COBENGE 2014
ENGENHARIA: Múltiplos saberes e atuações
16 a 19 de setembro | Juiz de Fora - MG

# SIMULAÇÃO DO MODELO CINEMÁTICO INVERSO DE UM ROBÔ ATRAVÉS DO USO DE REDES NEURAIS ARTIFICIAIS: UM COMPLEMENTO AO ENSINO DE ROBÓTICA

**José Tarcísio Franco de Camargo** – jtfc@bol.com.br
Faculdade Municipal "Professor Franco Montoro", FMPFM
Centro Regional Universitário de Espírito Santo do Pinhal, UNIPINHAL
Rua dos Estudantes s/n, Cachoeira de Cima
CEP 13840-000 – Mogi Guaçu - SP

**Estéfano Vizconde Veraszto** – estefanovv@cca.ufscar.br
Universidade Federal de São Carlos
Depto. de Ciências da Natureza, Matemática e Educação, UFSCar, CCA
Rodovia Anhanguera, Km 174
CEP 13604-900 – Araras - SP

**Gilmar Barreto** – gbarreto@dsif.fee.unicamp.br
Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação, UNICAMP, FEEC
Av. Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Barão Geraldo
CEP 13083-852 – Campinas - SP

***Resumo:*** *O ensino de robótica passa, necessariamente, pelo estudo dos modelos cinemáticos dos robôs manipuladores. Por sua vez, a cinemática de um robô manipulador pode ser descrita através de seus modelos direto e inverso. O modelo cinemático inverso, através do qual obtém-se o estado das juntas em função da posição desejada para a ferramenta do robô, normalmente é descrito e ensinado de forma algébrica nas aulas de robótica. Contudo, a representação algébrica deste modelo é, frequentemente, de difícil obtenção. Assim, embora seja inquestionável a necessidade de determinação exata do modelo cinemático inverso de um robô, o uso de redes neurais artificiais (RNAs) na fase de projeto pode ser muito atraente, pois nos permite prever o comportamento do robô antes do desenvolvimento formal de seu modelo. Neste sentido, este trabalho apresenta uma forma relativamente rápida de se simular o modelo cinemático inverso de um robô, permitindo, assim, que o aluno possa ter uma visão global do modelo, vindo a identificar pontos que devem ser corrigidos ou que podem ser otimizados na estrutura de um robô.*

***Palavras-chave:*** *Ensino de engenharia, Redes neurais artificiais, Robótica*

## 1. INTRODUÇÃO

O movimento descrito por um robô manipulador pode ser representado através de seus modelos cinemáticos direto e inverso, conforme descrito em Craig (1989). A obtenção do modelo cinemático direto é relativamente simples, sendo este definido através de um conjunto

de transformações entre os sistemas de referência de suas juntas ou "graus de liberdade". Através deste modelo podemos determinar a posição da "ferramenta" na extremidade livre do robô conhecendo-se a posição de suas juntas.

O modelo cinemático inverso, por sua vez, permite-nos determinar o estado das juntas de um robô em função da posição desejada para a sua ferramenta. Dessa forma, tendo sido definida uma trajetória para a ferramenta, é possível determinar o conjunto de posições das juntas que permitirão ao robô descrever o deslocamento desejado.

A obtenção do modelo cinemático inverso, contudo, é muito mais complexa do que a obtenção do modelo cinemático direto, uma vez que envolve a solução de um sistema de equações não lineares que pode admitir mais de uma solução. Mesmo em casos relativamente simples, como para o "robô planar" com dois graus de liberdade descrito a seguir, a definição do modelo cinemático inverso não é trivial.

Dessa forma, ser capaz de prever o comportamento de um robô, de forma relativamente simples, antes do desenvolvimento formal de seu modelo cinemático inverso, pode-se tornar um fator crucial para o sucesso de um projeto. Através do uso das redes neurais artificiais (RNAs), é possível simular o comportamento de um robô, determinando com relativa precisão o estado das juntas do mesmo em função da posição desejada para a sua ferramenta, permitindo que falhas de projeto possam ser detectadas, bem como pontos passíveis de otimização possam ser identificados.

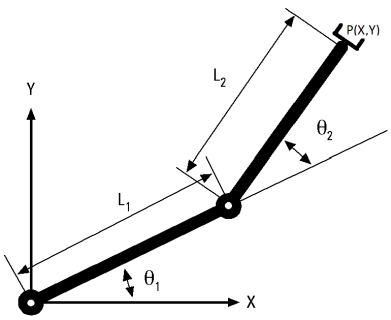
Assim, conforme exposto, este trabalho visa introduzir o tema das RNAs no contexto do ensino de robótica, objetivando contribuir para a formação de futuros profissionais que venham a se interessar pelo assunto e que sejam capazes de prever e simular situações nesta área, com vistas à verificação de falhas e oportunidades de otimização em projetos.

## 2. CINEMÁTICA DIRETA E INVERSA DO MOVIMENTO DE UM ROBÔ

O estudo dos modelos cinemáticos direto e inverso em robótica pode ser introduzido, de forma simples, através de um robô manipulador planar, conforme apresentado na Figura 1.

Na estrutura da Figura 1 temos um robô manipulador com dois graus de liberdade, estando sua extremidade inferior presa ao sistema de coordenadas referencial e seu extremo superior (ferramenta) livre para se deslocar sobre o plano (X,Y). Este robô é composto por dois braços de comprimento $L_1$ e $L_2$, sendo que a cada braço temos associado uma junta rotacional, cujos ângulos de rotação são $\theta_1$ e $\theta_2$, respectivamente. A extremidade livre deste robô encontra-se na posição P(X,Y) do plano.

Figura 1- Estrutura de um robô planar com dois graus de liberdade (Craig, 1989).



O modelo cinemático direto deste manipulador, que indica a posição da ferramenta do robô em função dos ângulos das juntas rotacionais, pode ser indicado, de forma bastante simples, através das equações (1) e (2).

$$X_P = L_1.\cos\theta_1 + L_2.\cos(\theta_1 + \theta_2) \quad (1)$$
$$Y_P = L_1.\sin\theta_1 + L_2.\sin(\theta_1 + \theta_2) \quad (2)$$

sendo "$X_P$" e "$Y_P$" as coordenadas do ponto P(X,Y).

Por sua vez, o modelo cinemático inverso deste robô, que indica os ângulos "$\theta_1$" e "$\theta_2$" das juntas em função da posição desejada para a ferramenta, pode ser descrito, segundo Craig (1989), através das equações (3) e (4).

$$\theta_2 = \pm\cos^{-1}\frac{X_P^2+Y_P^2-L_1^2-L_2^2}{2.L_1.L_2} \quad (3)$$

$$\theta_1 = \tan^{-1}\frac{Y_P.(L_1+L_2.\cos\theta_2)-X_P.L_2.\sin\theta_2}{X_P.(L_1+L_2.\cos\theta_2)+Y_P.L_2.\sin\theta_2} \quad (4)$$

Neste modelo, o ângulo "$\theta_2$" pode assumir valores positivos ou negativos, dependendo da posição do "cotovelo" do robô ("para cima" ou "para baixo").

É notável, através da observação das equações (3) e (4), que o modelo cinemático inverso realmente possui uma complexidade considerável, mesmo para um robô que possui apenas dois graus de liberdade em um espaço planar.

Neste caso, uma rede neural pode ser construída de tal forma que suas entradas sejam alimentadas pela posição desejada para a ferramenta do robô, sendo indicado em suas saídas os respectivos valores de seus graus de liberdade (ângulos, no caso de juntas rotacionais). Por meio desta estratégia, a RNA pode vir a ser inicialmente treinada a partir de um conjunto de valores referenciais determinados através do modelo cinemático direto, sendo que, após a fase de treinamento, a rede estará apta a gerar as informações desejadas (graus de liberdade em função da posição desejada para a ferramenta).

Um modelo de rede neural apropriado para este estudo em robótica é o perceptron em múltiplas camadas (MLP – multilayer perceptron). Através de uma MLP, é possível estudar a cinemática inversa de um robô sem a determinação explícita de seu modelo, permitindo, assim, que determinados comportamentos possam vir a ser identificados pela rede neural. Dessa forma, o estudo em robótica aqui proposto encontra-se focado no uso de redes neurais MLP, como alternativa aos métodos convencionais para a determinação do modelo cinemático inverso de um robô.

## 3. REDES NEURAIS E ROBÓTICA

Uma (RNA) constitui um sistema, comumente abstrato, que pode ser utilizado em várias aplicações, tais como aquelas apresentadas em Burke (1995), Mighell (1988), Reategui (1994) e Yoda (1994). Para tanto, antes de seu uso, muitas vezes a RNA deve ser “treinada” através de padrões. Assim, em sua fase de treinamento, a rede é apresentada a diversos padrões de entrada e seus respectivos padrões de saída desejados, de forma a “aprender” a “lei de formação” que correlaciona cada padrão de entrada ao seu respectivo padrão de saída.

Após esta fase inicial, espera-se que, caso seja apresentado um padrão de entrada não pertencente ao conjunto de treinamento, a rede seja capaz de inferir um provável padrão de saída.

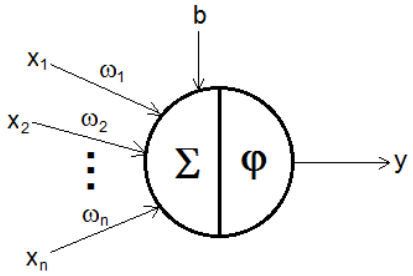
Um modelo de RNA bastante interessante que se encaixa dentro desta proposta é a rede formada por perceptrons em múltiplas camadas (ou Multilayer Perceptron ou MLP) (HAYKIN, 2001).

### 3.1. O perceptron

Uma RNA do tipo perceptrons em múltiplas camadas (MLP) é constituída pela organização em camadas do elemento computacional denominado "perceptron". Um perceptron, por sua vez, caracteriza-se por tentar expressar uma representação simbólica (e matemática) de um neurônio biológico. A Figura 2 apresenta o modelo simbólico de um perceptron, o qual é amplamente discutido em Lippmann (1987), Hush (1993) e Haykin (2001). Hush (1993), Haykin (2001) e Fahlman (1988) também apresentam detalhadamente o "algoritmo de retropropagação" (backpropagation), que será discutido mais adiante.

Neste modelo temos um "nó", que representa o corpo do neurônio, alimentado por diversos estímulos externos através de "conexões sinápticas". O resultado da computação dos estímulos externos pelo perceptron é exportado através de uma conexão sináptica de saída que encaminhará este sinal para a entrada de outros perceptrons (ou neurônios) da rede.

Figura 2 - Modelo simbólico de um perceptron (neurônio).




Os estímulos externos recebidos por um perceptron podem ser provenientes das saídas exportadas por outros perceptrons ou provenientes da entrada da rede neural ou, ainda, provenientes de um "sinal de polarização" ("bias") particular do próprio neurônio. Segundo (GERMAN, 1992), o sinal de polarização possui grande importância no controle de ruídos dos dados apresentados à rede. Todos os sinais que chegam a um neurônio (ou, neste caso, perceptron) são ponderados pelo "peso sináptico" da conexão que leva este sinal ao neurônio.

O equacionamento matemático que formaliza este modelo leva em consideração que os estímulos de entrada tem seus efeitos somados de forma ponderada. Ou seja:

$$v = \sum_{i=1}^{n} \omega_i \cdot x_i + b \qquad (5)$$

onde:

- "$v$" é a soma ponderada dos estímulos de entrada

- "$w_i.x_i$" é o produto entre um sinal de entrada "x" e o peso sináptico "$w$" da conexão que encaminha o sinal ao neurônio
- "$b$" é o sinal de polarização particular deste neurônio

A resposta do perceptron aos estímulos recebidos é função da soma ponderada destes. Ou seja:

$$y = \varphi(\nu) \tag{6}$$

onde:

- "y" é a resposta do neurônio
- φ($v$) é a "função de ativação" do neurônio, que correlaciona a saída aos estímulos de entrada.

A função de ativação "φ($v$)" do perceptron deve ser não-linear, de forma a permitir que a RNA como um todo possa representar funções não-lineares, além de ser contínua e "suave", de forma a ser diferenciável em todo o seu intervalo de consideração (para o cálculo do parâmetro "δ", que é utilizado no algoritmo de retropropagação e será exposto adiante).

Um exemplo de função de ativação comum em redes MLP é a "função logística":

$$\varphi(\nu) = \frac{1}{1+e^{-\alpha.\nu}} \tag{7}$$

onde "α" é uma constante que, segundo LeCun (1989), possui em 1,7159 um valor adequado.

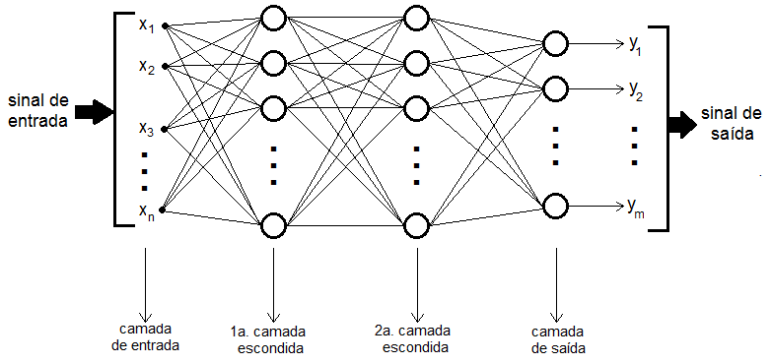
Neste caso, a derivada da função de ativação φ($v$) é dada por:

$$\varphi'(\nu) = \alpha.\varphi(\nu).(1 - \varphi(\nu)) \tag{8}$$

### 3.2. Redes neurais em camadas

Conforme mencionado, uma RNA do tipo MLP é composta pela associação em camadas de perceptrons. A Figura 3 apresenta um modelo para uma MLP.

Figura 3 - Modelo básico para uma MLP.




Nesta arquitetura temos uma "camada de entrada", onde são aplicados os estímulos (sinais) de entrada da rede, que são propagados camada por camada até a saída da rede, desde

a "1ª. camada escondida" de neurônios até a "camada de saída" da rede, que apresentará o resultado de todo o processamento da rede ao meio externo.

Também deve ser notado que, em uma MLP, as entradas de um determinado neurônio estão conectadas às saídas de todos os neurônios da camada imediatamente anterior e, a saída deste neurônio, está conectada a todas as entradas dos neurônios da camada imediatamente posterior.

Não existe uma regra de formação bem definida que indique quantas camadas uma rede deve possuir, ou mesmo quantos neurônios cada camada deve conter. Apenas é possível afirmar que uma rede com um maior número de camadas poderá assimilar melhor uma "lei de formação" que pretende-se ensinar (porém a um custo computacional possivelmente elevado). Por outro lado, redes com um número reduzido de camadas que contenham um número suficiente de neurônios podem ser capazes de obter um alto desempenho de aprendizado. Tais constatações, entretanto, são puramente empíricas.

### 3.3. O aprendizado de uma RNA - O algoritmo de retropropagação

Durante a fase de aprendizado, para que a rede possa aprender o comportamento desejado, pode ser utilizado um algoritmo recursivo como o "algoritmo de retropropagação" (ou backpropagation).

Neste algoritmo, durante a fase de treinamento da rede, um padrão de entrada "**x**" é apresentado à rede, sendo propagado em direção à saída, gerando nesta um padrão "**o**", o qual é comparado com o valor desejado "**d**" para o padrão de entrada em questão.

A diferença "**d-o**" entre o padrão desejado e o obtido de fato na saída da rede é o erro "**e**" desta fase de aprendizado para este padrão. Se **e=0**, então conclui-se que a rede "aprendeu" o padrão de entrada "**x**". Se **e≠0**, então o erro deve ser retropropagado (da camada de saída em direção à 1ª. camada escondida) de forma a ajustar os pesos das conexões entre os neurônios das camadas.

Formalmente, temos que:

sinal de erro: $ej = dj - oj$

onde:

- "dj" é um elemento do vetor de saída desejado para o padrão de entrada apresentado.
- "oj" é um elemento do vetor de saída obtido com a propagação do vetor de entrada.
- "ej" é um elemento do vetor de erro obtido.

De acordo com Andrews (1994), para minimizar as dificuldades de treinamento de uma RNA, a função de erro deve ser não linear. Assim, para o padrão de entrada apresentado, definimos a soma instantânea dos erros quadráticos na saída da rede por:

$$\varepsilon = \frac{1}{2} \cdot \sum_{j=1}^{m} e_j^2 \qquad (9)$$

onde:

- "m" é o número de elementos do vetor de erro (que equivale ao número de neurônios na camada de saída da rede).

Se considerarmos todos os padrões de entrada a serem apresentados na fase de treinamento, poderemos determinar o "erro médio quadrático" para estes padrões durante uma

"época" (dá-se o nome de "época" a cada propagação/retropropagação realizada para todos os padrões de entrada apresentados à rede em sua fase de treinamento). Assim:

$$\varepsilon_{QM} = \frac{1}{N} \cdot \sum_{n=1}^{N} \varepsilon(n) \tag{10}$$

onde:

- "N" é o número de padrões de treinamento apresentados à rede
- "$\varepsilon(n)$" é o erro obtido na apresentação do padrão "n"

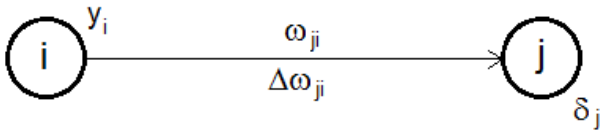
Portanto, tanto menor o valor do erro médio quadrático ($\varepsilon_{QM}$), tanto "melhor" foi o aprendizado dos padrões de entrada pela rede em uma determinada época. Dessa forma, se $\varepsilon_{QM} = 0$, a rede aprendeu com precisão absoluta todos os padrões apresentados.

Assim, uma proposta para treinamento de uma MLP pode ser:

- Enquanto $\varepsilon_{QM}$ não for suficientemente baixo:
  a) Para cada padrão de treinamento:
     i. propagar este padrão, em direção à saída, camada por camada
     ii. calcular o erro "e" entre a saída desejada e a saída obtida
     iii. Retropropagar o erro, da saída em direção à entrada, corrigindo os pesos sinápticos de cada conexão.
     iv. Fim
  b) Atualizar o valor de $\varepsilon_{QM}$
- Fim

No algoritmo mencionado acima, a correção do erro para ajuste dos pesos sinápticos pode ser realizada através da "regra delta". Este ajuste nos pesos sinápticos é fundamental para o aprendizado da RNA, uma vez que o conhecimento adquirido pela rede é condensado nestes. Considere a conexão existente entre dois neurônios de camadas consecutivas, conforme mostra a Figura 4.

Figura 4 - Interconexão entre dois neurônios e seus parâmetros.




Neste caso, a atualização do peso sináptico "$\omega_{ji}$" entre estes neurônios, de uma iteração para outra, será dada por:

$$\omega_{ji}(t+1) = \omega_{ji}(t) + \Delta\omega_{ji} \tag{11}$$

onde:

- "$\omega_{ji}(t)$" é o valor atual do peso sináptico
- "$\Delta\omega_{ji}$" é a correção do peso sináptico a ser aplicada
- "$\omega_{ji}(t+1)$" é o valor atualizado (próximo valor) do peso sináptico

O cálculo da correção do valor do peso sináptico ($\omega_{ji}$) deve vir na direção contrária ao gradiente do erro acumulado na propagação de uma época das amostras de entrada (conjunto

de treinamento). Assim, de acordo com a regra delta, a correção do peso sináptico pode ser dada por:

$$\Delta\omega_{ji} = \eta \cdot \delta_j \cdot y_i \tag{12}$$

onde:

- "η" é a "taxa de aprendizado" do neurônio (0 < η < 1), cujo valor, arbitrado pelo usuário da rede, define quão rapidamente a rede deverá convergir para o ponto de mínimo.
- ""δj" é o "gradiente local" do neurônio "j" para a correção do erro, calculado em função do erro retropropagado pela rede.
- "yi" é o valor da saída do neurônio "i".

O valor do gradiente local "δj" para o neurônio "j" pode ser calculado da seguinte forma:

$\delta_j = \varphi' \cdot e_j$, se o neurônio "j" pertence à camada de saída;

ou

$\delta_j = \varphi' \cdot \sum_k \delta_k \cdot \omega_{kj}$, Se o neurônio "j" pertence a uma camada escondida.

De forma a melhorar a convergência da regra delta, pode ser utilizado um "fator de momento" ("μ") que irá considerar o valor de $\Delta\omega_{ji}$ da iteração anterior para o cálculo do próximo valor:

$$\Delta\omega_{ji}(t) = \mu \cdot \Delta\omega_{ji}(t-1) + \eta \cdot \delta_j(t) \cdot y_i(t) \tag{13}$$

sendo o fator de momento "μ" arbitrado entre 0 e 1.

De forma simplificada, o processo de criação e treinamento de uma rede neural pode ser descrito na forma:

1. Criação da rede:
    a. Definir o tamanho do vetor de entrada da rede, ou seja, o número de nós da camada de entrada da rede.
    b. Definir o tamanho do vetor de saída da rede, ou seja, o número de neurônios da camada de saída da rede.
    c. Definir o número de camadas escondidas da rede.
    d. Definir o número de neurônios de cada camada escondida da rede.
    e. Atribuir valores aleatórios (entre -1 e 1) para os sinais de polarização de cada neurônio.
    f. Atribuir valores aleatórios (entre -1 e 1) para os pesos sinápticos de cada conexão da RNA.
    g. Definir a função de ativação dos neurônios e sua derivada.
2. Treinamento da rede:
    a. Definir os valores para "η" e "μ".
    b. Definir o valor limite para $\varepsilon_{MQ}$.
    c. Enquanto $\varepsilon_{MQ}$ não for suficientemente baixo:
        i. Para cada padrão de treinamento:
            1. Aplicar o padrão à entrada "x" da RNA.
            2. Calcular a saída "y" de cada neurônio da 1ª. camada escondida, aplicando estas saídas às entradas da camada

subsequente, calculando também as saídas desta camada até a camada de saída (propagação do sinal de entrada).

3. Calcular o erro "e" para este padrão, comparando o valor desejado para a saída ("d") e o valor obtido na saída ("o") a partir da entrada aplicada.
4. A partir da camada de saída, em direção à camada de entrada da rede (retropropagação do erro):

a. Determine o gradiente local ("δ") de cada neurônio de cada camada.

5. A partir da camada de saída, em direção à camada de entrada da rede:

a. Determine $\Delta\omega$ de cada conexão sináptica.
b. Atualize $\omega$ de cada conexão sináptica.

6. Fim.

ii. Atualize o valor de $\varepsilon_{MQ}$ para esta época.
iii. Fim.

Após o treinamento da rede, o uso desta é realizado simplesmente pela apresentação de um padrão de entrada qualquer, o qual é propagado em direção à saída da rede, onde poderá ser observada a resposta da rede ao estímulo de entrada apresentado.

## 4. IMPLEMENTAÇÃO SOBRE O CASO DO ROBÔ PLANAR

A implementação e a simulação de uma rede neural do tipo MLP pode ser realizada através do desenvolvimento de programas de computador em diversas linguagens. Estes autores realizaram a implementação de RNAs na linguagem de programação do "SciLab" (http://www.scilab.org).

De acordo com Cybenko (1989) e Hertz (1991), uma RNA com uma única camada intermediária é suficiente para aproximar qualquer função contínua. Por sua vez, (CYBENKO, 1988) afirma que qualquer função matemática pode ser modelada através de uma RNA com, no máximo, duas camadas intermediárias.

Para o modelo do robô planar apresentado neste artigo, todas as simulações foram realizadas com RNAs dotadas de duas camadas escondidas. Na simulação apresentada na Tabela 1 foi utilizada uma RNA com dois neurônios na camada de entrada (associados às coordenadas da posição desejada para a ferramenta: "$X_P$" e "$Y_P$"), duas camadas escondidas com seis neurônios cada uma e dois neurônios na camada de saída (associados aos ângulos "$\theta_1$" e "$\theta_2$" das juntas do robô). Nesta simulação foi utilizada uma taxa de aprendizado ($\eta$) igual a 0,1 e um fator de momento ($\mu$) igual a 0,3. Foi utilizada uma função de ativação do tipo "logística". A Tabela 1 apresenta os resultados para a simulação realizada.

Conforme pode ser notado através da Tabela 1, os resultados apresentados pela simulação encontram-se, em sua maioria, relativamente próximos dos valores de treinamento, o que pode justificar o uso deste método como ferramenta de análise do comportamento de um robô, previamente à concepção definitiva de seu modelo cinemático inverso através de métodos algébricos.

Tabela 1 - Resultados para a simulação do robô planar

| **Treinamento e simulação da cinemática inversa de um robô planar através de MLP** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Comprimento do braço L1 = 3 (unidades) Comprimento do braço L2 = 2 (unidades) | | | | | | | | | |
| **Padrões de treinamento** | | | | **Resultados da simulação** | | | | | |
| Entradas | | Saídas | | Entradas | | Saídas | | | |
| Xp (un.) | Yp (un.) | $\theta_1$ (graus) | $\theta_2$ (graus) | Xp (un.) | Yp (un.) | $\theta_1$ (graus) | $\theta_2$ (graus) | Dif. $\theta_1$ (graus) | Dif. $\theta_2$ (graus) |
| 5,0 | 0,0 | 0,0 | 0,0 | 5,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| 4,7 | 1,0 | 0,0 | 30,0 | 4,7 | 1,0 | 0,0 | 30,4 | 0,0 | 0,4 |
| 4,0 | 1,7 | 0,0 | 60,0 | 4,0 | 1,7 | 2,2 | 58,5 | 2,2 | 1,5 |
| 3,0 | 2,0 | 0,0 | 90,0 | 3,0 | 2,0 | 6,0 | 93,4 | 6,0 | 3,4 |
| 2,6 | 3,5 | 30,0 | 60,0 | 2,6 | 3,5 | 29,9 | 59,7 | 0,1 | 0,3 |
| 1,6 | 3,2 | 30,0 | 90,0 | 1,6 | 3,2 | 29,9 | 92,9 | 0,1 | 2,9 |
| 0,9 | 2,5 | 30,0 | 120,0 | 0,9 | 2,5 | 29,5 | 115,2 | 0,5 | 4,8 |
| 0,6 | 1,5 | 30,0 | 150,0 | 0,6 | 1,5 | 30,5 | 153,5 | 0,5 | 3,5 |
| -0,2 | 3,6 | 60,0 | 90,0 | -0,2 | 3,6 | 58,9 | 93,0 | 1,1 | 3,0 |
| -0,5 | 2,6 | 60,0 | 120,0 | -0,5 | 2,6 | 57,8 | 116,2 | 2,2 | 3,8 |
| -0,2 | 1,6 | 60,0 | 150,0 | -0,2 | 1,6 | 57,0 | 150,0 | 3,0 | 0,0 |
| 0,5 | 0,9 | 60,0 | 180,0 | 0,5 | 0,9 | 58,5 | 175,2 | 1,5 | 4,8 |
| -1,7 | 4,0 | 90,0 | 60,0 | -1,7 | 4,0 | 90,6 | 58,7 | 0,6 | 1,3 |
| -1,7 | 2,0 | 90,0 | 120,0 | -1,7 | 2,0 | 87,8 | 120,7 | 2,2 | 0,7 |
| -1,0 | 1,3 | 90,0 | 150,0 | -1,0 | 1,3 | 90,0 | 153,2 | 0,0 | 3,2 |
| -3,5 | 2,6 | 120,0 | 60,0 | -3,5 | 2,6 | 122,5 | 57,4 | 2,5 | 2,6 |
| -3,2 | 1,6 | 120,0 | 90,0 | -3,2 | 1,6 | 120,3 | 93,0 | 0,3 | 3,0 |
| -1,5 | 0,6 | 120,0 | 150,0 | -1,5 | 0,6 | 126,2 | 152,4 | 6,2 | 2,4 |
| -0,5 | 0,9 | 120,0 | 180,0 | -0,5 | 0,9 | 121,1 | 173,7 | 1,1 | 6,3 |
| -4,6 | 1,5 | 150,0 | 30,0 | -4,6 | 1,5 | 145,6 | 25,1 | 4,4 | 4,9 |
| -4,3 | 0,5 | 150,0 | 60,0 | -4,3 | 0,5 | 153,2 | 63,5 | 3,2 | 3,5 |
| -3,6 | -0,2 | 150,0 | 90,0 | -3,6 | -0,2 | 149,0 | 85,1 | 1,0 | 4,9 |
| -2,6 | -0,5 | 150,0 | 120,0 | -2,6 | -0,5 | 144,2 | 119,0 | 5,8 | 1,0 |
| -0,9 | 0,5 | 150,0 | 180,0 | -0,9 | 0,5 | 149,5 | 174,5 | 0,5 | 5,5 |
| -5,0 | 0,0 | 180,0 | 0,0 | -5,0 | 0,0 | 180,0 | 5,2 | 0,0 | 5,2 |
| -4,7 | -1,0 | 180,0 | 30,0 | -4,7 | -1,0 | 179,1 | 29,5 | 0,9 | 0,5 |
| -4,0 | -1,7 | 180,0 | 60,0 | -4,0 | -1,7 | 178,0 | 60,7 | 2,0 | 0,7 |
| -3,0 | -2,0 | 180,0 | 90,0 | -3,0 | -2,0 | 180,0 | 91,1 | 0,0 | 1,1 |
| -2,0 | -1,7 | 180,0 | 120,0 | -2,0 | -1,7 | 180,0 | 121,0 | 0,0 | 1,0 |
| -1,3 | -1,0 | 180,0 | 150,0 | -1,3 | -1,0 | 179,8 | 150,0 | 0,2 | 0,0 |

## 5. CONSIDERAÇÕES FINAIS

Através deste trabalho pode-se constatar que a aplicação das RNAs em robótica apresenta resultados satisfatórios para a simulação prévia do comportamento de robôs,

permitindo que falhas de projeto ou oportunidades de otimização possam vir a ser detectadas. O grande inconveniente no uso das RNAs em robótica encontra-se na dificuldade de se encontrar a topologia de rede mais adequada para cada caso a ser simulado. Utilizar um número grande de camadas não é recomendado, pois cada vez que o erro medido durante o treinamento é propagado para a camada anterior ele pode vir a se tornar mais elevado.

O número de nodos nas camadas intermediárias de uma RNA depende fortemente da distribuição dos padrões de treinamento e validação da rede. A utilização de muitas unidades pode facilitar a memorização dos padrões de treinamento. Contudo, isso limitará a capacidade de extrair as características gerais que permitem a generalização ou reconhecimento de padrões não vistos durante o treinamento (este problema é chamado de *overfitting*). Por outro lado, um número muito pequeno de nodos pode forçar a rede a gastar tempo em excesso tentando encontrar uma representação ótima. Se o número de exemplos for muito maior que o número de conexões entre nodos o *overfitting* é improvável, mas pode ocorrer *underfitting* (a rede não converge durante o seu treinamento).

Finalmente, a aplicabilidade desta proposta em sala de aula justifica-se pela necessidade de fornecer aos alunos dos cursos de engenharia novas ferramentas para o estudo de problemas em robótica. Neste sentido, o uso das RNAs procura instigar os alunos a pesquisarem novas e melhores ferramentas para a solução de problemas, considerando-se, assim, a indissociabilidade entre ensino e pesquisa. Especificamente, no caso destes autores, o método apresentado ao longo deste texto foi introduzido no decorrer da disciplina "Inteligência Artificial", do Curso de Ciência da Computação. Esta proposta foi apresentada procurando fomentar entre os alunos, além do uso propriamente deste método, a necessidade do constante desenvolvimento de pesquisas como forma de continuamente buscar o aperfeiçoamento de soluções em problemas de engenharia.

## REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS, R.; GEVA, S. Rule extraction from a constrained error backpropagation MLP. In: Proceedings of the 5th Australian Conference on Neural Networks, pp. 9-12, Brisbane, Austrália, 1994.

BURKE, H.; ROSEN, D.; GOODMAN, P. Comparing the prediction accuracy of artificial neural networks and other statistical models for breast cancer survival. In G. TESAURO, D. S. TOURETZKY, T. K. LEEN, editors, Neural Information Processing Systems 7. MIT Press, 1995.

GRAIG, J. J. Introduction to robotics: mechanics and control. 2nd ed. New York: Assison-Wesley Publishing Company, 1989.

CYBENKO, G. Continuos valued neural networks with two hidden layers are sufficient. Technicalreport, Departmentof Computer Science, Tufts University, 1988.

CYBENKO, G. Approximation by superpositions of a sigmoid function. Mathematic sof Control, Signals and Systems, 2:303-314, 1989.

FAHLMAN, S. E. An empirical study of learning speed in backpropagation networks. Technical report, carnegie Mellow University, 1988.

GERMAN, S.; BIENESTOCK, E.; DOURSAT, R. Neural networks and the bias-variance dilemma. Neural Computation, 4:1-58, 1992.

HAYKIN, S. 2001. Redes Neurais – Princípios e Prática, 2ª. Ed. Porto Alegre: Bookman, 2001. (ISBN: 978-85-7307-718-6)

HERTZ, J.; KROGH, A.; PALMER, R.G. Introduction to the Theory of Neural Computation, volume Lecture Notes. Vol. 1 of Santa Fe Institute Studies in The Science of Complexity. Addison-Wesley, 1991.

HUSH, D. R.; HORNE, B. G. Progress in Supervised Neural Networks – What's New Since Lippmann?, IEEE Signal Processing Magazine, January/1993, pp. 8-39.

LeCUN, Y. Generalization and network design strategies. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, Canada, 1989.

LIPPMANN, R. P. An Introduction to Computing with Neural Nets. IEEE ASSP Magazine, April/1987, pp. 4-22.

MIGHELL, D.A.; WIKINSON, T.S.; GOODMAN, J.W. Back propagations and its application to handwritten signature verification. In R.P. LIPPMANN, J.E. MODDY, D.S. TOURETZKY, editors, Advances in Neural Information Processing Systems 2. Morgan Kaufmann, 1988.

REATEGUI, E.; CAMPBELL, J.A. A classification system for credit card transactions. In Proceedings of the Second European Workshop on Case-Based Reasoning, pp.167-174, November 1994.

YODA, M. Predicting the Tokyo stock market. John Wiley & Sons, 1994.

# SIMULATION OF THE INVERSE KINEMATIC MODEL OF A ROBOT BY THE USE OF ARTIFICIAL NEURAL NETWORKS: A COMPLEMENT TO THE TEACHING OF ROBOTICS

***Abstract:*** *Teaching robotics necessarily involves the study of the kinematic models of robot manipulators. In turn, the kinematics of a robot manipulator can be described by its forward and reverse models. The inverse kinematic model, which provides the status of the joints according to the desired position for the tool of the robot, is typically taught and described in robotics classes through an algebraic way. However, the algebraic representation of this model is often difficult to obtain. Thus, although it is unquestionable the need for the accurate determination of the inverse kinematic model of a robot, the use of artificial neural networks (ANN) in the design phase can be very attractive, because it allows us to predict the behavior of the robot before the formal development of its model. In this way, this paper presents a relatively quick way to simulate the inverse kinematic model of a robot, thereby allowing the student to have an overview of the model, coming to identify points that should be corrected, or that can be optimized in the structure of a robot.*

***Keywords:*** Engineering education, Artificial neural networks, Robotics