From Deep Learning to Deep Reasoning
Part A: Learning to reason

Truyen Tran, Vuong Le, Hung Le and Thao Le
{truyen.tran,vuong.le,thai.le,thao.le}@deakin.edu.au

Logistics

Truyen Tran  Vuong Le  Hung Le  Thao Le

Agenda

• Introduction
• Part A: Learning-to-reason framework
• Part B: Reasoning over unstructured and structured data
• Part C: Memory | Data efficiency | Recursive reasoning
DL: 8 years snapshot

2012

I WAS WINNING IMAGENET

UNTIL A DEEPER MODEL CAME ALONG

AlphaGo

Turing Award 2018

AusDM 2016

GPT-3 2020

Extrapolating the spectacular performance of GPT3 into the future suggests that the answer to life, the universe and everything is just 4.398 trillion parameters.

6:26 AM · Jun 11, 2020 · Twitter Web App

GPT-3 2020

AusDM 2016

Deep Learning & Applications in Sensitive Domains

Truyen Tran
Deakin University
truyen.tran@deakin.edu.au

Excellent deep learning tutorial in AusDM2016 by Truyen Tran!
DL has been fantastic, but ...

• It is great at interpolating
  • → data hungry to cover all variations and smooth local manifolds
  • → little systematic generalization (novel combinations)

• Lack of human-perceived reasoning capability
  • Lack natural mechanism to incorporate prior knowledge, e.g., common sense
  • No built-in causal mechanisms
  • → Have trust issues!

• To be fair, may of these problems are common in statistical learning!
## Why still DL in 2021?

### Theoretical
- **Expressiveness**: Neural nets can approximate any function.
- **Learnability**: Neural nets are trained easily.
- **Generalisability**: Neural nets generalize surprisingly well to unseen data.

### Practical
- **Generality**: Applicable to many domains.
- **Competitive**: DL is hard to beat as long as there are data to train.
- **Scalability**: DL is better with more data, and it is very scalable.
The next AI/ML challenge

2020s-2030s
- Learning + reasoning, general purpose, human-like
- Has contextual and common-sense reasoning
- Requires less data
- Adapt to change
- Explainable
Toward deeper reasoning

System 1: Intuitive
- Fast
- Implicit/automatic
- Pattern recognition
- Multiple

System 2: Analytical
- Slow
- Deliberate/rational
- Careful analysis
- Single, sequential

Theory of mind
Recursive reasoning

Perception

Memory
- Facts
- Semantics
- Events and relations
- Working space

Facts
Semantics
Events and relations
Working space
System 2

• Holds hypothetical thought
• Decoupling from representation
• Working memory size is not essential. Its attentional control is.
Reasoning in Probabilistic Graphical Models (PGM)

• Assuming models are fully specified (e.g., by hand or learnt)
  • \( \rightarrow \) Estimate MAP as energy minimization
  • \( \rightarrow \) Compute marginal probability
  • \( \rightarrow \) Compute expectation & normalisation constant

• Key algorithm: Pearl's Belief Propagation, a.k.a Sum-Product algorithm in factor graphs.
  • Known result in 2001-2003: BP minimises Bethe free-energy minimization.

\[ m_{i\to j}(x_j) = \sum_{x_i} \phi(x_i) \phi(x_i, x_j) \prod_{t \in N(i) \setminus j} m_{t\to i}(x_i) \]

message from \( i \) to \( j \)

Figure credit: Jonathan Hui

Can we learn to infer directly from data without full specification of models?
Agenda

• Introduction

• **Part A: Learning-to-reason framework**
  • Part B: Reasoning over unstructured and structured data
  • Part C: Memory | Data efficiency | Recursive reasoning
Part A: Sub-topics

• Reasoning as a prediction skill that can be learnt from data.
  • Question answering as zero-shot learning.

• Neural network operations for learning to reason:
  • Concept-object binding.
  • Attention & transformers.
  • Dynamic neural networks, conditional computation & differentiable programming.

• Reasoning as iterative representation refinement & query-driven program synthesis and execution
  • Compositional attention networks.
  • Neural module networks.

• Combinatorics reasoning
Learning to reason

• Learning is to self-improve by experiencing ~ acquiring knowledge & skills

• Reasoning is to deduce knowledge from previously acquired knowledge in response to a query (or a cues)

• Learning to reason is to improve the ability to decide if a knowledge base entails a predicate.
  • E.g., given a video f, determines if the person with the hat turns before singing.

• Hypotheses:
  • Reasoning as just-in-time program synthesis.
  • It employs conditional computation.


(Dan Roth; ACM Fellow; IJCAI John McCarthy Award)
Learning to reason, a definition

Definition 2.1.1. An algorithm $A$ is an exact reasoning algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if for all $f \in \mathcal{F}$ and for all $\alpha \in \mathcal{Q}$, when $A$ is presented with input $(f, \alpha)$, $A$ runs in time polynomial in $n$ and the size of $f$ and $\alpha$, and answers “yes” if and only if $f \models \alpha$.

E.g., given a video $f$, determines if the person with the hat turns before singing.

Practical setting: (query, database, answer) triplets

• This is very general:
  • Classification: Query = *what is this?* Database = *data*.
  • Regression: Query = *how much?* Database = *data*.
  • QA: Query = *NLP question.* Database = *context/image/text*.
  • Multi-task learning: Query = *task ID.* Database = *data*.
  • Zero-shot learning: Query = *task description.* Database = *data*.
  • Drug-protein binding: Query = *drug.* Database = *protein*.
  • Recommender system: Query = *User (or item).* Database = *inventories (or user base)*;
Can neural networks reason?

Reasoning is not necessarily achieved by making logical inferences.

There is a continuity between [algebraically rich inference] and [connecting together trainable learning systems].

Central to reasoning is composition rules to guide the combinations of modules to address new tasks.

“When we observe a visual scene, when we hear a complex sentence, we are able to explain in formal terms the relation of the objects in the scene, or the precise meaning of the sentence components. However, there is no evidence that such a formal analysis necessarily takes place: we see a scene, we hear a sentence, and we just know what they mean. This suggests the existence of a middle layer, already a form of reasoning, but not yet formal or logical.”

Hypotheses

• Reasoning as just-in-time program synthesis.
• It employs conditional computation.
• Reasoning is recursive, e.g., mental travel.
Two approaches to neural reasoning

• **Implicit chaining of predicates through recurrence:**
  • Step-wise query-specific attention to relevant concepts & relations.
  • Iterative concept refinement & combination, e.g., through a working memory.
  • Answer is computed from the last memory state & question embedding.

• **Explicit program synthesis:**
  • There is a set of modules, each performs an pre-defined operation.
  • Question is parse into a symbolic program.
  • The program is implemented as a computational graph constructed by chaining separate modules.
  • The program is executed to compute an answer.
In search for basic neural operators for reasoning

• Basics:
  • Neuron as feature detector → Sensor, filter
  • Computational graph → Circuit
  • Skip-connection → Short circuit

• Essentials
  • Multiplicative gates → AND gate, Transistor, Resistor
  • Attention mechanism → SWITCH gate
  • Memory + forgetting → Capacitor + leakage
  • Compositionality → Modular design

..
Part A: Sub-topics

• Reasoning as a prediction skill that can be learnt from data.
• Question answering as zero-shot learning.

• **Neural network operations for learning to reason:**
  • Concept-object binding.
  • Attention & transformers.
  • Dynamic neural networks, conditional computation & differentiable programming.

• Reasoning as iterative representation refinement & query-driven program synthesis and execution.
  • Compositional attention networks.
  • Reasoning as Neural module networks.

• Combinatorics reasoning
Concept-object binding

- Perceived data (e.g., visual objects) may not share the same semantic space with high-level concepts.
- Binding between concept-object enables reasoning at the concept level.

Attentions: Picking up only what is needed at a step

• Need attention model to select or ignore certain **computations** or **inputs**
  • Can be “soft” (differentiable) or “hard” (requires RL)
  • Needed for selecting predicates in reasoning.

• Attention provides a short-cut → long-term dependencies
  • Needed for long chain of reasoning.
  • Also encourages sparsity if done right!

http://distill.pub/2016/augmented-rnns/
Fast weights | HyperNet – the multiplicative interaction

• Early ideas in early 1990s by Juergen Schmidhuber and collaborators.

• **Data-dependent weights** | Using a controller to generate weights of the main net.

---

**Figure:** The HyperRNN system. The black system represents the main RNN while the orange system represents the weight-generating HyperRNN cell.

Memory networks: Holding the data ready for inference

- Input is a set → Load into memory, which is NOT updated.
- State is a RNN with attention reading from inputs.
- Concepts: Query, key and content + Content addressing.
- Deep models, but constant path length from input to output.
- Equivalent to a RNN with shared input set.

Transformers: Analogical reasoning through self-attention

\[ A_h = \text{Softmax}(\alpha Q_h K_h^T) V_h \]

\[ Q_h = W_q X, \quad K_h = W_k X \quad \text{and} \quad V_h = W_v X \]

Transformer as implicit reasoning

• Recall: Reasoning as (free-) energy minimisation
  • The classic Belief Propagation algorithm is minimization algorithm of the Bethe free-energy!

• Transformer has relational, iterative state refinement makes it a great candidate for implicit relational reasoning.

Transformer v.s. memory networks

• Memory network:
  • Attention to input set
  • One hidden state update at a time.
  • Final state integrate information of the set, conditioned on the query.

• Transformer:
  • Loading all inputs into working memory
  • Assigns one hidden state per input element.
  • All hidden states (including those from the query) to compute the answer.
Universal transformers


Dynamic neural networks

• Memory-Augmented Neural Networks
• Modular program layout
• Program synthesis
Neural Turing machine (NTM)
A memory-augmented neural network (MANN)

- A controller that takes input/output and talks to an external memory module.
- Memory has read/write operations.
- The main issue is where to write, and how to update the memory state.
- All operations are differentiable.

Source: rylanschaeffer.github.io
MANN for reasoning

• Three steps:
  • Store data into memory
  • Read query, process sequentially, consult memory
  • Output answer

• Behind the scene:
  • Memory contains data & results of intermediate steps
  • LOGNet does the same, memory consists of object representations

• Drawbacks of current MANNs:
  • No memory of controllers → Less modularity and compositionality when query is complex
  • No memory of relations → Much harder to chain predicates.
Part A: Sub-topics

• Reasoning as a prediction skill that can be learnt from data.
  • Question answering as zero-shot learning.

• Neural network operations for learning to reason:
  • Concept-object binding.
  • Attention & transformers.
  • Dynamic neural networks, conditional computation & differentiable programming.

• **Reasoning as iterative representation refinement & query-driven program synthesis and execution.**
  • Compositional attention networks.
  • Reasoning as Neural module networks.

• Combinatorics reasoning
MAC Net: Recurrent, iterative representation refinement

Module networks
(reasoning by constructing and executing neural programs)

- Reasoning as laying out modules to reach an answer
- Composable neural architecture → question parsed as program (layout of modules)
- A module is a function \((x \rightarrow y)\), could be a sub-reasoning process \(((x, q) \rightarrow y)\).
Putting things together:
A framework for visual reasoning

@Truyen Tran & Vuong Le, Deakin Uni
Part A: Sub-topics

• Reasoning as a prediction skill that can be learnt from data.
  • Question answering as zero-shot learning.
• Neural network operations for learning to reason:
  • Concept-object binding.
  • Attention & transformers.
  • Dynamic neural networks, conditional computation & differentiable programming.
• Reasoning as iterative representation refinement & query-driven program synthesis and execution.
  • Compositional attention networks.
  • Reasoning as Neural module networks.

• **Combinatorics reasoning**
Implement combinatorial algorithms with neural networks

Train neural processor P to imitate algorithm A

Processor P:
(a) aligned with the computations of the target algorithm;
(b) operates by matrix multiplications, hence natively admits useful gradients;
(c) operates over high-dimensional latent spaces

Processor as RNN

• Do not assume knowing the structure of the input, input as a sequence
  → not really reasonable, harder to generalize

• RNN is Turing-complete
  → can simulate any algorithm

• But, it is not easy to learn the simulation from data (input-output)
  → Pointer network

Assume O(N) memory
And O(N^2) computation
N is the size of input

\[
\begin{align*}
  u^i_j &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
  p(C_i|C_1, \ldots, C_{i-1}, P) &= \text{softmax}(u^i)
\end{align*}
\]

Processor as MANN

- MANN simulates neural computers or Turing machine→ ideal for implement algorithms
- Sequential input, no assumption on input structure
- Assume O(1) memory and O(N) computation

Sequential encoding of graphs


- Each node is associated with random one-hot or binary features
- Output is the features of the solution

Geometry

\[
\begin{align*}
&[x_1, y_1, \text{feature1}], \\
&[x_2, y_2, \text{feature2}], \\
&\ldots \\
&[\text{feature4}], \\
&[\text{feature2}], \\
&\ldots
\end{align*}
\]

Graph

\[
\begin{align*}
&[\text{node_feature1}, \text{node_feature2}, \text{edge12}], \\
&[\text{node_feature1}, \text{node_feature2}, \text{edge13}], \\
&\ldots \\
&[\text{node_feature4}], \\
&[\text{node_feature2}], \\
&\ldots
\end{align*}
\]
DNC: graph reasoning
NUTM: learning multiple algorithms at once

Processor as graph neural network (GNN)

Motivation:
• Many algorithm operates on graphs
• Supervise graph neural networks with algorithm operation/step/final output
• Encoder-Process-Decode framework:

\[
\tilde{x}_i^{(t)} = f_A(x_i^{(t)}, \tilde{h}_i^{(t-1)})
\]

\[
H^{(t)} = P(Z^{(t)}, E^{(t)})
\]

\[
\tilde{y}_i^{(t)} = g_A(x_i^{(t)}, \tilde{h}_i^{(t)})
\]

\[
\tau^{(t)} = \sigma(T_A(H^{(t)}, \overline{H}^{(t)}))
\]

Attention passing

https://petar-v.com/talks/Algo-WWW.pdf
Veličković, Petar, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell.
Example: GNN for a specific problem (DNF counting)

- Count #assignments that satisfy disjunctive normal form (DNF) formula
- Classical algorithm is P-hard $O(mn)$
- $m$: #clauses, $n$: #variables
- Supervised training on output-level

Neural networks and algorithms alignment

Summary statistics
What is the maximum value difference among treasures?

Relational argmax
What are the colors of the furthest pair of objects?

Dynamic programming
What is the cost to defeat monster X by following the optimal path?

NP-hard problem
Subset sum: Is there a subset that sums to 0?

MLPs
- feature extraction

Deep Sets (Zaheer et al.,
- summary statistics

GNNs
- (pairwise) relations

GNNs
- (pairwise) relations

Neural exhaustive search

\[
y = \text{MLP}(\|_{s \in S} X_s) \\
y = \text{MLP}_2 \left( \sum_{s \in S} \text{MLP}_1(X_s) \right)
\]

\[
h_s^{(k)} = \sum_{t \in S} \text{MLP}_1^{(k)} \left( h_{s}^{(k-1)}, h_t^{(k-1)} \right)
\]

\[
y = \text{MLP}_2 \left( \sum_{s \in S} h_s^{(K)} \right)
\]


https://petar-v.com/talks/Algo-WWW.pdf
GNN is aligned with Dynamic Programming (DP)

Graph Neural Network

\[
\text{for } k = 1 \ldots \text{GNN iter:} \\
\text{for } u \in S: \quad \text{No need to learn for-loops} \\
h_u^{(k)} = \Sigma_v \text{MLP}(h_v^{(k-1)}, h_u^{(k-1)})
\]

Bellman-Ford algorithm

\[
\text{for } k = 1 \ldots |S| - 1: \\
\text{for } u \in S: \\
d[k][u] = \min_v d[k-1][v] + \text{cost}(v, u)
\]

Learns a simple reasoning step

\[
h_u^{(k)} \\
\Sigma_v \\
\text{MLP}(h_v^{(k-1)}, h_u^{(k-1)}) \\
d[k][u] \\
\min_v \\
d[k-1][v] + \text{cost}(v, u)
\]

\[
\text{MLP}_2(\max_{\tau \subseteq S} \text{MLP}_1 \circ \text{LSTM}(X_1, \ldots, X_{|\tau|} : X_1, \ldots, X_{|\tau|} \in \tau)).
\]
If alignment exists $\rightarrow$ step-by-step supervision

- Merely simulate the classical graph algorithm, generalizable
- No algorithm discovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Inputs</th>
<th>Supervision signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth-first search</td>
<td>$x_i^{(t)}$: is $i$ reachable from $s$ in $\leq t$ hops?</td>
<td>$x_i^{(t+1)}$, $\tau^{(t)}$: has the algorithm terminated?</td>
</tr>
<tr>
<td>Bellman-Ford</td>
<td>$x_i^{(t)}$: shortest distance from $s$ to $i$ (using $\leq t$ hops)</td>
<td>$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of $i$ in the shortest path tree (in $\leq t$ hops)</td>
</tr>
<tr>
<td>Prim’s algorithm</td>
<td>$x_i^{(t)}$: is node $i$ in the (partial) MST (built from $s$ after $t$ steps)?</td>
<td>$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of $i$ in the partial MST</td>
</tr>
</tbody>
</table>

Joint training is encouraged
Processor as Transformer

- Back to input sequence (set), but stronger generalization
- Transformer with encoder mask ~ graph attention
- Use Transformer with:
  - Binary representation of numbers
  - Dynamic conditional masking

Training with execution trace
End of part A