COMP5318 MACHINE LEARNING AND DATA MINING

ASSIGNMENT 2 REPORT

OPTIMIZED CLASSIFICATION ON FOREST COVERTYPE

Based on kNN, Linear Regression and Random Forest Algorithms

June 5, 2017 [1]

Lin Han 460461265

Wanyi Fang 460165019

Dasheng Ge 460440743

[1] Powered by LaTeX

Contents

Introduction 3
Specification 3
Previous Work 3
Methods and Design 5
    Preprocessing 5
    Algorithm Selection 5
    Algorithm Introduction 6
        kNN 6
        Linear Regression 7
        Random Forest 7
Experiments and Discussion 8
Methods and Design 8
References 9

ABSTRACT

pass

INTRODUCTION

Classification aims to classify input data with common characteristics into the same categories as efficient as possible. These years, methods of classification are proposed increasingly. However, the performance of each method are quite different.

In this assignment, we have chosen the Covtype dataset as an object, a classical dataset with 581,012 instances of forests and 54 dimensions in judging the cover type, divided into seven labels. Since the dataset has been classified, we chose three supervised learning methods to re-classify in order that we can compare the performance of these three methods, and through deeper analyzing, we would choose one of them as recommended method. The three algorithms are: kNN, Linear Regression and Random Forest.

As the performance of an algorithm mainly includes time consumption and accuracy, the implementation has strengthened our understanding of each algorithms, including their efficiency and situations they fits for separately. Moreover, we got experience on comparing and choosing the appropriate methods when confronting with real problems, which is premise for rewriting and optimizing classical machine learning methods.

SPECIFICATION

The whole project was running and tested on ThinkPad T540P with i5-4200m@2.5GHz and 8G Ram.
The OS type is Ubuntu 16.04LTS.

PREVIOUS WORK

Before working on the problem, we have referred to several literature to find successful instances in dealing with similar dataset. This process helped us define the algorithms we would use for achieving the target time-saving and cost-effective in the process.

The characteristics of Covtype dataset are summarized as following:
1.Massive instances but less dimensions.
There are over 580000 instances shown in the dataset, however, only over 50 dimensions used for classification

2. Data in the dataset are discrete objects, not continuous.
3. With 7 labels given, the re-classify process should be a supervised learning process.
4. There are 7 cover types. Therefore, this is a multi-classification problem.

Based on the former dataset characteristics, following appropriate methods was found.

Method 1: VFDT (Very Fast Decision Tree learner), a decision-tree leaning system based on Hoeffding trees.
This algorithm is a representation of data stream classification technology. Since data today usually appears in data stream format, with the real-time, continuous(not actually continuous, just numerous), infinite, and non-reproducible four properties, and static classification cannot satisfy the real needs, classification for data stream is becoming more and more prevalent.
VFDT has the ability to incorporate and classify new information online in shorter training time by dividing the income data stream. It is powerful in dealing with large datasets. Moreover, as a ready-to-use model, VFDT can be used after the first few data have been trained, and its quality increase smoothly with time.

Method 2: Bagging and Boosting.
The two methods also came up from data stream classification wave and usually used as ensemble classification methods to generate advanced classifiers.
Bagging can be used to enhance the effect of classifier, which produces several replicate training sets by random sampling, then getting corresponding weak classifiers by training them separately, and integrating them finally.
Similar with bagging, boosting uses all weak classifiers to form a strong classifier. However, instances in boosting are all based with certain weight corresponding to the importance of each repetition. So adjusting the weights can create more accurate classifiers.

Method 3: Round Robin classification, a method based on separate-and-conquer rule algorithms.
It has attracted much attention in neural networks and SVM (support vector machines) communities. The basic idea is to transform multi-classification problems into binary classification problems. During the process, one classifier is applied for each pair of classes and ignoring all others when using only training examples for these two classes. Then the complexity is lower. Round Robin classification has been proved to get further improvement

by integrated with bagging algorithm mentioned above.

METHODS AND DESIGN

Preprocessing

The preprocessing in this project can be divided into two types: Preprocessing of the dataset itself in order to generate predicting set and training set; preprocessing for a defined method for improving the accuracy. This part will only introduce the former, and the special preprocessing for each algorithms themselves will be followed with algorithms implementation.

In the preprocessing, we used the origin dataset to form four subsets called train sample, train target, predict sample, predict target. It was completed by generate samples and targets.py.

Firstly, the whole dataset was separated into ten parts equally and randomly. Choose one subset from the ten and extracting the last column from the subset, as the last column is the label of instance.
Secondly, defining the remaining 53 columns of the chosen subset as predicting set, called predict sample in the procedure, and naming the last column as predict target individually. It is kept for testing the accuracy.
Thirdly, integrate the remaining nine subsets into one, then repeating the former process to get training set, called train sample and train target in the procedure.
Finally, repeating the process above all to form ten-folds.

Algorithm Selection

For supervised learning, we have learnt four methods: kNN(k-NearestNeighbor), Naive Bayes, Linear Regression, and SVM(Support Vector Machine). And there are several algorithms that we haven't learnt, such as Random Forest, Adaboost, SVR etc. In this assignment, we would like to choose three algorithms with superior diversity in theory, so that the comparison result would be more obvious. At the same time, time consumption is also an important index to consider.

In theory, SVM and Linear Regression have several similarities, and both requires high-dimensional vector calculation with high accuracy also high time complexity. Naive Bayes and kNN are both low complexity relatively, however, Naive Bayes is better to deal with

equal-weight-dimension dataset.

As a result, we chose kNN, Linear Regression and Random Forest as the comparison objects, for these three algorithms are not only diverse in performance, but can be completed in the same external open-source library: sklearn. As is known, variable libraries are possible to influence algorithm performance.

Algorithm Introduction

This part will introduce the algorithms separately and describe the significance of each parameter in details.

kNN

K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification. Representing instance x as a1(x), as(x),

...

,an(x)
(*formula*)
Where:
ar(x) = the r.th attribute of x.
d(xi,xj) = distance between instance xi and xj
for discrete dataset, it performs:
(*formula*)
Where:
v is an element of V set,

$$x_1, x_2, \ldots, x_n$$

select k nearest instances represented as x1,x2,âĂę, xk. Then return
(*formula*)

In the Covtype dataset, the attribute was fixed as 53.
kNN needs no special preprocessing.The dataset was directly classified through knn.py

Linear Regression

Logistic Regression can be used as a binary as well as multi-classification regression when the dependent variable is dichotomous. Using the thought of logistic regression, in our assignment, we can build up a linear model:
(*formula*)
for i=1,2,... n, where :
w stands for the parameter of the learnt.
x stands for test data, namely forest instance here.

Equating the linear model to a probability p(x) with logistic transformation applied.
(*formula*)
Therefore, we could derive:
(*formula*)
Also, we can have loss function:
(*formula*)
Where yi is 0 or 1 in logistic regression.

Based on the above process and applying gradient descent algorithm for each label, we can get a estimated weights vector w for every label. Using this vector, we can get the most probable label for a specific data.

In this assignment, we designed a specific preprocessing part for improving the performance of LR method, accordingly, we used K-Means in advance in order to achieve statistical outlier removal. Thus, the original dataset had been re-classified with KMeans, and formed 7 new clusters. Then LR worked on the original dataset and the new clustering dataset both to observe if there was an optimization on LR performance.

Random Forest

A Random Forest consists of a collection of simple tree predictors, each of which has the ability to produce a response when presented with a set of predictor values and can also be used to classify the final result. The optimal size of predictor variables is given by log(2M+1), where M is the number of inputs.

Given a set of simple trees and a set of random predictor variables, the Random Forest

method defines a margin function that measures the extent to which the average number of votes for the correct class exceeds the average vote for any other class present in the dependent variable. Given an ensemble of classifiers h1(x), h2(x), . . . , hK (x), and with the training set drawn at random from the distribution of the random vector Y, X, define the margin function as:
(*formula*)
The error can be defines as:(*formula*)

While implementation of Random Forest, we did similar preprocessing as LR methods,

EXPERIMENTS AND DISCUSSION

METHODS AND DESIGN

REFERENCES

1. Easley, D., & Kleinberg, J. (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.

2. Scrapy, https://scrapy.org/