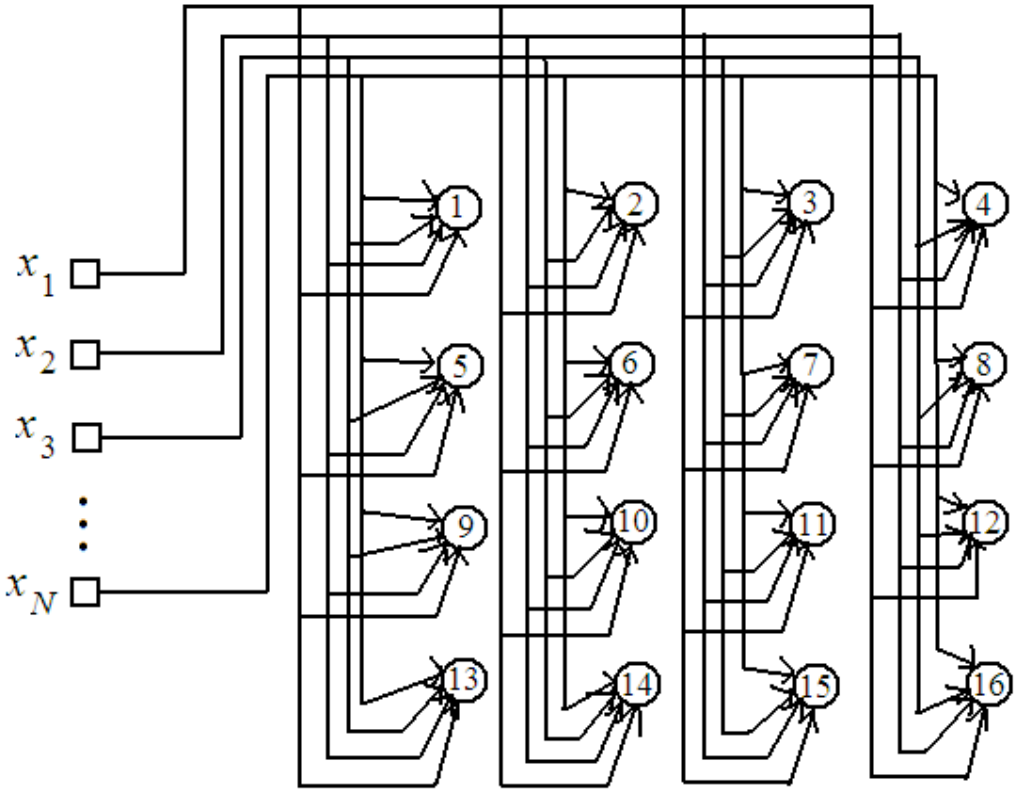
Aprendizado Competitivo e Auto-Organizado

Uma característica importante das redes neurais é a sua capacidade de aprender a partir de estímulos fornecidos pelo meio-ambiente. Nas aulas anteriores estudamos algoritmos de aprendizado supervisionado, em que a cada padrão vindo do ambiente é associado um padrão de resposta desejado, dado por um professor, e o aprendizado da rede consiste em aproximar a relação entrada-saída subjacente aos padrões fornecidos. Nesta aula, vamos estudar um algoritmo para aprendizado não-supervisionado ou auto-organizado cujo objetivo é fazer com que a rede descubra características significantes nos padrões de entrada sem o auxílio de um professor.

Para que um algoritmo de aprendizado não-supervisionado seja implementado ele deve se basear em um conjunto de regras que levem em conta apenas informação local, isto é, que façam com que as mudanças nos pesos sinápticos de um dado neurônio sejam conseqüência apenas de fenômenos que aconteçam nas vizinhanças do neurônio.

O algoritmo de aprendizado auto-organizado que vamos estudar é chamado de Mapa Auto-Organizado de Kohonen (*Self-Organizing Map*, ou SOM, em inglês), desenvolvido pelo finlandês Teuvo Kohonen (1934 -) no começo dos anos 1980.

O algoritmo SOM está baseado no chamado aprendizado competitivo. Uma rede neural típica com aprendizado competitivo é uma rede de uma única camada (uni- ou bi-dimensional) em que todos os neurônios recebem a mesma entrada. Cada neurônio computa o seu nível de ativação multiplicando o seu vetor de pesos pelo vetor de entrada, da maneira usual. O neurônio que tiver o maior nível de ativação é chamado de vencedor e apenas ele terá atividade diferente de zero na saída da rede, ou seja, o padrão de entrada que estiver sendo apresentado à rede provocará a ativação de apenas um neurônio da rede neural.



Um exemplo de uma rede com aprendizado competitivo em duas dimensões está dado acima. Os padrões de entrada são vetores N dimensionais $\boldsymbol{x}_i$, i = 1, ..., N. Vamos supor que existam P desses padrões. A rede neural consiste de M neurônios organizados em uma grade bi-dimensional. No nosso caso, M = 16. Quando um dos P padrões é apresentado na entrada, cada um dos M neurônios recebe este padrão e calcula o seu nível de ativação,

$$u_i = \sum_{k=1}^{N} w_{ik} x_k, \; i = 1, \ldots, M ,$$

onde $\boldsymbol{x}$ é o vetor de entrada, i é o índice que indica o neurônio e $\boldsymbol{w}_i$ é o vetor de pesos entre o padrão de entrada e o neurônio i (os pesos não estão mostrados na figura para não congestiona-la).

O neurônio vencedor quando o padrão $\boldsymbol{x}$ estiver sendo apresentado é aquele que tiver o maior valor de u_i. Vamos designa-lo por i^*. Segundo a regra do aprendizado competitivo, apenas o neurônio vencedor terá saída diferente de zero após a apresentação do padrão $\boldsymbol{x}$. A resposta da rede ao padrão $\boldsymbol{x}$ é,

$$y_i(\mathbf{x}) = \begin{cases} 1 \text{ para } i = i^* \\ 0 \text{ para } i \neq i^* \end{cases}.$$

O único neurônio ativo da rede em resposta ao padrão ***x*** (o neurônio vencedor) representa o padrão. É como se os *M* neurônios da rede competissem entre si para determinar quem vai ficar mais ativo em resposta ao padrão de entrada e apenas o vencedor ficasse ativo, deixando todos os perdedores inativos. Por causa disso, este tipo de rede é também chamada de rede do tipo "o vencedor fica com tudo".

Uma rede do tipo o vencedor fica com tudo implementa um mapa entre um espaço *N*-dimensional contínuo de vetores ***x*** e um espaço discreto de *M* neurônios. Note que podemos ter mais de um vetor ***x*** sendo representado pelo mesmo neurônio vencedor. Neste caso, este neurônio é o representante do grupo de padrões ***x*** que o fazem ser vencedor. Desta forma, este tipo de rede neural implementa um mecanismo de agrupamento (ou *clustering*) de padrões: padrões com características similares são representados pelo mesmo vetor da rede.

Após a determinação do neurônio vencedor em resposta a um padrão de entrada ***x***, ocorre a alteração nos pesos da rede. Pela regra do aprendizado competitivo, apenas os pesos do neurônio vencedor são modificados. Chamando o vetor de pesos do neurônio vencedor de $\boldsymbol{w}_{i^*}$, a regra do aprendizado competitivo implica em,

$$\mathbf{w}_{i^*}(n+1) = \mathbf{w}_{i^*}(n) + \eta\left(\mathbf{x}(n) - \mathbf{w}_{i^*}(n)\right)$$

e

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) \text{ para } i \neq i^*,$$

onde η é a constante de taxa de aprendizagem (um valor entre 0 e 1) que controla a rapidez com que as mudanças nos pesos são feitas.

Pare entender melhor a regra do aprendizado competitivo, vamos analisa-la vetorialmente. O neurônio vencedor é aquele que tiver o maior nível de ativação u_i. Como o nível de ativação de um neurônio i é o produto interno entre o vetor de entrada $\boldsymbol{x}$ e o vetor de pesos do neurônio i,

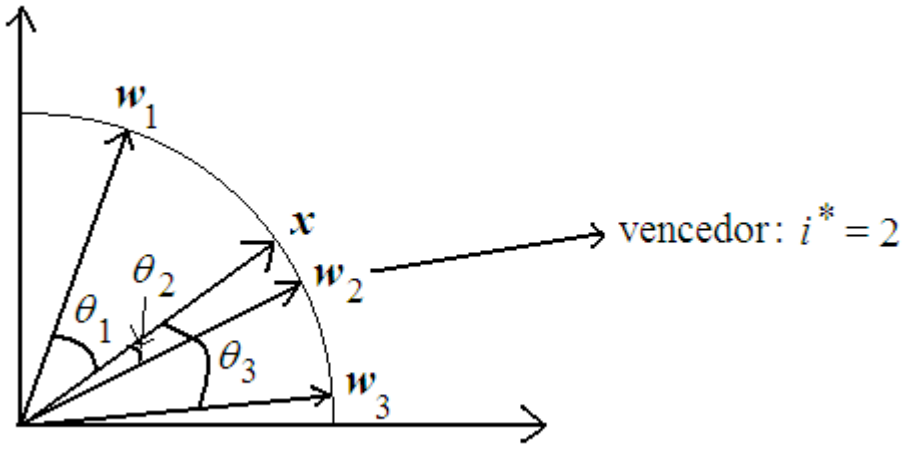
$$u_i = \mathbf{w}_i . \mathbf{x},$$

o critério para a escolha do neurônio vencedor é o de similaridade entre $\boldsymbol{x}$ e $\boldsymbol{w}_i$ tendo por medida de similaridade o produto interno. Sabemos que este produto pode ser escrito como

$$u_i = |\mathbf{w}_i||\mathbf{x}| \cos\theta,$$

onde θ é o ângulo entre os vetores $\boldsymbol{x}$ e $\boldsymbol{w}_i$.

Se os vetores de entrada e de pesos estiverem normalizados, um valor grande de u_i indica que o vetor de entrada $\boldsymbol{x}$ está próximo do vetor de pesos $\boldsymbol{w}_i$, ou seja, que $\boldsymbol{x}$ está nas vizinhanças de $\boldsymbol{w}_i$. Já um valor pequeno de u_i indica que o vetor de entrada $\boldsymbol{x}$ é quase perpendicular ao vetor de pesos $\boldsymbol{w}_i$.



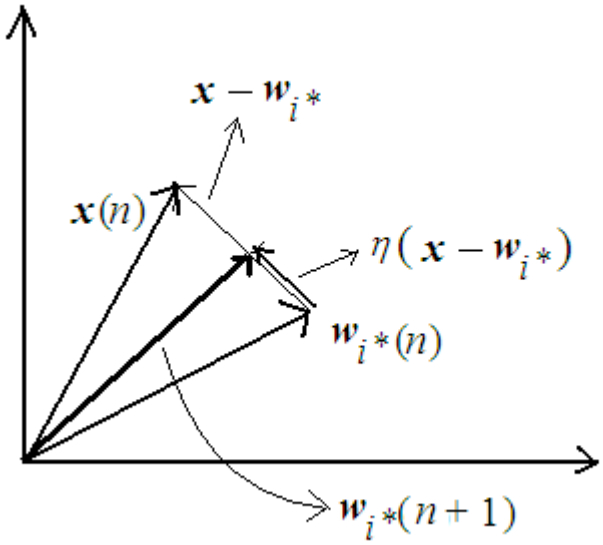
A figura acima ilustra o processo de escolha do neurônio vencedor em um caso em que os padrões de entrada são bi-dimensionais, com uma rede neural com três neurônios e os vetores de peso e de entrada normalizados. O neurônio vencedor i^* é o que tiver o valor máximo de $\boldsymbol{w}_i.\boldsymbol{x}$.

A normalização dos padrões de entrada provoca alterações neles, o que nem sempre é desejável. Para evitar isto, costuma-se usar como critério de definição do neurônio vencedor o cálculo da distância euclideana entre o vetor **x** e o vetor de pesos $\boldsymbol{w}_i$,

$$\|\mathbf{x} - \mathbf{w}_i\| = \sqrt{\sum_{j=1}^{N} \left(x_j - w_{ij}\right)^2} \ ,$$

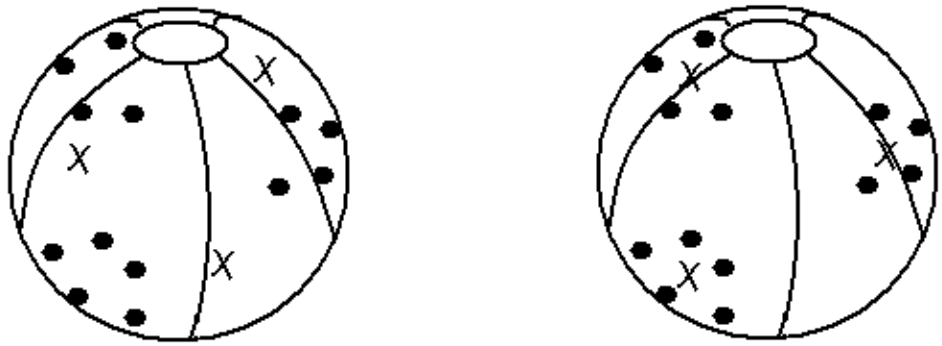
pois pode-se mostrar que maximizar o produto interno $\boldsymbol{w}_i.\boldsymbol{x}$ é matematicamente equivalente a minimizar a distância euclideana entre $\boldsymbol{x}$ e $\boldsymbol{w}_i$. Portanto, o neurônio vencedor i^* é aquele cujo vetor de pesos tiver a menor distância euclideana com o padrão de entrada. Este critério de escolha do neurônio vencedor foi utilizado pela primeira vez por Stephen Grossberg em 1969.

Uma vez determinado o neurônio vencedor, o seu vetor de pesos é modificado pela regra do aprendizado competitivo, $\Delta\boldsymbol{w}_{i*} = \eta(\boldsymbol{x} - \boldsymbol{w}_{i*})$. Ela nos diz que o vetor de pesos do neurônio vencedor deve ser modificado por um fator η na direção de $\boldsymbol{x} - \boldsymbol{w}_{i*}$. Geometricamente, em duas dimensões,



Esta regra de mudança de pesos faz com que o vetor de pesos do neurônio vencedor, que já era o mais próximo do padrão de entrada, seja arrastado na direção do padrão de entrada, ficando ainda mais próximo dele.

A partir de uma população de padrões de entrada retira-se, a cada passo, um dos padrões e aplica-o à entrada da rede. O neurônio vencedor é determinado e o seu peso alterado conforme a regra do aprendizado competitivo. Repetindo-se este procedimento várias vezes, os pesos da rede acabam convergindo para uma situação de relativa estabilidade em que eles ficam nos centros de massa de agrupamentos de padrões de entrada. A figura abaixo ilustra isso para um caso com 13 padrões de entrada tri-dimensionais e uma rede neural com 3 neurônios, onde por simplicidade os padrões de entrada e os vetores de peso têm todos módulo unitário, ou seja, podem ser representados por pontos na superfície de uma esfera.

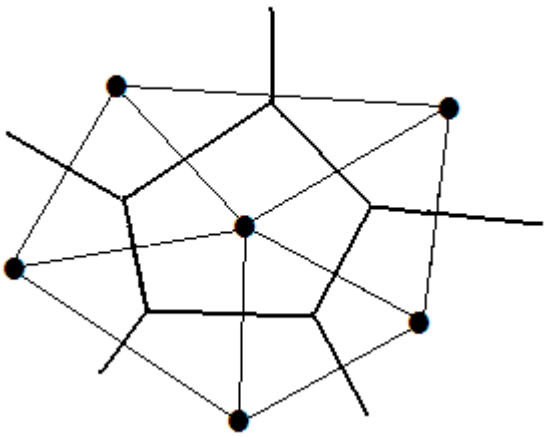


Início Fim

A convergência dos vetores de peso é controlada pelo valor de η. Se η for muito grande (próximo de 1) as mudanças nos pesos serão grandes e a convergência será rápida, mas os vetores de peso podem não se estabilizar. Se η for muito pequeno, a convergência pode ser muito vagarosa. Uma técnica muito adotada é fazer η variar com o passo de iteração t, começando com um valor próximo de 1 e diminuindo à medida que o aprendizado progride.

A regra de aprendizado competitivo faz com que a rede neural represente grupos de padrões que estejam próximos entre si por um vetor protótipo (por exemplo, os vetores indicados por cruzes na figura acima). O espaço de vetores de entrada fica dividido em regiões, ou vizinhanças, tais que cada uma delas está associada a um vetor protótipo que é o vetor de pesos de um dos neurônios da rede.

Os vetores protótipos são representantes de todos os possíveis vetores que estiverem na sua vizinhança. Se ligarmos os vetores protótipos por linhas e as cortamos perpendicularmente nos seus pontos médios, as linhas de corte vão se unir e formar uma divisão do espaço que é chamada de *mosaico de Voronoi*. O contínuo de pontos que pertence ao interior de uma das regiões do mosaico é representado pelo vetor protótipo correspondente (veja a figura abaixo).



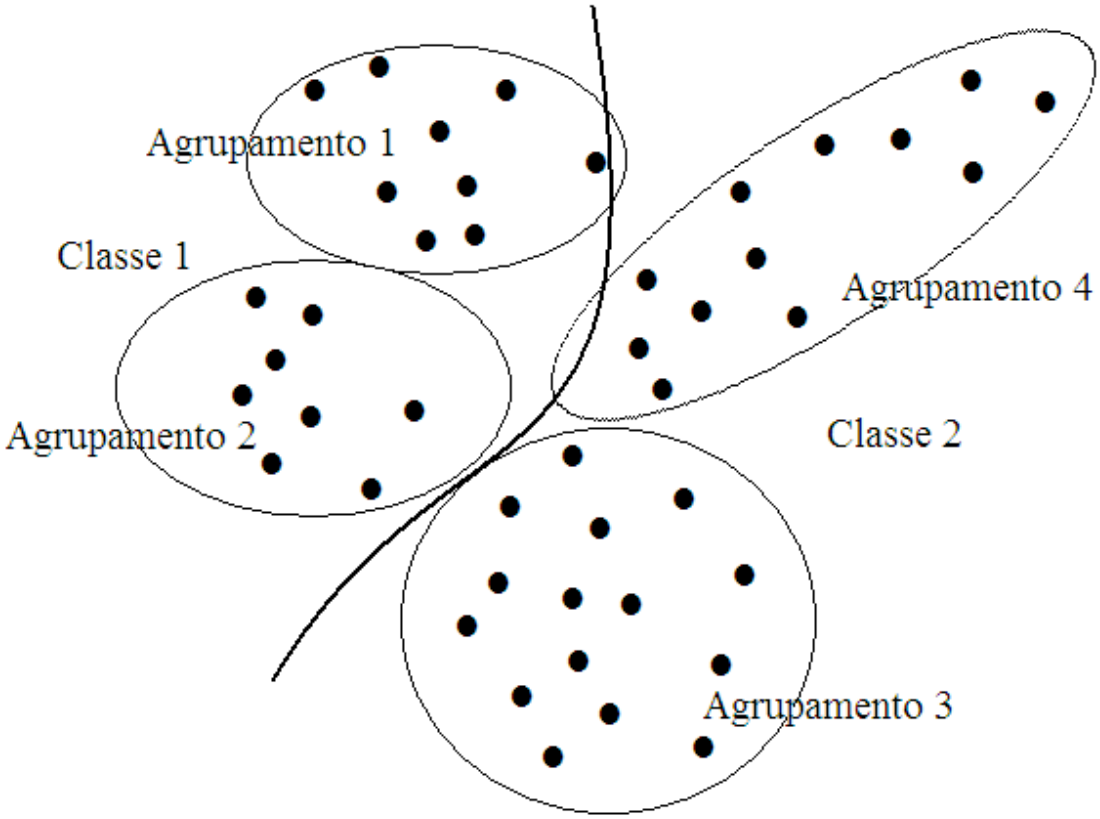
Uma das aplicações mais importantes do aprendizado competitivo está na quantização de vetores. Por exemplo, em telecomunicações a redução na dimensionalidade dos dados enviados é importante pela redução de custos que ela provoca. Uma das maneiras mais usadas de se reduzir a dimensão dos dados é por quantização de vetores. Nesta técnica, os vetores (dados) a serem transmitidos por uma linha de comunicação são agrupados em regiões como a do desenho acima e cada vetor protótipo recebe um número. Um livro de código é então criado de maneira que dado um número o vetor protótipo a ele associado pode ser obtido. Então, quando se quer transmitir um dado vetor o que se faz é enviar, ao invés dele, apenas o número do grupo ao qual ele pertence. O recebedor da mensagem tem o livro de código e consegue, a partir do número, construir o vetor protótipo que representa o vetor original que se queria transmitir. Se a quantização de vetores for feita com uma granularidade bem fina, de maneira que a distância entre um vetor de um dado agrupamento e o seu protótipo seja pequena e, portanto, o erro ao se usar o protótipo para representar o vetor também seja pequeno, a fidelidade deste método pode ser bastante boa.

O aprendizado competitivo é uma técnica de agrupamento de dados (*clustering*). É importante fazer uma distinção entre agrupamento e classificação de dados.

Um agrupamento é um processo pelo qual amostras de dados de uma população são colocadas juntas em um grupo de acordo com algum critério de vizinhança espacial (no R^N).

Uma classificação envolve a atribuição de rótulos às amostras de uma população, de acordo com algum critério externo (em geral, baseado nas convenções e preconceitos humanos).

Um agrupamento é um método não-supervisionado de se colocar padrões dentro de um mesmo grupo, enquanto que uma classificação é um método supervisionado. A figura abaixo (adaptada do livro de Principe, J.C., Euliano, N.R., Curt Lefebvre, W. *Neural and Adaptive Systems: fundamentals through simulations*. John Wiley & Sons, New York, NY, 2000.) ilustra a diferença entre agrupamento e classificação.



Na figura do exemplo existem duas classes e quatro agrupamentos. Como o agrupamento é determinado por um critério baseado apenas na proximidade entre os dados, pode haver diferentes maneiras de agrupá-los. Por exemplo, os dados dos agrupamentos 1 e 2 podem ser agrupados em um único grupo, ou os do agrupamento 4 podem ser divididos em dois, etc.

Note que se os dados dos agrupamentos 1 e 2 forem colocados dentro de um mesmo agrupamento, este coincidirá com a classe 1. Então, às vezes, uma dada maneira de se agrupar um conjunto de dados, baseada num critério natural de vizinhança entre eles, acaba coincidindo com um critério decidido antecipadamente para classificá-los.

Um estudo interessante que deve ser feito sempre que se usa um método de agrupamento (não supervisionado) é verificar se ele fornece uma divisão dos dados em grupos que tenham alguma similaridade com as classes em que nós, humanos, os classificamos. Em princípio, pode não haver qualquer relação entre os agrupamentos e as classes, e então o agrupamento pode estar revelando associações entre os dados completamente novas para nós (que podem ou não ser úteis), mas se for possível interpretar os agrupamentos em termos das classes usadas por nós, então o método de agrupamento pode já estar fazendo a tarefa de classificação para nós.

De um ponto de vista teórico, o agrupamento é uma forma de estimação de densidade de pontos não-paramétrica. Na ausência de um critério externo de classificação de dados, que nos diga em que categorias eles devem ser arranjados, o melhor que podemos fazer para dividi-los em grupos é usar a informação sobre a distribuição dos dados no espaço de entrada ($^{\circ N}$) para separá-los em conjuntos de elementos vizinhos neste espaço.

A idéia básica das técnicas de agrupamento é buscar regiões de alta densidade de pontos no espaço de entrada, os agrupamentos (*clusters*) de dados, e escolher os seus centros para representar os pontos das regiões. Portanto, o centro de cada uma dessas regiões é o protótipo que representa todos os pontos da região. No caso da rede neural treinada por aprendizado competitivo, os protótipos são os vetores de pesos dos M neurônios da rede.

Uma técnica não baseada em redes neurais de se fazer agrupamento de dados é a que usa o chamado algoritmo de agrupamento por K-médias (*K-means clustering*). A idéia deste algoritmo é encontrar a melhor divisão de P dados em K grupos C_i, $i = 1, ..., K$, de maneira que a distância total entre os dados de um grupo e o seu respectivo centro, somada por todos os grupos, seja minimizada.

Mais concretamente, suponha que temos P pontos N-dimensionais, $\boldsymbol{x}_i$, $i = 1,..., P$, e queremos encontrar um conjunto de K vetores protótipos $\boldsymbol{\mu}_j$, $j = 1,..., K$. O algoritmo deve separar os pontos $\{\boldsymbol{x}_i\}$ em K subconjuntos disjuntos S_j, cada um contendo n_j pontos, de maneira a minimizar a função que dá a soma total das distâncias entre os dados dos j grupos e os seus respectivos centros,

$$J = \sum_{j=1}^{K} \sum_{i \in S_j} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 ,$$

onde o centro $\boldsymbol{\mu}_j$ é a média dos n_j pontos no subconjunto S_j,

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i \in S_j} \mathbf{x}_i \, .$$

Note que este método é parecido com o de regressão linear, só que lá o que se minimiza é a soma dos quadrados das distâncias entre os pontos e a reta de regressão, e aqui o que se minimiza é a soma dos quadrados das distâncias entre os pontos de um grupo e o centro do grupo, tomado como a média dos pontos do grupo.

O algoritmo de agrupamento por *K*-médias começa atribuindo aleatoriamente os *P* pontos a *K* grupos e calculando as médias dos vetores de cada grupo. Depois, cada ponto é deslocado para o grupo correspondente ao vetor médio do qual ele está mais próximo. Com este novo rearranjo dos pontos em *K* grupos, novos vetores médios são calculados. O processo de re-alocação de pontos a novos grupos cujos vetores médios são os mais próximos deles continua até que se chegue a uma situação em que todos os pontos já estejam nos grupos dos seus vetores médios mais próximos. Pode-se mostrar matematicamente que durante esse processo iterativo o valor de *J* não aumenta; ele, ou permanece constante, ou diminui até atingir um mínimo.

A função *J* pode ser vista como uma função custo e o seu mínimo pode ser obtido usando-se o método do gradiente descendente com as componentes dos vetores médios dos *K* grupos sendo as variáveis em relação às quais as derivadas são calculadas. Se para calcular o gradiente de *J* em relação às componentes dos vetores médios μ_{jl} ($j = 1,...,K$; $l = 1,...,N$) usarmos uma aproximação parecida com a adotada por Widrow e Hoff no cálculo da regra LMS da Adaline, obtemos uma regra *on line* para a atualização dos vetores médios dos *K* grupos. Esta regra é a seguinte: Inicialmente, escolhe-se aleatoriamente *K* vetores $\boldsymbol{\mu}_j$ para ser os vetores centrais de cada grupo. Em seguida, os padrões $\boldsymbol{x}_i$ vão sendo apresentados, um por vez, e o vetor central mais próximo do padrão sendo apresentado é modificado pela regra,

$$\Delta\boldsymbol{\mu}_j(t) = \eta\left(\mathbf{x}(t) - \boldsymbol{\mu}_j(t)\right).$$

Note que esta é exatamente a regra de atualização do aprendizado competitivo se considerarmos que cada vetor central de um agrupamento é o vetor de pesos de um nos neurônios da rede.

Portanto, a aplicação do aprendizado competitivo a uma rede neural com *K* neurônios implementa uma versão *on line* (estocástica) do método de agrupamento por *K*-médias.

Alguns problemas apresentados pelo aprendizado competitivo do tipo "o vencedor fica com tudo" são:

1. Precisa-se de uma unidade de saída para cada agrupamento. Se a rede neural tem M neurônios ela pode agrupar os P dados em, no máximo, M classes distintas. Compare com o caso em que as M unidades podem ficar ativas simultaneamente e são binárias (0 ou 1): neste caso a rede pode representar 2^M classes diferentes;
2. Falta de robustez: se a unidade responsável por determinado agrupamento falhar, todos os padrões do agrupamento ficam sem representação;
3. Não se pode representar conhecimento hierárquico, com categorias dentro de categorias;
4. Unidades que tenham o seu vetor de pesos $\boldsymbol{w}(0)$ longe de qualquer um dos padrões $\boldsymbol{x}$ podem nunca vencer e, portanto, nunca terem seu vetor de pesos modificado. Elas ficam inativas por todo o aprendizado e, por isso, são chamadas de "unidades mortas". No entanto, é conveniente deixar algumas dessas unidades na rede para a eventualidade de aparecer um padrão novo que esteja próximo dos seus vetores de peso e longe dos demais (neste caso, será necessário um novo treinamento da rede).

Há maneiras de se evitar o aparecimento de unidades mortas em uma rede com aprendizado competitivo:

- Pode-se inicializar os pesos dos neurônios da rede com amostras dos próprios padrões de entrada;
- Pode-se atualizar os pesos dos neurônios perdedores também, mas com coeficientes η menores ("aprendizado vazado");
- Pode-se usar um termo de consciência, implementado como um termo de viés, proposto inicialmente por Grossberg em 1976 (veja a seguir):

  A idéia do termo de consciência é que cada unidade da rede deve ser penalizada se começar a ganhar demais (a sua "consciência" a faria se sentir

culpada de ganhar tanto, de maneira que ela própria dificultaria as suas vitórias para dar mais chances às demais unidades).

Para implementar o termo de consciência, cada unidade *i* deve ter uma espécie de contador do número de vezes que ela ganha,

$$c_i(t+1) = c_i(t) + \beta\left[r_i(t) - c_i(t)\right],$$

onde $c_i(0) = 0$, $r_i(t)$ é o resultado da competição no passo t (1 se a unidade i ganha, 0 se não) e β é uma constante positiva pequena (por exemplo, 0,0001). Com o uso desta fórmula, cada vez que a unidade i ganha c_i aumenta e cada vez que ela perde c_i diminui.

O termo de consciência do neurônio i é um viés b_i que é subtraído da distância entre o seu vetor de pesos $\boldsymbol{w}_i$ e o padrão atual $\boldsymbol{x}$, de maneira que a similaridade entre $\boldsymbol{w}_i$ e $\boldsymbol{x}$ passa a ser calculada como,

$$D(\mathbf{w}_i, \mathbf{x}) = \left\|\mathbf{x} - \mathbf{w}_i\right\| - b_i = \sqrt{\sum_{j=1}^{N}\left(x_j - w_{ij}\right)^2} - b_i,$$

e b_i é atualizado a cada passo de tempo pela fórmula,

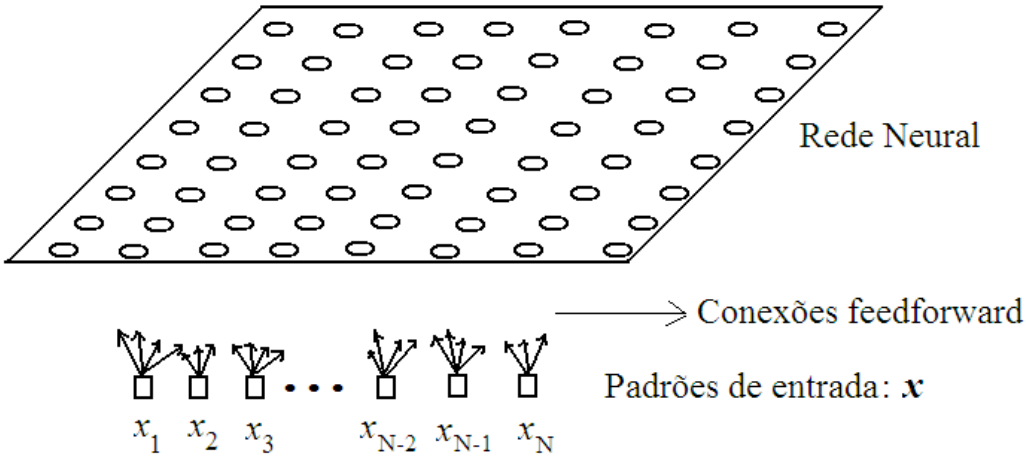
$$b_i(t) = \gamma\left(1/M - c_i(t)\right),$$

onde γ é uma constante positiva (por exemplo, 10).

Desta forma, se uma unidade ganha muito o seu c_i cresce fazendo o seu b_i tornar-se negativo, diminuindo as chances de que ela ganhe de novo. Por outro lado, uma unidade que ganha pouco tem c_i pequeno e b_i positivo, o que aumenta as suas chances de ganhar em um novo passo.

O algoritmo SOM de Kohonen esta baseado no aprendizado competitivo visto acima. Uma rede SOM é do tipo da mostrada abaixo. A camada de entrada recebe padrões que são vetores *N*-dimensionais vindos de alguma população de

de padrões e a rede SOM é uma camada bi-dimensional (como no caso do desenho) ou uni-dimensional.



Quando um padrão $\boldsymbol{x}$ é apresentado na entrada da rede, a unidade vencedora, i^*, é aquela cuja distância euclideana entre o seu vetor de pesos $\boldsymbol{w}_{i^*}$ e o padrão $\boldsymbol{x}$ for a menor de todas (como no caso do aprendizado competitivo),

$$i^*(\mathbf{x}) = \min_j \left\| \mathbf{x} - \mathbf{w}_j \right\|.$$

A diferença é que agora não é só o neurônio vencedor que tem o seu vetor de pesos atualizado, mas todos os neurônios vizinhos do neurônio vencedor. A regra de mudança de pesos do algoritmo SOM é,

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Lambda_{i,i^*}(t)\eta(t)\left[\mathbf{x}(t) - \mathbf{w}_i(t)\right],$$

onde $\Lambda_{i,i^*}(t)$ é a chamada função de vizinhança centrada no neurônio vencedor i^* e $\eta(t)$ é a taxa de aprendizagem. Em geral, tanto $\Lambda_{i,i^*}(t)$ como $\eta(t)$ variam com o passo de aprendizagem t.

O efeito da introdução da função de vizinhança $\Lambda_{i,i^*}(t)$ é fazer com que não apenas o vetor de pesos do neurônio vencedor seja modificado na direção do padrão atual, mas que os vetores de pesos de todos os neurônios vizinhos ao neurônio vencedor também sejam arrastados na direção do padrão atual, porém

por um fator menor vai diminuindo à medida que o neurônio correspondente vai ficando mais distante do vencedor. É como se todos os vetores de pesos dos neurônios da rede neural estivessem unidos por elásticos sendo que os elásticos mais fortes uniriam os vetores de pesos dos neurônios que fossem primeiros vizinhos, elásticos um pouco mais fracos uniriam os vetores de pesos dos neurônios que fossem segundos vizinhos, elásticos mais fracos uniriam os vetores de pesos dos neurônios que fossem terceiros vizinhos etc. Quando o vetor de pesos de um neurônio vencedor em um dado passo for modificado, por estar preso aos outros por elásticos, ele irá arrastar consigo os demais vetores de peso, e mais fortemente aqueles dos neurônios mais próximos.

Em geral, usa-se uma função gaussiana para implementar a função de vizinhança,

$$\Lambda_{i,i^*}(t) = \exp\left(-\frac{d_{i,i^*}^2}{2\sigma^2(t)} \right),$$

onde d_{i,i^*} é a distância entre um neurônio i e o neurônio vencedor i^*. Se a rede neural for uni-dimensional, essa distância é simplesmente o módulo da diferença entre os índices de i e i^*, d_{i,i^*} = |índice_de_i – índice_de_i^*|; se a rede for bi-dimensional essa distância é dada pela distância euclideana entre os seus vetores de posição,

$$d_{i,i^*}^2 = \left\| \mathbf{r}_i - \mathbf{r}_{i^*} \right\|^2,$$

onde $\boldsymbol{r}_i$ é o vetor posição do neurônio i e $\boldsymbol{r}_{i^*}$ é o vetor posição do neurônio vencedor, os dois sendo medidos nos espaço discreto definido pelos nodos da rede neural.

O desvio padrão da função de vizinhança, $\sigma_{i,i^*}(t)$, diminui com o número de passos t. Uma maneira comum de implementar essa diminuição é por um decaimento exponencial,

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1} \right),$$

onde τ_1 é uma constante temporal (determinada empiricamente).

Em geral, também faz-se a taxa de aprendizagem $\eta(t)$ diminuir com o passo de iteração de uma maneira exponencial,

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_2}\right),$$

onde τ_2 é outra constante temporal (também determinada empiricamente).

A função de vizinhança dá ao algoritmo SOM uma característica especial. Ele faz com que a representação dos padrões do espaço de entrada pela rede neural em grupos preserve a topografia do espaço de entrada. Isto quer dizer que padrões vizinhos no espaço de entrada são representados por neurônios vizinhos na rede neural. Além disso, regiões do espaço de entrada cujos padrões $\boldsymbol{x}$ tenham maior probabilidade de ocorrer são representadas por um número maior de neurônios da rede neural, isto é, elas são representadas com uma resolução maior do que as regiões cujos padrões ocorrem menos freqüentemente.

Portanto, um mapa entre o espaço contínuo de entrada e o espaço discreto da rede neural implementado pelo algoritmo SOM tende a preservar tanto a métrica como a distribuição do espaço de entrada. É importante lembrar que isto ocorre a partir apenas da informação contida nos padrões de entrada; o mapa criado pelo SOM não é supervisionado.

Durante o treinamento segundo o algoritmo SOM ocorrem, em geral, duas fases distintas:

1. Fase de auto-organização ou de ordenamento: É nesta fase que o ordenamento topográfico dos vetores de pesos ocorre. Inicialmente, os vetores de pesos têm valores aleatórios e não possuem qualquer tipo de ordenamento. À medida que a rede vai sendo treinada, vetores de neurônios vizinhos entre si no espaço da rede neural começam a se aproximar uns dos outros, de maneira que

neurônios de uma mesma área da rede neural acabam representando padrões vindos de uma mesma região do espaço de entrada. Esta fase pode levar muitas iterações para terminar, em geral mais do que 1000. Algumas dicas úteis para que ela ocorra da melhor maneira possível, dadas por Haykin em seu livro, são as seguintes:

- A taxa de aprendizagem $\eta(t)$ deve começar com um valor próximo de 0,1 e ir diminuindo gradualmente, mas sem ficar abaixo de 0,01. Para um decaimento exponencial como o dado pela fórmula acima, a constante τ_2 deve ser da ordem de 1000.
- Inicialmente, a função de vizinhança $\Lambda_{i,i^*}(t)$ deve incluir praticamente todos os neurônios da rede (ou seja, todo mundo sendo vizinho do neurônio vencedor i^*), mas depois ela deve ir encolhendo com o passo de iteração até que, após mais ou menos 1000 passos, apenas os quatro primeiros vizinhos de i^* (no caso de uma rede bi-dimensional quadrada) façam parte da sua função de vizinhança (e mesmo assim, um pouco mais tarde, somente o próprio neurônio vencedor é que tenha o seu vetor de pesos modificado, como no aprendizado competitivo do tipo o vencedor fica com tudo). Uma possível maneira de implementar isso é fazendo σ_0 ser igual ao "raio" da rede neural e $\tau_1 = 1000/\log\sigma_0$.

2. Fase de Convergência: Nesta fase ocorre o refinamento do mapa, levando a uma representação mais acurada do espaço de entrada por parte da rede neural. Como regra geral, o número de iterações nesta fase deve ser de pelo menos 500 vezes o número de neurônios na rede neural. Logo, o número de iterações pode atingir dezenas de milhares de passos. Nesta fase, a taxa de aprendizagem η deve permanecer pequena, da ordem de 0,01 e a função de vizinhança de englobar apenas o próprio neurônio vencedor e, no máximo, os seus primeiros vizinhos.

Uma síntese do algoritmo SOM de Kohonen é a seguinte:

1. Inicialização: Escolha valores aleatórios para as componentes iniciais dos vetores de pesos $\boldsymbol{w}_i(0)$, $i = 1,..., M$. A única restrição é que os M vetores de pesos devem ser diferentes uns dos outros, mas é, em geral, conveniente que os seus módulos sejam pequenos;
2. Escolha do padrão de entrada: De acordo com alguma distribuição de probabilidades $p(\boldsymbol{x})$, escolha um padrão $\boldsymbol{x}$ da população para ser colocado na camada de entrada da rede;
3. Determinação do neurônio vencedor: Use o critério de similaridade baseado na distância euclideana entre o vetor de entrada e os vetores de peso para determinar o neurônio vencedor i^* para o passo atual,

$$i^*(\mathbf{x}) = \min_j \left\| \mathbf{x} - \mathbf{w}_j \right\|;$$

4. Atualização dos pesos: Modifique os vetores de pesos dos neurônios da rede de acordo com a regra,

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Lambda_{i,i^*}(t)\eta(t)\left[\mathbf{x}(t) - \mathbf{w}_i(t)\right];$$

5. Continuação: Volte para o passo 2 e continue até que não sejam observadas mudanças significativas no mapa formado.