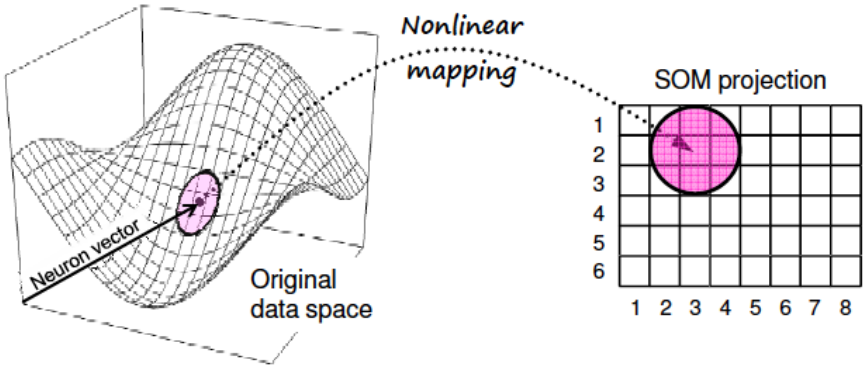
Problem 1

k-means will struggle to find these clusters because they are non-convex shaped. Some points (e.g. in the center of the graph) that lie in one cluster (from the two visibly obvious clusters) can be closer to the cluster center of the other cluster than the points in the other cluster. This can lead to these points being mistook as part of the other cluster. The closeness between data points, since it is defined as the Euclidean distance, does not inform the k-means algorithm about the non-convex nature of the layout of these data points.

Problem 2

The map layer defines a topological ordering over the neurons. The SOM algorithm works by making map space neighbor neurons zone in on to similarities in the data space. This is because the weights associated with all the map space neighbor neurons are moved towards the data points closest to the best matching neuron. This makes weights of map space neighbor neurons adapt towards similar weights in the data space (weights having the same dimensionality as data, makes weight be elements of and move around in the data space). Thus, similarities in the data are captured by map space neighbor neurons. Visualization techniques are needed to see the dissimilarities in data space. These techniques can allow seeing the separation between groups of neurons that represent different data clusters.

The relationship between the map space and data space can be seen in the figure below. The data points lying nearby the highlighted neuron (and its neighborhood) in the data space (left hand side graph), will be represented by spatially close neurons in the map space (right hand side graph).



Ideally, the winning and second place neurons will be close neighbors in the map space, because their weight vectors are similar. Weight vectors that are similar are similar because they get adapted towards similar data points, i.e. points within some cluster.

Problem 3

There are quite a few neighborhood relationship functions that have been investigated in SOM research. In general, the neighborhood should be larger (somewhere half the size of the larger dimension in map space, i.e. height or width of the map whichever is greater) in size in the beginning of learning and gradually become smaller. As long as such a reduction in time of the neighborhood is taken into consideration, different popular neighborhood functions lead to more or less similar learning results.

Typical functions other than Gaussian are:

- **Bubble/flat function:** value of is 1 up to a certain distance in the map space neighborhood, otherwise 0.

$$N(i,j) = \begin{cases} 1 & \text{if } \|i-j\| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

- **Cut Gaussian function:** the Gaussian function multiplied by the flat function, so that it is smooth for close neurons and then cuts straight to zero at a certain distance:

$$N(i,j) = \begin{cases} e^{-\frac{\|i-j\|^2}{2\sigma^2(t)}} & \text{if } \|i-j\| \leq \sigma_{flat} \\ 0 & \text{otherwise} \end{cases}$$

- **Epanechicov function:** piecewise polynomial function that is parabolic for near neighbors and zero above a certain distance:

$$N(i,j) = \max\left\{0,\ 1 - \left(\frac{\|i-j\|}{\sigma}\right)^2\right\}$$