机器学习课程实验五

2022 年 12 月 7 日 苏博南 202000460020

1 Linear Support Vector Machine(SVM)

1.1 理论推导

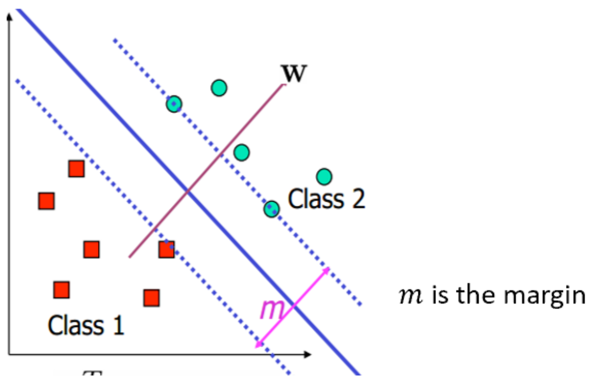
对于一个样本点 $\mathbf{x}_i$，和一个超平面 $\mathbf{w}^T\mathbf{x}+b=0$，我们定义点到平面的距离为:

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

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:

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

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:

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

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:

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

所以我们得到描述 SVM 的规划问题:

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

然后为了方便解这个问题，我们稍微将其进行等价变化:

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

然后利用拉格朗日乘子构造拉格朗日函数:

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

所以有:

$$f(\mathbf{w}') = \underset{\alpha_i \ge 0}{max}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \tag{8}$$

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

$$\underset{\mathbf{w}',b'}{min}\, f(\mathbf{w}') = \underset{\mathbf{w}',b'}{min}\ \underset{\alpha_i \ge 0}{max}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \ge \underset{\alpha_i \ge 0}{max}\ \underset{\mathbf{w}',b'}{min}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \tag{9}$$

因为**最小值的最大值永远小于等于最大值的最小值**:

$$\begin{aligned} &\underset{\mathbf{w}',b'}{min}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \le \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \le \underset{\alpha_i \ge 0}{max}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \\ \Rightarrow &\underset{\alpha_i \ge 0}{max}\, LHS \le \underset{\mathbf{w}',b'}{min}\, RHS \end{aligned} \tag{10}$$

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:

$$\begin{aligned} &\underset{\alpha_i \ge 0}{max}\ \underset{\mathbf{w}',b'}{min}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha}) \\ &s.t.\ \alpha \ge 0 \end{aligned} \tag{11}$$

然后就可以进一步化简，有:

$$\begin{aligned} &\underset{\mathbf{w}',b'}{min}\, \mathcal{L}(\mathbf{w}', b', \overrightarrow{\alpha})\text{可以直接求解:} \\ &\begin{cases} \frac{\partial \mathcal{L}}{\mathbf{w}'} = 0 \Rightarrow \mathbf{w}' = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{b'} = 0 \Rightarrow 0 = \sum_{i=1}^{m} \alpha_i y_i \end{cases} \end{aligned} \tag{12}$$

根据上述结论可得出:

$$\mathcal{L}(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, b', \overrightarrow{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{13}$$

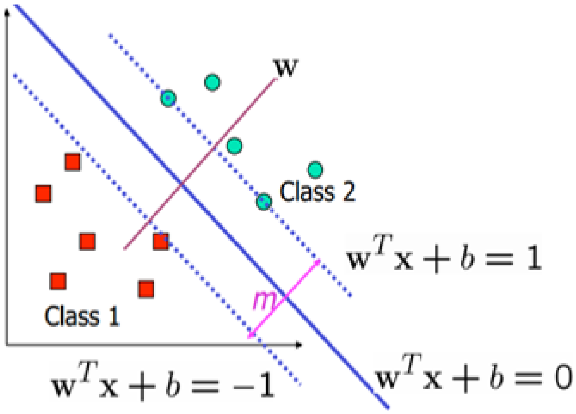
所以最终原规划问题等价为:

$$\begin{aligned} &\underset{\alpha}{max}\ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ &s.t.\ \alpha \ge 0, \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{14}$$

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:

$$\begin{aligned} &\mathbf{w}' = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \\ &y_s[\mathbf{w}'\mathbf{x}_s + b'] = 1 \Rightarrow b' = \frac{1}{y_s} - \mathbf{w}'\mathbf{x}_s \end{aligned} \tag{15}$$

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：

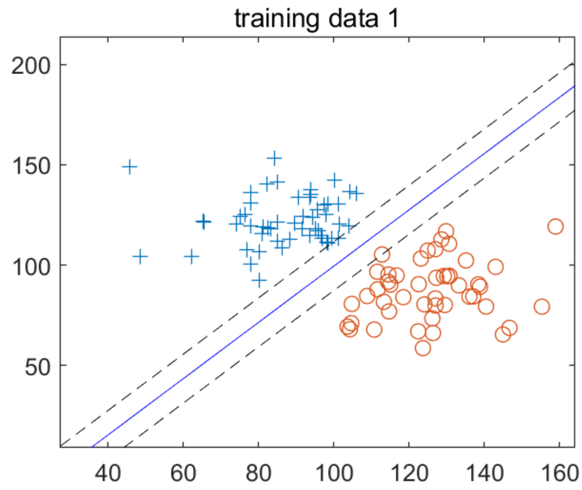


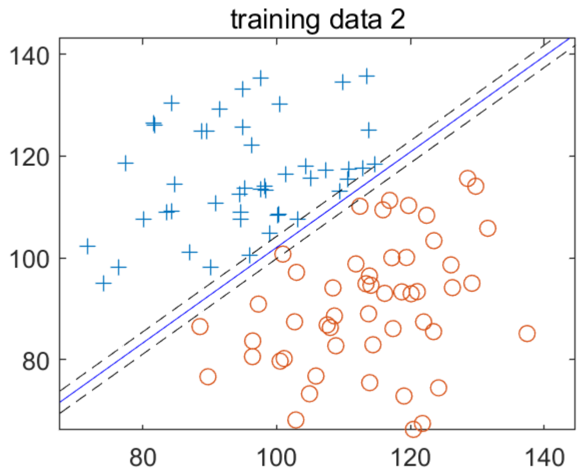
1.2 Matlab 代码

```
data = load("./data5/training_2.txt");
pos = [];
neg = [];
[m, n] = size(data);
m = 100;

for i = 1 : m
  if data(i, n) == 1
    pos = [pos; data(i, 1 : n - 1)];
  else
    neg = [neg; data(i, 1 : n - 1)];
  end
end

plot(pos(:, 1), pos(:, 2), '+');
hold on;
plot(neg(:, 1), neg(:, 2), 'o');

prob = optimproblem('ObjectiveSense', 'max');


X = data(1 : m, 1 : n - 1);
Y = data(1 : m, n);
clear data;

alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);
tmp = (Y * Y') .* (X * X') .* (alpha * alpha');
prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
prob.Constraints.con = sum(alpha .* Y) == 0;

```

```
[s, f] = solve(prob);

w = zeros(1, n - 1);
for i = 1 : size(X, 1)
  w = w + s.alpha(i) * Y(i) * X(i, :);
end

ys = 0;
xs = zeros(1, n - 1);
for i = 1 : size(X, 1)
  if (s.alpha(i) > 1e-6)
    ys = Y(i);
    xs = X(i, :);
    break;
  end
end

b = ys;
for i = 1 : size(X, 1)
  b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
end

xs = 20 : 1 : 180;
ys = (-w(1) * xs - b) / w(2);
plot(xs, ys, 'b-');
ys = (1 - w(1) * xs - b) / w(2);
plot(xs, ys, 'k--');
ys = (-1 - w(1) * xs - b) / w(2);
plot(xs, ys, 'k--');
title('training data 2');

test_set = load("./data5/test_2.txt");
pred = sign(test_set(:, 1 : 2) * w' + b);
acc = sum(pred == test_set(:, 3)) / size(test_set, 1)
% training data 1 acc = 99.6
% training data 2 acc = 99.6
```

1.3 运行结果

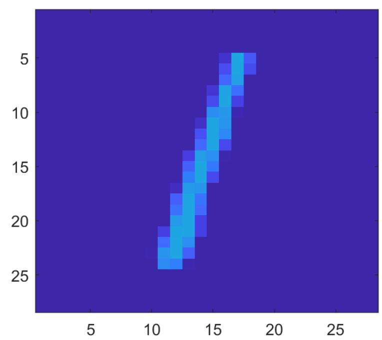




在测试集上测试通过率均为 99.6%，即 498/500。

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：

$$\alpha \leq C \tag{16}$$

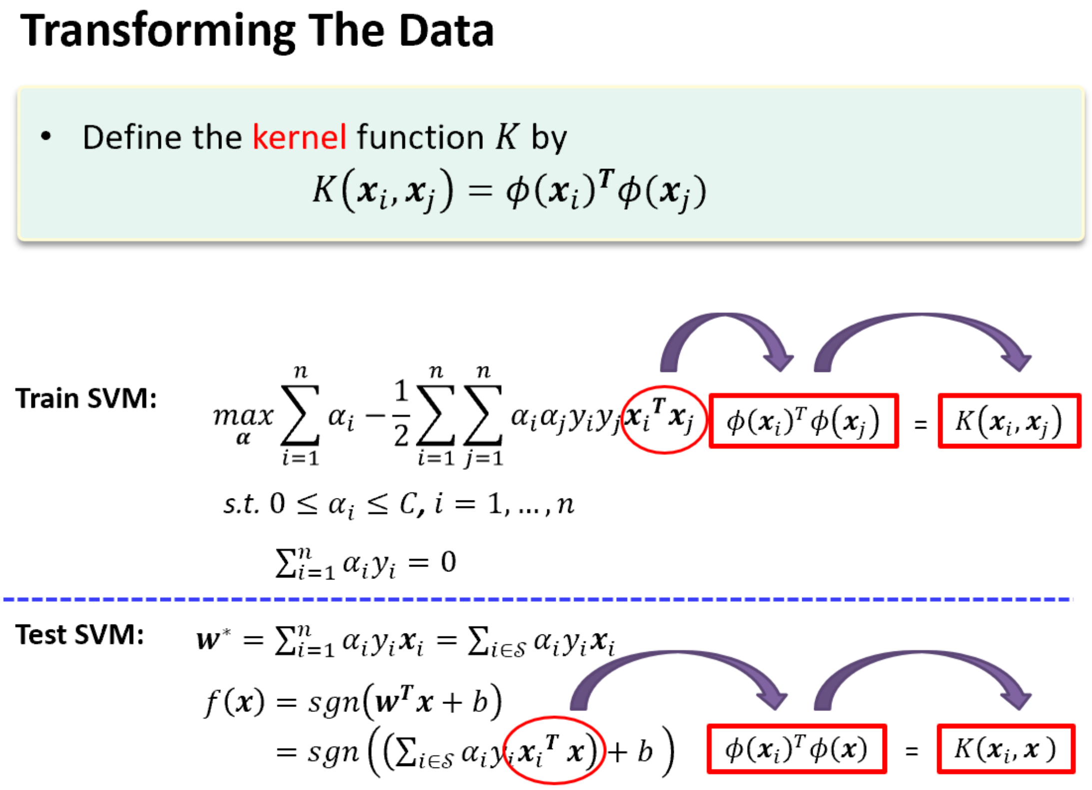
测试代码如下：

```
[Y, X] = strimage(100);

[m, n] = size(X);

prob = optimproblem('ObjectiveSense', 'max');
alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);
tmp = (Y * Y') .* (X * X') .* (alpha * alpha');
prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
prob.Constraints.con = sum(alpha .* Y) == 0;

prob.Constraints.con1 = alpha <= 10;

[s, f] = solve(prob);

w = zeros(1, n);
for i = 1 : size(X, 1)
  w = w + s.alpha(i) * Y(i) * X(i, :);
end

ys = 0;
xs = zeros(1, n);
for i = 1 : size(X, 1)
  if (s.alpha(i) > 1e-6)
    ys = Y(i);
    xs = X(i, :);
    break;
  end
end

b = ys;
for i = 1 : size(X, 1)
  b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
end

[test_label, test_set] = strimage_test(1500);
pred = sign(test_set(:, 1 : n) * w' + b);
acc = sum(pred == test_label) / 1500
% 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：



实验文档要求我们选择核函数为：

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-100 \times ||\mathbf{x}_i - \mathbf{x}_j||^2) \tag{17}$$

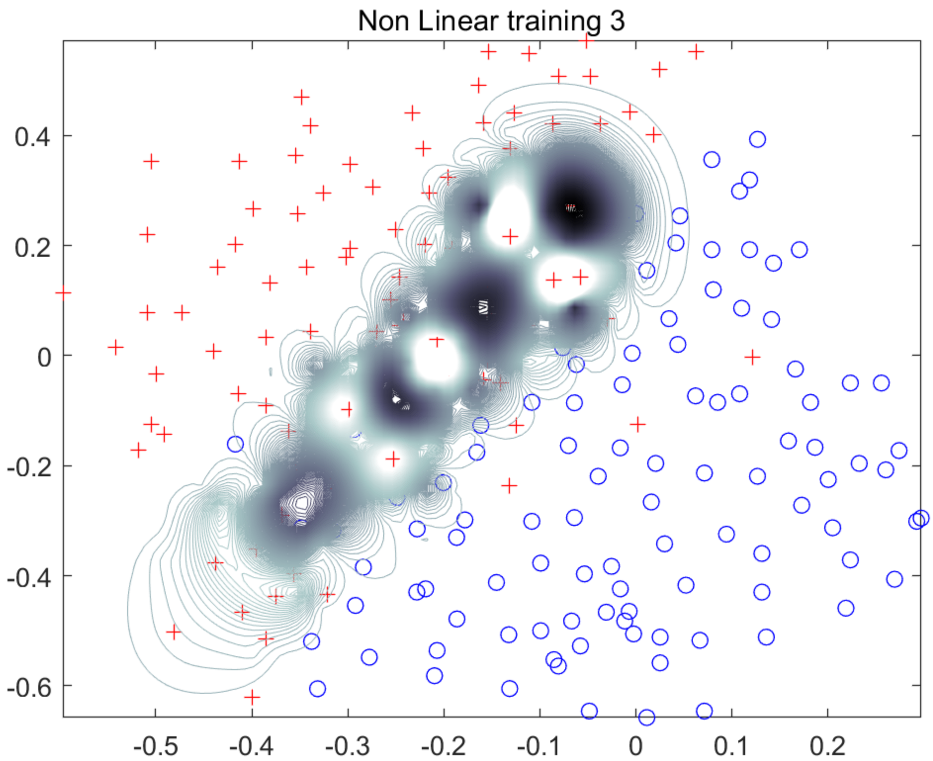
之后就类似前面的一样训练和测试了。代码如下：

```
data = load("./data5/training_3.txt");

X = data(:, 1 : 2);
Y = data(:, 3);
xplot = linspace(min(X(:, 1)), max(X(:, 1)), 100)';
yplot = linspace(min(X(:, 2)), max(X(:, 2)), 100)';
[Xs, Ys] = meshgrid(xplot, yplot);

prob = optimproblem('ObjectiveSense', 'max');
alpha = optimvar("alpha", size(X, 1), 1, 'LowerBound', 0);

K = zeros(size(X, 1), size(X, 1));
for i = 1 : size(X, 1)
  for j = 1 : size(X, 1)
```

```
    K(i, j) = exp(-100 * norm(X(i, :) - X(j, :))^2);
  end
end
tmp = (Y * Y') .* (alpha * alpha') .* K;
prob.Objective = sum(alpha) - 0.5 * sum(sum(tmp));
prob.Constraints.con = sum(alpha .* Y) == 0;

[s, f] = solve(prob);

ys = 0;
xs = zeros(1, 2);
for i = 1 : size(X, 1)
  if (s.alpha(i) > 1e-6)
    ys = Y(i);
    xs = X(i, :);
    break;
  end
end

b = ys;
for i = 1 : size(X, 1)
  b = b - s.alpha(i) * Y(i) * X(i, :) * xs';
end

vals = zeros(size(xplot, 1), size(yplot, 1));
for i = 1 : 100
  for j = 1 : 100
    x = [xplot(i); yplot(j)];
    vals(i, j) = b;
    for k = 1 : size(X, 1)
      vals(i, j) = vals(i, j) + s.alpha(k) * Y(k) * exp(-100 * norm(X(k, :) - x)
^2);
    end
  end
end

pos = [];
neg = [];
for i = 1 : size(X, 1)
  if (Y(i) == 1)
    pos = [pos; X(i, :)];
  else
    neg = [neg; X(i, :)];
  end
end
plot(pos(:, 1), pos(:, 2), 'r+', neg(:, 1), neg(:, 2), 'bo');
hold on;
colormap bone
contour(Xs, Ys, vals, -20 : 0.1 : 0);
```

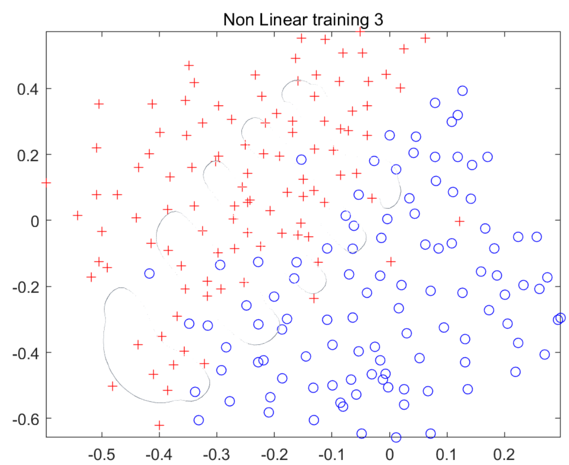
```
63    title('Non Linear training 3');
```

然后画出来的图如下：

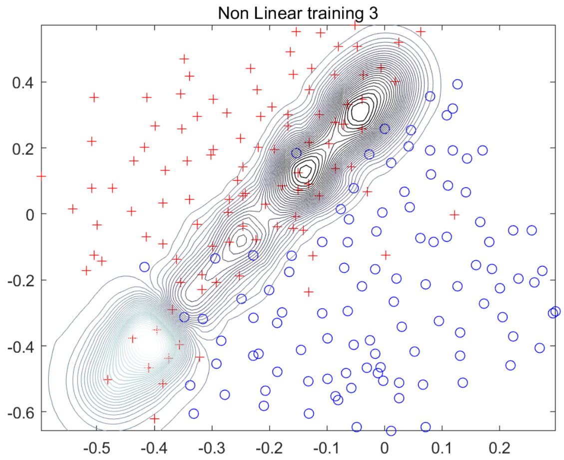


等高线圈起来的我画的是值为-20 到 0 的范围。可以看到这些等高线在分界处很好地把-1 类的数据点圈进去了，然后把 +1 类点丢在外面。实验文档还问了当核函数为 $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-1 \times ||\mathbf{x}_i - \mathbf{x}_j||^2), K(\mathbf{x}_i, \mathbf{x}_j) = exp(-10 \times ||\mathbf{x}_i - \mathbf{x}_j||^2), K(\mathbf{x}_i, \mathbf{x}_j) = exp(-1000 \times ||\mathbf{x}_i - \mathbf{x}_j||^2)$ 时的情况。

$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-10 \times ||\mathbf{x}_i - \mathbf{x}_j||^2)$ 时：



$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-1000 \times ||\mathbf{x}_i - \mathbf{x}_j||^2)$ 时：



感觉还是 $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-100 \times ||\mathbf{x}_i - \mathbf{x}_j||^2)$ 效果最好。常数太大太太小都会使得数据集的特征值趋近于相同，导致分类效果差。