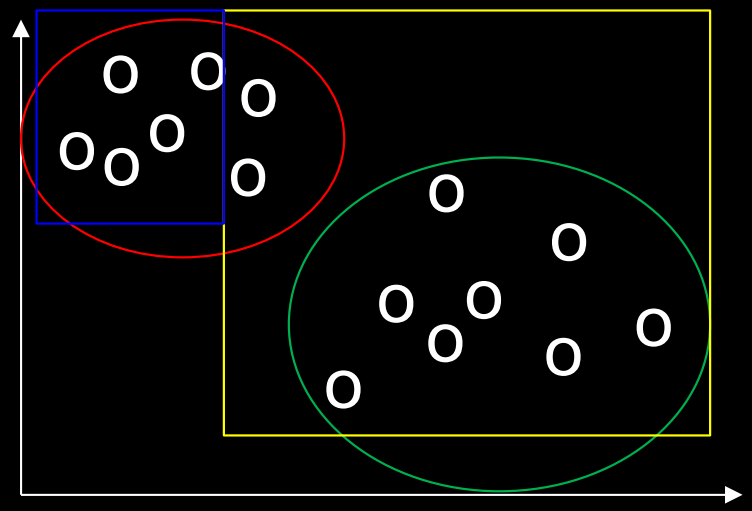
Unsupervised learning

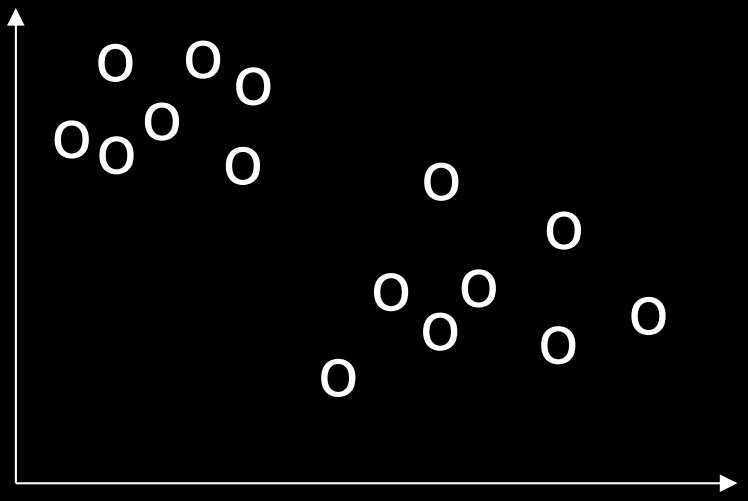
- The question: How do you take input patterns and group them sensibly, without knowing anything about what information the groupings should provide?

- Example – how could these data points be grouped?:



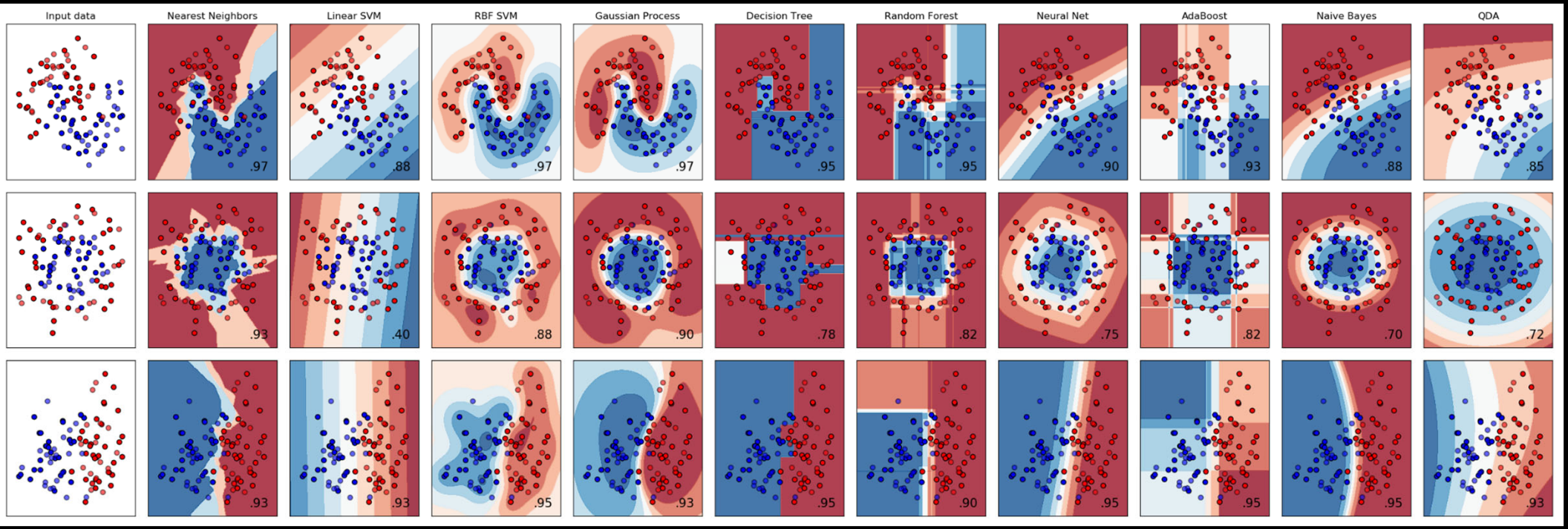
Unsupervised learning

- This is a problem of “Unsupervised learning”, i.e. where you **don’t know which stimuli should be assigned to what “output” category.**
- Without knowing what the outputs should be, you have to find some sensible way to cluster the inputs.
- Various ways to do that:
  - K means
  - Autoencoders
  - Blind source separation, like principal components analysis (PCA, i.e. eigenvectors), or independent component analysis (ICA).
  - Etc…



Unsupervised learning

- Different ways to divide up the space (n-dimensional)
- Example from scikit-learn python package



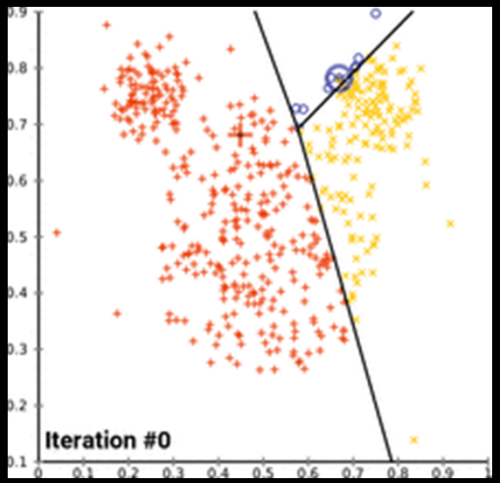
K means

- Simplest algorithm for unsupervised learning – k means
- Start with a set of N training vectors
- Assume there are k clusters, where k is an arbitrary number you decide on. Randomly sprinkle the initial cluster mean positions in the vector space
- Do a 2-step iteration until no further changes:
  - 1) Assign each data point to the nearest (Euclidean distance) of the k cluster centroids

$$\underset{c_i \in C}{\arg\min}\ dist(c_i, x)^2$$

  - 2) Re-compute each cluster centroid as the mean of all vectors nearest the cluster centroid

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$



Kohonen maps

- Basic idea: A set of output cells learn to respond most strongly to certain input patterns.
  - Input patterns that occur more often will have more output cells that respond to them
  - Conversely, input patterns that occur LESS often will have fewer output cells that respond to them.
  - Kohonen maps preserves the topology of the network by having nearest neighbors get co-activated

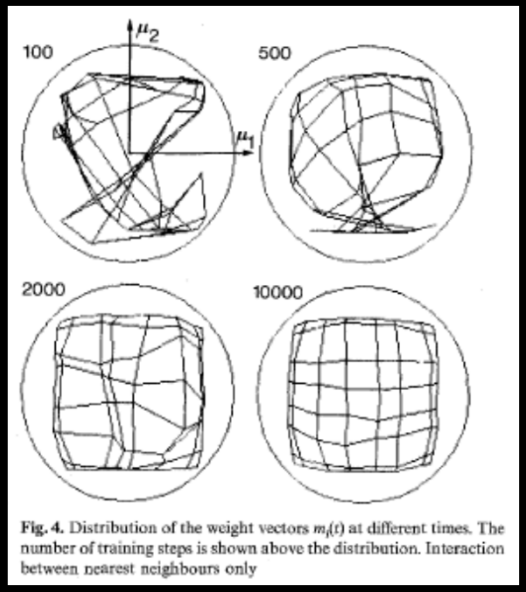
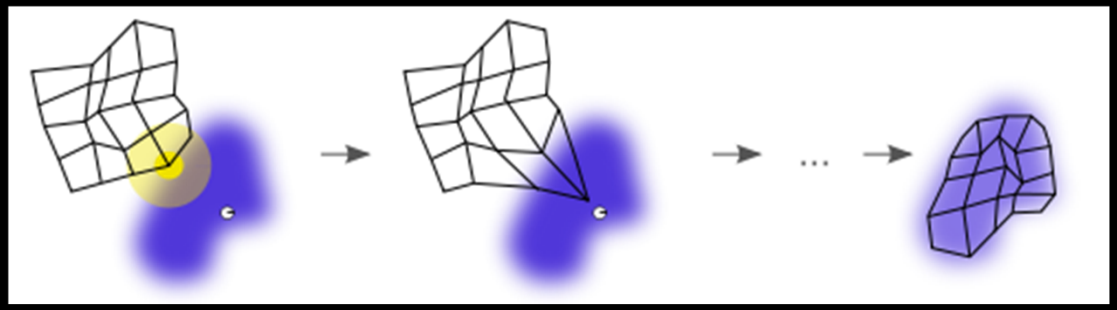


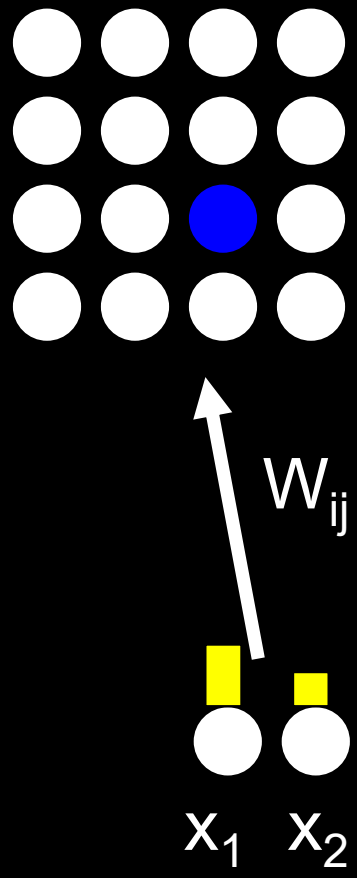
Fig. 4. Distribution of the weight vectors $m_i(t)$ at different times. The number of training steps is shown above the distribution. Interaction between nearest neighbours only



With training, the network learns to represent the input feature space

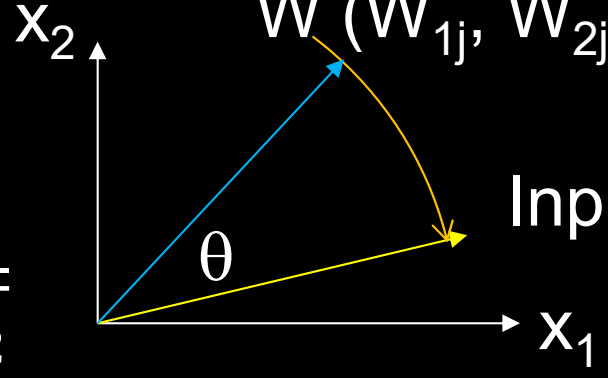
Kohonen maps

- Specifically, consider a network of 4x4 units, with 2 inputs
- We could plot the vector of weights W_{ij} as points in a 2D space, just like the 2D input layer.
- In fact a given map unit (e.g. in blue) will be activated most strongly when the weights to it from the input exactly match the input. This is because when the inputs are the same as the weights, the angle between them is 0, as in the dot product definition.



$$V_j = \frac{W_j \cdot x}{|W_j||x|} = \cos(\theta)$$

$$V_j = \frac{\sum_{i=1}^{2} W_{ji} x_i}{\sqrt{\sum_{i=1}^{2} W_{ji}{}^2 \sum_{i=1}^{2} x_i^2}}$$



Kohonen maps

- Topology: the map units that are next to each other tend to have similar weights (small angle between the weights)
- This happens because activation spreads to neighbors in the network, while more distant units are inhibited.

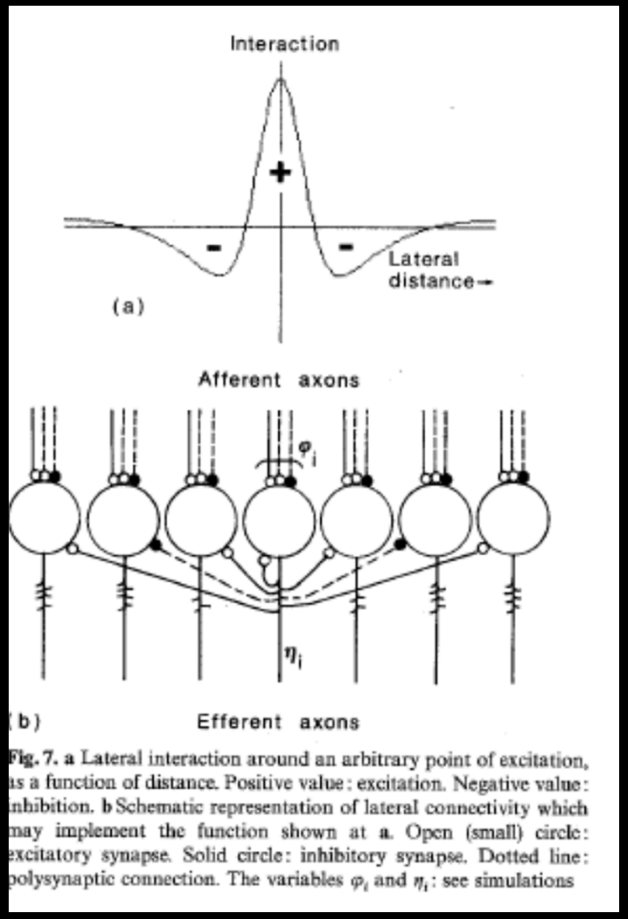
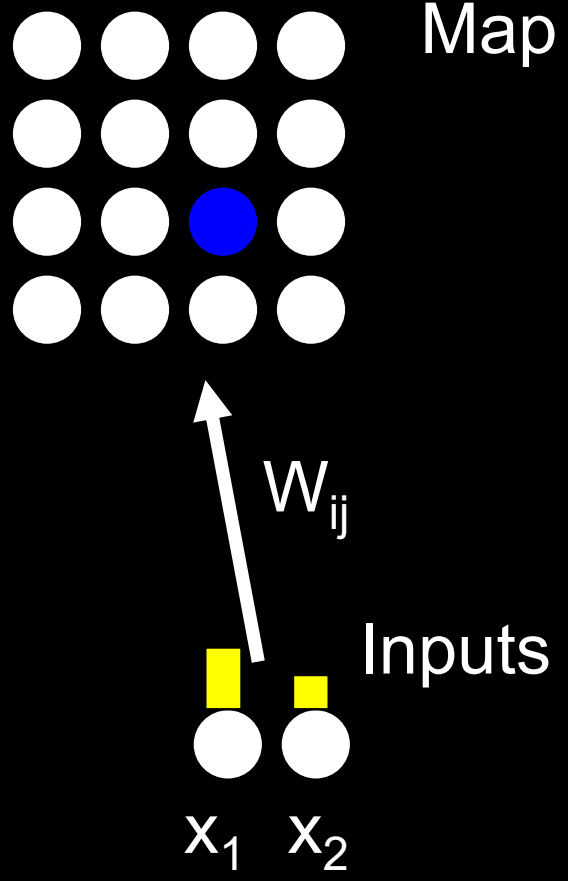


Fig. 7. a Lateral interaction around an arbitrary point of excitation, as a function of distance. Positive value: excitation. Negative value: inhibition. b Schematic representation of lateral connectivity which may implement the function shown at a. Open (small) circle: excitatory synapse. Solid circle: inhibitory synapse. Dotted line: polysynaptic connection. The variables φ_i and η_i: see simulations

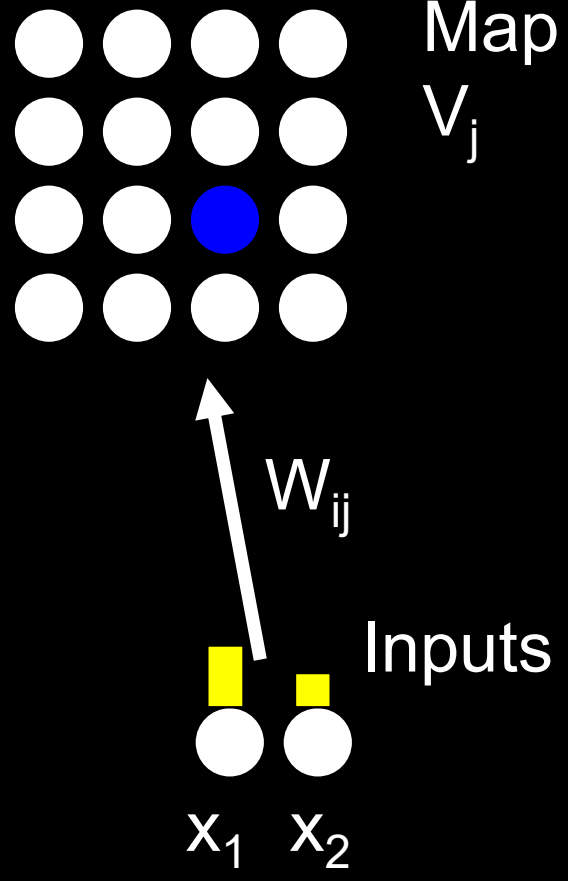
$$d\mu/dt = \alpha(\xi - \xi_b)\eta, \qquad (7)$$

where μ is the efficacy which corresponds to the input weight expressed in Eqs. (1) through (3), ξ is the presynaptic input (triggering frequency of the presynaptic neuron), ξ_b is an effective background value, η is the postsynaptic triggering frequency, and α is a proportionality constant (which depends on the type and location of the synapse). *Notice that all plastic synapses in the present model can be excitatory.*



Kohonen maps

- Algorithm:
  - For each input vector, compute map activation V_{ij}
  - Add lateral excitation and inhibition to the map, to adjust the map values V_j
  - Train the network, as:



$$W_{ij}(new) = W_{ij}(old) + learning\_rate * V_i * (x_j - W_{ij})$$

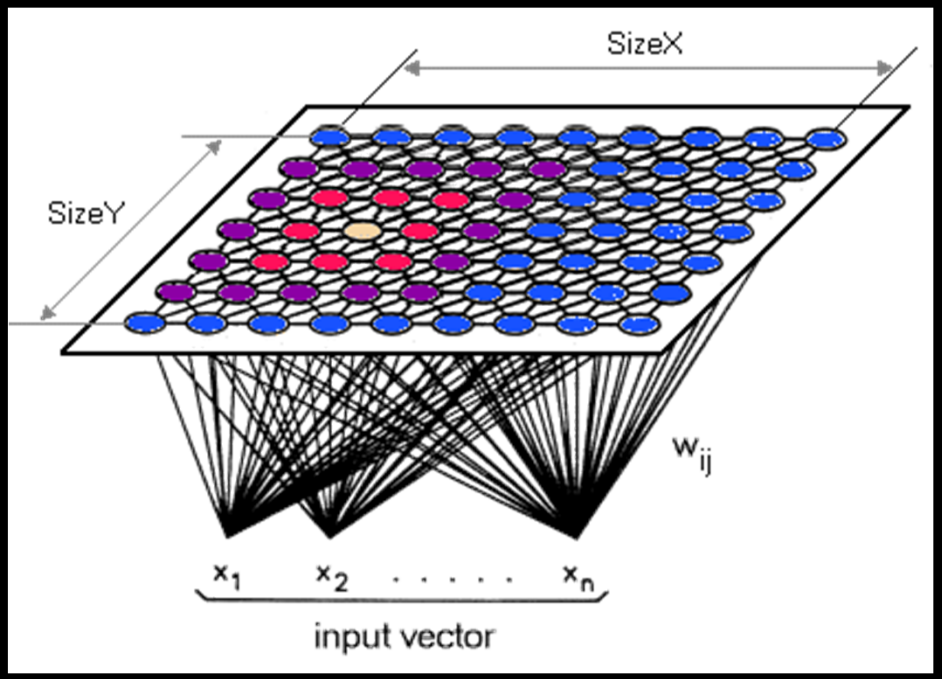
Normalize (optional):

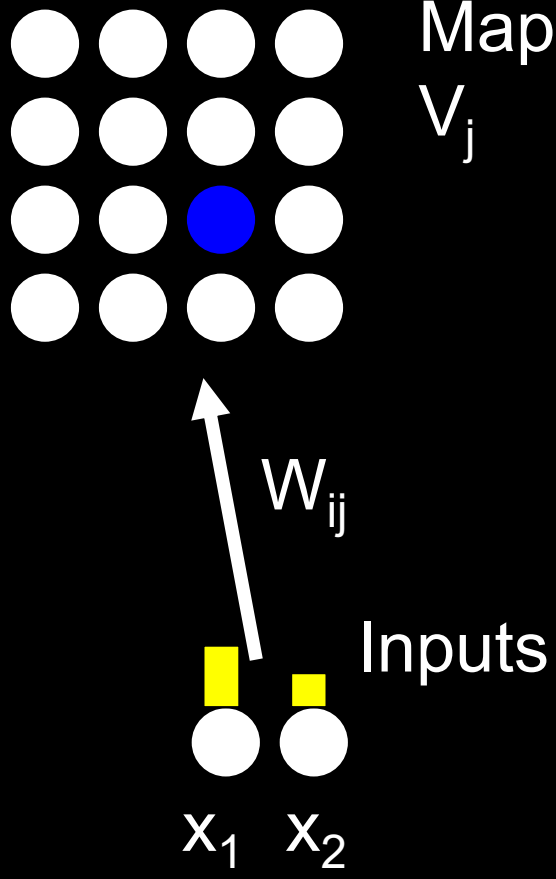
$$W_{ij} = \frac{W_{ij}}{\sqrt{\sum_{i=1}^{2}(W_{ij})^2}}$$

- This means "move the weight vector W toward the input activity vector x",

Kohonen maps

- Python assignment
- Examples
- https://www.youtube.com/watch?v=QvI6L-KqsT4





Kohonen maps

- What can you do with Kohonen maps?
- Hacker attack detection: https://www.youtube.com/watch?v=iXAuKTT5rPw

- Solving the traveling salesman problem:
- Create a 1-dimensional (ring) Kohonen map, with 2D input vectors representing each city.
- https://www.youtube.com/watch?v=wT9rEi9dY8M

Kohonen maps

- Your assignment – what it should look like