インタラクティブシステム論
第11回

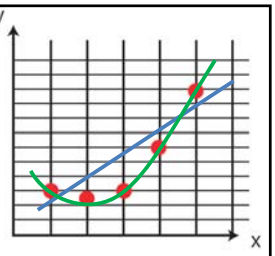
梶本裕之
Twitter ID kajimoto
ハッシュタグ #ninshiki

日程

4/13 第1回　イントロダクション
4/20 第2回　フーリエ変換
4/27 第3回フーリエ変換と線形システム
5/4　みどりの日
5/11　出張により休講
5/18 第4回　信号処理の基礎
5/25 第5回　信号処理応用1（相関）
6/1 第6回　信号処理応用2（画像処理）
6/08　インタラクティブシステムの実際（小泉先生）
6/15 第7回　ラプラス変換
6/22 第8回　古典制御の基礎
6/29　中間確認テスト（出張予定）
7/6 第9回　行列
7/13 第10回　行列と最小二乗法
7/20 第11回　インタラクティブシステムと機械学習
7/27 第12回　ロボティクス
8/3　期末テスト（出張中）

前回のレポート課題(1)

次のデータ系列に対して，
Scilabを用いて，
(1) 直線による近似，
(2) 2次曲線による近似を適用，
パラメータを求め，
曲線とデータをグラフに描け



x	1.0	2.0	3.0	4.0	5.0
y	3.0	2.5	3.0	6.0	10.0

なおScilabでは行列Aの擬似逆行列はpinv(A)で直接求めることができる．
当然自分でinv(A'*A)*A'とやっても同じ．

3

(1-1) 直線による近似

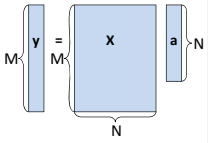
$$y = a_1 x + a_2$$

これは

$$y = a_1 x_1 + a_2 x_2 \quad \text{where} \quad x_2 = 1$$

とみなせる．

$$\begin{aligned} y_1 &= a_1 x_{11} + a_2 1 \\ y_2 &= a_1 x_{21} + a_2 1 \\ &\vdots \\ y_M &= a_1 x_{M1} + a_2 1 \end{aligned} \qquad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} x_{11} & 1 \\ x_{21} & 1 \\ \vdots & \vdots \\ x_{M1} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



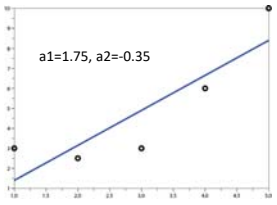
よって，

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る．

(1-1) 直線による近似

$$\begin{bmatrix} 3.0 \\ 2.5 \\ 3.0 \\ 6.0 \\ 10.0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



Scilabコード例

```
x=[1;2;3;4;5];
y=[3.0; 2.5; 3.0; 6.0; 10.0];
plot2d(x,y,style=[-9] );//元のデータをプロット

X=[1,1;2,1;3,1;4,1;5,1];        //擬似逆行列のための行列
a=inv(X'*X)*X'*y                //係数を最小二乗法で求める.
y_fitting = a(1) * x + a(2);    //フィッティング結果の直線
plot(x,y_fitting);              //表示
```

(1-2) 2次曲線による近似

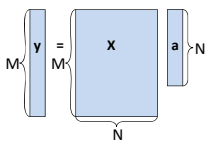
$$y = a_1 x^2 + a_2 x + a_3$$

これは

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3$$

とみなせる．

$$\begin{aligned} y_1 &= a_1 x_{11} + a_2 x_{12} + a_3 1 \\ y_2 &= a_1 x_{21} + a_2 x_{22} + a_3 1 \\ &\vdots \\ y_M &= a_1 x_{M1} + a_2 x_{M2} + a_3 1 \end{aligned}$$



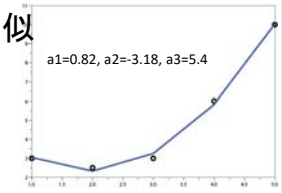
よって，

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る．

(1-2) 2次曲線による近似

$$\begin{bmatrix} 3.0 \\ 2.5 \\ 3.0 \\ 6.0 \\ 10.0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



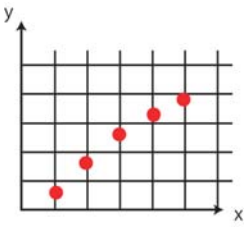
Scilabコード例

```
x=[1;2;3;4;5];
y=[3.0; 2.5; 3.0; 6.0; 10.0];
plot2d(x,y,style=[-9] );                    //元のデータをプロット

X=[1,1,1;4,2,1;9,3,1;16,4,1;25,5,1];   //擬似逆行列のための行列
a=inv(X'*X)*X'*y                            //係数を最小二乗法で求める.
y_fitting = a(1).*x .*x + a(2)*x +a(3); //フィッティング結果を示す
plot(x,y_fitting);                          //表示
```

前回のレポート課題(2)

次のデータ系列に対して，
y=a1 * log(x) + a2
を仮定してパラメータを求め，
曲線とデータをグラフに描け
（やや難？）



x	1.0	2.0	3.0	4.0	5.0	
y	0.5	1.9	2.7	3.3	3.7	

8

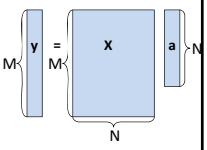
(2) logによる近似

$$y = a_1 \log(x+1) + a_2$$

これは

$$y = a_1 x_1 + a_2 x_2 \quad \text{where} \quad x_2 = 1$$

とみなせる.



$$\begin{aligned} y_1 &= a_1 \log(x_{11}+1) + a_2 1 \\ y_2 &= a_1 \log(x_{21}+1) + a_2 1 \\ &\vdots \\ y_M &= a_1 \log(x_{M1}+1) + a_2 1 \end{aligned} \qquad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} \log(x_{11}+1) & 1 \\ \log(x_{21}+1) & 1 \\ \vdots & \vdots \\ \log(x_{M1}+1) & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

よって，

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る.

レポートの成績評価とは関係なく，
これらの課題は，一度は打ち込んで
結果を見ることを勧めます.

最小二乗法は信号処理，制御，統計
などあらゆる分野で使われています.

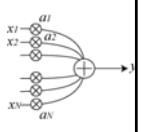
10

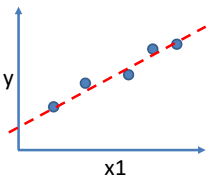
インタラクティブシステムにおける機械学習（のごく一部）

これまでの方法：線形回帰分析

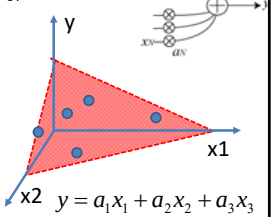
これまでの最小二乗法を用いた手法は，
N入力[$x_1, x_2, \ldots, x_N$]を変えた時に計測結果yが得られる時に，
yをxの「線形和」で表そうというものだった.

$$y = a_1 x_1 + a_2 x_2 + \cdots a_N x_N = \mathbf{a}^T \mathbf{x}$$





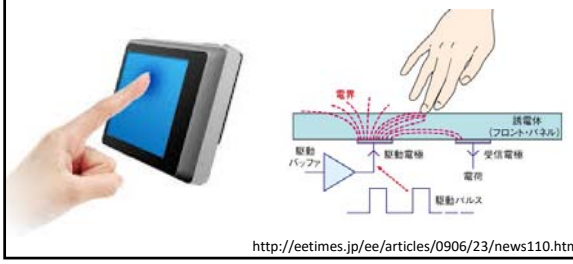
$y = a_1 x_1 + a_2 x_2$
$(x_2 = 1)$



$y = a_1 x_1 + a_2 x_2 + a_3 x_3$
$(x_3 = 1)$

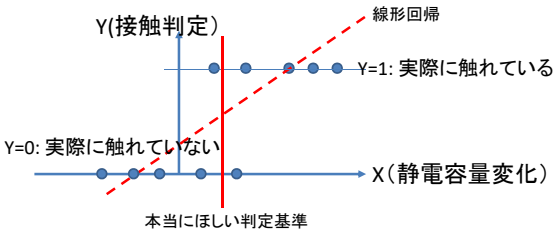
出力が連続な数値ではない場合

例えば出力が，ある「判定（カテゴリー判別）」の場合，yは連続量ではなくなる．
（例）タッチパネル（静電容量センサ）の値から，パネルに「触っている」「触っていない」を判断⇒y={0,1}



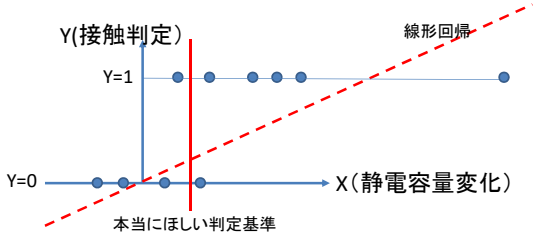
http://eetimes.jp/ee/articles/0906/23/news110.html

線形なフィッティングの不具合(1/2)



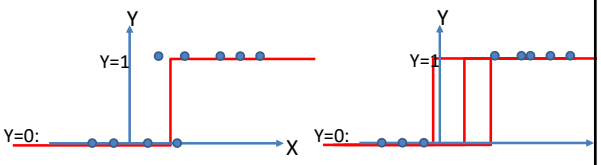
y-={0,1}に対して，y=ax+bで線形回帰させた場合，出力yは0,1の範囲に収まらず，意味がよくわからない．

線形なフィッティングの不具合(2/2)



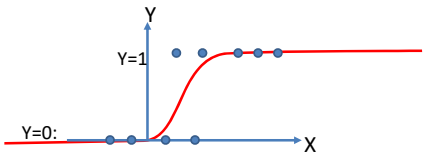
y-={0,1}に対して，y=ax+bで線形回帰させた場合，非常に大きな／小さな入力xに「引きずられる」現象が起きる．
これは直感的に正しい判定と言えない．

どのような関数でフィッティングすべきか



階段関数でフィッティング？
一見良さそうに見えるが，閾値付近のデータに大きく影響を受ける（閾値付近のデータ「だけ」で決まってしまうし，閾値付近にデータが無いときには不定になってしまう．

どのような関数でフィッティングすべきか



$h(x) = ax + b$ （線形回帰） $h(x) = \mathbf{a}^T\mathbf{x}$

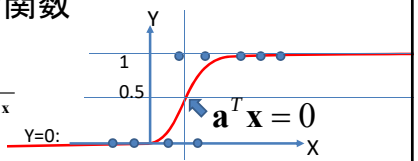
の代わりに，

$h(x) = \frac{1}{1+e^{-(ax+b)}}$（ロジスティック回帰） $h(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{a}^T\mathbf{x}}}$

を用いる．
(h: hypothesis function)

ロジスティック関数

$h(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{a}^T\mathbf{x}}}$



$\mathbf{a}^T\mathbf{x}$ が正の大きな値では：$h(x) =$

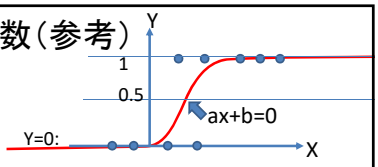
$\mathbf{a}^T\mathbf{x}$ が負の大きな値では：$h(x) =$

$\mathbf{a}^T\mathbf{x} = 0$では：$h(x) =$

最小値0，最大値1をとり$\mathbf{a}^T\mathbf{x} = 0$で1/2を取るため，
確率として解釈可能．
$\mathbf{a}$を求めるには最適化パッケージを用いる

ロジスティック関数（参考）

$$h(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{a}^T\mathbf{x}}}$$



フィッティングするために，最小化すべき「誤差」を考える

普通に考えると，
- Y=0の時：誤差はh(x_i)
- Y=1の時：誤差は1-h(x_i)

これをまとめて書くと，

$$\min\left[\sum_{i=0}^{N-1} -y_i h(x_i) - (1-y_i)(1-h(x_i))\right]$$

N:データ点数

ロジスティック関数（参考）

$$\min\left[\sum_{i=0}^{N-1} -y_i h(x_i) - (1-y_i)(1-h(x_i))\right]$$

しかしこの誤差だと一意の収束が難しいことが知られる．

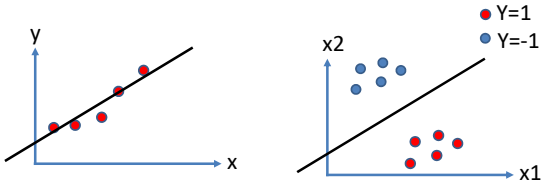
実際には，
- Y=0の時：誤差として-log(h(x_i))を用いる
- Y=1の時：誤差として-log(1-h(x_i))を用いる
- またパラメータa1,a2,….の大きさを抑える．

これらをまとめて書くと，

$$\min\left[\sum_{i=0}^{N-1} -y_i \log(h(x_i)) - (1-y_i)\log(1-h(x_i)) + \lambda\sum_{j=0}^{M-1} a_j^{\,2}\right]$$

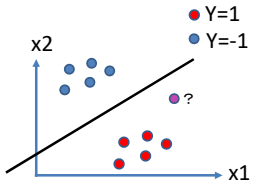
N:データ点数，M：説明変数の個数

回帰と分類（Regression & Classification）



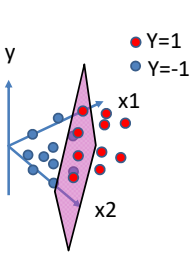
回帰：（主に連続値の）出力結果を予測するモデルを得る．
分類：（離散値の）領域を分割するモデルを得る．
線形回帰は回帰の代表．
ロジスティック回帰は，回帰によって分類するもの．

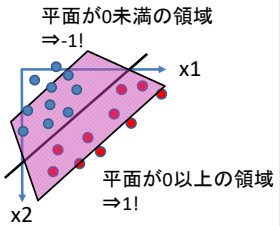
分類（Classification）



分類するとは：新たなデータ[x1,x2]がやってきた時に，それに対応するyが1なのか，-1なのかを判別できる識別器（＝判定式）を作ること．

平面で切る



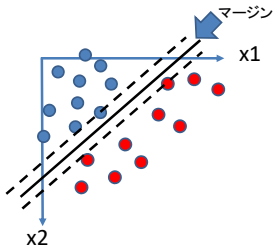


Y={0,1}の二つのクラスを分割する平面を探せば良い．

$y =$ $=$ (b:オフセット項)

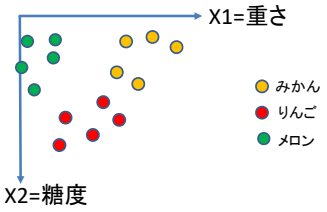
$$f(x_1, x_2) = sign(y) = \begin{cases} 1 \\ -1 \end{cases}$$

平面で切る



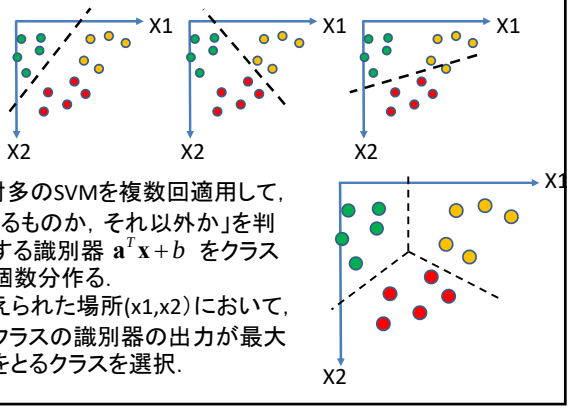
分割できる平面の中で，もっともマージンが大きいものを選ぶ．⇒（線形）SVM：State Vector Machineと呼ぶ

SVMによる複数クラスの分類



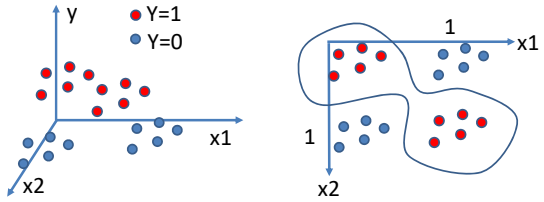
重さと糖度のセンサだけある状況？？で，みかんか，りんごか，メロンかを判別したい．

SVMによる複数クラスの分類



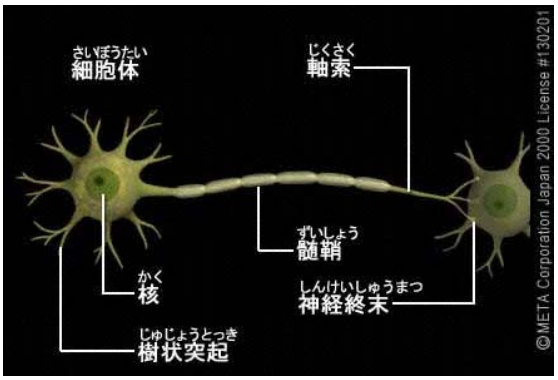
1対多のSVMを複数回適用して，「あるものか，それ以外か」を判定する識別器 $\mathbf{a}^T\mathbf{x}+b$ をクラスの個数分作る．
与えられた場所(x1,x2)において，各クラスの識別器の出力が最大値をとるクラスを選択．

容易に分類出来ない状況

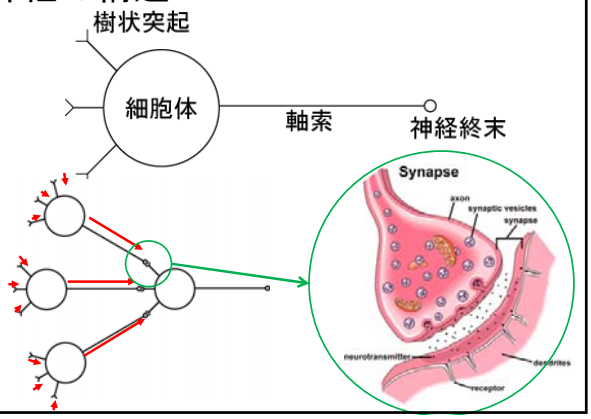


NOT(XOR)的な状況．明らかに二つのグループに分けられそうだが，平面によるフィッティングは明らかにうまくいかない．
次元を増やす（x3=x1*x2等の新しい軸を増やす）のも一つの方法だが．．．

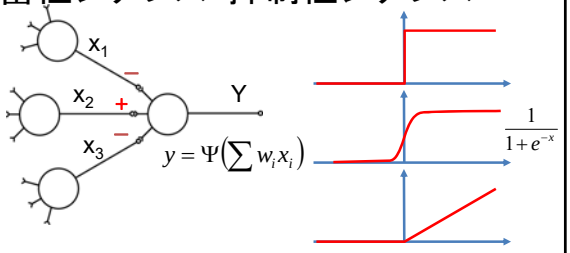
ニューラルネットワークへの第一歩：神経



神経の構造



興奮性シナプス・抑制性シナプス



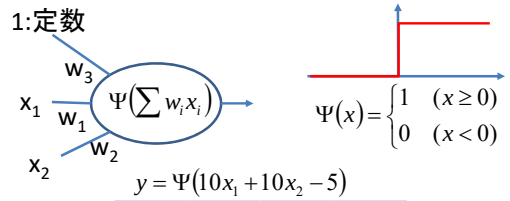
- シナプスには正負の重み付けがあり，積が計算され，
- 細胞体で和が取られ，
- 積和が閾値を超えると神経活動を生じる
- 人工ニューロンの場合，閾値の関数はステップ関数，シグモイド関数，ランプ関数などが用いられる

$w_i x_i$

$\sum w_i x_i$

$\Psi\left(\sum w_i x_i\right)$

ニューロンによるロジック回路の生成（1）



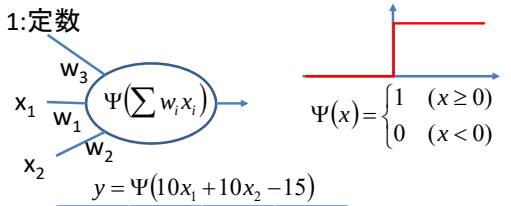
$y=\Psi(10x_1+10x_2-5)$

$\mathbf{w}=\begin{bmatrix}10\\10\\-5\end{bmatrix}$

x_1	x_2	Σ	y
0	0		
1	0		
0	1		
1	1		

ORが実現

ニューロンによるロジック回路の生成（2）



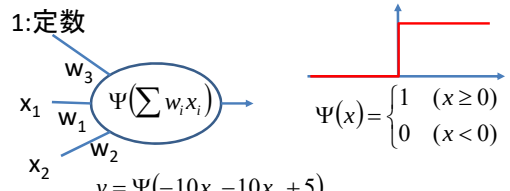
$y=\Psi(10x_1+10x_2-15)$

$\mathbf{w}=\begin{bmatrix}10\\10\\-15\end{bmatrix}$

x_1	x_2	Σ	y
0	0		
1	0		
0	1		
1	1		

ANDが実現
（積和でANDが実現するのは閾値の関数を用いているため）

ニューロンによるロジック回路の生成（3）



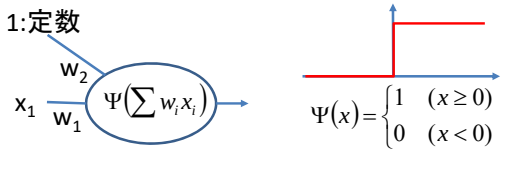
$y=\Psi(-10x_1-10x_2+5)$

$\mathbf{w}=\begin{bmatrix}-10\\-10\\5\end{bmatrix}$

x_1	x_2	Σ	y
0	0		
1	0		
0	1		
1	1		

NOR
(NOT(A) AND NOT(B))
が実現

ニューロンによるロジック回路の生成（4）

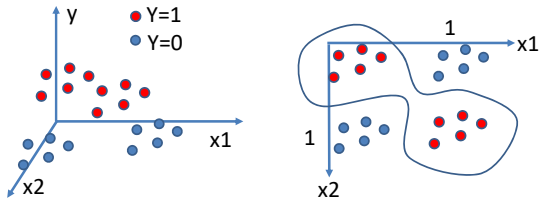


$\mathbf{w}=\begin{bmatrix}-10\\5\end{bmatrix}$

x_1	Σ	y
0	5	1
1	-5	0

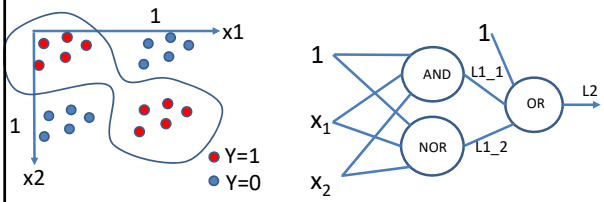
NOTが実現

簡単な平面で分割出来ない状況（再）



NOT(XOR)的な状況．明らかに二つのグループに分けられそうだが，平面によるフィッティングは明らかにうまくいかない．
次元を増やす（x3=x1*x2等の新しい軸を増やす）のも一つの方法だが．．．

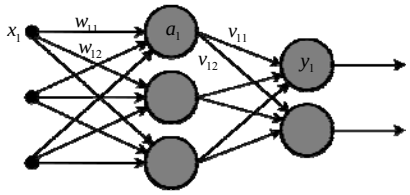
問題をロジック回路の組み合わせで解く



x_1	x_2	L1_1	L1_2	L2
0	0			
1	0			
0	1			
1	1			

ニューロンを「層」にすることで複雑な分類が可能となった．

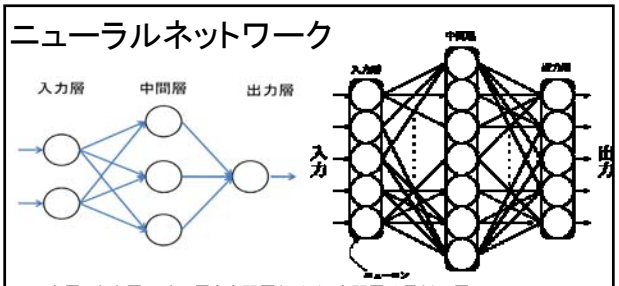
ニューラルネットワーク



ニューロンを多層化することで複雑な分類を可能とするもの.

- 入力: x_1, x_2, x_3
- 第1層重み付け: $w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}, w_{31}, w_{32}, w_{33}$
- 第1層出力: a_1, a_2, a_3
- 第2層重み付け: $v_{11}, v_{12}, v_{21}, v_{22}, v_{31}, v_{32}$
- 最終出力: y_1, y_2

ニューラルネットワーク



入力層，出力層以外の層を中間層と呼ぶ．中間層は最低一層．

出力1つの場合：何らかの判定．

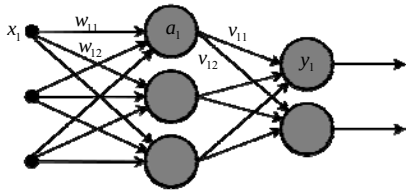
その写真は悪性腫瘍かどうか等

出力複数の場合：何らかのカテゴリ判定．

その写真はりんごか，みかんか，メロンか．カテゴリの数だけ出力あり

（例えば文字認識だと文字の種類だけの出力）

やっていること：2巡（以上）の「内積」



第一層の計算：

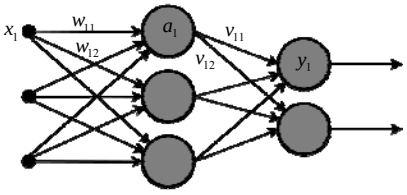
$$a^1_1 = \Psi(w_{11}x_1 + w_{21}x_2 + w_{31}x_3)$$

$$= \Psi\left(\quad\right)$$

$$= \Psi\left(\quad\right)$$

$\Psi(\ )$ はステップ関数，シグモイド関数，ランプ関数など

やっていること：2巡（以上）の「内積」



第一層の計算：$\Psi(\ )$ を無視すれば

（完全に無視するとニューラルネットワークの機能は大幅に無くなるがここでは説明のため）

$$a_1 = \mathbf{w}_1{}^t\mathbf{x} \quad a_2 = \mathbf{w}_2{}^t\mathbf{x} \quad a_3 = \mathbf{w}_3{}^t\mathbf{x}$$

つまり第一層では入力ベクトル**x**に対して，3つのベクトル**w1,w2,w3**の内積を計算

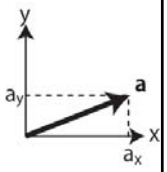
第二層でももう一度内積を計算している．

内積とは何だったか(復習)

ベクトル **a**=[a_x, a_y] のx成分は？. . . . . a_x

これはベクトル**a**とベクトル **x**=[1,0]との内積である.

a・**x** = [a_x, a_y]・[1,0] = a_x

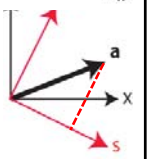


回転した座標軸，s, tを考える.

ベクトル **a**=[a_x, a_y] の，s 成分は？

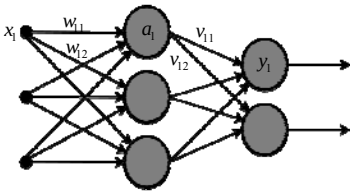
これはベクトル**a**とベクトル **s**=[s_x, s_y]との内積である.

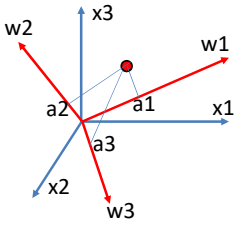
a・**s** = [a_x, a_y]・[s_x, s_y]



内積は，あるベクトルが別のベクトルの成分をどれだけ持つかを表す

1回目の内積：特徴表現



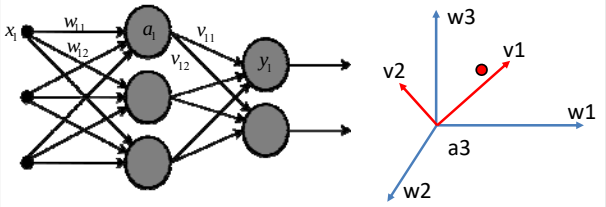


内積の意味から考えて，第一層の内積は，

入力ベクトル**x**の，**w1**ベクトル成分，**w2**ベクトル成分，**w3**ベクトル成分（射影）を計算している．それぞれの特徴の成分をどれだけ持っているか．

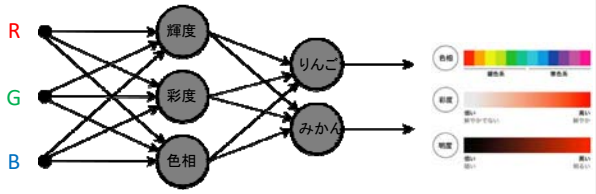
この特徴ベクトルの個数は入力の次元と同じとは限らない

2回目の内積：類似度による識別



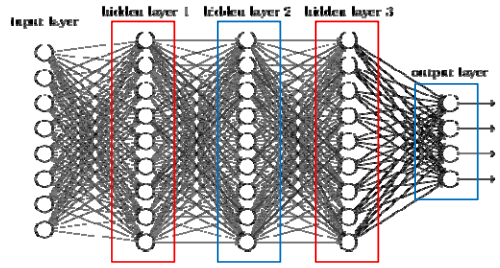
第2層の内積は，1層で得られた「特徴」のスコアをならべたベクトルが，
実際の平均的な事物（りんご，みかん，メロン）とどれだけ似ているかを内積で計算．

例：色による識別による類推



RGBから直接物体の種類を識別すると，撮影時の輝度に強く影響される．
しかし，輝度，色相に一度変換した後であれば，識別は容易になる．
（実際にはこの例ではニューラルネットワークは必要ありませんが，いったん特徴空間に変換するイメージ例として）

Deep Neural Network



特徴抽出レイヤと統合レイヤのペアを多段接続したものとして理解できる．

機械学習を使うために

以下の資料を参考にしました．お勧めします．

- Stanford大学の機械学習コース
  https://www.coursera.org/learn/machine-learning/home/welcome
- 内積を知っていれば分かる深層学習の非常に簡単な説明
  https://www.slideshare.net/SeiichiUchida/ss-71479583

Scilabにも今回紹介したロジスティック回帰，SVM，ニューラルネットワークのツールボックスがあります．試してみることを勧めます．
https://scilab.io/category/machine-learning/

今週はレポートなし