**Spark Mllib之相关性计算和假设检验**

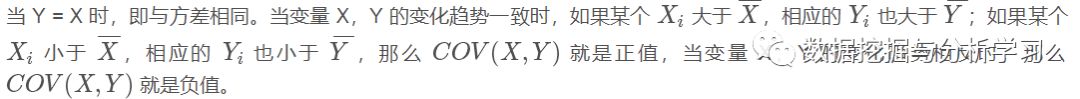
原创： 小小虫

一、皮尔逊相关性和斯皮尔曼相关性

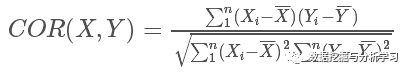
1.1 皮尔逊相关性

要理解 Pearson 相关系数，首先要理解协方差（Covariance）。协方差表示两个变量 X，Y 间相互关系的数字特征，其计算公式为：

251AD81B71494E3F9FCDAD21335B92D5.png

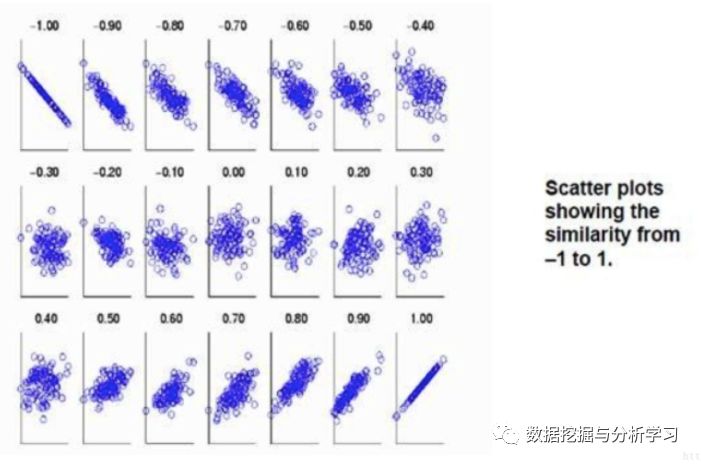


Pearson 相关系数公式如下：



由公式可知，Pearson 相关系数是用协方差除以两个变量的标准差得到的，虽然协方差能反映两个随机变量的相关程度（协方差大于0的时候表示两者正相关，小于0的时候表示两者负相关），但其数值上受量纲的影响很大，不能简单地从协方差的数值大小给出变量相关程度的判断。为了消除这种量纲的影响，于是就有了相关系数的概念。

当两个变量的方差都不为零时，相关系数才有意义，相关系数的取值范围为[-1,1]。《数据挖掘导论》中给了一个很形象的图来说明相关度大小与相关系数之间的联系：



由上图可以总结，当相关系数为1时，成为完全正相关；当相关系数为-1时，成为完全负相关；相关系数的绝对值越大，相关性越强；相关系数越接近于0，相关度越弱。

1.2 斯皮尔曼相关性

斯皮尔曼相关系数被定义成等级变量之间的皮尔逊相关系数。对于样本容量为*n*的样本，*n*个原始数据被转换成等级数据，相关系数ρ为

9266569A1F3D476AA0BF95A5A35B87AE.png

原始数据依据其在总体数据中平均的降序位置，被分配了一个相应的等级。如下表所示：

|  |  |  |
| --- | --- | --- |
| 变量Xi | 降序位置 | 等级xi |
| 0.8 | 5 | 5 |
| 1.2 | 4 |  |
| 1.2 | 3 |  |
| 2.3 | 2 | 2 |
| 18 | 1 | 1 |

实际应用中，变量间的连结是无关紧要的，于是可以通过简单的步骤计算ρ.被观测的两个变量的等级的差值，则ρ为

8BF7D17890E94F3CBB13248DCD703F67.png

两种相关性计算代码如下：

SparkSession session = SparkSession.builder().master("local").appName("correction").getOrCreate();

System.out.println("============相关性测试===============");

// Vectors.sparse()创建一个稀疏向量，三个参数分别为向量长度，向量索引数组，向量值数组,如果提供的索引和值数组长度小于要创建的向量长度，则其他的默认为0

List<Row> data = Arrays.asList(

RowFactory.create(Vectors.sparse(4, new int[] { 0, 3 }, new double[] { 1.0, -2.0 })),

RowFactory.create(Vectors.dense(4.0, 5.0, 0.0, 3.0)),

RowFactory.create(Vectors.dense(6.0, 7.0, 0.0, 8.0)),

RowFactory.create(Vectors.sparse(4, new int[] { 0, 3 }, new double[] { 9.0, 1.0 })));

double[] array = Vectors.sparse(8, new int[] { 0, 1, 2, 3 }, new double[] { 1.0, 3.0, 4.0, -5.0 }).toArray();

for (double d : array) {

System.out.print(d + "\t");

}

System.out.println();

StructType schema = new StructType(

new StructField[] { new StructField("features", new VectorUDT(), false, Metadata.empty()), });

Dataset<Row> df = session.createDataFrame(data, schema);

Row r1 = Correlation.corr(df, "features").head();

System.out.println("皮尔逊相关性矩阵:\n" + r1.get(0).toString());

Row r2 = Correlation.corr(df, "features", "spearman").head();

System.out.println("斯皮尔曼相关性矩阵:\n" + r2.get(0).toString());

session.stop();

输出结果如下：

============相关性测试===============

1.0 3.0 4.0 -5.0 0.0 0.0 0.0 0.0

皮尔逊相关性矩阵:

1.0                   0.055641488407465814  NaN  0.4004714203168137

0.055641488407465814  1.0                   NaN  0.9135958615342522

NaN                   NaN                   1.0  NaN

0.4004714203168137    0.9135958615342522    NaN  1.0

斯皮尔曼相关性矩阵:

1.0                  0.10540925533894532  NaN  0.40000000000000174

0.10540925533894532  1.0                  NaN  0.9486832980505141

NaN                  NaN                  1.0  NaN

0.40000000000000174  0.9486832980505141   NaN  1.0

二、假设检验

假设检验是统计学中一种强有力的工具，用于确定结果是否具有统计显着性，无论该结果是否偶然发生。 spark.ml目前支持Pearson的Chi-squared（χ2）独立性测试。

ChiSquareTest针对标签对每个特征进行Pearson独立测试。 对于每个特征，将（特征，标签）对转换为应急矩阵，对其计算卡方统计量。 所有标签和特征值必须是分类的。

代码如下：

package com.cb.spark.mllib;

import java.util.Arrays;

import java.util.List;

import org.apache.spark.ml.linalg.VectorUDT;

import org.apache.spark.ml.linalg.Vectors;

import org.apache.spark.ml.stat.ChiSquareTest;

import org.apache.spark.sql.Dataset;

import org.apache.spark.sql.Row;

import org.apache.spark.sql.RowFactory;

import org.apache.spark.sql.SparkSession;

import org.apache.spark.sql.types.DataTypes;

import org.apache.spark.sql.types.Metadata;

import org.apache.spark.sql.types.StructField;

import org.apache.spark.sql.types.StructType;

public class HypothesisTesting {

public static void main(String[] args) {

SparkSession session = SparkSession.builder().master("local").appName("hypothesisTesting").getOrCreate();

// 特征和标签数据

List<Row> data = Arrays.asList(RowFactory.create(0.0, Vectors.dense(0.5, 10.0)),

RowFactory.create(0.0, Vectors.dense(1.5, 20.0)), RowFactory.create(1.0, Vectors.dense(1.5, 30.0)),

RowFactory.create(0.0, Vectors.dense(3.5, 30.0)), RowFactory.create(0.0, Vectors.dense(3.5, 40.0)),

RowFactory.create(1.0, Vectors.dense(3.5, 40.0)));

StructType schema = new StructType(

new StructField[] { new StructField("label", DataTypes.DoubleType, false, Metadata.empty()),

new StructField("features", new VectorUDT(), false, Metadata.empty()) });

Dataset<Row> df = session.createDataFrame(data, schema);

Row r = ChiSquareTest.test(df, "features", "label").head();

System.out.println("p值:" + r.get(0).toString());

System.out.println("自由度:" + r.getList(1).toString());

System.out.println("统计值：" + r.get(2).toString());

}

}