Radial Basis Functions: An Algebraic Approach (with Data Mining Applications)

Tutorial

Amrit L. Goel
Dept. of EECS
Syracuse University
Syracuse, NY 13244
goel@ecs.syr.edu

Miyoung Shin
ETRI
Daejon, Korea, 305-350
shinmy@etri.re.kr

Tutorial notes for presentation at ECML/PKDD 2004, Pisa, Italy, September 20-24, 2004

Abstract

Radial basis functions have now become a popular model for classification and prediction tasks. Most algorithms for their design, however, are basically iterative and lead to irreproducible results.
In this tutorial, we present an innovative new approach (*Shin-Goel* algorithm) for the design and evaluation of the RBF model. It is based on purely algebraic concepts and yields reproducible designs.
Use of this algorithm is demonstrated on some benchmark data sets, and data mining applications in software engineering and cancer class prediction are described.

Outline

1. Problems of classification and prediction
2. RBF model structure
3. Brief overview of RBF design methods
4. Algebraic algorithm of Shin and Goel
5. RBF center selection algorithm
6. Benchmark data classification modeling
7. Data mining and knowledge discovery applications
8. Summary

Problems of Classification and Prediction

Classification and Prediction

- Classification and prediction encompass a wide range of tasks of great practical significance in science and engineering, ranging from speech recognition to classifying sky objects. These are collectively called pattern recognition tasks. Humans are good at some of these, such as speech recognition, while machines are good at others, such as bar code reading.
- The discipline of building these machines is the domain of pattern recognition.
- Traditionally, statistical methods have been used for such tasks but recently neural nets are increasing employed since they can handle very large problems, and are less restrictive than statistical methods. Radial basis function is one such type of neural network.

Radial Basis Function

- RBF model is currently very popular for pattern recognition problems.
- RBF has nonlinear and linear components which can be treated separately. Also, RBF possesses significant mathematical properties of universal and best approximation. These features make RBF models attractive for many applications.
- Range of fields in which RBF model has been employed is very impressive and includes geophysics, signal processing, meteorology, orthopedics, computational fluid dynamics, and cancer classification.

Problem Definition

- The pattern recognition task is to construct a model that captures an unknown input-output mapping on the basis of limited evidence about its nature. The evidence is called the *training sample*. We wish to construct the “best” model that is as close as possible to the true but unknown mapping function. This process is called *training or modeling*.
- The training process seeks model parameters that provide a good fit to the training data and also provide good predictions on future data.

Problem Definition (cont.)

- Formally, we are given data set

$$D = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in R^d, y_i, i = 1,..., n\} \qquad ,$$

in which both inputs and their corresponding outputs are made available and the outputs (y_i) are continuous or discrete values.

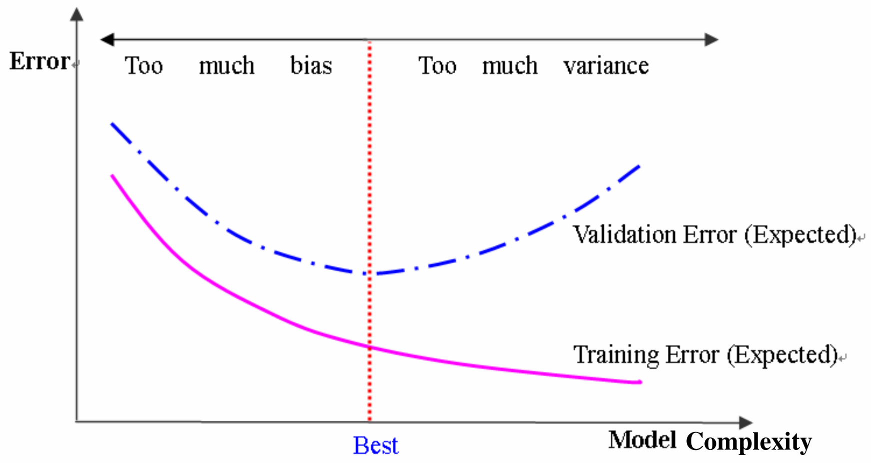
- Problem is to find a mapping function from the *d*-dimensional input space to the 1-dimensional output space based on the data.

Modeling Issues

- The objective of training or modeling is to determine model parameters so as to minimize the squared estimation error that can be decomposed into bias squared and variance. However, both cannot be simultaneously minimized. Therefore, we seek parameter values that give the best compromise between small bias and small variance.
- In practice, the bias squared and the variance cannot be computed because the computation requires knowledge of the true but unknown function. However, their trend can be analyzed from the shapes of the training and validation error curves.

Modeling Issues (cont.)

- Idealized relationship of these errors is shown below. Here we see the conceptual relationship between the expected training and validation errors, the so-called *bias-variance dilemma*.



Modeling Issues (cont.)

- Here, training error decreases with increasing model complexity; validation error decreases with model complexity up to a certain point and then begins to increase.
- We seek a model that is neither too simple nor too complex. A model that is too simple will suffer from underfitting because it does not learn enough from the data and hence provides a poor fit. On the other hand, a model that is too complicated would learn details including noise and thus suffers from overfitting. It cannot provide good generalization on unseen data.
- In summary, we seek a model that is
  - Not too simple: underfitting; not learn enough
  - Not too complicated: overfitting; not generalize well

RBF Model Structure

Function Approximation

- Suppose $D = \{(\mathbf{x}_i, y_i): \mathbf{x}_i \in R^{\mathrm{d}}, y_i \in R, i = 1, \ldots, n\}$ where the underlying true but unknown function is f^0.
- Then, for given D, how to find a "best" approximating function f^* for f^0?
  - Function approximation problem
- In practice, F, a certain class of functions, is assumed.
  - Approximation problem is to find a best approximation for f^0 from F.
  - An approximating function f^* is called a best approximation from $F = \{f^1, f^2, \ldots, f^p\}$ if f^* satisfies the following condition:

$$\|f^* - f^0\| \le \|f^j - f^0\|, j = 1, \ldots, p$$

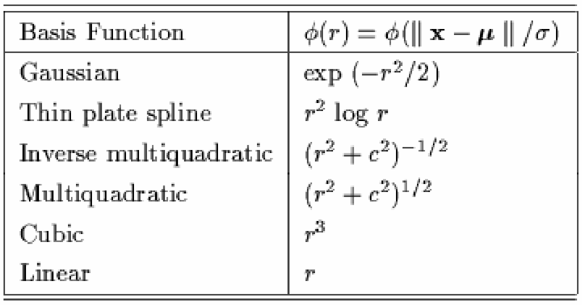
RBF Model for Function Approximation

- Assume
  - F is a class of RBF models
  - $f^* \in F$
- Why RBF?
  - Mathematical properties
    - Universal approximation property
    - Best approximation property
  - Fast learning ability due to separation of nonlinearity and linearity during training phase (model development).

RBF Model

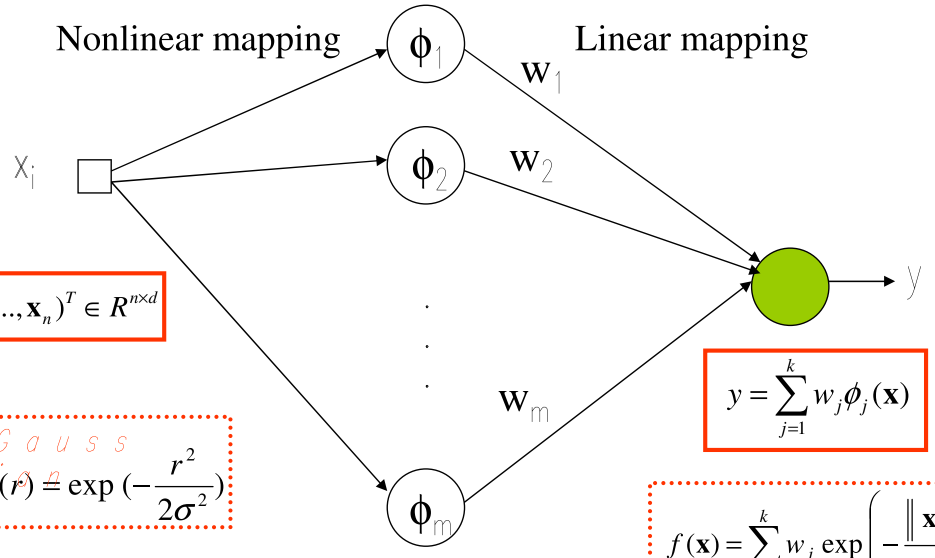
$$\hat{y}_i = f(\mathbf{x}_i) = \sum_{j=1}^{m} \mathrm{w}_j \phi(\mathbf{x}_i) = \sum_{j=1}^{m} \phi\left(\left\|\mathbf{x}_i - \mu_j\right\| / \sigma_j\right)$$

- Here
  - $\phi(\cdot)$ is a basis function
  - w_{i} : weight
  - μ_i : center
  - σ_i : width of basis function
  - m : number of basis functions
- Choices of basis function



Basis Function	$\phi(r) = \phi(\\| \mathbf{x} - \boldsymbol{\mu} \\| / \sigma)$
Gaussian	$\exp\ (-r^2/2)$
Thin plate spline	$r^2 \log r$
Inverse multiquadratic	$(r^2 + c^2)^{-1/2}$
Multiquadratic	$(r^2 + c^2)^{1/2}$
Cubic	r^3
Linear	r

Radial Basis Function Network



$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)^T \in R^{n \times d}$

Gaussian

$\phi(r) = \exp\ (-\frac{r^2}{2\sigma^2})$

$\phi_j(\mathbf{x}) = \phi(\left\| \mathbf{x} - \boldsymbol{\mu}_j \right\|),\ j = 1,2,...,m$

$y = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x})$

$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \exp\left(-\frac{\left\| \mathbf{x} - \boldsymbol{\mu}_j \right\|^2}{2\sigma_j^2}\right)$

Input layer

Hidden layer of *m* radial basis functions

Output layer

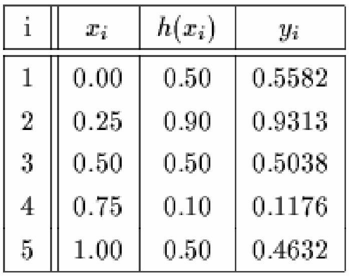
RBF Interpolation:
Sine Example

SINE EXAMPLE

- Consider sine function (Bishop, 1995) and its interpolation

$$h(x) = 0.5 + 0.4 sin(2\pi x).$$

- Compute five values of $h(x)$ at equal intervals of x in (0, 1), add random noise from normal with mean = 0, variance = 0.25



i	x_i	$h(x_i)$	y_i
1	0.00	0.50	0.5582
2	0.25	0.90	0.9313
3	0.50	0.50	0.5038
4	0.75	0.10	0.1176
5	1.00	0.50	0.4632

- Interpolation problem: Determine Gaussian RBF $f(x_i)$ such that

$$f(x_i) = y_i, \quad i = 1, 2, \ldots, 5.$$

SINE EXAMPLE (cont.)

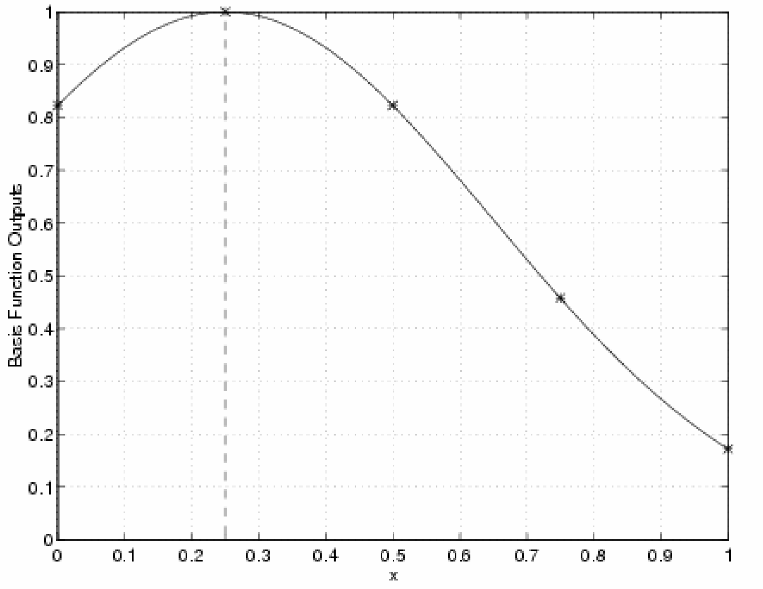
- Construct interpolation matrix with five basis functions centered at *x*'s (assume $\sigma = 0.4$) and compute G:

$$\mathrm{G} = \begin{array}{c} \begin{matrix} \quad\boldsymbol{g}_1 & \quad\boldsymbol{g}_2 & \quad\boldsymbol{g}_3 & \quad\boldsymbol{g}_4 & \quad\boldsymbol{g}_5 \end{matrix} \\ \begin{bmatrix} 1.0000 & 0.8226 & 0.4578 & 0.1724 & 0.0439 \\ 0.8226 & 1.0000 & 0.8226 & 0.4578 & 0.1724 \\ 0.4578 & 0.8226 & 1.0000 & 0.8226 & 0.4578 \\ 0.1724 & 0.4578 & 0.8226 & 1.0000 & 0.8226 \\ 0.0439 & 0.1724 & 0.4578 & 0.8226 & 1.0000 \end{bmatrix} \end{array}$$

- In above, e.g., g_2 is obtained as:

$$\boldsymbol{g}_2 = \begin{bmatrix} \phi_2(x_1) \\ \phi_2(x_2) \\ \phi_2(x_3) \\ \phi_2(x_4) \\ \phi_2(x_5) \end{bmatrix} = \begin{bmatrix} \exp\left(-\frac{\|x_1-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|x_2-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|x_3-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|x_4-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|x_5-0.25\|^2}{2(0.4)^2}\right) \end{bmatrix} = \begin{pmatrix} 0.8226 \\ 1.0000 \\ 0.8226 \\ 0.4578 \\ 0.1724 \end{pmatrix}.$$

SINE EXAMPLE (cont.)



Basis function centered at $\mu_2 = 0.25$

SINE EXAMPLE (cont.)

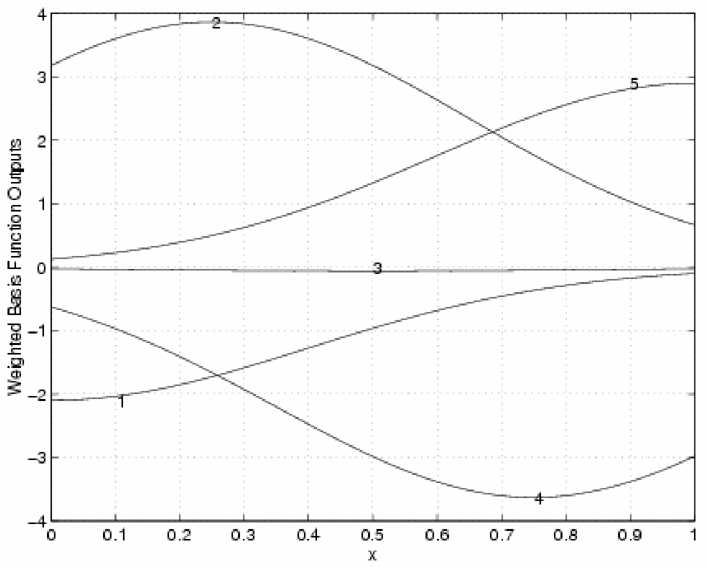
- The weights are computed from G and y_i and we get

$$\hat{\mathbf{y}} = \mathrm{G}\mathbf{w}.$$

$$\begin{aligned}\hat{y}_i \;=\; & -2.0978 \exp\left(-\frac{\| x_i - 0.0 \|^2}{2(0.4)^2}\right) + 3.8689 \exp\left(-\frac{\| x_i - 0.25 \|^2}{2(0.4)^2}\right) \\ & -0.0605 \exp\left(-\frac{\| x_i - 0.50 \|^2}{2(0.4)^2}\right) - 3.6329 \exp\left(-\frac{\| x_i - 0.75 \|^2}{2(0.4)^2}\right) \\ & +2.9063 \exp\left(-\frac{\| x_i - 1.0 \|^2}{2(0.4)^2}\right)\end{aligned}$$

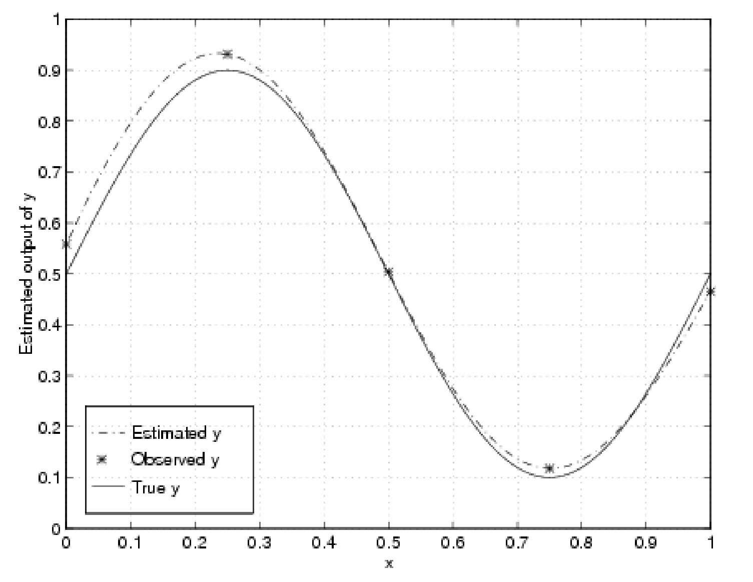
- Each term is a weighted basis function

SINE EXAMPLE (cont.)



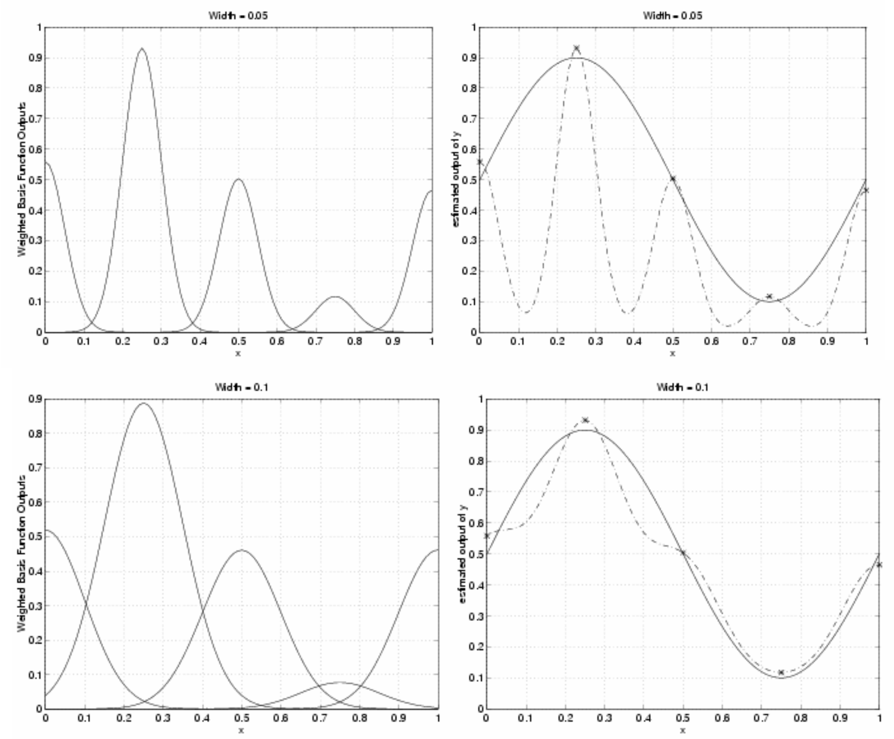
Plots of weighted basis functions outputs

SINE EXAMPLE (cont.)

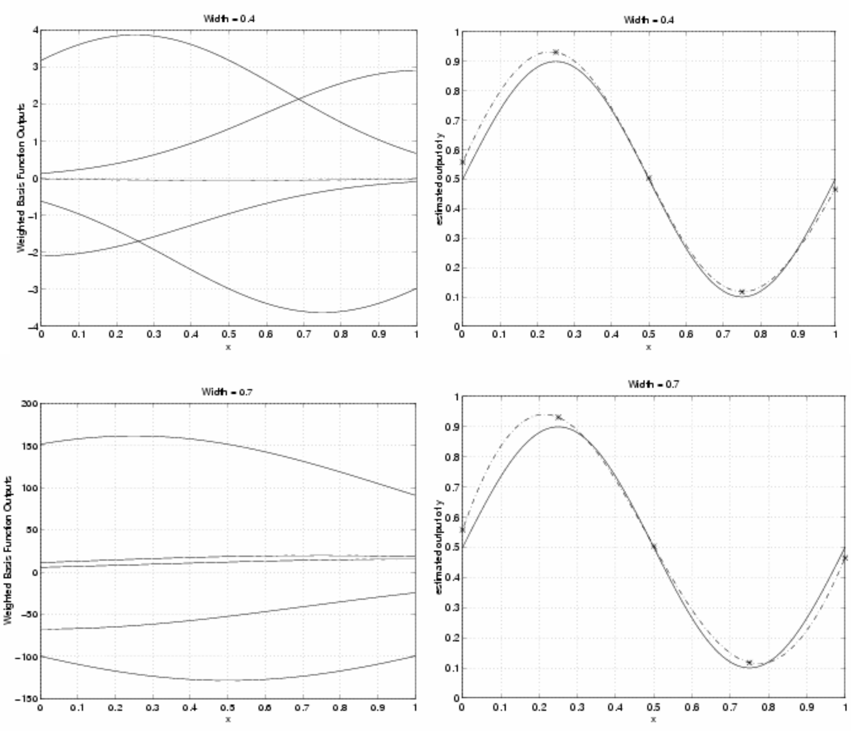


Plots of true, observed and estimated values by RBF model

SINE EXAMPLE (cont.)



SINE EXAMPLE (cont.)



Brief Overview of RBF Design Methods

Brief Overview of RBF Design

- Model Parameters P = ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\mathbf{w}$, m) where

  $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_m]$

  $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_m]$

  $\boldsymbol{w} = [w_1, w_2, \ldots, w_m]$
- Design problem of RBF model
  - How to determine ***P***?
- Some design approaches
  - Clustering
  - Subset selection
  - Regularization

Clustering

- Assume some value *k*, the number of basis functions is given
- Construct *k* clusters with randomly selected initial centers
- The parameters are taken to be

  μ_j : *j*th cluster center

  σ_j : average distance of each cluster to *P*-nearest clusters or individual distances

  w_j : weight
- Because of randomness in training phase, the design suffers from inconsistency

Subset Selection

- Assume some value of σ

  μ_j : a subset of *j* input vectors that most contribute to output variance

  m : number of basis functions that provides output variance enough to cover a prespecified threshold value

  w_j : weight

Regularization

m : data size, i.e., number of input vectors

μ_j : input vectors (x_i)

w_j : least squares method with regularized term

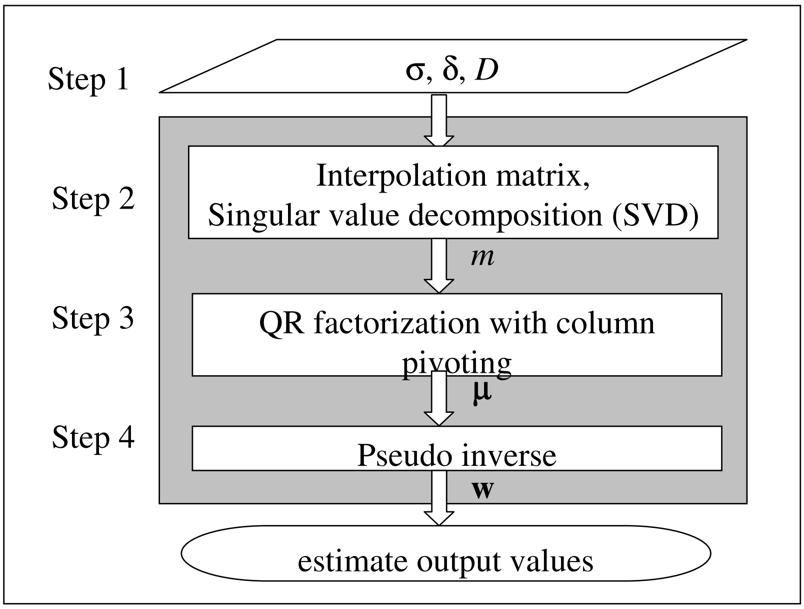
- Regularization parameter (λ) controls the smoothness and the degree of fit
- Computationally demanding

Algebraic Algorithm of Shin and Goel

Our Objective

- Derive a mathematical framework for design and evaluation of RBF model
- Develop an objective and systematic design methodology based on this mathematical framework

Four Step RBF Modeling Process of SG Algorithm



SG algorithm is a *learning* or *training* algorithm to determine the values for the number of basis functions (*m*), their centers (μ), widths (*σ*) and weights (**w**) to the output layer on the basis of the data set

Design Methodology

- $m = rank\left(G, s_1 \times \frac{\delta}{100}\right)$ where
  - G : Gaussian interpolation matrix
  - s_1 : first singular value of G
  - δ : 100(1 - δ)% RC of G
- μ : a subset of input vectors
  - Which provides a good compromise between structural stabilization and residual minimization
  - By QR factorization with column pivoting
- w : Φ^{+}y
  - Where Φ^{+} is pseudo-inverse of design matrix Φ

RBF Model Structure

- For $D = \{(\mathbf{x}_i, y_i): \mathbf{x}_i \in R^{\mathrm{d}}, y_i \in R\}$
  - input layer: $n \times d$ input matrix
  - hidden layer: $n \times m$ design matrix
  - output layer: $n \times 1$ output vector

$$\overset{X}{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}} \Rightarrow \overset{\Phi}{\begin{bmatrix} \phi_1(x_1) & \cdots & \phi_m(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \cdots & \phi_m(x_n) \end{bmatrix}} \overset{w}{\begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}} \Rightarrow \begin{bmatrix} \phi_1(x_1)w_1 & +\cdots+ & \phi_m(x_1)w_m \\ & \vdots & \\ \phi_1(x_n)w_1 & & \phi_m(x_n)w_m \end{bmatrix} = \overset{Y}{\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}}$$

- Φ is called design matrix
- For, $\phi_j(\mathbf{x}_i) = \phi(\|\mathbf{x}_i - \mu_j\| / \sigma_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$
  - If $m = n$ and $\mu_j = \mathbf{x}_j$, $j = 1, \ldots, n$ then, Φ is called *interpolation matrix*
  - If $m << n$, Design Matrix

Basic Matrix Properties

- Subspace spanned by a matrix
  - Given a matrix $A = [\mathrm{a}_1 \ \mathrm{a}_2 \ \ldots \ \mathrm{a}_n] \in R^{m \times n}$, the set of all linear combinations of these vectors builds the subspace $\boldsymbol{A}$ of R^n, i.e.,

    $$\boldsymbol{A} = \mathrm{span}\{\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_n\} = \{\sum_{j=1}^{n} c_j a_j : c_j \in R\}$$
  - Subspace $\boldsymbol{A}$ is said to be spanned by the matrix A
- Dimension of subspace
  - Let $\boldsymbol{A}$ be the subspace spanned by A. If $\exists$ independent basis vectors $b_1, b_2, .., b_{\mathrm{k}} \in \boldsymbol{A}$ such that

    $$\boldsymbol{A} = \mathrm{span}\{b_1, b_2, .., b_{\mathrm{k}}\}$$
  - Then the dimension of the subspace $\boldsymbol{A}$ is k, i.e., $\dim(\boldsymbol{A}) = k$

Basic Matrix Properties (cont.)

- Rank of a matrix
  - Let $A \in R^{m\times n}$ and $\boldsymbol{A}$ be the subspace spanned by the matrix A.Then, rank of A is defined by the dimension of $\boldsymbol{A}$, the subspace spanned by A. In other words,

$$\text{rank}(A) = \dim(\boldsymbol{A})$$

- Rank deficiency
  - A matrix $A \in R^{m\times n}$ is *rank-deficient* if rank(A) < min{m, n}
  - Implies that
    - ∃ some redundancy among its column or row vectors

Characterization of Interpolation Matrix

- Let $G = [g_1, g_2, \ldots, g_n] \in R^{n\times n}$ be an interpolation matrix.
  - Rank of G = dimension of its column space
  - If column vectors are linearly independent,
    - Rank(G) = number of column vectors
  - If column vectors are linearly dependent,
    - Rank(G) < number of column vectors
- Rank deficiency of G
  - It becomes rank-deficient if rank(G) < n
  - It happens
    - When two basis function outputs are collinear to each other,

      i.e., if two or more input vectors are very close to each other, then the outputs of the basis functions centered at those input vectors would be collinear

Characterization of Interpolation Matrix (cont.)

  - In such a situation, we do not need all the column vectors to represent the subspace spanned by G
  - Any one of those collinear vectors can be computed from other vectors
- In summary, if G is rank-deficient, it implies that
  - the intrinsic dimensionality of $G <$ number of columns (n)
  - the subspace spanned by G can be described by a smaller number ($m < n$) of independent column vectors

Rank Estimation Based on SVD

- The most popular rank estimation technique for dealing with large matrices in practical applications is Singular Value Decomposition (Golub, 1996)
  - If G is a real $n \times n$ matrix, then $\exists$ orthogonal matrices

    $U \in [u_1, u_2, \ldots, u_n] \in R^{n \times n}$, $V \in [v_1, v_2, \ldots, v_n] \in R^{n \times n}$*, such that*

    $U^T GV = \mathrm{diag}(s_1, s_2, \ldots, s_n) = S \in R^{n \times n}$

    where $s_1 \geq s_2 \geq \ldots \geq s_n \geq 0$
  - s_i : ith singular value
  - u_i : ith left singular vector
  - v_i : ith right singular vector
  - If we define r by $s_1 \geq \ldots \geq s_r \geq s_{r+1} = \ldots = s_n = 0$, then

$$\mathrm{rank}(G) = r \text{ and } \quad G = \sum_{i=1}^{r} s_i u_i v_i^T$$

Rank Estimation Based on SVD (cont.)

- In practice, data tend to be noisy
  - Interpolation matrix *G* generated from data is also noisy
  - Thus, the computed singular values from *G* are noisy and real rank of *G* should be estimated
- It is suggested to use effective rank(ε-rank) of G
- Effective rank r_ε = rank(*G*, ε), for $\varepsilon > 0$ such that

  $$s_1 \geq s_2 \geq \ldots \geq \varepsilon \geq \ \ldots \geq s_n$$
- How to determine ε?
  - We introduce RC (Representational Capability)

Representational Capability (RC)

- Definition : RC of G_m
  - Let *G* be an interpolation matrix of size $n \times n$, and SVD of *G* be given as above. If $m \leq n$ and $G_m = \sum_{i=1}^{m} s_i u_i v_i^T$, then RC of G_m is given by:

    $$\mathrm{RC}(G_m) = 1 - \frac{\|G - G_m\|_2}{\|G\|_2}$$
- Properties of RC
  - Corollary 1: Let SVD of *G* = diag(s_1, s_2, …, s_n) and $G_m = \sum_{i=1}^{m} s_i u_i v_i^T$ Then, for $m < n$

    $$\mathrm{RC}(G_m) = 1 - \frac{s_{m+1}}{s_1}$$
  - Corollary 2: Let r = rank(*G*) for $G \in R^{n\times n}$. If $m < r$, RC(G_m) < 1. Otherwise, RC(G_m) = 1

Determination of *m* based on RC Criterion

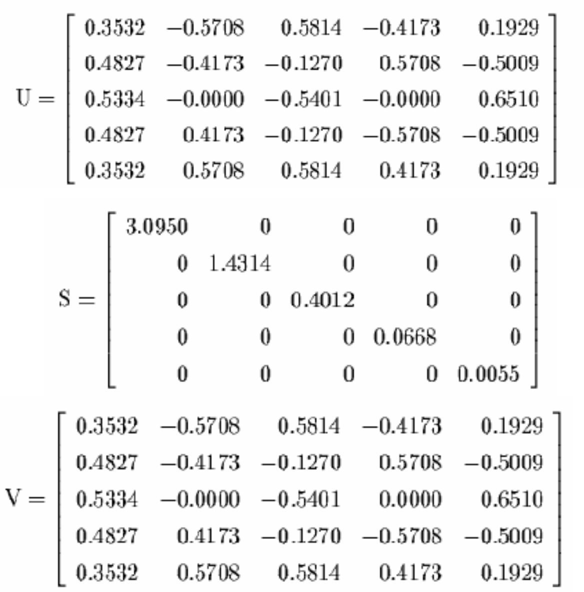
- For an interpolation matrix $G \in R^{n \times n}$, the number of basis functions which provides 100(1 - δ)% RC of G is given as

$$m = \text{rank}\left(G, s_1 \times \frac{\delta}{100} \right)$$

SVD and *m*: Sine Example

Singular Value Decomposition (SVD)

- SVD of the interpolation matrix produces three matrices, U, S, and V ($\sigma = 0.4$)



$$U = \begin{bmatrix} 0.3532 & -0.5708 & 0.5814 & -0.4173 & 0.1929 \\ 0.4827 & -0.4173 & -0.1270 & 0.5708 & -0.5009 \\ 0.5334 & -0.0000 & -0.5401 & -0.0000 & 0.6510 \\ 0.4827 & 0.4173 & -0.1270 & -0.5708 & -0.5009 \\ 0.3532 & 0.5708 & 0.5814 & 0.4173 & 0.1929 \end{bmatrix}$$

$$S = \begin{bmatrix} 3.0950 & 0 & 0 & 0 & 0 \\ 0 & 1.4314 & 0 & 0 & 0 \\ 0 & 0 & 0.4012 & 0 & 0 \\ 0 & 0 & 0 & 0.0668 & 0 \\ 0 & 0 & 0 & 0 & 0.0055 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.3532 & -0.5708 & 0.5814 & -0.4173 & 0.1929 \\ 0.4827 & -0.4173 & -0.1270 & 0.5708 & -0.5009 \\ 0.5334 & -0.0000 & -0.5401 & 0.0000 & 0.6510 \\ 0.4827 & 0.4173 & -0.1270 & -0.5708 & -0.5009 \\ 0.3532 & 0.5708 & 0.5814 & 0.4173 & 0.1929 \end{bmatrix}$$

Singular Value Decomposition (SVD) (cont.)

- Effective rank of G is obtained for several σ values

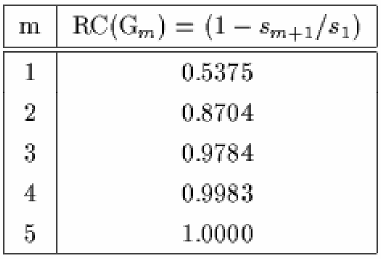
Width	Singular Values					Effective Rank r_ε	
(σ)	s_1	s_2	s_3	s_4	s_5	($\varepsilon = 0.01$)	($\varepsilon = 0.001$)
0.05	1.0	1.0	1.0	1.0	1.0	5	5
0.20	1.85	1.44	0.94	0.52	0.26	5	5
0.40	3.10	1.43	0.40	0.0067	0.0006	4	5
0.70	4.05	0.86	0.08	0.004	0.0001	3	4
1.00	4.47	0.51	0.02	0.0005	0.0000	3	3

RC of the Matrix G$_m$

$$G_m = \sum_{i=1}^{m} s_i \mathbf{u}_i \mathbf{v}_i^T.$$

$$\mathrm{RC}(G_m) = 1 - \frac{s_{m+1}}{s_1}.$$

- Consider σ = 0.4; then for *m* = 1, 2, 3, 4, 5, the RC is



m	$\mathrm{RC}(G_m) = (1 - s_{m+1}/s_1)$
1	0.5375
2	0.8704
3	0.9784
4	0.9983
5	1.0000

RC of the Matrix G$_m$ (cont.)

- Determine m for RC ≥ 80% or $\delta \leq$ 20%

$$s_{m+1} \leq s_1 \times \frac{\delta}{100}.$$

$$\begin{aligned} s_{m+1} &\leq (3.095) \times (0.20), \\ s_{m+1} &\leq 0.619. \end{aligned}$$

$s_3 = s_{2+1} = 0.4012 < 0.618$, so we get $m = 2$.

RBF Center Selection Algorithm

Center Selection Algorithm

- Given an interpolation matrix and the number of designed basis functions m, two questions are
  - Which columns should be chosen as the column vectors of the design matrix?
  - What criteria should be used?
- We use compromise between
  - Residual minimization for better approximation
  - Structural stabilization for better generalization

Center Selection Algorithm (cont.)

1. Compute the SVD of G to obtain matrices U, S, and V.

2. Partition matrix V and apply the QR factorization with column pivoting to $[\mathrm{V}_{11}^T \mathrm{V}_{21}^T]$ and obtain a permutation matrix P as follows:

$$\mathrm{V} \;=\; \begin{array}{cc} \left[\begin{array}{cc} \mathrm{V}_{11} & \mathrm{V}_{12} \\ \mathrm{V}_{21} & \mathrm{V}_{22} \end{array}\right] & \begin{array}{c} n \\ m \end{array} \\ \begin{array}{cc} m & n-m \end{array} & \end{array} .$$

$$\mathrm{Q}^T [\mathrm{V}_{11}^{\mathrm{T}}\,\mathrm{V}_{21}^{\mathrm{T}}]\mathrm{P} = [\mathrm{R}_{11}\,\mathrm{R}_{12}].$$

Center Selection Algorithm (cont.)

3. Compute GP and obtain the design matrix Φ by

$$\mathrm{GP} \;=\; \begin{array}{cc} [\Phi \quad \Phi'] & m. \\ \begin{array}{cc} m & n-m \end{array} & \end{array}$$

4. Compute $\mathrm{X}^{\mathrm{T}}\mathrm{P} = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \ldots, \mathbf{x}_{k_n})$ and determine *m* centers as

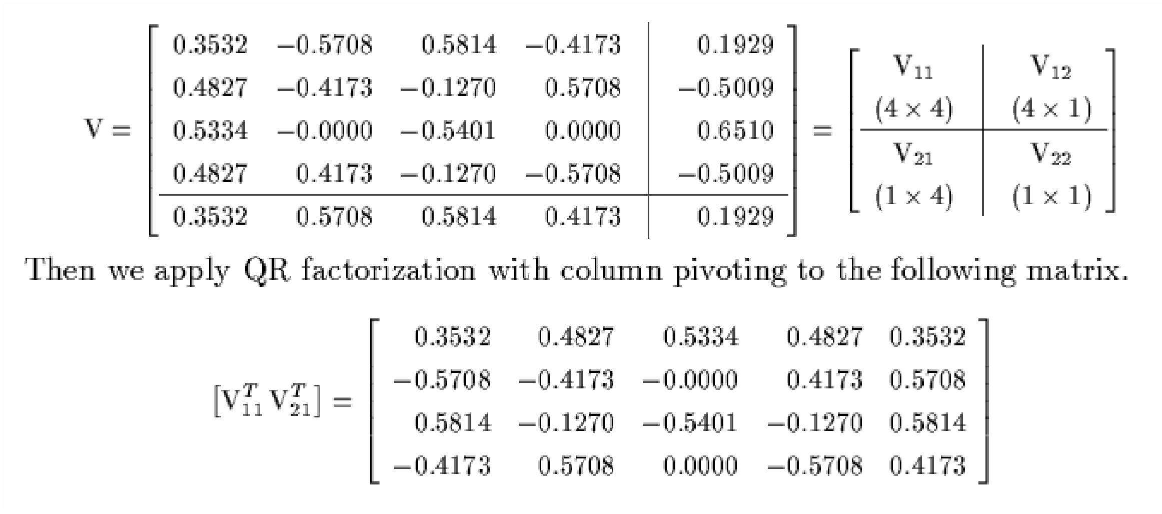
$$(\mu_1, \mu_2, \ldots, \mu_m) \equiv (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \ldots, \mathbf{x}_{k_m}).$$

Center Selection:
Sine Example

SG Center Selection Algorithm

Step 1: Compute the SVD of G and obtain matrices U, S, and V.

Step 2: Partition V as follows: ($\sigma = 0.4$)



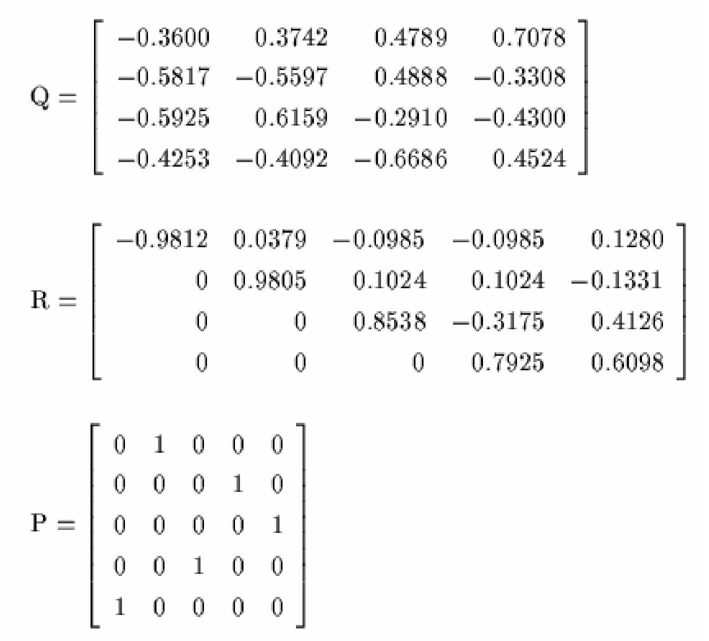
$$\mathrm{V} = \left[\begin{array}{cccc|c} 0.3532 & -0.5708 & 0.5814 & -0.4173 & 0.1929 \\ 0.4827 & -0.4173 & -0.1270 & 0.5708 & -0.5009 \\ 0.5334 & -0.0000 & -0.5401 & 0.0000 & 0.6510 \\ 0.4827 & 0.4173 & -0.1270 & -0.5708 & -0.5009 \\ \hline 0.3532 & 0.5708 & 0.5814 & 0.4173 & 0.1929 \end{array}\right] = \left[\begin{array}{c|c} \mathrm{V}_{11} & \mathrm{V}_{12} \\ (4 \times 4) & (4 \times 1) \\ \hline \mathrm{V}_{21} & \mathrm{V}_{22} \\ (1 \times 4) & (1 \times 1) \end{array}\right]$$

Then we apply QR factorization with column pivoting to the following matrix.

$$[\mathrm{V}_{11}^T \, \mathrm{V}_{21}^T] = \begin{bmatrix} 0.3532 & 0.4827 & 0.5334 & 0.4827 & 0.3532 \\ -0.5708 & -0.4173 & -0.0000 & 0.4173 & 0.5708 \\ 0.5814 & -0.1270 & -0.5401 & -0.1270 & 0.5814 \\ -0.4173 & 0.5708 & 0.0000 & -0.5708 & 0.4173 \end{bmatrix}$$

SG Center Selection Algorithm (cont.)

This results in Q, R, and P.



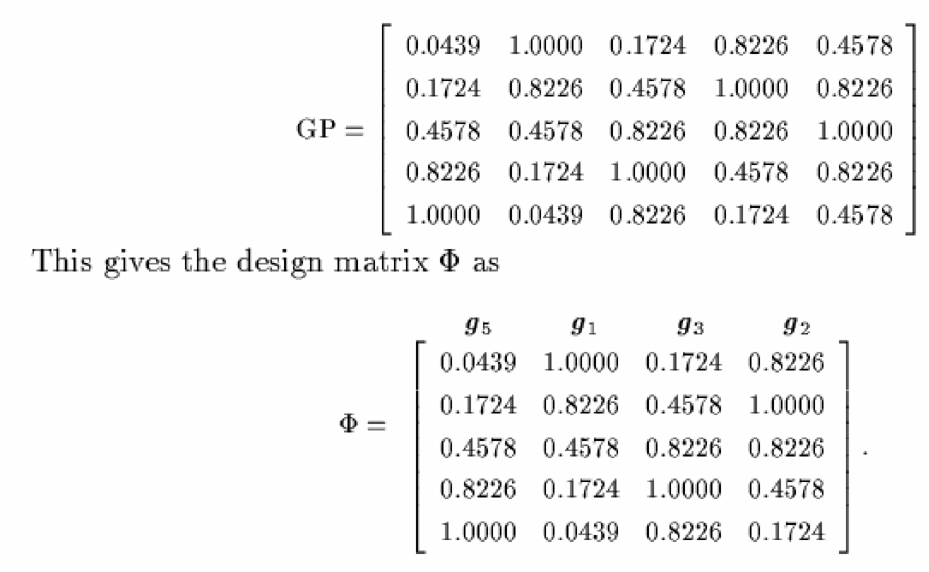
$$Q = \begin{bmatrix} -0.3600 & 0.3742 & 0.4789 & 0.7078 \\ -0.5817 & -0.5597 & 0.4888 & -0.3308 \\ -0.5925 & 0.6159 & -0.2910 & -0.4300 \\ -0.4253 & -0.4092 & -0.6686 & 0.4524 \end{bmatrix}$$

$$R = \begin{bmatrix} -0.9812 & 0.0379 & -0.0985 & -0.0985 & 0.1280 \\ 0 & 0.9805 & 0.1024 & 0.1024 & -0.1331 \\ 0 & 0 & 0.8538 & -0.3175 & 0.4126 \\ 0 & 0 & 0 & 0.7925 & 0.6098 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

SG Center Selection Algorithm (cont.)

Step 3: Compute GP.



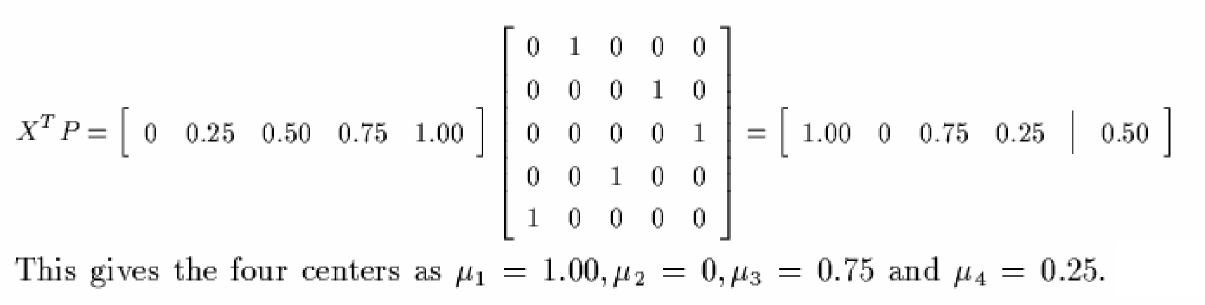
$$GP = \begin{bmatrix} 0.0439 & 1.0000 & 0.1724 & 0.8226 & 0.4578 \\ 0.1724 & 0.8226 & 0.4578 & 1.0000 & 0.8226 \\ 0.4578 & 0.4578 & 0.8226 & 0.8226 & 1.0000 \\ 0.8226 & 0.1724 & 1.0000 & 0.4578 & 0.8226 \\ 1.0000 & 0.0439 & 0.8226 & 0.1724 & 0.4578 \end{bmatrix}$$

This gives the design matrix Φ as

$$\Phi = \begin{array}{c} \begin{matrix} \boldsymbol{g}_5 & \boldsymbol{g}_1 & \boldsymbol{g}_3 & \boldsymbol{g}_2 \end{matrix} \\ \begin{bmatrix} 0.0439 & 1.0000 & 0.1724 & 0.8226 \\ 0.1724 & 0.8226 & 0.4578 & 1.0000 \\ 0.4578 & 0.4578 & 0.8226 & 0.8226 \\ 0.8226 & 0.1724 & 1.0000 & 0.4578 \\ 1.0000 & 0.0439 & 0.8226 & 0.1724 \end{bmatrix} \end{array}.$$

SG Center Selection Algorithm (cont.)

Step 4: Compute $X^{\mathrm{T}}P$ and determine m = 4 centers as the first four elements in $X^{\mathrm{T}}P$.

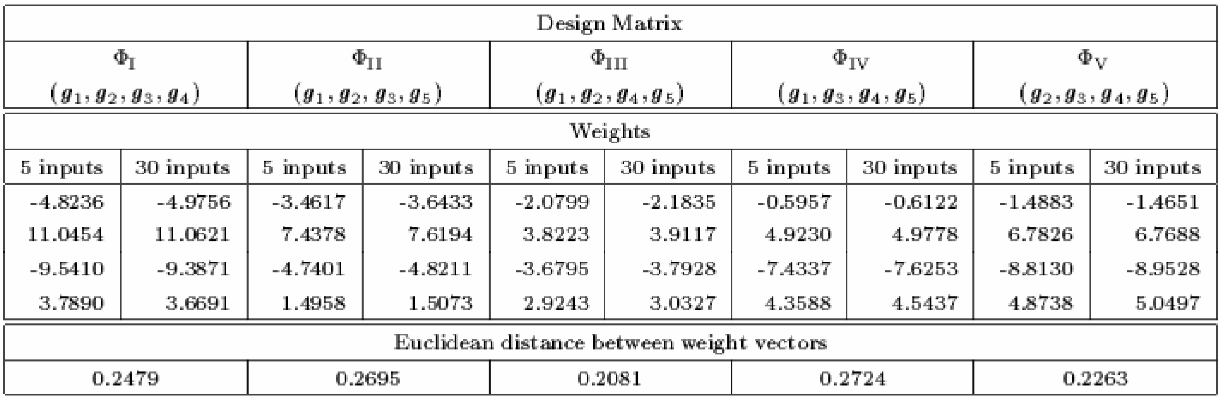


$$X^T P = \begin{bmatrix} 0 & 0.25 & 0.50 & 0.75 & 1.00 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \left[\begin{array}{cccc|c} 1.00 & 0 & 0.75 & 0.25 & 0.50 \end{array} \right]$$

This gives the four centers as $\mu_1 = 1.00, \mu_2 = 0, \mu_3 = 0.75$ and $\mu_4 = 0.25$.

Structural Stabilization

- Structural stabilization criterion is used for better generalization property of the designed RBF model
- Five possible combinations and potential design matrices are Φ_{I}, Φ_{II}, Φ_{III}, Φ_{IV}, Φ_{V}

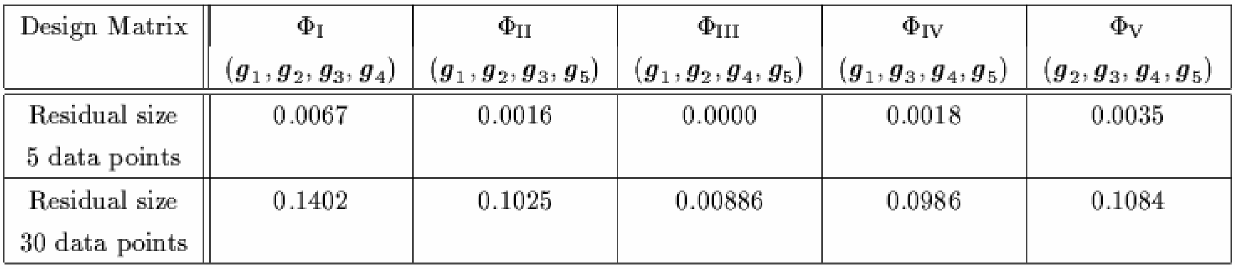


Design Matrix									
Φ_{I} (g_1, g_2, g_3, g_4)		Φ_{II} (g_1, g_2, g_3, g_5)		Φ_{III} (g_1, g_2, g_4, g_5)		Φ_{IV} (g_1, g_3, g_4, g_5)		Φ_{V} (g_2, g_3, g_4, g_5)	
Weights									
5 inputs	30 inputs	5 inputs	30 inputs	5 inputs	30 inputs	5 inputs	30 inputs	5 inputs	30 inputs
-4.8236	-4.9756	-3.4617	-3.6433	-2.0799	-2.1835	-0.5957	-0.6122	-1.4883	-1.4651
11.0454	11.0621	7.4378	7.6194	3.8223	3.9117	4.9230	4.9778	6.7826	6.7688
-9.5410	-9.3871	-4.7401	-4.8211	-3.6795	-3.7928	-7.4337	-7.6253	-8.8130	-8.9528
3.7890	3.6691	1.4958	1.5073	2.9243	3.0327	4.3588	4.5437	4.8738	5.0497
Euclidean distance between weight vectors									
0.2479		0.2695		0.2081		0.2724		0.2263	

Structural Stabilization

- Simulate additional 30 (x, y) data
- Compute 5 design matrices for Φ_{I}, Φ_{II}, Φ_{III}, Φ_{IV}, Φ_{V}
- Compute weights and compare
- Use euclidean distance

Residual Size



Design Matrix	Φ_{I} (g_1, g_2, g_3, g_4)	Φ_{II} (g_1, g_2, g_3, g_5)	Φ_{III} (g_1, g_2, g_4, g_5)	Φ_{IV} (g_1, g_3, g_4, g_5)	Φ_{V} (g_2, g_3, g_4, g_5)
Residual size 5 data points	0.0067	0.0016	0.0000	0.0018	0.0035
Residual size 30 data points	0.1402	0.1025	0.00886	0.0986	0.1084

Benchmark Data Classification Modeling

Benchmark Classification Problems

- Benchmark data for classifier learning are important for evaluating or comparing algorithms for learning from examples
- Consider two sets from Proben 1 database (Prechelt, 1994) in the UCI repository of machine learning databases:
  - Diabetes
  - Soybean

Diabetes Data: 2 Classes

- Determine if diabetes of Pima Indians is *positive* or *negative* based on description of personal data such as age, number of times pregnant, etc.
- 8 inputs, 2 outputs, 768 examples and no missing values in this data set
- The 768 example data is divided into 384 examples for training, 192 for validation and 192 for test
- Three permutations of data to generate three data sets: diabetes 1, 2, 3
- Error measure

$$Classification\ error = \frac{\#\ incorrectly\ classified\ \ patients}{\#\ total\ \ patients}$$

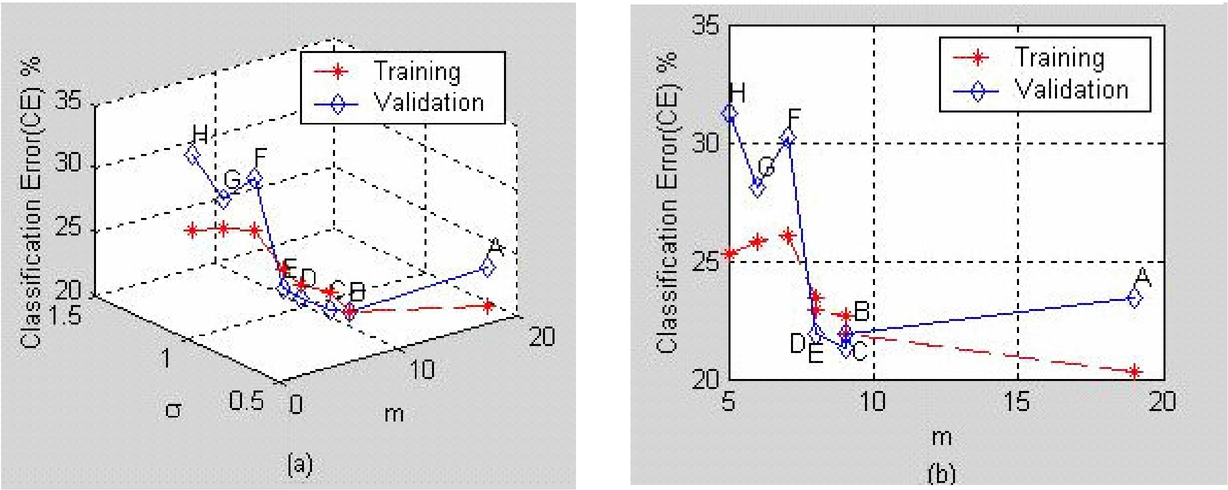
Description of Diabetes Input and Output Data

Attribute No.	No. of Attributes	Attribute Meaning	Values and Encoding
		Inputs (8)	
1	1	Number of times pregnant	0..17 → 0..1
2	1	Plasma glucose concentration after 2 hours in an oral glucose tolerance test	0..199 → 0..1
3	1	Diastolic blood pressure (mm Hg)	0..122 → 0..1
4	1	Triceps skin fold thickness (mm)	0..99 → 0..1
5	1	2-hour serum insulin (mu U/ml)	0..846 → 0..1
6	1	Body mass index (weight in kg/(height in m)^2)	0..67.1 → 0..1
7	1	Diabetes pedigree function	0.078..2.42 → 0..1
8	1	Age (years)	21..81 → 0..1
		Output (1)	
9	1	No diabetes Diabetes	-1 1

RBF Models for Diabetes 1

δ = 0.01					
Model	m	σ	Classification Error (CE), %		
			Training	Validation	Test
A	12	0.6	20.32	23.44	24.48
B	9	0.7	21.88	21.88	22.92
C	9	0.8	**22.66**	**21.35**	**23.44**
D	8	0.9	22.92	21.88	25.52
E	8	1.0	23.44	21.88	25.52
F	7	1.1	26.04	30.21	30.21
G	6	1.2	25.78	28.13	28.13
H	5	1.3	25.26	31.25	30.73

Plots of Training and Validation Errors for Diabetes 1 ($\delta = 0.01$)



Observations

- As model σ decreases (bottom to top)
  - Model complexity (*m*) increases
  - Training CE decreases
  - Validation CE decreases and then increases
  - Test CE decreases and then increases
- CE behavior as theoretically expected
- Choose model B with minimum validation CE
- Test CE is 23.44%
- Different models for other δ values
- Best model for each data set is given next

RBF Classification Models for Diabetes 1, Diabetes 2 and Diabetes 3

Problem	δ	m	σ	Classification Error (CE), %		
				Training	Validation	Test
diabetes1	0.001	10	1.2	22.66	20.83	23.96
diabetes2	0.005	25	0.5	18.23	20.31	28.13
diabetes3	0.001	15	1.0	18.49	24.48	21.88

Diabetes 1, 2, and 3

- Test error varies considerably
- Average about 24.7%

Comparison with Prechelt Results [1994]

- Linear Network (LN)
  - No hidden nodes, direct input-output connection
  - The error values are based on 10 runs
- Multilayer Network (MN)
  - Sigmoidal hidden nodes
  - 12 different topologies
  - "Best" test error reported

Diabetes Test CE for LN, MN and SG-RBF

Problem	Algorithm	Test CE %	
		Mean	Stddev
diabetes1	LN	25.83	0.56
	MN	24.57	3.53
	SG (model C)	23.96	–
diabetes2	LN	24.69	0.61
	MN	25.91	2.50
	SG (model C)	25.52	–
diabetes3	LN	22.92	0.35
	MN	23.06	1.91
	SG (model B)	23.01	–
Average	LN/MN/SG	24.48/24.46/24.20	–

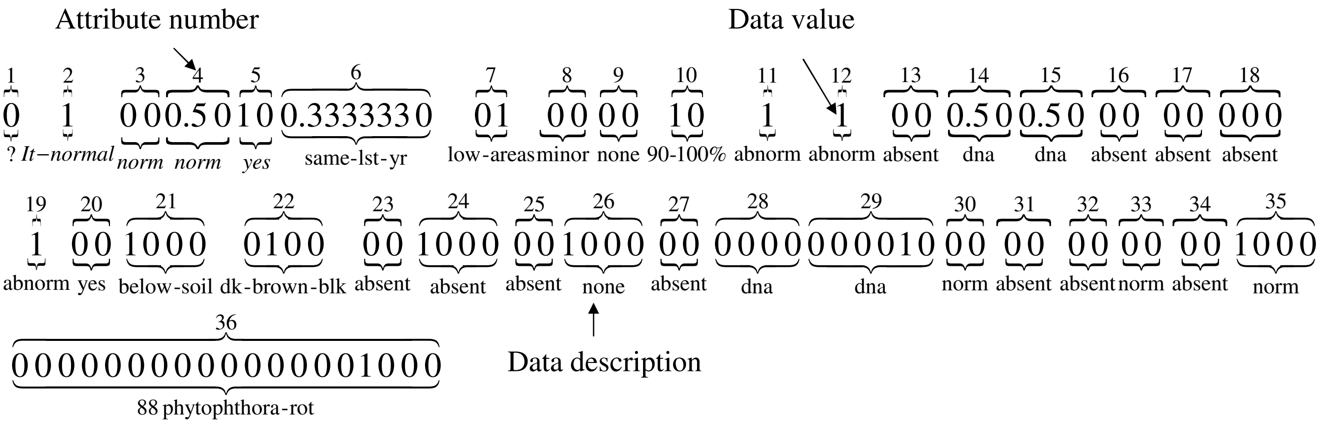
Compared to Prechelt, almost as good as best reported

RBF-SG results are fixed; no randomness

Soybean Disease Classification: 19 Classes

- Inputs (35): Description of bean, plant, plant history, etc
- Output: One of 19 disease types
- 683 examples: 342 training, 171 validation, 170 test
- Three permutations to generate Soybean 1, 2, 3
- σ: 1.1(0.2)2.5
- δ: 0.001, 0.005, 0.01

Description of One Soybean Data Point



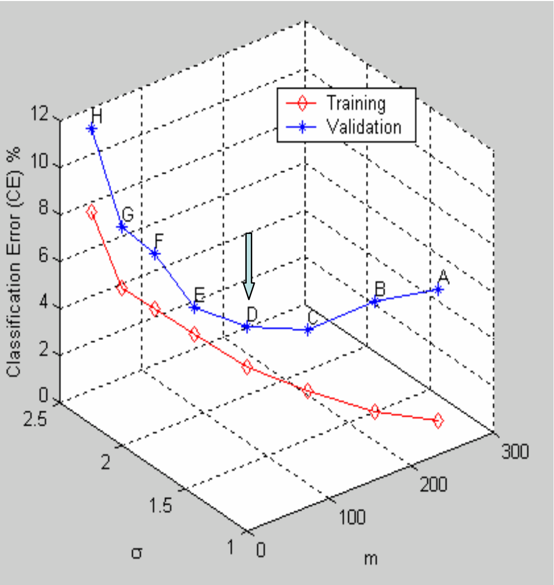
RBF Models for Soybean1 ($\delta = 0.01$)

The 683 example data set is divided into 342 examples for training set, 171 for validation set and 170 for test set

model	m	σ	CE %		
			Training	Val.	Test
A	249	1.1	0.88	6.43	8.23
B	202	1.3	2.27	5.85	7.65
C	150	1.5	2.05	4.68	8.23
D	107	1.7	2.92	4.68	10.00
E	73	1.8	4.09	5.26	10.00
F	56	2.1	4.68	7.02	10.00
G	46	2.3	4.97	7.60	11.18
H	39	2.5	7.60	11.11	15.88

The minimum validation CE equals 4.68% for two models C and D. Since, we generally prefer a simpler model, i.e., a model which smaller *m*, we choose model *D*

Plots of CE Training and Validation Errors for Sobean1 ($\delta = 0.01$)



Training error decreases from models *H* to A as *m* increases. The validation error, however, decreases up to a point and then begins to increase.

Soybean CE for LN, MN and SG-RBF

Problem	Algorithm	Test CE %	
		mean	stddev
soybean1	LN	9.47	0.51
	MN	9.06	0.80
	SG (model F)	7.65	–
soybean2	LN	4.24	0.25
	MN	5.84	0.87
	SG (model G)	4.71	–
soybean3	LN	7.00	0.19
	MN	7.27	1.16
	SG (model E)	4.12	–
Average	LN/MN/SG	6.90/7.39/5.49	–

The SG-RBF classifiers have ***smaller errors*** for soybean1 and soyben3. Overall better average error and no randomness

Data Mining and Knowledge Discovery

Knowledge Discovery: Software Engineering

- KDD is the nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data
- KDD includes data mining as a critical phase of the KDD process; activity of extracting patterns by employing a specific algorithm
- Currently KDD is used for, e.g., text mining, sky surveys, customer relations managements, etc
- We discuss knowledge discovery about criticality evaluation of software modules

KDD Process

- KDD refers to all activities from data collection to use of the discovered knowledge
- Typical steps in KDD
  - Learning the application domain: prior knowledge; study objectives
  - Creating dataset: identification of relevant variables or factors
  - Data cleaning and preprocessing: removal of wrong data and outliers, consistency checking, methods for dealing with missing data fields, and preprocessing
  - Data reduction and projection: finding useful features for data representation, data reduction and appropriate transformations
  - Choosing data mining function: decisions about modeling goal such as classification or prediction

KDD Process (cont.)

- Choosing data mining algorithms: algorithm selection for the task chosen in the previous step
- Data mining: actual activity of searching for patterns of interest such as classification rules, regression or neural network modeling as well as validation and accuracy assessment
- Interpretation and use of discovered knowledge: presentation of discovered knowledge; and taking specific steps consistent with the goals of knowledge discovery

KDD Goals: SE

- Software development is very much like an industrial production process consisting of several overlapping activities, formalized as life-cycle models
- Aim of collecting software data is to perform knowledge discovery activities to seek useful information
- Some typical questions of interest to software engineers and managers are
  - What features (metrics) are indicators of high quality systems
  - What metrics should be tracked to assess system readiness
  - What patterns of metrics indicate potentially high defect modules
  - What metrics can be related to software maturity during development
- Hundreds of such questions are of interest in SE

List of Metrics from NASA Metrics Database

x_7	Faults
x_9	Function Calls from This Component
x_{10}	Function Calls to This Component
x_{11}	Input/Output Statements
x_{12}	Total Statements
x_{13}	Size of Component in Number of Program Lines
x_{14}	Number of Comment Lines
x_{15}	Number of Decisions
x_{16}	Number of Assignment Statements
x_{17}	Number of Format Statements
x_{18}	Number of Input/Output Parameters
x_{19}	Number of Unique Operators
x_{20}	Number of Unique Operands
x_{21}	Total Number of Operators
x_{22}	Total Number of Operands

# of faults

Design metrics
x_9, x_{10}, x_{18}

Coding metrics
x_{13}, x_{14}, x_{15}

Module level product metrics

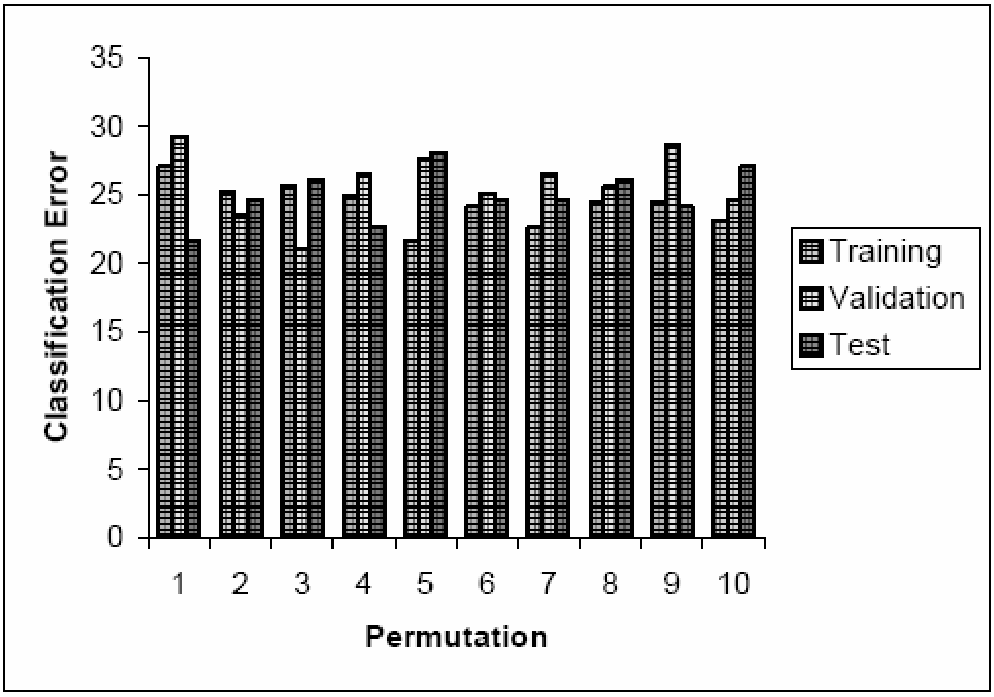
KDD Process for Software Modules

- Application domain: Early identification of critical modules which are subjected to additional testing, etc. to improve system quality
- Database: NASA metrics DB; 14 metrics; many projects; select 796 modules
- Transformation: Normalize metrics to (0, 1); class is +1 if number of faults exceeds five; -1 otherwise; ten permutation with (398 training; 199 validation ; 199 test)
- Function: RBF classifiers
- Data Mining: Classification modeling for design; coding; fourteen metrics
- Interpretation: Compare accuracy; determine relative adequacy of different sets of metrics

Classification: Design Metrics

		Classification Error (%)		
Permutation	*m*	Training	Validation	Test
1	4	27.1	29.2	21.6
2	6	25.2	23.6	24.6
3	4	25.6	21.1	26.1
4	7	24.9	26.6	22.6
5	4	21.6	27.6	28.1
6	7	24.1	25.1	24.6
7	3	22.6	26.6	24.6
8	5	24.4	28.6	24.1
9	7	24.4	28.6	24.1
10	4	23.1	24.6	27.1

Design Metrics (cont.)



Test Error Results

			Confidence Bounds and Width	
Metrics	Average	SD	90 %	95 %
Design Metrics	24.95	1.97	{23.81, 26.05}	{23.60, 26.40}
Coding Metrics	23.00	3.63	{20.89, 25.11}	{21.40, 25.80}
Fourteen Metrics	24.35	2.54	{22.89, 25.81}	{22.55, 26.15}

Confidence bound:

$$\left\{Avg\,TE \pm t_{9;\alpha/2}\left(SD\,of\,TE\right)\big/\sqrt{10}\right\}$$

Summary of Data Mining Results

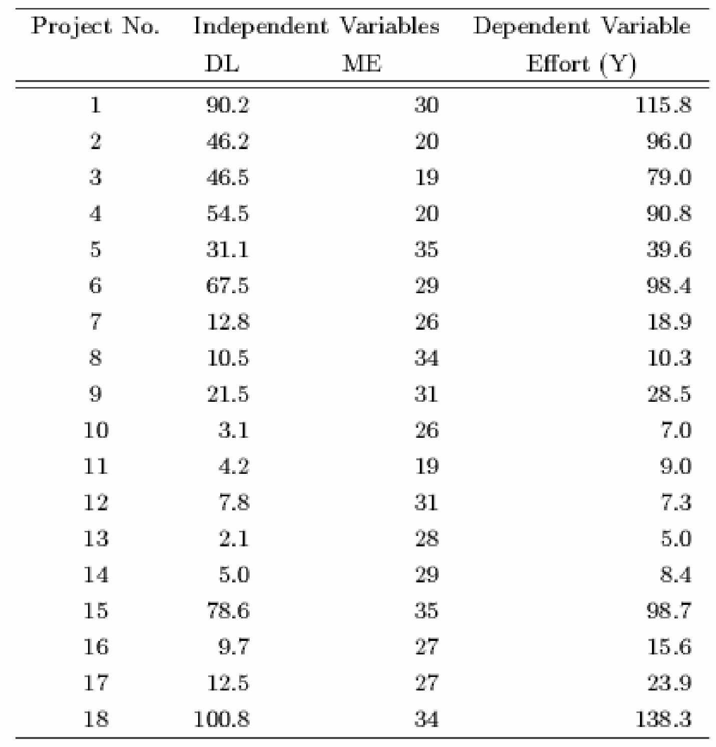
- Predictive error on test data about 23%
- Very good for software engineering data where low accuracy is common; errors can be as high 60% or more
- Classification errors are similar for design metrics, coding metrics, all (14) metrics
- However, design metrics are available in early development phases and are preferred for developing classification models
- Knowledge discovered
  - good classification accuracy
  - can use design metrics for criticality evaluation of software modules
- What next
  - KDD on other projects using RBF

Empirical Data Modeling in Software Engineering Project Effort Prediction

Software Effort Modeling

- Accurate estimation of soft project effort is one of the most important empirical modeling tasks in software engineering as indicated by the large number of models developed over the past twenty years
- Most of the popularly used models employ a regression type equation relating effort and size, which is then calibrated for local environment
- We use NASA data to develop RBF models for effort (Y) based on Developed Lines (DL) and Methodology (ME)
- DL is KLOC; ME is composite score; Y is Man-months

NASA Software Project Data



Project No.	Independent Variables		Dependent Variable
	DL	ME	Effort (Y)
1	90.2	30	115.8
2	46.2	20	96.0
3	46.5	19	79.0
4	54.5	20	90.8
5	31.1	35	39.6
6	67.5	29	98.4
7	12.8	26	18.9
8	10.5	34	10.3
9	21.5	31	28.5
10	3.1	26	7.0
11	4.2	19	9.0
12	7.8	31	7.3
13	2.1	28	5.0
14	5.0	29	8.4
15	78.6	35	98.7
16	9.7	27	15.6
17	12.5	27	23.9
18	100.8	34	138.3

RBF Based on DL

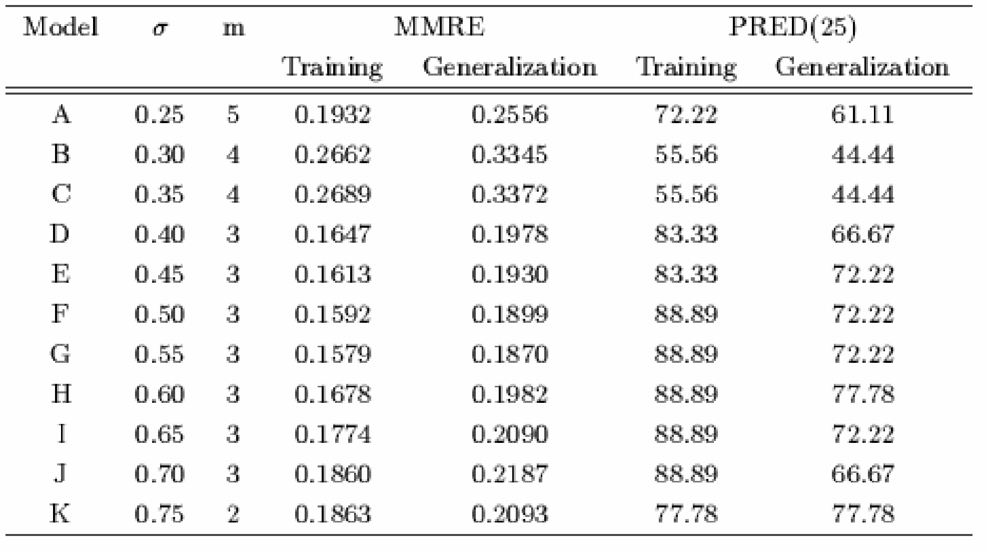
- Simple problem; for illustration
- Our goal is to seek a parsimonious model which provides a good fit and exhibits good generalization capability
- Modeling steps
  - Select δ= 1%, 2%, and 0.1% and a range of σ values
  - For each σ, determine the value of m which satisfies δ
  - Determine parameters μ and $\mathbf{w}$ according to the SG algorithm
  - Compute training error for the data on 18 projects
  - Use LOOCV technique to compute generalization error
  - Select the model which has minimum generalization error and small training error
  - Repeat above for each δ and select the most appropriate model

Two Error Measures

- MMRE = $\frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i - \hat{Y}_i\right|}{Y_i}$

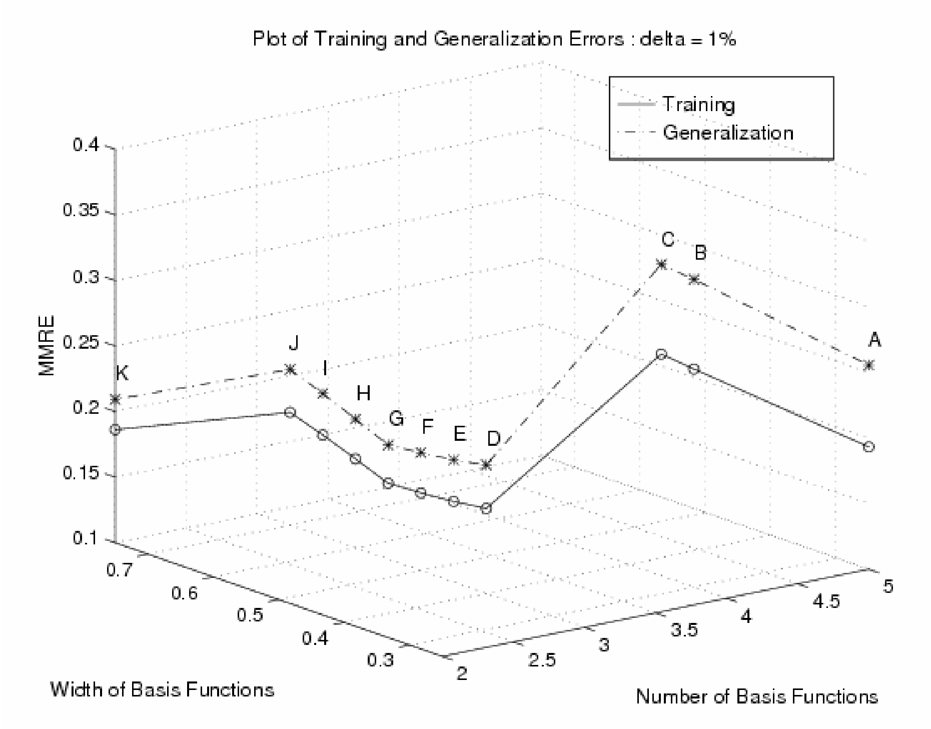
- PRED(25) = Percentage of predictions falling within 25% of the actual known values

RBF Designs and Performance Measure for (DL-Y) Models (δ= 1%)

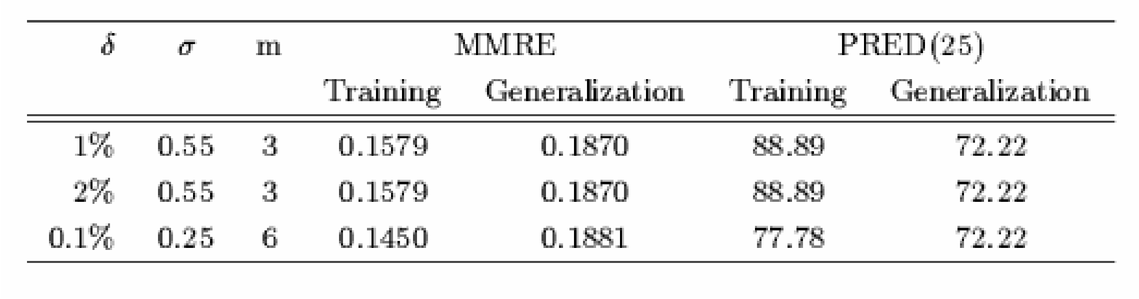


Model	σ	m	MMRE		PRED(25)	
			Training	Generalization	Training	Generalization
A	0.25	5	0.1932	0.2556	72.22	61.11
B	0.30	4	0.2662	0.3345	55.56	44.44
C	0.35	4	0.2689	0.3372	55.56	44.44
D	0.40	3	0.1647	0.1978	83.33	66.67
E	0.45	3	0.1613	0.1930	83.33	72.22
F	0.50	3	0.1592	0.1899	88.89	72.22
G	0.55	3	0.1579	0.1870	88.89	72.22
H	0.60	3	0.1678	0.1982	88.89	77.78
I	0.65	3	0.1774	0.2090	88.89	72.22
J	0.70	3	0.1860	0.2187	88.89	66.67
K	0.75	2	0.1863	0.2093	77.78	77.78

A Graphical Depiction of MMRE Measures for Candidate Models



RBF Models for (DL-Y) Data



δ	σ	m	MMRE		PRED(25)	
			Training	Generalization	Training	Generalization
1%	0.55	3	0.1579	0.1870	88.89	72.22
2%	0.55	3	0.1579	0.1870	88.89	72.22
0.1%	0.25	6	0.1450	0.1881	77.78	72.22

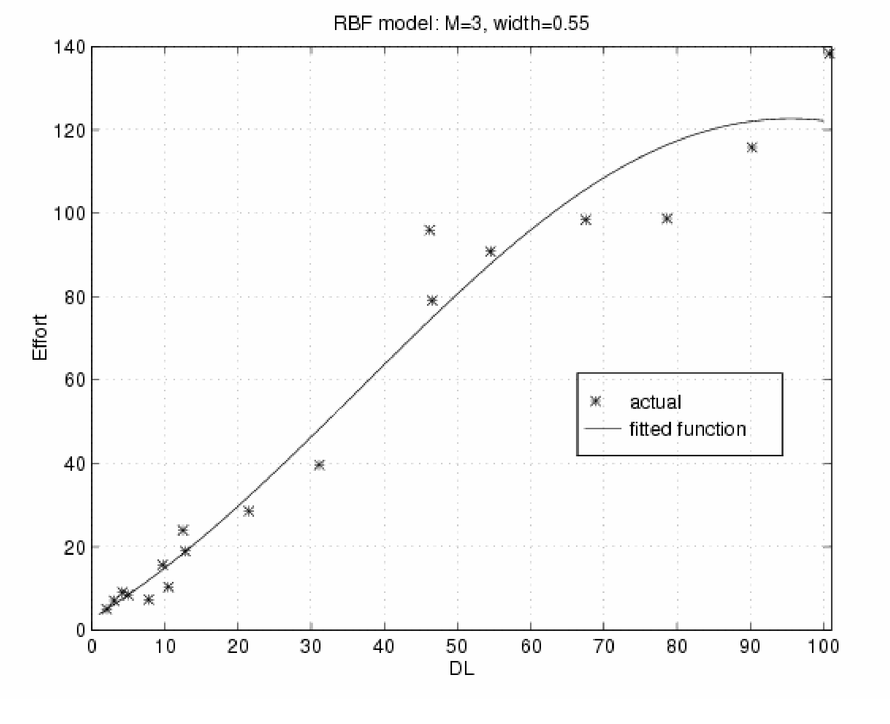
Estimation Model

$$\begin{aligned}\hat{\mathrm{Y}}_N \quad = \quad & -0.3652 \exp\left(-\frac{1}{2(.55)^2}\cdot(\mathrm{DL}_N - 0.0)^2\right) \\ & +0.3709 \exp\left(-\frac{1}{2(.55)^2}\cdot(\mathrm{DL}_N - 0.5309)^2\right) \\ & +0.6907 \exp\left(-\frac{1}{2(.55)^2}\cdot(\mathrm{DL}_N - 1.0)^2\right)\end{aligned}$$

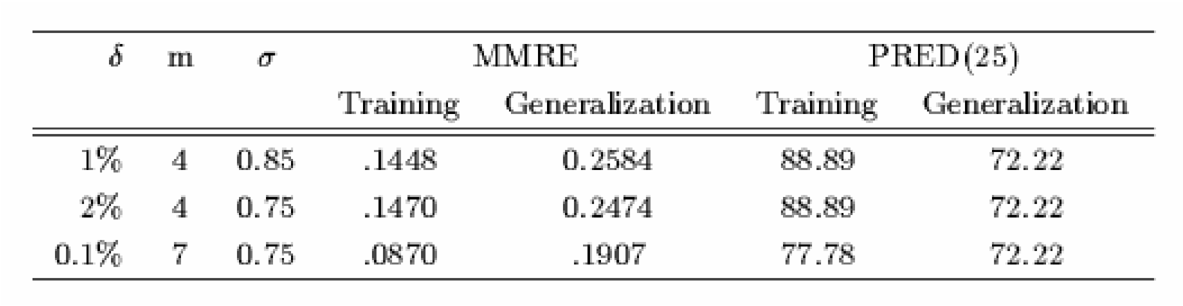
where

$$\mathrm{DL}_N = \frac{\mathrm{DL} - 2.1}{100.8 - 2.1} \quad \text{and} \quad \mathrm{Y}_N = \frac{\mathrm{DL} - 5.0}{138.3 - 5.0}$$

Plot of the Fitted RBF Estimation Model and Actual Effort as a Function of DL

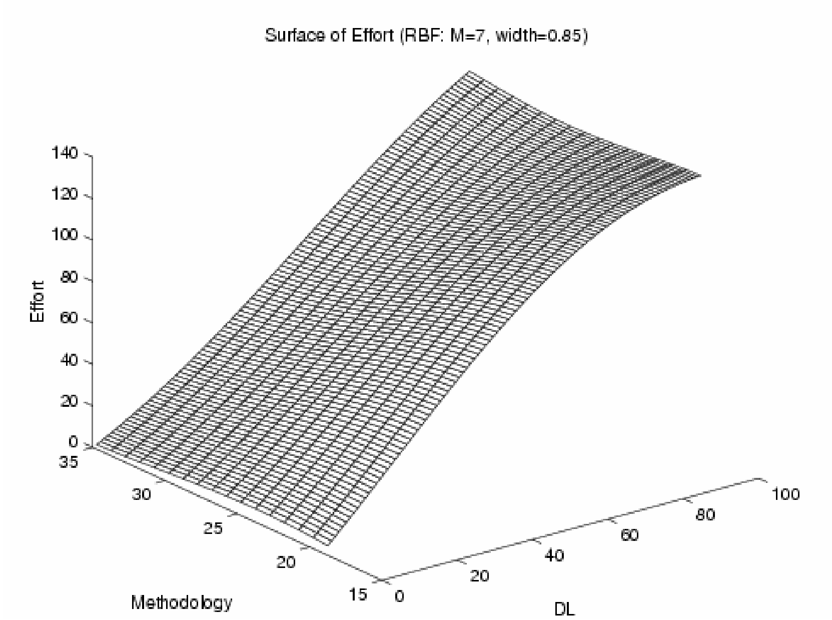


Models for DL and ME

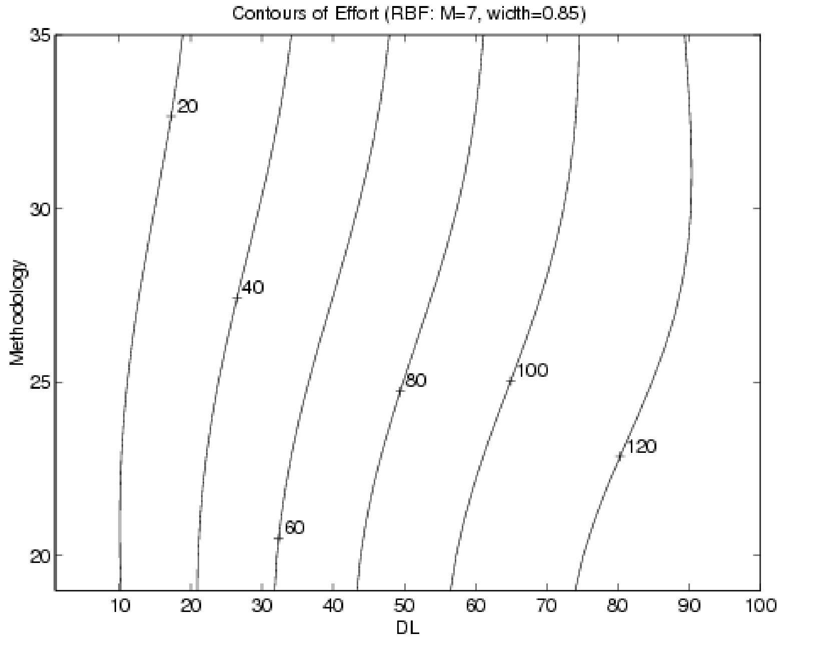


δ	m	σ	MMRE		PRED(25)	
			Training	Generalization	Training	Generalization
1%	4	0.85	.1448	0.2584	88.89	72.22
2%	4	0.75	.1470	0.2474	88.89	72.22
0.1%	7	0.75	.0870	.1907	77.78	72.22

Plot of the Fitted RBF Estimation Model and Actual Effort as a Function DL and ME



Plot of the Fitted RBF Estimation Model and Actual Effort as a Function DL and ME (cont.)



KDD: Microarray Data Analysis

OUTLINE

1. Microarray Data and Analysis Goals
2. Background
3. Classification Modeling and Results
4. Sensitivity Analyses
5. Remarks

MICROARRAY DATA AND ANALYSIS GOALS

Data[*]

- A matrix of gene expression values $X_{n\times d}$
- Cancer class vector y=1(ALL),y=0 (AML), $Y_{n\times d}$
- Training set n=38, Test set n=34
- Two data sets with number of genes d=7129 and d=50

[*] Golub et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531-537, 1999.

MICROARRAY DATA AND ANALYSIS GOALS (cont.)

Classification Goal

- Develop classification models to predict leukemia class (ALL or AML) based on training set
- Use Radial Basis Function (RBF) model and employ recently developed Shin-Goel (SG) design algorithm

Model selection

- Choose the model that achieves the best balance between fitting and model complexity
- Use tradeoffs between classification errors on training and test sets as model selection criterion

BACKGROUND

- Advances in microarray technology are producing very large datasets that require proper analytical techniques to understand the complexities of gene functions. To address this issue, presentations at CAMDA2000 conference* discussed analyses of the same data sets using different approaches
- Golub et al's dataset (one of two at CAMDA) involves classification into acute lymphoblastic (ALL) or acute myeloid (AML) leukemia based on 7129 attributes that correspond to human gene expression levels

* Critical Assessment of Microarray Data; for papers see Lin, S. M. and Johnson, K. E (Editors), Methods of Microarray Data Analysis, Kluwer, 2002

BACKGROUND (cont.)

- In this study, we formulate the classification problem as a two step process. First we construct a radial basis function model using a recent algorithm of Shin and Goel[**]. Then model performance is evaluated on test set classification

[**] Shin, M, Goel. A. L. Empirical Data Modeling in Software Engineering Using Radial Basis Functions. IEEE Transactions on Software Engineering, 26:567-576, 2000

Shin, M, Goel, A. L. Radial Basis Function Model Development and Analysis Using the SG Algorithm (Revised), Technical Report, Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, 2002

CLASSIFICATION MODELING

- Data of Golub et al[*] consists of 38 training samples (27 ALL, 11 AML) and 34 test samples (20 ALL, 14 AML). Each sample corresponds to 7129 genes. They also selected 50 most informative genes and used both sets for classification studies
- We develop several RBF classification models using the SG algorithm and study their performance on training and test data sets
- Classifier with best compromise between training and test errors is selected

[*] Golub et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531-537, 1999.

CLASSIFICATION MODELING (cont.)

<u>Summary of Results</u>

- For specified RC and σ, SG algorithm first computes minimum m and then the centers and weights
- We use RC = 99% and 99.5%
- 7129 gene set: $\sigma = 20(2)32$,

  50 gene set: $\sigma = 2(0.4)4$
- Table 1 lists the “Best” RBF models

Classification models and Their Performance

Data Set	RC	m	σ	Correct Classification		Classification error %	
				training	test	training	test
7129 genes	99.0%	29	26	38	29	0	14.71
	99.5%	35	30	38	29	0	14.71
50 genes	99.0%	6	3.2	38	33	0	2.94
	99.5%	13	3.2	38	33	0	2.94

SENSITIVITY ANALYSES (7129 Gene Data)

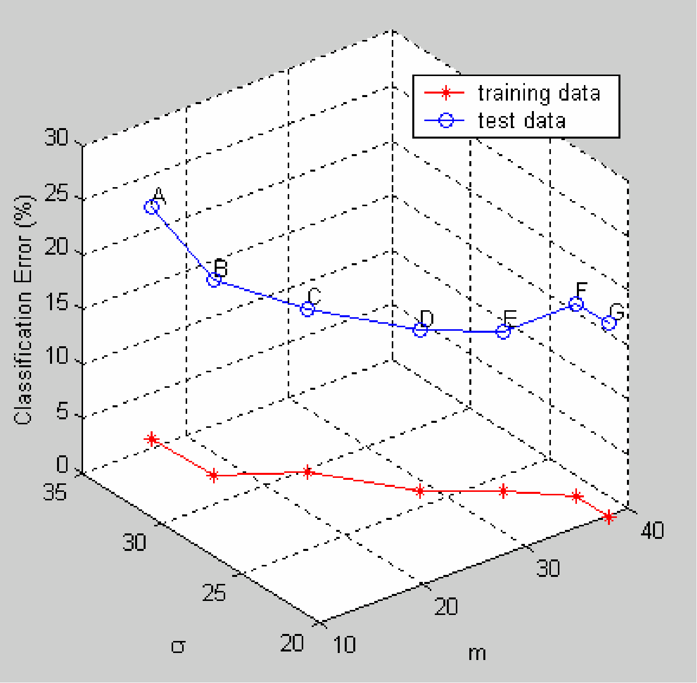
RC=99%; σ=20(2)32

- SG algorithm computes minimum m (no. of basis functions) that satisfies RC
- Table 2 and Figure 4, show models and their performance on training and test sets
- “Best” model is D: m=29, σ=26
- Correctly classifies 38/38 training samples; only 29/34 test samples
- Models A and B represent underfitting, F and G overfitting; Figure 1 shows underfit-overfit realization

Classification results (7129 Genes, RC=99%) (38 training, 34 test samples)

Model	σ	m	Correct classification		Classification error %	
			training	test	training	test
A	32	12	36	25	5.26	26.47
B	30	15	37	27	2.63	20.59
C	28	21	37	28	2.63	17.65
D	26	29	38	29	0	14.71
E	24	34	38	29	0	14.71
F	22	38	38	28	0	17.65
G	20	38	38	28	0	17.65

Classification Errors (7129 genes; RC=99%)



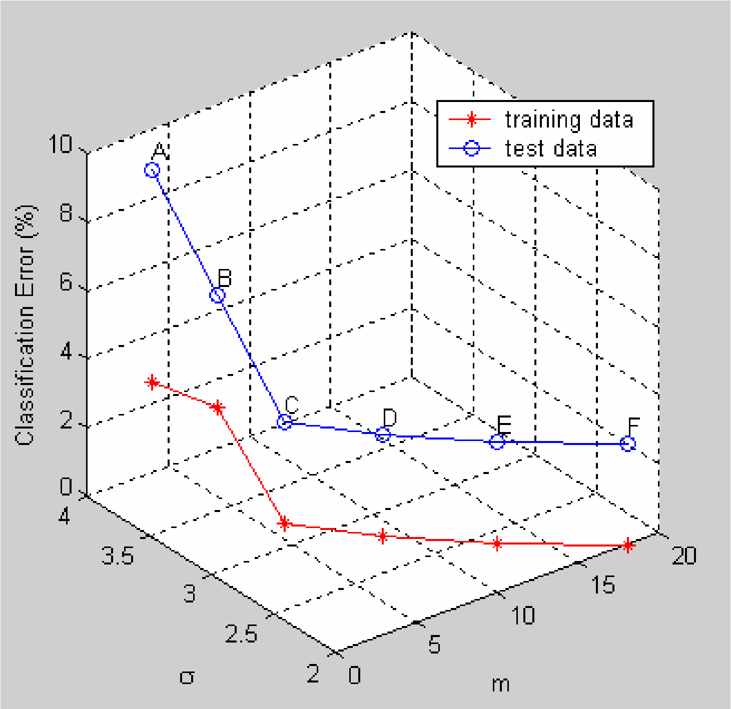
SENSITIVITY ANALYSES (cont.) (50 Gene Data)

- Table 3 and Figure 5 show several RBF models and their performance on 50 gene training and test data
- Model C (m=6, σ=3.2) seems to be the best one with 38/38 correct classification on training data and 33/34 on test data
- Model A represents underfit and models D, E and F seem to be unnecessarily complex, with no gain in classification accuracy

Classification Results
(50 Genes RC=99%)
(38 Training, 34 Test Sets)

Model	σ	Basis Functions (m)	Correct classification		Classification error (%)	
			training	test	training	test
A	4.0	4	37	31	2.63	8.82
B	3.6	5	37	32	2.63	5.88
C	3.2	6	38	33	0	2.94
D	2.8	9	38	33	0	2.94
E	2.4	13	38	33	0	2.94
F	2.0	18	38	33	0	2.94

Classification Errors (50 genes; RC = 99%)



REMARKS

- This study used Gaussian RBF model and the SG algorithm for the cancer classification problem of Golub et. al. Here we present some remarks about our methodology and future plans
- RBF models have been used for classification in a broad range of applications, from astronomy to medical diagnosis and from stock market to signal processing.
- Current algorithms, however, tend to produce inconsistent results due to their ad-hoc nature
- The SG algorithm produces consistent results, has strong mathematical underpinnings, primarily involves matrix computations and no search or optimization. It can be almost totally automated.

Summary

In this tutorial, we discussed the following issues

- Problems of classification and prediction; and the modeling considerations involved
- Structure of the RBF model and some design approaches
- Detailed coverage of the new (Shin-Goel) SG algebraic algorithm with illustrative examples
- Classification modeling using the SG algorithm for two benchmark data sets
- KDD and DM issues using RBF/SG in software engineering and cancer class prediction

Selected References

- C. M. Bishop, Neural Network for Pattern Recognition, Oxford, 1995.
- S. Haykin, Neural Networks, Prentice Hall, 1999.
- H. Lim, An Empirical Study of RBF Models Using SG Algorithm, MS Thesis, Syracuse University, 2002.
- M. Shin, Design and Evaluation of Radial Basis Function Model for Function Approximation, Ph.D. Thesis, Syracuse University, 1998.
- M. Shin and A. L. Goel, "Knowledge discovery and validation in software engineering," *Proceedings of Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, April 1999, Orlando, FL.

Selected References (cont.)

- M. Shin and A. L. Goel, "Empirical data modeling in software engineering using radial basis functions," *IEEE transactions on software engineering*, vol. 26, no. 6, June 2000.
- M. Shin and C. Park, "A Radial Basis Function approach for pattern recognition and its applications," *ETRI journal*, vol. 22 , no. 2, pp.1-10, June 2000.
- M. Shin, A. L. Goel and H. Lim, "A new radial basis function design methodology with applications in cancer classification," *Proceedings of the IASTED conference on Applied Modeling and Simulation,* November 4-6 2002, Cambridge, USA.