The goal of today’s lab is to implement the bootstrap algorithm described in section 4.9 of the draft third edition of *Speech and Natural Language Processing* and presented in pseudocode in Figure 4.9 of that textbook.

**Scoring a Classification Algorithm**

We use the bootstrap algorithm when trying to compare the performance of two different classifiers on the same set of test cases for which the correct classifications are known. We’ll assume that the test set is presented as a non-empty list of pairs in which the car of each pair is an input to the classifier and the cdr is a symbol or a natural number giving the correct classification. We’ll also assume that each classifier is implemented as a Scheme procedure that takes one argument and returns a symbol or a natural number representing the classifier’s attempt to determine the correct classification of the argument.

1. Design, write, and test a scoring procedure that takes as its arguments a classifier and a test set and returns a rational number in the range from 0 to 1 indicating the fraction of items in the test set that the classifier correctly classified. (Note: For purposes of testing this procedure, it is not necessary to code up realistic test data or realistic classifiers.)

2. Design, write, and test a procedure that takes as its arguments two classifiers (A and B) and a test set \(x\) and computes \(\delta(x)\): the amount by which A’s score on the test set exceeds B’s.

**Choosing a Random Sample**

The authors say that each “bootstrap sample” should be a sample “with replacement” of the test set, which is supposed to be considerably larger than the samples. (“With replacement” means that each element of the sample should be selected independently and at random from the test set, without worrying about whether the same item might be selected more than once.)

3. Define, write, and test a procedure that takes a non-empty list as its argument and returns a randomly selected element of that list. (Hint: When the `random` procedure, which is in the `(csc-151 random)` library, is given a positive integer \(n\) as its argument, it returns a randomly selected natural number less than \(n\).)

4. Let’s say that each bootstrap sample should be about one-fifth the size of the test set from which it is drawn. Define, write, and test a procedure that takes a test set as argument and returns a bootstrap sample from it. (If the size of the test set is not an exact multiple of 5, round up to get the appropriate size for a sample. We want the procedure never to return a null list as a sample.)

**The Bootstrap Algorithm**

5. Define, write, and test a procedure that takes as arguments two classifiers (A and B), a test set \(x\), and a positive integer \(b\) indicating how many bootstrap samples to take. The procedure should first compute \(\delta(x)\) (using the procedure you defined earlier for this purpose). It should then take \(b\) bootstrap samples \(x^{*}(i)\) from \(x\), compute \(\delta(x^{*}(i))\) for each one, and determine the number \(s\) of samples for which \(\delta(x^{*}(i)) > 2\delta(x)\). (The pseudocode in Figure 4.9 assumes that all of the \(\delta(x^{*}(i))\) values are saved as they are computed and then calculates \(s\) in a separate loop, but this isn’t strictly necessary.) Finally, your procedure should return \(s/b\) as the estimate of the probability that the null hypothesis is correct, that is, that A is not significantly better at the classification test than B is.