UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Proposta do Kernel Sigmoide (KSIG) e sua análise de convergência para a solução de problemas de filtragem adaptativa não linear

Éden Pereira da Silva

SÃO CRISTÓVÃO/ SE
2017

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Éden Pereira da Silva

Proposta do Kernel Sigmoide (KSIG) e sua análise de convergência para a solução de problemas de filtragem adaptativa não linear

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Carlos Alberto Estombelo Montesco
Co-orientador: Prof. Dr. Leonardo Nogueira Matos

SÃO CRISTÓVÃO/ SE
2017

Éden Pereira da Silva

Proposta do Kernel Sigmoide (KSIG) e sua análise de convergência para a solução de problemas de filtragem adaptativa não linear

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Carlos Alberto Estombelo Montesco, Presidente
Universidade Federal de Sergipe (UFS)

Prof. Dr. Leonardo Nogueira Matos, Co-Orientador
Universidade Federal de Sergipe (UFS)

Prof. Dr. Ewaldo E. C. Santana , Membro
Universidade Estadual do Maranhão (UEMA)

Prof. Dr. Jugurta Rosa Montalvão Filho, Membro
Universidade Federal de Sergipe (UFS)

Proposta do Kernel Sigmoide (KSIG) e sua análise de convergência para a solução de problemas de filtragem adaptativa não linear

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe, como parte dos requisitos necessários à obtenção do título de Mestre em Ciência da Computação.

São Cristóvão - SE, vinte e sete de janeiro de 2017

_____

Prof. Dr. Carlos Alberto Estombelo Montesco
Orientador

_____

Prof. Dr. Leonardo Nogueira Matos
Co-Orientador

_____

Prof. Dr. Ewaldo E. C. Santana
Membro

_____

Prof. Dr. Jugurta Rosa Montalvão Filho
Membro

Dedicatória

Aos meus pais, Ester e Sinval, verdadeiros guias nas minhas caminhadas na vida.

Agradecimentos

A Deus, pela oportunidade de concluir mais esta etapa da minha vida.

A CAPES/CNPQ, pela bolsa concedida durante o mestrado, essencial à realização deste trabalho.

Aos meus pais, pela ajuda durante este período na medida de suas possibilidades.

Aos meus irmãos, especialmente Éven por estar presente em muitos momentos dessa caminhada.

Aos meus familiares que também estiveram comigo nesta jornada.

Aos amigos da paróquia da Luz, especialmente Suzi pela ajuda na seleção do mestrado e ao padre João Bosco pelo acompanhamento neste período.

Aos meus companheiros e amigos da engenharia Joel e Reneilson, pela oportunidade de compartilhar nossos conhecimentos. A Reneilson, agradeço ainda pela ajuda na correção de muitos dos trabalhos que fizeram parte da minha jornada no mestrado, inclusive desta dissertação.

A meu orientador Prof. Dr. Carlos Estombelo-Montesco, que desde a graduação vem me ensinando sobre muitas coisas da vida acadêmica, dando-me oportunidades como a de desenvolver de modo autônomo, mas sempre sobre sua preciosa orientação, guiando-me por este caminho tão largo cujo um dos frutos foi este trabalho.

Ao professor Dr. Ewaldo Santana, que vem também desde a graduação auxiliando no desenvolvimento do trabalho desde o TCC até esta dissertação.

Ao prof. Dr. Leonardo Matos pelo auxílio na reta final deste trabalho dentre outros frutos deste tempo de estudos.

Aos colegas de mestrado e da administração do Laboratório, especialmente à amiga Luciana, grande companheira.

A todos aqueles que não citei, mas que à sua maneira contribuíram para à realização deste trabalho.

Resumo

A filtragem adaptativa é aplicada na solução de diversos problemas da engenharia. Há muitas alternativas para melhorá-la, uma delas é o uso de kernel e, em adição, o uso de um dicionário pré-definido de dados. Neste contexto, este trabalho apresenta o KSIG, a versão em *kernel* do algoritmo Sigmoide, um algoritmo que otimiza o erro do filtro pelo emprego de uma função de custo par e não linear. Ademais, é apresentada a versão do KSIG com dicionário de dados pré-definido, visando redução do grande número de dados utilizados para obtenção da saída decorrente do uso da técnica com *kernel*. A eficiência teórica do KSIG e de sua versão com dicionário pré-definido é um resultado presente nas provas de convergência construídas para ambos os algoritmos, as quais demonstraram que estes convergem em média. Já as curvas de aprendizagem obtidas nas simulações computacionais dos experimentos realizados demonstraram que o KSIG quando comparado ao KLMS, em diferentes problemas de filtragem adaptativa, apresenta convergência mais rápida, em menos iterações, tanto nas versões sem tanto com dicionário pré-definido de ambos os algoritmos.

Abstract

Adaptive filtering is applied as solution for many problems in engineer. There are many techniques to improve adaptive filtering as kernel methods and, in addiction, it is used a pre-tuned dictionary. In this context, here is presented the KSIG algorithm, the kernel version of Sigmoide, where is used the kernel, to decrease the error, and the non-linear and even cost function to increase the convergence speed. Here it is described also, the KSIG with a pre-tuned dictionary, to reduce the size of the data set used to calculate the filter output, which is a kernel method consequence . The KSIG and KSIG with pre-tuned dictionary theoretical efficiency is one result of their convergence proof, which evidence that the algorithms converge in average. The learning curves, which are results of some experiments, show that when KSIG and KLMS algorithms are compared, the first converges faster, in less iterations, than the second, in the version with and without pre-tuned dictionary of both algorithms.

Lista de Figuras

Lista de Siglas

APA - *Affine Project Algorithm*

KLMS - *Kernel Least Mean Square*

KMC - *Kernel Maximum Correntropy*

KSIG - *Kernel Sigmoide*

LMS - *Least Mean Square*

MSE - *Mean Square Error*

RKHS - *Reproducing Kernel Hilbert Space*

RLS - *Recursive Least Square*

SA - *Sigmoid Algorithm*

SVM - *Support Vector Machine*

Notação

1. Letras minúsculas em itálico são variáveis escalares. Exemplos: x, d, w, y;
2. Letras maiúsculas denotam constantes escalares. Exemplos: X, D ;
3. Letras minúsculas em negrito são vetores. Exemplos: $\mathbf{x}, \mathbf{w}$;
4. Todos os vetores (referidos no item anterior) são matrizes coluna, sem exceção. Exemplo: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$;
5. Letras maiúsculas em negrito são matrizes. Exemplos: $\mathbf{X}, \mathbf{R}$;
6. Parênteses indicam tempo ou iteração. Exemplos: $d(i), y(i)$;
7. Todas as variáveis são reais;
8. Subscrito é para índice em um vetor ou matriz, ou quando especificado, pode também indicar iteração.

Sumário

Capítulo 1

Introdução

A filtragem adaptativa é observada em diversas tarefas da engenharia, sendo solução de muitos problemas, principalmente de processamento de sinais, como a predição ou o cancelamento de ruído [Haykin 2014]. Para estes tipos de problema, é necessário que no processo de construção do filtro alguns de seus parâmetros se modifiquem para aproximar a saída deste de algum sinal de saída desejado, tarefa que não pode ser realizada quando se trata de filtros digitais mais comuns, cujos parâmetros são fixos, e não abrangem a complexidade desses tipos de atividades.

Dentre os algoritmos de filtragem adaptativa mais conhecidos, há o *Least Mean Square* (LMS), que tem como característica principal a facilidade de implementação e a eficiência (que é próxima do filtro adaptativo ótimo), o que ratifica o porquê da frequência com que é aplicado e estudado desde a sua criação em 1960 [Widrow and Stearns 1985].

O trabalho de [Santana et al. 2006b], traz o algoritmo Sigmoide, que tem uma função par e não linear como métrica de avaliação, ou função de custo, do erro na filtragem, substituindo o MSE (do inglês *mean square error*) utilizado no LMS. O resultado do Sigmoide foi um aumento na taxa de convergência, o que significa que o Sigmoide converge para a solução final em um número menor de iterações quando comparado ao LMS.

Todavia, tanto no LMS quanto no Sigmoide, a filtragem pressupõe a linearidade quanto ao modelo do filtro adaptativo. Alguns trabalhos, noutra perspectiva, apontam para técnicas com filtragem com modelo não linear, a partir do uso de funções positivas e simétricas, denominadas *kernel* [Liu et al. 2010]. Esta técnica lida com a não linearidade a partir dos dados transformados para um espaço de Hilbert gerado a partir da combinação de funções

kernel.

O *kernel* LMS (KLMS) é um dos exemplos da aplicação do *kernel* ("kernelização") construído a partir do algoritmo LMS [Liu et al. 2010]. Como o resultado da kernelização do LMS foi obtida diminuição do MSE no processo final de filtragem, o que significa que a saída do filtro após a fase de treinamento foi mais próxima da saída desejada no KLMS que no LMS [Liu et al. 2008].

As propostas do Sigmoide e do KLMS objetivaram a melhora da filtragem adaptativa respectivamente pela troca na função de avaliação e pela mudança de domínio dos dados para um espaço vetorial diferente. Uma alternativa para melhorar a filtragem seria utilizar ambas as técnicas, de modo a aproveitar a convergência mais rápida do Sigmoide e a queda no erro do trabalho com *kernel* do KLMS.

Esta alternativa consiste na kernelização do algoritmo Sigmoide, a qual denominaremos nesta dissertação de KSIG, que produz em hipótese, uma queda do erro como no KLMS, e uma convergência mais rápida que a encontrada neste, seguindo o princípio de que o Sigmoide é mais "rápido" que o LMS, o KSIG é também mais "rápido" em relação à convergência que o KLMS. Esta taxa de convergência mais rápida representaria uma melhora no contexto de aplicações cujo tempo de construção do filtro deve ser o mais curto possível.

Em complemento, dados alguns trabalhos recentes que fazem o uso de um dicionário de dados pré-definido no KLMS para lidar com o problema do alto custo associado ao cálculo da sua saída, é possível propor uma versão do KSIG com dicionário pré-definido, adaptando-o também a este tipo de problema.

Este trabalho tem como objetivo apresentar duas soluções para problemas de filtragem adaptativa, o KSIG, a versão *kernel* do Sigmoide e o KSIG em sua versão com dicionário pré-definido. Para este fim, nele estão contidas além das descrições dos algoritmos, suas análises de convergência, especificando sob qual condição cada um converge, e, na descrição de alguns experimentos em que se comparou o desempenho do KSIG com relação ao KLMS.

Para melhor compreensão desta dissertação, esta está organizado em capítulos, em que

- o capítulo 2 apresenta a fundamentação teórica de filtragem adaptativa, a técnica de uso de função *kernel* e como sua utilização na filtragem adaptativa;
- o capítulo 3 descreve trabalhos correlatos com alguns algoritmos para a solução de

problemas de filtragem adaptativa;

- no capítulo 4 é apresentado o algoritmo KSIG e descrita sua análise de convergência;
- o capítulo 5 traz a versão com dicionário pré-definido de dados do algoritmo KSIG, bem como a análise de convergência;
- no capítulo 6 são apresentados alguns experimentos comparando o desempenho do KSIG com o KLMS, bem como os resultados encontrados;
- e finalmente, no capítulo 7 são apresentadas as conclusões deste trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são tratados conceitos envolvidos na temática da filtragem adaptativa, o método do gradiente descendente, matriz de autocorrelação, correlação cruzada, o conceito de filtro adaptativo ótimo ou de Wiener; além disto, serão abordados o algoritmo LMS e o excesso de MSE na filtragem. Por fim, será apresentado o *kernel* e a técnica de "kernelização", dada a partir do truque do *kernel*, conceitos que serão alicerce para os capítulos seguintes.

2.1 Filtragem Adaptativa

A filtragem adaptativa consiste no processamento de sinais cujas propriedades estatísticas não são as mesmas para todas as entradas possíveis [Widrow and Stearns 1985]. Entretanto, mesmo que seja possível a adaptação, se a variação das propriedades estatísticas ocorrer em um período de tempo muito curto, a filtragem adaptativa pode não obter bons resultados, de modo que, para obtenção de resultados em filtros adaptativos é necessário haver certo período de estacionaridade dos dados o qual possibilite a adaptação no filtro [Sayed 2008].

Assim, para atender a variação das propriedades estatísticas, é necessária a adaptação dos parâmetros do filtro, os pesos, por um determinado período de tempo (treinamento), de modo que estes sejam os mais adequados possíveis (adaptados) às características do sinal de entrada, aproximando a saída do filtro da saída desejada. O esquema geral de um filtro adaptativo é ilustrado no diagrama de blocos na Figura 2.1.

É possível observar no diagrama de blocos na Figura 2.1 que há um sinal de entrada que é processado no filtro adaptativo, o qual gera uma saída que é dada como entrada a um bloco

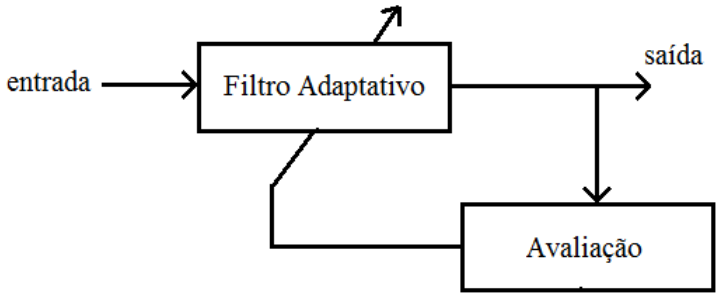

Figura 2.1: Ilustração genérica em diagrama de blocos do processo de filtragem adaptativa.

de avaliação. A saída deste último bloco é utilizada no filtro adaptativo para possíveis ajustes dos pesos, com objetivo de aproximar a saída do filtro adaptativo do valor ótimo da saída do filtro, ou seja, o que possui mínima diferença com relação ao sinal desejado para a saída.

A seta inclinada que atravessa o bloco do filtro na figura 2.1 indica que este é adaptativo, isto é, que seus pesos são ajustáveis, que podem ser modificados. Não obstante, o processo de adaptação é finito, ou seja, o filtro tem um período determinado para adaptação, para os ajustes dos pesos e melhora da filtragem, o que é similar a um treinamento em aprendizagem de máquina, que consiste no período em que um algoritmo tem que aprender a realizar a tarefa à qual é designado [Theodoridis and Koutroumbas 2008], de modo que a filtragem adaptativa é classificada como um tipo de aprendizagem supervisionada [Liu et al. 2010].

No processo de adaptação, ou período de aprendizagem, é necessário algum tipo de avaliação do filtro, a qual é realizada a partir de alguma função do erro[1], o qual é definido como a diferença entre a saída desejada e o resultado na saída do filtro,

$$e(i) = d(i) - y(i) \tag{2.1}$$

em que i é uma indicação de tempo discreto; $d(i)$ é o valor desejado para a saída; $y(i)$ é o valor de saída encontrado pelo filtro.

A saída do filtro adaptativo linear, $y(i)$, é definida como a relação entre a entrada dos dados e os pesos do filtro, sendo escrita como

$$y(i) = \sum_{k=1}^{n} x_k(i) w_k(i) = \mathbf{x}(i)^T \mathbf{w}(i) \tag{2.2}$$

[1] também conhecida como *função de custo*.

em que $\mathbf{x}(i) = [x_1(i), x_2(i), \ldots, x_n(i)]^T$ é um vetor (matriz coluna) de tamanho n que representa o dado n-dimensional de entrada e; $\mathbf{w}(i) = [w_1(i), w_2(i), \ldots, w_n(i)]^T$ é o vetor de pesos também de tamanho n.

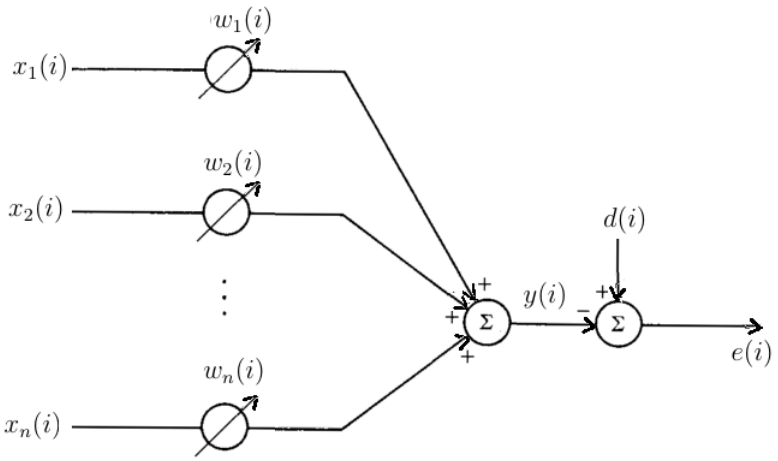

Figura 2.2: Diagrama de filtro adaptativo como combinação linear. Adaptado de [Widrow and Stearns 1985]

A Figura 2.2 é uma ilustração dessas relações entre a entrada, os pesos, a saída e o erro no filtro. Nesta figura, a saída do filtro $y(i)$ é expressa como o somatório dos produtos de cada elemento da entrada $x_k(i)$ por um peso $w_k(i)$ do filtro. Também o erro pode ser visto na figura como a diferença entre sinal desejado $d(i)$ e saída do filtro. Ao longo do processo de adaptação (i iterações de treino), o valor de cada w_k vai sendo modificado, para que o erro, se possível, diminua.

Conhecidas as relações e terminologia de filtros adaptativos, como a relação entre entrada, pesos e saída do filtro, na seção seguinte será apresentado o filtro linear ótimo sob o critério do erro quadrático médio.

2.1.1 Filtro linear ótimo

O filtro linear ótimo sob o critério do erro quadrático médio é aquele cujo erro é mínimo. Ele pode ser encontrado a partir de algumas manipulações das equações definidas na subseção anterior. Substituindo (2.2) em (2.1) tem-se

$$e(i) = d(i) - \mathbf{x}(i)^T\mathbf{w}(i) = d(i) - \mathbf{w}(i)^T\mathbf{x}(i) \tag{2.3}$$

Tomando o quadrado de (2.3) então

$$e(i)^2 = d(i)^2 + \mathbf{w}(i)^T\mathbf{x}(i)\mathbf{x}(i)^T\mathbf{w}(i) - 2d(i)\mathbf{x}(i)^T\mathbf{w}(i) \tag{2.4}$$

Se, por hipótese, assume-se que o valor do filtro está próximo do valor de convergência, isto significa que $\mathbf{w}(i)$, em média não varia muito ao longo de cada iteração i, assim, aplicando o operador expectância em (2.4), considerando a variação em cada iteração, obtém-se como resultado

$$E[e(i)^2] = E[d(i)^2] + \mathbf{w}(i)^T E[\mathbf{x}(i)\mathbf{x}(i)^T]\mathbf{w}(i) - 2E[d(i)\mathbf{x}(i)^T]\mathbf{w}(i) \tag{2.5}$$

A matriz $\mathbf{R_x}$, matriz de autocorrelação dos dados de entrada $\mathbf{x}(i)$, é definida como

$$\mathbf{R_x} = E[\mathbf{x}(i)\mathbf{x}(i)^T] \tag{2.6}$$

Já a matriz de correlação cruzada $\mathbf{p}$, por

$$\mathbf{p} = E[d(i)\mathbf{x}(i)] \tag{2.7}$$

Neste contexto, o erro quadrático médio, ou MSE, que se pode definir como $\xi \triangleq E[e(i)^2]$, é reescrito pela substituição de (2.6) e (2.7) em (2.5), como

$$\xi = E[d(i)^2] + \mathbf{w}(i)^T\mathbf{R_x}\mathbf{w}(i) - 2\mathbf{p}^T\mathbf{w}(i) \tag{2.8}$$

Dado este resultado, o gradiente de ξ em relação a $\mathbf{w}(i)$ é definido como

$$\nabla = \frac{\partial\xi}{\partial\mathbf{w}(i)} = \left[\frac{\partial\xi}{\partial w_1(i)}, \frac{\partial\xi}{\partial w_2(i)} \cdots, \frac{\partial\xi}{\partial w_n(i)}\right]^T = 2\mathbf{R_x}\mathbf{w}(i) - 2\mathbf{p} \tag{2.9}$$

No filtro ótimo o erro é mínimo, e o mínimo de ξ ocorre quando sua derivada em relação à $\mathbf{w}(i)$ é o vetor nulo $\mathbf{0}$, ou seja, quando o gradiente é $\mathbf{0}$. Considerando $\mathbf{w}_o$ como o vetor de pesos ótimo, então por (2.9)

$$\nabla = \mathbf{0} = 2\mathbf{R_x}\mathbf{w}_o - 2\mathbf{p} \tag{2.10}$$

Desenvolvendo (2.10) é possível obter a relação

$$\mathbf{0} = 2\mathbf{R_x}\mathbf{w}_o - 2\mathbf{p}$$

$$2\mathbf{R_x}\mathbf{w}_o = 2\mathbf{p}$$

$$\mathbf{R_x}\mathbf{w}_o = \mathbf{p}$$

e então, assumindo que a matriz de autocorrelação possui inversa, multiplicando ambos os lados deste último resultado por $\mathbf{R_x}^{-1}$, tem-se que

$$\mathbf{w}_o = \mathbf{R_x}^{-1}\mathbf{p} \tag{2.11}$$

Esta relação é conhecida na literatura como filtro de Wiener [Haykin 2014], sendo o filtro ótimo, com o mínimo possível de erro, definido como o produto do inverso da matriz de autocorrelação dos dados pela matriz de correlação cruzada.

Mesmo conhecendo a definição do filtro de Wiener, ou ótimo, este é na realidade um resultado teórico, não necessariamente alcançável, haja vista que a matriz de autocorrelação nem sempre é conhecida e o cálculo da sua inversa pode ser bastante custoso; além disto, não há como garantir a estacionaridade das propriedades estatísticas da mesma [Widrow and Stearns 1985, Haykin 2014]. Assim, os algoritmos de filtragem adaptativa levam a filtros que são aproximações do filtro de Wiener. Dentre estes algoritmos, está o LMS, o qual será apresentado na seção seguinte.

2.2 Least Mean Square

O LMS foi criado por Widrow e Hoff em 1960 e é um dos algoritmos de filtragem adaptativa mais conhecidos e utilizados devido à sua facilidade de implementação e aos seus resultados nas tarefas de filtragem [Haykin 2014].

A aproximação da solução ótima é feita pelo LMS, a partir do método do gradiente descendente, que toma como base o gradiente da função a qual se deseja minimizar (ou maximizar), função esta que no contexto de filtragem adaptativa é a função de avaliação do filtro, dada com relação ao erro, que é definido como a diferença entre sinal desejado para a saída e o valor na saída do filtro.

Pelo método do gradiente descendente, a cada passo, no LMS, o vetor de pesos é atualizado a partir de seu valor atual, acrescido de algum percentual do gradiente da função de avaliação, conforme a equação

$$\mathbf{w}(i+1) = \mathbf{w}(i) - \mu\nabla(i) \tag{2.12}$$

em que μ é conhecido como tamanho do passo, ou taxa de aprendizagem, e está no geral, dentro do intervalo de $(0, 1]$.

O tamanho do passo representa o quanto do gradiente é tomado em conta a cada atualização. Ele é um parâmetro experimental mas tem, como se verá ainda nesta seção, uma faixa de valores possíveis na qual é garantida a convergência do algoritmo.

O método do gradiente descendente adotado no LMS, em sua fase de atualização, leva em consideração apenas o erro atual, ou seja, o vetor de pesos é atualizado a partir da função de avaliação sobre o erro no tempo i, não levando em conta os erros anteriores. Com esta característica, a cada iteração, o erro pode indicar uma direção não determinística no gradiente, por isto, diz-se que o LMS utiliza um método de gradiente descendente estocástico, em que a cada novo passo o cálculo do gradiente é independente, não determinado por qualquer passo anterior [Liu et al. 2010, Haykin 2014].

Tomando a função de avaliação como o MSE, o gradiente instantâneo (da iteração) é definido como

$$\begin{aligned}\hat{\nabla}(i) &= \frac{\partial}{\partial \mathbf{w}(i)}\big(e(i)^2\big) = \frac{\partial}{\partial \mathbf{w}(i)}\big(d(i) - \mathbf{w}(i)^T\mathbf{x}(i)\big)^2 \\ &= -2(d(i) - \mathbf{w}(i)^T\mathbf{x}(i))\mathbf{x}(i) \\ \hat{\nabla}(i) &= -2e(i)\mathbf{x}(i) \end{aligned} \tag{2.13}$$

Então, substituindo (2.13) em (2.12) tem-se o algoritmo LMS

$$\mathbf{w}(i+1) = \mathbf{w}(i) - 2\mu e(i)\mathbf{x}(i) \tag{2.14}$$

A dependência na atualização do vetor de pesos em relação à entrada no LMS é evidenciada por (2.14). Não obstante, considerando o filtro próximo da região de convergência, o valor do vetor de pesos oscilará em torno de algum valor, seja o do vetor ótimo ou não, podendo-se assumir a independência em relação aos dados de entrada quando próximo do estado estacionário [Widrow and Stearns 1985].

Considerando que na região de convergência há independência em relação aos dados de entrada, na prova da convergência, obteve-se como resultado da análise uma relação desta com a taxa de aprendizagem [Haykin 2014], de modo que, em média, o algoritmo LMS converge se

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.15}$$

em que λ_{max} é o maior autovalor da matriz de autocorrelação $\mathbf{R}_\mathbf{x}$. Assim, é garantida a convergência do LMS quando a taxa de aprendizagem está no intervalo indicado em (2.15).

Além disto, se seu valor for muito pequeno a solução dada pelo algoritmo pode convergir mais lentamente, mas se for muito alto, esta pode oscilar e não se aproximar da melhor solução [Sayed 2008].

Mesmo em média convergindo para uma solução, o LMS é uma aproximação do filtro de Wiener cujo erro é mínimo, de modo que há um excesso de erro no LMS com relação ao erro do filtro ótimo e é sobre este excesso que trata a próxima subseção.

2.2.1 Excesso

No LMS, enquanto o vetor de pesos for diferente do vetor ótimo, $\mathbf{w} \neq \mathbf{w}_o$, haverá um excesso de erro, isto significa que, considerando o erro (MSE) como ξ e no filtro de Wiener como ξ_{min} (erro mínimo possível), enquanto $\xi > \xi_{min}$ haverá um excesso do MSE, definido [Widrow and Stearns 1985] como

$$ExcessoMSE = E[\xi_i - \xi_{min}] = \mu E[{n_i}^2] tr\{\mathbf{R_x}\} \tag{2.16}$$

em que o subíndice i indica a i-ésima iteração; n se refere ao ruído no sistema [Santana 2006]; μ é a taxa de aprendizagem e $tr\{\bullet\}$ é a função traço da matriz [2], que é a soma dos elementos da diagonal principal.

O desajuste, M, é uma medida comumente utilizada na literatura de filtragem adaptativa [Haykin 2014], ele é definido como a relação entre o excesso e o erro mínimo

$$M = \frac{E[\xi_i - \xi_{min}]}{\xi_{min}} \tag{2.17}$$

Considerando que o erro mínimo advém do ruído [Santana 2006], ou seja, $\xi_{min} = E[n_i^2]$, então, o desajuste em (2.17) é reescrito como

$$M_{LMS} = \mu tr\{\mathbf{R_x}\} \tag{2.18}$$

A curva de aprendizagem consiste no gráfico que apresenta o erro ao longo das iterações, durante a fase de treinamento do algoritmo. O excesso do MSE é observado na curva de aprendizagem do algoritmo LMS, como ilustrado na Figura 2.3. A métrica, que neste caso é

[2] O traço é utilizado como parâmetro, substituindo o maior autovalor λ_{max}, da matriz de autocorrelação $\mathbf{R_x}$, dado que $tr\{\mathbf{R}\} > \lambda_{max}$ e por ser mais simples o cômputo do traço da matriz de que o cálculo de seus autovalores [Liu et al. 2010].

o MSE do LMS, oscila em torno de uma curva de aprendizagem ideal, a qual será definida a seguir.

Na Figura 2.3, a curva de aprendizagem ideal, pontilhada, pode ser representada por uma função exponencial, enquanto a curva em linha contínua é a do LMS, que oscila em torno da curva ideal, esta oscilação é a ilustração do excesso do MSE.

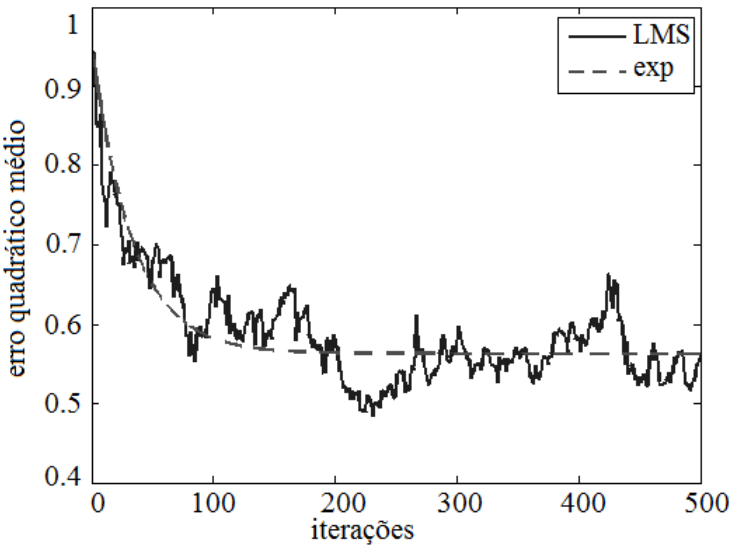

Figura 2.3: Curva de aprendizagem típica do LMS e aprendizagem ótima

A equação exponencial que modela a curva ideal do LMS é da forma $e^{-\frac{t}{\tau_n}}$, em que t se refere ao tempo (iteração) e τ é uma constante de tempo, que mantém uma relação estrita com o n-ésimo autovalor de $\mathbf{R_x}$ (λ_n), descrita segundo [Widrow and Stearns 1985] como

$$\tau_n = \frac{1}{2\mu\lambda_n} \tag{2.19}$$

O conceito de excesso é utilizado em um modelo de prova de convergência de algoritmos de filtragem adaptativa [Yousef and Sayed 2001], o que mostra sua importância e razão pela qual foi abordada neste capítulo. Além disto, o desajuste será utilizado como parâmetro noutra parte do próximo capítulo, por isso abordagem desses tópicos nesta seção.

A partir das definições contidas nesta e nas subsecções anteriores, fez-se uma revisão sobre filtragem adaptativa linear, filtro linear ótimo, métricas de avaliação, excesso de erro e desajuste, temas importantes para a proposta a ser apresentada. A seção seguinte trata de outra temática essencial ao trabalho aqui apresentado, as funções *kernel* e as definições concernentes ao uso da técnica que utiliza os espaços vetoriais gerados por estas funções.

2.3 Kernel

Esta seção trata brevemente sobre *kernel*, sua definição e propriedades, e por fim faz um apanhado geral dos espaços de Hilbert gerados por *kernel*.

2.3.1 Propriedades

Um *kernel* é uma função $\kappa : (\mathbb{X} \times \mathbb{X}) \to \mathbb{R}$, com $\mathbb{X} \subseteq \mathbb{R}^N$, tal que, dados $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$, valem as propriedades [Liu et al. 2010]

1. $\kappa(\mathbf{x}, \mathbf{x}') > 0$ (positividade)
2. $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}', \mathbf{x})$ (simetria)

Os tipos mais comuns de função *kernel* são

1. gaussiano: $\kappa(\mathbf{x}, \mathbf{x}') = exp\left(\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right), \quad \sigma > 0$
2. polinomial: $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^l, \quad l \geq 2$

2.3.2 Truque do *kernel*

Um espaço de Hilbert é um espaço vetorial completo e com produto interno definido [Debnath and Mikusiński 2005]. Existe a possibilidade de um espaço deste tipo ser gerado a partir de uma combinação linear de funções *kernel*, se o kernel possuir propriedade de reprodução [Liu et al. 2010].

Segundo o teorema de Moore-Aronszajn, um *kernel* reprodutor pode gerar um único RKHS (*reproducing kernel Hilbert space*, espaço de Hilbert reproduzido por *kernel*) e um RKHS é formado por apenas um *kernel* reprodutor [Santana Júnior 2012]. Isto significa que o espaço formado pela combinação linear de funções *kernel* é único e que todas operações sobre um RKHS refletem apenas sobre a função *kernel* que é a geradora do espaço.

Para um RKHS $\mathbb{H}$ gerado pelo *kernel* κ, há um elemento $g_{\mathbf{x}}(\bullet) \in \mathbb{H}$, tal que

$$g_{\mathbf{x}}(\bullet) = \kappa(\mathbf{x}, \bullet),$$

definido como representante de $\mathbf{x}$, de modo que para todo $f \in \mathbb{H}$ é válida a propriedade

$$\langle g_{\mathbf{x}}, f \rangle = f(\mathbf{x}), \quad \forall f \in \mathbb{H} \tag{2.20}$$

em que $\langle \bullet, \bullet \rangle$ é o produto interno de $\mathbb{H}$.

Como $g_{\mathbf{x}} \in \mathbb{H}$, então, para cada $\mathbf{x}' \in \mathbb{X}$

$$g_{\mathbf{x}}(\mathbf{x}') = \langle g_{\mathbf{x}}, g_{\mathbf{x}'} \rangle \tag{2.21}$$

Pela definição de representante, mostrada anteriormente, é possível escrever

$$g_{\mathbf{x}}(\mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}'). \tag{2.22}$$

Substituindo a igualdade de (2.22) em (2.21),

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle g_{\mathbf{x}}, g_{\mathbf{x}'} \rangle \tag{2.23}$$

Este resultado é conhecido na literatura como "truque do *kernel*" [Parreira et al. 2012]. A partir dele, fica explícito que é possível escrever o produto interno de dados do espaço de Hilbert a partir do *kernel* aplicado nos dados, os quais, os elementos deste mesmo espaço de Hilbert representam. O que significa que, a partir do truque do *kernel*, é possível se trabalhar com produto interno de um RKHS sem lidar com dados neste espaço e sim, com os dados ainda no espaço original como entrada da função *kernel*, a qual representa o produto interno dos dados no RKHS.

Para utilizar o *kernel*, e mais especificamente o espaço de Hilbert por ele gerado, é então necessário expressar os dados, ou mesmo o próprio algoritmo que se quer "kernelizar", como produto interno, e pelo truque do *kernel* substituir o produto interno pela função *kernel* aplicada nos dados ainda no espaço original, utilizando indiretamente a não linearidade dos dados no RKHS apenas com a aplicação do *kernel*.

Esta é a descrição da técnica de "kernelização", a qual permite observar como são estruturados os algoritmos em *kernel* existentes, além de possibilitar também o cabedal para a proposta de novos algoritmos em *kernel*, cuja exigência é expressá-los em termos de produtos internos do RKHS.

2.4 Considerações Finais

A filtragem adaptativa é importante sobretudo em atividades em que há variabilidade das propriedades estatísticas dos dados. O filtro adaptativo ótimo é uma construção teórica, a

aproximação deste filtro é feita a partir de algoritmos como o LMS, cuja faixa de valores para sua convergência está atrelada ao valor escolhido para a taxa de aprendizagem.

O excesso e o desajuste são formas de aferir a diferença entre o erro ótimo e o erro encontrado ao longo das iterações. A partir do truque do *kernel* é possível utilizar o poder de uma representação num espaço mais complexo sem lidar com os dados neste espaço mas, com sua representação do produto interno obtida pela função *kernel*, ainda no espaço original dos dados.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta alguns trabalhos relacionados à área de filtragem adaptativa, nele estão descritos os algoritmos Sigmoide e KLMS, e também, tópicos relativos à análise de convergência destes.

3.1 Algoritmo Sigmoide

O erro a cada iteração, como descrito no capítulo 2, é definido como a diferença entre sinal esperado e o obtido na saída do filtro

$$e(i) = d(i) - \mathbf{w}(i)^T \mathbf{x}(i) \tag{3.1}$$

O termo $F(e(i))$ é definido como a função de avaliação (ou custo) do filtro. O gradiente instantâneo, $\hat{\nabla}(i)$, da função de avaliação é definido como

$$\begin{aligned} \hat{\nabla}(i) &= \frac{\partial}{\partial \mathbf{w}(i)} F(e(i)) = F'(e(i)) \frac{\partial e(i)}{\partial \mathbf{w}(i)} = F'(e(i)) \frac{\partial [d(i) - \mathbf{w}(i)^T \mathbf{x}(i)]}{\partial \mathbf{w}(i)} \\ &= -F'(e(i)) \mathbf{x}(i) \end{aligned} \tag{3.2}$$

Denotando por $f = F'$, o algoritmo geral para atualização dos pesos pode ser definido como

$$\mathbf{w}(i+1) = \mathbf{w}(i) - \mu f(e(i)) \mathbf{x}(i) \tag{3.3}$$

No caso do MSE, $f(e(i)) = e(i)$.

Em seu trabalho, Santana [Santana 2006] propôs um modelo de filtragem com método do gradiente descendente com F como uma função par e não linear, cuja superfície obtida oferece maior velocidade de convergência no que tange a encontrar o vetor de pesos ótimo, ou

seja, melhora o tempo da filtragem adaptativa e diminui o excesso de MSE. Para exemplificar este seu modelo, o autor propõe o algoritmo Sigmoide (SA, do inglês *Sigmoid Algorithm*), no qual,

$$F(e(i)) = ln(cosh(\alpha e(i))) \tag{3.4}$$

em que α é uma constante que altera a inclinação da curva gerada por $ln(cosh(\bullet))$. O gradiente instantâneo de (3.4) é definido como

$$\frac{\partial}{\partial \mathbf{w}(i)} F(\alpha e(i)) = -\alpha tanh(\alpha e(i))\mathbf{x}(i) \tag{3.5}$$

Substituindo (3.5) então em (3.3)

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu\alpha tanh(\alpha e(i))\mathbf{x}(i) \tag{3.6}$$

Como no caso do LMS, a convergência está relacionada à taxa de aprendizagem, de modo que, por [Santana 2006], para garantir a convergência do Sigmoide,

$$0 < \mu < \frac{2}{\alpha tanh(\alpha\lambda_{max}) - \alpha^2 tanh(\alpha\lambda_{max}) sech(\alpha\lambda_{max})\sigma_n^2} \tag{3.7}$$

No que se refere ao desajuste, no SA, algoritmo Sigmoide, é dado por

$$M = \frac{\mu E[tanh^2(\alpha n_k)] tr(\mathbf{R_x})}{2E[sech^2(\alpha n_k)]\sigma_n^2} \tag{3.8}$$

Para comparar o desempenho do Sigmoide com o do LMS, [Santana 2006] considerou que ambos tinham igual desajuste (M), o que permitiu que se estabelecesse uma relação entre as taxas de aprendizagem dos algoritmos. Por (2.17) e o resultado em (3.8), considerando iguais os desajustes

$$\begin{aligned} \mu_{LMS} tr(\mathbf{R_x}(i)) &= \frac{\mu_{SA} E[tanh^2(\alpha n_k)] tr(\mathbf{R_x})}{2E[sech^2(\alpha n_k)]\sigma_n^2} \\ \Rightarrow \mu_{SA} &= \frac{2E[sech^2(\alpha n_k)]\sigma_n^2]\mu_{LMS}}{\mu E[tanh^2(\alpha n_k)]} \end{aligned} \tag{3.9}$$

Os experimentos realizados em [Santana 2006, Santana et al. 2006b] atestam o desempenho superior quanto a velocidade de convergência do Sigmoide em relação ao LMS em duas tarefas de filtragem adaptativa, a primeira foi a identificação de uma planta e a segunda a determinação do componente determinístico de impedância cardiográfica.

O resultado de um dos experimentos está ilustrado na Figura 3.1, que se refere à curva de aprendizagem da modelagem da planta de um sistema, descrito pela função de transferência

$P(z) = 0.2037z^{-1} + 0.5926z^{-2} + 0.2037z^{-3}$. Nesta modelagem foram executadas 100 simulações de Monte Carlo. Sendo $\mu_{LMS} = 0.03$, e μ_{SA} dado por (3.9). A partir desta figura é possível observar a convergência mais rápida do Sigmoide em relação ao LMS, pois o primeiro atinge a região estacionária entre 600 e 800 iterações, já o segundo, em 1200 iterações aproximadamente.

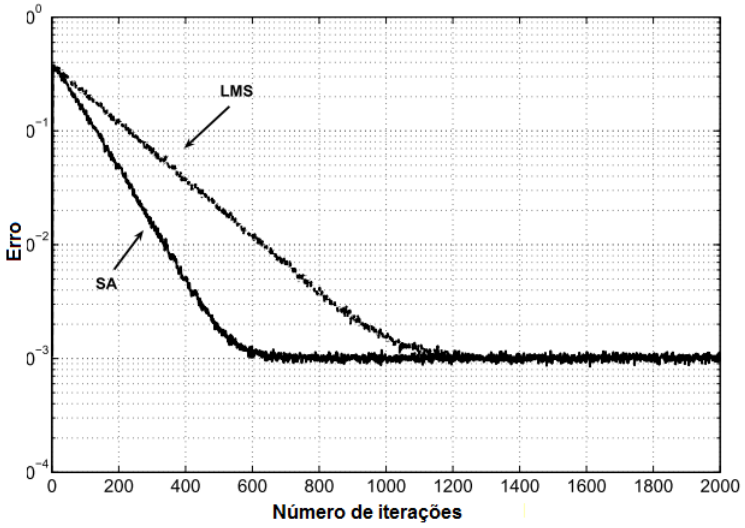

Figura 3.1: Curva de aprendizagem do LMS e do Sigmoide. Disponível em [Santana 2006]

Como descrito e explicitado pelo resultado supracitado, a mudança na função de avaliação provocou uma melhora no processo de filtragem adaptativa em termos do tempo de convergência. Outro modo de melhorar a filtragem é pelo uso da técnica de kernelização, é sobre esta técnica que versa a próxima seção.

3.2 Algoritmos *kernel*

A facilidade de lidar com o produto interno de um espaço de Hilbert sem a necessidade de trabalhar com os dados neste espaço, além da fundamentação matemática sólida da técnica baseada em funções *kernel*, a tornou uma técnica bastante utilizada [Liu et al. 2008]. Em filtragem adaptativa, o trabalho de [Liu et al. 2010] traz uma série de algoritmos em suas versões *kernel*, dentre eles o LMS, o RLS e o Adaline.

O uso de *kernel* é observado em aplicações de reconhecimento de padrões quando se objetiva por exemplo, a classificação não linear de dados. É o caso de aplicações com má-

quina de vetor de suporte (SVM) em que se faz uso do *kernel* [Theodoridis and Koutroumbas 2008].

No que se refere a uma formalização do uso da técnica, o trabalho de [Chen and Zhang 2007] traz uma descrição geral do que é a kernelização, que consiste em lidar com o algoritmo original, mas aplicado sobre os dados transformados tomados seus produtos internos, representados em elementos da matriz de Gram.

A matriz de Gram, $\mathbf{G}$, é definida como a matriz cujos elementos são o produto interno de algum conjunto de dados. Considerando $\{\mathbf{x}_i\}_{i=1}^{n}$ um conjunto de n vetores de entrada, cada elemento g_{ij} de $\mathbf{G}_{n\times n}$ é definido como $g_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Pelo truque do *kernel* (2.23), se os dados estiverem num RKHS, então $g_{ij} = \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

Segundo [Chen and Zhang 2007], se $\mathbf{F}(\mathbf{X})$ for o algoritmo original $\mathbf{F}$ aplicado nos dados $\mathbf{X}$ de entrada, o algoritmo kernelizado nada mais é o que $\mathbf{F}(\mathbf{G}(\mathbf{X}))$, em que $\mathbf{G}(\mathbf{X})$ é a matriz de Gram dos dados em $\mathbf{X}$. Isto significa que o algoritmo kernelizado ainda é o algoritmo original, mas aplicado na matriz de Gram dos dados de entrada.

Um exemplo desta relação é ilustrado em [Santana Júnior 2012], em que há a proposta da extração cega de sinais com uso da técnica de kernelização, trabalhando com o resultado diretamente escrito a partir da matriz de Gram dos dados transformados, cuja solução final possui similaridade com a encontrada a partir do algoritmo sem o *kernel*, uma solução que se assemelha ao algoritmo inicial em linhas mais gerais, mas aplicado na matriz de Gram dos dados no RKHS.

Outra aplicação da técnica de kernelização é o KLMS [Liu et al. 2008], e é sobre ele que trata a próxima seção.

3.3 KLMS

O algoritmo KLMS (ou *kernel* LMS), de [Liu et al. 2008], é a aplicação da técnica de kernelização ao algoritmo LMS, podendo ser definido também como o LMS em um RKHS.

Para se obter o KLMS são necessários alguns passos. Tomando a fórmula do LMS em

(2.14) e ignorando a constante[1], a equação de atualização do LMS pode ser reescrita como

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu e(i)\mathbf{x}(i) \tag{3.10}$$

Transformando então (3.10) para um dado do RKHS, tem-se a atualização do vetor de pesos redefinida como

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \mu e(i)\boldsymbol{\varphi}(i) \tag{3.11}$$

em que $\boldsymbol{\omega}(\bullet)$ e $\boldsymbol{\varphi}(\bullet)$ representam respectivamente o vetor de pesos ($\mathbf{w}(\bullet)$) e o dado de entrada ($\mathbf{x}(\bullet)$) no RKHS e $\boldsymbol{\varphi}(i) = \boldsymbol{\varphi}(\mathbf{x}(i))$, para simplificar a notação. Expandindo (3.11)

$$\begin{aligned}
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) + \mu e(i)\boldsymbol{\varphi}(i) \\
&= \boldsymbol{\omega}(i-1) + \mu e(i-1)\boldsymbol{\varphi}(i-1) + \mu e(i)\boldsymbol{\varphi}(i) \\
&\vdots \\
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(0) + \mu \sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)
\end{aligned} \tag{3.12}$$

Considerando que $\boldsymbol{\omega}(0) = \mathbf{0}$,

$$\boldsymbol{\omega}(i+1) = \mu \sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j) \tag{3.13}$$

A saída y no KLMS a partir do vetor de pesos (3.13) é definida como

$$\begin{aligned}
y(i) &= \boldsymbol{\omega}(i)^T \boldsymbol{\varphi}(i) \\
&= \mu \sum_{j=1}^{i-1} e(j)\boldsymbol{\varphi}(j)^T \boldsymbol{\varphi}(i)
\end{aligned} \tag{3.14}$$

Este resultado é uma soma de produtos internos do RKHS. A partir do truque do *kernel* (2.23), este somatório em (3.14) pode ser reescrito como

$$y(i) = \mu \sum_{j=1}^{i-1} e(j)\kappa(\mathbf{x}(j), \mathbf{x}(i)) \tag{3.15}$$

Considerando o filtro depois da fase de treino a sua saída para cada entrada é expressa como

$$f_i(\mathbf{x}) = \sum_{i=1}^{n} a(i)\kappa(\mathbf{x}(i), \mathbf{x}) \tag{3.16}$$

[1]Como a taxa de aprendizagem é ajustável experimentalmente e está multiplicando o valor 2, então este número se torna irrelevante, podendo-se adotar, por exemplo $\mu' = 2\mu$

em que n é o tamanho do dicionário de dados, em que este dicionário consiste no conjunto de dados utilizado para determinar a saída, cuja cardinalidade pode ser menor ou igual ao número de dados utilizados no treinamento [Gao et al. 2015, Parreira et al. 2012]; cada $a(i) = \mu(e(i))$ é um dos coeficientes da combinação linear que forma o elemento de saída, dada pelo somatório de cada $a(i)\kappa(\mathbf{x}(i), \mathbf{x})$.

Os resultados do algoritmo KLMS [Liu et al. 2008] apresentaram a queda nos valores de erro no processo de filtragem adaptativa, ou seja, na diferença entre sinal desejado e saída do filtro, se comparado o KLMS com o LMS. Em [Liu et al. 2010], por exemplo, o KLMS e o LMS foram utilizados na equalização de um canal não linear[2], como resultado, o algoritmo kernelizado apresentou um MSE menor na região de convergência, o que pode ser visto na Figura 3.2. Enquanto, o MSE de um está entre 0.6 e 0.4, o do outro está entre 0 e 0.2 o que evidencia superioridade quanto a filtragem adaptativa (menor erro) do KLMS em relação ao LMS na equalização de um canal não linear.

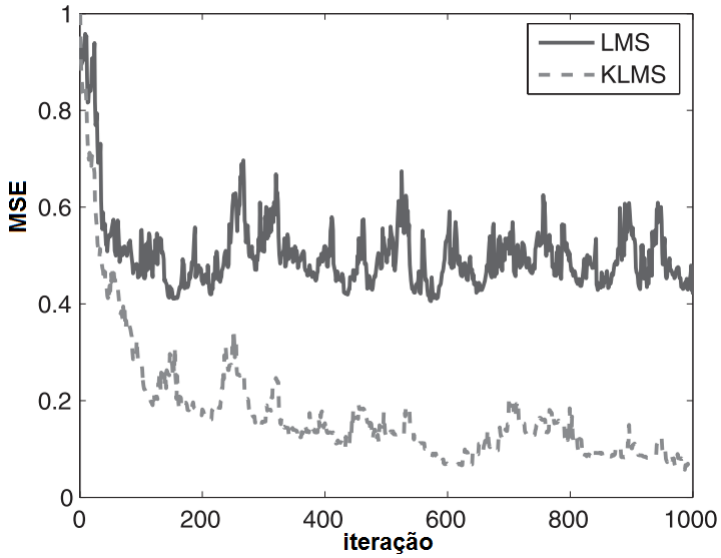

Figura 3.2: Curvas de aprendizagem do LMS e do KLMS para uma tarefa de equalização de um canal não linear. [Liu et al. 2010]

Alguns trabalhos trazem melhores resultados quanto à aplicação do KLMS em atividades como filtragem adaptativa online, [Gao et al. 2015], no cancelamento de ruído [Bao and Panahi 2009] ou na predição [Liu et al. 2010]. Enquanto alguns apresentam outros algoritmos

[2]cuja definição está na subseção 6.2.2.

em versão *kernel* [Zhao et al. 2011, Liu et al. 2010], e é sobre estes trabalhos que trata a seção seguinte.

3.3.1 Outras propostas

Mesmo com resultados já significativos quanto à diminuição da taxa de erro, alguns trabalhos propõem a melhora do algoritmo KLMS através do controle do tamanho do dicionário [Chen et al. 2014] ou por algum tipo de normalização feita nos dados [Takizawa et al. 2015, Constantin and Lengellé 2013].

O dicionário consiste no conjunto de vetores de entrada utilizados para determinar o valor de saída do KLMS (3.16), isto é, cada $\mathbf{x}(i)$ contido no somatório e que é utilizado na fase posterior a de treino para se obter o valor de saída. O uso do dicionário pré-definido e inicialmente limitado (menor que o total do conjunto de treino) é uma proposta de redução do custo de cálculo na saída.

Se o conjunto de dados de treino é muito grande, pode haver *overffiting* [Liu et al. 2010], além de poder tornar o processo de cálculo da saída muito custoso, no caso do dicionário conter todos os elementos do conjunto, o que é agravado pelo fato de que neste cálculo são computadas operações sobre alguma função *kernel*, que geralmente envolve operação de potenciação, tanto no *kernel* gaussiano quanto no polinomial, os kerneis mais utilizados.

Uma solução possível para esse problema é a inclusão de parte do conjunto de dados de treino e não todo o conjunto. A seleção de qual dado é incluído ou não no dicionário é feito sobre algum critério de relevância [Liu et al. 2010]. É a partir desta solução que alguns trabalhos propõem a melhora do KLMS [Gao et al. 2015, Zhao et al. 2015, Constantin and Lengellé 2013].

Além do dicionário, alguns trabalhos desenvolveram técnicas que lidam com o espalhamento e com risco empírico [Constantin and Lengellé 2013]. Algumas técnicas da literatura fazem uso de estimadores estatísticos [Liu et al. 2015], outras fazem a quantização a partir de distância euclidiana no espaço de entrada dos dados [Pokharel and Principe 2014]. Nas subseções seguintes serão apresentados exemplos de algoritmos baseados no KLMS.

3.3.2 Kernel Maximum Correntropy

O trabalho de [Zhao et al. 2011], inspirado no KLMS, propõe o *kernel Maximum Correntropy* (KMC), o qual utiliza como função de custo o critério de máxima correntropia, o que é bastante similar à proposta do Sigmoide: alterar a função de custo para aprimorar os resultados na filtragem adaptativa.

A correntropia consiste em um método alternativo para de estimar probabilisticamente a similaridade entre variáveis aleatórias [Zhao et al. 2011]. No KMC a função de custo adotada é uma função *kernel* $\kappa_\sigma(d(i), \boldsymbol{\omega}(i-1)^T\boldsymbol{\varphi}(i))$, a qual corresponde à medida de similaridade da saída desejada com a saída do filtro.

No KMC, pelo algoritmo do gradiente descente estocástico, a atualização dos pesos é dada por

$$
\begin{aligned}
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) + \mu\frac{\partial\kappa_\sigma(d(i), \boldsymbol{\omega}(i-1)^T\boldsymbol{\varphi}(i))}{\partial\boldsymbol{\omega}(i-1)} \\
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) + \mu\exp\left(\frac{-e^2(i)}{2\sigma^2}\right)e(i)\boldsymbol{\varphi}(i), \qquad \text{para } \kappa_\sigma(\mathbf{a},\mathbf{b}) = \exp\left(\frac{-\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right) \\
&= \boldsymbol{\omega}(i-1) + \mu\exp\left(\frac{-e^2(i-1)}{2\sigma^2}\right)e(i-1)\boldsymbol{\varphi}(i-1) + \mu\exp\left(\frac{-e^2(i)}{2\sigma^2}\right)e(i)\boldsymbol{\varphi}(i) \\
&\vdots \\
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(0) + \mu\sum_{j=1}^{i}\left[\exp\left(\frac{-e^2(j)}{2\sigma^2}\right)e(j)\boldsymbol{\varphi}(j)\right] \\
\boldsymbol{\omega}(i+1) &= \mu\sum_{j=1}^{i}\left[\exp\left(\frac{-e^2(j)}{2\sigma^2}\right)e(j)\boldsymbol{\varphi}(j)\right], \qquad \text{assumindo } \boldsymbol{\omega}(0) = \mathbf{0}
\end{aligned}
\tag{3.17}
$$

A saída do algoritmo do KMC, $y(i) = \boldsymbol{\omega}(i)^T\boldsymbol{\varphi}(i)$, dado (3.17)

$$
\begin{aligned}
y(i) &= \mu\sum_{j=1}^{i-1}\left[\exp\left(\frac{-e^2(i)}{2\sigma^2}\right)e(j)\boldsymbol{\varphi}(j)^T\right]\boldsymbol{\varphi}(i) \\
&= \mu\sum_{j=1}^{i-1}\left[\exp\left(\frac{-e^2(i)}{2\sigma^2}\right)e(j)\kappa\left(x(j), x(i)\right)\right]
\end{aligned}
\tag{3.18}
$$

A saída do KMC (3.18) possui semelhanças com a saída do KLMS (3.16), assim como a forma como ela é encontrada também apresenta similaridade com relação ao modo como ocorreu em (3.12). No que se refere aos resultados, estes foram melhores para o KMC, como ilustrado em [Zhao et al. 2011].

O uso da máxima correntropia como função de custo possibilitou a melhora na filtragem adaptativa com decaimento do erro quando comparado com o KLMS. Esta mudança na função de custo é a mesma observada em [Santana et al. 2006b], mas lidando com filtragem adaptativa não linear utilizando *kernel*.

Na subseção seguinte, será apresentado o algoritmo de [Chen et al. 2014], que apresenta uma solução baseada no KLMS com uso de dicionário pré-definido.

3.3.3 KLMS com dicionário pré-definido

Os estudos de [Gao et al. 2015,Chen et al. 2014,Parreira et al. 2012] apresentam uma versão do KLMS com um dicionário pré-definido, observando principalmente o uso excessivo de memória caso não seja utilizado algum critério de seleção de quais dados deverão estar no dicionário.

A saída do KLMS, definida em (3.16), pode ser reescrita em notação vetorial como

$$f(\mathbf{x}) = \mathbf{a}^T \boldsymbol{\kappa}_{\mathbf{x}} \tag{3.19}$$

em que $\mathbf{a} = [a(1), a(2), \ldots, a(n)]^T$ e $\boldsymbol{\kappa}_{\mathbf{x}} = [\kappa(\mathbf{x}(1), \mathbf{x}), \kappa(\mathbf{x}(2), \mathbf{x}), \ldots, \kappa(\mathbf{x}(n), \mathbf{x})]^T$. Considerando que há um dicionário pré-definido de dados, $\boldsymbol{\omega} = [\mathbf{x}_{\boldsymbol{\omega}(1)}, \mathbf{x}_{\boldsymbol{\omega}(2)}, \ldots, \mathbf{x}_{\boldsymbol{\omega}(n)}]$, para otimizar o valor da saída do filtro, é necessário encontrar o melhor conjunto de coeficientes $a(j)$, ou seja, o vetor $\mathbf{a}$, que minimize o erro, diferença entre a saída desejada e a obtida em (3.19). Isto pode ser feito a partir da regra de atualização dos coeficientes [Chen et al. 2014]

$$\mathbf{a}(i) = \mathbf{a}(i-1) + \mu e(i) \boldsymbol{\kappa}_{\boldsymbol{\omega}}(\mathbf{x}(i)) \tag{3.20}$$

em que $\boldsymbol{\kappa}_{\boldsymbol{\omega}}(\mathbf{x}(i)) = [\kappa(\mathbf{x}_{\boldsymbol{\omega}(1)}, \mathbf{x}(i)), \kappa(\mathbf{x}_{\boldsymbol{\omega}(2)}, \mathbf{x}(i)), \ldots, \kappa(\mathbf{x}_{\boldsymbol{\omega}(n)}, \mathbf{x}(i))]^T$.

Os experimentos apresentados em [Gao et al. 2015, Chen et al. 2014, Parreira et al. 2012] ilustram que o KLMS com dicionário pré-definido alcança resultados nas simulações próximos do que se espera em teoria, como ilustrado na Figura 3.3.

3.4 Análise de Convergência

O estudo da convergência de algoritmos de filtragem adaptativa comumente é realizado levando em consideração a estatística do filtro (algum momento estatístico), pelo fato da en-

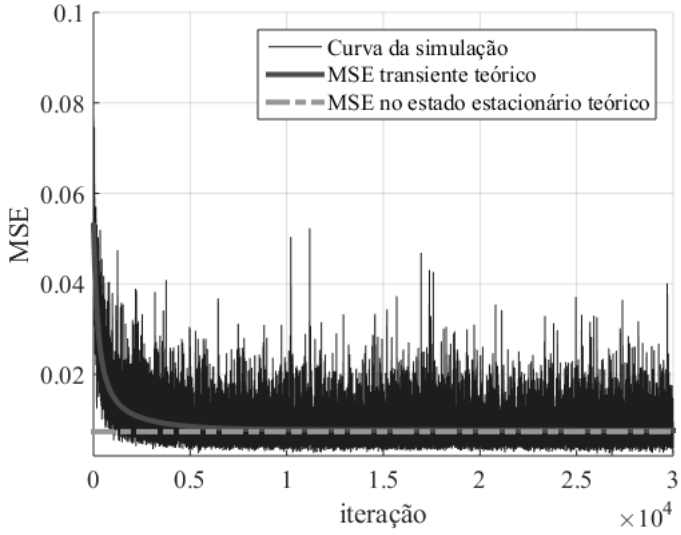

Figura 3.3: Curva de aprendizagem do KLMS com dicionário pré-definido para identificação de sistema. Adaptado de [Gao et al. 2015]

trada deste tipo de algoritmo ser uma variável aleatória, de modo que não é possível considerar um modelo determinístico para análise da convergência.

É possível encontrar vários estudos sobre a convergência do LMS e de suas variantes, em [Haykin 2014], por exemplo, há uma prova da convergência do LMS na média, isto significa que, em média, o algoritmo converge para a solução ótima.

A análise de convergência do LMS é feita a partir de (3.10), sendo necessário assumir algumas hipóteses de independência estatística do vetor de pesos em relação à entrada. O resultado obtido na análise é a relação entre a taxa de aprendizagem (μ) e o maior autovalor da matriz de autocorrelação, como expresso em (2.15). Isto significa que se μ estiver no intervalo $(0, 2/\lambda_{max}[$, em média, o LMS converge para a solução ótima [3].

Quanto à convergência do algoritmo Sigmoide, esta é detalhada em [Santana 2006] e segue em linhas gerais o modelo de análise do LMS, considerando por outro lado, não o MSE, mas uma função par não linear como métrica de avaliação. O resultado, assim como no LMS, está associado à taxa de aprendizagem, de modo que o Sigmoide converge como apresentado em (3.7) se o tamanho do passo estiver em $\left(0, \frac{2}{\alpha tanh(\alpha\lambda_{max}) - \alpha^2 tanh(\alpha\lambda_{max}) sech(\alpha\lambda_{max})\sigma_n^2}\right[$.

No que se refere ao KLMS entretanto, o número de trabalhos com estudo sobre a con-

[3] mais detalhes em [Haykin 2014] ou [Widrow and Stearns 1985]

vergência é bem mais reduzido, nas bases *IEEExplorer*, *Scopus* e *WebOfScience* foram encontrados alguns trabalhos sobre esta temática, com dois trabalhos fazendo um estudo mais completo [Chen et al. 2012a] e [Parreira et al. 2012]. Em ambos os casos a análise se baseou no emprego do *kernel* gaussiano. O segundo trabalho fez análise no *kernel* polinomial também. Alguns dos outros trabalhos encontrados derivavam inclusive deste segundo trabalho.

O estudo de [Chen et al. 2012a] traz uma abordagem sobre a convergência do KLMS baseada no trabalho de [Yousef and Sayed 2001], que propôs uma análise de filtros adaptativos tomando um número menor de hipóteses, o que torna a análise mais simplificada e, de certo modo, mais genérica. Para isto, [Yousef and Sayed 2001] propõem o estudo em termos da energia do filtro adaptativo, analisando tanto o estado transiente quanto o *steady state* da energia do filtro sobre algum momento estatístico.

Somente o KLMS com *kernel* gaussiano é analisado no trabalho de [Chen et al. 2012a]. Neste, é tomada a definição da regra de adaptação como definida em (3.11), considerando a convergência do algoritmo em relação a média ao quadrado (segundo momento estatístico). Como no estudo do LMS, são feitas algumas hipóteses sobre parâmetros e sobre o filtro (vetor de pesos). Como resultado do trabalho de [Chen et al. 2012a] é representada a convergência em função da taxa de aprendizagem, que deve ser escolhida de modo que

$$\mu \leq \frac{2E[e_a^2(i)]}{E[e_a^2(i)] + \sigma_\nu^2} \tag{3.21}$$

em que σ_ν^2 é a variância do ruído de distúrbio [4]; já $e_a(i)$ é o erro *a priori* da iteração i, definido como

$$\begin{aligned} e_a(i) &= [\boldsymbol{\omega}^* - \boldsymbol{\omega}(i)]^T \boldsymbol{\varphi}(i) \\ &= \tilde{\boldsymbol{\omega}}(i)^T \boldsymbol{\varphi}(i) \end{aligned} \tag{3.22}$$

em que $\tilde{\boldsymbol{\omega}}(i)$ é o vetor de erro de pesos, a diferença entre o vetor de pesos ótimo no RKHS $\boldsymbol{\omega}^*$ é o vetor de pesos na i-ésima iteração, $\boldsymbol{\omega}(i)$. O erro *a priori* é o produto do vetor de erro de pesos pelo vetor de dados na i-ésima iteração.

Um resultado desta primeira análise é a convergência do KLMS no sentido da média ao quadrado, se a taxa de aprendizagem for menor que a razão no lado direito de (3.21). Outro

[4] Este conceito é melhor definido no modelo de filtragem adaptativa contido na subseção 4.2.1.

resultado encontrado foi a relação que o tamanho (σ) do *kernel* exerce sobre o algoritmo, mesmo não afetando diretamente a convergência, como visto em (3.21), ele exerce certa importância no desempenho do KLMS [Chen et al. 2012a].

O segundo trabalho, [Parreira 2012], faz análise de convergência do KLMS considerando também (3.11), mas em seu desenvolvimento faz uso de um número maior de hipóteses. O estudo explora tanto o *kernel* gaussiano quanto o polinomial, além disto, pressupõe que há um dicionário de dados pré-definido. Em sua análise, este autor leva em conta o teorema dos discos de Gerschgorin e a teoria da estabilidade de sistemas de controle, aplicando-os sobre uma matriz $\mathbf{G}$, que é a matriz que tem origem na aplicação do *kernel* no dicionário de dados. A estabilidade (convergência) do KLMS está relacionada aos autovalores de $\mathbf{G}$, de modo que, se dentro do intervalo de $(-1, 1[$, garantem a estabilidade.

O resultado dos estudos em [Parreira 2012] traz a análise do comportamento estocástico do algoritmo, em que é expressa uma relação com a taxa de aprendizagem e a convergência, relacionando-a à matriz $\mathbf{G}$, bem como a convergência a partir do quadrado da média do vetor de erro de pesos, o qual é expresso também em função de $\mathbf{G}$. Deste estudo deriva o trabalho [Parreira et al. 2012], o qual é fundamento para outros trabalhos disponíveis nas bases de pesquisa consultadas, como [Chen et al. 2014, Paul and Ogunfunmi 2012].

A análise da convergência do algoritmo KMC como o trabalho de [Chen et al. 2012a] é baseada no modelo de [Yousef and Sayed 2001]. Enquanto os outros algoritmos que trazem o KLMS com dicionário pré-definido fazem análise baseada no trabalho de [Parreira et al. 2012], como citado anteriormente.

Na Figura 3.4 há um diagrama que faz relaciona as duas análises e alguns dos trabalhos aqui citados.

3.5 Considerações finais

Este capítulo tratou de alguns temas pertinentes ao desenvolvimento de um dos objetivos deste trabalho. O primeiro se refere ao algoritmo Sigmoide, o qual apresentou uma melhora no desempenho na filtragem em termos do tempo de convergência, quando comparado com o LMS, a partir da mudança da função de avaliação.

O segundo tema exposto foi um estudo breve de kernelização, o que permitiu expor por

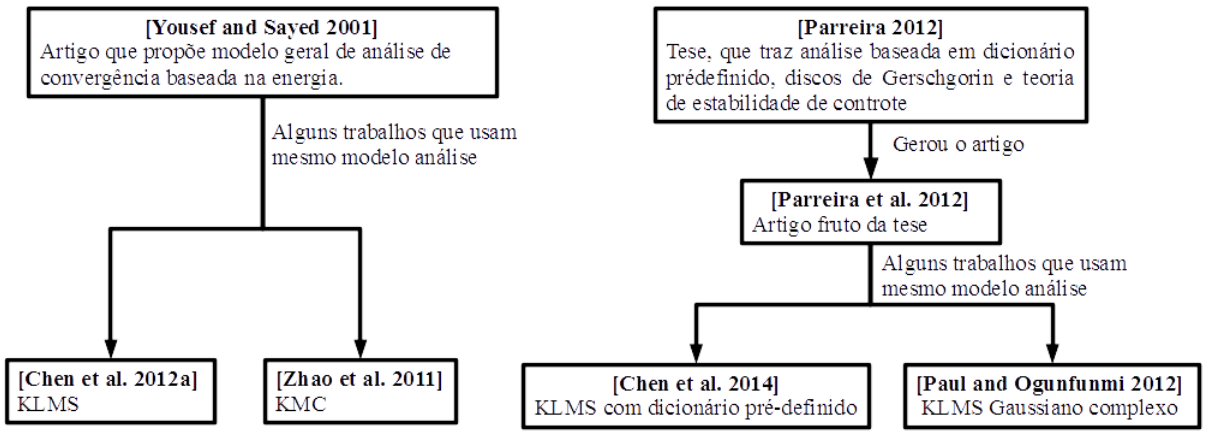

Figura 3.4: Diagrama de trabalhos e artigos derivados.

conseguinte o KLMS que é o LMS em um RKHS. Além do KLMS, foi exposto o KMC que apresenta, como o Sigmoide, uma mudança na função de custo. O resultado nos algoritmos apresentados foi o decréscimo do erro na filtragem adaptativa, e no caso do KMC, uma redução se comparada ao KLMS. Ainda foi apresentada uma versão com dicionário pré-definido do KLMS, observando o ganho desta técnica quanto ao uso de muitos dados no cálculo da saída.

Por fim, fez-se um breve apanhado acerca de convergência, cuja análise se dá, em linhas gerais, a partir do estudo de alguma característica estatística do filtro. Em complemento, deram-se alguns exemplos dos resultados das análises de convergência para o LMS, Sigmoide, KLMS e o KMC.

Capítulo 4

Kernel Sigmoide - KSIG

Este capítulo apresenta o cerne deste trabalho, o algoritmo proposto e o estudo sobre a convergência do mesmo, estudo este que é uma caracterização geral do algoritmo, como proposta que possibilita ou não a solução de problemas de filtragem adaptativa. E, como ponto inicial, a próxima seção traz a derivação do *kernel* Sigmoide.

4.1 KSIG

O KSIG, ou *kernel* Sigmoide, de [Silva et al. 2015], é a versão *kernel* do algoritmo Sigmoide de [Santana 2006]. A sua derivação é bastante similar à utilizada para chegar ao KLMS. A equação de atualização do vetor de pesos do Sigmoide é como descrita em (3.6)

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu\alpha tanh(\alpha e(i))\mathbf{x}(i) \tag{4.1}$$

A qual é escrita em um RKHS como

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \mu\alpha \tanh(\alpha e(i))\boldsymbol{\varphi}(i) \tag{4.2}$$

Considerando $\eta = \mu\alpha$, como em (3.12) é possível expandir (4.2)

$$\begin{aligned} \boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) + \eta \tanh(\alpha e(i))\boldsymbol{\varphi}(i) \\ &= \boldsymbol{\omega}(i-1) + \eta \tanh(\alpha e(i-1))\boldsymbol{\varphi}(i-1) + \eta \tanh(\alpha e(i))\boldsymbol{\varphi}(i) \\ &\vdots \\ \boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(0) + \eta \sum_{j=1}^{i} [\tanh(\alpha e(j))\boldsymbol{\varphi}(j)] \end{aligned} \tag{4.3}$$

Considerando que o valor inicial do vetor de pesos é $\boldsymbol{\omega}(0) = \mathbf{0}$, então (4.3) é reescrito como

$$\boldsymbol{\omega}(i+1) = \eta \sum_{j=1}^{i} [\tanh(\alpha e(j))\boldsymbol{\varphi}(j)] \tag{4.4}$$

A saída no RKHS com adição do resultado de (4.4) é definida como

$$\begin{aligned} y(i) &= \boldsymbol{\varphi}(i)^T \boldsymbol{\omega}(i-1) = \boldsymbol{\varphi}(i)^T \eta \sum_{j=1}^{i-1} [\tanh(\alpha e(j))\boldsymbol{\varphi}(j)] \\ &= \eta \sum_{j=1}^{i-1} [\tanh(\alpha e(j))\boldsymbol{\varphi}(i)^T \boldsymbol{\varphi}(j)] \end{aligned} \tag{4.5}$$

Aplicando então o truque do *kernel* (2.23) em (4.5), tem-se

$$\begin{aligned} y(i) &= \eta \sum_{j=1}^{i-1} [\tanh(\alpha e(j))\boldsymbol{\kappa}(\mathbf{x}(i), \mathbf{x}(j))]) \\ &= \sum_{j=1}^{i-1} [c_j \boldsymbol{\kappa}(\mathbf{x}(i), \mathbf{x}(j))] \end{aligned} \tag{4.6}$$

em que $c_j = \eta \tanh(\alpha e(j))$.

Ao fim do treinamento, a saída do algoritmo, dada uma entrada $\mathbf{x}$, é definida como

$$f(\mathbf{x}) = \sum_{j=1}^{n} [c_j \boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}(j))] \tag{4.7}$$

em que cada $\mathbf{x}(j)$ é um elemento do dicionário obtido durante a fase de treinamento.

Para realizar experimentos sobre o KSIG, é preciso dispor de um conjunto de treino. No treinamento, é formado o dicionário a partir de dados de entrada de treino, assim como o erro correspondente a cada entrada, que forma cada constante c_j.

Na Figura 4.1 pode-se observar o pseudocódigo do algoritmo KSIG. A entrada do algoritmo é o conjunto de dados de treino, os valores desejados, o fator de inclinação α e o tamanho do passo μ. Já a saída do algoritmo é o conjunto de constantes cujos valores são definidos no comando na linha 13.

Todavia, esta definição da saída pressupõe que todos os dados de entrada serão utilizados na saída, não sendo necessário o retorno dos próprios vetores de treino contido em $\mathbf{X}$. Caso fosse adotado algum critério de seleção dos dados a compor o dicionário, o retorno do algoritmo conteria além do vetor com as constantes c, o conjunto de vetores que compõe o dicionário.

1: **function** KSIG(X, d, α, μ)
2: $\eta \leftarrow \alpha \times \mu$
3: $n \leftarrow$ size(d) ▷ número de dados de treino
4: $e \leftarrow$ zerovector(n) ▷ erro, um vetor de zeros
5: $y \leftarrow$ zerovector(n) ▷ vetor de saída
6: $c \leftarrow$ zerovector(n) ▷ vetor de coeficientes
7: $c(1) \leftarrow \eta \times \tanh(\alpha \times e(1))$
8: **for** $j = 2, \ldots, n$ **do** ▷ treinamento
9: **for** $i = 1, \ldots, j-1$ **do**
10: $y(j) \leftarrow y(j) + c(i) \times \boldsymbol{\kappa}(X(:,i), X(:,j))$
11: **end for**
12: $e(j) \leftarrow d(j) - y(j)$
13: $c(j) \leftarrow \eta \times \tanh(\alpha \times e(j))$
14: **end for**
15: *return* c
16: **end function**

Figura 4.1: Pseudocódigo do algoritmo KSIG.

Ainda sobre o pseudocódigo na Figura 4.1, no que se refere a complexidade, considerando unitário o custo da atribuição nas linhas de código 2, 3 e 7, bem como as criações de vetores nas linhas 4, 5, 6; o laço na linha de código 8 faz com que as operações de custo unitário nas linhas 12 e 13 sejam executadas $(n-1)$ vezes; já o laço na linha 9 faz com que a atribuição na linha 10, considerando-a de custo unitário[1], seja executada uma vez na primeira iteração do laço na linha 8, duas na segunda, e assim por diante, o que resulta numa progressão aritmética de 1 a (n-1), cuja soma resulta em $n(n-1)/2$, que é o custo total de 10. Somando todos os custos, tem-se

$$1+1+1+1+1+1+n(n-1)/2+(n-1) = (n^2+n+10)/2$$

que é limitado por $O(n^2)$ e deste modo, a complexidade do algoritmo KSIG é $O(n^2)$, em que n é o número de vetores de treino.

[1]O que pode não ser verdade se a linguagem não tiver um tratamento eficaz para o cálculo de operações sobre matriz.

Para o cálculo da saída (4.5), considerando que o custo do cálculo da função *kernel* é unitário, o gasto resultante é de $O(m)$, em que m é o tamanho do dicionário de dados.

O algoritmo KSIG foi derivado a partir do Sigmoide de modo bastante similar ao KLMS, o cálculo de sua complexidade apresentou seu custo quadrático, e custo linear com relação a obtenção da saída. Na seção seguinte será apresentada a análise da convergência do algoritmo.

4.2 Análise de Convergência do KSIG

A análise de convergência visa validar a estabilidade do algoritmo, seja ela feita em estado estacionário, supondo que o algoritmo está na região de convergência, ou no estado transiente, observando o comportamento do algoritmo ao longo das iterações [Sayed 2008, Yousef and Sayed 2001].

Como destacado em [Sayed 2008, Yousef and Sayed 2001] a análise de estado transiente é complexa e usualmente não adotada. De modo que a análise de convergência do KSIG se baseará num modelo de análise de estado estacionário, que é uma forma menos complexa, mas válida, para atestar a efetividade do algoritmo. Além disto, a análise será feita considerando o kernel Gaussiano.

Por conveniência, a análise aqui desenvolvida é a mesma adotada para o KMC, dada a similaridade deste algoritmo com o próprio KSIG. Entretanto, antes de iniciá-la, é preciso esclarecer algumas definições essenciais que serão utilizadas durante a prova da convergência. É sobre estas definições que trata a seção seguinte.

4.2.1 Definições

No capítulo (2) foi definido o erro, $e(i)$, na filtragem adaptativa, como função da saída desejada e da saída do filtro. Esta definição de (2.1) pode ser reescrita no RKHS como

$$e(i) = d(i) - \boldsymbol{\omega}^T(i-1)\boldsymbol{\varphi}(i) \tag{4.8}$$

O objetivo no KSIG, como em qualquer outro algoritmo de filtragem adaptativa, é minimizar o erro. Este objetivo pode ser traduzido como encontrar o melhor vetor de pesos, o

mais próximo possível do vetor ótimo, $\boldsymbol{\omega}_o$, o qual está na definição do sinal desejado pela relação

$$d(i) = \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) + v(i) \tag{4.9}$$

em que $v(i)$ é o ruído de distúrbio (do inglês *disturbance noise*), que é consequência do erro no modelo, sendo inerente ao filtro [Haykin 2014]. Caso o filtro seja ótimo, ou seja, $\boldsymbol{\omega}(i-1) = \boldsymbol{\omega}_o$, então, tomando (4.8) e (4.9)

$$\begin{aligned} e(i) &= \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) + v(i) - \boldsymbol{\omega}^T(i-1)\boldsymbol{\varphi}(i) \qquad (4.10)\\ &= \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) + v(i) - \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) \\ e(i) &= v(i) \qquad (4.11) \end{aligned}$$

isto significa que o erro é o mínimo possível, o ruído de distúrbio, o qual não pode ser eliminado do modelo.

O vetor de erro dos pesos (*weight error vector*) é uma medida de quão distante o vetor de pesos está em relação ao vetor ótimo [Widrow and Stearns 1985]. Esta medida é definida pela diferença do filtro ótimo pelo valor atual do filtro, $\tilde{\boldsymbol{\omega}}(i) \triangleq \boldsymbol{\omega}_o - \boldsymbol{\omega}(i)$. É possível escrever o erro como função desta medida a partir de (4.10)

$$\begin{aligned} e(i) &= \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) + v(i) - \boldsymbol{\omega}^T(i-1)\boldsymbol{\varphi}(i) \\ &= \boldsymbol{\omega}_o^T \boldsymbol{\varphi}(i) - \boldsymbol{\omega}^T(i-1)\boldsymbol{\varphi}(i) + v(i) \\ e(i) &= \tilde{\boldsymbol{\omega}}^T(i-1)\boldsymbol{\varphi}(i) + v(i) \qquad (4.12) \end{aligned}$$

Há na literatura [Sayed 2008] dois tipos erros que são dados como função do vetor de erro dos pesos, o erro *a priori* ($e_a(i)$) e o erro *a posteriori* ($e_p(i)$), definidos como

$$e_a(i) = \tilde{\boldsymbol{\omega}}^T(i-1)\boldsymbol{\varphi}(i) \tag{4.13}$$

$$e_p(i) = \tilde{\boldsymbol{\omega}}^T(i)\boldsymbol{\varphi}(i) \tag{4.14}$$

E, a partir desta definição de erro *a priori* é possível reescrever (4.12) como

$$e(i) = e_a(i) + v(i) \tag{4.15}$$

A equação de atualização do filtro no KSIG em (4.2) pode ser reescrita para fim de simplificação como

$$\boldsymbol{\omega}(i) = \boldsymbol{\omega}(i-1) + \eta g[e(i)]\boldsymbol{\varphi}(i) \tag{4.16}$$

em que $g[e(i)] \triangleq \tanh(\alpha e(i))$. Então, subtraindo $\boldsymbol{\omega}_o$ de ambos os lados de (4.16)

$$\begin{aligned}
\boldsymbol{\omega}(i) - \boldsymbol{\omega}_o &= \boldsymbol{\omega}(i-1) + \eta g[e(i)]\boldsymbol{\varphi}(i) - \boldsymbol{\omega}_o \\
\boldsymbol{\omega}_o - \boldsymbol{\omega}(i) &= \boldsymbol{\omega}_o - \boldsymbol{\omega}(i-1) - \eta g[e(i)]\boldsymbol{\varphi}(i) \\
\tilde{\boldsymbol{\omega}}(i) &= \tilde{\boldsymbol{\omega}}(i-1) - \eta g[e(i)]\boldsymbol{\varphi}(i)
\end{aligned} \tag{4.17}$$

tem-se assim a equação de atualização do vetor de erro dos pesos. Utilizando (4.17) na definição do erro *a posteriori* em (4.14), obtém-se a relação

$$\begin{aligned}
e_p(i) &= \tilde{\boldsymbol{\omega}}^T(i)\boldsymbol{\varphi}(i) \\
&= \left[\tilde{\boldsymbol{\omega}}(i-1) - \eta g[e(i)]\boldsymbol{\varphi}(i)\right]^T \boldsymbol{\varphi}(i) \\
e_p(i) &= \tilde{\boldsymbol{\omega}}^T(i-1)\boldsymbol{\varphi}(i) - \eta g[e(i)]\boldsymbol{\varphi}^T(i)\boldsymbol{\varphi}(i)
\end{aligned} \tag{4.18}$$

Pela definição do erro *a priori* em (4.13), este resultado é reescrito como

$$e_p(i) = e_a(i) - \eta g[e(i)]\kappa(\mathbf{x}(i), \mathbf{x}(i)) \tag{4.19}$$

em que na última linha foi utilizado o truque do *kernel* [Liu et al. 2010]. Uma consequência de (4.19) é que

$$\begin{aligned}
e_p(i) - e_a(i) &= -\eta g[e(i)]\kappa(\mathbf{x}(i), \mathbf{x}(i)) \\
g[e(i)] &= -\frac{[e_p(i) - e_a(i)]}{\eta\kappa(\mathbf{x}(i), \mathbf{x}(i))}
\end{aligned} \tag{4.20}$$

Substituindo (4.20) em (4.17), tem-se

$$\tilde{\boldsymbol{\omega}}(i) = \tilde{\boldsymbol{\omega}}(i-1) + \frac{[e_p(i) - e_a(i)]}{\kappa(\mathbf{x}(i), \mathbf{x}(i))}\boldsymbol{\varphi}(i) \tag{4.21}$$

Esta expressão ilustra que é possível escrever o vetor de erro dos pesos em função dos erros *a priori* e *a posteriori*.

Conhecidas algumas definições de erro, seja sobre filtragem ou acerca do próprio vetor de pesos, a próxima seção trás a análise da convergência baseada na conservação da energia.

4.2.2 Relação da conservação de energia

A análise de convergência de algoritmos baseado na relação de conservação da energia foi desenvolvida por [Yousef and Sayed 2001] e é utilizado em vários trabalhos da literatura

[Chen et al. 2012a,Chen et al. 2012b,Zhao et al. 2011]. Por conveniência e semelhança com o trabalho de [Zhao et al. 2011] esta será a metodologia adotada para estudo da convergência do KSIG. Além disto, será considerado o kernel Gaussiano.

Para encontrar a energia, é necessário obter a norma do vetor [Sayed 2008], a qual é definida como $\|\mathbf{x}\|^2 \triangleq \mathbf{x}^T\mathbf{x}$, em que $\mathbf{x}$ é um vetor qualquer. A partir desta definição, tomando (4.21) e aplicando a norma para obter a energia,

$$\begin{aligned}
\|\tilde{\boldsymbol{\omega}}(i)\|^2 &= \left\|\tilde{\boldsymbol{\omega}}(i-1) + \frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}\boldsymbol{\varphi}(i)\right\|^2 \\
&= \|\tilde{\boldsymbol{\omega}}(i-1)\|^2 + \frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}\boldsymbol{\varphi}^T(i)\tilde{\boldsymbol{\omega}}(i-1) \\
&+ \frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}\tilde{\boldsymbol{\omega}}^T(i-1)\boldsymbol{\varphi}(i) + \left\|\frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}\boldsymbol{\varphi}(i)\right\|^2 \\
&= \|\tilde{\boldsymbol{\omega}}(i-1)\|^2 + 2\frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}e_a(i) + \frac{[e_p(i)-e_a(i)]^2}{[\kappa(\mathbf{x}(i),\mathbf{x}(i))]^2}\overbrace{\|\boldsymbol{\varphi}(i)\|^2}^{\kappa(\mathbf{x}(i),\mathbf{x}(i))} \\
\|\tilde{\boldsymbol{\omega}}(i)\|^2 &= \|\tilde{\boldsymbol{\omega}}(i-1)\|^2 + 2\frac{[e_p(i)-e_a(i)]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}e_a(i) + \frac{[e_p(i)-e_a(i)]^2}{[\kappa(\mathbf{x}(i),\mathbf{x}(i))]}
\end{aligned} \tag{4.22}$$

Com isto, substituindo (4.20) em (4.22) tem-se que

$$\begin{aligned}
\|\tilde{\boldsymbol{\omega}}(i)\|^2 &= \|\tilde{\boldsymbol{\omega}}(i-1)\|^2 - 2\frac{\eta g[e(i)]\kappa(\mathbf{x}(i),\mathbf{x}(i))}{\kappa(\mathbf{x}(i),\mathbf{x}(i))}e_a(i) + \frac{[\eta g[e(i)]\kappa(\mathbf{x}(i),\mathbf{x}(i))]^2}{[\kappa(\mathbf{x}(i),\mathbf{x}(i))]} \\
&= \|\tilde{\boldsymbol{\omega}}(i-1)\|^2 - 2\eta g[e(i)]e_a(i) + \eta^2 g^2[e(i)]\kappa(\mathbf{x}(i),\mathbf{x}(i))
\end{aligned} \tag{4.23}$$

Como explicitado na seção 3.4,a análise da convergência é feita em geral a partir de algum momento estatístico. Nesta esteira, aplicando o operador expectância em (4.23), tem-se que

$$\mathrm{E}\left[\|\tilde{\boldsymbol{\omega}}(i)\|^2\right] = \mathrm{E}\left[\|\tilde{\boldsymbol{\omega}}(i-1)\|^2\right] - 2\eta\,\mathrm{E}\left[g[e(i)]e_a(i)\right] + \eta^2\kappa(\mathbf{x}(i),\mathbf{x}(i))\,\mathrm{E}\left[g^2[e(i)]\right] \tag{4.24}$$

Em que o termo $\kappa(\mathbf{x}(i),\mathbf{x}(i))$ está fora do operador expectância pois seu valor no kernel Gaussiano, no qual a análise está sendo construída, é $\kappa(\mathbf{x}(i),\mathbf{x}(i)) = 1$, independente do valor de $\mathbf{x}(i)$.

O algoritmo convergirá se o vetor de erro dos pesos tender a zero, o que significa que o vetor de pesos está ficando mais próximo do vetor ótimo. O que pode ser interpretado com relação a média da energia, de modo que a convergência do algoritmo ocorre em média se a energia do vetor de erro dos pesos decresce monotonicamente ao longo de cada iteração.

Esta conclusão é expressa pela relação

$$\mathrm{E}\left[\|\tilde{\boldsymbol{\omega}}(i)\|^2\right] \leq \mathrm{E}\left[\|\tilde{\boldsymbol{\omega}}(i-1)\|^2\right] \tag{4.25}$$

a qual é equivalente a estabelecer que, dado (4.24), para que (4.25) ocorra, a soma dos termos a direita de $\mathrm{E}\left[\|\tilde{\boldsymbol{\omega}}(i-1)\|^2\right]$ é negativa ou nula, ou seja,

$$-2\eta\,\mathrm{E}\left[g[e(i)]e_a(i)\right] + \eta^2\kappa(\mathbf{x}(i),\mathbf{x}(i))\,\mathrm{E}\left[g^2[e(i)]\right] \leq 0 \tag{4.26}$$

É possível ainda a escrita deste resultado de (4.26) em função do valor de η

$$\begin{aligned} -2\eta\,\mathrm{E}\left[g[e(i)]e_a(i)\right] + \eta^2\kappa(\mathbf{x}(i),\mathbf{x}(i))\,\mathrm{E}\left[g^2[e(i)]\right] &\leq 0 \\ \Leftrightarrow \eta^2\kappa(\mathbf{x}(i),\mathbf{x}(i))\,\mathrm{E}\left[g^2[e(i)]\right] &\leq 2\eta\,\mathrm{E}\left[g[e(i)]e_a(i)\right] \\ \Leftrightarrow \eta &\leq \frac{2\,\mathrm{E}\left[g[e(i)]e_a(i)\right]}{\kappa(\mathbf{x}(i),\mathbf{x}(i))\,\mathrm{E}\left[g^2[e(i)]\right]} \end{aligned} \tag{4.27}$$

Assim, desde que η seja escolhido de acordo com a condição (4.27), o resultado do algoritmo será limitado, ou seja, (4.25) acontecerá, o que significa que o algoritmo convergirá. E deste modo, tem-se a relação entre a taxa de aprendizagem (μ) e o fator de inclinação α, expressos como η, para que a convergência do algoritmo aconteça.

4.3 Considerações finais

Este capítulo apresentou o algoritmo KSIG, cuja diferença com relação ao KLMS está na função de custo, mesma diferença do Sigmoide com relação ao LMS. Além disto foi explicada a análise da convergência a partir do método de análise proposto em [Yousef and Sayed 2001], de modo que a convergência é garantida se η, que é um parâmetro ajustável no algoritmo, atender à condição (4.27).

Determinado o algoritmo e comprovada sua convergência, faltam alguns experimentos para comprovar empiricamente o que se espera em teoria, uma convergência mais rápida, o que será feito em seção posterior.

Capítulo 5

KSIG com dicionário pré-definido

Apesar de diminuir o erro, algoritmos em *kernel* trazem algumas vezes um custo referente a memória, já que o dicionário de dados para obtenção da saída, se não for construído sobre algum critério de seleção/exclusão de algum(ns) elemento(s) desnecessário(s), pode tornar o gasto de memória uma desvantagem considerável.

Diversos autores, como [Gao et al. 2015, Chen et al. 2014, Parreira 2012] propuseram versões para o KLMS. Aqui será apresentada uma versão com dicionário pré-definido para o algoritmo KSIG. É sobre isto que trata esta seção.

5.1 Algoritmo KSIG com o dicionário pré-definido

Um dos resultados do capítulo anterior é a expressão de saída do algoritmo KSIG para uma entrada $\mathbf{x}$ qualquer, definida em (4.7) como

$$y = \sum_{i=1}^{N} c_i \kappa(\mathbf{x}_i, \mathbf{x}) \tag{5.1}$$

em que cada c_i é um coeficiente que dá um certo peso à função *kernel* $\kappa(\mathbf{x}_i, \bullet)$; e cada $\mathbf{x}_i$ é um elemento do dicionário de dados obtido durante a fase de treinamento [Liu et al. 2010], dicionário que no caso do algoritmo KSIG, é o conjunto de todos os dados de treino $\{\mathbf{x}_i\}_{i=1}^{N}$.

Como na seção 3.3.3, pode-se reescrever (5.1) na forma matricial

$$y = \mathbf{c}^T \boldsymbol{\kappa}_{\mathbf{x}} \tag{5.2}$$

em que $\mathbf{c} = [c_1, c_2, \ldots, c_N]^T$ e $\boldsymbol{\kappa}_{\mathbf{x}} = [\kappa(\mathbf{x}_1, \mathbf{x}), \kappa(\mathbf{x}_2, \mathbf{x}), \ldots, \kappa(\mathbf{x}_N, \mathbf{x})]^T$.

Supondo que o conjunto de dados do dicionário é pré-definido, ou seja, que há um $\boldsymbol{\omega} = [\mathbf{x}_{\boldsymbol{\omega}(1)}, \mathbf{x}_{\boldsymbol{\omega}(2)}, \ldots, \mathbf{x}_{\boldsymbol{\omega}(n)}]$ que é utilizado na determinação da saída mesmo na fase de treinamento, então, para melhorar a filtragem adaptativa durante a fase de treino, é preciso alterar o valor de cada coeficiente, ou seja, do peso de cada $\kappa(\mathbf{x}_i, \mathbf{x})$ na determinação da saída.

Com a forma matricial definida em (5.2), é possível propor uma equação adaptativa para a aprendizagem do melhor coeficiente como

$$\mathbf{c}(i+1) = \mathbf{c}(i) + \eta \tanh(\alpha e(i)) \boldsymbol{\kappa}_{\mathbf{x}_{\boldsymbol{\omega}}}(\mathbf{x}(i)) \tag{5.3}$$

em que $\eta \triangleq \alpha\mu$, com μ definido como a taxa de aprendizagem e α o coeficiente de inclinação; $\mathbf{x}(i)$ é o dado de entrada usado no treino e;

$$\boldsymbol{\kappa}_{\mathbf{x}_{\boldsymbol{\omega}}}(\mathbf{x}(i)) = [\kappa(\mathbf{x}_{\boldsymbol{\omega}(1)}, \mathbf{x}(i)), \kappa(\mathbf{x}_{\boldsymbol{\omega}(2)}, \mathbf{x}(i)), \ldots, \kappa(\mathbf{x}_{\boldsymbol{\omega}(n)}, \mathbf{x}(i))]^T.$$

Este resultado é o algoritmo KSIG com dicionário pré-definido, cujo pseudocódigo é ilustrado na Figura 5.1.

A partir do pseudocódigo da Figura 5.1, é possível calcular a complexidade do algoritmo. Considerando que as operações das linhas 2 a 8 tem custo constante; a operação mais custosa é a da linha 11, que se repete t vezes a cada iteração do laço mais externo, que se repete n vezes, com t como o tamanho do dicionário e n o do conjunto de treino. As operações de 13 a 16 se repetem n vezes. Considerando que todas as operações tem custo unitário, o custo total é de

$$1+1+1+1+1+1+1+1+(n \times t) \times 1 + n \times 1 + n \times 1 + n \times 1 + n \times 1 + n \times 1 = 7 + nt + 4n$$

Admitindo que o valor de $t > 4$, então, pode-se afirmar que o custo é $O(nt)$ para cômputo dos coeficientes. Já o cálculo da saída tem custo $O(t)$.

Conhecido o algoritmo KSIG com dicionário pré-determinado, no qual o vetor de coeficientes deve ser otimizado, um estudo sobre sua convergência é uma forma de verificar em teoria a funcionalidade do algoritmo.

1: **function** KSIG_DICIONARIO_PRE_DEF($X, d, dicionario, \alpha, \mu$)
2: $\eta \leftarrow \alpha \times \mu$
3: $n \leftarrow$ size(d) ▷ número de dados de treino
4: $e \leftarrow$ zero_vector(n) ▷ erro, um vetor de zeros
5: $y \leftarrow$ zero_vector(n) ▷ vetor de saída
6: $t \leftarrow$ size($dicionario$) ▷ tamanho do dicionário
7: $c \leftarrow$ ones_vector(t) ▷ coeficientes
8: $\kappa_x \leftarrow$ ones_vector(t) ▷ dicionário
9: **for** $j = 1, \ldots, n$ **do** ▷ treinamento
10: **for** $i = 1, \ldots, t$ **do**
11: $\kappa_x(i) \leftarrow \kappa(dicionario(:, i), X(:, j))$
12: **end for**
13: $y \leftarrow$ multiplica_vetores(c, κ_x)
14: $e(j) \leftarrow d(j) - y$
15: $fator\_atualizacao \leftarrow$ multiplica_constante_vetor$(\eta \times \tanh(\alpha, e(j)), \kappa_x)$
16: $c \leftarrow soma\_vetores(c, fator\_atualizazao)$
17: **end for**
18: *return* c
19: **end function**

Figura 5.1: Pseudocódigo do algoritmo KSIG com dicionário pré-definido.

5.2 Análise de convergência do KSIG com dicionário pré-definido

Na análise de convergência, desta seção, será adotada, por conveniência, uma análise de estado estacionário, como proposta em [Santana 2006], devido ao algoritmo KSIG com dicionário pré-definido ser similar ao algoritmo Sigmoide.

No intuito de simplificar a notação, definir-se-á o vetor aleatório $\boldsymbol{\kappa}_{\mathbf{x}(i)} \triangleq \boldsymbol{\kappa}_{\mathbf{x}_{\boldsymbol{\omega}}}(\mathbf{x}(i))$. Além desta consideração, algumas hipóteses são necessárias para validar a análise

- os vetores de entrada $\boldsymbol{\kappa}_{\mathbf{x}(i)}$ são aleatórios, pois dependem da variável aleatória $\mathbf{x}(i)$;

- os vetores de entrada $\boldsymbol{\kappa}_{\mathbf{x}(i)}$ são estatisticamente independentes uns dos outros, ou seja, para todo $\boldsymbol{\kappa}_{\mathbf{x}(i)}$, $\boldsymbol{\kappa}_{\mathbf{x}(j)}$, $i \neq j$, são independentes;
- cada elemento do vetor $\boldsymbol{\kappa}_{\mathbf{x}(i)}$ está no intervalo $[-\delta, \delta]$, em que $\delta \leq 1$ (o que é garantido caso κ seja o *kernel* gaussiano);
- o ruído de distúrbio $v(i)$ é estatisticamente independente de $\boldsymbol{\kappa}_{\mathbf{x}(i)}$
- todas as variáveis não têm necessariamente distribuição gaussiana;
- $\mathbf{c}(j)$ é estatisticamente independente de $\boldsymbol{\kappa}_{\mathbf{x}(i)}$.

A última hipótese é em particular possível pois, na análise em estado estacionário, a entrada não mais influencia no vetor de constantes [Sayed 2008].

Definindo o vetor de erro dos coeficientes como $\tilde{\mathbf{c}}(i) \triangleq \mathbf{c}(i) - \mathbf{c}_o$, em que $\mathbf{c}_o$ é o vetor de coeficientes ótimo, de modo que, dado o sinal desejado $d(i)$, $d(i) = \mathbf{c}_o^T \boldsymbol{\kappa}_{\mathbf{x}(i)} + v(i)$; é possível reescrever o erro $e(i)$ como

$$\begin{aligned} e(i) &= d(i) - y(i) \\ &= \mathbf{c}_o^T \boldsymbol{\kappa}_{\mathbf{x}(i)} + v(i) - \mathbf{c}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)} \\ &= v(i) - (\mathbf{c}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)} - \mathbf{c}_o^T \boldsymbol{\kappa}_{\mathbf{x}(i)}) \\ &= v(i) - (\mathbf{c}(i) - \mathbf{c}_o)^T \boldsymbol{\kappa}_{\mathbf{x}(i)} \\ e(i) &= v(i) - \tilde{\mathbf{c}}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)} \end{aligned} \tag{5.4}$$

Ou seja, é possível expressar o erro em função do ruído de distúrbio do modelo e do vetor de erro dos coeficientes. A partir desta definição do erro e considerando para fim de simplificação de notação $g(x) \triangleq \tanh(\alpha x)$, pode-se reescrever a equação de atualização do vetor de coeficientes em (5.3) como

$$\mathbf{c}(i+1) = \mathbf{c}(i) + \eta g\left(v(i) - \tilde{\mathbf{c}}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)} \tag{5.5}$$

Então, subtraindo $\mathbf{c}_o$ de ambos os lados em (5.5), obtém-se

$$\begin{aligned} \mathbf{c}(i+1) - \mathbf{c}_o &= \left[\mathbf{c}(i) + \eta g\left(v(i) - \tilde{\mathbf{c}}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] - \mathbf{c}_o \\ &= \mathbf{c}(i) - \mathbf{c}_o + \eta g\left(v(i) - \tilde{\mathbf{c}}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)} \\ \tilde{\mathbf{c}}(i+1) &= \tilde{\mathbf{c}}(i) + \eta g\left(v(i) - \tilde{\mathbf{c}}^T(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)} \end{aligned} \tag{5.6}$$

e assim, obtém-se a equação de atualização do vetor de erro dos coeficientes a cada iteração, a partir da qual é possível provar a convergência como em [Santana 2006].

Expandindo a função $g\left(v(i) - \tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right)$ em série de Taylor sobre $-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}$, tem-se como resultado

$$\begin{aligned}\tanh\left(v(i) - \tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) &= \sum_{k=0}^{\infty} \frac{g^{(k)}(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)})}{k!} v^k(i) \\ &\cong g\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) + g'\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v(i) \\ &+ \frac{1}{2} g^{(2)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v^2(i) \\ &+ \frac{1}{6} g^{(3)}(\delta) v^3(i), \end{aligned} \tag{5.7}$$

em que $g^{(k)}$ é a k-ésima derivada de g e δ é um valor entre $[0, v(i)]$. É importante denotar, que são omitidos os demais valores da série, $\left\{\frac{g^{(k)}(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)})}{k!} v^k(i)\right\}_{k=4}^{\infty}$ porque estes são relativamente pequenos, devido ao denominador $k!$.

Substituindo o resultado de (5.7) em (5.6) tem-se que

$$\begin{aligned}\tilde{\mathbf{c}}(i+1) \cong \tilde{\mathbf{c}}(i) + \eta \Big[& g\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) + g^{(1)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v(i) \\ & + \frac{1}{2} g^{(2)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v^2(i) + \frac{1}{6} g^{(3)}(\delta) v^3(i) \Big] \boldsymbol{\kappa}_{\mathbf{x}(i)} \end{aligned} \tag{5.8}$$

Dado que $\tilde{\mathbf{c}}(i)$ é variável aleatória, pois depende de $\boldsymbol{\kappa}_{\mathbf{x}(i)}$, também aleatório, aplicando a expectância de ambos os lados de (5.8), tem-se

$$\begin{aligned}\mathrm{E}\left[\tilde{\mathbf{c}}(i+1)\right] \cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] + \eta \Big(& \mathrm{E}\left[g\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \\ & + \mathrm{E}\left[g^{(1)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \\ & + \mathrm{E}\left[\frac{1}{2} g^{(2)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) v^2(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \\ & + \mathrm{E}\left[\frac{1}{6} g^{(3)}(\delta) v^3(i) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \Big) \end{aligned} \tag{5.9}$$

Como por hipótese, o ruído é independente das outras variáveis e, sabendo que os momentos ímpares do ruído, que tem média zero, são zero [Papoulis 1991], o resultado em (5.9) pode ser reescrito como

$$\begin{aligned}\mathrm{E}\left[\tilde{\mathbf{c}}(i+1)\right] \cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] + \eta \Big(& \mathrm{E}\left[g\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \\ & + \mathrm{E}\left[\frac{1}{2} g^{(2)}\left(-\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right) \boldsymbol{\kappa}_{\mathbf{x}(i)}\right] \sigma_v^2 \Big) \end{aligned} \tag{5.10}$$

em que $\sigma_v^2 \triangleq \mathrm{E}[v(i)]$ é a variância do ruído.

Em [Santana 2006] é proposto um teorema para resolver o problema da prova da convergência do algoritmo Sigmoide, neste teorema[1] é estabelecido que

Teorema S : seja f uma função ímpar e não linear, definida e contínua no intervalo $(-\delta, \delta)$, em que δ é um número positivo suficientemente pequeno, então $f(\alpha\beta) \approx \alpha f(\beta)$, para todo $\alpha, \beta \in (-\delta, \delta)$.

A partir deste teorema, a expectância em (5.10) pode ser reescrita como

$$
\begin{aligned}
\mathrm{E}\left[\tilde{\mathbf{c}}(i+1)\right] &\cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] - \eta\left(\mathrm{E}\left[g\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\right)\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right] + \mathrm{E}\left[\frac{1}{2}g^{(2)}\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\right)\tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)}\right]\sigma_v^2\right) \\
&\cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] - \eta\left(\mathrm{E}\left[g\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\right)\boldsymbol{\kappa}_{\mathbf{x}(i)}^T\tilde{\mathbf{c}}(i)\right]\right. \\
&\left. + \mathrm{E}\left[\frac{1}{2}g^{(2)}\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\right)\boldsymbol{\kappa}_{\mathbf{x}(i)}^T\tilde{\mathbf{c}}(i)\right]\sigma_v^2\right) \qquad \text{pois, } \tilde{\mathbf{c}}^T(i)\boldsymbol{\kappa}_{\mathbf{x}(i)} = \boldsymbol{\kappa}_{\mathbf{x}(i)}^T\tilde{\mathbf{c}}(i) \\
&\cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] - \eta\left(\mathrm{E}\left[g\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\boldsymbol{\kappa}_{\mathbf{x}(i)}^T\right)\tilde{\mathbf{c}}(i)\right]\right. \\
&\left. + \mathrm{E}\left[\frac{1}{2}g^{(2)}\left(\boldsymbol{\kappa}_{\mathbf{x}(i)}\boldsymbol{\kappa}_{\mathbf{x}(i)}^T\right)\tilde{\mathbf{c}}(i)\right]\sigma_v^2\right) \\
&\cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] - \eta\left(\mathrm{E}\left[g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\tilde{\mathbf{c}}(i)\right]\right. \\
&\left. + \mathrm{E}\left[\frac{1}{2}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\tilde{\mathbf{c}}(i)\right]\sigma_v^2\right) \\
&\cong \mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] - \eta\left(g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathrm{E}\left[\tilde{\mathbf{c}}(i)\right]\right. \\
&\left. + \frac{1}{2}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathrm{E}\left[\tilde{\mathbf{c}}(i)\right]\sigma_v^2\right) \\
&\cong \left[\mathbf{I} - \eta\left(g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right) + \frac{1}{2}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\sigma_v^2\right)\right]\mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] \qquad (5.11)
\end{aligned}
$$

em que $\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}} = \boldsymbol{\kappa}_{\mathbf{x}(i)}\boldsymbol{\kappa}_{\mathbf{x}(i)}^T$, é a matriz de autocorrelação.

É possível definir $\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}$ como decomposição de suas matrizes de autovalores λ_k e de autovetores, $\mathbf{\Lambda}$ e $\mathbf{Q}$ respectivamente, em que $\mathbf{\Lambda} \triangleq \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$, em que

$$\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \qquad (5.12)$$

Ademais, pode-se definir também uma transformação do vetor de erros dos coeficientes baseado na matriz de autovetores, $\overline{\mathbf{c}} = \mathbf{Q}^{-1}\tilde{\mathbf{c}} \quad (\Rightarrow \tilde{\mathbf{c}} = \mathbf{Q}\overline{\mathbf{c}})$, e com isto, reescrever (5.11)

[1] a prova do teorema se encontra em [Santana 2006]

como

$$\mathbf{Q}\,\mathrm{E}\left[\overline{\mathbf{c}}(i+1)\right] \cong \left[\mathbf{I} - \eta\left(g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right) + \frac{1}{2}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\sigma_v^2\right)\right]\mathbf{Q}\,\mathrm{E}\left[\overline{\mathbf{c}}(i)\right] \tag{5.13}$$

E então, multiplicando ambos os lados de (5.13) por $\mathbf{Q}^{-1}$ pela esquerda

$$\underbrace{\mathbf{Q}^{-1}\mathbf{Q}}_{\mathbf{I}}\mathrm{E}\left[\overline{\mathbf{c}}(i+1)\right] \cong \mathbf{Q}^{-1}\left[\mathbf{I} - \eta\left(g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right) + \frac{1}{2}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\sigma_v^2\right)\right]\mathbf{Q}\,\mathrm{E}\left[\overline{\mathbf{c}}(i)\right]$$

$$\begin{aligned}
\mathrm{E}\left[\overline{\mathbf{c}}(i+1)\right] &\cong \left[\mathbf{Q}^{-1} - \eta\mathbf{Q}^{-1}g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right) + \frac{\eta}{2}\mathbf{Q}^{-1}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\sigma_v^2\right]\mathbf{Q}\,\mathrm{E}\left[\overline{\mathbf{c}}(i)\right] \\
&\cong \left[\mathbf{Q}^{-1}\mathbf{Q} - \eta\mathbf{Q}^{-1}g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathbf{Q} + \frac{\eta}{2}\mathbf{Q}^{-1}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathbf{Q}\sigma_v^2\right]\mathrm{E}\left[\overline{\mathbf{c}}(i)\right] \\
&\cong \left[\mathbf{I} - \eta\mathbf{Q}^{-1}g\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathbf{Q} + \frac{\eta}{2}\mathbf{Q}^{-1}g^{(2)}\left(\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\right)\mathbf{Q}\sigma_v^2\right]\mathrm{E}\left[\tilde{\mathbf{c}}(i)\right] \\
&\cong \left[\mathbf{I} - \eta g\left(\mathbf{Q}^{-1}\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q}\right) + \frac{\eta}{2}g^{(2)}\left(\mathbf{Q}^{-1}\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q}\right)\sigma_v^2\right]\mathrm{E}\left[\overline{\mathbf{c}}(i)\right] \qquad (5.14) \\
\mathrm{E}\left[\overline{\mathbf{c}}(i+1)\right] &\cong \left[\mathbf{I} - \eta g\left(\boldsymbol{\Lambda}\right) + \frac{\eta}{2}g^{(2)}\left(\boldsymbol{\Lambda}\right)\sigma_v^2\right]\mathrm{E}\left[\overline{\mathbf{c}}(i)\right] \qquad (5.15)
\end{aligned}$$

em que (5.14) decorre do fato de considerar $\mathbf{Q}^{-1}\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q}$ um valor pequeno, podendo aplicar então o Teorema S, consistindo então, em uma aproximação; já (5.15) é um resultado da multiplicação de $\mathbf{Q}$ pela direita e de $\mathbf{Q}^{-1}$ pela esquerda em (5.12), ou seja,

$$\begin{aligned}
\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q} &= \mathbf{Q}\boldsymbol{\Lambda}\overbrace{\mathbf{Q}^{-1}\mathbf{Q}}^{\mathbf{I}} && \text{multiplicação de } \mathbf{Q} \text{ pela direita} \\
\mathbf{Q}^{-1}\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q} &= \underbrace{\mathbf{Q}^{-1}\mathbf{Q}}_{\mathbf{I}}\boldsymbol{\Lambda} && \text{multiplicação de } \mathbf{Q}^{-1} \text{pela esquerda} \\
\mathbf{Q}^{-1}\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}\mathbf{Q} &= \boldsymbol{\Lambda}
\end{aligned}$$

A equação em (5.15) é o valor esperado de

$$\overline{\mathbf{c}}(i+1) \cong \left[\mathbf{I} - \eta g\left(\boldsymbol{\Lambda}\right) - \frac{\eta}{2}g^{(2)}\left(\boldsymbol{\Lambda}\right)\sigma_v^2\right]\overline{\mathbf{c}}(i) \tag{5.16}$$

Resultado que pode ser resolvido por indução

$$\begin{aligned}
\overline{\mathbf{c}}(i+1) &\cong \left[\mathbf{I} - \eta g\left(\boldsymbol{\Lambda}\right) - \frac{\eta}{2}g^{(2)}\left(\boldsymbol{\Lambda}\right)\sigma_v^2\right]\overline{\mathbf{c}}(i) \\
&\cong \left[\mathbf{I} - \eta g\left(\boldsymbol{\Lambda}\right) - \frac{\eta}{2}g^{(2)}\left(\boldsymbol{\Lambda}\right)\sigma_v^2\right]^2\overline{\mathbf{c}}(i-1) \\
&\vdots \\
\overline{\mathbf{c}}(i+1) &\cong \left[\mathbf{I} - \eta g\left(\boldsymbol{\Lambda}\right) - \frac{\eta}{2}g^{(2)}\left(\boldsymbol{\Lambda}\right)\sigma_v^2\right]^{(i+1)}\overline{\mathbf{c}}(0) \qquad (5.17)
\end{aligned}$$

em que $\overline{\mathbf{c}}(0)$ é o vetor de erro dos coeficientes transformado, e está associado ao valores iniciais que são arbitrariamente escolhidos através da relação $\overline{\mathbf{c}}(0) = \mathbf{Q}^{-1}\mathbf{c}(0)$.

No que tange a convergência do algoritmo, esta ocorrerá se o vetor de erro dos coeficientes tende ao vetor nulo ao longo das iterações, o que significa que o lado direito de (5.17) quando $i \to \infty$ é, no limite,

$$\lim_{i\to\infty}\left[\mathbf{I} - \eta g\left(\mathbf{\Lambda}\right) - \frac{\eta}{2}g^{(2)}\left(\mathbf{\Lambda}\right)\sigma_v^2\right]^{(i+1)} = \mathbf{0} \tag{5.18}$$

Como $\mathbf{\Lambda}$ é uma matriz diagonal, os termos fora da diagonal são zero, então apenas os elementos da diagonal de $\mathbf{\Lambda}$, autovalores, têm influência no limite definido em (5.18). Deste modo, tomando o limite para cada autovalor, tem-se a relação

$$\lim_{i\to\infty}\left[1 - \eta g\left(\lambda_m\right) - \frac{\eta}{2}g^{(2)}\left(\lambda_m\right)\sigma_v^2\right]^{(i+1)} = 0 \tag{5.19}$$

em que $m = 1, 2, \ldots, M$, com M como o tamanho de $\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}$.

A convergência é garantida (limite tende a zero), para cada autovalor no limite se o termo entre colchetes em módulo for menor que um, isto é,

$$\begin{aligned}
\left|1 - \eta g\left(\lambda_m\right) - \frac{\eta}{2}g^{(2)}\left(\lambda_m\right)\sigma_v^2\right| \leq 1 \iff \\
-1 < 1 - \eta g\left(\lambda_{\max}\right) - \frac{\eta}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2 < 1 \xRightarrow{\times(-1)} \\
1 > -1 + \eta g\left(\lambda_{\max}\right) + \frac{\eta}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2 > -1 \xRightarrow{+1} \\
2 > \eta g\left(\lambda_{\max}\right) + \frac{\eta}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2 > 0 \Leftrightarrow \\
0 < \eta g\left(\lambda_{\max}\right) + \frac{\eta}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2 < 2 \Leftrightarrow \\
0 < \eta\left(g\left(\lambda_{\max}\right) + \frac{1}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2\right) < 2 \Leftrightarrow \\
0 < \eta < \frac{2}{g\left(\lambda_{\max}\right) + \frac{1}{2}g^{(2)}\left(\lambda_{\max}\right)\sigma_v^2}
\end{aligned} \tag{5.20}$$

em que $\lambda_{\max}$ é o maior autovalor de $\mathbf{R}_{\boldsymbol{\kappa}_{\mathbf{x}(i)}}$, o que garante a convergência para os outros autovalores de menor magnitude. Substituindo $g(x)$ por $\tanh(\alpha x)$ tem-se

$$0 < \eta < \frac{2}{\tanh\left(\alpha\lambda_{\max}\right) + \frac{1}{2}\tanh^{(2)}\left(\alpha\lambda_{\max}\right)\sigma_v^2} \tag{5.21}$$

Assim, desde que η esteja no intervalo delimitado em (5.21), é garantido que

$$\lim_{i\longrightarrow\infty}\overline{\mathbf{c}}(i+1) = \mathbf{0} \tag{5.22}$$

Conhecendo a transformação de $\tilde{\mathbf{c}}$ para o espaço gerado por $\mathbf{Q}$ e sua definição baseada no vetor de constantes, então

$$\overline{\mathbf{c}}(i+1) = \mathbf{Q}^{-1}\tilde{\mathbf{c}} = \mathbf{Q}^{-1}(\mathbf{c} - \mathbf{c}_o)$$

Substituindo este resultado no limite em (5.22) e sabendo que $\mathbf{Q} \neq \mathbf{0}$, então

$$\begin{aligned}
&\lim_{i \longrightarrow \infty} \mathbf{Q}^{-1}(\mathbf{c}(i+1) - \mathbf{c}_o) = \mathbf{0} \\
&\mathbf{Q}^{-1} \lim_{i \longrightarrow \infty} (\mathbf{c}(i+1) - \mathbf{c}_o) = \mathbf{0} \qquad \text{pois } \mathbf{Q}^{-1} \text{ não muda com } i \\
&\lim_{i \longrightarrow \infty} (\mathbf{c}(i+1) - \mathbf{c}_o) = \mathbf{0} \\
&\lim_{i \longrightarrow \infty} \mathbf{c}(i+1) - \mathbf{c}_o = \mathbf{0} \\
&\lim_{i \longrightarrow \infty} \mathbf{c}(i+1) = \mathbf{c}_o
\end{aligned} \tag{5.23}$$

Este resultado indica que, no estado estacionário, o vetor de constantes em média tende ao vetor ótimo, desde que a condição (5.21) seja respeitada. Isto significa que o algoritmo converge para a solução ótima, o que comprova teoricamente sua eficácia.

5.3 Considerações finais

Este capítulo apresentou a versão do KSIG com dicionário de dados pré-determinado, cujo objetivo principal é amenizar o gasto de memória recorrente do uso indiscriminado de todo o conjunto de dados, e mais especificamente a filtragem adaptativa online.

Além do algoritmo, foi apresentada a análise da convergência do mesmo, baseada no trabalho de Santana [Santana 2006], a partir da qual pode-se determinar que para haver convergência, a taxa de aprendizagem deve estar no intervalo delimitado em (5.21).

Capítulo 6

Experimentos

No capítulo anterior foram apresentados os algoritmos de filtragem adaptativa não linear KSIG e KSIG com dicionário pré-definido, bem como as suas respectivas análises de convergência, o que auxilia a compreender teoricamente se os algoritmos convergiam em média para uma solução.

Este capítulo apresenta uma série de simulações, com alguns problemas comuns na literatura de filtragem adaptativa, para verificar experimentalmente a eficácia de cada algoritmo proposto nesta dissertação[1]. Para tal fim, os resultados dos algoritmos serão comparados com os obtidos no algoritmo KLMS, para verificar se o KSIG converge mais rápido que este, como o Sigmoide convergiu em relação ao LMS [Santana et al. 2006a].

Por conveniência, adotou-se a mesma metodologia desenvolvida em [Santana 2006], supondo que o desajuste dos algoritmos é o mesmo, isto significa que, no fim, o erro tenderá ao mesmo valor em ambos os algoritmos, criando certa equidade na avaliação. A próxima seção detalha como foi feita a comparação.

6.1 Critério de Comparação

Para comparar a convergência dos algoritmos KSIG e KLMS, considerar-se-á que o desajuste de ambos é o mesmo como feito em [Santana 2006]. O desajuste $\mathcal{M}$, em funções de custo não lineares, é muito complexo para ser calculado. Não obstante, pode ser usada a

[1] Códigos disponíveis em https://github.com/edenpsilva/MasterExperiments

aproximação proposta por [Douglas and Meng 1994]

$$\mathcal{M} = \frac{\mu \,\mathrm{E}\left[f^2(v(n))\right] \mathrm{Tr}[\mathbf{R}]}{2\,\mathrm{E}\left[f'(v(n))\right] \sigma_v^2} \tag{6.1}$$

em que μ é a taxa de aprendizagem do algoritmo; f é a derivada da função de custo, f^2 é o quadrado de f, e f' a derivada de f; $\mathbf{R}$ é a matriz de autocorrelação dos dados; v o ruído do modelo e; σ_v^2 a variância do ruído do modelo.

Sabendo que $f(e(i)) = \alpha \tanh(\alpha e(i))$, adaptando (6.1) o desajuste para o KSIG ($\mathcal{M}_{KSIG}$) é equivalente a

$$\begin{aligned}
\mathcal{M}_{KSIG} &= \frac{\mu_{KSIG}\,\mathrm{E}\left[\alpha^2 \tanh^2(\alpha v(n))\right] \mathrm{Tr}[\mathbf{R}_\kappa]}{2\,\mathrm{E}\left[\alpha\,\mathrm{sech}^2(\alpha v(n))\right] \sigma_v^2} \\
&= \frac{\mu_{KSIG}\alpha^2\,\mathrm{E}\left[\tanh^2(\alpha v(n))\right] \mathrm{Tr}[\mathbf{R}_\kappa]}{2\alpha\,\mathrm{E}\left[\mathrm{sech}^2(\alpha v(n))\right] \sigma_v^2} \\
\mathcal{M}_{KSIG} &= \frac{\mu_{KSIG}\alpha\,\mathrm{E}\left[\tanh^2(\alpha v(n))\right] \mathrm{Tr}[\mathbf{R}_\kappa]}{2\,\mathrm{E}\left[\mathrm{sech}^2(\alpha v(n))\right] \sigma_v^2}
\end{aligned} \tag{6.2}$$

em que $\mathbf{R}_\kappa$ é a matriz de autocorrelação do dicionário de dados. O desajuste do KLMS é definido como [Liu et al. 2010]

$$\mathcal{M}_{KLMS} = \frac{\mu_{KLMS}}{2N} \mathrm{Tr}[\mathbf{G}_\varphi] \tag{6.3}$$

em que $\mathbf{G}_\varphi$ é a matriz de Gram do dicionário de dados, sendo equivalente a $\mathbf{R}_\kappa$, o que possibilita que o desajuste do KLMS seja reescrito como

$$\mathcal{M}_{KLMS} = \frac{\mu_{KLMS}}{2N} \mathrm{Tr}[\mathbf{R}_\kappa] \tag{6.4}$$

Como os desajustes são equivalentes, igualando (6.2) a (6.4)

$$\begin{aligned}
\frac{\mu_{KSIG}\alpha\,\mathrm{E}\left[\tanh^2(\alpha v(n))\right] \mathrm{Tr}[\mathbf{R}_\kappa]}{2\,\mathrm{E}\left[\mathrm{sech}^2(\alpha v(n))\right] \sigma_v^2} &= \frac{\mu_{KLMS}}{2N} \mathrm{Tr}[\mathbf{R}_\kappa] \\
\mu_{KSIG} &= \mu_{KLMS} \frac{\mathrm{E}\left[\mathrm{sech}^2(\alpha v(n))\right] \sigma_v^2}{\alpha N\,\mathrm{E}\left[\tanh^2(\alpha v(n))\right]}
\end{aligned} \tag{6.5}$$

Este resultado torna clara a relação entre as taxas de aprendizagens dos algoritmos considerando que ambos tem mesmo desajuste. É com base nele que são estabelecidos os experimentos, em que as taxas de aprendizagem mantêm esta relação.

Conhecida a definição das taxas de aprendizagem para mesmo desajuste, na próxima seção serão apresentados os experimentos relativos ao KSIG, para verificar experimentalmente a convergência mais rápidas deste com relação ao KLMS.

6.2 KSIG

Esta seção traz duas simulações do algoritmo KSIG, para a solução de dois problemas de filtragem adaptativa, o primeiro se refere a predição em série temporal e o segundo a equalização de um canal não linear. Estas simulações foram compiladas em [Silva et al. 2015], mas aqui trazem pequenas modificações, se adequando aos novos resultados teóricos desenvolvidos na seção anterior.

6.2.1 Experimento 1

A predição consiste em descobrir o valor de um sinal no tempo atual, dadas algumas amostras passadas deste mesmo sinal. Isto significa que, dadas várias amostras do sinal fonte em momentos anteriores a i, $\{\mathbf{s}(i-1), \mathbf{s}(i-2), \mathbf{s}(i-3), \ldots\}$, deve-se descobrir o valor de $\mathbf{s}(i)$.

Para o primeiro experimento, foi utilizada, por conveniência, a mesma simulação de predição encontrada em [Liu et al. 2010], a qual trata da série de Mackey-Glass, que é obtida a partir da equação

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1 + x(t-\tau)^{10}}. \tag{6.6}$$

Por conveniência, os parâmetros a, b e τ foram os mesmos utilizados para por [Liu et al. 2010], cujos valores são, $b = 0.1$, $a = 0.2$, e $\tau = 30$. O conjunto de dados é formado com 5000 pontos tomados a uma frequência de amostragem de seis segundos, armazenados em arquivo ".MAT". Algumas amostras da série estão ilustradas na figura 6.1.

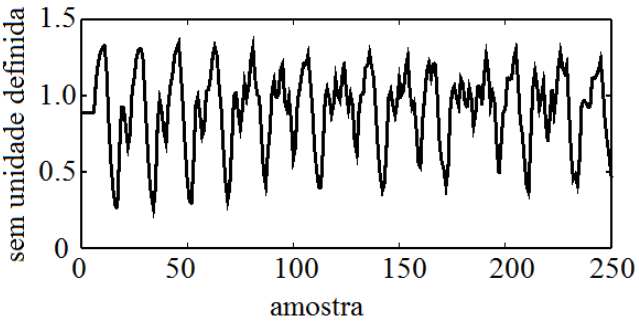

Figura 6.1: Amostras da série de Mackey-Glass

A predição ocorre a partir de um vetor de entrada $\mathbf{x}(i) = [x(i-10), x(i-9), ..., x(i-1)]^T$, em que cada $x(i-t)$ consiste nos dez valores anteriores ao tempo i, a partir dos quais faz-se

a previsão do valor da série no tempo i. A saída do filtro é então o valor da série previsto para o tempo i, dados dez valores da série anteriores ao tempo i.

O *kernel* utilizado nesta simulação foi o Gaussiano [Liu et al. 2008] com tamanho do *kernel* $h = 1/\sqrt{2}$; o tamanho do passo (ou taxa de aprendizagem) do KLMS foi de $\mu_{KLMS} = 0.2$, que são os valores encontrados em [Liu et al. 2010]; a constante de inclinação $\alpha = 0.027$, valor escolhido experimentalmente; já o tamanho do passo no KSIG obtido substituindo valores escolhidos de μ_{KLMS}, α, e o tamanho do dicionário, $N = 500$, em (6.5), foi de $\mu_{KSIG} = 1.632 \times 10^5$.

Em cada iteração na fase de treinamento foram utilizados 100 dados de teste e coletado o erro médio quadrático (MSE) para cada dado e feita a média aritmética a partir dos 100 MSEs coletados. A Figura 6.2 ilustra o resultado da simulação, nesta figura, o MSE médio ao longo das iterações vai diminuindo até um certo ponto em que estabiliza (estado estacionário) para ambos os algoritmos.

Na Figura 6.2, a curva tracejada, que é referente ao KSIG, está claramente abaixo da curva contínua que representa o MSE médio do KLMS, pelo menos até pouco mais da trecentésima iteração, na qual, o MSE médio do KSIG está com valor próximo ao do estado estacionário, cujo valor é praticamente similar para ambos, como se observa da iteração quatrocentos em diante. Isto significa que o KSIG chega no estado estacionário em menos tempo que o KLMS nesta atividade de filtragem adaptativa, neste caso, em aproximadamente 50 iterações.

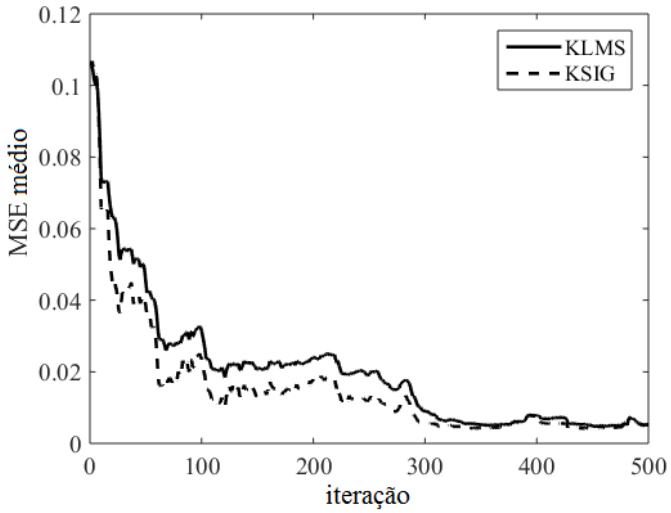

Figura 6.2: Curva de aprendizagem para predição da série temporal pelo KLMS e KSIG

Este resultado ilustra que o KSIG foi um pouco mais rápido quanto ao tempo de convergência em relação ao KLMS. Na subseção seguinte será apresentada outra tarefa de filtragem adaptativa.

6.2.2 Experimento 2

A equalização consiste em, dado um sinal de entrada que passa por um canal não linear, deve-se fazer a equalização (ou deconvolução), obtendo como saída da operação de equalização o sinal de entrada, ou seja, a recuperação do sinal que passa pelo canal não linear [Haykin 2014].

Por conveniência o experimento aqui descrito foi retirado de [Liu et al. 2010]. Neste, o problema consiste em, dado um sinal fonte binário que passa por um canal não linear deve-se obter o sinal fonte partir da saída do canal. Para isto, o sinal na saída do canal consiste na entrada de um filtro adaptativo de equalização, cuja saída desejada é o sinal fonte. A ilustração deste problema está na Figura 6.3.

A partir da Figura 6.3, observa-se que o sinal fonte $s(i)$ passa por um canal não linear indicado pelo retângulo tracejado cuja saída é $r(i)$. A primeira parte do canal é um filtro linear com função de transferência $H(z)$, com saída $x(i)$, a qual é entrada de um filtro não linear, em cuja saída é adicionada um ruído $n(i)$, resultando em $r(i)$, que é a entrada do filtro adaptativo, cuja saída desejada é o sinal fonte.

Neste experimento, as equações de $x(i)$ e $r(i)$ são

$$x(i) = s(i) + 0.5s(i-1)$$
$$r(i) = x(i) + 0.9x(i-1)^2 + n(i)$$

em que $n(i)$ é um ruído de distribuição gaussiana e σ^2 de variância.

Na simulação, foram utilizados os mesmos valores de parâmetros que em [Liu et al. 2010]. Nesta a entrada do filtro é o vetor $[r(i), r(i+1), ..., r(i+l)]$ e o sinal desejado $s(i-D)$, com $l = 5$ e $D = 2$ como em [Liu et al. 2010]. Foi adicionado um ruído Gaussiano com variância $\sigma_n^2 = 0.4$. Já o *kernel* foi o Gaussiano com tamanho $\sigma^2 = 0.1$ com tamanho do passo do KLMS de $\mu_{KLMS} = 0.01$.

O coeficiente $\alpha = 0.0015$ teve seu valor escolhido experimentalmente e a taxa de aprendizagem do KSIG, dada por (6.5), com $N = 1000$ é igual a $\mu_{KSIG} = 1.488 \times 10^5$.

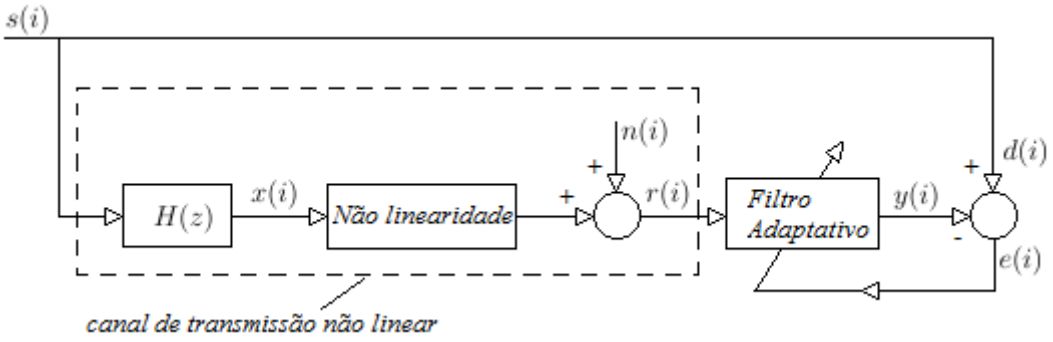

Figura 6.3: Problema da equalização de canal não linear. Adaptado de [Liu et al. 2010].

A Figura 6.4 ilustra o resultado da simulação. Nesta figura, a curva de aprendizagem do KSIG encontra-se ligeiramente abaixo da curva do KLMS, pelo menos até pouco depois da iteração 600, onde ambas as curvas tornam-se praticamente sobrepostas, ou seja, em estado estacionário. Assim, os resultados ilustram que o KSIG convergiu ligeiramente mais rápido que o KLMS, em algumas poucas iterações a menos.

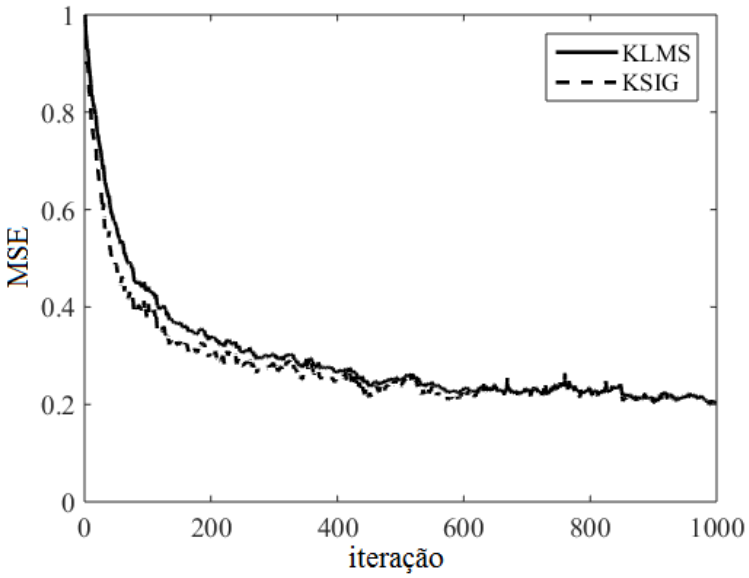

Figura 6.4: Curvas de aprendizagem do KSIG e do KLMS para equalização.

Assim como na simulação com predição, a equalização apontou que o algoritmo KSIG converge mais rápido para a solução que o KLMS, mas com um número não tão grande de iterações. Na próxima seção serão apresentados experimentos relativos ao KSIG com dicionário pré-definido, verificando sua maior velocidade de convergência quando comparado ao KLMS com dicionário pré-definido.

6.3 KSIG com dicionário pré-definido

Esta seção traz duas simulações com duas tarefas de filtragem adaptativa para o KSIG na versão com dicionário pré-definido, e para o KLMS também com dicionário pré-definido para fim de comparação de desempenho.

6.3.1 Experimento 1

A identificação de sistema é um dos quatro problemas comuns de filtragem adaptativa [Haykin 2014]. Dado um sistema qualquer cujas características de composição são desconhecidas, o filtro deve, com a mesma entrada do sistema a identificar, produzir a mesma saída que este, ou seja, o filtro é uma cópia do sistema, uma identificação do mesmo, mas sem necessariamente saber como o sistema original é construído. O diagrama de blocos na Figura (6.5) ilustra esta tarefa.

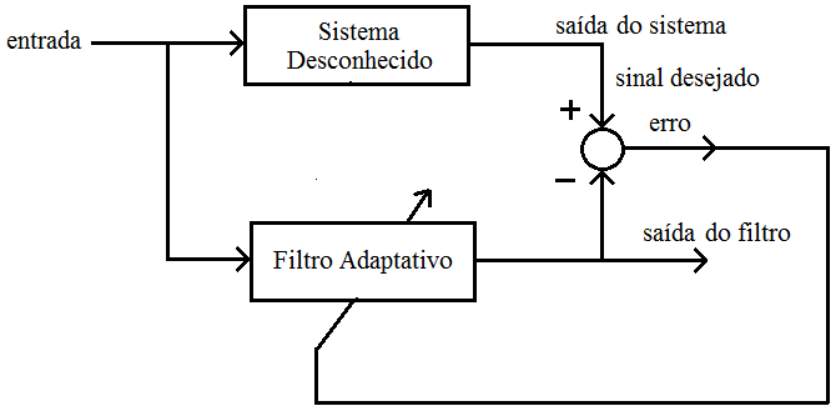

Figura 6.5: Diagrama de blocos da identificação de um sistema

Para a simulação, foi utilizada a identificação do sistema proposto em [Gao et al. 2015]. Dado um sinal de entrada em que cada termo é definido pela equação

$$x(i) = \rho x(i-1) + \sigma_x \sqrt{1-\rho^2} w(i) \tag{6.7}$$

em que ρ é um parâmetro de controle, definido por [Chen et al. 2014] como circularidade do dado de entrada, σ_x é o desvio padrão da entrada; $w(i)$ é um ruído com distribuição normal

padrão. Este sinal passa pelo sistema não linear

$$u(i) = 0.5x(i) - 0.3x(i-1) \tag{6.8}$$

$$y(i) = u(i) - 0.5u^2(i) + 0.1u^3(i) + n(i) \tag{6.9}$$

com $n(i)$ definido como um ruído gaussiano, com média zero e desvio padrão $\sigma_n = 0.05$. A entrada do sistema é o vetor $\mathbf{x}(i) = [x(i), x(i-1)]^T$ e sua saída $y(i)$. Neste experimento, a entrada do filtro é a mesma do sistema, $\mathbf{x}(i)$, e, o sinal desejado, a saída do sistema $y(i)$.

Para os parâmetros arbitrários foram utilizados os mesmos valores que em [Chen et al. 2014], $\sigma_x = 0.5$, $\rho = 0.5$, *kernel* Gaussiano com tamanho definido como $\sigma = 0.25$, o tamanho do passo para KLMS foi ajustado como $\mu_{KLMS} = 0.05$; já o coeficiente de inclinação $\alpha = 3 \times 10^{-3}$.

O dicionário, que é o mesmo para ambos os algoritmos, contém vinte e cinco elementos selecionados aleatoriamente da região $[-1, 1] \times [-1, 1]$. Determinado o tamanho, N, do dicionário, μ_{KLMS} e α, por (6.5) o valor do tamanho do passo no KSIG foi de $\mu_{KSIG} = 2.409 \times 10^4$.

Neste experimento foram executadas 300 simulações de Monte-Carlo. Para cada simulação extraiu-se a curva de aprendizagem, no final foi feita a média simples do MSE para cada iteração, obtendo uma curva de aprendizagem com um MSE médio para cada iteração de todas as repetições da tarefa.

O resultado do experimento está ilustrado na Figura 6.6, na qual a curva de aprendizagem do KSIG está abaixo da curva do KLMS até a iteração 2000 aproximadamente, e a partir deste ponto, ambas as curvas apresentam mesma posição e inclinação, ambas em estado estacionário. De fato, a curva do KSIG desde a iteração 500 já se encontra no ponto de estado estacionário diferentemente do KLMS cuja curva está neste estado após pouco mais de 1500 iterações. Isto significa que, a convergência do KSIG ocorreu mais rápido que no outro algoritmo na versão com dicionário pré-definido, com um número significativo de iterações, metade das iterações realizadas.

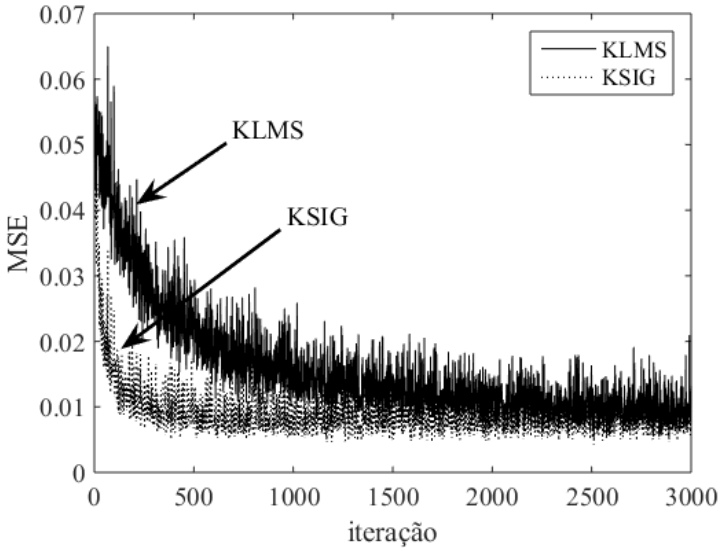

Figura 6.6: Curva de aprendizagem da identificação de sistema no KSIG e KLMS com dicionário pré-definido.

6.3.2 Experimento 2

O segundo experimento, que pode ser encontrado em [Parreira 2012], consiste na identificação de um sistema não linear, definido como

$$x(i) = \frac{x(i-1)}{1+x^2(i-1)} + u^3(i-1) \tag{6.10}$$

$$d(i) = x(i) + z(i) \tag{6.11}$$

em que $x(i)$ é a saída do sistema; $u(i)$ é o sinal de entrada; $d(i)$ é o sinal desejado; e $z(i)$ um ruído gaussiano adicionado a saída do sistema $x(i)$. Neste problema a entrada do filtro é o $u(i)$ e saída deseja $d(i)$.

No experimento, o ruído $z(i)$ foi gerado com média zero e variância $\sigma_z^2 = 10^{-4}$. A entrada $u(i)$ é uma sequência gaussiana de média nula e variância $\sigma_u = 0.15$, enquanto o *kernel* escolhido foi o Gaussiano, com tamanho $h = 0.025$; com o tamanho do passo para o KLMS como $\mu_{KLMS} = 0.07$. O dicionário, que é o mesmo para ambos os algoritmos, tem tamanho $N = 6$. Para o KSIG, o coeficiente de inclinação $\alpha = 3.1$. E, por (6.5), dados μ_{KLMS}, α e N, o valor do tamanho do passo do KSIG é igual a $\mu_{KSIG} = 0.0039$.

Foram realizadas 300 simulações de Monte Carlo O resultado das simulações está expresso nas curvas de aprendizagem do KSIG e do KLMS ilustradas na Figura (6.7). Pela

figura, é possível observar que a curva de aprendizagem do KSIG por volta da iteração 2500, encontra-se no estado estacionário, o qual é observado na curva do KLMS apenas a partir da iteração 5000. Este resultado ratifica a convergência mais rápida do algoritmo KSIG com relação ao KLMS

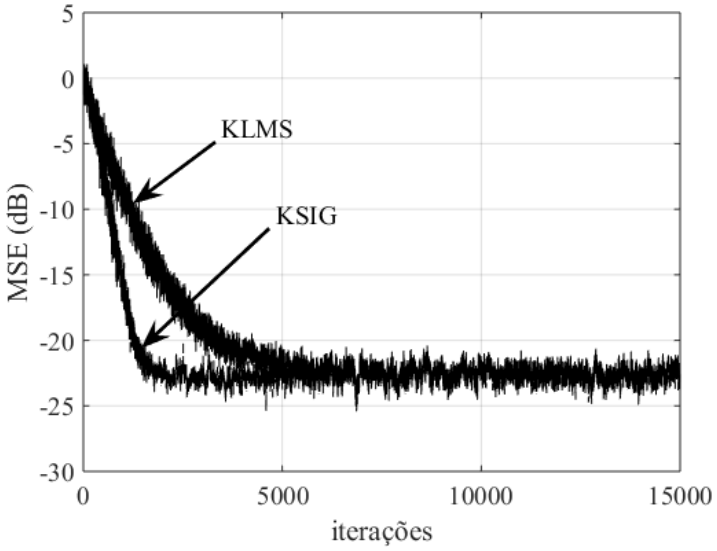

Figura 6.7: Curva de aprendizagem da segunda identificação de sistema não linear do KSIG e do KLMS com dicionário pré-definido.

6.4 Considerações finais

Este capítulo apresentou experimentos que constituem uma forma de vislumbrar a convergência mais rápida do algoritmo KSIG, seja em sua versão com ou sem dicionário pré-definido, com relação a um algoritmo bastante estudado na literatura, o KLMS.

Os resultados dos experimentos demonstraram que as curvas de aprendizagem do KSIG se mostraram sempre abaixo da do KLMS até que ambos atingissem o estado estacionário, que, para o KSIG, se deu em poucas iterações antes do KLMS; já, para o KSIG com dicionário pré-definido a entrada no estado estacionário se deu em mais de 500 iterações de antecedência, em relação ao KLMS, este também em versão com dicionário pré-definido.

Capítulo 7

Conclusões

A partir da revisão da literatura, pode-se conceber um modelo geral acerca das técnicas de *kernel* e da sua aplicação no Sigmoide, o que foi essencial para a criação do KSIG e sua versão com dicionário pré-definido.

No que se refere à análise de convergência, verificou-se que há na literatura dois tipos de análise para algoritmos que utilizam *kernel*, uma baseada na energia do filtro, o modelo geral de análise de estado estacionário proposta por [Yousef and Sayed 2001], e um segundo modelo de [Parreira 2012], no qual é tomada a teoria dos discos de Gerschgorin e a teoria da estabilidade de sistemas de controle. Em ambas as metodologias é adotada análise sobre algum momento estatístico

Observou-se que a análise de convergência do algoritmo de filtragem adaptativa KMC é baseada no modelo que faz uso da energia do filtro e, pela sua similaridade com o KSIG, foi adotado este modelo de análise de convergência do KSIG. Os resultados da análise demonstraram que a convergência do KSIG acontece desde que a relação em (4.27) seja respeitada.

Os estudos sobre a convergência do algoritmo Sigmoide propiciaram utilizar o mesmo método para análise da convergência do KSIG com dicionário pré-definido, dada a semelhança entre estes algoritmos. Como resultado, obteve-se que a convergência do KSIG ocorre desde que (5.21) aconteça.

Outra contribuição deste trabalho se refere ao cálculo da complexidade computacional dos algoritmos, que no KSIG foi de $O(n^2)$ na fase de treinamento e $O(n)$ para cálculo da saída, já para o KSIG com dicionário pré-definido de $O(tn)$ na fase de treinamento e $O(t)$ no cálculo da saída do filtro, em que n é o número de dados de teste e t o número de dados

no dicionário. Destas diferenças de complexidade na fase de treinamento e no cálculo da saída, percebeu-se que o algoritmo com dicionário pré-definido é menor em complexidade, portanto mais eficiente, pois, com $t < n$ então $O(nt) < O(n^2), O(t) < O(n)$.

Na parte experimental foi possível verificar a convergência mais rápida dos algoritmos propostos quando comparados ao KLMS em sua versão sem e com dicionário pré-definido. Isto, considerando que o KLMS e o KSIG possuíam o mesmo desajuste em estado estacionário, consideração possível a partir da relação entre suas taxas de aprendizagem em (6.5), elaborada neste trabalho.

Assim, pode-se concluir que o objetivo traçado para este trabalho foi alcançado, pois tanto o KSIG quanto o KSIG com dicionário pré-definido tiveram descritas a sua construção, sua análise de convergência, e alguns experimentos.

Como trabalhos futuros, há a possibilidade da análise da relação entre as constantes de tempo teóricas do KSIG e do KLMS para estabelecer uma relação de superioridade ou inferioridade dos algoritmos. Além disto, uma análise desconsiderando a equidade do desajuste possibilitaria vislumbrar se o algoritmo KSIG, além da convergência, pode apresentar uma melhora na filtragem diminuindo o erro pela melhor combinação de taxa de aprendizagem e coeficiente de inclinação.

REFERÊNCIAS

[Bao and Panahi 2009] Bao, H. and Panahi, I. (2009). Active noise control based on kernel least-mean-square algorithm. In *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, pages 642 – 644. IEEE.

[Chen et al. 2012a] Chen, B., Zhao, S., Zhu, P., and Príncipe, J. C. (2012a). Mean square convergence analysis for kernel least mean square algorithm. *Signal Processing*, 92:645 – 654.

[Chen et al. 2012b] Chen, B., Zhao, S., Zhu, P., and Principe, J. C. (2012b). Quantized kernel least mean square algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 23(1):22–32.

[Chen et al. 2014] Chen, J., Gao, W., Richard, C., and Bermudez, J. C. M. (2014). Convergence analysis of kernel lms algorithm with pre-tuned dictionary. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7243–7247.

[Chen and Zhang 2007] Chen, W. and Zhang, H. (2007). *The Condition of Kernelizing an Algorithm and an Equivalence Between Kernel Methods*, pages 338–345. Springer Berlin Heidelberg.

[Constantin and Lengellé 2013] Constantin, I. and Lengellé, R. (2013). Performance analysis of kernel apdaptive filters based on lms algorithm. In *Procedia Computer Science*, volume 60, pages 39 – 45. Elsevier.

[Debnath and Mikusiński 2005] Debnath, L. and Mikusiński, P. (2005). *Hilbert Spaces with Applications*. Elsevier Academic Press.

[Douglas and Meng 1994] Douglas, S. C. and Meng, T. H. Y. (1994). Stochastic gradient adaptation under general error criteria. *IEEE Transactions on Signal Processing*, 42(6):1335–1351.

[Gao et al. 2015] Gao, W., Chen, J., Richard, C., Bermudez, J. C. M., and Huang, J. (2015). Convergence analysis of the augmented complex klms algorithm with pre-tuned dictionary. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2006–2010.

[Haykin 2014] Haykin, S. (2014). *Adaptive Filter Theory*. Prentice-Hall information and system sciences series. Prentice Hall, 5th edition.

[Liu et al. 2015] Liu, F., Yuan, W., Ma, Y., Zhou, Y., and Liu, H. (2015). New enhanced robust kernel least mean square adaptive filtering algorithm. In *Estimation, Detection and Information Fusion (ICEDIF), 2015 International Conference on*, pages 282 – 285. IEEE.

[Liu et al. 2008] Liu, W., Principe, J. C., and Haykin, S. (2008). The kernel least-mean-square algorithm. *Signal Processing, IEEE Transactions on*, 56:543 – 554.

[Liu et al. 2010] Liu, W., Principe, J. C., and Haykin, S. (2010). *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley Publishing, 1st edition.

[Papoulis 1991] Papoulis, A. (1991). *Probability, random variables, and stochastic processes*. McGraw-Hill series in electrical engineering. McGraw-Hill, New York.

[Parreira et al. 2012] Parreira, W., Bermudez, J., Richard, C., and Tourneret, J.-Y. (2012). Stochastic behavior analysis of the gaussian kernel least-mean-square algorithm. *Signal Processing, IEEE Transactions on*, 60:2208 – 2222.

[Parreira 2012] Parreira, W. D. (2012). Comportamento estocástico do algoritmo kernel least-mean-square. Master's thesis, PGEEL/UFSC.

[Paul and Ogunfunmi 2012] Paul, T. and Ogunfunmi, T. (2012). Analysis of the convergence behavior of the complex gaussian kernel lms algorithm. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 2761 – 2764. IEEE.

[Pokharel and Principe 2014] Pokharel, R. and Principe, J. (2014). Quantized mixture kernel least mean square. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 4168 – 4174. IEEE.

[Santana et al. 2006a] Santana, E., Yasuda, Y., Takeuchi, Y., and Barros, A. K. (2006a). Adaptive estimation of impedance cardiographic signal by the sigmoidal algorithm. In *MeMea 2006, IEEE International Workshop on Medical Measurement and Application, Tokyo, Japan, September 6-8, 2006*, pages 117–120.

[Santana et al. 2006b] Santana, E. E., Principe, J. C., Barros, A. K., and Freire, R. C. S. (2006b). An adaptive algorithm based on the sigmoidal function. In *Neural Networks, 2006. SBRN '06. Ninth Brazilian Symposium on*, pages 1–5.

[Santana 2006] Santana, E. E. C. (2006). Estudo e desenvolvimento de uma família de algoritmos não lineares para filtragem adaptativa. Master's thesis, PPGEE/UFMA.

[Santana Júnior 2012] Santana Júnior, E. E. C. (2012). Extração cega de sinais com estruturas temporais utilizando espaços de hilbert reproduzidos por kerneis. Master's thesis, PPGEE/UFMA.

[Sayed 2008] Sayed, A. H. (2008). *Adaptive Filters*. Wiley-IEEE Press.

[Silva et al. 2015] Silva, É. P. d., Estombelo-Montesco, C. A., and Santana, E. (2015). Ksig: Improving the convergence rate in adaptive filtering using kernel hilbert space. In *Intelligent Systems (BRACIS), 2015 Brazilian Conference on*, pages 294–298.

[Takizawa et al. 2015] Takizawa, M.-A., Yukawa, M., and Richard, C. (2015). A stochastic behavior analysis of stochastic restricted-gradient descent algorithm in reproducing kernel hilbert spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2001 – 2005. IEEE.

[Theodoridis and Koutroumbas 2008] Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition, Fourth Edition*. Academic Press, 4th edition.

[Widrow and Stearns 1985] Widrow, B. and Stearns, S. D. (1985). *Adaptive Signal Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Yousef and Sayed 2001] Yousef, N. and Sayed, A. (2001). A unified approach to the steady-state and tracking analyses of adaptive filters. *Signal Processing, IEEE Transactions on*, 49(2):314–324.

[Zhao et al. 2015] Zhao, J., Liao, X., Wang, S., and Tse, C. K. (2015). Kernel least mean square with single feedback. *IEEE Signal Processing Letters*, 22(7):953–957.

[Zhao et al. 2011] Zhao, S., Chen, B., and Príncipe, J. C. (2011). Kernel adaptive filtering with maximum correntropy criterion. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2012–2017.