Mapas auto-organizáveis de Kohonen- (SOM)

Deep Learning

Material Complementar

Tutorial básico sobre Self Organizing Maps: Self-Organizing Maps Tutorial

Um dos tutoriais mais acessados sobre mapas auto organizáveis, inclusive o autor desenvolver um software em Windows que você pode fazer alguns experimentos bem interessantes: Kohonen's Self Organizing Feature Maps

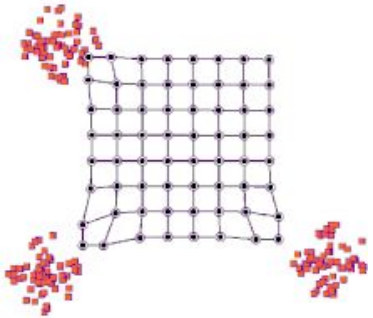
Caso você queira saber mais sobre os cálculos matemáticos e fórmulas detalhadas: The Self-Organizing Maps: Background, Theories, Extensions and Applications

Livro **Self-Organizing Maps** de **Teuvo Kohonen**: este é o livro definitivo caso você queira aprender sobre a teoria dos mapas auto organizáveis

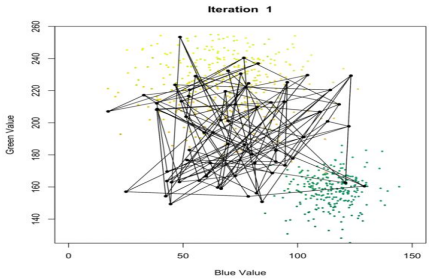
Arquiteturas de RNAs

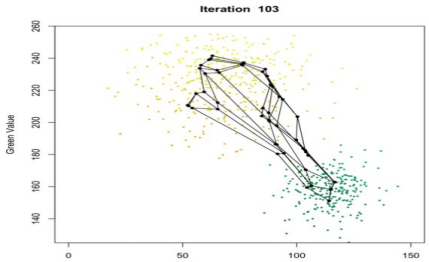
Aprendizagem supervisionada	Aprendizagem não supervisionada
Redes Neurais Artificiais classificação e regressão	Mapas auto organizáveis detecção de características e agrupamento
Redes Neurais Convolucionais visão computacional	Boltzmann machines sistemas de recomendação redução de dimensionalidade
Redes Neurais Recorrentes análise de séries temporais	Autoencoders redução de dimensionalidade
	Redes adversariais generativas geração de imagens

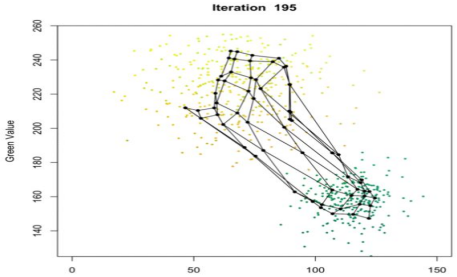
Contextualização



O algoritmo de Kohonen foi desenvolvido por Teuvo Kohonen em 1982, sendo considerado relativamente simples e com a capacidade de organizar dimensionalmente dados complexos em grupos (*clusters*), de acordo com suas relações. Este método solicita apenas os parâmetros de entrada, mostrando-se ideal para problemas onde os padrões são desconhecidos ou indeterminados.

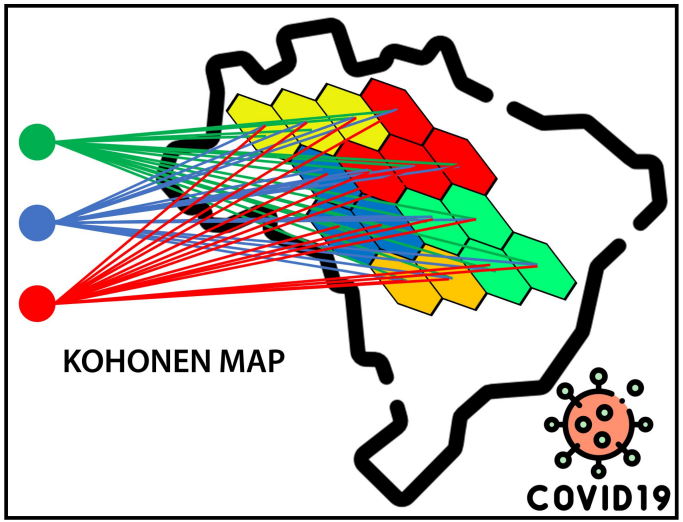






Mapas auto-organizáveis

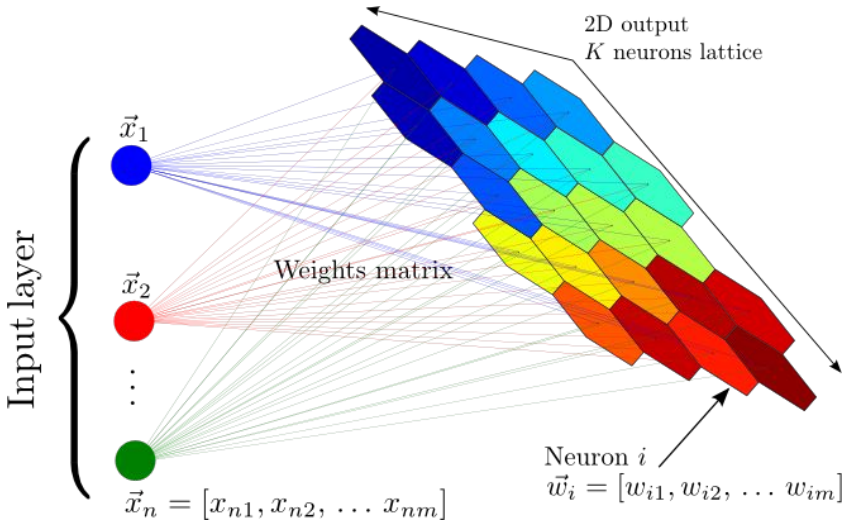
SOM é treinado por meio de aprendizagem não supervisionada, é um pouco diferente de outras redes neurais artificiais, O SOM não aprende por retropropagação com SGD, ele usa o aprendizado competitivo para ajustar os pesos dos neurônios. E usamos este tipo de redes neurais artificiais na redução de dimensão para reduzir nossos dados, criando uma representação espacialmente organizada, também nos ajuda a descobrir a correlação entre os dados.



Mapas auto-organizáveis

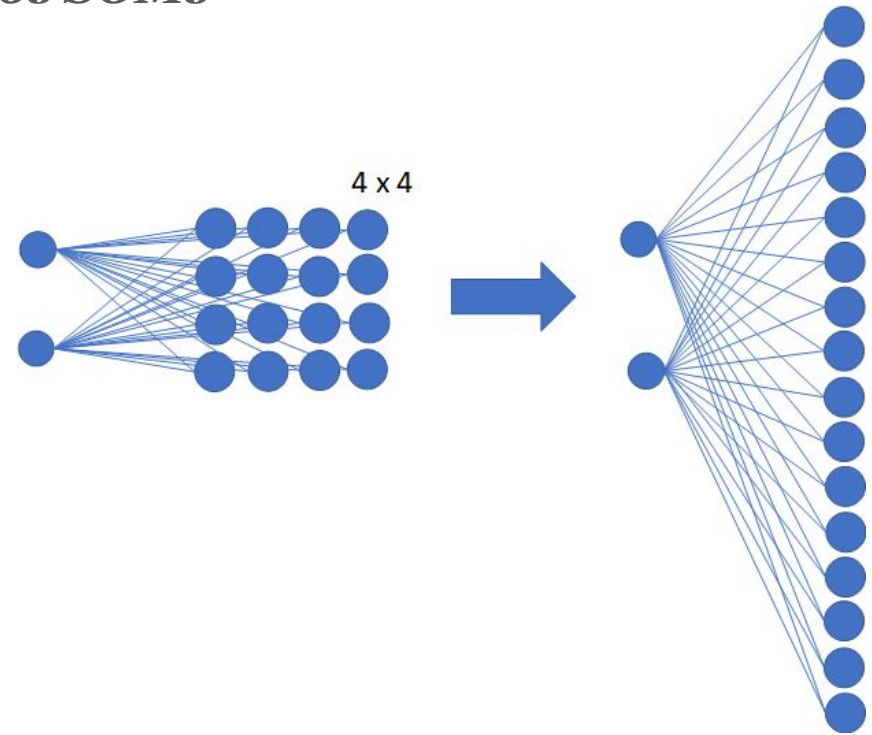
Arquitetura do SOM

Os SOMs têm duas camadas, a primeira é a camada de entrada e a segunda é a camada de saída ou o mapa de características. Ao contrário de outros tipos de RNA, SOM não tem função de ativação em neurônios, passamos pesos diretamente para a camada de saída sem fazer nada. Cada neurônio em um SOM é atribuído a um vetor de peso com a mesma dimensionalidade d do espaço de entrada.



Mapas auto-organizáveis

Arquiteturas dos SOMs



Mapas auto-organizáveis

Treinamento de mapas auto-organizáveis

Como mencionamos antes, o SOM não usa retropropagação com SGD para atualizar os pesos, este tipo de rede neural artificial não supervisionada usa aprendizado competitivo para atualizar seus pesos.

A aprendizagem competitiva é baseada em três processos:

- Competição
- Cooperação
- Adaptação

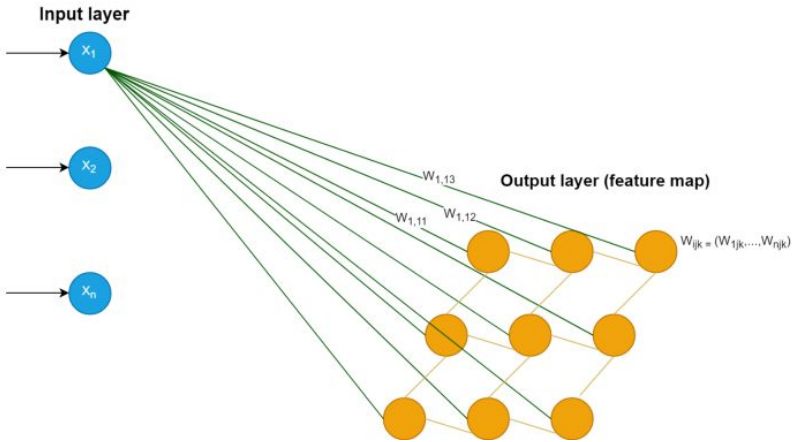
Mapas auto-organizáveis

Competição:

Como dissemos antes, a cada neurônio em um SOM é atribuído um vetor de peso com a mesma dimensionalidade do espaço de entrada.

No exemplo abaixo, em cada neurônio da camada de saída teremos um vetor com dimensão n.

Calculamos a distância entre cada neurônio (neurônio da camada de saída) e os dados de entrada, e o neurônio com a menor distância será o vencedor da competição.



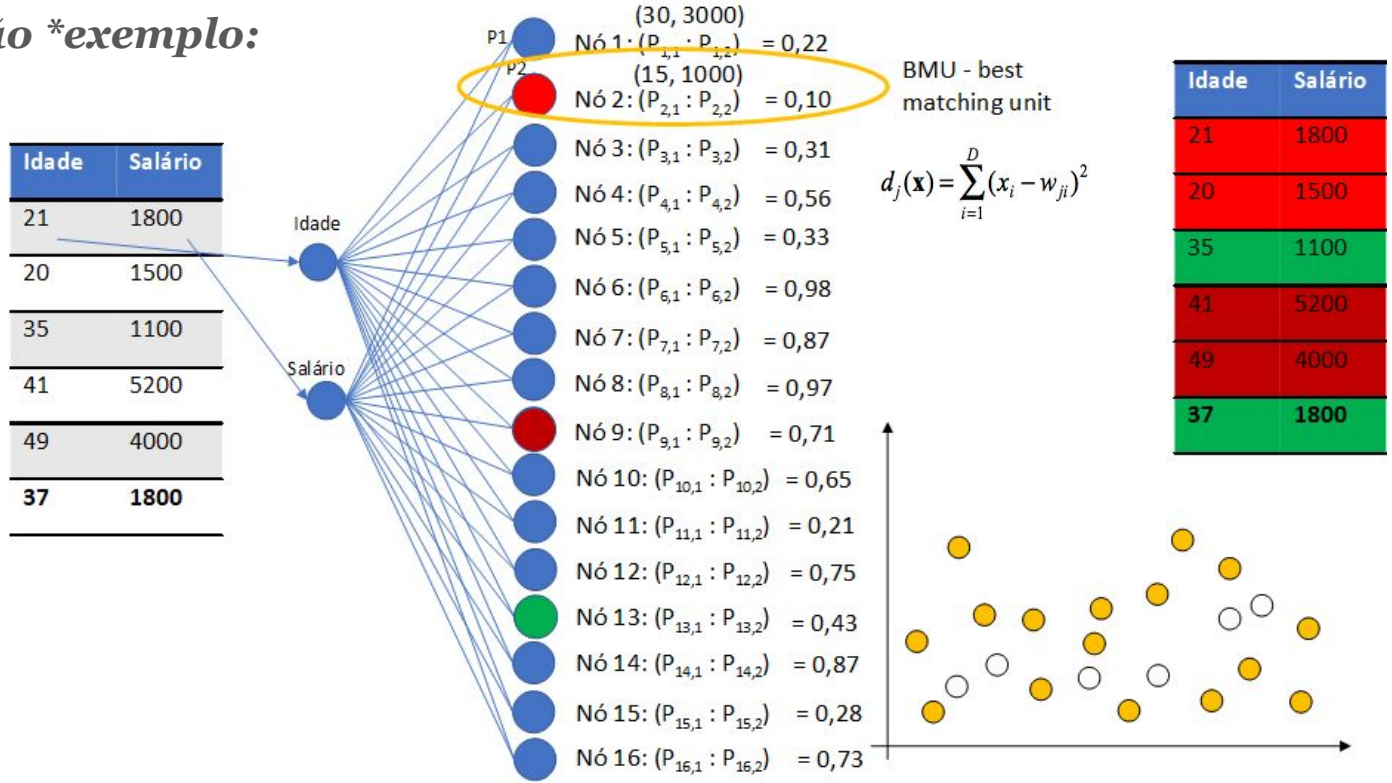
A métrica euclidiana ***dj*** é comumente usada para medir distâncias

$$d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

Mapas auto-organizáveis

Competição *exemplo:

Idade	Salário
21	1800
20	1500
35	1100
41	5200
49	4000
37	**1800**



Idade	Salário
21	1800
20	1500
35	1100
41	5200
49	4000
37	**1800**

Mapas auto-organizáveis

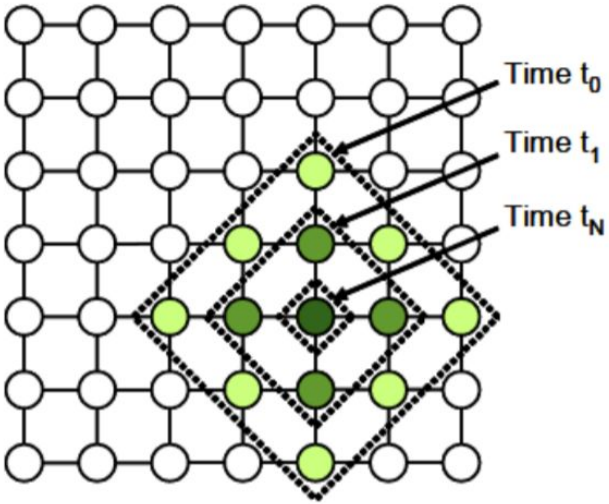
Cooperação:

Vamos atualizar o vetor do neurônio vencedor no processo final (adaptação), mas não é o único, também seu vizinho será atualizado.

Como escolhemos os vizinhos?

Para escolher vizinhos, usamos a função kernel de vizinhança, esta função depende de dois fatores: tempo (o tempo é incrementado a cada novo dado de entrada) e distância entre o neurônio vencedor e o outro neurônio (a que distância está o neurônio do neurônio vencedor).

A imagem ao lado mostra como os vizinhos do neurônio vencedor (o mais verde no centro) são escolhidos de acordo com os fatores de distância e tempo.



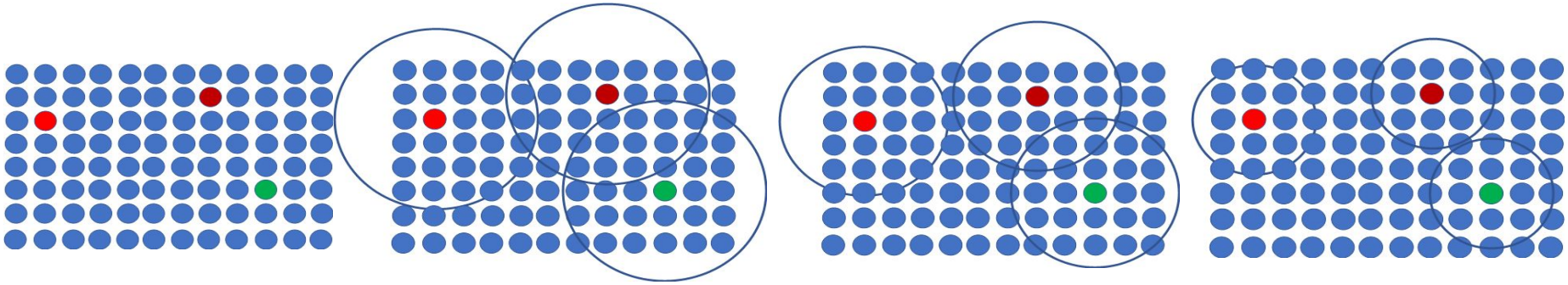
Mapas auto-organizáveis

Cooperação:

Quando um neurônio é ativado seus vizinhos ficarão mais excitados comparados com neurônios que estão mais distantes. Esse processo é chamado de vizinhança topológica e é calculada como mostrado abaixo:

$$T_{j,I(\mathbf{x})} = \exp(-S^2_{j,I(\mathbf{x})} / 2\sigma^2)$$

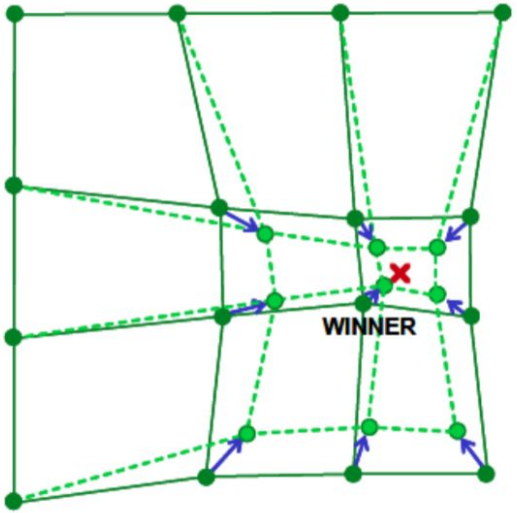
Onde S é a distância lateral entre os neurônios, I(x) é o índice do neurônio winner e o σ é o número de vizinhos que decresce com o tempo. A vizinhança topográfica irá diminuir, tendendo a zero quando um neurônio estiver muito distante do winner.



Mapas auto-organizáveis

Adaptação:

Depois de escolher o neurônio vencedor e seus vizinhos, calculamos a atualização dos neurônios. Os neurônios escolhidos serão atualizados, mas não a mesma atualização, mais a distância entre o neurônio e os dados de entrada aumenta, menos ajustamos como mostrado na imagem ao lado:



O neurônio vencedor e seus vizinhos serão atualizados usando esta fórmula:

$$\Delta w_{ji} = \eta(t) \,.\, T_{j,I(\mathbf{x})}(t) \,.\, (x_i - w_{ji})$$

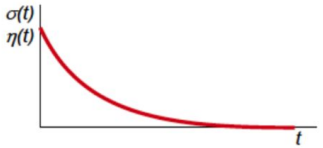
Mapas auto-organizáveis

Adaptação:

A taxa de aprendizagem indica o quanto queremos ajustar nossos pesos.

Após o tempo t (infinito positivo), essa taxa de aprendizado convergirá para zero, portanto, não teremos atualização nem mesmo para o neurônio vencedor.

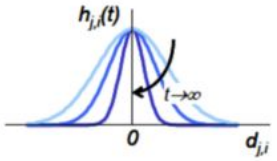
A learning rate decay rule $\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_1}\right)$



A neighborhood kernel function $h_{ik}(t) = \exp\left(-\frac{d_{ik}^2}{2\sigma^2(t)}\right)$

- where d_{ik} is the lattice distance between w_i and w_k

A neighborhood size decay rule $\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_2}\right)$



Mapas auto-organizáveis

Algoritmo SOM completo:

1. Initialize weights to some small, random values
2. Repeat until convergence
   - 2a. Select the next input pattern $x^{(n}$ from the database
     - 2a1. Find the unit w_i that best matches the input pattern $x^{(n}$

$$i(x^{(n)}) = \arg\min_j \left\| x^{(n} - w_j \right\|$$

     - 2a2. Update the weights of the winner w_i and all its neighbors w_k

$$w_k = w_k + \eta(t) \cdot h_{ik}(t) \cdot (x^{(n} - w_k)$$

   - 2b. Decrease the learning rate $\eta(t)$
   - 2c. Decrease neighborhood size $\sigma(t)$

Referências

Descobrindo SOM, uma Rede Neural com aprendizado não supervisionado

Mapas auto-organizáveis