**联合二部网络表示学习的矩阵分解推荐算法**

张以文1，袁梦祥1

（1.安徽大学计算机科学与技术学院，安徽 合肥 230601）

**摘要：**在煤矿安全生产过程中，及时准确地检测瓦斯异常对煤矿开采的稳定性和安全性有着举足轻重的影响。为解决上述问题，本文提出一种基于局部敏感哈希孤立森林的滑动窗口异常检测算法。该方法首先基于煤矿的多维数据，在每个滑动窗口内建立局部敏感哈希孤立森林并自动更新森林。在真实的淮南煤矿数据集上进行了大量的实验，结果表明，该方法有效地提高了检测精度。

**关键词：**瓦斯浓度；异常检测；滑动窗口；局部敏感哈希；孤立森林

**中图分类号: 文献标识码：**

**Coal Gas Anomaly Detection based on Locality-Sensitive Hashing iForest with Sliding Windows**

ZHANG Yi-Wen1 ，YUAN Meng-Xiang1

（1.School of Computer Science and Technology, Anhui University, Hefei 230601）

**Abstract: In the safe production process of coal mines, timely and accurate detection of gas anomalies has a significant impact on the stability and safety of coal mining. In order to solve the above problems, this paper proposes a sliding window anomaly detection algorithm based on Locally Sensitive Hashing Isolation Forest. The method firstly builds a locally sensitive hashed Isolation Forest and automatically updates the forest within each sliding window based on the multidimensional data of the coal mine. A lot of experiments were carried out on the real Huainan coal mine dataset. The results show that the method effectively improves the detection accuracy.**

**Key words:** Gas Concentration; Anomaly Detection; Sliding Windows; Locality-Sensitive Hashing; Isolation Forest

# 1引言.

随着大数据时代的到来，“万物互联”的理念使得物联网得到迅猛发展和广泛运用，也为实现矿山的全面感知提供了成熟的技术支撑。煤炭行业各级领导和工作人员的信息化意识普遍增强，煤矿的机械化、自动化、信息化程度不断提高，煤矿监控、通信与监视系统的大量运用，我国煤矿事故起数、死亡人数、百万吨死亡率均大幅下降，煤矿安全生产形势明显好转。煤矿安全生产的迫切需求，促进了物联网技术在煤矿的运用，智能传感器、数字传输系统与信息系统的使用为煤矿企业积累了海量数据。企业并未对煤矿生产过程中产生的海量数据进行充分挖掘和利用，数据信息独立保存或者物理集中，对数据的规律研究几乎为零，大量数据的价值未得以体现。

异常检测是一项重要而强大的数据挖掘技术，旨在自动识别数据集中的异常模式，并且通过分析异常模式的特性来解决特定应用领域的异常事件[1]。异常检测的研究正广泛应用于各个领域，包括在煤矿领域，矿井瓦斯异常检测系统是煤矿安全生产的重要保障，由于矿井下地质条件复杂，设备众多，环境恶劣，瓦斯传感器探头失效和不按规定放置等因素，仅靠煤矿安全监控系统的瓦斯传感器测量到超限报警，结果并不准确。针对这种情况，本文利用智慧矿山所产生的大数据进行多维、多方位的挖掘分析，采用多参数分析对CH4、CO、风速等数据进行综合处理和优化，找出数据的内在联系和规律，提出基于局部敏感哈希孤立森林与滑动窗口的煤矿瓦斯异常检测算法（Locality-Sensitive Hahsing iForest with Sliding windows Algorithm），简称SWLSHiForest，有效提高了煤矿事故风险防控能力。

本文的主要贡献如下：

（1）本文提出基于滑动窗口的局部敏感哈希孤立森林异常检测算法，利用局部敏感哈希在高维数据处理上的优势，建立了适应高维场景的异常检测方法。

（2）本文通过在每个滑动窗口建立相应的局部敏感哈希孤立森林，并根据异常率自动更新异常检测模型，对不同的异常模式有很高的检测精度。（突出自动特点）

（3）本文使用真实的煤矿安全监测数据集对所提出的方法进行了实验评估，结果表明，本文所提出的方法能够很好地处理瓦斯浓度异常检测问题。

# 2 相关工作

目前，已经有很多异常检测的方法，它们大致可分为基于统计的方法[2]，基于领域的方法[3]，基于分类的方法[4]，基于隔离的方法[5]，这些方法在特定的领域都有其优势，但是，在流数据中，这些方法都有一些缺点而不能直接运用于流数据异常检测。

基于统计的方法通常假定正常的数据对象产生于某一个统计模型而不属于该分布规律的数据对象为异常点[6] 。该方法拥有成熟的概率统计知识作为支撑，因此检测出异常数据可以有很好的解释。但它高度依赖于数据模型分布的假定，即要求已知数据服从某种分布，而实际情况中数据集很难服从该假定。其次，此方法检测的数据对象是单一维度的，并不合适用在高维度数据。

基于邻域的异常点检测方法主要通过比较每个数据对象其邻域来判断数据是否异常。LOF 算法[3]就是一种典型的基于邻域的检测算法，其主要的思想是通过比较每个点和第k 邻域局部密度来判断该点是否为异常点。由于LOF 算法对参数k 比较敏感，而不合理的k 值会导致较差的检测效果，为此，Pokrajac等人[7] 提出了基于连接性的异常因子算法(Connectivity based Outlier Factor, COF)。该算法根据最短路径和数据对象的连接性来确定邻域k，计算与其邻域的平均连接距离，并以此作为相对密度来判断异常点。由于COF 算法计算量大，因此在处理大规模数据集时效率较低。然而此算法计算高维数据的离散度，会存在部分数据维度信息没有使用，将会导致算法可靠性下降。基于局部距离的异常因子算法(Local Distance-based Outlier Factor, LDOF) [8] 将数据对象到k个近邻的距离的均值与k个近邻彼此之间的距离均值的比值作为该数据对象的异常度。注意到，此算法在大规模规模高维数据集下运行速度较慢。

基于分类的异常点检测方法主要通过学习数据对象的边界，将边界外的数据点作为异常点。由于数据标签种类不同，分类的形式有单分类和多分类，所以基于分类的异常检测方法分为单分类的异常点检测和多分类的异常点检测。单分类的异常点检测是学习数据集的一个边界，边界内包裹的数据属于正常点，边界之外的数据则是异常点。代表性的算法如一类支持向量机算法(One Class Support Vector Machine, One-class-SVM) [9] ，该算法在高维特征空间中通过非线性核映射计算一个最小超球体作为边界，将边界内的数据作为正常点，而边界外的数据作为异常点。通常这类问题要求已知的数据集大多数属于同一类，而另一类数据集的样本数目是很少，此方法效率会较慢。多分类的异常检测方法主要对数据集学习多个边界，将不包含在任何边界内的数据点定义为异常点。最具有代表性的是基于神经网络的多分类异常点检测[15]，此方法分为两个阶段：第一个阶段利用正常的多分类训练数据来训练模型；第二个阶段将测试数据输入模型，若网络接收则为正常点，反之为异常点。

# 3 背景知识

## 3.1 局部敏感哈希

局部敏感哈希（Locality Sensitive Hashing, LSH）[10]是一种从海量的高维数据集合中找到与某个数据最相似的一个数据或者多个数据的高效方法。其基本思想是，在高维数据空间中的两个相邻数据被映射到低维数据空间中后，将会有很大的概率保持相邻；原本不相邻的两个数据，在低位空间中也将有很大的概率不相邻。其定义如下：

对于原始数据空间中的任意两个点，表示和的距离，表示的哈希值，如果对哈希函数族中的任意一个哈希函数同时满足条件(1)和(2)，则称该哈希函数族为。





事件表示和会被映射到同一个桶中，为了保证局部敏感哈希函数有效，需满足。从哈希函数族中随机选取多个（）哈希函数，生成连接的组合哈希值，并且根据这些哈希值决定两个数据实例是否被哈希到同一个桶中。这种策略能减少负样本被模型预测为正样本的事件，即误报率（False Positive）降低，从而提高了相似搜索的精度。同时，建立多个（）哈希表减少了正样本被模型预测为负样本的事件，即漏报率（False Negative）。设为两个数据实例间的相似度，两个数据实例称为候选相似对的概率是 。LSH的另一个显著特征是其进行相似计算时的线性时间复杂度，而传统的往往是平方级复杂度。因此，LSH已被广泛的运用于各种运用，如数据库中的快速近似最近邻搜索和数据挖掘[11]。

由于相似搜索质量受影响，需要调整参数以求获得良好的性能。其后，一种名为LSH forest 的数据结构通过使用可变长度的组合密钥来减轻参数调整的任务量[12]。其核心思想是每个数据实例的组合密钥足够长，以确保每个实例具有不同的密钥。具体地，在所有组合密钥的集合上构造被称为LSH树的（逻辑）前缀树，其中每个叶节点对应于数据实例，使用不同的LSH函数对各个节点产生路径标签并进行分割。从根到叶节点的路径标签组成对应数据实例的组合键。 LSH树的森林通常被构造用于鲁棒近似最近邻搜索。

## 3.2 孤立森林

孤立森林（Isolation Forest, iForest）是一种基于集成方法的快速异常检测方法，具有线性时间复杂度和高精准度的特点[5]。iForest适用于连续数据的异常检测，将异常定义为“容易被孤立的离群点”，即分布稀疏且离密度高的群体较远的点。在数据空间里面，分布稀疏的区域表示数据发生在此区域的概率很低，因此可认为落在这些区域里的数据是异常的。

iForest属于无参数、无监督式方法，无需定义数学模型也不需要标记，iForest有大量称为isolation tree，简称iTree。iTree不同于决策树，其学习过程非常随机：随机抽取特征、随机选取分割值来建立决策树，从而将每一个样本分到一个独立的子节点上。从超空间的角度看，不断地用随机选取的超平面切分样本点，直到所有的样本点都被这些超平面“孤立”起来，即与其他样本点分隔开。

使用了较少超平面就可以孤立起来的点，即特别容易被孤立出来的点。一般距离密度较高的地方比较远，所处位置的样本点密度较小，因此更易被孤立。因此，我们把非常容易被孤立的那些点判定为异常值。

iForest具有线性时间复杂度，由于采用了基于集成的方法，所以可以用在海量数据的数据集上面。通常树的数量越多，算法越稳定。iForest不适用于特别高维的数据，由于每次分割数据空间都是随机选取维度，iTree建立完成后，仍然有大量的维度信息没有被使用，导致算法可靠性降低，高维空间还可能存在大量噪音维度或无关维度，影响数的构建[13]。

# 4 SWLSHiForest检测方法

从孤立的角度来看，LSH树本质上也可以视为孤立树，因为每个数据实例都与其余数据实例分隔开来。其因此，LSH森林可以用于基于集成方法的孤立机制的异常分析。异常检测包含两个阶段：训练阶段为从每个滑动窗口中以不放回采样的方法建立局部敏感哈希森林；评估阶段为利用已建立的局部敏感哈希森林，计算数据实例的异常得分及异常率，若异常率高于给定阈值，则在下个滑动窗口更新已建立的检测模型。



图2 方法总体描述

## 4.1 采样

在训练阶段，通过给定局部敏感哈希函数族，递归使用哈希函数族，建立LSH树，直到数据空间中的所有数据实例被分隔。使用多种不同的采样方法挑选出输入数据的数据空间[14]，数据采样率遵循均匀分布，基于该下采样方法生成的树高不能遵循均匀分布。因此，本文提出一种新的下采样技术，可以产生均匀分布的树高。控制着训练数据的大小，当增长到某个特定数值时，LSHiForest达到较高的可靠性，如果 继续增加，在检测精度上不会获得增长，只会增加处理时间和内存开销。在实际情况中，异常通常是少数而且不同，正常点是多数且很类似[5]。因此，较小的采样大小足以使LSHiForest区分异常和正常。

我们的采样率为，*s*遵循均匀分布。特别是如果，的变量范围为。我们的采样方法产生的树高很均衡，所以能产生更多的多样性。可以证明，如果粗略地假设树高近似为，我们提出的方法所建立前一棵树的方差为0.94375的前提下，加入一棵新树后，方差为1.3333。另一个优点是我们的方法能够降低计算成本，可以证明我们提出的方法中样本大小的期望值大约是346。

## 4.2 构建局部敏感哈希森林

在获得数据样本后，我们需要计算LSH tree的树高。如果两个数据实例十分接近，这会导致LSH tree增长得非常大，因为LSH函数为两个相同的数据实例生成了相同的哈希值，通常需要更多的哈希函数产生更多的是哈希值，以便区分这些非常接近的实例，但这样会生成长单分支路径。从计算成本的角度看，限制LSH tree的高度是必要的。高度限制对于异常检测的效果几乎没有影响，具有更大深度的数据实例通常是正常值而非异常。此外，压缩单分支路径，使得具有这种长路径的异常示例也可以被检测算法识别。但树高必须足够长，以确保数据能被充分隔离。

|  |
| --- |
| **Algorithm 1:** Constructing LSH Forest |
| **Input:** Data set *X*; LSH family ; number of LSH tree t.  **Output:** A forest of LSH trees . |
| **Begin**  1: **for** *i,*1 **do:**  2:;  3:Compute a height limit:;  4: ;  5: return.  **End** |

由于LSH tree可以被视为字典树[12, 15]，我们可以使用字典树的平均高度来计算树高。让我们考虑使用n个随机的数据实例来试验一下，假设有字母。为了构建数据实例的索引，选择具有概率的元素，当欧拉常量， 字典树的平均树高大约是。假设所有的数据实例*X*和由LSH函数产生的哈希值是独立并且等概率的，例如，，那么，的合适上限为



|  |
| --- |
| **Algorithm 2:** Constructing LSH Tree: LSHiTree(*S,F,H,I*) |
| **Input:** Input data *S*; LSH family ; height limit H; index *I*.  **Output:** A LSH tree T. |
| **Begin**  1: **if  then**  2:return **NULL**;  3: **else if  OR** **then**  4: return ;  5: **else**  6: ;  7: **While** AND  **do:**  8: ;  9: ;  10: **if  then**  11: return ;  12: Initialise child node indexing:;  13: for  **do:**  14: LSHiTree(*S,F,H,I+1)*;  15: ;  16: return .  **End** |

LSH tree的建立是一个递归过程，子程序位于算法1的第3行，详见算法2。在隔离的过程中，来自LSH函数族的哈希函数 用于为所有的数据实例产生哈希值。输入的数据被划分为与哈希值相关联的非重叠子集，即当且仅当，有和。为了生成没有单分支路径的紧凑树，我们重复隔离输入数据，直到生成多个哈希值或达到树高限制。随着哈希函数索引*I*的增加，它记录了未压缩的字典树中数据实例的路径长度信息，从而保留用于异常预测阶段。

## 4.3异常评估

在训练阶段完成了局部敏感哈希森林的建立后，我们需要在每个滑动窗口评估一个数据集中实例的异常得分，如算法3所示。对于一个数据实例，我们通过子程序path\_length评估其路径长度 与LSH tree  的关系，该部分将在算法4中详细说明。为了使来自每个LSH tree的路径长度有可比性，我们基于参考路径长度对进行归一化。不同于iForest对所有的树都使用单一的参考值，在我们的方法中，因为树的大小不同，所以不同的树使用不同的参考值。iForest中使用的参考路径长度是二叉搜索树中不成功搜索的平均路径长度。 然而，这在我们的情况下不适用，因为我们方法中分支因子*v*通常大于2。PARTICIA tries索引个数据实例的成功搜索期望是为，当，。在与高度估计相同的假设下，即，我们的参考值为



由于*v*的信息是未知的，我们需要从训练完成的LSH trees中评估它。具体来说，我们从平均分支因子来评估*v*。

与iForest相同，我们进一步用指数函数把将归一化路径长度非线性地缩放到区间。但是iForest的非线性缩放位于整合方法的路径归一化之后，不会影响最终异常分数的顺序。我们的方法在组合之前进行非线性缩放，使其有助于改善整体的多样性。我们使用非线性标度的算术平均值作为最终的异常得分，即。实际上，来自iForest的最终异常分数可以被视为非线性缩放值的几何平均值，即，是与参考路径长度。尽管非线性缩放 Charu [12]在异常评分组合方法中提出一个名为AOM（Average of Maximum）的方法是一个均衡的选择，性能非常好。我们的方法是AOM的变体，没有对基础检测器进行分组。

|  |
| --- |
| **Algorithm 3:** Predicating Anomaly Scores |
| **Input:** Test data *X*; LSH family ;LSH forest ;  depth adjustment factor ; granularity level .  **Output:** Anomaly scores . |
| **Begin**  1: **for**  **do:**  2:;  3: **for**  **do:**  4: ;  5: Normalisation: ;  6: ;  7: return .  **End** |

算法4详细介绍了子程序，计算实例从根节点到相应叶节点的路径长度。当，我们可以得到关于整棵树的，深度级别，是用户定义的用于限制遍历深度的参数，与iForest中的高度限制相同。此外，我们还使用来调整路径长度，因为叶节点或者深度等于 的内部节点可能对应有多个数据实例，通常来说，同一深度下较大子树中的数据实例比小子树中的更加正常。

|  |
| --- |
| **Algorithm 4:** Subroutine: path\_length() |
| **Input:** Data instance *x*; LSH family ; current node ;  depth adjustment factor ;granularity level ;index .  **Output:** The depth of  in the subtree with  as the root. |
| **Begin**  1: **if  then**  2:return-1;  3: **else if  OR  then**  4: return ;  5: **else**  6: ;  7: **if**  AND  **then**  8: return path\_length;  9: **else**  10: return .  **End** |

我们的方法与iForest有如下不同，在路径长度计算中，遍历可以在内部和外部节点处停止，如算法4的第4行和第10行所示，而iForest只能在外部节点处停止。这意味着LSHiForest具有更强的检测能力，因为不相近的数据实例，即潜在的异常往往具有更短的路径长度。另一个区别是来自字典树和PATRICIA前缀树中的的路径长度被组合成单个路径长度，使我们的方法更加灵活。具体地，，其中和分别是压缩和未压缩的路径长度，是在[0,1]之间变化的调节因子，该参数控制着LSH函数族定义的空间距离信息的粒度。当时，，意味着使用最精细的隔离粒度，并且检测器具有最佳的全局异常性能。然而，对于局部异常，应当是减小这种粒度，因为稀疏区域中点之间的距离可以大于异常和密集正常区域之间的距离。因此，较小的可用于局部异常检测。请注意，这仍然不足以解决无法检测局部异常的问题，一种有前景的解决方案是将数据转换到另一个空间上，并在变换空间中应用基于隔离的异常检测。

## 4.4 时间复杂度分析

关于样本大小，LSHiForest的计算复杂度类似于iForest，具体来说，训练阶段的平均情况时间复杂度为，评估阶段的平均情况时间复杂度是。由于大多数LSH函数具有时间复杂度，因此我们的方法可以非常快。此外，它通过处理定义LSH族的各种距离度量，数据空间和类型，并且能够检测不同类型的异常，具有较高的通用性。

# 5实验分析

## 5.1 数据集

为了验证基于局部敏感哈希森林的煤矿瓦斯异常检测模型的可行性，以及本文算法LSHiForest的有效性，我们使用真实数据集进行测试，该数据集由淮南煤矿集团朱集东矿提供的矿山监测数据，时间区间为2017年3月10日到2017年4月9日，所有传感器每间隔10秒收集一次数据。监测值包括同一工作面下的瓦斯浓度、风速、一氧化碳浓度、二氧化碳浓度、氧气浓度、温度、日产量、煤尘浓度。

实验环境为：PyCharm,编程语言为python，实验机器配置为16G内存，core i7-49703处理器，windows操作系统。

## 5.2 评估指标

为评估本文方法的检测性能，采用AUC（Area Under Curve）[16]作为评测指标，AUC是一个二分类模型评价指标，是ROC（Receiver Operating Characteristic）曲线下面区域的面积，量化ROC曲线性能。ROC曲线基于样本的真实类别和预测概率，其纵轴为真正率（True positive rate，TPR），即预测为正且实际为正的样本占所有正例样本的比例。横轴为假正率（False positive rate，FPR）,即预测为正但实际为负的样本占所有负例样本的比例。其公式如下：





AUC的值越大，说明分类模型的性能越好，对于最理想的分类模型，它的AUC值等于1；对于随机分分类模型，它的AUC值为0.5。

## 5.3 性能对比

为更好地评估本文方法LSHiForest的性能，选择以下几种较经典的方法在AUC指标上进行对比试验。

LOF（局部异常因子）[3]：局部异常因子LOF算法，一种基于距离的异常检测算法，通过待检测点及其领域点的密度来判断该点是否为异常点。

ISO（孤立森林）[5]：孤立森林算法，一种基于Ensemble的异常检测算法，通过递归地随机分割数据集，直到所有样本点都是孤立的，通过路径长短判断待检测点是否为异常点。

为了使实验部分更加完整，我们列出了本文方法及对比方法的重要参数和控制变量，如表1所示。

**表1 算法重要参数**

|  |  |
| --- | --- |
| 方法 | 参数 |
| LOF | 邻居数量n\_neighbors=20 |
| ISO | 采样大小subsample size=256;孤立树高tree height=8;  孤立树个数number of trees=100 |
| LSHiForest | 孤立树个数number of trees=100;采样大小subsample size=100 |

表2是采用0.1%至3%的异常样本比例在不同方法的检测效果。我们观察本文提出框架下的三种方法——ALSH、L1SH、L2SH，与其他较经典的异常检测方法相比，无论在何种异常发生率下均具有更大的AUC，特别是L1SH、L2SH与其他方法相比，在多数情况下都有更大的AUC，这表明本文方法具有更优的检测效果。ISO和LSHiForest框架下的ALSH、L1SH、L2SH在真实的煤矿瓦斯数据集下都有相对稳定的性能，并且能够避免最坏情况的出现，这是因为此类方法与数据分布无关。在现实情况中，如果出现瓦斯传感器损坏，瓦斯监测系统会维持某个恒定值，导致漏报，此刻的AUC值为0，而我们的方法能够综合利用与瓦斯具有关联关系的数据，诸如CO、风量，漏报率与传统办法相比，显著降低。因为综合使用了多维数据，我们可以发现我们框架下的方法在高维数据情形下表现很好，这是因为LSH最初的提出是为了解决高维数据下的最近邻搜索问题。总体而言，我们框架下的方法在高维数据空间、不同异常比例条件下，都有着优异的异常检测效果，特别是L2SH，同时拥有高检测精度和和鲁棒性。

**表2 检测性能对比**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指标 | 方法 | 0.1% | 0.2% | 0.3% | 1% | 2% | 3% |
| AUC | LOF | 0.9504 | 0.9509 | 0.9012 | 0.8939 | 0.8979 | 0.9021 |
| ISO | 0.9830 | 0.9798 | 0.9404 | 0.9482 | 0.9864 | 0.9905 |
| ALSH | 0.9902 | 0.9491 | 0.9853 | 0.9669 | 0.9781 | 0.9801 |
| L1SH | **0.9977** | 0.9851 | 0.9804 | 0.9936 | 0.9955 | **0.9914** |
| L2SH | 0.9972 | **0.9881** | **0.9893** | **0.9942** | **0.9936** | 0.9904 |

## 5.4 LSH tree个数对检测精度的影响

本节均采用1%的异常比例来评估LSH tree的个数*t*对本文框架下方法检测精度的影响。*t*决定了LSHiForest中森林的大小，*t*越大，森林越大，设置参数*t*值分别为50，100，150，200，250，300。如图3所示，本文框架下的三种方法在开始时，随着*t*值的增加，AUC的值随之增加，但是当*t*超过某个阈值时，AUC的值随之下降。这说明*t*取适当的值有利于提高检测精度。因为当*t*取值过小时，LSH tree的个数较小，数据的实际利用价值低，从而降低了检测精度。而当*t*取值为100时，每个数据的路径长度已经覆盖较好，检测效果最佳，并且L2SH方法具有最高的检测精度。



图3 iTree棵树*t*对检测效果的影响

## 5.4 采样大小对检测精度的影响

为了评估采样大小对本文框架下方法检测精度的影响，参数决定了建立每棵LSH tree所需样本的大小，我们设置异常样本比例为1%，*t*=100，并将采样大小从100变化到350，步长设为50。从图4可以看出，三种方法在开始时，随着采样大小的增加，AUC的值也在随之增加，当采样精度超过某个数值时，AUC的值会下降。这说明，适当大小的采样有利于提高检测精度。采样是为了更好地将正常数据和异常数据分离开来，采样数据越多，检测效果越好，对于本文方法而言，如果采样过大，建立LSH tree后仍然有大量的信息没有被使用，导致算法的可靠性降低，还能存在大量噪音数据，影响LSH tree的构建。图4显示，对于不同方法，在选取相应的取值时，AUC能达到其最大值，检测效果最佳。



图4 采样大小对检测精度的影响

# 6 总结

本文提出一种基于局部敏感哈希孤立森林的煤矿瓦斯异常检测方法。该方法在对包括瓦斯在内的多维数据采样后；利用局部敏感哈希建立局部敏感哈希孤立树，获得若干个LSH tree后，组建了LSHiForest；对于每个数据，遍历每一课LSH tree，通过其平均路径长度来计算异常得分。本文在真实的煤矿数据集上进行一系列实验，实验室结果表明本文方法的瓦斯检测精度与以往方法相比有显著提高，不仅解决了瓦斯异常检测问题，同时也解决了因瓦斯探头失效以造成异常漏报或误报问题。

# 参考文献

[1] CHANDOLA V, BANERJEE A, KUMAR V. Anomaly detection: A survey [J]. ACM computing surveys (CSUR), 2009, 41(3): 15.

[2] ANGIULLI F, FASSETTI F. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets [J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009, 3(1): 4.

[3] BREUNIG M M, KRIEGEL H-P, NG R T, et al. LOF: identifying density-based local outliers; proceedings of the ACM sigmod record, F, 2000 [C]. ACM.

[4] HE Z, XU X, DENG S. Discovering cluster-based local outliers [J]. Pattern Recognition Letters, 2003, 24(9-10): 1641-50.

[5] LIU F T, TING K M, ZHOU Z-H. Isolation-based anomaly detection [J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2012, 6(1): 3.

[6] BAUDER R A, KHOSHGOFTAAR T M. A probabilistic programming approach for outlier detection in healthcare claims; proceedings of the 2016 15th IEEE international conference on machine learning and applications (ICMLA), F, 2016 [C]. IEEE.

[7] POKRAJAC D, RELJIN N, PEJCIC N, et al. Incremental Connectivity-Based Outlier Factor Algorithm; proceedings of the BCS Int Acad Conf, F, 2008 [C].

[8] ZHANG K, HUTTER M, JIN H. A new local distance-based outlier detection approach for scattered real-world data; proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, F, 2009 [C]. Springer.

[9] SCHöLKOPF B, WILLIAMSON R C, SMOLA A J, et al. Support vector method for novelty detection; proceedings of the Advances in neural information processing systems, F, 2000 [C].

[10] GIONIS A, INDYK P, MOTWANI R. Similarity search in high dimensions via hashing; proceedings of the Vldb, F, 1999 [C].

[11] WANG J, SHEN H T, SONG J, et al. Hashing for similarity search: A survey [J]. arXiv preprint arXiv:14082927, 2014,

[12] BAWA M, CONDIE T, GANESAN P. LSH forest: self-tuning indexes for similarity search; proceedings of the Proceedings of the 14th international conference on World Wide Web, F, 2005 [C]. ACM.

[13] LIU F T, TING K M, ZHOU Z-H. On detecting clustered anomalies using SCiForest; proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, F, 2010 [C]. Springer.

[14] AGGARWAL C C, SATHE S. Theoretical foundations and algorithms for outlier ensembles [J]. ACM SIGKDD Explorations Newsletter, 2015, 17(1): 24-47.

[15] KIRSCHENHOFER P, PRODINGER H, SZPANKOWSKI W. On the variance of the external path length in a symmetric digital trie [J]. Discrete Applied Mathematics, 1989, 25(1-2): 129-43.

[16] KOMORI O, EGUCHI S. A boosting method for maximizing the partial area under the ROC curve [J]. BMC bioinformatics, 2010, 11(1): 314.