Machine Learning from Data

Lecture 18: Spring 2021

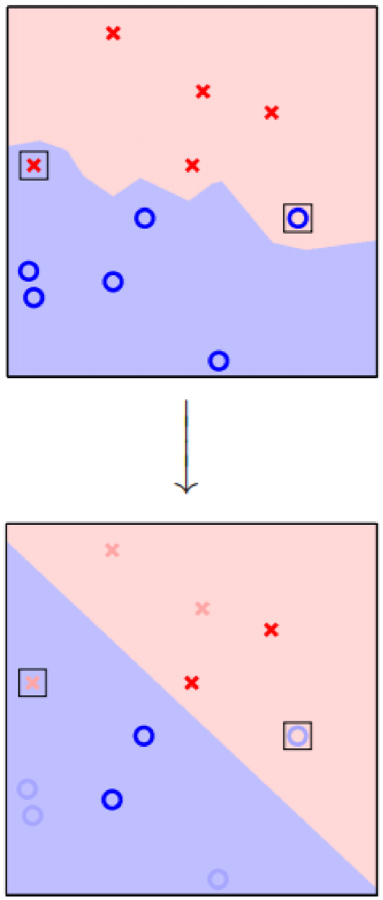
Today's Lecture

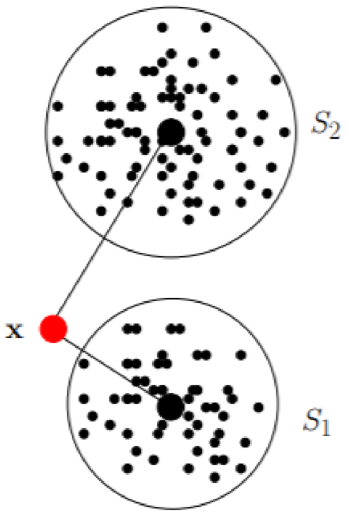
- Radial basis Functions (RBF)
- Non-Parametric RBF
- Parametric RBF
- K-RBF-Network

RECAP: **Data Condensation and Nearest Neighbor Search**

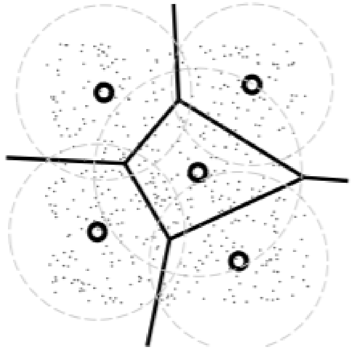
CNN

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.

Non-parametric RBFs

Regression

KNN: $g(x) = \frac{1}{K} \sum_{i=1}^{K} y_{[i]}$

or $g(x) = \sum_{i=1}^{K} \underline{\underline{\frac{1}{K}}}\, y_{[i]}$

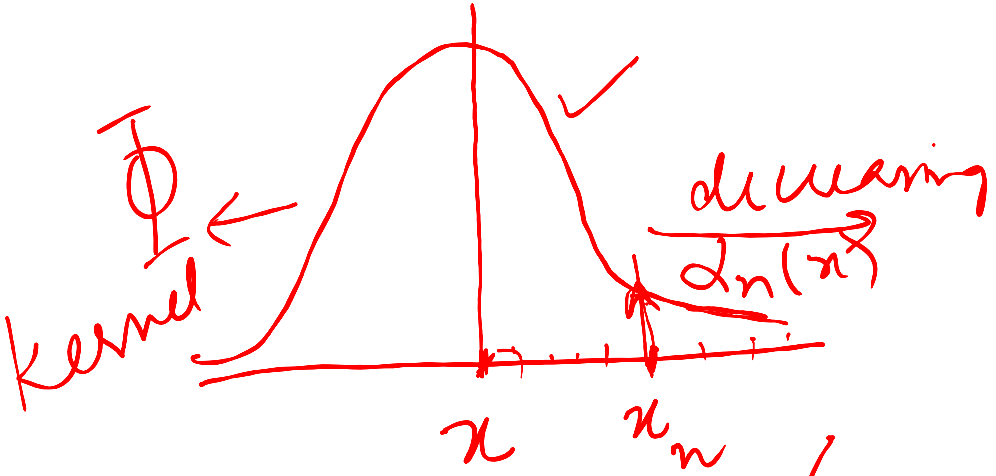
RBF

$$g(x) = \sum_{n=1}^{N} \underline{\underline{\alpha_n(x)}}$$

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

all the data

normalized to 1



$$\therefore \alpha_n(x) = \Phi\left(\underbrace{\|x - x_n\|}\right)$$

Units

$\gamma \rightarrow$ scale parameter.

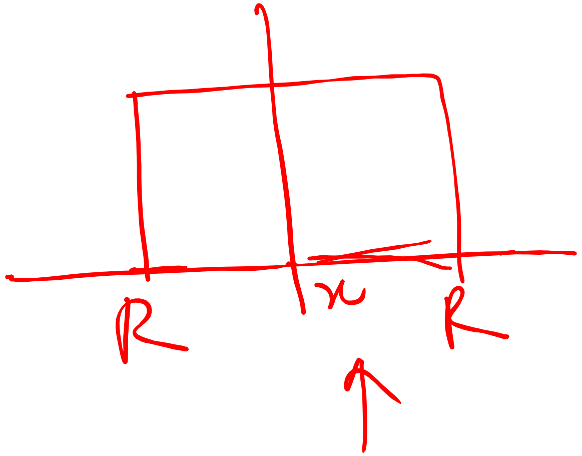
$$\alpha_n(x) = \Phi\left(\frac{\|x - x_n\|}{\gamma}\right)$$

$$\Phi(s) = \frac{1}{1 + s^2}$$

(Decreasing function)

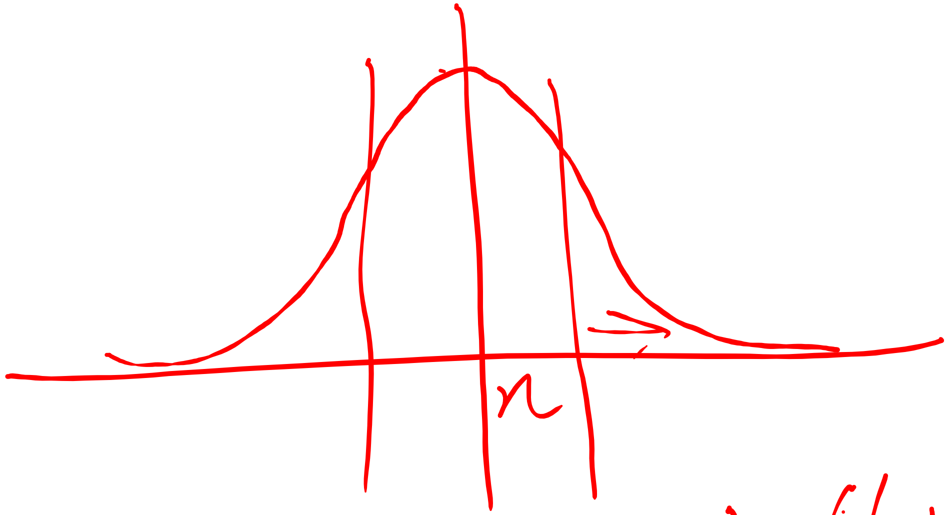
1) Window Kernel

$$\phi(s) = \begin{cases} 1 & \text{if } |s| \le R \\ 0 & \text{if } |s| > R \end{cases}$$



2) Gaussian Kernel

$$\phi(s) = e^{-s^2/2}$$



$$g(x) = \sum_{n=1}^{N} \frac{\alpha_n(x)}{\sum_{m=1}^{M} \alpha_m(x)}$$

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

Classification

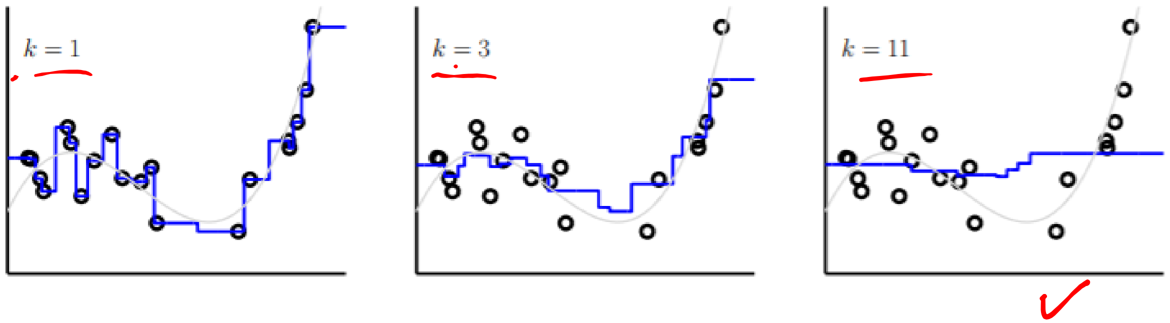
$$g(x) = \text{sign}\left(\sum_{n=1}^{N} \frac{\alpha_n(x)}{\sum_{m=1}^{M} \alpha_m(x)} \right) \to X$$

Logistic Regression

$$g(x) = X\,[[y_n = +1]]$$

Choice of Scale r

Nearest Neighbor



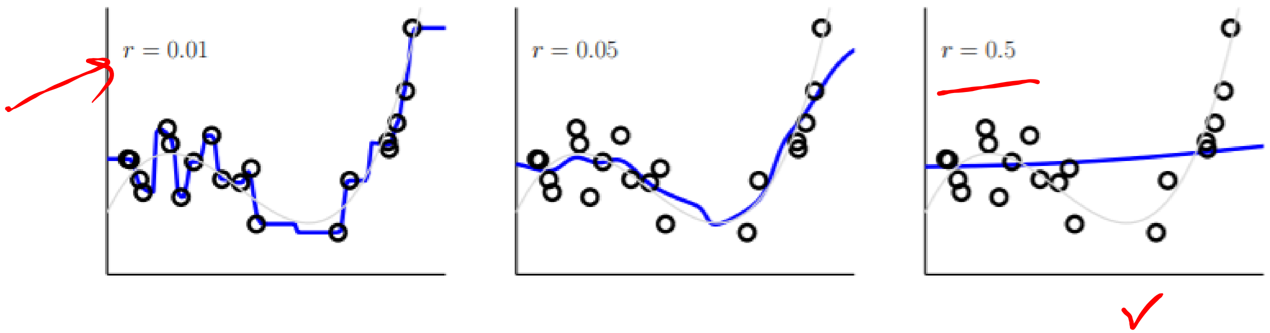
Choosing k:

$k = 3$

$\boxed{k = \sqrt{N}}$

CV

Nonparametric RBF



overfitting

underfitting

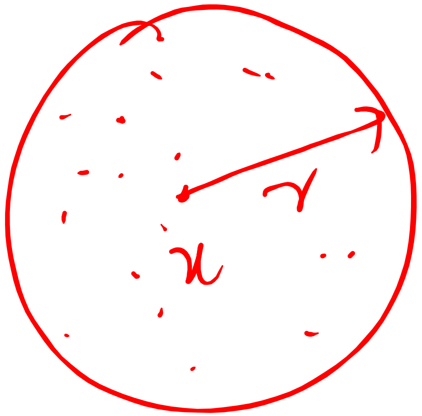
Choosing r:

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

CV

$$\frac{\|x - x_n\|}{r} \to 0$$

$k = \sqrt{N}$



$Vol \propto r^d$

$\#$ of data points $\propto \underline{N r^d}$

$\cancel{Nd}$ $\quad N r^d = \sqrt{N}$

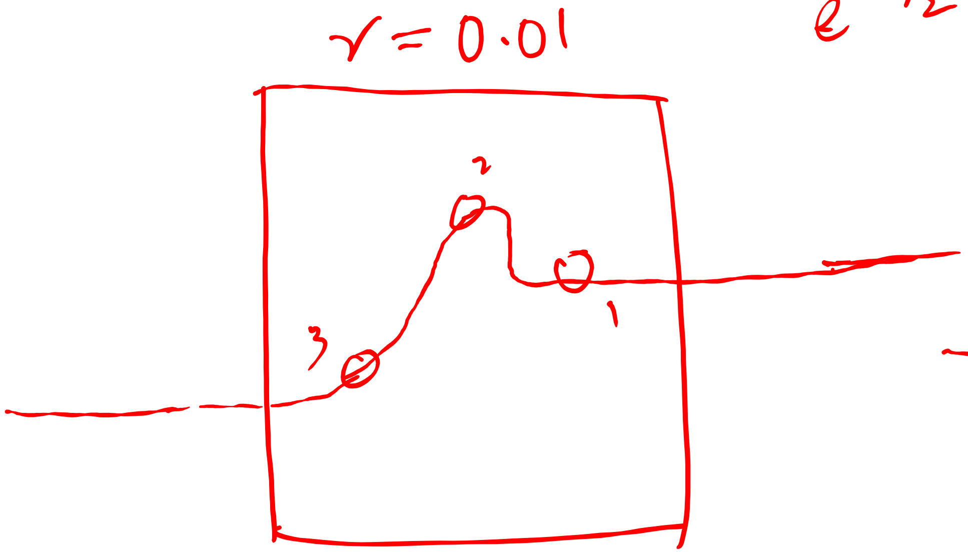
$$\boxed{r = \frac{1}{2\sqrt[d]{N}}}$$

(Recommendation)

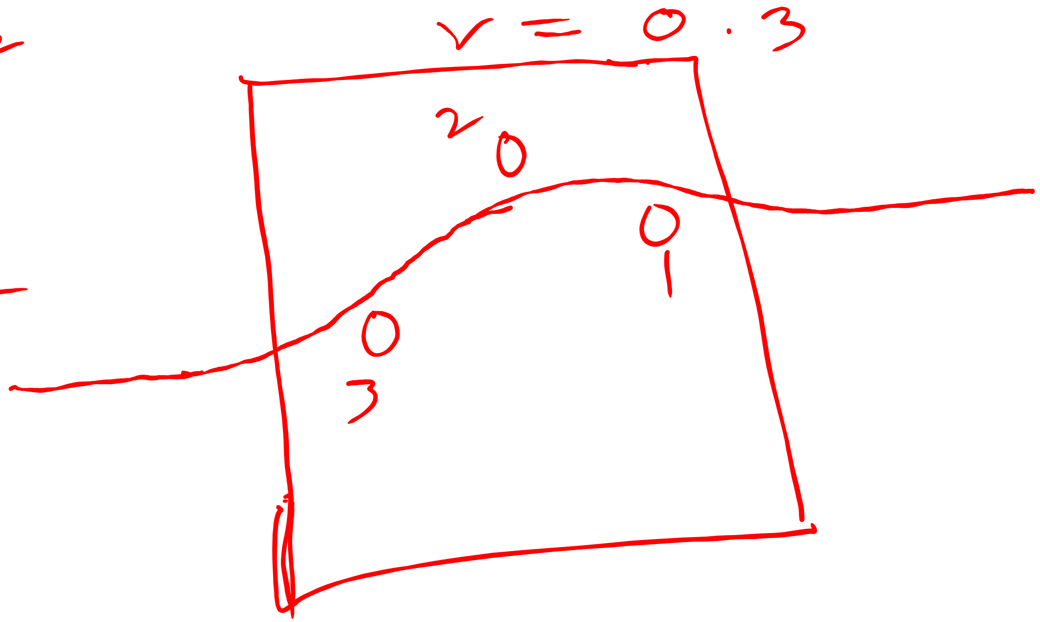
Highlights of Nonparametric RBF

1. Simple ('smooth' version of k-NN rule).
2. No training.
3. Near optimal E_{out}. $\leftarrow$ $K/N \to 0$
4. Easy to justify classification to customer.
5. Can do classification, multi-class, regression, logistic regression.
6. **Computationally demanding**.

A **good!** method



$e^{-s^2/2}$



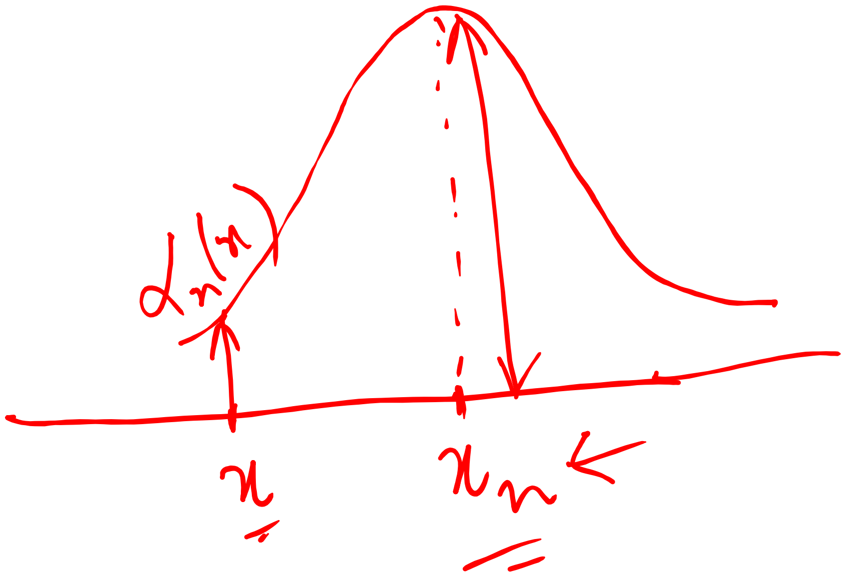
$$g(x) = \sum_{n=1}^{N} \left[\frac{\alpha_n(x)}{\sum_{m=1}^{M} \alpha_m(x)} \right] y_n$$

PARAMETRIC RBF

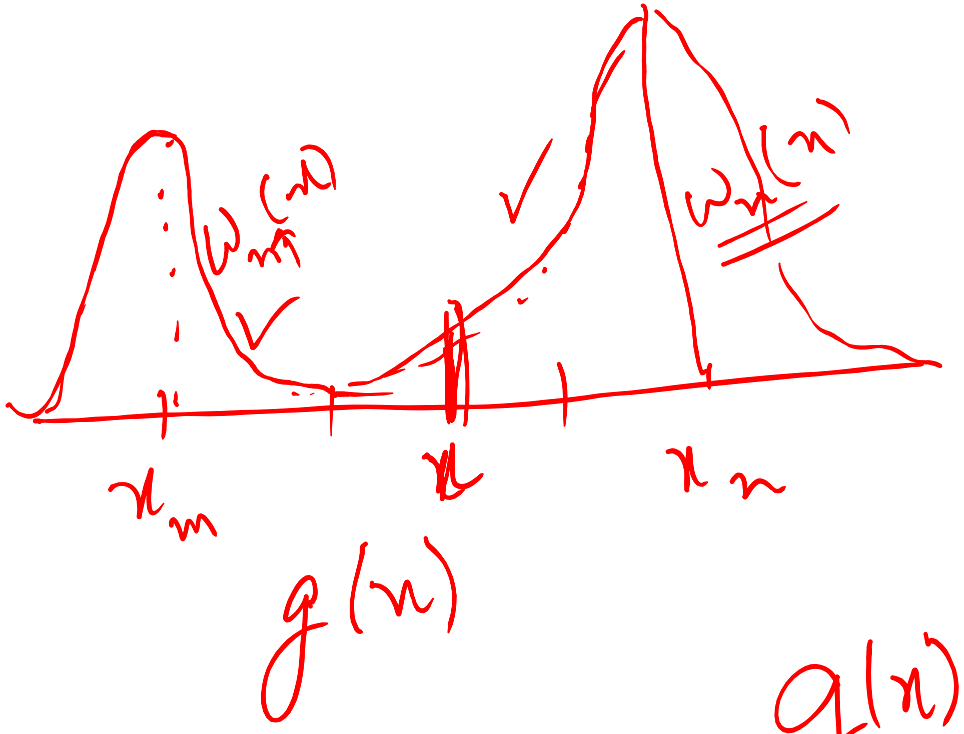
$$g(x) = \sum_{n=1}^{N} \underbrace{\frac{y_n}{\sum_{m=1}^{M} \alpha_m(x)}}_{\to\ \omega_n(x)} \cdot \phi\left(\frac{\|x - x_n\|}{r}\right)$$

→ symmetric

Earlier x as the center



$$g(x) = \sum_{n=1}^{N} \omega_n(x)\, \phi\left(\frac{\|x - x_n\|}{r}\right)$$

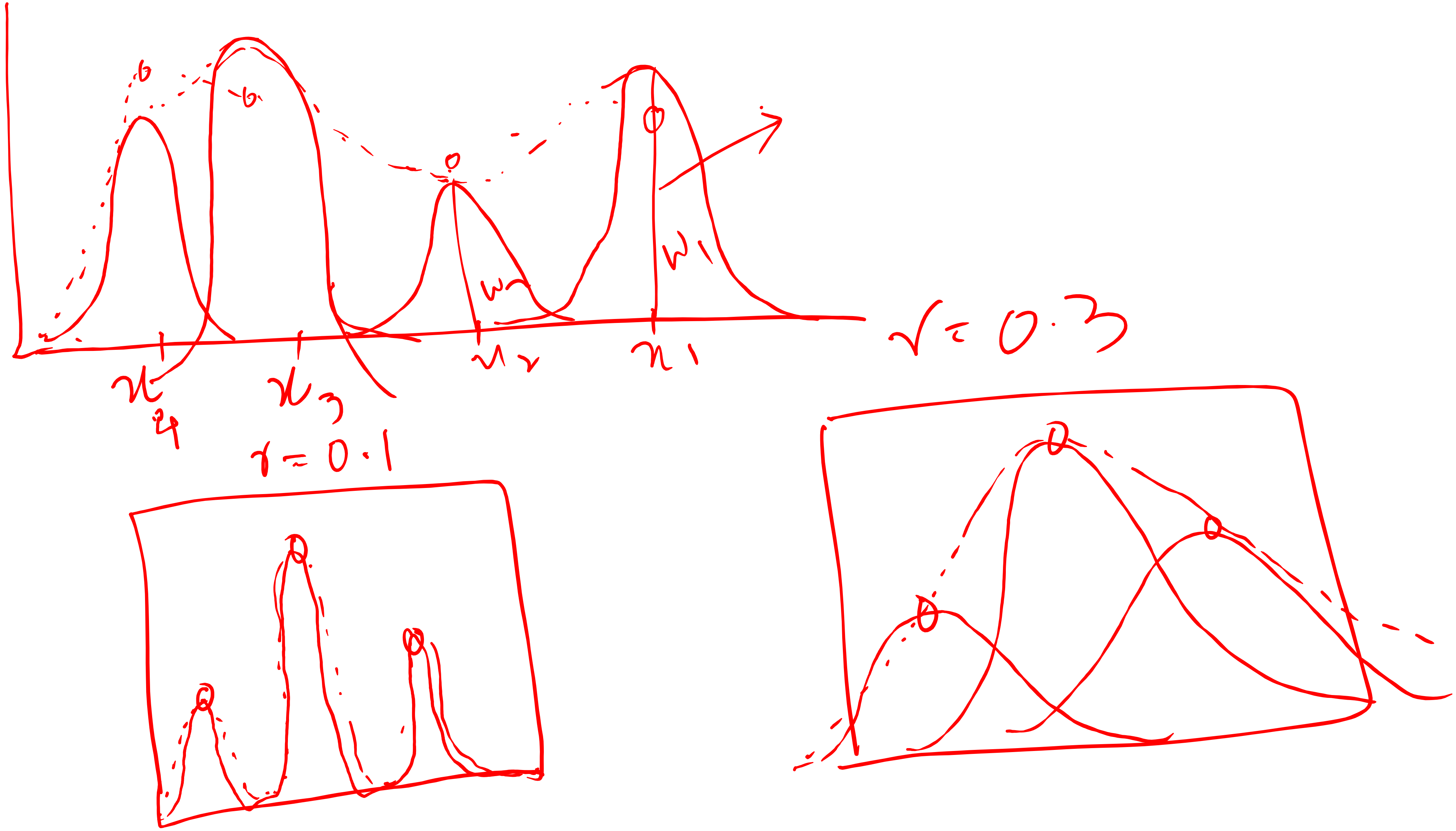


Chasing curve → every data point.

Approximation

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

PARAMETRIC RBF.


x4
x3
x2
x1
w2
w1
γ = 0.1
γ = 0.3

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

parameters

we want

$$g(x_1) = \sum_{n=1}^{N} w_n \, \phi\left(\frac{\|x_1 - x_n\|}{r}\right) = y_1$$

$$\vdots$$

$$g(x_N) = \sum_{n=1}^{N} w_n \, \phi\left(\frac{\|x_N - x_n\|}{r}\right) = y_N$$

N eqns & N unknowns.

$x \longrightarrow Z$ ($\underline{N}$ dimensional feature transform).

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} = \begin{bmatrix} \phi\left(\frac{\|x - x_1\|}{r}\right) \\ \vdots \\ \phi\left(\frac{\|x - x_N\|}{r}\right) \end{bmatrix}$$

Similarity features.

$\rightarrow$ Depend on the data.

$$Z = \begin{bmatrix} — z_1^T — \\ — z_2^T — \\ \vdots \\ — z_N^T — \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

data matrix $\leftarrow Zw \simeq y \rightarrow$ target vector.

$\rightarrow$ N X N matrix

<u>Regression</u>

$$w_{lin} = Z^{\dagger} y = (Z^T Z)^{-1} Z^T y$$

Classification $\rightarrow$ PLA $\rightarrow$ Pocket Algo.

Logistic Regression $\longrightarrow$ Gradient Descent.

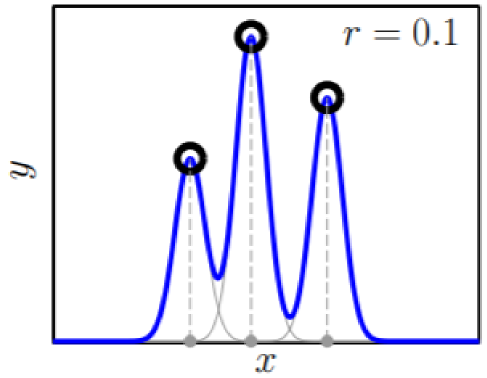
Lin. Model with similarity features.

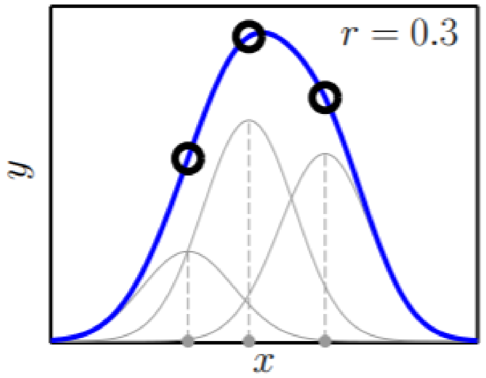
$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \cdot \phi\left(\frac{\| \mathbf{x} - \mathbf{x}_n \|}{r}\right) = \mathbf{w}^{\mathrm{T}}\mathbf{z}$$

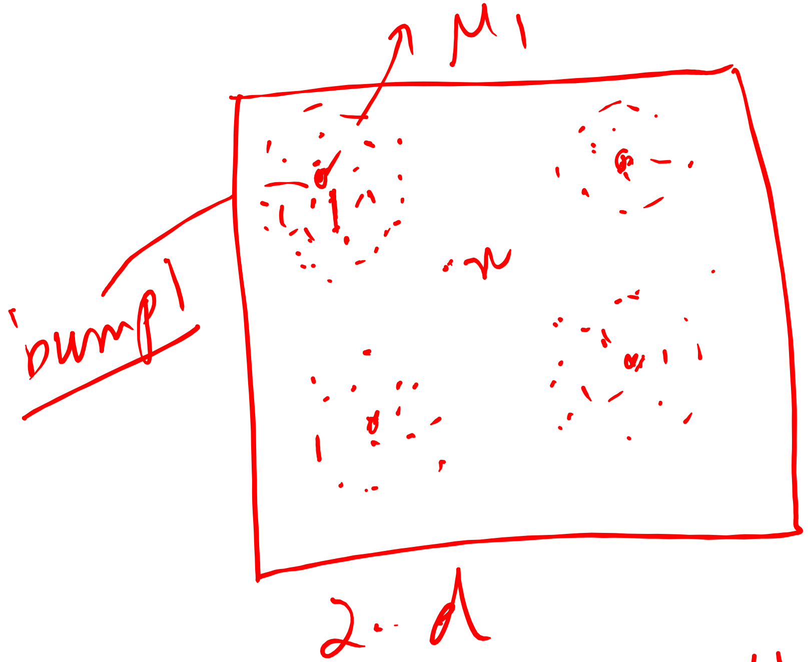
$$\mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_N(\mathbf{x}) \end{bmatrix}, \quad \phi_n(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x}-\mathbf{x}_n\|}{r}\right). \qquad \mathrm{Z} = \begin{bmatrix} — & \mathbf{z}_1^{\mathrm{T}} & — \\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

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

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







4 bumps/peaks

$\downarrow$

N peaks

Reduce the no. of parameter.

$\rightarrow$ Decide # of bumps $\therefore$ K with centers $\{\mu_1, \mu_2 \cdots \mu_k\}$

$\rightarrow$ Center must cover the data.

$$g(x) = \omega_0 + \sum_{j=1}^{K} \omega_j \, \phi\left(\frac{\|x-\mu_j\|}{r}\right)$$

$$x \rightarrow z = \begin{bmatrix} \phi\left(\frac{\|x-\mu_1\|}{r}\right) \\ \vdots \\ \phi\left(\frac{\|x-\mu_k\|}{r}\right) \end{bmatrix}$$

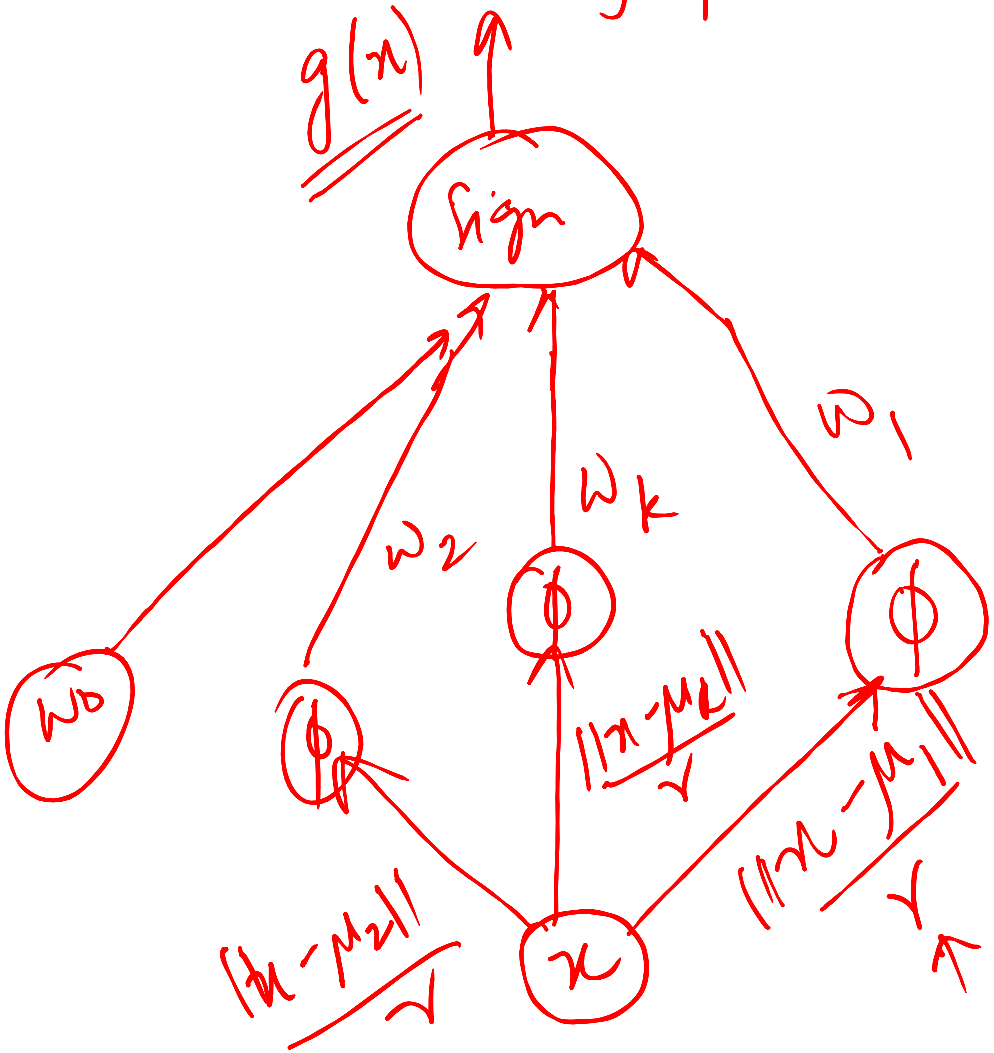
Reduced ✓

$\omega^T z$

After solving μ_j's

$$g(x) = \underline{\underline{w_0}} + \sum_{j=1}^{K} w_j \, \phi\left(\underline{\|x-\mu_j\|}\right)$$

similarity



Radia Basis Function Network

μ's ?

i) Find centers of clusters in the data.

Lloyd's Algorithm. ✓

Not using y values.

→ Unsupervised step.

ii) You have a linear model with w's

PLA, Pseudo Inverse, grad. descent

Picking r values. → k bumps in d dimensions. $r \propto \frac{1}{\sqrt[d]{K}}$

Vol. of data covered $\sim r^d$

K bumps $\sim K r^d$ ←

$K r^d \sim R$

$r \sim \frac{R}{\sqrt[d]{K}}$

Recommendation

Fitting the Data

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

1: Use the inputs X to determine k centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.

2: Compute the $N \times (k+1)$ feature matrix Z

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

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 $\mathbf{w}^*$.

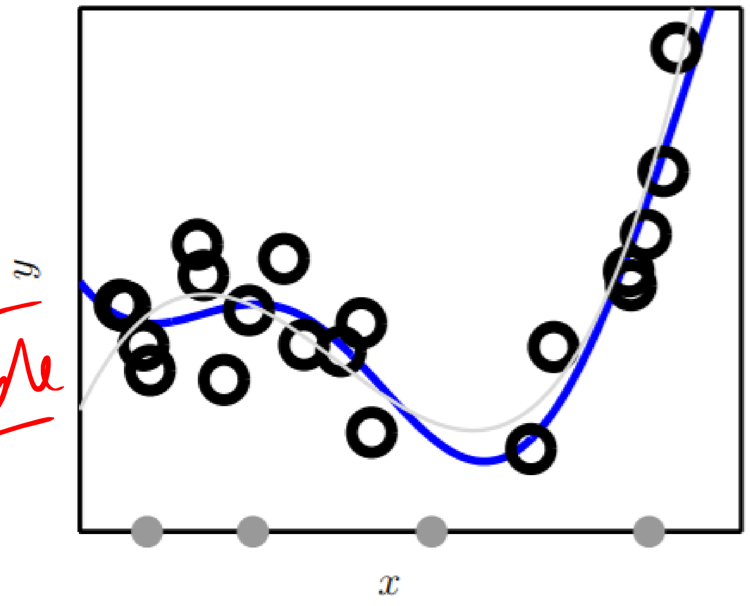
classification: PLA, pocket, linear programming,. . . .

regression: pseudoinverse.

logistic regression: gradient descent on cross entropy error.

CROSS VALIDATION

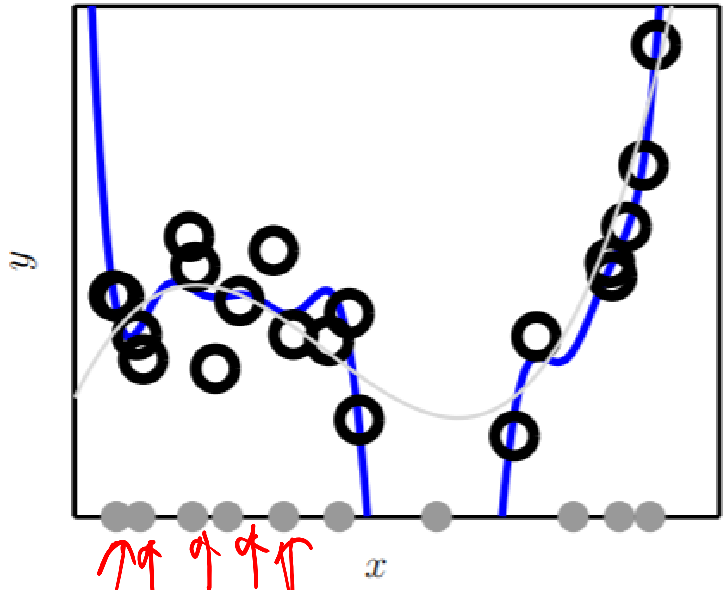
$k = 4, \ r = \frac{1}{k}$



Reasonable

White → target fn.

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



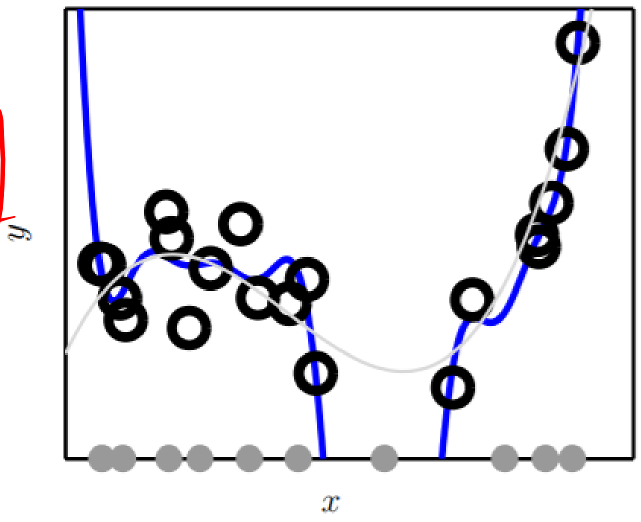
$$\boxed{\mathbf{w} = (\mathrm{Z}^{\mathrm{T}}\mathrm{Z})^{-1}\mathrm{Z}^{\mathrm{T}}\mathbf{y}}$$

Linear Model

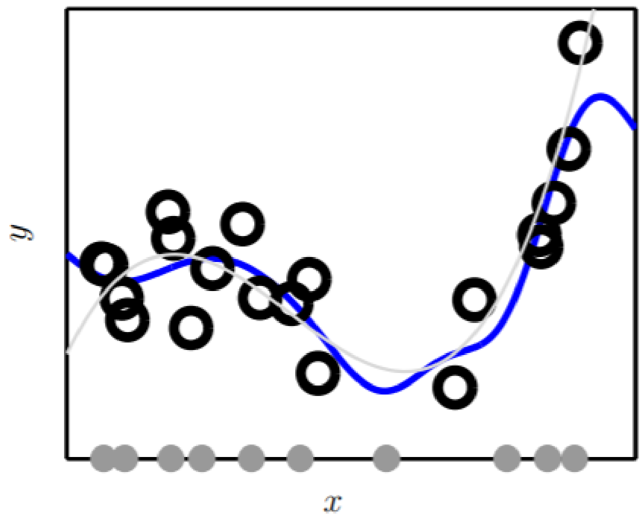
Use Regularization to Fight Overfitting

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

Overfitting



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



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

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 $\boxed{\text{linear models.}}$

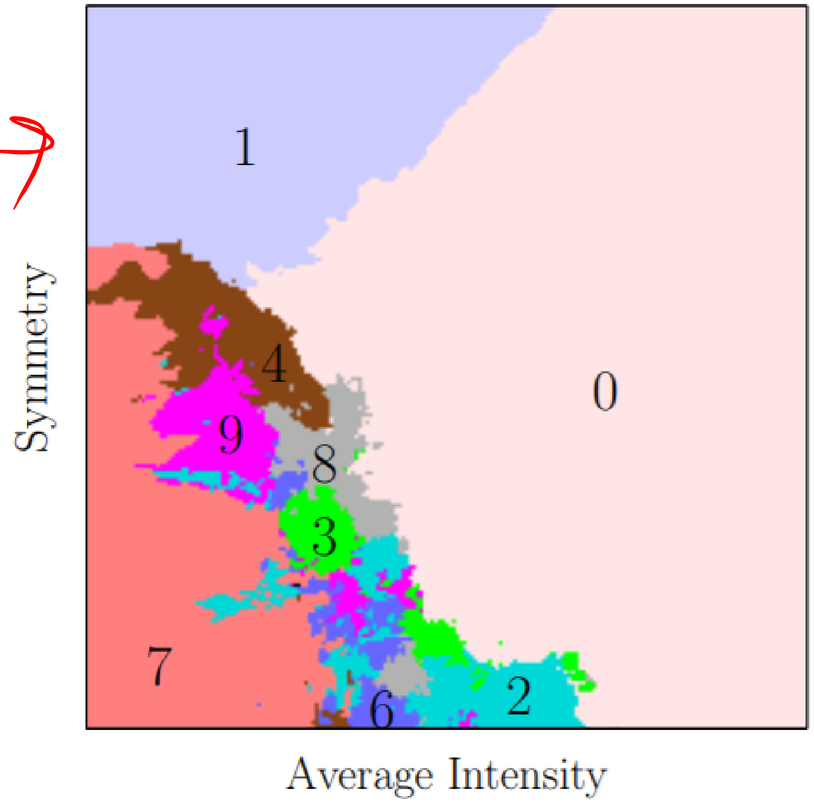
   A linear model with an adaptable nonlinear transform.

4. We used uniform bumps – can have different shapes Σ_j.

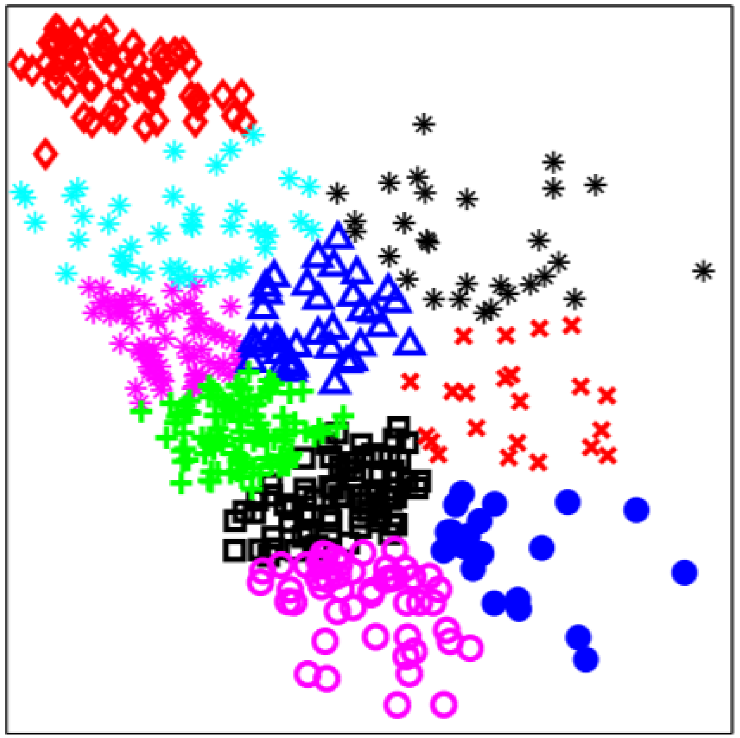
5. **NEXT:** How to better choose the centers: unsupervised learning.

A Peek at Unsupervised Learning

21-NN rule, 10 Classes



10 Clustering of Data



Thanks!