# Self-Organizing Maps Which Utilize Imposed Tree-Based Topologies

Cesar Astudillo[1] John Oommen[2]

[1] *Ex-PhD Student*, School of Computer Science, Carleton University, Canada.
[2] *Chancellor's Professor*, Fellow: IEEE; Fellow: IAPR
School of Computer Science, Carleton University, Canada.

# Outline

# What is the Goal of this Research?

- Merge two very rich fields of Computer Science.
  - Neural Networks (Self-Organizing Maps)
  - Adaptive Data Structures
- Question: Can the data structure we use Control anything?
- Question: Can we Modify the data structure as we train?

# What is the Goal of this Research?

- Question: What is more important
  - The Neural Network?
  - The Data Structure?
  - The technique for Modifying the latter?
- Open Issues...
  - Other Neural Networks?
  - Other Data Structures?
  - Other techniques for Modifying these?

# What is a SOM? What Does it Do?

- Artificial Neural Networks.
- Maps high dimensional spaces to 2D (or 3D).
- Used in clustering and visualization.
- Learns stochastic distribution of the data.
- Preserves the topology of the data.

# How does it work?

- Operates in two modes: Training and Mapping.
- Training: Uses Unsupervised learning.
- Mapping: Automatically classifies a new input vector.

# Vector Quantization

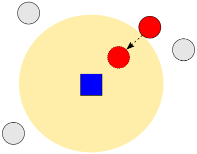
- Models probability density functions.
- Uses small subset of vectors – "Prototypes".
- Useful for lossy data compression.
- Useful for density estimation/approximation.

# Winning Neuron

- Competitive Learning.
- Best Matching Unit (BMU).

$$s(x) = \arg\min_{c \in \mathcal{C}} \| x - v_c \|$$

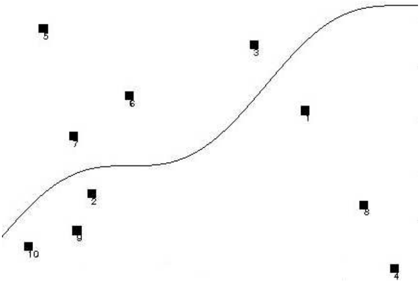
- BMU search may involve exploring all units.
- Neurons move towards the input point as below:



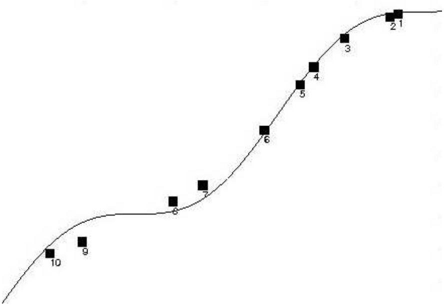
# Neighbors

- Units and edges constitute a network.
- A set of neurons "close" to the winner.
- Neighborhood function.
- Neighborhood "shrinks" with time.

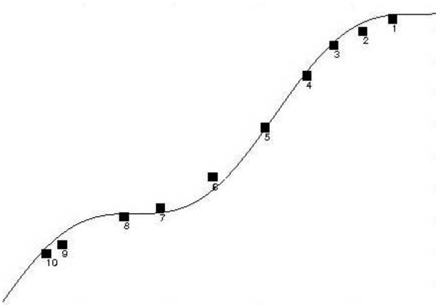
# SOM Example (1/5) – Initial Configuration



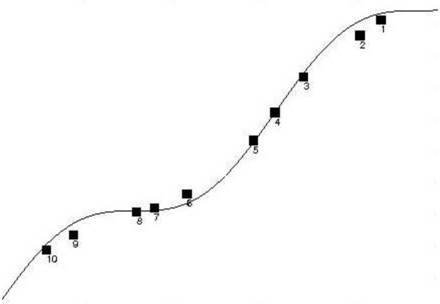
# SOM Example (2/5)



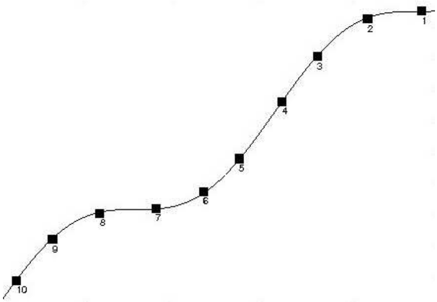
# SOM Example (3/5)



# SOM Example (4/5)



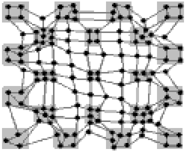
# SOM Example (5/5) – Final Configuration




Observe convergence: Distribution and Topology

# Known Drawbacks of the SOM.

- Nodes may converge to zero density areas.
- Finding the winner neuron is costly.
- Run many times to obtain suitable parameters.
- Few neurons often represent the data inaccurately.
- "Curse of Dimensionality".



from L. Guan's paper

# Why Tree-Based Variants?

- Represent complicated data distributions more accurately.
- Speed up costly tasks (i.e., finding the winner neuron).

# Possible Properties of Tree-Based SOMs

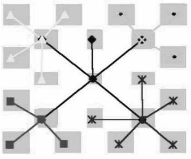
- **Dynamic**/**Static** tree.
- Distance in the **feature**/**tree** space.
- **Heuristic**/**Deterministic** winner search.
- "Frozen" neurons.
- SOMs arranged in layers.

# Tree-Based SOM Variants

- Self-Organizing Tree Maps (SOTM).
- Growing Hierarchical SOM (GHSOM).
- Tree Structured Vector Quantization (TSVQ).
- Evolving tree (ET).

# Self-Organizing Tree Maps (SOTM)

- Dynamic tree.
- Distance in the feature space.
- Distance threshold is used add nodes.



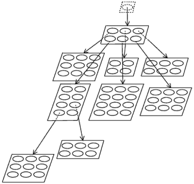
from L. Guan's paper

# Self-Organizing Tree Maps (SOTM)

1. Start with a single node
2. Main Loop (Repeat until Convergence)
   - 2.1 Obtain input
   - 2.2 Find BMU
   - 2.3.a If distance is greater than a threshold, create a new node
   - 2.3.b Else update BMU

# Growing Hierarchical SOM (GHSOM)

- Dynamic tree.
- SOMs arranged in layers.
- Error measure is used to add nodes.



GHSOM logo

# Growing Hierarchical SOM (GHSOM)

1. Start with a 2x2 SOM at layer-0
2. Main Loop (Repeat until Convergence)
   - 2.1 Train the SOM layer
   - 2.2 Calculate quantization error of each node (mqe)
   - 2.3 Calculate quantization error of the map (MQE)
   - 2.4.a If *MQE* > threshold, add row/column to SOM layer
   - 2.4.b If some unit contain *mqe* > threshold, create new SOM layer

# Tree Structured Vector Quantization (TSVQ)

- Static tree.
- "Frozen" neurons.
- Heuristic winner search.

# Tree Structured Vector Quantization (TSVQ)

1. Define tree structure
2. Main Loop (Repeat until All Units are Frozen)
   - 2.1 Find Best
     - 2.1.a If node is frozen select best child, call Find Best
     - 2.1.b Else return current node
   - 2.2 Update BMU
   - 2.3 If BMU is selected $N$ times, it becomes frozen

# Evolving Tree (ET)

- Dynamic tree.
- Fixed number of children.
- "Frozen" neurons.
- Heuristic winner search.

# Evolving Tree (ET)

1. Start with a single node
2. Main Loop (Repeat until All Units are Frozen)
   - 2.1 Find Best Procedure
     - 2.1.a If node is frozen select best child, call Find Best Procedure
     - 2.1.b Else return current node
   - 2.2.a If node reaches threshold then Split
     - 2.2.a.1 node become frozen
     - 2.2.a.2 create/initialize *k* children
   - 2.3 Update BMU and neighbors

# Hierarchical SOM in the literature

- No clear winner.
- Opportunity for novel ideas is still open.

# The Tree-Based Topology-Oriented SOM (TTO-SOM)

- Static tree.
- Arbitrary number of children.
- Neighborhood based on tree space.
- Deterministic winner search.
- Winner search based on feature space.
- Infers both the data distribution and its structured topology.
- Generalization of the 1D SOM.

# Input and Parameters of the TTO-SOM

- Input
  1. Training sample set.
  2. Configuration of the *k*-ary tree.
- Parameters
  1. Radius of the Bubble of Activity.
  2. The SOM learning rate.

# The Radius

- The Size of the Bubble of Activity.
- Between 0 and the number of neurons.
- Its value should decrease as time proceeds.

# The Learning Rate

- Neurons should be moved toward the input signal.
- Learning Rate: The Factor of such a movement.
- Should decrease as convergence take place.

# Declaration of the User-Defined Tree

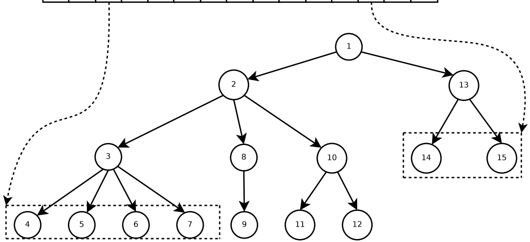
- The user describe the tree.
- Arbitrary number of children.
- Reflects *a priori* knowledge about the data distribution.
- Recursive definition.

# Declaration of the User-Defined Tree

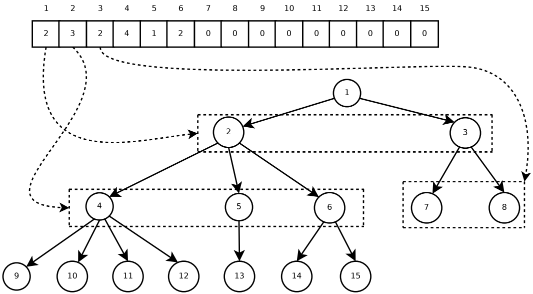
- Array specifying the number of children for each node.
- Depth-First/Breadth-First traversal.
- Index of the array: Node in the respective traversal.
- Content of the array: Number of children of each node.

# Example 1: Depth-First traversal

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 2 | 0 | 0 |



# Example 2: Breadth-First traversal



# Neural Distance Between Two Neurons...

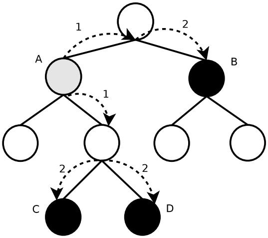
**Definition**

No. of edges in the shortest path connecting two given nodes.

- Depends on connections.
- Minimum Path.
- Includes non-leaf nodes.

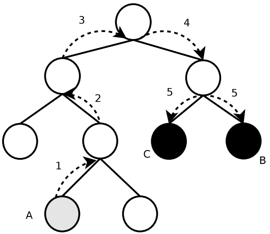
# Neural Distance: First example

- $B$, $C$ and $D$ are equidistant to $A$.
- $B$ and $D$ are at different levels.
- $B$ is a non-leaf node.



# Neural Distance: Another example

- $B$ and $C$ are at distance 5 to $A$.
- Distance from $A$ to $A$ is zero.



# The Bubble of Activity.

**Definition**

Subset of nodes: Distance $r$ away from the node examined.

- Intricately related to the notion of neural distance.
- Subset of nodes “close” to a given neuron.
- The Radius determines the size of the bubble.

# The Bubble of Activity.

## Formal Definition

$$B(v_i; T, r) = \{v | d_N(v_i, v; T) \leq r\}$$

Where,

- $v_i$ The node currently being examined.
- $v$ An arbitrary node in the tree $T$.
- $d_N$ The neural distance.
- $r$ The radius.

# The Bubble of Activity.

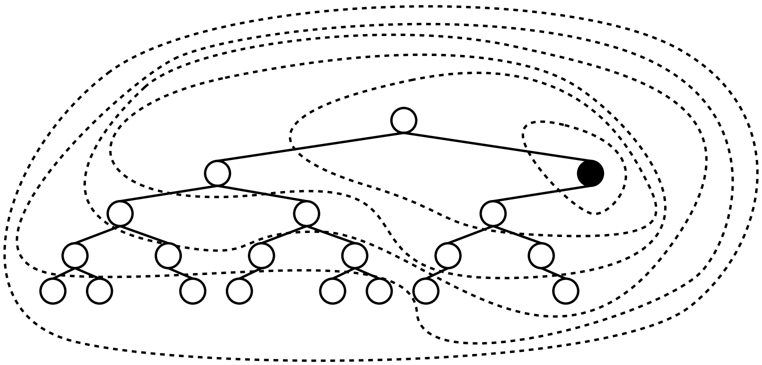
## Formal Definition

$$B(v_i; T, r) = \{v | d_N(v_i, v; T) \leq r\}$$

The Bubble of Activity also present the following properties:

1. $B(v_i, T, 0) = \{v_i\}$
2. $B(v_i, T, i) \supseteq B(v_i, T, i-1)$
3. $B(v_i, T, |V|) = V$

# Bubble of Activity: An Example



# TTOSOM properties

- Static tree.
- Distance in the Feature space for winner search.
- Distance in the Tree space for bubble of activity.
- Deterministic winner search.

# TTOSOM Training Algorithm

1. Describe Topology
   1.1 Read next item in the array of children.
   1.2 Create children.
   1.3 Call Descibe Topology recursively.
2. Main Loop (Repeat until Convergence)
   2.1 Receive input.
   2.2 Find BMU.
   2.3 Calculate Neighbors using neural distance.
   2.4 Move Neurons within the bubble of activity.
   2.5 Decrease radius/learning rate.

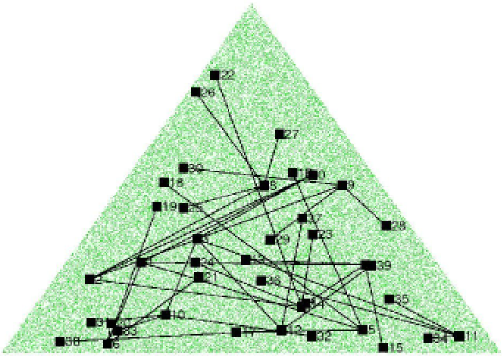
# TTOSOM Mapping Algorithm

1. Receive input.
2. Find BMU (in feature space).
3. Return BMU.

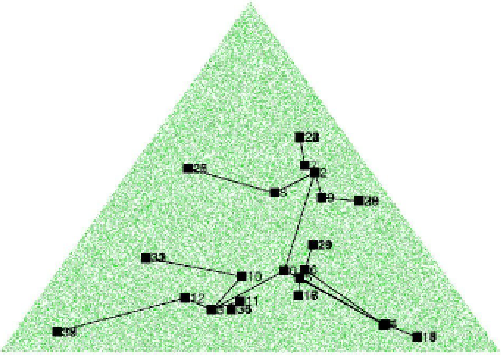
# Experiment 1: Triangular-Spaced Distribution

- Complete tree of 4 levels.
- Triangular shape.
- Capture essence of distribution.
- Define 3 children per node.

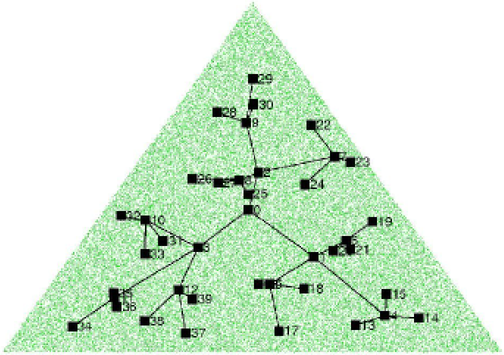
# Experiment 1: Triangular-Spaced Distribution



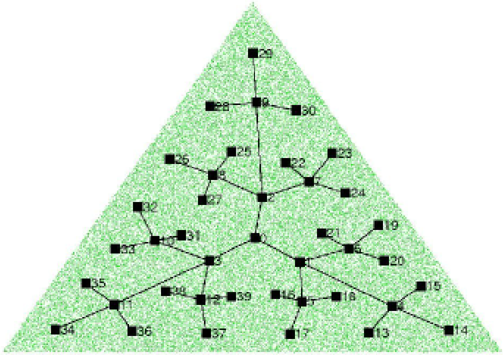
# Experiment 1: Triangular-Spaced Distribution



# Experiment 1: Triangular-Spaced Distribution

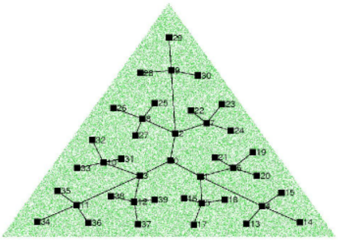


# Experiment 1: Triangular-Spaced Distribution



# Experiment 1: Triangular-Spaced Distribution

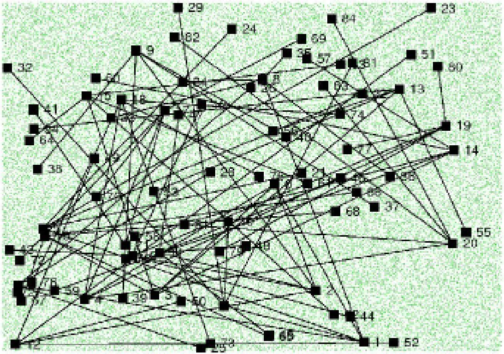
- The root roughly in the center.
- Each main branch towards a vertex.
- Sub-branches uniformly fill space around main branches.



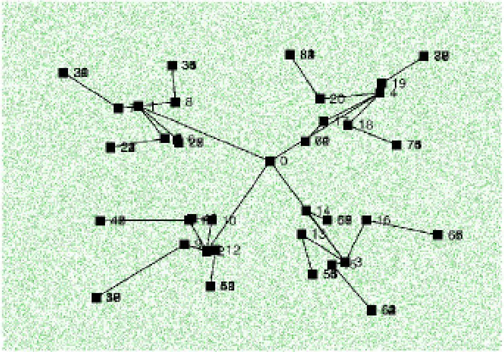
# Experiment 2: “Square-Shaped” Distribution

- Complete tree of depth 4.
- Rectangular shape.
- Capture essence of distribution.
- Define 4 children per node.

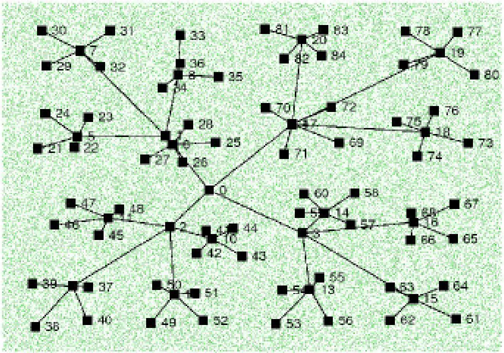
# Experiment 2: “Square-Shaped” Distribution



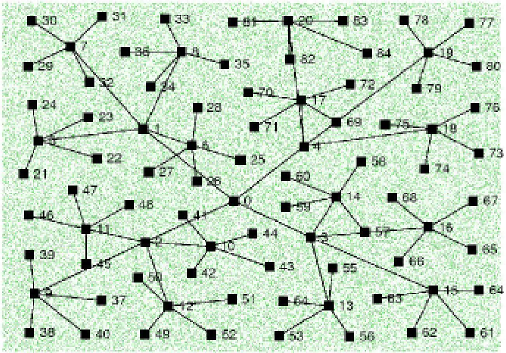
# Experiment 2: “Square-Shaped” Distribution



# Experiment 2: "Square-Shaped" Distribution

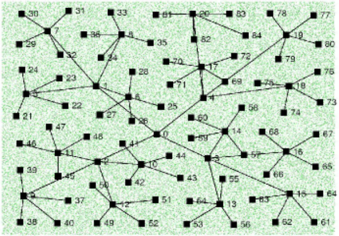


# Experiment 2: “Square-Shaped” Distribution



# Experiment 2: “Square-Shaped” Distribution

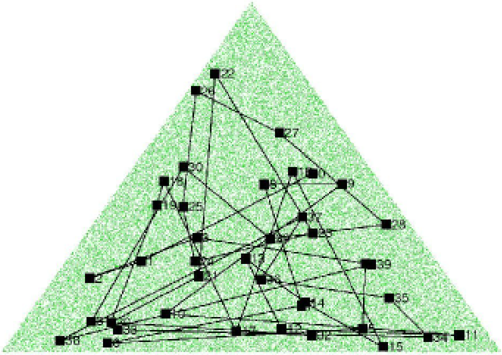
- Root near the center of mass.
- Main branches cover the principal diagonals.
- Sub-branches uniformly fill space around main branches.



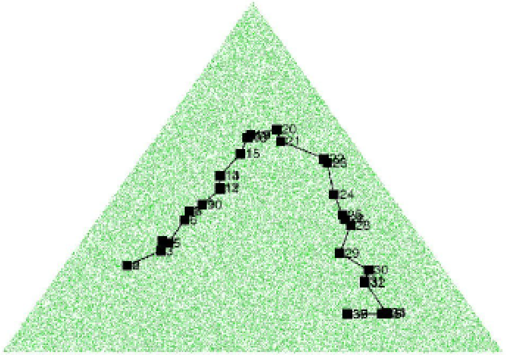
# Experiment 3: 1-ary tree

- Triangular shape.
- A list as the imposed topology.
- 1 children per node.

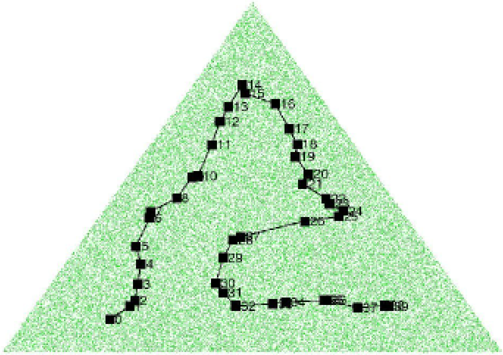
# Experiment 3: 1-ary tree



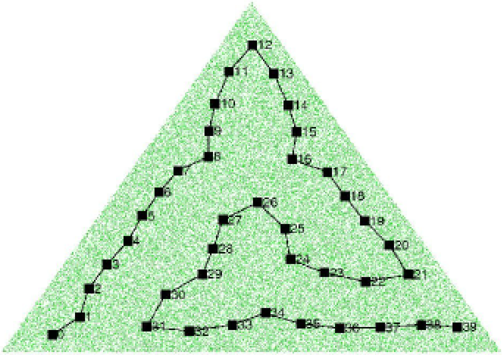
# Experiment 3: 1-ary tree



# Experiment 3: 1-ary tree

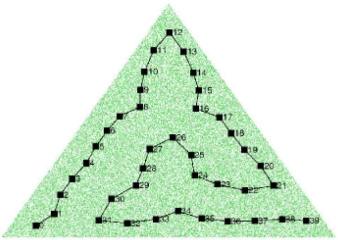


# Experiment 3: 1-ary tree



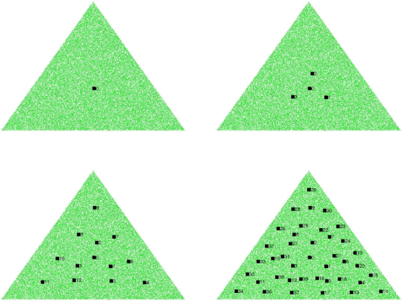
# Experiment 3: 1-ary tree

- The list represents the triangle effectively.
- Preserves “tree-like” topology.
- Generalization of 1D SOM.



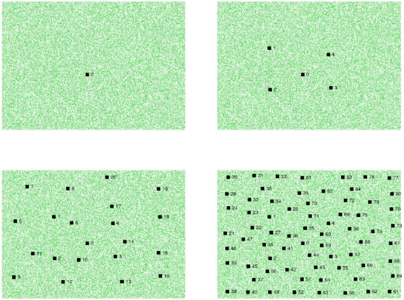
# Experiment 4: Multi-Resolution

- Hologram-like properties
- Not possessed by other SOM-based networks.
- Still represent the distribution and the structure.



# Experiment 5: Multi-Resolution

- Few level of the tree $\rightarrow$ lower resolution.
- More levels of the tree $\rightarrow$ finer resolution.



# Extracting the Skeleton of a Data Set

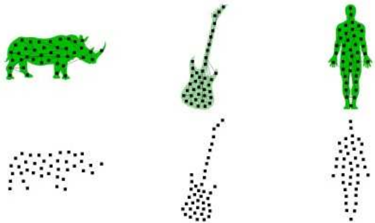
**Definition: Skeletonization**

Process by which a 2D shape is transformed into 1D.

- Dimensionality reduction technique.
- Traditional skeletonization: Assumes pixel connectivity.
- SOM variations have been used to tackle this situation.
  - GNG-like approach.
  - MST calculated over neurons.

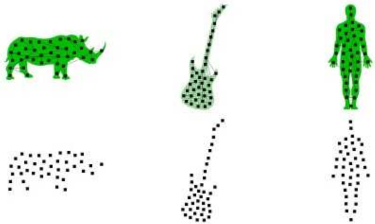
# Experiment 6: Skeletonization

- 3 objects processed using the same tree structure.
- Same schedule for parameters.
- No post processing of edges.



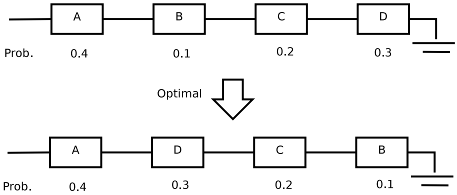
# Experiment 6: Skeletonization

- Effectively represents 2D objects in 1D.
- Without any specific adaption.



# Adaptive Data Structures

- Optimal arrangement of nodes.
- Probability of accesses known $\rightarrow$ Optimal Solution.
- We consider unknown probability of accesses.



# Adaptive Data Structures: Single Linked-List.

- Move-to-Head.
- Exchange Rule.

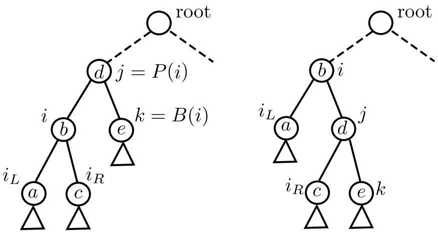
## Adaptive Data Structures: Trees.

- Move-to-Root.
- Exchange Rule.
- Conditional Rotations.
- Splay Trees.
- D-tree.
- Monotonic trees.

# Conditional Rotations

- Asymptotically produce an optimal tree.
- Rotations does not occur every step.
- Only if Weighted Path Length (WPL) is decreased.
- Rotation takes $O(1)$.
- Requires only 1 extra counter per node.

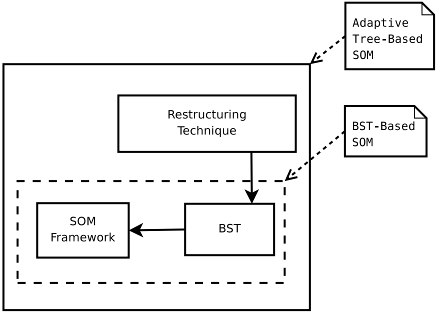
# Rotations for a Binary Search tree



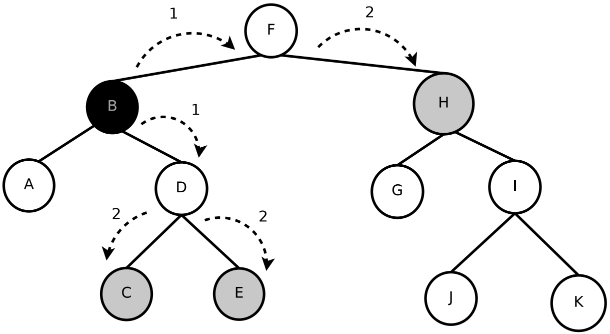
# Conditional Rotations: How?

- WPL is maintained for each node.
- On query: Update the visited nodes down on path.
- Stop once the node is found.
- Check if as a result of a rotation the WPL Decreases.
- Move node toward the root conditionally.

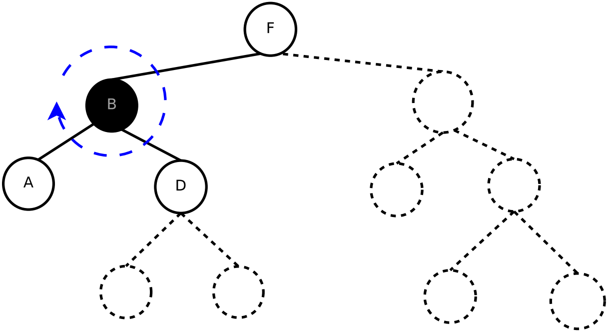
# ADS-TTOSOM Architecture



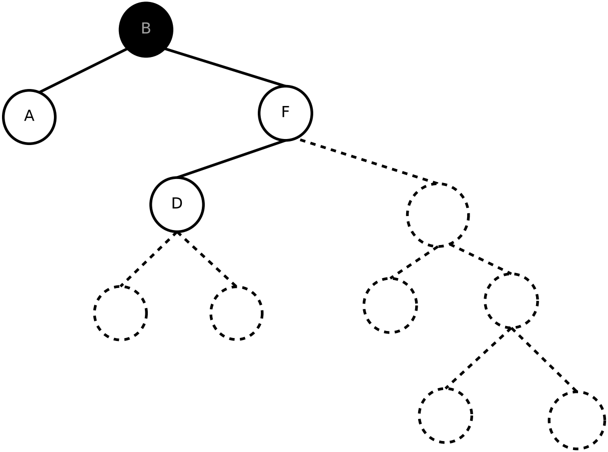
# Neural Distance – TTO-CONROT



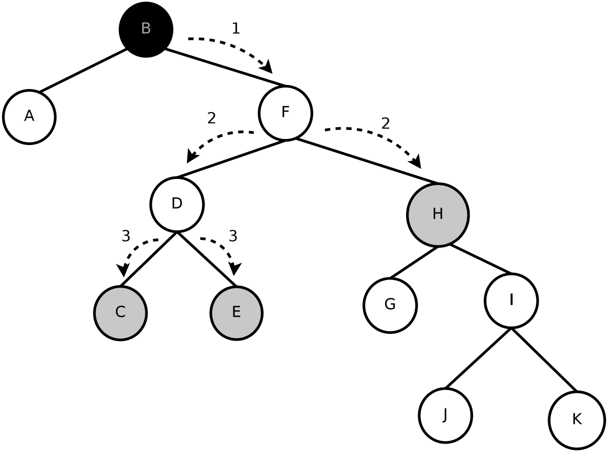
# Neural Distance – TTO-CONROT



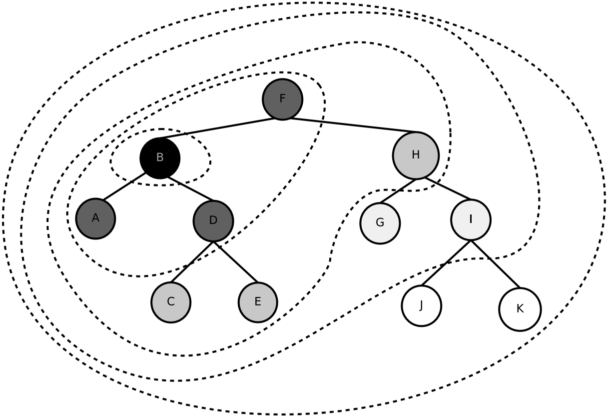
# Neural Distance – TTO-CONROT



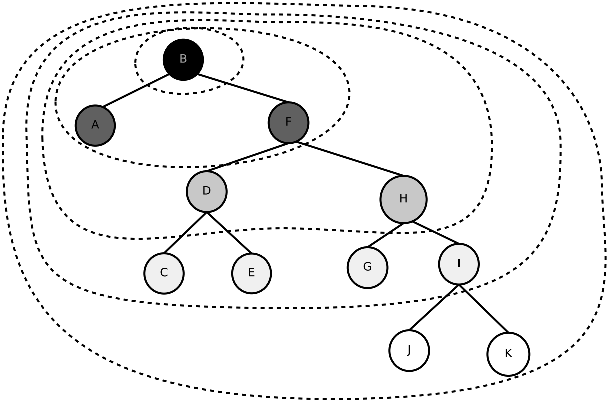
# Neural Distance – TTO-CONROT



# Bubble of Activity – TTO-CONROT



# Bubble of Activity – TTO-CONROT

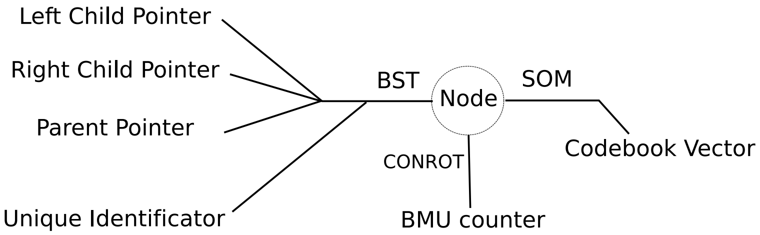


# Neuron Information

TTOSOM Neurons require fields for implementing:

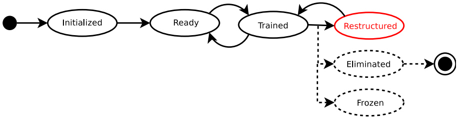
- SOM
- BST
- ADS – CONROT

# Neuron Information



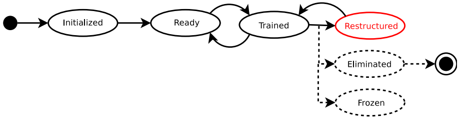
# Neural State

| | |
|---|---|
| Created | The new neuron is added to the tree. |
| Initialized | The codebook vector assumes a value. |
| Ready | Neuron is ready for training. |
| Trained | The neuron is under training. |
| Frozen | The neuron becomes static. |
| Eliminated | The neuron is no longer useful. |
| Restructured | The structure is modified (no deletion). |



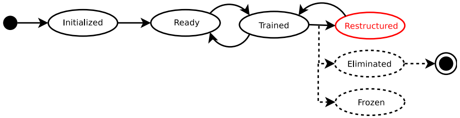
# Neural State

- **Created** The new neuron is added to the tree.
- **Initialized** The codebook vector assumes a value.
- **Ready** Neuron is ready for training.
- **Trained** The neuron is under training.
- **Frozen** The neuron becomes static.
- **Eliminated** The neuron is no longer useful.
- **Restructured** The structure is modified (no deletion).



# Neural State

| | |
|---|---|
| Created | The new neuron is added to the tree. |
| Initialized | The codebook vector assumes a value. |
| Ready | Neuron is ready for training. |
| Trained | The neuron is under training. |
| Frozen | The neuron becomes static. |
| Eliminated | The neuron is no longer useful. |
| Restructured | The structure is modified (no deletion). |



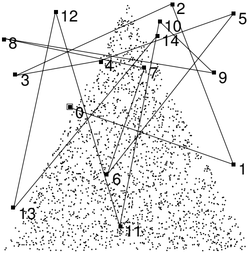
# TTO-CONROT Training Algorithm

1. Describe Topology
   - 1.1 Read next item in the array of children.
   - 1.2 Create children.
   - 1.3 Call Descibe Topology recursively.
2. Main Loop (Repeat until Convergence)
   - 2.1 Get input.
   - 2.2 Find BMU.
   - 2.3 Rotate BMU Conditionally.
   - 2.4 Calculate Neighbors using neural distance.
   - 2.5 Move Neurons within the bubble of activity.
   - 2.6 Decrease radius/learning rate.

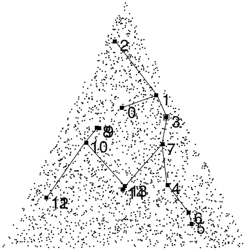
# Experiment 7: 1-ary tree – TTOSOM-CONROT

- List topology.
- Learns a Triangle distribution.
- Data structure is self-adapted.
- Most accessed nodes are moved to the root conditionally.
- BST property is also preserved.

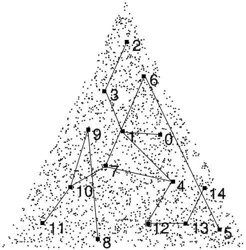
# Experiment 7: 1-ary tree – TTOSOM-CONROT



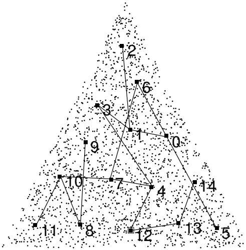
# Experiment 7: 1-ary tree – TTOSOM-CONROT



# Experiment 7: 1-ary tree – TTOSOM-CONROT



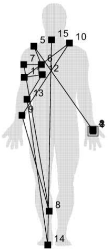
# Experiment 7: 1-ary tree – TTOSOM-CONROT



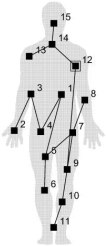
# Experiment 8: Human Shape – TTOSOM-CONROT

- Human Shape.
- Originaly a list topology .
- Most accessed nodes are moved to the root conditionally.

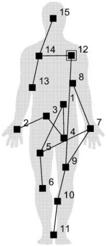
# Experiment 8: Human Shape – TTOSOM-CONROT



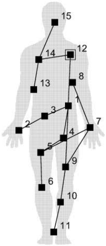
# Experiment 8: Human Shape – TTOSOM-CONROT



# Experiment 8: Human Shape – TTOSOM-CONROT



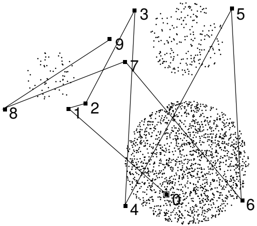
# Experiment 8: Human Shape – TTOSOM-CONROT



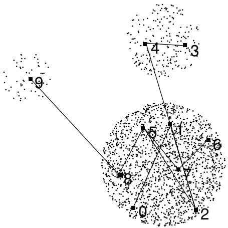
# Experiment 9: Circles – TTOSOM-CONROT

- Three cloud of Points.
- Originaly a list topology.
- Number of codebook proportional to area.
- Most accessed codebook closer to the root.

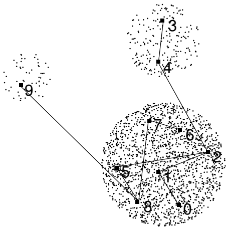
# Experiment 9: Circles – TTOSOM-CONROT



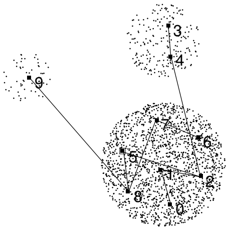
# Experiment 9: Circles – TTOSOM-CONROT



# Experiment 9: Circles – TTOSOM-CONROT



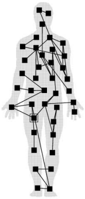
# Experiment 9: Circles – TTOSOM-CONROT



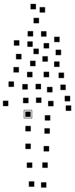
# Experiment 10: Sekeletonization – TTOSOM-CONROT

- Human, Rhinoceros.
- Originaly a list topology.
- Still represent shape accurately.

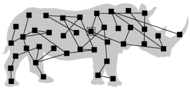
# Experiment 10: Sekeletonization – TTOSOM-CONROT



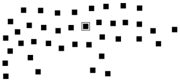
# Experiment 10: Sekeletonization – TTOSOM-CONROT



# Experiment 10: Sekeletonization – TTOSOM-CONROT



# Experiment 10: Sekeletonization – TTOSOM-CONROT



# Summary

- TTO-SOM is able to determine:
  - Distribution of the data.
  - Structured topology.
- TTO-SOM can represent data in multiple granularity levels.
- TTO-SOM can extract a skeleton from the shape.

- TTOSOM-CONROT is able to determine:
  - Distribution of the data.
  - Adaptive Structure of topology.
- TTOSOM-CONROT merges Neural Nets and Adaptive DS.
- TTOSOM-CONROT has huge potential; Many open areas.