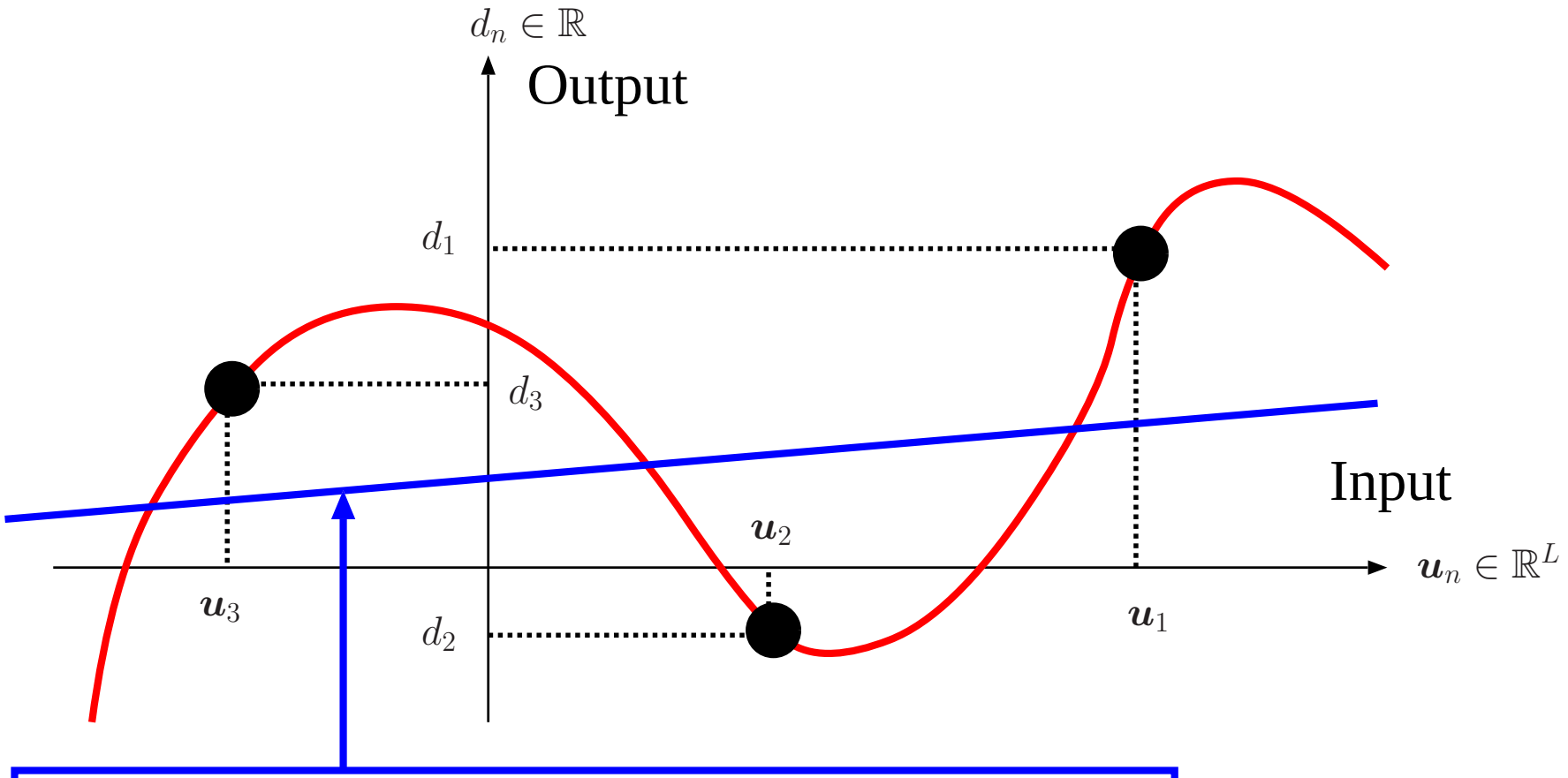
Multikernel Adaptive Filtering

Masahiro Yukawa
Dept. EEE, Niigata University, JAPAN

HIIT Seminar (Helsinki Institute for Information Technology)

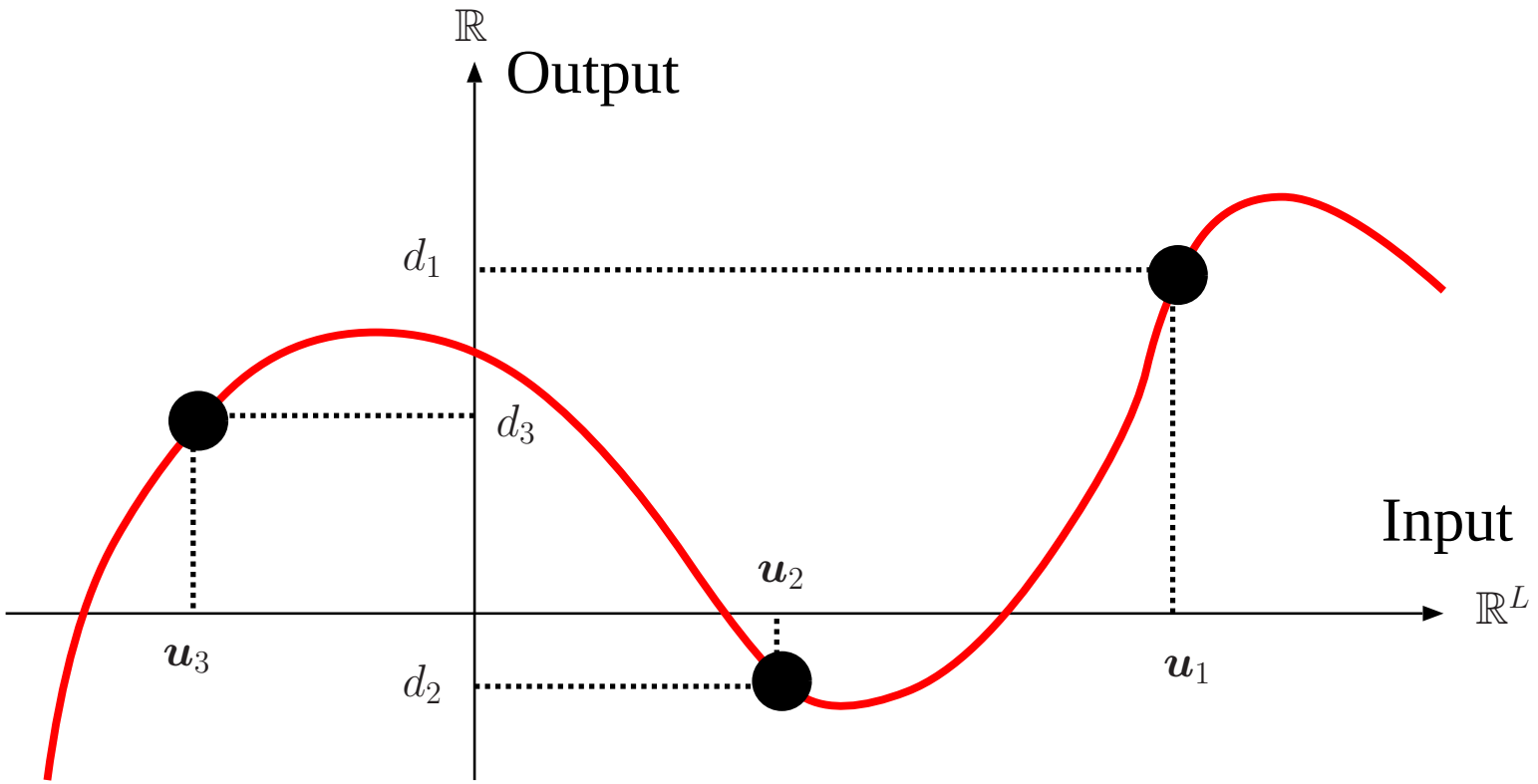
August 17, 2012 — @Aalto University, Finland

Nonlinear Function Estimation



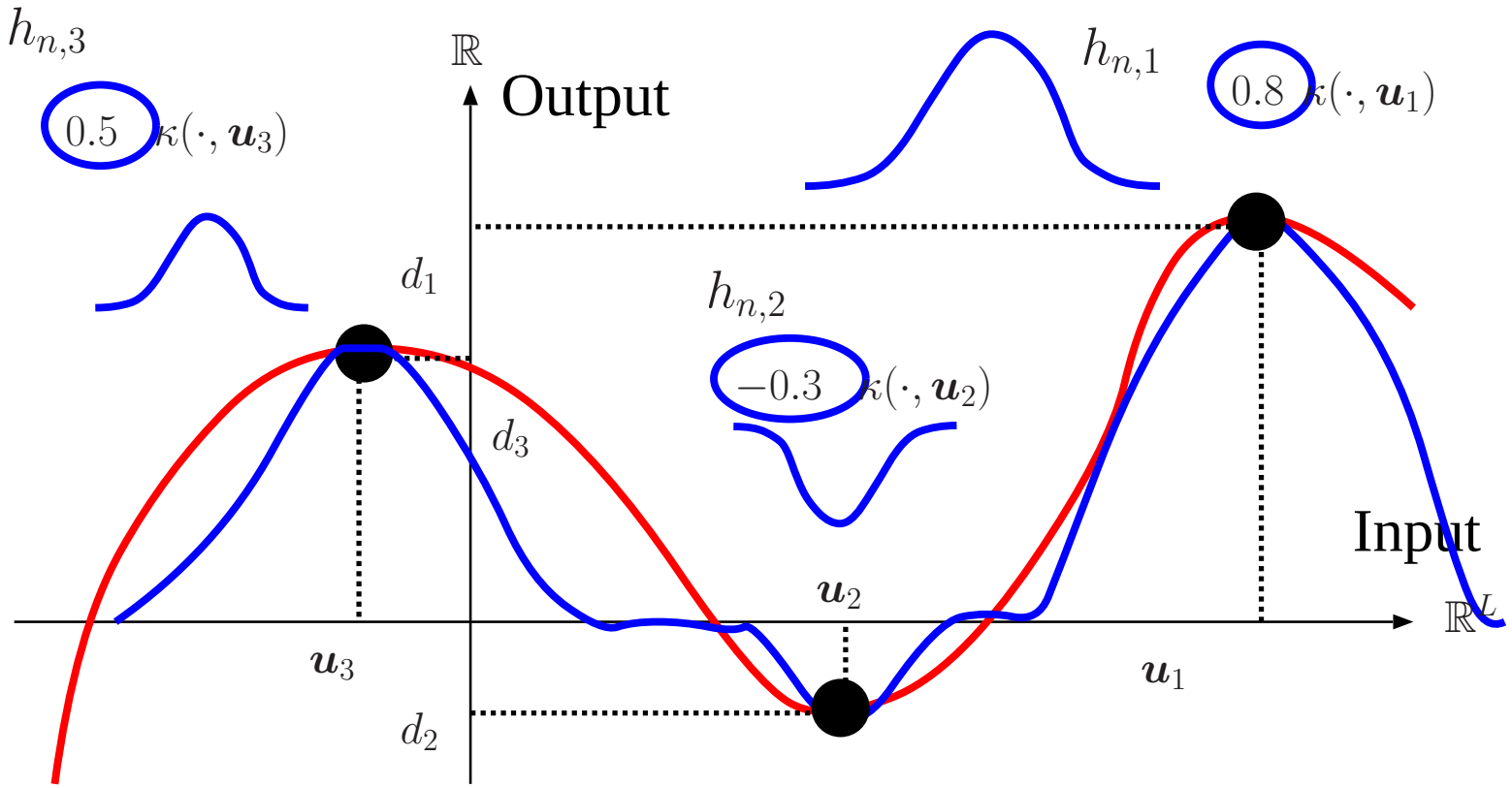
Linear approximation does not fit the data

Kernel-Based Adaptive Filtering



- $\hat{d}_n = \psi_n(\boldsymbol{u}_n) := \sum h_{n,j}\kappa(\boldsymbol{u}_n, \boldsymbol{u}_j), \;\; \boldsymbol{u}_n \in \mathbb{R}^L$

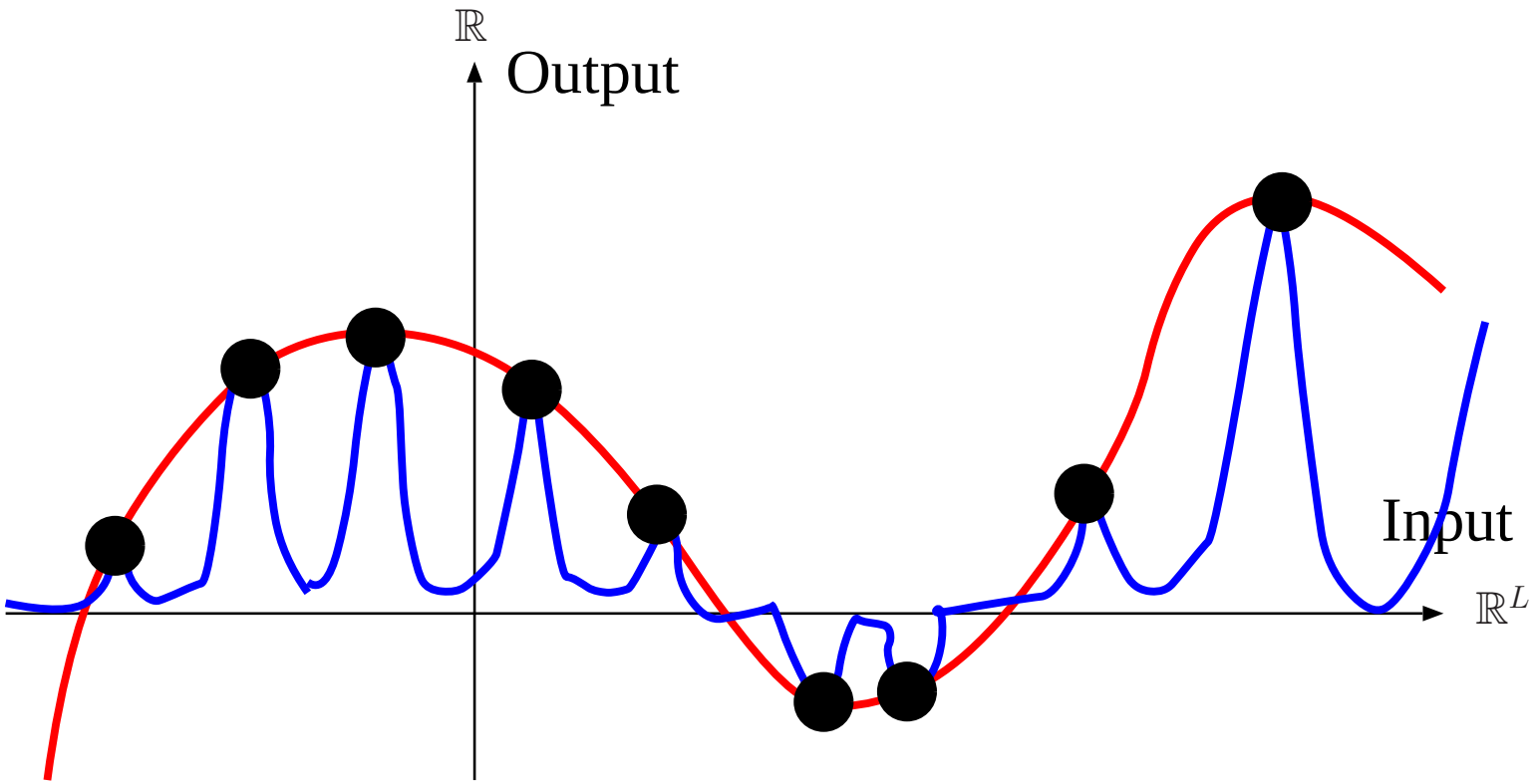
Kernel-Based Adaptive Filtering



- $\hat{d}_n = \psi_n(\boldsymbol{u}_n) := \sum h_{n,j} \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j), \;\; \boldsymbol{u}_n \in \mathbb{R}^L$
- Assume the use of Gaussian kernel:

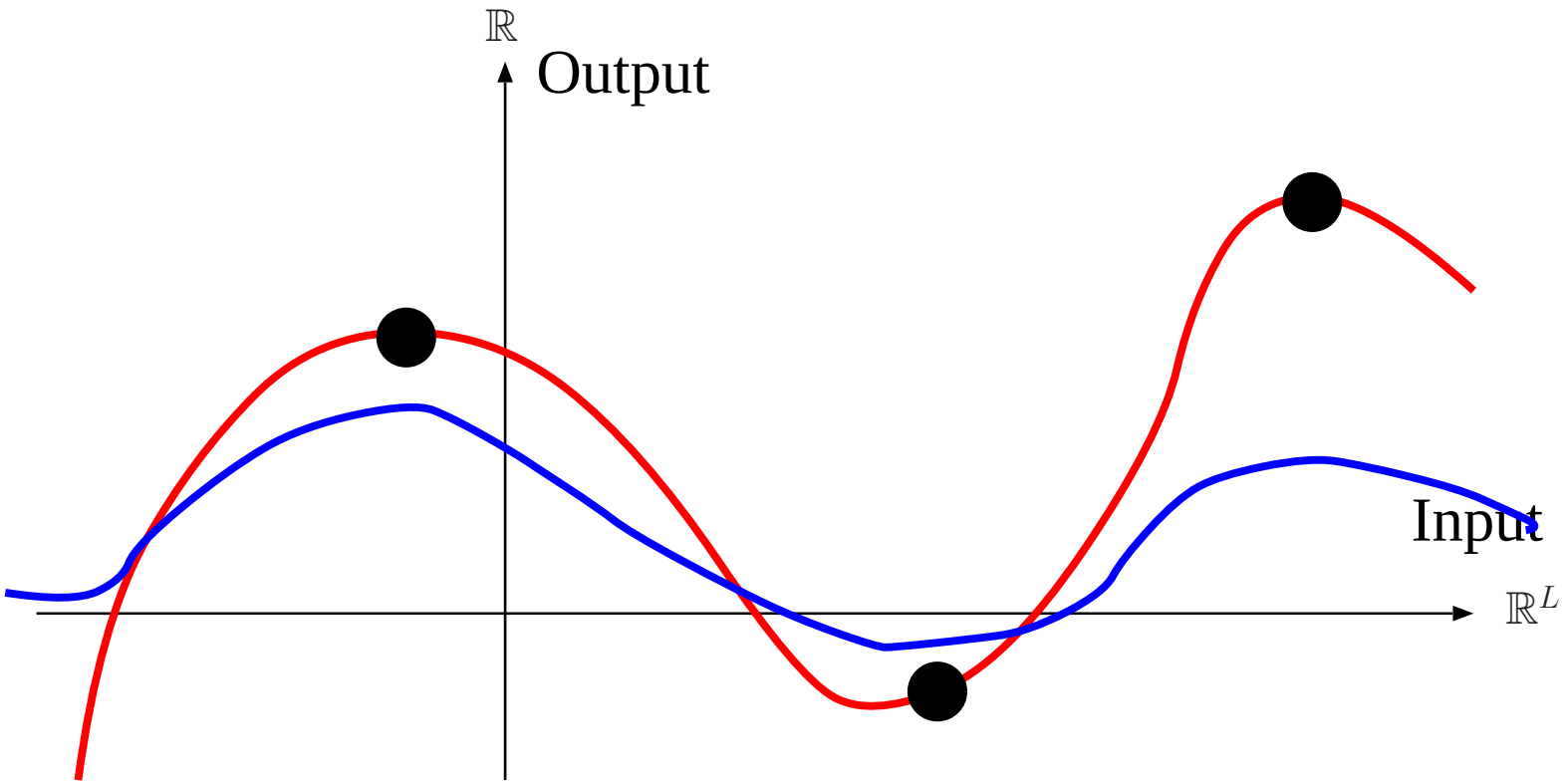
$$\kappa(\boldsymbol{u}, \boldsymbol{v}) := \exp\left(-\zeta \left\|\boldsymbol{u} - \boldsymbol{v}\right\|^2\right), \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L, \quad \zeta > 0$$

A Small Width Results in...



a huge number of center points to be used for a reasonable approximation!

A Large Width Results in...



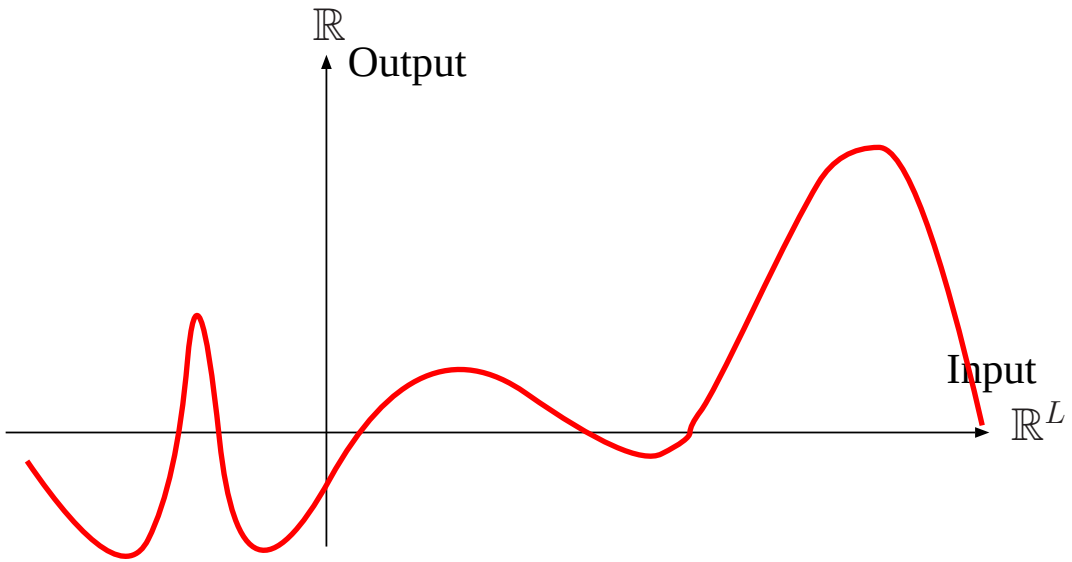
a large amount of estimation errors!

The choice of the kernel parameter is important!

Why to Use Multiple Kernels?

- In the single kernel approach, the kernel width should be small enough to express rapid changes

The multikernel approach reduces the number of center points to be used by allocating an individual shape for each center point



Why to Use Multiple Kernels?

- In the single kernel approach, the kernel width should be small enough to express rapid changes

The multikernel approach reduces the number of center points to be used by allocating an individual shape for each center point

- The function to be estimated could be time-varying
  $\Rightarrow$ It would be difficult to select an adequate kernel parameter prior to adaptation!

The multikernel approach changes the kernel shape adaptively.

Why to Use Multiple Kernels?

- In the single kernel approach, the kernel width should be small enough to express rapid changes

The multikernel approach reduces the number of center points to be used by allocating an individual shape for each center point

- The function to be estimated could be time-varying
  $\Rightarrow$ It would be difficult to select an adequate kernel parameter prior to adaptation!

The multikernel approach changes the kernel shape adaptively.

- Our multikernel approach is fully adaptive.
- It has a higher degree of freedom (larger # of parameters $r_n M$) compared to the typical MKL techniques ($r_n + M$).
  $r_n =$ "# of center points" and $M =$ "# of kernels".

Two Key Issues in Kernel AF

1. **Algorithm Design**
   - NLMS Algorithm
   - Affine Projection Algorithm
   - Adaptive Parallel Subgradient Projection Algorithm
2. **Sparsification Technique**
   = how to select data (centers of Gaussian) to be used in estimation

- Kivinen *et al.*, IEEE TSP 2004
- Engel *et al.*, IEEE TSP 2004
- Liu and Príncipe, EURASIP JASP 2008, Chen *et al.*, IEEE TNNLS 2011
- Slavakis *et al.*, IEEE TSP 2008, IEEE TSP 2009
- Richard *et al.*, IEEE TSP 2009, Parreira *et al.*, IEEE TSP 2012

A classification of algorithms:
"RKHS projection approach" and "parameter-space projection approach"

MKNLMS-CS Algorithm (1/2)

Sparsification Technique (Dictionary Design)

- $\mathcal{J}_n^{\mathrm{CS}} \subset \{0, 1, 2, \cdots, n\}$: dictionary at time n
  $i \in \mathcal{J}_n^{\mathrm{CS}} \Leftrightarrow \boldsymbol{u}_i$ is in the dictionary
- $\mathcal{J}_0^{\mathrm{CS}} := \emptyset$ (Initialization)
- Multiple kernels: κ_1, κ_2, $\cdots$, κ_M (kernel index set $\mathcal{M} := \{1, 2, \cdots, M\}$)
- Admission rule: If the coherence criterion is satisfied:

$$\|\boldsymbol{K}_n\|_{\max} := \max_{m \in \mathcal{M}} \max_{j \in \mathcal{J}_n^{\mathrm{CS}}} |\kappa_m(\boldsymbol{u}_n, \boldsymbol{u}_j)| \leq \delta \quad (\delta > 0),$$

  then $\boldsymbol{u}_n$ is inserted into the dictionary; i.e., $\mathcal{J}_n^{\mathrm{CS}} := \mathcal{J}_{n-1}^{(\mathrm{CS})} \cup \{n\}$
- $\lim_{n \to \infty} \left|\mathcal{J}_n^{\mathrm{CS}}\right| < \infty$ provided that the input space $\mathcal{U} \subset \mathbb{R}^L$ is compact

Richard *et al.*, "Online prediction of time series data with kernels," IEEE Trans. Signal Process., vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

MKNLMS-CS Algorithm (2/2)

Algorithm Design

- Our estimator: $\psi_n(\boldsymbol{u}) = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}_n^{\mathrm{CS}}} h_{j,n}^{(m)} \kappa_m(\boldsymbol{u}, \boldsymbol{u}_j), \qquad \boldsymbol{u} \in \mathcal{U}$
- $\widehat{d}_n = \psi_n(\boldsymbol{u}_n) = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}_n^{\mathrm{CS}}} h_{j,n}^{(m)} \kappa_m(\boldsymbol{u}_n, \boldsymbol{u}_j) = \mathrm{tr}\left(\boldsymbol{H}_n^{\mathsf{T}} \boldsymbol{K}_n\right) = \langle \boldsymbol{H}_n, \boldsymbol{K}_n \rangle$
  - $[\boldsymbol{H}_n]_{j,m} := h_{j,n}^{(m)}$
  - $[\boldsymbol{K}_n]_{j,m} := \kappa_m(\boldsymbol{u}_n, \boldsymbol{u}_j)$
- Update equation (step size $\eta \in [0, 2]$, regularization parameter $\rho > 0$):

$$\boldsymbol{H}_{n+1} := \boldsymbol{H}_n + \eta \frac{d_n - \langle \boldsymbol{H}_n, \boldsymbol{K}_n \rangle}{\|\boldsymbol{K}_n\|^2 + \rho} \boldsymbol{K}_n = \boldsymbol{H}_n + \eta' \left(P_{\Pi_n}(\boldsymbol{H_n}) - \boldsymbol{H}_n\right)$$

where $P_{\Pi_n}(\boldsymbol{H_n})$ is the projection of $\boldsymbol{H}_n$ onto the hyperplane

$$\Pi_n := \{\boldsymbol{H} : \langle \boldsymbol{H}, \boldsymbol{K}_n \rangle = d_n\}.$$

If $\|\boldsymbol{K}_n\|_{\max} \leq \delta$, the following modifications are required:

- $\boldsymbol{K}_n \to \bar{\boldsymbol{K}}_n := [\boldsymbol{K}_n^{\mathsf{T}} \ \bar{\boldsymbol{k}}_n]^{\mathsf{T}} \qquad \bar{\boldsymbol{k}}_n := [\kappa_1(\boldsymbol{u}_n, \boldsymbol{u}_n), \kappa_2(\boldsymbol{u}_n, \boldsymbol{u}_n), \cdots, \kappa_M(\boldsymbol{u}_n, \boldsymbol{u}_n)]^{\mathsf{T}}$
- $\boldsymbol{H}_n \to \bar{\boldsymbol{H}}_n := [\boldsymbol{H}_n^{\mathsf{T}} \ \boldsymbol{0}_M]^{\mathsf{T}}$

MKNLMS-BT Algorithm — Rough Idea

- Unlike MKNLMS-CS, it has no admission control.
  Namely, all the data are inserted temporarily.

MKNLMS-BT Algorithm — Rough Idea

- Unlike MKNLMS-CS, it has no admission control.
  Namely, all the data are inserted temporarily.

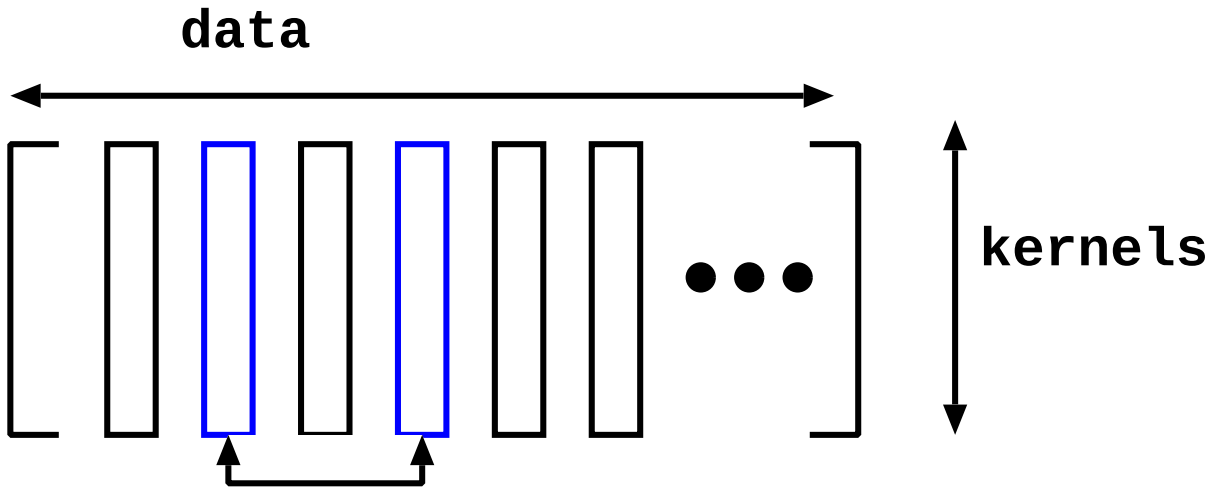
Step 1: The coefficient matrix $\boldsymbol{H}_n$ is augmented, and then updated by projection onto a bounded instantaneous-error hyperslab.

MKNLMS-BT Algorithm — Rough Idea

- Unlike MKNLMS-CS, it has no admission control.
  Namely, all the data are inserted temporarily.

Step 1: The coefficient matrix $\boldsymbol{H}_n$ is augmented, and then updated by projection onto a bounded instantaneous-error hyperslab.

Step 2: Such coefficients that have relatively minor contribution are removed **in blockwise fashion by block soft-thresholding**
$\Rightarrow$ **Computational efficiency** and **memory saving**

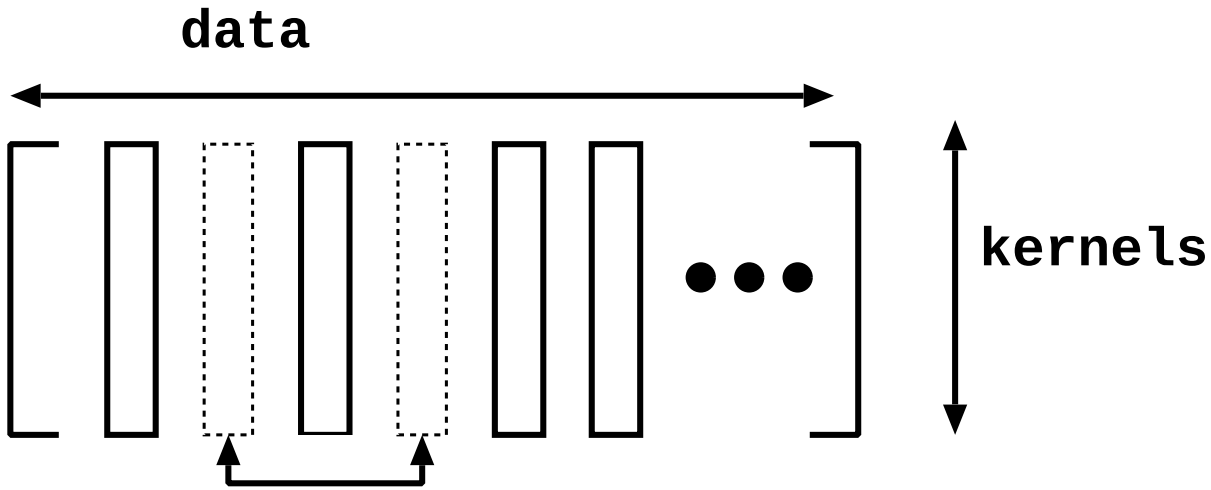


MKNLMS-BT Algorithm — Rough Idea

- Unlike MKNLMS-CS, it has no admission control.
  Namely, all the data are inserted temporarily.

Step 1: The coefficient matrix $\boldsymbol{H}_n$ is augmented, and then updated by projection onto a bounded instantaneous-error hyperslab.

Step 2: Such coefficients that have relatively minor contribution are removed **in blockwise fashion by block soft-thresholding**
$\Rightarrow$ **Computational efficiency** and **memory saving**

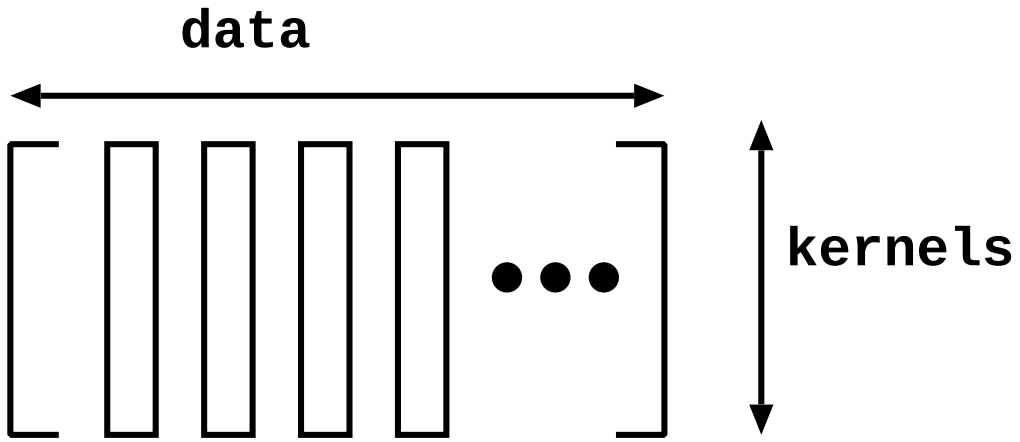


MKNLMS-BT Algorithm — Rough Idea

- Unlike MKNLMS-CS, it has no admission control.
  Namely, all the data are inserted temporarily.

Step 1: The coefficient matrix $\boldsymbol{H}_n$ is augmented, and then updated by projection onto a bounded instantaneous-error hyperslab.

Step 2: Such coefficients that have relatively minor contribution are removed **in blockwise fashion by block soft-thresholding**
$\Rightarrow$ **Computational efficiency** and **memory saving**



Move over!

MKNLMS-BT Algorithm — Derivation

- Cost function (time-varying):

$$\Theta_n(\boldsymbol{X}) := \underbrace{\frac{1}{2}d^2(\boldsymbol{X}, C_n)}_{=:\,\Theta_n^{(1)}(\boldsymbol{X})} + \underbrace{\lambda \sum_{i=1}^{r_n} w_{i,n} \|\boldsymbol{x}_i\|}_{=:\,\Theta_n^{(2)}(\boldsymbol{X})}, \quad n \in \mathbb{N}$$

$$\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \cdots \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M \times r_n}$$

  where

  $C_n := \{\boldsymbol{X} \in \mathbb{R}^{M \times r_n} : |\varepsilon_n(\boldsymbol{X})| \leq \epsilon\}$ (bounded instantaneous-error hyperslab).

  Here

  - the estimation-error function $\varepsilon_n(\boldsymbol{X}) := \langle \boldsymbol{X}, \boldsymbol{K}_n \rangle - d_n, \ \boldsymbol{X} \in \mathbb{R}^{M \times r_n}$
  - the error bound $\epsilon \geq 0$

- $\Theta_n^{(2)}$ is nondifferentiable but **"proximable"**

- Adaptive Proximal Forward-Backward Splitting Method [Murakami *et al.*, ICASSP 2010]

$$\widetilde{\boldsymbol{H}}_{n+1} := \mathrm{prox}_{\mu\Theta_n^{(2)}} \left(\boldsymbol{H}_n - \mu \boldsymbol{\nabla} \Theta_n^{(1)}(\boldsymbol{H}_n) \right), \quad n \in \mathbb{N}, \quad \mu \in (0, 2)$$

Proximity Operator

The proximity operator of $\Theta_n^{(2)}$ of index μ (Moreau '62):

$$\begin{aligned}\mathrm{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{X}) &:= \operatorname*{argmin}_{\boldsymbol{Y}\in\mathbb{R}^{M\times r_n}} \Theta_n^{(2)}(\boldsymbol{Y}) + \frac{1}{2\mu}\|\boldsymbol{X}-\boldsymbol{Y}\| \\ &= \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{x}_i\|}, 0\right\} \boldsymbol{x}_i \boldsymbol{e}_{i,n}^{\mathsf{T}} \qquad \textbf{(block soft-thresholding)} \\ \boldsymbol{X} &:= [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M\times r_n}\end{aligned}$$

Proximity Operator

The proximity operator of $\Theta_n^{(2)}$ of index μ (Moreau '62):

$$\begin{aligned}\operatorname{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{X}) &:= \operatorname*{argmin}_{\boldsymbol{Y}\in\mathbb{R}^{M\times r_n}} \Theta_n^{(2)}(\boldsymbol{Y}) + \frac{1}{2\mu}\|\boldsymbol{X}-\boldsymbol{Y}\| \\ &= \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{x}_i\|}, 0\right\} \boldsymbol{x}_i \boldsymbol{e}_{i,n}^{\mathsf{T}} \qquad \textbf{(block soft-thresholding)} \\ \boldsymbol{X} &:= [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M\times r_n}\end{aligned}$$

$$\widetilde{\boldsymbol{H}}_{n+1} := \operatorname{prox}_{\mu\Theta_n^{(2)}} \underbrace{\left(\boldsymbol{H}_n - \mu\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n)\right)}_{=:\,[\boldsymbol{g}_{1,n}\ \boldsymbol{g}_{2,n}\ \cdots\ \boldsymbol{g}_{r_n,n}]}, \qquad n\in\mathbb{N}, \qquad \mu\in(0,2)$$

Proximity Operator

The proximity operator of $\Theta_n^{(2)}$ of index μ (Moreau '62):

$$\begin{aligned}\operatorname{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{X}) &:= \operatorname*{argmin}_{\boldsymbol{Y}\in\mathbb{R}^{M\times r_n}} \Theta_n^{(2)}(\boldsymbol{Y}) + \frac{1}{2\mu}\|\boldsymbol{X}-\boldsymbol{Y}\| \\ &= \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{x}_i\|}, 0\right\} \boldsymbol{x}_i \boldsymbol{e}_{i,n}^{\mathsf{T}} \qquad \textbf{(block soft-thresholding)} \\ \boldsymbol{X} &:= [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M\times r_n}\end{aligned}$$

$$\widetilde{\boldsymbol{H}}_{n+1} := \operatorname{prox}_{\mu\Theta_n^{(2)}} \underbrace{\left(\boldsymbol{H}_n - \mu\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n)\right)}_{=:\,[\boldsymbol{g}_{1,n}\ \boldsymbol{g}_{2,n}\ \cdots\ \boldsymbol{g}_{r_n,n}]}, \qquad n\in\mathbb{N}, \qquad \mu\in(0,2)$$

1. Projection onto C_n: $\quad \boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n) = \boldsymbol{H}_n - P_{C_n}(\boldsymbol{H}_n)$

Proximity Operator

The proximity operator of $\Theta_n^{(2)}$ of index μ (Moreau '62):

$$\begin{aligned}\operatorname{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{X}) &:= \operatorname*{argmin}_{\boldsymbol{Y}\in\mathbb{R}^{M\times r_n}} \Theta_n^{(2)}(\boldsymbol{Y}) + \frac{1}{2\mu}\|\boldsymbol{X}-\boldsymbol{Y}\| \\ &= \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{x}_i\|}, 0\right\} \boldsymbol{x}_i \boldsymbol{e}_{i,n}^{\mathsf{T}} \qquad \textbf{(block soft-thresholding)} \\ \boldsymbol{X} &:= [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M\times r_n}\end{aligned}$$

$$\widetilde{\boldsymbol{H}}_{n+1} := \operatorname{prox}_{\mu\Theta_n^{(2)}} \underbrace{\left(\boldsymbol{H}_n - \mu\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n)\right)}_{=:\,[\boldsymbol{g}_{1,n}\ \boldsymbol{g}_{2,n}\ \cdots\ \boldsymbol{g}_{r_n,n}]}, \qquad n\in\mathbb{N}, \qquad \mu\in(0,2)$$

1. Projection onto C_n: $\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n) = \boldsymbol{H}_n - P_{C_n}(\boldsymbol{H}_n)$
2. Block soft-thresholding $\operatorname{prox}_{\mu\Theta_n^{(2)}}$

Proximity Operator

The proximity operator of $\Theta_n^{(2)}$ of index μ (Moreau '62):

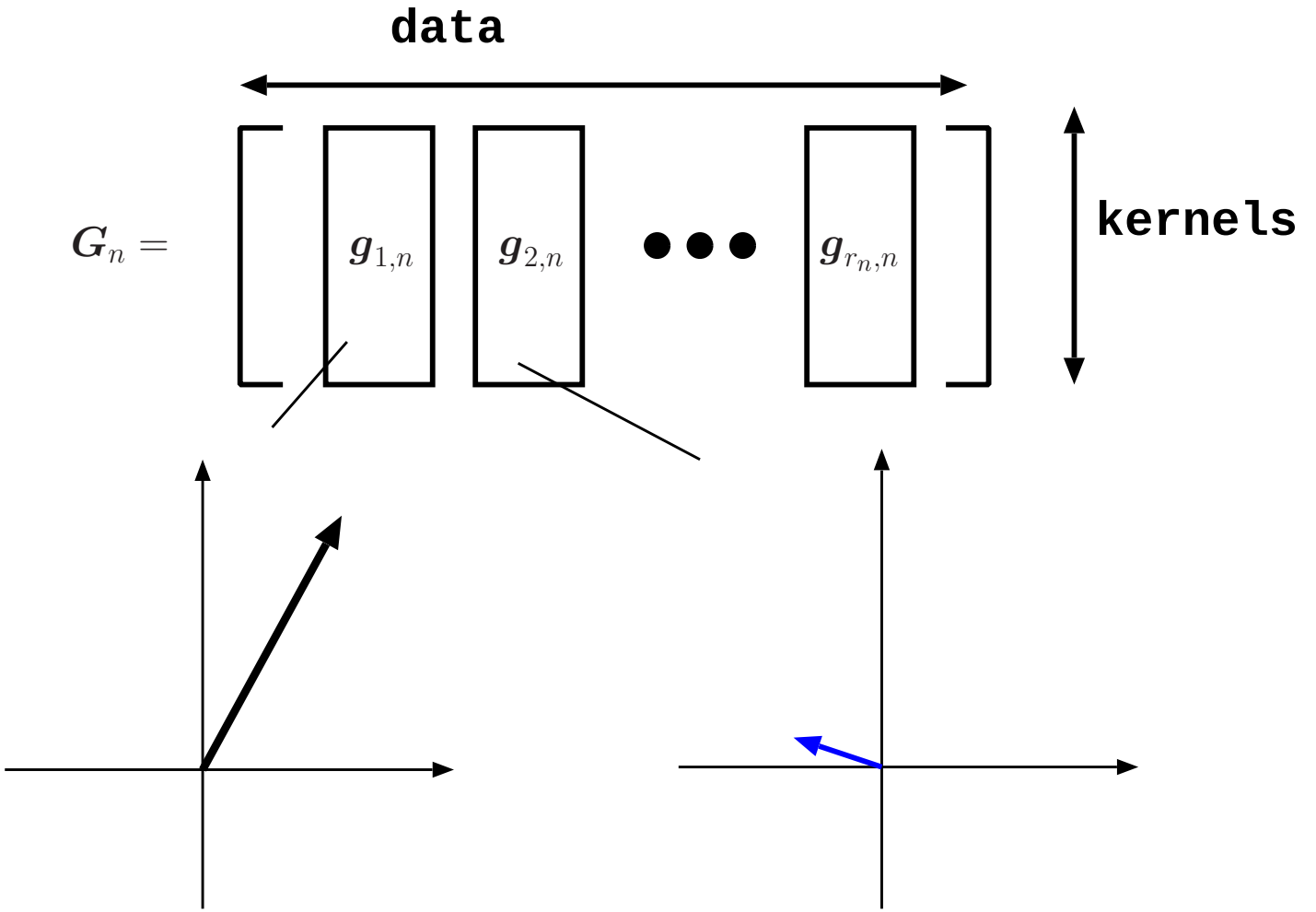
$$\begin{aligned}\operatorname{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{X}) &:= \operatorname*{argmin}_{\boldsymbol{Y}\in\mathbb{R}^{M\times r_n}} \Theta_n^{(2)}(\boldsymbol{Y}) + \frac{1}{2\mu}\|\boldsymbol{X}-\boldsymbol{Y}\| \\ &= \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{x}_i\|}, 0\right\} \boldsymbol{x}_i \boldsymbol{e}_{i,n}^{\mathsf{T}} \qquad \textbf{(block soft-thresholding)} \\ \boldsymbol{X} &:= [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_{r_n}] \in \mathbb{R}^{M\times r_n}\end{aligned}$$

$$\widetilde{\boldsymbol{H}}_{n+1} := \operatorname{prox}_{\mu\Theta_n^{(2)}} \underbrace{\left(\boldsymbol{H}_n - \mu\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n)\right)}_{=:\,[\boldsymbol{g}_{1,n}\ \boldsymbol{g}_{2,n}\ \cdots\ \boldsymbol{g}_{r_n,n}]}, \qquad n\in\mathbb{N}, \qquad \mu\in(0,2)$$

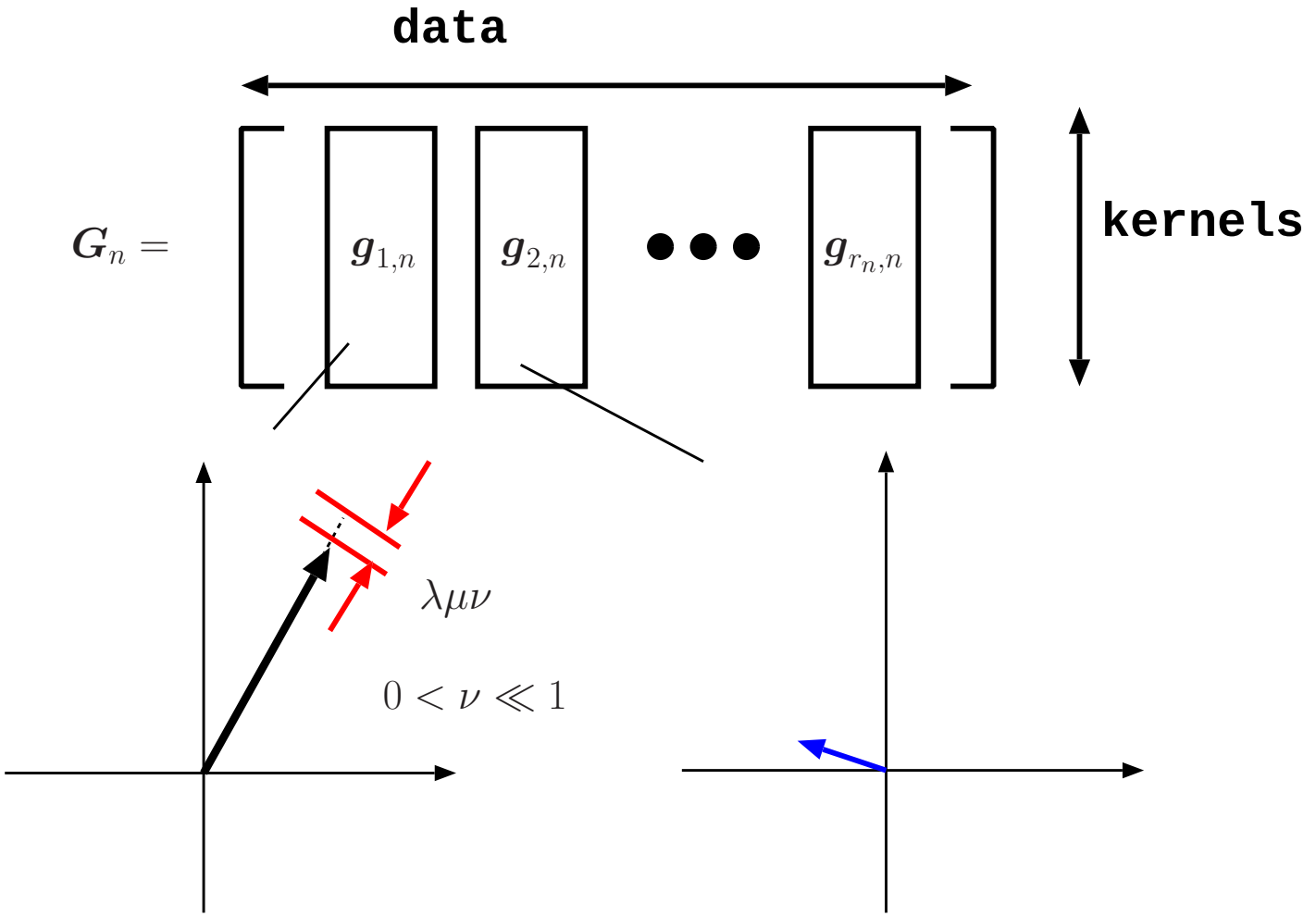
1. Projection onto C_n: $\boldsymbol{\nabla}\Theta_n^{(1)}(\boldsymbol{H}_n) = \boldsymbol{H}_n - P_{C_n}(\boldsymbol{H}_n)$
2. Block soft-thresholding $\operatorname{prox}_{\mu\Theta_n^{(2)}}$
3. Remove the zero row vectors and then move over

   $\mathcal{J}_{-1}^{\mathrm{BT}} := \emptyset$ and $\mathcal{J}_n^{\mathrm{BT}} := \{j\in\mathcal{J}_{n-1}^{\mathrm{BT}} : \ \boldsymbol{h}_{j,n}\neq\boldsymbol{0}\}\cup\{n\}$

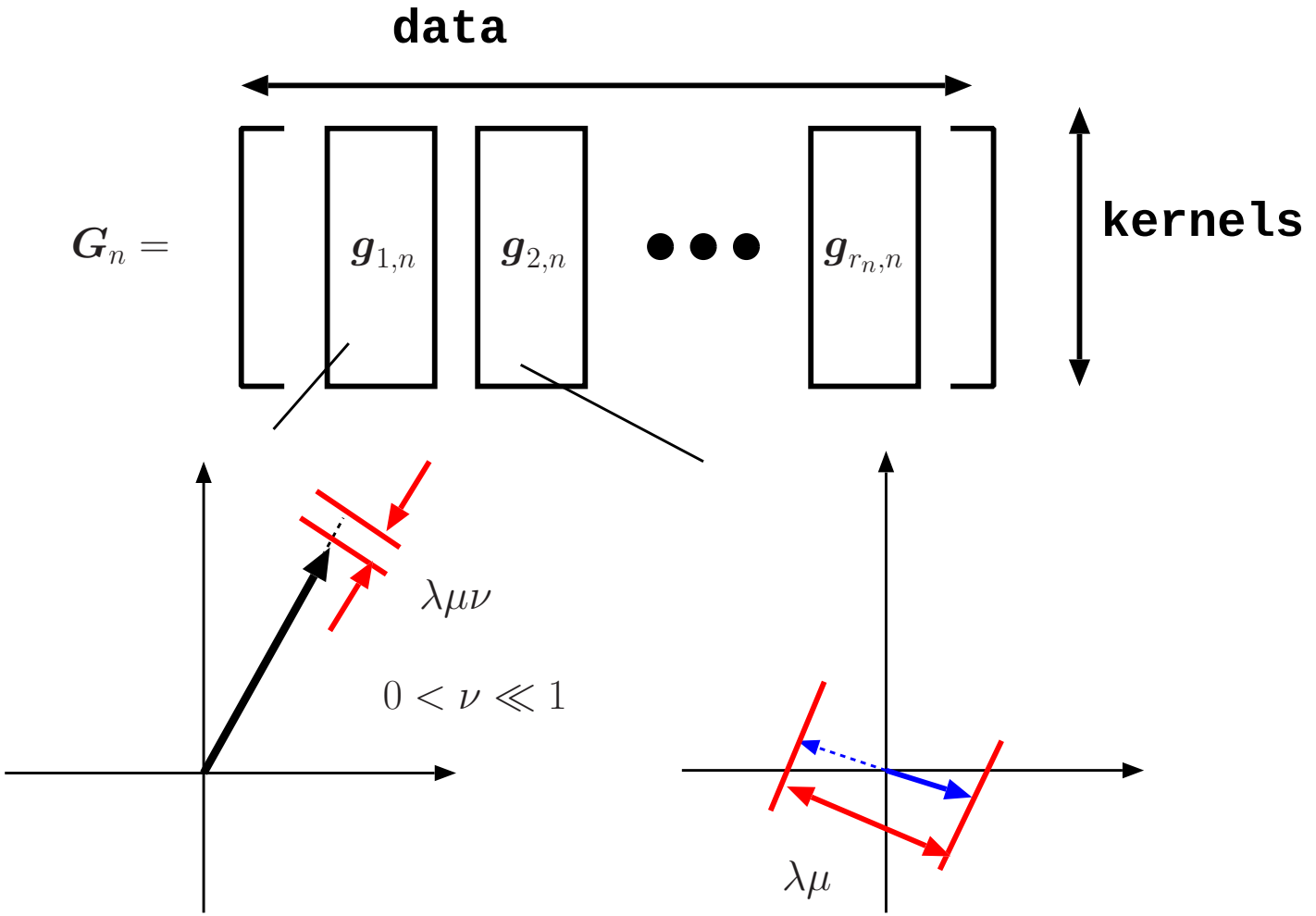
Block Soft-Thresholding



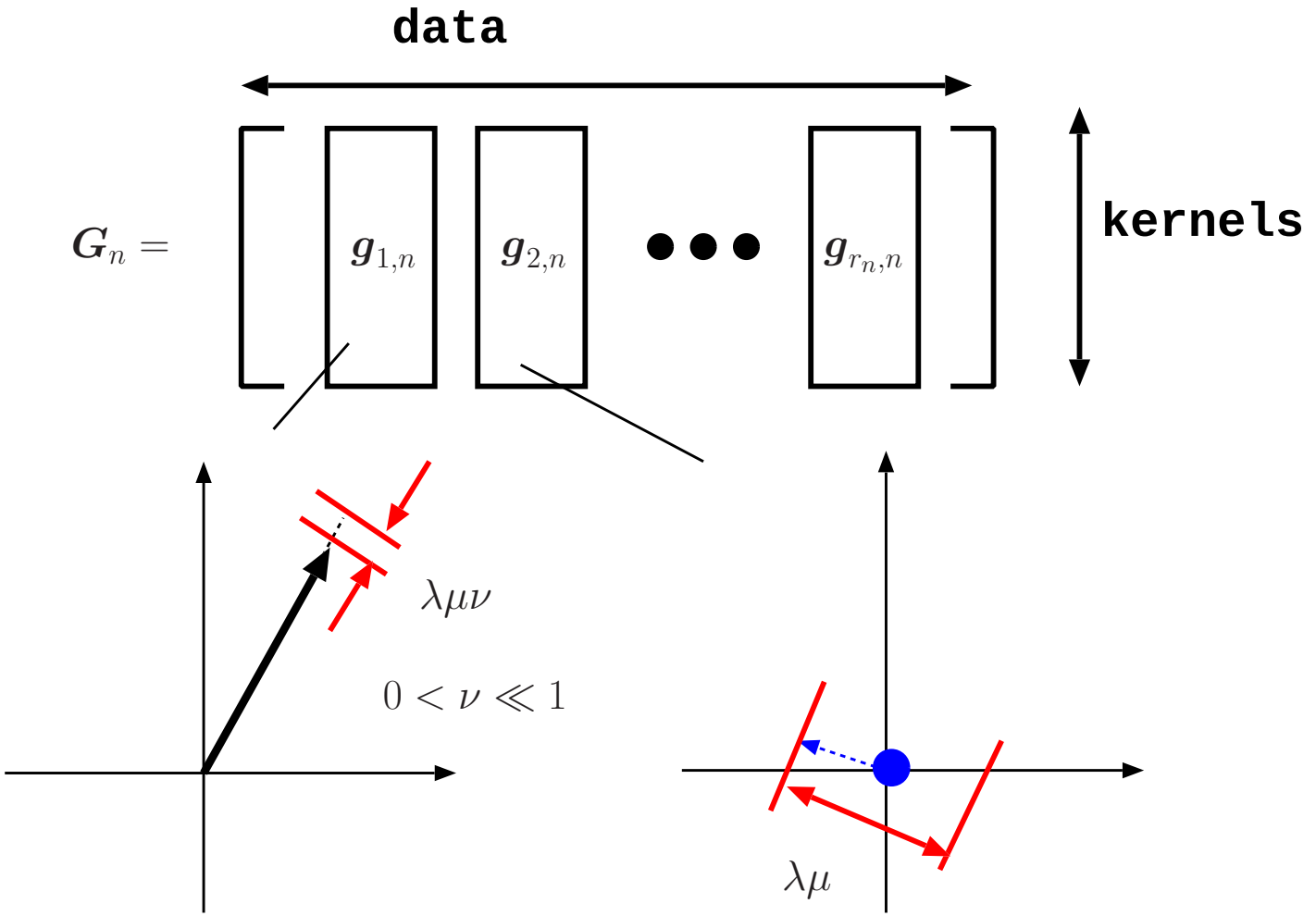
Block Soft-Thresholding



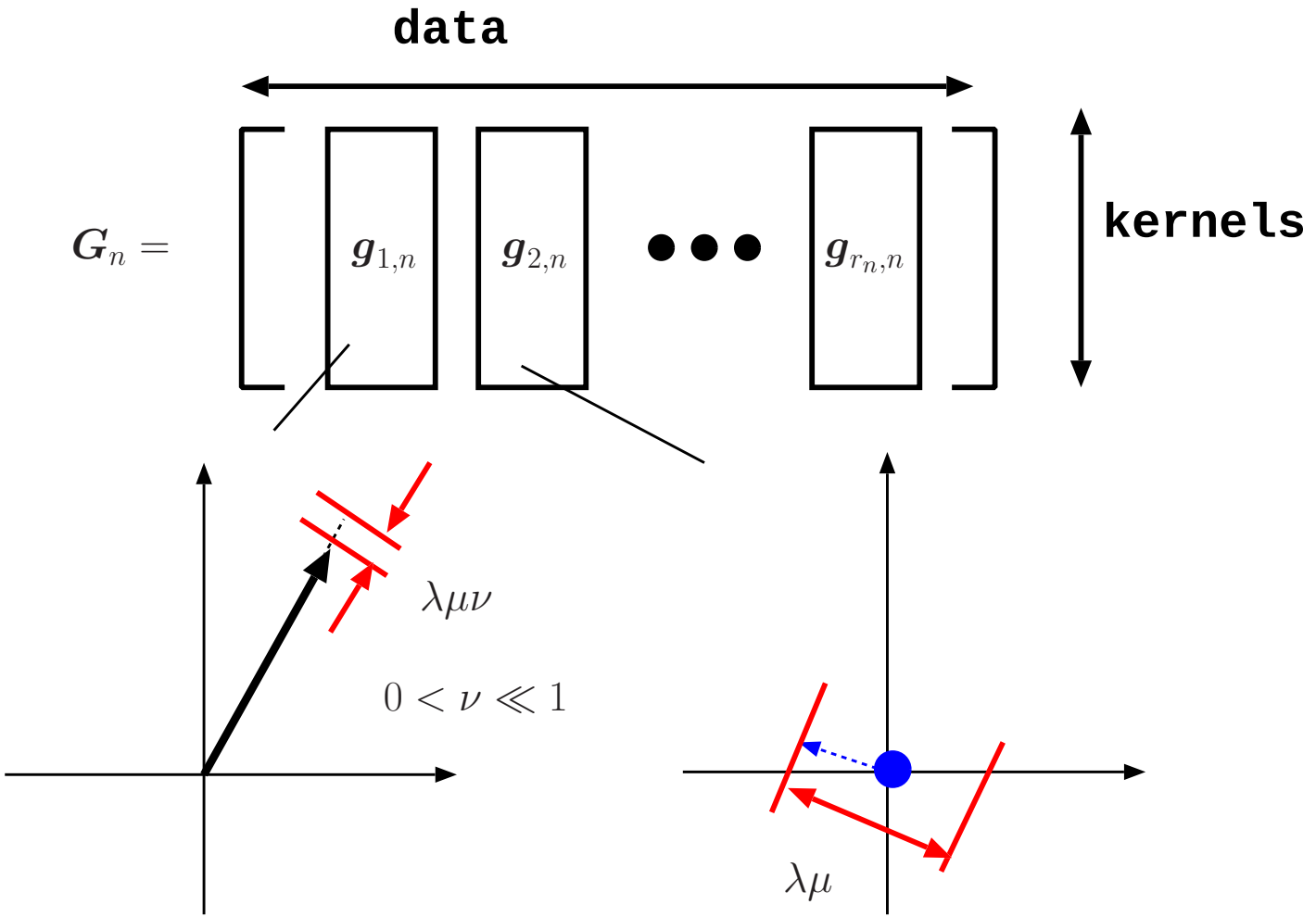
Block Soft-Thresholding



Block Soft-Thresholding



Block Soft-Thresholding



- $\operatorname{prox}_{\mu\Theta_n^{(2)}}(\boldsymbol{G}_n) = \sum_{i=1}^{r_n} \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\|\boldsymbol{g}_{i,n}\|}, 0\right\} \boldsymbol{g}_{i,n}\boldsymbol{e}_{i,n}^{\mathsf{T}}, \qquad w_{i,n} := \begin{cases} \nu, & \text{if } \|\boldsymbol{g}_{i,n}\| > \tau \\ 1, & \text{otherwise} \end{cases}$

Computational Complexity

	KNLMS	MKNLMS-CS	MKNLMS-BT
Multiplication	$(L+3)r_n$	$(L+3M)r_n$	$(L+5M)r_n$
Exponential	r_n	Mr_n	Mr_n
Memory	$(L+1)r_n$	$(L+M)r_n$	$(L+M)r_n$

- L: Dim. of the input vector $\boldsymbol{u}_n$
- M: # of kernels
- r_n: Dictionary size at the nth iteration

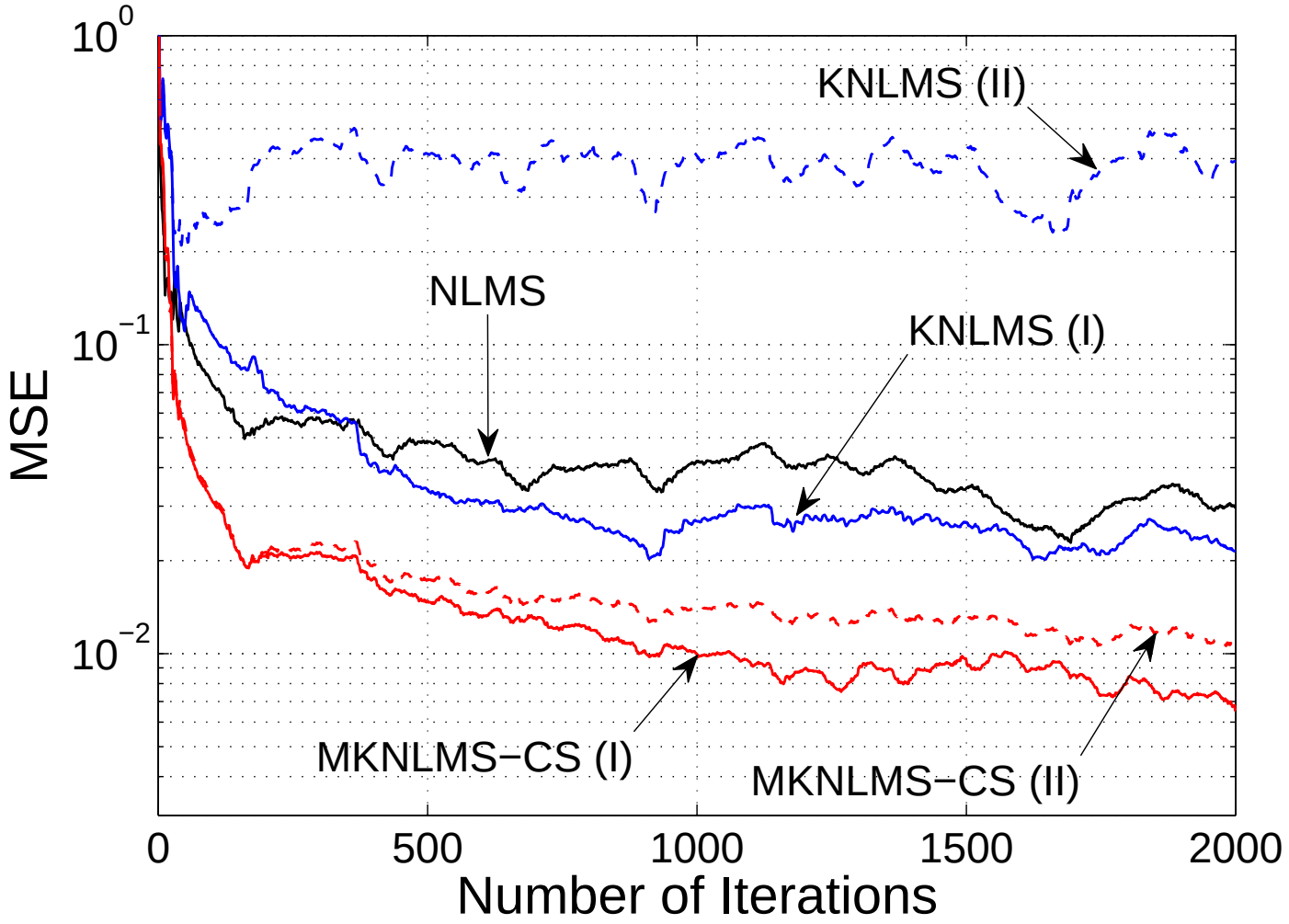
The complexity of the proposed algorithm is low!

Numerical Example A

- Online Prediction of the following time-series data
  - Mackey-Glass chaotic time series characterized by the following differential equation: $dx(t)/dt = -0.1x(t) + 0.2x(t-30)/\left[1 + x(t-30)^{10}\right]$.
- The signals are corrupted by additive Gaussian noise with zero mean and standard deviation 0.04.
- d_n is predicted with $\boldsymbol{u}_n := [d_{n-1}, d_{n-2}, \cdots, d_{n-L}]^{\mathsf{T}}$ $(L = 10)$
- NLMS: $\eta = 2.0 \times 10^{-2}$
- KNLMS: $\eta = 0.2,\ \rho = 3.0 \times 10^{-2},\ \zeta = 3.5$
- MKNLMS-CS: $\eta = 0.2,\ \rho = 6.0 \times 10^{-2},\ \zeta_1 = 0.5,\ \zeta_2 = 6.5,$ (I) $\delta = 0.5,$ (II) $\delta = 0.8$

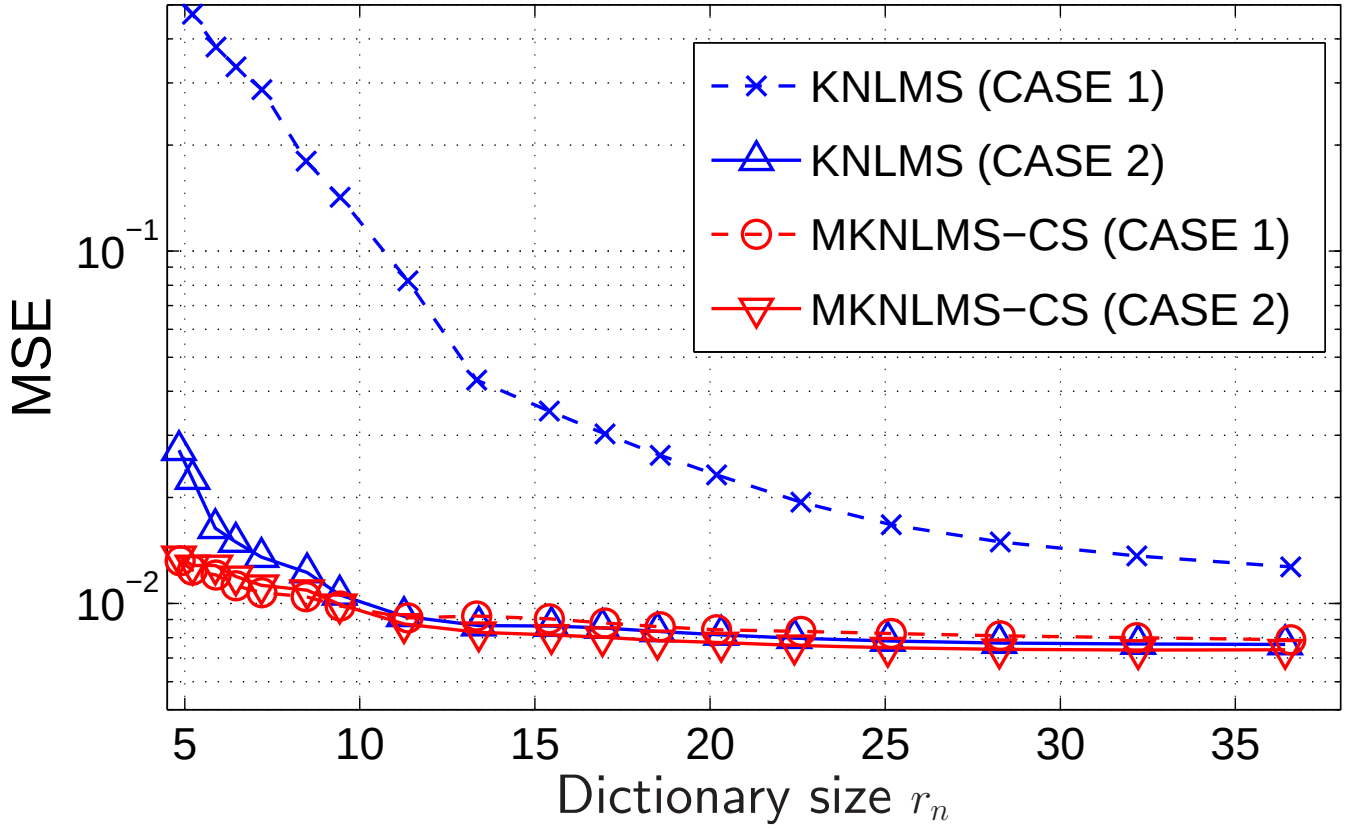
NOTE: Good kernel parameter is $\zeta = 1.0$

MSE Learning Curve



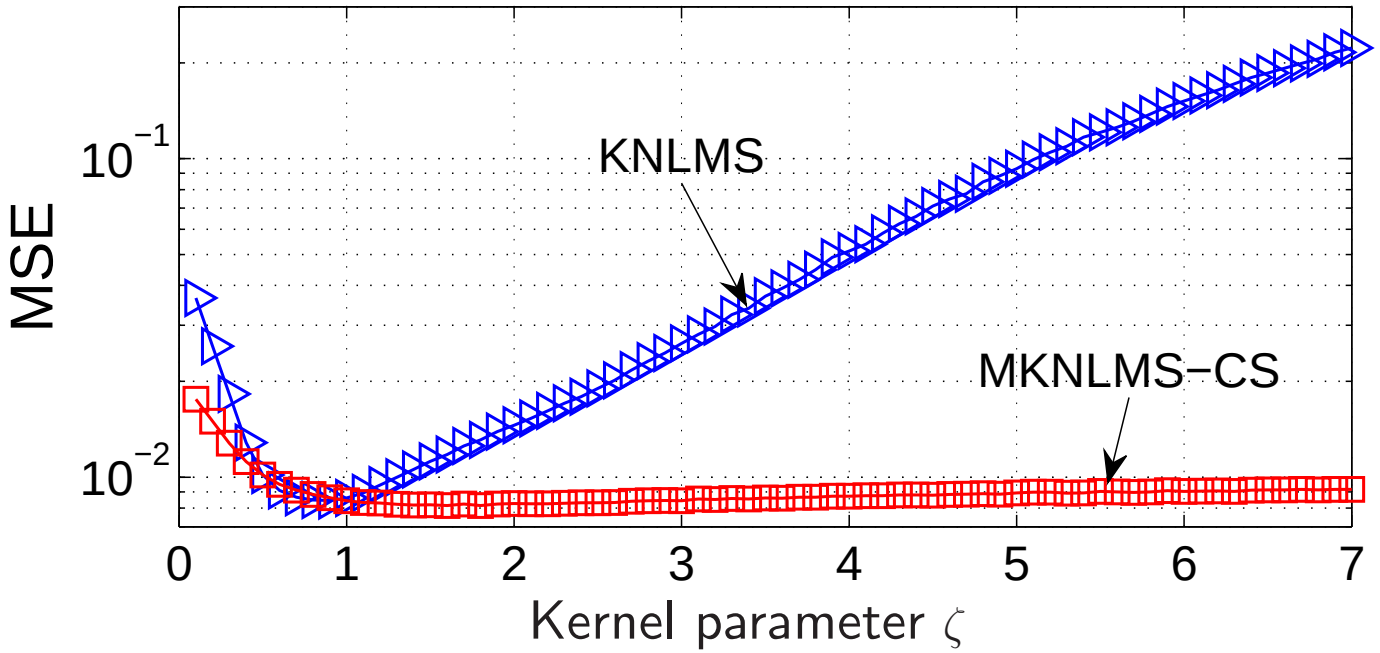
The average dictionary sizes: (I) $\bar{r} = 18.7$ and (II) $\bar{r} = 6.0$

Impact of Dictionary Size on MSE



- Case 1: a good kernel parameter is unavailable
- Case 2: a good kernel parameter is available

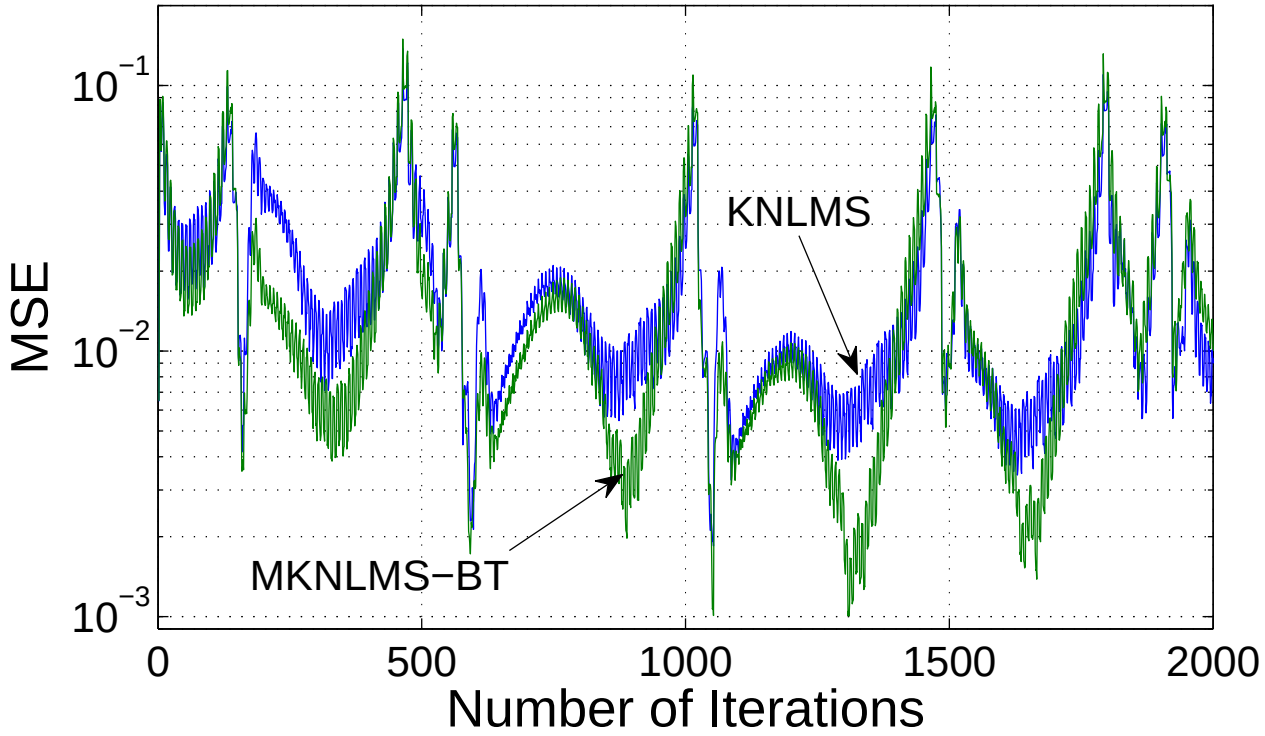
Impact of Kernel Parameter on MSE



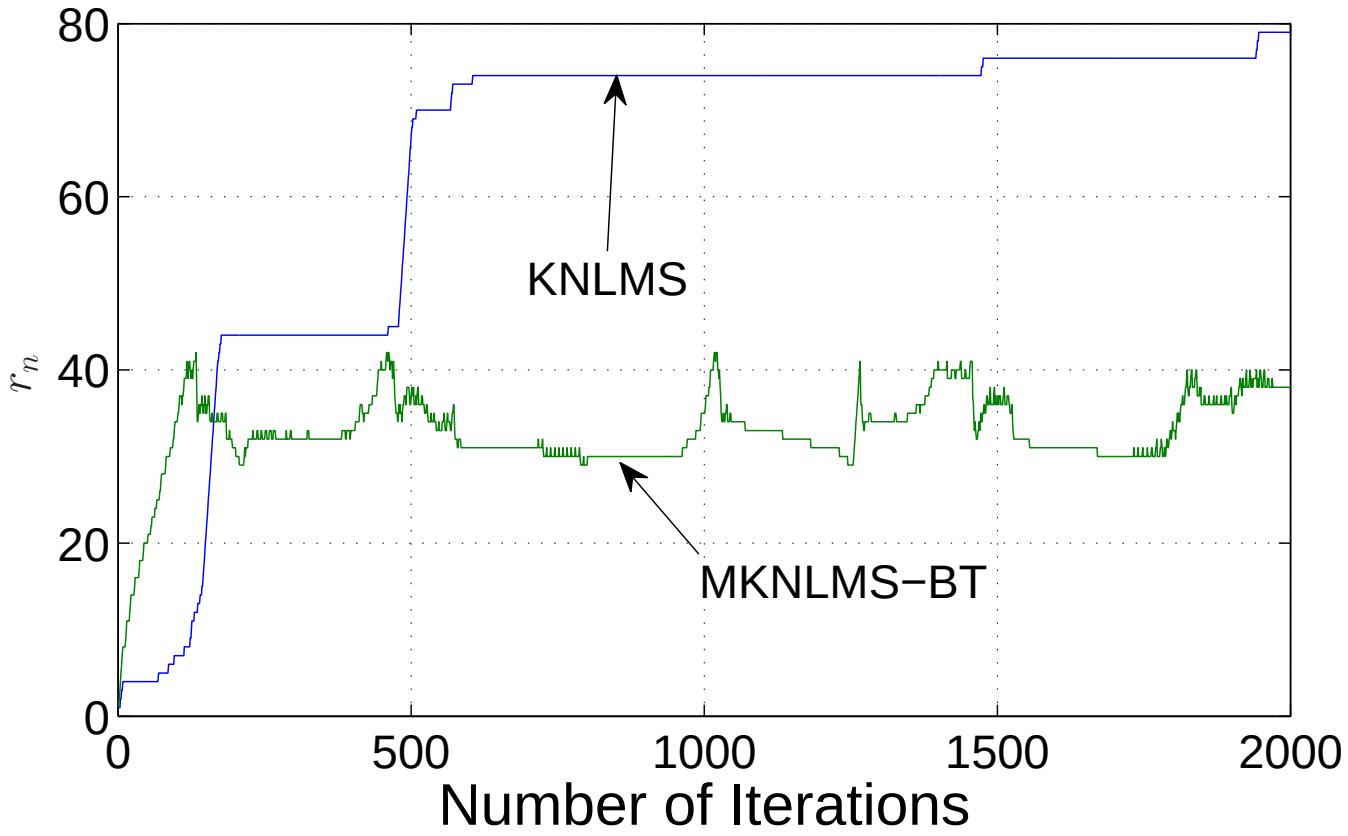
The average dictionary sizes: (a) $\bar{r} = 10.1$ and (b) $\bar{r} = 15.1$.

Numerical Example B

- Online Prediction of the chaotic laser time series from the Santa Fe time series competition $(L = 40)$
- KNLMS: $\eta = 0.1,\ \rho = 3.0 \times 10^{-2},\ \zeta = 1/2\sigma^2$ with $\sigma := 0.9,\ \delta = 0.476$
- MKNLMS-BT: $\mu = 0.1,\ \zeta_1 = 0.4,\ \zeta_2 = 3.0,\ \epsilon = 0,\ \lambda = 3.0 \times 10^{-2}$



Evolution of Dictionary Size



- Complexity and memory usage
  - KNLMS: 3215, 2996
  - MKNLMS-BT: 1775, 1433 (More efficient due to smaller r_n)

Advantages of Multikernel Approach

1. The use of multiple kernels brings a considerable gain either (i) in the case that the dictionary size is not sufficiently large or (ii) in the case that an inappropriate kernel is employed. From another viewpoint, the multikernel approach is relatively insensitive to the choice of parameters determining the dictionary size and the kernels themselves. This suggests that the multikernel approach only requires rough tuning of the parameters, and hence it can alleviate the computational burdens for finding good values of the parameters.

2. The multikernel approach significantly outperforms the single-kernel approach when applied to nonstationary data, even though the parameters for the single-kernel approach are carefully tuned. It should be mentioned that good parameter values could be time-dependent due to the nonstationarity of data.

Advantages of Multikernel Approach (Cont.)

3. The multikernel approach could attain comparable, or even better, performance with a smaller value of r_n compared to the single-kernel approach. When the input-space dimension L is large, the complexity and memory usage of the multikernel approach could be lower than that of the single-kernel approach.

In addition, the major advantage of MKNLMS-BT over MKNLMS-CS is given as follows.

4. For nonstationary data, the MKNLMS-BT algorithm attains reasonable performance $with\ a\ fairly\ small\ dictionary\ size$, while the KNLMS and MKNLMS-CS algorithms increase the dictionary size considerably as time goes by. As a result, the complexity of MKNLMS-BT could be lower than those of KNLMS and MKNLMS-CS.

Conclusion

1. This paper has presented a study on the multikernel approach to nonlinear adaptive filtering.
2. A pair of fully adaptive algorithms using multiple kernels have been proposed.
3. The MKNLMS-CS algorithm exploits the coherence-based criterion for dictionary designing, while the MKNLMS-BT algorithm exploits the weighted block soft-thresholding operator.
4. The remarkable advantages of the proposed algorithms have been demonstrated, including
   (a) insensitivity to the choice of the kernel parameters (as well as the coherence threshold) and
   (b) significant performance gains and potential computational/ memory efficiency for nonstationary data.

Open Issues

- Analyzing the performance of the approach and proving theoretically that it achieves low estimation error with a smaller number of center points.
- Finding killer apps.

- Published article:
  M. Yukawa, "Multikernel adaptive filtering", *IEEE Trans. Signal Processing*, vol. 60, no. 9, pp. 4672–4682, September 2012.
- References for MKNLMS-BT:

  1. P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting", *SIAM Journal on Multiscale Modeling and Simulation*, vol. 4, pp. 1168–1200, 2005.
  2. Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada, "A sparse adaptive filtering using time-varying soft-thresholding techniques," *in Proc. IEEE ICASSP*, pp.3734–3737, 2010.

- More recent results:
  M. Yukawa and R. Ishii, "An efficient kernel adaptive filtering algorithm using hyperplane projection along affine subspace", *European Signal Processing Conference (EUSIPCO) 2012*, to be presented.

Thank you for your kind attention!