实验课程名称：\_\_人工智能\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **ID3决策树算法** | | | **实验成绩** |  |
| **实 验 者** | **林文敏** | **专业班级** | **16级计科5班** | **学 号** | **3216004669** |
| **实验日期** | **2018.10.29** | | |  |  |
| 一、实验内容 利用ID3实现决策树 二、实验设计（原理分析及流程）1.决策树介绍 决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。Entropy = 系统的凌乱程度，使用算法ID3, C4.5和C5.0生成树算法使用熵。这一度量是基于信息学理论中熵的概念。决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别。  决策树是一种十分常用的分类方法。他是一种监管学习，所谓监管学习就是给定一堆样本，每个样本都有一组属性和一个类别，这些类别是事先确定的，那么通过学习得到一个分类器，这个分类器能够对新出现的对象给出正确的分类。这样的机器学习就被称之为监督学习。 2 ID3算法介绍 ID3算法是决策树的一种，它是基于奥卡姆剃刀原理的，即用尽量用较少的东西做更多的事。ID3算法，即Iterative Dichotomiser 3，迭代二叉树3代，是Ross Quinlan发明的一种决策树算法，这个算法的基础就是上面提到的奥卡姆剃刀原理，越是小型的决策树越优于大的决策树，尽管如此，也不总是生成最小的树型结构，而是一个启发式算法在信息论中，期望信息越小，那么信息增益就越大，从而纯度就越高。ID3算法的核心思想就是以信息增益来度量属性的选择，选择分裂后信息增益最大的属性进行分裂。该算法采用自顶向下的贪婪搜索遍历可能的决策空间。 3信息熵与信息增益 在信息增益中，重要性的衡量标准就是看特征能够为分类系统带来多少信息，带来的信息越多，该特征越重要。在认识信息增益之前，先来看看信息熵的定义熵这个概念最早起源于物理学，在物理学中是用来度量一个热力学系统的无序程度，而在信息学里面，熵是对不确定性的度量。在1948年，香农引入了信息熵，将其定义为离散随机事件出现的概率，一个系统越是有序，信息熵就越低，反之一个系统越是混乱，它的信息熵就越高。所以信息熵可以被认为是系统有序化程度的一个度量。  信息增益是针对一个一个特征而言的，就是看一个特征，系统有它和没有它时的信息量各是多少，两者的差值就是这个特征给系统带来的信息量，即信息增益。  数据结构：  存储表对象，具体为一个数组对象，内包含若干数组  var examples = [  {day:'D1',outlook:'Sunny', temp:'Hot', humidity:'High', wind: 'Weak',play:'No'},  {day:'D2',outlook:'Sunny', temp:'Hot', humidity:'High', wind: 'Strong',play:'No'},  {day:'D3',outlook:'Overcast', temp:'Hot', humidity:'High', wind: 'Weak',play:'Yes'},  {day:'D4',outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Weak',play:'Yes'},  {day:'D5',outlook:'Rain', temp:'Cool', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D6',outlook:'Rain', temp:'Cool', humidity:'Normal', wind: 'Strong',play:'No'},  {day:'D7',outlook:'Overcast', temp:'Cool', humidity:'Normal', wind: 'Strong',play:'Yes'},  {day:'D8',outlook:'Sunny', temp:'Mild', humidity:'High', wind: 'Weak',play:'No'},  {day:'D9',outlook:'Sunny', temp:'Cool', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D10',outlook:'Rain', temp:'Mild', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D11',outlook:'Sunny', temp:'Mild', humidity:'Normal', wind: 'Strong',play:'Yes'},  {day:'D12',outlook:'Overcast', temp:'Mild', humidity:'High', wind: 'Strong',play:'Yes'},  {day:'D13',outlook:'Overcast', temp:'Hot', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D14',outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Strong',play:'No'}  ];  存储属性的数组  var features = ['outlook', 'temp', 'humidity', 'wind'];  决策树存储示例  var samples = [{outlook:'Overcast', temp:'Mild', humidity:'High', wind: 'Strong',play: 'Yes'},  {outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Strong', play: 'No'},  {outlook:'Sunny', temp:'Cool', humidity:'Normal', wind: 'Weak', play: 'Yes'}] | | | | | |
| **三、实验代码及截图**  **1.代码**  Example.js  var examples = [  {day:'D1',outlook:'Sunny', temp:'Hot', humidity:'High', wind: 'Weak',play:'No'},  {day:'D2',outlook:'Sunny', temp:'Hot', humidity:'High', wind: 'Strong',play:'No'},  {day:'D3',outlook:'Overcast', temp:'Hot', humidity:'High', wind: 'Weak',play:'Yes'},  {day:'D4',outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Weak',play:'Yes'},  {day:'D5',outlook:'Rain', temp:'Cool', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D6',outlook:'Rain', temp:'Cool', humidity:'Normal', wind: 'Strong',play:'No'},  {day:'D7',outlook:'Overcast', temp:'Cool', humidity:'Normal', wind: 'Strong',play:'Yes'},  {day:'D8',outlook:'Sunny', temp:'Mild', humidity:'High', wind: 'Weak',play:'No'},  {day:'D9',outlook:'Sunny', temp:'Cool', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D10',outlook:'Rain', temp:'Mild', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D11',outlook:'Sunny', temp:'Mild', humidity:'Normal', wind: 'Strong',play:'Yes'},  {day:'D12',outlook:'Overcast', temp:'Mild', humidity:'High', wind: 'Strong',play:'Yes'},  {day:'D13',outlook:'Overcast', temp:'Hot', humidity:'Normal', wind: 'Weak',play:'Yes'},  {day:'D14',outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Strong',play:'No'}  ];  examples = \_(examples);  var features = ['outlook', 'temp', 'humidity', 'wind'];  var samples = [{outlook:'Overcast', temp:'Mild', humidity:'High', wind: 'Strong',play: 'Yes'},  {outlook:'Rain', temp:'Mild', humidity:'High', wind: 'Strong', play: 'No'},  {outlook:'Sunny', temp:'Cool', humidity:'Normal', wind: 'Weak', play: 'Yes'}]  ID3.js  var id3 = function(\_s,target,features){  var targets = \_.unique(\_s.pluck(target));  if (targets.length == 1){  console.log("end node! "+targets[0]);  return {type:"result", val: targets[0], name: targets[0],alias:targets[0]+randomTag() };  }  if(features.length == 0){  console.log("returning the most dominate feature!!!");  var topTarget = mostCommon(\_s.pluck(target));  return {type:"result", val: topTarget, name: topTarget, alias: topTarget+randomTag()};  }  var bestFeature = maxGain(\_s,target,features);  var remainingFeatures = \_.without(features,bestFeature);  var possibleValues = \_.unique(\_s.pluck(bestFeature));  console.log("node for "+bestFeature);  var node = {name: bestFeature,alias: bestFeature+randomTag()};  node.type = "feature";  node.vals = \_.map(possibleValues,function(v){  console.log("creating a branch for "+v);  var \_newS = \_(\_s.filter(function(x) {return x[bestFeature] == v}));  var child\_node = {name:v,alias:v+randomTag(),type: "feature\_value"};  child\_node.child = id3(\_newS,target,remainingFeatures);  return child\_node;  });  return node;  }  var predict = function(id3Model,sample) {  var root = id3Model;  while(root.type != "result"){  var attr = root.name;  var sampleVal = sample[attr];  var childNode = \_.detect(root.vals,function(x){return x.name == sampleVal});  root = childNode.child;  }  return root.val;  }  //necessary math functions  var entropy = function(vals){  var uniqueVals = \_.unique(vals);  var probs = uniqueVals.map(function(x){return prob(x,vals)});  var logVals = probs.map(function(p){return -p\*log2(p) });  return logVals.reduce(function(a,b){return a+b},0);  }  var gain = function(\_s,target,feature){  var attrVals = \_.unique(\_s.pluck(feature));  var setEntropy = entropy(\_s.pluck(target));  var setSize = \_s.size();  var entropies = attrVals.map(function(n){  var subset = \_s.filter(function(x){return x[feature] === n});  return (subset.length/setSize)\*entropy(\_.pluck(subset,target));  });  var sumOfEntropies = entropies.reduce(function(a,b){return a+b},0);  return setEntropy - sumOfEntropies;  }  var maxGain = function(\_s,target,features){  return \_.max(features,function(e){return gain(\_s,target,e)});  }  var prob = function(val,vals){  var instances = \_.filter(vals,function(x) {return x === val}).length;  var total = vals.length;  return instances/total;  }  var log2 = function(n){  return Math.log(n)/Math.log(2);  }  var mostCommon = function(l){  return \_.sortBy(l,function(a){  return count(a,l);  }).reverse()[0];  }  var count = function(a,l){  return \_.filter(l,function(b) { return b === a}).length  }  var randomTag = function(){  return "\_r"+Math.round(Math.random()\*1000000).toString();  }  //Display logic  var drawGraph = function(id3Model,divId){  var g = new Array();  g = addEdges(id3Model,g).reverse();  window.g = g;  var data = google.visualization.arrayToDataTable(g.concat(g));  var chart = new google.visualization.OrgChart(document.getElementById(divId));  google.visualization.events.addListener(chart, 'ready',function(){  \_.each($('.google-visualization-orgchart-node'),function(x){  var oldVal = $(x).html();  if(oldVal){  var cleanVal = oldVal.replace(/\_r[0-9]+/,'');  $(x).html(cleanVal);  }  });  });  chart.draw(data, {allowHtml: true});  }  var addEdges = function(node,g){  if(node.type == 'feature'){  \_.each(node.vals,function(m){  g.push([m.alias,node.alias,'']);  g = addEdges(m,g);  });  return g;  }  if(node.type == 'feature\_value'){  g.push([node.child.alias,node.alias,'']);  if(node.child.type != 'result'){  g = addEdges(node.child,g);  }  return g;  }  return g;  }  var renderSamples = function(samples,$el,model,target,features){  \_.each(samples,function(s){  var features\_for\_sample = \_.map(features,function(x){return s[x]});  $el.append("<tr><td>"+features\_for\_sample.join('</td><td>')+"</td><td><b>"+predict(model,s)+"</b></td><td>actual: "+s[target]+"</td></tr>");  })  }  var renderTrainingData = function(\_training,$el,target,features){  \_training.each(function(s){  $el.append("<tr><td>"+\_.map(features,function(x){return s[x]}).join('</td><td>')+"</td><td>"+s[target]+"</td></tr>");  })  }  var calcError = function(samples,model,target){  var total = 0;  var correct = 0;  \_.each(samples,function(s){  total++;  var pred = predict(model,s);  var actual = s[target];  if(pred == actual){  correct++;  }  });  return correct/total;  }  Main.html  <html>  <head>  <script type="text/javascript" src="http://code.jquery.com/jquery-1.8.1.min.js"></script>  <script type="text/javascript"  src="http://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.3.3/underscore-min.js"></script>  <script type="text/javascript"  src="https://raw.github.com/strathausen/dracula/master/source/dracula\_graph.js"></script>  <script type="text/javascript"  src="https://raw.github.com/DmitryBaranovskiy/raphael/master/raphael-min.js"></script>  <script type="text/javascript"  src="http://sigmajs.org/js/sigma.min.js"></script>  <script type="text/javascript" src="https://raw.github.com/strathausen/dracula/master/js/dracula\_graffle.js"></script>  <script type="text/javascript" src="https://raw.github.com/strathausen/dracula/master/js/dracula\_graph.js"></script>  <script type="text/javascript"  src="https://raw.github.com/strathausen/dracula/master/js/dracula\_algorithms.js"></script>  <script type="text/javascript" src="js/examples.js"></script>  <script type="text/javascript"  src="js/tic-tac-examples.js"></script>  <script type="text/javascript"  src="js/voting-examples.js"></script>  <script src="http://d3js.org/d3.v2.js"></script>  <script type="text/javascript" src="js/id3.js"></script>  <script type='text/javascript'  src='https://www.google.com/jsapi?autoload={"modules":[{"name":"visualization","version":"1","packages":["orgchart"]}]}'></script>  <script type="text/javascript">  $(document).ready(function(){  renderTrainingData(examples,$("#training"),'play',features);  $("#fire\_tennis").click(function(e){  e.preventDefault();  var testModel = id3(examples,'play',features);  drawGraph(testModel,'canvas');  renderSamples(samples,$("#samples"),testModel,'play',features);  // renderTrainingData(examples,$("#training"),'play',features);  console.log("error");  console.log(calcError(samples,testModel,'play'));  });  });  </script>  </head>  <body>  <div id="main">  <h1>ID3 决策树</h1>  <a id="fire\_tennis" href="#">进行决策：</a>  <div id="data-container">  <div >  <h3>表格如下：</h3>  <table id='training'>  </table>  <div id='canvas'></div>  </div>  </div>  </div>  </body>  </html>  **2. 结果截图**    打印出的树：  C:\Users\home123\AppData\Local\Temp\1511148761(1).png  整体页面显示  C:\Users\home123\AppData\Local\Temp\1511148793(1).png | | | | | |