实验1. 度量学习实验报告

2013302513，周小多，xiaoduo_zhou@163.com

2017 年 11 月 5 日

综述

机器学习中，物体的特征可以表示为维度空间的形式，每个空间对应了样本属性上定义的一个距离度量，而寻找合适的空间，实质上是在寻找一个合适的距离度量。如何找到一个合适的距离度量呢，这就是度量学习的基本动机。

任务1

度量函数学习目标

函数$f(A)$为样本正确分类的概率函数，即为优化目标：

$$f(A) = \sum_i \sum_{i \in C_i} p_{ij} = \sum_i p_i$$

其中：

$$p_{ij} = \frac{exp(- \parallel Ax_i - Ax_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel Ax_i - Ax_k \parallel^2)} \quad , \quad p_{ij} = 0$$

优化算法

利用梯度下降法可以通过迭代可以得到参数矩阵A的值：

$$A = A + \alpha \frac{\partial f}{\partial A}$$

其中：

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T \right)$$

α为步长

本程序参考了开源程序的实现，但开源程序只是单纯的用普通的梯度下降法实现，当数据量大时速度将会受到很大影响，本程序应用了随机梯度下降法，在每次迭代时对样本进行随机采样30个计算特征矩阵A，大大的提高了运行速度，在经过细致的参数调整后本程序效果也远超过开源程序。

程序实现如图1，效果如图2.

```python
def train(traindata):
    # 在此处完成你的训练函数，注意训练时间不要超过TRAINING_TIME_LIMIT(秒)。
    #time.sleep(1) # 这行仅用于测试训练超时，运行时请删除这行，否则你的TRAINING_TIME_LIMIT将-1s。
    train_data = traindata[0] # instances of training data
    train_lable = traindata[1] # labels of training data
    n, d = train_data.shape
    global A
    # Initialize A to a scaling matrix
    A = np.eye(d)#np.random.random([d,d])
    learning_rate = 0.001
    length = train_data.shape[1]
    for it in range(700):
        length = train_data.shape[0]
        a = range(0,length)

        n = random.sample(a,30)
        X=train_data[n]
        labels = train_lable[n]
        # Run NCA
        dX = X[:,None] - X[None]  # shape (n, n, d)
        tmp = np.einsum('...i,...j->...ij', dX, dX)  # shape (n, n, d, d)
        #print(labels[:,None] == labels[None] )
        masks = labels[:,None] == labels[None]
        for i, label in enumerate(labels):
            mask = masks[i]
            Ax = A.dot(X.T).T  # shape (n, d)
            softmax = np.exp(-((Ax[i] - Ax)**2).sum(axis=1))  # shape (n)
            softmax[i] = 0
            #sum = softmax.sum()
           # if sum == 0: sum = 0.0001
            softmax /= softmax.sum()
            t = softmax[:, None, None] * tmp[i]  # shape (n, d, d)
            #print(softmax[:, None, None] )
            d = softmax[mask].sum() * t.sum(axis=0) - t[mask].sum(axis=0)
            A += 2*learning_rate * A.dot(d)
    return 0
```

图 1: 代码实现

```
Euclidean_distance+knn(k=1):     0.000000
myDML.distance()+knn(k=1):       0.000000
Euclidean_distance+knn(k=3):     0.020000
myDML.distance()+knn(k=3):       0.015000
Euclidean_distance+knn(k=5):     0.030000
myDML.distance()+knn(k=5):       0.015000
```

图 2: 实现效果

任务2

在本任务中，，对数据集Letter Recognition Data Set进行预测通过随机梯度下降的方法，程序的运行速度明显加快，任务2 的数据量较大有30组数据，但运用本程序运行只需5分钟左右，效果较好，运行结果如图3

参考文献

1.Goldberger J, Roweis S, Hinton G, et al. Neighbourhood components analysis[C]// International Conference on Neural Information Processing Systems. MIT Press, 2004:513-520.

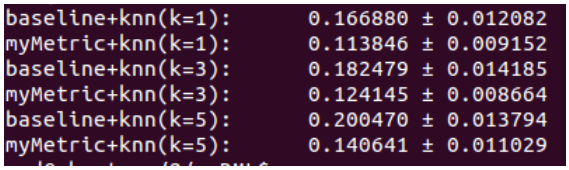
baseline+knn(k=1): 0.166880 ± 0.012082
myMetric+knn(k=1): 0.113846 ± 0.009152
baseline+knn(k=3): 0.182479 ± 0.014185
myMetric+knn(k=3): 0.124145 ± 0.008664
baseline+knn(k=5): 0.200470 ± 0.013794
myMetric+knn(k=5): 0.140641 ± 0.011029

图 3: 实现效果