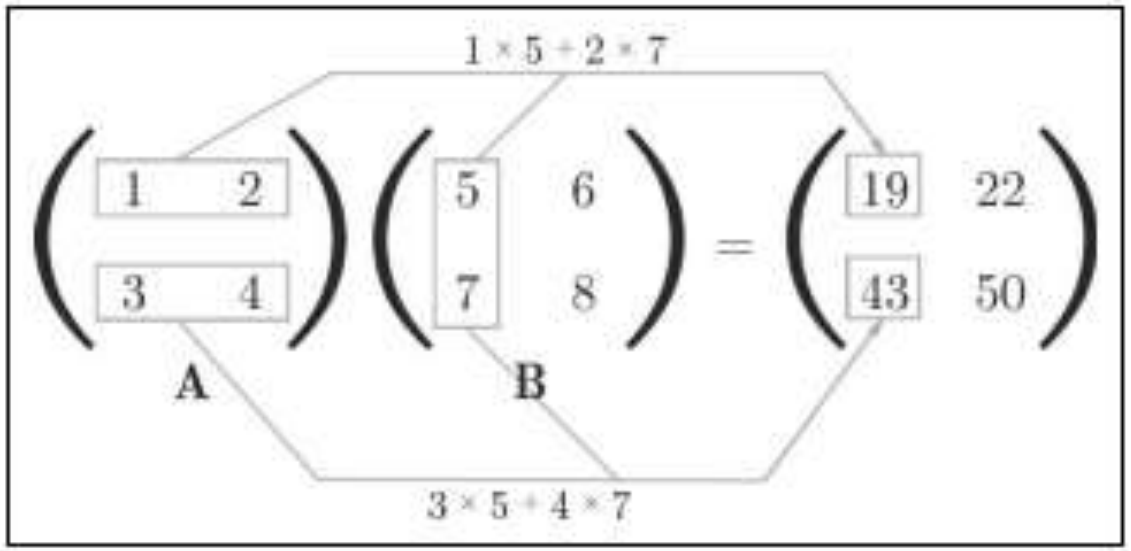
数学の便利な道具 – 行列、内積

なぜ便利かは後ほどわかる

3 x 2の行列の例



行列の内積の例



行列計算に慣れ親しむ

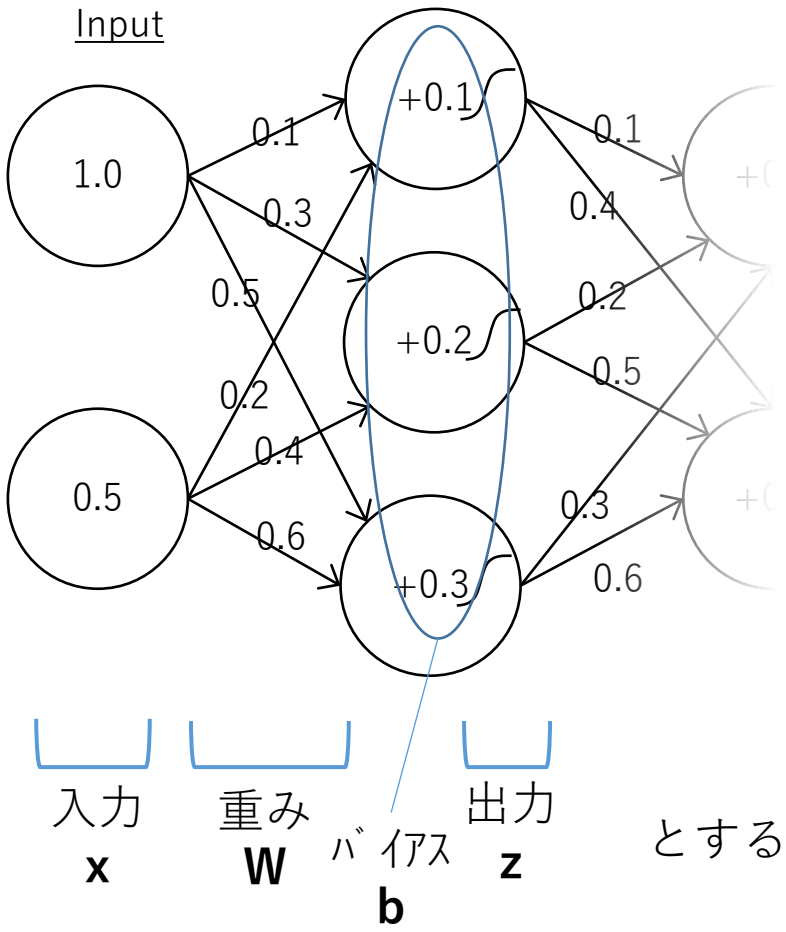
$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} \cdot \begin{pmatrix} 11 \\ 22 \end{pmatrix} = \begin{pmatrix} 77 \\ 209 \\ 341 \end{pmatrix}$$

$$(11\ 22) \cdot \begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} = \boxed{?}$$

ニューラルネットワークを行列で表す

sig() …シグモイド関数

先ほどの模型の前半部分



とすると、

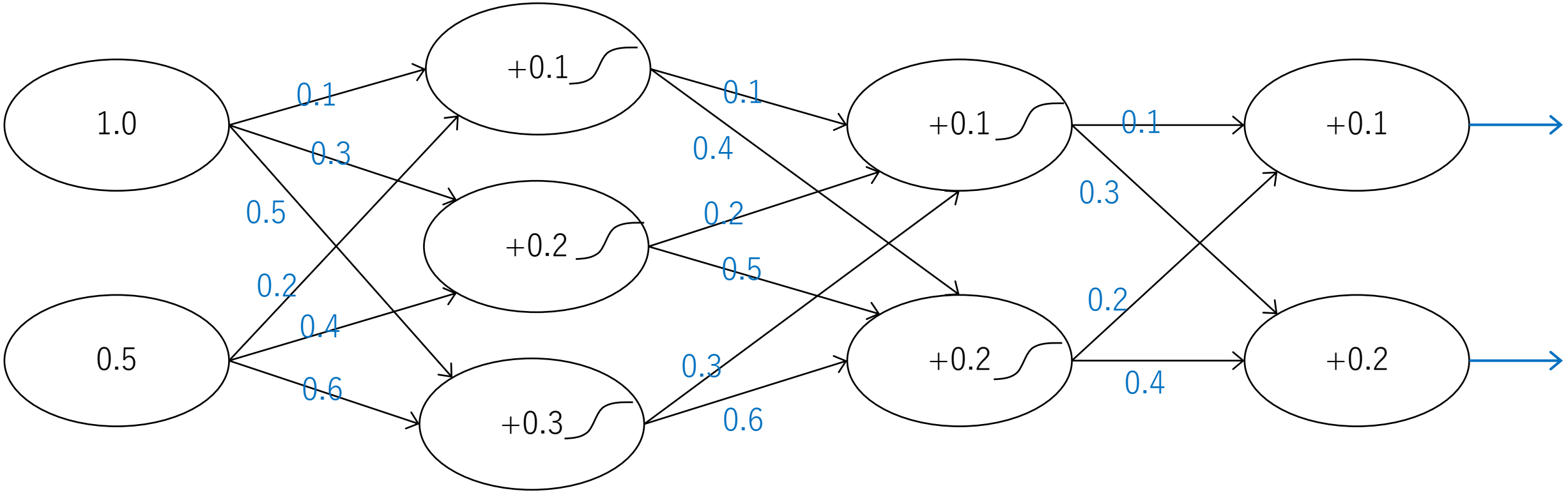
$$\mathbf{z} = \mathbf{sig}(\ \mathbf{W}\ \cdot\ \mathbf{x}\ +\ \mathbf{b}\ )$$

$$= \mathbf{sig}\left(\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix} \cdot \begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix} \right)$$

$$= \mathbf{sig}\left(\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix} \right)$$

$$= \mathbf{sig}\left(\begin{pmatrix} 0.3 \\ 0.7 \\ 1.1 \end{pmatrix} \right) = \begin{pmatrix} 0.574 \\ 0.668 \\ 0.750 \end{pmatrix}$$

先ほどのニューラルネットの例



x

$$\begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix}$$

W1

$$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix}$$

b1

$$\begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}$$

z1

$$\begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$

W2

$$\begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix}$$

b2

$$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$$

z2

$$\begin{pmatrix} ? \\ ? \end{pmatrix}$$

W3

$$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{pmatrix}$$

b3

$$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$$

z3

$$\begin{pmatrix} ? \\ ? \end{pmatrix}$$

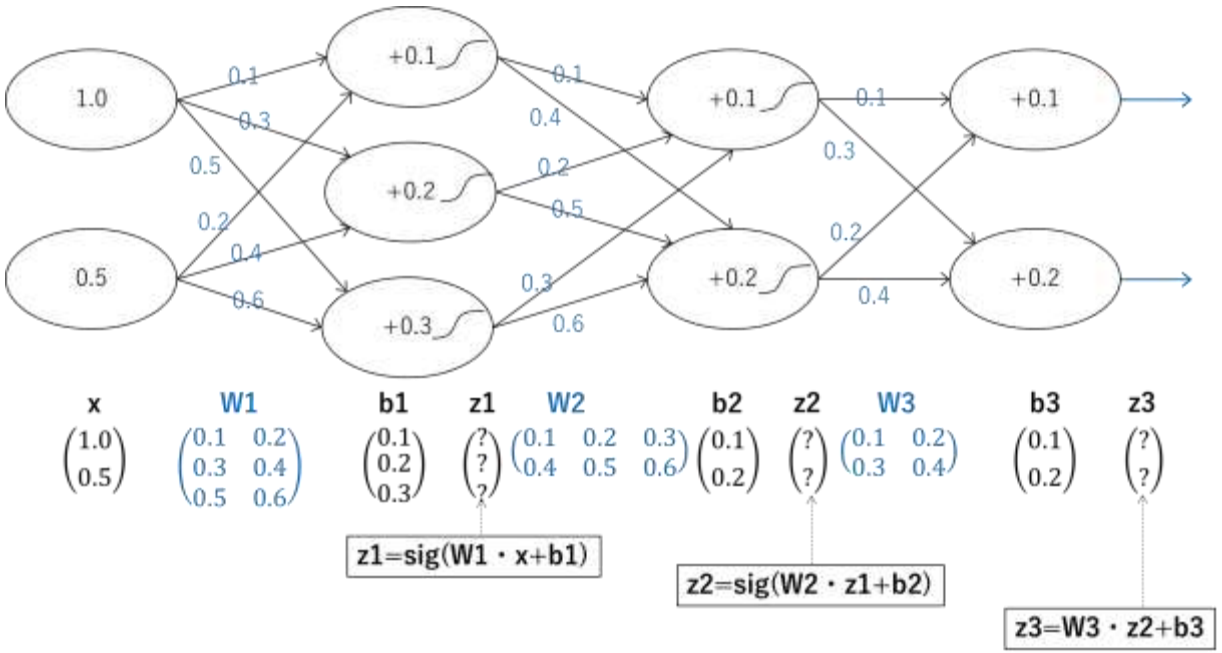
z1=sig(W1・x+b1)

z2=sig(W2・z1+b2)

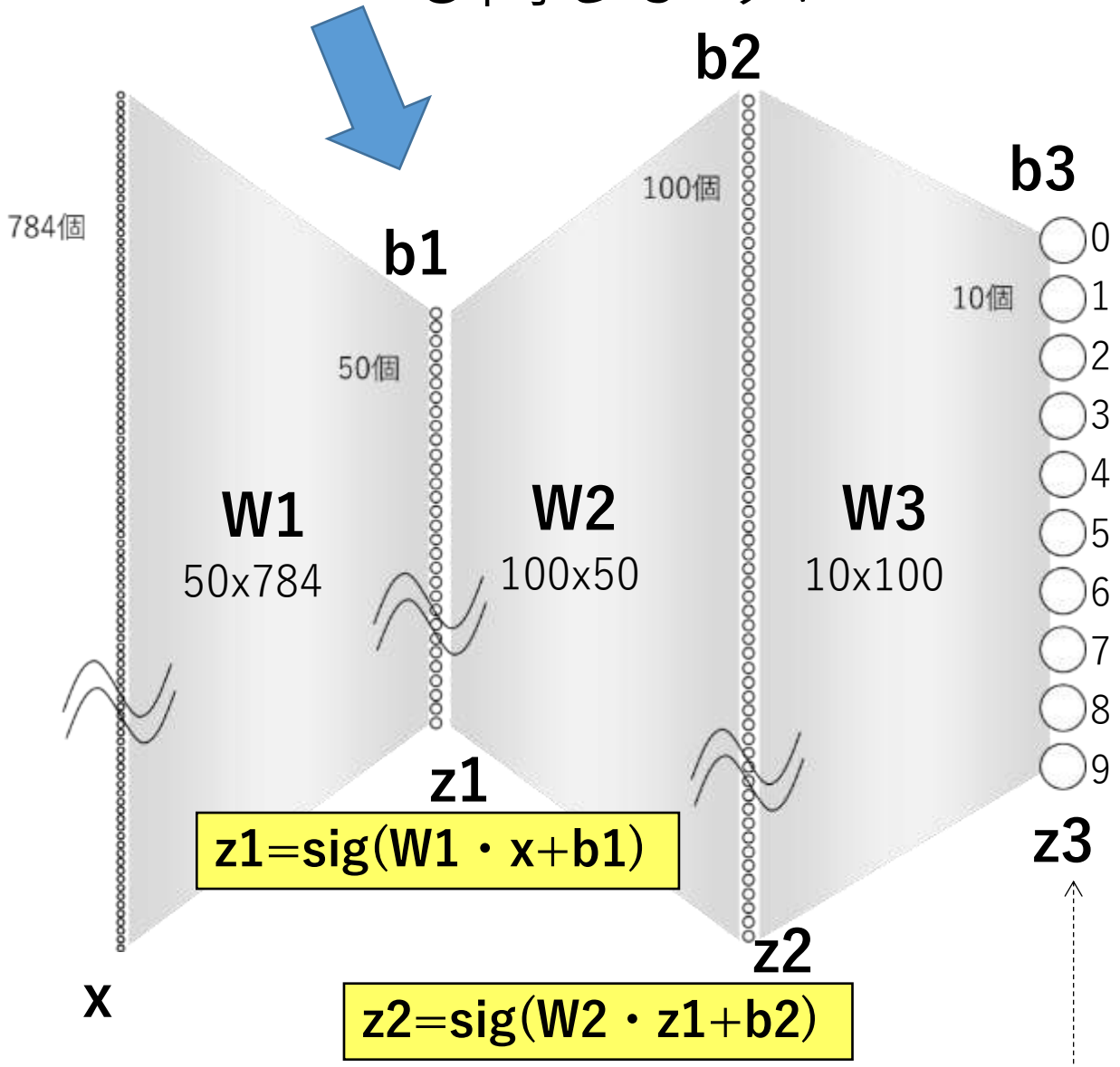
z3=W3・z2+b3

sig() …シグモイド関数

先ほどやったやつ

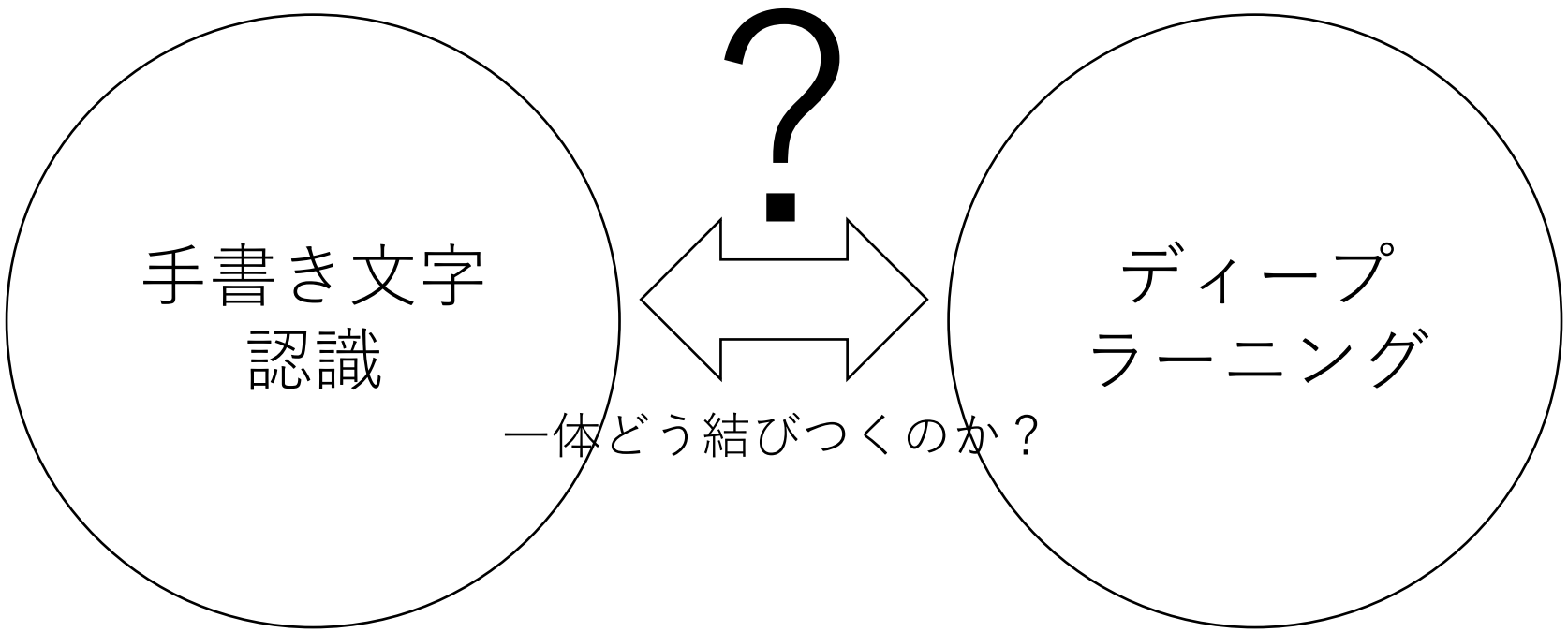


こいつも同じように‥



層やノードの数が増えても、行列を使うとシンプルに表現できる!

手書き文字(数字)認識をさせてみる



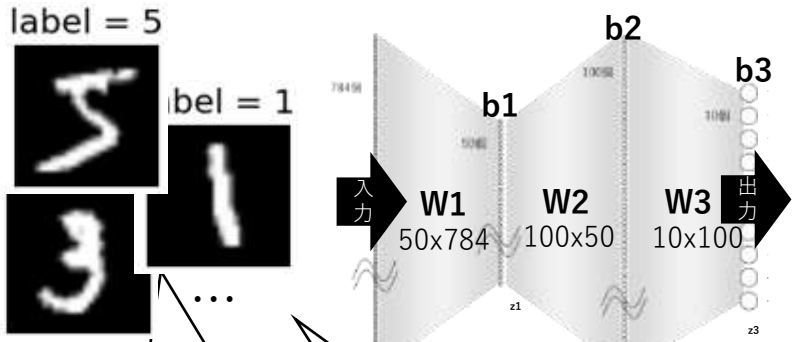
二つのフェーズ－学習と推論

※教師あり学習

学習
Training

正解がわかっている既知のデータを使って※学習させるフェーズ

学習データ
(正解ラベル付き)



これは「3」だぞ

これは「1」だぞ

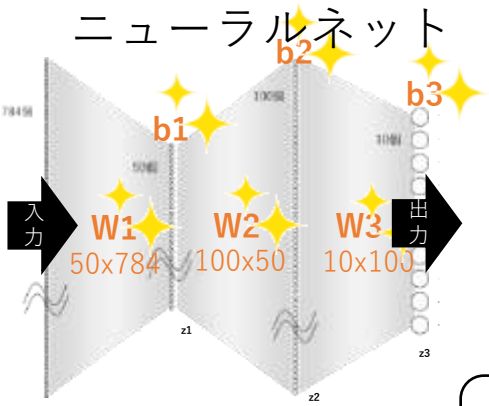
これは「5」だぞ

推論
Inference

未知のデータに対して結果を予測・分類させるフェーズ

未知のデータ

学習済のニューラルネット



これは
な～んだ？

推論の結果

90%ノカクリツデ
3ダトオモイマス‥

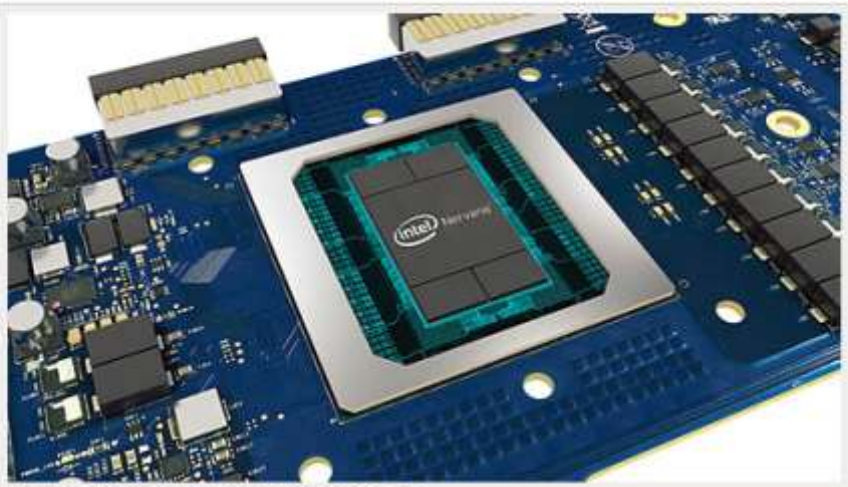
ちょい脱線

Intel、深層学習用プロセッサ「Nervana NNP」

～今年中に49量子ビットの量子演算チップ製造も予告

佐藤 岳大　2017年10月18日 14:41

ツイート リスト いいね！ 223 シェア B! 33 Pocket 86



Intel Nervana Neural Network Processor

米Intelは17日、業界初となる人工知能プロセッサ「Intel Nervana ニューラルネットワークプロセッサ(以下Nervana NNP)」を、2017年中に提供開始することを発表した。

https://pc.watch.impress.co.jp/docs/news/1086773.html

Intel、世界初のスティックPC型AIアクセラレータ

～PCにつないで推論処理を高速実行、79ドル

佐藤 岳大　2017年7月21日 14:08

ツイート リスト いいね！ 930 シェア B! 65 Pocket 176

Movidius Neural Compute Stick

米Intelは20日(米国時間)、スティック型人工知能アクセラレータ「Movidius Neural Compute Stick」を発売した。価格は79ドルで、一部のディストリビュータおよび、7月22日～25日の期間でハワイ ホノルルにて開催されている「Computer Vision and Pattern Recognition (CVPR)」カンファレンスで購入できる。

https://pc.watch.impress.co.jp/docs/news/1071787.html

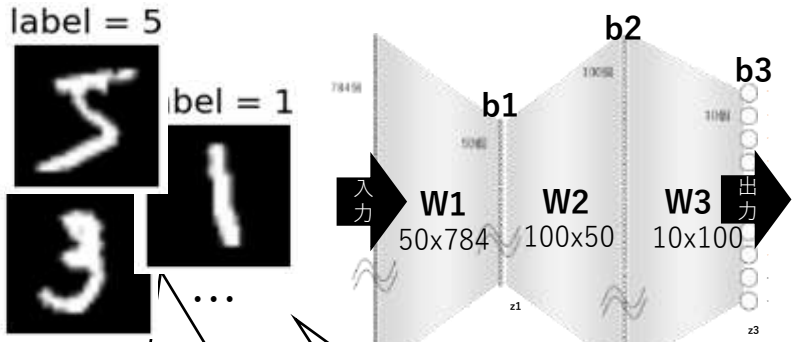
二つのフェーズ－学習と推論

※教師あり学習

学習
Training

正解がわかっている既知のデータを使って※学習させるフェーズ

学習データ
(正解ラベル付き)



これは「3」だぞ

これは「1」だぞ

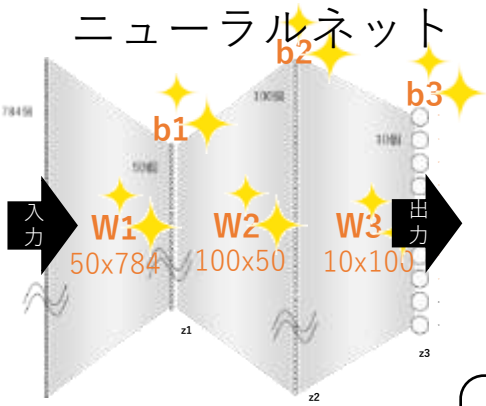
これは「5」だぞ

推論
Inference

未知のデータに対して結果を予測・分類させるフェーズ

未知のデータ

学習済のニューラルネット



これは
な～んだ？

推論の結果

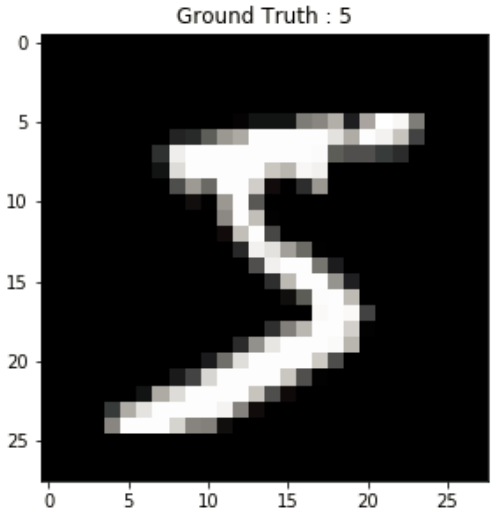
90%ノカクリツデ
3ダトオモイマス‥

データを用意する

MNIST手書き文字データ
http://yann.lecun.com/exdb/mnist/
ディープラーニングのアルゴリズムを評価(ベンチマーク)するためによく使われる

- 60,000の学習用データ
- 10,000の検証用データ

共に正解ラベルを持つが、学習用とは別のデータを用いて推論させることで、精度を評価できる



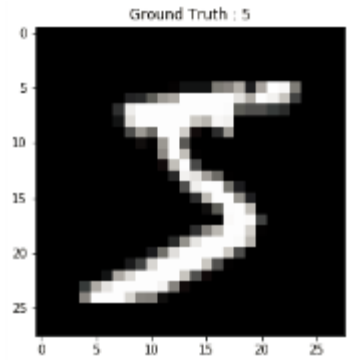
28x28=784個の格子(ピクセル)ごとに0~1の255段階の値で明るさを示すことで手書き文字を数値として表現

ちょい脱線

MNISTデータの他にも、10カテゴリーに分類された画像データCIFAR10や、さらに多くの分類を持つImageNetなど、多数のベンチマーク用データが公開されている
参照）Open Data for Deep Learning (https://deeplearning4j.org/opendata)

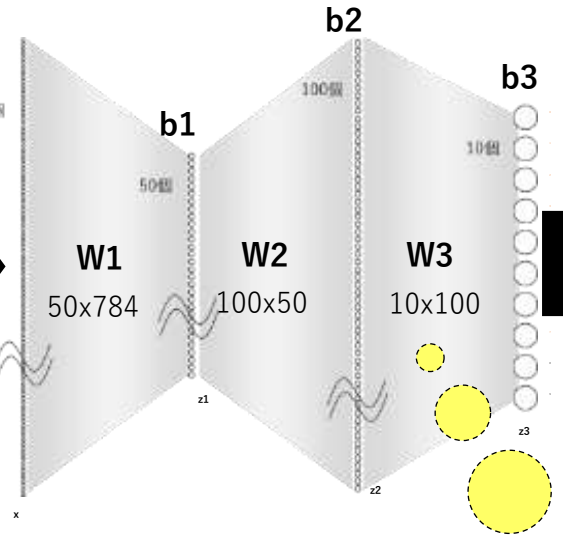
ディープラーニングにおける学習とは

数値化された手書き文字を、ニューラルネットに食わせ‥



28x28=784個の数値

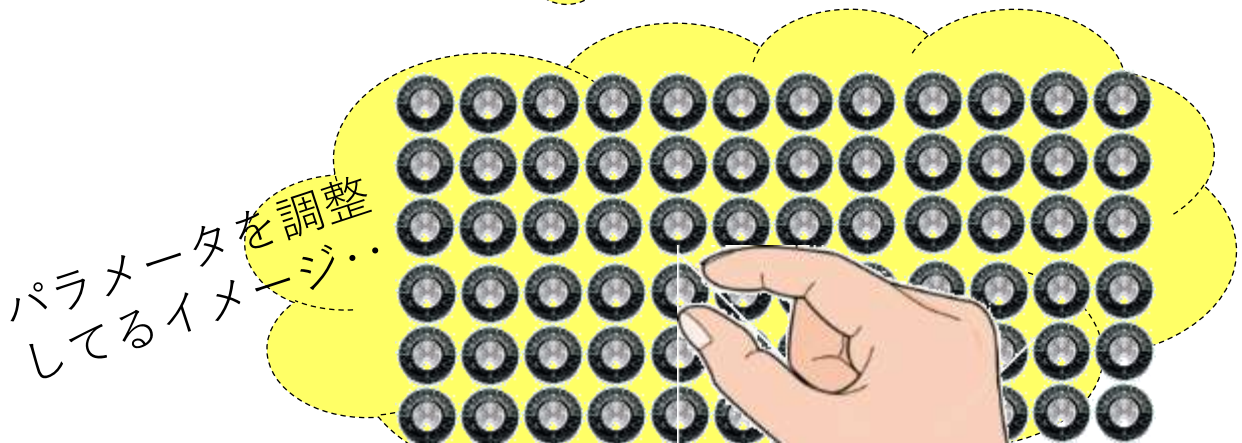
入力



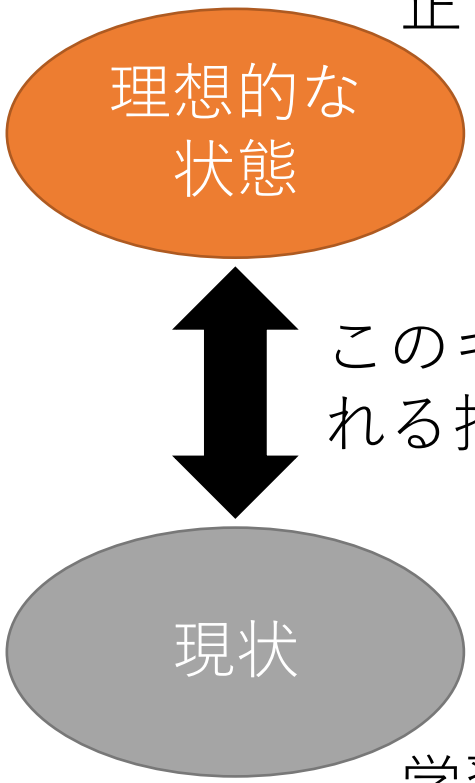
出力

ラベル	確率
0	4%
1	1%
2	3%
3	12%
4	1%
5	**51%**
6	8%
7	5%
8	10%
9	5%

パラメータ(W1, W2, W3, b1, b2, b3—全部で45,350個の数字)を少しずつ変えながら、正解ラベル箇所が大きな確率を示すように、**絶妙なパラメータの組み合わせを探す**プロセス



具体的にどうアプローチするか―指標の定義



正しく数字を判別できる状態

このギャップを示す、計算で求められる指標として損失関数を定義する

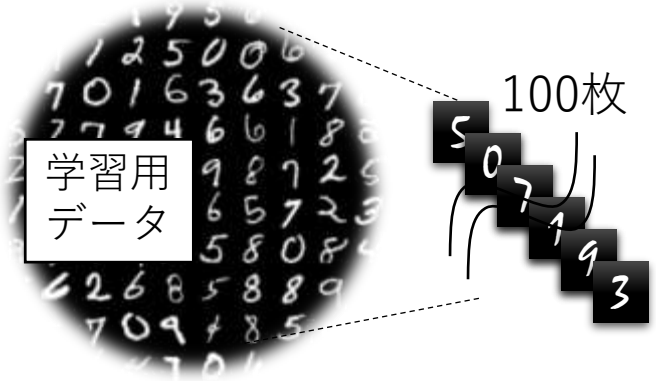
学習の途中段階における
ネットワークのダメさ具合

この損失関数が最小となるパラメータ(W1, W2, W3, b1, b2, b3 ―全部で45,350個の数字)の組み合わせを探す

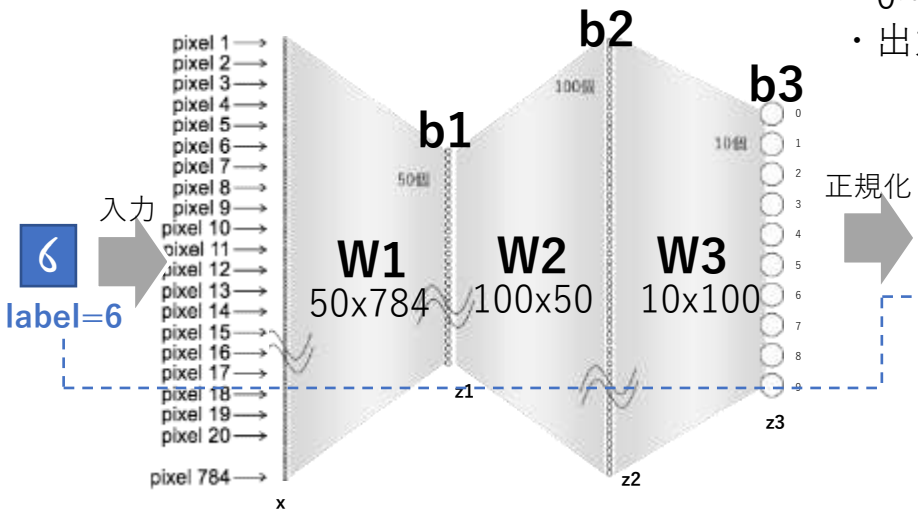
学習の途中段階

損失関数～ダメさ具合の指標

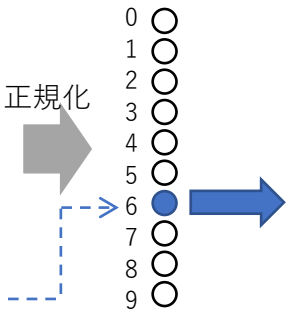
MNIST計6万枚のデータ群から、
ランダムに100枚を選び出す。



一枚ずつニューラルネット
に入力して計算

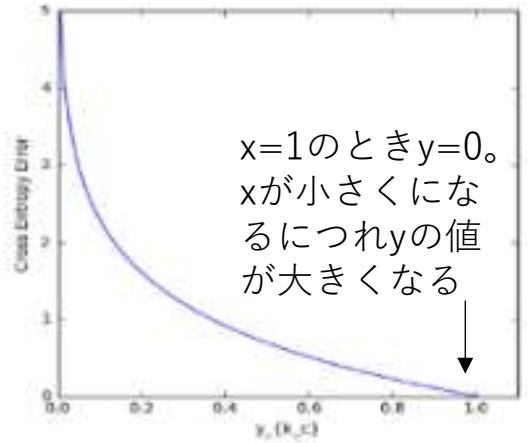


出力を正規化
・0~100%の値
・出力総和=1



正解箇所の値のｴﾝﾄﾛﾋﾟｰ
(-log)を計算

y=-log(x)



100枚分計算して平均を求める。
これが<u>損失関数の値</u>となる。

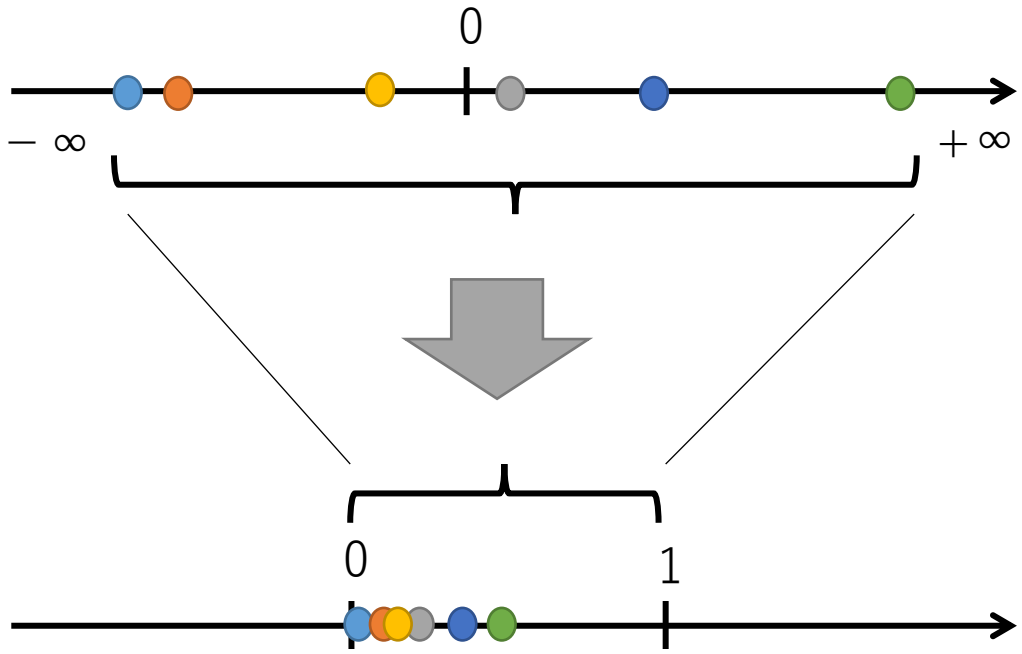
パラメータ**W1, W2, W3, b1, b2, b3**
によって、損失関数の値が決まる

（補足）指数関数を用いた正規化

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

正規化の様子を数直線で表現すると、
大小さまざまな数字について、それぞれの
位置関係は保ったままで、

- 0から1のあいだにギュッと押し込む
- 且つ、値の総和が1になる



Excelシートで実際に試してみた例

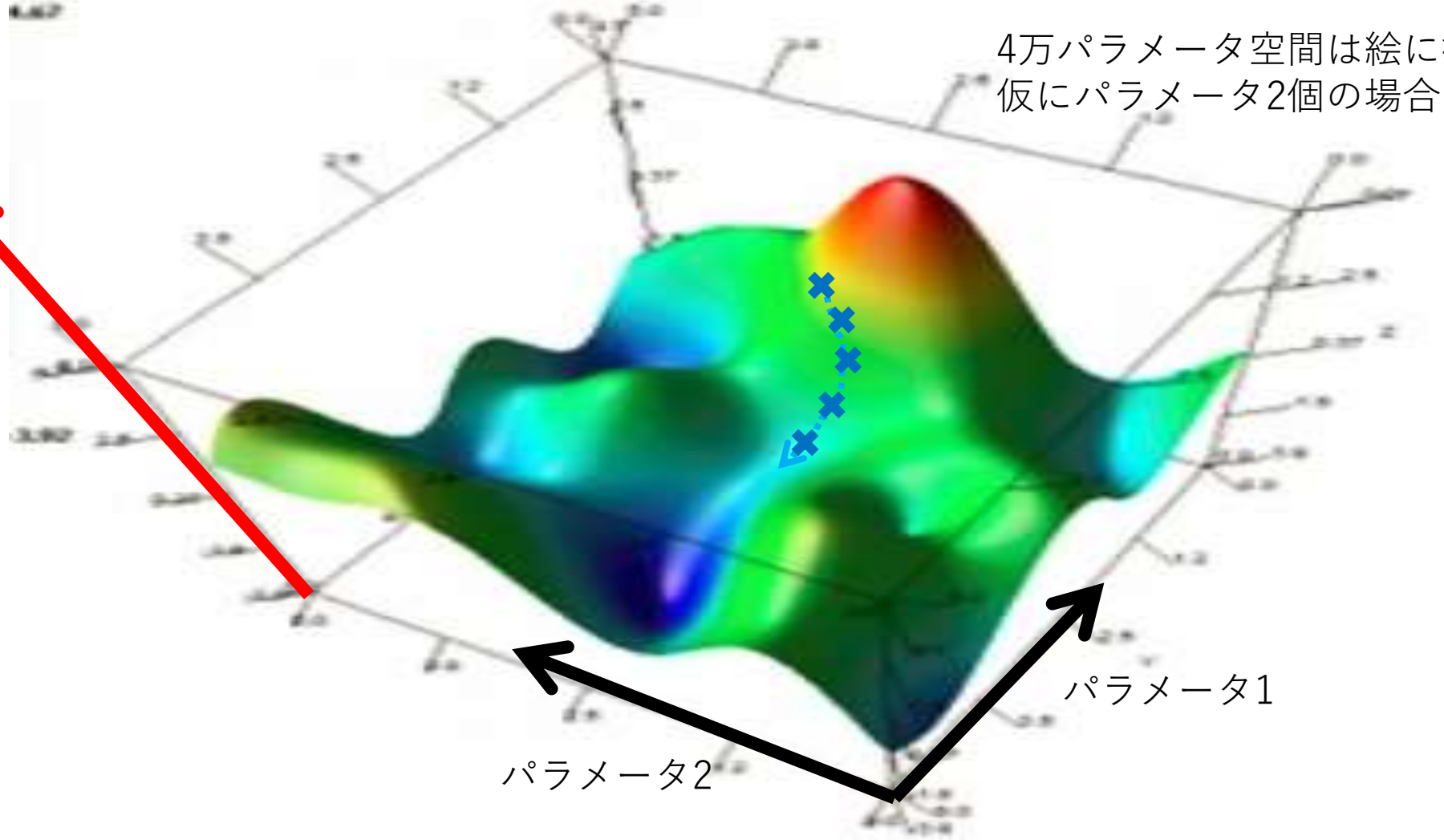
適当な数字
=EXP(左セル)
左セルを%表示

−1.2	0.301194	0.037784	4%
−0.3	0.740818	0.092934	9%
1.3	3.669297	0.460304	46%
0.1	1.105171	0.138641	14%
−1.1	0.332871	0.041758	4%
0.6	1.822119	0.22858	23%
	7.97147		100%

=SUM(列の合計)
=(左セル)÷SUMの値
列のSUM
確かに足したら1

損失関数が一番小さくなるパラメータを探す
＝パラメータ空間(山谷)の底を探す

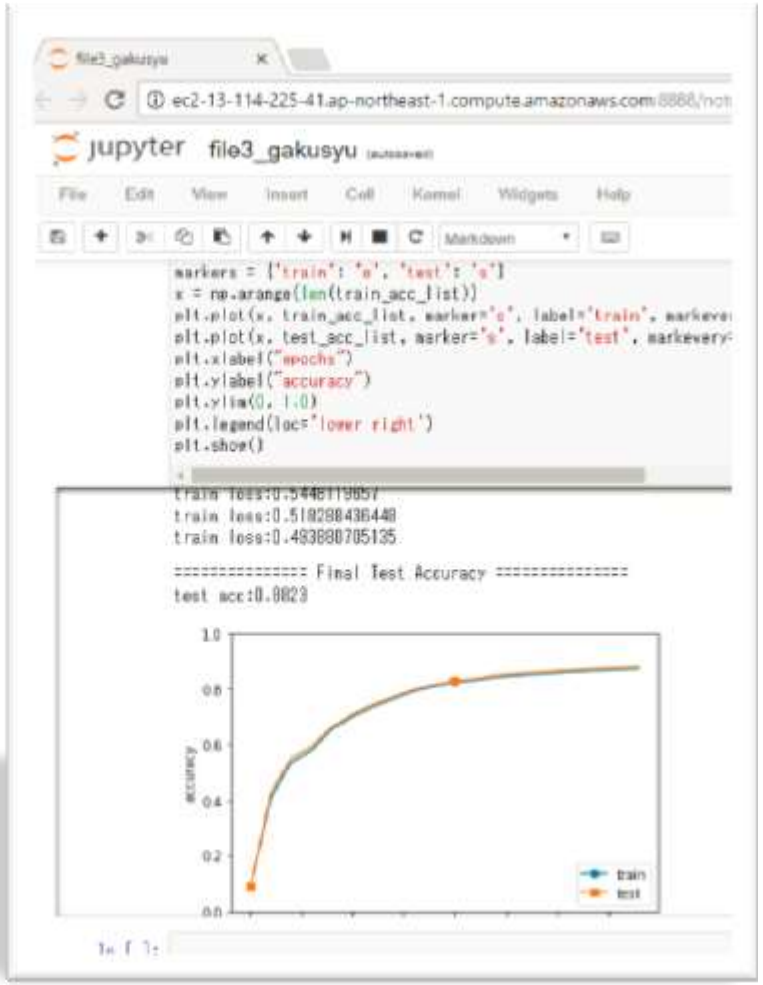
4万パラメータ空間は絵に描けないので、
仮にパラメータ2個の場合のイメージ



$$W \underset{\text{代入}}{\leftarrow} W - \alpha \frac{\partial E}{\partial W}$$

$$b \underset{\text{代入}}{\leftarrow} b - \alpha \frac{\partial E}{\partial b}$$

実際の学習の過程を見てみる

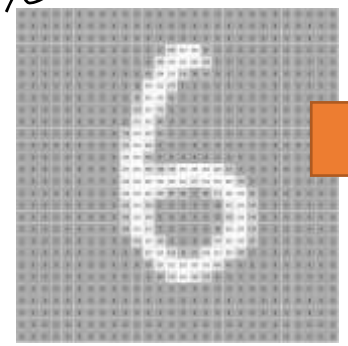


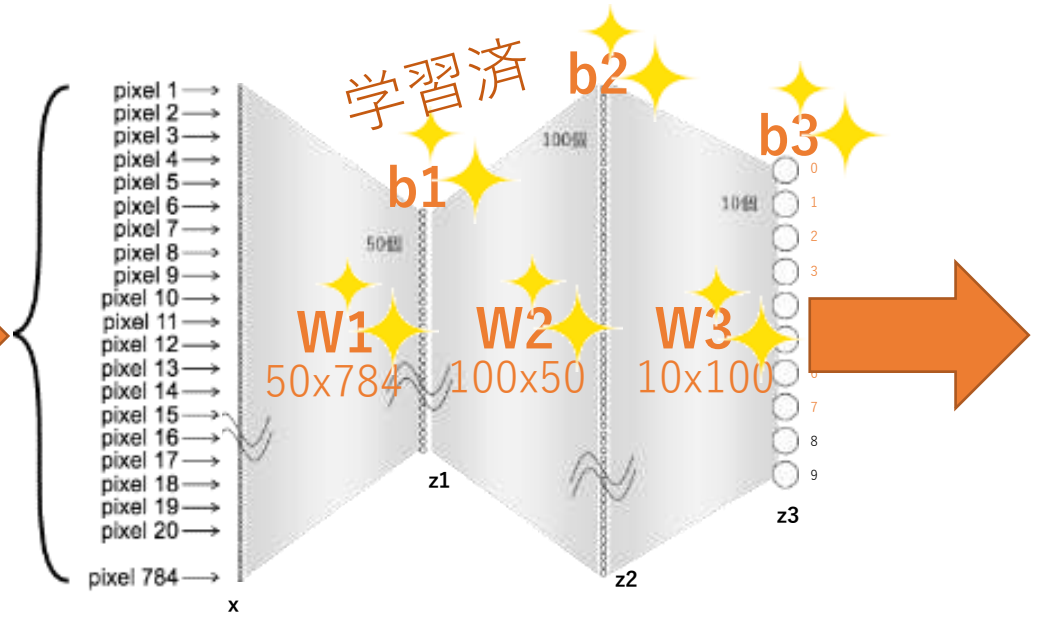
https://github.com/yoshihiroo/programming-workshop/blob/master/deep_learning_jupyter/file3_gakusyu.ipynb

学習済のパラメータを使って、文字認識が正しく行われていることを確かめる

検証用のデータ

例えば「6」

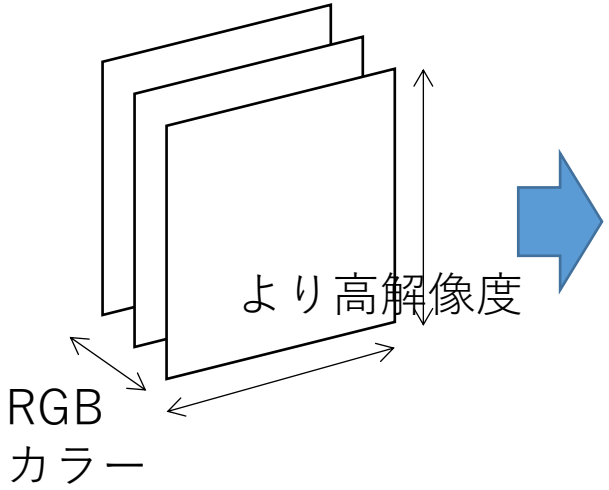




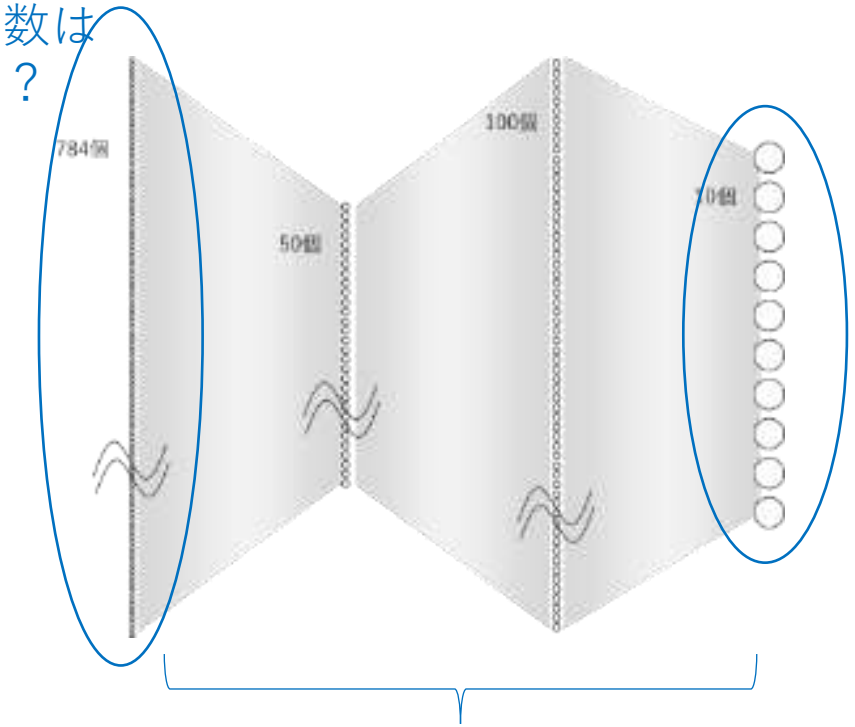
「6」と認識できるか？
実際に試してみる。

https://github.com/yoshihiroo/programming-workshop/blob/master/deep_learning_jupyter/file3_gakusyu.ipynb

より複雑な課題への適用を考えてみる



ノードの数は
どうなる？



0~9の10種類でなく、100、1000…とさらに多くの分類を行うとしたら、ノードの数はどうなる？

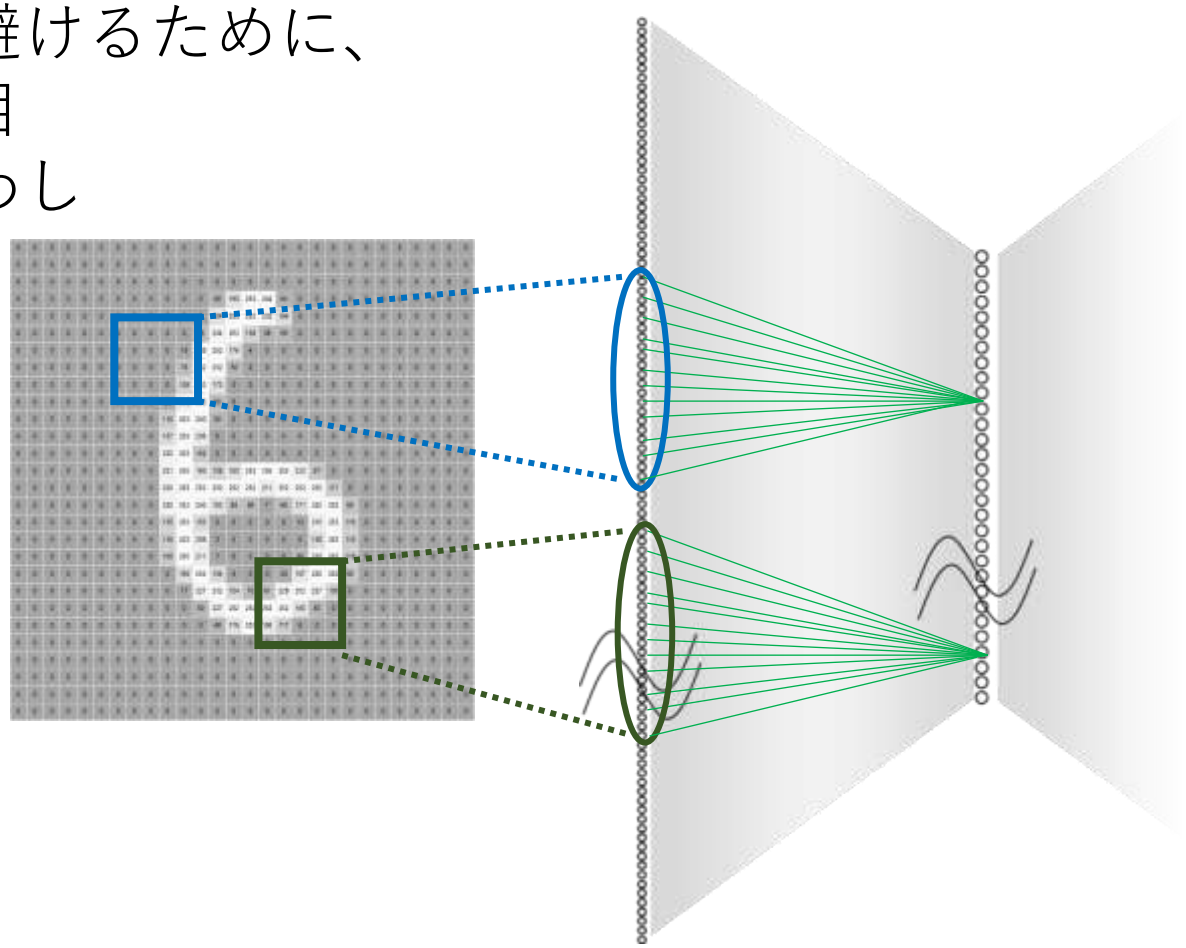
InとOutが大きくなった場足、レイヤーの数やレイヤーあたりのノードの数はどう設計すべきか？

→

その結果、パラメータの数が_____、計算に必要なリソースは_____。

畳み込みニューラルネットワーク (Convolutional Neural Network = CNN)

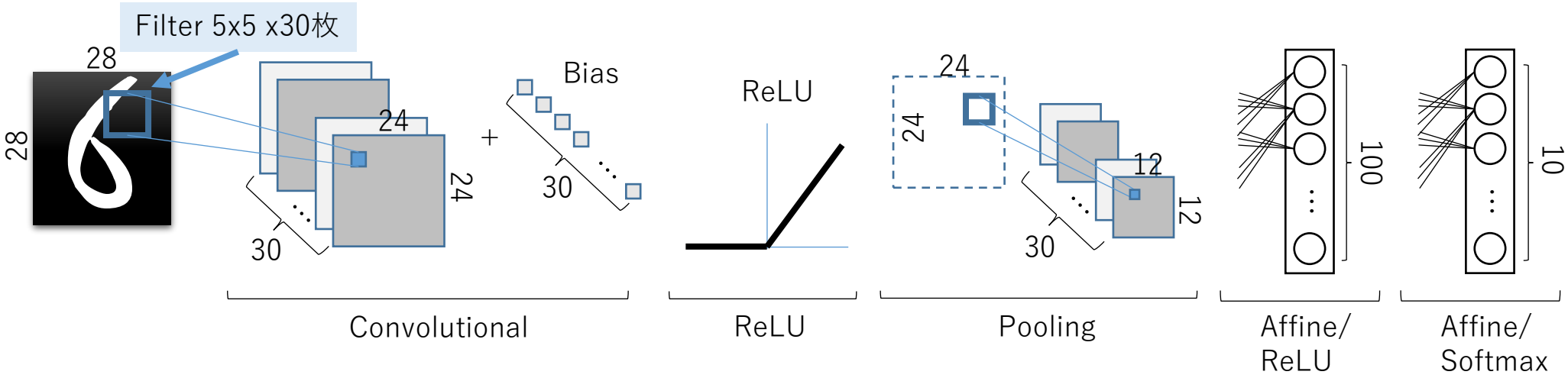
パラメータ数の爆発を避けるために、
・特徴量の局所性に着目
・パラメータの使いまわし



どういう計算をするか

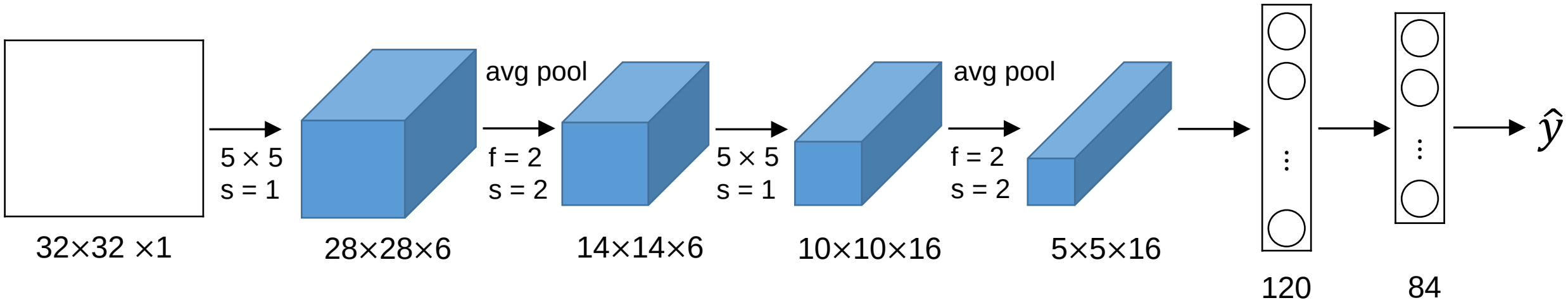
http://brohrer.github.io/how_convolutional_neural_networks_work.html
のパワーポイントを拝借して説明します。

評価に用いる畳み込みニューラルネットワークの構成



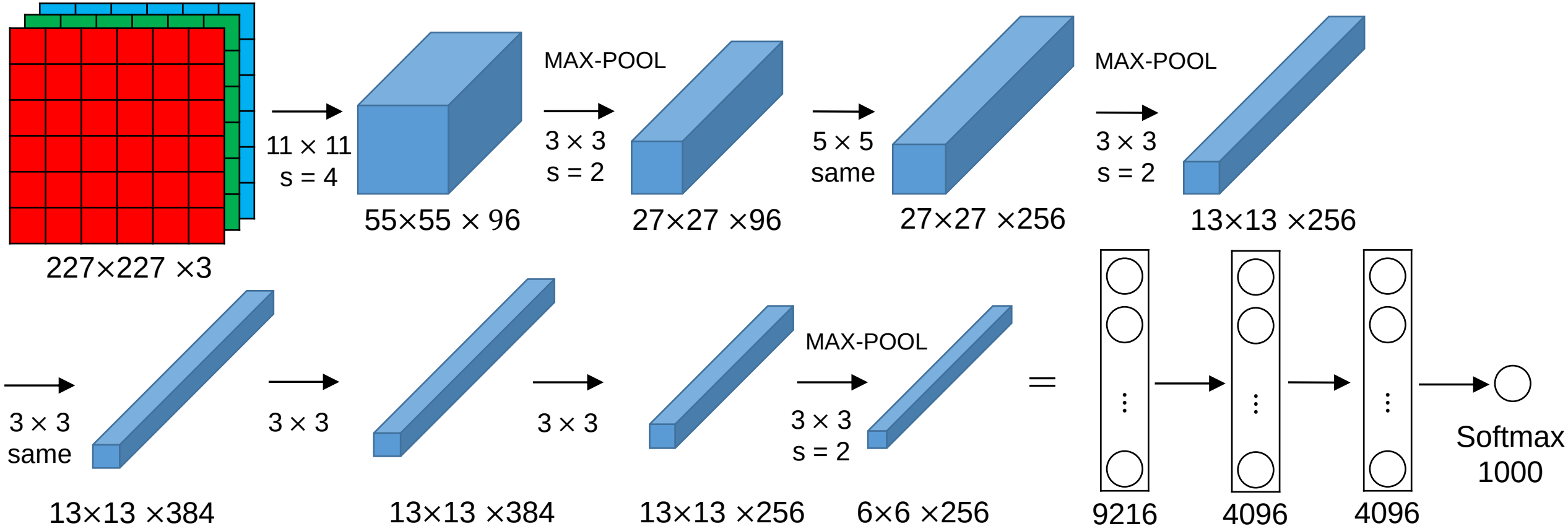
https://github.com/yoshihiroo/programming-workshop/blob/master/deep_learning_jupyter/file5_CNN.ipynb

LeNet - 5



[LeCun et al., 1998. Gradient-based learning applied to document recognition]

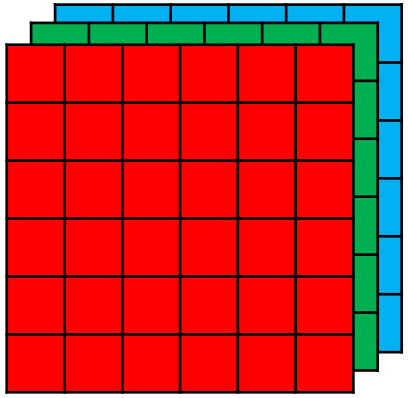
AlexNet



[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

VGG - 16

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2



224×224 ×3 → [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

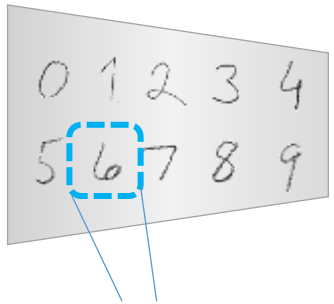
今日の時間配分

0:00 – 2:00	• ラズパイ基本セットアップ • カメラを使った画像配信

2:00 – 3:30	• 座学
	  • ニューラルネットワーク
	  • 手書き文字認識
	  • 畳み込みニューラルネットワーク

3:30 – 4:30	• 手書き文字認識システムの実装とテスト • 物体識別システムの実装とテスト
4:30 – 5:00	• クロージング・振り返り

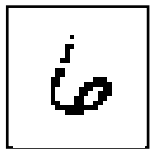
digit_recognition_NN.pyの概要



撮影

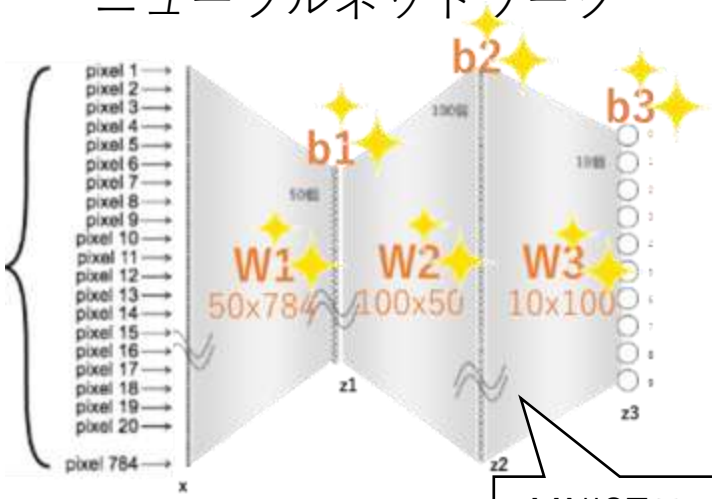
224x224
カラー

画像前処理



28x28
白黒(2値)

ニューラルネットワーク

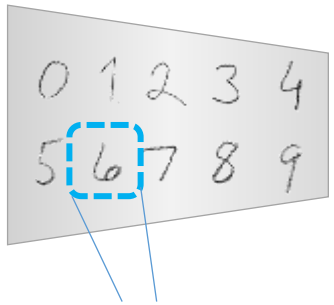


結果の出力

MNISTテストデータで93%の認識率

繰り返す

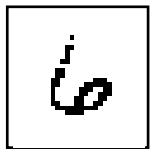
digit_recognition_CNN.pyの概要



撮影

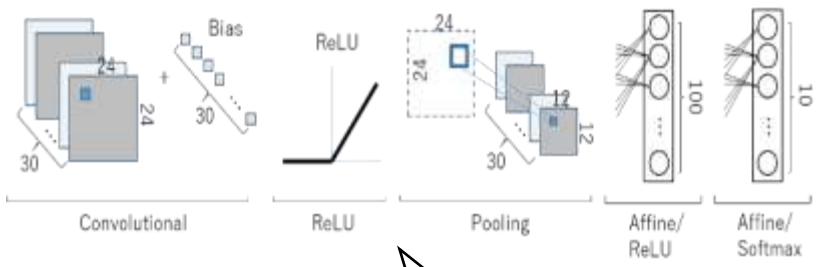
224x224
カラー

画像前処理



28x28
白黒(2値)

畳み込みニューラルネットワーク

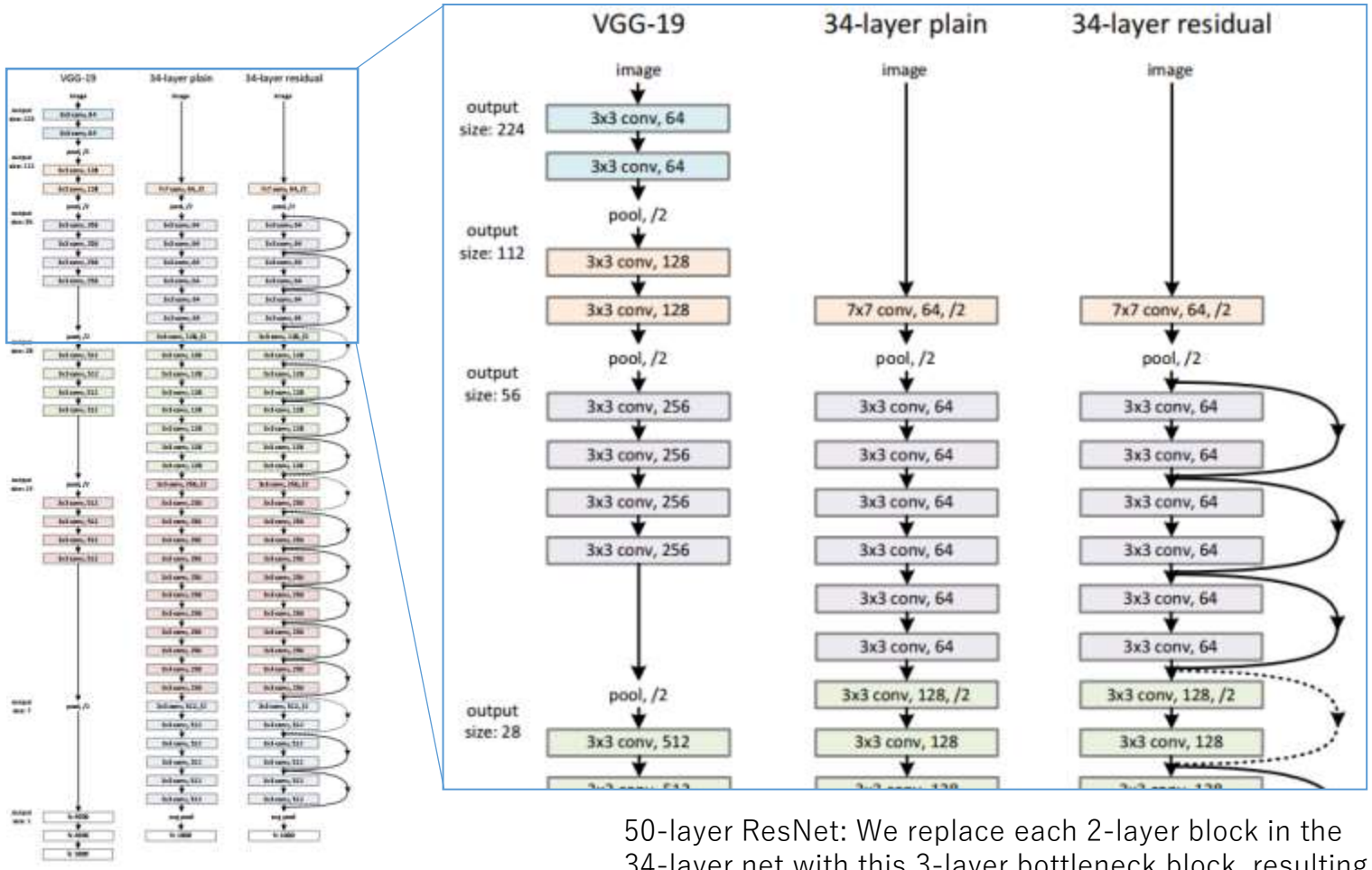


結果の出力

MNISTテスト
データで99%の
認識率

繰り返す

image_classification_resnet50.pyの概要

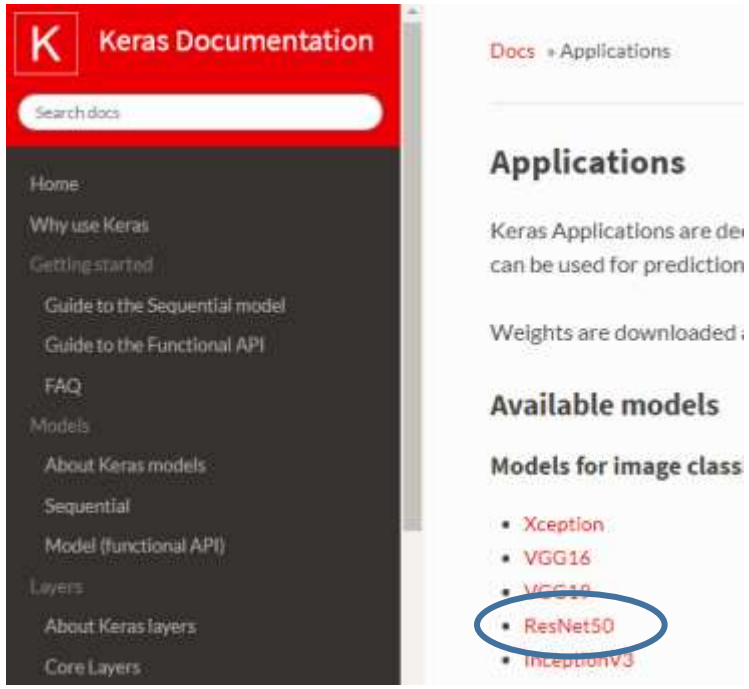


50-layer ResNet: We replace each 2-layer block in the 34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet

https://arxiv.org/pdf/1512.03385.pdf

Dec 2015

https://keras.io/applications/



上記サイトで公開されているKerasライブラリを用いたResNet50の実装コードをベースに、カメラ画像を取り込むように変更。

image_classification_mobilenet.pyの概要

Google Research Blog

The latest news from Research at Google

MobileNets: Open-Source Models for Efficient On-Device Vision

Wednesday, June 14, 2017

Posted by Andrew G. Howard, Senior Software Engineer and Menglong Zhu, Software Engineer

(Cross-posted on the Google Open Source Blog)

Deep learning has fueled tremendous progress in the field of computer vision in recent years, with neural networks repeatedly pushing the frontier of visual recognition technology. While many of those technologies such as object, landmark, logo and text recognition are provided for internet-connected devices through the Cloud Vision API, we believe that the ever-increasing computational power of mobile devices can enable the delivery of these technologies into the hands of our users,

https://research.googleblog.com/2017/06/mobilenets-open-source-models-for.html

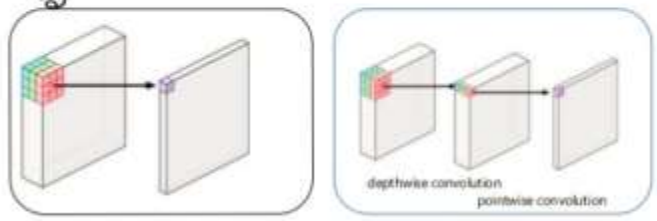
概要

- CNNの進歩しているが、精度に比べサイズと速度の面での進歩が少ない
- サイズを小さくする研究はあったが、速度を考慮しているものは少ない
- ロボットなどの実世界でのアプリケーションでは速度が必要になる

- 効率的なネットワークアーキテクチャと2つのハイパーパラメータを提案
  - サイズを小さく、処理速度を速くする

Depthwise separatable convolution

- 畳み込みを空間方向の畳込み(depthwise convolution)とチャネル方向の畳込み(pointwise convolution, 1*1 convolution)に分ける



通常の畳込み　　Depthwise separatable convolution

http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/ より

https://www.slideshare.net/harmonylab/mobilenet-81645825