ID3

算法实现及决策树可视化

徐遥

2017 年 3 月 28 日

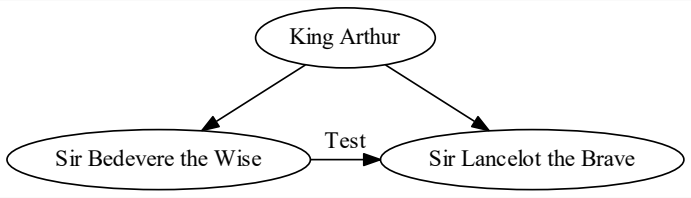
目录

可视化

Graphviz

决策树的可视化利用了贝尔实验室开发的 Graphviz 工具包。用户可以使用 DOT 语言来描述图形，然后利用该工具进行图形的布局与绘制，省去手动调整元素的大小与局部的繁琐过程。对于决策树而言，我们只需要了解如何往有向图(digraph)添加节点与边。

```
// The Round Table
digraph {
        A [label="King Arthur"]
        B [label="Sir Bedevere the Wise"]
        L [label="Sir Lancelot the Brave"]
                A -> B
                A -> L
                B -> L [label=Test constraint=false]
}
```



(a) DOT文件　　**(b)** 渲染后

图 1: Graphviz使用示例

Graphviz 的 Python 接口

graphviz 提供了创建 DOT 文件的 Pyhton 接口。实际上，图 1 就是其官方示例。

```
1  # The round Table
2  from graphviz import Digraph
3  # Create a graph object
4  dot = Digraph(comment='The Round Table')
5  # Add nodes and edges
6  dot.node('A', 'King Arthur')
7  dot.node('B', 'Sir Bedevere the Wise')
8  dot.node('L', 'Sir Lancelot the Brave')
9  dot.edges(['AB', 'AL'])
10 dot.edge('B', 'L', constraint='false', label='Test')
11 # Save and render the source code
12 dot.render('../figures/graphviz_demo')
```

自动生成决策树的 DOT 文件 I

有了上述工具，决策树的可视化就变得非常简单了:只需要递归地把所有节点和边加入有向图即可。

```
class ID3(object):
    class Node(object):
        @property
        def node_name(self):
            # 可视化时，每个node，必须要有独一无二的name
            return ''.join([self.attribute, str(self.id)])

        def add_to_graph(self, graph):
            graph.node(self.node_name, self.__str__())
            for edge_name, branch_node in self.branches.items():
                branch_node.add_to_graph(graph)
                graph.edge(self.node_name, branch_node.node_name,
    ↪  label=str(edge_name))
```

自动生成决策树的 DOT 文件 II

```
13
14      def render_decision_tree(self, filename):
15          if not self.root_node:
16              raise ValueError('Tree not decided!')
17          from graphviz import Digraph
18          dot_graph = Digraph(comment="Decision Tree")
19          self.root_node.add_to_graph(dot_graph)
20          dot_graph.render(filename)
```

生成树展示

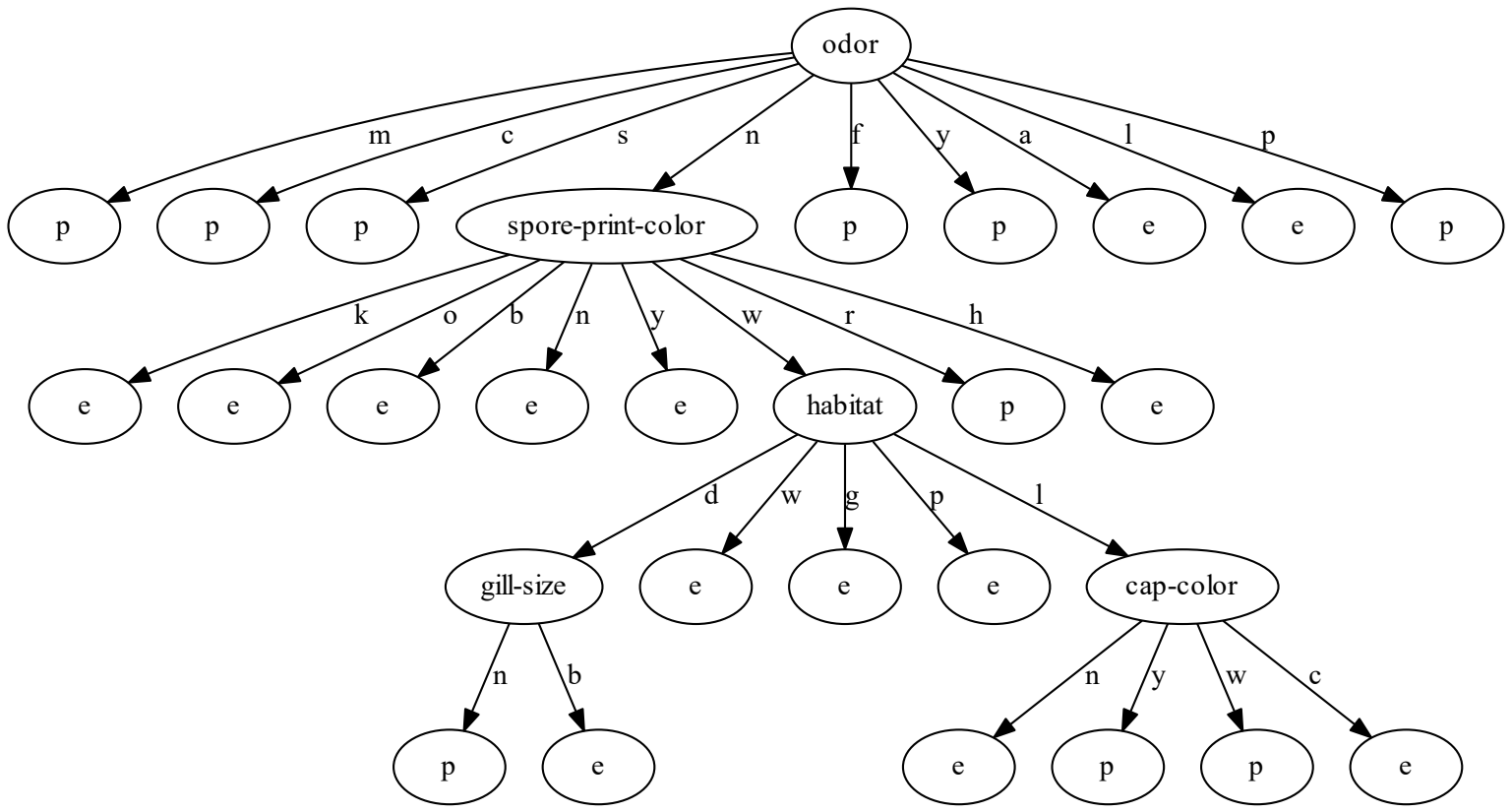


图 2: 蘑菇毒性分类决策树