4.3 確率的識別モデル

平成 28 年 9 月 11 日

概 要

PRML の「4.3 確率的識別モデル」についての実装と考察.

目 次

1　問題設定

2 クラス分類問題では, 多くのクラスの条件付確率分布 $p(x|C_k)$ に対して, そのクラス C_1 の事後確率分布が X の線形関数のロジスティックシグモイド関数として書ける. 同様に多クラスの場合, クラス C_k の事後分布は x の線形結合のソフトマックス変換によって与えられる. この線形結合のパラメータを求める.

2　4.3.2 ロジスティック回帰

2.1　アルゴリズム

まず, ロジスティック回帰は分類のためのモデルであることに注意する.
2 クラスの場合を考える.

$$p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}) \quad (4.87)$$

ここで, $p(C_2|\boldsymbol{\phi}) = 1 - p(C_1|\boldsymbol{\phi})$ である. $\sigma(\cdot)$ はロジスティックシグモイド関数である.
ロジスティック回帰モデルのパラメータを最尤法を用いて決定する. このため (4.87) を微分する.
ロジスティックシグモイド関数の微分は

$$\frac{d\sigma}{da} = \sigma(1-\sigma) \quad (4.88)$$

となる.
データ集合は $\{\boldsymbol{\phi}_n, t_n\}, \boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}), t_n \in \{0,1\}$ であり, $n = 1, \ldots, N$ に対する尤度関数は

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n} \quad (4.89)$$

と書ける. 尤度の負の対数が誤差関数 (交差エントロピー誤差関数) で

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \quad (4.90)$$

$\mathbf{w}$ に対する誤差関数の勾配をとって

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n \quad (4.91)$$

が得られる.

3　4.3.3 反復債重み付け最小二乗

ニュートン-ラフソン法に基づく反復最適化手順を用いることで, $E(W)$ を最小化する W を求める.

$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - H^{-1}\nabla E(\mathbf{w}) \quad (4.92)$$

3.1 二乗和誤差

3.1.1 アルゴリズム

二乗和誤差を誤差関数とする線形回帰モデル (3.3) にニュートン-ラフソン法を適用する.

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^T\boldsymbol{\phi}_n - t_n)\boldsymbol{\phi}_n = \Phi^T\Phi\mathbf{w} - \Phi^T\boldsymbol{t} \quad (4.93)$$

$$H = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\boldsymbol{\phi}\boldsymbol{\phi}_n^T = \Phi^T\Phi \quad (4.94)$$

これより

$$\begin{aligned}\mathbf{w}^{(new)} &= \mathbf{w}^{(old)} - (\Phi^T\Phi)^{-1}\left\{\Phi^T\Phi\mathbf{w}^{(old)} - \Phi^T\mathbf{t}\right\} \\ &= (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t} \quad (4.95) \\ &= \Phi^{\dagger}\mathbf{t}\end{aligned}$$

ここで $A^{\dagger}$ は A の疑似逆行列.

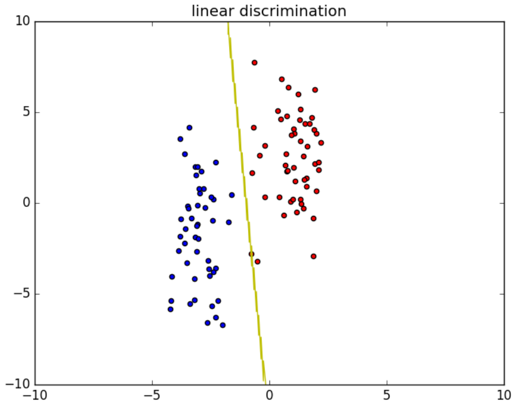
3.1.2 コード

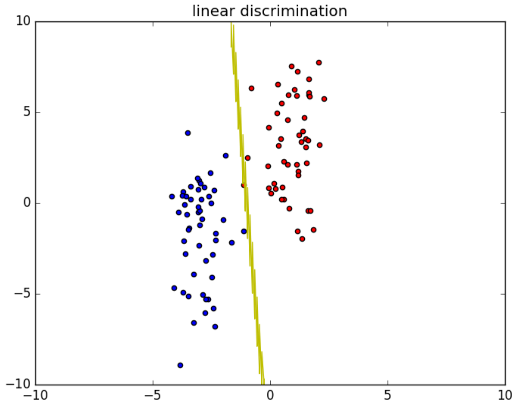
決定境界をプロットした.(2class1.py)

```
def sig(x):
  return 1/(1+np.exp(-x))
#MAP解を見つける(誤差関数が二乗和誤差関数のとき)
W=np.dot(pinv(x),t)

"""プロット"""
plt.title("linear␣discrimination")
plt.xlim([-10,10])
plt.ylim([-10,10])
#訓練データの散布図
for n in range(N):
        if t[n]==1:
                plt.scatter(x[n,1],x[n,2],c="red")
        else:
                plt.scatter(x[n,1],x[n,2],c="blue")
#決定境界のプロット
fx,fy=[],[]
p1,p2=sp.linspace(-10,10,200),sp.linspace(-10,10,200)
for i in range(200):
  for j in range(200):
    z=np.array([1,p1[i],p2[j]])
    if abs(sig(np.dot(W,z))-0.5)<10**(-2):
      fx.append(z[1])
      fy.append(z[2])
plt.plot(fx,fy,"y")

plt.savefig("1_2.png")
plt.show()
```

3.1.3　結果

左は$t \in \{0,1\}$として, $W^T\boldsymbol{\Phi} = 0.5$の決定境界を描いた．右は$t \in \{-1,1\}$として, $\sigma(W^T\boldsymbol{\Phi}) = 0.5$の決定境界を描いた．





3.2　交差エントロピー誤差

3.2.1　アルゴリズム

交差エントロピー誤差関数である

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\Phi_n \quad (4.91)$$

を用いると,

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n = \Phi^T(\mathbf{y} - \mathbf{t}) \quad (4.96)$$

$$H = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}\boldsymbol{\phi}_n^T = \Phi^T R\Phi \quad (4.97)$$

ここでRは

$$R_{nn} = y_n(1 - y_n) \quad (4.98)$$

の対角行列．そして

$$\begin{aligned}
\mathbf{w}^{(new)} &= \mathbf{w}^{(old)} - (\Phi^T R\Phi)^{-1}\Phi^T(\mathbf{y} - \mathbf{t}) \\
&= (\Phi^T R\Phi)^{-1}\left\{\Phi^T R\Phi\mathbf{w}^{(old)} - \Phi^T(\mathbf{y} - \mathbf{t})\right\} \\
&= (\Phi^T R\Phi)^{-1}\Phi^T R\mathbf{z} \quad (4.99)
\end{aligned}$$

ここで

$$\mathbf{z} = \Phi\mathbf{w}^{(old)} - R^{-1}(\mathbf{y} - \mathbf{t}) \quad (4.100)$$

この更新式を繰り返し収束したものを用いる．

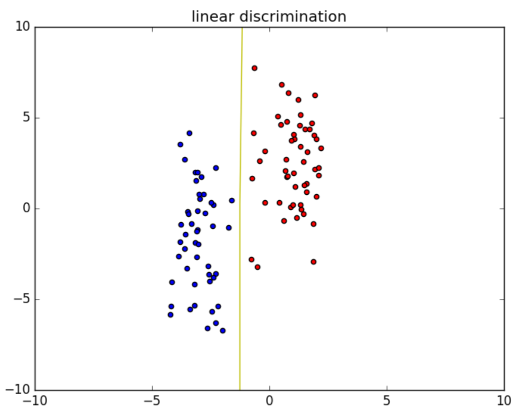
3.2.2 コード

$\mathbf{w}$ の更新式のみを記す.(2class2.py)

```
def sig(z):
  try:
    return 1/(1+np.exp(-z))
  except:
    return 0
#MAP解を見つける(誤差関数が交差エントロピー誤差関数のとき)
norm=1
W_new=np.zeros(3)
R=np.zeros((N,N))
i=0
while i!=10:#norm>10**(-3):
  W_old=W_new
  y=[sig(dot(W_old,x[n,:])) for n in range(N)]
  for n in range(N):
    R[n,n]=y[n]*(1-y[n])
  W_new=W_old-dot(inv(dot(x.T,dot(R,x))),dot(x.T,y-t))
  #z=dot(x,W_old)-dot(inv(R),y-t)
  #W_new=dot(inv(dot(x.T,dot(R,x))),dot(x.T,dot(R,z)))
  norm=np.linalg.norm(W_new-W_old)
  print(norm,R[0,0],W_old)
  i+=1
```

3.2.3 結果

左は $t \in \{0,1\}$ として, $\sigma(\mathbf{w}^T\mathbf{\Phi}) = 0.5$ の決定境界を描いた.



3.2.4 問題点

σ関数内のexp 関数が overflow を起こしてしまう. そのため W の更新を途中で切っている.

4 まとめ

確率的な結果が出る点ロジスティック回帰はいいと思ったが, ロジスティック回帰の結果が 0.8 と出た場合確率 0.8 と一致するのかは疑問である.