HW4

2015011313 徐鉴劲 计54

题目要求使用A2：1000个样本，B2：600个样本，C2：1600个样本进行学习，对A1,B1,C1各100个样例点进行分类。

KNN分类

KNN的算法是使用所有数据作为一个数据库，在测试的时候，计算所有样例点与它的距离。取距离前k小的数据点，进行投票，取最频繁的那个作为分类结果。

然而在实际的应用中，KNN所使用的特征空间是非常庞大的，哪怕是很多数据点都不足以描述一个足够准确的分类面。而且也存在着一些不太适合KNN的分类点，比如两个不同类别的数据点相隔非常近的情况。

使用sklearn库的K近邻分类方法`KNeighborsClassifier`，核心代码如下：

```
def do_KNN(data, label, testdata, testlabel):
    knn = sklearn.neighbors.KNeighborsClassifier()
    knn.fit(data, label)
    predict = knn.predict(testdata)
```

实验重复了五次，总体结果和分类别结果如下：

Type	Mean	Std	
0+1+2	84.87	1.15	
0	98.00	1.41	
1	67.60	5.24	
2	89.00	0.89	

在数据样本点少的1类例子中。

线性分类

我是用SGD训练一个线性分类器，将3维的数据点分为三个类，使用一个3x3的权重矩阵和3长度的偏置向量。数学表达式可以写成：

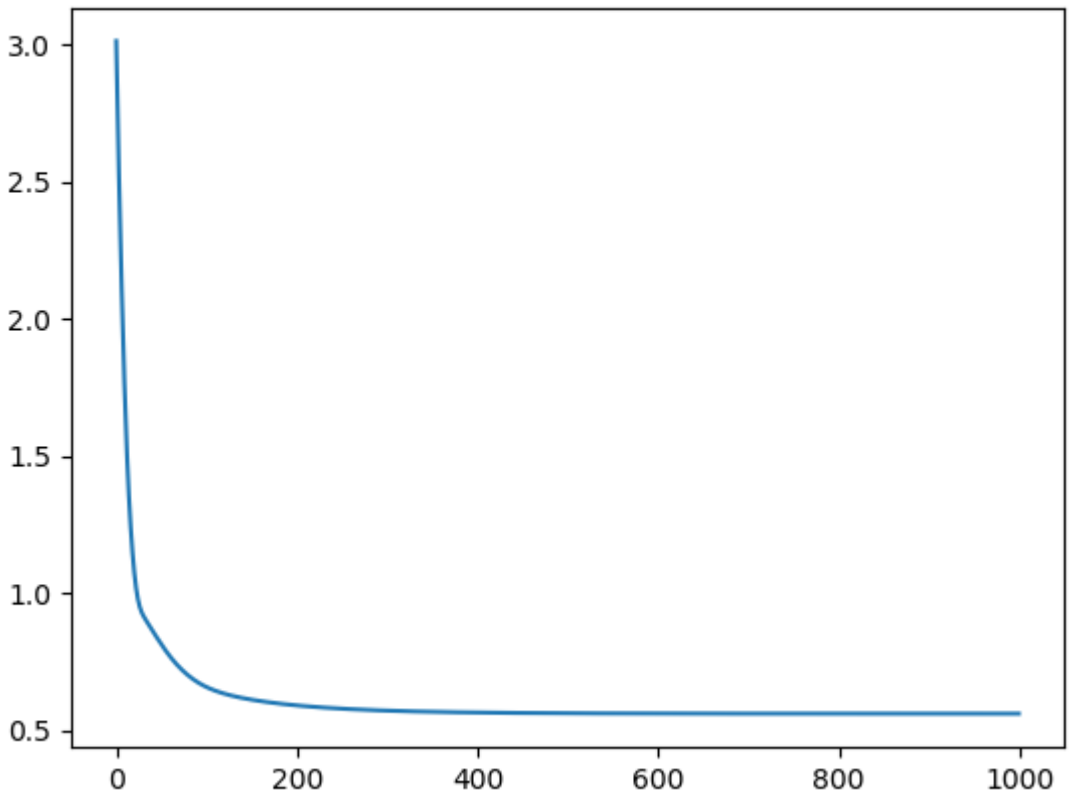
$$y = x^T w + b$$

训练loss使用cross entropy，使用Adam优化器进行训练，0.01学习旅，优化1000次得到结果。

使用tensorflow进行了实现，核心代码如下：

```
w1 = tf.Variable(np.random.uniform(size=(3, 3)), dtype=tf.float32)
b1 = tf.Variable(np.zeros((3,)), dtype=tf.float32)

y_linear = tf.matmul(x, w1) + b1

y_pred = tf.argmax(y_linear, axis=1)

loss_linear = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logi

train_op_linear = tf.train.AdamOptimizer(0.01).minimize(loss_linear, var_list=[w
```

训练loss变化图如下图所示：



实验重复了五次，最终结果和分类别结果如下表所示：

Type	Mean	Std	
0+1+2	70.67	2.51	
0	98.00	0.89	
1	23.40	2.87	
2	89.80	4.12	

可见类别之间正确率的差别是有着很大的区别的。第1类数量最小，而且也不利于使用线性分类器进行分类，在训练的时候被显著牺牲掉了，0类和2类则相差不如1类这样巨大。2类正确率少可能是因为它的方差大，不容易分类。

二次分类

这个二次分类的方程式可以表示成：

$y = x^T w_1 + x^T w_2 x + b$，输入是一个向量，输出是一个向量，表示在每一类上的得分。

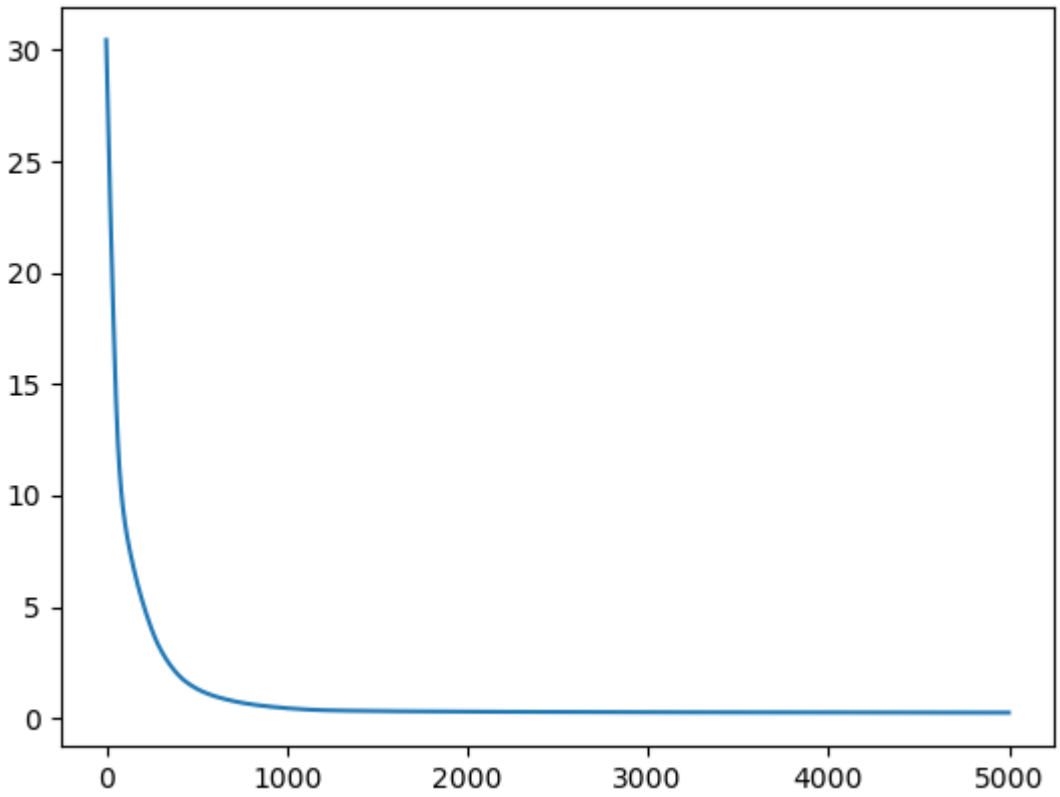
但是使用tensorflow来进行实现的时候，我遇到了不小的困难，因为实际上都是以一个batch进行操作的，即x是一个大小为(N, 3)的矩阵，然后这个方程最后要输出一个矩阵。

我实现的方法是采用张量乘法tensordot。张量乘法是说对于两个张量（维度大于2）指定一些轴，让两个张量在这些轴上的元素对应相乘并相加，比如说矩阵相乘就可以表示成 $A \times B | A(1), B(0)$。

核心代码如下：

```
# linear
w2 = tf.Variable(np.random.uniform(size=(3, 3)), dtype=tf.float32)
# bias
b2 = tf.Variable(np.zeros((3,)), dtype=tf.float32)
# quad tensor
w3 = tf.Variable(np.random.uniform(size=(3, 3, 3)), dtype=tf.float32)
# quad term
y_quad = tf.reduce_sum(tf.tensordot(x, w3, [[1], [0]]) * tf.reshape(x, [-1, 3, 1]
# linear term
y_quad += tf.matmul(x, w2) + b2

y_pred = tf.argmax(y_quad, axis=1)

loss_quad = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=

train_op_quad = tf.train.AdamOptimizer().minimize(loss_quad, var_list=[w2, w3, b2
```

实验曲线如下



实验重复了五次，最终结果和分类别结果如下表所示：

Type	Mean	Std	
0+1+2	88.27	1.10	
0	97.40	1.74	
1	75.00	2.83	
2	92.40	1.20	

使用二次分类器明显提高了正确率，同时大幅改善了1类的分类正确率，说明二次曲面比一次平面更加适合这个特征空间。

总结与分析

二次分类的效果是最好的，KNN次之，线性分类最差。三种方法都表现出了对于数据规模的敏感性。在实际的训练中，包括使用深度神经网络进行学习都需要注意类别之间的平衡。

运行代码

python hw4.py