Chapter 1
LCSs in a Nutshell

Abstract

This chapter aims to introduce readers to Learning Classifier Systems (LCSs) through the lens of an accessible but non-trivial classification problem. It offers a brief summary of the basic concepts and components of an LCS algorithm, concluding with code exercises that pair with this textbook to offer hands-on experience.

Let’s begin by exploring an example of a simple Learning Classifier System (LCS) and briefly answering important questions: What are the basic components of an LCS? What is a rule and how does it differ from a model? Furthermore, what does an LCS model look like and how is it used to make predictions? As we present LCS in a nutshell, keep in mind that we are using a generalised and simplistic example of an LCS algorithm. We will cover many of the key algorithmic variations and adaptations of this LCS framework in the chapters that follow. In the meantime, we will examine what is known as a Michigan-style LCS algorithm (see Sections 4.2 and 4.3) as it is flexible, well understood through research, and has a greater range of application than alternative LCS styles. To further simplify this introduction, we focus on an LCS that only uses supervised learning (i.e. the learner has access to the correct class/decision for every input instance). Supervised learning is common in data science tasks and single-step problems where the predicted class/decision is based on the state of the current instance alone. This is different from multi-step problems such as game strategy, where the current and prior states may be relevant.

1.1 A Non-trivial Example Problem: The Multiplexer

It is useful to define a clear, yet non-trivial example problem as a backdrop for describing LCS. To that end, the multiplexer problem was chosen, as it also reflects
properties that are relevant to challenges within complex real-world problems, such as bioinformatics, finance, and behavior modeling. Keep in mind that this example problem alone does not capture the full range of applications of which LCS is capable. The Boolean n-bit multiplexer defines a set of single-step supervised learning problems (e.g. 6-bit, 11-bit) that are conceptually based on the behavior of an electronic multiplexer (MUX), a device that takes multiple analog or digital input signals and switches them into a single output. The 6-bit multiplexer problem is illustrated in Figure 1.1. To generate a 6-bit multiplexer training dataset we can generate random bit-strings of length six, and for each, the class/output value is determined by the two address bits and the value at the register bit they point to. In Figure 1.1 we see that the first instance from the training data has the bit-string state of ‘010110’. Each binary digit represents a distinct feature in the dataset that can have one of two values: ‘0’ or ‘1’. Features can also be referred to as attributes, or independent variables. In the 6-bit multiplexer problem, the first two features are address bits. The address ‘01’ points to the register bit with ID = 1 (i.e. the second register bit). The value of that register bit equals 1, thus the class of this instance equals 1. The class of an instance can also be referred to as the endpoint, action, phenotype, or dependent variable based on the problem at hand. If you examine other example instances from the training data in Figure 1.1 you can verify this relationship between address and register bits. Since the 6-bit multiplexer has two address bits and four register bits, it has also been referred to as the 2-4 multiplexer problem.

Any multiplexer problem is non-trivial because it relies on a fairly complicated pattern to determine class. John Koza, best known for his work pioneering genetic programming (an evolutionary computation strategy) once wrote, “Multiplexer functions have long been identified by researchers as functions that often pose difficulties for paradigms for machine learning, artificial intelligence, neural nets, and classifier systems.” In particular every multiplexer problem involves an interaction effect between multiple features, also known as epistasis. They also exhibit het-