

基于 AutoEncoder 的增量式聚类算法

原 旭,杨镇楠,赵 亮,陈志奎

(大连理工大学 软件学院,辽宁 大连 116620)

摘 要: 针对目前数据量增长迅速,数据特征多,存储空间不足等问题,提出了基于 AutoEncoder 的增量式聚类算法 (ANIC). 首先利用 AutoEncoder 学习数据样本的特征,进行低维特征整合,得到数据样本的压缩表示形式,然后在原有聚类结果的基础上,通过一遍式读取数据和动态更新聚类中心点,对新生成样本进行增量式聚类. 在对 UCI 的四个数据集进行聚类时,实验结果表明该算法能够得到优于 k 均值算法 (Kmeans) 和模糊 c 均值算法 (FCM) 的聚类效果. 同时,该算法的时间消耗低,能够识别离群点,能够有效地对不断增加的数据集进行增量式聚类.

关键词: AutoEncoder; 增量聚类; 特征学习; 数据压缩

中图分类号: TP301.6

文献标识码: A

文章编号: 1000-7180(2016)03-0120-05

Incremental Clustering Based on AutoEncoder

YUAN Xu, YANG Zhen-nan, ZHAO Liang, CHEN Zhi-kui

(School of Software Technology, Dalian University of Technology, Dalian 116620, China)

Abstract: The rapid growth of data result in a lot of problems such as the data have too many features, the lack of storage space etc. This paper propose a new incremental clustering algorithm based on AutoEncoder. Firstly, the AutoEncoder are used to learn the features of the data, integrate the low-dimensional feature, and get the reduced representation from the raw data. Then run incremental clustering on the new data base on the original clustering results by reading the data once and dynamically update clustering centers. Experimental results show that the proposed algorithm can obtain a comparable clustering performance with k-means algorithm (Kmeans) and fuzzy c-means algorithm (FCM) on the four data set in UCI database. Meanwhile, the time consumption of the proposed algorithm is low, it can achieve incremental clustering and identify the outliers for the increasing data set effectively.

Key words: AutoEncoder; Incremental clustering; feature learning; data compression

1 引言

AutoEncoder 是一种深度学习模型,用来学习一种对数据集的压缩且分布式的表示方法. 广泛应用于图像识别、语音识别、自然语言处理等领域. 此外, Autoencoder 能够有效提取样本特征,并对样本进行压缩,可用于样本聚类中.

文献[1-2]改进了 AutoEncoder 的目标函数,文献[3]将堆叠稀疏模糊 C 聚类算法与 AutoEncoder 相结合,解决模糊 c 均值聚类算法对孤立点、随机初始化聚类中心比较敏感的问题. 但它是对静态数据

集进行聚类,不能够对动态增加的数据集进行聚类.

因此,本文提出一种基于 AutoEncoder 的增量式聚类算法,针对不断增加的数值属性数据集,进行动态增量聚类. 在实验中,通过对 UCI 数据库中的四个数据集进行聚类,将该算法与目前最广泛使用的 Kmeans^[4] 算法和 FCM^[5-6] 算法进行比较,来验证本文算法的聚类效果. 此外,通过将数据集划分成多个数据集进行增量聚类,将聚类结果与文献[7-8]中的实验结果进行比较,来验证本文算法进行增量聚类的性能.

2 基于 AutoEncoder 的增量式聚类算法

2.1 AutoEncoder

AutoEncoder 是一种深度学习模型,是一种尽可能复现输入样本的神经网络,包括输入层、隐藏层、输出层。隐藏层通过组合输入层的特征形成更加抽象的高层特征,以发现数据的分布式特征表示。AutoEncoder 包括两个阶段:编码阶段和解码阶段。

编码阶段是前向传播过程,通过激活函数使得输入样本 x 映射到隐藏层 h 。激活函数公式如式(1)所示。

$$f(z) = \frac{1}{1 + \exp(-z)} \quad (1)$$

通过激活函数,计算样本 x 在隐藏层的表示形式 h 时,计算公式如式(2)所示。

$$h = f(x) = \frac{1}{1 + \exp(-(w_1 x + b_1))} \quad (2)$$

式中, w_1 为编码权重, b_1 为对应的偏置单元。

解码阶段是通过 h 重新构建输入输入样本 x , 得到 x 重构的表示形式 x' , 计算公式如式(3)所示。

$$x' = g(h) = \frac{1}{1 + \exp(-(w_2 h + b_2))} \quad (3)$$

式中, w_2 为解码权重, b_2 为对应的偏置单元。

对于给定的 N 个样本, 每个样本都进行如上的编码和解码阶段, 通过反向传导梯度下降的方法更新 w_1, w_2, b_1 和 b_2 使得目标函数最小。目标函数的表示形式如式(4)所示。

$$\frac{1}{N} \|x - x'\|^2 \quad (4)$$

深度自动编码器采用的是逐层训练的方式, 上一层的自动编码器的输出作为下一层自动编码器的输入。

2.2 基于自动编码器的增量式聚类算法

本文提出的基于自动编码器的增量式聚类算法是将自动编码器与增量式算法相结合, 针对不断增加的样本数量, 将新到达的样本集经过自动编码器的学习和压缩后, 对每个样本进行逐条聚类, 使得每个样本都能到达合适的类中。在对样本进行增量聚类时, 采用欧式距离来表示样本与聚类中心之间的距离。欧式距离的计算公式为式(5)。

$$d_{12} = \sqrt{\sum_{k=1}^n (x_{1k} - x_{2k})^2} \quad (5)$$

式中, d_{12} 表示两个样本 x_1 和 x_2 之间的欧式距离, n 表示两个样本的属性数目。

采用式(6)更新每个簇的簇心。

$$u_c = \frac{1}{m_c} \sum_{p \in U_c} p \quad (6)$$

式中, u_c 表示簇 c 的簇心, p 表示簇 c 中样本, m_c 表示簇 c 中的样本数目。

为了使得样本中的每个属性能够在同等条件下进行比较, 样本集进入自动编码器前采用式(7)对样本集进行归一化预处理。

$$x_i^j = \frac{x_i^j - x_{\min}^j}{x_{\max}^j - x_{\min}^j} \quad (7)$$

式中, x_i^j 表示样本 x_i 的第 j 个属性,

$$x_{\min}^j = \min(x_1^j, x_2^j, \dots, x_N^j),$$

$$x_{\max}^j = \max(x_1^j, x_2^j, \dots, x_N^j).$$

算法 ANIC 的伪代码如 Algorithm 1 所示。其中参数 k 表示簇的最小数目; data 表示包含 m 个样本、 n 个属性的数据集; proximity 表示临近度矩阵, 用来存储簇心之间的距离; u 是存储簇心的集合; re 表示已经完成聚类的样本集合, 每个样本都有相应的所属簇的标号; ae 表示自动编码器; Sum 表示每个簇中所含样本的属性值; Num 表示每个簇中所含样本的数目。从 ANIC 的伪代码中可以看出算法 ANIC 主要步骤如下所示。步骤 1: 数据集归一化预处理; 步骤 2: 训练自动编码器; 步骤 3: 得到数据集的压缩表示形式; 步骤 4: 对数据集进行增量聚类。

Algorithm 1 ANIC

Input:

k ; data(包含 n 个样本, m 个属性); proximity; u ;
re; ae; Sum; Num;

Output:

proximity_matrix; u ; re; Sum; Num; ae;

Step1: /* 对样本进行归一化预处理 */

for $i = 1$ to m do

min = data(1, i);

max = data(1, i);

for $j = 2$ to n do

if min > data(j , i) then

min = data(j , i);

end

if max < data(j , i) then

max = data(j , i);

end

end

for $j = 1$ to n do

data(j , i) = (data(j , i) - min) / (max - min);

end

end

end

```

Step2:/* 训练自动编码器 */
    aetrain( ae, data );
end
Step3:/* 得到样本的压缩表示 */
    for j = 1 to n do
        x ← 通过式(2)得到 data 中每个样本的压缩表示;
    end
end
Step4:/* 对样本进行增量聚类 */
    for i = 1 to n do
        [ u, re, Num, Sum ] = NIC( xi, u, re, Num, Sum, k )
    end
end

```

步骤1对数据集进行归一化预处理的目的是使得样本的每个属性能够在同等条件下进行比较,因此采用式(7)对样本的每个属性进行归一化。

步骤2是使用新数据集对自动编码器进行训练,训练的过程是不断编码和解码的过程,使得目标函数(4)的值最小,采用反向传导梯度下降的方法更新编码权重及相应的偏置项、解码权重及响应的偏置项,使得重构的样本与原样本的差尽可能小. 初始情况下,为自动编码器随机指派初始权重,通过数据集对自动编码器训练得到新的权重,将该权重作为下一个数据集训练自动编码器的初始权重,即上一个数据集训练得到的权重作为下一个数据集训练的初始权重. 当数据集的属性较多时,构建深度自动编码器,这样能够更好地学习数据集的分布式特征表示,深度自动编码器是由多个自动编码器构成,采用逐层训练的过程,上一层自动编码器的输出作为下一层自动编码器的输入。

步骤3是在步骤2训练得到自动编码器的编码权重后,通过式(1),对数据集进行前向传播,得到数据集在隐藏层的表示形式,得到新的数据集 x , x 中的样本是原数据集的另一种表示形式,并且属性数目少于原数据集样本的属性数目. 是原数据集的一种更加抽象的压缩表示形式. 对于深度自动编码器,因为其包含多个隐藏层,因此,最后一层隐藏层的输出作为新的压缩数据集 x 。

步骤4是对新数据集 x 进行增量聚类. 聚类算法的伪代码如 Algorithm 2 所示. 对于每个样本,首先判断其到达的顺序,如果是前 k 个到达的样本,则将这 k 个样本用于初始化 k 个簇心,并利用式(5)计算簇心之间的距离并存储在临近度矩阵中. 如果是 $k+1$ 之后的样本,则利用式(5)计算样本与各个簇心之间的距离,并得到距离样本最近的簇心 c 及距离值 $\min_d$. 查找临近度矩阵,得到距离最近的两个

簇心 m 、 n 及其之间的距离 $\min$ 以及最远的簇心距离 $\max$. 判断 $\min_d$ 与 $\min$ 和 $\max$ 之间的关系来对样本进行处理。

如果 $\min_d > \max$, 则将该样本独立为一个新簇,这样能够有效识别离群点,计算该新簇与其他簇心的距离并存储在临近度矩阵中;

如果 $\min_d \leq \min$, 则将样本加入到距离其最近的簇 c 中,并更新 c 的簇心及临近度矩阵;

否则的话,将距离最近的两个簇 m 、 n 合并为一个新簇,并将样本独立为一个新簇。

2.3 算法复杂度分析

从2.2节中的算法可以看出,步骤1对数据集进行归一化需要循环处理数据 $m * (2n - 1)$ 次;步骤2假设 AutoEncoder 的输入层对应 m 个神经元,隐藏层对应 m_h 个神经元,则需要循环 $2n * m * m_h$ 次完成对 AutoEncoder 的训练;步骤3循环处理 $n * m * m_h$ 次得到 n 个样本的压缩表示;步骤4循环调用 n 次 NIC 的算法对样本进行增量聚类,从 NIC 的伪代码可以看出 NIC 中循环处理 k 个数据, k 表示簇心的个数. 因此算法 ANIC 的时间复杂度近似为

$$f(n) = m * (2n - 1) + 2n * m * m_h + n * m * m_h + n * k = O(n).$$

Algorithm 2 ANIC

Input:

k ; 样本 x ; proximity; u ; re; Sum; Num;

Output:

proximity_matrix; u ; re; Sum; Num;

index = size(re, 1); index = index + 1 ; num = size(u , 1);

Step1:/* 用前 k 个到达的样本初始化簇心 */

if index $\leq k$ then

$u(\text{index}) = x$; $r \leftarrow [x \text{ index}]$; Sum(index) = x ; Num(index) = 1;

for $i = 1$ to index do

proximity $\leftarrow$ 通过式(5)计算 x 与 u 中簇心距离;

end

end

end

Step2:/* 为 k 之后的样本选择合适的簇 */

else

for $j = 1$ to num do

temp_d $\leftarrow$ 通过式(5)计算 x 与 u 中簇心的距离;

end

[min_distance c] $\leftarrow$ min(temp_d);

max $\leftarrow$ max(proximity);

[min m n] $\leftarrow$ min(proximity);

if min_distance > max then

```
u(num + 1) = x;
re←[x index]; Sum(index) = x; Num(index) = 1;
for i = 1 to num + 1 do
    proximity←通过式(5)计算 x 与 u 中簇心的距离;
end
end
else
    if min_distance ≤ min then
        Sum(c) = Sum(c) + x; Num(c) = Num(c) + 1;
        u(c)←通过式(6)更新簇心
        for i = 1 to num do
            proximity←通过式(5)计算 x 与 u 中簇心距离;
        end
    end
end
end
else
    合并簇 m 和簇 n;
    Sum(m) = Sum(m) + Sum(n);
    Num(m) = Num(m) + Num(n);
    Sum(n) = x; Num(n) = 1;
    u(m)←通过式(6)更新簇心; u(n) = x;
    for i = 1 to num do
        proximity←通过式(5)计算 n 与 u 中簇心距离;
    end
end
end
end
end
end
```

3 仿真实验及结果

将我们的算法与最通用的两个聚类算法 Kmeans, FCM, 以及文献[7-8]中的增量聚类算法进行比较, 来验证算法的有效性.

3.1 实验环境

本次实验平台为 matlab7. 11. 0. 584 (R2010b), 64 bit(win64), 计算机配置为处理器 AMD Phenom (tm) II X4 945 Processor 3.00 GHz, 内存为6.00 GB, 64 位操作系统.

3.2 数据集

本次实验采用的数据集为 UCI 的四个数据集, 分别为 Iris、Balance-Scale、pima 及 Wine. 数据集详细信息如表 1 所示.

表 1 数据集信息			
数据集名称	样本个数	属性个数	类别个数
Iris	150	4	3
pima	768	8	2
Balance-Scale	675	4	3
Wine	178	13	3

3.3 评价指标

本次实验的评价指标为纯度(purity). purity 的定义为簇包含单个类的对象的程度. 簇 i 的纯度:

$$p_i = \max_j p_{ij} = \max_j m_{ij} / m_i.$$

式中, m_i 是簇 i 中对象个数, m_{ij} 是簇 i 中类 j 的对象个数.

聚类的总纯度: $\text{purity} = \sum_{i=1}^k \frac{m_i}{m} p_i$ (m 为样本总数). 纯度越高说明算法的聚类效果越好.

此外, 通过比较聚类的时间消耗来评价算法的性能.

3.4 实验结果分析

实验中, 对于算法 ANIC 进行两种聚类实验.

3.4.1 实验一

对整个数据集进行聚类, 每个数据集聚类 1 000 次, 求出其平均聚类纯度与时间消耗, 并将其与 Kmeans 和 FCM 的平均聚类纯度和时间消耗进行比较, 来验证算法 ANIC 的聚类效果.

实验结果如表 2 和表 3 所示. 从表 2 可以看出, 对于数据集 Iris、pima 和 Balance-scale, ANIC 的聚类效果优于 Kmeans 和 FCM 的聚类效果, 特别是对于数据集 Balance-scale, ANIC 聚类的平均纯度高于 FCM 和 Kmeans 的平均聚类纯度 20% 左右, 对于数据集 Wine, ANIC 的聚类效果优于 Kmeans 的聚类效果, 而比 FCM 的聚类效果差 3% 左右, 这说明 ANIC 对于不同数据集能够得到比较稳定的聚类效果, ANIC 具有更好的通用性. 从表 3 可以看出, 在对四个数据集进行聚类时, ANIC 的时间消耗整体低于 FCM 的时间消耗, 而与 Kmeans 的时间消耗基本持平, 这是因为 ANIC 算法的时间复杂度为 $O(n)$, 虽然在对自动编码器进行训练时会消耗一定的时间, 自动编码机会对样本属性进行压缩, 并且在聚类过程中只需要维护临近度矩阵, 导致聚类时时间消耗降低, 因此整体时间消耗很低.

表 2 整体聚类结果对比 %

数据集名称	FCM 平均纯度	FCM 最高纯度	k-means 平均纯度	k-means 最高纯度	ANIC 平均纯度	ANIC 最高纯度
Iris	89.333 3	89.333 3	83.692	90.667	95.095	98
pima	66.667	66.667	67.318	73.958	72.349	76.302
Balance-Scale	65.454	85.44	65.556	77.6	85.688	88.64
Wine	94.944	94.944	90.285	96.629	91.303	94.382

表 3 整体聚类平均时间消耗对比

数据集 名称	FCM 平均 时间消耗/s	k-means 平均时间 消耗/s	ANIC 平均 时间消耗/s
Iris	0.051 639	0.030 419	0.030 876
pima	0.247 71	0.112 51	0.097 042
Balance-Scale	0.530 68	0.071 902	0.073 182
Wine	0.058 983	0.036 241	0.032 724

3.4.2 实验二

将数据集划分为多个数据集,进行增量聚类,并将聚类结果与文献[7-8]中给出的结果进行比较,来验证算法 ANIC 对不断增加的数据集进行增量聚类时的聚类效果.数据集进行划分时,Iris 数据集划分成分别包含 100 和 50 个样本的数据集,pima 划分成 2 两个包含 384 个样本的数据集,Balance-scale 数据集划分成 5 个包含 125 个样本的数据集,Wine 数据集划分成两个包含 89 个样本的数据集.实验比较结果如表 4 和表 5 所示.从表 4 可以看出,对于数据集 pima,ANIC 的聚类效果比基于马氏距离的 FCM 增量算法低 14% 左右,其余三个数据集,ANIC 的平均聚类纯度均高于基于马氏聚类的 FCM 增量聚类算法,特别是对于数据集 Balance-scale 和数据集 Wine,ANIC 的聚类效果明显高于基于马氏距离的 FCM 增量算法.从表 5 可以看出,对于数据集 Iris 和 Wine,ANIC 的聚类效果要好于 FCM 增量聚类算法和 IFCM 增量聚类算法.说明 ANIC 能够对不断增加的数据集进行增量聚类,并得到很好的聚类效果.

表 4 与文献[7]的结果比较

数据集 名称	基于马氏距 离的 FCM 增量算法	ANIC 平均 纯度/%	ANIC 最高 纯度/%
Iris	91.2	93.411	97.333
pima	80.7	66.605	72.917
Balance-Scale	69.1	87.567	89.28
Wine	46.5	90.861	94.382

表 5 与文献[8]的结果比较

数据集 名称	FCM 增量 式算法	IFCM 增量 式算法	ANIC 平均 纯度/%	ANIC 最高 纯度/%
Iris	90.67	92.67	93.411	97.333
Wine	85.39	88.20	90.861	94.382

4 结束语

本文提出一种基于 AutoEncoder 的聚类算法 ANIC,可有效学习数据集的主要特征,对数据集样本特征进行压缩,提高聚类纯度,并且能够通过一遍式读取数据和动态更新簇类中心对不断增加的数据

集进行增量式聚类,在聚类过程中只需要维护临近度矩阵,降低时间消耗.实验结果表明 ANIC 无论是进行整体聚类还是进行分批增量聚类,都能得到很好的聚类效果,并且时间消耗低,能够识别离群点.

接下来的工作主要针对以下几个方面进行:如何构建自动编码器使得其能够得到数据集的最佳压缩表示形式;如何将该算法扩展到非数值属性数据集的聚类中.

参考文献:

[1] Song C, Huang Y, Liu F, et al. Deep auto-encoder based clustering[J]. Intelligent Data Analysis, 2014, 18(6):65-76.

[2] Huang P, Huang Y, Wang W, et al. Deep embedding network for clustering [C] // Pattern Recognition (ICPR), 2014 22nd International Conference on. stockholm: IEEE, 2014: 1532-1537.

[3] 段宝彬,韩立新,谢进. 基于堆叠稀疏自编码的模糊 C-均值聚类算法[J]. Computer Engineering and Applications, 2015, 51(4): 154-157.

[4] MacQueen J. Some methods for classification and analysis of multivariate observations[C] //Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Berkeley, California:IEEE,1967: 281-297.

[5] Dunn J C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters[J]. Journal of Cybernetics,1973,3(3):32-57.

[6] Bezdek J C. Pattern recognition with fuzzy objective function algorithms[M]. New York:Plenum Press,1981.

[7] 郑宏亮,王建英. 一种基于 Mahalanobis 距离的增量聚类算法[J]. 计算机应用与软件, 2011, 28(12): 38-40.

[8] 吴佳,罗可. 改进的模糊 C 均值的增量聚类算法[J]. 计算机工程与应用, 2011, 47(23): 141-143.

作者简介:

原旭男,(1970-),硕士,副教授.研究方向为网络安全、云计算安全、动态数据库技术、金融数据挖掘、电子商务.

杨镇楠(通讯作者)女,(1991-),硕士研究生.研究方向为数据分析. E-mail:jenyang0918@gmail.com.

赵亮男,(1988-),博士研究生.研究方向为大数据可用性.

陈志奎男,(1968-),博士,教授.研究方向为物联网、云计算、无线通信与数据库等.