Self-Organizing Maps

Background

- A neural net technique based on image processing in the brain
- Invented in the 1980s by Teuvo Kohonen
- Also called “Kohonen Maps”

Purpose

- Reduce dimensionality to 2 or 3 dimensions
- Preserves numerical relationship between data points
- A non-linear PCA

Uses

- Customer segmentation
- Document clustering
- Image processing
- Intrusion detection
- Speech recognition
- Data compression
- Gene research

Pros and Cons

- Pros:
  - Resistant to outliers/noise
  - Allows intuitive, visual exploration of clusters
  - Preserves structure of data
- Cons:
  - Can be computationally expensive
  - Can't handle missing values
  - Must have numerical attributes

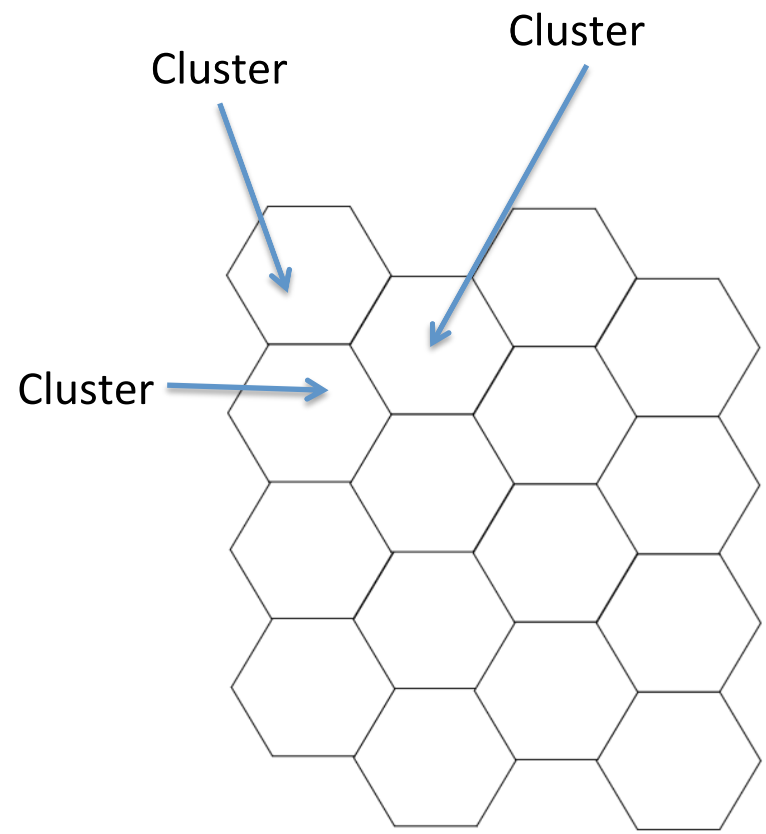
First, an analogy…

Analogy…

- Consider k-medoids
  - Choose k number of clusters
  - Each cluster has a representative centroid
  - Centroid is updated when an item is assigned to that cluster

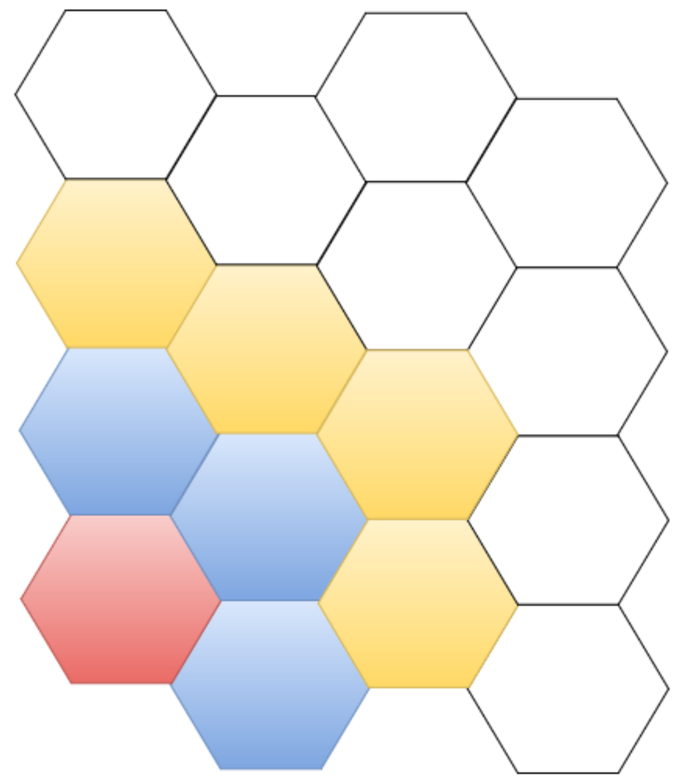
SOM ≈ k-medoids in a grid

- Choose k
- Clusters locked in a 2-D grid
- Each cluster has a centroid
- Items assigned to most similar centroid (Euclidian distance)
- Centroids updated with each assignment



How SOM is Different

- Large k: 10s to 10,000s
- Neighboring centroids update each time an item is assigned
- Amount the centroid is updated and the number of neighbors that are updated decreases each time a new item is assigned



How it actually works…

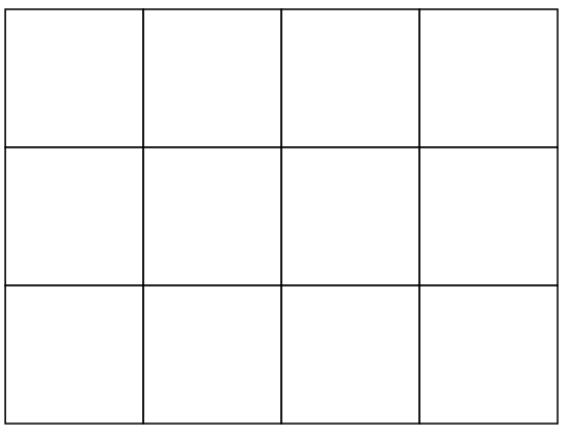
Step 0: Normalize the Data

Animal	Lifespan	Avg. Weight	Scariness
Lizard	6	5	3
Tiger	67	80	10
Snail	1	0.5	0
Lion	64	90	9

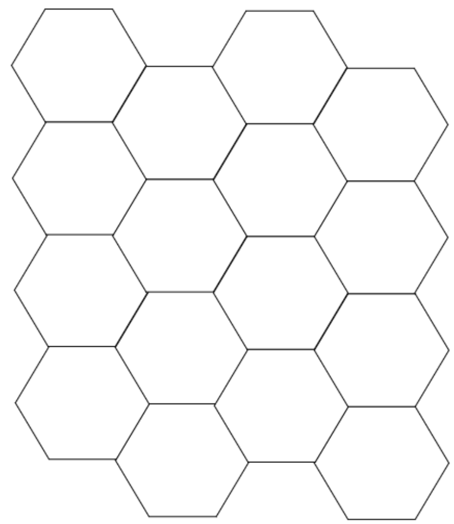
Animal	Lifespan	Avg. Weight	Scariness
Lizard	0.09	0.06	0.30
Tiger	1.00	0.89	1.00
Snail	0.01	0.01	0.00
Lion	0.96	1.00	0.90

Step 1: Create a Grid of Nodes

- Current debate over number of nodes to choose
- Grids can be rectangular or hexagonal lattice



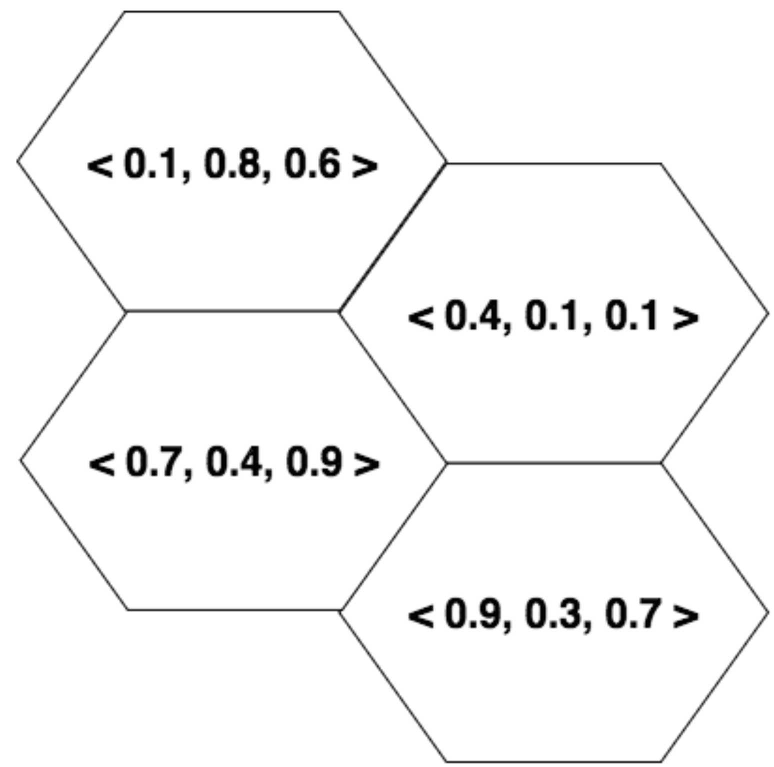
OR



(Gives more granular output)

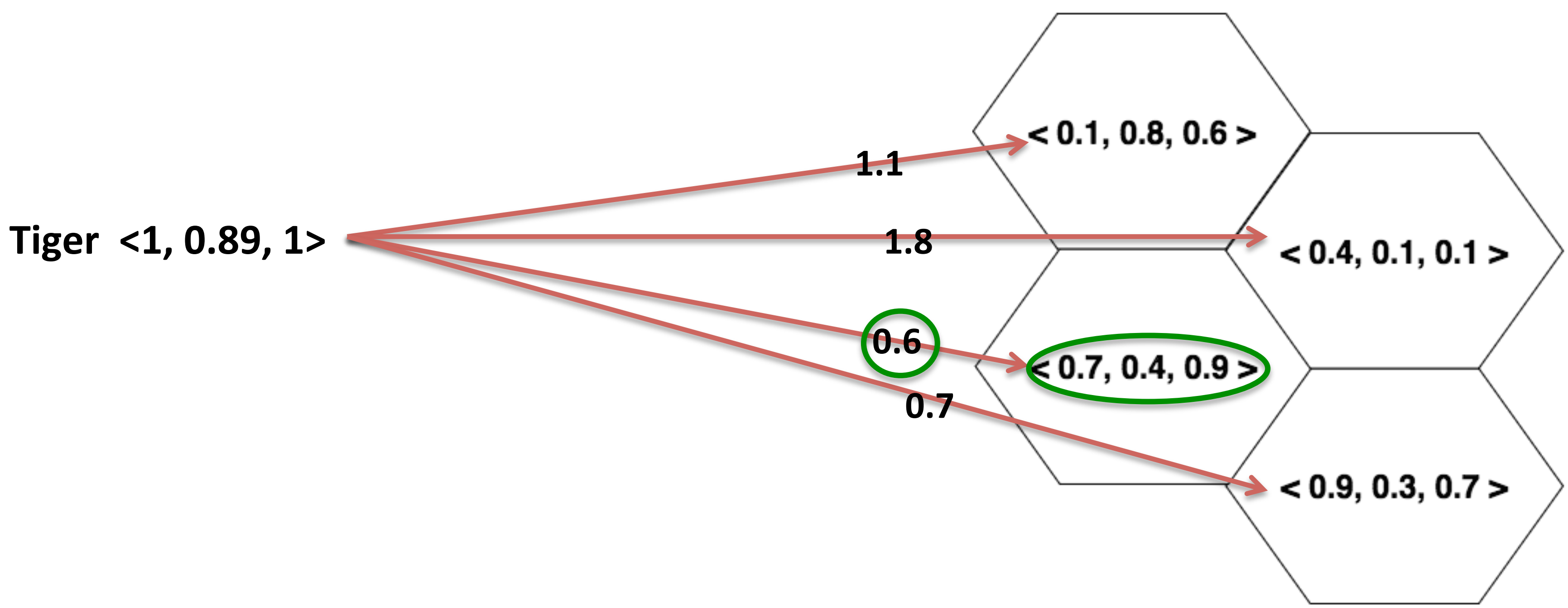
Step 2: Initialize Nodes

- Assign each node a random starting value (“weight”) for each dimension in the dataset
- Values can be:
  - Random sample from dataset
  - Random [0,1] values from the data space
  - Evenly sampled from the subspace of the 2 largest PCA eigenvectors



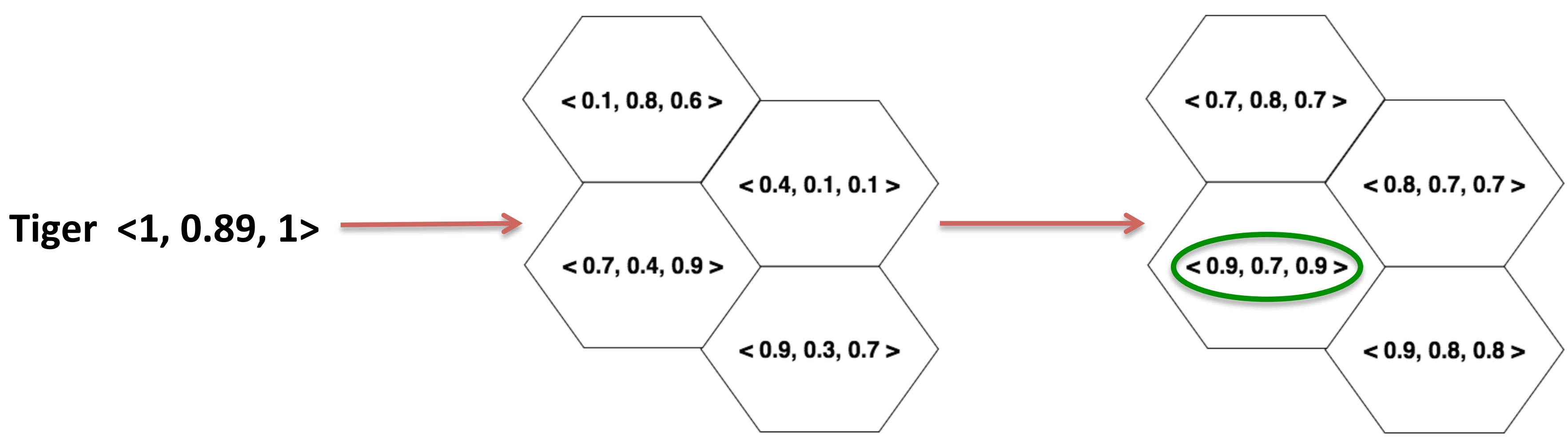
Step 3: Assign Nodes

- Randomly select an item from the dataset, and assign it to the node with the closest Euclidian distance to its attribute values



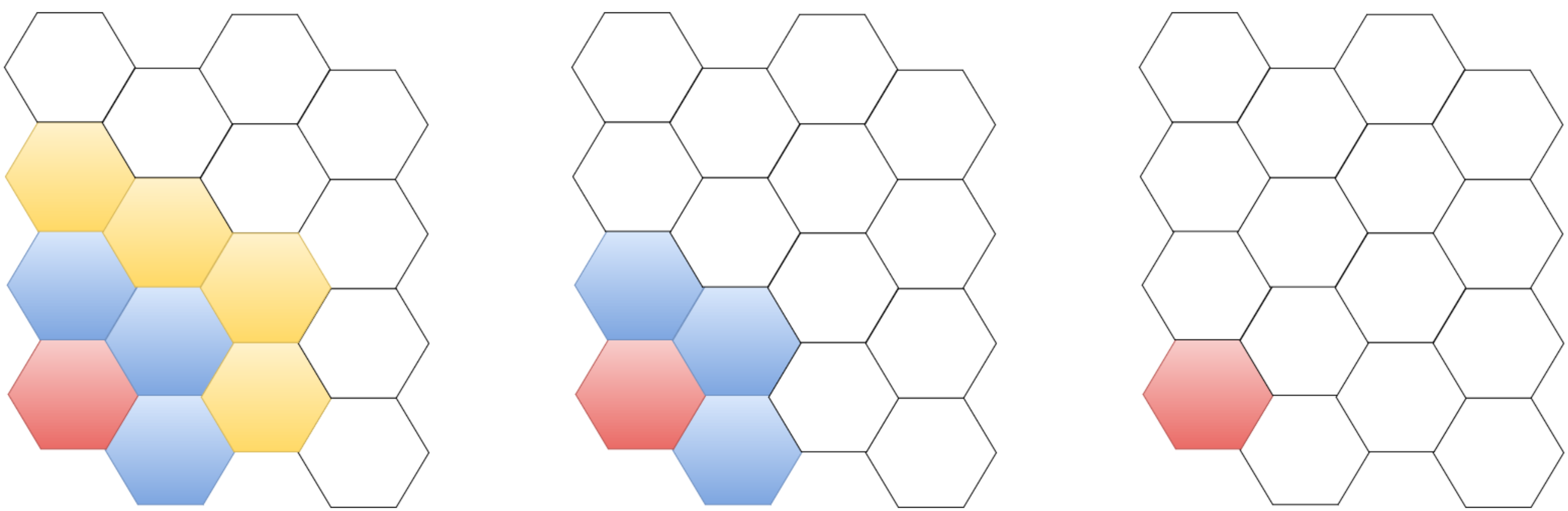
Step 4: Update the Node Values

- Assigned node and neighboring nodes (“neighborhood) centroids update to be more like the newly mapped item



Neighboring Node Changes

- Fewer nodes updated each time
- Number of neighboring nodes decreases with each new item assigned
- Debate over starting radius of nodes that get updated



Centroid Update Formula

$$w_i(t+1) = w_i(t) + f_i(t) * [d(t) - w_i(t)]$$

w_i = nearest centroid to the data point being assigned

$d(t)$ = data point being assigned

t = iteration number

$f_i(t)$ = neighborhood function that controls the impact that the winning node has on other nodes- Most commonly:

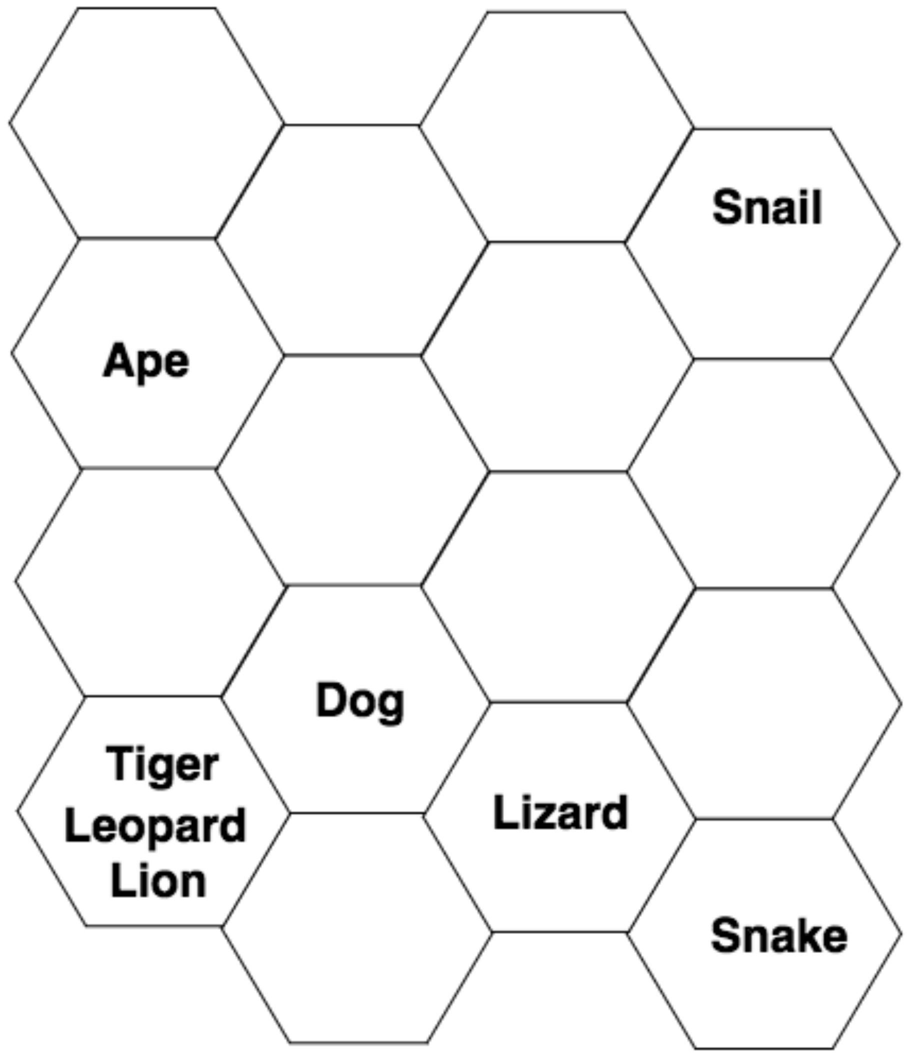
$$f_i(t) = \exp\left(-\frac{dist^2}{2\sigma^2}\right)$$

$dist^2$= Euclidian distance between point being classified and assigned node value
σ^2 = size of neighborhood (radius of points)- has its own decay function

Steps 5 and 6: Stopping

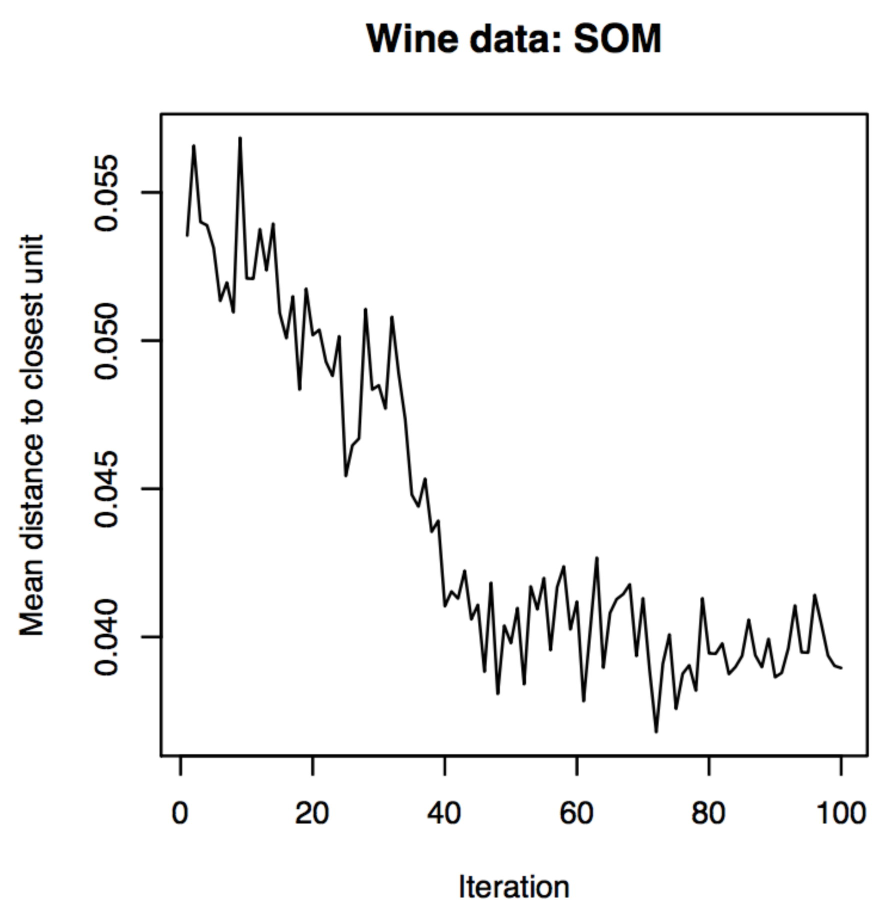
- Continue assignment and updates until all items are assigned to a node
- "Iterations" are number of items that are assigned. Used for classification/conceptual mapping

Final Outcome



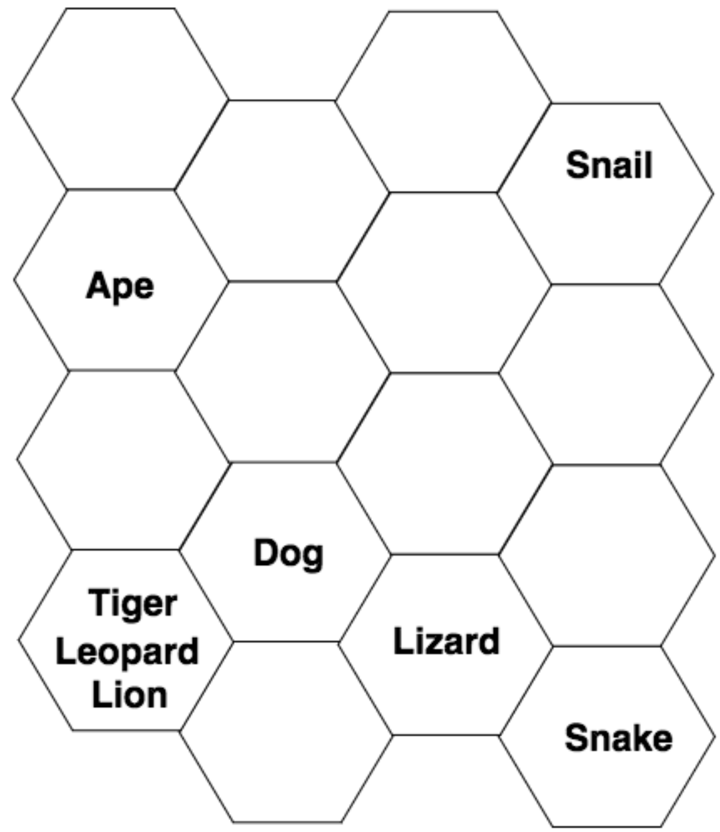
Validating Maps

- Quantization Error- mean distance between items in a node and node centroid
  – Decreases with more nodes
  – Decreases with more iterations



Clustering Approaches with SOM

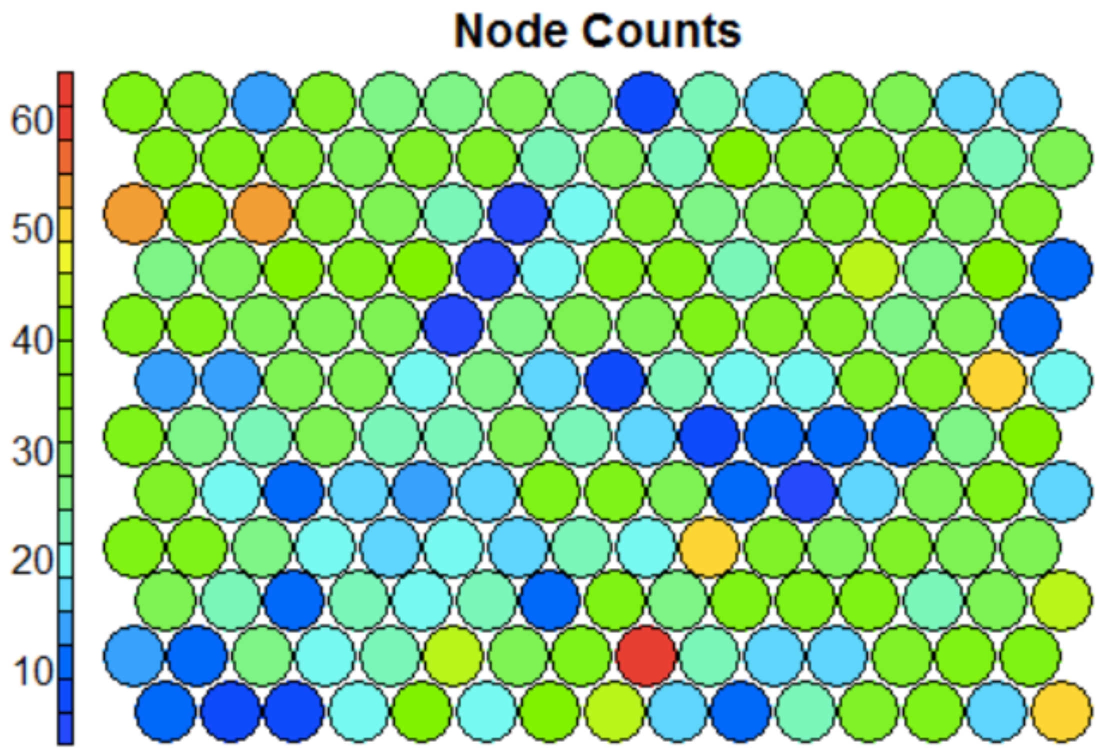
- Visually explore spatial patterns of items that are in nearby nodes
- Set a small k and each node is a cluster
- Use output of SOM for hierarchical clustering



Visualizing the Results

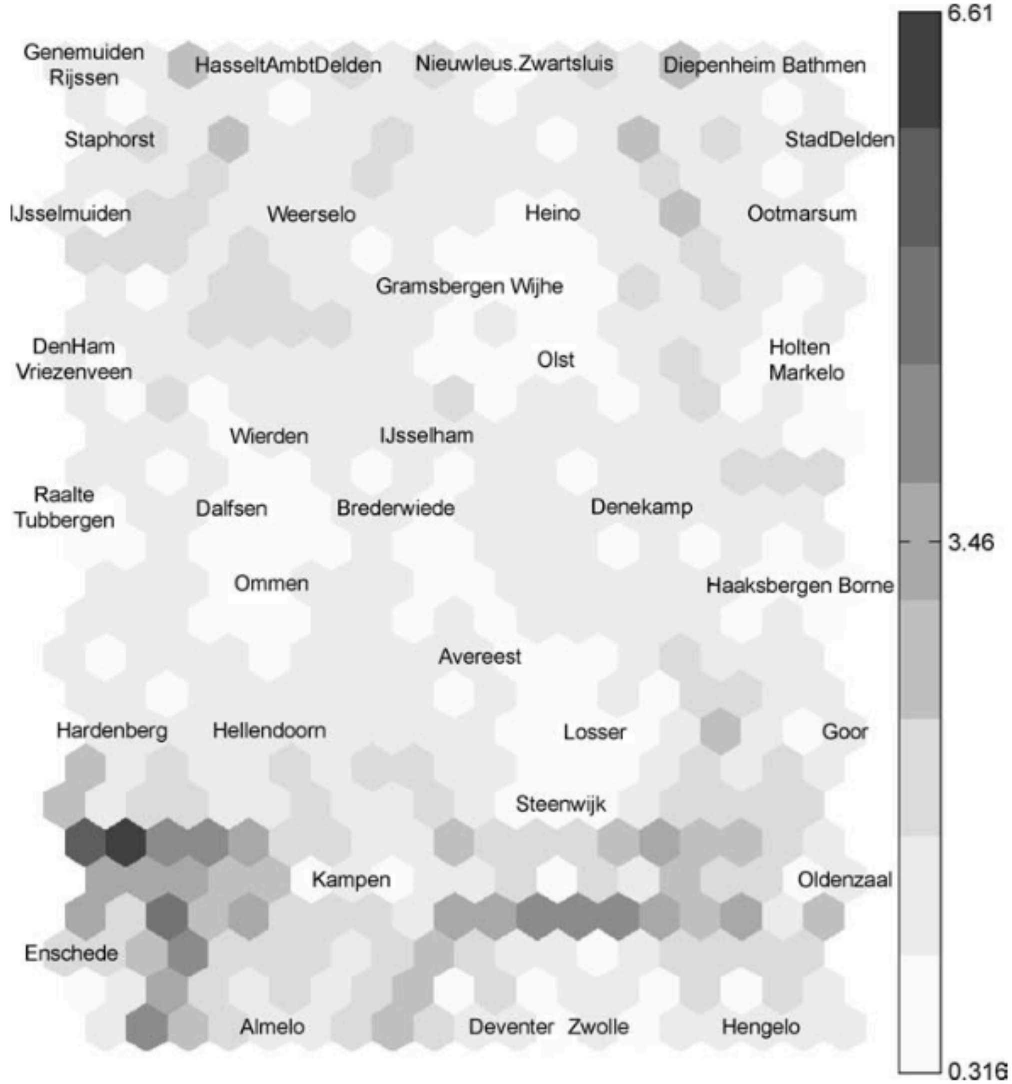
Node Counts

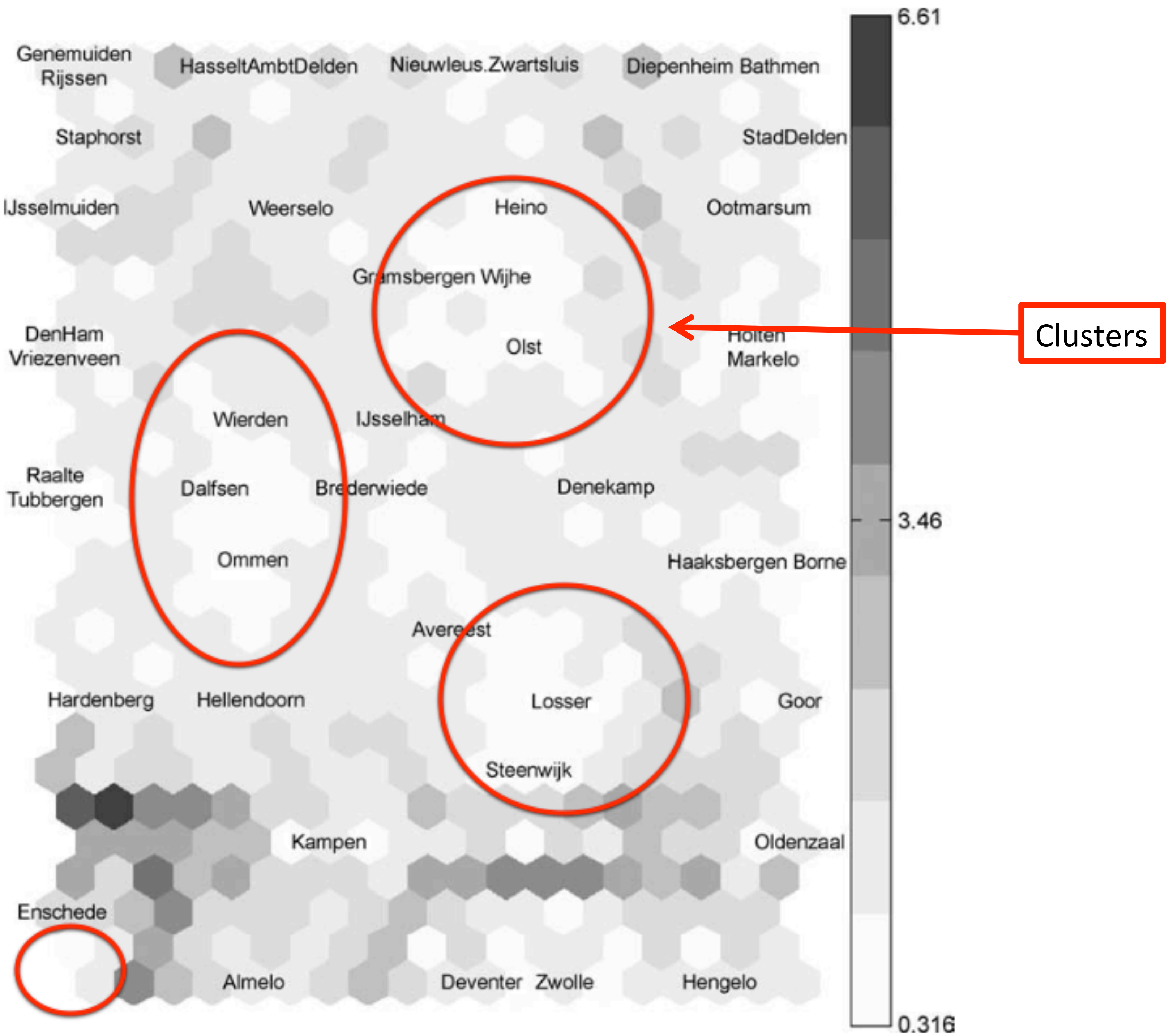
- Can see the number of items assigned to each node when viewing grid



U-maps / U-mats

- "Unified Distance Matrix"
- Distance between nodes in greyscale
  - White: most similar nodes
  - Black: most dissimilar
- x and y positions don't have meaning
- Clarity of separation another way of judging map quality
  - Light areas = clusters
  - Dark areas = cluster separators

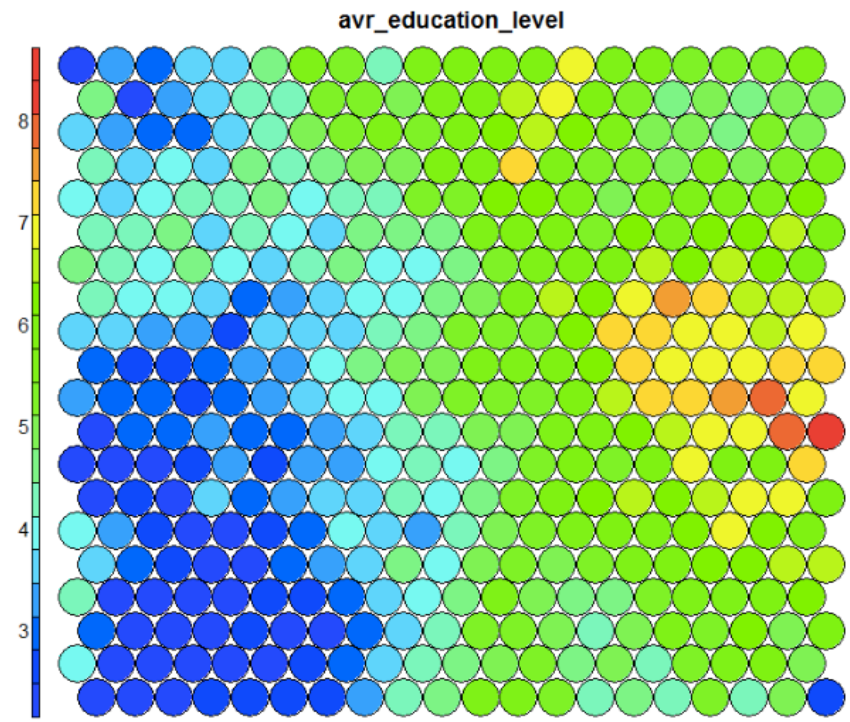




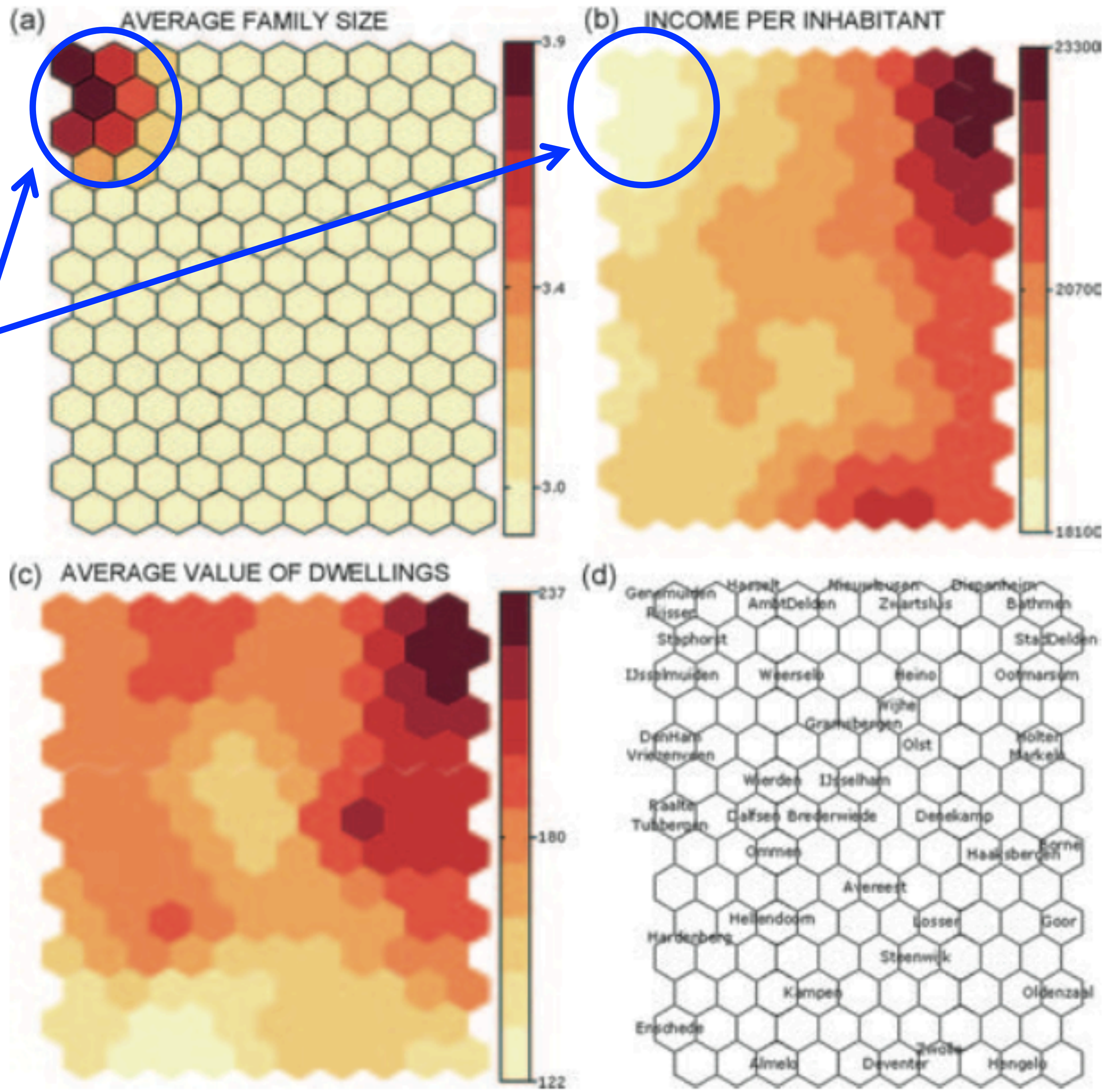
Genemuiden
Rijssen
HasseltAmbtDelden
Nieuwleus.Zwartsluis
Diepenheim Bathmen
Staphorst
StadDelden
IJsselmuiden
Weerselo
Heino
Ootmarsum
Gramsbergen Wijhe
DenHam
Vriezenveen
Olst
Holten
Markelo
Clusters
Wierden
IJsselham
Raalte
Tubbergen
Dalfsen
Brederwiede
Denekamp
Ommen
Haaksbergen Borne
Avereest
Hardenberg
Hellendoorn
Losser
Goor
Steenwijk
Kampen
Oldenzaal
Enschede
Almelo
Deventer
Zwolle
Hengelo
6.61
3.46
0.316

Component Plane Representation/ Heatmap

- Centroids of nodes as color
  - Red: High values
  - Blue: Low values
- x and y positions don't have meaning
- Because node arrangement doesn't change, can easily compare heatmaps for multiple variables

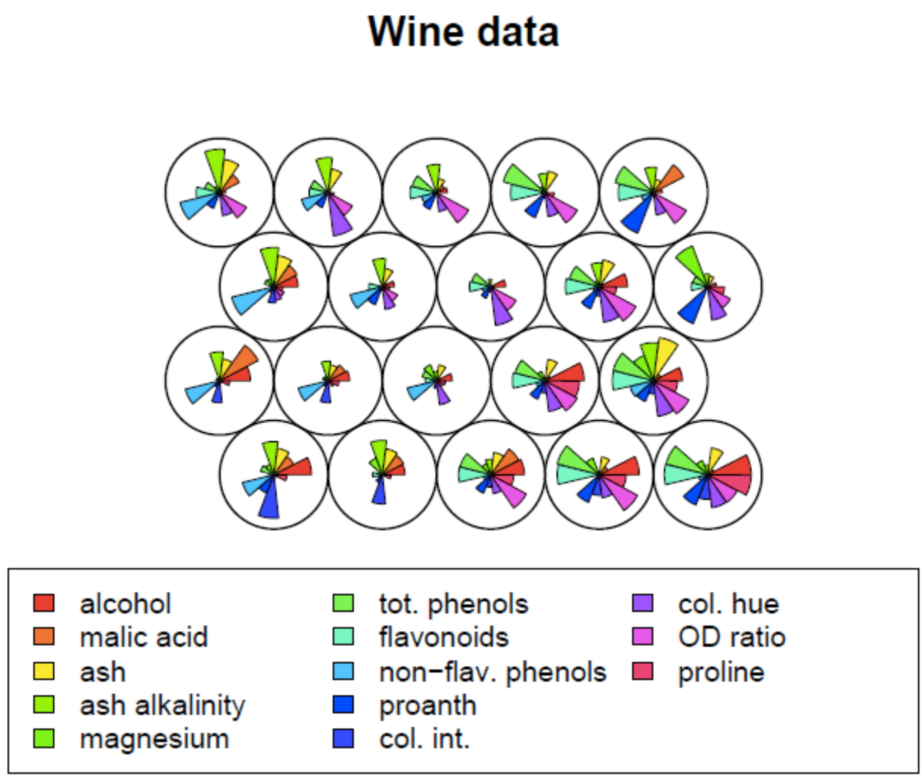


Large families have low income per inhabitant



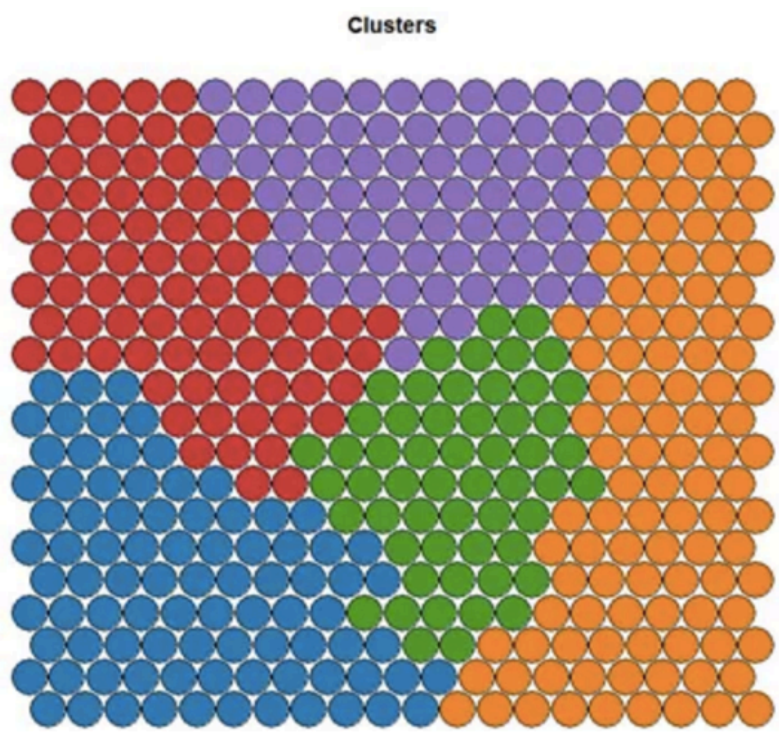
Visualizing Variable Values

- R can create a rose plot that shows the variable values for each node

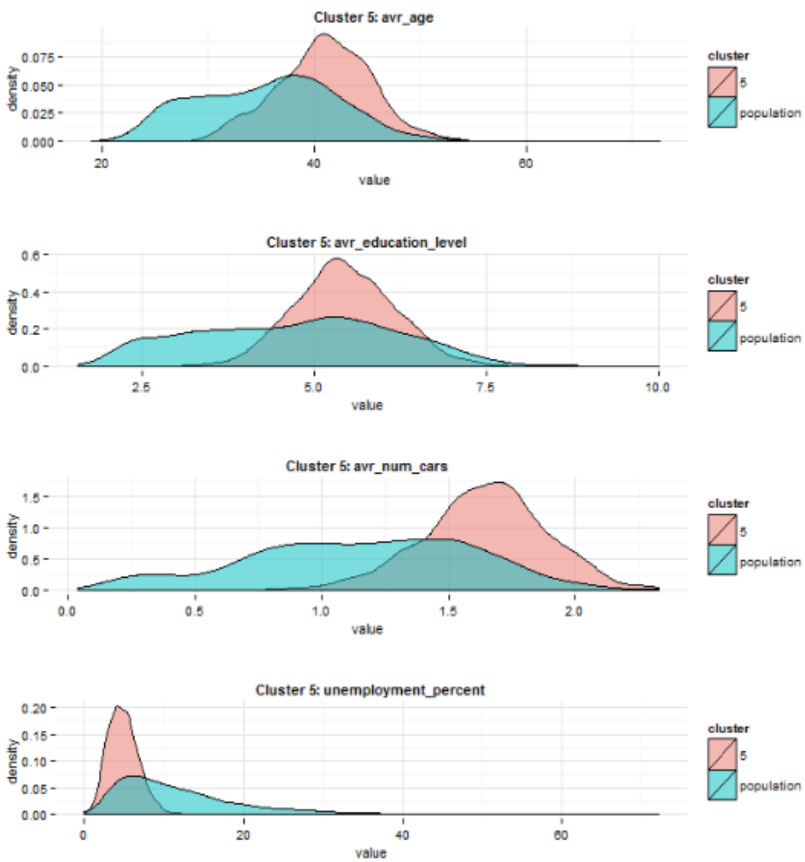


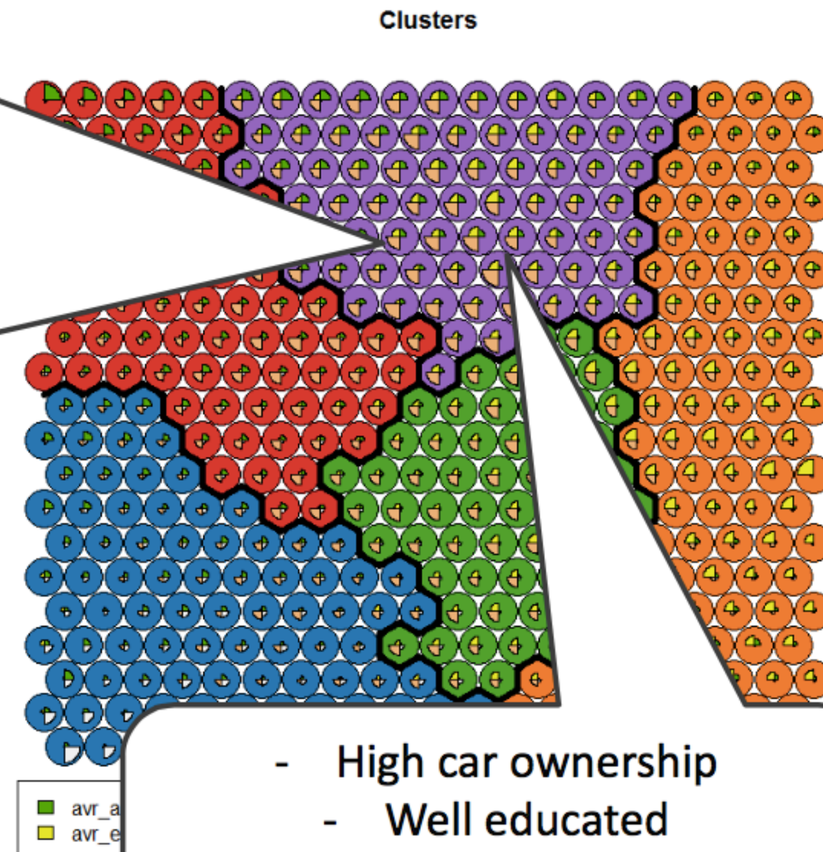
Hierarchical Clustering

- The node centroids can be used to hierarchically cluster nodes, to get clusters across the entire grid



Cluster 5 details





- High car ownership
- Well educated
- Relatively older demographic
- Larger households

"Commuter Belt"

Implementation

- Available in most statistical tools:
  - R packages: som and kohonen
  - Python: SOMPY
  - Matlab
  - SAS
  - Tensorflow
  - Open-source software

Further Reading

- Kohonen, Teuvo. 2001. *Self-organizing maps*. Berlin: Springer.
- Kohonen, Teuvo. 1982. “Self-organized formation of topologically correct feature maps”. *Biological Cybernetics*. 43(1): 59-69.
- “Kohonen’s self organizing feature maps”: http://bit.ly/1y7g40w
- “Self-organising maps for customer segmentation using R”: http://bit.ly/1mpErsh
- “Self-organizing maps”: http://bit.ly/1PaSXiR