决策树

package ai6;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

public class DicisionTree {

public static void main(String[] args) throws Exception {

String[] attrNames = new String[] { "AGE", "INCOME", "STUDENT",

"CREDIT\_RATING" };

Map<Object, List<Sample>> samples = readSamples(attrNames);

Object decisionTree = generateDecisionTree(samples, attrNames);

outputDecisionTree(decisionTree, 0, null);

}

static Map<Object, List<Sample>> readSamples(String[] attrNames) {

Object[][] rawData = new Object[][] {

{ "<30 ", "High ", "No ", "Fair ", "0" },

{ "<30 ", "High ", "No ", "Excellent", "0" },

{ "30-40", "High ", "No ", "Fair ", "1" },

{ ">40 ", "Medium", "No ", "Fair ", "1" },

{ ">40 ", "Low ", "Yes", "Fair ", "1" },

{ ">40 ", "Low ", "Yes", "Excellent", "0" },

{ "30-40", "Low ", "Yes", "Excellent", "1" },

{ "<30 ", "Medium", "No ", "Fair ", "0" },

{ "<30 ", "Low ", "Yes", "Fair ", "1" },

{ ">40 ", "Medium", "Yes", "Fair ", "1" },

{ "<30 ", "Medium", "Yes", "Excellent", "1" },

{ "30-40", "Medium", "No ", "Excellent", "1" },

{ "30-40", "High ", "Yes", "Fair ", "1" },

{ ">40 ", "Medium", "No ", "Excellent", "0" } };

Map<Object, List<Sample>> ret = new HashMap<Object, List<Sample>>();

for (Object[] row : rawData) {

Sample sample = new Sample();

int i = 0;

for (int n = row.length - 1; i < n; i++)

sample.setAttribute(attrNames[i], row[i]);

sample.setCategory(row[i]);

List<Sample> samples = ret.get(row[i]);

if (samples == null) {

samples = new LinkedList<Sample>();

ret.put(row[i], samples);

}

samples.add(sample);

}

return ret;

}

static Object generateDecisionTree(

Map<Object, List<Sample>> categoryToSamples, String[] attrNames) {

if (categoryToSamples.size() == 1)

return categoryToSamples.keySet().iterator().next();

if (attrNames.length == 0) {

int max = 0;

Object maxCategory = null;

for (Entry<Object, List<Sample>> entry : categoryToSamples

.entrySet()) {

int cur = entry.getValue().size();

if (cur > max) {

max = cur;

maxCategory = entry.getKey();

}

}

return maxCategory;

}

Object[] rst = chooseBestTestAttribute(categoryToSamples, attrNames);

Tree tree = new Tree(attrNames[(Integer) rst[0]]);

String[] subA = new String[attrNames.length - 1];

for (int i = 0, j = 0; i < attrNames.length; i++)

if (i != (Integer) rst[0])

subA[j++] = attrNames[i];

@SuppressWarnings("unchecked")

Map<Object, Map<Object, List<Sample>>> splits =

/\* NEW LINE \*/(Map<Object, Map<Object, List<Sample>>>) rst[2];

for (Entry<Object, Map<Object, List<Sample>>> entry : splits.entrySet()) {

Object attrValue = entry.getKey();

Map<Object, List<Sample>> split = entry.getValue();

Object child = generateDecisionTree(split, subA);

tree.setChild(attrValue, child);

}

return tree;

}

static Object[] chooseBestTestAttribute(

Map<Object, List<Sample>> categoryToSamples, String[] attrNames) {

int minIndex = -1;

double minValue = Double.MAX\_VALUE;

Map<Object, Map<Object, List<Sample>>> minSplits = null;

for (int attrIndex = 0; attrIndex < attrNames.length; attrIndex++) {

int allCount = 0;

Map<Object, Map<Object, List<Sample>>> curSplits =

/\* NEW LINE \*/new HashMap<Object, Map<Object, List<Sample>>>();

for (Entry<Object, List<Sample>> entry : categoryToSamples

.entrySet()) {

Object category = entry.getKey();

List<Sample> samples = entry.getValue();

for (Sample sample : samples) {

Object attrValue = sample

.getAttribute(attrNames[attrIndex]);

Map<Object, List<Sample>> split = curSplits.get(attrValue);

if (split == null) {

split = new HashMap<Object, List<Sample>>();

curSplits.put(attrValue, split);

}

List<Sample> splitSamples = split.get(category);

if (splitSamples == null) {

splitSamples = new LinkedList<Sample>();

split.put(category, splitSamples);

}

splitSamples.add(sample);

}

allCount += samples.size();

}

double curValue = 0.0;

for (Map<Object, List<Sample>> splits : curSplits.values()) {

double perSplitCount = 0;

for (List<Sample> list : splits.values())

perSplitCount += list.size();

double perSplitValue = 0.0;

for (List<Sample> list : splits.values()) {

double p = list.size() / perSplitCount;

perSplitValue -= p \* (Math.log(p) / Math.log(2));

}

curValue += (perSplitCount / allCount) \* perSplitValue;

}

if (minValue > curValue) {

minIndex = attrIndex;

minValue = curValue;

minSplits = curSplits;

}

}

return new Object[] { minIndex, minValue, minSplits };

}

static void outputDecisionTree(Object obj, int level, Object from) {

for (int i = 0; i < level; i++)

System.out.print("|-----");

if (from != null)

System.out.printf("(%s):", from);

if (obj instanceof Tree) {

Tree tree = (Tree) obj;

String attrName = tree.getAttribute();

System.out.printf("[%s = ?]\n", attrName);

for (Object attrValue : tree.getAttributeValues()) {

Object child = tree.getChild(attrValue);

outputDecisionTree(child, level + 1, attrName + " = "

+ attrValue);

}

} else {

System.out.printf("[CATEGORY = %s]\n", obj);

}

}

static class Sample {

private Map<String, Object> attributes = new HashMap<String, Object>();

private Object category;

public Object getAttribute(String name) {

return attributes.get(name);

}

public void setAttribute(String name, Object value) {

attributes.put(name, value);

}

public Object getCategory() {

return category;

}

public void setCategory(Object category) {

this.category = category;

}

public String toString() {

return attributes.toString();

}

}

static class Tree {

private String attribute;

private Map<Object, Object> children = new HashMap<Object, Object>();

public Tree(String attribute) {

this.attribute = attribute;

}

public String getAttribute() {

return attribute;

}

public Object getChild(Object attrValue) {

return children.get(attrValue);

}

public void setChild(Object attrValue, Object child) {

children.put(attrValue, child);

}

public Set<Object> getAttributeValues() {

return children.keySet();

}

}

}

结果：

