

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

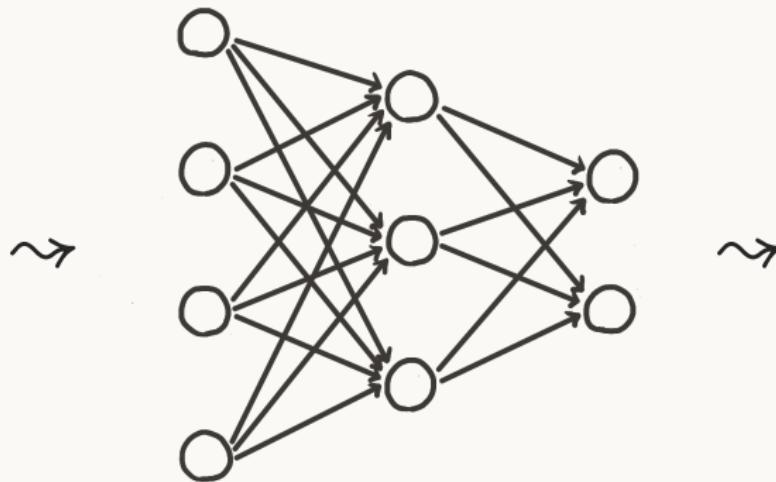
- 1 Overview of Neural Networks
- 2 A Logic for Neural Network Inference
- 3 Neural Network Update in Dynamic Logic

PLAN FOR TODAY

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

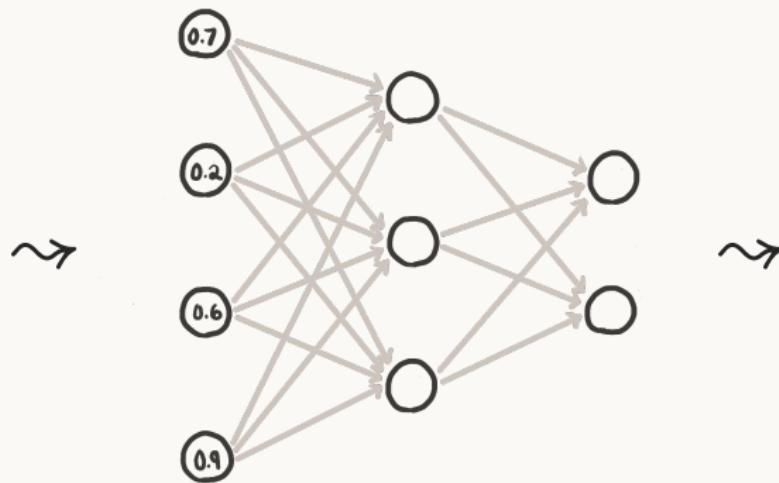
ARTIFICIAL NEURAL NETWORKS

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



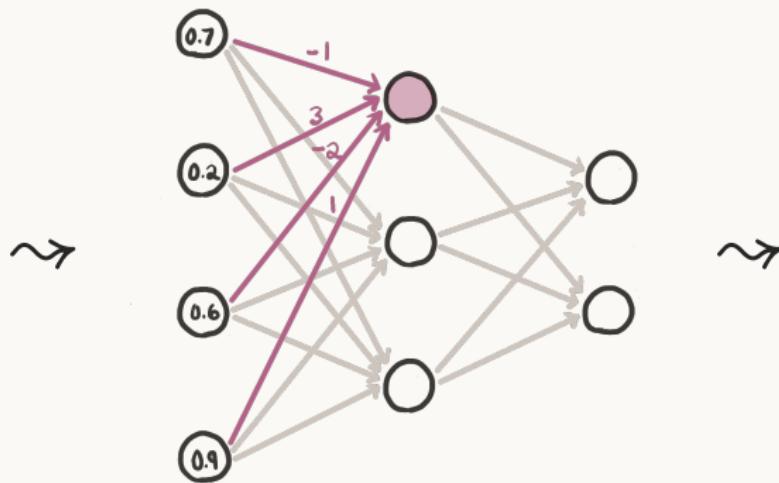
ARTIFICIAL NEURAL NETWORKS

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



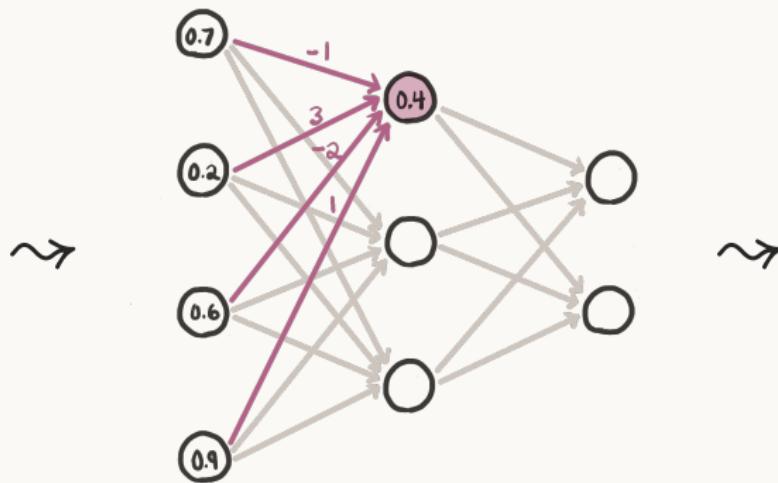
ARTIFICIAL NEURAL NETWORKS

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



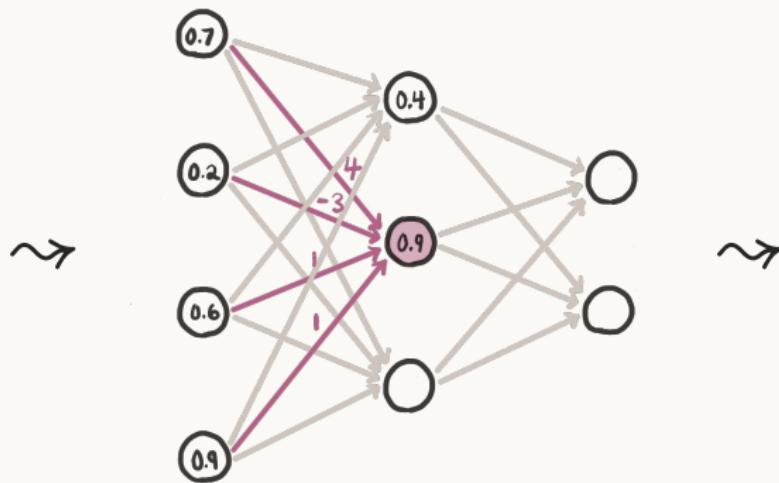
ARTIFICIAL NEURAL NETWORKS

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



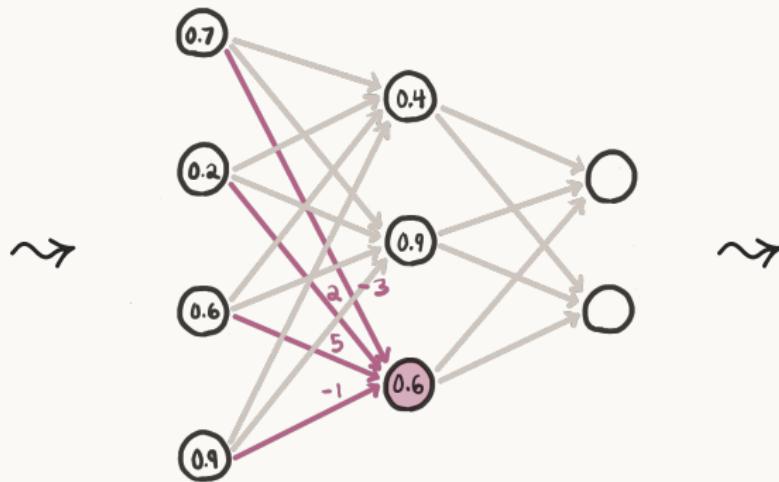
ARTIFICIAL NEURAL NETWORKS

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



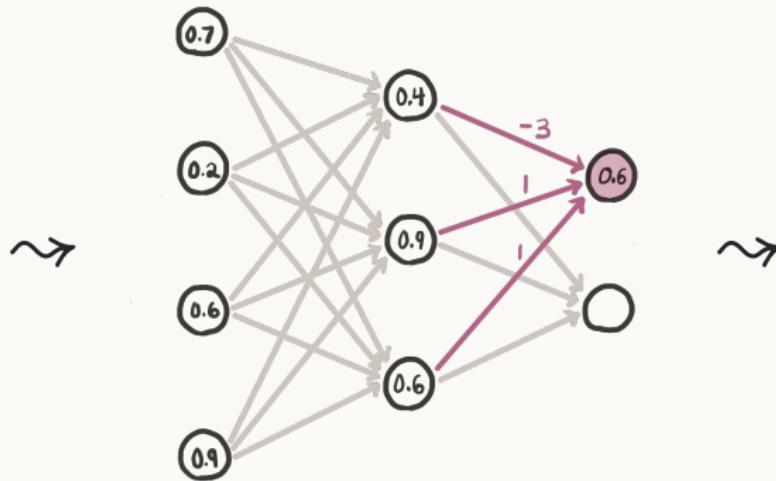
ARTIFICIAL NEURAL NETWORKS

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



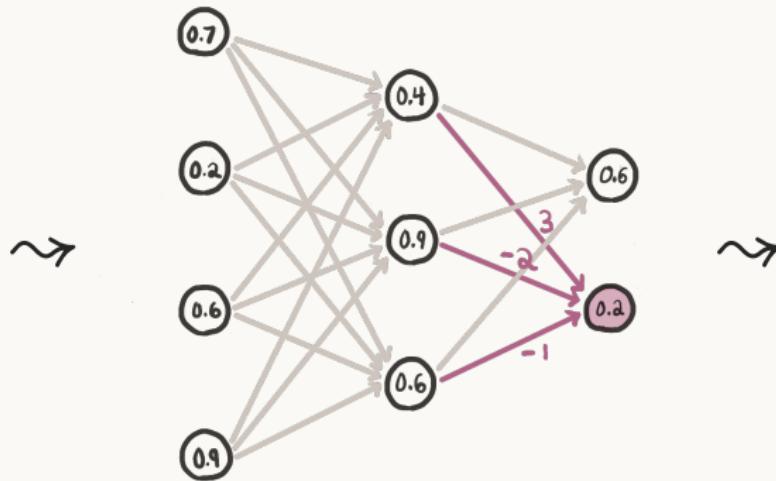
ARTIFICIAL NEURAL NETWORKS

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



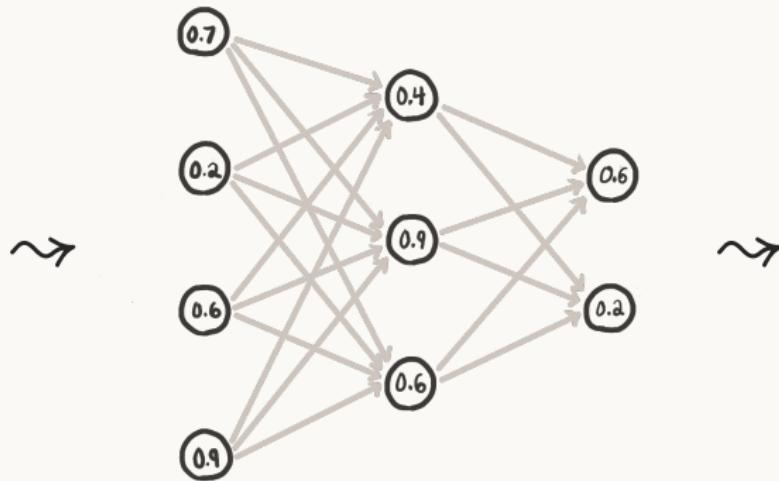
ARTIFICIAL NEURAL NETWORKS

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



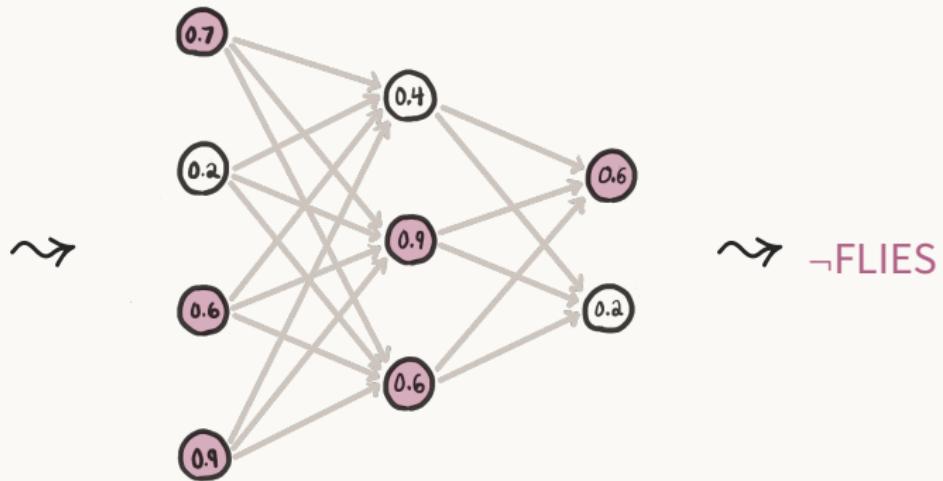
ARTIFICIAL NEURAL NETWORKS

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

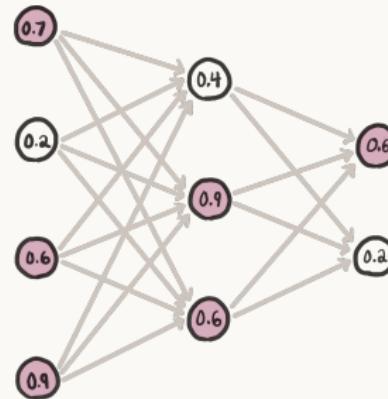
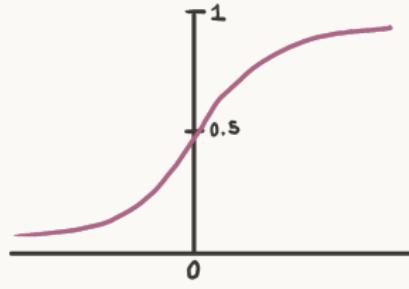


ARTIFICIAL NEURAL NETWORKS

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

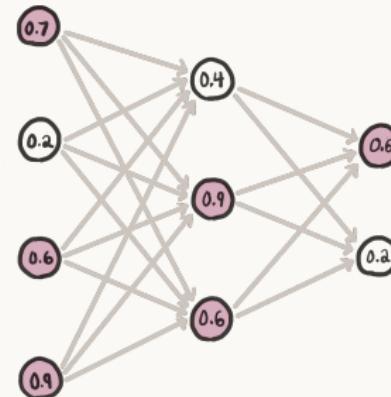
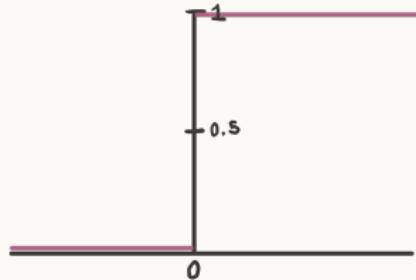


(BINARY) ARTIFICIAL NEURAL NETWORKS



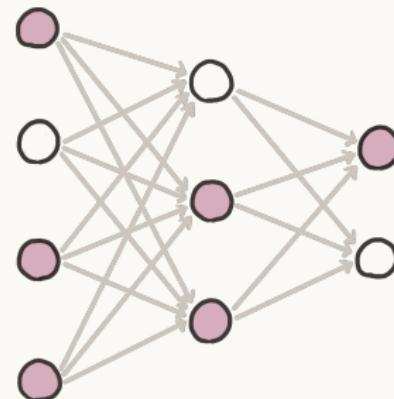
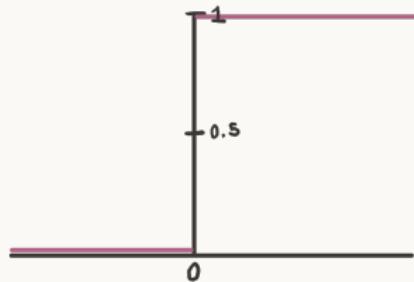
- We take the activation function A to be a binary step function
- This is a useful abstraction for connecting nets with logic, formal languages, and automata
- The net's activation patterns are just sets of neurons.

(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 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 automata
- The net's activation patterns are just sets of neurons.

PLAN FOR TODAY

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

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

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

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

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 $p \in \text{PROP}$. The usual abbreviations are \vee , \rightarrow , and \mathbf{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 $p \in \text{PROP}$. The usual abbreviations are \vee , \rightarrow , and \mathbf{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 , $F_{S_0} : \wp(N) \rightarrow \wp(N)$ is given by

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

“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 , $F_{S_0} : \wp(N) \rightarrow \wp(N)$ is given by

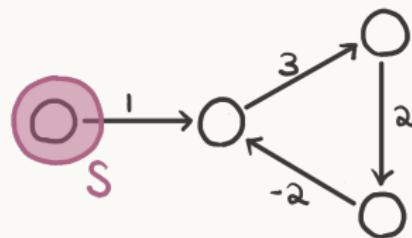
$$F_{S_0}(S) = S_0 \cup \{w \mid A\left(\sum_{u \in \text{preds}(w)} W(u, w) \cdot \chi_S(u)\right) = 1\}$$

“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

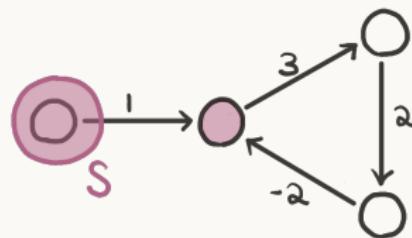
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



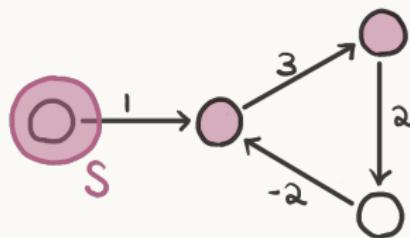
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



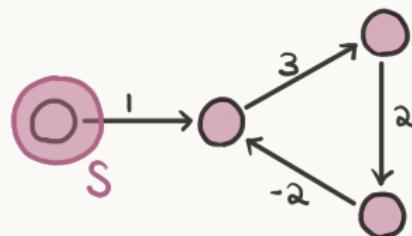
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



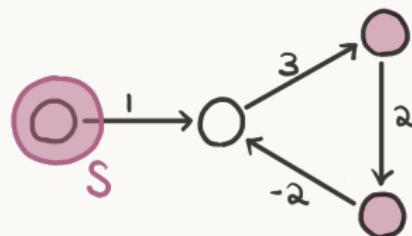
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



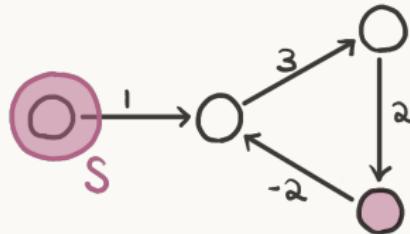
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



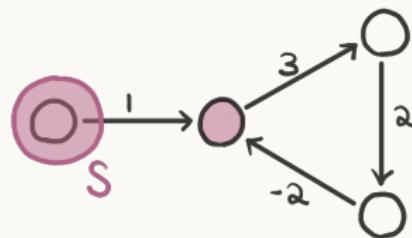
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



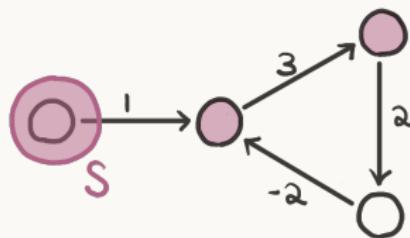
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



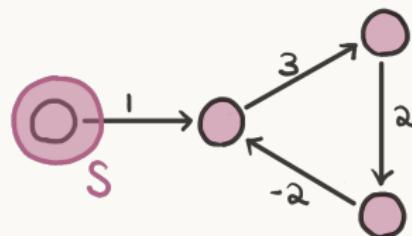
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



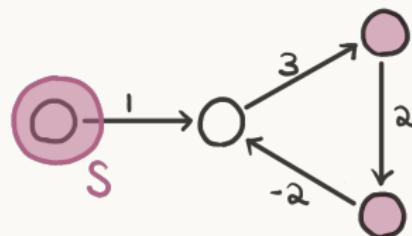
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

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



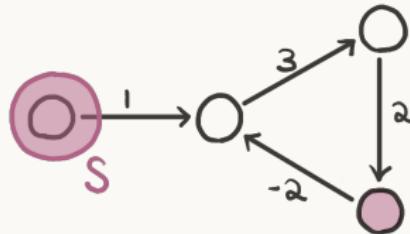
SEMANTICS: NEURAL NETWORK CLOSURE OPERATOR

- 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 ,

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

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 ,

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

eventually stabilizes to a unique activation pattern.

Definition

Let $\text{Clos} : \wp(N) \rightarrow \wp(N)$ 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

$V : Prop \rightarrow \wp(N)$, and a neuron (“world”) $w \in N$:

$$\mathcal{N}, w \models p \quad \text{iff} \quad w \in V(p) \text{ for each } p \in Prop$$

$$\mathcal{N}, w \models \neg\varphi \quad \text{iff} \quad \text{not } \mathcal{N}, w \models \varphi$$

$$\mathcal{N}, w \models \varphi \wedge \psi \quad \text{iff} \quad \mathcal{N}, w \models \varphi \text{ and } \mathcal{N}, w \models \psi$$

$$\mathcal{N}, w \models A\varphi \quad \text{iff} \quad \text{for all } w \in N \text{ whatsoever, } \mathcal{N}, w \models \varphi$$

$$\mathcal{N}, w \models \langle \mathbf{C} \rangle \varphi \quad \text{iff} \quad w \in \text{Clos}(\llbracket \varphi \rrbracket)$$

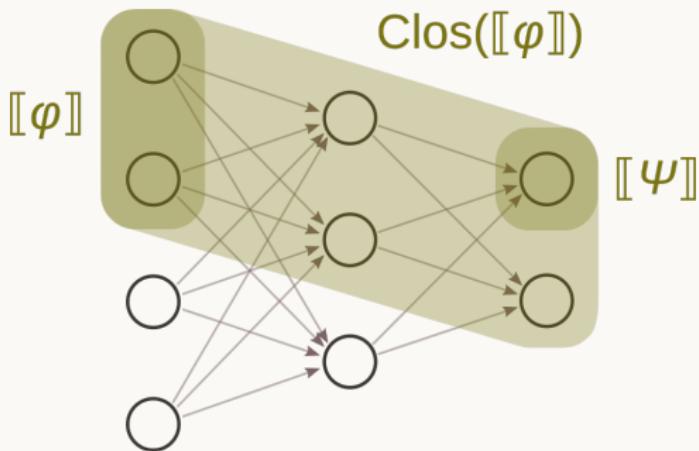
and dually:

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

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

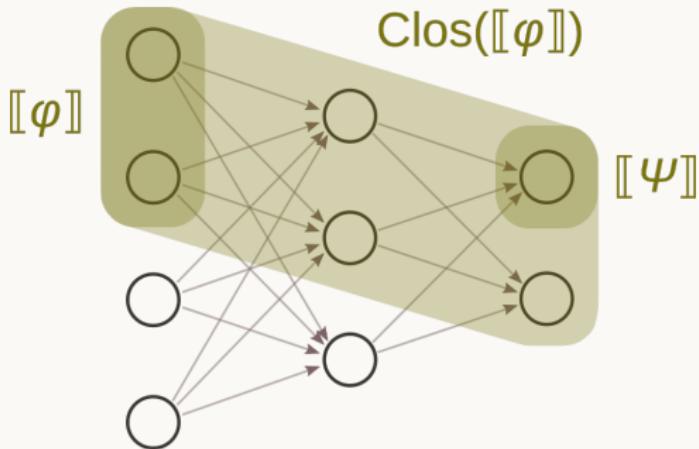
- The **C** modality gives information about the net's answer to an input



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

EXPRESSING NEURAL NETWORK INFERENCE

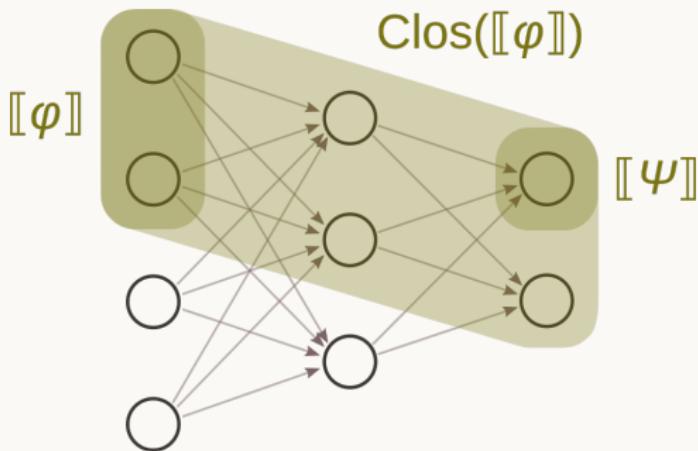
- The **C** modality gives information about the net's answer to an input



The net satisfies $\mathbf{A}(\mathbf{C}(\varphi) \rightarrow \psi)$ iff The net satisfies $\mathbf{A}(\psi \rightarrow \langle \mathbf{C} \rangle(\varphi))$
iff $\text{Clos}(\llbracket \varphi \rrbracket) \supseteq \llbracket \psi \rrbracket$

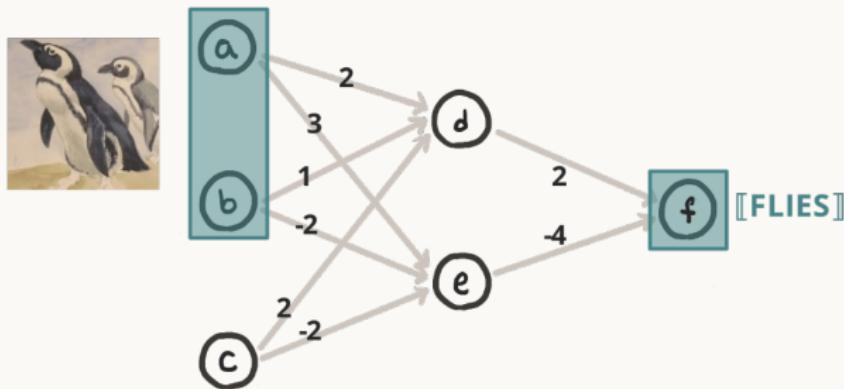
EXPRESSING NEURAL NETWORK INFERENCE

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



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

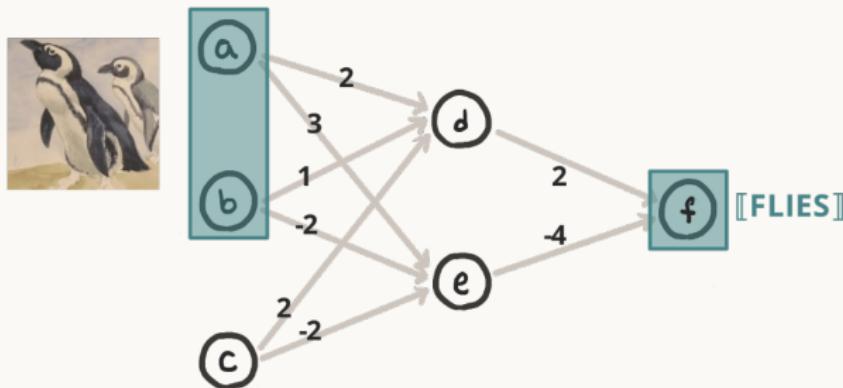
EXAMPLE: EXPRESSING NEURAL NETWORK INFERENCE



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

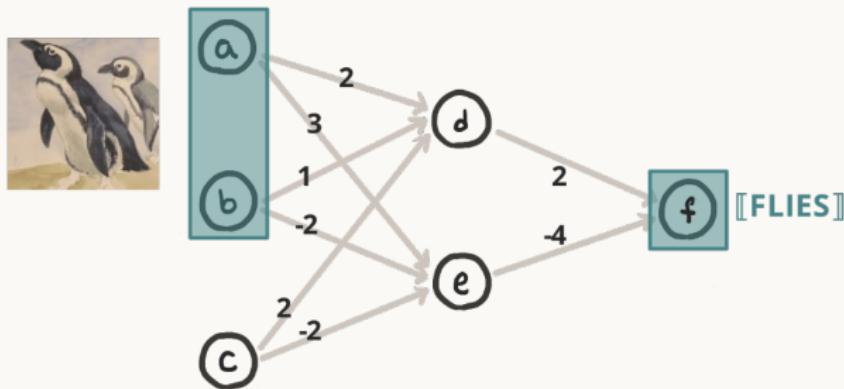
$$\mathcal{N} \not\models A(C(PENGUIN) \rightarrow FLIES)$$

EXAMPLE: EXPRESSING NEURAL NETWORK INFERENCE



- In the exercises, we will ask you will to show
$$\mathcal{N} \not\models \mathbf{A}(\mathbf{C}(\text{PENGUIN}) \rightarrow \text{FLIES})$$
- 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

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

- This means the net does not classify penguins as flying
- Yet, if we take 〔BIRD〕 = {a, b, c},

$$\mathcal{N} \models \mathbf{A}(\mathbf{C}(\text{BIRD}) \rightarrow \text{FLIES})$$

ADDITIONAL COMMENTS ON THIS LOGIC

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

- What does this remind you of?

ADDITIONAL COMMENTS ON THIS LOGIC

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

- What does this remind you of?
 - $\text{best}_\leq([\![\varphi]\!]) \subseteq [\![\psi]\!]$

ADDITIONAL COMMENTS ON THIS LOGIC

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

- What does this remind you of?
 - $\text{best}_\leq([\![\varphi]\!]) \subseteq [\![\psi]\!]$
 - $\mathbf{A}(\mathbf{C}(\varphi) \rightarrow \psi)$, taken as a conditional, behaves exactly like $B^\varphi \psi$

ADDITIONAL COMMENTS ON THIS LOGIC

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

- What does this remind you of?
 - $\text{best}_\leq([\![\varphi]\!]) \subseteq [\![\psi]\!]$
 - $\mathbf{A}(\mathbf{C}(\varphi) \rightarrow \psi)$, 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 $\mathbf{A}(\mathbf{C}(\varphi) \rightarrow \psi)$ iff The net satisfies $\mathbf{A}(\psi \rightarrow \langle \mathbf{C} \rangle(\varphi))$
iff $\text{Clos}([\![\varphi]\!]) \supseteq [\![\psi]\!]$
iff **The net classifies φ as ψ**

- What does this remind you of?
 - $\text{best}_{\leq}([\![\varphi]\!]) \subseteq [\![\psi]\!]$
 - $\mathbf{A}(\mathbf{C}(\varphi) \rightarrow \psi)$, 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

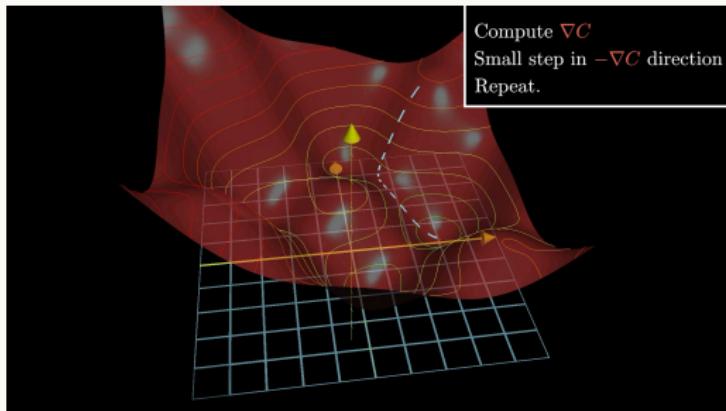
- **Unsupervised Updates**
 - The network learns from data that is unlabeled (no expected answer or classification)
 - Each update softly increases the net's preference for the input
 - Hebb's rule, Oja's rule, & competitive learning rule
- **Supervised Updates**
 - The network learns from labeled data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

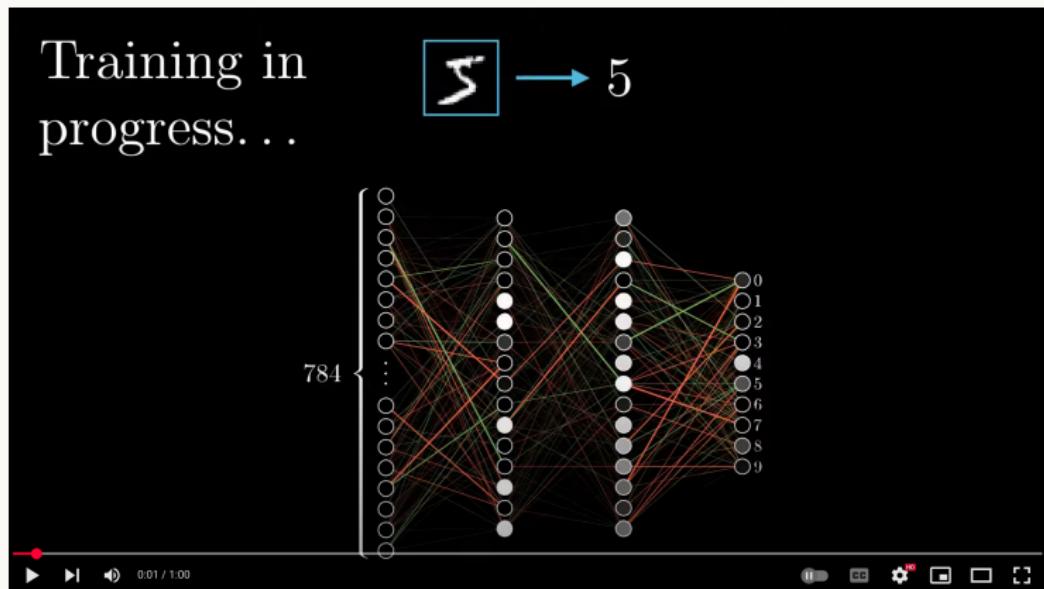
- Backpropagation is the most widely used neural network update rule
- **Main idea:** Backprop implements gradient descent on a net's weights



- 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=cANqroNVdI8>



ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

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

ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

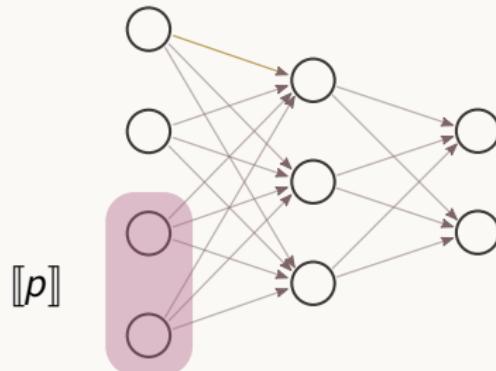
- What if we could have a complete characterization of Backprop, like we did for public announcement, L_{EX}, and MINI?
 - That would be wonderful!
 - Unfortunately, this is still an open problem
- **Proof of concept:** Can we do this for any neural network update at all?

ADDITIONAL COMMENTS ON NEURAL NETWORK UPDATES

- What if we could have a complete characterization of Backprop, like we did for public announcement, L_{EX}, 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**

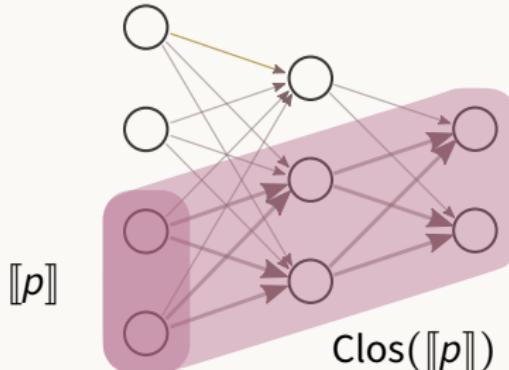
HEBBIAN UPDATE RULE

Neurons that fire together wire together



HEBBIAN UPDATE RULE

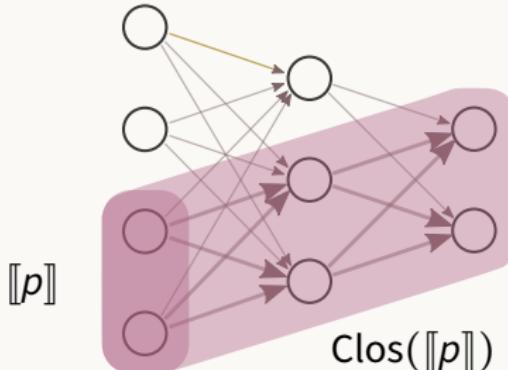
Neurons that fire together wire together



- 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: $\text{Hebb}(N, \llbracket \varphi \rrbracket) = (N, E, W', A)$, where $W'(u, w) = W(u, w) + \eta \cdot \chi_{\text{Clos}(\llbracket \varphi \rrbracket)}(u) \cdot \chi_{\text{Clos}(\llbracket \varphi \rrbracket)}(w)$

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

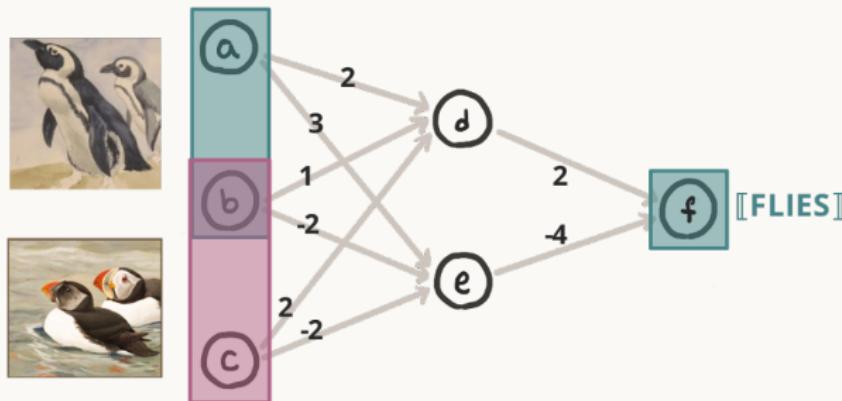
- If nobody ever told you that penguins don't fly, how could you come to believe they don't?
 - Observe animals with similar features that **don't** fly
- Now imagine you believe penguins don't fly. What could cause you to change your mind?

HEBBIAN UPDATE RULE: EXAMPLES

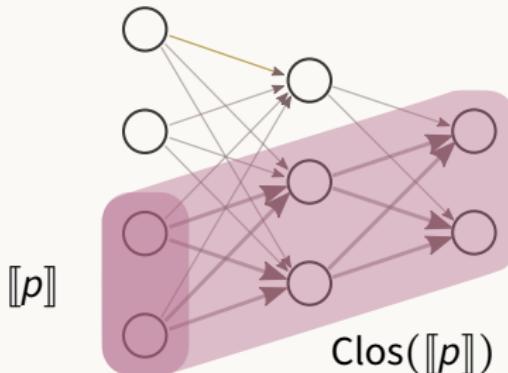
- If nobody ever told you that penguins don't fly, how could you come to believe they don't?
 - Observe animals with similar features that **don't** fly
- Now imagine you believe penguins don't fly. What could cause you to change your mind?
 - Observe animals with similar features that **do** fly

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

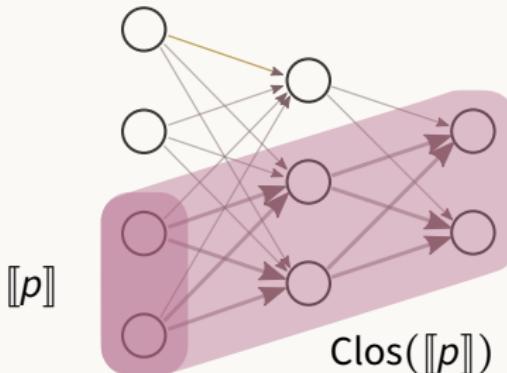


HEBB*: “FIXED-POINT” HEBBIAN UPDATE



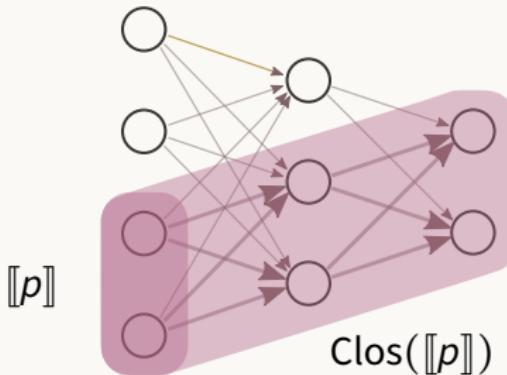
- If we repeatedly apply $\text{Hebb}((, N), S)$, eventually these weights will saturate (they will not inhibit any incoming activations)

HEBB*: “FIXED-POINT” HEBBIAN UPDATE



- If we repeatedly apply $\text{Hebb}((, N), S)$, 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 $\text{Hebb}((, N), S)$, 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 $\text{Hebb}^*(N, S) = (N, E, W', A)$, 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

$\mathcal{U} : \mathbf{Net} \rightarrow \mathcal{L} \rightarrow \mathbf{Net}$ be any unsupervised update:

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

For Hebbian updates in particular:

$$\mathcal{N}, w \vDash [\varphi]_{Hebb}\psi \quad \text{iff} \quad \text{Hebb}(N, [[\varphi]]), w \vDash \psi$$

$$\mathcal{N}, w \vDash [\varphi]_{Hebb^*}\psi \quad \text{iff} \quad \text{Hebb}^*(N, [[\varphi]]), w \vDash \psi$$

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

REDUCTION LAWS FOR HEBB*

The following formulas are valid over neural network models:

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

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

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

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

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

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

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

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

Hebb*-Closure $[\varphi]_{\text{Hebb}^*} \langle \mathbf{C} \rangle \psi \leftrightarrow$
 $\langle \mathbf{C} \rangle ([\varphi]_{\text{Hebb}^*} \psi \vee (\langle \mathbf{C} \rangle \varphi \wedge \diamond (\langle \mathbf{C} \rangle \varphi \wedge \langle \mathbf{C} \rangle [\varphi]_{\text{Hebb}^*} \psi)))$

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

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

Hebb*-Closure $[\varphi]_{\text{Hebb}^*} \langle \mathbf{C} \rangle \psi \leftrightarrow$
 $\langle \mathbf{C} \rangle ([\varphi]_{\text{Hebb}^*} \psi \vee (\langle \mathbf{C} \rangle \varphi \wedge \diamond (\langle \mathbf{C} \rangle \varphi \wedge \langle \mathbf{C} \rangle [\varphi]_{\text{Hebb}^*} \psi)))$

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

$\mathcal{N}, w \models \diamond \varphi \quad \text{iff} \quad \text{there is an } E\text{-path from some } u \in \llbracket \varphi \rrbracket \text{ to } w.$

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 “classic notes style” article gives a brief survey of neural network models of conditionals. After short introductions into the studies of neural networks and conditional logics, the article focuses on the work of Leitgeb (2017) on learning conditionals in the logical investigation of dynamic systems in general, and of neural networks in particular. It also discusses how neural networks learn conditionals in dynamical systems, which logical relations these conditionals obey, and what the main open problems in this area are.

Keywords: Conditionals, Neural networks, Dynamical systems, Neurosymbolic 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 faces to playing chess.

Logic deals with formal systems of reasoning; in particular, deductive logic studies formal inference rules. In philosophy of mind, one of the central questions is whether, and to what degree, the brain supports a hypothesis, as measured by the logic, whether or not the hypothesis is likely to be true.

AI and philosophy of mind are two fields that have been interacting, and, recently, these two areas developed in cooperation to each other: neural networks are quantitative dynamic systems, and deductive logic is a formal system of inference rules that can be described by mathematical equations, whereas logic is subject to numerous restrictions (such as the law of non-contradiction, the law of excluded middle, etc.).

The “problem of induction” is regarded as belonging to the boundary between logic and philosophy of mind. At present this assessment is changing: the emergence of logical formalism for neural networks has led to the development of a new field of research, which processes on the representational level give rise to the expectation that the dynamics of inference and belief formation can be understood in terms of logical rules, and thus the formal rules of inference.

1

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

The Thirty-Eighth Annual Conference on Artificial Intelligence (AAI-18)

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

Caleb Schultz-Kisby¹, Scott A. Blanco², Lawrence S. Moss³

¹Department of Mathematics, Indiana University

²Department of Mathematics, Indiana University

³Indiana University CSE Department

[kisby, blanco, lsmoss]@indiana.edu

Abstract

This paper is a contribution to neural network semantics, a formalized framework for neuro-symbolic AI. The key idea is to understand neural networks as learners that map between operators in neural network states. In this paper, we do this for the iterated Hebbian learning rule. We show that the iterated Hebbian learning rule is a simple learning rule that maps between operators in neural network states. This mapping is called the reduction rule. The reduction rule can be seen as a key finding of the paper. One motivation for this work is that it provides a way to understand the Hebbian learning rule as a reduction rule. This makes that computation for the logic of neural networks more transparent. The reduction rule is a neural network state operator, which is a neural network state operator that obeys basic laws. These reduction axioms are given in parts (1) a human-comprehensible form, and (2) a formalized form. Part (1) consists of plausibility arguments, and (2) an approach to building neural networks with the reduction rule in mind.

Introduction

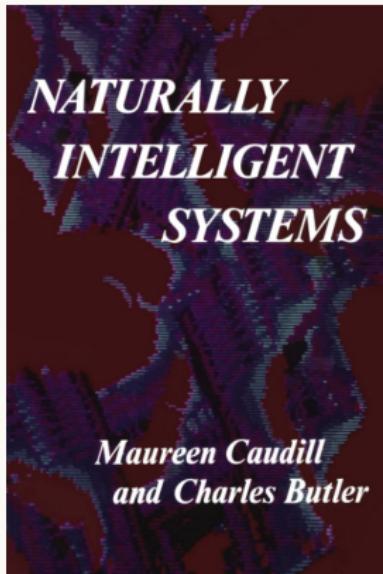
The two dominant paradigms of AI, connectionist neural networks and symbolic systems, have long seemed irreconcilable. Connectionist neural networks are able to learn complex inferences in a human-interpretable language, but the logic and inference rules of these networks are not transparent. Symbolic neural networks are flexible and robust at learning from an environment, but they lack the ability to learn complex inferences. In response to this dilemma, neurosymbolic systems have been proposed as a community-wide effort to integrate neural and symbolic systems, while retaining the advantages of both. Despite the many successes of neurosymbolic systems (cf., e.g., Littman et al. 2001; See & Butler 2005; Beival et al. 2017; Barker et al. 2017; Schultz-Kisby et al. 2018; Schultz-Kisby et al. 2019), there is still much work to be done.

In this paper, we focus on one specific aspect of neurosymbolic systems, which we call model network semantics.

In this paper, we focus on one specific aspect of neurosymbolic systems, which we call model network semantics.

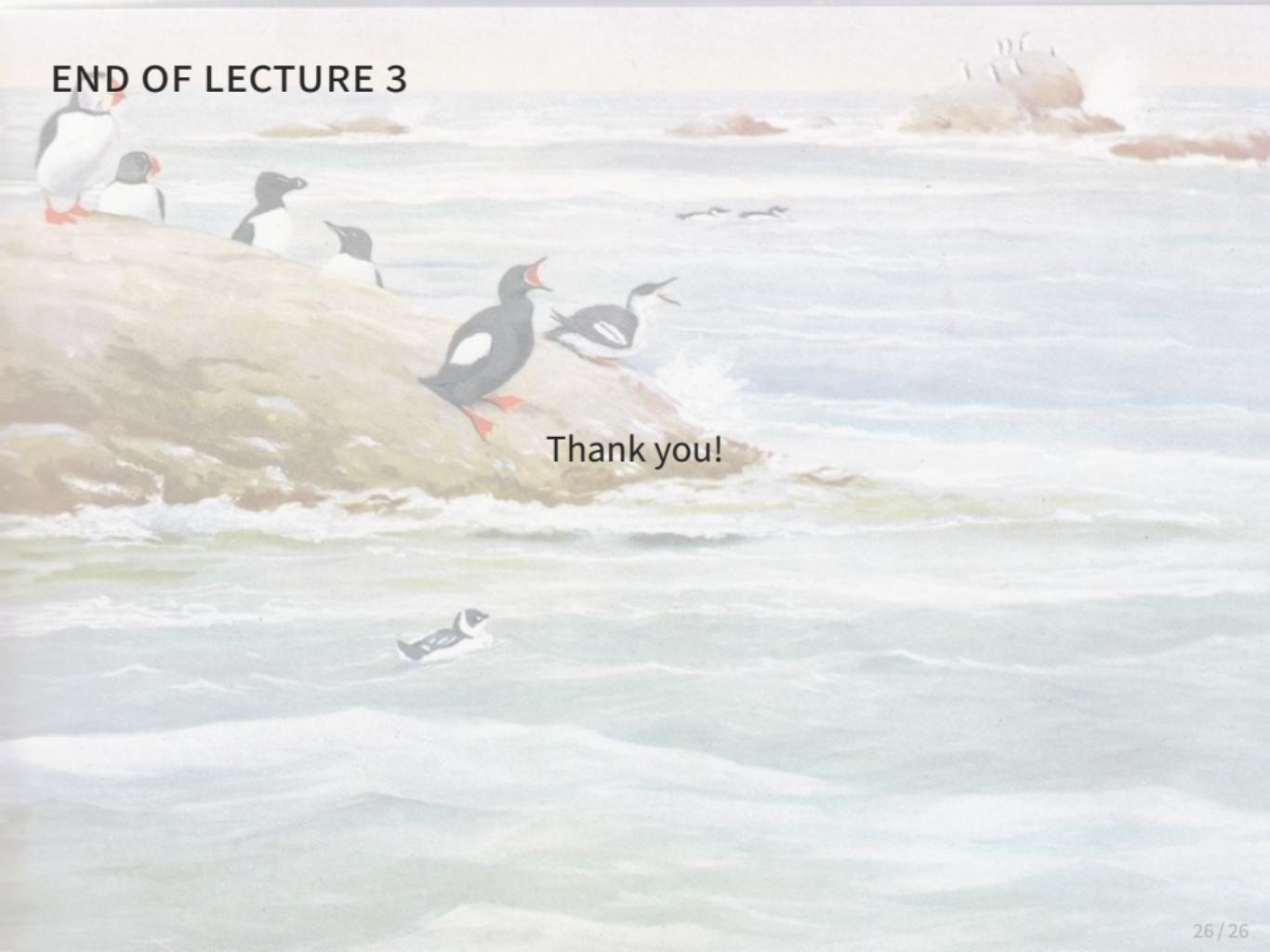
In this paper, we focus on one specific aspect of neurosymbolic systems, which we call model network semantics.

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



(c) Naturally Intelligent Systems by Caudill & Butler

END OF LECTURE 3

A painting of several puffins on a rocky shore. In the foreground, two puffins are standing on a light-colored rock, facing each other with their heads tilted upwards and mouths open, possibly communicating or singing. Another puffin is swimming in the choppy, greenish-blue water below. More puffins are scattered across the rocks in the background.

Thank you!