Lecture 1/Tip #10: Tips for doing experimental work

Data Analysis and Knowledge Discovery (DAKD-MIRI course) + IDADM course (MAI)

2013-14

Lluís A. Belanche

Experimentation

Has a goal or goals

Involves conjectures about the results

Involves algorithm design and implementation

Needs problem(s) to run the algorithm(s) on

Need to establish what is variable and what is held fixed

Amounts to running the algorithm(s) on the problem(s)

Delivers the results

Needs evaluating the results in the light of the goal(s) and conjectures

Possible goals for experimentation

Get a good solution for a given problem

Show that an algorithm is applicable to a problem or class of problems

Show that an algorithm is better than another algorithm

Find “optimal” setup for parameters of a given algorithm

Understand algorithm behavior:

* how it scales with problem size

* how it performs under extreme conditions

→ how performance is influenced by parameters

Hallmarks of a good experimental paper

- Clearly defined goals
- Large scale tests (number and size)
- Mixture of problems <u>or</u> clear statement of target problems
- Statistical analysis of results
- Reproducibility
- Whys and why nots
- Open and closed problems

Using real problems

Standard data sets in public repositories, e.g.:

UCI Machine Learning Repository
www.ics.uci.edu/~mlearn/MLRepository.html

Advantages:

Well-chosen problems and instances (?)

Much other work on these → results comparable

Disadvantages:

Not owner – might miss crucial information

Algorithms may get tuned for popular problems

Using synthetic problems

Problem instance generators to produce simulated data

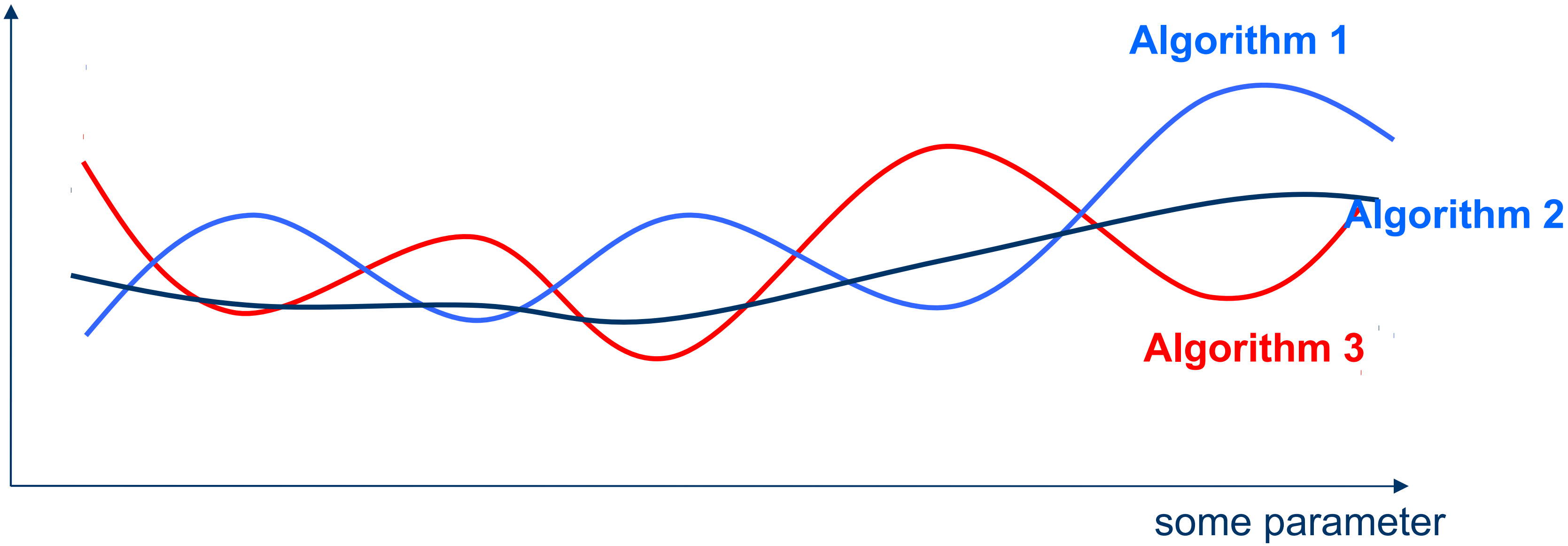
Advantages:

- Allow very systematic comparisons for they
  - can produce many instances with the same characteristics
  - enable gradual changes for many characteristics (hardness)
- Can be shared allowing comparisons with other researchers

Disadvantages

- Not the “real thing” – might be too simplistic (or too hard!)
- The generation process might have a hidden bias

Example



What algorithm is better? Why? When?

find a very good solution in specific situations

find a fairly good solution almost always

Basic rules of experimentation (I)

- Never draw any conclusion from a single run
  - perform sufficient number of independent runs
  - use statistical measures (averages, standard deviations)
  - use statistical tests to assess reliability of conclusions
- Always do a fair competition
  - use the same amount of resources for the competitors
  - use the same performance measures
  - use the same implementation decisions, specially if time/space usage is also being compared

Basic rules of experimentation (II)

- Some algorithms are stochastic
- We deal with finite data samples
- We need (hyper)parameter tuning
- For scientific papers, our procedure must meet scientific standards, specially if a claim is made
- Need to show statistical significance

Bad example (I)

I invented “the tricky machine learning algorithm”

Showed that it is a good idea by:

Running standard (?) MLA and tricky MLA

Used 10 objective functions (randomly) from the literature

Finding tricky MLA better on 7, equal on 1, worse on 2 cases ☺

I wrote everything down in a paper

And I got it published!

Bad example (II)

What did I learn from this experience?
Is this good work?
What did I the readers learn?

How relevant are these results? How trustworthy/honest are they?
What is the scope of claims about the superiority of the tricky MLA?
Is there a property distinguishing the 7 good and the 2 bad functions?
Are my results generalizable? (is the tricky MLA applicable to other problems? If so, which ones? And why these ones?)

Good example (I)

I invented my MLA for problem X

Looked and found 3 other MLAs and a traditional heuristic (used as benchmark) for problem X in the literature

Asked myself when and why is my MLA better than any/some of these

Good example (II)

- Found/made problem instance generator for problem X with 2 parameters:
  - *n* (problem size)
  - *k* (some problem specific indicator)
- Selected 5 values for *k* and 5 values for *n*
- Generated 100 problem instances for all combinations
- Executed all algorithms on each instance 100 times (since they were also stochastic)
- Recorded performance values w and w/o same computational resources
- Put my program code and the instances on the Web

Good example (III)

Arranged results "in 3D" (n,k) + performance

(with special attention to the effect of n, as for scale-up)

Assessed statistical significance of results

Found the "niche" for my MLA:

Weak in … cases, strong in - - - cases, comparable otherwise

Thereby I answered the "when question"

Analyzed the specific features and the niches of each algorithm thus answering the "why question"

Learned a lot about problem X and its solvers

Achieved generalizable results, or at least claims with well-identified scope based on solid data

Facilitated reproducing my results → further research

Some tips (I)

- ✓Be organized: devise an experiment plan
- ✓Decide what you want & define appropriate measures
- ✓Choose test problems carefully
- ✓Explain performance (why it is good/bad, under which conditions it might be better/worse)
- ✓Perform sufficient number of runs
- ✓Keep all experimental data (never throw away anything)
- ✓Use good statistics (“standard” tools)
- ✓Present results properly (figures, graphs, tables, …)
- ✓Watch the scope of your claims, but aim at general results
- ✓Publish code for reproducibility of results (if applicable)

Some tips (II)

- ✓Offer questions & hypotheses. Devise, test and refine them in light of the experimental results

- ✓If an experiment shows a lot of variability, standard tests (e.g. for the difference between two means) are not significant ---> you should try to reduce this variability:
  - ✓ Why do I get such variance? Extreme cases?
  - ✓ Devise new experiments

Distinguishing between good and reliable results

Better modeling methods and/or better data pre-processing lead to <u>better results</u> (e.g. lower modeling errors)

Better/more data leads to <u>more reliable results</u> (and narrower confidence intervals for them)

Example: 80%±7% vs. 78%±2%