5 TREINAMENTO NÃO SUPERVISIONADO

5.1-Redes auto organizadas

Até agora vimos redes em que o treinamento consistia na apresentação de padrões de dados que continham os dados da entrada e os dados desejáveis na saída. A partir destes padrões, as redes calculam a diferença entre o que é desejável e o que realmente aparece na saída e, a partir desta diferença ou erro, ajustam os pesos de forma a minimizar o erro, tanto para estes padrões de treinamento quanto para outros padrões que venham a ser apresentados após o treinamento. A este tipo de treinamento chamamos de supervisionado, na medida em que é necessário mostrar qual é a saída desejada.

Ocorre que em muitas aplicações não é possível ter exemplos de treinamento neste estilo (entradas e saídas), estando disponíveis somente dados de entrada. Por exemplo, em problemas de "clusterização" em que não conhecemos as classes dos nossos dados e desejamos que a rede separe os dados de entrada em grupos (*clusters*) ou em problemas de extração de caraterísticas em que desejamos saber quais as características básicas dos grupos de dados que possuímos.

Para estes casos, não havendo um professor que ensine à rede quais são as saídas, ela deve ser capaz, geralmente através de um processo de competição entre os nós, de mapear as entradas disponíveis em uma saída que geralmente possui uma dimensionalidade menor. Este processo é conhecido como aprendizado auto-organizado.

Há muitos tipos de redes com aprendizado auto-organizado, aplicáveis a uma área vasta área de problemas. A rede de *aprendizagem competitiva* proposta por Rumelhart e Zipser (1985) as redes ART propostas por Carpenter and Grossberg (1987), as redes cognitron de Fukushima (1975 e 1988) ou os Mapas Auto Organizáveis de Kohonen.

5.2-Aprendizado competitivo

O aprendizado competitivo é um algoritmo que divide uma série de dados de entradas em grupos (*clusters*) que são inerentes aos dados de entrada. A informação é extraída sem que haja um par entrada/saída alvo.

As redes para este tipo de problema possuem uma camada de nós de saída que estão ligados a uma só camada (de entrada, portanto), de tal forma que existam tantos nós na entrada quantas sejam as características dos padrões de entrada que possuímos e tantos nós de saída quantos sejam os *clusters* em que queiramos classificar as entradas.

O treinamento promove uma competição entre os nós da rede, de forma que somente um dos nós da saída seja ativado, para cada padrão de entrada apresentado. Este nó é justamente o nó representativo do *cluster* ao qual pertence aquele padrão de entrada. Por exemplo, podemos ter uma arquitetura como a mostrada na figura 18.

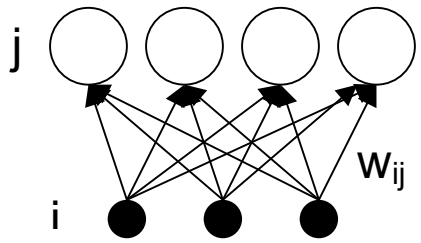


FIG. 18- REDE COMPETITIVA - SOMENTE UMA SAÍDA j É ACIONADA PARA CADA VETOR DE ENTRADA X

Para cada padrão (vetor) de entrada x, apresentado à rede, somente um dos nós da saída, chamado de nó vencedor, será ativado. Em uma rede já treinada, todos os vetores x de entrada que pertencerem a um mesmo *cluster*, ou seja, que tiverem características parecidas de tal forma que possam ser identificados como pertencentes a um mesmo *cluster*, acionarão o mesmo nó de saída (representativo do *cluster*, por conseguinte).

A idéia básica é que, durante o treinamento, a cada padrão apresentado, o algoritmo descubra qual dos nós da saída melhor representa aquele padrão de entrada (através de algum

tipo de distância entre os vetores de entrada e os nós de saída) e ajuste os pesos que ligam a entrada a este nó de saída. Isto faz com que o nó escolhido fique ainda mais representativo deste padrão, ou seja, diminui a "distância" que separa este padrão do nó de saída. Ao longo do treinamento, o nó de saída vai se tornando o representante típico dos padrões de entrada que o tornaram vencedor, ou seja, aquele nó passa a representar o *cluster* daquele grupo de padrões de entrada.

Para determinar a distância que existe entre o vetor de entrada (padrão de entrada) e cada um dos nós de saída são usados basicamente dois métodos:

Produto escalar - este método pressupõe que tanto o vetor de entrada x quanto o vetor de pesos w_i que liga o nó i às entradas, estejam normalizados para o valor unitário (isto é, o comprimento destes vetores deve ser 1). É calculado um valor de ativação a_i do nó i que corresponde ao produto escalar dos dois vetores. Uma vez que tenham sido calculados os valores de ativação a_i de cada nó i, é escolhido o nó k com o maior valor de ativação como o nó vencedor. Somente os pesos que ligam a entrada a este nó serão atualizados.

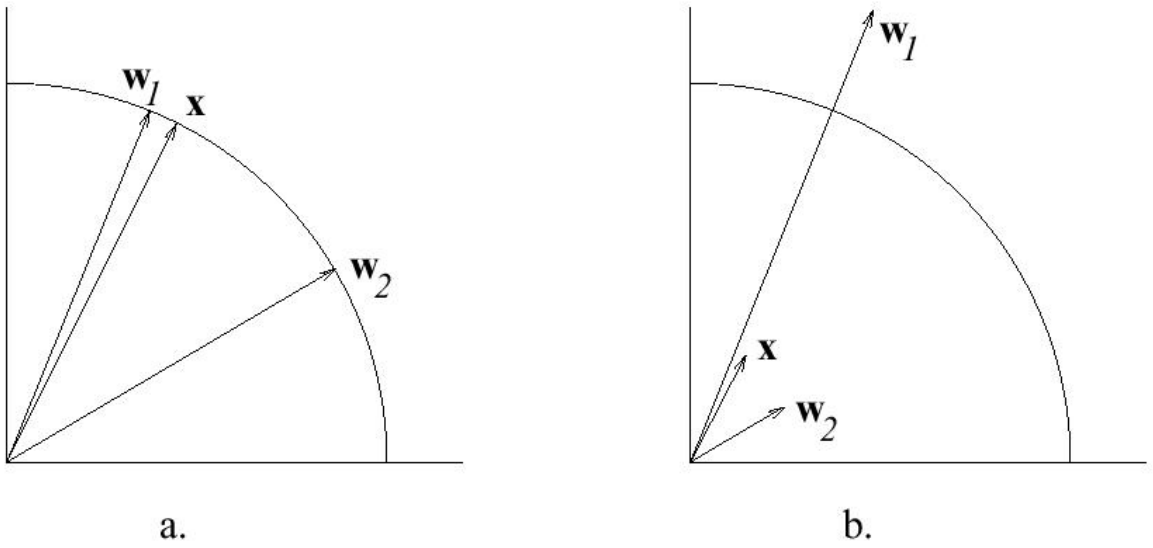


FIG. 19- PRODUTO INTERNO EXIGE QUE VETORES ESTEJAM NORMALIZADOS PARA SER PROPORCIONAL À DISTÂNCIA

A necessidade dos vetores estarem normalizados é demonstrada graficamente na figura 19. Note que o produto escalar entre dois vetores é também um vetor (perpendicular

ao plano dos dois vetores), cujo comprimento será tão maior quanto mais próximos estiverem os vetores originais (desde que sejam do mesmo tamanho), já que o produto escalar é definido como: $|x||w|\cos\theta$, onde θ é o ângulo entre os vetores. Assim, na figura 19-a, como os vetores são de mesmo tamanho, o produto escalar de x por w1 será maior do que o produto escalar de x por w2, dados que a norma (tamanho) dos vetores é 1 e que o coseno aumenta quando o ângulo se aproxima de zero. Já na figura 19-b, os vetores não estão normalizados e o produto escalar dos vetores não é proporcional somente ao coseno mas também à norma dos vetores. Assim, o produto escalar maior não corresponde necessariamente ao menor ângulo entre os vetores.
Uma representação gráfica do que acontece durante o treinamento, pode ser vista na figura 20.

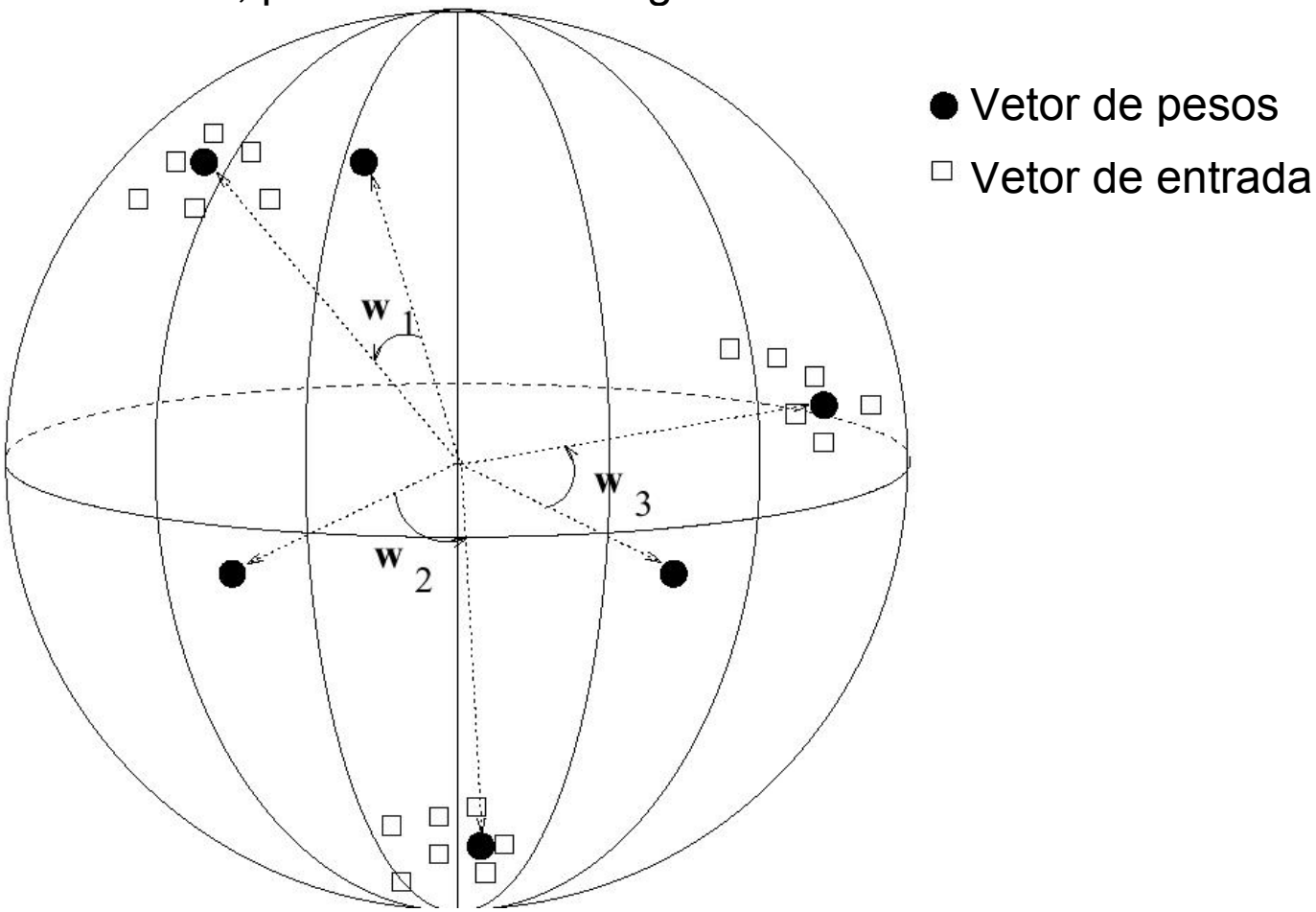


FIG. 20- VETORES DE PESOS E DE ENTRADA - VETORES DE PESOS TENDEM A SE ENCAMINHAR AO CENTRO DO CLUSTER

Para entende-la, é útil relembrar que tanto o vetor de pesos w_i, quanto o vetor de entrada x, possuem a mesma dimensão (no caso, dimensão três) e estão normalizados (ou seja seu comprimento é 1). Assim, se os padrões de entrada possuem três características, a dimensão da entrada é 3 (três nós de entrada), o vetor de pesos também possui dimensão três e os vetores são representáveis no espaço tridimensional, dentro de uma esfera de raio 1 (pois o comprimento de todos os vetores é 1). A figura 20 mostra os vetores de pesos representados por uma bola preta e os vetores de entrada (padrões), representados como um quadrado.

Ao longo do treinamento a atualização dos pesos vai rodando os vetores de pesos e fazendo com que cada vetor de pesos w_i se aproxime do centro do grupo de vetores de entrada para os quais o respectivo nó i sai vencedor. Ou seja, o vetor de pesos é o típico representante daquele *cluster* (grupo de vetores de entrada).

Distância Euclidiana - este outro método de determinar qual o vetor k de pesos mais próximo ao vetor de entrada, usa a distância Euclidiana entre os vetores, que é dada por $||w_k-x||$. Pode também ser usado o quadrado da distância Euclidiana: $(w_k-x)^2$. O vetor w_k com a menor distância Euclidiana, ganha a disputa. Da mesma forma, somente este vetor será ajustado, segundo a fórmula 5.1 (η é o coeficiente de aprendizagem).

$$w_k(novo) = w_k + \eta(x - w_k) \qquad (5.1)$$

Este método é, em geral, mais usado que o do produto escalar.

5.3-Redes de Kohonen

As redes de Kohonen, também conhecidas como Mapas Auto Organizáveis de Kohonen foram propostas por Teuto Kohonen em 1984 e são um procedimento de aprendizado não supervisionado. A arquitetura é constituída de uma só camada de nós, geralmente bidimensional, ligada ao vetor de nós da entrada (figura 21). O conceito novo introduzido é que o

comportamento de um determinado nó é diretamente afetado pelo comportamento dos nós vizinhos (vizinhança local).

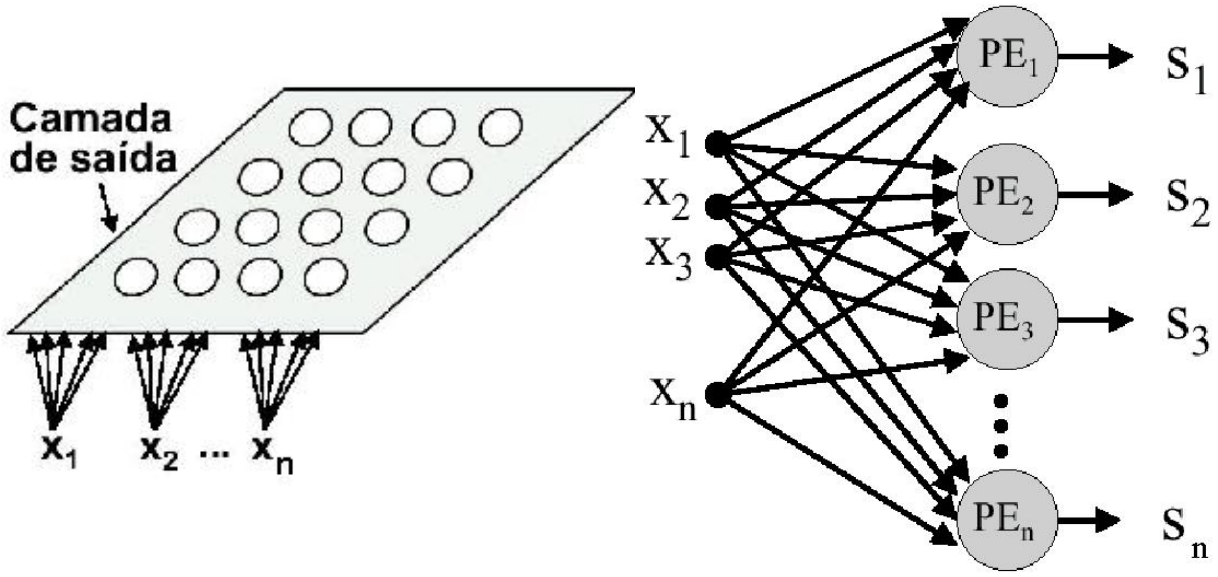


FIG. 21- ARQUITETURA DA REDE DE KOHONEN

Cada neurônio está portanto conectado a cada componente de entrada através de um vetor de pesos cuja dimensão é a mesma da do vetor de entrada.

Durante o treinamento os pesos alterados são não só os do neurônio vencedor mas também os pesos dos neurônios de sua vizinhança local. Padrões que estejam próximos no espaço original (tenham características similares) acionarão um mesmo neurônio ou acionarão neurônios próximos na malha de Kohonen. Desta forma é preciso definir quais os modos de medir distância na malha, ou seja definir qual a vizinhança que será afetada durante o treinamento. Duas formas são as mais comuns: distâncias de Manhatan e de Grid. Vamos considerar 2 neurônios em uma malha bidimensional N1 e N2, com coordenadas (i,j) e (k,h). $Dist_{Manhatan}(N1,N2) = abs(h-j) + abs(k-i)$

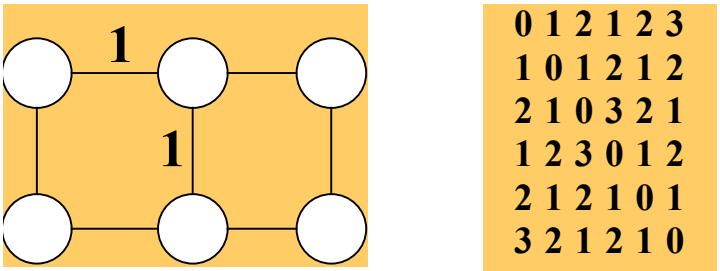
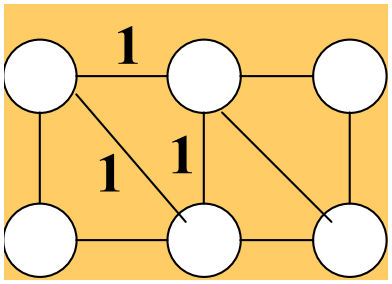


FIG. 22- DISTÂNCIA DE MANHATAN EM UMA MALHA 2 X 3 (NEURÔNIOS ORDENADOS PELAS LINHAS)

DistGrid(N1,N2) = Max{ (h-j) , (k-i)}



0	1	2	1	1	2
1	0	1	1	1	1
2	1	0	2	1	1
1	1	2	0	1	2
1	1	1	1	0	1
2	1	1	2	1	0

FIG. 23- DISTÂNCIA DE GRID EM UMA MALHA 2 X 3 (NEURÔNIOS ORDENADOS PELAS LINHAS)

1) Escolha aleatoriamente q vetores de pesos iniciais.
2) Apresente a entrada x(k) (k = 0 na primeira iteração).
3) Calcule a distância de x(k) a cada um dos q vetores de pesos $w_j(k)$, j=1. . .q, e nomeie como neurônio vencedor aquele para o qual $(w_j(k) - x(k))^2$ ou $||w_j(k) - x(k)||$ é minimizado.
4) Escolha uma vizinhança $V_i(n)$ para o neurônio vencedor i.
5) Atualize os pesos de todos os neurônios de $V_i(n)$:

$$w_j(n+1) = \begin{cases} w_j(n) + \eta(n)[x(n) - w_j(n)] & j \in V_i(n) \\ w_j(n) & caso\ contrario \end{cases}$$

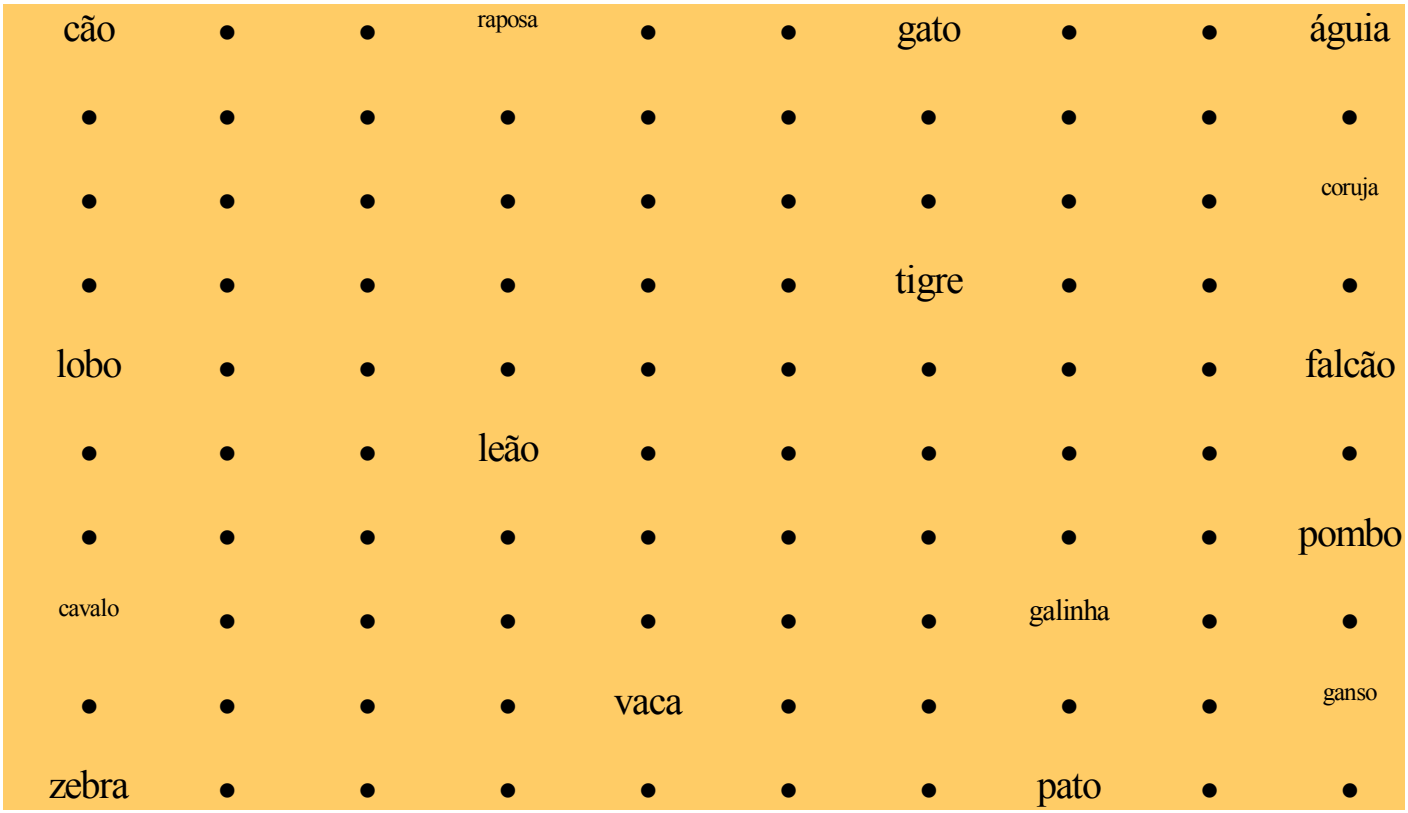
onde η é a taxa de aprendizado, em geral decrescente ao logo do treinamento (a vizinhança $V_i(n)$ também pode diminuir).
6) Retorne ao passo 2 até a convergência.

A entrada da rede de Kohonen é um conjunto de vetores de padrões. Por exemplo, poderíamos ter 10 mil padrões de clientes bancários, cada um formado por um vetor de tamanho 5, contendo, para cada cliente, informações de: idade; imóvel próprio ou não; CEP; profissão; e, renda. Se escolhermos uma malha bidimensional 4X4, teremos 16 neurônios na malha, cada um deles ligado ao vetor de entradas através de um vetor de pesos de tamanho 5. Ao final da fase de treinamento, poderemos atribuir 1 dos 16 neurônios da malha a cada um dos 10 mil padrões.

Outro exemplo poderia ser um vetor de 13 características de animais e padrões de vetores correspondentes a diversos animais, dos quais alguns estão na tabela abaixo:

Animal →	pombo	galinha	pato	ganso	falcão	coruja
Pequeno	1	1	1	1	1	1
Médio	0	0	0	0	0	0
Grande	0	0	0	0	0	0
2 pernas	1	1	1	1	1	1
4 pernas	0	0	0	0	0	0
Pêlo	0	0	0	0	0	0
Chifre	0	0	0	0	0	0
Crina	0	0	0	0	0	0
Penas	1	1	1	1	1	1
Caça	0	0	0	0	1	1
Corre	0	0	0	0	0	0
Voa	1	0	0	1	1	1
Nada	0	0	1	1	0	0

A classificação de uma rede de kohonen, usando uma malha de 10x10, foi calculada como:



5.4-Exemplos com rede de Kohonen

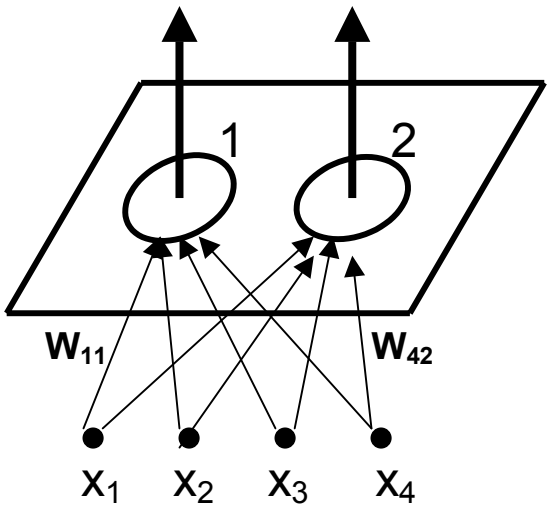
Vamos a um exemplo de treinamento com rede de Kohonen destinado a classificar 4 vetores, com quatro características cada, em dois *clusters* diferentes. Os vetores são:

a =(1,1,0,0) b=(0,0,0,1) c=(1,0,0,0) d=(0,0,1,1)

A taxa de aprendizado η começará com 0,6 e será cortada pela metade a cada iteração, isto é, $\eta(t+1) = \eta(t)/2$

O treinamento se encerrará quando a taxa de treinamento cair abaixo de 0,01 ou o erro médio para os quatro vetores cair também abaixo de 0,01 (estes critérios são arbitrários).

Como queremos classificar em dois *clusters*, vamos usar, para efeito de simplificar as contas, uma arquitetura bem simples com apenas dois nós (1 e 2) e uma vizinhança de raio zero, isto é, somente o nó vencedor terá seus pesos alterados para cada padrão apresentado.



Iniciamos os pesos, aleatoriamente, com os valores:

W_1 = (0,2 0,6 0,5 0,9) e W_2 = (0,8 0,4 0,7 0,3)

Que formam a matriz de pesos:

$$W = \begin{pmatrix} 0,2 & 0,8 \\ 0,6 & 0,4 \\ 0,5 & 0,7 \\ 0,9 & 0,3 \end{pmatrix}$$

Calculando as distâncias dos vetores de entrada aos vetores de pesos de cada nó, teremos, para o vetor a=(1,1,0,0):

$D_1 = (0,2 - 1)^2 + (0,6 - 1)^2 + (0,5 - 0)^2 + (0,9 - 0)^2 = 1,86$

$D_2 = (0,8 - 1)^2 + (0,4 - 1)^2 + (0,7 - 0)^2 + (0,3 - 0)^2 = 0,98$

Assim, o vetor *a* está mais próximo do nó 2, que é, portanto, o nó vencedor. Atualizando os pesos do nó vencedor, temos:

$W_{12}(t) = W_{12}(t-1) + 0,6(1-0,8) = 0,8 + 0,12 = 0,92$

$W_{22}(t) = W_{22}(t-1) + 0,6(1-0,4) = 0,4 + 0,36 = 0,76$

$W_{32}(t) = W_{32}(t-1) + 0,6(0-0,7) = 0,7 - 0,42 = 0,28$

$W_{42}(t) = W_{32}(t-1) + 0,6(0-0,3) = 0,3 - 0,18 = 0,12$

Os novos valores dos pesos, são, portanto:

$W_1 = (0,2 \;\; 0,6 \;\; 0,5 \;\; 0,9)$ e $W_2 = (0,92 \;\; 0,74 \;\; 0,28 \;\; 0,12)$

E a matriz é, portanto:

$$W = \begin{pmatrix} 0,2 & 0,92 \\ 0,6 & 0,74 \\ 0,5 & 0,28 \\ 0,9 & 0,12 \end{pmatrix}$$

Para o vetor b=(0,0,0,1), temos:

$D_1 = (0,2 - 0)^2 + (0,6 - 0)^2 + (0,5 - 0)^2 + (0,9 - 1)^2 = 0,66$

$D_2 = (0,92 - 0)^2 + (0,76 - 0)^2 + (0,28 - 0)^2 + (0,12 - 1)^2 = 2,27$

Para este caso, o vetor *b* está mais próximo do nó 1, que é, portanto, o nó vencedor. Atualizando os pesos do nó vencedor:

$W_{11}(t) = W_{11}(t-1) + 0,6(0-0,2) = 0,2 - 0,12 = 0,08$

$W_{21}(t) = W_{21}(t-1) + 0,6(0-0,6) = 0,6 - 0,36 = 0,24$

$W_{31}(t) = W_{31}(t-1) + 0,6(0-0,5) = 0,5 - 0,30 = 0,20$

$W_{41}(t) = W_{31}(t-1) + 0,6(1-0,9) = 0,9 + 0,06 = 0,96$

Os novos valores dos pesos, são, portanto:

$W_1 = (0,08 \;\; 0,24 \;\; 0,20 \;\; 0,96)$ e $W_2 = (0,92 \;\; 0,74 \;\; 0,28 \;\; 0,12)$

E a matriz de pesos é:

$$W = \begin{pmatrix} 0,08 & 0,92 \\ 0,24 & 0,74 \\ 0,20 & 0,28 \\ 0,96 & 0,12 \end{pmatrix}$$

Para o vetor c=(1,0,0,0), temos:

$D_1 = 1,86$

$D_2 = 0,67$

O nó 2 será o vencedor e os novos pesos serão:

W_1=(0,08 0,24 0,20 0,96) e W_2=(0,968 0,304 0,112 0,048)

$$W = \begin{pmatrix} 0,08 & 0,968 \\ 0,24 & 0,304 \\ 0,20 & 0,112 \\ 0,96 & 0,048 \end{pmatrix}$$

Para o vetor d=(0,0,1,1), temos:

$D_1 = 0,70$

$D_2 = 2,72$

O nó 1 será o vencedor e os novos pesos serão:

W_1=(0,032 0,096 0,680 0,984) e

W_2=(0,968 0,304 0,112 0,048),

com a matriz de pesos sendo:

$$W = \begin{pmatrix} 0,032 & 0,968 \\ 0,096 & 0,304 \\ 0,680 & 0,112 \\ 0,984 & 0,048 \end{pmatrix}$$

No ciclo seguinte a taxa de treinamento η será de 0,6 / 2 = 0,3.

A aplicação do segundo ciclo de treinamento, trará:

W_1=(0,016 0,047 0,630 0,999) e

W_2=(0,980 0,360 0,155 0,024)

Assim, temos a matriz de pesos:

$$W = \begin{pmatrix} 0,016 & 0,980 \\ 0,047 & 0,360 \\ 0,630 & 0,155 \\ 0,999 & 0,024 \end{pmatrix}$$

Após 100 épocas, o treinamento converge para

W_1=(0,0 0,0 0,5 1,0) e

W_2=(1,0 0,5 0,0 0,0)

Ou seja, a matriz de pesos fica com:

$$W = \begin{pmatrix} 0,0 & 1,0 \\ 0,0 & 0,5 \\ 0,5 & 0,0 \\ 1,0 & 0,0 \end{pmatrix}$$

Com esses pesos, a rede classifica os vetores *a* e *b* no *cluster* 1 e *c* e *d* no *cluster* 2.

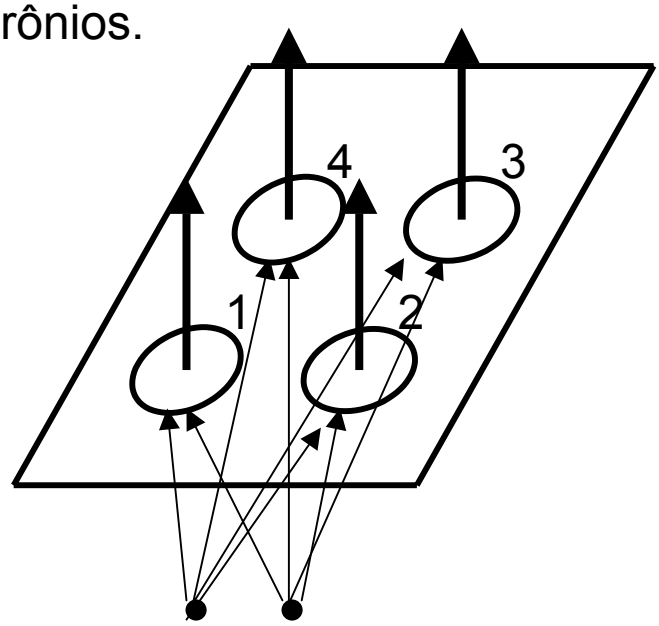
Outro exemplo de rede de Kohonen

Suponha uma rede simples para classificar veículos. Como o exemplo é simples, vamos considerar a vizinhança sendo o próprio neurônio, duas iterações e fazer $\eta = 0.8$ (fixo).

Como características serão usadas número de rodas e a existência ou não de motor. O valor 1 indica a existência de motor enquanto 0 indica a ausência. Vejamos os padrões de entrada para os dois veículos que usaremos no treinamento:

	Rodas	Motor
Bicicleta	2	0
Carro	4	1

Como há duas características, a camada de entrada da rede consistirá de dois neurônios. Na camada de saída serão utilizados quatro neurônios.



Aleatoriamente os pesos são inicializados para:

neurônio 1 = { w11, w21} = {1, 2}

neurônio 2 = { w12, w22} = {2, 2}

neurônio 3 = { w13, w23} = {1, 3}

neurônio 4 = { w14, w24} = {3, 2}

PRIMEIRA ITERAÇÃO

Apresentado as característica da bicicleta {2,0} e calculando as distâncias:

d1 = (2-1)2 + (0-2)2 = 5

d2 = (2-2)2 + (0-2)2 = 4

d3 = (2-1)2 + (0-3)2 = 10
d4 = (2-3)2 + (0-2)2 = 5
O neurônio vencedor é o segundo (d2). Ajustando seu peso:
w12 = w12 + 0.8(x1(t) - w12(t)) = 2 + 0.8.(2-2) = 2
w22 = w22 + 0.8(x2(t) - w22(t)) = 2 + 0.8.(0-2) = 0.4
Apresentado as características do automóvel {4,1} e calculando as distâncias:
d1 = (4-1)2 + (1-2)2 = 10
d2 = (4-2)2 + (1-0.4)2 = 4.36
d3 = (4-1)2 + (1-3)2 = 13
d4 = (4-3)2 + (1-2)2 = 2
O neurônio vencedor é o quarto (d4). Ajustando seu peso:
w14 = w14 + 0.8(x1(t) - w14(t)) = 3 + 0.8.(4-3) = 3.8
w24 = w24 + 0.8(x2(t) - w24(t)) = 2 + 0.8.(1-2) = 1.2

SEGUNDA ITERAÇÃO
Apresentado as característica da bicicleta {2,0} e calculando as distâncias:
d1 = (2-1)2 + (0-2)2 = 5
d2 = (2-2)2 + (0-0.4)2 = 0.16
d3 = (2-1)2 + (0-3)2 = 10
d4 = (2-3.8)2 + (0-1.2)2 = 4.68
Novamente o neurônio vencedor é o segundo (d2). Ajustando seu peso:
w12 = w12 + 0.8(x1(t) - w12(t)) = 2 + 0.8.(2-2) = 2
w22 = w22 + 0.8(x2(t) - w22(t)) = 0.4 + 0.8.(0-0.4) = 0.08
Apresentado as características do automóvel {4,1} e calculando as distâncias:
d1 = (4-1)2 + (1-2)2 = 10
d2 = (4-2)2 + (1-0.08)2 = 4.8464
d3 = (4-1)2 + (1-3)2 = 13
d4 = (4-3.8)2 + (1-1.2)2 = 0.08
Novamente o neurônio vencedor é o quarto (d4). Ajustando seu peso:
w14 = w14 + 0.8(x1(t) - w14(t)) = 3.8 + 0.8.(4-3.8) = 3.96

w24 = w24 + 0.8(x2(t) - w24(t)) = 1.2 + 0.8.(1-1.2) = 1.04

NOVA ENTRADA

Apresentando as características de uma moto que possui 2 rodas e motor. Como as entradas são mais parecidas à da bicicleta, a rede deve classificar como bicicleta. A isto se dá o nome de generalização. Para classificar como moto, a rede deveria ser treinada para este fim, às vezes sendo necessário outras características de entrada.

Verificando a saída da rede treinada para a entrada da moto:

x = {2,1}

d1 = (2-1)2 + (1-2)2 = 2

d2 = (2-2)2 + (1-0.08)2 = 0.8464

d3 = (2-1)2 + (1-3)2 = 5

d4 = (2-3.8)2 + (1-1.2)2 = 3.8432

Como era de se esperar o neurônio vencedor foi o segundo indicando que o modelo da moto, apresentado à rede, é da classe das bicicletas.