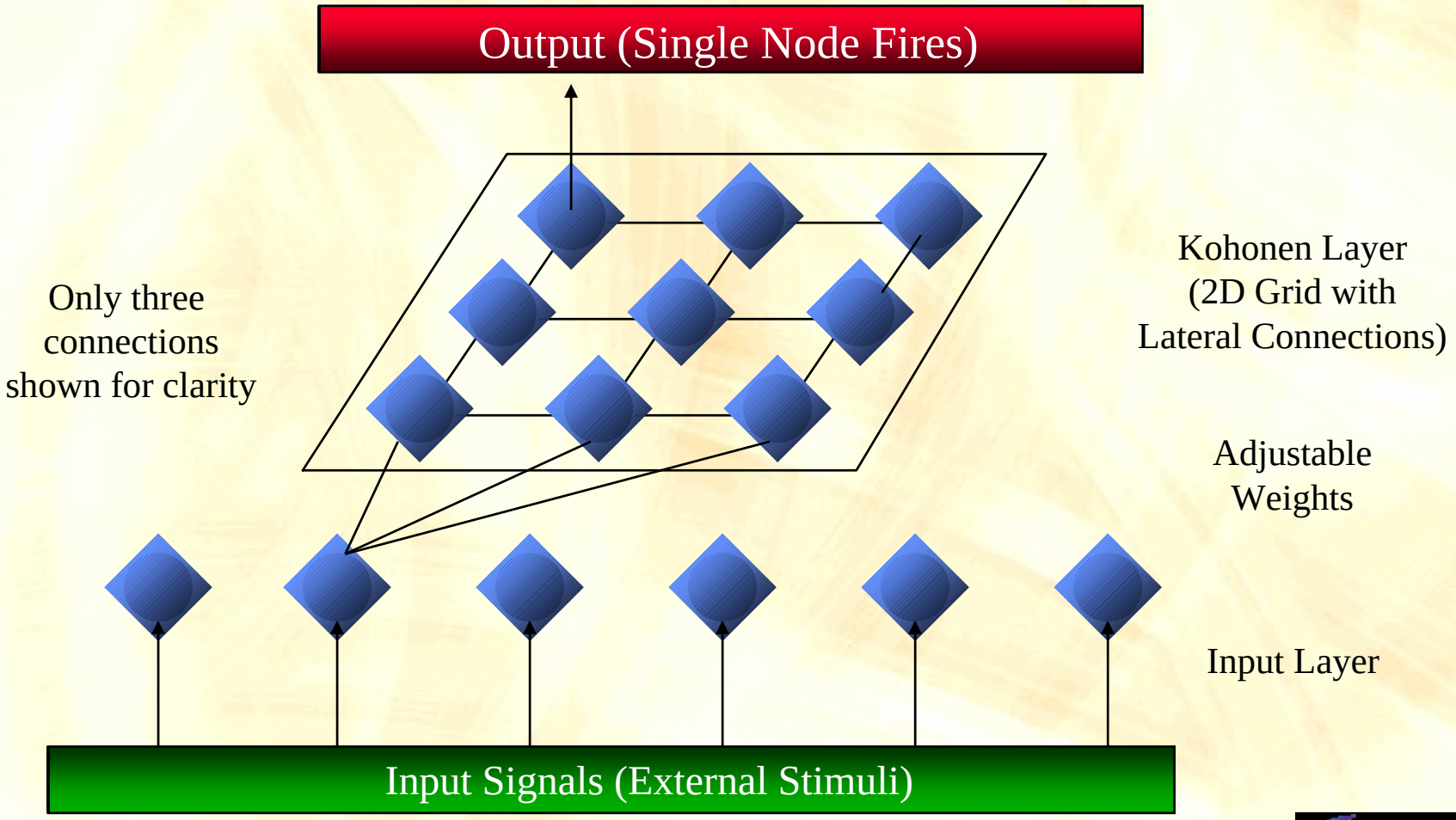
Kohonen SOMs

Dr. Kaustubh Chokshi

Kohonen Self-Organising Maps

- Unsupervised Networks
- Closely related to clustering
- Do not require target outputs for each input vector in the training data
- Inputs are connected to a two-dimensional grid of neurons
  - Neighbourhood relations can be explicitly maintained, or
  - each neuron can have lateral connections to its ‘neighbours’
- Multi-dimensional data can be mapped onto a two-dimensional surface
  - Facilitates representation of clusters in the data

General Architecture of a Kohonen Self-Organising Map



Trained Kohonen SOM

New Input Vector **E**

$\boldsymbol{E}=[e_1,e_2,....e_n]$

n elements

Vector passed through input neurons

Weight Vector, **U**, between the input and each Kohonen layer neuron

$\boldsymbol{U}_i=[u_{i1},u_{i2},...u_{in}]$

i is the neuron in the Kohonen layer

Each Kohonen layer neuron produces a value

Euclidean distance, E_d, of neuron in the data space from the original vector

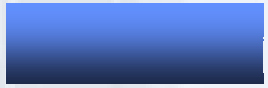
$E_d=\| E - U_i \|$

$E_d=\sqrt{\sum_j (e_j - u_{ij})^2}$

u_{ij} is the weight between input j and Kohonen neuron i

Training Of a Kohonen SOM

Begins with a random initialisation of the weights between the input and Kohonen layers



Each training vector is presented to the network

The winning neuron is found

Plus the winning neuron’s neighbours are identified

Training Of a Kohonen SOM

Weights for the winning neuron and its neighbours
are updated, so that they move closer to the input vector

The change in weights is calculated as follows:

$$\Delta u_{ij} = \alpha(e_j - u_{ij})$$

 - is a learning rate parameter

Training Of a Kohonen SOM

Only the weights on connections to the winning neuron and its neighbours are updated

The weights are updated as follows:

$$u_{ij}^{new} = u_{ij}^{old} + \Delta u_{ij}$$

Both the learning rate and the neighbourhood size decay during training

Training Of a Kohonen SOM

The learning rate is usually set to a relatively high value, such as 0.5, and is decreased as follows:

$$\alpha_t = \alpha_0 (1 - (t/T))$$

T - total number of training iterations

t - is the current training iteration

α_t - is the learning rate for the current training iteration

α_0 - is the initial value of the learning rate

Training Of a Kohonen SOM

The neighbourhood size is also decreased iteratively

Initialise to take in approx. half the layer

Neighbourhood size is reduced linearly at each epoch

Trained Kohonen SOM

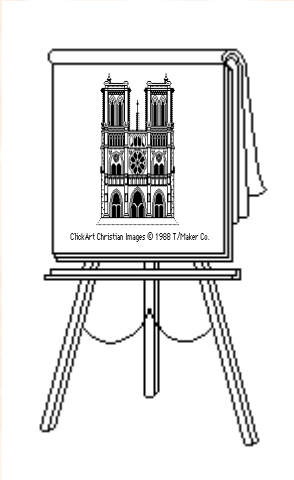
The Kohonen layer unit with the lowest
Euclidean distance,
i.e. the unit closest to the original input vector,
is chosen,as follows:

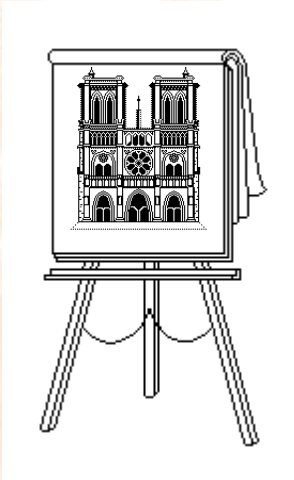
$$\| E - U_c \| = \min_i \{ \| E - U_i \| \}$$

c denotes the '**winning**' neuron in the Kohonen layer

The '**winning**' neuron is considered as the output of the network -"**winner takes all**"

Kohonen Architecture Variations





An interpolation algorithm is used so that the neuron with the lowest distance fires with a high value, and a pre-determined number of other neurons which are the next closest to the data fire with lower values

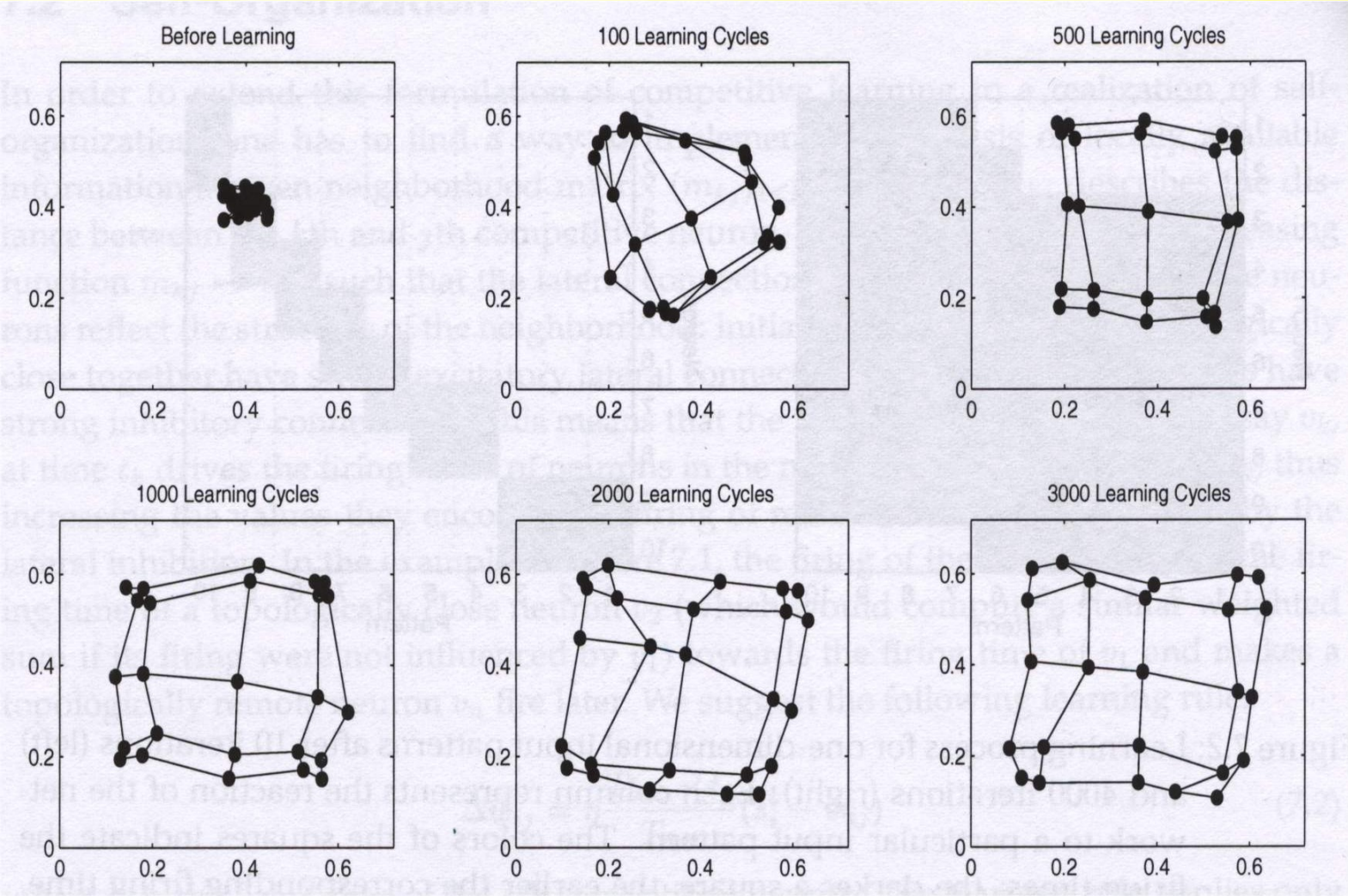
The Strength Of The Kohonen SOM

- Unsupervised architecture
- Requires no target output vectors
- Simply organises itself into the best representation for the data used in training

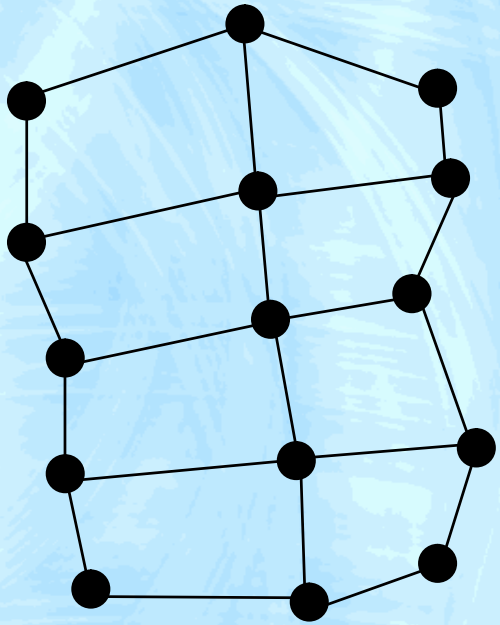
Limitations of Kohonen network

- Provides no information other than identifying where in the data space a particular vector lies
- Therefore interpretation of this information must be made
- Interpretation process can be time-consuming and requires data for which the classification is known

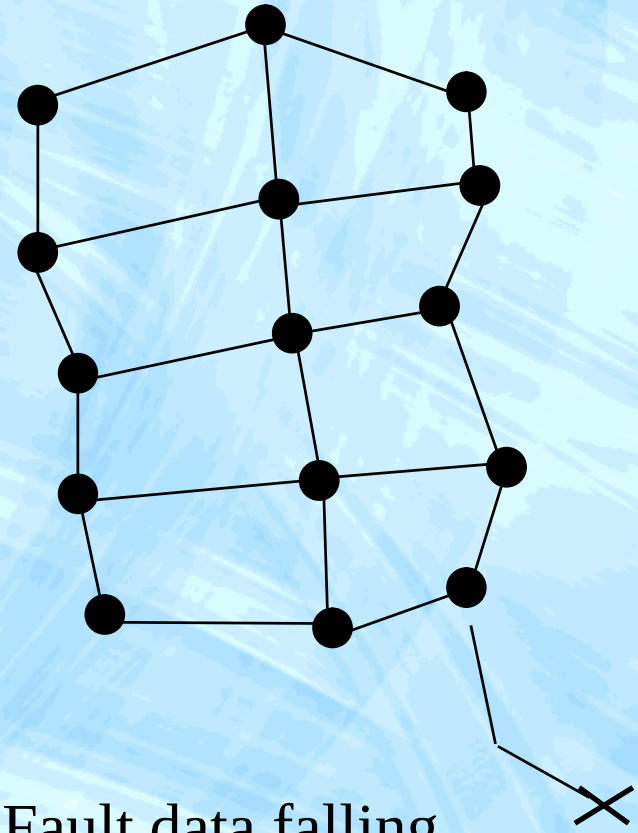
SOM for 2D Square Region



Kohonen SOM for Novelty Detection



Kohonen network representing ‘Normal’ space



Fault data falling outside ‘Normal’ space

Labelling Kohonen Networks

- Class labels can be applied if data is labelled
- Use nearest-neighbour or voting strategies
  - Nearest Neighbour - Set class label to most common label of K nearest training cases
  - Voting - Identify all cases that are “assigned to” that neuron, and assign most common class

Learned Vector Quantisation

- If labelled data is available, it can be used to improve the distribution of neurons
- Move neurons towards correctly-classified cases
- Move away from incorrectly-classified

Conclusion

- Unsupervised learning requires no class labelling of data
  - Discover clusters (and then possibly label)
  - Visualisation
  - Novelty detection