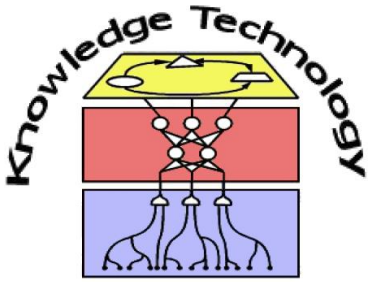
Neural Networks

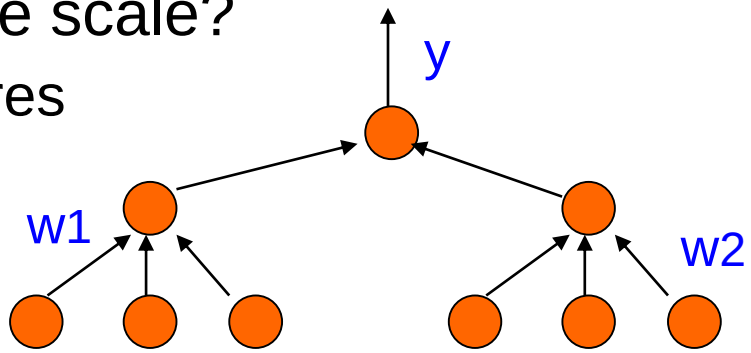
Lecture 12: Unsupervised Learning



http://www.informatik.uni-hamburg.de/WTM/

Some issues with backpropagation

- Where does the supervision come from?
  - Most data is unlabelled
    - …The vestibular-ocular reflex is an exception….
- How well does the learning time scale?
  - Its is impossible to learn features for different parts of an image independently if they all use the same error signal.



- Can neurons implement backpropagation?
  - Not in the obvious way but perhaps at a higher level cognitive view (to some extent)

Three kinds of learning

- Supervised Learning: this models p(y|x)
  - Learn to predict a real valued output or a class label from an input.
- Reinforcement learning: tries to be successful at the end
  - Choose actions that maximize payoff
- Unsupervised Learning: this models p(x)
  - Build a causal generative model that explains why some data vectors occur and not others

or

  - Discover interesting features; separate sources that have been mixed together; find temporal invariants etc. etc.

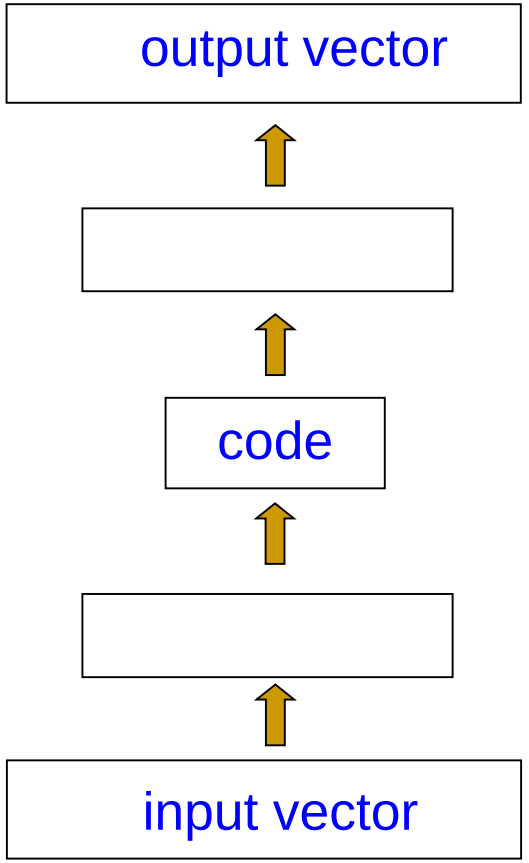
The goals of unsupervised learning

- The general goal of unsupervised learning is to ***discover*** useful ***structure*** in large data sets, without requiring a desired target output or reinforcement signal.
  - It is not obvious how to turn this general goal into a specific objective function that can be used to drive the learning.

- A more specific goal:
  - Create representations that are better for subsequent supervised or reinforcement learning.

Using backprop for unsupervised learning

- Try to make the output the same as the input in a network with a central bottleneck.
  - The activities of the hidden units in the bottleneck form an efficient code.
    - The bottleneck does not have room for redundant features.
  - Good for extracting independent features (as in the family trees)
  - **Autoassociator** (same input and output)



Self-supervised backprop in a linear network

- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared reconstruction error.
  - This is what PCA (Principal Components Analysis) does.

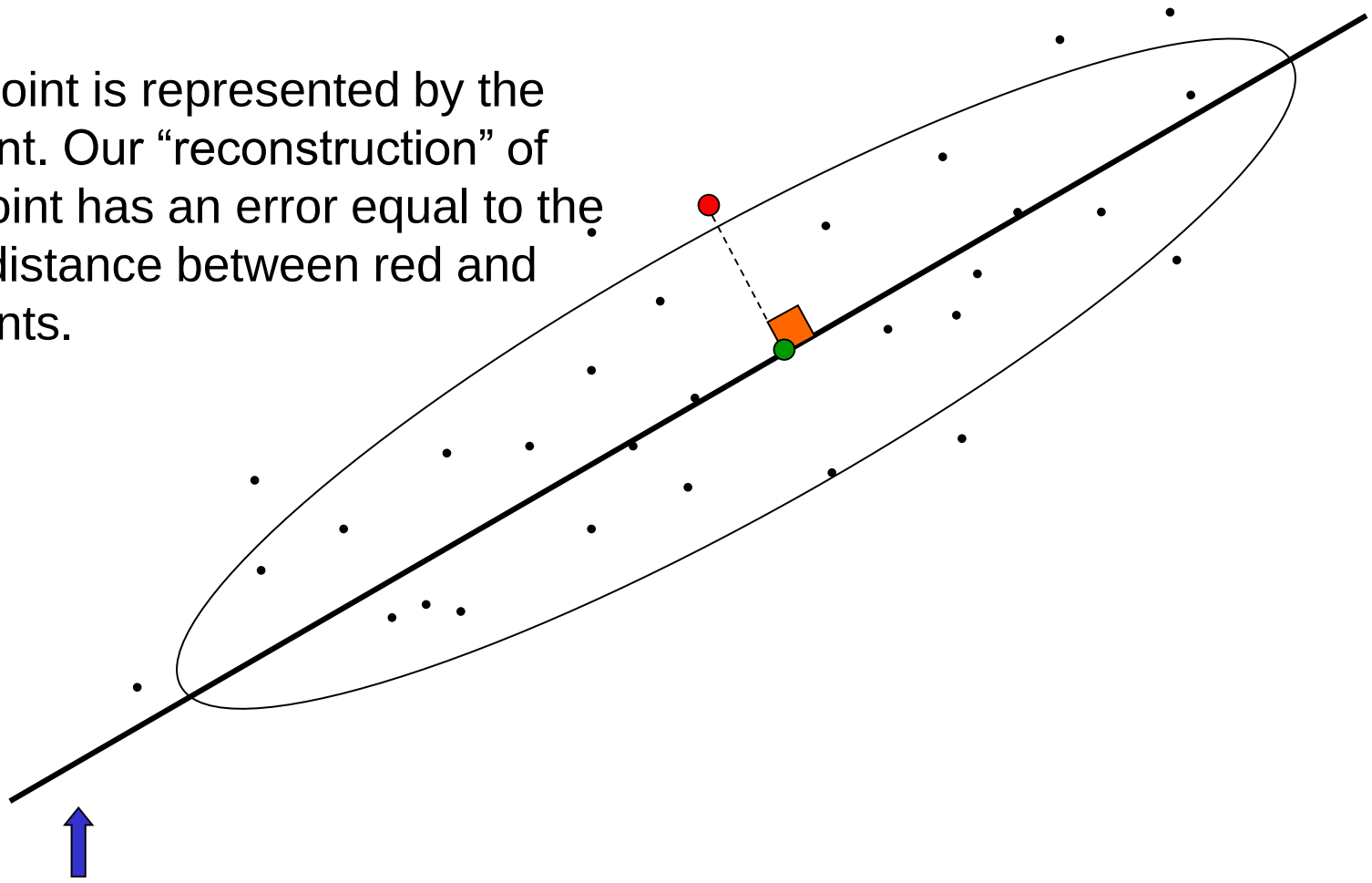
- The M hidden units will span the same space as the first M principal components found by PCA
  - Their weight vectors may not be orthogonal
  - They will tend to have equal variances

Principal Components Analysis

- This takes N-dimensional data and finds the M orthogonal directions in which the data have the most variance
- These M principal directions form a subspace.
- We can represent an N-dimensional datapoint by its projections onto the M principal directions
  - This loses information about where the datapoint is located in the remaining orthogonal directions.
- We reconstruct by using the mean value (over all the data) on the N-M directions that are not represented.
  - The reconstruction error is the sum over all these unrepresented directions of the squared differences from the mean.

A picture of PCA with N=2 and M=1

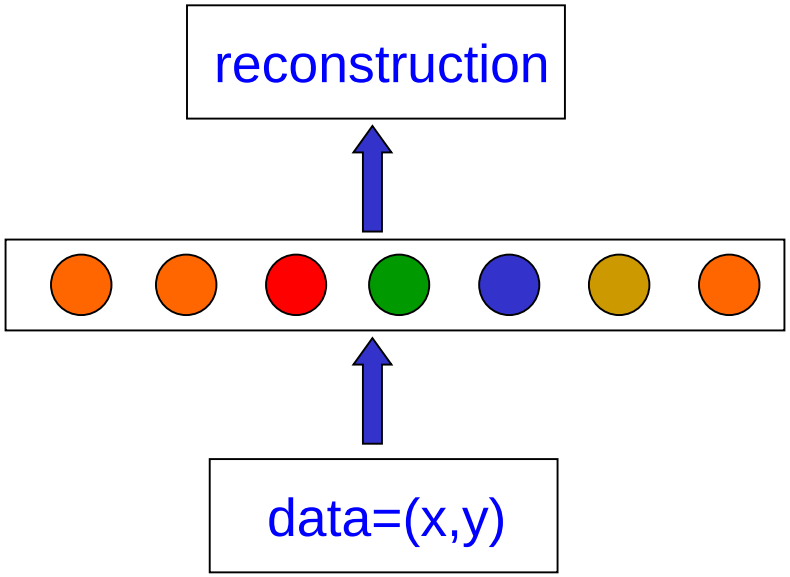
The red point is represented by the green point. Our “reconstruction” of the red point has an error equal to the squared distance between red and green points.

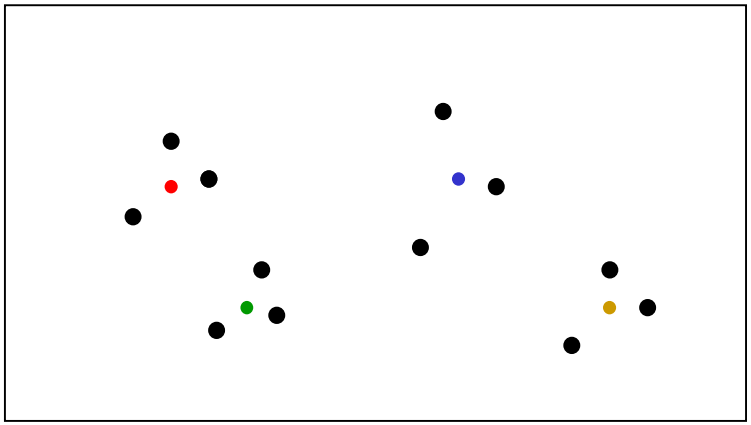


First principal component:
Direction of greatest variance

Self-supervised backprop and clustering

- If we force the hidden unit whose weight vector is closest to the input vector to have an activity of 1 and the rest to have activities of 0, we get clustering.
- The weight vector of each hidden unit represents the center of a cluster.
- Input vectors are reconstructed as the nearest cluster center.



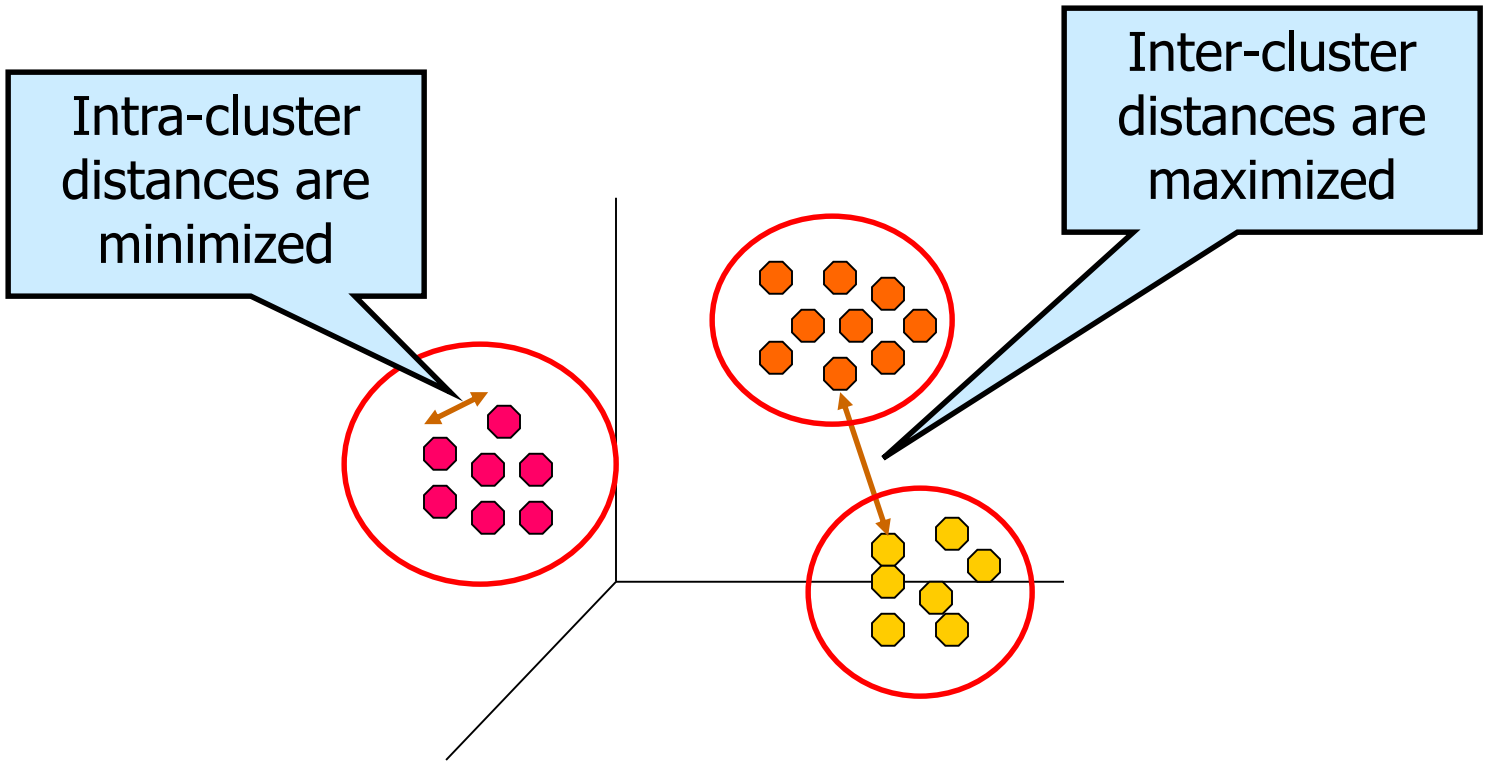


Clustering questions and motivation

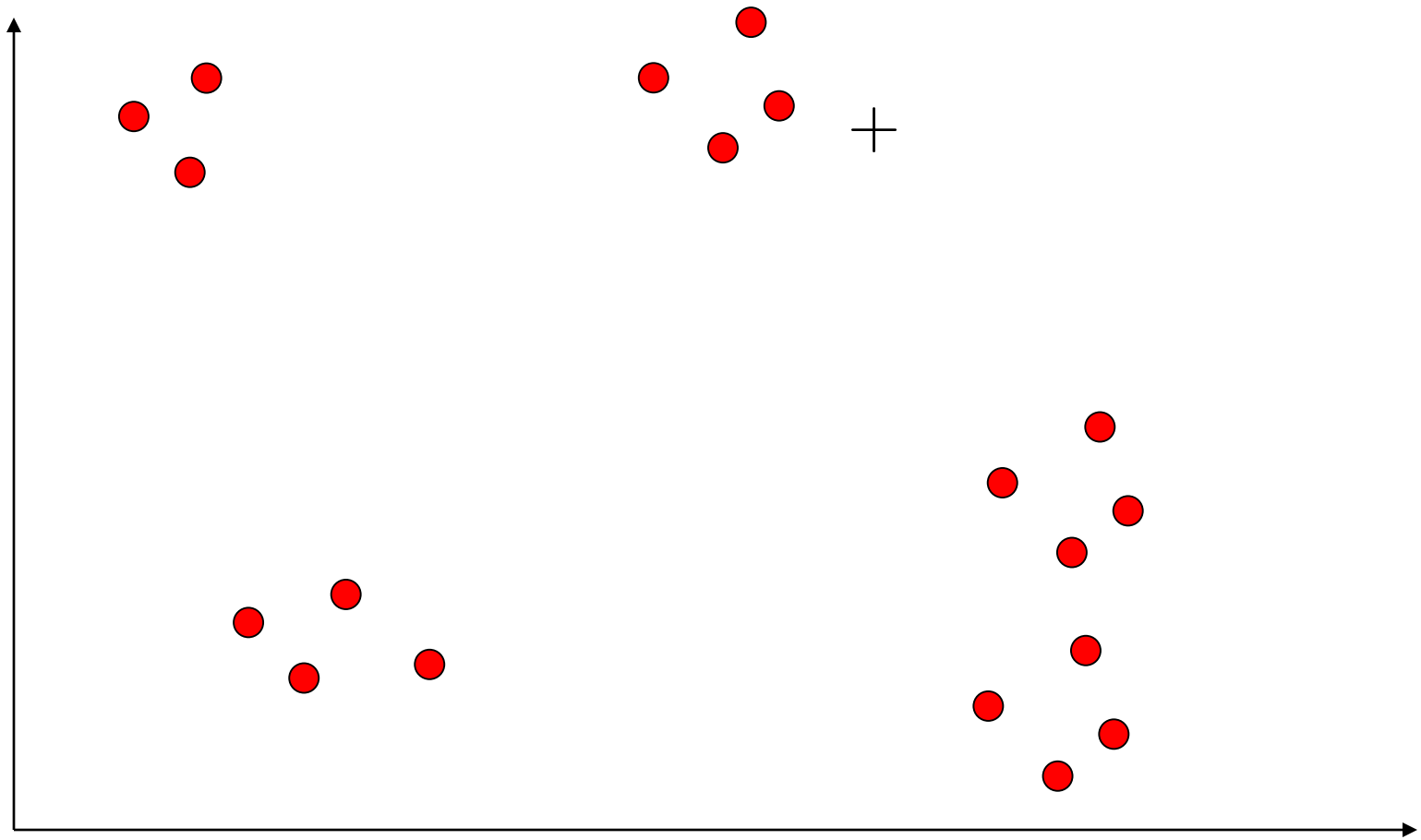
- We assume that the data was generated from a number of different classes. The aim is to cluster data from the same class together.
- How do we decide the number of classes?
  - Why not put each datapoint into a separate class?
  - What is the payoff for clustering things together?
- What if the classes are hierarchical?
- What if each datavector can be classified in many different ways? A one-out-of-N classification is not nearly as informative as a distributed code.

What is Cluster Analysis?

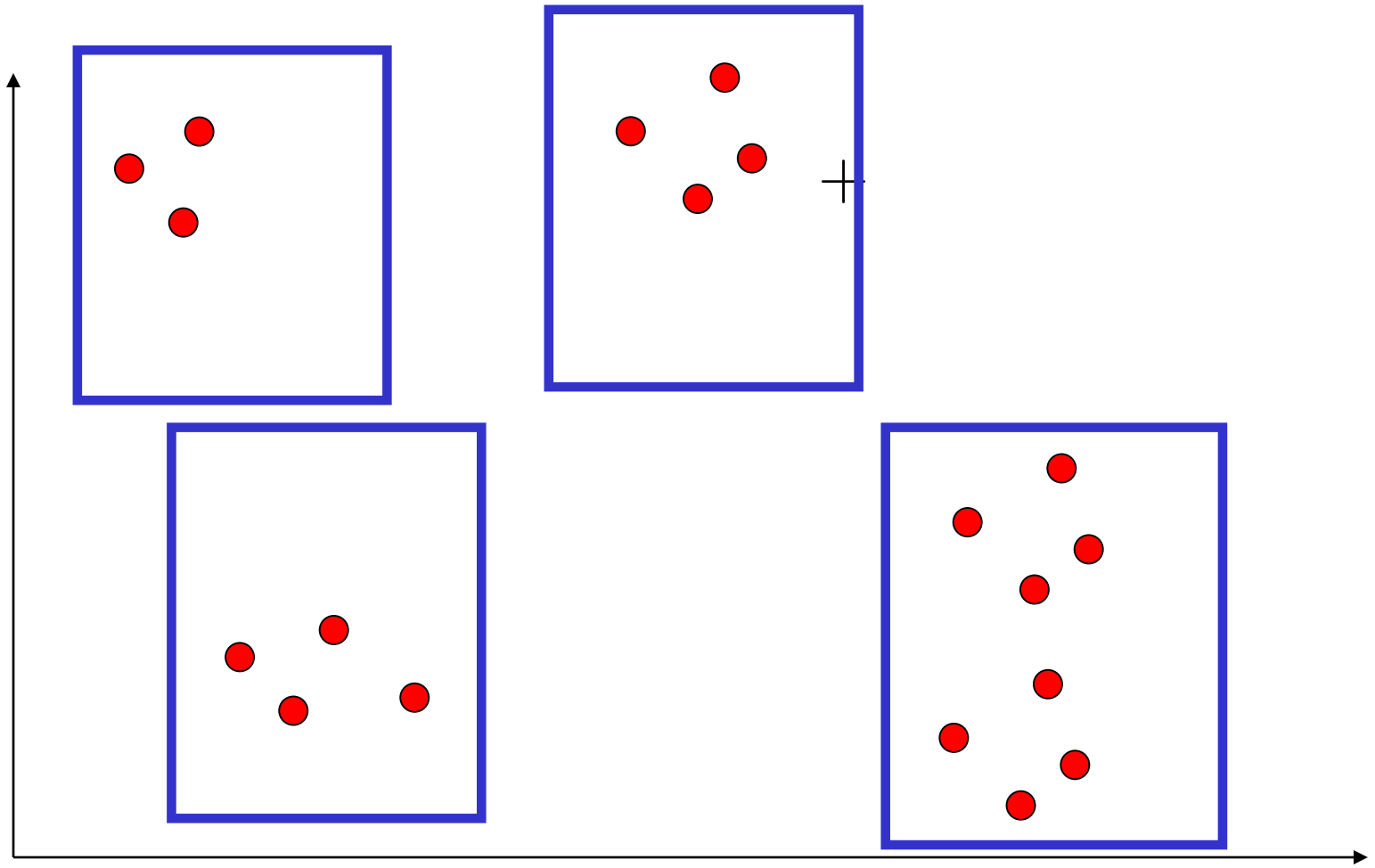
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



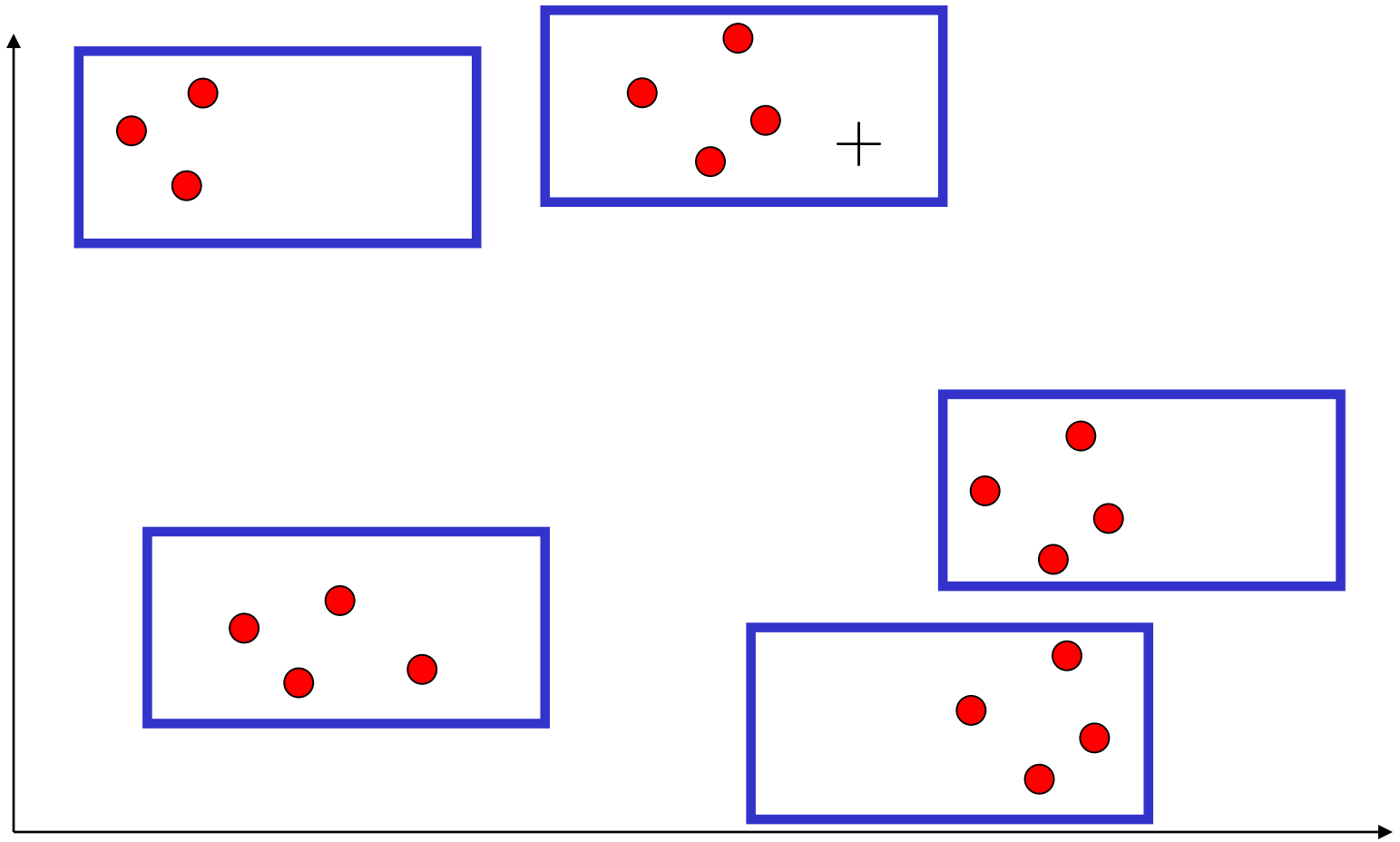
Clustering: Classes unknown



Clustering of previous objects



Clustering of previous objects



Clustering Problem

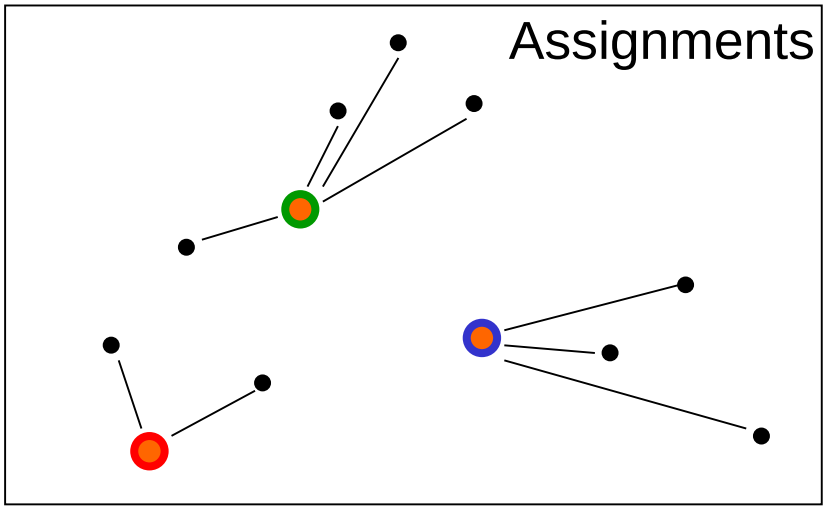
- Clustering differs from classification in that clusters for input data are not known a priori.

- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of input data (tuples) and an integer value k, the ***Clustering Problem*** is to define a mapping $f:D\rightarrow\{1,..,k\}$ where each t_i is assigned to one cluster K_j, $1<=j<=k$.

- A ***Cluster***, K_j, contains precisely those tuples mapped to it.

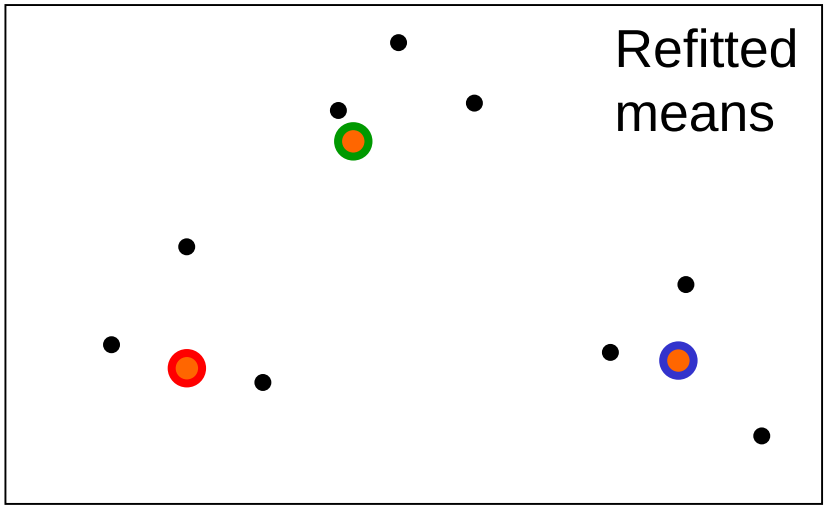
The k-means algorithm (reminder)

- Assume the data lives in a Euclidean space.
- Assume we want k classes.
- Assume we start with randomly located cluster centers



The algorithm alternates between two steps:

1. ***Assignment step***: Assign each datapoint to the closest cluster.
2. ***Refitting step***: Move each cluster center to the center of gravity of the data assigned to it.



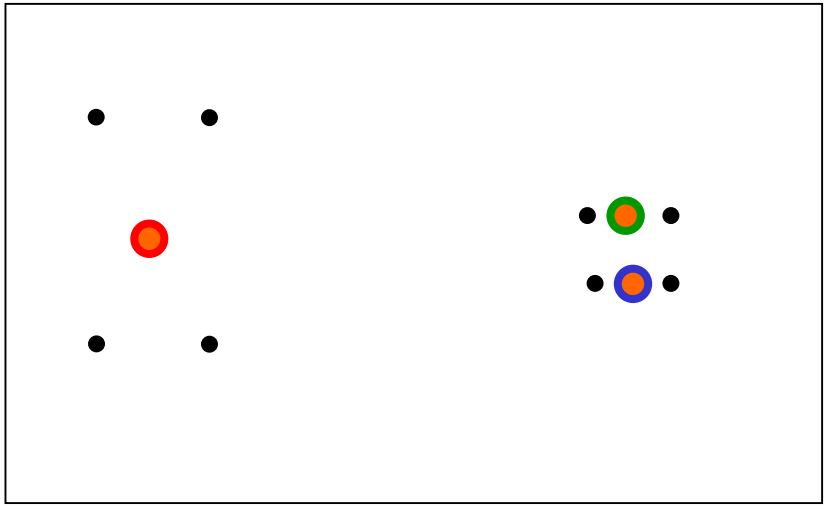
Why k-means converges

- Change of assignments reduces the sum squared distances of the datapoints to their assigned cluster centers.
- Moving a cluster center reduces the sum squared distances of the datapoints to their assigned cluster centers.
- If the assignments do not change in the assignment step, we have converged.

Local Minima

- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves: Simultaneously merge two nearby clusters and split a big cluster into two.

A bad local optimum



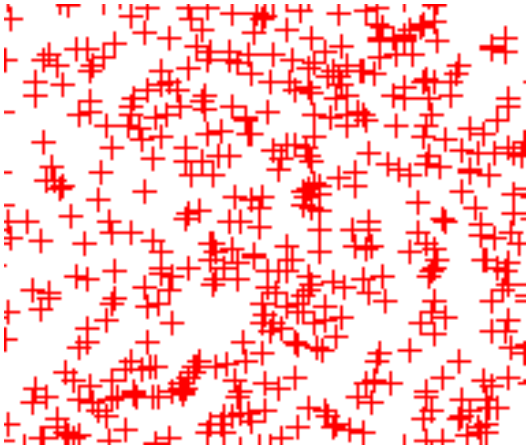
The Self-Organising Map

- One neural method of clustering was proposed by Kohonen
- Idea inspired by the way that mappings are learnt in the topographic feature maps found in many brain areas
- These ***feature maps*** consist of one- or two-dimensional sheets of neurons with lateral interactions

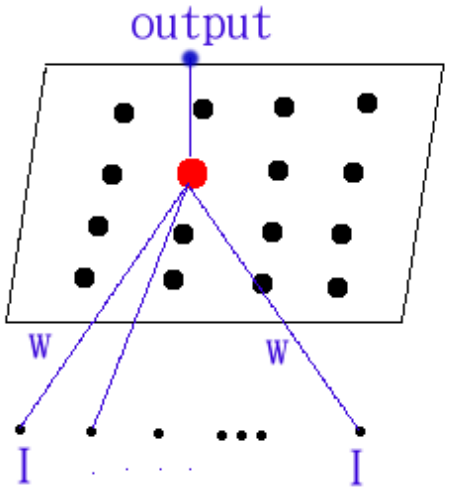
Feature Maps

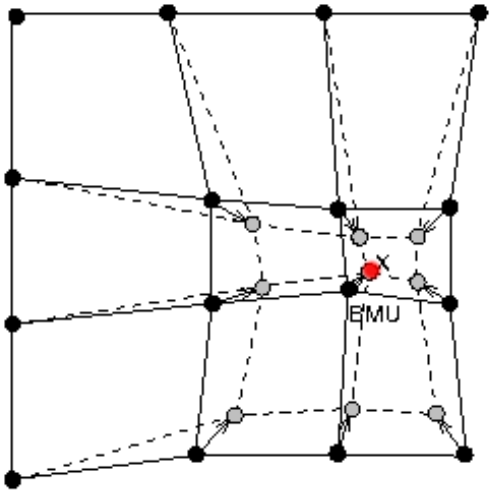
- Patterns that are similar (‘close together’) excite neurons that are near to each other
- Patterns that are very different excite neurons far away
- This is known as ***topology preservation***
- The ordering of the inputs is preserved
  - if possible (perfectly topology-preserving)

Representation in Self-organizing Networks



Raw Data



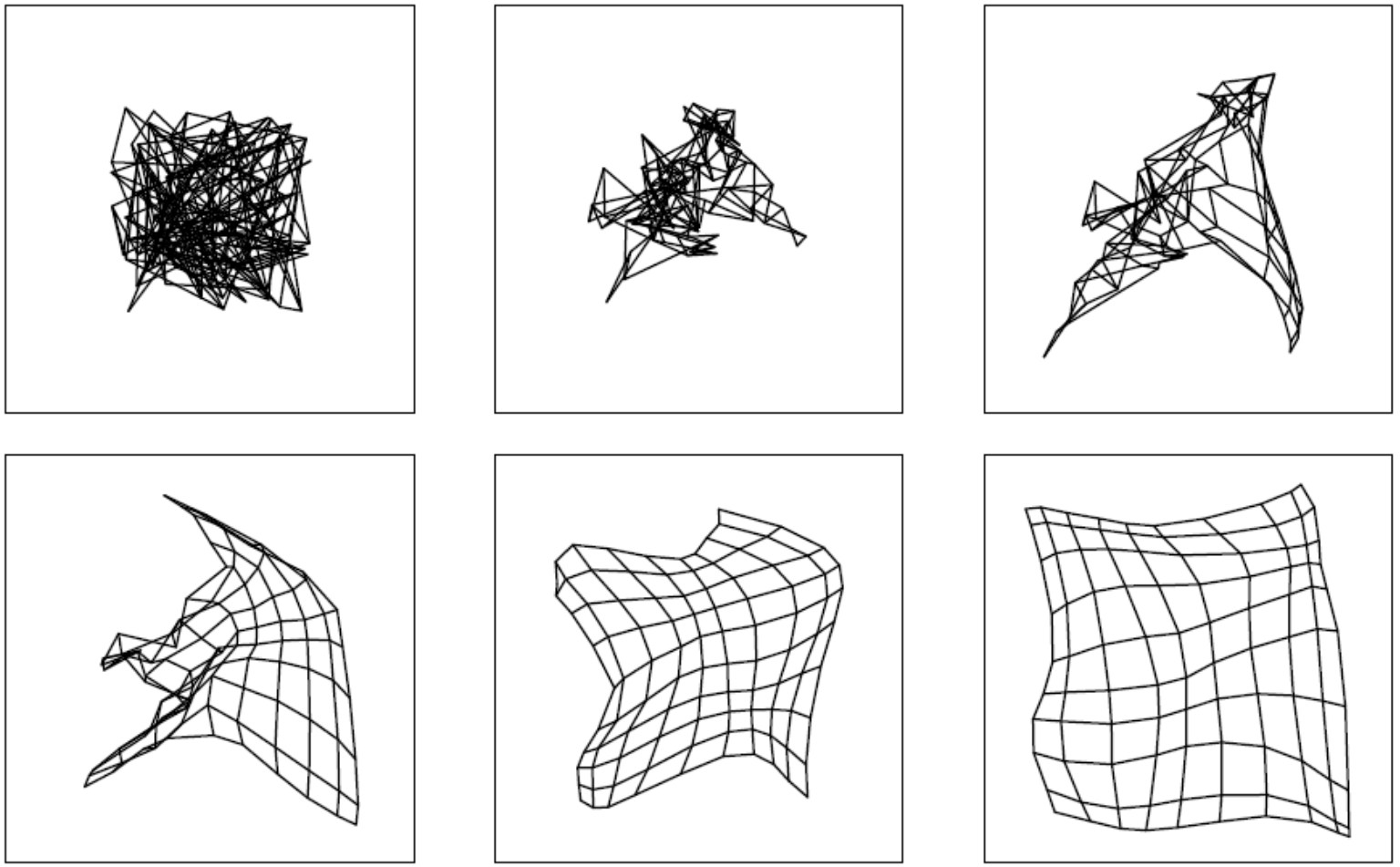


While computational bounds are not exceeded **do**

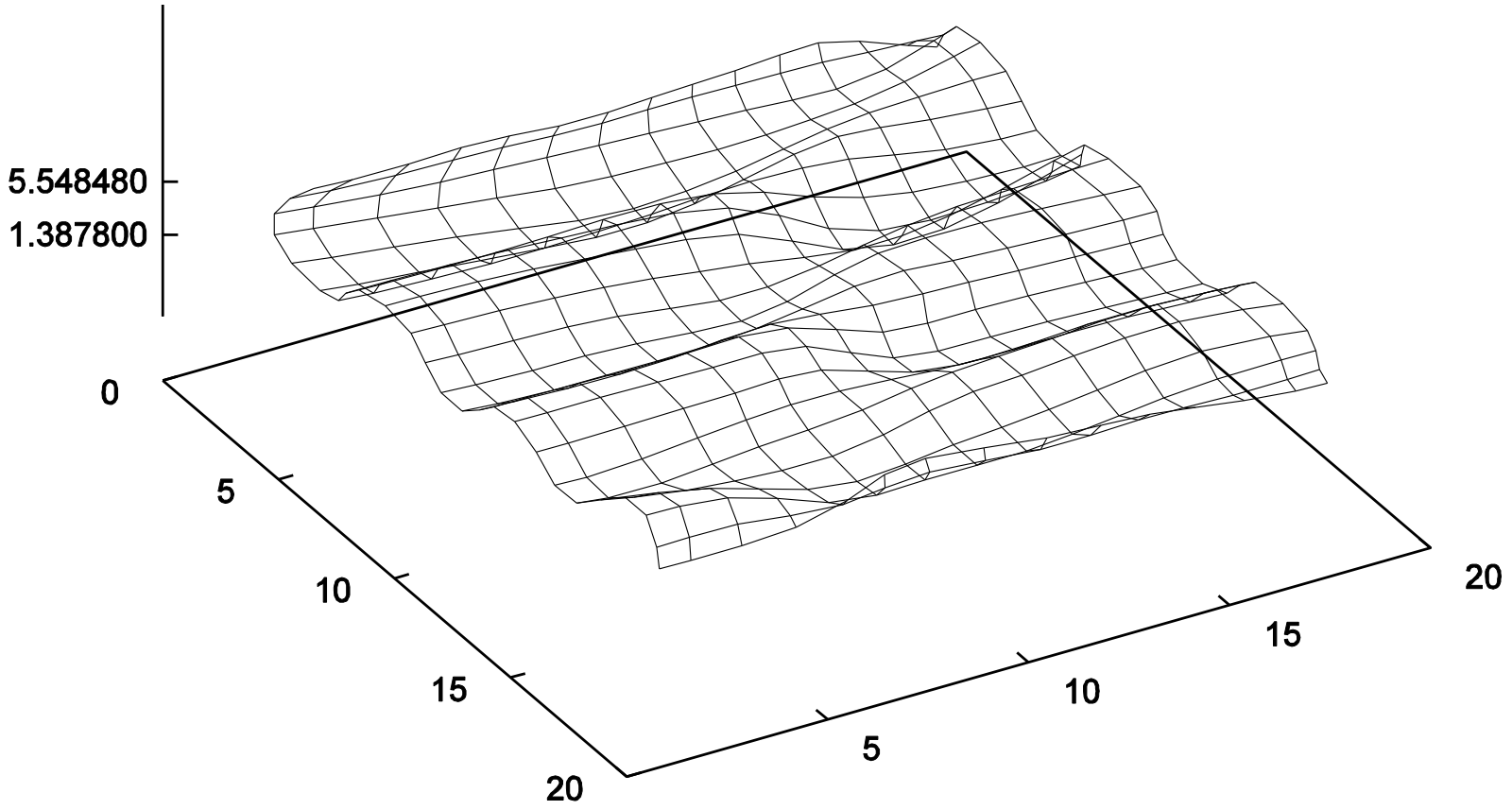
1. Select an input sample
2. Compute the Euclidean distance between input and weight vector for each output node
3. Select the output node with minimum distance
4. Update weights to all nodes within a topological distance of winning node

End-while

Unfolding of a 2-D Map in a 2-D Data Space



Unfolding of a 2-D Map in a 3-D Data Space



The Self-Organizing Map Algorithm

- The weight vectors are randomly initialized
- Input vectors are presented to the network
  - Determine ***best matching neuron*** n_b with the minimal Euclidean distance between input x and the weight vector w

$$n_b = \min_j \left\| x - w_j^T \right\|$$

  - The winning node (and neighbors) have the weight vector moved ***closer to the input*** (with a learning rate $\eta(t)$)

$$w_j^T \leftarrow w_j^T + \eta(t) \cdot \left(x - w_j^T \right)$$

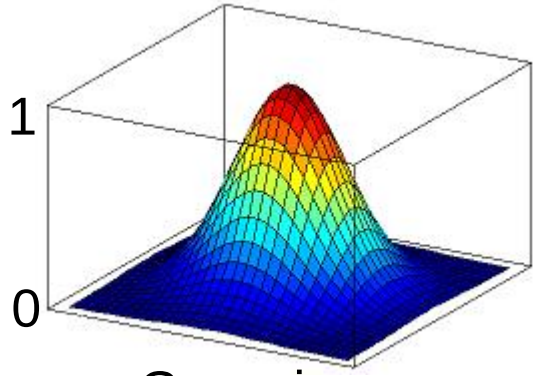
- Over time, the network shall ***self-organize*** so that the input topology is preserved
- (but something is yet missing; this is on-line k-means!)

Neighborhood Function Preserves Topology

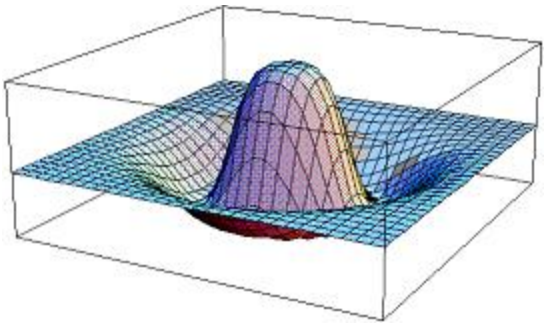
- The neighborhood function $h(n_b,t)$ determines the degree of weight vector change of the neighbors

$$w_j^T \leftarrow w_j^T + \eta(t) \cdot h(n_b,t) \cdot \left(x - w_j^T\right)$$

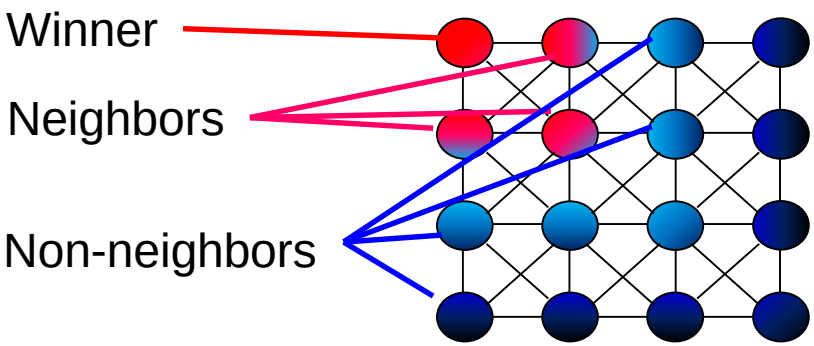
  - Mostly: Gaussian function
    rarely: Mexican Hat function
  - Width decreases during training
    (→ implicit decrease of learning rate)
  - *May* decrease to zero (→ k-means)



Gaussian
(not normalized)

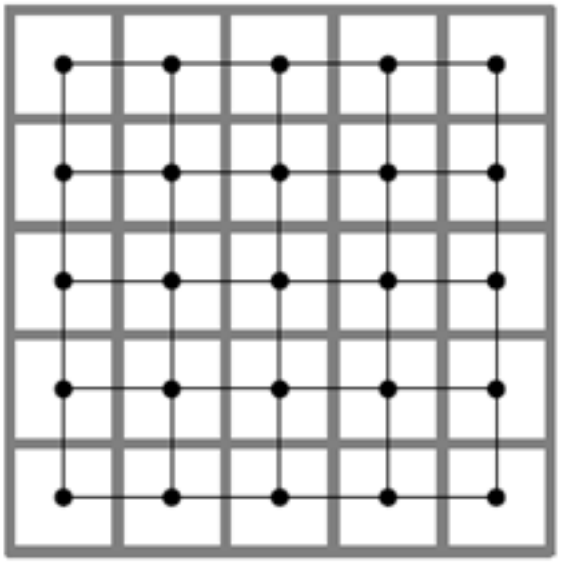


Mexican Hat
(Difference of Gaussian)

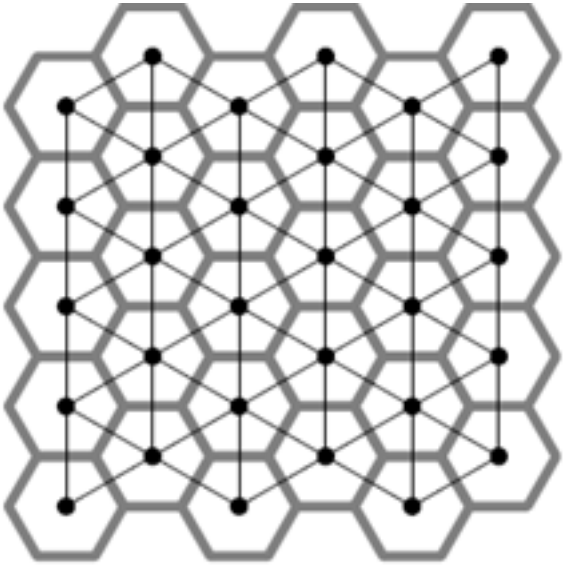


Neighborhood of Output Neurons

Quadratic



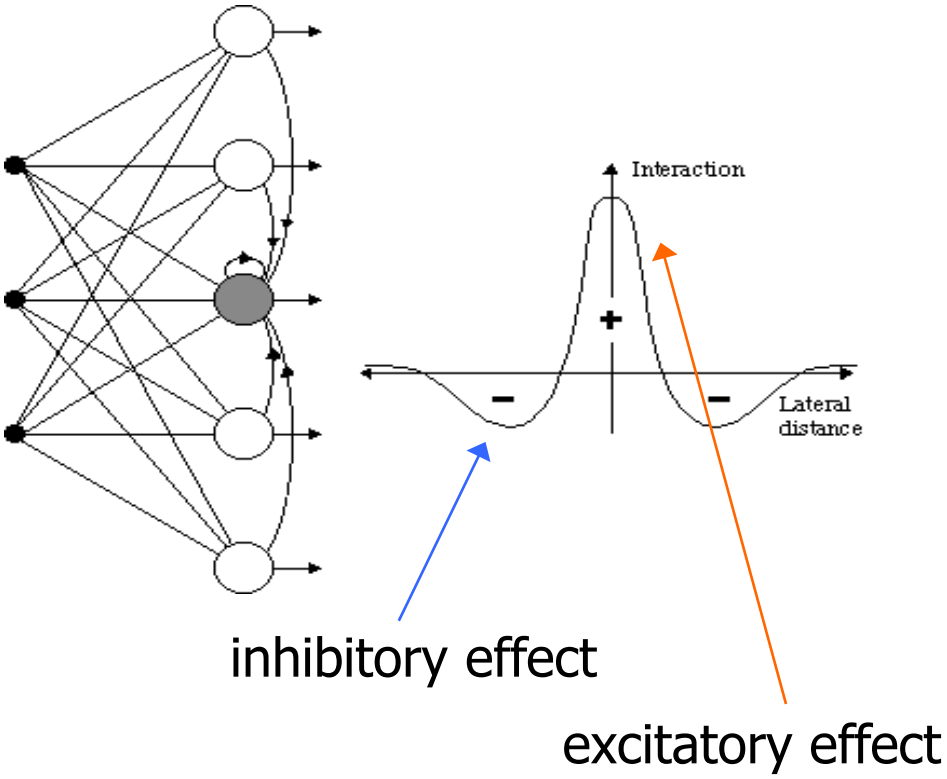
Hexagonal



- Thin lines show next neighbors of a neuron
- Thick gray lines show regions of a neuron

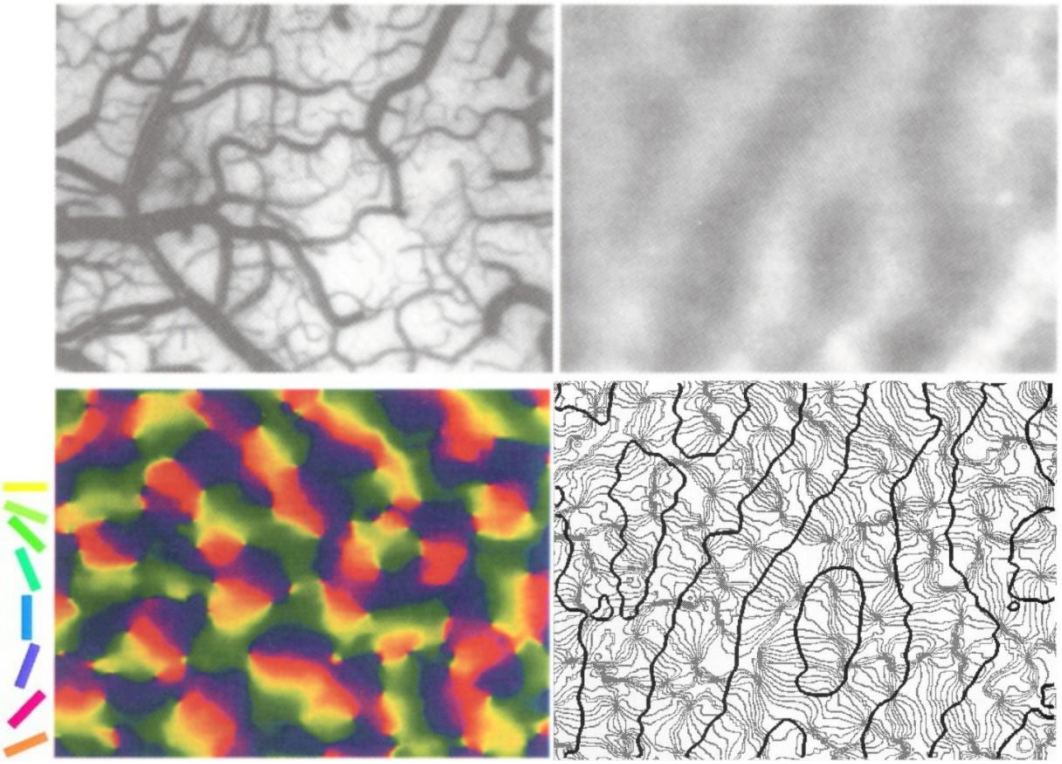
Feature Map and Mexican Hat Function

- The Mexican hat function represents the relationship between the distance from the BMU and the strength of connections within the network layer
- Near neighbourhood – short range lateral excitation area has strong positive effect
- Remote neighbourhood – has a weak negative inhibitory effect



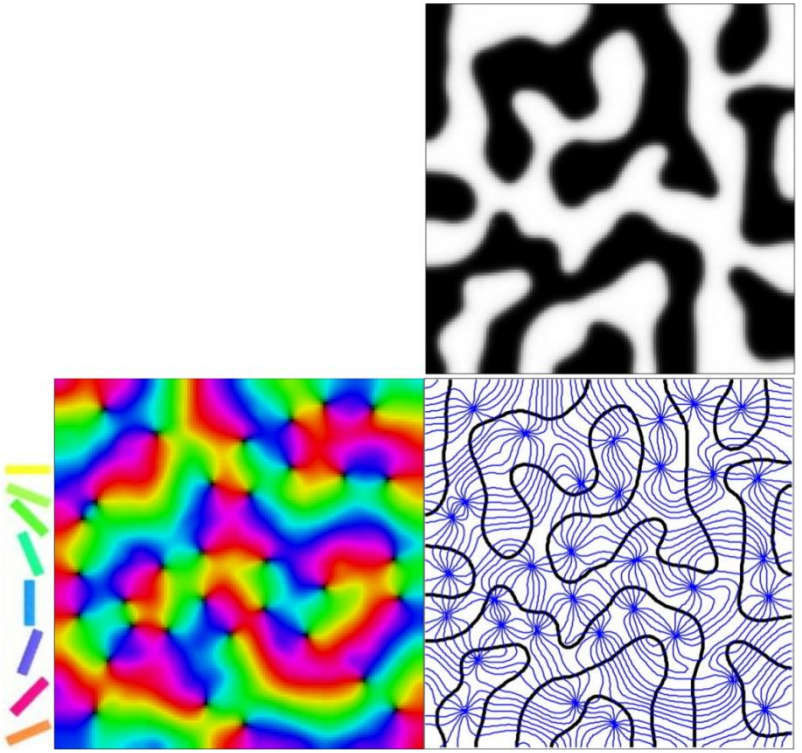
Unfolding of a 2-D Map in a 4-D Data Space

ocular dominance



orientation preference

[Obermayer, Blasdel 1993]



[Goodhill 2007]

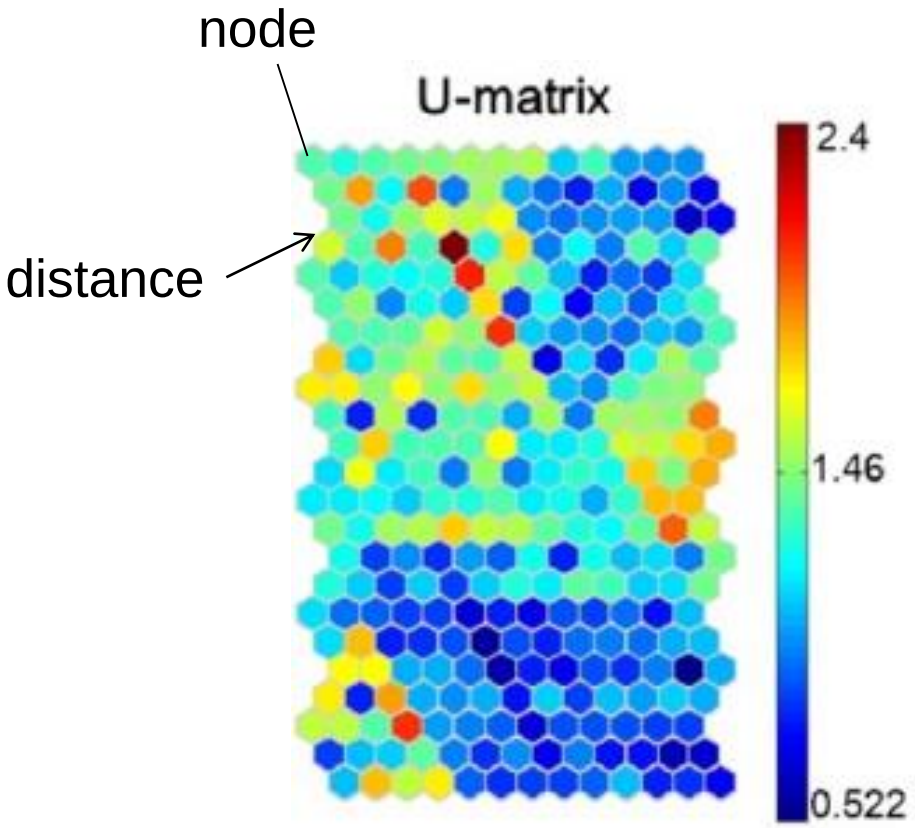
U-matrix of a SOM for identifying cluster boundaries

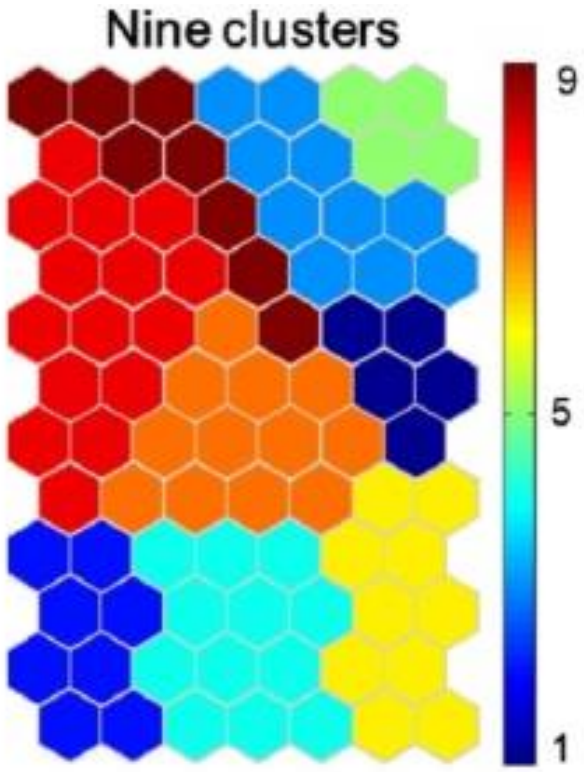
- Interpret the cluster boundaries by taking the average distance d of a weight vector of a unit i to the weight vectors of its neighbors N

$$u(i) = \frac{1}{n}\sum_{j} d(w_i, w_j), n_j \in N(i), \qquad n = |N(i)|$$

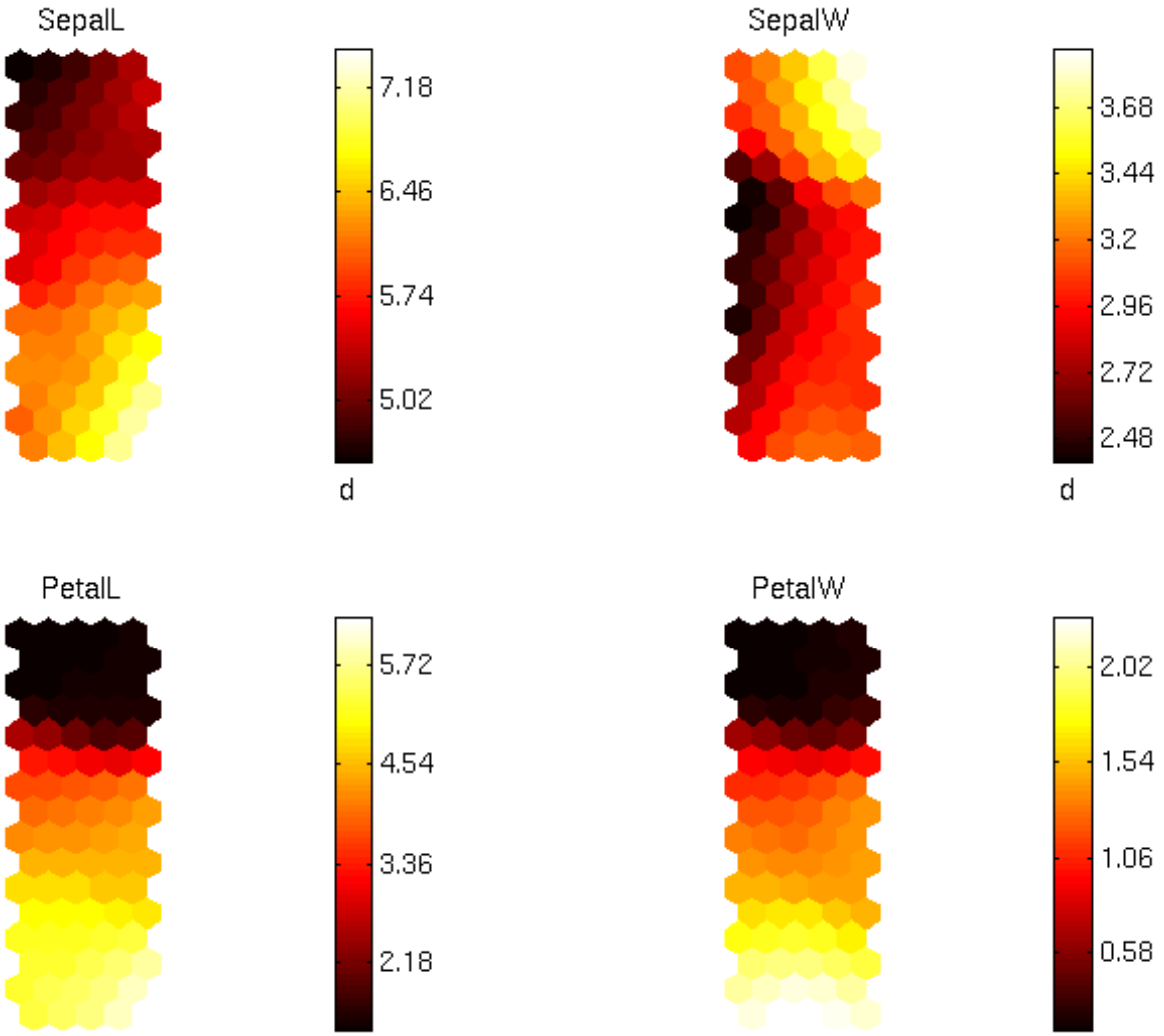
- ***Unified distance matrix*** by Ultsch & Siemon, 1990

Rule extraction from SOMs based on U-Matrix





Rule extraction from SOMs on IRIS data set



Advanced Neural Clustering models

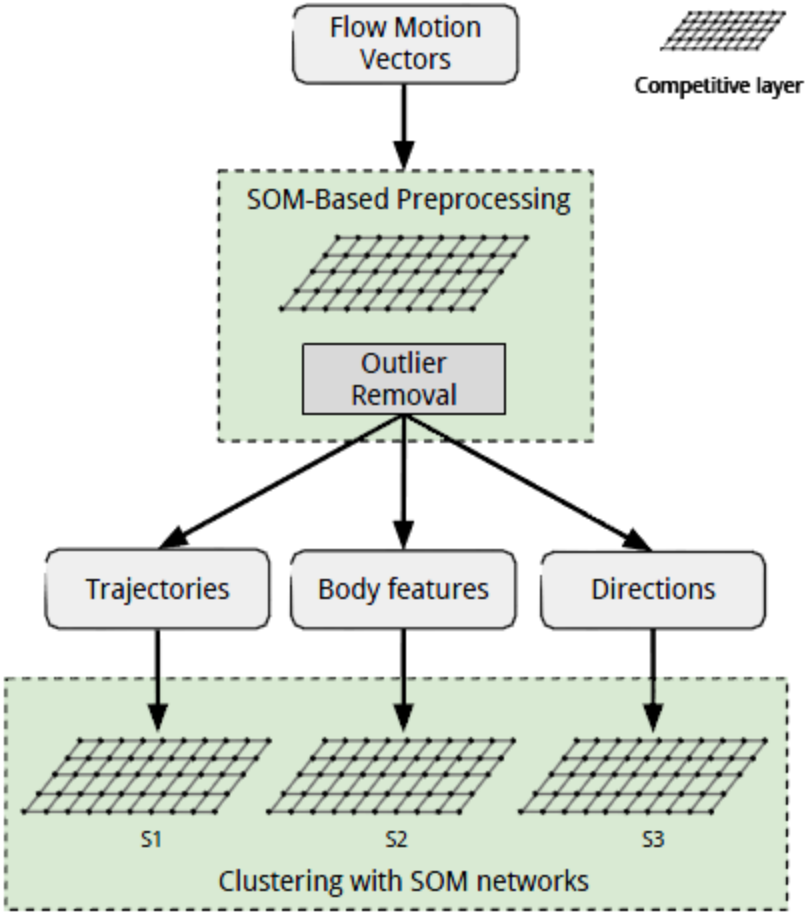
- ***Static Models***: Competitive Learning (CL), Self-Organizing Map (SOM), Neural Gas (NG), …
- ***Dynamic Models***: Growing Grid (GG), Growing Cell Structure (GCS), Growing Neural Gas (GNG), Grow When Required (GWR), Dynamic Adaptive Self-organizing Hybrid model (DASH), etc.
- ***Hierarchical Models***: Multilayered Self-Organising Feature Maps (M-SOM), Growing Hierarchical Self-Organizing Map (GHSOM), etc.

Non-stationary environments are hard to master!

SOM-based Novelty Detection

(Parisi & Wermter 2013)

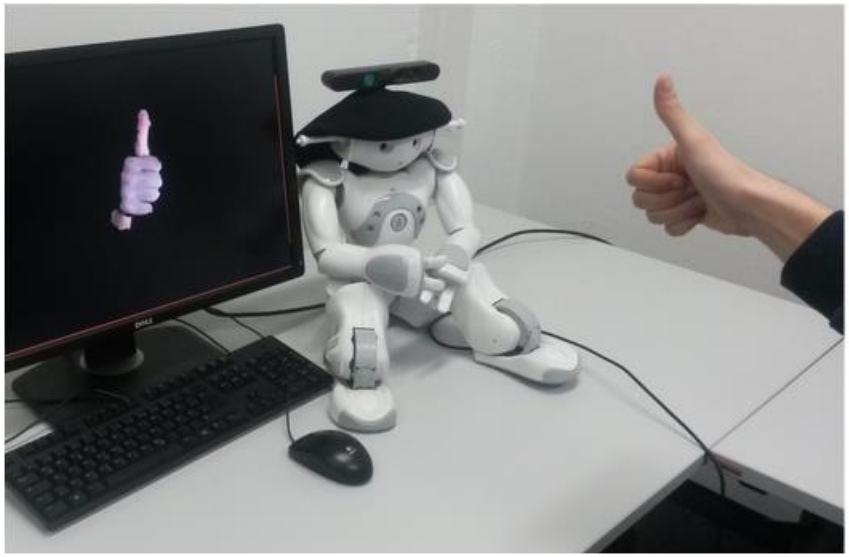
- Detection of novel behavioral patterns
- Networks trained on domestic actions in terms of body motion descriptors
- Neural-statistical architecture for detecting novel behavior

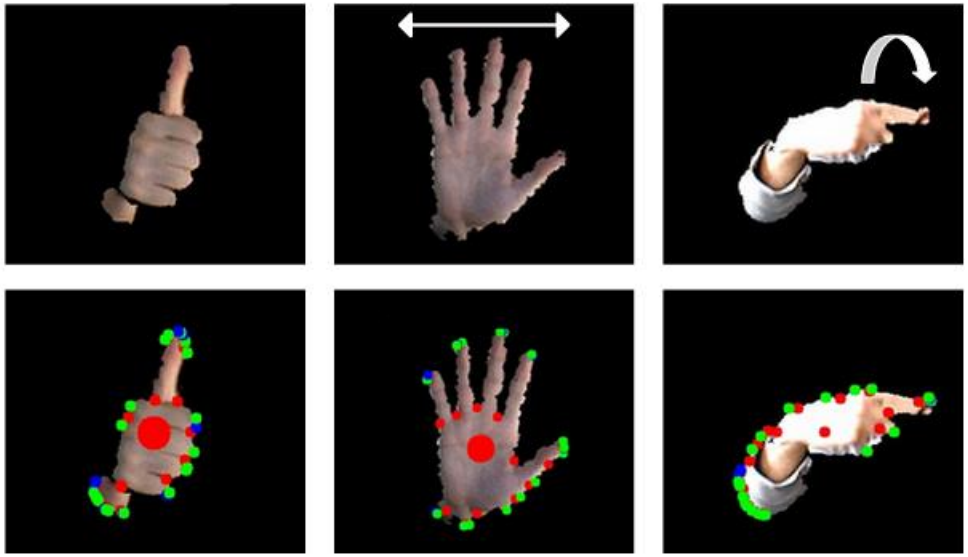


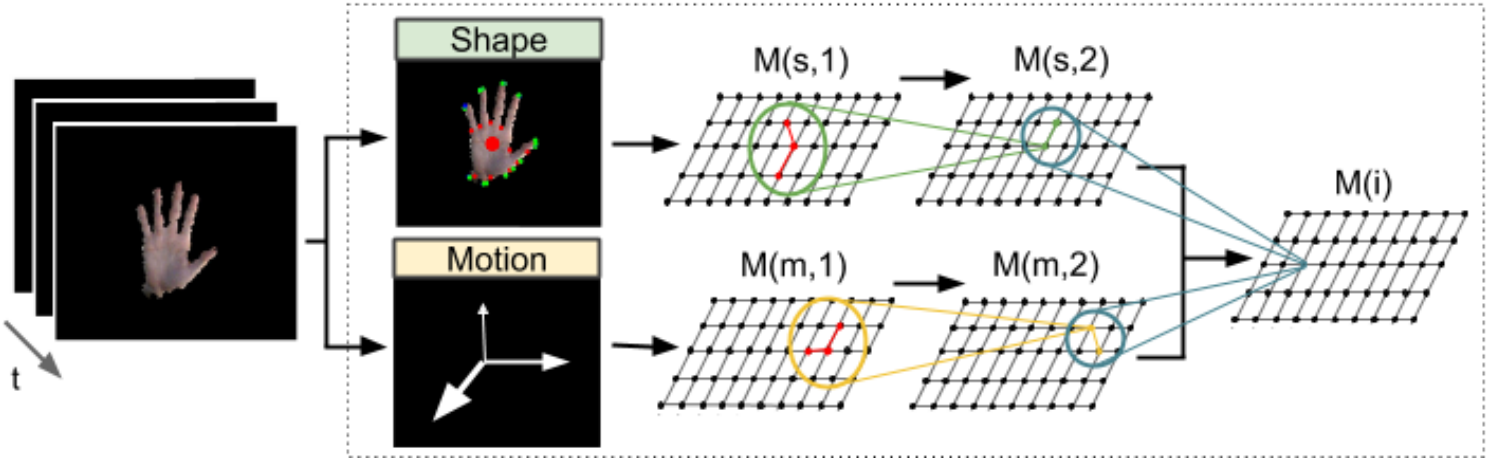
Fall Detection

Hierarchical SOM-based Gesture Recognition

(Parisi, Barros, Wermter 2014)



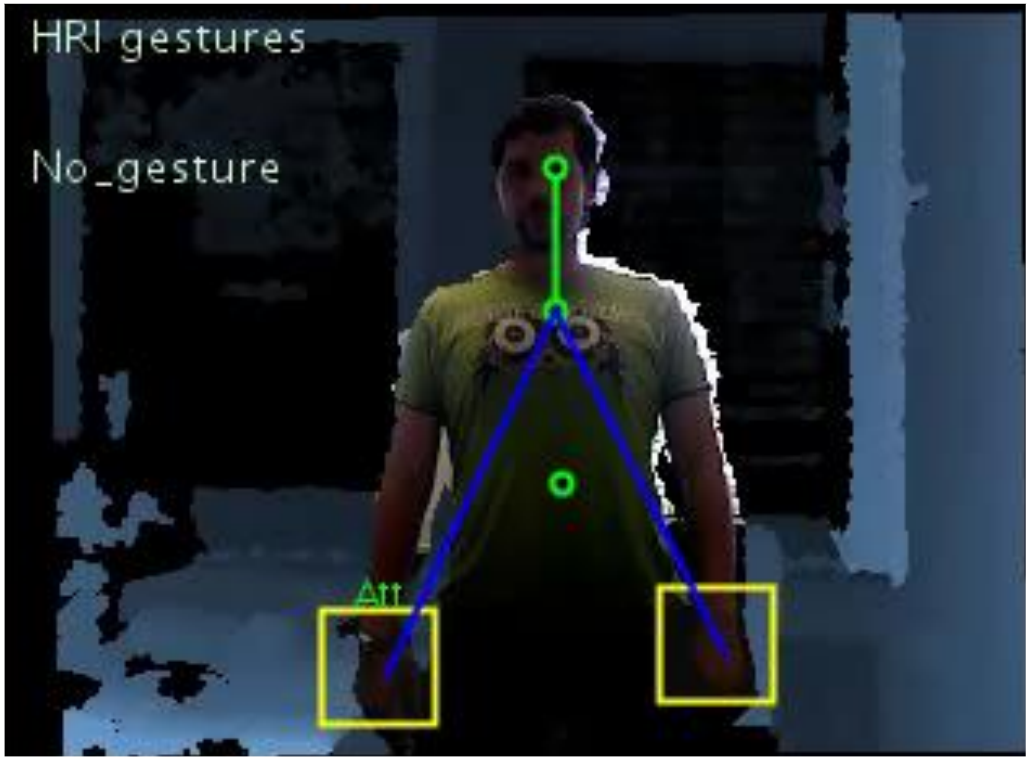




HandSOM

(Parisi, Jirak, Wermter 2014)

- SOM-based motion clustering of salient hand motion features
- Recognition of a set of training hand gestures in real time



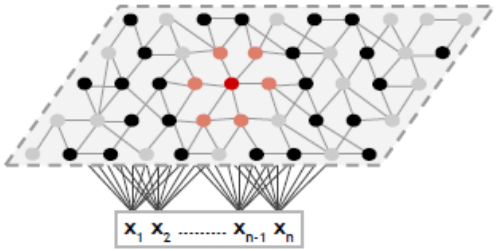
Growing Self-Organization

- Incremental update of the topological structure to the distribution of the inputs
- On-line/incremental learning:
  - Dynamic input distribution
  - Inputs available over time
- Growth and forget factor
  - Creation and removal of neurons
- Growing Neural Gas (Fritzke, 1995)
- Growing When Required (Marsland et al, 2002)

Growing When Required
(Marsland et al., 2002)

- Competitive layer fully connected to the input: $\boldsymbol{x}(t),\ \boldsymbol{w_j} \in \mathbb{R}^n$
- Best-maching neuron:

$$BMU = argmin_{j \in N}\{d_j = \| \boldsymbol{x}(t) - \boldsymbol{w}_j \|\}$$



- Neural update:

$$\Delta \boldsymbol{w}_i = \epsilon_i \cdot h_i \cdot (\boldsymbol{x}(t) - \boldsymbol{w}_i)$$

$$\Delta h_i = \tau_i \cdot 1.05 \cdot (1 - h_i) - \tau_i \text{ (habituation counter)}$$

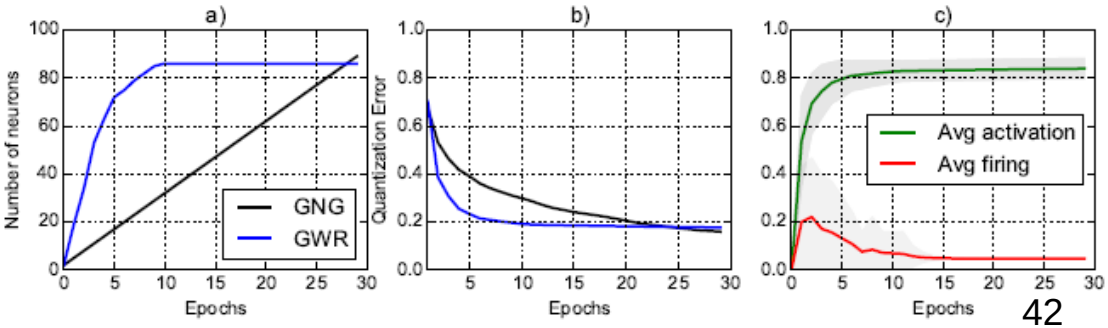
Quantization error:

$$QE = \sum^{N} \frac{1}{N} \| \boldsymbol{x}_n - \boldsymbol{w}_{BMU} \|$$

- Neurogenesis:

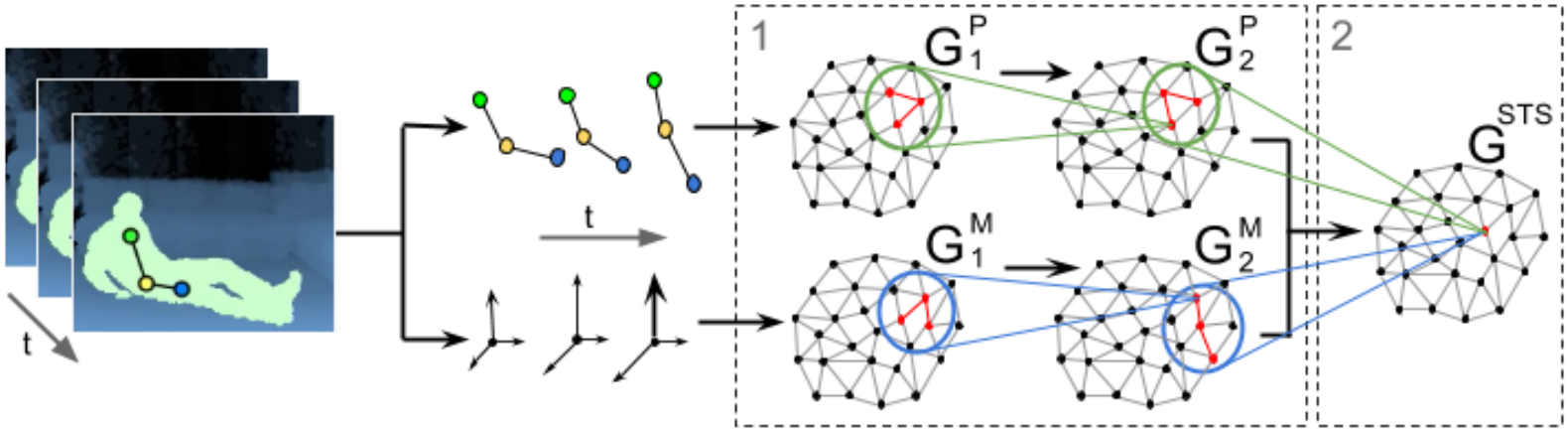
$$\mathrm{a(t)} = \exp(-d_{BMU}) < a_T \quad AND \quad h_b < h_T$$

- Neuron removal
  - Each connection has an age
  - Delete old connections
  - Delete neurons without connections



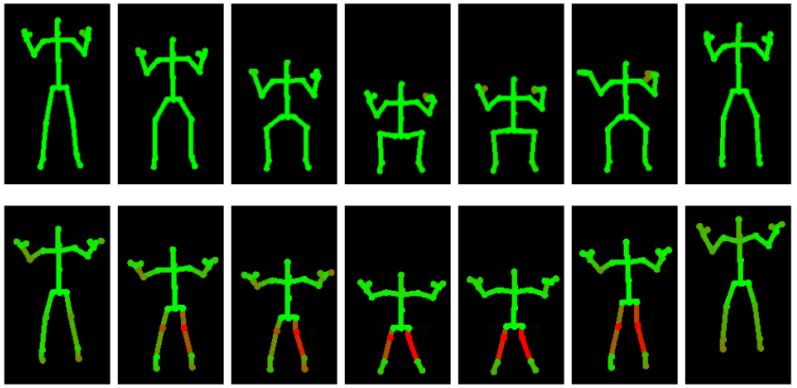
Self-Organizing Neural Integration
(Parisi, Weber, Wermter 2015)

- Self-organizing neural integration of action features
- Hierarchical learning with neuron activation trajectories
- Multi-cue trajectories provide action dynamics the joint feature space
- Growing self-organizing networks
  - Unsupervised learning extended for classification



Human Motion Assessment
(Parisi, Magg, Wermter 2016)

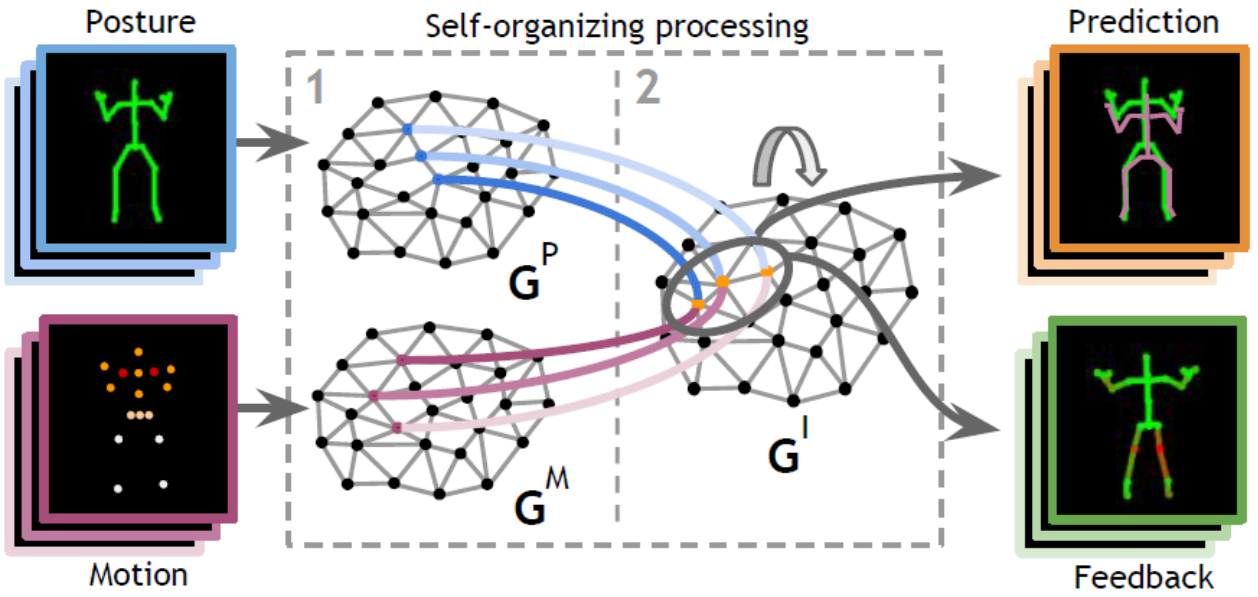
- Extensive research in human action recognition
  - Less attention on the correctness of body motion
- Correct execution of well-defined movements
  - Sports, physical rehabilitation, and gait analysis



- Provide real-time feedback to correct movement mistakes

Body Motion Feedback

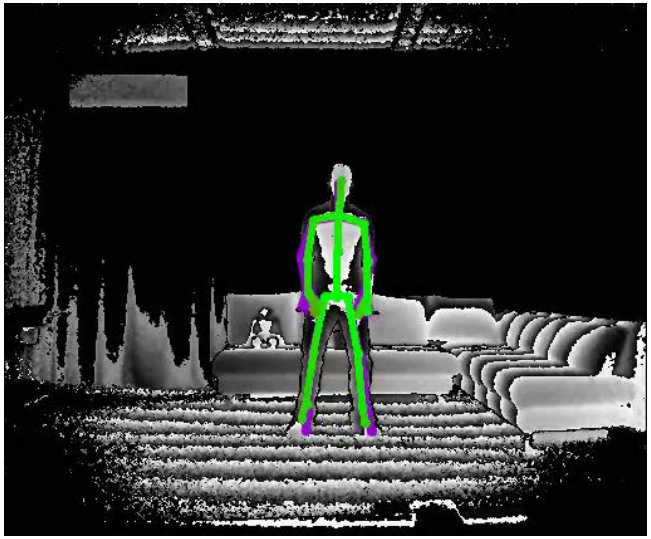
- Visual feedback in real time for body posture and individual joints

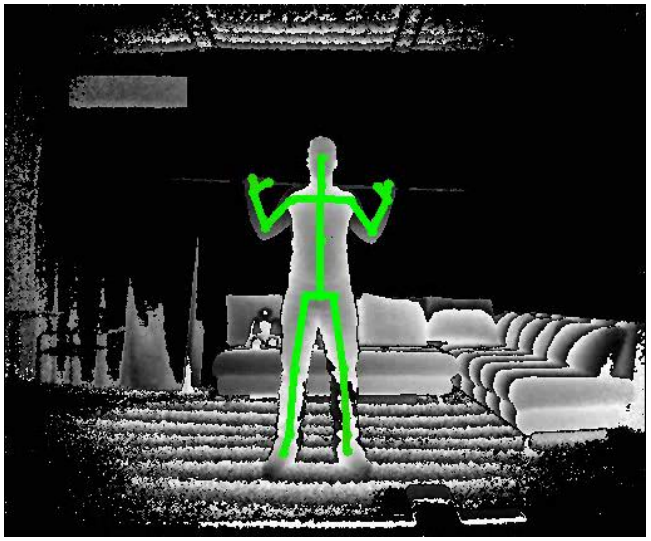


- Recurrent self-organization for prediction-driven feedback
  - GWR context learning (neurons activate for sequences)
  - Smaller TQE than other models of recurrent self-organization
  - Body motion prediction and prediction-driven visual feedback

Prediction-driven Feedback

- Dataset of 3 powerlifting routines
  - 17 participants (9 male, 8 female)
  - High bar back squat, deadlift, dumbbell lateral raise

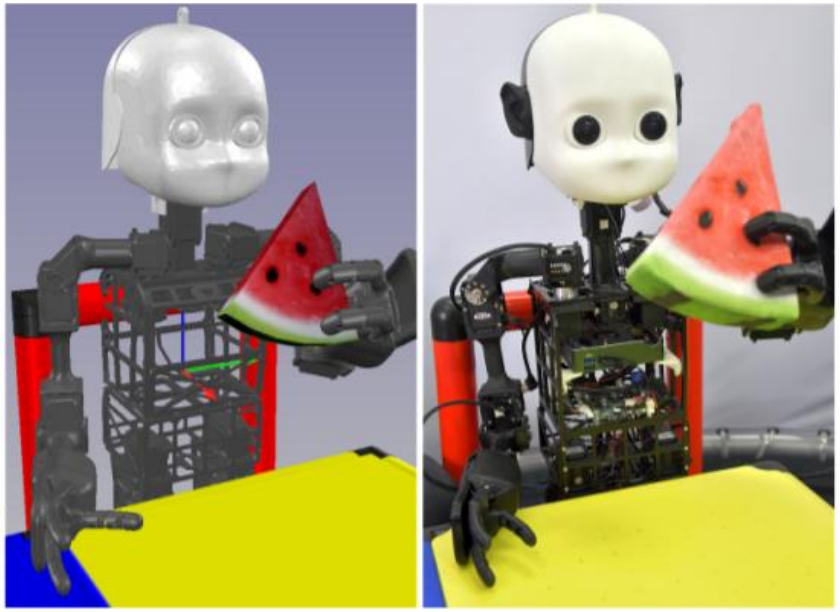




- Single-subject evaluation: MergeGWR = MergeSOM = 1 PPV
- Multiple-subject evaluation: MergeGWR = 0.69, MergeSOM = 0.66

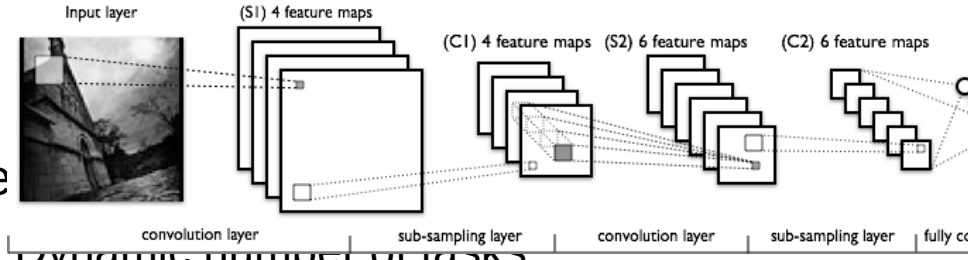
Lifelong Learning

- Artificial agents interacting autonomously in **dynamic** environments
- **Continually** acquire and refine knowledge over time
- Number of **tasks** not known a priori
- Sensory input becomes available over **time**

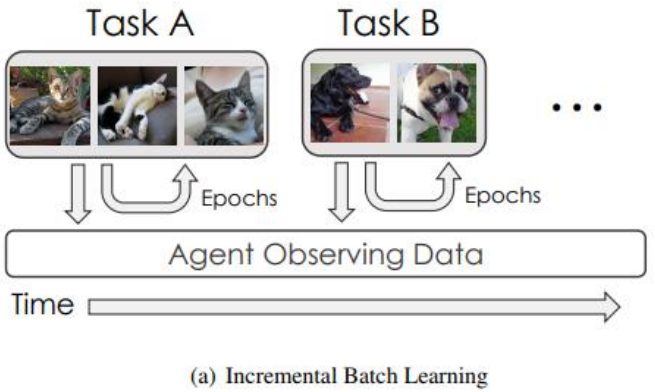


Batch vs Incremental / Lifelong

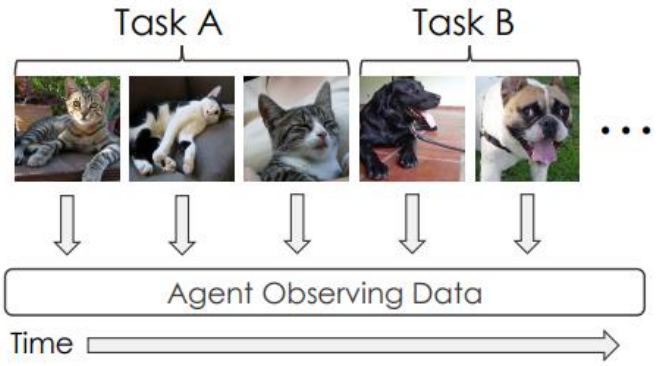
- Batch learning
  - A fixed set of (annotated) training samples
  - Fixed number of tasks
  - Prone to catastrophic forgetting or interference
    (French, 1999; Goodfellow et al., 2013)



- Life
  - Dynamic number of tasks
  - No distinction beween training and test phases
  - Incrementally learn new tasks without overwriting existing knowledge
  - Learn in the absense of sensory input



(a) Incremental Batch Learning

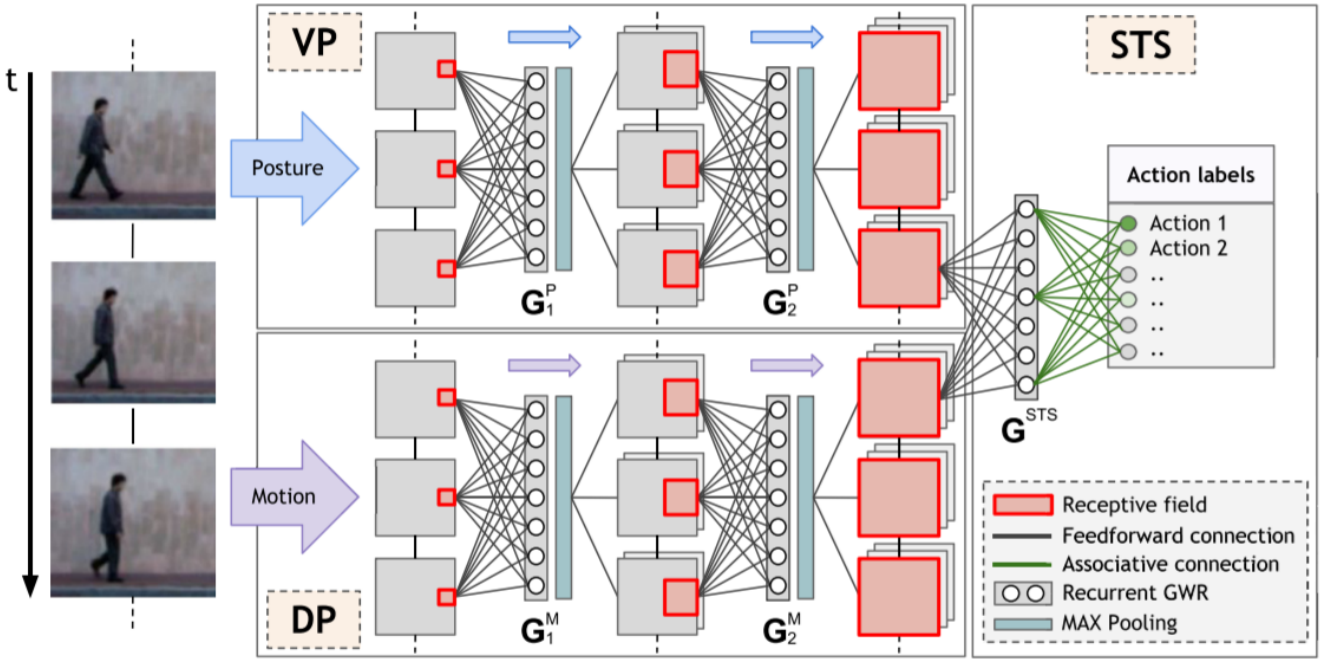


(b) Continual Learning

Deep Neural Network Self-Organization

(Parisi, Tani, Weber, Wermter 2017)

- Unsupervised incremental learning of representations
  - Iteratively **tuning** internal representations embedded in **recurrent** neurons
  - Increasingly large spatiotemporal receptive fields
  - Prediction-driven neurogenesis and neural update rate



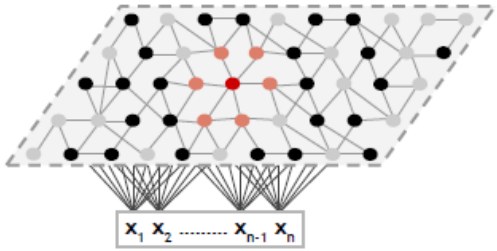
Parisi et al. (2017) Neural Networks

GammaGWR

- Recurrent self-organizing network: $\boldsymbol{x}(t),\ \boldsymbol{w_j},\ \ \boldsymbol{c}_{k,j},\ \ \boldsymbol{C}_k\ \ \in \mathbb{R}^n$
- Best-maching neuron:

$$b = argmin_{j \in N}\left\{ d_j = \ a_w \cdot \| \ \boldsymbol{x}(t) - \boldsymbol{w}_j \ \| + \sum_{k=1}^{K} \alpha_k \cdot \| \ \boldsymbol{C}_k \ (t) - \boldsymbol{c}_{k,j} \ \| \right\}$$

$$\boldsymbol{C}_k(t) = \ \beta \ \cdot \boldsymbol{w}_{I-1} + (1-\beta) \cdot \boldsymbol{c}_{k-1,I-1}$$



- Neural update:

$$\Delta \boldsymbol{w}_i = \epsilon_i \cdot h_i \cdot (\boldsymbol{x}(t) - \boldsymbol{w}_i), \quad \Delta \boldsymbol{c}_{k,i} = \epsilon_i \cdot h_i \cdot (\boldsymbol{C}_k(t) - \boldsymbol{c}_{k,i})$$

$$\Delta h_{\boldsymbol{i}} = \tau_i \cdot 1.05 \cdot (1 - h_i) - \tau_i$$

- Neurogenesis:

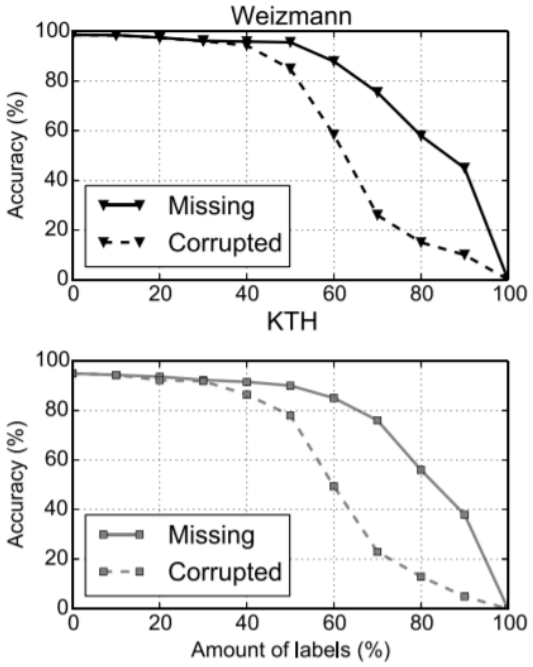
$$\mathrm{a(t)} = \exp(-d_b) < a_T \ \ AND \ \ h_b < h_T$$

GammaGWR Results

- **Weizmann dataset**

10 actions, 9 subjects

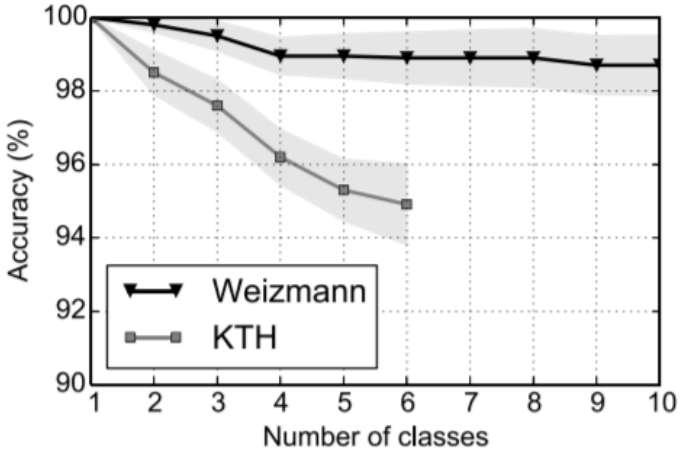
- Best (Gorelick et al., 2005): 99.64 %
- Our approach: 98.6%
- CNN: 92.9%
- MSTNN: 95.3%
- 3D-CNN: 96.2%



- **KTH dataset**

6 actions, 25 subjects

- Best (Gao et al. 2010): 95.04 %
- Our approach: 94.91%
- 3D-CNN: 90.20%
- 3D-CNN + LSTM: 94.39%

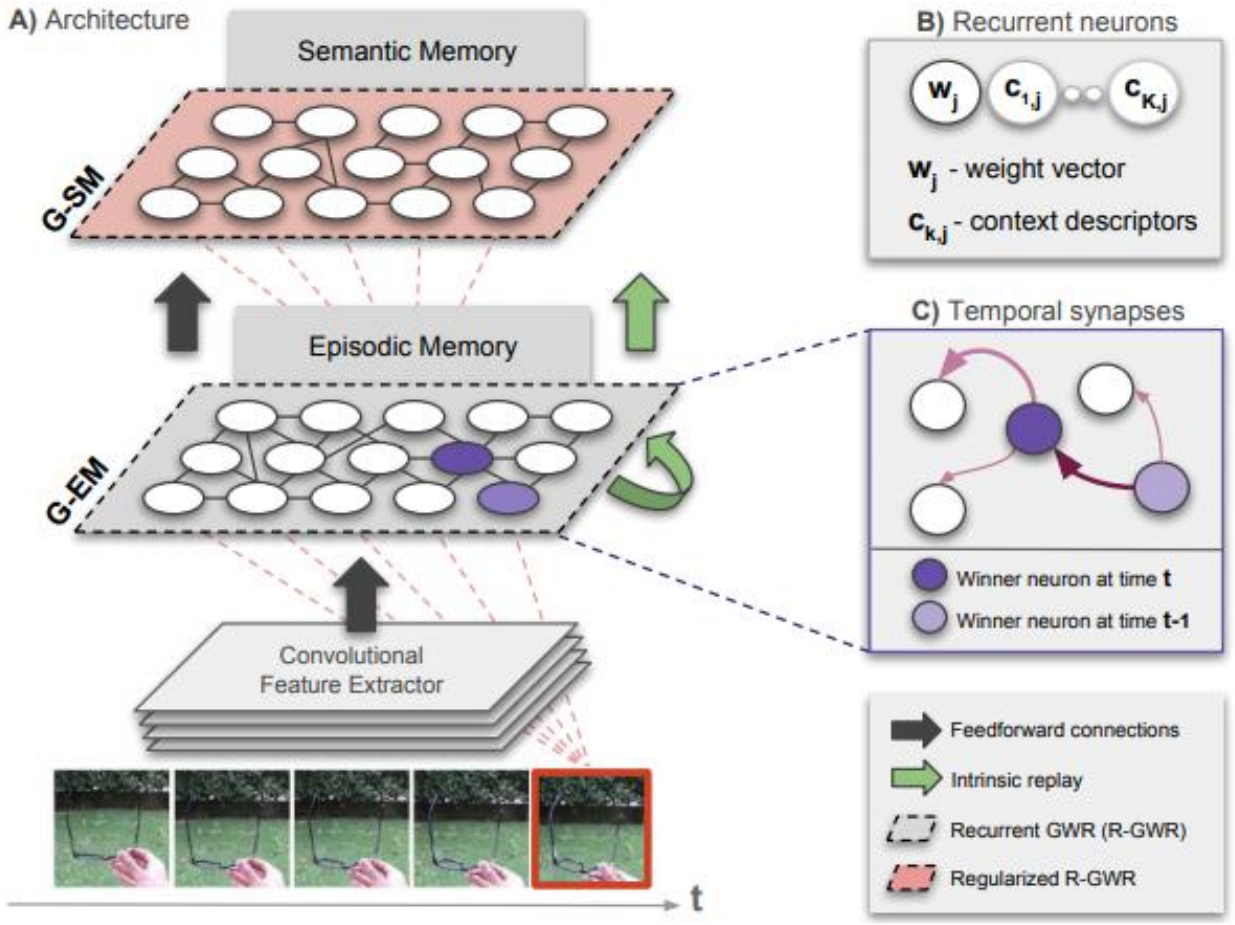


Biological Lifelong Learning

- Humans continually acquire and fine-tune knowledge over their lifespan
- Learning without catastrophically forgetting
- Plasticity-Stability balance
  - Adapt to new knowledge
  - Stabilize neural activity

- Complementary Learning Systems (CLS) theory
  - Interplay of complementary brain areas for memory consolidation
  - Episodic memory (hippocampal system)
  - Semantic memory (neocortical system)

Dual-Memory Self-Organizing Networks

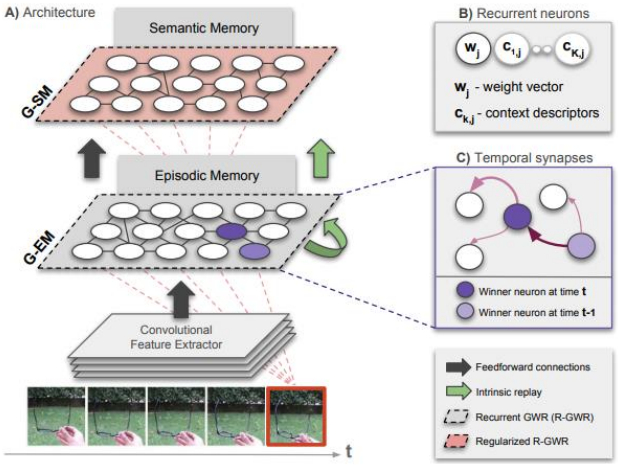


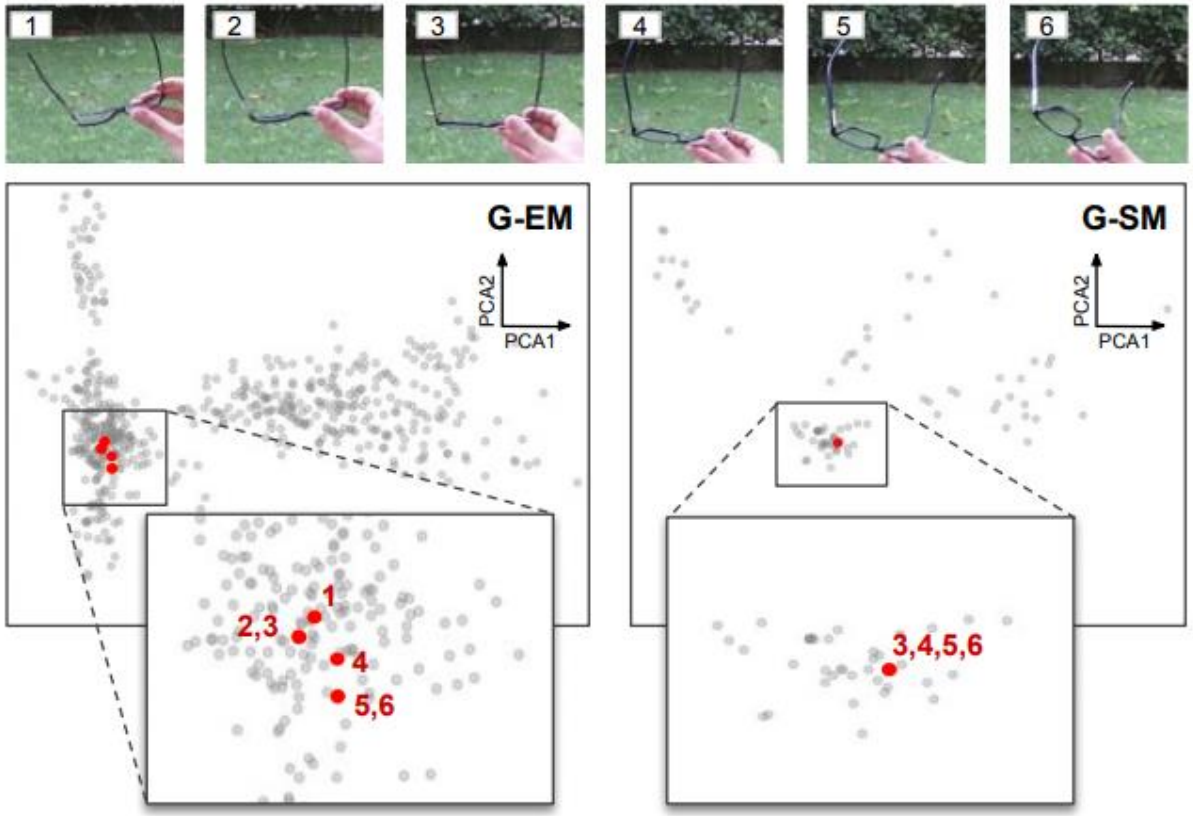
Parisi, Tani, Weber, Wermter (2018) Frontiers in Neurorobotics (under review)

Continuous Object Recognition

- CORe50 (Lomonaco & Maltoni, 2017)
  - Designed for assessing continual learning techniques
  - 50 objects (10 categories, RGB-D @ 20 fps)

Complementary Neural Representations





CORe50 Results

Method	Avg. Acc. (%)	Std. Dev. (%)
New Instances (NI)		
Our approach (w/ intrinsic replay)	**87.93**	1.05
Our approach (baseline)	75,56	2.35
Cumulative (Lomonaco and Maltoni, 2017)	65,15	0,66
LwF* (Zhizhong and Hoiem, 2016)	59,42	2,71
EWC* (Kirkpatrick et al., 2017)	57,40	3,80
Naïve (Lomonaco and Maltoni, 2017)	54,69	6,18
New Classes (NC)		
Our approach (w/ intrinsic replay)	**85.01**	1.93
Our approach (baseline)	72.43	2.87
Cumulative	64,65	1,04
iCaRL* (Rebuffi et al., 2016)	43,62	0,66
CWR (Lomonaco and Maltoni, 2017)	42,32	1,09
LwF*	27,60	1,70
EWC*	26,22	1,18
Naïve	10,75	0,84
New Instances and Classes (NIC)		
Our approach (w/ intrinsic replay)	**86.08**	2.34
Our approach (baseline)	73.32	3.04
Cumulative	64,13	0,88
CWR	29,56	0
LwF*	28,94	4,30
EWC*	28,31	4,30
Naïve	19,39	2,90

Summary

- Unsupervised learning for describing the structure of unlabelled data
- Neural clustering is a useful technique to analyse and plot data based on its latent structure
- Self-organizing neural networks can be used for
  - Dimensionality reduction
  - Clustering
  - Feature extraction
  - Classification of spatio-temporal patterns
  - Unsupervised incremental / lifelong learning