支持向量机

授课教师：庞善民

助教：张浩、刘卓

2022年10月8日

分别使用三种损失函数

实现佩加索斯（Pegasos）算法

在给定的数据集上进行训练与测试

回顾：软SVM

引入软SVM

软SVM建模：惩罚错误分类的数目

$$\min w^T w/2 + C * (\#mistakes)$$

等价于

$$\min \boxed{w^T w/2 + C * \sum_{i=1}^{n} \ell_{0/1}(y_i(w^T x_i + b))}$$

其中，C是正则化常数，$\ell_{0/1}$是“0/1”损失函数：

$$\ell_{0/1}(z) = \begin{cases} 1, & if\ z < 1 \\ 0, & otherwise \end{cases}$$

回顾：替代损失

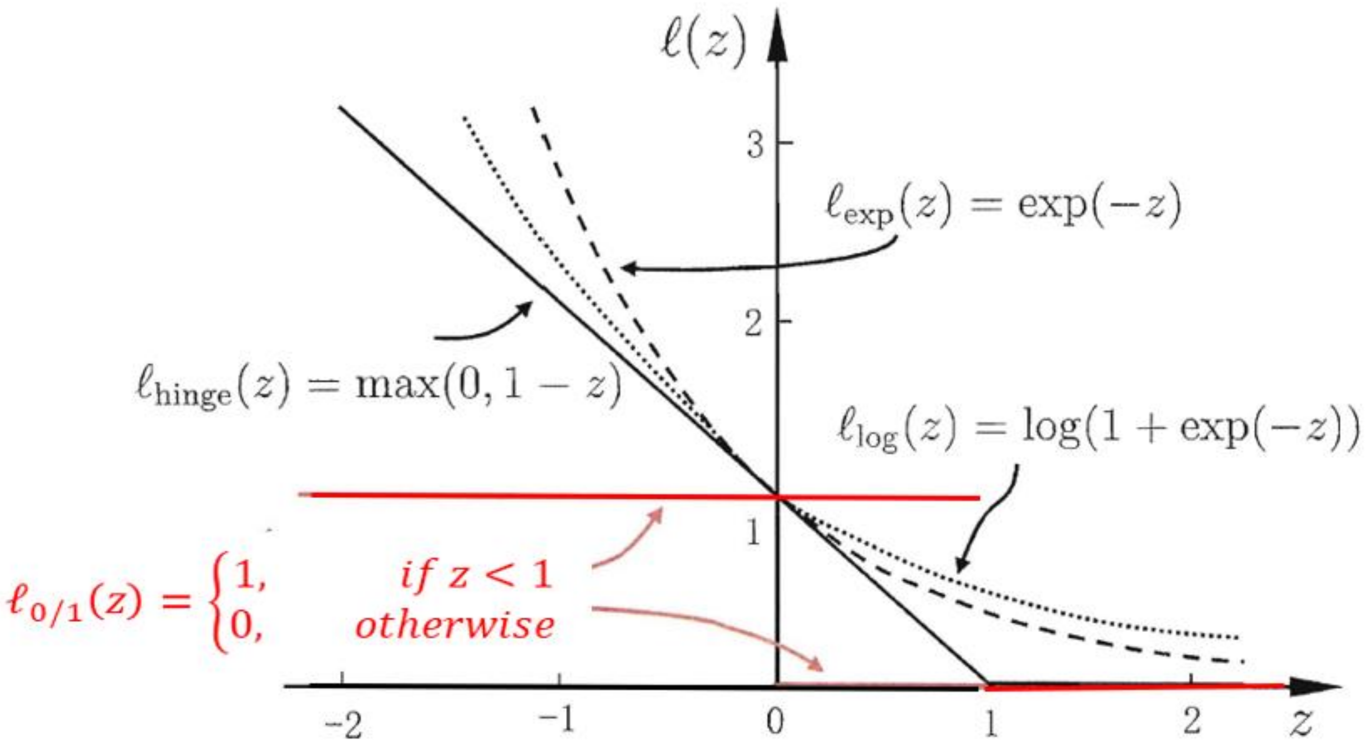
$\ell_{0/1}$ 非凸、非连续，数学性质不好

可使用如下三种替代损失：

hinge 损失: $\ell_{hinge}(z) = \max(0, 1-z)$;

指数损失(exponential loss): $\ell_{exp}(z) = \exp(-z)$;

对率损失(logistic loss): $\ell_{log}(z) = \log(1+\exp(-z))$.



回顾：hinge损失

选用hinge损失时，目标函数即：

$$f(w,b)=\frac{w^Tw}{2}+\mathrm{C}\sum_{i=1}^{n}\max(0,1-y_i(w^Tx_i+b))$$

令$C=\frac{1}{n\lambda}(\lambda>0)$，希望目标函数最小，则目标函数等价于：

$$f(w,b)=\underbrace{\frac{\lambda}{2}\|w\|^2}_{\text{正则化项}}+\underbrace{\frac{1}{n}\sum_{i=1}^{n}\max(0,1-y_i(w^Tx_i+b))}_{\text{经验损失}}$$

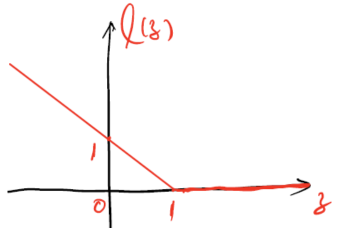
使用（子）梯度下降求解目标函数
随机近似（随机选择一个样本点）

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+\max(0,1-y_i(w^Tx_i+b))$$

回顾：hinge损失

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+\max(0,1-y_i(w^Tx_i+b))$$

$$\frac{d\ell(z)}{dz}=\begin{cases}-1, & z<1\\ 0, & z\geq 1\end{cases}$$



$$\ell_{\text{hinge}}(z)=\max(0,1-z)$$

若$y_i(w^Tx_i+b)<1$：

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+1-y_i(w^Tx_i+b)$$

$$\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w-y_ix_i\\ -y_i\end{bmatrix}$$

若$y_i(w^Tx_i+b)\geq 1$：

$$f(w,b)=\frac{\lambda}{2}\|w\|^2$$

$$\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w\\ 0\end{bmatrix}$$

hinge梯度

回顾：指数损失

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+\exp(-y_i(w^Tx_i+b))$$

$$\frac{\partial X^Ta}{\partial X}=\frac{\partial a^TX}{\partial X}=a$$

$$\boxed{\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w-y_ix_i\exp(-y_i(w^Tx_i+b))\\ -y_i\exp(-y_i(w^Tx_i+b))\end{bmatrix}}$$

exp梯度

回顾：对率损失

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+\log(1+\exp(-y_i(w^Tx_i+b)))$$

$$\boxed{\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w-\frac{y_ix_i\exp(-y_i(w^Tx_i+b))}{1+\exp(-y_i(w^Tx_i+b))}\\ -\frac{y_i\exp(-y_i(w^Tx_i+b))}{1+\exp(-y_i(w^Tx_i+b))}\end{bmatrix}}$$ log梯度

佩加索斯（Pegasos）算法（hinge损失）

初始化：$t=0;\ w_1, b_1;\ T$、λ自定义

$C=\frac{1}{n\lambda}$

For $iter\ =\ 1,2,\dots,T$

随机挑选第i个样本进行参数更新

$t\ +=\ 1\ ;\ \eta_t=\frac{1}{\lambda t}$

if $y_i(w^Tx_i+b)<1$

$$\begin{bmatrix} w_{t+1} \\ b_{t+1} \end{bmatrix}=\begin{bmatrix} w_t \\ b_t \end{bmatrix}-\eta_t\begin{bmatrix} \lambda w_t-y_ix_i \\ -y_i \end{bmatrix}$$

else

$$\begin{bmatrix} w_{t+1} \\ b_{t+1} \end{bmatrix}=\begin{bmatrix} w_t \\ b_t \end{bmatrix}-\eta_t\begin{bmatrix} \lambda w_t \\ 0 \end{bmatrix}$$

hinge梯度

End

算法核心：

下降步长η_t（逐渐减小）

下降方向（子梯度）

用到的库

第三方：numpy（安装略）
import numpy as np

第三方：scipy
安装：
命令行中，输入pip install scipy
import scipy.io

实验流程

数据读取

↓

佩加索斯算法

- hinge损失
- 指数损失
- 对率损失

↓

计算准确率
画目标函数曲线

代码

```python
def pegasos(train, test, C, T, loss_type='hinge', func_interval=100):
    train_x = train['X']  # 4000*1899
    train_y = train['y']  # 4000*1

    test_x = test['Xtest']  # 1000*1899
    test_y = test['ytest']  # 1000*1

    num_train = train_x.shape[0]  # 4000
    num_test = test_x.shape[0]  # 1000
    num_features = train_x.shape[1]  # 1899

    lambda_ = 1 / (num_train * C)
```

代码

```
# 高斯初始化权重W和偏置b
W = np.random.randn(num_features, 1)  # 1899*1
b = np.random.randn(1)

func_list = []

# 随机生成一组长度为T,元素范围在[0, num_train-1]的下标,供算法中随机选取训练样本
choose = np.random.randint(0, num_train, T)
```

代码

```
for t in range(1, T+1):
    # TODO:写出eta_t的计算公式
        # eta_t =

    # 随机选取的训练样本下标
        i = choose[t-1]
    x_i = train_x[[i]].T  # 1899*1
    y_i = train_y[i]  # 1

    if loss_type == 'hinge':
        # TODO:写出hinge_loss下的梯度更新公式
        elif loss_type == 'exp':
        # TODO:写出exp_loss下的梯度更新公式
    elif loss_type == 'log':
        # TODO:写出log_loss下的梯度更新公式
        t += 1
```

代码

```
if __name__ == '__main__':
    C = 0.001
    T = 10000  # 迭代次数
        func_interval = 500  # 每隔多少次迭代计算一次目标函数
    times = 2  # 测试次数

        # loss类型切换
        loss_types = ['hinge', 'exp', 'log']
    loss_type = loss_types[1]
```

改下标，切换损失类型

补充

1. 指数操作(尤其是在指数损失中)具有不稳定性，可能会导致梯度过大，为避免这种情况，可以对指数进行判断，如果指数过大，则暂时不用当前样本来训练。

如对指数损失：

如果红框内容<3(或其它值，自行调节)，则使用当前样本更新梯度，否则跳过当前样本，不使用当前样本更新梯度，尝试下一个

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} \lambda w - y_i x_i \exp\left(\boxed{-y_i(w^T x_i + b)}\right) \\ -y_i \exp(-y_i(w^T x_i + b)) \end{bmatrix}$$

但该方法并不能完全消除这种不稳定性，建议多次运行程序，选取其中较好的结果。而在对率损失情况下，虽然也有指数操作，但如果C设置得当，一般不会出现溢出，可不采用上述方法。

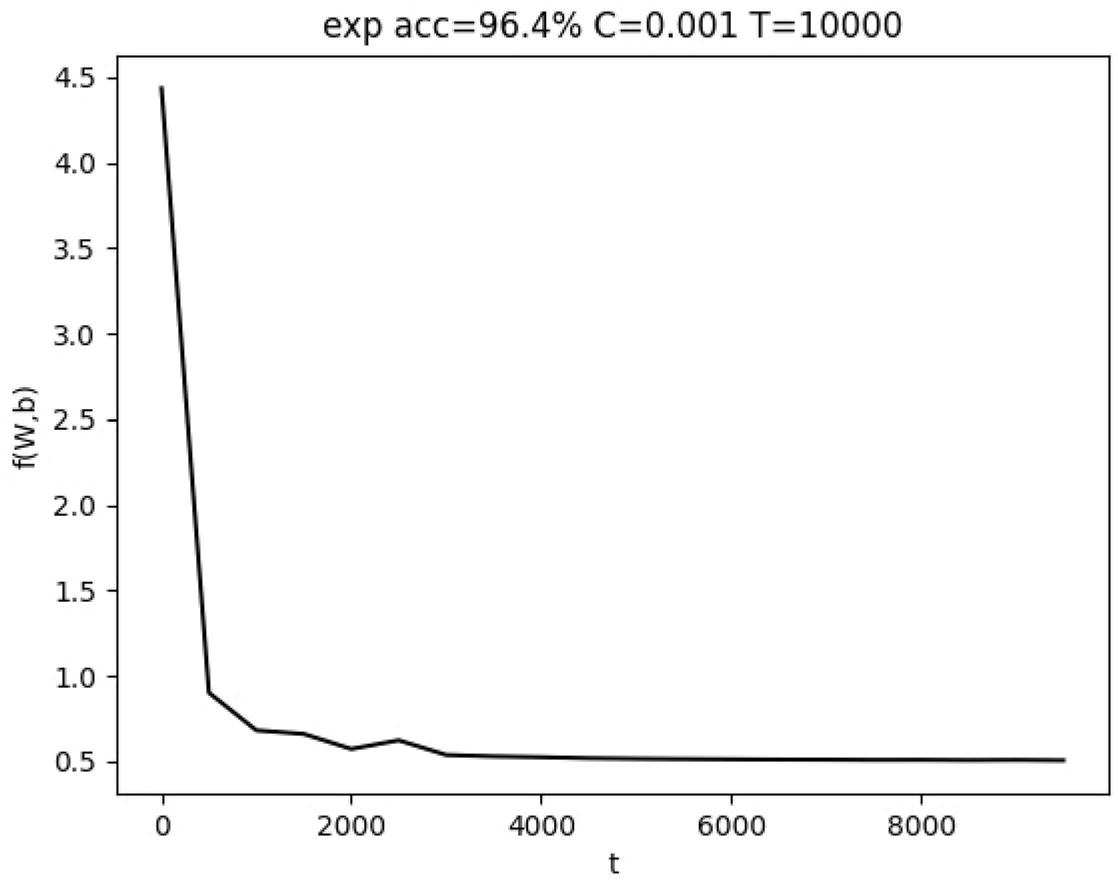
2. 提供的测试集中，正负样本数为308:692，所以如果准确率出现30.8%或69.2%，说明模型将样本全部判定为正/负，相当于训练失败。

运行结果

控制台输出

times:1
t = 500, func = 802318630587.8149
……
t = 10000, func = 1129.7242
accuracy = 69.2%
times:2
t = 500, func = 4.4340
……
t = 10000, func = 0.5067
accuracy = 96.4%
stat_acc [69.2, 96.4]
acc_best 96.4
func_list_best [……]

./output



实验要求

1. 4个填空，让程序在3种loss下都可以得出准确率90%以上的结果

2. 调整C、T参数，看看不同的C、T会导致什么变化

3. 将上述内容写入实验文档，并附上对应情况下的所得目标函数曲线图

4. 思源空间提交：

整个程序的压缩包(可运行)+实验文档(学号_姓名_实验2.pdf)

+ 作业1即将截止，及时提交，互相转告

Thank You　Q & A

张浩：1050852440@qq.com

刘卓：lzpmbw@163.com