UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMÇÃO

DCA0121 - INTELIGÊNCIA ARTIFICIAL APLICADA - PROJETO UNIDADE I

Componentes: **José Genilson da Silva Filho**
Renato Maia Chacon

RESUMO

Este projeto consiste na implementação de uma MLP (Multilayer Perceptron) na linguagem python com a ferramenta V-Rep para que um determinado robô seja capaz de desviar de forma autônoma de obstáculos colocados em uma pista virtual de treinamento criada manualmente. A escolha de uma MLP foi preferível à de um Perceptron simples, visto que nela existem neurônios dispostos em camadas escondidas permitindo que a classificação de mais de duas classes seja possível. O MLP faz uso de funções de ativação deriváveis para os casos de backpropagation. Este artigo irá primeiramente apresentar a rede trabalhada, assim como o esquema de treinamento utilizado e os resultados obtidos ao fim.

DESCRIÇÃO DO PROBLEMA A SER RESOLVIDO

A dupla precisou pensar em uma maneira de implementar uma rede neural que seja capaz de locomover um determinado robô em um ambiente desconhecido, de forma que não haja colisão do mesmo com obstáculos. O robô também deverá possuir apenas duas rodas que possam ser giradas de forma livre, obedecendo assim ao modelo cinemático diferencial.

ESTRATÉGIA UTILIZADA

Para que implementássemos uma Rede Neural Artificial (RNA) do tipo MLP para o treinamento do robô no V-Rep a dupla escolheu fazer uso da linguagem Python. Foi escolhido pelo grupo trabalhar com o robô Pioneer do V-Rep, de forma que sejam utilizados apenas 6 dos 16 sensores de ultrasons frontais dele para desviar de obstáculos através do controle da velocidade angular das duas rodas.

O Pioneer foi escolhido para facilitar uma das etapas mais difíceis da implementação de uma RNA: a escolha do conjunto do treinamento. Aproveitamos o fato do Pioneer ter sido desenvolvido para se locomover de forma autônoma em um ambiente, e armazenamos os valores das velocidades das rodas enquanto ele se locomovia por vários ambientes.

ESTRUTURA DA RNA

Utilizamos na nossa MLP 6 entradas (os 6 sensores ultrasons), 2 saídas, uma camada escondida com 10 neurônios.

Os pesos sinápticos foram iniciados com 0. Limitamos o máximo de 60000 iterações ou erro menor igual a 10^{-5}. Abaixo segue o esquemático da nossa MLP.

Abaixo segue um exemplo do esquema da nossa MLP sem as saídas (nossa MLP tem dois neurônios de saída):

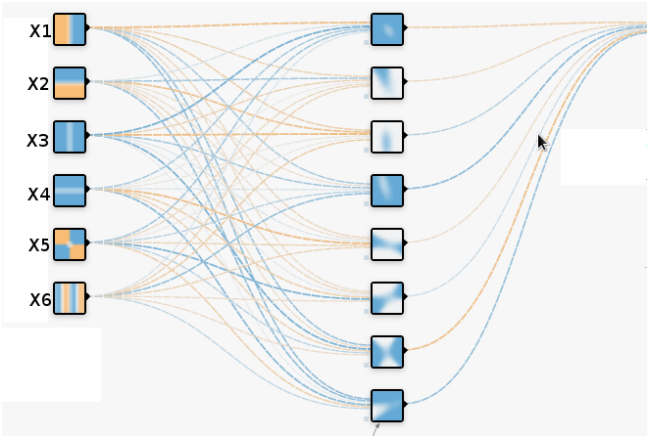


Figura 1: Esquema da MLP utilizada no treinamento

Para a função de ativação foi utilizada a tangente hiperbólica (tanh), que tem a seguinte característica:

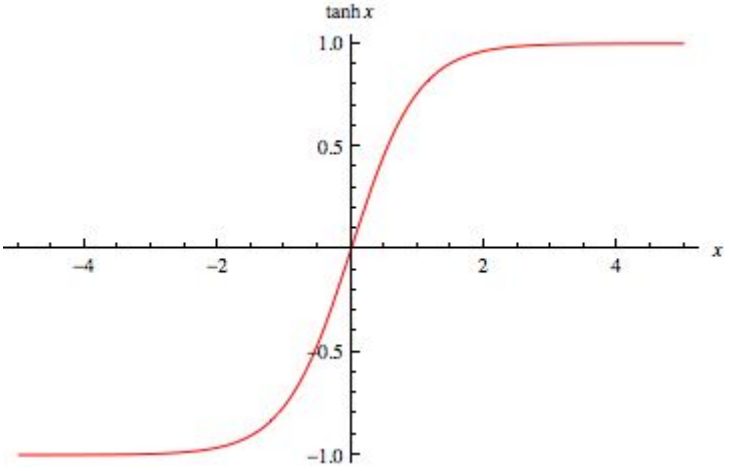


Figura 2: Função de ativação tanh

ESQUEMA DE TREINAMENTO

A função de treinamento, como o próprio nome sugere, irá treinar a rede a partir dos dados determinados logo abaixo. Foi definido um conjunto de treinamento: para tal, foi escolhido pelo grupo trabalhar com o robô Pioneer do V-Rep, de forma que sejam utilizados apenas 6 dos 16 sensores de ultrasom frontais dele para o conjunto de treinamento. Esse robô por si só já é capaz de andar em um determinado ambiente sem colidir, devido aos pesos utilizados por seu criador. Levando este dado em consideração, foi observado um volume de dados (conjunto de treinamento) para que o robô vague pelo ambiente construído de forma que os valores lidos pelos sensores sejam armazenados assim como as velocidades das rodas para criação do dataset. Foi notado que inicialmente, quando não era utilizados dados como intervalo e threshold, o conjunto de treinamento se tornava extenso e com saídas diferentes para entradas iguais, fazendo com que a rede não convergisse. A solução encontrada pelo grupo foi o uso do threshold. Foram então usados seis sensores de forma que os mesmos só responderiam quando alguma coisa fosse detectada à frente pelos mesmos. Atingida essa condição, quando um objeto for encontrado entre uma coordenada de 0.2 e 0 de detecção, o valor obtido é armazenado no vetor de conjunto de treinamento. E quando o valor da coordenada for maior que a máxima distância de detecção, será colocado '1' no vetor de treinamento. Mesmo obtendo essa solução, o grupo notou que no fim das contas é exigido um grande custo de treinamento para realização da tarefa.

Após gerado o conjunto de treinamento, foram escolhidos a olho valores julgados por serem os melhores para que fizessem com que o robô se locomovesse nas situações em que estava inserido no ambiente de teste. Abaixo está nosso conjunto de treinamento:

SENSORES				MOTORES			
s1	s2	s3	s4	s4	s6	VL	VR
0.20	1	1	1	1	1	1	0.5
0.20	0.20	1	1	1	1	1.5	0.5
0.10	0.10	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	0.10	0.10	1	1	0	0.75
1	1	0.20	0.20	1	1	0.6	1
1	1	1	1	0.20	0.20	0.5	1

1	1	1	1	1	0.20	0.5	1
1	1	1	1	0.10	0.10	0	1
1	1	1	0.20	1	1	0	1
1	1	0.20	1	1	1	0	1
1	1	1	0.10	0.10	1	0	0.5

Tabela1: Conjunto de treinamento

DESCRIÇÃO DETALHADA DO ALGORITMO

Foi necessário fazer dois códigos para o funcionamento desejado do robô: um para a MLP e outro para obter o conjunto de treinamento. Nesta seção será abordado primeiramente o da MLP, responsável pelo backpropagation. Após importar as devidas bibliotecas foram feitos algumas definições e atribuições básicas para garantir o funcionamento da MLP, como a função rand que servirá para gerar números aleatórios entre ‘a’ e ‘b’, tanto a função de ativação quanto a derivada da mesma para a fase de backpropagation. Para começo de uso da rede neural, foi definida uma para inicialização dos valores importantes que necessitam ser esclarecidos, tais como o total de 60000 para o número de máximo de iterações. Isto se dá pelo grupo querer obter um valor ainda menor de erro, como será visto adiante.

Adiante, a matriz dos pesos é criada de forma que seja preenchida por zeros. Na fase forward do treinamento, uma combinação linear é feita com os valores dos pesos para que passem pela função de ativação de acordo com quantas camadas estiverem na rede. Na fase do backpropagation denominada backward, as derivadas da função são obtidas para que seja calculado o erro. Abaixo, é definida a função do erro quadrático, que retornará o erro em cada iteração feita. A função test determinará como será a saída dada uma determinada entrada após a rede estar treinada com seus devidos pesos.

backpropagation.py
(algoritmo de treinamento da MLP)

```
27  class RedeNeural:
28      #
29      def __init__(self, nos_entrada, nos_ocultos, nos_saida):
30          # camada de entrada
31          self.nos_entrada = nos_entrada + 1 # +1 por causa do no do bias
32          # camada oculta
33          self.nos_ocultos = nos_ocultos
34          # camada de saida
35          self.nos_saida = nos_saida
36          # quantidade maxima de interacoes
37          self.max_interacoes = 20000
38          # taxa de aprendizado
39          self.taxa_aprendizado = 0.085
40          self.momentum = 0.0001
41          self.erro_minumum = 0.00001
42
43          # activations for nodes
44          # cria uma matriz, preenchida com uns, de uma linha pela quantidade de nos
45          self.ativacao_entrada = numpy.ones(self.nos_entrada)
46          self.ativacao_ocultos = numpy.ones(self.nos_ocultos)
47          self.ativacao_saida = numpy.ones(self.nos_saida)
48
49          # contém os resultados das ativações de saida
50          self.resultados_ativacao_saida = numpy.ones(self.nos_saida)
51
52          # criar a matriz de pesos, preenchidas com zeros
53          self.wi = numpy.zeros((self.nos_entrada, self.nos_ocultos))
54          self.wo = numpy.zeros((self.nos_ocultos, self.nos_saida))
55
```

wander.py
(código utilizado para aquisição do conjunto de treinamento)

```
95
96
97              if ((i == 1 or i == 2 or i == 3 or i == 4 or i ==5 or i == 6) and state != 0):
98                  if(coord[2] <= 0.2 and coord[2] >= 0):
99                      distancias.append(coord[2])
100                 elif (coord[2] > maxDetectionDist):
101                     distancias.append(1)
```

RESULTADOS E TESTES

Quanto ao erro quadrático, foi observado que a um total de 60000 iterações, foi obtido um valor de 0.0000425, considerado satisfatório para o experimento. Na curva de erro gerada abaixo, pode-se verificar uma oscilação em seu valor ainda alto antes das 10000 iterações. Entretanto, na medida em que o número de iterações cresce, o erro diminui consideravelmente até atingir o valor de 0.0000425.

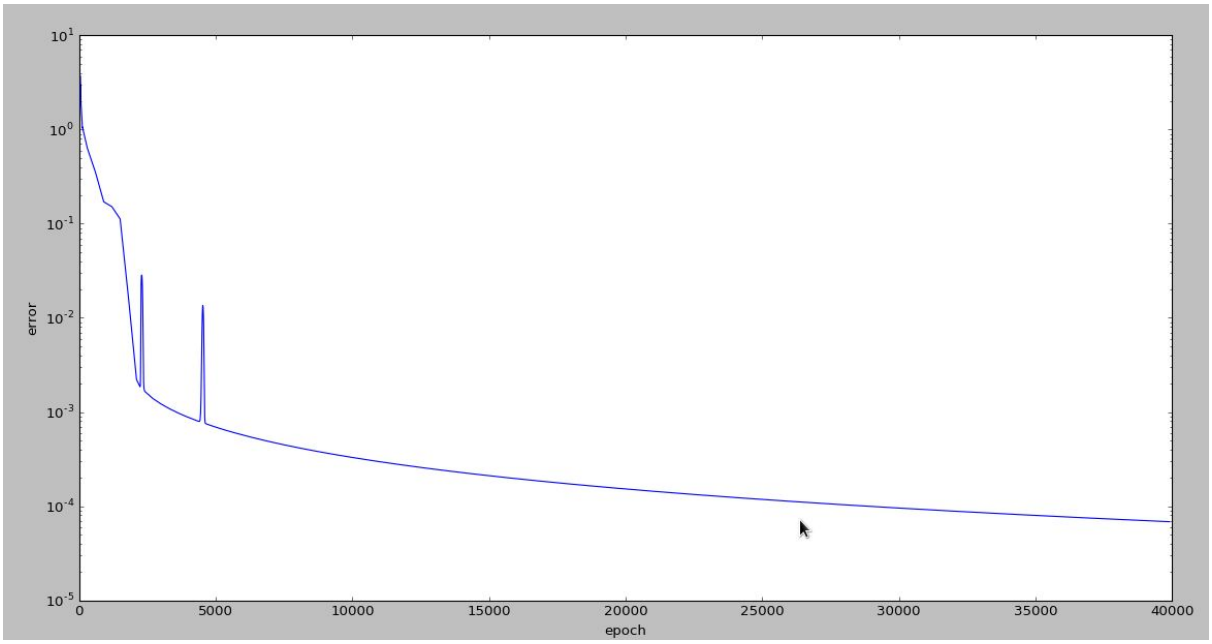
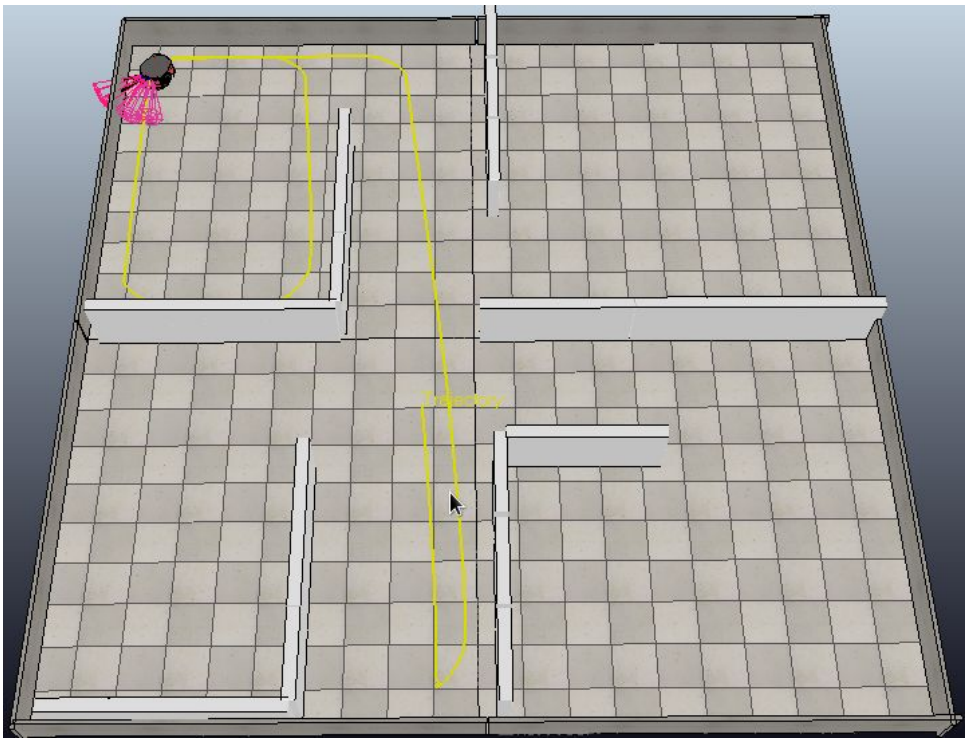


Figura 3: Curva de erro quadrático mínimo

O robô foi testado em mapas diversos para verificar o resultado obtido pela nossa RNA. Verificamos que obtivemos ótimos resultados, já que o robô evita os obstáculos presentes no mapa, consequentemente não colide em nenhuma situação. No entanto, por a MLP ser naturalmente uma técnica de Inteligência Artificial para a classificação, o robô em certos ambientes ficava andando em círculos por não existir um 'target' definido, um alvo a ser alcançado. Abaixo temos a imagem do percurso do robô feito em um dos mapas.



No final deste artigo contém um link que para um vídeo mostrando tudo que foi feito para a construção da rede e o funcionamento do robô após seu treinamento.

REQUISITOS FUNCIONAIS DA IMPLEMENTAÇÃO

Neste projeto, todos as etapas até aqui citadas foram de total importância para chegarmos até o resultado final. A começar pela implementação do algoritmo de treinamento. A MLP faz o cálculo da saída da camada escondida fazendo o somatório das entradas multiplicadas por seus respectivos pesos, encontrando um resultado que passa pela função de ativação tanh, que dá a entrada da camada de saída. No caso desse trabalho, temos seis entradas externas além do bias, onde as seis entradas definem a distância do robô a um possível obstáculo à sua frente ou lados.

Na camada de saída temos um número de neurônios igual ao número de saídas desejadas (duas), onde o mesmo processo da camada anterior é repetido, após isso, teremos a saída da primeira iteração do sistema. Caso o erro não seja satisfatório, acontecerá a atualização de pesos, até que se chegue em um erro aceitável para o problema. A atualização dos pesos é feita pelo algoritmo backpropagation.

Começamos o treinamento do robô utilizando um dataset muito grande, e com valores que faziam com que a rede divergisse. Após um pouco estudo, decidimos utilizar um robô

que já andava só por ambientes sem colidir, e usá-lo para obter o nosso conjunto de treinamento.

Com esses dados em mãos, fizemos um filtro manual do que seriam os melhores pesos, e utilizamos-os para treinar a rede.

CONCLUSÃO

Partindo dos estudos relacionados a RNA's, e levando em conta o objetivo principal deste trabalho que era a aplicação do método e suas dificuldades, chegamos a conclusão que tivemos bons resultados de aprendizado utilizando o algoritmo RNA do tipo Multi-Layer-Perceptron (MLP) com backpropagation..

Para terminar, concluirmos que os objetivos específicos do trabalho foram realizados, bem como o estudo e aprendizado de RNA. Além disso, adquirimos experiência no uso de uma nova ferramenta, que foi o VRep.

LINKS

- http://www.coppeliarobotics.com/downloads.html (site do VRep)
- https://www.sites.google.com/site/vrepjai/ (Código utilizado para a integração do VRep com Python)
- https://mega.nz/#F!hkAhxRJL!43fm22pZASW3UCMMq7-8ZQ (link com vídeo e códigos)