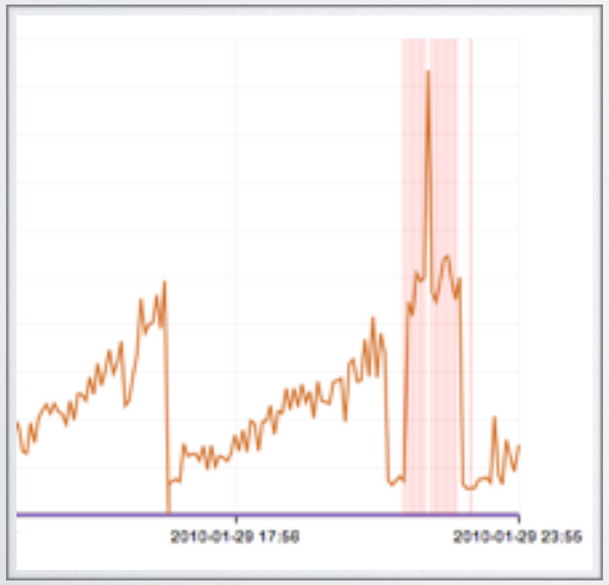
Clustering & Self Organizing Maps

Outline

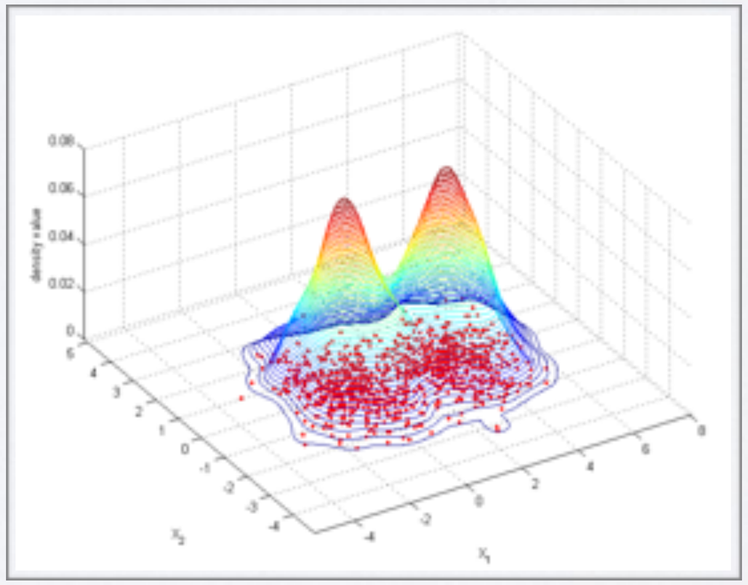
- Unsupervised Learning
- Clustering
  - Centroid-Based (k-means / ISODATA)
  - Distribution-Based (Gaussian Mixture Models)
  - Density-Based (DBSCAN / kNN)
  - Connectivity-Based (Hierarchical)
- Self Organizing Maps

Unsupervised Learning

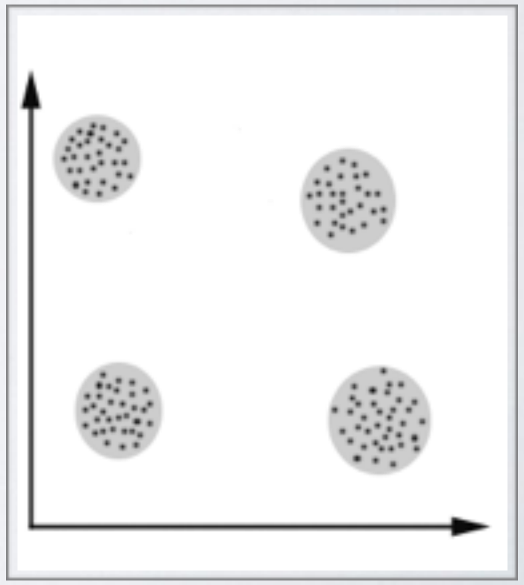
- All unlabeled data (usually because labels are expensive)
  - Learn from $p(x)$ instead of $p(y|x)$
  - Learn the structure of the data
- Problems:
  - Anomaly Detection
  - Density Estimation (next week)
  - Compression / Clustering



http://blog.serverdensity.com



http://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation



http://home.deib.polimi.it/

Clustering

- Goal
    - Find which instances belong to locally grouped regions (i.e. instances of distinct similarity)
- Applications
    - Marketing
    - Biology
    - Big Data
    - Image Processing
- Common Parameters
    - Distance (spread) of data
    - Number of instances in a cluster
    - Number of clusters

Clustering Challenges

- Parameter Selection
- Scalability (Dimensionality & Cardinality)
- Feature types (nominal, ordinal, categorical)
- Arbitrarily shaped clusters
- Imbalanced Clusters

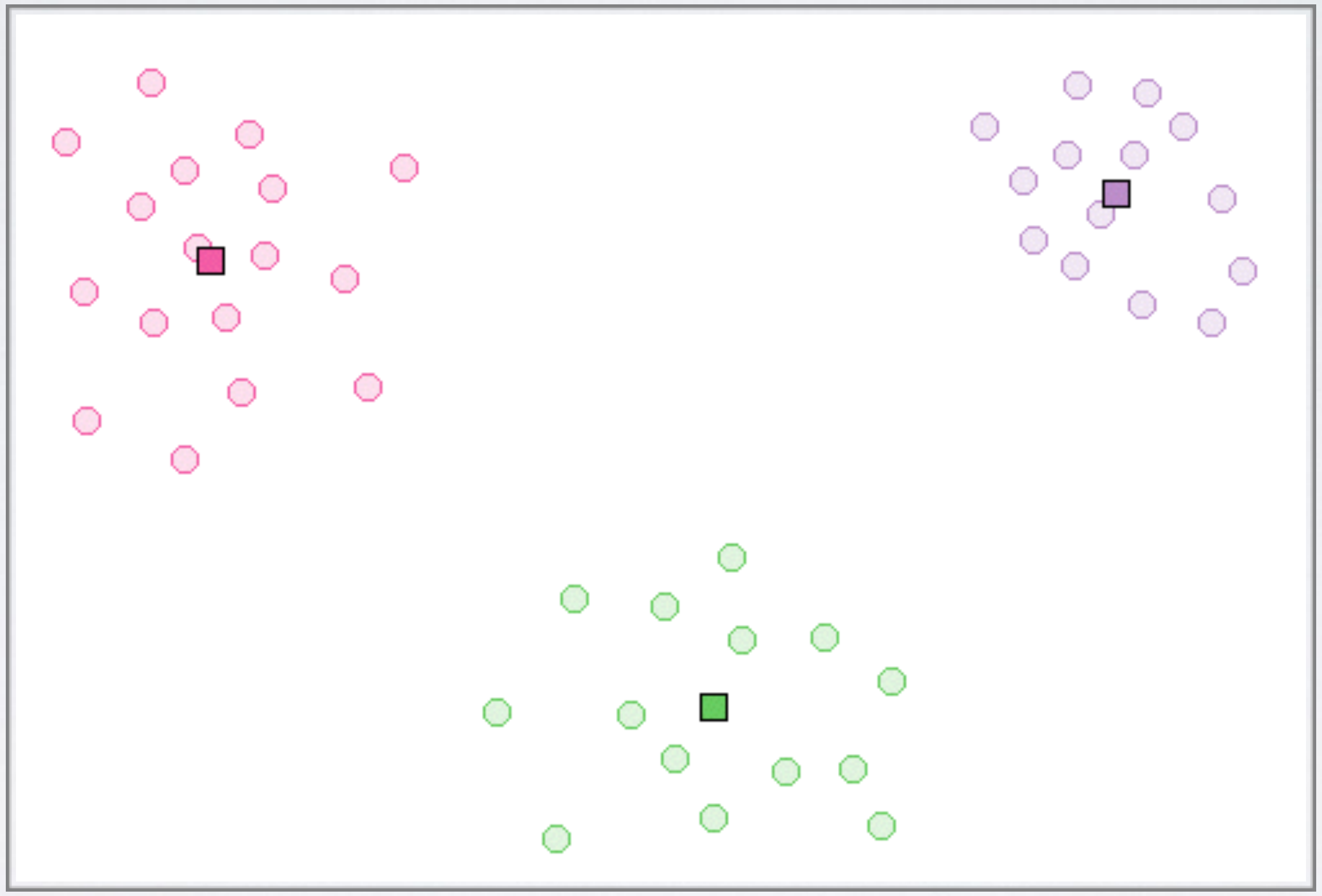
Centroid Based Clustering

- Assume all instances belong to one and only one "centroid"
- Each centroid represents a cluster
- Centroids lie in the center of the cluster they represent
- Implies circular clusters

k-means

- Input: k, number of clusters in data
- Procedure
    1. Initialize k cluster centroids
    2. Repeat until convergence:
        a. Assign all instances to their nearest cluster center
        b. Reset cluster centers to the mean of all assigned instances

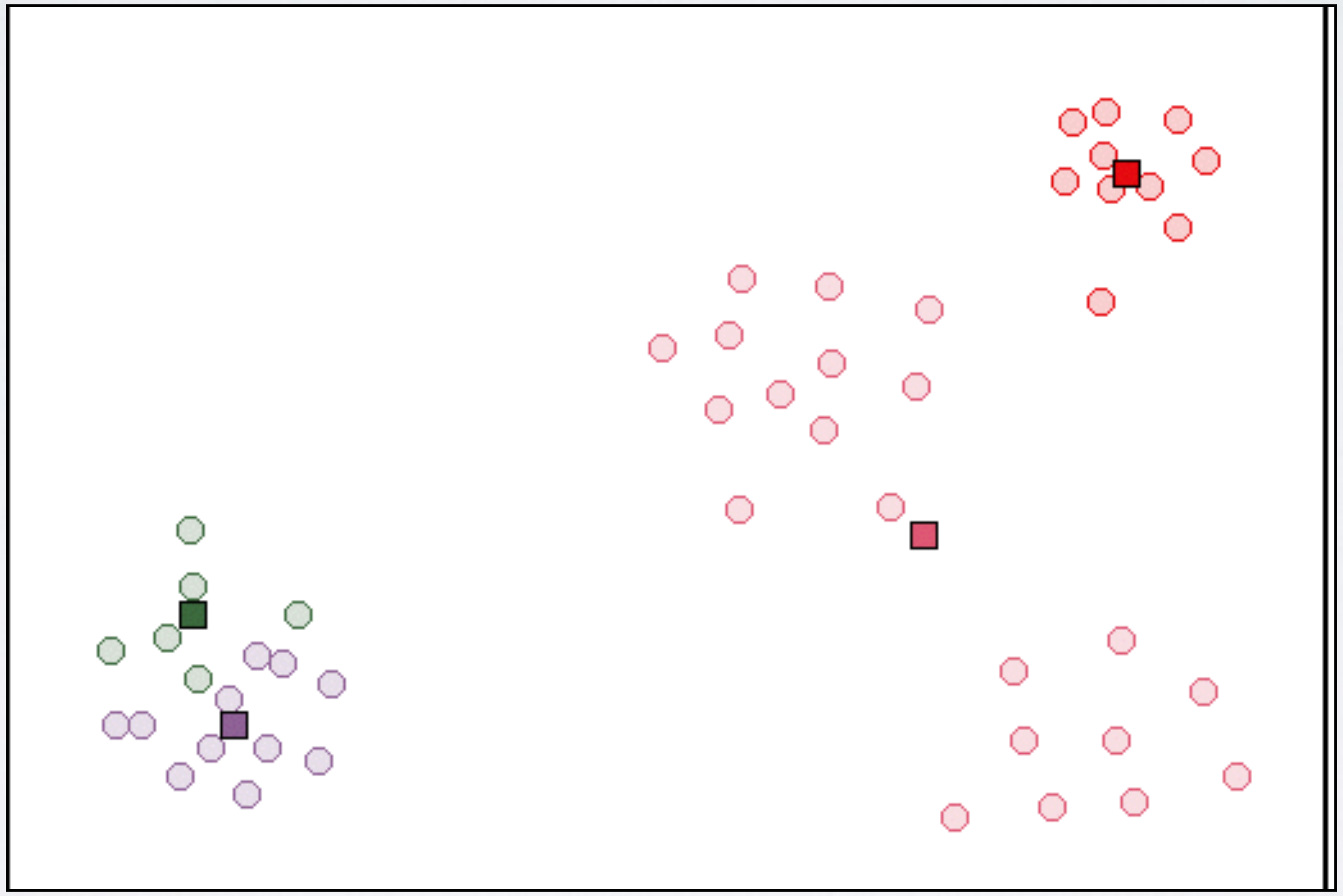
k-means (demo)



k-means

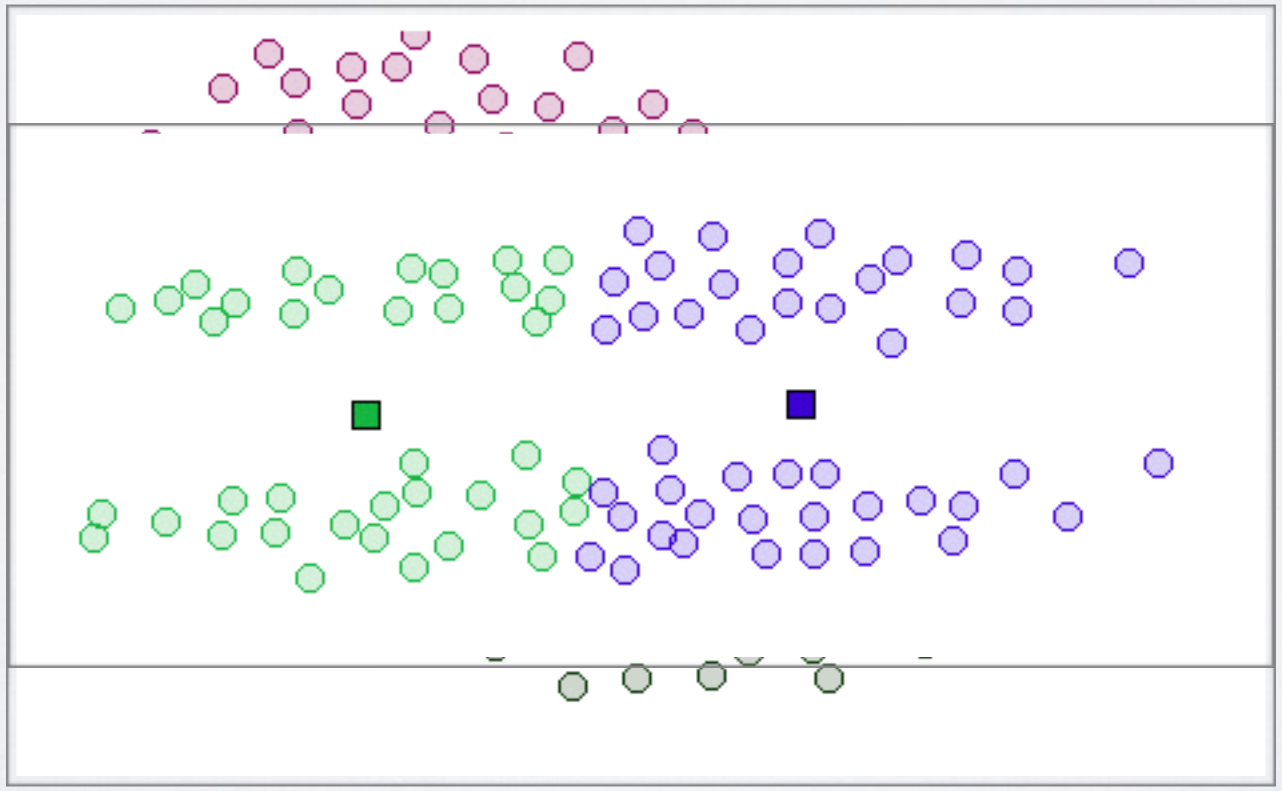
- Expectation
  - Determine which instances belong to which centroid
- Maximization
  - Maximize the likelihood that your centroid means represent their assigned data
- Free Parameters
  - k (number of centroids)
  - Distance Metric
  - Initial centroids

k-means (when it doesn't work)



Problems with k-means

- Number of clusters must be known
- May converge to incorrect solution (even when correct k is chosen)
- Highly dependent on initialization of centroids
- Complexity?
- Assumption that cluster center defines a cluster accurately



ISODATA

Iterative Self Organizing Data Analysis Technique yAy!

- Extension to k-means
  - No longer need exact number of clusters (just approximate)
- Procedure
  1. Run like k-means
  2. Split clusters whose variance is above some threshold
  3. Merge clusters that are close enough (by some threshold)
  4. Repeat
- Free Parameters:
  - N_D – desired (or approximate) number of clusters
  - N_{MIN_EX} – minimum number of instances per cluster
  - σ_s^2 – maximum spread of any cluster (for splitting)
  - D_{MERGE} – minimum distance between two clusters (for merging)
  - N_{MERGE} – maximum number of clusters that can be merged

ISODATA

1. Randomly initialize centroids and assign all instances to their closest centroid. Keep N_C up to date as current number of clusters
2. **Eliminate clusters** that contain less than N_{MIN_EX} examples. Reassign examples to their nearest clusters
3. **Calculate Cluster Parameters:** For each cluster, compute the center $\boldsymbol{\mu_k}$**,** the average distance, $\mathbf{d_k}$ between all assigned examples and μ_k, and the cluster's axis variance $\sigma_k^2(d^*)$ where d* is the axis with max variance
4. **Splitting:** For all clusters with $\sigma_k^2(d^*) > \sigma_s^2$
    1. If $d_k > d_{avg}$ AND ($N_K > 2N_{MIN\_EX}$ OR $N_C < N_D / 2$) split cluster on d*
        1. $\mu_{k1}(d^*) = \mu_k(d^*) + \epsilon\sigma_k^2(d^*)$ $\mu_{k2}(d^*) = \mu_k(d^*) - \epsilon\sigma_k^2(d^*)$ $0 < \epsilon < 1$
        2. Reassign examples to appropriate centers
5. **Merging:** Compute all distances between cluster centers, Dij. For all, Dij < D_{MERGE},
    1. If in this iteration, clusters i and j haven't already been merged AND (not more than N_{MERGE} merges have occurred OR $N_C > 2N_D$)
        1. Merge clusters i and j, compute new means, assign instances accordingly, and reassign instances
6. Repeat from step 2 until convergence

ISODATA

- Commonly used in image processing (remote sensing)
- Can eliminate clusters with few examples
- Can merge / divide clusters when needed
- Problems
  - Circular cluster assumption
  - Lots of free parameters (on which performance is highly dependent)
  - Increased computational complexity from k-means
  - Convergence is not guaranteed
- Usually ISODATA is ran multiple times with different parameters and the clustering with minimum MSE is chosen

Distribution Based Clustering

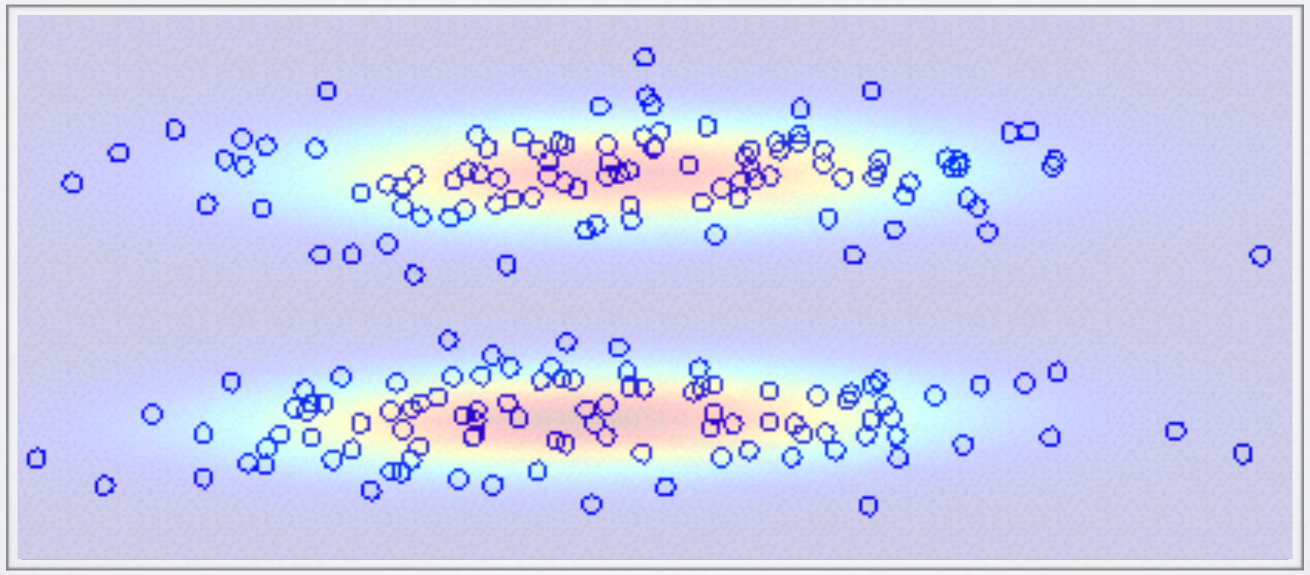
- Assume each cluster follows a known distribution
- The parameters of the distribution are unknown and are to be estimated
- Each instance contributes to the location of all clusters

Gaussian Mixture Models

- Assume now that each "centroid" represents a Gaussian distribution
  - We now have mean (**μ**), covariance (**Σ**), and mixing coefficient (**π**), for each distribution
  - Instead of assigning each instance to a center, we calculate $p(x_i|\mu_k, \Sigma_k, \pi_k)$
- Using expectation and maximization, we maximize the total likelihood

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \quad \theta = \{\mu_k, \Sigma_k, \pi_k \forall k\}$$

Why is this multiplied?



Density Based Clustering

- Assume clusters are above some density threshold
- Connect instances in regions where the density does not fall below this threshold
- Different clusters are separated by areas of low density

DBSCAN

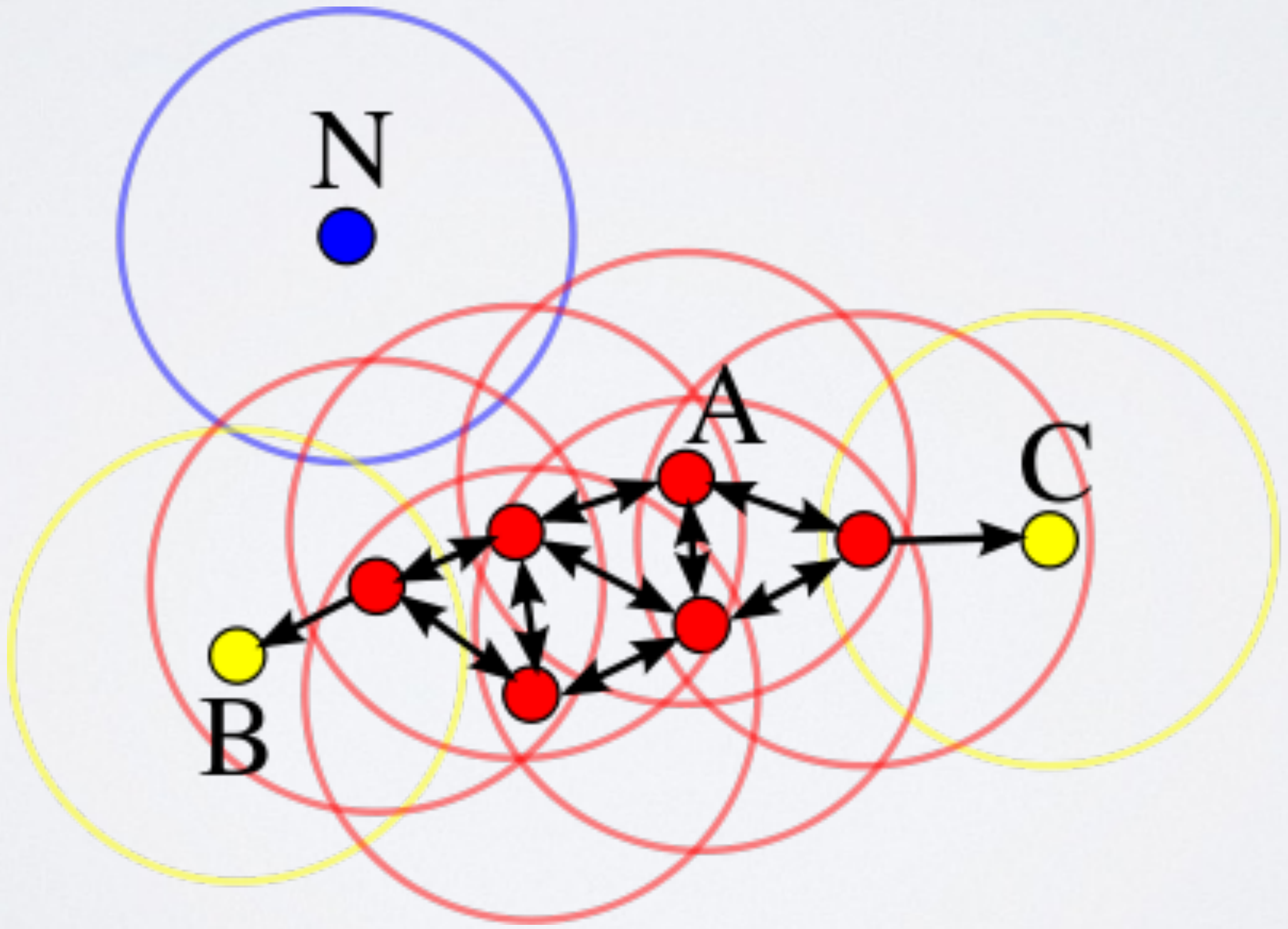
- Connect instances to nearby instances if there are enough of them in a neighborhood
- **Parameters:** minimum number of instances required in a neighborhood (n), and neighborhood radius (r)

Procedure:

1. Initialize X_U, the set of unchecked instances, as all of the data, X
2. Create a "neighborhood" centered at a test instance $x_T \in X_U$ containing the set of instances X_N within distance r of x_T
   a. Remove x_T from X_U
   b. If the number of instances in X_N is larger than n, connect all $x \in X_N$ to x_T
   c. Recursively repeat step 2 for all $x \in X_N \cap X_U$ until all connected instances are checked
3. Repeat step 2 with a new test instance $x_T \in X_U$

DBSCAN

- Directly density reachable
- Density reachable
- Density connected



http://en.wikipedia.org/wiki/DBSCAN

DBSCAN

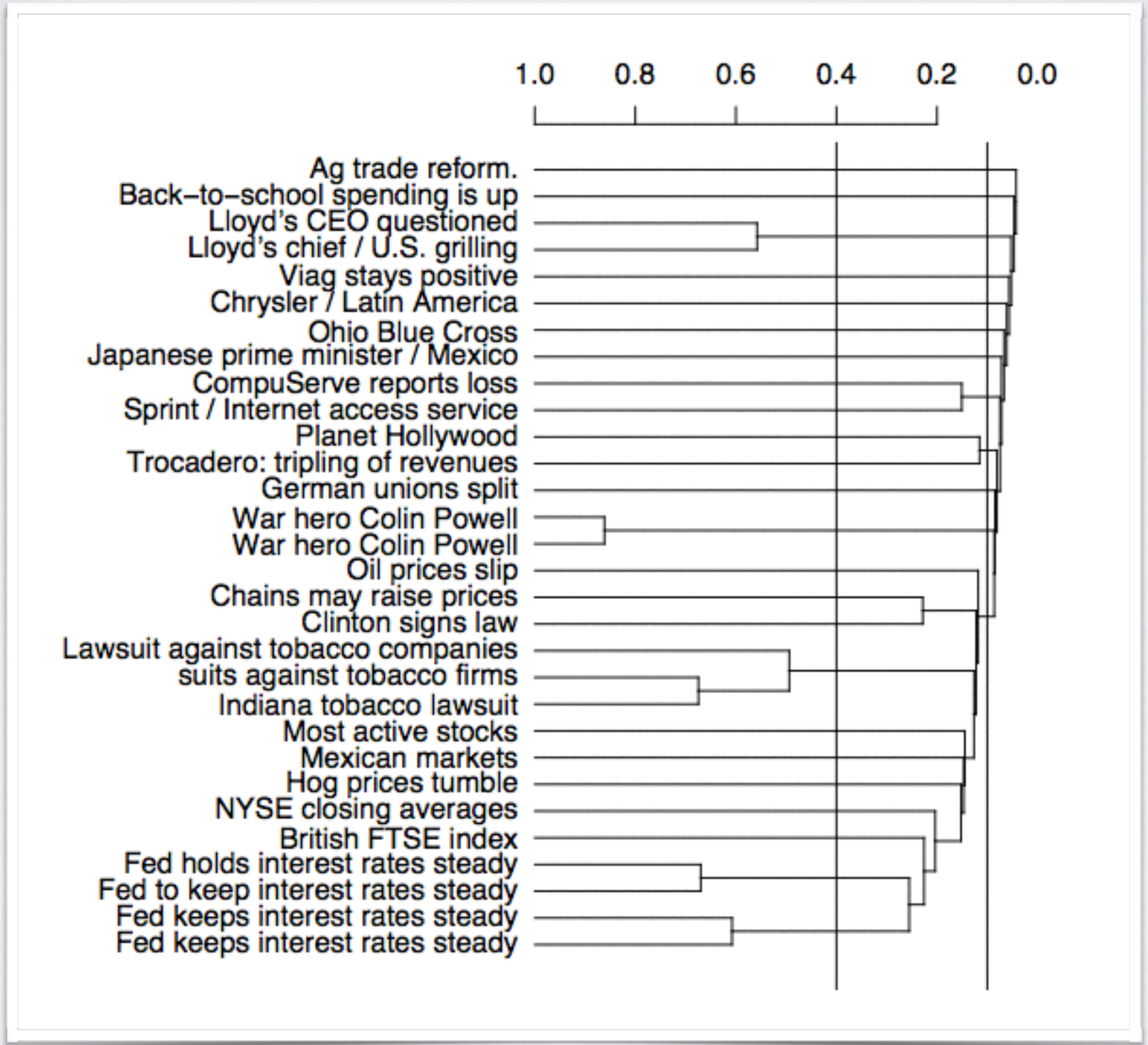
- Free parameters:
  - r – radius of neighborhood
  - n – number of instances in local region to propagate cluster
- Benefits
  - Don’t need to know number of clusters
  - No assumptions on shape of clusters (what is the primary assumption?)
- Problems
  - Accidentally joining clusters that should be separate (noise) – how is this alleviated?
  - Need to approximate spread of the data
  - Complexity?

kNN?

Hierarchical

- Types
  - **Agglomerative** (bottom-up)
  - Divisive (top-down)
- Initially consider all instances as "clusters"
- Parameter: distance measure between two clusters $d(c_i, c_j)$
  - Single linkage (distance between nearest two instances)
  - Complete linkage (distance between farthest two instances)
  - Group Average (average distance between all instances)
- Group the most similar clusters, one at a time, based on d

Dendrogram



http://nlp.stanford.edu/IR-book/pdf/17hier.pdf

Hierarchical Agglomerative

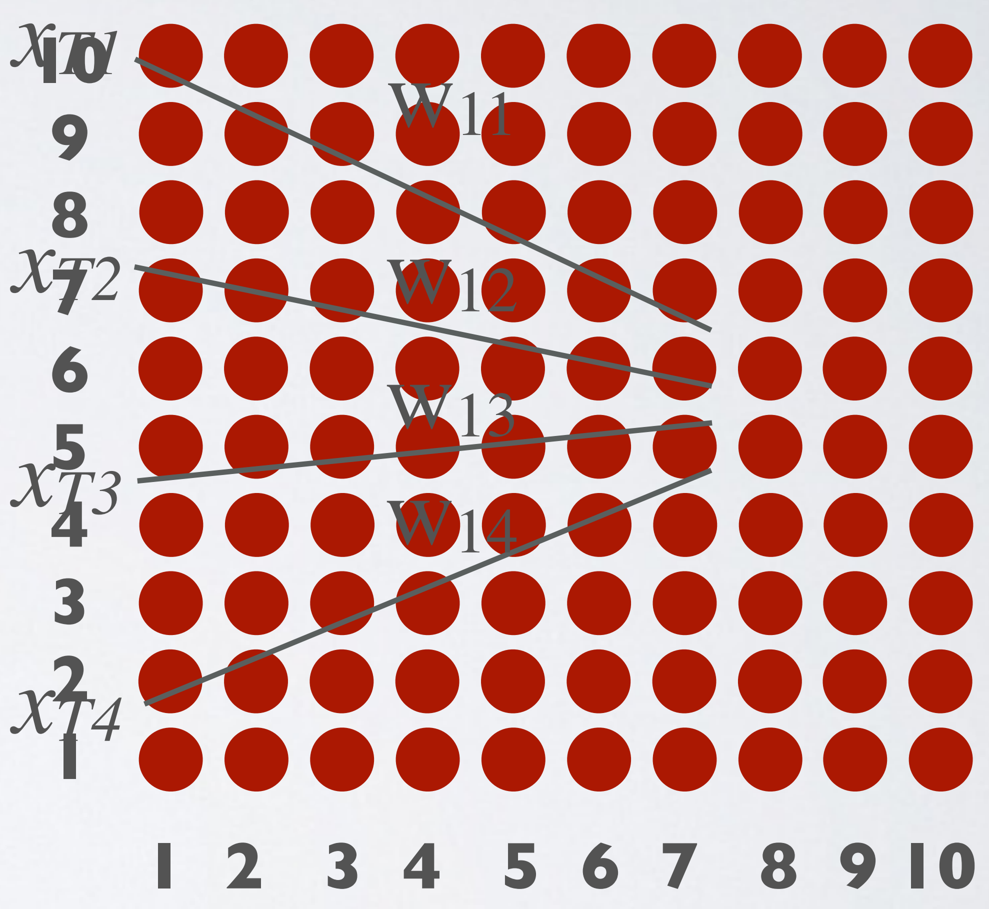
- How do we decide actual clustering?
- The dendrogram needs to be cut somewhere, because a bunch of different possible clusterings doesn't tell us much
- Cut criterion
  - Threshold $d(c_i, c_j)$ or combination similarity
  - Specify number of clusters
  - $K = \underset{K'}{arg\,min}[RSS(K') + \lambda K']$
- Problems with Hierarchical Agglomerative Clustering?

Self Organizing Maps (SOMs)

- Teuvo Kohonen (University of Finland – 1982)
- Clustering, Visualization, & Dimensionality Reduction
- Neural Network (sort of)
  - Contains a lattice of artificial "neurons" with input weights
  - Iteratively trained like a neural network
  - No back propagation
  - No output labels
  - No activation functions
- Translate the instances in the feature space to nodes in the lattice space

SOMs

- Connected lattice of nodes ("neurons")
- Each node has a position
  - x, y coordinate in lattice
- Each node has a weight vector
  - Initialized randomly
  - Same dimensionality as the data
  - Map the feature space to the lattice



SOM Procedure

In General:

1. Randomly initialize the weight of each node
2. Sample a random instance, x_T
3. Determine which node's weight is closest to x_T. This node is the best matching unit (BMU)
4. Create a kernel function, $\theta(t)$, centered at the BMU with radius σ_t

$$\theta(t) = e^{-\frac{d^2}{2\sigma_t^2}} \qquad \sigma_t = \sigma_0 e^{-\frac{t}{\lambda}}$$

5. Update all nodes within σ_t of the BMU

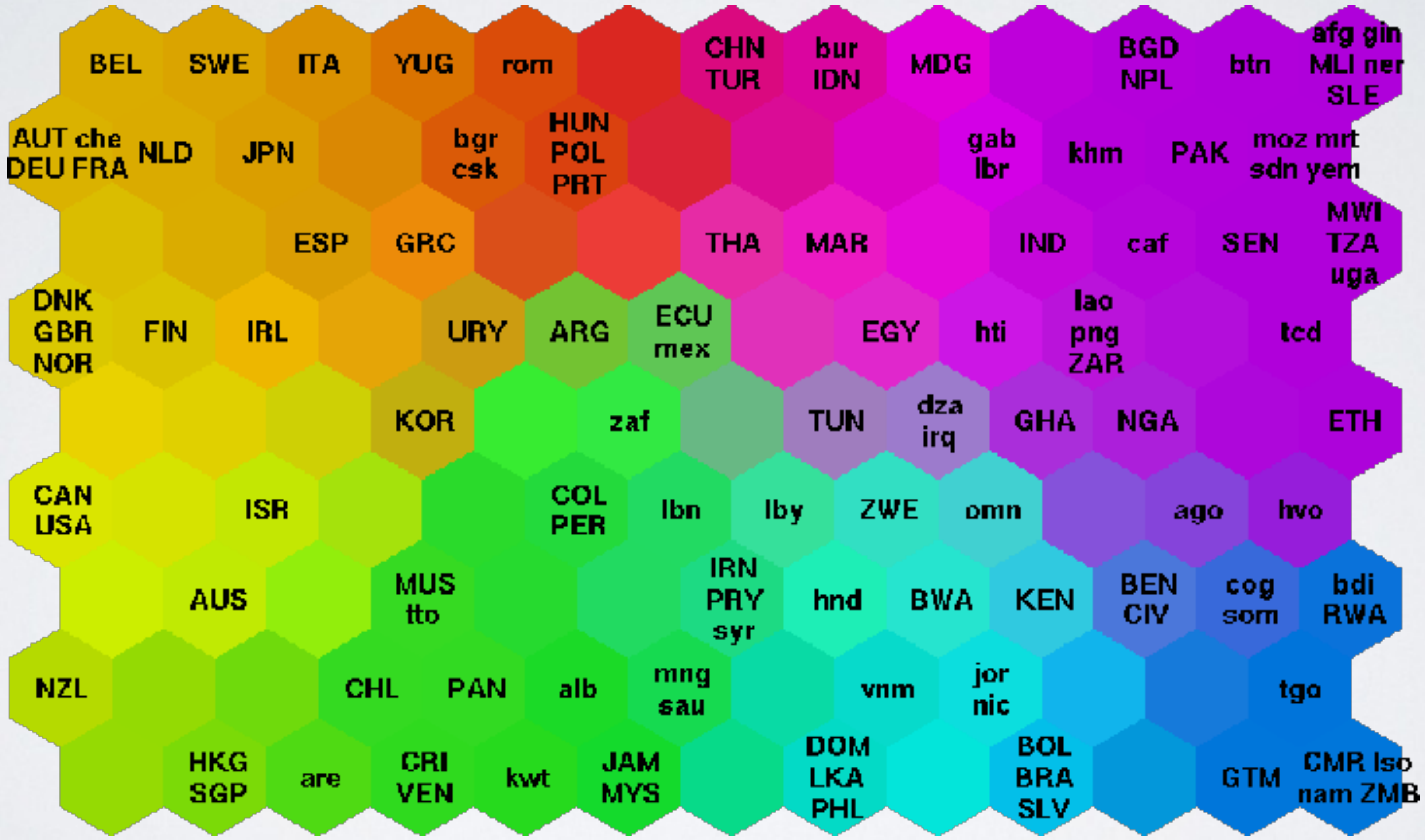
$$w_i(t+1) = w_i(t) + \theta(t)L(t)(x_T - w_i(t)) \qquad L(t) = L_0 e^{-\frac{t}{\lambda}}$$

6. Return to step 2 and repeat N times

SOMs

- Maps data from a continuous input space to a low dimensional discrete output space
- Preserves topological structure of data
- Competitive & Cooperative
- Can be used to classify new data into the lattice
- Parameters:
  - Number of nodes and dimensionality of lattice (usually 2D)
  - Initial neighborhood radius
  - Neighborhood Kernel function
  - Time decay constant λ
  - Initial learning rate
  - Distance measures
  - Number of iterations

SOMs for Visualization



http://www.cis.hut.fi/research/som-research/worldmap.html

References

- Clustering
  - http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/
  - http://www.cs.unc.edu/~lazebnik/fall09/clustering_techniques_and_applications.pptx
  - http://research.cs.tamu.edu/prism/lectures/pr/pr_l15.pdf
  - http://www.cs.princeton.edu/courses/archive/spr08/cos424/slides/clustering-2.pdf

- Self Organizing Maps
  - http://www.cs.bham.ac.uk/~jxb/NN/l16.pdf
  - http://www.ai-junkie.com/ann/som/som1.html
  - http://davis.wpi.edu/~matt/courses/soms/