Learning (and reliability)

Matti Kääriäinen (ICSI)

Talk in ToNC workshop, March 16, 2006, Berkeley.

On Reliability

From the ToNC draft:

- “5 nines” rule is overkill
- Users would be happy to pay less and get less, as long as they can accomplish their goals most of the time

Thus, limited resources should be targeted on ensuring good quality of service to the requests the users really care about

Theme of talk: How to approach reliability (or QoS) using ideas from learning theory?

Non-adaptive approach to reliability

1. Make conservative assumptions on
    - Requests to system
    - System's capabilities
2. Design system so that it works under the conservative assumptions
3. Implement and deploy system

Non-adaptive, since all assumptions are static.

Examples of non-adaptive reliability

- Roof
  - Assumptions on building materials and weather
  - Construction engineering knowledge
- Telephone network
  - Assumptions on component systems and user demands
  - Telecommunications engineering knowledge
- Internet — fundamentally different (?)

Limitations of the non-adaptive approach

- Requires good understanding of
  - Demands to the system
  - Subsystems
  - How subsystems behave when combined
- If understanding is limited or faulty, then
  - Price of conservatism increases, and/or
  - Reliability guarantees no more met
- Non-adaptive — hard to handle, e.g., demands that change in time

Learning

Instead of the non-adaptive approach, one can try to learn:

- A model of demands to system/user behavior
- A model of how the system works (less emphasized here)
- Which aspects of system behavior are valuable to users

Or, skip (some of) the above:

- Learn a system that satisfies users' demands

Adaptive approach

- Basic idea:
  1. Deploy an initial system
  2. Gather data on how it is used and how well it works
  3. Tune or rebuild the system
- Single/multiple iterations $\longrightarrow$ static/non-static solution
- Reliability guarantees for
  - Final tuned system
  - System’s performance during tuning (non-static case)
- Intimately connected to batch/online learning

Static setting I

Assume:	Collected some data of past interaction of users and the system
Goal:	Design a system with good performance
Solution:	Find the design that has best performance on collected data

The solution is obviously wrong (overfitting) — goal to perform well on future service requests, not the past

Static setting II

Assume:	The collected data modelled as a random sample from an unknown distribution that models the (past and future) demands to the system
Goal:	Design a system with good performance
Solution:	Find a design that has good expected performance

Making a distinction between observations and the “truth” a necessary first step in addressing the overfitting problem.

Concrete example of overfitting

Task: Learn how network topology should be changed to improve cost/quality ratio on data transmission requests by users

Model the users' requests, e.g., as iid samples of

- (source,destination)-pairs
- snapshots of traffic loads on all the links

Assume we can simulate how changes affect the network on the observed data.

Solutions

- The overfitting solution:
  - Find a minimal network that satisfies all observed requests
  - For example, drop all links on which no traffic was observed
- A solution suggested by learning theory
  (the Occam's razor principle):
  - Make big changes in network only if improvements in quality are big
  - Size of a change measured by description length in bits

In simple cases overfitting can be avoided by using common sense, but lessons from learning theory might become invaluable with increasing complexity

Beyond IID

Most learning theory relies on statistical modelling assumptions, e.g., the IID assumption.

- How to take dynamics into account?

Possible solutions provided by *online learning*:

- No statistical assumptions
- Learning takes place while the system is in use
- Reliability guarantees in terms of loss against
  - best solution in hindsight
  - best sequence of solutions in hindsight

  Interpretation and relevance of relative guarantees?

Further challenges

- In learning, quality typically measured as an average over data (examples to be classified, services to be served, . . .)
  - How to handle more involved quality measures?
  - Avoiding dictatorship of the majority
- Decentralized learning
- Multiple learners
- Non-simulatable systems, . . .

Problems caused by adaptivity

- Performance guarantees for learning algorithms
  - Depend on hard to verify assumptions
  - May be loose or non-existent
- If learning fails, adaptability may make the system even less reliable!
- Thus, adaptivity by learning adds yet another complicated component to the already complex systems

Summary

- Learning theory gives concepts and tools for proving performance guarantees to systems that adapt to observed data
- The current tools may not be perfect as they are, especially in non-stationary cases
- Still, existing learning theory might provide a good starting point in building a theory of reliability for adaptive networked systems