Artificial Intelligence

Empirical Evaluation of AI Systems, 2

Ian Gent
ipg@cs.st-and.ac.uk

Artificial Intelligence

Exploratory Data Analysis, or … How NOT To Do It

Tales from the Coal Face

- Those ignorant of history are doomed to repeat it
- We have committed many howlers in experiment
- We hope to help others avoid similar ones …
  - … and illustrate how easy it is to screw up!
- "How Not To Do It"
  - I Gent, S A Grant, E. MacIntyre, P Prosser, P Shaw, B M Smith, and T Walsh
  - University of Leeds Research Report, May 1997
  - Every howler we report committed by at least one of the above authors!

3

Experimental Life Cycle

- Getting Started
- Exploratory Data Analysis
- Problems with Benchmark Problems
- Analysis of Data
- Presenting Results

4

Getting Started

- To get started, you need your experimental algorithm
  - usually novel algorithm, or variant of existing one
    - e.g. new heuristic in existing search algorithm
  - novelty of algorithm should imply extra care
  - more often, encourages lax implementation
    - it's only a preliminary version
- *Don't Trust Yourself*
  - bug in innermost loop found by chance
  - all experiments re run with urgent deadline
  - curiously, sometimes bugged version was better!

5

Getting Started

- *Do Make it Fast Enough*
  - emphasis on enough
    - it's often not necessary to have optimal code
    - in lifecycle of experiment, extra coding time not won back
  - e.g. published papers with inefficient code
    - compared to state of the art
      - first version $O(N^2)$ too slow!
      - *Do Report Important Implementation Details*
    - Intermediate versions produced good results
- *Do Preserve Your Code*
  - Or end up fixing the same error twice (Do use version control!)

6

Exploratory Data Analysis

- Exploratory data analysis involves *exploration*
  - exploration of your results will suggest hypotheses
    - that more formal experiments later can confirm/refute
  - To suggest hypotheses you need the data in the first place
- *Do measure with many instruments*
  - In exploring hard problems we used our *best* algorithms
  - missed important effects in worse algorithms
    - and these might affect best algorithms on larger instances

7

Exploratory Data Analysis

- *Do vary all relevant factors*
- *Don't change two things at once*
  - Ascribed effects of heuristic to the algorithm
    - changed heuristic and algorithm at the same time
    - didn't perform factorial experiment
  - But it's not always easy/possible to do the "right" experiments if there are many factors
- *Do measure CPU time*
  - In exploratory code, CPU time often misleading
    - but can also be very informative
    - e.g. heuristic needed more search but was faster

8

Exploratory Data Analysis

- *Do Collect All Data Possible ….* (*within reason)*
  - One year Santa Claus had to repeat all our experiments
    - paper deadline just after new year!
  - We had collected number of branches in search tree
    - *but not* the number of backtracks
    - performance scaled with backtracks, not branches
    - all experiments had to be rerun
- *Don't Kill Your Machines*
  - We have got into trouble with sysadmins
    - … over experimental data we never used
  - Often the vital experiment is small and quick

9

Exploratory Data Analysis

- *Do It All Again … (or at least be able to)*
- *Do Be Paranoid*
- *Do Use The Same Problems*
  - Reproducibility is a key to science (c.f. Cold fusion)
  - Being able to do it all again makes it possible
    - e.g. storing random seeds used in experiments
    - We didn't do that and might have lost important result
  - Being paranoid allows health-checking
    - e.g. confirm that 'minor' code changes do not change results
    - "identical" implementations in C, Scheme, C, gave different results
  - Using the same problems can reduce variance

10

Problems with Benchmarks

- We've seen the possible problem of overfitting
  - remember machine learning benchmarks?
- Two common approaches are used
  - benchmark libraries
    - should include hard problems and expand over time
  - random problems
    - should include problems believed to be hard
    - allows unlimited test sets to be constructed
    - disallows "cheating" by hardwiring algorithms
    - so what's the problem?

11

Problems with Random Problems

- *Do Understand Your Problem Generator*
  - Constraint satisfaction provides an undying example
  - 40+ papers over 5 years by many authors
    - used random problems from "Models A, B, C, D"
  - All four models were "flawed"
    - Achlioptas et al, 1997
    - asymptotically almost all problems are trivial
    - brings into doubt many experimental results
      - some experiments at typical sizes affected
      - fortunately not many
  - How should we generate problems in future

12

Flawed and Flawless Problems

- Gent et al (1998) fixed flaw ….
  - Introduced "flawless" problem generation
  - defined in two equivalent ways
  - though no proof that problems are truly flawless
- Third year student at Strathclyde found new bug
  - two definitions of flawless not equivalent
- Finally we settled on final definition of flawless
  - and gave proof of asymptotic non-triviality
- So we *think* we understand the problem generator!

13

Analysis of Data

- Assuming you've got everything right so far …
  - there are still lots of mistakes to make
- *Do Look at the Raw Data*
  - Summaries obscure important aspects of behaviour
  - Many statistical measures explicitly designed to minimise effect of outliers
  - Sometimes outliers are vital
    - "exceptionally hard problems" dominate mean
    - we missed them until they hit us on the head
      - when experiments "crashed" overnight
      - old data on smaller problems showed clear behaviour

14

Analysis of Data

- *Do face up to the consequences of your results*
  - e.g. preprocessing on 450 problems
    - should "obviously" reduce search
    - reduced search 448 times
    - *increased* search 2 times
  - Forget algorithm, it's useless?
  - Or study in detail the two exceptional cases
    - and achieve new understanding of an important algorithm

15

Presentation of Results

- *Do Present Statistics*
  - It's easy to present "average" behaviour
    - We failed to understand mismatch with published data
    - our mean was different to their median!
  - Readers need better understanding of your data
    - *e.g. what was standard deviation, best, worst case?*
- *Do Report Negative Results*
  - The experiment that disappoints you …
    - might disappoint lots of others unless you report it!

16

Summary

- Empirical AI is an exacting science
  - There are many ways to do experiments wrong
- We are experts in doing experiments *badly*
- As you perform experiments, you'll make many mistakes
- Learn from those mistakes, and ours!

17

And Finally …

- … the most important advice of all?

- *Do Be Stupid*
  - (would "refreshingly naïve" sound better?)
  - nature sometimes is less subtle than you think
  - e.g. scaling of behaviour in GSAT
    - it's linear, stupid
  - e.g. understanding nature of arc consistency in CSP's
    - use a stupid algorithm, stupid

18