实验二 SVM 支持向量机

- PB20111635
- 蒋磊

实验目的

- 熟悉SVM的原理并实现
- 了解机器学习及模型学习及评估的基本方法

实验原理

SVM模型

支持向量机是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器。

间隔可以使用 $\gamma = \frac{2}{||w||}$ 来表示，这样求解 `SVM` 模型可以变成下面的优化问题：

$$\begin{gathered}\max_{w,b} \frac{2}{\|\boldsymbol{w}\|} \\ \text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b) \geq 1 \quad i=1,\ldots,m\end{gathered} \tag{2}$$

等价于：

$$\begin{gathered}\min_{w,b} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b) \geq 1 \quad i=1,\ldots,m\end{gathered} \tag{3}$$

上面的模型只能解决线性可分的问题，为了解决有部分数据点线性不可分的情况，需要增加软间隔， 软间隔允许某些样本不满足约束 $y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b) \geq 1$. 为了使不满足约束的样本尽可能少, 优化目标可以写为

$$\min_{w,b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} l_{0/1}(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)-1) \tag{4}$$

其中 $C > 0$ 是一个常数，$l_{0/1}$ 是 “0/1损失函数”

$$l_{0/1} = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

由于 $l_{0/1}$ 非凸、非连续，数学性质不太好，下面使用如下的 hinge loss 函数来替代它：

$$l_{\text{hinge}}(z) = \max(0, 1 - z) \tag{6}$$

如此， 优化问题变成：

$$\min_{\boldsymbol{w}, b} \ \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{m} \max(0, 1 - y_i(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b)) \tag{7}$$

模型学习方法

（1）梯度下降法

为了求解模型中的参数 $\boldsymbol{w}$ 和 b，我们可以使用**梯度下降法**.

记要优化的式子为 L, 记 $\xi_i = 1 - y_i(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b)$, 则

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{w}} &= \boldsymbol{w} - C \sum_{\xi_i \geq 0} y_i \boldsymbol{x}_i \\ \frac{\partial L}{\partial b} &= -C \sum_{\xi_i \geq 0} y_i \end{aligned} \tag{8}$$

梯度下降法：

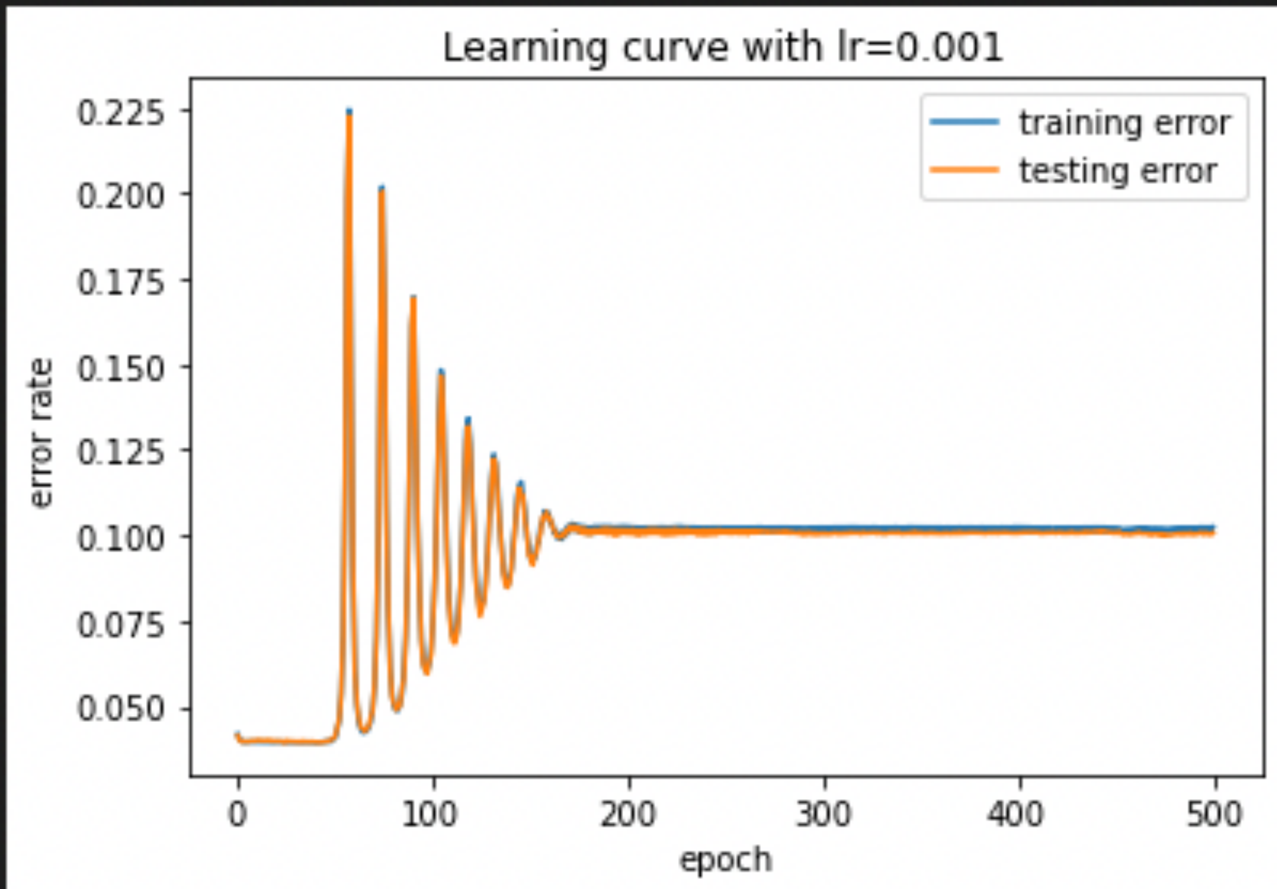
$$\begin{aligned} & \text{while} \, \| \frac{\partial L}{\partial \boldsymbol{w}} \| + \| \frac{\partial L}{\partial b} \| > \delta \, \text{do} \\ & \quad \text{for } i = 1 \text{ to } m \text{ do} \\ & \quad \quad \xi_i \leftarrow 1 - y_i(\boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{x}_i + b_t) \\ & \quad \boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta(\boldsymbol{w}_t - C \sum_{\xi_i \geq 0} y_i \boldsymbol{x}_i) \\ & \quad b_{t+1} \leftarrow b_t - \eta(-C \sum_{\xi_i \geq 0} y_i) \\ & \text{end while} \end{aligned} \tag{9}$$

梯度下降法的实验结果

在梯度下降法中，我生成的数据是 20 维，100000 组数据，其中 99000 组数据作为训练集，剩下 10000 组作为测试集，学习率为 0.001，最大迭代次数为 500（经过测试模型大概在跑完 300 个 epoch 后收敛），惩罚系数为 1，初始生成数据的错标率我没有改动，也就是助教设置的 4%左右。

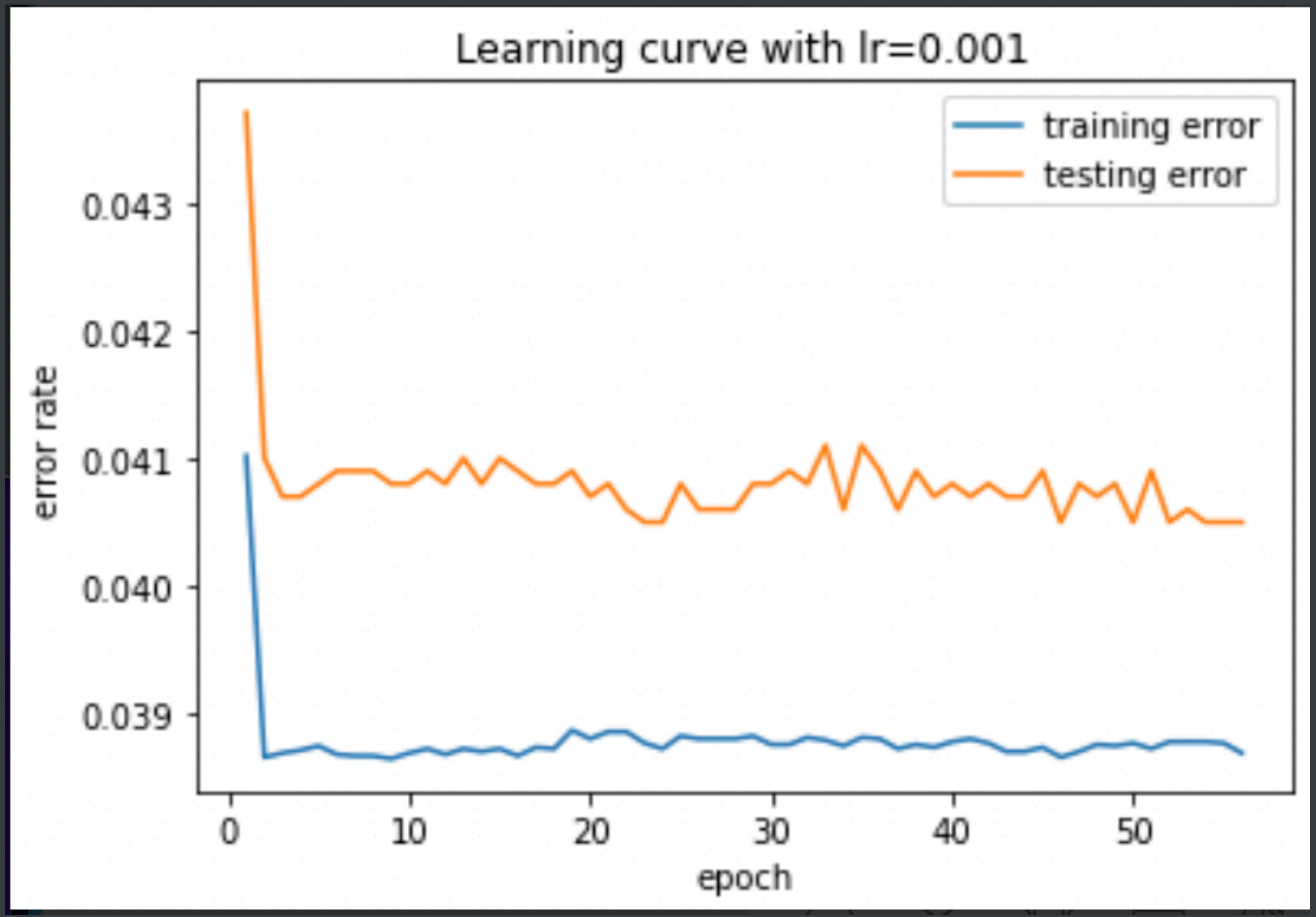
```
X_data, y_data, mislabel = generate_data(20, 100000)

# split data
X_train = X_data[0:90000]
y_train = y_data[0:90000]
X_test = X_data[90001:]
y_test = y_data[90001:]

# constrcut model and train (remember record time)
model1 = SVM1(20, learning_rate=0.001, max_iter=500, C=1)
model1.fit(X_train, y_train, val_data=(X_test, y_test))
```

```
已经490个epoch  train acc = 0.898848
train acc = 0.8976161616161616
test acc = 0.8992899289928993
错误率曲线如下:
```



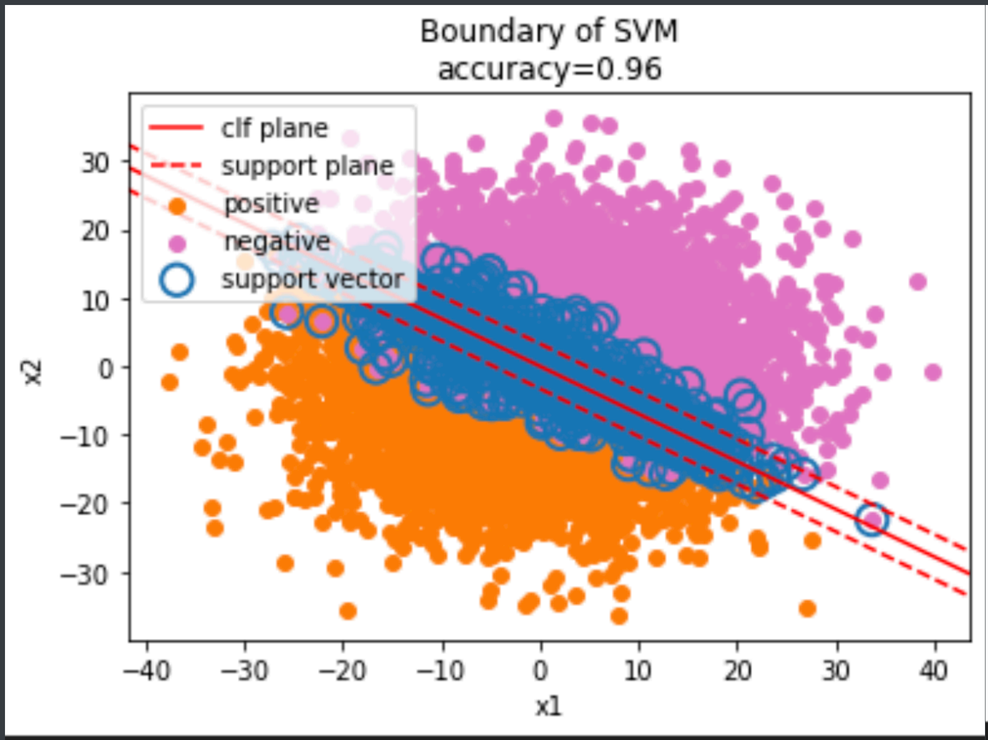
如图所示，最终在此参数条件下，模型的训练集与测试集均可以达到大约 90% 的正确率。

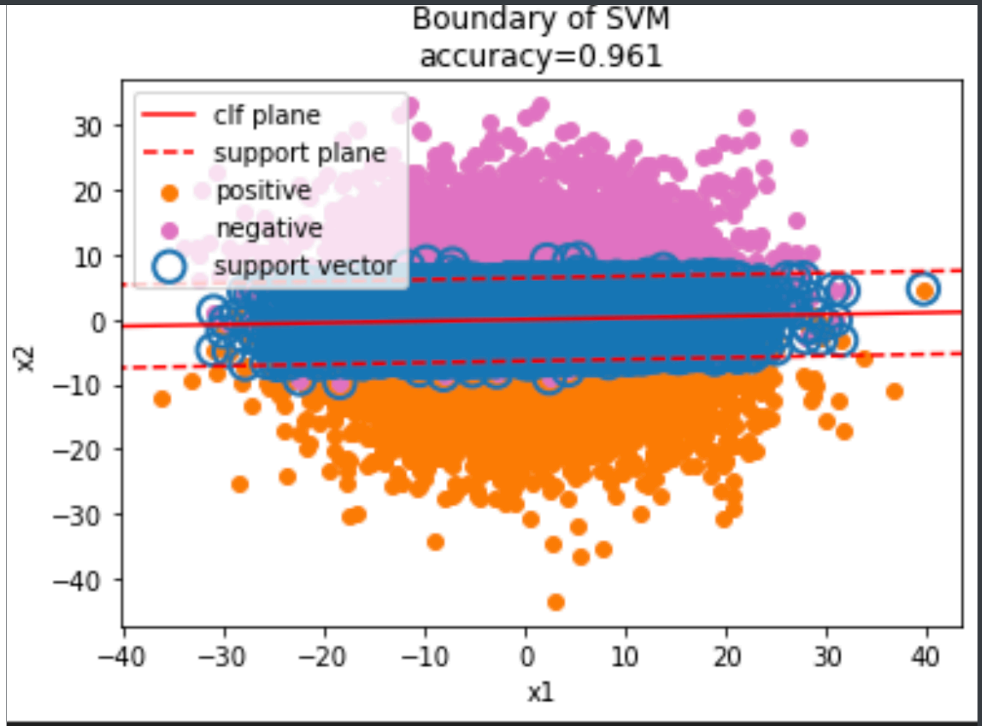
但由于初始错误率较低，后不断振荡最终收敛，这里有过拟合的的风险，于是我将参数进行了一些修改，由于是采取了软间隔的模型，最初的惩罚系数 C=1，我认为设置的有些过大了，这将导致间隔较小，所以我将惩罚系数 C 改为 0.001，故意将间隔放大，以容忍更多的错误，不过这样调参的合理性仍有待考证，最终能够得到这样的结果：

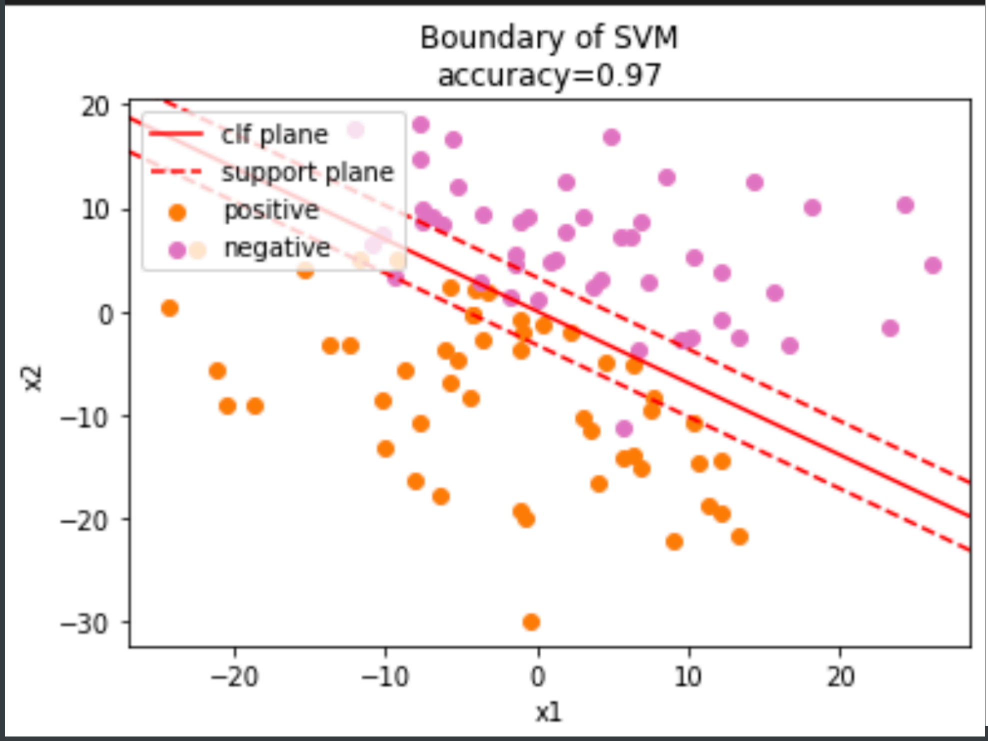


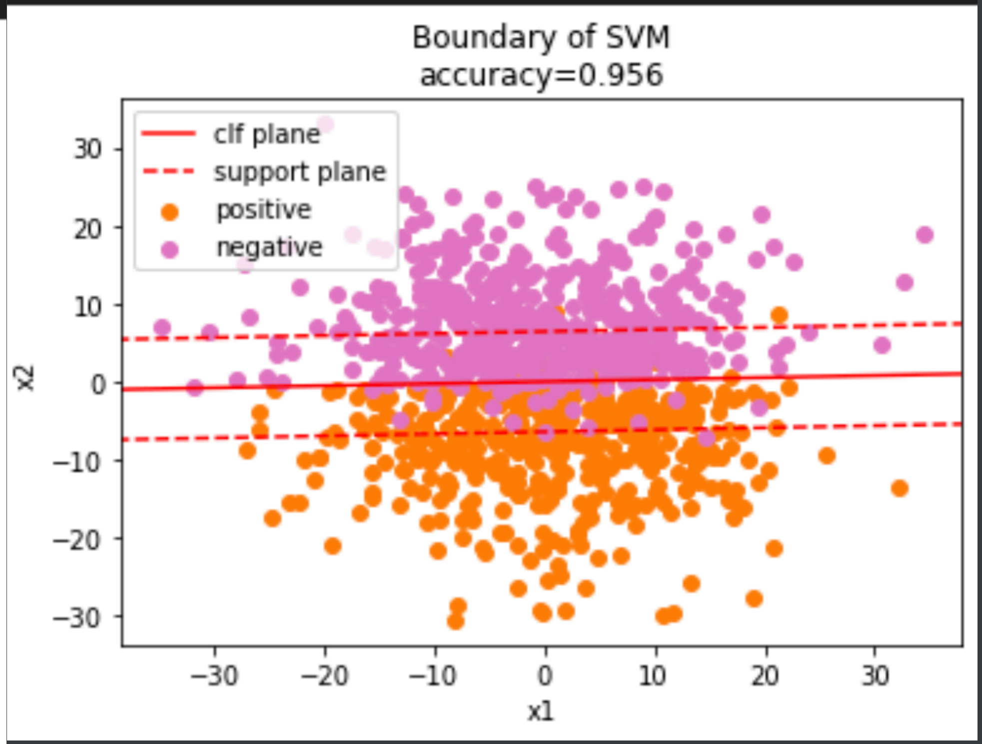
可以看到，训练集的错误率在 3.8%左右，测试集的错误率在 4.1%左右，那么正确率大约在 96%，这与生成数据时设置的错标率是非常接近的。

为了更直观的展示决策边界，我将生成的数据集改为了二维，以便利用散点图进行可视化：


Boundary of SVM
accuracy=0.96
clf plane
support plane
positive
negative
support vector
x1
x2


Boundary of SVM
accuracy=0.961
clf plane
support plane
positive
negative
support vector
x1
x2


Boundary of SVM
accuracy=0.97
clf plane
support plane
positive
negative
x1
x2


Boundary of SVM
accuracy=0.956
clf plane
support plane
positive
negative
x1
x2

可见软间隔模型在二维的情况下的分类情况是比较理想的，正确率能够达到 95%左右，这与之后调用 sklearn 库的结果非常接近。

（2）SMO 序列最小优化算法

SMO 算法的原理我认为和坐标轮换法非常类似，只不过它每次选择两个坐标，而且SVM 的对偶函数具有两个约束条件，而坐标轮换法适用于求无约束条件的情况。

我们知道 SVM 的对偶函数如下：

$$\begin{aligned}&\max_{\alpha} W(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^T x^{(j)}\\&s.t: \quad \alpha_i\geq 0, i=1,2,\cdots n\\&\qquad\quad \sum_{i=1}^{n}\alpha_i y^{(i)}=0\end{aligned}$$

求解步骤如下：

1.类比坐标轮换法，每次选取两个变量。先选取α_1,α_2,根据约束条件可以得到$\alpha_1 y^{(1)}+\alpha_2 y^{(2)}=-\sum_{(i=3)}^{n}\alpha_i y^{(i)}$

2.由于$-\sum_{(i=3)}^{n}\alpha_i y^{(i)}$可以看成常量，用$\zeta$代替，得到$\alpha_1=(\zeta-\alpha_2 y^{(2)})y^{(1)}$

3.原优化函数可以写为$W(\alpha_1,\alpha_2,\alpha_3,\cdots\alpha_n)=W((\zeta-\alpha_2 y^{(2)})y^{(1)},\alpha_2,\alpha_3,\cdots\alpha_n)$

4.因为$\alpha_3\cdots\alpha_n$为常数,则W为关于α_2的一元二次函数，通过对其求导，既可以求出W的最优解。
由于α_2存在约束,则存在有上下限的问题，需要对在W取得最优解时的α_2加以判断，最终更新α_2与α_1, b的值。

5.类比坐标轮换法，在一轮结束后,判断条件$\|\alpha_k^n-\alpha_{k-1}^n\|\leq\varepsilon$，　$\alpha=(\alpha_1,\alpha_2,\alpha_3,\cdots\alpha_n)$，　k为轮数
若不满足既可以开启下一轮，重复步骤2。若满足，则停止迭代，输出最优解$\alpha^*=\alpha_k^n$

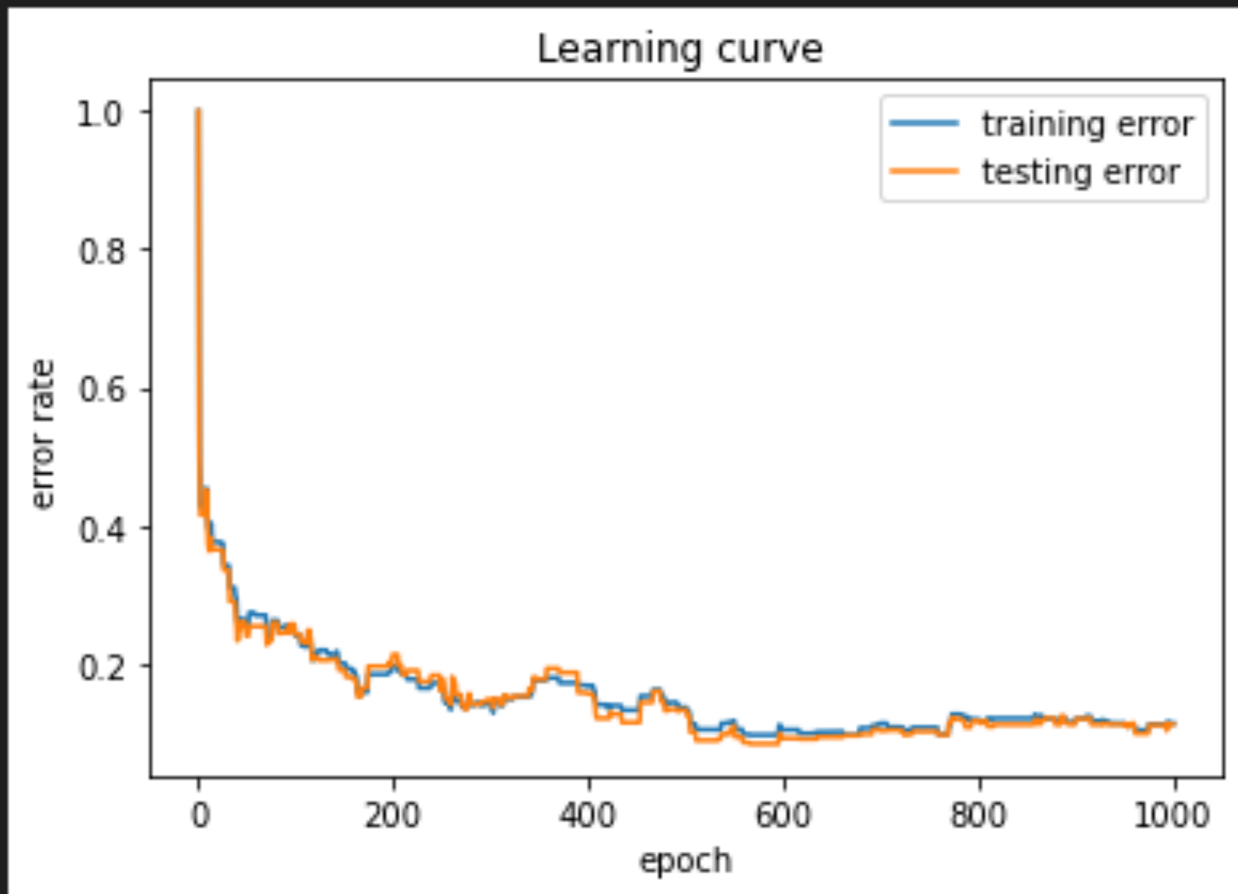
我在实现 SMO 算法时，由于觉得实现起来稍微有些麻烦，所以没有采取启发式寻找向量a_i和a_j，而是用了随机选取，这样做会导致效率低不少，于是只能通过暴力增加迭代次数来换取正确率的提升。

这里我所选取的参数是最大迭代次数 `maxiter=1000`，惩罚系数 `C=1`，由于随机选取导致的效率低下，我实现的 SMO 需要较长时间才能收敛，于是我修改了数据集大小，这次我只生成了 20 维，10000 组数据。由于在实现时出现了不少 bug 所以我保留了不少注释，希望助教在检查时多多包涵。

```
# generate data
X_data, y_data, mislabel = generate_data(20, 10000)
# print(X_data)
# print(y_data)

# split data
X_train = X_data[0:9000]
y_train = y_data[0:9000]
X_test = X_data[9001:]
y_test = y_data[9001:]

# constrcut model and train (remember record time)
model2 = SVM2(20, maxiter=1000, C=1)
model2.fit(X_train, y_train, val_data=(X_test, y_test))
```

SMO 的实验结果

```
train acc = 0.8860606060606061
test acc = 0.8878787878787879
错误率曲线如下：
```



可以看到，在最初的 200 个 epoch 中，SMO 效率较高，之后出现了振荡直至收敛，最终正确率在 90% 左右，这略低于梯度下降法的正确率，不过这个结果我是可以接受的。

（3）调用 sklearn 库进行对比

由于对 sklearn 库的使用并不是非常熟练，这里我只是简单的得到了调用标准库的正确率：

调用 sklearn 库进行对比（跑完大概需要 30s）

```python
from sklearn.svm import SVC, LinearSVC
from sklearn import metrics
model = SVC(kernel='linear')
model.fit(X_train, y_train)
prediction = model.predict(X_test)
print("标准库准确率：", metrics.accuracy_score(prediction, y_test))
```

[681] ✓ 21.6s Python

```
/Users/jianglei/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

标准库准确率：  0.954954954954955
```

可以看到调用标准库得到的准确率在 95%左右。

总结

本次实验对于我来说难度较大，尤其是 SMO 算法的实现，在完成实验时对理论推到进行了仔细地查阅和推导，但仍遇到了不少问题，花费了不少时间在本次实验上，不过确实是对 SVM 支持向量机的原理有了比较清楚的认识，总的来说收获很大。不过还是建议助教可以在实验文档中多增加一些提示和教学，帮助同学们更加轻松地完成实验。