An Exercise

Consider the following learning algorithm:

---

1. Create a table out of all training examples.
2. Determine which classification occurs most often in the table, and call it **d**.
3. For inputs that are in the table, return the classification associated with it, or the most common one if there are multiple ones.
4. For inputs not in the table, return **d**.

How well does this algorithm work on the training set for our restaurant waiting problem?

Can you think of a (kind of) problem for which this algorithm works well? Not well at all?

Why Does Learning Work?

Let's call this...

The Principle of Convergent Intelligence:

> The world manifests constraints and regularities. If an agent is to exhibit intelligence, then it must exploit these constraints and regularities, no matter the nature of its physical make-up.

This principle has three parts, each of which tells us something about intelligence (and so AI):

- Constraints and regularities exist in the world.
- Intelligence *implies* the ability to recognize and exploit them
- It doesn't matter how you do it, only that you do it.

Humans seem to have innate ability to recognize and exploit certain regularities (language, vision).

Learning works only if the world contains constraints and regularities. And if the world is regular, then learning is indispensable to recognizing and exploiting previously unknown regularities.

Motivation for a New Kind of Learning

The intelligent agents we know about in the universe are not “programmed”, at least not in the sense that we write programs to handle payroll.

Instead, they are born as a product of two intelligent agents already in existence, and begin with a “program” derived from the programs of their parents.

The programs of their parents were created in a similar fashion, and the chain of such creation goes back many, many years.

Furthermore, we believe that (at least way back in time) the ability to produce offspring favored those agents that could survive long enough to do so, and that the family of agents grew better able to survive over time.

Isn’t this a form of “corporate” learning?

The Genetic Metaphor

Our theories about the experience of living organisms on Earth over time caused some people interested in AI to consider how we might try to create an intelligent agent more in the style of the one existence proof that we have that intelligent agents exist.

What would it be like to program a computer by writing some simple programs, teaching them how to reproduce, and then letting the "family" of programs **evolve**?

Work in groups of three or four to "complete the metaphor" by identifying the programming equivalents of the following biological terms:

- a gene
- an allele[1]
- a population
- a genotype[2]
- a phenotype[3]
- reproduction
- natural selection

(1) any of the alternative forms of a gene that may occur at a given locus
(2) all or part of the genetic constitution of an individual or group
(3) the visible properties of an organism that are produced by the interaction of the genotype and the environment

Genetic Algorithms

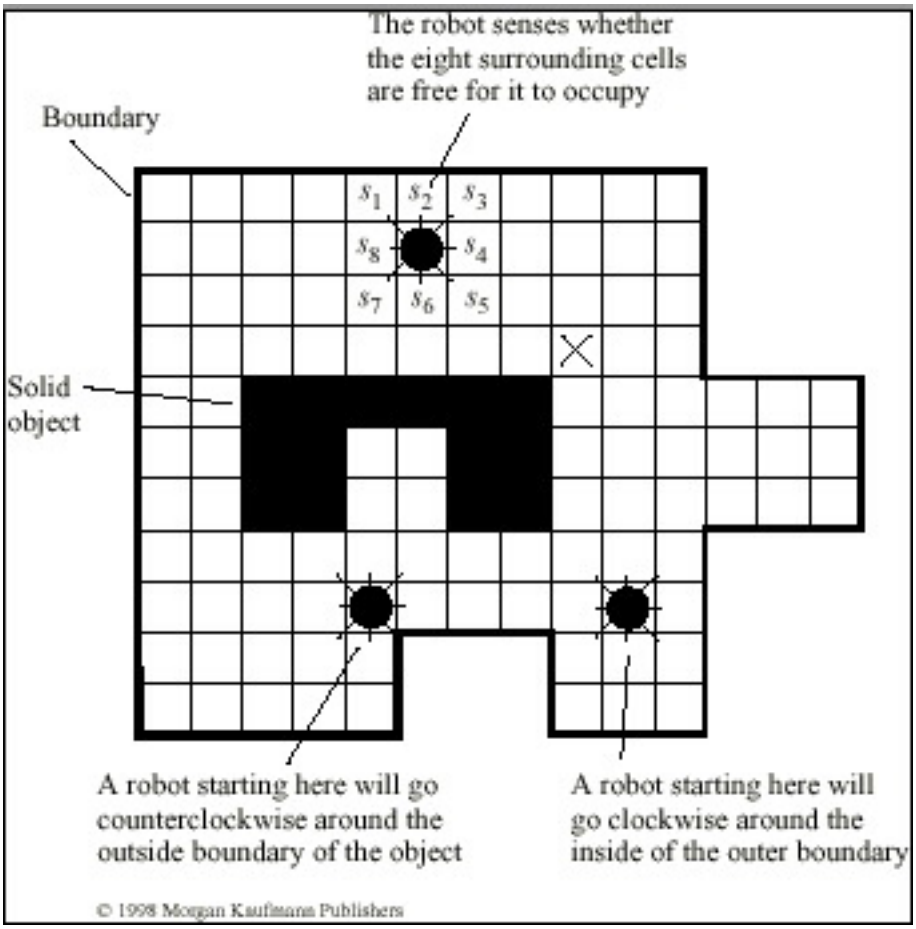
Genetic approaches to programming originated with practitioners who wanted to write programs to decide what kind of “thing” their input described.

This process is often called **classification**. Agents that do only classification can be implemented as reflex agents.

One way to do classification is to use a set of **production rules** that map inputs onto categories.

One way to write a program that uses production rules to classify is, well, to write the rules.

Boundary-Following Robots



Features:

- x1 = s2 v s3
- x2 = s4 v s5
- x3 = s6 v s7
- x4 = s8 v s1

Rules

- if x1 and not x2 then move east
- if x2 and not x3 then move south
- if x3 and not x4 then move west
- if x4 and not x1 then move north
- default action: move north

What are the Weaknesses of the Rule-Writing Approach?

Problem: lack of domain knowledge
Problem: lack of time to learn the domain knowledge
Problem: brittleness in the program

Solution: Write a program that learns better rules than the ones it currently has.

Two steps in learning:

1. Determine how good each rule is.

2. Create new rules that might be better.

The Learning Process

Start with a set of problems.
Start with a population of rules.

For some number of iterations:

a. For each rule,

- Run the rule against each problem.
- Combine the results of these runs to determine the **fitness** of the rule.

b. Construct a new population of rules from the current set based on the fitness of each rule.

The effect: a preference for successful phenotypes.

How to Generate New Rules

Direct (asexual)

A rule:	`1#00# –> 0111#`
Generalization:	`1##0# –> 0111#`
Specialization:	`1#00# –> 01110`

Crossover (sexual)

Rule 1:	`1#00# –> 0111#`
Rule 2:	`01#10 –> 11001`
Example:	`1#00# –> 11001`
Example:	`1##10 –> 1101#`

Mutation

A rule:	`1#00# –> 0111#`
Mutation!	`1#01# –> 0111#`
Mutation!	`1#00# –> 0011#`