Processamento Digital de Sinais

Introdução ao Processamento Adaptativo de Sinais Digitais

1.Introdução

O objetivo deste estudo é estabelecer os conceitos básicos em que se fundamenta o processamento adaptativo de sinais (*Adaptive Digital Signal Processing* - ADSP). Para tal propósito é apresentada a descrição e a análise do Combinador Linear Adaptativo (*Adaptive Linear Combiner* - ALC), que é o módulo funcional básico dos sistemas utilizados em ADSP. O processo adaptativo aplicado ao ALC é descrito no contexto da Filtragem Adaptativa. É apresentado o processo de minimização de funções de custo. As funções de custo são convenientemente definidas, de forma a medir o quanto o processo adotado está sendo incapaz de reduzir o erro de adaptação. Os procedimentos adaptativos envolvem deslocar o ponto de operação de algum particular algoritmo em direção ao ponto de mínimo (ou vale) da superfície de erro definida para avaliar o desempenho deste algoritmo.

2. Sistemas Adaptativos

Um sistema adaptativo é aquele cuja estrutura é alterável ou ajustável, de tal forma que seu comportamento ou desempenho melhore, de acordo com algum critério desejado, através da exposição ao ambiente no qual está inserido.

O ADSP faz de nós, pesquisadores, meros (e tardios) imitadores da natureza. Os exemplos de processos naturais adaptativos nos cercam de forma abundante e estão presentes em múltiplas funções, em nosso próprio organismo. Apesar de sermos

organismos biológicos – em muitos sentidos – adaptativos, inseridos em um ecossistema essencialmente adaptativo e submetidos a processos sociais que resultam em alterações individuais e coletivas ajustadas a mudanças no ambiente cultural que nos cerca, demoramos até 1960 para começar a aplicar tal heurística a nossos problemas de engenharia...

Modelar um sistema adaptativo, por exemplo, nada mais é do que imitar o comportamento da pupila face a variações da quantidade de luz incidente, ou imitar um simples organismo biológico cuja estrutura se torna melhor ajustada a sobreviver e a se multiplicar em um ambiente mutável.

Em engenharia, um exemplo simples de um sistema adaptativo é o controle de ganho automático usado em receptores de rádio e televisão. A função deste circuito é ajustar a sensibilidade do receptor inversamente, à medida que é elevado o nível médio do sinal recebido. O receptor é apto a adaptar seu fator de amplificação global para um grande intervalo de níveis do sinal recebido (ampla faixa dinâmica), produzindo um nível de sinal de saída o mais constante possível.

As aplicações de sistemas adaptativos à engenharia abrangem uma ampla gama de campos, entre eles (apenas para citar alguns) estão: aplicações em comunicações, radar, sonar, sismologia, projeto mecânico, sistemas de navegação, engenharia biomédica, etc..

Os sistemas adaptativos costumam apresentar todas ou algumas das seguintes características:

1. Adaptação automática (auto-otimizada) em face a ambientes e sistemas que variam com o tempo (não-estacionários).

2. Podem ser treinados para desempenhar tarefas específicas, tais como filtragem e tomada de decisões; ou seja, podem sintetizar sistemas responsáveis por estas tarefas através de um processo de treinamento.

3. Devido às características 1 e 2, não requerem procedimentos de síntese elaborados, usualmente necessários para sistemas não-adaptativos. Sistemas adaptativos tendem a se "auto-projetar".

4. Podem extrapolar um modelo de comportamento para lidar com novas situações, após terem sido treinados com um finito e freqüentemente pequeno número de sinais (ou padrões) de treino.

5. Dentro de certos limites, podem se auto-restaurar; ou seja, podem se adaptar "ao redor" de certos tipos de defeitos internos.

6. Podem ser descritos como sistemas lineares ou não-lineares, com parâmetros que variam no tempo (um exemplo de sistema adaptativo não-linear é uma Rede Neural Artificial).

7. Em geral, são mais complexos e difíceis de analisar do que sistemas não-adaptativos, mas oferecem a possibilidade de apresentar um desempenho substancialmente melhor quando as características dos sinais de entrada são desconhecidas ou variantes no tempo.

A principal (e essencial) propriedade de um sistema adaptativo é seu desempenho auto-ajustável, variante com o tempo.

Não necessitaríamos de sistemas adaptativos se fôssemos capazes de projetar um sistema "fixo" (não-adaptativo), considerado ótimo, para o qual tivéssemos previsto todas as possíveis condições de entrada (pelo menos em um sentido estatístico) e do qual soubéssemos exatamente qual desempenho esperaríamos sob cada uma destas condições. Precisaríamos estabelecer um critério específico para avaliação do desempenho do sistema a ser projetado e escolher (dentre uma classe restrita de projetos desenvolvidos *a priori*) o modelo de sistema que apresentasse melhor desempenho, de acordo com o critério previamente selecionado.

No entanto, na maioria dos problemas extraídos do universo dos problemas práticos (especialmente aqueles problemas com que nossos chefes ou orientadores costumam nos presentear), não é conhecido o intervalo completo das condições de entrada (nem mesmo estatisticamente) ou as condições podem mudar ao longo do tempo. Em tais circunstâncias, um sistema adaptativo pode buscar continuadamente a classe ótima dentre um conjunto de classes de possibilidades, usando um processo de busca ordenado, apresentando, assim, um desempenho superior comparativamente ao desempenho do sistema não-adaptativo.

O desenvolvimento dos algoritmos adaptativos é devido grandemente ao clássico trabalho de Widrow e Hoff (1960), no qual foi apresentado, pela primeira vez o algoritmo *Least-Mean-Square* (LMS), também conhecido como a Regra Delta.

O ALC é o bloco funcional básico das aplicações concernentes a processamento de sinais. Além da imensurável contribuição à ciência trazida pelos professores Widrow e Hoff através do processamento adaptativo de sinais, outra contribuição de valor no mínimo identicamente comparável foi a possibilidade de o modelo do ALC e a regra de adaptação proposta terem inspirado o desenvolvimento do Perceptron elementar. O Perceptron elementar é a menor estrutura que pode apresentar uma Rede Neural Artificial, reconhecidamente a ferramenta mais "poderosa", senão a única capaz de lidar com processos (geradores de conjuntos de dados) não-lineares e/ou não-estacionários, de cujas descrições não se tenha NENHUM conhecimento *a priori*, quer determinístico, quer estatístico.

3. O Combinador Linear Adaptativo

O grafo de fluxo de sinal de um combinador linear adaptativo é mostrado na Figura 1. Observe que o ALC possui M nós de entrada, aos quais são apresentados N vetores de entrada $\underline{x}(i) \in \Re^M$, $i = 0,1,\cdots,N-1$, um somador (ou combinador linear) e um único sinal de saída, $y(i)$. As conexões entre os nós de entrada do ALC e o somador são representadas

por um conjunto de pesos ajustáveis, descrito pelo vetor $\underline{w}(i) \in \Re^M$, $i = 0,1,\cdots,N-1$, sendo *i* o índice temporal do passo de adaptação do algoritmo.

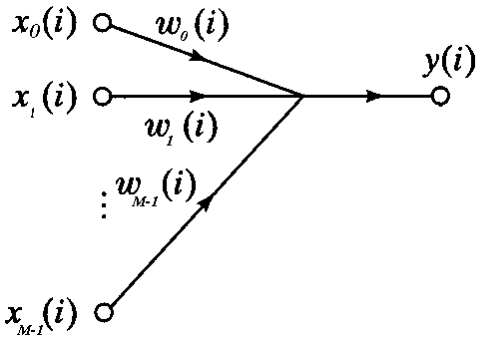

Figura 1: Grafo de fluxo de sinal para o combinador linear adaptativo.

O combinador é dito linear porque, para um conjunto já estabilizado de pesos, sua saída é uma combinação linear dos componentes da entrada. Entretanto, durante o processo de ajuste dos pesos, os pesos se tornam função dos vetores de entrada, e a saída do combinador linear não é uma função linear da entrada.

Há duas maneiras de interpretar fisicamente os elementos do vetor de entrada do combinador linear adaptativo:

1. Os *M* elementos do vetor $\underline{x}(i)$ podem originar-se de *M* distintas fontes de informação localizadas em diferentes pontos no espaço, sendo todos os *M* elementos obtidos no mesmo instante, de todas as *M* fontes (como, por exemplo, em um *array* de antenas adaptativas).

2. Os *M* elementos do vetor $\underline{x}(i)$ representam o valor presente e os $M-1$ valores passados de amostras seqüencialmente originadas de uma única fonte de informação (como, por exemplo, em um preditor de séries temporais adaptativo).

Para o caso descrito em 2, o processador adaptativo pode ser implementado com um ALC e elementos de atraso, caso em que é denominado filtro tranversal adaptativo. Este

tipo de filtro possui um grande número de aplicações nos campos de modelamento adaptativo e ADSP. A maior parte dos sistemas adaptativos existentes são baseados no uso de filtros transversais adaptativos.

Para o caso descrito em 1, em muitas aplicações, será necessário incluir um peso adicional variável denominado *bias*, cuja entrada do ALC associada é mantida fixa em +1.

A relação entrada-saída para o ALC mostrado na Figura 1 pode ser descrita por

$$y(i) = \sum_{k=0}^{M-1} w_k(i) x_k(i) \tag{1}$$

que, sob a forma vetorial, pode ser expressa como

$$y(i) = \underline{w}^T(i)\,\underline{x}(i) = \underline{x}^T(i)\,\underline{w}(i) \ . \tag{2}$$

Tendo descrito a operação do ALC, podemos agora proceder à discussão sobre a adaptação dos vetores de pesos, ao longo das iterações i . Esta discussão será desenvolvida, para fins didáticos, no contexto da Filtragem Linear Adaptativa, em que o ALC é usado para modelar um desconhecido sistema dinâmico.

4. A Filtragem Adaptativa

Consideremos um sistema dinâmico $\mathbf{\Gamma}$ cuja caracterização matemática é **desconhecida**. O máximo de conhecimento que temos a respeito de $\mathbf{\Gamma}$ é um conjunto finito de dados, que é um subconjunto χ do universo Ω de todos os possíveis mapeamentos entrada-saída que podem ser gerados pelo sistema $\mathbf{\Gamma}$.

Suponhamos que os elementos do subconjunto χ sejam pares $(\underline{x}(i), d(i)) \in \chi$, onde :

$\underline{x}(i)$ é o i-ésimo vetor M-dimensional de χ aplicado na entrada de $\mathbf{\Gamma}$ e

$d(i)$ é a saída de $\mathbf{\Gamma}$ à entrada $\underline{x}(i)$, $i = 0,1,\cdots,N-1$, sendo

N o número de elementos de χ.

Especificamente, quando um estímulo real *M*-dimensional $\underline{x}(i) \in \Re^M$ é aplicado aos *M* nós de entrada do sistema **Γ**, **Γ** responde gerando a saída escalar $d(i)$, como mostra a Figura 2(a).

A dimensão *M* dos vetores $\underline{x}(i)$ é usualmente referida como dimensionalidade do espaço de entrada.

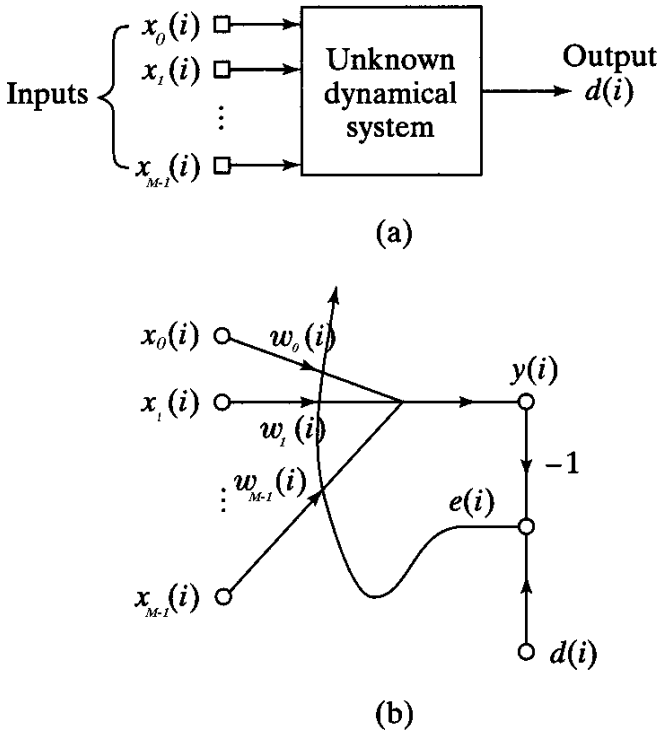

Figura 2: (a) Sistema dinâmico desconhecido **Γ**. (b) Grafo de fluxo de sinal para o modelo adaptativo do sistema.

Portanto, o comportamento externo do sistema **Γ** é descrito pelo mapeamento

$$\Gamma : \underline{x}(i) \in \Re^M \rightarrow d(i) \in \Re, \quad i = 0,1,\cdots,N-1, \tag{3}$$

onde $\underline{x}(i)$ é o *i*-ésimo vetor de χ, definido por

$$\underline{x}(i) = [x_0(i) \quad x_1(i) \quad \cdots \quad x_{M-1}(i)]^T. \tag{4}$$

Note que, na grande maioria dos casos, também não se conhece com precisão a distribuição de probabilidade dos elementos do conjunto χ , de modo que a tentativa de resolver um problema de filtragem através de uma abordagem estatística (através da matriz de correlação, por exemplo) não raro conduz a resultados não satisfatórios.

Conforme discutimos na Seção 3, um estímulo $\underline{x}(i)$ aplicado a um sistema $\boldsymbol{\Gamma}$ pode originar-se de dois cenários fundamentais, um espacial e outro temporal.

Um problema clássico em filtragem adaptativa, conhecido como identificação de sistema, é determinar o modelo que rege o comportamento do sistema dinâmico desconhecido $\boldsymbol{\Gamma}$, caracterizado por (3), utilizando para tanto um único ALC. O ALC opera sob a influência de um algoritmo **A** que controla os ajustes necessários às transmitâncias (≡pesos) para que, à medida que os ajustes se sucedem, o mapeamento efetuado pelo ALC tenda a aproximar o mapeamento efetuado pelo sistema $\boldsymbol{\Gamma}$. Este processo de ajustes sucessivos dos pesos do ALC é efetuado observando as seguintes características:

- O algoritmo **A** inicia o processo de ajuste a partir de um conjunto de transmitâncias (≡pesos), com valor inicial arbitrário atribuído a cada uma delas.

- O algoritmo **A** ajusta as transmitâncias do ALC continuamente ao longo do intervalo de operação do sistema $\boldsymbol{\Gamma}$, para permitir que eventuais variações no padrão de comportamento de $\boldsymbol{\Gamma}$ (variações na estatística do comportamento de $\boldsymbol{\Gamma}$) também possam influenciar o processo de ajuste.

- Para cada valor de i, o algoritmo **A** deve ser rápido o suficiente para ajustar todas as M transmitâncias do ALC dentro do intervalo de tempo que transcorre entre a ocorrência das entradas $\underline{x}(i)$ e $\underline{x}(i+1)$.

A Figura 2 (b) mostra o grafo de fluxo de sinal de um filtro adaptativo baseado no ALC, aplicado ao contexto de identificação do sistema desconhecido **Γ**. A operação do filtro consiste de dois processos continuamente executados em seqüência:

1. Processo de Filtragem, o qual envolve o cômputo de dois sinais :

   1.1. Uma saída, denotada por $y(i)$, que é produzida em resposta ao vetor estímulo $\underline{x}(i)$.

   1.2. Um sinal de erro, denotado por $e(i)$, que é obtido pela comparação da saída $y(i)$ com a saída desejada $d(i)$ correspondente, sendo $d(i)$ produzida por **Γ** quando o estímulo $\underline{x}(i)$ é aplicado à sua entrada. Em outras palavras, $d(i)$ constitui a resposta desejada ou o sinal alvo (*target signal*).

2. Processo de Adaptação, o qual envolve o ajuste automático dos pesos do ALC através de um algoritmo **A**, tendo como base o sinal de erro $e(i)$.

Desta maneira, a combinação destes dois processos (operando em conjunto) constitui um elo de realimentação (*feedback loop*) na operação do ALC. Uma vez tendo sido aplicados todos os N vetores $\underline{x}(i)$ à entrada do ALC e tendo sido executados todos os N ajustes através do algoritmo **A**, repete-se novamente as etapas 1 e 2 até que $e(n)$ seja suficientemente pequeno, onde $e(n)$ é o valor do sinal de erro e em um instante n qualquer da operação do filtro.

Uma vez que o ALC é estritamente linear, a saída $y(i)$ é dada por,

$$y(i) = \sum_{k=0}^{M-1} w_k(i) x_k(i), \tag{5}$$

onde $w_k(i)$ é o valor da k-ésima transmitância medida no instante discreto i. Em forma vetorial, podemos expressar $y(i)$ como o produto interno entre os vetores $\underline{x}(i)$ e $\underline{w}(i)$, conforme segue:

$$y(i) = \underline{x}^T(i)\,\underline{w}(i), \tag{6}$$

onde

$$\underline{w}(i) = [w_0(i) \quad w_1(i) \quad \cdots \quad w_{M-1}(i)]^T. \tag{7}$$

A saída $y(i)$ do ALC é comparada com a saída $d(i)$ do sistema desconhecido $\boldsymbol{\Gamma}$ no instante discreto i. Tipicamente, a comparação é estabelecida pela diferença entre $d(i)$ e $y(i)$, portanto o processo de comparação define o sinal de erro $e(i)$ dado por

$$e(i) = d(i) - y(i). \tag{8}$$

Observe de (6) e (8) que o sinal de erro $e(i)$ depende do vetor $\underline{w}(i)$. Note também que $\underline{w}(i)$ é o parâmetro livre do ALC que será sucessivamente ajustado pelo algoritmo **A,** objetivando minimizar $e(i)$. Portanto, para que se possa medir a ineficiência do processo de ajuste de $\underline{w}$, e, em função disto adotar as correções necessárias, é útil definir uma função $\mathrm{J}(e)$ (ou $\mathrm{J}(\underline{w})$, já que e depende de $\underline{w}$) que defina da maneira o mais inequívoca possível o "grau de incompetência" do ALC em aproximar sua saída $y(i)$ de $d(i)$.

A função $\mathrm{J}(\underline{w})$, cujo valor resultante é uma grandeza escalar real, é denominada de função de custo. A definição de $\mathrm{J}(\underline{w})$ deve ser tal que meça o quanto o processo de ajuste está sendo incapaz de reduzir o erro $e(i)$ entre $d(i)$ e $y(i)$. Por exemplo, uma popular definição de J é $\mathrm{J} = \mathrm{J}(e) = \frac{1}{2}e^2$. Em especial, o algoritmo **A** e a função de custo J idealmente devem ser tais que $\mathrm{J}(\underline{w}(n+1)) < \mathrm{J}(\underline{w}(n))$, onde n é um instante qualquer do processo de ajuste.

4.1 O Processo de Minimização da Função de Custo

Consideraremos neste estudo o denominado Algoritmo de Descida Mais Íngreme (SD – *Steepest Descent*), por ser um dos mais utilizados, e de baixo custo computacional. Existem, no entanto, outros algoritmos, como o Método de Newton e o Método de Gauss-Newton, que são descritos em [5].

No algoritmo SD os sucessivos ajustes aplicados à $\underline{w}$ estão na direção da descida mais íngreme da superfície $S = \mathrm{H}(w_0, w_1, \ldots, w_{M-1})$ formada pelos valores escalares H do conjunto imagem de $\mathrm{J}(\underline{w})$ em função do domínio M-dimensional $\underline{w} = [w_0 \quad w_1 \quad \cdots \quad w_{M-1}]^T$, isto é, $\mathrm{H} = \mathrm{J}(\underline{w})$. Em outras palavras, os sucessivos ajustes aplicados à $\underline{w}$ estão na direção oposta do vetor gradiente $\underline{\nabla}\,\mathrm{J}(\underline{w})$ da superfície formada por $\mathrm{J}(\underline{w})$.

Uma interpretação intuitiva do método SD é imaginarmos um observador míope que enxergue apenas a distância de um passo ao seu redor, caminhando sobre a superfície $\mathrm{J}(\underline{w})$, e cujo objetivo é chegar ao ponto de cota mínima de $\mathrm{J}(\underline{w})$ o mais rapidamente possível. No instante n o observador, localizado na coordenada $\underline{w}(n)$, olha ao redor e localiza a direção $\underline{\nabla}\,\mathrm{J}(\underline{w}(n))$ de subida mais íngreme em $\mathrm{J}(\underline{w})$. A seguir o observador dá um passo na direção contrária à $\underline{\nabla}\,\mathrm{J}(\underline{w}(n))$ de tamanho proporcional à declividade $|\underline{\nabla}\,\mathrm{J}(\underline{w}(n))|$ encontrada na coordenada $\underline{w}(n)$ e desloca-se para a nova coordenada $\underline{w}(n+1)$. Supondo que não existam mínimos locais (buracos e/ou depressões) na superfície $\mathrm{J}(\underline{w})$ de diâmetro algo maior que o passo do observador, o mesmo atingirá a cota mínima $\mathrm{J}(\underline{w}^*)$ na coordenada $\underline{w}^*$ após repetir este procedimento um número suficiente de vezes.

Formalmente, o algoritmo SD é descrito por

$$\underline{w}(n+1) = \underline{w}(n) - \eta\,\underline{\nabla}\,\mathrm{J}(\underline{w}(n)), \tag{9}$$

onde $\eta > 0$ é chamado passo de adaptação (*stepsize*) ou razão de aprendizado (*learning rate*).

Para a função de custo $\mathrm{J}(n) = \mathrm{J}(\underline{w}(n)) = \frac{1}{2}e^2(n)$, a superfície $\mathrm{J}(\underline{w})$ é um parabolóide M+1-dimensional (i.e., uma "tigela" em $\Re^{M+1}$, não necessariamente de boca circular), e, portanto, apresenta um mínimo global mas não apresenta mínimos locais (qualquer função quadrática possui um e somente um mínimo). Por isto, para esta função de custo, o algoritmo SD converge para $\underline{w}^*$ de modo lento mas seguro desde que η não seja demasiadamente grande (caso em que o observador míope pularia fora da "tigela").

➡ É importante observar que o passo de adaptação η tem profunda influência na trajetória do "observador míope" até a convergência para $\underline{w}^*$, e, não raro, o valor de η é alterado convenientemente ao longo do processo de minimização de J para que η se adeqüe às exigências da coordenada instantânea da trajetória. Para filtros cuja função de custo é $\mathrm{J}(n) = \mathrm{J}(\underline{w}(n)) = \frac{1}{2}e^2(n)$ são válidas as seguintes observações:

- Para η pequeno, a resposta transiente do algoritmo SD é super-amortecida (*overdamped*) e a trajetória percorrida por $\underline{w}(n)$ é uma curva suave em $\Re^M$, conforme mostrado na Figura 3(a).

- Para η grande, a resposta transiente do algoritmo SD é sub-amortecida (*underdamped*) e a trajetória percorrida por $\underline{w}(n)$ é uma curva em zig-zag (oscilatória) em $\Re^M$, conforme mostrado na Figura 3(b).

- Para η acima de um determinado valor crítico, o algoritmo SD torna-se instável e termina divergindo.

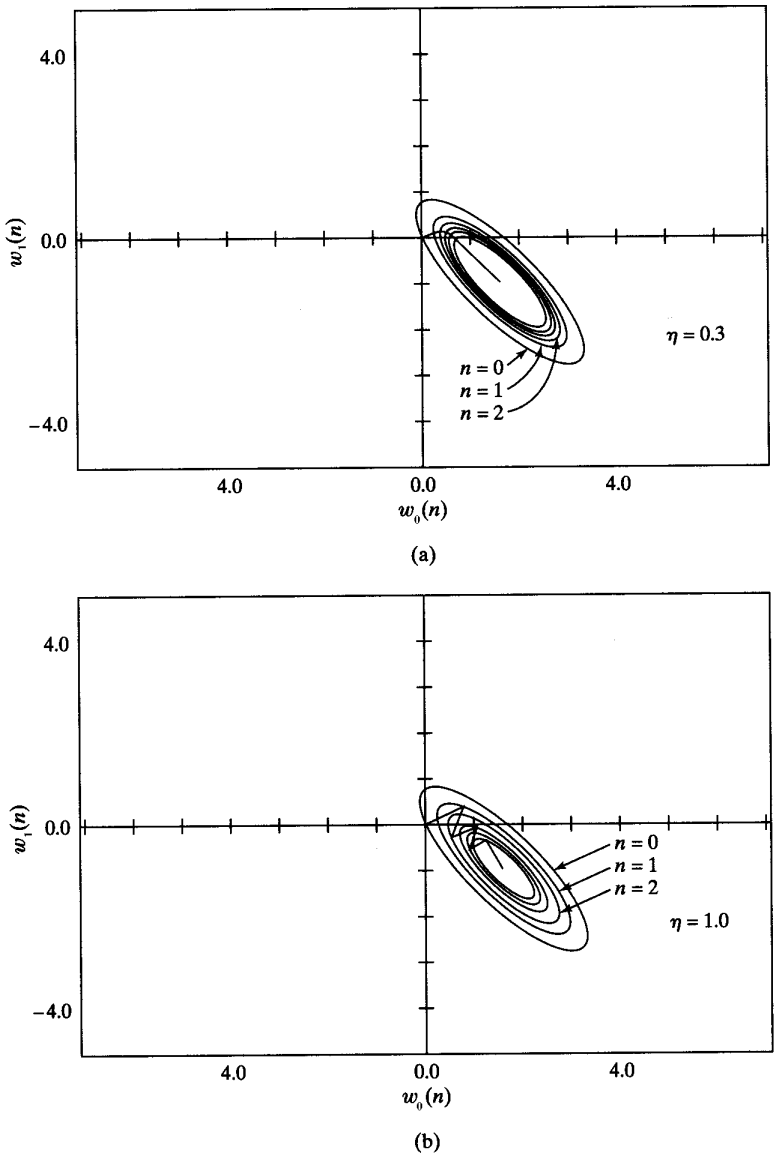

Figura 3: Trajetória do método de Descida Mais Íngreme (*steepest descent*) em um espaço bi-dimensional, para dois valores diferentes de parâmetros razão de aprendizado: (a) $\eta = 0.3$, (b) $\eta = 1.0$. As coordenadas w_0 e w_1 são elementos do vetor de pesos $\underline{w}$.

5. O Algoritmo LMS

O Algoritmo LMS (*Least Mean Square*) procura minimizar uma função de custo J definida por $\mathrm{J} = \mathrm{J}(e) = \frac{1}{2}e^2$ com base nos valores instantâneos da mesma, isto é,

$$\mathrm{J} = \mathrm{J}(e(n)) = \tfrac{1}{2}e^2(n), \tag{10}$$

onde $e(n)$ é o sinal de erro medido em um instante *n* qualquer do processo de minimização de J.

Nota: Diferentemente do algoritmo **LMS**, apenas como exemplo comparativo, o algoritmo **RLS** (*Recursive Least Squares*) baseia-se em uma função de custo J definida por uma soma ponderada do erro quadrático $e^2(n)$ do instante atual *n* com os erros quadráticos ocorridos anteriormente a *n*, isto é, $\mathrm{J}(n) = \beta_0 e^2(n) + \beta_1 e^2(n-1) + \beta_2 e^2(n-2) + \cdots$, onde $0 < \beta_k \leq 1$ são os coeficientes de ponderação. Os coeficientes β_k são tais que $\beta_k > \beta_{k+1}$, de forma que erros ocorridos em um passado distante sejam "esquecidos" por J objetivando minimizar sua influência sobre ela. Assim, se o conjunto χ de $(\underline{x}(n), d(n)) \in \chi$ (entradas, saídas desejadas) não for um processo estacionário (i.e., os parâmetros estatísticos de χ variam com o tempo), o "esquecer do passado" auxilia a melhorar a velocidade de convergência. No entanto, como é fácil perceber, o custo computacional do algoritmo RLS é maior que o do algoritmo LMS, o que o torna inadequado para certas aplicações que requeiram alta velocidade de processamento, como por exemplo, em equalização de canal para um *link* de microondas com alta taxa de transmissão.

O gradiente $\underline{\nabla}\,\mathrm{J}(\underline{w}(n))$ da superfície $\mathrm{J}(n) = \mathrm{J}(\underline{w}(n)) = \frac{1}{2}e^2(n)$ no instante *n* é obtido através da variação de $\mathrm{J}(\underline{w}(n))$ em resposta a uma variação infinitesimal na coordenada $\underline{w}(n)$, isto é,

$$\underline{\nabla}\,\mathrm{J}(\underline{w}(n)) = \frac{\partial\,\mathrm{J}(\underline{w}(n))}{\partial\,\underline{w}(n)}, \tag{11}$$

mas, visto que $\mathrm{J}(\underline{w}(n)) = \frac{1}{2} e^2(n)$, temos

$$\underline{\nabla}\,\mathrm{J}(\underline{w}(n)) = \frac{\partial\left\{\frac{1}{2} e^2(n)\right\}}{\partial \underline{w}(n)} = e(n)\frac{\partial e(n)}{\partial \underline{w}(n)}. \tag{12}$$

Vimos que

$$e(n) = d(n) - y(n) = d(n) - \underline{x}^T(n)\underline{w}(n) \tag{13}$$

e, como $d(n)$ não depende de $\underline{w}(n)$, temos que

$$\frac{\partial e(n)}{\partial \underline{w}(n)} = -\underline{x}(n). \tag{14}$$

De (14) e (12) temos

$$\underline{\nabla}\,\mathrm{J}(\underline{w}(n)) = -e(n)\underline{x}(n) \tag{15}$$

e, substituindo (15) em (9), encontraremos para $\underline{w}(n+1)$,

$$\underline{w}(n+1) = \underline{w}(n) + \eta\, e(n)\underline{x}(n), \tag{16}$$

onde η é o passo de adaptação ou razão de aprendizado.

A Equação (16) define o processo de ajuste do vetor de pesos $\underline{w}$ de um ALC objetivando minimizar J através do algoritmo LMS.

É instrutivo comparar os algoritmos SD e LMS utilizando a alegoria do "observador míope", cujo objetivo é atingir o mais rapidamente possível a coordenada $\underline{w}^*$, a qual define a coordenada da cota mínima da superfície $\mathrm{J}(\underline{w})$.

No algoritmo SD, o observador localizado na coordenada $\underline{w}(n)$ olha ao redor, localiza a direção $\underline{\nabla}\,\mathrm{J}(\underline{w}(n))$ de subida mais íngreme na superfície $\mathrm{J}(\underline{w})$ e dá um passo em direção contrária à ela, conforme já discutido. O ato de "olhar ao redor" significa matematicamente ter o conhecimento da

- matriz de correlação **R** do conjunto de vetores de entrada $\underline{x}$, e
- do vetor de correlação cruzada $\underline{p}$ entre o conjunto de saídas desejadas *d* e o conjunto de vetores $\underline{x}$.

O conhecimento destes elementos é necessário porque, no algoritmo SD, o gradiente no instante n é calculado através de $\underline{\nabla} \mathrm{J}(\underline{w}(n)) = -2\underline{p} + 2\mathbf{R}\underline{w}(n)$ (conforme S. Haykin em *Adaptive Filter Theory*, referenciado em [4]).

No algoritmo LMS, o observador não é somente míope como também é totalmente cego. O observador, localizado na coordenada $\underline{w}(n)$, consegue "observar" sua posição relativa porque segura em sua mão um cordão infinitamente elástico cuja outra extremidade encontra-se fixa na coordenada $\underline{w}^*$. A cada instante *n*, o observador dá um passo na direção em que ele percebe a maior redução na tensão τ do elástico (diminuição do valor absoluto do erro $e(n)$), com tamanho de passo proporcional à redução de τ. Como não existem mínimos locais na superfície $\mathrm{J}(\underline{w})$, porque ela é quadrática, o observador se aproximará da cota mínima $\mathrm{J}(\underline{w}^*)$ na coordenada $\underline{w}^*$ após repetir este procedimento um número suficiente de vezes. Note que, como o tamanho e sentido do passo do observador dependem da redução na tensão τ do elástico, quando o observador chegar próximo à coordenada $\underline{w}^*$ ele ficará eternamente "pulando" sobre e ao redor dela a menos que, por um raro golpe de sorte, a coordenada resultante do último passo do observador coincida com $\underline{w}^*$ (situação que ocorrerá para um valor bastante particular e crítico de η e para uma bastante particular coordenada inicial $\underline{w}^0$ da trajetória do observador). Apesar disto, o algoritmo LMS tem a vantagem de não necessitar do conhecimento de **R** e de $\underline{p}$, ao contrário do algoritmo SD.

Em suma, no algoritmo SD o vetor $\underline{w}(n)$ segue uma trajetória bem definida no espaço de pesos, para um valor não excessivo de η. Em contraste, no algoritmo LMS o vetor $\underline{w}(n)$ segue uma trajetória aleatória, especialmente nas vizinhanças de $\underline{w}^*$.

A Tabela 3.1 apresenta um sumário do procedimento do algoritmo LMS.

Conjunto de Treino:	Sinal de entrada em forma vetorial = $\underline{x}(n)$ Sinal resposta desejada escalar = $d(n)$
Parâmetro ajustável pelo usuário:	η
Inicialização do vetor $\underline{w}$:	$\underline{w}^0 = \underline{w}(0) = \underline{0}$
Procedimento Computacional:	Para $n = 0,1,\cdots$ computar $e(n) = d(n) - \underline{x}^T(n)\underline{w}(n)$ $\underline{w}(n+1) = \underline{w}(n) + \eta\, e(n)\underline{x}(n)$

Tabela 3.1: Sumário do algoritmo LMS. O Procedimento Computacional é executado até que a média de $e^2(n)$ atinja um patamar suficientemente baixo para a solução do problema em questão ou estabilize em um valor constante.

5.1 Considerações quanto à Convergência do LMS

Combinando (13) e (16) podemos expressar a evolução do vetor $\underline{w}$ através de

$$\begin{aligned}\underline{w}(n+1) &= \underline{w}(n) + \eta\underline{x}(n)[d(n) - \underline{x}^T(n)\underline{w}(n)] = \underline{w}(n) + \eta\underline{x}(n)d(n) - \eta\underline{x}(n)\underline{x}^T(n)\underline{w}(n) = \\ &= \left[\mathbf{I} - \eta\underline{x}(n)\underline{x}^T(n)\right]\underline{w}(n) + \eta\underline{x}(n)d(n),\end{aligned} \tag{17}$$

onde **I** é a matriz identidade.

O processo de ajuste do vetor $\underline{w}(n)$ é uma operação iterativa indexada pela variável inteira *n*. Em função disto, podemos então reconhecer que o valor de $\underline{w}(n+1)$ será o valor de $\underline{w}(n)$ quando a variável *n* for incrementada de 1 na próxima iteração. Em outras palavras, o valor obtido de (17) para $\underline{w}(n+1)$ no instante *n* é armazenado em uma posição de memória para ser utilizado como o valor de $\underline{w}(n)$ em (17) no instante $n+1$.

No domínio z, esta relação entre $\underline{w}(n)$ e $\underline{w}(n+1)$ é expressa por

$$\mathbf{Z}\{\underline{w}(n)\} = z^{-1}\mathbf{Z}\{\underline{w}(n+1)\} \tag{18}$$

onde $\mathbf{Z}\{\cdot\}$ é o operador Transformada Z e z^{-1} é o operador atraso unitário (*unit delay*). A partir das equações (17) e (18) podemos representar o algoritmo LMS através do grafo de fluxo de sinal mostrado na Figura 4.

A Figura 4 revela que o algoritmo LMS pode ser considerado como um sistema realimentado, já que existem dois *loops* de *feedback*, um superior e outro inferior. A presença de realimentação exerce um profundo impacto no comportamento do algoritmo LMS, visto que os parâmetros dos *loops* definem a estabilidade da trajetória dos estados de qualquer sistema realimentado.

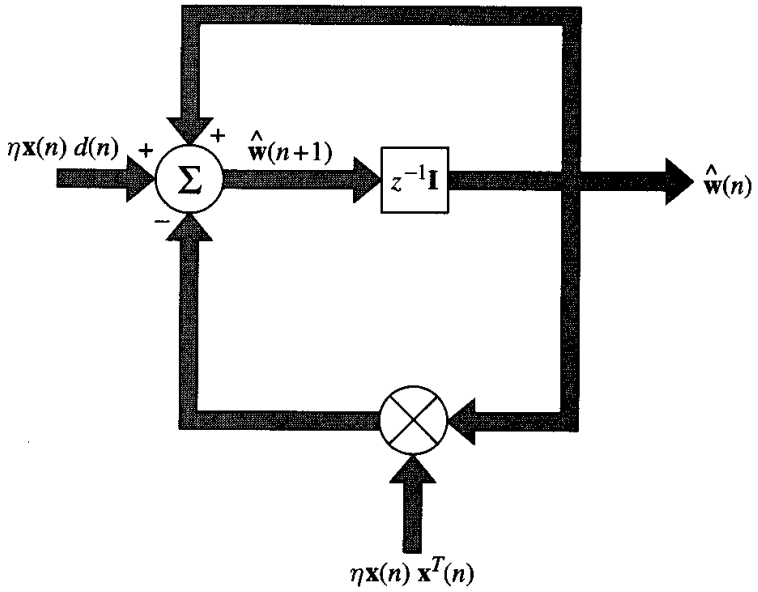

Figura 4: Grafo de fluxo de sinal representativo do algoritmo LMS.

Observe na Figura 4 que o *loop* inferior impõe variabilidade ao comportamento do LMS, particularmente porque a transmitância deste *loop* é controlada pela matriz $\eta\,\underline{x}(n)\underline{x}^T(n)$, a qual depende do vetor de entrada $\underline{x}(n)$, com parâmetro de controle dado pela razão de aprendizado η. Infere-se, portanto, que a estabilidade da trajetória de $\underline{w}(n)$ é

influenciada pelas características estatísticas do conjunto de vetores de entrada $\underline{x}$ e pelo valor da razão de aprendizado η.

Expressando este fato de outro modo, para um dado conjunto de vetores de entrada $\underline{x}$ deve-se escolher η tal que a trajetória de $\underline{w}(n)$ seja estável o suficiente para permitir a convergência para as vizinhanças de $\underline{w}^*$. A convergência da trajetória de $\underline{w}(n)$ para as vizinhanças de $\underline{w}^*$ é caracterizada por uma constância no valor médio de $e^2(n)$.

Como regra geral, a razão de aprendizado η deve obedecer à relação:

$$0 < \eta < \frac{2}{\frac{1}{N}\sum_{i=0}^{N-1} \underline{x}^T(i)\underline{x}(i)}, \tag{19}$$

onde *N* é o número total de vetores no conjunto de vetores de entrada $\underline{x}$.

6. Referências Bibliográficas

[1] B. Widrow e S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[2] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Massachusetts, 1995.

[3] R. D. Strum e D. E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing*, Addison-Wesley, 1989.

[4] S. Haykin, *Adaptive Filter Theory*, 3rd ed., Prentice Hall, Upper Saddle River, New Jersey, 1996.

[5] S. Haykin, *Neural Networks*, 2nd ed., Prentice Hall, Upper Saddle River, New Jersey, 1999.