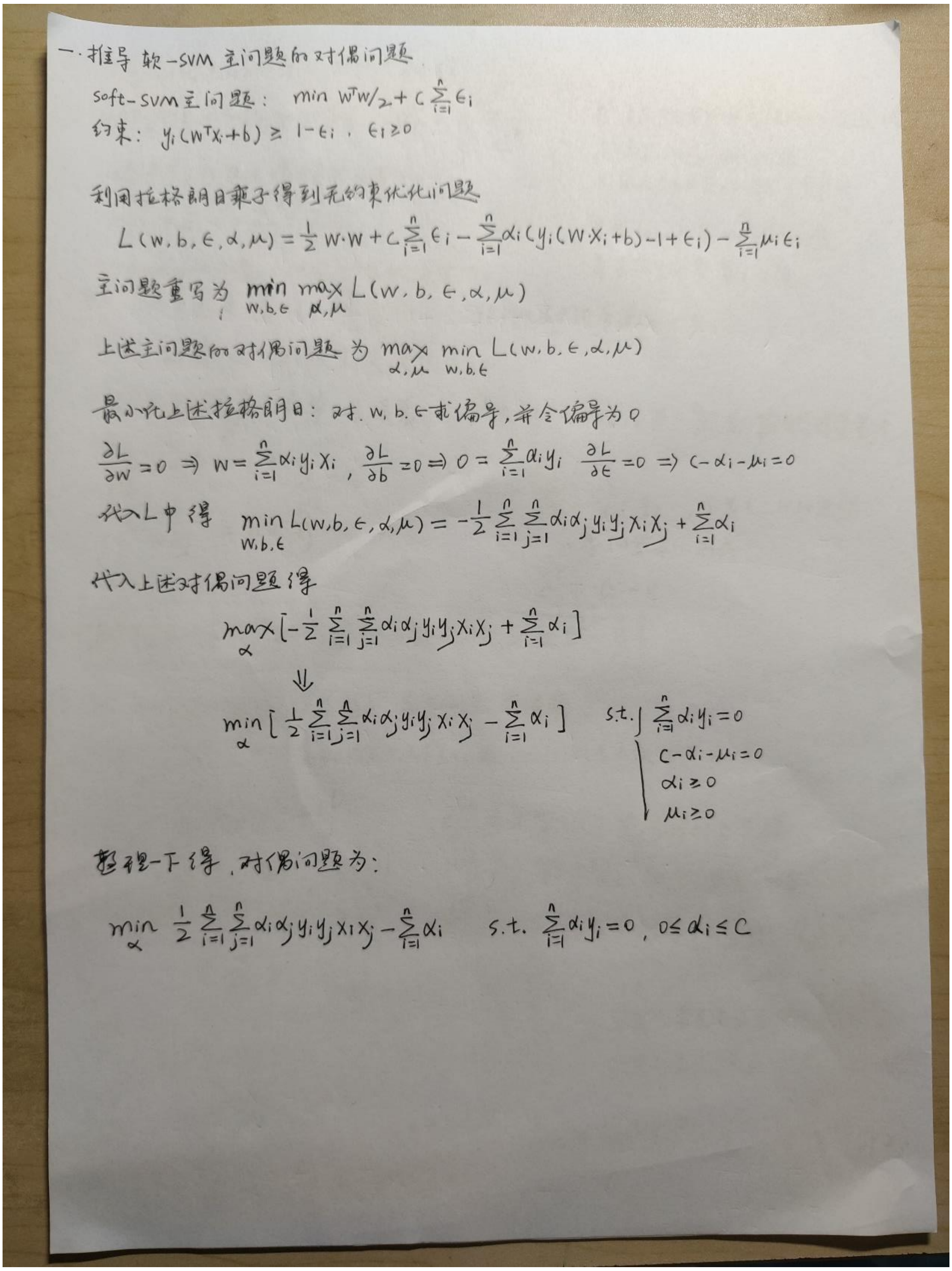
第三章

一、推导软-SVM主问题的对偶问题



一. 推导 软-SVM 主问题的对偶问题

soft-SVM主问题： $\min w^Tw/2+C\sum_{i=1}^{n}\epsilon_i$

约束： $y_i(w^Tx_i+b)\geq 1-\epsilon_i\ ,\ \epsilon_i\geq 0$

利用拉格朗日乘子得到无约束优化问题

$$L(w,b,\epsilon,\alpha,\mu)=\frac{1}{2}w\cdot w+C\sum_{i=1}^{n}\epsilon_i-\sum_{i=1}^{n}\alpha_i(y_i(w\cdot x_i+b)-1+\epsilon_i)-\sum_{i=1}^{n}\mu_i\epsilon_i$$

主问题重写为 $\min\limits_{w,b,\epsilon}\max\limits_{\alpha,\mu}L(w,b,\epsilon,\alpha,\mu)$

上述主问题的对偶问题为 $\max\limits_{\alpha,\mu}\min\limits_{w,b,\epsilon}L(w,b,\epsilon,\alpha,\mu)$

最小化上述拉格朗日：对 w, b, ε 求偏导，并令偏导为0

$$\frac{\partial L}{\partial w}=0\Rightarrow w=\sum_{i=1}^{n}\alpha_i y_i x_i\ ,\ \frac{\partial L}{\partial b}=0\Rightarrow 0=\sum_{i=1}^{n}\alpha_i y_i\quad \frac{\partial L}{\partial \epsilon}=0\Rightarrow C-\alpha_i-\mu_i=0$$

代入L中得 $\min\limits_{w,b,\epsilon}L(w,b,\epsilon,\alpha,\mu)=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i x_j+\sum_{i=1}^{n}\alpha_i$

代入上述对偶问题得

$$\max_{\alpha}\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i x_j+\sum_{i=1}^{n}\alpha_i\right]$$

$$\Downarrow$$

$$\min_{\alpha}\left[\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i x_j-\sum_{i=1}^{n}\alpha_i\right]\quad s.t.\begin{cases}\sum_{i=1}^{n}\alpha_i y_i=0\\ C-\alpha_i-\mu_i=0\\ \alpha_i\geq 0\\ \mu_i\geq 0\end{cases}$$

整理一下得，对偶问题为：

$$\min_{\alpha}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i x_j-\sum_{i=1}^{n}\alpha_i\quad s.t.\ \sum_{i=1}^{n}\alpha_i y_i=0\ ,\ 0\leq\alpha_i\leq C$$

二、垃圾邮件分类

1、代码解析

（1）加载训练集和测试集数据

```
def load_data():
 # 加载 mat 格式的字典文件
 spam_train = loadmat(file_name="spamTrain.mat")
 print(spam_train.keys())
 spam_train_x = spam_train["X"]
 spam_train_y = spam_train["y"]
 # 一个数据的长度是 1899，也就是说垃圾邮件一共有 1899 个特征
 spam_train_y = [math.pow(-1, i+1) for i in spam_train_y]
 spam_train_y = np.array(spam_train_y, dtype=int).reshape(-1, 1)
 # print(spam_train_y)
 # 同样的方式对测试集进行处理
 spam_test = loadmat(file_name="spamTest.mat")
 print(spam_test.keys())
 spam_test_x = spam_test["Xtest"]
 spam_test_y = spam_test["ytest"]
 spam_test_y = [math.pow(-1, i + 1) for i in spam_test_y]
 spam_test_y = np.array(spam_test_y, dtype=int).reshape(-1, 1)
 for x in spam_train_x:
     print("训练集特征长度:{}".format(len(x)))
     break
 for x in spam_test_x:
     print("测试集特征长度:{}".format(len(x)))
     break
 print("训练集样本数量:{}".format(spam_train_y.shape[0]))
 print("测试集样本数量:{}".format(spam_test_y.shape[0]))
 return spam_train_x, spam_train_y, spam_test_x, spam_test_y
```

（2）批量 Pegasos 算法，参数分别是数据，数据标签，C=0.1，训练轮数，batch 大小

```
# 批量 Pegasos 算法，参数分别是数据，数据标签，C=0.1，训练轮数，batch 大小
def batchPegasos(x, y, C, T, k):
    lam = 1 / (k * C)
    m, n = np.shape(x)
    w = np.zeros(n)
    dataIndex = np.array([i for i in range(m)])
    for t in range(1, T + 1):
        wDelta = np.zeros(n)
        eta = 1.0 / (lam * t)
        np.random.shuffle(dataIndex)
        for j in range(k):
            i = dataIndex[j]
            p = predict(w, x[i, :])
            if y[i][0] * p < 1:
                wDelta += y[i] * x[i, :]
        w = (1.0 - 1 / t) * w + (eta / k) * wDelta
    return w


# 预测 wx+b
def predict(w, x):
    return w.T @ x
```

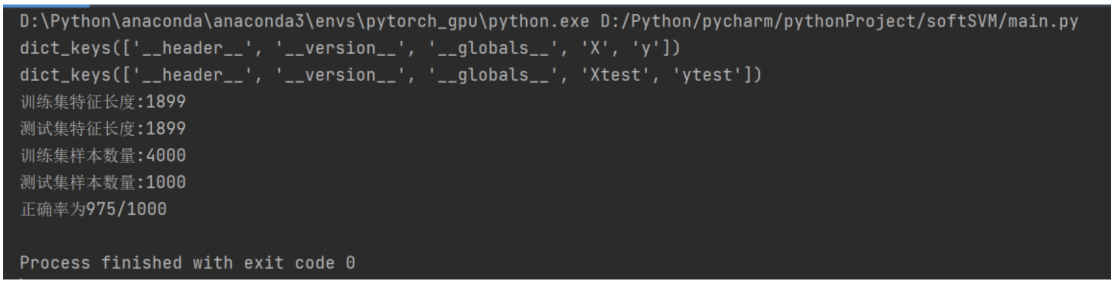
（3）对测试集进行测试

```
def test(x, y, w):
 predict_y = []
 label_y = y.reshape(-1)
 # print(label_y)
 for x_i, y_i in zip(x, label_y):
     tmp = predict(w, x_i)
     if tmp <= 0:
         predict_y.append(-1)
     else:
         predict_y.append(1)
 predict_y = np.asarray(predict_y)
 # print(np.sum(predict_y == label_y))
 print("正确率为{}/{}".format(np.sum(predict_y == label_y), len(predict_y)))
```

（4）主函数

```
if __name__ == '__main__':
 spam_train_x, spam_train_y, spam_test_x, spam_test_y = load_data()
 # 训练
 c = 0.1
 epochs = 100
 batch_size = 100
 w = batchPegasos(spam_train_x, spam_train_y, c, epochs, batch_size)
 # 测试
 test(spam_test_x, spam_test_y, w)
```

2、实验结果



```
D:\Python\anaconda\anaconda3\envs\pytorch_gpu\python.exe D:/Python/pycharm/pythonProject/softSVM/main.py
dict_keys(['__header__', '__version__', '__globals__', 'X', 'y'])
dict_keys(['__header__', '__version__', '__globals__', 'Xtest', 'ytest'])
训练集特征长度:1899
测试集特征长度:1899
训练集样本数量:4000
测试集样本数量:1000
正确率为975/1000

Process finished with exit code 0
```