**实验3 序贯最小优化算法（SMO）**

#### 一、实验目的

理解SMO算法的工作原理，编程实现该算法并用于非线性分类问题。

#### 二、实验内容

基于SMO算法，自定义一个实现核SVM分类器的类，其中核函数分别取线性核和多项式核。利用所设计的SVM类，在乳腺癌数据训练一个SVM分类器，测试其分类精度，并与sklearn中的SVM进行比较。

#### 三、实验环境

硬件：CPU 2.3GHz Intel Core i5，

内存 8GB 2133MHz LPDDR3，

硬盘 Macintosh HD

软件：macOS Catalina 10.15.1、Anaconda 3.6.1

数据：乳腺癌数据集

#### 四、实验原理

1. 序贯最小优化算法（Sequential Minimal Optimization，SMO）基本思想

若所有变量的解都满足优化问题的KKT条件，则解求得。否则，选两个变量（拉格朗日乘子）作为待更新的变量，固定其他变量，针对选出的变量构造一个二次规划问题，求解优化，反复如此，直到收敛。

2. 算法描述

表1 算法描述

|  |
| --- |
| **输入：**数据X，标签y |
| **过程：**  （1）取初值alpha=0，w=0，w0=0，k=0  （2）从alpha中选取优化变量记为alpha1和alpha2，求解两个变量的优化问题，更新得到新的alpha1和alpha2，将新的alpha1和alpha2更新到alpha对应的分量中。  （3）若精度epsilon范围内满足停机条件  则转到4，否则转到2  （4）取最新更新后的alpha为算法的解 |
| **输出：**近似解alpha |

3. 类设计

表2 SVM类的方法

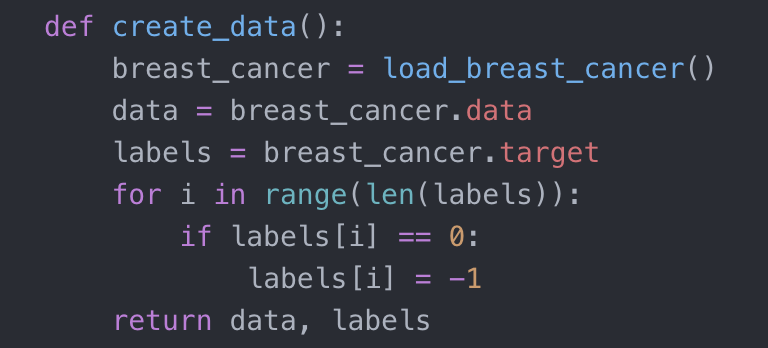
|  |  |
| --- | --- |
| **方法** | **描述** |
| \_\_init\_\_ | 类的构造函数 |
| \_g | 计算g(x)预测值 |
| \_E | 计算g(x)预测值和真实值的距离 |
| init\_args | 初始化各项参数 |
| \_KKT | 判断是否满足KKT条件 |
| kernel | 核函数的计算 |
| \_init\_alpha | 初始化 |
| \_compare | 截断alpha |
| fit | SMO训练过程 |
| predict | 对测试数据进行预测 |
| socre | 计算预测准确率 |
| \_weight | 计算w的值 |

#### 实验步骤

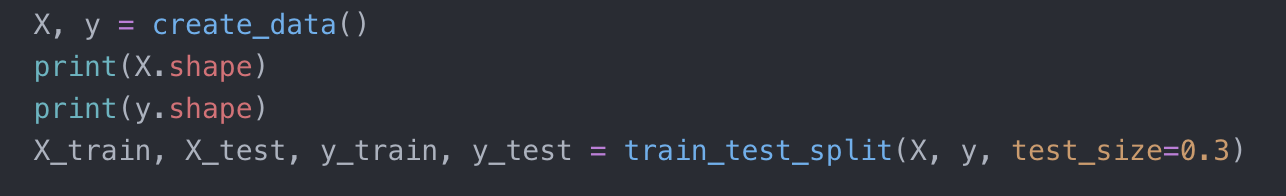
实验数据简介

乳腺癌数据集，从sklearn库中导入，其中包括569条数据，每条数据都包含30项指标，标签为0或1，表示是否确诊为患病。为了在SVM中使用，需要将0标签改为-1。30项指标所代表的含义可以通过feature\_names属性获得。

1. **载入数据**



通过sklearn.datasets中的load\_breast\_cancer函数加载数据集，并将标签中的0都改为-1，以便于传入SVM。最后返回数据和标签，



调用create\_data函数后，使用sklearn.model\_selection中的train\_test\_split函数，将数据集分为训练集和测试集，设定测试集占30%

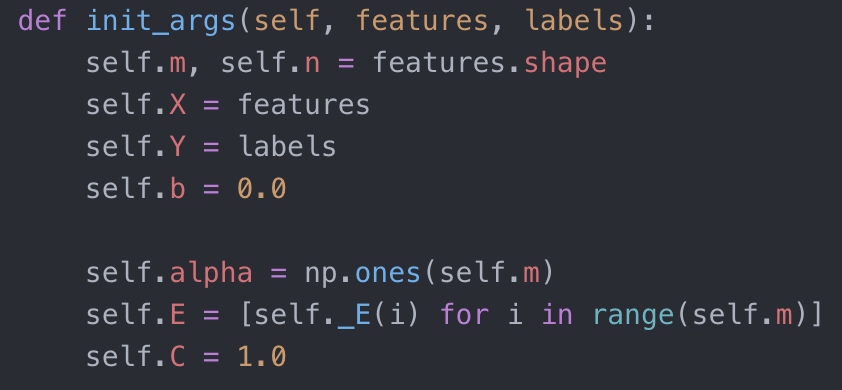
1. **代码描述**

SVM类中的fit函数是SMO算法的主要流程。

首先选初始化各个参数



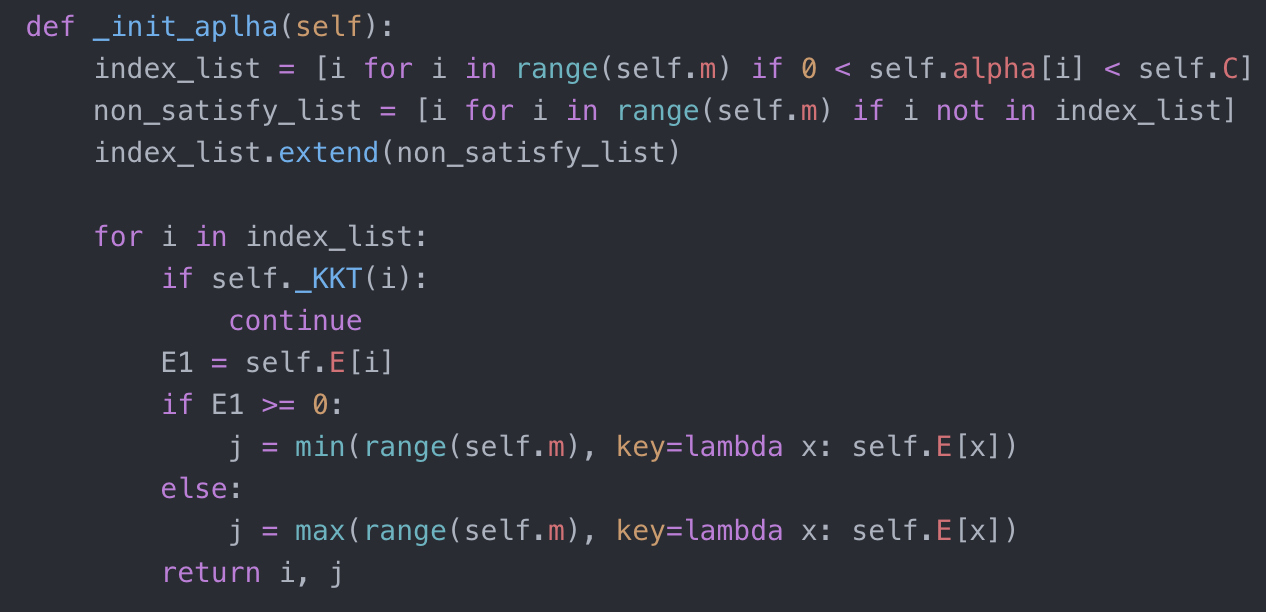
这个函数中更新的参数有：训练集中的数据量和特征维度，保存数据和标签，参数b初始化为0，alpha向量均初始化为1，E表示的是g(x)预测值和真实值的距离，将其初始化，C初始化为1.



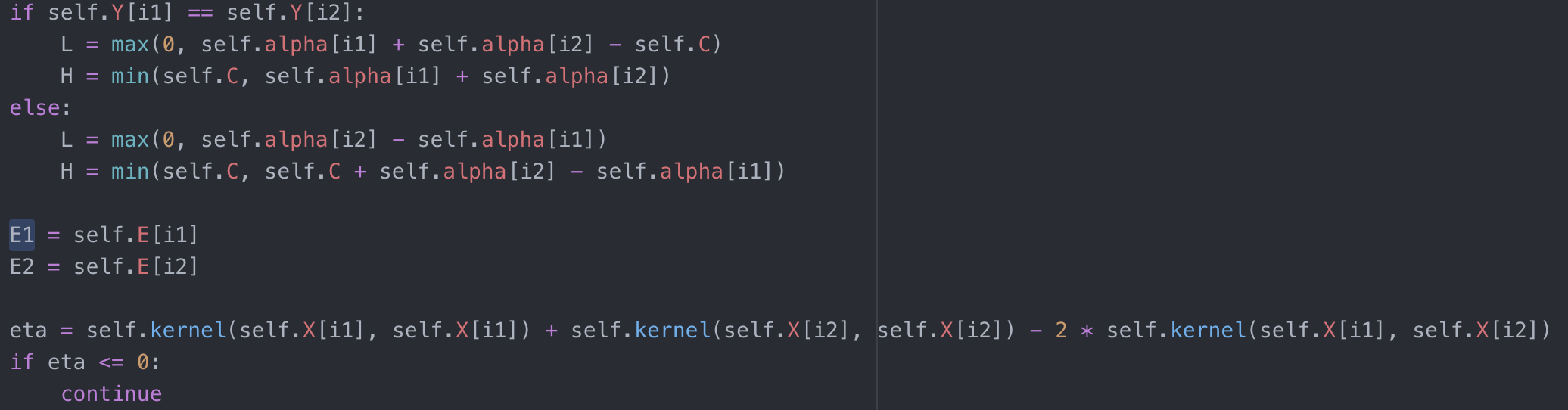
接下来开始迭代更新，最大迭代次数作为参数可以设置。每一轮迭代中先选出待更新的两个alpha的索引i1和i2



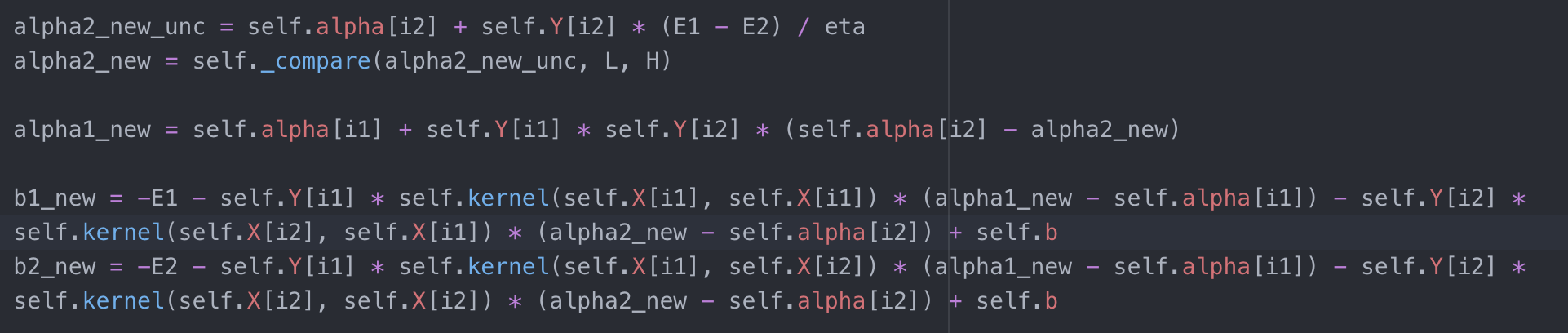
这个函数中，选取的策略是：先选一个不满足KKT条件的，再选出与之距离最远的，优先选择(0,C)之间的。



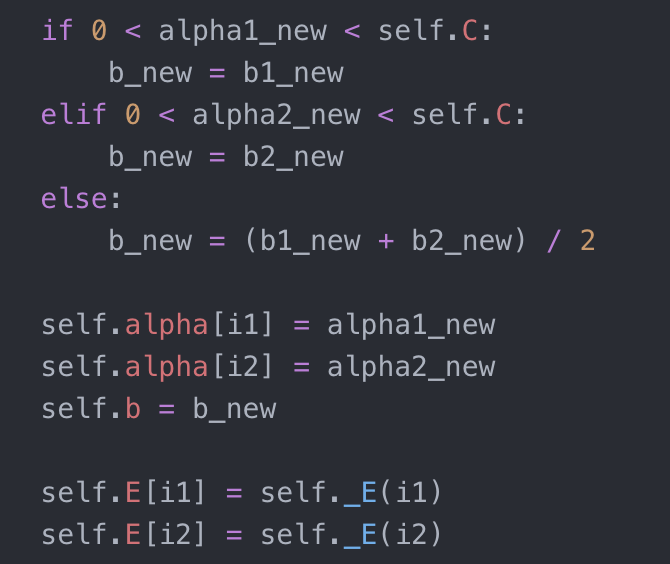
接下来计算alpha的上界和下界，调用核函数计算eta，从E向量中取出对应的分量。



预更新alpha1、alpha2和b

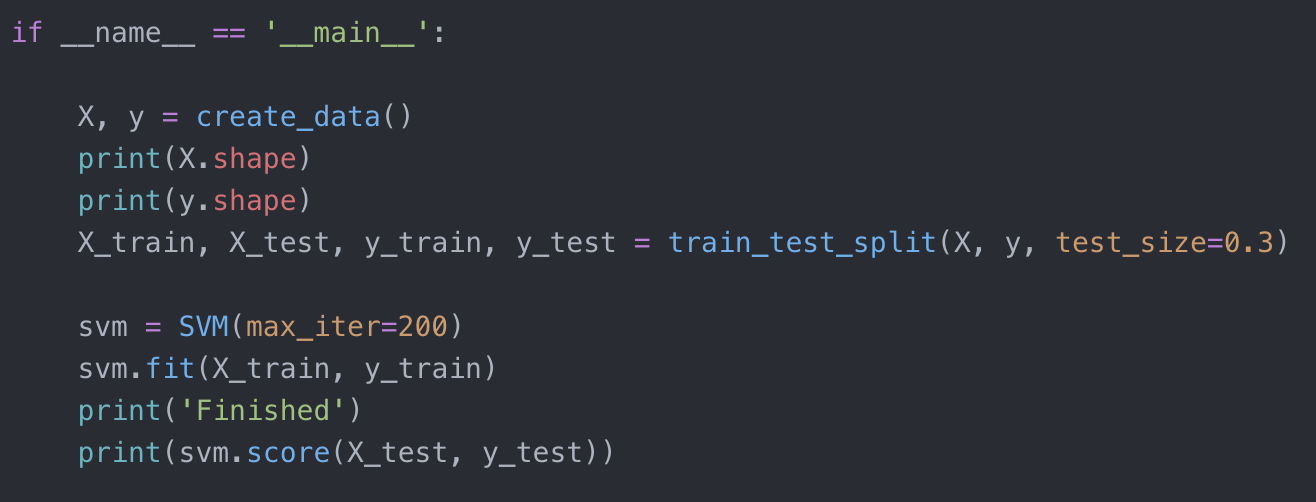


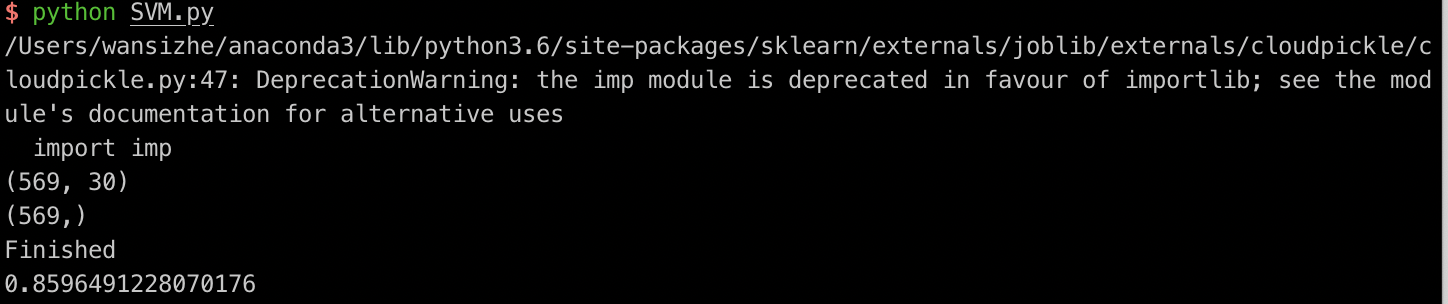
将预更新的参数更新到模型的参数中。

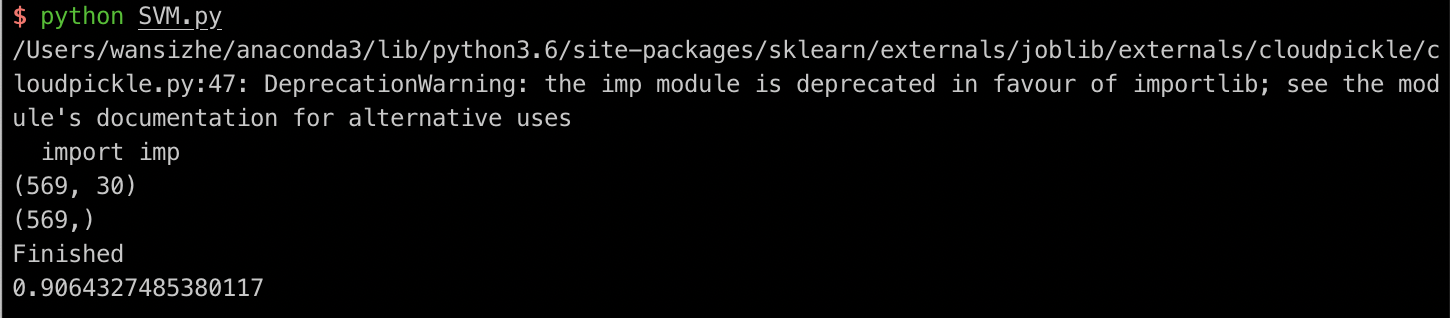


如果迭代没有结束，则继续。

最后加载乳腺癌数据集，用自己编写的SVM训练后，在测试集上的较好成绩如下图所示，远不及sklearn库中的SVC模型







#### 附件

#### 实验代码：SMO算法

1. import numpy as np
2. import pandas as pd
3. from sklearn.datasets import load\_breast\_cancer
4. from sklearn.model\_selection import train\_test\_split
5. from sklearn.svm import SVC
6. class SVM(object):
7. def \_\_init\_\_(self, max\_iter=100, kernel='linear'):
8. self.max\_iter = max\_iter
9. self.\_kernel = kernel
10. def \_g(self, i):
11. r = self.b
12. for j in range(self.m):
13. r += self.alpha[j] \* self.Y[j] \* self.kernel(self.X[i], self.X[j])
14. return r
15. def \_E(self, i):
16. return self.\_g(i) - self.Y[i]
17. def init\_args(self, features, labels):
18. self.m, self.n = features.shape
19. self.X = features
20. self.Y = labels
21. self.b = 0.0
22. self.alpha = np.ones(self.m)
23. self.E = [self.\_E(i) for i in range(self.m)]
24. self.C = 1.0
25. def \_KKT(self, i):
26. y\_g = self.\_g(i) \* self.Y[i]
27. if self.alpha[i] == 0:
28. return y\_g >= 1
29. elif 0 < self.alpha[i] < self.C:
30. return y\_g == 1
31. else:
32. return y\_g <= 1
33. def kernel(self, x1, x2):
34. if self.\_kernel == 'linear':
35. return sum([x1[k] \* x2[k] for k in range(self.n)])
36. elif self.\_kernel == 'poly':
37. return (sum([x1[k] \* x2[k] for k in range(self.n)]) + 1)\*\*2
38. else:
39. return 0
40. def \_init\_aplha(self):
41. index\_list = [i for i in range(self.m) if 0 < self.alpha[i] < self.C]
42. non\_satisfy\_list = [i for i in range(self.m) if i not in index\_list]
43. index\_list.extend(non\_satisfy\_list)
44. for i in index\_list:
45. if self.\_KKT(i):
46. continue
47. E1 = self.E[i]
48. if E1 >= 0:
49. j = min(range(self.m), key=lambda x: self.E[x])
50. else:
51. j = max(range(self.m), key=lambda x: self.E[x])
52. return i, j
53. def \_compare(self, \_alpha, L, H):
54. if \_alpha > H:
55. return H
56. elif \_alpha < L:
57. return L
58. else:
59. return \_alpha
60. def fit(self, features, labels):
61. self.init\_args(features, labels)
62. for t in range(self.max\_iter):
63. i1, i2 = self.\_init\_aplha()
64. if self.Y[i1] == self.Y[i2]:
65. L = max(0, self.alpha[i1] + self.alpha[i2] - self.C)
66. H = min(self.C, self.alpha[i1] + self.alpha[i2])
67. else:
68. L = max(0, self.alpha[i2] - self.alpha[i1])
69. H = min(self.C, self.C + self.alpha[i2] - self.alpha[i1])
70. E1 = self.E[i1]
71. E2 = self.E[i2]
72. eta = self.kernel(self.X[i1], self.X[i1]) + self.kernel(self.X[i2], self.X[i2]) - 2 \* self.kernel(self.X[i1], self.X[i2])
73. if eta <= 0:
74. continue
75. alpha2\_new\_unc = self.alpha[i2] + self.Y[i2] \* (E1 - E2) / eta
76. alpha2\_new = self.\_compare(alpha2\_new\_unc, L, H)
77. alpha1\_new = self.alpha[i1] + self.Y[i1] \* self.Y[i2] \* (self.alpha[i2] - alpha2\_new)
78. b1\_new = -E1 - self.Y[i1] \* self.kernel(self.X[i1], self.X[i1]) \* (alpha1\_new - self.alpha[i1]) - self.Y[i2] \* self.kernel(self.X[i2], self.X[i1]) \* (alpha2\_new - self.alpha[i2]) + self.b
79. b2\_new = -E2 - self.Y[i1] \* self.kernel(self.X[i1], self.X[i2]) \* (alpha1\_new - self.alpha[i1]) - self.Y[i2] \* self.kernel(self.X[i2], self.X[i2]) \* (alpha2\_new - self.alpha[i2]) + self.b
80. if 0 < alpha1\_new < self.C:
81. b\_new = b1\_new
82. elif 0 < alpha2\_new < self.C:
83. b\_new = b2\_new
84. else:
85. b\_new = (b1\_new + b2\_new) / 2
86. self.alpha[i1] = alpha1\_new
87. self.alpha[i2] = alpha2\_new
88. self.b = b\_new
89. self.E[i1] = self.\_E(i1)
90. self.E[i2] = self.\_E(i2)
91. return 'Finished'
92. def predict(self, data):
93. r = self.b
94. for i in range(self.m):
95. r += self.alpha[i] \* self.Y[i] \* self.kernel(data, self.X[i])
96. return 1 if r > 0 else -1
97. def score(self, X\_test, y\_test):
98. right\_count = 0
99. for i in range(len(X\_test)):
100. result = self.predict(X\_test[i])
101. if result == y\_test[i]:
102. right\_count += 1
103. return right\_count / len(X\_test)
104. def \_weight(self):
105. yx = self.Y.reshape(-1, 1) \* self.X
106. self.w = np.dot(yx.T, self.alpha)
107. return self.w
108. def create\_data():
109. breast\_cancer = load\_breast\_cancer()
110. data = breast\_cancer.data
111. labels = breast\_cancer.target
112. for i in range(len(labels)):
113. if labels[i] == 0:
114. labels[i] = -1
115. return data, labels
116. if \_\_name\_\_ == '\_\_main\_\_':
118. X, y = create\_data()
119. print(X.shape)
120. print(y.shape)
121. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)
122. svm = SVM(max\_iter=200)
123. svm.fit(X\_train, y\_train)
124. print('Finished')
125. print(svm.score(X\_test, y\_test))
126. # print('sklearn svc:')
127. # clf = SVC(kernel='linear')
128. # clf.fit(X\_train, y\_train)
129. # print(clf.score(X\_test, y\_test))