A Variational EM Acceleration for Efficient Clustering at Very Large Scales

Florian Hirschberger, Dennis Forster and Jörg Lücke

摘要

如何从可能具有高维度 D 的大数据集 N 中高效地找出大量簇类 C 是当前大数据聚类的一个很重要的课题。在此文，我们通过使用一种新型变分方法来优化有对角协方差矩阵的高斯混合模型 GMM。该变分方法通过将剪枝先验概率作为变分分布以及 E 步 (EM 算法) 与核心集结合两种方法来近似期望最大化算法。经过算法与核心集的改进，在核心集中的运行时间从传统 EM 算法的每次迭代的 $O(NCD)$ 降低到了核心集变分 EM 算法的 $O(N'G^2D)$。其中，$N' < N$ 是核心集的大小，$G^2 < C$ 是剪枝参数。基于每次迭代的时间复杂度以亚线性级放缩，文章提供了一种具体的、可行的、并行的以及高效的聚类算法。

关键词：大数据；聚类；高斯混合模型；变分优化；期望最大化算法

1 引言

聚类是机器学习中一个非常核心且应用广泛的课题。随着数据量的快速增加以及数据驱动事务需求的快速增长，改进聚簇效率迫在眉睫。这个需求对具有许多高维数据点以及簇类个数非常大的数据集可能是非常重要的。[1-7]簇类算法常常被分为两种类型，分别是“基于密度”和“基于模型”。

基于密度的方法[8-9]往往在聚类形状上更灵活，而基于模型的方法往往更加高效。而基于模型中最出名的两个算法是高斯混合模型 (Gaussian Mixture Models) 和 K-Means 算法。[10-11]然而，GMMs 中期望最大化 (Expectation Maximization) 的执行，甚至是标准 K-Means 都难以适应大数据，因为他们的迭代时间开销都以 $O(NCD)$ 线性增长。其中，N 是数据点个数，C 是簇类个数，D 是数据维度。由于 N, C, D 都大大增加了，哪怕是执行一次迭代的代价都已经超过了当前最好的计算硬件上限。通过结合当前高效的变分近似以及高速核心集的方法，我们意在克服此限制。

经过模型的更新，EM 算法的每次迭代时间都会从 $O(NCD)$ 降低到 $O(N'G^2D)$，大大降低了时间复杂度，为大量数据的聚类降低开销，同时为大数据聚类提供较好的处理方案。在大数据聚类中，各个分片数据块都需要进行聚类，并最终进行聚合。通过优化迭代聚类的方法，可以降低大数据聚类的聚类计算开销。

2 相关工作

为了降低算法的复杂度，提高算法的效率，往往有几种常见方法：1. 在误差容许范围内尽可能减少数据量，增加抽样工作。2. 降低簇间信息更新的复杂度。3. 减少数据维度，降低计算复杂度。4. 降低计算迭代次数。

2.1 抽样减少数据量

核心集相关的研究都是通过权重子集取代全数据集进行计算，减少对数据集大小 N 的线性依赖。例如：Verbeek 等人[12]使用 kd 树对成组数据进行加速 GMM 聚类；随机优化技术也通过采样减少了对

数据集大小的依赖，但一般要与更长的收敛时间做权衡。[13-14]

2.2 降低簇信息更新复杂度

在使用 EM 算法的混合模型中，参数更新是期望最大化算法中 M 步的任务。通过降低参数更新步骤的计算复杂度，可以线性降低整个模型的复杂度。[15-18]变分 EM 方法降低了参数更新的复杂度，kd 树的构造与剪枝也能减少每个簇的计算开销。

2.3 减少数据维度依赖

通过降低数据维度 D 的方法，来降低算法处理数据的复杂度。例如：Otto 等人[7]首先使用前沿的卷积神经网络来减少图像数据的维度；随后出现了近似顺序聚类，一种使用了基于随机 kd 树的近似近邻计算的层次聚类[19]；另外，Gong 等人[20]首次将图片数据转为二进制哈希码，增加了聚簇效率。他们还展现了用给定数据集生成二进制码的工作。[21-22]

数据维度 D 的依赖性可以通过几何方法[23-25]或者随机投影[26]来展现。相关的研究包括：维度缩减的 GMMs——其使用了因素分析、PCA 方法来减少 GMM 的参数个数，以此来减少高维 D 在参数更新时的计算需求。[27-29]

2.4 减少收敛迭代次数

大多数神经网络中，迭代次数往往是作为超参数由用户定义。而迭代的次数是可以通过优秀的参数初始化方法来减少的，如:Seeding 方法[30]。在 EM 类型的算法中，迭代次数可以通过单调性控制[31]或步长优化[32]来减少。

3 本文方法

3.1 本文方法概述

本文使用 Coreset+ 截断式 GMMs 的方法来对传统 EM 算法求解 GMMs 的开销进行优化。通过核心集的构造来实现快速的近似计算，同时在 GMMs 的求解上加入截断系数，降低簇中心更新复杂度，以聚类问题规模的亚线性放缩，最终达成迭代时间复杂度的降低。

3.2 核心集构造

核心集定义如下：

定义 1. 核心集是原数据集的带权子集，并满足性质：在核心集中进行任何聚类的效果近似于在原数据集中进行的效果。对于 *k-means* 聚类问题，该性质表示如下：

$$|\phi_{\mathcal{X}}(Q)-\phi_{\mathcal{C}}(Q)| \leqslant \varepsilon\phi_{\mathcal{X}}(Q) \tag{1}$$

其中，$\mathcal{X}$ 表示原数据集，$\mathcal{C}$ 表示构造的核心集，ε 表示容许误差大小。

本篇通过构造核心集的方法来降低每次迭代的开销，而只需要增加一次构造核心集的开销，尤其在大数据集的情况下优化更明显。核心集使用 Lightweight Coreset(LWCS) 的构造方法，如下。

通过先验分布 $q=\frac{1}{2|\mathcal{X}|}+\frac{1}{2}\frac{d(x,\mu(\mathcal{X}))^2}{\sum_{x'\in\mathcal{X}}d(x',\mu(\mathcal{X}))^2}$ 来对总数据集 $\mathcal{X}$ 进行概率核心集抽样，抽样大小 m 满足 $m\geqslant c\frac{dk\log k+\log\frac{1}{\delta}}{\varepsilon^2}$。

其中，c 是一个正值常数，k 是簇的个数，$d(\cdot,\cdot)$ 是距离度量函数，可以是欧氏距离 $d(\vec{a},\vec{b}) = \left((a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_d-b_d)^2\right)^{\frac{1}{2}}$，也可以是相对距离度量 $d(a,b)=\dfrac{|a-b|}{a+b+v}, a,b>0$。超参数 δ 是置信度，ε 是容许误差大小，d 是数据维度。

随着样本的量的增加，LWCS 将更加近似总体数据集，抽样和计算开销也会增加。相对的，使用抽样减少数据量可以减少每次迭代的时间开销，为大数据聚类提供可能。

Procedure 1 Coreset Sampling Algorithm

Input: Dataset X
Output: Coreset C
$count = 0$;
for $x \in X$ **do**
  compute the sum of distance S;
  $count++$;
end
$\mu = S/count$;
$sum = 0$;
for $x \in X$ **do**
  $s = 0$;
  sample distribution $s = 0$;
  $s = x^2 + \mu^2 - 2 * x * \mu$
  $q[x] = s$;
  $sum = sum + s$;
end
for $x \in X$ **do**
  $q[x] = 0.5 * (q[x]/sum + 1.0/count)$;
end
draw coreset C from dataset with distribution $q[x]$

3.3 似然函数下界分析

对于给定的核心集 $\{y_n,\gamma_n\}_{n=1}^{N'}$ 以及一个带有 Θ 参数的生成模型 $p(y|\Theta)$，我们根据 Lucic 等人的贡献，提出核心集对数似然函数如下：

$$\mathcal{L}^c(\Theta)=\sum_{n=1}^{N'}\gamma_n \log\left(p\left(y_n|\Theta\right)\right) \tag{2}$$

对于此似然函数，我们使用变分推断方法进行分析处理。通过引入任意随机分布 $q_n(c)$ 表示各点所属分布，我们可以定义出似然函数下界如下：

$$\mathcal{F}^c(q,\Theta)=\sum_{n=1}^{N'}\gamma_n\left(\left(\sum_{c=1}^{C}q_n(c)\log p\left(c,y_n|\Theta\right)\right)+\mathcal{H}\left[q_n\right]\right) \tag{3}$$

其中，$\mathcal{H}[q]$ 表示一个分布 q 的熵。该界仅在分布 $q_n(c)$ 与后验概率 $p(c|y_n,\Theta)$ 相同时达到，即 q 分布与 p 分布相同，KL 散度最小，界函数最大。本模型通过最大化式 (2) 来完成 EM 算法中参数更新的步骤。

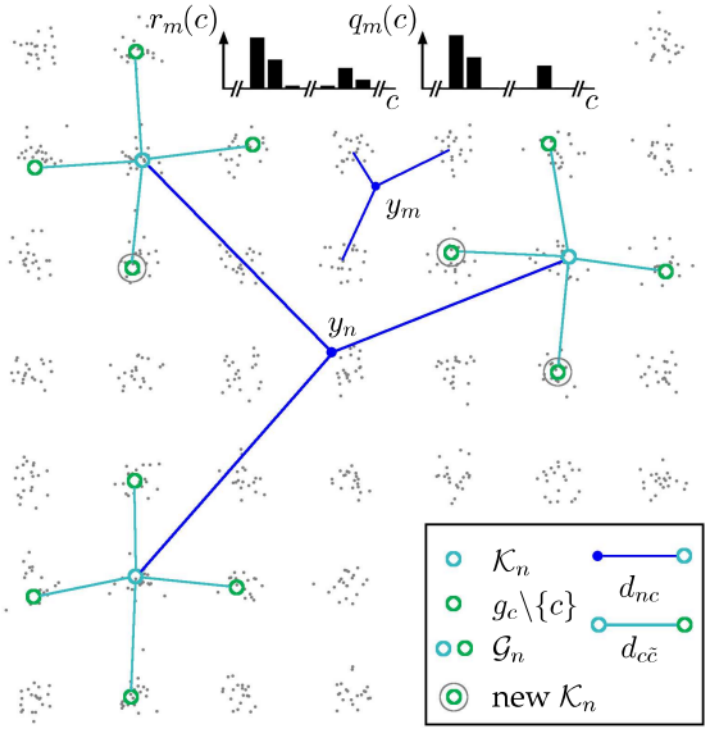


图 1: 搜索新的簇中心备选集

3.4 截断系数定义

截断系数 G 为用于寻找数据所属簇的备选簇个数。对于各数据点 y_n，有其备选所属簇集 $\mathcal{K}_n$，而截断系数就是各备选簇集的 G 个邻近备选簇，通过似然函数计算出最近备选簇，从而更新簇集分布情况。如图 1所示，对于数据点 y_n，从当前簇集中寻找其最接近的 $|\mathcal{K}_n|$ 个簇中心，并将簇中心最接近的其他簇心加入搜索空间中，最后从搜索空间中选择 $|\mathcal{K}_n|$ 个簇中心作为新的簇集。持续迭代后即可计算得到合适的概率簇集。

3.5 伪代码

Procedure 2 *vc-GMM Partial Variational E-Step*

for $n = 1 : N'$ **do**
  $\mathcal{G}_n = \bigcup_{c \in \mathcal{K}_n} g_c$; ▷ 获取各临近簇中心的备选簇集
  for $c \in \mathcal{G}_n$ **do**
    compute joints $p(c, y_n|\Theta)$; ▷ 计算后验概率
  end
  $\mathcal{K}_n = \{c|p(c, y_n|\Theta)$ is among the C' largest joints $\}$; ▷ 选取最高概率点作为可行簇中心
end

for $n = 1 : N'$ **do**
  $c_n = \arg\max_{c \in \mathcal{K}_n} p(c, y_n|\Theta)$; ▷ 该数据点最可能归属于该簇
  $\mathcal{I}_{c_n} = \mathcal{I}_{c_n} \cup \{n\}$; ▷ 收集该簇集中的数据点
end

for $c = 1 : C$ **do**
  for $\widetilde{c} = 1 : C$ **do**
    for $n \in \mathcal{I}_c$ **do**
      if $\widetilde{c} \in \mathcal{G}_n$ **then**
        $d^2_{c\widetilde{c}} = d^2_{c\widetilde{c}} - \log p(\widetilde{c}, y_n|\Theta)$; ▷ 计算簇与备选簇的距离度量差
        $|\mathcal{I}^*| = |\mathcal{I}^*| + 1$;
      end
    end
    $d^2_{c\widetilde{c}} = d^2_{c\widetilde{c}}/|\mathcal{I}^*|$;
    $|\mathcal{I}^*| = 0$;
  end
  $g_c = \{\widetilde{c}\ | d^2_{c\widetilde{c}}$ is among the $G - 1$ smallest values $d^2_{c:}$ with $\widetilde{c} \neq c\} \cup \{c\}$;
  ▷ 将距离度量最小的备选簇截断出来作为下一步迭代的备选簇
end

for $n = 1 : N'$ **do**
  for $c \in \mathcal{K}_n$ **do**
    $p_n^{sum} = p_n^{sum} + p(c, y_n|\Theta)$; ▷ 求总和
  end
  for $c \in \mathcal{K}_n$ **do**
    $q_n(c; \mathcal{K}, \hat{\Theta}) = p(c, y_n|\Theta)/p_n^{sum}$; ▷ 求平均作为分布
  end
end

4 复现细节

4.1 与已有开源代码对比

开源代码中有关于 vc-GMM 的 C++ 代码实现，但没有 vc-GMM 的 python 代码实现。Coreset 的构造可以通过 Python 代码进行实现，并以此加速 variational GMM 在 Python 语言平台上的运行速度。

本篇代码引用原文章 A Variational EM Acceleration for Efficient Clustering at Very Large Scales 的 C++ 代码与部分 Python 代码。源码可以在 Github 网站 https://github.com/variational-sublinear-clustering 上下载，也可以在 Git Bash 等 Git 平台中使用 Git clone https://github.com/variational-sublinear-clustering/vc-

GMM.git 命令下载 C++ 版本的 vc-GMM 代码，或 Git clone https://github.com/variational-sublinear-clustering/var
GMM.git 命令下载 Python 版本的 var-GMM 代码。

4.2 实验环境搭建

本实验使用 Pycharm 作为集成开发环境，开发库包版本如下：

- Python : 3.6.13
- afkmc2 : 0.1
- h5py : 3.1.0
- kmc2 : 0.1
- matplotlib : 3.3.4
- mpi4py : 3.1.4
- numpy : 1.19.5
- panda : 0.3.1
- scikit-learn : 0.24.2
- scipy : 1.5.4

运行主机环境配置：处理器为 Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz，核显 Intel(R) UHD Graphics 630。

4.3 创新点

核心集的构造可以在误差范围内极大程度地减少算法对大数据集的处理时间。在满足采样大小为 $m=\Theta\left(\frac{dk\log k+\frac{1}{\delta}}{\varepsilon^2}\right)$ 的范围内，可以满足核心集的泛化误差界限 $|\phi_{\mathcal{X}}(Q)-\phi_{\mathcal{C}}(Q)|\leqslant\varepsilon\phi_{\mathcal{X}}(Q)$。在实际工程中，可以通过手动设置抽样大小或设置模型超参 δ,ε 两种方式控制采样大小。通过 Python 代码对 C++ 代码的复现，完成 Pytorch 平台上的模型运行，并通过 matplotlib,pylab 等绘图库进行结果展示。

5 实验结果分析

本文使用 BIRCH 数据集以及 CIFAR-100 数据集进行实验，通过运行时间对比，vc-GMM 与 V-GMM 的迭代时间与数据集大小 N 为线性关系。经过 Coreset 抽样，可以线性降低运行时间。

通过对核心集与全数据集进行聚类，可以得到图 3中的聚类结果。从图中可以对比出核心集的聚类结果显然没有全核心集的聚类结果那么方整，中间的部分簇集的聚类边界不明显。

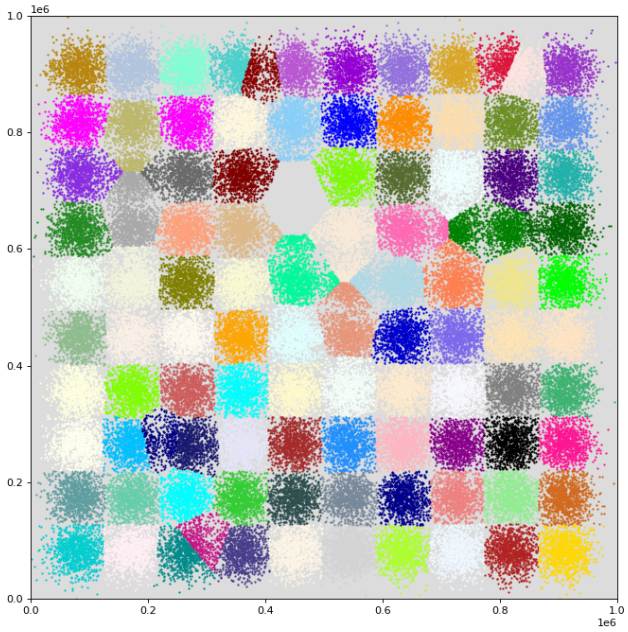
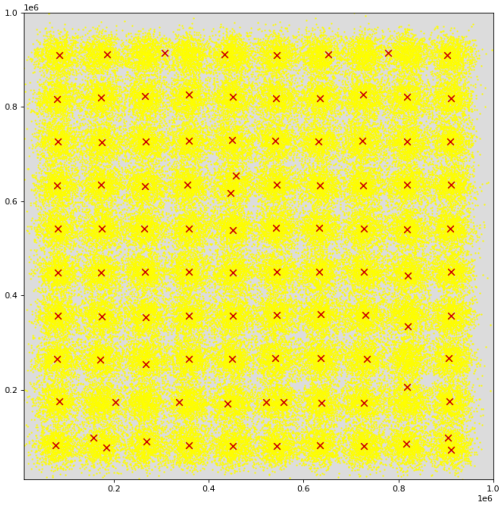
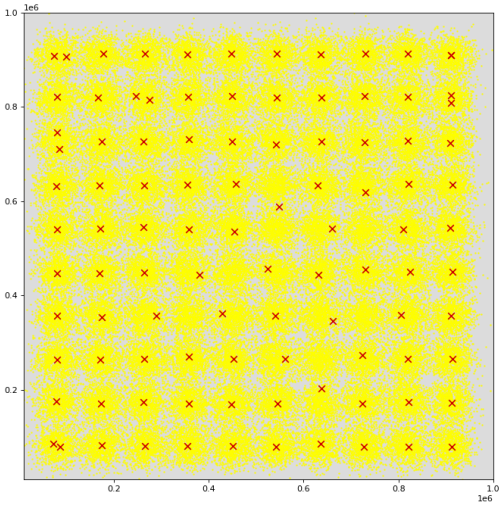


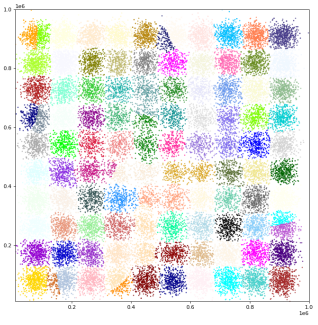
图 2: BIRCH 聚类结果



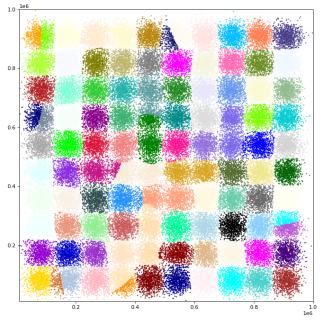
(a) 全核心集聚类结果



(b) 核心集聚类结果



(c) 核心集拟合模型



(d) 通过核心集拟合的模型预估全数据集

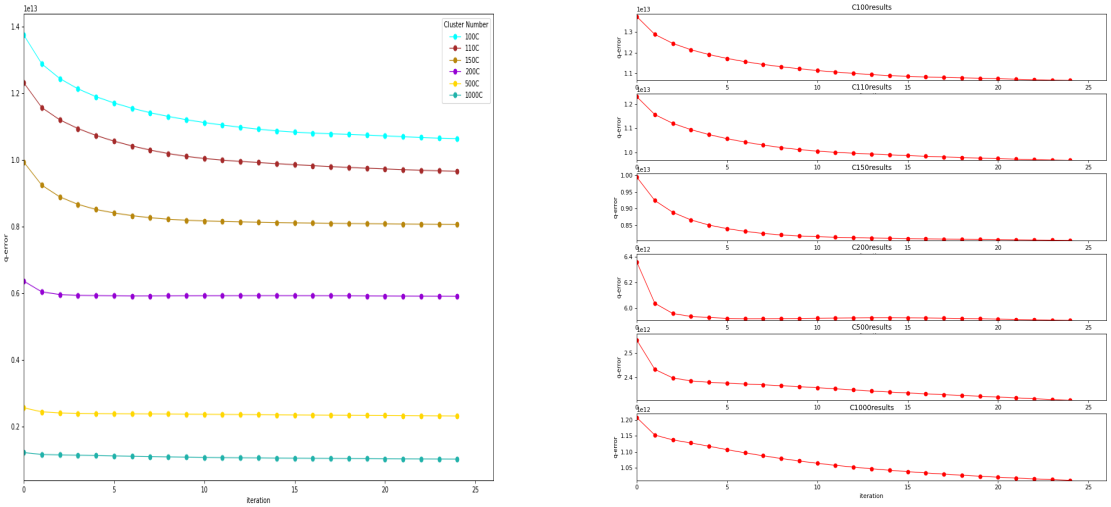
图 3: 聚类结果对比

图 3中展示了核心集采样的聚类结果（簇类中心及各数据点所属簇），不同颜色代表了不同的簇

类。模型选择了 100 个簇类作为模型参数，并进行此实验。

通过核心集采样，并对核心集进行模型训练，可以近似达到在全数据集中进行模型训练的结果。核心集与全数据集的误差大小取决于核心集的采样大小，核心集采样数量越大，越接近全数据集训练效果。

从图 3(c)可以看出在核心集中训练的模型去预估全数据集可以得到较好的区分度。而核心集的大小在此实验中仅为全数据集的 40%，大大减少了大数据集的训练开销。



(a) 总体对比　　(b) 局部对比

图 4: 不同簇类模型的误差收敛图

根据量化误差的变化图 4，我们可以得到基本一致的实验结果。量化误差值能较快地收敛，同时发现，簇类个数越多，模型就能把簇类边界收敛得越紧，也更快达到收敛。

6 总结与展望

文档使用了截断式高斯混合模型进行聚类，结合核心集的抽样技术，目的在于降低传统 GMM 聚类模型的迭代复杂度，通过一次核心集抽样的 $O\left(\frac{dk\log k+\frac{1}{\delta}}{\varepsilon^2}\right)$ 复杂度以及截断参数来降低传统模型中多次迭代 $O(NCD)$ 的耗时。

实现中发现的不足：由于 GMMs 进行聚类的结果是簇中心信息，数据所属簇类为较近簇中心的概率性分布，使用 GMMs 聚类的结果会出现边界不分明等情况，而且 GMM 容易达到收敛到局部最优解，难以确保全局最优。

根据似然函数决定数据所属簇可以相对分明的分离簇类边界，也许可以通过结合优化目标函数的方式来决定数据所属簇。

核心集大小的选取也是一大问题，尽管可以通过 Oliver Bachem 提出的分布与界限来选择核心集大小，但在实际情况中，期望得到较低的误差往往会极大增加核心集的大小，降低核心集采样的效率。如何选择合适的采样大小是工程中需要考虑的一个问题。

参考文献

[1] PELLEG D, MOORE A. Accelerating exact k-means algorithms with geometric reasoning[C]//Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999: 277-281.

[2] HILBERT M, LÓPEZ P. The world’s technological capacity to store, communicate, and compute information[J]. science, 2011, 332(6025): 60-65.

[3] COATES A, NG A, LEE H. An analysis of single-layer networks in unsupervised feature learning[C]//Proceedings of the fourteenth international conference on artificial intelligence and statistics. 2011: 215-223.

[4] CURTIN R R. A dual-tree algorithm for fast k-means clustering with large k[C]//Proceedings of the 2017 SIAM International Conference on Data Mining. 2017: 300-308.

[5] KOBREN A, MONATH N, KRISHNAMURTHY A, et al. A hierarchical algorithm for extreme clustering[C]//Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. 2017: 255-264.

[6] NECH A, KEMELMACHER-SHLIZERMAN I. Level playing field for million scale face recognition [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 7044-7053.

[7] OTTO C, WANG D, JAIN A K. Clustering millions of faces by identity[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(2): 289-303.

[8] ESTER M, KRIEGEL H P, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.[C]//kdd: vol. 96: 34. 1996: 226-231.

[9] CAMPELLO R J, KRÖGER P, SANDER J, et al. Density-based clustering[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2020, 10(2): e1343.

[10] BERKHIN P. A survey of clustering data mining techniques[G]//Grouping multidimensional data. Springer, 2006: 25-71.

[11] MCLACHLAN G J, LEE S X, RATHNAYAKE S I. Finite mixture models[J]. Annual review of statistics and its application, 2019, 6: 355-378.

[12] VERBEEK J J, NUNNINK J R, VLASSIS N. Accelerated EM-based clustering of large data sets[J]. Data Mining and Knowledge Discovery, 2006, 13(3): 291-307.

[13] JOHNSON R, ZHANG T. Accelerating stochastic gradient descent using predictive variance reduction [J]. Advances in neural information processing systems, 2013, 26.

[14] DEFAZIO A, BACH F, LACOSTE-JULIEN S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives[J]. Advances in neural information processing systems,

2014, 27.

[15] DAI Z, LÜCKE J. Autonomous document cleaning—a generative approach to reconstruct strongly corrupted scanned texts[J]. IEEE transactions on pattern analysis and machine intelligence, 2014, 36(10): 1950-1962.

[16] SHELTON J A, GASTHAUS J, DAI Z, et al. GP-select: Accelerating EM using adaptive subspace preselection[J]. Neural Computation, 2017, 29(8): 2177-2202.

[17] HUGHES M C, SUDDERTH E B. Fast learning of clusters and topics via sparse posteriors[J]. arXiv preprint arXiv:1609.07521, 2016.

[18] FORSTER D, SHEIKH A S, LÜCKE J. Neural simpletrons: Learning in the limit of few labels with directed generative networks[J]. Neural computation, 2018, 30(8): 2113-2174.

[19] MUJA M, LOWE D G. Scalable nearest neighbor algorithms for high dimensional data[J]. IEEE transactions on pattern analysis and machine intelligence, 2014, 36(11): 2227-2240.

[20] GONG Y, PAWLOWSKI M, YANG F, et al. Web scale photo hash clustering on a single machine[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 19-27.

[21] ERIN LIONG V, LU J, WANG G, et al. Deep hashing for compact binary codes learning[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 2475-2483.

[22] SHEN X, LIU W, TSANG I, et al. Compressed k-means for large-scale clustering[C]//Thirty-first AAAI conference on artificial intelligence. 2017.

[23] CHENG D Y, GERSHO A, RAMAMURTHI B, et al. Fast search algorithms for vector quantization and pattern matching[C]//ICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing: vol. 9. 1984: 372-375.

[24] BEI C D, GRAY R. An improvement of the minimum distortion encoding algorithm for vector quantization[J]. IEEE Transactions on communications, 1985, 33(10): 1132-1133.

[25] ELKAN C. Using the triangle inequality to accelerate k-means[C]//Proceedings of the 20th international conference on Machine Learning (ICML-03). 2003: 147-153.

[26] CHAN J Y, LEUNG A P. Efficient k-means++ with random projection[C]//2017 International Joint Conference on Neural Networks (IJCNN). 2017: 94-100.

[27] BOUVEYRON C, GIRARD S, SCHMID C. High-dimensional data clustering[J]. Computational statistics & data analysis, 2007, 52(1): 502-519.

[28] RICHARDSON E, WEISS Y. On gans and gmms[J]. Advances in Neural Information Processing Systems, 2018, 31.

[29] HERTRICH J, NGUYEN D P L, AUJOL J F, et al. PCA reduced Gaussian mixture models with applications in superresolution[J]. arXiv preprint arXiv:2009.07520, 2020.

[30] ARTHUR D, VASSILVITSKII S. k-means++: The advantages of careful seeding[R]. Stanford, 2006.

[31] HENDERSON N C, VARADHAN R. Damped Anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms[J]. Journal of Computational and Graphical Statistics, 2019, 28(4): 834-846.

[32] VARADHAN R, ROLAND C. Simple and globally convergent methods for accelerating the convergence of any EM algorithm[J]. Scandinavian Journal of Statistics, 2008, 35(2): 335-353.