硕 士 学 位 论 文

Hadoop 环境下基于 SVR 的短时交通流预测

Short-term Traffic Flow Forecasting Based on SVR in Hadoop Environment

作 者 姓 名：周常胜

学科、专业：计算机应用技术

学　　号：21109201

指 导 教 师：姚卫红 副教授

完 成 日 期：2014-05-08

大连理工大学

Dalian University of Technology

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下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内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：Hadoop环境下基于SVR的短时交通流预测

作 者 签 名：周常胜　　日期：2014 年 6 月 7 日

摘　要

实时准确的交通流预测是实现智能交通诱导的前提和关键，也是智能交通系统的主要组成部分。早期研究人员利用微积分和概率论等基本数学工具开发了几种基本的短时交通流预测模型，这些模型主要包括趋势外推法、时间序列法、回归分析法等。然而，城市道路交通系统是一个有人参与的非平稳随机系统，具有时空相关性、非线性等特点。天气情况、驾驶员状态、突发性事件等都会对交通系统的性能产生影响，因此传统的方法预测效果并不理想，这使得人工智能理论越来越受到人们的关注。

支持向量回归机（SVR）是一种基于统计理论的新兴机器学习方法。它坚持结构风险最小化的构建原则，能够在小样本学习环境下有效解决非线性复杂系统的拟合问题，在短时交通流预测中表现出了优越的性能。大量实验表明，其预测精度相对于神经网络和传统线性预测方法具有明显的优势。本文分析了参数选择对支持向量回归机性能的影响以及不同参数优化算法的优劣势,在此基础上选择模拟退火算法（SA）对 SVR 的参数行进优化，进而提出了基于 SA-SVR 的短时交通流预测模型。该模型将支持向量回归机的结构风险最小化特点和模拟退火算法的快速全局寻优优势相结合，在实时性和精度两个方面达到了短时交通流预测的要求。实验表明，该预测模型的平均相对预测误差为 4.96%,最大相对预测误差为 9.81%。

随着训练样本规模的增大，传统 SVR 训练算法的时间和空间复杂度急剧增加，以至于这些算法的单机实现在具体应用中往往会失效。本文从“迭代”和“并行”两个角度详细阐述了解决这一问题的具体算法，并将 MapReduce 分布式处理框架与大规模 SVR 训练算法的“迭代”思想相结合，提出了 Hadoop 环境下大规模 SVR 训练算法，进而构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型，在保证预测精度的情况下进一步提高了短时交通流预测的实时性。实验表明，在训练样本数量为 100000 的情况下，该并行算法相对单机算法的加速比达到了 16.03，平均相对预测误差为 5.20%，最大相对预测误差为 10.70%。

关键词：短时交通流预测；支持向量回归机；参数优化；模拟退火；Hadoop

Short-term Traffic Flow Forecasting Based on SVR in Hadoop Environment

Abstract

An accurate forecasting of short-term traffic flow is a prerequisite of intelligent traffic guidance as well as an essential comportment of Intelligent Transportation System(ITS). Traditional forecasting method consist of regression analysis, trend expansion and time series. However, a traffic system is a time-varying, non-stationary stochastic system which tends to be influenced by climatic conditions, driver psychological status, emergencies and accidents, and many other factors. So the traditional linear forecasting methods are not able to forecast the short-term traffic flow with an ideal accuracy and the theory of artificial intelligence enjoys its popularity among an increasing number of researchers.

Support Vector Regression machine(SVR), a machine learning method based on statistical learning theory, can solve the nonlinear problem with a small sample set provided and has proved superior to neural networks and other methods in short-term traffic flow forecasting. This thesis analyzes the importance of parameter selection for support vector regression and the advantages and disadvantages of different parameter optimization algorithms. Then it selects the simulated annealing algorithm to optimize the parameters of SVR, and proposes a short-term traffic flow forecasting model based on the SA-SVR algorithm. Since the model takes full advantage of the minimum structure risk of SVR and the globally optimizing ability of SA, it can meet the requirement of real-time and get excellent forecasting accuracy. The mean forecasting deviation of the model is 4.96%, with the maximum is 9.81%.

With the increase of the training sample size, the time and space complexity of the traditional SVR training algorithms increase dramatically. So these algorithms implemented on a stand-alone server often fail to run. This thesis elaborates the existed algorithms used to solve this problem, they are based on the idea of "iteration" and "parallel" separately. Then we combine MapReduce, an excellent distributed processing model, and the ideas mentioned above to develop a large-scale SVR training algorithm running on Hadoop platform and build a short-term traffic flow forecasting model based on SA-SVR and Hadoop. The speedup of this model is up to 16.03. The mean forecasting error is 5.20%, with the maximum is 10.70%.

Key Words: Short-term Traffic Flow Prediction; Support Vector Regression; Parameter Optimization; Simulated Annealing; Hadoop

目　录

1 绪论

1.1 研究背景和意义

交通运输与人们的生活息息相关，也是拉动国民经济的重要支柱。随着经济的发展，我国机动车持有数量急剧增加，由此带来的交通拥堵、交通环境污染和交通事故等问题给城市的可持续发展带来了严峻的挑战。其中，交通拥堵问题与人们的驾驶行为密切相关，同时具备随机性、时空相关性和混沌性等特点，一直是世界公认的研究难题。据统计，美国 39 个主要城市每年因交通拥堵造成的经济损失约为 410 亿美元，欧洲每年因交通拥堵造成的经济损失约为 5000 亿欧元，日本因道路交通拥堵造成的年经济损失约为 15 兆日元，2013 年交通拥堵给我国带来的经济损失约为 2000 亿人民币。2013 年北京市居民每天上下班途中所消耗的时间平均为 45.04 分钟，相比 2012 年增长了 18.53%。上海市居民的平均通勤时间为 43.21 分钟，而这一时间 2012 年仅为 36.52 分钟。严重的交通拥堵还会诱发城市环境污染问题，据不完全统计，我国大城市 78%的一氧化碳（CO）、46%的氮氧化物（NO_x）和 83%的碳氢化物（HC）污染是由机动车排放造成的。机动车排放的尾气中，CO、NOx、HC 和 PM2.5 等有害物质对人的健康和城市环境造成了巨大的危害。由此可见，交通拥堵不仅浪费了人们的时间而且引发了严重的环境污染问题，其带来的损失更是难以估量，已经成为影响国计民生和经济可持续发展的重大问题。

目前，各个国家主要采用三种方法治理交通拥堵问题：一是增加供给，就是超前进行城市基础设施建设，提前修路建桥，增加公交路线、地铁和轻轨等，在交通拥堵问题严重之前消除拥堵隐患；二是限制需求，即当发生拥堵以后，采取必要的措施降低交通资源的需求量，如限制购车、单号或双号限行等；三是规划空间，即以科学的城市规划为基础加大主城区周围卫星城市的建设力度，并使其承接主城区的部分城市职能，最终实现城市人口和交通压力的自然分流。以上三种方案虽然能够在某种程度上对当前的交通拥堵问题起到缓解作用，但是仍然不能从本质上将其解决。最终，人们还是应该通过建立科学化、智能化的交通体系来实现消除交通拥堵的目的。

智能交通系统（Intelligent Transportation System，ITS）是综合考虑了人、车、路，将信息技术、电子传感技术、数据通讯技术、计算机技术及控制技术等进行有效集成并运用于整个地面的交通管理系统[1]。智能交通系统以现有的交通设施为基础，结合高效的信息采集和处理技术，可以在大范围内对交通系统实现状态监测和反馈控制，具备很强的实时性、准确性和高效性。实际应用表明，智能交通系统可以有效地降低交通负荷、

缓解交通环境污染，极大地提高交通运输的安全性和实效性，因而该系统在欧美日等发达国家已经得到了广泛的应用。据统计，智能交通在美国的应用率达到 80%以上，2010 年市场规模达到 5000 亿美元；日本 1998-2015 年的市场规模累计将达 6250 亿美元，其中基础设施投资为 750 亿美元、车载设备为 3500 亿美元、服务等领域为 2000 亿美元；欧洲智能交通在 2010 年产生了 1000 亿欧元左右的经济效益。我国在 2009 年制定的十大振兴产业计划中，做出了将 4 万亿投入到交通基础设施建设中的重要决定，其中至少有 2%-4%用来做信息化建设投资到智能交通系统。截至 2013 年 8 月 10 日，全国共有 19 个省市公布了智能交通投资计划，涉及投资金额高达 78.05 亿元。智能交通的迅速发展带动了传感器技术、通讯技术、GIS 技术(地理信息系统)、3S 技术(遥感技术、地理信息系统、全球定位系统三种技术) 的快速发展，同时物联网技术和移动终端技术也取得了突破性的进展，在此背景下“车联网技术”应运而生。

车联网将车与车、车与路、车与人互联成一个庞大的交通系统网络，以此为基础对路网的实时交通信息进行收集，并在云计算信息平台上对采集到的交通信息进行存储、计算和安全发布，实现路网信息的全面共享[2]。车联网是物联网在智能交通领域的应用，本质上仍是智能交通系统的重要组成部分。车联网以多源的数据采集方式组合出大量、全面的交通数据信息，可以为交通系统管理程序提供充分的数据支撑。我们通过对这些交通数据的研究，可以挖掘出它们内在的依赖关系，设计出合理的模式，然后再将具体的操作反馈到实际的交通系统中，以闭环的方式实现交通系统的监测与管理。然而，车联网应用服务得以实施的先决条件是拥有准确、实时的交通流量预测。如果我们可以准确预知未来几分钟的交通流量，就可以提前通过交通信号灯、交通控制管理系统等对路面交通状况进行调控，从而达到智能调控交通系统的目的，确保人们安全、有序通行。按照预测时间的长短，交通流预测可分为长时交通流预测和短时交通流预测。

长时交通流预测通常以小时、天、月甚至年为单位进行预测，其意义是宏观上的。短时交通流预测是指对某个路段未来 5-15 分钟时间内的交通流状态进行预测，其意义是微观上的，两者之间有着本质的不同。短时交通流预测具有高度的非线性、不确定性和时间相关性，属于典型的时间序列预测。它以历史交通流数据为基础进行深层次的数据挖掘，结合有效的道路交通流动态分析技术，实现对交通流状态实时、准确的预测。短时交通流预测是智能交通系统和车联网应用服务的前提和关键，是实现交通系统中交通控制、交通诱导和交通信息服务的重要基础。

1.2 短时交通流预测研究现状

短时交通流预测技术以历史交通流量的变化趋势为基础，结合当前的交通流量对未来的交通流量进行预测。从上个世纪 90 年代开始，研究人员就一直尝试着用各种方法进行短时交通流预测。早期的预测模型有自回归滑动平均模型、指数平滑模型、Kalman 滤波模型和小波模型等[3]。然而，由于受到天气情况、驾驶员状态、突发事件等多种因素的影响，短时交通流表现出高度的不确定性，这使得传统预测方法的预测效果并不理想。随着非线性动力学的不断发展，一些更加先进的短时交通流预测模型被开发出来，它们具备预测精度高、实时性强等特点。根据预测理论的不同，短时交通流预测模型可以分为：统计理论模型、非线性理论模型、智能理论模型、微观交通仿真模型和混合模型[4]，详细分类如图 1.1 所示。

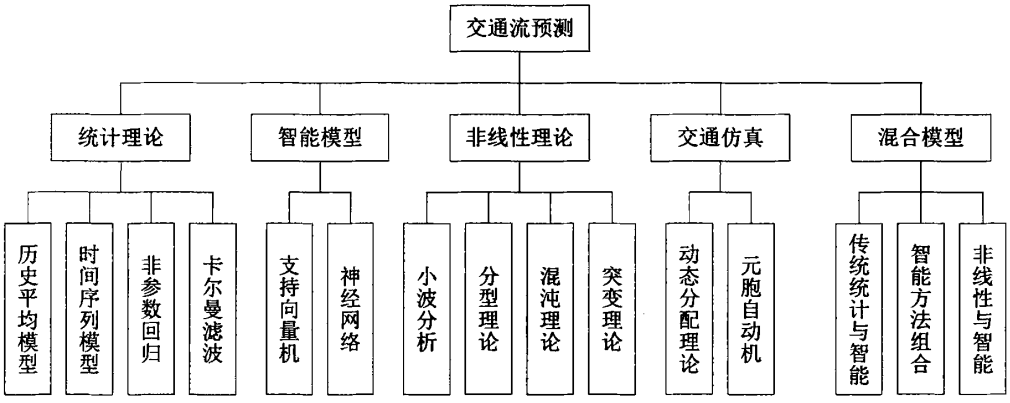

图 1.1 交通流预测模型分类示意图

Fig. 1.1 Classification of traffic flow forecasting models

（1）基于统计理论的模型

该类模型运用数理统计和微积分的基本原理构建预测模型。它以历史交通流数据、现有交通流数据和未来交通流数据的共同统计特性为基础，通过严格数学推导构建短时交通流预测模型，最后实现预测。该类模型主要包括：时间序列法、参数回归法、历史平均模型、自回归滑动模型和卡尔曼滤波模型等。1979 年 AhmaedS.A 和 CookA.R 首次利用时间序列模型进行短时交通流预测。Okutani 和 Stephanedes 于 1984 年提出利用自回归移动模型来解决交通流预测问题。国内，杨兆生在 1998 年以卡尔曼滤波模型为基础进行了交通流预测方面的相关研究，证明了该模型相比时间序列模型具备更高的预测精度。

（2）基于非线性理论的模型

非线性预测主要以混沌理论、自组织理论为基础，利用分型和相空间重构方法构建交通流预测模型。该模型能够描述短时交通流的不确定性和非线性问题，取得了良好的预测效果。20 世纪 70 年代 B.B.Mandelbrot 提出了分形理论，该理论能够充分描述交通流系统确定性和随机性的内在联系，是一种相对全面的交通流分析工具。以此为基础，研究人员建立了有效地交通流预测模型并在实际应用中取得了很好的预测效果。1989 年，Disbro 和 Frame 等人首次将混沌理论应用到交通流预测领域。2003 年，宗春光等人基于相空间重构理论对短时交通流预测进行了深入研究，取得了满意的预测效果。

（3）基于智能理论的模型

基于智能理论的预测模型无需再输入和输出之间建立一个严格的数学关系。该类模型首先基于输入数据进行训练学习，进而通过学习到的知识对后续数据进行预测。目前，应用比较广泛的智能理论有神经网络和支持向量机两种。神经网络(ANN)算法具有很强的自学习能力。它基于人为设定的初始条件和输入样本进行训练并以更新权重因子的方式来习得非线性系统的外在映射特性，从而可以在不考虑数学模型内部结构的前提下实现对短时交通流量等任意非线性函数的模拟和逼近[5]。LGIS 中心在 1998 年将神经网络应用于交通流预测，证明了该模型比时间序列模型和卡尔曼滤波模型具有更好的性能。但是神经网络的学习方法采用经验风险最小化原理，在实际应用中常常会出现过学习、欠学习和局部极值等问题。针对上述缺陷，Vapnik 等人提出了基于统计理论的 SVM 模型，该模型能够较好地解决小样本、非线性等问题，泛化能力很强。徐启华等人基于 SVM 对短时交通流预测进行了深入研究，证明了该模型优于 ANN 和其他方法。

（4）微观交通仿真模型

交通仿真模型是一种行之有效的交通分析工具。该模型把车辆当作研究对象，利用计算机仿真技术对真实的道路交通系统进行模拟，进而通过模拟系统来获得相应交通信息的估计值。Chrobo 和 Wahl 首次引入元胞自动机算法对大尺度交通网络进行交通流预测。2008 年李奎等基于微观交通仿真技术对短时交通流进行了预测研究，该研究只针对工作日的交通流进行了预测，并没有把节假日和突发性事件考虑在内。2009 年李一龙在充分研究路段容量对交通分配影响的基础上，利用动态规划思想构建了路网流量的预测模型和分配模型，为交通流的预测提供了一定的参考。

（5）组合预测模型

在研究交通流预测模型的过程中，人们通过对比发现基于单一理论的交通流预测模型，不管是基于传统数学方法的，还是基于现代科技理论的，其预测性能具有很大的局

限性，无法保证绝对好的性能指标。即随着交通流预测场景和预测时段的变化，不同预测模型的预测性能各有优劣。于是人们基于数据融合思想开发出组合交通流预测模型，该模型可以从不同的视角、基于不同的交通流预测模型收集不同的交通流信息，并通过对各方面信息的融合来实现全面地、精确地、稳定地交通流预测[6]。由于该模型可以做到对各个预测模型进行扬长避短，从而在实际应用中获得了很好的预测效果。20 世纪 60 年代，J.N.Bates 和 C.W.J.Granger 提出了组合预测理论方法，并把多种预测模型进行合理的组合取得了良好的预测效果。1998 年，Abdulhai·B 等人将遗传算法和神经网络相结合构建了遗传神经网络综合模型，提高了预测精度。2000 年，S.C.Chang 将神经网络和传统统计理论相结合，进行了相关预测研究。2008 年，谭满春等提出了基于约束卡尔曼滤波的短时交通流组合预测模型，该模型具备良好的预测效果。

综上所述，基于智能理论的预测模型具有很好的预测精度和鲁棒性。尤其是基于结构风险最小化的支持向量回归机模型，其可以很好的解决小样本和非线性问题。然而模型参数的选择对于支持向量回归机的学习精度和泛化能力起着决定性的作用，如何为模型选择合适的参数已成为有待进一步研究的关键问题。本文针对短时交通流预测中存在的问题，在充分结合支持向量回归机（SVR）结构风险最小化和模拟退火算法（SA）全局参数寻优能力的基础上，构建了 SA-SVR 短时交通流预测模型，并基于美国加州高速公路性能评测系统提供的交通流数据进行了仿真实验，验证了该模型的有效性。同时，本文基于 Hadoop 分布式处理平台开发了并行 SVR 训练算法，在保证预测精度的前提下进一步缩短了模型的训练时间，提高了短时交通流预测的实时性。

1.3 论文主要研究内容

短时交通流预测使智能交通系统实现了从“被动应对状态”到“主动控制状态”的蜕变与转换。论文在分析各种短时交通流预测模型优劣势的基础上，选择基于支持向量回归机（SVR）的短时交通流预测作为主要研究内容。首先，本文在分析参数选择对支持向量回归机（SVR）性能影响的基础上,选择模拟退火算法（SA）对 SVR 的参数行进优化，进而提出了基于 SA-SVR 的短时交通流预测模型。该模型将支持向量回归机的结构风险最小化特点和模拟退火算法的快速全局寻优优势进行了充分结合，取得了很快的收敛速度和预测精度。然后，针对大规模样本 SVR 训练时间过长这一问题，本文将 Hadoop 平台提供的 MapReduce 并行处理框架与 SVR 训练过程中支持向量的稀疏性特点结合起来，提出了 Hadoop 环境下大规模 SVR 并行训练算法，进而构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型，进一步提高了短时交通流预测的实时性。

详细研究情况如下所示：

（1） 交通流特点及预处理问题。包括交通流特点描述，交通流数据常见预处理方法介绍以及本文实验数据的预处理。

（2） 研究基于 SA-SVR 的短时交通流预测。包括参数选择对支持向量回归机的性能影响，支持向量回归机模型对比神经网络模型所具有的优势以及模拟退火算法相对于其它参数优化算法在收敛速度方面的优越性等。

（3） 研究 Hadoop 环境下基于 SA-SVR 的短时交通流预测。包括 SVR 训练过程中支持向量的稀疏性特点，大规模 SVR 并行训练算法，MapReduce 分布式处理框架的运行特点以及 Hadoop 环境下 SA-SVR 短时交通流预测模型的性能评价等。

1.4 论文组织结构

本文共 5 章内容，文中各部分之间的关系如图 1.2 所示。

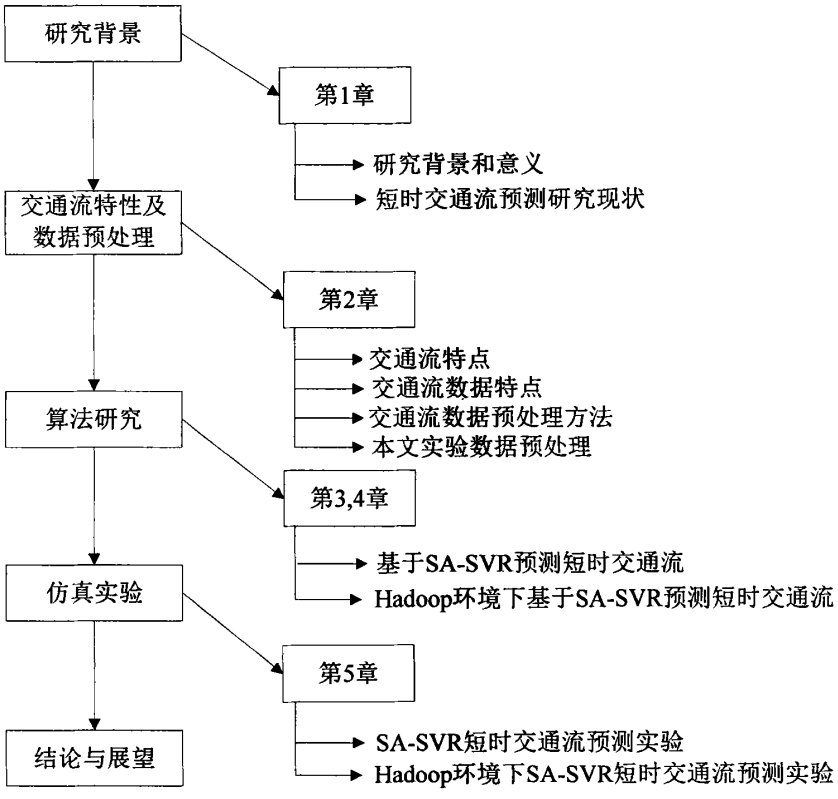

图 1.2 本文体系结构

Fig. 1.2 Thesis orgnization

第 1 章是绪论，对智能交通系统和短时交通流预测问题的研究背景和国内外发展现状进行了详细的介绍，对现有的短时交通流预测算法进了行分类描述。

第 2 章是交通流特点及数据预处理。对交通流的特点进行描述；介绍了需要对交通流数据进行预处理的原因；对常见的交通流数据预处理方法进行了归纳总结；对本文的实验数据进行了预处理。

第 3 章是基于 SA-SVR 预测短时交通流。本章在分析参数选择对支持向量回归机（SVR）性能影响的基础上,选择模拟退火算法（SA）对 SVR 的参数行进优化，将支持向量回归机结构风险最小化和模拟退火算法快速全局优化的特点充分结合，进而提出了基于 SA-SVR 的短时交通流预测模型。

第 4 章是 Hadoop 环境下基于 SA-SVR 预测短时交通流。本章充分研究了大规模样本 SVR 训练的瓶颈问题，详细描述了各种大规模 SVR 并行训练算法的优劣势，进而将 MapReduce 并行处理框架的运行特点和 SVR 训练过程中支持向量的稀疏性特点结合起来，提出了 Hadoop 环境下大规模 SVR 并行训练算法，最终构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型。

第 5 章是实验。本章对第 3、4 章提出的短时交通流预测模型分别进行了仿真实验。针对第 3 章的实验包括 BP 神经网络短时交通流预测模型与支持向量回归机短时交通流预测模型性能对比实验和基于参数选择的 SVR 短时交通流预测模型性能对比试验。针对第 4 章的实验为 Hadoop 环境下 SA-SVR 短时交通流预测模型与单机 SA-SVR 短时交通流预测模型性能对比试验。

最后是总结。总结了本文的研究内容、存在的不足及对未来研究工作的展望。

2 交通流特性及数据预处理

2.1 交通流特性

交通流是一个复杂的、与人相关的、非线性的多维非平稳系统。随着研究的深入，人们发现交通流除了具备一些流体力学特性之外还具备周期性、随机性和时空相关性等特点，天气情况、司机状态、突发事件等因素都会对交通流产生影响[7]。综合各种因素，交通流的几个主要特性归纳如下：

（1） 交通流的力学特性

研究表明交通流具有流体特性的本质，因此研究人员可以利用流体力学理论建立模型来描述宏观交通流。该模型使科学化的交通流研究成为可能，为交通流预测奠定了理论基础。

（2） 交通流的周期性

日常生活中人们的出行带有一定的规律性，因此交通流受此影响也表现出一定的周期性。例如在每个工作日的上班时间和下班时间，交通流会对应出现早高峰和晚高峰，其它时间则相对平缓。可见，以天为单位交通流变化呈现出明显的周期性。虽然工作日和周末的交通流特性存在明显的区别，但是这从另外一个角度说明交通流以周为单位也呈现出一定的周期性。交通流的周期性也为交通流的可预测性奠定了基础。

（3） 交通流的随机性

由于人们的日常活动带有一定的随机性，出行目的地不同，选择的路线也不同，加之车辆的行驶要收到天气、交通事故等客观环境因素的影响，因此交通流呈现出一定的不确定性。随着预测时间的缩短，其不确定性和随机性也随之增强。

（4） 交通流的时空相关性

交通流会随着时间的变化而变化，与时间维度有着明显的相关性。另外，道路之间的相互连通，决定了多个路段的交通流之间存在着必然的联系。各个相邻路口的交通流彼此相互作用，表现出很强的空间相关性。而且，空间相关的程度也会根据时段的不同而不同，这使交通流具备了显著的时空相关性特点。

（5） 内在约定性

由于车辆的行驶总要遵循一定的交通规则，而且有着追求安全、快速和畅通的共同目标，因此各个车辆可以彼此约束、相互合作，从而在宏观上表现出良好的有序性。这样交通流也就成为了一个具备内在约定性的自组织系统。

综上所述，交通路流并不是一个完全不确定的随机系统，而是一个以内在约束性为基础，同时具备力学特性、周期性和时空相关性的复杂非线性系统。交通流的这些特点使其表现出很强的可预测性，研究人员可以通过构建合理的数学模型来取得良好的预测效果。

2.2 交通流数据特点

道路交通系统是一个时变的、非结构化的复杂大系统，道路交通流数据有以下特点：

（1） 数据种类众多。智能交通系统中的数据采集终端各种各样，主要包括：线圈检测器、红外检测器、超声波检测器、车辆 GPS 等[8]。每种检测器都按照各自定义好的数据格式进行数据采集，例如：线圈检测器的检测数据项包括线圈 ID、时间、速度、车流向、车间距、交通流量、占有率等；车辆 GPS 的检测数据项包括 ID、GPS 时间、GPS 经度、GPS 纬度、GPS 速度、GPS 方向等；视频检测器的检测数据项包括 ID、速度、占有率、车类车色、车流向、车辆行驶轨迹、车头时距、通过时间、交通流密度等。而在实际应用中，不同的应用程序需要不同的数据项，我们往往需要根据应用程序的实际需求来对原始交通数据进行数据抽取和属性约减，进而得到对我们有用的数据。

（2） 数据量大。智能交通系统依靠大量的传感器采集交通数据，这些传感器分布在城市的不同位置，并以固定的频率将其采集到的交通信息发送给后台的数据中心。传感器上传的每一组数据都包含时间、位置以及其它数据项，这些元数据都很容易形成海量数据。因此，虽然通过这些数据我们可以准确掌握城市的交通状况，但是交通数据也随着时间增长而爆炸式地增加，以至于面向车联网的海量数据处理技术已逐渐成为智能交通系统的核心与关键。

（3） 数据量分布不均。道路交通与人的活动息息相关，人们出行时间、出行路线和出行方式等都带有一定的随机性，这使得交通流特性也呈现出不确定性、时空相关性等特点。从而数据采集终端采集到的数据量根据时间段和路段的不同而呈现出巨大的变化，分布非常不均匀。

综上所述，如果我们不对交通流数据进行预处理，它们很难被应用程序使用。对于交通流数据而言，我们根据其预处理时间长短的不同可以将其分为历史交通流预处理和实时交通流预处理，历史交通流预处理往往对预处理后的数据质量要求比较高，对处理过程中消耗时间的长短不予考虑；而实时交通流预处理则要求在相对短的时间内完成预处理工作，对预处理后的数据质量要求相对较低。交通流数据预处理的方法很多，结合实际需求选择合适的数据预处理方法往往有助于提高交通流预测模型的预测精度和预测效率。

2.3 交通流数据预处理方法

高质量的交通流数据是交通流预测的前提和关键。数据质量能够对交通流预测模型的预测精度产生直接的影响，因此在交通流预测之前，我们需要对交通流数据执行相关预处理操作。交通流数据预处理方法主要包括：数据集成、数据清洗、数据转换以及数据规约等[9]。

（1） 数据集成：交通流数据来自于不同的采集终端，具备不同的数据格式和数据特点。为了给应用程序提供全面的数据支持，我们通常把原始交通流数据在逻辑上或物理上进行有机地集中，将之称为数据集成。由于智能交通系统中传感器的种类繁多，而且返回的数据在数据格式上有很大的不同。为了给交通流预测应用提供有效地交通流数据，我们需要从不同的数据中筛选出对我们有用的数据并通过数据融合方法生成程序可以使用的基础数据。数据集成可以生成有效的数据集来全面的反应交通系统的运行状况，进而为交通流预测应用和研究决策人员提供有效的数据支持。

（2） 数据清洗：数据终端在传输数据的过程中会受到各种各样的干扰，这使数据中心接收到的数据偶尔出现数值失真、数值缺失、数值重复等现象。通常我们对原始交通流数据进行缺失值修补、噪声光滑和删除离群点等操作，将之称为数据清洗。通过数据清洗我们可以达到格式标准化、异常数据清除、错误纠正的主要目标。

（3） 数据转换：原始的交通流数据不能直接用于数据挖掘，我们通常对其执行平滑、聚集、规范化等操作，从而将数据转换成适用于数据挖掘的形式。通常，交通流应用对交通流数据集的数据格式都有固定的要求，因此我们要按照一定的格式规范对交通流数据进行处理。例如：短时交通流预测应用一般是以时间序列的方式进行预测，需要用某个路段前几个时间段的交通流量值来预测下一个时间段的交通流量，所以数据集应该提前被转化为时间序列的格式。

（4） 数据规约：在保证数据完整性的前提下，去除数据冗余，缩小数据规模。常用数据规约方法有数据压缩、数值压缩和离散化等。有些交通流预测应用要求数据的大小必须在规定的范围之内，比如神经网络算法要求数据大小位于[-1,1]之间，支持向量机也推荐用户将数据大小限制在[0,1]或[-1,1]之间，此时数据规约具有极为重要的实际意义。

智能交通系统中数据采集终端采集到的数据往往存在丢失、异常、噪声污染等问题，而科学的数据预处理方法在保证数据完整性的前提下可以有效地改善数据的质量。预处理生成的基础数据集可以更加全面地反应交通系统运行状况，为交通研究、交通应用和交通决策等提供可靠的数据支持。

2.4 论文实验数据准备

2.4.1 实测数据描述

本文使用的数据来源于美国加利福尼亚高速公路性能评估系统(PeMS)，采集时间为2013年4月1日到2014年03月31日，采样时间间隔为5分钟。数据格式如表2.1所示，各数据项描述如表2.2所示。

表2.1 实测交通流数据格式

Tab. 2.1 Traffic data format

Timestamp	Station	District	Freeway	Direction	Station Type	Total Flow	Avg Occupancy	Avg Speed
01/01/2014 00:00:00	1107608	11	15	N	FR	0	0	0
01/01/2014 00:00:00	1108148	11	15	S	ML	78	0.0099	69.9
01/01/2014 00:00:00	1108285	11	94	W	ML	37	0.0086	66.8
01/01/2014 00:00:00	1108287	11	94	W	ML	15	0.003	66

表2.2 数据项描述

Tab. 2.2 Data item description

序号	数据项	描述
1	Timestamp	时间戳
2	Station	检测器标识
3	District	检测器所在区域
4	Freeway	快速路编号
5	Direction	行驶方向（N|S|E|W）
6	Station Type	FR=下坡路；OR=上坡路；ML=主干道
7	Total Flow	5分钟内经过某传感器覆盖的所有车道的车辆数总和
8	Avg Occupancy	5分钟内某传感器覆盖的的所有车道的平均占有率，值在0-1之间
9	Avg Speed	5分钟内经过某传感器的所有车辆的车速平均值

由于PeMS已经对传感器采集到的数据进行了一些修复和平稳化处理，所以本文省去了部分预处理工作。然而，由表2.1可知，交通流原始数据项中包含了许多短时交通

流预测中用不到的数据属性，所以后续要对上述交通流数据进行属性约减，筛选对交通流预测有用的数据。同时，我们还要通过一些处理方式尽量还原数据的真实性。

2.4.2 实验数据预处理

为了使试验样本能够充分反应交通流系统的特点，需要对交通流数据进行预处理，方法包括：交通流数据的抽取、交通流数据时序相空间重构和交通流数据归一化等。

（1） 交通流数据抽取

我们将 1108148 号传感器的监测数据从元数据中抽取出来，从 2013 年 4 月 1 日到 2014 年 3 月 30 日，时间跨度为一年，采样间隔为 5 分钟，共计 105120 组实验数据，其中 2014 年 1 月 6 日到 2014 年 1 月 19 日（半个月）的交通流数据如图 2.1 所示。仔细观察后可以发现，交通流数据分别以天和以周为单位呈现很强的周期性，但是工作日和周末的交通流数据有明显的不同。所以采用周五（2014 年 3 月 28 日）、周六（2014 年 3 月 29 日）和周日（2014 年 3 月 30 日）的交通流数据作为预测样本，分别用于第 5 章中的 3 种对比试验，从而进一步验证交通流预测模型的有效性。同时，因为每天 0:00-4:00 的交通流非常稀疏，不存在拥堵，所以该时段内的交通流数据不计入预测样本，即实验中我们只对实验日 4:05-23：55 的交通流进行预测。训练样本则根据每个实验的需要从剩余的 104256 组数据中获取。

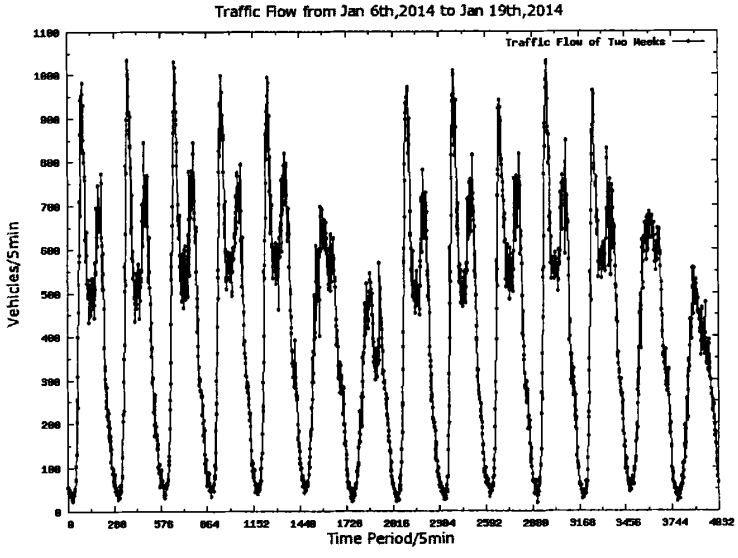

图 2.1 2014 年 1 月 6 日-2014 年 1 月 19 日交通流数据

Fig. 2.1 Traffic Data from 2014 Jan 6th to 2014 Jan 19th

（2） 交通流数据时序相空间重构

某一时刻的交通流数据总是与前几个时段的数据密切相关，这是交通流预测可以实现的理论前提。为了使模型能够充分体现交通流系统的数据特点，我们对原始样本集进行时序相空间重构。在相空间重构之前需要确定嵌入维维数 m 和延迟时间 T。我们令 m=5, 我们令 T 等于序列检测间隔时间（5 分钟）构造滑动时间窗口 $x_t=(x_{t-1},x_{t-2},\cdots x_{t-m})$[10]，得到新的样本集：

$$X=\begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ x_2 & x_3 & \cdots & x_{m+1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n-m} & x_{n-m+1} & \cdots & x_{n-1} \end{bmatrix}, Y=\begin{bmatrix} x_{m+1} \\ x_{m+2} \\ \cdots \\ x_n \end{bmatrix}$$

对于训练样本集，由本节（1）可知 n=104256, 重构后可用的训练样本数为 n-m=104251，预测样本采用同样的处理方法，每组样本数为 288-5=283。

（3） 交通流数据归一化处理

为了降低交通流预测的计算量、提高交通流预测的精度，我们需要对试验样本进行归一化处理。

样本数据归一化处理选用函数为：

$$\bar{x}=(x_i-x_{\min})/(x_{\max}-x_{\min}) \tag{2.1}$$

算法执行完后，数据更新公式为：

$$x_i=x_{\min}+\bar{x}_i(x_{\max}-x_{\min}) \tag{2.2}$$

其中，x_i 为原交通流流量数值，$\bar{x}_i$ 为变换后的交通流流量数值，x_{min} 为样本数据中的最小流量值，x_{max} 为样本数据中的最大流量值[11]。

2.5 本章小结

本章简要介绍了交通流所具备的力学特性、周期性、随机性、时空相关性和内在约束性，进而对交通流的可预测性做出了定性总结。然后，在分析原始交通流数据特点的基础上，进一步阐述了对其进行数据预处理的必要性，并给出了数据预处理的常用方法。最后，结合本文实验的具体需求，使用上述数据预处理方法对 PeMS 提供的真实交通流数据进行了数据抽取、时序相空间重构和数据规约等操作，为本文第 5 章提供了必要的实验数据。

3 基于 SA-SVR 预测短时交通流

3.1 支持向量机

支持向量机是一种基于统计学习理论的机器学习方法，90 年代中期由 Vapnik 等人提出。它基于结构风险最小化原则来提高学习算法的泛化能力，从而能够在统计样本量相对较少的情况下仍能获得良好的统计学习效果[12]。支持向量机在学习过程中不易出现过拟合，可以避免特征过多造成的维数灾难，可以收敛于全局最优解并且具备很强的抗干扰能力。

3.1.1 分类问题描述

已知训练集 $T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$，其中 $x_i\in X=\mathrm{R}^n, y_i\in Y=\{1,-1\}, i=1,\cdots,l$，寻找 $X\in R^n$ 上的实值函数 $g(x)$，以便用决策函数 $f(x)=\mathrm{sgn}(g(x))$ 推断任意值 x 相对应的 y 值。通常来讲，$x_i\in X=\mathrm{R}^n$ 是输入指标向量，或称模式，其分量称为特征或属性；$y_i\in Y=\{1,-1\}$ 是输出；$i=1,\cdots,l$ 这 l 个样本点组成的集合称为样本集。如果样本集为线性可分我们将该问题成为线性可分分类问题，反之称为线性不可分分类问题[13]。即如果 $g(x)$ 为线性函数我们称该问题为线性分类问题，反之称为非线性分类问题。由此可见，求解分类问题实际上就是找到一个规则，使其能够把 R^n 上的点分成或近似分成两部分。

3.1.2 线性可分支持向量分类机

（1）线性可分分类问题的凸优化模型

通常，如果样本集为线性可分，基于最大间隔原则我们可以把上述分类问转化为一个凸优化问题进行求解，详细描述如下。

① 已知训练集 $T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$，其中 $x_i\in X=\mathrm{R}^n, y_i\in Y=\{1,-1\}, i=1,\cdots,l$；

②对变量 ω 和 b 构造并求解如下最优化问题：

$$\min_{\omega,b}\ \frac{1}{2}\|\omega\|^2 \tag{3.1}$$

$$s.t.\ \ y_i((\omega\cdot x_i)+b)\geq 1, i=1,\cdots,l \tag{3.2}$$

求得最优解 ω^* 和 b^*；

③ 构造划分超平面 $(\omega^*\cdot x)+b^*=0$，进而得到决策函数 $f(x)=\mathrm{sgn}((\omega^*\cdot x)+b^*)$ ；

（2） 线性可分支持向量机算法描述

为了求解上述最优化问题，我们通常基于对偶原理以引入拉格朗日乘子的方式将其转化为一个二次规划问题来解答，详细算法描述如下。

① 已知训练集$T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$,其中$x_i\in X=\mathrm{R}^n, y_i\in Y=\{1,-1\}, i=1,\cdots,l$；

② 构造并求解最优化问题：

$$\min_{\alpha}\ \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_iy_j\alpha_i\alpha_j(x_i\cdot x_j)-\sum_{j=1}^{l}\alpha_j \tag{3.3}$$

$$s.t.\ \sum_{i=1}^{l}y_i\alpha_i=0 \tag{3.4}$$

$$\alpha_i\geq 0, i=1,\cdots,l. \tag{3.5}$$

得到最优解$\alpha^*=(\alpha^*_1,\cdots\alpha^*_l)^T$；

③ 计算$\omega^*=\sum_{i=1}^{l}y_i\alpha_i^*x_i$,选择$\alpha^*$的一个的正分量$\alpha^*_j$并使其小于$C$，然后计算$b^*=y_i-\sum_{i=1}^{l}y_i\alpha_i^*K(x_i,x_j)$；

④ 构造划分超平面$(\omega^*\cdot x)+b^*=0$，进而求出决策函数$f(x)=\mathrm{sgn}((\omega^*\cdot x)+b^*)$；

3.1.3 线性支持向量分类机

（1）线性不可分分类问题的凸优化模型

考虑样本集可能存在线性不可分的情况，我们引入松弛向量$\xi=(\xi_1,\ldots\xi_{2l})^T$来软化约束条件，体现训练集被划错的程度，同时引入惩罚系数 C 来避免ξ_i取值过大，基于扩展的间隔最大原则将 3.1.1 中的分类问题转化为以下凸优化问题来求解。

①已知训练集$T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$,其中$x_i\in X=\mathrm{R}^n, y_i\in Y=\{1,-1\}, i=1,\cdots,l$；

②确定合理的惩罚参数$C>0$，对变量ω、b和$\xi=(\xi_1,\ldots\xi_{2l})^T$构造并求解如下最优化问题：

$$\min_{\omega,b,\xi}\ \frac{1}{2}\|\omega\|^2+C\sum_{i=1}^{l}\xi_i \tag{3.6}$$

$$s.t.\ \ y_i((\omega\cdot x_i)+b)+\xi_i\geq 1, i=1,\cdots,l \tag{3.7}$$

$$\xi_i \geq 0, i = 1, \cdots, l \tag{3.8}$$

求得最优解 ω^*、b^* 和 ξ^*；

③ 构造划分超平面 $(\omega^* \cdot x) + b^* = 0$，进而得到决策函数 $f(x) = \mathrm{sgn}((\omega^* \cdot x) + b^*)$；

（2）线性支持向量分类机

基于对偶原理，我们引入拉格朗日乘子进而将上述凸优化问题转化为二次规划问题来求解，详细算法描述如下。

① 已知训练集 $T = \{(x_i \cdot y_j), \cdots, (x_l \cdot y_l)\} \in (X \times Y)^l$，其中 $x_i \in X = \mathrm{R}^n$，$y_i \in Y = \{1, -1\}$，$i = 1, \cdots, l$；

② 选择合适的惩罚参数 $C > 0$，构造并求解最优化问题：

$$\min_{\alpha} \ \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) - \sum_{j=1}^{l} \alpha_j \tag{3.9}$$

$$s.t. \ \sum_{i=1}^{l} y_i \alpha_i = 0 \tag{3.10}$$

$$0 \leq \alpha_i \leq C, i = 1, \cdots, l \tag{3.11}$$

得到最优解 $\alpha^* = (\alpha^*_1, \cdots \alpha^*_l)^T$；

③ 计算 $\omega^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$，选择 α^* 的一个小于 C 的正分量 α^*_j，并据此计算 $b^* = y_i - \sum_{i=1}^{l} y_i \alpha_i^* K(x_i, x_j)$；

④ 构造分化超平面 $(\omega^* \cdot x) + b^* = 0$，由此求得决策函数 $f(x) = \mathrm{sgn}((\omega^* \cdot x) + b^*)$；

3.1.4 支持向量分类机

对于非线性不可分分类问题，我们将样本集映射到一个高维的特征空间，即 $x \to \Phi(x)$，从而将输入空间的非线性问题转换为特征空间的线性问题。然后，根据 3.1.2 中的线性不可分分类问题求解方案在特征空间中构造最优分类超平面。同时，求解过程中，通过引入核函数 $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ 巧妙地将高维特征空间中的内积运算转化为原空间的内积运算，从而避免了维数灾难。详细算法如下所示。

①已知训练集 $T = \{(x_i \cdot y_j), \cdots, (x_l \cdot y_l)\} \in (X \times Y)^l$，其中 $x_i \in X = \mathrm{R}^n$，$y_i \in Y = \{1, -1\}$，$i = 1, \cdots, l$；

②确定合理的惩罚参数$C>0$和适当的核函数$K(x,x')$，构造求解如下最优化问题：

$$\min_{\alpha} \ \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_i y_j \alpha_i \alpha_j K(x_i,x_j)+\sum_{j=1}^{l}\alpha_j \tag{3.12}$$

$$s.t. \ \sum_{i=1}^{l}y_i\alpha_i=0 \tag{3.13}$$

$$0\le\alpha_i\le C, i=1,2,\cdots l \tag{3.14}$$

得到最优解$\alpha^*=(\alpha^*_1,\cdots\alpha^*_l)^T$；

③选择α^*的一个正分量α^*_j使其小于C，进而计算$b^*=y_i-\sum_{i=1}^{l}y_i\alpha_i^*K(x_i,x_j)$；

④ 构造决策函数$f(x)=\mathrm{sgn}(\sum_{i=1}^{l}(y_i\alpha_i^*K(x_i,x)+b^*)$；

3.1.5 支持向量回归机

（1） 回归问题描述

已知训练集$T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$，其中$x_i\in X=\mathrm{R}^n, y_i\in Y=R, i=1,\cdots,l$，寻找$R^n$上的实值函数$f(x)$，使用$y=f(x)$来推断任意模式$x$所对应的$y$值。其中，$x_i\in X=\mathrm{R}^n$为输入指标向量，其分量称之为属性；$y_i\in Y=R$为输出；$i=1,\cdots,l$这$l$个样本点组成的集合称为样本集。当$f(x)$为线性函数时，我们称上述问题为线性回归问题，反之，我们称之为非线性回归问题[13]。通常，我们将非线性回归问题转化为线性回归问题来解决。本文采用基于间隔最大原则构建的ε-支持向量回归机来解决上述回归问题，具体算法如下。

（2） 支持向量回归机算法

支持向量回归机最终是由支持向量分类机来实现的，详情如下。已知一回归问题，其训练集为$T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$，其中$x_i\in X=\mathrm{R}^n, y_i\in Y=R, i=1,\cdots,l$。令$(x_{l+i}^T, y_{l+i}-\varepsilon)^T=(x_i^T, y_i-\varepsilon)^T, i=1,\cdots,l$，我们可以得到新的训练集$\{((x_1^T,y_1+\varepsilon)^T,1),\cdots,((x_l^T,y_l+\varepsilon)^T,1),((x_{l+1}^T,y_{l+1}-\varepsilon)^T,-1),\cdots,((x_{2l}^T,y_{2l}-\varepsilon)^T,-1)\}$，从而将原训练集的回归问题转化为新训练集的二分类问题。基于新的训练集，使用 3.1.3 中描述的支持向量分类计算法求得最优分类超平面，进而得到回归决策函数，详情如下。

① 已知训练集为$T=\{(x_i\cdot y_j),\cdots,(x_l\cdot y_l)\}\in(X\times Y)^l$，其中$x_i\in X=\mathrm{R}^n, y_i\in Y=R, i=1,\cdots,l$；

② 选择合适的不敏感损失参数ε和惩罚因子C，确定适当的核$K(x,x')$；

③ 构造并求解如下最优化问题：

$$\min_{\alpha^{(*)}\in R^{2l}} \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)K(x_i,x_j)+\varepsilon\sum_{i=1}^{l}(\alpha_i^*+\alpha_i)-\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \quad (3.15)$$

$$s.t.\ \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0 \quad (3.16)$$

$$0\le\alpha_i,\alpha_i^*\le\frac{C}{l},i=1,2,\cdots l. \quad (3.17)$$

得到最优解 $\bar{\alpha}=(\bar{\alpha}_1,\bar{\alpha}_1^*,\cdots,\bar{\alpha}_l,\bar{\alpha}_l^*)^T$；

④ 决策函数 $f(x)=\sum_{i=1}^{l}(\bar{\alpha}_i^*-\bar{\alpha}_i)K(x_i,x_j)+\bar{b}$，其中 $\bar{b}=y_j-\sum_{i=1}^{l}(\bar{\alpha}_i^*-\bar{\alpha}_i)(x_i\cdot x_j)+\varepsilon$，$\bar{\alpha}_j\in(0,\frac{C}{l})$；

3.2 基于模拟退火算法优化 SVR 参数

3.2.1 参数选择对 SVR 性能的影响及研究现状

支持向量回归机模型参数包括惩罚因子 C、不敏感损失参数 ε 和核函数参数 σ。其中，惩罚因子 C 越大，模型的学习精度就越高，但泛化能力变差，反之，误差就越大，泛化能力随之增强。不敏感损失系数 ε 越大，支持向量的个数就越少，模型学习精度越差，容易造成欠拟合。反之，容易造成过拟合。核宽度 σ 决定了学习样本的输入范围，，σ 取值越大，样本输入范围越大，反之，样本输入范围越小[14]。因此，为支持向量回归机选择合适的参数是使其发挥最优性能的关键。

目前，优化 SVR 参数的方法主要包括试凑法、网格法、梯度下降法、启发式算法等。试凑法主要依靠专业人员根据自己的经验调整参数，带有一定的盲目性和随机性，不能保证达到全局最优。网格算法以固定步长对参数空间的所有组合进行穷举实验，对于参数较少的情况该方法取得了很好的效果，但是在实际应用中该算法效率不高。梯度下降算法比较简单，但是具有收敛速度慢、对初始值比较敏感和容易陷入局部极值等缺点[15]。启发式算法可以通过多点并行搜索来使自己达到很高的寻优效率和寻优精度。作为一种启发式算法，模拟退火算法是一种新型的随机搜索算法，它适合于解决大规模组合优化问题，具有描述简单、运行效率高和较少受到初始条件约束等优点。因此，我们选择模拟退火算法来进行 SVR 参数优化，进而构建 SA-SVR 短时交通流预测模型，以增强短时交通流预测的实时性。

3.2.2 基于模拟退火的 SVR 参数优化算法

模拟退火算法来源于固体退火原理，于 1953 年由 N.Metropolis 等人提出。在处理固体金属的过程中，我们将其加温至充分高，使之内部粒子无序运动并处在相对高能的液化状态，再让其缓慢降温（即退火），其最终总能达到能量最低的晶体状态。此时分子排列也总会趋于有序，与初始的无序运动状态无关。反之，如果急速降温（即淬火），金属则只能达到一种较高内能的多晶体状态或非结晶状态，不能达到能量最低点[16]。与此类似，我们得到求解多变量函数全局极值的模拟退火算法，如图 3.1 所示。

(1) 初始化：确定解空间 T，最大迭代次数 K_{max}，初始解 ξ，T 值的迭代次数 L；

(2) 对 k=1,……,L 做第(3)至第 6 步；

(3) 产生新解 ξ_{new}；

(4) 计算增量 $\Delta E = C(\xi_{new})-C(\xi)$，其中 $C(\xi)$为评价函数；

(5) 若 $\Delta E > 0$ 则接受 ξ_{new} 为当前解，否则以概率 $\exp(-K/K_{max})$ 接受 ξ_{new} 为当前解；

(6) 如果满足终止条件则输出当前解作为最优解，结束程序；

(7) T 逐渐减少，且 T > 0，然后转第 2 步；

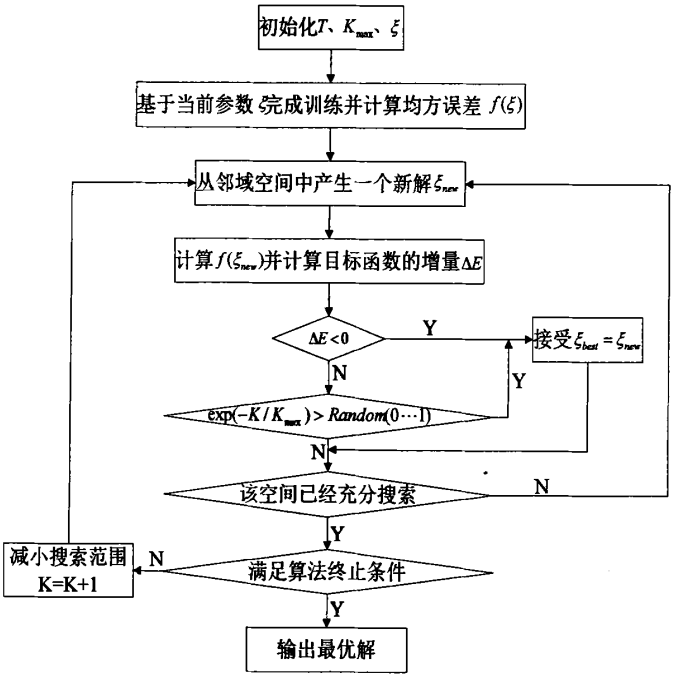

图 3.1 基于模拟退火算法的 SVR 参数优化算法

Fig. 3.1 Algorithm for optimizing SVR parameters based on annealing

基于上述模拟退火算法，我们来对 SVR 的参数：惩罚因子C、不敏损失系数ε、核函数参数σ进行优化。在此过程中，基于交通流训练样本，我们使用当前参数以交叉验证的方式对支持向量回归机进行训练，并将均方误差作为评价函数，当迭代次数达到最大值或均方误差小于某一数值时算法停止，具体流程如图 3.1 所示。

3.3 SA-SVR 短时交通流预测模型

基于 3.2.2 提出的 SVR 参数优化算法，结合短时交通流预测的数据特点，我们构建了 SA-SVR 短时交通流预测模型。该模型通过对交通流数据的重构和规约来获得训练样本集，使用模拟退火算法获得最优参数，进而基于该参数和训练样本集对 SVR 进行训练，最终得到用于短时交通流的预测的决策函数[17]。SA-SVR 实现短时交通流预测的具体步骤如图 3.2 所示。

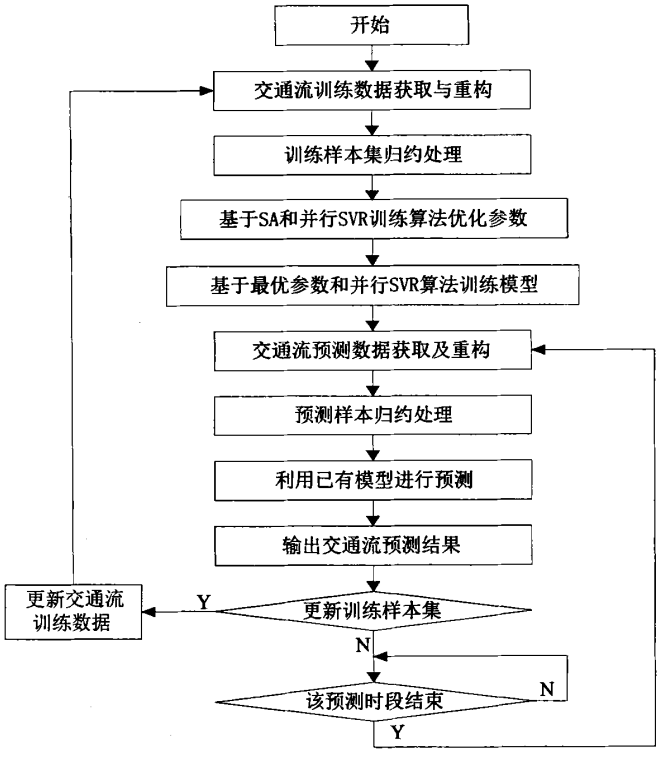

图 3.2 SA-SVR 短时交通流预测模型

Fig. 3.2 Short-term traffic flow forecasting model based on SA-SVR

（1） 获取交通流数据并进行时序相空间重构；
（2） 对训练样本集进行归约化处理；
（3） 基于 SA 算法优化 SVR 参数；
（4） 基于最优参数和训练样本集训练模型；
（5） 交通流预测数据获取及样本重构；
（6） 对预测样本集进行归约化处理；
（7） 利用已有模型进行预测；
（8） 输出交通流预测结果；
（9） 如果需要更新样本集转到（1）；
（10）如果该预测时段结束转到（5），否则，转到（10）；

3.4 本章小结

支持向量机是一种基于统计学习理论的机器学习方法，90 年代中期由 Vapnik 等人提出。它基于结构风险最小化原则来提高学习算法的泛化能力，从而可以在样本较少的情况下解决复杂非线性系统的拟合问题，在短时交通流量预测中表现了出优越的性能。本章详细介绍了支持向量机的基本理论，阐述了支持向量分类机从线性可分到线性不可分，再到非线性不可分的发展脉络，说明了支持向量分类及和支持向量回归机的本质联系。然后，详细分析了参数选择对支持向量回归机性能的影响以及支持向量回归机参数选择的研究现状。在对比各种参数优化算法优劣势的基础上，选择使用模拟退火算法对 SVR 进行参数优化，提出了 SA-SVR 算法。最后，基于该算法和第 2 章提供的交通流数据构建了基于 SA-SVR 的短时交通流预测模型并给出了详细的预测步骤。

4 Hadoop 环境下基于 SA-SVR 预测短时交通流

本文第 3 章已经详细介绍了基于单机的 SA-SVR 短时交通流预测算法，其具备良好的预测精度。然而，实际应用中面对大规模的训练样本集，该算法往往因训练速度慢、耗时太长而无法满足实时交通流预测的需要。本文将在研究大规模 SVR 训练算法的基础上，结合 MapReduce 分布式处理框架的运行机制，提出了 Hadoop 环境下大规模 SVR 训练算法，并构建了基于该算法的短时交通流预测模型。实验表明，该算法在保证精度的前提下具有很高的处理速度，满足大规模数据环境下短时交通流预测的实时性要求。

4.1 云计算平台 Hadoop

随着经济的高速发展，现代社会的信息量爆炸式地增长并衍生出传统技术难以处理的数据规模，最终将人类社会带入了大数据时代。据统计，到 2012 年为止，全球每年产生的数据量已经跃升至 ZB(1ZB=2^{40}GB)级别。根据国际数据公司 IDC 的研究结果，2009 年全球产生的数据量为 0.49ZB，这一数值在 2010 年增长至 1.2ZB，在 2011 年的更是达到 1.82ZB。这意味着全球平均每人每年产生的数据达到 200G 以上。而截止到 2012 年人类所生产的所有印刷品的数据量也才仅仅是 200PB，全人类历史上人们说过的所有话的数据量还不到 5EB[18]。IBM 研究表明，人们过去两年所产生的数据量在人类全部历史上产生的数据中所占的比例达到 90%。预计到 2020 年，全世界所产生的数据量将是现有数据量的 44 倍。其中，将有超过 1/3 的内容需要存储在云平台上或需要在云平台的帮助下进行处理[19]。如何高效地对这些数据进行存储、分析和处理，以获取更有价值的信息已经成为当今时代的巨大挑战。Hadoop 作为一个优秀开源分布式处理系统，采用分布式文件系统（HDFS）来提高数据存储容量、提高数据读写速度，并采用分布式处理框架（MapReduce）来高效处理分布式系统上的数据，具备强大的分布式数据存储能力和处理能力，在众多行业中得到广泛的应用。

4.1.1 Hadoop 简介

Hadoop 云计算平台由三大组件构成，它们分别是：HDFS 分布式文件系统、MapReduce 分布式处理框架以及 Hbase 分布式数据库。这三项主要技术正好与 Google 的 GFS、MapReduce 与 BigTable 三大云计算内核技术相类似[20]。所以，在 Hadoop 的帮助下研究人员可以相对容易地构建与 Google 相似的云计算系统。

Hadoop 平台主要提供分布式存储和分布式处理两大功能，这两种功能的运行机制均采用主/从结构。Hadoop 集群在运行过程中往往需要启动一系列的后台进程，不同的

进程扮演着不同的角色，这些角色主要包括：NameNode、DataNode、Secondary NameNode、JobTracker 和 Task Tracker 等[20]。对于 Master 节点，运行在其上的进程有 NameNode、Secondary NameNode 和 JobTracker，而对于 Slave 节点，每个节点上分别运行一个 DataNode 和 TaskTracker，这样某个 Slave 节点上运行的数据处理程序可以选择直接处理本地的数据，避免了远程处理带来的通信开销，具体原理如图 4.1 所示。

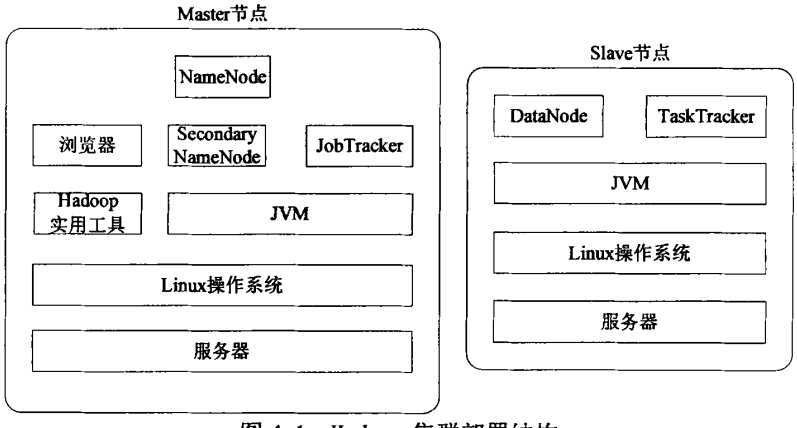

图 4.1 Hadoop 集群部署结构

Fig. 4.1 Deployment structure of Hadoop cluster

4.1.2 HDFS 简介

作为 Hadoop 云计算平台的核心组件之一，HDFS（Hadoop Distributed File System）为分布式计算提供了最基本的数据存储和数据管理功能。该分布式文件系统具备高容错性、高扩展性和高吞吐率等特点，可以在出现节点故障的情况下实现对海量数据的可靠存储。另外，它还可以与 MapReduce 分布式计算框架很好的结合，在此基础上以极高的吞吐量为分布式处理程序提供数据访问[21]。

（1）HDFS 的基本架构

HDFS 采用主/从结构，如图 4.2 所示。从用户的角度来看，该分布式文件系统与传统的文件系统没有区别。用户可以可基于文件路径执行创建、读写和删除等操作。然而，由于该文件系统的分布式特性，人们往往将其构建在一组特定的节点之上，这些节点由一个 NameNode 和若干个 DataNode 组成。

NameNode 是 HDFS 的关键进程，负责管理文件系统的元数据并对外部客户机的访问请求进行处理。通常，该进程运行在一个单独的服务器之上。在分布式文件系统中，大文件通常被划分为固定大小的文件块并被分别存储在相应的数据节点之上。在此过程

中，NameNode 负责对大文件进行切分并生成小文件块与数据节点之间的映射规则，然后将最终的映射信息生成元数据存储在本地，为客户机提供索引支持。当收到客户机的某种文件请求信息时，NameNode 会将其请求文件块儿的标识以及该文件块儿第一副本所在 DataNode 的 IP 地址作为相应信息发送给客户机。客户机通过得到的地址信息直接与 DataNode 进行数据交互，实现相应的文件系统访问操作。由此可见，NameNode 并不负责真正的 I/O 文件操作，只负责将包含文件块儿和 DataNode 映射关系的元数据发送给客户机。

DataNode 负责最实际的数据块进行存储，通常运行在单独的从节点之上。一个完整的 Hadoop 集群由一个 NameNode 和若干个 DataNode 组成。其中，DataNode 负责对来自客户机的读写请求进行处理，并同时对 NameNode 发出的创建、删除、复制等文件操作命令进行响应。为了实现 NameNode 对所有 DataNode 的状态监控，Datanode 会以固定的时间间隔向 NameNode 发送心跳消息，该消息由一系列的文件块儿报告组成。NameNode 可以根据该消息来对文件块儿的映射关系以及其他文件系统元数据进行验证。如果在规定的时间内没有收到心跳消息，NameNode 将采取修复措施将相应 DataNode 上丢失的文件块儿进行重新复制。

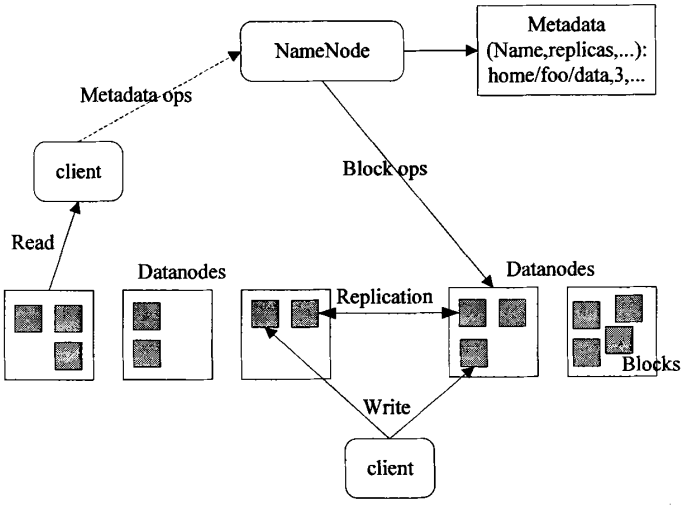

图 4.2　Hadoop 集群基本结构

Fig. 4.2　Structure of Hadoop cluster

（2）HDFS 的主要特点

① 高吞吐量访问

HDFS 将大文件的数据块分布在不同的数据节点上进行存储。当客户机访问某个数据块时，NameNode 将会在包含该数据块儿副本的所有空闲数据节点中选择网络距离最近并且访问负载最小一个与客户机进行数据交互。可见，存有某个数据块儿副本的所有数据节点可以同时为若干客户机提供该数据块儿访问操作。这使得 HDFS 某个单数据块儿访问速度达到传统单机存储方案的数倍。由于 HDFS 对大文件进行了分布式存储，因此当分布式处理应用访问某个大文件时完全可以以极高的带宽从多个服务器上面并发读入，从而可以把大文件的访问负载分摊到多个数据节点之上。这使得 HDFS 的吞吐量相对于单硬盘或单服务器的存储系统有了数十倍甚至数百倍的提高。

② 无缝容量扩充

HDFS 将大文件的各个文件块儿与相应数据节点的映射关系存储在 NameNode 之上。当需要对系统的容量进行扩充时，我们只需要将运行 DataNode 进程的数据节点加入到集群网络中，文件系统会通过相应机制进行发现扫描并将新发现的硬件自动匹配到存储阵列之中，最终在无需任何系统宕机和人为干预的情况下实现存储容量的实时线性增加。新的数据节点加入之后，文件系统的分布算法会将相应的数据块儿搬迁至该节点，整个文件系统的重新分布也不需要人工干预。

③ 高度容错

HDFS 在设计之初就将诸如服务器故障、网络故障、存储故障之类的情况考虑在内，因此该文件系统的构建将这些故障视为常态而非异常，这使得该系统天生就具备保证数据可靠性的能力。大文件的每个数据块再写入文件系统时往往被复制为多个备份，每个备份根据用户定义的分配策略被分配到分布到不同的数据节点上，从而极大地提高了系统数据的可靠性。文件系统在读取数据块时还会对其进行数据校验，如果发现数据存在错误则会对其进行重新复制。另外，HDFS 系统会在后台一直进行相关数据的一致性检测，从而可以保证每个文件块儿的副本数量维持在指定的数据水平上。综上所述，HDFS 具备极强的容错能力，可以为用户提供高可靠性的数据存储服务。

4.1.3 MapReduce 简介

MapReduce 是由 Google 提出的一种并行化编程模型，在该模型的帮助下用户可以在不改变单机编程习惯的前提下轻松实现对大规模数据的并行化处理。MapReduce 分布式处理框架包含一个统一的任务调度器，该调度器可以把一个巨型任务切分称多个子任务，并将每个任务分配到对应的处理节点上，最后将每个节点的处理结果合并起来得到

最终结果[22]。上述过程所涉及到的分布算法、调度机制、机器之间的通信等都被系统隐藏在统一编程接口之下，这使得程序员可以使用相对简单的程序迅速完成大规模的处理任务，工作效率得到了很大的提高。

（1） MapReduce 的基本原理

MapReduce 分布式处理框架由 Map 和 Reduce 两个核心操作构成。用户可以根据实际需要对 Map 函数进行自定义，该函数通常可以对切割过得原始 key/value 对集进行处理并生成一个中间的 key/value 对集。而后，MapReduce 提供给的库函数会将所有 Map 生成的 key/value 对集进行排序、聚合并将处理结果交给用户自定义的 Reduce 函数来处理。Reduce 函数可以对同一个 key 值的所有 value 进行合并操作并生成输出文件[22]，具体原理如图 4.3 所示。在整个过程中，MapReduce 主要使用两个进程：JobTracker 和 TaskTracker。它们负责 key/value 对集聚合、任务调度、通信控制等基础工作，而用户只需要给出 Map 和 Reduce 两个函数的定义以及其输入的数据类型。

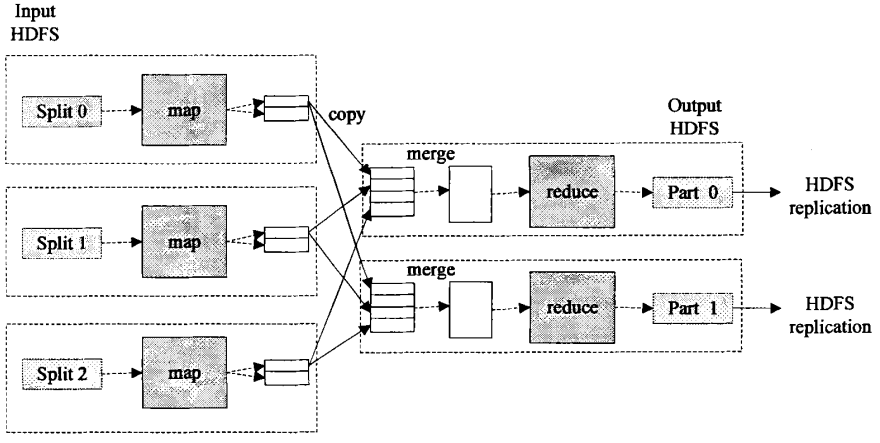

图 4. 3 MapReduce 应用程序执行过程简化视图

Fig. 4.3 Overview of MapReduce application execution

（2） MapReduce 的实现机制

MapReduce 分布式处理架构的实现采用了分而治之的思想，其使用一个主节点将大规模数据处理任务切分成多个子任务，并将它们分配到每个从节点上来处理。

具体的执行过程涉及到如下三个进程：

① JobClient

对于每一个作业，用户可以在客户端通过 MapReduce 提供的 JobClient 类将该作业的程序和参数配置进行打包形成 jar 文件，并将该文件上传到 HDFS 上，同时将该文件的存储路径提供给 JobTracker。JobTracker 会根据得到的信息创建一个 task，该 task 由 MapTask 和 ReduceTask 组成，并最终将被拷贝到不同的 TaskTracker 上面去执行。

② JobTracker

JobTracker 后台进程负责应用程序和 Hadoop 之间的链接。在用户将作业提交给系统之后，JobTracker 会根据得到的信息来确定将要被处理的分件，进而生成 Task 并将其分配到若干个处理节点之上。在每个 Task 的执行过程中，JobTracker 还负责对它们的状态进行监测。如果某个 Task 因故失败，JobTracker 就会根据预先设定的重启值对该任务进行重启操作。通常，JobTracker 会将重启后的任务分配到不同的节点上。Hadoop 系统只有一个 JobTracker，我们通常将其部署在集群的 Master 节点上运行。

③ TaskTracker

TaskTracker 运行在集群的从节点之上与相应的 DataNode 进程相结合完成本地数据的处理工作。虽然每个从节点上只运行一个 TaskTracker 进程，但是每个 TaskTracker 可以在本地启动若干个 Java 虚拟机，每个虚拟机可以独立运行一组 Map 和 Reduce 任务。从而，每个 TaskTracker 可以同时并行处理多个 Task。在具体运行过程中，TaskTracker 还要基于主/从结构与 JobTracker 相配合完成整个作业的处理工作。其中，JobTracker 位于主节点负责分配 Task，TaskTracker 位于从节点负责执行 Task。如果 JobTracker 在指定的时间内没有获得 TaskTracker 发来的确认信息，JobTracker 就认为 TaskTracker 已经因故崩溃，并将其负责的任务重启到其他数据节点上。

4.2 大规模 SVM 训练算法

支持向量机是非常强大的分类和回归工具。传统的 SVM 算法有牛顿法、内点法和下降法，然而由于这些算法对计算和存储需求随着训练向量数目的增加而快速增长，因此，在很多实际应用中这些算法往往会失去实际意义[12]。确切来讲，这些算法在执行过程中需要对与训练样本集相对应的核矩阵进行存储，而核矩阵所需要存储空间与训练样本集规模的平方称正比。当样本点数称千计时，所需内存已相当大，对于 100000 级别的样本，这些算法的单机运行更是无法实现。另外这些算法包含大量的矩阵运算，所需的运算时间往往过长，不能满足实际需要。

为了解决上述问题，研究人员一直在尝试开发新的支持向量机训练算法。经过观察发现，支持向量机在求解时表现出一些可以使训练算法得到简化的特性，例如：支持向量机求解过程中的最优化问题具有良好的凸性、支持向量机训练结果中的支持向量具备

很好的稀疏性等。这些性质为我们开发使用存储相对较少、运算速度相对较快的专用算法奠定了良好的基础。通常，基于支持向量的稀疏性，我们先把支持向量从训练样本集中筛选出来生成新的小规模样本集，进而基于该样本集对支持向量机进行训练，从而获得较快的收敛速度。显然，如何又快、又省地筛选出支持向量已经成为研究新算法的关键问题。通常，人们选用“迭代”和“并行”来控制筛选问题的时间和空间复杂度，详情如下。

4.2.1 大规模 SVM 训练算法的停机条件

通常，大规模 SVM 训练算法都或多或少的与迭代算法有关。它们往往从一个训练集的一个子集出发，将其视为工作集，通过训练得到局部最优解。然后通过预先设定的迭代停止条件来判断当前局部解是否在规定误差范围内满足全局最优。如果是则停止算法，否则，应该按照设定的策略对工作集进行更新，并重复上述迭代过程。因此，在介绍各种大规模 SVM 训练算法之前，有必要先介绍一下迭代停止条件。

常用的迭代停止条件有以下三种：

（1）很多算法都是从一个初始可行解出发，不断地进行迭代，在保证当前解在可行域范围内的情况下，是目标函数的值逐渐向最优值收敛。据此，我们可以指定一个简单的停机规则，即当目标函数值的上升或下降减缓到一定程度时停止算法。以目标函数具备最小值为例，我们可以设定一个大于零的值 $\xi > 0$，当目标函数值的下降值小于该阈值时算法停止。然而，有时候在某次迭代过程中目标函数的值不降反升，易于造成伪停机。因此，仅仅依靠这一种规则判断是否停机往往使算法表现出很强的不稳定性。

（2）由 3.1.4 可知，支持向量机最终要求解的是一个包含 Lagrange 乘子的二次规划问题，由 William Karush 提出的 KKT（Karush-Kuhn-Tucher conditions）条件为可以作为一个充分必要条件来判断某个乘子向量是否满足全局最优，从而可以作为迭代停止的依据。KKT 条件详述如下：

$$y_j\left(\sum_{i=1}^{l}\alpha_i y_i K(x_i,x_j)+b\right)\begin{cases}\geq 1, \{x_j \mid \alpha_j = 0\} \\ = 1, \{x_j \mid 0 < \alpha_j < C\} \\ \leq 1, \{x_j \mid \alpha_j = C\}\end{cases} \tag{4.1}$$

在实际应用中，我们很难做到使当前解严格满足上述条件。只要在一定误差范围内满足上述条件，当前解便可以视为该规划问题的最优解。

（3）在最优化问题的实际求解过程中，我们往往无法做到求得理论上的最优解，只能在一定精度内确保全局最优，这使得合理设定停机条件精度变得尤为重要。如果精度设定太高，则算法会因迭代次数过多而浪费很多时间，此时预测精度也不会有明显的增加。因此，我们可以设定一个最大迭代次数，在合理保证预测精度的前提下，避免不必要的时间浪费。

在具体应用中，我们往往根据实际需要对上述三种迭代停止条件进行合理的选择和组合，从而在满足预测精度和预测实效性的前提下生成合理的迭代停止条件。

4.2.2 基于迭代的大规模 SVM 训练算法

迭代思想是解决大规模 SVM 训练的行之有效的一种方法，其基本原理就是把原来的大规模问题分解成多个小规模的子问题来求解。通常，该算法从一个初始的子问题出发，通过求解子问题过得原问题的近似解，然后根据预先设定的迭代策略对子问题进行更新并对新生成的子问题进行再次求解。重复上述求解过程，直至子问题的近似解逐步收敛为原问题的最优解。在实际应用中人们尝试过许多不同的子问题选取方法和迭代策略，进而衍生出多种基于迭代的大规模 SVM 训练算法，主要包括：选块法、分解法、序列最小最优化方法等。

（1） 选块儿算法（chunking）

选块儿算法通常首先确定一个原数据集的任意子集，针对该子集使用标准优化算法进行求解，得到原始问题的局部最优解$\alpha=(\alpha_1,\cdots,\alpha_l)^T$，然后通过以下步骤对当前的块儿进行更新。

① 如果α_j为非零则保留该子集中与α_j（支持向量）相对应的训练样本，丢弃子集中其它的训练样本。

② 利用求得的决策函数对训练集中除去该子集后的所有训练样本进行检验，把其中 M 个违背 KTT 条件最严重的样本（M 是预先设定的参数）添加到新块儿中，以新块儿为基础重新进行求解。

重复以上过程，其中每一个新的子问题的α初始值都取为前一个子问题的解，最后直到满足某一个停机准则为止。通常，工作集的规模是逐渐增加的，虽然有时候可能出现临时性的减小。上述算法在支持向量的数量远远小于训练集样本数目的时候表现出很好的性能，可以极大地提高运算速度。但是，如果原训练集中支持向量所占的比例比较大，那么随着迭代次数的逐渐增加，当前块儿的样本规模也会越来越大，甚至有时候与原始训练集的样本数目相差无几。此时，算法会变得十分缓慢，失去实际意义。

（2） 分解算法(decomposing)

选块儿算法的最终目标就是要找出所有的支持向量，因此在算法执行过程中要对当前块儿所对应的核函数矩阵进行存储。这样，当支持向量的规模很大时，该算法会因为运算速度太慢、内存要求太高而失去实际意义。此时，我们可以使用分解的技巧来解决上述问题。分解算法与选块儿算法十分相似，但是他每次只对固定数目的 Lagrange 乘子进行更新，其它的 Lagrange 乘子保持不变。因此，如果我们将工作集之外的若干个情况最糟的样本加入到工作集，就必须同时将同等数目的原工作集样本移除。可见，即使支持向量的规模超过而来工作集的大小，工作集的规模在算法运行过程中也保持不变。这使得每次迭代时算法只针对固定规模的训练集进行求解，具备很快的训练速度。由于每次迭代被移出的样本所对应的 Lagrange 乘子保持不变，所以经过若干次的迭代之后所有样本对应的 Lagrange 将趋于全局最优。

（3） 序列最小最优化方法（SMO）

序列最小化优化算法是分解算法的一种特出情况，它将工作集中的样本数确定为 2，这也是工作集所能达到的最小工作规模。因此在每次迭代中，该算法只针对两个样本点 (x_i, y_i) 和 (x_j, y_j) 所对应的 α_i 和 α_j 进行调整[23]。由于该算法每次只对拥有两个变量的最优化问题进行求解，因此我们完全可以使用解析方法代替二次规划方法来求得局部最优解。由于该算法的工作集规模小，所以相对分解算法其迭代的次数往往比较多。但是，解析方法使得每次迭代的计算量变得很少，算法从整体上表现出较快的收敛速度。同时，该算法在执行过程中没有矩阵运算，无需存储核函数矩阵，空间复杂度大大降低。

4.2.3 基于并行的大规模 SVM 训练算法

由 4.2.1 中论述可知，迭代方法可以有效解决大规模的支持向量筛选问题，具备较少的内存需求和较好的收敛速度。然而，仔细观察后可以发现，支持向量筛选问题具有自然的可分性，因此采用并行思想来进一步降低大规模 SVM 训练算法的需求、提高算法的收敛速度成为了可能。自 20 世纪 70 年代以来，随着并行和分布式计算的发展，并行化的 SVM 训练算法应运而生。并行算法将时间、空间开销分配到多个计算节点上，然后对多个训练结果进行合并更新并重新分配训练任务，直到达到预期的训练效果。大量实验表明，并行 SVM 算法对内存的需求量很小，在保证分类精度的前提下收敛速度快获得了很大的提高。下面介绍几种常见的 SVM 并行算法。

（1） 并行 SMO 算法

Deng Kun 等人提出了一种并行的 SMO 支持向量机训练算法。该算法把大规模数据及随机划分成 n 个较小的子数据集，然后基于每个子数据集进行训练，得到该子数据集

的支持向量机。最后将这些局部支持向量机的乘子进行简单的链接得到最终的支持向量乘子[24]。由于每个子数据集都是单独训练，因此算法可以实现并行处理。由于 SMO 的时间复杂度为 $O(n^2)$,所以每个自训练集的时间可以下降到原来的 $1/n^2$。大量实验表明，该算法对于线性支持向量机具有很好的分类效果，而对于非线性支持向量机，其分类器正确率总是小于某个子数据集得到的支持向量机。划分数为 8 时，该算法如图 4.4 所示。

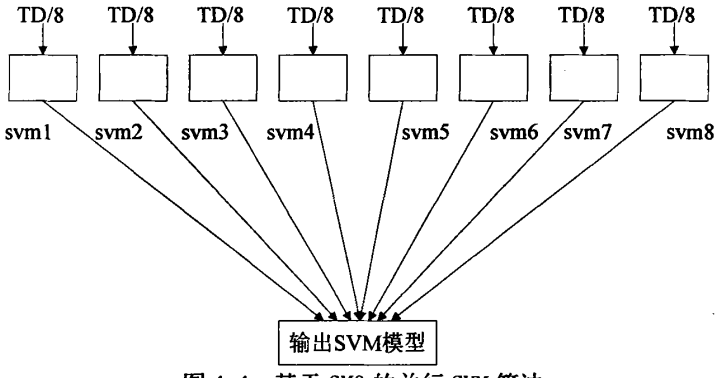

图 4.4　基于 SMO 的并行 SVM 算法

Fig. 4.4　Parallel SVM training algorithm based on SMO

（2）　分层级联并行 SVM 训练算法

Grafs等人提出了一种分层级联SVM训练算法用来解决百万级样本规模的支持向量机训练问题。该算法把大规模数据及随机划分成 n 个较小的子数据集，然后基于每个子

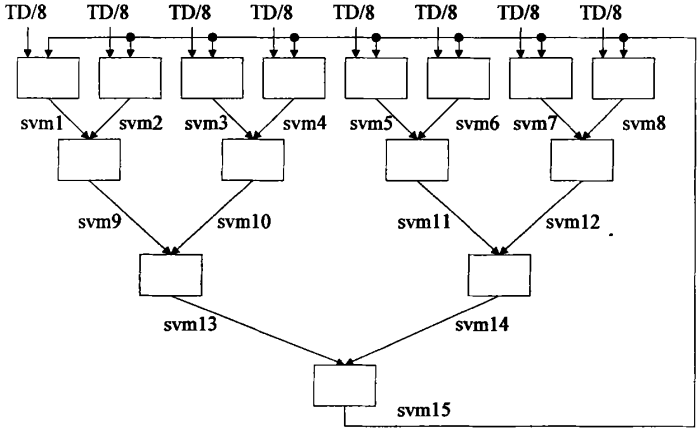

图 4.5　分层级联并行 SVM 算法

Fig. 4.5　Schematic of binary cascade architecture

数据集进行训练，筛选出其支持向量，并将训练结果两两合并。在新得出的数据集上继续进行训练并重复上述合并策略直至得到最终的决策函数。通常，经过一次多级筛选我们就可以得到分类效果比较好的决策函数，如果我们想要得到全局最优解，我们可以把最后一层的训练结果反馈到第一层，根据迭代停止条件判断其是都为全局最优，如果是则停止迭代，反之，则除去当前工作集中的非支持向量，并将总样本空间中除当前工作集以外的违背 KKT 条件最严重的若干个样本加入到工作集中进行更新，然后重复上述筛选工作[25]，具体流程如图 4.5 所示。

4.3 Hadoop 环境下大规模 SVR 训练算法

SVR 训练最终更是由 SVM 训练来实现的，所以以上提到的大规模 SVM 训练算法可以直接应用到大规模 SVR 训练中。由 4.2 中可知，简单的并行 SMO 算法具有训练速度快、实现简单等优点，但是对于非线性的分类问题效果不佳。分层级联并行 SVM 训练算法理论上可以求得全局最优解，但是实现过程过于复杂，迭代次数相对较多，收敛相对较慢[26]。中和以上两种算法的优劣势，基于 MapReduce 分布式处理模式的运行机制，本文提出了 Hadoop 环境下大规模 SVR 并行训练算法，训练模式如图 4.6 所示。其中，Map 负责训练每个子数据集筛选出局部支持向量，Reduce 负责整合所有的局部支持向量并生成新的训练样本集，然后基于此样本集进行训练得到新的支持向量机模型，外部程序根据停机条件决定是否要进行更新和迭代。

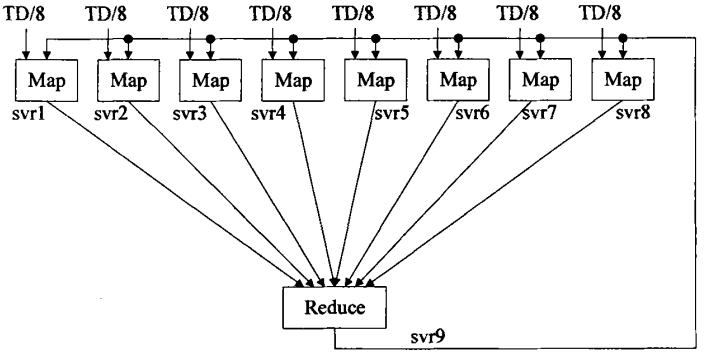

图 4.6 Hadoop 环境下大规模 SVR 训练算法基本原理

Fig. 4.6 Overview of large scale SVR training algorithm in Hadoop environment

4.3.1 算法详细设计

由于上述 Hadoop 环境下大规模 SVR 训练算法在实现过程中需要进行多次迭代，所以我们应该首先明确它的迭代停止条件。由 4.2.1 可知，单独使用一种迭代条件难以获得很好的训练效果，结合短时交通流预测在精度和实时性方面的实际需要，本文采用如下迭代停止条件，即如果当前 Lagrange 乘子向量能够使式 3.15 所示的目标函数的下降量大于零并且小于预设阈值或者当前迭代次数已经达到最大值算法停止。以该迭代停止条件和更新标准为基础，结合 MapReduce 的运行机制，本文设计了 Hadoop 环境性大规模 SVR 训练算法，如图 4.7 所示。

（1） 将训练样本集划分为 n 个子数据集，设定训练精度。

（2） 为每个子数据集分配一个 Map 操作进行 SVR 训练，筛选出支持向量。

（3） Reduce 合并所有的局部支持向量并对新的样本集进行 SVR 训练。

（4） 如果满足迭代停止条件，算法结束，否则，转到（5）。

（5） 将所有样本中除当前模型所包含的支持向量外违背 KTT 条件最严重的 K 个样本加入训练样本集，进行更新，转到（1）。

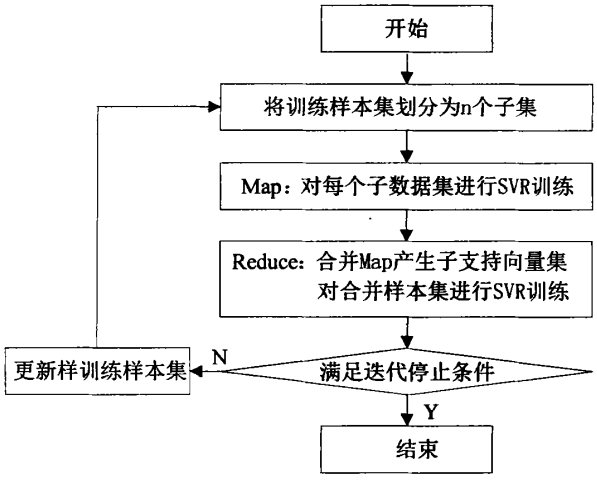

图 4.7 Hadoop 环境下大规模 SVR 训练算法

Fig. 4.7 Large scale SVR training algorithm in Hadoop environment

4.3.2 算法伪代码

Hadoop 环境下 SVM 训练算法主要包含三个部分：提交 Hadoop 作业的迭代训练、单次迭代的 Map 操作和单次迭代的 Reduce 操作，详情如下所示。

（1）提交 Hadoop 作业的迭代训练伪代码

```
Input: training dataset
Output: dataset of support vectors
While (stop condition) do
    Configuration
        Set the configurations which can be used to run the parallel algorithm of SVR
    Define JOB
        Generate a job using the configurations above
        Set the input and output directory of current job
        Set the class of mapper
        Set the class of reducer
        Set the input and output format
        Set the class of output key
        Set the class of output value
    Run the job defined
End while
```

（2）单次迭代 map 函数伪代码

Map 函数负责对相应的子数据集进行 SVR 训练，筛选出其支持向量，得到支持向量机模型。伪代码如下所示。

```
Input: one block of training dataset
Output: results of key/value after of training of SVR
If keyStr != null && !keyStr.equals("-1") then
    Block Index = read file (key, value)
Else
    If keyStr.equals("-1") then
        SVRMapDatas = key/value
        SVs = train_svr (SVRMapDatas)
        Save into SVs
    End if
End if
```

（3）单次迭代 reduce 函数伪代码

Reduce 函数负责将 Map 函数产生的子支持向量集合并，然后基于新数据进行训练，产生新的支持向量机模型，详细伪代码如下。

```
Input：key/values from the map function's result
Output：the file used for the next input
While (condition) do
    combine_subsets (SVRMapDatas, Key Value)
End while
If (!Condition) then
    SVs = train_svr (SVRMapDatas)
    Current Value = compute_descent_value (SVs)//use this value to confirm the main
loop continues or not
End if
```

4.4 Hadoop 环境下 SA-SVR 短时交通流预测模型

基于 4.3 提出的并行 SVR 训练算法，结合短时交通流预测的数据特点，我们构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型。该模型通过对交通流数据的重构和规约

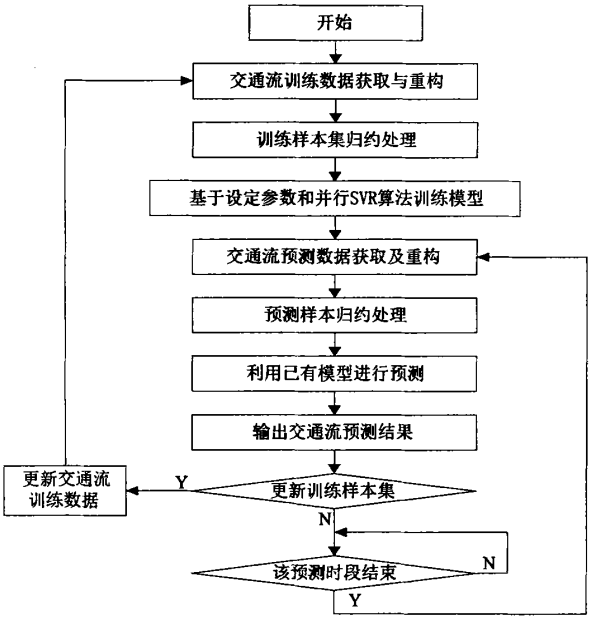

图 4.8 Hadoop 环境下 SA-SVR 短时交通流预测模型

Fig. 4.8 Short term traffic flow forecasting model based on SA-SVR in Hadoop environment

来获得训练样本集，使用模拟退火算法获得最优参数，进而基于该参数和训练样本集对 SVR 进行训练，最终得到用于短时交通流的预测的决策函数。Hadoop 环境下 SA-SVR 实现短时交通流预测的具体步骤如图 4.8 所示。

（1） 获取交通流数据并进行时序相空间重构。

（2） 对训练样本集进行归约化处理。

（3） 基于设定参数和并行 SVR 算法训练模型。

（4） 交通流预测数据获取及样本重构

（5） 对预测样本集进行归约化处理。

（6） 利用已有模型进行预测。

（7） 输出交通流预测结果。

（8） 如果需要更新样本集转到（1）。

（9） 如果该预测时段结束转到（4），否则转到（9）。

4.5 本章小结

随着训练样本规模的变大，传统 SVR 训练算法的时间和空间复杂度急剧增加，以至于这些算法的单机实现在具体应用中往往会失去实际意义。本章详细介绍了分布式处理平台 Hadoop 的基本原理，包括分布式文件系统 HDFS 的运行结构和分布式处理框架 MapReduce 的处理机制。然后，在分析单机大规模 SVM 训练算法瓶颈问题的基础上，从“迭代”和“并行”两个角度详细阐述了解决这一问题的具体算法，并分析了不同算法的优劣势。最后，将 MapReduce 的分布式处理特点与大规模 SVM 训练算法的“迭代”思想结合起来，提出了 Hadoop 环境下大规模 SVR 训练算法，结合第 2 章给出的交通流数据构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型，并给出了详细的预测步骤。

5 实验

由第 3 章和第 4 章提出的实时交通流预测模型可知，在预测某一个时间点的交通流流量时，我们需要获得其前 5 个时段的交通流数据，将其重构后形成预测样本并将样本带入决策函数获得交通流预测值。然而，本文实验过程中不具备接入真实交通运行环境的条件，所以采用以下方法进行仿真实验：将某一时间段内的历史交通流数据看作待预测数据。根据时序相空间重构原则，基于每个预测时间段前 5 个时间段的交通流数据生成该预测时间段的预测样本，将样本带入决策函数获得改时间段的交通流预测值，并与该时间段的真实历史交通流数值进行比较[11]。显然，该方法不会对评价短时交通流预测模型的预测精度产生影响。

5.1 SA-SVR 短时交通流预测实验

5.1.1 实验环境介绍

本实验在单机环境下实现,服务器型号为曙光 I420-G10，CPU 为 I5-2320 3.0GHZ，内存为 4G，硬盘为 500G，预装 Ubuntu11 操作系统。算法开发语言为 Java，使用 eclipse 集成开发环境和 libsvm 开发工具包。其中，libsvm 工具包为用户立提供了丰富的开发工具和编程接口，可以实现数据归一、训练集读取、训练、预测等机器学习功能[27]。

5.1.2 评价指标与对比算法

为了验证 SA-SVR 短时交通流预测模型的有效性，本章共进行了两组实验。第一组是基于 BP 神经网络的短时交通流预测模型与基于支持向量回归机的短时交通流预测模型之间的性能比较，将周五（2014 年 3 月 28 日）的数据作为预测样本；第二组是基于 GS-SVR、PSO-SVR 和 SA-SVR 的短时交通流预测模型之间的性能比较，将周日（2014 年 3 月 30 日）的数据作为预测样本。训练样本集由第二章所述实验数据中最接近预测日期的 10000 组样本组成。实验中，模型的性能评价指标采用均方根误差（RMSE）、平均相对误差百分率（MAPE）、训练时间和泛化能力，RMSE 和 MAPE 定义如下:

$$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y}_i)^2} \tag{5.1}$$

$$MAPE=\frac{\sum_{i=1}^{n}|y_i-\bar{y}_i|/y_i}{n}\times 100\% \tag{5.2}$$

其中，y_i 表示实际值，$\bar{y}_i$ 表示预测值。

5.1.3 BP 神经网络与支持向量回归机对比试验

本实验中，神经网络采用 5-5-1 网络结构，分别代表输入层、隐含层和输出层的神经元的个数。网络初始权值为[-0.1,0.1]之间的随机数，系统误差是 10^{-6}，最大迭代次数为 1000。支持向量机采用高斯径向基函数作为核函数，通过网格算法获得参数：C=1.0，ε=0.0039，σ=1.0[28]。预测结果如图 5.1 和图 5.2 所示。

从表 5.1 可以看出，基于支持向量回归机的交通流预测模型在预测精度、收敛时间和泛化能力上均优于 BP 神经网络。预测精度：SVR 的 MAPE 为 5.01%，优于 BP 神经网络的 7.57%。训练时间：支持向量机预测模型的训练时间为 7.53 秒，远低于 BP 神经网络预测模型的训练时间 22.04 秒。泛化能力：由图 5.2 可以看出，支持向量回归机预测模型在拟合误差比较大的时候，其预测误差仍能保持相对稳定，泛化能力强于 BP 神经网络。

表 5.1 BP 神经网络和 SVR 性能比较

Tab. 5.1 Comparison of BP-neural network and SVR

模型	MAPE	RMSE	训练时间	泛化性能
BP 神经网络	7.57%	52.51	22.04	一般
SVR	5.01%	35.081	7.53	强

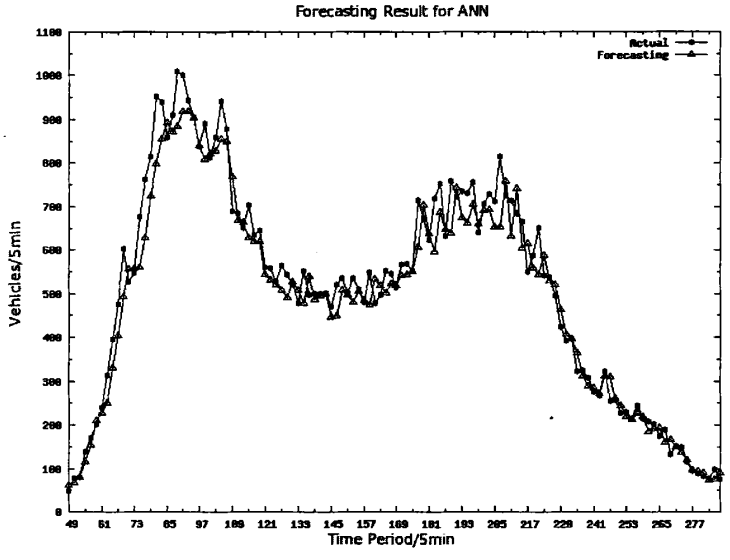

图 5.1 BP 神经网络交通流预测结果

Fig. 5.1 Forecasting result of traffic flow using BP-artificial neural network

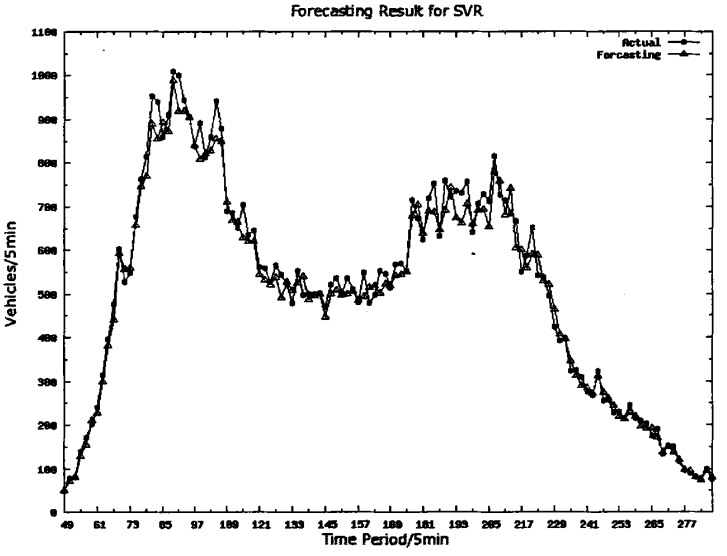

图 5.2 SVR 交通流预测结果

Fig. 5.2 Forecasting result of traffic flow using SVR

5.1.4 基于参数优化的 SVR 模型对比试验

本实验基于周日（2014 年 3 月 30 日）的样本集，分别采用网格算法、粒子群算法和模拟退火算法对支持向量回归机的参数：惩罚因子C、不敏感损失参数ε和核宽度σ进行优化，结果如表 5.2 所示。其中粒子群优化方法采用群体规模为 20，最大迭代次数为 10000，加速常数为 C1=C2=2.0，惯性权重 Wmax=0.9，Wmin=0.4。基于最优参数，SA-SVR 交通流预测模型的预测结果如图 5.3 所示，预测误差如图 5.4 所示。

表 5.2 不同算法的参数优化比较

Tab. 5.2 Result of Different Parameter Optimization Algorithm

算法	参数	MAPE	RMSE	收敛时间
网格算法	C=1.00，ε=0.0039，σ=1.000	4.43%	35.081	304.64s
粒子群算法	C=1.12，ε=0.0042，σ=0.935	5.54%	49.79	39.20s
模拟退火算法	C=1.05，ε=0.0041，σ=0.943	4.96%	42.98	25.13s

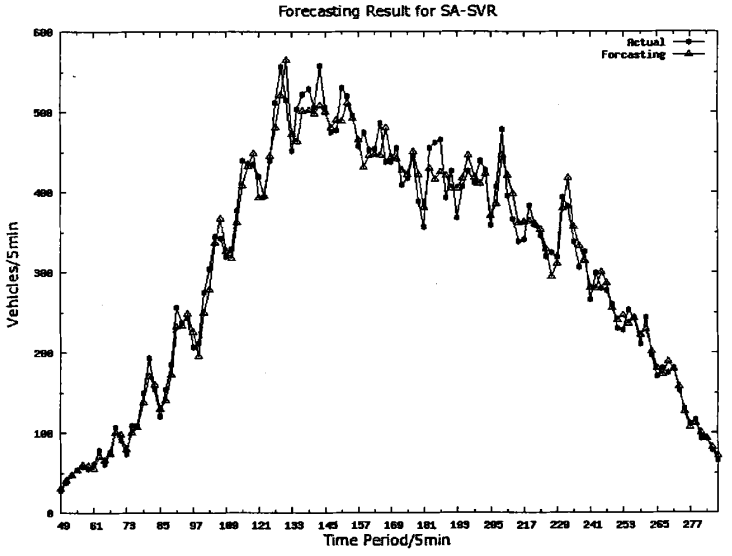

图 5.3 SA-SVR 交通流预测结果

Fig. 5.3 Forecasting result of traffic flow using SA-SVR

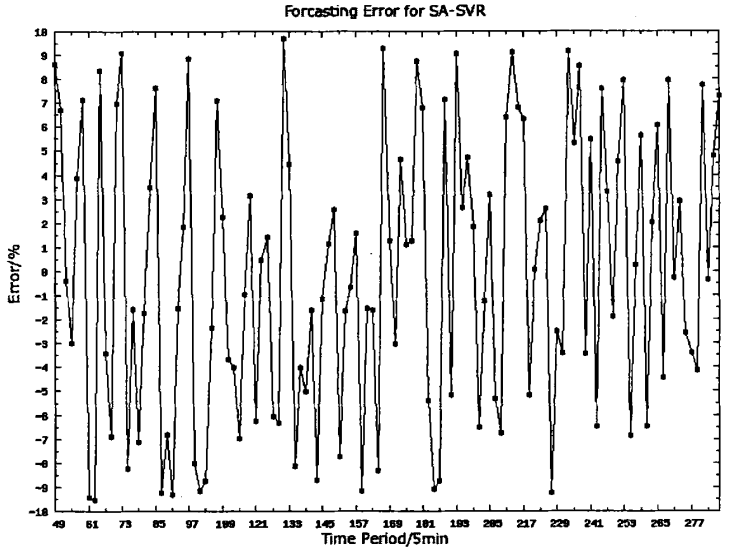

图 5.4 SA-SVR 交通流预测误差

Fig. 5.4 Forecasting error of traffic flow using SA-SVR

从表 5.2 可以看出，基于网格算法、粒子群算法和模拟退火算法的参数优化结果比较接近，基于不同优化参数的 SVR 预测精度也相差不多，说明模拟退火算法可以为 SVR 选择合适的参数。虽然基于网格算法的 SVR 预测精度稍微高一些，但是其收敛时间比

较长，为 304.64s，不满足短时交通流预测的实时性要求。基于模拟退火算法的 SVR 在保证预测精度的前提下，寻优时间具有明显的优势，仅为 25.13s。

由图 5.3 和图 5.4 可见，SA-SVR 短时交通流预测模型具有很好的跟随能力，其预测误差百分率最大为 9.81%。同时，在拟合误差比较大的时候，该模型预测误差相对稳定，表现出很强的泛化能力。

5.2 Hadoop 环境下 SA-SVR 短时交通流预测实验

5.2.1 实验环境介绍

（1）Hadoop 分布式平台硬件环境

本实验使用 8 台服务器,服务器型号为曙光 I420-G10，CPU 为 I5-2320 3.0GHZ，内存为 4G，硬盘为 500G，预装 Ubuntu11 操作系统。8 台机器通过路由器和交换机连接在一个小局域网里面，组成一个分布式集群，集群详细配置和系统结构如图 5.5 所示。

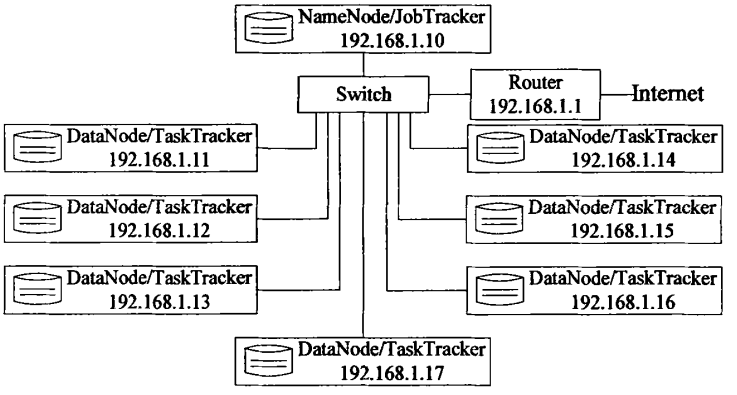

图 5.5 Hadoop 集群架构图

Fig. 5.5 Cluster constructer for Hadoop

（2）Hadoop 分布式平台软件环境

Hadoop 在运行过程中要启动若干个进程。其中，NameNode、SecondaryNameNode 和 JobTracker 运行在 namenode 主机上。每一台 datanode 主机上都会运行一个 DataNode 和 TaskTracker 进程。各个进程之间的协作关系如图 5.6 所示，其中 NameNode 进程和 DataNode 进程属于 Hadoop 分布式文件系统，彼此进行通信。SecondaryNameNode 对 NameNode 中分布式文件系统的元数据进行备份，防止出现单点故障引起的系统失败。JobTracker 负责为每个 TaskTracker 分配 map 和 reduce 任务并监控它们的运行，彼此协作构成 MapReduce 架构。

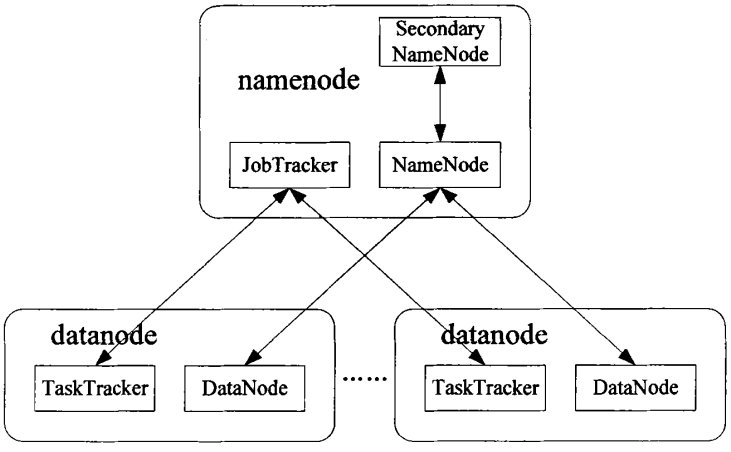

图 5.6 Hadoop 分布式平台软件环境

Fig. 5.6 Software structure of Hadoop platform

（2）实验平台具体配置

结合试验具体需要我们需要对 Hadoop 分布式平台做一些相应的配置，需要修改 core-site.xml、hdfs-site.xml 和 mapred-site.xml 配置文件，详细配置情况如表 5.3 所示。

表 5.3 数据项描述

Tab. 5.3 Data item description

配置文件	配置项	配置值
core-site.xml	fs.default.name	192.168.1.10:9000
hdfs-site.xml	dfs.name.dir	usr/hadoop/hdfs/name
hdfs-site.xml	dfs.data.dir	usr/hadoop/hdfs/data
hdfs-site.xml	dfs.replication	2
mapred-site.xml	mapred.job.tracker	192.168.1.10:9001
mapred-site.xml	mapred.system.dir	usr/hadoop/mapred/system
mapred-site.xml	mapred.local.dir	usr/hadoop/mapred/local

5.2.2 评价标准与对比算法

为了验证 Hadoop 环境下并行 SVR 的有效性，本章共进行了 20 组对比实验，实验训练样本数量最小为 1000，最大为 100000。实验过程中我们对单机 SVR 训练算法和 Hadoop 环境下 SVR 算法的训练时间和预测精度进行比较，采用以下模型评价指标：均

方根误差（RMSE）、平均相对误差百分率（MAPE）、训练时间和加速比（T_a），RMSE、MAPE 和 T_a 定义如下：

$$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y}_i)^2} \tag{5.3}$$

$$MAPE=\frac{\sum_{i=1}^{n}|y_i-\bar{y}_i|/y_i}{n}\times 100\% \tag{5.4}$$

$$T_a=T_s/T_p \tag{5.5}$$

其中，y_i 表示实际值，$\bar{y}_i$ 表示预测值，T_s 表示单机 SVR 的训练时间，T_p 表示 Hadoop 环境下 SVR 的训练时间[29]。

5.2.3 并行 SVR 与单机 SVR 性能对比试验

本文第二章已经准备了 2013 年 4 月 1 号到 2014 年 3 月 27 号共计 104256 组交通流训练数据，训练数据进行时序相空间重构后为产生 104251 组训练样本。本实验从中选取前 100000 组用于 SVR 训练，并基于 2014 年 3 月 29 日（周六）所对应的样本进行交通流预测。实验详情及结果表 5.4 和表 5.5 所示。单机 SVR 和并行 SVR 训练时间对比如图 5.7 和图 5.8 所示，加速比如图 5.9 和图 5.10 所示。训练样本数为 100000 的情况下预测精度对比如图 5.11 和 5.12 所示，并行 SVR 算法预测误差如图 5.13 所示。

表 5.4 基于小数据集的并行 SVR 与单机 SVR 性能对比

Tab. 5.4 Comparison between parallel SVR and stand-alone SVR based on small data set

样本数量	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k
单机(s)	0.109	0.328	0.687	1.263	1.919	2.698	3.697	4.758	6.022	7.534
并行(s)	7.224	7.244	7.264	7.304	7.324	7.388	7.448	7.576	7.636	7.7
加速比	0.015	0.045	0.094	0.172	0.262	0.365	0.496	0.628	0.788	0.978

表 5.5 基于大数据集的并行 SVR 与单机 SVR 性能对比

Tab. 5.4 Comparison between parallel SVR and stand-alone SVR based on large data set

样本数量	1w	2w	3w	4w	5w	6w	7w	8w	9w	10w
单机(s)	7.534	31.54	72.35	129.7	206.1	320.0	410.2	541.0	697.1	869.1
并行(s)	7.7	9.136	11.44	14.87	18.99	23.98	29.97	37.33	44.89	54.18
加速比	0.978	3.452	6.322	8.722	10.85	13.34	13.68	14.49	15.53	16.03

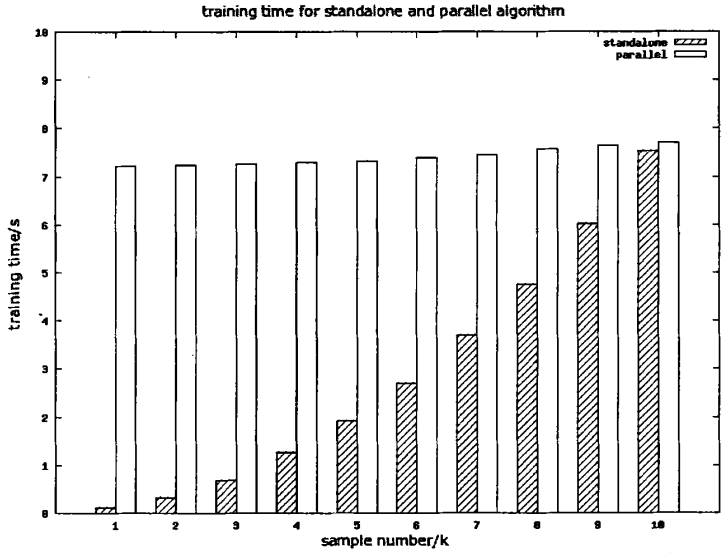

图 5.7 基于小数据集的单机 SVR 和并行 SVR 训练时间比较

Fig. 5.7 Comparison between stand-alone SVR and parallel SVR based on small data set

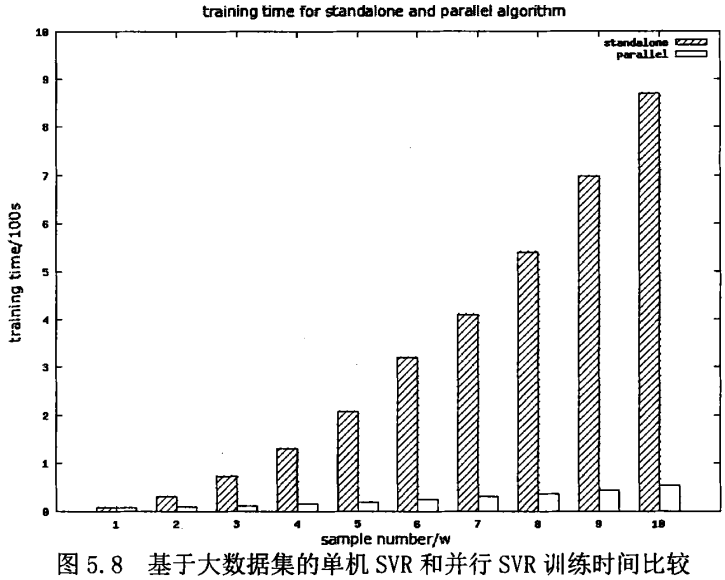

图 5.8 基于大数据集的单机 SVR 和并行 SVR 训练时间比较

Fig. 5.8 Comparison between stand-alone SVR and parallel SVR based on large data set

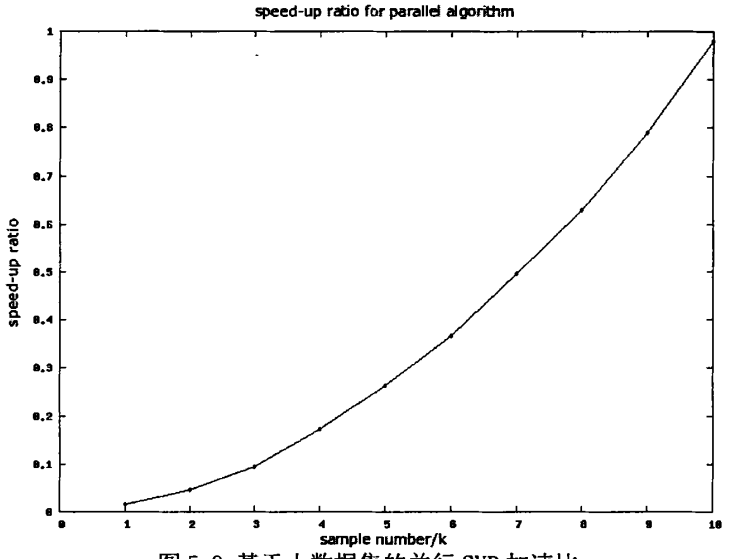

图 5.9 基于小数据集的并行 SVR 加速比

Fig. 5.9 Speed-up ratio of parallel SVR based on small data set

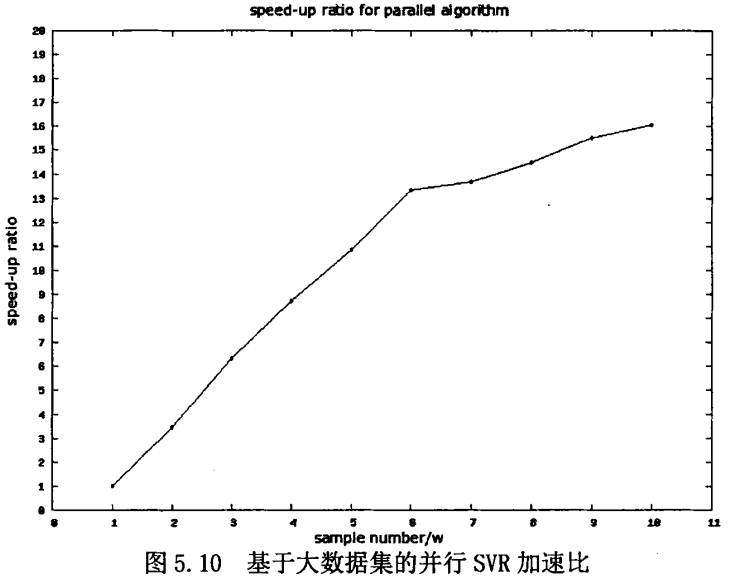

图 5.10 基于大数据集的并行 SVR 加速比

Fig. 5.10 Speed-up ratio of parallel SVR based on large data set

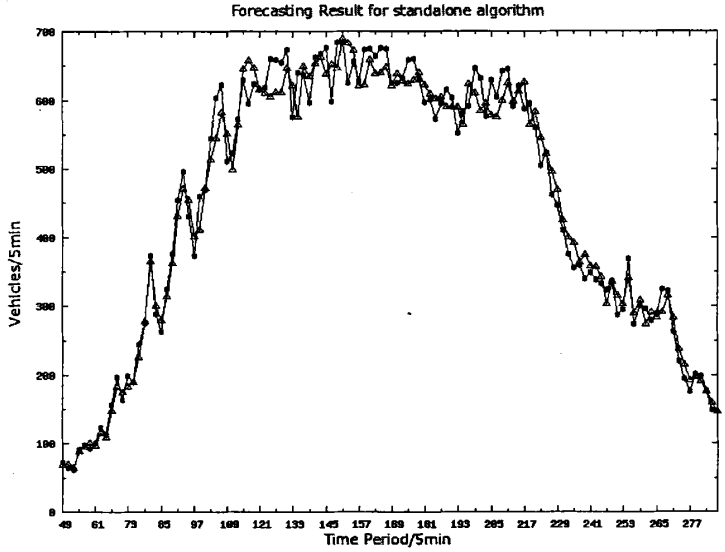

图 5.11　单机 SVR 交通流预测结果

Fig. 5.11　Forecasting result of standalone SVR

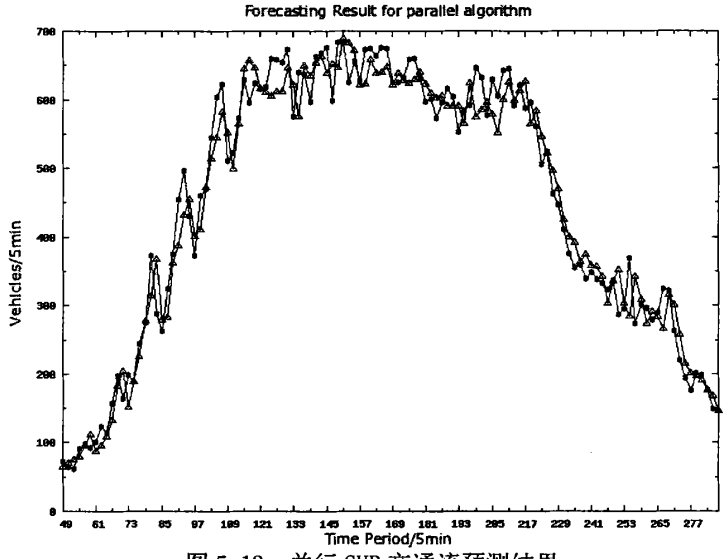

图 5.12　并行 SVR 交通流预测结果

Fig. 5.12　Forecasting result of parallel SVR

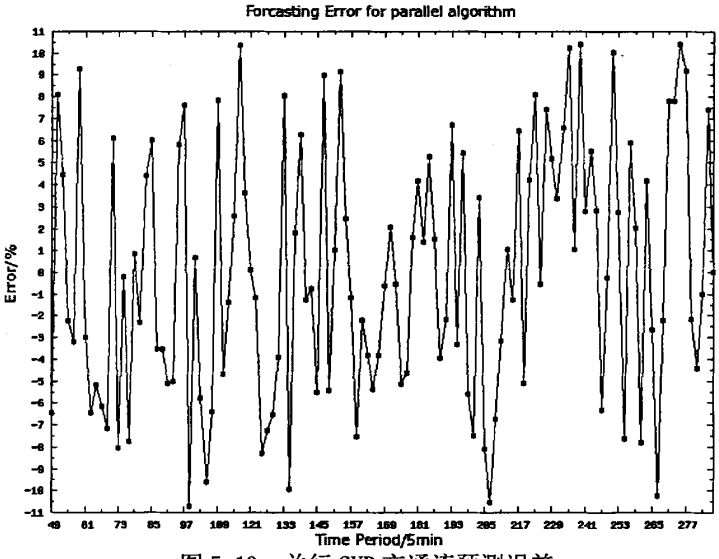

图 5.13　并行 SVR 交通流预测误差

Fig. 5.13　Forecasting error of parallel SVR

由图 5.7 和图 5.9 可知，在实验样本数据较少（少于 10000）的情况下，Hadoop 环境下 SVR 由于存在一些通讯协作开销，其训练时间反而比单机训练时间长，加速比也小于 1。由图 5.8 和 5.10 可知，随着实验样本数量的增加，单机 SVR 的训练时间迅速增长，几乎与样本数量的平方成正比，而并行 SVR 的训练时间增长相对比较缓慢。加速比也随着样本数量的增加而增加，并在训练样本数量大于 60000 以后增速放缓，最终在训练样本数量为 100000 的情况下达到 16.03，表现出很好的训练效率。由图 5.11、5.12 和 5.13 可知，在训练样本数量为 100000 的情况下，并行 SVR 的预测精度与单机 SVR 的预测精度相差不大，平均相对预测误差为 5.20%，最大相对预测误差为 10.70%。

5.3　本章小结

本章对第 3、4 章提出的短时交通流预测模型分别进行了仿真实验。针对第 3 章的 SA-SVR 短时交通流预测模型本文进行了两组对比试验：第一组是基于 BP 神经网络的短时交通流预测模型与基于支持向量回归机的短时交通流预测模型之间的性能比较；第二组是基于 GS-SVR、PSO-SVR 和 SA-SVR 的短时交通流预测模型之间的性能比较。实验结果表明，SA-SVR 短时交通流预测模型具备很高的预测精度，预测平均误差为 4.96%，最大平均误差为 9.81%。针对第 4 章的实验为 Hadoop 环境下 SA-SVR 短时交通流预测模型与单机 SA-SVR 短时交通流预测模型性能对比试验。实验表明，该模型面对大规模

训练样本具备很快的训练速度，相对单机算法加速非常明显，最终加速比为 16.03。同时，该模型具备良好的预测精度，预测平均误差为 5.20%，最大不超过 10.70%。

结　论

随着社会经济的发展，我国城市化进程逐渐加快，机动车持有量和交通需求量也急剧增加，交通运输已经逐步成为拉动国民经济的重要产业。交通运输业的快速发展也同时带来了交通拥堵、交通事故、交通环境污染等“城市病”。一直以来，研究人员尝试用各种方法解决日益严重的交通拥堵问题，智能交通系统的出现使得人们可以综合利用数据采集、数据分析、智能诱导等现代科技对交通系统进行有效的控制与优化。其中，实时准确的交通流预测是实现智能诱导的前提和关键，也是智能交通系统中的重要组成部分。近几年来，随着物联网和云计算技术的出现智能交通迎来了第二次高速发展进而衍生出“车联网技术”，这为短时交通流预测提供了绝佳的硬件平台。因此，如何在新技术环境下进一步提高短时交通流预测的速度和精度已经成为迫在眉睫的研究挑战。本文在充分研究现有短时交通流预测模型优劣势的基础上，基于支持向量回归机和模拟退火算法构建了 SA-SVR 短时交通流预测模型，取得了良好的预测精度。同时，本文基于支持向量的稀疏性特点将该模型成功移植到 Hadoop 环境下，构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型。最终在不影响预测精度的前提下，利用 Hadoop 强大的分布式处理能力进一步提高了短时交通流预测的实时性。

（1） 创新点

① 针对传统参数优化算法收敛速度慢这一问题，本文提出了使用模拟退火算法来对 SVR 的参数进行优化。模拟退火算法是一种新型的随机搜索算法，它适合于解决大规模组合优化问题，具有描述简单、运行效率高和较少受到初始条件约束等优点。实验证明该算法相对于网格算法和粒子群优化算法具备更快的收敛速度。

② 针对单机大规模 SVR 训练算法训练时间太长这一问题，本文在分析各种并行 SVR 训练算法的基础上，结合 Hadoop 环境下 MapReduce 的分布式处理特点，提出了 Hadoop 环境下大规模 SVR 训练算法。从而利用 Hadoop 强大的分布式处理能力大大缩短了大规模样本下 SVR 的训练时间。本文实验环境下，该并行算法相对单机算法的加速比达到 16.03。

③ 针对智能交通和车联网技术发展中对于基于云平台的短时交通流预测算法的需求问题，本文基于 SA-SVR 算法构建了 SA-SVR 短时交通流预测模型，并结合 Hadoop 环境下大规模 SVR 训练算法成功地将该模型移植到 Hadoop 平台下，从而构建了 Hadoop 环境下 SA-SVR 短时交通流预测模型。实验证明该模型具备良好的预测精度和较快的收敛速度。

（2） 工作展望

以智能交通、车联网、云计算等技术的快速发展为背景，本文对基于云平台的短时交通流预测算法进行了尝试研究，但仍存在很多问题。以下问题在未来还需要进一步研究：

① 本文的研究工作只使用了原始交通流数据中的交通流量这一项，对于平均速度、占有率等数据项没有加以利用。如何充分利用多维的交通流数据构建一个多输入的、适应性更强的、预测精度更高的短时交通流预测模型是有待进一步深入研究的问题。

② 本文用于优化 SVR 参数的模拟退火算法具备并发执行的潜质。如何科学的划分解空间，进而通过在每个子解空间上的并行搜索来提高参数优化的收敛速度有待进一步深入研究。

③ Hadoop 平台具备很强的分布式处理能力，但是过多的处理进程快带来巨大的通信开销。如何通过确定合理的 Map 数和 Reduce 数，开发相应的优化调度算法来进一步提高 Hadoop 平台的吞吐量，进而提高短时交通流预测模型的预测速度有待于研究人员进一步研究。

参 考 文 献

[1] Weiland R J, Purser L B. Intelligent transportation systems [J]. Transportation in the new millennium, 2000.

[2] Miche M, Bohnert T M. The internet of vehicles or the second generation of telematics services [J]. ERCIM News, 2009, 77: 43-45.

[3] 刘静，关伟. 交通流预测方法综述[J]. 公路交通科技, 2004, 21(3):82-85.

[4] SMITH B L, DEMETSKY M J. Traffic flow forecasting: Comparison of modeling approaches [J]. Journal of Transportation Engineering, 1997, 123(4):261-266.

[5] GUO X J, ZHU Q. A Traffic Flow Forecasting Model Based on BP Neural Network[C]. Shenzhen:2009 2nd International Conference on Power Electronics and Intelligent Transportation System(PEITS), 2009:311-314.

[6] XIN G, DENG F Q. Short-term prediction of intelligent traffic flow based on BP neural network and ARIMA model[C]. Henan:2010 International Conference on E-Product, E-Service and E-Entertainment(ICEEE), 2010:1-4.

[7] 翁小雄，谭国贤，姚树申. 城市交叉口交通流特征与短时预测模型[J]. 交通运输工程学报，2006, 6（2）:103-107.

[8] 许可. 公路隧道交通数据融合技术的研究[D]. 重庆:重庆交通大学, 2010.

[9] 王静静, 林海伦. 基于路网划分的交通流数据处理模型[J]. 现代计算机（专业版），2012:3-8.

[10] 向红艳, 朱顺应. 小流量下短时交通量预测最佳窗口长度与时间间隔[J]. 武汉理工大学学报:交通科学与工程版, 2006, 30(4):626-628.

[11] 杨兆生，王媛，管青. 基于支持向量机方法的短时交通流预测方法[J]. 吉林大学学报（工学版），2006, 36（6）:881-884.

[12] Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other kernel-based learning methods [M]. Cambridge university press, 2000.

[13] 邓乃扬，田英杰. 数据挖掘中的新方法-支持向量机[M]. 北京：科学出版社，2004.

[14] Chapelle O, Vapnik V, Bousquet O, et al. Choosing multiple parameters for support vector machines[J]. Machine learning, 2002, 46(1-3):131-159.

[15] 刘昌平，范明钰，王光卫. 基于梯度算法的支持向量机参数优化方法[J], 控制与决策，2008, 23（11）:1291-1295.

[16] Kirkpatrick S, Vecchi M P. Optimization by simulated annealing [J]. Science, 1983, 220(4598): 671-680.

[17] YANG Y N, LU H P. Short-term Traffic Flow Combined Forecasting Model Based on SVM[C]. Chengdu: 2010 International Conference on Computational and Information Sciences (ICCIS), 2010:262-265.

[18] Lohr S. The age of big data [J]. New York Times, 2012, 11.

[19] Lynch C. Big data: How do your data grow? [J]. Nature, 2008, 455(7209): 28-29.

[20] White T. Hadoop: The Definitive Guide: The Definitive Guide [M]. O'Reilly Media, 2009.

[21] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//ACM SIGOPS Operating Systems Review. ACM, 2003, 37(5): 29-43.

[22] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[23] Platt J. Sequential minimal optimization: A fast algorithm for training support vector machines [J]. 1998.

[24] Cao L J, Keerthi S S, Ong C J, et al. Parallel sequential minimal optimization for the training of support vector machines [J]. Neural Networks, IEEE Transactions on, 2006, 17(4): 1039-1049.

[25] Graf H P, Cosatto E, Bottou L, et al. Parallel Support Vector Machines: The Cascade SVM[C]//NIPS. 2004: 521-528.

[26] LI S S. Implementing Short-term Traffic Flow Forecasting Based on Multipoint WPRA with MapReduce[C]. Suzhou: 2012 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA), 2012:287-291.

[27] Chang C C, Lin C J. LIBSVM: a library for support vector machines [J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 27.

[28] Vanajakshi L, Rilett L R. A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed[C]//Intelligent Vehicles Symposium, 2004 IEEE. IEEE, 2004: 194-199.

[29] Gillick D, Faria A, DeNero J. Mapreduce: Distributed computing for machine learning [J]. Berkley (December 18, 2006), 2006.

攻读硕士学位期间发表学术论文情况

周常胜，姚卫红. 基于 SA-SVR 的短时交通流预测. 大连理工大学网络学刊.2014. （本硕士学位论文第 3 章）

致　谢

转眼三个春秋的轮回，回首我在大连理工大学攻读硕士学位的时光，一情一景仍历历在目，心中感恩之情尤甚。感恩自己能够在如此优秀的学校深造，感恩能够遇到如此敬业、慈善的姚卫红老师，感恩能够与众多优秀的同窗相逢。

三年来，姚卫红老师给了我很多无私的帮助和指导，让我不断成长，在计算机专业知识和技术水平上有了很大的提高。同时，姚老师一丝不苟的治学精神、和蔼可亲的沟通方式和兢兢业业的工作态度也深深地影响了我，为我以后参加工作树立了标杆和榜样。忘不了姚老师一次一次的叮嘱教导，忘不了姚老师一次一次的耐心解惑，更忘不了姚老师一字一词、不厌其烦的给我改论文。行文至此，由衷的说一声：老师，您辛苦了！科研路上有您的帮助和支持是我最大的幸运，真心感谢姚老师三年来对我的教导和关爱。

郑重感谢实验室的谭国真教授。感谢谭教授不辞辛苦为实验室申请到国家项目；感谢谭教授为我们提供了良好的实验室环境和科研设备；感谢您在科研到路上对我的点拨和督促；感谢您言传身教、身体力行，给我们职业生涯树立了学习的榜样。

感谢实验室的兄弟姐妹们。三年来跟你们在一块儿度过真的很开心，你们刻苦努力的学习态度、力争上游的工作作风都使我受益良多。与你们相识是我人生中一笔宝贵的财富。

最后非常感谢我的家人。感谢父母在我成长路上的言传身教；感谢父母辛勤劳作为我提供读书的机会；感谢父母在我人生迷途中给予的支持和鼓励；更感谢父母在我犯错的时候给予的理解和包容。同时，也感谢我的姐姐，感谢姐姐耐心倾听我心中的苦恼，更感谢姐姐一直以来对父母的照顾，你是我人生中最重要的伙伴。千言万语难叙感恩之心，唯有砥砺前行、修身齐家方可报养育之恩、尽为兄之道。行文至此，与君共勉。

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：Hadoop环境下基于SVR的短时交通流预测

作者签名：周荣胜　　日期：2014 年 6 月 7 日

导师签名：姚卫红　　日期：2014 年 6 月 10 日