Optimal Kernel Filtering for System Identification

Jose C. Principe

Computational NeuroEngineering Laboratory (CNEL)

University of Florida

principe@cnel.ufl.edu

Acknowledgments

Dr. Weifeng Liu, Amazon

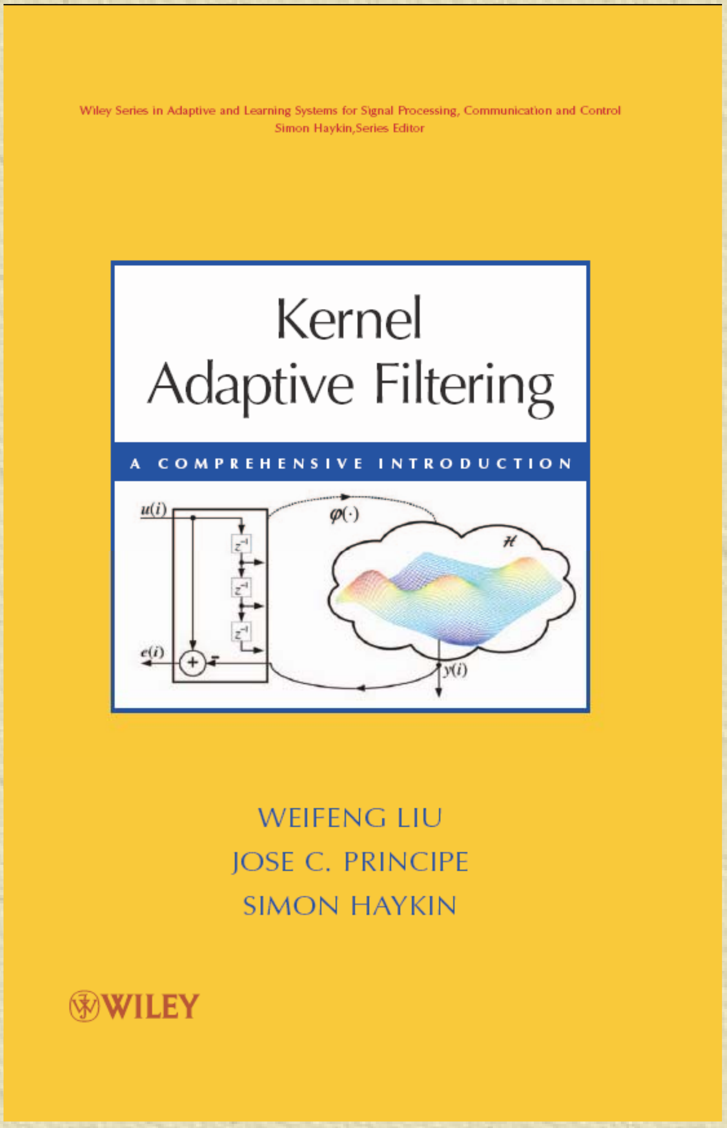
Dr. Badong Chen, Tsinghua University and Post Doc CNEL

NSF ECS – 0300340, 0601271, 0964197 (Neuroengineering program)

Outline

1. Introduction
2. Least-mean-square algorithm in Kernel Space
3. Kernel Affine Projection Algorithms and Recursive Least Squares
4. Active Learning in Kernel Filtering
5. Conclusion

Wiley Book (2010)


Wiley Series in Adaptive and Learning Systems for Signal Processing, Communication and Control
Simon Haykin, Series Editor

Kernel Adaptive Filtering

A COMPREHENSIVE INTRODUCTION

WEIFENG LIU
JOSE C. PRINCIPE
SIMON HAYKIN

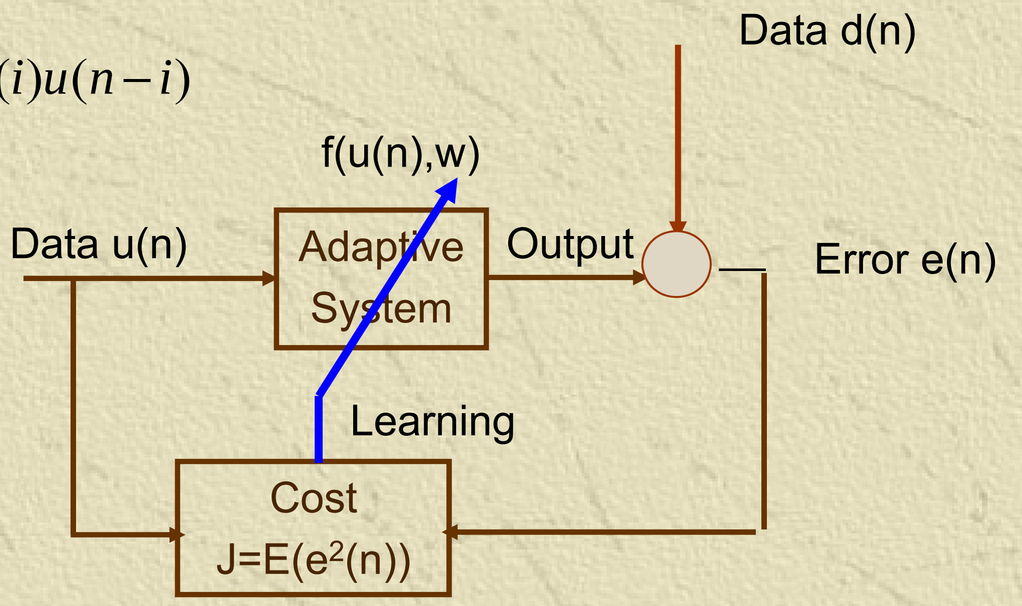
WILEY

Papers are available at
www.cnel.ufl.edu

Optimal System ID Fundamentals

- System identification is regression in functional spaces: Given data pairs {u(n),d(n)} and a functional mapper y=f(u,w), minimize J(e)

$$y(n)=\sum_{i=0}^{N-1} w(i)u(n-i)$$



- Optimal solution is least squares $w^* = R^{-1}p$ where *R* is the autocorrelation matrix of the input data over the lags and *p* is the crosscorrelation vector between input and desired.

On-Line Learning for Linear Filters

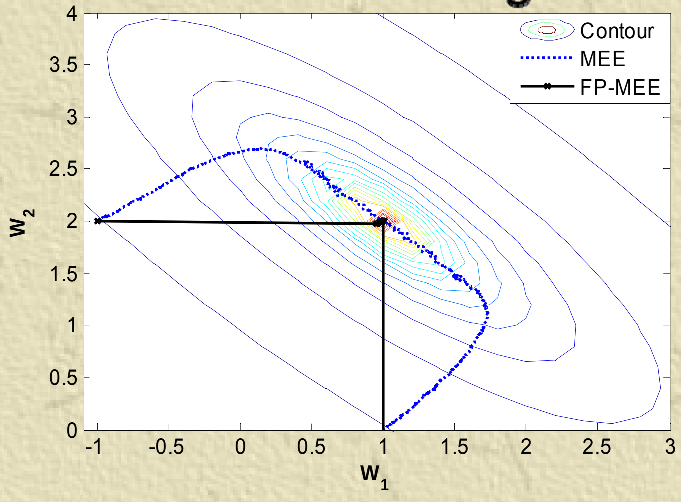
- Easiest technique is to search the performance surface *J* using **gradient descent learning** (batch).

$$w_i = w_{i-1} - \eta \nabla J_i \qquad w_i = w_{i-1} - \eta H^{-1} \nabla J_{i-1}$$

$$\lim_i E[w_i] = w^*$$

$$J = E[e^2(i)]$$

$$\eta \qquad step\ size$$



- Gradient descent learning has well known compromises:
  - Stepsize η must be smaller than $1/\lambda_{max}$ (of R) for convergence
  - Speed of adaptation is controlled by λ_{min}
  - So eigenvalue spread of signal autocorrelation matrix controls speed of adaptation
  - The misadjustment (penalty w.r.t. optimum error) is proportional to stepsize, so fundamental compromise between adapting fast, and small misadjustment.

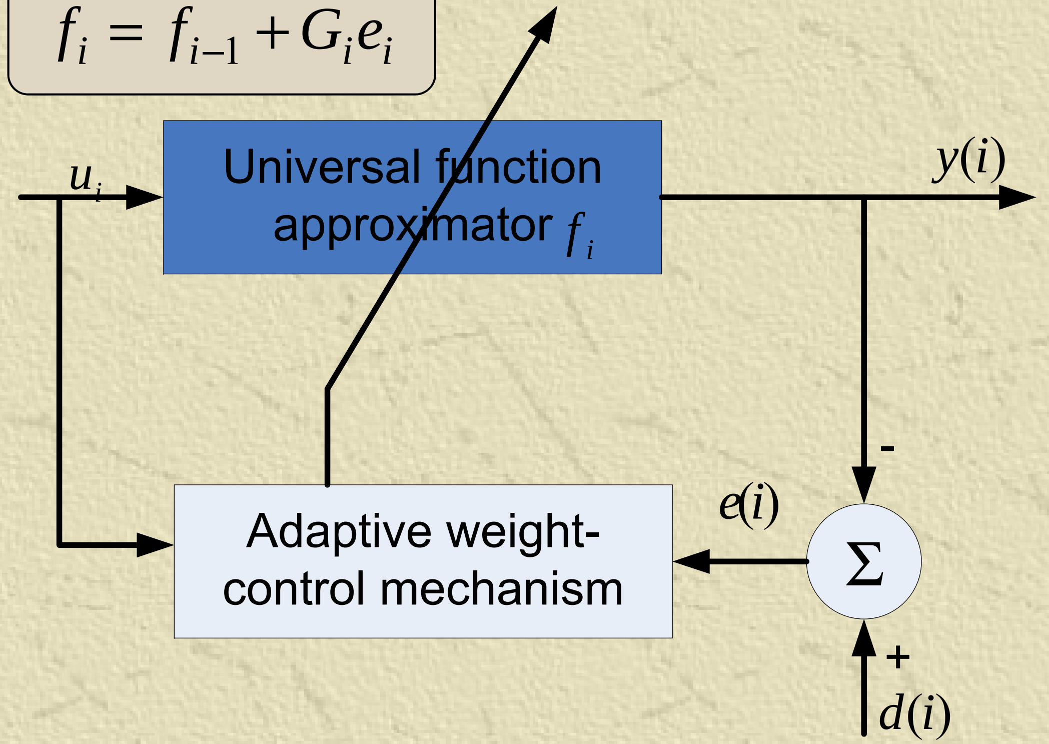
On-Line Learning for Non-Linear Filters?

- Can we generalize $w_i = w_{i-1} + G_i e_i$ to *nonlinear* models?

$$y = w^T u \longrightarrow y = f(u)$$

and create incrementally the nonlinear mapping?

$$\boxed{f_i = f_{i-1} + G_i e_i}$$



u_i
Universal function approximator f_i
$y(i)$
-
$e(i)$
Σ
Adaptive weight-control mechanism
+
$d(i)$

Non-Linear Models - Traditional
(Fixed topologies)

- Hammerstein and Wiener models
  - An explicit nonlinearity followed (preceded) by a linear filter
  - Nonlinearity is problem dependent
  - Do not possess universal approximation property
- Multi-layer perceptrons (MLPs) with back-propagation
  - Non-convex optimization
  - Local minima
- Least-mean-square for radial basis function (RBF) networks
  - Non-convex optimization for adjustment of centers
  - Local minima
- Volterra models, Recurrent Networks, etc

Non-linear Methods with Kernels

- Universal approximation property (kernel dependent)
- Convex optimization (no local minima)
- Still easy to compute (kernel trick)
- But require regularization
- **Sequential (On-line) Learning with Kernels**

- (Platt 1991) Resource-allocating networks
  - Heuristic
  - No convergence and well-posedness analysis
- (Frieb 1999) Kernel adaline
  - Formulated in a batch mode
  - well-posedness not guaranteed
- (Kivinen 2004) Regularized kernel LMS
  - with explicit regularization
  - Solution is usually biased
- (Engel 2004) Kernel Recursive Least-Squares
- (Vaerenbergh 2006) Sliding-window kernel recursive least-squares
- Liu, Principe 2008,2009, 2010.

Neural Networks versus Kernel Filters

	ANNs	Kernel filters
Universal Approximators	YES	YES
Convex Optimization	NO	YES
Model Topology grows with data	NO	YES
Require Explicit Regularization	NO	YES/NO (KLMS)
Online Learning	YES	YES
Computational Complexity	LOW	MEDIUM

ANNs are semi-parametric, nonlinear approximators

Kernel filters are non-parametric, nonlinear approximators

Kernel Methods

- Kernel filters operate in a very special Hilbert space of functions called a Reproducing Kernel Hilbert Space (RKHS).
- A RKHS is an Hilbert space where all function evaluations are finite
- Operating with functions seems complicated and it is! But it becomes much easier in RKHS if we restrict the computation to inner products.
- Most linear algorithms can be expressed as inner products. Remember the FIR

$$y(n) = \sum_{i=0}^{L-1} w_i x(n-i) = \left\langle \mathbf{w}^{\mathbf{T}} \mathbf{x}(n) \right\rangle$$

Kernel methods

- Moore-Aronszajn theorem
  - Every symmetric positive definite function of two real variables has a unique Reproducing Kernel Hilbert Space (RKHS).

$$k(x, y) = \exp(-h\|x - y\|^2)$$

- Mercer's theorem
  - Let *K(x,y)* be symmetric positive definite. The kernel can be expanded in the series

$$\kappa(x, y) = \sum_{i=1}^{m} \lambda_i \varphi_i(x)\varphi_i(y)$$

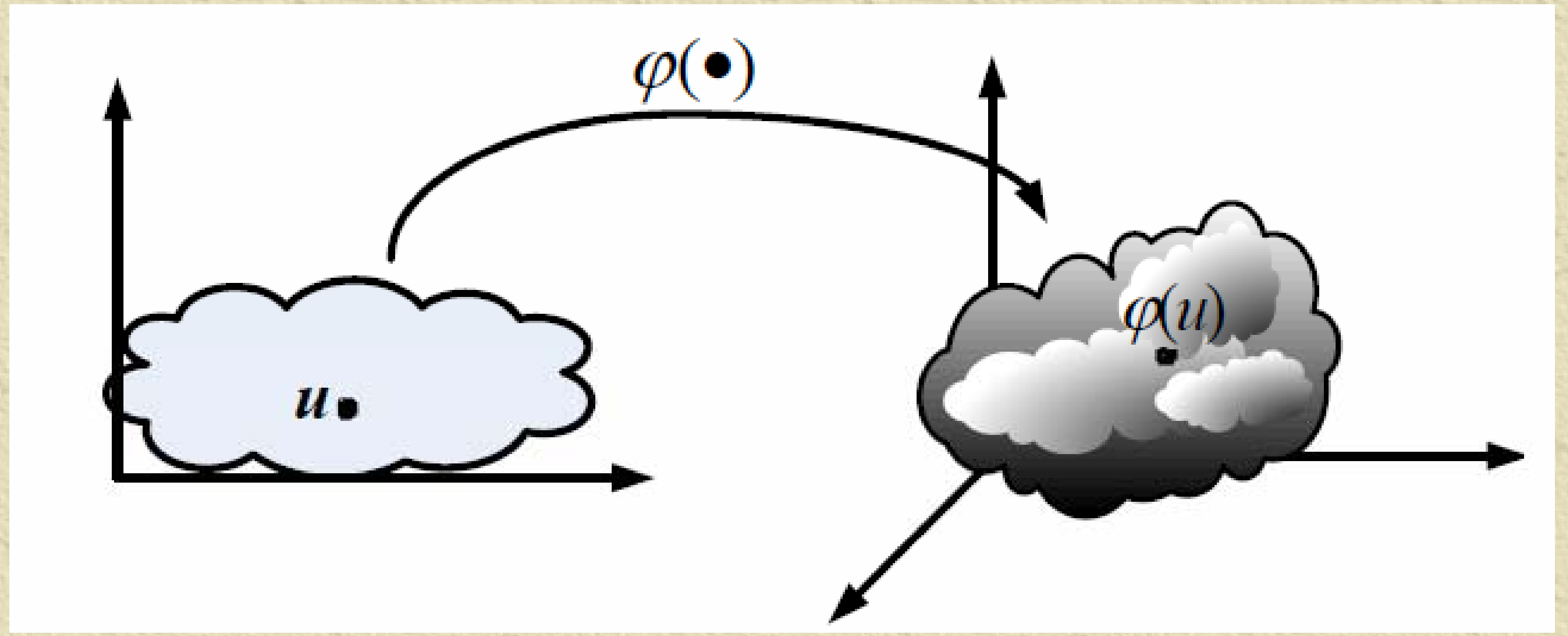
  - Construct the transform as

$$\varphi(x) = [\sqrt{\lambda_1}\varphi_1(x), \sqrt{\lambda_2}\varphi_2(x), \ldots, \sqrt{\lambda_m}\varphi_m(x)]^T$$

  - Inner product

$$\boxed{\langle \varphi(x), \varphi(y) \rangle = \kappa(x, y)}$$

Kernel methods



Mate L., Hilbert Space Methods in Science and Engineering, A. Hildger, 1989

Berlinet A., and Thomas-Agnan C., “Reproducing kernel Hilbert Spaces in probaability and Statistics, Kluwer 2004

Basic idea of on-line kernel filtering

- Transform data into a high dimensional feature space $\varphi_i := \varphi(u_i)$
- Construct a linear model in the feature space *F*

$$y = \langle \Omega, \varphi(u) \rangle_F$$

- Adapt iteratively parameters with gradient information

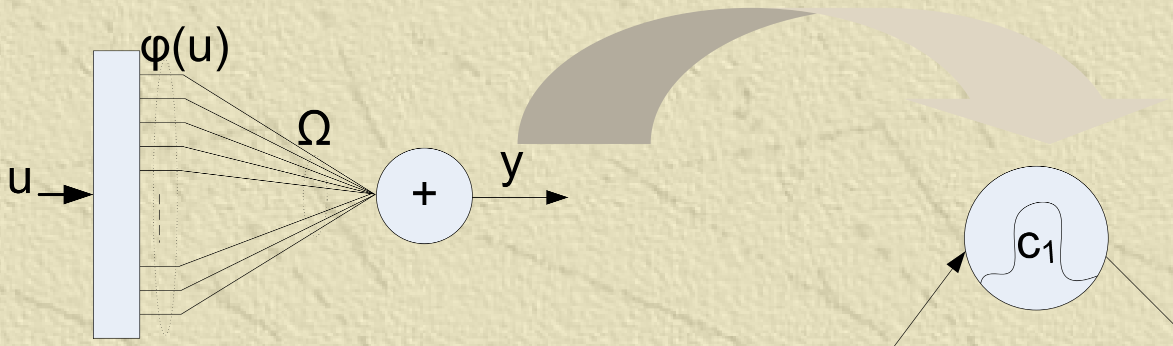
$$\Omega_i = \Omega_{i-1} - \eta \nabla J_i$$

- Compute the output

$$f_i(u) = \langle \Omega_i, \varphi(u) \rangle_F = \sum_{j=1}^{m_i} a_j \kappa(u, c_j)$$

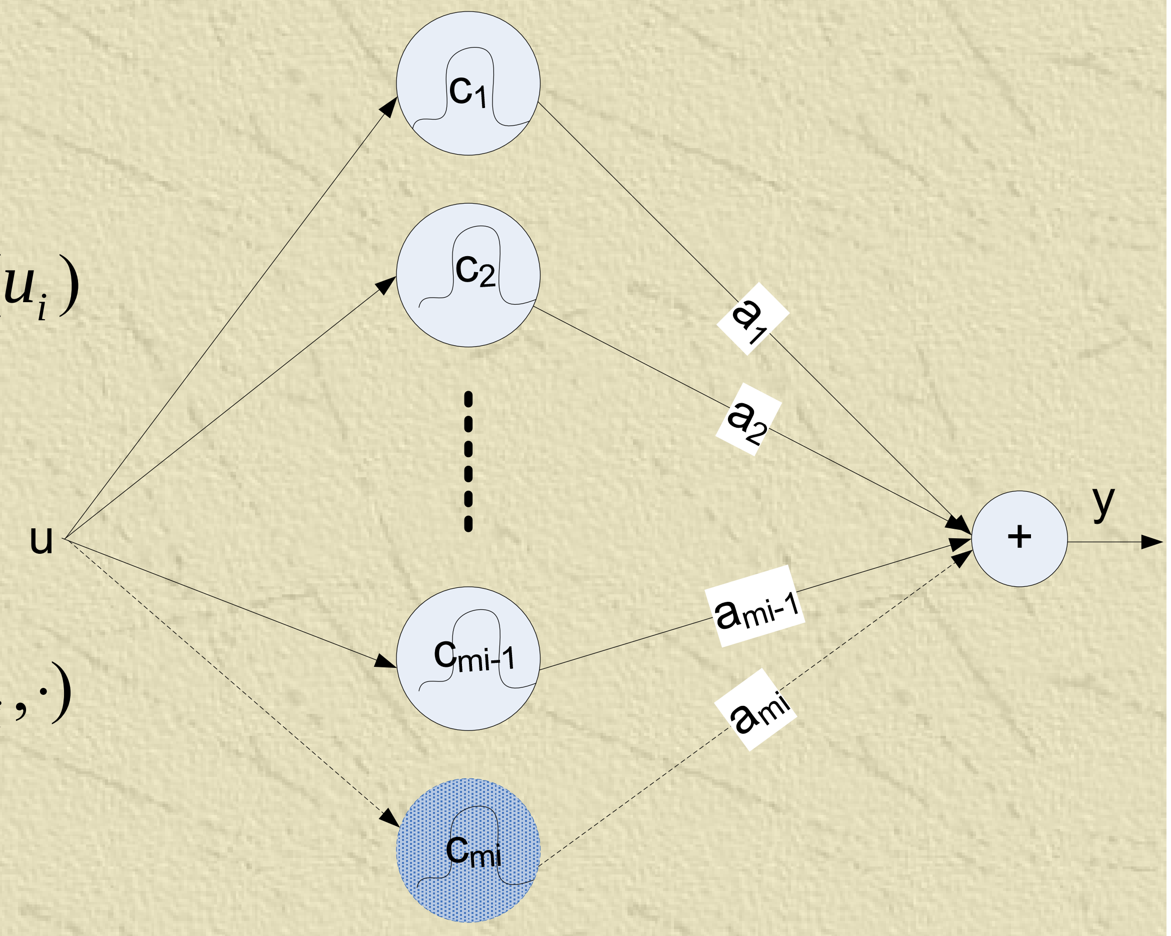
- Universal approximation theorem
  - For the Gaussian kernel and a sufficient large m_i, $f_i(u)$ can approximate any continuous input-output mapping arbitrarily close in the L_p norm.

Growing network structure



$$\Omega_i = \Omega_{i-1} + \eta e(i)\varphi(u_i)$$

$$f_i = f_{i-1} + \eta e(i)\kappa(u_i, \cdot)$$



Kernel Least-Mean-Square (KLMS)

- Least-mean-square

$$w_i = w_{i-1} + \eta u_i e(i) \qquad e(i) = d(i) - w_{i-1}^T u_i \qquad w_0$$

- Transform data into a high dimensional feature space *F* $\varphi_i := \varphi(u_i)$

$$\Omega_0 = 0$$

$$e(i) = d(i) - \langle \Omega_{i-1}, \varphi(u_i) \rangle_F$$

$$\Omega_i = \Omega_{i-1} + \eta \varphi(u_i) e(i)$$

$$\Omega_i = \sum_{j=1}^{i} \eta e(j) \varphi(u_j)$$

$$f_i(u) = \langle \Omega_i, \varphi(u) \rangle_F = \sum_{j=1}^{i} \eta e(j) \kappa(u, u_j)$$

$$\Omega_0 = 0$$

$$e(1) = d(1) - \langle \Omega_0, \varphi(u_1) \rangle_F = d(1)$$

$$\Omega_1 = \Omega_0 + \eta \varphi(u_1) e(1) = a_1 \varphi(u_1)$$

$$\begin{aligned} e(2) &= d(2) - \langle \Omega_1, \varphi(u_2) \rangle_F \\ &= d(2) - \langle a_1 \varphi(u_1), \varphi(u_2) \rangle_F \\ &= d(2) - a_1 \kappa(u_1, u_2) \end{aligned}$$

$$\begin{aligned} \Omega_2 &= \Omega_1 + \eta \varphi(u_2) e(2) \\ &= a_1 \varphi(u_1) + a_2 \varphi(u_2) \end{aligned}$$

$$\vdots$$

- RBF Centers are the samples, and Weights are the errors!

Kernel Least-Mean-Square (KLMS)

$$f_{i-1} = \eta \sum_{j=1}^{i-1} e(j)\kappa(\mathbf{u}(j),.)$$

$$f_{i-1}(\mathbf{u}(i)) = \eta \sum_{j=1}^{i-1} e(j)\kappa(\mathbf{u}(j),\mathbf{u}(i))$$

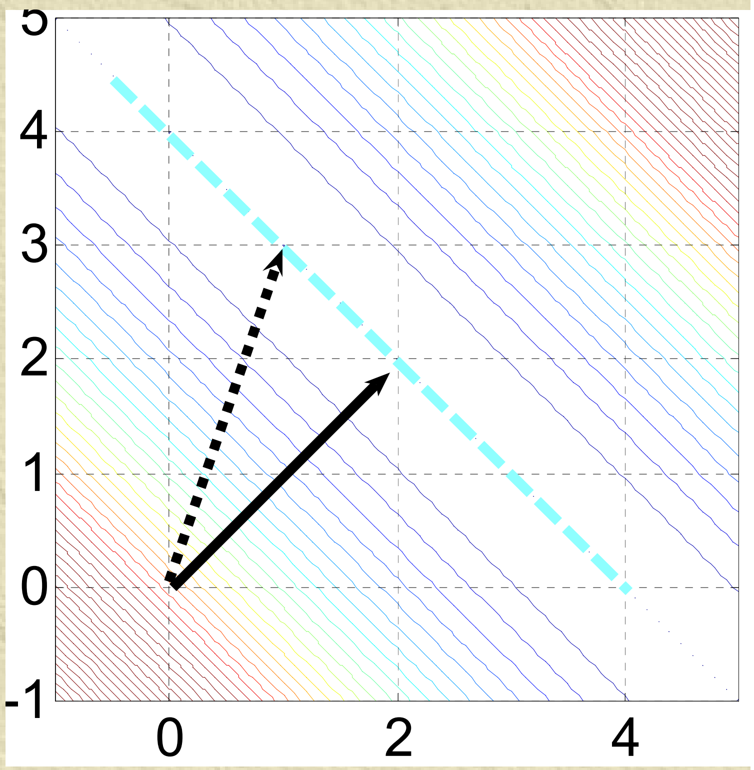
$$e(i) = d(i) - f_{i-1}(\mathbf{u}(i))$$

$$f_i = f_{i-1} + \eta e(i)\kappa(\mathbf{u}(i),.)$$

Free Parameters in KLMS
Initial Condition

- The initialization $\Omega_0 = 0$ gives the minimum possible norm solution.
- $$\Omega_i = \sum_{n=1}^{m} c_n P_n$$
  $$\varsigma_1 \geq \ldots \geq \varsigma_k > 0$$
  $$\varsigma_{k+1} = \ldots = \varsigma_m = 0$$

$$\| \Omega_i \|^2 = \sum_{n=1}^{k} \| c_n \|^2 + \sum_{n=k+1}^{m} \| c_n \|^2$$



Liu W., Pokarel P., Principe J., "The Kernel LMS Algorithm", IEEE Trans. Signal Processing, Vol 56, # 2, 543 – 554, 2008.

Free Parameters in KLMS
Step size

- Traditional wisdom in LMS still applies here.

$$\eta < \frac{N}{tr[\mathbf{G}_\varphi]} = \frac{N}{\sum_{j=1}^{N} \kappa(\mathbf{u}(j), \mathbf{u}(j))}$$

  where $\mathbf{G}_\varphi$ is the Gram matrix, and N its dimensionality.
- For translation invariant kernels, $\kappa(u(j),u(j))=g_0$, is a constant independent of the data.
- The misadjustment is therefore $M = \frac{\eta}{2N} tr[\mathbf{G}_\varphi]$

Free Parameters in KLMS
Rule of Thumb for *h*

- Although KLMS is not kernel density estimation, these rules of thumb still provide a starting point.
- Silverman’s rule can be applied

$$h = 1.06 \min\{\sigma, R/1.34\} N^{-1/(5L)}$$

  where σ is the input data s.d., R is the interquartile, N is the number of samples and L is the dimension.
- Alternatively: take a look at the dynamic range of the data, assume it uniformly distributed and select h to put 10 samples in 3 σ.
- Use cross validation for more accurate estimation

Free Parameters in KLMS
Kernel Design

- The Kernel defines the inner product in RKHS
  - Any positive definite function (Gaussian, polynomial, Laplacian, etc.) can be used.
  - A *strictly positive definite* function will always yield universal mappers (Gaussian, Laplacian).
  - For infinite number of samples all spd kernels converge in the mean to the same solution.
  - For finite number of samples kernel function and free parameters matter.

See Sriperumbudur et al, *On the Relation Between Universality, Characteristic Kernels and RKHS Embedding of Measures, AISTATS 2010*

Sparsification

- Filter size increases linearly with samples!
- If RKHS is compact and the environment stationary, we see that there is no need to keep increasing the filter size.
- Issue is that we would like to implement it on-line!
- Two ways to cope with growth:
  - Novelty Criterion (NC)
  - Approximate Linear Dependency (ALD)
- NC is very simple and intuitive to implement.

Sparsification

Novelty Criterion (NC)

- Present dictionary is $C(i) = \{c_j\}_{j=1}^{m_i}$. When a new data pair arrives (**u**(i+1),d(i+1)).
- First compute the distance to the present dictionary

$$dis = \min_{c_j \in C} \left\| u(i+1) - c_j \right\|$$

- If smaller than threshold δ_1 do not create new center
- Otherwise check if the prediction error is larger than δ_2 to augment the dictionary.
- δ_1 ~ 0.1 kernel size and δ_2 ~ sqrt of MSE

Sparsification

Approximate Linear Dependency (ALD)

- Engel proposed to estimate the distance to the linear span of the centers, i.e. compute

$$dis = \min_{\vee b} \left\| \varphi(u(i+1)) - \sum_{c_j \in C} b_j \varphi(c_j) \right\|$$

Which can be estimated by

$$dis^2 = \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \mathbf{h}(i+1)^T \mathbf{G}^{-1}(i)\mathbf{h}(i+1)$$

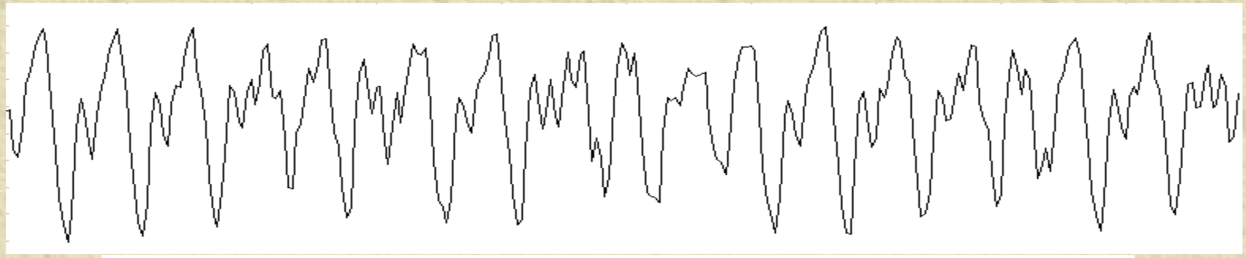
Only increase dictionary if *dis* larger than threshold

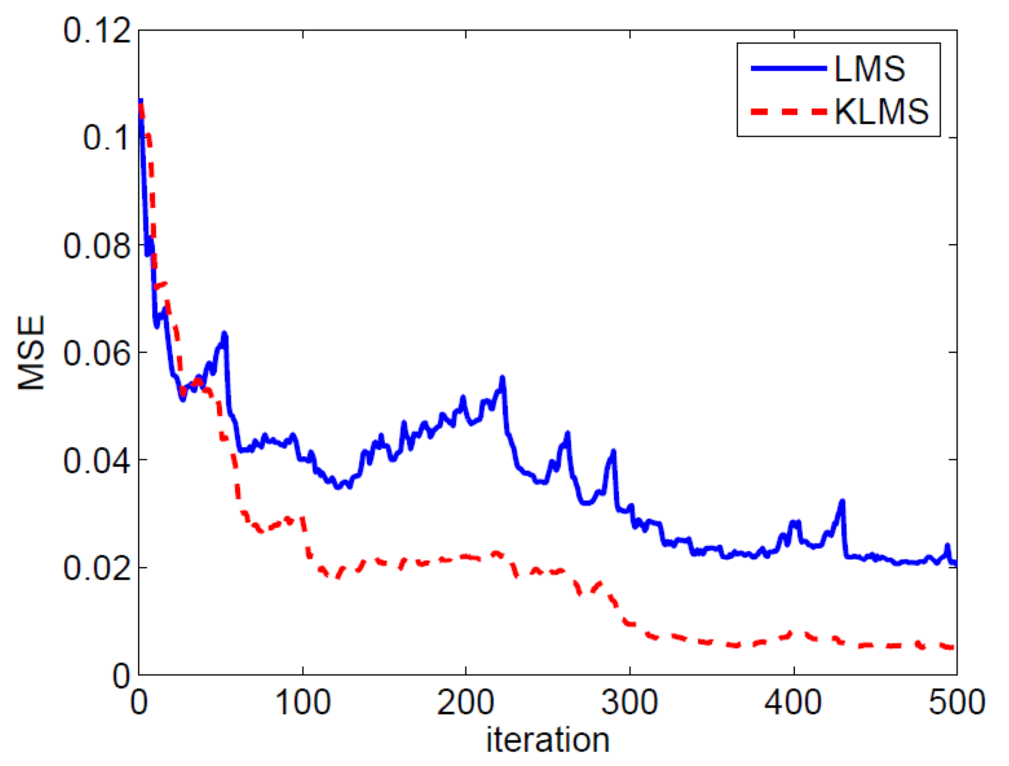
- Complexity is O(m^2)
- Easy to estimate in KRLS (*dis*~r(i+1))
- Can simplify the sum to the nearest center, and it defaults to NC

$$dis = \min_{\vee b, c_j \in C} \left\| \varphi(u(i+1)) - \varphi(c_j) \right\|$$

KLMS- Mackey-Glass Prediction

$$\dot{x}(t) = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}} \qquad \tau = 30$$





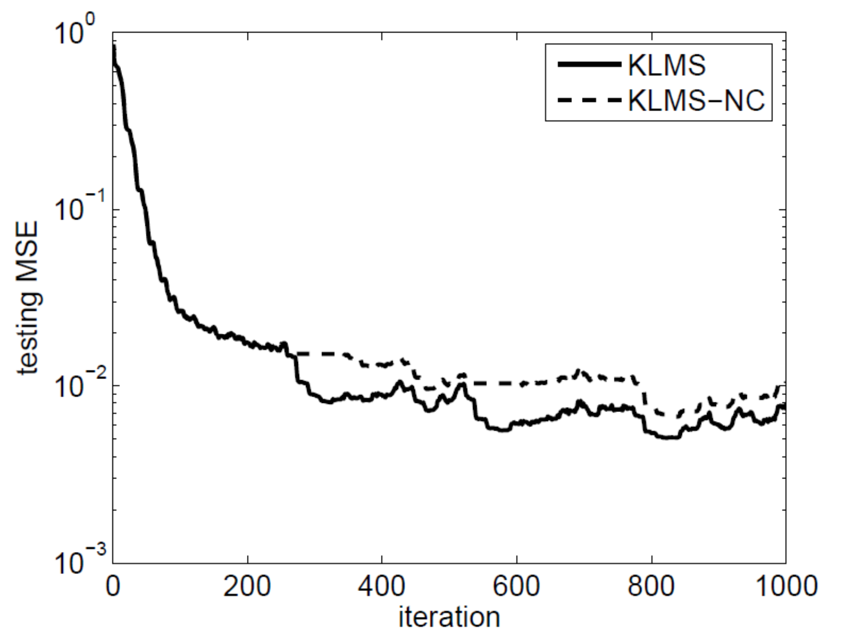
LMS

η=0.2

KLMS

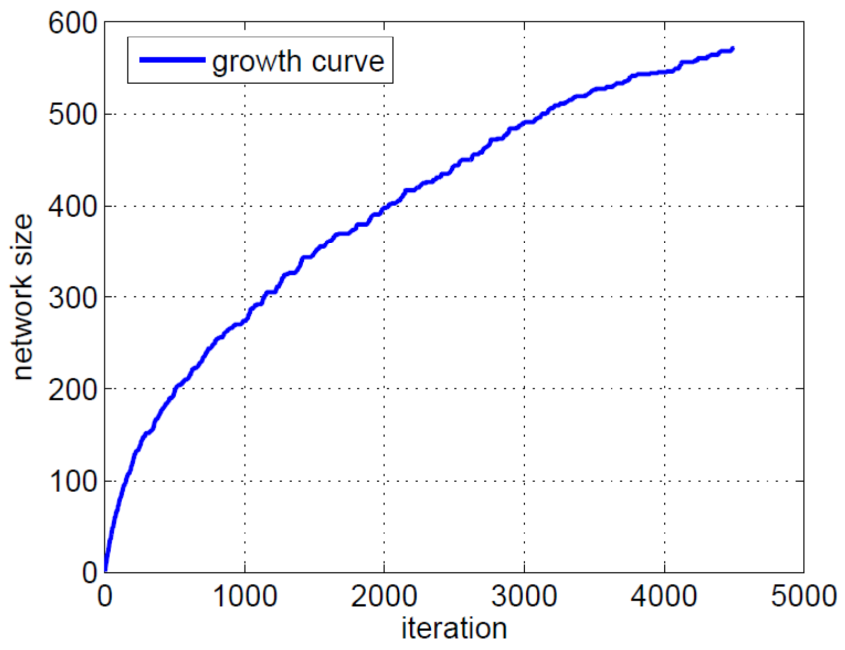
h=1, η=0.2

Performance Growth Trade-off



δ_1=0.1, δ_2=0.05

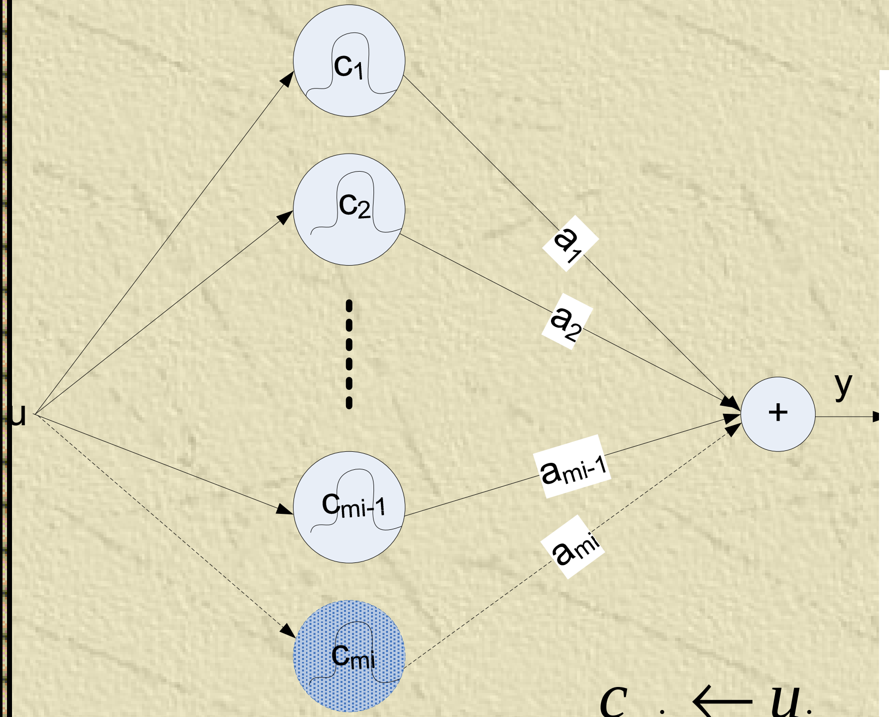
η=0.1, h=1

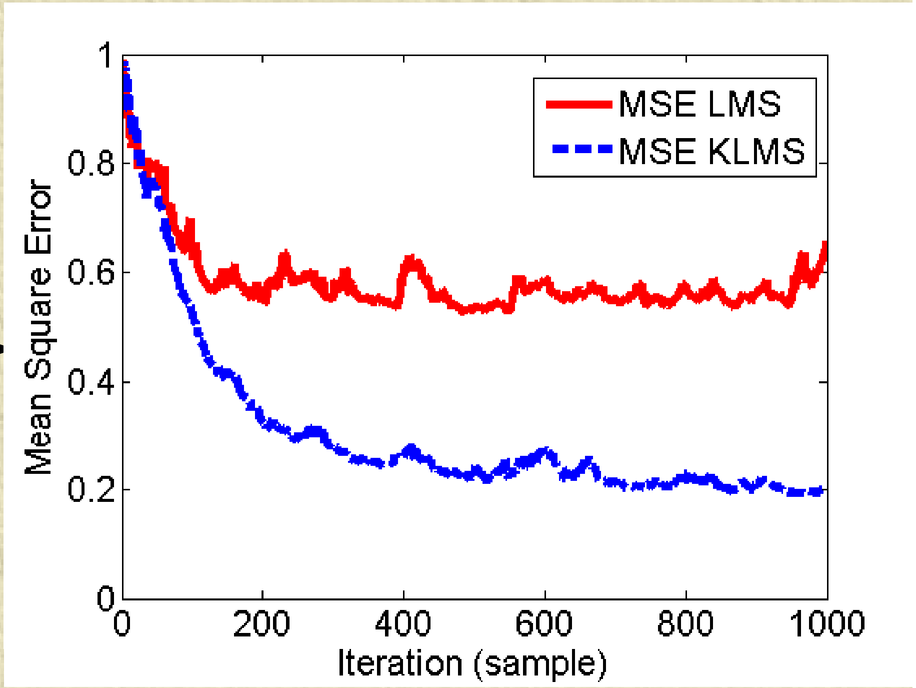


KLMS- Nonlinear Channel Equalization

$$f_i(u) = \langle \Omega_i, \varphi(u) \rangle_F = \sum_{j=1}^{i} \eta e(j) \kappa(u, u_j)$$

$$s_t \rightarrow \boxed{z_t = s_t + 0.5 s_{t-1}} \rightarrow \boxed{r_t = z_t - 0.9 z_t^2 + n_\sigma} \rightarrow r_t$$





$$c_{mi} \leftarrow u_i$$

$$a_{mi} \leftarrow \eta e(i)$$

Nonlinear Channel Equalization

Algorithms	Linear LMS (η=0.005)	KLMS (η=0.1) (NO REGULARIZATION)	RN (REGULARIZED λ=1)
BER (σ = .1)	0.162 ± 0.014	0.020 ± 0.012	0.008 ± 0.001
BER (σ = .4)	0.177 ± 0.012	0.058 ± 0.008	0.046 ± 0.003
BER (σ = .8)	0.218 ± 0.012	0.130 ± 0.010	0.118 ± 0.004

$$\kappa(u_i, u_j) = \exp(-0.1 \| u_i - u_j \|^2)$$

Algorithms	Linear LMS	KLMS	RN
Computation (training)	O(l)	O(i)	$O(i^3)$
Memory (training)	O(l)	O(i)	$O(i^2)$
Computation (test)	O(l)	O(i)	O(i)
Memory (test)	O(l)	O(i)	O(i)

Why don’t we need to explicitly regularize the KLMS?

Self-Regularization Property of KLMS

- Assume the data model $d(i) = \Omega^o(\varphi_i) + v(i)$ then for any unknown vector Ω^0 the following inequality holds

$$\frac{\sum_{j=1}^{i} |e(j) - v(j)|^2}{\eta^{-1} \|\Omega^o\|^2 + \sum_{j=1}^{i-1} |v(j)|^2} < 1, \quad for\ all\ i = 1, 2, \dots, N$$

as long as the matrix $\{\eta^{-1} I - \varphi(i)\varphi(i)^T\}$ is positive definite. So

- H^∞ robustness

$$\|\vec{e}\|^2 < \eta^{-1} \|\Omega^o\|^2 + 2\|\vec{v}\|^2$$

- And $\Omega(n)$ is upper bounded

$$\|\Omega_N\|^2 < \sigma_1 \eta (\|\Omega^o\|^2 + 2\eta \|\vec{v}\|^2)$$

σ_1 is the largest eigenvalue of $G\varphi$

The solution norm of KLMS is always upper bounded i.e. the algorithm is well posed in the sense of Hadamard.

Liu W., Pokarel P., Principe J., "The Kernel LMS Algorithm", IEEE Trans. Signal Processing, Vol 56, # 2, 543 – 554, 2008.

Intuition: KLMS and the Data Space

- KLMS search is insensitive to the 0-eigenvalue directions

$$E[\varepsilon_n(i)] = (1-\eta\varsigma_n)^i \varepsilon_n(0)$$

$$E[|\varepsilon_i(n)|^2] = \frac{\eta J_{\min}}{2-\eta\varsigma_n} + (1-\eta\varsigma_n)^{2i}\left(|\varepsilon_0(n)|^2 - \frac{\eta J_{\min}}{2-\eta\varsigma_n}\right)$$

So if $\varsigma_n = 0$, $E[\varepsilon_n(i)] = \varepsilon_n(0)$ and $E[|\varepsilon_n(i)|^2] = |\varepsilon_n(0)|^2$

- The 0-eigenvalue directions do not affect the MSE

$$J(i) = E[|d - \Omega_i^T \varphi|^2]$$

$$J(i) = J_{\min} + \frac{\eta J_{\min}}{2}\sum_{n=1}^{m}\varsigma_n + \sum_{n=1}^{m}\varsigma_n\left(|\varepsilon_n(0)|^2 - \frac{\eta J_{\min}}{2}\right)(1-\eta\varsigma_n)^{2i}$$

KLMS only finds solutions on the data subspace! It does not care about the null space!

Liu W., Pokarel P., Principe J., “The Kernel LMS Algorithm”, IEEE Trans. Signal Processing, Vol 56, # 2, 543 – 554, 2008.

Tikhonov Regularization

- In numerical analysis the methodology constrains the condition number of the solution matrix (or its eigenvalues)
- The singular value decomposition of Φ can be written

$$\mathbf{\Phi} = \mathbf{P}\begin{bmatrix} \mathbf{S} & 0 \\ 0 & 0 \end{bmatrix}\mathbf{Q}^T \qquad S = diag\{s_1, s_2, \ldots, s_r\}$$

Singular value

- The pseudo inverse to estimate Ω in $d(i) = \varphi(i)^T \Omega^0 + v(i)$ is

$$\Omega_{PI} = \mathbf{P} diag[s_1^{-1}, \ldots, s_r^{-1}, 0 \ldots 0]\mathbf{Q}^T \mathbf{d}$$

which can be still ill-posed (very small s_r). Tikhonov regularized the least square solution to penalize the solution norm to yield

$$J(\Omega) = \left\| \mathbf{d} - \mathbf{\Phi}^T \Omega \right\| + \lambda \left\| \Omega \right\|^2$$

$$\Omega = P diag(\frac{s_1}{s_1^2 + \lambda}, \ldots, \frac{s_r}{s_r^2 + \lambda}, 0, \ldots, 0) Q^T d$$

Notice that if λ = 0, when s_r is very small, $s_r/(s_r^2 + \lambda) = 1/s_r \to \infty$.

However if λ > 0, when s_r is very small, $s_r/(s_r^2 + \lambda) = s_r/\lambda \to 0$.

Tikhonov and KLMS

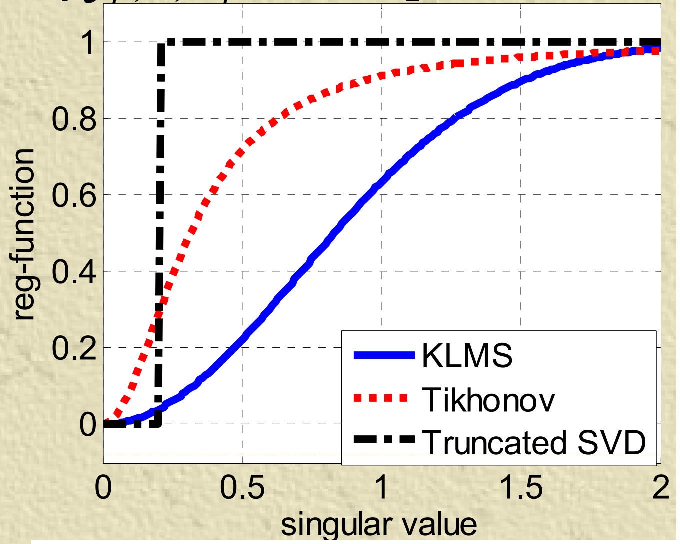
- In the worst case, substitute the optimal weight by the pseudo inverse

$$E[\Omega(i)] = \mathbf{P}diag[(1-(1-\eta\varsigma_1)^i)s_1^{-1},...,(1-(1-\eta\varsigma_r)^i)s_r^{-1},0....0]\mathbf{Q}^T\mathbf{d}$$

- No regularization yields s_n^{-1}
- Tikhonov

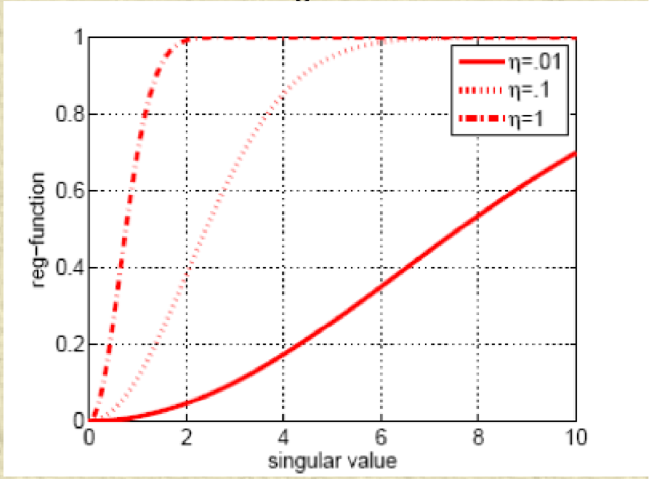
$$[s_n^2/(s_n^2+\lambda)]\cdot s_n^{-1}$$

- PCA $\begin{cases} s_n^{-1} & \text{if } s_n > \text{th} \\ 0 & \text{if } s_n \le \text{th} \end{cases}$



- Regularization function for finite N in KLMS

$$[1-(1-\eta s_n^2/N)^N]\cdot s_n^{-1}$$



The stepsize and N control the reg-function in KLMS.

Liu W., Principe J. The Well-posedness Analysis of the Kernel Adaline, Proc WCCI, Hong-Kong, 2008

Energy Conservation Relation

The fundamental energy conservation relation holds in RKHS!

- Energy conservation in RKHS

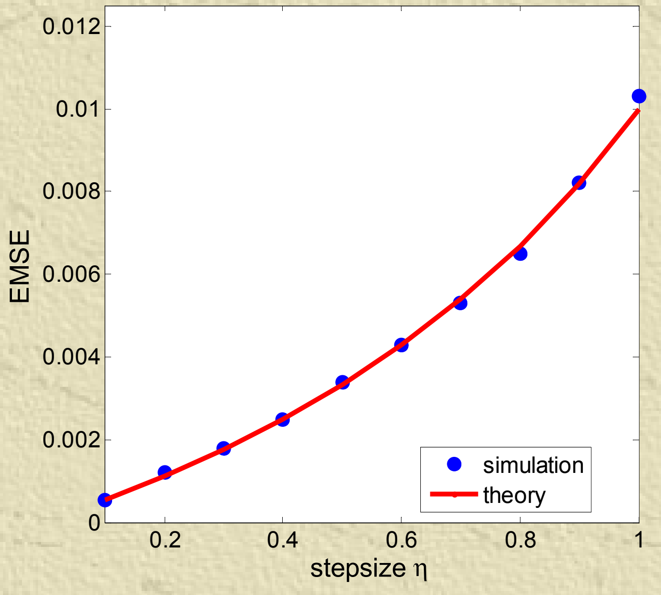
$$\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathrm{F}}^{2}+\frac{e_a^2(i)}{\kappa\left(\boldsymbol{u}(i),\boldsymbol{u}(i)\right)}=\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathrm{F}}^{2}+\frac{e_p^2(i)}{\kappa\left(\boldsymbol{u}(i),\boldsymbol{u}(i)\right)}$$

- Upper bound on step size for mean square convergence

$$\eta \le \frac{2E\left[\left\|\boldsymbol{\Omega}^*\right\|_{\mathrm{F}}^{2}\right]}{E\left[\left\|\boldsymbol{\Omega}^*\right\|_{\mathrm{F}}^{2}\right]+\sigma_v^2}$$

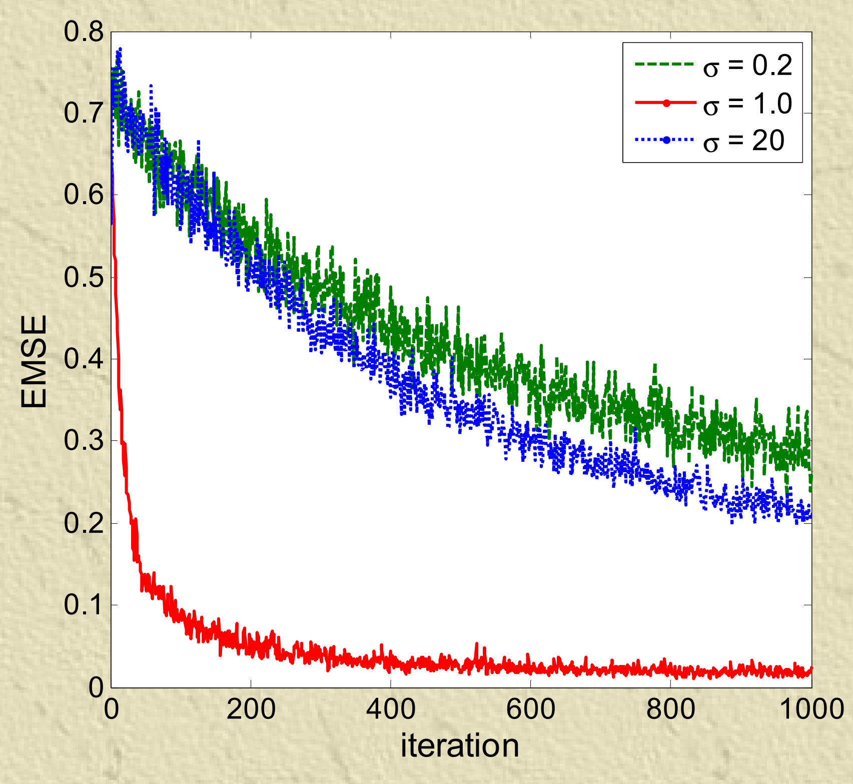
- Steady-state mean square performance

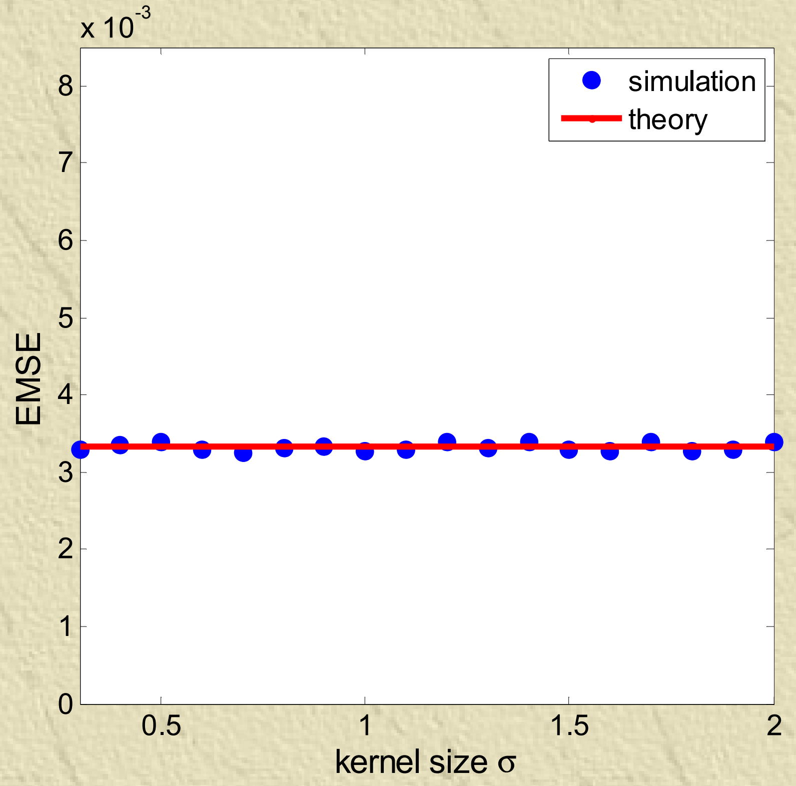
$$\lim_{i\to\infty} E\left[e_a^2(i)\right]=\frac{\eta\sigma_v^2}{2-\eta}$$



Chen B., Zhao S., Zhu P., Principe J. **Mean Square Convergence Analysis of the Kernel Least Mean Square Algorithm**, submitted to IEEE Trans. Signal Processing

Effects of Kernel Size

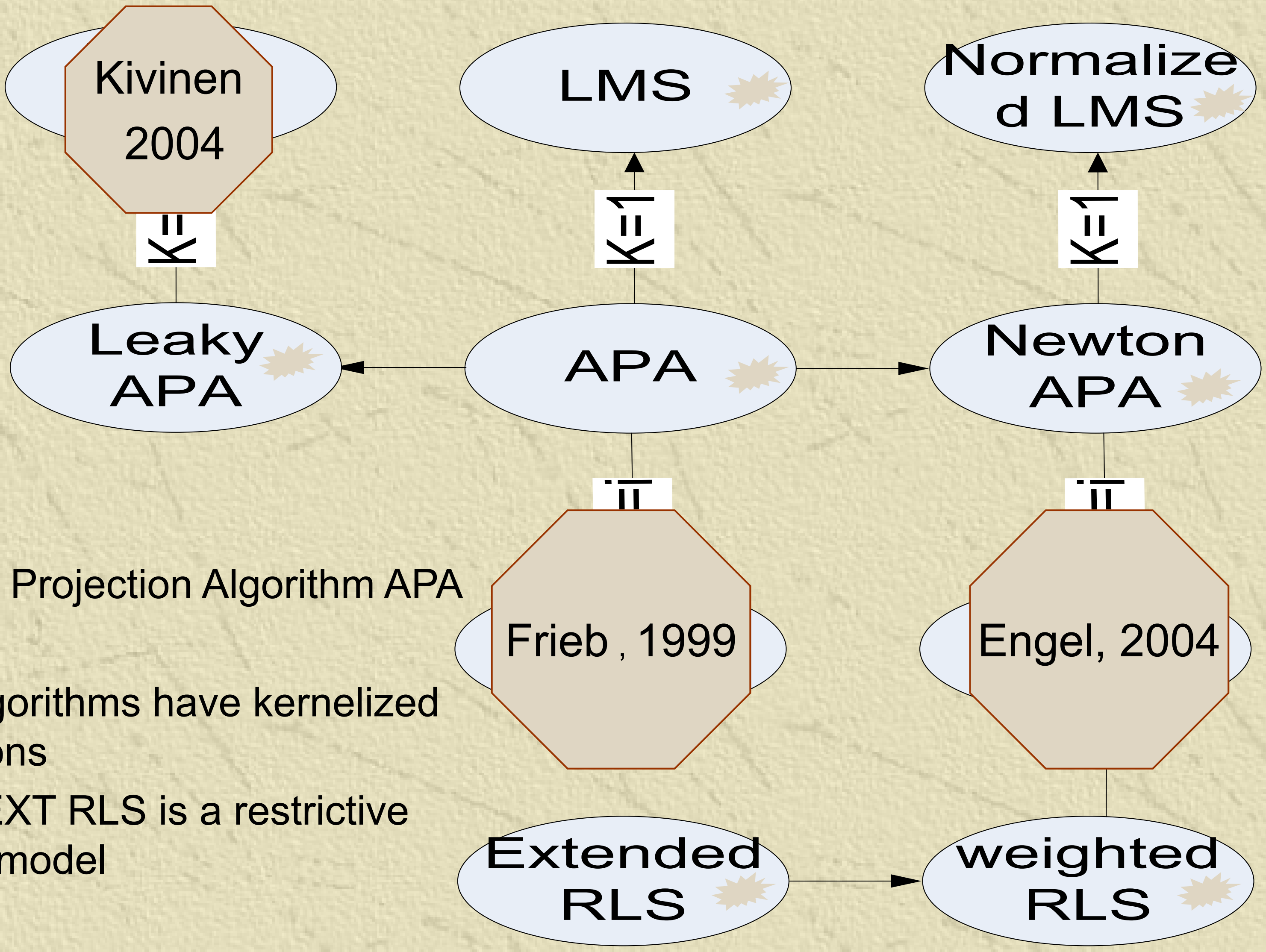




- Kernel size affects the convergence speed! (How to choose a suitable kernel size is still an open problem)

- However, it does not affect the final misadjustment! (universal approximation with infinite samples)

The Big Picture for Gradient Based Learning



Affine Projection Algorithm APA

All algorithms have kernelized versions

The EXT RLS is a restrictive state model

Liu W., Principe J., "Kernel Affine Projection Algorithms", European J. of Signal Processing, ID 784292, 2008.

Affine projection algorithms

- Solve $\min_{w} J(\mathbf{w}) = E\left|d - \mathbf{w}^T \mathbf{u}\right|^2$ which yields $\mathbf{w}^0 = \mathbf{R}_\mathbf{u}^{-1}\mathbf{r}_\mathbf{du}$

- There are several ways to approximate this solution iteratively using
  - Gradient Descent Method

    $$\mathbf{w}(0) \qquad \mathbf{w}(i) = \mathbf{w}(i-1) + \eta[\mathbf{r}_\mathbf{du} - \mathbf{R}_\mathbf{u}\mathbf{w}(i-1)]$$
  - Newton's recursion

    $$\mathbf{w}(0) \qquad \mathbf{w}(i) = \mathbf{w}(i-1) + \eta(\mathbf{R}_\mathbf{u} + \varepsilon\mathbf{I})^{-1}[\mathbf{r}_\mathbf{du} - \mathbf{R}_\mathbf{u}\mathbf{w}(i-1)]$$

- LMS uses a stochastic gradient that approximates

  $$\hat{\mathbf{R}}_\mathbf{u} = \mathbf{u}(i)\mathbf{u}(i)^T \qquad \hat{\mathbf{r}}_\mathbf{du} = d(i)\mathbf{u}(i)$$

- Affine projection algorithms (APA) utilize better approximations
- Therefore APA is a family of online gradient based algorithms of intermediate complexity between the LMS and RLS.

Affine projection algorithms

- APA are of the general form

$$\mathbf{U}(i)=[\mathbf{u}(i-K+1),\ldots,\mathbf{u}(i)]_{LxK} \qquad \mathbf{d}(i)=[d(i-K+1),\ldots,d(i)]^T$$

$$\hat{\mathbf{R}}_{\mathbf{u}}=\frac{1}{K}\mathbf{U}(i)\mathbf{U}(i)^T \qquad \hat{\mathbf{r}}_{\mathbf{du}}=\frac{1}{K}\mathbf{U}(i)\mathbf{d}(i)$$

Gradient $\mathbf{w}(0)$ $\quad \mathbf{w}(i)=\mathbf{w}(i-1)+\eta\mathbf{U}(i)[\mathbf{d}(i)-\mathbf{U}(i)^T\mathbf{w}(i-1)]$

Newton

$$\mathbf{w}(i)=\mathbf{w}(i-1)+\eta(\mathbf{U}(i)\mathbf{U}(i)^T+\varepsilon\mathbf{I})^{-1}\mathbf{U}(i)[\mathbf{d}(i)-\mathbf{U}(i)^T\mathbf{w}(i-1)]$$

- Notice that

$$(\mathbf{U}(i)\mathbf{U}(i)^T+\varepsilon\mathbf{I})^{-1}\mathbf{U}(i)=\mathbf{U}(i)(\mathbf{U}(i)^T\mathbf{U}(i)+\varepsilon\mathbf{I})^{-1}$$

- So

$$\mathbf{w}(i)=\mathbf{w}(i-1)+\eta\mathbf{U}(i)[\mathbf{U}(i)^T\mathbf{U}(i)+\varepsilon\mathbf{I}]^{-1}[\mathbf{d}(i)-\mathbf{U}(i)^T\mathbf{w}(i-1)]$$

Affine projection algorithms

- If a regularized cost function is preferred

$$\min_{w} J(\mathbf{w}) = E\left|d - \mathbf{w}^T\mathbf{u}\right|^2 + \lambda\|\mathbf{w}\|^2$$

- The gradient method becomes

$$\mathbf{w}(0) \qquad \mathbf{w}(i) = (1-\eta\lambda)\mathbf{w}(i-1) + \eta\mathbf{U}(i)[\mathbf{d}(i) - \mathbf{U}(i)^T\mathbf{w}(i-1)]$$

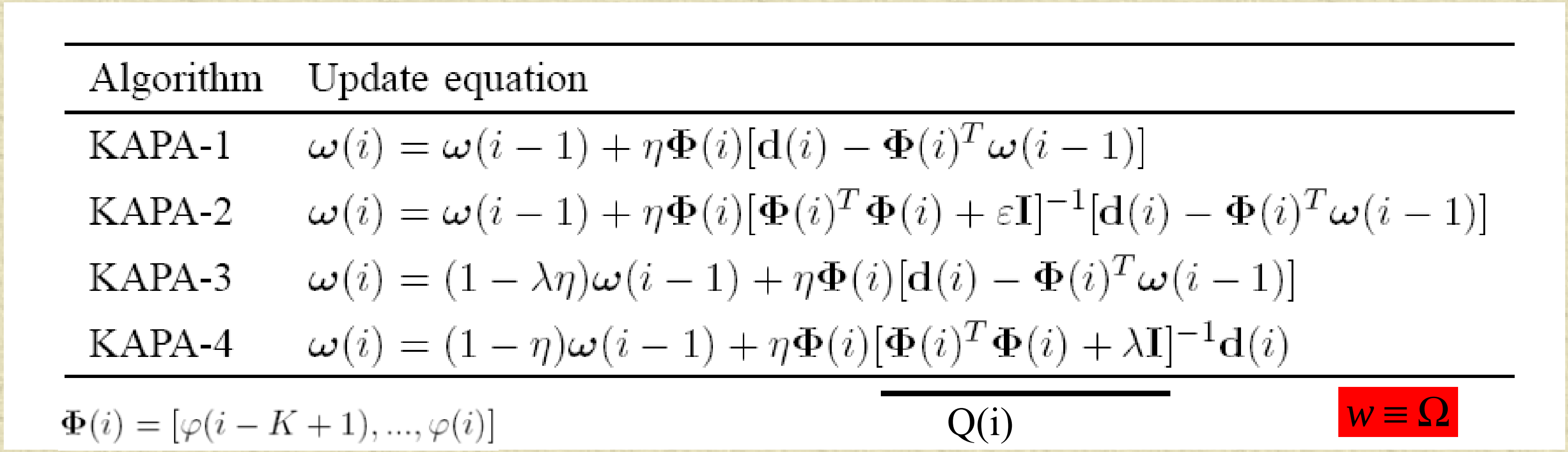
Newton

$$\mathbf{w}(i) = (1-\eta\lambda)\mathbf{w}(i-1) + \eta(\mathbf{U}(i)\mathbf{U}(i)^T + \varepsilon\mathbf{I})^{-1}\mathbf{U}(i)\mathbf{d}(i)$$

Or

$$\mathbf{w}(i) = (1-\eta\lambda)\mathbf{w}(i-1) + \eta\mathbf{U}(i)[\mathbf{U}(i)^T\mathbf{U}(i) + \varepsilon\mathbf{I}]^{-1}\mathbf{d}(i)$$

Kernel Affine Projection Algorithms



Algorithm	Update equation
KAPA-1	$\omega(i) = \omega(i-1) + \eta\Phi(i)[\mathbf{d}(i) - \Phi(i)^T\omega(i-1)]$
KAPA-2	$\omega(i) = \omega(i-1) + \eta\Phi(i)[\Phi(i)^T\Phi(i) + \varepsilon\mathbf{I}]^{-1}[\mathbf{d}(i) - \Phi(i)^T\omega(i-1)]$
KAPA-3	$\omega(i) = (1-\lambda\eta)\omega(i-1) + \eta\Phi(i)[\mathbf{d}(i) - \Phi(i)^T\omega(i-1)]$
KAPA-4	$\omega(i) = (1-\eta)\omega(i-1) + \eta\Phi(i)[\Phi(i)^T\Phi(i) + \lambda\mathbf{I}]^{-1}\mathbf{d}(i)$

$\Phi(i) = [\varphi(i-K+1), ..., \varphi(i)]$

$\overline{}$
Q(i)

$w \equiv \Omega$

KAPA 1,2 use the least squares cost, while KAPA 3,4 are regularized

KAPA 1,3 use gradient descent and KAPA 2,4 use Newton update

Note that KAPA 4 does not require the calculation of the error by rewriting the error with the matrix inversion lemma and using the kernel trick

Note that one does not have access to the weights, so need recursion as in KLMS.

Care must be taken to minimize computations.

Recursive Least-Squares

- The RLS algorithm estimates a weight vector **w**(i-1) by minimizing the cost function

$$\min_{w} \sum_{j=1}^{i-1} \left| d(j) - \mathbf{u}(j)^T \mathbf{w} \right|^2$$

- The solution becomes

$$\mathbf{w}(i-1) = (\mathbf{U}(i-1)\mathbf{U}(i-1)^T)^{-1}\mathbf{U}(i-1)\mathbf{d}(i-1)$$

And can be recursively computed as

$$\mathbf{w}(i) = \mathbf{w}(i-1) + \frac{\mathbf{P}(i-1)\mathbf{u}(i)}{1+\mathbf{u}(i)^T\mathbf{P}(i-1)\mathbf{u}(i)}[d(i) - \mathbf{u}(i)^T\mathbf{w}(i-1)]$$

Where $\mathbf{P}(i) = (\mathbf{U}(i)\mathbf{U}(i)^T)^{-1}$. Start with zero weights and $\mathbf{P}(0) = \lambda^{-1} I$

$$r(i) = 1 + \mathbf{u}(i)^T\mathbf{P}(i-1)\mathbf{u}(i) \qquad \mathbf{w}(i) = \mathbf{w}(i-1) + \mathbf{k}(i)e(i)$$

$$\mathbf{k}(i) = \mathbf{P}(i-1)\mathbf{u}(i)/r(i) \qquad \mathbf{P}(i) = [\mathbf{P}(i-1) - \mathbf{k}(i)\mathbf{k}(i)^T r(i)]$$

$$e(i) = d(i) - \mathbf{u}(i)^T\mathbf{w}(i-1)$$

Kernel Recursive Least-Squares

- The KRLS algorithm estimates a weight function **w**(i) by minimizing

$$\min_{w} \sum_{j=1}^{i-1} \left| d(j) - \mathbf{w}^T \varphi(j) \right|^2 + \lambda \|\mathbf{w}\|^2$$

  - The solution in RKHS becomes

$$\mathbf{w}(i) = \Phi(i)\left[\lambda I + \Phi(i)^T \Phi(i)\right]^{-1} \mathbf{d}(i) = \Phi(i)\mathbf{a}(i) \qquad \mathbf{a}(i) = \mathbf{Q}(i)\mathbf{d}(i)$$

$\mathbf{Q}^{-1}(i)$ can be computed recursively as

$$\mathbf{Q}^{-1}(i) = \begin{bmatrix} \mathbf{Q}(i-1)^{-1} & \mathbf{h}(i) \\ \mathbf{h}(i)^T & \lambda + \varphi(i))^T \varphi(i) \end{bmatrix} \qquad \mathbf{h}(i) = \Phi(i-1)^T \varphi(i)$$

From this we can also recursively compute Q(i)

$$\mathbf{Q}(i) = r(i)^{-1} \begin{bmatrix} \mathbf{Q}(i-1)r(i) + \mathbf{z}(i)^T \mathbf{z}(i) & -\mathbf{z}(i) \\ -\mathbf{z}(i)^T & 1 \end{bmatrix} \qquad \begin{array}{c} \mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{h}(i) \\ r(i) = \lambda + \kappa(\mathbf{u}(i), \mathbf{u}(i)) - \mathbf{z}(i)^T \mathbf{h}(i) \end{array}$$

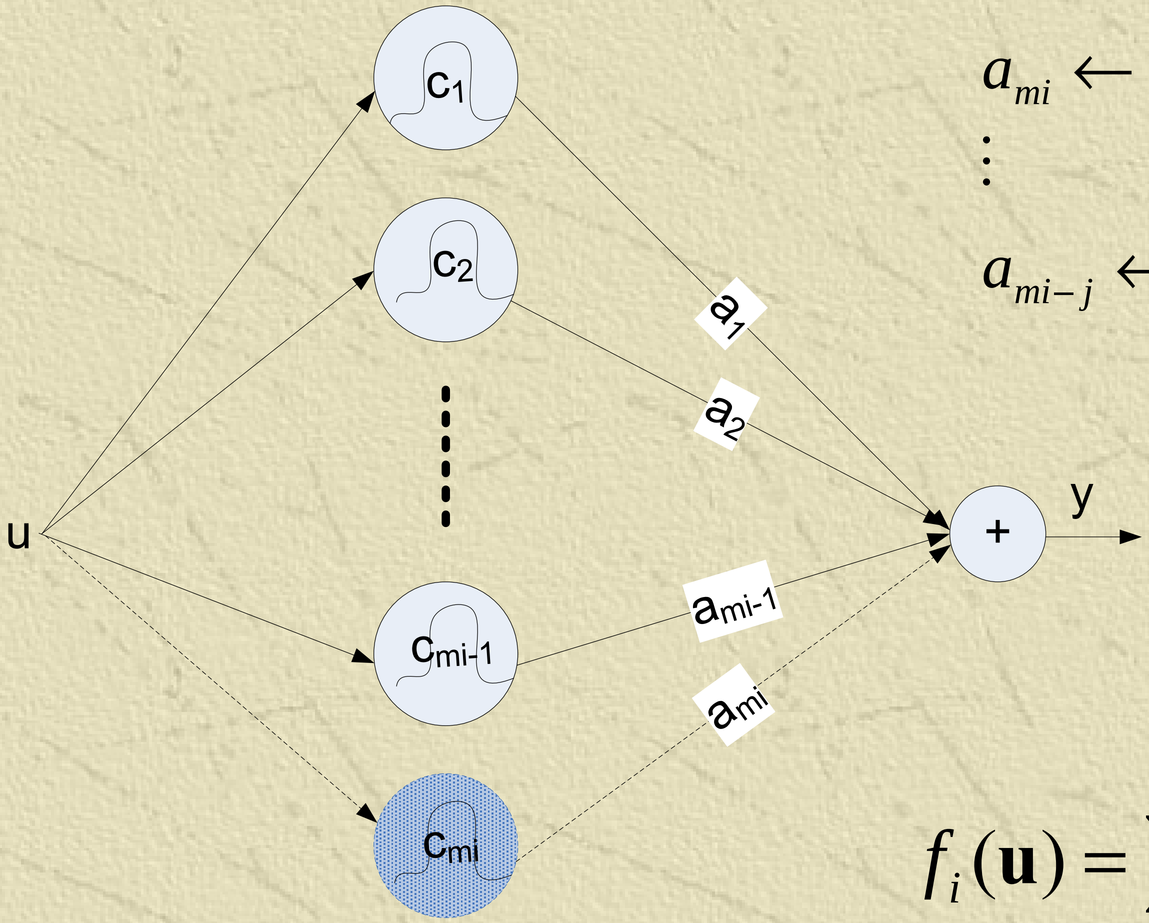
And compose back a(i) recursively

$$\mathbf{a}(i) = \begin{bmatrix} \mathbf{a}(i) - \mathbf{z}(i) r^{-1}(i) e(i) \\ r^{-1}(i) e(i) \end{bmatrix} \qquad e(i) = d(i) - \mathbf{h}(i)^T \mathbf{a}(i-1)$$

with initial conditions

$$\mathbf{Q}(1) = \left[\lambda + \kappa(\mathbf{u}(i), \mathbf{u}(i)^T)\right]^{-1}, \qquad \mathbf{a}(1) = \mathbf{Q}(1) d(1)$$

KRLS



$$c_{mi} \leftarrow u_i$$

$$a_{mi} \leftarrow r(i)^{-1} e(i)$$

$$\vdots$$

$$a_{mi-j} \leftarrow a_{mi-j} - r(i)^{-1} e(i) \mathbf{z}_j(i)$$

$$f_i(\mathbf{u}) = \sum_{j=1}^{i} \mathbf{a}(i) \kappa(\mathbf{u}(j), \mathbf{u})$$

Engel Y., Mannor S., Meir R. "The kernel recursive least square algorithm", IEEE Trans. Signal Processing, 52 (8), 2275-2285, 2004.

KRLS

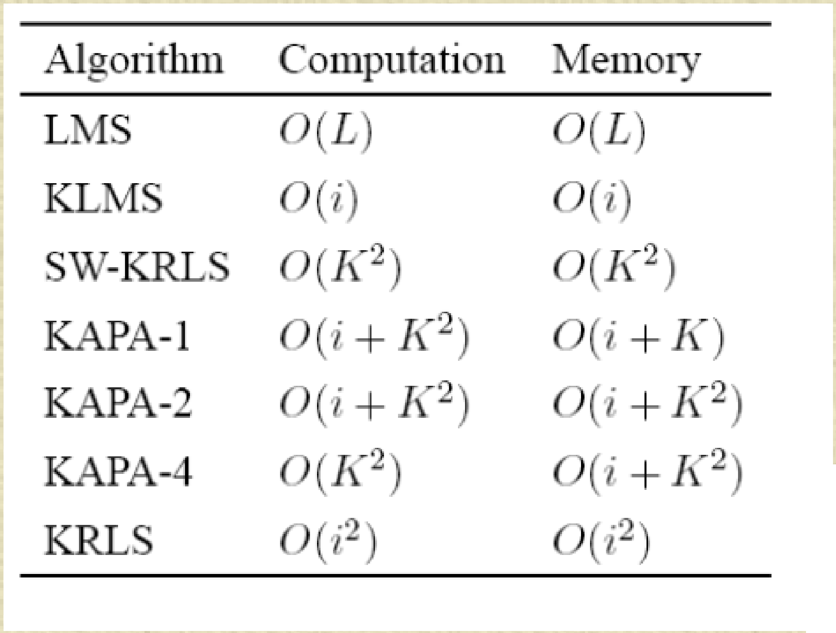
$$f_i = f_{i-1} + r(i)^{-1}\left[\kappa(\mathbf{u}(i),\cdot) - \sum_{j=1}^{i-1}\mathbf{z}_j(i)\kappa(\mathbf{u}(j),\cdot)\right]e(i)$$

$$\mathbf{a}_i(i) = r(i)^{-1}e(i)$$

$$\mathbf{a}_j(i) = \mathbf{a}_j(i) - r(i)^{-1}e(i)\mathbf{z}_j(i) \qquad j = 1,\ldots,i-1$$

$$C(i) = \{C(i-1), u(i)\}$$

Computation complexity



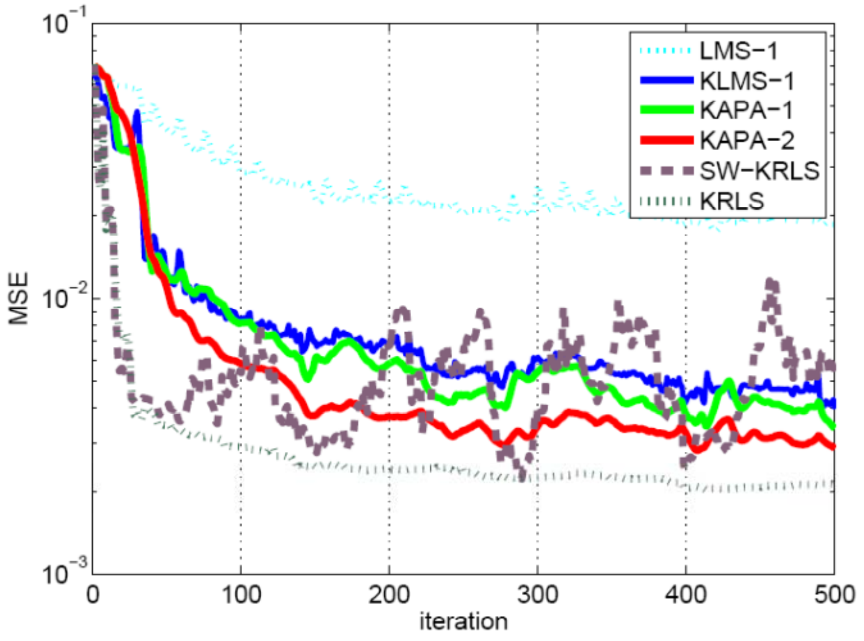
Algorithm	Computation	Memory
LMS	$O(L)$	$O(L)$
KLMS	$O(i)$	$O(i)$
SW-KRLS	$O(K^2)$	$O(K^2)$
KAPA-1	$O(i+K^2)$	$O(i+K)$
KAPA-2	$O(i+K^2)$	$O(i+K^2)$
KAPA-4	$O(K^2)$	$O(i+K^2)$
KRLS	$O(i^2)$	$O(i^2)$

Prediction of Mackey-Glass

L=10

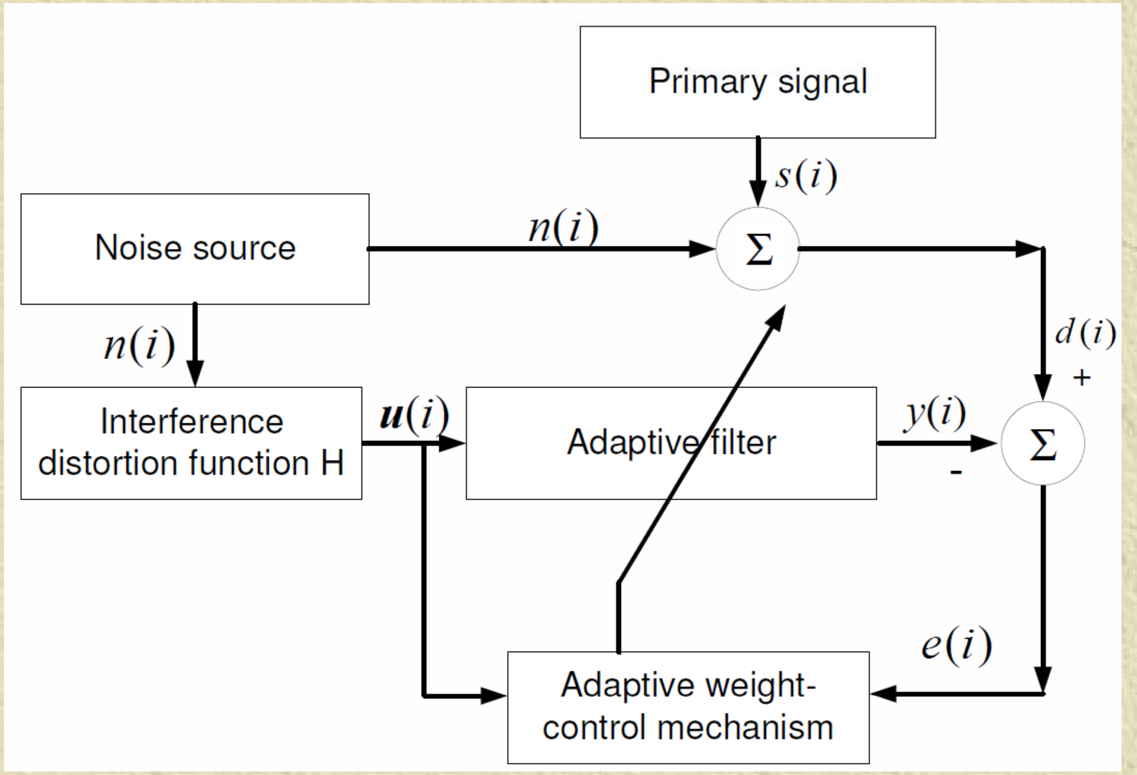
K=10

K=50 SW KRLS



Simulation 1: Noise Cancellation

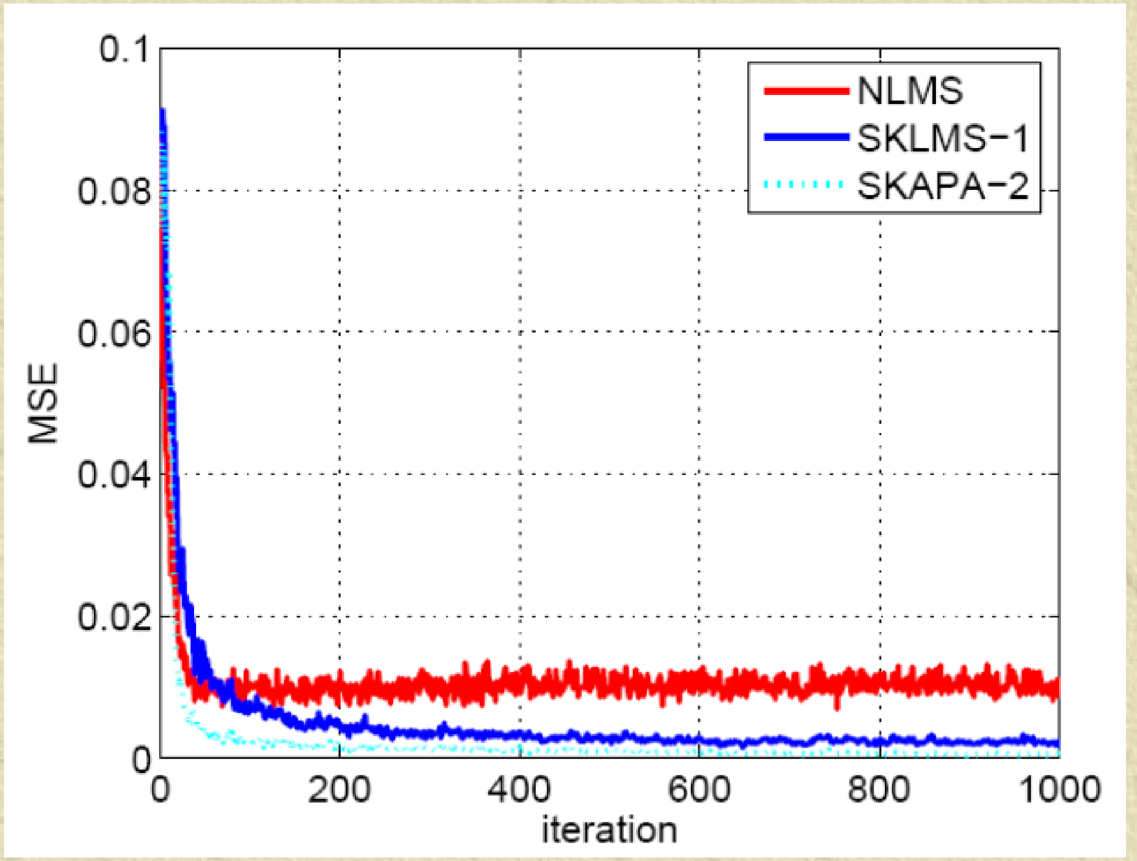
n(i) ~ uniform [-0.5, 05]

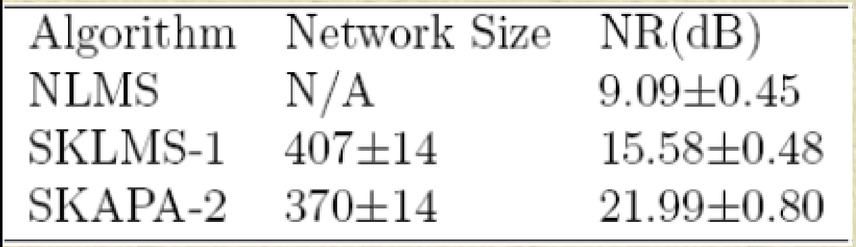


$$u(i) = n(i) - 0.2u(i-1) - u(i-1)n(i-1) + 0.1n(i-1) + 0.4u(i-2)$$

$$= H(n(i), n(i-1), u(i-1), u(i-2))$$

Simulation 1: Noise Cancellation



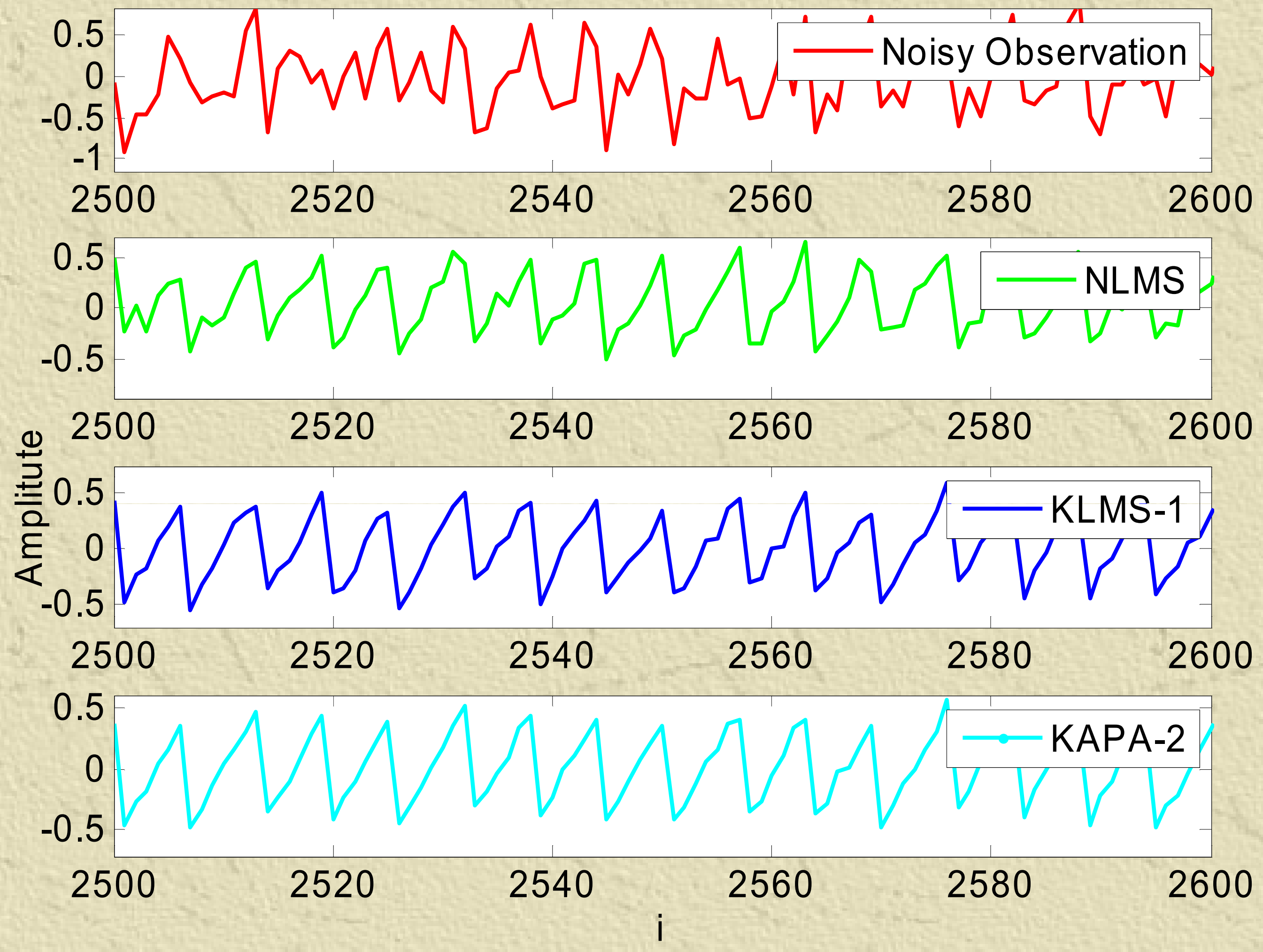


Algorithm	Network Size	NR(dB)
NLMS	N/A	9.09±0.45
SKLMS-1	407±14	15.58±0.48
SKAPA-2	370±14	21.99±0.80

$$\kappa(u(i),u(j)) = \exp(-\| u(i) - u(j) \|^2)$$

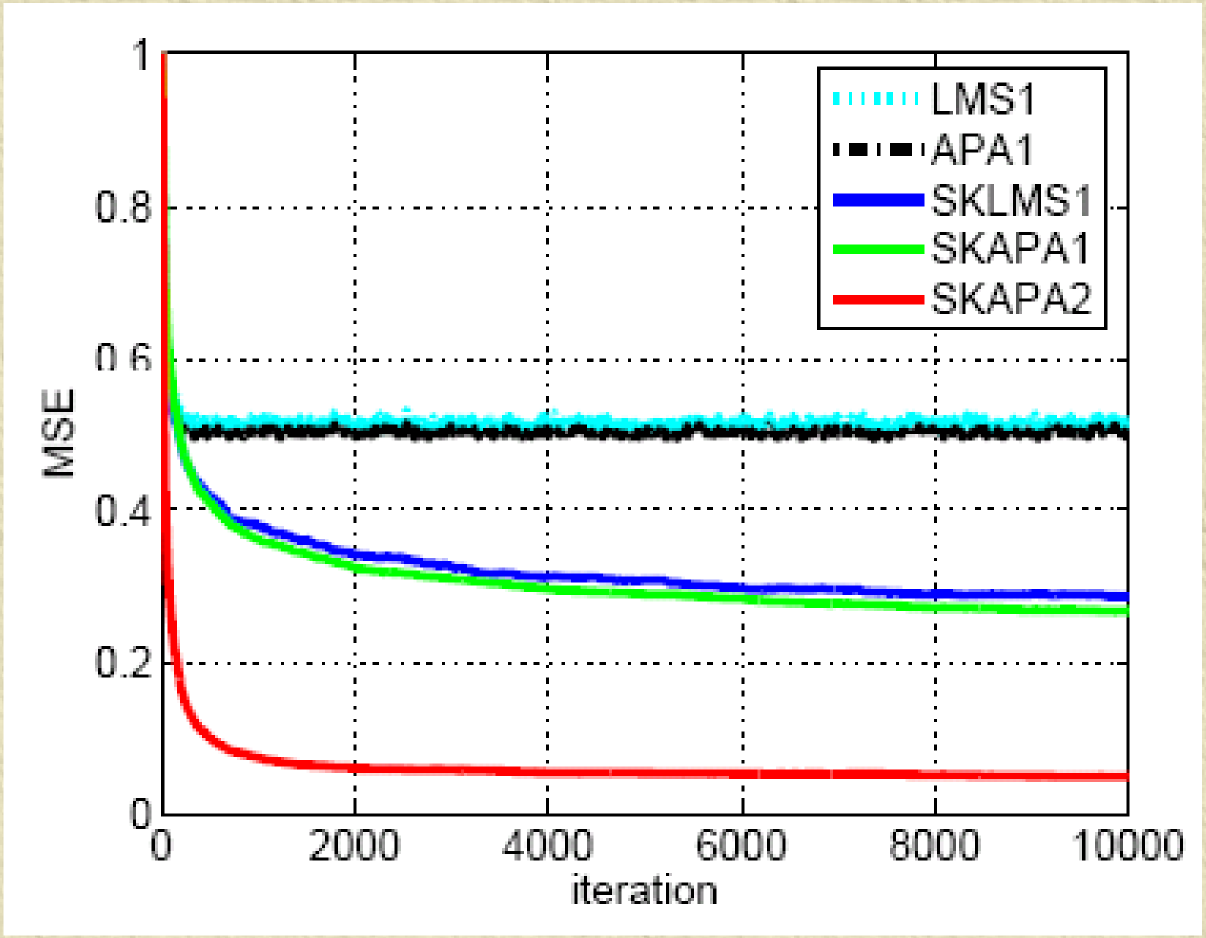
K=10

Simulation 1:Noise Cancellation



Simulation-2: nonlinear channel equalization

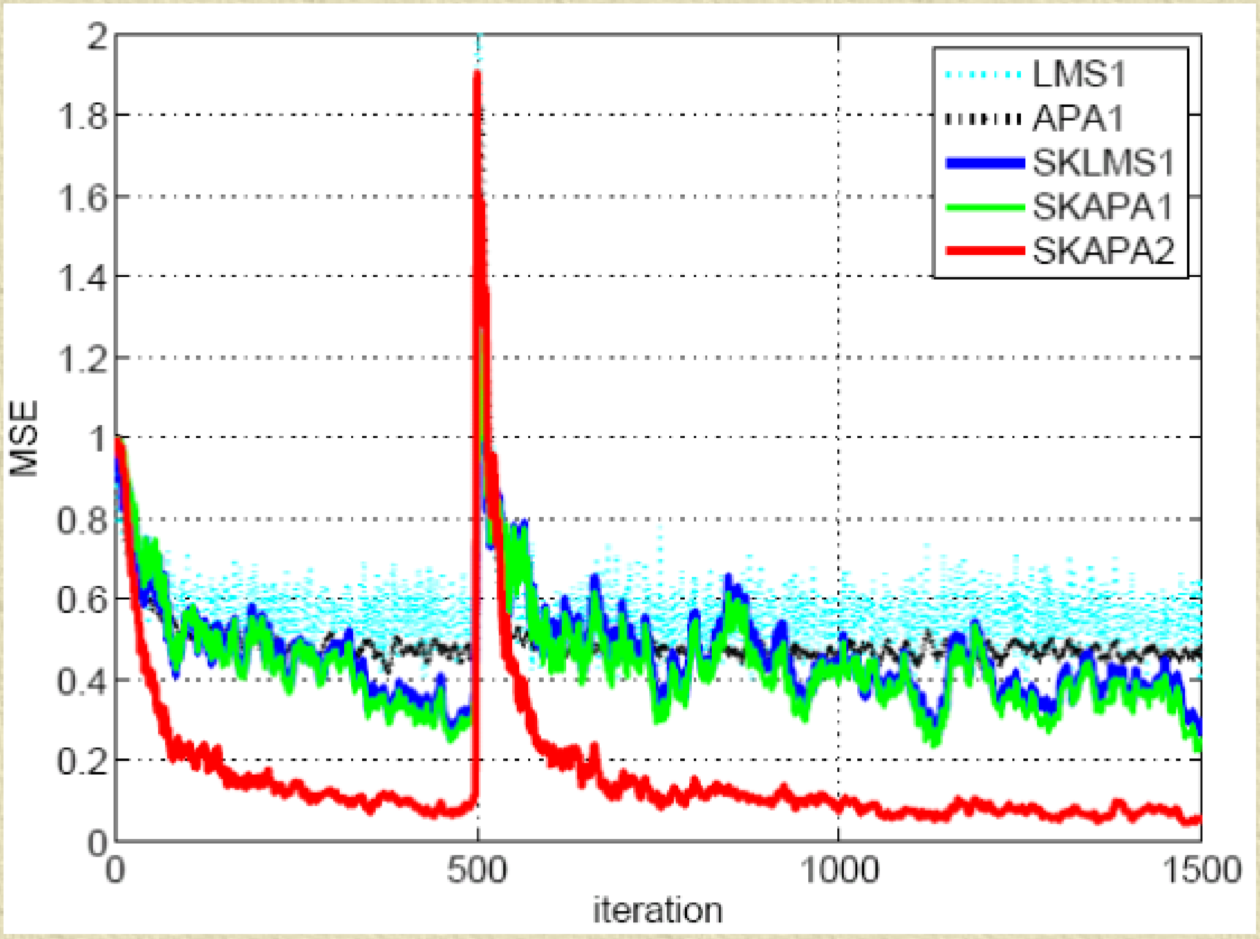
$$s_t \longrightarrow \boxed{z_t = s_t + 0.5 s_{t-1}} \longrightarrow \boxed{r_t = z_t - 0.9 z_t^2 + n_\sigma} \longrightarrow r_t$$



K=10

σ=0.1

Simulation-2: nonlinear channel equalization



$\uparrow$

Nonlinearity changed (inverted signs)

Active Data Selection

- Is the Kernel trick a “free lunch”?
  - The price we pay is large memory to store centers
  - Pointwise evaluations of the function
- But remember we are working on an on-line scenario, so most of the methods out there need to be modified.

Active Data Selection

- The goal is to build a constant length (fixed budget) filter in RKHS. There are two complementary methods of achieving this goal:
  - Discard unimportant centers (pruning)
  - Accept only some of the new centers (sparsification)
- Apart from heuristics, in either case a methodology to evaluate the importance of the centers for the overall nonlinear function approximation is needed.
- Another requirement is that this evaluation should be no more expensive computationally than the filter adaptation.

Previous Approaches – Sparsification

- Novelty condition (Platt, 1991)
  - Compute the distance to the current dictionary

$$dis = \min_{c_j \in D(i)} \left\| u(i+1) - c_j \right\|$$

  - If it is less than a threshold δ_1 discard
  - If the prediction error

$$e(i+1) = d(i+1) - \varphi(i+1)^T \Omega(i)$$

  - Is larger than another threshold δ_2 include new center.
- Approximate linear dependency (Engel, 2004)
  - If the new input is a linear combination of the previous centers discard

$$dis_2 = \min \left\| \varphi(u(i+1) - \sum_{c_j \in D(i)} b_j \varphi(c_j) \right\|$$

which is the Schur Complement of Gram matrix and fits KAPA 2 and 4 very well. Problem is computational complexity

Previous Approaches – Pruning

- Sliding Window (Vaerenbergh, 2010)
  - Impose $m_i<B$ in $f_i = \sum_{j=1}^{m_i} a_j(i)\kappa(c_j,.)$
  - Create the Gram matrix of size B+1 recursively from size B

$$\breve{G}(i+1) = \begin{bmatrix} G(i) & h \\ h^T & \kappa(c_{B+1}, c_{B+1}) \end{bmatrix} \qquad h = [\kappa(c_{B+1}, c_1), \ldots, \kappa(c_{B+1}, c_B)]^T$$

$$Q(i) = (\lambda I + G(i))^{-1} \qquad z = Q(i)h \qquad r = \lambda + \kappa(c_{B+1}, c_{B+1}) - z^T h$$

$$\breve{Q}(i+1) = \begin{bmatrix} Q(i) + zz^T / r & -z/r \\ -z^T / r & 1/r \end{bmatrix}$$

  - Downsize: reorder centers and include last (see KAPA2)

$$Q(i+1) = H - ff^T / e \quad a(i+1) = Q(i+1)d(i+1) \qquad f_{i+1} = \sum_{j=1}^{B} a_j(i+1)\kappa(c_j,.)$$

  - See also the Forgetron and the Projectron that provide error bounds for the approximation.

O. Dekel, S. Shalev-Shwartz, and Y. Singer, “The Forgetron: A kernel-based perceptron on a fixed budget,” in *Advances in Neural Information Processing Systems 18. Cambridge, MA: MIT Press, 2006, pp. 1342–1372.*

F. Orabona, J. Keshet, and B. Caputo, “Bounded kernel-based online learning,” *Journal of Machine Learning Research,* vol. 10, pp. 2643–2666, 2009.

Information Theoretic Statement

- The learning system $y(u;T(i))$
  - Already processed (the dictionary)

$$D(i) = \{u(j), d(j)\}_{j=1}^{i}$$

- A new data pair $\{u(i+1), d(i+1)\}$
  - How much new information it contains?
- Is this the right question? NO

How much information it contains with respect to the learning system $y(u;T(i))$?

Information Measure

- Hartley and Shannon's definition of information
  - How much information it contains?

$$I(i+1) = -\ln p(u(i+1), d(i+1))$$

- Learning is unlike digital communications:

  The machine never knows the joint distribution!
- When the same message is presented to a learning system information (the degree of uncertainty) changes because the system learned with the first presentation!
- Need to bring back MEANING into information theory!

Surprise as an Information Measure

- Learning is very much like an experiment that we do in the laboratory.
- Fedorov (1972) proposed to measure the importance of an experiment as the Kulback Leibler distance between the prior (the hypothesis we have) and the posterior (the results after measurement).
- Mackay (1992) formulated this concept under a Bayesian approach and it has become one of the key concepts in active learning.

Surprise as an Information Measure

- Pfaffelhuber in 1972 formulated the concept of subjective or redundant information for learning systems as

$$I_S(x) = -\log(q(x))$$

the PDF of the data is $p(x)$ and $q(x)$ is the learner's subjective estimation of it.

- Palm in 1981 defined surprise (or conditional information) for a learning system $y(u;T(i))$ as

$$S_{T(i)}(u(i+1)) = CI(i+1) = -\ln p(u(i+1) \mid T(i))$$

Shannon versus Surprise

Shannon (absolute information)	**Surprise (conditional information)**
Objective	Subjective
Receptor independent	Receptor dependent (on time and agent)
Message is meaningless	Message has meaning for the agent

Evaluation of Conditional Information (surprise)

- Gaussian process theory

$$CI(i+1) = -\ln[p(\mathbf{u}(i+1), d(i+1) \mid T(i))] =$$

$$\ln\sqrt{2\pi} + \ln\sigma(i+1) + \frac{(d(i+1)-\hat{d}(i+1))^2}{2\sigma^2(i+1)} - \ln[p(\mathbf{u}(i+1) \mid T(i))]$$

- where

$$\hat{d}(i+1) = \mathbf{h}(i+1)^T[\sigma_n^2\mathbf{I} + \mathbf{G}(i)]^{-1} d(i)$$

$$\sigma^2(i+1) = \sigma_n^2 + \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \mathbf{h}(i+1)^T[\sigma_n^2\mathbf{I} + \mathbf{G}(i)]^{-1}\mathbf{h}(i+1)$$

Interpretation of Conditional Information (surprise)

$$CI(i+1) = -\ln[p(\mathbf{u}(i+1), d(i+1) \mid T(i))] =$$

$$\ln\sqrt{2\pi} + \ln\sigma(i+1) + \frac{(d(i+1)-\hat{d}(i+1))^2}{2\sigma^2(i+1)} - \ln[p(\mathbf{u}(i+1) \mid T(i))]$$

- Prediction error $e(i+1) = d(i+1) - \hat{d}(i+1)$
  - Large error → large conditional information
- Prediction variance $\sigma^2(i+1)$
  - Small error, large variance → large CI
  - Large error, small variance → large CI (abnormal)
- Input distribution $p(\mathbf{u}(i+1) \mid T(i))$
  - Rare occurrence → large CI

Redundant, abnormal and learnable

$$Abnormal: \quad S(i+1) > T_1$$

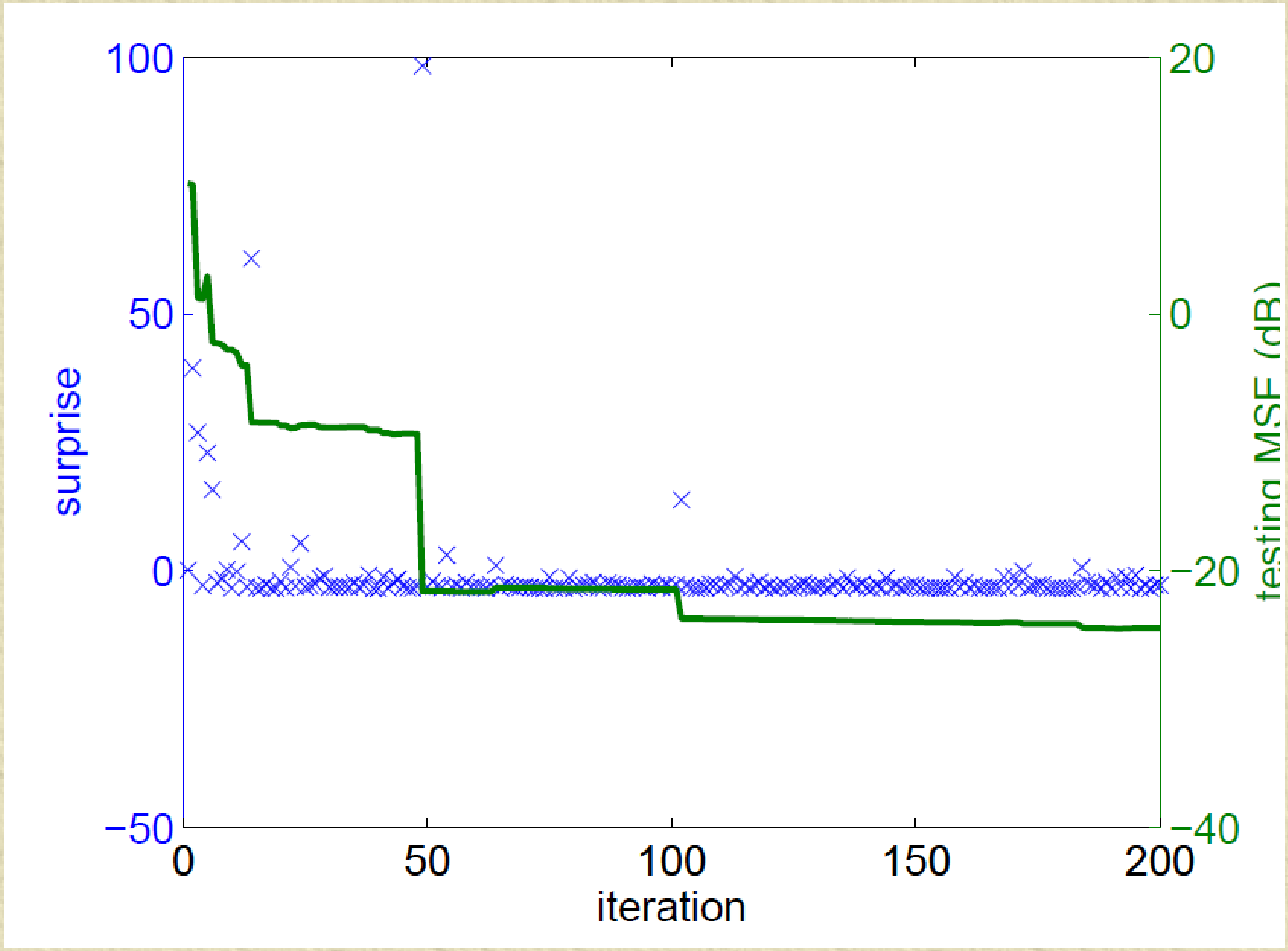
$$Learnable: \quad T_1 \geq S(i+1) \geq T_2$$

$$\mathrm{Re}\,dundant: \quad S(i+1) < T_2$$

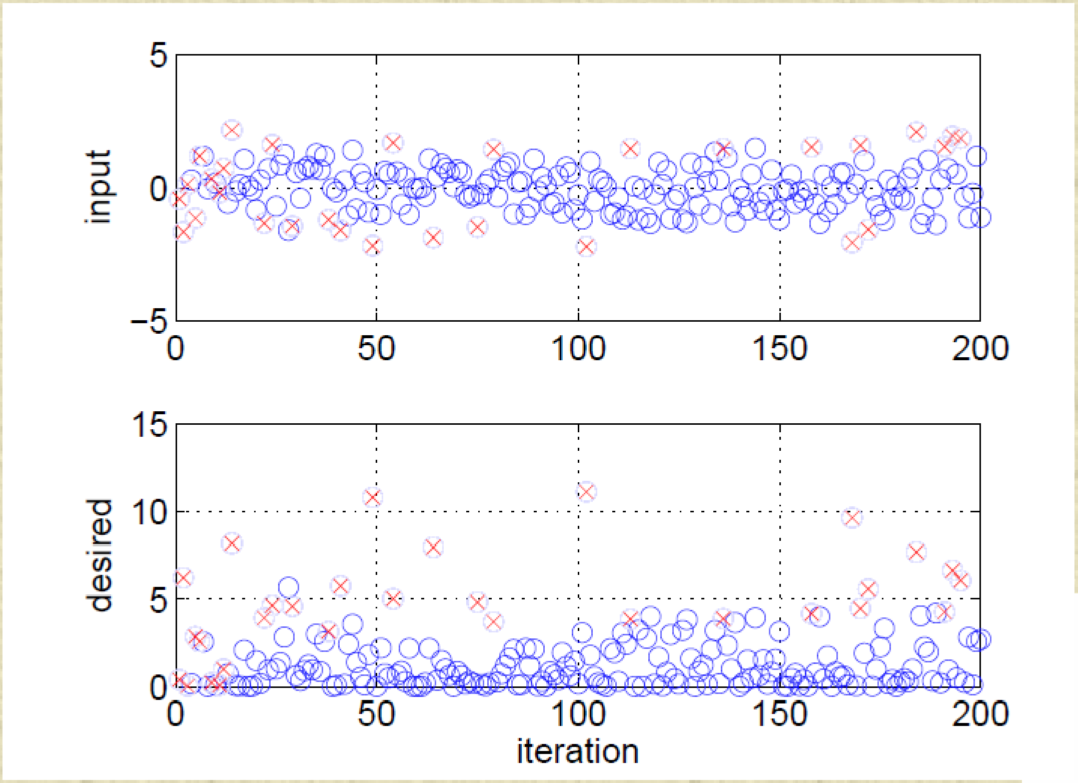
- Still need to find a systematic way to select these thresholds which are hyperparameters.

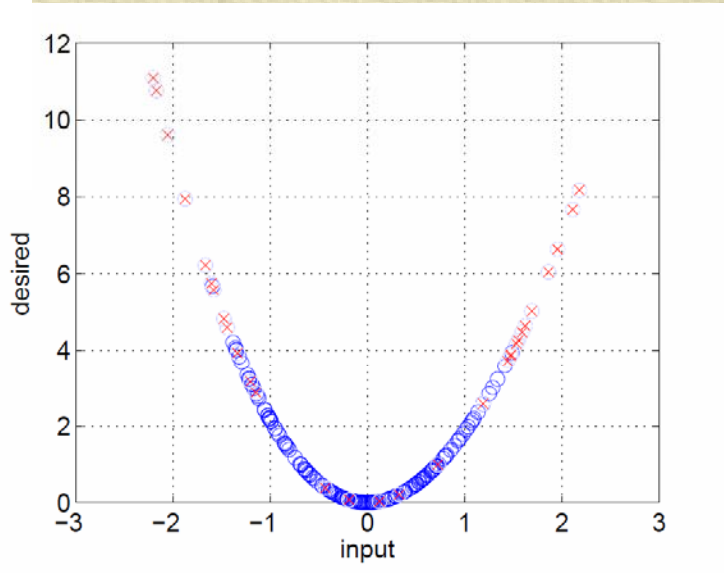
Simulation-5: KRLS-SC nonlinear regression

- Nonlinear mapping is $y=-x+2x^2+\sin x$ in unit variance Gaussian noise

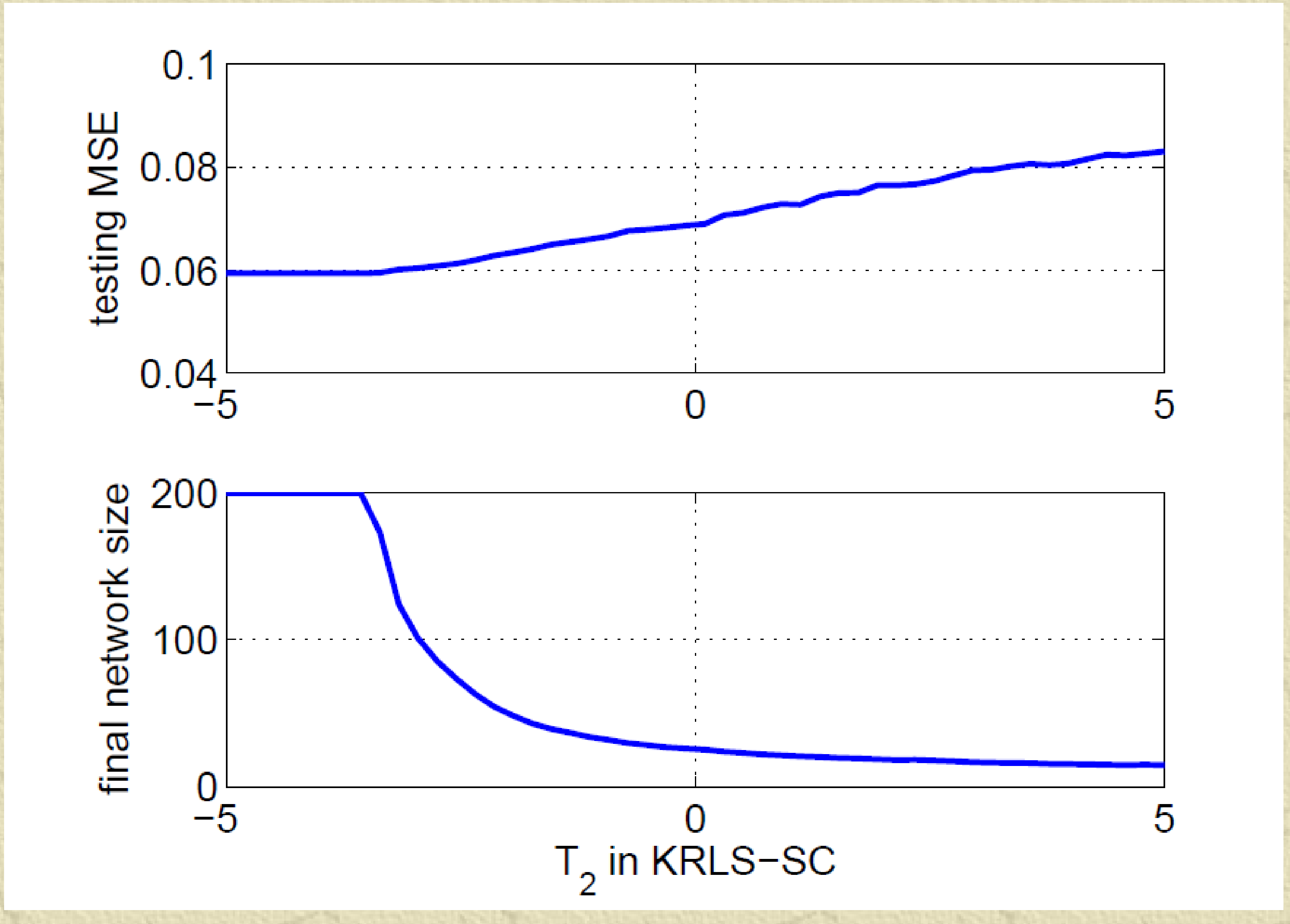


Simulation-5: nonlinear regression–5% most surprising data

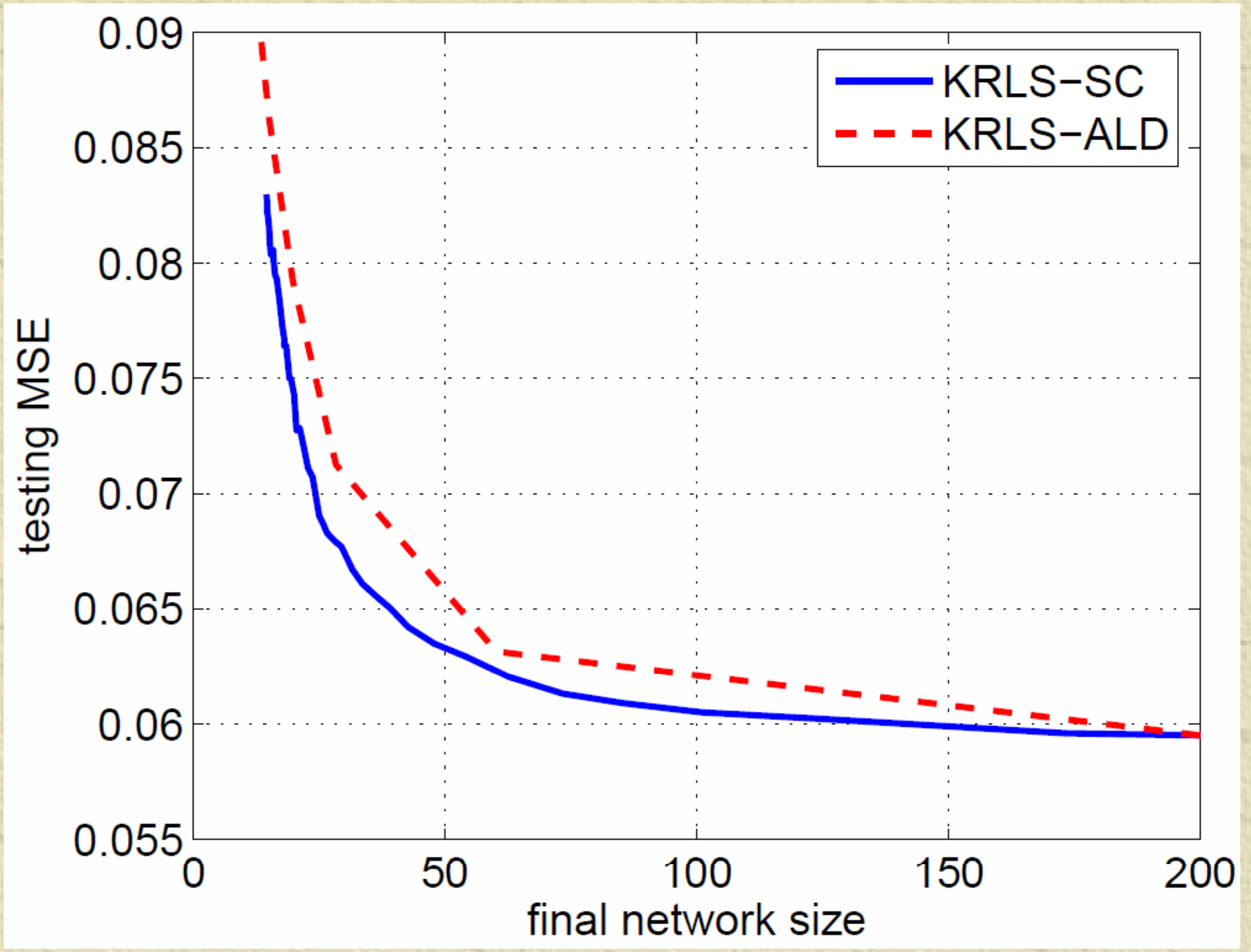




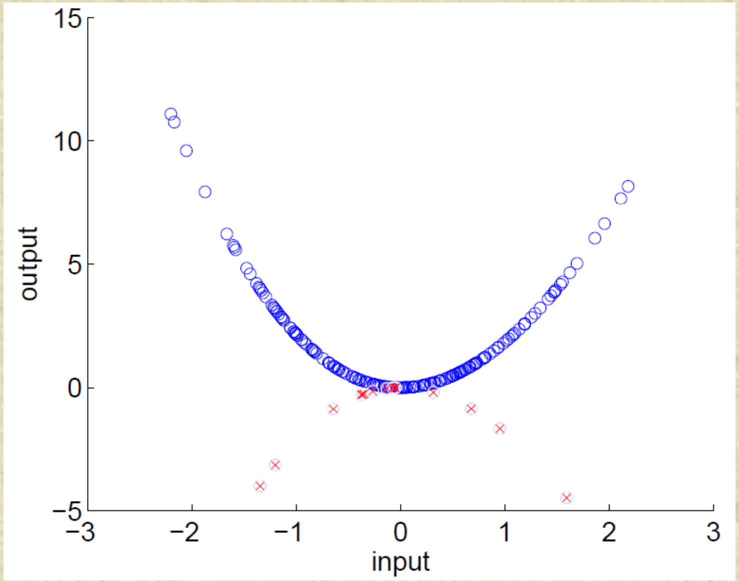
Simulation-5: nonlinear regression—redundancy removal



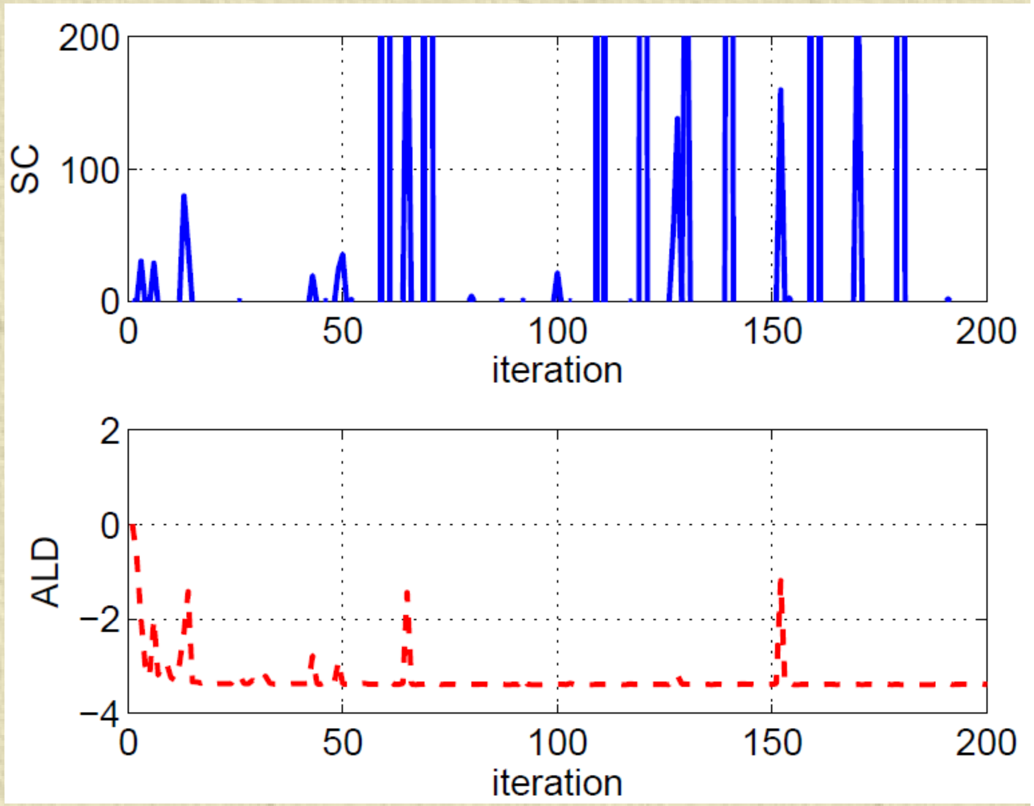
Simulation-5: nonlinear regression

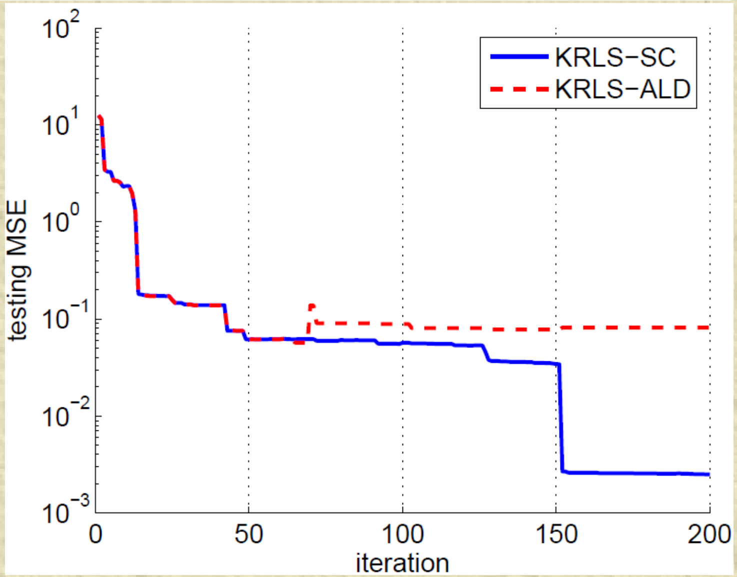


Simulation-6: nonlinear regression—abnormality detection (15 outliers)

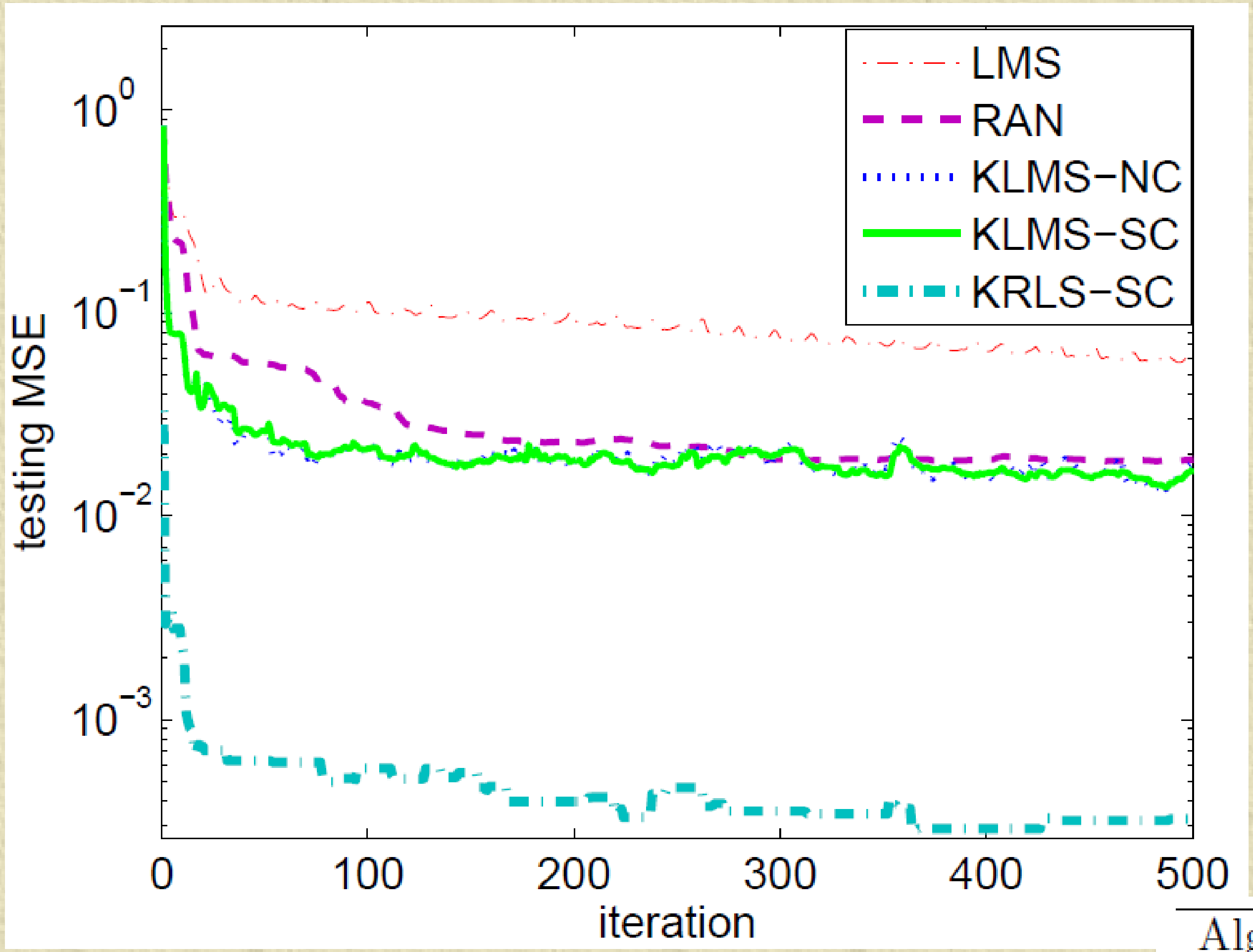


KRLS-SC



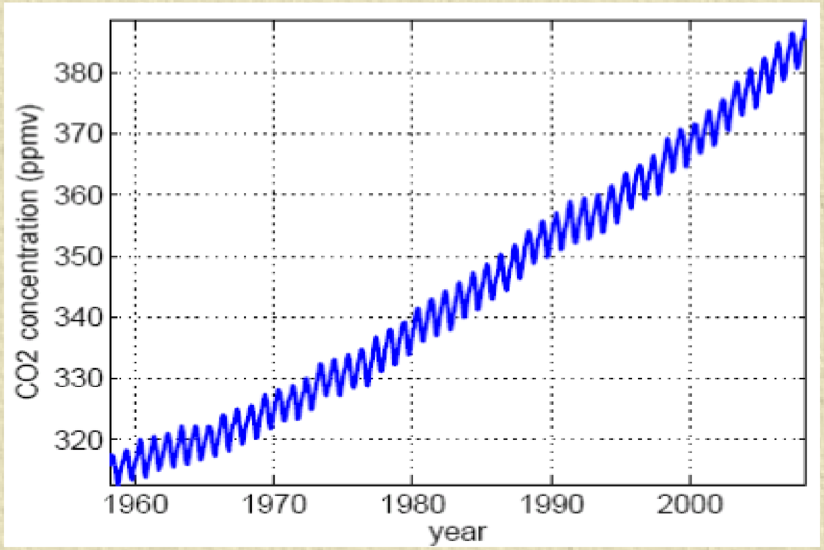


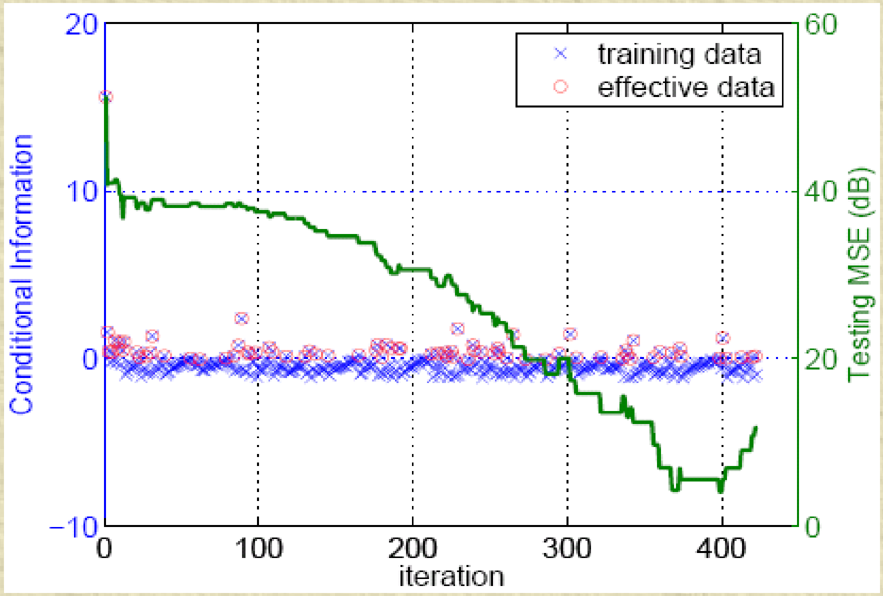
Simulation-7: Mackey-Glass time series prediction

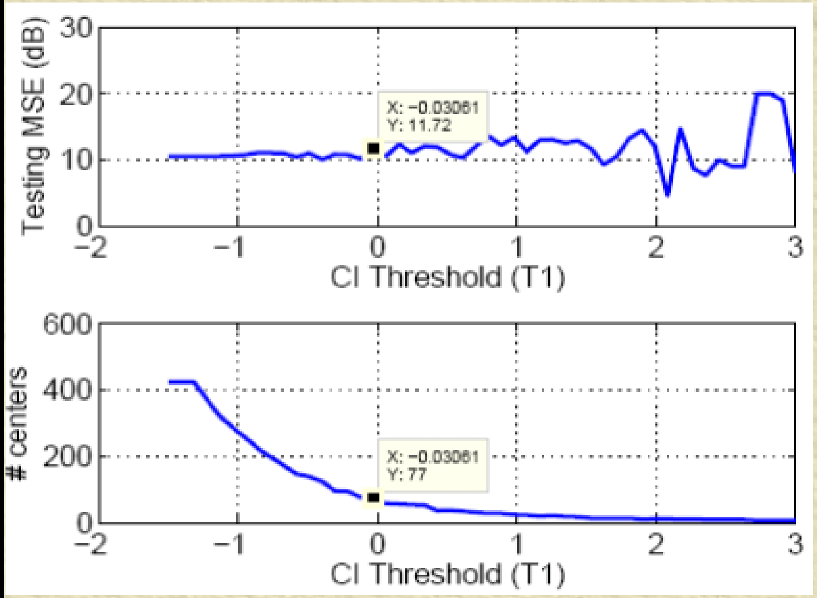


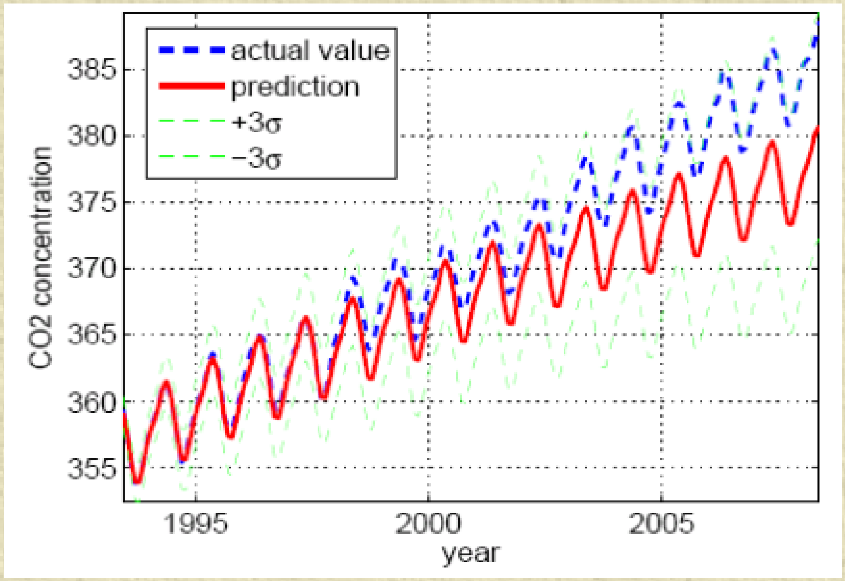
Algorithm	network size
RAN	361 ± 11
KLMS-NC	201 ± 11
KLMS-SC	109 ± 8
KRLS-SC	70 ± 9

Simulation-8: CO2 concentration forecasting









Quantized Kernel Least Mean Square

- A common drawback of sparsification methods: the *redundant* input data are purely discarded!
- The redundant data are *very useful* to update the coefficients of the current network (not so important for structure updating).
- Quantization approach: if the current quantized input has already been assigned a center, we don't need to add a new center, but update the coefficient of that center with the new information!
- Intuitively, the coefficient update may yield better accuracy and therefore a more compact network.

Chen B., Zhao S., Zhu P., Principe J. **Quantized Kernel Least Mean Square Algorithm**, submitted to IEEE Trans. Neural Networks

Quantized Kernel Least Mean Square

- Quantization in Input Space

$$\begin{cases} f_0 = 0 \\ e(i) = d(i) - f_{i-1}(\mathbf{u}(i)) \\ f_i = f_{i-1} + \eta e(i)\kappa\left(Q[\mathbf{u}(i)], .\right) \end{cases}$$

- Quantization in RKHS

$$\begin{cases} \boldsymbol{\Omega}(0) = \mathbf{0} \\ e(i) = d(i) - \boldsymbol{\Omega}(i-1)^T \boldsymbol{\varphi}(i) \\ \boldsymbol{\Omega}(i) = \boldsymbol{\Omega}(i-1) + \eta e(i)\mathcal{Q}[\boldsymbol{\varphi}(i)] \end{cases}$$

Quantization operator

- Using the quantization method to

compress the input (or feature) space

and hence to compact the RBF structure of the kernel adaptive filter

Quantized Kernel Least Mean Square

- Most of the existing VQ algorithms are not suitable for online implementation because the codebook must be supplied in advance (which is usually achieved offline), and the computational burden is rather heavy.
- A simple online VQ method:

1. Compute the distance between u(i) and C(i-1)

$$dis\left(\boldsymbol{u}(i), C(i-1)\right) = \min_{1 \le j \le size\left(C(i-1)\right)} \left\| \boldsymbol{u}(i) - C_j(i-1) \right\|$$

2. If $dis\left(\boldsymbol{u}(i), C(i-1)\right) \le \varepsilon_{\mathrm{U}}$ keep the codebook unchanged, and quantize u(i) to the closest code-vector by $\boldsymbol{a}_{j^*}(i) = \boldsymbol{a}_{j^*}(i-1) + \eta e(i)$
3. Otherwise, update the codebook: $C(i) = \{C(i-1), \boldsymbol{u}(i)\}$, and do not quantize u(i).

Quantized Kernel Least Mean Square

- Quantized Energy Conservation Relation

$$\left\|\tilde{\mathbf{\Omega}}(i)\right\|_{\mathrm{F}}^2 + \frac{e_a^2(i)}{\kappa\left(\boldsymbol{u}_q(i),\boldsymbol{u}(i)\right)^2} = \left\|\tilde{\mathbf{\Omega}}(i-1)\right\|_{\mathrm{F}}^2 + \frac{e_p^2(i)}{\kappa\left(\boldsymbol{u}_q(i),\boldsymbol{u}(i)\right)^2} + \beta_q$$

- A Sufficient Condition for Mean Square Convergence

$$\forall i,\ \begin{cases} E\left[e_a(i)\tilde{\mathbf{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] > 0 & (C1) \\ 0 < \eta \le \dfrac{2E\left[e_a(i)\tilde{\mathbf{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right]}{E\left[e_a^2(i)\right]+\sigma_v^2} & (C2) \end{cases}$$

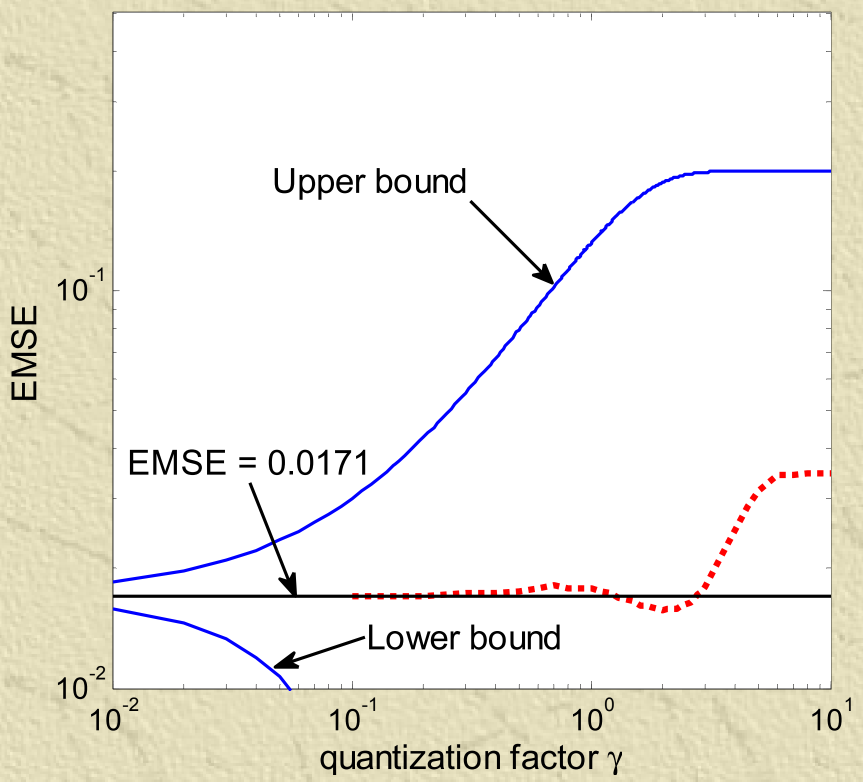
- Steady-state Mean Square Performance

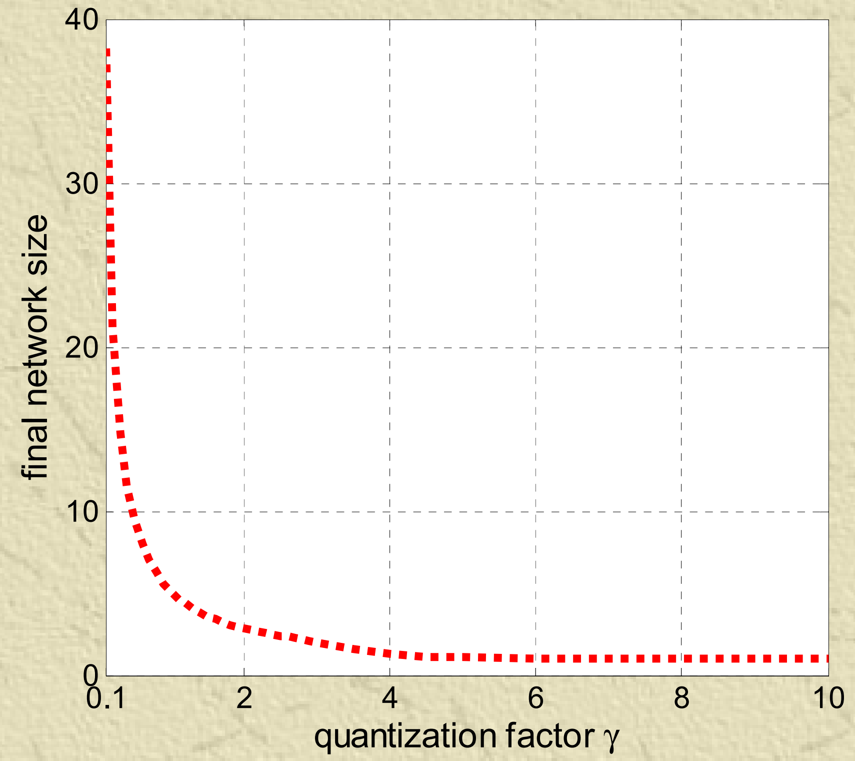
$$\max\left\{\frac{\eta\sigma_v^2 - 2\xi_\gamma}{2-\eta}, 0\right\} \le \lim_{i\to\infty} E\left[e_a^2(i)\right] \le \frac{\eta\sigma_v^2 + 2\xi_\gamma}{2-\eta}$$

Quantized Kernel Least Mean Square

- Static Function Estimation

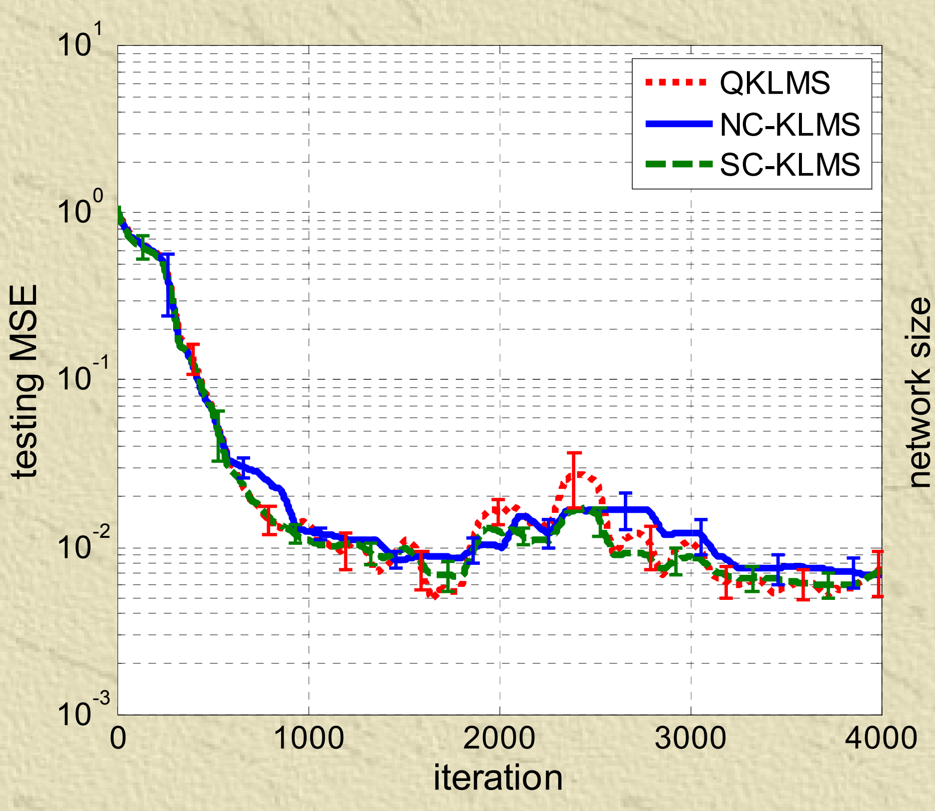
$$d(i) = 0.2\times\left[\exp\left(-\frac{(u(i)+1)^2}{2}\right)+\exp\left(-\frac{(u(i)-1)^2}{2}\right)\right]+v(i)$$

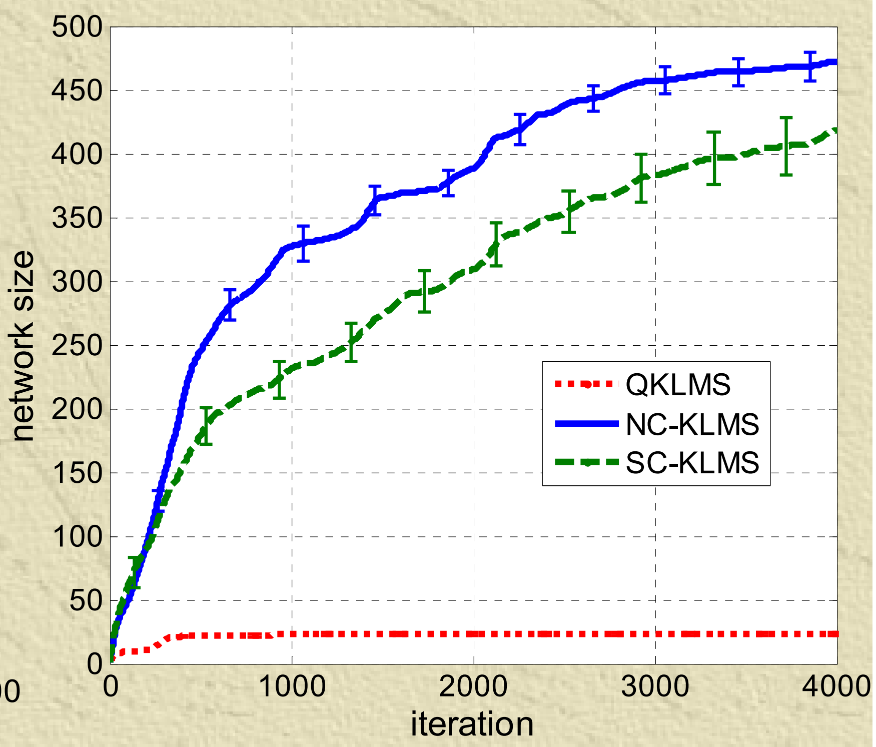




Quantized Kernel Least Mean Square

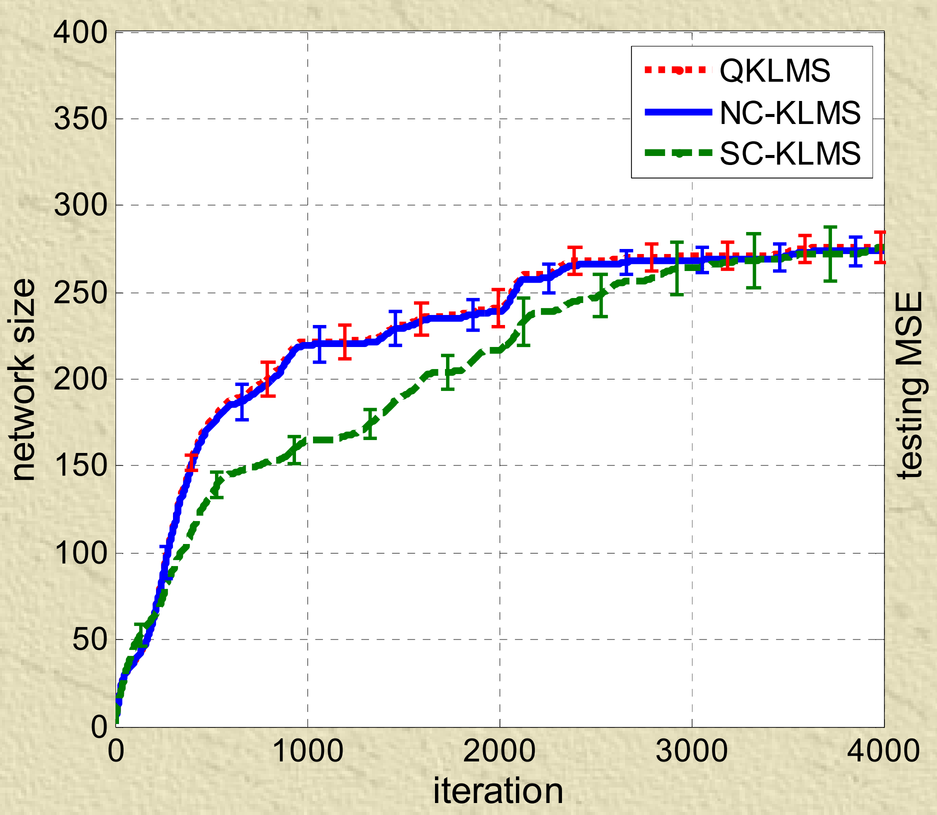
- Short Term Lorenz Time Series Prediction

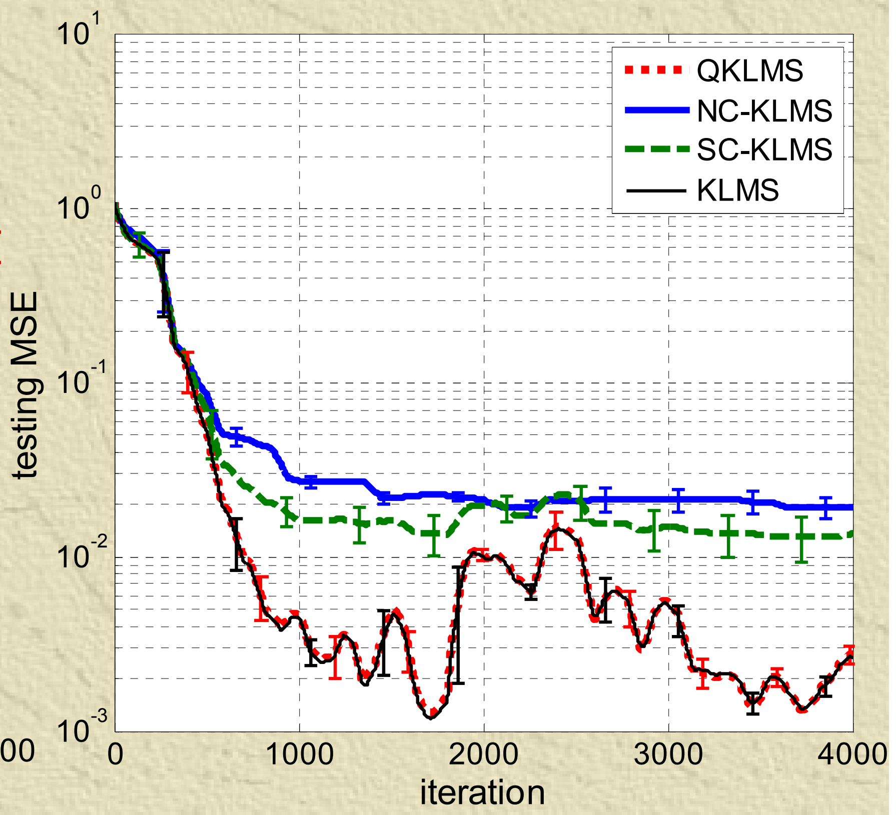




Quantized Kernel Least Mean Square

- Short Term Lorenz Time Series Prediction





Generality of the Methods Presented

- The methods presented are general tools for designing optimal universal mappings, and they can be applied in statistical learning.
- Can we apply online kernel learning for Reinforcement learning? **Definitely YES**.
- Can we apply online kernel learning algorithms for classification? **Definitely YES**.
- Can we apply online kernel learning for more abstract objects, such as point processes or graphs? **Definitely YES**

Redefinition of On-line Kernel Learning

- Notice how problem constraints affected the form of the learning algorithms.
- On-line Learning: A process by which the *free parameters* and the *topology* of a ‘learning system’ are *adapted* through a process of stimulation by the environment in which the system is embedded.
- Error-correction learning + memory-based learning
  - What an interesting (biological plausible?) combination.

Impacts on Machine Learning

- KAPA algorithms can be very useful in large scale learning problems.
- Just sample randomly the data from the data base and apply on-line learning algorithms
- There is an extra optimization error associated with these methods, but they can be easily fit to the machine contraints (memory, FLOPS) or the processing time constraints (best solution in x seconds).