Weka[37] Chi-square 源代码分析

作者：Koala++/屈伟

卡方(chi-square)核心代码在 buildEvaluator 中，而 buildEvalutor 中的代码绝大部分是与 InfoGainAttributeEval，因为只是加一个每个类别值，每个属性的每一个属性值的次数，保存在 counts 中，下面的代码是不同的几句：

```
// Compute chi-squared values
m ChiSquareds = new double[data.numAttributes()];
for (int i = 0; i < data.numAttributes(); i++) {
    if (i != classIndex) {
        m ChiSquareds[i] = ContingencyTables.chiVal(ContingencyTables
                .reduceMatrix(counts[i]), false);
    }
}
```

所调用的 reduceMatrix 代码如下：

```
/**
 * Reduces a matrix by deleting all zero rows and columns.
*/
public static double[][] reduceMatrix(double[][] matrix) {

    int row, col, currCol, currRow, nrows, ncols, nonZeroRows = 0,
         nonZeroColumns = 0;
    double[] rtotal, ctotal;
    double[][] newMatrix;

    nrows = matrix.length;
    ncols = matrix[0].length;
    rtotal = new double[nrows];
    ctotal = new double[ncols];
    for (row = 0; row < nrows; row++) {
        for (col = 0; col < ncols; col++) {
            rtotal[row] += matrix[row][col];
            ctotal[col] += matrix[row][col];
        }
    }
    for (row = 0; row < nrows; row++) {
        if (Utils.gr(rtotal[row], 0)) {
            nonZeroRows++;
        }
    }
    for (col = 0; col < ncols; col++) {
        if (Utils.gr(ctotal[col], 0)) {
            nonZeroColumns++;
        }
    }
    newMatrix = new double[nonZeroRows][nonZeroColumns];
    currRow = 0;
    for (row = 0; row < nrows; row++) {
        if (Utils.gr(rtotal[row], 0)) {
            currCol = 0;
            for (col = 0; col < ncols; col++) {
                if (Utils.gr(ctotal[col], 0)) {
                    newMatrix[currRow][currCol] = matrix[row][col];
                    currCol++;
                }
            }
            currRow++;
        }
    }
    return newMatrix;
}
```

rtotal，ctotal 分别是每个行与列的的全部元素之和，nonZeroRows 和 nonZeroColumns 分别是非 0 行与列的值，将这些元素值全为 0 的行或列删去，得到一个新的矩阵 newMatrix。

```
/**
 * Computes chi-squared statistic for a contingency table.
*/
public static double chiVal(double[][] matrix, boolean useYates) {

    int df, nrows, ncols, row, col;
    double[] rtotal, ctotal;
    double expect = 0, chival = 0, n = 0;
    boolean yates = true;

    nrows = matrix.length;
    ncols = matrix[0].length;
    rtotal = new double[nrows];
    ctotal = new double[ncols];
    for (row = 0; row < nrows; row++) {
        for (col = 0; col < ncols; col++) {
            rtotal[row] += matrix[row][col];
            ctotal[col] += matrix[row][col];
            n += matrix[row][col];
        }
    }
    df = (nrows - 1) * (ncols - 1);
    if ((df > 1) || (!useYates)) {
        yates = false;
    } else if (df <= 0) {
        return 0;
    }
    chival = 0.0;
    for (row = 0; row < nrows; row++) {
        if (Utils.gr(rtotal[row], 0)) {
            for (col = 0; col < ncols; col++) {
                if (Utils.gr(ctotal[col], 0)) {
                    expect = (ctotal[col] * rtotal[row]) / n;
                    chival += chiCell(matrix[row][col], expect, yates);
                }
            }
        }
    }
    return chival;
}
```

rtotal，ctotal，n 分别是一行的元素值之和，一列的元素值之和，全部元素之和。Expect 就是(A+C)*(A+B)/N，下面看 chiCell 中的代码：

```
/**
 * Computes chi-value for one cell in a contingency table.
*/
private static double chiCell(double freq, double expected, boolean yates)
```

```
{

    // Cell in empty row and column?
    if (Utils.smOrEq(expected, 0)) {
        return 0;
    }

    // Compute difference between observed and expected value
    double diff = Math.abs(freq - expected);
    if (yates) {

        // Apply Yates' correction if wanted
        diff -= 0.5;

        // The difference should never be negative
        if (diff < 0) {
            diff = 0;
        }
    }

    // Return chi-value for the cell
    return (diff * diff / expected);
}
```

关于 yates，可以看一下 wiki，词条：Yates' correction for continuity。它是用于小样本，防止高估它的统计显著性，它存在过度校正的可能。

Diff * diff / expected 就是公式了，没什么好讲的。在 chiValue 中，要把所有的 chiValue 加起来。

具体的，可以看一下 Jasper 写的，他写的还蛮有意思的：

http://www.blogjava.net/zhenandaci/archive/2008/08/31/225966.html

比较权威的资料，我以前看的，我自己也找不到了。