COMPUTATIONAL LEARNING IN DYNAMIC LOGICS

DAY 3: UPDATES ON NEURAL NETWORKS

Nina Gierasimczuk and Caleb Schultz Kisby

@NASSLLI, June 2025

Course Homepage:

https://sites.google.com/view/nasslli25-learning-in-del

PLAN FOR TODAY

1. Overview of Neural Networks
2. A Logic for Neural Network Inference
3. Neural Network Update in Dynamic Logic

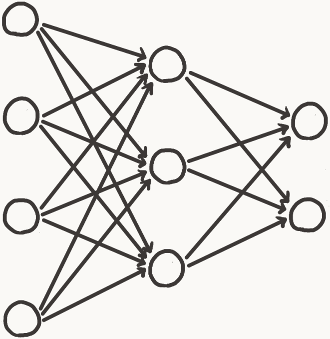
PLAN FOR TODAY

1. Overview of Neural Networks
2. A Logic for Neural Network Inference
3. Neural Network Update in Dynamic Logic

ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:

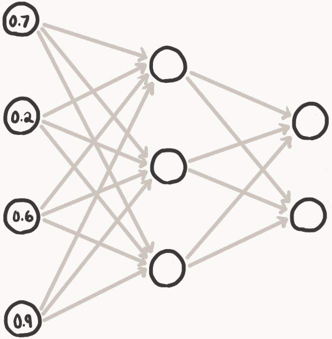
$\rightsquigarrow$



$\rightsquigarrow$

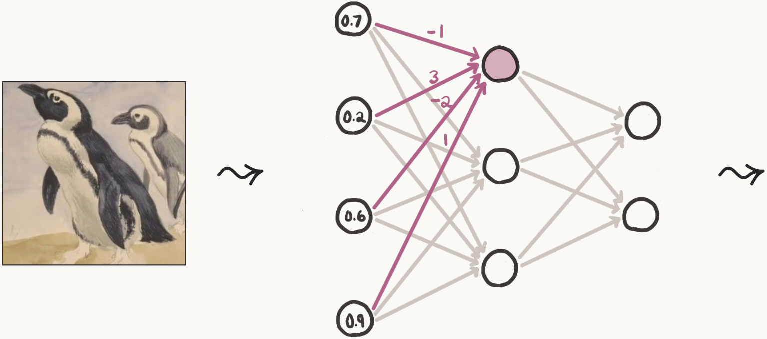
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



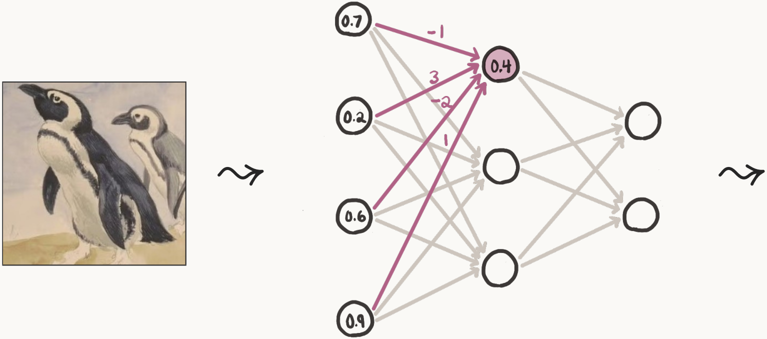
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



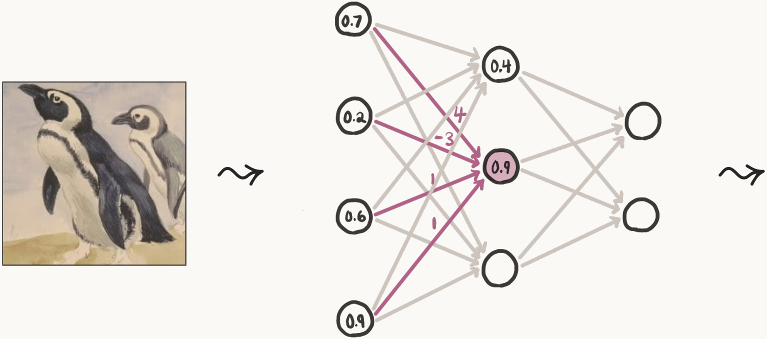
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



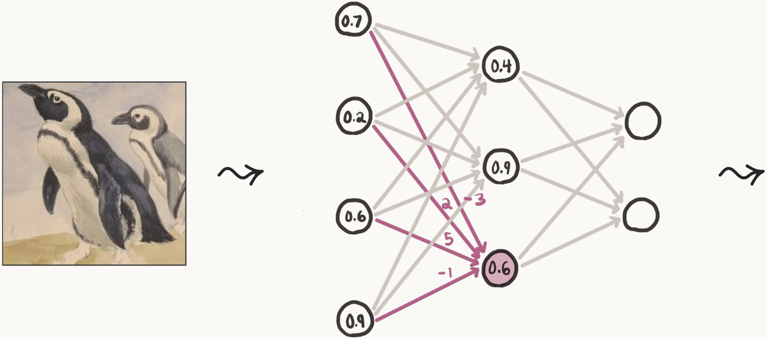
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



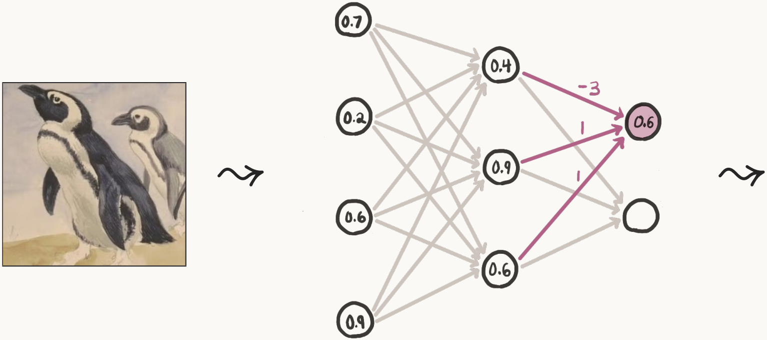
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



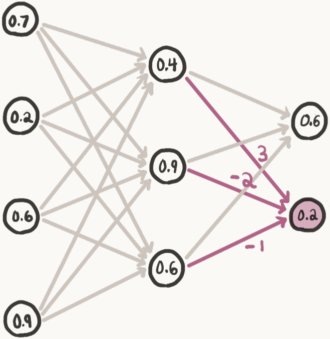
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



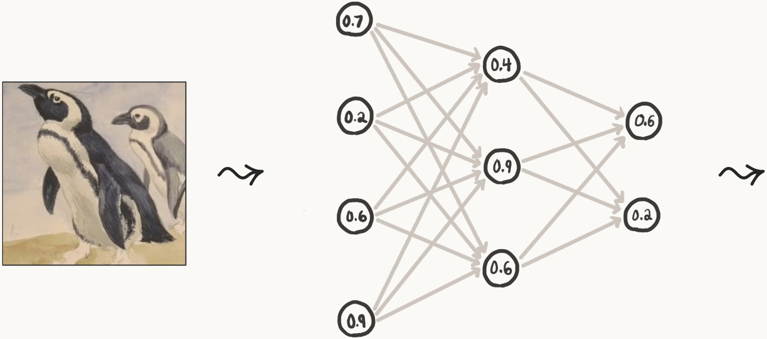
ARTIFICIAL NEURAL NETWORKS

- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



ARTIFICIAL NEURAL NETWORKS

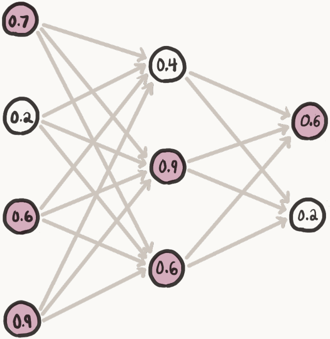
- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:



ARTIFICIAL NEURAL NETWORKS

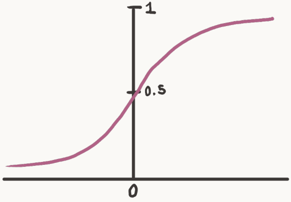
- A neural network is just $\mathcal{N} = (N, E, W, A)$
  - neurons, edges, weights, activation function
- Neurons are successively activated by their predecessors:

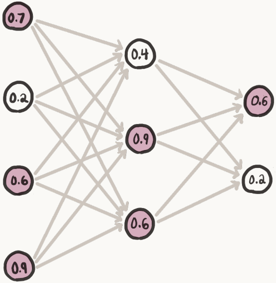
⇝



⇝ ¬FLIES

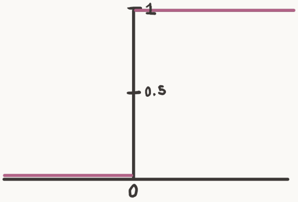
(BINARY) ARTIFICIAL NEURAL NETWORKS

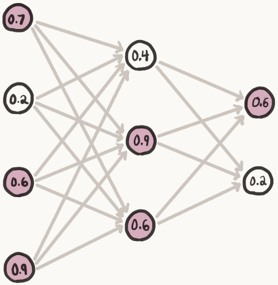




- We take the activation function *A* to be a binary step function
- This is a useful abstraction for connecting nets with logic, formal languages, and and automata
- The net's activation patterns are just **sets of neurons**.

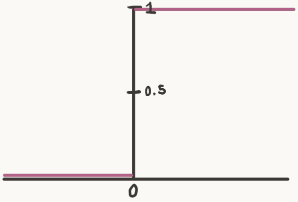
(BINARY) ARTIFICIAL NEURAL NETWORKS

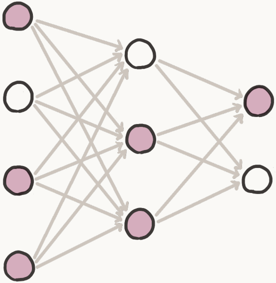




- We take the activation function *A* to be a binary step function
- This is a useful abstraction for connecting nets with logic, formal languages, and and automata
- The net's activation patterns are just **sets of neurons**.

(BINARY) ARTIFICIAL NEURAL NETWORKS





- We take the activation function *A* to be a binary step function
- This is a useful abstraction for connecting nets with logic, formal languages, and and automata
- The net's activation patterns are just **sets of neurons**.

PLAN FOR TODAY

SYNTAX: LANGUAGE OF NEURAL NETWORK INFERENCE

Definition (Language of Epistemic Logic)

Take a countable set of propositions $\textsc{Prop}$.

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi \land \varphi \mid \mathbf{A}\varphi \mid \langle \mathbf{C} \rangle \varphi$$

for all $p \in \textsc{Prop}$. The usual abbreviations are $\lor$, $\to$, and $\mathbf{C}$ (dual to $\langle \mathbf{C} \rangle$)

Definition (Language of Epistemic Logic)

Take a countable set of propositions Prop.

$$\varphi := \top \;\Big|\; p \;\Big|\; \neg\varphi \;\Big|\; \varphi \wedge \varphi \;\Big|\; \mathbf{A}\varphi \;\Big|\; \langle \mathbf{C} \rangle \varphi$$

for all $p \in$ Prop. The usual abbreviations are $\vee$, $\rightarrow$, and **C** (dual to $\langle \mathbf{C} \rangle$)

- Notice that we're giving the semantics in terms of the $\Diamond$-variant $\langle \mathbf{C} \rangle$

Definition (Language of Epistemic Logic)

Take a countable set of propositions PROP.

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{A}\varphi \mid \langle \mathbf{C} \rangle \varphi$$

for all $p \in$ PROP. The usual abbreviations are $\vee$, $\rightarrow$, and **C** (dual to $\langle \mathbf{C} \rangle$)

- Notice that we're giving the semantics in terms of the $\Diamond$-variant $\langle \mathbf{C} \rangle$
- Just like before, we will interpret $\langle \mathbf{C} \rangle \varphi$ in a model, at a world.

SYNTAX: LANGUAGE OF NEURAL NETWORK INFERENCE

Definition (Language of Epistemic Logic)

Take a countable set of propositions Prop.

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{A}\varphi \mid \langle \mathbf{C} \rangle \varphi$$

for all $p \in$ Prop. The usual abbreviations are $\vee$, $\rightarrow$, and **C** (dual to $\langle \mathbf{C} \rangle$)

- Notice that we're giving the semantics in terms of the $\Diamond$-variant $\langle \mathbf{C} \rangle$
- Just like before, we will interpret $\langle \mathbf{C} \rangle \varphi$ in a model, at a world.
- **The intended interpretation:**

  $\langle \mathbf{C} \rangle \varphi$ holds in a net, at a neuron w if w is activated by input φ.

SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- A binary neural network is just $\mathcal{N} = (N, E, W, A)$

SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- A binary neural network is just $\mathcal{N} = (N, E, W, A)$
- Each choice of E, W, A specifies a transition function from one activation pattern $S \subseteq N$ to the next

SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- A binary neural network is just $\mathcal{N} = (N, E, W, A)$
- Each choice of E, W, A specifies a transition function from one activation pattern $S \subseteq N$ to the next
- Given initial state S_0, $F_{S_0} : \wp(N) \to \wp(N)$ is given by

$$F_{S_0}(S) = S_0 \cup \{w \mid A(\sum_{u \in \text{preds}(w)} W(u, w) \cdot \chi_S(u)) = 1\}$$

"the set of all nodes w activated by their immediate predecessors u"

SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- A binary neural network is just $\mathcal{N} = (N, E, W, A)$
- Each choice of E, W, A specifies a transition function from one activation pattern $S \subseteq N$ to the next
- Given initial state S_0, $F_{S_0} : \wp(N) \to \wp(N)$ is given by

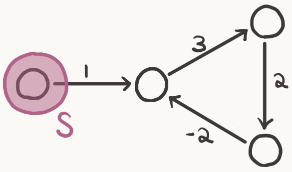
$$F_{S_0}(S) = S_0 \cup \{w \mid A(\sum_{u \in \text{preds}(w)} W(u, w) \cdot \chi_S(u)) = 1\}$$

"the set of all nodes w activated by their immediate predecessors u"

- $\chi_S(u) = 1$ iff $u \in S$ indicates whether u was activated previously

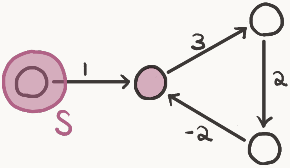
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



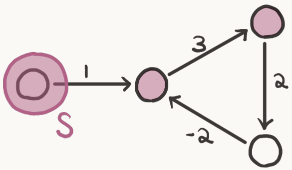
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique "answer" for each input



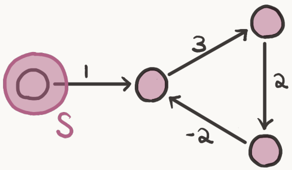
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



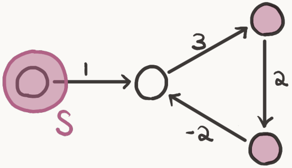
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



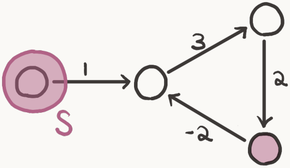
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



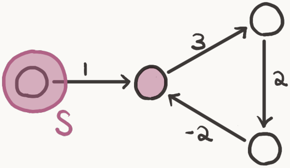
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



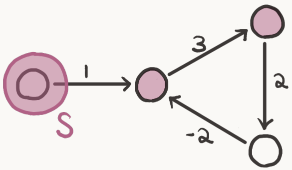
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique "answer" for each input



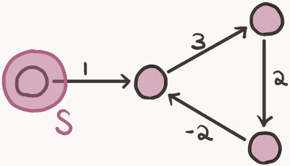
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique "answer" for each input



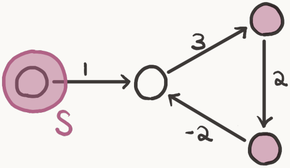
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique “answer” for each input



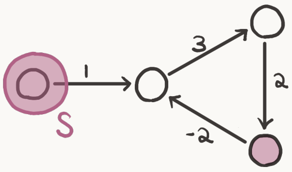
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique "answer" for each input



SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- Notice that the activated nodes could have oscillatory behavior!
- But we only want nets that have a unique "answer" for each input



SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

Postulate

We assume for all $S_0 \subseteq N$, F_{S_0} repeatedly applied to S_0,

$$S_0, F_{S_0}(S_0), F_{S_0}(F_{S_0}(S_0)), \ldots, F_{S_0}^k(S_0), \ldots$$

eventually stabilizes to a unique activation pattern.

SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

Postulate

We assume for all $S_0 \subseteq N$, F_{S_0} repeatedly applied to S_0,

$$S_0, F_{S_0}(S_0), F_{S_0}(F_{S_0}(S_0)), \ldots, F_{S_0}^k(S_0), \ldots$$

eventually stabilizes to a unique activation pattern.

Definition

Let $\mathrm{Clos} : \wp(N) \to \wp(N)$ be the function that produces this stable activation pattern.

SEMANTICS: FORMAL DEFINITION

Definition (Neural Network Semantics)

Given a binary neural network model $\mathcal{N} = (N, E, W, A, V)$, where $V : Prop \to \wp(N)$, and a neuron ("world") $w \in N$:

$$
\begin{array}{lll}
\mathcal{N}, w \models p & \text{iff} & w \in V(p) \text{ for each } p \in Prop \\
\mathcal{N}, w \models \neg\varphi & \text{iff} & \text{not } \mathcal{N}, w \models \varphi \\
\mathcal{N}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{N}, w \models \varphi \text{ and } \mathcal{N}, w \models \psi \\
\mathcal{N}, w \models A\varphi & \text{iff} & \text{for all } w \in N \text{ whatsoever, } \mathcal{N}, w \models \varphi \\
\mathcal{N}, w \models \langle \mathbf{C} \rangle \varphi & \text{iff} & w \in \text{Clos}(\llbracket \varphi \rrbracket)
\end{array}
$$

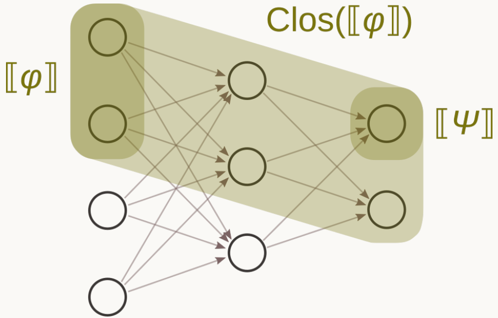
and dually:

$$
\mathcal{N}, w \models \mathbf{C}\varphi \quad \text{iff} \quad w \in \left(\text{Clos}(\llbracket \varphi \rrbracket)^{\mathsf{C}}\right)^{\mathsf{C}}
$$

where $\llbracket \varphi \rrbracket = \{u \mid \mathcal{N}, u \models \varphi\}$ is the set of worlds where φ holds (the set of neurons that are active for φ)

EXPRESSING NEURAL NETWORK INFERENCE

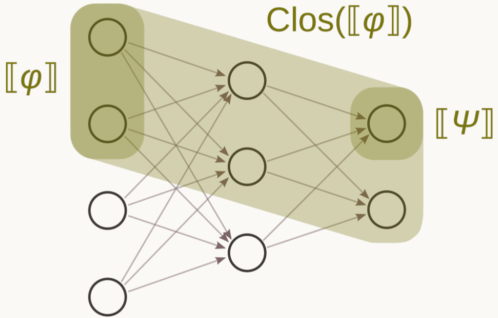
- The **C** modality gives information about the net's answer to an input



The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$ iff The net satisfies $\mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi))$

EXPRESSING NEURAL NETWORK INFERENCE

- The **C** modality gives information about the net's answer to an input

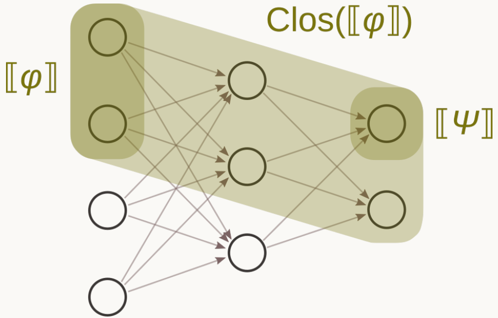


The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$ iff The net satisfies $\mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi))$

iff $\mathrm{Clos}(\llbracket \varphi \rrbracket) \supseteq \llbracket \psi \rrbracket$

EXPRESSING NEURAL NETWORK INFERENCE

- The **C** modality gives information about the net's answer to an input

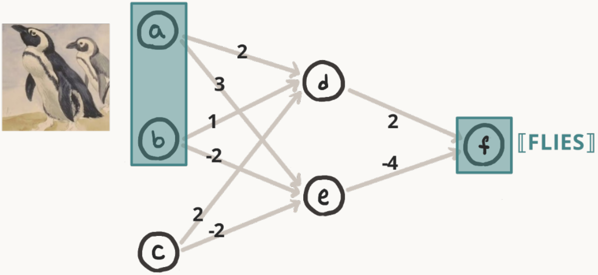


The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$ iff The net satisfies $\mathbf{A}(\psi \to \langle\mathbf{C}\rangle(\varphi))$

iff $\mathrm{Clos}(\llbracket\varphi\rrbracket) \supseteq \llbracket\psi\rrbracket$

iff **The net classifies φ as ψ**

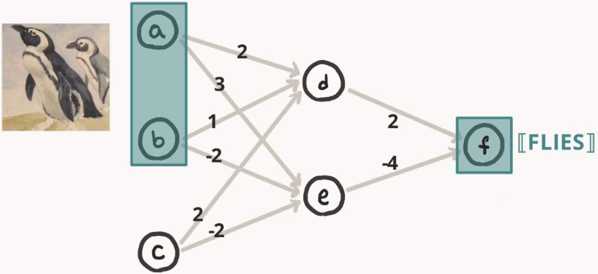
EXAMPLE: EXPRESSING NEURAL NETWORK INFERENCE



- In the exercises, we will ask you will to show

$$\mathcal{N} \not\models \mathbf{A}(\mathbf{C}(\text{PENGUIN}) \rightarrow \text{FLIES})$$

EXAMPLE: EXPRESSING NEURAL NETWORK INFERENCE

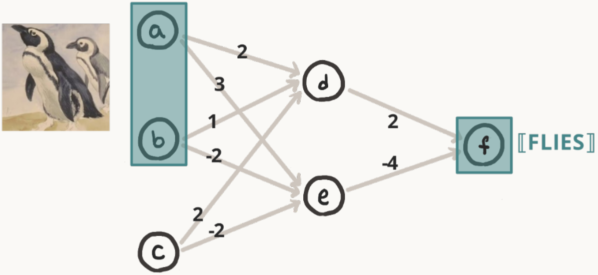


- In the exercises, we will ask you will to show

$$\mathcal{N} \not\models \mathbf{A}(\mathbf{C}(\textsc{penguin}) \to \textsc{flies})$$

- This means the net does not classify penguins as flying

EXAMPLE: EXPRESSING NEURAL NETWORK INFERENCE



- In the exercises, we will ask you will to show

$$\mathcal{N} \not\models \mathbf{A}(\mathbf{C}(\textsc{penguin}) \rightarrow \textsc{flies})$$

- This means the net does not classify penguins as flying
- Yet, if we take $[\![\textsc{bird}]\!] = \{a, b, c\}$,

$$\mathcal{N} \models \mathbf{A}(\mathbf{C}(\textsc{bird}) \rightarrow \textsc{flies})$$

ADDITIONAL COMMENTS ON THIS LOGIC

$$
\begin{aligned}
\text{The net satisfies } \mathbf{A}(\mathbf{C}(\varphi) \to \psi) \quad &\text{iff} \quad \text{The net satisfies } \mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi)) \\
&\text{iff} \quad \mathrm{Clos}(\llbracket \varphi \rrbracket) \supseteq \llbracket \psi \rrbracket \\
&\text{iff} \quad \textbf{The net classifies } \varphi \textbf{ as } \psi
\end{aligned}
$$

- What does this remind you of?

ADDITIONAL COMMENTS ON THIS LOGIC

$$\begin{aligned} \text{The net satisfies } \mathbf{A}(\mathbf{C}(\varphi) \to \psi) \quad &\text{iff} \quad \text{The net satisfies } \mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi)) \\ &\text{iff} \quad \mathrm{Clos}(\llbracket \varphi \rrbracket) \supseteq \llbracket \psi \rrbracket \\ &\text{iff} \quad \textbf{The net classifies } \varphi \textbf{ as } \psi \end{aligned}$$

- What does this remind you of?
  - $\mathrm{best}_{\preceq}(\llbracket \varphi \rrbracket) \subseteq \llbracket \psi \rrbracket$

ADDITIONAL COMMENTS ON THIS LOGIC

The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$ iff The net satisfies $\mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi))$

iff $\text{Clos}(\llbracket \varphi \rrbracket) \supseteq \llbracket \psi \rrbracket$

iff **The net classifies** φ **as** ψ

- What does this remind you of?
  - $\text{best}_{\preceq}(\llbracket \varphi \rrbracket) \subseteq \llbracket \psi \rrbracket$
  - $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$, taken as a conditional, behaves exactly like $B^{\varphi}\psi$

ADDITIONAL COMMENTS ON THIS LOGIC

The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$	iff	The net satisfies $\mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi))$
	iff	$\mathrm{Clos}(\llbracket\varphi\rrbracket) \supseteq \llbracket\psi\rrbracket$
	iff	**The net classifies** φ **as** ψ

- What does this remind you of?
  - $\mathrm{best}_{\preceq}(\llbracket\varphi\rrbracket) \subseteq \llbracket\psi\rrbracket$
  - $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$, taken as a conditional, behaves exactly like $B^{\varphi}\psi$
  - You can think of this as the net's conditional belief

ADDITIONAL COMMENTS ON THIS LOGIC

The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$	iff	The net satisfies $\mathbf{A}(\psi \to \langle \mathbf{C} \rangle(\varphi))$
	iff	$\mathrm{Clos}([\![\varphi]\!]) \supseteq [\![\psi]\!]$
	iff	**The net classifies** φ **as** ψ

- What does this remind you of?
  - $\mathrm{best}_{\preceq}([\![\varphi]\!]) \subseteq [\![\psi]\!]$
  - $\mathbf{A}(\mathbf{C}(\varphi) \to \psi)$, taken as a conditional, behaves exactly like $B^{\varphi}\psi$
  - You can think of this as the net's conditional belief
- Interpreting **C** on its own is less clear...

PLAN FOR TODAY

1. Overview of Neural Networks
2. A Logic for Neural Network Inference
3. Neural Network Update in Dynamic Logic

UPDATES ON NEURAL NETWORKS

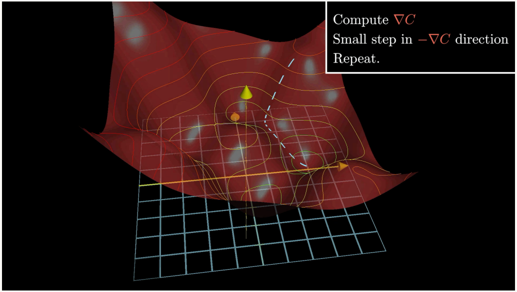
- **Unsupervised Updates**
  - The network learns from data that is **unlabeled** (no expected answer or classification)
  - Each update softly increases the net's preference for the input
  - Hebb's rule, Oja's rule, & competitive learning rule
- **Supervised Updates**
  - The network learns from **labeled** data with an **expected answer**
  - Each update softly increases the net's accuracy on a function
  - Backpropagation rule & delta learning rule

UPDATES ON NEURAL NETWORKS

- **Unsupervised Updates**
  - The network learns from data that is **unlabeled** (no expected answer or classification)
  - Each update softly increases the net's preference for the input
  - **Hebb's rule**, Oja's rule, & competitive learning rule
- **Supervised Updates**
  - The network learns from **labeled** data with an **expected answer**
  - Each update softly increases the net's accuracy on a function
  - **Backpropagation rule** & delta learning rule

BACKPROPAGATION RULE

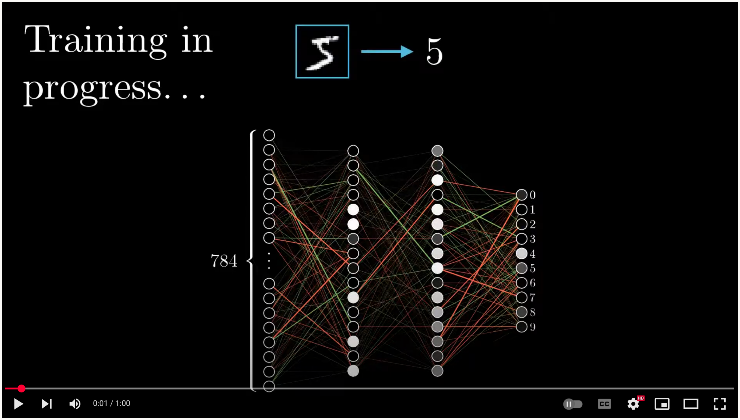
- Backpropagation is the most widely used neural network update rule
- **Main idea:** Backprop implements **gradient descent** on a net's weights



- Given an input $\vec{x}$ with label y, the neural network gives its answer y' to $\vec{x}$, and each weight of the net is adjusted according to its contribution to the error (difference between y' and y).

BACKPROPAGATION RULE

https://www.youtube.com/watch?v=cANqroNVdl8



ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

- What if we could have a complete characterization of Backprop, like we did for public announcement, Lex, and Mini?
  - That would be wonderful!

ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

- What if we could have a complete characterization of Backprop, like we did for public announcement, LEX, and MINI?
  - That would be wonderful!
  - Unfortunately, this is still an open problem

ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

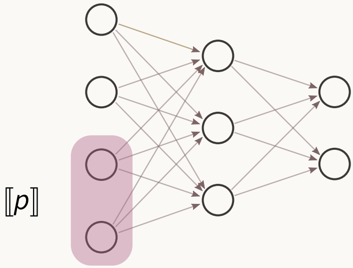
- What if we could have a complete characterization of Backprop, like we did for public announcement, Lex, and Mini?
  - That would be wonderful!
  - Unfortunately, this is still an open problem
- **Proof of concept:** Can we do this for any neural network update at all?

ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

- What if we could have a complete characterization of Backprop, like we did for public announcement, LEX, and MINI?
  - That would be wonderful!
  - Unfortunately, this is still an open problem
- **Proof of concept:** Can we do this for any neural network update at all?
  - Let's consider the simplest possible one: **Hebbian learning**

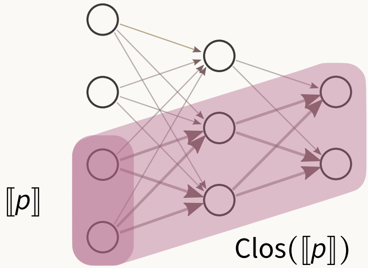
HEBBIAN UPDATE RULE

Neurons that fire together wire together



HEBBIAN UPDATE RULE

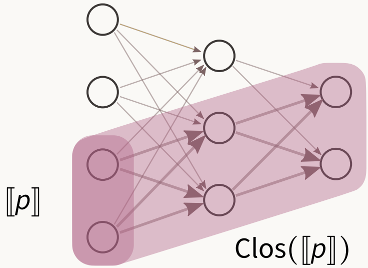
Neurons that fire together wire together



- Each edge involved in the activation is “bumped up” by a fixed learning rate $\eta \geq 0$

HEBBIAN UPDATE RULE

Neurons that fire together wire together



- Each edge involved in the activation is "bumped up" by a fixed learning rate $\eta \geq 0$
- Formally: $\textsc{Hebb}(\mathcal{N}, [\![\varphi]\!]) = (N, E, W', A)$, where
$$W'(u,w) = W(u,w) + \eta \cdot \chi_{\text{Clos}([\![\varphi]\!])}(u) \cdot \chi_{\text{Clos}([\![\varphi]\!])}(w)$$

HEBBIAN UPDATE RULE: EXAMPLES

- If nobody ever told you that penguins don’t fly, how could you come to believe they don’t?

HEBBIAN UPDATE RULE: EXAMPLES

- If nobody ever told you that penguins don't fly, how could you come to believe they don't?
  - Observe animals with similar features that **don't** fly

HEBBIAN UPDATE RULE: EXAMPLES

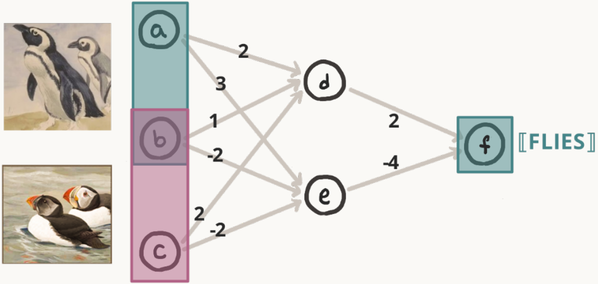
- If nobody ever told you that penguins don’t fly, how could you come to believe they don’t?
  - Observe animals with similar features that **don’t** fly
- Now imagine you believe penguins don’t fly. What could cause you to change your mind?

HEBBIAN UPDATE RULE: EXAMPLES

- If nobody ever told you that penguins don’t fly, how could you come to believe they don’t?
  - Observe animals with similar features that **don’t** fly
- Now imagine you believe penguins don’t fly. What could cause you to change your mind?
  - Observe animals with similar features that **do** fly

HEBBIAN UPDATE RULE: EXAMPLES

- If nobody ever told you that penguins don’t fly, how could you come to believe they don’t?
  - Observe animals with similar features that **don’t** fly
- Now imagine you believe penguins don’t fly. What could cause you to change your mind?
  - Observe animals with similar features that **do** fly



ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate

ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar

ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar
  - **Key idea:** All weights must sum to 1
  - "use it or lose it"

ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar
  - **Key idea:** All weights must sum to 1
  - "use it or lose it"
  - It's an open problem to completely characterize stable Hebbian update rules

ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar
  - **Key idea:** All weights must sum to 1
  - "use it or lose it"
  - It's an open problem to completely characterize stable Hebbian update rules
- HEBB is more gradual than LEX or MINI

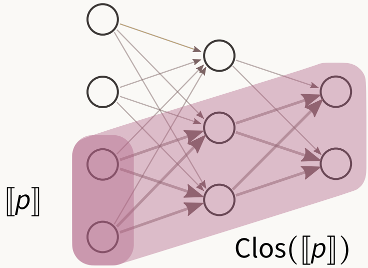
ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar
  - **Key idea:** All weights must sum to 1
  - "use it or lose it"
  - It's an open problem to completely characterize stable Hebbian update rules
- HEBB is more gradual than LEX or MINI
  - The result of LEX and MINI is a change in belief

ADDITIONAL COMMENTS ABOUT HEBBIAN UPDATE

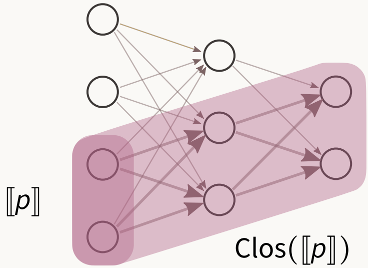
- This kind of Hebbian update is **unstable** — weights will continue to increase until they saturate
  - It's possible to prevent this by using **Oja's rule** or similar
  - **Key idea:** All weights must sum to 1
  - "use it or lose it"
  - It's an open problem to completely characterize stable Hebbian update rules
- HEBB is more gradual than LEX or MINI
  - The result of LEX and MINI is a change in belief
  - HEBB gently nudges us in the direction of a belief

HEBB*: "FIXED-POINT" HEBBIAN UPDATE



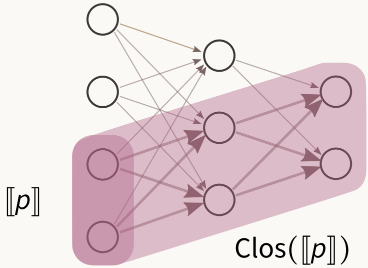
- If we repeatedly apply $\textsc{Hebb}(\mathcal{N}, S)$, eventually these weights will saturate (they will not inhibit any incoming activations)

HEBB*: "FIXED-POINT" HEBBIAN UPDATE



- If we repeatedly apply $\textsc{Hebb}(\mathcal{N}, S)$, eventually these weights will saturate (they will not inhibit any incoming activations)
- Let iter be the number of iterations needed to reach this fixed point

Hebb*: "Fixed-point" Hebbian update



- If we repeatedly apply $\textsc{Hebb}(\mathcal{N}, S)$, eventually these weights will saturate (they will not inhibit any incoming activations)
- Let iter be the number of iterations needed to reach this fixed point
- Let $\textsc{Hebb}^*(\mathcal{N}, S) = (N, E, W', A)$, where

$$W'(u,w) = W(u,w) + \text{iter} \cdot \eta \cdot \chi_{\text{Clos}(\llbracket \varphi \rrbracket)}(u) \cdot \chi_{\text{Clos}(\llbracket \varphi \rrbracket)}(w)$$

NEURAL NETWORK UPDATES IN DYNAMIC LOGIC

- We can use the DEL trick to give semantics using neural network updates

Definition (Neural Network Semantics)
Let $\mathcal{N}$ be a binary neural network model, $w \in N$, and let $\mathcal{U} : \mathbf{Net} \to \mathcal{L} \to \mathbf{Net}$ be any unsupervised update:

$$\mathcal{N}, w \models [\varphi]\psi \quad \text{iff} \quad \text{Update}(N, [\![\varphi]\!]), w \models \psi$$

For Hebbian updates in particular:

$$\begin{aligned} \mathcal{N}, w \models [\varphi]_{\textsc{Hebb}}\psi \quad & \text{iff} \quad \textsc{Hebb}(\mathcal{N}, [\![\varphi]\!]), w \models \psi \\ \mathcal{N}, w \models [\varphi]_{\textsc{Hebb}^*}\psi \quad & \text{iff} \quad \textsc{Hebb}^*(\mathcal{N}, [\![\varphi]\!]), w \models \psi \end{aligned}$$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\textsc{Hebb}^*} p \leftrightarrow p$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\textsc{Hebb}^*} p \leftrightarrow p$

Hebb*-Negation $[\varphi]_{\textsc{Hebb}^*} \neg\psi \leftrightarrow \neg[\varphi]_{\textsc{Hebb}^*}\psi$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\text{HEBB}^*} p \leftrightarrow p$

Hebb*-Negation $[\varphi]_{\text{HEBB}^*} \neg\psi \leftrightarrow \neg[\varphi]_{\text{HEBB}^*}\psi$

Hebb*-Conjunction $[\varphi]_{\text{HEBB}^*}(\psi \wedge \theta) \leftrightarrow [\varphi]_{\text{HEBB}^*}\psi \wedge [\varphi]_{\text{HEBB}^*}\theta$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\textsc{Hebb}^*} p \leftrightarrow p$

Hebb*-Negation $[\varphi]_{\textsc{Hebb}^*} \neg\psi \leftrightarrow \neg[\varphi]_{\textsc{Hebb}^*}\psi$

Hebb*-Conjunction $[\varphi]_{\textsc{Hebb}^*}(\psi \wedge \theta) \leftrightarrow [\varphi]_{\textsc{Hebb}^*}\psi \wedge [\varphi]_{\textsc{Hebb}^*}\theta$

Hebb*-Diamond $[\varphi]_{\textsc{Hebb}^*} \Diamond \psi \leftrightarrow \Diamond[\varphi]_{\textsc{Hebb}^*}\psi$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\textsc{Hebb}^*} p \leftrightarrow p$

Hebb*-Negation $[\varphi]_{\textsc{Hebb}^*} \neg\psi \leftrightarrow \neg[\varphi]_{\textsc{Hebb}^*}\psi$

Hebb*-Conjunction $[\varphi]_{\textsc{Hebb}^*}(\psi \wedge \theta) \leftrightarrow [\varphi]_{\textsc{Hebb}^*}\psi \wedge [\varphi]_{\textsc{Hebb}^*}\theta$

Hebb*-Diamond $[\varphi]_{\textsc{Hebb}^*} \Diamond \psi \leftrightarrow \Diamond[\varphi]_{\textsc{Hebb}^*}\psi$

Hebb*-Closure $[\varphi]_{\textsc{Hebb}^*}\langle \mathbf{C} \rangle\psi \leftrightarrow$

$$\langle \mathbf{C} \rangle([\varphi]_{\textsc{Hebb}^*}\psi \vee (\langle \mathbf{C} \rangle\varphi \wedge \Diamond(\langle \mathbf{C} \rangle\varphi \wedge \langle \mathbf{C} \rangle[\varphi]_{\textsc{Hebb}^*}\psi)))$$

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

Hebb*-Propositions $[\varphi]_{\textsc{Hebb}^*} p \leftrightarrow p$

Hebb*-Negation $[\varphi]_{\textsc{Hebb}^*} \neg\psi \leftrightarrow \neg[\varphi]_{\textsc{Hebb}^*}\psi$

Hebb*-Conjunction $[\varphi]_{\textsc{Hebb}^*}(\psi \wedge \theta) \leftrightarrow [\varphi]_{\textsc{Hebb}^*}\psi \wedge [\varphi]_{\textsc{Hebb}^*}\theta$

Hebb*-Diamond $[\varphi]_{\textsc{Hebb}^*} \Diamond \psi \leftrightarrow \Diamond[\varphi]_{\textsc{Hebb}^*}\psi$

Hebb*-Closure $[\varphi]_{\textsc{Hebb}^*}\langle \mathbf{C} \rangle\psi \leftrightarrow$

$$\langle \mathbf{C} \rangle([\varphi]_{\textsc{Hebb}^*}\psi \vee (\langle \mathbf{C} \rangle\varphi \wedge \Diamond(\langle \mathbf{C} \rangle\varphi \wedge \langle \mathbf{C} \rangle[\varphi]_{\textsc{Hebb}^*}\psi)))$$

Note: We didn't define $\Diamond$ for neural networks; here are its semantics:

$$\mathcal{N}, w \models \Diamond\varphi \quad \text{iff} \quad \text{there is an } E\text{-path from some } u \in [\![\varphi]\!] \text{ to } w.$$

REDUCTION LAWS FOR HEBB*: INTUITION

Let's look at this last law:

Hebb*-Closure. $[\varphi]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle\psi \leftrightarrow$

$$\langle \mathbf{C}\rangle([\varphi]_{\mathsf{Hebb}^\star}\psi \vee (\langle \mathbf{C}\rangle\varphi \wedge \diamond(\langle \mathbf{C}\rangle\varphi \wedge \langle \mathbf{C}\rangle[\varphi]_{\mathsf{Hebb}^\star}\psi)))$$

What does it mean? Why does it hold?

REDUCTION LAWS FOR HEBB*: INTUITION

Let's look at this last law:

Hebb*-Closure. $[\varphi]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle\psi \leftrightarrow$

$$\langle \mathbf{C}\rangle([\varphi]_{\mathsf{Hebb}^\star}\psi \vee (\langle \mathbf{C}\rangle\varphi \wedge \diamond(\langle \mathbf{C}\rangle\varphi \wedge \langle \mathbf{C}\rangle[\varphi]_{\mathsf{Hebb}^\star}\psi)))$$

What does it mean? Why does it hold?

Again, look at what this means for the propositional case:

REDUCTION LAWS FOR HEBB*: INTUITION

Let's look at this last law:

Hebb*-Closure. $[\varphi]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle\psi \leftrightarrow$

$$\langle \mathbf{C}\rangle([\varphi]_{\mathsf{Hebb}^\star}\psi \vee (\langle \mathbf{C}\rangle\varphi \wedge \diamond(\langle \mathbf{C}\rangle\varphi \wedge \langle \mathbf{C}\rangle[\varphi]_{\mathsf{Hebb}^\star}\psi)))$$

What does it mean? Why does it hold?

Again, look at what this means for the propositional case:

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle(q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle(q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle (q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect $\mathsf{Hebb}^\star$ has on the closure Clos. This is the description:

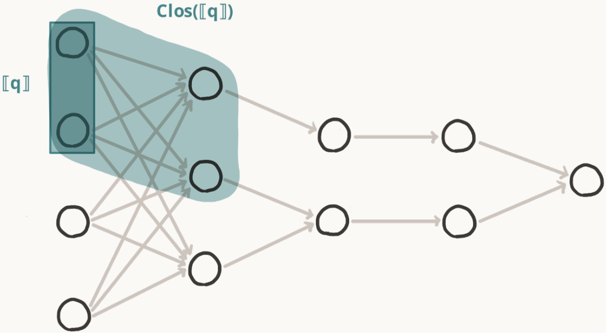
$$\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N}, [\![p]\!])}([\![q]\!]) = \mathsf{Clos}([\![q]\!] \cup (\mathsf{Clos}([\![p]\!]) \cap \mathsf{Reach}(\mathsf{Clos}([\![p]\!]) \cap \mathsf{Clos}([\![q]\!]))))$$

REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle (q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect Hebb* has on the closure Clos. This is the description:

$$\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},[\![p]\!])}([\![q]\!]) = \mathsf{Clos}([\![q]\!] \cup (\mathsf{Clos}([\![p]\!]) \cap \mathsf{Reach}(\mathsf{Clos}([\![p]\!]) \cap \mathsf{Clos}([\![q]\!]))))$$

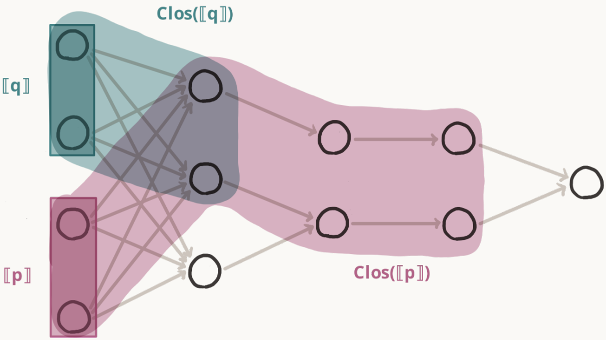


REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle (q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect $\mathsf{Hebb}^\star$ has on the closure Clos. This is the description:

$$\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},[\![p]\!])}([\![q]\!]) = \mathsf{Clos}([\![q]\!] \cup (\mathsf{Clos}([\![p]\!]) \cap \mathsf{Reach}(\mathsf{Clos}([\![p]\!]) \cap \mathsf{Clos}([\![q]\!]))))$$

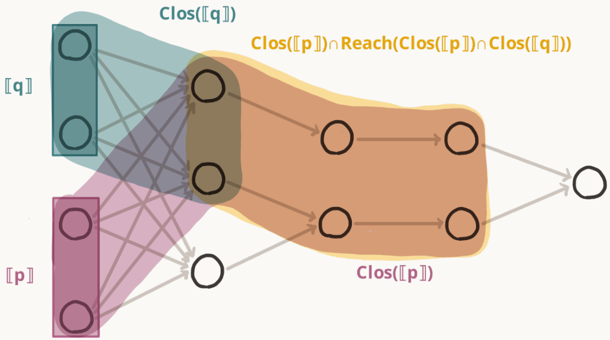


REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle (q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect $\mathsf{Hebb}^\star$ has on the closure Clos. This is the description:

$$\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},[\![p]\!])}([\![q]\!]) = \mathsf{Clos}([\![q]\!] \cup (\mathsf{Clos}([\![p]\!]) \cap \mathsf{Reach}(\mathsf{Clos}([\![p]\!]) \cap \mathsf{Clos}([\![q]\!]))))$$

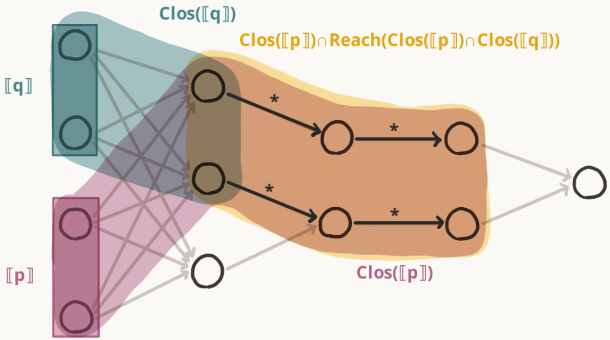


REDUCTION LAWS FOR HEBB*: INTUITION

$$[p]_{\text{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle (q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect $\text{Hebb}^\star$ has on the closure Clos. This is the description:

$$\text{Clos}_{\text{Hebb}^\star(\mathcal{N},[\![p]\!])}([\![q]\!]) = \text{Clos}([\![q]\!] \cup (\text{Clos}([\![p]\!]) \cap \text{Reach}(\text{Clos}([\![p]\!]) \cap \text{Clos}([\![q]\!]))))$$

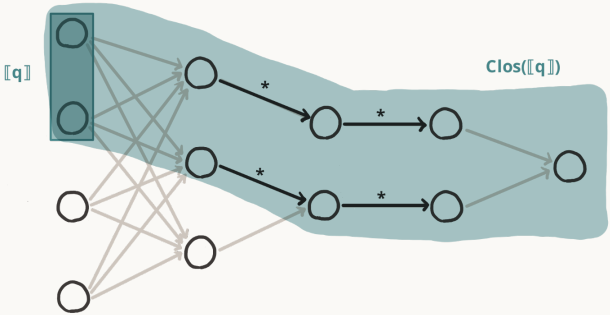


REDUCTION LAWS FOR HEBB*: INTUITION

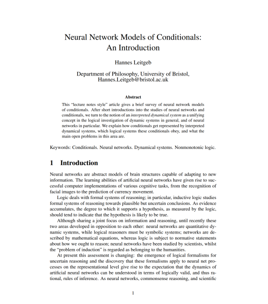
$$[p]_{\mathsf{Hebb}^\star}\langle \mathbf{C}\rangle q \leftrightarrow \langle \mathbf{C}\rangle(q \vee (\langle \mathbf{C}\rangle p \wedge \diamond(\langle \mathbf{C}\rangle p \wedge \langle \mathbf{C}\rangle q)))$$

- This encodes in our logic a complete description of the effect $\mathsf{Hebb}^\star$ has on the closure Clos. This is the description:

$$\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},[\![p]\!])}([\![q]\!]) = \mathsf{Clos}([\![q]\!] \cup (\mathsf{Clos}([\![p]\!]) \cap \mathsf{Reach}(\mathsf{Clos}([\![p]\!]) \cap \mathsf{Clos}([\![q]\!]))))$$



TO LEARN MORE CHECK OUT...



Neural Network Models of Conditionals:
An Introduction

Hannes Leitgeb

Department of Philosophy, University of Bristol,
Hannes.Leitgeb@bristol.ac.uk

Abstract

This "lecture notes style" article gives a brief survey of neural network models of conditionals. After short introductions into the studies of neural networks and conditionals, we turn to the notion of an *interpreted dynamical system* as a unifying concept in the logical investigation of dynamic systems in general, and of neural networks in particular. We explain how conditionals get represented by interpreted dynamical systems, which logical systems these conditionals obey, and what the main open problems in this area are.

Keywords: Conditionals. Neural networks. Dynamical systems. Nonmonotonic logic.

1 Introduction

Neural networks are abstract models of brain structures capable of adapting to new information. The learning abilities of artificial neural networks have given rise to successful computer implementations of various cognitive tasks, from the recognition of facial images to the prediction of currency movement.

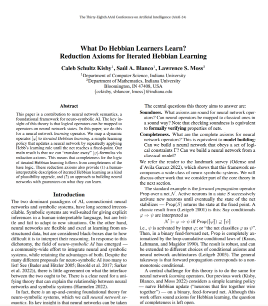
Logic deals with formal systems of reasoning; in particular, inductive logic studies formal systems of reasoning towards plausible but uncertain conclusions. As evidence accumulates, the degree to which it supports a hypothesis, as measured by the logic, should tend to indicate that the hypothesis is likely to be true.

Although sharing a joint focus on information and reasoning, until recently these two areas developed in opposition to each other: neural networks are quantitative dynamic systems, while logical reasoners must be symbolic systems; networks are described by mathematical equations, whereas logic is subject to normative statements about how we ought to reason; neural networks have been studied by scientists, whilst the "problem of induction" is regarded as belonging to the humanities.

At present this assessment is changing: the emergence of logical formalisms for uncertain reasoning and the discovery that these formalisms apply to neural net processes on the representational level give rise to the expectation that the dynamics of artificial neural networks can be understood in terms of logically valid, and thus rational, rules of inference. As neural networks, commonsense reasoning, and scientific

1

(a) **Neural Network Models of Conditionals: An Introduction** by Leitgeb



The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

**What Do Hebbian Learners Learn?
Reduction Axioms for Iterated Hebbian Learning**

Caleb Schultz Kisby[1], Saúl A. Blanco[1], Lawrence S. Moss[2]
[1]Department of Computer Science, Indiana University
[2]Department of Mathematics, Indiana University
Bloomington, IN 47408, USA
{cckisby, sblancor, lmoss}@indiana.edu

Abstract

This paper is a contribution to neural network semantics, a foundational framework for neuro-symbolic AI. The key insight of this theory is that logical operators can be mapped to operators on neural network states. In this paper, we do this for a neural network *learning* operator. We map a dynamic operator $[\varphi]$ to *iterated Hebbian learning*, a simple learning policy that updates a neural network by repeatedly applying Hebb's learning rule until the net reaches a fixed-point. Our main result is that we can "translate away" $[\varphi]$-formulas via reduction axioms. This means that completeness for the logic of iterated Hebbian learning follows from completeness of the base logic. These reduction axioms also provide (1) a human-interpretable description of iterated Hebbian learning as a kind of plausibility upgrade, and (2) an approach to building neural networks with guarantees on what they can learn.

Introduction

The two dominant paradigms of AI, connectionist neural networks and symbolic systems, have long seemed irreconcilable. Symbolic systems are well-suited for giving explicit inferences in a human-interpretable language, but are brittle and fail to adapt to new situations. On the other hand, neural networks are flexible and excel at learning from unstructured data, but are considered black-boxes due to how difficult it is to interpret their reasoning. In response to this dichotomy, the field of *neuro-symbolic AI* has emerged — a community-wide effort to integrate neural and symbolic systems, while retaining the advantages of both. Despite the many different proposals for neuro-symbolic AI (too many to list! See (Bader and Hitzler 2005; Besold et al. 2017; Sarker et al. 2022)), there is little agreement on what the interface between the two ought to be. There is a clear need for a unifying theory that can explain the relationship between neural networks and symbolic systems (Harmelen 2022).

In fact, there *is* an up-and-coming foundational theory for neuro-symbolic systems, which we call *neural network semantics*. Its key insight is that neural networks can be taken

The central questions this theory aims to answer are:

Soundness. What axioms are sound for neural network operators? Can neural operators be mapped to classical ones in a sound way? Note that checking soundness is equivalent to **formally verifying** properties of nets.

Completeness. What are the complete axioms for neural network operators? This is equivalent to **model building**: Can we build a neural network that obeys a set of logical constraints Γ? Can we build a neural network from a classical model?

We refer the reader to the landmark survey (Odense and d'Avila Garcez 2022), which shows that this framework encompasses a wide class of neuro-symbolic systems. We will discuss other work that we consider part of the core theory in the next section.

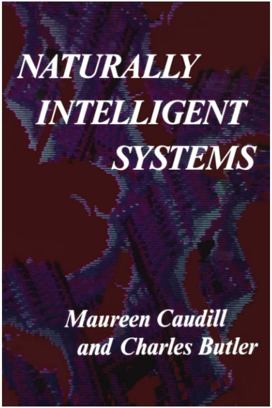
The standard example is the *forward propagation* operator Prop over a net $\mathcal{N}$. Active neurons in a state S successively activate new neurons until eventually the state of the net stabilizes — $\mathsf{Prop}(S)$ returns the state at the fixed point. A classic result from (Leitgeb 2001) is this: Say conditionals $\varphi \Rightarrow \psi$ are interpreted as

$$\mathcal{N} \models \varphi \Rightarrow \psi \text{ iff } \mathsf{Prop}([\varphi]) \supseteq [\psi]$$

i.e., ψ is activated by input φ; or "the net classifies φ as ψ". Then, in a binary feed-forward net, Prop is completely axiomatized by the loop-cumulative conditional laws of (Kraus, Lehmann, and Magidor 1990). The result is robust, and can be extended to different choices of conditional axioms and neural network architectures (Leitgeb 2003). The general takeaway is that forward propagation corresponds to a non-monotonic conditional.

A central challenge for this theory is to do the same for neural network *learning* operators. Our previous work (Kisby, Blanco, and Moss 2022) considers a simple learning policy — naïve Hebbian update ("neurons that fire together wire together") — on a binary, feed-forward net. Although this work offers sound axioms for Hebbian learning, the question of completeness is left open.

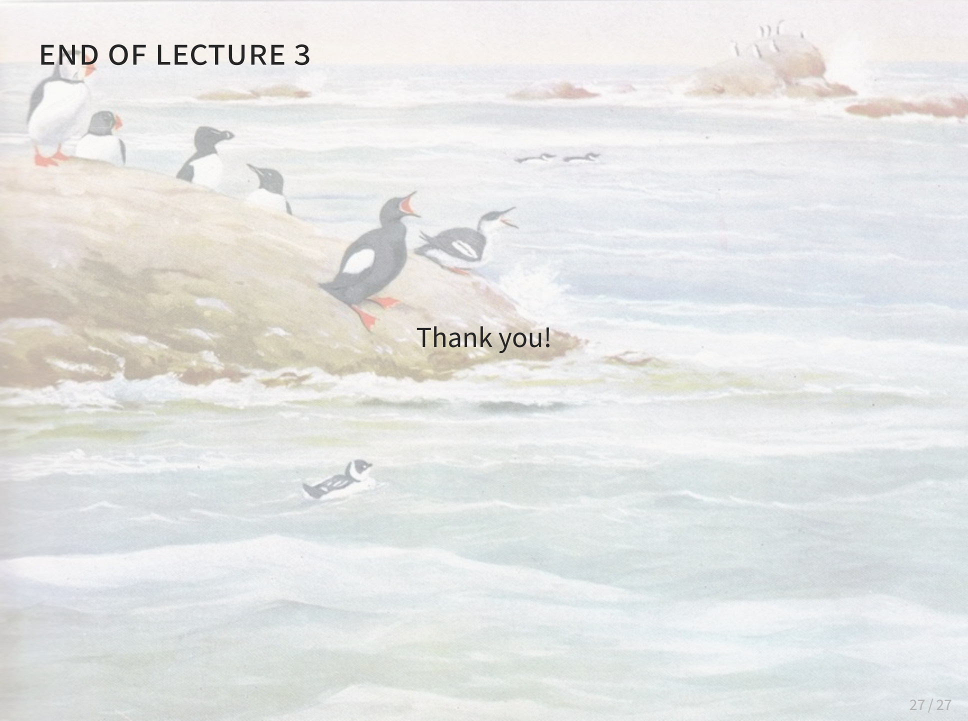
(b) **Reduction Axioms for Iterated Hebbian Learning** by Schultz Kisby, Blanco, & Moss



(c) **Naturally Intelligent Systems** by Caudill & Butler

Banner images are Penguins (1895) and Black Guillemot, Puffin, … (1932) by Louis Agassiz Fuertes. Watercolor. Backpropagation

END OF LECTURE 3



Thank you!