Documentation for Kernel Adaptive Filtering Toolbox

Steven Van Vaerenbergh

May 27, 2014

Abstract

The Kernel Adaptive Filtering Toolbox is a benchmarking toolbox to evaluate and compare kernel adaptive filtering algorithms in Matlab. Kernel adaptive filtering algorithms are online techniques suitable for nonlinear filtering, prediction, tracking and regression. This toolbox contains implementations of all major kernel adaptive filtering algorithms, and tools to measure and compare the performance, memory usage, speed and other properties.

Contents

Chapter 1

Introduction

This document is a work in progress.

1.1 Goals of the toolbox

The goals of this toolbox are twofold:

1. To provide a repository for Matlab implementations of kernel adaptive filtering algorithms;

2. To provide tools that allow to compare all aspects of the different algorithms.

A list of the included algorithms and tools can be found in Chapter 5.

Chapter 2

First use

2.1 Installation

1. Download the toolbox and run the `installation.m` file in the root folder.
2. Type `savepath` to save the changes to the path.

2.2 Directories included in the toolbox

- `data/` contains data sets.
- `demo/` contains demos and test files.
- `doc/` contains the documentation for the toolbox.
- `lib/` contains the algorithm libraries and utilities.

2.3 Octave / Matlab pre-2008a

This toolbox uses the `classdef` command which is not supported in Matlab pre-2008a and not yet in Octave. The older 0.x versions of this toolbox do not use classdef and can therefore be used with all versions of Matlab and Octave. `http://sourceforge.net/projects/kafbox/files/`

Chapter 3

Kernel adaptive filtering

3.1 Framework for online kernel methods

Online learning methods update their solution iteratively. In the standard online learning framework, each iteration consists of several trials [Wik14]. During the n-th iteration:

1. The algorithm first receives an input datum, $\mathbf{x}_n$;
2. Then, it calculates the estimated output $\hat{y}_n$ corresponding to this datum.
3. The true outcome y_n is made available shortly thereafter, which enables the algorithm to calculate the loss $L(\cdot)$ incurred on the data pair $(\mathbf{x}_n, y_n)$;
4. Finally, the solution is updated.

A typical setup for online system identification with a kernel-based method is depicted in Fig. 3.1. It represents an unknown nonlinear system, whose input data $\mathbf{x}_n$ and response y_n (including additive noise r_n) can be measured at different time steps, and an adaptive kernel-based algorithm, which is used to identify the system's response.

Online algorithms should be capable of operating during extended periods of time, processing large amounts of data. Kernel methods rely on a functional representation that grows as the amount of observations increases. A naïve implementation of an online kernel method will therefore require growing computational resources during operation, leading to performance issues once the available memory is insufficient to store the training data or once the computations for one update take more time than the interval between incoming data [KSW04].

3.2 Kernel adaptive filtering algorithms

Kernel adaptive filtering algorithms are online regression algorithms. While early kernel adaptive filtering algorithms had a clear connection to classical

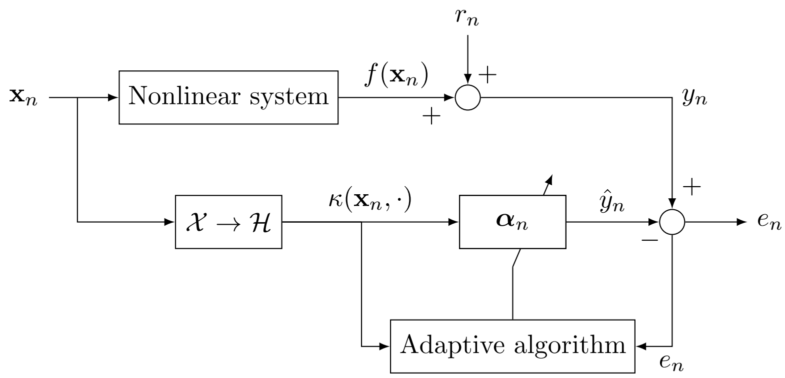

Figure 3.1: Kernel-based adaptive system identification. Figure adapted from [PBRT12].

adaptive filters, such as LMS, RLS and KAPA filters, later implementations explored Gaussian Process and projections based algorithms.

The different kernel adaptive filtering algorithms proposed in the literature vary in several aspects:

- The type of filter they relate to: typically LMS, RLS or KAPA;
- The computational complexity w.r.t. the amount of elements in memory, M: typically $\mathcal{O}(M)$ or $\mathcal{O}(M^2)$;
- The approach to building a sparse dictionary and to possible prune data from it afterwards;
- The convergence speed;
- The ability to employ multiple kernels and learn their respective weights.

The algorithm profiler described in Chapter 4 allows to compare these different aspects empirically.

3.3 Algorithm code structure

Since all algorithms share the same structure, they are coded in a common framework. In particular:

- Each algorithm is contained in a single file in the `/lib` folder.
- Each algorithm is implemented as an object in matlab, using the `classdef` syntax.

- Each algorithm has two basic public methods: `evaluate` and `train`. The object code contains only one iteration of the algorithm. The for-loop over the time index that governs the online operation should go in an external script.

Furthermore, efforts are made to make the code

1. As human-readable as possible, in the first place;
2. Short and structured, in the second place.

As far as possible, algorithm structure should follow the pseudocode from the corresponding publication.

Finally, the following best practices for scientific computing are taken into account [WAB+14]:

- Variable naming should correspond to the nomenclature used in the corresponding publication whenever possible;
- Comments should be used sparingly. Document the design and purpose of the code rather than its mechanics.

3.4 Template for kernel adaptive filtering algorithms

```
% This is the template used for kernel adaptive filtering algorithms in
% the kernel adaptive filtering toolbox. [Delete this line.]
%
% [The name of the algorithm goes here]
%
% [A reference to the original publication of the algorithm goes here,
% including a link to its DOI url: http://dx.doi.org/xxx]
%
% This file is part of the Kernel Adaptive Filtering Toolbox for Matlab.
% http://sourceforge.net/projects/kafbox/

classdef kafbox_template

    properties (GetAccess = 'public', SetAccess = 'private') % parameters
        param1 = 1;
        param2 = 2;
        kerneltype = 'gauss'; % kernel type
        kernelpar = 1; % kernel parameter
    end

    properties (GetAccess = 'public', SetAccess = 'private') % variables
        dict = []; % dictionary
        alpha = []; % expansion coefficients
    end

    methods

```

```
        function kaf = kafbox_template(parameters) % constructor
            if (nargin > 0) % copy valid parameters
                for fn = fieldnames(parameters)',
                    if strmatch(fn,fieldnames(kaf),'exact'),
                        kaf.(fn{1}) = parameters.(fn{1});
                    end
                end
            end
        end

        function y_est = evaluate(kaf,x) % evaluate the algorithm
            if size(kaf.dict,1)>0
                k = kernel(kaf.dict,x,kaf.kerneltype,kaf.kernelpar);
                y_est = k'*kaf.alpha;
            else
                y_est = zeros(size(x,1),1);
            end
        end

        function kaf = train(kaf,x,y) % train the algorithm
            if size(kaf.dict,2)==0 % initialize
                kaf.dict = x;
                kaf.alpha = 0;
            else

                % [main algorithm training goes here]

                % [example of a helper function]
                z = kaf.helper1(x,y);
            end

        end

    end

    methods (Static = true) % [helper functions go here]

        function z = helper1(x,y)
            z = x*y;
            % operations
        end

    end
end
```

3.5 Performing online learning

Each kernel adaptive filtering algorithm is implemented as a Matlab class. To perform online learning with an algorithm, first define its options:

```
options = struct('nu',1E-4,'kerneltype','gauss','kernelpar',32);
```

Next, create an instance of the filter. E.g., for an instance of the KRLS algorithm that uses the ALD criterion run:

```
kaf = aldkrls(options);
```

One iteration of training is performed by feeding one input-output data pair to the filter:

```
kaf = kaf.train(x,y);
```

The outputs for one or more test inputs are evaluated as follows:

```
Y_test = kaf.evaluate(X_test);
```

3.6 Example learning script: time-series prediction

```
% 1-step ahead prediction on Lorenz attractor time-series data
%
% This file is part of the Kernel Adaptive Filtering Toolbox for Matlab.
% http://sourceforge.net/projects/kafbox/

[X,Y] = kafbox_data(struct('file','lorenz.dat','embedding',6));

% make a kernel adaptive filter object of class aldkrls with options:
% ALD threshold 1E-4, Gaussian kernel, and kernel width 32
kaf = aldkrls(struct('nu',1E-4,'kerneltype','gauss','kernelpar',32));

%% RUN ALGORITHM
N = size(X,1);
Y_est = zeros(N,1);
for i=1:N,
    if ~mod(i,floor(N/10)), fprintf('.'); end % progress indicator, 10 dots
    Y_est(i) = kaf.evaluate(X(i,:)); % predict the next output
    kaf = kaf.train(X(i,:),Y(i)); % train with one input-output pair
end
fprintf('\n');
SE = (Y-Y_est).^2; % test error

%% OUTPUT
fprintf('MSE after first 1000 samples: %.2fdB\n\n',10*log10(mean(SE(1001:end))));
```

Chapter 4

Algorithm profiler

The algorithm profiler is a tool that allows to compare the trade-off between cost and performance between several algorithm configurations and/or several algorithms.

A demo configuration script is given in the file `demo/demo_profiler_prediction_lorenz.m`.

Chapter 5

List of included algorithms

Matlab implementations of the following algorithms are included in the toolbox:

- Approximate Linear Dependency Kernel Recursive Least-Squares (ALD-KRLS) [EMM04].
- Sliding-Window Kernel Recursive Least-Squares (SW-KRLS) [VVVS06].
- Naive Online Regularized Risk Minimization Algorithm (NORMA) [KSW04].
- Kernel Least-Mean-Square (KLMS) [LPP08].
- Fixed-Budget Kernel Recursive Least-Squares (FB-KRLS) [VVSLP10].
- Kernel Recursive Least-Squares Tracker (KRLS-T) [VVLGS12].
- Quantized Kernel Least Mean Squares (QKLMS) [CZZP12].
- Random Fourier Feature Kernel Least Mean Square (RFF-KLMS) algorithm [SAM12].
- Extended Kernel Recursive Least Squares (EX-KRLS) [LPWP09].
- Gaussian-Process based estimation of the parameters of KRLS-T [VVSLG12].
- Kernel Affine Projection (KAP) algorithm with Coherence Criterion [RBH09].
- Kernel Normalized Least-Mean-Square (KNLMS) algorithm with Coherence Criterion [RBH09].
- Recursive Least-Squares algorithm with exponential weighting (RLS) [Say03].
- Multikernel Normalized Least Mean Square algorithm with Coherence-based Sparsification (MKNLMS-CS) [Yuk12].
- Parallel HYperslab Projection along Affine SubSpace (PHYPASS) algorithm [TY13].

- Fixed-budget kernel least mean squares (FB-KLMS) algorithm [Rze12].
- Leaky Kernel Affine Projection Algorithm (LKAPA, including KAPA-1 and KAPA-3) and Normalized Leaky Kernel Affine Projection Algorithm (NLKAPA, including KAPA-2 and KAPA-4) [LPP08].
- Kernel Affine Projection Subgradient Method (KAPSM) [STY08].
- Kernel Least Mean Squares algorithm with Coherence-Sparsification criterion and L1-norm regularization (KLMS-CSL1) and with active L1-norm regularization (KLMS-CSAL1) [GCR+13].
- Mixture Kernel Least Mean Square (MxKLMS) algorithm [PSP13].

Chapter 6

About

- **Name:** Kernel Adaptive Filtering Toolbox
- **Abbreviation:** KAFBOX
- **Punchline:** a Matlab benchmarking toolbox for kernel adaptive filtering
- **Author:** Steven Van Vaerenbergh
  - Homepage: `http://gtas.unican.es/people/steven/`
  - Email: `steven@gtas.dicom.unican.es`
  - Affiliation: Advanced Signal Processing Group, Department of Communications Engineering, University of Cantabria, Spain.
- **Collaborators:** Miguel Lazaro-Gredilla, Sohan Seth, Masahiro Yukawa, Masa-aki Takizawa, Osamu Toda, Dominik Rzepka, Pantelis Bouboulis
- **URL:** `https://github.com/steven2358/kafbox`
- **Citing:** Published reports of research using this code (or a modified version) should cite [VVS13].
- **Environment:** Matlab[1].
- **Origin:** KAFBOX is a fork of Kernel Methods Toolbox (KMBOX) v0.6 (`http://sourceforge.net/p/kmbox/`)
- **Extending the code:** Template files are provided to encourage external authors to include their own code into the toolbox. Contributions can be made through Github's fork and pull system.
- **License:** FreeBSD

[1] `http://www.mathworks.com/products/matlab/`

Bibliography

[CZZP12] Badong Chen, Songlin Zhao, Pingping Zhu, and José C. Príncipe. Quantized kernel least mean square algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 23(1):22–32, January 2012.

[EMM04] Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, August 2004.

[GCR+13] Wei Gao, Jie Chen, C. Richard, Jianguo Huang, and R. Flamary. Kernel lms algorithm with forward-backward splitting for dictionary learning. In *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5735–5739, May 2013.

[KSW04] Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, August 2004.

[LPP08] Weifeng Liu, Puskal P. Pokharel, and José C. Príncipe. The kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554, 2008.

[LPWP09] Weifeng Liu, Il Park, Yiwen Wang, and José C. Príncipe. Extended kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, 57(10):3801 –3814, October 2009.

[PBRT12] Wemerson D. Parreira, JosÉ Carlos M. Bermudez, Cédric Richard, and Jean-Yves Tourneret. Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 60(5):2208–2222, 2012.

[PSP13] R. Pokharel, S. Seth, and J.C. Principe. Mixture kernel least mean square. In *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1–7, August 2013.

[RBH09] Cédric Richard, José Carlos M. Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57(3):1058–1067, March 2009.

[Rze12] D. Rzepka. Fixed-budget kernel least mean squares. In *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, September 2012.

[SAM12] Abhishek Singh, Narendra Ahuja, and Pierre Moulin. Online learning with kernels: Overcoming the growing sum problem. In *2012 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2012.

[Say03] Ali H. Sayed. *Fundamentals of Adaptive Filtering*. Wiley-IEEE Press, 2003.

[STY08] Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada. Online kernel-based classification using adaptive projection algorithms. *IEEE Transactions on Signal Processing*, 56(7):2781–2796, 2008.

[TY13] M.-A. Takizawa and M. Yukawa. An efficient data-reusing kernel adaptive filtering algorithm based on parallel hyperslab projection along affine subspaces. In *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3557–3561, May 2013.

[VVLGS12] Steven Van Vaerenbergh, Miguel Lázaro-Gredilla, and Ignacio Santamaría. Kernel recursive least-squares tracker for time-varying regression. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1313–1326, August 2012.

[VVS13] Steven Van Vaerenbergh and Ignacio Santamaría. A comparative study of kernel adaptive filtering algorithms. In *2013 IEEE Digital Signal Processing (DSP) Workshop and IEEE Signal Processing Education (SPE)*, 2013.

[VVSLG12] Steven Van Vaerenbergh, Ignacio Santamaría, and Miguel Lázaro-Gredilla. Estimation of the forgetting factor in kernel recursive least squares. In *2012 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, September 2012.

[VVSLP10] Steven Van Vaerenbergh, Ignacio Santamaría, Weifeng Liu, and José C. Príncipe. Fixed-budget kernel recursive least-squares. In *2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, USA, April 2010.

[VVVS06] Steven Van Vaerenbergh, Javier Vía, and Ignacio Santamaría. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In *2006 IEEE International Conference* *on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 789–792, Toulouse, France, May 2006.

[WAB+14] Greg Wilson, DA Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. Best practices for scientific computing. *PLoS biology*, 12(1):e1001745, 2014.

[Wik14] Wikipedia. Online machine learning — wikipedia, the free encyclopedia, 2014. [Online; accessed 26-May-2014].

[Yuk12] Masahiro Yukawa. Multikernel adaptive filtering. *IEEE Transactions on Signal Processing*, 60(9):4672–4682, 2012.