Recurrent Neural Networks and Logic Programs

- The Very Idea
- Propositional Logic Programs
- Propositional Logic Programs and Learning
- Propositional Logic Programs and Modalities
- First Order Logic Programs

A contraction mapping \( f \) defined on a complete metric space \((X, d)\) has a unique fixed point. The sequence \( y, f(y), f(f(y)), \ldots \) converges to this fixed point for any \( y \in X \). [Banach Contraction Mapping Theorem]

Consider logic programs, whose immediate consequence operator is a contraction. Fitting: Metric Methods – Three Examples and a Theorem, Journal Logic Programming 21, 113-127: 1994

Every continuous function on the reals can be uniformly approximated by feedforward connectionist networks. Funahashi: On the Approximate Realization of Continuous Mappings by Neural Networks, Neural Networks 2, 183-192: 1989.

Consider logic programs, whose immediate consequence operator is continuous on the reals. H., Kalinke, Störr: Approximating the Semantics of Logic Programs by Recurrent Neural Networks, Applied Intelligence 11, 45-59: 1999.
Propositional Logic Programs


- We consider normal propositional logic programs $\mathcal{P}$ over propositional variables $p, q, \ldots$.

- Literals are denoted by $L_1, L_2, \ldots$.

- $B_\mathcal{P}$ is the Herbrand base for $\mathcal{P}$.

- A level mapping for $\mathcal{P}$ is a function $|\_| : B_\mathcal{P} \rightarrow \mathbb{N}$.

- $|q|$ is called the level of $q$.

- $p$ refers to $q$ if there is a clause in $\mathcal{P}$ with head $p$ and $q$ occurring in its body.

- $p$ depends on $q$ if either $p = q$ or there is a sequence $p = p_1, \ldots, p_n = q$, where each $p_i$ refers to $p_{i+1}$, $1 \leq i < n$. 
Acceptable Programs

- Let $\text{Neg}_P$ be the set of propositional variables occurring in $P$ which occur in a negative literal in the body of some clause in $P$.

- Let $\text{Neg}_P^*$ be the set of propositional variables occurring in $P$ on which the variables in $\text{Neg}_P$ depend.

- Let $P^-$ be the set of clauses occurring in $P$ whose head is in $\text{Neg}_P^*$.

- Let $\text{comp}(P)$ denote the completion of $P$.

- Given $P$ and $|\_|$. Let $I$ be a model for $P$ whose restriction to the propositional variables in $\text{Neg}_P^*$ is a model for $\text{comp}(P)$.
  
  - $P$ is acceptable wrt $|\_|$ and $I$ if for every clause $p \leftarrow L_1 \land \ldots \land L_n \in P$ and for every $i$, $1 \leq i \leq n$, we find that $I \models \land_{j=1}^{i-1} L_j$ implies $|p| > |L_j|$.
  
  - $P$ is acceptable if it is acceptable wrt some level mapping and some model.

- The question whether a given program is acceptable is undecidable.
Metric Spaces

Let $X$ be a non-empty set. $d : X \times X \rightarrow \mathbb{R}$ is called a metric (on $X$) and $(X, d)$ is called a metric space if

1. for all $y, z \in X$ we have $d(y, z) \geq 0$ and $d(y, z) = 0$ iff $y = z$.
2. for all $y, z \in X$ we have $d(y, z) = d(z, y)$.
3. for all $w, y, z \in X$ we have $d(w, z) \leq d(w, y) + d(y, z)$.

A sequence $(y_n)$ in $X$ is said to converge to $y \in X$ if

$$(\forall \epsilon > 0)(\exists k \in \mathbb{N})(\forall m \geq k) \ d(y_m, y) \leq \epsilon.$$ 

A sequence $(y_n)$ in $X$ is a Cauchy sequence if

$$(\forall \epsilon > 0)(\exists k \in \mathbb{N})(\forall m, n \geq k) \ d(x_m, x_n) \leq \epsilon.$$ 

$(X, d)$ is complete if every Cauchy sequence converges.
Contraction Mappings

- Let \((X, d)\) be a metric space.

- \(f : X \rightarrow X\) is called contraction if
  \[
  (\exists 0 \leq k < 1)(\forall y, z \in X) d(f(y), f(z)) \leq kd(y, z).
  \]

- \(y \in X\) is called fixed point of \(f\) iff \(f(y) = y\).

- **Banach Contraction Mapping Theorem**
  Let \(f\) be a contraction defined on a complete metric space \((X, d)\). Then,
  \[
  \begin{align*}
  &\text{\(f\) has a unique fixed point } y \in X, \\
  &\text{the sequence } z, f(z), f(f(z)), \ldots \text{ converges to } y \text{ for any } z \in X.
  \end{align*}
  \]

- **Proposition** Let \(\mathcal{P}\) be an acceptable program. There exists a complete metric space \((X_\mathcal{P}, d_\mathcal{P})\) such that \(T_\mathcal{P}\) is a contraction on \((X_\mathcal{P}, d_\mathcal{P})\).
3-Layer Recurrent Networks

► all units at each point in time do:
  ▶ compute sum of their weighted inputs.
  ▶ apply linear, threshold or sigmoidal function to obtain output.
Hidden Layers are Needed

- XOR can be encoded as a normal logic program \( \{ r \leftarrow p \land \neg q, \ r \leftarrow \neg p \land q \} \).

- Proposition
  2-layer networks of binary threshold units cannot compute \( T_P \) for definite \( P \).

  - Consider \( \{ p \leftarrow q, \ p \leftarrow r \land s, \ p \leftarrow t \land u \} \).
Relating Propositional Programs to Networks

▶ **Theorem**
For each program $\mathcal{P}$ there exists a 3-layer feedforward network computing $T_\mathcal{P}$.

▶ Let $m$ be the number of propositional variables occurring in $\mathcal{P}$.
Let $n$ be the number of clauses occurring in $\mathcal{P}$.

▶ **Translation Algorithm**

1. **Input and output layer:** vector of length $n$, where the $i$-th unit represents the $i$-th variable, $1 \leq i \leq n$. Set their thresholds to $.5$.

2. For each clause of the form $p \leftarrow L_1 \land \ldots \land L_k \in \mathcal{P}$, $k \geq 0$, do:
   2.1 Add a binary threshold unit $c$ to the hidden layer.
   2.2 Connect $c$ to the unit representing $p$ in the output layer with weight 1.
   2.3 For each literal $L_j$, $1 \leq j \leq k$ connect the unit representing $L_j$ in the input layer to $c$ and, if $L_j$ is an atom, then set the weight to 1; otherwise set the weight to $-1$.
   2.4 Set the threshold $\theta_c$ of $c$ to $l - .5$, where $l$ is the number of positive literals occurring in $L_1 \land \ldots \land L_k$. 

ICCL
International Center for Computational Logic
Algebra, Logic and Formal Methods in Computer Science
Applying the Translation Algorithm

Consider \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}.

Blue connections have weight 1; red connections have weight $-1$. 
Some Observations

- Let $m$ be the number of propositional variables occurring in $\mathcal{P}$. Let $n$ be the number of clauses occurring in $\mathcal{P}$.

- The number of units is bound by $O(m + n)$.
  The number of connections is bound by $O(m \times n)$.

- $T_{\mathcal{P}}(I)$ is computed in 2 steps.

- The parallel computational model is optimal.

- Corollary Let $\mathcal{P}$ be an acceptable program. There exists a 3-layer recurrent network such that each computation starting with an arbitrary initial input converges and yields the unique fixed point of $T_{\mathcal{P}}$.

- The time needed to settle down to the unique stable state is bound by $O(n)$. 
Propositional Logic Programs and Learning

- **Observation**: Networks of binary threshold units cannot be trained.
- **Idea**: Replace binary threshold units by sigmoidal ones and apply backpropagation.
  - When is such a unit considered to be active / passive?
  - Minimum activation for a unit to be considered “active”: $0 < A_{\text{min}} < 1$.
    - Units with output $o \in (A_{\text{min}}, 1]$ are active.
  - Maximum activation for a unit to be considered “passive”: $-1 < A_{\text{max}} < 0$.
    - Units with output $o \in [-1, A_{\text{max}})$ are passive.
  - Wlog, let $A_{\text{max}} = -A_{\text{min}}$.
  - What happens of output $o \in [A_{\text{max}}, A_{\text{min}}]$?
    - Modified translation algorithm prevents such cases.
    - Experiments suggest that it hardly occurs during training.
    - Penalties may be added to the training algorithm.
Some Notation

- \( q \): number of clauses \( C_l \) occurring in \( \mathcal{P} \).
- \( W / -W \): weight of connections associated with positive / negative literals.
- \( \theta_l \): threshold of hidden unit \( u_l \) associated with clause \( C_l \).
- \( \theta_p \): threshold of output unit representing atom \( p \).
- \( k_l \): number of literals in the body of clause \( C_l \).
- \( p_l \): number of positive literals in the body of clause \( C_l \).
- \( n_l \): number of negative literals in the body of clause \( C_l \).
- \( \mu_l \): number of clauses in \( \mathcal{P} \) with the same atom in the head as \( C_l \).
A Modified Translation Algorithm

0 Input and output layer: vector of units representing propositional variables.
1 Calculate $m = \max(\{k_l \mid C_l \in \mathcal{P}\} \cup \{\mu_l \mid C_l \in \mathcal{P}\})$.
2 Calculate $A_{\text{min}} > \frac{m-1}{m+1}$.
3 Calculate $W \geq \frac{2}{\beta} \cdot \frac{\ln(1+A_{\text{min}})-\ln(1-A_{\text{min}})}{m(A_{\text{min}}-1)+A_{\text{min}}+1}$.
4 For each clause $C_l \in \mathcal{P}$ of the form $p \leftarrow L_1 \land \ldots \land L_k$ do:
   4.1 Add a neuron $u_l$ to the hidden layer.
   4.2 Connect $u_l$ to the unit representing $p$ in the output layer with weight $W$.
   4.3 For each $L_j$, $1 \leq j \leq k$, connect the unit representing $L_j$ in the input layer to $u_l$ and, if $L_j$ is an atom, then set the weight to $W$, otherwise set the weight to $-W$.
   4.4 Set $\theta_l := \frac{(1+A_{\text{min}})(k_l-1)}{2}W$.
   4.5 Set $\theta_p := \frac{(1+A_{\text{min}})(1-\mu_l)}{2}W$.
5 Set $g(x) = x$ as the activation function for all units in the input layer.
6 Set $h(x) = \frac{2}{1+e^{\beta x}} - 1$ as the activation function for all units in the hidden and output layer.
7 Fully connect the network setting all other weights to 0.
Results

▶ Theorem  For each propositional logic program $\mathcal{P}$, there exists a 3-layer feedforward connectionist network computing $T_\mathcal{P}$.

▶ Corollary  Let $\mathcal{P}$ be an acceptable program. There exists a recurrent connectionist network with a 3-layer feedforward kernel such that, starting from an arbitrary input, the network converges to a stable state and yields the unique fixpoint of $T_\mathcal{P}$.

▶ Extensions

▷ Answer set programming: classical and default negation.
▷ Default reasoning with priorities among defaults.
Theory Refinement

► Given: a priori knowledge about a problem in the form of a logic program $\mathcal{P}$.

▷ Use the modified translation algorithm to construct a connectionist network computing $T_{\mathcal{P}}$.

► If training examples are additionally given, then:

▷ Apply backpropagation to the kernel of the network to obtain a modified $T_{\mathcal{P}}$.

▷ Desired result: better performance on test examples.


▷ d’Avila Garcez, Broda, Gabbay: 2002 report on applications from

- DNA sequence analysis and
- power systems fault diagnosis

with very good performance.
After training the kernel we obtain a modified $T_P$.

- But, $T_P$ is encoded in the weights of the kernel.
- There is no direct declarative reading of these weights.
- What is the corresponding logic program $P$?
- Is it acceptable?

For details see:

- Andrews, Diedrich, Tickle: A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. Knowledge-Based Systems 8: 1995,
Propositional Logic Programs and Modalities

- Introduce modalities □ and ◊ as well as relations between worlds.
- Refine translation algorithm such that the immediate consequence operator is again computed by a kernel.
- For each world, turn the kernel into a recurrent network.
- Connect worlds with respect to the given set of relations and the Kripke semantics of the modalities.
- Details: later, if there is time.
First Order Logic Programs

▶ **Given:** Logic program \( \mathcal{P} \) together with \( T_\mathcal{P} : 2^{B_\mathcal{P}} \rightarrow 2^{B_\mathcal{P}} \).

▶ **Goal:** Find a class \( C \) of programs such that for each \( P \in C \) we find

\[
\iota : 2^{B_\mathcal{P}} \rightarrow \mathbb{R} \quad \text{and} \quad f_\mathcal{P} : \mathbb{R} \rightarrow \mathbb{R}
\]

satisfying:

1. \( T_\mathcal{P}(I) = I' \) implies \( f_\mathcal{P}(\iota(I)) = \iota(I') \).

2. \( f_\mathcal{P}(r) = r' \) implies \( T_\mathcal{P}(\iota^{-1}(r)) = \iota^{-1}(r') \),

\[ \Rightarrow f_\mathcal{P} \text{ is a sound and complete encoding of } T_\mathcal{P}, \]

3. \( T_\mathcal{P} \) is a contraction on \( 2^{B_\mathcal{P}} \) iff \( f_\mathcal{P} \) is a contraction on \( \mathbb{R} \),

\[ \Rightarrow \text{ contraction property and fixed points are preserved}, \]

4. \( f_\mathcal{P} \) is continuous on \( \mathbb{R} \),

\[ \Rightarrow \text{ network approximating } f_\mathcal{P} \text{ is known to exist}. \]

The Connectionist Network

output layer/unit

hidden layer

input layer/unit
Acyclic Logic Programs

- Let \( \mathcal{P} \) be a logic program.
- A level mapping for \( \mathcal{P} \) is a function \(|-| : B_\mathcal{P} \rightarrow \mathbb{N} \).
  - We define \(|\neg A| = |A|\).
- We can associate a metric \( d_\mathcal{P} \) with \( \mathcal{P} \) and \(|-|\). Let \( I, J \in 2^{B_\mathcal{P}} \):
  \[
d_\mathcal{P}(I, J) = \begin{cases} 
0 & \text{if } I = J \\
2^{-n} & \text{if } n \text{ is the smallest level on which } I \text{ and } J \text{ differ.}
\end{cases}
\]
- Proposition \((2^{B_\mathcal{P}}, d_\mathcal{P})\) is a complete metric space. [Fitting:94]
- \( \mathcal{P} \) is said to be acyclic wrt a level mapping \(|-|\), if for every \( A \leftarrow L_1 \land \ldots \land L_n \in \text{ground}(\mathcal{P}) \) we find \(|A| > |L_i|\) for all \( i \).
- \( \mathcal{P} \) is said to be acyclic if \( \mathcal{P} \) is acyclic wrt some level mapping.
- Proposition Let \( \mathcal{P} \) be an acyclic logic program wrt \(|-|\) and \( d_\mathcal{P} \) the metric associated with \( \mathcal{P} \) and \(|-|\), then \( T_\mathcal{P} \) is a contraction on \((2^{B_\mathcal{P}}, d_\mathcal{P})\).
Let \( \mathcal{P} \) be an acyclic logic program, \( I \in 2^{BP} \) an interpretation and \( |_\_| \) an injective level mapping.

We define
\[
\iota(I) = \sum_{A \in I} 4^{-|A|}.
\]

Let \( D \subseteq \mathbb{R} \) be the range of \( \iota \).

\[ D \subseteq [0, \frac{1}{3}] \]

Proposition \( D \) is a closed subset of \( \mathbb{R} \).

|\( |_\_| \) injective \( \Rightarrow \) \( \iota \) invertable.

Extend inverse to \( \iota^{-1} : \mathbb{R} \rightarrow 2^{BP} \).

We have a sound and complete encoding!
We define

\[ f_P : D \rightarrow D : r \mapsto \iota(T_P(\iota^{-1}(r))). \]

\[ I \in 2^{B_P} \xrightarrow{T_P} I' \in 2^{B_P} \]

\[ \iota \downarrow \iota^{-1} \quad \iota \downarrow \iota^{-1} \]

\[ r \in D \xrightarrow{f} r' \in D \]

\( f_P \) is extended to \( f_P : \mathbb{R} \rightarrow \mathbb{R} \) by linear interpolation.

**Proposition** \( f_P \) is a contraction on \( \mathbb{R} \), i.e.

\[ \forall r, r' \in \mathbb{R} : |f_P(r) - f_P(r')| \leq \frac{1}{2} |r - r'|. \]

**Contraction property and fixed points are preserved!**
Approximating the Immediate Consequence Operator

- **Corollary** $f_P$ is continuous.
- **Theorem** [Funahashi:89] Let $\phi(x)$ be a non constant, bounded and monotone increasing continuous function. Let $K \subseteq \mathbb{R}$ be compact and $g : K \rightarrow \mathbb{R}$ continuous. Then for an arbitrary $\varepsilon > 0$, there exists a 3-layer feed forward network with sigmoidal activation function $\Phi$ for the hidden layer, linear activation function for the input and output layer, and input-output function $\tilde{g} : K \rightarrow \mathbb{R}$ such that

\[
\max_{x \in K} |g(x) - \tilde{g}(x)| < \varepsilon.
\]

- Each continuous function $g : K \rightarrow \mathbb{R}$ can be uniformly approximated by input-output functions of 3-layer feed forward networks.

- **Theorem** $f_P$ can be uniformly approximated by input-output functions of 3-layer feed forward networks.

- $T_P$ can be approximated as well by applying $\iota^{-1}$.

Connectionist network approximating immediate consequence operator exists!
Consider $P = \{q(0), q(s(X)) \leftarrow q(X)\}$ and let $|q(s^n(0))| = n + 1$.

- $P$ is acyclic wrt $|\_|$, $|\_|$ is injective.
- $f_P(\nu(I)) = 4^{-|q(0)|} + \sum_{q(X) \in I} 4^{-|q(s(X))|}$
  
  $= 4^{-|q(0)|} + \sum_{q(X) \in I} 4^{-(|q(X)|+1)} = \frac{1+\nu(I)}{4}.

- $B_P$ is least model of $P$, $\nu(B_P) = \frac{1}{3}$.

Approximation of $f_P$ to accuracy $\varepsilon$ yields

$$\tilde{f}(x) \in \left[\frac{1 + x}{4} - \varepsilon, \frac{1 + x}{4} + \varepsilon\right].$$

Starting with some $x$ and iterating $\tilde{f}$ yields in the limit a value

$$r \in \left[\frac{1 - 4\varepsilon}{3}, \frac{1 + 4\varepsilon}{3}\right].$$

Applying $\nu^{-1}$ to $r$ we find

$$q(s^n(0)) \in \nu^{-1}(r) \text{ if } n < -\log_4 \varepsilon - 1.$$
Approximation of Interpretations

Let \( \mathcal{P} \) be a logic program and \(|-|\) a level mapping.

An interpretation \( I \) approximates an interpretation \( J \) to a degree \( n \in \mathbb{N} \) if for all atoms \( A \in B_\mathcal{P} \) with \(|A| < n\), \( A \in I \) iff \( A \in J \).

\( I \) approximates \( J \) to a degree \( n \) iff \( d_\mathcal{P}(I, J) \leq 2^{-n} \).
Approximation of Supported Models

- Given acyclic $\mathcal{P}$ with injective level mapping.
- Let $T_{\mathcal{P}}$ be the immediate consequence operator associated with $\mathcal{P}$ and $M_{\mathcal{P}}$ the least model of $\mathcal{P}$.
- We can approximate $T_{\mathcal{P}}$ by a 3-layer feed forward network.
- We can turn this network into a recurrent one.

Does the recurrent network approximate the supported model of $\mathcal{P}$?

- **Theorem** For an arbitrary $m \in \mathbb{N}$ there exists a recursive network with sigmoidal activation function for the hidden layer units and linear activation functions for the input and output layer units computing a function $\tilde{f}_{\mathcal{P}}$ such that there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and for all $x \in [-1, 1]$ we find

$$d_{\mathcal{P}}(\iota^{-1}(\tilde{f}_{\mathcal{P}}^n(x)), M_{\mathcal{P}}) \leq 2^{-m}.$$