Scientific Communication CITS7200

Computer Science & Software Engineering

Lecture 9

Designing and analysing experiments

1 Introduction

Computer Science is a very broad discipline, but at the core of all its activities is a commitment to the *science* of computation and the *engineering* of software. In this lecture we are going to focus on the scientific method, since this is most closely related to the research activities of computer scientists. At the heart of the scientific method is the hypothesis and the experiment. The notes for this lecture follow closely the developments of Justin Zobel [1] and Kevin Korb [2].

We use experiments to verify hypotheses: this is really what science is all about. Recall that a hypothesis is

> ... a supposition or conjecture put forth to account for known facts.

However, until a hypothesis has been tested it is not a law—it remains a plausible explanation of the nature of things.

In computing, experiments are generally used to verify hypotheses about algorithms. The hypotheses might relate to the performance of a specific task, or to the computational efficiency or scalability. In all cases, the communication of experiments must be done in such a way that the experiment can

be repeated by others. This repetition is known as *validation*: along with the formulation of hypotheses and experimentation, validation is the other key ingredient to the scientific process.

Validation by experiment is often an important part of the presentation of algorithms. The experiment provides concrete evidence that, for some data, the algorithm terminates correctly and performs as predicted.

Thus, a reader would expect to find some or all of the following:

- The steps that make up the algorithm.
- The input and output, and the internal data structures used by the algorithm.
- The scope of application of the algorithm and its limitations.
- The properties that will allow demonstration of correctness, such as preconditions, postconditions, and loop invariants.
- A demonstration of correctness.
- A complexity analysis, for both space and time requirements.
- Experiments confirming the theoretical results.

2 Stating the hypothesis

The hypothesis needs to be stated clearly, precisely, and unambiguously. If this is not done, the reader may even be able to see that the hypothesis is satisfied by mutually contradictory things. In stating the hypothesis it can often be helpful to clarify what is *not* being tested as well. Such bounds help immensely in the validation process.

Let's consider an example. Suppose P-lists are a well-known data structure used in a range of applications but particularly as an in-memory search structure that is fast and compact. Suppose you develop a new data structure, called Q-lists. You do a formal complexity analysis and discover that both structures exhibit the same asymptotic complexity in space and time, but on all the tests you have run, the Q-lists perform better than the P-lists.

You might state your hypothesis as

> Q-lists are superior to P-lists.

Such a statement would be hopeless for experimental verification because it would have to apply in all cases, in all applications, under all conditions, and for all time. It could never be tested to be true, and one single counterexample would crush the hypothesis for ever.

A more testable hypothesis might be

> As an in-memory search structure for large data sets, Q-lists are faster and more compact than P-lists.

Now the hypothesis qualifies the scope of the claims you are making. You would test both data-structures on ever-larger data sets and show a clear trend, with the performance graph of run time versus input size for Q-lists always lying below that of P-lists after a certain input size. Of course, "always" is also a dangerous word, for you can't test data sets of arbitrarily large size, but you can test enough to show a clear trend. Having no evidence that the trend is likely to alter, the hypothesis will stand as the *simplest* explanation of the data. This is known as invoking Occam's Razor.

3 Developing hypotheses

After some initial testing you might find that your original hypothesis is not quite correct. Of course, there is always the possibility that your original hypothesis was entirely wrong: this can be as exciting as if you can analytically prove that it is entirely true in all cases. Many great discoveries in science have occurred by the complete falsification of the original hypothesis (for example, Columbus proved that the Earth is not flat!)

So your hypothesis may need some alteration, but this process should not be an incremental following of a series of experiments. For example, the original hypothesis does not become

> Q-lists are superior to P-lists, except for data sets of size greater than $2,000,000$.

A hypothesis will often be based on observation, but it can only be considered as confirmed once it is able to successfully make predictions. It is this power of predictions that makes science as powerful as it has become.

Thus, we distinguish between "the algorithm worked on our data", and "the algorithm was predicted to work on any data in this class, and this prediction has been confirmed on our data". Typically, we partition the "world" of data

into (usually) independent sets, choose some representative examples from each set, test those (successfully), and claim therefore that the hypothesis is true for the world.

Another way of saying this is to say that tests should be blind. The hypothesis should not be tested on the data from which it arose.

4 Argumentation

For the successful presentation of a hypothesis, you need to construct an argument relating your hypothesis to the evidence. Argumentation is a critical part of all research writing.

In the construction of any argument, or the analysis of anyone else's argument you need firstly to identify the *conclusion*. The next question is: what *premises* are offered in support of this conclusion? New ideas that are not supported by argument will be lost to the scientific community, and unargued new ideas will not be believed.

Now, an argument is *valid* if and only if its premises logically imply its conclusions—that is, if it is *impossible* for its conclusions to be false if its premises are true; an argument is also *sound* if additionally its premises are true.

Thus, the argument

> Today is Friday and thus tomorrow is a holiday.

is valid but not sound, because the premise ("Today is Friday") is false—today is Wednesday.

Nevertheless, outside of mathematics it is very difficult to develop arguments that stand up to the rigour of logical analysis. Often this is because it is difficult or impossible to list all the premises to an argument. In such a case it is always best to apply the *principle of charity*: if the step from an invalid argument to a valid argument requires a suppressed premise that is known to be true, or likely to be true, or non-controversial, then assume that the argument contains that premise. However, it is wise to apply also the *principle of minimality*: if the argument needs only some particular assumption to become valid, then do not assume anything beyond that.

Note that a valid argument can fail to be sound in only one way: one of its premises is false. By explicitly formulating all the premises required for an

argument to become valid, you will automatically be making any weaknesses in the argument explicit.

A good way to do this is to imagine yourself defending your hypothesis to a colleague, so that you play the role of the inquisitor or the examiner. For example, say the hypothesis is

> The new string hashing argument is fast because it doesn't use multiplication or division.

- Why are multiplication and division an issue?

  On most machines they use several cycles, or may not be implemented in hardware at all. The new algorithm instead uses two exclusive-or operations per character and a modulo in the final step. I agree that for pipelined machines with floating-point accelerators that the difference might not be great.

- Modulo isn't always in hardware either.

  True, but it is required only once.

- So there is also an array lookup? That can be slow.

  Not if the array is cache resident.

- What happens if the hash table size is not 2^8?

  Good point. This function is most effective for hash tables of size 2^8, 2^{16} *etc.*

Such reasoning can be useful in writing up your argument in your paper. If you find an objection that you can't refute, don't leave it out of your paper; at the very least you should mention it, but you really might need to reconsider your whole argument.

Test your hypothesis in a preliminary way by seeing if there is any simple reason for keeping or discarding it. For example, are there any improbable consequences if your hypothesis is true? If so, then it's probably wrong. Does your hypothesis displace or contradict some currently-held belief? If so, then again it may be wrong: remember the rule that "extraordinary claims require extraordinary evidence". Does your hypothesis cover all the observations explained by some currently-held belief? If so, then your hypothesis is probably uninteresting.

Michael Scriven has developed a list of the steps required in the analysis of an argument:

1. Clarification of meanings, including the use of dictionaries and encyclopedias where appropriate. Removing ambiguities of terms. Identifying and labeling propositions.

2. Identification of conclusions, stated and unstated.

3. Portrayal of the argument in graphical form.

4. Formulation of unstated premises. This is the most difficult step: it requires creativity and relevant background knowledge to find the premises most appropriate to produce a valid argument. This is also the step most prone to bias and emotion.

5. Criticism of the inferences and premises—try to find counterexamples or, in the case of inductive arguments (see below), evidence of bias.

6. Introduction of other relevant arguments.

7. Overall evaluation. How good is your argument? How much would you bet on it? At what odds?

Inductive arguments

Deductive arguments require the extremely strong condition that it is strictly impossible for the conclusion to be false. Inductive arguments are more common in science, and still require analysis. Sherlock Holmes, often attributed with great deductive reasoning, in fact used inductive reasoning. Here is a typical example: Holmes deduces that the passenger sitting opposite him in the train is a teacher, because he has chalk on his hands.

The argument goes as follows:

1. *This man has white powder on his hands.*

2. *White powder that looks like chalk is chalk.*

3. This man has chalk on his hands.

4. *Most people with chalk on their hands are teachers.*

5. This man is a teacher.

Items 1 and 2 make explicit Holmes' induction in the observation; item 4 is the main suppressed premise. As a deductive argument this argument is manifestly flawed: not all white powder that looks like chalk is chalk, and not all people with chalk on their hands are teachers. As an inductive argument though it is not bad, as items 2 and 4 are highly probable premises (at least in Holmes' day). Thus, to criticise the inductive strength of an argument you must show that the truth of the premises would fail to make the conclusion highly probable.

There are various kinds of inductive argument:

- **Direct inference** This involves inferring from the proportion of individuals in a class having some property the probability of a particular individual having that property, for example "99% of birds can fly, and this is a bird, therefore it can fly". Such an inference is defeated by an additional biasing factor: you might have a penguin.

- **Inverse inference** This inference proceeds in the reverse direction from a sample of individuals to characteristics of the whole population, for example "99% of these 100 birds can fly, therefore 99% of all birds can fly". It is common in political surveys. Such inferences can be defeated if the sampling procedure can be shown to be biased: again, you might have penguins.

- **Analogies** Analogies provide the basis for many inductive arguments. By drawing attention to similarities between structural or law-like features of two systems, some support may be found for claiming that a further, unobserved feature of one is likely to be similar to a corresponding feature of the other. For example, some argue that artificial neural networks are more likely to succeed in AI than rule-based systems because ANNs are more like the human brain. Such arguments are defeated by the use of disanalogies, that is by pointing out how the systems differ.

- **Correlations** Some kinds of inductive reasoning which may support or undermine causal theories invoke the existence of (or failure of) correlations between observed variables. Such arguments can get very complicated and need to be treated with care.

5 Supporting the hypothesis

In inductive reasoning you need to have some framework for assessing the degree to which some evidence supports a conclusion. One such framework is known as *Bayesian evaluation theory.*

Bayes' theorem gives the relation between the probability of a hypothesis (conclusion) given its evidence—$P(h|e)$—and its probability prior to any evidence $P(h)$ and its likelihood $P(e|h)$. It states

$$P(h|e) = \frac{P(e|h) \times P(h)}{P(e)} \tag{1}$$

The theorem is fine; it is the use of Bayes' theorem to evaluate hypotheses in science that is controversial. Bayesian evaluation theory says that we should judge hypotheses according to the relation above between likelihood and prior probability when the evidence is taken to be the total relevant set of observations and experimental outcomes available.

It turns out that evidence will support a hypothesis if and only if the likelihood ratio

$$\lambda(e|h) = P(e|h)/P(e|\neg h) \tag{2}$$

is greater than one. Here $\neg h$ is the negation of h.

The concept of the likelihood ratio provides a simple but effective tool for analysing the impact of evidence on conclusions (hypotheses).

6 Logical fallacies

There are many varieties of logical fallacy, and here are some:

- **Equivocation** Equivocation occurs when a term in an argument can have more than one meaning, and the argument relies on the *appearance* of validity of a term that is actually being used in another way.
- **Affirming the consequent** A valid deductive inference is *modus ponens*:
  1. If A, then B.
  2. A.
  3. Therefore, B.

It is possible to run the conditional in reverse, by denying the consequent; this results in *modus tollens*. It is perfectly correct to argue that

1. If A, then B.
2. Not B.
3. Therefore, not A.

Affirming the consequent attempts to use *modus ponens* in reverse, but without denying the consequent. It is blatantly illogical, and yet there are times when it makes sense.

- **Red Herrings and Straw Men**

  Red herrings are arguments or facts that are irrelevant to the main issue under discussion but are brought up in order to distract people from that issue.

  Straw men are what you invent when you violate the principle of charity: the attribution to your opponents of arguments which they in fact do not endorse. This is usually done in order to have a target that is easier to demolish than the real argument.

- **Appeals to Authority** Improper appeals to authority occur when

  1. the authority invoked pertains to some other domain;
  2. the expert opinion is widely disputed by other experts in the field;
  3. the expert has a history of offering unreliable opinions.

- **Ad Hominem** Ad hominem attacks are those against the person rather than the argument.

- **Stereotyped Reasoning and Prejudice** Everybody argues using stereotypes, and mostly this is not a bad thing. Inappropriate uses of stereotypes involve the failure to adjust them in the light of new information. Such behavior is called prejudice.

7 Evidence

There are four kinds of evidence that may be used to support a hypothesis: analysis or proof, modeling, simulation, and experiment.

An analysis or proof is a formal argument that the hypothesis is correct. It is a mistake to suppose that the correctness of the proof is absolute—the confidence of the proof may be high but that does not guarantee that it is free from error. However, many hypotheses are not amenable to formal proof.

A model is a mathematical description of the hypothesis and usually there is a demonstration that the hypothesis and the model correspond.

A simulation is usually an implementation or partial implementation of a simplified form of the hypothesis, in which the difficulties of a full implementation are side stepped by omission or approximation. Often a simulation involves testing the hypothesis on artificial data. Although a simulation is artificial, isolated, and conducted in a tightly controlled environment, it is one way of "grounding" the hypothesis—you can prove that your algorithm behaves correctly on hand-generated data, for example.

An experiment is a full test of the hypothesis, based on an implementation of the proposal and on real—or at least realistic—data.

8 Fair experiments

Tests should be fair rather than designed to support the hypothesis. Thus, when considering what experiments to conduct, you should identify the cases in which your hypothesis is least likely to hold. These are the interesting cases: if they are not tested then the experiments won't prove much at all. So, for example, if you design a new algorithm for robot path planning where the idea is that the robot must avoid all obstacles, you should test all the hard configurations of obstacles, not the easy ones. Make the obstacles non polygonal, non convex, nested, put the robot in a situation where no path exists and see how it performs etc.

Try to consider whether there are other possible interpretations of your results and then design further tests to eliminate these possibilities. Be particularly careful in situations where your experiments fail—you need to know why they have failed. Check to see whether your results are sensible.

Conclusions need to be supported by results, so make sure that you haven't tested a special case, rather than a general case. Be particularly careful with regard to scale.

References

[1] Justin Zobel. *Writing for Computer Science*. Springer, 1997.

[2] Kevin B. Korb. Research writing in Computer Science. Technical Report, Department of Computer Science, Monash University.