

机器学习课程实验十

2022 年 12 月 1 日 苏博南 202000460020

1 Linear Support Vector Machine(SVM)

1.1 理论推导

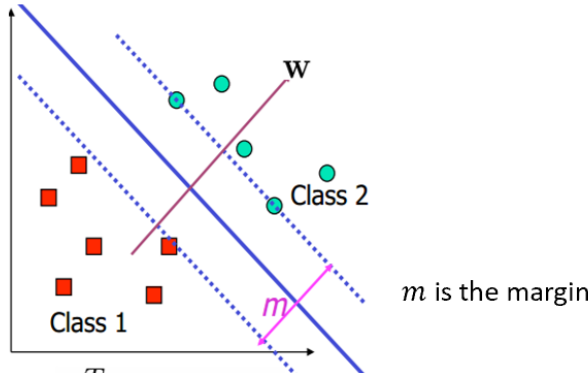
对于一个样本点 \mathbf{x}_i , 和一个超平面 $\mathbf{w}^T \mathbf{x} + b = 0$, 我们定义点到平面的距离为:

$$distance = \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|} \quad (1)$$

那么对于一个可线性二分类的数据集, 我们不仅希望得到一个超平面划分它, 还希望两边正负数据离超平面的最短距离尽量大。即我们希望:

$$\begin{cases} \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|} \geq \frac{m}{2} & y_i = 1 \\ \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|} \leq -\frac{m}{2} & y_i = -1 \end{cases} \quad (2)$$

其中, $\frac{m}{2}$ 就是正, 负类样本点距离超平面的最小距离:



然后处理下, 我们可以记 $\mathbf{w}' = \frac{2\mathbf{w}}{\|\mathbf{w}\|m}$, $b' = \frac{2b}{\|\mathbf{w}\|m}$, 则有:

$$\begin{cases} \mathbf{w}'^T \mathbf{x}_i + b' \geq 1 & y_i = 1 \\ \mathbf{w}'^T \mathbf{x}_i + b' \leq -1 & y_i = -1 \end{cases} \Leftrightarrow y_i(\mathbf{w}'^T \mathbf{x}_i + b') \geq 1 \quad (3)$$

那些满足 $y_i(\mathbf{w}'^T \mathbf{x}_i + b') = 1$ 的样本点, 即落在超平面两侧 margin 边缘上的样本点被称为 **Support Vector**, (注意到 $\mathbf{w}'^T \mathbf{x} + b' = 0$ 和 $\mathbf{w}^T \mathbf{x} + b = 0$ 是同一个平面, 即分界超平面) 它们这些点到分界超平面的距离为:

$$distance = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|} = \frac{|\mathbf{w}'^T \mathbf{x} + b'|}{\|\mathbf{w}'\|} = \frac{1}{\|\mathbf{w}'\|} m = \frac{2}{\|\mathbf{w}'\|} \quad (4)$$

所以我们得到描述 SVM 的规划问题:

$$\begin{aligned} \max & \quad 2/\|\mathbf{w}'\| \\ \text{s.t.} & \quad y_i(\mathbf{w}'^T \mathbf{x}_i + b') \geq 1, i = 1, 2, \dots, m \end{aligned} \quad (5)$$

然后为了方便解这个问题，我们稍微将其进行等价变化：

$$\begin{aligned} \min f(\mathbf{w}') &= \frac{1}{2} \|\mathbf{w}'\|^2 = \frac{1}{2} \mathbf{w}'^T \mathbf{w}' \\ \text{s.t. } 1 - y_i(\mathbf{w}' \mathbf{x}_i + b') &\leq 0, i = 1, 2, \dots, m \end{aligned} \quad (6)$$

然后利用拉格朗日乘子构造拉格朗日函数：

$$\mathcal{L}(\mathbf{w}', b', \vec{\alpha}) = f(\mathbf{w}') + \sum_{i=1}^m \alpha_i [1 - y_i(\mathbf{w}' \mathbf{x}_i + b')] \quad (7)$$

所以有：

$$f(\mathbf{w}') = \max_{\alpha_i \geq 0} \mathcal{L}(\mathbf{w}', b', \vec{\alpha}) \quad (8)$$

因为 $\alpha_i \geq 0$ ，然后它乘的 $[1 - y_i(\mathbf{w}' \mathbf{x}_i + b')] \leq 0$ ，所以 $\alpha_i \equiv 0$ 时取得最大值。然后重点是：

$$\min_{\mathbf{w}'} f(\mathbf{w}') = \min_{\mathbf{w}'} \max_{\alpha_i \geq 0} \mathcal{L}(\mathbf{w}', b', \vec{\alpha}) \geq \max_{\alpha_i \geq 0} \min_{\mathbf{w}'} \mathcal{L}(\mathbf{w}', b', \vec{\alpha}) \quad (9)$$

因为最小值的最大值永远小于等于最大值的最小值。根据规划问题的对偶性，如果原问题有最优解的话，上述等号可以取等。具体可参考 Karush-Kuhn-Tucker(KKT) 条件。于是原问题转化为：

$$\begin{aligned} \max_{\alpha_i \geq 0} \min_{\mathbf{w}'} \mathcal{L}(\mathbf{w}', b', \vec{\alpha}) \\ \text{s.t. } \alpha \geq 0 \end{aligned} \quad (10)$$

然后就可以进一步化简，有：

$$\begin{aligned} \min_{\mathbf{w}'} \mathcal{L}(\mathbf{w}', b', \vec{\alpha}) \text{ 可以直接求解：} \\ \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{w}'} = 0 \Rightarrow \mathbf{w}' = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{\partial b'} = 0 \Rightarrow 0 = \sum_{i=1}^m \alpha_i y_i \end{cases} \end{aligned} \quad (11)$$

根据上述结论可得出：

$$\mathcal{L}(\sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, b', \vec{\alpha}) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (12)$$

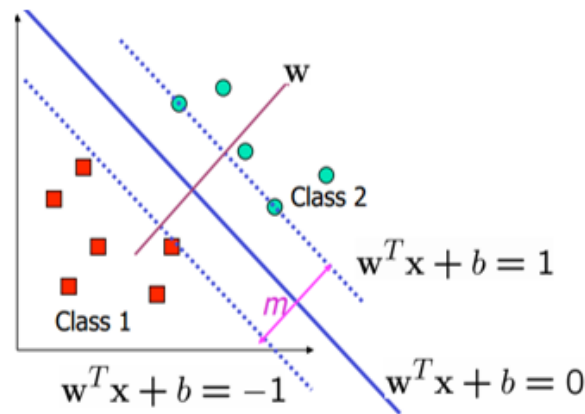
所以最终原规划问题等价于：

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t. } \alpha \geq 0, \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (13)$$

实际上，求解完成后，那些 $\alpha_i > 0$ 对应的点就是 support vector。其余的点都有 $\alpha_i = 0$ 。求解完成后，可以得到：

$$\begin{aligned} \mathbf{w}' &= \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \\ y_s [\mathbf{w}' \mathbf{x}_s + b'] &= 1 \Rightarrow b' = \frac{1}{y_s} - \mathbf{w}' \mathbf{x}_s \end{aligned} \quad (14)$$

其中， (\mathbf{x}_s, y_s) 是任意一个 support vector，即取一个 $\alpha_s > 0$ 的点就好，就满足代入方程等于 1。至此，我们便求出了下图所示三个超平面：



1.2 Matlab 代码

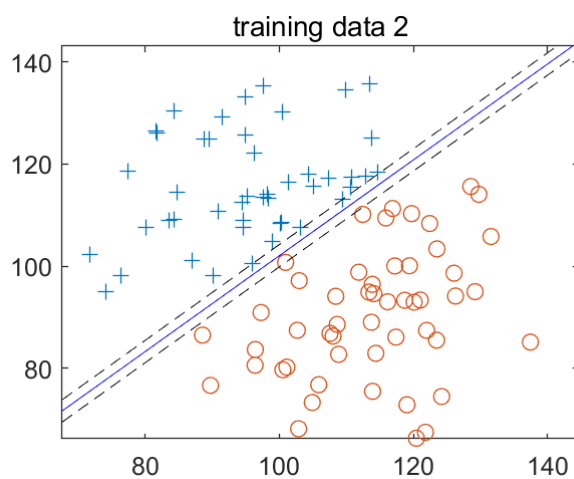
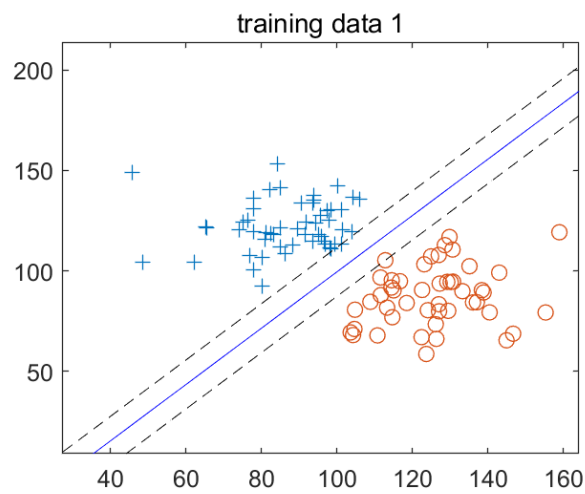
```

1  data = load("./data5/training_2.txt");
2  pos = [];
3  neg = [];
4  [m, n] = size(data);
5  m = 100;
6
7  for i = 1 : m
8      if data(i, n) == 1
9          pos = [pos; data(i, 1 : n - 1)];
10     else
11         neg = [neg; data(i, 1 : n - 1)];
12     end
13 end
14
15 plot(pos(:, 1), pos(:, 2), '+');
16 hold on;
17 plot(neg(:, 1), neg(:, 2), 'o');
18
19 prob = optimproblem('ObjectiveSense', 'max');
20
21
22 X = data(1 : m, 1 : n - 1);
23 Y = data(1 : m, n);
24 clear data;
25
26 alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);
27 tmp = (Y * Y') .* (X * X') .* (alpha * alpha');
28 prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
29 prob.Constraints.con = sum(alpha .* Y) == 0;
30
31 [s, f] = solve(prob);
32

```

```
33 w = zeros(1, n - 1);
34 for i = 1 : size(X, 1)
35     w = w + s.alpha(i) * Y(i) * X(i, :);
36 end
37
38 ys = 0;
39 xs = zeros(1, n - 1);
40 for i = 1 : size(X, 1)
41     if (s.alpha(i) > 1e-6)
42         ys = Y(i);
43         xs = X(i, :);
44         break;
45     end
46 end
47
48 b = ys;
49 for i = 1 : size(X, 1)
50     b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
51 end
52
53 xs = 20 : 1 : 180;
54 ys = (-w(1) * xs - b) / w(2);
55 plot(xs, ys, 'b-');
56 ys = (1 - w(1) * xs - b) / w(2);
57 plot(xs, ys, 'k--');
58 ys = (-1 - w(1) * xs - b) / w(2);
59 plot(xs, ys, 'k--');
60 title('training data 2');
61
62 test_set = load('./data5/test_2.txt');
63 pred = sign(test_set(:, 1 : 2) * w' + b);
64 acc = sum(pred == test_set(:, 3)) / size(test_set, 1)
65 % training data 1 acc = 99.6
66 % training data 2 acc = 99.6
```

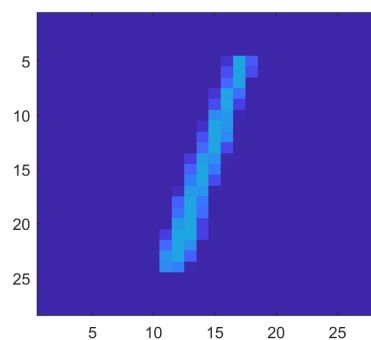
1.3 运行结果



在测试集上测试通过率均为 99.6%，即 498/500。

2 手写数字识别

首先可以用给顶的 `strimage.m` 函数进行 load 数据：



然后求解策略就是用 Linear SVM 求解。实验文档要求增加如下约束条件来 regularization:

$$\alpha \leq C \quad (15)$$

测试代码如下:

```

1  [Y, X] = strimage(100);
2
3  [m, n] = size(X);
4
5  prob = optimproblem('ObjectiveSense', 'max');
6  alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);
7  tmp = (Y * Y') .* (X * X') .* (alpha * alpha');
8  prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
9  prob.Constraints.con = sum(alpha .* Y) == 0;
10
11 prob.Constraints.con1 = alpha <= 10;
12
13 [s, f] = solve(prob);
14
15 w = zeros(1, n);
16 for i = 1 : size(X, 1)
17     w = w + s.alpha(i) * Y(i) * X(i, :);
18 end
19
20 ys = 0;
21 xs = zeros(1, n);
22 for i = 1 : size(X, 1)
23     if (s.alpha(i) > 1e-6)
24         ys = Y(i);
25         xs = X(i, :);
26         break;
27     end
28 end
29
30 b = ys;
31 for i = 1 : size(X, 1)
32     b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
33 end
34
35 [test_label, test_set] = strimage_test(1500);
36 pred = sign(test_set(:, 1 : n) * w' + b);
37 acc = sum(pred == test_label) / 1500
38 % C = 0, 99.67

```

根据我的测试结果, 选 $C = 1, 2, 3, 10$ 均有着 99.8% 的准确率。

3 Nonlinear SVM

当数据集线性不可分的情况, 可以考虑使用**核函数**, 即对每个数据点 \mathbf{x}_i , 构造一个新的特征值 $\phi(\mathbf{x}_i)$, 使得数据集重新变得线性可分, 是一个升维度的思想。譬如若二维数据集 (x, y) 是 $x^2 + y^2 \leq 1$ 的为类-1, 若 $x^2 + y^2 > 2$ 的为类+1, 则它显然线性不可分, 分界线是一个圆。但如果我增加特征值 $z = \phi(x, y) = x^2 + y^2$, 那么数据集 (x, y, z) 就重现变得线性可分了, 分界平面为 $z = 1.5$ 。一张图说明数据转换过程:

Transforming The Data

- Define the **kernel** function K by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Train SVM:

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i^T \mathbf{x}_j}_{\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)} = \underbrace{K(\mathbf{x}_i, \mathbf{x}_j)}$$

s.t. $0 \leq \alpha_i \leq C, i = 1, \dots, n$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

Test SVM:

$$\mathbf{w}^* = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i$$

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b)$$

$$= \text{sgn}\left(\left(\sum_{i \in \mathcal{S}} \alpha_i y_i \underbrace{\mathbf{x}_i^T \mathbf{x}}_{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})}\right) + b\right) = \underbrace{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})}_{K(\mathbf{x}_i, \mathbf{x})}$$

实验文档要求我们选择核函数为:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-100 \times \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (16)$$

之后就类似前面的一样训练和测试了。代码如下:

```

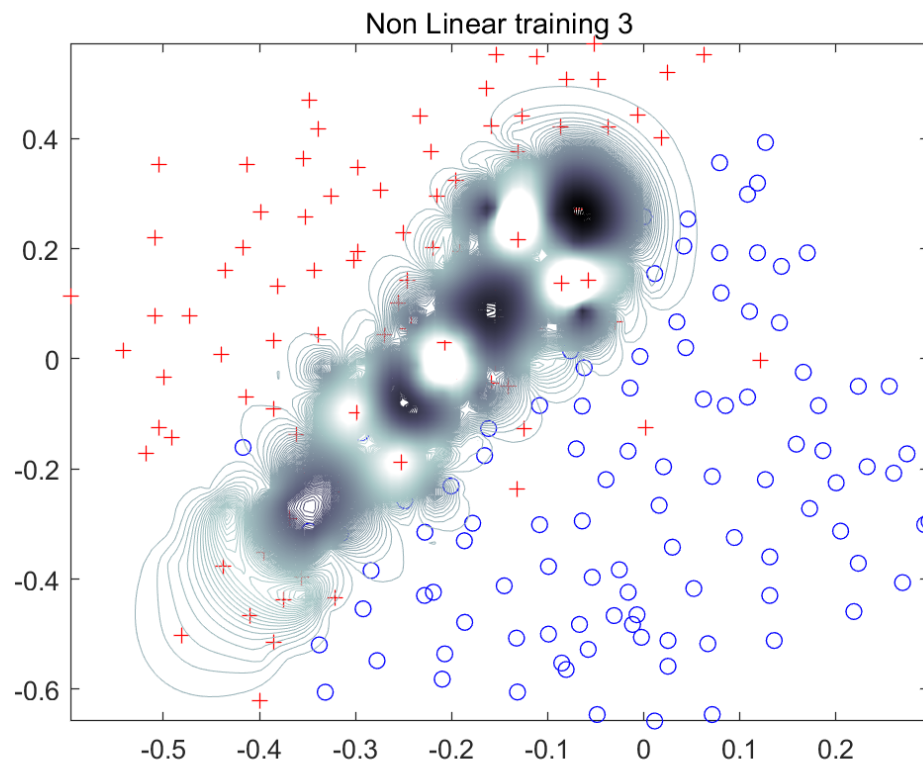
1 data = load('./data5/training_3.txt');
2
3 X = data(:, 1 : 2);
4 Y = data(:, 3);
5 xplot = linspace(min(X(:, 1)), max(X(:, 1)), 100)';
6 yplot = linspace(min(X(:, 2)), max(X(:, 2)), 100)';
7 [Xs, Ys] = meshgrid(xplot, yplot);
8
9 prob = optimproblem('ObjectiveSense', 'max');
10 alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);
11
12 K = zeros(size(X, 1), size(X, 1));
13 for i = 1 : size(X, 1)
14     for j = 1 : size(X, 1)

```

```
15     K(i, j) = exp(-100 * norm(X(i, :) - X(j, :))^2);
16     end
17 end
18 tmp = (Y * Y') .* (alpha * alpha') .* K;
19 prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
20 prob.Constraints.con = sum(alpha .* Y) == 0;
21
22 [s, f] = solve(prob);
23
24 ys = 0;
25 xs = zeros(1, 2);
26 for i = 1 : size(X, 1)
27     if (s.alpha(i) > 1e-6)
28         ys = Y(i);
29         xs = X(i, :);
30         break;
31     end
32 end
33
34 b = ys;
35 for i = 1 : size(X, 1)
36     b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
37 end
38
39 vals = zeros(size(xplot, 1), size(yplot, 1));
40 for i = 1 : 100
41     for j = 1 : 100
42         x = [xplot(i); yplot(j)];
43         vals(i, j) = b;
44         for k = 1 : size(X, 1)
45             vals(i, j) = vals(i, j) + s.alpha(k) * Y(k) * exp(-100 * norm(X(k, :) - x)
46             ^2);
47         end
48     end
49 end
50 pos = [];
51 neg = [];
52 for i = 1 : size(X, 1)
53     if (Y(i) == 1)
54         pos = [pos; X(i, :)];
55     else
56         neg = [neg; X(i, :)];
57     end
58 end
59 plot(pos(:, 1), pos(:, 2), 'r+', neg(:, 1), neg(:, 2), 'bo');
60 hold on;
61 colormap bone
62 contour(Xs, Ys, vals, -20 : 0.1 : 0);
```


63 `title('Non Linear training 3');`

然后画出来的图如下:



等高线圈起来的画的是值为-20 到 0 的范围。可以看到这些等高线在分界处很好地
把-1 类的数据点圈进去了，然后把 +1 类点丢在外面。