目录

1、数据集介绍 2

2、SVM 支持向量机模型 2

2.1、支持向量机模型介绍 2

2.2、支持向量机 SVM 算法 5

3、习题解答 6

3.1、习题 6.2 6

3.2、习题 6.8 9

4、附录 11

第六章 支持向量机

1、数据集介绍

本次实验使用到一个数据集，为西瓜数据集 3.0α。西瓜数据集 3.0α包含 17 条信息，每条信息对应西瓜的 2 种属性，给出了该西瓜是否为好瓜，“是”表示该西瓜是好瓜，“否”表该西瓜不是好瓜。西瓜数据集 3.0α的具体内容如下图所示。

表 1 西瓜数据集 3.0α

编号	密度	含糖率	好瓜
1	0.697	0.460	是
2	0.774	0.376	是
3	0.634	0.264	是
4	0.608	0.318	是
5	0.556	0.215	是
6	0.403	0.237	是
7	0.481	0.149	是
8	0.437	0.211	是
9	0.666	0.091	否
10	0.243	0.267	否
11	0.245	0.057	否
12	0.343	0.099	否
13	0.639	0.161	否
14	0.657	0.198	否
15	0.360	0.370	否
16	0.593	0.042	否
17	0.719	0.103	否

2、SVM 支持向量机模型

2.1、支持向量机模型介绍

支持向量机（support vector machines, SVM）是一类按监督学习方式对数据进行二分类的广义线性分类器，其决策边界是对学习样本求解的最大边距超平面，间隔最大使它有别于感知机；SVM 还包括核技巧，这使它成为实质上的非线性分类器。SVM 的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划

的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的的学习算法就是求解凸二次规划的最优化算法。

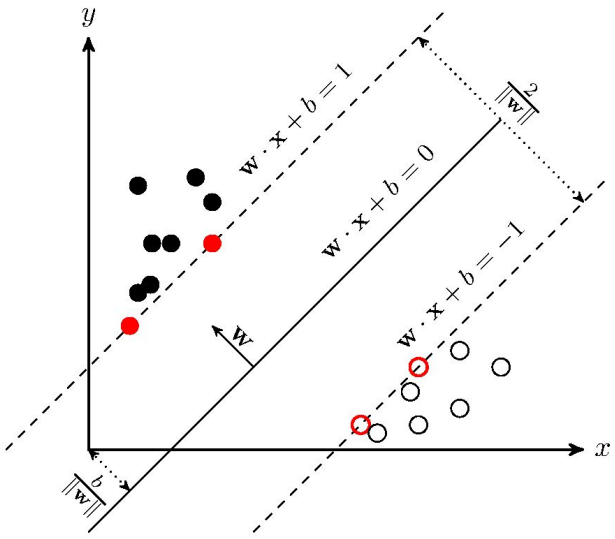


图 2.1.1 超平面模型

SVM 学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，$\boldsymbol{w}\cdot x+b=0$即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

- **间隔最大化**

假设给定一个特征空间上的训练数据集$T=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\dots,(\boldsymbol{x}_N,y_N)\}$，其中，$\boldsymbol{x}_i\in\mathbb{R}^n, y_i\in\{+1,-1\}, i=1,2,\dots N$，$\boldsymbol{x}_i$为第$i$个特征向量，$y_i$为类标记，当它等于+1 时为正例；为-1 时为负例。再假设训练数据集是线性可分的。

对于给定的数据集T和超平面$\boldsymbol{w}\cdot x+b=0$，定义超平面关于样本点$(\boldsymbol{x}_i,y_i)$的几何间隔为:

$$\gamma_i=y_i\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\cdot\boldsymbol{x}_i+\frac{b}{\|\boldsymbol{w}\|}\right)$$

超平面关于所有样本点的几何间隔的最小值为:

$$\gamma=\min_{i=1,2\dots,N}\gamma_i$$

根据以上定义，SVM 模型的求解最大分割超平面问题可以表示为以下约束最优化问题:

$$\min_{w,b} \frac{1}{2} \| \boldsymbol{w} \|^2$$

$$s.t.\ y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, i = 1,2, \dots, N$$

• 对偶问题

这是一个含有不等式约束的凸二次规划问题，可以对其使用拉格朗日乘子法得到其对偶问题（dual problem）。

首先，我们将有约束的原始目标函数转换为无约束的新构造的拉格朗日目标函数：

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \| \boldsymbol{w} \|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1)$$

根据拉格朗日函数对偶性，要满足对偶性，需要满足：①优化问题是凸优化问题；②满足 KKT 条件。为了得到求解对偶问题的具体形式，令$L(\boldsymbol{w}, b, \boldsymbol{\alpha})$对$\boldsymbol{w}$和$b$的偏导为 0，再带入拉格朗日目标函数，消去$\boldsymbol{w}$和$b$，得：

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) - \sum_{i=1}^{N} \alpha_i y_i \left(\left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x_j} \right) \cdot \boldsymbol{x_i} + b \right) + \sum_{i=1}^{N} \alpha_i$$

$$= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x_j}) + \sum_{i=1}^{N} \alpha_i$$

原问题转换为以下优化问题：

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{N} \alpha_i$$

$$\text{s.t.} \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, i = 1,2, \dots, N$$

对于任意训练样本$(\boldsymbol{x}_i, y_i)$，总有$\alpha_i = 0$或者$y_i(\boldsymbol{w} \cdot x + b) = 0$。若$\alpha_i = 0$，则该样本不会在最后求解模型参数的式子中出现。若$\alpha_i > 0$，则必有$y_i(\boldsymbol{w} \cdot x + b) = 1$，所对应的样本点位于最大间隔边界上，是一个支持向量。这显示出支持向量机的一个重要性质：训练完成后，大部分的训练样本都不需要保留，最终模型仅与支持向量有关。

• **非线性可分与核函数**

以上讨论都是在样本完全线性可分或者大部分样本点线性可分，但我们可能会遇到线性不可分的情况。在这种情况下，将二维线性不可分样本映射到高维空间中，让样本点在高维空间线性可分。

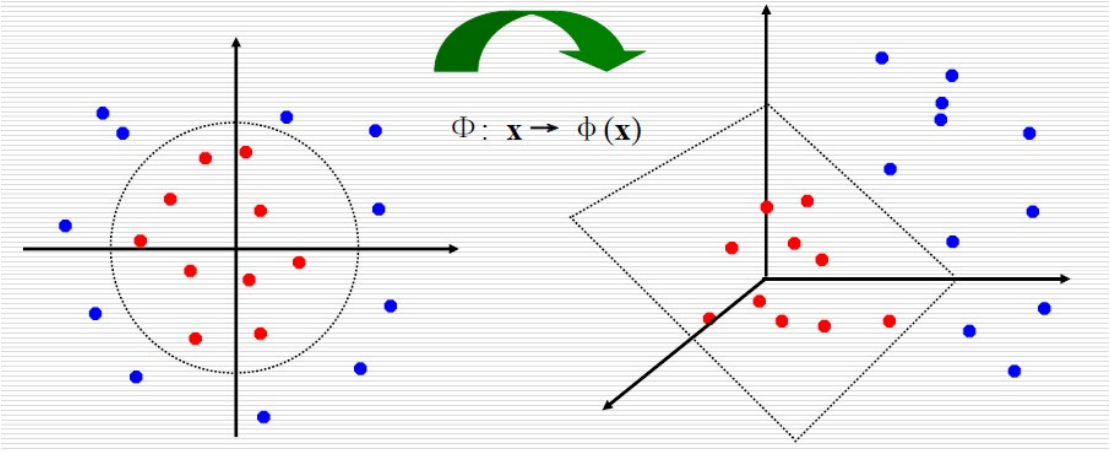


图 2.1.2 线性不可分转化为线性可分

对于在有限维度向量空间中线性不可分的样本，我们将其映射到更高维度的向量空间里，再通过间隔最大化的方式，学习得到支持向量机。

我们用 x 表示原来的样本点，用$\phi(x)$表示 x 映射到特征新的特征空间后到新向量。那么分割超平面可以表示为：$f(x) = \boldsymbol{w} \cdot \phi(x) + b$。

对于非线性 SVM 的对偶问题就变成了：

$$\min_{\lambda} \left[\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \left(\phi(x_i) \cdot \phi(x_j) \right) - \sum_{j=1}^{n} \lambda_i \right]$$

$$\text{s.t.} \sum_{i=1}^{n} \lambda_i y_i = 0, \lambda_i \geq 0, C - \lambda_i - \mu_i = 0$$

由于特征空间的维数可能很高，甚至是无穷维，因此直接计算$\phi(x_i) \cdot \phi(x_j)$通常是困难的，于是引入了核函数$\kappa(x_i, x_j) =< \phi(x_i), \phi(x_j) >= \phi(x_i)^T \phi(x_j)$。

2.2、支持向量机 SVM 算法

支持向量机学习算法如下：

输入： 训练数据集$T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$，其中，$\boldsymbol{x}_i \in \mathbb{R}^n, y_i \in \{+1, -1\}, i = 1,2, \dots N$;

输出： 分离超平面和分类决策函数

(1)选取适当的核函数$K(x_i, x_j)$和惩罚参数$C > 0$，构造并解决凸二次规划问题：

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) - \sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.} \sum_{i}^{N}\alpha_i y_i = 0$$

得到最优解$\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)^T$

(2)计算$\boldsymbol{\alpha}^*$的一个分量α_j^*满足条件$0 < \alpha_j^* < C$，计算

$$\boldsymbol{b}^* = \boldsymbol{\alpha}^* - \sum_{i=1}^{N}\boldsymbol{\alpha}^* y_i K(x_i, x_i)。$$

(3)分类决策函数：

$$f(x) = \text{sign}\left(\sum_{i=1}^{N}\alpha_i^* y_i K(x, x_i) + b^*\right)$$

3、习题解答

3.1、习题 6.2

【习题 5.5】试用 LIBSVM，在西瓜数据集 3.0α上分别用线性核和高斯核训练一个 SVM，并比较其支持向量的差别。

代码解析：

• LIBSVM 包的安装

打开 https://www.lfd.uci.edu/~gohlke/pythonlibs/#libsvm，下载 python3.8 对应的文件，在 whl 文件目录下执行 pip install 安装。

• 数据处理

LIBSVM 的数据格式为：<label> <index1>:<value1> <index2>:<value2> ...

第一列为标签；第二列为索引（1）和第一个特征值；第三列为索引（2）和第二个特征值；

代码如下：

```
# 将西瓜数据集3.0a转化为LIBSvm包规定的数据格式
def process_data(filename):
    df = pd.read_csv(filename, sep=' ')
    data = df.values
    x1 = data[:, 1]
    x2 = data[:, 2]
    y = data[:, 3]
    with open('watermelon_3a_svm.txt', 'w') as f:
        for m1, m2, n in zip(x1, x2, y):
            f.write("{} 1:{} 2:{}\n".format(int(n), round(m1, 3), round(m2, 3)))
    f.close()
```

生成的数据如下：

```
1. 1 1:0.697 2:0.46
2. 1 1:0.774 2:0.376
3. 1 1:0.634 2:0.264
4. 1 1:0.608 2:0.318
5. 1 1:0.556 2:0.215
6. 1 1:0.403 2:0.237
7. 1 1:0.481 2:0.149
8. 1 1:0.437 2:0.211
9. 0 1:0.666 2:0.091
10.0 1:0.243 2:0.267
11.0 1:0.245 2:0.057
12.0 1:0.343 2:0.099
13.0 1:0.639 2:0.161
14.0 1:0.657 2:0.198
15.0 1:0.36 2:0.37
16.0 1:0.593 2:0.042
17.0 1:0.719 2:0.103
```

• 主函数调用

本次使用 LIBSVM 进行训练，所以只需要调用相应函数即可实现，svm_read_problem 函数读取已经修改好格式的数据集，svm_train 函数负责 svm 训练，可以对 svm 模型参数进行调整，训练好的模型可以通过 svm_save_model 进行文件存储，svm_predict 函数负责通过已知模型预测输出。

```
1. def main():
2.     process_data('watermelon_3a.txt')
3.     y, x = svm_read_problem('watermelon_3a_svm.txt')
4.     model1 = svm_train(y, x, '-t 0 -c 1000')  # 线性核
5.     model2 = svm_train(y, x, '-t 2 -c 900 -g 0.8')  # 高斯核
6.     svm_save_model('Linear.model', model1)
7.     svm_save_model('Gauss.model', model2)
8.     svm_predict(y, x, model2)
```

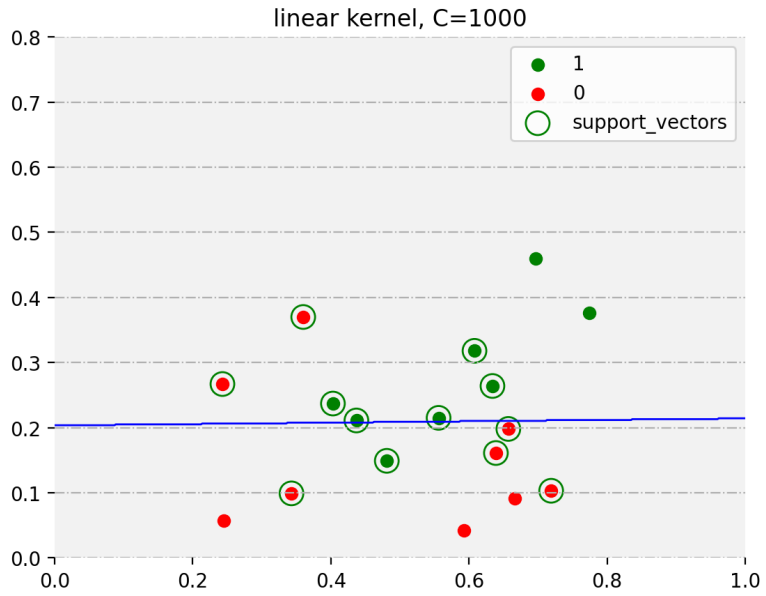
完整代码参见附录 1，SVM.py

线性核和高斯核 SVM 输出如下图所示：

①线性核 SVM，C = 1000， 准确率 82.4%。

```
D:\Anaconda3\python.exe D:/我的坚果云/课件/机器学习/作业/作业五-SVM/SVM.py
........*...*......*
optimization finished, #iter = 290
nu = 0.605166
obj = -10245.041181, rho = 1.881653
nSV = 12, nBSV = 9
Total nSV = 12
Accuracy = 82.3529% (14/17) (classification)
```

参数解读：
nu: 错误率 nu 参数
obj: SVM 文件转换为的二次规划求解得到的最小值
rho:为判决函数的常数项 b
nSV:为支持向量个数
nBSV: 边界上的支持向量个数
Total nSV:为支持向量总个数
另外，为了更加直观的看到 SVM 的分类情况，绘制了数据集的散点分布图，标出了支持向量以及决策边界：

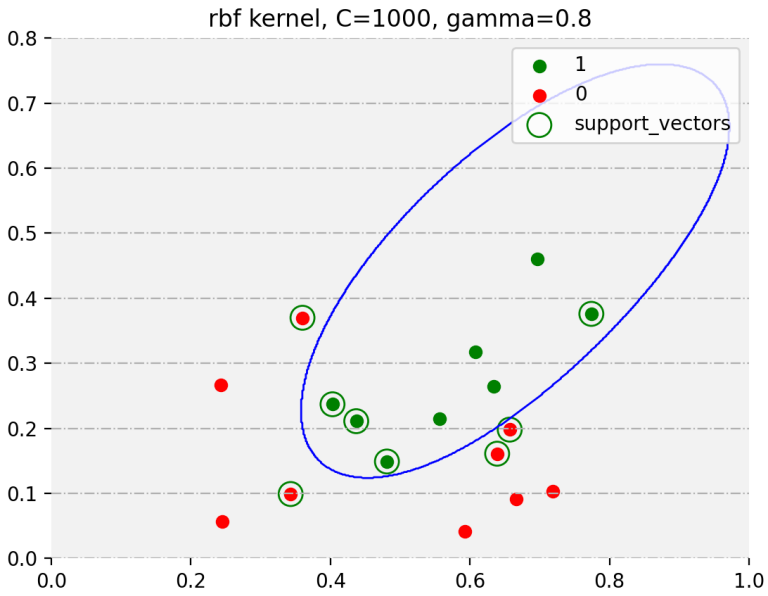


②高斯性核 SVM，C = 1000，gamma = 0.8 准确率 100%。

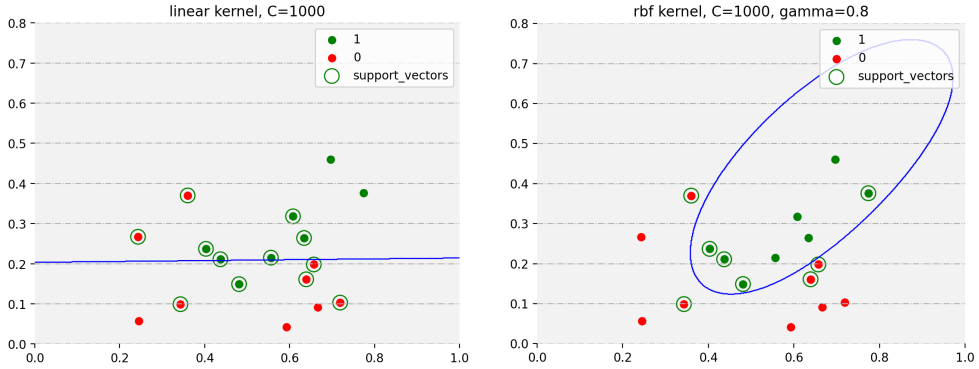
```
D:\Anaconda3\python.exe D:/我的坚果云/课件/机器学习/作业/作业五-SVM/SVM.py
............*..*
optimization finished, #iter = 241
nu = 0.387831
obj = -3722.591158, rho = 32.165448
nSV = 8, nBSV = 4
Total nSV = 8
Accuracy = 100% (17/17) (classification)
```

参数解读：
nu: 错误率 nu 参数
obj: SVM 文件转换为的二次规划求解得到的最小值
rho:为判决函数的常数项 b
nSV:为支持向量个数
nBSV: 边界上的支持向量个数
Total nSV:为支持向量总个数

另外，为了更加直观的看到 SVM 的分类情况，绘制了数据集的散点分布图，标出了支持向量以及决策边界：



线性核与高斯核的支持向量对比



对比线性核与高斯核的支持向量，在二者 C=1000 的情况下，线性核的支持向量为 12 个，高斯核支持向量为 8 个，线性核比高斯核的支持向量要多，但是，从分类结果来看，高斯核的准确率更高。

3.2、习题 6.8

以西瓜数据集 3.0α的“密度”为输入，“含糖率”为输出，试用 LIBSVM 训练一个 SVR。

代码解析：

- **LIBSVM 包的安装**

打开 https://www.lfd.uci.edu/~gohlke/pythonlibs/#libsvm，下载 python3.8 对应的文件，在 whl 文件目录下执行 pip install 安装。

- **数据处理**

LIBSVM 的数据格式为：<label> <index1>:<value1> <index2>:<value2> ...

第一列为标签；第二列为索引（1）和第一个特征值；第三列为索引（2）和第二个特征值；

代码如下：

```
# 将西瓜数据集3.0a转化为LIBSvm包规定的数据格式
def process_data(filename):
    df = pd.read_csv(filename, sep=' ')
    data = df.values
    x1 = data[:, 1]
    y = data[:, 2]
    with open('watermelon_3a_svr.txt', 'w') as f:
        for m1, m2 in zip(x1, y):
            f.write("{} 1:{}\n".format(round(m2, 3), round(m1, 3)
))
    f.close()
```

生成的数据如下

```
0.46 1:0.697
0.376 1:0.774
0.264 1:0.634
0.318 1:0.608
0.215 1:0.556
0.237 1:0.403
0.149 1:0.481
0.211 1:0.437
0.091 1:0.666
0.267 1:0.243
0.057 1:0.245
0.099 1:0.343
0.161 1:0.639
0.198 1:0.657
0.37 1:0.36
0.042 1:0.593
0.103 1:0.719
```

- **主函数调用**

本次使用 LIBSVM 进行训练，所以只需要调用相应函数即可实现，svm_read_problem 函数读取已经修改好格式的数据集，svm_train 函数负责 svm 训练，可以对 svm 模型参数进行调整 -t 3 即选择 SVR 模型，训练好的模型可以通过 svm_save_model 进行文件存储，svm_predict 函数负责通过已知模型预测输出。

```
def main():
    process_data('watermelon_3a.txt')
    y, x = svm_read_problem('watermelon_3a_svm.txt')
    model3 = svm_train(y, x, '-t 3 -c 10 -g 8')  # 线性核
    svm_save_model('SVR.model', model3)
    svm_predict(y, x, model3)
```

运行程序输出结果如下：

```
D:\Anaconda3\python.exe D:/我的坚果云/课件/机器学习/作业/作业五-SVM/SVR.py
*
optimization finished, #iter = 4
nu = 0.470588
obj = -1435.336637, rho = -1.816311
nSV = 8, nBSV = 8
Total nSV = 8
Accuracy = 52.9412% (9/17) (classification)
```

参数解读：
nu: 错误率 nu 参数
obj: SVM 文件转换为的二次规划求解得到的最小值
rho:为判决函数的常数项 b
nSV:为支持向量个数
nBSV: 边界上的支持向量个数
Total nSV:为支持向量总个数

从输出结果可以看到，最终有 8 个支持向量，模型准确率 53%

4、附录

1、SVM.py

```
# -*- coding: utf-8 -*
# @Time: 2021/5/13 13:20

import pandas as pd
from svmutil import *


# 将西瓜数据集 3.0a 转化为 LIBSvm 包规定的数据格式
def process_data(filename):
    df = pd.read_csv(filename, sep=' ')
    data = df.values
    x1 = data[:, 1]
    x2 = data[:, 2]
    y = data[:, 3]
    with open('watermelon_3a_svm.txt', 'w') as f:
        for m1, m2, n in zip(x1, x2, y):
            f.write("{} 1:{} 2:{}\n".format(int(n), round(m1, 3), round(m2, 3)))
    f.close()


def main():
```

```
    process_data('watermelon_3a.txt')
    y, x = svm_read_problem('watermelon_3a_svm.txt')
    model1 = svm_train(y, x, '-t 0 -c 1000')  # 线性核
    model2 = svm_train(y, x, '-t 2 -c 1000 -g 0.8')  # 高斯核
    svm_save_model('Linear.model', model1)
    svm_save_model('Gauss.model', model2)
    svm_predict(y, x, model2)


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

2、SVR.py

```
# -*- coding: utf-8 -*
# @Time: 2021/5/13 13:20

import pandas as pd
from svmutil import *


# 将西瓜数据集 3.0a 转化为 LIBSvm 包规定的数据格式
def process_data(filename):
    df = pd.read_csv(filename, sep=' ')
    data = df.values
    x1 = data[:, 1]
    y = data[:, 2]
    with open('watermelon_3a_svr.txt', 'w') as f:
        for m1, m2 in zip(x1, y):
            f.write("{} 1:{}\n".format(round(m2, 3), round(m1, 3)))
    f.close()


def main():
    process_data('watermelon_3a.txt')
    y, x = svm_read_problem('watermelon_3a_svm.txt')
    model3 = svm_train(y, x, '-t 3 -c 100 -g 10')  # 线性核
    svm_save_model('SVR.model', model3)
    svm_predict(y, x, model3)


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