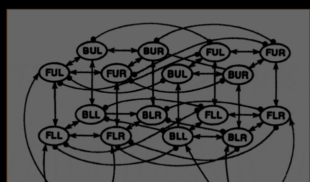
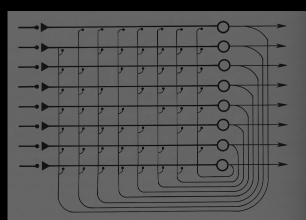
Lecture 3:

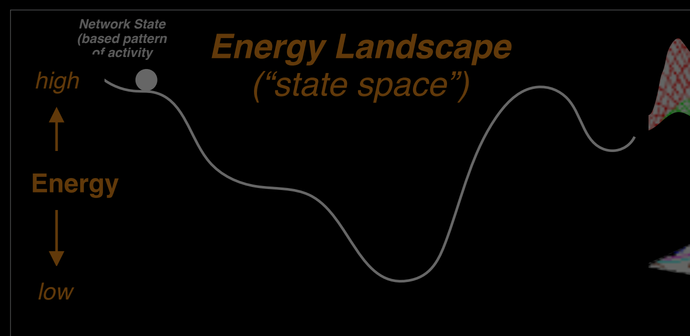
Associative Learning and Feature Maps

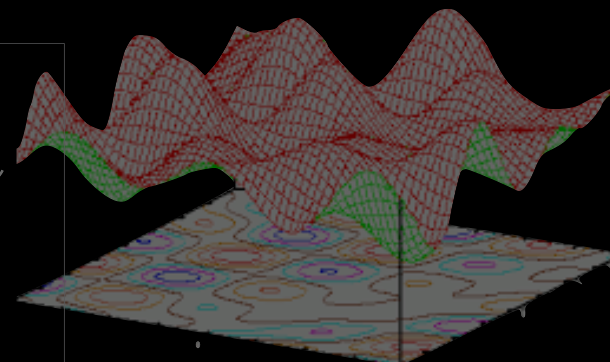
Learning

- **So far, we've focused on processing:**
  - **– dynamics of *encoding* and *representation* information** *(≈ weather)*









- **What about learning?**
  - **– how is the landscape shaped?** *(≈geology)*
  - **– dynamics of *acquisition***

Learning

- Unsupervised Learning
  - Hebbian Learning Rule
  - Self-organized maps
  - Topographic structure
  - Pattern associator
  - Pattern detectors
- Supervised Learning
  - Scalar Learning
    - Classical and Instrumental Conditioning
    - Sequential learning and Prediction
  - Vector-Based Learning
    - Generalized Delta Rule
    - Backpropagation
    - Deep Learning

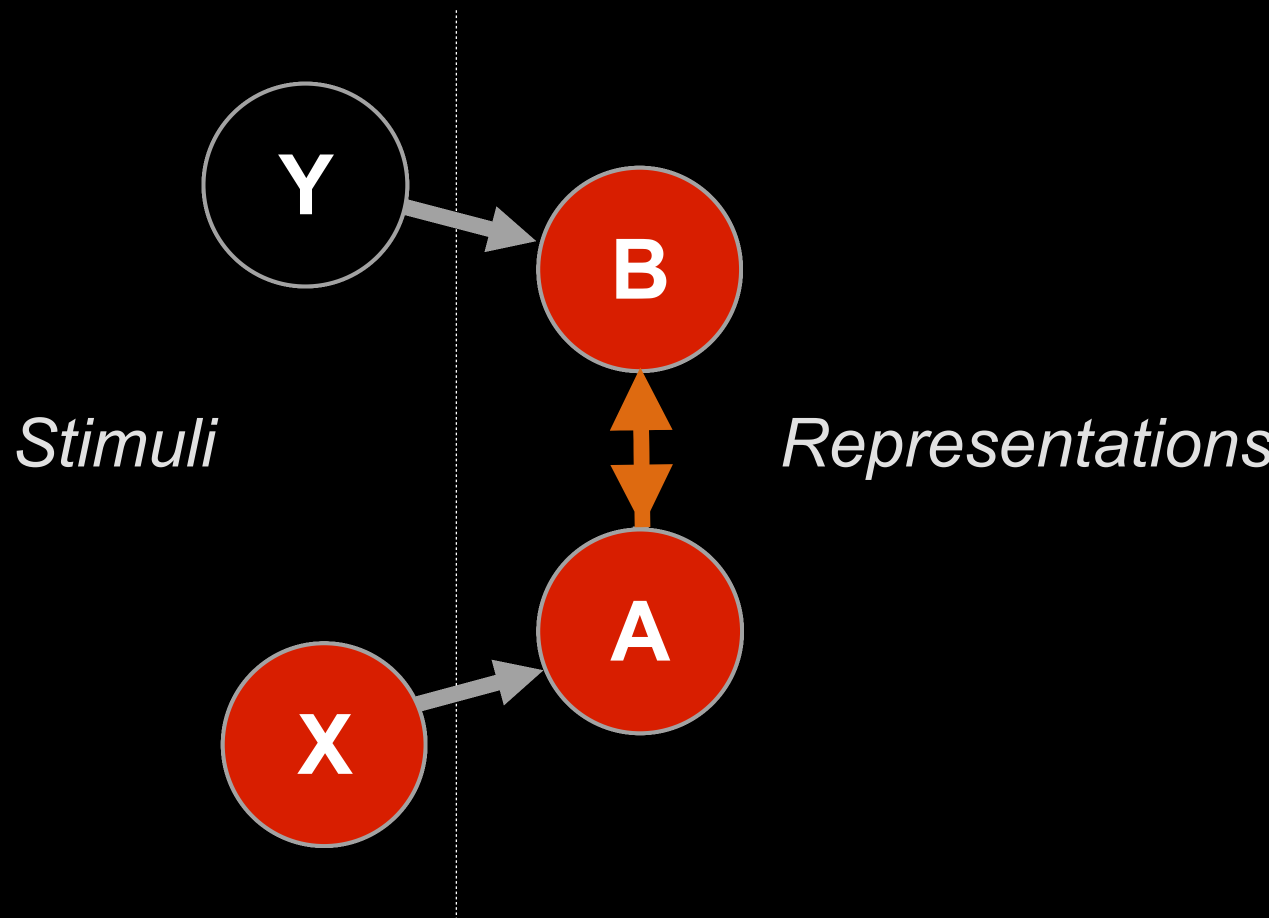
Hebbian Learning

- **D. O. Hebb:** *(1949)*
  "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

- **Critical factor**
  - **concurrent presynatptic and postsynaptic activity:** ***correlation***
  - **units that *"fire together wire together"***

- **Fundamental learning mechanism**
  - **responsible for much of how we gain our knowledge**

Hebbian Learning



Ac ...which strengthens ıd B

Hebbian Learning

Formalism:

$$\Delta w_{ij} = \alpha \, a_i a_j$$

where α is the learning rate and a can be any real number

After n "experiences:"

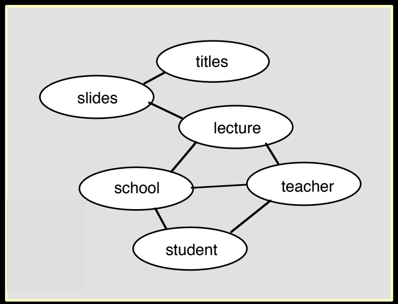
$$w_{ij} = \alpha \, \Sigma_n a_{in} a_{jn}$$

"Correlational Learning:

$w_{ij} \equiv$ correlation of a_i and a_j over time *(patterns)*

if a_i and a_j vary linearly from -1 to +1 (i.e., mean=0 and unit variance)

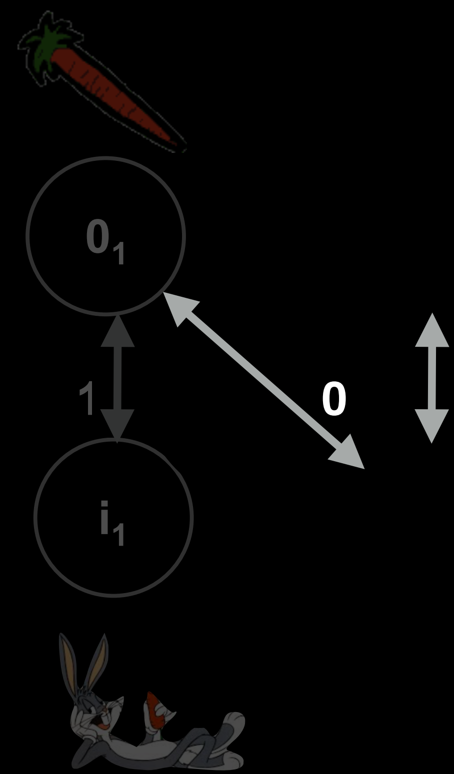
Captures statistical relationship among co-occurring features



Multiple Associations

	 i_1	 o_1
trial 1	+	+
trial 2	+	+
trial 3	−	−
trial 4	−	−

Correlated ***Uncorrelated*** ***Correlated***

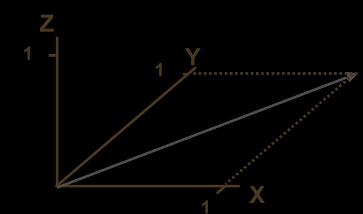


A Bit O' Math

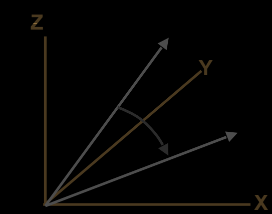
- Patterns can be considered as vectors (lists of activation values) and relationships between them described using linear algebra
- Normalized Dot Product (NDP) of two patterns a and b over n units:

  $$a \cdot b = (\Sigma_i a_i b_i) / n$$

- NDP combines measure of strength and similarity
  - Strength of pattern: vector length, normalized for # of elements
    - *Tip: this is the Euclidean distance from the origin to the point defined by the vector; (≈ hypotenuse of the triangle defined by the vector and its distance along each axis*



  - Similarity: correlation, independent of length
    - *Tip: this is the angle between the two vectors*
      0° = similar (+ correlation)
      90° = unrelated (0 correlation)
      180° = opposite (- correlation)



- Two patterns whose NDP = 0 are said to be "orthogonal"
  - *Tip: Vectors that are "perpendicular" in 3D space are orthogonal (compute the NDP for the x axis against the y axis); this is because they are uncorrelated*

Associative Learning and Internal Representations / Model Building

- The role of associative learning in model building
  - Correlations are important for building internal models of the world:
    - the world is inhabited by objects and agents with features that are in consistent relationship to one another
    - these regularities are useful for identification and prediction (predators have fangs; when it is warm fruit will be available; types of faces)
    - correlations among features define dimensions that are relevant for and efficient at describing and understanding the world
- Extracting regularities is a fundamental job of cognition:
  - Parsimony/ Abstraction: can describe a complex world with finite resources
  - Generalization: infer properties of the world in novel circumstances
  - Efficiency of learning: can represent novel items with existing codes

Pattern Detector

- **Formalism:**
  - Detector unit y receives connections from a set of input units x_k
  - Activation of detector unit:

    $$y_j = \Sigma_k x_k w_{kj}$$

  - Weight change between x_i and y_j over a set of n input patterns t

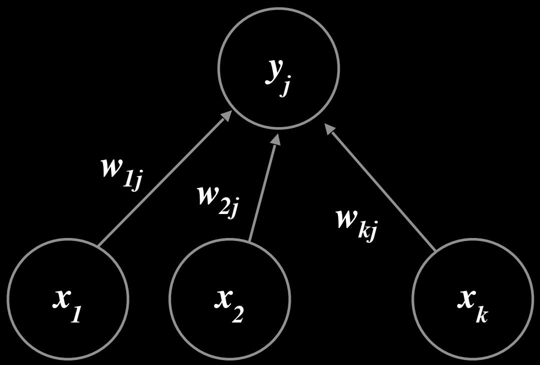
    $$\Delta w_{ij} = \varepsilon \Sigma_t x_{it} y_{jt}$$

  - If $\varepsilon = 1/n$, then

    $$\Delta w_{ij} = <x_i y_j>_t$$ (average product, or "expected value," of $x_i y_j$ over t)

  - Substitute for y_j and some algebra:

    $$\Delta w_{ij} = \Sigma_k <x_i x_k>_t <w_{kj}>_t$$

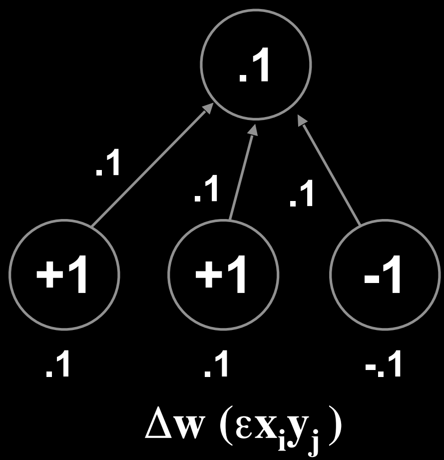


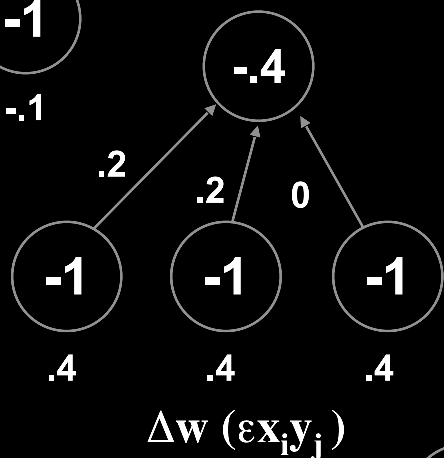
- **In words:**

Changes in the weight from input unit X_i to the detector y_j are a weighted average of the correlations that X_i exhibits with the other input units X_k in the network

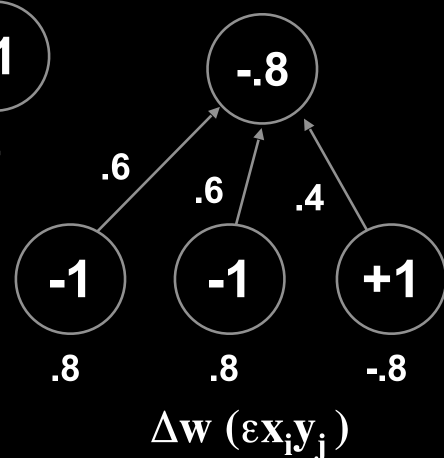
Net effect: weights will adjust to produce the greatest variance in y, by responding to "conspiracies" of correlated input units

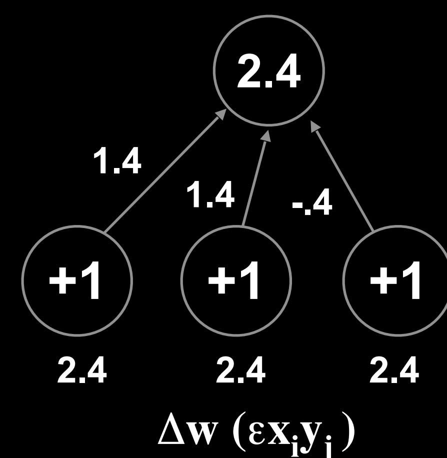
Example





- **Observe:**
  - Units 1 and 2 are highly correlated across the input patterns
  - Their weights consistently grow
  - The weight for unit 3 "thrashes" and, on average, goes nowhere
  - Weights adjust to produce the greatest variance in *y,* by responding to the fact that the *combined* influence of 1 and 2 is strong





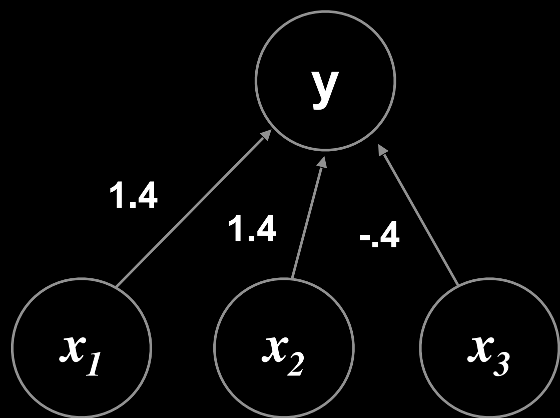
Initial weights: .1

Training Patterns:

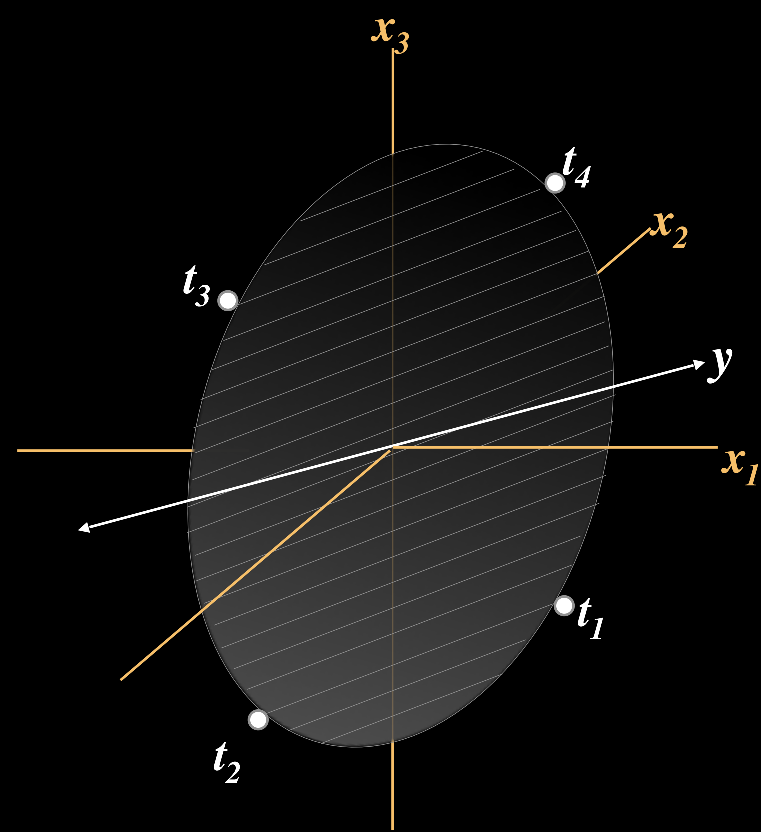
t_1 +1 +1 −1

Principal Components Analysis (PCA)

- Unit *y* extracts the principal Eigen Vector (i.e., one with the largest Eigen Value)



t_1	+1	+1	−1
t_1	−1	−1	−1
t_1	−1	−1	+1
t_1	+1	+1	+1

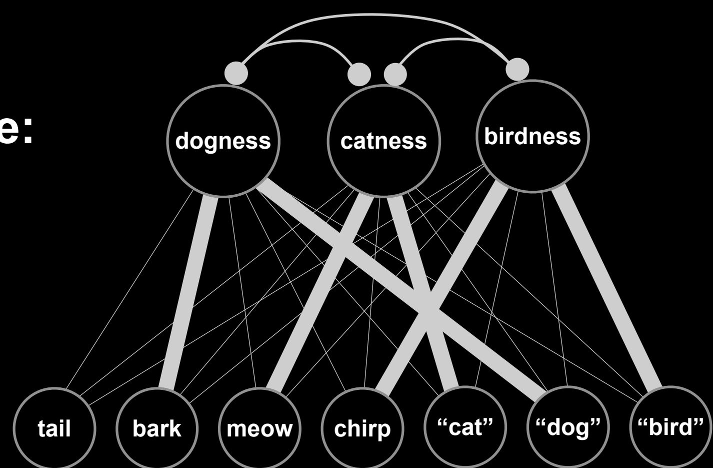


Principal Components Analysis (PCA)

- Unit y_i extracts the principal Eigen Vector (i.e., one with the largest Eigen Value)

- **If we have multiple detector units y_j, we can extract additional components**
  - **Lateral competition required to prevent redundancy**
    (otherwise all detector units would encode the same principal component)

Example:



Cat	+1	−1	+1	−1	+1	0	0
Dog	+1	+1	−1	−1	0	+1	0
Bird	+1	−1	−1	+1	0	0	+1

Principal Components Analysis (PCA)

- Unit y_i extracts the principal Eigen Vector (i.e., one with the largest Eigen Value)
- If we have multiple detector units y_j, we can extract additional components
  - Lateral competition required to prevent redundancy (otherwise all detector units would encode the same principal component)
  - Schemes can be devised to enforce orthogonality of components
    = Standard hierarchical PCA
  - However, other schemes (e.g., weight normalization) provide mechanisms of parallel ("heterarchical") PCA:
    - encourages detectors to specialize for different features
    - better fit to structure of real world
      (world is not hierarchically arranged)

Other Approaches

- **Linsker's Information Maximization**
  - **Multiple detector units, similar to PCA network:**
    maximizing variance in output units ≈ maximizing information
    (in limit not useful, since no dimension reduction ∴ no generalization)
- 👉 **Kohonen Network**
  - **Multiple detector units with structured local connections among them: captures neighborhood relationships among features; topographic maps**
    (Ken Miller's simulations of ocular dominance columns)
- **Competitive Learning (winner-take-all)**
  - **Multiple detector units but only one allowed to be active;**
    forces different detectors to identify different correlations among input units
- **Minimum Description Length (K-winners-take-all)**
  - **Similar to competitive learning, but a small set of detectors can be active;**
    trades off maximizing information against minimizing complexity
- **LEABRA**
  - **Combines K-winners-take-all competition with error-driven learning**

More Generally…

- Can think of associative networks as implementing "exploratory" analysis of environment

- Parameterization implements different classes of statistical functions

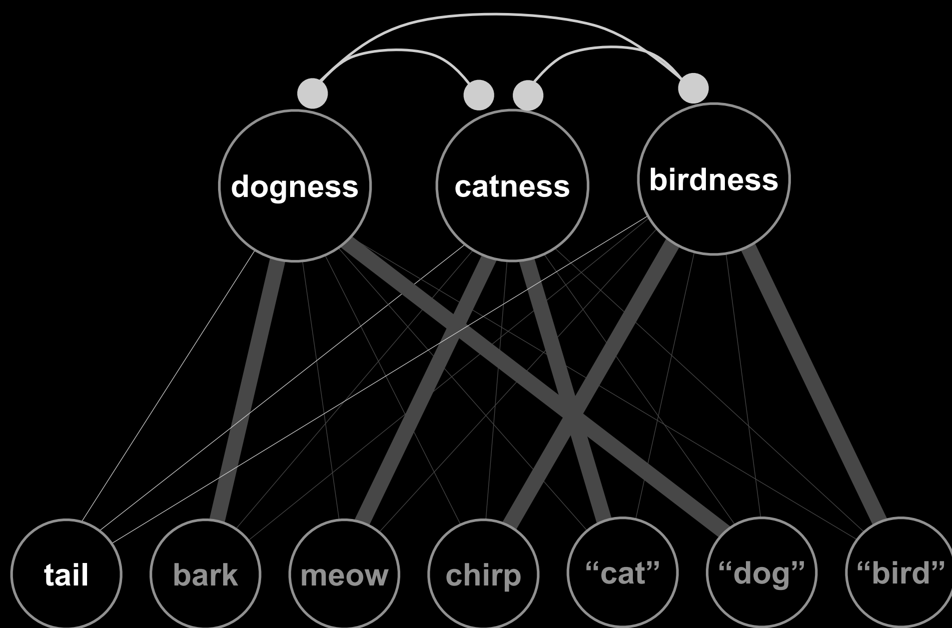
Limitations of Associative Learning
and Some Solutions

- Recalls each test pattern as a weighted function of its similarity to ones that it has learned: *blends*, doesn't make *"decisions"*
  - Recurrent connections + non-linear units → settling processes:
    - auto-associator
    - attractor networks

- Weights *unbounded* and *never decrement*
  - Weight normalization
  - Weight decrements for non-correlation: Long-Term Depression (LTD)

- Pattern associator can only learn *orthogonal representations*; pattern detector restricted to *linear* correlational structure
  - Error-driven learning
  - Example of problem...

Example



Cat	+1	−1	+1	−1	+1	0	0
Dog	+1	+1	−1	−1	0	+1	0
Bird	+1	−1	−1	+1	0	0	+1

Observe: tail is active for *all* of the animals (no variance) so it doesn’t correlate with any of the other animal features and therefore is not part of their representation

Summary

- Associative (Hebbian) learning provides a biologically plausible mechanism for setting weights in a network
  - Relationship to Long Term Potentiation (LTP)
- Hebbian pattern associators can learn relationships between features of the world
  - patterns constrained to be orthogonal
- Hebbian pattern detectors can represent correlational structure
  - implement various forms of PCA
- Basic Hebbian rule needs augmentation
  - Weight decay (LTD), normalization (competition), etc.
- Even still, important behaviors that it can explain...

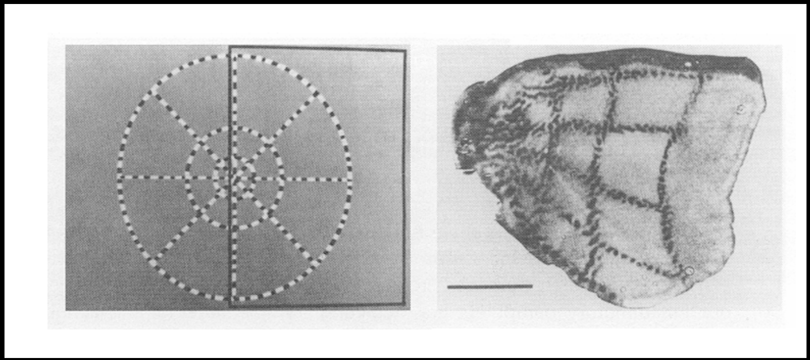
Topographic Organization

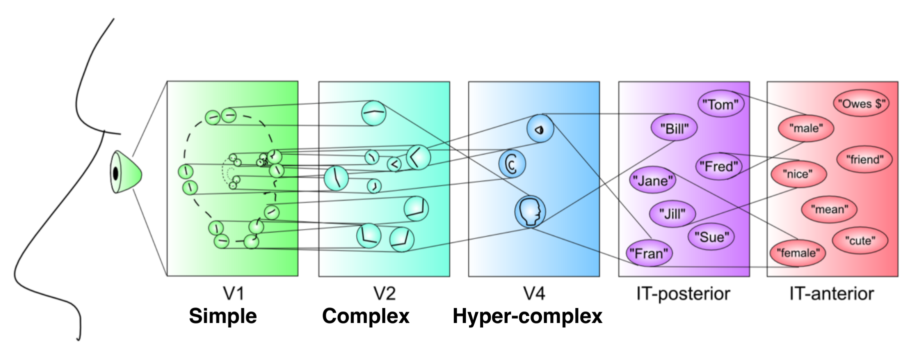
- **Associative learning can extract structure in the world, and represent it *structurally* (topographically)**

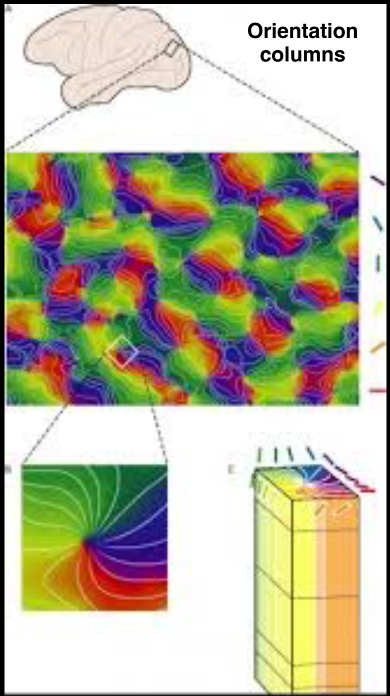
- **There is (lots of) topographic organization in the nervous system:**
  - **Retina (spatiotopic), inner ear (tonotopic), sensory and motor cortex**
  - **Exploited for imaging** *(e.g., retinotopic mapping of primary visual cortex)*
  - **Even as it gets more complex, some topography is maintained:**
    - **Occular dominance columns** *(Miller, 1989)*
    - **Ocular dominance, orientation and retinotopic positions "pinwheels"** *(Durbin & Mitchison, 1990)*

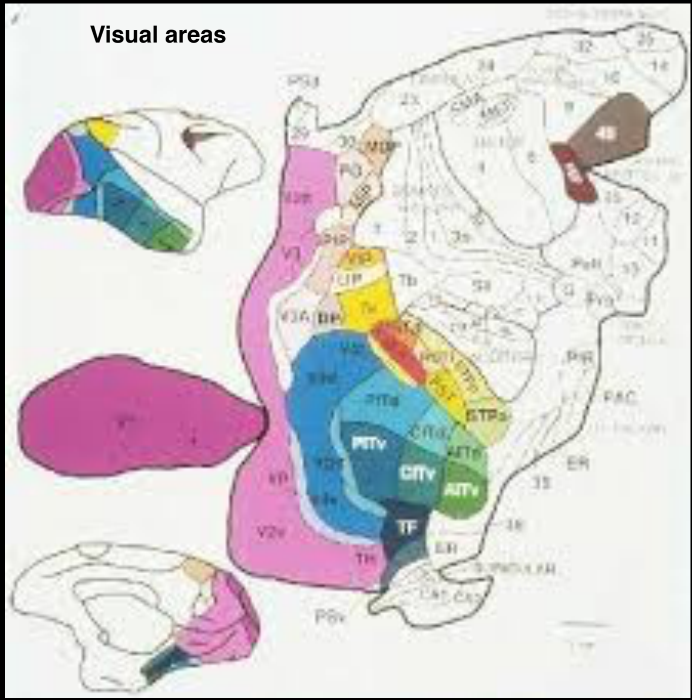
- **These may reflect meaningful relationships**  **xist in the "data"** *(i.e., the "real world")*

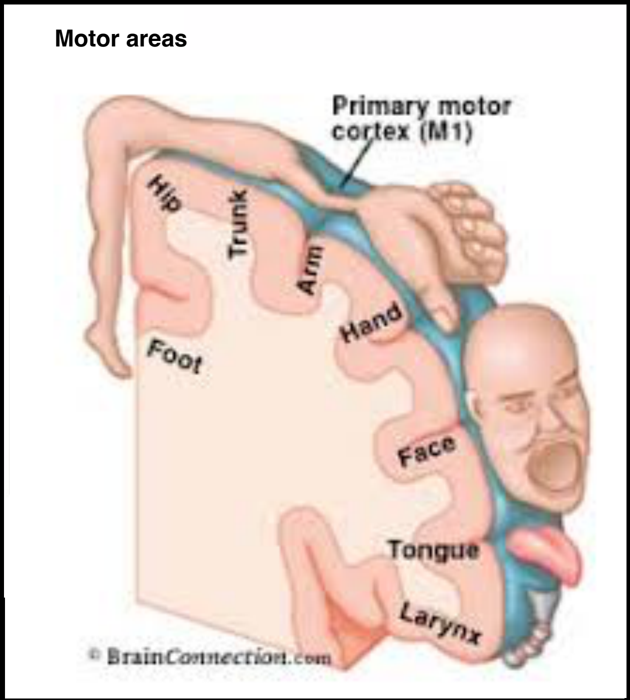
Topographic Organization





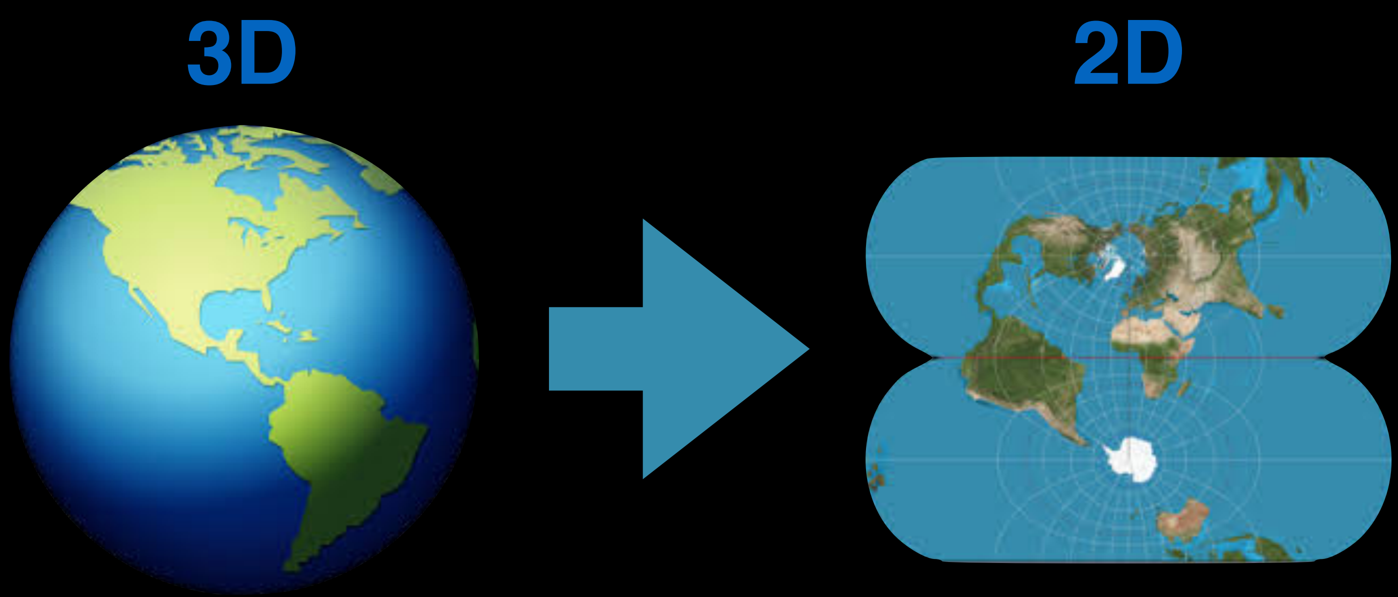






“Dimension Reduction”

- How does this structure arise?
- Challenge:



- What about even higher dimensional data?

Self-Organizing Maps (SOMs)

Kohonen Network (1982)

- **Objectives:**
  - Map input vectors (patterns) of dimension *N* onto a map with 1 or 2 dimensions.
  - Patterns *close* to one another in the input space should project to *nearby* units ("map" should be *topographically ordered*)
- **Network architecture and input environment *("training")***
  - Input layer:
    - units that code a space of *vectors with structure*, but *not spatially arranged*
  - Output layer:
    - each unit *j* has *weights from all units* in input layer
    - each unit *j* has a *defined distance* from all other units in the output layer

j

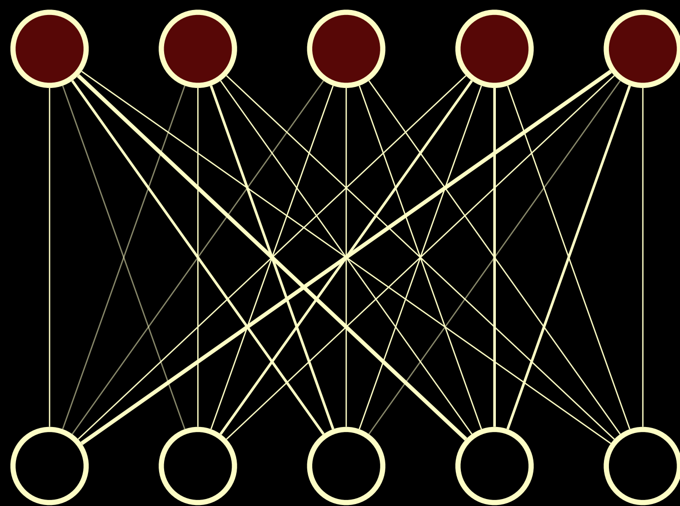
- **Learning rule:**
  - present input pattern, and identify *best matching* (most active) *output unit:*
    - one with input *current weights closest to input pattern* *("winner" of lateral competition)*
  - **adjust weights for that unit using following rule:** $\propto$ **correlation of output unit with pattern of activity over input units**

$$W_{b(t+1)} = W_{b(t)} + \boldsymbol{c}_{wb(t)} \cdot \boldsymbol{g}(t) \cdot (I - W_{b(t)})$$

*change in weights to **b*** *closeness to **b*** *gain* *difference from Input pattern*

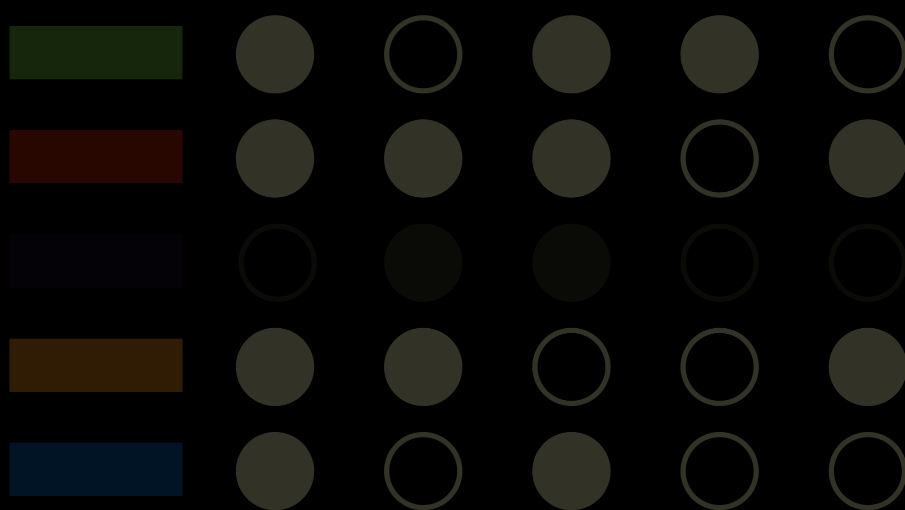
Self-Organizing Maps

Network



Small random initial weights

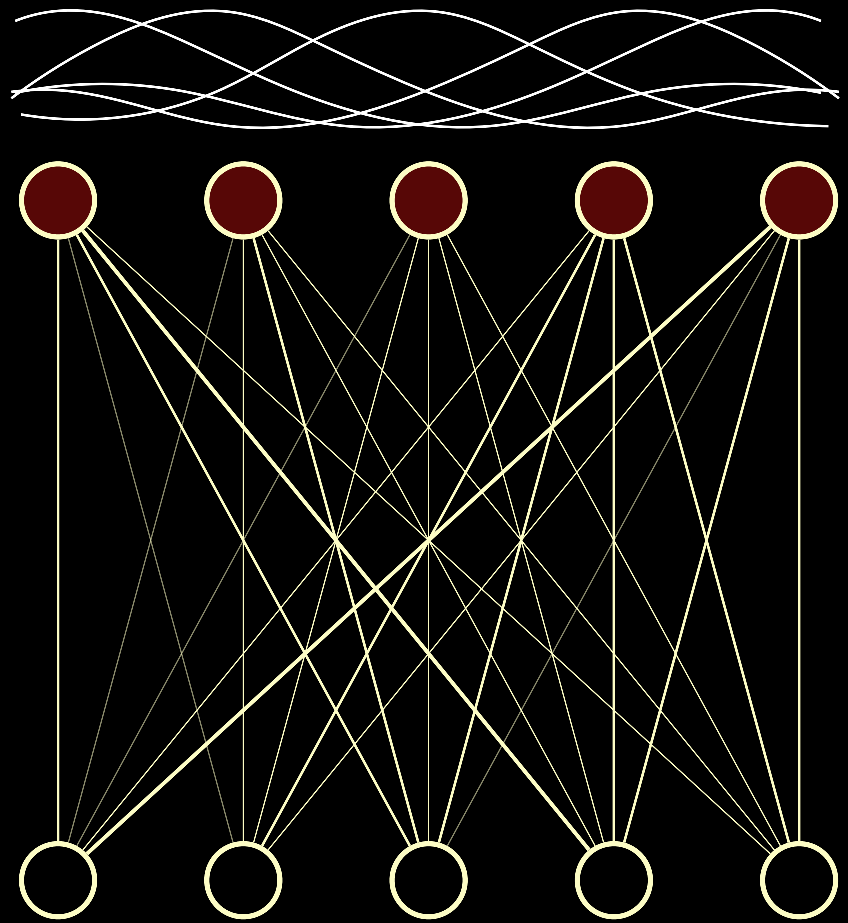
Input Patterns



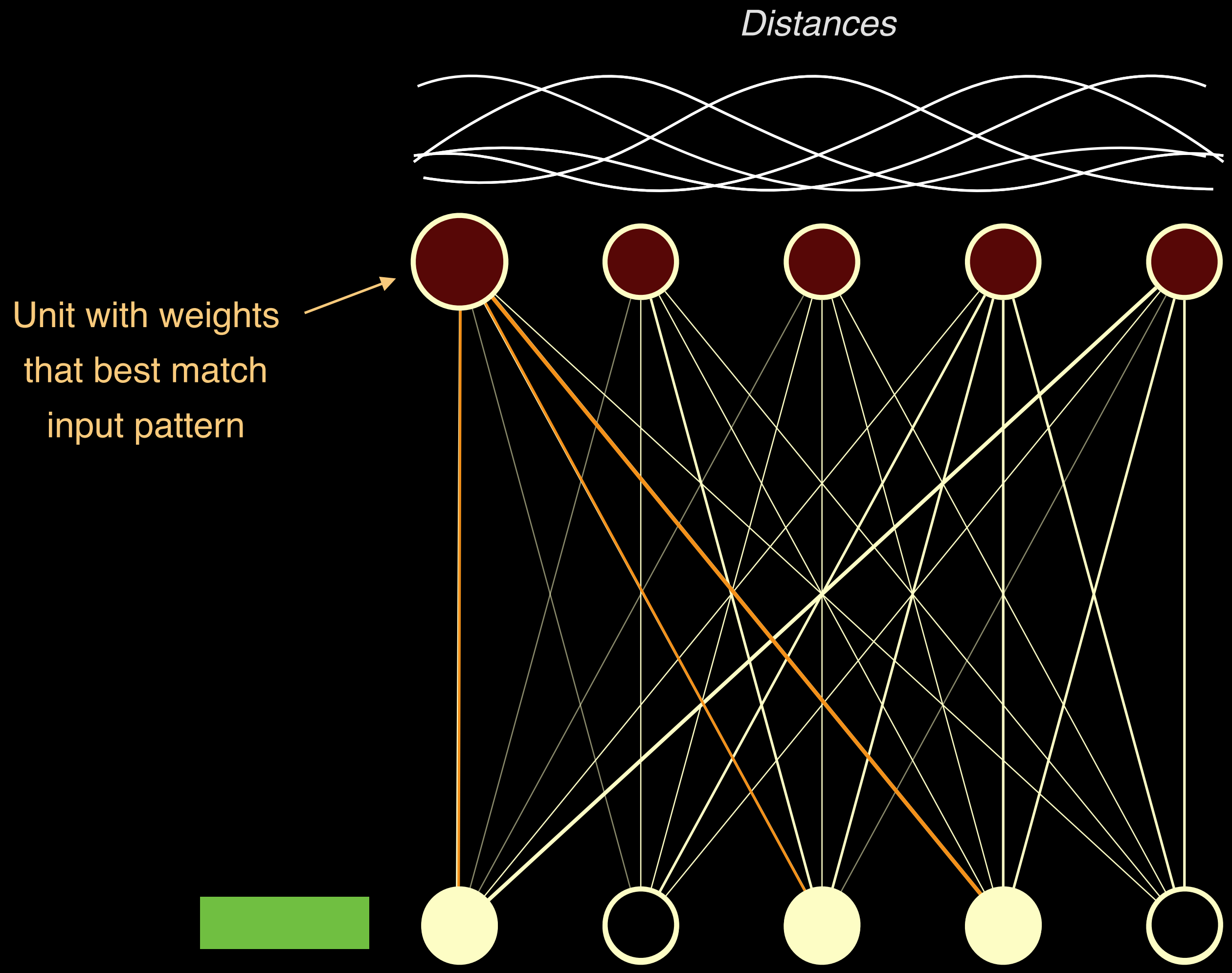
Similar colors have similar patterns

Self-Organizing Maps

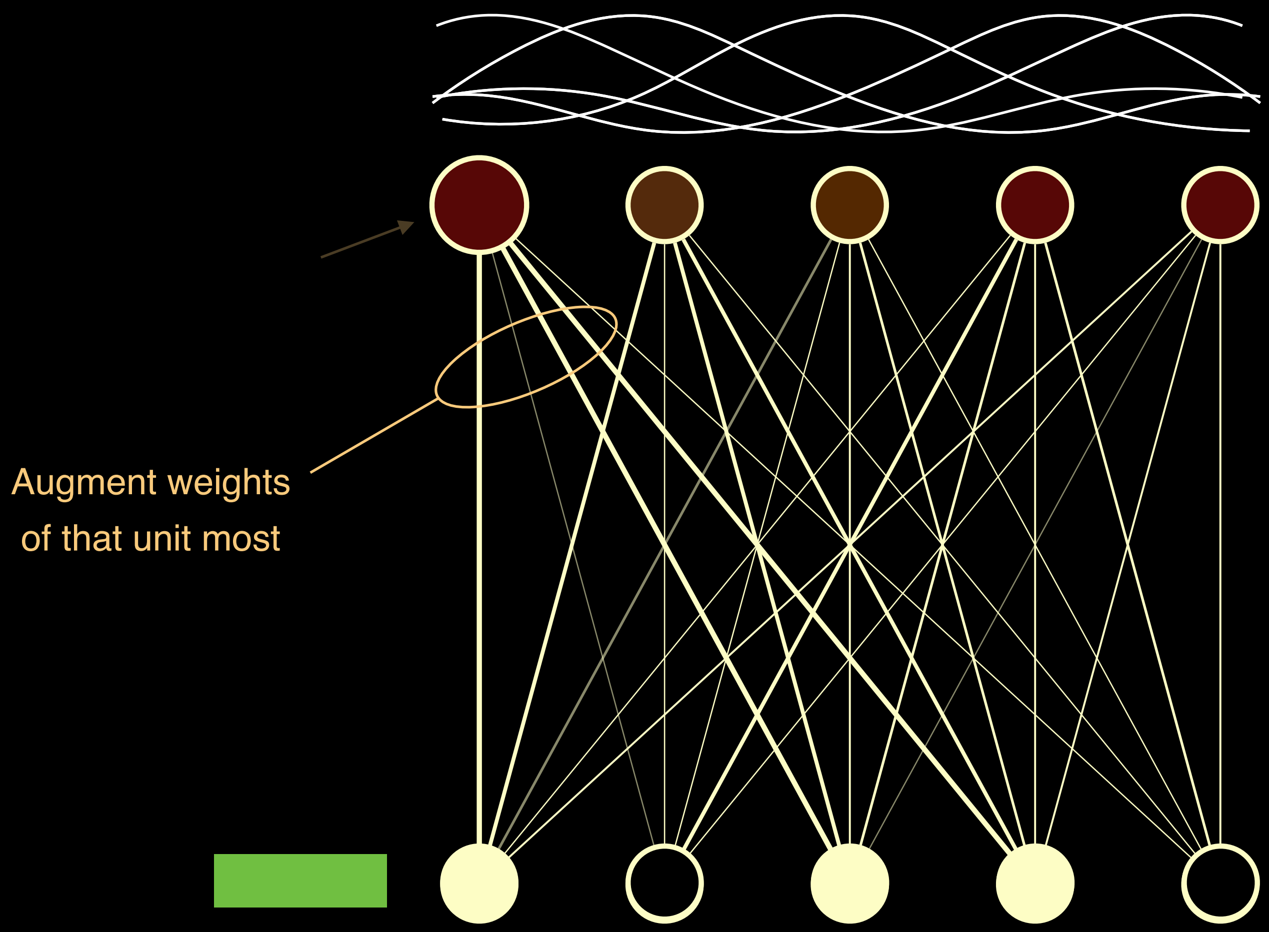
Distances



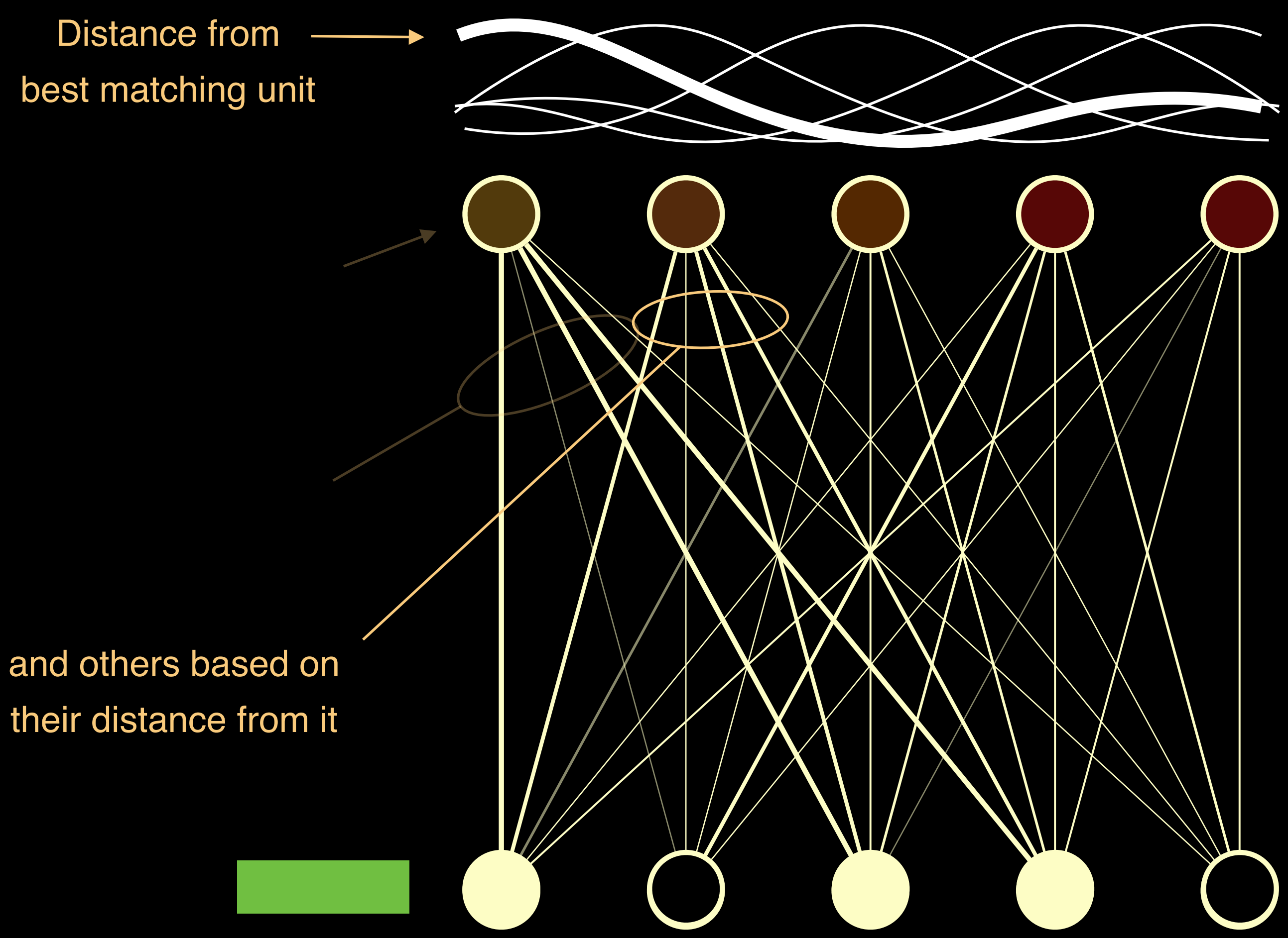
Self-Organizing Maps



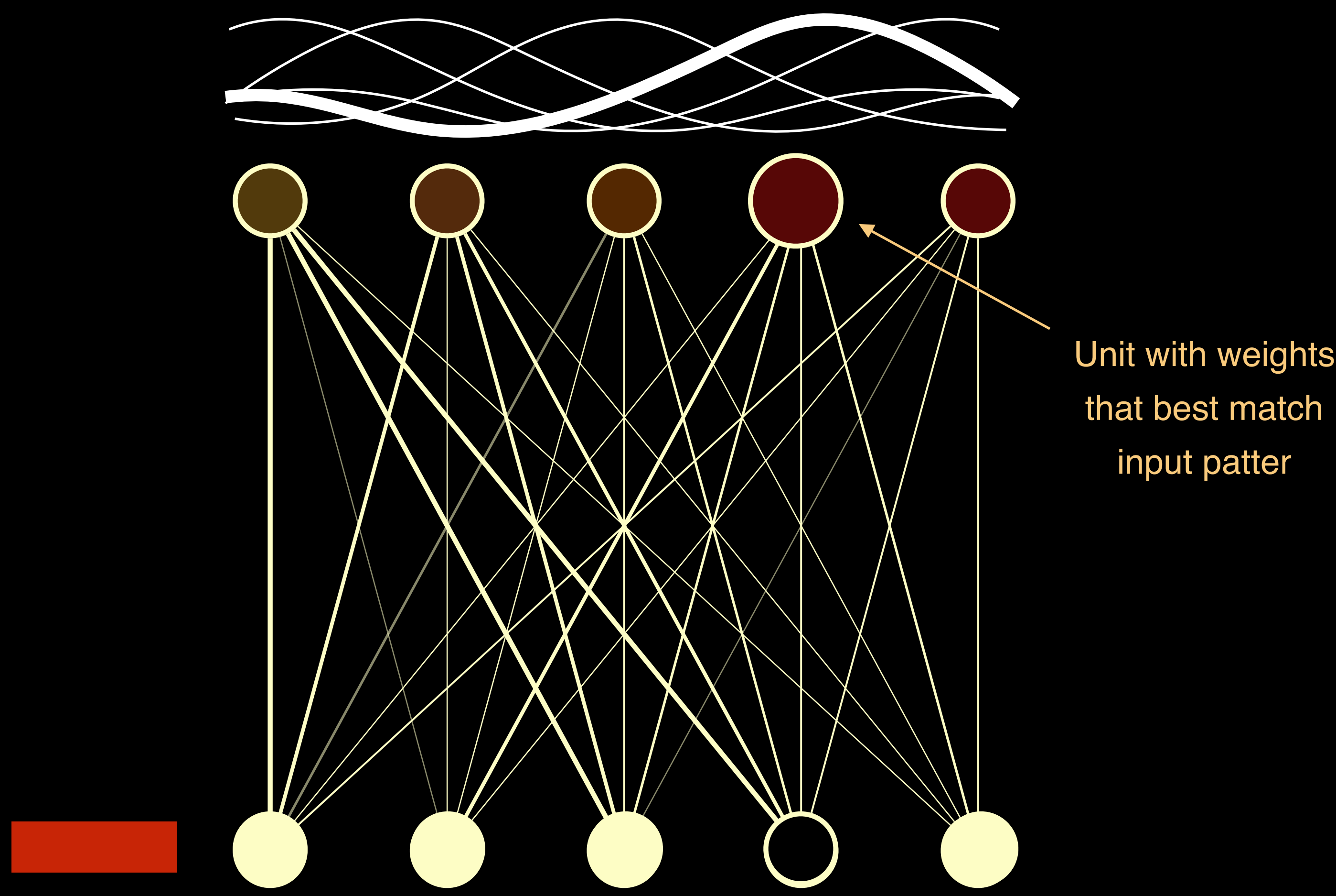
Self-Organizing Maps



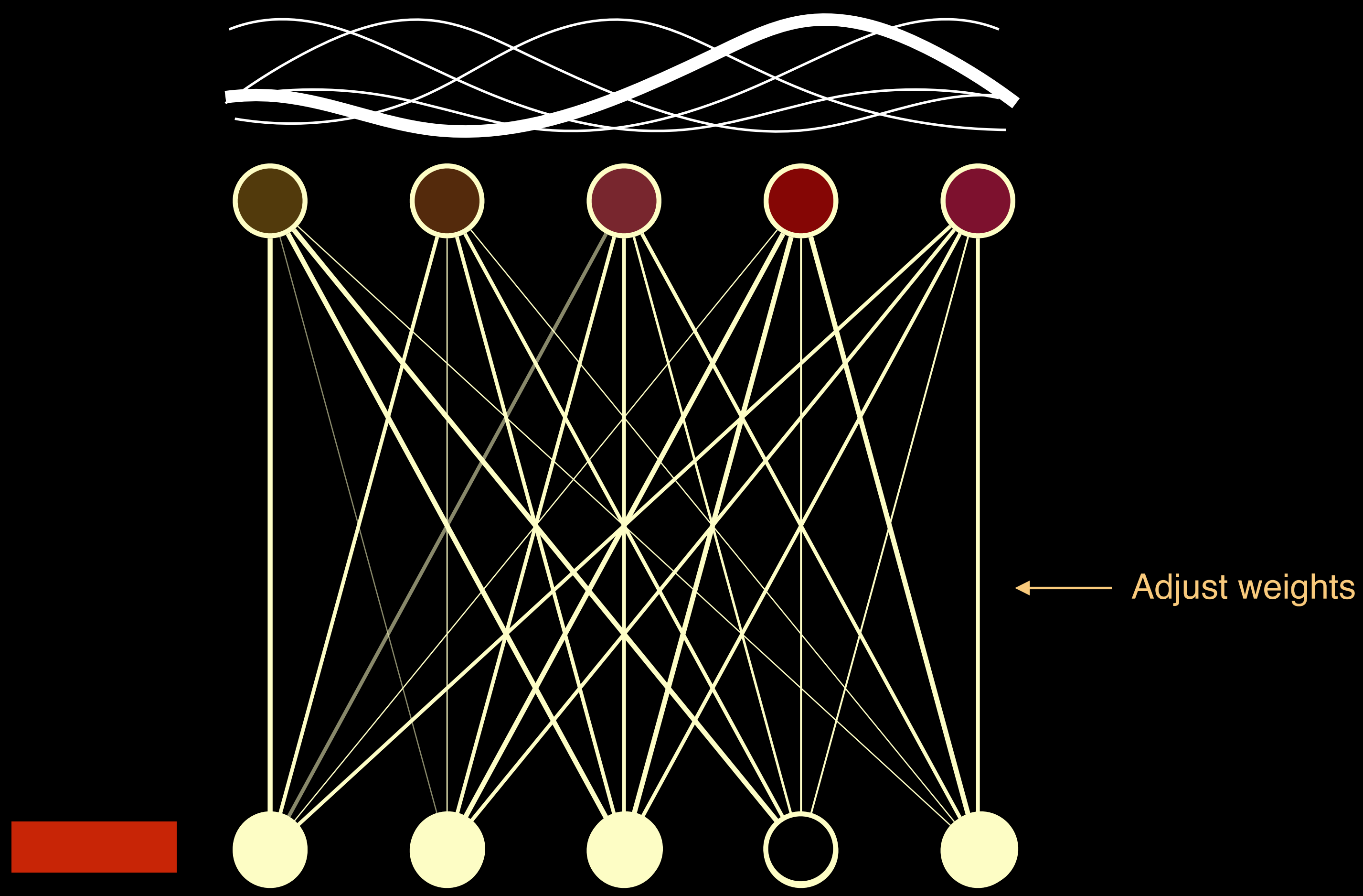
Self-Organizing Maps



Self-Organizing Maps

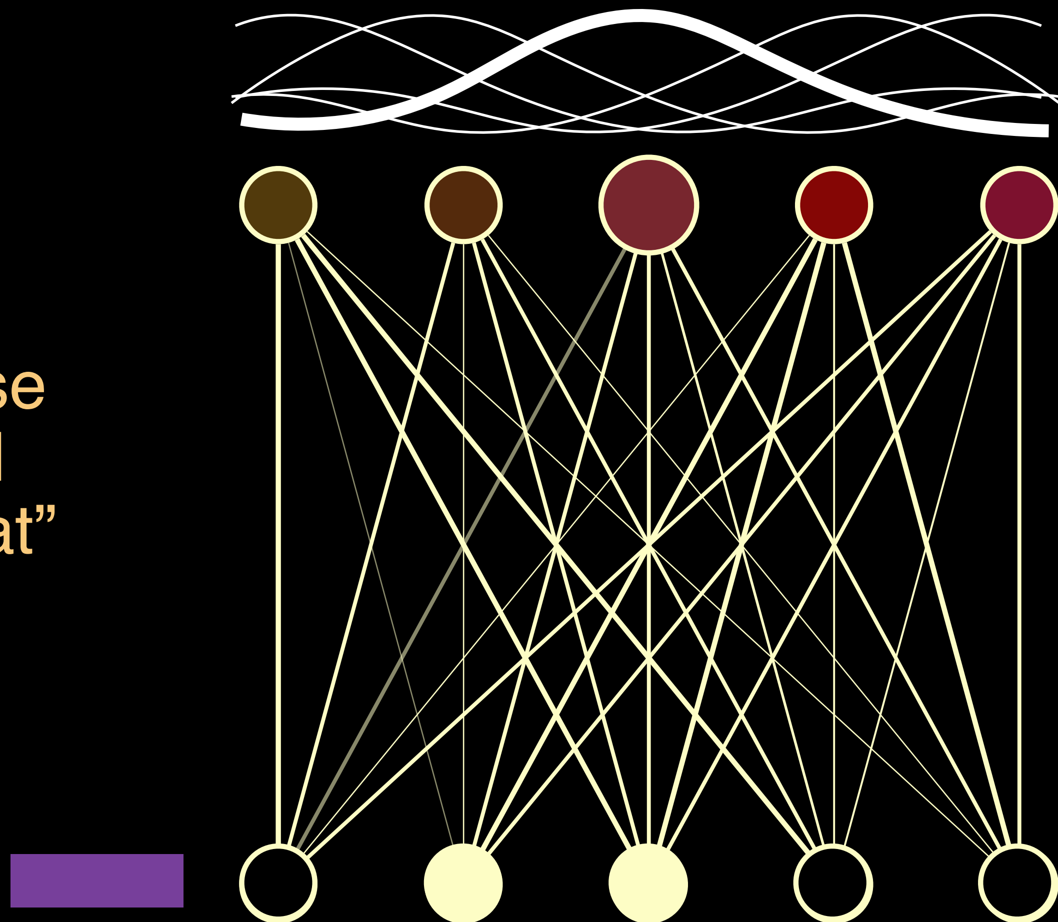


Self-Organizing Maps

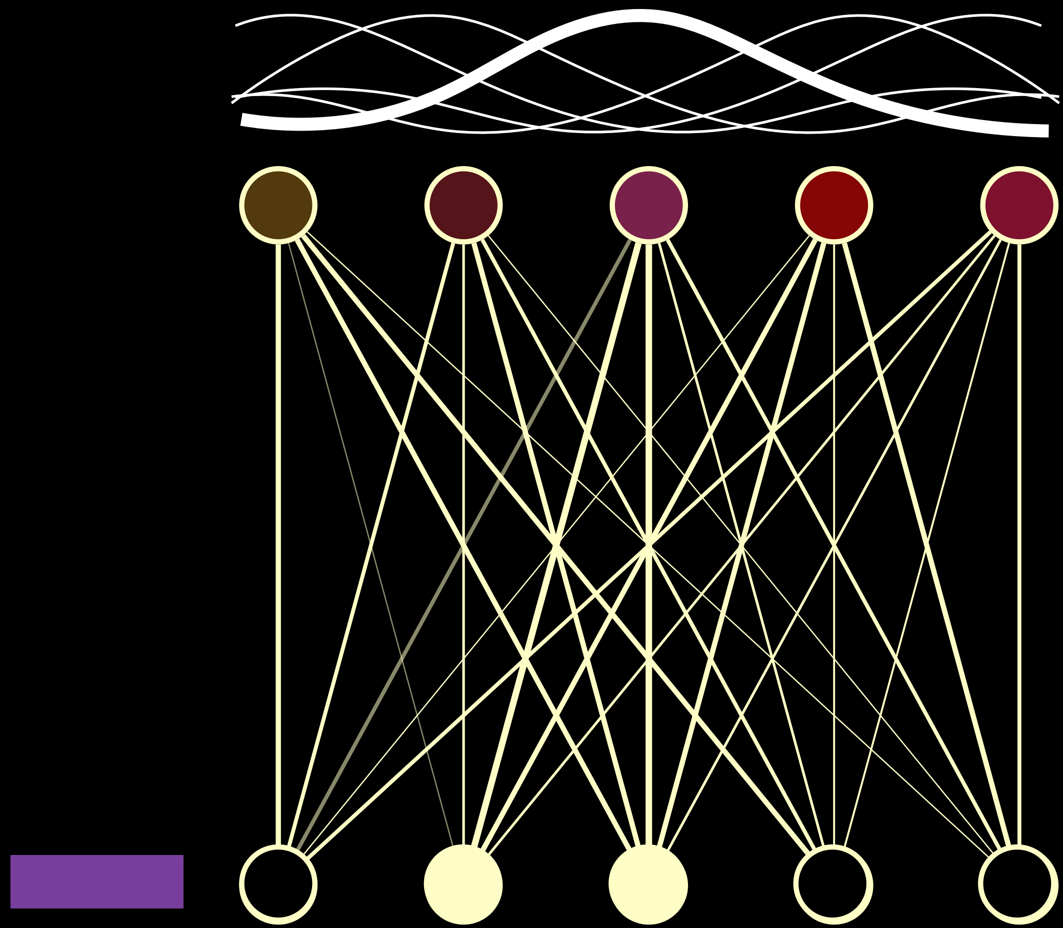


Self-Organizing Maps

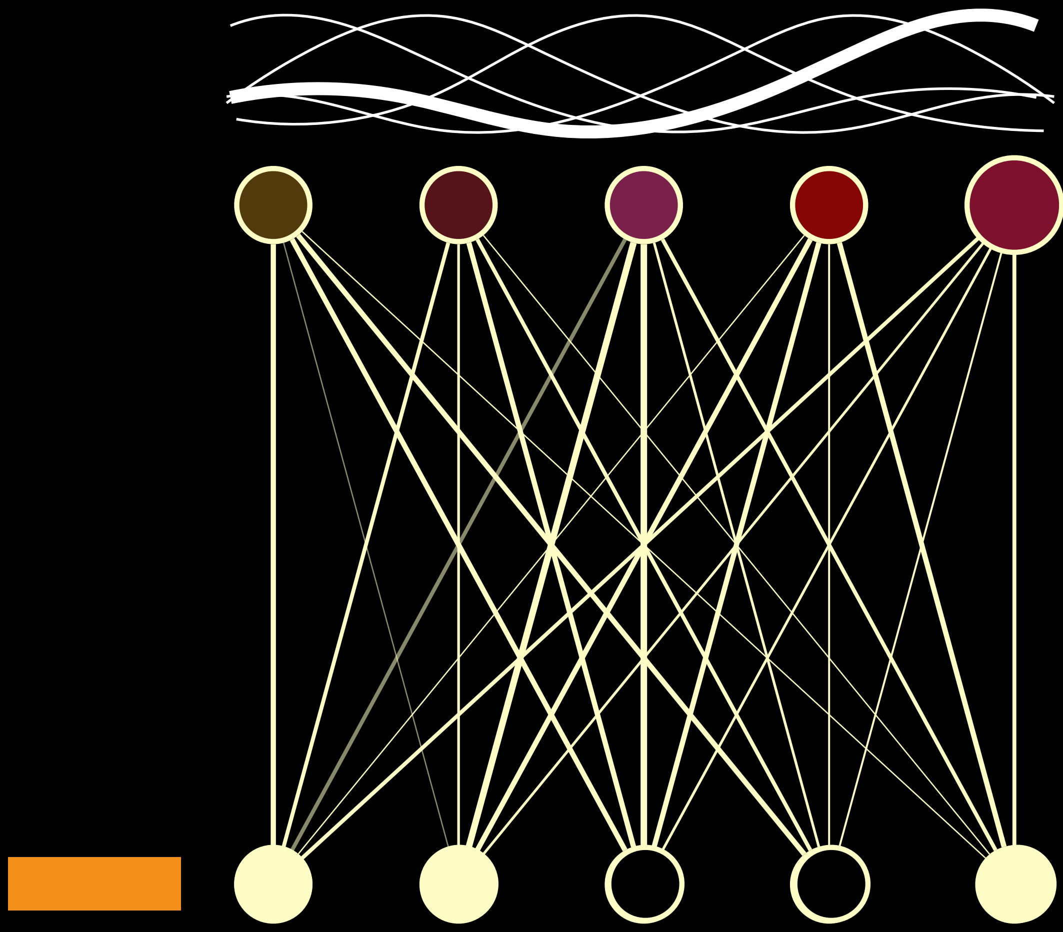
“Rinse
and
repeat”



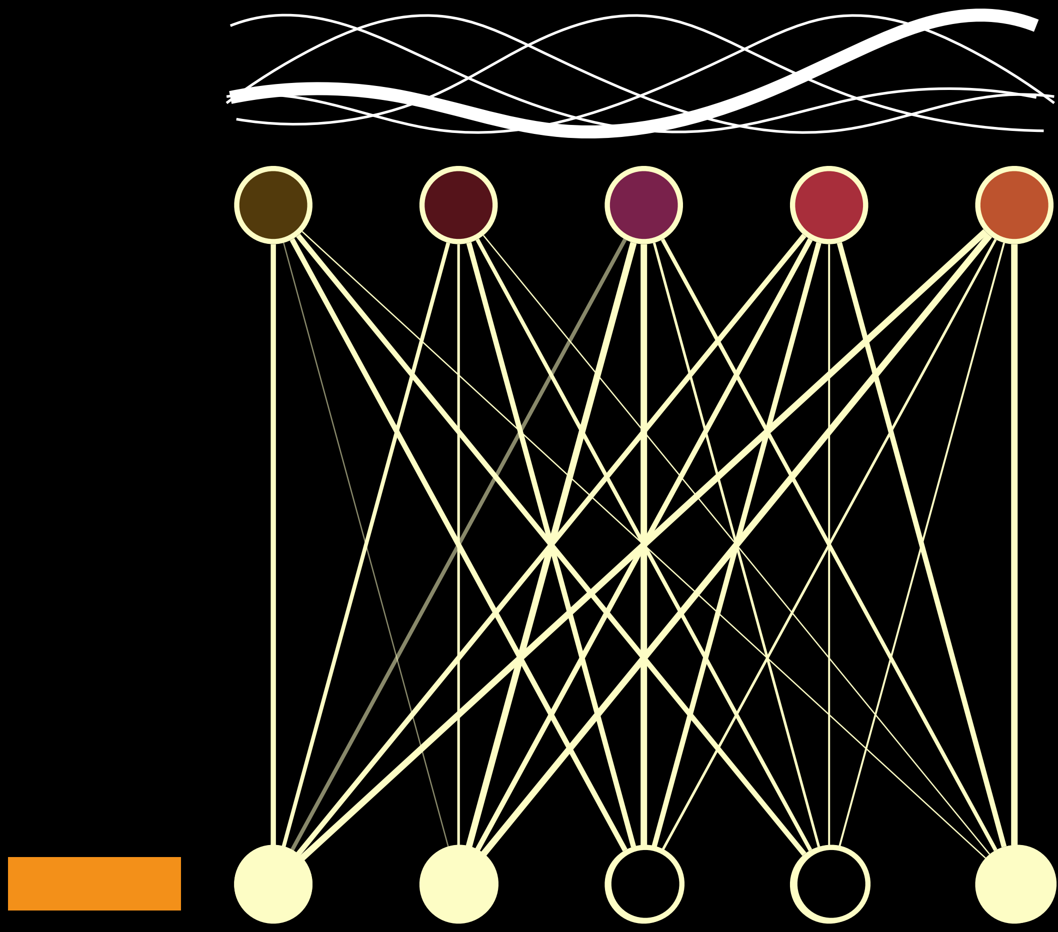
Self-Organizing Maps



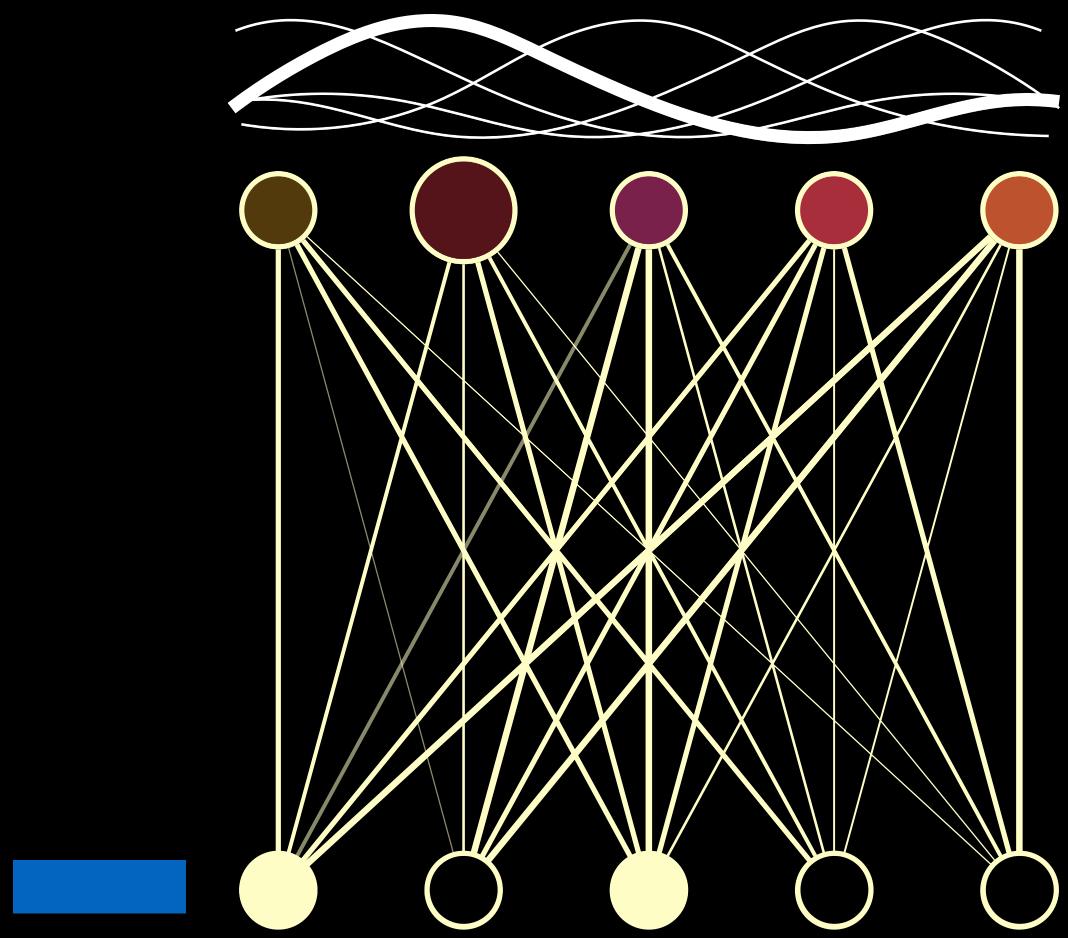
Self-Organizing Maps



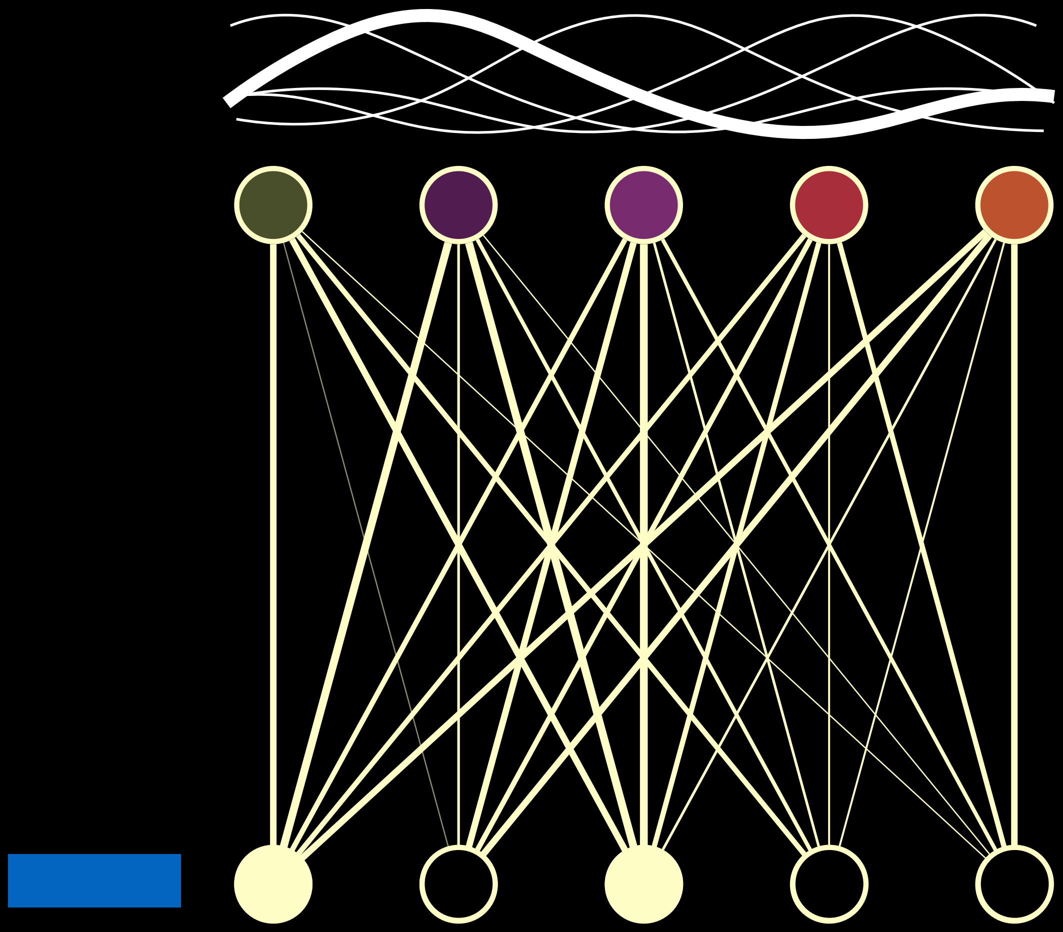
Self-Organizing Maps



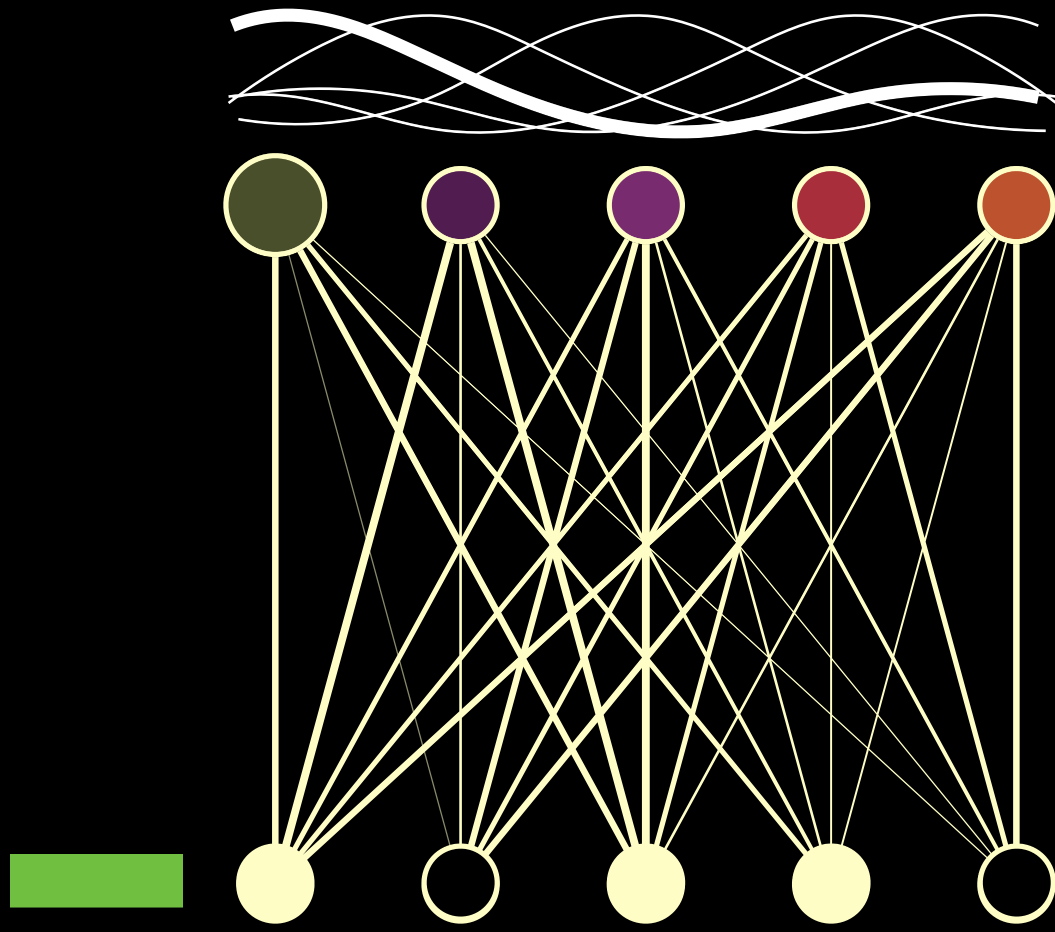
Self-Organizing Maps



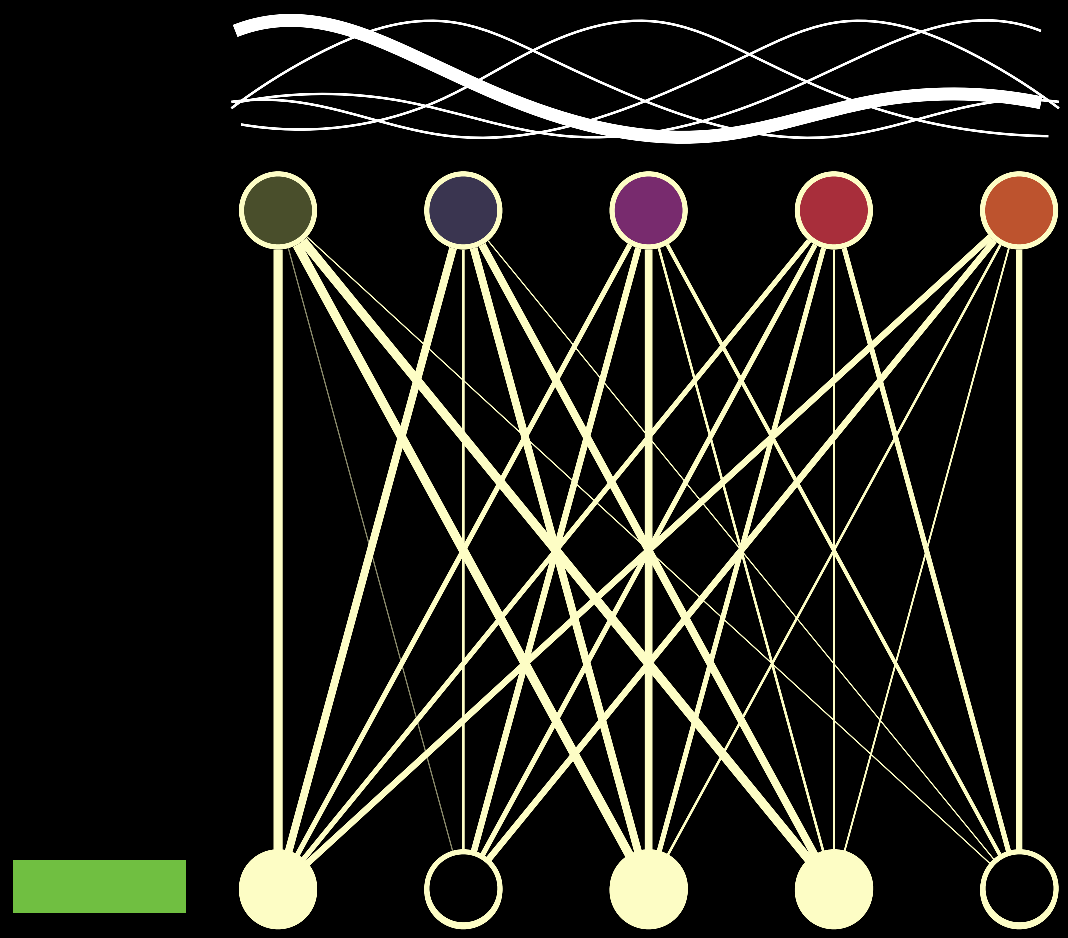
Self-Organizing Maps



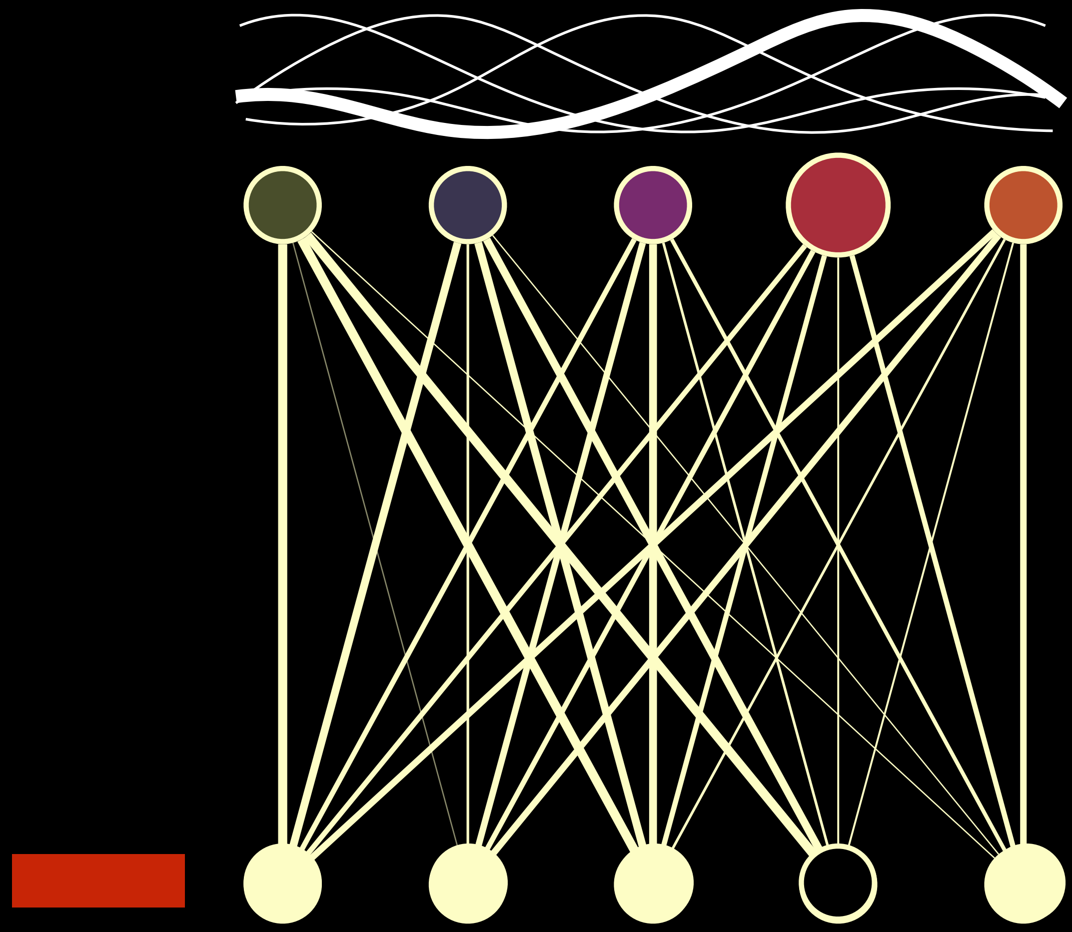
Self-Organizing Maps



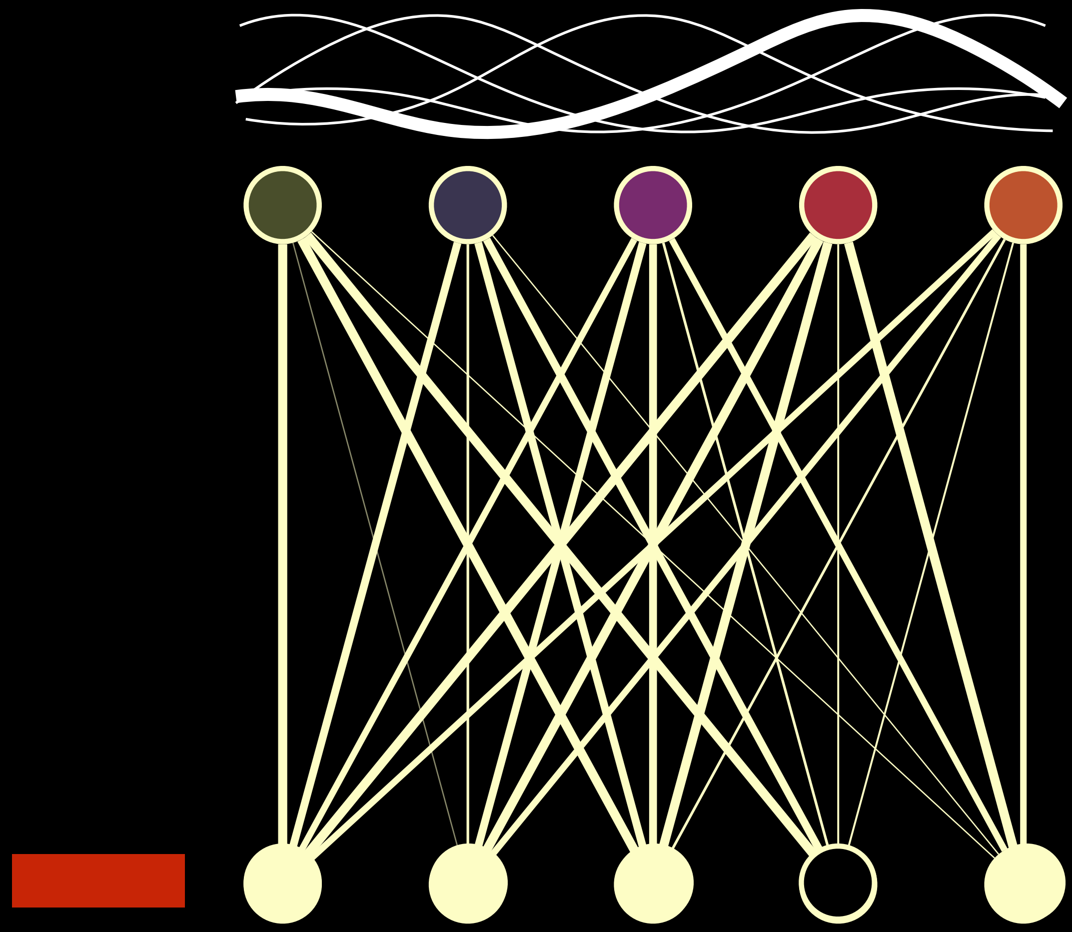
Self-Organizing Maps



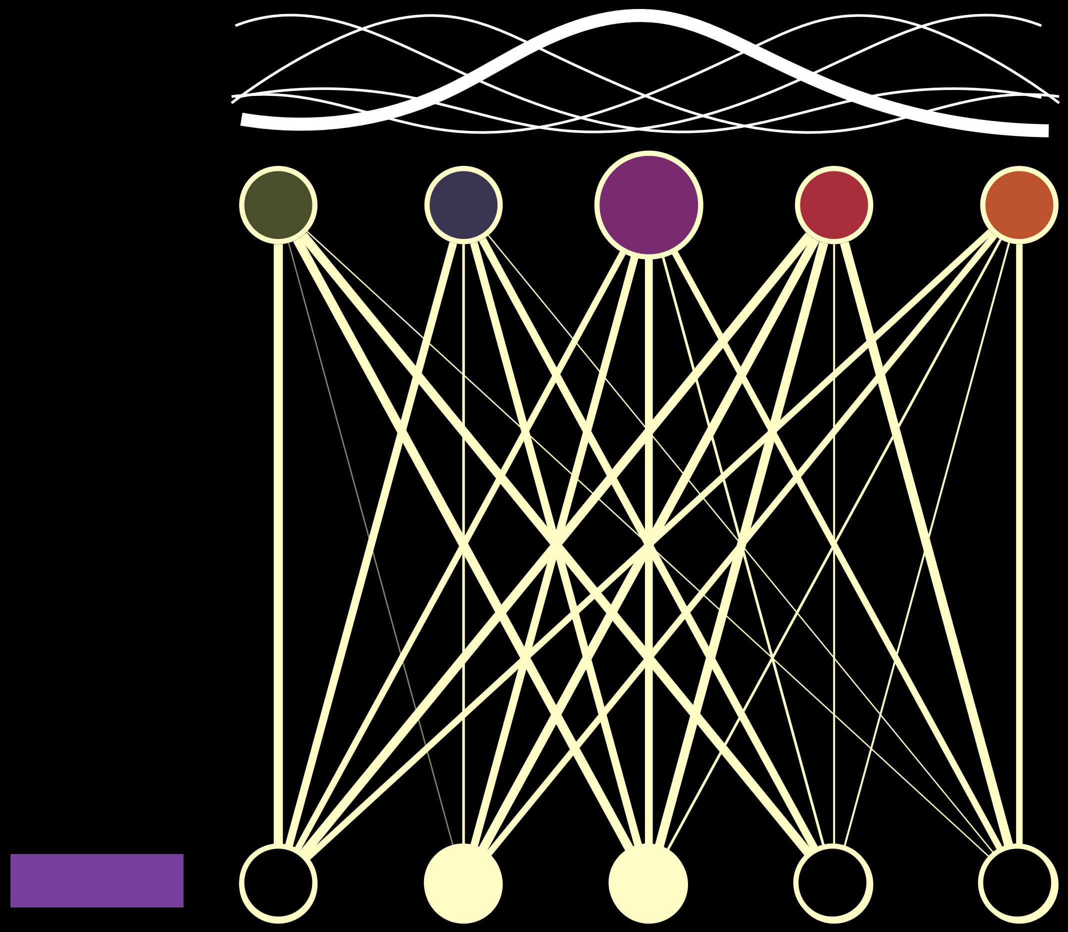
Self-Organizing Maps



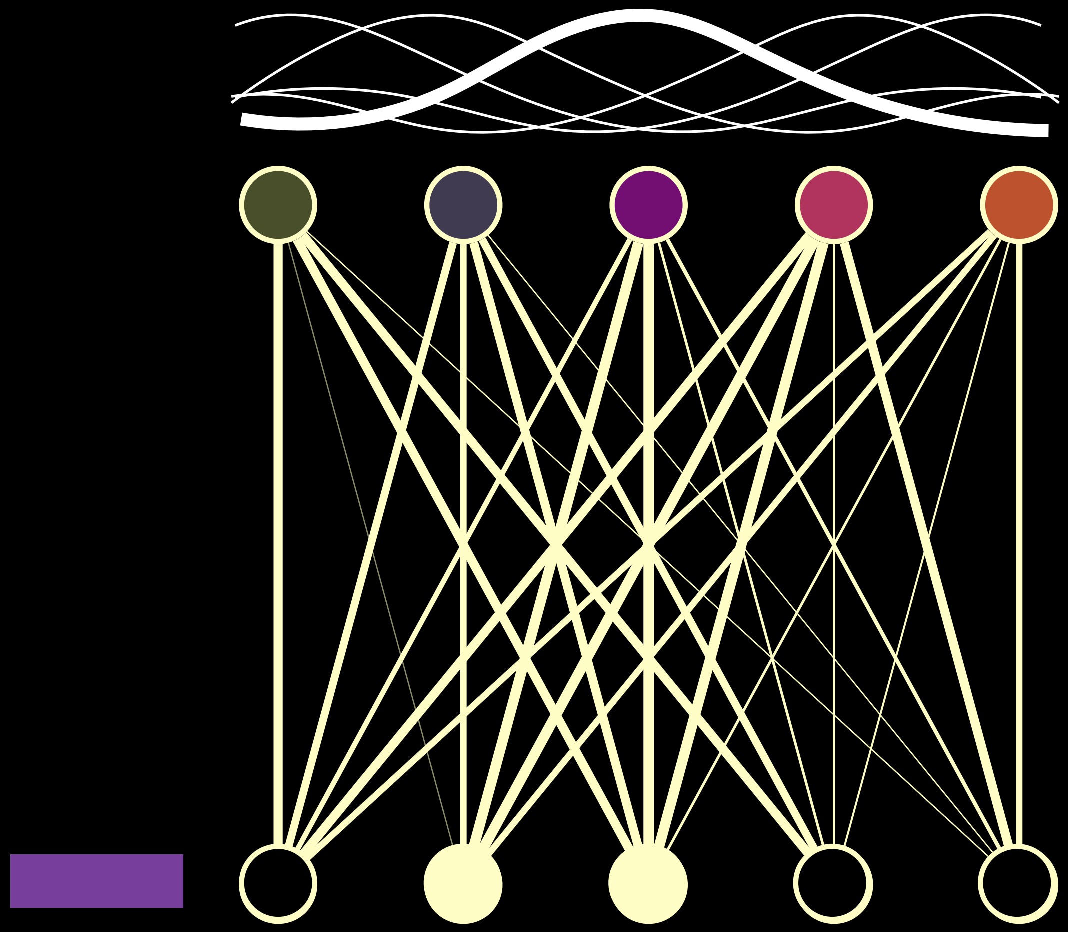
Self-Organizing Maps



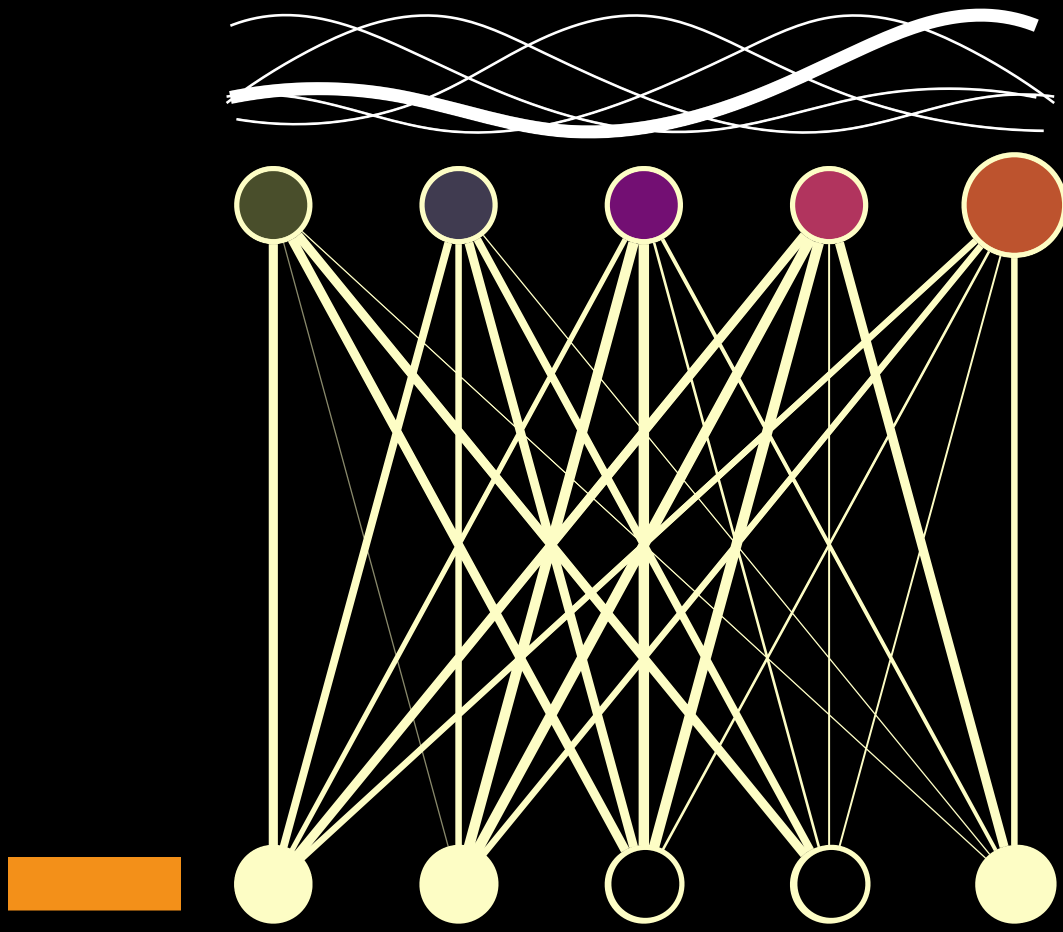
Self-Organizing Maps



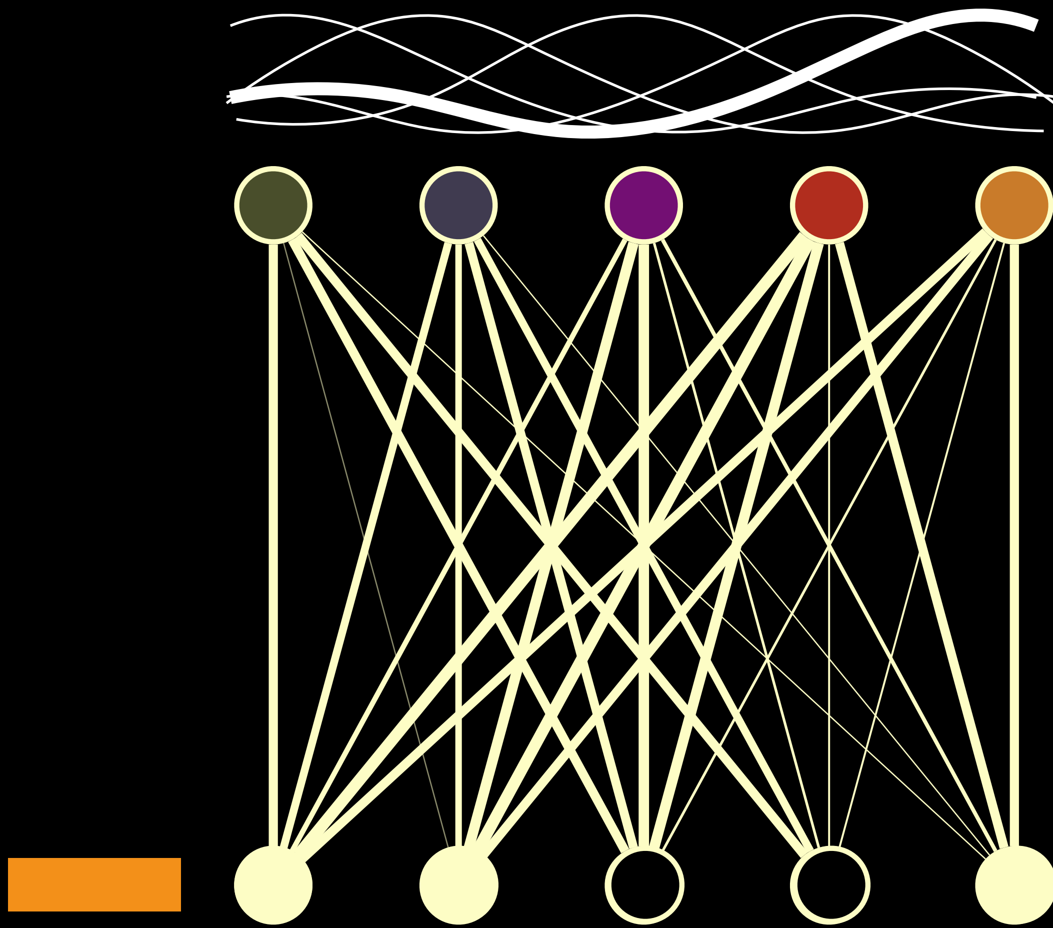
Self-Organizing Maps



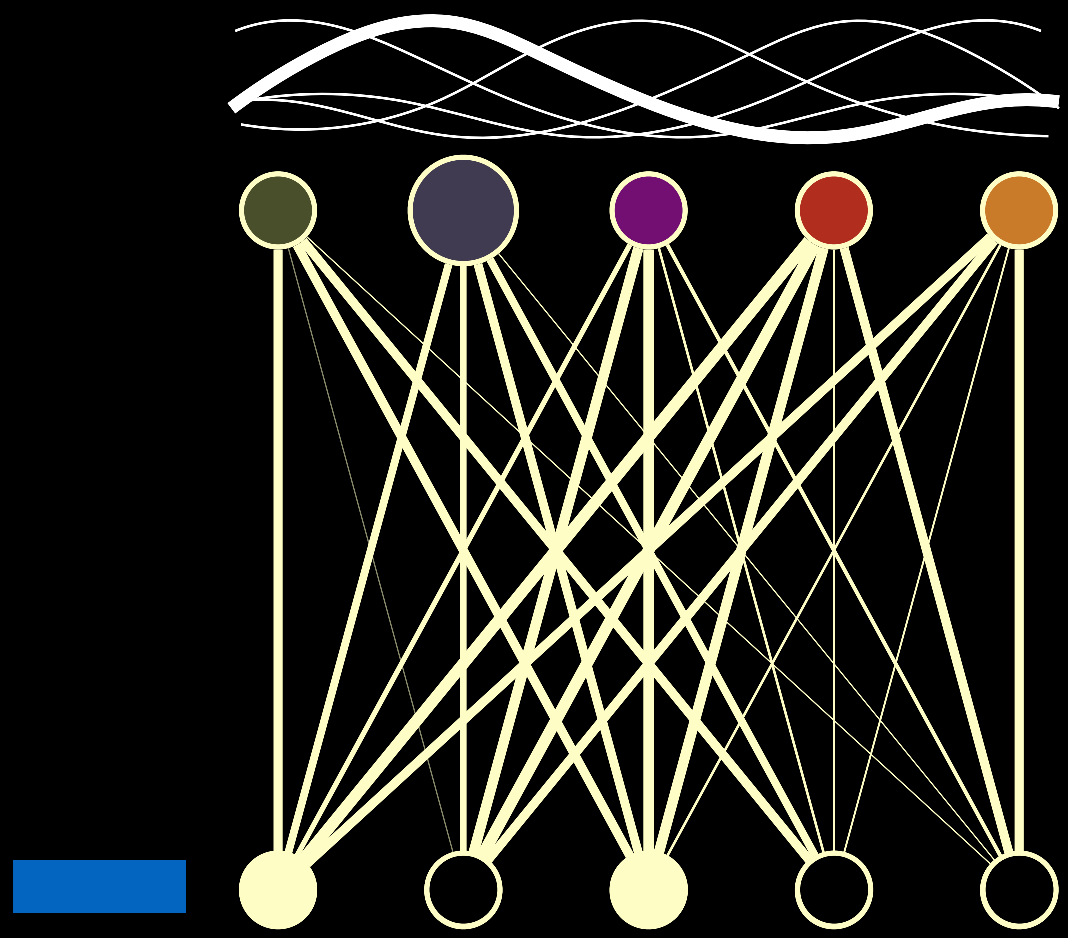
Self-Organizing Maps



Self-Organizing Maps



Self-Organizing Maps



Self-Organizing Maps