

Artificial Neural Networks

Radial Basis Function Networks

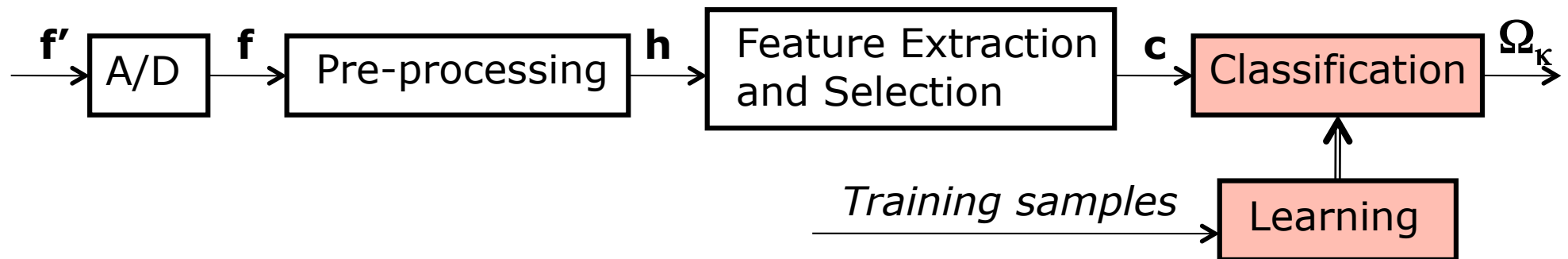


Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Pattern Recognition Pipeline



■ Classification

- Statistical classifiers
 - Bayesian classifier
 - Gaussian classifier
- Polynomial classifiers
- Non-Parametric classifiers
 - k-Nearest-Neighbor density estimation
 - Parzen windows
 - Artificial neural networks



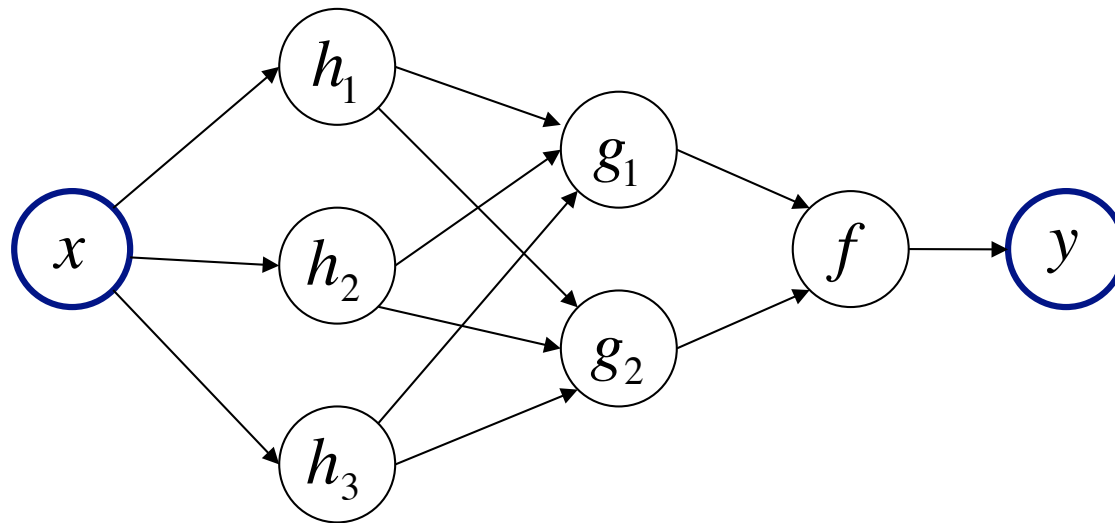
Artificial Neural Network (ANN)

- There is no precise agreed definition among researchers as to what is an artificial neural network.
- Most would agree that it involves a network of simple processing elements (neurons), which can exhibit complex global behavior, determined by
 - the connections between the processing elements and
 - the element parameters.
- In a neural network model, simple nodes (*neurons*, or *processing elements* or *units*) are connected together to form a network of nodes.

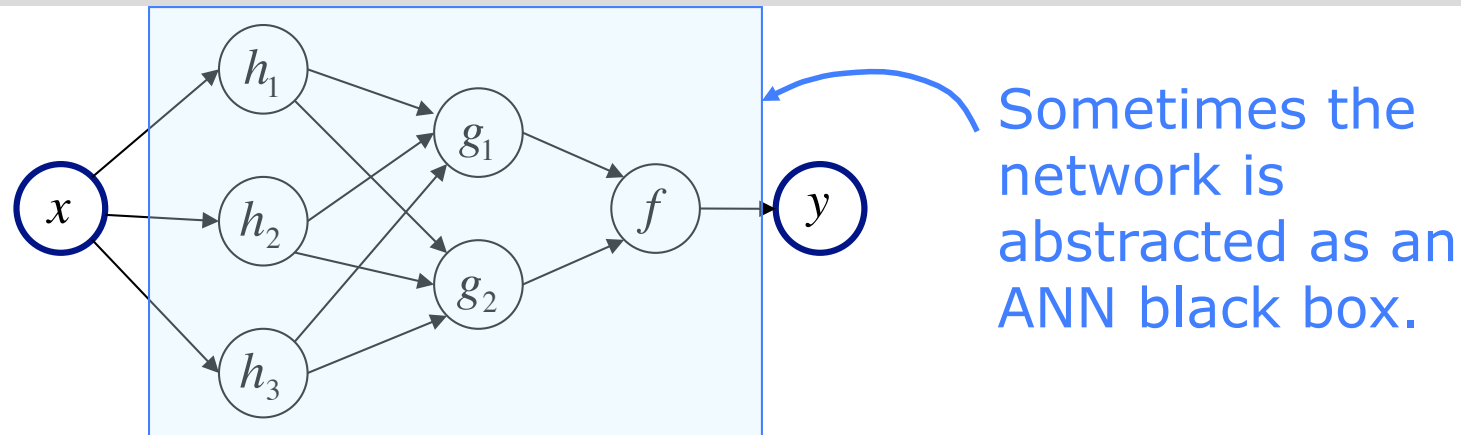


ANN Operation

- In general an ANN operates as a function $f : x \rightarrow y$.
- The “network” arises because the function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions, e.g. $h_j(x)$.

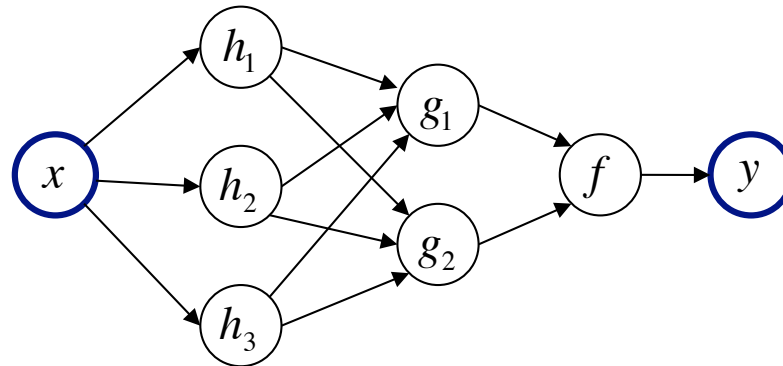


General Form of ANN



- There is great variation in ANNs, depending on:
 - The number of layers
 - Whether there are hidden layers or not
 - The connectivity (We could have feedback loops.)
 - The adaptability
- An ANN does not have to be adaptive. In practice, part of their strength comes from adapting: changing the weights of the connections in order to produce a desired signal flow.

Mathematical Description of an ANN



- A widely used type of composition is the nonlinear weighted sum:

$$f(x) = \phi \left(\sum_i w_i g_i(x) \right)$$

where ϕ is a predefined function that forces the output of a neuron to be in a certain range, typically $[0,1]$ or $[-1,1]$.

- ϕ is often referred to as an **activation function**.



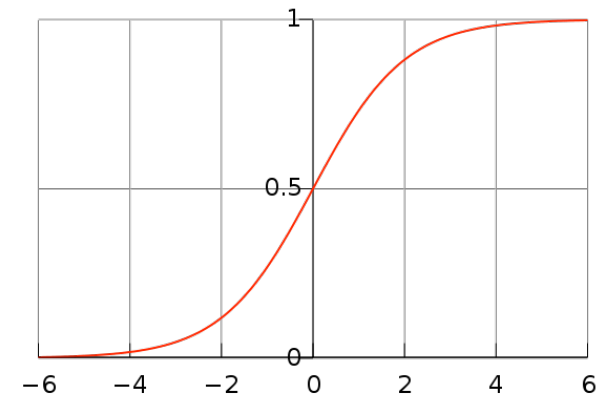
Activation Function

- An activation function tries to mimic the firing of the neuron if the incoming signal is sufficiently strong.

- Mathematically, this is usually achieved with a sigmoid function,

e.g.:

$$\phi(t) = \frac{1}{1 + e^{-t}}$$



- Sigmoid functions have the following characteristic properties:
 - They are differentiable
 - They have 1 inflection point
 - They have a pair of horizontal asymptotes
- Another typical sigmoid function employed in ANNs is the hyperbolic tangent, $\phi(t) = \tanh(t)$.

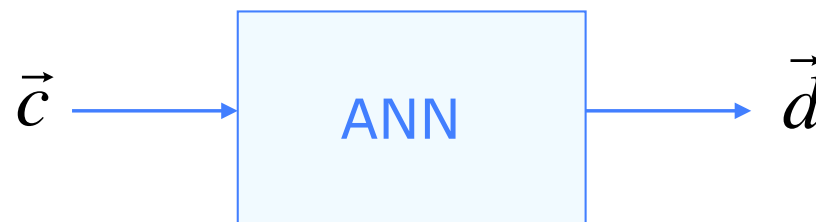


ANN and Classification

- The ANNs that we will examine are used in computing discriminant functions.
- Recall that, a discriminant function for class Ω_k is a polynomial that evaluates to 1 if the feature vector belongs to that class. Otherwise it evaluates to zero.

$$d_k(\vec{c}) = \begin{cases} 1 & \text{if } \vec{c} \in \Omega_k \\ 0 & \text{otherwise} \end{cases}$$

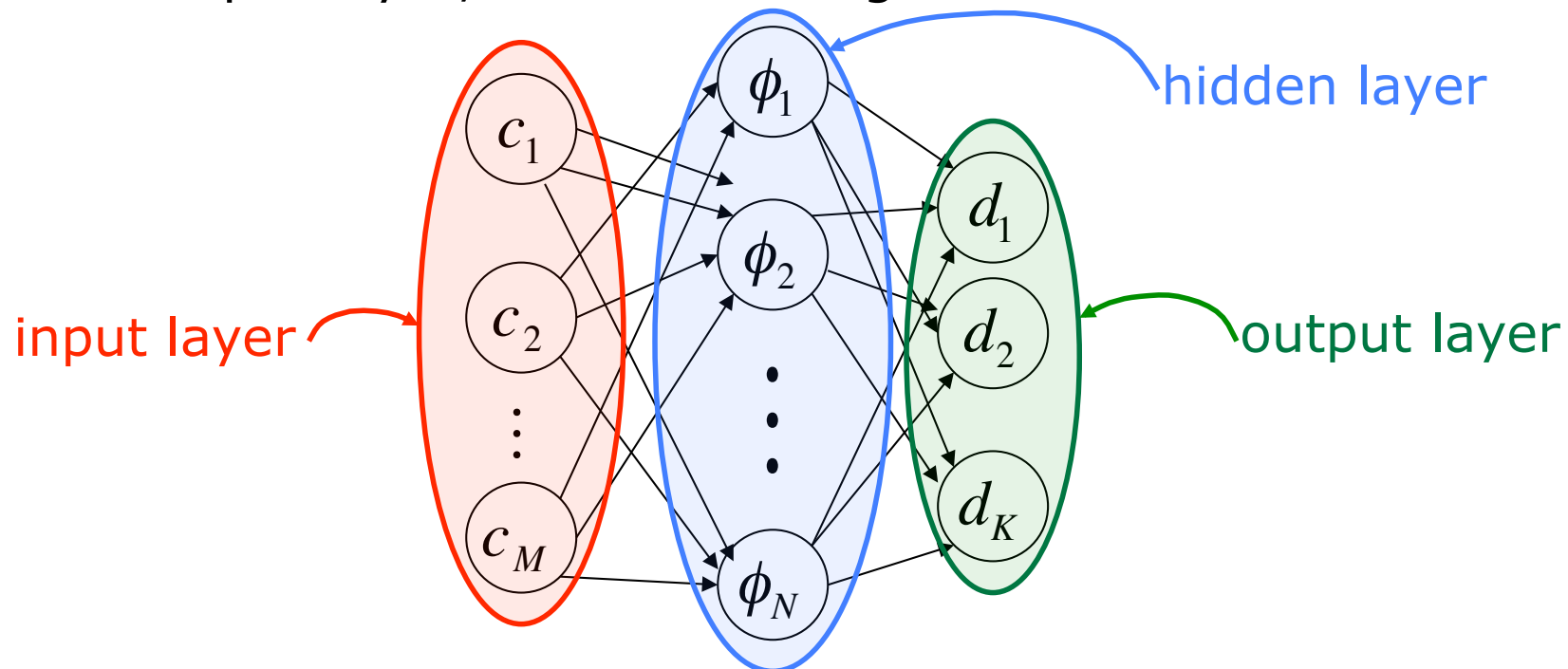
- The input of such an ANN is a feature vector \vec{c} and the output is a discriminant vector, $\vec{d} = (d_1, d_2, \dots, d_K)$.



Radial Basis Function ANNs



- Radial Basis Function (RBF) networks use Radial Basis Functions as their activation function.
- An RBF network is a feed-forward 3 layer network:
 - input layer, \vec{c} in our case
 - a hidden layer, where each node ϕ_i is a separate RBF
 - an output layer, which is a weighted sum of the hidden layers.



Radial Basis Functions



- Radial basis functions were first used in 1987 by Powell.
- He introduced RBFs as a means of mapping an input vector to an output vector.
- A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin, so that

$$\phi(\vec{x}) = \phi(\|\vec{x}\|)$$

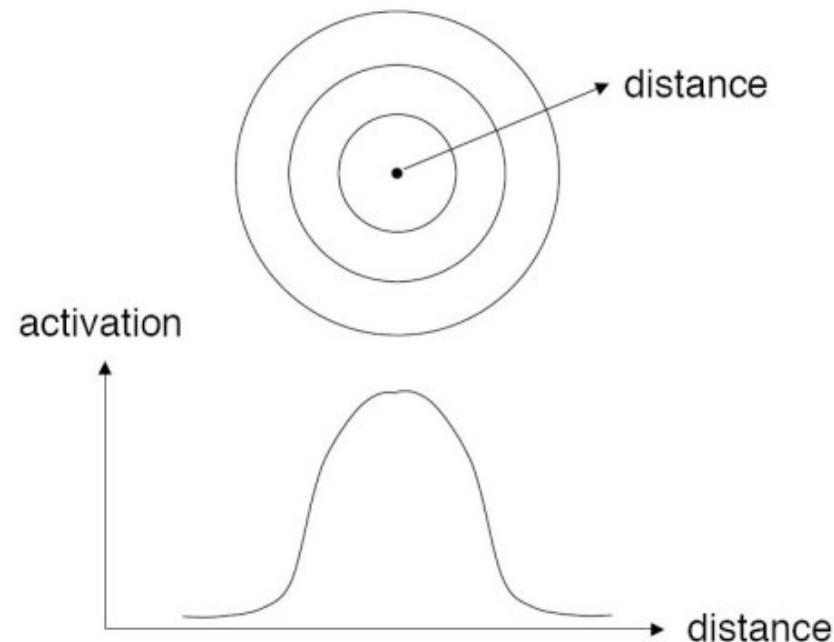
- Alternatively, the RBF can be based on the distance from some other point \vec{q} , called a center:

$$\phi(\vec{x}, \vec{q}) = \phi(\|\vec{x} - \vec{q}\|)$$

Radial Basis Functions - continued



- So RBFs are a type of distance function.
- As a distance function, RBFs have the key characteristic that response decreases monotonically with distance from a central point.
- Its response decreases radially.



Different RBFs



- Any distance function that decreases radially can be considered a radial basis function. Some commonly used RBFs are:
- Two different forms of Gaussians:

$$\phi(\|\vec{x} - \vec{q}\|) = e^{-\frac{(\|\vec{x} - \vec{q}\|)^2}{\sigma^2}} \quad \phi(\|\vec{x} - \vec{q}\|) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{1}{2}(\|\vec{x} - \vec{q}\|)^T \Sigma^{-1} (\|\vec{x} - \vec{q}\|)}$$

- Multiquadric:

$$\phi(\|\vec{x} - \vec{q}\|) = \frac{\sqrt{r^2 + \|\vec{x} - \vec{q}\|^2}}{r^2}$$

- Spline (a.k.a Logarithmic):

$$\phi(\|\vec{x} - \vec{q}\|) = \|\vec{x} - \vec{q}\|^2 \log(\|\vec{x} - \vec{q}\|)$$

RBF and Classification



- Within the context of classification, RBFs work as follows.
- We are given a set of N training samples $\vec{c}_1, \vec{c}_2, \dots, \vec{c}_N$ and we want to find the best discriminant functions.
- One radial basis function (RBF) approach is to use a set of N basis functions, each centered around one of the training samples, i.e. $\vec{q}_i = \vec{c}_i$.
- Given a new feature vector \vec{c} we use RBFs to compute how far away it is from each of the training samples.

$$\phi(\vec{c} - \vec{c}_i) = \phi_i(\vec{c})$$

RBF and Classification – continued



- The discriminant function is then treated as a linear combination of these radial basis functions.

$$d_{\kappa}(\vec{c}) = \sum_{i=1}^N w_i \phi(\|\vec{c} - \vec{c}_i\|) = \sum_{i=1}^N w_i \phi_i(\vec{c})$$

- In this type of RBFs training corresponds to the estimation of the weights w_i from the training data.
- In more detail, recall that each $d_{\kappa}(\vec{c})$ is a binary function. Thus the training set has the form:

$$T = \left\{ \left(\vec{c}_l, d_{\kappa(l)}(\vec{c}_l) \right), l = 1, 2, \dots, N \right\}$$

where $d_{\kappa(l)}(\vec{c})$ is the discriminant function of the class $\Omega_{\kappa(l)}$ to which the sample \vec{c}_l belongs.

RBFN Training



- So for each training pair $(\vec{c}_l, d_{\kappa(l)}(\vec{c}_l))$ we have:

$$d_{\kappa(l)}(\vec{c}_l) = \sum_{i=1}^N w_i \phi(\|\vec{c}_l - \vec{c}_i\|)$$

- This can be written as a vector product:

$$d_{\kappa(l)}(\vec{c}_l) = (w_1, w_2, \dots, w_N) \begin{bmatrix} \phi(\|\vec{c}_l - \vec{c}_1\|) \\ \phi(\|\vec{c}_l - \vec{c}_2\|) \\ \vdots \\ \phi(\|\vec{c}_l - \vec{c}_N\|) \end{bmatrix}$$

$$d_{\kappa(l)}(\vec{c}_l) = (\phi(\|\vec{c}_l - \vec{c}_1\|), \phi(\|\vec{c}_l - \vec{c}_2\|), \dots, \phi(\|\vec{c}_l - \vec{c}_N\|)) \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

RBFN Training - continued



- Since there are N samples in my training set, I have N such equations.

$$d_{\kappa(1)}(\vec{c}_1) = \left(\phi(\|\vec{c}_1 - \vec{c}_1\|), \phi(\|\vec{c}_1 - \vec{c}_2\|), \dots, \phi(\|\vec{c}_1 - \vec{c}_N\|) \right) \vec{w}$$

$$d_{\kappa(2)}(\vec{c}_2) = \left(\phi(\|\vec{c}_2 - \vec{c}_1\|), \phi(\|\vec{c}_2 - \vec{c}_2\|), \dots, \phi(\|\vec{c}_2 - \vec{c}_N\|) \right) \vec{w}$$

$$\vdots$$

$$d_{\kappa(N)}(\vec{c}_N) = \left(\phi(\|\vec{c}_N - \vec{c}_1\|), \phi(\|\vec{c}_N - \vec{c}_2\|), \dots, \phi(\|\vec{c}_N - \vec{c}_N\|) \right) \vec{w}$$

which can be written more compactly as:

$$\vec{d}' = \Phi \vec{w}$$

$$\Rightarrow \vec{w} = \Phi^+ \vec{d}'$$

Important Comment on RBFN Training



- If we have many feature vectors in our training data and we have an RBF estimate for each individual training sample we end up with too many RBFs, too many nodes => Slow training and **Overfitting!!**
- Solution: Use centers of clusters of feature vectors for the RBFs, instead of the individual feature vectors.
- Each RBF is now centered around $\vec{\mu}_j, j = 1, 2, \dots, s$ instead of $\vec{c}_i, i = 1, 2, \dots, N$:

$$\phi(\vec{c} - \vec{\mu}_j) = \phi_j(\vec{c})$$

Updated Training of RBFNs



■ 2-stage process:

1. Unsupervised selection of RBF centers $\vec{\mu}_j$

K-means:

pick s $\vec{\mu}_j$ values at random.

Assign each training sample to its nearest $\vec{\mu}_j$.

Recompute $\vec{\mu}_j$ as the mean value of the samples of cluster j .

Repeat this process until the $\vec{\mu}_j$ s are stabilized.

If using a Gaussian RBF, use MLE to compute Σ_j

2. The estimation of \vec{w} can be done as before via linear algebra methods (e.g. SVD)

Weaknesses of the 2-stage Approach



- The estimation of $\vec{\mu}_j$ and Σ_j is not guided by the discriminant function that is used to compute \vec{w} .
- Hence we have a non-symmetric approach.
- Stage 2 relies on the results of Stage 1.
- Thus, we have a propagation of estimation errors which often means an amplification of errors.
- Better solution: use an integrated, fully supervised approach like the Orthogonal Least Squares approach.

RBFN Training via Orthogonal Least Squares



- Main idea of OLS: Do not cluster as a preprocessing step.
- Rather do a sequential selection of the centers $\vec{\mu}_j$ which leads to the largest reduction in the sum of squared errors.
- Which sum of squared error (SSE)?
- The difference between the computed and the expected result (value) of the discriminant functions:

$$SSE = \sum_{i=1}^N \left(\hat{d}_i - \vec{d}_i \right)^2$$



Orthogonal Least Squares Algorithm

1. Start with N pairs $(\vec{c}_l, d_{\kappa(l)}(\vec{c}_l))$ and $s=0$
2. For each training pair i of the $n=N-s$ features vectors
 - 2a. Add the current feature \vec{c}_i to the s centers
 The new vector becomes an additional $\vec{\mu}_j$
 - 2b. Compute the weights \vec{w}
 Use linear algebra as previously described.
 - 2c. Compute the sum of squared error, SSE.
3. Out of the n candidate cluster centers, select the one with the smallest SSE as the next cluster center.
4. $s++$;

Repeat until the desired # of clusters is reached.

References



1. The sigmoid function plot is courtesy of Wikipedia <http://en.wikipedia.org/wiki/File:Logistic-curve.svg>
2. The RBF graph is courtesy of P. Sherrod <http://www.dtrek.com/rbf.htm>
3. Additional information on Orthogonal Least Squares can be found at S. Chen, C.F.N. Cowan and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, March 1991.