Table of Contents

Breast Cancer Wisconsin (Diagnostic) Data Set

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

Attribute Information:

1. ID number

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

   - M:悪性(-1)，B:良性(1) => M:悪性(1)，B:良性(-1)
   - cancer detectorの評価なので，positiveは悪性であるべき.

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)

のそれぞれのmean, stderr, worst数値を保持している.

http://people.idsia.ch/~juergen/deeplearningwinsMICCAIgrandchallenge.html

分類器

用意された訓練(training)データには，$\boldsymbol{A}$に上に記した特徴量が，$\boldsymbol{b}$に悪性(1, positive)か良性(-1, negative)かを 示す数値が入っている．

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．

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$h : \mathbb{R}^D \to \mathbb{R}$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．

問題を単純化すると，分類器を単なるベクトルとみなして，データとの掛け算で予測がつきます．本来は予測を-1,1とかに投影しないといけないんですが，単純化のためにそのままの値を用います．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．

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番目の要素を意味する． 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．

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):
    import numpy as np
    params = [
        "radius", "texture", "perimeter", "area",
        "smoothness", "compactness", "concavity", "concave_pts",
        "symmetry", "fractal_dim"
    ]
    w = np.array(w).flatten()
    print("{:<13} {:>13} {:>13} {:>13}".format("Parameter", "Mean", "StdErr"
    for i, param in enumerate(params):
        mean, stderr, worst = w[i*3], w[i*3+1], w[i*3+2]
        print("{:<13} {:13.9f} {:13.9f} {:13.9f}".format(param, mean, stderr
```

データの読み込みと初期化

In [2]:
```
import numpy as np
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/train_A.data', np.flo
A = tmp.reshape(300,30)
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/train_b.data', np.flo
b = tmp.reshape(300,1)
w = np.zeros(30).reshape(30,1)
for i in range(30):
    w[i] = 0
```

In [3]:
```
A[0]
```

Out[3]:
```
array([1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
       3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
       8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
       3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
       1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01])
```

In [4]:
```
b[15:25]
```

Out[4]:
```
array([[ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [-1.],
       [-1.],
       [-1.],
       [ 1.],
       [ 1.],
       [ 1.]])
```

最急降下法によるw探索(steepest descent)

In [5]:
```
loop, sigma = 300, 3.0*10**(-9)
for i in range(loop):
  dLw = A.dot(w)-b
  w = w - (A.transpose().dot(dLw))*sigma

print_w(w)
```

Parameter	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_pts	-0.000921972	-0.002395138	0.001932789
symmetry	-0.000005930	0.000003750	0.000008147
fractal_dim	0.000002341	-0.000011565	-0.000003523

結果

In [6]:
```
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): #  悪性なのに良性と予測：見落としcritica
            critical_error += 1
        elif (predict[i]>0 and vb[i]<0): # 良性なのに悪性と予測：再検査safe(fp)
            safe_error += 1
    print("     correct(true predicttion): %4d/%4d" % (correct,n))
    print("    safe error(false positive): %4d" % safe_error)
    print("critical error(false negative): %4d" % critical_error)
```

In [7]:
```
def show_accuracy2(mA, vb, vw):
    # M:悪性(1) positive, B:良性(-1) negative
    # true(真) , false(偽)
    tp, tn, fp, fn = 0, 0, 0, 0
    predict = mA.dot(vw)
    n = vb.size
    for i in range(n):
        if predict[i]*vb[i]>0:
            if predict[i]>0:
                tp += 1
            else:
                tn += 1
        elif (predict[i]<0 and vb[i]>0): #  悪性なのに良性と予測：見落としcritica
            fn += 1
        elif (predict[i]>0 and vb[i]<0): # 良性なのに悪性と予測：再検査safe(fp)
            fp += 1
    print("%21s %12s %10s" %  ("predict","positive", "negative"))
    print("%21s | %10d %10d" % ("actual positive",tp, fn))
    print("%21s | %10d %10d" % ("actual negative",fp, tn))
```

In [8]:
```
show_accuracy(A, b, w)
```

```
    correct(true predicttion):  274/ 300
   safe error(false positive):    5
critical error(false negative):   21
```

In [9]:
```
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/validate_A.data', np.
A = tmp.reshape(269,30)
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/validate_b.data', np.
b = tmp.reshape(269,1)

show_accuracy(A, b, w)
```

```
    correct(true predicttion):  248/ 269
   safe error(false positive):   10
critical error(false negative):   11
```

In [10]:
```
show_accuracy2(A, b, w)
```

```
                predict     positive   negative
     actual positive |          55         11
     actual negative |          10        193
```

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$m \times m$の正方行列で，逆行列を求めることができる．

$||Aw - b||$が最小となるのは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 [11]:
```
import numpy as np

tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/train_A.data', np.flo
A = tmp.reshape(300,30)
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/train_b.data', np.flo
b = tmp.reshape(300,1)

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

In [12]:
```
ww = np.linalg.inv(r).dot(np.transpose(q).dot(b))
```

In [13]:
```
q.shape
```

Out[13]:
```
(300, 30)
```

In [14]:
```
print(r[0,0:5])
```

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

In [15]:
```
show_accuracy(A, b, ww)
show_accuracy2(A, b, ww)
```

```
    correct(true predicttion):  286/ 300
   safe error(false positive):    1
critical error(false negative):   13
              predict     positive   negative
     actual positive |        133         13
     actual negative |          1        153
```

In [16]:
```
print_w(ww)
```

```
Parameter              Mean        StdErr         Worst
radius         -0.869922200   0.024313945   0.062679604
texture         0.003274620   8.790304633  -1.747147681
perimeter       0.202847615   6.506449469  -5.061760150
area          -49.167544936   0.956592244   0.082052701
smoothness      0.007943079  -0.004976910  27.841937812
compactness    -3.301518835  -4.985956723  16.318887300
concavity     -10.316290100  21.332168405   0.408605799
concave_pts     0.003345717   0.000677877  -0.002510735
symmetry       -4.531369160  -0.590110921   0.719368538
fractal_dim     2.158967280   3.803466937  12.298424597
```

In [17]:
```
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/validate_A.data', np.
A = tmp.reshape(269,30)
tmp = np.fromfile('./wisconsin_diag_breast_cancer_data/validate_b.data', np.
b = tmp.reshape(269,1)

show_accuracy(A, b, ww)
show_accuracy2(A, b, ww)
```

```
    correct(true predicttion):  261/ 269
   safe error(false positive):    6
critical error(false negative):    2
              predict     positive   negative
     actual positive |          64          2
     actual negative |           6        197
```

In [ ]:

In [ ]: