NN in Low Dimensional Problems

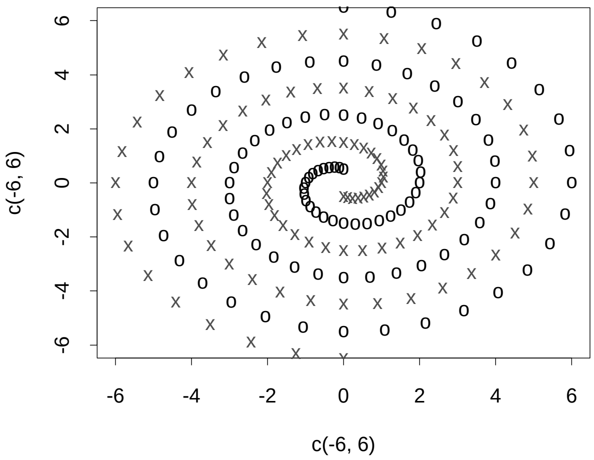
- Consider a highly non-natural problem for NN

- Low dimensional (highly nonlinear)

- Study the ability to control model properties *Capacity*, *Variance*, *Bias/Smoothness*

- Easy visualization of Generalization Properties

The Two-Spiral Problem

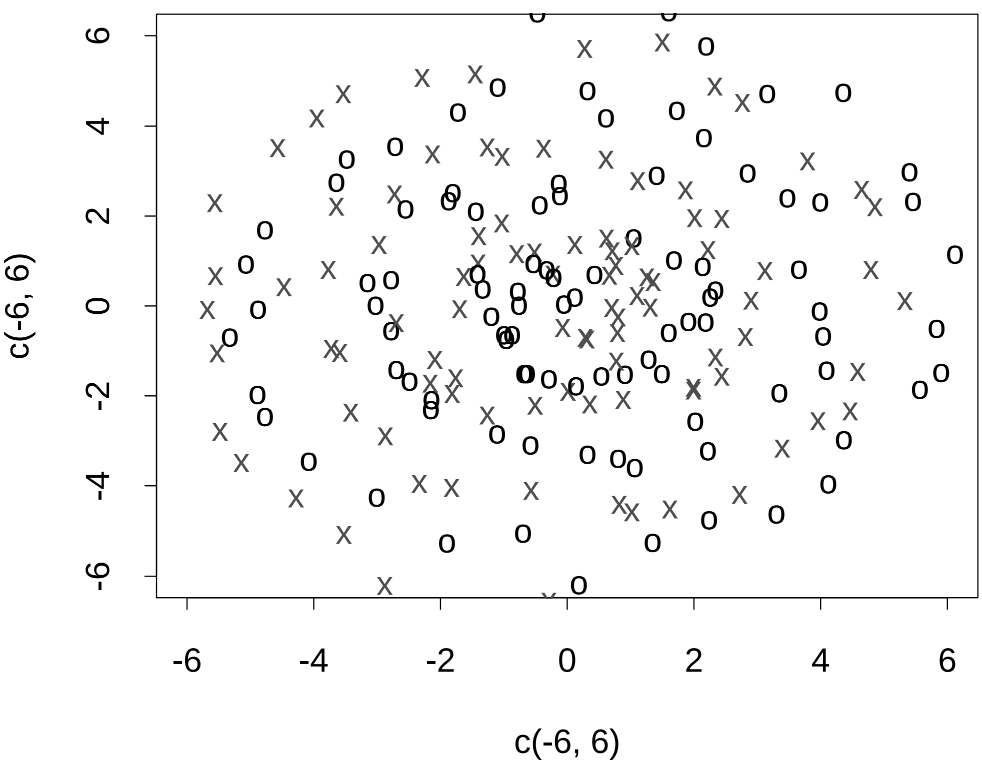


- 194 X,Y values. Half produce a 1 output, and half produce 0
- Lang and Witbrock (1988) proposed a 2-5-5-5-1 net (138 weights)
- Fahlman Lebiere (1990) Cascade Correlation architecture
- Baum and Lang (1991) Net of $2 - 50 - 1$ could be consistent with training set, but could not be found from random initial weights
- Deffuant (1995) suggested the "Perceptron Membrane": piecewise linear discriminating surfaces using 29 perceptrons. Non smooth solution

The BEN Algorithm

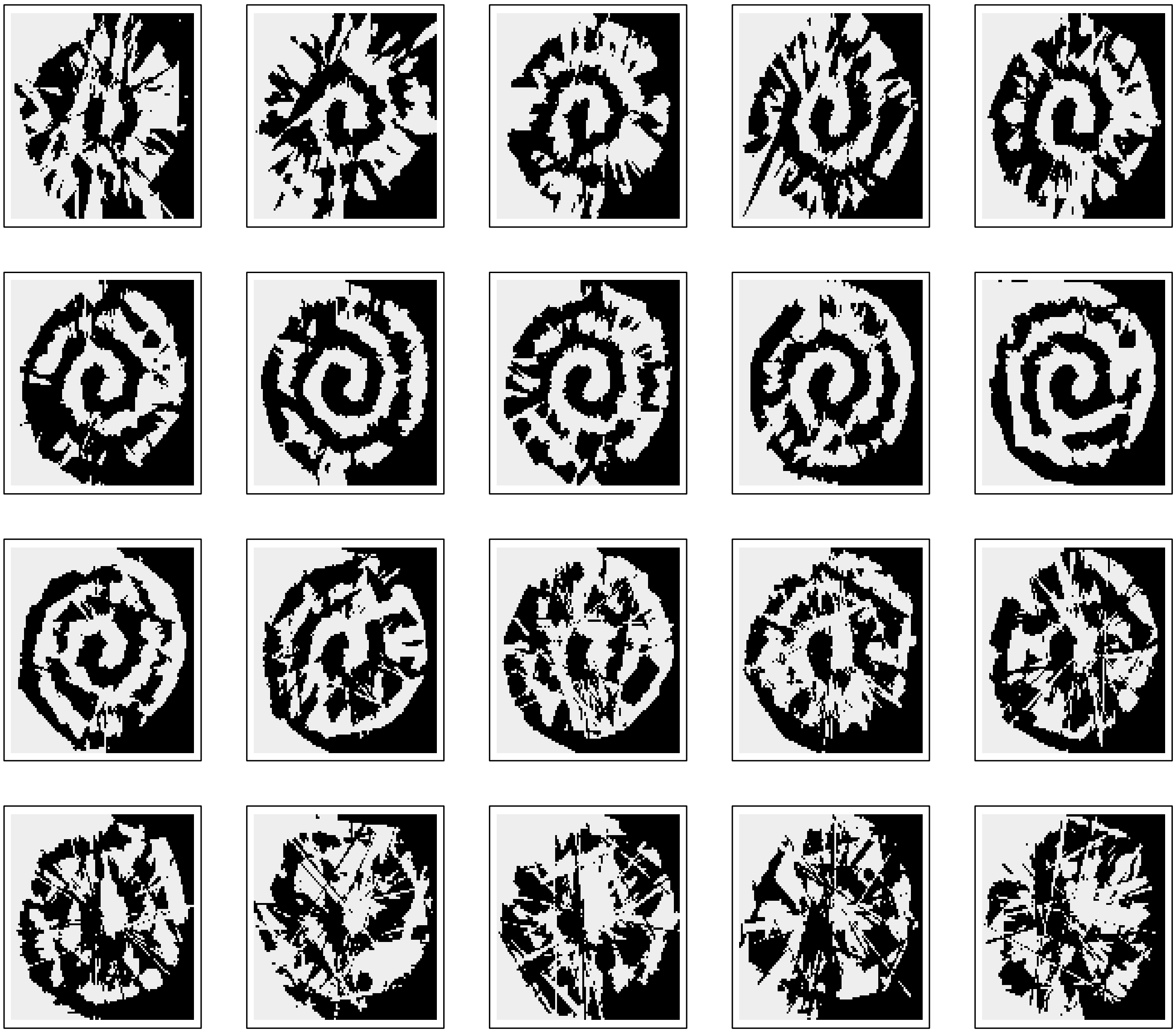
- Let $\{(x_i, y_i)\}$ be a set of training patterns for $i = 1, \ldots, N$.
- Set the noise level ϵ to zero
- For a noise level ϵ estimate an optimal penalty term for weight decay λ:
  - Fix a size K for the bootstrap sample, such that $K \gg N$ (we used $K = 10N$).
  - Let $s_1, s_2, \ldots, s_K$ be a set of indices, chosen from a uniform distribution, $s_i \sim U(1, N)$.
  - For a set of noise SD ϵ, create a Gaussian noisy bootstrap re-sample of the training set inputs: $\{x_{s_i} + \epsilon_i\}_{i=1,\ldots,K}$ and the corresponding re-sampled outputs $\{y_{s_i}\}_{i=1,\ldots,K}$ where $\epsilon_i \sim N(0, \epsilon)$.
  - Train several networks with the noisy samples using weight decay λ.
  - Generate an ensemble average of the set of networks.
  - Choose via cross-validation or a test set, the optimal noise level ϵ.
- Repeat the process for the new choice of noise ϵ until there is no improvement in prediction.

The noisy spirals



Additional Gaussian noise (SD=0.3)

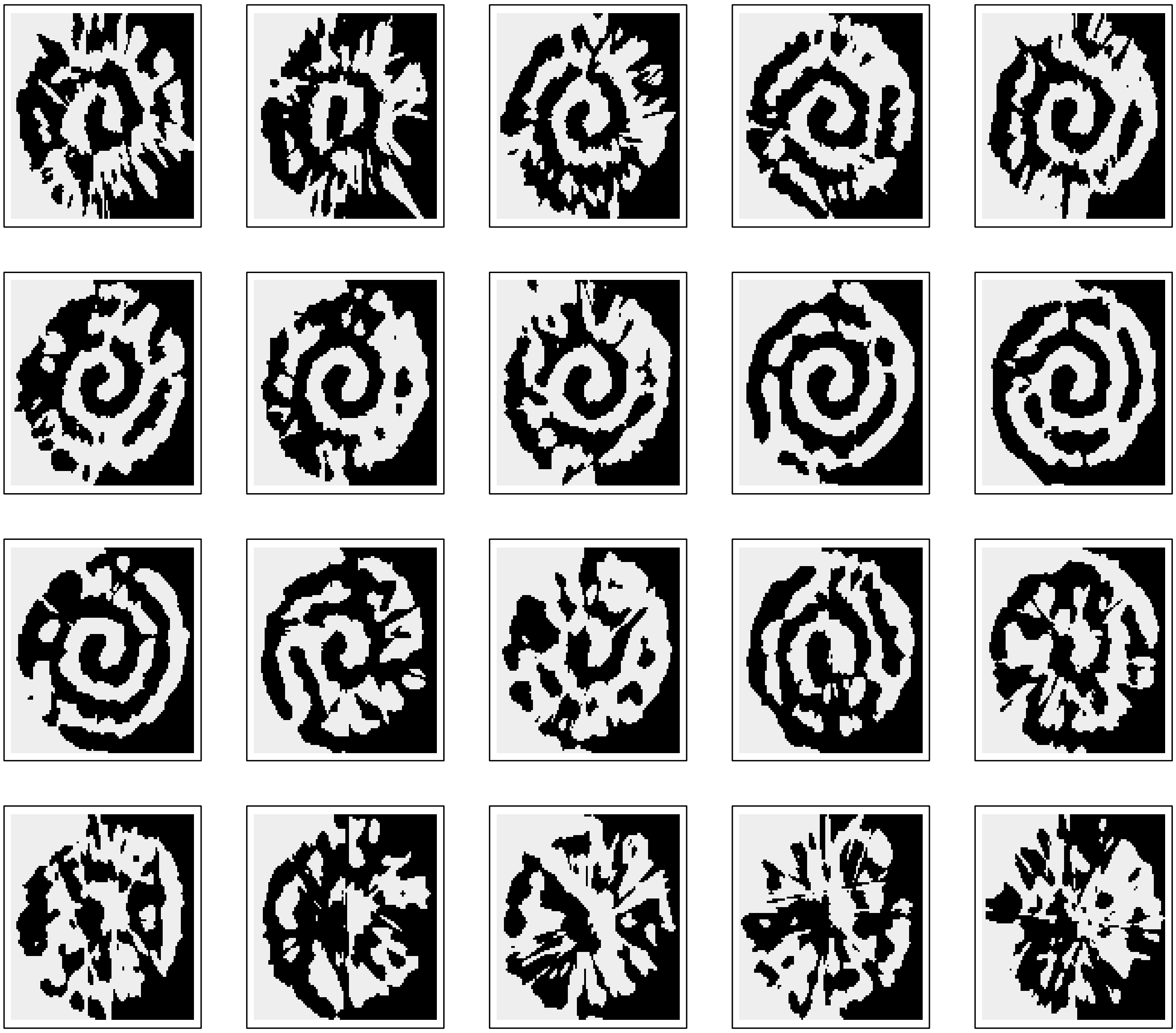
Different noise levels



Results of training with different noise levels.

$$\epsilon = 0, \ldots, 0.8$$

Different noise levels and optimal weight decay



Summary: 40 Net Ensemble

Top left: No constrains (no WD or noise).

Top right: Optimal WD ($\lambda = 3e^{-4}$), no noise.

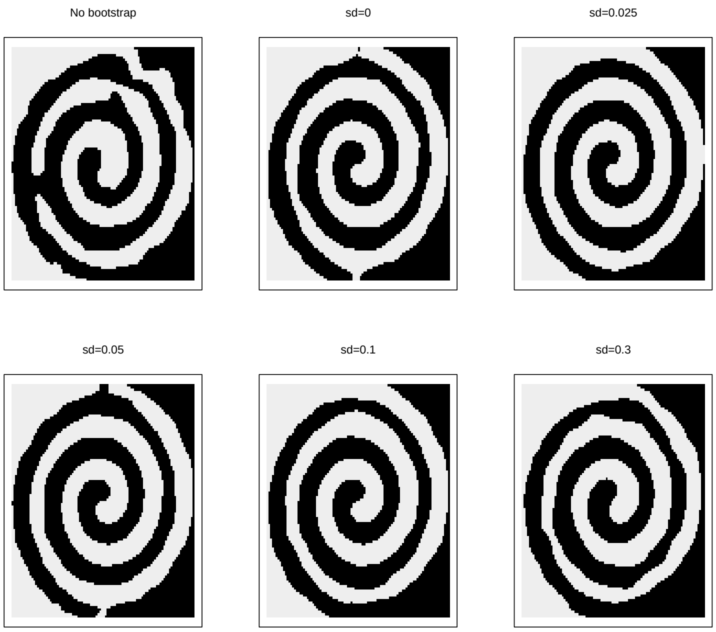
Bottom left: Optimal noise (Gaussian SD=0.35), no WD.

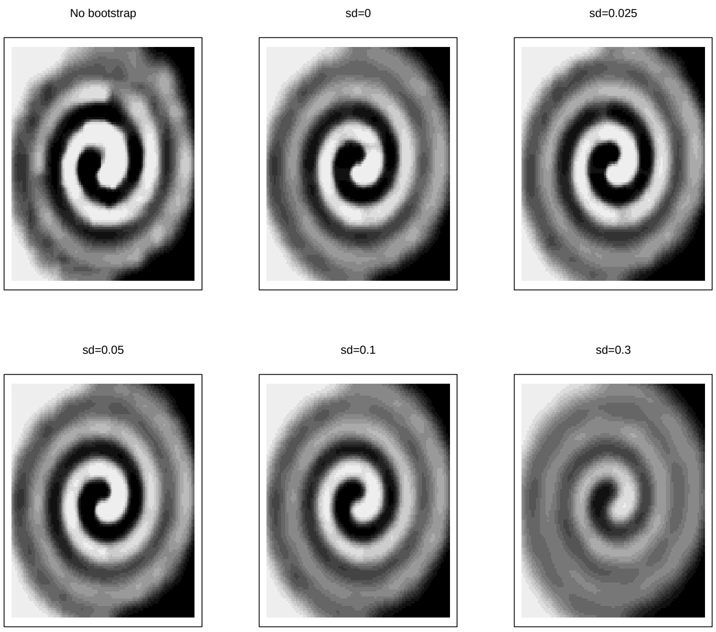
Bottom right: Optimal noise, optimal WD.

Local GAM

- Local generalized additive model (Hastie Tibshirani, 1986)
- Uses a polynomial fit of degree 1 (optimal)
- The span parameter determines the degree of locality of the estimation
- Ideal model for the problem
  - Local with control on locality
  - No ridge constraints
  - Provides a unique model (less variability)
  - Smoothness controlled by locality and degree of the polynomial

Noisy GAM





Average of 20 GAMs with varying degrees of noise

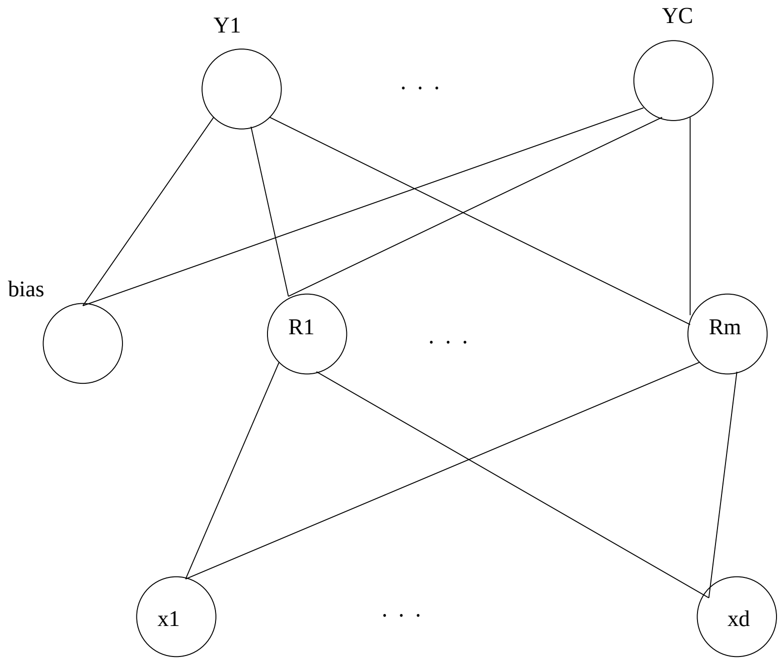
Conclusions

- NN are easy to regularize
  - Weight decay (smoothing)
  - Ensemble average
  - Bootstrap with noise is effective and is not equivalent to smoothing

- Bootstrap with noise is useful for other models' regularization and is **not** equivalent to smoothing

Radial Basis Functions: Global Optimization

Shimon Cohen and Nathan Intrator



RBF network: M hidden units, and C outputs

- For regression C (the number of outputs) is one

- Common RBF functions:

$$\phi(x) = \exp^{\frac{-(x-m)^2}{2r^2}}.$$

$$\phi(x) = (x^2 + \sigma^2)^{-a}, a > 0.$$

Calculation of the forward weights

- The output of the network is a linear combination of the basis functions:

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x)$$

- In matrix form:

$$\begin{aligned} Y &= W\Phi^T \\ Y^T &= \Phi W^T \end{aligned}$$

- Generally Φ is not a square matrix and thus can not be inverted. However, $\Phi^T\Phi$ is a square matrix that has an inverse if its determinant is not zero, and using the pseudo-inverse matrix:

$$\Phi' = (\Phi^T\Phi)^{-1}\Phi^T,$$

  we get

$$W^T = pinv(\Phi)Y^T.$$

  The pseudo-inverse has the property $\Phi'\Phi = I$.

Finding the centers for the RBF's

- Inefficient way: take each observation point as a center.

- Disadvantages

  - Many prototypes for large training data.

  - Likely to over-fit.

K-means Clustering

Input: patterns and requested number of clusters.
Output: clusters centers.

1. Initialization step: take k patterns randomly as initial centers.

2. Loop: for each pattern find closest center and associate it with its cluster.

3. Recompute centers and mean of patterns in cluster.

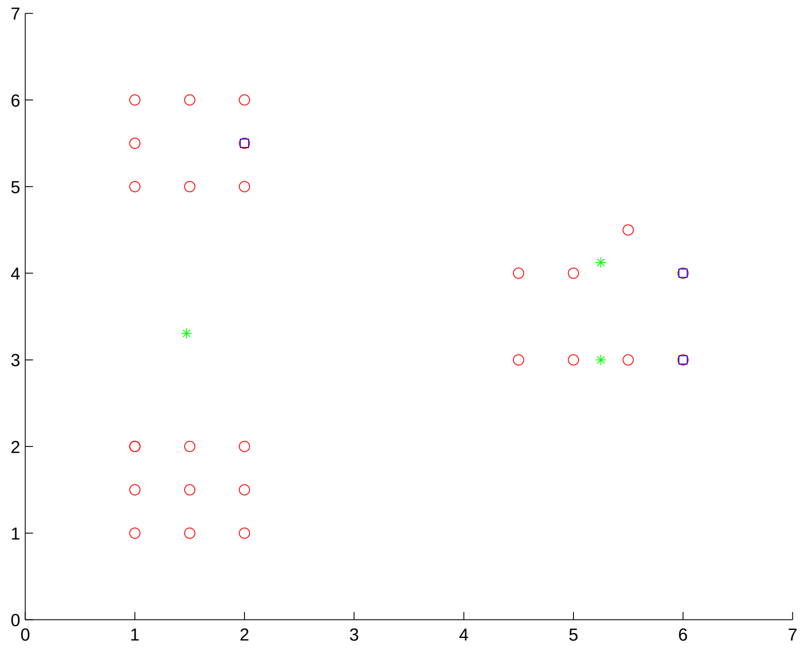
4. Any vector move continue otherwise: done.

5. End: output centers.

Problems with K-means clustering

- The initial vector set can derive the algorithm to a poor local minimum

- Clusters which are not radials cannot be handled properly by this algorithm.



Bad initial vectors of K-means, two were chosen from the same cluster

Addressing the first problem

1. Chose first vector

2. *Loop:* while initial set smaller then k

3. Choose the next vector which has the largest minimum distance to the current initial set.

The EM Algorithm

We are trying to estimate the following:

$$
\begin{aligned}
\hat{m}_j &= \frac{\sum_n P(j|x^n)x^n}{\sum_n P(j|x^n)} \\
\hat{\sigma}_j^2 &= \frac{1}{d}\frac{\sum_n P(j|x^n) \parallel x^n - m_j \parallel^2}{\sum_n P(j|x^n)} \\
\hat{P}(j) &= \frac{1}{N}\sum_n^N P(j|x^n).
\end{aligned}
$$

- This is a set of nonlinear equations since they are coupled.
- The EM Algorithm gives an iterative solution.

- In the E step (expectation) we substitute the latest estimation of $\widehat{P}(j)$.

- In the M step (maximization) we minimize the negative log

likelihood:

$$
\begin{aligned}
E &= -\ln L \\
&= -\sum_n \ln p(x^n) \\
&= -\sum_n \ln \sum_j p(x^n|j)P(j).
\end{aligned}
$$

We use the following for $p(x|j)$

$$p(x|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} exp(-\frac{\| x^n - m_i \|^2}{2\sigma_j^2}),$$

and using Bayes rule we get the mixture (or weight) values

$$P(j|x) = \frac{p(x|j)P(j)}{p(x)},$$

which gives the desired output

$$p(x) = \sum_{j=1}^{M} p(x|j)P(j).$$

$P(j)$, the prior probability of a data point generated from the J'th component has to satisfy:

$$\sum_{j=1}^{M} P(j) = 1.$$

Also, the components density are normalized so:

$$1 = \int p(x|j)dx.$$

- Following the EM which adjusts the RBF centers, we recompute the weights.

- In the EM algorithm the widths of the RBF may go to zero.

- Bishop resets the center width to the initial values if this happens.

Estimation via gradient descent

The centers can be refined via gradient descent as well:

$$E = \sum_{n=1}^{N} E^n.$$

$$E^n = \frac{1}{2} \sum_{k=1}^{M} (y_k(w; x^n) - t_k^n)^2.$$

We define:

$$\delta_k^n = (y_k(w; x^n) - t_k^n).$$

The derivatives of the error with respect to the weights is given by:

$$\frac{\partial E^n}{\partial w_{ij}} = \sum_{n=1}^{N} (\delta_i^n \phi_j^n).$$

The derivatives of the error with respect to the center i is:

$$\frac{\partial E^n}{\partial m_i} = \sum_{k=1}^{C} (\delta_k^n w_{ki} \partial \phi_i / \partial m_i).$$

Where C is the number of the network outputs.

The derivatives of the error function with respect to radius i is:

$$\frac{\partial E^n}{\partial r_i} = \sum_{k=1}^{C} (\delta_k^n w_k i \partial\phi_i / \partial r_i)$$

The derivatives of the RBF function with respect to center i:

$$\frac{\partial \phi_i}{\partial m_i} = \phi_i(x^n) \frac{x^n - m_i}{r_i^2}.$$

The derivatives of the RBF function with respect to the radius i:

$$\frac{\partial \phi_i}{\partial r_i} = \phi_i(x^n)(\frac{\| x^n - m_i \|^2}{r_i^3} - \frac{d}{r_i}).$$

Where d is the dimension of the input pattern vector.

Mark Orr's RBF

The algorithm uses a different radius for each axis for each cluster:x

$$\phi(x) = \sum_{j=1}^{d} \text{exp}[-(x_j - m_j)^2/2r_j^2].$$

It uses regression trees to estimate the centers and the radii:

- The root of the tree is the smallest hyper-rectangle which contains all the data.
- For each dimenstion the mean is the average between the maximum and minimum point.
- The size is half of the maximum diffrence in each dimension.

Detailed algorithm

1. Divide the input space in each dimension, to right and left nodes, to minimize a criterion function.

$$E(k,b) = \sum_{i=1}^{p_l} (y_i - y_l)^2 + \sum_{i=1}^{p_r} (y_i - y_r)^2.$$

   Where y_l is the average on the left side and y_r is the average on the right side. This is done on each partition separatly. The dimension and partition which is choosen is the one which reduces the error most. The stopping crieteria is the number of points in each node. (A node is not divided if it has too small number of observations.)

2. The RBF net is then built using a likelihood maximization (via Bayes formula as before) when considering each possible combination of subtree nodes.