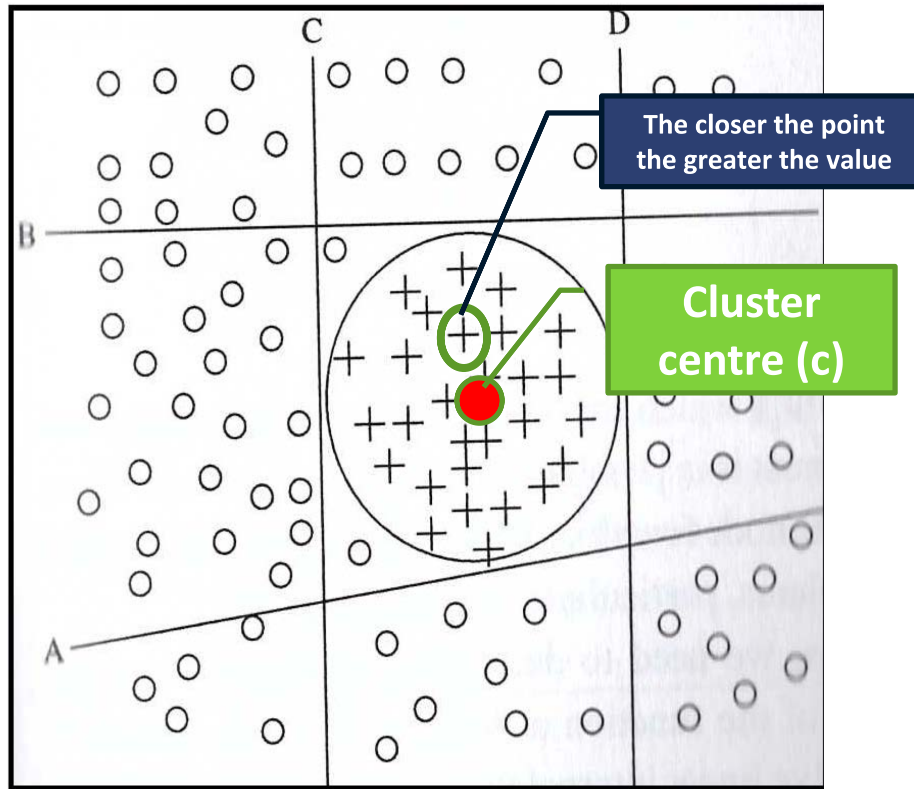
CISC452/CMPE452/COGS 400
Radial Basis Functions and Polynomial Networks

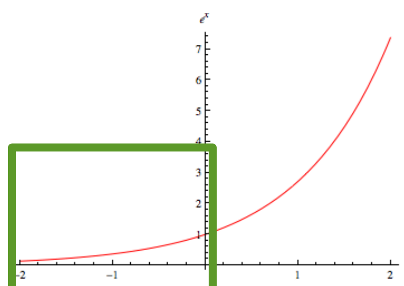
Ch. 4 - Text book

Farhana Zulkernine

Radial Basis Function (RBF)

- Good for cases where all samples of one class are clustered together.
- Possible to solve using a FF network with sigmoidal function with *one hidden layer* having multiple nodes – **RBF is simpler**.
- Instead of 4 or 5 hidden nodes only one node that approximates a circle can be used.
  - The closer a point is to the center, the greater should the output be.
  - $\rho(\|x-c\|) = e^{-\gamma\|x-c\|}$





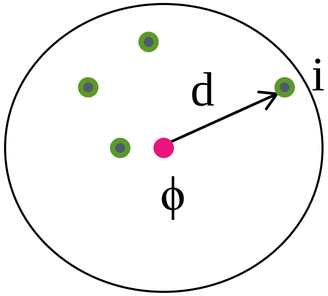
Radial Basis Function (RBF)

- A function is radially symmetric (is an RBF) if its output depends on the distance of the input vector from a stored vector specific to that function.
- Neural networks whose node functions are radially symmetric functions are referred as RBF-nets.
- Typically, RBF-nets use as RBF a non-increasing function ρ of a distance measure u, with ρ(u1) >= ρ(u2) whenever u1 < u2.
- The Gaussian function most widely used in RBF is

$$\rho(u) \propto e^{-u/\sigma^2}$$

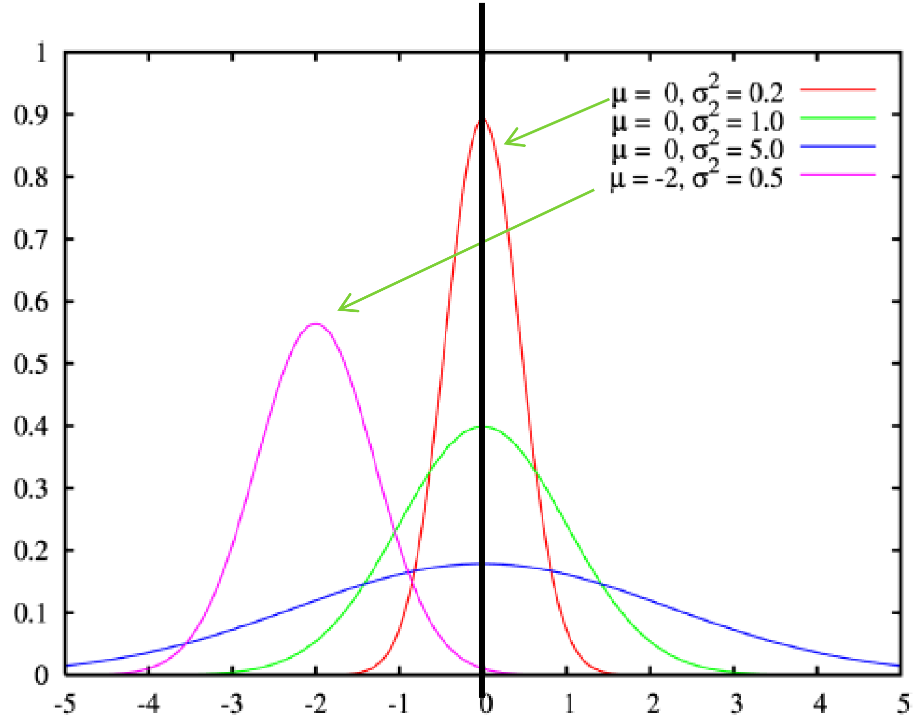
Radial Basis Function (cont…)

- RBF function ρ is applied to the Euclidean distance $d = \| \phi - i \|$, between the center or stored vector ϕ and the input vector i.
- The key idea is that within a given radius (some distance from the stored vector) the output of the node is high and outside it is low i.e., $\rho(d) \propto 1/d$
- RBF-nets are generally called upon *for use in function approximation problems*, particularly for *interpolation*.



General Gaussian Function

- Small σ => very small neighbourhood of interpolation (circle is small).
- Large σ => large circle encompassing all training samples => stored value becomes average of all.
- Euclidean distance is +ve. So, we consider the –ve part of the graph for which μ=0.
- Thus we may use the Gaussian to control the extent to which predicted values depend on the individual training trials.



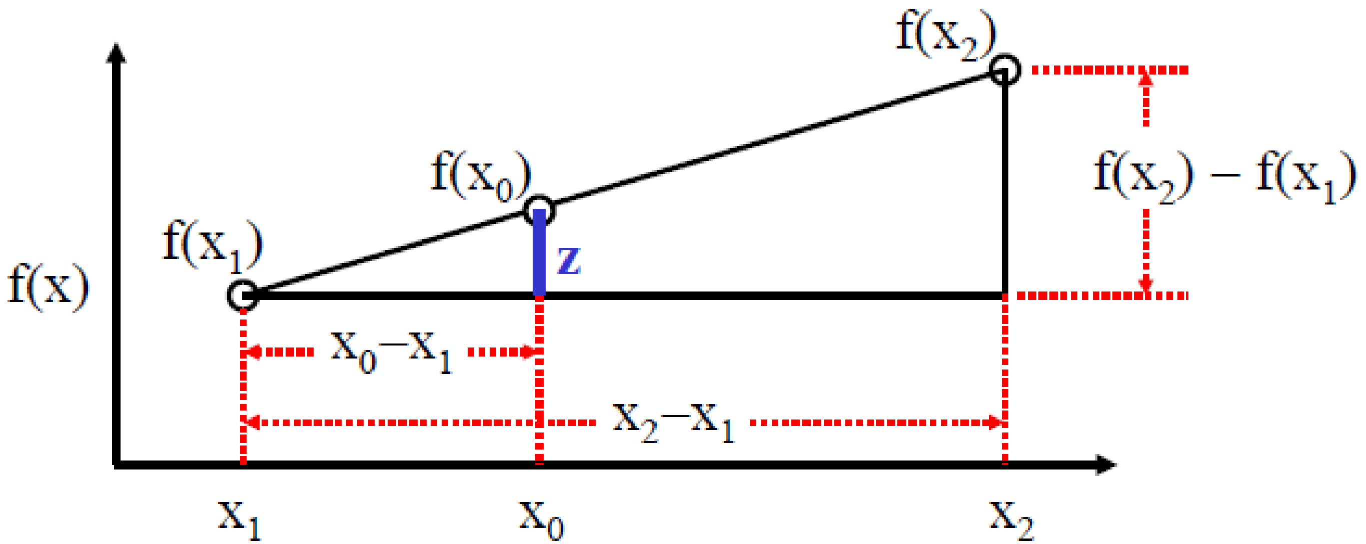
The shape of the Gaussian is controlled by the parameter σ for $f(x) = a \exp(-(x-\mu)/2\sigma^2)$

Linear Interpolation

- In many function approximation problems, we need to **determine the behaviour of a function at a new input, given the behaviour of the function at training samples**. Such problems are often solved by linear interpolation.
  - Given $f(x_1)$ and $f(x_2)$, we need to determine $f(x_0)$ where x_1 and x_2 are one dimensional input samples (training data) and x_0 is the new data point that lies in between x_1 and x_2.

Example

- 1-D interpolation with 2 known points x_1 and x_2, whose output values are $f(x_1)$ and $f(x_2)$. x_0 is in between x_1 and $x_{2.}$
- What will be the value of $f(x_0)$?



$$f(x_0) = f(x_1) + z$$

$$f(x_0) = f(x_1) + \frac{x_0 - x_1}{x_2 - x_1}\,(f(x_2) - f(x_1))$$

$$\boxed{\frac{z}{x_0 - x_1} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}}$$

Example (cont…)

$$f(x_0) = f(x_1) + \frac{(x_0 - x_1)}{(x_2 - x_1)}(f(x_2) - f(x_1))$$

$$= \frac{D_1^{-1} f(x_1) + D_2^{-1} f(x_2)}{D_1^{-1} + D_2^{-1}}$$

where $D_n = \| x_0 - x_i \|$ is Euclidean distance of the new point x_0 from the training samples and $f(x_i)$ is the desired output for training sample x_i.

$$= \frac{f(x_1)(x_2 - x_0) - f(x_2)(x_1 - x_0)}{(x_2 - x_0) - (x_1 - x_0)}$$

$$= \frac{\dfrac{f(x_1)}{(x_1 - x_0)} - \dfrac{f(x_2)}{(x_2 - x_0)}}{\dfrac{1}{(x_1 - x_0)} - \dfrac{1}{(x_2 - x_0)}}$$

$$= \frac{D_1^{-1} f(x_1) + D_2^{-1} f(x_2)}{D_1^{-1} + D_2^{-1}}$$

where $D_i^{-1} = \dfrac{1}{\| x_0 - x_i \|}$

Linear Interpolation (cont…)

For P input samples where each sample is an n-dimensional point on a hyperplane, the equation would become

$$f(x_0) = \frac{(D_1^{-1} f(x_1) + \ldots + D_P^{-1} f(x_P))}{(D_1^{-1} + \ldots + D_P^{-1})} \propto (1/P)\sum_{p=1..P} d_p\, \rho(\| x_0 - x_p \|)$$

Where

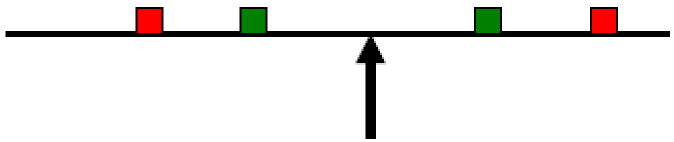
x_p is a sample of n-dimensional training data points considering a total of P nearby points,

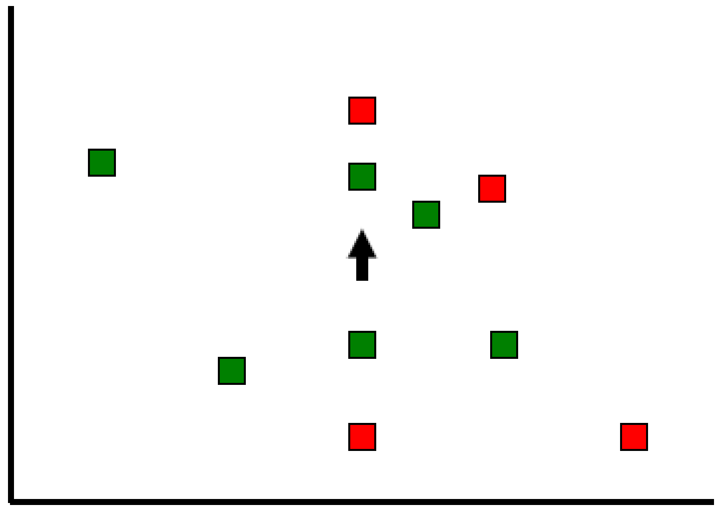
x_0 is the new data (test data) and $f(x_0)$ is its projection

$f(x_p) = d_p$ is the desired outputs for the input sample x_p

$D_p^{-1} \approx \rho(D_p) = \rho(\|x_0 - x_p\|)$ output of the RBF function

Finding Nearest Neighbours





- If there is only one dimension upon which we must interpolate, we know immediately which nearby values need to be considered.
- But with more dimensions it becomes difficult to determine how relevant each of the neighbouring point might be.
- So generally some fixed number of nearest neighbours are used.
- Otherwise for simplicity just use all of the training samples.

RBF (cont…)

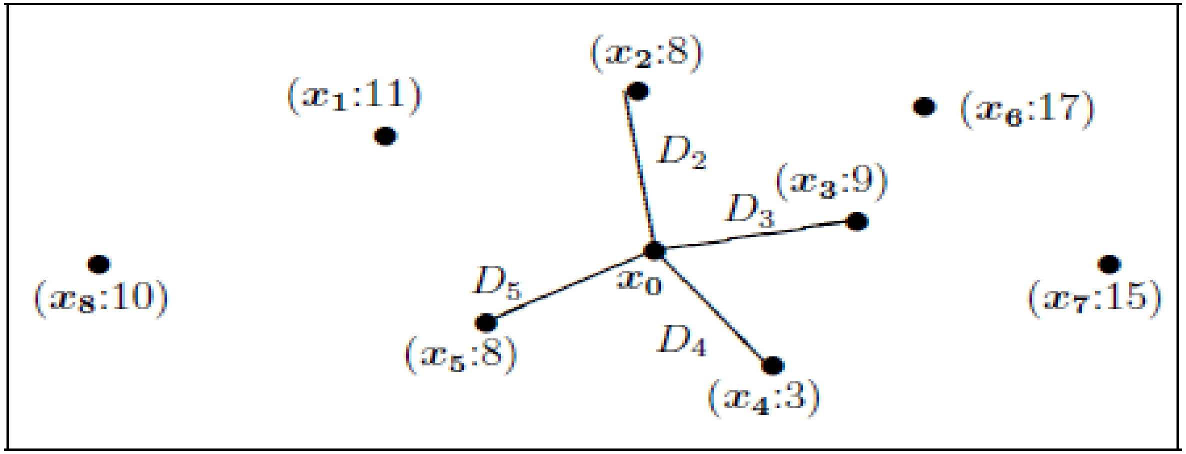
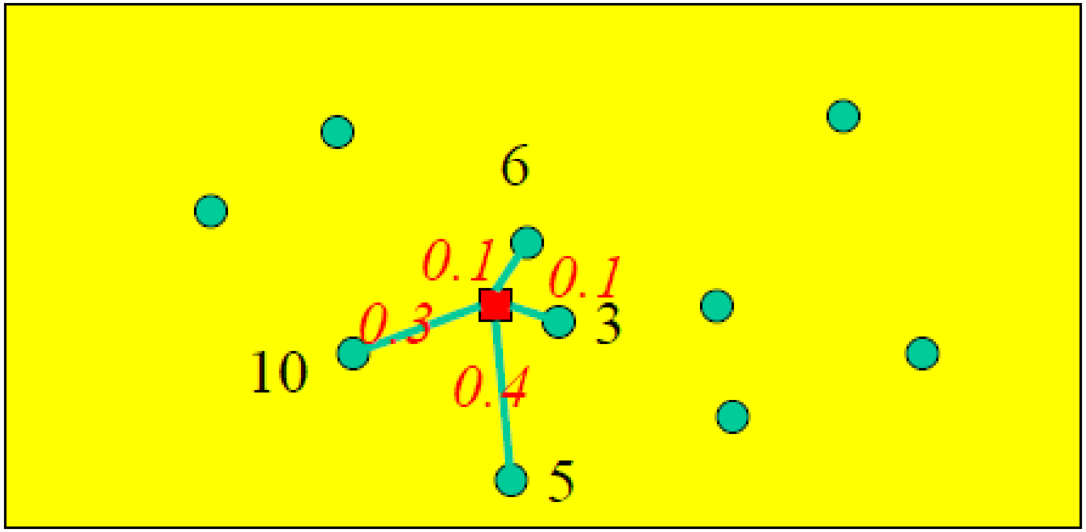


Figure 4.7: D_j is the Euclidean distance between x_j and x_0. $(x_5 : 8)$ indicates that $f(x_5) = 8$. The four nearest observations can be used for interpolation at x_0, giving
$(8D_2^{-1} + 9D_3^{-1} + 3D_4^{-1} + 8D_5^{-1})/(D_2^{-1} + D_3^{-1} + D_4^{-1} + D_5^{-1})$

- In RBF-net, each node implements RBF function $\rho_n(D_n) = D_n^{-1}$.

RBF (cont…)

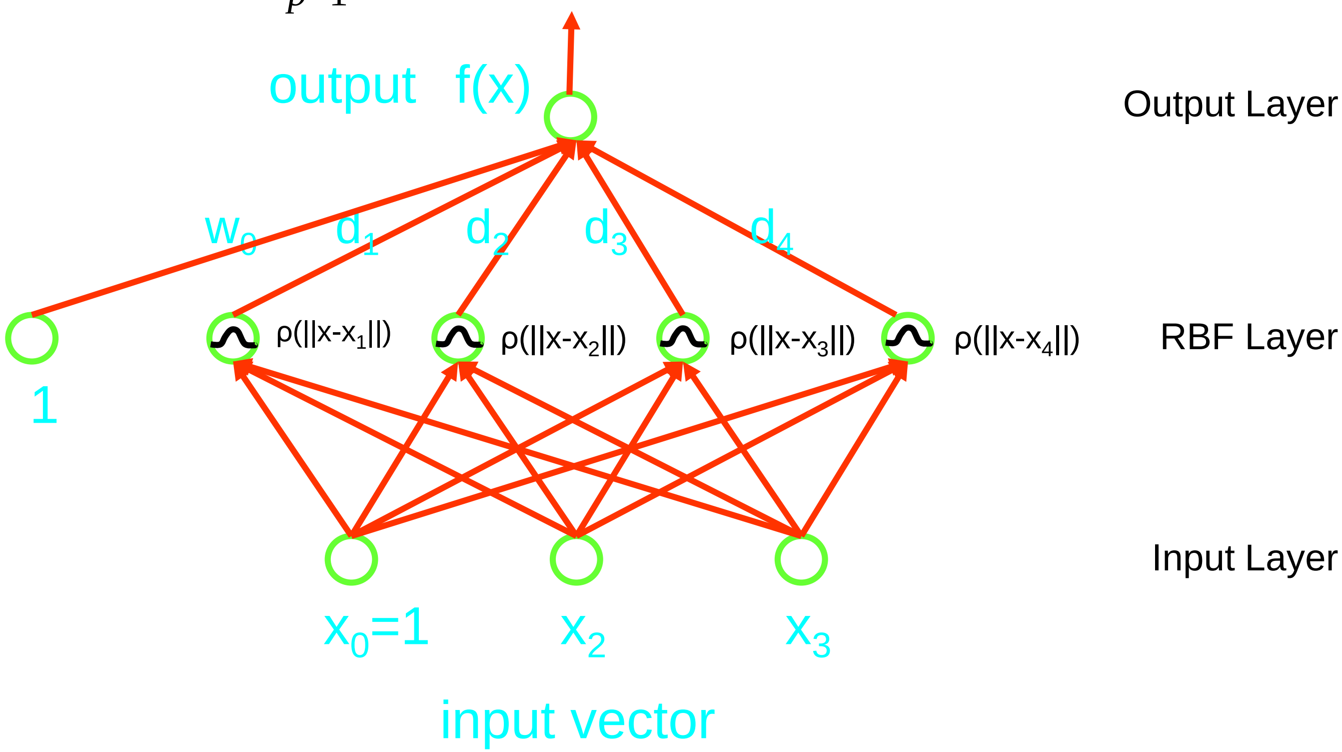
$$\frac{(1/.3 * 10) + (1/.4 * 5) + (1/.1 * 6) + (1/.1 * 3)}{1/.3 + 1/.4 + 1/.1 + 1/.1} = 5.258$$



$P = 4$

The RBF Network

$$f(x) = \frac{1}{P}\sum_{p=1}^{P} f(x_p) D_p^{-1} = \frac{1}{P}\sum_{p=1}^{P} d_p \rho(\| x - x_p \|)$$



- Example: Network function f: $R^2 \rightarrow R$ and 4 training samples
- As many nodes as there are training samples → too large metwork

RBF (cont…)

For network size to be reasonably small, we cannot have one node to represent each x_p. Hence similar training samples are clustered together, and output

$$o = \frac{1}{N}\sum_{i=1}^{N} \varphi_i \rho(||\mu_i - x||)$$

where N is the number of clusters, μ_i is the center of the ith cluster, and φ_i is the desired mean output of all samples of the ith cluster.

Training involves learning the values of

$$w_1 = \frac{\varphi_1}{N}, \ldots, w_N = \frac{\varphi_N}{N}, \mu_1, \ldots, \mu_N$$

minimizing

$$E = \sum_{p=1}^{P} E_p = \sum_{p=1}^{P} (d_p - o_p)^2$$

Learning in RBF Networks

The specific update rules are now:

$$\Delta w_i = \eta_i (d_p - o_p) \exp\left(\frac{-\left(\|\mathbf{x}_p - \boldsymbol{\mu}_i\|\right)^2}{\sigma^2} \right)$$

and

$$\Delta \mu_{i,j} = -\eta_{i,j} w_i (d_p - o_p)(x_{p,j} - \mu_{i,j}) \exp\left(\frac{-\left(\|\mathbf{x}_p - \boldsymbol{\mu}_i\|\right)^2}{\sigma^2} \right)$$

where the (positive) learning rates η_i and $\eta_{i,j}$ could be chosen individually for each parameter w_i and $\mu_{i,j}$.

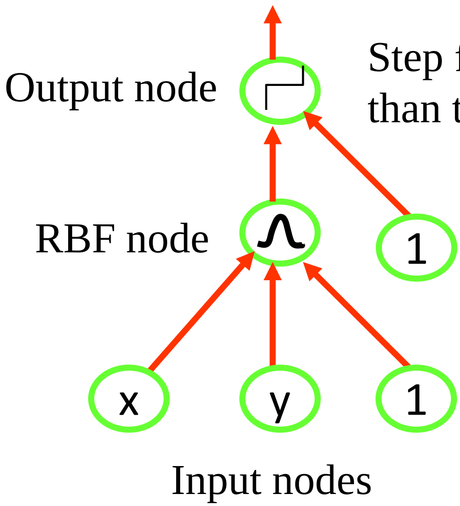
As usual, we can start with random parameters and then iterate these rules for learning until a given error threshold is reached.

- Problem: Requires considerable computation to train for both μ and w
- Better approach: partially offline training

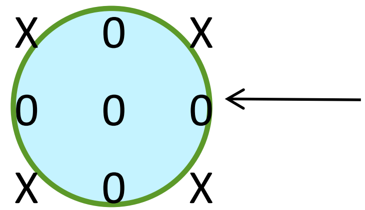
Learning in RBF Networks

- Apply some clustering procedure to estimate cluster centers μ_i, and their spreads (standard deviations) σ_i.
- Use one node per cluster with fixed μ_i
- Gradient descent method as described above is used to determine the weights w_i.

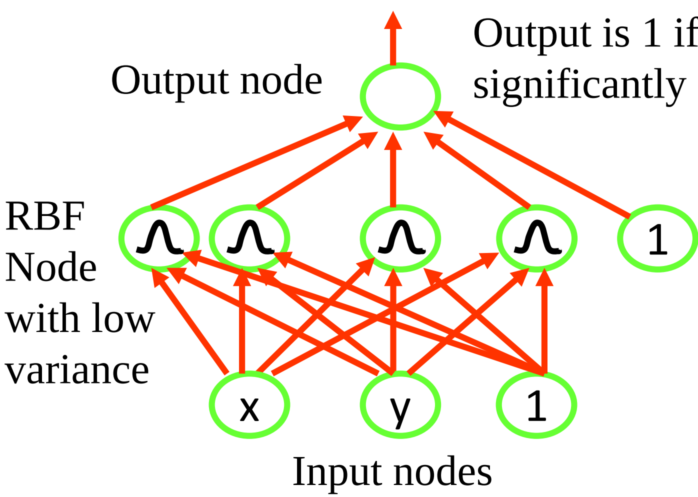
Corner Detection using RBF



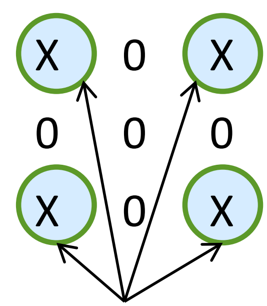
Step function generates -1 when RBF output is higher than that generated by the corners



Region with activation of RBF node centered in the square



Output is 1 if and only if one of the RBF nodes is significantly activated



Region with activation of RBF node centered in the square

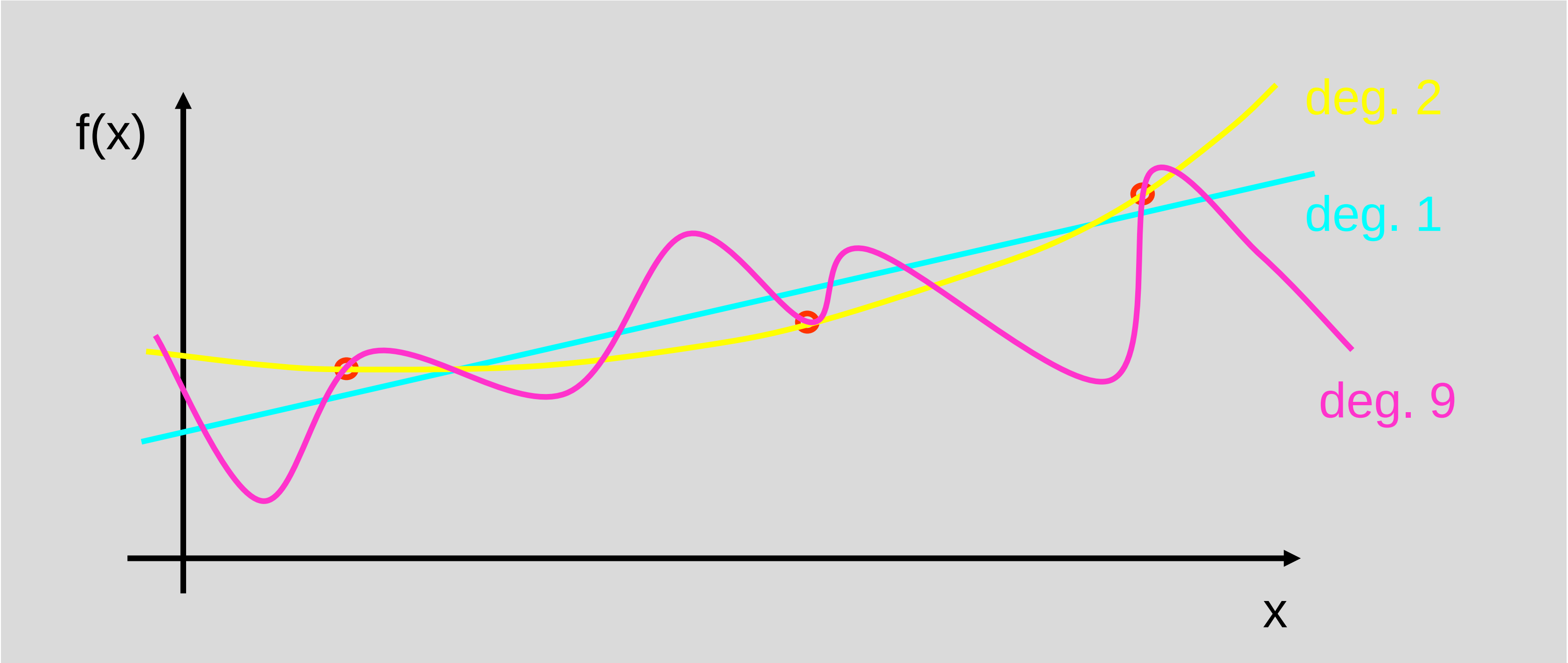
Polynomial Networks

- Many practical problems require computing or approximating functions that are polynomials of the input variables, a task that can require many nodes and extensive training if familiar node functions (sigmoids, Gaussians, etc.) are employed.
- Networks whose node functions allow them to directly compute polynomials and functions of polynomials are referred to as “polynomial networks”.

Polynomial Networks (cont…)

- A single non-input node is sufficient for two-class classification when separating surface is a quadratic or cubic function rather than a hyperplane.
- A polynomial network for approximating a quadratic function would be much smaller than a network using only sigmoid functions.
- Different kinds of polynomial networks have been suggested in the literature.

Supervised Function Approximation



- Obviously, the polynomial of degree 2 provides the most plausible fit.