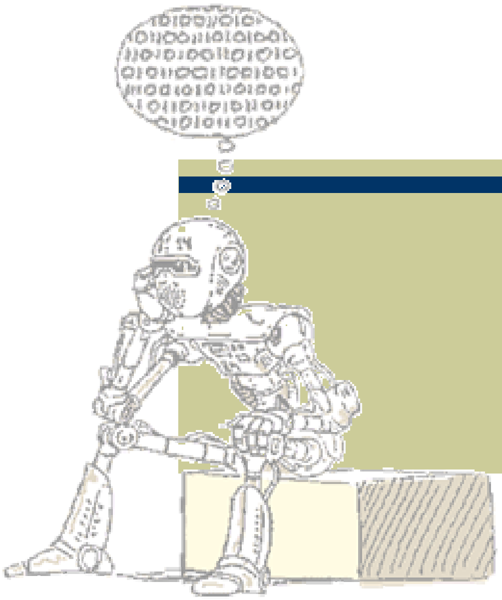


Lecture VII– Regression
(Nonlinear/Nonparametric Methods)

Contents:

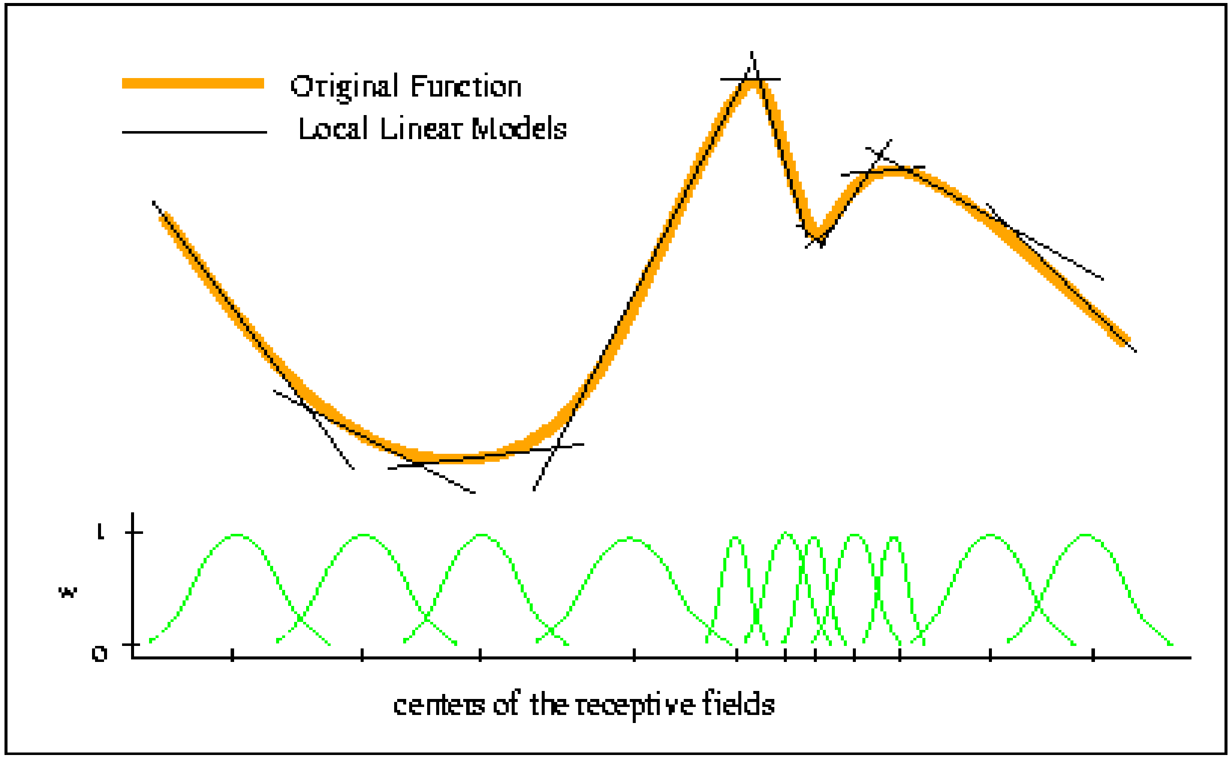
- Local Weighted and Lazy learning techniques
- Nonparametric methods

Nonparametric Methods

- Working Definition
  - The name “nonparametric” is to indicate that the data to be modeled stem from very large families of distributions which cannot be indexed by a finite dimensional parameter vector in a natural way.
- Remarks
  - this does **not** mean that nonparametric methods have no parameters!
  - nonparametric methods ***avoid making assumptions*** about the parametric form of the underlying distributions (except some smoothness properties).
  - nonparametric methods are often memory-based (but not necessarily)
  - sometimes called “lazy learning”
- Can be applied to
  - density estimation
  - classification
  - regression

Locally Weighted Regression (LWR)

- Fit ***locally*** lower order polynomials, e.g., first order polynomials



Approximate non-linear functions with a mixture of *k* piecewise linear models

LWR: Formalization

- Minimize Weighted Squared Error

$$J = \sum_{n=1}^{N} w_n \left(\mathbf{t}^n - \mathbf{y}^n\right)^T \left(\mathbf{t}^n - \mathbf{y}^n\right), \text{ where } \quad \mathbf{y}^n = \mathbf{x}^{nT} \boldsymbol{\beta}$$

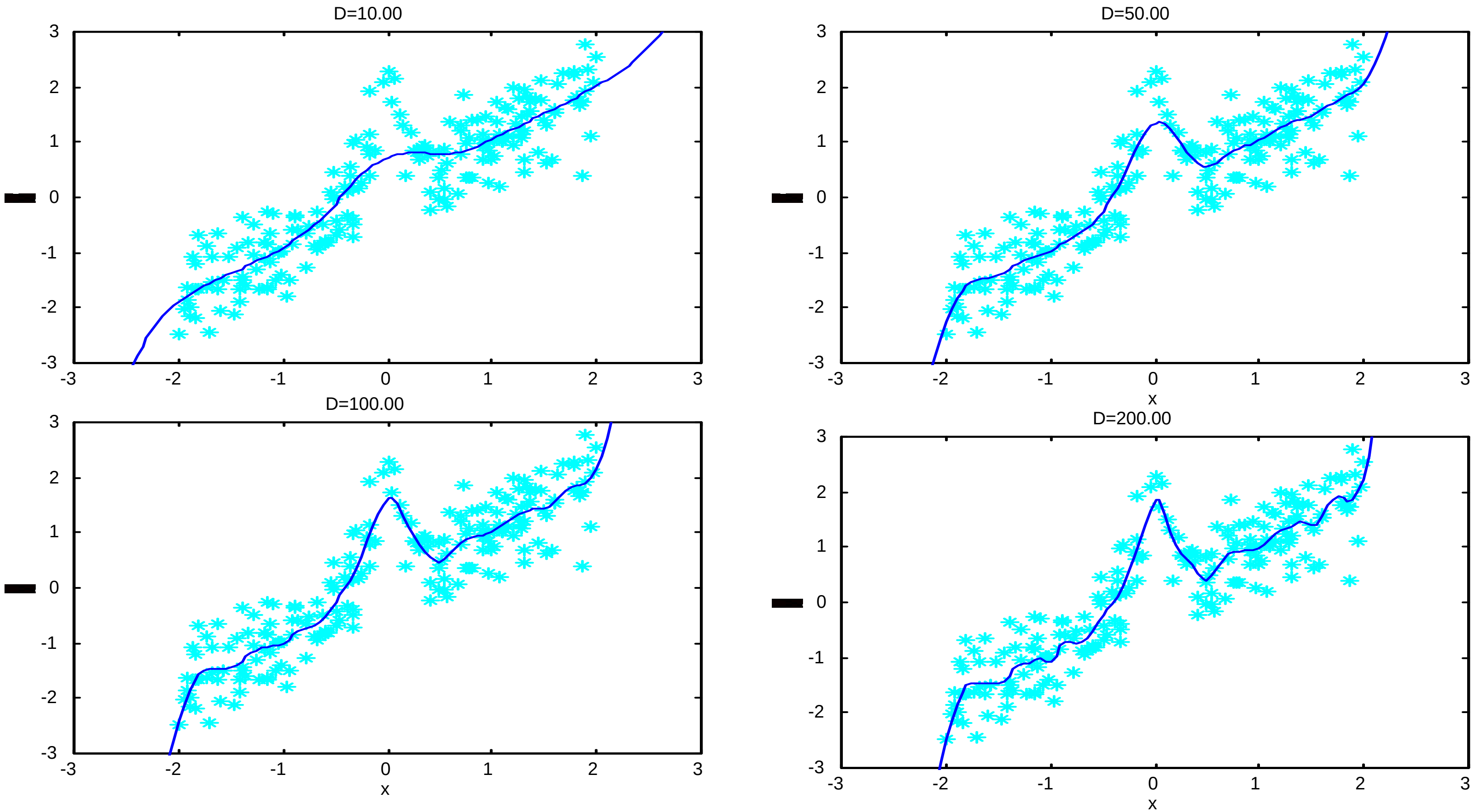
- **Solution:** Weighted Least Squares

$$\beta = \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y} \qquad \text{where } \mathbf{W} = \begin{bmatrix} w^1 & 0 & 0 & 0 \\ 0 & w^2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & w^n \end{bmatrix}$$

- Weight can be calculated from any weighting kernel, e.g., a Gaussian:

$$w = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \mathbf{D} (\mathbf{x} - \mathbf{c})\right)$$

LWR: Examples (cont'd)



The fits exhibit the bias-variance tradeoff effect with respect to parameter D.

How to Optimize the Distance Metric?

Two possible options :

- **Global Optimization**
  - find the most suitable **D** for the entire data set
- **Local Optimization**
  - find the most suitable **D** as a function of the kernel location **c**

Global Distance Metric Optimization

- **Leave-one-out Cross Validation**
  - compute the prediction of every data point in the training set by:
    - centering the kernel at every data point
    - but excluding the data point at the center from the training set

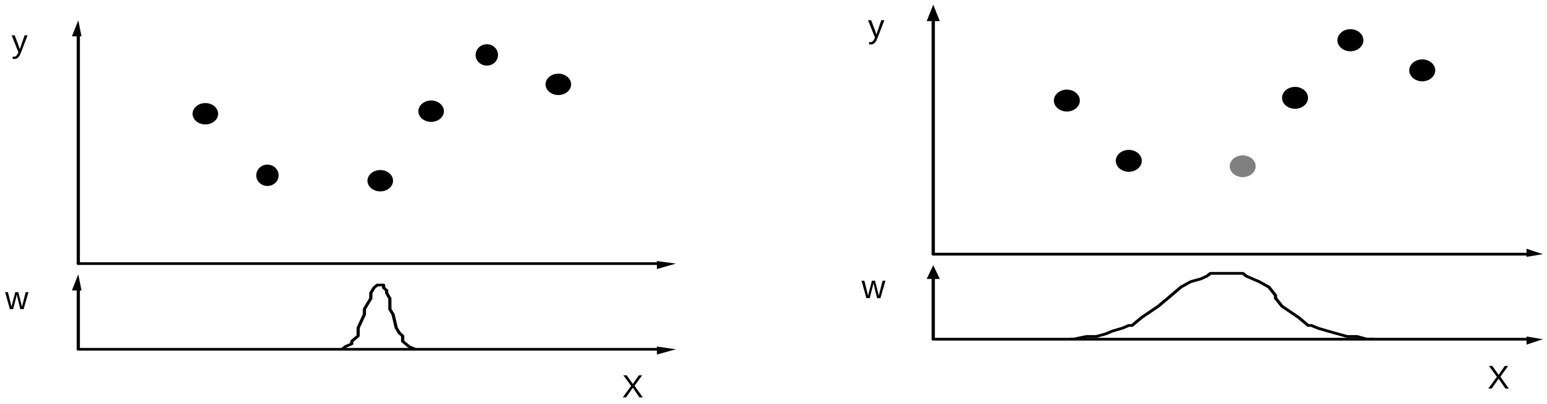
$$J_c = \sum_{n=1}^{N} \left(\mathbf{t}^n - \mathbf{y}_{-n}^n\right)^T \left(\mathbf{t}^n - \mathbf{y}_{-n}^n\right)$$

  - find distance metric that minimizes the cross validation error
    - depending on how many parameters are allowed in the distance metric, this is a multidimensional optimization problem

NOTE: Leave-one-out Cross Validation is very cheap in nonparametric methods (in comparison to most parametric methods) since there is no iterative training of the learning system

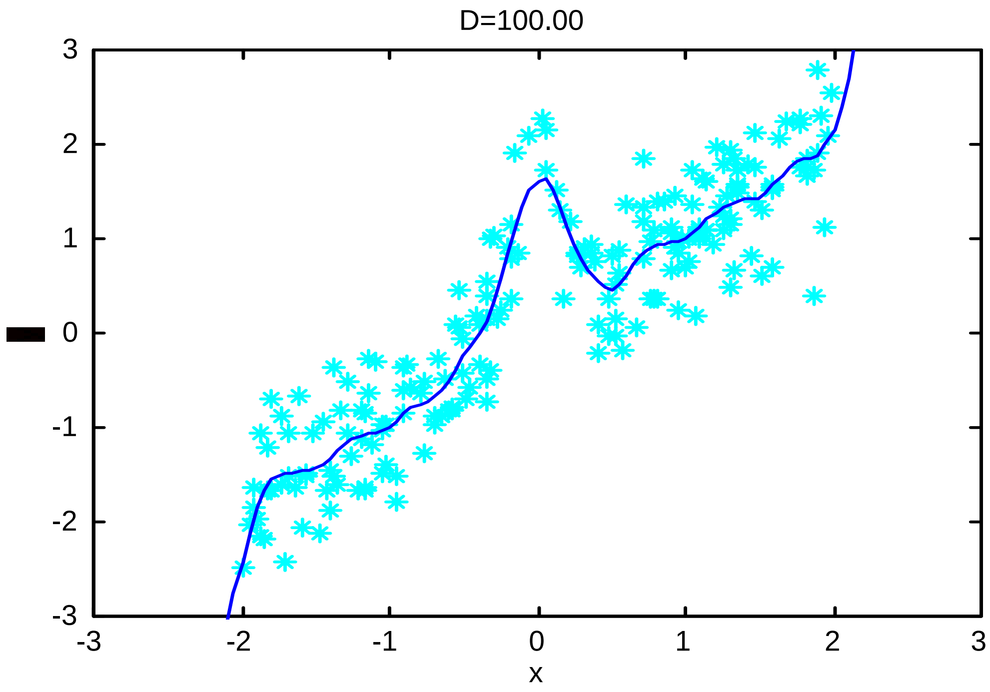
Why Cross Validation ?

*** Avoids degenerate Solutions for **D**

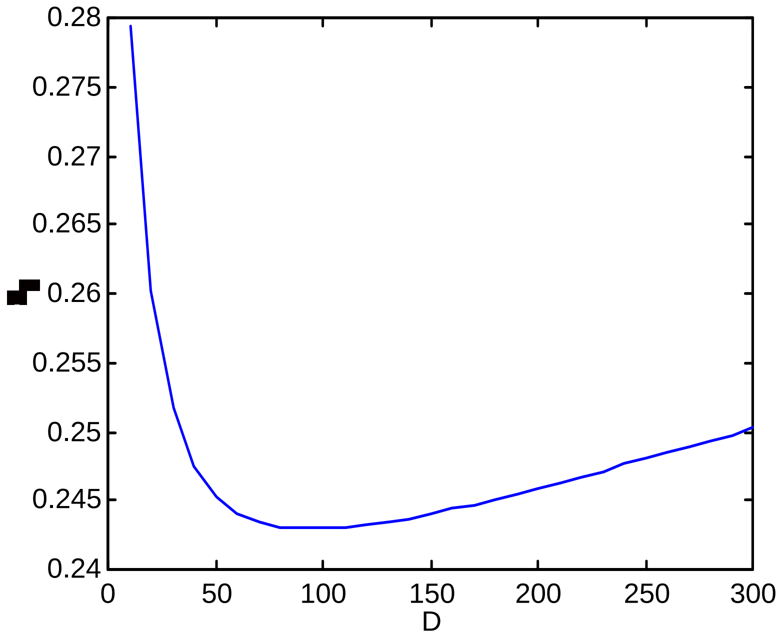


Without Cross Validation, the kernel could just shrink to zero and focus on one data point only.

Global Optimization of Distance Metric: Example



Resultant fit from Global Optimization



Locally Weighted Cost function

Local Distance Metric Optimization

- Why not optimize the distance metric as a function of the location of the kernel center?
  - Local Cross Validation Criterion

$$J_c = \sum_{n=1}^{N} w^n \left(\mathbf{t}^n - \mathbf{y}^n_{-n}\right)^T \left(\mathbf{t}^n - \mathbf{y}^n_{-n}\right)$$

- Something Exceptionally Cool: The local leave-one-out cross validation error can be computed analytically—WITHOUT an n-fold recalculation of the prediction for ***linear local models*** !

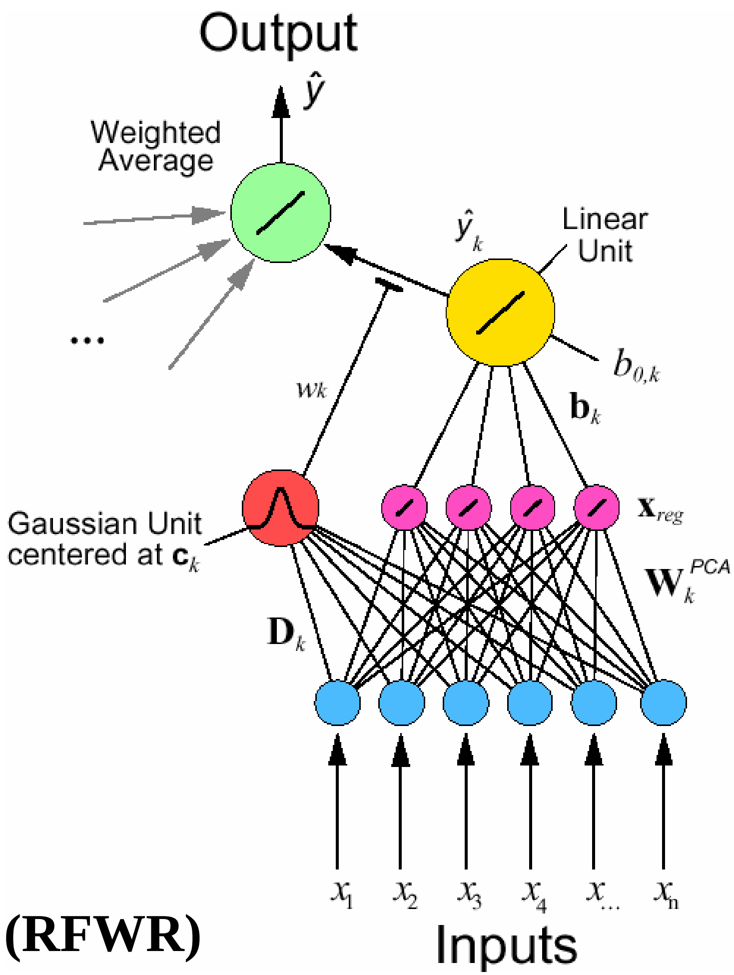
$$J_c = \sum_{n=1}^{N} w^n \left(\mathbf{t}^n - \mathbf{y}^n_{-n}\right)^T \left(\mathbf{t}^n - \mathbf{y}^n_{-n}\right) = \sum_{n=1}^{N} \frac{w^n \left(\mathbf{t}^n - \mathbf{y}^n\right)^T \left(\mathbf{t}^n - \mathbf{y}^n\right)}{\left(1 - w^n \mathbf{x}^{n^T} P \mathbf{x}^n\right)^2}$$

where $\beta = \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y} = \mathbf{P} \mathbf{X}^T \mathbf{W} \mathbf{Y}$

a.k.a.: Press Residual Error

A Nonparametric Regression Network

- **Ideas:**
  - Create new (Gaussian) kernels as needed (i.e., when no other kernel in the net is activated sufficiently (=> a constructive network)
  - ***update the linear model*** in each kernel by weighted recursive least squares
  - ***adjust the distance metric*** by gradient descent in local cross validation criteria
  - The weighted output of all kernels is the prediction

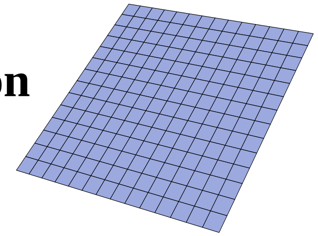


Receptive Field Weighted Regression (RFWR)

Nonparametric Regression Network (cont'd)

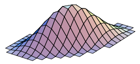
Elements of each Local Linear Model

Regression Slope



$$\mathbf{y} = \beta_x^T \mathbf{x} + \beta_0 = \beta^T \tilde{\mathbf{x}} \quad \text{where} \quad \tilde{\mathbf{x}} = \left[\mathbf{x}^T \; 1\right]^T$$

Receptive Field



$$w = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^T \mathbf{D}(\mathbf{x}-\mathbf{c})\right) \quad \text{where} \quad \mathbf{D} = \mathbf{M}^T\mathbf{M}$$

Penalized local cross validation error

$$J = \frac{1}{\sum_{i=1}^{p} w_i} \sum_{i=1}^{p} w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_{i,-i} \right\|^2 + \gamma \sum_{i=1,j=1}^{n} D_{ij}^2$$

Nonparametric Regression Network (cont'd)

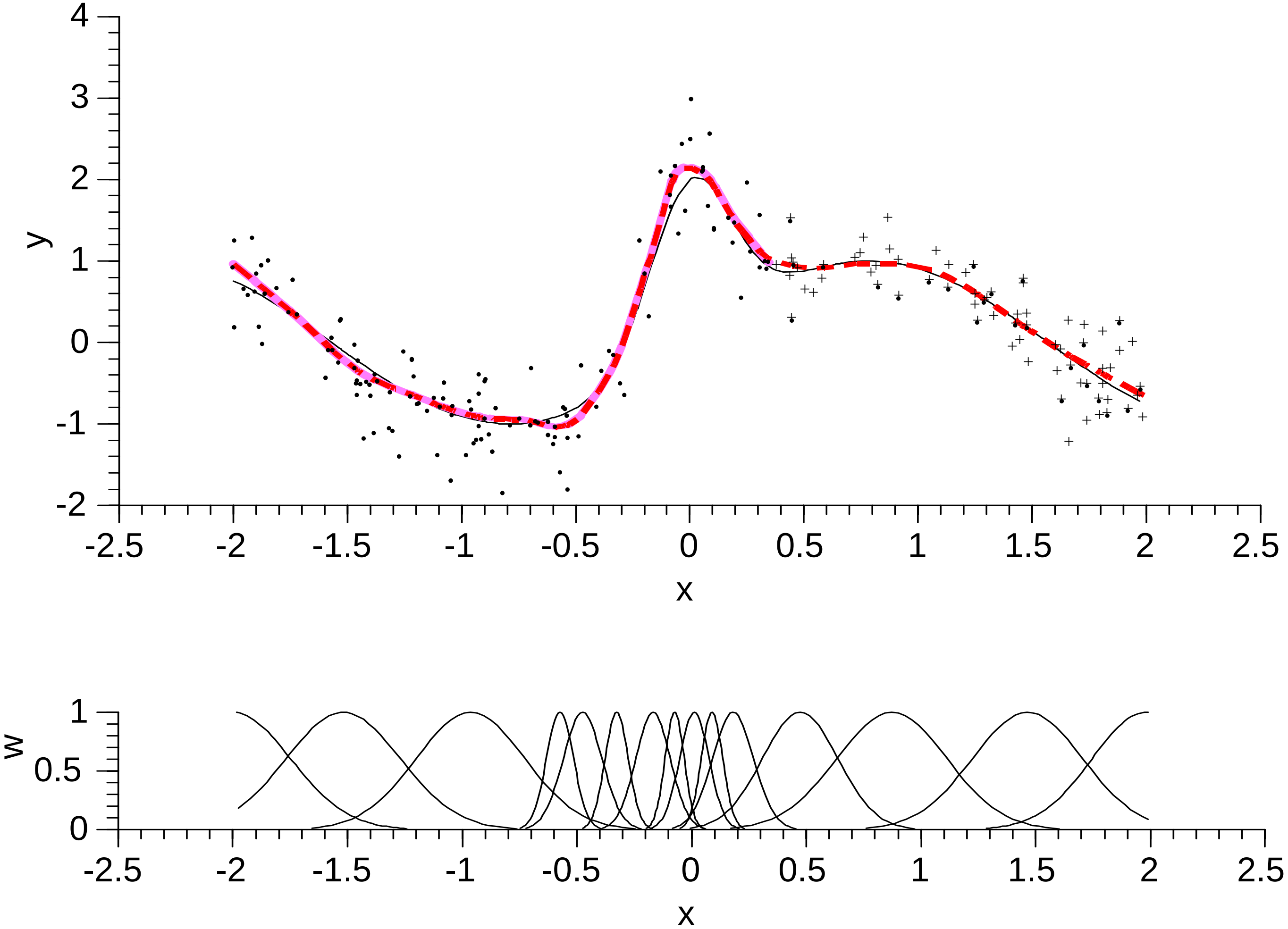
- **Update of the parameters:**
  - Slope of local model : $\beta^{n+1} = \beta^{n} + w\,\mathbf{P}^{n+1}\tilde{\mathbf{x}}\,\mathbf{e}_{cv}^{T}$

$$\text{where} \quad \mathbf{P}^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}^{n} - \frac{\mathbf{P}^{n}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{T}\mathbf{P}^{n}}{\frac{\lambda}{w} + \tilde{\mathbf{x}}^{T}\mathbf{P}^{n}\tilde{\mathbf{x}}}\right) \quad \text{and} \quad \mathbf{e}_{cv} = \left(\mathbf{y} - \tilde{\mathbf{x}}^{T}\beta^{n}\right)$$

  - Distance Metric Adaptation : $\mathbf{M}^{n+1} = \mathbf{M}^{n} - \alpha \frac{\partial J}{d\mathbf{M}}$

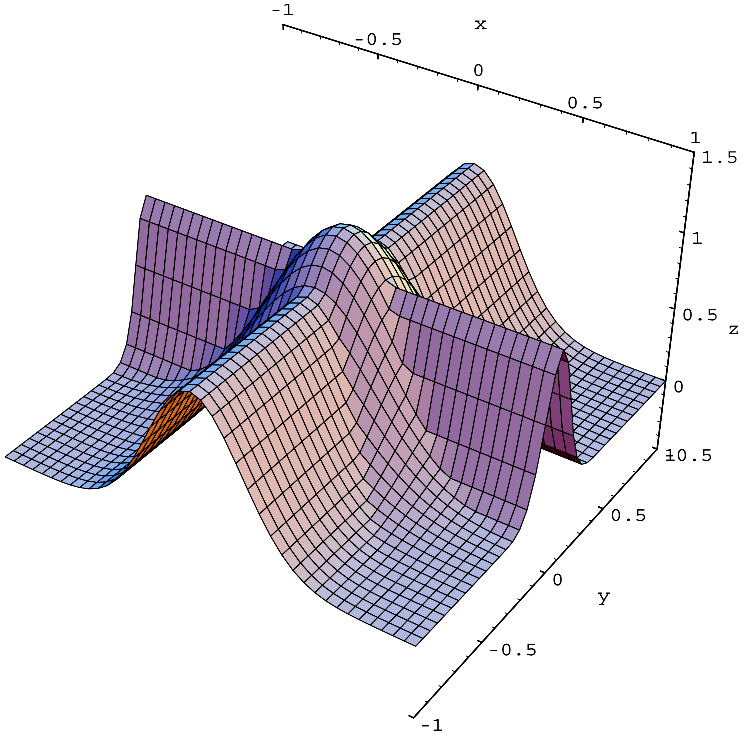
Another very cool thing: For linear systems, leave-one-out cross validation can be approximated INCREMENTALLY!
Thus, no data has to be kept in memory!

Example of learning

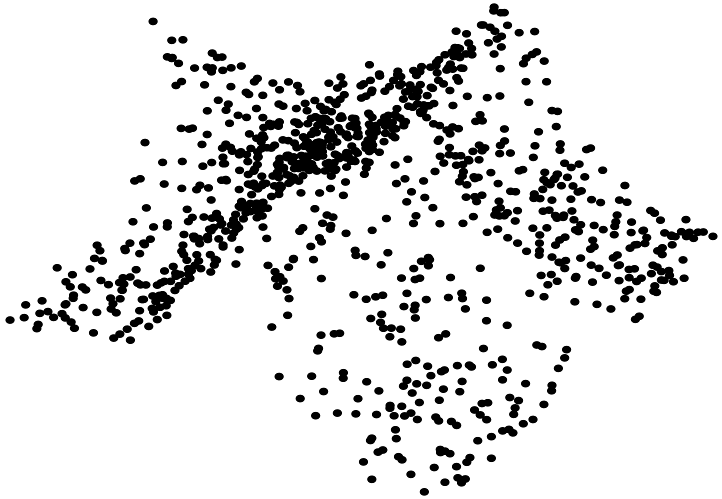


A 2D Example

Actual Data Generating function



Sampled Data with noise

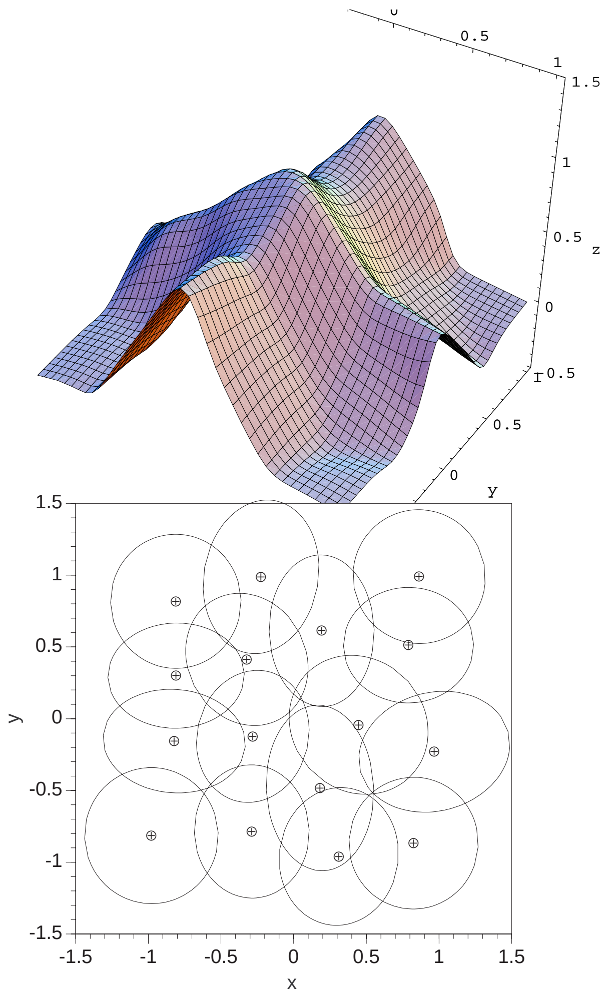


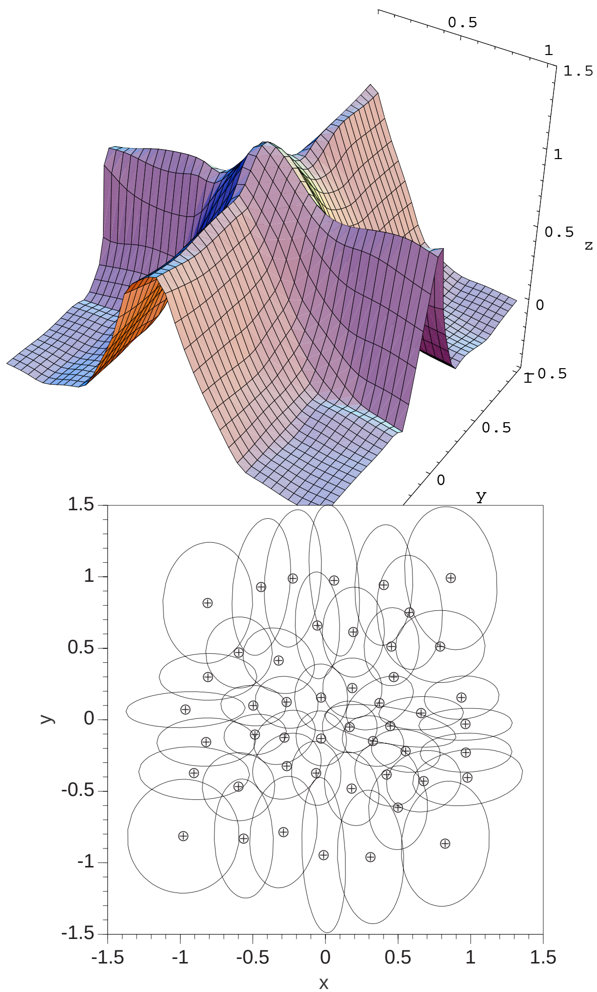
Why is this a tough function to learn ??

A 2D Example (cont'd)

Results after one Epoch of Training

Results after 50 Epochs of Training





Nonparametric Regression Network (Summary)

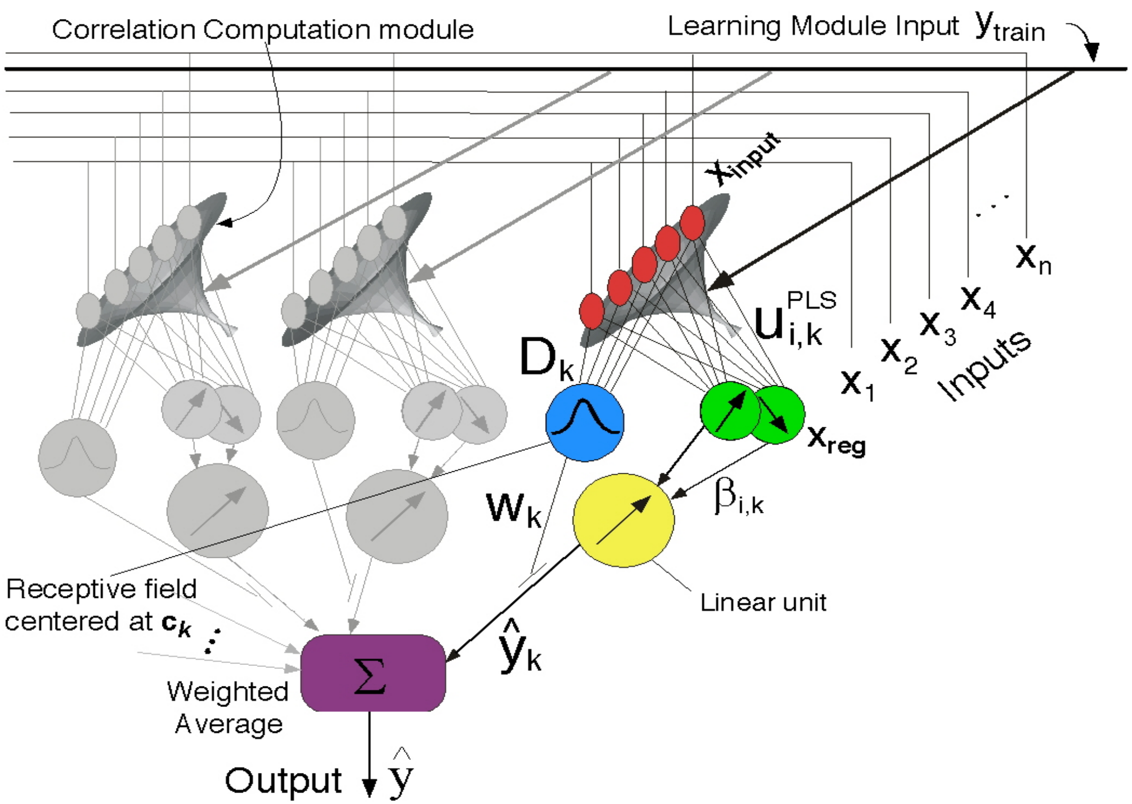
- **The LWR scheme we developed (RFWR)--**
  - can incrementally deal with the bias-variance dilemma
  - grows with the data (constructive)
  - learns very fast
  - is similar to a mixture of experts, but does not need a pre-specified number of experts (no competitive learning)
  - is similar to committee networks (by averaging the outputs over many independently trained networks)
- **… but still has problems with the curse of dimensionality, as all spatially localized networks**

Handling curse of Dimensionality

We developed a method to make Local Learning methods like RFWR scale

…the resultant system is called Locally Weighted Projection Regression (LWPR)

LWPR module



We will learn more about this and other dimensionality reduction techniques in the **next class**.