The Propositional CORE Method

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- The Very Idea
- Propositional Logic Programs
- The CORE Method Using Threshold Units
- Abduction
- Backpropagation
- KBANN
- The CORE Method Using Squashing Units

"Logic is everywhere..."
The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators (e.g., Apt, vanEmden 1982).

- **Banach Contraction Mapping Theorem** 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 $y \in X$.

- **Fitting 1994**: Consider logic programs, whose immediate consequence operator is a contraction.

- **Funahashi 1989**: Every continuous function on the reals can be uniformly approximated by feed-forward connectionist networks.

- **Hölldobler, Kalinke, Störr 1999**: Consider logic programs, whose immediate consequence operator is continuous on the reals.
Propositional Logic Programs

▶ We extend propositional languages by the symbol $\top$, which is true under all interpretations.

▶ A propositional logic program over a propositional language $\mathcal{L}$ is a finite set of clauses of the form $A \leftarrow L_1 \land \ldots \land L_n$, where
  
  ▶ $A$ is an atom,
  ▶ $L_i, 1 \leq i \leq n$, are literals or $\top$,
  ▶ $n \geq 1$,
  ▶ $A$ is called head, and
  ▶ $L_1 \land \ldots \land L_n$ is called body.

▶ A clause of the form $A \leftarrow \top$ is called (positive) fact.

▶ $\mathcal{P}$ is definite if the bodies of all clauses of $\mathcal{P}$ consist only of atoms and $\top$.

▶ Notation
  
  ▶ For convenience, $\top$ is often omitted and $A \leftarrow \top$ becomes $A$.
  ▶ In the sequel, let $\mathcal{P}$ denote a propositional program.
  ▶ $R_\mathcal{P}$ denotes the set of propositional variables occurring in $\mathcal{P}$. 
Interpretations

- Let $\mathcal{L}$ be a propositional language and $\{\top, \bot\}$ the set of truth values.
- An **interpretation** $I$ is a mapping $\mathcal{L} \rightarrow \{\top, \bot\}$.
- For a given program $\mathcal{P}$, $I$ can be represented by the set of atoms occurring in $\mathcal{P}$ which are mapped to $\top$ under $I$, i.e. $I \subseteq \mathcal{R}_\mathcal{P}$.
- $2^{\mathcal{R}_\mathcal{P}}$ is the set of all interpretations for $\mathcal{P}$.
- $(2^{\mathcal{R}_\mathcal{P}}, \subseteq)$ is a complete lattice.
- An interpretation $I$ for $\mathcal{P}$ is a model for $\mathcal{P}$ iff $I(\mathcal{P}) = \top$.
- Exercise Consider $\mathcal{P} = \{p, q \leftarrow p, r \leftarrow q\}$.
  - Draw the lattice of all interpretations for $\mathcal{P}$ wrt the $\subseteq$ ordering.
  - Mark the models of $\mathcal{P}$.
Immediate Consequence Operator

- Immediate consequence operator $\mathcal{T}_P : 2^{\mathcal{R}P} \rightarrow 2^{\mathcal{R}P}$:

  $$\mathcal{T}_P(I) = \{ A \mid \text{there is a clause } A \leftarrow L_1 \land \ldots \land L_n \in P \text{ such that } I \models L_1 \land \ldots \land L_n \}.$$  

- *I* is a supported model iff $\mathcal{T}_P(I) = I$.  

- Let $\text{lfp } \mathcal{T}_P$ be the least fixed point of $\mathcal{T}_P$ if it exists.  

- **Exercise** Reconsider $\mathcal{P} = \{p, q \leftarrow p, r \leftarrow q\}$
  
  - Compute $\mathcal{T}_P(\emptyset), \mathcal{T}_P(\mathcal{T}_P(\emptyset)), \ldots$  
  - Mark the supported models of $\mathcal{P}$ in the lattice $(2^{\mathcal{R}P}, \subseteq)$.  

- **Exercise** Let $\mathcal{P}$ be a definite program.
  
  - Show that if $M_1$ and $M_2$ are models of $\mathcal{P}$ then so is $M_1 \cap M_2$.  
  - Let $M_\mathcal{P}$ be the least model of $\mathcal{P}$. Show that $M_\mathcal{P}$ is a supported model.  

- **Exercise** Give an example for $\mathcal{P}$ such that $\text{lfp } \mathcal{T}_P$ does not exist.
The Propositional CORE Method

- Let $\mathcal{L}$ be a propositional logic language.
- Given a logic program $\mathcal{P}$ together with immediate consequence operator $T_\mathcal{P}$.
- Let $|\mathcal{R}_\mathcal{P}| = n$ and $2^{\mathcal{R}_\mathcal{P}}$ be the set of interpretations for $\mathcal{P}$.
- Find a mapping $\text{rep} : 2^{\mathcal{R}_\mathcal{P}} \rightarrow \mathbb{R}^n$.
- Construct a feed-forward network computing $f_\mathcal{P} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, called the core, such that the following holds:
  - If $T_\mathcal{P}(I) = J$ then $f_\mathcal{P}(\text{rep}(I)) = \text{rep}(J)$, where $I, J \in 2^{\mathcal{R}_\mathcal{P}}$.
  - If $f_\mathcal{P}(\bar{s}) = \bar{t}$ then $T_\mathcal{P}(\text{rep}^{-1}(\bar{s})) = \text{rep}^{-1}(\bar{t})$, where $\bar{s}, \bar{t} \in \mathbb{R}^n$.
- Connect the units in the output layer recursively to the units in the input layer.
- Show that the following holds
  - $I = \text{lfp } T_\mathcal{P}$ iff the recurrent network converges to $\text{rep}(I)$.
- Connectionist model generation using recurrent networks with feed-forward core.
3-Layer Recurrent Networks

At each point in time all units do:

- apply activation function to obtain potential,
- apply output function to obtain output.
Propositional CORE Method using Binary Threshold Units

► Let $\mathcal{L}$ be the language of propositional logic.
► Let $\mathcal{P}$ be a propositional logic program, e.g.,

$$\mathcal{P} = \{ p, r \leftarrow p \land \neg q, \ r \leftarrow \neg p \land q \}. $$

$$\mathcal{T}_\mathcal{P}(I) = \{ A \mid A \leftarrow L_1 \land \ldots \land L_m \in \mathcal{P} \text{ such that } I \models L_1 \land \ldots \land L_m \}. $$

$$\mathcal{T}_\mathcal{P}(\emptyset) = \{ p \}$$

$$\mathcal{T}_\mathcal{P}(\{ p \}) = \{ p, r \}$$

$$\mathcal{T}_\mathcal{P}(\{ p, r \}) = \{ p, r \} = \text{lfp } \mathcal{T}_\mathcal{P}$$
Representing Interpretations

- $2^{\mathcal{R}_P}$
- Let $n = |\mathcal{R}_P|$ and identify $\mathcal{R}_P$ with $\{1, \ldots, n\}$.
- Define $\text{rep} : 2^{\mathcal{R}_P} \rightarrow \mathbb{R}^n$ such that for all $1 \leq j \leq n$ we find:

$$\text{rep}(I)[j] = \begin{cases} 
1 & \text{if } j \in I, \\
0 & \text{if } j \not\in I.
\end{cases}$$

E.g., if $\mathcal{R}_P = \{p, q, r\} = \{1, 2, 3\}$ and $I = \{p, r\}$ then $\text{rep}(I) = (1, 0, 1)$.

- Other encodings are possible, e.g.,

$$\text{rep}'(I)[j] = \begin{cases} 
1 & \text{if } j \in I, \\
-1 & \text{if } j \not\in I.
\end{cases}$$
Computing the Core

- Consider again \( \mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\} \).
- A translation algorithm translates \( \mathcal{P} \) into a core of binary threshold units, where \( \omega \in \mathbb{R}^+ \):

- Exercise Specify the core for \( \{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6\} \).
- Proposition 2-layer networks cannot compute \( \top_{\mathcal{P}} \) for definite \( \mathcal{P} \).
- Theorem For each program \( \mathcal{P} \), there exists a core computing \( \top_{\mathcal{P}} \).
Adding Recurrent Connections

Recall $\mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$. 

![Diagram showing the process of adding recurrent connections](image)
Metrics

▶ A metric on a space $M$ is a mapping $d : M \times M \to \mathbb{R}$ such that

- $d(x, y) = 0$ iff $x = y$,
- $d(x, y) = d(y, x)$, and
- $d(x, y) \leq d(x, z) + d(z, y)$.

▶ Let $(M, d)$ be a metric space and $S = (s_i | s_i \in M)$ a sequence.

- $S$ converges if $(\exists s \in M)(\forall \varepsilon > 0)(\exists N)(\forall n \geq N) d(s_n, s) \leq \varepsilon$.
- $S$ is Cauchy if $(\forall \varepsilon > 0)(\exists N)(\forall n, m \geq N) d(s_n, s_m) \leq \varepsilon$.
- $(M, d)$ is complete if every Cauchy sequence converges.

▶ A mapping $f : M \to M$ is a contraction on $(M, d)$ if $(\exists 0 < k < 1)(\forall x, y \in M) d(f(x), f(y)) \leq k \times d(x, y)$. 
Exercise  Let $\mathcal{P}$ be a logic program, $2^\mathcal{R}\mathcal{P}$ the set of all interpretations for $\mathcal{P}$, and $T_\mathcal{P}$ the associated immediate consequence operator.

- Specify a metric $d_\mathcal{P}$ such that $T_\mathcal{P}$ is a contraction on $(2^\mathcal{R}\mathcal{P}, d_\mathcal{P})$ for definite $\mathcal{P}$.
- Can the specification be extendend to arbitrary programs?
Strongly Determined Programs

A logic program $\mathcal{P}$ is said to be strongly determined if there exists a metric $d$ on the $2^{\mathcal{R}}$ such that $T_{\mathcal{P}}$ is a contraction wrt $d$.

Exercise Are the following programs strongly determined?

- $\{p, q \leftarrow p, r \leftarrow q\}$,
- $\{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6\}$,
- $\{p \leftarrow \neg p\}$.

Corollary Let $\mathcal{P}$ be a strongly determined program. Then there exists a core with recurrent connections such that the computation with an arbitrary initial input converges and yields the unique fixed point of $T_{\mathcal{P}}$. 
Time and Space Complexity

- Let $n$ be the number of clauses and $m$ be the number of propositional variables occurring in $\mathcal{P}$.
  - $2m + n$ units, $2mn$ connections in the core.
  - $T_\mathcal{P}(I)$ is computed in 2 steps.
  - The parallel computational model to compute $T_\mathcal{P}(I)$ is optimal.
  - The recurrent network settles down in $3n$ steps in the worst case.

- Exercise Give an example of a program with worst case time behavior.
Short Term versus Long Term Memory

- Consider $\mathcal{P} = \{ l \leftarrow e \land o, o \leftarrow T \}$.
- $\text{lfp } T_{\mathcal{P}} = \{ o \}$.

As long as unit $e$ in the output layer is externally activated (or clamped) the input and output layers represent the fact $e \leftarrow T$.

If the external activation is withdrawn, the network returns to its original stable state.

This can be used in abductive frameworks to test whether certain sets of abducibles explain some given observations.
Stable Coalitions

- Consider $\mathcal{P} = \{p \leftarrow r, p \leftarrow q, q \leftarrow p\}$.
- lfp $T_{\mathcal{P}} = \emptyset$.

- After externally activating unit $r$ in the output layer until the net is stable, stable coalitions remain stable even if the external activation is withdrawn.

- We may need to break such stable coalitions, e.g. by externally deactivating units $p$ and $q$ for three time steps.
Abductive Frameworks and Observations

- Let $\mathcal{L}$ be a language.
- Let $\mathcal{K} \subseteq \mathcal{L}$ be a set of formulas called the knowledge base, $\mathcal{A} \subseteq \mathcal{L}$ be a set of formulas called abducibles, and $\models \subseteq 2^{\mathcal{L}} \times \mathcal{L}$ a logical consequence relation. $\langle \mathcal{K}, \mathcal{A}, \models \rangle$ is called abductive framework.
- An observation $\mathcal{O}$ is a subset of $\mathcal{L}$.
- Here
  - $\mathcal{K}$ is a logic program $\mathcal{P}$.
  - $\mathcal{L}$ is the language underlying $\mathcal{P}$.
  - $\mathcal{R}_P^D = \{ A \in \mathcal{R}_P | A \leftarrow \text{Body} \in \mathcal{P} \}$ is the set of defined predicates in $\mathcal{P}$.
  - $\mathcal{R}_P^U = \mathcal{R}_P \setminus \mathcal{R}_P^D$ is the set of undefined predicates in $\mathcal{P}$.
  - $\mathcal{A}$ is the set $\{ A \leftarrow \top | A \in \mathcal{R}_P^U \}$.
  - $\models$ is $\models_{lm}$, where $\mathcal{P} \models_{lm} F$ iff $\text{lfp } T_\mathcal{P}(F') = \top$.
  - Observations are usually sets containing a single literal, in which case we simply write $\mathcal{O} = L$ instead of $\mathcal{O} = \{L\}$. 
Explanations

- Let $\mathcal{L}$ be a language.
- Let $\langle \mathcal{K}, \mathcal{A}, \models \rangle$ be an abductive framework and $\mathcal{O}$ an observation.
- $\mathcal{O}$ is explained by $\mathcal{E}$ (or $\mathcal{E}$ is an explanation for $\mathcal{O}$) iff
  - $\mathcal{E} \subseteq \mathcal{A}$,
  - $\mathcal{K} \cup \mathcal{E}$ is satisfiable,
  - $\mathcal{K} \cup \mathcal{E} \models L$ for each $L \in \mathcal{O}$.
- An explanation $\mathcal{E}$ for $\mathcal{O}$ is said to be a minimal iff there is no explanation $\mathcal{E}' \subset \mathcal{E}$ for $\mathcal{O}$.
Why is the Grass is Wet?

- Consider $\mathcal{P} = \{\text{wetGrass} \leftarrow \text{raining}, \ w \leftarrow \text{sprinklerOn}\}$ and let $\mathcal{O} = w$.
- Then,
  - $\mathcal{A} = \{r \leftarrow \top, \ s \leftarrow \top\}$.
  - $\{r \leftarrow \top\}$ and $\{s \leftarrow \top\}$ are the minimal explanations for $w$. 

![Diagram showing the process of explanation and stability]
Some Remarks on Abduction

- $E$ generates all possible explanations in a predefined sequence.
  - This can be organized such that minimal explanations are generated first.
- The approach can be extended to
  - deactivate stable coalitions before new possible explanations are tested,
  - handle integrity constraints, i.e., expressions of the form $\bot \leftarrow L_1 \land \ldots \land L_n$,
  - use a more sophisticated control which – for example – signals and stops the process if no explanation has been found.
  - use of a counter instead of $S$,
  - incorporate integrity constraints,
  - encode the control as a logic program.
Theorem (Funahashi 1989) Suppose that $\Psi : \mathbb{R} \rightarrow \mathbb{R}$ is a squashing function. Let $K \subseteq \mathbb{R}^n$ be compact, let $f : K \rightarrow \mathbb{R}$ be continuous, and let $\varepsilon > 0$. Then there exists a 3-layer feed-forward network with output function $\Psi$ for the hidden layer and linear output function for the input and output layer whose input-output mapping $\hat{f} : K \rightarrow \mathbb{R}$ satisfies

$$\max_{x \in K} |f(x) - \hat{f}(x)| < \varepsilon.$$ 

Every continuous function $f : K \rightarrow \mathbb{R}$ can be uniformly approximated by input-output functions of 3-layer feed-forward networks.
Squashing Functions and Squashing Units

- A function is a **squashing function** if it is non-constant, bounded, monotone increasing and continuous.

**Examples**

- Sigmoidal function,
- Bipolar sigmoidal function, also called hyperbolic tangent.

**Squashing Units**

- $u_k$ is a sigmoidal unit if

$$\Phi(i_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j$$
$$\Psi(p_k) = v_k = \frac{1}{1 + e^{-\beta(p_k - \theta_k)}}$$

where $\theta_k \in \mathbb{R}$ is a threshold (or bias) and $\beta > 0$ a steepness parameter.

- $u_k$ is a bipolar sigmoidal unit if

$$\Phi(i_k) = p_k = \sum_{j=1}^{m} w_{kj} v_j$$
$$\Psi(p_k) = p_k = \frac{2}{1 + e^{-2p_k}} - 1 = \tanh(p_k)$$
Backpropagation

- Training set of input-output pairs \( \{(i^l, o^l) \mid 1 \leq l \leq n\} \).
- Minimize \( E = \sum_l E^l \) where \( E^l = \frac{1}{2} \sum_k (o_k^l - v_k^l)^2 \).
- Gradient descent algorithm to learn appropriate weights.
- Backpropagation
  - Initialize weights arbitrarily.
  - Do until all input-output patterns are correctly classified.
    1. Present input pattern \( \vec{i}^t \) at time \( t \).
    2. Compute output pattern \( \vec{v}^t \) at time \( t + 2 \).
    3. Change weights according to \( \Delta w_{ij}^l = \eta \delta_i^l v_j^l \), where

\[
\delta_i^l = \left\{ \begin{array}{ll}
\Psi'_i(p_i^l) \times (o_i^l - v_i^l) & \text{if } i \text{ is output unit,} \\
\Psi'_i(p_i^l) \times \sum_k \delta_k^l w_{ki} & \text{if } i \text{ is hidden unit,} 
\end{array} \right.
\]

\( \eta > 0 \) is called learning rate.
Output Functions Revisited

- Remember sigmoidal function (with $\beta = 1$):
  \[ v_i = \frac{1}{1 + e^{-(\sum_j w_{ij} v_j - \theta_i)}} \]

- We find
  \[ \frac{dv_i}{d(\sum_j w_{ij} v_j + \theta_i)} = v_i(1 - v_i). \]

- Hence
  \[ \delta_i^l = \begin{cases} 
  v_i^l(1 - v_i^l)(o_i^l - v_i^l) & \text{if } u_i \text{ is an output unit,} \\
  v_i^l(1 - v_i^l) \sum_k \delta_k^l w_{ki} & \text{if } u_i \text{ is a hidden unit.} 
  \end{cases} \]

- When is a unit active?
  - Let $a \in [0, 0.5]$.
  - Units are active if its value $v \in [a, 1]$.
  - Units are passive if its value $v \in [0, a]$. 
Properties

- Learning rate $\eta$:
  - If $\eta$ is large, then system learns rapidly but may oscillate.
  - If $\eta$ is small, then system learns slowly but will not oscillate.
  - In the ideal case $\eta$ should be adapted during learning:

$$\Delta w_{ij}(t + 1) = \eta \delta_i(t) v_j(t) + \alpha \Delta w_{ij}(t)$$

where $\alpha$ is a constant and $\alpha \Delta w_{ij}(t)$ is called \textit{momentum term}.

- Almost all functions can be learned.
- Learning is NP-hard.
Level Mappings and Hierarchical Logic Programs

- Let $\mathcal{P}$ be a propositional logic program wrt $\mathcal{V}$.
- A level mapping for $\mathcal{P}$ is a function $l: \mathcal{R}_\mathcal{P} \rightarrow \mathbb{N}$.
  - We define $l(\neg A) = l(A)$.
- $\mathcal{P}$ is hierarchical if for all clauses $A \leftarrow L_1 \land \ldots \land L_n \in \mathcal{P}$ we find $l(A) > l(L_i)$ for all $1 \leq i \leq n$.
- **Exercise** Are the following programs hierarchical?
  - $\{p, q \leftarrow p, r \leftarrow q\}$,
  - $\{p_1 \leftarrow p_2, p_1 \leftarrow p_3 \land p_4, p_1 \leftarrow p_5 \land p_6\}$,
  - $\{p \leftarrow \neg p\}$.
- **Exercise** Show that hierarchical programs are strongly determined.
KBANN – Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1993: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ P = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a \land \neg h \}. \]
KBANN – Learning

- Given hierarchical sets of propositional rules as background knowledge.
- Map rules into multi-layer feed-forward networks with **sigmoidal** units.
- Add hidden units (optional).
- Add units for known input features that are not referenced in the rules.
- Fully connect layers.
- Add near-zero random numbers to all links and thresholds.
- Apply backpropagation.

- Empirical evaluation: system performs better than purely empirical and purely hand-built classifiers.
KBANN – A Problem

▶ “Works if rules have few conditions and there are few rules with the same head.”

\[ p_q = p_r = 9\omega \quad \text{and} \quad v_q = v_r = \frac{1}{1 + e^{-\beta(9\omega - 9.5\omega)}} \approx 0.46 \quad \text{with} \quad \beta = 1. \]

\[ p_s = 0.92\omega \quad \text{and} \quad v_s = \frac{1}{1 + e^{-\beta(0.92\omega - 0.5\omega)}} \approx 0.6 \quad \text{with} \quad \beta = 1. \]
Propositional CORE Method using Squashing Units

- d’Avila Garcez, Zaverucha, Carvalho 1997:
  Can we combine the ideas of the propositional CORE method and KBANN while avoiding the above mentioned problem?

- Here Generalization due to Bader 2009.

- Let \( u \) be a squashing unit.
  \( \Psi^- = \lim_{p \to -\infty} \Psi(p) \) and \( \Psi^+ = \lim_{p \to \infty} \Psi(p) \).
  \( a^-, a^\circ, a^+ \in \mathbb{R} \) such that \( \Psi^- < a^- < a^\circ < a^+ < \Psi^+ \).

- \( u \) is active \( \iff \) \( v \geq a^+ \).
- \( u \) is passive \( \iff \) \( v \leq a^- \).
- \( u \) is in \( \circ \)-state \( \iff \) \( v = a^\circ \).
- \( u \) is undecided \( \iff \) \( a^- < (v \neq a^\circ) < a^+ \).

- \( p^+ = \Psi^{-1}(a^+) \) is called minimal activation potential.
- \( p^- = \Psi^{-1}(a^-) \) is called maximal inactivation potential.
- \( p^\circ = \Psi^{-1}(a^\circ) \) is called \( \circ \)-state potential.
The Problem

- How can we guarantee that a unit is either active, passive or in the $\circ$-state?
- Suppose input layer units output only finitely many values.
- Let $u$ be a hidden layer unit.
- If the input layer is finite, then its potential may only take finitely many different values.
- Let $P = \{p_1, \ldots, p_n\}$ be the set of possible values for the potential of $u$.
- Let $P^+ = \{p \in P \mid p \geq p^\circ\}$ and $P^- = \{p \in P \mid p \leq p^\circ\}$.
- Let $m = \max(m^-, m^+)$, where

$$m^+ = \begin{cases} 0 & \text{if } P^+ = \emptyset \\ \frac{p^+}{\min(P^+) - p^\circ} & \text{otherwise} \end{cases}, \quad m^- = \begin{cases} 0 & \text{if } P^- = \emptyset \\ \frac{p^-}{p^\circ - \min(P^-)} & \text{otherwise} \end{cases}.$$ 

- Observations
  - If the weights on the connections to $u$ and the threshold of $u$ are multiplied with $m$, then $u$ is either active, passive or in the $\circ$-state.
  - $u$ produces only finitely many different output values.
  - The transformation can be applied to output layer units as well.
Example

Consider a bipolar sigmoidal unit $u$ with $\Psi(p) = \tanh(p)$.

Let $P = \{-0.9, -0.5, -0.3, 0.0, 0.2, 0.4, 0.8\}$ and $a^- = -0.8, a^0 = 0.0, a^+ = 0.8$.

Then, $p^- = \Psi^{-1}(a^-) \approx -1.1, p^0 = \Psi^{-1}(a^0) = 0.0, p^+ = \Psi^{-1}(a^+) \approx 1.1$.

Hence, $m^- = |\frac{-1.1}{0.0 + 0.3}| = 3.662$ and $m^+ = |\frac{1.1}{0.2 - 0.0}| = 5.493$.

Thus, $m = \max(m^-, m^+) = \max(3.662, 5.493) = 5.493$.

We obtain:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\tanh(p)$</th>
<th>$\tanh(m \times p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.9</td>
<td>-0.716</td>
<td>-0.999</td>
</tr>
<tr>
<td>-0.5</td>
<td>-0.462</td>
<td>-0.992</td>
</tr>
<tr>
<td>-0.3</td>
<td>-0.291</td>
<td>-0.929</td>
</tr>
<tr>
<td>0.0</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.2</td>
<td>0.197</td>
<td>0.800</td>
</tr>
<tr>
<td>0.4</td>
<td>0.380</td>
<td>0.976</td>
</tr>
<tr>
<td>0.8</td>
<td>0.664</td>
<td>0.999</td>
</tr>
</tbody>
</table>

Exercise Specify a core of bipolar sigmoidal units for $P = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$ with $a^- = -0.9, a^0 = 0.0, a^+ = 0.9$. 
Results

- Relation to logic programs is preserved.
- The core is trainable by backpropagation.
- Many interesting applications, e.g.:
  - DNA sequence analysis.
  - Power system fault diagnosis.
- Empirical evaluation:
  system performs better than well-known machine learning systems.
- See d’Avila Garcez, Broda, Gabbay 2002 for details.
Further Extensions

- Many-valued logic programs (see Lecture 3 & 4)
- Modal logic programs
- Answer set programming
- Metalevel priorities