HW1 线性模型

1 线性回归 Linear Regression（50）

1.1 输入数据集 （10）

data1.txt为回归数据集，每一行为一个样本，前两列为特征，最后一列为目标值。按照7:3的比率划分训练集和验证集。

```
data = np.loadtxt('data1.txt',delimiter=',')
print(data.shape)
num_feature = data.shape[1] - 1
data = data.astype('float32')
# data normalization
data_norm = data.copy()
...
data_train, data_test = train_test_split(data_norm, test_size=0.3, random_state=42)
```

```
[[2.10400e+03 3.00000e+00 3.99900e+05]
 [1.60000e+03 3.00000e+00 3.29900e+05]
 [2.40000e+03 3.00000e+00 3.69000e+05]
 [1.41600e+03 2.00000e+00 2.32000e+05]
 [3.00000e+03 4.00000e+00 5.39900e+05]
 [1.98500e+03 4.00000e+00 2.99900e+05]
 [1.53400e+03 3.00000e+00 3.14900e+05]
 [1.42700e+03 3.00000e+00 1.98999e+05]
 [1.38000e+03 3.00000e+00 2.12000e+05]
 [1.49400e+03 3.00000e+00 2.42500e+05]
 [1.94000e+03 4.00000e+00 2.39999e+05]
```

1.2 线性回归（20）

建立线性回归模型，分别使用正规方程和梯度下降法求得参数解。

- 正规方程

$$ w=(X^TX)^{-1}X^Ty $$

```
term = np.matmul(X_train.T, X_train)
term_inv = np.linalg.inv(term)
w = np.matmul(np.matmul(term_inv,X_train.T),y_train.reshape(-1,1))
```

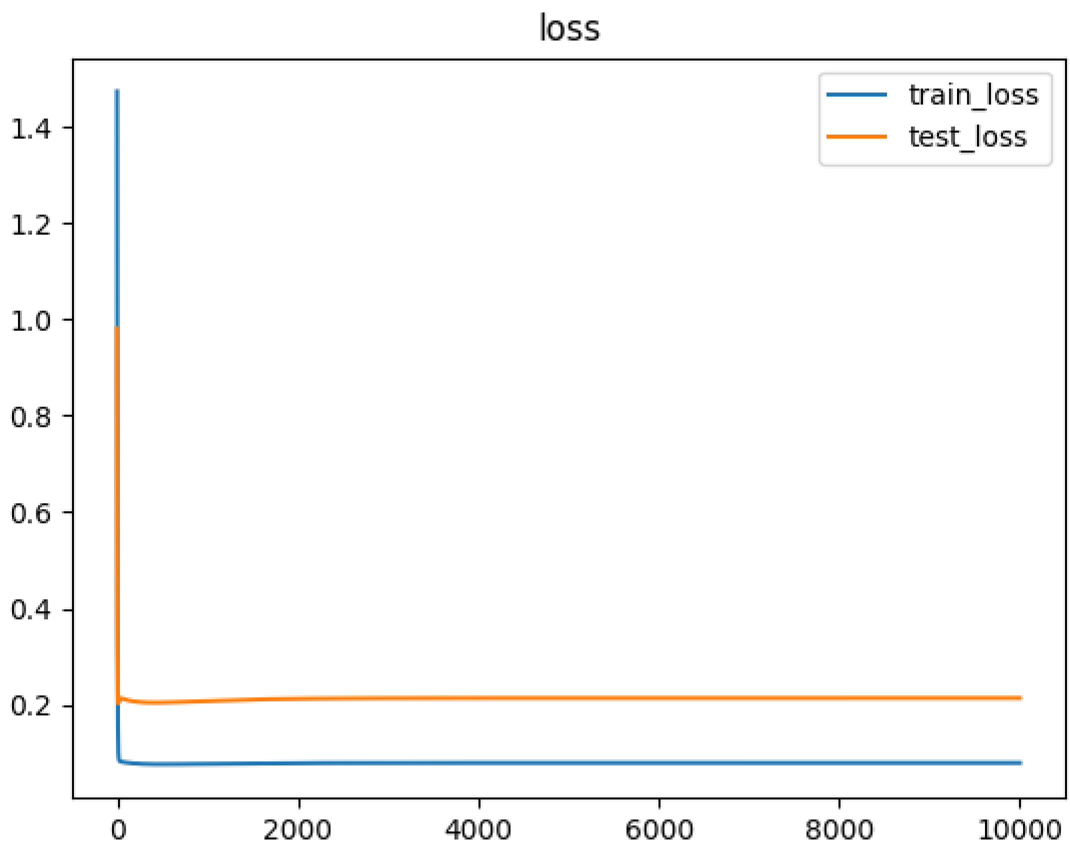
- 梯度计算

$$ g=\frac{1}{m}\sum^m_{i=1}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)} $$

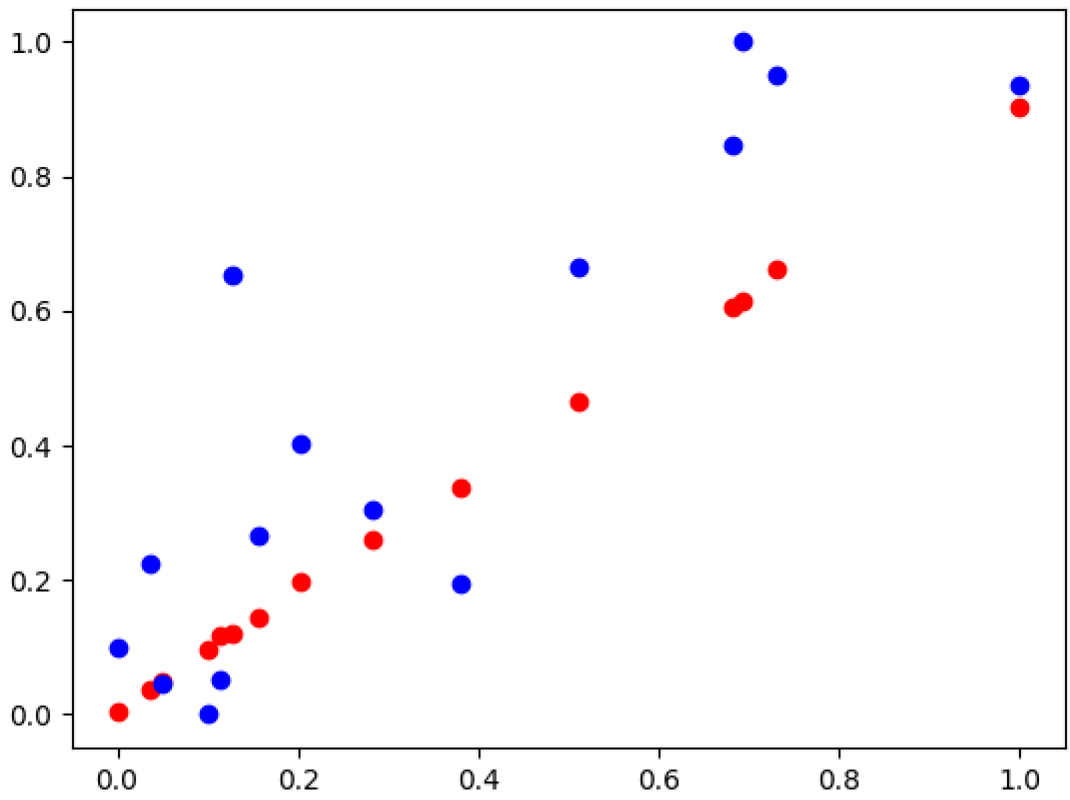
```
iterations = 10000
lr = 0.01
log = []
for i in range(iterations):
    y_pred = np.matmul(X_train, w)
    test = y_train.reshape(-1,1)
    term = lr*np.mean((y_pred-y_train.reshape(-1,1))*X_train, axis=0).reshape(-1,1)
    w -= term
    loss = L2_loss(y_pred,y_train)
    print('iter:{},loss:{}'.format(i,loss))
    log.append([i,loss])
```

1.3 可视化（20）

- 使用梯度下降法时请可视化loss曲线



- 请可视化验证集上所求回归直线



2 逻辑回归 Logitstic Regression/Percetron（50）

1.1 输入数据集（10）

data2.txt为分类数据集，每一行为一个样本，前两列为特征，最后一列为目标值。按照7:3的比率划分训练集和验证集。

```
data = np.loadtxt('data2.txt',delimiter=',')
print(data.shape)
num_feature = data.shape[1] - 1

data = data.astype('float32')
# data normalization
data_ori = data.copy()

# train val split
data_train, data_test = train_test_split(data, test_size=0.3, random_state=45)
print(data_train,data_test)

def data_normalization(data_train):
    data_max = np.max(data_train,axis=0,keepdims=True)
    data_min = np.min(data_train,axis=0,keepdims=True)
    data_train = (data_train - data_min)/(data_max - data_min)
    return data_train
data_train = data_normalization(data_train)
data_test = data_normalization(data_test)
```

```
[[0.9926494  0.55916625 1.        ]
 [0.9178213  0.68197525 1.        ]
 [0.1250839  0.50379884 0.        ]
 [0.27252406 0.31172025 0.        ]
 [0.21534562 0.37665957 0.        ]
 [0.53405404 0.52714384 1.        ]
 [0.45537946 0.28788552 0.        ]
 [0.74975854 0.14670505 0.        ]
 [0.         0.2781716  0.        ]
 [0.14904502 0.98045516 1.        ]
 [0.30083445 0.22294258 0.        ]
```

1.2 逻辑回归（20）

建立逻辑回归模型，分别使用梯度下降法求得参数解。可尝试使用L2正则化。

- 梯度计算

$$ g=\frac{1}{m}\sum^m_{i=1}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)} $$

```
iterations = 10000
lr = 0.6

log = []
log_test = []
R = 0.00001
# gradient descent
for i in range(iterations):
    y_pred = sigmoid(np.matmul(X_train, w))
    g = lr*(np.mean((y_pred-y_train)*X_train, axis=0).reshape(-1,1))
    w -= g
    loss = cross_entropy_loss(y_pred,y_train)
    print('iter:{},loss:{}'.format(i,loss))
    log.append([i,loss])

    y_pred_test = sigmoid(np.matmul(X_test, w))
    loss = cross_entropy_loss(y_pred_test,y_test)
    log_test.append([i,loss])
#     print('iter:{},val_loss:{}'.format(i,loss))
```

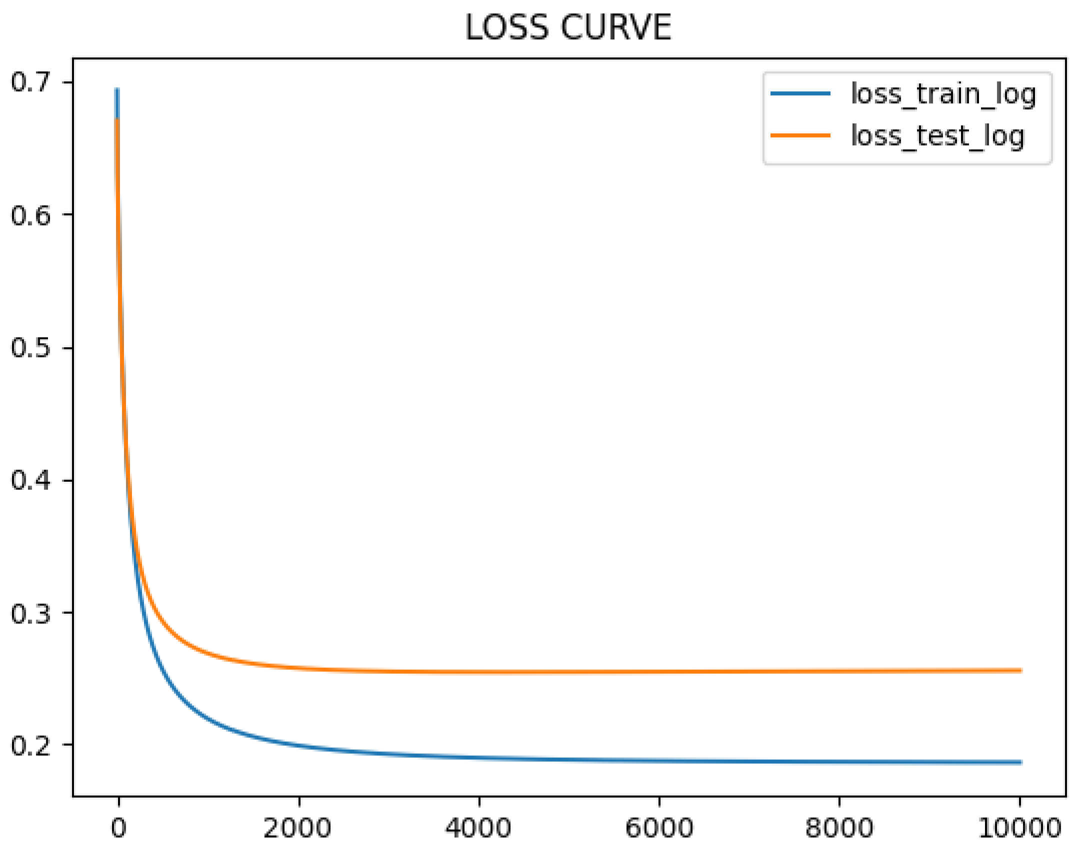
- 梯度计算（L2正则化）

$$ g_j=\frac{1}{m}\sum^m_{i=1}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}+2*\lambda*\theta_j $$

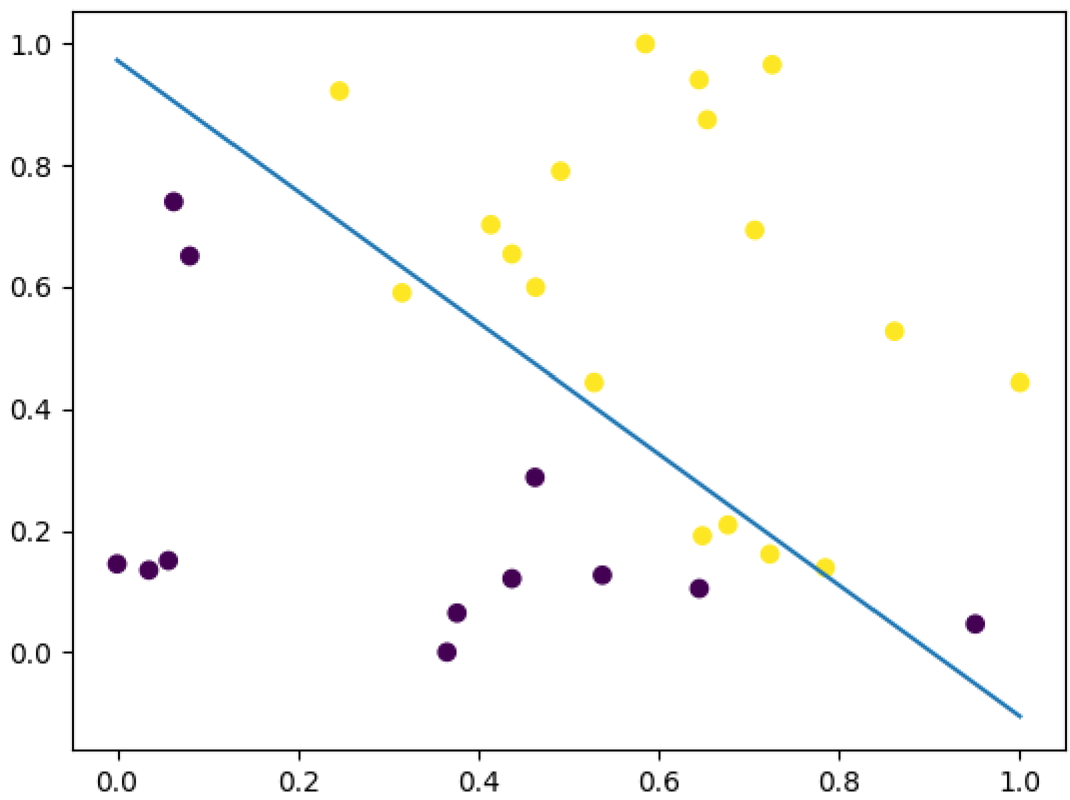
```
iterations = 10000
lr = 0.6

log = []
log_test = []
R = 0.00001
# gradient descent
for i in range(iterations):
    y_pred = sigmoid(np.matmul(X_train, w))
    g = lr*(np.mean((y_pred-y_train)*X_train, axis=0).reshape(-1,1)+2*R*w)
    w -= g
    loss = cross_entropy_loss(y_pred,y_train)
    print('iter:{},loss:{}'.format(i,loss))
    log.append([i,loss])

    y_pred_test = sigmoid(np.matmul(X_test, w))
    loss = cross_entropy_loss(y_pred_test,y_test)
    log_test.append([i,loss])
#     print('iter:{},val_loss:{}'.format(i,loss))
```

1.3 可视化（20）

- 使用梯度下降法时请可视化loss曲线



- 请可视化验证集上所求分类直线



3 Bonus：分析 （10）

- 对比正规方程和梯度下降法，基于实验结果比较两者之间的优劣。
- 答：梯度训练2000个Epoch得到的W的值与正规方程直接求出的值的对比：

```
梯度下降次数:  2000,梯度下降求解:[[ 0.87936694]
 [-0.0016742 ]
 [ 0.01074254]],
正规方程求解:[[ 0.89707607]
 [-0.02132246]
 [ 0.01620799]]
```

- 梯度训练5000个Epoch得到的W的值与正规方程直接求出的值的对比：

```
梯度下降次数:  5000,梯度下降求解:[[ 0.89689827]
 [-0.02112526]
 [ 0.01615316]],
正规方程求解:[[ 0.89707607]
 [-0.02132246]
 [ 0.01620799]]
```

- 基于实验结果，对比没有正则化的情况和L2正则化的逻辑回归模型。
- 答：在逻辑回归模型中，没有使用正则化在一定情况下可能会导致过拟合的教易产生，而在实际情况中，我发现我们当前所写的逻辑回归模型正在正则化后的loss却还有些许的上升。
- 分析特征归一化和不做归一化对模型训练的影响。
- 答：使用了归一化的数据内容会更容易计算，并且也更规整。同样的，使用了特征的归一化能使得求最优解的过程变得平缓，更容易正确收敛，能有效提高梯度下降的速度。
-