Redes Auto-Organizáveis: Mapas de Kohonen

Gustavo Scaloni Vendramini
Guilherme José Henrique
Sean Carlisto de Alvarenga
Vinícius Fernandes de Jesus

11 de setembro de 2013

Resumo

Este artigo apresenta uma implementação de um modelo de rede neural auto-organizável conhecida como mapas auto-organizáveis de Kohonen ou redes SOM (do inglês *Self-Organizing Maps*).

1 Redes SOM

Mapas auto-organizáveis (Self-Organized Maps) ou Mapas de Kohonen são um tipo de rede neural. Foram desenvolvidos em 1982 por Tuevo Kohonen, professor emérito da Academia da Finlândia. "Auto-organização" é porque a supervisão não é necessária. SOMs aprendem por conta própria através da aprendizagem competitiva não supervisionada. "Mapas" significa que eles tentam mapear seus pesos para entrar em conformidade com os determinados dados de entrada. Os nós de uma rede SOM tentam tornar-se igual às entradas que lhes são apresentados. Neste sentido, é assim que eles aprendem.

1.1 Estrutura

A Figura 1 é uma rede SOM 4x4 (4 linhas por 4 colunas de neurônios). Cada mapa está ligado a cada nó de entrada. Para esta pequena rede 4x4, resulta em 4x4x3 = 48 ligações.

Nota-se que os nós da camada de saída do mapa não são ligados uns aos outros. Os nós organizados desta maneira, como uma grade de duas dimensões, faz com que seja fácil de visualizar os resultados.

Nesta configuração, cada nó tem um única coordenada (i, j), tornando mais fácil fazer referência a um nó na rede, e calcular as distâncias entre os nós. Devido as ligações serem apenas dos nós de saída para o nós de entrada, os nós da camada de saída são indiferentes quanto ao valores dos pesos de seus vizinhos. A atualização dos pesos será explicada em sequência 1.2.

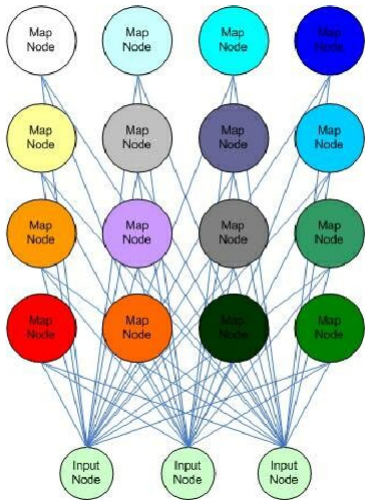

Figura 1: Nó do Mapa de Kohonen

1.2 Treinamento

A rede SOM utiliza um algoritmo de aprendizado competitivo e não-supervisionado. Um padrão de entrada é apresentado a rede, um neurônio vence e inicia a atualização de seus pesos e de seus vizinhos (até um raio de vizinhança). Isto repete para cada nova entrada e a taxa de aprendizado e o raio de vizinhança são decrementados durante o processo [1, 2]. A atualização dos pesos do neurônio vencedor e de seus vizinhos é dada como segue:

$$w_{ji}(t+1) = \begin{cases} w_{ji}(t) + \eta(t)(x_i - w_{ji}(t)) & \text{se j} \in \Lambda(\text{t}) \\ w_{ji}(t) & \text{caso contrário} \end{cases}$$

Onde w_{ji} é o peso entre os neurônios i e j, $\eta(t)$ é a taxa de aprendizado e Λ é a vizinhança.

```
1:  Inicializar pesos e parâmetros;
2:  repita
3:     para todo padrão de treinamento faça
4:        Definir neurônio vencedor;
5:        Atualizar os pesos deste neurônio e de seus vizinhos;
6:        se o número do ciclo for múltiplo de N então
7:           Então reduzir a taxa de aprendizado;
8:        fim-se
9:     fim-para
10: até que mapa de características não mudar
```

1.3 Classificação de Cor

Este é considerado o "Hello World" de SOMs. A maioria dos tutoriais publicados no SOMs usam a classificação de cor como seu principal exemplo. Isto é para uma boa razão. Ao passar por este exemplo, o conceito de SOMs pode ser solidamente apreendido. A razão pela qual a classificação de cor é bastante fácil de entender é por causa da quantidade relativamente pequena de dados utilizados, bem como o aspecto visual dos dados.

A classificação de cores do SOM usam apenas três pesos por nó de saída e nós de entrada. Estes pesos representam a tripla (r, g, b) para a cor. Por exemplo, as cores podem ser apresentado à rede, (1,0,0) para o vermelho, (0,1,0) para o verde, etc. A meta para a rede aqui é aprender a representar todas essas cores de entrada em sua grade de duas dimensões. Com isso em mente, se azul escuro e azul claro são apresentados ao SOM, eles devem acabar ao lado do outro na grade de rede.

Para ilustrar o processo passo a passo do algoritmo de classificação da cor, o primeiro passo consiste em inicializar a rede, como mostra a figura 2.

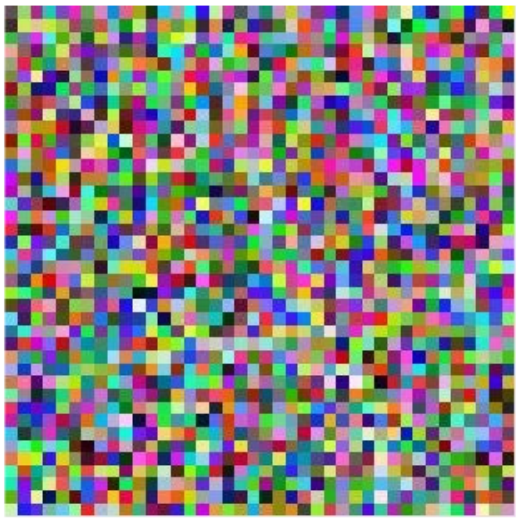

Figura 2: Disposição inicial dos neurônios

Em seguida, temos que escolher um vetor de forma aleatória a partir dos vetores de entrada. Oito vetores de entrada são utilizados neste exemplo, que vão do vermelho para o amarelo para verde escuro.

Posteriormente, o próximo passo é passar por cada nó e encontrar BMU (Best Matching Unit, utilizando a distância euclidiana, por exemplo). A Figura 3 mostra o BMU sendo selecionado na rede 4x4.

Com o BMU encontrada, o raio vizinhança deve ser calculado, como também mostrado na Figura 3. Todos os nós em vermelho estão dentro do raio.

Por último, aplica-se o aprendizado para todos esses nós. O aprendizado baseia-se na distância do neurônio em relação a entrada apresentada.

Em seguida, um novo vetor aleatório é escolhido e todo processo se repete.

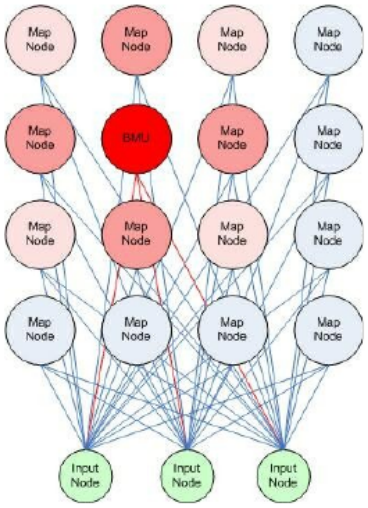

Figura 3: Nó vencedor

A Figura 4 mostra um SOM treinado, representando todas as cores das oito entradas de cores. Observe como a cor verde fica ao lado do verde escuro e a cor vermelha está ao lado da cor laranja.

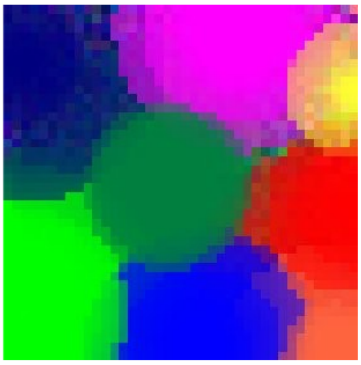

Figura 4: Estrutura do nó final

2 Resultados

A rede neural auto-organizável implementada nesse trabalho, utilizou o modelo Mapa de Kohonen, desenvolvido por Teuvo Kohonen em 1982 [3]. A implementação deste trabalho pode ser vista em `https://github.com/QSF/ia-kohonen`.

Para os resultados práticos, a rede foi treinada com um conjunto de 1728 exemplos de dados reais, fornecidos pelo site *UCI Machine Learning Repository*[1], variando valores da taxa de aprendizado, pesos iniciais, raio da vizinhança e quantidade de neurônios na rede. O conjunto de exemplos escolhido foi o *Car Evaluation*[2], onde dados os atributos preço de compra do veículo, custo da manutenção, quantidade de portas do veículo, número de passageiros, tamanho do bagageiro e segurança estimada do veículo, avalia se o veículo é aceitável, inaceitável, bom ou muito bom.

Uma vez que o algoritmo Mapa de Kohonen organiza ou agrupa dimensionalmente os dados, o conjunto de exemplos foi adaptado, apresentando somente três dimensões (preço de compra do veículo, custo da manutenção e segurança estimada do veículo). O primeiro e segundo atributo aceitam os valores {low, med, high e vhigh}, enquanto que o último aceita os valores {low, med, high}.

Com o conjunto de dados coletados, algumas entradas foram testadas para verificar a modificação final da rede. Esses testes serão exibidos a seguir, sendo dividido pela quantidade de neurônios na camada de saída (16, 20, 25 e 36 neurônios).

2.1 16 neurônios

Nos conjuntos de teste apresentados por essa seção, a rede possuia 16 neurônios na camada de saída, sendo que sua topologia é da forma de uma grade 4x4.

Para realizar os testes na rede mencionada, foi variado o valor da taxa de aprendizagem e dos pesos inicias, com raio fixo em 0.

Quatro testes foram executados, onde a configuração da rede para cada caso é ilustrada em 1.

Tabela 1: Configurações da Rede com 16 Neurônios na Saída

Teste	Taxa de Aprendizagem	Pesos Inicias
1	0.1	todos 1
2	0.1	Random
3	0.3	todos 1
4	0.3	Random

As figuras 5, 6, 7 e 8 representam a rede após a execução dos testes 1, 2, 3 e 4; respectivamente.

2.2 20 neurônios

Nesta seção, a rede possuia 20 neurônios na camada de saída, sendo uma grade 4x5.

[1] http://archive.ics.uci.edu/ml/

[2] http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

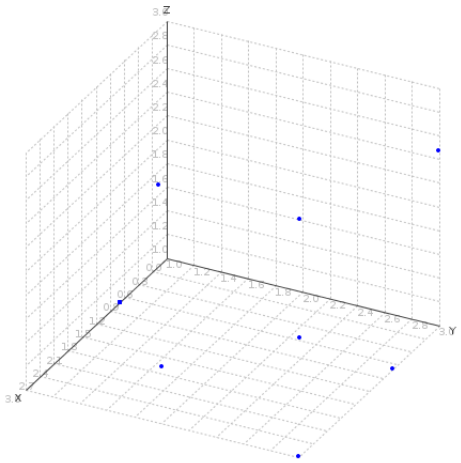

Figura 5: 16 Neurônios - Teste 1

Para realizar os testes na rede mencionada, novamente foi variado o valor da taxa de aprendizagem e dos pesos inicias, mas com raio fixo em 1.

Também foram executados quatro testes que tem sua configuração ilustrada na tabela 2.

Tabela 2: Configurações da Rede com 20 Neurônios na Saída

Teste	Taxa de Aprendizagem	Pesos Inicias
1	0.5	todos 1
2	0.5	Random
3	0.75	todos 1
4	0.75	Random

Os testes 1, 2, 3 e 4 são representados pelas imagens 9, 10, 11 e 12; respectivamente.

2.3 25 neurônios

Os gráficos a seguir mostram a plotação dos pontos após a execução do algoritmo para alguns valores de parâmetros e para uma rede com 25 neurônios. Apesar de não se poder afirmar se o algoritmo convergiu, vemos que aumentando o raio de vizinhança, o algoritmo consegue "especializar" melhor o conjunto. O efeito da taxa de aprendizado está sobre a velocidade na qual o algoritmo irá convergir. Ela geralmente é dada empiricamente, assim como os pesos iniciais, que segundo

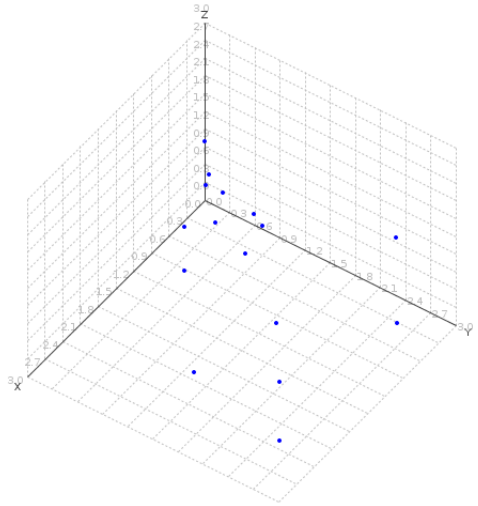

Figura 6: 16 Neurônios - Teste 2

[3], se forem inicializados com valores iguais, faz o algoritmo convergir mais rapidamente.

2.4 36 neurônios

Os gráficos a seguir mostram a plotação dos pontos após a execução do algoritmo para alguns valores de parâmetros e para uma rede com 36 neurônios.

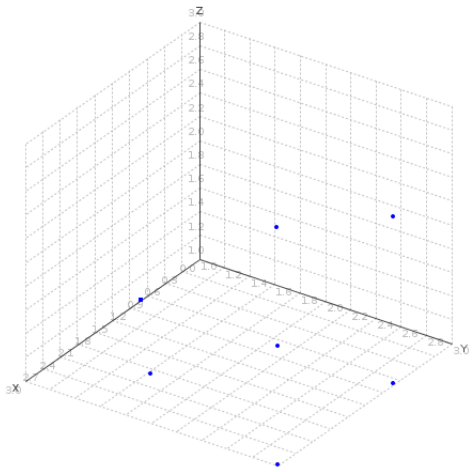

Figura 7: 16 Neurônios - Teste 3

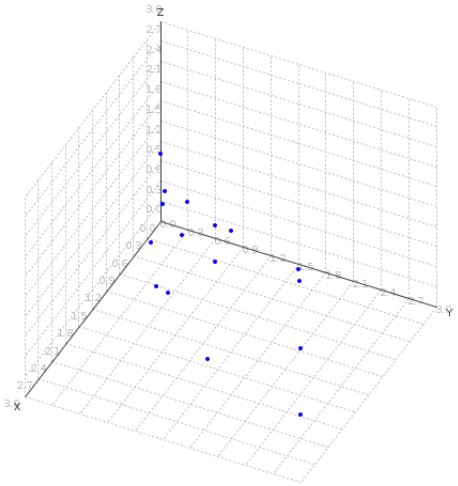

Figura 8: 16 Neurônios - Teste 4

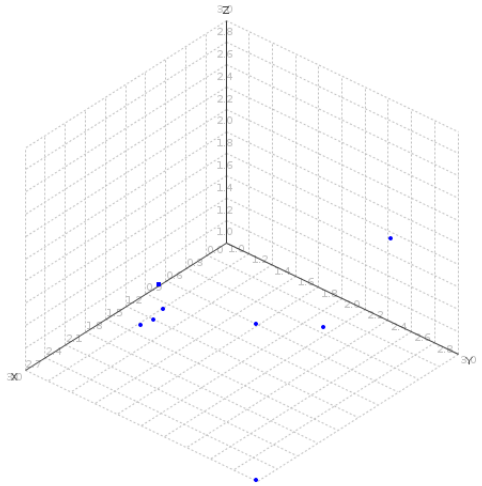

Figura 9: 20 Neurônios - Teste 1

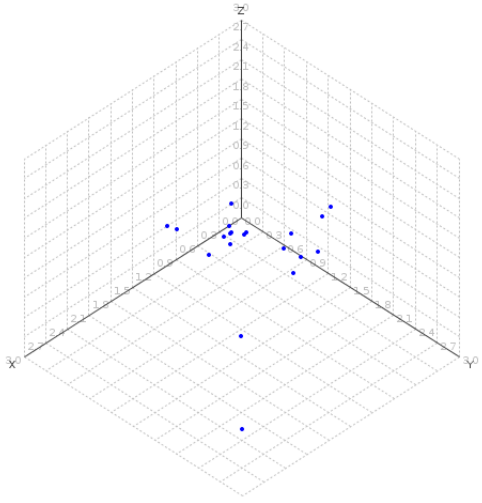

Figura 10: 20 Neurônios - Teste 2

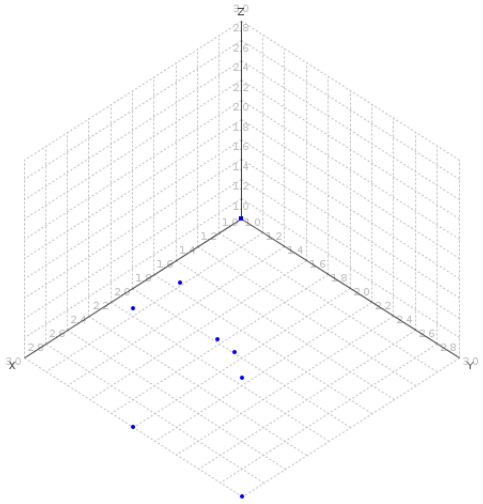

Figura 11: 20 Neurônios - Teste 3

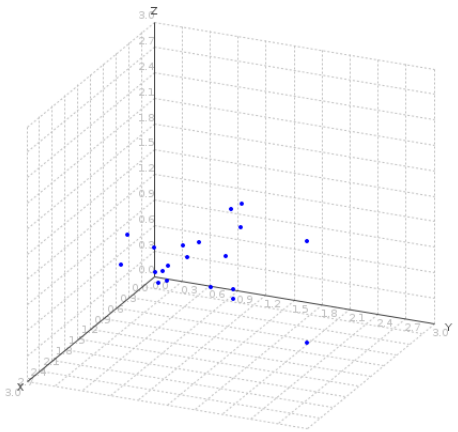

Figura 12: 20 Neurônios - Teste 4

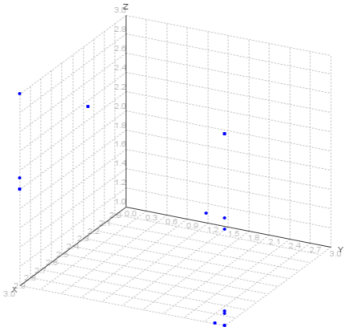

Figura 13: Taxa de aprendizado: 0.8662815917277779; Raio: 1; Pesos iniciais: A

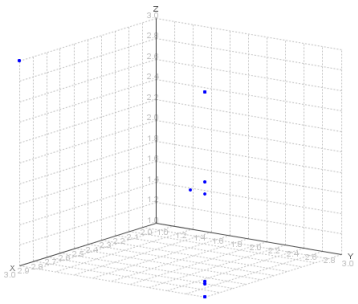

Figura 14: Taxa de aprendizado: 0.8662815917277779; Raio: 2; Pesos iniciais: A

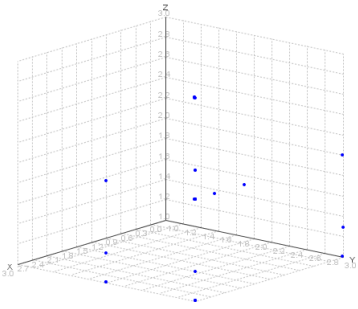

Figura 15: Taxa de aprendizado: 0.5980629675758204; Raio: 1; Pesos iniciais: B

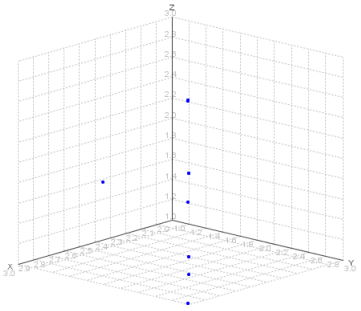

Figura 16: Taxa de aprendizado: 0.5980629675758204; Raio: 2; Pesos iniciais: B

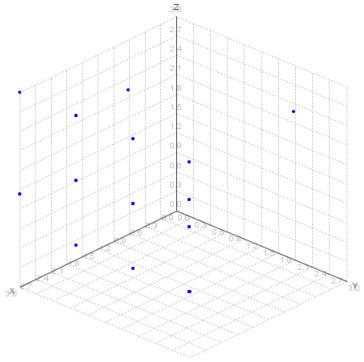

Figura 17: Taxa de aprendizado: 0.2625656059885696; Raio: 1; Pesos iniciais: D

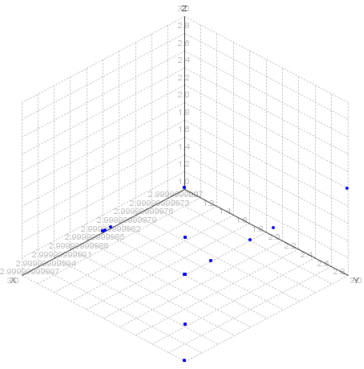

Figura 18: Taxa de aprendizado: 0.2625656059885696; Raio: 2; Pesos iniciais: D

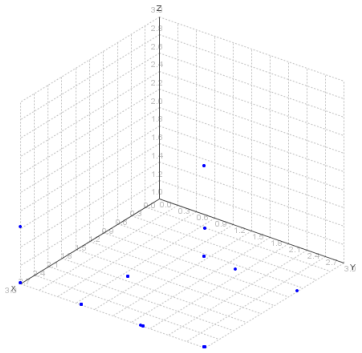

Figura 19: Taxa de aprendizado: 0.5517820314689292; Raio: 1; Pesos iniciais: E

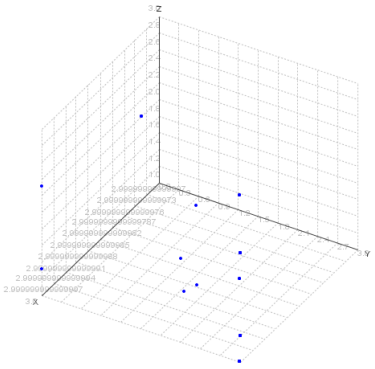

Figura 20: Taxa de aprendizado: 0.5517820314689292; Raio: 2; Pesos iniciais: E

Referências

[1] A. P. BRAGA A. P.; LUDERMIR T. B.; CARVALHO. *Redes Neurais Artificiais: Teoria e Aplicações. 2 ed.* LTC, Rio de Janeiro, 2007.

[2] Simon Haykin. *Redes Neurais: Princípios e Prática. 2 ed.* Bookman, Porto Alegre, 2001.

[3] Teuvo Kohonen. *Self-Organizing Maps.* Springer, 2000.