ML-PJ1-KNN 文档

叶嘉睿
20307130048

2023 年 9 月 13 日

任务：使用 KNN 完成 MNIST 数据集分类任务

1 KNN

KNN 是一种惰性学习的监督学习方法。该方法不对训练集进行学习，而是在收到测试集后才开始计算。

KNN 的基本思路是：一个数据点所属的类就是与它在向量空间最近的 k 个点中数量最多的那个类。

2 任务实现

MNIST 数据集是 1*28*28 的图片，将其归一化后转化为一维向量，完成数据点到向量空间的映射。对于每个查询的测试样本，计算它与训练集中每个点的距离。这里距离直接使用欧氏距离。完成计算后，选取距离最小的 k 个点进行投票，多数表决决定测试样本的类别。

3 Baseline

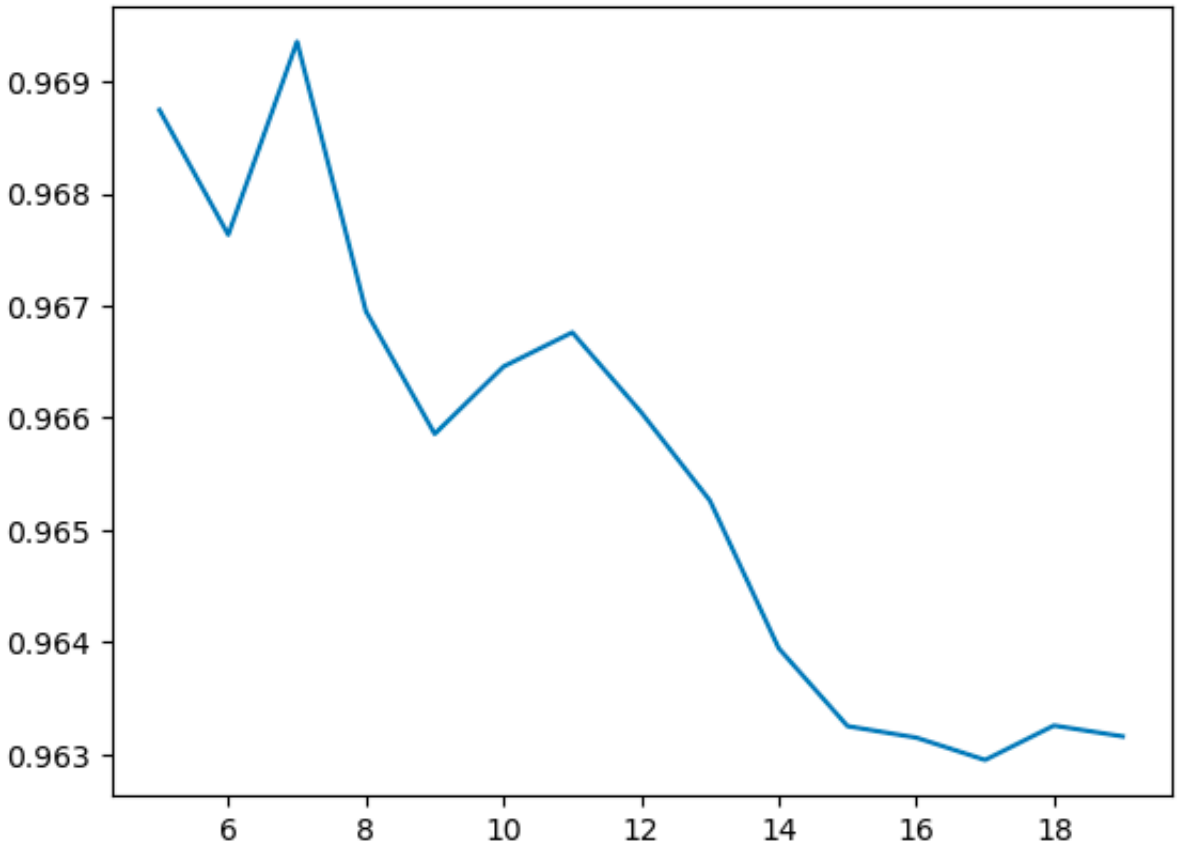
k=10

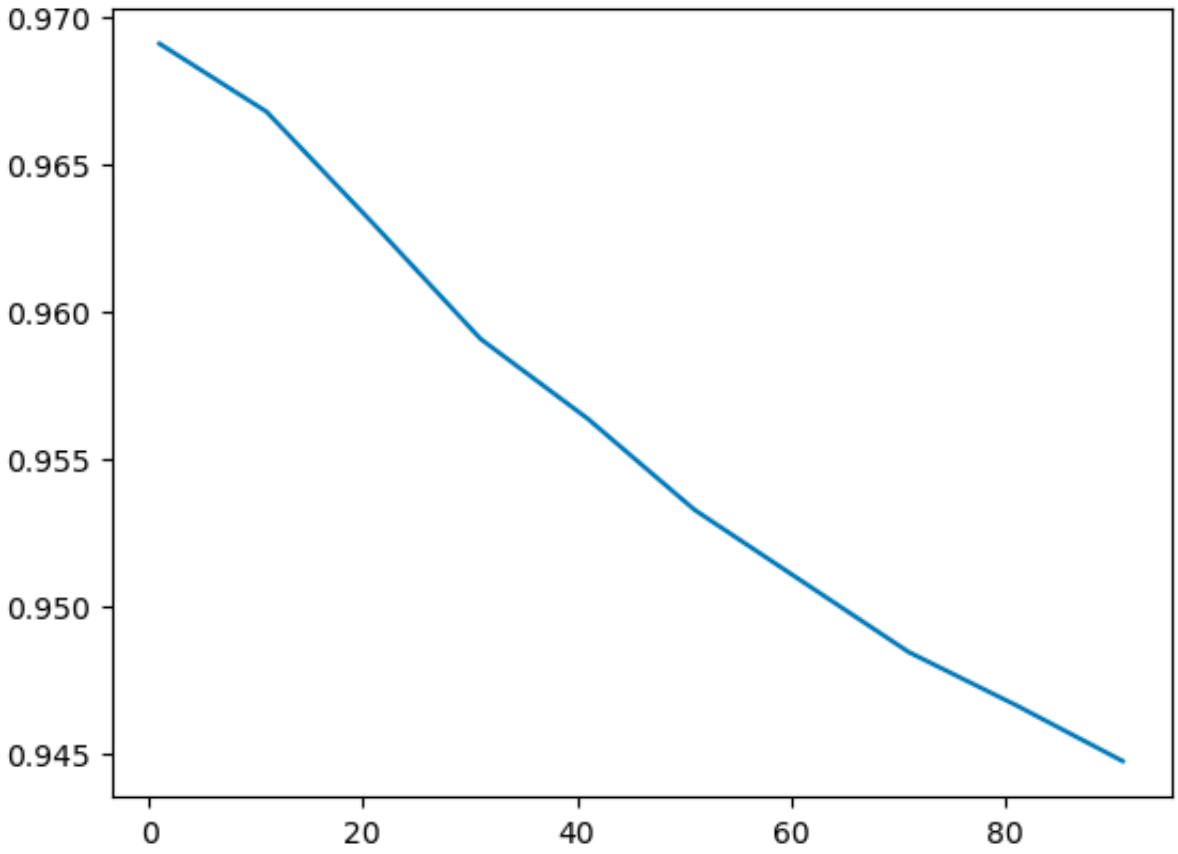
	precision	recall	f1-score	support
0	0.96	0.99	0.98	980
1	0.94	1.00	0.97	1135
2	0.98	0.95	0.97	1032
3	0.97	0.97	0.97	1010
4	0.97	0.96	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.98	0.98	958
7	0.96	0.96	0.96	1028
8	0.99	0.94	0.96	974
9	0.95	0.95	0.95	1009
accuracy			0.97	10000

```
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

4 k 值对算法影响

算法中 k 的值是人为事先指定的，不同的 k 值会得到不同分类结果。实验中选取不同 k 值作为变量，观察 f1-score。





可以看到在本例中，k 的值在 7 左右时最佳。

5 优化

考虑到逐个计算距离效率较低，故采用矩阵计算进行优化。

```
def euc_dist_matrix(matrix1,matrix2):
    size1=matrix1.size(0)
    size2=matrix2.size(0)
    m1=torch.pow(matrix1,2).sum(1,keepdim=True).expand(size1,size2)
    m2=torch.pow(matrix2,2).sum(1,keepdim=True).expand(size2,size1).t()

    return torch.addmm((m1+m2),matrix1,matrix2.t(),beta=1,alpha=-2).clamp(min=1e-12).sqrt()
```

此外，更改数据集到向量空间的映射方法与距离计算方法均可影响 knn 在特定数据集上的表现。