K-Means和GMM算法在MNIST数据集上的应用

19335015 陈恩婷

概述

本实验仅用了python的numpy和scipy工具包，手动实现了K-Means算法和EM算法训练的GMM模型，在MNIST数据集上进行训练，尝试了不同的参数初始化方法、优化方法以及其他的训练技巧，并探究了他们对于聚类性能的影响。

K-Means和GMM的算法流程

K-Means算法

根据课本和课件上的讲解，K-Means算法的流程如下：

1. 选取K个初始质心 μ_k for k = 1,⋯,K, 计算出每一个数据点 $x^{(n)}$ 和所有质心 μ_k之前的距离
2. 将每个数据点 $x^{(n)}$归到最近的质心所对应的cluster下

$$r_{nk} = \begin{cases} 1,\ if\ k = \arg\min_{\theta} \|x^{(n)} - \mu_j\|^2 \\ 0,\ otherwise \end{cases}$$

3. 计算出相同cluster下的所有数据点的平均值，作为新的质心的位置

$$\mu_k \leftarrow \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$

4. 重复2，3步骤的计算，直到收敛

GMM算法

GMM是K-Means的一个延申，将每一个cluster推广为一个高斯分布。它和K-Means的硬聚类算法不同，的EM算法如下：

1. 初始化平均值μ_k, 协方差 Σ_k 和混合系数π_k, 并计算log-likelihood的初始值
2. **E 步骤**. 用目前的参数值计算每个高斯分布的权值

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)}$$

3. **M 步骤**. 用目前的权值重新计算参数值

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk}) x_n$$

$$\sum_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

其中

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

4. 计算log likelihood

$$\ln p(X|\mu, \Sigma, \pi) = \sum_{k=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k)\}$$

5. 重复2至4步骤的计算，直到收敛

训练方法

本实验采用了K-Means算法和EM算法训练的GMM模型，在MNIST数据集上进行训练，尝试了不同的参数初始化方法、优化方法以及其他的训练技巧，并探究了他们对于聚类性能的影响。

数据集

本实验使用了**MNIST数据集**。MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST). 训练集 (training set) 由来自 250 个不同人手写的数字构成，其中 50% 是高中学生，50% 来自人口普查局 (the Census Bureau) 的工作人员。测试集(test set) 也是同样比例的手写数字数据组成。MNIST数据集中的所有图片都为28*28像素的灰度图像。

本实验直接将每张图片展开成784 维，在784维的向量空间上进行聚类操作。

质心初始化与协方差

本实验采用了K-Means算法和EM算法训练的GMM模型，其中K-Means算法初始化的方法包括：

1. Random(随机从训练集选取10个点作为初始质心)
2. Random from each label(从每一label中随机选)
3. Distance based(随机选取第一个初始质心，选离最近中心点的距离最远的样本点作为下一个初始质心)

GMM模型则有两种初始化方法：

1. Random(随机从训练集选取10个点作为初始质心)
2. Data classes mean(将每一label的平均值作为初始质心)

GMM模型采用了两种协方差：

1. ‘full’: each component has its own general covariance matrix.
2. ‘diag’: each component has its own diagonal covariance matrix.

训练流程

训练的主要流程如下：

1. 初始化质心
2. 用相应的算法，在训练集上进行训练
3. 在测试集上测试模型，并记录聚类精度(clustering accuracy)

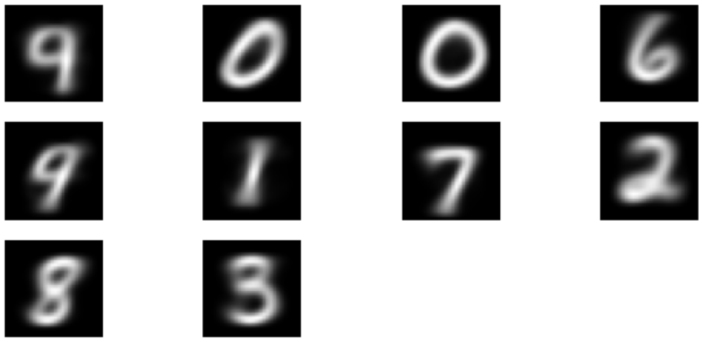
聚类算法的评价标准

本实验采用了聚类精度作为聚类算法的评价和比较标准，其中聚类精度的定义为聚到一类的样本中标签相同的比例。其中K-Means在测试集中的归类方法非常简单，只需要找出最近的质心即可；而GMM中则是选出log-likelihood最大的一个高斯分布作为分类结果。

实验结果和讨论

K-Means算法

Random初始化



		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	3	423	448	30	3	3	1	2	18	49
	1	0	0	0	3	0	1110	0	14	5	3
	2	23	20	3	24	7	139	15	710	29	62
	3	17	19	1	8	5	61	4	40	141	714
	4	492	1	0	23	434	28	0	4	0	0
	5	54	41	9	21	132	95	4	5	238	293
	6	24	24	21	794	2	55	0	18	17	3
	7	167	2	1	1	58	83	696	15	5	0
	8	26	11	6	11	44	76	14	7	577	202
	9	475	10	3	2	427	25	42	2	15	8

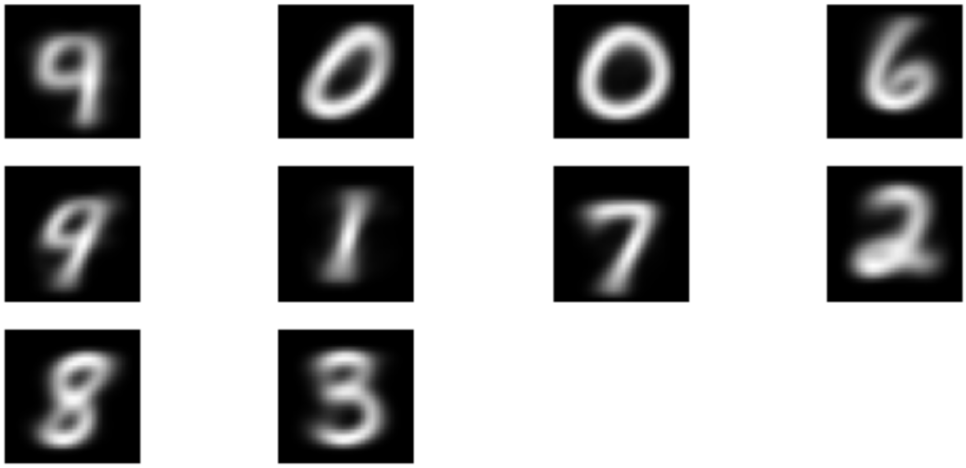
如图所示，表格的每一列代表一个Cluster，每一行代表一个真实标签，可见数码1，3，6，7的聚类效果比较好，而4则与9混淆了，5与8和3混淆了，0则被聚到了两个cluster下。总的聚类精度为**0.6398**。

Distance Based算法

		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	2	426	3	43	2	8	20	426	2	48
	1	1107	0	2	4	13	6	0	0	0	3
	2	142	3	27	19	711	28	7	20	10	65
	3	53	1	21	13	42	138	4	14	8	716
	4	37	0	545	40	2	2	38	1	317	0
	5	112	7	65	42	2	255	8	33	72	296
	6	25	9	8	425	1	4	469	14	0	3
	7	84	1	318	0	13	6	1	1	604	0
	8	77	6	35	8	8	573	7	11	36	213
	9	26	3	555	2	2	13	11	10	380	7

如图所示，可见数码1，2，3，7的聚类效果比较好，而4则依然与9混淆了，5与8和3混淆了。总的聚类精度为**0.6012**

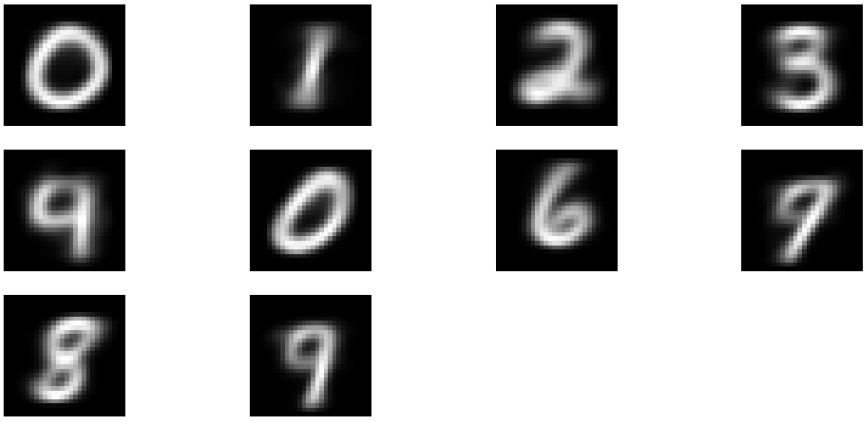
Random from Each Label初始化



		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	781	0	4	57	14	74	41	2	5	2
	1	0	660	0	3	0	1	2	0	469	0
	2	13	50	709	76	34	24	20	11	86	9
	3	4	58	38	685	7	178	9	8	5	18
	4	0	18	3	0	366	2	23	249	24	297
	5	6	20	4	297	28	316	23	56	92	50
	6	16	24	8	1	125	26	737	0	19	2
	7	1	36	11	0	76	1	1	425	48	429
	8	7	33	6	250	18	525	10	34	47	44
	9	7	16	2	7	225	13	2	241	7	489

如图所示，可见数码0，1，2，3，6的聚类效果比较好，4与9，3与5，8产生了一些混淆，总的聚类精度为**0.5846**

Data classes mean初始化



		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	442	3	2	49	6	425	30	2	18	3
	1	0	1109	12	3	0	0	3	0	5	3
	2	3	139	712	64	27	20	20	10	27	10
	3	1	56	42	713	6	16	9	8	139	20
	4	0	28	4	0	378	1	23	267	0	281
	5	10	109	4	299	28	37	22	70	265	48
	6	23	54	14	2	79	17	753	0	15	1
	7	0	70	12	0	87	1	1	410	1	446
	8	6	77	7	219	18	11	10	29	563	34
	9	2	18	2	6	253	11	2	224	9	482

如图所示，可见数码1，3的聚类效果比较好，这次其中4和9，7和9，5和3，8容易混淆，0被聚到了两个cluster下。总的聚类精度为**0.5987**。

GMM算法

diag + random

		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	3	25	8	35	1	120	548	189	0	51
	1	0	5	0	1	946	95	0	82	0	6
	2	34	306	6	177	2	19	134	154	1	199
	3	85	48	25	11	21	48	204	373	28	167
	4	188	11	85	9	79	219	91	168	92	40
	5	32	17	103	8	14	380	153	148	9	28
	6	0	423	2	114	6	39	9	363	0	2
	7	65	1	192	2	118	42	8	17	553	30
	8	74	5	84	11	107	232	110	314	12	25
	9	113	0	65	2	214	50	17	38	503	7

如图所示，可见diag+random的组合性能比较差，只有在数码1上聚类效果比较好，总的聚类精度**0.3979**。

diag + data class mean

		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	541	0	30	19	32	153	23	1	180	1
	1	1	947	2	5	1	48	10	1	98	23
	2	151	1	395	230	25	19	89	4	115	3
	3	19	1	38	405	42	44	22	27	373	39
	4	140	0	18	81	75	201	10	47	140	270
	5	100	0	16	86	59	396	18	26	171	20
	6	50	5	166	1	1	15	683	0	36	1
	7	4	0	4	14	44	39	0	389	50	484
	8	79	10	10	85	37	193	3	35	449	73
	9	11	4	3	21	31	54	0	169	35	681

如图所示，初始化方法换用data class mean后，聚类的性能有了很大的提升。虽然聚类精度只有**0.4961**，但是在表格中我们可以看到，每个cluster都能正确对应到一个label

full + data class mean

		Cluster									
		0	1	2	3	4	5	6	7	8	9
Label	0	629	0	11	16	8	28	29	1	258	0
	1	1	999	3	5	1	21	12	0	89	4
	2	47	4	490	209	20	13	19	2	128	2
	3	19	7	24	381	42	28	11	6	433	40
	4	14	4	15	7	303	111	17	5	338	174
	5	83	4	12	49	29	438	18	6	240	14
	6	29	13	58	0	2	22	795	0	39	0
	7	1	4	5	14	45	55	0	314	25	565
	8	56	45	8	55	36	181	3	2	552	36
	9	10	9	3	7	74	36	0	57	61	752

如图所示，可见0，1，6等数码的聚类效果比较好，每个cluster都能正确对应到一个label，总的聚类精度达到了**0.5653**。

实验结果与讨论

聚类精度的比较

1. 总的来说，K-Means和GMM都达到了60%左右的聚类精度，根据matplotlib包画出的K-Means聚类结果，可以看到聚类算法很容易混淆相似的数字（4，7，9互相混淆，3，8，5互相混淆）。
2. 总体来说，本实验的K-Means算法得到的测试集聚类精度要优于GMM，这可能的原因是实验中尝试过的初始方法不是非常适用于GMM，导致不同标签下的图片被聚类到了同一个cluster下面

初始化方法

1. GMM和K-means的迭代求解法非常相似（都是EM算法的具体化），所以会有相同的问题——有可能会收敛到一个局部最优解，不能保证每次都收敛到全局最优，如果运气比较差，取到不是特别好的初始值，就有可能得到相对不那么理想的结果。
2. K-Means算法中，不同初始化方法的聚类精度基本持平，其中Random的初始化方法稍微优于其他的初始化方法；而data classes mean更适用于GMM。

GMM的其他讨论

1. GMM算法的分类相比K-Means更准，也就是说，每一个不同cluster下最多图片的label都不相同。
2. GMM算法中，diagonal的协方差效果劣于full协方差，但是full协方差训练时间比diagonal要长很多，这是因为full的协方差矩阵更加复杂，计算复杂度高很多。

参考文献与链接

1. Pattern Recognition and Machine Learning
2. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html
3. https://blog.csdn.net/simple_the_best/article/details/75267863