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

1 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))

1.1 Attribute Information:

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

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 (perimeter^2 / 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)

http://people.idsia.ch/~juergen/deeplearningwinsMICCAIgrandchallenge.ht
(http://people.idsia.ch/~juergen/deeplearningwinsMICCAIgrandchallenge.h

1.2 分類器

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

1.3 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．

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

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$\boldsymbol{w}$が存在する．

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

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$\boldsymbol{w}$を 選ぶことに等しい．特に，$\boldsymbol{w}$を選ぶことは，仮説クラスhを 選ぶことと等価なので，$\boldsymbol{w}$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$\boldsymbol{w}$の各要素の値を決定するか？ですよね．

2 最急降下法

損失関数に

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

を選ぶと，ベクトル$\boldsymbol{w}$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 $\boldsymbol{w}_j$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．

3 code(python)

- /Users/bob/python/doing_math_with_python/numerical_calc/breast_c

3.1 print_w

出てきた$\boldsymbol{w}$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:" %param, end="")
        for j in range(3):
            print("%13.9f" % w[i*3+j], end="")
        print()
```

3.2 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
```

3.3 最急降下法によるw探索(steepest descent)

In [3]:

```
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)
```

```
   (params)      :       (mean)     (stderr)      (w
orst)
          radius:  0.000426997  0.000741817  0.0025
48876
         texture:  0.001687946  0.000004707  0.0000
00127
       perimeter: -0.000003968 -0.000002078  0.0000
08954
            area:  0.000003595  0.000002569  0.0000
70324
      smoothness:  0.000001139 -0.000881778  0.0000
00430
     compactness:  0.000000441  0.000000723  0.0000
00267
       concavity:  0.000001200  0.000000191  0.0004
11499
  concave points:  0.000921972  0.002395138 -0.0019
32789
        symmetry:  0.000005930 -0.000003750 -0.0000
08147
fractal dimension: -0.000002341  0.000011565  0.0000
03523
```

4 結果

In [4]:

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

    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 [5]:

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

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

In [6]:

```
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)
```

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

5 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$\|A.w-b\|^2$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

$$A = QR$$

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，

$$Q.R.w = b$$

$$R.w = Q^t.b$$

$$R^{-1}.R.w = R^{-1}.Q^t.b$$

となりそう．

In [7]:

```
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 [8]:
```
ww = np.linalg.inv(r).dot(np.transpose(q).dot(b))
```

In [9]:
```
q.shape
```

Out[9]: (300, 30)

In [10]:
```
print(r[0,0:5])
```

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

In [11]:
```
show_accuracy(A, b, ww)
```

```
       correct:  286/ 300
    safe error:    1
critical error:   13
```

In [12]:
```
print_w(ww)
```

```
    (params)      :       (mean)     (stderr)      (w
orst)
            radius:  0.869921844 -0.024313948 -0.0626
79561
           texture: -0.003274619 -8.790300861  1.7471
47500
         perimeter: -0.202849407 -6.506451098  5.0617
60446
              area: 49.167541566 -0.956591421 -0.0820
52658
        smoothness: -0.007943157  0.004976908-27.8419
44367
       compactness:  3.301527110  4.985959134-16.3188
86295
         concavity: 10.316289081-21.332232171 -0.4086
05816
    concave points: -0.003345722 -0.000677873  0.0025
10735
          symmetry:  4.531369718  0.590110016 -0.7193
68704
 fractal dimension: -2.158965299 -3.803467225-12.2984
17038
```


In [13]:
```
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)
```

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

In [ ]:
```

```