Weka[10] NBTree 源代码分析

作者：Koala++/屈伟

我不多的读者之一发 E-mail 给我，说他没有看出 NBTree 和 J48 的区别是什么，当时我也没什么空，所以拖到今天才草草看了看。大概讲一下。

下面是 J48 中的 buildClassifier 代码：

```
public void buildClassifier(Instances instances) throws Exception {

    ModelSelection modSelection;

    if (m_binarySplits)
        modSelection = new BinC45ModelSelection(m_minNumObj, instances);
    else
        modSelection = new C45ModelSelection(m_minNumObj, instances);
    if (!m_reducedErrorPruning)
        m_root = new C45PruneableClassifierTree(modSelection,
            !m_unpruned,m_CF, m_subtreeRaising, !m_noCleanup);
    else
        m_root = new PruneableClassifierTree(modSelection,
            !m_unpruned,m_numFolds, !m_noCleanup, m_Seed);
        m_root.buildClassifier(instances);
}
```

下面是 NBTree 中的 buildClassifier 代码：

```
public void buildClassifier(Instances instances) throws Exception {

    NBTreeModelSelection modSelection =
        new NBTreeModelSelection(m_minNumObj, instances);

    m_root = new NBTreeClassifierTree(modSelection);
    m_root.buildClassifier(instances);
}
```

这里有一个比较特殊的 ModelSelection 类，这个类以前没提过，它是决定树的模型类，比如上面 J48 代码中的 BinC45ModelSelection 表示对于连续属性，分裂时它只分出两个子结点。多扯两句，其实 BinC45ModelSelection 和 C45ModelSelection 类，包括以后要讲的 NBTreeModelSelection 内容都差不多，***ModelSelection 类中的 selectModel 函数返回一个 ClassifierSplitModel 对象，ClassifierSplitModel 故名思意是如何分裂的一个模型。

从上面讲的我们已经可以看出 J48 与 NBTree 在代码中选择了不同的构造树的模型，当然它们是不同的（当然这更是一句废话），另一点值得说的是在 J48 里 m_minNumObj 默认值是 2，而在 NBTree 中 m_minNumObj 的默认值是 30。

考虑到 NBTree 也不是什么经典算法，有人可能不知道是怎么回事，大概讲一下：与决策树的构造方法相似（认为相同也可以）先构造出一个决策树，再在每一个叶子结点构造

一个贝叶斯分类器（这也就是为什么默认 m_minNumObj 是 30 原因）。具体的内容见论文：Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid。

我们先看一下 NBTreeModelSelection 类，找到 selectModel 函数，刚才说这个函数与别的***MocelSelection 差不多，我也是有根据，可以看到 NBTreeModelSelection 中的两个警告，这两个变量在 NBTreeModelSelection 中是没用的，明显是拷贝的。请注意下面三个变量：

```
    NBTreeSplit[] currentModel;
    NBTreeSplit bestModel = null;
    NBTreeNoSplit noSplitModel = null;
```

NBTreeSplit 和 NBTreeNoSplit 都继承自 ClassifierSplitModel，selectModel 函数中注释也不少，我也不解释代码了，大概就是：比如在样本都属于一个样本这种情况就不分裂了，那么就返回 noSplitModel 对象，否则，针对第 j 个属性，调 currentModel[i].buildClassifier 函数，最后根据 getErrors 来决定哪一个属性是最好的分裂属性。

到现在为止，列出来的代码的确与 J48 差不多，它们的主要区别是在 NBTreeNoSplit 类中。下面先列出 J48 中所用的 NoSplit 类中的 buildClassifer 函数：

```
public final void buildClassifier(Instances instances)
throws Exception {
    m_distribution = new Distribution(instances);
    m_numSubsets = 1;
}
```

再列出 NBTreeNoSplit 类中的 buildClassifer 函数：

```
public final void buildClassifier(Instances instances) throws Exception
{
    m_nb = new NaiveBayesUpdateable();
    m_disc = new Discretize();
    m_disc.setInputFormat(instances);
    Instances temp = Filter.useFilter(instances, m_disc);
    m_nb.buildClassifier(temp);
    if (temp.numInstances() >= 5) {
        m_errors = crossValidate(m_nb, temp, new Random(1));
    }
    m_numSubsets = 1;
}
```

区别还是挺明显的，除了 m_numSubset=1 这个标志是叶子结点的语句。在 NBTreeNoSplit 类的 buildClassifier 中，在叶子结点构造一个 m_nb Naive Bayes 分类器，不过又说回来，讲了半天，也就是这一点点区别产生了 NBTree 这个新的分类器。

m_root 是一个 NBTreeClassifierTree 对象，我们再看一下 NBTreeClassifierTree 对象，我们直接看 buildClassifier 函数：

```
public void buildClassifier(Instances data) throws Exception {
    super.buildClassifier(data);
    cleanup(new Instances(data, 0));
    assignIDs(-1);
```

```
}
```

可以看到它直接调用的父类的buildClssifier，而它的父类就是ClassifierTree，在J48中同样使用的是ClssifierTree类。

对于分类一个样本，在NBTree的classifyInstance函数中，返回：

```
    return m_root.classifyInstance(instance);
```

刚才说对m_root是一个NBTreeClassifierTree对象，但NBTreeClassifier没有实现classifyInstance函数，那么m_root调用的classifyInstance实际上是ClassifierTree类的函数。在其classifyInstance中：

```
for (j = 0; j < instance.numClasses(); j++) {
    currentProb = getProbs(j, instance, 1);
    if (Utils.gr(currentProb, maxProb)) {
        maxIndex = j;
        maxProb = currentProb;
    }
}
```

这一段代码没什么意思，样本属于哪个类别概率最高，那么它就被分类为该类别。这里面的getProbs函数中才是我们关心的：

```
private double getProbs(int classIndex, Instance instance, double weight)
throws Exception {

    double prob = 0;

    if (m_isLeaf) {
        return weight * localModel().classProb(classIndex, instance, -1);
    } else {
        int treeIndex = localModel().whichSubset(instance);
        if (treeIndex == -1) {
            double[] weights = localModel().weights(instance);
            for (int i = 0; i < m_sons.length; i++) {
                if (!son(i).m_isEmpty) {
                prob += son(i).getProbs(classIndex, instance,
                    weights[i] * weight);
                }
            }
            return prob;
        } else {
            if (son(treeIndex).m_isEmpty) {
                 return weight * localModel().classProb(classIndex,
                     instance, treeIndex);
            } else {
                return son(treeIndex).getProbs(classIndex,
                     instance, weight);
            }
        }
```

```
    }
}
```

如果不是叶子结点：先得到这个样本属于应该是哪个子结点的，如果treeIndex=-1表示这个属属性值是缺失的，计算它的方法就是用对每个子结点分开算，再加起来。如果不是缺失的，如果子结点是空的，与是子结点的计算方法相同，否则，递归。

如果是叶子结点：localModel返回的是ClassifierSplitModel对象，该对象调用classProb函数，我们看一下NBTreeNoSplit函数的classProb函数：

```
public double classProb(int classIndex, Instance instance, int theSubset)
throws Exception {
    m_disc.input(instance);
    Instance temp = m_disc.output();
    return m_nb.distributionForInstance(temp)[classIndex];
}
```

刚才所提到的m_nb这个Naive Bayes分类器调用distributionForInstance。NBTree差不多讲完了，最后来点打击人的，我真感觉这个分类器没有太大的必要搞懂，不过提出它的作者我认为算是一个想象力丰富的人，至于NBTree 的应用，我仅知道它在VFDTc中用到了，还在它的几个改进版中用到过，其它的用到它的地方我也不知道，有人知道，请告诉我。