AUTOMATIC KERNEL WEIGHTING FOR MULTIKERNEL ADAPTIVE FILTERING: MULTISCALE ASPECTS

Kwangjin Jeong†, *Masahiro Yukawa*†‡

† Dept. of Electronics and Electrical Engineering, Keio University, JAPAN
‡ Center for Advanced Intelligence Project, RIKEN, JAPAN

ABSTRACT

This paper presents an automatic kernel weighting technique for multikernel adaptive filtering. The full potential of the multikernel adaptive filtering approach can only be achieved when the kernels are weighted appropriately. The proposed technique balances the dominance of the kernels by making the mean eigenvalues of their associated autocorrelation matrices be equal to each other. The overall complexity of the proposed approach is low because the mean eigenvalues can be computed efficiently. The numerical results verify that the proposed technique balances the coefficient updates and yields reasonable performance.

Index Terms— kernel adaptive filtering, reproducing kernel Hilbert space (RKHS), parameter tuning, multiscale analysis

1. INTRODUCTION

Signals encountered in the engineering often include multiple components with different scales. As we introduce later, there exist two types of the *multiscaleness*, which are related to the target function to be estimated and the sample data. Such multiscaleness has been witnessed in many engineering applications including biomedical engineering, material engineering and data analysis [1–7], as well as image and acoustic signal processing. Multikernel adaptive filtering [8–14] is a promising approach for online estimation of a multiscale function, since it considers multiple reproducing kernel Hilbert spaces simultaneously. Using multiple kernels, an adaptive filtering algorithm may also be more effective in both senses of computational complexity and memory usage than that using only a single kernel. Several criteria [15–18] can be employed to prevent unnecessary growth of the dictionaries and achieve reasonable dictionary size. It has been reported that multikernel adaptive filtering works efficiently in some applications including communication [19, 20], sensor network [21] and robotics [22].

Since any positive scaling on a reproducing kernel generates another reproducing kernel, one can also consider multikernel adaptive filtering with "weighted" kernels. With appropriately weighted reproducing kernels, a multikernel adaptive filtering algorithm can achieve both of higher accuracy and smaller dictionary than the case using unweighted kernels. However, inappropriate weights on kernels cause unbalance among updates of the coefficients corresponding to those kernels and consequently fail to achieve desirable performance. In the worst case, the performance would be worse than when we use only a single kernel. It becomes more difficult to impose proper weights on kernels as we have larger number of, or more kinds of reproducing kernels to weight. The use of inappropriately weighted kernels may cause considerable differences among convergence speeds of the coefficients corresponding to the kernels. More specifically, it may happen that the coefficients corresponding to some of the kernels grow fast while those corresponding to the others hardly grow. It is of significant importance to avoid such situations by developing a well-designed kernel weighting technique for exploiting the full potential of the multikernel adaptive filtering.

This work was supported by JSPS KAKENHI Grant Numbers JP18J22032, JP18H01446.

In this paper, we propose an automatic kernel weighting technique for a multikernel adaptive filtering algorithm to achieve reasonably high accuracy and small dictionary size simultaneously. We mainly consider multikernel adaptive filtering with the multiple Gaussian kernels in this paper, since we can deal with the multiscaleness simply by only considering a single parameter. Among many multikernel adaptive filtering algorithms [8–13], we focus on the multikernel normalized least mean squares (MKNLMS) algorithm with the coherence criterion [18] and Platt's criterion [15]. In the algorithm, we consider a concatenated vector of the kernelized input vectors with multiple kernels and the autocorrelation matrix of that concatenated vector. The proposed automatic weighting technique aims to adjust the coefficient updates corresponding to reproducing kernels in a good balance. On developing the proposed technique, we focus on the submatrices of the autocorrelation matrix. If one submatrix is dominant, the corresponding coefficients will be updated in a large amplitude, whereas the other coefficients hardly change. To evaluate such dominance of each submatrix on update of the corresponding coefficients, we exploit its mean eigenvalue. The mean eigenvalue of each submatrix can be calculated effectively, since the sum of eigenvalues is obtained by averaging the squared ℓ_2-norm of the corresponding kernelized input vector. At each iteration, the weights are adjusted to balance the mean eigenvalues of submatrices. The proposed kernel weighting technique can be implemented by using either (i) the weighted reproducing kernels directly, or (ii) the weights to design a metric distance. The two implementations are equivalent in the sense of mean squared error (MSE) when the weights are fixed, while that is not guaranteed if the weights are time-varying. Numerical examples using two Gaussian kernels are presented to verify the performance of the proposed automatic kernel weighting technique.

2. MULTIKERNEL ADAPTIVE FILTERING AND MULTISCALENESS

Consider the following nonlinear adaptive filtering model:

$$d_n = \psi^*(\boldsymbol{u}_n) + \nu_n, \tag{1}$$

where $\boldsymbol{u}_n \in \mathbb{R}^L$ is the input vector, y_n is the corresponding output, ν_n is the unknown additive noise, and ψ^* is an unknown function to be estimated. Typical kernel adaptive filtering schemes aim to approximate

$$\psi^*(\cdot) \approx \sum_{j=1}^{r} \alpha_j \kappa(\boldsymbol{x}_j, \cdot), \tag{2}$$

where $\kappa(\cdot,\cdot)$ is a positive definite kernel function, and r is the cardinality of a dictionary $\mathcal{D} := \{\kappa(\boldsymbol{x}_1,\cdot), \cdots, \kappa(\boldsymbol{x}_r,\cdot)\}$. One of the representative examples is a normalized Gaussian kernel, which is defined as

$$\kappa_G(\boldsymbol{x},\boldsymbol{u}) = \frac{1}{(\sqrt{2\pi}\sigma)^L} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{u}\|_2^2}{2\sigma^2}\right) \tag{3}$$

for $\sigma > 0$. A dictionary with sufficiently large cardinality needs for small error, but an extremely large dictionary causes heavy computational burden. Several criteria [15–18] have been proposed in order

to construct a dictionary with reasonable size. Multikernel adaptive filtering schemes utilize multiple kernels simultaneously so that the approximation becomes

$$\psi^*(\cdot) \approx \sum_{q=1}^{Q} \sum_{j=1}^{r^{(q)}} \alpha_j^{(q)} \kappa^{(q)}(\boldsymbol{x}_j^{(q)}, \cdot), \tag{4}$$

where $\kappa^{(q)}(\cdot,\cdot)$ is the qth kernel function, and $r^{(q)}$ is the cardinality of corresponding dictionary $\mathcal{D}^{(q)}$. Yukawa [8, 11] has proposed the MKNLMS algorithm which estimates $\psi^*(\cdot)$ by applying NLMS scheme with the kernelized input. At each iteration, the MKNLMS algorithm updates its estimate as follows:

1. For all $q = 1, 2, \cdots, Q$, calculate the kernelized input vector associated with qth kernel function as follows:
$$\boldsymbol{k}_n^{(q)} = \left[\kappa^{(q)}(\boldsymbol{x}_1^{(q)}, \boldsymbol{u}_n), \kappa^{(q)}(\boldsymbol{x}_2^{(q)}, \boldsymbol{u}_n), \cdots, \kappa^{(q)}(\boldsymbol{x}_{r^{(q)}}^{(q)}, \boldsymbol{u}_n)\right]^{\mathsf{T}}.$$
2. After constructing the dictionaries $\mathcal{D}_n^{(q)}$, $\forall q$, if $\{\kappa^{(q)}(\boldsymbol{u}_n, \cdot)\} \notin \mathcal{D}_{n-1}^{(q)}$ and $\mathcal{D}_n^{(q)} = \mathcal{D}_{n-1}^{(q)} \cup \{\kappa^{(q)}(\boldsymbol{u}_n, \cdot)\}$, $\tilde{\boldsymbol{\alpha}}_n^{(q)} = \left[(\boldsymbol{\alpha}_n^{(q)})^{\mathsf{T}}\ 0\right]^{\mathsf{T}}, \tilde{\boldsymbol{k}}_n^{(q)} = \left[(\boldsymbol{k}_n^{(q)})^{\mathsf{T}}\ 1\right]^{\mathsf{T}}$. Otherwise, $\tilde{\boldsymbol{\alpha}}_n^{(q)} = \boldsymbol{\alpha}_n^{(q)}$, $\tilde{\boldsymbol{k}}_n^{(q)} = \boldsymbol{k}_n^{(q)}$.
3. Concatenate the vectors as follows:
$$\tilde{\boldsymbol{\alpha}}_n = \left[(\tilde{\boldsymbol{\alpha}}_n^{(1)})^{\mathsf{T}}, (\tilde{\boldsymbol{\alpha}}_n^{(2)})^{\mathsf{T}}, \cdots, (\tilde{\boldsymbol{\alpha}}_n^{(Q)})^{\mathsf{T}}\right]^{\mathsf{T}},$$
$$\tilde{\boldsymbol{k}}_n = \left[(\tilde{\boldsymbol{k}}_n^{(1)})^{\mathsf{T}}, (\tilde{\boldsymbol{k}}_n^{(2)})^{\mathsf{T}}, \cdots, (\tilde{\boldsymbol{k}}_n^{(Q)})^{\mathsf{T}}\right]^{\mathsf{T}}.$$
4. Update the estimate $\boldsymbol{\alpha}$ as follows:
$$\boldsymbol{\alpha}_{n+1} = \tilde{\boldsymbol{\alpha}}_n - \mu \frac{\tilde{\boldsymbol{k}}_n^{\mathsf{T}} \tilde{\boldsymbol{\alpha}}_n - d_n}{\|\tilde{\boldsymbol{k}}_n\|_2^2} \tilde{\boldsymbol{k}}_n. \tag{5}$$

From (4), multikernel adaptive filtering can be seen as nonlinear approximation of $\psi^*(\cdot)$, which consists of different reproducing kernels $\kappa^{(q)}$. Below we introduce two types of multiscaleness.

Task-relevant multiscaleness: The target function consists of multiple components with different scales from each other (Fig. 1 (a)). One might approximate the function accurately with a single basis function. However, it will need a dictionary with extremely large size. In other words, it takes extremely long time to approximate the function. By using multiple basis functions with different scales, the approximation can be calculated in shorter time, even if the same or higher accuracy is required.

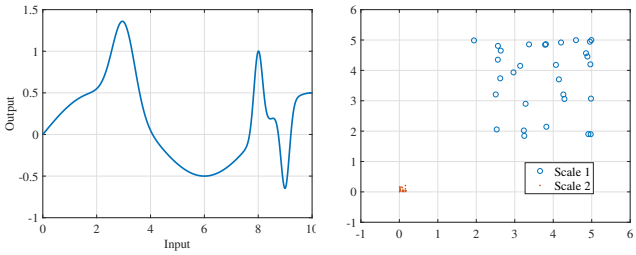
Sample-relevant multiscaleness: The distribution of the sample data in the input space has a multiscale characteristic. Suppose that we estimate an unknown system from sampled input data given as Fig. 2 (b) and corresponding output data. In this case, estimation with multiple basis function will be more accurate than that with a single basis function.

Throughout this paper, we mainly consider the case that all the kernel functions $\kappa^{(q)}(\cdot,\cdot)$, $(q = 1, 2, \cdots, Q)$ are Gaussian kernel defined as (3) with the scaling parameters $\sigma^{(1)}, \sigma^{(2)}, \cdots, \sigma^{(Q)}$, where we can deal with the multiscaleness by only considering those parameters.

3. AUTOMATIC KERNEL WEIGHTING FOR MULTIKERNEL ADAPTIVE FILTERING

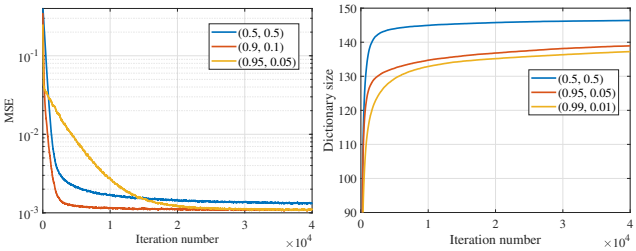
3.1. Weighting on Kernels

Given a positive definite kernel function κ and a constant $w > 0$, it is trivial that $\kappa_w(\cdot,\cdot) := w\kappa(\cdot,\cdot)$ is also a positive definite kernel



(a) Multiscaleness of function (b) Multiscaleness of data

Fig. 1. Two types of multiscaleness.



(a) MSE (b) Dictionary Evolution

Fig. 2. The MSE and dictionary evolutions under different weights.

function. Therefore, one can utilize weights in multikernel adaptive filtering and rewrite (4) as follows:

$$\begin{aligned}\psi^*(\cdot) &\approx \sum_{q=1}^{Q} \sum_{j=1}^{r^{(q)}} \alpha_j^{(q)} \kappa_{w^{(q)}}^{(q)}(\boldsymbol{x}_j^{(q)}, \cdot) \\ &= \sum_{q=1}^{Q} \sum_{j=1}^{r^{(q)}} \alpha_j^{(q)} w^{(q)} \kappa^{(q)}(\boldsymbol{x}_j^{(q)}, \cdot).\end{aligned} \tag{6}$$

This kind of weighting can be simply applied to multikernel adaptive filtering by replacing $\kappa^{(q)}(\cdot,\cdot)$ into $\kappa_{w^{(q)}}^{(q)}(\cdot,\cdot)$. Applying carefully chosen weights, one can attain small error while maintaining the dictionary sizes in an affordable range. However, an inappropriate choice of the weights causes large error without reasonable reduction of dictionary sizes.

Figs. 2 (a) and (b) show the MSE and the dictionary evolution of the MKNLMS algorithm with different weights for two Gaussian kernels. See the blue curve, which stands for the unweighted case. Both of the MSE and dictionary size are larger than the other cases. The weight setting shown as the red curve seems to achieve both of small MSE and reasonable dictionary size. On the other hand, the yellow curve shows the same MSE as the red curve with slightly smaller dictionaries, while its convergence speed is much slower than that of the red curve. More kernels we deal with, more difficult to find appropriate weights. To avoid such difficulty, we propose an automatic kernel weighting technique.

3.2. Automatic Kernel Weighting

The autocorrelation matrix of the (concatenated) kernelized input with unweighted kernels is given as follows:

$$\boldsymbol{K} := E[\boldsymbol{k}_n \boldsymbol{k}_n^{\mathsf{T}}] = \begin{bmatrix} \boldsymbol{K}^{(1,1)} & \boldsymbol{K}^{(1,2)} & \cdots & \boldsymbol{K}^{(1,Q)} \\ \boldsymbol{K}^{(2,1)} & \boldsymbol{K}^{(2,2)} & \cdots & \boldsymbol{K}^{(2,Q)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}^{(Q,1)} & \boldsymbol{K}^{(Q,2)} & \cdots & \boldsymbol{K}^{(Q,Q)} \end{bmatrix}, \tag{7}$$

where $\boldsymbol{K}^{(p,q)}$ is a correlation matrix of the subvectors $\boldsymbol{k}_n^{(p)} = [\kappa^{(p)}(\boldsymbol{x}_1^{(p)}, \boldsymbol{u}_n), \kappa^{(p)}(\boldsymbol{x}_2^{(p)}, \boldsymbol{u}_n), \cdots, \kappa^{(p)}(\boldsymbol{x}_{r^{(p)}}^{(p)}, \boldsymbol{u}_n)]^\mathsf{T}$ and $\boldsymbol{k}_n^{(q)} = [\kappa^{(q)}(\boldsymbol{x}_1^{(q)}, \boldsymbol{u}_n), \kappa^{(q)}(\boldsymbol{x}_2^{(q)}, \boldsymbol{u}_n), \cdots, \kappa^{(q)}(\boldsymbol{x}_{r^{(q)}}^{(q)}, \boldsymbol{u}_n)]^\mathsf{T}$, which are corresponding to the pth and qth kernel function, respectively. Given weights $(w^{(1)}, w^{(2)}, \cdots, w^{(Q)})$, the corresponding weighted version of $\boldsymbol{K}$ can be written as follows:

$$\begin{bmatrix} (w^{(1)})^2\boldsymbol{K}^{(1,1)} & w^{(1)}w^{(2)}\boldsymbol{K}^{(1,2)} & \cdots & w^{(1)}w^{(N)}\boldsymbol{K}^{(1,N)} \\ w^{(2)}w^{(1)}\boldsymbol{K}^{(2,1)} & (w^{(2)})^2\boldsymbol{K}^{(2,2)} & \cdots & w^{(2)}w^{(Q)}\boldsymbol{K}^{(2,Q)} \\ \vdots & \vdots & \ddots & \vdots \\ w^{(Q)}w^{(1)}\boldsymbol{K}^{(Q,1)} & w^{(Q)}w^{(2)}\boldsymbol{K}^{(Q,2)} & \cdots & (w^{(Q)})^2\boldsymbol{K}^{(Q,Q)} \end{bmatrix}.$$

If one submatrix $\boldsymbol{K}^{(q,q)}$ is dominant compared with the other $\boldsymbol{K}^{(p,p)}$, $q \neq p$, then the coefficients corresponding to any pth kernel will hardly change. Under an appropriate weight setting, one can balance the submatrices so that all the coefficients are updated fairly.

Here we evaluate the dominance of $\boldsymbol{K}^{(q,q)}$ by its mean eigenvalue, which can be computed as

$$\frac{\mathrm{trace}\big(\boldsymbol{K}^{(q,q)}\big)}{r^{(q)}} = \frac{\mathrm{trace}\big(E[\boldsymbol{k}_n^{(q)}(\boldsymbol{k}_n^{(q)})^\mathsf{T}]\big)}{r^{(q)}} = \frac{E[\|\boldsymbol{k}_n^{(q)}\|_2^2]}{r^{(q)}}, \quad (8)$$

where $r^{(q)}$ is the size of $\boldsymbol{K}^{(q,q)}$. With sufficiently large n, we can use the following approximation:

$$\boldsymbol{K} = E\big[\boldsymbol{k}_n\boldsymbol{k}_n^\mathsf{T}\big] \approx \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{k}_l\boldsymbol{k}_l^\mathsf{T} =: \hat{\boldsymbol{K}}. \quad (9)$$

This implies that the matrix $\hat{\boldsymbol{K}}$ is a reasonable substitute of $\boldsymbol{K}$. Suppose that we set the weights $w_n^{(1)}$, $\forall q \in \{1, 2, \cdots, Q\}$ at every nth iteration. Along the above discussion, given submatrices $\hat{\boldsymbol{K}}_n^{(q,q)}$ of $\hat{\boldsymbol{K}}$, the mean eigenvalue of the weighted submatrix $(w_n^{(q)})^2\hat{\boldsymbol{K}}_n^{(q,q)}$ is calculated from (8) as follows:

$$\begin{aligned} &(\text{mean eigenvalue of } (\mathrm{w}_\mathrm{n}^{(\mathrm{q})})^2\hat{\boldsymbol{K}}_\mathrm{n}^{(\mathrm{q,q})}) \\ =&\frac{\mathrm{trace}\big((w_n^{(q)})^2\hat{\boldsymbol{K}}_n^{(q,q)}\big)}{r_n^{(q)}} = (w_n^{(q)})^2\frac{\mathrm{trace}\big(\hat{\boldsymbol{K}}_n^{(q,q)}\big)}{r_n^{(q)}} \\ =&(w_n^{(q)})^2\frac{\frac{1}{n}\sum_{l=1}^{n}\|\boldsymbol{k}_l\|_2^2}{r_n^{(q)}}. \end{aligned} \quad (10)$$

The proposed weight design is thus formulated by

$$(w_n^{(q)})^2\frac{\frac{1}{n}\sum_{l=1}^{n}\|\boldsymbol{k}_l\|_2^2}{r_n^{(q)}} = \mathrm{Const.}, \quad (11)$$

and

$$\sum_{q=1}^{Q} w_n^{(q)} = 1,\ w_n^{(q)} \geq 0, \quad (12)$$

where $r_n^{(q)}$ is the size of $\hat{\boldsymbol{K}}_n^{(q,q)}$. Then combining (11) and (12) straightforwardly leads to the following weight:

$$w_n^{(q)} = \frac{\sqrt{\dfrac{r^{(q)}}{\sum_{l=1}^{n}\big\|\boldsymbol{k}_l^{(q)}\big\|_2^2}}}{\displaystyle\sum_{p=1}^{Q}\sqrt{\frac{r^{(p)}}{\sum_{l=1}^{n}\big\|\boldsymbol{k}_l^{(p)}\big\|_2^2}}}. \quad (13)$$

Table 1. Initial setting for Experiments 1, 2

Parameter	Value(s)
Step size	$\mu = 0.1$
Kernel parameters	$\sigma^{(1)} = 1,\ \sigma^{(2)} = 0.02$
Coherence thresholds	$\delta^{(1)} = 0.92,\ \delta^{(2)} = 0.6$
Const. for Platt's criterion	$\epsilon = 0.01$

3.3. Implementation of the Automatic Weighting

Basically, the kernel weighting means imposing a weight w on a kernel $\kappa(\cdot,\cdot)$ to yield the weighted kernel $\kappa_w(\cdot,\cdot) := w\kappa(\cdot,\cdot)$. This weighted kernel can be directly applied to the MKNLMS algorithm by replacing the unweighted kernels into the weighted ones, and then doing the same update (5). Let us call such an implementation "straightforward". On the other hand, one can implement the kernel weighting to the MKNLMS algorithm in a different way by applying the weights, not to the inputs, but to the update equation as follows:

$$\boldsymbol{\alpha}_{n+1} = \tilde{\boldsymbol{\alpha}}_n - \mu\frac{\tilde{\boldsymbol{k}}_n^\mathsf{T}\tilde{\boldsymbol{\alpha}}_n - d_n}{\|\boldsymbol{W}_n\tilde{\boldsymbol{k}}_n\|_2^2}\boldsymbol{W}_n^2\tilde{\boldsymbol{k}}_n \quad (14)$$

with a weighting matrix

$$\boldsymbol{W}_n = \begin{bmatrix} w_n^{(1)}\boldsymbol{I}_{r_n^{(1)}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & w_n^{(2)}\boldsymbol{I}_{r_n^{(2)}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & w_n^{(Q)}\boldsymbol{I}_{r_n^{(Q)}} \end{bmatrix} \in \mathbb{R}^{r_n\times r_n},$$

where $r_n = \sum_{q=1}^{Q} r_n^{(q)}$, $\boldsymbol{I}_a \in \mathbb{R}^{a\times a}$ is the $a \times a$ identity matrix, and $\boldsymbol{O}$ is the zero matrix of appropriate size. Given the same fixed weights, this implementation works equivalently to the straightforward implementation in the MSE sense. The update (14) can be seen as using a metric $\boldsymbol{W}_n^{-2}$. In other words, we use the calculated weights (13) to design a metric for the update at each iteration. Therefore, let us call the kernel weighting by (14) the "metric-design" implementation. It has been reported that such a usage of metric unchanges the limit point of the projection algorithm, while it affects the convergence speed significantly [23–25]. The two ways of implementation are compared by a toy numerical example in the next section.

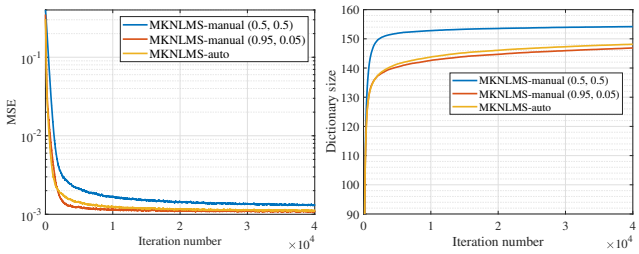
4. NUMERICAL EXPERIMENTS

Experiment 1: We consider the following simple system:

$$d_n = \sin\left(\frac{\pi}{3}u_n\right) - \exp\left(-\frac{(u_n - 0.5)^2}{2\cdot 0.1^2}\right) + \nu_n, \quad (15)$$

where ν_n is the additive i.i.d. Gaussian noise with its variance 10^{-3}. We estimate the above system by the MKNLMS algorithm with two Gaussian kernels defined as (3). In this experiment we employ the straightforward implementation of the kernel weighting. The initial setting is given in Table 1. All the figures show averaged results of 300 independent trials.

Figs. 3 (a) and (b) show the MSE and the dictionary evolutions under several weighting, respectively. The blue curve stands for the case $(w^{(1)},\ w^{(2)}) = (0.5,\ 0.5)$, which is equivalent to no weighting. The red curve stands for the weight setting $(w^{(1)},\ w^{(2)}) = (0.95,\ 0.05)$, which is carefully chosen. Finally, the yellow curves stand for the automatic weighting. In this case, the weights are set as (13) at each iteration. From Fig. 3 (a), one can see that the automatic weighting performs better than the unweighted settings in



(a) MSE (b) Dictionary evolution

Fig. 3. Performance of the MKNLMS algorithms. (straightforward kernel weighting)

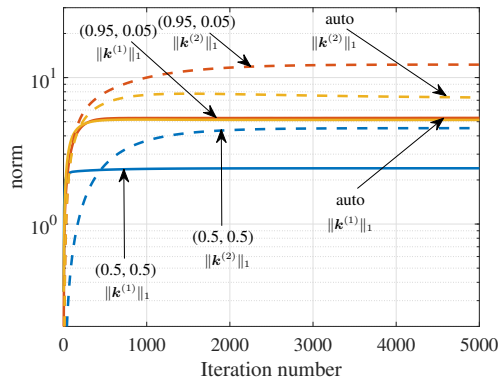
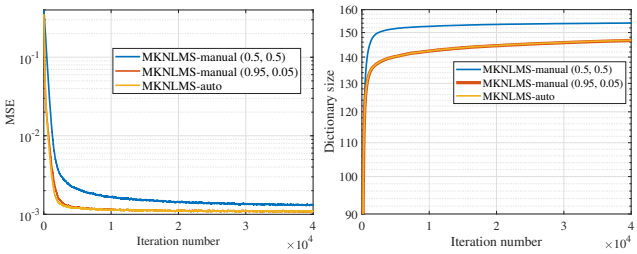


Fig. 4. Evolution of coefficients under different weights at initial iterations. (straightforward kernel weighting)



(a) MSE (b) Dictionary evolution

Fig. 5. Performance of the MKNLMS algorithms. (metric-design kernel weighting)

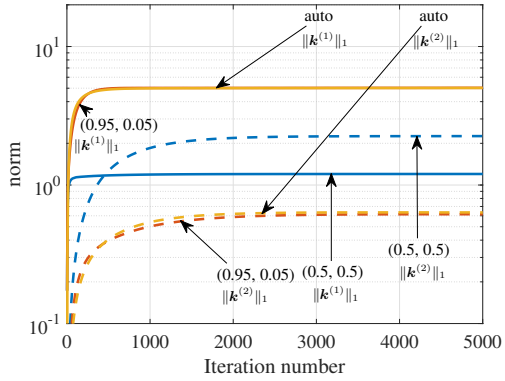


Fig. 6. Evolution of coefficients under different weights at initial iterations. (metric-design kernel weighting)

both of convergence speed and the MSE. It also achieves almost the same MSE with the carefully tuned setting at its steady state, although its convergence speed slows down drastically around the 2000–3000th iterations. Fig. 3 (b) shows that the proposed weighting leads to slightly more dictionary size than the carefully tuned setting. It seems that it does not need sufficiently large dictionary size to achieve fast convergence speed. The unweighted setting results neither fast convergence nor small dictionary.

Fig. 4 shows the ℓ_1-norms of the coefficient vectors, each of which corresponds to a kernel. The carefully tuned setting (red curves) seems to grow the coefficients enough. On the other hand, the coefficients grow less under the other settings. The coefficients corresponding different kernels grow in different speeds under the unweighted setting. The growing speeds are balanced under both of the carefully tuned setting and the proposed automatic weighting.

Experiment 2: We consider the same system and the same initial setting with Experiment 1. The difference from Experiment 1 is that we use the MKNLMS algorithm using the metric-design kernel weighting. From Fig. 5 together with Fig. 3, one can see that the two ways of implementation of weighting are equivalent in the both senses of MSE and dictionary size when the weights are fixed. On the other hand, the automatic weighting achieves better performance when the weights are used to design a metric than using the weighted kernels directly. Moreover, the automatic weighting leads to the similar performance to the carefully tuned weight setting, in both senses of the convergence speed and the dictionary size.

5. CONCLUSION

We have proposed an automatic kernel weighting technique for a multikernel adaptive filtering algorithm. Multikernel adaptive filtering is a promising approach for nonlinear estimation. Moreover, it can deal effectively with a multiscale target function. We have introduced two types of multiscaleness related to the target function and the input data. With positive weights, one can consider multikernel adaptive filtering with weighted kernels, since imposing a positive weight on a positive definite kernel preserves its positive definiteness. However, the weights for kernels must be chosen carefully in order to achieve desirable performance, in both senses of accuracy and computational cost. With a toy example, we have seen how the weights affect on the MSE and dictionary size which are yielded by a multikernel adaptive filtering algorithm.

The proposed automatic kernel weighting has been developed to adjust each update of the coefficients in a good balance. The balance among kernels has been evaluated using the mean eigenvalues of the autocorrelation matrices of corresponding kernelized inputs. At each iteration, we change the weights to achieve a good balance of the mean eigenvalues. There have been two ways to implement the proposed weighting technique: (i) the straightforward implementation which uses the weighted kernels directly, and (ii) the metric-design implementation which uses the weights to design a time-varying metric. By numerical experiments, the proposed weighting technique has been shown to reduce the MSE and the dictionary size simultaneously, while maintaining compatible convergence speed. It has also been seen that the proposed technique balances the growing speed of coefficients well, whereas the unweighted setting causes unbalance of the coefficient growth. Especially, the metric-design implementation of the proposed technique leads to compatible performance to the carefully tuned weight setting.

6. REFERENCES

[1] M. Costa, A. L. Goldberger and C.-K. Peng, "Multiscale entropy analysis of biological signals," *PHYSICAL REVIEW E*, vol. 71, no. 2, pp. 021906 1–18, Feb. 2005.

[2] D. Kushnir, M. Galun and A. Brandt, "Fast multiscale clustering and manifold identification," *Pattern Recognition*, vol. 39, pp. 1876–1891, 2006.

[3] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat and H. Wendt, "Multiscale discrete approximations of fourier integral operators associated with canonical transformations and caustics," *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, vol. 11, no. 2, pp. 566–585, 2013.

[4] R. F. Leonarduzzi, M. E. Torres and P. Abry, "Scaling range automated selection for wavelet leader multifractal analysis," *Signal Process.*, vol. 105, pp. 243–257, 2014.

[5] L. N. Sharma, R. K. Tripathy and S. Dandapat, "Multiscale energy and eigenspace approach to detection and localization of myocardial infarction," *IEEE Trans. Biomedical Eng.*, vol. 62, no. 7, pp. 1827–1837, Jul. 2015.

[6] T. Nakamura, K. Kiyono, H. Wendt, P. Abry and Y. Yamamoto, "Multiscale analysis of intensive longitudinal biomedical signals and its clinical applications," *Proc. IEEE*, vol. 104, no. 2, pp. 242–261, Feb. 2016.

[7] J. Liang, "A review of multiscale science: Materials, biology, multiscale data analysis and examples from complex physiological systems," in *Proc. of 2017 IEEE Int. Conf. Mechatronics and Automation*, 2017.

[8] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[9] R. Pokharel, S. Seth and J. C. Principe, "Mixture kernel least mean square," in *Proc. 2013 Int. Joint Conf. on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[10] M. Yukawa and R. Ishii, "Online model selection and learning by multikernel adaptive filtering," in *Proc. 21th Eur. Signal Process. Conf. (EUSIPCO)*, 2013, pp. 1–5.

[11] M. Yukawa, "Adaptive learning in cartesian product of reproducing kernel hilbert spaces," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6037–6048, Nov. 2015.

[12] S. V. Vaerenbergh, S. Scardapane and I. Santamaria, "Recursive multikernel filters exploiting nonlinear temporal structure," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, 2017, pp. 2674–2678.

[13] T. Wada, K. Fukumori and T. Tanaka, "Dictionary learning for gaussian kernel adaptive filtering with variable kernel center and width," in *Proc. 43th Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 2766–2770.

[14] M. Ohnishi and M. Yukawa, "Online nonlinear estimation via iterative L^2-space projections: Reproducing kernel of subspace," *IEEE Trans. Signal Process.*, vol. 66, no. 15, pp. 4050–4064, Aug. 2018.

[15] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.

[16] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Trans. signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[17] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[18] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[19] M. Kasparick, R. L. G. Cavalcante, S. Valentin, S. Stánczak and M. Yukawa, "Kernel-based adaptive online reconstruction of coverage maps with side information," *IEEE Trans. Vehicular Tech.*, vol. 65, no. 7, pp. 5461–5473, Jul. 2016.

[20] D. A. Awan, R. L. G. Cavalcante, M. Yukawa and S. Stanczak, "Detection for 5G-NOMA: An online adaptive machine learning approach," in *2018 IEEE International Conference on Communications (ICC)*, 2018.

[21] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5505–5519, Nov 2018.

[22] M. Ohnishi, L. Wang, G. Notomista and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Trans. Robotics*, under review, https://arxiv.org/abs/1801.09627.

[23] M. Yukawa, K. Slavakis, and I. Yamada, "Adaptive parallel quadratic-metric projection algorithms," *IEEE Trans. Audio, Speech and Language Process.*, vol. 15, no. 5, pp. 1665–1680, Jul. 2007.

[24] M. Yukawa and I. Yamada, "A unified view of adaptive variable-metric projection algorithms," *EURASIP Jour. Adv. Signal Process.*, vol. 2009, pp. 1–13, 2009.

[25] M. Yukawa and I. Yamada, "Krylov-proportionate adaptive filtering techniques not limited to sparse systems," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 927–943, 2009.