# Implementation of Competitive Learning Networks for WEKA

## Prefácio

Em um artigo anterior mostramos que usando WEKA um pesquisador pode facilmente implementar seus próprios algoritmos sem outras preocupações técnicas como vincular um algoritmo a uma GUI ou mesmo carregar os dados de um arquivo/database, uma vez que estas tarefas e muitas outras são tratadas transparentemente pelo framework WEKA[1].

Neste artigo estudaremos Redes de Aprendizado competitivo e mais especificamente as arquiteturas de Aprendizado por Quantização Vetorial e os Mapas Auto Organizáveis. Finalmente iremos descrever a implementação destas redes para o WEKA [2].

## 1-Redes de Aprendizado Competitivo

Aprendizado competitivo é normalmente implementado com Redes Neurais Artificiais, que contêm uma camada Escondida (Hiden Layer) que é comumente chamada de "Camada Competitiva" (Figura 1). Cada neurônio competitivo i é descrito por um vetor de pesos e calcula a medida de similaridade entre o dado de entrada e outro vetor de pesos . [3]

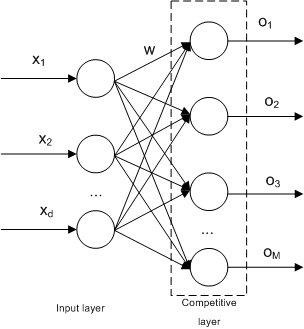


Figura 1: Arquitetura de uma rede neural competitiva.

Para cada vetor de entrada, os neurônios competitivos "competem" entre si para se tornar o neurônio vencedor. O neurônio vencedor m define sua saída e todos os outros neurônios competidores definem suas saídas [3].

### 1.2- Aprendizado por Quantização Vetorial (LVQ)

O algoritmo para Aprendizado por Quantização Vetorial pode ser facilmente implementado utilizando uma rede neural com uma camada competitiva contendo um número de neurônios competitivos igual ao numero de clusters. Cada neurônio competitivo i corresponde em um cluster e seu vetor de pesos corresponde ao centróide do cluster [3].

O algoritmo é repetitivo e requer que a inicialização dos vetores de pesos da rede. Em cada repetição ou época, um vetor é apresentado como entrada para a rede e as distâncias de cada centróide são calculadas e finalmente é selecionado o neurônio vencedor com o menor valor de distância Euclidiana . O passo final é atualizar os vetores de pesos "movendo" o centróide do neurônio vencedor " mais perto" do vetor de entrada .

A quantidade de "movimento" depende de um parâmetro (taxa de aprendizagem).

- Algoritmo de agrupamento LVQ:

1. Define the number of clusters .

2. Initialize the centroids .

3. Initialize learning rate , epochs counter and repetitions counter .

4. For every epoch do the following steps for

- Set vector as the Neural Network’s input.

- Select the winner neuron m.

- Update the weight vector for the winner neuron [Function]

- .

5. Check for termination. If not set and return to step 4.

### 1.3-Mapas Auto Organizáveis

A rede de Mapas Auto Organizáveis (SOM) é outro algoritmo competitivo muito comum que foi introduzido por T. Kohonen [4] em uma tentativa de modelar o processo de auto-organização do cérebro humano.

Uma rede SOM consiste em uma camada de entrada e uma camada contendo os neurônios competitivos que estão dispostos em uma estrutura de duas dimensões (Figura 2)[5]. Cada um destes neurônios é descrito como um vetor de pesos . Quando um vetor de entrada é apresentado à rede, a grade de neurônios compete e o vencedor m é selecionado com seu vetor de pesos tendo a maior similaridade com . Então é possível argumentar que uma rede SOM é uma função de mapeamento de uma entrada d-dimensional para uma grade bidimensional [3].

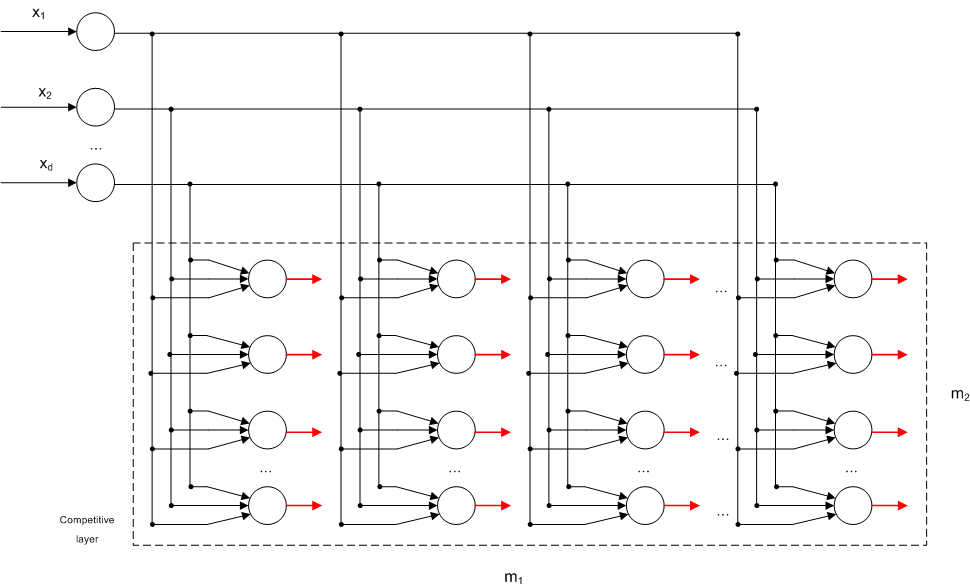


Figura 2: Uma rede SOM com d entradas e uma grade 2d m1 x m2

#### 1.3.1 Algoritmo de Treinamento SOM:

O algoritmo começa inicializando os vetores de pesos a pequenos valores randômicos produzidos por um gerador de números randômicos. Depois deste passo há três estágios principais, que podem ser resumidos como se segue [3][5]:

* Competição: Para cada amostra de treinamento , a grade de neurônios calcula o valor da função de similaridade (?). O neurônio com maior similaridade se torna o neurônio vencedor. Para a função de Similaridade é utilizada a distância Euclidiana entre o vetor de entrada e os vetores de pesos para cada neurônio competidor.
* Cooperação: O neurônio vencedor define na grade sua vizinhança topológica. Os neurônios que pertencem àquela vizinhança irão todos atualizar seus pesos para a dada entrada. Uma questão básica tem a ver com a definição topológica da vizinhança. Vamos definir como o vizinho topológico, com seu centro no neurônio vencedor . A vizinhança topológica consiste em um conjunto de neurônios, e é um neurônio aleatório neste conjunto. Nós também definimos como a distância entre o neurônio vencedor e o neurônio . Podemos agora assumir que o vizinho topológico é uma função da dj,i que satisfaz dois critérios: [5].
  + É simétrico ao ponto que a função tem seu valor mais elevado e para o qual é , ou em outras palavras para o ponto do neurônio vencedor. (?)
  + A amplitude da função diminui sucessivamente, assim como a distância do neurônio vencedor aumenta, aproximando de zero quando a distancia tende ao infinito.   
    Uma função que satisfaz os critérios acima é a função Gaussiana:

(1)

Onde 'o' é o valor inicial da largura efetiva da vizinhança topológica, que determina a extensão em que cada neurônio na vizinhança topológica participa no processo de aprendizagem. Este parâmetro é decrescido exponencialmente em cada época n de acordo com a seguinte fórmula: [5]

(2)

- Adaptação Sináptica: Neste último estágio tomam lugar os ajustes do vetor de pesos para os neurônios na grade, de acordo com a seguinte equação: [5]

(3)

Finalmente, dado um vetor de pesos Wj(n) para a época n, é facilmente possível calcular o novo vetor para a época n+1 utilizando a equação: [5]

(4)

Como podemos ver na última fórmula, a taxa de aprendizagem N(n) é também dependente do tempo (época). Para ser mais específico, a taxa de aprendizagem deve ser inicializada com um valor N0 e diminuir exponencialmente com o aumento do contador do tempo (época) n: [5]

(5)

Os procedimentos descritos acima podem ainda ser divididos em duas fases: [4]

- Fase de Ornamento: É a primeira fase na qual é realizado o arranjo topológico dos vetores de pesos dos neurônios na camada competitiva. Nesta fase a taxa de aprendizagem deve iniciar em um valor próximo a 0.1 e decrescer gradualmente para 0.01. Estes valores podem ser produzidos através da atribuição dos seguintes valores na equação (5): [5]

(6)

Além disto, a função da vizinhança topológica deve conter inicialmente todos os neurônios na grade com o centro sendo o neurônio vencedor e reduzir gradualmente para conter poucos neurônios ou apenas o neurônio vencedor. No caso de uma grade bidimensional, podemos definir o valor da largura inicial igual à largura do raio da grade e também o definir o valor de na equação (2) igual a: [5]

(7)

- Fase de Convergência: É a fase em que os pesos recebem seus valores finais, se ajustando ainda mais aos dados de entrada. Nesta fase o número de repetições (épocas) depende das dimensões dos vetores de entrada e têm uma regra de ouro onde deve ser no mínimo 500 vezes o número de neurônios da rede. A taxa de aprendizagem deve assumir valores perto de 0.01 e finalmente, a função topológica da vizinhança deve incluir apenas os neurônios mais próximos ao neurônio vencedor, mas ainda podem acabar incluindo apenas o neurônio vencedor. [5]

Para resumir os descritivos/as descrições acima, o algoritmo de aprendizagem SOM fica como o seguinte:

Algoritmo de treinamento SOM (vizinhança contínua)

1. Calcula o número de neurônios na camada competitiva M=m1\*m2

2. Inicializa os centróides .

3. Inicializa a taxa de aprendizagem η(0), o parâmetro σ(0), o contador de épocas k=0 e o contador de repetições κ=0.

4. Para cada época k faça os seguintes passos: para

- Defina o vetor como a entrada da Rede Neural.

- Selecione o neurônio vencedor m.

- Atualize os vetores de pesos para todos os neurônios na vizinhança do neurônio vencedor:

(8)

(9)

-

5. Gradualmente decrescer a taxa de aprendizagem N(k)

6. Gradualmente decrescer a largura efetiva da vizinhança o(k)

7. Verificar o término. Se não, definir k=k+1 e retornar ao passo 4

## 2- Implementação das redes LVQ e SOM no WEKA

### 2.1-Rede SOM

A rede SOM foi implementada de acordo com a descrição do parágrafo 1.3. Para o calculo da taxa de aprendizagem e a largura efetiva da vizinhança na fase de ordenação, em cada época foram usadas as fórmulas (5) e (2) de acordo com os valores dos parâmetros:

E

Onde n0 é o número de épocas na fase de ordenação, N0 a taxa de aprendizado inicial e o0 a largura efetiva inicial que foi dada pela seguinte fórmula:

Onde e são a largura e altura de acordo com uma grade bidimensional.

Na fase de convergência, os valores da taxa de aprendizado e largura efetiva da vizinhança permanecem, portanto constantemente iguais a 0.01 e 0.0001.

## Referências

[1] J.Salatas, “Extending Weka”, ICT Research Blog, August 2011.

[2] “WEKA Neural Networks Algorithms”, last accessed: 23/08/2011

[3] Α. Λύκας, “Τεχνητά Νευρωνικά Δίκτυα – Εφαρμογές”, Τεχνητή Νοημοσύνη Εφαρμογές, Τόμος Β, Ελληνικό Ανοικτό Πανεπιστήμιο, 2008.

[4] T. Kohonen, “Self-Organization and Associative memory”, 3rd Edition, Springer 1989.

[5] S. Haykin, “Neural Networks and Learning Machines”, 3rd Edition, Pearson Education, 2008.

[6] “How do I use the package manager?”, Weka wiki, last accessed: 24/08/2011