

Table of Contents

- 1 Breast Cancer Wisconsin (Diagnostic) Data Set
- 1.1 Attribute Information:
- 1.2 分類器
- 1.3 仮説クラス
- 2 最急降下法
- 3 code(python)
- 3.1 print_w
- 3.2 データの読み込みと初期化
- 3.3 最急降下法によるw探索(steepest descent)
- 4 結果
- 5 QR decomposition

9619

乳がん

Breast Cancer Detector

cc by Shigeto R. Nishitani 2018-19

Breast Cancer Wisconsin (Diagnostic) Data Set

[https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(diagnostic\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic))

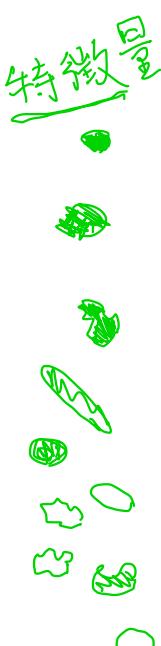
Attribute Information:

- 1. ID number
- 2. Diagnosis (M = malignant, B = benign) M:悪性, B:良性
- 3. 3-32

胞核

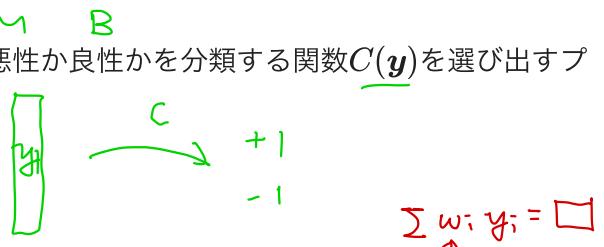
Ten real-valued features are computed for each cell nucleus:

- 半径 radius (mean of distances from center to points on the perimeter)
- テクスチャ texture (standard deviation of gray-scale values)
- 境界の長さ perimeter
- 面積 area
- なめらかさ smoothness (local variation in radius lengths)
- コンパクトさ compactness ($\text{perimeter}^2 / \text{area} - 1.0$)
- くぼみ度合い concavity (severity of concave portions of the contour)
- くぼみの数 concave points (number of concave portions of the contour)
- 対称性 symmetry
- フラクタル次元 fractal dimension ("coastline approximation" - 1)



分類器

与えられた特徴ベクトル \mathbf{y} に対し、細胞組織が悪性か良性かを分類する関数 $C(\mathbf{y})$ を選び出すプログラムを作成しよう。



仮説クラス

分類器は可能な分類器の集合(仮説クラス)から選ばれる。この場合、仮説クラスとは特徴ベクトルの空間 \mathbb{R}^D から \mathbb{R} への線形関数 $h(\cdot)$ である。すると分類器は次のような関数として定義される。

$$C(\mathbf{y}) = \begin{cases} +1 & \text{when } h(\mathbf{y}) \geq 0 \\ -1 & \text{when } h(\mathbf{y}) < 0 \end{cases}$$

各線形関数 $h : \mathbb{R}^D \rightarrow \mathbb{R}$ に対して、次のような D ベクトル w が存在する。

$$h(\mathbf{y}) = \mathbf{w} \cdot \mathbf{y}$$

したがって、そのような線形関数を選ぶことは、結局 D ベクトル w を選ぶことに等しい。特に、wを選ぶことは、仮説クラス h を選ぶことと等価なので、wを仮説ベクトルと呼ぶ。

単に、ベクトルの掛け算で分類器はできそう。問題はどうやってこの仮説ベクトル w の各要素の値を決定するか？ですよね。

最急降下法

$$\underline{\underline{A}} \underline{\underline{w}} + b = ?$$

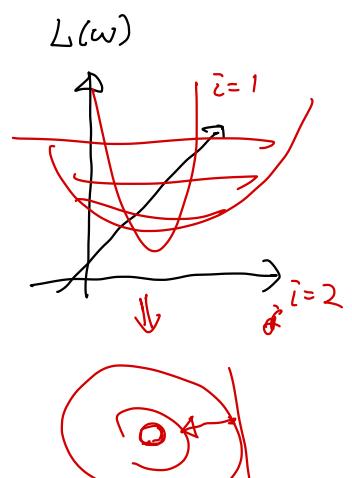
損失関数に

$$L(w) = \sum_{i=1}^n (A_i \cdot w - b_i)^2$$

を選ぶと、ベクトル w の j 偏微分は、

$$\begin{aligned} \frac{\partial L}{\partial w_j} &= \sum_{i=1}^n \frac{\partial}{\partial w_j} (A_i \cdot w - b_i)^2 \\ &= \sum_{i=1}^n 2(A_i \cdot w - b_i) A_{ij} \end{aligned}$$

となる。ここで、 A_{ij} は A_i の j 番目の要素です。この偏微分 $\frac{\partial L}{\partial w_j}$ を w_j の 勾配(slope) として、L(w) の極小値(local minimum)を求める。



このような探索方法を最急降下法(steepest descent method)と呼ぶ。

code(python)

- /Users/bob/python/doing_math_with_python/numerical_calc/breast_cancer_detector/codes,

print_w

出てきた w の j 要素をきれいに表示する関数を用意しておきます。

```
In [1]: def print_w(w):
    params = ["radius", "texture", "perimeter", "area",
              "smoothness", "compactness", "concavity", "concave points",
              "symmetry", "fractal dimension"]
    print(" (params) : ", end="")
    print(" (mean) (stderr) (worst)")
    for i, param in enumerate(params):
        print("%18s: %s" % (param, end=""))
        for j in range(3):
            print("%13.9f" % w[i*3+j], end="")
    print()
```

データの読み込みと初期化

```
In [2]: import numpy as np  
tmp = np.fromfile('./codes/train_A.data', np.float64, -1, " ")  
A = tmp.reshape(300,30)  
tmp = np.fromfile('./codes/train_b.data', np.float64, -1, " ")  
b = tmp.reshape(300,1)  
w = np.zeros(30).reshape(30,1)  
for i in range(30):  
    w[i] = 0.0  
print(w.shape)
```

(30, 1)

In [22]: b[0:20]

```
Out[22]: array([[-1],  
                [ 1.], B  
                [-1.], M  
                [ 1.], B  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [ 1.],  
                [-1.], M  
                [ 1.],  
                [ 1.]])
```

最急降下法によるw探索(steepest descent)

```
In [16]: loop, sigma = 300, 3.0*10**(-9)
for i in range(loop):
    dLw = A.dot(w)-b
    w = w - (dLw.transpose().dot(A)).transpose()*sigma
```

```
print_w(w)
```

手書き注釈:

- w_i (誤り) と b (正解) の関係を示す図。
- 30 (正解数) と 3 (誤り数) の合計が 33 であることを示す。

```
(params) : (mean) (stderr) (worst)
radius: 0.000426997 0.000741817 0.002548876
texture: 0.001687946 0.000004707 0.000000127
perimeter: -0.000003968 -0.000002078 0.000008954
area: 0.000003595 0.000002569 0.000070324
smoothness: 0.000001139 -0.000881778 0.000000430
compactness: 0.000000441 0.000000723 0.000000267
concavity: 0.000001200 0.000000191 0.000411499
concave points: 0.000921972 0.002395138 -0.001932789
symmetry: 0.000005930 -0.000003750 -0.000008147
fractal dimension: -0.000002341 0.000011565 0.000003523
```

結果

```
In [19]: def show_accuracy(mA, vb, vw):  
    # M: 悪性(-1), B: 良性(1)
```

手書き注釈:

- 各条件のカウントを示す。正解: 274, 安全な誤り: 5, 危険な誤り: 21。

```
correct,safe_error,critical_error=0,0,0  
predict = mA.dot(vw)  
n = vb.size  
for i in range(n):  
    if predict[i]*vb[i]>0:  
        correct += 1  
    elif (predict[i]<0 and vb[i]>0):  
        safe_error += 1  
    elif (predict[i]>0 and vb[i]<0):  
        critical_error += 1  
print(" correct: %4d/%4d" % (correct,n))  
print(" safe error: %4d" % safe_error)  
print("critical error: %4d" % critical_error)
```

```
In [20]: show_accuracy(A, b, w)
```

手書き注釈:

- 各条件のカウントを示す。正解: 274/300, 安全な誤り: 5, 危険な誤り: 21。

```
correct: 274 / 300  
safe error: 5  
critical error: 21
```

```
In [21]: tmp = np.fromfile('./codes/validate_A.data', np.float64, -1, " ")
```

```
A = tmp.reshape(260,30)
```

```
tmp = np.fromfile('./codes/validate_b.data', np.float64, -1, " ")
```

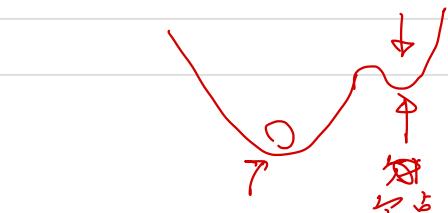
```
b = tmp.reshape(260,1)
```

```
show_accuracy(A, b, w)
```

手書き注釈:

- 各条件のカウントを示す。正解: 240/260, 安全な誤り: 10, 危険な誤り: 10。

```
correct: 240 / 260  
safe error: 10  
critical error: 10
```



QR decomposition

QR分解を使うとより簡単に最小値を求めることができる。行列 A は正方行列でないので、逆行列をもとめることができない。しかし、その場合でも $\|A \cdot w - b\|^2$ を最小にする w を求めることができる。

QR分解によって、 $n \times m$ 行列は

$$\underline{A} = \underline{Q} \underline{R}$$

と分解される。ここで、Qは $n \times m$ 行列、Rは $m \times m$ の正方行列。逆行列を求めることができる。

$\|Aw - b\|$ がzeroとなるのはQRを使って、

$$\begin{aligned} Q \cdot R \cdot w &= b \\ R \cdot w &= Q^t \cdot b \\ \boxed{R^{-1} \cdot R} \cdot w &= \underline{\underline{R^{-1} \cdot Q^t \cdot b}} \end{aligned}$$

となりそう。

$w =$

In [23]: `import numpy as np`

```
tmp = np.fromfile('./codes/train_A.data', np.float64, -1, " ")
A = tmp.reshape(300,30)
tmp = np.fromfile('./codes/train_b.data', np.float64, -1, " ")
b = tmp.reshape(300,1)

q, r = np.linalg.qr(A)
```

In [24]: `ww = np.linalg.inv(r).dot(np.transpose(q).dot(b))`

$R^{-1} \cdot (Q^T \cdot b)$

In [25]: `q.shape`

Out[25]: (300, 30)

In [26]: `print(r[0,0:5])`

```
[-2.57579883e+02 -3.32324268e+02 -1.68607899e+03 -1.29450676e+04
 -1.65446346e+00]
```

In [27]: `show_accuracy(A, b, ww)`

274 correct: 286 / 300
 5 → safe error: 1
 21 critical error: 13

In [28]: `print_w(ww)`

```
(params) : (mean) (stderr) (worst)
radius: 0.869921844 -0.024313948 -0.062679561
texture: -0.003274619 -8.790300861 1.747147500
perimeter: -0.202849407 -6.506451098 5.061760446
area: 49.167541566 -0.956591421 -0.082052658
smoothness: -0.007943157 0.004976908 -27.841944367
compactness: 3.301527110 4.985959134 -16.318886295
concavity: 10.316289081 -21.332232171 -0.408605816
concave points: -0.003345722 -0.000677873 0.002510735
symmetry: 4.531369718 0.590110016 -0.719368704
fractal dimension: -2.158965299 -3.803467225 -12.298417038
```

In [29]: `tmp = np.fromfile('./codes/validate_A.data', np.float64, -1, " ")`

A = tmp.reshape(260,30)

`tmp = np.fromfile('./codes/validate_b.data', np.float64, -1, " ")`

b = tmp.reshape(260,1)

`show_accuracy(A, b, ww)`

96%
4%

correct: 252 / 260
safe error: 6
critical error: 2

CNN ln []:
99%