Information-gain Computation

Anthony Di Franco
(un)Natural Computation Spring 2017

What?

Work with Turing-complete model class, not TMs directly.

Prolog-like language: recursive compositions of joint spaces of (discrete) variation

Measure information about a query and adapt evaluation strategy accordingly

Compress search space

Seek to achieve information-theoretic bounds on efficiency of query answering

What?

Perspectives:

- Kowalski, Algorithm = Logic + Control
  - So derive algorithm from logic (specification) by determining control (eval strategy)
- Curry-Howard isomorphism
- Pyke’s interleaving of programs with plans / proofs

What?

Precedents:

- Recurrent Neural Networks
  - Parameter space sampling for fitting
  - Complexity-based regularization

Why?

Correctness in software is elusive despite large incentives

Examples:

- Heartbleed: OpenSSL bug, est. >$500 mil. Damage
- Cryptocurrency: theDao hack, $60 mil.

Why?

Also, general efficiency of software engineering.

Working with specifications is order of magnitude more efficient than working with algorithms (Kowalski).

Describing constraint graph vs. describing many / all paths through graph.

Illustrate.

X > 3 && X < 5

How?

don't enumerate models, start from data and use data bias to consider only models that fit it well

c.f. universal compressors

How?

Universal compressor:

Incrementally / adaptively builds up dictionary of subsequences or uses already-decoded sequence as implicit dictionary to compress source sequence and achieve coding at entropy rate.

Variants (PPM) that work with (predictions of) probabilities of subsequences.

Model is implicit.

How?

So:

- Search / choose evaluation strategies adaptively to gain information quickly
- Compress joint spaces resulting from propagation of information
- Compress sequences of inferences leading to information gain at query
- Use these most frequently informative, compressed sequences with first priority thereafter

How?

Information measure is Total Correlation

Intuition: maximal uncertainty is all parts of joint space overlap completely with universe / each other

TC measures reduction in this.

Illustrate.

How?

Adapting evaluation strategy

Predicates can have disjunctions, we should try the most informative one first

Bandit problem. (Illustrate UCB.)

Future: CE method for high-D.

How?

Compressing search space

generalize Schmidhuber's history compression (RNN) to >1D

hinges on recursively finding and conditioning on sufficient statistics in hierarchy of scales (Illustrate).

State of predictor as sufficient statistic for past to build recursive hierarchy of predictors at larger (time) scales.

How?

Compressing search space

Apparently nothing special about time.

Generalize => info-clustering on joint spaces of adjacent predicates on derivation paths, recursively at hierarchy of scales.

Then do distribution estimation within those variable clusters.

How?

Compressing search space

Joint space compression expands alphabet in which paths can be compressed / creates tree of perhaps exponentially shorter paths from facts to query.

Alphabet expansion + sequence encoding as in large-alphabet compressed self-indices. Codes at zero-order entropy.

Turing-class models of given data then fall out by writing CNN-style predicates.

What now?

Adaptive evaluation + search-space and joint-variation compression
= optimal proven-correct computing

(I hope.)

On the agenda for next week

Small relational Prolog-like language embedded in Python

Adaptive evaluation strategy with bandit algorithm done.

Search space / joint space compression not done.

Perhaps smart-contracts-based demo.

(J-M Eber, J Seward, Simon Peyton Jones, “Composing contracts: an adventure in financial engineering,” September 1, 2000)