流数据密度估计*)

聂国梁　卢正鼎
(华中科技大学计算机科学与技术学院　武汉 430074)

摘　要　密度估计在许多流数据决策系统中都有重要的意义。本文考虑了最近数据的重要性，利用核心密度估计方法，提出了一种适合流数据特点的密度估计算法。该算法利用远远小于数据长度的内存，通过对流数据进行窗口划分，为单个窗口保留少量的分布信息，再综合这些窗口信息，从而实时评估流数据的密度分布。理论和实验证明，该算法是快速有效的。

关键词　流数据，密度估计，流数据统计，近似算法

Density Estimation over Data Stream

NIE Guo-Liang　LU Zheng-Ding
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Density estimation over data stream plays an important role in many data stream's decision support systems. Considers the importance of the recent data, and adopts kernel method, proposes an algorithm for density estimation over data stream. Using memory as little as possible, this algorithm divides data stream into sub-windows, and store a little information for each sub-window. By integrating information of sub-windows, this algorithm can estimate the density of data stream timely. Theory and experiments prove that the algorithm is accurate and effective for density estimation over data stream.

Keywords　Data stream, Density estimation, Data stream statistics, Approximation algorithms

1　引言

随着互连网络的发展，越来越多的数据呈现出流数据的状态：连续到达，长度未知。流数据的特点决定了其相关的算法必须是单遍访问，且只能利用远远小于数据长度的内存空间，所以传统的算法不再实用。目前，流数据已成为国内外的研究热点，大量研究主要集中在流数据管理系统，流数据热门元素的查找，流数据的挖掘，等等。但是对流数据密度估计的研究却很少，而密度估计在流数据的挖掘和决策系统中，却起着举足轻重的作用，因此，研究流数据的密度估计具有重要的理论和实际意义。

本文对流数据的密度问题进行了研究，提出了一种实时估计算法。

2　相关工作

密度估计方法，大体上分为两种：有参估计和无参估计。有参估计的前提是已知数据密度符合某种分布，问题在于如何确定密度参数。这种密度估计方法的准确性，强烈的依赖于前提假设：符合一种已知分布。一旦假设错误，估计结果的参考价值就会大大降低。现实中，数据的分布往往是无法提前得知的，这就需要使用无参估计方法来估计数据的密度。

国内外的学者对无参估计方法进行了大量研究。

直方图[1,2]作为一种无参密度估计方法，已被广泛应用。但是直方图有其缺点：应用与多维数据时太复杂；数据的范围必须事先知道；得到的密度分布不够平滑。而核心密度估计法恰恰避免了直方图的这些缺点。核心密度估计法定义如下：对包含 N 个数据的集合，其密度分布可表示为：$D=\frac{1}{N}\frac{1}{h}\sum_{i=1}^{N}K(\frac{x-x_i}{h})$。$x_i$ 代表单个数据，h 表示平滑参数，$K(x)$ 是一个任意选定的单峰，对称且有界的密度函数。Wand 等人[3]对核心密度估计法进行了详细的描述，并且对密度估计的平滑过渡提出了崭新的思想。Gisbert 等人[4]引入了加权采样策略，把核心密度估计方法进行了扩展。Katkovnik 等人[5]采用调整窗口尺寸的方法，提出了一种新的核心密度估计法。它对一个窗口生成了左密度函数和右密度函数，通过合并给出整个窗口的密度分布。这种合并极大地提高了密度估计的准确性。Vincent 等人[6]提出了一种局部加权核心密度估计法，并将局部加权的思想应用到不同的方法中：k-最近邻居密度估计[7,8]，固定宽度最近邻居密度估计[9]。Loader 等人[10]针对密度估计中的平滑参数的选择问题，就传统方法和插入方法进行了分析比较，提出了自己的见解。核心密度估计法因为需要保留所有数据，所以不适用于海量数据。为了解决此问题，Zhang 等人[11]充分考虑了核心密度估计法和 CF-树的动态增长[12]特性，提出了 CF-核心密度估计方法。该密度估计方法虽然适用于海量数据，但却忽略了对实时性的要求，也没突出最近元素的重要性。

以上的方法虽然解决了传统数据的密度问题，但这些方法却不能直接应用于流数据的密度估计。流数据密度估计的研究尚处于初级阶段。Zhou 等人[13]基于核心密度估计方

*)国家自然科学基金资助项目(60403027)。聂国梁　博士研究生，主要研究数据库技术和流数据算法；卢正鼎　教授，博士生导师，从事分布异构系统和数据库技术研究。

法，研究了流数据的密度估计。其主要思想就是：通过简单合并相邻核心，在保持较小使用内存的前提下，对流数据的密度进行了估计。这种不断的合并导致其单个元素处理时间过长，这违背了流数据的实时要求，而且此算法也没有突出最近元素的重要性。

本文充分考虑了最近元素的重要性，基于子窗口，采用核心密度估计方法，提出了一种快速算法，解决了流数据的密度估计问题。

3 算法

把流数据划分成若干子窗口，每个窗口的尺寸固定，其具体的值由所提供的内存决定。对每个窗口中的数据进行压缩统计，统计结果保存在一个四元组中。四元组定义：(TIMESTAMP，NUMBER，MEAN，DEVIATION)。

TIMESTAMP 代表子窗口所包含的最新数据的时间戳。时间戳是动态变化的，随着新数据的到达，其相应地增加 1。每个最新到达的数据，其时间戳都设定为 1。实时更新时间戳耗时巨大，也是没必要的。本文时间戳的变化方式采用 Datar 等人[14]提供的时间戳的变化方式。不需要实时更新时间戳，只需在访问窗口对应的四元组的时候，通过变化即可得到窗口的正确时间戳。

NUMBER 表示子窗口所包含的数据个数，其最大值为窗口尺寸加 1，1 是由于过期窗口导致的。

MEAN 代表子窗口所包含数据的均值。实际上也可以用中值来代替。本文是采用均值方式。

DEVIATION 表示 $\frac{1}{\text{NUMBER}}\sum_{D_i\in \text{SubWindow}}(D_i-\text{MEAN})^2$。即表示窗口包含数据的方差。

当一个新数据 E 到达时，处理如下：

(1)COUNT 加 1(初始化为 0)。

(2)如果 COUNT % SWS 为 1(SWS 为子窗口尺寸)，创建一个新的子窗口及其对应四元组(TIMESTAMP，NUMBER，MEAN，DEVIATION)，其初始化为(0,0,0,0)。

(3)把 E 加入到时间戳最近的一个四元组 FA(TIMESTAMP，NUMBER，MEAN，DEVIATION)，使得 FA：TIMESTAMP 为 E 的时间戳，MEAN=(MEAN * NUMBER+E 的值)/(NUMBER+1)，NUMBER 增加 1，DEVIATION=(NUMBER−1) * DEVIATION / NUMBER+(MEAN-E 的值)^2/(NUMBER−1)。

(4)如果 COUNT % SWS 为 1，则对过期子窗口及其四元组进行检测处理：如果元组 FA1(T_1，NUMBER_1，MEAN_1，DEVIATION_1)过期，而 FA2 是未过期且时间戳最大的元组(T_2，NUMBER_2，MEAN_2，DEVIATION_2)，则把 FA1 看作一个单独的点而合并到 FA2，使得 FA2 变成：$\text{MEAN}_2=(\text{MEAN}_2*\text{NUMBER}_2+\text{MEAN}_1)/(\text{NUMBER}_2+1)$，$\text{NUMBER}_2$ 增加 1，$\text{DEVIATION}_2=(\text{NUMBER}_2-1)*\text{DEVIATION}_2/\text{NUMBER}_2+(\text{MEAN}_2-\text{MEAN}_1)^2/(\text{NUMBER}_2-1)+\text{DEVIATION}_1/\text{NUMBER}_2$，时间戳不变。删除 FA1。

当需要输出数据密度分布时，操作过程如下：

遍历所有四元组，统计数据个数：$\text{COUNT}=\sum_i \text{NUMBER}_i$。

对第 I 个子窗口，找到其对应的四元组 FA_i。用正态分布密度 $D_i(x)$来表示第 I 个子窗口的密度。子窗口密度函数 $D_i(x)$的均值为 MEAN_i，而方差为 DEVIATION_i。$\text{DEVIATION}^{1/2}$实际上就是平滑参数。

通过对当前所有子窗口的密度函数进行累加，求得全体数据的密度分布 $AD(x)$：$AD(x)=\frac{1}{\text{COUNT}}\sum_i \text{NUMBER}_i * D_i(x)$。

本文采用正态分布密度来表示子窗口的密度分布，实际上也可以采用别的分布函数，如均匀分布，泊松分布等等。只要子窗口的数量满足一定的条件，具体采用哪种分布来表示子窗口密度，是不会影响总体数据密度分布描述的，Wand 等人[3]已经给出了证明。

4 分析

本算法的空间复杂度和时间复杂度与具体的数据量没有直接的关系，而仅与子窗口的数目有关系。当为流数据维护 m 个子窗口时，则算法的空间复杂度为 $O(m)$，时间复杂度也为 $O(m)$。很明显，单个数据的平均处理时间为 $O(1)$。子窗口的数目 m 影响密度估计的精度，m 越大，结果的精度越高。相对地，m 越小，结果的精度越低。因此，可以根据应用的要求，结合所能提供的内存，选择一个合适的 m 值，来对流数据进行划分。

对任意的一个子窗口，对 DEVIATION 进行维护，使之随时保持：$\text{DEVIATION}=\frac{1}{\text{NUMBER}}\sum_{D_i\in \text{SubWindow}}(D_i-\text{MEAM})^2$。不失一般性，假定当前子窗口的四元组为($T_0$，$N_0$，$\text{MEAN}_0$，$\text{DEVIATION}_0$)，包含的数据集为 S。当一个新的元素 E 到达，其值假定为 e，则窗口对应的四元组变为(T_1，N_1，MEAN_1，DEVIATION_1)。明显可见，N_1 比 N_0 增加 1，$\text{MEAN}_1=(\text{MEAN}_0*N_0+e)/(N_1)$。而 DEVIATION_1 应该等于$\frac{1}{N_1}\sum_{D_i\in S\cup\{E\}}(D_i-\text{MEAN}_1)^2$，即 $\text{DEVIATION}_1=\frac{1}{N_1}\sum_{D_i\in S}(D_i-\text{MEAN}_1)^2+\frac{1}{N_1}(e-\text{MEAN}_1)^2$。

根据 N_1 与 N_0 和 MEAN_1 与 MEAN_0 之间的关系，以及 DEVIATION 的定义，得到 DEVIATION_1 与 DEVIATION_0 之间的关系：$\text{DEVIATION}_1=(\text{NUMBER}-1)*\text{DEVIATION}_0/\text{NUMBER}+(\text{MEAN}-e)^2/(\text{NUMBER}-1)$。所以，本算法时刻保持着 DEVIATION 的定义。

当合并过期窗口四元组(T_0，N_0，MEAN_0，DEVIATION_0)到活跃窗口时，算法是把过期元组看作一个值为 MEAN_0 的元素来处理的。与新元素直接插入不同的是活跃窗口的 DEVIATION 会多增加一个与 DEVIATION_0 有关的增量。这样做的目的是为了突出最近元素对数据分布影响的同时，也不完全忽略过期数据的影响。

5 实验

本文对算法进行了实验模拟。

实验环境：CPU：Pentium 1.6 GHz；内存：256M DDR；操作系统：Windows2000；编程环境：VC；编程语言：C。

本文采用的数据，是模拟数据，分布在 0 到 6000 之间。为了描述直观，对数据的密度分布进行了转变，采用数据出现的次数来描述分布。

图 1 描述了原始数据的分布，数据量为 1000000。从图 1 可以看出，原始数据大体上符合正态分布，虽然不是严格的遵守。两个临近点出现的次数可能变化比较大，这导致了 y 轴

上的线段反复比较严重。而图 2，是利用本算法得到的对相同数据的分布描述。采用 400 个窗口，对 1000000 个数据进行了统计，通过计算，得到了图 2。比较图 1 和图 2，可以清楚地看到：图 2 没有太注重临近点之间出现次数的变化，而从大趋势上体现了数据的分布特点。当然，误差也是有的。很明显，图 2 表示出来的数据出现次数的最大值接近 500，而实际上的最大值在 550 左右。这种误差，可以通过增大子窗口的数量来减少。尽管有误差，本文的算法还是能够很好地描述数据的分布，而且，相比原始的分布，本文算法提供的分布描述更加简单明了。

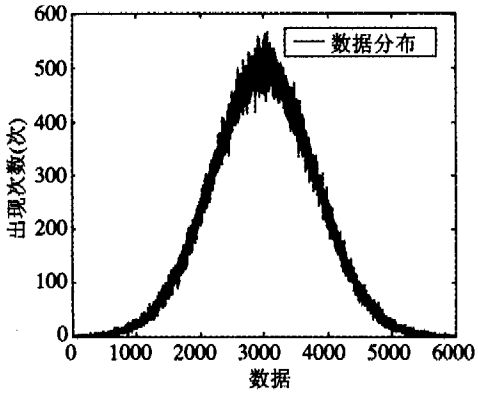

图 1　原始数据分布(1000000)

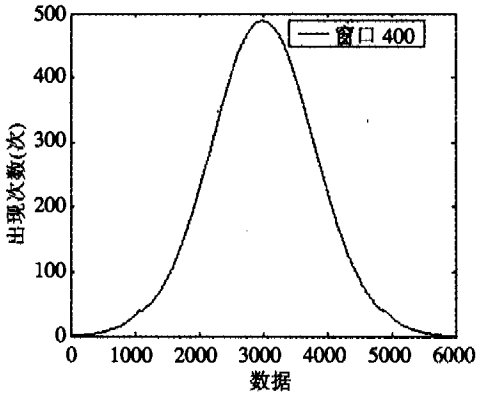

图 2　400 个窗口还原的数据分布(1000000)

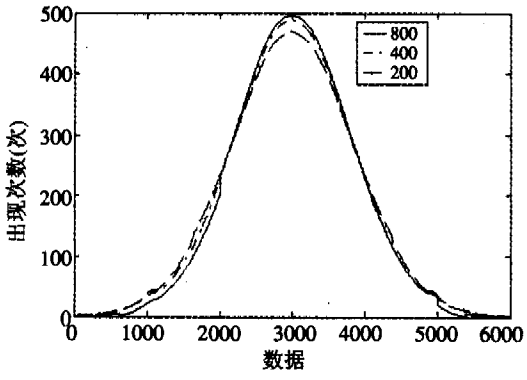

图 3　利用不同子窗口数目得到的分布之间的比较

图 3 描述了不同窗口数目，对密度估计的影响。结合图 1，可以清楚地看到，随着子窗口数目的增加，算法所产生的密度估计更加逼近数据的真实分布。数据的真实分布中，最高出现次数为 550 左右。而本文算法，当采用子窗口数目为 200 时，产生的最大出现次数为 450；当采用子窗口数目为 400 时，最大出现次数为 480；对于子窗口数目为 800 时，最大出现次数变为 500。在数据的两端，3 条曲线中，子窗口数目为 800 的更加能表现真实分布。从图形上来看，随着子窗口数目的增加，曲线的形状也变得越来越复杂一些，也越来越接近数据的真实分布。

总之，本文算法能够描述数据的密度分布，而且，随着采用更多的子窗口，更加能够逼近真实的密度分布。

总结　密度估计在许多流数据决策系统中都有重要的意义，但是传统的密度估计方法由于种种原因不适合流数据。迄今为止，对流数据的密度估计研究很少。本文在考虑了新近数据重要性的前提下，利用核心密度估计法，提出了一种适合流数据特点的密度估计算法。该方法利用远远小于流数据长度的内存，通过对数据进行窗口划分，为单个窗口保留少量的分布信息，从而实时描述流数据的分布。理论和实验证明，该算法是快速有效的。有关流数据的密度估计，还有许多问题需要进一步解决，如，根据数据的特点，如何动态的调整平滑参数；根据数据的离散程度，如何调整子窗口的尺寸等等。流数据密度估计的有关工作，尚处于初级阶段，仍需进一步研究。

参 考 文 献

1　Silvernab B W. Density Estimation for Statistics and Data Analysis. Chapman and Hall, 1986

2　Devroye L. A Course in Density Estimation. Birkhauser Noston, 1987

3　Wand M P, Jones M C. Kernel Smoothing. Chapman and Hall, 1995

4　Gisbert F J G. Weighted Samples, Kernel Density Estimators and Convergence. Empirical Economics, 2003, 28:335～351

5　Katkovnik V, Shmulevich I. Nonparametric Density Estimation with Adaptive Varying Window Size. In: SPIE Conference on Image and Signal Processing for Remote Sensing VI, European Symposium on Remote Sensing, Barcelona, Spain, 2000

6　Vincent P, Bengio Y. Locally Weighted Full Covariance Gaussian Density Estimation:[Technical report]. 2004

7　Fix E, Hodges J L. Discriminatory analysis, non-parametric discrimination, consistency properties:[Technical report]. USAF School of Aviation Medicine, Texas, 1951

8　Loftsgaarden D O, Quesenberry C P. A nonparametric estimate of amultivariate density function. Annals of Mathematical Statistics, 1965, 36:1049～1051

9　Parzen E. On the estimation of a probability density function and mode. Annals of Mathematical Statistics, 1962, 33:1064～1076

10　Loader C R. Bandwidth Selection: Classical OR Plug-in?. The Annals of Statistics, 1999, 27(2):415～438

11　Zhang T, Ramakrishnan R, Livny M. Fast density estimation using cf-kernel for very large databases. In: Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), San Diego, CA, USA, 1999

12　Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large database. In: Proceedings of ACM SIGMOD Int'l Conf. on Management of Data, Montreal, Canada, 1996

13　Zhou Aoying, Cai Zhiyuan, Wei Li, et al. M-Kernel Merging: Towards Density Estimation over Data Streams. In: Proceedings of the 8th International Conference on Database Systems for Advanced Applications, Japan, 2003

14　Datar M, Gionis A, Indyk P, et al. Maintaining Stream Statistics over Sliding Windows. In: Proceedings of the Thirteenth Annual Acm-siam Symposium on Discrete Algorithms, San Francisco, 2002