LMS/RLS Kernel-LMS/RLS

Manuel Forrer, Markus Fröhle

Signal Processing and Speech Communication Laboratory

December 17, 2009

Outline

Motivation

Applications for Adaptive Filters:

- echo/noise cancellation

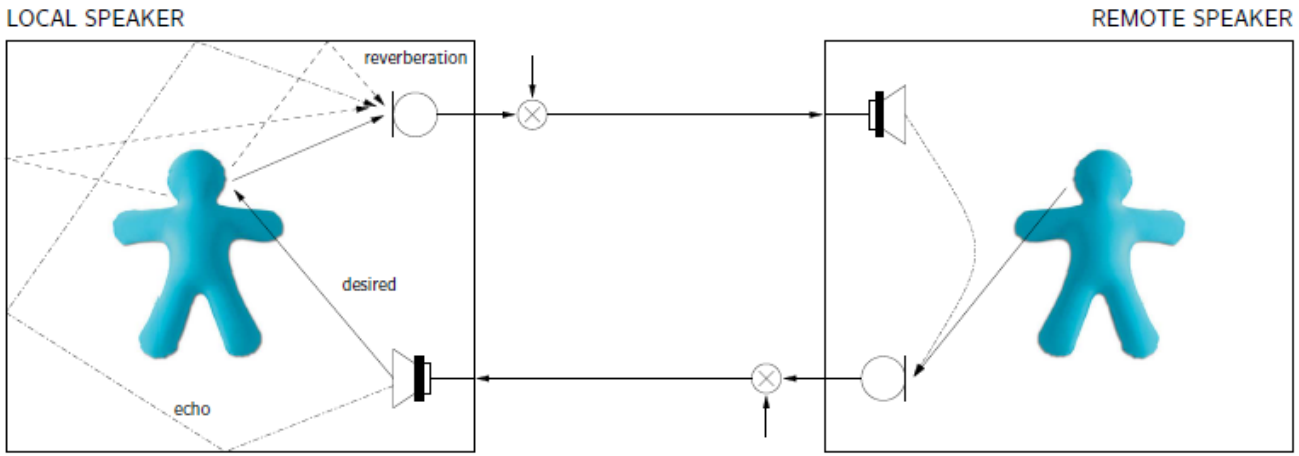

Figure: Echo cancellation [6]

Motivation

- channel equalization

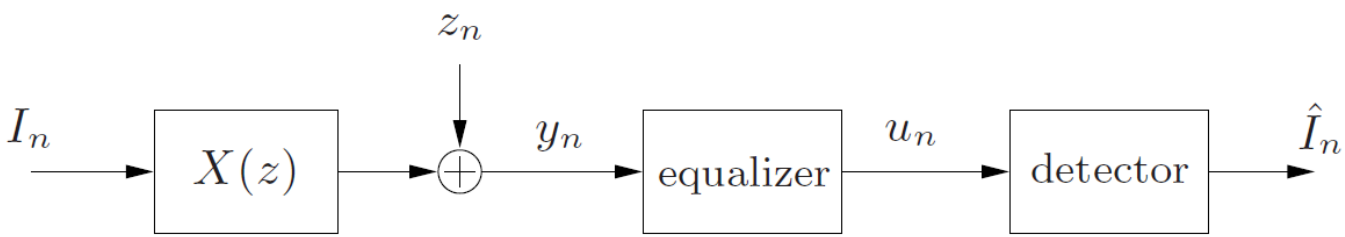

Figure: Channel Equalization [7]

- system identification

Definitions

- stochastic variables X scalar and $Y = [y(1), y(2), \ldots, y(N)]^T$
- covariance matrices

$$\Sigma_{YX} = E[YX]$$

$$\Sigma_{YY} = E[YY]$$

The linear MMSE estimator

- often the MMSE estimator $\hat{x}_{opt}(Y) = E[X|Y]$ is hard to compute
- therefore introduce a linear estimate

$$\hat{x}(Y) = \sum_{n=1}^{N} \theta(n) y(n) = Y^T \theta$$

- where

$$\theta = [\theta(1), \theta(2), \ldots, \theta(N)]^T$$

The linear MMSE estimator

- The optimal choice of the weight vector θ minimizes the MSE

$$MSE(\hat{x}(Y)) = E[(X - \hat{x}(Y))^2] = E[(X - Y^T\theta)^2]$$

- derive with respect to θ and set to zero:

$$\frac{d}{d\theta} MSE(\hat{x}(Y)) = E[(X - Y^T\theta)Y] = 0$$

- it follows that

$$\Sigma_{YX} - \Sigma_{YY}\theta = 0$$

the linear MMSE estimator

$$\hat{x}_{opt}(Y) = \Sigma_{YY}^{-1}\Sigma_{YX}Y = \theta_{opt}^T Y$$

Least Mean Squares (LMS) - Algorithm

- **Problem:** in practise second order moments are not available
- if the MSE criterion is time dependent, we can rewrite it to

$$MSE(n, \theta) = E[(x(n) - Y^T(n)\theta)^2]$$

- where

$$Y(n) = [y(n), y(n-1), \ldots, y(n-N+1)]^T$$

$$\theta = [\theta(0), \ldots, \theta(N-1)]^T$$

LMS - Algorithm

- weight update criterion:

$$\hat{\theta}(n) = \hat{\theta}(n-1) - \frac{\mu}{2}\frac{\partial}{\partial\theta} MSE(n,\theta) \mid_{\theta=\hat{\theta}(n-1)}$$

- drop expectation of MSE, use *instantaneous* error instead

$$\widehat{MSE}_{LMS}(n,\theta) = (x(n) - Y^T(n)\theta)^2$$

- calculate gradient:

$$\frac{\partial}{\partial\theta}\widehat{MSE}_{LMS}(n,\theta) = -2(x(n) - Y^T(n)\theta)Y(n)$$

LMS-Algorithm

$$\hat{\theta}(n) = \hat{\theta}(n-1) + \mu Y(n)[x(n) - Y^T(n)\hat{\theta}(n-1)]$$

Properties of the LMS-Algorithm

- both $x(n)$ and $y(n)$ have to be available
- the LMS-Algorithm produces estimates ("guesses") of the optimal parameter vector

$$\theta_{opt}(Y) = \Sigma_{YY}^{-1}\Sigma_{Yx}$$

- computational cost of $\mathcal{O}(N)$ operations in each iteration

Analysis of the LMS-Algorithm

- What are the stability and convergence properties of the LMS-Algorithm?

Analysis of the LMS-Algorithm

- What are the stability and convergence properties of the LMS-Algorithm?
- writing out the LMS formula gives

$$\hat{\theta}(n) = (I - \mu Y(n)Y^T(n))\hat{\theta}(n-1) + \mu x(n)Y(n)$$

- this is a linear state-space system with *stochastic* transition matrix

Analysis of the LMS-Algorithm

- What are the stability and convergence properties of the LMS-Algorithm?
- writing out the LMS formula gives

$$\hat{\theta}(n) = (I - \mu Y(n)Y^T(n))\hat{\theta}(n-1) + \mu x(n)Y(n)$$

- this is a linear state-space system with *stochastic* transition matrix
- difficult to analyse!
- analyse the averaged system instead
- only possible if μ is small [1]

Analysis of the LMS-Algorithm

- introduce

$$\theta_{opt} = \Sigma_{YY}^{-1} \Sigma_{Yx}$$

$$\hat{x}_{opt}(n) = Y^T(n)\theta_{opt}$$

$$\tilde{x}(n) = x(n) - \hat{x}_{opt}(n)$$

$$\tilde{\theta}(n) = \theta_{opt} - \hat{\theta}(n)$$

- the averaged system describing the error propagation
- (blackboard)

Analysis of the LMS-Algorithm

- introduce

$$\theta_{opt} = \Sigma_{YY}^{-1}\Sigma_{Yx}$$

$$\hat{x}_{opt}(n) = Y^T(n)\theta_{opt}$$

$$\tilde{x}(n) = x(n) - \hat{x}_{opt}(n)$$

$$\tilde{\theta}(n) = \theta_{opt} - \hat{\theta}(n)$$

- the averaged system describing the error propagation
- (blackboard)

$$\tilde{\theta}(n) = [I - \mu\Sigma_{YY}]\tilde{\theta}(n-1) - \mu\tilde{x}(n)Y(n)$$

Stability

- the LMS-Algorithm is stable if the eigenvalues of $I - \mu\Sigma_{YY}$ are strictly inside the unit circle
- (blackboard)

Stability

- the LMS-Algorithm is stable if the eigenvalues of $I - \mu\Sigma_{YY}$ are strictly inside the unit circle
- (blackboard)

stable if

$$0 < \mu < \frac{2}{\lambda_1}$$

$\lambda_1 \cdots$ largest eigenvalue of Σ_{YY}

- also the mean of the parameter error $\tilde{\theta}(n)$ converges then to zero

More Properties of the LMS

Misadjustment (= steady state prediciton error)

- for large n it holds that

$$MSE_{LMS}(n) \approx MSE(\theta_{opt})(1 + M)$$

- where the Misadjustment M is given by

$$M \approx \frac{\mu N \sigma_y^2}{2}$$

- M is proportional to
  - the step-size μ
  - the number of parameters N
  - the signal power (variance) σ_y^2

Properties of the LMS

thus there is a tradeoff between:

- small $\mu-$ slow convergence, small misadjustment
- large $\mu-$ fast convergence, large misadjustment

- $\frac{\lambda_1}{\lambda_N}$ large $\Rightarrow$ slow convergence
- $\frac{\lambda_1}{\lambda_N}$ close to one $\Rightarrow$ fast convergence
- (contour plot)

Normalized LMS - NLMS

- Stability and misadjustment depend on the signal power (variance) σ_y^2
- LMS can be made insensitive by *normalizing* the step size

$$\mu(n) = \frac{\bar{\mu}}{c + ||Y(n)||^2}$$

where $c > 0$ is a constant

MATLAB-Demo

System identification

Would like to

- identify an unknown FIR filter with LMS

Recursive Least Squares (RLS) - Algorithm

Problems with LMS:

- Convergence rate is poor when Σ_{YY} is ill-conditioned (large eigenvalue spread)
- LMS is noise sensitive

To avoid this, use a $MSE(\theta)$ with:

- second derivative information (Hessian)
- a less noise sensitive approximation

Recursive Least Squares (RLS) - Algorithm

Use second derivative information and set to zero

- since $MSE(\theta)$ is quadratic it holds for any parameter value θ

$$\left.\frac{\partial MSE(\theta)}{\partial \theta}\right|_{\theta=\theta_{opt}} = \frac{\partial MSE(\theta)}{\partial \theta} + \frac{\partial^2 MSE(\theta)}{\partial \theta^2}(\theta_{opt} - \theta) \stackrel{!}{=} 0$$

$$\Rightarrow \theta_{opt} = \theta - \left[\frac{\partial^2 MSE(\theta)}{\partial \theta^2}\right]^{-1} \frac{\partial MSE(\theta)}{\partial \theta}$$

Recursive Least Squares (RLS) - Algorithm

- Denote $\widehat{MSE}'(n,\theta)$ and $\widehat{MSE}''(n,\theta)$ as approximations of the first and second order derivative

$$\hat{\theta}(n) = \hat{\theta}(n-1) - \mu \left[\widehat{MSE}''(n,\hat{\theta}(n-1))\right]^{-1} \widehat{MSE}'(n,\hat{\theta}(n-1))$$

- Use a less noise sensitve approximation for the $MSE(\theta)$ e.g.

$$\widehat{MSE}_{RLS}(n,\theta) = \sum_{k=1}^{n} \lambda^{n-k}(x(k) - Y^T(k)\theta)^2$$

where $0 < \lambda \leq 1$ is known as the forgetting factor

Recursive Least Squares (RLS) - Algorithm

- Define

$$\hat{\Sigma}_{Yx}^{RLS}(n) = \sum_{k=1}^{n} \lambda^{n-k} Y(k) x(k)$$

$$\hat{\Sigma}_{YY}^{RLS}(n) = \sum_{k=1}^{n} \lambda^{n-k} Y(k) Y^T(k)$$

- Derive the corresponding gradient and Hessian approximations of the $\widehat{MSE}_{RLS}$ (blackboard)

Recursive Least Squares (RLS) - Algorithm

approximation of the $\widehat{MSE}_{RLS}$

$$\Rightarrow \frac{\partial}{\partial \theta} \widehat{MSE}_{RLS}(n,\theta) = -2\hat{\Sigma}_{Yx}^{RLS}(n) + 2\hat{\Sigma}_{YY}^{RLS}(n)\theta$$

$$\Rightarrow \frac{\partial^2}{\partial \theta^2} \widehat{MSE}_{RLS}(n,\theta) = 2\hat{\Sigma}_{YY}^{RLS}(n)$$

plug into weight update criterion and get

basic form of the RLS-Algorithm (with $\mu = 1$)

$$\hat{\theta}(n) = \hat{\theta}(n-1) + \left[\hat{\Sigma}_{YY}^{RLS}(n)\right]^{-1} (\hat{\Sigma}_{Yx}^{RLS}(n) - \hat{\Sigma}_{YY}^{RLS}(n)\hat{\theta}(n-1))$$

Alternative expression of RLS and comparison to LMS

RLS can be rewritten as

RLS-Algorithm ($\mu = 1$) [1]

$$\hat{\theta}(n) = \hat{\theta}(n-1) + \left[\hat{\Sigma}_{YY}^{RLS}(n)\right]^{-1} Y(n)[x(n) - Y^T(n)\hat{\theta}(n-1)]$$

with $\hat{\Sigma}_{YY}^{RLS}(n) = \sum_{k=1}^{n} \lambda^{n-k} Y(k) Y^T(k)$

Recall LMS-Algorithm ($\mu = 1$)

$$\hat{\theta}(n) = \hat{\theta}(n-1) + Y(n)[x(n) - Y^T(n)\hat{\theta}(n-1)]$$

$\Rightarrow$ only difference is the update direction

Analysis of the RLS-Algorithm

How does the forgetting factor λ influence the covariance matrix?

- $\lambda = 1$: all data are treated equally (no forgetting)
- $\lambda < 1$: old data has less influence (able to adapt changing signal properties)

Computational complexity:

- $\mathcal{O}(N^3)$ for the RLS-Algorithm $\left(\left[\hat{\Sigma}_{YY}^{RLS}(n) \right]^{-1} \right)$
- with Matrix Inversion Lemma [1] $\Rightarrow \mathcal{O}(N^2)$

Analysis of the RLS-Algorithm

Misadjustment:

- for $\lambda = 1$:

$$M = \frac{N}{n}$$

- for $\lambda < 1$:

$$M = \frac{(1-\lambda)N}{2}$$

- independent of signal power (variance)
- decays for large n (for the case where $\lambda = 1$)

MATLAB-Demo

System identification

Would like to

- identify an unknown FIR filter with RLS

MATLAB-Demo (Comparison between LMS and RLS)

Noise Cancellation

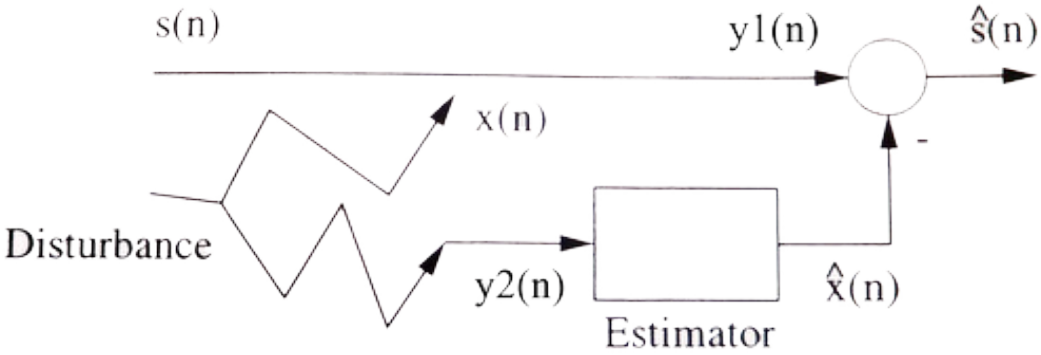

Figure: Source: [1]

$s(n)$ ... signal of interest
$x(n)$ ... disturbance
$\hat{x}(n)$ ... estimated disturbance (with adaptive filter)
$\hat{s}(n)$ ... estimated signal of interest

Kernel-LMS

Why kernels?

- Non-linear adaptive filters require a training set

Kernel Methods:

- proposed to produce non-linear algorithms from linear ones expressed with **inner product** by employing the **kernel trick**

Idea:

- Apply the LMS directly in kernel feature space and employ the kernel trick to obtain the solution in the input space

Kernel Methods

- Map data x_i from input space to high-dimensional feature space using mapping $\Phi(x_i)$
  - $\Rightarrow$ Non-linear mapping
- Kernel function:

$$\kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

Kernel Methods

- Map data x_i from input space to high-dimensional feature space using mapping $\Phi(x_i)$
  - $\Rightarrow$ Non-linear mapping
- Kernel function:

$$\kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

- Gaussian kernel:

$$\kappa(x, y) = exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$$

  - $\Rightarrow$infinite dimensional Hilbert space

From LMS to Kernel-LMS

LMS-Algorithm

$$\theta(n) = \theta(n-1) + \mu Y(n)[\underbrace{x(n) - Y^T(n)\theta(n-1)}_{e(n)}]$$

- transfom to feature space using mapping $\Phi(Y(n))$

$$\Omega(n) = \Omega(n-1) + \mu\Phi(Y(n))e(n)$$

From LMS to Kernel-LMS

$$\Omega(n) = \underbrace{\Omega(0)}_{=0} + \mu \sum_{i=0}^{n-1} e(i)\Phi(Y(i))$$

- Now exploit the *kernel trick*:

$$\hat{x}(n) = \langle \Omega(n), \Phi(Y(n)) \rangle = \mu \sum_{i=0}^{n-1} e(i) \langle \Phi(Y(i)), \Phi(Y(n)) \rangle$$

From LMS to Kernel-LMS

$$\Omega(n) = \underbrace{\Omega(0)}_{=0} + \mu \sum_{i=0}^{n-1} e(i)\Phi(Y(i))$$

- Now exploit the *kernel trick*:

$$\hat{x}(n) = \langle \Omega(n), \Phi(Y(n)) \rangle = \mu \sum_{i=0}^{n-1} e(i) \langle \Phi(Y(i)), \Phi(Y(n)) \rangle$$

Kernel LMS Algorithm

$$\hat{x}(n) = \mu \sum_{i=0}^{n-1} e(i)\kappa(Y(i), Y(n))$$

Properties

- unique solution, because it solves a quadratic problem in feature space
- weights of the filter are never used explicitly
- order of the filter is not user controllable

present output $\hat{x}(n)$ is determined by:

- previous inputs $Y(n)$
- all previous errors $e(i)$

$\Rightarrow$ can be computed in input space

More Properties

Computational complexity:

- $\mathcal{O}(N)$ for each output sample, if errors are reasonable small after N samples

No need for regularization [2], [5]:

- because of the non-parametric improvement in the output samples (believed)

Stepsize μ controls the convergence, speed and misadjustment

- just like for the LMS-Algorithm
- μ is upper bounded by the largest eigenvalue of the data covariance
  - lies in feature space
  - difficult to estimate

Simulation

One-step ahead prediction of Mackey-Glass time series

Setup:

- normalized input data $\sigma^2 = 1$
- step size $\mu = 0.01$ for LMS
- $\mu = 0.5$ for KLMS

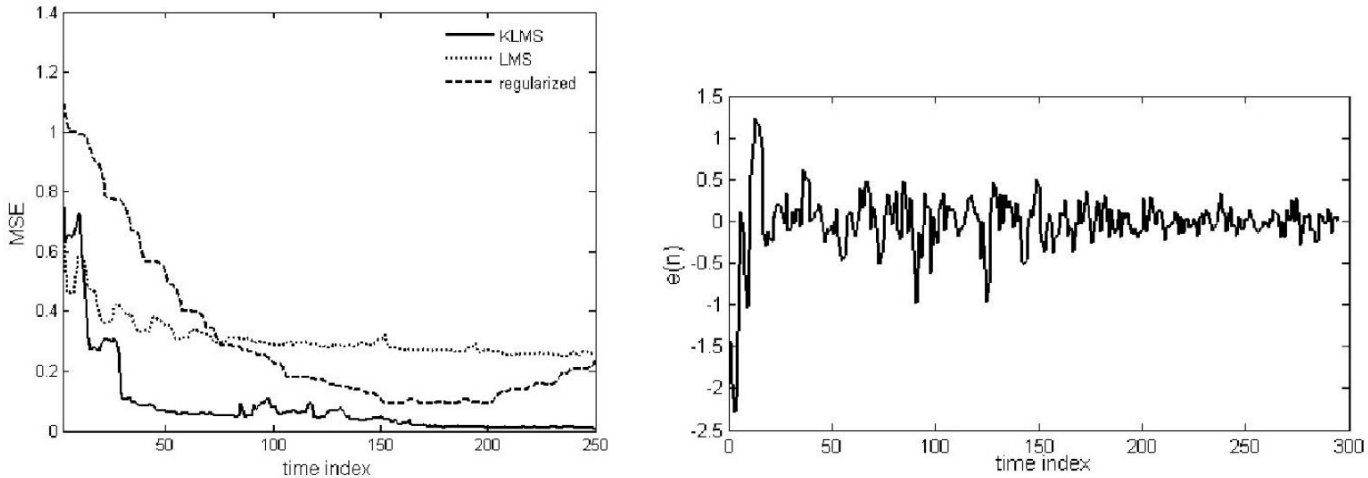

Figure: left: learning curves; right: error samples for KLMS [2]

Kernel-RLS

- **Basic Idea:** Extend the linear Least Squares (LS) method to a non-linear version by transforming the data into feature space
- consider the LS criterion

$$J = \min_{\theta} ||\mathbf{x} - \mathbf{Y}^T\theta||^2$$

with a given vector $\mathbf{x} \in \mathbb{R}^{N\times 1}$ and a data matrix $\mathbf{Y}^T \in \mathbb{R}^{N\times M}$ of observations, $\theta \in \mathbb{R}^{M\times 1}$ the solution vector

Kernel-RLS

- **Basic Idea:** Extend the linear Least Squares (LS) method to a non-linear version by transforming the data into feature space
- consider the LS criterion

$$J = \min_{\theta} ||\mathbf{x} - \mathbf{Y}^T\theta||^2$$

with a given vector $\mathbf{x} \in \mathbb{R}^{N\times 1}$ and a data matrix $\mathbf{Y}^T \in \mathbb{R}^{N\times M}$ of observations, $\theta \in \mathbb{R}^{M\times 1}$ the solution vector

- transform data to feature space, where $\theta' \in \mathbb{R}^{M'\times 1}$ and $\mathbf{Y}' \in \mathbb{R}^{N\times M'}$

$$J' = \min_{\theta'} ||\mathbf{x} - \mathbf{Y}'^T\theta'||^2$$

Kernel-RLS

- The transformed solution θ' can also be represented in the basis defined by the rows of the transformed matrix $\mathbf{Y}'$ with the solution vector $\alpha \in \mathbb{R}^{N \times 1}$ (high dim. of M')

$$\theta' = \mathbf{Y}'^T \alpha$$

- Introduce the kernel matrix $\mathbf{K} = \mathbf{Y}'\mathbf{Y}'^T$
- The LS problem in feature space:

$$J' = \min_{\alpha} ||\mathbf{x} - \mathbf{K}\alpha||^2$$

Kernel-RLS

- The transformed solution θ' can also be represented in the basis defined by the rows of the transformed matrix $\mathbf{Y}'$ with the solution vector $\alpha \in \mathbb{R}^{N \times 1}$ (high dim. of M')

$$\theta' = \mathbf{Y}'^T \alpha$$

- Introduce the kernel matrix $\mathbf{K} = \mathbf{Y}'\mathbf{Y}'^T$
- The LS problem in feature space:

$$J' = \min_{\alpha} ||\mathbf{x} - \mathbf{K}\alpha||^2$$

- Take usage of the kernel trick (Gaussian Kernel):

$$\mathbf{K}(i,j) = \kappa(\mathbf{Y}_i^T, \mathbf{Y}_j^T)$$

$\mathbf{Y}_i^T$ and $\mathbf{Y}_j^T$ are the i-th and j-th rows of $\mathbf{Y}^T$

Measures Against Overfitting

- Dimension of feature space M' often higher than the number of available data points N ($M' = \infty$ for Gaussian kernel)
- J' can have an infinite number of solutions
- Overfitting problem

One possible solution: Regularize the solution vector α

- Penalize the norm of θ'

Measures Against Overfitting

- Rewriting the problem:

$$J'' = \min_{\theta'} ||\mathbf{x} - \mathbf{Y}'^T\theta'||^2 + c||\theta'||^2 = \min_{\alpha} ||\mathbf{x} - \mathbf{K}\alpha||^2 + c\alpha^T\mathbf{K}\alpha$$

- with the solution

$$\alpha = \mathbf{K}_{reg}^{-1}\mathbf{x}$$

and

$$\mathbf{K}_{reg} = \mathbf{K} + c\mathbf{I}$$

c is a regularization constant

A sliding window approach

- Taking only the last N pairs of input-output pairs $\{(\mathbf{y}_1, x_1), (\mathbf{y}_2, x_2), \ldots\}$ into account

we get the regularized kernel matrix

$$\mathbf{K}_n = \mathbf{Y}_n \mathbf{Y}_n^T + c\mathbf{I}$$

$$\mathbf{x}_n = [x(n), x(n-1), \ldots, x(n-N+1)]$$

$$\mathbf{Y}_n = [\mathbf{y}(n), \mathbf{y}(n-1), \ldots, \mathbf{y}(n-N+1)]$$

- This reduces the complexity of the algorithm
- track changes without any extra burden

Updating the Kernel Matrix

- The updated solution: $\alpha_n = \mathbf{K}_n^{-1}\mathbf{x}_n$
- Update the Kernel Matrix $\mathbf{K}_n$ as follows:

$$\mathbf{K}_n = \begin{bmatrix} \hat{\mathbf{K}}_{n-1} & \mathbf{k}_{n-1}(\mathbf{x}_n) \\ \mathbf{k}_{n-1}(\mathbf{x}_n)^T & k_{nn} + c \end{bmatrix}$$

- where $\hat{\mathbf{K}}_{n-1}$ is $\mathbf{K}_{n-1}$ removed by the first row and column
  $\mathbf{k}_{n-1}(\mathbf{x}_n) = [\kappa(\mathbf{x}_{n-N+1}, \mathbf{x}_n), \ldots, \kappa(\mathbf{x}_{n-1}, \mathbf{x}_n)]$
  $k_{nn} = \kappa(\mathbf{x}_n, \mathbf{x}_n)$

Summary of the Kernel-RLS algorithm

Initialize $\mathbf{K}_0$ as $(1+c)\mathbf{I}$ and $\mathbf{K}_0^{-1}$ as $\mathbf{I}/(1+c)$
for $n = 1, 2, \ldots$ **do**

1. Obtain $\hat{\mathbf{K}}_{n-1}$ out of $\mathbf{K}_{n-1}$
2. Calculate $\hat{\mathbf{K}}_{n-1}^{-1}$
3. Obtain $\mathbf{K}_n$
4. Calculate $\mathbf{K}_n^{-1}$ (blackboard)
5. Obtain the updated solution $\alpha_n = \mathbf{K}_n^{-1}\mathbf{x}_n$

end for

Analysis of Kernel-RLS algorithm

- calculating $\mathbf{K}_n^{-1}$ explicitly would cost computationally and memory-wise $\mathcal{O}(N^3)$ operations for each window
- derive them from known matrices $\Rightarrow \mathcal{O}(N^2)$

Conclusion

- Basic algorithms LMS and RLS
- Kernel LMS
- Kernel RLS

References

 [1] **Håkan Hjalmarsson, Björn Ottersten** (2002). "Lecture Notes in Adaptive Signal Processing". Kungliga Tekniska Högskolan, Royal Institute of Technology, Department of Signals, Sensors and Systems

 [2] **Puskal P. Pokharel, Weifing Liu, Jose C. Principe** (2007). "Kernel LMS". Computational NeuroEngineering Laboratory, University of Florida, Department of Electrical and Computer Engineering, ICASSP 2007

 [3] **Bernhard Schölkopf, Alexander J. Smola** (2002). "Learning with Kernels". Massachusetts Institute of Technology (MIT)

References

 [4] **Steven van Vaerenbergh; Javier Via; Ignacio Santamaria** (2006). "A Sliding-Window Kernel RLS Algorithm and its Application to Non-linear Channel Identification". University of Cantabaria, Spain, Department of Communications Engineering, ICASSP 2006

 [5] **Weifing Liu, Puskal P. Pokharel,Jose C. Principe** (2008). "The Kernel Least-Mean-Square Algorithm". Computational NeuroEngineering Laboratory, University of Florida, Department of Electrical and Computer Engineering, 2008-February

References

 [6] **Christoph Krall** (2008). "Lecture Notes in Adaptive Systems". Graz University of Technology, Department of Signal Processing and Speech Communication Laboratory

 [7] **Ragnar Thobaben** (2009). "Lecture Notes in Advanced Digital Communications". Kungliga Tekniska Högskolan, Royal Institute of Technology, Department of Electrical Engineering