logistic regression

yfpeng

September 2022

1 算法实现

根据 PPT 中提供的公式，我们依次计算出交叉熵损失和对应的梯度，然后进行多次迭代，以下是具体的算法实现：

```
#SGD
def SGD(train_set,learning_rate,rand_init=True,error=1e-5,max_iter=20,batch_size=10):
    B=batch_size
    loss_epoch=[]
    if rand_init==True:
        np.random.seed(4)
        w=np.random.normal(0,1,3)
    else:
        w=np.array([0,0,0])
    for i in range(max_iter):
        batch_idx=np.random.choice(train_set.shape[0],B,replace=False)
        batch=train_set[batch_idx,:]
        X=batch[:,0:-1]
        Y=batch[:,-1]
        #求loss
        raw_loss=np.log(1+np.exp(-np.sum(w*X,-1)*Y))
        loss=float(np.sum(raw_loss,0)/B)
        loss_epoch.append(loss)
        #求梯度
        raw_grad=((1/(1+np.exp(np.sum(w*X,-1)*Y)))*(-Y)).reshape(-1,1)*X
        grad=np.sum(raw_grad,0)/B
        w=w-learning_rate*grad
        if np.linalg.norm(w)<error:
            break
    return w,loss_epoch
```

图 1: logistic 算法实现

注意到，上面我没有在每个 epoch 的迭代后对数据集进行打乱，因为 SGD 每次选择样本都是随机的，随意打乱与否，我个人感觉影响不大。

2 可视化

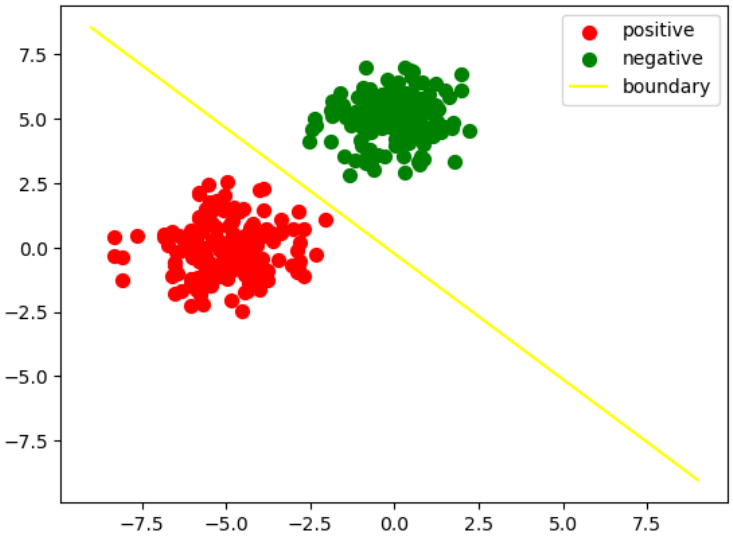

图 2: 训练集可视化

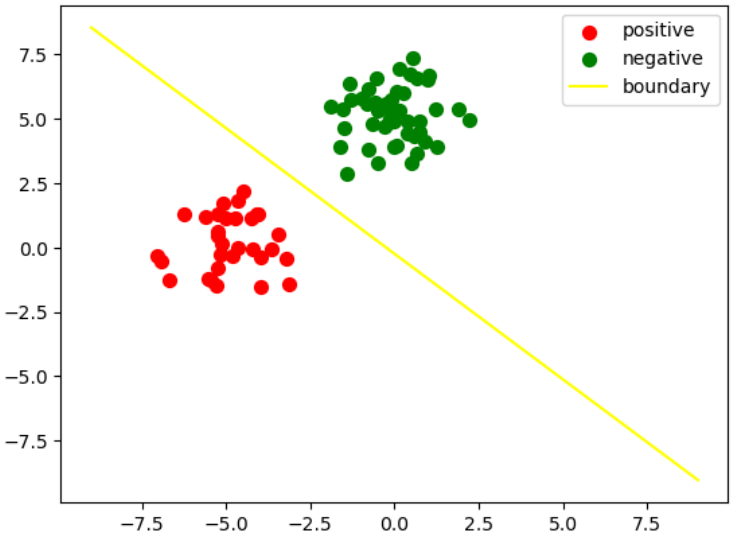

图 3: 测试集可视化

可以看到，对这样一个线性可分数据集，这个算法几乎可以做到完全分类正确。

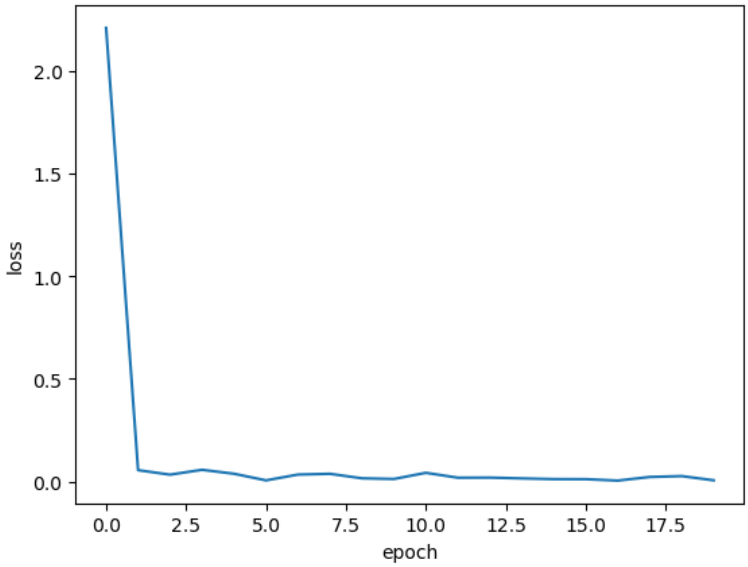

图 4: loss 变化情况

更确切地讲，这里的 epoch 实际上指的是 step，在本次作业中我们不加区分。

3 改变超参

3.1 改变学习率

将学习率改为原来 $\frac{1}{10}$:

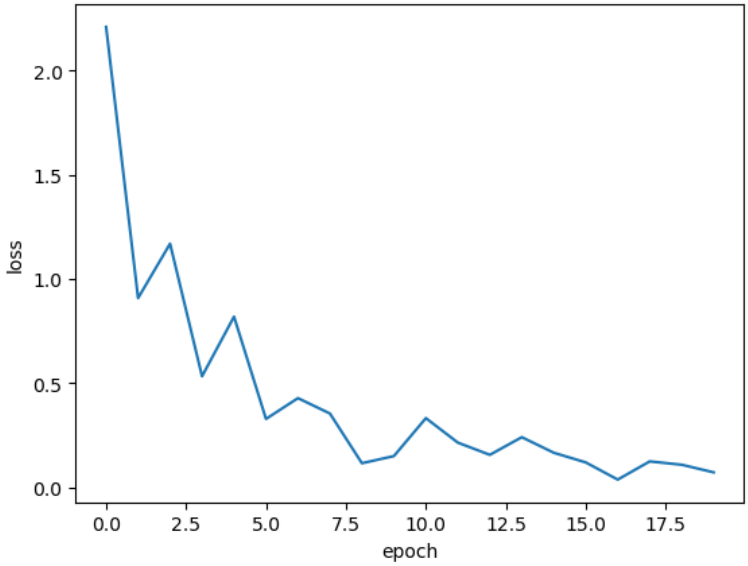

图 5: 改变学习率后 loss 变化情况

loss 反而不如原来稳定，这是个奇怪的现象，目前还每太想通。

3.2 改变批量大小

批量改为原来 2 倍:

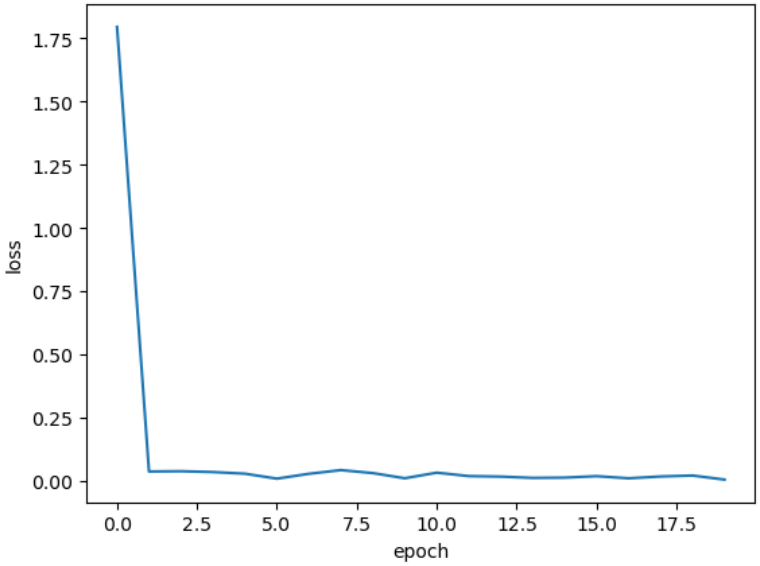

图 6: 改变批量大小后 loss 变化情况

在这样简单的数据集上，增大批量大小，收敛加快

3.3 改变样本分布

将两个分布拉近：

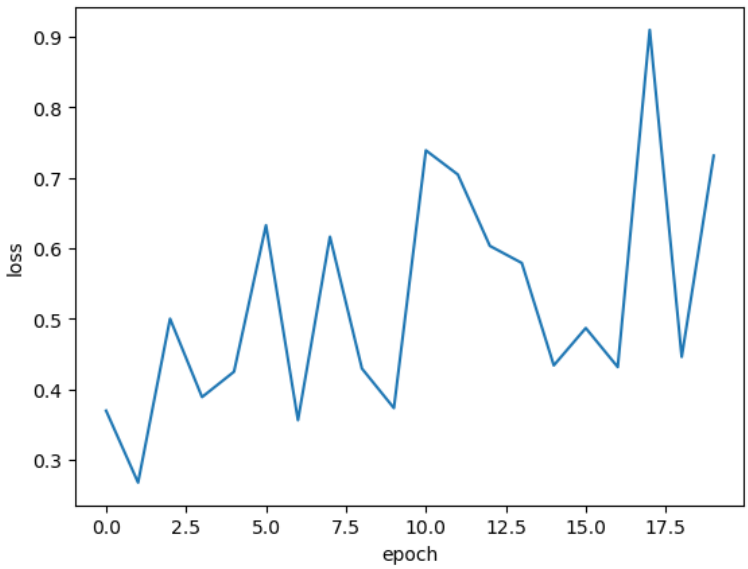

图 7: 改变样本分布后 loss 变化情况

数据集线性不可分，loss 不收敛

3.4 改变样本分布后调参

改变样本分布后调参，试图让效果变好：

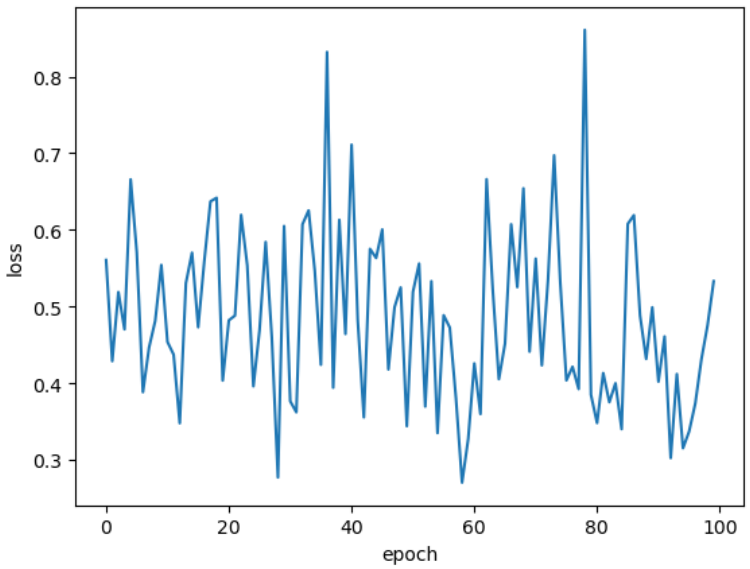

图 8: 改变批量大小并调参后 loss 变化情况

可以看出，降低学习率，增强迭代次数，虽然 loss 仍不收敛，但效果有一定提升