Logic, Computing, Neural Networks

Alexander Sakharov

http://alex.sakharov.net

1 History and motivation

– Logic as formalization of intellectual activity
– Formalization of computing
– AI lessons
– Merging logic and computing
– Neural networks for fuzzy reasoning
– Neuaral-symbolic computing

Logic could serve as a framework encompassing reasoning, computing, and neural networks. This lecture does not discuss all the above issues and known solutions but rather focuses on the so-called Horn fragment of logic.

2 Logic

Firts-order classical logic

Language

Atoms are expressions $P(t_1, ..., t_k)$ where P is a predicate and $t_1, ..., t_k$ are terms.
Terms are built recursively from object variables, constants, and functions.
Formulas are built recursively from constants and atoms using standard logical connectives and quantifiers.
Binary connectives: $\vee \wedge \Rightarrow$ Unary connective: $\neg$
Quantifiers: $\forall \exists$ Constants: $\top$ and $\bot$
A literal is an atom or its negation. A formula/literal/atom/term is called ground if it does not contain variables.

Model theory

Every atom is mapped to one of the two values: 0 (false) or 1 (true). Truth tables (or boolean functions) for logical connectives: $\{0,1\}^2 \to \{0,1\}$ for binary connectives and $\{0,1\} \to \{0,1\}$ for negation
Valid formula: true in any model. Unsatisfiable formula: false in any model.

Proof theory

Hilbert-type systems: many axioms and one inference rule called Modus Ponens:

$$\frac{A \quad A \Rightarrow B}{B} \; MP$$

Gentzen-type systems: one axiom and many inference rules

It is expected that model theories and the respective proof theories match each other.

Theorem. In FOL, a formula A is derivable if and only if it is valid.

Theorem. The set of valid (derivable) formulas of FOL is undecidable, i.e there is no such total algorithm that yields one value for valid formulas and a different value for all other formulas.

3 Horn rules, logic programs, knowledge bases

Horn (definite) rules are atoms and expressions $A_1 \wedge ... \wedge A_k \Rightarrow A$ where $A, A_1, ..., A_k$ are atoms. In classical FOL, Horn rules are equivalent to disjunctions: $\neg A_1 \vee ... \vee \neg A_k \vee A$. These disjunctions are also called Horn clauses.

The core of the AI systems known as knowledge bases is domain knowledge specified by a finite set of rules. Any such rule expresses a piece of domain knowledge. These rules often have the form of Horn clauses. Rules of the form A (atoms) are also called facts. Logic programs (Prolog) are also sets of Horn clauses. The sets of knowledge base rules or logic programs can be treated as conjunctions.

In classical logic, sets of Horn rules are first-order logic formulas in the conjunctive normal form. Any formula in the conjunctive normal form can be Skolemized. After that, universal quantifiers are removed. If all disjunctions in the conjunctive normal form of a formula has exctly one positive atom, then this formula can be written as a logic program or a set of knowledge base rules.

4 Horn clauses examples

Predicate and function properties

Commutativity:
$T(x, y) \Rightarrow T(y, x)$

Transitivity of order relation T:
$T(x, y) \wedge T(y, z) \Rightarrow T(x, z)$

Monotonicity:
$x \leq y \Rightarrow f(x) \leq f(y)$

Odd/even numbers:
$Odd(n) \wedge Even(m) \Rightarrow Odd(n + m)$

Predicate definitions

Minimum of total order R:
$R(x, s(x)) \Rightarrow M(x)$ (Skolemized version of $\forall y R(x, y) \Rightarrow M(x)$)

Transitive closure:
$E(x, y) \Rightarrow C(x, z)$
$C(x, y) \wedge E(y, z) \Rightarrow C(x, z)$

5 Equality

Additional axioms:
$x = x$
$x = y \Rightarrow (x = z \Rightarrow y = z)$
$x = y \Rightarrow f(x_1, ..., x, ..., x_k) = f(x_1, ..., y, ..., x_k)$
$x = y \Rightarrow P(x_1, ..., x, ..., x_k) = P(x_1, ..., y, ..., x_k)$

If the equality axioms are added to FOL, then more complex calculi and inference methods are required.

6 Negative literals

The absence of negative literals in the notation of Horn rules can be partially compensated by introducing new predicate P^- for every predicate P. Atoms $P^-(x, ..., y)$ correspond to $\neg P(x, ..., y)$. Although, this is a lousy solution. If we derive $P^-(a, ..., b)$, nothing can be concluded about $P(a, ..., b)$. An inference procedure may continue trying to find inference for the latter. Also, rules specifying properies of P^- should be given in addition to rules specifying properies of P.

7 Inference for Horn formulas

FOL view of inference from Horn rules

Let $\{R_1, ..., R_r\}$ be the set of Horn rules (including facts). Usually, the outcome of inference in AI systems is atoms. From the perspective of FOL, we are concerned about deriving implications $\wedge_{i\in[1...r]}R_i \Rightarrow F$ whose conclusions F are atoms. The predicates of these atoms occur in the Horn rules. These atoms F are called goals.

Axiomatic view of inference from Horn rules

Another possible interpretation of inference of atoms from a set of Horn rules is that all R_i where $i = 1...r$ are treated as axioms. These axioms are added to FOL. In this case, atoms F themselves are the subject of inference. Any particular set of these axioms given by Horn rules corresponds to a calculus. We consider a family of such calculi. They will be called Horn logics (HL).

Equivalence of the two views

These two views of KB inference are equivalent.

1. HL $\Rightarrow$ FOL

Suppose FOL is extended with axioms $\{R_1, ..., R_r\}$. Formula $\wedge_{i\in[1...r]}R_i$ is derivable in FOL from the axioms. If atom F is derivable in FOL extended with axioms $\{R_1, ..., R_r\}$, then we can replace this set of axioms with one axiom $\wedge_{i\in[1...r]}R_i$, and then $\wedge_{i\in[1...r]}R_i \Rightarrow F$ is derivable in FOL due to the deduction theorem.

1. FOL $\Rightarrow$ HL

If $\wedge_{i\in[1...r]}R_i \Rightarrow F$ is derivable in FOL, then F is derivable in HL because it can be obtained by application of Modus Ponens to this formula and $\wedge_{i\in[1...r]}R_i$.

8 HL models

Definition. An HL model is such assignment of truth values (0, 1) to every atom with constant arguments that all ground instances of axioms (Horn rules) have truth value 1. (The truth values of Horn rules are calculated as in FOL models.)

Theorem. Atom F is valid wrt HL models if and only if $\wedge_{i\in[1...r]}R_i \Rightarrow F$ is valid in FOL.

The proof is straightforward. As a corollary, an atom is derivable in HL if and only if it is valid wrt HL models.

9 Chaining inference

A substitution is a finite set of mappings of variables to terms: $\theta = \{x_1 \to t_,..., x_n \to t_n\}$. The result of applying any substitution θ to a formula A is called its instance, it is denoted $A\theta$. $A\theta$ is obtained from A by simultaneously replacing occurrences of $x_1, ..., x_n$ with $t_1, ..., t_n$, respectively.

Inference in AI systems

Chaining inference methods are widely used in Horn knowledge bases. Prolog inference is implemented as backward chaining in multiple systems. There are two chaining mehods: forward chaining and backward chaining.

Chaining methods are formalized as inference using Generalized Nodus Ponens (GMP) as the sole inference rule. If substitution θ is a unifier of literals A'_i and A_i, i.e. $A'_i\theta = A_i\theta$, for $i = 1...k$,

$$\frac{A'_1 \quad ... \quad A'_k \quad A_1 \wedge ... \wedge A_k \Rightarrow A}{A\theta} \; GMP$$

It is assumed that all variables in a Horn rule are replaced with new variables before any application of GMP.

Theorem. GMP is a sound inference rule for FOL.

The proof is by a straightforward induction on the depth of derivations.

Backward and forward chaining apply GMP in opposite directions. A forward chaining step derives $A\theta$ given that $A'_1, ..., A'_k$ are axioms or derived atoms. Forward chaining stops when an atom unifiable with the goal is obtained.

Given goal list $L = \{...G...\}$ and such substitution θ that $G\theta = A\theta$, every step of backward chaining replaces goal G with $A_1\theta, ..., A_k\theta$ in L and also applies θ to the other goals in L. The initial goal is the only element of the goal list for the first inference step. Backward chaining proceeds until the goal list is empty.

Backward chaining is a goal-directed method whereas forward chaining is data-driven. Backward chaining derivations are linear. Forward chaining is similar to HL minus FOL, i.e. without FOL axioms.

10 Example of chaining inference

Horn rules:
$man(x) \Rightarrow person(x)$
$person(x) \Rightarrow mother(x, m(x))$ (m is a Skolem function)
$mother(x, y) \Rightarrow loves(y, x)$
$man(John)$
Goal:
$loves(y, John)$

Forward chaining:
$person(John)$ $\{x_1 \rightarrow John\}$
$mother(John, m(John))$ $\{x_2 \rightarrow John\}$
$loves(m(John), John)$ $\{x_3 \rightarrow John, y_3 \rightarrow m(John)\}$

Backward chaining:
$mother(John, y)$ $\{y_1 \rightarrow y, x_1 \rightarrow John\}$
$person(John)$ $\{x_2 \rightarrow John, y \rightarrow m(John)\}$
$man(John)$ $\{x_3 \rightarrow John\}$

11 Equivalence of forward and backward chaining

Theorem. An atom F is derivable by forward chaining from a set of Horn rules $\{R_1, ..., R_r\}$ if and only if it is derivable by backward chaining from the same set of Horn rules.

Look at a forward/backward chaining derivation. Let us ground this derivation. We trasform a forward chaining derivation into a backward chaining derivation and vice versa. The proof in both direction is by induction on the size of derivations.

1. Forward chaining $\rightarrow$ backward chaining

Suppose

$$\frac{A'_1 \quad ... \quad A'_k \quad A_1 \wedge ... \wedge A_k \Rightarrow A}{A\theta}\ GMP$$

is the last step of a forward chaining derivation. By the induction assumption, all $A'_1, ..., A'_k$ are derivable by backward chaining. If this application of GMP is the first step of backward chaining, then it creates goal list $\{A'_1, ..., A'_k\}$. Now we can combine this step with the backward chaining derivations of $A'_1, ..., A'_k$ in sequence. The only difference from the original derivations of these atoms is goal lists in the backward chaining deivation of A'_i are augmented with $A'_{i+1}, ..., A'_k$.

2. Backward chaining $\rightarrow$ forward chaining.

We can consider backward chaining derivations staring with goal lists with more than one item. Suppose one step of backward chaining transforms goal list $F_1, ..F_i, ..., F_j, ..., F_k$ into $F_1, ..F_i, ..., G_1, ..., G_m, ..., F_k$ and the next step transforms this list into $F_1, ..H_1, ..., H_n, ..., G_1$, $..., G_m, ..., F_k$. Clearly, these two steps can be permuted like any other such consecutive steps that the second step does not replace a goal generated in the first step. By induction on the depth of derivations, any backward chaining derivation can be transformed into such derivation that any GMP replaces the first item in the goal list.

If the GMP step shown earlier is the first step of a backward chaining deivation, then let us rearrange this derivation so that the first element in the goal list is always picked first. Consider the part of the derivation ending with the goal list $A'_2, ..., A'_k$. This derivation part constitutes a backward chaining derivation of A'_1. Likewise, the derivation part starting with goal list $A'_i, ..., A'_k$ and ending with goal list $A'_{i+1}, ..., A'_k$ is a backward chaining derivation of A'_i. By the induction assumption, all $A'_1, ..., A'_k$ are derivable by forward chaining. The respective forward chaining derivations are combined with the step in question, which becomes the last step.

12 Resolution

The resolution calculus has two rules: resolution (left) and factoring (right).

$$\frac{A_1 \vee ... \vee A \vee ... \vee A_k \quad B_1 \vee ... \vee B \vee ... \vee B_m}{A_1\theta \vee ... \vee A_k\theta \vee B_1\theta \vee ... \vee B_m\theta} \qquad \frac{A_1 \vee ... \vee A \vee ... \vee A' \vee ... \vee A_k}{A_1\theta \vee ... \vee A\theta \vee ... \vee A_k\theta}$$

$A_1\theta \vee ... \vee A_k\theta \vee B_1\theta \vee ... \vee B_m\theta$ is called the resolvent of $A_1 \vee ... \vee A \vee ... \vee A_k$ and $B_1 \vee ... \vee B \vee ... \vee B_m$. In the resolution rule, A is an atom, B is a neagative literal, substitution θ is the most general unifier (MGU) of A and the atom of B, that is, any other unifier of the two atoms is a composition of θ and another substitution. In the factoring rule, substitution θ is the MGU of A and A'.

Input: a set of disjunctions. Output: empty clause.

Theorem. The resolution calculus is a complete inference method for FOL.

In application to HL, Horn rules are input clauses for resolution along with the negated goal atom.

13 Resolution strategies

Input resolution

- One of two premises of the resolution rule is a clause from the input set -

Theorem. Input resolution is complete for Horn clauses but incomplete for FOL.

Backward chaining is similar to input resolution in which the goal is used in the first step

Unit resolution

- One of two premises of the resolution rule is comprised of just one literal -

Theorem. Unit resolution is complete for Horn clauses but incomplete for FOL.

Forward chaining is somewhat similar to unit resolution, every application of GMP combines several resolution steps

Ordered resolution

Definition. Order relation $\succ$ on formulas is called a simplification order if it is:
- well-founded: there is no infinite sequence of formulas $t_0 \succ t_1 \succ ...$
- monotone: if r is a subformula of l and $l \neq r$, then $l \succ r$
- stable: if $l \succ r$, then $l\theta \succ r\theta$ for any substitution θ

Definition. Formula A is maximal with respect to the set of formulas Γ if $B \succ A$ does not hold for any other formula $B \in \Gamma$.

Suppose a simplification order is defined for atoms.

- Every resolved literal is maximal wrt other literals in each premise; every factored literal is maximal among literals in the premise -

Theorem. Ordered resolution is complete for FOL.

For Horn logic, the maximality condition only applies to the premise in which a negative literal is selected. Any input and derived clause has no more than one positive literal.

14 Equivalence of resolution and chaining

Lifting Lemma. If C and D are instances of disjunctions C' and D', respectively, and E is the resolvent of C and D, then there exists such disjunction E' that E is an instance of E' and E' is the resolvent of C' and D'.

Theorem. Atom F is derivable by chaining from Horn axioms $\{R_1, ..., R_r\}$ if and only if $\neg F$ is refutable by input resolution without factoring in which $\{R_1, ..., R_r, \neg F\}$ are input clauses and $\neg F$ is used in the first resolution step.

Proof. Consider an input resolution refutation satisfying the condition of this theorem. Every application of the resolution rule to a Horn rule (treated as disjunction) maps to a backward chaining step. By induction on the depth of resolution refutations, the entire refutation corresponds to a backward chaining derivation.

Consider a backward chaining derivation. Let us ground it. Suppose a step of this derivation produces goal list H from goal list G and an instance of Horn rule R. This step corresponds to an application of the resolution rule to the disjunction $\neg G$ obtained from G by negating all atoms in it and to the instance of R treated as a disjunction. Due to the lifting lemma, if G is an instance of another goal list G', then there is such goal list H' that H is its instance and $\neg H'$ is the result of applying the resolution rule to $\neg G'$ and R. By induction on the depth of backward chaining derivations, for any such derivation of goal list G which is an instance of G', there exists such resolution refutation that some R_i is used in every step and $\neg F$ is used in its first step. Consequently, there exist a resolution refutation satisfying the condition of this theorem.

It is possible to prove that an atom is derivable by forward chaining from a set of Horn rules if and only if it is derivable in HL for the same set of Horn rules. This is basically a corollary of the above theorem and the completeness of input resolution for Horn clauses, but the proof involves more details. The meaning of the equivalence of chaining and HL is that once MP is replaced by GMP, logical axioms are not needed for inference of atoms from Horn axioms.

15 Evaluable functions and predicates

Computer languages $\Leftrightarrow$ *Recursive functions* $\Leftrightarrow$ *Turing machines* $\Leftrightarrow$ *Markov algorithms* $\Leftrightarrow$ *...*

All recursive functions can be represented by Horn clauses but it is rather inconvenient.

Many tasks are most conveniently specified as a combination of recursive functions (or computer programs) and logical rules. For that reason, so-called evaluable functions and predicates are included in Prolog implementations. They are built-in or user-defined computer programs. Evaluable functions may occur in terms that are predicate arguments. Predicates may also be defined as boolean recursive functions. Languages including both recursive functions and FOL has been extensively researched.

In logic, it is assumed that all functions and predicates are total. Skolem functions must be total. Recursive functions may be partial and their values may be undefined. Nonetheless, HL calculi and models can be extended to include evaluable functions and predicates. But these functions and predicates should not appear in the heads of Horn rules (axioms). Let us call these HL extensions HLE.

Let us adopt backward chaining for HL to its extension with evaluable functions and predicates. During inference, occurrences of evaluable functions and predicates with constant arguments are evaluated as soon as they appear in goal lists. Any complete search strategy for backward chaining in the presence of evaluable functions or predicates should continue derivation search simultaneously with evaluations because some evaluations may never end. If A is an evaluable predicate and the evaluation of ground atom $A(...)$ yields true, then $A(...)$ is considered an axiom. If the evaluation of atom $A(...)$ yields false, then the respective derivation branch should be dropped because it will never lead to a legitimate derivation.

The proof theory of HLE is similar to that of HL: axioms are Horn rules, GMP is the only inference rule. The difference is the following:
- all terms $f(a_1, ..., a_k)$, where f is an evaluable function and $a_1, ..., a_k$ are constants, are replaced by their values provided that they yield ones
- all atoms of evaluable predicates with constant arguments yielding true are considered axioms

16 Models for logic with evaluable predicates

Models for FOL rely on the totality of predicates and functions. For example, what is the truth value of $A \wedge B$ if A is true and B is undefined. Standard boolean functions are not sufficient for defining the truth values of formulas with evaluable predicates so that the axioms of FOL hold.

Example: If the truth value of atom A is undefined, then the truth value of $\neg A$ should also be undefined. What is the truth value of $A \vee \neg A$ is then. The only reasonable answer is that it is undefined. What about the law of excluded middle?

Assumptions: If a predicate argument is undefined, then the predicate truth value is also undefined. If a function argument is undefined, then the function value is also undefined.

Let # denote an undefined truth value. Let $|A|$ denote the truth value of formula A. Fortunately, models for the Horn fragment of FOL with evaluable predicates can be easily defined. Let us define

$$|A_1 \wedge ... \wedge A_k \Rightarrow A| = \begin{cases} |A| & \text{if } |A_1| = 1, ..., |A_k| = 1 \\ 1 & \text{otherwise} \end{cases}$$

Definition. An HLE model is such assignment of truth values (0, 1, #) to every atom with constant arguments that all ground instances of axioms (Horn rules) have truth value 1.

We basically extended HL models onto Horn formulas with evaluable predicates because $|A_1 \wedge ... \wedge A_k \Rightarrow A|$ is defined as in HL when none of the atoms in this Horn clause is undefined.

Summary: We specified both proof and model theories for Horn logic with evaluable functions and predicates. Let us call it HLE. There exist efficient inference algorithms for for it. Prolog implements the concept of Horn logic with evaluable functions and predicates.

17 HLE soundness and completeness

Theorem. Any atom with constant arguments is derivable in HLE if and only if it is valid.

1. Derivable ⇒ valid

This is proved by induction on the depth of derivations. Clearly, $|B| = 1$ for derivations of depth zero because B is an axiom instance. Suppose $|A| = 1$ for any atom derivable with depth n. Consider a derivation of depth $n + 1$. Let us ground this derivation, i.e. apply substitutions from GMP recursively from bottom up. Look at the last application of MP in this derivation:

$$\frac{A_1\theta \quad ... \quad A_k\theta \quad A_1\theta \wedge ... \wedge A_k\theta \Rightarrow A\theta}{A\theta}\, GMP$$

By the induction assumption, $|A_i\theta| = 1$ for $i = 1...k$. Since $|A_1\theta \wedge ... \wedge A_k\theta \Rightarrow A\theta| = 1$, $|A\theta|$ must be 1.

2. Vaid ⇒ derivable

Suppose B is an atom with constant arguments, $|B| = 1$ in all HLE models, and B is not derivable. Let us look at an assignment M of truth values to atoms with constant arguments in which $|C| = 1$ for every such atom C with constant arguments that is derivable, and $|D| = \#$ for every other ground atom D. Clearly, $|B| = \#$ in M.

If A is an axiom instance, then $|A| = 1$ in M. If $|A_1 \wedge ... \wedge A_k \Rightarrow A| \neq 1$ for ground rule instance $A_1 \wedge ... \wedge A_k \Rightarrow A$, then all $|A_i| = 1$ for $i = 1...k$ and $|A| \neq 1$. Therefore, all A_i are derivable according to the definition of M, and thus, A is derivable from $A_1, ..., A_k$ and the above Horn rule by applying GMP. Therefore M satisfies all the conditions for HLE models and the assumption that $|B| = 1$ in all HLE models but not derivable cannot be true.

18 Fuzzy truth values

It makes sense to allow evaluable predicates whose range is real numbers in interval $[0, 1]$ as opposed to dicrete values from the set $\{0, 1\}$. The evaluable predicates yielding real numbers are called fuzzy. Fuzzy evaluable predicates are quite natural in real-life applications. Often, it is only possible to calculate an approximate truth value of a certain predicate for given constant arguments. It will be shown later that one important type of fuzzy evaluable predicates is the predicates implemented by neural networks.

It is possible to fit fuzzy evaluable predicates into the framework of Horn logic with evaluable predicates by mapping real values into the set $\{0, 1, \#\}$.

$$P'(x_1, ..., x_k) = \begin{cases} 1 & \text{if } P(x_1, ..., x_k) \geq 1 - t \\ 0 & \text{if } P(x_1, ..., x_k) \leq t \\ \# & \text{otherwise} \end{cases}$$

This may not be the best solution. If evaluable predicates can be fuzzy, then other predicates connected by Horn rules with the former should also be fuzzy, i.e. their truth values are real numbers from interval $[0, 1]$. As a result, models for logics with fuzzy predicates are real-valued as opposed to boolean. These logics would also require different calculi than those based on FOL.

? Fuzzy logic ?

19 Neural networks

Let $\mathbb{C}$ be a compact subset of $\mathbb{R}$, for example, interval $[-1, 1]$. Neural network is a function $f : \mathbb{C}^n \to \mathbb{E}$

or a function

$f_i : \bigcup_{i=1...\infty} \mathbb{C}^{in} \to \mathbb{E}$

where $\mathbb{E}$ is a metric space, $n = 1, 2, ...$ Usually, $\mathbb{E}$ is $\mathbb{R}$ or $\mathbb{R}^k$. If $\mathbb{E}$ is $\mathbb{R}$, then the following is a standard form of NN:

$f(x_1, ..., x_m) = \Sigma_{i=1...k} c_i \phi(\Sigma_{j=1...m} a_{ij} x_j + b_i)$

where function $\phi : \mathbb{R} \to \mathbb{R}$ is differentiable, ϕ is applied elementwise. Function f, i.e. a NN, could be defined by another expressions which is differentiable wrt all its parameters such as a_{ij}, b_i, c_i for f. All the parameters are real numbers.

Theorem. For any continuous function $g : \mathbb{R} \to \mathbb{R}$ and $\epsilon > 0$, there are such n, a_{ij}, b_i, c_i for $i \in \{1, ..., k\}, j \in \{1, ..., m\}$ that $|f(x_1, ..., x_m) - g(x_1, ..., x_m)| < \epsilon$ for any $x_1, ..., x_m$.

Function f approximates unknown function g. The parameters of f such as a_{ij}, b_i, c_i for the above form of f are learned by means of an iterative process. For the success of the learning process, it is necessary to have a huge selection of g samples. A sample is such constants $d_1, ..., d_m < e$ that $g(d_1, ..., d_m) = e$.

The learning process starts with assigning arbitrary values to the parameters of f. At every step, the parameters are changed according to the value of their partial derivatives wrt the parameters such as a_{ij}, b_i, c_i and the difference between the value of f and the value of g for some samples.

Object embedding

$e : \mathbb{D} \to \mathbb{R}^n$ (fixed length)

Some domains require variable-length embeddings:

$e : \mathbb{D} \to \bigcup_{i=1...\infty} \mathbb{R}^{in}$ (variable length)

Examples of fixed-length embedding: finite sets, words

Examples of variable-length embeddings: text, trees, graphs

If a parameter of g requires variable-length embeddings, then the form of a NN approximating g should be different than the aforementioned form of f.

20 Neural networks implementing fuzzy evaluable predicates

Suppose objects from various domains are embedded as real-valued vectors of a fixed length. Let $[\![x]\!]$ denote the embedding of x: vector of length n. Predicates of one argument $P(x)$ can be approximated by the following NN $\mathcal{N} : \mathbb{C}^n \to [0, 1]$:
$\mathcal{N}([\![x]\!]) = \sigma(c \cdot \varphi(A[\![x]\!] + b))$
where A is a matrix, b, c are vectors, σ and φ are differentiable functions, φ is applied elementwise. The range of σ is the interval $[0, 1]$. Usually φ is tanh, and σ is the sigmoid function:
$\sigma(x) = \frac{1}{1+e^{-x}}$
This representation of predicate P is differentiable with respect to all parameters: elements of matrix A and elements of vectors b, c.

Any predicate $P(x_1...x_m)$ can be approximated similarly by considering the concatenation of vectors $[\![x_1]\!]...[\![x_m]\!]$. Another approximation has become more prevalent for relations, i.e. predicates with two arguments $P(x, y)$:
$\mathcal{N}([\![x]\!], [\![y]\!]) = \sigma(c \cdot \varphi(D[\![x]\!][\![y]\!] + E\left|\begin{matrix}[\![x]\!] \\ [\![y]\!]\end{matrix}\right| + b))$
where D is a tensor of rank three, E is a matrix. This approximation with tensors works better than the aforementioned matrix approximation of functions with multiple arguments because it captures argument dependencies via products of elements of their embeddings. This approximation expression can be extended onto predicates with more than two arguments. For predicates with three arguments, tensors of rank four are utilized, and so on.

These NNs play the role of fuzzy evaluable predicates.

21 Variable-length embeddings

The most impressive results of using NNs are related to text processing such as automatic translation, essay writing, etc. These results are due to the representation of text as a sequence of word embeddings. Every word is represented by a fixed-length vector of real numbers. The size of these vectors is usually between 100 and 1000. This size makes it possible to capture important semantic aspects of words. The domain of a NN implementing a predicate with one textual argument is $\bigcup_{i=1...\infty} \mathbb{C}^{in}$ where $100 < n < 1000$.

Structural data such as trees or lists can be encoded as sequences, and thus, embedded as sequences of fixed-length vectors as well. Graph embeddings are an area of active research now. NNs implementing predicates with such arguments are more complicated.

So-called recurrent NNs accept variable-length vectors as input. The output of recurrent NNs can have a variable or fixed length. Example: text translation to another language.

Such recurrent NNs shrinking variable-length real-valued vectors into fixed-length vectors can be incorporated into NNs implementing predicates. For example, a predicate having two arguments (x, z) with fixed-length embeddings and one argument (y) with a vector sequence embedding is approximated as

$$\mathcal{M}([\![x]\!], [\![y]\!], [\![z]\!]) = \sigma(c \cdot \varphi(D[\![x]\!]\mathcal{N}([\![y]\!])[\![z]\!] + E\left|\begin{matrix}[\![x]\!] \\ \mathcal{N}([\![y]\!]) \\ [\![z]\!]\end{matrix}\right| + b))$$

where $\mathcal{N}$ refers to a recurrent NN converting y to a fixed-length vector, i.e. $\mathcal{N} : \bigcup_{i=1...\infty} \mathbb{C}^{in} \to \mathcal{C}^n$. Here, the representation of predicate P is differentiable with respect to all parameters.

The design and implementation of NNs implementing predicates is an area of active research at the moment.

22 References

This file
http://sakharov.net/download/LogicComputingNN.pdf

Textbooks

Artificial Intelligence. A Modern Approach.
https://scholar.alaqsa.edu.ps/9195/1/Artificial Intelligence A Modern Approach (3rd Edition).pdf (PDFDrive).pdf

Symbolic logic and mechanical theorem proving.
https://obuchalka.org/20200209118258/matematicheskaya-logika-i-avtomaticheskoe-dokazatelstvo-teorem-chen-ch-li-r-1983.html