Day 17

30 Days of Machine Learning

Experimentation Strategies & Evaluating Hypotheses

Prepared By

Ahmed Ali

Quaid-e-Awam University of Engineering Science & Technology Nawabshah

Experimentation Strategies:

The Q learning algorithm does not specify how actions are chosen by the agent.

• One obvious strategy would be for the agent in state s to select the action a that maximizes $\hat{Q}$(s, a), thereby exploiting its current approximation $\hat{Q}$.

• However, with this strategy the agent runs the risk that it will overcommit to actions that are found during early training to have high Q values, while failing to explore other actions that have even higher values.

• For this reason, Q learning uses a probabilistic approach to selecting actions. Actions with higher $\hat{Q}$ values are assigned higher probabilities, but every action is assigned a nonzero probability.

• One way to assign such probabilities is.

$$P(a_i|s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

• Where, P(ai |s) is the probability of selecting action ai, given that the agent is in state s, and k > 0 is a constant that determines how strongly the selection favors actions with high $\hat{Q}$values.

Evaluating Hypotheses:

Motivation:

It is important to evaluate the performance of learned hypotheses as precisely as possible.

- One reason is simply to understand whether to use the hypothesis.
- A second reason is that evaluating hypotheses is an integral component of many learning methods.

Two key difficulties arise:while learning a hypothesis and estimating its future accuracy given only a limited set of data:

1. Bias in the estimate.

The observed accuracy of the learned hypothesis over the training examples is often a poor estimator of its accuracy over future examples. Because the learned hypothesis was derived from these examples, they will typically provide an optimistically biased estimate of hypothesis accuracy over future examples. This is especially likely when the learner considers a very rich hypothesis space, enabling it to overfit the training examples. To obtain an unbiased estimate of future accuracy, test the hypothesis on some set of test examples chosen independently of the training examples and the hypothesis.

2. Variance in the estimate:

Even if the hypothesis accuracy is measured over an unbiased set of test examples independent of the training examples, the measured accuracy can still vary from the true accuracy, depending on the makeup of the set of test examples. The smaller the set of test examples, the greater the expected variance.

Estimating Hypothesis Accuracy:

Sample Error –

The sample error of a hypothesis with respect to some sample S of instances drawn from X is the fraction of S that it misclassifies.

Definition: The sample error (errors(h)) of hypothesis h with respect to target function f and data sample S is

$$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$

Where n is the number of examples in S, and the quantity δ(f(x), h(x)) is 1 if f (x) ≠ h(x), and 0 otherwise.

True Error –

The true error of a hypothesis is the probability that it will misclassify a single randomly drawn instance from the distribution **D**.

Definition: The true error (error **D** (**h**)) of hypothesis h with respect to target function f and distribution **D**, is the probability that h will misclassify an instance drawn at random according to **D**.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$

Confidence Intervals for Discrete-Valued Hypotheses

Suppose we wish to estimate the true error for some discrete valued hypothesis h, based on its observed sample error over a sample S, where

• The sample S contains n examples drawn independent of one another, and independent of h, according to the probability distribution D

• n ≥ 30

• Hypothesis h commits r errors over these n examples (i.e., errors (h) = r/n).

Under these conditions, statistical theory allows to make the following assertions:

1. Given no other information, the most probable value of error***D*** (h) is errors(h)

2. With approximately ***95% probability***, the true error error***D*** (h) lies in the interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Example:

Suppose the data sample S contains ***n = 40*** examples and that hypothesis h commits ***r = 12*** errors over this data.

- The sample error is errors(h) = ***r/n = 12/40 = 0.30***
- Given no other information, true error is errorD (h) = errors(h), ***i.e., errorD (h) = 0.30***
- With the ***95% confidence*** interval estimate for errorD (h).

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

= 0.30 ± (1.96 * 0.07)
= 0.30 ± 0.14

3. A different constant, ***ZN***, is used to calculate the ***N% confidence interval***. The general expression for approximate N% confidence intervals for error***D*** (h) is

$$error_S(h) \pm z_N\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Where

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

The above equation describes how to calculate the confidence intervals, or error bars, for estimates of error***D*** (h) that are based on errors(h)

Example: Suppose the data sample S contains ***n = 40*** examples and that hypothesis h commits ***r = 12*** errors over this data.

- The sample error is errors(h) ***= r/n = 12/40 = 0.30***
- With the ***68% confidenc***e interval estimate for error***D*** (h).

$$error_S(h) \pm 1.00\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

= 0.30 ± (1.00 * 0.07)
= 0.30 ± 0.07

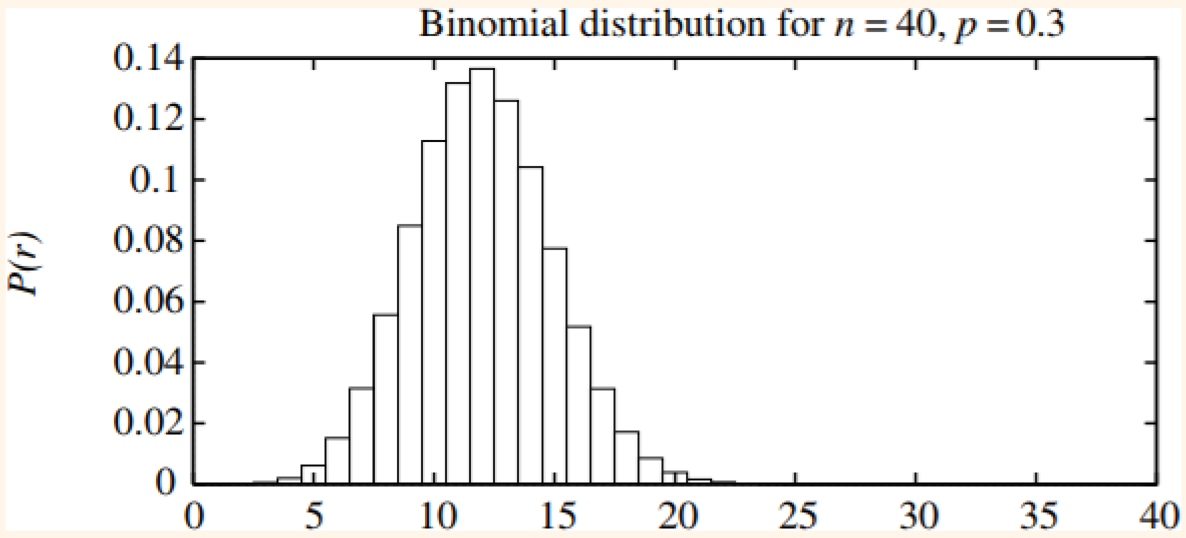
Basics of Sampling Theory

Error Estimation and Estimating Binomial Proportions

• Collect a random sample S of n independently drawn instances from the distribution D, and then measure the sample error errors(h). Repeat this experiment many times, each time drawing a different random sample Si of size n, we would expect to observe different values for the various errorsi(h), depending on random differences in the makeup of the various Si. We say that errorsi(h), the outcome of the ith such experiment, is a ***random variable.***

• Imagine that we were to run k random experiments, measuring the random variables errors1(h), errors2(h) . . . errorssk(h) and plotted a histogram displaying the frequency with which each possible error value is observed.

• As k grows, the histogram would approach a particular probability distribution called the Binomial distribution which is shown in below figure.



A Binomial distribution is defined by the probability function

$$P(r) = \frac{n!}{r!(n-r)!}\, p^r (1-p)^{n-r}$$

If the random variable ***X*** follows a Binomial distribution, then:

• The probability ***Pr(X = r)*** that X will take on the value ***r*** is given by P(r)

Ahmed Ali

- Expected, or mean value of X, $E[X]$, is
$$E[X] \equiv \sum_{i=0}^{n} iP(i) = np$$
- Variance of X is
$$Var(X) \equiv E[(X - E[X])^2] = np(1-p)$$
- Standard deviation of X, σ_X, is
$$\sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1-p)}$$

The Binomial Distribution

Consider the following problem for better understanding of Binomial Distribution

- Given a worn and bent coin and estimate the probability that the coin will turn up heads when tossed.
- Unknown probability of heads ***p.*** Toss the coin ***n*** times and record the number of times ***r*** that it turns up heads. Estimate of p ***= r / n***
- If the experiment were rerun, generating a new set of ***n*** coin tosses, we might expect the number of heads ***r*** to vary somewhat from the value measured in the first experiment, yielding a somewhat different estimate for ***p***.
- The Binomial distribution describes for each possible value of ***r*** (***i.e., from 0 to n***), the probability of observing exactly ***r*** heads given a sample of ***n*** independent tosses of a coin whose true probability of heads is ***p***.

The general setting to which the Binomial distribution applies is:

1. There is a base experiment (e.g., toss of the coin) whose outcome can be described by a random variable 'Y'. The random variable Y can take on two possible values (e.g., Y = 1 if heads, Y = 0 if tails).

2. The probability that Y = 1 on any single trial of the base experiment is given by some constant p, independent of the outcome of any other experiment. The probability that Y = 0 is therefore (1 - p). Typically, p is not known in advance, and the problem is to estimate it.

3. A series of n independent trials of the underlying experiment is performed (e.g., n independent coin tosses), producing the sequence of independent, identically distributed random variables Y1, Y2, . . ., Yn. Let R denote the number of trials for which Yi = 1 in this series of n experiments

$$R \equiv \sum_{i=1}^{n} Y_i$$

4. The probability that the random variable R will take on a specific value r (e.g., the probability of observing exactly r heads) is given by the Binomial distribution

$$\Pr(R = r) = \frac{n!}{r!(n-r)!}\, p^r (1-p)^{n-r} \qquad \textbf{equ (1)}$$

Ahmed Ali

Mean, Variance and Standard Deviation

The Mean (expected value) is the average of the values taken on by repeatedly sampling the random variable

Definition:

Consider a random variable Y that takes on the possible values y1, . . . yn. The expected value (Mean) of Y, E[Y], is

$$E[Y] \equiv \sum_{i=1}^{n} y_i \Pr(Y = y_i)$$

The Variance captures how far the random variable is expected to vary from its mean value.

Definition: The variance of a random variable Y, Var[Y], is

$$Var[Y] \equiv E[(Y - E[Y])^2]$$

The variance describes the expected squared error in using a single observation of Y to estimate its mean E[Y].

The square root of the variance is called the standard deviation of Y, denoted σy

Definition: The standard deviation of a random variable Y, σy, is

$$\sigma_Y \equiv \sqrt{E[(Y - E[Y])^2]}$$

In case the random variable Y is governed by a Binomial distribution, then the Mean, Variance and standard deviation are given by

$$E[Y] = np$$

$$Var[Y] = np(1-p)$$

$$\sigma_Y = \sqrt{np(1-p)}$$

Estimators, Bias, and Variance

Let us describe errors(h) and errorD(h) using the terms in Equation (1) defining the Binomial distribution. We then have

$$error_S(h) = \frac{r}{n}$$

$$error_{\mathcal{D}}(h) = p$$

Where,

- n is the number of instances in the sample S,

• r is the number of instances from S misclassified by h

• p is the probability of misclassifying a single instance drawn from D

• Estimator: errors(h) an estimator for the true error error***D***(h): An estimator is any random variable used to estimate some parameter of the underlying population from which the sample is drawn

• Estimation bias: is the difference between the expected value of the estimator and the true value of the parameter.

Definition: The estimation bias of an estimator ***Y*** for an arbitrary parameter ***p*** is

$$E[Y] - p$$

Genetic Algorithms:

Motivation

Genetic algorithms (GAS) provide a learning method motivated by an analogy to biological evolution. Rather than search from general-to-specific hypotheses, or from simple-to-complex, GAS generate successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses. At each step, a collection of hypotheses called the current population is updated by replacing some fraction of the population by offspring of the most fit current hypotheses. The process forms a generate-and-test beam-search of hypotheses, in which variants of the best current hypotheses are most likely to be considered next.

The popularity of GAS is motivated by a few factors including:

• Evolution is known to be a successful, robust method for adaptation within biological systems.

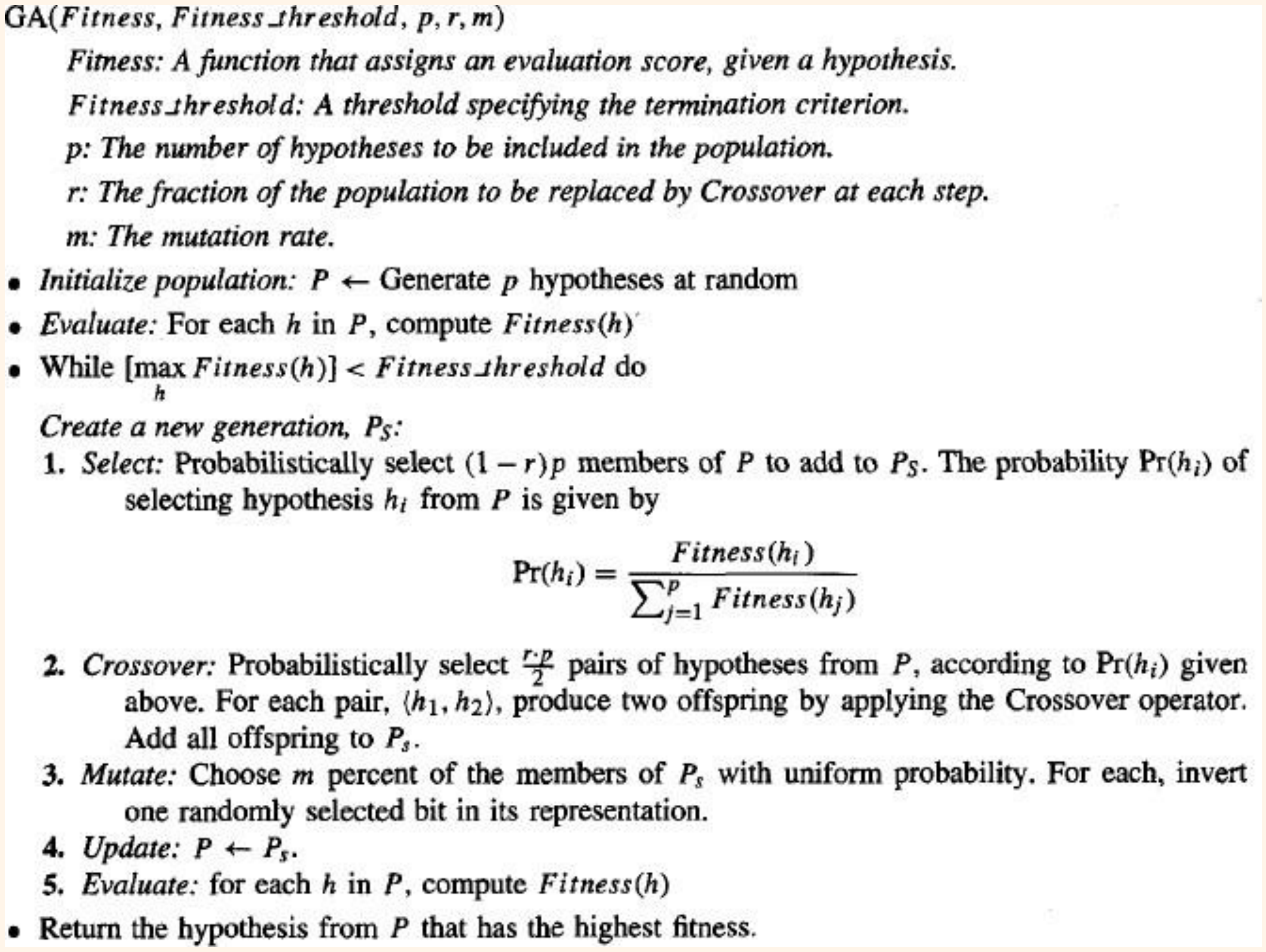
• GAS can search spaces of hypotheses containing complex interacting parts, where the impact of each part on overall hypothesis fitness may be difficult to model.

• Genetic algorithms are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware.

Genetic Algorithms

The problem addressed by GAS is to search a space of candidate hypotheses to identify the best hypothesis. In GAS the "best hypothesis" is defined as the one that optimizes a predefined numerical measure for the problem at hand, called b the hypothesis fitness. For example, if the learning task is the problem of approximating an unknown function given training examples of its input and output, then fitness could be defined as the accuracy of the hypothesis over this training data. If the task is to learn a strategy for playing chess, fitness could be defined as the number of games won by the individual when playing against other individuals in the current population.

Although different implementations of genetic algorithms vary in their details, they typically share the following structure: The algorithm operates by iteratively updating a pool of hypotheses, called the population. On each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the fit individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation.



```
GA($Fitness, Fitness\_threshold, p, r, m$)
    Fitness: A function that assigns an evaluation score, given a hypothesis.
    Fitness_threshold: A threshold specifying the termination criterion.
    p: The number of hypotheses to be included in the population.
    r: The fraction of the population to be replaced by Crossover at each step.
    m: The mutation rate.
• Initialize population: $P \leftarrow$ Generate $p$ hypotheses at random
• Evaluate: For each $h$ in $P$, compute $Fitness(h)$
• While $[\max_h Fitness(h)] < Fitness\_threshold$ do
  Create a new generation, $P_S$:
  1. Select: Probabilistically select $(1-r)p$ members of $P$ to add to $P_S$. The probability $\Pr(h_i)$ of
        selecting hypothesis $h_i$ from $P$ is given by
                 $\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$
  2. Crossover: Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from $P$, according to $\Pr(h_i)$ given
        above. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator.
        Add all offspring to $P_s$.
  3. Mutate: Choose $m$ percent of the members of $P_s$ with uniform probability. For each, invert
        one randomly selected bit in its representation.
  4. Update: $P \leftarrow P_s$.
  5. Evaluate: for each $h$ in $P$, compute $Fitness(h)$
• Return the hypothesis from $P$ that has the highest fitness.
```

The inputs to this algorithm include the fitness function for ranking candidate hypotheses, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate. Notice in this algorithm each iteration through the main loop produces a new generation of hypotheses based on the current population. First, a certain number of hypotheses from the current population are selected for inclusion in the next generation. These are selected probabilistically, where the probability of selecting hypothesis hi is given by

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

Ahmed Ali

Thus, the probability that a hypothesis will be selected is proportional to its own fitness and is inversely proportional to the fitness of the other competing hypotheses in the current population. Once these members of the current generation have been selected for inclusion in the next generation population, additional members are generated using a crossover operation. Crossover, defined in detail in the next section, takes two parent hypotheses from the current generation and creates two offspring hypotheses by recombining portions of both parents. The parent hypotheses are chosen probabilistically from the current population, again using the probability function given by Equation (9.1). After new members have been created by this crossover operation, the new generation population now contains the 102 desired number of members. At this point, a certain fraction m of these members is chosen at random, and random mutations all performed to alter these members.

This GA algorithm thus performs a randomized, parallel beam search for hypotheses that perform well according to the fitness function. In the following subsections, we describe in more detail the representation of hypotheses and genetic operators used in this algorithm. Representing Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. The hypotheses represented by these bit strings can be quite complex. For example, sets of if-then rules can easily be represented in this way, by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondition.

To see how if-then rules can be encoded by bit strings, first consider how we might use a bit string to describe a constraint on the value of a single attribute. To pick an example, consider the attribute ***Outlook***, which can take on any of the three values ***Sunny***, ***Overcast***, or ***Rain***. One obvious way to represent a constraint on ***Outlook*** is to use a bit string of length three, in which each bit position corresponds to one of its three possible values. Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value. For example, the string 010 represents the constraint that ***Outlook*** must take on the second of these values, or ***Outlook = Overcast***. Similarly, the string 011 represents the more general constraint that allows two possible values, or ***(Outlook = Overcast v Rain)***.

Note 11 1 represents the most general possible constraint, indicating that we don't care which of its possible values the attribute takes on.

Given this method for representing constraints on a single attribute, conjunctions of constraints on multiple attributes can easily be represented by concatenating the corresponding bit strings. For example, consider a second attribute, Wind, that can take on the value Strong or Weak. A rule precondition such as:

(Outlook = Overcast ^Rain)

A (Wind = Strong) can then be represented by the following bit string of length five:

Outlook	***Wind***
01 1	***10***

Rule postconditions (such as ***PlayTennis = yes***) can be represented in a similar fashion. Thus, an entire rule can be described by concatenating the bit strings describing the rule preconditions, together with the bit string describing the rule postcondition. For example, the rule

IF Wind = Strong THEN PlayTennis = yes

Ahmed Ali

would be represented by the string.

Outlook	***Wind***	***PlayTennis***
111	***10***	***10***

where the first three bits describe the "don't care" constraint on ***Outlook*****,** the next two bits describe the constraint on ***Wind*****,** and the final two bits describe the rule postcondition (here we assume ***PlayTennis*** can take on the values ***Yes*** or ***No***). Note the bit string representing the rule contains a substring for each attribute in the hypothesis space, even if that attribute is not constrained by the rule preconditions.

This yields a fixed length bit-string representation for rules, in which substrings at specific locations describe constraints on specific attributes. Given this representation for single rules, we can represent sets of rules by similarly concatenating the bit string representations of the individual rules. In designing a bit string encoding for some hypothesis space, it is useful to arrange for every syntactically legal bit string to represent a well-defined hypothesis. To illustrate, note in the rule encoding in the above paragraph the bit string 11 1 10 11 represents a rule whose postcondition does not constrain the target attribute ***PlayTennis***.

If we wish to avoid considering this hypothesis, we may employ a different encoding (e.g., allocate just one bit to the ***PlayTennis*** postcondition to indicate whether the value is ***Yes*** or ***No***), alter the genetic operators so that they explicitly avoid constructing such bit strings, or simply assign a very low fitness to such bit strings. In some GAS, hypotheses are represented by symbolic descriptions rather than bit strings.

Up next:

Genetic Operators.