パターン認識
2019-06-11 授業分 レポート

37-196360　森田涼介

2019 年 6 月 24 日

宿題 1

宿題 2

(1)

二つの超平面

$$\text{平面 1:}\quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - b = 1 \tag{1}$$
$$\text{平面 2:}\quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - b = -1 \tag{2}$$

の間のマージンが

$$\frac{2}{||\boldsymbol{w}||} \tag{3}$$

であることを示す。

それぞれ平面 1，平面 2 上にあり，かつその差分ベクトルがその二平面に垂直であるような 2 点 $\boldsymbol{x}_1$，$\boldsymbol{x}_2$ を考える。つまり，

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1 - b = 1 \tag{4}$$
$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_2 - b = -1 \tag{5}$$
$$\boldsymbol{m} = \boldsymbol{x}_1 - \boldsymbol{x}_2 = k\boldsymbol{w} \qquad (k \in \mathbb{R}_{\neq 0}) \tag{6}$$

これらより，

$$\begin{aligned} \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m} && (7)\\ &= \boldsymbol{w}^{\mathrm{T}} \cdot k\boldsymbol{w} && (8)\\ &= k||\boldsymbol{w}||^2 && (9)\\ &= 2 && (10) \end{aligned}$$

となる。よって，

$$k = \frac{2}{||\boldsymbol{w}||^2} \tag{11}$$
$$\boldsymbol{m} = k\boldsymbol{w} = \frac{2}{||\boldsymbol{w}||^2}\boldsymbol{w} \tag{12}$$

となるから，結局，二平面の間のマージン $||\boldsymbol{m}||$ は，

$$||\boldsymbol{m}|| = \frac{2}{||\boldsymbol{w}||} \tag{13}$$

となる。

(2)

ソフトマージン SVM の主形式

$$\arg\min_{\boldsymbol{w},\, b,\, \boldsymbol{\xi}} \left\{ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^n \xi_i \right\} \tag{14}$$

subject to

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) \geq 1 - \xi_i \tag{15}$$
$$\xi \geq 0 \tag{16}$$

から，双対形式を導く。

一般に，主問題

$$f^* = \min_{x \in \Xi} f(\boldsymbol{x}) \tag{17}$$

subject to

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \tag{18}$$
$$\boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0} \tag{19}$$

に対する Lagrange 関数は，

$$L(\boldsymbol{x},\, \boldsymbol{\lambda},\, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{20}$$

となり，Lagrange 双対問題は，

$$f^* = \max_{\boldsymbol{\lambda},\, \boldsymbol{\mu}} \inf{}_{\boldsymbol{x} \in \boldsymbol{X}} L(\boldsymbol{x},\, \boldsymbol{\lambda},\, \boldsymbol{\mu}) \tag{21}$$

subject to

$$\boldsymbol{\mu} \geq \boldsymbol{0} \tag{22}$$

となる。また，相補性条件より，

$$\boldsymbol{\mu} \odot \boldsymbol{h} = \boldsymbol{0} \tag{23}$$

が成立する。ここで，$\odot$ は要素ごとの積を表す。

これを今回の条件に適用すると，Lagrange 関数は，

$$L(\boldsymbol{w},\, b,\, \boldsymbol{\alpha},\, \boldsymbol{\beta}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^n \xi_i - \sum_i^n \alpha_i \left\{ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) - 1 + \xi_i \right\} - \sum_i^n \beta_i \xi_i \tag{24}$$

subject to

$$\boldsymbol{\alpha} \geq \boldsymbol{0} \tag{25}$$
$$\boldsymbol{\beta} \geq \boldsymbol{0} \tag{26}$$
$$\alpha_i \left\{ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) - 1 + \xi_i \right\} = 0 \tag{27}$$
$$\beta_i \xi_i = 0 \tag{28}$$

L の $\boldsymbol{w}$, b, $\boldsymbol{\xi}$ による偏微分を考える。

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i^n \alpha_i y_i \boldsymbol{x}_i = \boldsymbol{0} \tag{29}$$

$$\hat{\boldsymbol{w}} = \sum_i^n \alpha_i y_i \boldsymbol{x}_i \tag{30}$$

また,

$$\frac{\partial L}{\partial b} = \sum_i^n \alpha_i y_i = 0 \tag{31}$$

また,

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{32}$$

から,

$$\alpha_i + \beta_i = C \tag{33}$$

式 (**??**) の Lagrange 関数を，式 (**??**)，(**??**)，(**??**) を用いて変形する。

$$L(\boldsymbol{w},\ b,\ \boldsymbol{\alpha},\ \boldsymbol{\beta}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^n \xi_i - \sum_i^n \alpha_i \{y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b) - 1 + \xi_i\} - \sum_i^n \beta_i \xi_i \tag{34}$$

$$= \frac{1}{2}||\boldsymbol{w}||^2 - \boldsymbol{w}^{\mathrm{T}}\sum_i^n \alpha_i y_i \boldsymbol{x}_i + +b\sum_i^n \alpha_i y_i \sum_i^n \alpha_i + \sum_i^n (C - \alpha_i - \beta_i)\xi_i \tag{35}$$

$$= \frac{1}{2}||\sum_i^n \alpha_i y_i \boldsymbol{x}_i||^2 - \sum_j^n \alpha_j y_j \boldsymbol{x}_j^{\mathrm{T}} \sum_i^n \alpha_i y_i \boldsymbol{x}_i + \sum_i^n \alpha_i \tag{36}$$

$$= \sum_i^n \alpha_i - \frac{1}{2}\sum_{i,j}^n \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \tag{37}$$

よって，$\boldsymbol{w}$, $\boldsymbol{\xi}$, $\boldsymbol{\beta}$ が Lagrange 関数から消去されたので，結局，

$$L(\boldsymbol{\alpha}) = \sum_i^n \alpha_i - \frac{1}{2}\sum_{i,j}^n \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \tag{38}$$

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) \tag{39}$$

が解くべき問題となる。また，式 (**??**)，(**??**)，(**??**) から，

$$0 \leq \alpha_i \leq C \tag{40}$$

$$0 \leq \beta_i \leq C \tag{41}$$

なので，式 (**??**) と合わせて，制約は，

$$0 \leq \alpha_i \leq C \tag{42}$$

$$\sum_i^n \alpha_i y_i = 0 \tag{43}$$

となる。

1 プログラム

実行環境と用いた言語・ライブラリを以下の表 3 に示す。

表 1: プログラムの実行環境

OS	:	Microsoft Windows 10 Pro (64bit)
CPU	:	Intel(R) Core(TM) i5-4300U
RAM	:	4.00 GB
使用言語	:	Python3.6
可視化	:	matplotlib ライブラリ

ソースコードは Listing 1 に示した。以下に簡単に各関数の説明を記す。

- load_data
  .mat ファイルからデータを取り出す。
- plot
  点群及び境界をプロットする。2 クラスを o と x で表し，分類結果が正しいものを青，誤っているものを赤で示す。
- perceptron
  パーセプトロンを用いて重みを求める関数。
- mse
  MSE 法を用いて重みを求める関数。LMS 法を用いる場合と解析的に求める場合とを使い分けられる。

Listings 1: assignment1.py

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
import cvxopt


def load_data(path):
    data = loadmat(path)
    #print(data.keys())
    x = data['x'].T
    l = data['l'].T
    n = data['n'][0, 0]
    d = data['d'][0, 0]
    return x, l, n, d


def inner_prod(x1, x2):
```

```
    value = x1.T.dot(x2)
    return value


def gauss_kernel(x1, x2, h):
    value = (1/2) * np.exp(-(x1 - x2)**2 / (2*h**2))
    return value


def solve_svm(x, y, kernel=inner_prod, eps=1e-5):
    n = y.shape[0]
    #K = kernel(x.T, x.T)
    #P = K * y.dot(y.T)
    h = x * y
    P = h.dot(h.T)
    qpP = cvxopt.matrix(P)
    qpq = cvxopt.matrix(-np.ones(n), (n, 1))
    qpG = cvxopt.matrix(-np.eye(n))
    qph = cvxopt.matrix(np.zeros(n), (n, 1))
    qpA = cvxopt.matrix(y.T.astype(float), (1, n))
    qpb = cvxopt.matrix(0.)

    cvxopt.solvers.options['abstol'] = eps

    res = cvxopt.solvers.qp(qpP, qpq, qpG, qph, qpA, qpb)
    alpha = np.reshape(np.array(res['x']), -1)[:, np.newaxis]

    sv = (alpha > eps)
    isv = np.where(sv)[-1]

    w = np.sum(x * ((y*alpha) * np.ones(n)[:, np.newaxis]), axis=0)
    b = np.sum(x[isv, :].dot(w) - y[isv]) / np.sum(sv)
    return w, b, alpha


def plot(x, y, w, b, alpha, eps=1e-5, fig_path=None):
    plt.figure()
    plt.xlim([-1, 1])
    plt.ylim([-1, 1])

    sv = (alpha > eps)
    plt.plot(x[np.where((y>0) & sv), 0], x[np.where((y>0) & sv), 1], 'bo')
    plt.plot(x[np.where((y>0) & ~sv), 0], x[np.where((y>0) & ~sv), 1], 'bx')
    plt.plot(x[np.where((y<0) & sv), 0], x[np.where((y<0) & sv), 1], 'ro')
    plt.plot(x[np.where((y<0) & ~sv), 0], x[np.where((y<0) & ~sv), 1], 'rx')

    if abs(w[0]) > abs(w[1]):
```

```
        plt.plot([-1, 1], [(b+1+w[0])/w[1], (b+1-w[0])/w[1]])
        plt.plot([-1, 1], [(b-1+w[0])/w[1], (b-1-w[0])/w[1]])
    else:
        plt.plot([(b+1+w[1])/w[0], (b+1-w[1])/w[0]], [-1, 1])
        plt.plot([(b-1+w[1])/w[0], (b-1-w[1])/w[0]], [-1, 1])

    if fig_path:
        plt.savefig(fig_path)
    plt.show()


def main():
    # settings
    eps = 1e-5
    data_type = ’linear’
    #data_type = ’qlinear’
    #data_type = ’slinear’
    #data_type = ’nonlinear’
    data_path = f’../data/{data_type}-data.mat’
    fig_path = f’../figures/assignment1_{data_type}_result.png’
    np.random.seed(0)
    cvxopt.solvers.options[’reltol’] = 1e-10
    cvxopt.solvers.options[’show_progress’] = False


    # load data
    x, y, n, d = load_data(data_path)
    print(f’Data Type: {data_type}  #Sample: {n}  #Dim: {d}\n’)


    # calc
    kernel = inner_prod
    w, b, alpha = solve_svm(x, y, kernel=kernel, eps=eps)


    # result
    print(f’w = {w}  b = {b}’)
    #print(’alpha =’, alpha)

    plot(x, y, w, b, alpha, eps=eps, fig_path=fig_path)


if __name__ == ’__main__’:
    main()
```