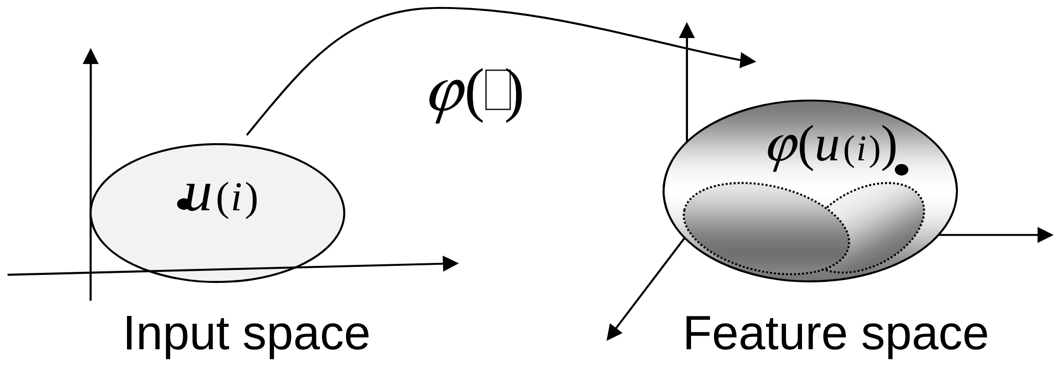
Some Recent Advances in Kernel Adaptive Filtering

Badong Chen

- Computational NeuroEngineering Laboratory
- University of Florida

Adaptive Filtering in Feature Space

Can we learn nonlinear structure using knowledge of linear adaptive filtering?



Linear model in F

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

How do we choose the mappings?

Kernel Methods

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

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

- Mercer's theorem
  - Let $\kappa(x,y)$ 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

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

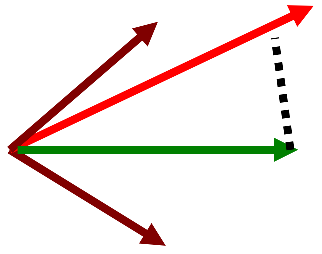
Kernel Methods

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \Longrightarrow \{(\phi(\mathbf{x}_i), y_i)\}_{i=1}^{n}$$

$$\mathbf{w}^* = \min_{\mathbf{w}\in\mathbf{R}^m} ||\mathbf{y} - \Phi^\top \mathbf{w}||_2^2$$

Representer Theorem: The optimal filter exists in the span of input data!

$$\underline{w} = \sum_{i=1}^{n} \alpha_i \underline{\phi(x_i)}$$



$\boxed{n \text{ parameters to learn}}$

Kernel Least Mean Square (KLMS)

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

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

$$\Omega_0 = 0$$
$$e(i) = d(i) - \langle \Omega_{i-1}, \varphi(u_i) \rangle_F$$
$$\Omega_i = \Omega_{i-1} + \eta \varphi(u_i) e(i)$$
$$\Omega_i = \sum_{j=1}^{i} \eta e(j) \varphi(u_j)$$

$$\Omega_0 = 0$$
$$e(1) = d(1) - \langle \Omega_0, \varphi(u_1) \rangle_F = d(1)$$
$$\Omega_1 = \Omega_0 + \eta \varphi(u_1) e(1) = a_1 \varphi(u_1)$$
$$\begin{aligned} e(2) &= d(2) - \langle \Omega_1, \varphi(u_2) \rangle_F \\ &= d(2) - \langle a_1 \varphi(u_1), \varphi(u_2) \rangle_F \\ &= d(2) - a_1 \kappa(u_1, u_2) \end{aligned}$$
$$\begin{aligned} \Omega_2 &= \Omega_1 + \eta \varphi(u_2) e(2) \\ &= a_1 \varphi(u_1) + a_2 \varphi(u_2) \end{aligned}$$
...

- Compute output at a new sample u as

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

Energy Conservation Relation

Energy conservation relation holds in RKHS!

- Energy conservation in RKHS

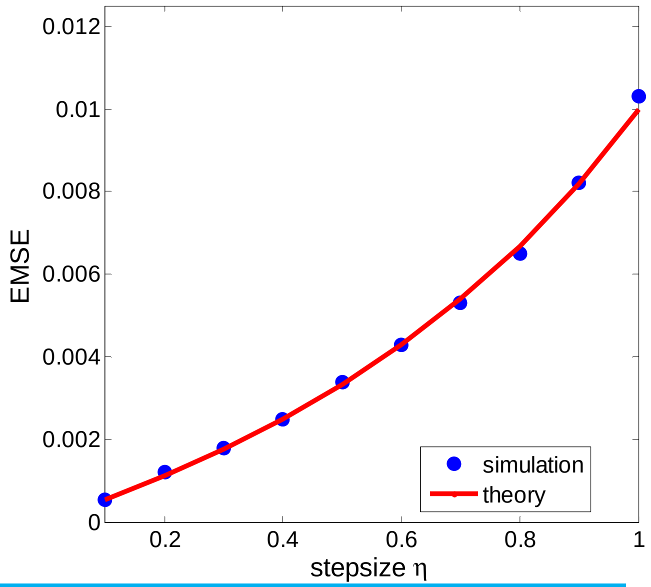
$$\left\|\tilde{\mathbf{\Omega}}(i)\right\|_{\mathrm{F}}^{2}+\frac{e_a^2(i)}{\kappa\left(\mathbf{u}(i),\mathbf{u}(i)\right)}=\left\|\tilde{\mathbf{\Omega}}(i-1)\right\|_{\mathrm{F}}^{2}+\frac{e_p^2(i)}{\kappa\left(\mathbf{u}(i),\mathbf{u}(i)\right)}$$

- Upper bound on step size for mean square convergence

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

- Steady-state mean square performance

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



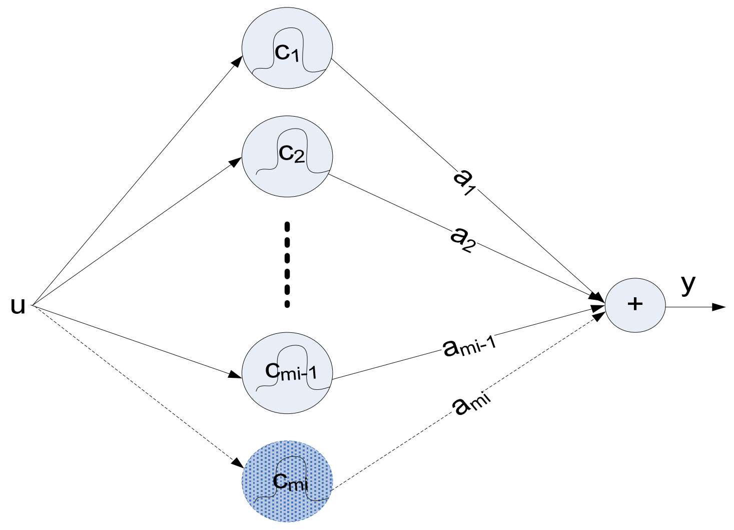
Chen B., Zhao S., Zhu P., Principe J. Mean Square Convergence Analysis of the Kernel Least Mean Square Algorithm. ***Signal Processing*** (Accepted)

Challenges in Implementation

Curbing network growth

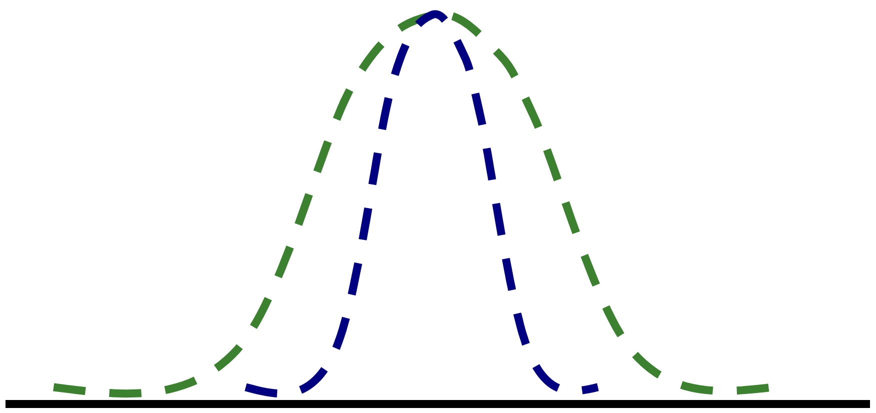
$$y_n = \sum_{i=1}^{n-1} \mathbf{a}(i)\kappa(\mathbf{x}_n, \mathbf{x}_i)$$

$$\mathbf{a}(n) = \eta e(n)$$



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

Choosing kernel size



Basic Sparsification Criteria

- Novelty Criterion (NC)

NC first computes the distance of u(i + 1) to the present dictionary

$$dis_1 = \min_{c_j \in C(i)} \left\| u(i+1) - c_j \right\|$$

If dis$_1$ < δ_1, u(i+1) will not be added into the dictionary. Otherwise, the prediction error is computed and only if |e(i + 1)| > δ_2, u(i + 1) will be accepted as a new center. δ_1 and δ_2 are two user-specified parameters.

- Approximate Linear Dependency (ALD)

Test the distance of the new input to the linear span of the present dictionary in the feature space.

$$dis_2 = \min_{\forall b} \| \varphi(u(i+1)) - \sum_{c_j \in C(i)} b_j \varphi(c_j) \|$$

This criterion is computational demanding but the needed values are already available in the KRLS algorithm. For KLMS it can be simplified, and defaults to δ_1 for RBFs

$$dis_3 = \min_{\forall b, c_j \in C(i)} \| \varphi(u(i+1)) - b\varphi(c_j) \|$$

Basic Sparsification Criteria

- Surprise, a new criterion based on Information Theory

Definition: Surprise $S_T(u;d)$ is a subjective information measure of an exemplar $(u;d)$ with respective to a learning system T . It is defined as the negative log likelihood of the exemplar given the the learning system's hypothesis on the data distribution:

$$S_T(u,d) = -\ln p(u,d \mid T)$$

where $p(u;d|T)$ is the subjective probability of $(u;d)$ hypothesized by T .

According to this measure, we can classify the new exemplar into 3 categories:

- *Abnormal*: $S_T(u,d) > T_1$. throw away or control training
- *Learnable*: $T_1 > S_T(u,d) > T_2$. train
- *Redundant*: $S_T(u,d) < T_2$. throw away

Basic Sparsification Criteria

- Evaluation of Surprise

In order to evaluate surprise the posterior distribution must be evaluated. We used Gaussian processes theory and estimate as

$$S_T(u(i+1),d(i+1)) = \ln\sigma(i+1) + \frac{(d(i+1)-\hat{d}(i+1))^2}{2\sigma^2(i+1)} - \ln p(u(i+1)\,|\,T(i)) + \ln\sqrt{2\pi}$$

Surprise is

- Proportional to the prediction error square
- Proportional to the prediction variance if the magnitude of the error is small
- Very high if the prediction error is large and the prediction variance is small.
- Larger for rare data occurrences (for filtering)
- For KLMS use the simplified equation $r(i+1) = \lambda + \kappa(0) - \max_{\forall c_j \in C(i)} \frac{\kappa^2(u(i+1),c_j)}{\kappa(0)}$

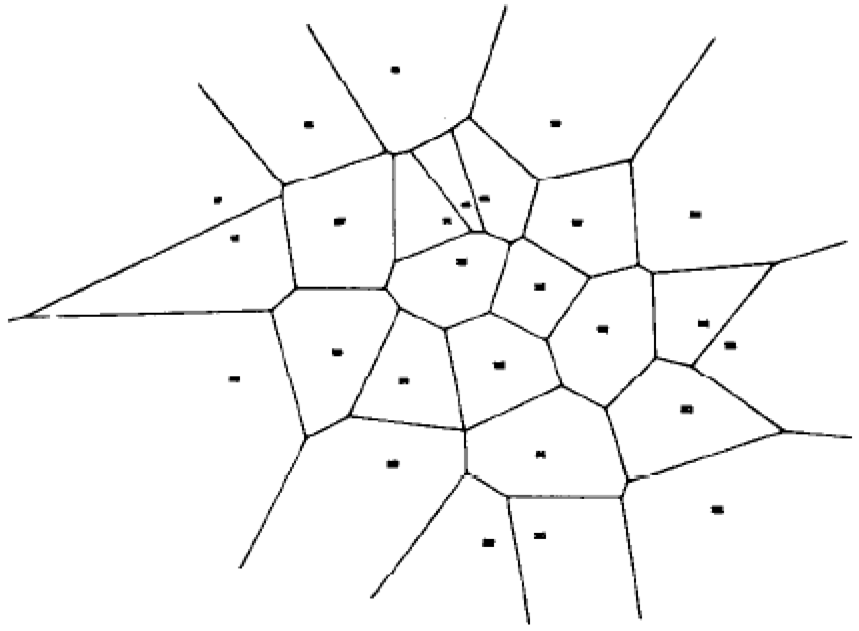
$$S_T(u(i+1),d(i+1)) = \frac{\ln r(i+1)}{2} + \frac{(d(i+1)-\hat{d}(i+1))^2}{2r(i+1)} - \ln p(u(i+1)\,|\,T(i)) + \ln\sqrt{2\pi}$$

Quantization Approach

- A common drawback of the previous sparsification methods**:** the *redundant* input data are purely discarded! Actually the redundant data are *very useful* and can be, for example, utilized to update the coefficients of the current network, although they are not so important for structure update (adding a new center).
- Quantization approach: the input space is quantized, if the current quantized input has already been assigned a center, we don't need to add a new, but update the coefficient of that center !

Quantization Approach

Partition of the Input Space

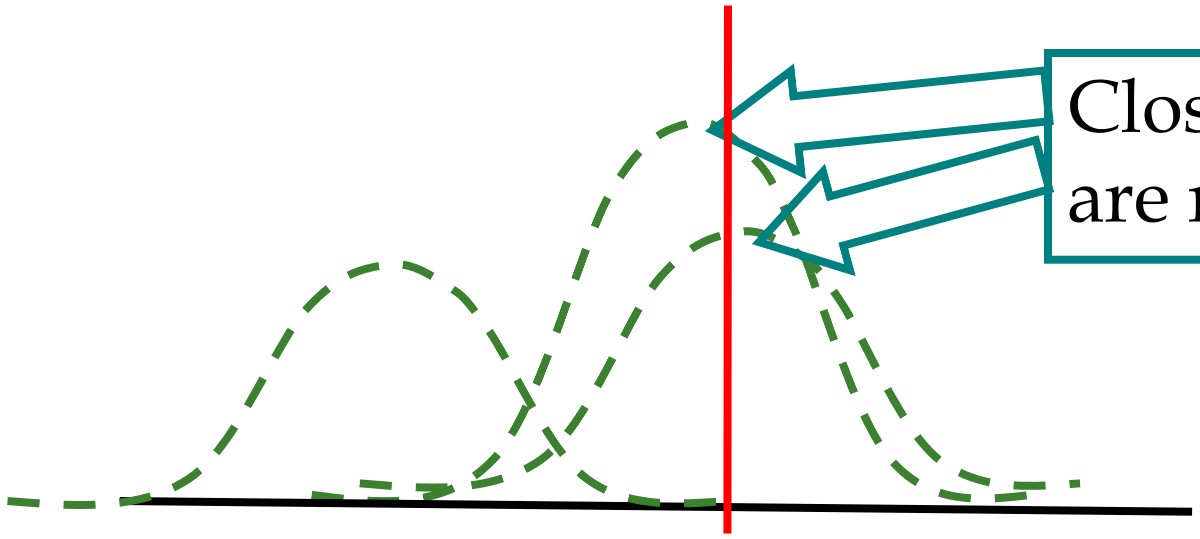


$$S_n = \{\boldsymbol{u} \mid \boldsymbol{u} \in \mathrm{U}, \|\boldsymbol{u} - \boldsymbol{c}(n)\| \le \|\boldsymbol{u} - \boldsymbol{c}(n')\| \text{ for each } n' \neq n\}$$

Quantized KLMS

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

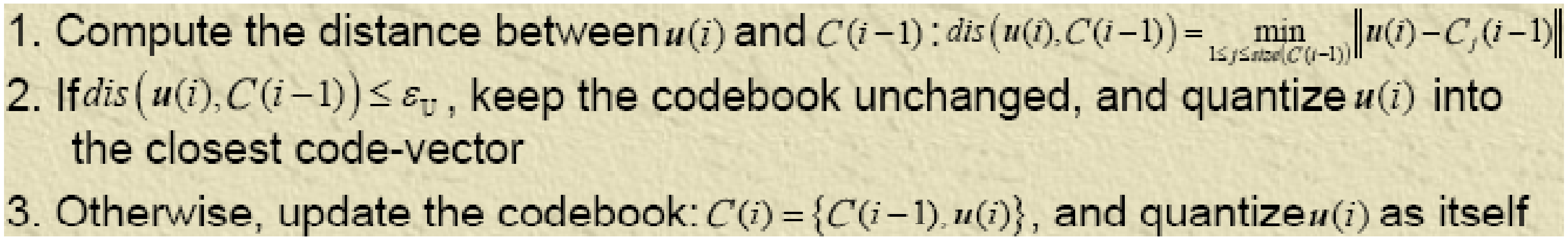
Quantization operator



Close centers are redundant

Online Vector Quantization

- The key problem is the vector quantization (VQ): Information Theory? Information Bottleneck? ……
- There are many VQ methods in literature. Most of the existing VQ algorithms, however, are not suitable for online implementation because the codebook must be supplied in advance (which is usually trained on an offline data set), and the computational burden is rather heavy.
- A simple online VQ method:



1. Compute the distance between $u(i)$ and $C(i-1)$: $dis\left(u(i), C(i-1)\right) = \min_{1 \le j \le size(C(i-1))} \left\| u(i) - C_j(i-1) \right\|$
2. If $dis\left(u(i), C(i-1)\right) \le \varepsilon_U$, keep the codebook unchanged, and quantize $u(i)$ into the closest code-vector
3. Otherwise, update the codebook: $C(i) = \{C(i-1), u(i)\}$, and quantize $u(i)$ as itself

Online Vector Quantization

Algorithm 1 QKLMS Algorithm

```
*Initialization:* stepsize $\eta$, kernel width $\sigma_M$, quantization threshold $\varepsilon_{\mathbb{U}} > 0$, center dictionary
$\boldsymbol{C}(1) = \{\boldsymbol{u}(1)\}$ and coefficient vector $\boldsymbol{a}(1) = [\eta d(1)]$
**while** $\{\boldsymbol{u}(1), d(1)\}$ is avaiable **do**
  $e(i) = d(i) - \sum_{j=1}^{K(i-1)} \boldsymbol{a}_j(i-1)\kappa_{\sigma_M}(\boldsymbol{u}(i), \boldsymbol{C}_j(i-1))$
  $dis(\boldsymbol{u}(i), \boldsymbol{C}(i-1)) = \min_{1 \le j \le K(i-1)} \|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\|_{\mathbb{F}}$
  $j^* = \arg\min_{1 \le j \le K(i-1)} \|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\|_{\mathbb{F}}$
  **if** $dis(\boldsymbol{u}(i), \boldsymbol{C}(i-1)) \le \varepsilon_u$ **then**
    $\boldsymbol{C}(i) = \boldsymbol{C}(i-1)$, $\boldsymbol{a}_{j^*}(i) = \boldsymbol{a}_{j^*}(i-1) + \eta e(i)$
  **else**
    $\boldsymbol{C}(i) = \{\boldsymbol{C}(i-1), \boldsymbol{u}(i)\}$, $\boldsymbol{a}(i) = [\boldsymbol{a}(i-1), \eta e(i)]$
  **end if**
**end while**
```

Convergence Analysis

- Quantized Energy Conservation Relation

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

- Sufficient Condition for MS Convergence

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

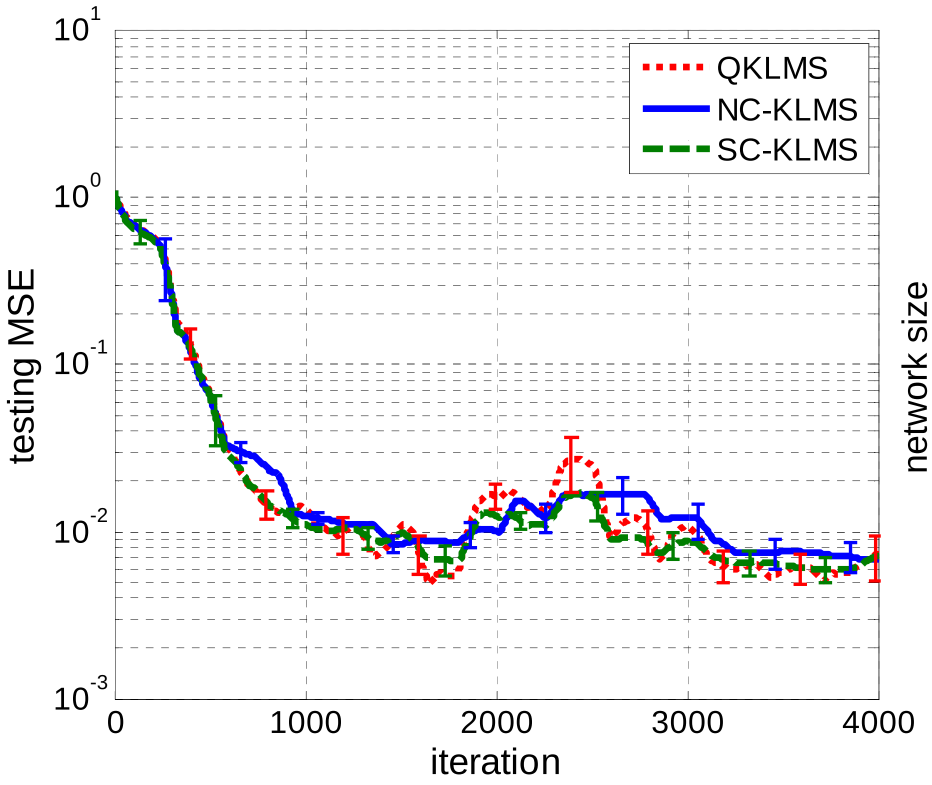
- Steady-state Mean Square Performance

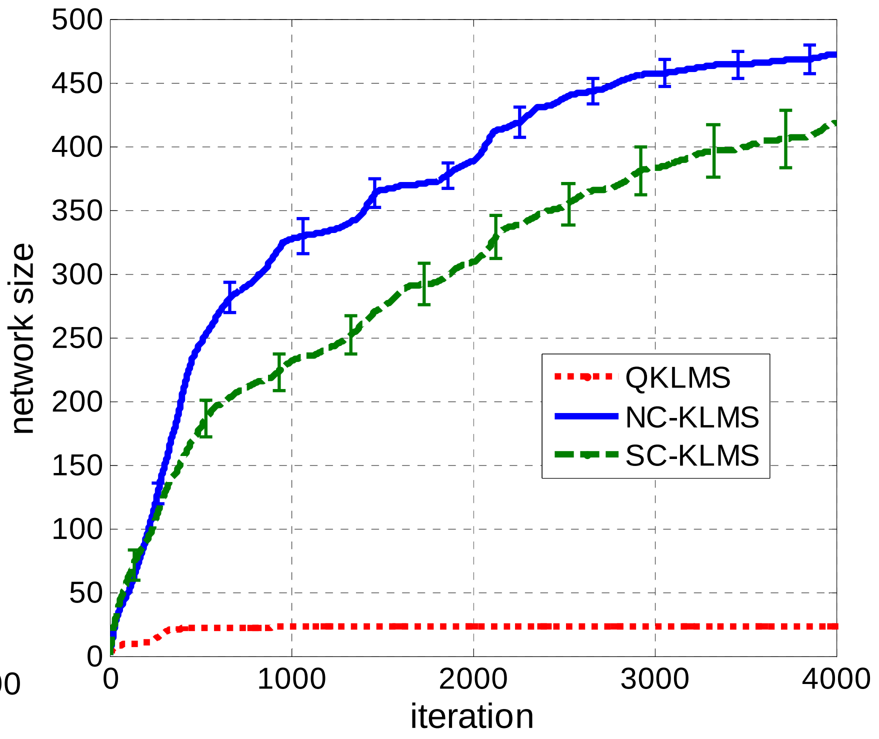
$$\max\left\{\frac{\eta\sigma_v^2-2\xi_\gamma}{2-\eta},0\right\}\le\lim_{i\to\infty}E\left[e_a^2(i)\right]\le\frac{\eta\sigma_v^2+2\xi_\gamma}{2-\eta}$$

Chen B., Zhao S., Zhu P., Principe J. Quantized kernel least mean square algorithm. ***IEEE Trans. Neural Networks and Learning Systems***, *vol.23, 22-32*

QKLMS

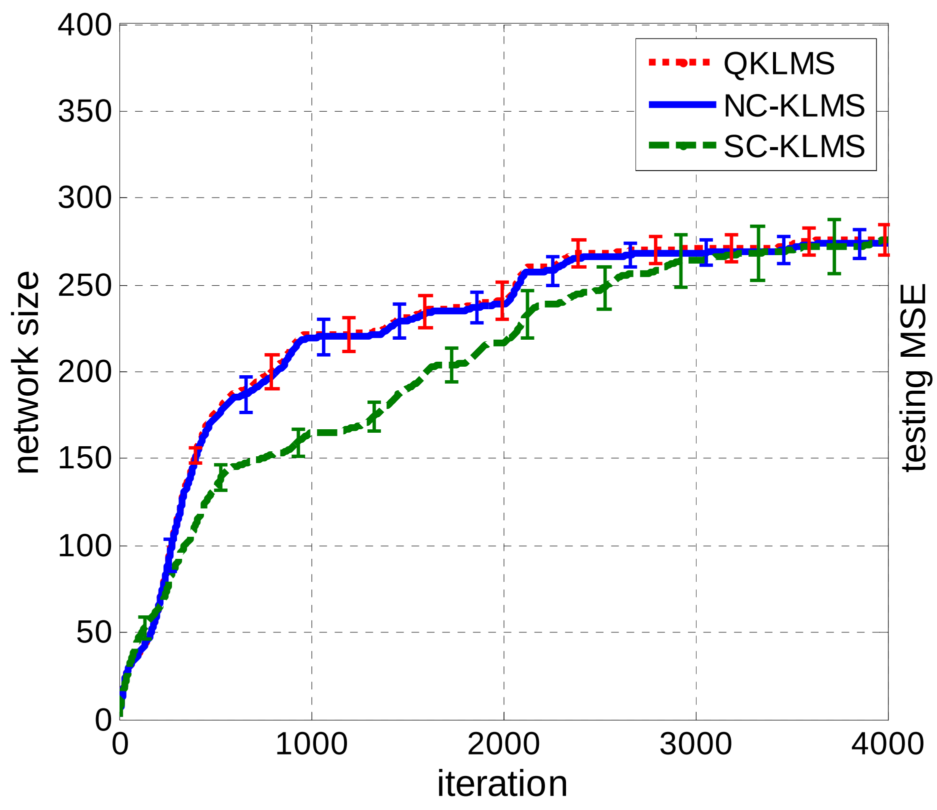
- Short Term Lorenz Time Series Prediction

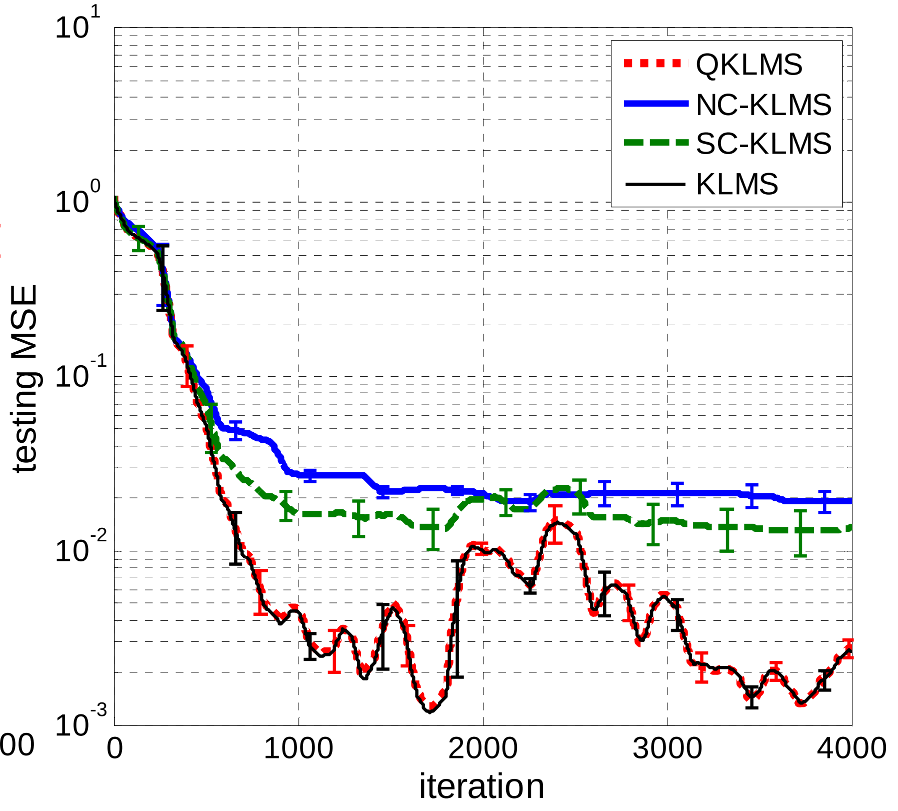




QKLMS (cont)

- Short Term Lorenz Time Series Prediction


network size
iteration
QKLMS
NC-KLMS
SC-KLMS


testing MSE
iteration
QKLMS
NC-KLMS
SC-KLMS
KLMS
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}