人工知能A

Topic 7: Neural Networks

7

ニューラルネットワーク

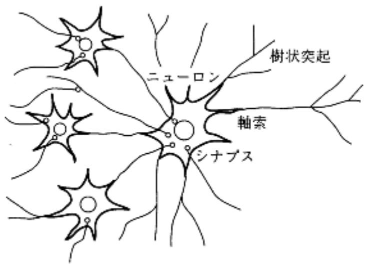
- 目標：ニューラルネットワークの基礎について学ぶ。

- 講義の内容
  - ニューラルネットワークの基本概念
  - 教師無し学習
    - 自己組織化マップ
  - 教師有り学習
    - 誤差逆伝搬法、
  - 深層学習の概要
    - オートエンコーダ、CNN、RNN、Deep Q-network (DQN)

3 ニューラルネットワークの基本概念

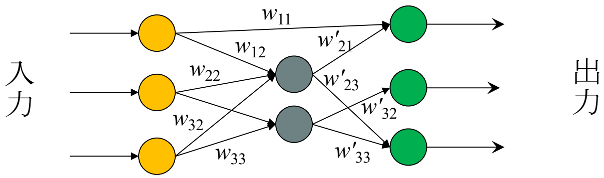
ニューラルネットワーク

- 概要
  - ニューラルネット(neural network)は、生物の神経系が、相互に結ばれた複雑なネットワークを構成していることをヒントとして考案された学習法（知識の埋込み法）。
  - ニューロンは複数の入力繊維と出力繊維を持ち、
    - 電気パルスで刺激が与えられ、
    - パルスはシナプスで化学信号に変換され、
    - その内容により次のニューロンへ電気パルスとして伝わる。



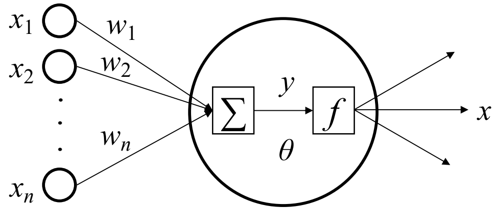
ニューラルネットワーク

- 概要
  - ニューロンの構造と電気パルスの伝達（シナプス）をモデル化。
  - 単純な計算を行うユニットが多数結合され、ネットワークを構成する。
  - それぞれの結びつきに重みが定義され（結合荷重という）、ネットワークの結びつきの強さを表す。



ニューロン

- ユニットの動作：別のユニットからの入力に重みを考慮した荷重和 $y = \sum_{i=1}^{n} w_i \cdot x_i - \theta$ を計算し、簡単な計算 $f(y)$ を施してそれを出力 (x) とし、次のユニットに渡す。
  - ここで、w_i $(i = 1, \ldots, n)$ を重み、θ は閾値。
  - このユニットをニューロン（あるいた単純にノード）といい、w_i をシナプスの結合荷重という。（シナプスはニューロン間の結合）
  - f を活性化関数という。



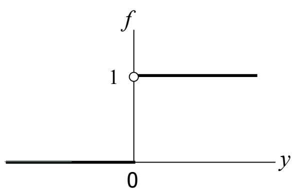
← ユニットの構成図

x_i は対応する別のユニットからの出力

ニューロンの演算 (1)

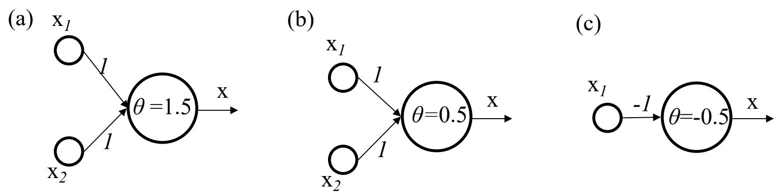
- 活性化関数 (f) にはいくつかの方法がある。以下はその例。
- 閾（しきい）素子モデル（ステップ関数 $s(x)$）
  - 入力が正数なら1を、0以下なら0を出力する（適当な数値より大きければ1としてもよい）

$$s(y) = f(y) = \begin{cases} 0, & y \leq 0 \\ 1, & y > 0 \end{cases}$$



  - 生物の細胞の反応に近く（全か無かの法則、all-or-none principle）、直感的ではあり、初期はこれが普通は使われていた。ただし、この関数自体、連続でもなければ、0では微分もできない。
  - しばらくはステップ関数を使う。

ニューロンとシナプスの演算例



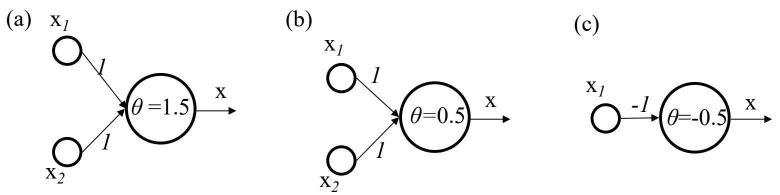
ステップ関数 s を使って

a. $x = s(1{\times}1 + 1{\times}0 - 1.5) = 0$ ➔ AND
$x = s(1{\times}1 + 1{\times}1 - 1.5) = 1$

b. $x = s(1{\times}1 + 1{\times}0 - 0.5) = 1$ ➔ OR
$x = s(1{\times}0 + 1{\times}0 - 0.5) = 0$

c. $x = s(-1{\times}1 + 0.5) = 0$ ➔ NOT
$x = s(-1{\times}0 + 0.5) = 1$

ニューロンとシナプスの演算例



<u>ステップ関数 s を使って</u>

a. $x = s(1{\times}1 + 1{\times}0 - 1.5) = 0$ ➔ AND
   $x = s(1{\times}1 + 1{\times}1 - 1.5) = 1$

b. $x = s(1{\times}1 + 1{\times}0 - 0.5) = 1$ ➔ OR
   $x = s(1{\times}0 + 1{\times}0 - 0.5) = 0$

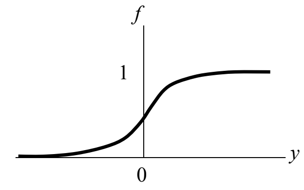
c. $x = s(-1{\times}1 + 0.5) = 0$ ➔ NOT
   $x = s(-1{\times}0 + 0.5) = 1$

でも一つのニューロンではXORは書けない。

ニューロンの演算 (2)

- 準線形素子モデル (sigmoid)
  - 微分可能とするために、シグモイド関数を使う。計算が簡単になる。

$$\varphi(y) = f(y) = \frac{1}{1+\exp(-y)}$$

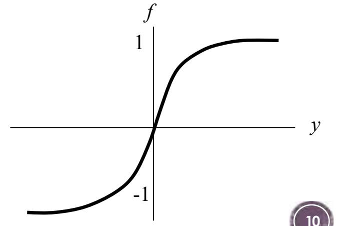


- 原点を通るsigmoid関数
  - 原点を通る方がよいという考えから、上記を線形変換したもの。

$$f(y) = \tanh(y)$$

上記の$\varphi(y)$を使うと、

$$\tanh(y) = 2\varphi(2y) - 1$$

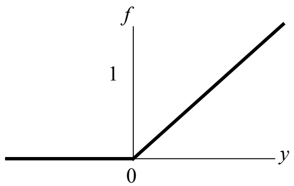


ニューロンの演算 (3)

- ReLU (Rectified Linear Unit)
  - 0付近で連続微分ではないが、 0の前後で変化が緩慢というsigmoid関数の問題点を解決できる（収束性などは示せないかもしれない）

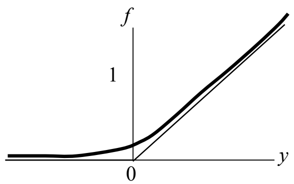
$$f(y) = \max(0, y)$$

  - 一般にこのような関数はランプ関数 (ramp function) と呼ばれる。



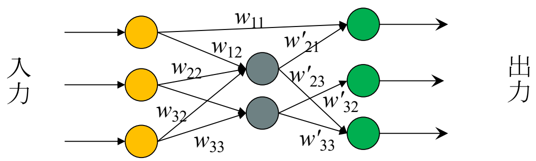
- ソフトプラス関数 (softplus)
  - ReLUを近似し、連続微分可能としたもの。

$$f(y) = \log(1 + e^y)$$



シナプスの結合荷重の学習

- 結合されたニューロンをニューラルネットという。
  - 入力に対し、望む出力となるようにシナプスの結合荷重を変更（学習）することがニューラルネットの学習となる。



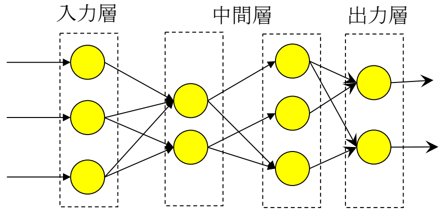
- t回目の学習のときの荷重を$w_{ij}(t)$と表す。例が一つ与えられるたびに、$w_{ij}(t)$を変更して$w_{ij}(t+1)$を求め 、適切なものに近づける。

$$w_{ij}(t+1) = kw_{ij}(t) + \Delta w_{ij} \quad (0 < k \le 1)$$
$$\Delta w_{ij}(t) = \alpha\delta_j x_i$$

- αとkは、学習効率を決定する係数で定数。x_iはw_iに対応した入力。
- δ_jは学習信号と呼ばれる値（これは後で）

ニューラルネットワークの種類

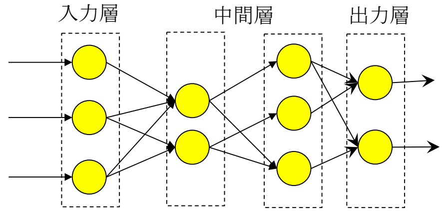
- 階層型（フィードフォワード型）ニューラルネット



一方向

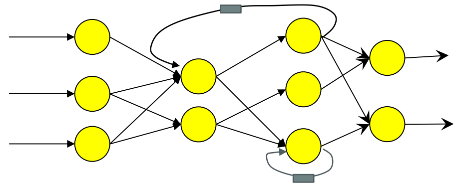
ニューラルネットワークの種類

- 階層型（フィードフォワード型）ニューラルネット



一方向

- 再帰型 (リカレント) ニューラルネット

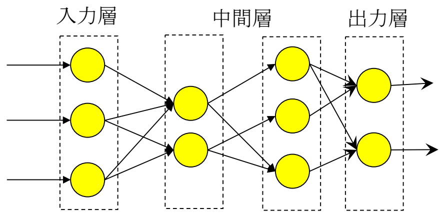


ループがある。フィードバックともいう。例えば文のように順序がある場合に使われる。
- Recurrent neural network (RNN)

次の信号と同期

ニューラルネットワークの種類

- 階層型（フィードフォワード型）ニューラルネット

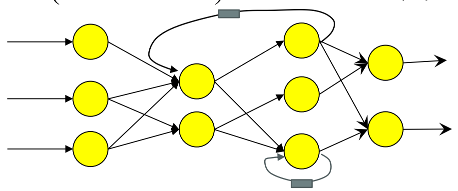


一方向

本講義では、これを仮定。
- Layered neural network
- Hierarchical neural network
- Feed-forward neural network

図では4層だが、実際にはもっとたくさんあるかもしれない。

- 再帰型 (リカレント) ニューラルネット



ループがある。フィードバックともいう。例えば文のように順序場ある場合に使われる。
- Recurrent neural network (RNN)

次の信号と同期

16 ニューラルネットワークを使った教師無し学習

教師無し学習を少しだけ紹介する。
最近の主流は教師有り学習であるが、基本として知っておくとよい。

教師なし学習(ヘッブ (D. O. Hebb))

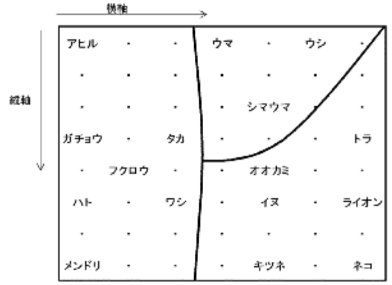
- Hebbの学習では、

$$\delta = x$$

と定義する。つまりあるニューロンの出力が大きくなる（興奮する）と、その入力も大きくなるように荷重が増える。xは出力 (x_iは入力だった)。
- この学習では、類似したある特定のパターンの出力が学習され、反応しやすくなるという効果がある（結果としてクラスタリングのような分類・抽出ができる）。
- 荷重が大きくなりすぎるので工夫は必要。

自己組織化写像 (self-organizing map, SOM)

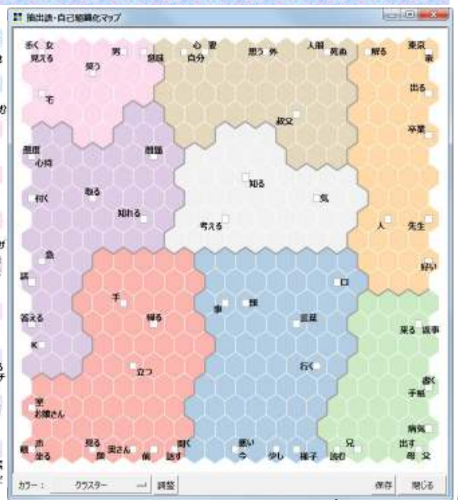
- 自己組織化写像とは、データをたとえば２次元平面上に、ある分類ごとにまとめる。たとえば、動物の特徴を分けると肉食獣、草食獣に分けられる、健康診断の数値（身長、体重、胸囲、血圧、血糖値、尿酸値、…..)などから、同傾向の人を分類（たとえば「やせ形で尿酸値が高い」グループとか。。。）、言語のもつ類似性（次ページ）など
- 視覚的に捉えることができる。他方、このため次元を高くできないので表示には限界もある。



Googleからコピー

例：共にgoogle検索から禁転用

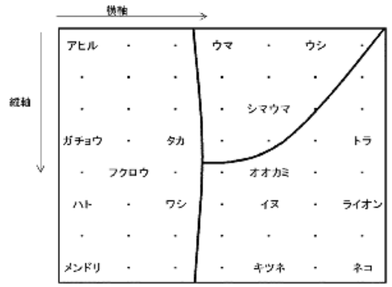
Googleからコピー



Googleからコピー

自己組織化写像 (self-organizing map, SOM)

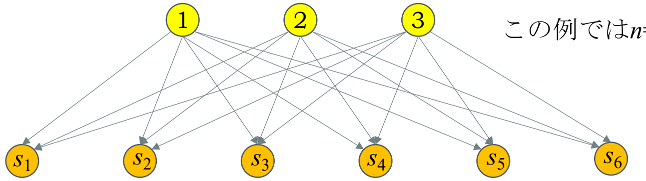
- 自己組織化写像とは、データをたとえば２次元平面上に、ある分類ごとにまとめる。たとえば、動物の特徴を分けると肉食獣、草食獣に分けられる、健康診断の数値（身長、体重、胸囲、血圧、血糖値、尿酸値、…..)などから、同傾向の人を分類（たとえば「やせ形で尿酸値が高い」グループとか。。。）、言語のもつ類似性（次ページ）など
- 視覚的に捉えることができる。
- 注意：SOMは、ニューラルネットワークを使わない方法もある。



Googleからコピー

自己組織化写像の作り方(1)

- 入力データはn次元で、N個あるとする。
  - 入力のニューロン（以下ノード）をn, 出力ノードをm (> n) 個として出力ノードを2次元 (1次元でも3次元でもよい) に並べ、全結合とする (以下の例は1次元)。
  - 出力ノード間に距離を定義できる。$dist(s_i, s_j)$と表す。
  - 結合荷重$w_{ij}(t)$は入力iから出力jノード間のものとする。初期値$w_{ij}(0)$を (0, 1)の範囲で、ランダムに設定。



この例ではn=3

自己組織化写像の作り方(2)

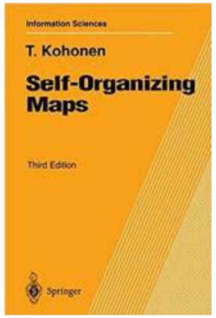
- $\boldsymbol{x}_t$=(x_{t1}, …, x_{tn})を入力層に与え、最大出力ノードをkとする。
- すべての結合荷重$w_{ij}(t)$について、以下のように再計算。

$$w_{ij}(t+1) = w_{ij}(t) + c \cdot e^{-\left(dist(s_j,s_k)/\varphi(t)\right)}\left(x_{ti} - w_{ij}(t)\right)$$

  - ここで$\varphi(t) = \frac{T-t}{T}$、$T$は学習回数の上限とする (もしくは$\varphi(t)$を温度とすることもある。$c$は正の定数で学習の重み。
  - $dist(s_j, s_k)$は、たとえば2 次元であればグリッド上に配置し、ユークリッド距離とする。他の次元でも同様。赤の部分は学習率に相当するので、他の関数もありうる。
  - 最大出力については考慮が必要 (例えばマイナスの数字の取り扱い)。これはデータを表現する各要素の意味に依存する。
  - Rなどでパッケージがあるので、詳しい人は試してみてください。

自己組織化写像（自己組織化マップ）

- SOM自体で大きな研究分野の一つであり、多くの生成アルゴリズムが提案されている。
  - あくまでここで紹介したニューラルネットワークを使ったSOMの生成は一例である
  - 効率面の工夫、高次元データ表示の工夫などが必要。
  - 興味のある人は、右のような本がある（が、今はとても高い）。



24 ニューラルネットワークを使った教師あり学習

先ずは誤り訂正学習から

教師データによる学習

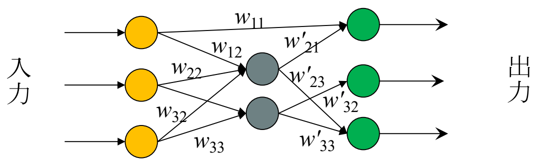
- F. Rosenbrattが1957年に提案。各入力と出力の組み合わせを教師データ（教師信号、訓練データともいう）として与える。これらの信号に合うようにニューロンが学習する。誤り訂正学習 (error-correction learning) という。
  - 入力層＋中間層＋出力層の三層構成を利用（パーセプトロン(perceptron)と呼んだ）。
  - 以下の式を使う。

$$\delta = d - x$$

  ここで、d は教師データとして与えられた出力値である。
  - たとえば、実際の出力が教師データより小さければ、$\delta > 0$となり、$x_i > 0$なら$w_i(t+1)$は $w_i(t)$ より大きくなり、$x_i < 0$ならこの逆になる（逆となるのは、出力を多くするため）。
  - なお入力層＋出力層の二層構成のものを単純パーセプトロンと呼ぶ。

シナプスの結合荷重の学習(再掲)

- 結合されたニューロンをニューラルネットという。
  - 入力に対し、望む出力となるようにシナプスの結合荷重を変更（学習）することがニューラルネットの学習となる。



  - t回目の学習のときの荷重を$w_{ij}(t)$と表す。例が一つ与えられるたびに、$w_{ij}(t)$を変更して$w_{ij}(t+1)$を求め、適切なものに近づける。

$$w_{ij}(t+1) = kw_{ij}(t) + \Delta w_{ij} \quad (0 < k \leq 1)$$
$$\Delta w_{ij}(t) = \alpha \delta_j x_i$$

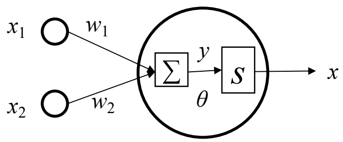
  - αとkは、学習効率を決定する係数で定数。x_iはw_iに対応した入力。
  - δ_jは学習信号と呼ばれる値（これは後で）

誤り訂正学習の例

- ２入力のニューロンで以下の誤り訂正学習の例を考る。

初期値： $w_1(0) = 0.0, w_2(0) = 0.2, \theta = 0.5, k = 0.9, \alpha = 0.2$
教師データ：$x_1 = 0, x_2 = 1, d(= x) = 1$ （これを何回か入力）

$t = 0$のとき
$x = s(w_1(0) \cdot x_1 + w_2(0) \cdot x_2 - \theta) = s(-0.3) = 0$
$w_1(1) = kw_1(0) + \alpha(d - x)x_1 = 0$
$w_2(1) = kw_2(0) + \alpha(d - x)x_2 = 0.38$



$t = 1$のとき
$x = s(w_1(1) \cdot x_1 + w_2(1) \cdot x_2 - \theta) = s(-0.12) = 0$
$w_1(2) = kw_1(1) + \alpha(d - x)x_1 = 0$
$w_2(2) = kw_2(1) + \alpha(d - x)x_2 = 0.542$

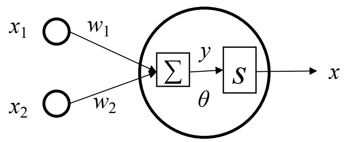
$t = 2$のとき
$x = s(w_1(2) \cdot x_1 + w_2(2) \cdot x_2 - \theta) = s(0.042) = 1$
$w_1(3) = kw_1(2) + \alpha(d - x)x_1 = 0$
$w_2(3) = kw_2(2) + \alpha(d - x)x_2 = 0.4878$

誤り訂正学習の例

- ２入力のニューロンで以下の誤り訂正学習の例を考る。

初期値： $w_1(0) = 0.0, w_2(0) = 0.2, \theta = 0.5, k = 0.9, \alpha = 0.2$
教師データ：$x_1 = 0, x_2 = 1, x = 1$ （これを何回か入力）

$t = 0$のとき
$x = s(w_1(0) \cdot x_1 + w_2(0) \cdot x_2 - \theta) = s(-0.3) = 0$
$w_1(1) = kw_1(0) + \alpha(d - x)x_1 = 0$
$w_2(1) = kw_2(0) + \alpha(d - x)x_2 = 0.38$

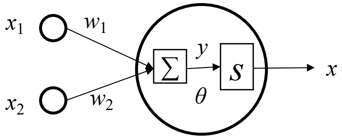


$t = 1$のとき
$x = s(w_1(1) \cdot x_1 + w_2(1) \cdot x_2 - \theta) = s(-0.12) = 0$
$w_1(2) = kw_1(1) + \alpha(d - x)x_1 = 0$
$w_2(2) = kw_2(1) + \alpha(d - x)x_2 = 0.542$

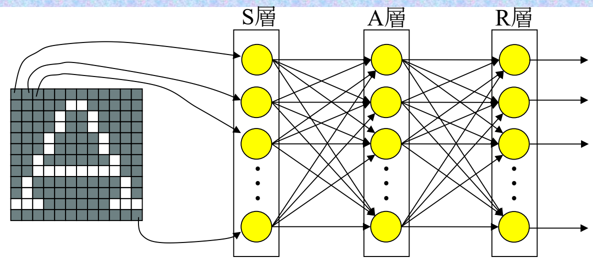
$t = 2$のとき
$x = s(w_1(2) \cdot x_1 + w_2(2) \cdot x_2 - \theta) = s(0.042) = 1$ ←ここで教師データと一致
$w_1(3) = kw_1(2) + \alpha(d - x)x_1 = 0$
$w_2(3) = kw_2(2) + \alpha(d - x)x_2 = 0.4878$ ← でもその次は一致しない。繰り返して安定する。

課題７－１

- ２入力のニューロンで以下の誤り訂正学習の例を考る。
  初期値： $w_1(0) = 0.1, w_2(0) = 0.2, \theta = 0.5, k = 0.9, \alpha = 0.2$
  教師データ：以下の信号を適当に繰り返し与えて学習させる。。
  (1) $x_1 = 0, x_2 = 1, x = 1$
  (2) $x_1 = 1, x_2 = 1, x = 1$
  (3) $x_1 = 1, x_2 = 0, x = 0$
- 前スライドの学習を行うプログラムを書き、各重みがどのように変わるか確認せよ。



例：パーセプトロンによる文字認識



- 概略（適用のイメージ）の紹介
  - 文字の入力パターン（白黒で0,1表現）に合わせて、教師データを与えて、それに近づける。
  - ただし、学習はA-R層間の結合のみ（これは当時の計算機パワーの限界もあったと思う）。
  - 学習後、うまく判別できるぐらい出力層の値が分離できるようになれば、文字を認識できる。

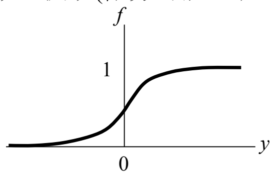
31

誤差逆伝搬法 (Back Propagation)

誤差逆伝搬法（１）

- パーセプトロン（３層構造のニューラルネットワーク）の誤り訂正学習は一層分の学習で、これを最後の層の結合荷重だけ学習するので、学習能力は低い。複雑なものに適用するために、
  - 中間層を増やす。
  - すべての層で学習する。➔　誤差伝搬法（誤差伝搬学習法）
  - パーセプトロンの拡張としてのニューラルネットワークの定義。
- 誤差伝搬法の説明の前に：
  - ニューロンの関数fはシグモイド関数を使う(微分可能とする)

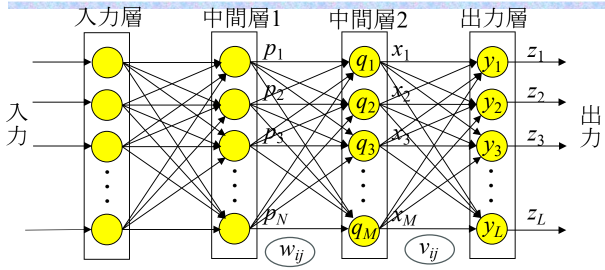
$$\varphi(y) = f(y) = \frac{1}{1 + \exp(-y)}$$



誤差逆伝搬法（２）

- 誤差 (逆)伝搬法 (back propagation)とは
  - 当初、パーセプトロンを発案。しかし、1969年、パーセプトロンの学習能力の限界が証明された。
  - Rumelhartにより提案（1986年ごろ）。
    - 文献：D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning internal representations by error propagation," D. Rumelhart and J. McClelland, editors. *Parallel Data Processing*, Vol.1, Chapter 8, the M.I.T. Press, Cambridge, MA 1986
  - ネットワークを多層化し、各ニューロンが多層で学習できるように拡張。
    - 考え方として、教師データは出力側で与えられるので、その値に合うようにニューロンを学習させ、それに基づいて次の層を学習、…..とこれを前方まで繰り返す。

誤差逆伝搬法の手順（１）



ニューラルネットの構造。中間層は、２層とは限らない。

w_{ij}: 中間層間の荷重
v_{ij}: 中間-出力層の荷重
※これらを学習する

- 入力とそれに対する教師データ $(d_1, \ldots, d_L)$ が与えられる。
- 手順0：階層型ニューラルネットを用意し、各層間の荷重を0に近い小さな数字で初期化する。
- 手順1：入力データを入力層に入力する。各層でニューロンの出力計算をする。

誤差逆伝搬法の手順（２）

- 出力計算の方法　（中間層2と出力層のみに着目）
  - 中間層2の場合：

$$q_j = \sum_{i=1}^{N} w_{ij} p_i - \theta, \qquad j = 1,2,\ldots,M$$
$$x_i = f(q_j), \qquad j = 1,2,\ldots,M$$

  - 出力層の場合：

$$y_k = \sum_{j=1}^{M} v_{jk} x_j - \theta, \qquad k = 1,2,\ldots,L$$
$$z_k = f(y_k), \qquad k = 1,2,\ldots,L$$

  - これらは、前に説明した通りの式に当てはめただけで、通常のデータの流れ。次のステップから実質的な誤差逆伝搬法。

誤差逆伝搬法の手順（３）

- 手順３：出力と教師データとの二乗和誤差(E)を計算する。Eを損失誤差とも呼ぶ。

$$E = \frac{1}{2}\sum_{k=1}^{L}(d_k - z_k)^2$$

- 手順４：Eを最小にするように、出力層にあるニューロンの結合荷重を調整（学習）する。

$$v_{jk}(t+1) = k v_{jk}(t) + \Delta v_{jk}$$
$$\Delta v_{jk} = \alpha \delta_k x_j$$

具体的にはEの値を小さくするよう v_{jk}の値を変える

$$\Delta v_{jk} = -\alpha\frac{\partial E}{\partial v_{jk}} = -\alpha\frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial y_k}\frac{\partial y_k}{\partial v_{jk}}$$
$$= \alpha(d_k - z_k) f'(y_k) x_j$$

誤差逆伝搬法の手順（４）

- 手順４（続き）：ここでfはシグモイド関数だったので、下記のようにδを定義して学習をする。

$$\delta_k = (d_k - z_k) f'(y_k)$$
$$= (d_k - z_k) f(y_k)\big(1 - f(y_k)\big)$$

なお、αは変更量（学習量）を表すパラメータであった。

- 以上でΔv_{jk}が求まり、出力層と中間層の重みv_{jk}の学習が完了する。

課題７－２：

fをシグモイド関数とすると、$f'(x) = f(x)\big(1 - f(x)\big)$となることを確かめよ。

誤差逆伝搬法の手順（５）

- 手順５：中間層間の学習。同じようにEを小さくするように、w_{ij}の値を修正する。ただし、途中に出力層があるので、合成関数として考えなくてはならない。

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} = -\alpha \sum_{k=1}^{L} \left(\frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial y_k} \frac{\partial y_k}{\partial x_j} \frac{\partial x_j}{\partial q_j} \frac{\partial q_j}{\partial w_{ij}} \right)$$
$$= \alpha \sum_{k=1}^{L} (d_k - z_k) f'(y_k) v_{jk} f'(q_j) p_i$$
$$= \alpha \sum_{k=1}^{L} \delta_k \, v_{jk} f'(q_j) p_i$$

誤差逆伝搬法の手順（６）

- 手順６：ここでのδ_j (区別のためδ_j'と書く) を求めると、

$$\delta_j' = \sum_{k=1}^{L} \delta_k v_{jk} f'(q_j) = \sum_{k=1}^{L} \delta_k v_{jk} f(q_j)(1 - f(q_j))$$

- これですべての値が決定する。上記でΔw_{ij}を確定でき、w_{ij}を更新できる。
- 手順７：
  - 手順５－６を入力層に達するまで繰り返す。
- 手順８：
  - 手順７までを各入力（教師データ）に対して繰り返す。

誤差逆伝搬法のまとめ。

- データの計算は入力から出力に進むが、結合荷重は出力側から入力側と逆向きに進むため、誤差逆伝播法と言う。
- パーセプトロンと違い、すべてのニューロン間の結合荷重が学習されるので、学習能力は高い。
- 学習時の計算量は大きいが、一度学習すれば、認識などは可能となる。
  - 問題の複雑さによるが多くのデータが必要（万単位）。
- 音声認識、文字認識、画像認識などの分野で効果が認められている。それ以外の知識を学習する分野では、必ずしも効果は十分ではない。
  - たとえば変化する環境では（不）正解データも変わるため、そのたびに学習の仕直しが必要（変化の度合いによるが）。

44 深層学習の概要

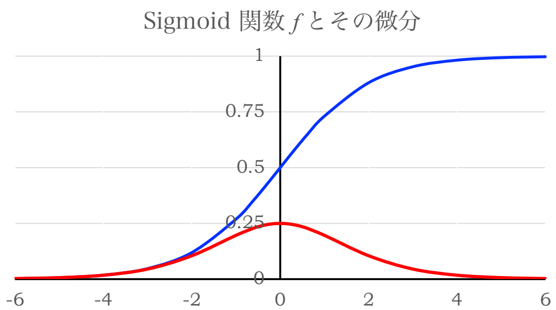
ニューラルネットワークの多層化

- 誤差逆伝搬法が使われたネットワークはしばらくは３～４層程度が多く使われていた。これでもそれなりの認識率があった。
  - 当然、計算機の能力が上がり、さらに多層化できる。ならば学習能力も上がるのではないかという期待があった (2000年前後)
- やってみたがうまくいかなかった。
  - 勾配喪失（と勾配爆発）
  - 過学習 (overfitting)
  - シグモイド関数の問題
  - 重みの初期値に大きく左右される

多層化への技術的課題

- 勾配喪失（と勾配爆発）
  - 誤差逆伝搬では誤差を重みで吸収するが、その誤差の勾配が伝搬にしたがい数層（３とか４とか）で消える。逆に調整して、それがある閾値を超えると勾配が拡大し、勾配が爆発的に増加する。これは微分したシグモイド関数の積が多段になるため。たとえば、「誤差逆伝搬法の手順（５）」の式を参照。
- 過学習--- overfitting
  - 現象としては、うまく学習したように見えても与えた教師データ以外で正解が出なくなる。（学習は教師データをベースにやや一般化し、少し違う状況でもうまく結果を出せることが望ましい。学習が過度となり、判定が厳格になる）。
- 重みの初期値に大きく左右される
  - 各重みの初期値によって結果が変動する。初期値を試行錯誤的に試す必要があるが、これは容易ではない。

Sigmoid関数とその微分



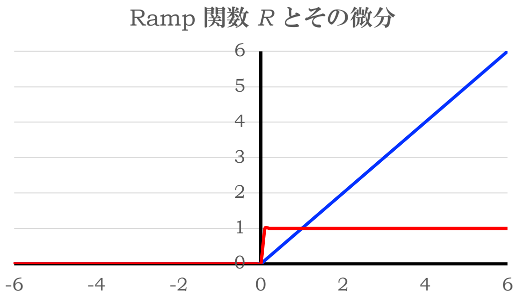
$$f(y) = \frac{1}{1 + \exp(-y)}$$

- 希望があり、グラフを書きました。
- f'は最大でも0.25, $|y|$が大きいとさらに0に近づく。この掛け算の繰り返しは0に近くなる。

課題に対するいくつかの対処

- ReLU (Rectified Linear Unit, Ramp function)の活用
  - シグモイド関数に変わり、ReLUを使う。ただし0付近で連続微分可能ではないが、x>0で微分して1と定数で、勾配喪失を防げる。（次のスライド）
  - 代わりにsoftmaxなども使われる。
- オートエンコーダー（自己符号化器）
  - 初期値を予め学習する。特に入力層に近い部分。
  - これにより勾配喪失を防ぐ効果がある。
  - これを初期値として、ネットワークの学習がうまく進む（ようだ）。

Rectified Linear Unitとその微分

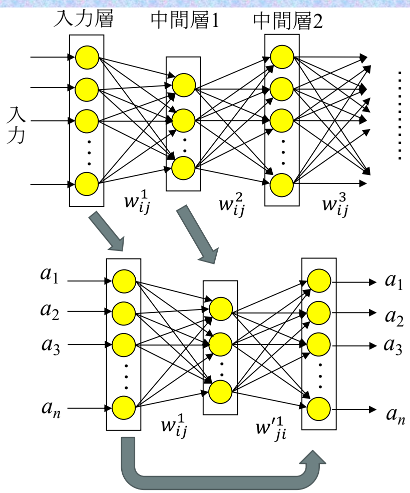


$R(y)$

- R'はyが正の時に1となる
- 活性化関数がReLUならシナプスの出力は0以上で、0のときは重みは効かない。

オートエンコーダ（１）

- 入力層から順にニューロンが減少し、情報が圧縮される状況（それには限らないが）。
- まず、最初の2層をとりだし、その鏡面の構造を作成し、3層のニューラルネットを作る。
- 教師データを入力し、出力も同じデータとなるように重みを誤差逆伝搬法で学習。
- その最初の2層間の重みw^1_{ij}とw'^1_{ij}を求める。 w^1_{ij}を保存。



オートエンコーダ（２）

- 第2と第3層のネットワークを取り出し、第2層の鏡面の関係で第4層を生成する。
  - 前のステップで求めたw^1_{ij}を利用し、教師データから第2層への入力を求め、その値が鏡面の第4層の出力と同じになるように、重みを学習する。
  - そのうち、第2層と第3層間の重みを保存。
- 第3と第4層のネットワークを取り出し、第3層の鏡面の関係で第5層を生成する。
  - 前のステップで求めたw^1_{ij}とw^2_{ij}を利用し、教師データから第3層への入力を求め、その値が鏡面の第5層の出力と同じになるように、重みを学習する。
  - そのうち、第3層と第4層間の重みを保存。
  - 以上を後方へ繰り返す。

オートエンコーダ（３）

- オートエンコーダは勾配喪失と重みの初期値の問題を解決できる
  - しかし現状のニューラルネットワークでは他の工夫もあり、必ず使われているわけではない。
  - 他の工夫として、ネットワーク構造を特殊化することで、オートエンコーダなしでも勾配喪失を防ぎ、学習結果も効果的にするものがある。
    (cf. たたみ込みニューラルネットワーク、convolutional neural network, CNN。画像処理でよく使われる)
  - たとえばCNNは、重みを意図的に定義して、画像の特徴量を抽出している。

53 まとめ

ニューラルネットワークの利点

- 非線形の特徴抽出にも対応可能
- ノイズや異常値に強いと言われる（相対的に多数の学習データも必要）
- 例のみだけで機能する。
- 学習に必要な計算量は莫大（欠点）だが、一度学習が済めば高い学習能力の結果を使える。
- 学習能力は分野に依存（画像認識、音声認識、自然言語処理、対話など）

ニューラルネットワーク共通の課題

- ニューラルネットワークは多層化により学習能力があがり多くの応用例が考えられる。ただ現状としては、いくつかの共通の問題がある。
  - 過学習：与えた教師データに合わせすぎ、類似データでも思った通りの結果がでないことがある。これが頻繁に起こる。
  - 逆にレイヤーを減らすとうまく学習できないこともがある。
  - パラメータが多過ぎ。アーキテクチャにも工夫が必要。初期値、何層にするか、ノード間の繋がり構成、CNNであればたたみ込み層とプーリング層の数など多数のパラメータがあり、これらは試行錯誤で、よいものを選択する。これには多大な計算資源と時間が必要。
  - 仮に学習できたとしてもその理由の解析などは不明。このため、ときどき間違うことがあっても、それが何故、また何時起こるかも予測がつかない。
  - これらに対する研究はあるが、まだこれからの課題。

パッケージ

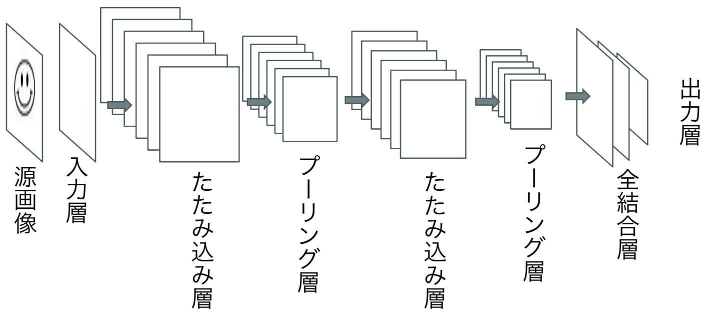
- たくさんのNeural network (deep learning) 用のパッケージが用意されている。たとえば、
  - Chainer (https://chainer.org/), Preferred networks社（終了）
  - Tensor flow (https://www.tensorflow.org/), Google社
  - Caffe (http://caffe.berkeleyvision.org/), UCB
  - PyTorch (https://pytorch.org/) Facebook社
  - ほかにもたくさんある。

  - ただし、計算処理が重いので通常のPCでは遅く、高性能でメモリをたくさん載せたGPUボードを搭載したものが必要となる。一枚十数万から百万円を超えるものまで。これらが複数枚必要。

57 少しだけ発展内容（導入のみ）

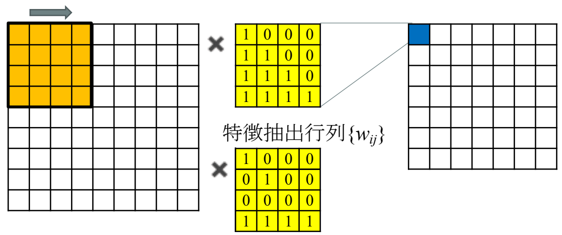
Convolutional neural networkの概要

- たたみ込み層 (convolutional layer)と情報圧縮 (プーリング層, pooling layer)を交互に何段か組み合わせ、最後に出力のために層間のノードを全結合させた層を入れる。



Convolutional neural networkの概要

- たたみ込み層

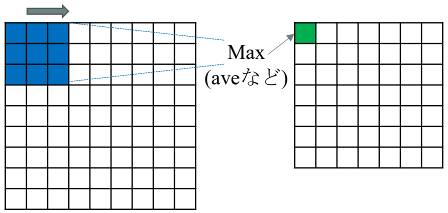


- 元の画像の成分をa_{ij}とすると、$\sum a_{ij}w_{ij} - \theta$の値が次のノードの入力となる。
- 活性化関数は ReLU

- 元画像のRGB成分の値を分けて、さらに特徴抽出で1の部分を取り出し、その和を次の層の対応部分に書く。特徴抽出行列は複数ある。
- これを1から数マス横に移動して同じ計算を繰り返す。
- 端は特徴抽出の機会が減るので、周辺にパディング(0, 黒) を置く。

Convolutional neural networkの概要

- プーリング層
  - 分割した領域の情報圧縮（最大値、平均値など）。
  - ここでもReLUを使う。ある閾値θ以下の値は0になり、情報の簡略化が行われる。



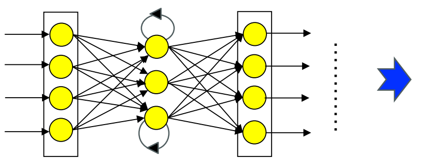
- たたみ込みとプーリングを数回繰り返し、最後の全結合層で教師データの学習を行い、その重みを調整する。

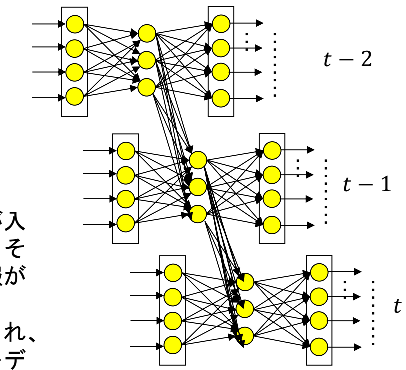
Convolutional neural networkの概要

- これまで述べてきたCNNは、基本の形であり、加えて工夫が多数提案されている。
- 過学習を防ぐためにドロップアウト層（全結合層の後半で、一部のつながりを切る）を設けることもある。
- 一般に何層が良いかは問題ごとに異なる
  - たたみ込み層やプーリング層を何回繰り返すか、全結合層の構成などは試行錯誤。何度も実験を繰り返し、適切と思われるニューラルネットワークを構成する。
  - このような試行錯誤はニューラルネットワークではよく行われること。

その他のネットワーク (1)

- リカレントニューラルネットワーク(RNN)
  - フィードバック（ループ）がある





- **概念的には一つ前の中間層のデータが入力の一つとしてニューロンに入るが、それはさらに前のニューロンの活性情報が入る。**
- **例えば文章。単語が時間と共に入力され、その前の語彙の影響を受けることをモデル化できる。**

その他のネットワーク (2)

- ホップフィールドネットワーク (Hopfield network)
  - ノードが双方向につながり、重みが左右対称 ($w_{ij}=w_{ji}$) のネットワーク。ある一つのノードだけが確率的にアクティブになり出力を更新、その変化が他のノードの入力に影響を与える。学習はしない。
- ボルツマンマシン (Boltzmann machine)
  - 入力層と中間層がホップフィールドネットワークのように相互に結合し、学習する。
  - 上記の２つは画像処理の講義で（たぶん）。

Deep Q-Network (DQN)

- Q学習は、Q関数$Q(s,a)$で、エージェントが認知した状態sにおける行動aの価値を表す。

$$a^* = arg \max_a Q(s,a)$$

となるa^*を行動として選択すればよい。

  - 実応用問題では、sの種類は多く膨大なメモリが必要なこともある。
- ならQ関数をニューラルネットワークで実現しよう。
  - 状態のパラメータ値を入力として適切な行動（または行動と価値の組）を出力したり、状態と行動を入力し、その価値を出力する。
  - 状態のパラメータはそれを表すいくつかの値の組としてもよいが、多くのDQNでは、入力をその場面を記録した画像（ときどき前フレームも同時に）をそのまま与え、CNNを通し、出力を行動と価値とすることが多い。

Deep Q-Network (DQN)

- 強化学習の「Q学習のアルゴリズム」のスライド。

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot (r(s,a) + \gamma \max_{a' \in A} Q(s',a'))$$

変形すると

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot (r(s,a) + \gamma \max_{a' \in A} Q(s',a') - Q(s,a))$$

- 赤の部分が差分（誤差）と考えられる部分。ニューラルネットワークの手法に真似てこの自乗誤差を求める。

$$E(w) = \frac{1}{2} \mathbb{E}\left[\left(r(s,a) + \gamma \max_{a' \in A} Q(s',a',w') - Q(s,a,w)\right)^2\right]$$

ここで$\mathbb{E}$は期待値とする。そもそもQ値は（報酬の）期待値を想定しているが、ここで$r(s, a)$は実際の値なので、一応$\mathbb{E}$をつけた。wはニューラルネットワークの重みで、Q関数は重みに依存するので明示した。w'は、一つ前の重み。

- 上の式の青の部分はニューラルネットワークの教師データに相当する。したがって、この部分は定数として考える。

Deep Q-Network (DQN)

- この式をwで微分

$$\frac{\partial E(w)}{\partial w} = \mathbb{E}\left[\left(r(s,a) + \gamma \max_{a' \in A} Q(s',a',w') - Q(s,a,w)\right)\frac{\partial Q(s,a,w)}{\partial w}\right]$$

- あとは$\frac{\partial E(w)}{\partial w}$に基づいて勾配降下法で修正する（誤差逆伝搬法）。

Deep Q-Networkの問題点 (1)

- 安定性と収束の問題（１）
  - Q値が安定しない（Q学習の収束性の定理はもはや成り立たない）
  - Q学習で使っていた列は、一連の行動と状態、報酬の列である。他方マルコフ決定過程の仮定で、直前の状態のみに依存するはず。
    - 列のまま与えると、その時系列順に関連性（相関性）があるように学習する。
    - そもそもニューラルネットワークでは、教師データのごとに順に重みを学習するので、その順序には多少なりとも依存する。
    - 実際の観測データは、世の中の起こりやすいものを多く観測する。その結果、あまり起こらない部分については学習が進まず、逆に頻繁に発生する状態については過学習が起こりうる。

Deep Q-Networkの問題点 (2)

- 安定性と収束の問題（２）
  - 仮にQ値が収束してもmaxを含む項は問題となる。
    - w'は直前の状態である。せっかく行動を選択し教師学習を得てもQ値のわずかな変更で選択される行動が変わる。その結果、次のステップのwでは、もはや異なる行動が適切となり、以前の教師データの正確性も疑問となる。
    - これは小さなQ値の変更が振動をまなくという大きな不安定要因となる。
  - 問題や状況によっては報酬のレンジが異なる。あるときは、[0, 100]また、[-1, 1]など。

Deep Q-Networkの問題点への工夫

- 対処法
  - Reward clipping:
    - 報酬のレンジが違うものについては、標準化（たとえば、[-1, 1]にマップする）が利用される。これを<u>報酬のクリッピング</u>と言う。
  - Experience replay:
    - Q値を使った実際のデータの列は、
      $(s_0, a_0, r_0) \Rightarrow (s_1, a_1, r_1) \Rightarrow (s_2, a_2, r_2) \Rightarrow (s_3, a_3, r_3) \Rightarrow \ldots \Rightarrow (s_n, a_n, r_n)$
      これを学習に直ぐには使わずにある程度記憶する。
    - これを分割して切り出し、ランダムに選択する。たとえば、
      $(s_i, a_i, r_i) \Rightarrow s_{i+1}$ (これをよく(s_i, a_i, r_i, s_{i+1})と表す)を選択し、これを学習する。データ間の独立性を強くする（相関性を消す）。
    - Prioritized experience replay：「重要な場面を学習する」ことの他に、前出の問題で、よく遭遇する頻度の高いものだけが多く観測されて記憶されることを防ぐために、適切な順位をつける。Priorityの付け方についても多くの工夫がある。たとえば、これまであまり観測されていないもの優先するなど。

Deep Q-Networkの問題点への工夫

- 対処法（続き）
  - Fixed target Q-network (or Freezing target Q-network):
    - Q値（つまりその値を出力したネットワーク）を固定する。
      $$\frac{\partial E(w)}{\partial w} = \mathbb{E}\left[\left(r(s,a) + \gamma \max_{a' \in A} Q(s',a',w') - Q(s,a,w)\right)\frac{\partial Q(s,a,w)}{\partial w}\right]$$
      のmaxの項を計算するためのネットワーク (つまりはw') をしばらく固定して勾配降下法を適用する。
    - maxの項が揺らぎの一因なので、安定性に貢献する。
    - これは相関性を低減する効果もある。

  - 参考： Atariのゲームを使ったDQNの一連の研究があるので、検索して調べてみるとよい。