Neural guidance in E

Using neural networks

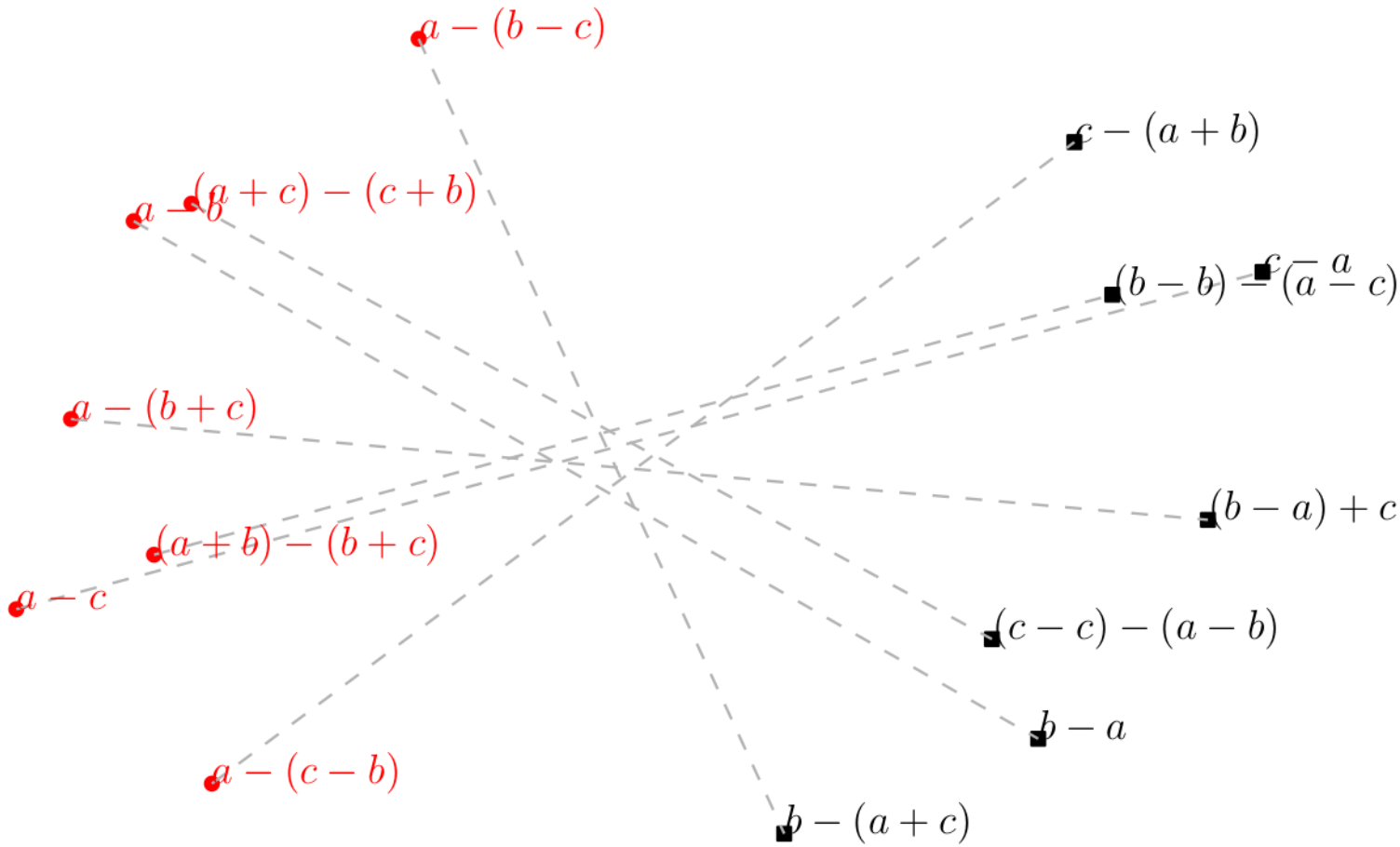
- we can use neural networks (NNs) directly on our hand-crafted feature vectors,
- however, we can also extract features using NNs,
- ideally: we want to represent objects semantically not syntactically,
- our representation: terms, literals, and clauses are represented by vectors

$$a - (b + c) \text{ is represented by } v \in \mathbf{R}^n,$$

- note that our model was designed to be reasonably close to the previous approach (not as smart as possible) and all these things are only initial steps...

Vector representations of terms (very simplified example)

Objective here: equal terms should be as close as possible

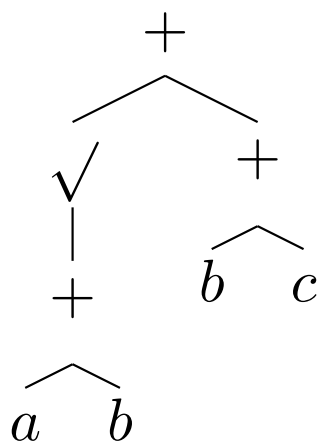


In our case the problem is much more complex, e.g., we can have simultaneously two predicates representing syntactic and semantic equivalence, respectively.

image source: Allamanis et al. 2017

How to obtain these representations?

- we can exploit compositionality and the tree structure of our objects



term	representation
a	$\mathbf{R}^n$
b	$\mathbf{R}^n$
c	$\mathbf{R}^n$
$\sqrt{}$	$\mathbf{R}^n \to \mathbf{R}^n$
$+$	$\mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$

(for simplicity we assume that everything lives in $\mathbf{R}^n$)

Notes on compositionality

- in many cases it is clear how to produce a more complex object from simpler objects, but

$$f(x,y) = \begin{cases} 1 & \text{if } x \text{ halts on } y, \\ 0 & \text{otherwise.} \end{cases}$$

- even constants can be complex, e.g., $\{\, x : \forall y (f(x,y) = 1) \,\}$,
- very special objects are variables and Skolem functions (constants),
- note that different types of objects can live in different spaces as long as we can connect things together

Recursive NNs

- popularized by Socher, Lin, et al. 2011 in NLP,
- unlike feed-forward NNs do not have a static structure, but the structure is different for different inputs

Our representation

- a constant is represented by a learned vector (embedding),
- a variable is represented by a learned vector (embedding),
  - for simplicity (and as in the previous part) all variables are represented by one vector and we treat Skolem symbols similarly
- a function symbol f is represented by a learned function (NN) $v_f \colon \underbrace{\mathbf{R}^n \times \cdots \times \mathbf{R}^n}_{k\text{-times}} \to \mathbf{R}^n$, where k is the arity of f,
- a predicate symbol P is represented by a learned function (NN) $v_P \colon \underbrace{\mathbf{R}^n \times \cdots \times \mathbf{R}^n}_{k\text{-times}} \to \mathbf{R}^n$, where k is the arity of P,
- note that we treat equality as a learned binary predicate

Clauses and conjectures

- we represent negation as a learned unary operation,
- we can represent disjunction similarly, but a clause is more like a sequence (set) of literals

Recurrent NNs (RNNs)

- consume sequences of vectors,
- a representation of a clause is obtained by a RNN (Cl) from the representations of literals in the clause,
- a representation of a conjecture is obtained by a RNN ($Conj$) from the representations of clauses in the conjecture,

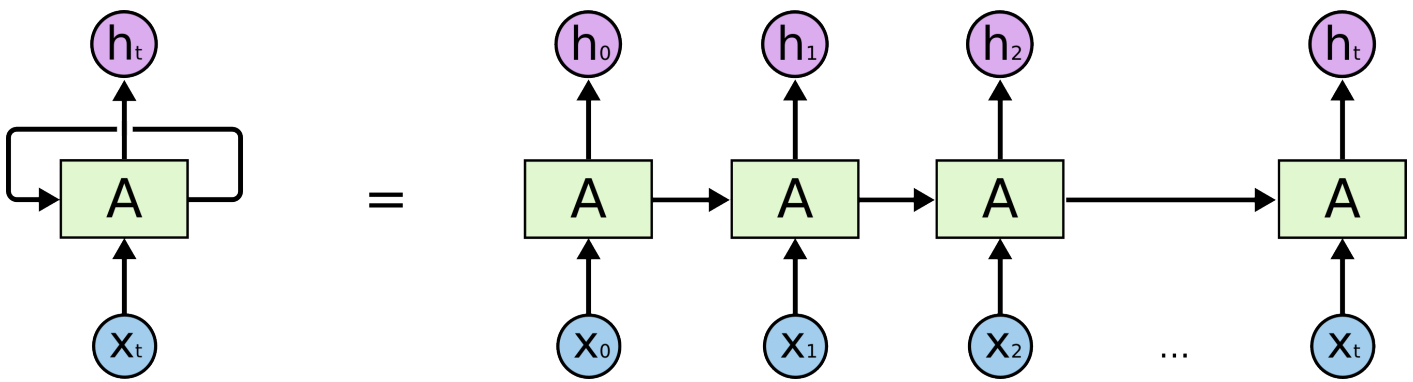


image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM and GRU

- in principle RNNs can learn long dependencies,
- Long short-term memory (LSTM) was developed to help with vanishing and exploding gradients in vanilla RNNs,
- Gated recurrent unit (GRU) is a "simplified" LSTM,
- many variants — bidirectional, stacked, . . .

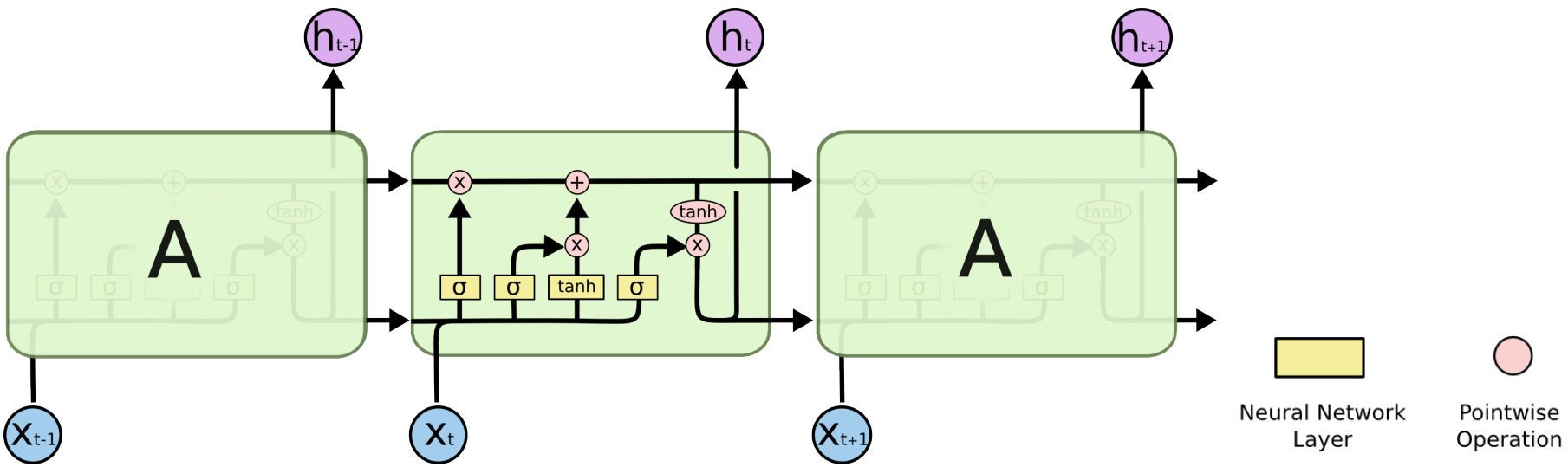


image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Final layer

- input: the represenations of a conjecture, which we want to prove, and a clause, which we want to evaluate
- output: two real numbers
  - we can normalize them into a probability distribution,
  - or just say that the clause is a good/bad given clause based on them (that is what we do now)
- note that it does not have a direct access to the properties of clauses like length, age, #variables

Current neural model parameters

- $n = 64$,
- function and predicate symbols are represented by a linear layer and ReLU6 ($\min(\max(0, x), 6)$),
- Cl and $Conj$ are LSTMs (GRUs are faster),
- the output vector of $Conj$ has length $m = 16$,
- the final layer is a sequence of linear, ReLU, linear, ReLU, and linear layers ($\mathbf{R}^{n+m} \to \mathbf{R}^{\frac{n}{2}} \to \mathbf{R}^2$)
- rare symbols are grouped together — we can loosely speaking obtain a general constant, binary function, ...

Various possible modifications

- too many to even list them...
- our representation of variables and Skolem symbols is clearly an oversimplification and can be improved in various ways,
- note that different types of objects can be represented by vectors of different lengths and different function and predicate symbols can have very different representations (NNs),
- we can use an apply function instead (even recurrent one mainly to improve the representations of rare or out of vocabulary symbols),
- in NLP explicit typing helps, see Socher, Huval, et al. 2012,
- it is also possible to take into account already selected (or even generated) clauses

Optimizations

Training

- we use minibatches, where we group together examples that share the same conjecture and we cache all the representations obtained in one batch

ATP evaluation

- all the computed representations of objects are cached during a proof search and hence there is no need to recompute them again,
- the most interesting thing about all this is that this whole neural approach really works even though it has non-trivial overhead and caching helps a lot

Bibliography I


Allamanis, Miltiadis et al. (2017). "Learning Continuous Semantic Representations of Symbolic Expressions". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 80–88. URL: http://proceedings.mlr.press/v70/allamanis17a.html.


Socher, Richard, Brody Huval, et al. (2012). "Semantic Compositionality through Recursive Matrix-Vector Spaces". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pp. 1201–1211. URL: http://www.aclweb.org/anthology/D12-1110.

Bibliography II

Socher, Richard, Cliff Chiung-Yu Lin, et al. (2011). "Parsing Natural Scenes and Natural Language with Recursive Neural Networks". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, pp. 129–136. ISBN: 978-1-4503-0619-5.