UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CMP135 – ARQUITETURAS ESPECIAIS DE COMPUTADORES

REDES NEURAIS ARTIFICIAS
SOM (KOHONEN)

Prof. Philippe O. A. Navaux
Marcos Rafael Boschetti

Sumário

INTRODUÇÃO

Este trabalho trata de um tópico sobre o qual ressurge grande interesse da comunidade científica mundial: as **redes neurais artificiais**.

Serão apresentados os principais conceitos e aplicações dessas redes, bem como as constatações realizadas sobre a experiência prática de implementação de uma rede de Kohonen, a qual recebeu a ênfase principal neste estudo.

REDES NEURAIS ARTIFICIAIS (RNAS)

O final da década de 80 marcou o ressurgimento da área de redes neurais artificiais, também conhecida como conexionismo ou sistemas de processamento paralelo e distribuído. Trata-se de uma forma de computação não algoritmica, caracterizadas por sistemas que, em algum nível, relembram a estrutura do cérebro humano. Por não ser baseada em regras ou programas, a computação neural se constitui em uma alternativa à computação algoritmica convencional.

RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos) que computam determinadas funções matemáticas e lógicas. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, normalmente unidirecionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. O funcionamento dessas redes é inspirado na estrutura funcional do cérebro humano.

A solução de problemas através de RNAs é bastante atrativa, já que a forma como estes são representados internamente pela rede e o paralelismo natural inerente à arquitetura das RNAs criam a possibilidade de um desempenho superior ao dos modelos convencionais. Em RNAs, o procedimento usual na solução de problemas passa, inicialmente, por uma fase de aprendizagem, na qual um conjunto de exemplos é apresentado para a rede, a qual extrai automaticamente as características necessárias para representar a informação fornecida. Essas características são usadas posteriormente para gerar respostas para o problema.

A capacidade de aprender atrevés de exemplos e de generalizar a informação aprendida são, sem dúvida, os atrativos principais da solução de problmas através de RNAs. A generalização, que está associada à capacidade de a rede aprender através de um conjunto reduzido de exemplos e posteriormente dar respostas coerentes para dados não conhecidos, é uma demonstração de que a capacidade das RNAs vai muito além do que simplesmente mapear relações de entrada e saída. As RNAS são capazes de extrair informações não apresentadas de forma explícita através dos exemplos . Não obstante, as RNAs são capazes de atuar como mapeadores universais de funções multi-variáveis, com custo computacional que cresce apenas linearmente com o número de variáveis. Outras características importantes são a capacidade de auto-organização e de processamento temporal que, aliadas àquelas citadas anteriormente, fazem das RNAs uma ferramenta computacional extremamente poderosa e atrativa para a solução e problemas complexos.

MOTIVAÇÃO (REDES BIOLÓGICAS)

O cérebro humano contém cerca de 10^{11} nodos. Cada um desses nodos processa e se comunica com milhares de outros nodos continuamente e em paralelo. A estrutura individual dos nodos, a topologia de suas conexões e o comportamento conjunto destes nodos naturais formam a base para o estudo das RNAs.

O cérebro humano tem a capacidade de reconhecer padrões e relacioná-los, usar e armazenar conhecimento por experiência, além de interpretar observações. Apesar do estudo contínuo, o funcionamento das redes biológicas ainda não foi totalmente desvendado pelo homem. Não se sabe ao certo a forma como as funções cerebrais são realizadas. O que se tem até o momento são modelos, os quais são atualizados a cada nova descoberta. No entanto, a estrutura fisiológica básica destas redes de nodos naturais é conhecida e é exatamente nessa estrutura fisiológica que se baseiam as RNAs.

As RNAs tentam reproduzir as funções das redes biológicas, buscando implementar seu comportamento básico e sua dinâmica. No entanto, do ponto de vista físico, no momento, as redes artificiais se diferem bastante das redes biológicas. É importante, contudo, observar as similaridades entre esses dois tipos de sistemas. Como características comuns podem ser citadas que os dois sistemas são baseados em unidades de computação paralela e distribuída que se comunicam através de conexões sinápticas, possuem detetores de características, redundância e modularização das conexões. Acredita-se, portanto, que o futuro

da neurocomputação possa se beneficiar ainda mais do desenvolvimento de modelos que tenham apelo biológico.

NEURÔNIOS ARTIFICIAIS : MODELO MCP

O modelo foi inicialmente proposto por McCulloch e Pitts e trata-se de uma simplificação sobre o que se sabia à respeito do neurônio biológico naquela época. A sua descrição matemática resultou em um modelo com n terminais de entrada x1, x2, ..., xn (que representam os dendritos), e apenas um terminal de saída y (representando o axônio). Para emular o comportamento das sinapses os terminais de entrada do neurônio tem pesos acoplados w1, w2,....,wn, cujos valores podem ser positivos ou negativos, dependendo das sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular i no neurônio pós-sináptico é dado por: xiwi. Os pesos determinam em que grau o neurônio deve considerar sinais de disparo que ocorrem naquela conexão.

Um neurônio biológico dispara quando a soma dos impulsos que ele recebe ultrapassa o seu limiar de excitação (threshold). O corpo do neurônio, por sua vez, é emulado por um mecanismo simples que faz a soma dos valores xiwi recebidos pelo neurônio (soma ponderada), e decide se o neurônio deve ou não disparar (saída igual a 1 ou 0) comparando a soma obtida ao limiar ou threshold do neurônio. No modelo MCP, a ativação do neurônio é obtida através da aplicação de uma função de ativação, que ativa a saída ou não dependendo do valor da soma ponderada de suas entradas. Na descrição original do modelo MCP,a função de ativação é dada pela função de limiar descrita abaixo:

$$\sum_{i=1..n} xiwi >= q$$

onde n é o número de entradas do neurônio, wi é o peso associado à entrada xi, e q é o limiar (threshold) do neurônio.

McCulloch e Pitts simplificaram seu modelo assumindo que os nodos em cada camada da rede disparam sincronamente, isto é, todos os nodos são avaliados ao mesmo tempo e também que as entradas em um instante de tempo t produzem a sua saída no tempo t+1. Em sistemas biológicos, sabe-se que não existe um mecanismo para sincronizar as ações dos nodos, nem há restrição para que as suas saídas sejam ativadas em tempos discretos como no modelo MCP. Sabe-se também que o valor da próxima saída dos nodos biológicos depende enormemente das ativações dos estados anteriores, já que até mesmo os neuro-transmissores liberados anteriormente levam algum tempo para se

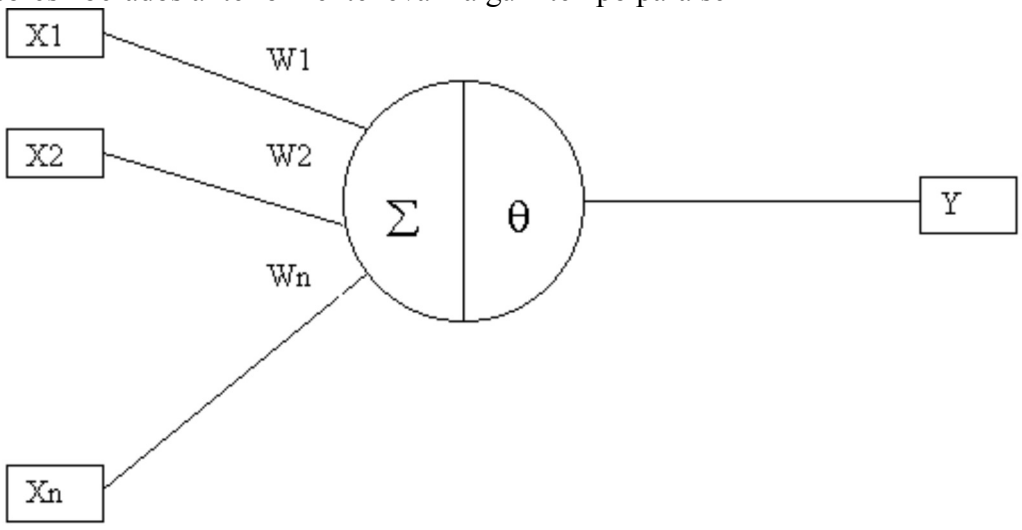


FIG1: Neurônio de McCulloch e Pitts

recombinarem, influenciando, assim, as ativações seguintes.

PRINCIPAIS ARQUITETURAS DE RNAS

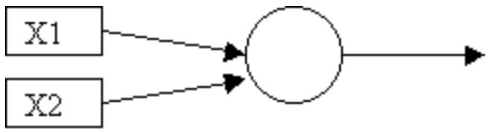
A definição da arquitetura de uma RNA é um parâmetro importante na sua concepção, uma vez que ela restringe o tipo de problema que pode ser tratado pela rede. Redes com uma camada única de nodos MCP, por exemplo, só conseguem resolver problemas linearmente separáveis. Redes recorrentes (feedback), por sua vez, são mais apropriadas para resolver problemas que envolvem processamento temporal. Fazem parte da definição da arquitetura os seguintes parâmetros: número de camadas da rede, número de nodos em cada camada, tipo de conexão entre os nodos e topologia da rede.

Quanto ao número de camadas, pode-se ter:

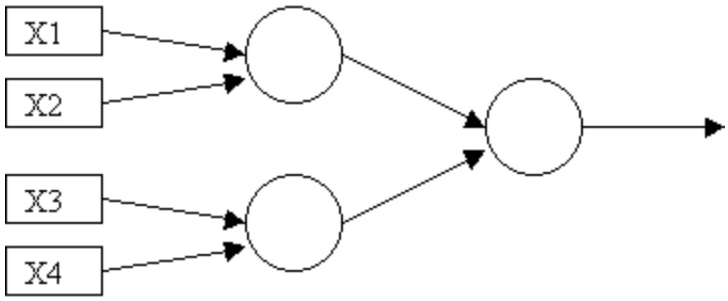
- **Redes de Camada Única:** só existe um nó entre qualquer entrada e qualquer saída da rede.
- **Redes de Múltiplas Camadas:** existe mais de um neurônio entre alguma entrada e alguma saída da rede.

Os nodos podem ter conexões do tipo:

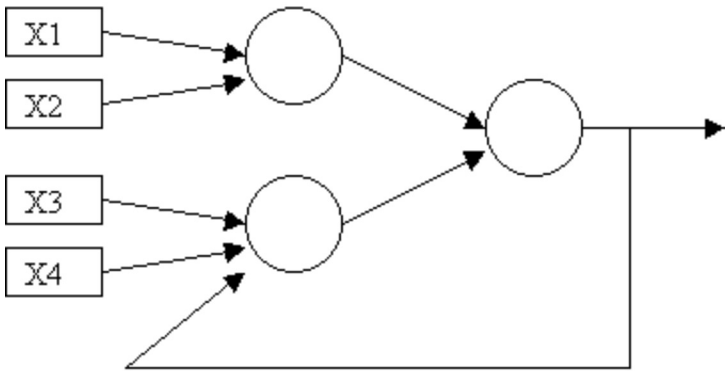
- **Feedforward:** a saída de um neurônio na i-ésima camada da rede não pode ser usada como entrada em nodos em camadas de índice menor ou igual a i.
- **Feedback:** a saída de algum neurônio na i-ésima camada da rede é usada como entrada de nodos em camadas de índice menor ou igual a i.



(a) Rede de camada única feedforward



(b) Rede de múltiplas camadas feedforward



(c) Rede de múltiplas camadas feedback

Para se obter uma boa generalização com RNAs, deve-se fornecer para a rede a maior quantidade possível de informação a respeito do problema a ser tratado. Isto significa que uma grande quantidade de nodos deve ser usada para sub-tarefas específicas. Contudo, por problemas de complexidade computacional (espaço e tempo), deve-se buscar reduzir ao mínimo o número de nodos e a quantidade de conexões entre eles. Portanto, é importante, a definição de algoritmos que não somente otimizem os pesos para uma dada arquitetura, mas também otimizem a própria arquitetura. Isto significa, em particular, a otimização do número de camadas e do número de nodos por camada.

APRENDIZAGEM

Redes neurais artificiais possuem a capacidade de aprender por exemplos e fazer interpolações e extrapolações do que aprenderam. No aprendizado conexionista não se procura obter regras como na

abordagem simbólica da Inteligência Artificial, mas determinar a intensidade de conexões entre neurônios. Um conjunto de procedimentos bem definidos para adaptar os parâmetros de uma RNA para que a mesma possa aprender uma determinada função é chamado de algoritmo de aprendizado.

A utilização de uma RNA na solução de uma tarefa passa, inicialmente, por uma fase de aprendizagem, onde a rede extrai informações relevantes de padrões de informação a ela apresentados. A etapa de aprendizagem consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões entre as unidades de processamento, que guardam, ao final do processo, o conhecimento que a rede adquiriu do ambiente que está operando.

APRENDIZADO SUPERVISIONADO

Este método de aprendizado é o mais comum no treinamento das RNAs, tanto de neurônios com pesos, como de neurônios sem pesos, sendo chamado aprendizado supervisionado porque a entrada e saída desejadas para a rede são fornecidas por um supervisor externo. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos.

A rede tem sua saída corrente calculada comparada com a saída desejada, recebendo informações do supervisor sobre o erro da resposta atual. A minimização da diferença é incremental, já que pequenos ajustes são feitos nos pesos à cada etapa de treinamento, de tal forma que estes caminhem para uma solução.

A desvantagem do aprendizado supervisionado é que, na ausência de professor, a rede não conseguirá aprender novas estratégias para situações não cobertas pelos exemplos de treinamento da rede.

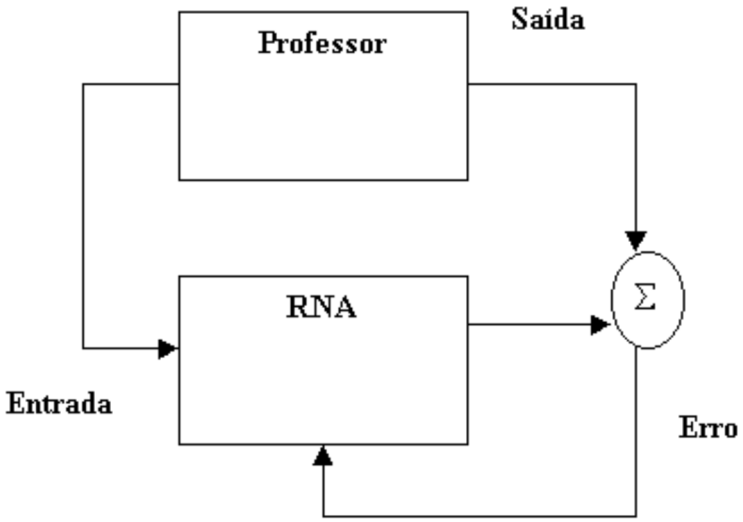


FIG2: Aprendizado supervisionado

APRENDIZADO NÃO SUPERVISIONADO

No aprendizado não supervisionado, como o próprio nome sugere, não há um professor ou supervisor para acompanhar o processo de aprendizado. Muitos dos sistemas biológicos ocorrem através de aprendizado não supervisionado. Para esses algoritmos, somente os padrões de entrada estão disponíveis para a rede, ao contrário do aprendizado supervisionado, cujo conjunto de treinamento possui pares de entrada e saída.

A partir do momento em que a rede estabelece uma harmonia com as regularidades estatísticas da entrada de dados, desenvolve-se nela uma capacidade de formar representações internas para codificar características da entrada e criar novas classes ou grupos automaticamente.

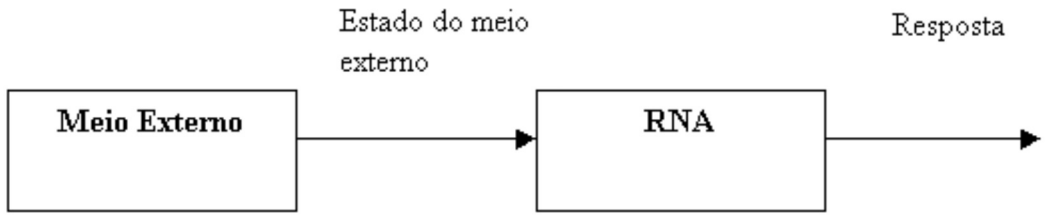


FIG3: Aprendizado não supervisionado

REDES SOM

As redes *self-organizing*, SOM (Self-Organizing Maps), também chamadas de mapas de características *self-organizing,* foram desenvolvidas por Teuvo Kohonen.

Essas redes posuem forte inspiração neurofisiológica e funcionam, basicamente, da seguinte forma: quando um padrão de entrada **p** é apresentado a rede procura a unidade mais parecida com **p**. Durante o seu treinamento, a rede aumenta a semelhança do nodo escolhido e de seus vizinhos ao padrão **p**. Desta forma, a rede constrói um mapa topológico onde nodos que estão topologicamente próximos respondem de forma semelhante a padrões de entrada semelhantes.

A rede SOM utiliza um algoritmo de aprendizado competitivo, onde os nodos da camada de saída competem entre si para se tornarem ativos, ou seja, para ver quem gera o maior valor de saída. Assim, para cada padrão de entrada apenas um nodo de saída ou nodo por grupo se torna ativo. Esta competição é camada de winner-takes-all.

ARQUITETURA

Em uma rede SOM, os nodos se organizam em uma grade ou reticulado, geralmente bidimensional, podendo algumas vezes ser unidimensional. Cada nodo da rede de som recebe todas as entradas e funciona como um discriminador ou extrator de características . Quanto mais semelhante for a entrada dos pesos de um nodo, maior a saída produzida pelo nodo. A saída da rede é formada pela saída de todos os seus nodos. Durante a fase de aprendizagem, os nodos se especializam para a detecção de uma série de padrões de entrada. Os nodos se organizam topologicamente, fazendo com que padrões detectados por um dado nodo estejam relacionados com a coordenada da posição do nodo dentro da rede. Desta forma, um mapa de características *self-organizing* forma mapas topológicos de padrões de entrada, onde padrões semelhantes são detectados por nodos próximos dentro do reticulado .

A função abaixo mostra que o estado de ativação de um nodo é determinado pela distância Euclidiana entre seu peso e o vetor de entrada.

$$Yj = \sum_{i=1...n} || xi - wji||$$

TREINAMENTO

O treinamento de uma rede SOM é competitivo e não supervisionado. O algoritmo de treinamento original organiza os nodos em vizinhanças locais. Cada vez que um novo padrão de treinamento é apresentado à rede, os nodos competem entre si para ver quem gera a maior saída. Definido o nodo vencedor tem ínicio o processo de atualização de pesos. Neste processo, apenas o nodo vencedor e seus vizinhos dentro de um certo raio ou área de vizinhança atualizam seus pesos. Durante o treinamento, a taxa de aprendizado e o raio de vizinhança são continuamente decrementados.

O algoritmo de treinamento para redes tipo SOM pode ser resumido da seguinte maneira:

Passo 1: Inicializar pesos e parâmetros
Passo2 : Repetir (até o mapa de características não mudar)
Para cada padrão de treinamento x faça
Definir nodo vencedor
Atualizar os pesos deste nodo e de seus vizinhos
Se número do ciclo for múltiplo de n então
Reduzir taxa de aprendizagem e vizinhança

Os pesos iniciais podem ser escolhidos aleatóiamente. Após o treinamento, a rede SOM agrupa os padrões em clusters ou agrupamentos. De acordo com Kohonen, a rede SOM não foi criada para reconhecimento de padrões, mas para agrupamento, visualização e abstração. Ainda assim, as redes SOM podem ser utilizadas para o reconhecimento ou classificação de padrões. Nesse caso, a rede deve ser utilizada junto com um modelo de aprendizado supervisionado.

APLICAÇÕES COMERCIAIS COM REDES NEURAIS

A tabela abaixo mostra uma lista de empresas e produtos que utilizam redes neurais. Obviamente, a tabela é apenas um pequeno exemplo, mas pode-se perceber que entre as principais funcionalidades estão reconhecimento de padrões como calígrafia, o que será cada vez mais importante na interface homem-computador e também análise de padrões para previsão. Empresas de cartão de crédito abstecem seus bancos de dados com o histórico de gastos de seus clientes e utilizam redes neurais para varrer esse dados e encontrar comportamentos comerciais que não se enquadram na normalidade de um determinado indivíduo. Através dessas técnicas fica mais fácil detectar eventuais fraudes.

Empresa	**Produto (funcionalidade)**
Adaptative Decision Systems	Avalia decisões de marketing direto

BehavHeuristics Inc.	Prevê demanda de passagens aéreas
Britvic	Previsão de venda de refrigerantes
Microsoft	Sistema de marketing direto via mail
HNC Software	Sistema de valor imobiliário
Synaptics	Leitor de cheques
Eletronic Data publishing Inc.	Reconhecimento de caracteres
Fujitso	Software de entradas em pcs em forma de caneta
Lexicus	Reconhecedor de letras cursivas para PC
MasterCard	Detecção de desvios em hábitos comuns

UMA APLICAÇÃO PRÁTICA

PROBLEMA

O problema resolvido utilizando-se redes neurais foi a classificação de digitos numéricos de **0** até **9**. Para isso desenvolveu-se um software em ambiente Delphi Ó que implementa uma rede neural artificial SOM.

A SOLUÇÃO

Para resover esse problema foi implementado um softaware que replica o comportamento de uma rede neural do tipo SOM (figura 4) .

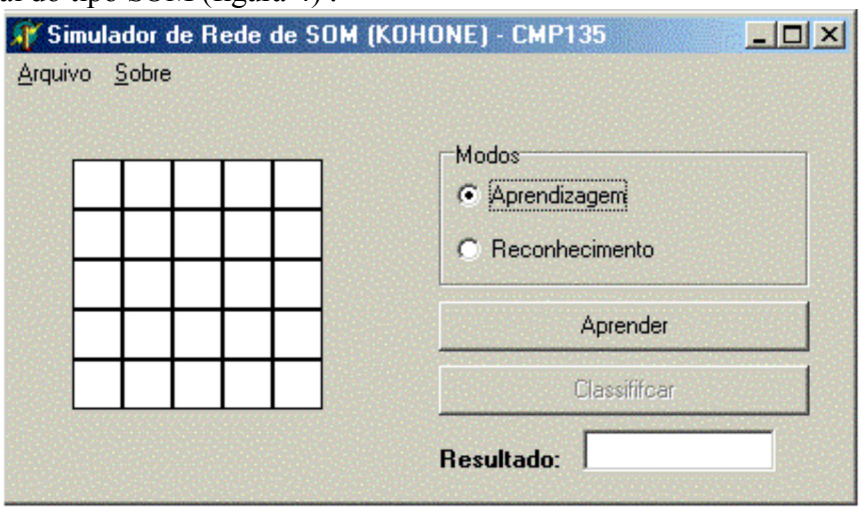


FIG4: Interface

Foram modelados 11 neurônios, sendo o primeiro, um neurônio de entrada e os demais neurônios de reconhecimento, cada um para um dos dez digitos. A topologia utilizada pode ser visualizada na figura abaixo:

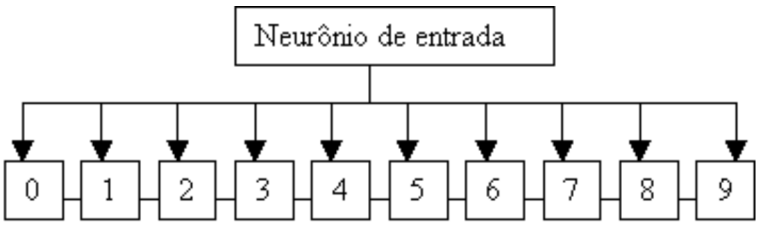


FIG5: Topologia da rede SOM

A entrada de novos padrões tanto para treinamento quanto para classificação é realizada através de cliques do mouse sobre a grade. É possível salvar em arquivo um determinado treinamento e carregá-lo novamente quando for necessário.

O programa atua em dois modos possíveis:

- **Modo de Aprendizagem:** Neste modo o software espera que os padrões a serem reconhecidos sejam apresentados a ele.
- **Modo de Reconhecimento:** Nesta fase o usuário entra com um padrão e o software tenta reconhecê-lo.

É importante ressaltar que nesta versão não foi implementada etapa de rotulação dos nodos, assim o software espera que os padrões de treinamento sejam fornecidos na ordem pelo usuário. Por exemplo, se o treinamento for feito na seguinte sequência 1,0,2,3,4,5,6,7,8 e 9, na fase de reconhecimento é aplicado o mapa (desenho) do padrão 1 o software dirá que o padrão fornecido é o zero, pois o mapa de bits do padrão 1 estará asociado ao símbolo zero.

TREINAMENTO

Conforme Kohonen falou, as redes SOM não foram projetadas para reconhecimento de padrões, mas primordialmente para agrupamentos. Contudo, é possível utilizar a rede para essa finalidade, desde que incluída uma etapa de aprendizagem supervisionada. O algoritmo de treinamento utilizado neste software é idêntico ao apresentado na seção sobre redes SOM, com a diferença de que espera-se que o usuário forneça inicialmente os dez padrões de análise pretendidos, de forma que o software possa fazer a carga supervisionada desses padrões, o que permitirá que se saiba quais padrões a serem encontrados/reconhecidos.

Para o treinamento é utilizada a distância entre dois vetores, a função abaixo demonstra a realização do cálculo:

```
function Distancia(var X,Y:array of extended):extended; // uvect.pas (jpss)
var I:longint;
   R:extended;
begin
R:=0;
for I:=Low(X) to High(X) do
   R:=R+sqr(X[I]-Y[I]);
Distancia:=sqrt(R);
end;
```

Conforme mencionado, em uma rede SOM o aprendizado é competitivo, ou seja, segue a filosofia *winner-takes-all*. O nodo vencedor e sua vizinhança devem ser atualizados. A vizinhança para a topologia adotada (figura 5) refere-se ao vizinho imediatamente à direita e à esquerda (se existirem). Os testes realizados mostraram que para esse conjunto pequeno de dados e padrões, a atualização da vizinhança não trouxe melhorias.

RECONHECIMENTO

Uma vez treinados os neurônios o usuário pode experimentar a rede. O reconhecimento demonstra a característica fantástica da rede neural artificial que é a generalização das respostas. O usuário insere um padrão clicando sobre a grade apresentada na interface e "desenha" o número pretendido. Após, basta clicar em classificar. Novamente, o neurônio vencedor é aquele que apresenta o maior valor para a função de ativação. Ter o maior valor neste caso é ter a menor distância Euclidiana para o padrão inserido. Durante a fase de reconhecimento, conforme esperado não existe nenhuma espécie de ajustes nos pesos.

FIG6: Reconhecimento do número 4

Neste exemplo é possível observar que apesar dos "ruídos" inseridos, a rede consegue generalizar seu conhecimento e reconhecer o número 4.

CONCLUSÃO

Neste trabalho foi possível ver a capacidade de aprendizagem e reconhecimento de padrões do mecanismo conhecido como rede neural artificial.

Os conceitos de uma rede de SOM (*self-organizing*) foram aplicados para a construção de um software que replica o comportamento desse tipo particular de rede, a qual tem profundo relacionamento com estruturas biológicas existentes na natureza.

A rede SOM desenvolvida, possui onze neurônios e a capacidade de generalização de conhecimento pôde ser observada em diversos testes realizados. Embora redes tipo SOM não tenham sido projetadas para reconhecimento de padrões, a inclusão de uma etapa supervisionada em seu algoritmo de aprendizagem possibilita que seja atingida essa finalidade, conforme demonstrou a aplicação prática implementada.

BIBLIOGRAFIA

[BRA98] Braga, Antônio de P.; Carvalho André P. De L.;Ludermir, Teresa B. *"Fundamentos de Redes Neurais Artificiais"; 11ª Escola de Computação, Brasil, 1998*

[BEN93] Krose ,Ben J. A.; Van der Smagt, P.; "*An Introduction to Neural Networks*";em : http://www.ele.ita.br/cnrn/outros/Intro_NN_Krose_Smagt_1993.zip

Conselho Nacional de Redes Neurais Links Interessantes sobre Redes Neurais
e Assuntos Correlatos, em: http://www.ele.ita.br/cnrn/links.html

Lista de aplicações comerciais de redes neurais, em: http://www.emsl.pnl.gov:2080/proj/neuron/neural/products/

ANEXO (CÓDIGO FONTE)

```
unit Principal;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,ExtCtrls, StdCtrls, Menus,uvect;
//---------------------------------------------------------------------------

const

   TamanhoSquare = 25;
   Branco =  0;
   Preto  =  1;

//---------------------------------------------------------------------------
type



  TSquare = class(Timage)

  private

       Status : Integer;
       Indice : Integer;
       xPos   : Integer;
       yPos   : Integer;

       procedure inicializa(x,y:integer);

  public

       procedure clicaSquare(Sender : TObject);

  end;
//---------------------------------------------------------------------------
  TSquareMatriz = array[1..5,1..5] of TSquare;
//---------------------------------------------------------------------------
  TVetorPesos = array [1..25] of extended;
//---------------------------------------------------------------------------

  //--- 11 entradas, uma para o Neuronio de Entrada e as demais para reconhecer
  //--- de 0 a 9

  ArrayNeuronios = array[0..10] of TVetorPesos;
//---------------------------------------------------------------------------

  TFormPrincipal = class(TForm)

    PanelImagem: TPanel;
    RadioGroup1: TRadioGroup;
    BotAprender: TButton;
    MainMenu1: TMainMenu;
    Arquivo1: TMenuItem;
    Sobre1: TMenuItem;
    Sobre2: TMenuItem;
```

```
    BotClassificar: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    SaveDialog1: TSaveDialog;
    OpenDialog1: TOpenDialog;
    Abrir1: TMenuItem;
    Salvarcomo1: TMenuItem;
    Sair1: TMenuItem;

    procedure FormCreate(Sender: TObject);
    procedure RadioGroup1Click(Sender: TObject);
    procedure Sair1Click(Sender: TObject);
    procedure BotAprenderClick(Sender: TObject);
    procedure BotClassificarClick(Sender: TObject);
    procedure SalvarComo1Click(Sender: TObject);
    procedure Abrir1Click(Sender: TObject);

  private
    { Private declarations }
    procedure novaImagem(var MatrizDeSquares : TSquareMatriz);
    procedure defineStatusBotoes;

  public
    { Public declarations }
    procedure matriz2Vetor(Matriz : TSquareMatriz; var Vetor : TVetorPesos);
    function  selecionaVencedor(Matriz : TSquareMatriz; var VNeuronios : ArrayNeuronios; modo:boolean) : Integer;
    procedure resetaVetor(var VNeuronios : ArrayNeuronios);
    function  procuraZero(var VNeuronios : ArrayNeuronios;indice:integer):integer;
    procedure inicializaRandomico(var VNeuronios : ArrayNeuronios);
    //procedure mostraNeuronios;
    procedure salvaArquivo(FileName : String);
    procedure abreArquivo(FileName : String);

  end;
//---------------------------------------------------------------------------
var
  FormPrincipal : TFormPrincipal;
  MatrizSquares : TSquareMatriz;
  Neuronios     : ArrayNeuronios;
  n             : real;
implementation

{$R *.dfm}

//---------------------------------------------------------------------------
procedure TSquare.inicializa(x,y:integer);

begin

    //Imagem :=  TImage.Create(FormPrincipal.PanelImagem);

    with Self do
       begin
         Parent := FormPrincipal.PanelImagem;
         SetBounds(TamanhoSquare*x,TamanhoSquare*y,TamanhoSquare,TamanhoSquare);
         with canvas do
            begin
               Pen.Color:= clBlack;
               Pen.Width := 1;
               Brush.Style:= bsSolid;
               Canvas.Brush.Color := clWhite;
               Canvas.Rectangle(0,0,TamanhoSquare,TamanhoSquare);
```

```
            end;
          Visible := true;
          Cursor:=crHandPoint;
        end;

    Status:= Branco;
    Indice := (x-1)*5+y;
    xPos := x;
    yPos := y;
    Self.OnClick := Self.clicaSquare;

end;
//------------------------------------------------------------------------------
procedure TSquare.clicaSquare;

begin
    if (MatrizSquares[Self.xPos,Self.yPos].Status = Branco) then
      begin
          with MatrizSquares[Self.xPos,Self.yPos] do
              begin
                  Canvas.Brush.Color := clBlack;
                  Canvas.Rectangle(0,0,TamanhoSquare,TamanhoSquare);
                  Repaint;
              end;
          MatrizSquares[Self.xPos,Self.yPos].Status:= Preto;
      end
    else
      begin
          with MatrizSquares[Self.xPos,Self.yPos] do
              begin
                  Canvas.Brush.Color := clWhite;
                  Canvas.Rectangle(0,0,TamanhoSquare,TamanhoSquare);
                  Repaint;
              end;
          MatrizSquares[Self.xPos,Self.yPos].Status:= Branco;
      end;

end;
//------------------------------------------------------------------------------

  procedure TFormPrincipal.novaImagem(var MatrizDeSquares : TSquareMatriz);
  var
    i,j : Integer;
  begin
      for i:= 1 to 5 do
          for j := 1 to 5 do
            begin
              MatrizDeSquares[i,j] := TSquare.Create(FormPrincipal.PanelImagem);
              MatrizDeSquares[i,j].inicializa(i,j);
            end;
end;

//------------------------------------------------------------------------------
  procedure TFormPrincipal.defineStatusBotoes;
  begin

      if( RadioGroup1.ItemIndex = 0) //--- Escolhido modo de aprendizagem
        then
            begin
              FormPrincipal.BotClassificar.Enabled := false;
              FormPrincipal.BotAprender.Enabled := true
            end
```

```
        else
            begin
             FormPrincipal.BotAprender.Enabled := false;
             FormPrincipal.BotClassificar.Enabled := true;
            end;
 end;
//---------------------------------------------------------------------------
 procedure TFormPrincipal.matriz2Vetor(Matriz : TSquareMatriz; var Vetor : TVetorPesos);

 var
    i,j : Integer;
 begin

    for I:= 1 to 5 do
       for j:= 1 to 5 do
          Vetor[Matriz[i,j].Indice] := Matriz[i,j].Status;

 end;
//---------------------------------------------------------------------------
 function TFormPrincipal.selecionaVencedor(Matriz : TSquareMatriz; var VNeuronios : ArrayNeuronios; modo:boolean) : Integer;
 var
  aux_indice,i : Integer;
  aux : extended;
  begin
     FormPrincipal.matriz2Vetor(MatrizSquares,VNeuronios[0]);
     aux_indice := 1;
     aux := Distancia(VNeuronios[0],VNeuronios[1]); //www.schulers.com/jpss
     for i := 2 to 10 do
        begin
          if(aux > Distancia(VNeuronios[0],VNeuronios[i])) then
            begin
               aux_indice := i;
               aux := Distancia(VNeuronios[0],VNeuronios[i]);
            end;
        end;
     //cf
     if modo= true then
        begin
           i:= procuraZero(VNeuronios,aux_indice);
           if i<>aux_indice then
                 aux_indice:=i;
        end;
     //cf
     selecionaVencedor := aux_indice;
   end;
//---------------------------------------------------------------------------
  //cf
  function TFormPrincipal.procuraZero(var VNeuronios : ArrayNeuronios;indice:integer):integer;
  var
    j,i : Integer;
    zero : boolean;
  begin
     i:=indice-1;
     zero:=false;
     while (zero<>true)and(i<10) do
        begin
           inc(i);
           zero:= true;
           for j:= 1 to 25 do
              if Neuronios[i][j] <> 0 then
                 zero:= false;
        end;
```

```
      procuraZero:= i;
  end;
//---------------------------------------------------------------------------
  procedure TFormPrincipal.resetaVetor(var VNeuronios : ArrayNeuronios);
  var
    i,j : Integer;
  begin
     for i := 0 to 10 do
        for j := 1 to 25 do
            Neuronios[i][j] := 0;

  end;
//---------------------------------------------------------------------------
  procedure TFormPrincipal.inicializaRandomico(var VNeuronios : ArrayNeuronios);
  var
    i,j : Integer;
  begin
     randomize;
     for i := 1 to 10 do
        for j := 1 to 25 do
            Neuronios[i][j] := random(2);

  end;
//---------------------------------------------------------------------------
  {
  procedure TFormPrincipal.mostraNeuronios;
  var
    i,j :Integer;
    temp : String;
  begin
    Memo1.Clear;
    for i := 1 to 10 do
       begin
         temp := '';
         for j := 1 to 25 do
           begin
              temp := temp + floattostr(Neuronios[i][j]);
           end;
         Memo1.Lines.Add(temp);
       end;

  end;
  }
  //---------------------------------------------------------------------------
  procedure TFormPrincipal.salvaArquivo(FileName : String);

  var
    arquivo : File of ArrayNeuronios;
  begin

    assignFile(arquivo,FileName);
    rewrite(arquivo);
    write(arquivo,Neuronios);
    closefile(arquivo);

  end;
//---------------------------------------------------------------------------
  procedure TFormPrincipal.abreArquivo(FileName : String);

  var
    arquivo : File of ArrayNeuronios;
```

```
  begin

     if(FileExists(FileName)) then
       begin
        assignfile(arquivo,FileName);
        reset(arquivo);
        read(arquivo,Neuronios);
        closefile(arquivo);
       end
     else
       showmessage('Arquivo não encontrado...');

  end;
//---------------------------------------------------------------------------
  procedure TFormPrincipal.FormCreate(Sender: TObject);
  begin
     novaImagem(MatrizSquares);
     defineStatusBotoes;
     //inicializaRandomico(Neuronios);
     resetaVetor(Neuronios);
     n:=1;
  end;
//---------------------------------------------------------------------------
procedure TFormPrincipal.RadioGroup1Click(Sender: TObject);
begin
      FormPrincipal.defineStatusBotoes;
end;
//---------------------------------------------------------------------------
procedure TFormPrincipal.Sair1Click(Sender: TObject);
begin
   Application.Terminate;
end;
//---------------------------------------------------------------------------
procedure TFormPrincipal.BotAprenderClick(Sender: TObject);
var
  i,aux_indice : Integer;

begin
      //if n>0.1 then n:= n-n*0.1; //cf
      aux_indice := selecionaVencedor(MatrizSquares,Neuronios,true);
      for i := 1 to 25 do
         begin //cf
             Neuronios[aux_indice][i] :=  Neuronios[aux_indice][i] + n*(Neuronios[0][i]-Neuronios[aux_indice][i]);
             { if aux_indice>1 then//cf
                   Neuronios[aux_indice-1][i] :=  Neuronios[aux_indice-1][i] + (n*0.5)*(Neuronios[0][i]-Neuronios[aux_indice-
1][i]);//cf
             if aux_indice<9 then//cf
                   Neuronios[aux_indice+1][i] :=  Neuronios[aux_indice+1][i] + (n*0.5)*(Neuronios[0][i]-Neuronios[aux_indice+1]
[i]);//cf
             }
         end; //cf
      //mostraNeuronios;
end;

//---------------------------------------------------------------------------
procedure TFormPrincipal.BotClassificarClick(Sender: TObject);

begin
     edit1.Text := IntToStr(selecionaVencedor(MatrizSquares,Neuronios,false)-1);
end;
//---------------------------------------------------------------------------
procedure TFormPrincipal.SalvarComo1Click(Sender: TObject);
```

```
var
   NomeArq : String;
begin
   if SaveDialog1.Execute then
      begin
         NomeArq := SaveDialog1.FileName;
         if(length(NomeArq) > 0) then
            begin
              salvaArquivo(NomeArq);
            end;
      end;
end;
//---------------------------------------------------------------------------

procedure TFormPrincipal.Abrir1Click(Sender: TObject);

var
  NomeArq : String;

begin
     if OpenDialog1.Execute then
       begin
          NomeArq := OpenDialog1.FileName;
          abreArquivo(NomeArq);
       end;
end;
//---------------------------------------------------------------------------
end.
```