机器学习第五次作业

姓名：刘文晨 学号：202222280328

题目1

题面：

从网上下载或自己编程实现TSVM算法 选择两个UCI数据集，将其中30%的样例用作测试样本，10%的样例用作有标记样本，60%的样例用作无标记样本。分别训练出利用无标记样本的TSVM以及仅利用有标记样本的SVM，并比较其性能。

解：

选择最常用的iris数据集，将数据集标准化之后，将其中30个样例用作测试样本，10个样例用作有标记样本，60个样例用作无标记样本。以sklearn的SVM算法为基础建立TSVM。模型训练好后，输出经过有标记的样本训练后对未标记样本的预测正确率、经过TSVM训练后，对未标记样本的预测正确率和经过TSVM训练后对测试样本的预测正确率。最后绘制散点图和分别由SVM和TSVM得到的超平面。代码如下：

1. **import** matplotlib.pyplot as plt
2. **import** numpy as np
3. **import** pandas as pd
4. **from** sklearn **import** svm
5. **from** sklearn.datasets **import** load\_iris
6. **from** sklearn.preprocessing **import** MinMaxScaler

9. **def** create\_data():
10. iris = load\_iris()
11. df = pd.DataFrame(iris.data, columns=iris.feature\_names)
12. df['label'] = iris.target
13. df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'label']
14. data = np.array(df.iloc[:100, [0, 1, -1]])
15. # 对数据进行标准化处理
16. sc = MinMaxScaler()
17. sc.fit(data)
18. data = sc.transform(data)
19. data[:, -1] = data[:, -1] \* 2 - 1
20. # 30个测试样本
21. test = np.vstack((data[:15], data[50:65]))
22. # 10个有标记样本
23. labeled\_sample = np.vstack((data[15:20], data[65:70]))
24. # 60个无标记样本
25. unlabeled\_sample = np.vstack((data[20:50], data[70:]))
26. **return** test, labeled\_sample, unlabeled\_sample

29. test, labeled, unlabeled = create\_data()
30. clf = svm.SVC(C=1, kernel='linear')
31. # 有标记的样本训练SVM
32. clf.fit(labeled[:, :2], labeled[:, -1])
33. positive\_labeled = labeled[5:]
34. negative\_labeled = labeled[:5]
35. plt.scatter(labeled[:5, :2][:, 0], labeled[:5, :2][:, 1], color='red', s=40, label=-1)
36. plt.scatter(labeled[5:, :2][:, 0], labeled[5:, :2][:, 1], color='blue', s=40, label=1)
37. x\_points = np.linspace(0, 1, 10)
38. y\_points = -(clf.coef\_[0][0] \* x\_points + clf.intercept\_) / clf.coef\_[0][1]
39. plt.plot(x\_points, y\_points, color='green')
40. plt.legend()
41. # 伪标记
42. fake\_label = clf.predict(unlabeled[:, :2])
43. unlabeled\_positive\_x = []
44. unlabeled\_positive\_y = []
45. unlabeled\_negative\_x = []
46. unlabeled\_negative\_y = []
47. **for** i **in** range(len(unlabeled)):
48. **if** int(fake\_label[i]) == 1:
49. unlabeled\_positive\_x.append(unlabeled[i, 0])
50. unlabeled\_positive\_y.append(unlabeled[i, 1])
51. **else**:
52. unlabeled\_negative\_x.append(unlabeled[i, 0])
53. unlabeled\_negative\_y.append(unlabeled[i, 1])
55. plt.scatter(unlabeled\_positive\_x, unlabeled\_positive\_y, color='red', s=15)
56. plt.scatter(unlabeled\_negative\_x, unlabeled\_negative\_y, color='blue', s=15)
57. **print**('经过有标记的样本训练后，对未标记样本的预测正确率为{}'.format(clf.score(unlabeled[:, :2], unlabeled[:, -1])))
59. Cu = 0.1
60. Cl = 1  # 初始化Cu,Cl
61. weight = np.ones(len(labeled) + len(unlabeled))
62. # 样本权重
63. weight[len(unlabeled):] = Cu
64. # 用于训练有标记与无标记样本集合
65. train\_sample = np.vstack((labeled[:, :2], unlabeled[:, :2]))
66. # 用于训练的标记集合
67. train\_label = np.hstack((labeled[:, -1], fake\_label))
68. unlabeled\_id = np.arange(len(unlabeled))
70. **while** Cu < Cl:
71. clf.fit(train\_sample, train\_label, sample\_weight=weight)
72. **while** True:
73. # 通过训练得到的预测标记
74. predicted\_y = clf.decision\_function(unlabeled[:, :2])
75. # 伪标记，这里为与预测的区分开，写为real\_y
76. real\_y = fake\_label
77. epsilon = 1 - predicted\_y \* real\_y
78. positive\_set, positive\_id = epsilon[real\_y > 0], unlabeled\_id[real\_y > 0]
79. negative\_set, negative\_id = epsilon[real\_y < 0], unlabeled\_id[real\_y < 0]
80. positive\_max\_id = positive\_id[np.argmax(positive\_set)]
81. negative\_max\_id = negative\_id[np.argmax(negative\_set)]
82. epsilon1, epsilon2 = epsilon[positive\_max\_id], epsilon[negative\_max\_id]
83. **if** epsilon1 > 0 **and** epsilon2 > 0 **and** round(epsilon1 + epsilon2, 3) >= 2:
84. fake\_label[positive\_max\_id] = -fake\_label[positive\_max\_id]
85. fake\_label[negative\_max\_id] = -fake\_label[negative\_max\_id]
86. train\_label = np.hstack((labeled[:, -1], fake\_label))
87. clf.fit(train\_sample, train\_label, sample\_weight=weight)
88. **else**:
89. **break**
90. # 更新Cu
91. Cu = min(2 \* Cu, Cl)
92. # 更新样本权重
93. weight[len(unlabeled):] = Cu
94. # 绘图
95. x\_points = np.linspace(0, 1, 10)
96. y\_points = -(clf.coef\_[0][0] \* x\_points + clf.intercept\_) / clf.coef\_[0][1]
97. plt.plot(x\_points, y\_points, color='yellow')
98. plt.savefig('运行结果.jpg')
99. plt.show()
100. # 打印结果
101. **print**('经过TSVM训练后，对未标记样本的预测正确率为{}'.format(clf.score(unlabeled[:, :2], unlabeled[:, -1])))
102. **print**('经过TSVM训练后，对测试样本的预测正确率为{}'.format(clf.score(test[:, :2], test[:, -1])))

运行程序后，我们得到：经过有标记的样本训练后，对未标记样本的预测正确率为0.9666666666666667，经过TSVM训练后，对未标记样本的预测正确率为0.9833333333333333，经过TSVM训练后，对测试样本的预测正确率为1.0。

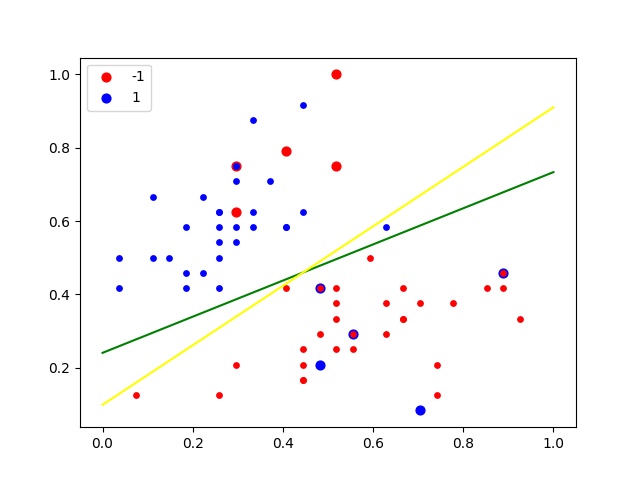


图1 散点图与超平面

程序绘制的散点图和超平面如图1所示。其中，绿线是仅利用有标记样本的SVM得到的超平面，黄线是利用无标记样本的TSVM模型得到的超平面。

由实验结果可知，对于iris数据集，TSVM通过利用未标记数据能提高最终分类的准确率，从SVM的96.67%提高到了TSVM的98.33%，并且预测标记与测试集的真实标记一致，预测正确率为100%。

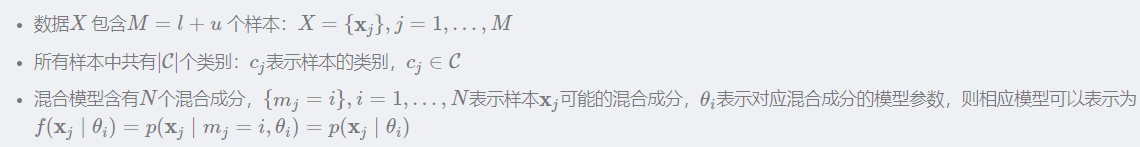
题目2

题面：

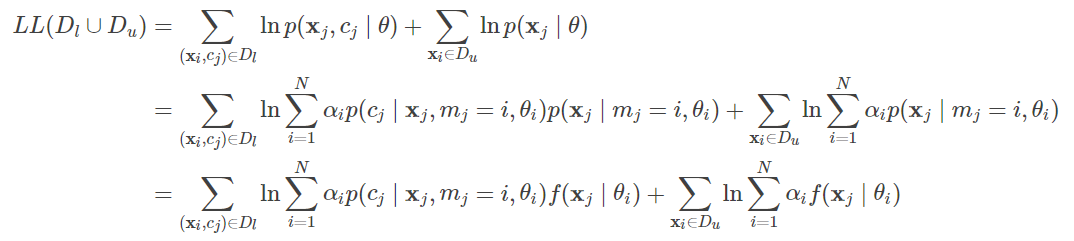
假设数据由混合专家（mixture of experts）模型生成，即数据是基于个成分混合而得的概率密度生成：，其中，是模型参数，是第个混合成分的概率密度，混合系数，。假设每个混合成分对应一个类别，但每个类别可能包含多个混合成分。试推导相应的生成式半监督学习算法。

解：

首先，我们假定：



在此处：

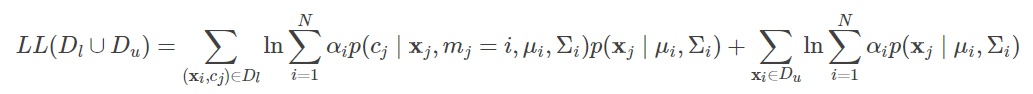


我们采用广义混合模型（The Generalized Mixture Model, GM），采用高斯分布作为混合成分，来推导EM算法的更新参数。

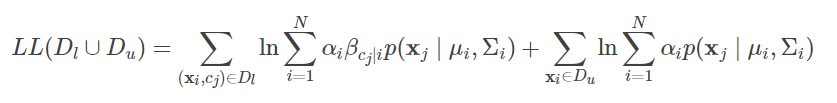
显然，此时：



则第一个式子变为：

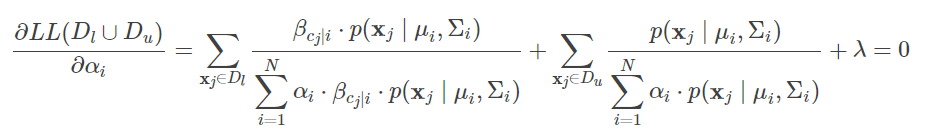


代入GM公式，得：

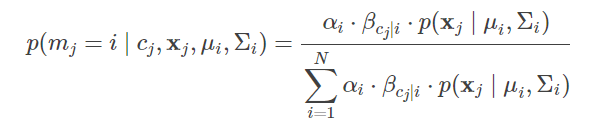


我们的目的是要求得最优的使上式取得最大值。

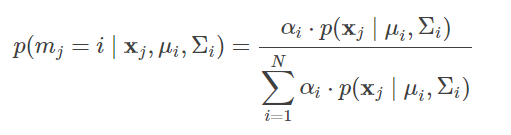
对于混合系数，除了要最大化，还应满足隐含条件：，，因此考虑对使用拉格朗日乘子法，变为优化：，代入的计算式，得到：

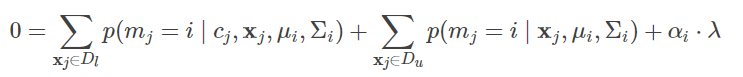


令：

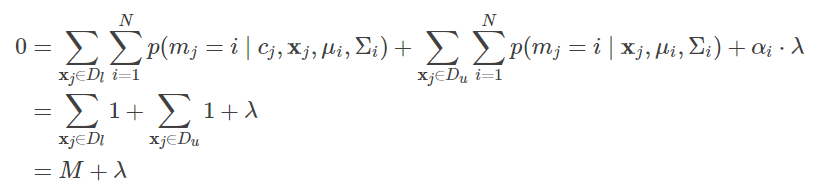


同时，将高斯模型代入方程，得：

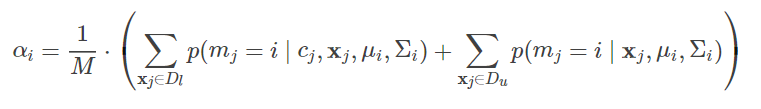




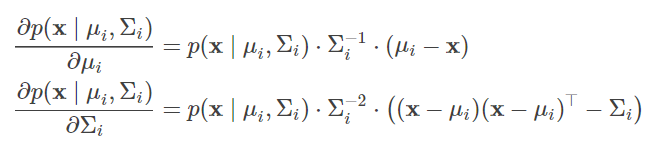
令上式对所有高斯混合成分求和：



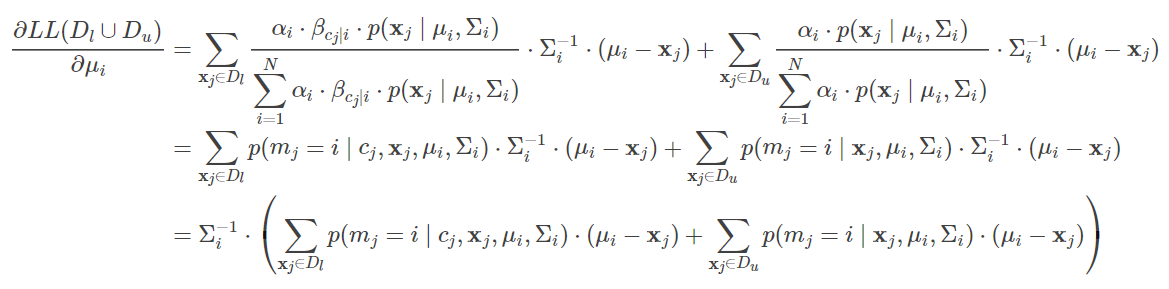
令，将其带入上式，得：



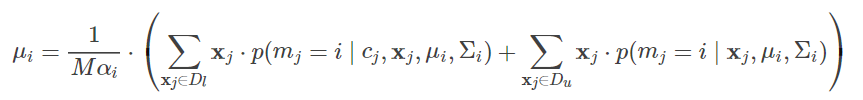
对于高斯分布，其偏导具有如下性质：

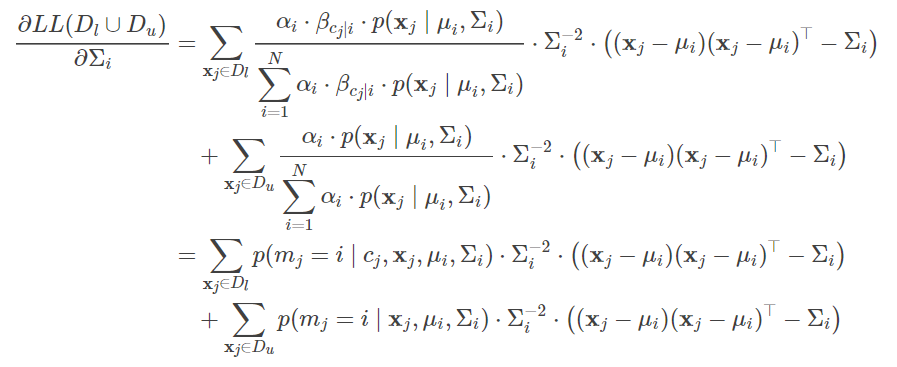


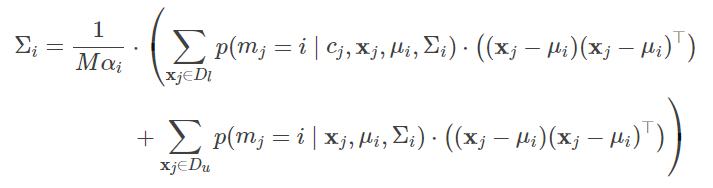
求偏导，得：



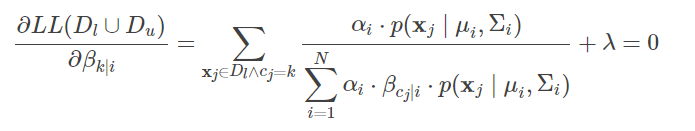
继续计算：

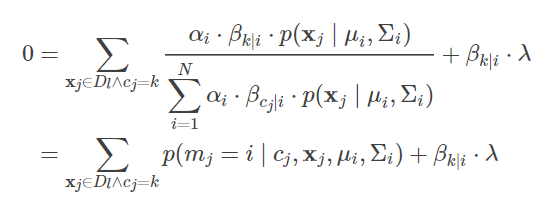


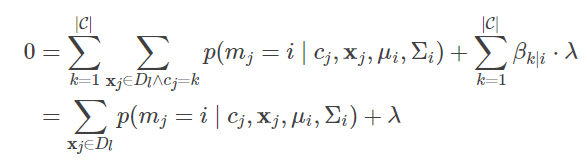


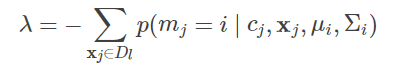


对于混合系数，除了要最大化，还应满足隐含条件：，，因此考虑对使用拉格朗日乘子法，变为优化：，代入的计算式，得到：









最终得到：

