Radial Basis Functions (RBF)

- **k-Nearest Neighbor**: Only considers $k$-nearest neighbors. Each neighbor has equal weight.

  What about using *all* data to compute $g(x)$?

- **RBF**: Use all data. Data further away from $x$ have less weight.

Weighting the Data Points: $\alpha_n$

Test point $x$.

$\alpha_n$: the weight of $x_n$ in $g(x)$.

$$\alpha_n(x) = \phi \left( \frac{||x - x_n||}{r} \right)$$
Weighting the Data Points: $\alpha_n$

Test point $x$.

$\alpha_n$: the weight of $x_n$ in $g(x)$.

$$\alpha_n(x) = \phi \left( \frac{|x - x_n|}{r} \right)$$

weighting depends on the distance $|x - x_n|$.

Most popular kernel: Gaussian

$$\phi(z) = e^{-\frac{1}{2}z^2}.$$
Nonparametric RBF – Regression

\[ \alpha_n(x) = \phi \left( \frac{||x - x_n||}{r} \right) \]

\[ g(x) = \sum_{n=1}^{N} \left( \frac{\alpha_n(x)}{\sum_{m=1}^{N} \alpha_m(x)} \right) \cdot y_n \]

Weighted average of target values

Nonparametric RBF – Classification

\[ \alpha_n(x) = \phi \left( \frac{||x - x_n||}{r} \right) \]

\[ g(x) = \text{sign} \left( \sum_{n=1}^{N} \left( \frac{\alpha_n(x)}{\sum_{m=1}^{N} \alpha_m(x)} \right) \cdot y_n \right) \]

Choice of Scale \( r \)

Nearest Neighbor

Choosing \( k \):

- \( k = 3 \)
- \( k = \sqrt{N} \)
- \( CV \)

Choosing \( r \):

- \( r \sim \frac{1}{\sqrt{N}} \)
- \( CV \)

overfitting

underfitting

Nonparametric RBF – Logistic Regression

\[ \alpha_n(x) = \phi \left( \frac{||x - x_n||}{r} \right) \]

\[ g(x) = \sum_{n=1}^{N} \left( \frac{\alpha_n(x)}{\sum_{m=1}^{N} \alpha_m(x)} \right) \cdot \left[ y_n = +1 \right] \]
Highlights of Nonparametric RBF

1. Simple (‘smooth’ version of k-NN rule).
2. No training.
3. Near optimal $E_{\text{out}}$.
4. Easy to justify classification to customer.
5. Can do classification, multi-class, regression, logistic regression.
6. Computationally demanding.

Scaled Bumps on Each Data Point

$$g(x) = \sum_{n=1}^{N} \left( \frac{\alpha_n(x)}{\sum_{m=1}^{N} \alpha_m(x)} \right) \cdot y_n$$

Weighted average of $y_n$.

$$g(x) = \sum_{n=1}^{N} \left( \frac{y_n}{\sum_{m=1}^{N} \alpha_m(x)} \right) \cdot \phi \left( \frac{|x - x_n|}{r} \right)$$

Sum of bumps at $x_n$ scaled by $w_n(x)$.
Nonparametric RBF: \( w_n(x) \)

\[
g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{\|x - x_n\|}{r} \right)
\]

Only need to specify \( r \).

Parametric RBF – A Linear Model

Nonparametric RBF

\[
g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{\|x - x_n\|}{r} \right)
\]

Only need to specify \( r \).

Parametric RBF

\[
h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{\|x - x_n\|}{r} \right)
\]

Fix \( r \); need to determine the parameters \( w_n \).

RBF-Nonlinear Transform Depends on Data

\[
h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{\|x - x_n\|}{r} \right) = w^T z
\]

\[
z = \Phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_N(x) \end{bmatrix}, \quad \phi_n(x) = \phi \left( \frac{\|x - x_n\|}{r} \right).
\]

\[
Z = \begin{bmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_N^T \end{bmatrix} = -\begin{bmatrix} \Phi(x_1)^T \\ \Phi(x_2)^T \\ \vdots \\ \Phi(x_N)^T \end{bmatrix}
\]
**RBF-Nonlinear Transform Depends on Data**

\[ h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{|x - x_n|}{r} \right) = w^T z \]

\[ z = \Phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_N(x) \end{bmatrix}, \quad \phi_n(x) = \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ Z = \begin{bmatrix} -z_1^T \\ -z_2^T \\ \vdots \\ -z_N^T \end{bmatrix} = \begin{bmatrix} -\Phi(x_1)^T \\ -\Phi(x_2)^T \\ \vdots \\ -\Phi(x_N)^T \end{bmatrix} \]

Fit the data \( h(x_i) = y_i \)

\[ w = Z^T y = (Z^T Z)^{-1} Z^T y \]

**Reducing the Number of Bumps: Nonparametric**

\[ g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

**Reducing the Number of Bumps: Parametric**

\[ g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

**Reducing the Number of Bumps: \( k \)-RBF-Network**

\[ g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ h(x) = w_0 + \sum_{j=1}^{k} w_j \cdot \phi \left( \frac{|x - \mu_j|}{r} \right) = w^T \Phi(x) \]

\[ \Phi(x)^T = [\phi_1(x), \ldots, \phi_N(x)], \quad \text{where} \quad \phi_j(x) = \phi \left( \frac{|x - \mu_j|}{r} \right). \]
Reducing the Number of Bumps: \textit{k-RBF-Network}

\[ g(x) = \sum_{n=1}^{N} w_n(x) \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ h(x) = \sum_{n=1}^{N} w_n \cdot \phi \left( \frac{|x - x_n|}{r} \right) \]

\[ h(x) = w_0 + \sum_{j=1}^{k} w_j \cdot \phi \left( \frac{|x - \mu_j|}{r} \right) \]

\[ w \cdot \Phi(x) \]

\[ \Phi(x)^T = [\Phi_1(x), \ldots, \Phi_n(x)] \text{, where } \Phi_i(x) = \phi \left( \frac{|x - \mu_i|}{r} \right) \]

\[ w = (Z^T Z)^{-1} Z^T y \]

Fitting the RBF-network to the data (given \(k, r\)):
\begin{itemize}
  \item Use the inputs \(X\) to determine \(k\) centers \(\mu_1, \ldots, \mu_k\).
  \item Compute the \(N \times (k + 1)\) feature matrix \(Z\)
\end{itemize}

\[ Z = \begin{bmatrix}
  -x_1^T & -x_2^T & \cdots & -x_N^T
  \\
  -x_1^T & -x_2^T & \cdots & -x_N^T
  \\
  -x_1^T & -x_2^T & \cdots & -x_N^T
\end{bmatrix} \quad \text{, where } \Phi(x) = \begin{bmatrix}
  \Phi_1(x)^T \\
  \Phi_2(x)^T \\
  \vdots \\
  \Phi_n(x)^T
\end{bmatrix}, \quad \phi \left( \frac{|x - \mu|}{r} \right) \]

Each row of \(Z\) is the RBF-feature corresponding to \(x_n\) (with dummy bias coordinate 1).

\begin{itemize}
  \item Fit the linear model \(Zw\) to \(y\) to determine the weights \(w^*\).
    \begin{itemize}
      \item classification: PLA, pocket, linear programming, \ldots
      \item regression: pseudoinverse.
      \item logistic regression: gradient descent on cross entropy error.
    \end{itemize}
\end{itemize}

Fitting the Data

Before: bumps were centered on \(x_n\) — no choice

Now: we may choose the bump centers \(\mu_j\)

Choose them to 'cover' the data

As the centers of \(k\) 'clusters'

Given the bump centers, \textbf{we have a linear model to determine the }\(w_j\)

That's 'easy', we know how to do that.

Our Example

\[ k = 4, \ r = \frac{1}{x} \]

\[ k = 10, \ r = \frac{1}{x} \]

Choose \(r\) using CV, or (a heuristic):

\[ r \sim \frac{\text{radius of data}}{k^{1/2}} \quad \text{(so your clusters 'cover' the data)} \]
Use Regularization to Fight Overfitting

\[ k = 10, \quad r = \frac{1}{k} \]

\[ k = 10, \quad r = \frac{1}{k}, \text{ regularized} \]

**A Peek at Unsupervised Learning**

**Reflecting on the \( k \)-RBF-Network**

1. We derived it as a ‘soft’ generalization of \( k \)-NN rule.
   - Can also be derived from regularization theory.
   - Can also be derived from noisy interpolation theory.

2. Can use nonparametric or parametric versions.

3. Given centers, ‘easy’ to learn the weights using techniques from linear models.
   - A linear model with an adaptable nonlinear transform.

4. We used uniform bumps – can have different shapes \( \Sigma_j \).

5. **NEXT:** How to better choose the centers: unsupervised learning.

**Summary of \( k \)-RBF-Network**

- \( w = (Z^T Z + \lambda I)^{-1} Z^T y \)