清 华 大 学

综 合 论 文 训 练

题目：针对罗吉斯回归模型优化算法并行加速的研究

系 别：计算机科学与技术系

专 业：计算机科学与技术

姓 名：彭昊若

指导教师：赵颖 教授

2013 年 6 月 8 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签 名： 导师签名： 日 期：

中文摘要

罗吉斯回归模型是一个被广泛应用的有监督学习模型。本文着眼于大数据背景下，解决针对罗吉斯回归模型优化算法的计算效率问题。我们的研究主要是在算法层面。我们研究了三种不同的并行计算系统以及三种不同的计算算法来有效地提升算法的可扩展性。Hadoop，Mahout以及Spark是当前的三个为人所熟知的系统。它们都能支持大规模机器学习算法在大数据及上的运算。并行梯度下降和随机梯度下降是两个当前非常经典的优化算法，并且可以专门用于解决罗吉斯回归模型的学习问题。另外，我们还开发出一种新的并行次线性算法。该并行算法是建立在线性次线性算法基础上的。我们会比较这些算法在不同并行系统上的运行结果。实验结果显示，在不同的数据情况以及不同的系统资源下，我们可以选择有针对性的算法更为高效地解决罗吉斯回归模型的学习问题。同时，更进一步的研究表明，对于此种需要较长时间执行的分布式并行计算而言，容错能力既可以体现在系统层面，也可以体现在算法层面，我们可以根据解决问题的需要进行平衡。

关键词：罗吉斯回归模型；大数据；并行计算；次线性方法

ABSTRACT

Penalized logistic regression (PLR) is a widely used supervised learning model. This paper addresses the issue of computational efficiency for solving PLR in big data scenario. We focus on algorithm level and employ three different parallel computation systems with three different types of parallel algorithms to efficiently improve scalability. Hadoop, Mahout and Spark are three existing well-known parallel systems that can be implemented with large scale machine learning algorithms. Parallel gradient Descent and stochastic gradient descent are two existing state-of-art algorithms for parameter optimization in PLR. We also present a novel parallel sublinear method based on its sequential version. We then make a comparison between these algorithms implemented on different systems. As the outcome shows, we can select different algorithms for the purpose of efficiency in different situations of datasets and machine resources. Moreover, fault tolerance can be provided for those lengthy distributed computations both on algorithm level and system level.

Keywords：Penalized Logistic Regression Model; Big Data; Parallel Computing; Sublinear Method

目 录

第1章 引 言 1

1.1 研究背景 1

1.2 论文 1

1.3 论文结构 3

第2章 已有工作 4

2.1 黑洞攻击与防御 4

2.1.1 黑洞攻击原理 4

2.1.2 黑洞攻击的防御机制 4

2.1.3 黑洞攻击的防御算法 4

2.2 数据污染攻击与防御 6

2.2.1 数据污染攻击 6

2.2.2 数据污染攻击的防御机制 6

2.2.3 数据污染攻击的防御算法 6

第3章 WSN TMOTE平台 8

3.1 WSN TMOTE平台简介 8

3.1.1 硬件环境 8

3.1.2 编译环境 8

3.2 WSN TMOTE平台搭建 9

3.2.1 基础路由协议 9

3.2.2 网络监控及调试方法 11

3.2.3 WSN TMOTE平台实现功能 12

第4章 黑洞攻击与防御 13

4.1 黑洞攻击的实现 13

4.1.1 基于WSN TMOTE平台的攻击实现 13

4.1.2 进行黑洞攻击后的效果 13

4.2 黑洞攻击的防御算法分析 14

4.3 算法实现 15

4.3.1 算法简化 15

4.3.2 基于WSN TMOTE平台的防御实现 15

4.4 实测防御效果及分析 16

4.5 黑洞攻击与防御小结 18

第5章 数据污染攻击与防御 19

5.1 数据污染攻击的实现 19

5.1.1 基于WSN TMOTE平台上的攻击实现 19

5.1.2 进行数据污染攻击后的效果 19

5.2 数据污染攻击的防御算法分析 19

5.3 算法实现 20

5.3.1 算法简化 20

5.3.2 基于WSN TMOTE平台的防御实现 20

5.4 实测防御效果及分析 21

5.5 数据污染攻击与防御小结 23

第6章 总结 24

6.1 论文主要工作的总结 24

6.1.1 工作成果 24

6.1.2 工作难点 24

6.2 进一步的研究工作 25

插图索引 26

表格索引 27

参考文献 28

致 谢 1

声明

附录A

附录B

# 引 言

## 研究背景

罗吉斯回归模型（Logistic Regression Model）[14]在机器学习和数据挖掘领域的作用越来越大。这个模型主要针对分类问题，并且已经有大量的理论和算法基础。罗吉斯回归模型相对于另一种十分流行的支撑向量模型（Support Vector Machines）[25]，同样是很有竞争力的。其优势是，罗吉斯回归模型不仅往往具有很高的拟合精度，而且还有很强的可解释性。这一点体现在罗吉斯回归模型可以直接估计一个条件类的概率。并且，此优势使得一个可以被罗吉斯回归模型所解决的二分类问题同样可以很接单的扩展为一个可以被罗吉斯回归模型所解决的多分类问题。在本文中，我们将集中于二分类问题。

在机器学习领域中，解决分类问题的模型有很多。我们之所以选择罗吉斯回归模型，是因为它是一个在数据挖掘领域中常用的较为快速为成熟的分类器。它的统计基础使得在罗吉斯回归模型上的研究往往可以给其他分类问题的模型研究带来启发和指导。

近些年来，大规模数据集越来越普遍，从中演化产生的大规模数据应用也越来越多。这些应用的一个核心特征是，训练数据的样本数量非常大，并且数据唯独非常高。例如，在医疗诊断应用中，医生和患者都希望能从上百万条的医疗诊断记录中获得经验，提高诊断正确率，并且一条医疗诊断记录往往有上百维特征可供查询[24]。另一个更明显的例子是现代搜索引擎。这些搜索引擎需要处理十亿级别以上的文本数据以及多媒体数据，并且每条数据的特征空间都在上千维以上[11]。可以想见，如此大规模的数据和高维度数据空间会对机器学习算法在计算能力上提出很高的要求，以致产生很大的挑战。

而罗吉斯回归模型作为其中的典型代表，由于其在PageRank[21]以及Anti-Spam Filtering[1]方面的广泛应用而使其计算效率问题尤为突出。

## 论文工作

本文将从算法层面分析和解决罗吉斯回归模型优化算法的计算效率问题。我们将主要采用并行方式来进行算法加速。

在开发针对罗吉斯回归模型优化算法的并行模式的过程中，我们无法回避这样一个问题：选择什么并行系统，以及选择怎样的并行计算框架。在经过深入调研之后，我们选择了三个独特而流行的系统来进行测试，它们是Hadoop[27]，Mahout[20]以及Spark[29]。

其中Hadoop支持HDFS[3]和MapReduce[7]。而Mahout更像是一个代码库，既可以建立在Hadoop上运行，也可以独立运行。Mahout不能在严格意义上称作为一个系统，但由于其在罗吉斯回归模型上所支持的算法运行框架是独立于Hadoop的，我们在此表述为非严格意义上的系统。同时请读者注意，这并不表明我们认为Mahout和Hadoop完全在一个层次上，因为大部分Mahout所支持的机器学习算法还是依赖Hadoop的。

Spark系统最大的特色在于其对含迭代过程算法的良好支持。它采用了更好的内存和缓存控制策略，并且支持HDFS。我们将会在相关工作部分分别介绍它们的特点。

在针对罗吉斯回归模型的优化算法中，有过很多经典的串行算法。一种的一个代表性例子是随机近似方法。随机近似方法，具体而言又有随机梯度下降（Stochastic Gradient Descent）方法[30]以及随机对偶平均（Stochastic Dual Averaging）方法[28]。它们都可以通过极少量的数据迭代，来趋近于最优优化结果。因而，在训练过程中，当控制预期优化趋近率时，这些算法的运行时间都是正比于数据总规模的，即呈线性。其中最著名的当属随机梯度下降方法。此算法以在线（online）方式运行，我们可以把随机抽取的数据点看作不断引入的数据流。针对此算法进行并行化处理较为困难，但即使是串行，它的运行也十分高效。

如果我们退一步来考察用普通梯度下降算法来解决罗吉斯回归模型的优化问题，尽管作为串行算法它不够高效，但它极易进行并行化处理，从而在大数据背景下大幅提高运行效率，不失为一个潜在的良好算法。我们可以在一次迭代中载入全部数据，以MapReduce的形式计算它们的平均梯度下井方向，从而实现并行。更进一步，我们可以使用次线性方法[22]，更进一步的提高运行效率。由于次线性方法可以充分利用在随机近似方法上，所以解决罗吉斯回归模型的优化问题，同样可以利用次线性方法来降低算法复杂度，提高运算效率。次线性方法在每次迭代中只利用训练数据的一维来代替所有维度进行处理，从而实现加速。我们将在后文的算法部分提出基于该算法的一个并行版本。它可以在保证近似率的进出上更为快速的收敛。

## 论文结构

第1章 引言，介绍了本文的研究背景，论文的相关作，同时简要概括了一下论文结构。

第2章 相关工作，详细介绍了大数据背景下的机器学习算法发展，机器学习算法并行计算框架以及次线性优化方法。

第3章 带惩罚项的罗吉斯回归模型和次线性方法，详细介绍了罗吉斯回归模型的定义，引入了带惩罚项的罗吉斯回归模型；并对次线性方法做了更深入的介绍。

第4章 罗吉斯回归模型优化算法并行框架，介绍了Hadoop MapReduce架构、Spark系统以及Mahout软件上所执行的针对逻辑斯回归模型的优化算法。

第5章 实验环境准备，分别介绍了实验数据集信息、集群信息和具体的测试程序。

第6章 实验结果，给出了实验在2D数据集上的可视化结果，以及所有测试程序在所有数据集上的学习准确度和运行时间，我们分析了上述结果并给出了在不同的数据情况以及不同的系统资源下，我们选择有针对性的算法更为高效地解决罗吉斯回归模型的学习问题的建议。更进一步的，我们研究了不同计算资源对算法的影响以及算法和平台的容错性。

第7章 总结，主要回顾了针对解决罗吉斯回归模型学习问题，在不同数据和不同计算资源下的合理选择，以及次线性方法在此类凸优化问题中的应用。最后提出下一步更深层的研究方向。

# 相关工作

## 大数据背景下的机器学习算法

在大数据背景下，大规模数据集要求我们开发出的机器学习算法更加高效，并行度更大。在机器学习研究领域，学者们已经针对大规模数据集做了不少卓有成效的工作。

早期的工作，有比方说PSVM（Parallel Support Vector Machines）[4]的开发。PSVM利用了近似矩阵分解的方法。这种分解方法是基于行抽取的，它可以在算法运行时大大降低内存使用。该算法同时从而可以通过增加参与并行计算节点的数量来减少算法运行时间。该算法针对机群的可扩展性在上百台这样一个数量级上。

此后，研究者又提出了PLDA (Parallel Latent Dirichlet Allocation) [26]算法。该算法可以大幅提高LDA算法的运行效率。它采用随机抽取的方法。PLDA算法所发布的Hadoop版本具有很强的鲁棒性，它具有承受节点失败的能力，这其中既利用了算法的随机性，也利用了Hadoop的容错优势。

最近，Dean等学者的工作[18]表明并行化的优势可以再深度学习（deep learning）领域起到重大作用。他们的工作可以把机器学习算法的并行度推进到上亿个计算节点。这是迄今为止，机器学习研究领域所达到的最大并行度，同时也促使深度学习算法在图像识别、语音识别和很多其他应用上取得了目前最好的学习效果。

除了Hadoop并行系统所代表的框架，机器学习学术界也有很多对其他并行框架的尝试。GraphLab[17]是卡耐基梅隆大学开发的，专门针对机器学习中大规模图算法的一个计算工具。它可以大幅提升此类型算法的运算效率和在机群上的可扩展性。

## 并行算法框架

本文工作建立在三个并行系统上，这三个系统在运行机器学习算法时的算法框架都有其各自特点。

### Apache Hadoop

Apache Hadoop[27]作为一个软件代码库，实现了对大规模数据集的分布式处理。它被广泛应用与各类机器集群上，并且其使用了简洁的编程模型。它是一个开源并行框架，支持对高密度分布式数据的处理。而Hadoop所支持的机器集群可以是大规模的并且支持商品级硬件。Hadoop是从Google的MapReduce和GFS（Google File System）中演化而来的。Hadoop使用Java语言。它的开源特性保证了全球大量开发者的贡献都体现在了这个Apache的顶层项目中。Hadoop平台在一般意义上，包含Hadoop kernel，MapReduce和Hadoop Distributed File System (HDFS)，以及一些其他相关的项目，包括Apache Hive和Apache HBase。Hadoop的设计目标就是要大幅提高机器集群的可扩展性，从几台服务器上升到成千上万个运算节点。在Hadoop中每个节点都能提供本地的计算和存储。Hadoop框架同时对运行在其上的应用同时透明的提供数据可靠性保证以及数据通信保证。

Hadoop所执行的运算模型为MapReduce[7]。每个应用被分割为大量小片段，每个片段可以在机器集群中的任意一个节点上执行和重复执行。在MapReduce运算模型能够应用的问题中，输入会被解析为键-值对的集合。一个map函数会把这些键值对经过用户所定义的计算转为中间结果的键-值对。一个reduce函数会接着把这些中间结果按照键值索引聚集（其聚集方法是用户所定义的）。事实上，MapReduce所能应用的问题很广泛，不少问题都可以转化为一个或者一系列MapReduce任务。这种运算模型也使得它很易于底层的并行实现。所有的处理都是独立的，所以可以分配给不同的节点分别执行。除此之外，Hadoop还提供了分布式的文件系统HDFS。它可以给整个机器集群提供很高的通信带宽。这用map/reduce以及HDFS的设计保证了当节点失败时可以自动被系统处理而不影响整个任务执行。而这种容错性保证，并不依赖于硬件，而是Hadoop代码库本身的设计在应用层考虑了节点失败的检测和处理。所以尽管每个节点可能是易于出现问题的，但整个机器集群作为一个运算资源来说是可靠地。

### Apache Mahout

Apache Mahout[20]是一个专门针对大规模机器学习算法所开发的代码库。其核心设计目标就是要提高在大规模数据上，提高机器学习算法的可扩展性。它的核心算法包括聚类算法、分类算法和协同过滤。这些算法的很大一部分是建立在Apache Hadoop的基础之上运用MapReduce框架的。同时，Mahout的核心代码库也含有一部分非并行计算程序（针对罗吉斯回归模型的算法就属于这一部分）。它们在代码优化上做了大量的工作，从而达到了很多好的运算效果。Mahout甚至广泛应用与商业领域，它良好的可扩展性，以及对很多流行机器学习算法的完整支持都促进了它的推广。

目前，Mahout主要支持以下四个应用场景：

1. 推荐系统：把用户行为作为输入，并根据用户行为进行个性化推荐。
2. 聚类：例如把文档按照主题进行分类。
3. 分类：例如从已经分好类的文档中学习模型，并将一个新的未被分类的文档分入正确的分类中。
4. 频繁模式：将一系列事项集合作为输入（例如查询词、销售图表等等），从中识别出那些事项模式出现的较为频繁。

Mahout当前支持很多机器学习算法，包括协同过滤算法、基于用户和基于产品的推荐算法、K-Means算法、模糊K-means聚类算法、平均偏移聚类算法、狄利克雷过程聚类算法、LDA算法、奇异值分解算法、并行频繁模式算法、互补朴素贝叶斯分类器和随机决策树分类器。

### Spark

Spark[29]系统是由加州大学伯克利分校的AMPLab开发的。它是一个开源的分布式计算系统。Spark的设计目标是使得数据分析更加快速：一方面是运行时间的加速，另一方面是读写的加速。为了使程序运行得更快，Spark提供了基于内存和缓存的集群计算基本操作：一个任务可以将数据最充分地载入内存中，从而使得反复不断地访问操作要比硬盘读写来的更快。而硬盘读写多这一点，正是Hadoop系统的最大劣势。为了使得在Spark系统上的编程开发更为快速，Spark支持Scala和Python语言，并提供了简洁的API。用户甚至可以以交互式的方式，通过Scala和Python语言来快速处理大数据集。Spark系统在设计之初，主要是面向两类应用问题：迭代程序（这是在机器学习算法中广泛使用的）和交互式数据挖掘。如果能够把需要重复利用的数据始终保存在内存中，这两类应用问题的算法执行效率都必然能够获得极大提升。而随后的测试也表明，相比于Hadoop系统，Spark确实在这两个应用中取得了100倍以上的加速效果。

Spark系统较为年轻，但也已经在工业界获得了不少应用。Spark系统是Shark系统的后端引擎，而Shark系统是一个与Apache Hive系统相兼容的数据仓储系统。同样的，Shark系统也比Hive系统有100倍以上的加速。尽管Spark系统是新开发的一个并行框架，它支持访问HDFS上的数据，从这个意义上讲，它也与Hadoop兼容。这个特性使得很多基于Hadoop的程序不用再重新开发，从而给Spark也带来了更广的应用前景。

Spark是建立在Mesos上的系统。Mesos是一种运行于集群上的操作系统。Mesos可以使得多个并行程序良好地共享一个集群。它也同时提供了API来支持其上的应用在集群上部署并行任务。Spark通过利用Mesos系统可以在集群上与其他系统良好共存，比如说同样建立在Mesos系统上的Hadoop和MPI。另外，利用Mesos系统，使得Spark的开发过程节约了原本所需的大量编程工作。

Spark系统得以成功的关键在于运用了弹性分布式数据集（Resilient Distributed Dataset，简称RDD）这个概念。每个弹性分布式数据集代表一块分布在集群中一部分节点上的只读对象。它具有容错特性，既当一个弹性分布式数据集的分布方式信息丢失时，系统可以自动重构这个弹性分布式数据集，从而保证数据的可靠性。用户可以清晰地在应用程序代码中创建RDD，并根据需要把它载入整个集群的缓存中，并在像如前所述的MapReduce框架下的map和reduce函数中不断重复访问这些数据。弹性分布式数据集通过仅支持线性操作（lineage）来保证容错性。如果一个弹性分布式数据集丢失，系统可以通过它所对应的线性操作方式（也可以是一串线性操作）来对重新生成这个弹性分布式数据集。

## 次线性方法

近些年来，次线性方法逐渐为人所熟知。Clarkson等学者充分运用随机算法提出了这一新的方法[5]。他们在机器学习领域首先提出了这种次线性时间的近似优化算法。他们把这个方法应用在线性分类器和最小闭包球这两个基础问题上。次线性方法结合了一种新型抽样方法和一种新的可乘式更新算法。他们同事证明了算法下界。该算法下界表明算法理论运行时间对于RAM模型来说，已接近最优。

Hazan等学者把这种次线性方法用在了带二阶惩罚项的支持向量机模型上[16]。此后，Cotter更进一步，把次线性方法运用在了了带核函数的支持向量机模型中[6]。随后，Hazan继续把此方法做进一步推广，分别发展到了带一阶惩罚项和带二阶惩罚项的线性回归模型中[15]。Garber和Hazan合作，又把此方法用在了半正定规划问题（Semidenfinite Programming）[10]当中。在2012年，彭昊若等研究人员，在次线性方法上做了进一步研究，把它用在了解决带一阶或二阶惩罚项的罗吉斯回归模型中，并提出了对应的串行算法[22]。

# 带惩罚项的罗吉斯回归模型和次线性方法

## 罗吉斯回归模型

罗吉斯回归模型在机器学习领域中的分类问题上获得了广泛应用。在本文中，特别的，我们将讨论二分类问题。具体的模型定义如下：

### 数据集定义

假设有一个训练数据集为



其中  代表输入的训练样本，而  是相对应的分类标签。

这里  是样本个数，即样本空间大小；而  是样本维度，即特征空间大小。

为了简化表达起见，我们可以将训练数据重组为两部分。一部分为训练数据矩阵，其中每行代表一个训练样本，每列代表一维特征，即



另一部分为分类标签向量，其中的每个元素与训练数据矩阵中的每一行对应，即



### 模型定义

不带惩罚项的简单罗吉斯回归模型可以表述为下面的形式



即定义了每个数据样本所对应的分类标签的条件概率。

其中 是回归向量，而  是偏移量。

这两个变量正是学习算法需要近似优化求解的目标。

### 优化问题定义

为了优化求解上述变量，对于这种显示表达的模型，一种常用的经典方法是考虑最大似然。更进一步，为了后面的计算方便，这里我们考虑取过对数后的最大似然。由于对数函数本身是单调增函数，所以在优化问题中，优化取对数后的最大似然与优化原始最大似然完全等价。因而，在整个训练数据集上的对数最大似然可以表达为



## 带惩罚项的罗吉斯回归模型

在带惩罚项的模型框架中，我们往往需要假设回归向量服从某一特定分布 。这一过程也可以称作先验假设。它使得我们可以将最大似然估计转换为考虑最大后验估计（maximum a posteriori，简称MAP）。所以，在整个训练数据集上的最大后验估计可以表达为



我们下面将会分别推导出一阶和二阶带惩罚项的罗吉斯回归模型优化求解目标的表达式。对于二阶模型，我们会引入拉布拉斯先验假设；对于一阶模型，我们会引入高斯先验。

### 二阶惩罚项

如果我们假设回归向量服从一个高斯分布。假设该高斯分布的均值为 ，协方差矩阵为  。其中 表示一个  的单位矩阵。数学上表达为，



那么，由于



在这种情况下，在式中带入上述表达式，得到



从而，我们就把罗吉斯回归模型的参数优化问题发展为一个带二阶惩罚项的优化目标。

带二阶惩罚项进行优化的优势在于可以得到更为稳定的优化近似解，在算法迭代过程中促进更为快速的求解收敛。

### 一阶惩罚项

同样的过程，如果我们假设回归向量服从一个参数为 的拉普拉斯分布。我们可以得到，



那么，在这种情况下，在式中带入上述表达式，得到



从而，我们就把罗吉斯回归模型的参数优化问题发展为一个带一阶惩罚项的优化目标。

带一阶惩罚项进行优化的相比于带二阶惩罚项进行优化往往更有优势。其独特性在于，一阶惩罚项易于进行稀疏模型构建[16]。因而，从带一阶惩罚项的罗吉斯回归问题求解得到的结果往往既有良好的分类效果，也可以同时用来进行对数据的特征选择。

## 次线性方法

我们所使用的次线性方法的框架是一个分别同时处理的硬边际量和软边际量的混合方法。这种方法可以在同时在硬边际量和软边际量的优化求解过程中进行快速收敛。

### 次线性方法算法框架

在次线性方法中，每个迭代过程需要经过两步。第一步是随机原始更新（Stochastic Primal Update）。它又包含以下两个步骤：

1 从样本数据中随机抽取出一个。其中，每个样本数据的被抽取的概率组成长度为的概率分布向量  。（初始为每个样本等概率，随后概率分布向量 会被不断更新。）

2 通过计算上一步骤被抽取的样本数据的梯度方向，来更新回归向量以及偏移量。注意，这是一考虑带惩罚项的在线(online)更新过程。

第二步是随机对偶更新（Stochastic Dual Update）。它也包含两个步骤：

1 对训练数据集中的每个数据样本都使用的时间计算一个在该数据样本上的硬边际量加软边际量的近似估计值。

2 使用针对在单纯形（Simplex）上做在线优化的可乘式更新（Multiplicative Updates，简称MW）算法，来更新第一个步骤中的概率分布向量 。

### 使用次线性方法的罗吉斯回归模型串行优化算法

我们借助以下符号来定义使用次线性方法的罗吉斯回归模型串行优化算法。我们在引文[22]的基础上做了必要的简化。辅助定义的符号也会在随后涉及的其他算法中使用。

1 我们定义一个投射函数，如下



2 我们用符号代表示性函数，定义为



3 我们用符号来表示罗吉斯函数，即



在算法1中，我们给出了使用次线性方法的罗吉斯回归模型串行优化算法的具体细节。

|  |
| --- |
| Algorithm 1 SLLR |
| 1: Input parameters: |
| 2: Initialize parameters: |
| 3: Iterations: |
| 4: |
| 5: Choose  with probability |
| 6: |
| 7: |
| 8: Update soft margin if input  for -penalty |
| 9: Update  by soft-threshold operations if input  for -penalty |
| 10: |
| 11: |
| 12: Choose  with probability |
| 13: Iterations: |
| 14: |
| 15: |
| 16: |
| 17: Output: |

在上述所示的算法1中，从第4行到第11行是原始更新部分。其中是对梯度方向的估计。从第12行到第16行是对偶更新部分。在这一部分中，是对硬边际量和软边际量加和的估计值。而与此同时，也是的梯度值。尽管第15行和第16行的计算使得成为一个对的带偏移近似值，但这对保障整个算法的稳定性至关重要。而由此所产生的近似误差对最终优化结果的影响很小，可以忽略不计。在这个算法描述中，我们统一了带一阶惩罚项和二阶惩罚项的次线性方法罗吉斯回归模型串行优化算法。两者不同的处理体现在第8行和第9行。如果需要更进一步研究，请参考引文[22]中分开定义的两种算法。

# 罗吉斯回归模型优化算法并行框架

在此章，我们首先将给出我们在Hadoop MapReduce架构下设计的运用次线性方法的并行优化算法。紧接着，我们将会给出一个在Spark系统上运行的运用次线性方法的并行优化算法。该版本算法与Hadoop MapReduce架构下设计的算法略有不同。然后我们将介绍在我们测试中作为基准所使用的传统算法。它们包括在Spark系统下运行的并行梯度下降方法，以及在Mahout中集成的在线随机梯度下降法。

## Hadoop MapReduce架构下使用次线性方法的并行优化算法

我们通过充分利用MapReduce的变成模式，在Hadoop系统上设计了一个针对罗吉斯回归模型的并行优化算法。具体的算法请看下面的算法2、原始Map过程、原始Reduce过程、原始更新过程、对偶Map过程和对偶更新过程。

### 算法框图

算法2 并行-次线性-罗吉斯回归模型-MapReduce架构算法

|  |
| --- |
| Algorithm 2 PSUBPLR-MR |
| 1: Input parameters: |
| 2: Initialize parameters: |
| 3: Iterations: |
| 4: storeInHdfsFile("hdfs://paraw").addToDistributedCache() |
| 5: storeInHdfsFile("hdfs://parap").addToDistributedCache() |
| 6: conf\_primalnew Configuration() |
| 7: job\_primalnew MapReduce-Job(conf\_primal) |
| 8: conf\_primal.passParameters() |
| 9: job\_primal.setInputPath("...") |
| 10: job\_primal.setOutputPath("tmp/primal") |
| 11: job\_primal.run() |
| 12: ()PrimalUpdate() |
| 13: Choose  with probability |
| 14: storeInHdfsFile("hdfs://paraw").addToDistributedCache() |
| 15: conf\_dual  new Configuration() |
| 16: job\_dual  new MapReduce-Job(conf\_dual) |
| 17: conf\_dual.passParameters() |
| 18: job\_primal.setInputPath("...") |
| 19: job\_dual.setOutputPath("tmp/dual") |
| 20: job\_dual.run() |
| 21: DualUpdate() |
| 22: Output: |

过程 原始Map

|  |
| --- |
| Procedure Primal-Map(inputfile) |
| 1: Configuration.getParameters() |
| 2: readCachedHdfsFile("paraw") |
| 3: readCachedHdfsFile("parap") |
| 4: parseRowIndx(inputfile) |
| 5: parseRowVector(inputfile) |
| 6: parseRowLabel(inputfile) |
| 7: *random(seed)* |
| 8: *if* |
| 9: |
| 10: *else* |
| 11: |
| 12: Iterations: |
| 13: Set |
| 14: Set |
| 15: *Output* (key, value) |

过程 原始Reduce

|  |
| --- |
| Procedure Primal-Reduce(key\_in, value\_in) |
| 1: key\_outkey\_in |
| 2: |
| 3: *Output* (key\_out, value\_out) |

过程 原始更新

|  |
| --- |
| Procedure PrimalUpdate() |
| 1: readFromHdfsFile("tmp/primal") |
| 2: |
| 3: |
| 4: |

过程 对偶Map

|  |
| --- |
| Procedure Dual-Map(inputfile) |
| 1: Configuration.getParameters() |
| 2: readCachedHdfsFile("paraw") |
| 3: parseRowIndx(inputfile) |
| 4: parseRowVector(inputfile) |
| 5: parseRowLabel(inputfile) |
| 6: |
| 7: |
| 8: |
| 9: |
| 10: |
| 11: *Output* (key, value) |

过程 对偶更新

|  |
| --- |
| Procedure DualUpdate () |
| 1: readFromHdfsFile("tmp/dual$t$") |
| 2: Iterations: |
| 3: |

### 算法描述

算法2的并行设计基本遵循了使用次线性方法的罗吉斯回归模型串行优化算法的算法框架。我们在并行算法2中依然在每次迭代计算中保留了两个主要部分：从第4行到第12行是原始更新过程，而从第13开始，一直到第21行是对偶更新过程。在原始更新过程中，有并行执行的部分，它们是从第4行到第11行部分；但同时，也又无法避免的串行部分，即由第12行的调用PrimalUpdate函数开体现。而在对偶更新的过程中，情况是类似的，从第14行到第21行是并行执行的的部分，但第13行和21行是串行执行的。

在算法2的并行执行部分，我们充分考虑了MapReduce本身的设计特点。在算法2的job\_primal中，我们将训练数据矩阵全部载入并进行解析和处理，这符合MapReduce以相同方式同时处理大量数据的设计思想。在原本的次线性串行算法中，我们在原始更新的过程中只随机抽取一个样本数据用来计算梯度；而在新设计的并行框架中，我们计算了“一部分”样本数据上的梯度，并根据概率向量进行了加权叠加，并以Reduce过程的的最终结果作为更新的梯度。实现的细节具体参考原始Map过程和原始Reduce过程。在此，我们使用随机方法来确定计算哪些样本数据上的梯度，并进行加权。从原始Map过程的第7行和第8行可以看出，如果取，那么所有的样本节点都将参与梯度计算。又由于概率向量中的期望值是，所以在算法执行中，我们一般会取。

在算法2的job\_dual中，我们设计的是原始串行算法的简单并行处理。并行算法对每个样本数据根据可乘式更新算法计算出一个值。这个过程可以直接分散到各个计算节点上单独完成，甚至可以省略Reduce的过程。

### 进一步增大并行度

更进一步考察，我们事实上可以在实际程序执行中进一步增大并行度，其方法是在同一次迭代计算过程中，同时启动原始MapReduce任务和对偶MapReduce任务。由于这两个MapReduce任务中所访问和修改的参数通过延后更新的准则考虑可以互相隔离，所以这样同时执行的方式和两个任务串行执行的方式是等价的。从而，我们可以在Hadoop MapReduce的框架下更充分地增大并行度，是算法执行更为高效。这样的并行设计在图。。。中得到了更加清晰的说明。

图。。。算法2并行框架设计(需要改)

### 其他设计问题

1 有关参数传递的问题

在Hadoop框架下，合适的选择一种高效地传递参数的方式对性能是尤为关键的。在算法2中，参数传递的过程既有不同迭代步之间的传递，也有不同MapReduce任务之间的传递。同时，另一个挑战是，需要针对Hadoop框架下的HDFS有针对性的进行处理。

很显然，最为快速的通信方式是通过函数进行参数传递。但是，这样的传参方法在Hadoop环境下总是会遇到缓冲区大小限制的问题。一旦当需要传递的参数很大时（高维数据正符合这一点），如果仅使用此方法，将严重危害算法的可扩展性。

另一种方法是把需要传递的参数在信道传递时进行信息压缩。这需要在发送时进行压缩，在接收时进行解压。其弊端是会造成额外的计算开销。并且这种方法也不能从本质上解决参数传递时缓冲区大小限制的问题。

当我们重新回顾Hadoop的设计思想时，可以发现，最自然而然的选择是通过文件传递参数。因为Hadoop本身在自动连接Map任务和Reduce任务之间就包含了通过文件传递键-值对的方法。当然，由于频繁的读写HDFS系统，这种方法并不高效，但这是为了支持大数据环境而采取的合理选择。

2 有关一阶和二阶惩罚项的问题

算法2在数学上严格来说是针对不带惩罚项的罗吉斯回归模型的。如果要是模型带有一阶或二阶惩罚项，我们必须对算法2进行一些小的改动。这些改动只会在原始更新的过程中出现。而改动方法和算法1中第8和第9行所示的方法是完全一致的。这些改动仅发正在串行执行部分，对本身的算法框架和执行没有本质影响。

3 训练数据集的数据稀疏问题

可以被罗吉斯模型所描述的训练数据集，如果具有很高的维度，往往具有数据稀疏的特点。这就要求我们在真正实现算法时，充分考虑这一点，在代码中进行有针对性的优化。但这一点在算法2自身的描述中难以体现，在此特别说明。

我们放弃了针对稠密数据有效地简单存储和运算模式。针对数据稀疏性，我们会分别存储每个样本数据对应的稀疏向量中每个数据点的维度坐标和具体数值。采用这样的方式，自然也会使得所有涉及稀疏向量的运算都要对代码进行相应更改优化，比如说最简单的稀疏向量求点积过程。考虑数据稀疏性的代码优化是极为有效的，也对算法性能带来了显著提升。我们在后文所展示的实验结果均为使用针对数据稀疏性优化后的代码进行测试所得。

## Spark系统上运行的运用次线性方法的并行优化算法

我们设计了在Spark系统上运行的运用次线性方法的并行优化算法。具体的算法请看下面的算法3。而其中所使用的原始更新过程和对偶更新过程是与前面介绍的算法2完全相同的。

该版本算法与Hadoop MapReduce架构下设计的算法略有不同。然后我们将介绍在我们测试中作为基准所使用的传统算法。它们包括在Spark系统下运行的并行梯度下降方法，以及在Mahout中集成的在线随机梯度下降法。

### 算法框图

算法3 并行-次线性-罗吉斯回归模型-Spark架构算法

|  |
| --- |
| Algorithm 3 PSUBPLR-SPARK |
| 1: Input parameters: |
| 2: Initialize parameters: |
| 3: pointsspark.textFile(inputfile).map(parsePoint()).cache() |
| 4: Iterations: |
| 5: gradpoints.map().reduce( \_+\_ ) |
| 6: ()PrimalUpdate() |
| 7: Choose  with probability |
| 8: pAdjustpoints.map(MW-Update()).reduce(copy()) |
| 9: DualUpdate() |
| 10: Output: () |

### 算法描述

算法3与算法2在基本并行框架上的设计非常相似。而它们最主要的区别在于算法3的第3行执行了操作。为了使得算法能够适合于Spark环境，我们遵循Spark的规则生成RDD来更合理高效的利用内存和缓存。

同样，考虑到数据稀疏性问题，我们设计的RDD也采用了存储稀疏向量每个数据点的维度坐标和具体数值的方式。同样的，数据点的维度坐标将和数据点具体数值一样参与针对稀疏向量专门设计的运算。

而针对带惩罚项的罗吉斯回归问题，算法所需要的改动由于体现在原始更新的过程中。由于这一部分算法3与算法2一致，其改动方法也是完全相同的。

### 时间复杂度理论分析

现在，我们可以着手分析罗吉斯回归模型优化问题并行算法的时间复杂度。在理想的并行状态下，原始更新的串行部分需要更新回归向量，这需要的时间来完成；并行部分可以看做在时间内完成。而在对偶更新过程中，串行部分含有一个随机抽样过程来抽取，需要的时间；并行部分更行概率向量，可以看做在时间内完成。所以，总的来说，每个迭代需要的时间，这正体现除了对训练数据集的次线性。

将此结果和算法1所代表的次线性串行算法相比较，可以发现两者相同，即上述的并行算法设计并没有降低算法复杂度。但是考虑常量的话，并行算法可以把2降为1，即在完全不考虑额外计算和通信开销的情况下，理论上速度能快一倍。更进一步，如果同前所述，将每次迭代过程中的两个MapReduce任务同时启动，那么理论上的算法复杂度可以降为。

## Spark系统上运行的并行梯度下降法

我们设计了在Spark系统上运行的并行梯度下降法。具体的算法请看下面的算法4。此算法为经典算法，但与前文统一了描述风格。

### 算法框图

算法4 并行-梯度下降-罗吉斯回归模型-Spark架构算法

|  |
| --- |
| Algorithm 4 PGDPLR-SPARK |
| 1: Input parameters: |
| 2: Initialize parameters: |
| 3: pointsspark.textFile(inputfile).map(parsePoint()).cache() |
| 4: Iterations: |
| 5: gradientpoints.map().reduce( \_+\_ ) |
| 6: |
| 7: |
| 8: Output: () |

### 算法描述

算法4与其对应的串行算法而言，是经过了最为简单直接的并行化处理。算法4每次载入所有数据，并以类似于MapReduce任务的方式来计算出平均梯度。而至于算法4的函数运用以及针对数据稀疏性而特别设计的RDD都与算法3一致，在此不做赘述。

## Mahout系统上运行的在线随机梯度下降法

尽管随机梯度下降方法本质上是一个串行算法，但它运行非常高效。并且又由于是在线算法，运行时内存占用低。所以此算法并不影响Mahout处理百万数量级以上的训练样本集。由于该算法采用自顶向下的抽样方法，这样的可扩展性其实是和其他算法直接处理十亿级别以上的数据量等价。在线算法其实也是一个增量式训练模式，并且我们还可以在训练的过程中就进行性能测试。这样从而可以使我们在训练模型达到预期训练效果时就终止训练过程。

在Mahout中使用的随机梯度下降模块包含使用交叉验证（Cross Validation）的在线评估方法（the CrossFoldLearner）和一个革命性的可以在运算过程中继续进行超参数优化的系统(the AdaptiveLogisticRegression)。这个系统大量使用多线程来弥补并行上的不足，从而提高机器节点利用效率。Mahout会控制一系列的CrossFoldLearners运行在不同的线程上。其中的每个学习器在进行学习时都设定了不同的学习参数。当更好的学习设定被找到时，这些新的学习设定将会被系统传播到其他学习器上。

由于随机梯度下降的算法需要固定长度的特征向量，又因为在学习过程之前就建立完全索引的开销太大，大部分的随机梯度算法使用哈希后的特征向量空间系统。这个系统来自于RandomAccessSparseVector。用户可以根据需求，使用多种特征编码方式来大幅提升这个向量的特征维数。而由于要满足哈希条件，所以向量需要足够大来防止特征冲突。Mahout中本身集成了很多专门为各类数据类型提供的特征编码器。一般而言，用于可以将数据编码为字符串形式，或者为了避免字符串转换操作而把数据按字节编码。

在我们的实际实验执行中，我们直接使用RandomAccessSparseVector来应对数据稀疏性，并用OnlineLogisticRegression来进行模型训练。同时也进行交叉验证。为了使得该算法的程序能够更好地与其他算法程序兼容，我们自己写出代买进行交叉验证。

# 数据污染攻击与防御

## 数据污染攻击的实现

### 基于WSN TMOTE平台上的攻击实现

数据污染攻击的实现我们有两个方案。

第一个方案，数据污染的攻击实现可以基于黑洞攻击来完成，将接收到的数据进行修改再发送到正常的Sink节点。然而由于实验的限制，无法同时观测更多的节点。所以这里将进行一些改动，提出了第二个方案，将直接将从原始节点发送的数据包进行改动，直接将其改成错误的数据包，这样的效果是一样的，并且将其发送到Sink节点，这样Sink节点将接收到错误的数据包。这就是数据污染攻击实现的原理，这实际上是模拟了传输中的误码，也就是在传输过程中某些数据出现传输错误导致最后结果的错误，由于只有简单的检错机制，所以要将数据完全还原后才能知道数据错误，或者是由于凑巧校验位未能检测出错误，这时可能就会引起严重的后果。这就是数据污染攻击的实现方法。

### 进行数据污染攻击后的效果

经过数据污染攻击后，Sink节点依然能够接收到数据，但是接收到的数据经过还原以后不再是原始数据，而是错误的数据，虽然一些数据依然能够通过校验码检测出错误，但是这样浪费了许多资源，而且要将数据还原后才能看出来。经过这种攻击后，往往效果不是很明显，但是危害却十分大。

## 数据污染攻击的防御算法分析

之前谈防御算法时说到了动态验证的机制，这个算法的主体思想是由Qian Wang提出的，然而这个算法应用在这个平台上却有问题。

最主要的问题是原算法是将数据发送到周围的节点上，而这里实现起来不太方便，由于节点容量限制，这样的话需要在所有Sink节点上和普通节点上同时实现该机制，并且需要将这些数据存储下来才能进行动态验证，而这里的节点在防御黑洞攻击时在普通节点上已经加了几个模块，可能导致空间不足。这就需要对其进行一些简化以及修改，否则可能将无法实现其基本思想。

## 算法实现

### 算法简化

首先，鉴于前面讨论的问题，我们可以对其进行一些修改，既然普通节点无法容纳更多的内容，那么我们就让Sink节点来进行这一环节，普通节点只负责进行一些预处理，发送的包中首先是经过之前防御黑洞攻击时的一些处理，得到了一些包，这些包包含这个包属于数据的序号，黑洞攻击中形成的RS码。在这基础上添加了AS签名，以及前面算法中所说的生成中的任意元素。

而现在只有Sink节点负责接收这些数据，并且将这些数据保存起来。验证数据完整性时，首先需要发送完整性验证请求，这个请求中首先需要需要包含的是序号，即你需要验证哪个数据的完整性，这是必然要存在的。其次需要一个验证的特征码，是从中随机选出的一个基数。另外还需要一个，这是需要获取的数据数量。接收到验证信息后，Sink节点将返回一个自己计算的AS签名，根据前面理论部分分析的机制，验证是否成立即可。若该式成立则验证通过，说明数据时完整的，可以被还原。

### 基于WSN TMOTE平台的防御实现

起初，我试验了在所有节点加上该机制，发现普通节点无法容下，所以才有了上面的想法，只对Sink节点实现该机制，普通节点仅作一些数据的处理上的工作。

在Sink节点上，增加了一个接收和发送数据的模块，定义了一个常量AM\_TESTPACKET，用以区分其他正常包，这个数据包就是接收和发送验证信息的包类型。由于AM\_TESTPACKET又分为两种，但是这两种包很容易区分开来。验证请求由于是通过广播的形式发送出去的，而验证请求的返回则是发送到具体一个地址。所以只要判断其目的，若是广播地址则为请求，此时将作回应，相反，若不是广播地址，则进行接收并且按照前面所述的算法进行验证。验证以后若通过则该包为完整的包，若不通过则该包不完整，将被丢弃。

这就是总的实现思路。然而具体的实现过程依然还是有许多问题的。最大的问题依然是空间问题，硬件上本身是无法保存较多包的，只能保存一小部分，那么这个数组开多大则是需要详细考虑的。另外，由于一些数据结构上的问题，还要对实际实现作一个简化，由于是以队列来保存接收到的包，然而对于队列来说丢弃一个中间包是比较困难的，而且调试起来较困难。所以此处将最后校验的过程移出程序，让其将收到的包都发送到Uart端口，另监控软件接收包并且记录下来，我们最后再通过包的分析来看结果。通过这样的简化，就可以进行调试以及分析。

## 实测防御效果及分析

首先我们发送正常数据。接收到的所有数据都是正确的，可以通过AS的检测。通过抓到的包可以验证其结果是正确的。

接下来将节点4替换为攻击的程序，将第三个包的第一位进行更改，然后将其发送，此时抓包得到其结果。

每10秒发送一个数据，按照前面理论，Node 5将发送三个数据包到三个不同的Sink节点Node 2,Node 3,Node 4。每隔5分钟记录一次并且对结果进行验证。可以得知每5分钟发送30个数据，首先取单个节点Node 2的验证结果，测试结果如图5.1。可以看到单个节点验证成功率约为50%。这实际上是因为采用该伽罗华域时，AS签名实际上只有一位，所以签名只有两种可能，故理论上单个节点只有50%的成功率。

而接下来看看综合的验证结果，三个节点的验证结果若有一个为失败则结果不正确。测试结果如图5.2。这里的成功率大大提升。达到约90%的成功率。理论上分析得知，3个节点碰撞概率约为12.5%。

这里仅仅使用了GF(4)若采用更高次的伽罗华域，碰撞概率将会非常小。比如GF(8)上的验证采用7个节点，验证成功率将超过99.99%。

关于通信的开销，可以看到该方案对于通信的开销是非常小的，取决于采用的伽罗华域，本文的实验中采用的GF(4)在通信上的开销，实测得到去掉包头是约为20个字节，包括包头则是40个字节，该值不随数据大小而变，因此是比较稳定的。若采用更高次的伽罗华域，开销将会增大，但是依然是可控范围内的。因为它不随数据位数的变化而变化。若是采用GF(8)其开销也在每个包500B以内，测试得到的结果还是比较好的。

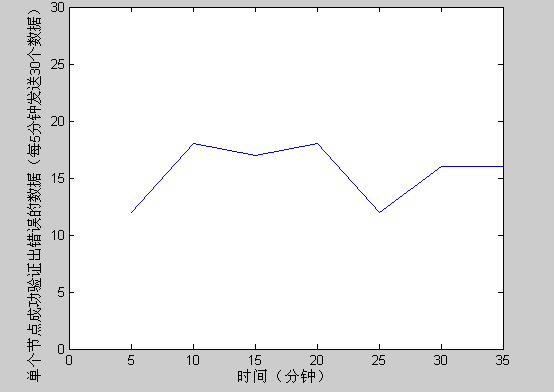
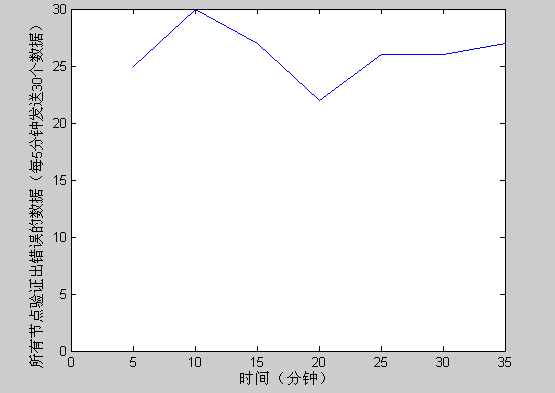


图 .2 多个节点检测数据污染攻击效果图

图 .1 单个节点检测数据污染攻击效果图

## 数据污染攻击与防御小结

本章主要介绍了数据污染攻击与防御的实现，可以成功地检测出大部分被污染的数据，以提高数据的可用性，可以减少一些误码或者是人为篡改数据造成的影响。但是，实际上这种基于AS签名的验证还是有碰撞的，前面所述的GF(4)上的算法碰撞率约为30%。尽管如此，其安全性已经得到了明显的提升。

# 总结

## 论文主要工作的总结

### 工作成果

建立了一个WSN TMOTE平台，基于该平台可以进行一些无线传感器网络的研究，并且在上面实现了黑洞攻击以及数据污染攻击，并且利用了一些防御机制一定程度地减弱了黑洞攻击和数据污染攻击的影响。

针对黑洞攻击主要采用分散发包的思想，抑制了黑洞攻击的效果，使得接收包的成功率得到了显著地提升，并且在WSN TMOTE平台上通过了测试，验证其可行性。

而对于数据污染攻击则进行了一些验证，采用AS签名进行验证，能够识别出大多数被污染的数据，以将其丢弃，保留正常的节点。在实现时将算法简化，以让其能够在平台上成功运行。最后依然有比较好的效果。

### 工作难点

首先是理论部分，理论部分涉及到一些比较深层次的数学理论，其中查阅了许多资料用以理解该理论，主要是对于伽罗华域的理解，这是一个比较抽象的内容，因此花了不少时间。

其次是关于实验平台，本平台依然有着许多缺陷亟待改进。首先是采用tinyos-1.x，无法与eclipse比较好的整合，这样就没有一个好的IDE，编写起来十分困难。并且编译一次要耗费较长的时间，经常若想要其语法通过就需要大约6-7次的编译，而编译一次需要将近10分钟。这与其编译机制有关，首先对各个模块进行编译，由于模块比较多，所以要等待很长时间才会得知具体出错是在哪个部分。编译通过以后，后面的调试就更是困难。由于其没有很好的模拟机制，只能在硬件上直接运行，通过发包来进行调试，因此修改一次要花费至少15分钟时间才能抓到足够的包进行分析，这些都是平台的一些问题。

最大的问题依然是实验平台的空间限制，曾经想将算法完整实现，但是后来发现完全不可行，其可编程容量太小，花了大量时间将其简化至可以容下，并且依然可以保持原来的效果，文中提到的几个是比较大的修改，程序中更有无数细节上的修改。

## 进一步的研究工作

首先是需要升级平台，若采用tinyOS-2.x将会与eclipse更好地结合起来，这将减少大量的工作量。

其次WSN TMOTE平台虽然容量小，但是依然可以作一些基本的研究，只是只能进行小规模的测试。但这对于一些研究来说已经足够。在这上面可以进行一些无线传感器网络协议的研究，无线传感器网络安全的研究等等。应该说还是有一定的可扩展性。通过这个平台可以创造一个新的研究思路，首先理论分析，然后通过软件模拟，再通过该平台进行小规模的测试，可以改善现在大多数研究仅仅停留在理论分析或者模拟的层面。

插图索引

[图 3.1 Delta程序结构图 10](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775836)

[图 3.2 MultiHop程序结构图 11](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775837)

[图 3.3 监控程序界面图 12](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775838)

[图 4.1 黑洞攻击之前的状态图 14](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775839)

[图 4.2 黑洞攻击后攻击节点的状态图 15](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775840)

[图 4.3 黑洞攻击后原正常节点的状态图 15](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775841)

[图 4.4 遭受黑洞攻击后接受到数据 16](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775842)

[图 4.5 实现防御算法后的效果图 17](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775843)

[图 4.6 采用随机发送2个节点效果图 18](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775844)

[图 4.7 采用RS码后成功接收到数据效果图 19](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775845)

[图 5.2 多个节点检测数据污染攻击效果图 23](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775846)

[图 5.1 单个节点检测数据污染攻击效果图 23](file:///D:\TDDOWNLOAD\综合论文训练论文写作指南\综合论文训练-无翻译.docx#_Toc265775847)

表格索引

[表 2.1 伽罗华域的三种表示形式 6](#_Toc265775848)

[表 2.2 伽罗华域GF(4)上的加法表 6](#_Toc265775849)

[表 2.3 伽罗华域GF(4)上的乘法表 6](#_Toc265775850)

[表 4.1 针对黑洞攻击的两种冗余码效果比较 18](#_Toc265775851)

参考文献

1. C. Karlof, D. Wagner, Secure routing in sensor networks: attacks and countermeasures, in: Proceedings of the 1st IEEE Workshop on Sensor Network Protocols and Applications, May 2003, pp. 1–15.
2. A.D. Wood and J.A. Stankovic, Denial of service in sensor networks, *IEEE Computer* **35** (2002), pp. 54–62.
3. C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: Proceedings of MobiCom ’00, Aug 2000, pp. 56–67.
4. F. Ye, A. Chen, S. Lu, L. Zhang, A scalable solution to minimum cost forwarding in large sensor networks, in: Proceedings of ICCCN ’01, Oct 2001, pp. 304–309.
5. D. Braginsky, D. Estrin, Rumour routing algorithm for sensor networks, in: Proceedings of the WSNA ’02, September 2002, pp. 22–31.
6. Masateru MINAMI, Takashi MORITO, Hiroyuki MORIKAWA, Tomonori AOYAMA, Solar Biscuit: A Battery-less Wireless Sensor Network System, The 2nd International Workshop on Networked Sensing Systems, 2005.
7. Dawei Xia, Natalija Vlajic, Near-Optimal Node Clustering in Wireless sensor Networks for Environment Monitoring, 21st International Conference on Advanced Networking and Applications (AINA '07)
8. S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In IPSN ’07: Proceedings of the 6th international conference on Information processing in sensor networks, 2007.
9. J. A. Stankovic, et al, “Wireless Sensor Networks for In-Home Health J. A. Stankovic, et al, “Wireless Sensor Networks for In-Home Health J. A. Stankovic, et al, “Wireless Sensor Networks for In-Home Health 2-3, 2005.
10. Bekmezci, Ilker Alagoz, Fatih, "Energy Efficient, Delay Sensitive, Fault Tolerant Wireless Sensor Network for Military Monitoring," Sensors Applications Symposium, SAS 2008. IEEE, pp. 172-177, 2008.
11. M Hamdi, N Boudriga, M S.Obaidat, An optimized broadband sensor network for military vehicle tracking, Volume 21, Number 3, pages 277-300, 2008
12. M. Al-Shurman, S-M. Yoo, and S. Park, “Black Hole Attack in Mobile Ad Hoc Networks,” ACM Southeast Regional Conf. 2004.
13. DENG Hongmei，LI Wei，AGRAWAL D P．Routing Security in Wireless Ad Hoc Networks，[J]．IEEE Communication Magazine，2002，40(10)：70～75
14. SONG Joo—han．Secure Routing with Tamper Resistant Module for Mobile Ad Hoc Networks[EB/OL]．
15. Yang-De Ming, PanJin, Zhao Shuang , Instrusion detection method for mobile ad-hoc networks based on machine learning, Computer Applications,2005-11
16. SJ Lee, M Gerla, [AODV-BR: Backup routing in ad hoc networks](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.3261&rep=rep1&type=pdf), Proceedings of IEEE WCNC, 2000
17. K Walsh, EG Sirer ,Fighting peer-to-peer spam and decoys with object reputation, Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems
18. BD Klein, DF Rossin,Data quality in linear regression models: effect of errors in test data and errors in training data on predictive accuracy, Informing Science, 1999
19. Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” Proc. of IEEE INFOCOM, 2009.
20. 彭志楠，叶丹霞，范明钰 [移动Ad hoc 网络的黑洞攻击研究](http://www.cqvip.com/qk/93231x/2009011/31896089.html)计算机应用研究, 2009
21. L.E.Dickson, Linear Groups with an Exposition of the Galois Field Theory,Teubner,Leipzig, 1901.:3-14
22. S. Lin and J. J. Costello, Error Control Coding. Englewood Cliffs, NJ: Prentice-Hall, 1983.:236-237
23. J. Dong, R. Curtmola, and C. Nita-Rotaru, “Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks,” in WiSec ’09: Proceedings of the second ACM conference on Wireless network security. NewYork, NY, USA: ACM, 2009, pp. 111–122
24. D Sannella, A Tarlecki [Toward formal development of ML programs: foundations and methodology](http://www.springerlink.com/index/P833856246K0273P.pdf) TAPSOFT'89, 1989
25. C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, “System architecture of a wireless body area sensor network for ubiquitous health monitoring,” Journal of Mobile Multimedia, vol. 1, no. 4, pp. 307–326, 2006.
26. P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for wireless sensor networks. In Ambient Intelligence. Springer-Verlag, 2004.
27. GAY, D.,LEVIS, P., VON BEHREN, R.,WELSH, M., BREWER, E., AND CULLER, D. 2003. The nesC language: A holistic approach to networked embedded systems. In Proceedings of Programming Language Design and Implementation (PLDI 2003).
28. S Nellen Eclipse Plugin for TinyOS Debugging 2009
29. Ed Callaway, Paul Gorday, and Lance Hester, Motorola Laboratories Jose A. Gutierrez and Marco Naeve, Eaton Corporation Bob Heile, Appairent Technologies Venkat Bahl, Philips Semiconductors
30. CP Lin, CK Hsieh, USB/UART converter and its control method, US Patent 6,151,653, 2000

致 谢

衷心感谢林闯教授给予我如此好的实验环境。

感谢实验室的师兄们。由衷感谢曾荣飞师兄，姚敏师兄，孟坤师兄没有你们的悉心指导，与你们的讨论使我受益良多。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：\_\_\_\_\_\_\_\_\_\_\_\_\_ 日 期：\_\_\_\_\_\_\_\_\_\_\_\_