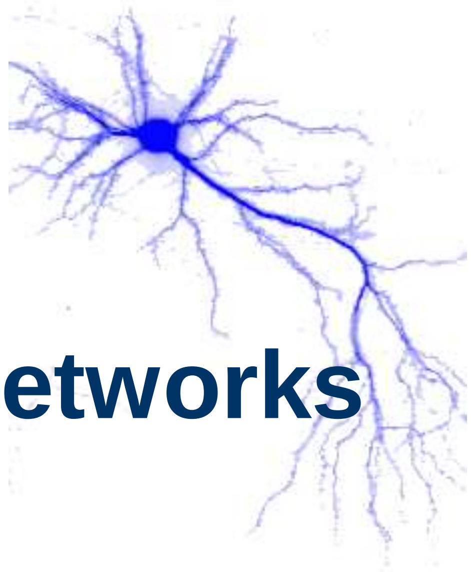
Introduction to
Radial Basis Function Networks



Contents

- Overview
- The Models of Function Approximator
- The Radial Basis Function Networks
- RBFN's for Function Approximation
- The Projection Matrix
- Learning the Kernels
- Bias-Variance Dilemma
- The Effective Number of Parameters
- Model Selection

RBF

- Linear models have been studied in statistics for about 200 years and the theory is applicable to RBF networks which are just one particular type of linear model.

- However, the fashion for neural networks which started in the mid-80 has given rise to new names for concepts already familiar to statisticians

Typical Applications of NN

- Pattern Classification

$$l = f(\mathbf{x}) \qquad \begin{array}{l} \mathbf{x} \in X \subset R^m \\ l \in C \subset N \end{array}$$

- Function Approximation

$$\mathbf{y} = f(\mathbf{x}) \qquad \begin{array}{l} \mathbf{x} \in X \subset R^n \\ \mathbf{y} \in Y \subset R^m \end{array}$$

- Time-Series Forecasting

$$\mathbf{x}(t) = f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \mathbf{x}_{t-3}, \ldots)$$

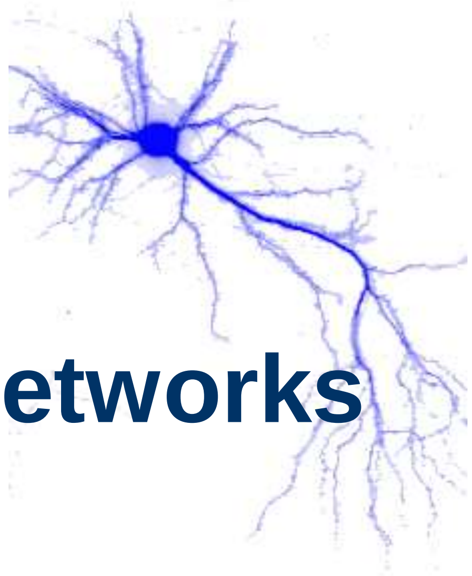
Function Approximation

Unknown $f : X \to Y$

$X \subseteq R^m$  $Y \subseteq R^n$

Approximator $\hat{f} : X \to Y$

$X \subseteq R^m$  $Y \subseteq R^n$



Introduction to Radial Basis Function Networks

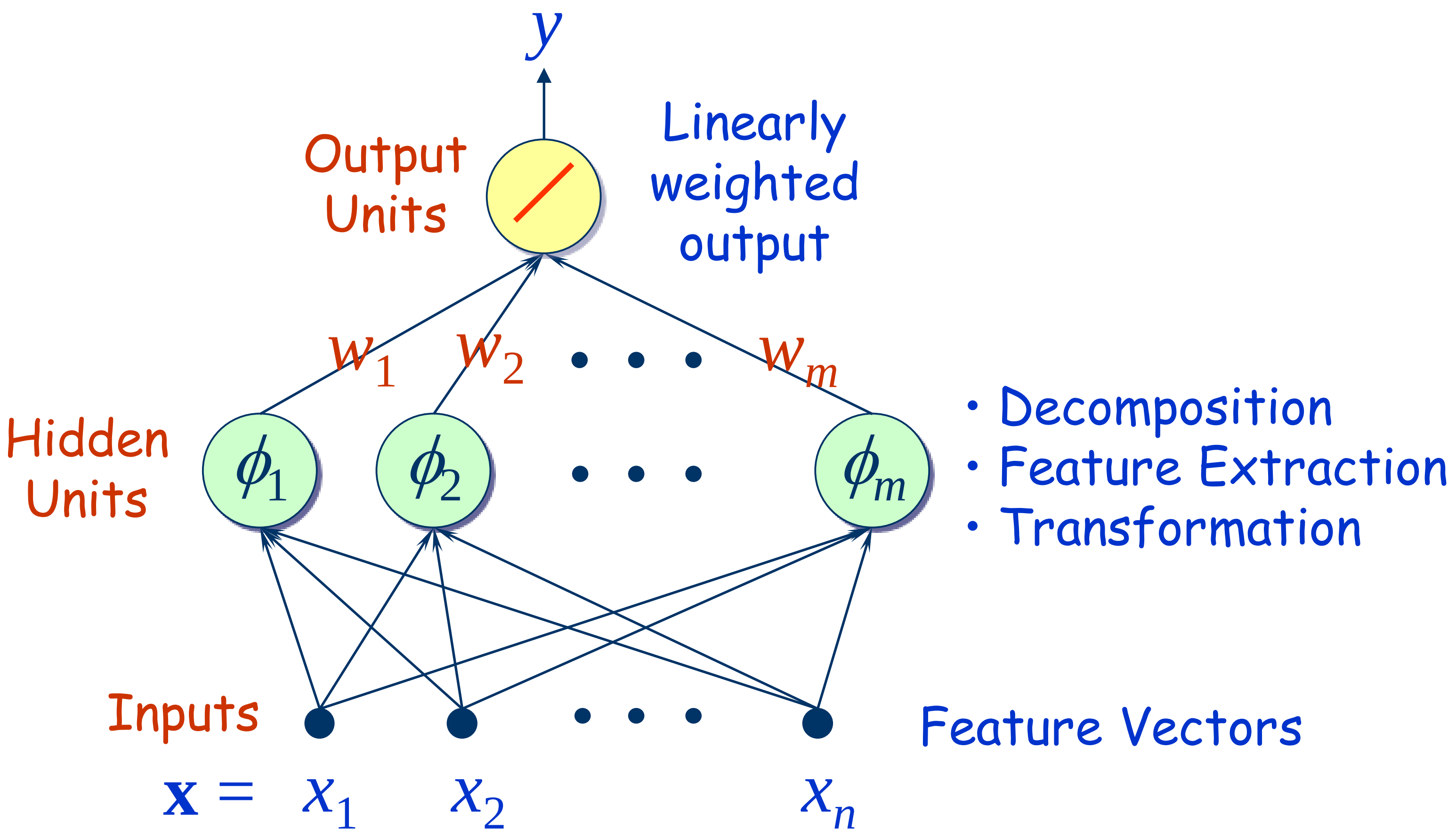
The Model of
Function Approximator

Linear Models

$$f(\mathbf{x}) = \sum_{i=1}^{m} \overbrace{w_i}^{\text{Weights}} \underbrace{\phi_i(\mathbf{x})}_{\text{Fixed Basis Functions}}$$

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

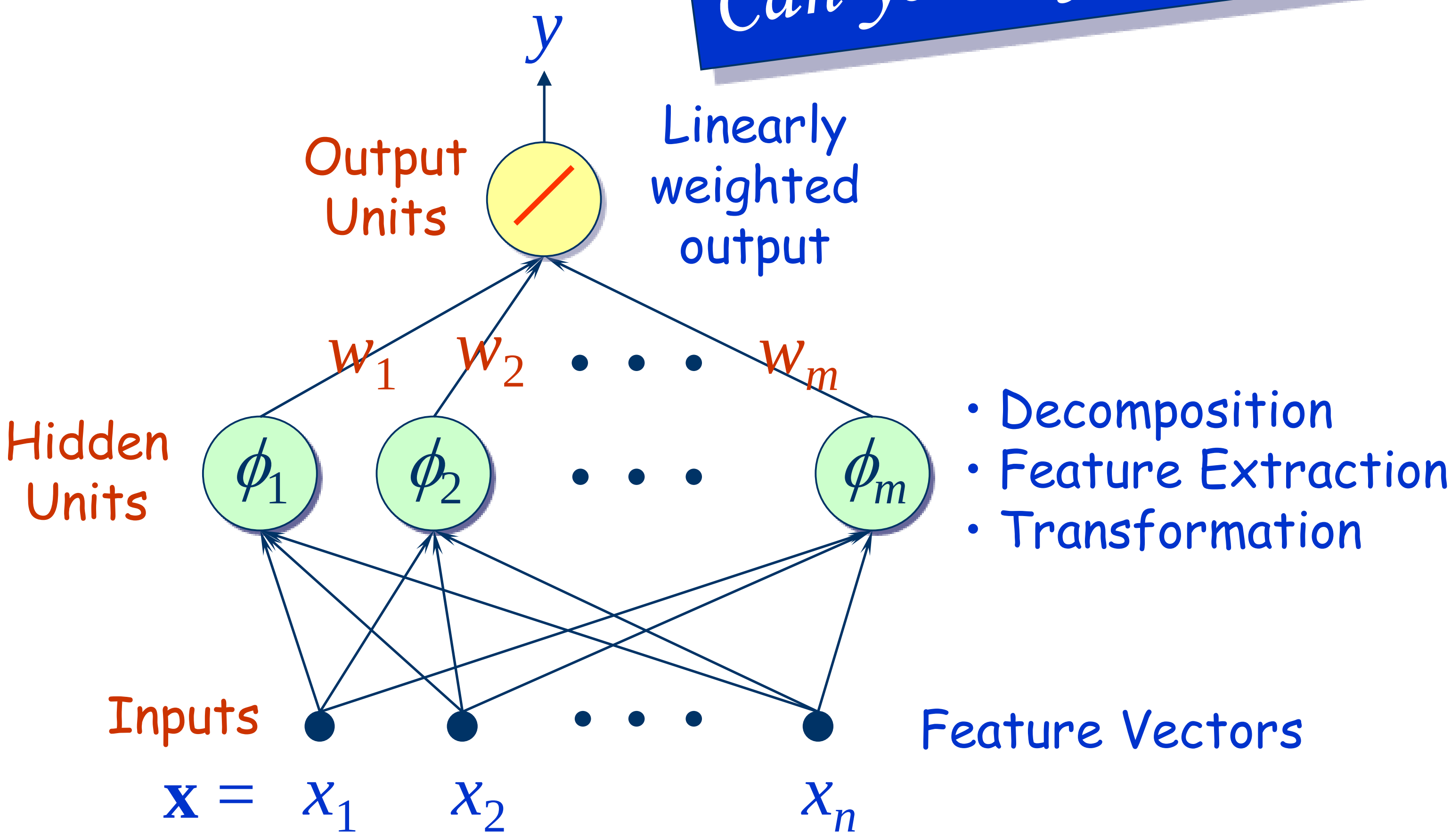
Linear Models



$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Linear Models

Can you say some bases?



$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Example Linear Models

Are they orthogonal bases?

- Polynomial

$$f(x) = \sum_i w_i x^i \qquad \phi_i(x) = x^i, \quad i = 0, 1, 2, \ldots$$

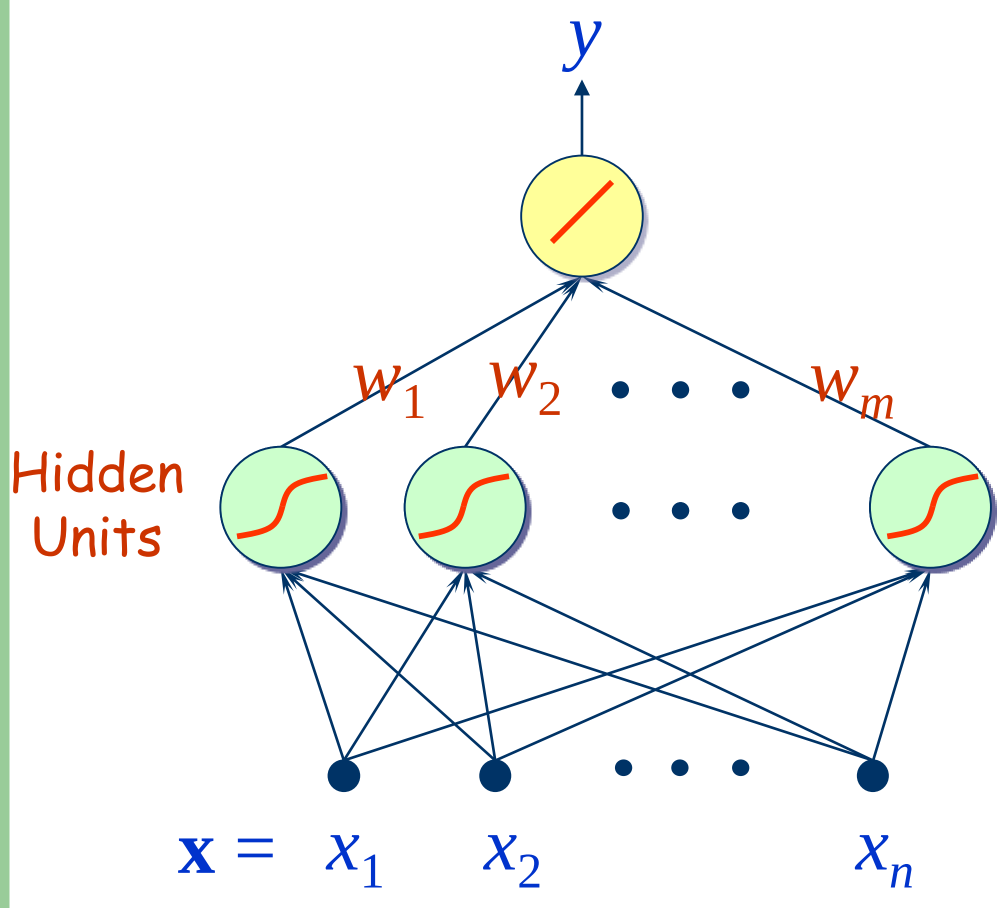
- Fourier Series

$$f(x) = \sum_k w_k \exp\left(j2k\omega_0 x\right)$$

$$\phi_k(x) = \exp\left(j2k\omega_0 x\right), \quad k = 0, 1, 2, \ldots$$

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

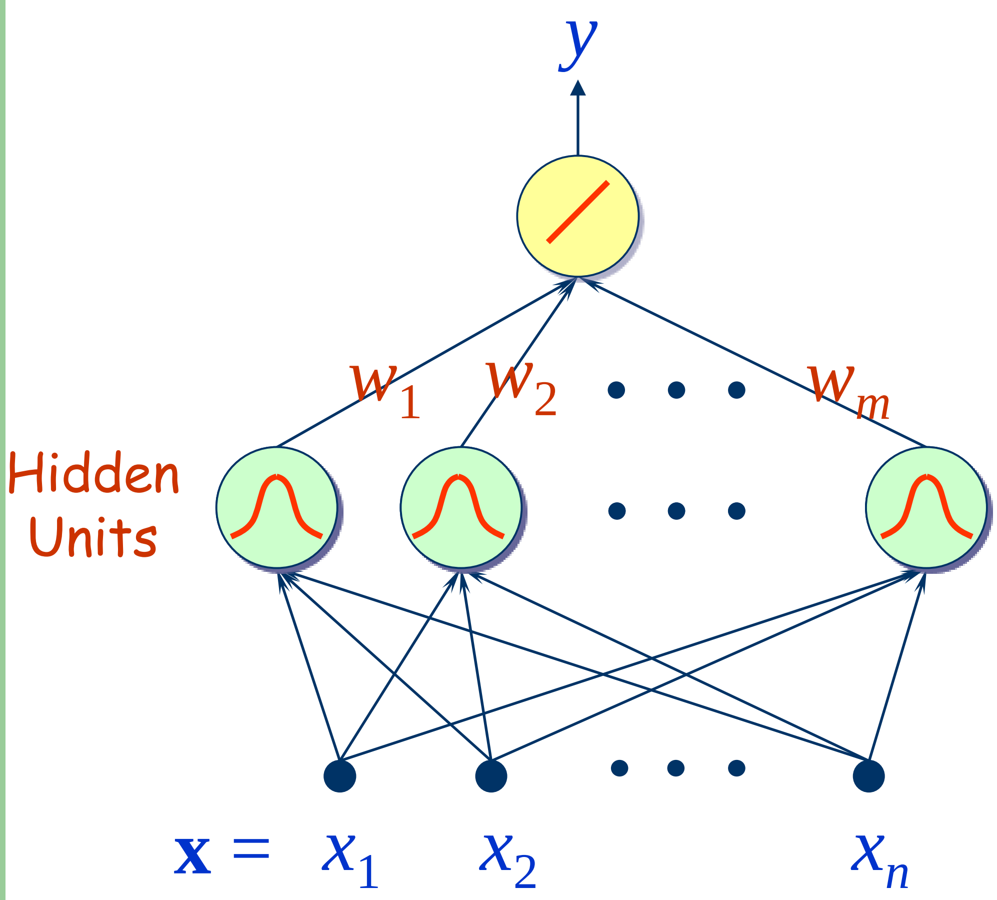
Single-Layer Perceptrons as Universal Aproximators



With sufficient number of sigmoidal units, it can be a universal approximator.

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Radial Basis Function Networks as Universal Aproximators

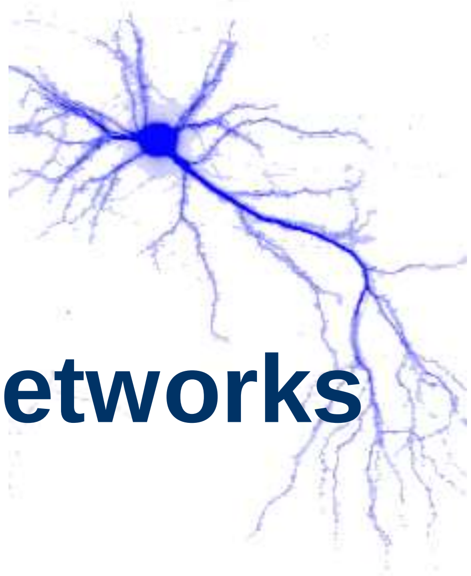


With sufficient number of radial-basis-function units, it can also be a universal approximator.

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Non-Linear Models

$$f(\mathbf{x}) = \sum_{i=1}^{m} \overbrace{w_i}^{\text{Weights}} \underbrace{\phi_i(\mathbf{x})}_{\text{Adjusted by the Learning process}}$$



Introduction to Radial Basis Function Networks

The Radial Basis Function Networks

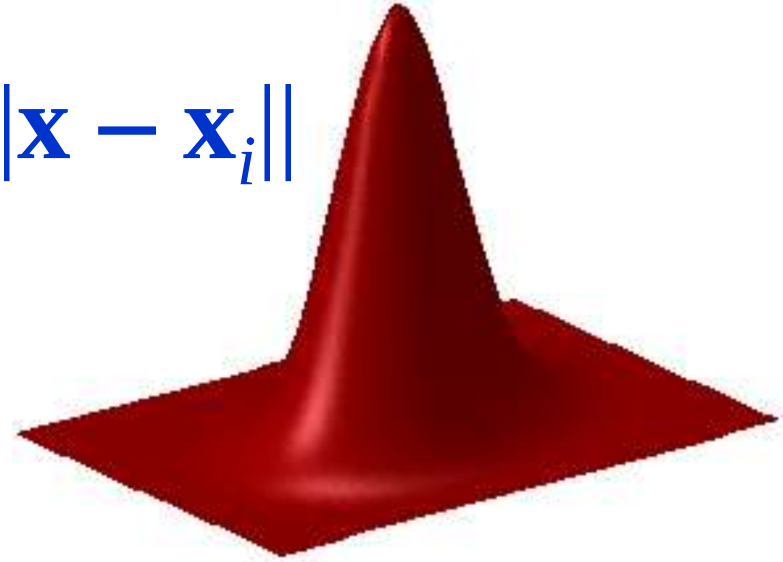
$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Radial Basis Functions

Three parameters for a radial function:

$$\phi_i(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{x}_i\|)$$

- Center $\mathbf{x}_i$
- Distance Measure $r = \|\mathbf{x} - \mathbf{x}_i\|$
- Shape ϕ



Typical Radial Functions

- Gaussian

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$

- Hardy-Multiquadratic (1971)

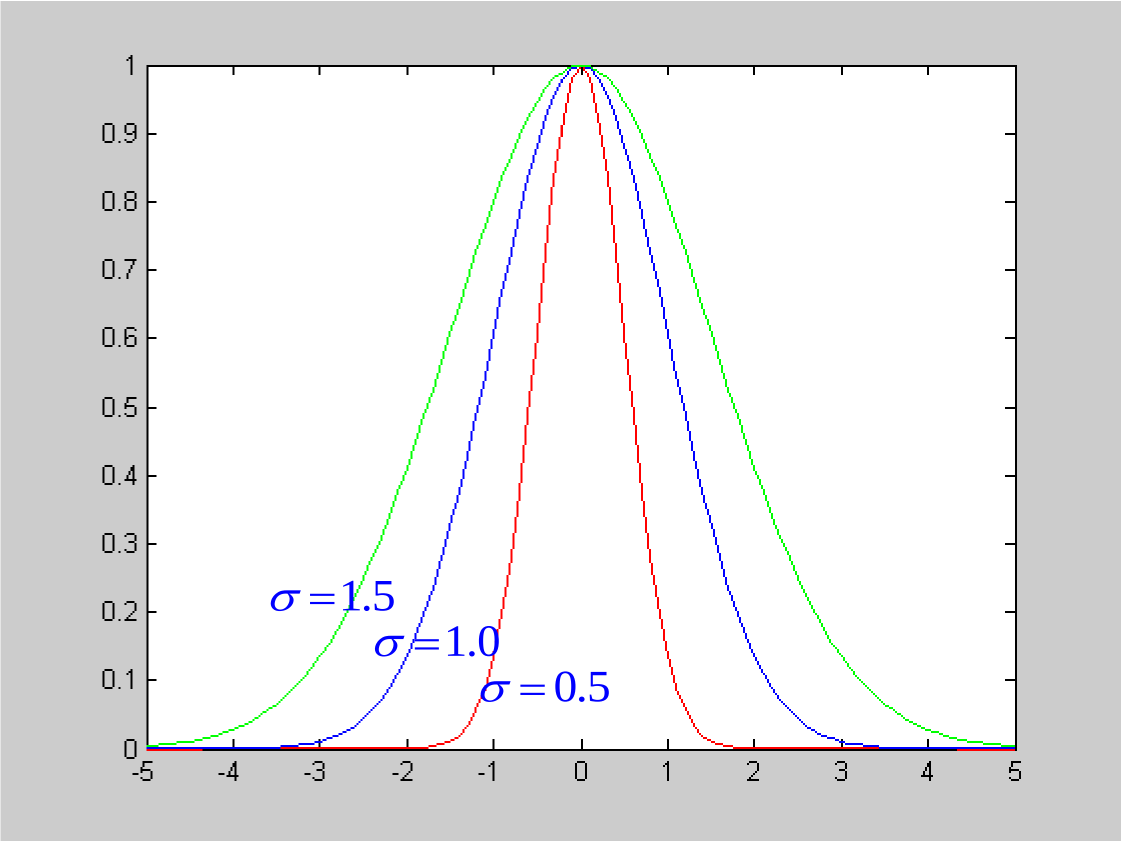
$$\phi(r) = \sqrt{r^2 + c^2} \big/ c \quad c > 0 \quad \text{and} \quad r \in \Re$$

- Inverse Multiquadratic

$$\phi(r) = c \big/ \sqrt{r^2 + c^2} \quad c > 0 \quad \text{and} \quad r \in \Re$$

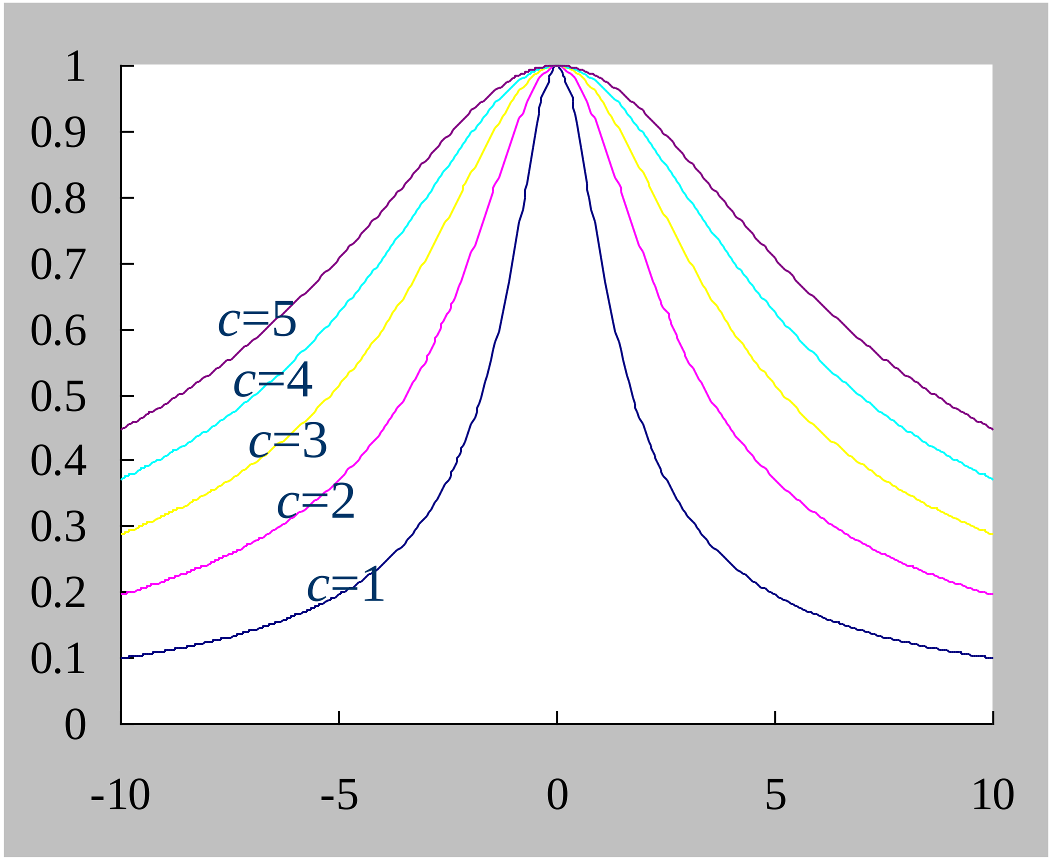
$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$

Gaussian Basis Function (σ=0.5,1.0,1.5)



$$\phi(r) = c\big/\sqrt{r^2 + c^2} \quad c > 0 \quad \text{and} \quad r \in \Re$$

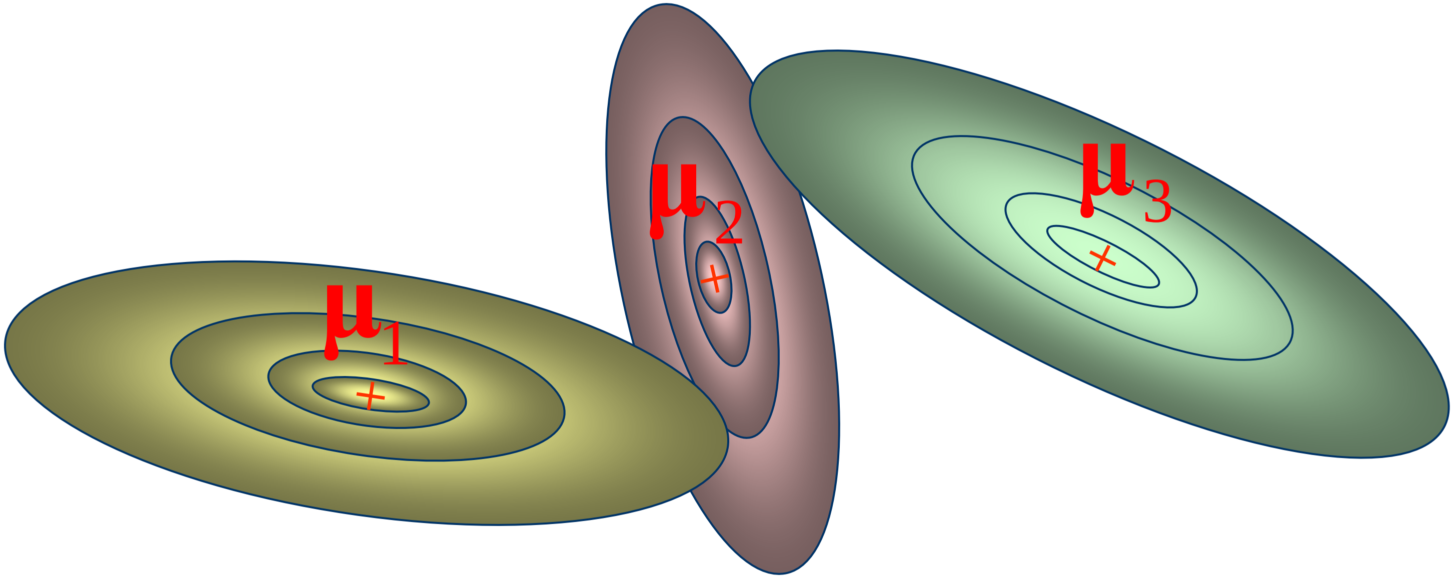
Inverse Multiquadratic



Basis $\{\phi_i: i = 1, 2, \ldots\}$ is 'near' orthogonal.

Most General RBF

$$\phi_i(\mathbf{x}) = \phi\left((\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right)$$

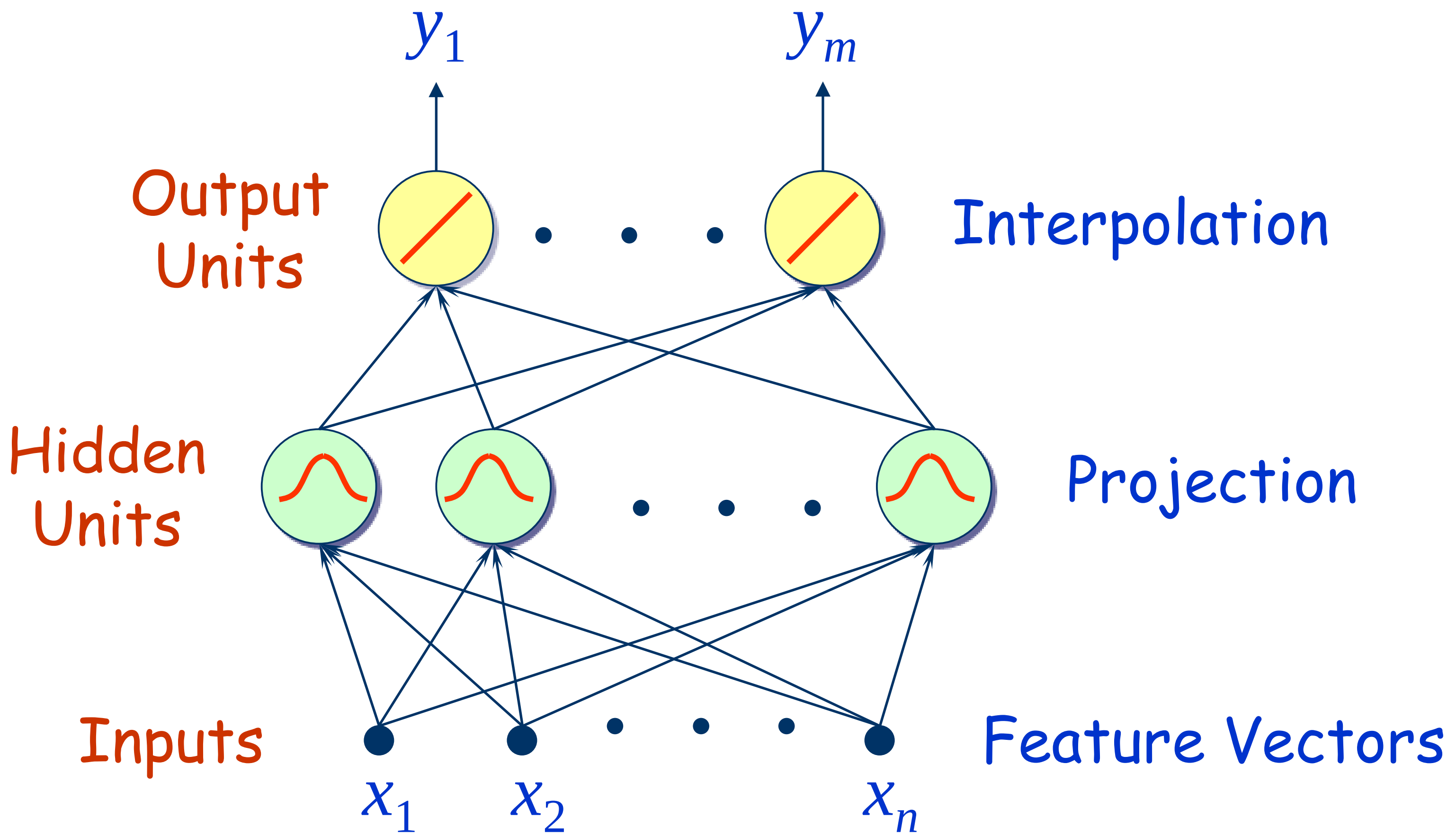


Properties of RBF's

- On-Center, Off Surround
- Analogies with localized receptive fields found in several biological structures, e.g.,
  - visual cortex;
  - ganglion cells

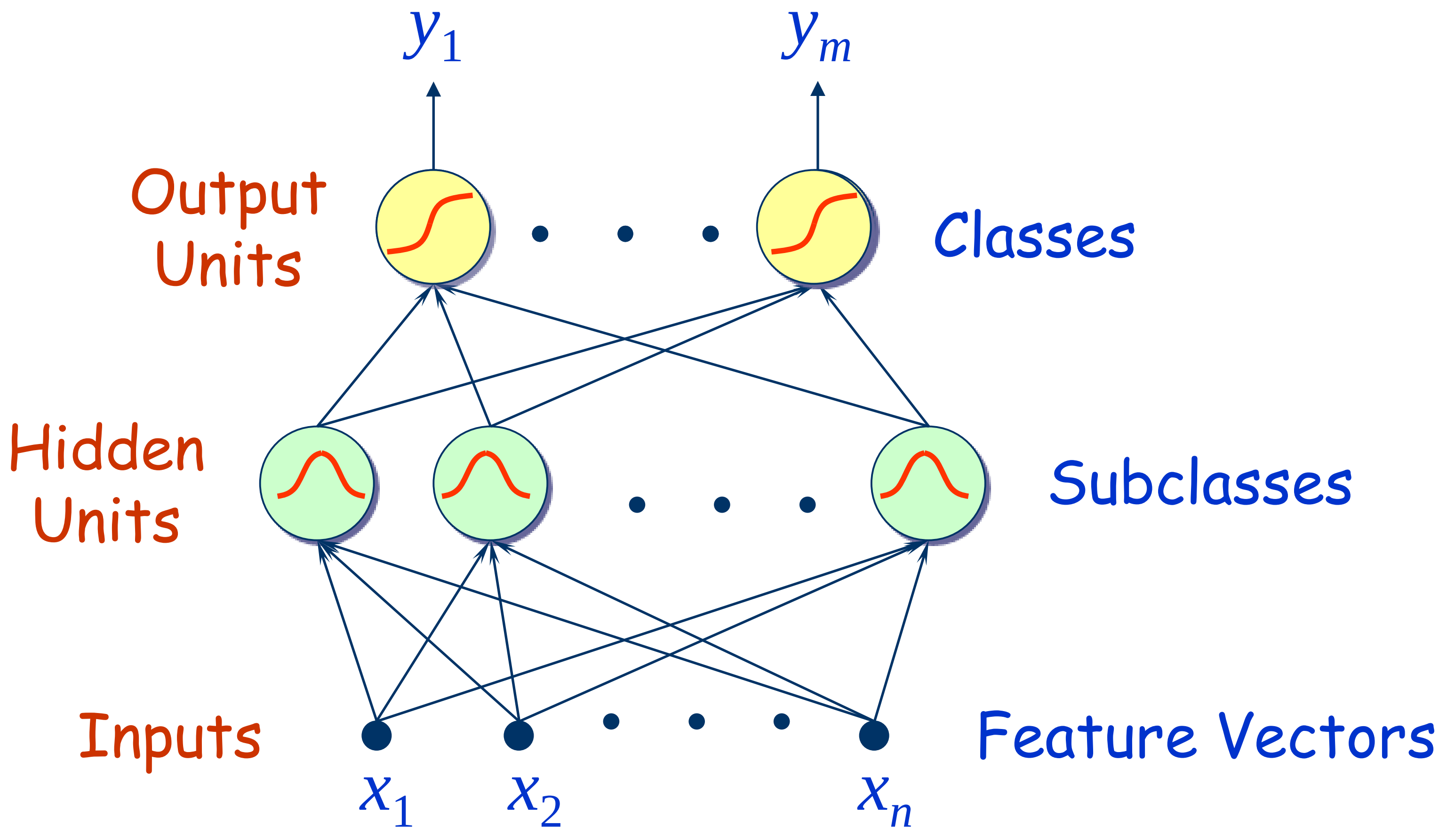
As a function approximator $\mathbf{y} = \mathbf{f}(\mathbf{x})$

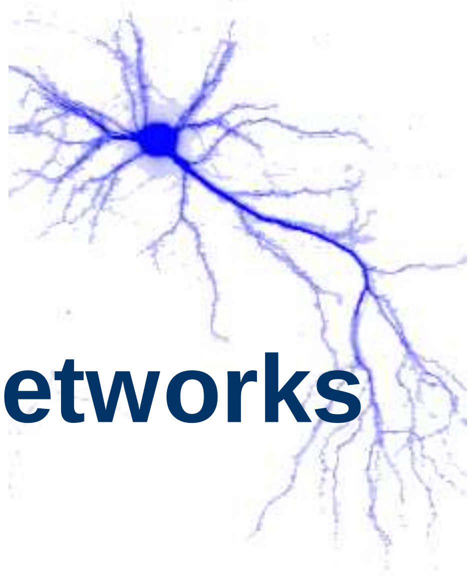
The Topology of RBF



As a pattern classifier.

The Topology of RBF





Introduction to Radial Basis Function Networks

RBFN's for
Function Approximation

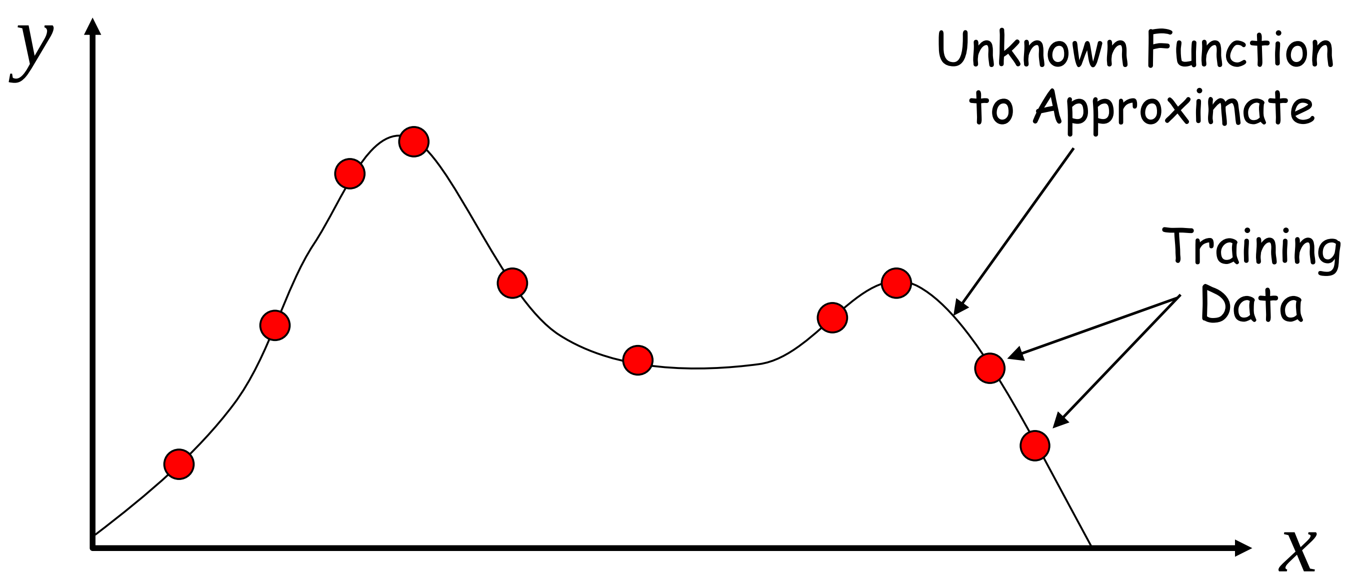
Radial Basis Function Networks

- Radial basis function (RBF) networks are feed-forward networks trained using a supervised training algorithm.
- The activation function is selected from a class of functions called basis functions.
- They usually train much faster than BP.
- They are less susceptible to problems with non-stationary inputs

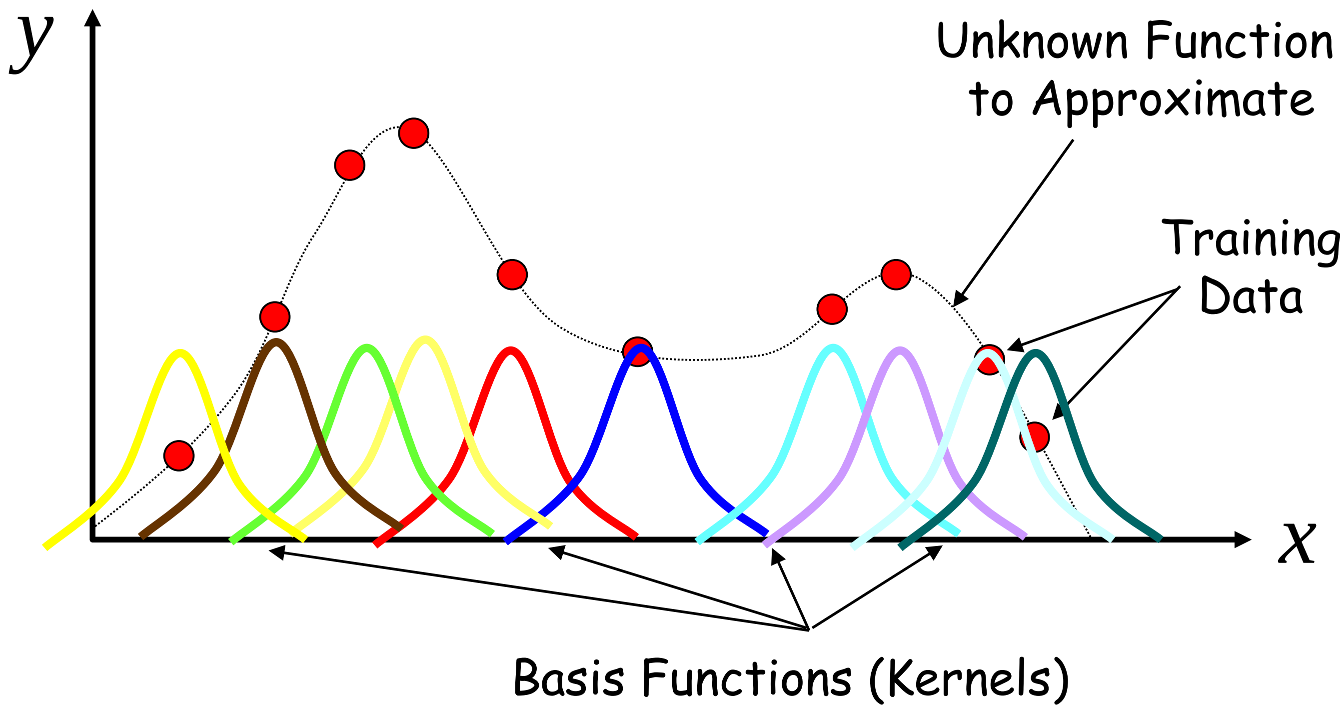
Radial Basis Function Networks

- Popularized by Broomhead and Lowe (1988), and Moody and Darken (1989), RBF networks have proven to be a useful neural network architecture.
- The major difference between RBF and BP is the behavior of the single hidden layer.
- Rather than using the sigmoidal or S-shaped activation function as in BP, the hidden units in RBF networks use a Gaussian or some other basis kernel function.

The idea

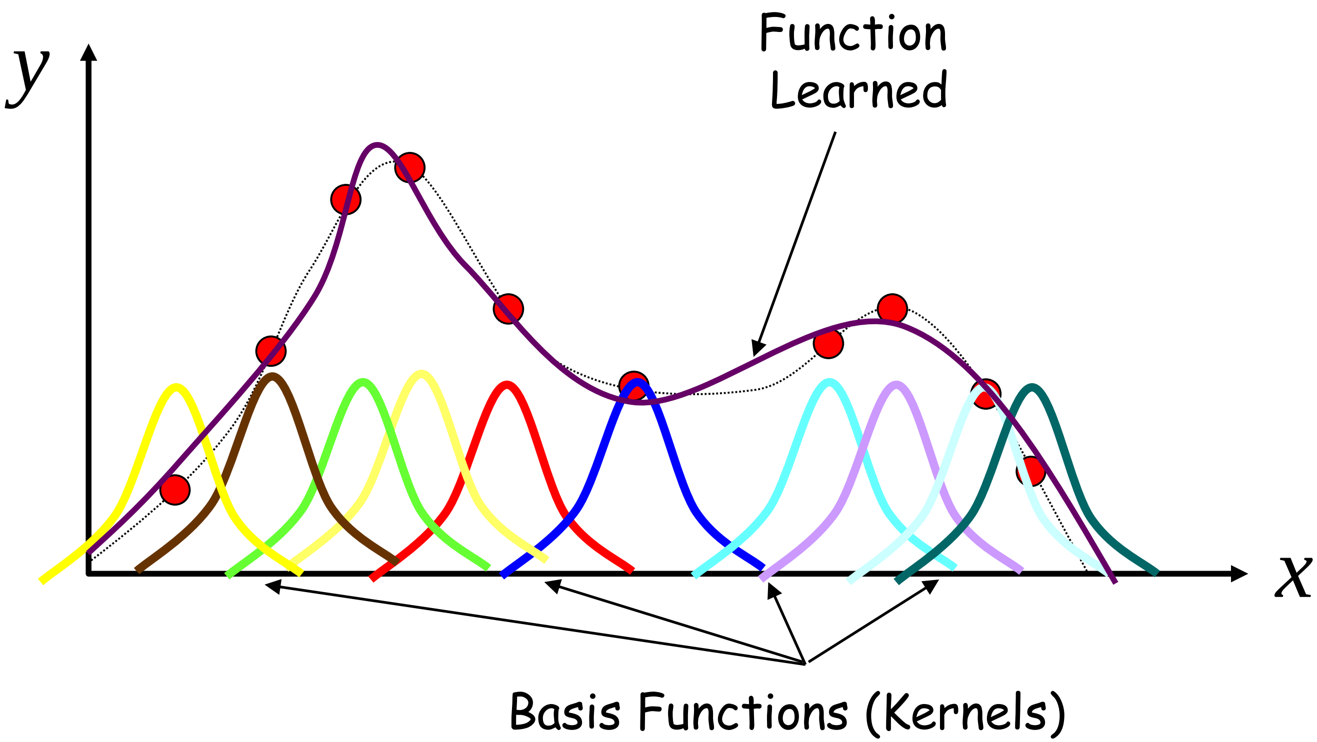


The idea


y
x
Unknown Function
to Approximate
Training
Data
Basis Functions (Kernels)

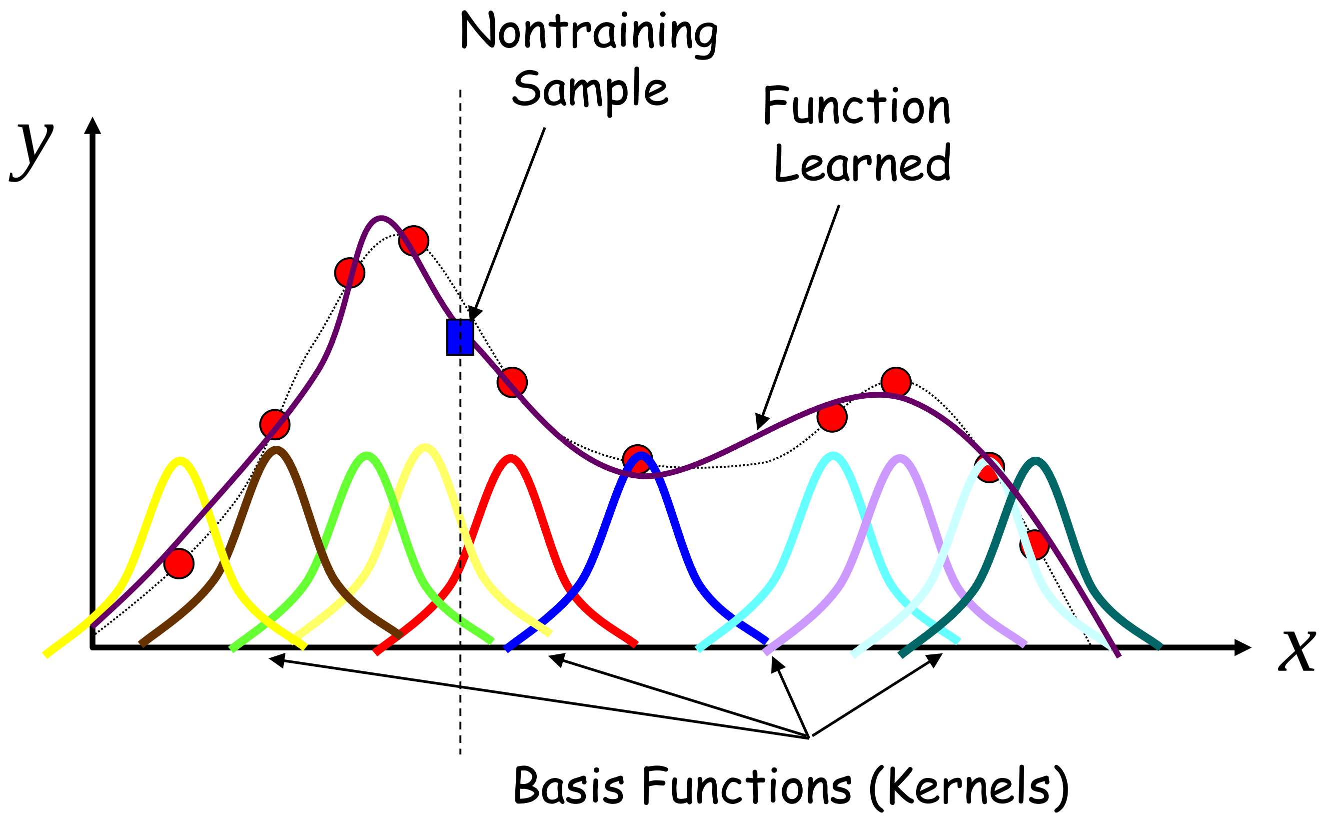
The idea

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$



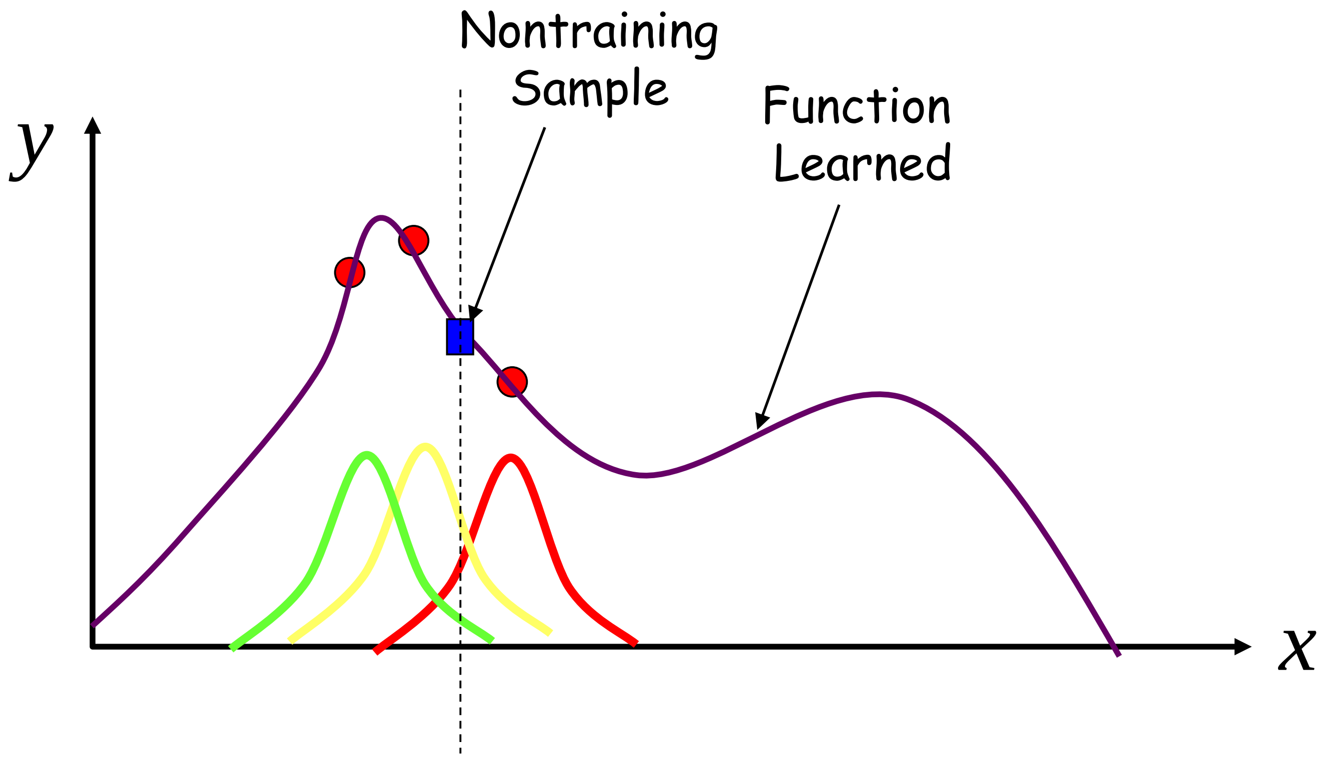
The idea

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$



The idea

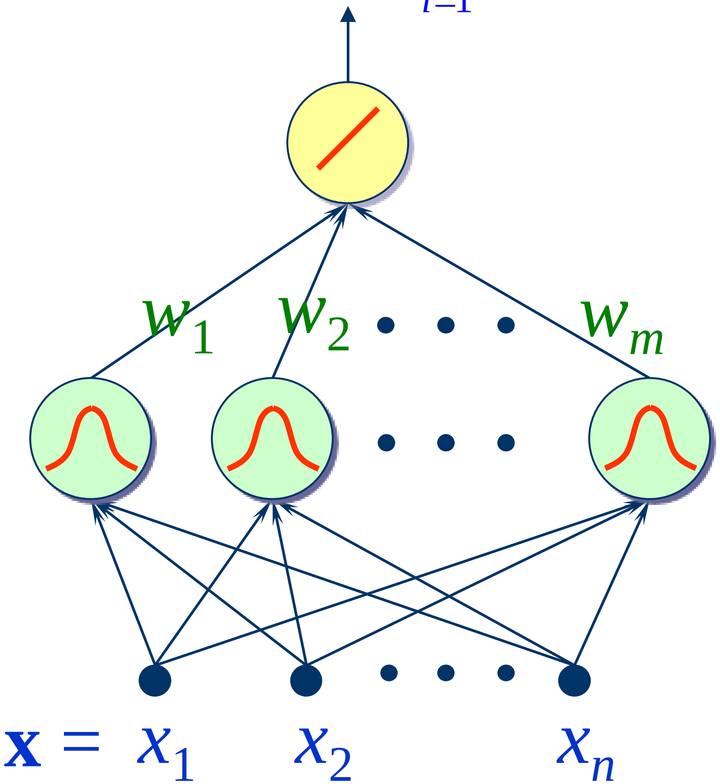
$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$



Radial Basis Function Networks as Universal Aproximators

Training set $\mathcal{T} = \left\{ \left(\mathbf{x}^{(k)}, y^{(k)} \right) \right\}_{k=1}^{p}$

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$



Goal $y^{(k)} \approx f\left(\mathbf{x}^{(k)}\right)$ for all k

$$\min SSE = \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right]^2$$

$$= \sum_{k=1}^{p} \left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i \left(\mathbf{x}^{(k)}\right) \right]^2$$

Learn the Optimal Weight Vector

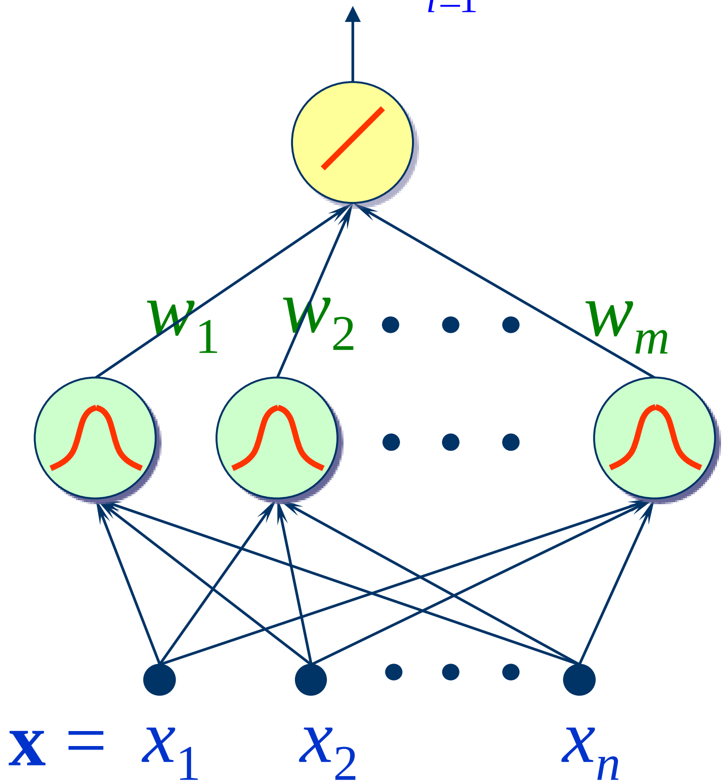
Training set $\mathcal{T} = \left\{ \left(\mathbf{x}^{(k)}, y^{(k)} \right) \right\}_{k=1}^{p}$

Goal $y^{(k)} \approx f\left(\mathbf{x}^{(k)}\right)$ for all k

$$\min SSE = \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right]^2$$

$$= \sum_{k=1}^{p} \left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i \left(\mathbf{x}^{(k)}\right) \right]^2$$

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i (\mathbf{x})$$



Regularization $\lambda_i \geq 0$

Training set $\mathcal{T} = \left\{ \left(\mathbf{x}^{(k)}, y^{(k)} \right) \right\}_{k=1}^{p}$

If regularization is unneeded, set $\lambda_i = 0$

Goal $y^{(k)} \approx f\left(\mathbf{x}^{(k)} \right)$ for all k

$$\min C = \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)} \right) \right]^2 + \sum_{i=1}^{m} \lambda_i w_i^2$$

$$= \sum_{k=1}^{p} \left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i \left(\mathbf{x}^{(k)} \right) \right]^2 + \sum_{i=1}^{m} \lambda_i w_i^2$$

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}) \qquad f^*(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x})$$

Learn the Optimal Weight Vector

Minimize $\quad C = \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right]^2 + \sum_{i=1}^{m} \lambda_i w_i^2$

$$0 = \frac{\partial C}{\partial w_j} = -2 \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right] \frac{\partial f\left(\mathbf{x}^{(k)}\right)}{\partial w_j} + 2\lambda_j w_j$$

$$= -2 \sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right] \phi_j\left(\mathbf{x}^{(k)}\right) + 2\lambda_j w_j$$

$$\Rightarrow \quad \sum_{k=1}^{p} \phi_j\left(\mathbf{x}^{(k)}\right) f^*\left(\mathbf{x}^{(k)}\right) + \lambda_j w_j^* = \sum_{k=1}^{p} \phi_j\left(\mathbf{x}^{(k)}\right) y^{(k)}$$

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}) \qquad f^*(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x})$$

Learn the Optimal Weight Vector

$$\boldsymbol{\varphi}_j^T \mathbf{f}^* + \lambda_j w_j^* = \boldsymbol{\varphi}_j^T \mathbf{y} \qquad j = 1, \ldots, m$$

Define
$$\begin{cases} \boldsymbol{\varphi}_j = \left(\phi_j\left(\mathbf{x}^{(1)}\right), \ldots, \phi_j\left(\mathbf{x}^{(p)}\right)\right)^T \\ \mathbf{f}^* = \left(f^*\left(\mathbf{x}^{(1)}\right), \ldots, f^*\left(\mathbf{x}^{(p)}\right)\right)^T \\ \mathbf{y} = \left(y^{(1)}, \ldots, y^{(p)}\right)^T \end{cases}$$

$$\sum_{k=1}^{p} \phi_j\left(\mathbf{x}^{(k)}\right) f^*\left(\mathbf{x}^{(k)}\right) + \lambda_j w_j^* = \sum_{k=1}^{p} \phi_j\left(\mathbf{x}^{(k)}\right) y^{(k)}$$

$$\boldsymbol{\varphi}_j^T \mathbf{f}^* + \lambda_j w_j^* = \boldsymbol{\varphi}_j^T \mathbf{y} \qquad j = 1, \ldots, m$$

Learn the Optimal Weight Vector

$$\boldsymbol{\varphi}_1^T \mathbf{f}^* + \lambda_1 w_1^* = \boldsymbol{\varphi}_1^T \mathbf{y}$$

$$\boldsymbol{\varphi}_2^T \mathbf{f}^* + \lambda_2 w_2^* = \boldsymbol{\varphi}_2^T \mathbf{y}$$

$$\vdots \qquad \vdots$$

$$\boldsymbol{\varphi}_m^T \mathbf{f}^* + \lambda_m w_m^* = \boldsymbol{\varphi}_m^T \mathbf{y}$$

Define

$$\boldsymbol{\Phi} = \left(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_m\right)$$

$$\mathbf{w}^* = \left(w_1^*, w_2^*, \ldots, w_m^*\right)^T$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix}$$

$$\Rightarrow \boldsymbol{\Phi}^T \mathbf{f}^* + \boldsymbol{\Lambda} \mathbf{w}^* = \boldsymbol{\Phi}^T \mathbf{y}$$

$$f^*(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x})$$

Learn the Optimal Weight Vector

$$\mathbf{f}^* = \begin{bmatrix} f^*\left(\mathbf{x}^{(1)}\right) \\ f^*\left(\mathbf{x}^{(2)}\right) \\ \vdots \\ f^*\left(\mathbf{x}^{(p)}\right) \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m} w_k^* \phi_k\left(\mathbf{x}^{(1)}\right) \\ \sum_{k=1}^{m} w_k^* \phi_k\left(\mathbf{x}^{(2)}\right) \\ \vdots \\ \sum_{k=1}^{m} w_k^* \phi_k\left(\mathbf{x}^{(P)}\right) \end{bmatrix} = \begin{bmatrix} \phi_1\left(\mathbf{x}^{(1)}\right) & \phi_2\left(\mathbf{x}^{(1)}\right) & \cdots & \phi_m\left(\mathbf{x}^{(1)}\right) \\ \phi_1\left(\mathbf{x}^{(2)}\right) & \phi_2\left(\mathbf{x}^{(2)}\right) & \cdots & \phi_m\left(\mathbf{x}^{(2)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1\left(\mathbf{x}^{(p)}\right) & \phi_2\left(\mathbf{x}^{(p)}\right) & \cdots & \phi_m\left(\mathbf{x}^{(p)}\right) \end{bmatrix} \begin{bmatrix} w_1^* \\ w_2^* \\ \vdots \\ w_m^* \end{bmatrix} = \mathbf{\Phi} \mathbf{w}^*$$

$$\mathbf{\Phi}^T \mathbf{\Phi} \mathbf{w}^* + \mathbf{\Lambda} \mathbf{w}^* = \mathbf{\Phi}^T \mathbf{y}$$

$$\mathbf{\Phi}^T \mathbf{f}^* + \mathbf{\Lambda} \mathbf{w}^* = \mathbf{\Phi}^T \mathbf{y}$$

$$\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w}^* + \mathbf{\Lambda}\mathbf{w}^* = \mathbf{\Phi}^T\mathbf{y}$$

Learn the Optimal Weight Vector

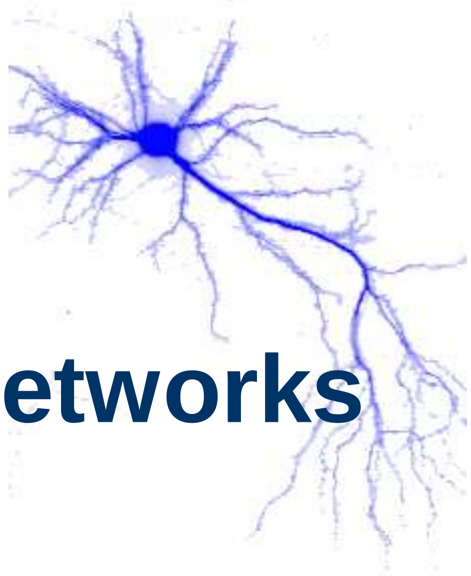
$$\left(\mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{\Lambda}\right)\mathbf{w}^* = \mathbf{\Phi}^T\mathbf{y}$$

$$\mathbf{w}^* = \left(\mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{\Lambda}\right)^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$$= \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$\mathbf{\Phi}$: Design Matrix

$\mathbf{A}^{-1}$: Variance Matrix

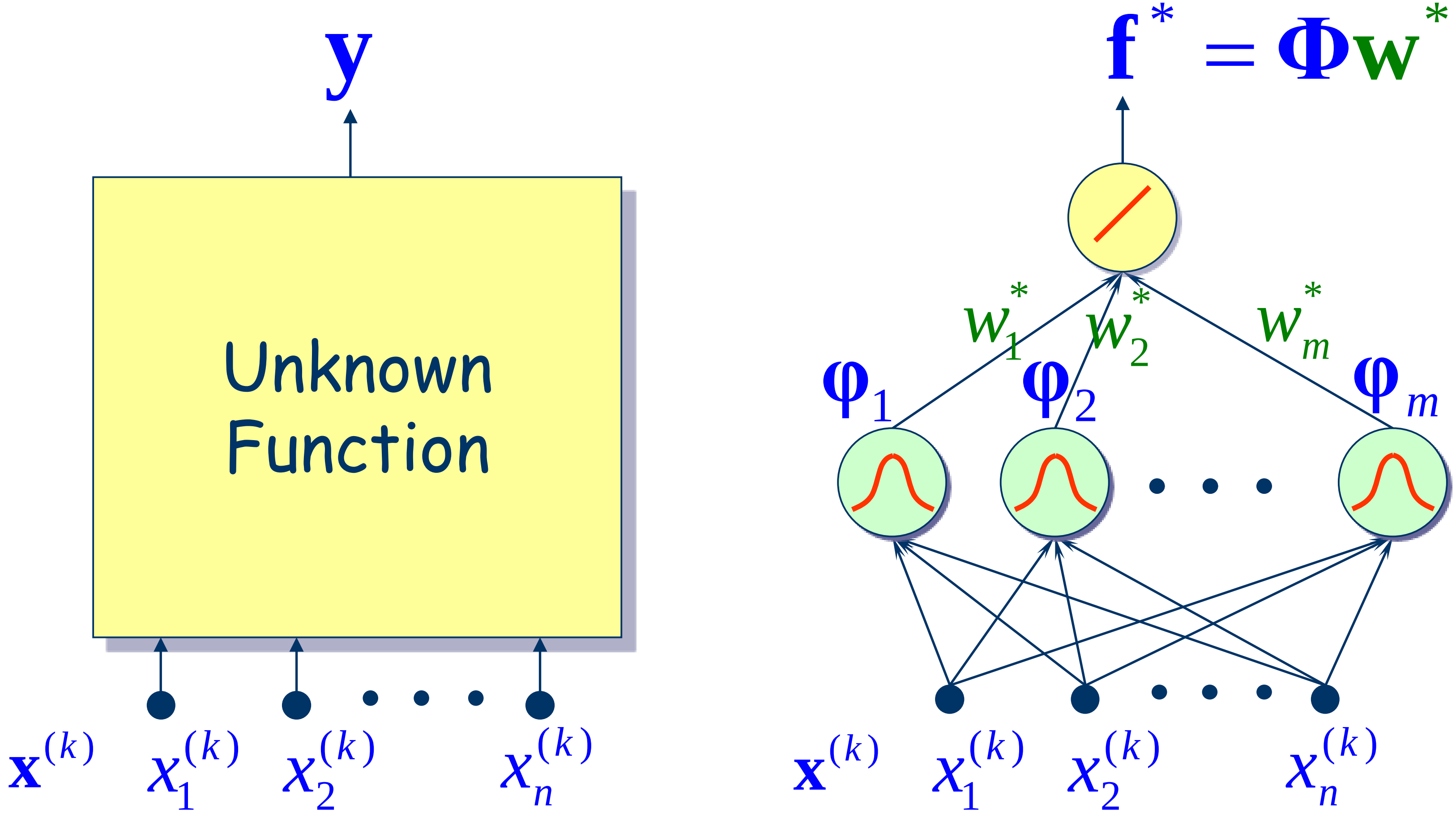


Introduction to Radial Basis Function Networks

The Projection Matrix

$$\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

The Empirical-Error Vector



$$\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

The Empirical-Error Vector

$\mathbf{y}$

$\mathbf{f}^* = \mathbf{\Phi}\mathbf{w}^*$

Error Vector

$$\mathbf{e} = \mathbf{y} - \mathbf{f}^* = \mathbf{y} - \mathbf{\Phi}\mathbf{w}^* = \mathbf{y} - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$$= \left(\mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T\right)\mathbf{y} = \mathbf{P}\mathbf{y}$$

$\mathbf{x}^{(k)}$ $x_1^{(k)}$ $x_2^{(k)}$ $\cdots$ $x_n^{(k)}$

$\mathbf{x}^{(k)}$ $x_1^{(k)}$ $x_2^{(k)}$ $\cdots$ $x_n^{(k)}$

If $\Lambda=\mathbf{0}$, the RBFN's learning algorithm is to minimize SSE (MSE).

Sum-Squared-Error

Error Vector

$$\mathbf{e} = \mathbf{P}\mathbf{y}$$

$$\boldsymbol{\Phi} = \left(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_m\right)$$

$$\mathbf{A} = \boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Lambda}$$

$$\mathbf{P} = \mathbf{I}_p - \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T$$

$$SSE = \sum_{k=1}^{p}\left[y^{(k)} - f^*\left(\mathbf{x}^{(k)}\right)\right]^2 = \left(\mathbf{P}\mathbf{y}\right)^T\mathbf{P}\mathbf{y}$$

$$= \mathbf{y}^T\mathbf{P}^2\mathbf{y}$$

$$SSE = \mathbf{y}^T \mathbf{P}^2 \mathbf{y} = \mathbf{y}^T \mathbf{P} \mathbf{y}$$

The Projection Matrix

Error Vector

$$\mathbf{e} = \mathbf{P}\mathbf{y}$$

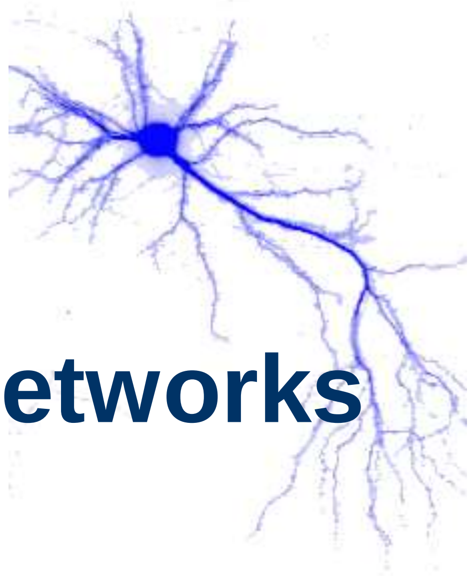
$$\boldsymbol{\Phi} = \left(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_m \right)$$

$$\mathbf{A} = \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \cancel{\boldsymbol{\Lambda}}$$

$$\mathbf{P} = \mathbf{I}_p - \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^T$$

$$\boldsymbol{\Lambda} = \mathbf{0} \Rightarrow \mathbf{e} \perp span\left(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_m \right)$$

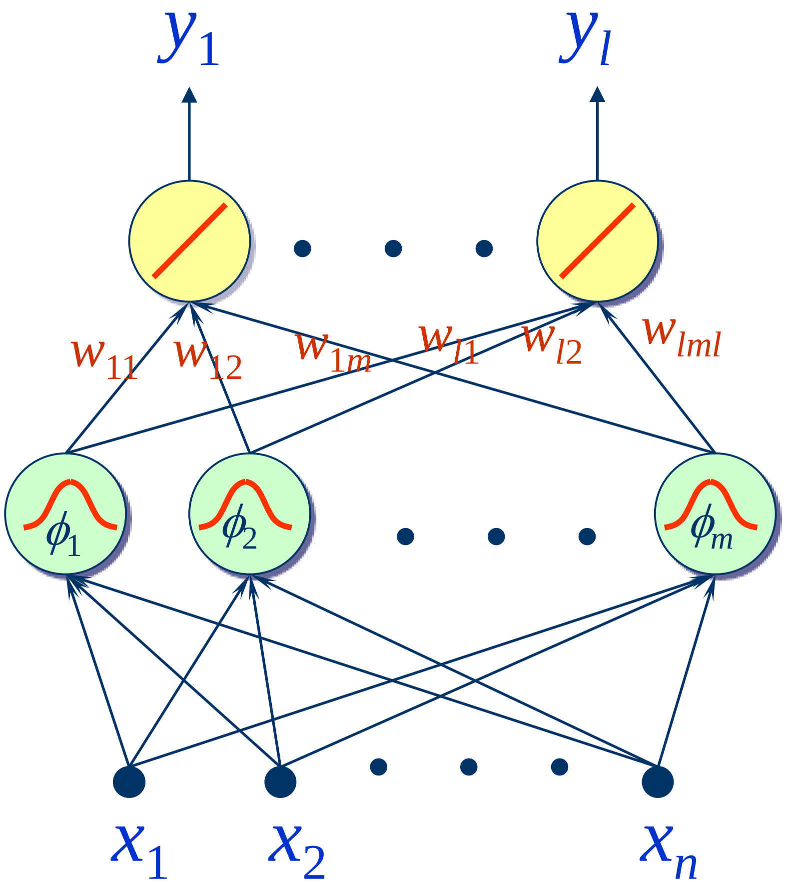
$$\Rightarrow \mathbf{P}(\mathbf{P}\mathbf{y}) = \mathbf{P}\mathbf{e} = \mathbf{e} = \mathbf{P}\mathbf{y}$$



Introduction to Radial Basis Function Networks

Learning the Kernels

RBFN's as Universal Approximators



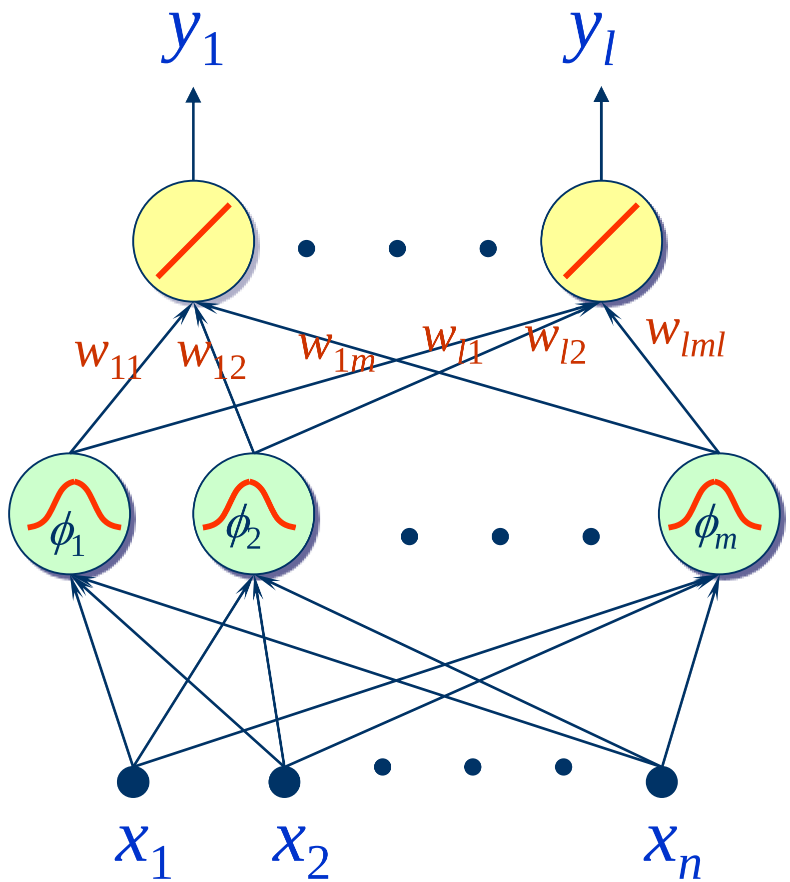
Training set

$$\mathcal{T} = \left\{ \left(\mathbf{x}^{(k)}, \mathbf{y}^{(k)} \right) \right\}_{k=1}^{p}$$

Kernels

$$\phi_j(\mathbf{x}) = \exp\left[-\frac{\left\| \mathbf{x} - \boldsymbol{\mu}_j \right\|^2}{2\sigma_j^2} \right]$$

What to Learn?



- Weights w_{ij}'s
- Centers μ_j's of ϕ_j's
- Widths σ_j's of ϕ_j's

- Number of ϕ_j's – Model Selection

$$f_i\left(\mathbf{x}^{(k)}\right)=\sum_{j=1}^{l} w_{ij}\phi_j\left(\mathbf{x}^{(k)}\right)$$

One-Stage Learning

$$\Delta w_{ij}=\eta_1\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)\phi_j\left(\mathbf{x}^{(k)}\right)$$

$$\Delta\boldsymbol{\mu}_j=\eta_2\phi_j\left(\mathbf{x}^{(k)}\right)\frac{\mathbf{x}-\boldsymbol{\mu}_j}{\sigma_j^2}\sum_{i=1}^{l} w_{ij}\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)$$

$$\Delta\sigma_j=\eta_3\phi_j\left(\mathbf{x}^{(k)}\right)\frac{\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|^2}{\sigma_j^3}\sum_{i=1}^{l} w_{ij}\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)$$

On

$$f_{i}\left(\mathbf{x}^{(k)}\right)=\sum^{l} w_{ij}\phi_{j}\left(\mathbf{x}^{(k)}\right)$$

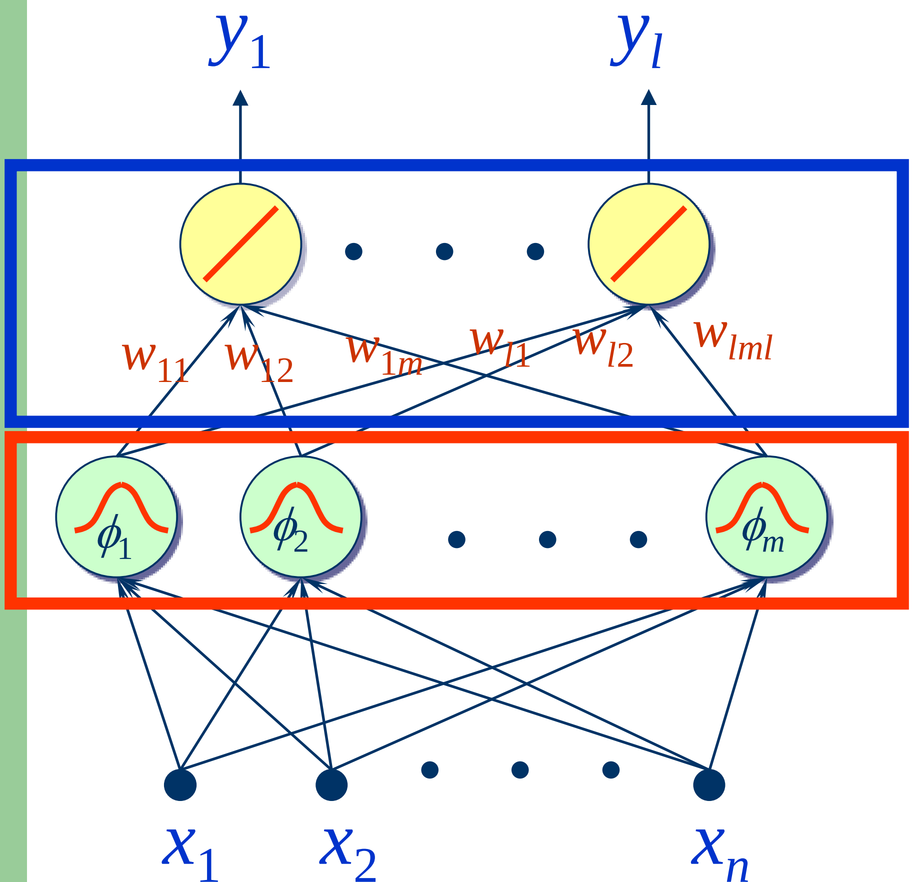
The simultaneous updates of all three sets of parameters may be suitable for non-stationary environments or on-line setting.

$$\Delta w_{ij}=\eta_1\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)\phi_j\left(\mathbf{x}^{(k)}\right)$$

$$\Delta\boldsymbol{\mu}_j=\eta_2\phi_j\left(\mathbf{x}^{(k)}\right)\frac{\mathbf{x}-\boldsymbol{\mu}_j}{\sigma_j^2}\sum_{i=1}^{l}w_{ij}\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)$$

$$\Delta\sigma_j=\eta_3\phi_j\left(\mathbf{x}^{(k)}\right)\frac{\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|^2}{\sigma_j^3}\sum_{i=1}^{l}w_{ij}\left(y_i^{(k)}-f_i\left(\mathbf{x}^{(k)}\right)\right)$$

Two-Stage Training



Step 2

Determines w_{ij}'s.

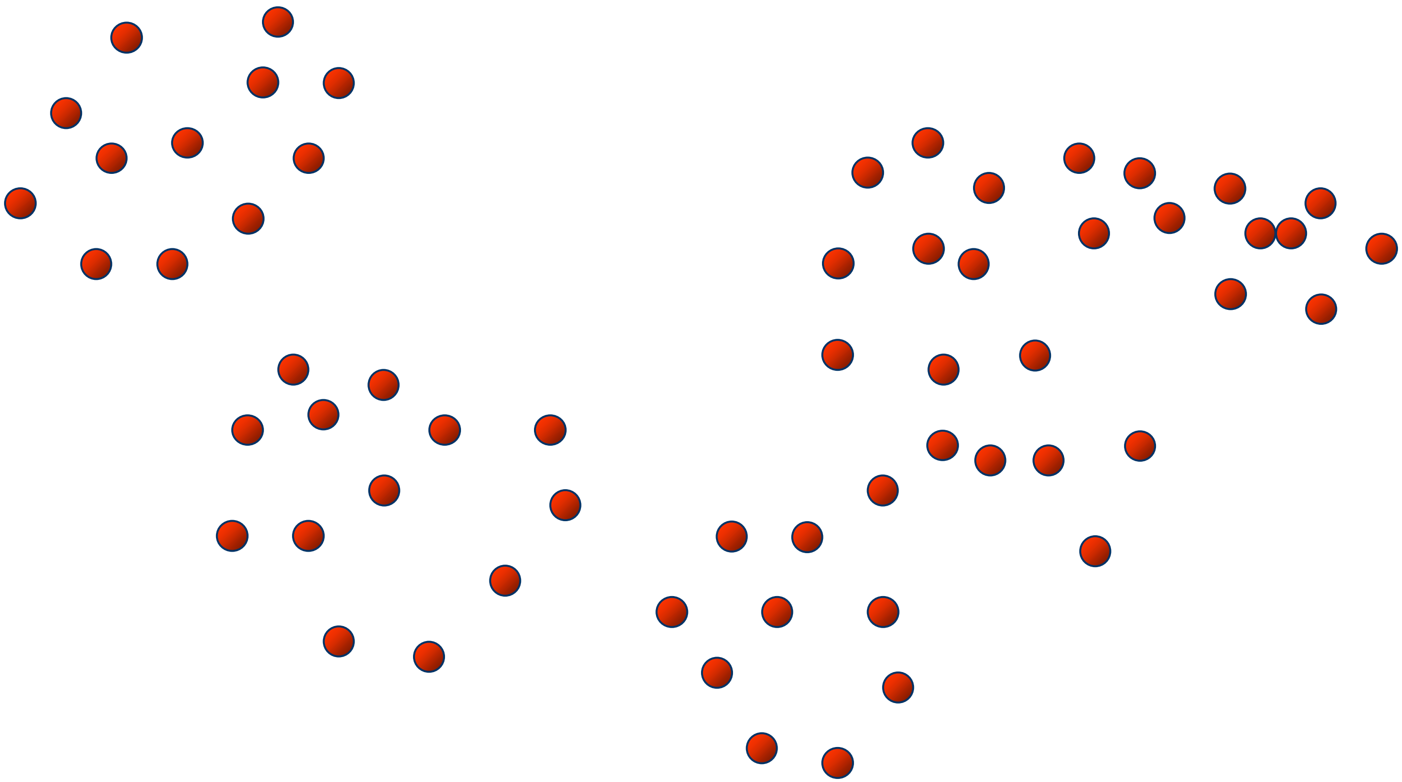
E.g., using batch-learning.

Step 1

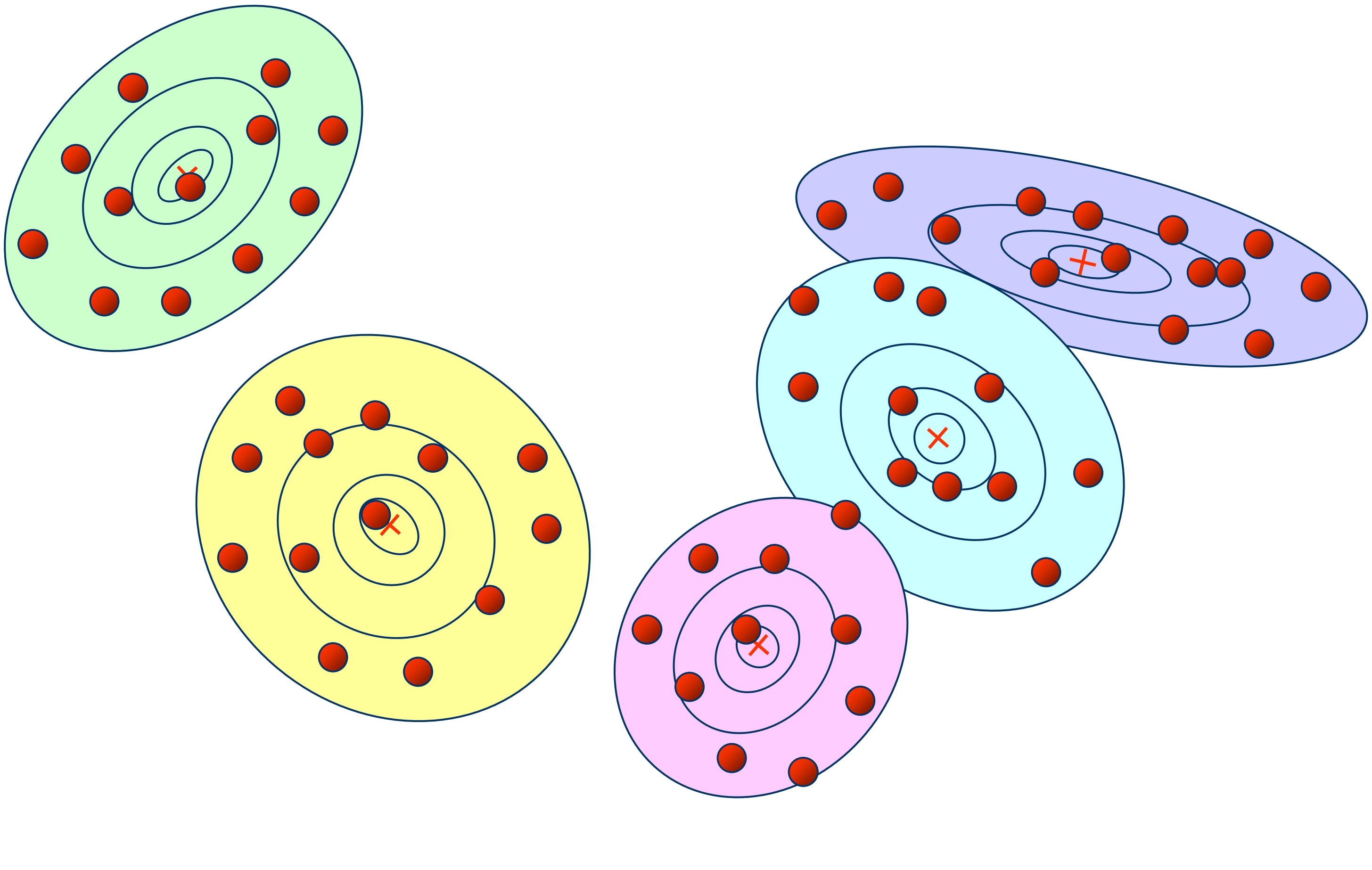
Determines

- Centers μ_j's of ϕ_j's.
- Widths σ_j's of ϕ_j's.
- Number of ϕ_j's.

Train the Kernels



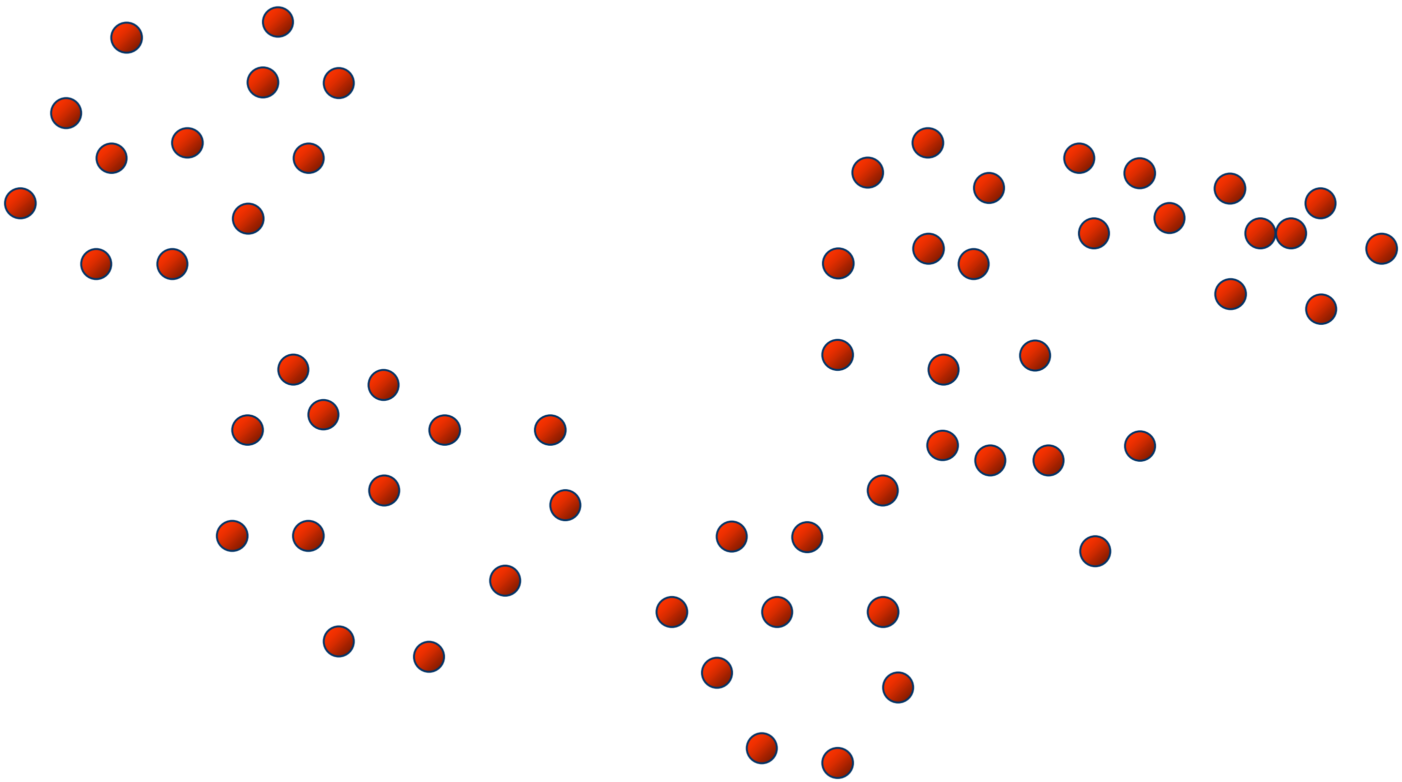
Unsupervised Training



Methods

- Subset Selection
  - Random Subset Selection
  - Forward Selection
  - Backward Elimination
- Clustering Algorithms
  - KMEANS
  - LVQ
- Mixture Models
  - GMM

Subset Selection

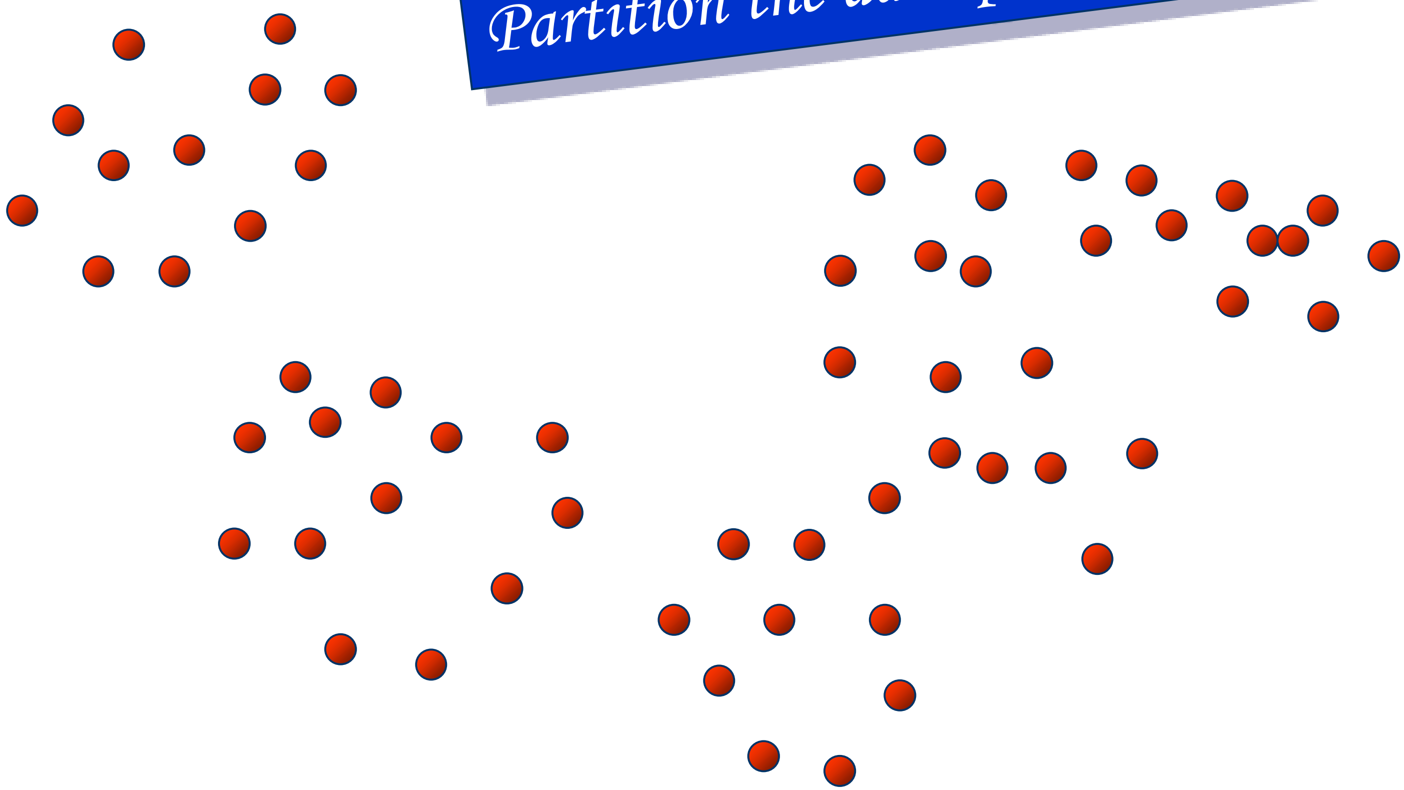


Random Subset Selection

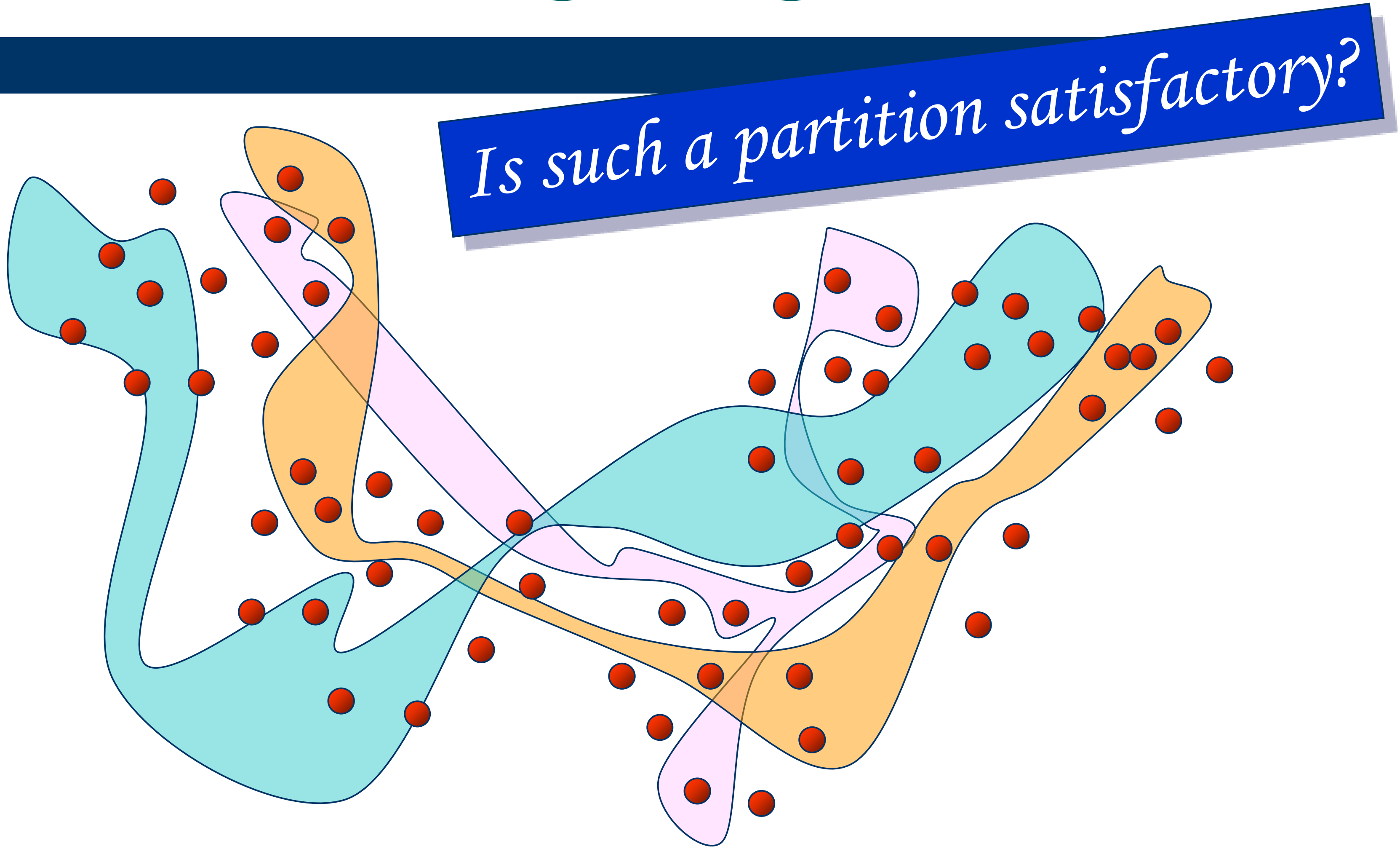
- Randomly choosing a subset of points from training set
- Sensitive to the initially chosen points.
- Using some adaptive techniques to tune
  - Centers
  - Widths
  - #points

Clustering Algorithms

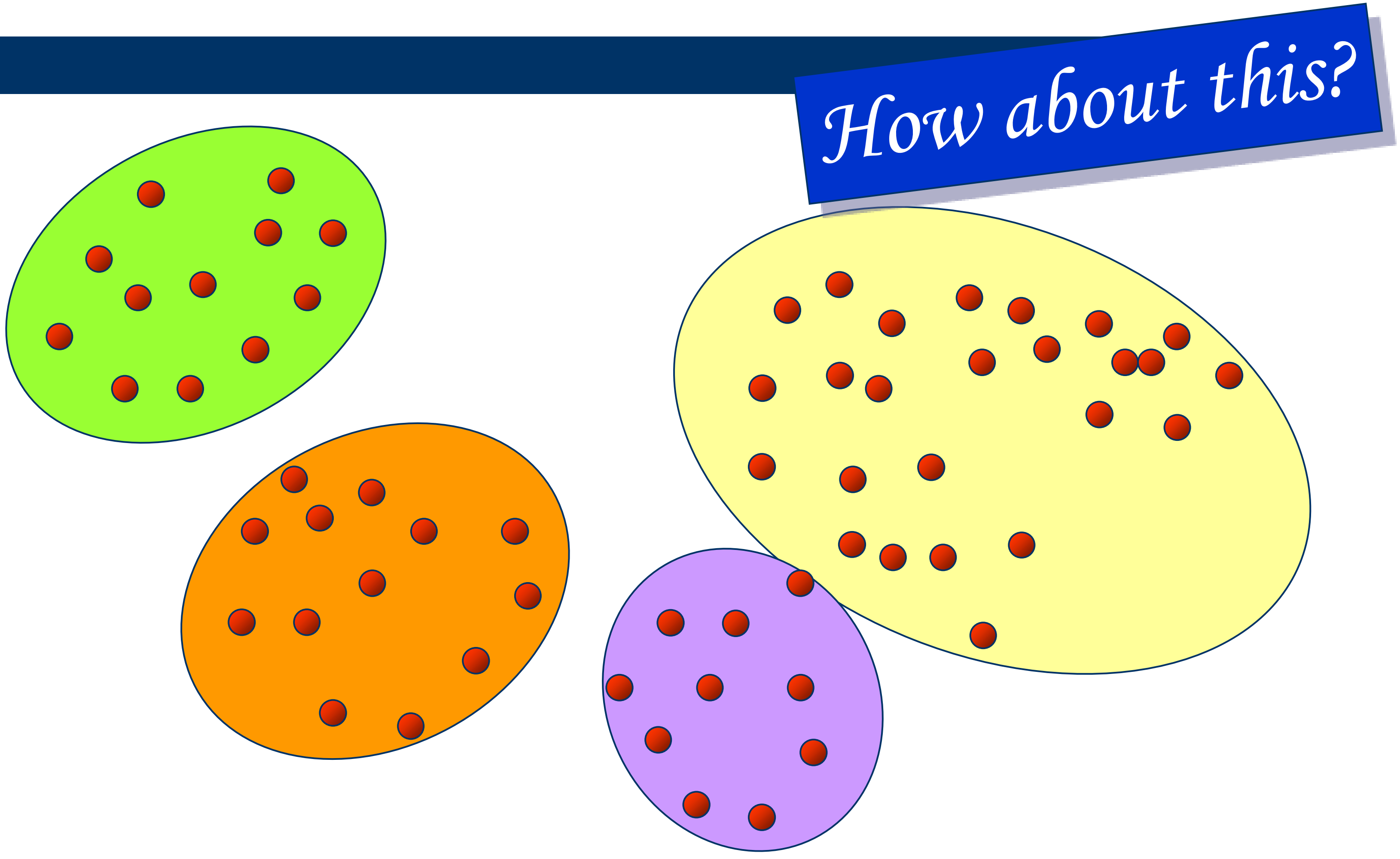
Partition the data points into K clusters.



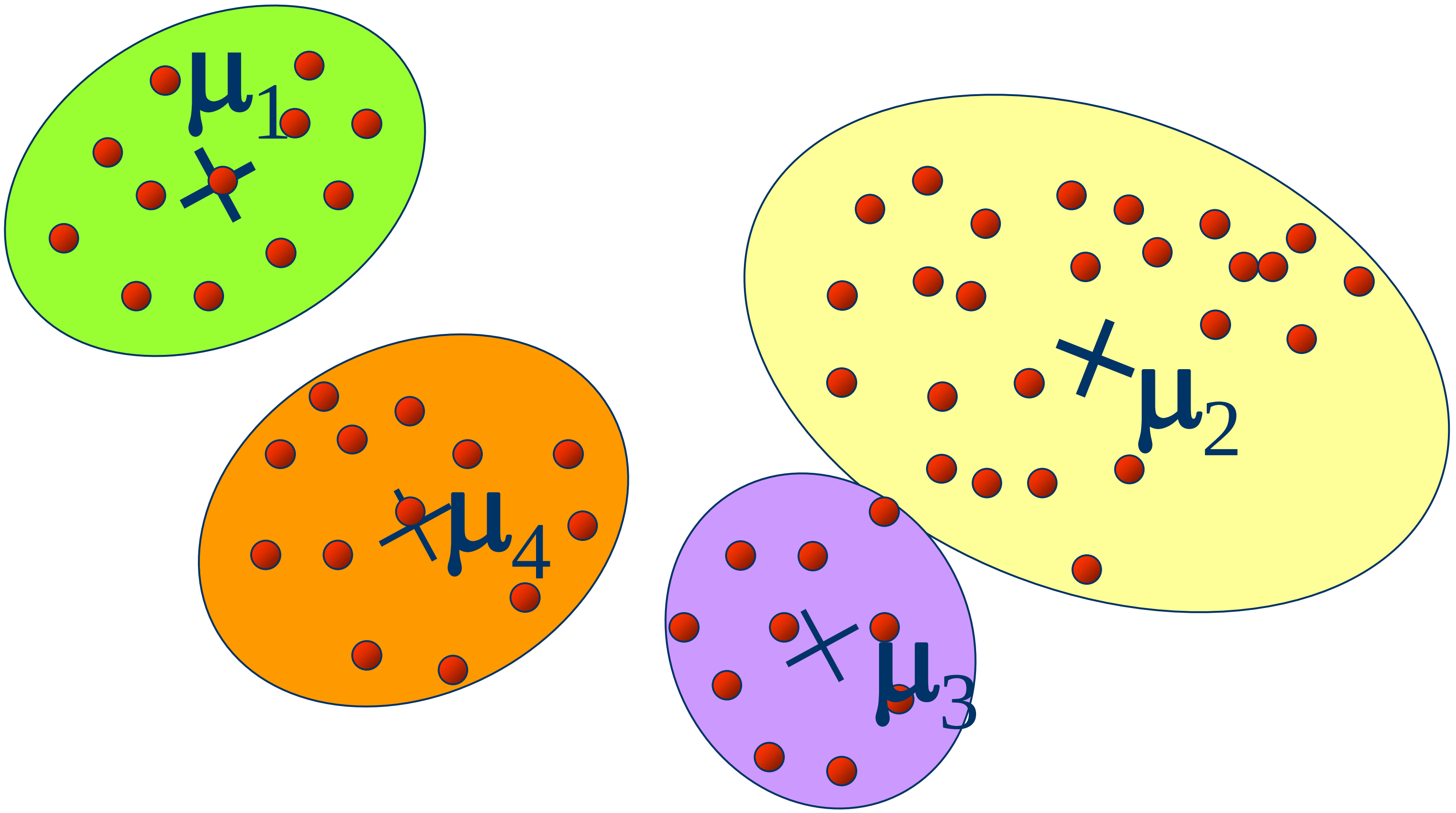
Clustering Algorithms

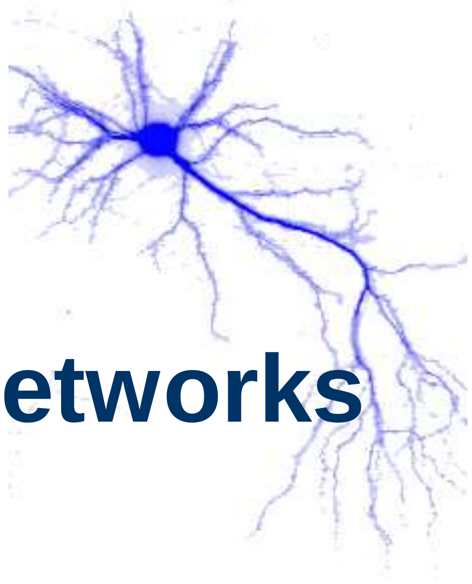


Clustering Algorithms



Clustering Algorithms





Introduction to Radial Basis Function Networks

Bias-Variance Dilemma

Questions -How should the user choose the kernel?

- Problem similar to that of selecting features for other learning algorithms.
  - *Poor choice* ---learning made very difficult.
  - *Good choice* ---even poor learners could succeed.
- The requirement from the user is thus critical.
  - can this requirement be lessened?
  - is a more automatic selection of features possible?

Goal Revisit

- Ultimate Goal – Generalization

Minimize Prediction Error

- Goal of Our Learning Procedure

Minimize Empirical Error

Badness of Fit

- Underfitting
  - A model (e.g., network) that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting.
  - Produces excessive bias in the outputs.
- Overfitting
  - A model (e.g., network) that is too complex may fit the noise, not just the signal, leading to overfitting.
  - Produces excessive variance in the outputs.

Underfitting/Overfitting Avoidance

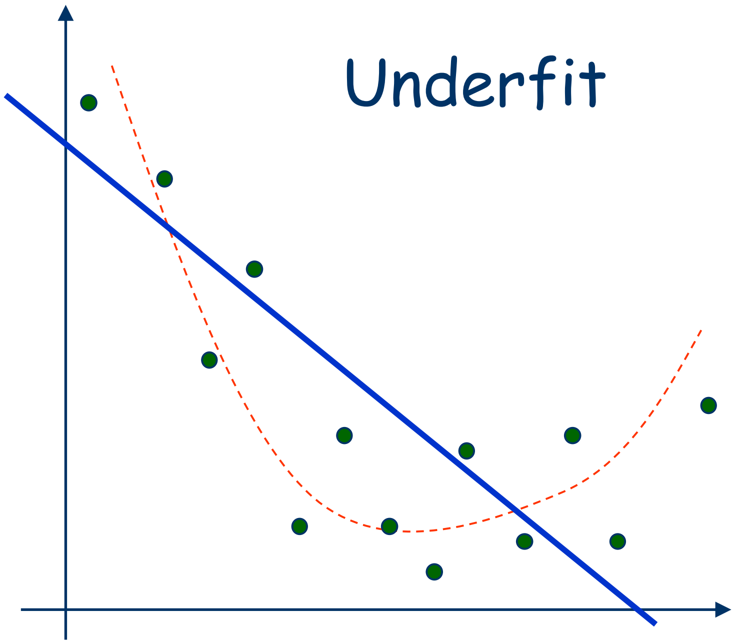
- Model selection
- Jittering
- Early stopping
- Weight decay
  - Regularization
  - Ridge Regression
- Bayesian learning
- Combining networks

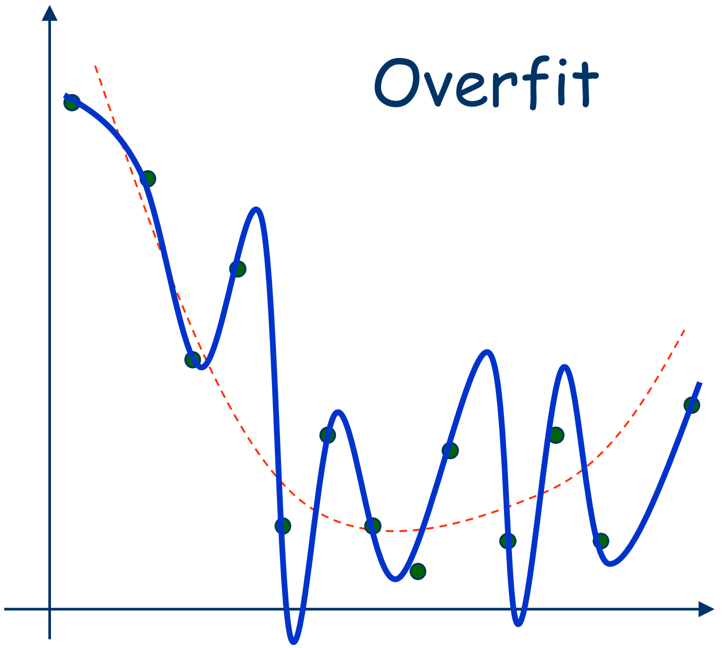
Best Way to Avoid Overfitting

- Use lots of training data, e.g.,
  - 30 times as many training cases as there are weights in the network.
  - for noise-free data, 5 times as many training cases as weights may be sufficient.

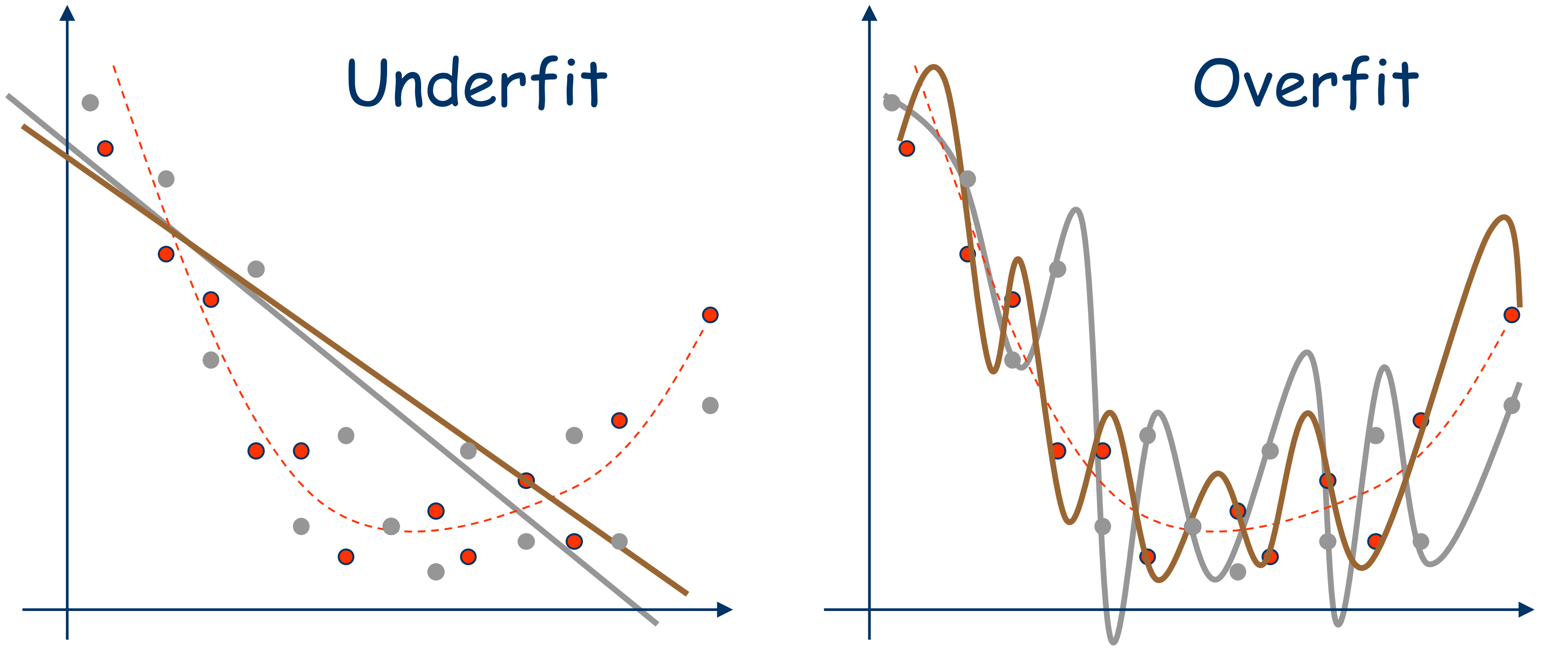
- Don't arbitrarily reduce the number of weights for fear of underfitting.

Badness of Fit



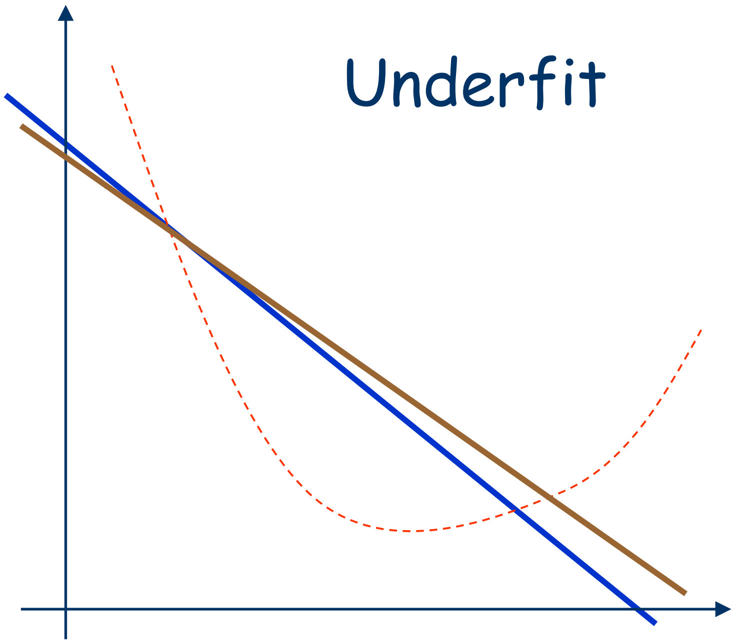


Badness of Fit



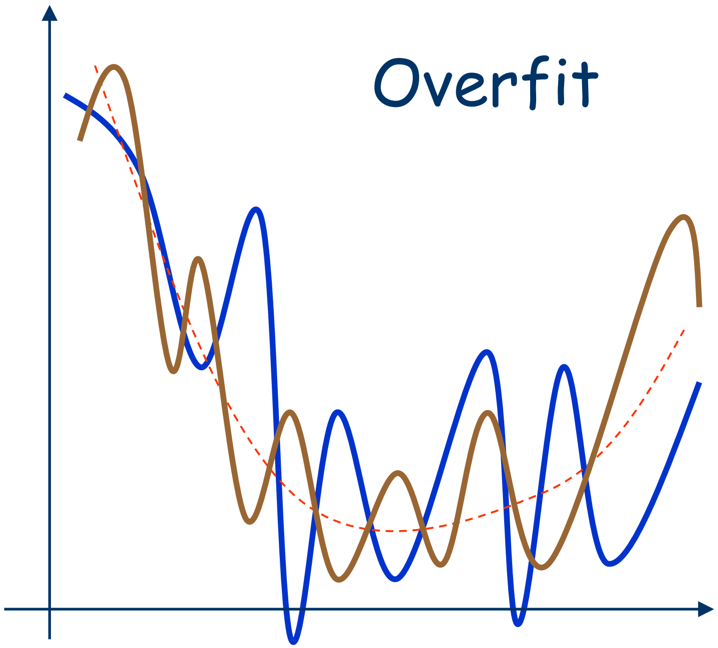
However, it's *not* really a dilemma.

Bias-Variance Dilemma



Large bias

Small variance



Small bias

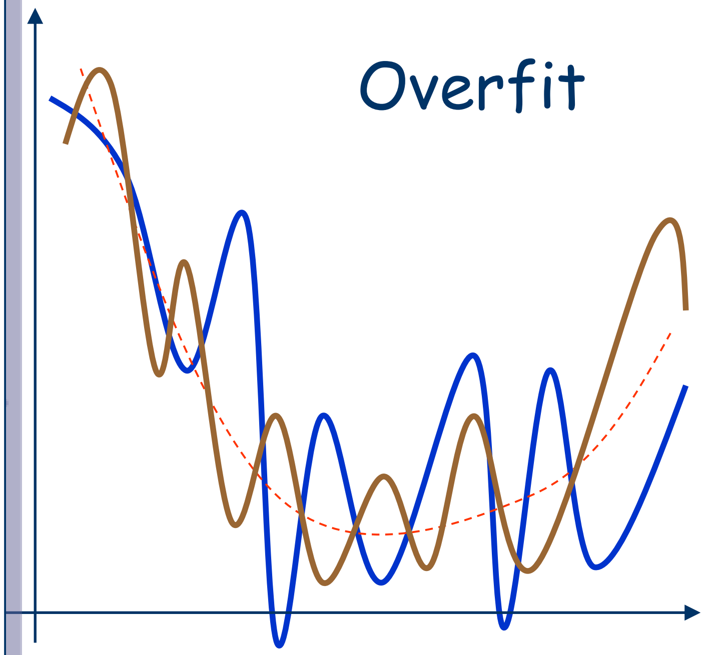
Large variance

Bias-Variance Dilemma

More on overfitting

- Easily lead to **predictions** that are far beyond the range of the training data.
- Produce wild predictions in multilayer perceptrons even with noise-free data.

Small variance

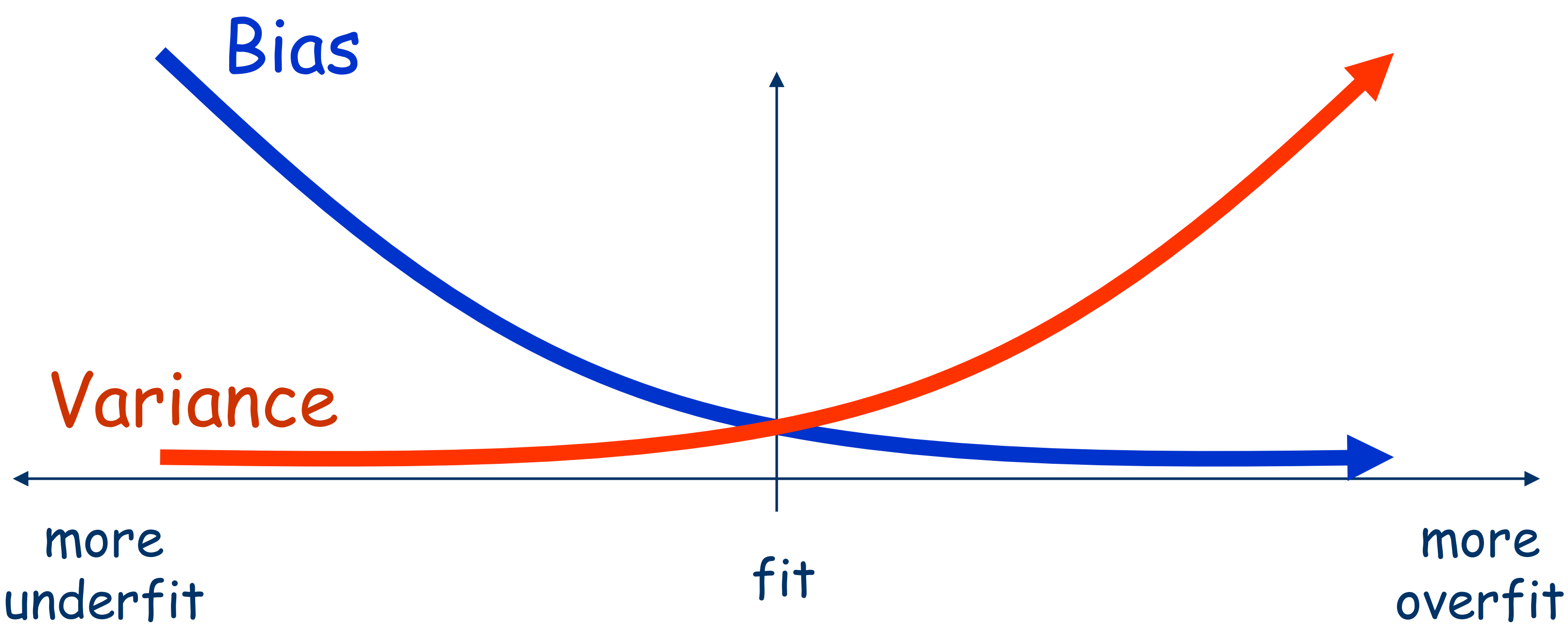


Small bias

Large variance

It's *not* really a dilemma.

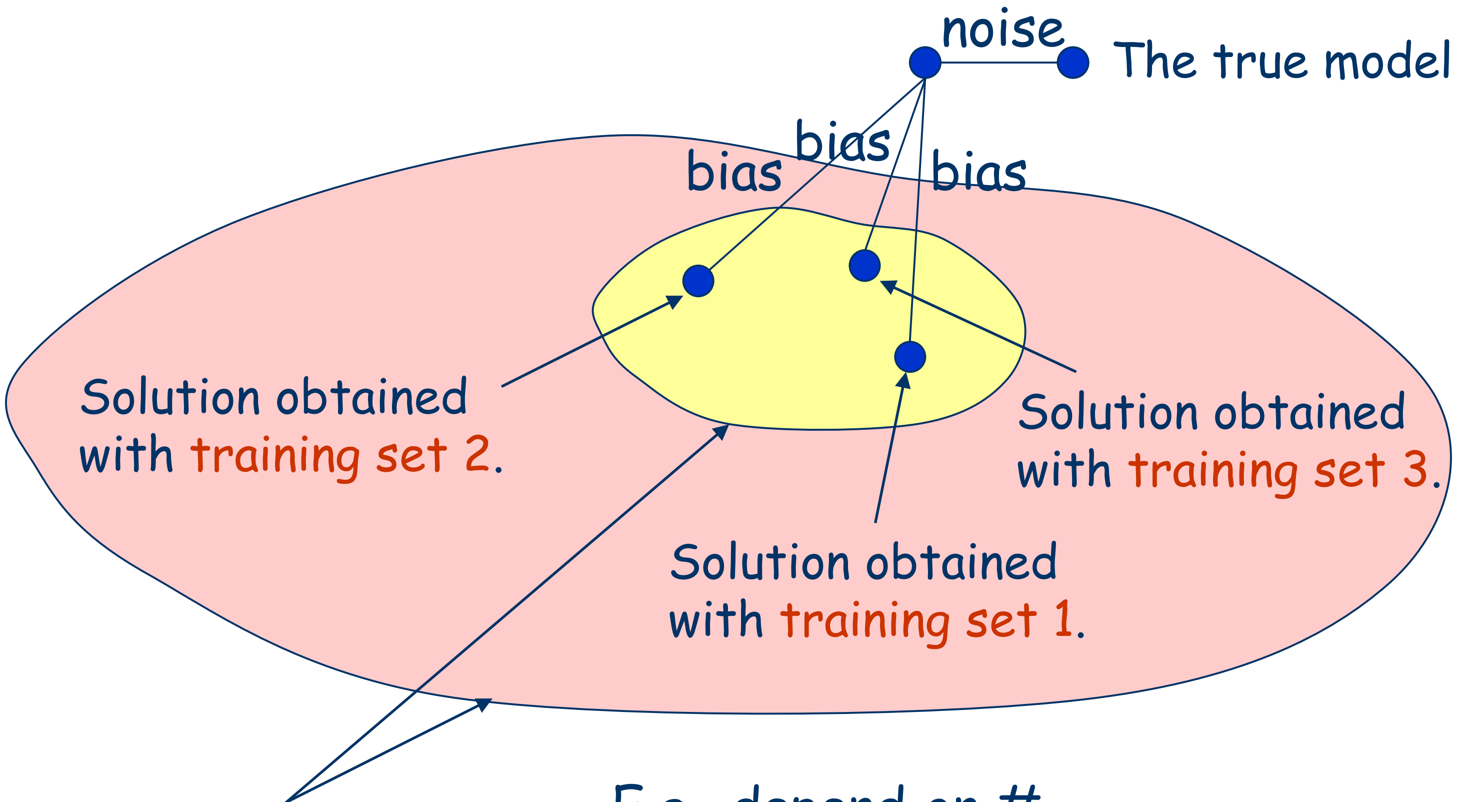
Bias-Variance Dilemma



The mean of the bias=?

The variance of the bias=?

Bias-Variance Dilemma



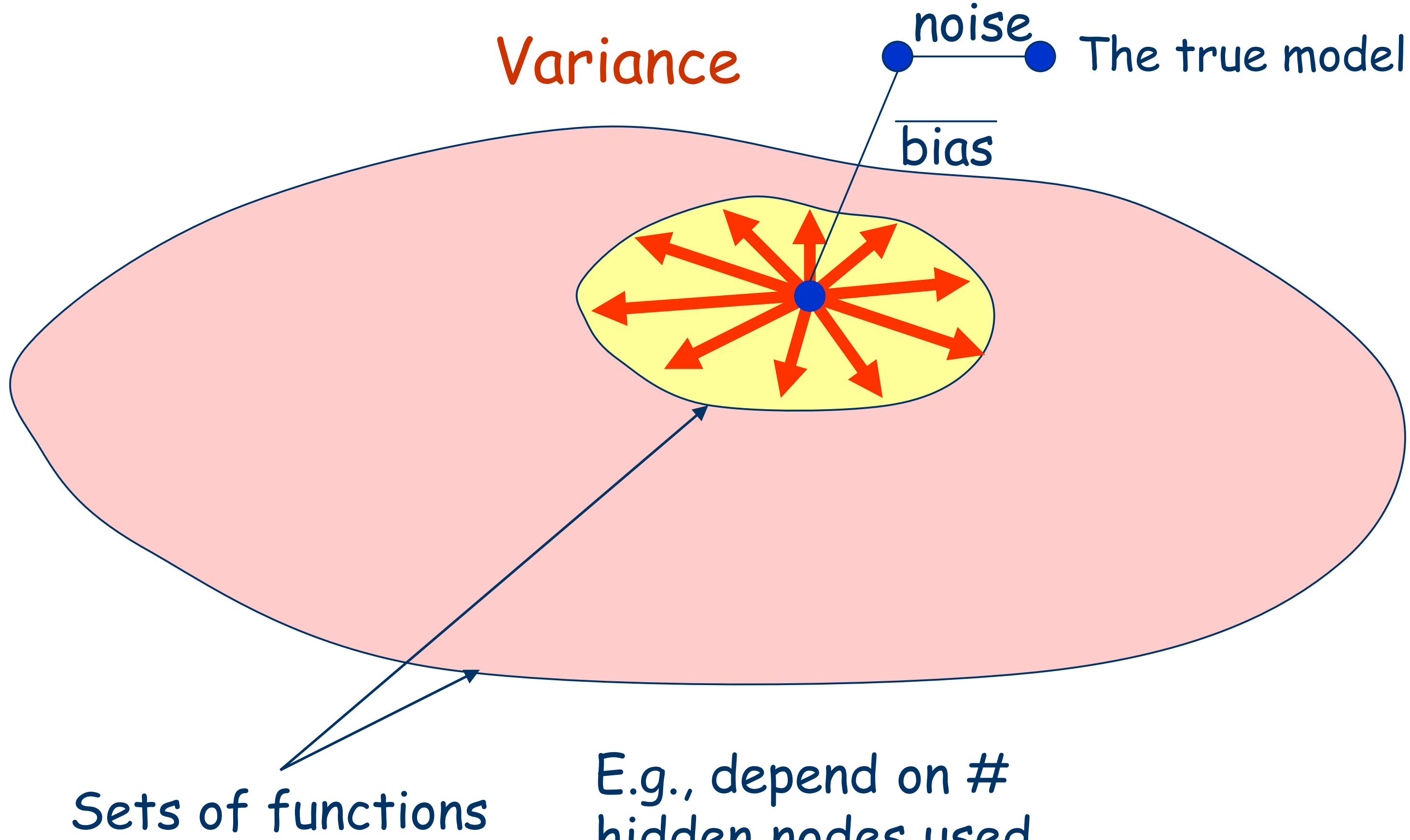
Sets of functions

E.g., depend on # hidden nodes used.

The mean of the bias=?

The variance of the bias=?

Bias-Variance Dilemma

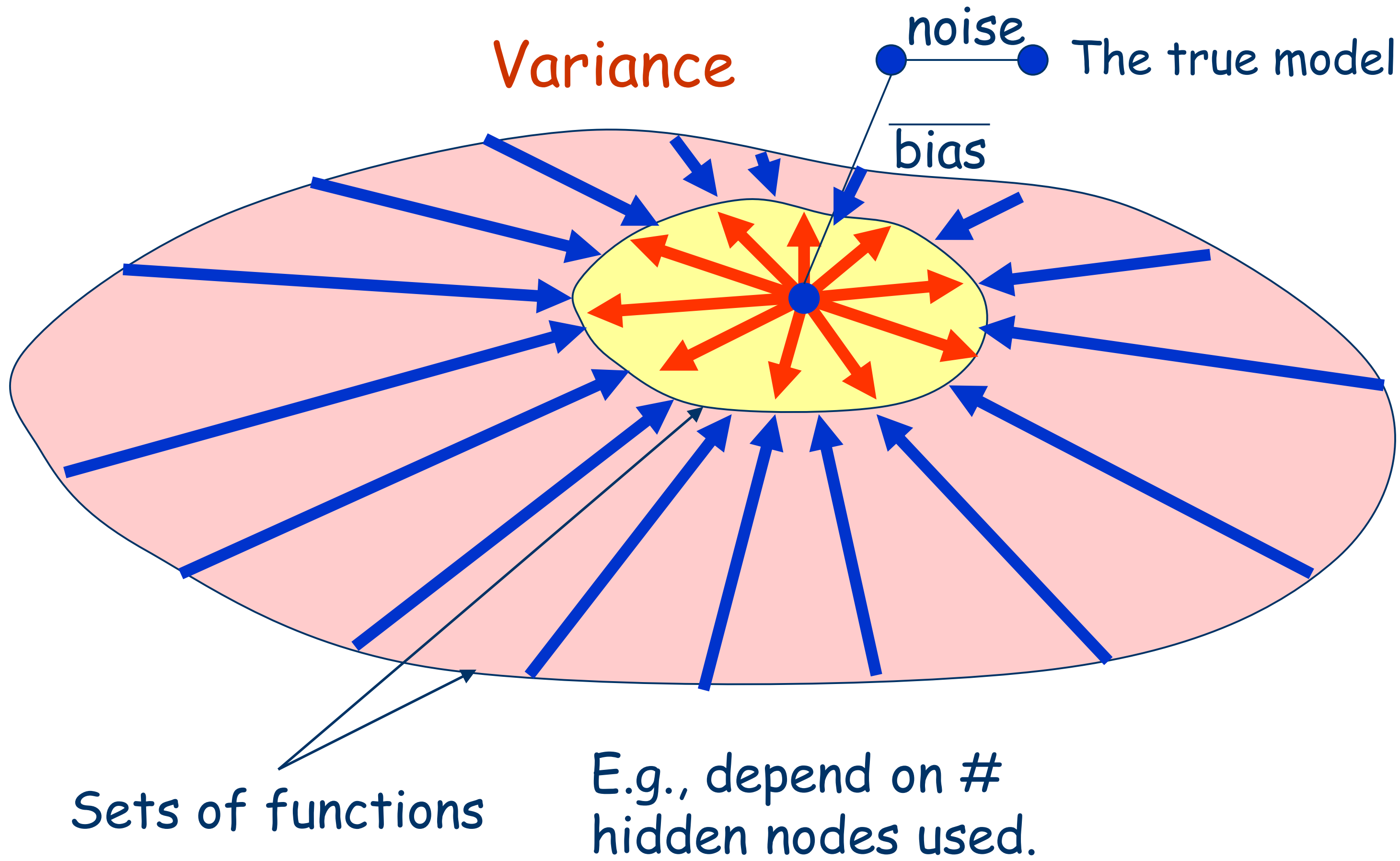


E.g., depend on # hidden nodes used.

Reduce the effective number of parameters.

Reduce the number of hidden nodes.

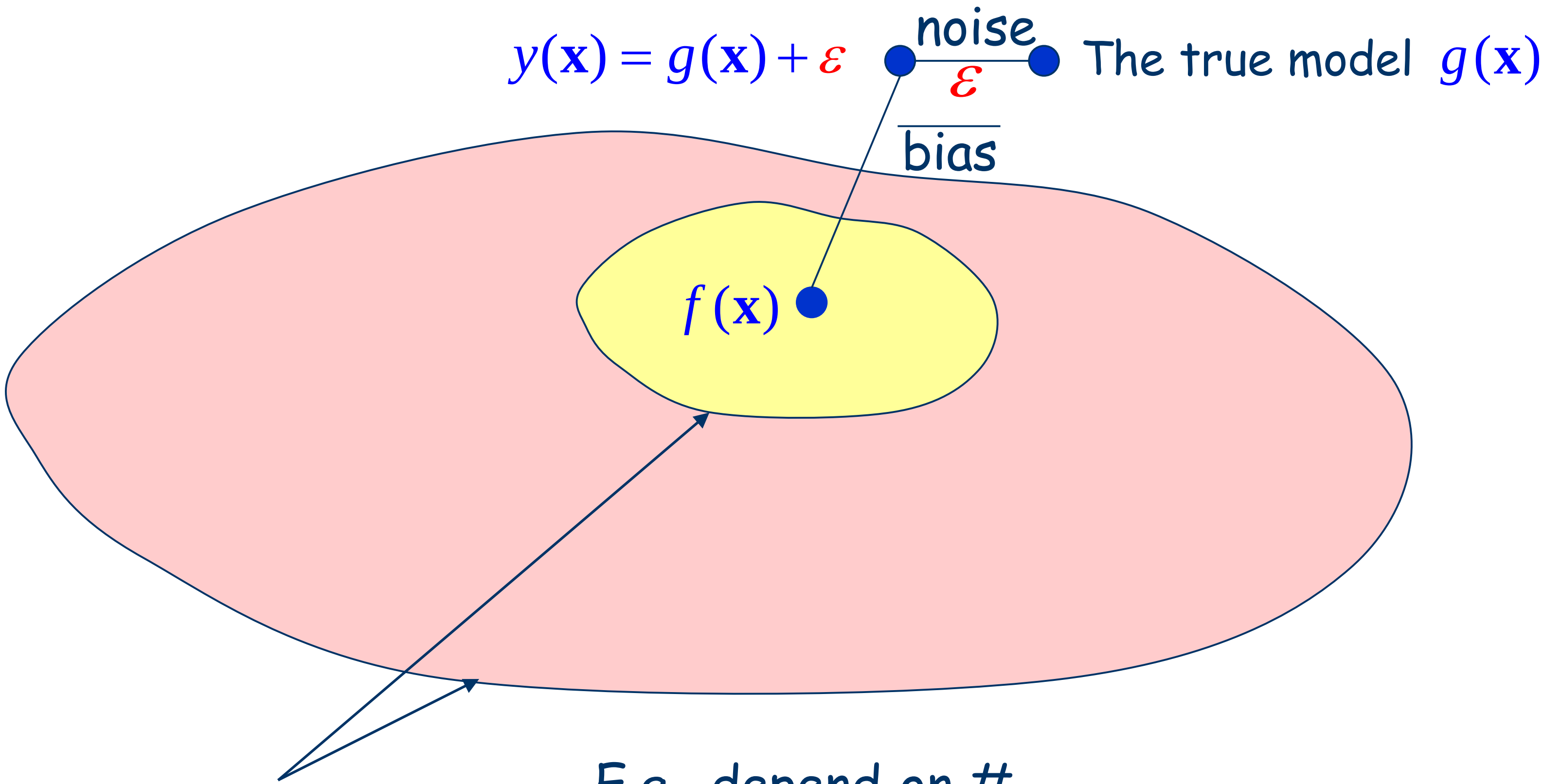
Model Selection



Goal: $\min E\left[\left(y(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$

Bias-Variance Dilemma

$y(\mathbf{x}) = g(\mathbf{x}) + \varepsilon$



Sets of functions

E.g., depend on #
hidden nodes used.

Goal: $\min E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]$

Bias-Variance Dilemma

Goal: $\min E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right] \equiv \min E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right]$

$$E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right] = E\left[\left(g(\mathbf{x})+\varepsilon-f(\mathbf{x})\right)^2\right]$$

$$= E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2 + 2\varepsilon\left(g(\mathbf{x})-f(\mathbf{x})\right)+\varepsilon^2\right]$$

$$= E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right] + \underbrace{2E[\varepsilon][g(\mathbf{x})-f(\mathbf{x})]}_{0} + \underbrace{E\left[\varepsilon^2\right]}_{\text{constant}}$$

$$E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]=E\left[\varepsilon^2\right]+\boxed{E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right]}$$

Bias-Variance Dilemma

$$E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right]=E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]+E[f(\mathbf{x})]-f(\mathbf{x})\right)^2\right]$$

$$=E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)^2+\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)^2-2\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)\right]$$

$$=E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]+E\left[\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]$$

$$\underbrace{-2E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)\right]}_{0}$$

$$E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)\right]$$

$$=E\left[g(\mathbf{x})f(\mathbf{x})\right]-E\left[g(\mathbf{x})E[f(\mathbf{x})]\right]-E\left[E[f(\mathbf{x})]f(\mathbf{x})\right]+E\left[E[f(\mathbf{x})]E[f(\mathbf{x})]\right]$$

$$=E\left[g(\mathbf{x})\right]E\left[f(\mathbf{x})\right]-E\left[g(\mathbf{x})\right]E[f(\mathbf{x})]-E[f(\mathbf{x})]E\left[f(\mathbf{x})\right]+E[f(\mathbf{x})]E[f(\mathbf{x})]=0$$

Goal: $\min E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]$

Bias-Variance Dilemma

$$E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]=E\left[\varepsilon^2\right]+E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right]$$

$$=\underbrace{E\left[\varepsilon^2\right]}_{\text{noise}}+\underbrace{E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]}_{\text{bias}^2}+\underbrace{E\left[\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]}_{\text{variance}}$$

Cannot be minimized

Minimize both bias² and variance

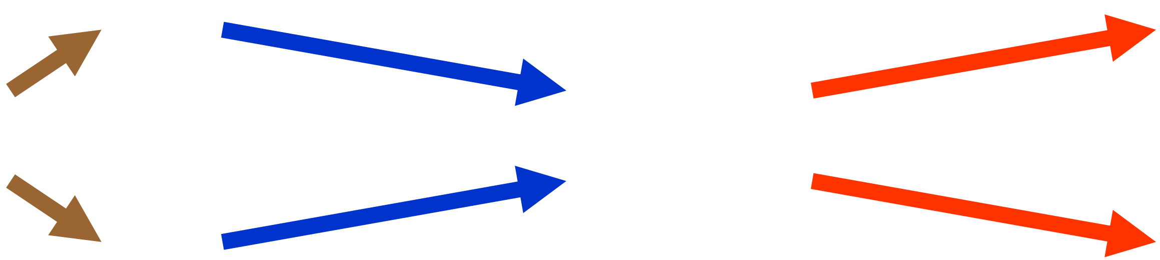
Goal: $\min E\left[\left(y(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$

Model Complexity vs. Bias-Variance

$$E\left[\left(y(\mathbf{x}) - f(\mathbf{x})\right)^2\right] = E\left[\varepsilon^2\right] + E\left[\left(g(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

$$= \underbrace{E\left[\varepsilon^2\right]}_{\text{noise}} + \underbrace{E\left[\left(g(\mathbf{x}) - E[f(\mathbf{x})]\right)^2\right]}_{\text{bias}^2} + \underbrace{E\left[\left(f(\mathbf{x}) - E[f(\mathbf{x})]\right)^2\right]}_{\text{variance}}$$

Model Complexity (Capacity)



Goal: $\min E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]$

Bias-Variance Dilemma

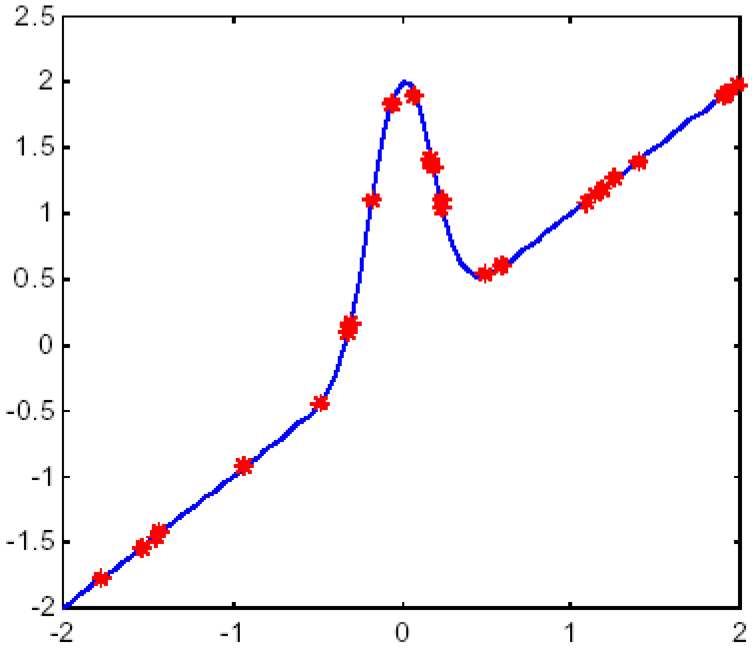
$$E\left[\left(y(\mathbf{x})-f(\mathbf{x})\right)^2\right]=E\left[\varepsilon^2\right]+E\left[\left(g(\mathbf{x})-f(\mathbf{x})\right)^2\right]$$

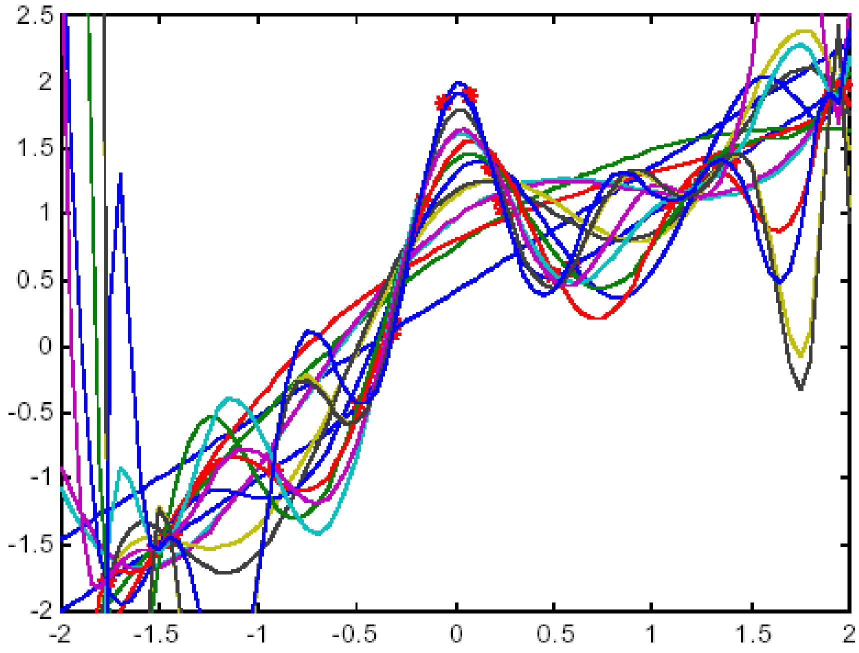
$$=\underbrace{E\left[\varepsilon^2\right]}_{\text{noise}}+\underbrace{E\left[\left(g(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]}_{\text{bias}^2}+\underbrace{E\left[\left(f(\mathbf{x})-E[f(\mathbf{x})]\right)^2\right]}_{\text{variance}}$$

Model Complexity (Capacity)

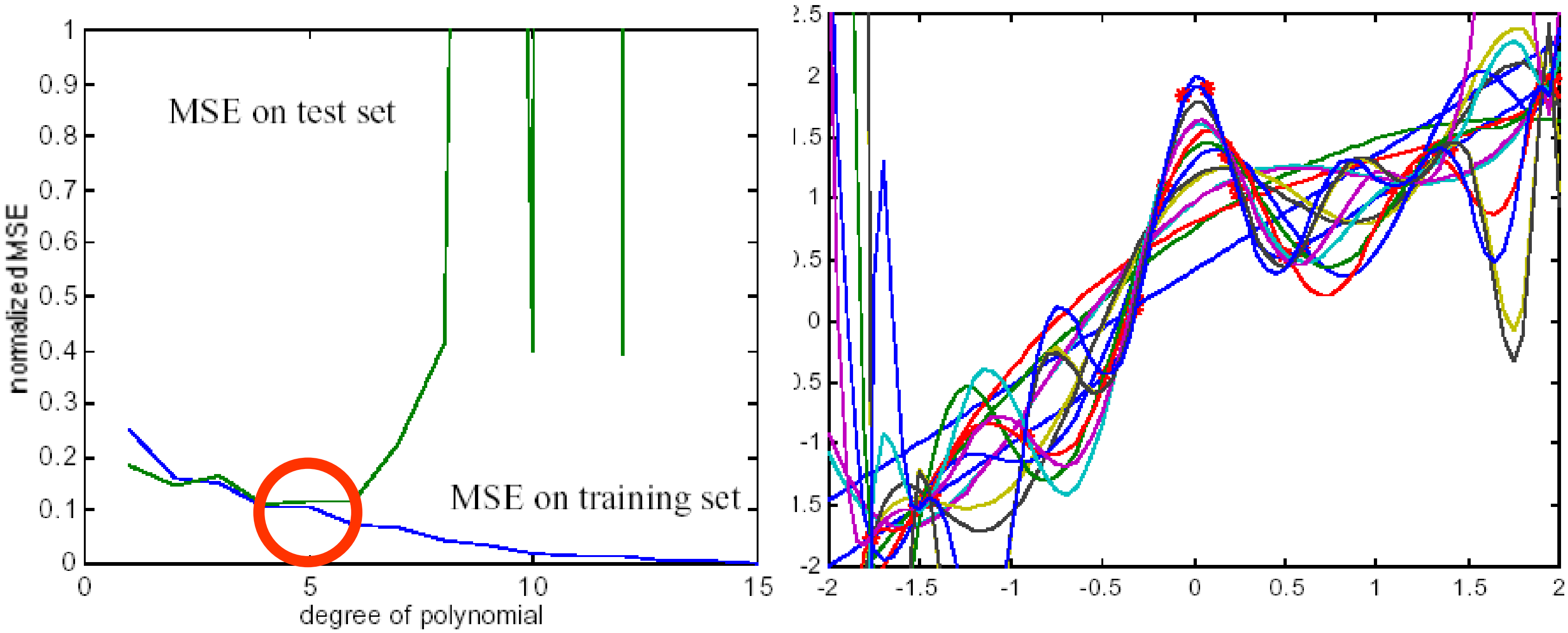
Example (Polynomial Fits)

$$y = x + 2\exp(-16x^2)$$



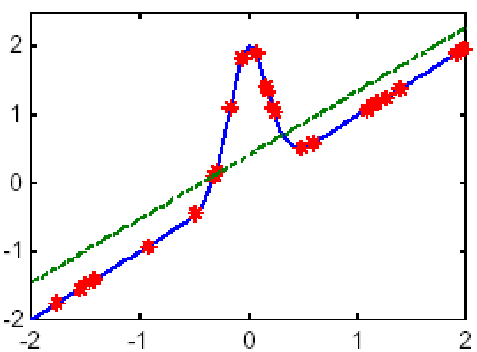


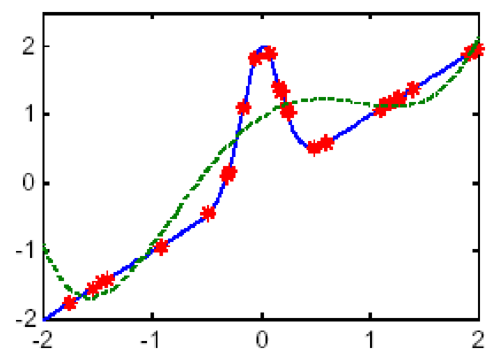
Example (Polynomial Fits)



Example (Polynomial Fits)

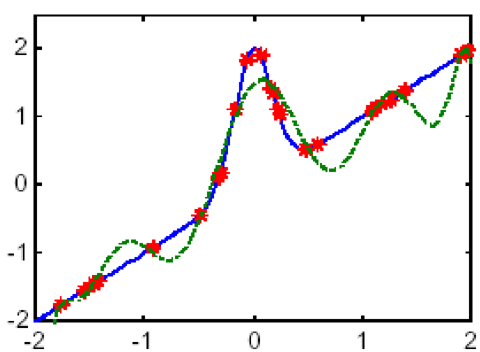
Degree 1

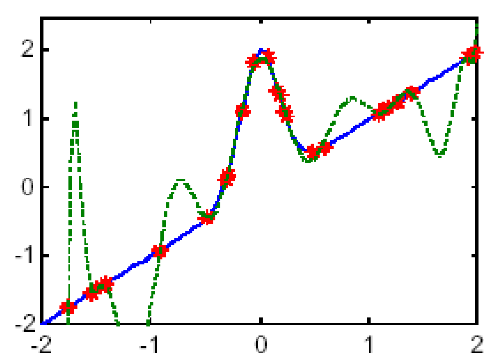




Degree 5

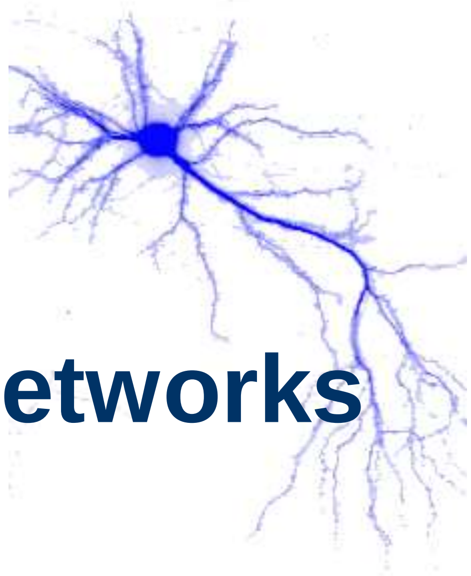
Degree 10





Degree 15

Introduction to
Radial Basis Function Networks



The Effective Number of Parameters

Variance Estimation

Mean μ —— In general, *not* available.

Variance $$\hat{\sigma}^2 = \frac{1}{p}\sum_{i=1}^{p}\left(x_i - \mu\right)^2$$

Variance Estimation

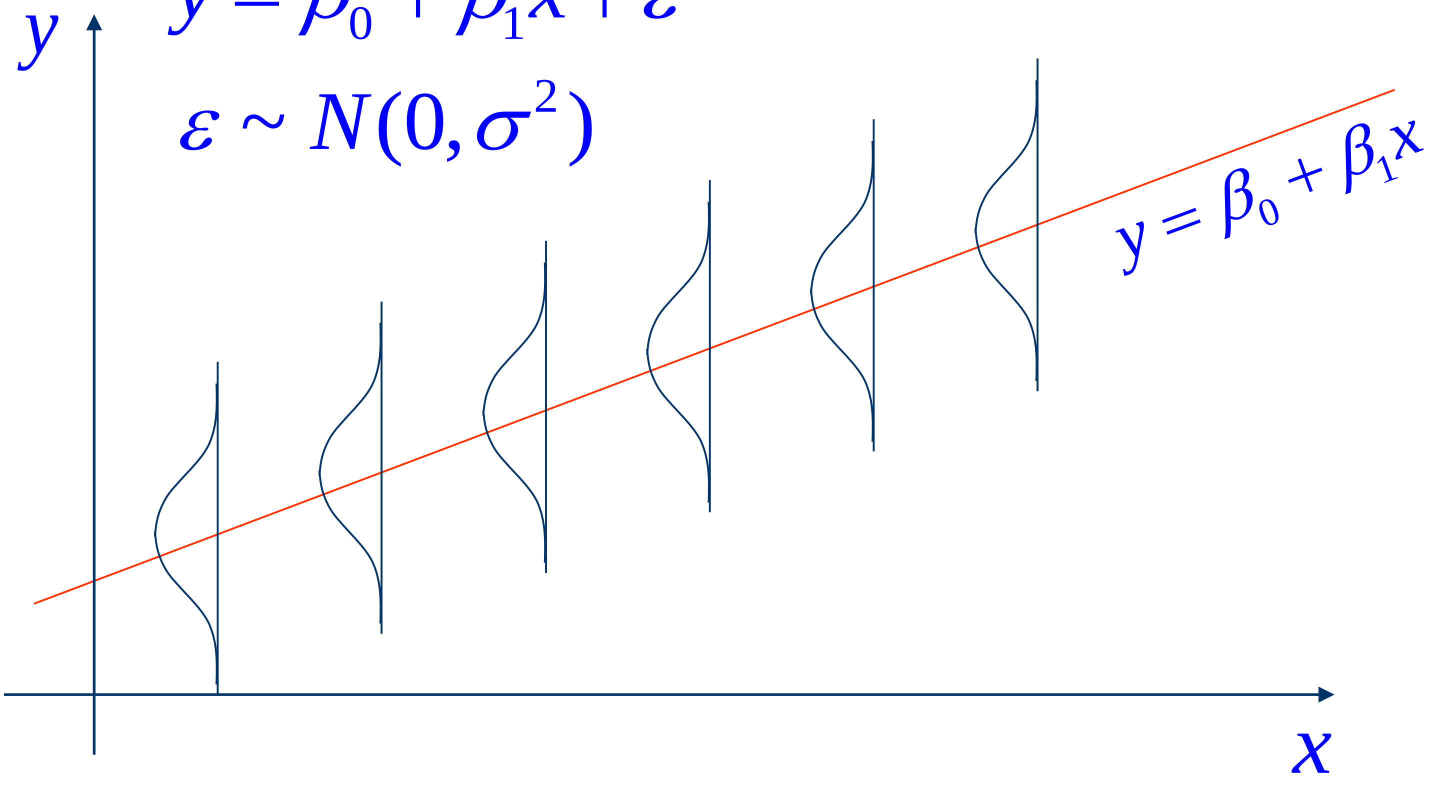
Mean $\hat{\mu} = \bar{x} = \frac{1}{p}\sum_{i=1}^{p} x_i$

Variance $\hat{\sigma}^2 = s^2 = \frac{1}{\underbrace{p-1}_{\text{Loss 1 degree of freedom}}}\sum_{i=1}^{p}\left(x_i - \hat{\mu}\right)^2$

Simple Linear Regression

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

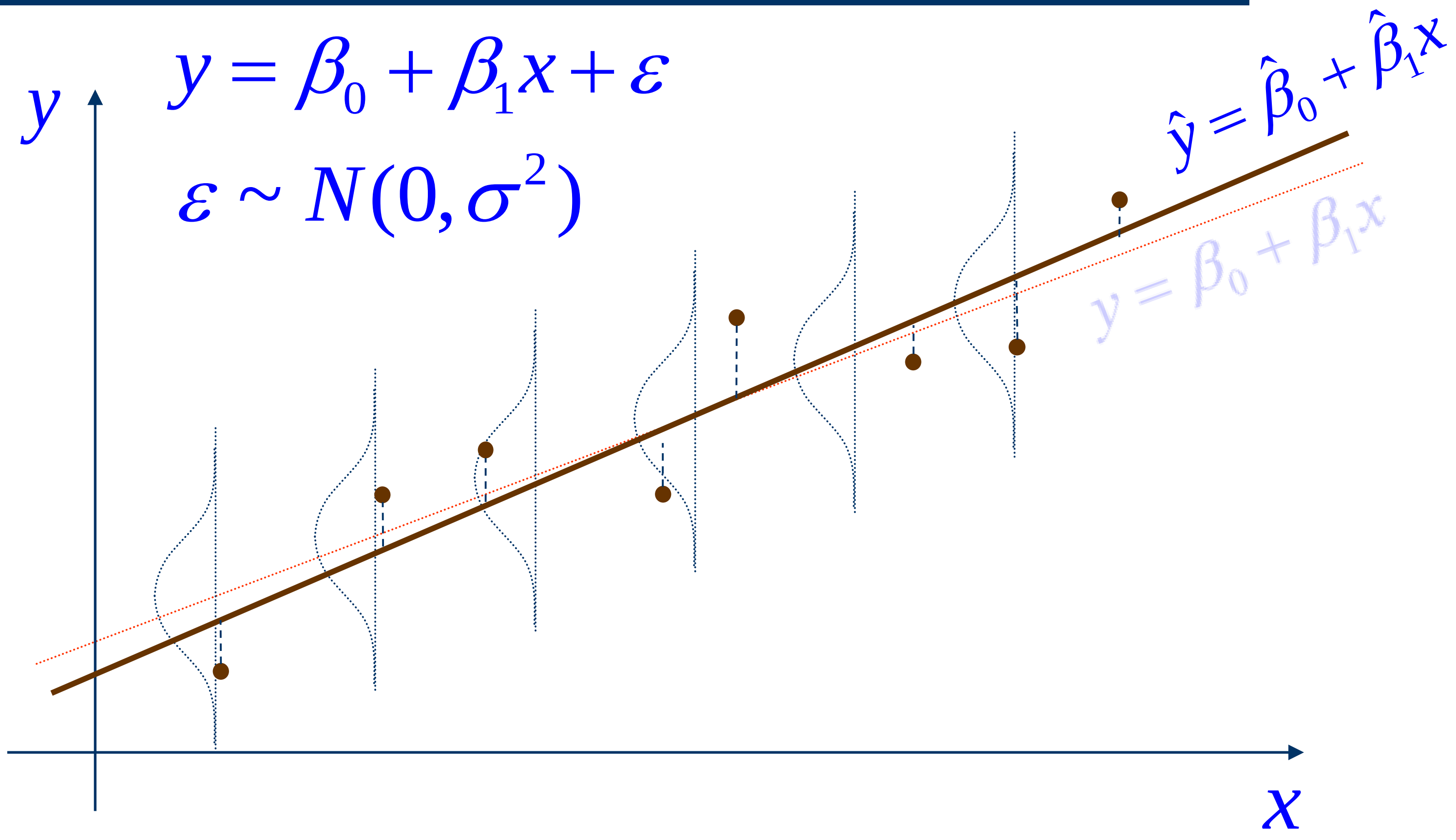


Minimize $SSE = \sum_{i=1}^{p} \left(y_i - \hat{y}_i \right)^2$

Simple Linear Regression

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

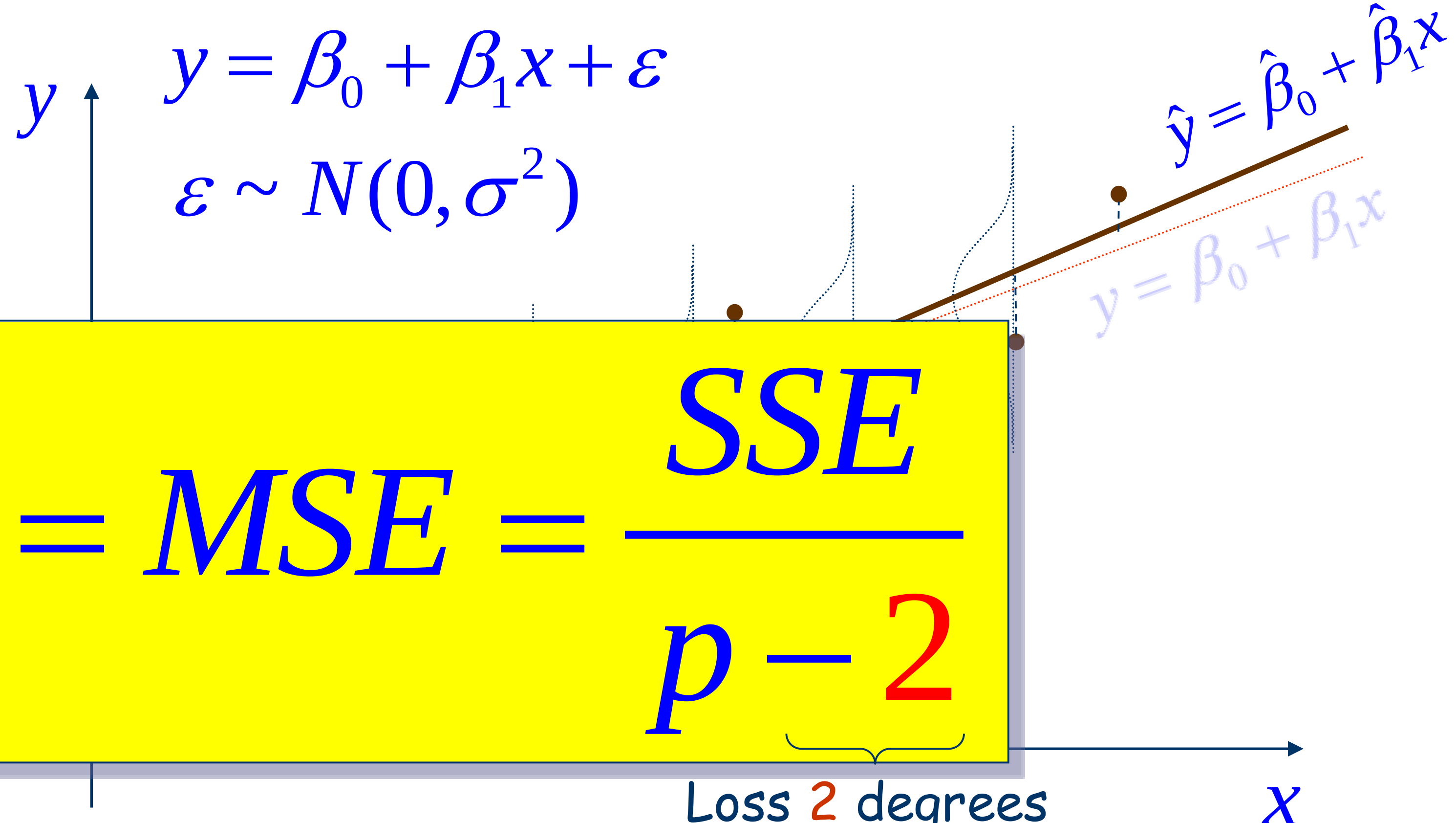


Minimize $SSE = \sum_{i=1}^{p} (y_i - \hat{y}_i)^2$

Mean Squared Error (MSE)

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$



$$\hat{\sigma}^2 = MSE = \frac{SSE}{p - 2}$$

Loss 2 degrees of freedom

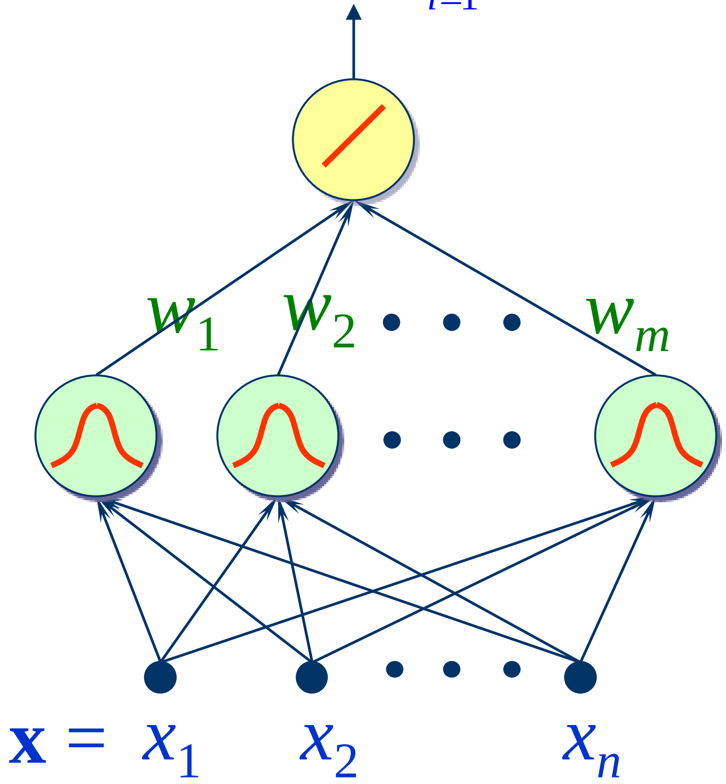
Variance Estimation

$$\hat{\sigma}^2 = MSE = \frac{SSE}{\underbrace{p - m}_{\text{Loss } m \text{ degrees of freedom}}}$$

m: #parameters of the model

The Number of Parameters

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

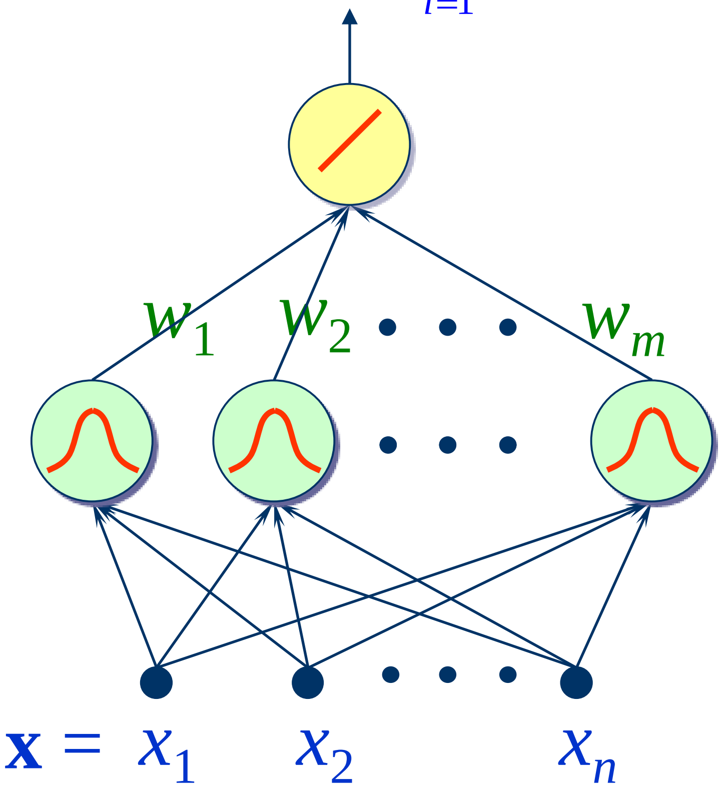


#degrees of freedom:

m

The Effective Number of Parameters (γ)

$$y = f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$



The projection Matrix

$$\mathbf{P} = \mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T$$

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{\Lambda}$$

$$\mathbf{\Lambda} = \mathbf{0} \Rightarrow$$

$$\gamma = p - trace(\mathbf{P}) = m$$

Facts: $trace(\mathbf{A}+\mathbf{B}) = trace(\mathbf{A}) + trace(\mathbf{B})$

$trace(\mathbf{AB}) = trace(\mathbf{BA})$

The Effective Number of Parameters (γ)

Pf)

$$trace(\mathbf{P}) = trace\left(\mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T\right)$$
$$= p - trace\left(\mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T\right)$$
$$= p - trace\left(\mathbf{\Phi}\left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\right)$$
$$= p - trace\left(\left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{\Phi}\right)$$
$$= p - trace\left(\mathbf{I}_m\right)$$
$$= p - m$$

<u>The projection Matrix</u>

$$\mathbf{P} = \mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T$$

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{\Lambda}$$

$$\mathbf{\Lambda} = \mathbf{0} \Rightarrow$$

$$\gamma = p - trace(\mathbf{P}) = m$$

The effective number of parameters:

$$\gamma = p - trace(\mathbf{P})$$

Regularization

SSE

Penalize models with large weights

$$Cost = \begin{matrix} Empirial \\ Error \end{matrix} + \begin{matrix} Model's \\ penalty \end{matrix}$$

$$\sum_{k=1}^{p}\left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{i=1}^{m} \lambda_i w_i^2$$

The effective number of parameters:

$$\gamma = p - trace(\mathbf{P})$$

Regularization

SSE

$$Cost = \begin{matrix} Empirial \\ Error \end{matrix}$$

$$\sum_{k=1}^{p}\left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2$$

Without penalty (λ_i=0), there are m degrees of freedom to minimize SSE (Cost).

The effective number of parameters γ=m.

The effective number of parameters:

$$\gamma = p - trace(\mathbf{P})$$

Regularization

SSE

$$Cost = \text{Empirial Error}$$

$$\sum_{k=1}^{p}\left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i\phi_i\left(\mathbf{x}^{(k)}\right)\right]^2$$

Penalize models with large weights

Model's penalty

$$\sum_{i=1}^{m}\lambda_i w_i^2$$

With penalty ($\lambda_i>0$), the liberty to minimize SSE will be reduced.

The effective number of parameters $\gamma<m$.

The effective number of parameters:

$$\gamma = p - trace(\mathbf{P})$$

Variance Estimation

$$\hat{\sigma}^2 = MSE = \frac{SSE}{p - \underbrace{\gamma}_{}}$$

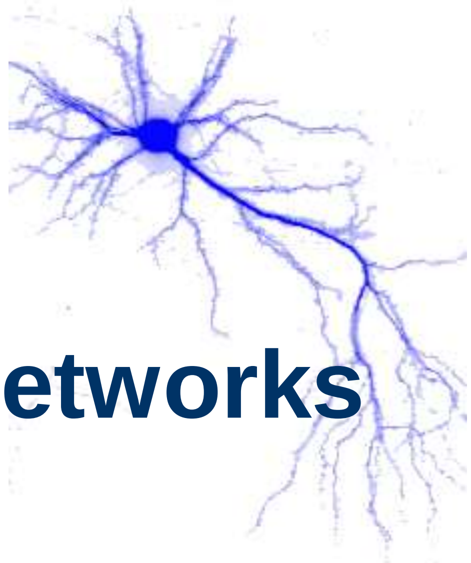
Loss γ degrees of freedom

The effective number of parameters:

$$\gamma = p - trace(\mathbf{P})$$

Variance Estimation

$$\hat{\sigma}^2 = MSE = \frac{SSE}{trace(\mathbf{P})}$$



Introduction to Radial Basis Function Networks

Model Selection

Model Selection

- Goal
  - Choose the fittest model
- Criteria
  - Least prediction error
- Main Tools (Estimate Model Fitness)
  - Cross validation
  - Projection matrix
- Methods
  - Weight decay (Ridge regression)
  - Pruning and Growing RBFN's

Empirical Error vs. Model Fitness

- Ultimate Goal – Generalization

  Minimize Prediction Error

- Goal of Our Learning Procedure

  Minimize Empirical Error (MSE)

  $$\nparallel$$

  Minimize Prediction Error

Estimating Prediction Error

- When you have plenty of data use independent test sets
  - E.g., use the same training set to train different models, and choose the best model by comparing on the test set.
- When data is scarce, use
  - Cross-Validation
  - Bootstrap

Cross Validation

- Simplest and most widely used method for estimating prediction error.

- Partition the original set into several different ways and to compute an average score over the different partitions, e.g.,
  - K-fold Cross-Validation
  - Leave-One-Out Cross-Validation
  - Generalize Cross-Validation

K-Fold CV

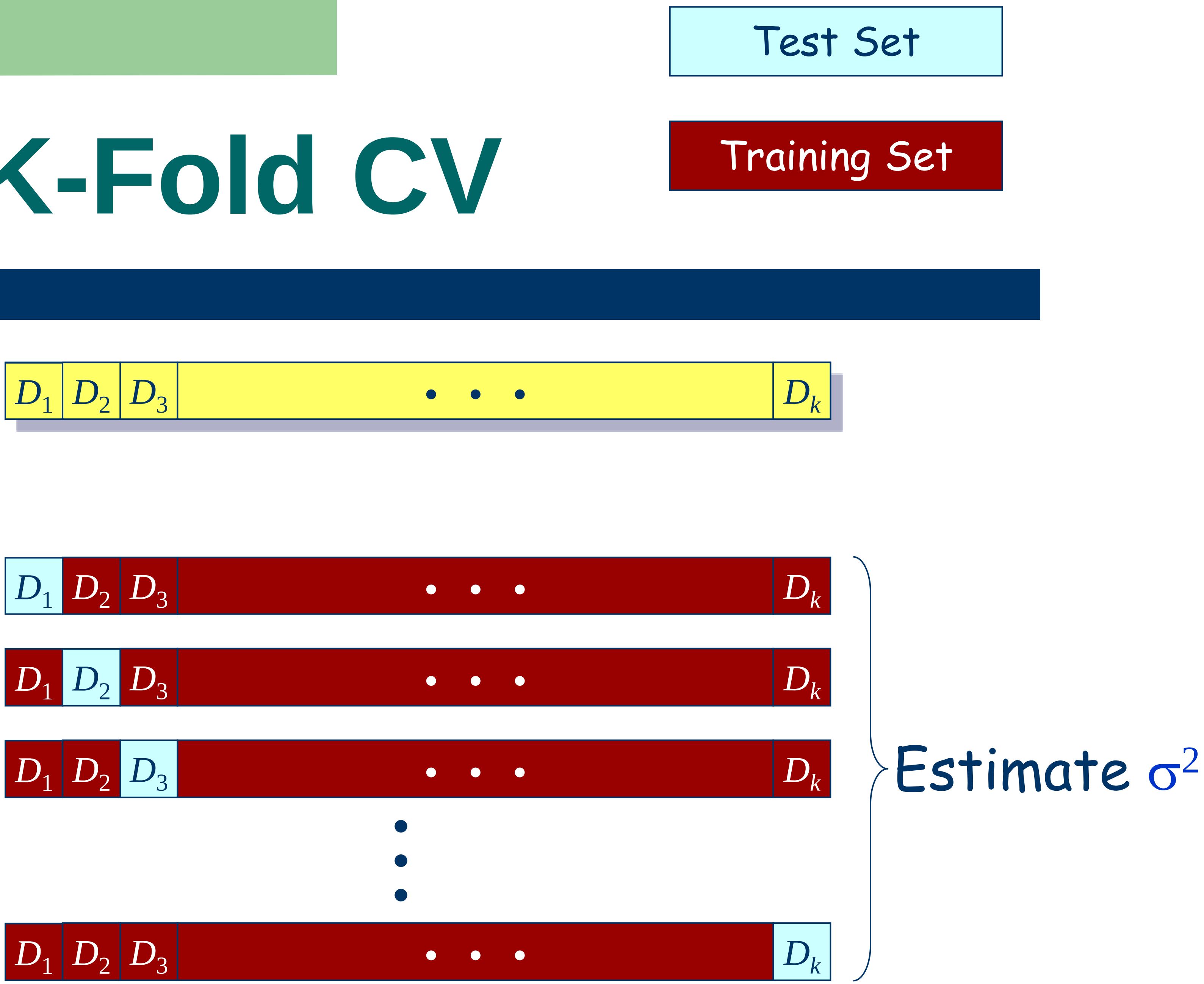
- Split the set, say, D of available input-output patterns into k mutually exclusive subsets, say $D_1, D_2, \ldots, D_k$.

- Train and test the learning algorithm k times, each time it is trained on $D \backslash D_i$ and tested on D_i.

K-Fold CV

Available Data

K-Fold CV


Test Set
Training Set
D_1 D_2 D_3 • • • D_k
D_1 D_2 D_3 • • • D_k
D_1 D_2 D_3 • • • D_k
D_1 D_2 D_3 • • • D_k
D_1 D_2 D_3 • • • D_k
Estimate σ^2

A special case of k-fold CV.

Leave-One-Out CV

- Split the p available input-output patterns into a training set of size p–1 and a test set of size 1.
- Average the squared error on the left-out pattern over the p possible ways of partition.

A special case of k-fold CV.

Error Variance Predicted by LOO

$D = \{(\mathbf{x}_k, y_k) : k = 1, 2, \ldots, p\}$ — Available input-output patterns.

$D_i = D \setminus \{(\mathbf{x}_i, y_i)\} \quad i = 1, \ldots, p$ — Training sets of LOO.

f_i — Function learned using D_i as training set.

The estimate for the variance of prediction error using LOO:

$$\hat{\sigma}^2_{LOO} = \frac{1}{p} \sum_{i=1}^{p} \underbrace{\left(y_i - f_i(\mathbf{x}_i) \right)^2}_{\text{Error-square for the left-out element.}}$$

A special case of k-fold CV.

Error Va… …O

Given a model, the function with least empirical error for D_i.

$D=\{(\mathbf{x}_k, y_k): k=1,2,\ldots,p\}$ — Available input-output patterns.

$D_i = D \setminus \{(\mathbf{x}_i, y_i)\} \quad i=1,\ldots,p$ — Training sets of LOO.

f_i — Function learned using D_i as training set.

As an index of model's fitness.
We want to find a model also minimize this.

Th… using LOO:

$$\hat{\sigma}^2_{LOO} = \frac{1}{p}\sum_{i=1}^{p}\underbrace{\left(y_i - f_i(\mathbf{x}_i)\right)^2}_{\text{Error-square for the left-out element.}}$$

A special case of k-fold CV.

Error Variance Predicted by LOO

$D = \{(\mathbf{x}_k, y_k) : k = 1, 2, \ldots, p\}$ — Available input-output patterns.

$D_i = D \setminus \{(\mathbf{x}_i, y_i)\} \quad i = 1, \ldots, p$ — Training sets of LOO.

f_i — … using D_i as training…

How to estimate?

Are there any efficient ways?

The estimate for the … of prediction error using LOO:

$$\hat{\sigma}^2_{LOO} = \frac{1}{p} \sum_{i=1}^{p} \underbrace{\left(y_i - f_i(\mathbf{x}_i)\right)^2}_{\text{Error-square for the left-out element.}}$$

Error Variance Predicted by LOO

$$\hat{\sigma}^2_{LOO} = \frac{1}{p}\hat{\mathbf{y}}^T \mathbf{P}\left(diag(\mathbf{P})\right)^{-2}\mathbf{P}\hat{\mathbf{y}}$$

$$\hat{\sigma}^2_{LOO} = \frac{1}{p}\sum_{i=1}^{p}\underbrace{\left(y_i - f_i(\mathbf{x}_i)\right)^2}_{\text{Error-square for the left-out element.}}$$

$$\hat{\sigma}^2_{LOO} = \frac{1}{p}\hat{\mathbf{y}}^T \mathbf{P}(diag(\mathbf{P}))^{-2}\mathbf{P}\hat{\mathbf{y}}$$

Generalized Cross-Validation

$$diag(\mathbf{P}) \rightarrow \frac{trace(\mathbf{P})}{p}\mathbf{I}_p$$

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{\left[trace(\mathbf{P})\right]^2} = \frac{p\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{(p-\gamma)^2}$$

More Criteria Based on CV

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{(p-\gamma)^2}$$

GCV
(Generalized CV)

$$\hat{\sigma}^2_{UEV} = \frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p-\gamma}$$

UEV
(Unbiased estimate of variance)

Akaike's Information Criterion

$$\hat{\sigma}^2_{FPE} = \frac{p+\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p}$$

FPE
(Final Prediction Error)

$$\hat{\sigma}^2_{BIC} = \frac{p+\left(\ln(p)-1\right)\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p}$$

BIC
(Bayesian Information Criterio)

$$\hat{\sigma}^2_{UEV} \le \hat{\sigma}^2_{FPE} \le \hat{\sigma}^2_{GCV} \le \hat{\sigma}^2_{BIC}$$

More Criteria Based on CV

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{(p-\gamma)^2} = \frac{p}{p-\gamma}\hat{\sigma}^2_{UEV}$$

$$\hat{\sigma}^2_{UEV} = \frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p-\gamma}$$

$$\hat{\sigma}^2_{FPE} = \frac{p+\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p} = \frac{p+\gamma}{p}\hat{\sigma}^2_{UEV}$$

$$\hat{\sigma}^2_{BIC} = \frac{p+\left(\ln(p)-1\right)\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p} = \frac{p+\left(\ln(p)-1\right)\gamma}{p}\hat{\sigma}^2_{UEV}$$

$$\hat{\sigma}^2_{UEV} \le \hat{\sigma}^2_{FPE} \le \hat{\sigma}^2_{GCV} \le \hat{\sigma}^2_{BIC}$$

More Criteria Based on CV

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{(p-\gamma)^2} = \frac{p}{p-\gamma}\hat{\sigma}^2_{UEV}$$

$$\hat{\sigma}^2_{UEV} = \frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p-\gamma}$$

$$\mathbf{P} = ?$$

$$\hat{\sigma}^2_{FPE} = \frac{p+\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p} = \frac{p+\gamma}{p}\hat{\sigma}^2_{UEV}$$

$$\hat{\sigma}^2_{BIC} = \frac{p+\left(\ln(p)-1\right)\gamma}{p-\gamma}\frac{\hat{\mathbf{y}}^T\mathbf{P}^2\hat{\mathbf{y}}}{p} = \frac{p+\left(\ln(p)-1\right)\gamma}{p}\hat{\sigma}^2_{UEV}$$

Regularization

Standard Ridge Regression,

$$\lambda = \lambda_i, \forall i$$

SSE

Penalize models with large weights

$$Cost = \frac{Empirial}{Error} + \frac{Model's}{penalty}$$

$$\sum_{k=1}^{p}\left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2$$

$$\sum_{i=1}^{m} \lambda_i w_i^2$$

Standard Ridge Regression,

$\lambda = \lambda_i, \forall i$

Regularization

SSE

Penalize models with large weights

$$Cost = \begin{matrix} Empirial \\ Error \end{matrix} + \begin{matrix} Model's \\ penalty \end{matrix}$$

$$\sum_{k=1}^{p} \left[y^{(k)} - f\left(\mathbf{x}^{(k)}\right) \right]^2$$

$$\sum_{k=1}^{p} \left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i \left(\mathbf{x}^{(k)}\right) \right]^2$$

$$\boxed{\sum_{i=1}^{m} \lambda w_i^2}$$

$$\min \quad C = \sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2 + \sum_{i=1}^{m} \lambda w_i^2$$

Solution Review

$$\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I}_m$$

$$\mathbf{P} = \mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T$$

Used to compute model selection criteria

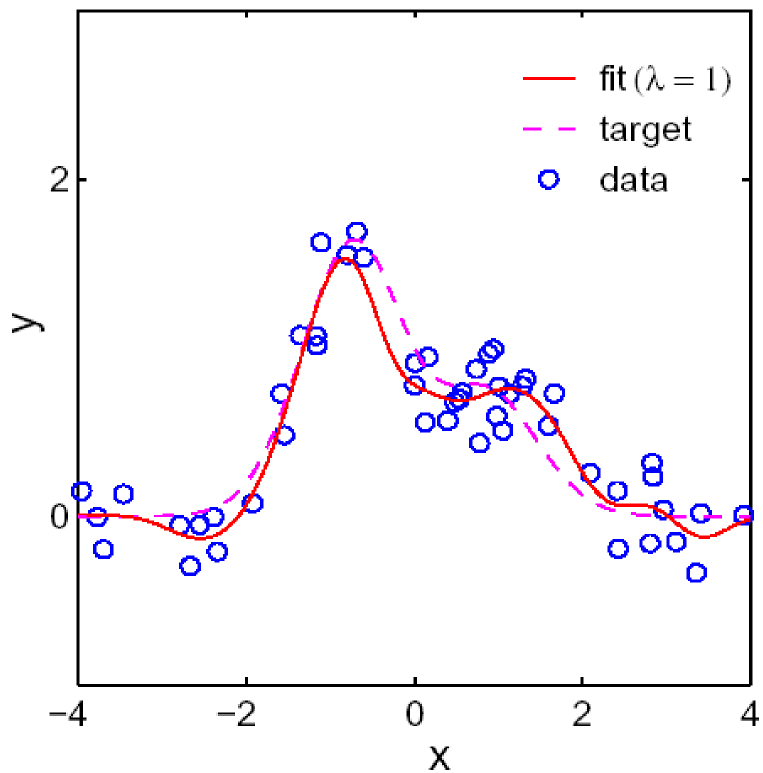
Example

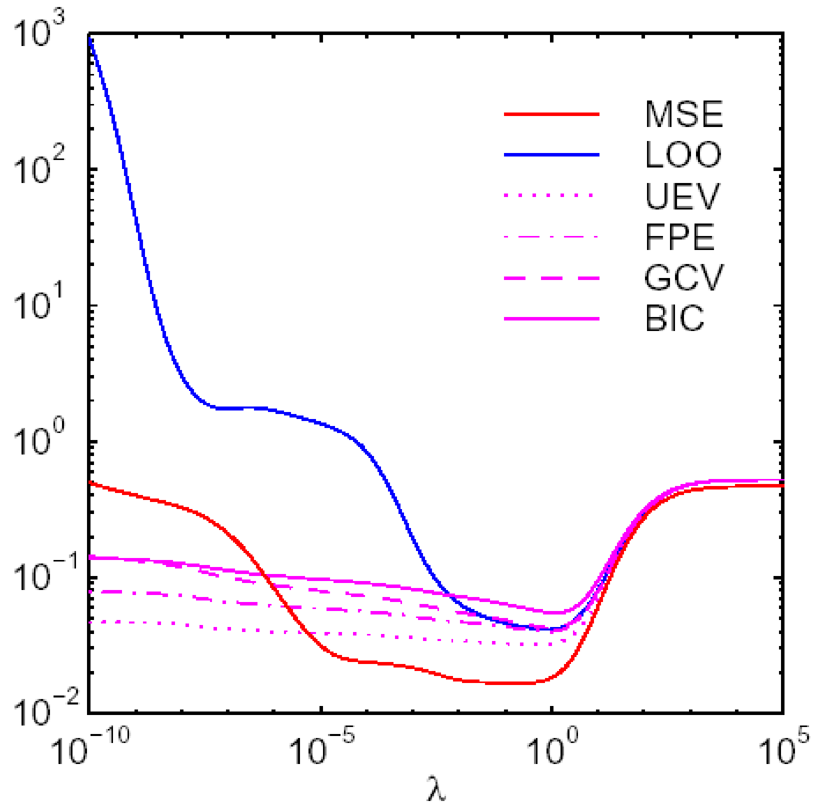
$$y(x) = (1 + x - 2x^2)e^{-x^2} + \varepsilon$$

$$\varepsilon \sim N(0, 0.2^2)$$

$$p = 50$$

Width of RBF $r = 0.5$





Example

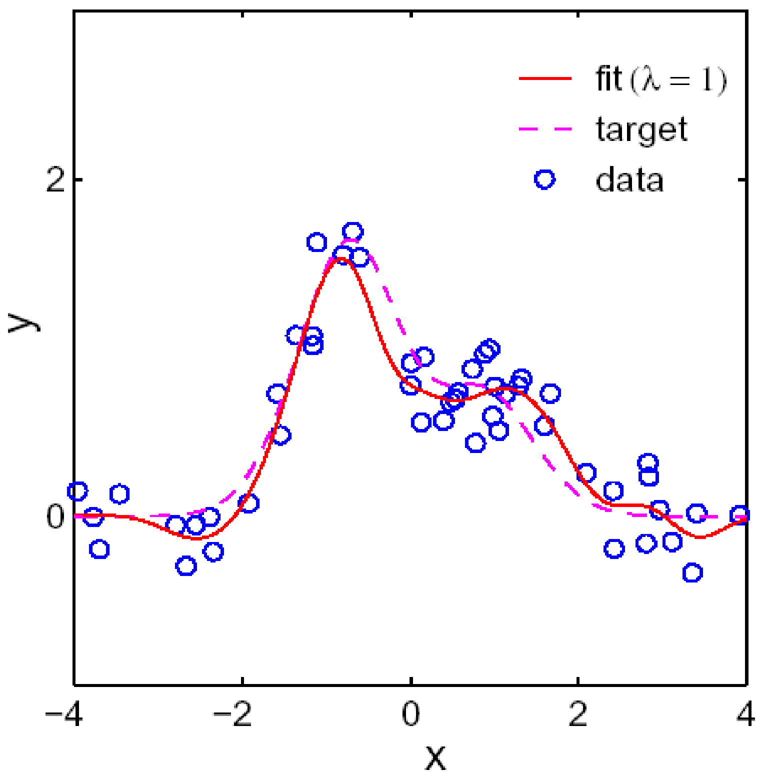
$$y(x) = (1 + x - 2x^2)e^{-x^2} + \varepsilon$$

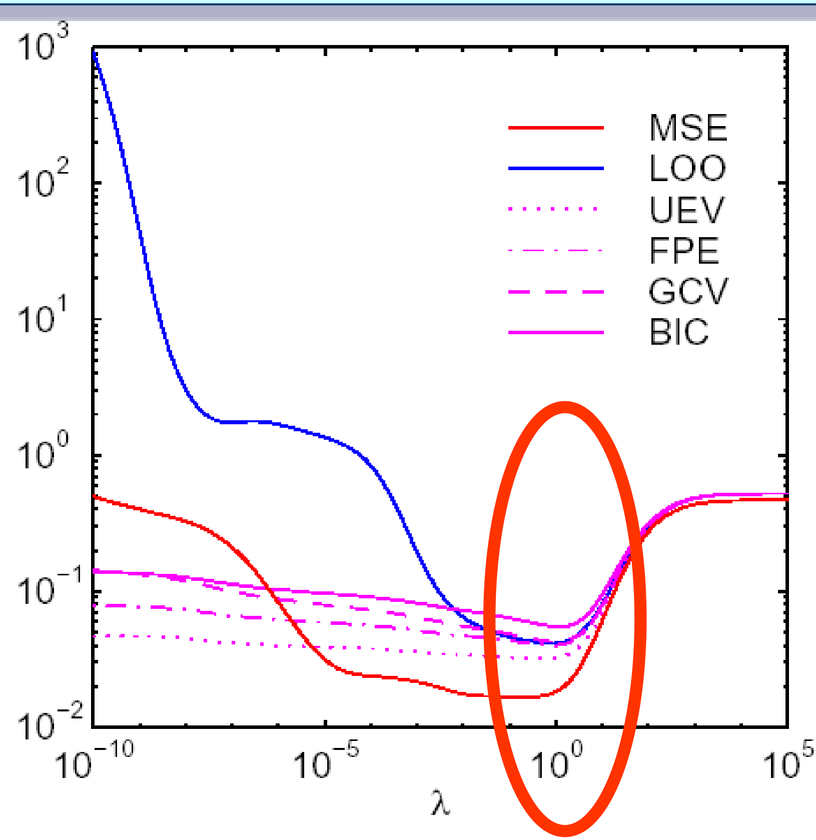
$$\varepsilon \sim N(0, 0.2^2)$$

$$p = 50$$

Width of RBF $r = 0.5$

$$\hat{\sigma}^2_{UEV} \le \hat{\sigma}^2_{FPE} \le \hat{\sigma}^2_{GCV} \le \hat{\sigma}^2_{BIC}$$





Example

$$y(x) = (1 + x - 2x^2)e^{-x^2} + \varepsilon$$

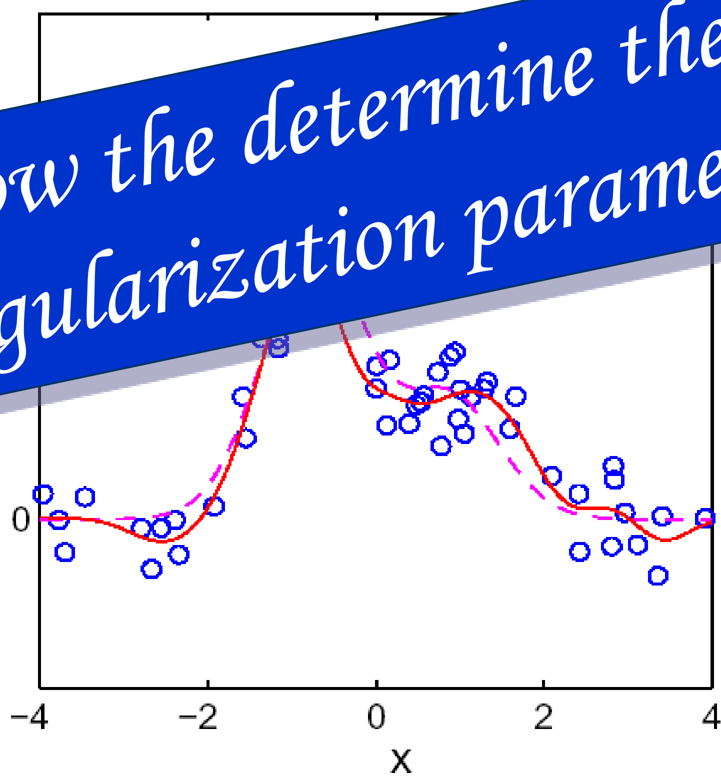
$$\varepsilon \sim N(0, 0.2^2)$$

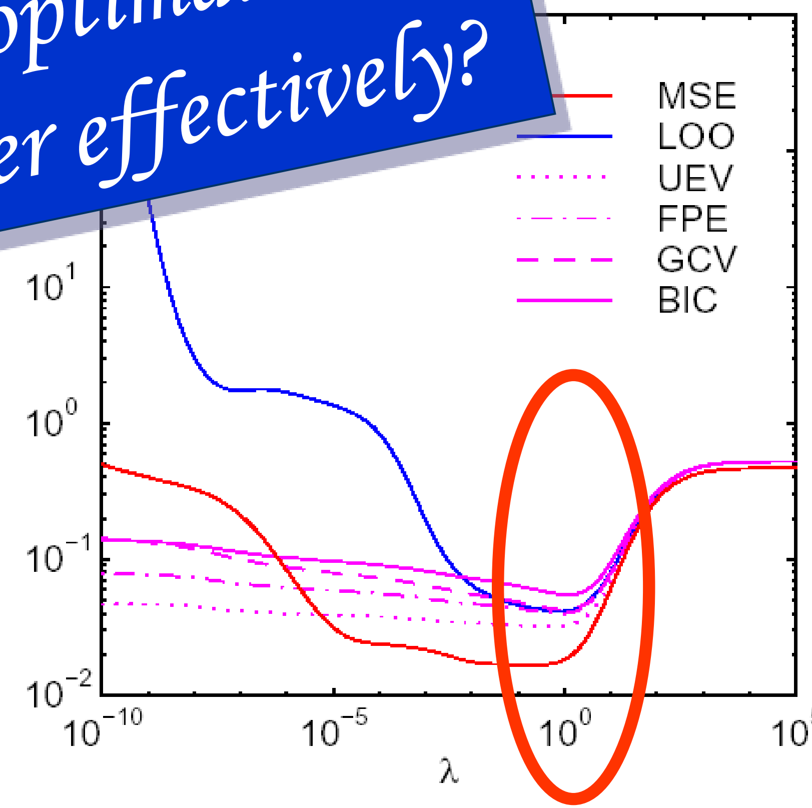
$$p = 50$$

Width of RBF $r = 0.5$

$\hat{\sigma}^2_{UEV} \le \hat{\sigma}^2 \ldots \hat{\sigma}^2_{GCV} \le \hat{\sigma}^2_{BIC}$

How the determine the optimal regularization parameter effectively?





Optimizing the Regularization Parameter

Re-Estimation Formula

$$\hat{\lambda} = \frac{\mathbf{y}^T \mathbf{P}^2 \mathbf{y}\, trace\left(\mathbf{A}^{-1} - \hat{\lambda}\mathbf{A}^{-2}\right)}{\mathbf{w}^T \mathbf{A}^{-1} \mathbf{w}\, trace(\mathbf{P})}$$

Local Ridge Regression

Re-Estimation Formula

$$\hat{\lambda} = \frac{\mathbf{y}^T \mathbf{P}^2 \mathbf{y} \, trace\left(\mathbf{A}^{-1} - \hat{\lambda}\mathbf{A}^{-2}\right)}{\mathbf{w}^T \mathbf{A}^{-1} \mathbf{w} \, trace(\mathbf{P})}$$

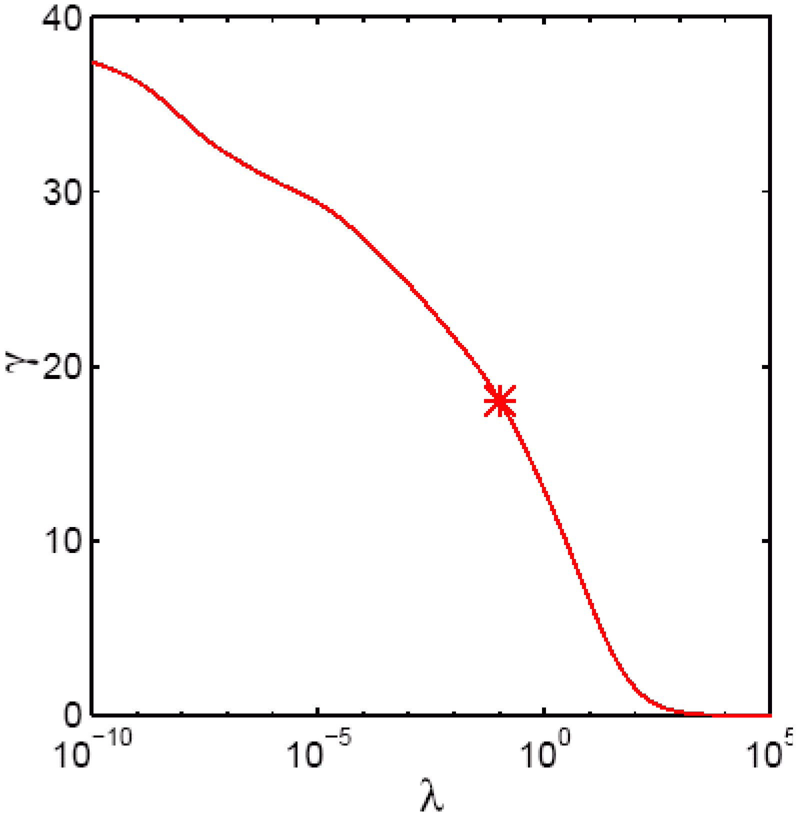
Example

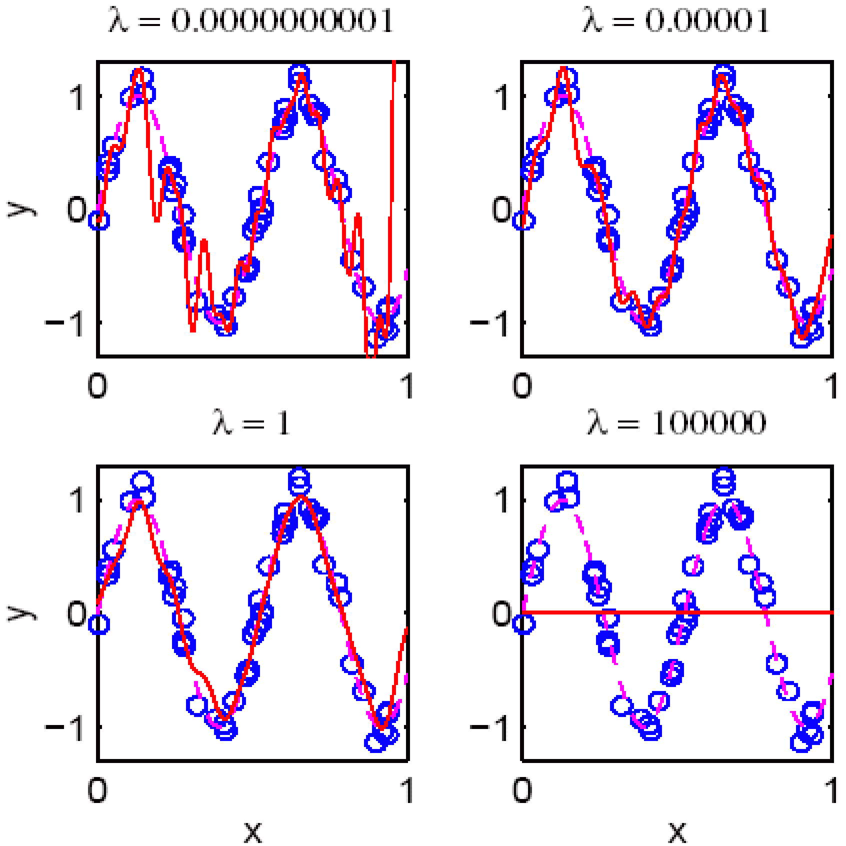
$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

$r = 0.05$ – Width of RBF

$$\gamma = p - trace(\mathbf{P})$$





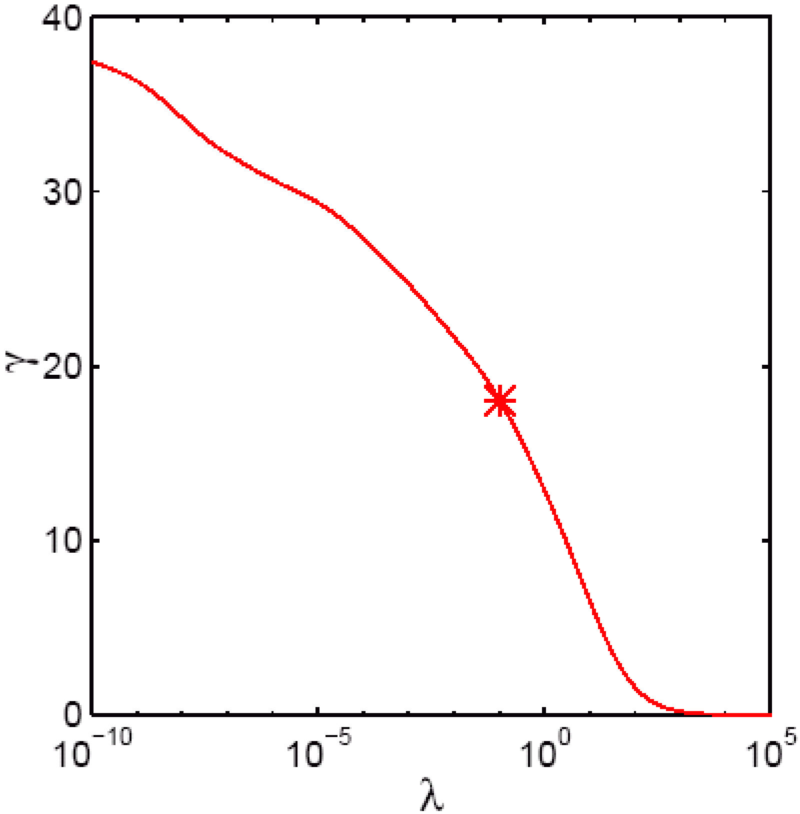
Example

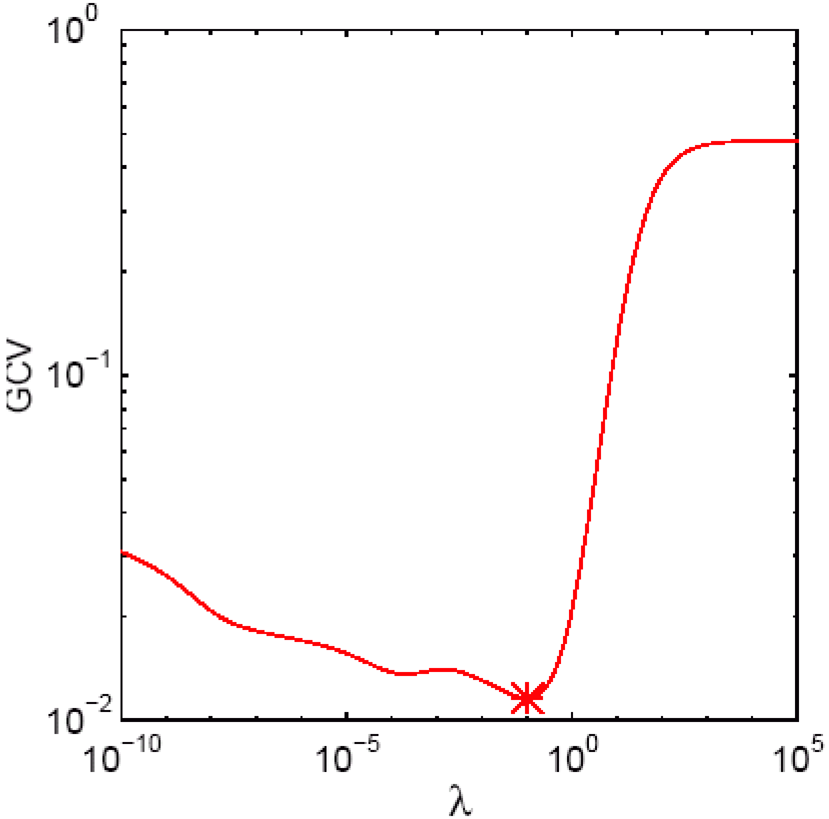
$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

$r = 0.05$ — Width of RBF

$$\gamma = p - trace(\mathbf{P})$$





Example

$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

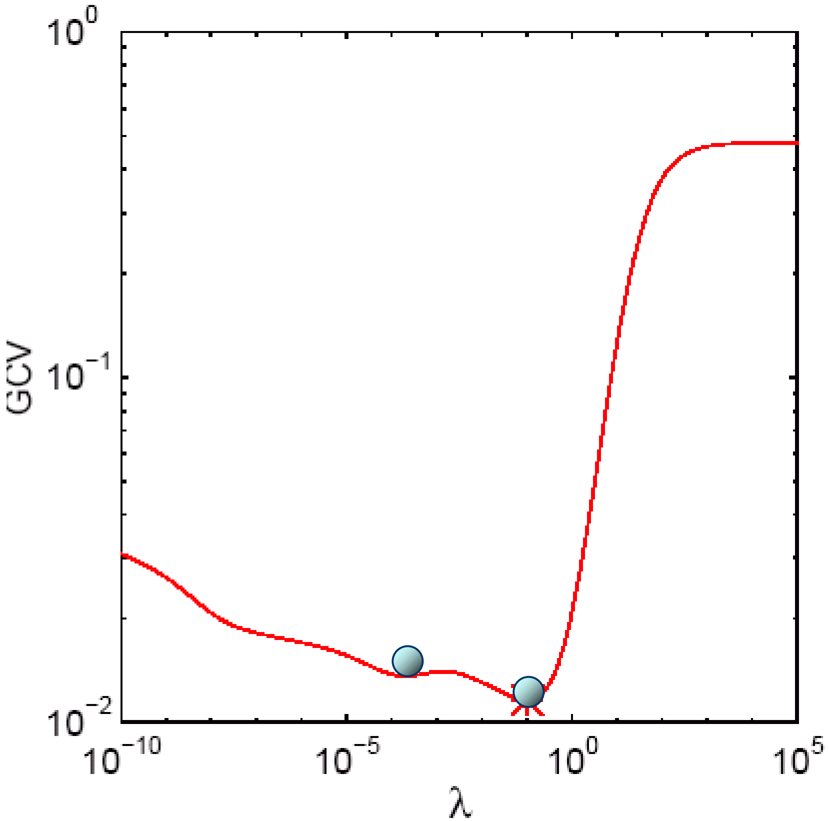
$$r = 0.05$$ — Width of RBF

$$\hat{\lambda} = \frac{\mathbf{y}^T \mathbf{P}^2 \mathbf{y}\, trace\left(\mathbf{A}^{-1} - \hat{\lambda}\mathbf{A}^{-2}\right)}{\mathbf{w}^T \mathbf{A}^{-1} \mathbf{w}\, trace(\mathbf{P})}$$

Using the about re-estimation formula, it will be stuck at the nearest local minimum.

That is, the solution depends on the initial setting.

There are two local-minima.



Example

$y = \sin(12x) + \varepsilon \quad \varepsilon \sim N(0, 0.1^2)$

$m = p = 50$

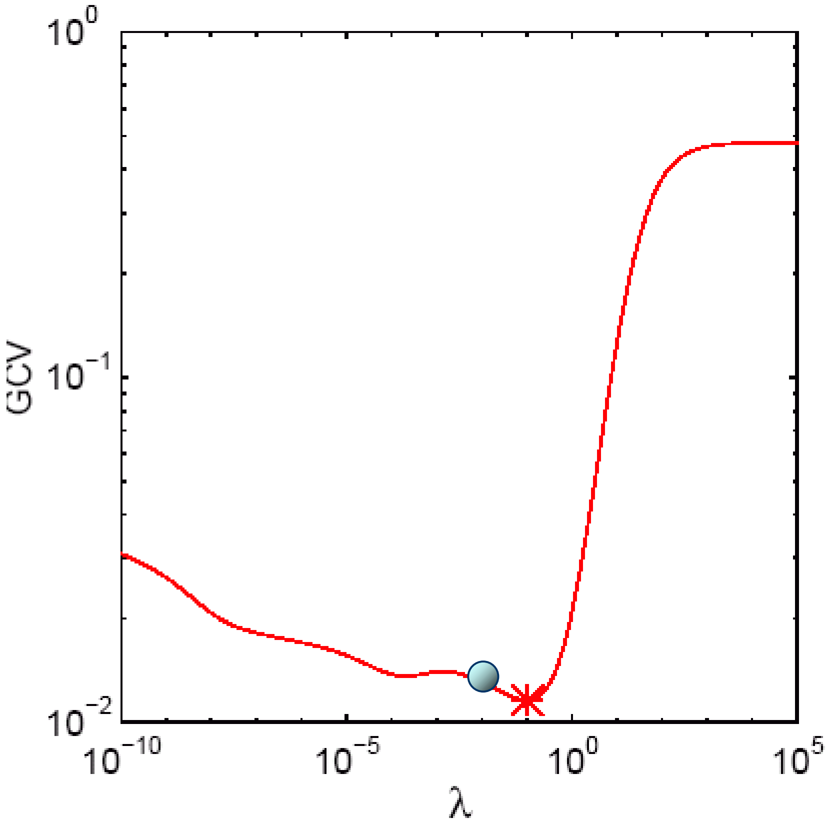
$r = 0.05$ — Width of RBF

$$\hat{\lambda} = \frac{\mathbf{y}^T \mathbf{P}^2 \mathbf{y}\, trace\left(\mathbf{A}^{-1} - \hat{\lambda}\mathbf{A}^{-2}\right)}{\mathbf{w}^T \mathbf{A}^{-1} \mathbf{w}\, trace(\mathbf{P})}$$

$$\hat{\lambda}(0) = 0.01$$

$$\lim_{t \to \infty} \hat{\lambda}(t) = 0.1$$

There are two local-minima.



Example

$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

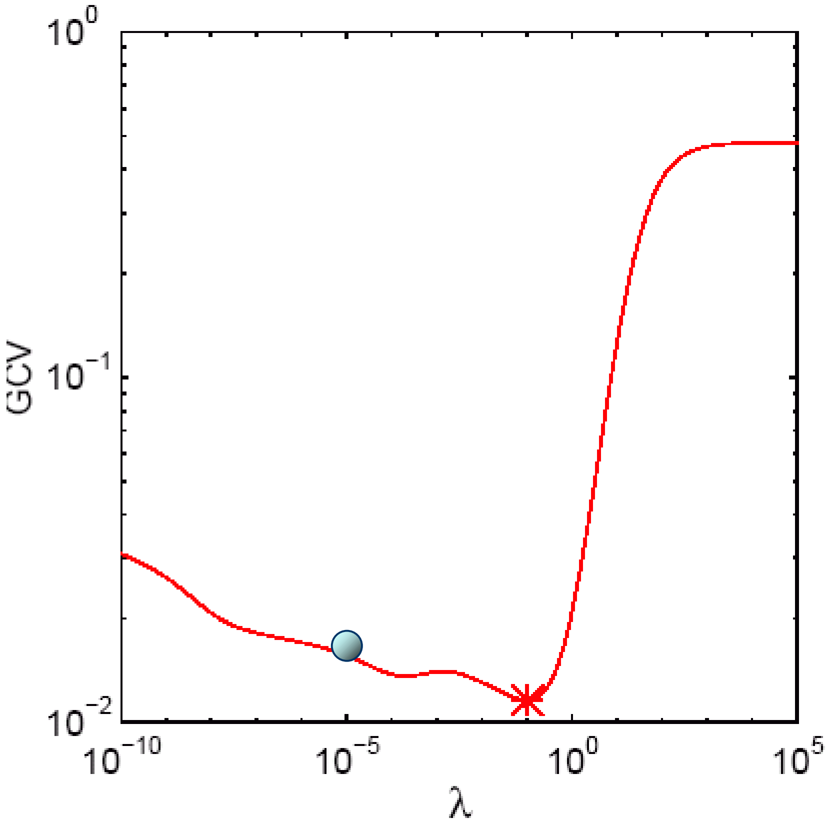
$r = 0.05$ — Width of RBF

$$\hat{\lambda} = \frac{\mathbf{y}^T \mathbf{P}^2 \mathbf{y}\, trace\left(\mathbf{A}^{-1} - \hat{\lambda}\mathbf{A}^{-2}\right)}{\mathbf{w}^T \mathbf{A}^{-1} \mathbf{w}\, trace(\mathbf{P})}$$

$$\hat{\lambda}(0) = 10^{-5}$$

$$\lim_{t \to \infty} \hat{\lambda}(t) = 2.1 \times 10^{-4}$$

There are two local-minima.



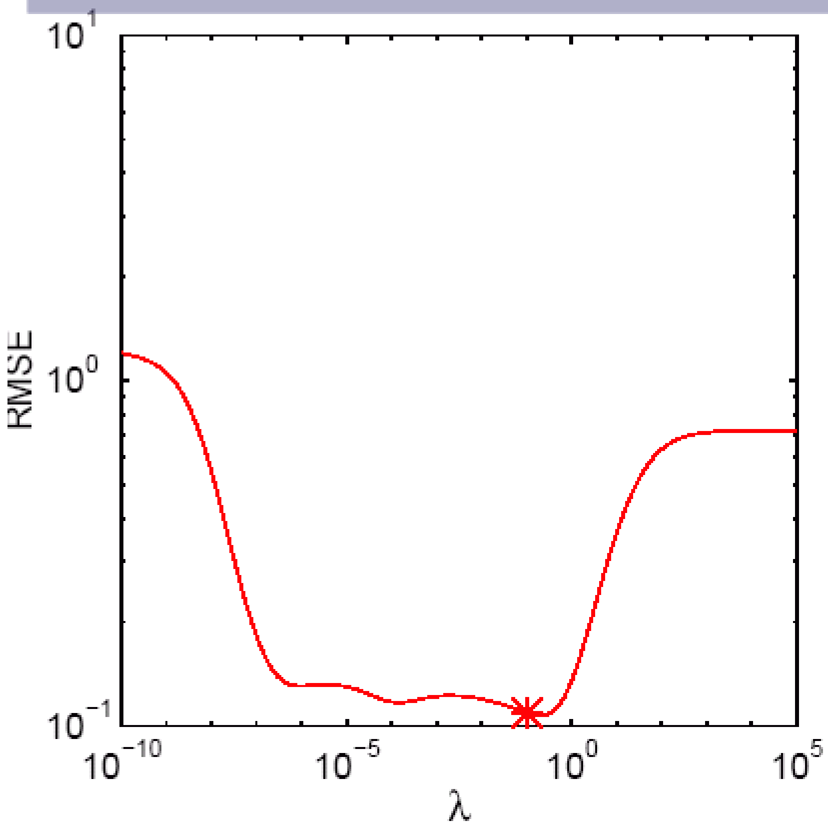
Example

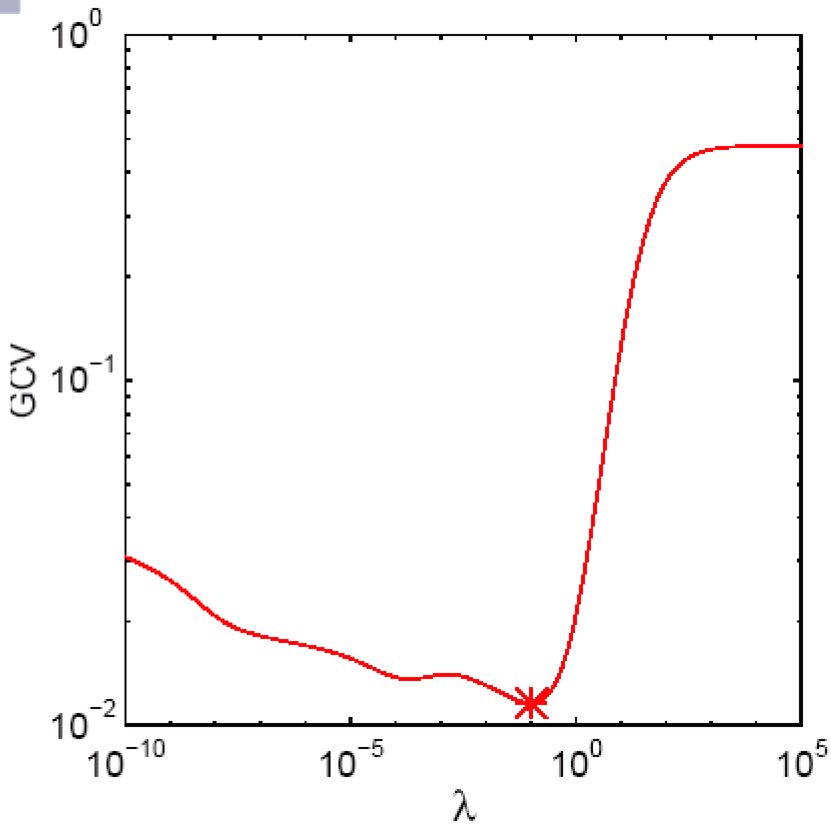
$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

$r = 0.05$ — Width of RBF

RMSE: Root Mean Squared Error
In real case, it is not available.





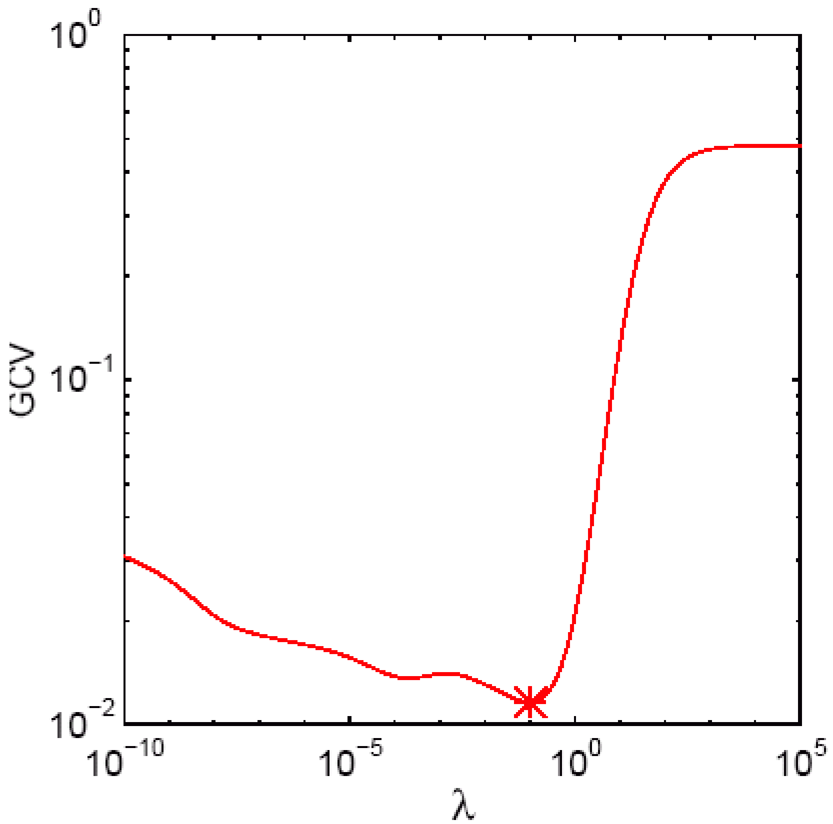
Example

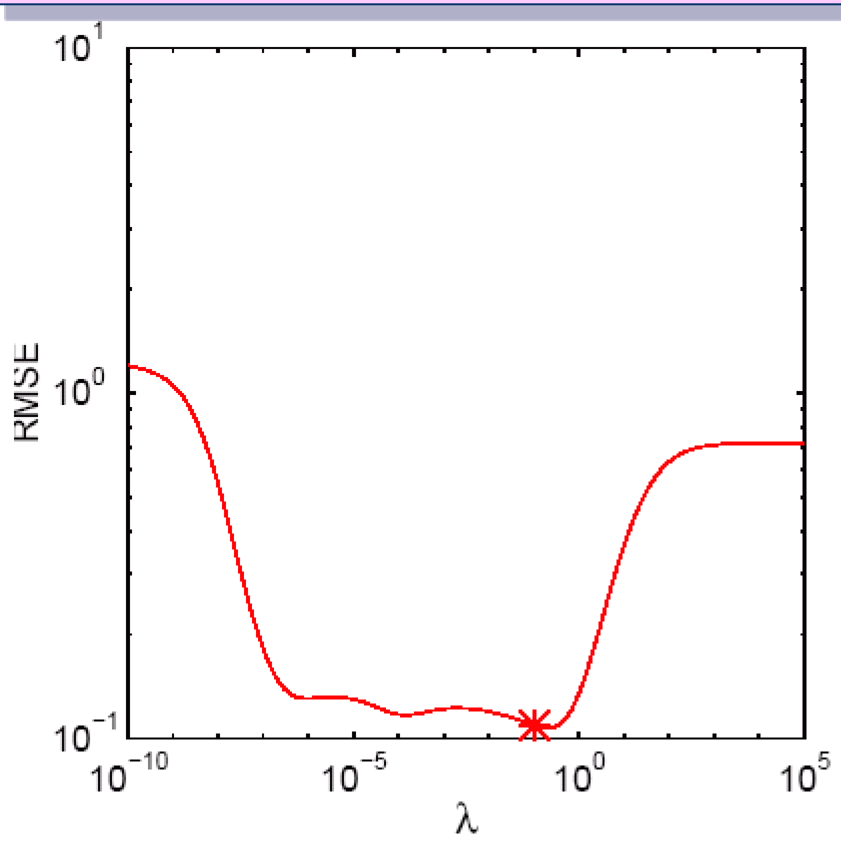
$$y = \sin(12x) + \varepsilon \qquad \varepsilon \sim N(0, 0.1^2)$$

$$m = p = 50$$

$r = 0.05$ — Width of RBF

RMSE: Root Mean Squared Error
In real case, it is not available.





Local Ridge Regression

Standard Ridge Regression

$$\min \quad C = \sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2 + \sum_{i=1}^{m} \lambda w_i^2$$

Local Ridge Regression

$$\min \quad C = \sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2 + \sum_{i=1}^{m} \lambda_i w_i^2$$

Local Ridge Regression

Standard Ridge Re...

$\lambda_j \rightarrow \infty$ implies that $\phi_j(\cdot)$ can be removed.

Local Ridge Regression

$$\min \quad C = \sum_{k=1}^{p}\left[y^{(k)} - \sum_{i=1}^{m} w_i \phi_i\left(\mathbf{x}^{(k)}\right)\right]^2 + \sum_{i=1}^{m} \lambda_i w_i^2$$

The Solutions

$$\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi}$$ Linear Regression

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I}_m$$ Standard Ridge Regression

$$\mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{\Lambda}$$ Local Ridge Regression

$$\mathbf{P} = \mathbf{I}_p - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T$$

Used to compute model selection criteria

Optimizing the Regularization Parameters

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T \mathbf{P}^2 \hat{\mathbf{y}}}{\left[trace(\mathbf{P})\right]^2}$$

Incremental Operation

$\mathbf{P}$: The current projection Matrix.

$\mathbf{P}_j$: The projection Matrix obtained by removing $\phi_j(\cdot)$.

$$\mathbf{P} = \mathbf{P}_j - \frac{\mathbf{P}_j \boldsymbol{\varphi}_j \boldsymbol{\varphi}_j^T \mathbf{P}_j}{\lambda_j + \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j}$$

Optimizing the Regularization Parameters

Solve $\dfrac{\partial \hat{\sigma}^2_{GCV}}{\partial \lambda_j} = 0$

Subject to $\lambda_j \geq 0$

$$\hat{\sigma}^2_{GCV} = \frac{p\hat{\mathbf{y}}^T \mathbf{P}^2 \hat{\mathbf{y}}}{\left[trace(\mathbf{P})\right]^2}$$

$$\mathbf{P} = \mathbf{P}_j - \frac{\mathbf{P}_j \boldsymbol{\varphi}_j \boldsymbol{\varphi}_j^T \mathbf{P}_j}{\lambda_j + \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j}$$

Optimizing the Regularization Parameters

Solve $\dfrac{\partial \hat{\sigma}^2_{GCV}}{\partial \lambda_j} = 0$

Subject to $\lambda_j \ge 0$

$$a = \mathbf{y}^T \mathbf{P}_j^2 \mathbf{y}$$

$$b = \mathbf{y}^T \mathbf{P}_j^2 \boldsymbol{\varphi}_j \mathbf{y}^T \mathbf{P}_j \boldsymbol{\varphi}_j$$

$$c = \boldsymbol{\varphi}_j^T \mathbf{P}_j^2 \boldsymbol{\varphi}_j (\mathbf{y}^T \mathbf{P}_j \boldsymbol{\varphi}_j)^2$$

$$\alpha = trace(\mathbf{P}_j)$$

$$\beta = \boldsymbol{\varphi}_j^T \mathbf{P}_j^2 \boldsymbol{\varphi}_j$$

$$\hat{\lambda}_j = \begin{cases} \infty & \text{if } a\beta = \alpha b \\ \infty & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j > \frac{c\alpha - b\beta}{b\alpha - a\beta} \text{ and } a\beta > \alpha b \\ 0 & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j > \frac{c\alpha - b\beta}{b\alpha - a\beta} \text{ and } a\beta < \alpha b \\ \frac{c\alpha - b\beta}{b\alpha - a\beta} - \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j \le \frac{c\alpha - b\beta}{b\alpha - a\beta} \end{cases}$$

Optimizing the Regularization Parameters

Solve $\dfrac{\partial \hat{\sigma}^2_{GCV}}{\partial \lambda_j} = 0$

Subject to $\lambda_j \geq 0$

Remove $\phi_j(\cdot)$

$\alpha = \mathrm{trace}(\mathbf{P}_j)$

$\beta = \boldsymbol{\varphi}_j^T \mathbf{P}_j^2 \boldsymbol{\varphi}_j$

$$\hat{\lambda}_j = \begin{cases} \infty & \text{if } a\beta = \alpha b \\ \infty & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j > \frac{c\alpha - b\beta}{b\alpha - a\beta} \text{ and } a\beta > \alpha b \\ 0 & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j > \frac{c\alpha - b\beta}{b\alpha - a\beta} \text{ and } a\beta < \alpha b \\ \frac{c\alpha - b\beta}{b\alpha - a\beta} - \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j & \text{if } \boldsymbol{\varphi}_j^T \mathbf{P}_j \boldsymbol{\varphi}_j \leq \frac{c\alpha - b\beta}{b\alpha - a\beta} \end{cases}$$

The Algorithm

- Initialize λ_i's.
  - e.g., performing standard ridge regression.
- Repeat the following until GCV converges:
  - Randomly select j and compute $\hat{\lambda}_j$
  - Perform local ridge regression
  - If GCV reduce & $\hat{\lambda}_j = \infty$ remove $\phi_j(\cdot)$