机器学习实验报告

学院：软件学院

专业：软件工程

作者:

侯斯扬

指导教师:

谭明奎

学号:

201930380488

班级:

软件 2 班

2021-10-7

逻辑回归和支持向量机

摘要**—*对比理解梯度下降和批量随机梯度下降；理解逻辑回归和线性分类的区别；理解逻辑回归和支持向量机的原理*

I. 介绍

本实验的主要目的如下：对比理解梯度下降和批量随机梯度下降的区别与联系。对比理解逻辑回归和线性分类的区别与联系。进一步理解 SVM 的原理并在较大数据上实践。

II. 方法和理论

A. 逻辑回归

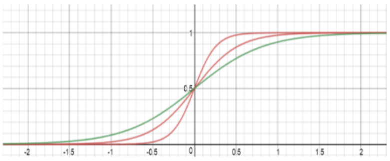
区别于线性回归：

线性回归：一般是指通过计算输入变量的加权和，并加上一个常数偏置项（截距项）来得到一个预测值。

逻辑回归：如何用连续的数值去预测离散的标签值？我们能否将线性回归输出的一个连续的数值变成一个标签呢？一个比较直观的想法是设定一个阈值，比如回归模型输出的 y 大于 0 时，属于正类，y 小于 0 时属于负类，由此，我们有一个更好的方法，将 y 等于 1 和 y 等于 100 都归为正类的同时，也考虑它们各自属于正类的置信度，这便是逻辑回归

逻辑回归的 sigmoid 函数

$$g(z) = \frac{1}{1+e^{-z}}$$



逻辑回归通过估计样本属于正类或者负类的概率，再通过以上公式判断是属于哪个类别

If $z \to +\infty$, then $g(z) \to 1$; if $z \to -\infty$, then $g(z) \to 0$

逻辑回归的损失函数：（由对数的似然函数构造损失函数）

$$J(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1-y_i)\log(1-h_{\mathbf{w}}(\mathbf{x}_i)))$$

运用梯度下降（Gradient Descent）来求解损失函数的最小值

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i + \lambda\mathbf{w}$$

Compute gradient $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ of $J(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i + \lambda\mathbf{w}$$

学习率：是一个大于 0 的数，能够控制沿着某个方向走多长一段距离（但不是步长）一般随着迭代次数的增加，学习率逐渐减小

Update parameters with **learning rate** η

$$\mathbf{w} := \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

B. 支持向量机

在这个实验中我们讨论的是支持向量机的软间隔：实践中由于异常数据的存在，导致超平面不能完全将数据分为两部分，我们希望这样的样本越少越好

支持向量机的损失函数：

Hinge loss:

$$\xi_i = \max\left(0, 1 - y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right)\right)$$

其优化：（就是在原优化函数的后面加了一截，这一截里面的 C 表示一个大于 0 的常数，损失函数取值为 0 或者 1）

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

通过梯度下降，进行迭代

$$\mathbf{w} = \mathbf{w} - \eta\nabla_{\mathbf{w}}L(\mathbf{w}, b)$$

$$b = b - \eta\nabla_b L(\mathbf{w}, b)$$

（本实验主要采用批量随机梯度下降）

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \mathbf{w} + \frac{C}{n} \sum_{i=1}^{n} g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\nabla_b L(\mathbf{w}, b) = \frac{C}{n} \sum_{i=1}^{n} g_b(\mathbf{x}_i)$$

Algorithm 1: GD

Initialize parameter **w** and learning ra

while stopping condition is not achie

$\quad \mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}, b)$

$\quad b = b - \eta \nabla_b L(\mathbf{w}, b)$

end

C. 支持向量机（SVM）与逻辑回归的比较

支持向量机：

$$\min\ J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

逻辑回归：

$$\min J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i}) + \frac{\lambda}{2} ||\mathbf{w}||_2^2$$

III. 实验

A. 数据集

实验使用的是 LIBSVM Data 的中的 a9a 数据，包含 32561/16281(testing)个样本，每个样本 123/123 (testing)个属性。在读取数据时可能会出现维度不对的问题，是因为数据最后列全为零而被忽略，可以在下载的数据集文件后面自行添加后再读取，也可在读取数据集时指定 n_features=123 来解决。

B. 实现

1. 对数据的处理（导入包的依赖，读取数据并切分为实验的训练集和验证集，进一步对数据进行类型的转换）
2. 图 1.定义 sigmoid 函数

```
def sigmoid(z):
    return 1/(1 + np.exp(-z))
```

3. 图 2.定义逻辑回归的损失函数

```
def logistic_gradient(X,y,theta):
    return X.T.dot(sigmoid(X.dot(theta))-y)
```

4. 图 3.定义逻辑回归的批量随机梯度下降函数

```
def logistic_descent(X,y,theta,alpha,num_iters,batch_size,X_valid,y_valid):
    loss_train = np.zeros((num_iters,1))
    loss_valid = np.zeros((num_iters,1))
    data = np.concatenate((y,X),axis = 1)
    for i in range(num_iters):
        sample = np.matrix(random.sample(data.tolist(),batch_size))
        grad = logistic_gradient(sample[:,1:125],sample[:,0],theta)
        theta = theta - alpha * grad
        loss_train[i] = logistic_loss(X,y,theta)
        loss_valid[i] = logistic_loss(X_valid,y_valid,theta)
    return theta,loss_train,loss_valid
```

5. 图 4.确定 batch_size 的大小，进行参数的初始化

```
theta = np.zeros((X_train.shape[1],1))
alpha = 0.0001
num_iters = 6
```

6. 后执行梯度下降，进行 6 次迭代求解得到测试集的 Lvalidation，其结果展示如下图 5

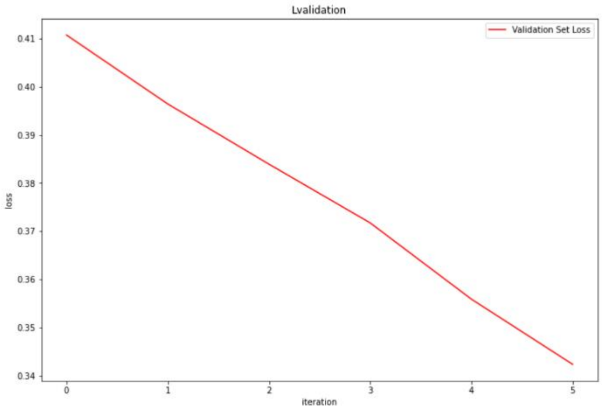


图 5. Lvalidation 迭代结果图

7. 选取 hinge 作为损失函数
   图 6.Hinge loss 的定义

```
def hinge_loss(X, y, theta, C):
    loss = np.maximum(0, 1 - np.multiply(y, X.dot(theta))).mean()
    reg = np.multiply(theta,theta).sum() / 2
    return C * loss + reg
```

8. 图 7.hinge 回归函数的定义

```
def hinge_gradient(X, y, theta, C):
    error = np.maximum(0, 1 - np.multiply(y, X.dot(theta)))
    index = np.where(error==0)
    x = X.copy()
    x[index,:] = 0
    grad = theta - C * x.T.dot(y) / len(y)
    grad[-1] = grad[-1] - theta[-1]
    return grad
```

9. 图 8.支持向量机的梯度下降定义

```
def svm_descent(X, y, theta, alpha, num_iters, batch_size, X_valid, y_valid, C):
    loss_train = np.zeros((num_iters,1))
    loss_valid = np.zeros((num_iters,1))
    data = np.concatenate((y, X), axis=1)
    for i in range(num_iters):
        sample = np.matrix(random.sample(data.tolist(), batch_size))
        grad = hinge_gradient(sample[:,1:125], sample[:,0], theta, C)
        theta = theta - alpha * grad
        loss_train[i] = hinge_loss(X, y, theta, C)
        loss_valid[i] = hinge_loss(X_valid, y_valid, theta, C)
    return theta, loss_train, loss_valid
```

10. 图 9.进行模型参数的初始化

```
theta = np.random.random((X_train.shape[1],1))
alpha = 0.01
num_iters = 6
```

11. 调用 SVM 的梯度下降求解 Lvalidation，迭代 6 次的结果如图 10 所示

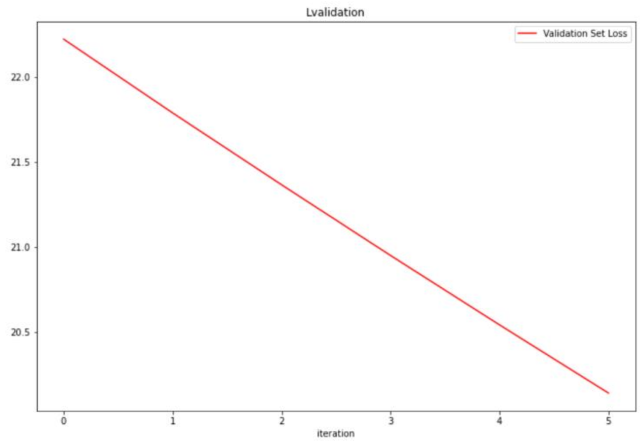


图 10. Lvalidation 迭代结果

IV. 结论

通过本次实验，我对于线性回归，线性分类有了更加明确的区分，对于逻辑回归以及支持向量机的原理有了更进一步的了解；

对于梯度下降的了解：批量梯度下降有准确度高但训练速度慢的特点，而随机梯度下降虽然准确度不高但速度快，本次实验也有才用到了二者的结合——批量随机梯度下降；

在此次实验中，也接触到了梯度上升，同时也 Adam、SGD、Mini-Batch 等算法，有一些虽然照着代码打哩一遍，但仍然似懂非懂，课后需要再花时间继续了解。