# 4.1.7 パーセプトロンアルゴリズム

平成 28 年 9 月 11 日

概 要

PRML の「4.1.7 パーセプトロンアルゴリズム」についての実装と考察

## 目 次

# 1　問題設定

パーセプトロンは 2 クラスの線形識別モデルである．線形分離可能なデータ集合をクラス $C_1, C_2$ に分離する．
このモデルには, 次の一般化線形モデルを用いる．

$$y(\mathbf{x}) = f(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \quad (4.52)$$

ここで, $\boldsymbol{\phi}(\mathbf{x})$ はバイアス成分 $\phi_0(\mathbf{x})$ を含み, 非線形活性化関数 $f(\cdot)$ はステップ関数

$$f(a) = \begin{cases} 1 & (a \geq 0) \\ -1 & (a < 0) \end{cases} \quad (4.53)$$

で与えられる．
クラス $C_1$ に属するデータ点に対する目標変数は $t = 1$, クラス $C_2$ に属するデータ点に対する目標変数は $t = -1$ と置いているのである．
目標はパラメータ $\mathbf{w}$ を決定することである．これには誤差関数の最小化を用いる．

# 2　アルゴリズム

誤差関数には以下を用いる

$$E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)t_n \quad (4.54)$$

ただし, $M$ は誤識別されたパターンの集合を表す．
この誤差関数の最小化には確率的最急降下法を用い, $x_n$ が誤識別されているときは

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\boldsymbol{\phi}(\mathbf{x}_n)t_n \quad (4.55)$$

で $\mathbf{w}$ を更新する．

# 3　コード

パーセプトロン (parseptron_1.py)

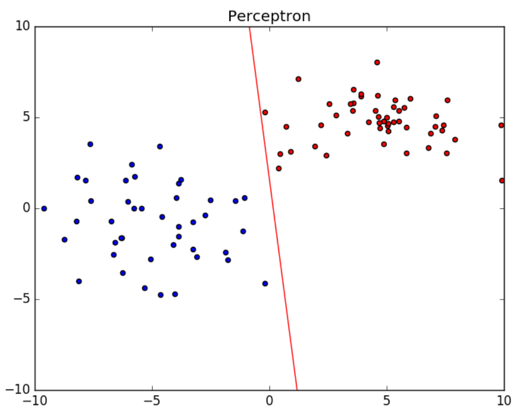
```
#gifアニメ関係
fig = plt.figure()
ims = []
p1 = np.arange(-10, 10, 0.4)
plt.xlim([-10,10])
plt.ylim([-10,10])

for n in range(N):
        if t[n]==1:
                plt.scatter(x1[n],x2[n],c="red")
        else:
                plt.scatter(x1[n],x2[n],c="blue")

"""Wの最適化"""
#Wの初期化
W=np.array([0,0,1])

#線形識別関数a=W,b=x
def model_f(a,b):
```

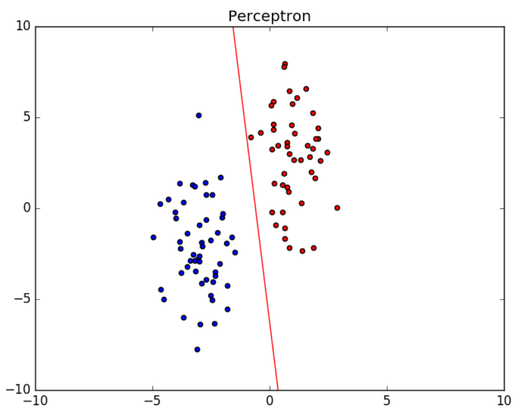
```
                return -(a[1]*b+a[0])/a[2]

for m in range(N*5):
        n=m%N
        #パラメータWの更新
        if np.dot(W,x[n][:])*t[n]<0:
                W = W + x[n][:]*t[n]

        #プロット
        p2 = model_f(W,p1)
        im = plt.plot(p1, p2, "r")
        #plt.title("n=%d" %n)
        #plt.savefig("%d.png" %n)
        """
        if m==N*5-1:
                p2 = model_f(W,p1)
                plt.plot(p1, p2, "r")
                plt.title("Perceptron")
                plt.savefig("percptron_1.png")
        """
        ims.append(im)

#gifアニメ関係
#"""
plt.title("Perceptron")
ani = animation.ArtistAnimation(fig, ims,blit=True,interval=N,repeat=False)
ani.save('perceptron_1.gif',fps=10,writer='imagemagick')
#"""
plt.show()
```

ここでは, 最終結果の png または過程の gif アニメーションを出力するコードが含まれ, それが大部分を含む.

## 4 結果

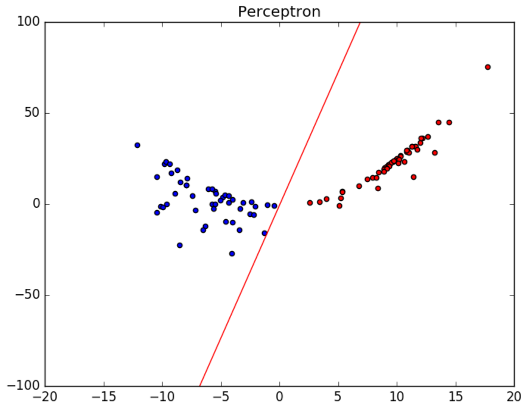
特徴ベクトルは $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x} = (1, x_1, x_2)$ とし, 二つの例で試した. 最終結果 (不適切であるが, $N \times 5$ 回のループで切った) は以下となった.

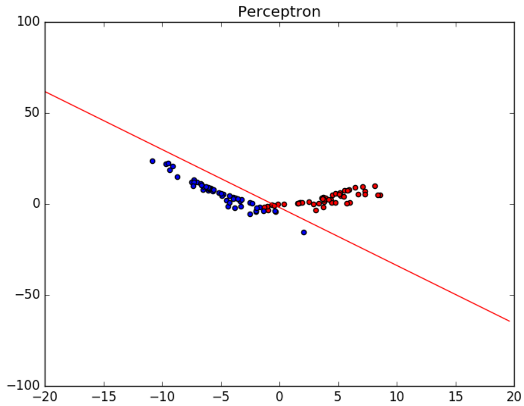



見ても分かるようどちらも線形分離可能なので, パーセプトロンで識別することができた.

他にも入力ベクトル X を変換し特徴ベクトル $\boldsymbol{\phi}(\mathbf{x}) = (1, x_1 + x_2, x_1 x_2)$ とすると




となった. 図 4 は線形識別に失敗しているが, これは特徴ベクトルのとり方がデータ点にあっていなかったためと思われる.
この結果より, 特徴ベクトルのとり方が重要であるように感じる.

## 5 まとめ

パーセプトロンは旋回分離可能なものにしかうまく働かない点と確率的でない点がいまいち.