Chapter 1.11
A Tutorial on Hierarchical Classification with Applications in Bioinformatics

Alex Freitas
University of Kent, UK

André C.P.L.F. de Carvalho
University of São Paulo, Brazil

ABSTRACT

In machine learning and data mining, most of the works in classification problems deal with flat classification, where each instance is classified in one of a set of possible classes and there is no hierarchical relationship between the classes. There are, however, more complex classification problems where the classes to be predicted are hierarchically related. This chapter presents a tutorial on the hierarchical classification techniques found in the literature. We also discuss how hierarchical classification techniques have been applied to the area of bioinformatics (particularly the prediction of protein function), where hierarchical classification problems are often found.

INTRODUCTION

Classification is one of the most important problems in machine learning (ML) and data mining (DM). In general, a classification problem can be formally defined as:

Given a set of training examples composed of pairs $\{x_i, y_i\}$, find a function f(x) that maps each x_i to its associated class y_i, i = 1, 2, ..., n, where n is the total number of training examples.

After training, the predictive accuracy of the classification function induced is evaluated by using it to classify a set of unlabeled examples, unseen during training. This evaluation measures the generalization ability (predictive accuracy) of the classification function induced.

The vast majority of classification problems addressed in the literature involves flat classification, where each example is assigned to a class out of a finite (and usually small) set of classes. By contrast, in hierarchical classification problems, the classes are disposed in a hierarchical structure, such as a tree or a directed acyclic graph (DAG). In these structures, the nodes represent classes. Figure 1 illustrates the difference between flat and hierarchical classification problems. To keep the example simple, Figure 1b shows a tree-structured class hierarchy. The more complex case of DAG-structured class hierarchies will be discussed later. In Figure 1, each node—except the root nodes—is labeled with the number of a class. In Figure 1b, class 1 is divided into two sub-classes, 1.1 and 1.2, and class 3 is divided into three sub-classes. The root nodes are labeled "any class" to denote the case where the class of an example is unknown. Figure 1 clearly shows that flat classification problems are actually a particular case of hierarchical classification problems where there is a single level of classes—that is, where no class is divided into sub-classes.

In the flat classification problem of Figure 1a, there is a single level of classes to be assigned to an example, but the class hierarchy of Figure 1b offers us more flexibility to specify at which level of the hierarchy a class will be assigned to an example.

For instance, one could require that an example should be assigned to a leaf, most specific, class. In the case of Figure 1b, this means that the candidate classes to be assigned to this example are 1.1, 1.2, 2, 3.1, 3.2, and 3.3. At first glance, by defining that only leaf classes can be assigned to an example, we are implicitly transforming the hierarchical classification problem into a flat one, since we could use a flat classification algorithm to solve it. Note, however, that in this case the flat classification algorithm would ignore valuable information in the structure of the class hierarchy. For instance, the fact that class 1.1 is more similar to class 1.2 than to class 3.1. By contrast, a truly hierarchical classification algorithm will take into account the structure of the class hierarchy. Even if we require the hierarchical algorithm to perform class assignments at the leaf level, the algorithm exploits the structure of the class hierarchy to look for a more accurate classification function.

On the other hand, we could be more flexible and allow the hierarchical classification algorithm to classify an example at any appropriate level, depending on the predictive power of the available data. For instance, an example could be reliably classified as having the specific class 1.1, while perhaps another example could only be reliably classified as having the general class 1.

Note that in general, class assignments at internal nodes can be carried out in a more reliable manner than class assignments at leaf nodes, because discriminating between the most specific classes at leaf nodes is more difficult than discriminating between the more general classes at internal nodes and, as a related factor, the number of examples per leaf node tends to be considerably smaller than the number of examples per internal node. On the other hand, class assignments at the leaf node tend to be more useful than class assignments at internal nodes, because the former provides more information about the class of an example. This trade-off between the reliability of a class assignment and its usefulness is common in hierarchical classification problems.

This chapter also presents applications of concepts and methods of hierarchical classification to problems in bioinformatics, at present one of the most important groups of DM applications. The application of DM techniques to bioinformatics is a very active research area, and it is not feasible to address the entire area in a single book chapter. Hence, this chapter focuses on a particular kind of bioinformatics problem, namely the prediction of protein function.

Proteins are large molecules that execute nearly all of the functions of a cell in a living organism (Alberts et al., 2002). They consist essentially of long sequences of amino acids, which fold into