

Learning From Data
Lecture 18
Radial Basis Functions

Non-Parametric RBF

Parametric RBF

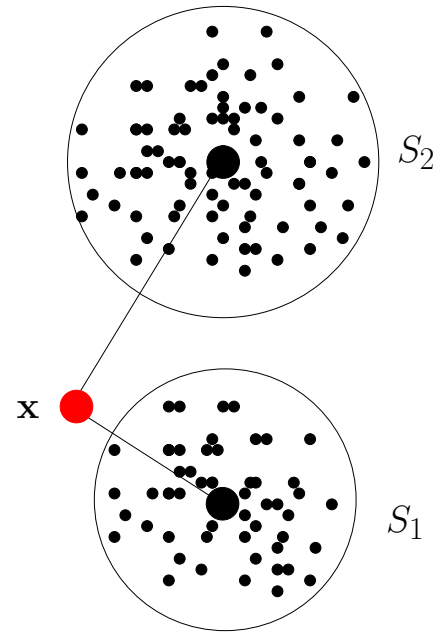
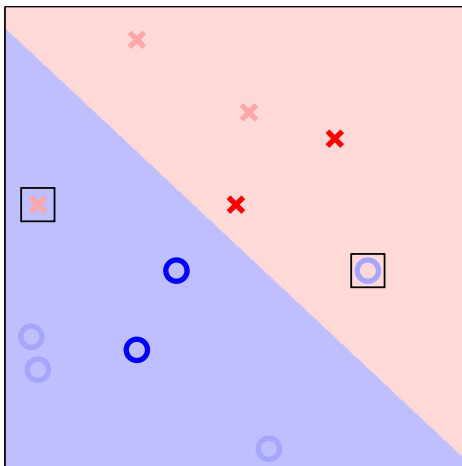
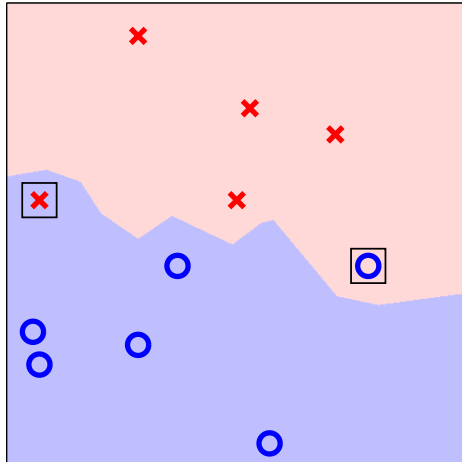
k -RBF-Network

M. Magdon-Ismail

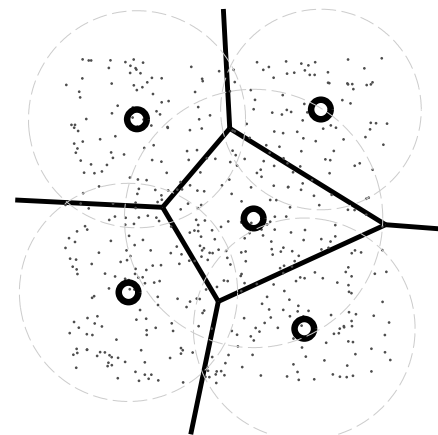
CSCI 4100/6100

RECAP: Data Condensation and Nearest Neighbor Search

Training Set Consistent



Branch and bound for finding nearest neighbors.



Lloyd's algorithm for finding a good clustering.

Radial Basis Functions (RBF)

k -Nearest Neighbor: Only considers k -nearest neighbors.
each neighbor has equal weight

What about using *all* data to compute $g(\mathbf{x})$?

RBF: Use all data.
data further away from \mathbf{x} have less weight.

Weighting the Data Points: α_n

Test point \mathbf{x} .

α_n : the weight of \mathbf{x}_n in $g(\mathbf{x})$.

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

Weighting the Data Points: α_n

Test point \mathbf{x} .

α_n : the weight of \mathbf{x}_n in $g(\mathbf{x})$.

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

weighting depends on the distance $\|\mathbf{x} - \mathbf{x}_n\|$

Weighting the Data Points: α_n

Test point \mathbf{x} .

α_n : the weight of \mathbf{x}_n in $g(\mathbf{x})$.

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

... relative to a *scale parameter* r

Weighting the Data Points: α_n

Test point \mathbf{x} .

α_n : the weight of \mathbf{x}_n in $g(\mathbf{x})$.

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

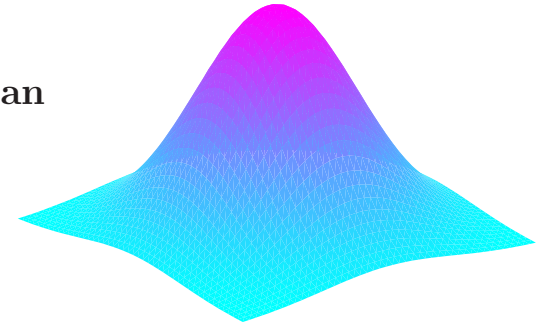
kernel ϕ determines how the weighting decreases with distance

Weighting the Data Points: α_n

Test point \mathbf{x} .

Most popular kernel: Gaussian

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



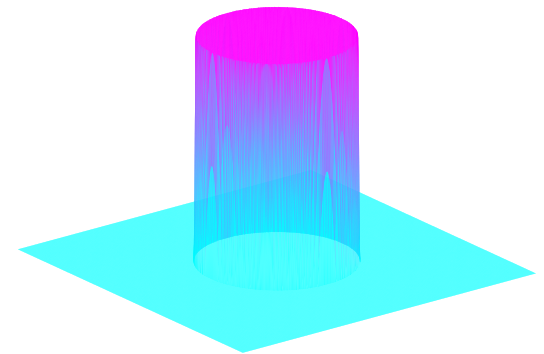
α_n : the weight of \mathbf{x}_n in $g(\mathbf{x})$.

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

kernel ϕ determines how the weighting decreases with distance

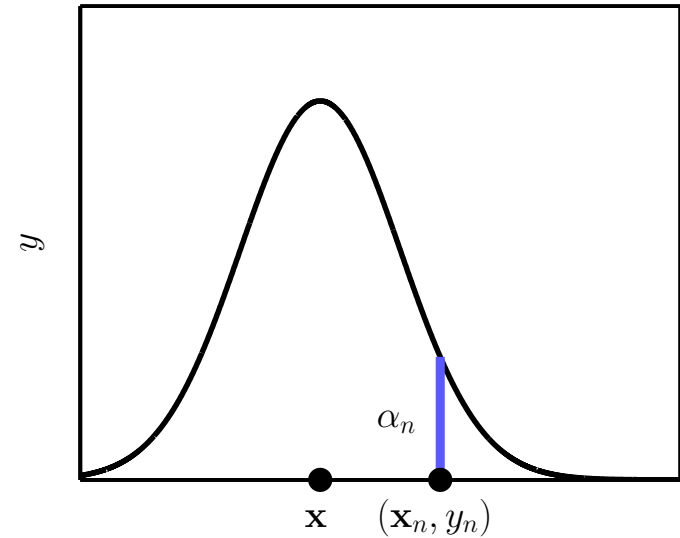
Window kernel, mimics k -NN,

$$\phi(z) = \begin{cases} 1 & z \leq 1, \\ 0 & z > 1, \end{cases}$$



Nonparametric RBF – Regression

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

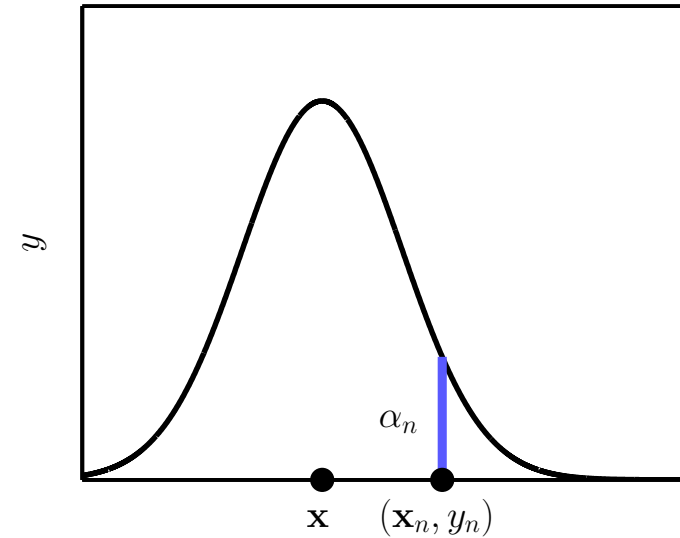


$$g(\mathbf{x}) = \sum_{n=1}^N \left(\frac{\alpha_n(\mathbf{x})}{\sum_{m=1}^N \alpha_m(\mathbf{x})} \right) \cdot y_n$$

Weighted average of target values

Nonparametric RBF – Classification

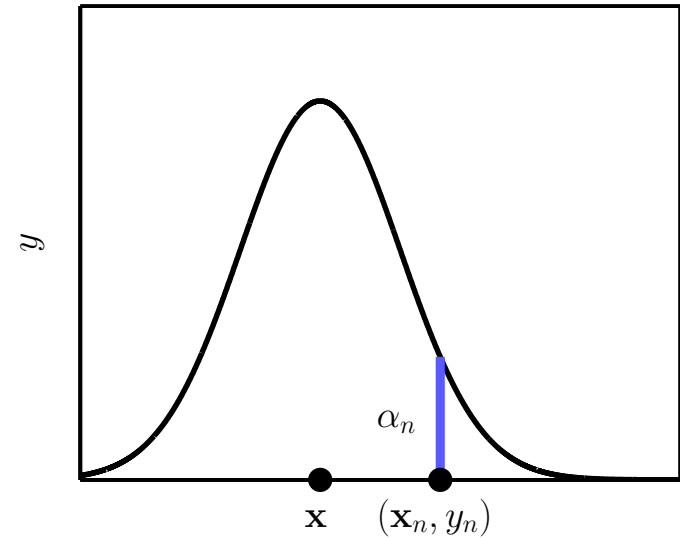
$$\alpha_n(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



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

Nonparametric RBF – Logistic Regression

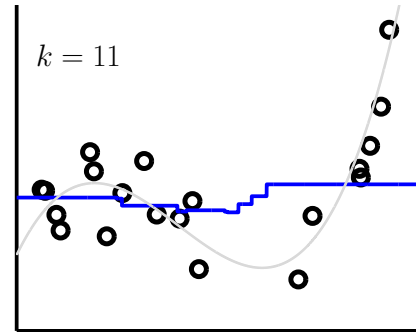
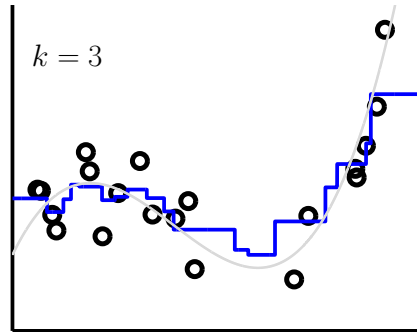
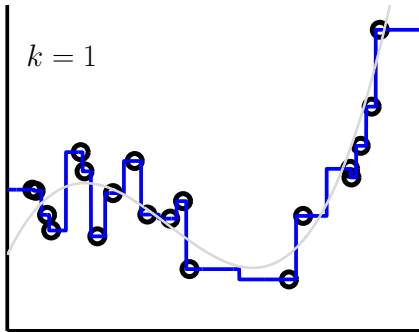
$$\alpha_n(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



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

Choice of Scale r

Nearest Neighbor



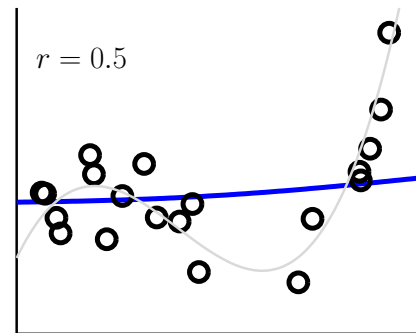
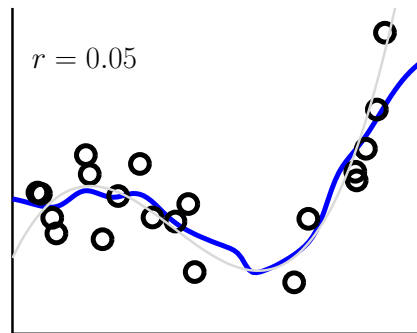
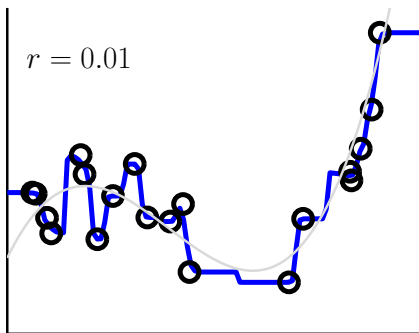
Choosing k :

$$k = 3$$

$$k = \sqrt{N}$$

CV

Nonparametric RBF



Choosing r :

$$r \sim \frac{1}{\sqrt[2d]{N}}$$


CV

overfitting

underfitting

Highlights of Nonparametric RBF

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

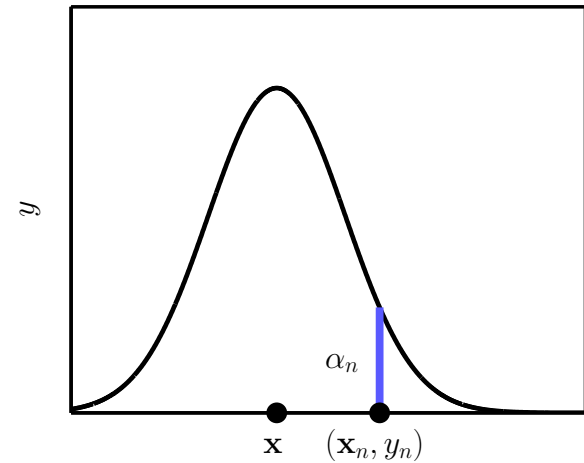


A **good!** method

Scaled Bumps on Each Data Point

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

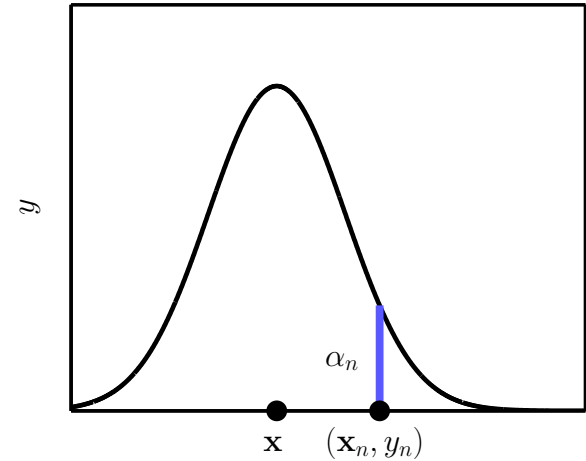
Weighted average of y_n



Scaled Bumps on Each Data Point

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

Weighted average of y_n

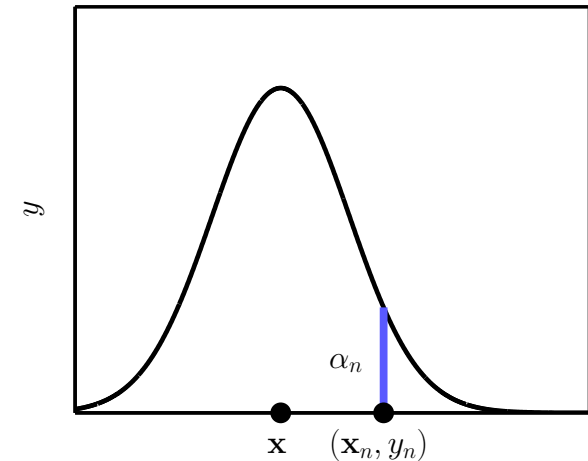


$$g(\mathbf{x}) = \sum_{n=1}^N \left(\frac{y_n}{\sum_{m=1}^N \alpha_m(\mathbf{x})} \right) \cdot \phi \left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r} \right)$$

Scaled Bumps on Each Data Point

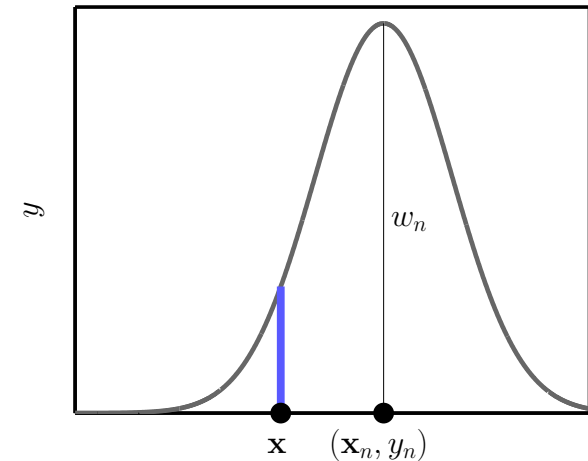
$$g(\mathbf{x}) = \sum_{n=1}^N \left(\frac{\alpha_n(\mathbf{x})}{\sum_{m=1}^N \alpha_m(\mathbf{x})} \right) \cdot y_n$$

Weighted average of y_n



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

$$= \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi \left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r} \right)$$



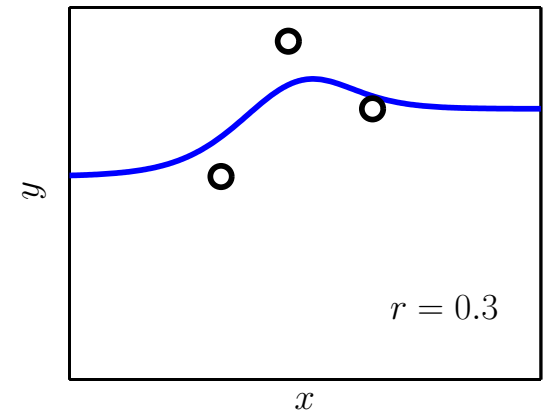
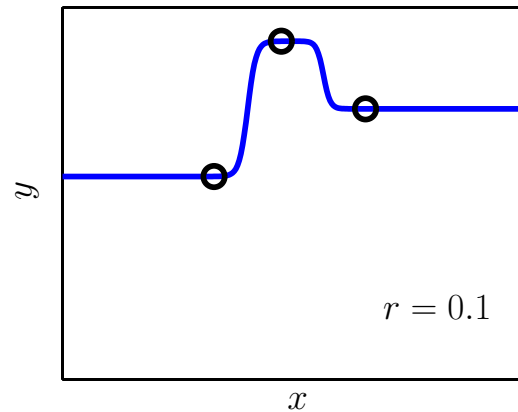
Sum of bumps at \mathbf{x}_n scaled by $w_n(\mathbf{x})$

Nonparametric RBF: $w_n(\mathbf{x})$

Nonparametric RBF

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

Only need to specify r .

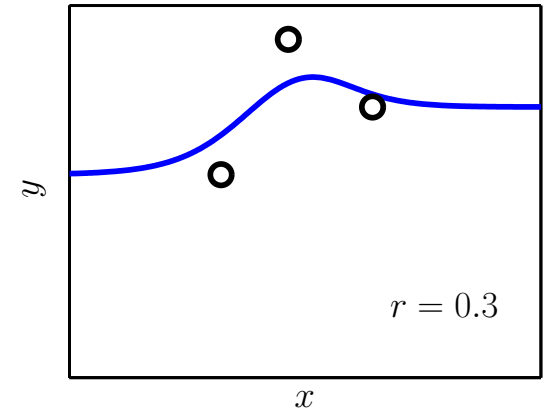
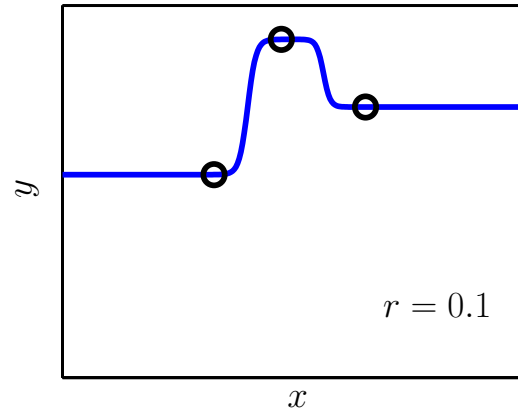


Parametric RBF, w_n – A Linear Model

Nonparametric RBF

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

Only need to specify r .



Parametric RBF

$$h(\mathbf{x}) = \sum_{n=1}^N w_n \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

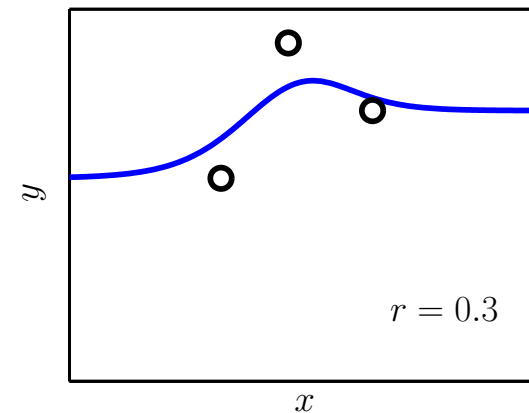
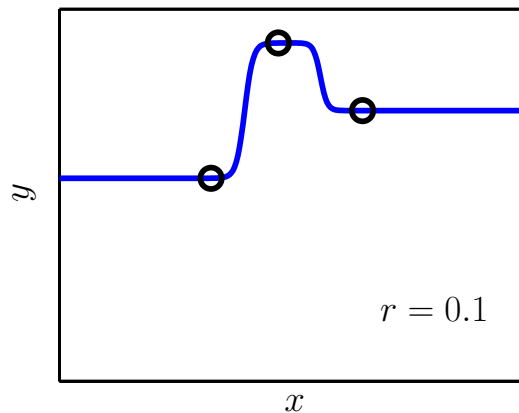
Fix r ; need to determine the **parameters** w_n .
— fit the data.

Parametric RBF – A Linear Model

Nonparametric RBF

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

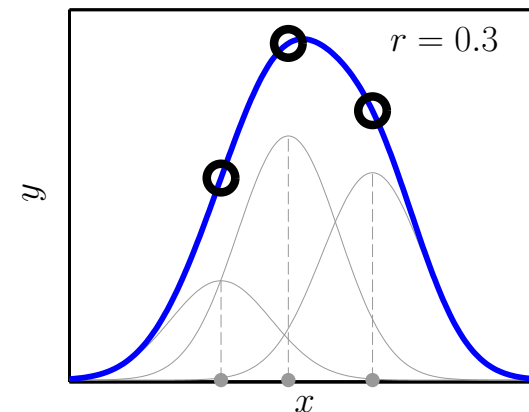
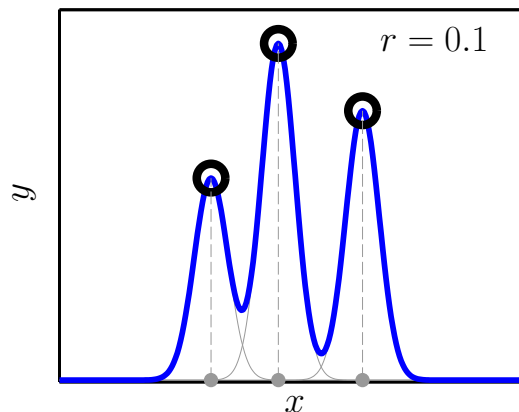
Only need to specify r .



Parametric RBF

$$h(\mathbf{x}) = \sum_{n=1}^N w_n \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

Fix r ; need to determine the parameters w_n .
 — fit the data.
 — **overfit the data?**



RBF-Nonlinear Transform Depends on Data

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

$$\mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_N(\mathbf{x}) \end{bmatrix}, \quad \phi_n(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right). \quad \mathbf{Z} = \begin{bmatrix} - & \mathbf{z}_1^T & - \\ - & \mathbf{z}_2^T & - \\ & \vdots & \\ - & \mathbf{z}_N^T & - \end{bmatrix} = \begin{bmatrix} - & \mathbf{\Phi}(\mathbf{x}_1)^T & - \\ - & \mathbf{\Phi}(\mathbf{x}_2)^T & - \\ & \vdots & \\ - & \mathbf{\Phi}(\mathbf{x}_N)^T & - \end{bmatrix}$$

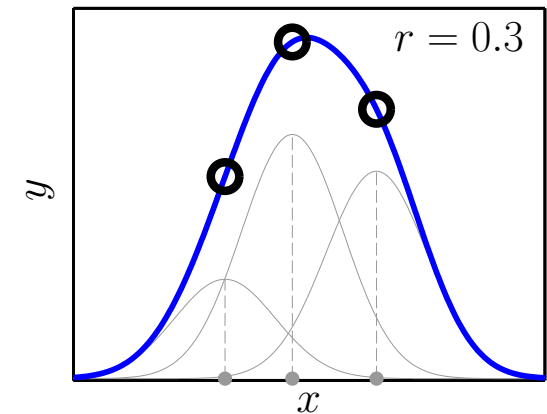
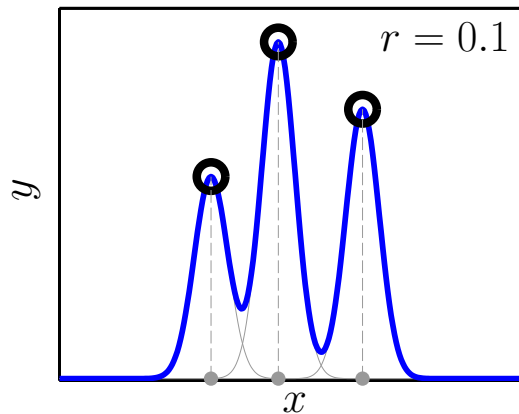
RBF-Nonlinear Transform Depends on Data

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

$$\mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_N(\mathbf{x}) \end{bmatrix}, \quad \phi_n(\mathbf{x}) = \phi \left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r} \right), \quad \mathbf{Z} = \begin{bmatrix} - & \mathbf{z}_1^T & - \\ - & \mathbf{z}_2^T & - \\ & \vdots & \\ - & \mathbf{z}_N^T & - \end{bmatrix} = \begin{bmatrix} - & \mathbf{\Phi}(\mathbf{x}_1)^T & - \\ - & \mathbf{\Phi}(\mathbf{x}_2)^T & - \\ & \vdots & \\ - & \mathbf{\Phi}(\mathbf{x}_N)^T & - \end{bmatrix}$$

Fit the data ($h(\mathbf{x}_n) = y_n$):

$$\mathbf{w} = \mathbf{Z}^\dagger \mathbf{y} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y}$$



Reducing the Number of Bumps: Nonparametric

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

Reducing the Number of Bumps: Parametric

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



$$h(\mathbf{x}) = \sum_{n=1}^N w_n \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$

Reducing the Number of Bumps: k -RBF-Network

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



$$h(\mathbf{x}) = \sum_{n=1}^N w_n \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



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

nonlinear in μ_j

$$= \mathbf{w}^T \Phi(\mathbf{x})$$

$$\Phi(\mathbf{x})^T = [1, \Phi_1(\mathbf{x}), \dots, \Phi_k(\mathbf{x})], \text{ where } \Phi_j(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \mu_j\|}{r}\right).$$

Reducing the Number of Bumps: k -RBF-Network

$$g(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x}) \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



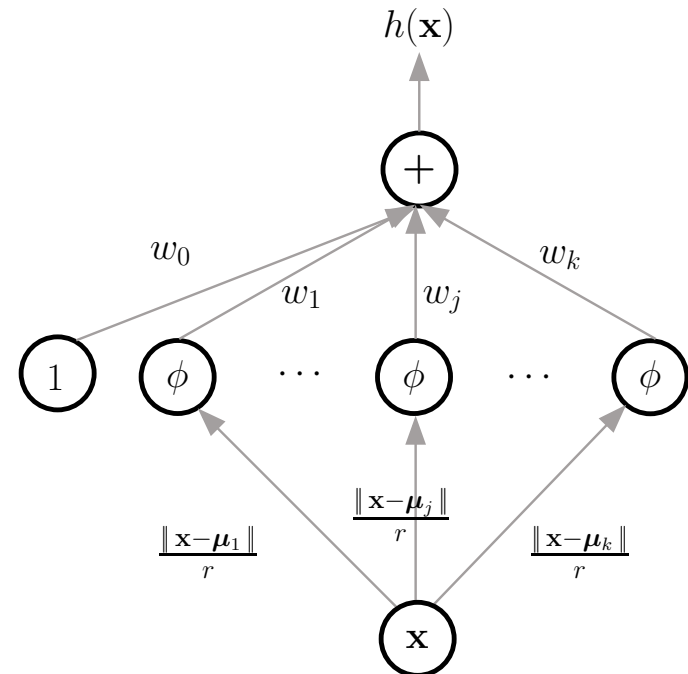
$$h(\mathbf{x}) = \sum_{n=1}^N w_n \cdot \phi\left(\frac{\|\mathbf{x} - \mathbf{x}_n\|}{r}\right)$$



$$h(\mathbf{x}) = w_0 + \sum_{j=1}^k w_j \cdot \phi\left(\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|}{r}\right)$$

nonlinear in $\boldsymbol{\mu}_j$

$$= \mathbf{w}^T \boldsymbol{\Phi}(\mathbf{x})$$



$$\boldsymbol{\Phi}(\mathbf{x})^T = [1, \Phi_1(\mathbf{x}), \dots, \Phi_k(\mathbf{x})], \text{ where } \Phi_j(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|}{r}\right).$$

Fitting the Data

Before: bumps were centered on \mathbf{x}_n — no choice

Now: we may choose the bump centers $\boldsymbol{\mu}_j$

Choose them to ‘cover’ the data

As the centers of k ‘clusters’

Given the bump centers, **we have a linear model to determine the w_j**

That’s ‘easy’, we know how to do that.

Fitting the Data

Fitting the RBF-network to the data (given k, r):

- 1: Use the inputs X to determine k centers $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$.
- 2: Compute the $N \times (k + 1)$ feature matrix Z

$$Z = \begin{bmatrix} - & \mathbf{z}_1^T & - \\ - & \mathbf{z}_2^T & - \\ & \vdots & \\ - & \mathbf{z}_N^T & - \end{bmatrix} = \begin{bmatrix} - & \Phi(\mathbf{x}_1)^T & - \\ - & \Phi(\mathbf{x}_2)^T & - \\ & \vdots & \\ - & \Phi(\mathbf{x}_N)^T & - \end{bmatrix}, \text{ where } \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}) \\ \vdots \\ \phi_k(\mathbf{x}) \end{bmatrix}, \phi_j(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|}{r}\right)$$

Each row of Z is the RBF-feature corresponding to \mathbf{x}_n (with dummy bias coordinate 1).

- 3: Fit the *linear* model $Z\mathbf{w}$ to \mathbf{y} to determine the weights \mathbf{w}^* .
 - classification: PLA, pocket, linear programming,
 - regression: pseudoinverse.
 - logistic regression: gradient descent on cross entropy error.

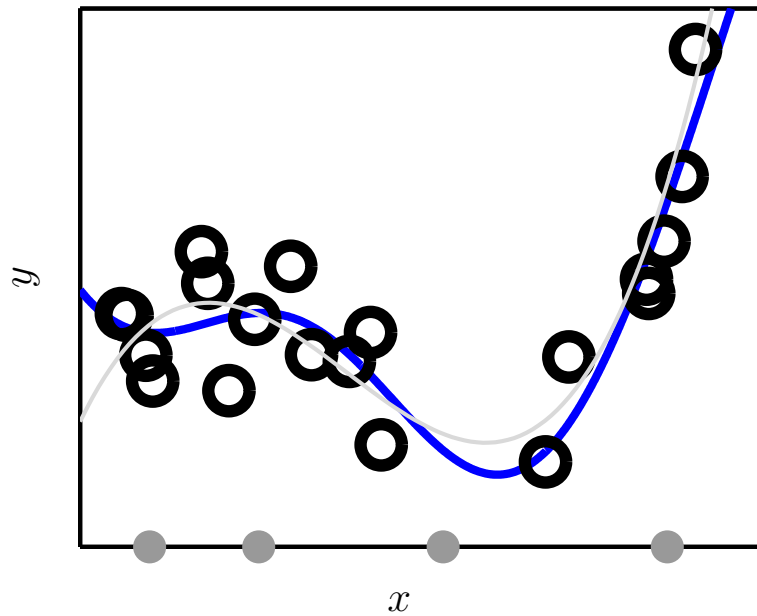
Choose r using CV, or (a heuristic):

$$r \sim \frac{\text{radius of data}}{k^{1/d}}$$

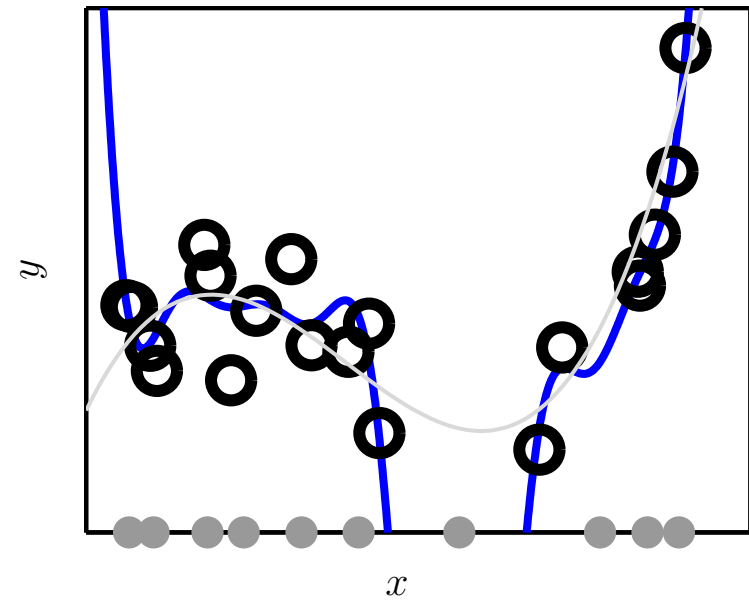
(so your clusters ‘cover’ the data)

Our Example

$$k = 4, r = \frac{1}{k}$$



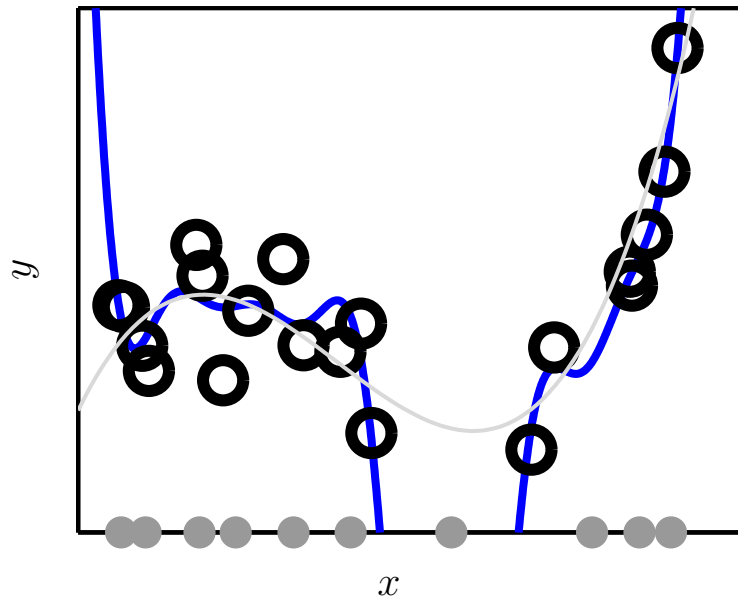
$$k = 10, r = \frac{1}{k}$$



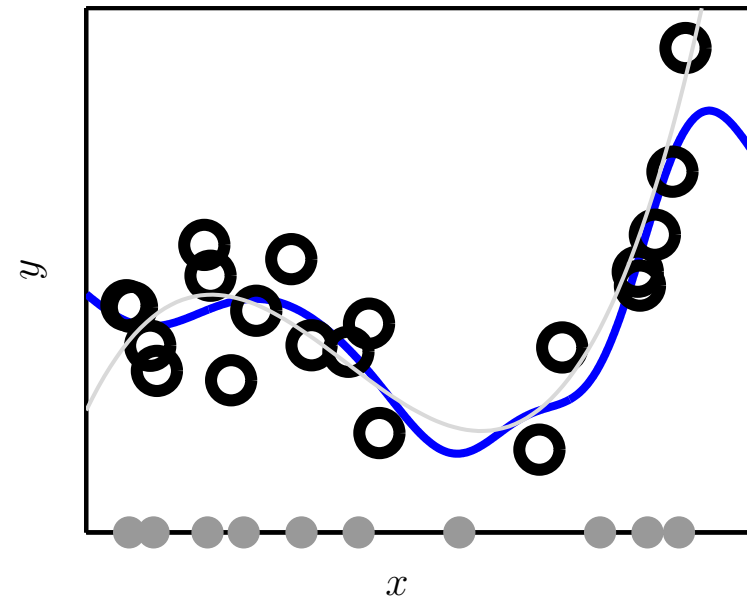
$$\mathbf{w} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y}$$

Use Regularization to Fight Overfitting

$$k = 10, r = \frac{1}{k}$$



$$k = 10, r = \frac{1}{k}, \text{regularized}$$



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

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

A Peek at Unsupervised Learning

21-NN rule, 10 Classes

10 Clustering of Data

