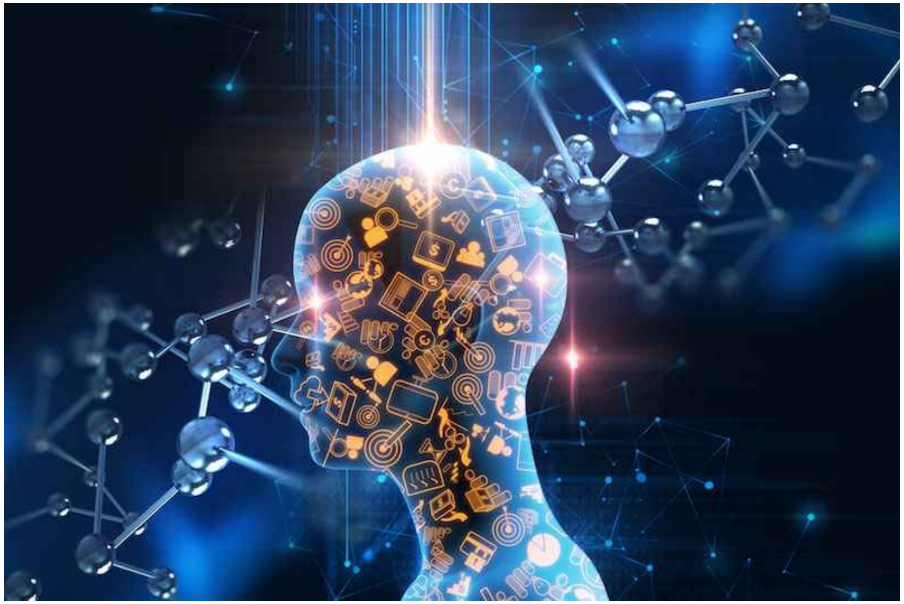
LEARNING HIERARCHICALLY STRUCTURED CONCEPTS

Nancy Lynch

Spring, 2020 Brain Algorithms Reading Group

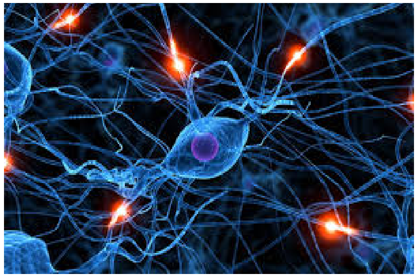
Meeting 11, May 1, 2020

Joint work with Frederik Mallmann-Trenn



An Algorithmic Theory of Brain Networks

- We use a distributed algorithms approach to study abstract versions of problems solved by real brains: Decision-making, attention, encoding and representation, recognition, learning.
- Define problems as probabilistic functions from input firing sequences to output firing sequences.

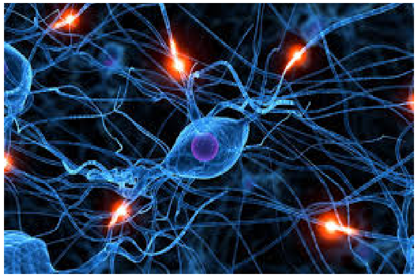


- Define abstract algorithms, based on those that occur in brains, modeled as discrete, stochastic Spiking Neural Networks (SNNs).
- Prove that the algorithms solve the problems.

- Analyze algorithms: network size, convergence time, energy usage.
- Prove corresponding lower bounds.

An Algorithmic Theory of Brain Networks

- General questions:
  - How do results depend on model assumptions (about timing, memory, probability)?
  - How robust are algorithms (to noise, errors, changes)?
  - What can be learned, what must be pre-designed?
  - Describe algorithms using composition, abstraction?
- Work should lead to:
  - New understanding of brain behavior.
  - Opportunities for work in two communities:
    - Theoretical computer scientists can study abstract problems, prove upper and lower bounds.
    - Neuroscientists can model real brain mechanisms, validate models with experiments.

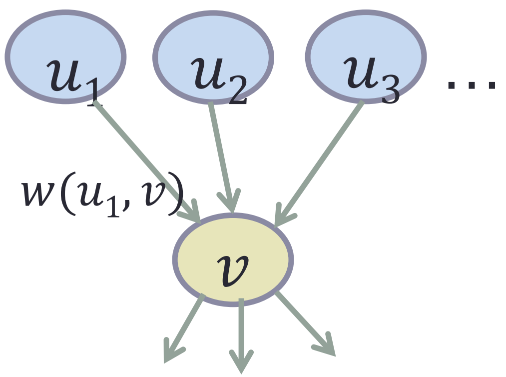


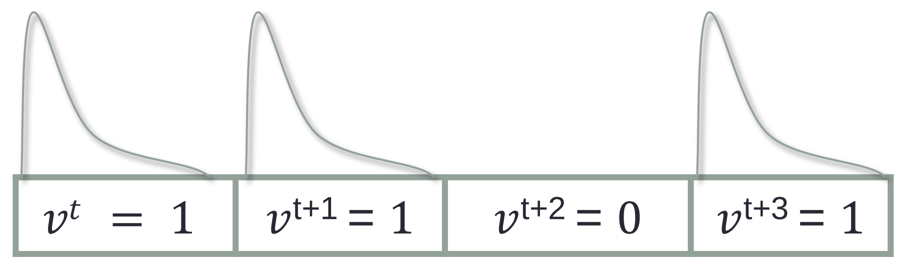
Our Relevant Prior Work

1. Model: Stochastic Spiking Neural Networks
2. Winner-Take-All algorithms and lower bounds
3. Similarity testing, compression, short-term memory,…

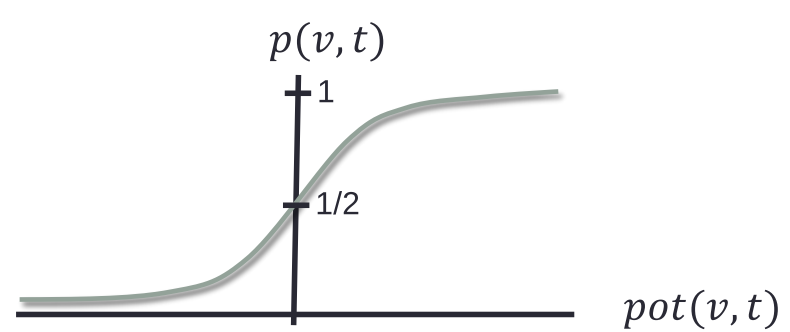
1. Model: Stochastic Spiking Neural Networks

- Nancy Lynch, Cameron Musco, Merav Parter. Computational tradeoffs in biological neural networks: Self-stabilizing Winner-Take-All networks. ITCS 2017. ArXiv 2019.
- $v^t = 1$ if and only if neuron v spikes at time t.



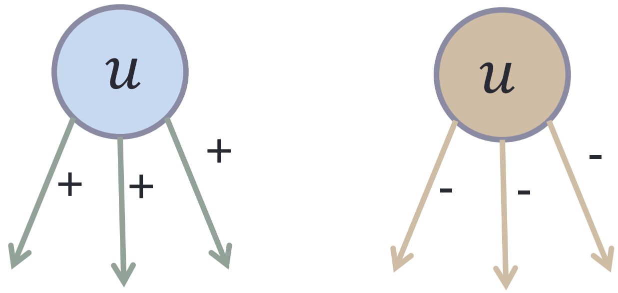


- $pot(v,t) = \Sigma_u \, u^{t-1} \, w(u,v) - b(v)$
- $\Pr[v^t = 1] = 1/(1 + e^{-pot(v,t)})$



Stochastic Spiking Neural Networks

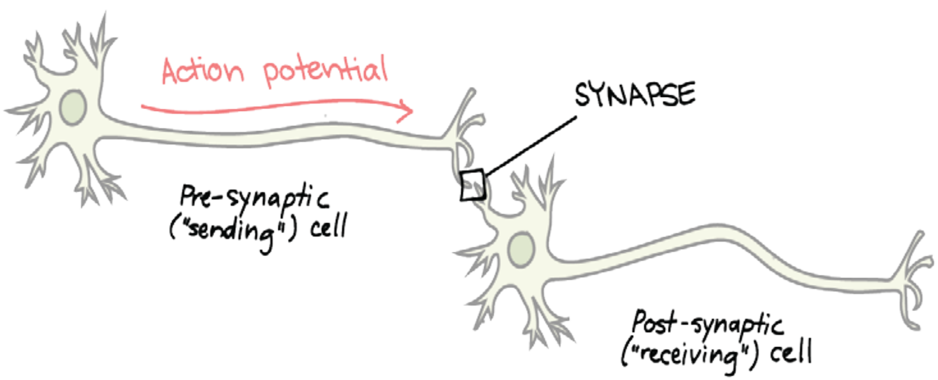
- We usually assume that neurons are strictly inhibitory or strictly excitatory, i.e., $w(u,v) \geq 0$ for all v or $w(u,v) \leq 0$ for all v.

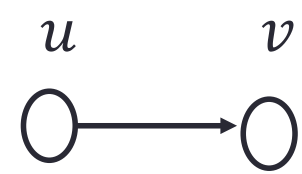


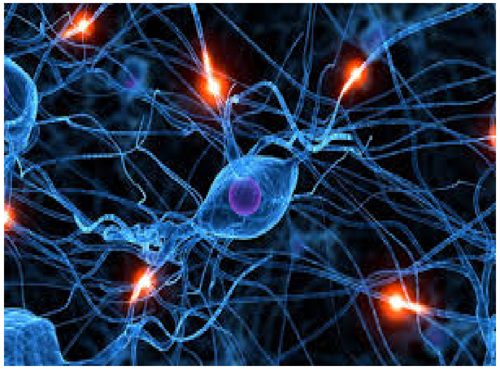
- We usually ignore other biological features: Refractory period, spike propagation delay, memory, noise on synapses,…
- Some can be simulated in our model.
- We also sometimes augment the model.

Neural Network Model

- A weighted directed graph, nodes represent neurons, edges represent synapses, weights represent synaptic strength.
- Regard $weight = 0$ as absence of edge, $weight > 0$ as excitatory, $weight < 0$ as inhibitory.

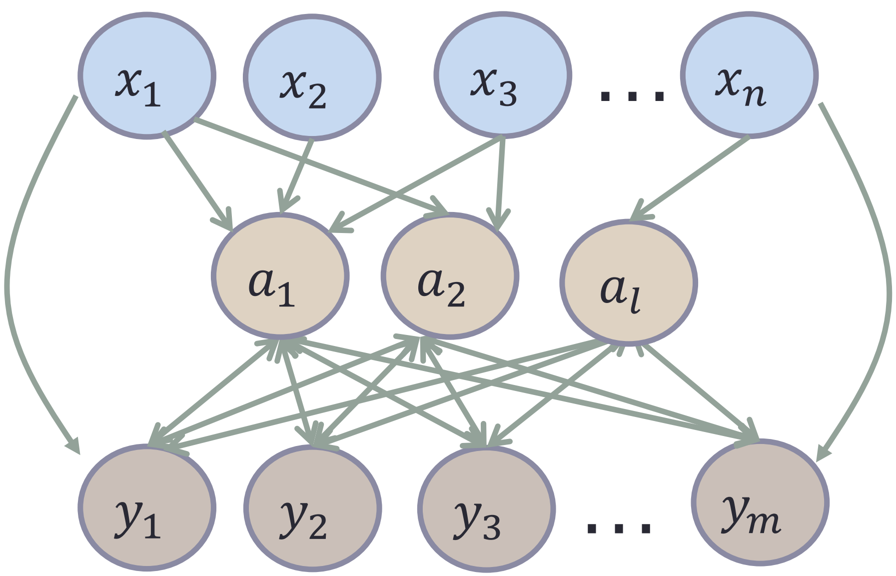






Neural Network Model

- Neurons are either input neurons X, output neurons Y, or auxiliary neurons A.
- Input and output neurons are excitatory.
- Auxiliary neurons may be either excitatory or inhibitory.



Network Dynamics

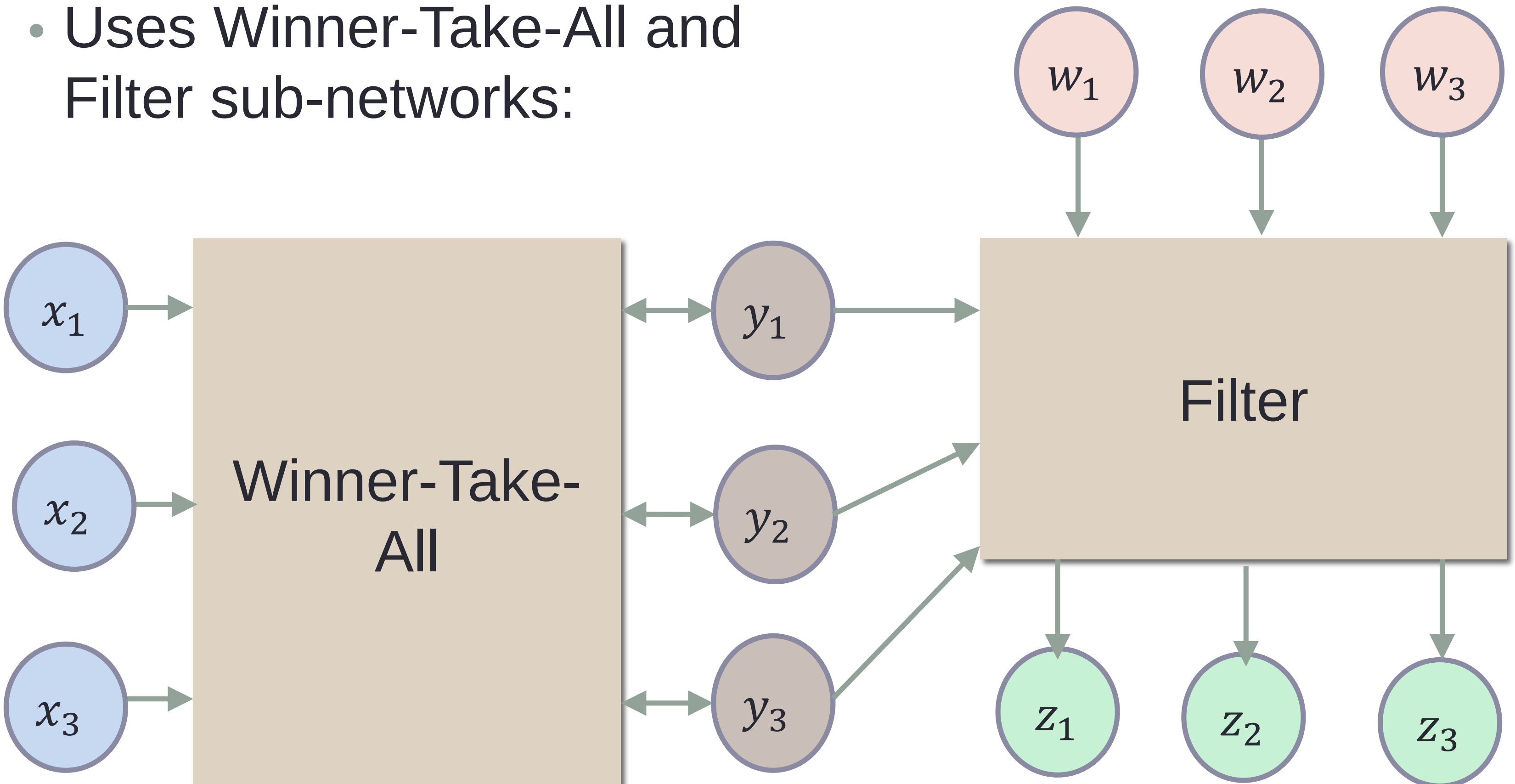
- Configuration C: Assigns a firing state, 0 or 1, to each neuron; $C(u) = 1$ means it's firing and $= 0$ means it's not.
- Execution $\alpha = C^0, C^1, C^2, \ldots$, a sequence of configurations.
- $u^t = C^t(u)$ denotes the firing state of neuron u at time t.
- Input firing patterns are arbitrary.
- Initial firing patterns for non-input (auxiliary and output) neurons are part of the network definition.
- For every infinite input execution, the network produces a probability distribution on infinite executions, by applying the stochastic firing dynamics for all non-input neurons at all rounds.

Composing Spiking Neural Networks

- Nancy Lynch, Cameron Musco. A Compositional Model for Spiking Neural Networks. arXiv 1808.03884
- Idea: Combine networks that solve simple problems into larger networks that solve more complex problems.
- E.g., consider two networks $\mathcal{N}_1$ and $\mathcal{N}_2$.
- Compatibility:
  - Internal neurons of $\mathcal{N}_1$ cannot be neurons of $\mathcal{N}_2$, and vice versa.
  - $\mathcal{N}_1$ and $\mathcal{N}_2$ have no common output neurons.
  - May have common input neurons.
  - Outputs of one may be inputs of the other.
- Composition rules:
  - Neurons of $\mathcal{N}_1 \circ \mathcal{N}_2$ = union of neurons of $\mathcal{N}_1$ and $\mathcal{N}_2$.
  - Outputs of $\mathcal{N}_1 \circ \mathcal{N}_2$ = union of outputs of $\mathcal{N}_1$ and $\mathcal{N}_2$.
  - Likewise for internal neurons.
  - Inputs: Inputs of $\mathcal{N}_1$ that aren't outputs of $\mathcal{N}_2$, and vice versa.

Composing Spiking Neural Networks

- Attention network: Processes a sequence of inputs and focuses attention on the "relevant" ones.
- Uses Winner-Take-All and Filter sub-networks:

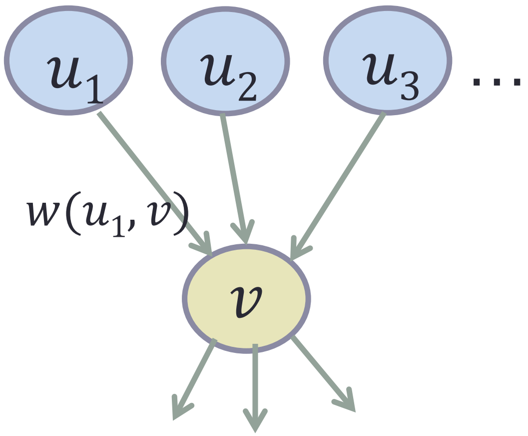


Adding memory to neuron states

- Lili Su, C. J. Chang, Nancy Lynch. Spike-Based Winner-Take-All Computation. Neural Computation 2019.
- Basic model: Neuron's state at each time is a Boolean,
  - 1 = firing, 0 = not firing.
- Augmented model with local memory:
  - Useful in some algorithms, bio-realistic
  - Neuron may remember its own m previous firing states.
  - And/or its own incoming potentials based on its incoming neighbors' m previous firing states:

$$pot(v,t) = \Sigma_u \, u^{t-1} \, w(u,v) - b(v)$$

  - General memory model, allows modeling of accumulated potential, refractory periods.

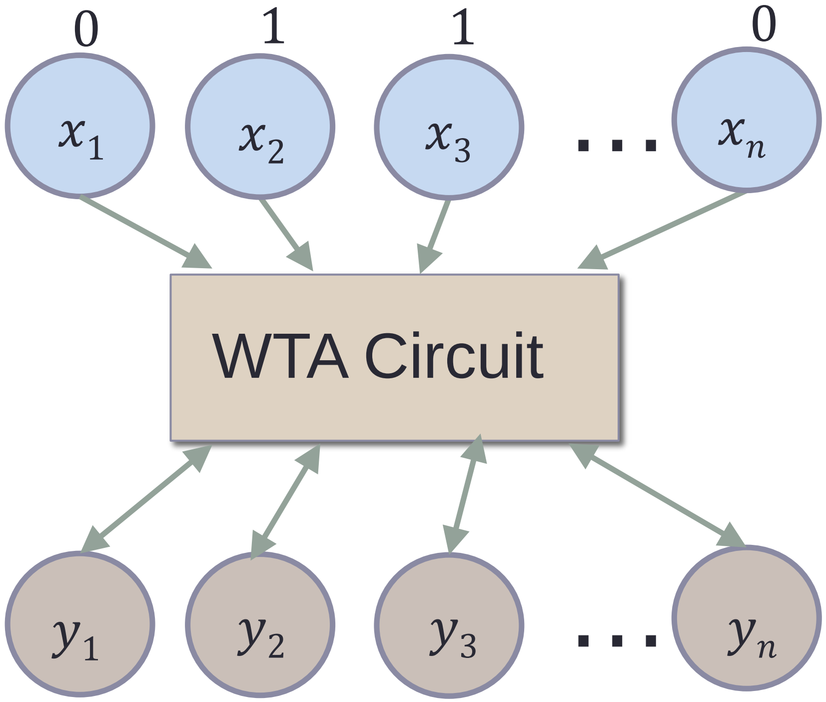


Learning

- Lynch, Mallmann-Trenn. Learning Hierarchically Structured Concepts, ArXiv 2019, 2020.
- We have added features to model changing edge weights, as needed to support learning algorithms.
- New state component, a vector of incoming weights.
- Changes based on Oja's rule, which incrementally adjusts weights to correspond to firing patterns for incoming neighbors.

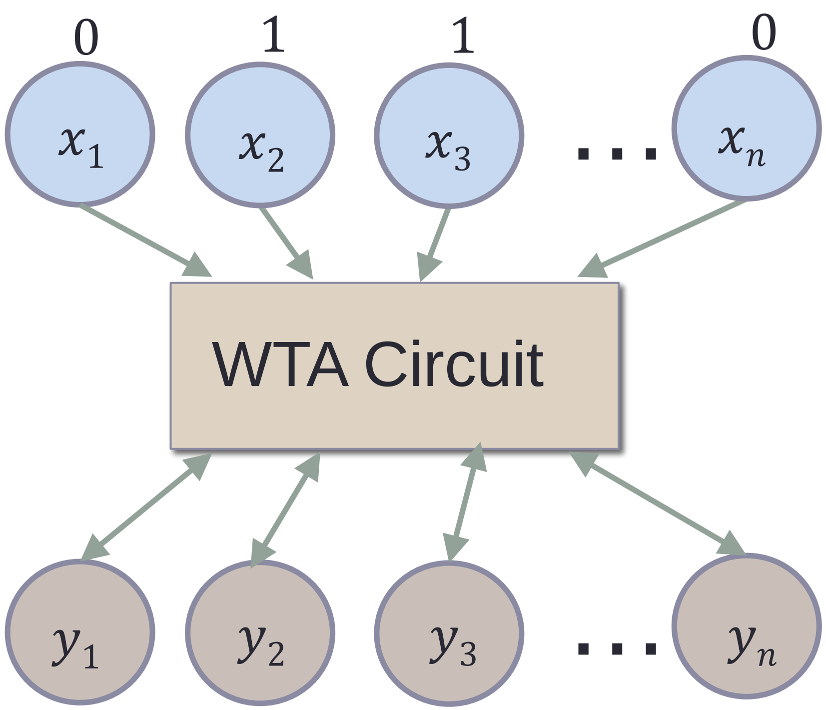
2. Winner-Take-All

- Nancy Lynch, Cameron Musco, Merav Parter. Computational Tradeoffs in Biological Neural Networks: Self-Stabilizing Winner-Take-All Networks. ITCS 2017.
- Cameron Musco PhD thesis, Chapter 5
- New ArXiv version 2019.



Winner-Take-All: $WTA(n, t_c, t_s, \delta)$

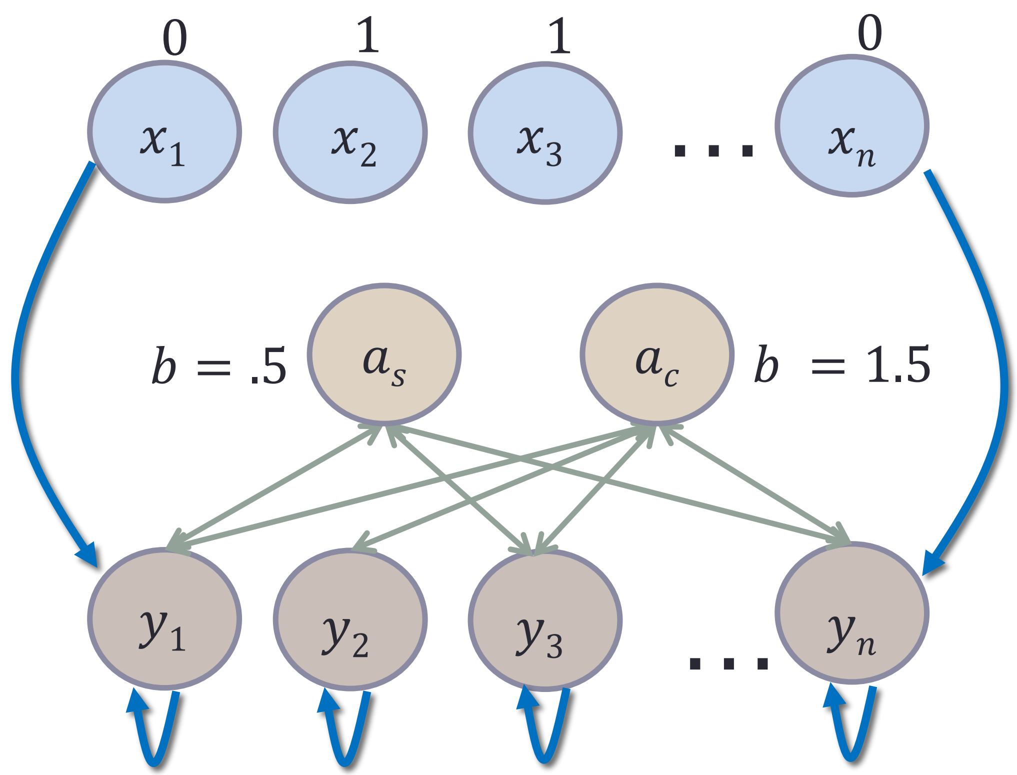
- n fixed inputs, n corresponding outputs.
- Starting from any state, with probability $\geq 1 - \delta$, network:
  - Converges, within a (short) time t_c, to a single firing output, which corresponds to a firing input, and then
  - Remains stable for a (long) time t_s.



Simple Solution with Two Inhibitors

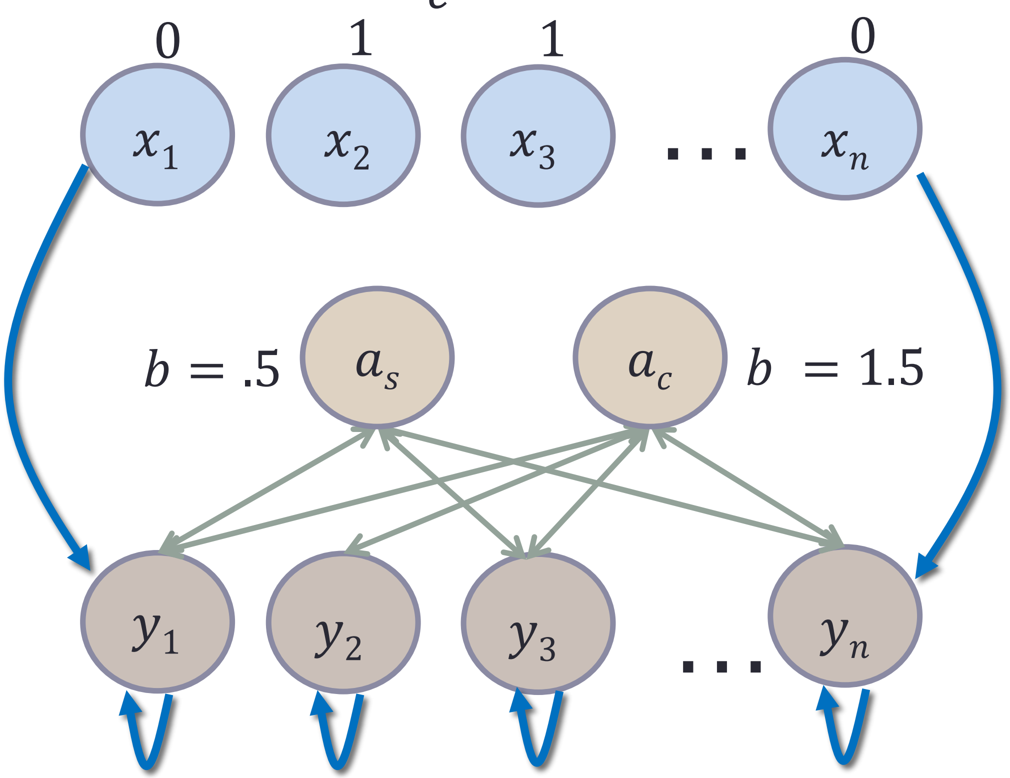
- Stability inhibitor a_s:
  - Fires with high probability whenever ≥ 1 outputs fire.
  - Prevents outputs that didn't fire at time t from firing at time $t+1$.

- Convergence inhibitor a_c:
  - Fires with high probability whenever ≥ 2 outputs fire.
  - Causes any output that fires at time t to fire at time $t+1$ with probability ~½.



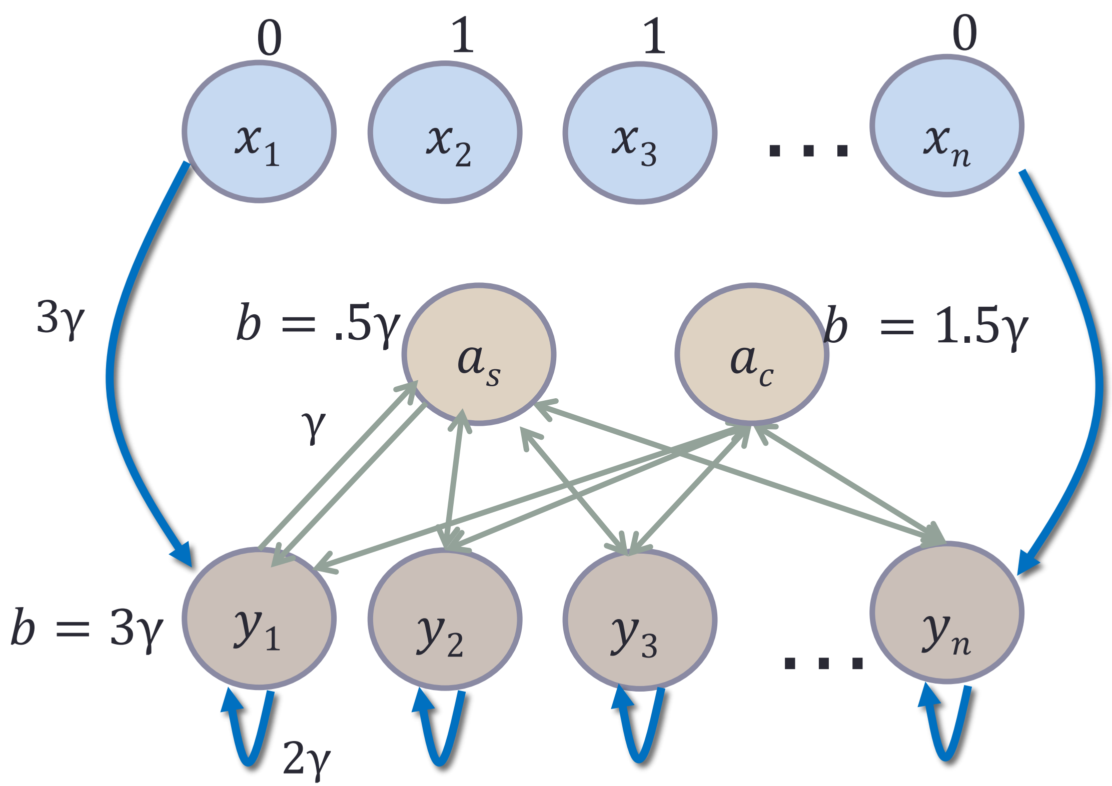
Simple Solution with Two Inhibitors

- Main idea: Approximately half of currently-firing outputs stop firing at each step.
- So with constant probability, there is some time $t_c \leq \log n$ such that exactly one output fires at time t_c.
- Moreover, after time t_c, with high probability, this selected output continues to fire for a long time t_s.
- During this stable period, only a_s fires, preventing all other outputs from firing.



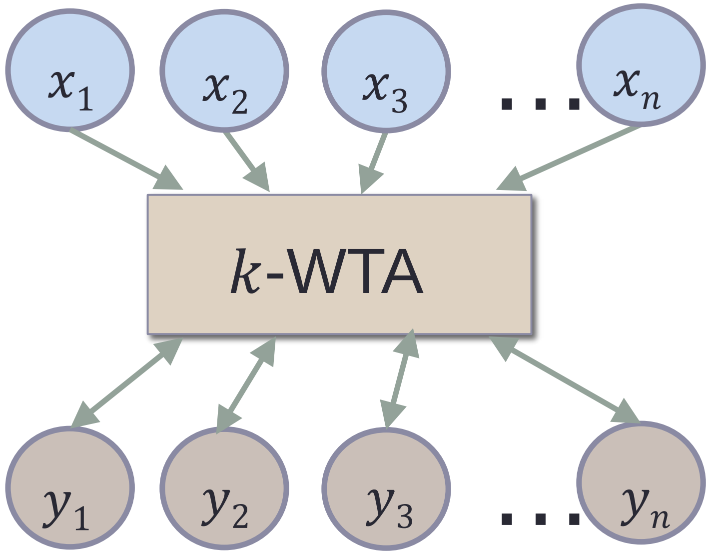
Main Theorem

- Theorem 1: Assume $\gamma \geq c \log(n\, t_s/\delta)$. Then starting from any state, with probability $\geq 1-\delta$, the network converges, within time $t_c \approx c \log n \log\left(\frac{1}{\delta}\right)$, to a single firing output corresponding to a firing input, and remains stable for time t_s.
- Also:
  - More than two inhibitors can give faster convergence.
  - Can't solve WTA much faster with two inhibitors.
  - Can't solve it at all with one inhibitor.



Extension to k-WTA

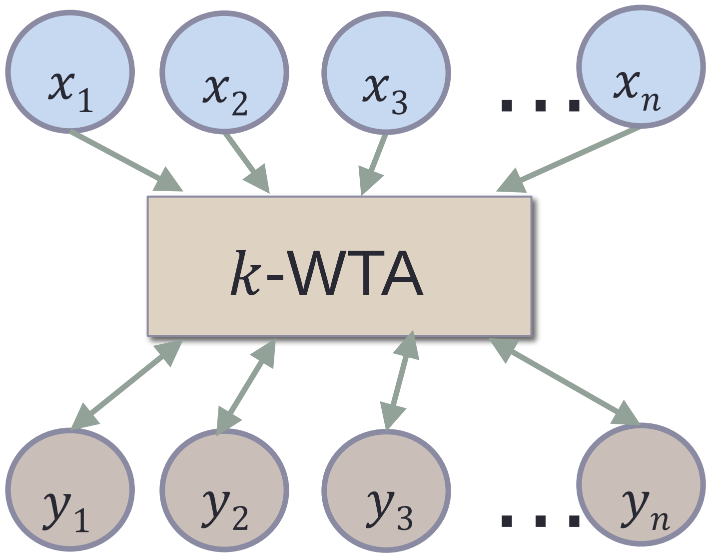
- Lili Su, CJ Chang, Nancy Lynch. Spike-Based Winner-Take-All Computation. Neural Computation 2019.
- Now inputs fire, not at every round, but at different “rates”.
- Model input firing by independent Bernoulli processes.
- Problem: Choose the k neurons with highest firing rates.



Lower Bound

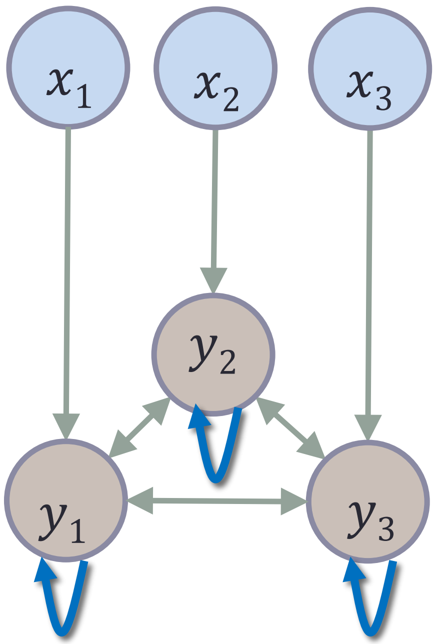
- Fix n, k.
- For a set R of possible rates, define $D(R)$ to be a certain statistic, capturing the "minimum distance" between different rates in R (related to KL-divergence).
- Fix an error probability $\delta \in (0,1)$.
- Lower bound theorem: There is no algorithm that solves k-WTA with error probability δ, for all rate assignments from R, and that converges within time

$$\Big((1-\delta)\log\big(k(n-k)\big)-1\Big)\,D(R).$$



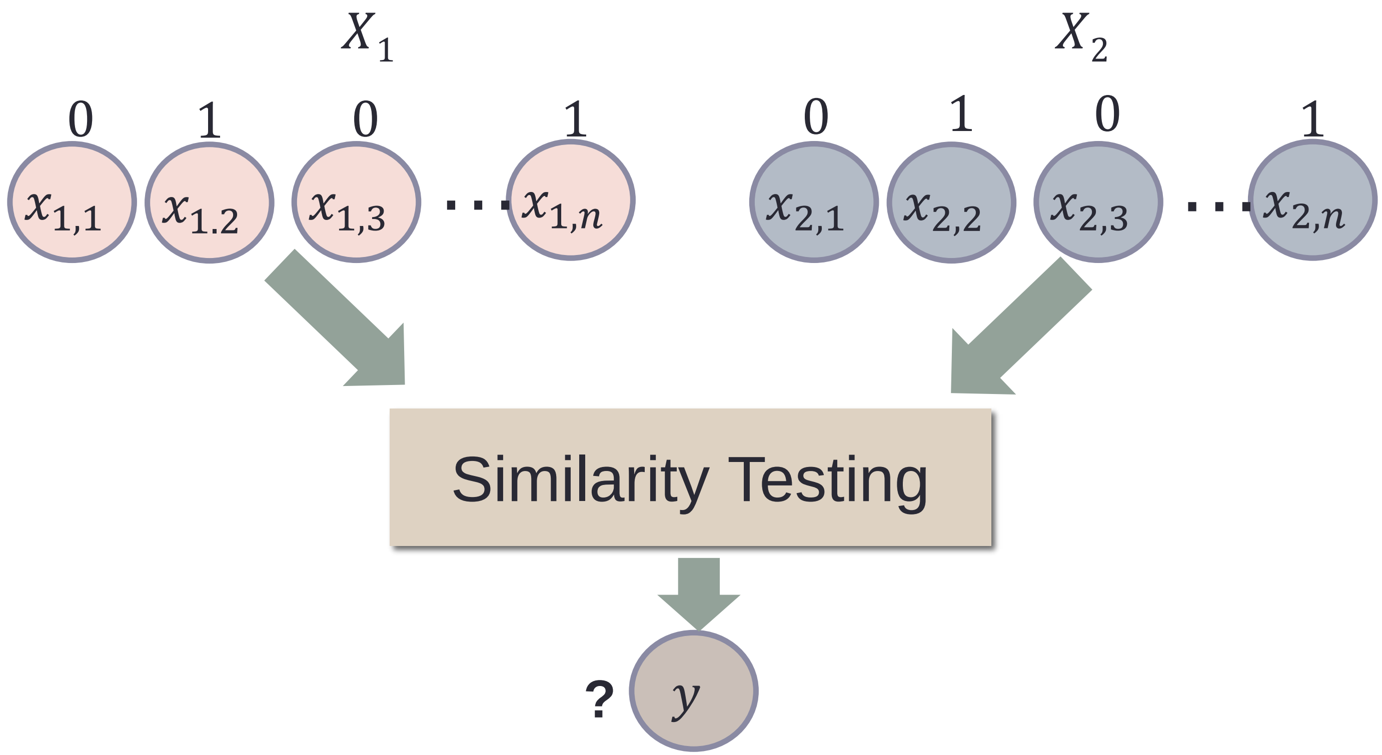
Upper Bound

- Simple algorithm, time $O(\log\left(\frac{1}{\delta}\right) + \log\big(k(n-k)\big)\,D(R))$
- Uses memory: m previous firing states, where $m = \Omega(\log\left(\frac{1}{\delta}\right) + \log\big(k(n-k)\big)\,D(R))$.
- Algorithm idea: Output neurons that fire excite themselves (self-loops), inhibit others (clique).
- Neuron v_i fires at time t exactly if either:
  - It didn't fire at time $t-1$, and its total incoming potential, based on firings at times $t-1, \ldots,$ $t-m,$ is $\geq b$ (its bias), or
  - It did fire at time $t-1$, and its total incoming potential, based on firings at times $t-1, \ldots,$ $t-m,$ is ≥ 1.
- Making this work to solve k-WTA requires fine-tuning the weights and biases.



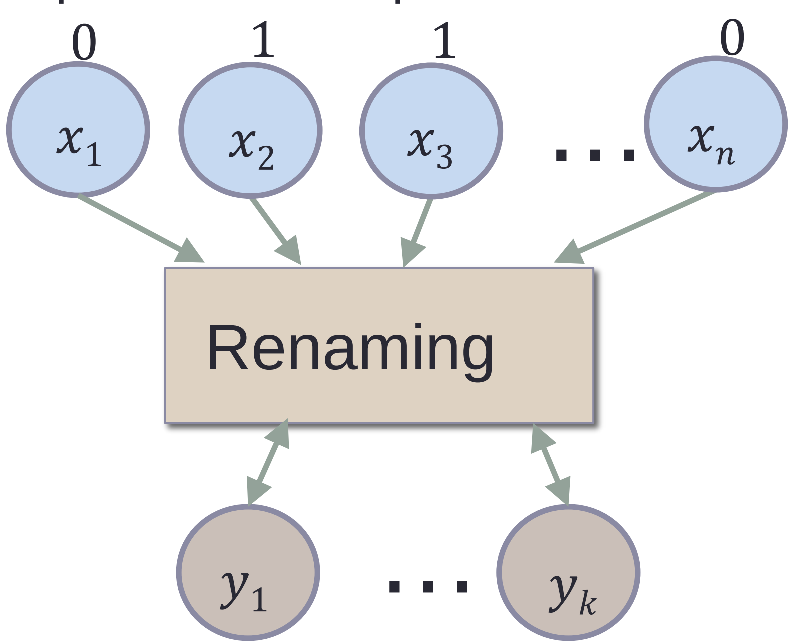
3. Similarity Testing, Compression, Clustering

- Nancy Lynch, Cameron Musco, Merav Parter. Neuro-RAM Unit with Applications to Similarity Testing and Compression in Spiking Neural Networks. DISC 2017.



Short-Term Memory

- Yael Hitron, Nancy Lynch, Cameron Musco, Merav Parter. Random Sketching, Clustering, and Short-Term Memory in Spiking Neural Networks, ITCS 2020.
- n input neurons, $k \ll n$ output neurons.
- An arbitrary set of k distinct input firing patterns are presented, each for "sufficiently long".
- Network should learn a distinct "short code" for each input pattern: a single output neuron should learn to fire in response to later presentation of that same pattern.
- Short-term memory: Coding remembered by persistent firing, self-loops.
- Not long-term: no changes to network.
- Algorithm requires few internal neurons, short training periods.
- Techniques: Random projection, WTA, inhibition of already-assigned outputs.



Learning Hierarchically-Structured Concepts

Nancy Lynch, Frederik Mallmann-Trenn. Learning Hierarchically-Structured Concepts. arXiv:1909.04559v2, February, 2020.

1. Introduction
2. Data Model
3. Network Model
4. Problem Statements
5. Algorithms for Recognition and Noise-Free Learning
6. Extension: Noisy Learning
7. Lower Bounds
8. Conclusions

1. Introduction

- Q: How are concepts with structure represented in the brain? How are they recognized? How are they learned?
- Inspiration: Network dissection in deep convolutional neural networks for computer vision [Zhou, Bau, Oliva, Torralba 2017].
- Lower layers of the network learn basic concepts, higher layers learn higher-level concepts.
- General thesis: Structure that is naturally present in concepts gets mirrored in its brain representation, in some way that facilitates both learning and recognition.
- Consistent with research on visual processing in mammalian brains [Hubel, Wiesel 1959].
- We approach this problem using our SNN-based methods.
- Initial project: Concept hierarchies, in which concepts are built from other concepts,…
- Example: Human consists of a body, a head, two legs,…; Head consists of eyes, nose, mouth, etc.

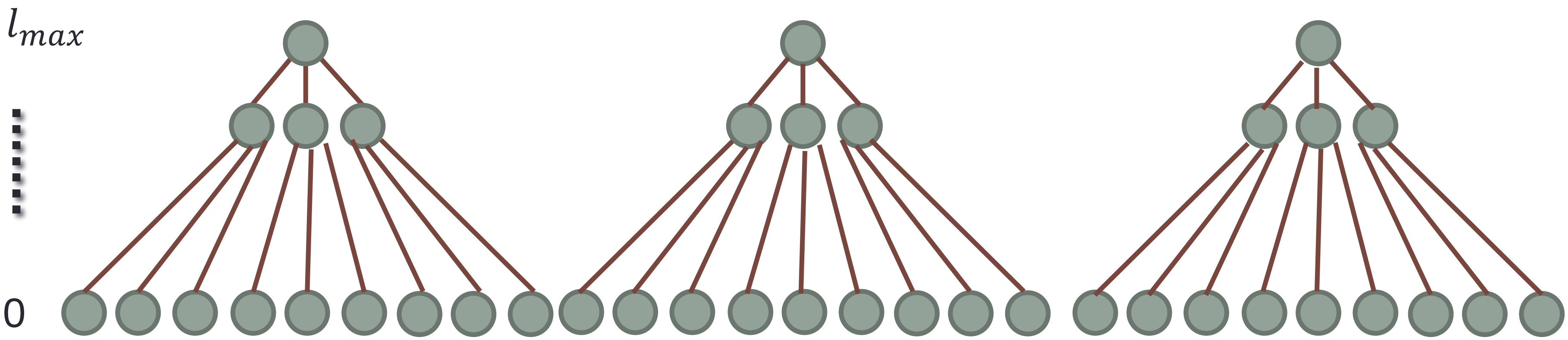
Introduction

- Simplifications:
  - Ignore additional structure, e.g., arms and legs are positioned symmetrically.
  - Our hierarchies are trees, always with the same height and same number of children.

- What we do:
  - Define concept hierarchies, and a layered SNN model.
  - Define what it means for a layered SNN to recognize a particular concept hierarchy; notion is robust to bounded noise.
  - Define what it means to learn a concept hierarchy.
  - Two algorithms (SNNs) that can learn to recognize concept hierarchies (with/without noise during learning).
  - A lower bound showing that, in order to recognize concepts with hierarchical depth ℓ, an SNN must have at least ℓ layers.

2. Data model: Concept hierarchies

- A concept hierarchy C consists of a set C of concepts arranged into a forest.
- Assume uniform degree k.
- l_{max} levels.
- For concept $c \in C_\ell$, define $children(c),\ descendants(c)$.
- $leaves(c)$ = level 0 descendants of c.

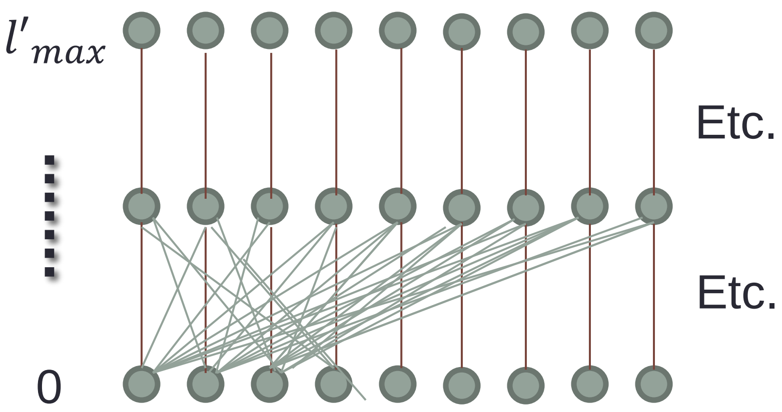


Data model

- Concepts are chosen from a universal set D, which is partitioned into ℓ_{max} levels $D_0, D_1, \ldots$
- $n = |D_0|$

- Support:
  - Fix a concept hierarchy C.
  - For ratio $r \in [0,1]$, recursively define which concepts are r-supported by a particular set B of level 0 concepts:
    - $B_0 = B$.
    - $B_1 =$ level 1 concepts with at least an r-fraction of their children in B_0.
    - $B_l =$ level l concepts with at least an r-fraction of their children in B_{l-1}.

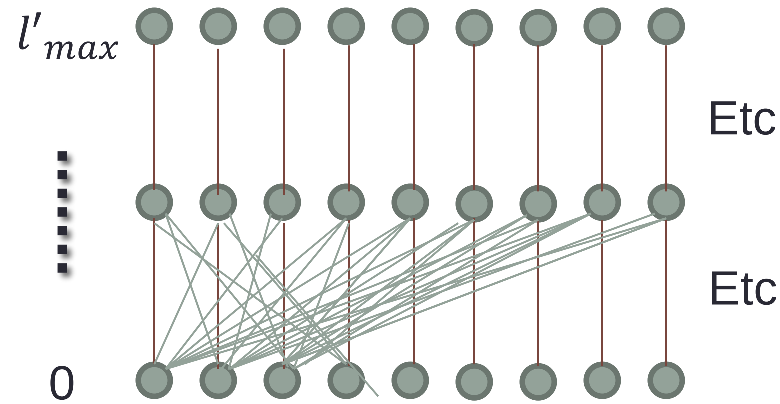
3. Network model



- Feed-forward, layered network *N*.
- Each layer contains n neurons.
- l'_{max} layers.
- All-to-all connections between consecutive levels.
- Assume each level 0 concept has a unique representation neuron $rep(c)$ in layer 0.
- Neuron states:
  - All neurons have a firing status flag in $\{0,1\}$, indicating whether the neuron is currently firing.
  - Higher layer neurons also keep track of incoming weights, represented by n-vectors of reals in the range $[0,1]$.
  - Higher layer neurons record whether they are engaged in learning.

Network model

- Activation function: We use a deterministic threshold τ, rather than stochastic (for simplicity).



- Learning rule: Oja's rule for weight updates [Oja 1982], for a neuron u that is currently engaged:

$$\boldsymbol{w}(t) = \boldsymbol{w}(t-1) + \eta\, z(\boldsymbol{x}(t-1) - z\,\boldsymbol{w}(t-1))$$, where

  - η is the learning rate,
  - $\boldsymbol{x}(t-1)$ is the vector of input firing status values,
  - z is the dot product of $\boldsymbol{x}(t-1)$ and $\boldsymbol{w}(t-1)$, which is the incoming potential at u for round t.

- Network operation: At each round t, first calculate incoming potential, then use activation function to determine the new firing status, then (if engaged) use Oja to update the weights.

Learning Hierarchically-Structured Concepts

1. Introduction
2. Data Model
3. Network Model
4. Problem Statements
5. Algorithms for Recognition and Noise-Free Learning
6. Extension: Noisy Learning
7. Lower Bounds
8. Conclusions

4. Problem statements

- Two problems:
  - Recognizing a concept hierarchy, and
  - Learning to recognize a concept hierarchy.
- We assume here that each item is represented by exactly one neuron (over-simplification, or abstraction).
- In both cases, we are interested in noisy recognition, captured formally using two fractions (ratios) $r_1, r_2 \in [0,1]$, $r_1 \leq r_2$.
- For recognition, we assume a particular concept hierarchy, C.
- For learning, we must accommodate any arbitrary concept hierarchy C that might be presented as input.
- Presenting a set B of level 0 concepts: Allow exactly the $reps(B)$ input neurons to fire (together).

The recognition problem

- Support (recall):
  - Assumes a particular concept hierarchy C.
  - For ratio $r \in [0,1]$, recursively define which concepts are r-supported by a particular set B of level 0 concepts:
    - $B_0 = B$.
    - B_1 = level 1 concepts with at least an r-fraction of their children in B_0.
    - B_l = level l concepts with at least an r-fraction of their children in B_{l-1}.
- For ratios $r_1, r_2, r_1 \leq r_2$, network *N* (r_1, r_2)-recognizes concept hierarchy C provided that for each concept $c \in C$:
  - Concept c has a unique representation neuron $rep(c)$.
  - Suppose that a set B of level 0 concepts in C is presented. Then:
    - If c is r_2-supported by B then $rep(c)$ must fire.
    - if c is not r_1-supported by B then $rep(c)$ must not fire.

Learning problem

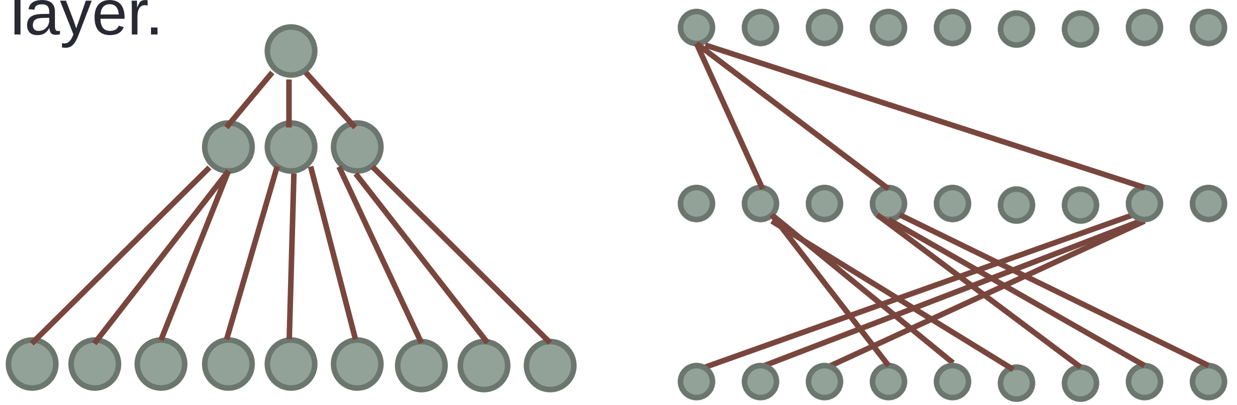
- The network *N* initially doesn't know which concept hierarchy it should learn; suppose in some execution, a particular concept hierarchy *C* is to be learned.
- To show a concept c, present all its leaves (level 0 descendants).
- Work bottom-up, showing each concept only after each of its children has been shown "sufficiently many" times (σ times, for a parameter σ).
- Otherwise, arbitrary interleaving is allowed.
- Then after not too long, the network *N* should reach a state from which it (r_1, r_2)-recognizes concept hierarchy *C*.
- We say that network *N* (r_1, r_2)-learns concept hierarchy *C*.

Learning Hierarchically-Structured Concepts

1. Introduction
2. Data Model
3. Network Model
4. Problem Statements
5. Algorithms for Recognition and Noise-Free Learning
6. Extension: Noisy Learning
7. Lower Bounds
8. Conclusions

5. Algorithms

- Recognition algorithm, for a given concept hierarchy C:
- Embed the hierarchy in the layered network, each level at the same-numbered layer.



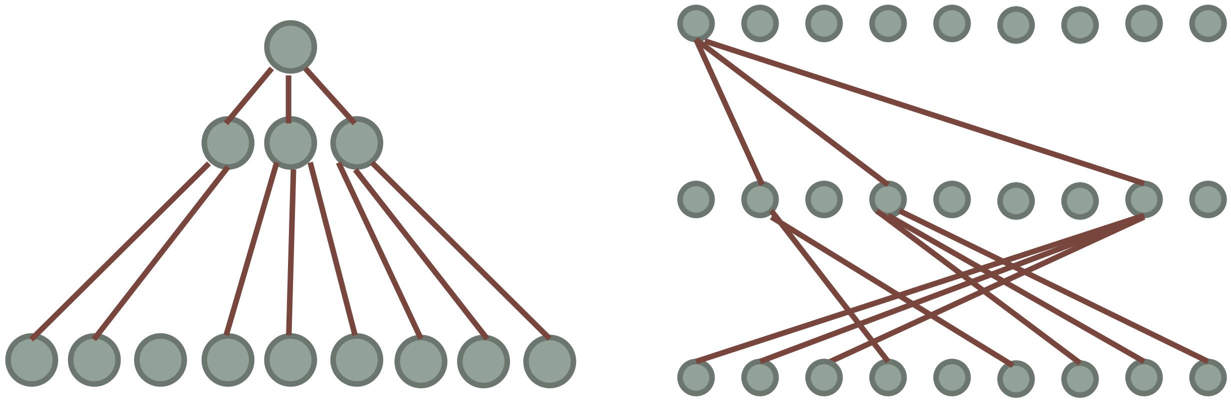
- Weights from reps of children to reps of parents can be 1, others 0 (for example).
- For a given r_1, r_2, set threshold τ for every non-input neuron to $(r_1 + r_2)k/2$.
- This network solves the (r_1, r_2)-recognition problem for concept hierarchy *C;* time is ℓ_{max}.

Learning algorithm

- Assume the network starts in a "clean state", where weights are all $1 / k^{\ell max}$.
- Threshold $\tau = (r_1 + r_2)\sqrt{k}$ / 2.
- Use a bottom-up discipline for showing the concepts in *C*:
  - To show a concept c, present all its leaves (level 0 descendants).
  - Work bottom-up in the concept hierarchy, showing each concept only after each of its children has been shown σ times, for a parameter σ.
  - Otherwise, arbitrary interleaving is allowed.
- Given these inputs and starting conditions, network just executes normally, following the given activation function and Oja's learning rule.
- Results in learning the concepts in *C* bottom-up.

Learning algorithm

- Level l concepts acquire representations in layer l; the algorithm embeds the hierarchy in the network graph, level by level.



- When trying to learn a level l concept c:
  - We assume (inductively) that each of $c's$ children has already acquired a rep in layer $l-1$, which has already learned to fire in response to presentation of its leaves.
  - So presenting all the $reps$ of all the leaves of c together results in firing of the $reps$ of all these children.
  - These induce potential at layer l neurons.

Learning algorithm

- When trying to learn a level l concept c:
  - We assume (inductively) that each of $c's$ children has already acquired a rep in layer $l-1$, which has already learned to fire in response to presentation of its leaves.
  - So presenting all the $reps$ of all the leaves of c together results in firing of the $reps$ of all these children.
  - These induce potential at layer l neurons.
  - We use a Winner-Take-All module to select one neuron u (the one with the highest potential), and put it into "engaged" mode for learning.
  - Neuron u learns using Oja's rule: Incoming edges from $reps$ of c's children get strengthened, others get weakened.
  - Even one step ensures that the same u will later be selected for the same concept c, and u will not later be selected for any other concept.
  - After c has been shown σ times, u will have learned to fire in response to presentation of all its leaves, and more strongly, to a sufficient fraction of the leaves.

Learning algorithm

Theorem 1: Let *N* be the network described above.

Assume that the learning rate η is $\frac{1}{4k}$.

Let $r_1, r_2 \in [0,1]$, $r_1 < r_2$.

Let $\epsilon = (r_2 - r_1)$ / $(r_1 + r_2)$.

Let C be any concept hierarchy with max level ≤ max layer in *N.*

Suppose that the concepts in C are shown according to a σ–bottom-up presentation schedule, where

$$\sigma = O(\ (1\ /\eta\ k\ )(\ell_{max}\ \log(k) + 1/\epsilon) \quad + \quad b\log(k)\ ).$$

Then *N* (r_1, r_2)-learns C.

- Proof: A series of lemmas analyzing the step-by-step changes caused by using Oja's rule.
- The first term bounds the time to increase the weights of the needed edges to something in the range $[1$ / $(1+\epsilon)\sqrt{k},\ 1$ / $\sqrt{k}]$. That is, to roughly 1 / $\sqrt{k}$.
- The second term bounds the time to decrease the weights of the unwanted edges to at most 1 / k^{lmax+b}.

6. Extension: Noisy Learning

- Bottom-up discipline for the noise-free learning algorithm:
  - To show a concept c, present all its leaves (level 0 descendants).
  - Work bottom-up in the concept hierarchy, showing each concept only after each of its children has been shown σ times, for a parameter σ.
  - Otherwise, arbitrary interleaving is allowed.
- Now relax this discipline so that not all the children need to be shown all the time.
- To show a concept c, determine a random size pk subset of its children, and for each, a random size pk subset of their children,…(recursively).
- Present the resulting set B of leaves of c.
- Work bottom-up as before, showing each concept only after each of its children has been shown σ times, for parameter σ.

Noisy Learning

- To show a concept c, determine a random size pk subset of its children, and for each, a random size pk subset of their children,…(recursively).
- Present the resulting set B of leaves of c.
- Work bottom-up as before, showing each concept only after each of its children has been shown σ times, for parameter σ.

Theorem 2: Analogous to Theorem 1, but with a larger value of σ.
Let *N* be the network described above, with a different constraint on the learning rate η.
Then *N* (r_1, r_2)-learns C, with high probability.
Proof: Similar to Theorem 1, but it takes a bit longer to learn each concept, and the learning occurs only with high probability.

Learning Hierarchically-Structured Concepts

1. Introduction
2. Data Model
3. Network Model
4. Problem Statements
5. Algorithms for Recognition and Noise-Free Learning
6. Extension: Noisy Learning
7. Lower Bounds
8. Conclusions

7. Lower bounds

Theorem 3: If network *N* (r_1, r_2)-recognizes concept hierarchy *C* with $r_2^2 < 2r_1 - r_1^2$, then the number of layers in *N* must be $\geq$ number of levels in *C.*

For example, consider $r_1 = \frac{1}{3}, r_2 = \frac{2}{3}$.

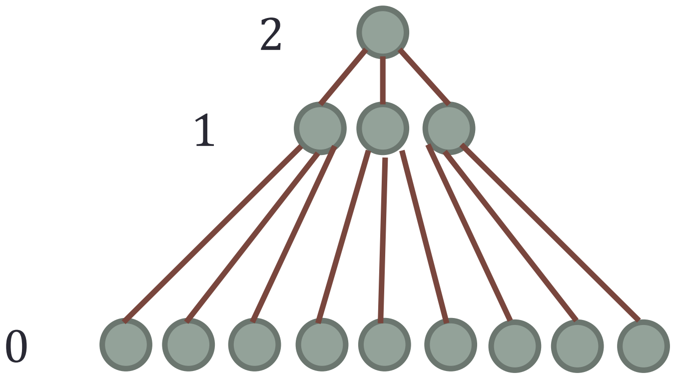
Proof idea: If there are too few layers, some child relationships aren't explicitly represented. Causes confusion between cases where the network is supposed to recognize a concept and cases where it is required not to.

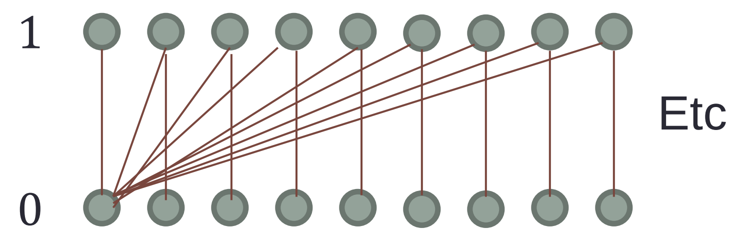
Similar-sounding lower bounds on number of levels have been proved [Mhaskar, Liao, Poggio 2016], [Telgarsky 2016], but using very different methods (function approximation theory).

Proof: Uses induction on the number of levels in *C*; first consider the base case.

Base case: 2 levels, 1 layer

Theorem 4: Suppose that concept hierarchy C has max level 2 and network N has max layer 1. Suppose that $r_2^2 < 2r_1 - r_1^2$. Then N does not (r_1, r_2)-recognize concept hierarchy C.





Proof: Suppose it does, and consider any level 2 concept c.

Reps for c and its children must be in layer 1, so $rep(c)$ cannot be influenced by reps of c's children, but only its grandchildren.

For each child b of c, define $W(b) =$ total weight of all edges to $rep(c)$ from $reps$ of grandchildren of c that are children of b.

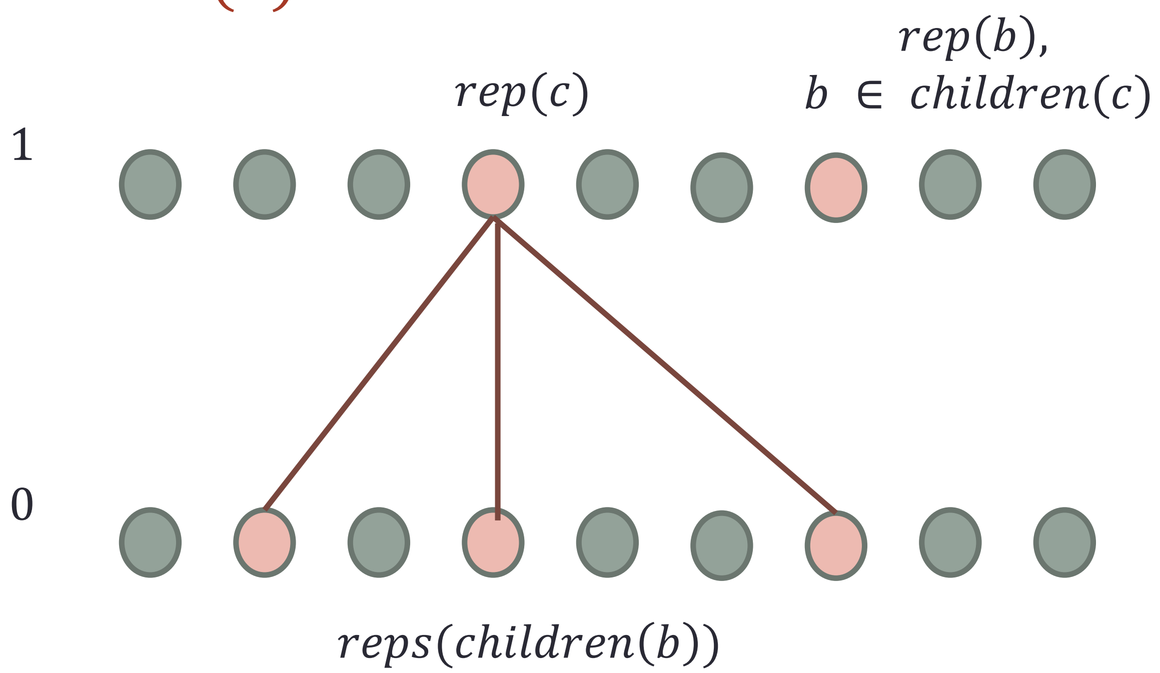
Define $W =$ total weight of all edges to $rep(c)$ from $reps$ of grandchildren of c, $= \Sigma_b W(b)$.

Base case: 2 levels, 1 layer

For each child b of c, define $W(b) =$ total weight of all edges to $rep(c)$ from grandchildren of c that are children of b.

Define $W =$ total weight of all edges to $rep(c)$ from reps of grandchildren of c, $= \Sigma_b W(b)$.

Illustration of $W(b)$:



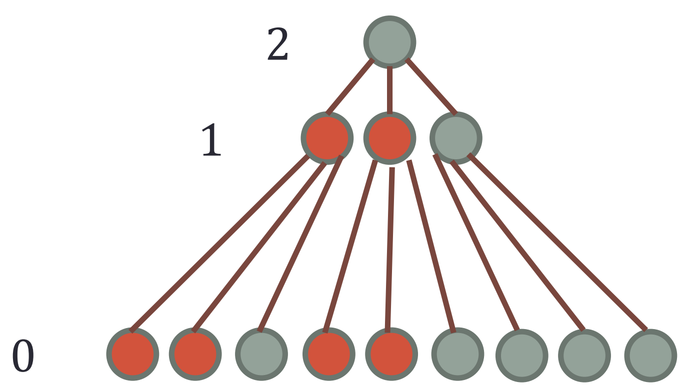
Base case: 2 levels, 1 layer

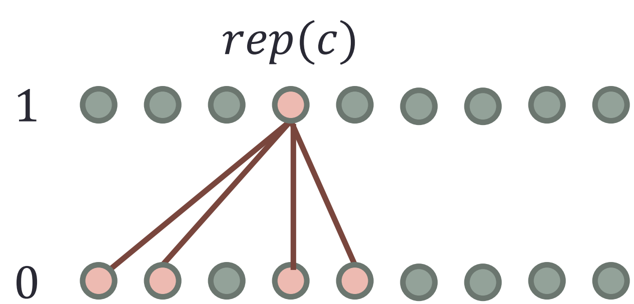
$W(b)$ $=$ total weight of all edges to $rep(c)$ from $reps$ of grandchildren of c that are children of b.

W $=$ total weight of all edges to $rep(c)$ from $reps$ of grandchildren of c, $= \Sigma_b W(b)$.

Scenario A ($rep(c)$ should fire): Choose B $= r_2$ fraction of c's children with smallest $W(b)$, and for each, its r_2 fraction of children with smallest weights.

Lemma: Total incoming potential to $rep(c)$ is $\leq r_2^2\, W$.





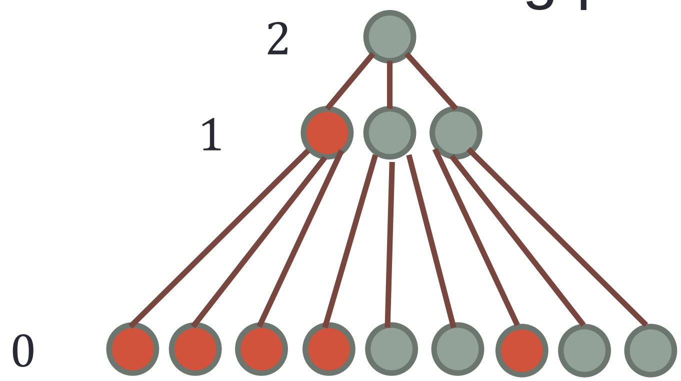
Base case: 2 levels, 1 layer

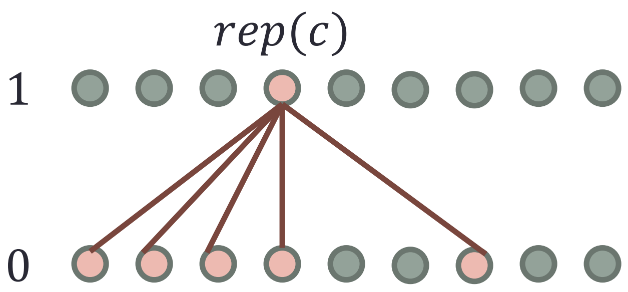
Scenario A ($rep(c)$ should fire): Choose $B = r_2$ fraction of c's children with smallest $W(b)$, and for each, its r_2 fraction of children with smallest weights.

Lemma: Total incoming potential to c is $\leq r_2^2\, W$.

Scenario B ($rep(c)$ should not fire): Choose $B = r_1$ fraction of c's children with largest $W(b)$, and for each, all of its children. For each other child of c, chose r_1 fraction of children with largest weights.

Lemma: Total incoming potential to c is $\geq (2\, r_1 - r_1^2)\, W$.





Base case: 2 levels, 1 layer

Scenario A ($rep(c)$ should fire): Choose $B = r_2$ fraction of c's children with smallest $W(b)$, and for each, its r_2 fraction of children with smallest weights.

Lemma: Total incoming potential to c is $\leq r_2^2 \, W$.

Scenario B ($rep(c)$ should not fire): Choose $B = r_1$ fraction of c's children with largest $W(b)$, and for each, all of its children. For each other child of c, chose r_1 fraction of children with largest weights.

Lemma: Total incoming potential to c is $\geq (2\, r_1 - r_1^2)\, W$.

So firing threshold of $rep(c)$ must be $\leq r_2^2 \, W$ and $\geq (2\, r_1 - r_1^2)\, W$.

Contradiction since we have assumed that $r_2^2 < 2r_1 - r_1^2$.

General case

Theorem 3 (Restated): Assume that the network *N* has fewer layers than the number of levels in the concept hierarchy *C*.
Assume $r_2^2 < 2r_1 - r_1^2$.
Then *N* does not (r_1, r_2)-recognize concept hierarchy *C*.

Note: Here we add a non-interference assumption:
Consider any level ℓ and any set B of level ℓ concepts in C.
For any $b \in B$, let $N(b)$ be the set of neurons at layers $\leq \ell$ whose firing is triggered by showing b.
Let N be the set of neurons at layers $\leq \ell$ whose firing is triggered by showing all the concepts in B together.
Then all the $N(b)$ sets are disjoint, and $N = \bigcup_{b \in B} N(b)$.

General case

Theorem 3: Assume that the network *N* has fewer layers than the number of levels in the concept hierarchy *C*.
Assume $r_2^2 < 2r_1 - r_1^2$.
Then *N* does not (r_1, r_2)-recognize concept hierarchy *C*.

Key Lemma: Suppose that network *N* (r_1, r_2)-recognizes concept hierarchy *C* (with non-interference assumption).
Then for every $\ell, 1 \leq \ell \leq \ell_{max}$, and for every level ℓ concept $c \in C$, $rep(c)$ appears in a layer $\geq \ell$.

Proof of Lemma: By induction on ℓ.

Inductive step: Assume a level ℓ concept c with $layer(rep(c)) \leq \ell - 1$.

General case

Lemma: Suppose that network *N* (r_1, r_2)-recognizes concept hierarchy *C* (with non-interference assumption).

Then for every $\ell, 1 \leq \ell \leq \ell_{max}$, and for every level ℓ concept $c \in C$, $rep(c)$ appears in a layer $\geq \ell$.

Proof: Assume level ℓ concept c with $layer(rep(c)) \leq \ell - 1$.

By I.H., all reps of $children(c)$ are at layers $\geq \ell - 1$, hence cannot influence the firing of $rep(c)$.

So again, we focus on c's grandchildren.

Define W, and $W(b)$ for each child b of c, as in the 1-layer proof, based on total weights incoming to $rep(c)$ that are contributed by showing grandchildren of c.

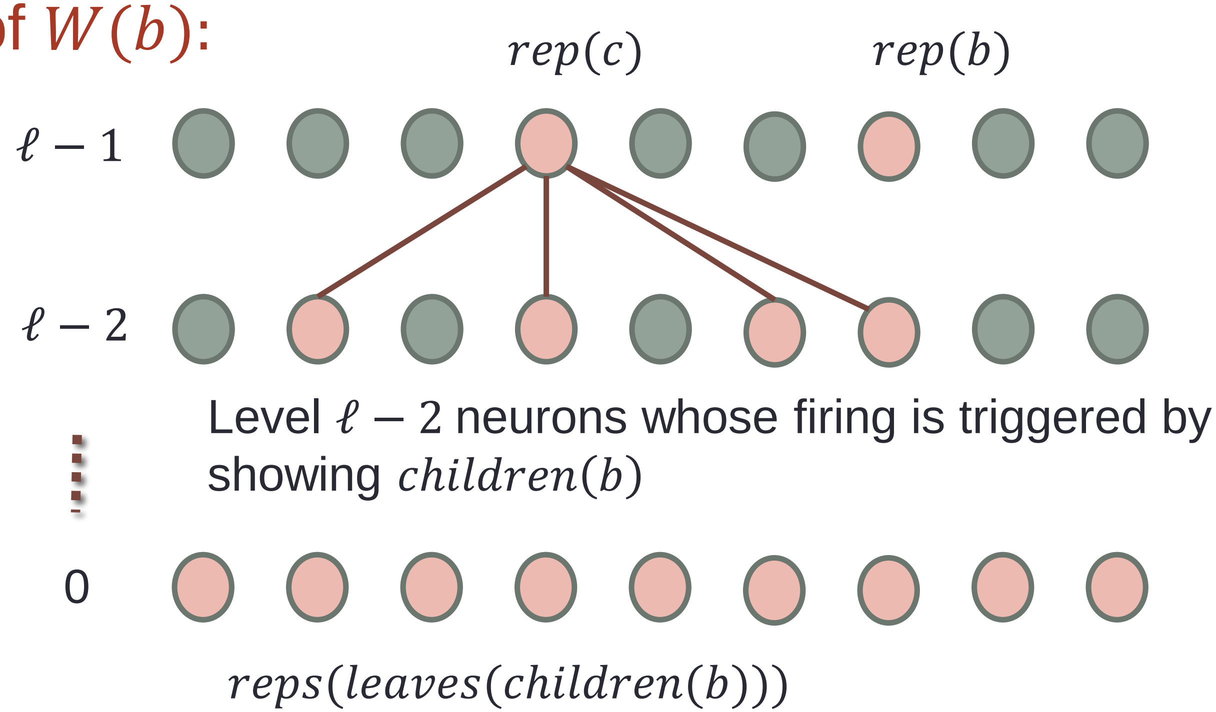
But now the contributions are from whatever layer $\ell - 1$ neurons they cause to fire, not necessarily just $reps$ of the grandchildren.

Then argue similarly to before.

General case

Proof: Define W, and $W(b)$ for each child b of c, based on total weights incoming to $rep(c)$ that are contributed by showing grandchildren of c.

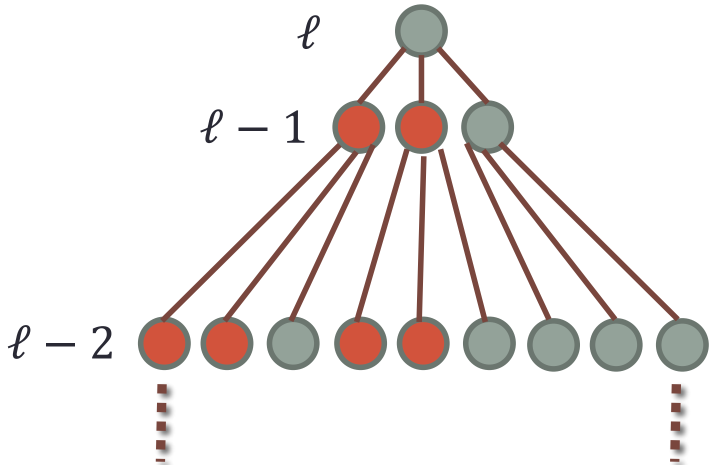
- Illustration of $W(b)$:

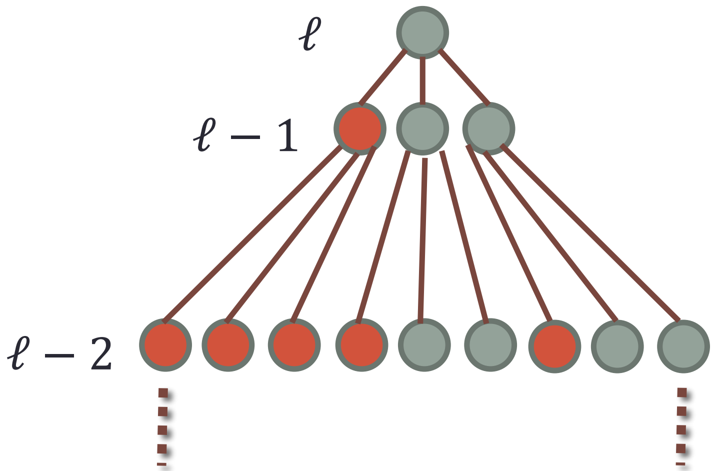


- Then argue similarly to before.

General case

- Define Scenario A ($rep(c)$ should fire even though few grandchildren are included), and Scenario B ($rep(c)$ should not fire even though many grandchildren are included).





- When we include a grandchild, present all its leaves.
- Reach the same contradiction as before, based on assuming $r_2^2 < 2r_1 - r_1^2$.
- Non-interference allows us to simply sum weights to account for contributions from multiple grandchildren.

Learning Hierarchically-Structured Concepts

1. Introduction
2. Data Model
3. Network Model
4. Problem Statements
5. Algorithms for Recognition and Noise-Free Learning
6. Extension: Noisy Learning
7. Lower Bounds
8. Conclusions

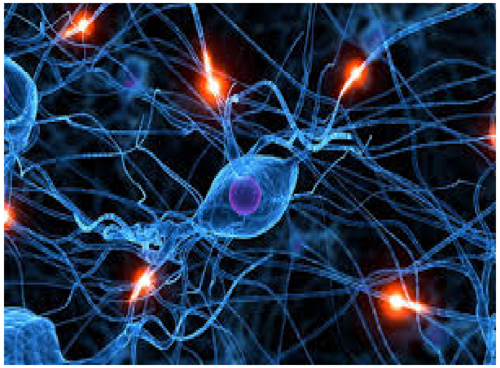
8. Conclusions

- Summary:
  - Hierarchically-structured concepts, based (initially) on a simple tree structure.
  - Noise-tolerant recognition problem.
  - Learning problem, leading to noise-tolerant recognition.
  - Learning algorithms, with/without noise during the learning process.
  - Lower bounds on number of layers, for noise-tolerant recognition.
- Discussion:
  - Gives some insight into how concepts with certain types of logical structure can be learned, and into limitations on networks that recognize such concepts.
  - Very simplified data model, needs many extensions.

Future Work on Learning Structured Concepts

- Different kinds of concept hierarchies, esp. with some overlap between child sets (DAG instead of forest).
- Different network structures, e.g., with sparse random connections instead of all-to-all, or with feedback edges.
- Learning different kinds of structure (logical relationships, geometric, physical).
- Different forms of representation (coding), not just single neurons.
- Strengthen connections with biology.

Other Future Work on Brain Algorithms



- Models
- Algorithms, for decision problems, neural representation problems, recognition, learning and recall.
- Representation of various kinds of concepts in the brain.
- Issues:
  - Role of randomness, inhibition.
  - Modularity.
  - Noise-tolerance, fault-tolerance.
  - To what extent can network mechanisms be learned, vs. pre-designed or evolved?
  - …