Deep Learning in Maple

Stephen Forrest
26 July 2018
ICMS 2018, Notre Dame

How might Maple use deep learning?

- Heuristics which decide between algorithms which are equivalent in practice but may vary considerably in performance
- Heuristics which require exactness but nevertheless permit arbitrary choices in the form of the output (e.g. symbolic simplification)
- User-generated associations between commands enabling user-interface optimizations

Maplesoft

Example: numeric integration

```
> evalf(Int(sin(x),x=0..9*Pi,method=_d01ajc));
                              2.                                    (1)
> evalf(Int(sin(x),x=0..9*Pi,method=_CCquad));
                          2.000000000                               (2)
> evalf(Int(sin(x),x=0..9*Pi,method=_Gquad));
                          2.000000000                               (3)
> evalf(Int(sin(x),x=0..9*Pi,method=_Dexp));
                          2.000000000                               (4)
> evalf(Int(sin(x),x=0..9*Pi,method=_Sinc));
                          2.000000000                               (5)
> evalf(Int(sin(x),x=0..9*Pi,method=_NCrule));
                          2.000000000                               (6)
```

From the online Maple help page for evalf/Int [1]

Other examples

- **LinearAlgebra:-Determinant**
  - Choose one of algnum, float, fracfree, integer, ipseudo, minor, modular[p], multivar, rational, unifloat, unifloat[x], univar, or univar[x] methods
  - Minor expansion is O(n^4) while LU decomposition is O(n^3)
- **LinearAlgebra:-LinearSolve**
  - Choose one of 'solve', 'subs', 'Cholesky', 'LU', 'QR', 'hybrid', 'modular', 'SparseLU', 'SparseDirect', or 'SparseIterative' methods
- **Optimization:-Minimize**
  - Dispatches automatically to one of LPSolve, QPSolve, or NLPSolve

Third-party tools in Maple

- Originally, Maple routines were either implemented directly in the Maple kernel or as library code.

- We now have many third party-libraries, including:
  - **Mathematical**: LAPACK/BLAS, GMP, nauty, qhull
  - **SAT/SMT**: MiniSAT/MapleSAT, Z3
  - **Scientific**: CoolProp (thermophysical data)
  - **General-purpose**: Curl, Java, PostgreSQL, Python, SQLite, many parsers

SC^2 and SAT/SMT in Maple

- The **SC^2 project** [2] aims to further develop links between CAS and SAT/SMT communities and their associated tools.

SC^2 Horizon 2020 European Union Funding for Research & Innovation

- Recent inclusion in Maple of links to SAT/SMT solvers (MapleSAT, Z3) in connection with SC^2 offers a glimpse of what linking to a deep learning tool could look like.

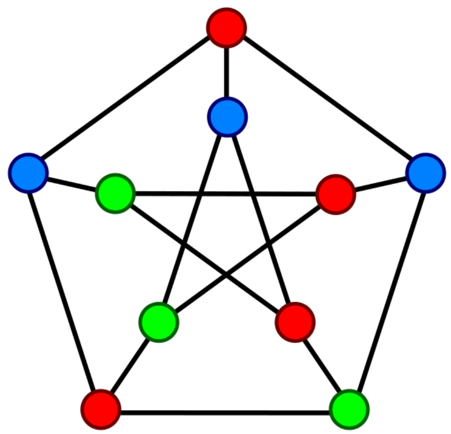
- In both SAT/SMT and ML, CAS must dispatch queries to a tool with a quite different methodology and accept some inherent uncertainty:
  - SAT/SMT solvers cannot guarantee a meaningful result will be returned within given resources bounds (as SAT is NP-complete)
  - A trained neural network classifier cannot promise 100% correctness on classification tasks against new data

TensorFlow Integration

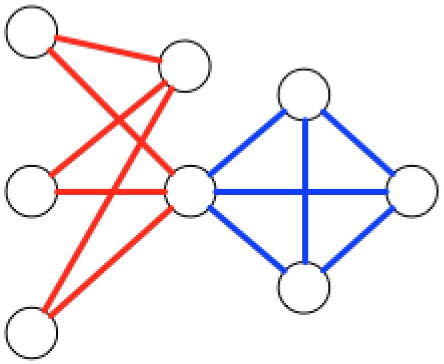
- **DeepLearning** package introduced in Maple 2018
- Provides interface to **Google TensorFlow** [3]
- TensorFlow v1.5 included in Maple 2018 distribution for MacOS, Linux, and 64-bit Windows
- Primarily intended to provide access to deep learning methods for Maple users
- Support for a subset of the TensorFlow Python API [4] in Maple 2018; further expansion targeted for next release, including support for recurrent and convolutional neural networks.

Graph properties (1)

- Let **G** be an undirected graph.
- The *chromatic number* of a graph is the minimal number of colors needed to color the vertices of **G** such that no adjacent vertices have the same color.

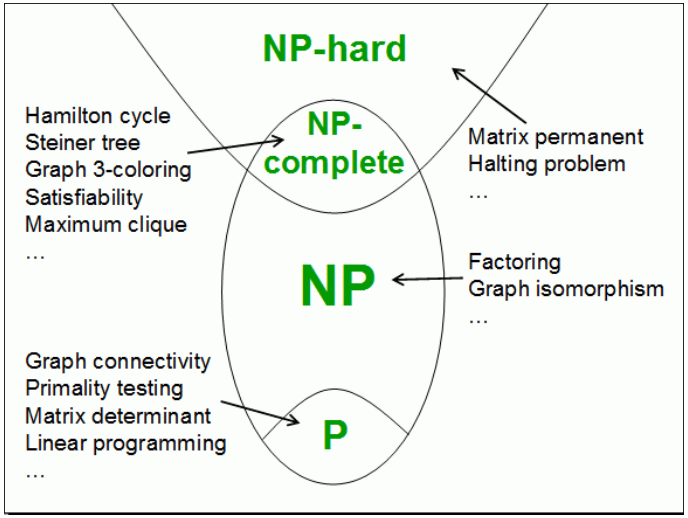


- A *maximum clique* of **G** is a largest subset **S** of the vertices of **G** in which any pair of vertices from **S** is connected in **G**. The *clique number* of **G** is the cardinality of **S**.



Graph properties (2)

- The task of finding a minimum coloring or a maximum clique are (famously) NP-complete.



- For any undirected graph **G** on **n** vertices we also have the following:
  - $CliqueNumber(G) \leq ChromaticNumber(G) \leq n$
  - $CliqueNumber(G^C) * ChromaticNumber(G) \geq n$

Case study: chromatic number (1)

- The **ChromaticNumber** command in the **GraphTheory** package has two exact algorithms for computing the chromatic number of a graph **G**, called **optimal** and **sat.**

- Method **optimal** first finds a maximal clique in **G**.
  - It then tries to generalize the maximal clique to a coloring of all of **G**.
  - If this fails, it iterates exhaustively over possible colorings using the clique number as a lower bound and growing upwards.

- Method **sat** iterates upwards from **k = 2.**
  - For each value of **k** it solves the k-colorability decision problem by generating a Boolean satisfiability instance and dispatching it to a SAT solver (MapleSAT), and returns the first successful coloring.

\

Case study: chromatic number (2)

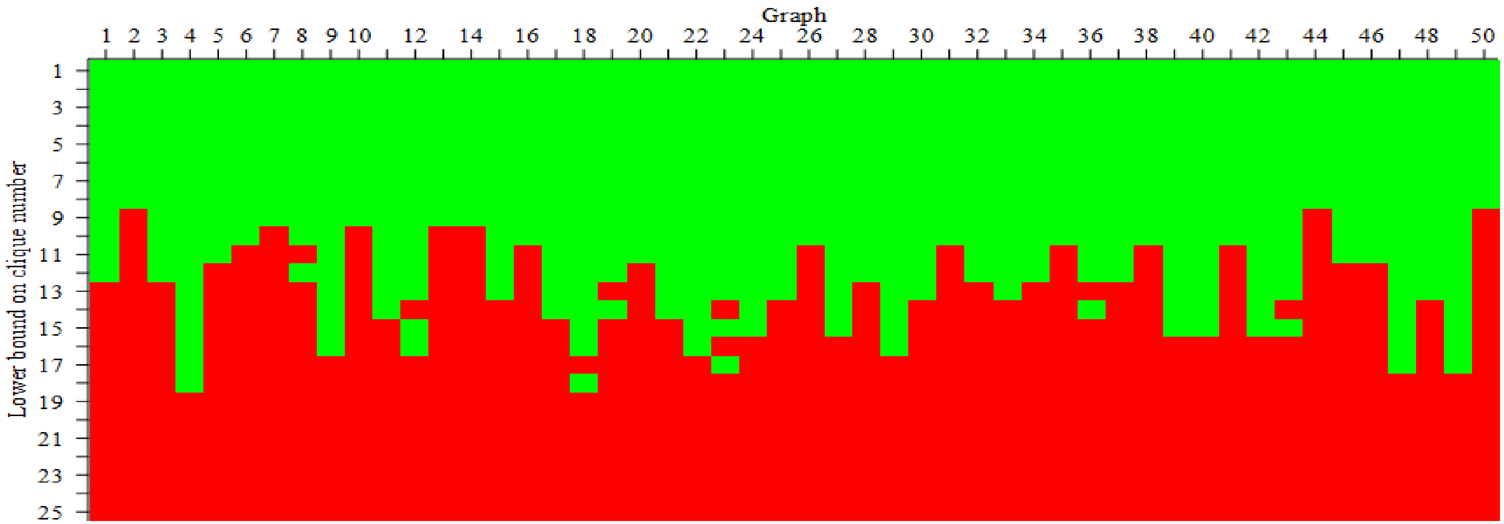
- In practice it seems **optimal** is fastest when there is a relatively large clique in **G** (e.g. size n/4 or more), and otherwise **sat** is much faster.
- But computing the clique number is costly!
- Instead, to give users the benefit of speed we could:
- Approximate the clique number
- Use some other heuristic to decide between **sat** & **optimal**
- Run both on separate threads and return the first result

Case study: chromatic number (3)

Idea: generate **M** random connected undirected graphs G[1],…,G[M] with **N** vertices and with edge density in [0.3,0.7].

For each i in 1,…,M and each j in 1,…,floor(N/2), compute the chromatic number of the graph union of G[i] with the complete graph on j vertices. Record which of **optimal** or **sat** produced the fastest coloring.

Results for **M**=**N**=50:



Case study: chromatic number (4)

Q: Can we train a DNN classifier to guess with high probability which of the two methods would be fastest?

A: Probably yes, but it is highly dependent on the way the graph is encoded.

Future Work

- Make special-purpose neural network applications (RNN, LSTM, CNN) available in DeepLearning
- Investigate replacing current human-coded heuristics (often very old and written by many authors) with trained machine-generated heuristics
- Investigate how neural networks might be used to guide user interfaces
- Build high-level classification tool which makes use of computer algebra's expressivity with symbolics

Questions

- How can neural network techniques be usefully harnessed in computer algebra despite the latter's need for exactness and consistency?
- Where do we obtain the data necessary to train machine models for computer algebra?
- Practical question: must a trained model always be harnessed to training infrastructure (e.g. TensorFlow, Theano, etc.) when deployed?

References

1. evalf/Int command. Maple 2018 online help.
2. SC² project. http://www.sc-square.org/
3. Google TensorFlow. https://www.tensorflow.org/
4. TensorFlow Python API. https://www.tensorflow.org/api_docs/python/