

# AI 基礎セミナー

## 第2回 機械学習のモデルとアルゴリズム

### 改訂履歴

日付	担当者	内容
2021/05/08	M. Takeda	Git 公開
2021/10/03	M. Takeda	「(3.3) 様々な活性化関数」のグラフ差替え
2021/12/26	M. Takeda	「(4.5) リカレントニューラルネットワーク (Recurrent Neural Network、RNN)」の説明更新

### 目次

- (1) はじめに
- (2) 機械学習とAI
  - (2.1) ディープラーニング、ニューラルネットワーク、機械学習
  - (2.2) 機械学習アルゴリズム
- (3) 脳神経系のモデル化
  - (3.1) 神経細胞間の信号伝達の様子
  - (3.2) 神経細胞間の信号伝達のモデル化
  - (3.3) 様々な活性化関数
- (4) 様々なニューラルネットワーク
  - (4.1) パーセプトロン (Perceptron、P)
  - (4.2) 順伝播型ニューラルネットワーク (Feed forward neural networks、FF、FFNN)
  - (4.3) ラジアル基底関数ネットワーク (Radial Basis Network、RBN)
  - (4.4) 深層順伝播型ニューラルネットワーク (Deep Feed forward neural networks、DFF)

- (4.5) リカレントニューラルネットワーク (Recurrent Neural Network、RNN)
- (4.6) Long/Short Term Memory (LSTM)
- (4.7) Gated Recurrent Unit (GRU)
- (4.8) Autoencoder (AE)
- (4.9) Variational Autoencoder (VAE)
- (4.10) Denoising Autoencoder (DAE)
- (4.11) Sparse Autoencoder (SAE)
- (4.12) マルコフ連鎖 (Markov chain、MC)
- (4.13) ホップフィールド・ネットワーク (Hopfield network、HN)
- (4.14) ボルツマンマシン (Boltzmann Machine、BM)
- (4.15) 制限ボルツマンマシン (Restricted Boltzmann Machine、RBM)
- (4.16) 深層信念ネットワーク (Deep Belief Network、DBN)
- (4.17) 深層畳み込みニューラルネットワーク (Deep Convolutional neural network、DCN)
- (4.18) 逆畳み込みネットワーク (Deconvolutional Network、DN)
- (4.19) Deep Convolutional Inverse Graphics Network (DCIGN)
- (4.20) 敵対的生成ネットワーク (Generative Adversarial Network、GAN)
- (4.21) リキッド・ステート・マシン (Liquid State Machine、LSM)
- (4.22) エクストリーム・ラーニング・マシン (Extreme Learning Machine、ELM)
- (4.23) エコー・ステート・ネットワーク (Echo State Network、ESN)
- (4.24) Deep Residual Network (DRN)
- (4.25) コホーネンネットワーク (Kohonen Network、KN)
- (4.26) サポートベクターマシン (Support Vector Machine、SVM)
- (4.27) ニューラルチューリングマシン (Neural Turing Machine、NTM)
- (5) 確認問題
- (6) 確認問題回答用紙

## (1) はじめに

第2回では、脳神経系のモデル化と、様々な機械学習モデルとそのアルゴリズムについて概観します。

## (2) 機械学習とAI

### (2.1) ディープラーニング、ニューラルネットワーク、機械学習

- ・「人工知能（artificial intelligence、略称：AI）」とは、人間の知的能力をコンピュータ上で実現する、様々な技術・ソフトウェア・コンピューターシステムを言います。
- ・「機械学習（Machine Learning、略称：ML）」とは、「人間が自然に行っている学習能力と同様の機能をコンピュータで実現しようとする技術・手法」のことです。「機械学習」は「人工知能」の構成要素の一つとっていいでしょう。
- ・「機械学習」の一分野として「ニューラルネットワーク（Neural Network、神経回路網、略称：NN）」があります。生物の脳神経系が強力な学習能力を持つことから、脳神経系に見られるいくつかの特性を、計算機上のシミュレーションによって表現することを目指したものが「ニューラルネットワーク」です。
- ・「ニューラルネットワーク」の中間層（隠れ層）を多層化したモデルを用いた機械学習を、「ディープラーニング（深層学習、しんそうがくしゅう、Deep Learning、略称：DL）」と総称します。「多層」の層数は、狭義には4層以上とのことです。

#### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/人工知能>

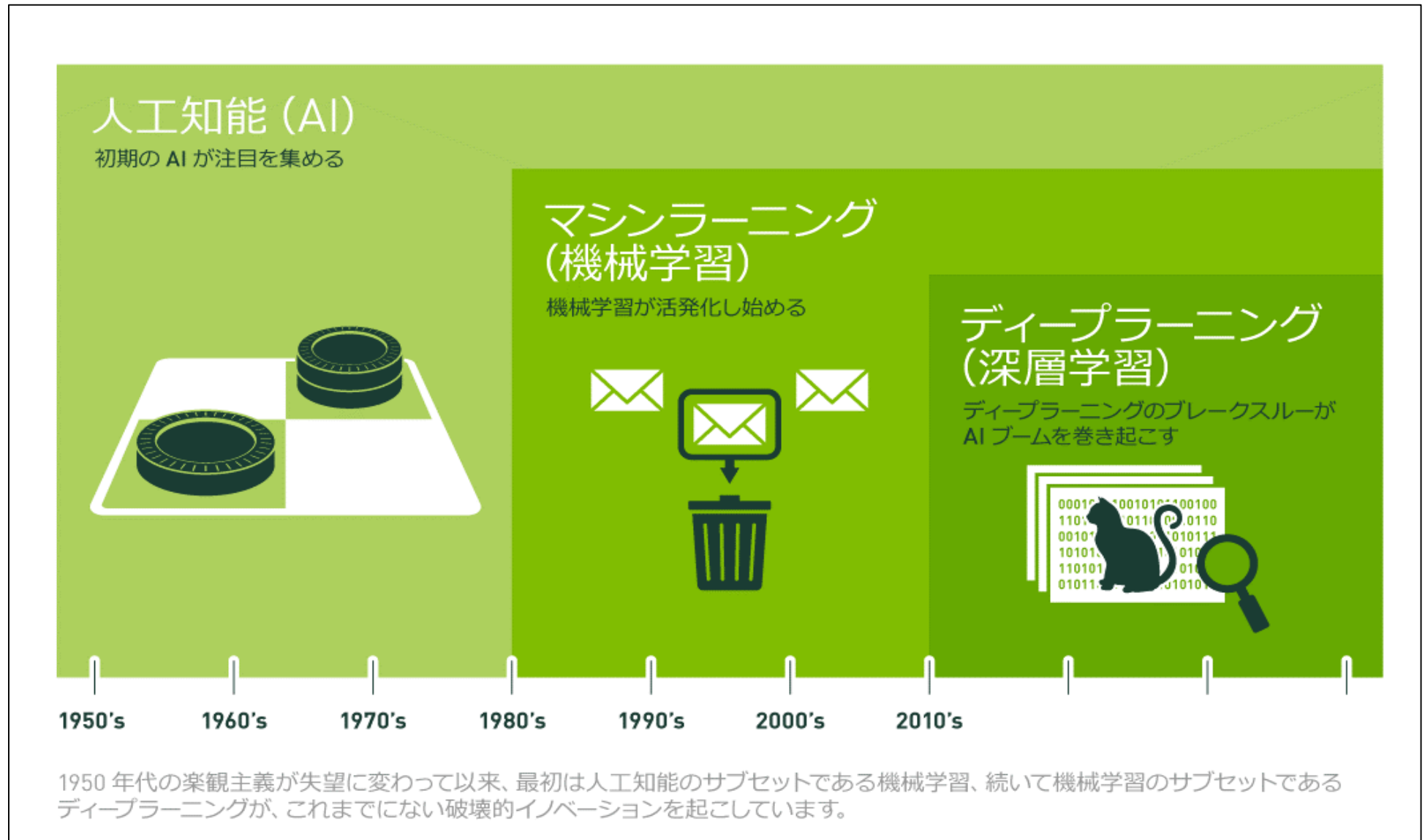
⇒ <https://ja.wikipedia.org/wiki/機械学習>

⇒ <https://ja.wikipedia.org/wiki/ニューラルネットワーク>

⇒ <https://ja.wikipedia.org/wiki/ディープラーニング>

「深層学習 Deep Learning」 近代科学社 人工知能学会監修 2015年11月

- ・参考までに、NVIDIA のブログから上記の関係を示す記事があるので、掲載します。

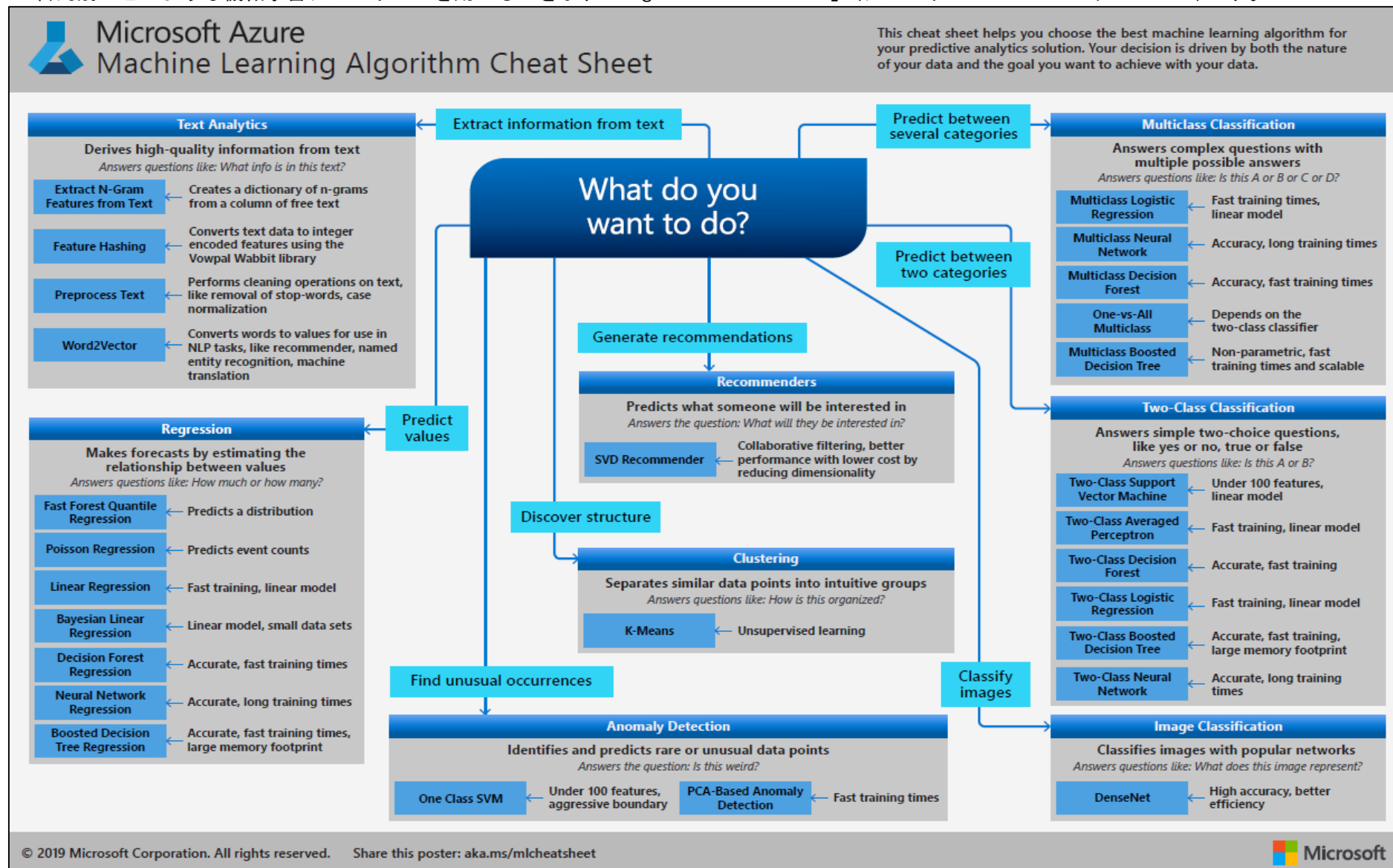


【出典・参考】

⇒ <https://blogs.nvidia.co.jp/2016/08/09/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

## (2.2) 機械学習アルゴリズム

- ・ 機械学習アルゴリズムには様々なものがあり、目的に応じて使い分けます。
- ・ 以下は、マイクロソフトの機械学習サービス「Microsoft Azure Machine Learning」のホームページに公開されており、目的別にどのような機械学習アルゴリズムを用いるかを示す「Algorithm Cheat Sheet」(アルゴリズムのカニングペーパー)です。



⇒ <https://docs.microsoft.com/ja-jp/azure/machine-learning/algorithm-cheat-sheet> (azure-machine-learning-algorithm-cheat-sheet-nov2019.pdf)

- ・以下に、機械学習アルゴリズムの分類について紹介します(梅田弘之さんの記事からの引用)。

【出典・参考】

機械学習とアルゴリズム (梅田 弘之さん) ⇒ <https://thinkit.co.jp/article/13280>

### (2.2.1) 分類 (Classification)

- ・「分類」は、名前（ラベル）を付けた学習データをもとに、それらの特徴点を見つけて学習し、データを分類するものです。
- ・「分類」には、複数のカテゴリーに分類する「マルチ分類 (Multi-class classification)」と  
2つのカテゴリーに分類する「2クラス分類 (Two-class classification)」があります。
- ・「分類」は「教師あり学習」です。

目的	用いる機械学習アルゴリズム
2クラス分類 (Two-class classification)	・ ロジスティック回帰 (Logistic Regression) (※1)
	・ ランダムフォレスト (Random(Decision) Forest)
	・ デシジョンジャングル (Decison Jungle)
	・ ブースト 決定木 (Boosed decision tree)
	・ ニューラルネットワーク (Neural network) (※1)
	・ 平均化パーセプトロン (Averaged perceptron)
	・ サポートベクターマシーン (Support vector machine(SVM)、SVC(線形SVM)、Linear SVC(RBM SVM)) (※2)
	・ ローカル詳細SVM (Locally deep SVM)
	・ ベイズポイントマシン (Bayes point machine) 単純ベイズ (Naive Bayes)
マルチ分類 (Multi-class classification)	・ ロジスティック回帰 (Logistic Regression) (※1)
	・ ランダムフォレスト (Random(Decision) Forest)
	・ デシジョンジャングル (Decison Jungle)
	・ ニューラルネットワーク (Neural network) (※1)
	・ 一対全多クラス (One-v-all multiclass)
	・ k近傍法 (k-nearest neighbors)

## (2.2.2) 回帰 (Regression)

- ・「回帰」は、実績データをもとに、入力データと出力データの数値間の関連性を導き出し、入力データに対する出力データを数値で予測するものです。
- ・「回帰」は「教師あり学習」です。

目的	用いる機械学習アルゴリズム
回帰 (Regression)	・ 線形回帰 (Linear (Ordinary) Regression) (※1)
	・ ベイズ線形回帰 (Bayesian Linear Regression)
	・ ランダムフォレスト (Random (Decision) Forest)
	・ ブースト 決定木 (Boosed decision tree)
	・ 高速フォレスト分布 (Fast forest quantile)
	・ ニューラルネットワーク (Neural network) (※1)
	・ ポアソン回帰 (Poisson Regression)
	・ サポートベクトル序数回帰 (Ordinal Regression)
	・ リッジ回帰 (Ridge Regression)
	・ ラッソ回帰 (Lasso Regression)
	・ サポートベクター回帰 (Support vector Regression (SVR))
	・ One Class SVM

## (2.2.3) クラスタリング (Clustering)

- ・「クラスタリング (Clustering)」もまた、分類するための手法です。
- ・分類するという点では「分類」と同じですが、「クラスタリング」は「分類」と異なり、似たデータを集めてデータ構造を発見する「教師なし学習」です。

目的	用いる機械学習アルゴリズム
クラスタ分析 (Clustering)	・ k-means (k平均法) (※1)
	・ 混合ガウス分布 (GMM (Gaussian mixture models)) (※1)
	・ スペクトラルクラスタリング (spectral clustering)



#### (2.2.4) 次元削減 (dimensionality reduction)

- ・「次元削減 (dimensionality reduction)」は、多くの次元を持ったものを、その意味を保持しながら（落としても影響のない次元を削減して）少ない次元にする手法です。

目的	用いる機械学習アルゴリズム
次元削減 (dimensionality reduction)	・ 主成分分析 (PCA)
	・ 非負値行列因子分析 (NMF)
	・ LDA
	・ Deep Learning

#### (2.2.5) 異常検出 (Anomaly Detection)

- ・「異常検出 (Anomaly Detection)」は、現在の異常を検知したり、未来の予知をして保全に備えたりする為の技術です。

目的	用いる機械学習アルゴリズム
異常検出 (Anomaly Detection)	・ サポートベクターマシーン (Support vector machine (SVM))
	・ PCAによる異常検出 (PCA-based anomaly detection)

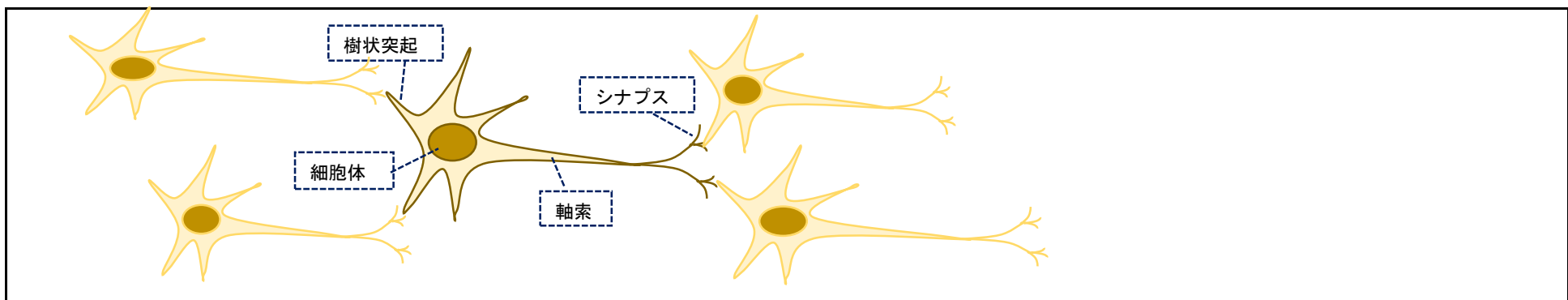
(※1) 本セミナーでも詳説します。

(※2) 本セミナーでは概説します。

### (3) 脳神経系のモデル化

#### (3.1) 神経細胞間の信号伝達の様子

- ・ 20世紀の初頭に、人間の脳は「ニューロン (neuron)」と呼ばれる神経細胞が、「シナプス (synapse)」と呼ばれる結合部位を介して、多数結合してできているネットワークであることが示されました (下図)。
- ・ 神経細胞は主に、細胞核のある細胞体、他の細胞からの入力を受ける樹状突起、他の細胞に出力する軸索から構成されています。樹状突起が受け取った信号は細胞体で処理され、軸索を通して、次の神経細胞に伝達されます。



- 細胞体 : 細胞核とそれを取り巻く細胞質からなり、細胞としての機能はほとんどここで行われます。  
樹状突起と軸索が会合する部位でもあります。
- 樹状突起 : 細胞体から木の枝のように分岐しながら広がる構造であり、他の神経細胞などから信号を受け取る働きをします。  
一つの神経細胞に、樹状突起は複数あります。
- 軸索 : 細胞体から延びている突起状の構造で、神経細胞において信号の出力を担います。  
一つの神経細胞に、軸索は基本的には一本だけあります。
- シナプス : 前の細胞の軸索終末と後ろの細胞の樹状突起の間の情報を伝達する部分にある、伝達構造です。

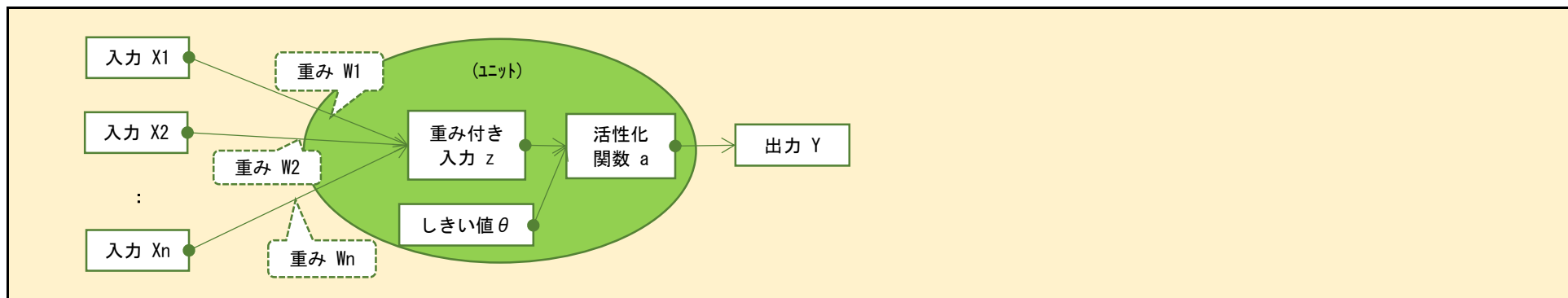
#### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/神経細胞>

「深層学習 Deep Learning」 近代科学社 人工知能学会監修 2015年11月

### (3.2) 神経細胞間の信号伝達のモデル化

- 個々の神経細胞間の信号伝達の様子を以下のようにモデル化します。これを「形式ニューロン」、「ユニット」または「セル」と呼びます。



- 関連する変数名・関数名は、以下のとおり：

$X_i$  : ニューロン  $i$  ( $1 \sim n$ ) からの入力信号

$W_i$  : ニューロン  $i$  ( $1 \sim n$ ) からの入力信号  $X_i$  の重み

$\theta$  : 入力信号の重み付き線形和 ( $\sum_{i=1}^n (X_i * W_i)$ ) に対し、出力信号の有無を判別する為のしきい値

$b$  : バイアス (しきい値  $\theta$  の符号を変えたもの  $b = -\theta$ )

$z$  : 重み付き入力

$a$  : 活性化関数 (activation function、伝達関数 (transfer function) ともいいます)

$Y$  : 出力信号

と表現した時、重み付き入力  $z$  は以下の式で与えられます：

$$\begin{aligned} z &= \{ X_1 * W_1 + X_2 * W_2 + \dots + X_n * W_n \} - \theta \\ &= \{ \sum_{i=1}^n (X_i * W_i) \} - \theta \\ &= \{ \sum_{i=1}^n (X_i * W_i) \} + b \end{aligned} \quad \left\{ \begin{array}{l} \text{入力信号の重み付き線形和} < \text{しきい値 } \theta \text{ ならば、} z < 0 \\ \text{入力信号の重み付き線形和} \geq \text{しきい値 } \theta \text{ ならば、} z \geq 0 \end{array} \right.$$

- 出力信号  $Y$  は、重み付き入力  $z$  を入力変数として、活性化関数  $a$  で、以下の式で与えられます：

活性化関数  $a$  は、重み付き入力  $z$  に応じて、出力信号  $Y$  の大きさを規定する関数です：

$$\begin{aligned} Y &= a(z) \\ &= a\left(\left\{\sum_{i=1}^n (X_i * W_i)\right\} + b\right) \end{aligned}$$

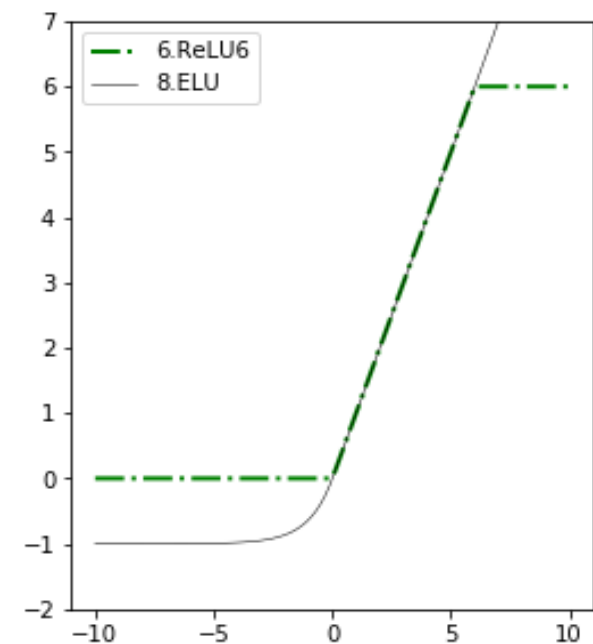
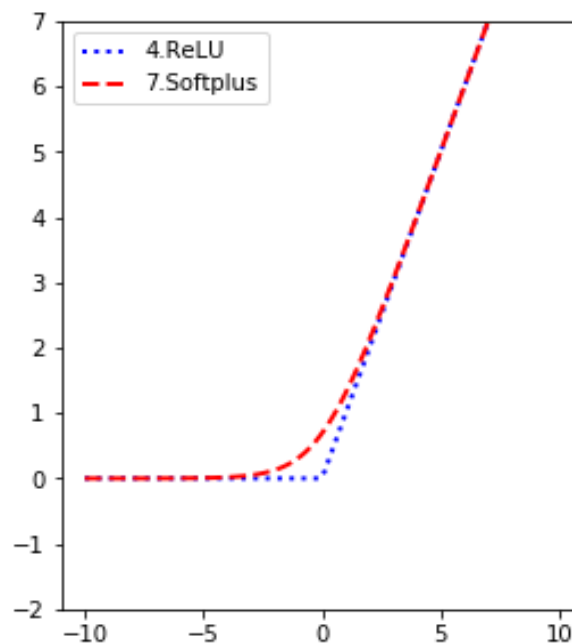
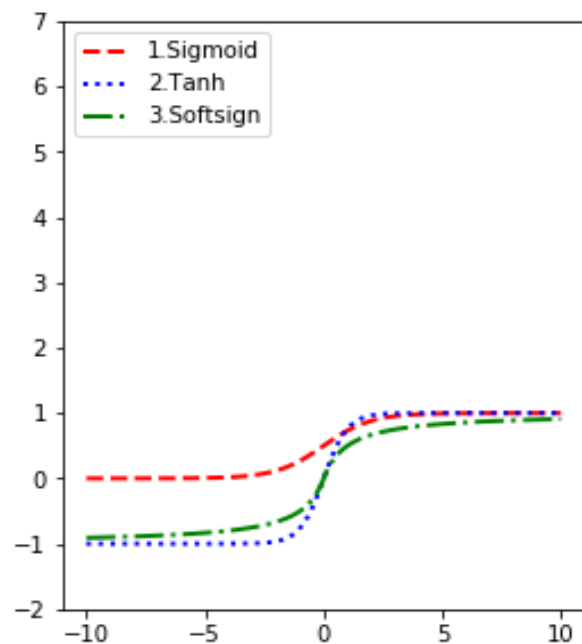
### (3.3) 様々な活性化関数

・「活性化関数(activation function)」には様々なものが考案されており、主なものを以下に示します：

No.	活性化関数名	和名	英語	式「 $Y = a(z)$ 」	値域	tensorflow の関数名	備考
1	sigmoid	シグモイド関数（ロジスティック関数）	sigmoid function (logistic function)	$1 / \{ 1 + \exp(-z) \}$	0 ~ 1	sigmoid	生物の神経細胞が持つ性質をモデル化したものとして用いられる。微分可能で解析的に扱いが容易だが、誤差逆伝播法の誤差項が、学習中に0に近づいていく傾向にある為（勾配消失問題 Vanishing Gradient problem）、余り使用されていない。（1986）
2	tanh	ハイパボリックタンジェント（双曲線正接関数）	hyperbolic tangent	$\{ \exp(z) - \exp(-z) \} / \{ \exp(z) + \exp(-z) \}$	-1 ~ +1	tanh	1990年代になり、活性化関数は原点を通すべきという考えから、標準シグモイド関数よりもそれを線形変換した tanh の方が良いと提案された。（1998）
3	softsign	ソフトサイン関数	softsign function	$z / \{ \text{abs}(z) + 1 \}$	-1 ~ +1	softsign	正弦関数に対する連続近似として位置づけられる。（2010 Xavier Glorot）
4	ReLU	正規化線形関数（ランプ関数）	Rectified Linear Unit	$\max(0, z)$	0 ~ $\infty$	relu	ReLU (Rectified Linear Units) は、入力値が負の値の場合は0、正の値の場合は入力値をそのまま出力します。最も一般的で基本的な方法です。（2011 Xavier Glorot）
5	Clipped RELU	Clipped RELU	Clipped Rectified Linear Unit	$\min(\max(0, z), \text{clipVal})$ clipVal : 0以上の浮動小数点数。正の部分の最大値。	0 ~ clipVal	(?)	Clipped RELU は、ReLUの特別なバージョンで、出力値が一定の大きさ以上にはならないように変更されたものです。ReLUでは、入力が正の値のときに線形な変換が適用されますが、Clipped RELU では入力に対して、出力は頭打ち状態（Clipping）になります。
6	ReLU6	正規化線形関数（ランプ関数）	Rectified Linear Unit 6	$\min(\max(0, z), 6)$	0 ~ +6	relu6	ReLU関数で上限を6としたもの。Clipped RELUの一つで、計算速度が速く、値の消失や発散の問題にも悩まされない。
7	softplus	ソフトプラス関数	softplus function	$\log(\exp(z) + 1)$	0 ~ $\infty$	softplus	ReLU関数を滑らかにしたもの。
8	ELU		Exponential Linear Unit	$\exp(z) - 1$ for $z < 0$ $z$ for $z \geq 0$	-1 ~ $\infty$	elu	ELUは、ReLUの特別なバージョンで、負の入力に対して指数関数（Exp）から1を引いた値を適用しています。

No.	活性化関数名	和名	英語	式「 $Y = a(z)$ 」	値域	tensorflow.nn の関数名	備考
9	LeakyReLU	LeakyReLU	Leaky Rectified Linear Unit	$\alpha * z \quad \text{for } z < 0$ $z \quad \text{for } z \geq 0$ <p><math>\alpha</math> : 0以上の浮動小数点数。負の部分の傾き。</p>	$-\infty \sim \infty$	leaky_relu	LeakyReLUは、ReLUの特別なバージョンで、ユニットがアクティブでないとき ( $z < 0$ ) でも微少な勾配を可能とします。これにより勾配消失を防ぎ、学習速度を向上させる効果が期待できます。 DCGAN (Deep Convolutional Generative Adversarial Network) と呼ばれるモデルでよく利用されます。
10	softmax	ソフトマックス関数	softmax function	$Y_i = \frac{\exp(X_i)}{\sum_{j=1}^k \exp(X_j)}$ <p>(<math>i=1 \sim k</math>)</p>	$0 \sim 1$	softmax	出力が $k$ 個ある場合に、 $k$ 個の入力値 $X_i$ ( $i=1 \sim k$ ) の順序関係を保ちながら、確率としての値 $Y_i$ ( $i=1 \sim k$ 、各値は0~1の範囲で、総和が1になる) に変換する関数。 $k$ 個のクラスへ分類する際の所属の確率を示すのによく用いられる。

・上記で紹介した「活性化関数」のうち、幾つかを図示します。



実装は (リスト 02-(03)-1) を参照。

(リスト 02-(03)-1)

```
#####  
# (リスト 02-(03)-1)  
# 様々な活性化関数  
#####  
import numpy as np  
import matplotlib.pyplot as plt  
%matplotlib inline  
  
#(1) シグモイド関数  
def f_sigmoid(x):  
    return 1 / ( 1 + np.exp(-x) )  
  
#(2) ハイパーボリックタンジェント (双曲線正接関数)  
def f_tanh(x):  
    return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))  
  
#(3) ソフトサイン関数  
def f_softsign(x):  
    return x / (np.abs(x) + 1)  
  
#(4) 正規化線形関数 (ランプ関数)  
def f_ReLU(x):  
    xcnt = len(x)  
    ylist = np.zeros(xcnt)  
    for ii in range(xcnt):  
        if x[ii] < 0:  
            ylist[ii] = 0  
        else:  
            ylist[ii] = x[ii]  
    return ylist  
  
#(6) 正規化線形関数 (ランプ関数、上限=6)  
def f_ReLU6(x):  
    xcnt = len(x)  
    ylist = np.zeros(xcnt)  
    for ii in range(xcnt):  
        if x[ii] < 0:  
            ylist[ii] = 0  
        elif x[ii] > 6:  
            ylist[ii] = 6  
        else:  
            ylist[ii] = xlist[ii]  
    return ylist
```

```

#(7) ソフトプラス関数
def f_softplus(x):
    return np.log(np.exp(x) + 1)

#(8) Exponential Linear Unit
def f_ELU(x):
    xcnt = len(x)
    ylist = np.zeros(xcnt)
    for ii in range(xcnt):
        if x[ii] < 0:
            ylist[ii] = np.exp(x[ii]) - 1
        else:
            ylist[ii] = x[ii]
    return ylist

# x データ列
xlist = np.linspace(-10, 10, 100)

# y データ列
y_sigmoid = f_sigmoid(xlist)
y_tanh = f_tanh(xlist)
y_softsign = f_softsign(xlist)
y_ReLU = f_ReLU(xlist)
y_ReLU6 = f_ReLU6(xlist)
y_softplus = f_softplus(xlist)
y_ELU = f_ELU(xlist)

# グラフ表示
plt.figure(figsize=(14, 5))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

plt.subplot(1, 3, 1)
plt.plot(xlist, y_sigmoid, 'r--', label='1. Sigmoid', linewidth=2)
plt.plot(xlist, y_tanh, 'b:', label='2. Tanh', linewidth=2)
plt.plot(xlist, y_softsign, 'g-.', label='3. Softsign', linewidth=2)
plt.ylim([-2, 7])
plt.legend(loc='upper left')

plt.subplot(1, 3, 2)
plt.plot(xlist, y_ReLU, 'b:', label='4. ReLU', linewidth=2)
plt.plot(xlist, y_softplus, 'r--', label='7. Softplus', linewidth=2)
plt.ylim([-2, 7])
plt.legend(loc='upper left')

```

```
plt.subplot(1, 3, 3)
plt.plot(xlist, y_ReLU6, 'g-', label='6. ReLU6', linewidth=2)
plt.plot(xlist, y_ELU, 'k-', label='8. ELU', linewidth=0.5)
plt.ylim([-2, 7])
plt.legend(loc='upper left')

plt.show()
```

#### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/活性化関数>

⇒ <http://yagami12.hatenablog.com/entry/2017/09/17/111935>

keras ライブラリ ⇒ <https://keras.io/ja/activations/>

keras ライブラリ ⇒ <https://keras.io/ja/layers/advanced-activations/>

ReLU6 ⇒ [https://www.tensorflow.org/api\\_docs/python/tf/nn/relu6](https://www.tensorflow.org/api_docs/python/tf/nn/relu6)

ReLU6 ⇒ <http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf>

LeakyReLU ⇒ [https://web.stanford.edu/~awni/papers/relu\\_hybrid\\_icml2013\\_final.pdf](https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf)

LeakyReLU ⇒ [https://www.tensorflow.org/api\\_docs/python/tf/nn/leaky\\_relu](https://www.tensorflow.org/api_docs/python/tf/nn/leaky_relu)

ELU ⇒ <https://arxiv.org/pdf/1511.07289v1.pdf>
















## (4) 様々なニューラルネットワーク

- ・「ユニット（形式ニューロン）」を複数つなげて「数理的な神経回路網」を構成したものが「ニューラルネットワーク（Neural Network, NN）」で、基本的に以下の層から構成されます：

入力層	ニューラルネットワークへの情報を取り込む層
隠れ層	入力側からの信号に対して、重みづけ評価を行いながら、出力側へ信号伝達する層（中間層ともいいます）
出力層	ニューラルネットワークが算出した結果を出力する層

- ・サイト「The Neural Network Zoo - The Asimov Institute (2016/09)」に、様々なニューラルネットワークについてチャートとしてまとめたものがあるので、それを中心に他のサイトも参考にしながら紹介します。  
（凡例）

記号	英名	日本名	所属層
 Backfed Input Cell	Backfed Input Cell	入力層のセル	入力層
 Input Cell	Input Cell	入力層のセル	入力層
 Noisy Input Cell	Noisy Input Cell	入力層のセル（ノイズを含む入力）	入力層
 Hidden Cell	Hidden Cell	隠れ層のセル	隠れ層
 Probablistic Hidden Cell	Probablistic Hidden Cell	隠れ層のセル（値が確率分布に従うようにしたもの）	隠れ層
 Spiking Hidden Cell	Spiking Hidden Cell	隠れ層のセル	隠れ層
 Output Cell	Output Cell	出力層のセル	出力層
 Match Input Output Cell	Match Input Output Cell	出力層のセル（入力値または、その復元値を出力としたもの）	出力層
 Recurrent Cell	Recurrent Cell	隠れ層のセル（回帰結合ニューラルネットワーク用）	隠れ層
 Memory Cell	Memory Cell	メモリセル（以前の時間における値を記憶できるセル）	隠れ層
 Different Memory Cell	Different Memory Cell	メモリセル（以前の時間における値を記憶できる別のタイプのセル）	隠れ層
 Kernel	Kernel	畳み込みにおけるカーネル（フィルタ）	隠れ層
 Convolution or Pool	Convolution or Pool	畳み込み層あるいはプーリング層のセル	隠れ層

【出典・参考】

- ⇒ <http://www.asimovinstitute.org/neural-network-zoo/>
- ⇒ <https://www.asimovinstitute.org/author/fjodorvanveen/>
- ⇒ <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- ⇒ <https://postd.cc/neural-network-zoo/>
- ⇒ <https://postd.cc/neural-network-zoo-latter/>

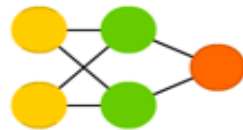
*A mostly complete chart of*  
**Neural Networks**

©2016 Fjodor van Veen - asimovinstitute.org

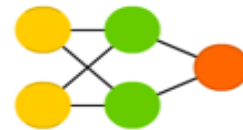
Perceptron (P)



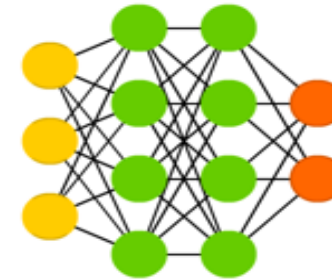
Feed Forward (FF)



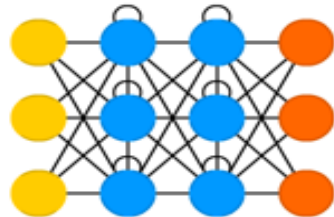
Radial Basis Network (RBF)



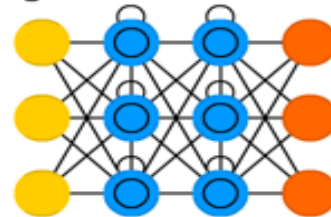
Deep Feed Forward (DFF)



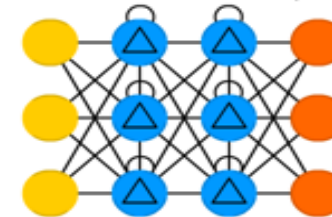
Recurrent Neural Network (RNN)



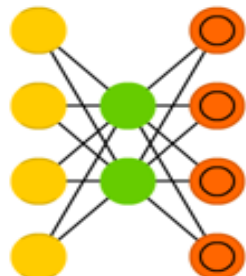
Long / Short Term Memory (LSTM)



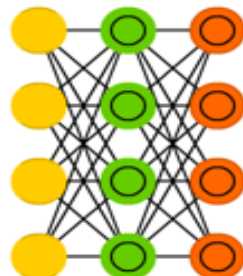
Gated Recurrent Unit (GRU)



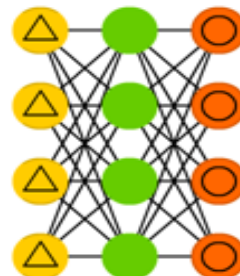
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)

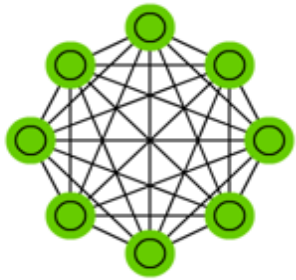


Sparse AE (SAE)

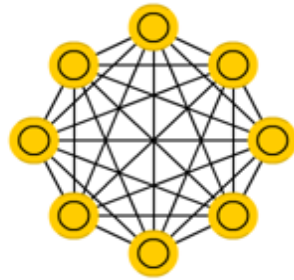


・ニューラルネットワークモデルの一覧（その2）

Markov Chain (MC)



Hopfield Network (HN)



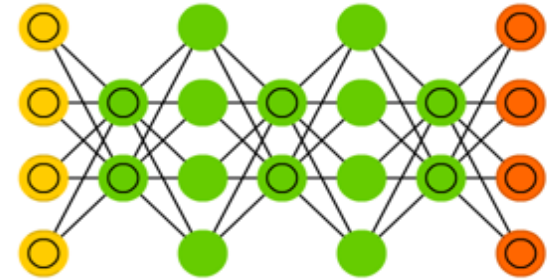
Boltzmann Machine (BM)



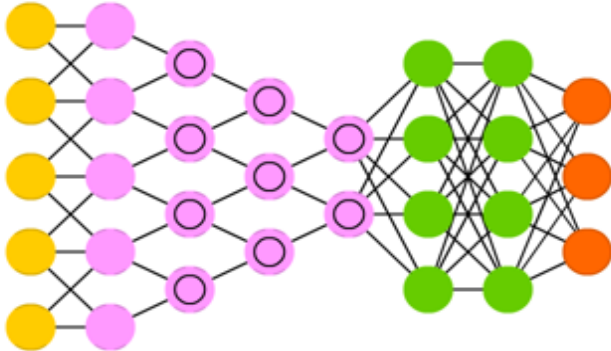
Restricted BM (RBM)



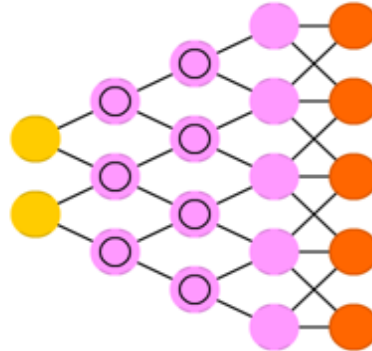
Deep Belief Network (DBN)



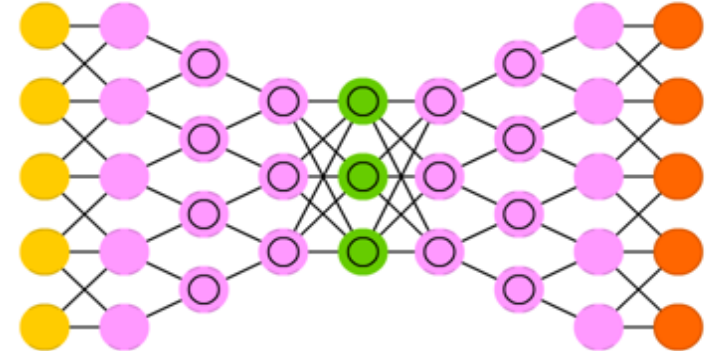
Deep Convolutional Network (DCN)



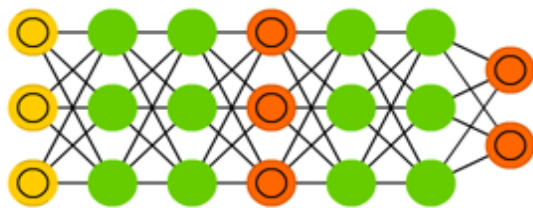
Deconvolutional Network (DN)



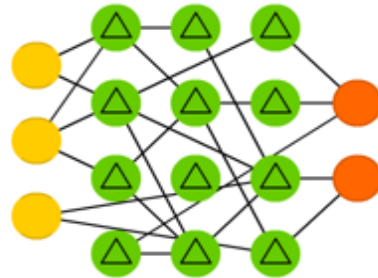
Deep Convolutional Inverse Graphics Network (DCIGN)



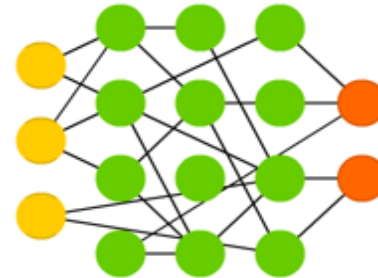
Generative Adversarial Network (GAN)



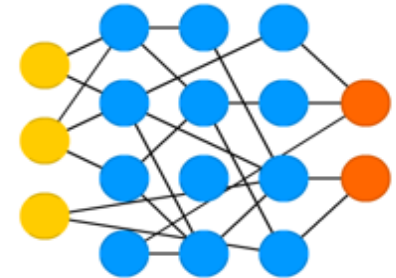
Liquid State Machine (LSM)



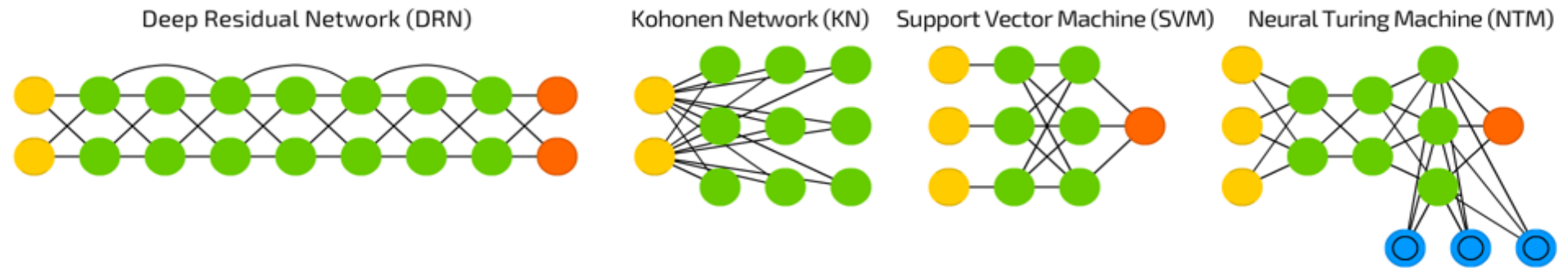
Extreme Learning Machine (ELM)



Echo State Network (ESN)

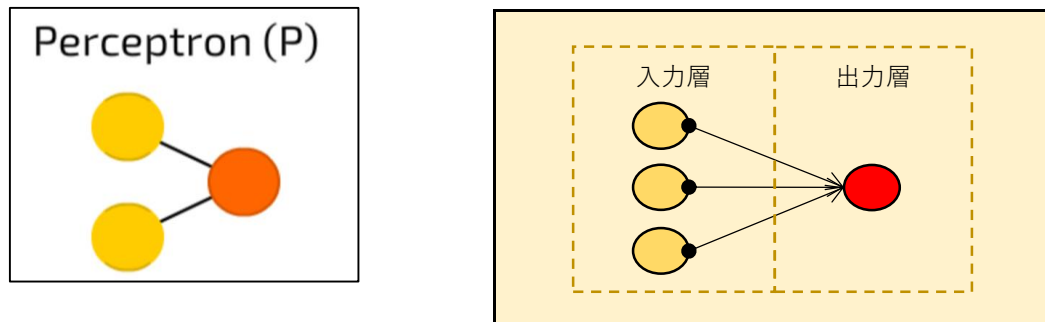


- ・ニューラルネットワークモデルの一覧（その3）



- ・以下に個々のニューラルネットワークについて、訳出・追記して紹介します。

#### (4.1) パーセプトロン (Perceptron、P)



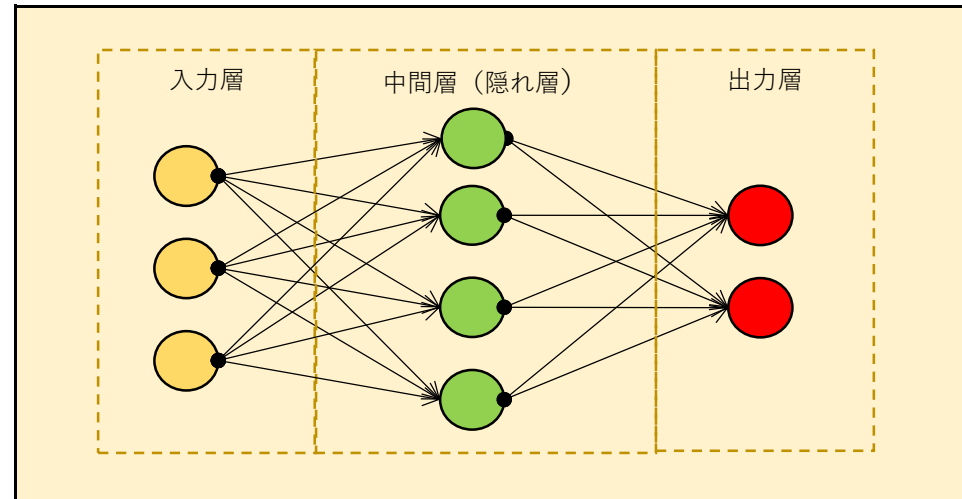
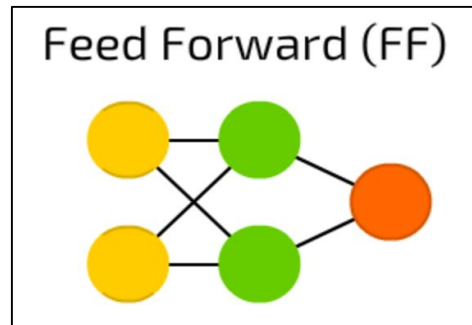
- ・ パーセプトロン (Perceptron、 P)は、視覚と脳の機能をモデル化したものであり、パターン認識を行います。  
いくつかの入力を受け取り、それらを合計し、活性化関数を適用して出力レイヤに渡す、  
というシンプルなネットワークでありながら学習能力を持ちます。
- ・ これは、心理学者・ 計算機科学者のフランク・ ローゼンブラットが1957年に考案し、1958年に論文を発表したものです。  
1960年代に爆発的なニューラルネットブーム（第一次ニューラルネットブーム）を巻き起こしましたが、  
1969年に、入力層と出力層のみの2層からなる、単純パーセプトロン（Simple perceptron）は線形非分離な問題を解けないことが  
人工知能学者マービン・ ミンスキーとシーモア・ パパートによって指摘されたことによって下火となりました。  
（「XOR問題」など）
- ・ 1986年に、デビッド・ ラメルハートとジェームズ・ マクレランドはパーセプトロンを多層にし、  
バックプロパゲーション（誤差逆伝播学習法）で学習させることで、線型分離不可能な問題が解けるように、  
単純パーセプトロンの限界を克服したことなどによって再び注目を集めました（第二次ニューラルネットブーム）。
- ・ パーセプトロンは、現在でも広く使われている機械学習アルゴリズムの基礎となっています。

#### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/パーセプトロン>

⇒ [http://hokuts.com/2015/11/25/ml2\\_perceptron/](http://hokuts.com/2015/11/25/ml2_perceptron/)

#### (4.2) 順伝播型ニューラルネットワーク (Feed forward neural networks、FF, FFNN)



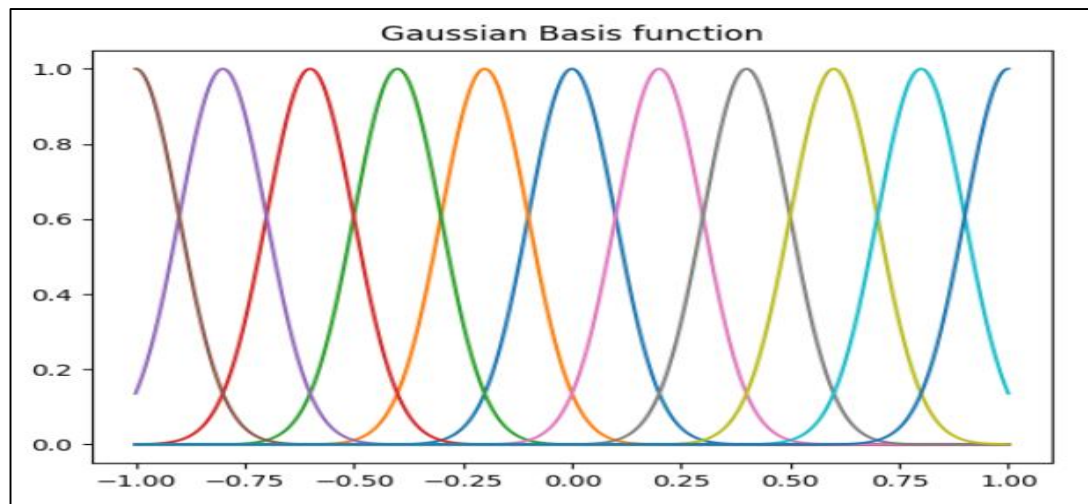
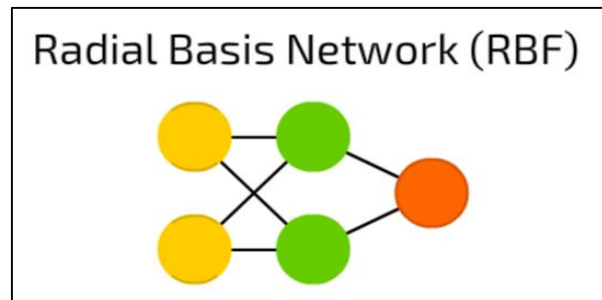
- ・ 順伝播型ニューラルネットワーク (Feed forward neural networks、FF) もかなり古く、1950年代に始まります。
- ・ 一般的に次の規則に従います：
  1. すべてのノードが完全に接続されている
  2. 入力層から出力層に向けて一方通行で流れる
  3. 入力と出力の間に、1つの隠れ層がある
- ・ ほとんどの場合、このタイプのネットワークは「誤差逆伝播法 (ごさぎゃくでんぱほう、Backpropagation)」を使用して学習します。

#### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/ニューラルネットワーク#順伝播型ニューラルネットワーク>



#### (4.3) ラジアル基底関数ネットワーク (Radial Basis Network、RBN)



- ・ラジアル基底関数ネットワーク (Radial Basis Network、RBN) は、順伝播型ニューラルネットワーク (FF) であり、活性化関数としてロジスティック関数の代わりにラジアル基底関数を使用します。
- ・ロジスティック関数では、任意の値を  $[0 \sim 1]$  の範囲にマップし、「はい、または、いいえ」という質問に答えます。ロジスティック関数は、分類や意思決定システムには適していますが、連続値には向いていません。逆に、ラジアル基底関数は「ターゲットからどれくらい離れているか」という判定に向いています。
- ・各々適当な点に関して球対称となる実数値関数からなる基底を考えると (基底関数の合成としてデータ分布が表現できるとする)、各基底関数はラジアル基底関数 (英: radial basis function、RBF、放射基底関数、動径基底関数) と呼ばれます。RBFにはさまざまなものがありますが、その中でも最もよく用いられるのが、ガウス基底関数 (Gaussian Basis function、上右図は1次元の例) です。

##### 【出典・参考】

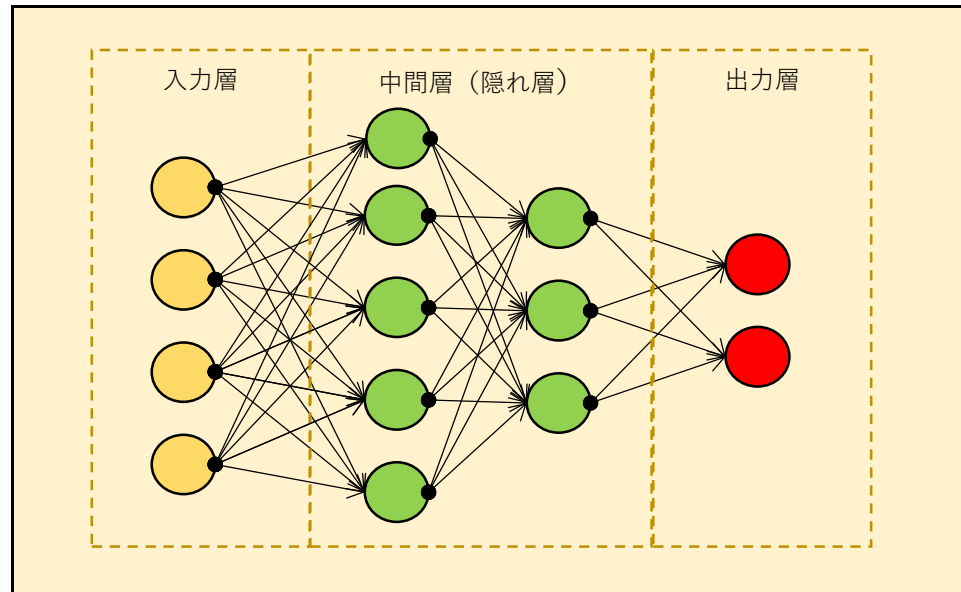
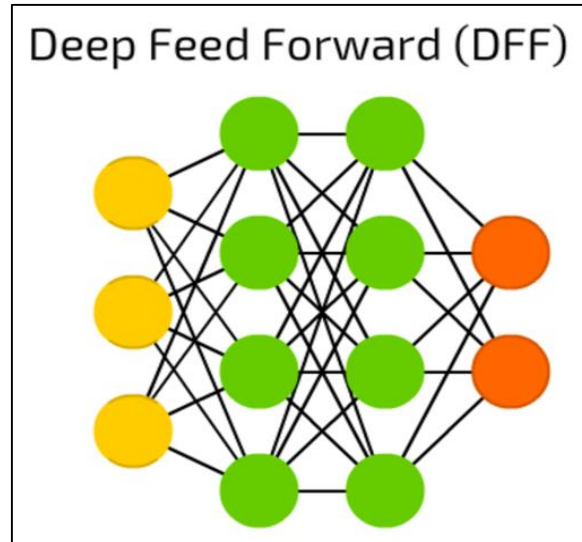
⇒ <http://www.brain.kyutech.ac.jp/~furukawa/data/rbf.html>

⇒ <http://linearml.hatenablog.com/entry/2017/11/01/041955>

⇒ <https://ja.wikipedia.org/wiki/放射基底関数>

(教材1) 「Pythonで動かして学ぶ! あたらしい機械学習の教科書」(2018年01月 翔泳社 伊藤真著) 「5.4 線形基底関数モデル」

#### (4.4) 深層順伝播型ニューラルネットワーク (Deep Feed forward neural networks、DFF)



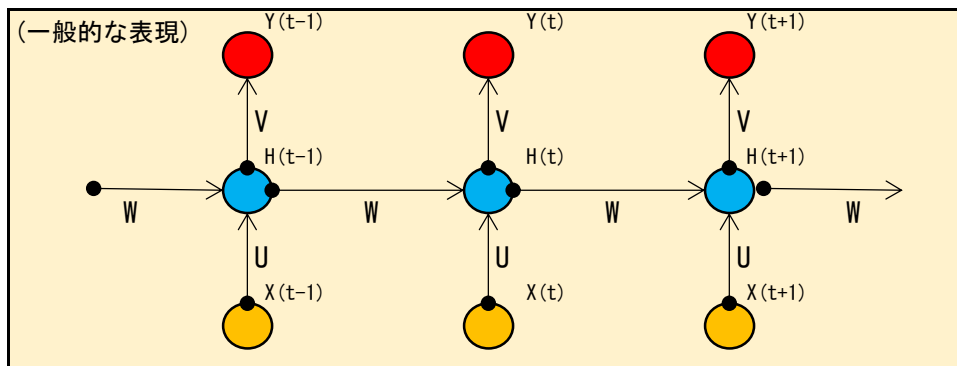
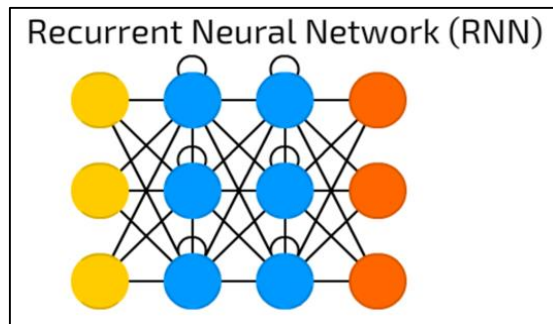
- ・ 深層順伝播型ニューラルネットワーク (Deep Feed forward neural networks、DFF)は、1990年代初めにディプラーニングのパンドラの箱を開きました。
- ・ これは単純な順伝播型ニューラルネットワーク (FF) の一つですが、複数の隠れ層があります。
- ・ かつてはコンピュータの性能の制約から、より多くのレイヤーを積み重ねることで、学習時間の指数関数的な増加につながり、DFFは非常に非実用的なものでしたが、コンピュータの性能の十分な現在では、FFと同じ目的の最新の機械学習システムの中核を形成しており、より良い結果が得られています。

【出典・参考】

⇒ <https://qiita.com/ma-oshita/items/99b2cf313494adbb964d>

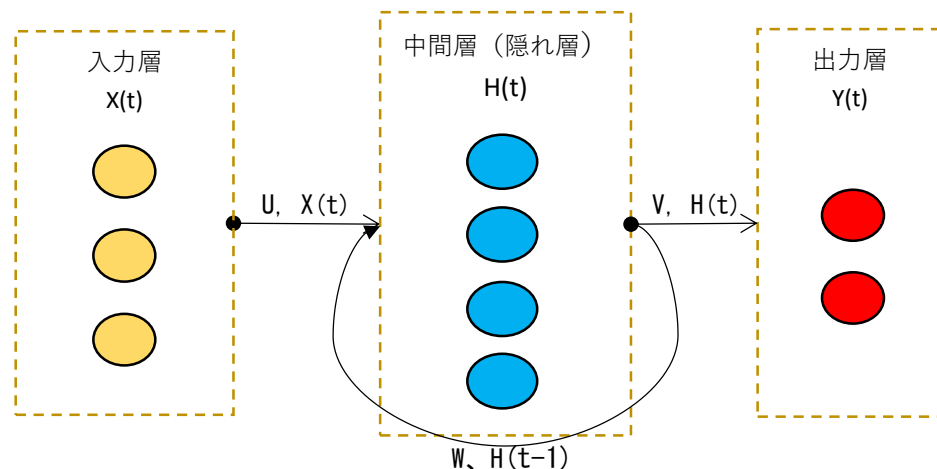


#### (4.5) リカレントニューラルネットワーク (Recurrent Neural Network、RNN)



- ・「回帰結合ニューラルネットワーク (Recurrent neural network、RNN)」は、ある時刻におけるユニットの出力が次の時刻における自身のユニットの入力になっているようなネットワークモデルで、時間的な依存関係を学習します。
- ・RNNは主に、過去の事象や決定が、現在の事象や決定に影響を与える可能性がある「文脈が重要な場合」に使用されます。その最も一般的な例はテキスト解析で、単語はそれが出現するより前の文脈の中で分析されます。

(別の表現)・・・時刻  $t$



$$H(t) = Ah( U * X(t) + W * H(t-1) + Bh )$$

$$Y(t) = Ay( V * H(t) + By )$$

$X(t)$  : 時刻  $t$  における入力層のユニット

$H(t)$  : 時刻  $t$  における隠れ層のユニット

$Y(t)$  : 時刻  $t$  における出力層のユニット

$U$  : 隠れ層  $H(t)$  に対する、入力層  $X(t)$  の重み係数

$V$  : 出力層  $Y(t)$  に対する、隠れ層  $H(t)$  の重み係数

$W$  : 隠れ層  $H(t)$  に対する、隠れ層  $H(t-1)$  の重み係数

$Bh$  : バイアス (隠れ層  $H(t)$  計算用)

$By$  : バイアス (出力層  $Y(t)$  計算用)

$Ah$  : 活性化関数 (隠れ層  $H(t)$  計算用)

$Ay$  : 活性化関数 (出力層  $Y(t)$  計算用、softmax関数など)

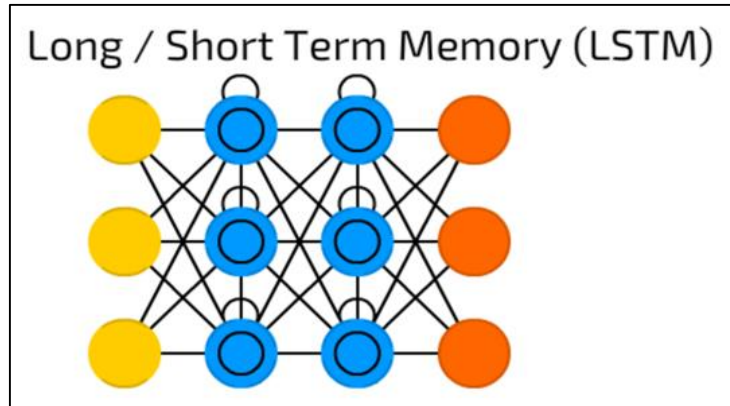
【出典・参考】

⇒ <https://qiita.com/kiminaka/items/87afd4a433dc655d8cfd>

⇒ <https://qiita.com/KojiOhki/items/89cd7b69a8a6239d67ca>

⇒ [https://www.researchgate.net/publication/277411157\\_Deep\\_Learning](https://www.researchgate.net/publication/277411157_Deep_Learning)

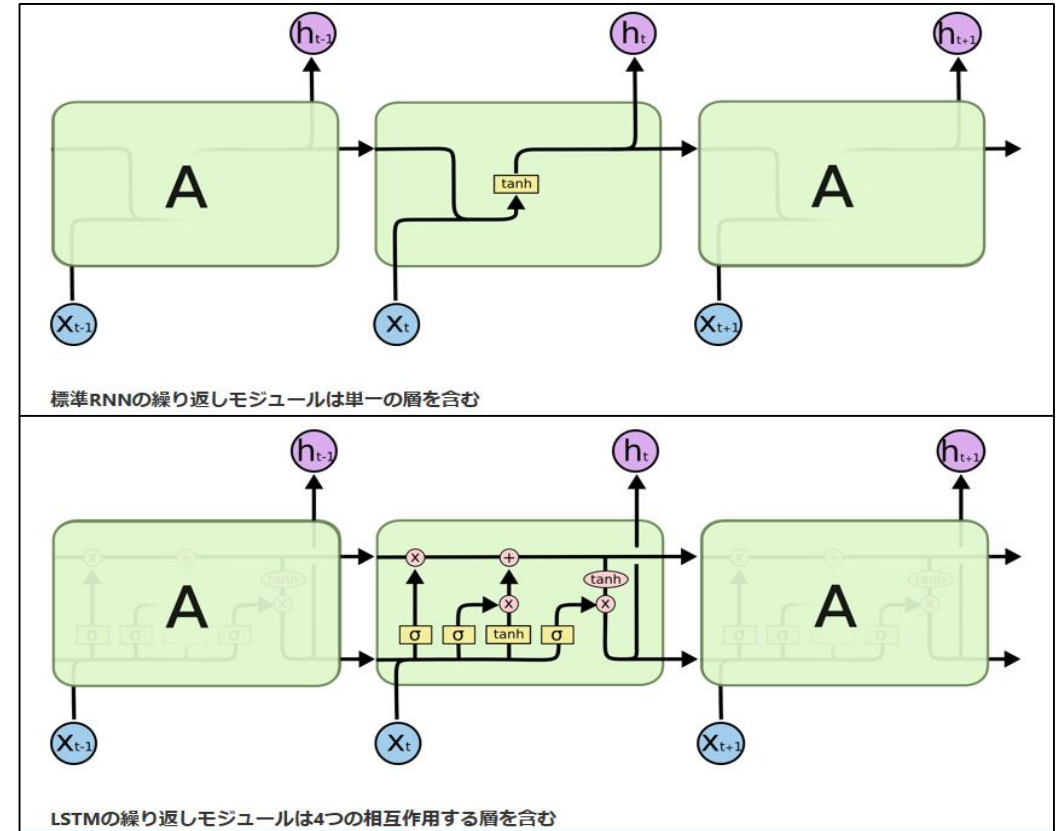
#### (4.6) Long/Short Term Memory (LSTM)



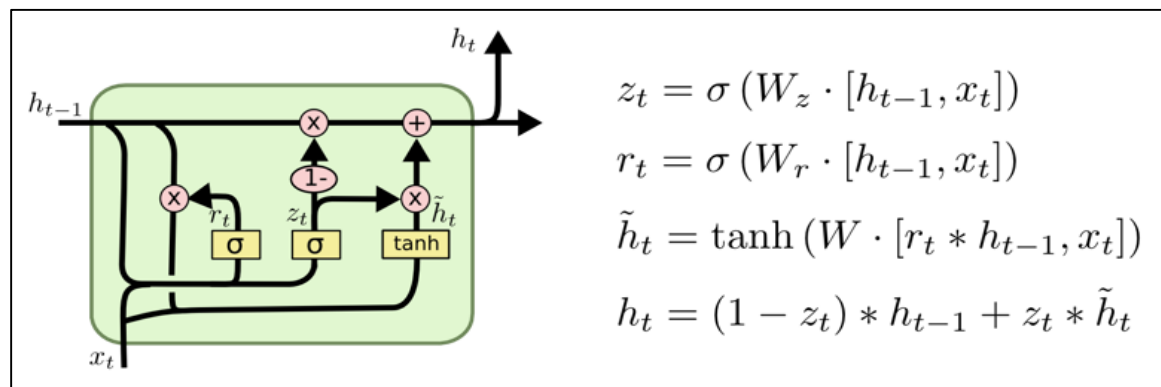
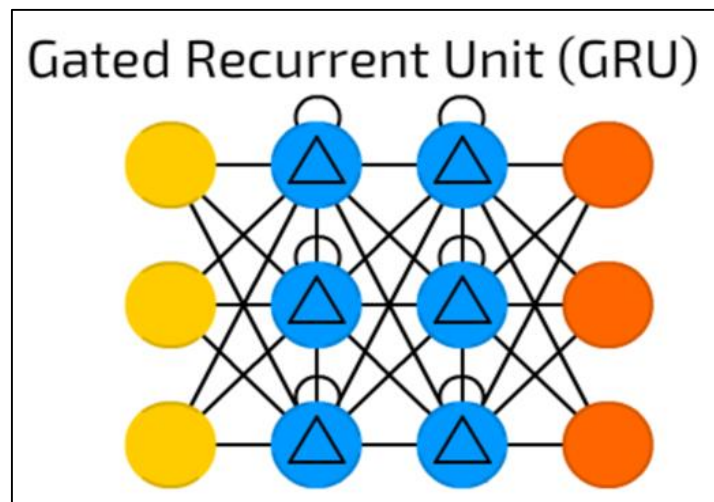
- ・ LSTM(Long short-term memory)は、RNN(Recurrent Neural Network)の拡張として1995年に登場した、時系列データ (sequential data) に対するモデル、あるいは構造 (architecture) の1種です。
- ・ LSTM(Long short-term memory)という名は、Long term memory (長期記憶) と Short term memory (短期記憶) という神経科学における用語から取られています。
- ・ LSTMはRNNの中間層のユニットをLSTM blockと呼ばれるメモリと3つのゲートを持つブロックに置き換えることで実現されています。
- ・ LSTMのその最も大きな特長は、従来のRNNでは学習できなかった長期依存 (long-term dependencies) が学習可能である点にあります。

#### 【出典・参考】

- ⇒ [https://qiita.com/t\\_Signull/items/21b82be280b46f467d1b](https://qiita.com/t_Signull/items/21b82be280b46f467d1b)
- ⇒ <https://qiita.com/KojiOhki/items/89cd7b69a8a6239d67ca>



## (4.7) Gated Recurrent Unit (GRU)

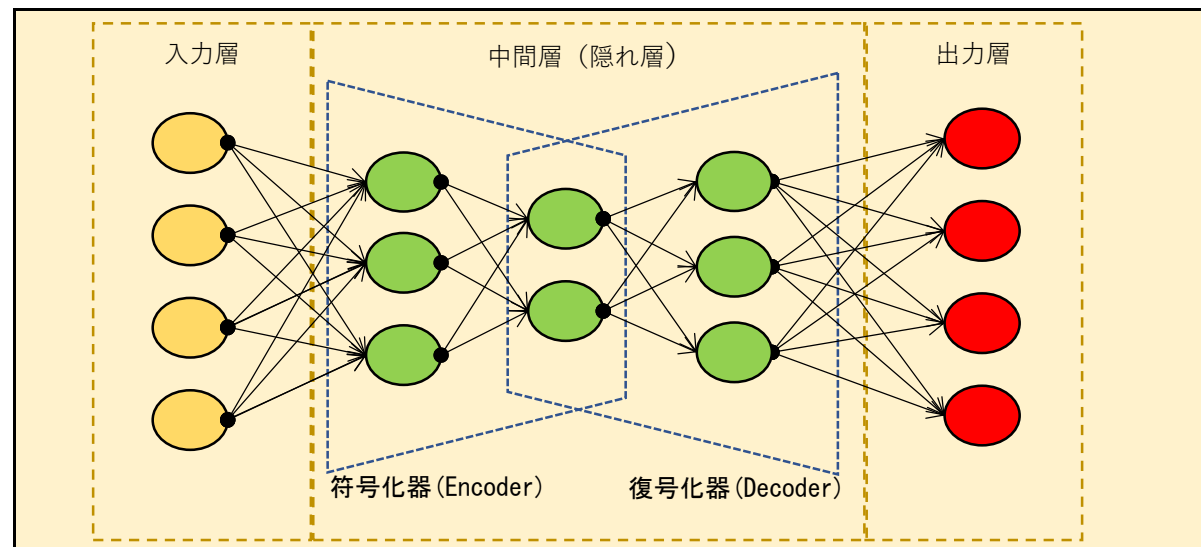
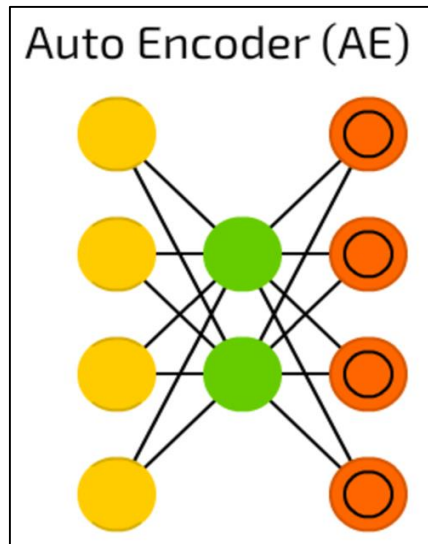


- ・「Gated Recurrent Unit (GRU)」は、LSTMの変形の一つで  
「Cho、 et al. (2014 「<https://qiita.com/Koji0hki/items/89cd7b69a8a6239d67ca>」)」により導入されたものです。
- ・これは忘却ゲートと入力ゲートを単一の「更新ゲート」に組み合わせます。  
また、セル状態と隠れ状態をマージし、他のいくつかの変更を加えます。
- ・現在はサウンド（音楽）や音声合成で最も使用されています。

### 【出典・参考】

- ⇒ <https://qiita.com/Koji0hki/items/89cd7b69a8a6239d67ca>
- ⇒ <https://arxiv.org/pdf/1406.1078v3.pdf>
- ⇒ <https://arxiv.org/pdf/1508.03790v2.pdf>
- ⇒ <http://proceedings.mlr.press/v37/jozefowicz15.pdf>

#### (4.8) Autoencoder (AE)



- ・ オートエンコーダー(自己符号化器 Autoencoder)とは、出力データが入力データに近づくように学習を行う、ニューラルネットワークモデルです。
- ・ 具体的には、入力データを中間層に圧縮しようとする符号化器(Encoder)と中間層を入力データに復元しようとする復号化器(Decoder)を直接つないだ「砂時計型ニューラルネットワーク (hourglass-type neural network)」として構成されるシンプルな構造になっています。
- ・ 中間層の次元の大きさを、入力層よりも小さい次元にすることで、入力データを再現することができるような、低次元の特徴を持った中間層を得ることができます。特徴を捉えたまま次元を削減することで、データを圧縮することができるため、従来の次元削減と同じような効果を得ることができます。

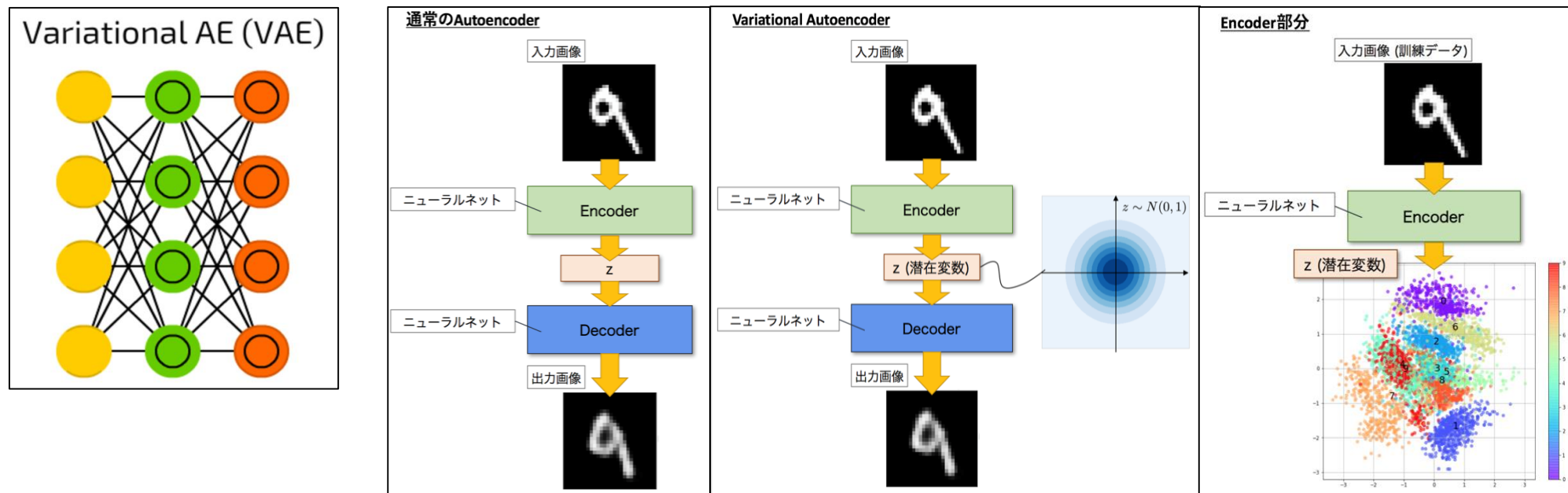
#### 【出典・参考】

⇒ <https://blog.keras.io/building-autoencoders-in-keras.html>

「深層学習 Deep Learning」 近代科学社 人工知能学会監修 2015年11月

(教材2)「現場で使える！TensorFlow開発入門 Kerasによる深層学習モデル構築手法」 (2018年04月 翔泳社 太田満久 他著)

## (4.9) Variational Autoencoder (VAE)



- ・ Variational Autoencoder (VAE) は、AE の符号化器 (Encoder) の出力結果  $z$  (潜在変数) が、標準正規分布  $N(0, 1)$  をなすようにしたものです。
- ・ VAEでは、符号化器 (Encoder) の出力に揺らぎを持たせることにより、ランダムな入力に対してもきれいな画像を出力できることを目指していますが、「出力がぼやけてしまう」といった問題を抱えています。

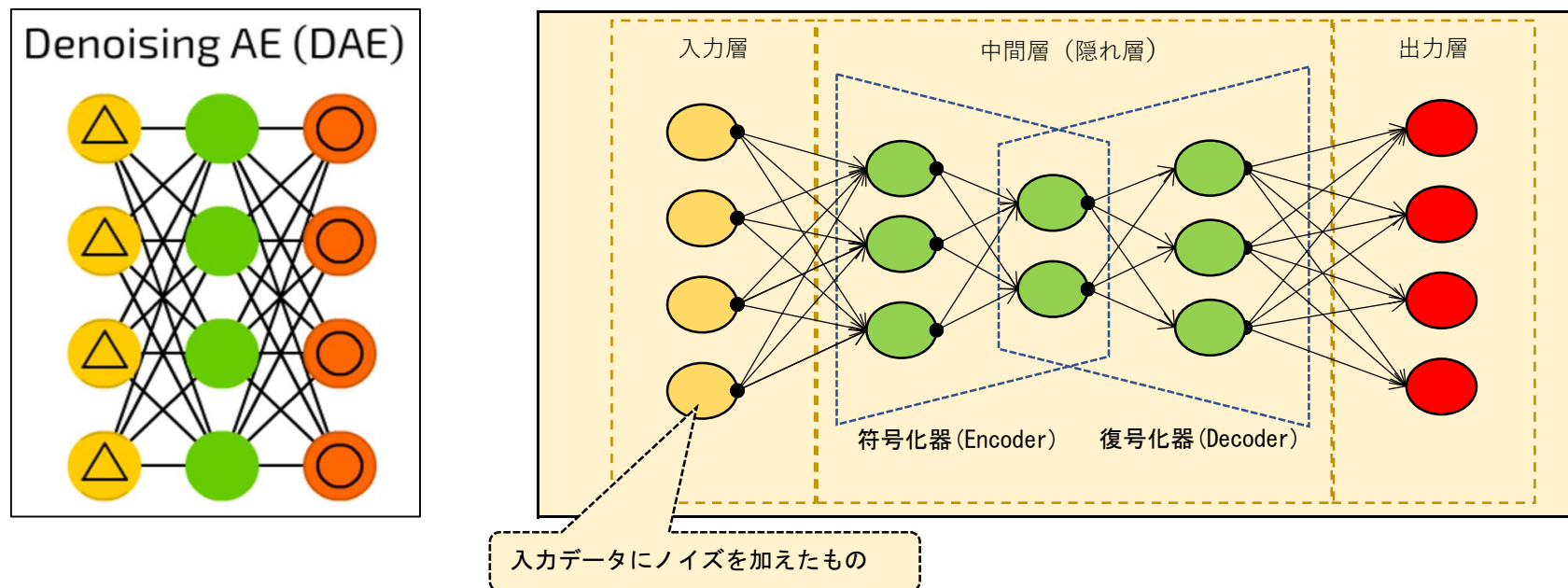
### 【出典・参考】

⇒ <https://qiita.com/kenmatsu4/items/b029d697e9995d93aa24>

⇒ <https://blog.keras.io/building-autoencoders-in-keras.html>

(教材2)「現場で使える！TensorFlow開発入門 Kerasによる深層学習モデル構築手法」 (2018年04月 翔泳社 太田満久 他著)

#### (4.10) Denoising Autoencoder (DAE)



- ・ Denoising Autoencoder (デノイジングオートエンコーダー、DAE) は、AEで入力データにノイズ加えたものをモデルの入力とするものです。
- ・ DAEはノイズ混じりの入力データを入力して、ノイズを除去しながらオリジナルデータに近づけるように学習を行っていくため、Autoencoderより頑健に再構成ができると言われています。

#### 【出典・参考】

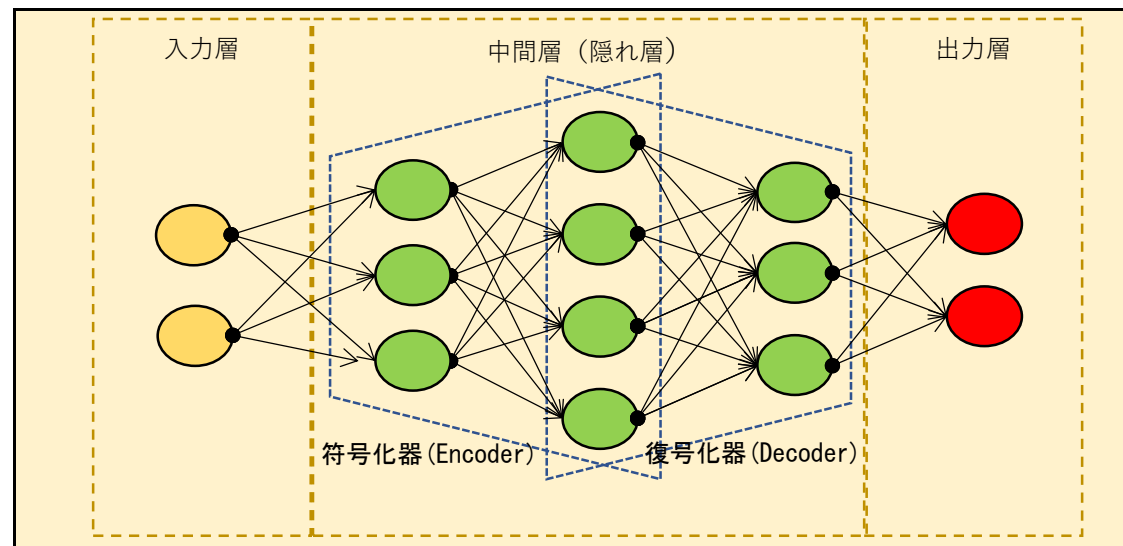
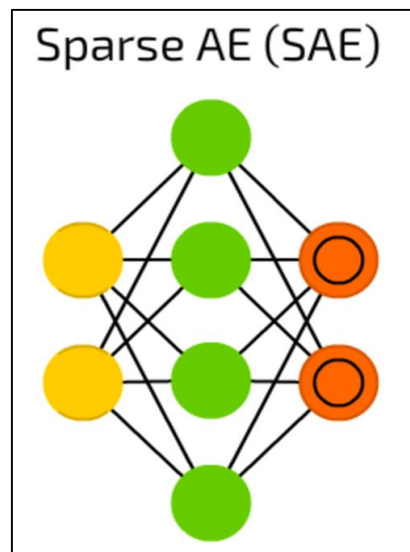
⇒ <https://elix-tech.github.io/ja/2016/07/17/autoencoder.html>

⇒ <http://www.cs.toronto.edu/~larochepublications/icml-2008-denoising-autoencoders.pdf>

(教材2)「現場で使える！TensorFlow開発入門 Kerasによる深層学習モデル構築手法」 (2018年04月 翔泳社 太田満久 他著)



## (4.11) Sparse Autoencoder (SAE)

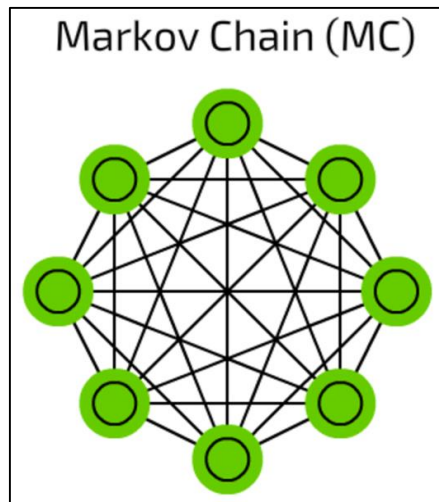


- ・ Sparse Autoencoder (SAE)の構造は AE と同じですが、中間層のセル数は入出力層セル数よりも大きくなっています。
- ・ SAEでは、中間層で活性化するニューロンがより少なくなり疎(sparse)になります。  
より少ないニューロンの発火で再現できる方がエネルギー効率の面でも良いとみられています。
- ・ 「脳では少数のニューロンが反応することで複雑な表現をする機能があるのではないか」という仮説があり、この機構をスパースコーディング (Sparse Coding)と呼んでいます。このスパースコーディングが「汎化性能と効率の向上の鍵ではないか」と言われており、機械学習では重要なテーマになっています。

### 【出典・参考】

- ⇒ <https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf>
- ⇒ <https://elix-tech.github.io/ja/2016/07/17/autoencoder.html#sparse>
- ⇒ <https://blog.keras.io/building-autoencoders-in-keras.html>
- 「あたらしい人工知能の教科書」P.219 (2017年8月 翔泳社 多田智史著)

## (4.12) マルコフ連鎖 (Markov chain, MC)



- ・「マルコフ連鎖 (マルコフレんさ、Markov chain, MC)」とは、マルコフ過程のうち、とりうる状態が離散的 (有限または可算) なもの (離散状態マルコフ過程) をいいます。また特に、時間が離散的なもの (時刻は添え字で表される) を指すことが多いです。マルコフ連鎖は、未来の挙動が現在の値だけで決定され、過去の挙動と無関係です。
- ・「未来の挙動が現在の値だけで決定され、過去の挙動と無関係であるという性質」を「マルコフ性 (マルコフ性、Markov property)」と言います。各時刻において起こる状態変化 (遷移または推移) に関して、マルコフ連鎖は遷移確率が過去の状態によらず、現在の状態のみによる系列です。特に重要な確率過程として、様々な分野に応用されます。
- ・「マルコフ過程 (マルコフかてい)」とは、マルコフ性をもつ確率過程のことをいいます。このような過程は例えば、確率的にしか記述できない物理現象の時間発展の様子に見られます。
- ・「確率過程 (かくりつかてい、stochastic process。不規則過程 (random process) とも言う)」とは、時間とともに変化する確率変数のことです。株価や為替の変動、ブラウン運動などの粒子のランダムな運動を数学的に記述するモデルとして利用しています。
- ・「確率変数 (かくりつへんすう、random variable, aleatory variable, stochastic variable)」とは、確率論ならびに統計学において、ランダムな実験により得られ得る全ての結果を指す変数です。数学で言う変数は関数により一義的に決まるのに対し、確率変数は確率に従って定義域内の様々な値を取ることができます。
- ・上図のモデルでは、各ユニット  $i$  の次時刻 ( $t+1$ ) の状態値  $X_i(t+1)$  が、現時刻 ( $t$ ) の他ユニット  $j$  の状態値  $X_j(t)$  に依存することを示しています。このモデルは、確率 (ベイズフィルタなど) に基づく分類、クラスタリング (何らかの種類のもの)、有限状態機械として使用することができます。強化学習では、環境を表す数理モデルになります。

### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/マルコフ連鎖>

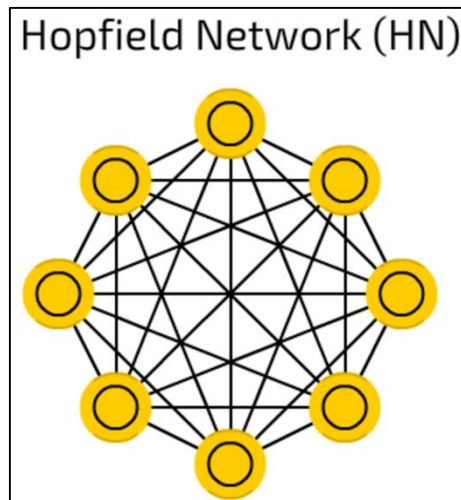
⇒ <https://ja.wikipedia.org/wiki/マルコフ過程>

⇒ <https://ja.wikipedia.org/wiki/確率過程>

⇒ <https://ja.wikipedia.org/wiki/確率変数>



#### (4.13) ホップフィールド・ネットワーク (Hopfield network、HN)



- ・ホップフィールド・ネットワーク (Hopfield network, HN) は、ジョン・ホップフィールド (J. J. Hopfield) が1982年に提唱したニューラルネットワークモデルです。
- ・HNは、各ユニット間に対称的な相互作用がある非同期型ネットワークで、信号は双方向にやり取りします。「非同期型ネットワーク」というのは、或る時刻に一つのユニットだけが他のユニットからの入力刺激の総和に基づき状態更新を行なうものです。
- ・HNの各ユニット*i* (状態値  $X_i$ ) は、発火状態 (=1) か、非発火状態 (=0) かの何れかの状態を取ります。HNのユニット*i* (状態値  $X_i$ ) について、ある時刻*t* での他ユニットから受け取る信号  $\lambda_i$  は次式のとおり：

$$\lambda_i(t) = \left\{ \sum_{j \neq i} W_{ij} * X_j(t) \right\} + b_i$$

$\lambda_i(t)$  : 時刻*t* において、着目ユニット*i* が他ユニットから受け取る信号

$b_i$  : 着目ユニット*i* の他ユニットからの信号取り込み時のバイアス

$X_j(t)$  : 時刻*t* における、他ユニット*j* の状態値 ( $X_j$ )

$W_{ij}$  : 他ユニット*j* の状態値 ( $X_j$ ) の、着目ユニット*i* への信号の重み  
( $W_{ij} = W_{ji}$  ・ ・ 対称的、 $W_{ii}=0$  ・ ・ 自己結合なし)

$\lambda_i(t)$  の値により、ユニット*i*の状態値  $X_i(t)$  が次のように更新されます：

$\lambda_i(t) > 0$  の時、 $X_i(t) = 1$

$\lambda_i(t) = 0$  の時、 $X_i(t)$  は更新無し

$\lambda_i(t) < 0$  の時、 $X_i(t) = 0$

- ・ここでは紹介しませんが、HNはネットワークのエネルギーという概念を導入しており、巡回セールス問題などの最短経路探索問題や、入力画像のノイズ除去などの解法として使用されています。

##### 【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/ホップフィールド・ネットワーク>

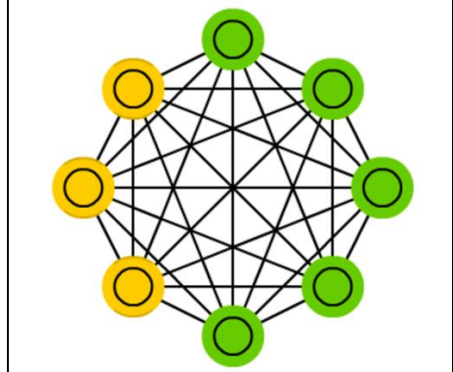
⇒ <http://www.sist.ac.jp/~kanakubo/research/neuro/hopfieldnetwork.html>

⇒ [http://www.scholarpedia.org/article/Hopfield\\_network](http://www.scholarpedia.org/article/Hopfield_network)

「深層学習 Deep Learning」P. 39 近代科学社 人工知能学会監修 2015年11月

#### (4.14) ボルツマンマシン (Boltzmann Machine, BM)

Boltzmann Machine (BM)



- ・ボルツマンマシン (Boltzmann Machine, BM) は、1985年にジェフリー・ヒントン (Geoffrey Hinton) とテリー・セジュノスキー (Terry Sejnowski) によって開発された、HN の一種です。
- ・BM では、全てが隠れユニットで構成される HN と異なり、幾つかのユニットが入力ユニットになります。隠れユニットの更新は、HN同様に非同期に行われます。隠れユニットの更新が済んだ時点で、入力ユニットは出力ユニットになります。
- ・最急降下法による誤差逆伝播法や HN が持つ学習時の局所解への捕捉の問題に対応する為に、BM では、ネットワークの動作に温度の概念を取り入れ、最初は激しく徐々に穏やかに逐次的に最適解に近づくように工夫しています (擬似焼きなまし法 (Simulated annealing) といいます)。

- ・BM では、HN 同様にユニットの値は発火状態 (=1) か、非発火状態 (=0) かの何れかの状態を取りますが、擬似焼きなまし法 (Simulated annealing) では、ユニット値が1となる確率Pをシグモイド関数 $\sigma$ を少し変形し、温度Tを入れた式で表現します：

$$P(z) = 1 / \{ 1 + \exp(-z / T) \}$$

そして、非同期に一つのユニットが状態変化する毎に、温度Tを下げていくことで、解の候補値の変化の振幅を徐々に狭め、逐次的に最適解に近づくように工夫しています。

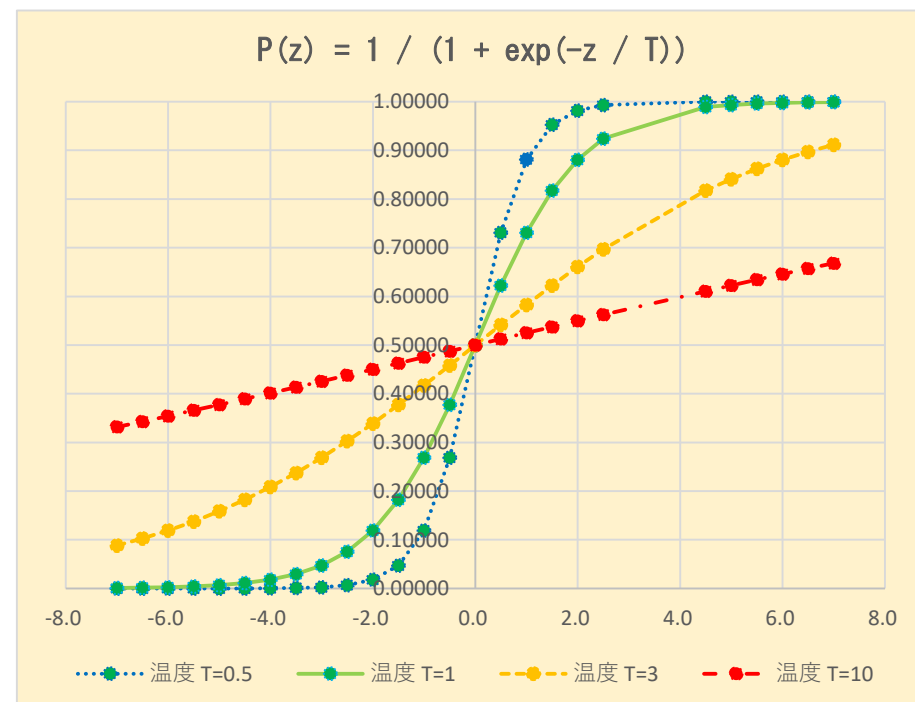
- ・右図は、確率  $P(z)$  が温度  $T$  に応じてどう変化するかを示す図です。温度  $T$  が高い程 1となる確率が低く、最適解に納まり難いことを示します。温度  $T$  の初期値と減少程度はモデルの設計に依存します。

#### 【出典・参考】

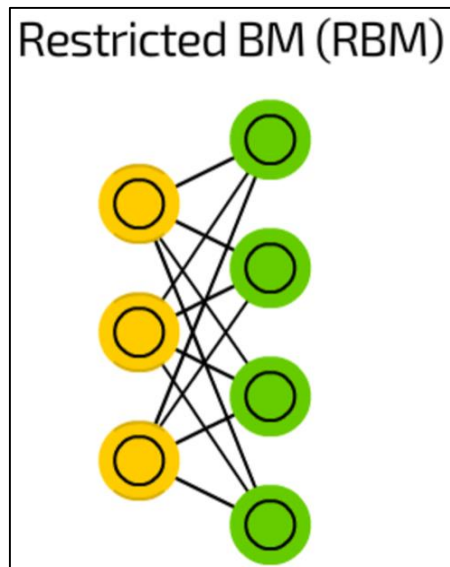
⇒ <https://ja.wikipedia.org/wiki/ボルツマンマシン>

⇒ <http://www.sist.ac.jp/~kanakubo/research/neuro/boltzmannmachine.html>

「深層学習 Deep Learning」P.39 近代科学社 人工知能学会監修 2015年11月



#### (4.15) 制限ボルツマンマシン (Restricted Boltzmann Machine、RBM)

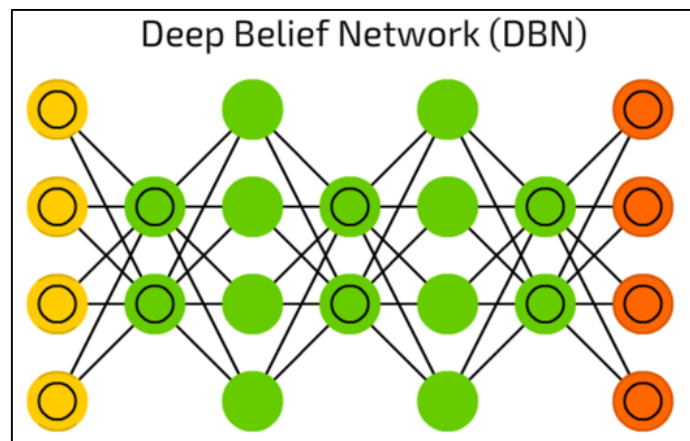


- ・制限ボルツマンマシン (Restricted Boltzmann Machine, RBM) は、BM の一種であり、可視層 (Visible Layer) と隠れ層 (Hidden Layer) の 2 層構造を持ちます。  
RBM は、可視層のユニットと隠れ層のユニット間でのみ接続しているという制限が付いたモデルです。  
(同一層のユニット同士、つまり、可視層のユニット同士、または隠れ層のユニット同士は接続無し)
- ・RBMは、1986年に Paul Smolensky によって Harmonium という名前で最初に発明され、Geoffrey Hinton と共同研究者が、2000年半ばに、RBMの高速学習アルゴリズムを発明したことで、見直されるようになりました。
- ・一般的な BM の学習では、ユニット数の指数関数的な計算コストがかかるため非実用的ですが、RBM ではコントラストティブ・ダイバージェンス (Contrastive Divergence) 法(※1)で、効率化が図られています。(※1)ここでは紹介しません。
- ・RBM は、次元削減、分類、回帰、協調フィルタリング、特徴学習、トピックモデルなどに役立ちます。

##### 【出典・参考】

- ⇒ <https://ja.wikipedia.org/wiki/ボルツマンマシン>
  - ⇒ <https://postd.cc/a-beginners-guide-to-restricted-boltzmann-machines/>
  - ⇒ [https://en.wikipedia.org/wiki/Restricted\\_Boltzmann\\_machine](https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine)
  - ⇒ [https://qiita.com/t\\_Signull/items/f776aecb4909b7c5c116](https://qiita.com/t_Signull/items/f776aecb4909b7c5c116)
- 「深層学習 Deep Learning」P.57 近代科学社 人工知能学会監修 2015年11月

#### (4.16) 深層信念ネットワーク (Deep Belief Network、DBN)

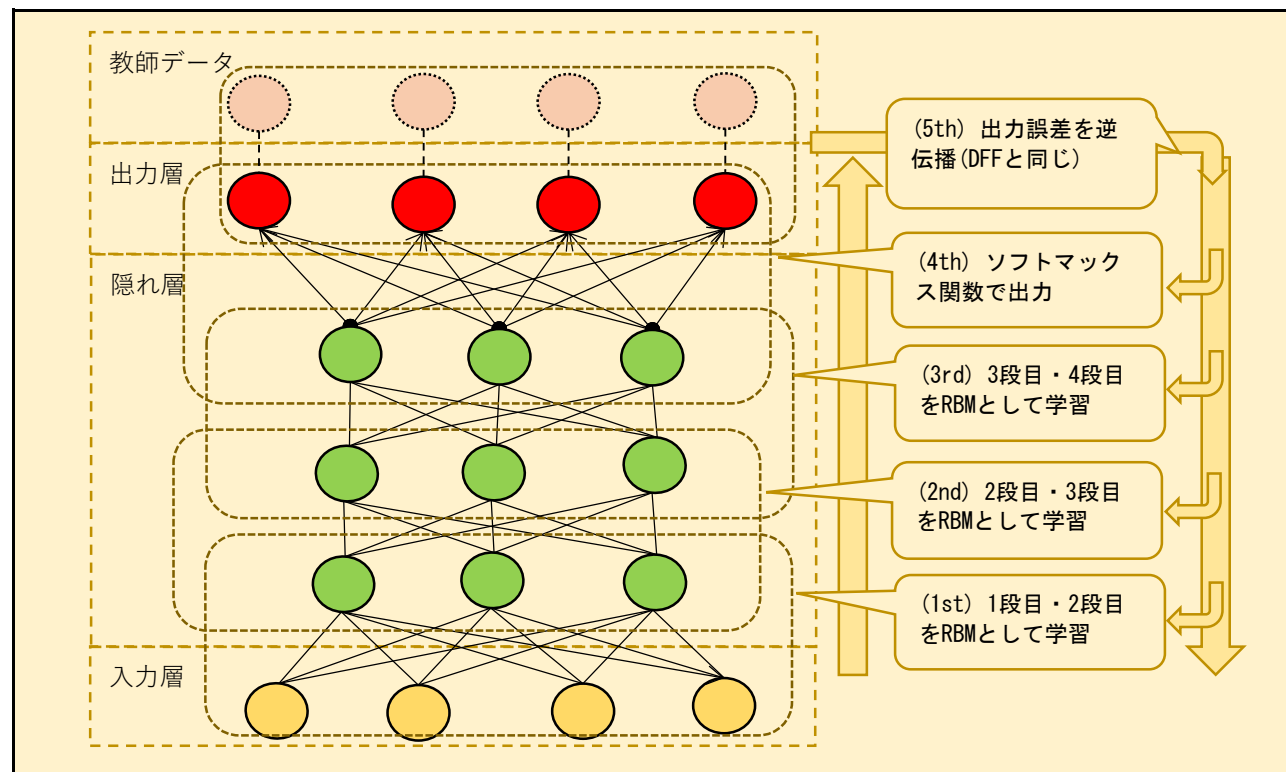


- ・ 深層信念ネットワーク

(Deep Belief Network, DBN) は、2006年に Geoffrey Hinton と共同研究者が、RBMを多数重ねて積み上げた多層のネットワークモデルとして考案したものです。

- ・ これが今日まで続くDeep Learningブームの火付け役となりました（第三次ニューラルネットブームの始まり）。

- ・ 学習に際しては、下の層から順にRBMを1つずつ学習させます（右上図）。最上位層としてソフトマックス層（ソフトマックス関数を適用する層）を追加することで、教師データとのすり合わせを行い、下層の全ネットワークに対して逆伝播を行うようにすることが出来ます。これにより教師あり学習として振る舞うことが可能になります。



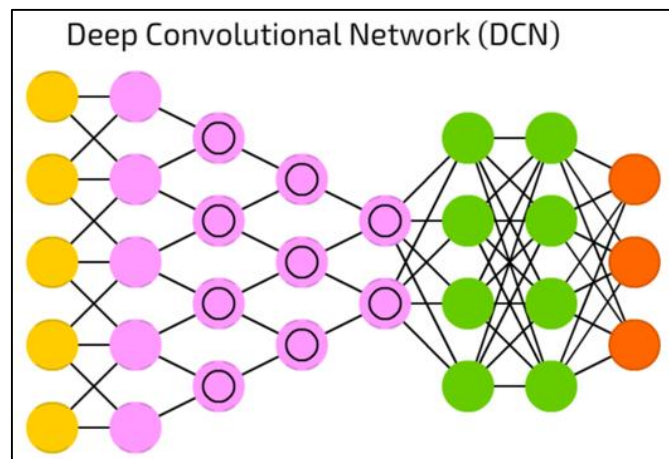
【出典・参考】

⇒ [https://qiita.com/t\\_Signull/items/f776aecb4909b7c5c116](https://qiita.com/t_Signull/items/f776aecb4909b7c5c116)

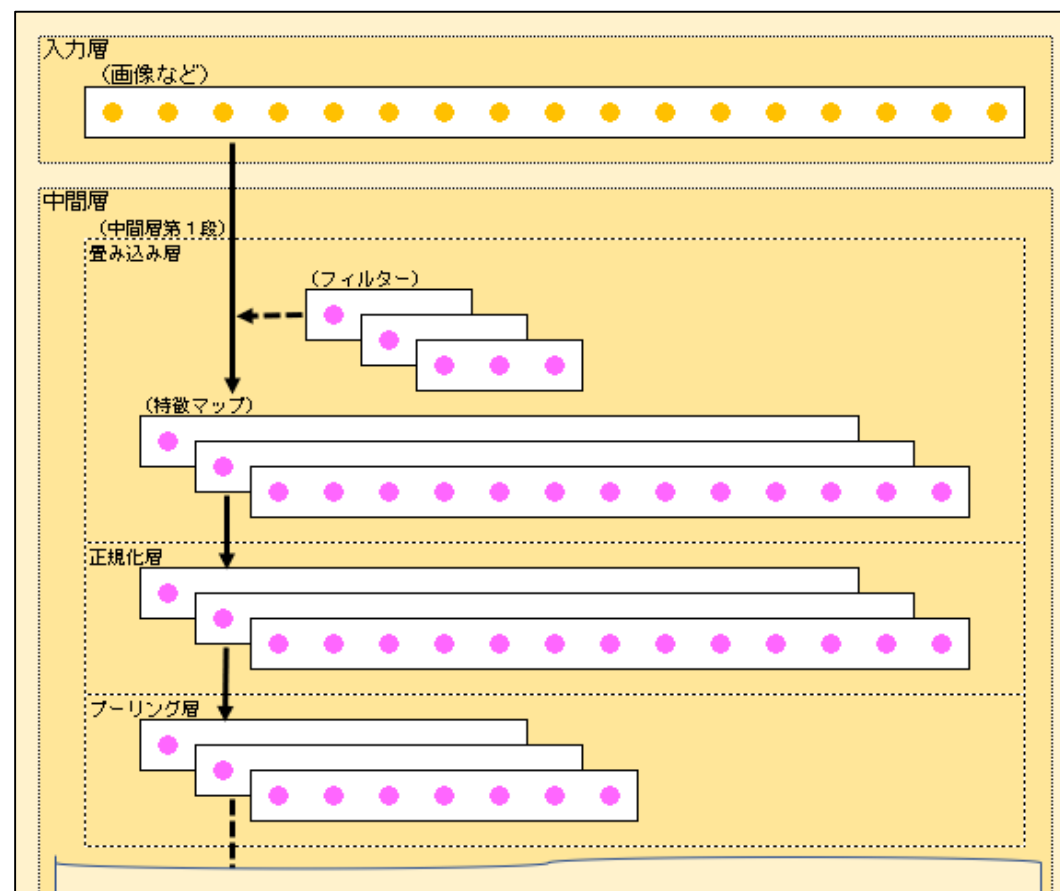
「深層学習 Deep Learning」P.72 近代科学社 人工知能学会監修 2015年11月

「あたらしい人工知能の教科書」P.217 (2017年8月 翔泳社 多田智史著)

#### (4.17) 深層畳み込みニューラルネットワーク (Deep Convolutional neural network、DCN)



- ・畳み込みニューラルネットワーク  
(たたみこみニューラルネットワーク、Convolutional neural network、CNN) は、階層型ニューラルネットワークに、畳み込み構造を入れたモデルです。
- ・深層畳み込みニューラルネットワーク  
(Deep Convolutional neural network、DCN) は、CNN の中間層（隠れ層）を深くしたものです。



(次ページへ)

- ・これは、人の視覚をモデルにした構成になっており、画像認識分野を中心に幅広く利用されているディープラーニング（深層学習）手法の一つです。
- ・このモデルでは、中間層（隠れ層）が、畳み込み層（コンボリューション層）、正規化層、プーリング層、全結合層で構成された複数の層からなります。正規化層とプーリング層は必須ではありません。畳み込み層でフィルタの移動量（ストライド、stride）を調整することにより情報縮約を行って、プーリング層を代行することもあります。

**畳み込み層 : Convolution Layer**

コンボリューション層ともいいます。

入力側からの信号に対してフィルタ処理を行い、特徴マップ (feature map) を作成する層です。

フィルタはカーネルともいいます。

**正規化層 : Normalization Layer**

各サンプルの平均と分散をそれぞれ0と1になるように、入力側からの信号を変換する層。

これで、入力データ間のばらつきを補正します。

**プーリング層 : Pooling Layer**

畳み込み層で作成された特徴マップ (feature map) に対し情報縮約操作を行う層。

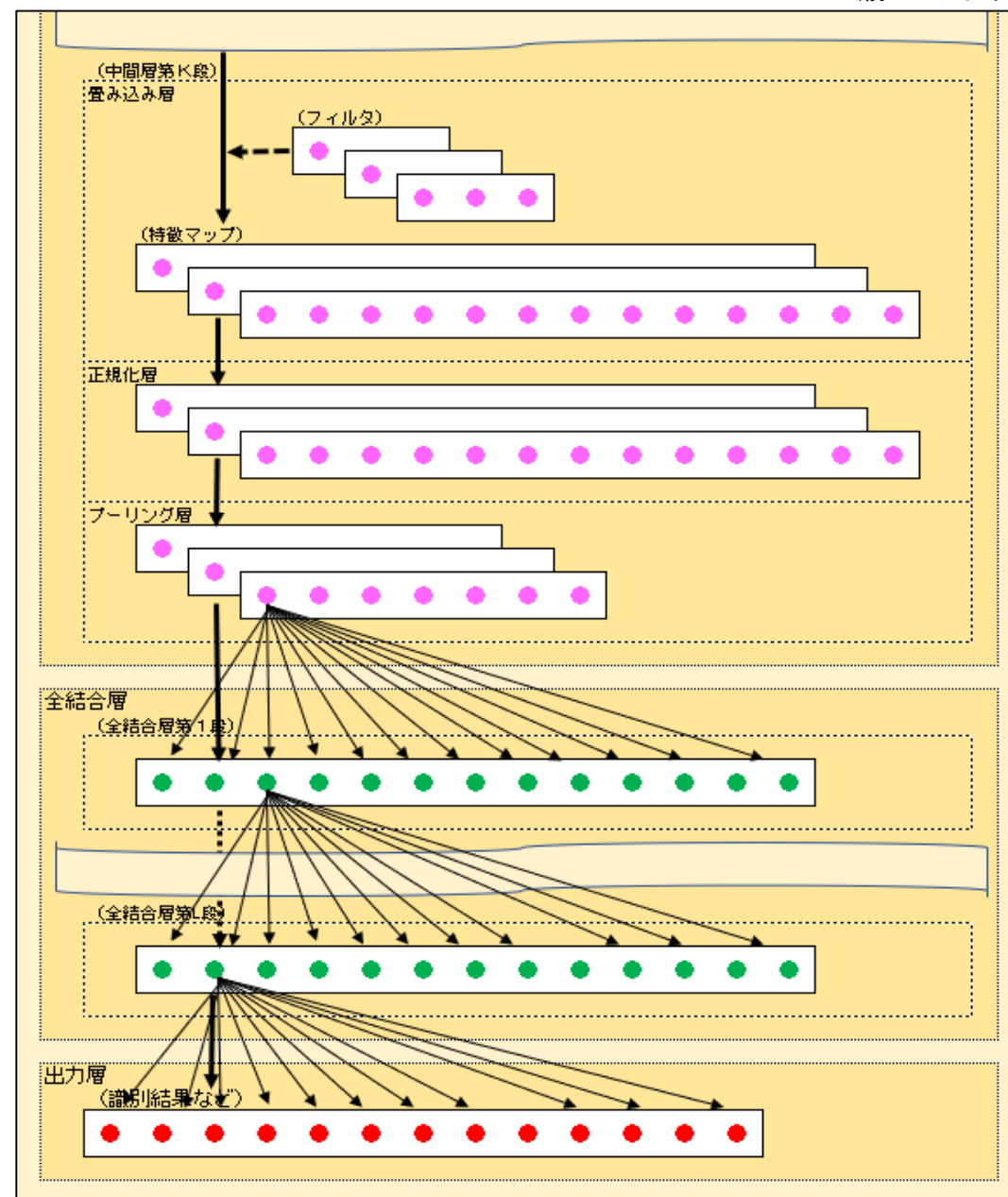
プーリングには、以下のようなものがあります：

- ・ 最大値プーリング  
(着目領域での最大値で縮約する)
- ・ 平均値プーリング  
(着目領域での平均値で縮約する)
- ・ Lpプーリング  
(着目領域での中央値を強調した値で縮約する)

**全結合層 : Fully Connected Layer**

異なる層の各ユニット同士が、全ての組合せで結合することを全結合と言います。

通常、出力層と前段の数層を全結合とし、これを全結合層としてモデル化します。

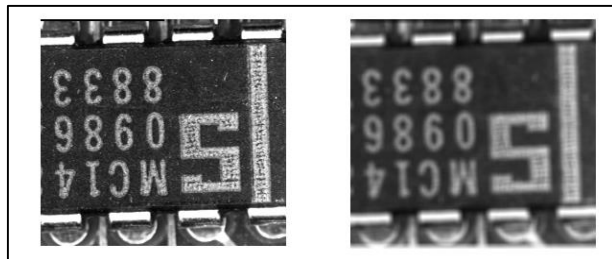




- ・畳み込みニューラルネットワークには、以下のような特徴があります：
  - ① 複雑なパターン認識問題にも、簡潔なネットワークで対応できます。
  - ② 全体としてユニット数が少なくて済むので、計算が容易です。
- ・画像処理では様々なフィルタがあり、これによって画像解析の為に前処理を行っています。  
CNN では、学習を通してこのフィルタを自動作成します。

以下は画像処理フィルタの例であり、「<https://imaging-solution.net/imaging/filter-algorithm/>」からの引用です。

移動平均フィルタ (※1)



(3 × 3)

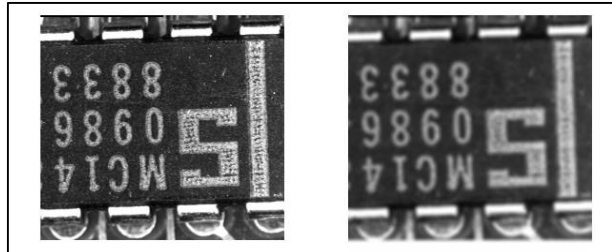
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

(5 × 5)

$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$

- ・移動平均フィルタ（平均化フィルタ、単に平滑化フィルタともいう）では、注目画素のその周辺の輝度値を用いて、輝度値の平均を求め、処理後画像の輝度値とするフィルタです。  
(※1 左が原画で右がフィルタ処理後の画像)

ガウシアンフィルタ (※1)



(3 × 3)

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

(5 × 5)

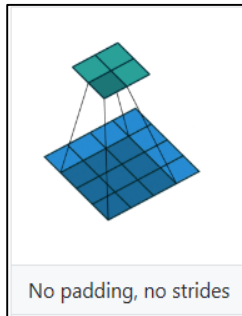
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

- ・ガウシアンフィルタは、注目画素に近いほど、平均値を計算するときの重みを大きくし、遠くなるほど重みを小さくなるようにガウス分布の関数を用いてレートを計算しているフィルタです。

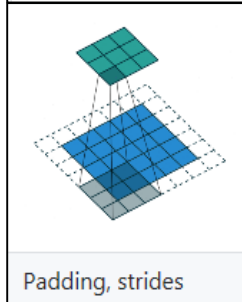
- ・ 以下に畳み込み時のフィルタ・ストライド・パディングのパターンを幾つか示します。

(「[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)」からの引用です)

例1



例2



- ・ (例1)は、4×4の画像(下図)に対して、フィルタのサイズを3×3、ストライドのサイズを1×1、パディング無しで畳み込みを行った結果、2×2の特徴マップが作成されたものです(上図)。
- ・ (例2)は、5×5の画像(下図)に対して、フィルタのサイズを3×3、ストライドのサイズを2×2、パディングあり(片側サイズ1)で畳み込みを行った結果、3×3の特徴マップが作成されたものです(上図)。
- ・ 「パディング (padding)」はフィルターを適用する際に、周囲を0などの固定した値で埋めておく方法です。これによりフィルター適用後画像 (特徴マップ) のサイズが小さくなるのを防ぐことができます。
- ・ 「ストライド (stride)」はフィルターを適用する際に、フィルターを移動させる間隔の値です。ストライドを大きくするとフィルター適用後画像 (特徴マップ) は小さくなります。

#### 【出典・参考】

- ⇒ [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)
- ⇒ [https://deeppage.net/deep\\_learning/2016/11/07/convolutional\\_neural\\_network.html](https://deeppage.net/deep_learning/2016/11/07/convolutional_neural_network.html)
- ⇒ [http://deeplearning.stanford.edu/wiki/index.php/Feature\\_extraction\\_using\\_convolution](http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution)
- ⇒ [http://www.nlab.ci.i.u-tokyo.ac.jp/pdf/CNN\\_survey.pdf](http://www.nlab.ci.i.u-tokyo.ac.jp/pdf/CNN_survey.pdf)
- ⇒ [https://www.nttcom.co.jp/research/keyword/dl/pdf/interactive\\_full.pdf](https://www.nttcom.co.jp/research/keyword/dl/pdf/interactive_full.pdf)
- ⇒ <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- ⇒ [https://deeppage.net/deep\\_learning/2016/10/26/batch\\_normalization.html](https://deeppage.net/deep_learning/2016/10/26/batch_normalization.html)



- ・ 畳み込み層の出力サイズ（特徴マップのサイズ）は、次式で求めることができます：

$$\text{Output} = (W - F + 2P) / S + 1$$

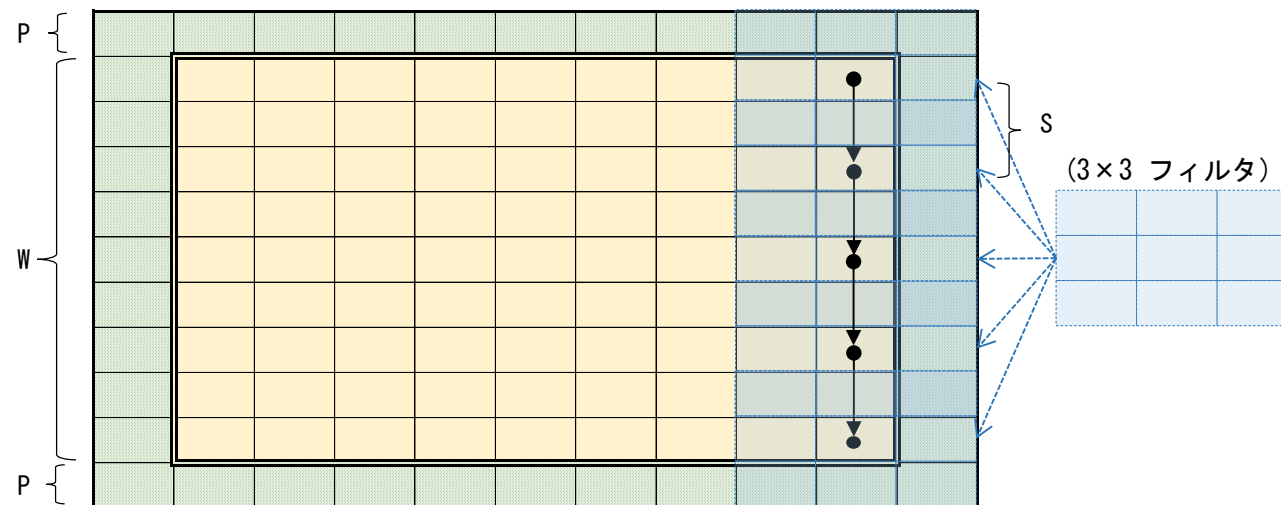
Output : 畳み込み層の出力サイズ（特徴マップのサイズ）

W : 畳み込み層の入力サイズ

F : フィルタサイズ

P : パディングサイズ

S : ストライドサイズ



上記例の縦方向では、

W : 9

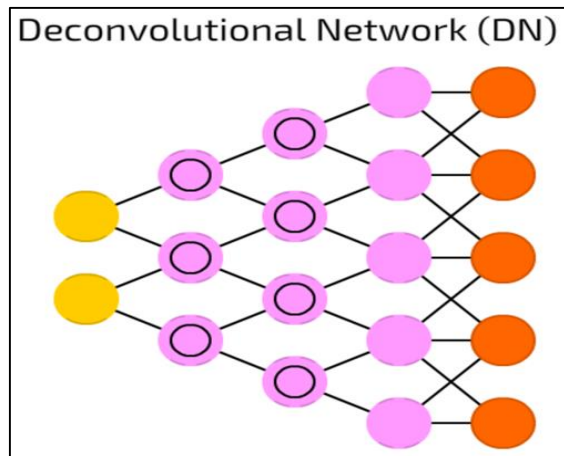
F : 3

P : 1

S : 2

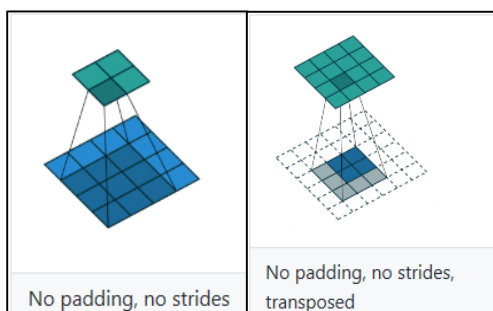
Output : 5 = ( 9 - 3 + 2\*1 ) / 2 + 1

## (4.18) 逆畳み込みネットワーク (Deconvolutional Network, DN)



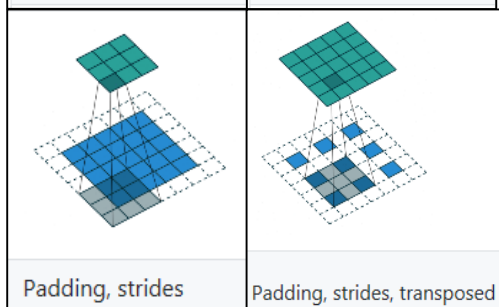
- ・ 逆畳み込みネットワーク (Deconvolutional Network, DN) は、DCNを逆にしたものです。つまり、画像のフィルタ結果と使用したフィルタおよびフィルタ適用時の移動量(ストライド)とパディングの指定から、元画像を復元しようとするものです。
- ・ 畳み込み処理時にフィルタ適用後の画像サイズが、元画像のサイズと同じになるように元画像サイズの周辺にフィルタのサイズに合わせてパディング(padding)という埋め合わせを行います。逆畳み込み時にはこれも考慮します。
- ・ 逆畳み込み(Deconvolution)は以前の呼び名で、転置畳み込み(Transposed Convolution)と言われることが多いです。

例1



- ・ (例1)の左側は、 $4 \times 4$ の画像(下図)に対して、フィルタのサイズを $3 \times 3$ 、ストライドのサイズを $1 \times 1$ 、パディング無しで畳み込みを行った結果、 $2 \times 2$ の特徴マップが作成されたものです(上図)。
- ・ (例1)の右側は、これで作成された $2 \times 2$ の特徴マップ(下図)に対して、フィルタのサイズ/ストライドのサイズ/パディングの有無を考慮して逆畳み込みを行い、 $4 \times 4$ の画像(上図)を復元をしたものです。

例2



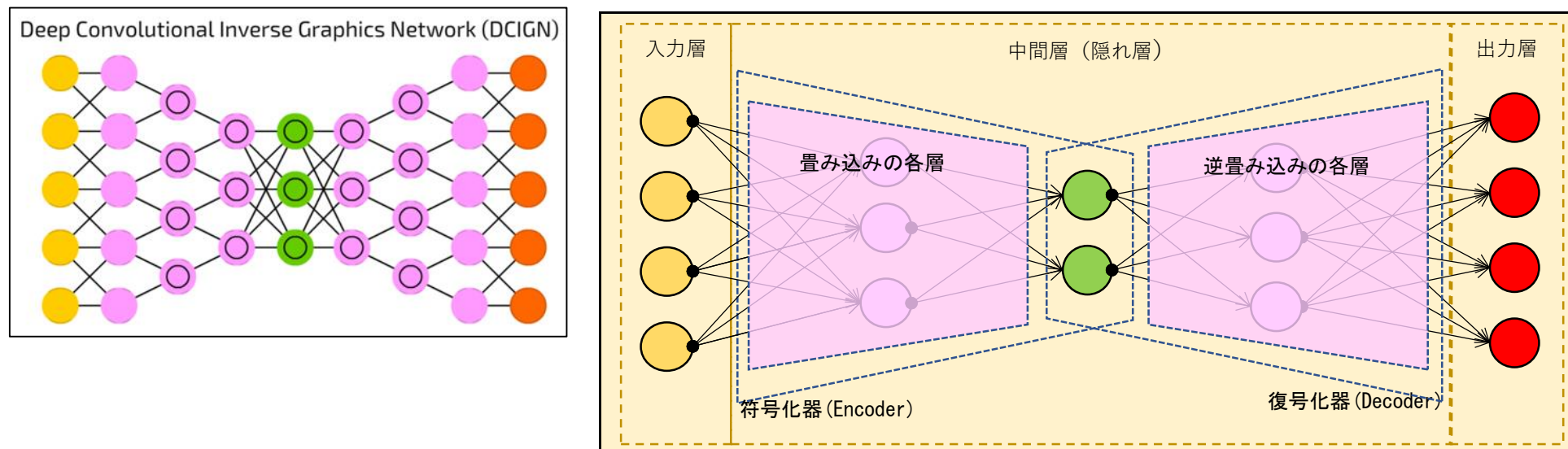
- ・ (例2)の左側は、 $5 \times 5$ の画像(下図)に対して、フィルタのサイズを $3 \times 3$ 、ストライドのサイズを $2 \times 2$ 、パディングあり(片側サイズ1)で畳み込みを行った結果、 $3 \times 3$ の特徴マップが作成されたものです(上図)。
- ・ (例2)の右側は、これで作成された $3 \times 3$ の特徴マップ(下図、但しストライドのサイズが $2 \times 2$ なので、間が1空いている。更にパディングあり(片側サイズ1)なので元画像と同じ $5 \times 5$ の画像が、逆畳み込みの対象画像になる)に対し、フィルタのサイズ/ストライドのサイズ/パディングの有無を考慮して逆畳み込みを行い、 $5 \times 5$ の画像(上図)を復元したものです。

### 【出典・参考】

⇒ <http://www.matthewzeiler.com/wp-content/uploads/2017/07/cvpr2010.pdf>

⇒ [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

#### (4.19) Deep Convolutional Inverse Graphics Network (DCIGN)

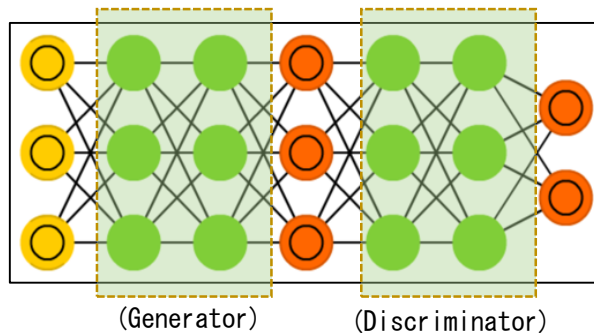
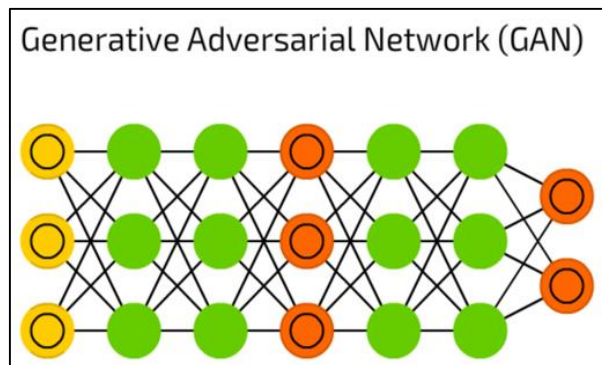


- ・ Deep Convolutional Inverse Graphics Network (DCIGN) は、DCN と DN をつなげてできたネットワークのように見えます。DCIGN は実際は、自動エンコーダー (AE) で、DCN と DN は、各々符号化器 (Encoder) と復号化器 (Decoder) として機能します。
- ・ DCIGN は画像処理に主に使用され、事前学習で訓練されていない画像を処理することができます。
- ・ DCIGN を用いて特定のオブジェクトをイメージから削除したり、塗りなおしたり、馬をシマウマで置き換える、・・・といったことができます。

#### 【出典・参考】

<https://www.google.co.jp/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=0ahUKEwjNr8rNtqDcAhXCzLwKHe8RCNUQFgh7MAg&url=https%3A%2F%2Fdspace.mit.edu%2Fopenaccess-disseminate%2F1721.1%2F112752&usg=AOvVaw3T178nAt8DVHt1CVoNCXKt>

#### (4.20) 敵対的生成ネットワーク (Generative Adversarial Network, GAN)



- ・ 敵対的生成ネットワーク (てきたいてきせいせいネットワーク、Generative Adversarial Network, GAN) は、訓練データを学習し、そのデータと似たような新しいデータを生成する「生成モデル」の一種で、2014年に Ian Goodfellow 等によって提案されました。GAN のうち、多段の畳み込みネットワークを利用しているものは深層畳み込み敵対的生成ネットワーク (Deep Convolutional Generative Adversarial Network, DCGAN) といい、高解像な画像の生成が可能になりました。
- ・ GAN は生成ネットワーク (Generator) と識別ネットワーク (Discriminator) の2つのネットワークから構成されます。例として画像生成を目的とする場合、
  - (1) 生成ネットワークは本物と同じような画像を生成しようとします。
  - (2) 識別ネットワークは生成された画像がレプリカか本物なのかを識別します。生成側は識別側を欺こうと (本物らしい画像を作成できるように) 学習し、識別側はより正確に本物と偽物を識別しようと学習します。このように2つのネットワークが相反した目的のもとに学習する様が敵対的と呼ばれる所以となっています。
- ・ 識別ネットワークの識別能力が次第に上がり、本物とレプリカをうまく見分けられるようになったとすると、生成ネットワークは更に本物に近いレプリカを造るようになります。更に識別ネットワークが本物とレプリカを見分けられるようにさらに精度を上げて・・・と繰り返していくと、最終的には本物と区別が付かないレプリカを、生成ネットワークが製造できるようになると期待できます。
- ・ GANの登場は、人工知能が、何かを単に調べるだけの作業だけではなく、何かを生み出す、すなわち「生成」の作業にも活用できることを意味します。つまり人工知能は答えを与えてもらう受け身な「受動的認識」から、自発的に生み出せる「能動的行動」が可能となりました。今後、GANは、画像データだけではなく、音声や自然言語などのデータにも適用される見通しで、将来は、音声生成や編集、音声変換や復元なども可能になるのではないかと注目されています。

【出典・参考】

⇒ <https://ja.wikipedia.org/wiki/敵対的生成ネットワーク>

⇒ <https://arxiv.org/pdf/1406.2661v1.pdf>

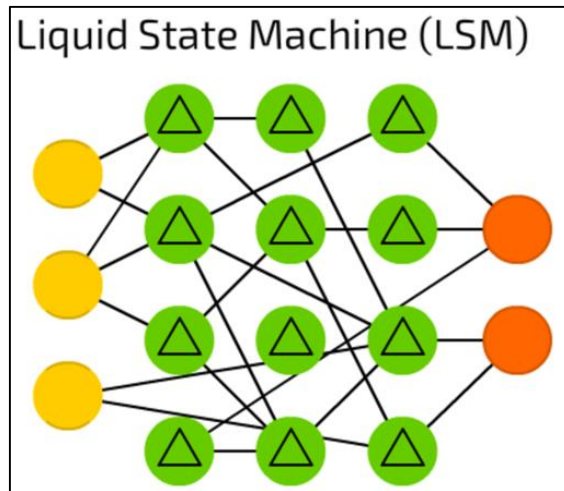
⇒ <https://arxiv.org/pdf/1511.06434.pdf>

⇒ <http://db-event.jpn.org/deim2017/papers/111.pdf>

⇒ <https://roboteer-tokyo.com/archives/8066>

(教材2)「現場で使える！TensorFlow開発入門 Kerasによる深層学習モデル構築手法」P. 251 (2018年04月 翔泳社 太田満久 他著)

#### (4.21) リキッド・ステート・マシン (Liquid State Machine, LSM)



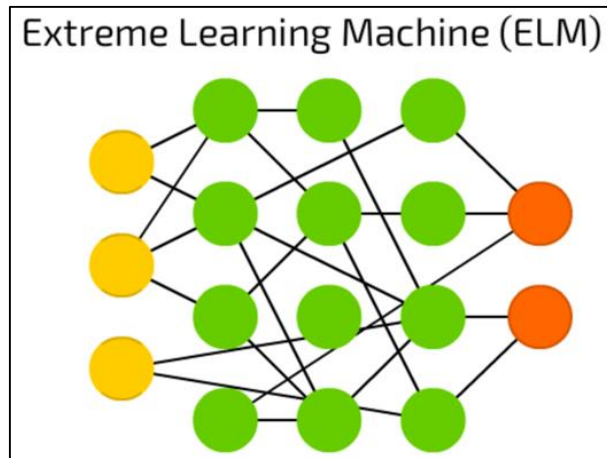
- ・リキッド・ステート・マシン (Liquid State Machine, LSM) は、Maass, Wolfgang, Thomas Natschläger, Henry Markram 等が2002年に発表しました。
- ・LSM ではニューロン間はランダム接続しており、きちんとした層群に編成されていません。
- ・シグモイド関数は閾値関数に置き換えられ、各ニューロンも蓄積記憶セルです。そのため、ニューロンが更新されると、隣接するニューロンとの合計数にはならず、独自に蓄積された数になります。閾値に達するまで何も起きませんが、閾値に達すると他のニューロンへ放出されるため、急上昇したかのようなパターンを描きます。

(※申し訳ありませんが、これは「<https://postd.cc/neural-network-zoo-latter/>」の訳出のままで載せています。)

##### 【出典・参考】

- ⇒ <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.709.4645&rep=rep1&type=pdf>
- ⇒ [https://web.archive.org/web/20120222154641/http://ramsesii.upf.es/seminar/Maass\\_et\\_al\\_2002.pdf](https://web.archive.org/web/20120222154641/http://ramsesii.upf.es/seminar/Maass_et_al_2002.pdf)
- ⇒ <http://www.jnns.org/20/yamazaki.pdf>
- ⇒ [https://en.wikipedia.org/wiki/Liquid\\_state\\_machine](https://en.wikipedia.org/wiki/Liquid_state_machine)

#### (4.22) エクストリーム・ラーニング・マシン (Extreme Learning Machine, ELM)

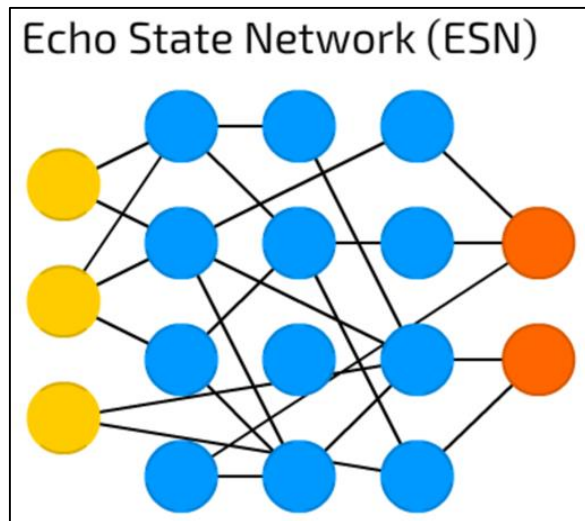


- ・ エクストリーム・ラーニング・マシン (Extreme Learning Machine, ELM) は、基本的には FF ですが、ランダム接続になります。  
LSMやESNに非常によく似ていますが、回帰も活動電位もありません。  
バックプロパゲーションも使用しません。  
その代わり、ランダムな重みから始め、(関数の中でも最も誤差が小さい) 最小二乗法に従って重みを1段階ずつ訓練していきます。  
結果としては表現力のあまりないネットワークができますが、誤差逆伝播法(バックプロパゲーション)より断然高速です。
- ・ Cambria, Erikらが、2013年に発表したものです。

(※申し訳ありませんが、これは「<https://postd.cc/neural-network-zoo-latter/>」の訳出のままで載せています。)



#### (4.23) エコー・ステート・ネットワーク (Echo State Network, ESN)

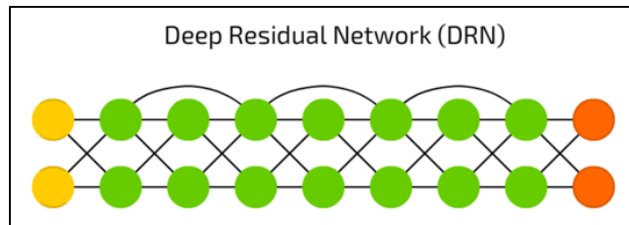


- ・エコー・ステート・ネットワーク (Echo State Network, ESN) は、RNN の一つで、Jaeger, Herbert, Harald Haas らが 2004年に発表しました。
- ・このタイプのネットワークはニューロン間にランダム接続を持ち（つまり、きちんとした層群に編成されていません）、異なる方法で訓練されます。入力して誤差逆伝播法（バックプロパゲーション）でエラーを見つけるのではなく、入力した情報を転送しニューロンを少しの間更新し、時間の経過とともに出力を観察します。入力層はネットワークを下塗りするために使用され、出力層は時間をかけて展開するアクティベーションパターンを観察するオブザーバ役を務めます。このように、入力層と出力層は少し型破りな役割を担っています。訓練の際、オブザーバと隠れユニット（の集合体）間の接続のみが変更されます。

（※申し訳ありませんが、これは「<https://postd.cc/neural-network-zoo-latter/>」の訳出のままで載せています。）

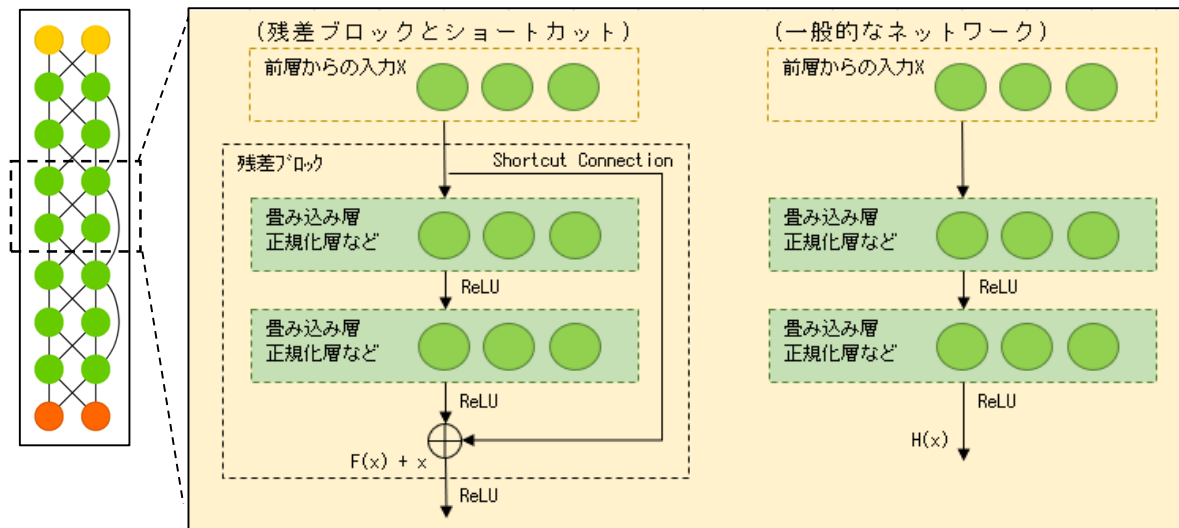


## (4.24) Deep Residual Network (DRN)



- Deep Residual Network (DRN) は Microsoft Research (現Facebook AI Research) のKaiming He氏が2015年に考案したニューラルネットワークのモデルです。

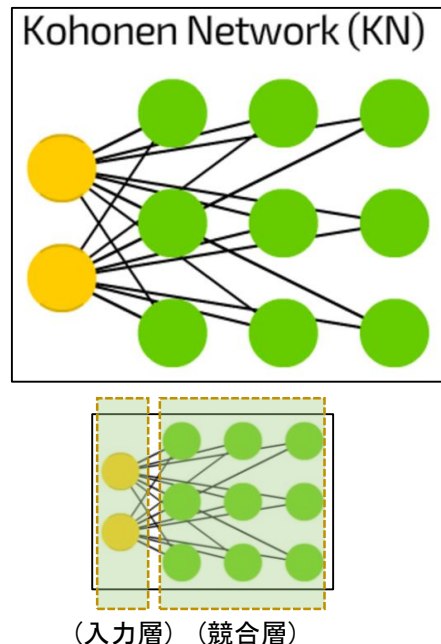
- ネットワークを深層化することは表現能力を向上させ、認識精度を改善しますが、あまりにも深いネットワークは効率的な学習が困難でした。
- DRN では、「ある層で求める最適な出力  $H(x)$  を学習するのではなく、層の入力  $x$  を参照した残差関数  $F(x)$  を学習する」ことで最適化しやすくしています。層の入力  $x$  と層の出力  $H(x)$  との差分  $F(x)$  は「 $F(x) = H(x) - x$ 」で、この関係式から「 $H(x) = F(x) + x$ 」を学習するように再定義します。
- DRN では、「残差ブロック (residual block)」と「ショートカット (Shortcut Connection)」を導入することでこれを実現します。DRN では、通常のネットワークのように、何かしらの処理ブロックによる変換  $H(x)$  を単純に次の層に渡していくのではなく、その処理ブロックへの入力  $x$  をショートカットし、「 $H(x) = F(x) + x$ 」を次の層に渡すことが行われます。このショートカットを含めた処理単位を「残差ブロック (residual block)」と呼びます。DRN では、ショートカットを通して、誤差逆伝播時に勾配が直接下層に伝わっていくことになり、非常に深いネットワークにおいても効率的に学習ができるようになりました。



### 【出典・参考】

- ⇒ <https://arxiv.org/pdf/1409.1556v6.pdf>
- ⇒ [http://www.vision.cs.chubu.ac.jp/MPRG/F\\_group/F188\\_uchida2017.pdf](http://www.vision.cs.chubu.ac.jp/MPRG/F_group/F188_uchida2017.pdf)
- ⇒ [https://deeppage.net/deep\\_learning/2016/11/30/resnet.html](https://deeppage.net/deep_learning/2016/11/30/resnet.html)
- ⇒ <https://arxiv.org/pdf/1606.08921.pdf>

#### (4. 25) コホーネンネットワーク (Kohonen Network, KN)



・コホーネンネットワーク (Kohonen Network, KN)はヘルシンキ大学の Kohonen(コホーネン)教授が、1982年に考案したもので、自己組織化写像 (Self Organizing Map, SOM)とも呼ばれます。KN では、分類指標が無いときに類似度 (距離ともいいます) を元にして競合学習を使用し教師なしにデータを分類することができます。

応用例として、体の様々な形状や大きさ等の特徴に基づく動物の分類等や、  
或る文学作品の全文を入力し、登場する単語間の相関関係等を調べた例もありません。

・KN は、基本的には入力層と競合層 (出力層) からなる2層構造で、  
入力層のユニット  $X_i$  ( $i=1\sim N$ ) は全ての競合層のユニット  $Y_j$  ( $j=1\sim K$ ) と結合し、結合係数  $W(i, j)$  を持ちます。  $W(i, j)$  ( $i=1\sim N, j=1\sim K$ ) の初期値は任意ですが、 $[0\sim 1]$  の乱数等で与えられます。  
競合層のユニット  $Y_j$  の結合係数の組  $(W(1, j), W(2, j), \dots, W(N, j))$  を参照ベクトルと言います。  
学習が進むに連れて、参照ベクトルは入力パターンのどれかと似てきます。  
そして互いに近傍に在るユニットの参照ベクトルは、同じ入力パターンに似る事になります。  
KN による入力パターンの分類は、参照ベクトルによって行なわれます。

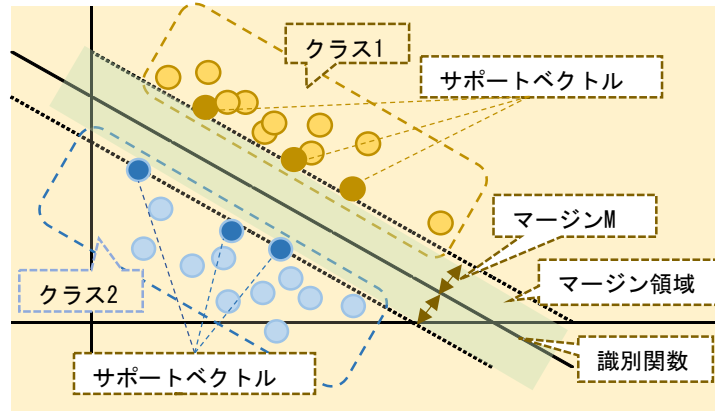
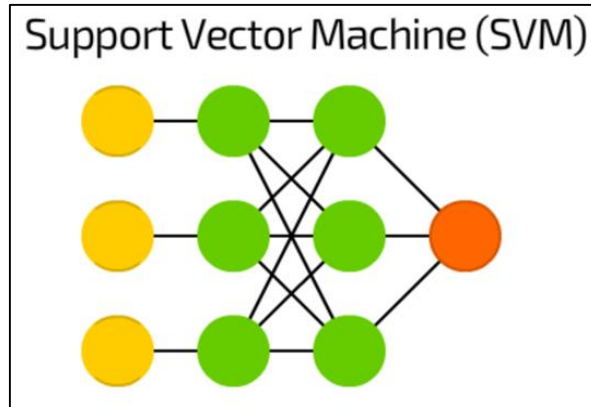
・KN の学習の手順はまとめると次のようになります。

1. 分類したい代表点を (分類の数分)、競合層のユニットとして作成する。
2. 母体のデータから任意のサンプル点を選択し、入力データとする。
3. 競合層の各ユニットと入力データとの類似度を計算する。 (類似度は地理的距離など、モデルに応じた指標)
4. 一番類似度の高い競合層のユニット (勝者と言います) を入力データに少しだけ接近させる。 (距離を  $1/10$  だけ接近するなど)
5. 競合層の特定のユニットが常に勝者とならないように、平均回数以上勝った場合は、2. ~5. の手順を暫くの間保留する。
6. 競合層の複数のユニットの変動が落ち着く (収束すると言います) まで、2. ~5. の手順を繰り返す。

#### 【出典・参考】

- ⇒ <https://cioslab.vcu.edu/alg/Visualize/kohonen-82.pdf>
- ⇒ <https://ja.wikipedia.org/wiki/自己組織化写像>
- ⇒ [http://cicsj.chemistry.or.jp/15\\_6/funa.html](http://cicsj.chemistry.or.jp/15_6/funa.html)
- ⇒ [http://www.zetta.co.jp/column/column1\\_09.htm](http://www.zetta.co.jp/column/column1_09.htm)
- ⇒ <http://www.sist.ac.jp/~kanakubo/research/neuro/selforganizingmap.html>

#### (4.26) サポートベクターマシン (Support Vector Machine, SVM)

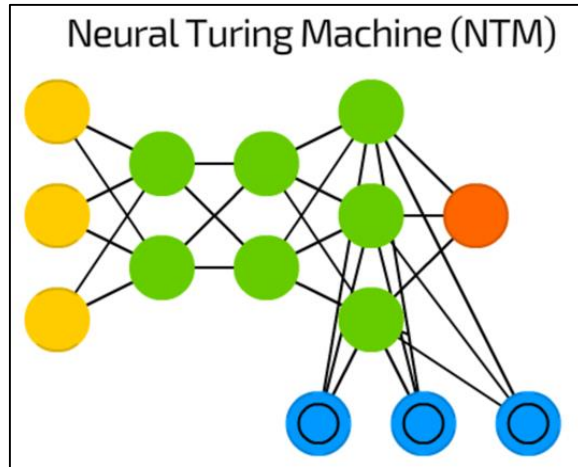


- ・ サポートベクターマシン (Support Vector Machine, SVM) は、教師あり学習を用いるパターン認識モデルの一つで、分類や回帰へ適用できます。1963年に Vladimir N. Vapnik, Alexey Ya. Chervonenkis が線形サポートベクターマシンを発表し、1992年に Bernhard E. Boser, Isabelle M. Guyon, Vladimir N. Vapnik が非線形へと拡張しました。
- ・ SVM はデータの分布を分ける溝を決める手法の一つで、パターン認識におけるデータの分類をすることが出来ます。SVM では正例と負例のデータからの距離を最大にして真ん中を通る「識別関数」を解析的に求めます（正値／負値は識別関数の値です）。この距離を「マージン」と呼び、マージン最大化を行いながら識別関数を決めます。識別関数を挟んでマージンの幅内の領域を「マージン領域」と呼びます。マージン領域の端に相当する場所（分類するクラスの境界の近く）に位置するデータを「サポートベクトル」と呼びます。このサポートベクトルのみを用いて、クラス分類を行います。
- ・ 「非線形な問題で利用できない」という問題は「カーネルトリック」を導入して解決しました。「カーネルトリック」とは、元々のデータ空間から高次元空間にデータを写像し、その高次元空間上で線形データ解析を行うことです。

#### 【出典・参考】

- ⇒ [http://image.diku.dk/imagecanon/material/cortes\\_vapnik95.pdf](http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf)
- ⇒ <http://home.hiroshima-u.ac.jp/tkurita/lecture/svm.pdf>
- ⇒ <https://qiita.com/pesuchin/items/c55f40b69aa1aec2bd19>

#### (4.27) ニューラルチューリングマシン (Neural Turing Machine, NTM)



- ・ニューラルチューリングマシン (Neural Turing Machine, NTM) は、2014年に米グーグルの DeepMind が発表したモデルです。
- ・人間がプログラムで記述するような複雑なアルゴリズムの実現には、途中の計算結果を保存するメモリーが不可欠です。しかし、従来、ニューラルネットワークなどコンピュータによる学習には記憶の仕組みが欠けていました。  
状態（途中結果）を持つ問題を扱うためには従来、RNN (recurrent neural network) が利用されてきましたが、短期記憶やワーキングメモリのようなものであり、その容量はニューロン数で制限されます。
- ・NTMでは、ニューラルネットワークの外側にある外部記憶（左図では青の二重丸）を利用することにより、ニューロン数に制約されない記憶を実現しました。

- ・NTMは、コントローラと外部記憶で構成されます。

コントローラはニューラルネットワークで構成し、外部記憶へアクセスを行う「読み込みヘッダ」と「書き込みヘッダ」の状態を決めます。

コントローラにRNNを利用することもできます。

この場合、コンピュータでいえばRNNの内部状態がレジスタに相当し、外部記憶がメモリに相当します。

NTMではヘッダを含む全ての構成要素が微分可能となっているため、逆誤差伝播法を用いて学習できます。

#### 【出典・参考】

⇒ <https://tech.nikkeibp.co.jp/dm/atcl/mag/15/00144/00008/>

(5) 確認問題

以下の空欄に最もあてはまる用語を選択肢から選び、その記号を回答欄に記入してください。（回答提出用の Excel ファイルは別途配布します）

(1) 【機械学習アルゴリズムの使い分け】

機械学習アルゴリズムには様々なものがあり、目的に応じて使い分けます。

- ・ (1.1) \_\_\_\_\_ には、複数のカテゴリーに分類する「マルチ分類 (Multi-class classification)」と 2つのカテゴリーに分類する「2クラス分類 (Two-class classification)」があります。
- ・ (1.2) \_\_\_\_\_ は、実績データをもとに、入力データと出力データの数値間の関連性を導き出し、入力データに対する出力データを数値で予測するものです。
- ・ (1.3) \_\_\_\_\_ もまた、分類するための手法です。  
分類するという点では (1.2) \_\_\_\_\_ と同じですが、(1.3) \_\_\_\_\_ は (1.2) \_\_\_\_\_ と異なり、似たデータを集めてデータ構造を発見する「教師なし学習」です。
- ・ (1.4) \_\_\_\_\_ は、多くの次元を持ったものを、その意味を保持しながら（落としても影響のない次元を削減して）少ない次元にする手法です。
- ・ (1.5) \_\_\_\_\_ は、現在の異常を検知したり、未来の予知をして保全に備えたりする為の技術です。

- (選択肢)    (a) 「クラスタリング (Clustering)」  
                  (b) 「次元削減 (dimensionality reduction)」  
                  (c) 「回帰 (Regression)」  
                  (d) 「分類 (Classification)」  
                  (e) 「異常検出 (Anomaly Detection)」

(回答)	(1.1)	
	(1.2)	
	(1.3)	
	(1.4)	
	(1.5)	

(2) 【人工知能・機械学習・ディープラーニングの包含関係】

AI：「人工知能 (artificial intelligence)」、ML：「機械学習 (Machine Learning)」、DL：「深層学習 (Deep Learning)」という3つのカテゴリーの集合関係として正しいのは、以下の(a), (b), (c)のうち、(2.1) \_\_\_\_\_ である。

- (選択肢)    (a) AI ⊃ ML ⊃ DL  
                  (b) ML ⊃ AI ⊃ DL  
                  (c) DL ⊃ AI ⊃ ML

(回答)	(2.1)	
------	-------	--

## (5) 確認問題

### (3) 【ニューラルネットブームの流れ】

- ・ローゼンブラットが1958年に発表した (3.1) は、第一次ニューラルネットブームを巻き起こしましたが、「XOR問題」が解決できないといった線形分離問題があり、間もなくブームも下火になりました。
- ・その解決策として、1986年にデビッド・ラメルハートらが多層化した (3.2) を考案し、第二次ニューラルネットブームを起こしました。
- ・2006年に Geoffrey Hinton と共同研究者が、(3.3) を多数重ねて積み上げて多層のネットワークモデルとして考案した (3.4) は、今日まで続く第三次ニューラルネットブームの始まりとなりました。

- (選択肢) (a) 「深層信念ネットワーク (Deep Belief Network, DBN)」  
(b) 「単純パーセプトロン (Perceptron)」  
(c) 「制限ボルツマンマシン (Restricted Boltzmann Machine, RBM)」  
(d) 「順伝播型ニューラルネットワーク (Feed forward neural networks)」

(回答)

(3.1)	
(3.2)	
(3.3)	
(3.4)	

### (4) 【ニューラルネットワークの層】

- ・神経細胞である「ニューロン (neuron)」をモデル化した (4.1) を複数つなげて「数理的な神経回路網」を構成したものが (4.2) で、基本的に入力層、隠れ層 (中間層)、出力層の3層から構成されます。

- (選択肢) (a) 「ニューラルネットワーク (Neural Network, NN)」  
(b) 「ユニット (形式ニューロン)」

(回答)

(4.1)	
(4.2)	

### (5) 【活性化関数】

- ・「活性化関数 (activation function)」は「伝達関数 (transfer function)」ともいい、重み付き入力に応じて、出力信号の大きさを規定する関数です。活性化関数の一つとしてシグモイド関数  $\sigma$  があり、微分可能で解析的に扱いやすいという良い性質がありますが、誤差逆伝播時の (5.1) があり、代わりに「ReLU (ランブ関数)」などがよく用いられます。

- (選択肢) (a) 勾配消失問題  
(b) XOR問題  
(c) 線形分離問題

(回答)

(5.1)	
-------	--



## (5) 確認問題

### (6) 【過去の状態値を再度ネットワークに取り込むモデル】

- ・過去の状態値を再度ネットワークに取り込むモデルの一つが、(6.1) で、その発展形が「Long/Short Term Memory (LSTM)」、「Gated Recurrent Unit (GRU)」です。これらのモデルは、音声認識やテキスト解析といった、前後の文脈が意味を持つような問題に適用されます。
- ・然しながら、これらのモデルは短期記憶やワーキングメモリのようなものであり、その容量はニューロン数で制限されます。それを解決するモデルとして、2014年に米グーグルのDeepMindが発表したものが (6.2) です。(6.3) では、ニューラルネットワークの外側にある外部記憶を利用することにより、ニューロン数に制約されない記憶を実現しました。

- (選択肢) (a) 「回帰結合ニューラルネットワーク (Recurrent neural network、RNN)」  
(b) 「ニューラルチューリングマシン (Neural Turing Machine、NTM)」  
(c) 「敵対的生成ネットワーク (Generative Adversarial Network、GAN)」

(回答)

(6.1)	
(6.2)	
(6.3)	

### (7) 【オートエンコーダーの仲間】

- ・オートエンコーダー(自己符号化器 Autoencoder)とは、出力データが入力データに近づくように学習を行う、ニューラルネットワークモデルです。
- ・具体的には、入力データを中間層に圧縮しようとする (7.1) と中間層を入力データに復元しようとする (7.2) を直接つないだ「砂時計型ニューラルネットワーク (hourglass-type neural network)」として構成されるシンプルな構造になっています。
- ・オートエンコーダーを発展させたモデルまたは類似のモデルとして、「Variational Autoencoder (VAE)」、「Denoising Autoencoder (デノイジングオートエンコーダー、DAE)」、「Sparse Autoencoder (SAE)」、更に畳み込み構造を入れた「Deep Convolutional Inverse Graphics Network (DCIGN)」などがあります。

- (選択肢) (a) 復号化器(Decoder)  
(b) 符号化器(Encoder)

(回答)

(7.1)	
(7.2)	

## (5) 確認問題

### (8) 【脳の働きに近いとみられているネットワーク】

- ・「Sparse Autoencoder (SAE)」の構造では、隠れセル数は入出力層セル数よりも (8.1) \_\_\_\_\_ なっています。  
SAEでは、中間層で活性化するニューロンがより少なくなり (8.2) \_\_\_\_\_ になります。
- ・「脳では少数のニューロンが反応することで複雑な表現をする機能があるのではないか」という仮説があり、この機構をスパースコーディング (Sparse Coding) と呼んでいます。  
このスパースコーディングが「汎化性能と効率の向上の鍵ではないか」と言われており、機械学習では重要なテーマになっています。

- (選択肢) (a) 大きく  
(b) 小さく  
(c) 疎 (sparse) に

(回答)

(8.1)	
(8.2)	

### (9) 【マルコフ連鎖】

- ・「マルコフ連鎖 (マルコフれんさ、Markov chain, MC)」とは、  
(9.1) \_\_\_\_\_ のうち、とりうる状態が離散的 (有限または可算) なもの (離散状態マルコフ過程) をいいます。  
また特に、時間が離散的なもの (時刻は添え字で表される) を指すことが多いです。  
マルコフ連鎖は、未来の挙動が現在の値だけで決定され、過去の挙動と無関係です。
- ・「未来の挙動が現在の値だけで決定され、過去の挙動と無関係であるという性質」を (9.2) \_\_\_\_\_ と言います。  
各時刻において起こる状態変化 (遷移または推移) に関して、マルコフ連鎖は遷移確率が過去の状態によらず、現在の状態のみによる系列です。
- ・ (9.1) \_\_\_\_\_ とは、マルコフ性をもつ確率過程のことをいいます。  
このような過程は例えば、確率的にしか記述できない物理現象の時間発展の様子に見られます。

- (選択肢) (a) マルコフ性  
(b) 「マルコフ連鎖 (マルコフれんさ、Markov chain, MC)」  
(c) 「マルコフ過程 (マルコフかてい)」

(回答)

(9.1)	
(9.2)	



## (5) 確認問題

### (10) 【ホップフィールド・ネットワークの仲間】

- ・ホップフィールド・ネットワーク (Hopfield network, HN) は、  
ジョン・ホップフィールド (J. J. Hopfield) が1982年に提唱したニューラルネットワークモデルです。
- ・HNは、各ユニット間に対称的な相互作用がある (10.1) \_\_\_\_\_ で、信号は双方向にやり取りします。  
(10.1) \_\_\_\_\_ というのは、或る時刻に一つのユニットだけが他のユニットからの入力刺激の総和に基づき状態更新を行なうものです。
- ・ボルツマンマシン (Boltzmann Machine, BM) は、HNの一種です。  
全てが隠れユニットで構成される HN と異なり、幾つかのユニットが入力ユニットになります。
- ・最急降下法による誤差逆伝播法や HN が持つ学習時の局所解への捕捉の問題に対応する為に、  
BM では、ネットワークの動作に温度の概念を取り入れ、最初は激しく徐々に穏やかに逐次的に最適解に近づくように工夫しています。  
これを (10.2) \_\_\_\_\_ と言います。
- ・制限ボルツマンマシン (Restricted Boltzmann Machine, RBM) は、BM の一種であり、  
可視層 (Visible Layer) と隠れ層 (Hidden Layer) の2層構造を持ちます。  
RBM は、可視層のユニットと隠れ層のユニット間でのみ接続しているという制限が付いたモデルです。
- ・2006年に Geoffrey Hinton と共同研究者が、RBM を多数重ねて積み上げて多層のネットワークモデルとして考案した  
「深層信念ネットワーク (Deep Belief Network, DBN)」は、今日まで続く第三次ニューラルネットブームの始まりとなりました。  
最上位層としてソフトマックス層 (ソフトマックス関数を適用する層) を追加することで、教師データとのすり合わせを行い、  
下層の全ネットワークに対して逆伝播を行うようにすることが出来ます。これにより教師あり学習として振る舞うことが可能になります。

- (選択肢) (a) 同期型ネットワーク  
(b) 非同期型ネットワーク  
(c) 「擬似焼きなまし法 (Simulated annealing)」  
(d) 逐次近似法

(回答)

(10.1)	
(10.2)	

## (5) 確認問題

### (11) 【画像認識向きのモデル】

- ・「深層畳み込みニューラルネットワーク (Deep Convolutional neural network、DCN)」は、「順伝播型ニューラルネットワーク (Feed forward neural networks、FF、FFNN)」に畳み込み構造を入れ込んだモデルです。これは、人の視覚をモデルにした構成になっており、画像認識分野を中心に幅広く利用されています。
- ・このモデルでは、中間層（隠れ層）が、(11.1) \_\_\_\_\_ 、 (11.2) \_\_\_\_\_ 、 (11.3) \_\_\_\_\_ 、 (11.4) \_\_\_\_\_ で構成された複数の層からなります (11.1) \_\_\_\_\_ でフィルタの移動量 (ストライド、stride) を調整することにより情報縮約を行い、(11.3) \_\_\_\_\_ を代行することもあります。
- ・このモデルを逆にして、画像のフィルタ結果から元画像を復元しようとするのが (11.5) \_\_\_\_\_ です。これは「転置畳み込み (Transposed Convolution)」と言われることが多いです。
- ・「深層畳み込みニューラルネットワーク (Deep Convolutional neural network、DCN)」と「逆畳み込みネットワーク (Deconvolutional Network, DN)」をつなげてできたネットワークが、(11.6) \_\_\_\_\_ で、画像処理に主に使用され、事前学習で訓練されていない画像を処理することができます。

- (選択肢) (a) プーリング層  
(b) 正規化層  
(c) 全結合層  
(d) 畳み込み層 (コンボリューション層)  
(e) 「Deep Convolutional Inverse Graphics Network (DCIGN)」  
(f) 「逆畳み込みネットワーク (Deconvolutional Network, DN)」

(回答)

(11.1)	
(11.2)	
(11.3)	
(11.4)	
(11.5)	
(11.6)	

### (12) 【ネットワークの深層化】

- ・「Deep Residual Network (DRN)」は Microsoft Research の Kaiming He 氏が2015年に考案したニューラルネットワークのモデルです。DRN では、(12.1) \_\_\_\_\_ と (12.2) \_\_\_\_\_ を導入することで、1000層以上の深いネットワークにおいても効率的に学習ができるようになりました

- (選択肢) (a) 「ショートカット (Shortcut Connection)」  
(b) 「交差エントロピー誤差 (cross entropy)」  
(c) 「残差ブロック (residual block)」

(回答)

(12.1)	
(12.2)	

## (5) 確認問題

### (13) 【生成モデル】

- ・「敵対的生成ネットワーク(てきたいてきせいせいネットワーク、Generative Adversarial Network, GAN)」は、訓練データを学習し、そのデータと似たような新しいデータを生成する「生成モデル」の一種で、2014年に Ian Goodfellow 等によって提案されました。GAN のうち、多段の畳み込みネットワークを利用しているものは「深層畳み込み敵対的生成ネットワーク (Deep Convolutional Generative Adversarial Network, DCGAN)」といい、高解像な画像の生成が可能になりました。
- ・GAN は (13.1)\_\_\_\_\_ と (13.2)\_\_\_\_\_ の2つのネットワークから構成されます。
- ・GANの登場は、人工知能が、何かを単に調べるだけの作業だけではなく、何かを生み出す、すなわち「生成」の作業にも活用できることを意味します。つまり人工知能は答えを与えてもらう受け身な「受動的認識」から、自発的に生み出せる「能動的行動」が可能となりました。今後、GANは、画像データだけではなく、音声や自然言語などのデータにも適用される見通しで、将来は、音声生成や編集、音声変換や復元なども可能になるのではないかと注目されています。

- (選択肢) (a) 「復号化器 (Decoder)」  
(b) 「符号化器 (Encoder)」  
(c) 「識別ネットワーク (Discriminator)」  
(d) 「生成ネットワーク (Generator)」

(回答)	(13. 1)	
	(13. 2)	

### (14) 【サポートベクターマシン】

- ・「サポートベクターマシン (Support Vector Machine, SVM)」は、教師あり学習を用いるパターン認識モデルの一つで、分類や回帰へ適用できます。
- ・SVM はデータの分布を分ける溝を決める手法の一つで、パターン認識におけるデータの分類をすることが出来ます。SVM では正例と負例のデータからの距離を最大にして真ん中を通る (14. 1)\_\_\_\_\_ を解析的に求めます (正值／負値は (14. 1)\_\_\_\_\_ の値です)。この距離を「マージン」と呼び、マージン最大化を行いながら (14. 1)\_\_\_\_\_ を決めます。(14. 1)\_\_\_\_\_ を挟んでマージンの幅内の領域を「マージン領域」と呼びます。マージン領域の端に相当する場所 (分類するクラスの境界の近く) に位置するデータを (14. 2)\_\_\_\_\_ と呼びます。この (14. 2)\_\_\_\_\_ のみを用いて、クラス分類を行います。

- (選択肢) (a) 「伝達関数」  
(b) 「識別関数」  
(c) 「クラスタ」  
(d) 「サポートベクトル」

(回答)	(14. 1)	
	(14. 2)	

(6) 確認問題回答用紙

提出者 : \_\_\_\_\_  
提出日 : \_\_\_\_\_年 \_\_\_\_\_月 \_\_\_\_\_日

回答

No.	回答	正誤
(1. 1)		
(1. 2)		
(1. 3)		
(1. 4)		
(1. 5)		
(2. 1)		
(3. 1)		
(3. 2)		
(3. 3)		
(3. 4)		
(4. 1)		
(4. 2)		
(5. 1)		
(6. 1)		
(6. 2)		
(6. 3)		
(7. 1)		
(7. 2)		

No.	回答	正誤
(8. 1)		
(8. 2)		
(9. 1)		
(9. 2)		
(10. 1)		
(10. 2)		
(11. 1)		
(11. 2)		
(11. 3)		
(11. 4)		
(11. 5)		
(11. 6)		
(12. 1)		
(12. 2)		
(13. 1)		
(13. 2)		
(14. 1)		
(14. 2)		

採点結果

 / 36