Extreme Learning Machine (ELM)

Prof. Dr. Guilherme de Alencar Barreto

Junho /2010

Departmento de Engenharia de Teleinformática
Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC), Fortaleza-CE

guilherme@deti.ufc.br

1 Definições Preliminares

De início, vamos assumir que existe uma lei matemática $\mathbf{F}(\cdot)$, também chamada aqui de função ou mapeamento, que relaciona um vetor de entrada qualquer, $\mathbf{x} \in \mathbb{R}^{p+1}$, com um vetor de saída, $\mathbf{d} \in \mathbb{R}^m$. Esta relação, representada genericamente na Figura 1, pode ser descrita matematicamente da seguinte forma:

$$\mathbf{d} = \mathbf{F}[\mathbf{x}] \tag{1}$$

em que se assume que $\mathbf{F}(\cdot)$ é totalmente desconhecida, ou seja, não sabemos de antemão quais são as *fórmulas* usadas para associar um vetor de entrada $\mathbf{x}$ com seu vetor de saída $\mathbf{d}$ correspondente.

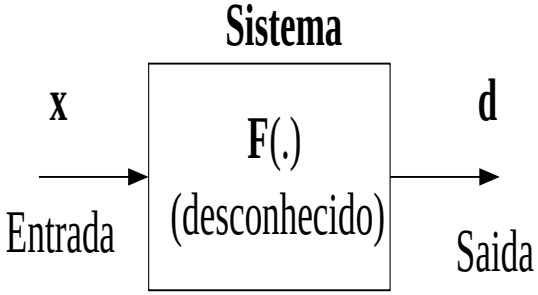

Figura 1: Representação simplificada de um mapeamento entrada-saída genérico.

O mapeamento $\mathbf{F}(\cdot)$ pode ser tão simples quanto um mapeamento linear, tal como

$$\mathbf{d} = \mathbf{M}\mathbf{x} \tag{2}$$

em que $\mathbf{M}$ é uma matriz de dimensão $(p+1) \times m$. Contudo, $\mathbf{F}(\cdot)$ pode ser bastante complexo, envolvendo relações não-lineares entre as variáveis de entrada e saída. É justamente o funcionamento da relação matemática $\mathbf{F}(\cdot)$ que se deseja *imitar* através do uso de algoritmos adaptativos, tais como as redes neurais.

Supondo que a única fonte de informação que nós temos a respeito de $\mathbf{F}(\cdot)$ é conjunto finito de N pares entrada-saída observados (ou medidos), ou seja:

$$\begin{aligned} \mathbf{x}_1, &\quad \mathbf{d}_1 \\ \mathbf{x}_2, &\quad \mathbf{d}_2 \\ \vdots &\quad \vdots \\ \mathbf{x}_N, &\quad \mathbf{d}_N \end{aligned} \tag{3}$$

Os pares entrada-saída mostrados acima podem ser representados de maneira simplificada como $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$, em que μ é um apenas indíce simbolizando o μ-ésimo par do conjunto de dados. Uma maneira de se adquirir conhecimento sobre $\mathbf{F}(\cdot)$ se dá exatamente através dos uso destes pares.

Para isto pode-se utilizar uma rede neural qualquer para implementar um mapeamento entrada-saída aproximado, representado como $\hat{\mathbf{F}}(\cdot)$, tal que:

$$\mathbf{y}_\mu = \hat{\mathbf{F}}[\mathbf{x}_\mu] \tag{4}$$

em que $\mathbf{y}_\mu$ é a saída gerada pela rede neural em resposta ao vetor de entrada $\mathbf{x}_\mu$. Esta saída, espera-se, seja muito próxima da saída real $\mathbf{d}_\mu$. Dá-se o nome de *Aprendizado Indutivo* ao processo de obtenção da relação matemática geral $\hat{\mathbf{F}}(\cdot)$ a partir de apenas alguns pares $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$ disponíveis.

A seguir será mostrado uma arquitetura de redes neurais recentemente proposta com o propósito de obter uma representação aproximada de um mapeamento entrada-saída genérico.

2 Máquina de Aprendizado Extremo

Estamos considerando nas definições e cálculos a seguir uma arquitetura de rede neural do tipo *feedforward* (i.e. sem realimentação) com apenas uma camada de neurônios ocultos, conhecida como Máquina de Aprendizado Extremo (*Extreme Learning Machine*, ELM) [1]. Esta arquitetura de rede neural é semelhante à rede MLP, porém apresenta uma fase de aprendizado infinitamente mais rápida que a da rede MLP. Começaremos a seguir com a descrição de sua arquitetura, para em seguida escrever sobre o funcionamento e o treinamento da rede ELM.

Os neurônios da camada oculta (primeira camada de pesos sinápticos) são representados conforme mostrado na Figura 2a, enquanto os neurônios da camada de saída segunda camada de pesos sinápticos) são representados conforme mostrado na Figura 2b.

O vetor de pesos associado a cada neurônio i da camada escondida, também chamada de *camada oculta* ou *camada intermediária*, é representado como

$$\mathbf{w}_i = \begin{pmatrix} w_{i0} \\ \vdots \\ w_{ip} \end{pmatrix} = \begin{pmatrix} \theta_i \\ \vdots \\ w_{ip} \end{pmatrix} \tag{5}$$

em que θ_i é o limiar (*bias* ou *threshold*) associado ao neurônio i. Os neurônios desta camada são chamados de neurônios escondidos por não terem acesso direto à sáida da rede, onde são calculados os erros de aproximação.

De modo semelhante, o vetor de pesos associado a cada neurônio k da camada de saída é representado como

$$\mathbf{m}_k = \begin{pmatrix} m_{k0} \\ \vdots \\ m_{kq} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \vdots \\ m_{kq} \end{pmatrix} \tag{6}$$

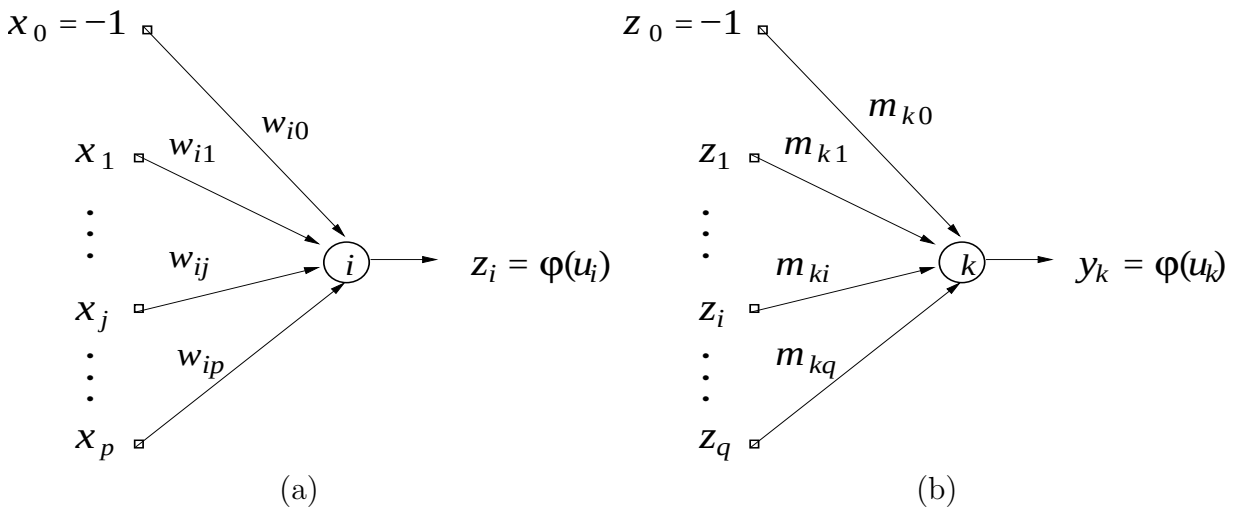

Figura 2: (a) Neurônio da camada escondida. (b) Neurônio da camada de saída.

em que θ_k é o limiar associado ao neurônio de saída k.

O treinamento da rede ELM se dá em duas etapas, que são descritas a seguir.

3 Fase 1: Inicialização Aleatória dos Pesos dos Neurônios Ocultos

Esta etapa de funcionamento da rede ELM envolve o cálculo das ativações e saídas de todos os neurônios da camada escondida e de todos os neurônios da camada de saída, uma vez que os pesos w_{ij}, $i = 1, \ldots, q$ e $j = 0, \ldots, p$, tenham sido inicialidos com valores aleatórios. Formalmente, podemos escrever:

$$w_{ij} \sim U(a, b) \qquad \text{ou} \qquad w_{ij} \sim N(0, \sigma^2) \tag{7}$$

em que $U(a, b)$ é um número (pseudo-)aleatório uniformemente distribuído no intervalo (a, b), enquanto $N(0, \sigma^2)$ é um número (pseudo-)aleatório normalmente distribuído com média zero e variância σ^2.

Em ambientes de programação tais Matlab© e Octave, esta fase é facilmente implementada em uma linha apenas de código. Para isso, precisamos definir uma matriz de pesos $\mathbf{W}$, com q linhas e $p + 1$ colunas:

$$\mathbf{W} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix}_{q \times (p+1)} = \begin{pmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_q^T \end{pmatrix} \tag{8}$$

em que notamos que i-ésima linha da matrix $\mathbf{W}$ é composta pelo vetor de pesos do i-ésimo neurônio oculto.

Uma vez definida a matriz $\mathbf{W}$, podemos realizar a etapa 1 através das seguintes linhas de código Octave, caso os pesos sejam inicializados com números aleatórios uniformes:

```
>> a=0; b=0.1;     % define intervalo dos pesos
>> W=a+(b-a).*rand(q,p+1);     % gera numeros uniformes
```

ou pelas seguintes linhas se preferirmos números aleatórios gaussianos:

```
>> sig=0.1;     % define desvio-padrao dos pesos
>> W=sig*randn(q,p+1);     % gera numeros gaussianos
```

4 Fase 2: Acúmulo das Saídas dos Neurônios Ocultos

Esta fase do treinamento da rede

O fluxo de sinais (informação) se dá dos neurônios de entrada para os neurônios de saída, passando obviamente pelos neurônios da camada escondida. Por isso, diz-se que o informação está fluindo no sentido **direto** (*forward*), ou seja:

Entrada → Camada Intermediária → Camada de Saída

Assim, após a apresentação de um vetor de entrada $\mathbf{x}$, na iteração t, o primeiro passo é calcular as ativações dos neurônios da camada escondida:

$$u_i(t) = \sum_{j=0}^{p} w_{ij} x_j(t) = \mathbf{w}_i^T \mathbf{x}(t), \quad i = 1, \ldots, q \tag{9}$$

em que T indica o vetor (ou matriz) transposto e q indica o número de neurônios da camada escondida.

A operação seqüencial da Eq. (9) pode ser feita de uma única vez se utilizarmos a notação vetor-matriz. Esta notação é particularmente útil em ambinetes do tipo Matlab/Octave. Neste caso, temos que o vetor de ativações $\mathbf{u}_i(t) \in \mathbb{R}^q$ do i-ésimo neurônio oculto na iteração t é calculado como

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t). \tag{10}$$

Em seguida, as saídas correspondentes são calculadas como

$$z_i(t) = \phi_i(u_i(t)) = \phi_i\left(\sum_{j=0}^{p} w_{ij}(t) x_j(t)\right) = \phi_i\left(\mathbf{w}_i^T(t)\mathbf{x}(t)\right) \tag{11}$$

tal que a função de ativação ϕ assume geralmente uma das seguintes formas:

$$\phi_i(u_i(t)) = \frac{1}{1+\exp[-u_i(t)]}, \qquad \text{(Logística)} \tag{12}$$

$$\phi_i(u_i(t)) = \frac{1-\exp[-u_i(t)]}{1+\exp[-u_i(t)]}, \qquad \text{(Tangente Hiperbólica)} \tag{13}$$

Em notação matriz-vetor, a Eq. (11) pode ser escrita como

$$\mathbf{z}(t) = \phi_i(\mathbf{u}_i(t)) = \phi_i(\mathbf{W}\mathbf{x}(t)). \tag{14}$$

em que a função de ativação $\phi_i(\cdot)$ é aplicada a cada um dos q componente do vetor $\mathbf{u}(t)$.

Para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor $\mathbf{z}(t)$ correspondente, que deve ser organizado (disposto) como uma coluna de uma matriz $\mathbf{Z}$. Esta matriz terá q linhas por N colunas:

$$\mathbf{Z} = [\mathbf{z}(1) \mid \mathbf{z}(2) \mid \cdots \mid \mathbf{z}(N)]. \tag{15}$$

A matriz $\mathbf{Z}$ será usada na Fase 3 para calcular os valores do pesos dos neurônios de saída da rede ELM.

5 Fase 3: Cálculo dos Pesos dos Neurônios de Saída

Sabemos que para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor de saídas desejadas $\mathbf{d}(t)$ correspondente. Se organizamos estes N vetores ao longo das colunas de uma matriz $\mathbf{D}$, então temos que esta matriz terá dimensão m linhas e N colunas:

$$\mathbf{D} = [\mathbf{d}(1) \mid \mathbf{d}(2) \mid \cdots \mid \mathbf{d}(N)]. \tag{16}$$

Podemos entender o cálculo dos pesos da camada de saída como o cálculo dos parâmetros de um mapeamento linear entre a camada oculta e a camada de saída. O papel de vetor de "entrada" para a camada de saída na iteração t é desempenhado pelo vetor $\mathbf{z}(t)$ enquanto o vetor de "saída" é representado pelo vetor $\mathbf{d}(t)$. Assim, buscamos determinar a matriz $\mathbf{M}$ que melhor represente a transformação

$$\mathbf{d}(t) = \mathbf{M}\mathbf{z}(t). \tag{17}$$

Para isso, podemos usar o método dos mínimos quadrados, também conhecido como método da pseudoinversa, já discutido nas notas de aula sobre o classificador OLAM. Assim, usando as matrizes $\mathbf{Z}$ e $\mathbf{D}$, a matriz de pesos $\mathbf{M}$ é calculada por meio das seguinte expressão:

$$\mathbf{M} = \mathbf{D}\mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T\right)^{-1}. \tag{18}$$

Alguns comentários sobre a matriz $\mathbf{M}$ fazem-se necessários:

- Note que para satisfazer a Eq. (17) a matriz $\mathbf{M}$ tem dimensão $m \times q$.
- A k-ésima linha da matriz $\mathbf{M}$, denotado aqui por $\mathbf{m}_k$, $k = 1, 2, \ldots, m$, corresponde ao vetor de pesos do k-ésimo neurônio de saída.

6 Teste e Capacidade de Generalização da Rede ELM

Uma vez determinadas as matrizes de pesos $\mathbf{W}$ e $\mathbf{M}$ temos a rede ELM pronta para uso. Durante o uso (teste) da rede ELM, calculamos as ativações dos neurônios da camada de saída por meio da seguinte expressão:

$$a_k(t) = \sum_{i=0}^{q} m_{ki}(t) z_i(t) = \mathbf{m}_k^T \mathbf{z}(t), \quad k = 1, \ldots, m \tag{19}$$

em que m é o número de neurônios de saída. Note que as saídas dos neurônios da camada oculta, $z_i(t)$, fazem o papel de entrada para os neurônios da camada de saída.

Em notação vetor-matriz, as operações da Eq. (19) podem ser executados de uma só vez por meio da seguinte expressão:

$$\mathbf{a}(t) = \mathbf{M}\mathbf{z}(t). \tag{20}$$

Para a rede ELM, assumimos que os neurônios de saída usam a função identidade como função de ativação, ou seja, as saídas destes neurônios são iguais às suas ativações: calculadas como:

$$y_k(t) = \phi_k(a_k(t)) = a_k(t). \tag{21}$$

Por generalização adequada entende-se a habilidade da rede em utilizar o conhecimento armazenado nos seus pesos e limiares para gerar saídas coerentes para novos vetores de entrada, ou seja, vetores que não foram utilizados durante o treinamento. A generalização é considerada

boa quando a rede, durante o treinamento, foi capaz de capturar (aprender) adequadamente a relação entrada-saída do mapeamento de interesse.

O bom treinamento de uma rede ELM, de modo que a mesma seja capaz de lidar com novos vetores de entrada, depende de uma série de fatores, dentre os quais podemos listar os seguintes

1. Excesso de graus de liberdade de uma rede ELM, na forma de elevado número de parâmetros ajustáveis (pesos e limiares).

2. Excesso de parâmetros de treinamento, tais como taxa de aprendizagem, fator de momento, número de camadas ocultas, critério de parada, dimensão da entrada, dimensão da saída, método de treinamento, separação dos conjuntos de treinamento e teste na proporção adequada, critério de validação, dentre outros.

Em particular, no que tange ao número de parâmetros ajustáveis, uma das principais conseqüências de um treinamento inadequado é a ocorrência de um subdimensionamento ou sobredimensionamento da rede ELM, o que pode levar, respectivamente, à ocorrência de *underfitting* (subajustamento) ou *overfitting* (sobreajustamento) da rede aos dados de treinamento. Em ambos os casos, a capacidade de generalização é ruim.

Dito de maneira simples, o subajuste da rede aos dados ocorre quando a rede não tem poder computacional (i.e. neurônios na camada oculta) suficiente para aprender o mapeamento de interesse. No ouro extremo está o sobreajuste, que ocorre quando a rede tem neurônios ocultos demais (dispostos em uma ou duas camadas ocultas) e passar a memorizar os dados de treinamento. O ajuste ideal é obtido para um número de camadas ocultas e neurônios nestas camadas que confere à rede um bom desempenho durante a fase de teste, quando sua generalização é avaliada.

7 Dicas para um Bom Desempenho da Rede ELM

O projeto de uma rede neural envolve a especificação de diversos itens, cujos valores influenciam consideravelmente funcionamento do algoritmo. A seguir especificaremos a lista destes itens juntamente com as faixas de valores que os mesmos podem assumir:

Dimensão do vetor de Entrada (p): Este item pode assumir em tese valores entre 1 e ∞. Porém, existe um limite superior que depende da aplicação de interesse e do custo de se medir (observar) as variáveis x_j. É importante ter em mente que um valor alto para p não indica necessariamente um melhor desempenho para a rede neural, pois pode haver redundância no processo de medição. Neste caso, uma certa medida é, na verdade, a combinação linear de outras medidas, podendo ser descartada sem prejuízo ao desempenho da rede. Quando é muito caro, ou até impossível, medir um elevado número de variáveis x_j, deve-se escolher aquelas que o especialista da área considera como mais relevante ou representativas para o problema. O ideal seria que cada variável x_j, $j = 1, \ldots, p$, "carregasse" informação que somente ela contivesse. Do ponto de vista estatístico, isto equivale a dizer que as variáveis são *independentes* ou *não-correlacionadas* entre si.

Dimensão do vetor de saída (M): Assim como o primeiro item, este também depende da aplicação. Se o interesse está em problemas de aproximação de funções, $\mathbf{y} = F(\mathbf{x})$, o número de neurônios deve refletir diretamente a quantidades de funções de saída desejadas (ou seja, a dimensão de $\mathbf{y}$).

Se o interesse está em problemas de classificação de padrões, a coisa muda um pouco de figura. Neste caso, o número de neurônios deve codificar o número de classes desejadas.

É importante perceber que estamos chamando as classes às quais pertencem os vetores de dados de uma forma bastante genérica: classe 1, classe 2, ..., etc. Contudo, à cada classe pode estar associado um rótulo (e.g. classe dos empregados, classe dos desempregados, classe dos trabalhadores informais, etc.), cujo significado depende da interpretação que o especialista na aplicação dá a cada uma delas. Estes rótulos normalmente não estão na forma numérica, de modo que para serem utilizados para treinar a rede ELM eles devem ser convertidos apra a forma numérica. A este procedimento dá-se o nome de codificação da saída da rede.

A codificação mais comum define como vetor de saídas desejadas um vetor binário de comprimento unitário; ou seja, apenas uma componente deste vetor terá o valor "1", enquanto as outras terão o valor "0" (ou -1). A dimensão do vetor de saídas desejadas corresponde ao número de classes do problema em questão. Usando esta codificação define-se automaticamente um neurônio de saída para cada classe. Por exemplo, se existem três classes possíveis, existirão três neurônios de saída, cada um representando uma classe. Como um vetor de entrada não pode pertencer a mais de uma classe ao mesmo tempo, o vetor de saídas desejadas terá valor 1 (um) na componente correspondente à classe deste vetor, e 0 (ou -1) para as outras componentes. Por exemplo, se o vetor de entrada $\mathbf{x}(t)$ pertence à classe 1, então seu vetor de saídas desejadas é $\mathbf{d}(t) = [1 \;\; 0 \;\; 0]^T$. Se o vetor $\mathbf{x}(t)$ pertence à classe 2, então seu vetor de saídas desejadas é $\mathbf{d}(t) = [0 \;\; 1 \;\; 0]^T$ e assim por diante para cada exemplo de treinamento.

Número de neurônios na camada escondida (q): Encontrar o número ideal de neurônios da camada escondida não é uma tarefa fácil porque depende de uma série de fatores, muito dos quais não temos controle total. Entre os fatores mais importantes podemos destacar os seguintes:

1. Quantidade de dados disponíveis para treinar e testar a rede.
2. Qualidade dos dados disponíveis (ruidosos, com elementos faltantes, etc.)
3. Número de parâmetros ajustáveis (pesos e limiares) da rede.
4. Nível de complexidade do problema (não-linear, discontínuo, etc.).

O valor de q é geralmente encontrado por tentativa-e-erro, em função da capacidade de *generalização* da rede (ver definição logo abaixo). Grosso modo, esta propriedade avalia o desempenho da rede neural ante situações não-previstas, ou seja, que resposta ela dá quando novos dados de entrada forem apresentados. Se muitos neurônios existirem na camada escondida, o desempenho será muito bom para os dados de treinamento, mas tende a ser ruim para os novos dados. Se existirem poucos neurônios, o desempenho será ruim também para os dados de treinamento. O valor ideal é aquele que permite atingir as especificações de desempenho adequadas tanto para os dados de treinamento, quanto para os novos dados.

Existem algumas fórmulas heurísticas (*ad hoc*) que sugerem valores para o número de neurônios na camada escondida da rede ELM, porém estas regras devem ser usadas apenas para dar um valor inicial para q. O projetista deve sempre treinar e testar várias vezes uma dada rede ELM para diferentes valores de q, a fim de se certificar que a rede neural generaliza bem para dados novos, ou seja, não usados durante a fase de treinamento.

Dentre a regras heurísticas citamos a seguir três, que são comumente encontradas na literatura especializada:

Regra do valor médio - De acordo com esta fórmula o número de neurônios da camada escondida é igual ao valor médio do número de entradas e o número de saídas da rede, ou seja:

$$q = \frac{p + M}{2} \tag{22}$$

Regra da raiz quadrada - De acordo com esta fórmula o número de neurônios da camada escondida é igual a raiz quadrada do produto do número de entradas pelo número de saídas da rede, ou seja:

$$q = \sqrt{p \cdot M} \tag{23}$$

Regra de Kolmogorov De acordo com esta fórmula o número de neurônios da camada escondida é igual a duas vezes o número de entradas da rede adicionado de 1, ou seja:

$$q = 2p + 1 \tag{24}$$

Perceba que as regras só levam em consideração características da rede em si, como número de entradas e número de saídas, desprezando informações úteis, tais como número de dados disponíveis para treinar/testar a rede e o erro de generalização máximo aceitável.

Uma regra que define um valor inferior para q levando em consideração o número de dados de treinamento/teste é dada por:

$$q \geq \frac{N - 1}{p + 2} \tag{25}$$

A regra geral que se deve sempre ter em mente é a seguinte: *devemos sempre ter muito mais dados que parâmetros ajustáveis*. Assim, se o número total de parâmetros (pesos + limiares) da rede é dado por $Z = (p + 1) \cdot q + (q + 1) \cdot M$, então devemos sempre tentar obedecer à seguinte relação:

$$N \gg Z \tag{26}$$

Um refinamento da Equação (26), proposto por Baum & Haussler (1991), sugere que a relação entre o número total de parâmetros da rede (Z) e a quantidade de dados disponíveis (N) deve obedecer à seguinte relação:

$$N > \frac{Z}{\varepsilon} \tag{27}$$

em que $\varepsilon > 0$ é o erro percentual máximo aceitável durante o teste da rede; ou seja, se o erro aceitável é 10%, então $\varepsilon = 0{,}1$. Para o desenvolvimento desta equação, os autores assumem que o erro percentual durante o treinamento não deverá ser maior que $\varepsilon/2$.

Para exemplificar, assumindo que $\varepsilon = 0{,}1$, então temos que $N > 10Z$. Isto significa que para uma rede de Z parâmetros ajustáveis, devemos ter uma quantidade dez vezes maior de padrões de treinamento.

Note que se substituirmos Z na Equação (27) e isolarmos para q, chegaremos à seguinte expressão que fornece o valor aproximado do número de neurônios na camada oculta:

$$q \approx \left\lceil \frac{\varepsilon N - M}{p + M + 1} \right\rceil \tag{28}$$

em que $\lceil u \rceil$ denota o menor inteiro maior que u.

A Equação (28) é bastante completa, visto que leva em consideração não só aspectos estruturais da rede ELM (número de entradas e de saídas), mas também o erro máximo tolerado para teste e o número de dados disponíveis. Portanto, seu uso é bastante recomendado.

Funções de ativação (ϕ_i) e (ϕ_k): Em tese, cada neurônio pode ter a sua própria função de ativação, diferente de todos os outros neurônios. Contudo, para simplificar o projeto da rede é comum adotar a mesma para todos os neurônios. Em geral, escolhe-se a função logística ou a tangente hiperbólica para os neurônios da camada escondida. Aquela que for escolhida para estes neurônios será adotada também para os neurônios da camada de saída. Em algumas aplicações é comum adotar uma função de ativação linear para os neurônios da camada de saída, ou seja, $\phi_k(u_k(t)) = C_k \cdot u_k(t)$, onde C_k é uma constante (ganho) positiva. Neste caso, tem-se que $\phi_k'(u_k(t)) = C_k$. O fato de $\phi_k(u_k(t))$ ser linear não altera o poder computacional da rede, o que devemos lembrar sempre é que os neurônios da camada escondida devem ter uma função de ativação não-linear, obrigatoriamente.

Avaliação de Desempenho: O desempenhado da rede ELM é, em geral, avaliada com base nos valores do erro quadrático médio (ε_{teste}) por padrão de teste:

$$\varepsilon_{teste} = \frac{1}{N}\sum_{t=1}^{N}\varepsilon(t) = \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{n}e_k^2(t) \tag{29}$$

em que $e_k(t) = d_k(t) - y_k(t)$ é o erro do k-ésimo neurônio de saída na iteração t.

Por outro lado, quando se utiliza a rede para classificar padrões, o desempenho da mesma é avaliado pela *taxa de acerto na classificação*, definida como:

$$P_{acerto} = \frac{\text{Número de vetores classificados corretamente}}{\text{Número de total de vetores}} \tag{30}$$

Outras métricas de avaliação do desempenho da rede ELM em tarefas de reconhecimento de padrões são a matriz de confusão e os valores de sensibilidade e especificidade para o caso de problemas de classificação binária.

Para validar a rede treinada, ou seja, dizer que ela está apta para ser utilizada, é importante testar a sua resposta (saída) para dados de entrada diferentes daqueles vistos durante o treinamento. Estes novos dados podem ser obtidos através de novas medições, o que nem sempre é viável. Durante o teste os pesos de saída da rede, em geral, não são ajustados.

Para contornar este obstáculo, o procedimento mais comum consiste em treinar a rede apenas com uma parte dos dados selecionados *aleatoriamente*, guardando a parte restante para ser usada para testar o desempenho da rede. Assim, ter-se-á dois conjuntos de dados, um para treinamento, de tamanho $N_1 < N$, e outro de tamanho $N_2 = N - N_1$. Em geral, escolhe-se N_1 tal que a razão N_1/N esteja na faixa de $0,75$ a $0,90$.

Em outras palavras, se $N_1/N \approx 0,75$ tem-se que 75% dos vetores de dados devem ser selecionados aleatoriamente, sem reposição, para serem utilizados durante o treinamento. Os 25% restantes serão usados para testar a rede. O valor de ε_{teste} calculado com os dados de teste é chamado de *erro de generalização* da rede, pois testa a capacidade da mesma em "extrapolar" o conhecimento aprendido durante o treinamento para novas situações. É importante ressaltar que, geralmente, o erro de generalização é maior do que o erro de treinamento, pois trata-se de um novo conjunto de dados.

8 Dicas para um Bom Projeto da Rede ELM

A seguir são dadas algumas sugestões para aumentar a chance de ser bem-sucedido no projeto de uma rede neural artificial.

Pré-processamento dos pares entrada-saída Antes de apresentar os exemplos de treinamento para a rede ELM é comum mudar a escala original das componentes dos vetores $\mathbf{x}$ e $\mathbf{d}$ para a escala das funções de ativação logística (0 e 1) ou da tangente hiperbólica (-1 e 1). As duas maneiras mais comuns de se fazer esta mudança de escala são apresentadas a seguir:

Procedimento 1: Indicado para quando as componentes x_j do vetor de entrada só assumem valores positivos e a função de ativação, $\phi(u)$, é a função logística. Neste caso, aplicar a seguinte transformação a cada componente de $\mathbf{x}$:

$$x_j^* = \frac{x_j}{x_j^{max}} \tag{31}$$

em que, ao dividir cada x_j pelo seu maior valor $x_j^{max} = \max_{\forall t}\{x_j(t)\}$, tem-se que $x_j^* \in [0, 1]$.

Procedimento 2: Indicado para quando as componentes x_j do vetor de entrada assumem valores positivos e negativos, e a função de ativação, $\phi(u)$, é a função tangente hiperbólica. Neste caso, aplicar a seguinte transformação a cada componente de $\mathbf{x}$:

$$x_j^* = 2\left(\frac{x_j - x_j^{min}}{x_j^{max} - x_j^{min}}\right) - 1 \tag{32}$$

em que $x_j^{min} = \min_{\forall t}\{x_j(t)\}$ é o menor valor de x_j. Neste caso, tem-se que $x_j^* \in [-1, +1]$.

Os dois procedimentos descritos acima também devem ser igualmente aplicados às componentes d_k dos vetores de saída, $\mathbf{d}$, caso estes possuam amplitudes fora da faixa definida pelas funções de ativação.

Função Tangente Hiperbólica: Tem sido demonstrado empiricamente, ou seja, através de simulação computacional que o processo de treinamento converge mais rápido quando se utiliza a função de ativação tangente hiperbólica do que quando se usa a função logística. A justificativa para isto está no fato da tangente hiperbólica ser uma função ímpar, ou seja, $\phi(-u_i) = -\phi(u_i)$. Daí sugere-se utilizar a função tangente hiperbólica sempre que o problema permitir.

Classificação de padrões: Quando se treina a rede ELM para classificar padrões é comum usar a codificação de saída descrita na Seção 7, em que na especificação do vetor de saídas desejadas assume-se o valor de saída unitário (1) para o neurônio que representa a classe e nulo (0) para os outros neurônios. Conforme dito no item anterior este valores são assintóticos e portanto, dificilmente serão observados durante a fase de teste.

Assim para evitar ambigüidades durante o cálculo da taxa de acerto P_{acerto} durante as fases de treinamento e teste define-se como a classe do vetor de entrada atual, $\mathbf{x}(t)$, como sendo a classe representada pelo neurônio que tiver maior valor de saída. Em palavras, podemos afirmar que se o índice do neurônio de maior saída é c, ou seja

$$y_c(t) = \max_{\forall k}\{y_k(t)\} \tag{33}$$

então a Classe de $\mathbf{x}(t)$ é a Classe c.

9 Exercícios Computacionais

Exercício 1 - Usar os pares de vetores entrada-saída disponíveis para um problema específico para avaliar a rede ELM em um problema de classificação de padrões. Pede-se determinar as taxas de acerto médio, máximo e mínimo na classificação, além do desvio-padrão da taxa de acerto para 50 rodadas treinamento/teste.

Exercício 2 - Usar a rede ELM para aproximar a função

$$y(x_1, x_2) = \sin^2(x_1) \cdot \cos^2(x_2) + x_1 x_2^3 \tag{34}$$

através da geração de 500 pares entrada-saída de treinamento. Pede-se determinar o gráfico da função $z(x, y)$ usando a fórmula mostrada na Equação (34) e através da rede ELM. Determinar o erro quadrático médio de generalização.

Referências

[1] G. B. Huang, Q. Y. Zhu, and C. K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.