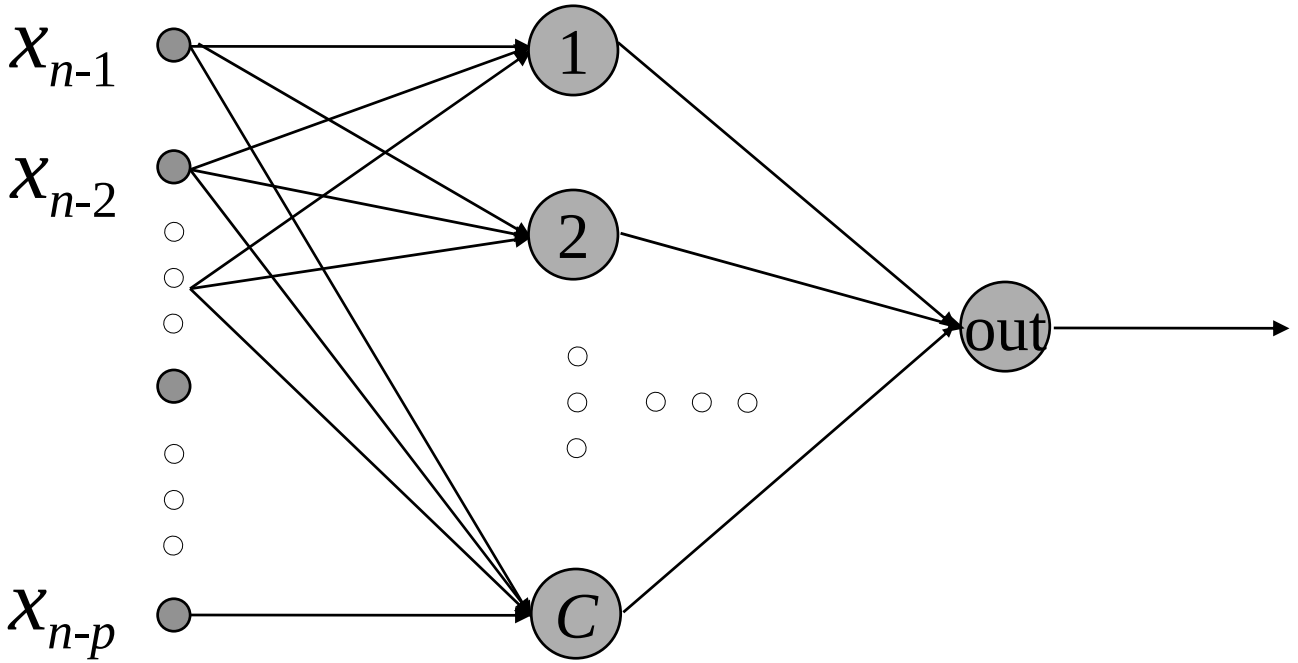
- Multi-layer perceptrons (MLP) constitute the most famous and most employed type of neural networks [1].
- At first, they were introduced in the context of classification, and it was soon recognized that they had the property of universal approximation capability.
- The introduction of the back-propagation algorithm made them flexible enough for many applications.

- Their basic structure for prediction is:



- The output of the ith neuron in the intermediate (hidden) layer is:

$$y_i = f\left(\theta_j + \sum_{j=1}^{p} w_{ij} x_{n-j}\right)$$

with $f(.)$ a sigmoidal function such as $f(u) = \tanh(au)$.

- The final output is a weighted sum of the outputs in the hidden layer.

- As for the polynomial predictor, one considers a NAR formulation:

$$x_n = g(x_{n-1},\ldots, x_{n-p}) + \varepsilon_n$$

and one estimates $g(.)$ in the least-square sense using the data at hand. That is, one uses the universal approximation capability of MLPs.

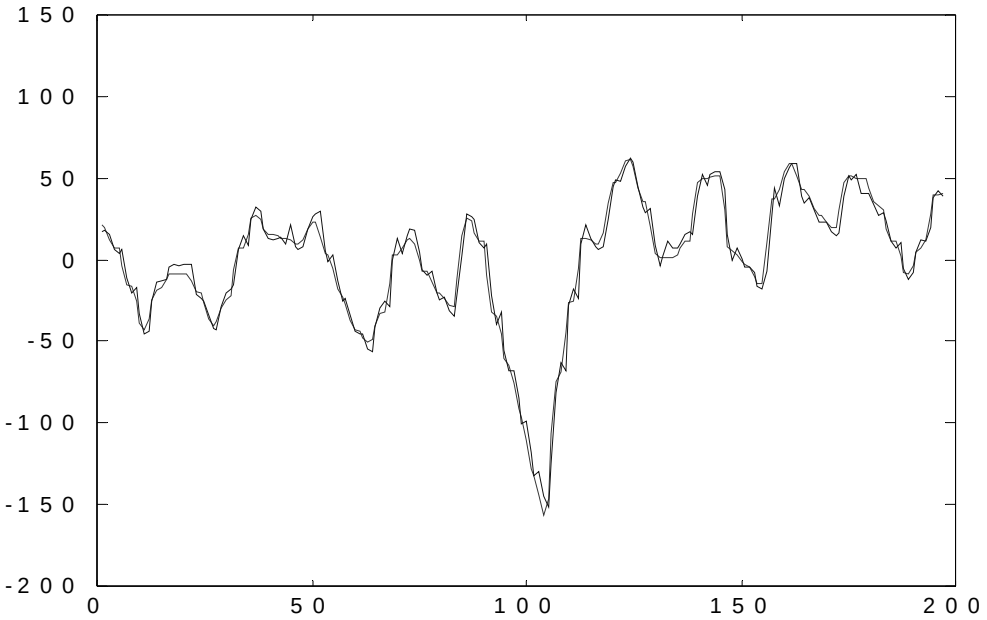
- Of course, the input vector to predict $x_n$ is

$\boldsymbol{x}_n = [x_{n-1}, x_{n-2},\ldots, x_{n-p}]$.

- It is possible to show that, when *p* increases, the size of the MLP increases more slowly than that of a polynomial predictor. But:
  - It is almost impossible to relate the values of the MLP parameters to the characteristic of the predicted signal (black box effect).
  - Convergence of the back-propagation algorithm is sometimes problematic.
  - Adaptive prediction is not very efficient.

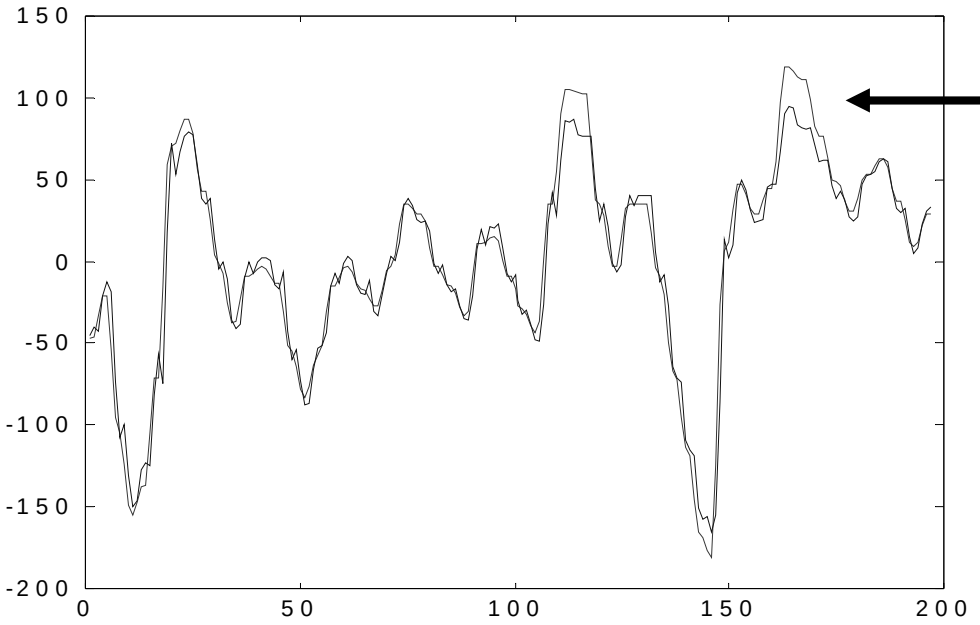
- Example: prediction of RR intervals ($p$ = 3, 5 neurons in the hidden layer). ESR: - 16 dB

Signal
Prediction



- Generalization: prediction on another part of the signal. ESR = -13 dB

Signal
Prediction




Failure to generalize

- Radial basis functions (RBF) constitute another type of neural networks, issued from interpolation theory.
- By some aspects, RBF combine the advantages of polynomial models and MLP:
  - Universal approximation property,
  - Estimation of parameters is simple,
  - There is a physical interpretation of these parameters.

- Let us suppose we have $N$ pairs $\{\boldsymbol{x}_i, y_i=g(\boldsymbol{x}_i)\}$, $i = 1, \ldots, N$, with $\boldsymbol{x}_i$ $p$-dimensional vectors, $y_i$ scalars, and $g(.)$ a continuous function.
- the *interpolation* of $g(.)$ from these $N$ pairs is *an ill-posed* one, because there is usually no information on $g(.)$ between these pairs.
- A sensible approach, called *regularization*, has been developed to tackle this type of problem.

- In this approach, one hypothesizes that $g(.)$ should not oscillate erraticaly , and should present some degree of smoothness. This translates into defining an interpolating function $G(.)$ minimizing a composite criterion:

$$\mathsf{C}(G) = \sum_{i=1}^{N} \left[ y_i - G(\mathbf{x}_i) \right]^2 + \rho \left\| \mathsf{P}G \right\|^2$$

with $\mathsf{P}$ a differential operator.

- Parameter $\rho$ balances the relative influence of the two terms. The first one deals with the fidelity with respect to the data, the second with interpolation smoothness.
- It is possible to show that, if P is an infinite sum of differential operators of increasing degree, and rotation and translation invariant, then $G(.)$ is:

$$G(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi\left(\left\|\mathbf{x} - \mathbf{x}_i\right\|\right) \quad \text{with} \quad \phi(s) = \exp\left(-\frac{s^2}{2\beta^2}\right)$$

- Parameter $\beta$ must be specified with respect to the regularization parameter $\rho$. If there is a true interpolation then one must have:

$$G(\boldsymbol{x}_i) = y_i, \;\; i = 1, \ldots, N$$

and one must solve the following linear system:

$$\begin{bmatrix} \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) & \cdots & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_N\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\boldsymbol{x}_N - \boldsymbol{x}_1\|) & \cdots & \phi(\|\boldsymbol{x}_N - \boldsymbol{x}_N\|) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

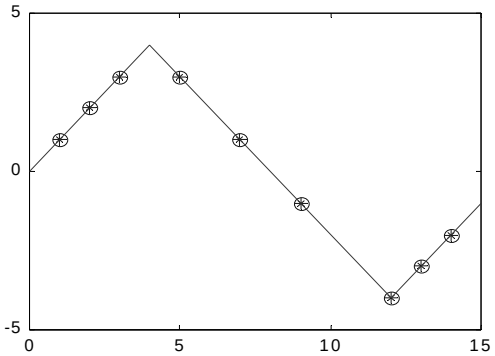
Thus a matrix equation:

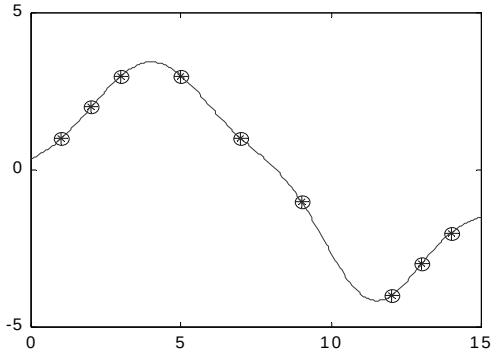
$$\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$$

- It is possible to show that if all vectors $\{\boldsymbol{x}_i\}$ are different, matrix $\mathbf{A}$ is always invertible, and it will always be possible to determine the coefficients $\{w_i\}$ of $\boldsymbol{w}$.
- The value of $\beta$ defines the behavior of $G(.)$ between the $\{\boldsymbol{x}_i\}$. The larger it is, the smoother $G(.)$ is.

- Example

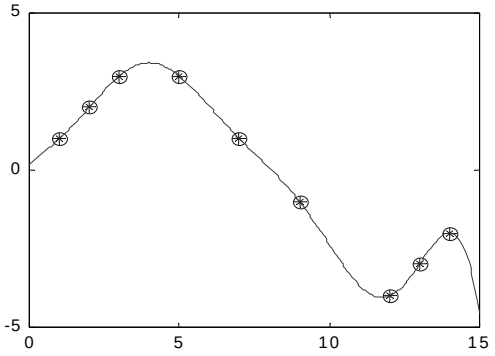
True function (not known).

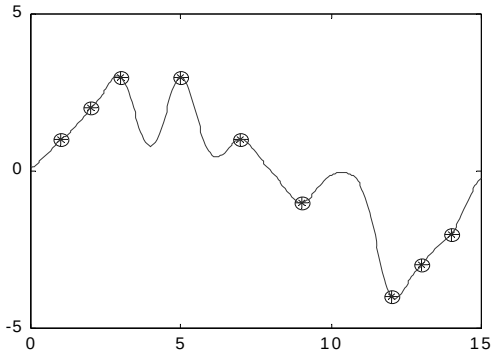



$\beta = 2$

$\beta = 5$




$\beta = 0.5$

- This approach of interpolation presents some limitations:
  - The fact that the interpolating function must include all pairs $\{\boldsymbol{x}_i, y_i=g(\boldsymbol{x}_i)\}$ makes the approach highly sensitive to noise (erroneous values) and thus prone to overfitting.
  - When the number of pairs increases $G(.)$ becomes soon complex.

- The idea that has been proposed to create a new type of neural networks is quite simple. One considers a slightly different formulation:

$$G(\boldsymbol{x}) = \sum_{j=1}^{M} w_j \phi\left(\left\| \boldsymbol{x} - \boldsymbol{c}_j \right\|\right) \quad \text{avec } \phi(s) = \exp\left(-\frac{s^2}{2\beta^2}\right)$$

with $M < N$, and vectors $\{\boldsymbol{c}_j\}$ called *centers*, which are not constrained to be a subset of the $\{\boldsymbol{x}_i\}$. Function $G(.)$ will now *approximate* $g(.)$, typically in the least-square sense.

- To sum up, the RBF network approximates locally $g(.)$ in the neighborhood of each center, and merges these local approximations to create a global one.
- The nice feature of RBF networks is that, once the centers and β have been defined, *determination of coefficients* $\{w_j\}$ *is simple, since* $G(.)$ *depends linearly on them*.

- If least-square estimation is used, one must solve:

$$\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{c}_1\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{c}_M\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{c}_1\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{c}_M\|) \\ \vdots & & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{c}_1\|) & \cdots & \phi(\|\mathbf{x}_N - \mathbf{c}_M\|) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}$$

that is $\mathbf{A}\boldsymbol{w} = \boldsymbol{y} + e$. $\mathbf{A}$ is now an $N \times M$ matrix.

- One may add a column of "1" to $\mathbf{A}$ if $\boldsymbol{y}$ is significantly non zero-mean.

- When minimizing $\|\mathbf{A}\boldsymbol{w} - \boldsymbol{y}\|$, one performs of course an orthogonal projection of vector $\boldsymbol{y}$ on the subspace generated by the columns of $\mathbf{A}$.
- Since matrix $\mathbf{A}$ may be ill-conditioned it is better to use a robust estimation scheme, typically SVD.
- But of course centers should not be too close to each other.

- Note first there is no really optimal way to select the centers. Also, we will come back later to the problem of determining the *number* of centers.
- One of the firts methods proposed consisted in selecting centers randomly (uniformly), in the hypercube defined by the $\{\boldsymbol{x}_i\}$. This is not very efficient, because the spatial distribution of the $\{\boldsymbol{x}_i\}$ is not taken into account.

- A second approach consists in selecting randomly the centers among the $\{\boldsymbol{x}_i\}$. This is already better, since statistically those centers are represntative of the spatial distribution of the $\{\boldsymbol{x}_i\}$. That is, there are more centers in the regions of space where the $\{\boldsymbol{x}_i\}$ are clustered. However, the constraint remains that the centers are a subset of the $\{\boldsymbol{x}_i\}$.

- A solution yielding better performance is to select centers using *learning vector quantization* (LVQ), which is used in other contexts too to find representatives of vector sets. LVQ works as follows:

1. The initial centers are chosen $\{\boldsymbol{c}_j\}$ at random in the hypercube defined by the $\{\boldsymbol{x}_i\}$.

2. Vectors $\{\boldsymbol{x}_i\}$ are presented, generally for several epochs. At iteration $k$, the following correction si applied to center $\boldsymbol{c}_j$:

$$\mathbf{c}_j(k+1) = \mathbf{c}_j(k) + \alpha(k)\left[\mathbf{x}(k) - \mathbf{c}_j(k)\right]$$

$$\text{if} \quad \left\|\mathbf{x}(k) - \mathbf{c}_j(k)\right\| < \left\|\mathbf{x}(k) - \mathbf{c}_m(k)\right\|, \ m \neq j$$

with $\boldsymbol{x}(k)$ the vector presented at iteration $k$. the net effect is to draw $\boldsymbol{c}_j$ closer to $\boldsymbol{x}(k)$, and thus possibly to a cluster.

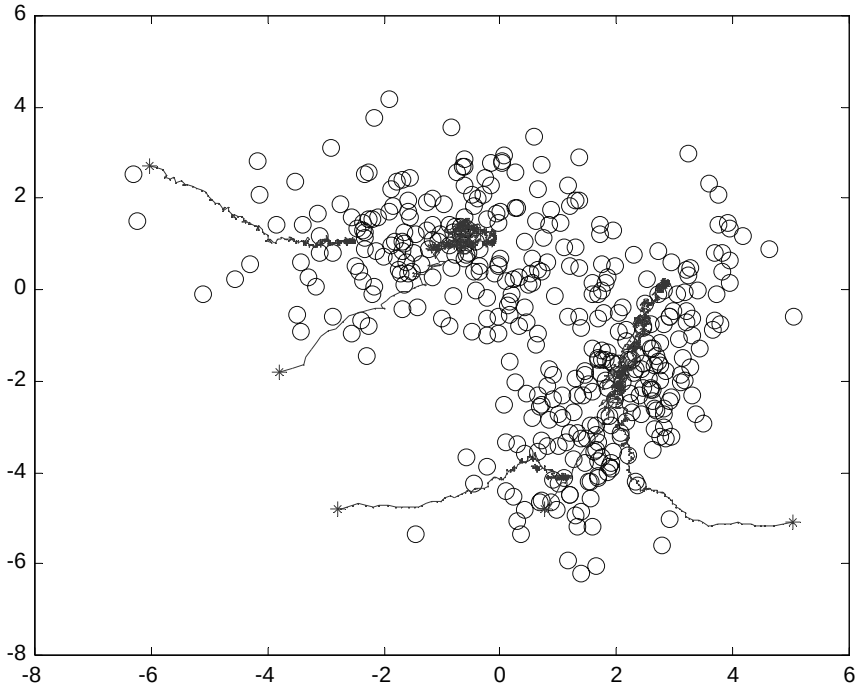
Parameter $\alpha(k)$ must decrease with $k$, generally using:

$$\alpha(k) = \alpha_0 \left( 1 - \frac{k}{K} \right)$$

$K$ total number of iterations, and $\alpha_0$ a constant (typically $\alpha_0 = 0.05$).

- It is to be noted that LVQ is sensitive to the initial center selection.

- Example: LVQ for $M = 5$ centers, vectors $\{\boldsymbol{x}_i\}$ drawn from two bi-dimensional Gaussian pdf.



- Another scheme, less sensitive to intial center choice, is *Lloyd-Max vector quantization algorithm*.
- Centers $\{\boldsymbol{c}_j\}$ (sometimes called in this context *centroids*) are chosen such that they minimize a distortion criterion:

$$D = \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in V_j} \| \mathbf{x}_i - \mathbf{c}_j \|^2$$

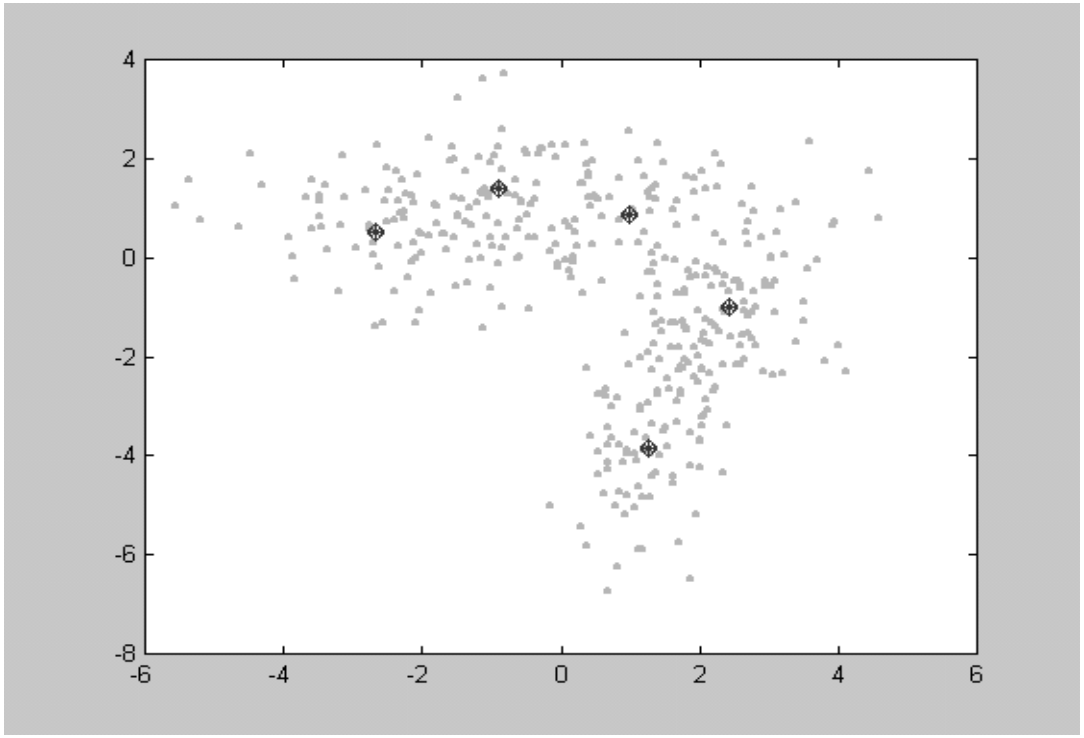
- Subsets $V_j$ contain vectors $\boldsymbol{x}_j$ such that $\boldsymbol{c}_j$ is the center closest to them. So to speak, $\boldsymbol{c}_j$ must be a good representative of $V_j$. Lloyd-Max algorithm works as follows

1. Initial random selection of $\{\boldsymbol{c}_j\}$ among the $\{\boldsymbol{x}_i\}$
2. Update:

$$\boldsymbol{c}_j \rightarrow \boldsymbol{c}_j = \text{mean}(\boldsymbol{x}_i \in V_j)$$

3. If distortion $D$ stabilizes, stop, otherwise back to 2.

- Example: Lloyd-Max for $M = 5$ centers, vectors $\{\boldsymbol{x}_i\}$ drawn from two bi-dimensional Gaussian pdf.




- data
- centroids

- Once the centers $\{\boldsymbol{c}_j\}$ have been chosen, parameter $\beta$ can be determined.
- This is the same as for interpolation: too large a $\beta$ means too strong an interaction (overlap) between the local approximations. Too small a $\beta$ means a ″bumpy″ approximation.
- A good empirical rule is:

$$\beta = \frac{d}{\sqrt{M}} \quad \text{with} \quad d = \max \left\| \mathbf{c}_i - \mathbf{c}_j \right\|, \quad 1 \le i, j \le M$$

- Nothing imposes to use the same value for $\beta$ for each RBF.
- It is even intuitevly appealing that $\beta$ be smaller in regions with many centers, since the spatial influence of the RBFs corresponding to these centers should be smaller
- A good empirical rule for $\beta_j$ corresponding to center $\boldsymbol{c}_j$ is:

$$\beta_j = \frac{1}{N\sqrt{8}} \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - \boldsymbol{c}_j \right\|$$

- As for MLP, one considers a NAR formulation:

$$x_n = g(x_{n-1},\ldots, x_{n-p}) + \varepsilon_n$$

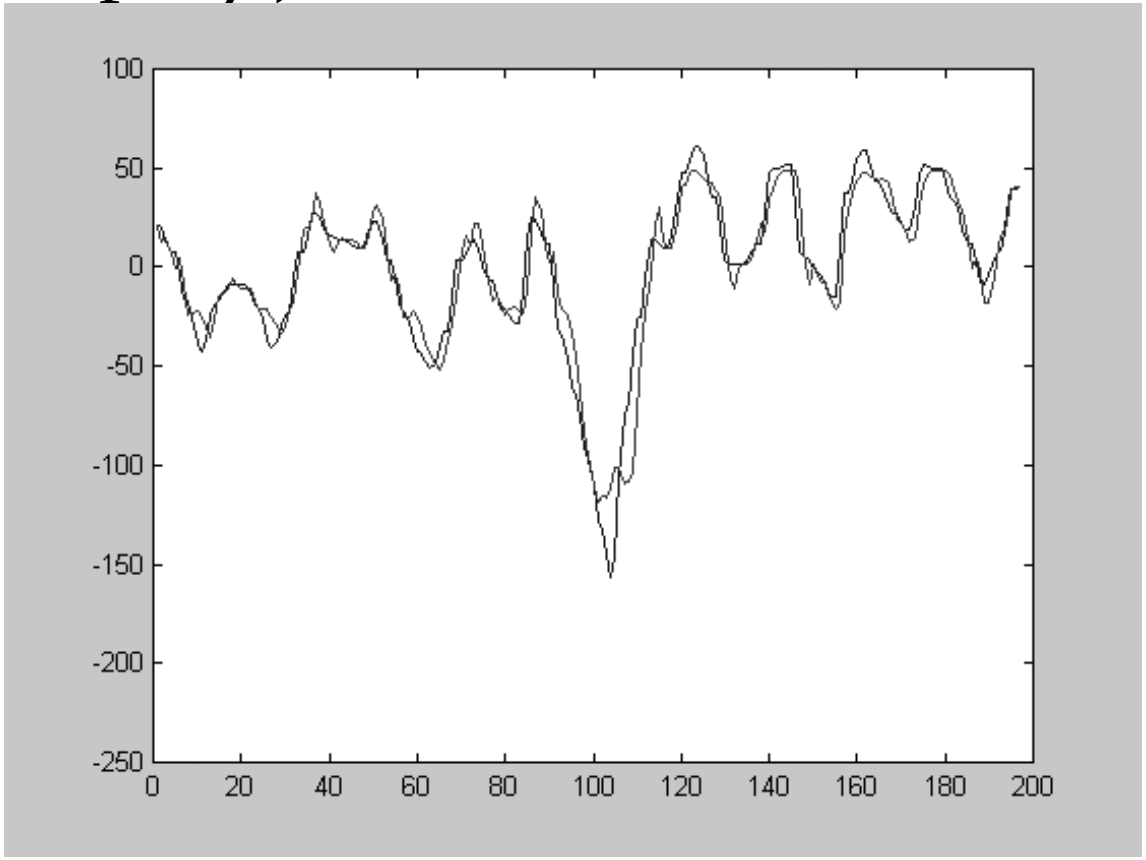
and one estimates $g(.)$ in the least-square sense using the data at hand. That is, one uses the universal approximation capability of RBF networks.

- Of course, the input vector to predict $x_n$ is

$\boldsymbol{x}_n = [x_{n-1}, x_{n-2},\ldots, x_{n-p}]$.

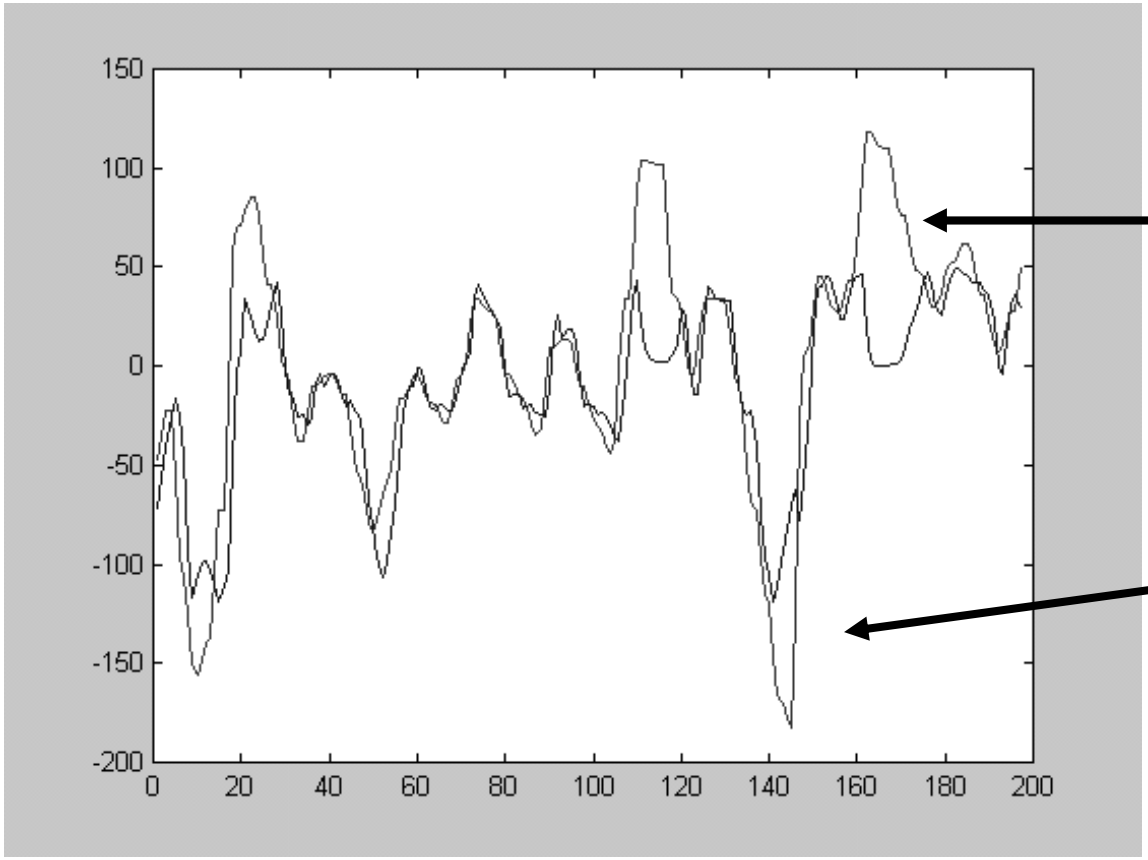
- Example: prediction of RR intervals ($p$ = 3, 20 RBF, multiple $\beta$). ESR: - 10 dB

Signal
Prediction



- Generalization: prediction on another part of the signal. ESR = -4 dB

Signal
Prediction




failure to generalize

- Center selection, which conditions the choice of parameter $\beta$ also, has an important role in RBF network performance.
- LVQ and Lloyd-Max algorithms are intuitively appealing. There are however other interesting schemes [3].
- Again, we note that solving $\mathbf{A}\boldsymbol{w} = \boldsymbol{y} + \boldsymbol{e}$ in the least squares sense means performing an *orthogonal projection of* $\boldsymbol{y}$ on the linear subspace generated by the *columns of* $\mathbf{A}$.

- One can apply the same procedure as for term selection in polynomial prediction, that is a Gram-Schmidt orthogonalization on selected columns (*orthogonal least squares*, OLS).
- At each iteration, one adds in $\mathbf{A}$ the column yielding maximum erro reduction. This column is defined by a pair $(\boldsymbol{c}_j, \beta_j)$, and center $\boldsymbol{c}_j$ is selected among the vectors $\{\boldsymbol{x}_i\}$.

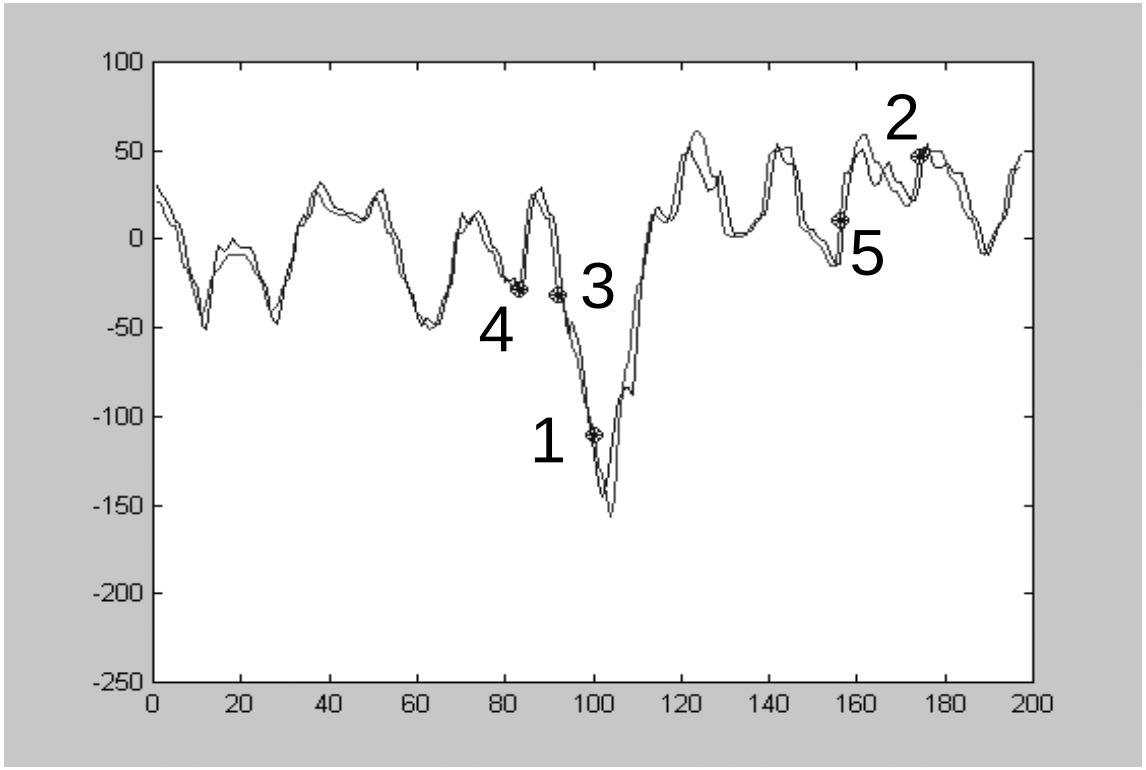
- One proceeds as follows:

1) The first center is selected by examining all the $\{\boldsymbol{x}_i\}$, and keeping the smallest least squares error.

2) Successive centers are chosen among the remaining $\{\boldsymbol{x}_i\}$, with an orthogonalization of $[\phi(\boldsymbol{x}_1),\ldots, \phi(\boldsymbol{x}_N)]^T$ with respect to the terms already selected. In this way the center giving maximum error reduction is spotted.

- Example: prediction of RR intervals ($p$ = 3, 5 RBF, multiple $\beta$). ESR: - 11 dB

Signal
Prediction
• samples corresponding to selected $\boldsymbol{x}_i$



# How many RBFs ?



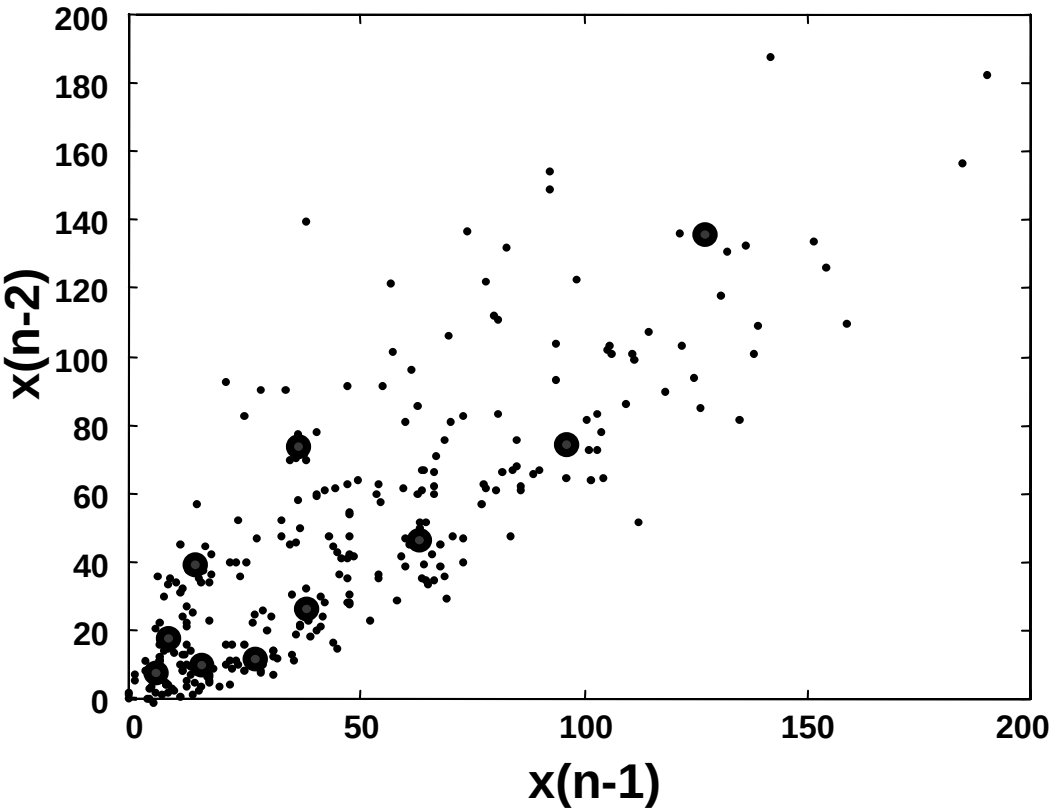
$$MDL = \underbrace{N \cdot \ln \sigma_e^2}_{\text{Data}} + \underbrace{M \cdot \ln N}_{\text{Model}} + \underbrace{?}_{\text{Model}}$$

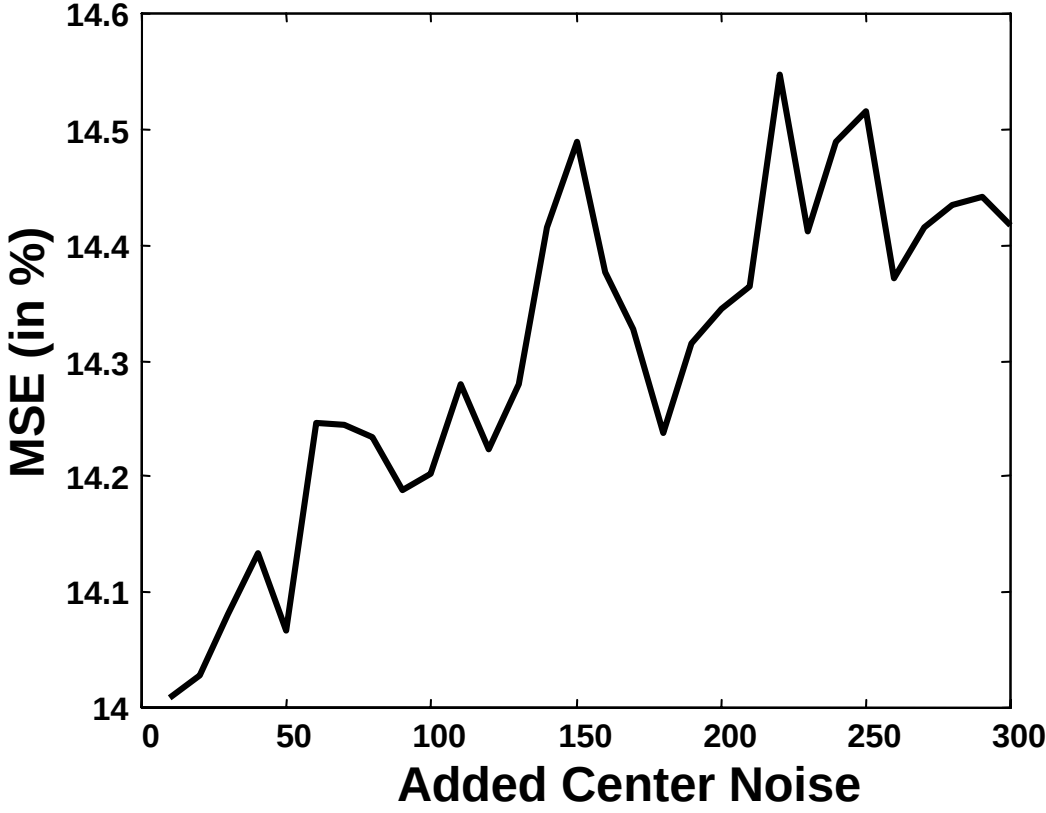
linear parameters only !

nonlinear parameters

# Clustering of the State Space (Sunspot Series)

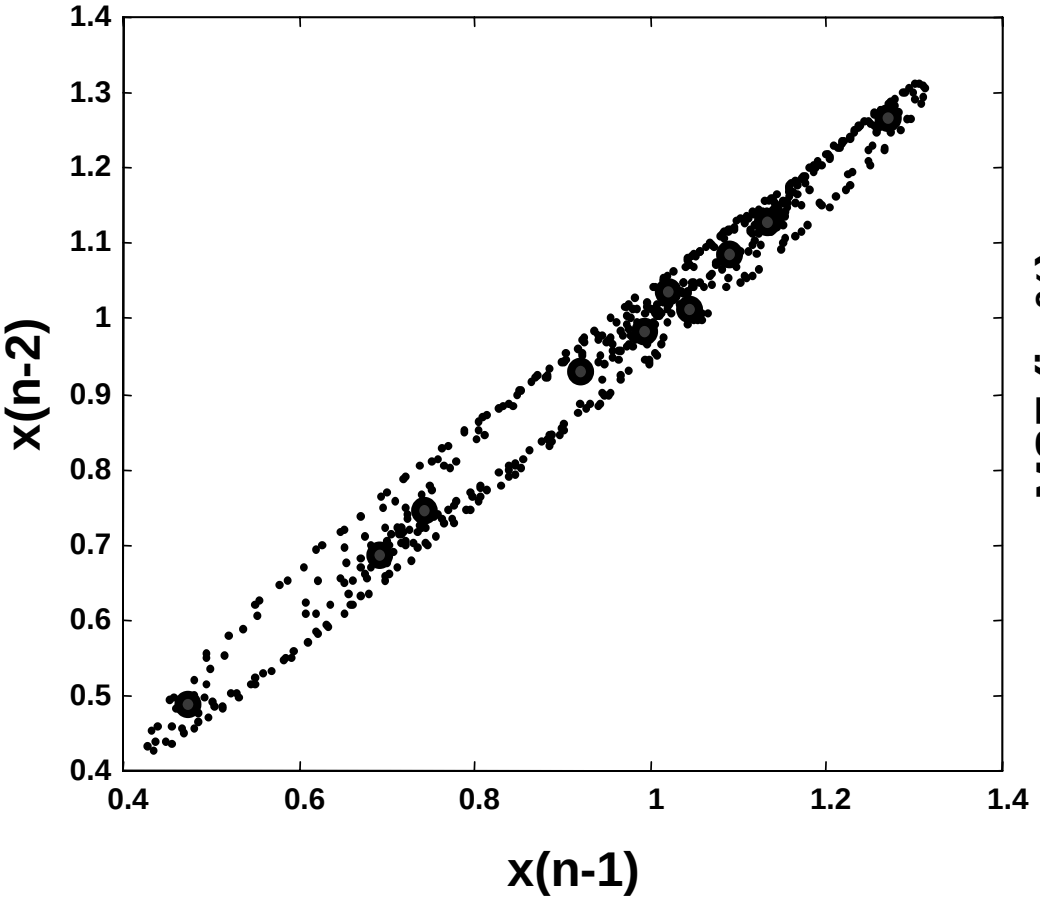


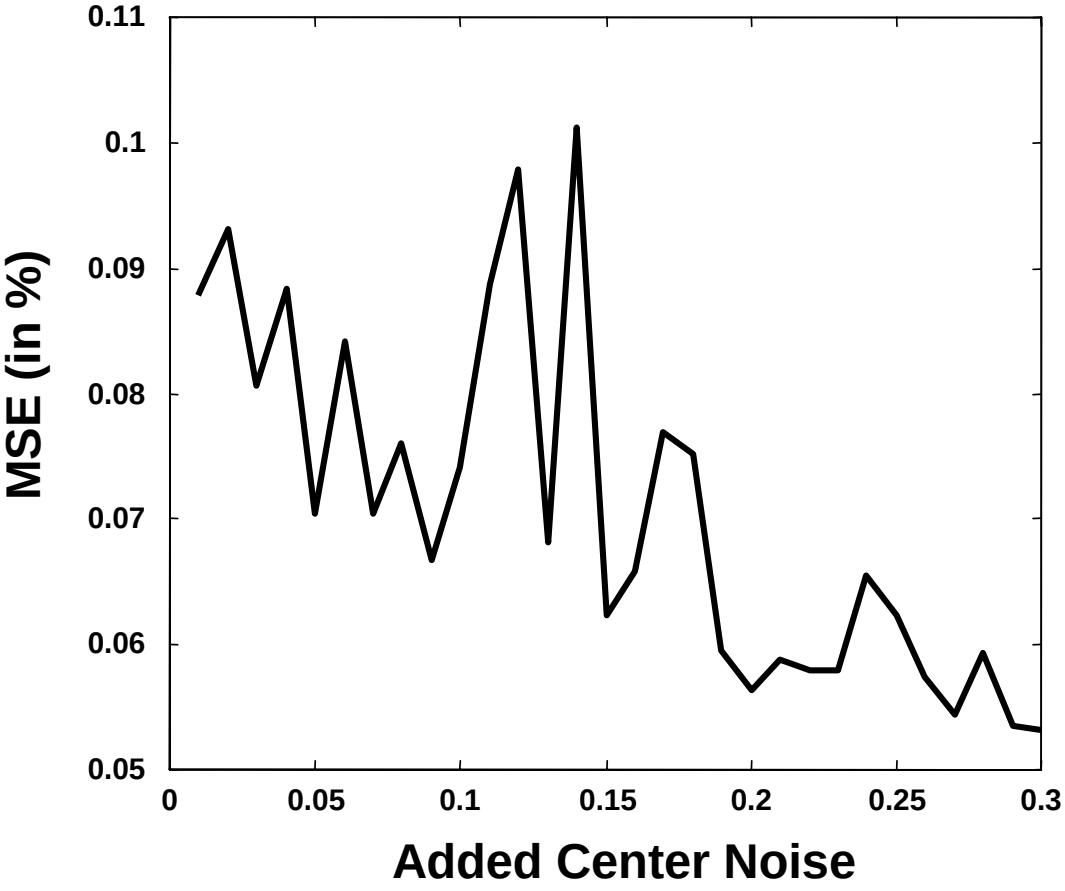


# Mackey-Glass Time Series





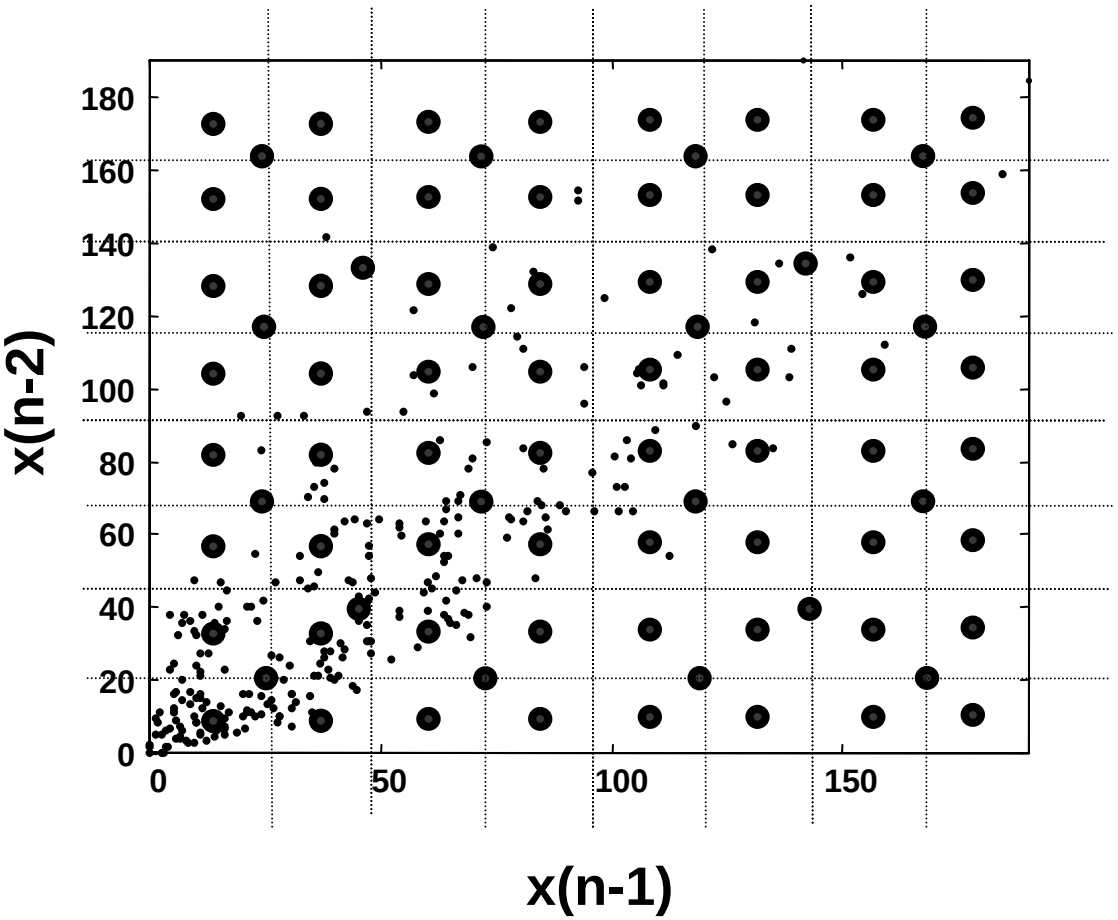



# State-space Quantization

2 bits

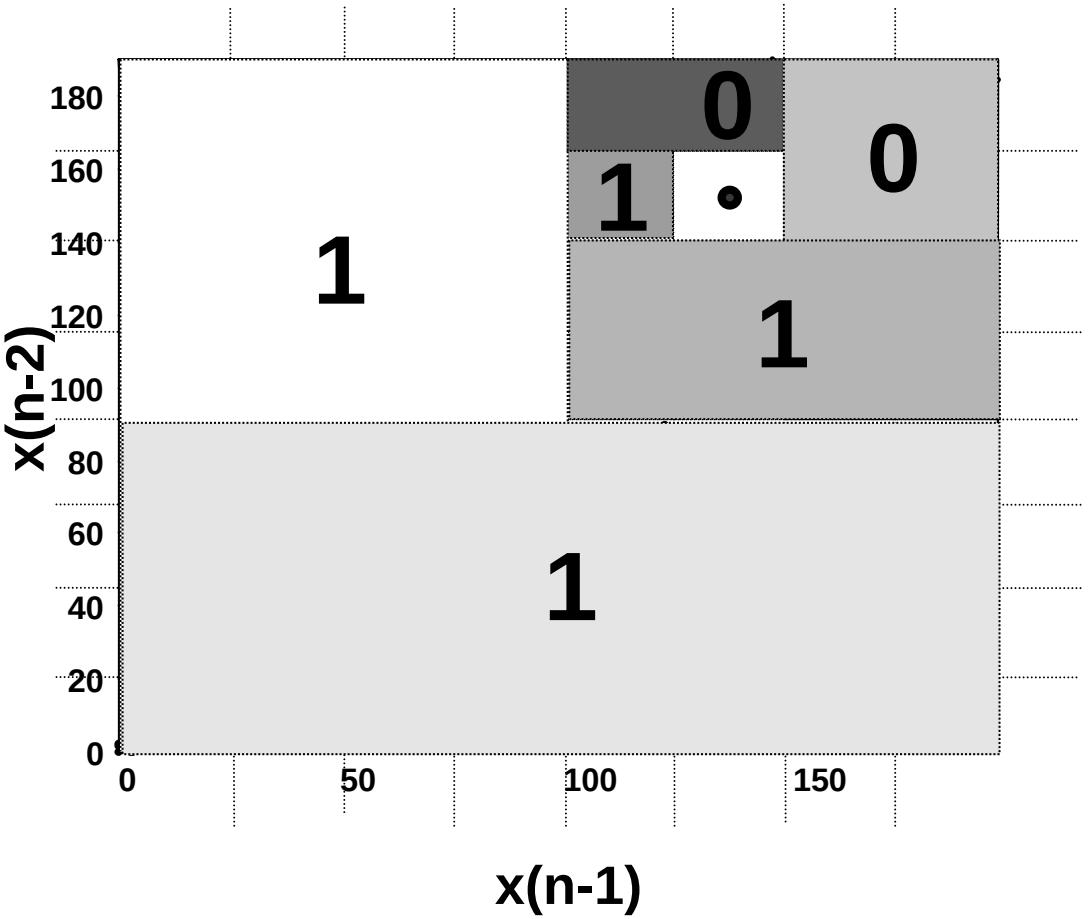
4 bits

6 bits



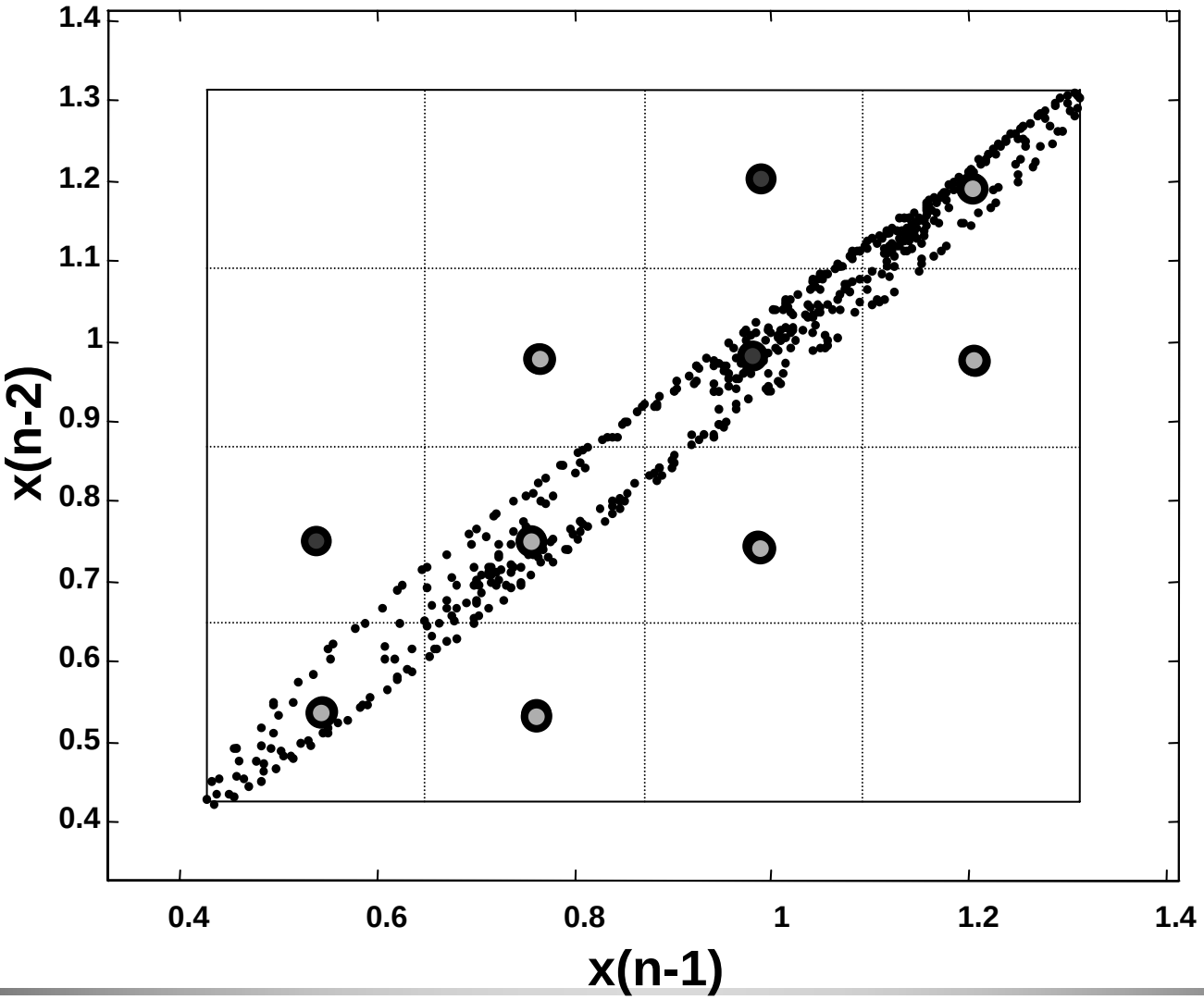
# The Coding of a Center

**1 1 0 1 1 0**



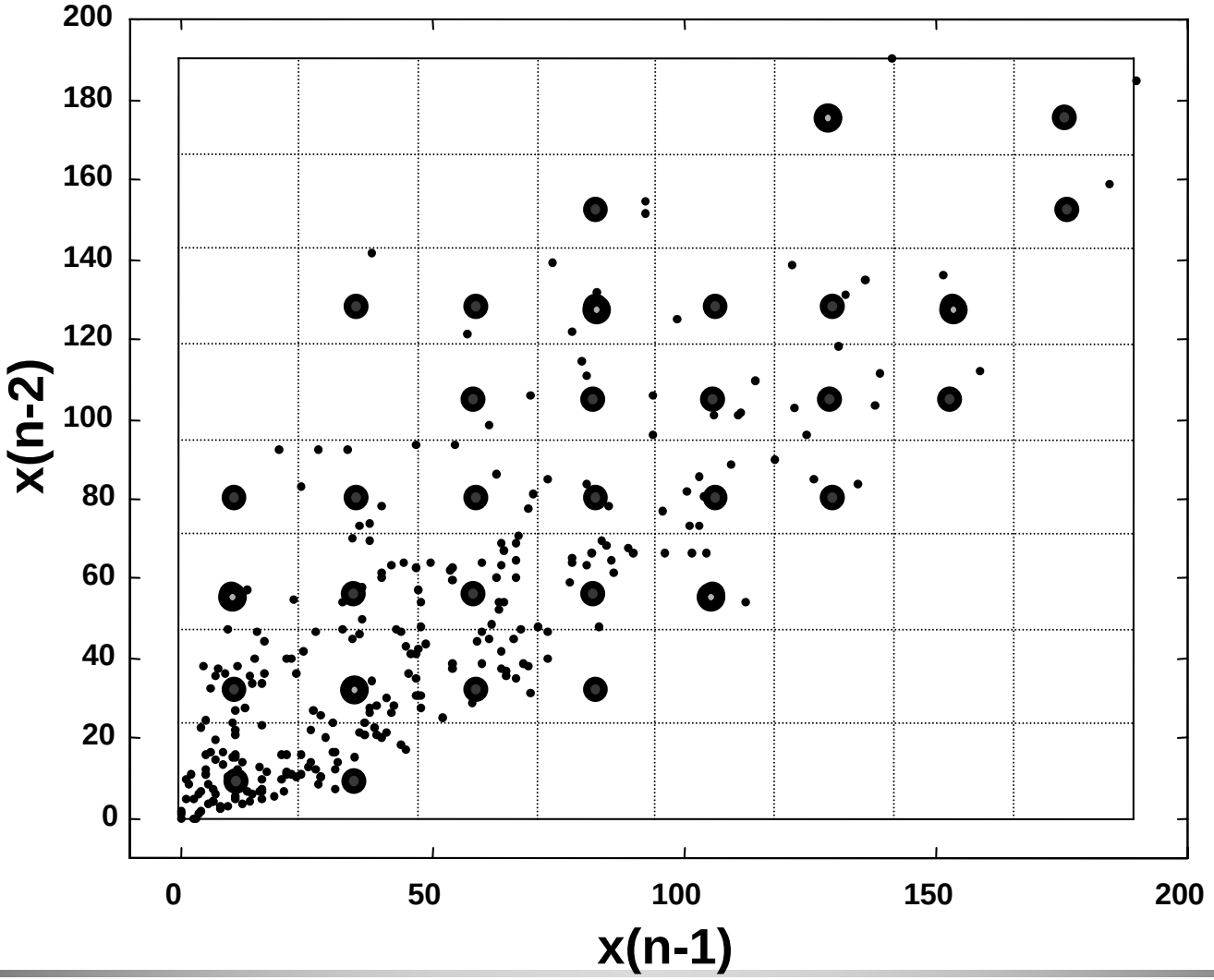
# The Best Subset of Centers (4 bits)





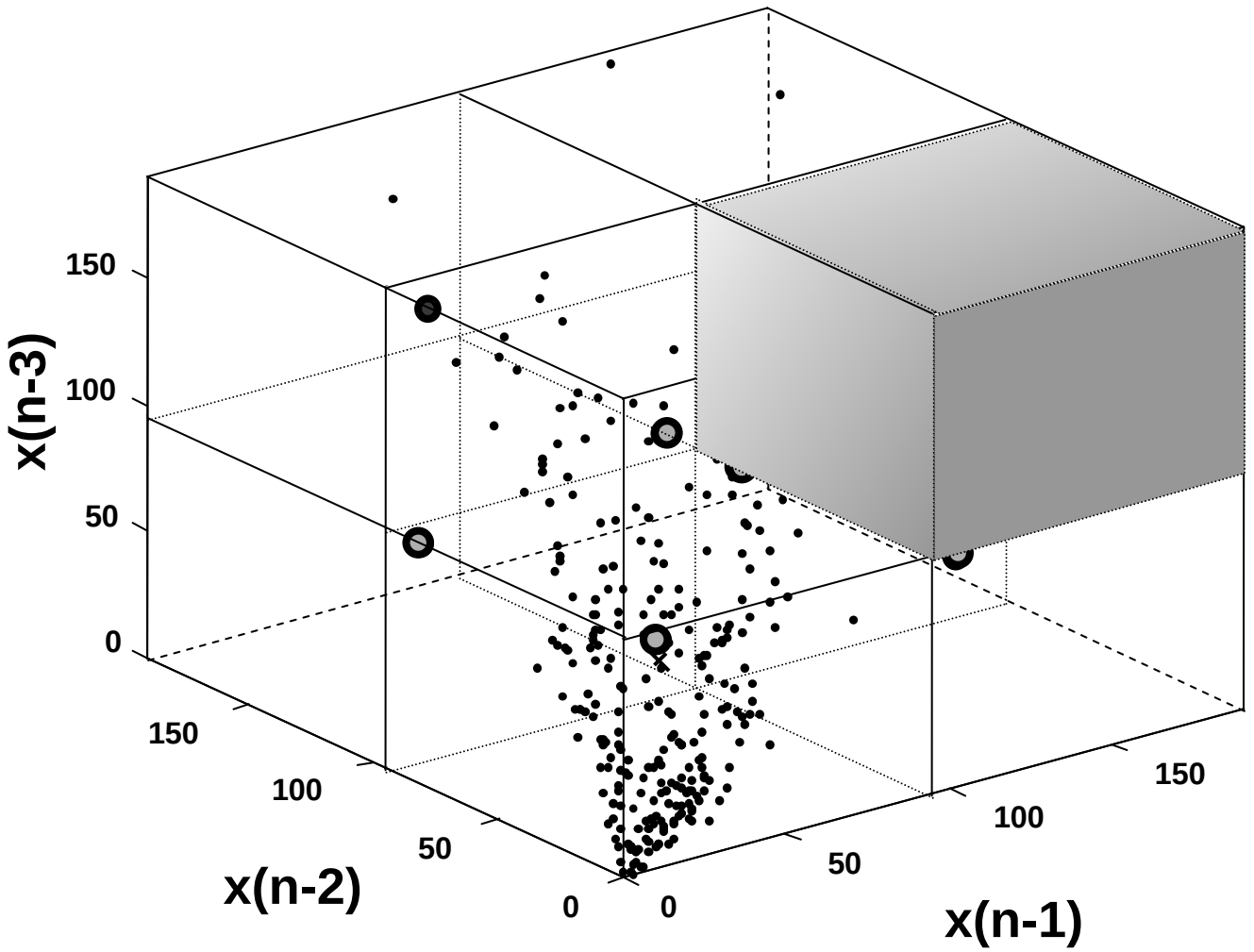
# The Best Subset of Centers (6 bits)





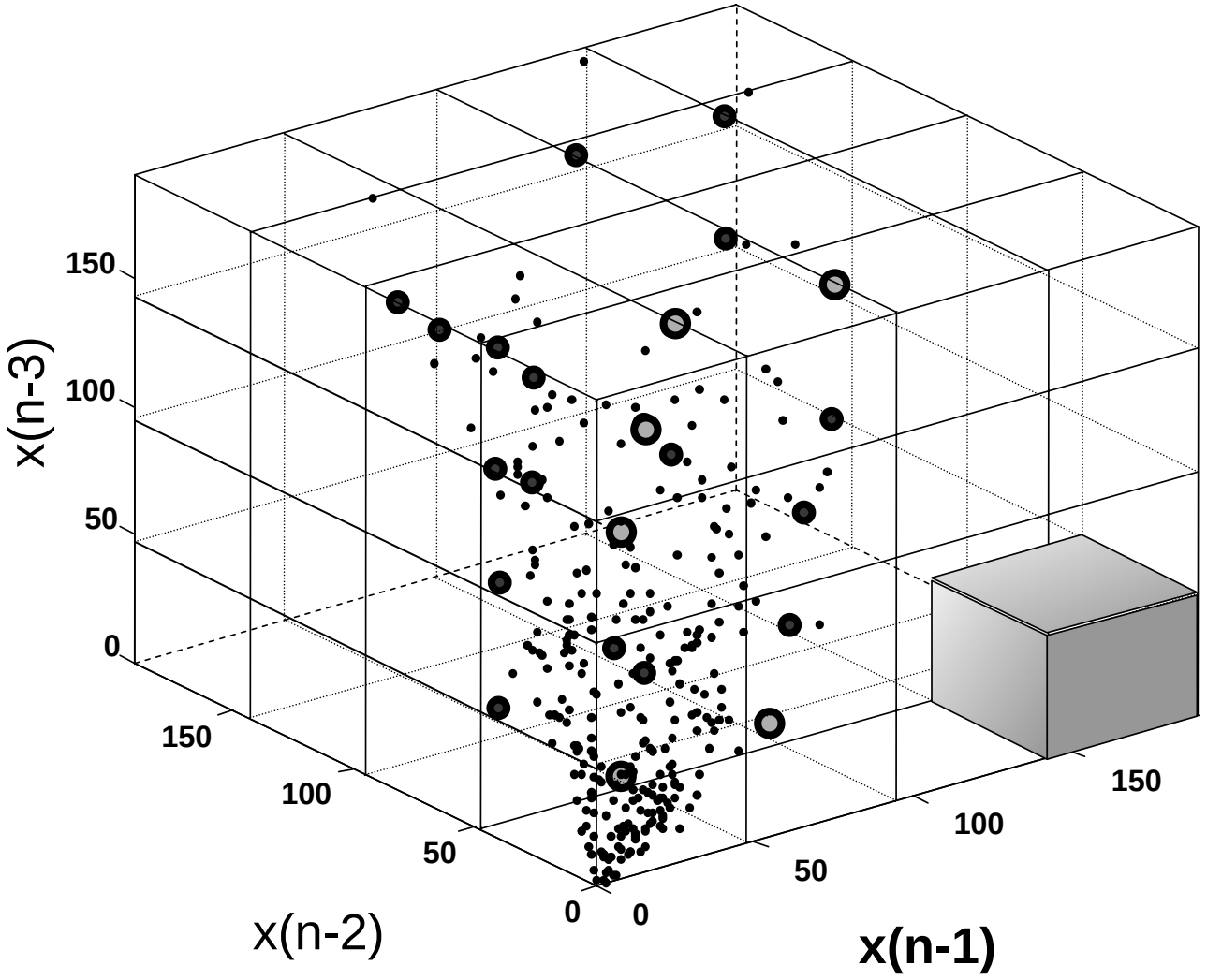
# 3-D State Space Quantization (3 bits)





# 3-D State Space Quantization (6 bits)



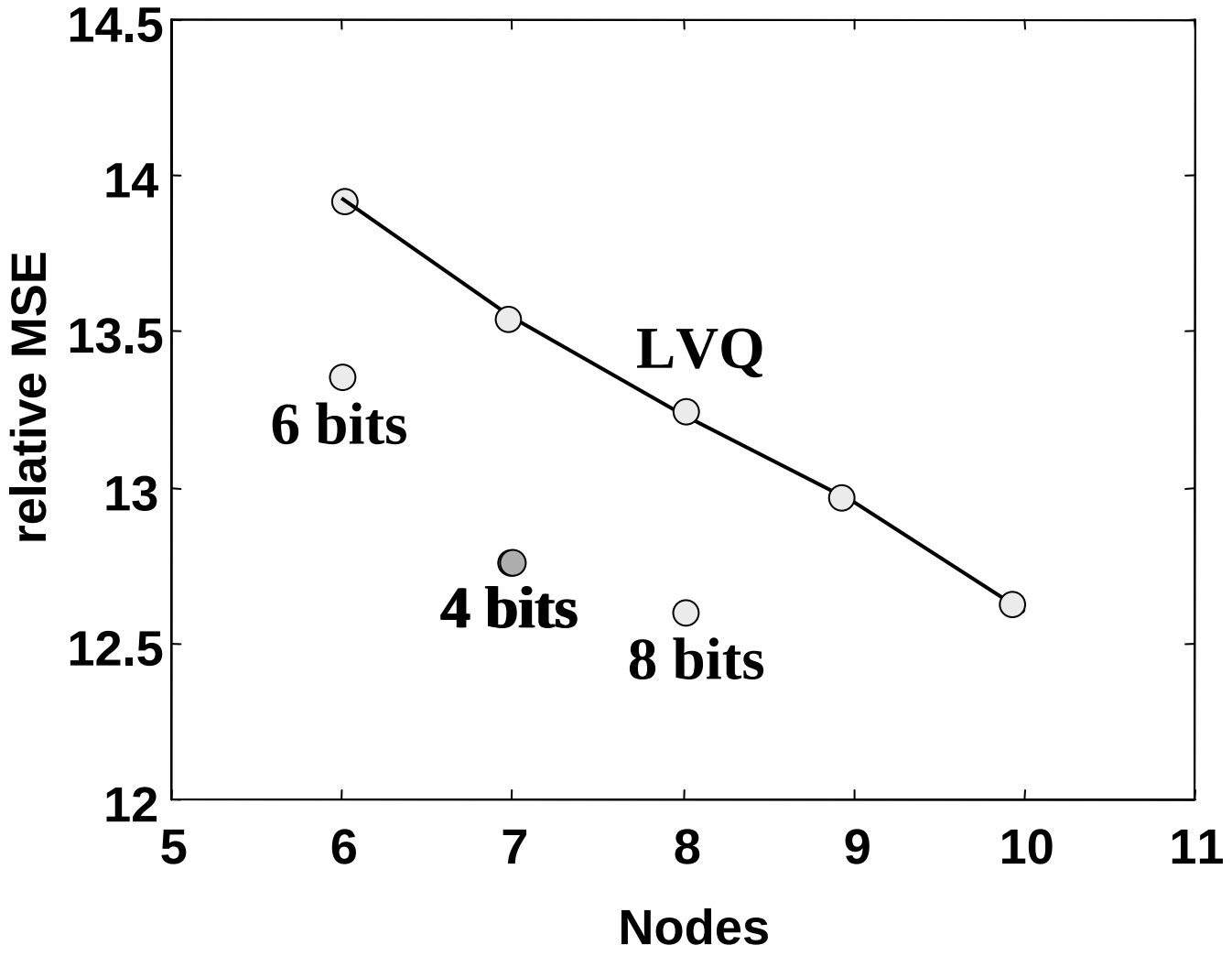


$$MDL = N \cdot \ln \sigma_e^2 + M \cdot (\ln N + B \cdot \ln 2)$$

**B: Number of bits to encode one center**

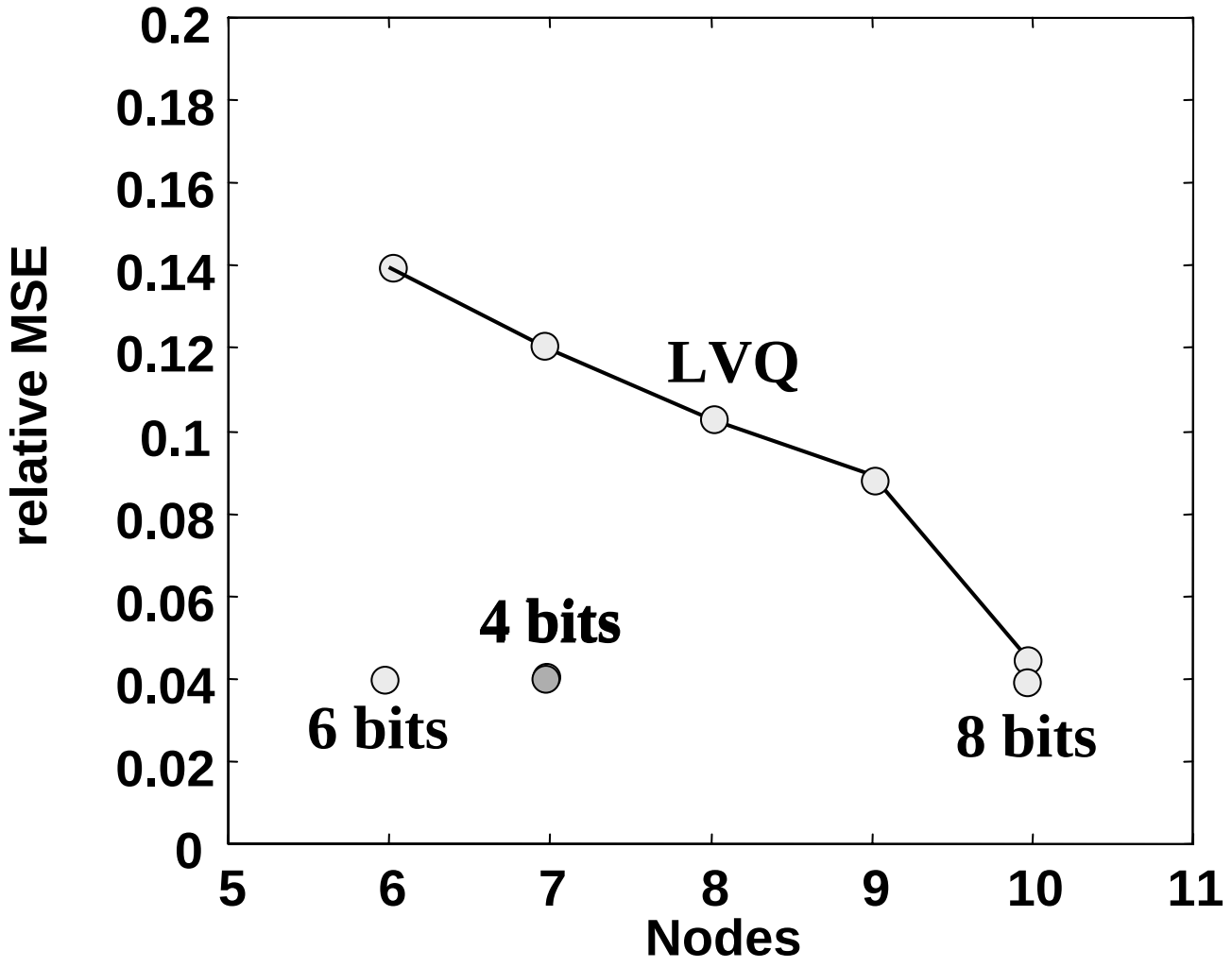
# Prediction Comparison (Sunspot)





# Prediction Comparison (Mackey-Glass)





- It is of course possible to derive an adaptive version of RBF network prediction.
- In what follows, reference to time index $k$ will not be explicit.

$\boldsymbol{w}(i)$ ith component of coefficient vector $\boldsymbol{w}$

$\phi(i)$ ith RBF

$\boldsymbol{\phi}$ RBF vector

- instantaneous square error is thus

$$\varepsilon^2 = [y - G(\boldsymbol{x})]^2 = [y - \boldsymbol{w}^T\boldsymbol{\phi}]^2$$

- In LMS, the updates are:

$$\boldsymbol{w} \rightarrow \boldsymbol{w} - \mu_1 \frac{\partial \varepsilon^2}{\partial \boldsymbol{w}} = \boldsymbol{w} - 2\mu_1 \varepsilon \frac{\partial \varepsilon}{\partial \boldsymbol{w}} = \boldsymbol{w} + 2\mu_1 \varepsilon \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w}}$$

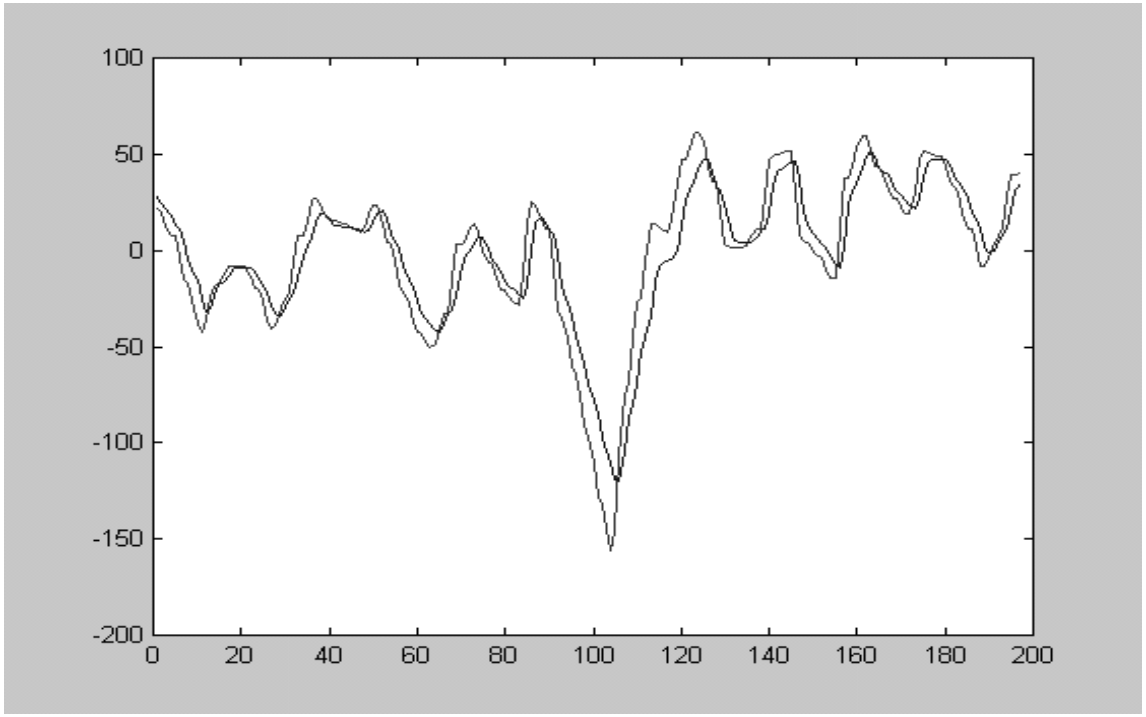
$$= \boldsymbol{w} + 2\mu_1 \varepsilon \boldsymbol{\phi}$$

$$\boldsymbol{c}_j \rightarrow \boldsymbol{c}_j + 2\mu_2 \varepsilon \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{c}_j} = \boldsymbol{c}_j + 2\mu_2 \varepsilon \boldsymbol{w}(j) \boldsymbol{\phi}(j)(\boldsymbol{x} - \boldsymbol{c}_j)$$

$$\beta \rightarrow \beta + 2\mu_3 \varepsilon \frac{\partial G(\boldsymbol{x})}{\partial \beta} = \beta + \frac{2\mu_3 \varepsilon}{\beta^3} \sum_{i=1}^{M} \|\boldsymbol{x} - \boldsymbol{c}_i\|^2 \boldsymbol{\phi}(i)$$

- Example: prediction of RR intervals ($p$ = 3, 5 RBF, unique $\beta$). ESR: - 8 dB

Signal
Prediction




*Multiple epochs before convergence*

- Remarks:
  - Better thsn batch algorithm, but not impressive
  - Tuning of adaptation parameters is uneasy (except for $\boldsymbol{w}$ for which it is possible to normalize by $\boldsymbol{\phi}^T\boldsymbol{\phi}$)
- *But maybe the gradient approach is not suited to center and $\beta$ parameter update*. For instance, nothing guarantees that the mean suqare error has a unique minimum.

- Why not *decouple* the update of $\boldsymbol{w}$, for which LMS should work well, from that of the centers and $\beta$?
- One might use an LVQ-style update for the centers, then modify $\beta$ by using the empirical rule based on the distance between centers.
- But it is necessary to define a link between the two update schemes.

1. One first computes the modification on $\boldsymbol{c}_i$, the center closest to $\boldsymbol{x}$ with LVQ (but $\alpha$ must remain constant to preserve adaptability):

$$\boldsymbol{c}_i \rightarrow \boldsymbol{c}_i + \Delta\boldsymbol{c}_i = \boldsymbol{c}_i + \alpha(\boldsymbol{x} - \boldsymbol{c}_i)$$

2. If the maximum distance between centers changes, $\beta$ should become $\beta + \Delta\beta$, by using the empirical rule for unique $\beta$.

The output changes from $G(\boldsymbol{x})$ to $G(\boldsymbol{x}) + \Delta G(\boldsymbol{x})$.

- But: $$\Delta G(\boldsymbol{x}) \approx (\Delta \boldsymbol{c}_i)^{\mathsf{T}} \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{c}_i} + \Delta\beta \frac{\partial G(\boldsymbol{x})}{\partial \beta}$$
- Thus:

$$\frac{\partial\big(G(\boldsymbol{x}) + \Delta G(\boldsymbol{x})\big)}{\partial \boldsymbol{w}} \approx \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w}} + (\Delta \boldsymbol{c}_i)^{\mathsf{T}} \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w}\partial \boldsymbol{c}_i} + \Delta\beta \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w}\partial \beta}$$

- And the instantaneous square error becomes:

$$\varepsilon^2 = [y - \mathrm{G}(\boldsymbol{x}) - \Delta G(\boldsymbol{x})]^2$$

The coefficient vector $\boldsymbol{w}$ is updated with:

$$\boldsymbol{w} \rightarrow \boldsymbol{w} - \mu_1 \frac{\partial \varepsilon^2}{\partial \boldsymbol{w}} = \boldsymbol{w} + 2\mu\varepsilon \frac{\partial\big(G(\boldsymbol{x}) + \Delta G(\boldsymbol{x})\big)}{\partial \boldsymbol{w}}$$

- With some computations:

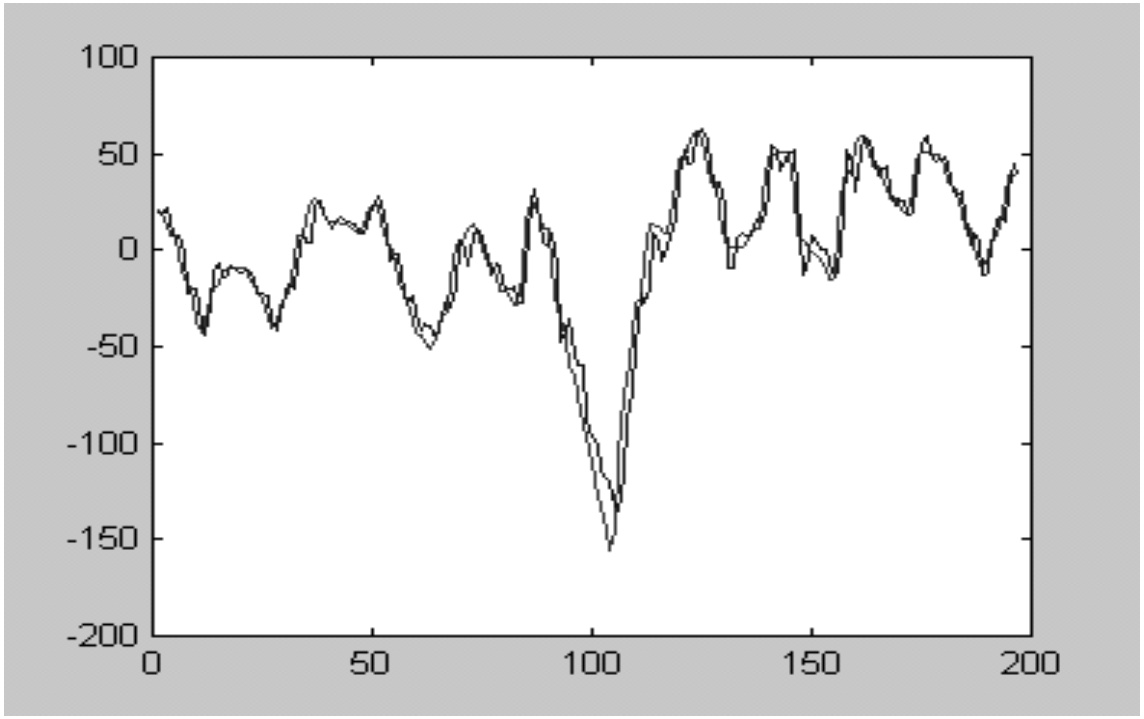
$$(\Delta \boldsymbol{c}_i)^{\mathsf{T}} \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w} \partial \boldsymbol{c}_i} = \frac{\alpha}{\beta^2} \|\boldsymbol{x} - \boldsymbol{c}_i\|^2 \boldsymbol{\phi}(i) \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

ith position

$$\frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{w} \partial \beta} = \frac{1}{\beta^3} \begin{bmatrix} \|\boldsymbol{x} - \boldsymbol{c}_1\|^2 \boldsymbol{\phi}(1) \\ \vdots \\ \|\boldsymbol{x} - \boldsymbol{c}_M\|^2 \boldsymbol{\phi}(M) \end{bmatrix}$$

- Example: prediction of RR intervals ($p$ = 3, 5 RBF, unique $\beta$). ESR: - 10.7 dB

Signal
Prediction




*Multiple epochs before convergence*

- Remarques:
  - Prediction performance improves in the non-stationary intervals.
  - It is a lot easier to tune $\mu$ for $\boldsymbol{w}$ and $\alpha$ for the centers.
  - This approach is easily modified for the multiple $\beta$ case.

1. S. Haykin, *Neural Networks, a Comprehensive Foundation*, McMillan, NY, 1994.
2. J. C. Principe, N. R. Euliano, and W. Curt Lefebvre, *Neural and Adaptive Systems*, J. Wiley, NY, 2000.
3. S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, 1991, pp. 302-309.
4. B. Mulgrew, "Applying radial basis functions," *IEEE Signal Process. Mag.*, vol. 13, no. 2, 1996, pp. 50-65.