What Is a Learning Classifier System?

A learning classifier system (LCS) is an adaptive system that learns to perform the best action given its input. By "best" is generally meant the action that will receive the most reward or reinforcement from the system's environment. By "input" is meant the environment as sensed by the system, usually a vector of numerical values. The set of available actions depends on the system context: if the system is a mobile robot, the available actions may be physical: "turn left", "turn right", etc. In a classification context, the available actions may be "yes", "no", or "benign", "malignant", etc. In a decision context, for instance a financial one, the actions might be "buy", "sell", etc. In general, an LCS is a simple model of an intelligent agent interacting with an environment.

An LCS is "adaptive" in the sense that its ability to choose the best action improves with experience. The source of the improvement is reinforcement—technically, *payoff*—provided by the environment. In many cases, the payoff is arranged by the experimenter or trainer of the LCS. For instance, in a classification context, the payoff may be 1.0 for "correct" and 0.0 for "incorrect". In a robotic context, the payoff could be a number representing the change in distance to a recharging source, with more desirable changes (getting closer) represented by larger positive numbers, etc. Often, systems can be set up so that effective reinforcement is provided automatically, for instance via a distance sensor. Payoff received for a given action is used by the LCS to alter the likelihood of taking that action, in those circumstances, in the future. To understand how this works, it is necessary to describe some of the LCS mechanics.

Inside the LCS is a set—technically, a *population*—of "condition-action rules" called *classifiers*. There may be hundreds of classifiers in the population. When a particular input occurs, the LCS forms a so-called *match set* of classifiers whose conditions are satisfied by that input. Technically, a condition is a *truth function* $t(x)$ which is satisfied for certain input vectors x. For instance, in a certain classifier, it may be that $t(x) = 1$ (true) for $43 < x_3 < 54$, where x_3 is a component of x, and represents, say, the age of a medical patient. In general, a classifier's condition will refer to more than one of the input components, usually all of them. If a classifier's condition is satisfied, i.e. its $t(x) = 1$, then that classifier joins the match set and influences the system's action decision. In a sense, the match set consists of classifiers in the population that *recognize* the current input.

Among the classifiers—the condition-action rules—of the match set will be some that advocate one of the possible actions, some that advocate another of the actions, and so forth. Besides advocating an action, a classifier will also contain a *prediction* of the amount of payoff which, speaking loosely, "it thinks" will be received if the system takes that action. How can the LCS decide which action to take? Clearly, it should pick the action that is likely to receive the highest payoff, but with all the classifiers making (in general) different predictions, how can it decide? The technique adopted is to compute, for each action, an average of the predictions of the classifiers advocating that action—and then choose the action with the largest average. The prediction average is in fact weighted by another classifier quantity, its *fitness*, which will be described later but is intended to reflect the reliability of the classifier's prediction.

The LCS takes the action with the largest average prediction, and in response the environment returns some amount of payoff. If it is in a learning mode, the LCS will use this payoff, P, to alter the predictions of the responsible classifiers, namely those advocating the chosen action; they form what is called the *action set*. In this adjustment, each action set classi-

fier's prediction p is changed mathematically to bring it slightly closer to P, with the aim of increasing its accuracy. Besides its prediction, each classifier maintains an estimate ϵ of the error of its predictions. Like p, ϵ is adjusted on each learning encounter with the environment by moving ϵ slightly closer to the current absolute error $|p - P|$. Finally, a quantity called the classifier's *fitness* is adjusted by moving it closer to an inverse function of ϵ, which can be regarded as measuring the accuracy of the classifier. The result of these adjustments will hopefully be to improve the classifier's prediction and to derive a measure—the fitness—that indicates its accuracy.

The adaptivity of the LCS is not, however, limited to adjusting classifier predictions. At a deeper level, the system treats the classifiers as an evolving population in which accurate—i.e. high fitness—classifiers are reproduced over less accurate ones and the "offspring" are modified by genetic operators such as mutation and crossover. In this way, the population of classifiers gradually changes over time, that is, it adapts *structurally*. Evolution of the population is the key to high performance since the accuracy of predictions depends closely on the classifier conditions, which are changed by evolution.

Evolution takes place in the background as the system is interacting with its environment. Each time an action set is formed, there is finite chance that a genetic algorithm will occur in the set. Specifically, two classifiers are selected from the set with probabilities proportional to their fitnesses. The two are copied and the copies (offspring) may, with certain probabilities, be mutated and recombined ("crossed"). Mutation means changing, slightly, some quantity or aspect of the classifier condition; the action may also be changed to one of the other actions. Crossover means exchanging parts of the two classifiers. Then the offspring are inserted into the population and two classifiers are deleted to keep the population at a constant size. The new classifiers, in effect, compete with their parents, which are still (with high probability) in the population.

The effect of classifier evolution is to modify their conditions so as to increase the overall prediction accuracy of the population. This occurs because fitness is based on accuracy. In addition, however, the evolution leads to an increase in what can be called the "accurate generality" of the population. That is, classifier conditions evolve to be as general as possible without sacrificing accuracy. Here, general means maximizing the number of input vectors that the condition matches. The increase in generality results in the population needing fewer distinct classifiers to cover all inputs, which means (if identical classifiers are merged) that populations are smaller, and also that the knowledge contained in the population is more visible to humans—which is important in many applications. The specific mechanism by which generality increases is a major, if subtle, side-effect of the overall evolution.

Summarizing, a learning classifier system is a broadly-applicable adaptive system that learns from external reinforcement and through an internal structural evolution derived from that reinforcement. In addition to adaptively increasing its performance, the LCS develops knowledge in the form of rules that respond to different aspects of the environment and capture environmental regularities through the generality of their conditions.

Many important aspects of LCS were omitted in the above presentation, including among others: use in sequential (multi-step) tasks, modifications for non-Markov (locally ambiguous) environments, learning in the presence of noise, incorporation of continuous-valued actions, learning of relational concepts, learning of hyper-heuristics, and use for on-line function approximation and clustering. An LCS appears to be a widely applicable cognitive/agent model that can act as a framework for a diversity of learning investigations and practical applications.