机器学习 hw3

理论题

1. 试把 k 均值的目标函数进行变换，使得表达式中每项只包含$x^T x$形式

Sol. 记样本集为$D = \{x_1, x_2, \cdots, x_n\}$，kmeans 算法划分的类别为$C = \{C_1, C_2, \cdots, C_k\}$，则目标函数为

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \left|\left|x - \mu_i\right|\right|^2$$

其中，

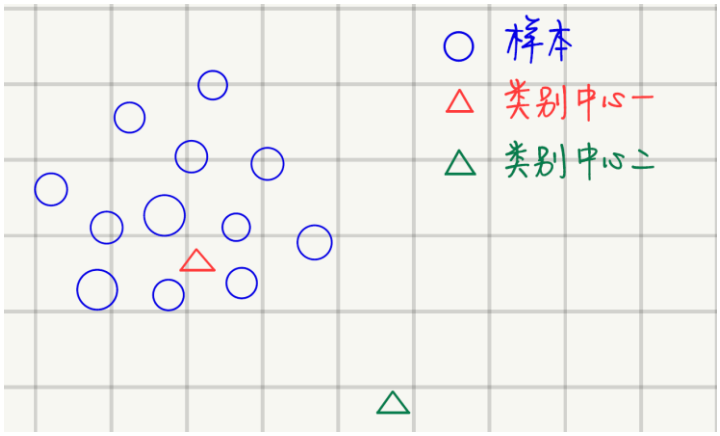
$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

为C_i的均值向量，即C_i类的中心。目标函数可表示为

$$\begin{aligned} J &= \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)^T (x - \mu_i) \\ &= \sum_{i=1}^{k} \sum_{x \in C_i} \left(x^T x - 2\mu_i^T x + \mu_i^T \mu_i\right) \\ &= \sum_{i=1}^{k} \left(\sum_{x \in C_i} x^T x - 2\mu_i^T \sum_{x \in C_i} x + |C_i| \mu_i^T \mu_i\right) \\ &= \sum_{i=1}^{k} \sum_{x \in C_i} x^T x - \sum_{i=1}^{k} |C_i| \mu_i^T \mu_i \end{aligned}$$

2. 如果 k 均值聚类过程中出现了错误提示某个类是空集，试画图例分析什么情况下会造成这种错误以及对应的改正方法

Sol.



某个类空集出现在选取某一点为类别中心时，没有样本属于这一类，示例如上图所

示。当随机选择类别中心点时，一个中心点离样本很远，在更新类别时，所有的样本都被归为另一类，而该类为空集并无法更新类别中心。
改正方法：在初始化类别中心时，随机选择 k 个样本作为中心；若在聚类过程中出现某一类为空集，则在样本中随机选择一个作为新的类别中心，或在样本数最多的类别中选择离样本中心点最远的一个样本作为新的类别中心。

实践题

1. 编程实现 MDS“求解方法二”的算法，并分别对 MNIST12 数据集做二维和三维降维
Sol. 根据 MDS 求解方法二的理论推导，先计算出样本的距离矩阵D，再求出 B 矩阵，对 B 进行特征值分解，选择前 k 大的特征值及其特征向量，计算降维后的低维样本坐标。算法如下所示

输入： 距离矩阵 $\mathbf{D} \in \mathbb{R}^{m \times m}$, 其元素 $dist_{ij}$ 为样本 $\boldsymbol{x}_i$ 到 $\boldsymbol{x}_j$ 的距离;
低维空间维数 d'.

过程：

1: 根据式(10.7)–(10.9)计算 $dist_{i\cdot}^2$, $dist_{\cdot j}^2$, $dist_{\cdot\cdot}^2$;
2: 根据式(10.10)计算矩阵 $\mathbf{B}$;
3: 对矩阵 $\mathbf{B}$ 做特征值分解;
4: 取 $\tilde{\mathbf{\Lambda}}$ 为 d' 个最大特征值所构成的对角矩阵, $\tilde{\mathbf{V}}$ 为相应的特征向量矩阵.

输出： 矩阵 $\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}^{1/2} \in \mathbb{R}^{m \times d'}$, 每行是一个样本的低维坐标

其中

$$D_{i\cdot} = \sum_{j=1}^{n} D_{ij}^2$$

$$D_{\cdot j} = \sum_{i=1}^{n} D_{ij}^2$$

$$D_{\cdot\cdot} = \sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}^2$$

$$B_{ij} = \frac{1}{2n}(D_{i\cdot} + D_{\cdot j} - nD_{ij}^2 - \frac{1}{n}D_{\cdot\cdot}$$

降维后的结果如下图所示

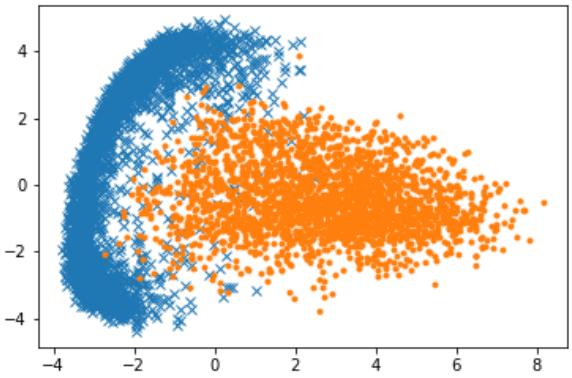


图 2. MNIST12 数据集二维降维

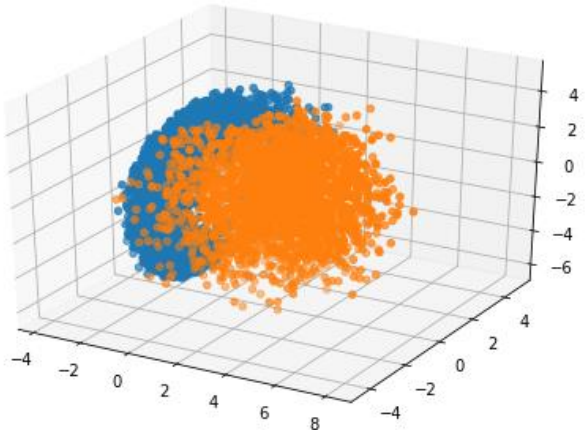


图 3. MNIST12 数据集三维降维

2. 对 k 均值算法进行提速，要求比课程演示中的 kmeans2 版本快一倍

Sol. K-mean 聚类算法分两步交替迭代，直到收敛

第一步类别划分，把每个样本划分到距离最近的中心点类

第二步计算均值，把每类的样本平均值作为新的类别中心

课堂上实现的 kmeans1 套了三层循环，对迭代次数、每个样本、每个点进行循环，在计算样本与中心的距离时，一个个进行计算，效率较低。而 kmeans2 在 kmeans1 的基础上进行改进，利用了 numpy 向量化的高效性，对每个点进行循环，计算每个点与所有类别中心的距离，找出最小的一个。Kmeans2 比 kmeans1 效率提升了一倍。

根据 kmeans2 的改进思路，我对 kmeans2 进一步改进。Kmeans 进行聚类时，样本数一定远大于类别数，对每个类别进行循环，计算出每一类的中心点与所有样本之间的距离，再选择最小距离。这样更充分的利用了向量化的高效性，对于每次迭代，循环次数由原来的 2000 次减少至 2 次。实现代码如下，与 kmeans2 进行对比，速度提升了 100 倍。

```
1.  def kmeans3(samples, K, max_iter=20):
2.
3.      # vectorized implementation, high efficiency
4.      N, D = samples.shape
```

```
5.
6.      idx = np.random.randint(N, size=K)
7.      centers = samples[idx]
8.      labels = -np.ones(N)
9.
10.     # to store distance, each line is distance to every center
11.     dist = np.zeros((N,K))
12.
13.     for i in range(max_iter):
14.
15.         # step 1: update labels
16.         # the adaption to kmeans2 is that the number of circulation is reduc
    ed to number of centers
17.         for c in range(K):
18.             center = centers[c]
19.             dist[:, c] = np.sum((samples-center)**2, 1)
20.
21.         labels = np.argmin(dist, axis=1)
22.
23.         # step 2: update centers
24.         for k in range(K):
25.             centers[k] = np.mean(samples[labels==k])
26.
27.     return centers, labels
```

将 sklearn 库的 kmeans 函数、课堂上实现的 kmeans1、kmeans2 以及自己实现的 kmeans3 进行对比，迭代 100 次，发现自己实现的 kmeans3 能达到和 sklearn.cluster.KMeans 同样的速度，比 kmeans2 快了 100 倍之多

In [12]:
```
print('Execution time for %s is %f s' % ('sklearn.cluster.KMeans', t_sklearnkmeans))
print('Execution time for %s is %f s' % ('kmeans1', t_kmeans1))
print('Execution time for %s is %f s' % ('kmeans2', t_kmeans2))
print('Execution time for %s is %f s' % ('kmeans3', t_kmeans3))
```

```
Execution time for sklearn.cluster.KMeans is 0.015625 s
Execution time for kmeans1 is 2.875000 s
Execution time for kmeans2 is 1.625000 s
Execution time for kmeans3 is 0.015625 s
```