

《人工智能与深度学习》课程

实验报告

(第 1 次实验)

学号：61522529

姓名：李勃璘

日期：2025 年 3 月 25 日

实验 1：支持向量机

一、实验目的

了解支持向量机 (Supported Vector Machine, SVM) 的基本概念及原理，了解核函数的基本原理以及作用。通过实验学会 SVM 在简单分类问题中的应用并通过 SVM 构建一个垃圾邮件分类系统。

二、实验内容

本实验使用 Python 语言 (Python 3.10.6) 完成，依赖的第三方库包括 Sklearn (一个强力的机器学习库)、Scipy (用于数学处理的软件包)、Numpy (一种开源的数值计算扩展库) 以及 nltk (自然语言处理工具包)。

本实验使用的数据集如表 1 所示。在 SVM 训练的实验中主要使用前三个训练集，而对于垃圾邮件分类，则需要训练集、测试集以及垃圾邮件与正常文件的样本。多维样本以 .mat 格式给出，文本格式的样本以 .txt 格式给出。

2.1 线性可分 SVM

对应代码：附录 A.1 的 part1 函数

数据集	意义
dataset_1.mat	线性可分 SVM 数据集
dataset_2.mat	非线性可分 SVM 数据集
dataset_3.mat	非线性可分 SVM 数据集
spamTrain.mat	垃圾邮件训练集
spamTest.mat	垃圾邮件测试集
vocab.txt	词典映射文件
emailSample1.txt	邮件样本 1
emailSample2.txt	邮件样本 2
spamSample1.txt	垃圾邮件样本 1
spamSample2.txt	垃圾邮件样本 2

表 1: 实验使用数据集

首先使用 `scipy` 库导入 `.mat` 类型的线性可分数据集 1，并使用 `matplotlib` 库绘图。绘图的结果如图 1 所示。

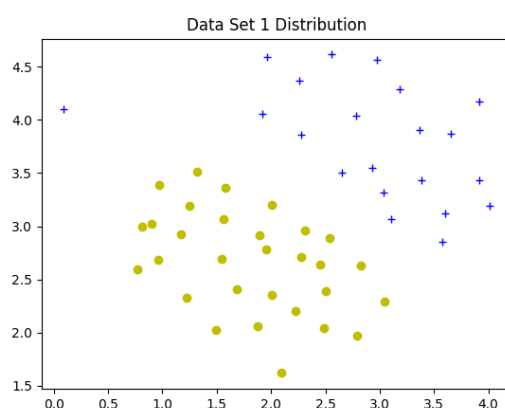


图 1: 数据集 1（线性可分）可视化

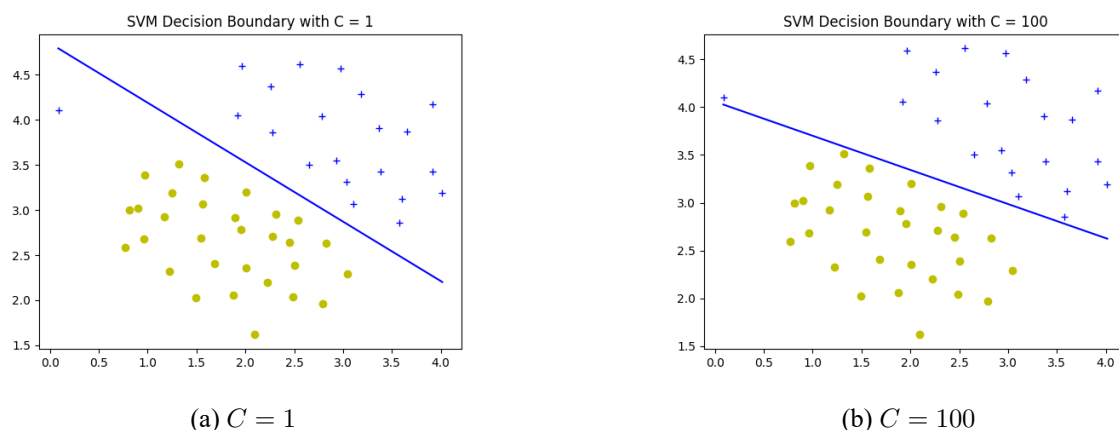
图表分析：显然可以看到两种类型的数据大体可以通过线性决策边界分开，但有少量用于测试模型性能的离群点。因此可称这种数据集为线性可分数据集。

分别调用 $C = 1$ 和 $C = 100$ 下的绘制 SVM 决策边界函数 (`visualize_boundary`)，得到不同 C 参数下数据集 1 的 SVM 决策边界如图 2。可以看到， $C = 100$ 时的决策边界包含了蓝色离群点，这是因为 $C = 100$ 时模型错误分类的惩罚权重更大，因此模型更倾向于包含离群点。

2.2 非线性可分 SVM

对应代码：附录 A.1 的 `part2` 函数

线性不可分指的是无法通过一个线性决策面来完美分隔数据集，这时候会出现分类错误。尽管线性可分的数据集分类方式简单，但这种数据集不具备普遍性。因此，需要找到一种方法处理非线性可分的数据集。

图 2: 不同 C 参数下数据集 1 的 SVM 决策边界

为了处理非线性可分的数据集，需要引入高斯核。它在 SVM 中也被称为径向基核函数（Radial Basis Function, RBF），定义如下：

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \quad (1)$$

其中， γ 是一个超参数，控制高斯核的影响范围。较大的 γ 使得模型更加关注局部信息，而较小的 γ 使得模型能够捕捉全局结构。

为了测试高斯核函数，使用它计算两个一维样本 $[1, 2, 1]$ 和 $[0, 4, -1]$ 的相似度。这段代码的输出结果为：Similarity between object 1 and object 2: 0.324652。（对应代码：附录 A.1 的 part2 函数，gaussian_kernel 函数）结果验证正确。

接下来训练非线性 SVM。首先绘制出非线性可分的数据集 2 如图 3。可以看到正负样本间存在非线性的边界。

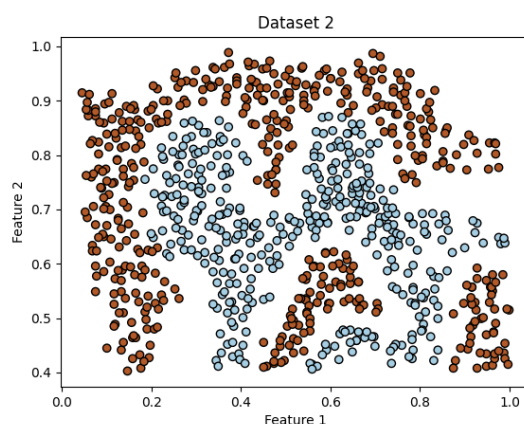


图 3: 数据集 2（线性不可分）可视化

使用 SVM 的 `fit` 方法训练 SVM，画出拟合决策边界如图 4。可以看到 SVM 基本正确地拟合了数据集 2 的非线性边界。

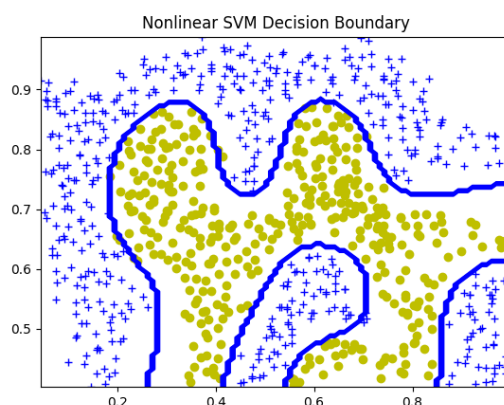


图 4: 数据集 2 的 SVM 决策边界

实验结果表明，高斯核方法能够有效处理非线性可分的数据集，并在较为复杂的数据结构中取得良好的分类效果。此外，超参数 γ 的选择对分类性能有重要影响，需要通过交叉验证或网格搜索进行优化。

2.3 最优参数搜索

对应代码：附录 A.1 的 `part3` 函数

在数据集 3 中，数据被划分为了训练集以及验证集。绘制训练集和验证集以观察数据（图 5）：

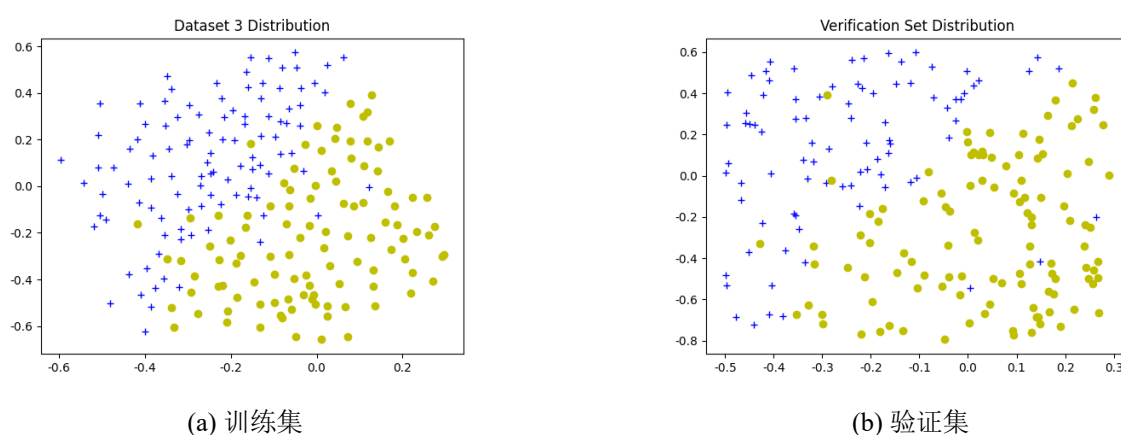


图 5: 数据集 3 的训练集和验证集

基于训练集训练的模型将在验证集上进行测试并根据验证集的反馈结果对模型参数进行微调。参数 C 和 σ 的搜索空间为 $[0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]$ 。确定最优参数组合后（对应代码：附录 A.1 的 `params_search` 函数）训练，得到的决策边界如图 6 所示。从图中可以看出 SVM 模型已经尽可能正负样本分开，但是还是存在少量样本未被彻底分开，如果要将其分开可能导致过拟合。

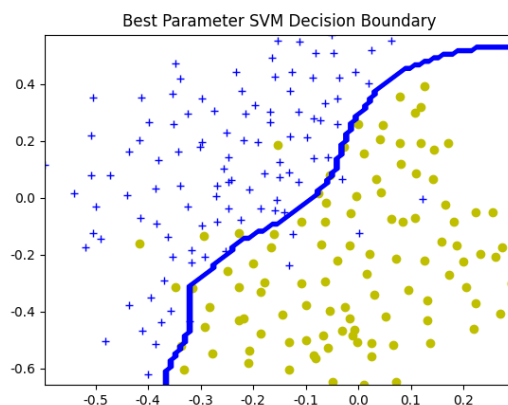


图 6: 数据集 3 的决策边界

三、提高训练

基于实验手册的内容, 尝试对现有实验做修改和调整, 如损失函数、激活函数等。

四、实验心得

在本次实验中, 我理解了支持向量机的基本工作流程以及使用 Python 训练 SVM 模型, 拟合线性、非线性边界的基本方法。并基于 SVM 完成了垃圾邮件分类识别的小实验。另外, 我还独立编写了 \LaTeX 模板 (SEU-AI-Report.cls), 用于本实验报告的撰写。在这个过程中也学到了不少关于 \LaTeX 的进阶知识。

参考文献

[1] 邱锡鹏神经网络与深度学习. 北京: 机械工业出版社, 2020. <https://nndl.github.io/>.

附录 A 实验代码

该部分列举本实验中使用的所有实验代码。

A.1 SVM 训练相关代码

```
1 # -*- coding: utf-8 -*-
2
3 import scipy.misc, scipy.io, scipy.optimize
4 from sklearn import svm
5 from sklearn import model_selection
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 plt.rcParams['axes.unicode_minus'] = False
10
11 def plot(data):
12     positives = data[data[:, 2] == 1]
13     negatives = data[data[:, 2] == 0]
14
15     plt.plot(positives[:, 0], positives[:, 1], 'b+')
16     plt.plot(negatives[:, 0], negatives[:, 1], 'yo')
17
18 # 绘制SVM决策边界
19 def visualize_boundary(X, trained_svm):
20     kernel = trained_svm.get_params()['kernel']
21     if kernel == 'linear':
22         w = trained_svm.coef_[0]
23         i = trained_svm.intercept_
24         xp = np.linspace(min(X[:, 0]), max(X[:, 0]), 100)
25         a = -w[0] / w[1]
26         b = i[0] / w[1]
27         yp = a * xp - b
28         plt.plot(xp, yp, 'b-')
29     elif kernel == 'rbf':
30         x1plot = np.linspace(min(X[:, 0]), max(X[:, 0]), 100)
31         x2plot = np.linspace(min(X[:, 1]), max(X[:, 1]), 100)
32
33         X1, X2 = np.meshgrid(x1plot, x2plot)
34         vals = np.zeros(np.shape(X1))
35
```

```
36         for i in range(0, np.shape(X1)[1]):
37             this_X = np.c_[X1[:, i], X2[:, i]]
38             vals[:, i] = trained_svm.predict(this_X)
39
40         plt.contour(X1, X2, vals, colors='blue')
41
42 def gaussian_kernel(x1, x2, sigma):
43     return np.exp(-np.sum((x1 - x2) ** 2) / (2 * sigma ** 2))
44
45 def part1():
46     mat = scipy.io.loadmat("dataset_1.mat")
47     X, y = mat['X'], mat['y']
48
49     plt.title('Data Set 1 Distribution')
50     plot(np.c_[X, y])
51     plt.show(block=True)
52
53     linear_svm = svm.SVC(C=1, kernel='linear')
54     linear_svm.fit(X, y.ravel())
55
56     plt.title('SVM Decision Boundary with C = 1')
57     plot(np.c_[X, y])
58     visualize_boundary(X, linear_svm)
59     plt.show(block=True)
60
61     linear_svm = svm.SVC(C=100, kernel='linear')
62     linear_svm.fit(X, y.ravel())
63
64     plt.title('SVM Decision Boundary with C = 100')
65     plot(np.c_[X, y])
66     visualize_boundary(X, linear_svm)
67     plt.show(block=True)
68
69 def part2():
70     x1 = np.array([1, 2, 1])
71     x2 = np.array([0, 4, -1])
72     sigma = 2
73     print("Similarity between object 1 and object 2: %f" %
74           gaussian_kernel(x1, x2, sigma))
75
76     mat_data = scipy.io.loadmat("dataset_2.mat")
```

```
76 X = mat_data['X']
77 y = mat_data['y'].ravel()
78
79 plt.figure()
80 plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired, edgecolors
81 = 'k')
82 plt.xlabel("Feature 1")
83 plt.ylabel("Feature 2")
84 plt.title("Dataset 2")
85 plt.show()
86
87 sigma = 0.1
88 rbf_svm = svm.SVC(C=1, kernel='rbf', gamma=1.0 / (2 * sigma ** 2)
89 )
90 rbf_svm.fit(X, y)
91
92 plt.title('Nonlinear SVM Decision Boundary')
93 plot(np.c_[X, y])
94 visualize_boundary(X, rbf_svm)
95 plt.show(block=True)
96
97 def part3():
98     mat = scipy.io.loadmat("dataset_3.mat")
99     X, y = mat['X'], mat['y']
100     X_val, y_val = mat['Xval'], mat['yval']
101
102     plt.title('Dataset 3 Distribution')
103     plot(np.c_[X, y])
104     plt.show(block=True)
105
106     plt.title('Verification Set Distribution')
107     plot(np.c_[X_val, y_val])
108     plt.show(block=True)
109
110     best = params_search(X, y, X_val, y_val)
111     rbf_svm = svm.SVC(C=best['C'], kernel='rbf', gamma=best['gamma'])
112     rbf_svm.fit(X, y.ravel())
113
114     plt.title('Best Parameter SVM Decision Boundary')
115     plot(np.c_[X, y])
116     visualize_boundary(X, rbf_svm)
```



```
115     plt.show(block=True)
116
117 def params_search(X, y, X_val, y_val):
118     c_values = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
119     sigma_values = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
120
121     best = {'error': 999, 'C': 0.0, 'sigma': 0.0}
122
123     for C in c_values:
124         for sigma in sigma_values:
125             rbf_svm = svm.SVC(C=C, kernel='rbf', gamma=1.0 / (2 *
126 sigma ** 2))
127             rbf_svm.fit(X, y.ravel())
128             error = 1 - rbf_svm.score(X_val, y_val)
129
130             if error < best['error']:
131                 best['error'] = error
132                 best['C'] = C
133                 best['sigma'] = sigma
134
135     best['gamma'] = 1.0 / (2 * best['sigma'] ** 2)
136     return best
137
138 def main():
139     np.set_printoptions(precision=6, linewidth=200)
140     part1()
141     part2()
142     part3()
143
144 if __name__ == '__main__':
145     main()
```

程序 1: ex3.py

A.2 垃圾邮件分类相关代码

```
1 # -*- coding: utf-8 -*-
2
3 import csv
4 import re
5 import pickle
```

```
6
7 import numpy as np
8 import nltk, nltk.stem.porter
9 import scipy.misc, scipy.io, scipy.optimize
10 from sklearn import svm
11
12
13 def vocaburary_mapping():
14     vocab_list = {}
15     with open('vocab.txt', 'r') as file:
16         reader = csv.reader(file, delimiter='\t')
17         for row in reader:
18             vocab_list[row[1]] = int(row[0])
19
20     return vocab_list
21
22 def feature_extraction(word_indices):
23     features = np.zeros((1899, 1))
24     for index in word_indices:
25         features[index] = 1
26     return features
27
28
29 def email_preprocess(email):
30     with open(email, 'r') as f:
31         email_contents = f.read()
32     vocab_list = vocaburary_mapping()
33     word_indices = []
34     email_contents = email_contents.lower()
35     email_contents = re.sub('<[^>]+>', ' ', email_contents)
36     email_contents = re.sub('[0-9]+', 'number', email_contents)
37     email_contents = re.sub('(http|https)://[^\s]*', 'httpaddr',
email_contents)
38     email_contents = re.sub('[^\s]+@[^\s]+', 'emailaddr',
email_contents)
39     email_contents = re.sub('[\$]+', 'dollar', email_contents)
40     stemmer = nltk.stem.porter.PorterStemmer()
41     tokens = re.split('[ ' + re.escape("@$/#.-:&*+=[]?!(){},'\">_<;%\\
n") + ']', email_contents)
42
43     for token in tokens:
```

```
44     token = re.sub('[^a-zA-Z0-9]', '', token)
45     token = stemmer.stem(token.strip())
46
47     if len(token) == 0:
48         continue
49
50     if token in vocab_list:
51         word_indices.append(vocab_list[token])
52
53     return word_indices, ' '.join(tokens)
54
55
56
57 # 预处理
58 def part_1():
59     word_indices, processed_contents = email_preprocess('emailSample1
60     .txt')
61     print(word_indices)
62     print(processed_contents)
63
64 # 特征提取
65 def part_2():
66     word_indices, processed_contents = email_preprocess('emailSample1
67     .txt')
68     features = feature_extraction(word_indices)
69     print(features)
70
71 # SVM模型训练
72 def part_3():
73     #加载训练集
74     mat = scipy.io.loadmat("spamTrain.mat")
75     X, y = mat['X'], mat['y']
76
77     # linear_svm = pickle.load(open("linear_svm.svm", "rb")) # 模型加
78     载
79     #训练SVM
80     linear_svm = svm.SVC(C=0.1, kernel='linear')
81     linear_svm.fit(X, y.ravel())
82     # pickle.dump(linear_svm, open("linear_svm.svm", "wb")) # 模型保
```

存

```
82 # 预测并计算训练集正确率
83 predictions = linear_svm.predict(X)
84 predictions = predictions.reshape(np.shape(predictions)[0], 1)
85 print('{}%'.format((predictions == y).mean() * 100.0))
86 # 加载测试集
87 mat = scipy.io.loadmat("spamTest.mat")
88 X_test, y_test = mat['Xtest'], mat['ytest']
89 # 预测并计算测试集正确率
90 predictions = linear_svm.predict(X_test)
91 predictions = predictions.reshape(np.shape(predictions)[0], 1)
92 print('{}%'.format((predictions == y_test).mean() * 100.0))
93
94 vocab_list = vocaburary_mapping()
95 reversed_vocab_list = dict((v, k) for (k, v) in vocab_list.items
96 ())
97
98 sorted_indices = np.argsort(linear_svm.coef_, axis=None)
99
100 for i in sorted_indices[0:15]:
101     print(reversed_vocab_list[i])
102
103 def part_4():
104
105     word_indices, processed_contents = email_preprocess('spamSample2.
106 txt')
107     features = feature_extraction(word_indices)
108     linear_svm = svm.SVC(C=0.1, kernel='linear')
109     linear_svm = pickle.load(open("linear_svm.svm", "rb"))
110     # Prediction
111     prediction = linear_svm.predict(features.T)
112     print(prediction)
113     # Output
114     if prediction == 1:
115         print("邮件被分类为垃圾邮件。")
116     else:
117         print("邮件被分类为非垃圾邮件。")
118
119 part_1()
120 part_2()
121 part_3()
```

```
120 part_4()  
121 # print(vocaburary_mapping())
```

程序 2: ex3_spam.py