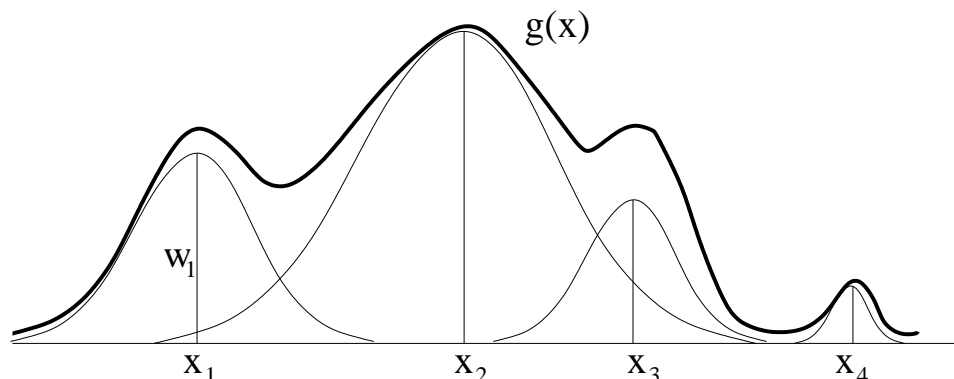


Radial Basis Functions

The next learning model that we present is the radial basis function (RBF) network. There are many theoretical justifications for this model, ranging from regularization theory, to noisy input learning. We, however, will present the RBF as a natural generalization of the r -Near Neighbors classification rule that has the two key properties that we require of learning models, namely universal approximation and an efficient learning algorithm.

Instead of picking K or r in the nearest neighbor type rules, suppose that we allow every point to influence the value of $g(\mathbf{x})$. Then, each point cannot affect the value equally, otherwise, the function would be a constant. Points that are further away from \mathbf{x} should contribute less than closer points. Some points may be more important than others, so we should also allow “more” important points to influence the value of $g(\mathbf{x})$ more than “less” important points. In order to achieve this, we can imagine placing a “bump” function at each data point. We allow the height of the bump to vary from data point to data point, and perhaps also the width. The value of $g(\mathbf{x})$ can then be obtained by summing the contributions due to each bump. If the bump value decreases as to zero, then further points have less influence than closer points. The height and width of each bump would reflect the importance of each point. The situation is illustrated in the following figure.



Its time to make these vague notions more concrete. We call the bump function the kernel function,

$$K(\mathbf{x}) = K(\|\mathbf{x}\|) \quad (1)$$

where it is usual to assume that the kernel function depends only on the norm of its argument. Thus once again, it is necessary to introduce a distance measure, and thus we assume that input normalization has been performed, as is usually a good idea anyway. The height of the bump can be varied by scaling by a parameter w , and the width of the bump can be varied by scaling inside the kernel function by a parameter γ , thus,

$$wK\left(\frac{\mathbf{x}}{\gamma}\right) \quad (2)$$

represents a bump of height w and width γ . The learning model is then defined by functions of the form

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^N w_i K_i\left(\frac{\mathbf{x} - \mathbf{x}_i}{\gamma_i}\right) = w_0 + \sum_{i=1}^N w_i K_i\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|}{\gamma_i}\right) \quad (3)$$

This formula is for regression, and for classification one finally takes the *sign*, or perhaps for learning purposes (as with the smoothening of threshold in the case of the perceptron), one can take the *tanh*. The threshold w_0 provides an offset parameter, which we will usually assume to be zero. The following things can be varied.

1. Shape of each bump. This can be varied by changing the width, γ_i , and the functional form of the bump kernel, K_i . We will usually assume that the shape of each bump is constant, i.e., we will take each γ_i to be a constant, γ , and the functional form of the kernel to be some fixed kernel function $K(\cdot)$.
2. The height of each bump.
3. The number of bumps. For the moment, we take the number of bumps to equal the number of data points.
4. Position of each bump. For the moment, a bump is centered on every data point.

Example (Gaussian Radial Basis Function): If we use a gaussian kernel, $K(\mathbf{x}) = e^{-\frac{1}{2}\|\mathbf{x}\|^2}$ with constant widths, the learning model has the form

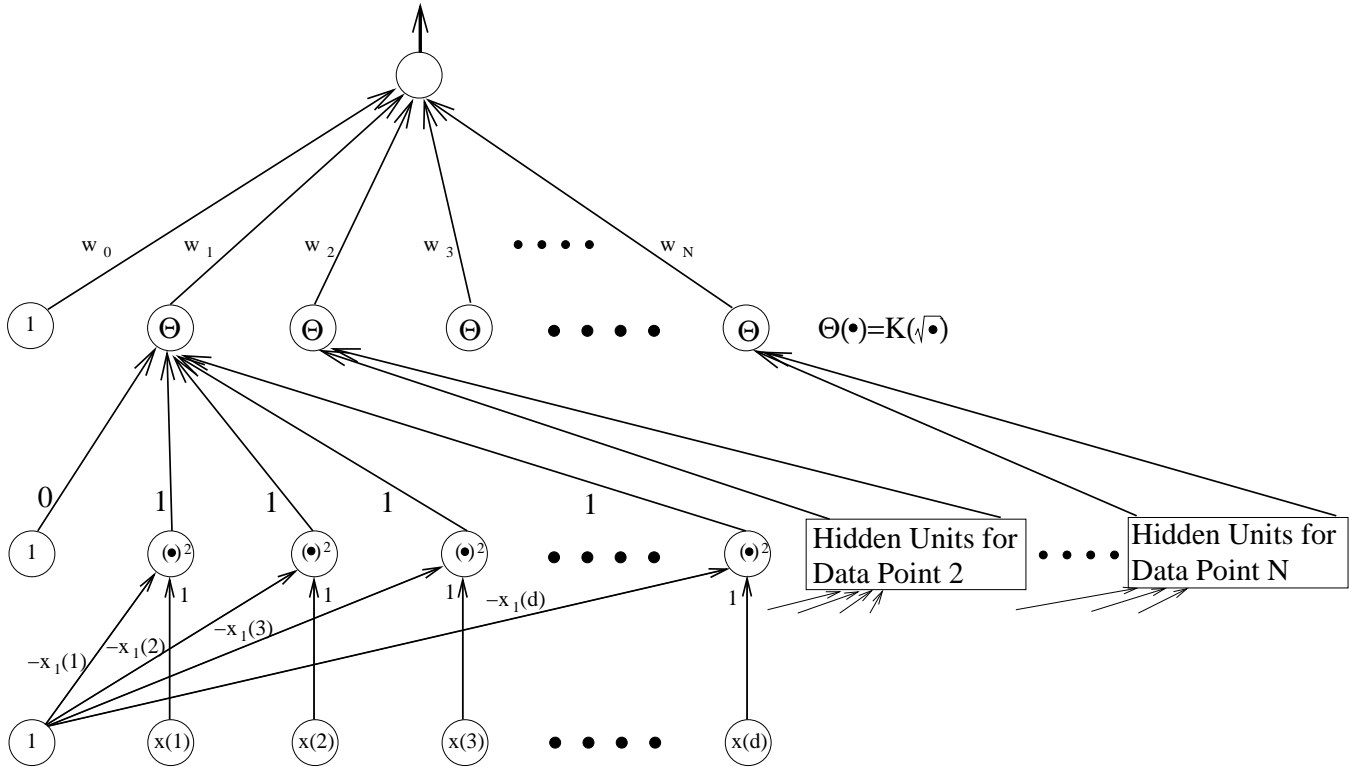
$$g(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\gamma^2}} \quad (4)$$

where it becomes clear that γ is related to the width.

Example (Representation of RBF's as Multilayer Perceptrons): Here we present an explicit representation of the RBF learning model, (3), using a multilayer perceptron. In fact a two hidden layer network will suffice. Even without this explicit representation, we know ahead of time that a similar kind of result should be possible, since the multilayer networks are universal approximators. Thus, any function that can be represented as an RBF could be arbitrarily closely approximated by a multilayer network. Hence, once one has one learning model that is a universal approximator, there seems no need to study other models. However, the reality is that certain functions can be represented much more compactly by one learning model than another. More generally, different learning models are tailored to different kinds of prior information, and hence each learning model has a set of problems for which it will be better than most others.

For the representation that we present, the number of hidden nodes needed in the first layer is Nd and in the second hidden layer is $N + 1$. All the weights in the input to hidden layer are fixed by the values of the data points, and the hidden to hidden layer weights are fixed to either one or zero. The only things that can be varied are the width of the kernel function K , which appears as the activation function in the second hidden layer, and the weights w_i in the output layer, which represent the heights of the bumps. Thus, if we can represent the RBF as a multilayer network with $\Theta(N)$ nodes. This is a huge and inefficient representation as multilayer network, so we rather pursue this kind of function as a *RBF*.

The situation with N data points in d dimensions is illustrated in the figure below. All connection weights that are not shown are zero.



The first hidden layer nodes serve to compute $\| \mathbf{x} - \mathbf{x}_i \|^2$ and these nodes can be divided into N groups of d nodes each. The output of one of these d nodes is $(x(\alpha) - x_i(\alpha))^2$, where α indexes the dimension. The purpose of each node in the next layer is to sum up these d squared differences from the group in the previous layer and send it through the kernel function, thus computing $K(\| \mathbf{x} - \mathbf{x}_i \|)$ for a single data point. The output node simply sums these up to produce the final output.

What remains is to discuss the determination of γ , the width of the kernels, and the heights of the kernels.

1 Target Value Height RBF's for Classification

For classification problems, one chooses the heights, w_i , equal to the target value for the data point i ($y_i = \pm 1$), i.e.,

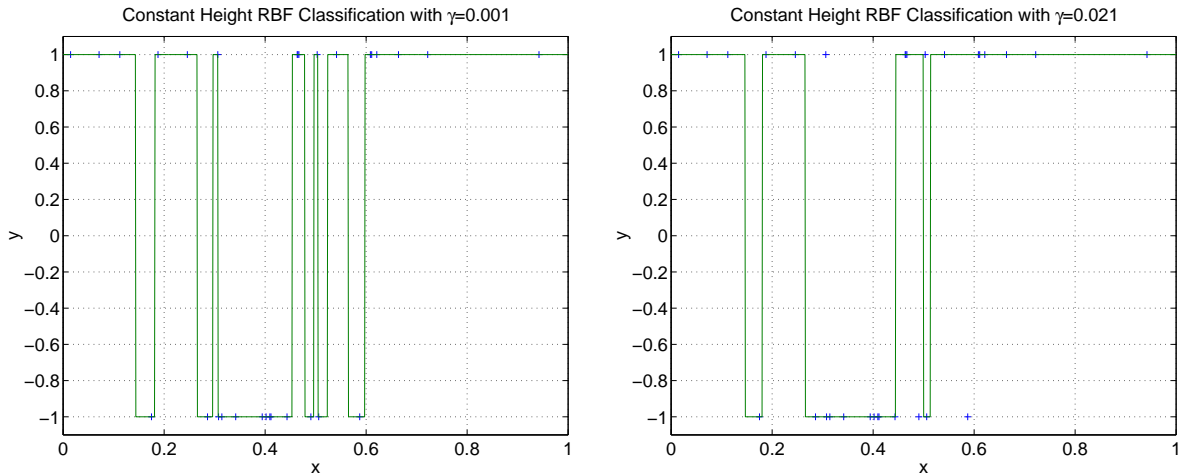
$$w_i = y_i \quad (5)$$

This choice for the weights results in the “learned” function

$$g(\mathbf{x}) = \text{sign} \left[\sum_{i=1}^N y_i K \left(\frac{\mathbf{x} - \mathbf{x}_i}{\gamma} \right) \right] \quad (6)$$

This learning model is very similar to the r -Near Neighbor rule. All that remains is to set γ , the width of the kernels. we can intuitively understand the effect of γ as follows. The larger γ , the

larger the width of the kernel. This means that points further away from \mathbf{x} will have an effect on the function value at \mathbf{x} , and this would correspond to increasing r in the r -Near Neighbor rule. Thus, smooth functions are represented by large values of γ , hence a large value of γ should be used when the prior on the target function is smooth and when there is a large amount of noise in the data set. A small value for gamma should be used when the target function is not expected to be smooth or when there is little noise in the data set. Ultimately, one could choose γ in a more principled fashion using cross validation. It is clear that in the limit $\gamma \rightarrow 0$, the data points are fit exactly and the model becomes non-falsifiable in this limit. Hence, one should choose $\gamma > 0$. The following figures provide an illustration of how the choice of γ can affect the classification function, using Gaussian kernels.



Since this classification RBF can be viewed as a generalization of the r -Near Neighbor rule, it is no surprise that a powerful theorem regarding the convergence of the test performance to the Bayes optimal can be obtained. Since $\gamma \sim r$, the conditions for the convergence of the empirical risk and the true risk to the Bayes optimal are that $\gamma \rightarrow 0$ and $\gamma^d N \rightarrow \infty$. The intuition is exactly the same as was presented for the r -Near Neighbor rule, and will not be discussed further.

2 Weighted Average RBF's for Regression

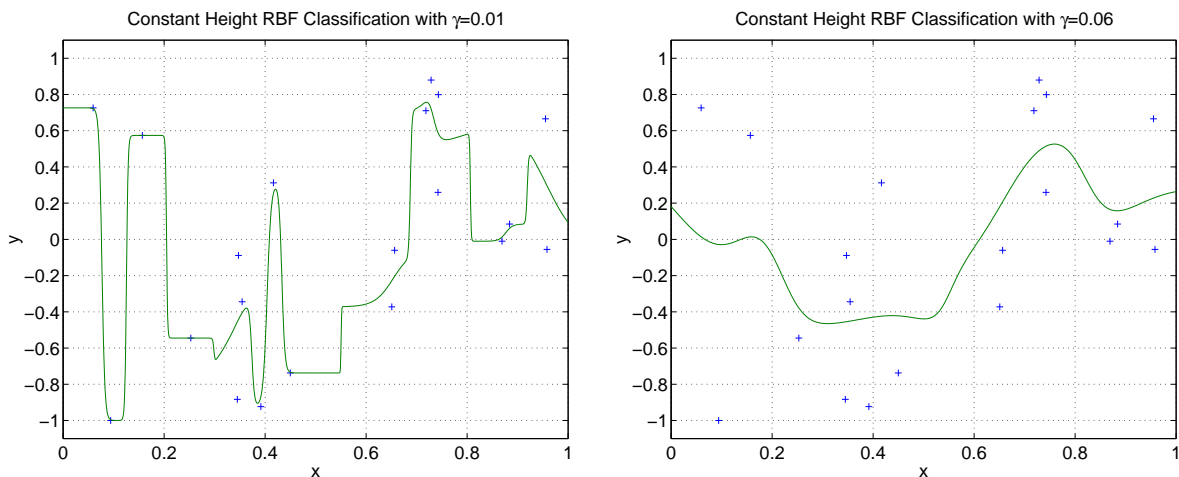
We would like each data point to contribute a percentage of its output to the output at \mathbf{x} . The percentage that it contributes should be proportional to the value of the bump due to that data point at \mathbf{x} . This leads to an \mathbf{x} -dependent choice for the heights, which can be viewed as a slight generalization of (3). The “learned” function is given by

$$g(\mathbf{x}) = \sum_{i=1}^N y_i \frac{K\left(\frac{\mathbf{x}-\mathbf{x}_i}{\gamma}\right)}{Z(\mathbf{x})} \quad (7)$$

where

$$Z(\mathbf{x}) = \sum_{j=1}^N K\left(\frac{\mathbf{x}-\mathbf{x}_j}{\gamma}\right) \quad (8)$$

(6) may also be viewed as (7) where one additionally takes the sign, as the positive number $Z(\mathbf{x})$ has no effect inside the sign function. The effect that the choice of γ has on the output function is similar to the classification case, and is illustrated in the following figures using Gaussian kernels.



Once again, one could choose γ using cross validation. What about the relationship between the risk and the Bayes optimal risk? For certain types of risks, for example the squared error, one can prove that the risk converges to the Bayes optimal risk provided that $\gamma \rightarrow 0$ and $\gamma^d N \rightarrow \infty$. The intuition is exactly the same as before.

3 Exact Interpolation for Classification or Regression

One can pick the heights, w_i , so as to fit the training data set exactly. If the data set is $D = \{\mathbf{x}_i, y_i\}$, then if one picks w_i so that

$$y_i = \sum_{i=1}^N w_i K\left(\frac{\mathbf{x} - \mathbf{x}_i}{\gamma}\right) \quad (9)$$

then the function output will equal the target output on the data set. Rewriting (9) in matrix form,

$$\mathbf{y} = \mathbf{K}\mathbf{w} \quad (10)$$

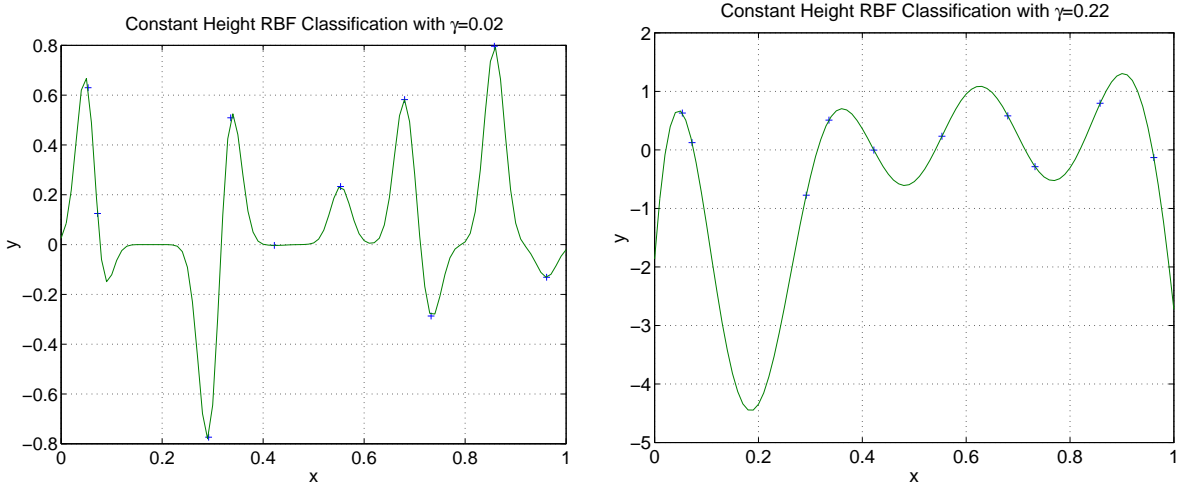
where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \quad (11)$$

where we assume that there is no threshold. \mathbf{K} is an $N \times N$ matrix whose components are given by

$$K_{ij} = K\left(\frac{\mathbf{x}_i - \mathbf{x}_j}{\gamma}\right) \quad (12)$$

For very general classes of kernels, \mathbf{K} will be invertible if the data points \mathbf{x}_i are distinct. This property is true, for example, for Gaussian kernels. The following figures illustrate how the choice of γ affects the regression function, using Gaussian kernels.



Once again, the recurring theme is that larger values for γ leads to smoother functions.

Unfortunately, no theorem can be claimed regarding the convergence of the risk of the output function to the Bayes optimal risk, and it is not even the case that the empirical risk will converge to the true risk, as is already clear because the model is not falsifiable. However, the learning is fast ($O(N^3)$ for the inversion of \mathbf{K}), and well defined.

4 RBF's with Fixed Number of Centers

We have discussed three methods for the determination of the heights, assuming that we placed a center at each data point. In many cases, this can lead to non-falsifiable models, even though the learning was fast. Another major problem with these techniques is that they require all the data to be stored for function evaluation which is an $O(Nd)$ memory requirement, and the computation of the output is then an $O(Nd)$ calculation. These are unreasonable requirements for a practical algorithm. One approach is to edit the data as was discussed with the nearest neighbor methods. Another more suitable approach is to not require a center at each data point. Rather we use M centers. The question now becomes where to place the centers. One obvious solution is to put the centers on the most important data points, but then one needs to discuss exactly how to determine the importance of a data point. For now, we allow the position of the centers to be anywhere, defined by the M vectors μ_1, \dots, μ_M . The learning model is then given by

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^M w_i K_i \left(\frac{\|\mathbf{x} - \mu_i\|}{\gamma_i} \right) \quad (13)$$

This learning model has $M(d+1)+2$ unspecified parameters, that need to be picked based upon the data. The memory requirement once the learning is done is $O(Md)$ and the computation time is also $O(Md)$ which are independent of the number of data points. The parameter γ is a smoothness parameter and so can be used in a regularization term. Thus one approach is now immediate. Construct an error function to optimize, for example,

$$E(\mathbf{w}, \mu_1, \dots, \mu_M, \gamma) = R_{emp}(\mathbf{w}, \mu_1, \dots, \mu_M, \gamma) + \frac{\lambda}{\gamma} \quad (14)$$

and optimize this error function with respect to all the parameters of the learning model. The regularization term encourages smoothness, and the regularization parameter, λ could be picked using a cross validation technique. All the optimization techniques that have already been discussed can now be brought to bear. This is certainly one approach. A second, much faster approach suggests itself by going back to the interpretation of the μ_i 's. They are the centers of the bumps, so they should be placed in positions that are representative of the data. This interpretation of the centers leads to the following algorithm.

1. Cluster the data into M clusters.
2. Place the M centers, μ_i at the “centers” of the clusters.
3. The μ_i are now fixed, and the optimization of (14) becomes an optimization problem in \mathbf{w} and γ . Assuming that the empirical risk is a squared error, the objective function, (14), can be written as follows

$$E(\mathbf{w}, \gamma) = \frac{1}{N}(\mathbf{y} - \mathbf{K}\mathbf{w})^T(\mathbf{y} - \mathbf{K}\mathbf{w}) + \frac{\lambda}{\gamma} \quad (15)$$

where the vectors \mathbf{y} and \mathbf{w} are given by (where we assume that the threshold \mathbf{w}_0 is set to 0.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \quad (16)$$

and $\mathbf{K}(\gamma)$ is an $N \times M$ matrix whose components are given by

$$K_{ij} = K\left(\frac{\mathbf{x}_i - \mu_j}{\gamma}\right) \quad (17)$$

Differentiating (15) with respect to \mathbf{w} and setting to zero to get the optimal \mathbf{w} given γ , we find that

$$\mathbf{w} = (\mathbf{K}^T\mathbf{K})^{-1}\mathbf{K}^T\mathbf{y} \quad (18)$$

and the objective function to minimize becomes

$$E(\gamma) = \frac{1}{N}\mathbf{y}^T(\mathbf{I} - \mathbf{K}(\mathbf{K}^T\mathbf{K})^{-1}\mathbf{K}^T)\mathbf{y} + \frac{\lambda}{\gamma} \quad (19)$$

which is now only a one dimensional minimization in γ which should be relatively easy to accomplish.

Alternatively, instead of minimizing with respect to γ , one could pick γ using cross validation. In either event, the main part of the algorithm (selecting the μ_i 's and the \mathbf{w}_i 's is fast as the \mathbf{w}_i are given in closed form once the μ_i 's are fixed and the μ_i 's are fixed by clustering.

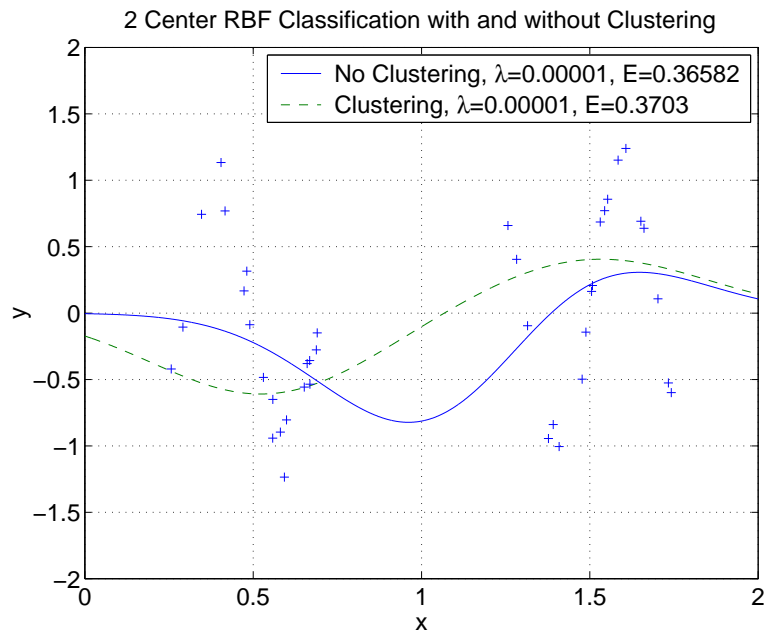
We have assumed that the threshold is 0, but it should be clear that this was not a necessary assumption and the same ideas carry through with non-zero threshold.

What remains is to discuss how to implement steps (1) and (2) in the algorithm, which takes us into the topic of clustering, a whole field unto itself. We will discuss this topic separately since it is useful for a variety of techniques (we have already met a need for it once in the branch and bound algorithm in the Nearest Neighbor method).

The RBF with a fixed number of centers has a variety of appealing features.

1. The training is fast (much faster than for neural networks, and this is largely due to the separation of the learning into a clustering approach to getting the centers, which are the variables controlling the non-linearity. Since the center positions
2. Un-labelled data can be used for obtaining the μ_i 's.
3. RBF's are rooted in a considerable amount of theoretical motivation.

The following figure compares a 2 center RBF on a data set when clustering is used to determine the μ_i 's versus a full optimization to a local minimum with respect to all the parameters.



As can be seen, though the classification functions are somewhat different (due to different μ_i 's, the errors achieved are comparable.

One thing that one has to be careful with is that irrelevant dimensions (dimensions in which the function value does not change, but there is dispersion in the data points) will have an unwarranted effect on the learning because the dispersion in the irrelevant dimensions can significantly affect the positioning of the centers, which can drastically and detrimentally affect the performance. Some of these issues largely disappear when one normalizes the inputs, but the concern remains that RBF networks as we have presented can be adversely affected if some of the input dimensions are irrelevant.