|  |
| --- |
| **基于大数据分析的城市道路交通预测** |

|  |
| --- |
| Urban Road Traffic Forecast Based on Big Data Analysis |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学生姓名 | 陈玉梅 | 学号 | | 201411921603 | | |
| 所在学院 | 数学与计算机学院 | | 班级 | | | 金数1146班 |
| 所在专业 | 信息与计算科学 | | | | | |
| 申请学位 | 理学学士 | | | | | |
| 指导教师 | 周永雄 | | 职称 | | 讲师 | |
| 副指导教师 |  | | 职称 | |  | |
| 答辩时间 | 2018年 6 月 3 日 | | | | | |

**目 录**

[摘 要 I](#_Toc19879)

[abstract II](#_Toc9379)

[1 绪论 4](#_Toc31314)

[1.1 论文的研究背景 4](#_Toc2956)

[1.2 论文的研究意义和研究过程 4](#_Toc31352)

[2 机器学习几大分析算法的比较 6](#_Toc31128)

[2.1决策树 6](#_Toc15740)

[2.1.1决策树的简介 6](#_Toc32568)

[2.1.2决策树算法原理 6](#_Toc24123)

[2.2 支持向量机 8](#_Toc12093)

[2.2.1 支持向量机的简介 8](#_Toc8710)

[2.2.2 支持向量机的原理和应用 9](#_Toc18543)

[2.3 逻辑回归........................ 11](#_Toc2133)

[2.3.1 逻辑回归的简介 11](#_Toc24818)

[3 车流量数据处理 11](#_Toc14623)

[3.1 城市道路车流量原始数据 11](#_Toc18871)

[3.2 城市道路车流量数据预处理 12](#_Toc20229)

[4 城市道路车流量模型训练 13](#_Toc21128)

[4.1 关于城市道路车流量的模型训练方法 13](#_Toc18093)

[4.2城市道路车流量模型训练过程 14](#_Toc6571)

[4.3 城市道路车流量预测模型正确率检测 19](#_Toc17480)

[4.4 预测结果 20](#_Toc514)

[5 城市道路车流量预测模型优化 20](#_Toc11155)

[6 结论 20](#_Toc29779)

[鸣 谢 21](#_Toc16560)

[参考文献 22](#_Toc17138)

[附 录 23](#_Toc13457)

# 摘 要

随着电子监控的普及，交通路网基础建设发展迅速，城市到处密布了监控车辆进出的卡口，每个卡口下面有很多个摄像头，形成一个网状、非线性的交通路网电子监控系统。城市道路车流量随着时间段，道路不同，节假日或者工作日受到很多不同的影响。

统计车流量主要有路口监控技术和车辆GPS定位技术。一个是在路口设置检测点，安装摄像头，可以得到交通流量，车辆通过的时间和其他车辆的信息，速度。移动定位技术是基于车辆的GPS进行定位，可以获得车辆的轨迹，行程，坐标等。本文主要分析的是路口监控技术下的车流量分析。国家交通局在很多路口都会遍布一些摄像头，进行拍照采集数据。像一些小区里面也开始有一些停车系统，上面也有车牌识别这一项功能。这些电子监控已经远远不能满足人类的要求。一线城市如北上广深客流量依旧暴增，每逢工作日上班堵车，节假日也会塞车，导致交通问题日益突出。因此交通问题渐渐的现出人类的视线，使得大家不得不开始重视这个问题，开始规划自己的出行路线。因此城市道路交通预测开始变得有意义。

本篇论文主要讨论了下面几个部分：

第一，对这些数据进行预处理后，我们可以分析得出卡口通过车辆的平均速度。如果车辆通过的平均速度快的话，则我们认为当下道路通畅，如果车辆通过的平均速度慢的话，我们认为当下道路拥挤。

第二，讨论了机器学习里面的几大分类算法的比较和分析，以及为什么使用了逻辑回归

第三，研究了道路预测使用到的算法，使用逻辑回归开始训练模型，并且对训练模型准确率进行了验证。

本文的模型是针对某条道路的数据训练出来的，因为城市的道路错交复杂，每一条道路的交通情况都不尽相同，并且受到很多周围的因素的影响。这条道路周围如果是产业密集型的话，车辆通过也会较多。道路车流量也会受到周围道路的影响，比如周围道路出现了交通拥挤，也会相应的影响到本条道路。所以道路预测应该每一条路进行采集数据来进行训练一个模型，本文使用逻辑回归进行多分类。在我们经常中经过会看到天气预报说今天90%的可能会下雨，那么90%可能会下雨表示今天90%的时间会下雨吗？当然不是。其中意思是说在过去这样的天气中，假设有一百天是这样的天气，其中90天下雨了。本文运用在这条道路上的车流量中的研究方法和研究思路，完全可以从本文中剥离出来，运用在其他道路上的预测和其他项目中。

关键词：道路交通预测；机器学习；LR（逻辑回归）

# **abstract**

With the popularity of electronic monitoring, the infrastructure of the traffic network has developed rapidly. Cities are everywhere to monitor the entrance and exit of vehicles. There are many cameras below each intersection, forming a network-like, non-linear electronic monitoring system for traffic networks. The traffic volume of urban roads varies with time, roads, holidays or working days are affected by many different influences.

Statistics traffic flow mainly includes intersection monitoring technology and vehicle GPS positioning technology. One is to set up a detection point at the intersection, install the camera, you can get traffic flow, vehicle passing time and other vehicle information, speed. Mobile positioning technology is based on the vehicle's GPS positioning, you can get the vehicle's trajectory, travel, coordinates and so on. This article mainly analyzes the traffic flow analysis under the intersection monitoring technology. The National Transportation Bureau will have a number of cameras at many intersections for taking pictures and collecting data. For example, there are some parking systems in some communities, and there is also a license plate recognition function. These electronic surveillances are far from meeting human requirements. In the first-tier cities such as the northern and southern GuangZhou、ShenZhen passenger traffic, the number of passenger traffic continues to increase. Every traffic jam on workdays and traffic jams on holidays will cause traffic problems to become increasingly prominent. Therefore, the problem of transportation gradually emerged from the sight of human beings, making everyone have to start paying attention to this issue and start planning their own travel routes. Therefore, the urban road traffic forecast begins to make sense.

In this paper we mainly discuss the following sections:

First, after preprocessing these data, we can analyze the average speed of vehicles passing through intersections. If the average speed of passing vehicles is fast, we can think that the current road is open, and if the average speed of vehicles passing through is slow, we think that the current road is crowded.

Second, it discusses the comparison and analysis of several major classification algorithms in machine learning, and why logistic regression was used.

Thirdly, the algorithm used in road prediction was studied, logistic regression was used to start the training model, and the accuracy of the training model was verified.

The model of this paper is trained on the data of a certain road. Because the roads of the city are misunderstood and complicated, the traffic conditions of each road are different and affected by many surrounding factors. If the area around this road is industrially intensive, more vehicles will pass. Road traffic volume can also be affected by surrounding roads. For example, traffic congestion on surrounding roads will also affect this road accordingly. Therefore, road prediction should collect data for each road to train a model. This article uses logistic regression for multiple classification. As we often go through, we can see that the weather forecast says that 90% of today's may rain, then 90% may rain, it means that 90% of the time will rain today? of course not. The implication of this is that in such weather in the past, it was assumed that there were one hundred days of such weather, of which 90 days were raining. The research methods and research ideas used in this paper for traffic flow on this road can be completely separated from this article and used in other road forecasts and other projects.

**Keywords:** Road Traffic Prediction; Machine Learning; LR (Logical Regression)

**基于大数据分析的城市道路交通预测**

信息与计算科学，201411921603，陈玉梅

指导教师：周永雄

# 绪论

## 论文的研究背景

从很久很久的远古时代，人们就开始发明了一些交通工具，比如马车，牛车，随着人类文明和科技的进步，现在的出行工具越来越方便快捷，越来越多样化，高铁、轮船、公交车、地铁、飞机等，我们出行可以选择的交通工具也变得丰富多彩。现在每家每户几乎都会有一些出行工具，比如小汽车、电动车这些，加上城市里面的公交车，大卡车等等，几乎每条道路上都会看到各式各样的交通工具。步行和自行车的慢行交通，并不会造成什么交通拥挤，在这个到处都是共享单车的时代，也意味着以后的共享经济会越来越收到关注，现在开始已经有了共享宝马。几乎每个在城市里打拼的年轻人都会有一个梦想，要么就是买房买车，要么就是精神食粮，旅行摄影吉他等等。随着大家买车的欲望越来越强烈，道路上的车越来越多，一些城市北京上海广州深圳，每一个十字路口都会设置一个红绿灯，这样不仅仅为了保障行人的安全，还可以让车辆达到分流的目的。因为车辆越来越多，开始有了越来越多的停车场，很多道路上也开始有了一些僵尸车停在路边，一停就是一天，因此关于城市交通的管理变得越来越严峻。

关于交通拥堵的新闻大概大家都屡见不鲜了，某某城市交通路段拥挤，还有一些城市有专门的一些交通电台，可以从那里知道一些交通情况，道路拥挤严重影响了人们的出行生活，一些上班族本来可以晚点出门的，结果这些时间都被浪费在了上班堵车的路上，甚至有些因为交通拥挤出现的交通也不在少数，汽车追尾等等。交通出行跟我们每一个人息息相关，不管是步行还是自行车还是公交车、汽车等等。在每年的春运、节假日想必每一个人都不会陌生，交通出行简直就是一整个节假日就在路上，花在景点上的时间都没有花在路上的时间多，因此关于城市交通的管理变得越来越严峻。

## 论文的研究意义和研究过程

本文基于以上的背景讨论了基于大数据下的车流量的分析和道路拥挤情况的预测。目标是进行研究道路流量，进行车辆流量的预测，来进行城市规划的管理。本文主要使用的技术是道路监控技术，也就是对每个卡口号进行拍摄，进行读取数据。对车牌号码的识别有两种方式，一种是对车牌上的识别卡信息进行采集识别，只要摄像头的焦点对准车牌号上的小白点就能马上识别出来。二是对车辆的车牌号码进行拍摄图片，进行图像识别划分一个个字母进行匹配识别。两者最终得到的信息都是一样的。现在的图像识别技术日益完善，很多拍摄的一些图片都可以进行图像识别。

车牌识别技术[7]要求能够对运动状态中的汽车从一整张背景种抽取出来，通过图像拍摄，找到车的位置，然后再寻找到车牌的位置，通过图像预处理和特征提取等阶段，进行车牌上的字符切割识别等技术，识别车辆牌号、颜色、车型等信息，目前最新的车牌识别技术水平的对字母和数字识别率可以达到99.7%，汉字的识别率可达到99%[9]。

现在城市的卡口下的摄像头仿佛就是交通监管的第三只眼一样。在很多地方都运用广泛，俗话说，"天网恢恢疏而不漏",整个城市的监控系统就像一只大网一样遍布了整个城市。接下来介绍了一下城市监控系统的几种应用场景。

第一，监测报警。对于被通缉车辆、违规车辆、肇事车辆等加入车辆黑名单，将读取到的车牌号码与车辆黑名单进行对比。一旦发现指定黑名单车辆将进行报警。通过车牌监控可以大大提高执法效率。

第二，超速违章罚款。在路上设置测速监测点，抓拍车辆通过识别车牌号码，系统自动判定如果超过一定的速度时自动进行罚款。将违章车辆的牌照号码及图片发往各出口；在各出口设置处罚点，用车牌识别设备识别通过车辆并将号码与已经收到的超速车辆的号码比对，一旦号码相同即启动警示设备通知执法人员处理。

第三，车辆出入管理。将车牌识别设备安装于出入口，记录车辆的[牌照](https://baike.baidu.com/item/%E7%89%8C%E7%85%A7" \t "_blank)号码、出入时间，并与[自动门](https://baike.baidu.com/item/%E8%87%AA%E5%8A%A8%E9%97%A8" \t "_blank)、栏杆机的控制设备结合，实现车辆的自动管理。应用于停车场可以实现自动计时收费，也可以自动计算可用[车位](https://baike.baidu.com/item/%E8%BD%A6%E4%BD%8D" \t "_blank)数量并给出提示，实现停车收费自动管理节省人力、提高效率。应用于[智能小区](https://baike.baidu.com/item/%E6%99%BA%E8%83%BD%E5%B0%8F%E5%8C%BA" \t "_blank)可以自动判别驶入车辆是否属于本小区，对非内部车辆实现自动计时收费。在一些单位这种应用还可以同车辆调度系统相结合，自动地、客观地记录本单位车辆的出车情况，[车牌识别](https://baike.baidu.com/item/%E8%BD%A6%E7%89%8C%E8%AF%86%E5%88%AB" \t "_blank)管理系统采用了车牌识别技术，达到不停车、免取卡，有效提高车辆出入通行效率[15]。

第四，高速公路收费管理。在高速路的各个出入口安装车牌识别设备，车辆驶入时识别车辆牌照将入口资料存入收费系统，车辆到达出口时再次识别其牌照并根据牌照信息调用入口资料，结合出入口资料实现收费管理。这种应用可以实现自动计费并可防止作弊，避免了应收款的流失。

随着现在的技术越来越成型，车辆检测技术也越来越成熟。车辆检测可以采用埋地线检测、红外检测、[雷达检测技术](https://baike.baidu.com/item/%E9%9B%B7%E8%BE%BE%E6%A3%80%E6%B5%8B%E6%8A%80%E6%9C%AF" \t "_blank)、视频检测等多种方式。其中采用视频检测可以避免破坏路面、不必附加外部检测设备、不需矫正触发位置、节省开支，而且更适合移动式、便携式应用的要求。

接下来简单介绍一下本文的处理流程：

步骤一：通过车牌识别拿到原始数据之后，仔细分析每个字段之间的含义，为下面数据预处理做准备

步骤二：数据预处理，对表中一些空字段和无效字段进行处理。其中无效字段就是处理数据中不需要的字段，可以进行过滤掉，通过sql语句进行筛选

步骤三：机器学习几大分类算法的比较和分析

步骤四：训练城市道路车流量的模型

步骤五：检验城市道路车流量预测模型的正确性

步骤六：城市道路车流量预测模型优化

# 机器学习几大分析算法的比较

机器学习是人工智能的一个分支，可以赋予机器一种新的能力。机器学习是一门多领域交叉学科，涉及到统计学、概率论、算法复杂度等。用来研究计算机怎么模拟或者实现人类的学习行为，来获得新的知识，不断改善自身的性能。机器学习是人工智能的核心，使计算机具有智能的根本途径。

机器学习内容包含很多，按照不同的分类规则可以有不同的分类。机器学习基于学习策略的分类，可以分为机械学习、示教学习、演绎学习、类比学习。基于所获取知识的表现形式进行分类，学习系统获取的知识可能有：行为规则、问题求解策略、物理对象描述、各种分类及其其它用于任务实现的知识类型，可以分为代数表达参数、决策树、形式文法、形式逻辑表达、产生式规则、框架和模式、图和网络、神经网络、计算机程序和其它的过程编码、多种表达形式的组合。

机器学习按照学习的形式还可以分为有监督的机器学习和无监督的机器学习。有监督学习是指对有概念标记的训练样本进行分类学习，来对测试集的数据来进行标记分类预测屋监督机器学习的标记和分类都是未知的，因此训练样本的歧义性很高。

机器学习中用于分类的算法有以下几种，包括决策树（Decision Tree）、朴素贝叶斯、k-近邻（KNN）、逻辑回归、支持向量机、随机森林、SVM等。接下来我们来了解一下机器学习用于分类的这几个算法。

## 2.1决策树

### 2.1.1决策树的简介

决策树[12]是一种基本的分类回归方法，它的模型呈树形结构[8]。表示基于特征对实例进行分类的过程。决策树用来预测的对象都是固定的，其中每一个分支都代表一个分支向量，每个节点代表一个输出结果或者分类。沿着根节点进行出发，要么是要么否，不同的选择决定去到不同的分支直到分支不可再分为止，也就是达到叶子节点，这个时候也就是我们需要的结果。这个判断是或者否的节点也叫逻辑判断，表示该属性的某个分支，供下一步判断判定。

### 2.1.2决策树算法原理

接触决策树，我们先来了解一下决策树的理论基础--信息熵。

信息熵[6]，是香农在1948年提出来量化信息的信息量的，指的是对事件中不确定的信息的度量。像我们经常说的，这个事情的信息量很大或者这个事情的信息量很少，就可以用信息熵来表示，可以理解成一种特定信息出现的概率。一个事件中，信息熵越大，不确定性越大，包含的信息越多。在信源中，考虑的是这个信源中所有可能会发生情况的不确定性，若信源符号有n种取值：,,...,,对应概率为：,,...,，且各种符号出现彼此独立，这个时候信源的平均不确定性为单个符号的不确定性的平均值，可称为信息熵，即

，

底为2，单位为比特。

我们进行构造决策树的时候，为了让信息越来越明显，也就是说到最后一步的时候，信息应该是已经清晰了的，而不是模糊不确定的，因此这个时候我们构造决策树，信息熵就有利于我们判断我们构造这个决策书是否合理。

在我们经常做决策的时候，都会进行提问自己，我做这件事情是否是我喜欢的，如果是，我要进行下一步判断，是否会浪费很多我的时间，如果是，我还应不应该继续做下去，这个时候决策树可以帮助我们很好的进行下去，这也是我们日常生活种很多事情都需要运用上的一种思维。

下面我们来看一个例子：

假设一家出租车公司，需要分析用户什么时候会进行用车，拿到的数据如下

表1 用户信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 顾客信息 | 天气 | 衣着 | 购物 | 周末 | 温度>90 |  |
| 顾客1 | 下雨 | 正式的 | 是 | 是 | 否 | 步行 |
| 顾客2 | 下雪 | 休闲的 | 否 | 否 | 是 | 开车 |
| 顾客3 | 晴朗 | 休闲的 | 否 | 否 | 否 | 步行 |
| 顾客4 | 下雨 | 正式的 | 是 | 是 | 否 | 步行 |
| 顾客5 | 晴朗 | 休闲的 | 是 | 否 | 是 | 步行 |
| 顾客6 | 晴朗 | 正式的 | 否 | 否 | 否 | 开车 |
| 顾客7 | 下雪 | 正式的 | 否 | 是 | 是 | 开车 |
| 顾客8 | 下雨 | 正式的 | 是 | 否 | 是 | 步行 |
| 顾客9 | 下雪 | 休闲的 | 是 | 是 | 是 | 开车 |
| 顾客10 | 晴朗 | 正式的 | 否 | 否 | 是 | 开车 |

这个时候我们需要构造决策树来判断用户是否会开车，对于那些需要开车的人，显然就是我们的潜在用户。

对于这个的处理我们可以假设为在下雨的前提下，大部分人都选择了步行，那么就可以记录叶子节点下雨时大家选择步行，为叶子节点，判断条件为是否下雨。下雨的选项为晴朗，衣着时正式的人都选择了开车，这个时候叶子节点也为开车。

那么这个时候我们如何确定是衣着还是天气作为作为根节点的。这个时候就需要用上我们的信息熵了。用以上数据来判断是否为根节点的过程。

在未对数据做任何处理的情况下，需要预测结果当中有两种选择：开车与步行。开车出现了5次，步行出现了5次，这个时候的信息熵为：



第一种情况，如果我们按照天气来划分，“下雪”条件中的分布是（开车 3；步行 0），“下雨”条件中的分布是（开车 0；步行 3），“晴朗”条件中的分布是（开车 2；步行 2）。三个条件的熵分别是

，

，

。

由于10条数据被分为了三组数据，所以分割后的信息熵平均值为0.3\*0 + 0.3\*0 + 0.4\*1 = 0.4。

第二种情况，如果我们按照衣着来划分，“正式的”条件的分布是（开车 3；步行 3），“休闲的”条件中的分布是（开车 2；步行 2）。两个条件的熵分别是





由于10条数据被分为了两组数据，所以分割后的信息熵平均值为 0.6\*1 + 0.4\*1 = 1。

这个时候我们选择衣着的时候，它的信息量比按照天气来分的信息量少了很多，也就是说，这个时候信息就变得比较确定了。所以我们可以根据这个信息熵来划分决策树的根节点是哪个判断条件。以此类推，因此这个时候的决策树就可以确定出来了，如下图所示。

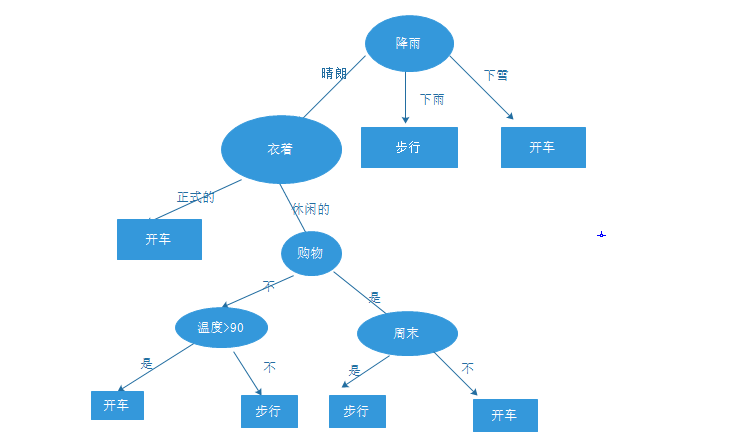


图1 关于用户出行的决策树

## 2.2 支持向量机

### 2.2.1 支持向量机的简介

支持向量机[14]主要应用在解决小样本、非线性及[高维](https://baike.baidu.com/item/%E9%AB%98%E7%BB%B4" \t "https://baike.baidu.com/item/%E6%94%AF%E6%8C%81%E5%90%91%E9%87%8F%E6%9C%BA/_blank)模式识别中。一个支持向量机可以构造一个超平面，或在高或无限维空间，其可以用于分类，回归。

### 2.2.2 支持向量机的原理和应用

假设有两个类别或者特征，这个时候我们需要将他们分开，在一个二维平面中，看起来就是一条直线可以将他们分开，这条线的界定边界就是将这两个类分开。

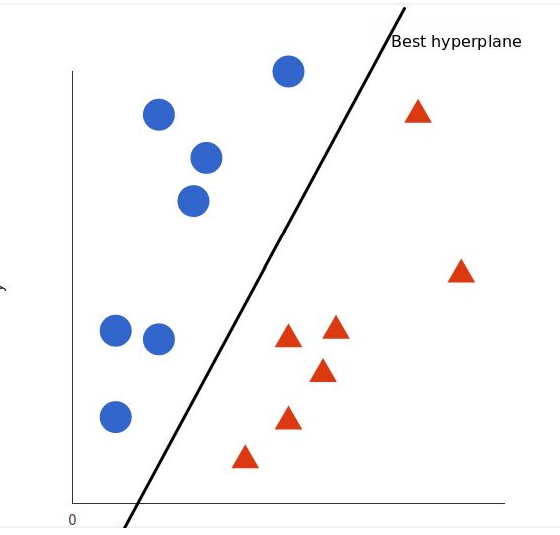


图2 SVM线性可分时的分类

最好的超平面对于SVM来说就是可以最大化两个类别之间的间距，超平面对每个距离最近的元素相隔最远，达到最大化了两个类别之间的间距。

如下图所示：

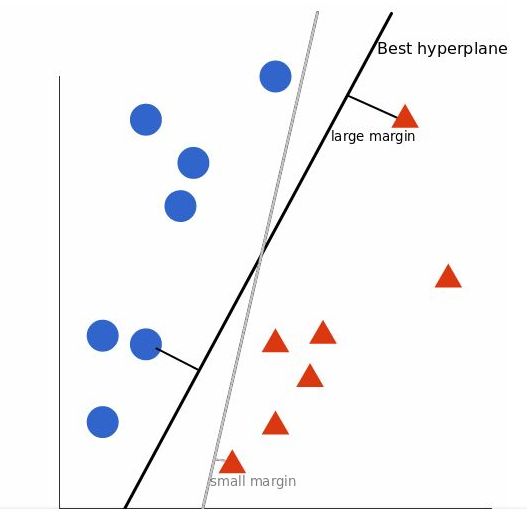


图3 SVM优化后的分类

因为上面的例子很简单，都是一眼就可以看出来是线性可分的，如果遇到了线性不可分的情况，这个时候我们应该怎么办呢。看一下下图：

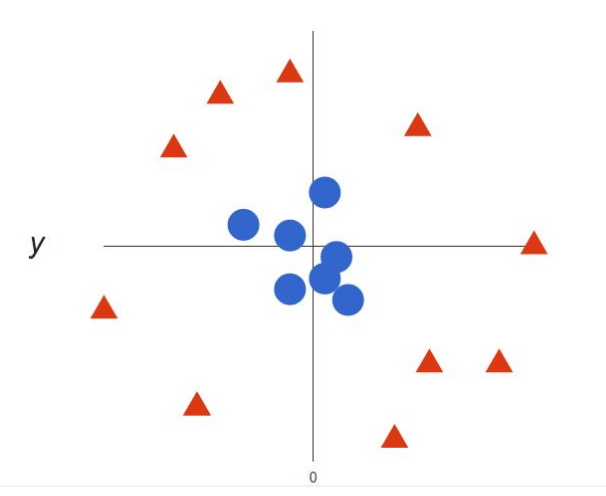


图4 SVM线性不可分时的分类

这个时候直接用一条直线我们是无法区分这两个类出来的，所以我们就需要引入到三维立体了，其中x为横坐标，y为纵坐标，我们引入第三个维度，z坐标，可以假设

就可以得到每个小球的z的高度了，从而得到一个三维空间，这个三维空间如图所示：

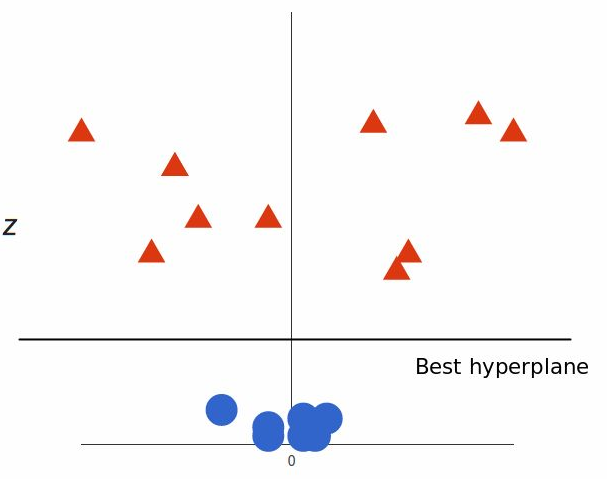


图5 SVM线性不可分时进行分类后的结果显示

这样我们就可以用一个平面把这个类别区分出来。这个过程也叫做升维。

## 2.3 逻辑回归

### 2.3.1 逻辑回归的简介

逻辑回归[11]是一种线性有监督的分类模型，相当于y=f(x)，相当于输入跟输出的关系。用方程式可以表示为：



确定w的过程，就是训练模型的过程。

逻辑回归的模型引入了sigmoid函数映射，是非线性模型，但本质上又是一个线性回归模型，因为除去sigmoid映射函数关系，其他的步骤，算法都是线性回归的。可以说，逻辑回归，都是以线性回归为理论支持的。

# 车流量数据处理

## 3.1 城市道路车流量原始数据

原始数据如下图6所示，其中原始数据中包含的一些字段为无效字段和预留空字段。

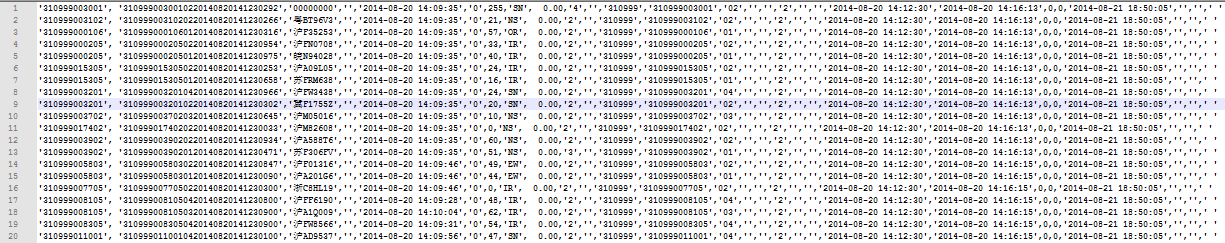


图6 车流量原始数据

摘取其中一条字段展示信息，我们需要的字段的含义如表2所示，其中其他为预留字段或者无效字段。

表2 原始数据及对应的字段信息

|  |  |
| --- | --- |
| 字段值 | 字段含义 |
| 310999000106 | 拍摄的卡口号的ID |
| 3109990001060120140820141230316 | 拍摄的这条记录的ID |
| 沪F35253 | 拍摄的车牌号 |
| 2014-08-20 14:09:35 | 拍摄的时间 |
| 57 | 车辆通过的速度 |
| OR | 车辆通过的方位 |
| 01 | 拍摄的摄像头 |

从图6可知，我们每个字段值与字段值之间都是使用的 ',' 来进行分割，这个时候我们以逗号进行分割将txt文本导入进数据库中，建立一个初始的表：

|  |  |
| --- | --- |
| 01  02 | load data local infile "E:/xiazi/carData.txt"  into table rowData; |

## 3.2 城市道路车流量数据预处理

数据预处理主要包括数据异常值处理和缺失值处理等。其中我们可以看到字段中有很多事缺失值和预留值，这个时候我们需要去除这些预留值和字段缺失值，留下自己想要的字段。

进行数据处理时所需要保留的字段所形成的数据字典如表3所示

表3 数据预处理之后的数据字段

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 字段类型 | 字段含义 |
| 1 | monitor\_id | VARCHAR(100) | 拍摄的卡口号的ID |
| 2 | Id | VARCHAR(100) | 拍摄的这条记录的ID |
| 3 | License\_plate\_number | VARCHAR(10) | 拍摄的车牌号 |
| 4 | Time | VARCHAR(50) | 拍摄的时间 |
| 5 | Speed | INT | 车辆通过的速度 |
| 6 | Direction | VARCHAR(50) | 车辆通过的方位 |
| 7 | camera\_id | VARCHAR(10) | 拍摄的摄像头 |

根据数据字典的建表语句如下所示：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09 | create table carData(  monitor\_id VARCHAR(100) not null,  id VARCHAR(100) not null PRIMARY KEY,  License\_plate\_number VARCHAR(10),  time VARCHAR(50),  speed INT,  direction VARCHAR(50),  camera\_id int  ) row format delimited fields terminated by ','; |

进行数据预处理时我们主要有两种方法。

第一种，我们可以通过sql语句来进行筛选字段提取来获得有效字段，我们所需要的有效字段如表二所示，通过sql语句来写进行数据预处理时的sql语句为：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | INSERT INTO carData (  monitor\_id,  id,  License\_plate\_number,  time,  speed,  direction,  camera\_id  ) SELECT  monitor\_id,  id,  License\_plate\_number,  time,  speed,  direction,  camera\_id  FROM  rowdata  WHERE  License\_plate\_number != '000000'  AND monitor\_id IS NOT NULL  AND camera\_id IS NOT NULL; |

第二种，我们可以直接在代码中体现。直接对原始数据进行切分，比如split，然后放进去一个集合中，在集合中读取我们需要的那几个字段。

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08 | string a[]=text.split(',')  string monitor\_id=a[0]  string id=a[1]  string License\_plate\_number=a[2]  string time=a[4]  int speed=a[6]  string direction=a[7]  string camera\_id=a[13] |

# 城市道路车流量模型训练



## 关于城市道路车流量的模型训练方法

本文我们训练模型的时候使用到的是逻辑回归，逻辑回归实际上也是一个二分类，要么是要么不是。

当然我们也可以把一个多分类看成二分类，把这些分类作为一个大的二分类，判断是某一个大分类里面的时候，在对里面的一个大分类进行小分类。做二分类时，它有一个分类阈值，大于0.5为正例，小于0.5为负例，做三分类时，可以把分类阈值调一下，作为0.33和0.66。



## 4.2城市道路车流量模型训练过程

为了更好的理解我们接下来的模型训练过程，我们先来了解一下路况预测：



图7 路况预测

由图可以看出来我们的路况预测实际上就是一个分类，设严重拥堵为1分类号，拥堵为2分类号，缓行为3分类号，通畅为4分类号。取平均速度为划分这个分类号的依据，当车平均速度达到一定的速度时候，我们可以说道路是通畅的。当然这个时候我们间隔也需要取大一点，来降低偶然性。

由图可得，我们可以把x轴看成一个时间，y看成一个个分类号，严重拥堵设为1，拥堵设为2，缓行设为3，通畅设为4，我们可以把四分类提升到12分类来提升模型的抗干扰能力。我们可以把一个多分类看成二分类，首先我们把通畅和拥堵分为两大类，其中大于60km/h为通畅，小于60为拥堵，当速度小于60时，再进行下一步分类，分为缓行（30-60）和拥堵（0-30）。接下来拥堵继续分为一般拥堵（10-30）和严重拥堵（0-10）

可以进行如下的二分类：

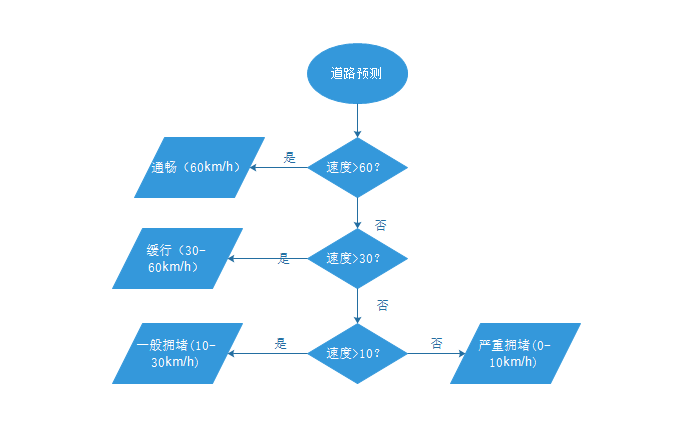


图8 基于逻辑回归的路况分类

接下来我们再来了解一下，现在我们比如要预测湛江路这条路接下来的交通情况怎么样，这个时候我们就需要了解一下湛江路的前五分钟或者前三分钟的交通情况，这个时候湛江路这个的交通情况可能还跟隔壁相邻的路，比如湛江东路，湛江西路的交通情况会相关，这个时候我们采用逻辑回归预测湛江路的20：11分的交通情况，那么x的取值就是湛江东路的20：08，20：09，20：10分的交通情况，还有湛江西路的20：08，20：09，20：10分的交通情况，加上湛江路的20：08，20：09，20：10分的交通情况。我们可以通过下图来进行理解。

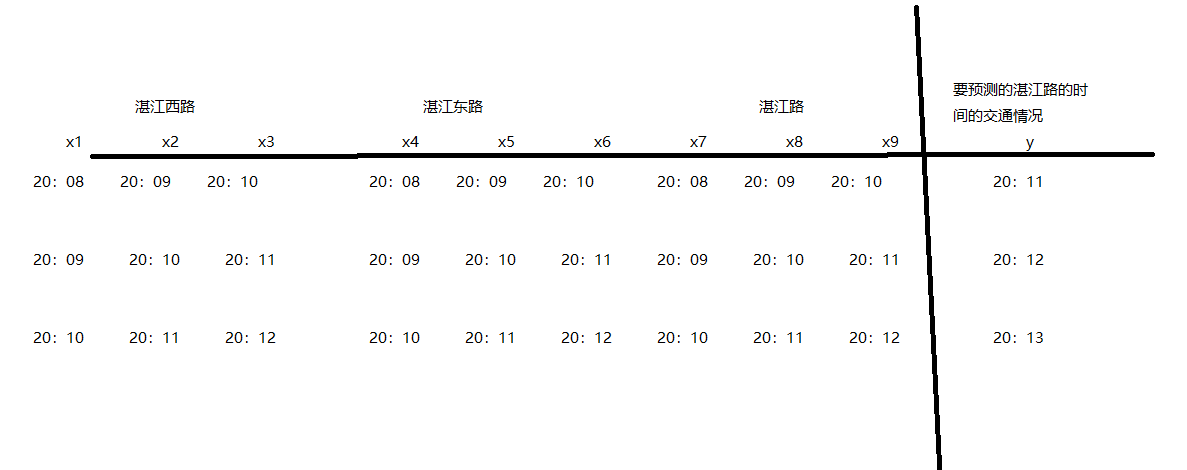


图9 道路车流量预测

其中x表示为我们需要拿到的数据，湛江西路，湛江东路，湛江路前三分钟的交通情况，来预测湛江路下一分钟的交通情况，因为这条路下一分钟的交通情况跟前几分钟的交通情况息息相关。

假设我们是需要用前五个小时的交通情况来预测接下来一个小时的交通情况，也就是要把x换成相应的前五个小时的交通情况。而这个交通情况是根据通过的车辆的总速度除于通过的总车辆，可以得到这一分钟这条路车辆通过的平均速度，从而预测下一分钟湛江路的平均速度。

这个时候我们先看一下求每条路的平均速度的代码，如下：

|  |  |
| --- | --- |
| 01  02 03 04 05 | val carSpeed = events.map(x =>  (x.getString("camera\_id"),x.getInt("speed")))  .mapValues((x:Int)=>(x,1))  .reduceByKeyAndWindow((a:Tuple2[Int,Int], b:Tuple2[Int,Int]) =>{(a.\_1 + b.\_1, a.\_2 + b.\_2)},Seconds(20),Seconds(10)) |

这里计算的是车的总速度，每隔十秒计算过去二十秒的计算时间。计算的时候有一个速度就是有一辆车通过。

|  |  |
| --- | --- |
| 01  02  03 | reduceByKeyAndWindow((a:Tuple2[Int,Int], b:Tuple2[Int,Int]) =>  {(a.\_1 + b.\_1, a.\_2 + b.\_2)},(a:Tuple2[Int,Int], b:Tuple2[Int,Int])  => {(a.\_1 - b.\_1, a.\_2 - b.\_2)},Seconds(20),Seconds(10)) |

这两行的代码我们可以结合下图来进行理解：

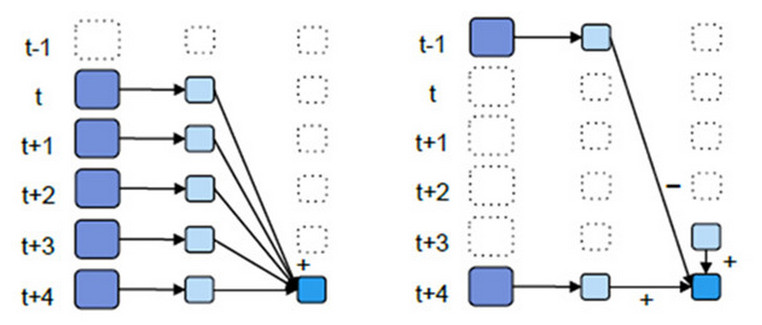


图10 sparkStreaming的窗口函数

本来我们是需要十分钟计算过去二十秒通过的车辆的总速度，这个时候每隔十秒我们需要把最近二十秒通过车辆的所有速度加起来，但是使用了这个优化之后的代码的时候，在接下来的十秒之后，我们可以用上一次计算的数据加上最近十秒通过的车辆的速度，减去之前十秒的车辆通过的速度，也就说少算了之前这一部分已经计算过了的数据。避免了数据的重复计算。

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | carSpeed.foreachRDD(rdd => {  rdd.foreachPartition(partitionOfRecords => {  val jedis = RedisClient.pool.getResource  partitionOfRecords.foreach(pair => {  val camera\_id = pair.\_1 摄像头的ID  val total = pair.\_2.\_1 总速度  val count = pair.\_2.\_2 车辆总数  val now = Calendar.getInstance().getTime()  // create the date/time formatters  val minuteFormat = new SimpleDateFormat("HHmm")  val dayFormat = new SimpleDateFormat("yyyyMMdd")  val time = minuteFormat.format(now)  val day = dayFormat.format(now)  })  })  }) |

这里我们就可以得到车辆的总速度和通过的总车辆数。

得到的数据的格式为（k,(k,v)）,即("20180501\_'310999015605'",("1724","37\_1"))

其中20180501表示时间年月日，310999015605卡口号，1724表示17点24分，37\_1表示速度为总速度为37，车辆总数为1辆。我们的平均速度为总速度除于平均速度，这个时候的平均速度也是37km/h，也就是我们对应的分类号为3。

加下来我们开始训练模型，我们可以把计算出来的平均速度用一个分类号来表示，比如0-10的时候是0分类号，11-20的时候是1分类号，21-30的时候是3分类号，以此类推。我们最后预测出来的结果是一个四分类，也就是说，我们0分类号，1分类号和2分类号其实最后展现出来的可能都是畅通的这个可能，大大增加了模型的容错性。

我们拿到这个摄像头的ID，一条道路只有一个摄像头ID，假设湛江路的摄像头ID为310999003001，湛江南路的摄像头ID为310999003102。剩下的list集合都是这条路和这条路相邻的路，比如湛江路相邻的路为湛江西路310999000106，湛江东路310999000205，湛江北路310999007204。如下图所示：

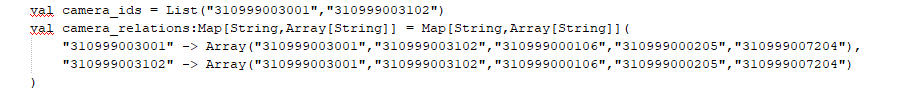


图11 相邻道路

我们可以先获取当前系统的时间

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | val temp = camera\_ids.map({ camera\_id =>  val hours = 5  val nowtimelong = System.currentTimeMillis();  val now = new Date(nowtimelong)  val day = dayFormat.format(now)  val list = camera\_relations.get(camera\_id).get  val relations = list.map({ camera\_id =>  (camera\_id, jedis.hgetAll(day + "\_'" + camera\_id+"'"))  })  val dataSet = ArrayBuffer[LabeledPoint]()  for(i <- Range(60\*hours,0,-1)){  val features = ArrayBuffer[Double]()  val labels = ArrayBuffer[Double]()  for(index <- 0 to 2){  val tempOne = nowtimelong - 60 \* 1000 \* (i-index)  val d = new Date(tempOne)  val tempMinute = minuteFormat.format(d)  val tempNext = tempOne - 60 \* 1000 \* (-1)  val dNext = new Date(tempNext)  val tempMinuteNext = minuteFormat.format(dNext)  for((k,v) <- relations){  val map = v  if(index == 2 && k == camera\_id){  if (map.containsKey(tempMinuteNext)) {  val info = map.get(tempMinuteNext).split("\_")  val f = info(0).toFloat / info(1).toFloat  labels += f  }  }  if (map.containsKey(tempMinute)){  val info = map.get(tempMinute).split("\_")  val f = info(0).toFloat / info(1).toFloat  features += f  } else{  features += -1.0}  }  }  } |

index从0-2开始遍历，表示遍历了前三分钟的时间。我们前面的labels存储了平均速度，这个时候需要对labels进行处理，也就是变成分类号。

|  |  |
| --- | --- |
| 01  02  03  04  05  06 | if(labels.toArray.length == 1 ){  val label = (labels.toArray).head  val record = LabeledPoint(if ((label.toInt/10)<10) (label.toInt/10) else 10.0, Vectors.dense(features.toArray))  dataSet += record  } |

处理后的结果为0-10时候，结果为0，11-20的时候分类号为1，以此类推。把数据传进来开始使用逻辑回归训练模型，对数据进行随机切分，0.9的作为训练集的数据，0.1作为测试集的数据，对逻辑回归进行一个十二分类。

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | val data = sc.parallelize(dataSet)  val splits = data.randomSplit(Array(0.9, 0.1), seed = 11L)  val training = splits(0)  val test = splits(1)  if(!data.isEmpty()){  val model = new LogisticRegressionWithLBFGS()  .setNumClasses(11)  .setIntercept(true)  .run(training)  val predictionAndLabels = test.map { case LabeledPoint(label, features) =>  val prediction = model.predict(features)  (prediction, label) //返回一个分类号和真实的分类号  }  } |

## 4.3 城市道路车流量预测模型正确率检测

接下来计算模型的准确率

|  |  |
| --- | --- |
| 01  02  03  04 | val metrics = new MulticlassMetrics(predictionAndLabels)  //计算模型的准确率  val precision = metrics.precision  println("Precision = " + precision) |

判断模型准确率大于0.8保存模型

|  |  |
| --- | --- |
| 01  02  03  04  05  06 | if(precision > 0.8){  val path = "E://xiazi/ "+camera\_id+"\_"+nowtimelong  model.save(sc, path)  println("saved model to "+ path)  jedis.hset("model", camera\_id , path)  } |

## 4.4 预测结果

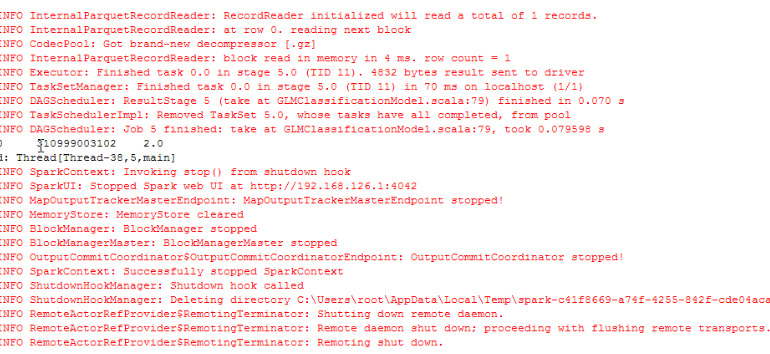


图12 预测结果

有图可知我们预测出来的分类号为2.0，因为我们做的是十二分类，这个时候0-2都是第一类都是为拥堵，他的平均速度为22.5km/h，拥堵也是第一类。所以模型预测正确。

# 城市道路车流量预测模型优化

为了是模型可以预测更准确，我们进行了如下的优化：

第一，预测道路是否拥挤的时候，采用一分钟的数据具有偶然性，如果这个时候同时通过两辆车，一辆车很快180km/h，一辆车较慢速度通过60km/h，这个时候对这两辆车求平均值会出现太大的偶然性，导致不准确。所以这个时候求实时道路预测，预测这个时间点的车辆通过速度的时候，比如17：18分，就不能只是单纯的计算这一分钟，我们可以把时间的跨度拉长一点，使用十分钟的数据可以降低这个偶然性。所以这个时候可以使用到sparkstreaming的窗口函数。sparkstreamin可以进行微批处理，比如每五分钟计算过去半个小时的平均速度，这个时候这个平均速度就代表这一分钟的平均速度。这样以降低偶然性。使结果预测更加准准确。

第二，当我们对这个进行分类的时候，虽然我们只需要把这个分类当成四分类就行了，但是为了让模型更加准确，这个时候我们分成十几类，更能增加模型的容错性，如果模型出现点偏差，当我们预测出来分类号为1和2的时候，只做四分类的时候，这个时候就变成了两个不同的分类了，但是变成十几分类的时候，这个时候1和2都是拥堵了。大大增加了模型的容错性。

# 结论

模型不是写出来就是最好的，都是需要经过不断的优化，不断的改进。只有当我们深入去了解一件事情的本质的时候，才能对他进行更好的运用。

就像我们每天经过红绿灯和电梯，但是大部分人都只是烦恼于红绿灯还是红的，电梯还是没来，不如静下心来感受一下生活，思考一下红绿灯算法和电梯的调度算法，也许会有不同的收获呢。

# 鸣 谢

在海大度过了人生中最美好的大学四年的生活，在这里收获了很多，不仅仅收获了知识，还学习到了各位老师们的思维方法还有一些待人处事的学问，学会了用辩证的思维去思考一个问题，而不是像以前一样片面的看待问题。

首先我要感谢我的母校，非常浓厚的学术氛围，让我受益匪浅。

感谢我的专业老师，是他们传道授业解惑，让我们能接触到更广的知识面，了解到他们对知识的不同见解。

感谢我的导师，对我的毕业论文一直很关心，一直在背后默默的支持我们，在写毕业论文的时间，不断的给予鼓励、监督、指导。

感谢我的大学同学们，正是因为有这么一群优秀的同学，才能让我不断进步，不断成长。

# 参考文献

[1]刘丽娜. 城市道路交通流量短时预测的研究[D]. 北京邮电大学, 2010.

[2]范腾腾. 城市道路交通流量短时预测研究[D]. 北京交通大学, 2012.

[3]董春娇, 邵春福, 李娟,等. 基于混沌分析的道路网交通流短时预测[J]. 系统工程学报, 2011, 26(3):340-345.

[4]李晓菲. 数据预处理算法的研究与应用[D]. 西南交通大学, 2006.

[5]王晓华. SPARK MLLIB机器学习实践[M]. 清华大学出版社, 2015.

[6]唐朝京, 雷菁. 信息论与编码基础[M]. 国防科技大学出版社, 2003.

[7]王刚, 冀小平. 基于MATLAB的车牌识别系统的研究[J]. 电子设计工程, 2009, 17(11):72-73.

[8]姚凯飞. 基于聚类后弱分类器自适应选择的AdaBoost算法改进[D]. 上海交通大学, 2016.

[9]於毅. 城市道路交通状态判别方法研究[D]. 北京交通大学, 2007.

[10]薄树奎, 孙新德, 丁琳. 一种基于彩色图像分割的车牌检测方法[J]. 计算机科学, 2009, 36(4):261-263.

[11]王珏, 周志华, 周傲英. 机器学习及其应用[M]. 清华大学出版社, 2006.

[12]Robert C. Machine Learning, a Probabilistic Perspective[M]. MIT Press, 2012.

[13]Mitchell. Machine Learning[M]. McGraw-Hill, 2003.

[14]Cauwenberghs G, Poggio T. Incremental and decremental support vector machine learning[C]// International Conference on Neural Information Processing Systems. MIT Press, 2000:388-394.

[15]吴凌峰. 基于DSP芯片的车牌识别系统[D]. 浙江大学. 2006

# 附 录

TrainLRwithLBFGS.scala

package com.ic.traffic.streaming

import java.text.SimpleDateFormat

import java.util

import java.util.{Date}

import org.apache.spark.mllib.classification.LogisticRegressionWithLBFGS

import org.apache.spark.mllib.evaluation.MulticlassMetrics

import org.apache.spark.{SparkContext, SparkConf}

import org.apache.spark.mllib.linalg.Vectors

import org.apache.spark.mllib.regression.LabeledPoint

import org.apache.spark.mllib.util.MLUtils

import scala.collection.mutable.ArrayBuffer

import scala.Array

import scala.collection.mutable.ArrayBuffer

object TrainLRwithLBFGS {

val sparkConf = new SparkConf().setAppName("ZhanJiang traffic").setMaster("local[4]")

val sc = new SparkContext(sparkConf)

val dayFormat = new SimpleDateFormat("yyyyMMdd")

val minuteFormat = new SimpleDateFormat("HHmm")

def main(args: Array[String]) {

val jedis = RedisClient.pool.getResource

jedis.select(1)

val camera\_ids = List("310999003001","310999003102")

val camera\_relations:Map[String,Array[String]] = Map[String,Array[String]](

"310999003001" -> Array("310999003001","310999003102","310999000106","310999000205","310999007204"),

"310999003102" -> Array("310999003001","310999003102","310999000106","310999000205","310999007204")

)

val temp = camera\_ids.map({ camera\_id =>

val hours = 5

val nowtimelong = System.currentTimeMillis();

val now = new Date(nowtimelong)

val day = dayFormat.format(now)

val list = camera\_relations.get(camera\_id).get

val relations = list.map({ camera\_id =>

// println(camera\_id)

(camera\_id, jedis.hgetAll(day + "\_'" + camera\_id+"'"))

})

relations.foreach(println)

// organize above records per minute to train data set format (MLUtils.loadLibSVMFile)

val dataSet = ArrayBuffer[LabeledPoint]()

// start begin at index 3

for(i <- Range(60\*hours,0,-1)){

val features = ArrayBuffer[Double]()

val labels = ArrayBuffer[Double]()

// get current minute and recent two minutes

for(index <- 0 to 2){

val tempOne = nowtimelong - 60 \* 1000 \* (i-index)

val d = new Date(tempOne)

val tempMinute = minuteFormat.format(d)

val tempNext = tempOne - 60 \* 1000 \* (-1)

val dNext = new Date(tempNext)

val tempMinuteNext = minuteFormat.format(dNext)

for((k,v) <- relations){

// k->camera\_id ; v->speed

val map = v

if(index == 2 && k == camera\_id){

if (map.containsKey(tempMinuteNext)) {

val info = map.get(tempMinuteNext).split("\_")

val f = info(0).toFloat / info(1).toFloat

labels += f

}

}

if (map.containsKey(tempMinute)){

val info = map.get(tempMinute).split("\_")

val f = info(0).toFloat / info(1).toFloat

features += f

} else{

features += -1.0

}

}

}

if(labels.toArray.length == 1 ){

val label = (labels.toArray).head

val record = LabeledPoint(if ((label.toInt/10)<10) (label.toInt/10) else 10.0, Vectors.dense(features.toArray))

// println(record)

dataSet += record

}

}

dataSet.foreach(println)

println(dataSet.length)

val data = sc.parallelize(dataSet)

val splits = data.randomSplit(Array(0.9, 0.1), seed = 11L)

val training = splits(0)

val test = splits(1)

if(!data.isEmpty()){

// Run training algorithm to build the model

val model = new LogisticRegressionWithLBFGS()

.setNumClasses(11)

.setIntercept(true)

.run(training)

// Compute raw scores on the test set.

val predictionAndLabels = test.map { case LabeledPoint(label, features) =>

val prediction = model.predict(features)

(prediction, label)

}

predictionAndLabels.foreach(x=> println(x))

// Get evaluation metrics.

val metrics = new MulticlassMetrics(predictionAndLabels)

val precision = metrics.precision

println("Precision = " + precision)

if(precision > 0.8){

val path = "hdfs://hadoop1:9000/model\_"+camera\_id+"\_"+nowtimelong

model.save(sc, path)

println("saved model to "+ path)

jedis.hset("model", camera\_id , path)

}

}

})

RedisClient.pool.returnResource(jedis)

}

}