

Inteligência Artificial – ACH2016

Aula22 – Aprendizado Não-Supervisionado

K-Médias e Mapas Auto-Organizáveis

Norton Trevisan Roman
(norton@usp.br)

3 de junho de 2019

Aprendizado Não-Supervisionado

Agrupamentos

- Até agora, todos nossos dados de treinamento possuíam um rótulo ou valor associados a eles
 - Conhecíamos o valor da função objetivo para esses dados
 - Tínhamos uma medida clara do sucesso ou não de nosso algoritmo

Aprendizado Não-Supervisionado

Agrupamentos

- Até agora, todos nossos dados de treinamento possuíam um rótulo ou valor associados a eles
 - Conhecíamos o valor da função objetivo para esses dados
 - Tínhamos uma medida clara do sucesso ou não de nosso algoritmo
- E se esse não for o caso?

Agrupamentos

- Até agora, todos nossos dados de treinamento possuíam um rótulo ou valor associados a eles
 - Conhecíamos o valor da função objetivo para esses dados
 - Tínhamos uma medida clara do sucesso ou não de nosso algoritmo
- E se esse não for o caso?
- Mais, e se sequer saibamos que rótulos usar?
 - Apelamos ao **agrupamento** (*clustering*) dos dados

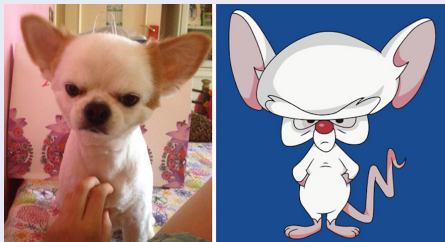
Análise de Agrupamentos

- A análise de agrupamentos busca agrupar objetos em subconjuntos (os agrupamentos)
- De modo a que objetos em um agrupamento estejam mais relacionados uns aos outros que a objetos de fora

Aprendizado Não-Supervisionado

Análise de Agrupamentos

- A análise de agrupamentos busca agrupar objetos em subconjuntos (os agrupamentos)
 - De modo a que objetos em um agrupamento estejam mais relacionados uns aos outros que a objetos de fora
- Central a isso é a noção de grau de similaridade entre dois objetos
 - Decisão bastante subjetiva



Fonte: http://earthporm.com/wp-content/uploads/2015/09/similar-things-look-alike-29__700.jpg

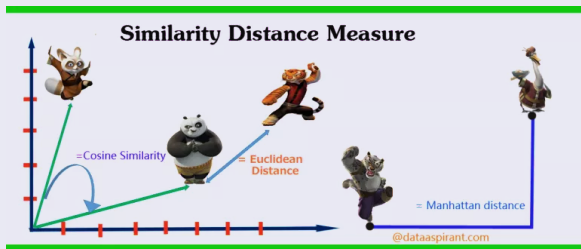
Similaridade

- Medida, em geral, como a distância entre pares de pontos

Aprendizado Não-Supervisionado

Similaridade

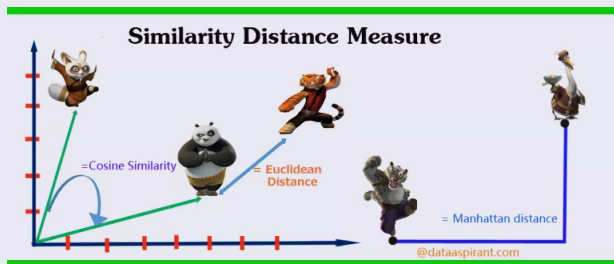
- Medida, em geral, como a distância entre pares de pontos
- Medidas comuns:
 - Distância Euclidiana
 - Distância Manhattan
 - Similaridade de Cosseno
 - etc



Fonte: [9]

Aprendizado Não-Supervisionado

Similaridade



Fonte: [9]

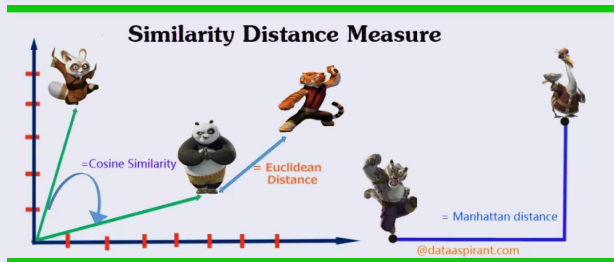
- Distância Euclidiana

$$d(\vec{x}_i, \vec{x}_j) = \|\vec{x}_i - \vec{x}_j\|^2 = \sum_{k=1}^n (x_{ik} - x_{jk})^2$$

(onde $\vec{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]$)

Aprendizado Não-Supervisionado

Similaridade



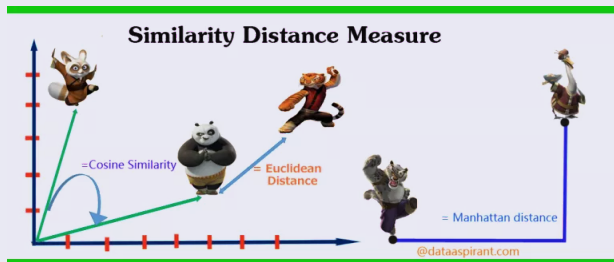
Fonte: [9]

- Distância Manhattan

$$d(\vec{x}_i, \vec{x}_j) = \|\vec{x}_i - \vec{x}_j\| = \sum_{k=1}^n (x_{ik} - x_{jk})$$

(onde $\vec{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]$)

Similaridade



Fonte: [9]

- Similaridade de Cosseno

$$\text{sim}(\vec{x}_i, \vec{x}_j) = \cos(\theta) = \frac{\vec{x}_i \cdot \vec{x}_j}{\|\vec{x}_i\| \|\vec{x}_j\|}$$

(onde $\vec{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]$)

K-Means

Funcionamento

- Técnica de agrupamento em que
 - Agrupamentos possuem pontos como representantes
 - O quadrado da distância euclidiana é adotado como medida de dissimilaridade entre vetores e representantes de grupos

Funcionamento

- Técnica de agrupamento em que
 - Agrupamentos possuem pontos como representantes
 - O quadrado da distância euclidiana é adotado como medida de dissimilaridade entre vetores e representantes de grupos
- *K-Means* cria um conjunto de exatamente k categorias → seus **centros** (ou **centróides**)
 - k previamente definido
 - Inicialmente, cada centro é escolhido aleatoriamente
 - O procedimento iterativamente move esses centros
 - Minimizando a variância total dentro do agrupamento

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: agrupamento

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

repita

para $i = 1$ até n **faça**

 Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

para $j = 1$ até k **faça**

 Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até *que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas*

retorna \mathcal{C}

K-Means

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: agrupamento

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

repita

para $i = 1$ até n **faça**

 Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

para $j = 1$ até k **faça**

 Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas

retorna \mathcal{C}

Base de dados



K-Means

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: **agrupamento**

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

Escolha k exemplos aleatoriamente para representar os k grupos

repita

para $i = 1$ até n **faça**

Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

para $j = 1$ até k **faça**

Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas

retorna \mathcal{C}

K-Means

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: **agrupamento**

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

repita

para $i = 1$ até n **faça**

Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

Associa x_i à categoria mais próxima p

para $j = 1$ até k **faça**

Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas

retorna \mathcal{C}

K-Means

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: agrupamento

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

repita

para $i = 1$ até n **faça**

Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

para $j = 1$ até k **faça**

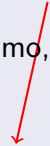
Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas

retorna \mathcal{C}

Recalcula a média
de cada categoria



K-Means

Funcionamento: Algoritmo

Função $K\text{-Means}(X = \{\vec{x}_1, \dots, \vec{x}_n\}, k)$: **agrupamento**

$\mathcal{C} \leftarrow \text{escolheAleatorio}(X, k)$

repita

para $i = 1$ até n **faça**

Determine o representante mais próximo, $c_p \in \mathcal{C}$, de \vec{x}_i

$b(i) \leftarrow p$

para $j = 1$ até k **faça**

Redefina c_j como a média dos vetores $\vec{x}_i \in X$ com

$b(i) = j$

até que nenhuma mudança ocorra em \mathcal{C} entre duas iterações sucessivas

retorna \mathcal{C}

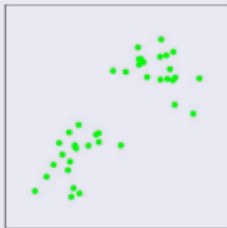
As k médias como representantes das k categorias

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento



Fonte: [12]

Começamos com um conjunto de pontos

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento

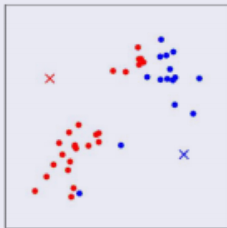


Fonte: [12]

Criamos então centros aleatórios

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento

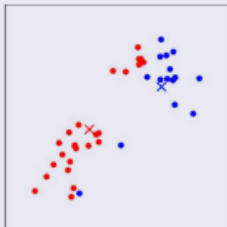


Fonte: [12]

Associamos cada exemplo
ao centro mais próximo

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento



Fonte: [12]

Então movemos cada centro para a média dos pontos em seu agrupamento

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento

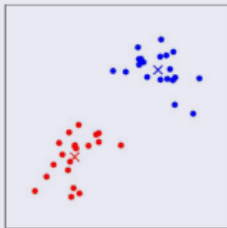


Fonte: [12]

Fazemos nova associação dos pontos ao centro mais próximo

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento

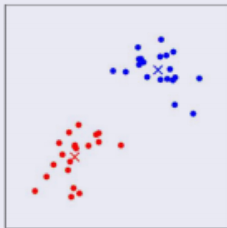


Fonte: [12]

E movimentamos novamente o centro para a média no novo agrupamento

Funcionamento

- K-means encontra os melhores centros alternando
 - Associação, baseada nos centros atuais, de pontos a agrupamentos
 - Escolha de novos centros a partir da atual associação de pontos ao agrupamento



Fonte: [12]

E assim sucessivamente, até que nenhum centro seja movido

Vantagens e Desvantagens

- Vantagens
 - Simplicidade computacional
 - Rápido

Vantagens e Desvantagens

- Vantagens
 - Simplicidade computacional
 - Rápido
- Desvantagens
 - Não é aplicável a dados categóricos (como ficaria a média ou a distância?)
 - Sensível a ruído ou *outliers* (por conta da distância euclidiana)
 - O valor de k precisa ser definido de antemão

K-Means: Variação

K-Medoids

- *K-means*:
 - Cada grupo é representado pela média de seus vetores

K-Means: Variação

K-Medoids

- *K-means*:
 - Cada grupo é representado pela média de seus vetores
- *K-medoids*:
 - Cada grupo é representado por um vetor selecionado dentre os elementos do conjunto de dados – o *medoid*
 - Recalculado a cada iteração do algoritmo

K-Means: Variação

K-Medoids

- *K-means*:
 - Cada grupo é representado pela média de seus vetores
- *K-medoids*:
 - Cada grupo é representado por um vetor selecionado dentre os elementos do conjunto de dados – o *medoid*
 - Recalculado a cada iteração do algoritmo
 - Além do *medoid*, cada grupo contém todos os vetores que:
 - Não são usados como *medoids* em outros grupos
 - Estão mais próximos de seu *medoid* que dos *medoids* dos demais grupos

K-Medoids

- Vantagens
 - Pode ser usado tanto com dados contínuos quanto discretos
 - *K-means* aceita somente contínuos, pelo fato da média de vetores não resultar necessariamente num ponto do domínio
 - Tende a ser menos sensível a *outliers*

K-Means: Variação

K-Medoids

- Vantagens
 - Pode ser usado tanto com dados contínuos quanto discretos
 - *K-means* aceita somente contínuos, pelo fato da média de vetores não resultar necessariamente num ponto do domínio
 - Tende a ser menos sensível a *outliers*
- Desvantagem
 - Calcular o *medoid* pode ser mais custoso que a média
 - Médias têm um significado geométrico e estatístico claro... *medoids* não

Self-Organizing Maps

Mapas Auto-Organizáveis

Aprendizado Competitivo

- Forma de aprendizado em redes neurais na qual os neurônios de saída da rede competem entre si para serem ativados
 - Apenas um neurônio, ou um por grupo, estará ativo por vez
 - Forma de disputa em que o vencedor fica com tudo

Mapas Auto-Organizáveis

Aprendizado Competitivo

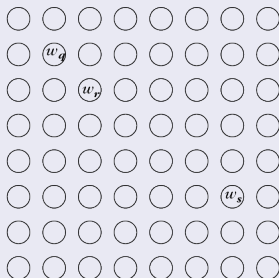
- Forma de aprendizado em redes neurais na qual os neurônios de saída da rede competem entre si para serem ativados
 - Apenas um neurônio, ou um por grupo, estará ativo por vez
 - Forma de disputa em que o vencedor fica com tudo
- A competição pode ser induzida a partir da topologia da rede
 - A atualização dos pesos força os neurônios a se **auto-organizarem**
 - Criando assim um **mapa auto-organizável**

Mapas Auto-Organizáveis

Topologia

- Os neurônios são topologicamente ordenados em uma malha
- Em geral uni ou bidimensional
- Não há conexões entre eles
- Dimensões maiores são possíveis mas incomuns

Malha Bidimensional



Malha Unidimensional



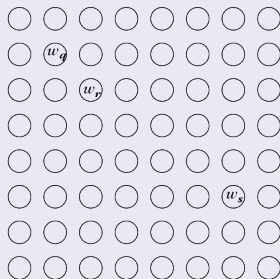
Fonte: PR. T&K.

Mapas Auto-Organizáveis

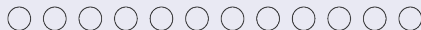
Topologia

- São então ajustados seletivamente a cada entrada
- Sua localização na malha fica organizada de modo a criar um sistema de coordenadas Φ para cada atributo da entrada

Malha Bidimensional



Malha Unidimensional

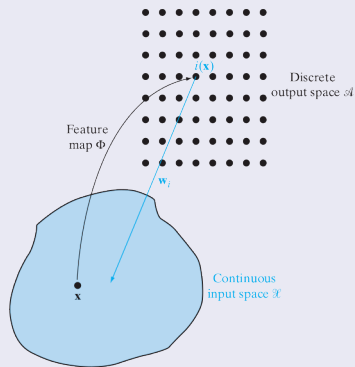


Fonte: PR. T&K.

Mapas Auto-Organizáveis

Topologia: Significado

- A malha torna-se então um **mapa** topográfico
- Em que a localização espacial de um neurônio na malha corresponde a um domínio ou característica particular dos dados de entrada

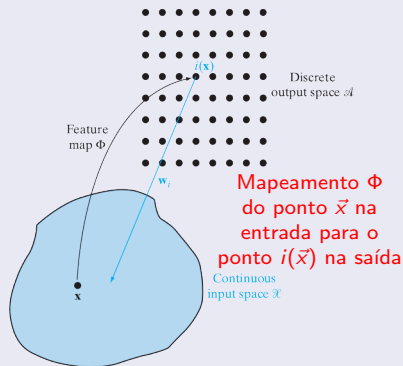


Fonte: NNaLM. Haykin.

Mapas Auto-Organizáveis

Topologia: Significado

- A malha torna-se então um **mapa** topográfico
- Em que a localização espacial de um neurônio na malha corresponde a um domínio ou característica particular dos dados de entrada

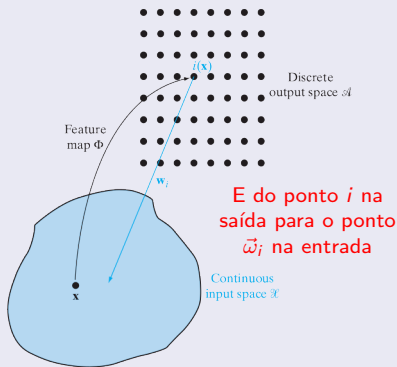


Fonte: NNaLM. Haykin.

Mapas Auto-Organizáveis

Topologia: Significado

- A malha torna-se então um **mapa** topográfico
- Em que a localização espacial de um neurônio na malha corresponde a um domínio ou característica particular dos dados de entrada

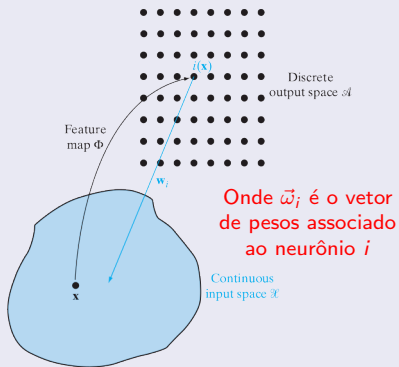


Fonte: NNaLM. Haykin.

Mapas Auto-Organizáveis

Topologia: Significado

- A malha torna-se então um **mapa** topográfico
- Em que a localização espacial de um neurônio na malha corresponde a um domínio ou característica particular dos dados de entrada

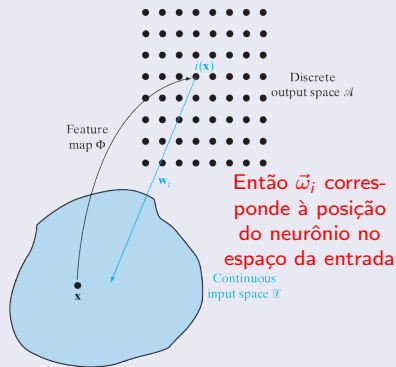


Fonte: NNaLM. Haykin.

Mapas Auto-Organizáveis

Topologia: Significado

- A malha torna-se então um **mapa** topográfico
- Em que a localização espacial de um neurônio na malha corresponde a um domínio ou característica particular dos dados de entrada

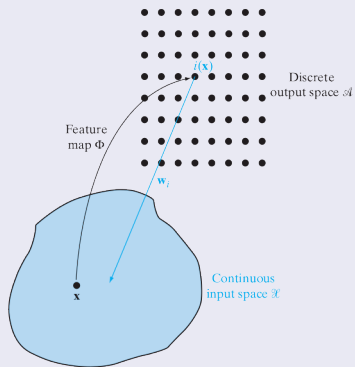


Fonte: NNaLM. Haykin.

Mapas Auto-Organizáveis

Topologia: Significado

- Cada neurônio representa um conjunto de pontos
- (Auto-)Arranjos de modo a que neurônios próximos correspondam a pontos próximos no espaço de entrada



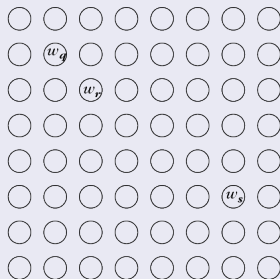
Fonte: NNLM. Haykin.

Mapas Auto-Organizáveis

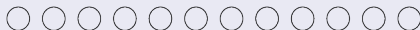
Topologia: Distância Topológica

- A ordenação topológica implica uma distância entre neurônios
- Ex: Distância Manhattan ou Euclidiana
- E \vec{w}_r está mais próximo topologicamente de \vec{w}_q que de \vec{w}_s

Malha Bidimensional



Malha Unidimensional

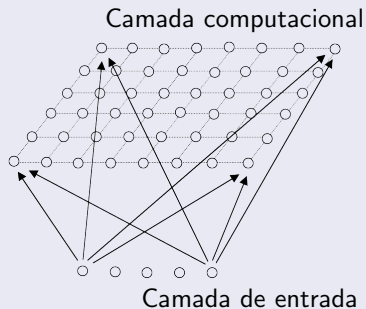


Fonte: PR. T&K.

Mapas Auto-Organizáveis

Redes de Kohonen

- Embora haja outros tipos, veremos apenas as redes de Kohonen
- Uma camada bidimensional de células (sem conexão direta entre elas)
- Cada neurônio conectado a todos os nós de entrada

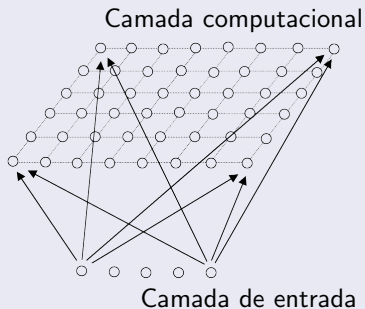


Fonte: SOM. Bullinaria.

Mapas Auto-Organizáveis

Redes de Kohonen

- Embora haja outros tipos, veremos apenas as redes de Kohonen
- Uma camada bidimensional de células (sem conexão direta entre elas)
- Cada neurônio conectado a todos os nós de entrada
- Sem camada de saída
 - Cada nó na malha é um nó de saída



Fonte: SOM. Bullinaria.

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ *ou* $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Conjunto de k exemplos de entrada

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Um exemplo de entrada
é um vetor no espaço d-
dimensional $\vec{x}_i = [x_{i1}, \dots, x_{id}]$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Haverá então d atributos
e consequentes d unidades
de entrada para a malha

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Número máximo de iterações

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Uma iteração é a análise de um exemplo de entrada

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \epsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Ponto a partir do qual consideramos a malha estável

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \epsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Taxa de aprendizado da rede

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Função de vizinhança. Define que pontos pertencem à vizinhança de um neurônio

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Note que tanto $\eta(t)$
quanto $h(j_m, j, t)$ depen-
dem do tempo (iteração) t

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até ($\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon$) ou ($t > t_{max}$)

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Inicializamos aleatoriamente os pesos $\vec{\omega}_j(t) = [\omega_{ij}(t) : 1 \leq i \leq d; 1 \leq j \leq L]$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Onde $\omega_{ij}(t)$ é o peso entre a unidade de entrada i e o neurônio j no instante t

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

E L é o número de neurônios na malha

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

A única restrição aqui é que $\vec{\omega}_j(0)$ seja diferente para $j = 1, 2, \dots, L$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Note que o vetor de pesos de cada neurônio tem as mesmas dimensões que o espaço de entrada

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Por isso podemos mapeá-lo diretamente a esse espaço

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$

$t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Outra forma de inicializar é selecionar aleatoriamente os $\vec{\omega}_j(0)$ a partir de valores nos vetores de entrada \vec{x}_i

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2 \leftarrow$ Calculamos a distância d_j entre a entrada $\vec{x}(t)$ e cada nó j na malha

$j_m \leftarrow$ Nó da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Selecionando o neurônio mais próximo ao exemplo de entrada \rightarrow o vencedor da disputa

$j_m \leftarrow$ *Nó da malha com o menor d_j*

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Também conhecido como
Best Matching Unit (BMU)

$j_m \leftarrow$ *Nó da malha com o menor* d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ *ou* $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ *ou* $(t > t_{max})$

Atualizamos os pesos
de j_m e seus vizinhos

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

$0 < \eta(t) < 1$ é um termo de ganho que decresce com o tempo (iteração), desacelerando assim a adaptação dos pesos

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$h(j_m, j, t)$ também decresce em tamanho com o tempo, deixando menor a vizinhança considerada

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ *ou* $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Então, a cada iteração, desaceleramos a adaptação dos pesos e reduzimos a vizinhança de j_m

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (inclusive j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j na malha **faça**

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ ou $(t > t_{max})$

E na medida em que j_m é trazido para mais perto de $\vec{x}(t)$, seus vizinhos também o são, com vizinhos mais afastados se movendo menos

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Passamos ao próximo exemplo

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até $(\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon)$ *ou* $(t > t_{max})$

Redes de Kohonen

Algoritmo

Função $SOM(X = \{\vec{x}_1, \dots, \vec{x}_k\}, t_{max}, \varepsilon, \eta(t), h(j_m, j, t))$: mapeamento

$\vec{\omega}_j(0) \leftarrow$ Escolha aleatoriamente $[\omega_{ij}(0)]$ pequenos, $1 \leq i \leq d$ e $1 \leq j \leq L$
 $t \leftarrow 0$

repita

$\vec{x}(t) \leftarrow$ Escolha aleatoriamente um exemplo $[x_1(t), \dots, x_d(t)] \in X$

para cada *Nó* j *na malha faça*

$$d_j \leftarrow \sum_{i=1}^d (x_i(t) - \omega_{ij}(t))^2$$

Repetimos até não haver mudança perceptível no mapa (as posições dos neurônios se estabilizaram)

$j_m \leftarrow$ *Nó* da malha com o menor d_j

para cada *Nó* $j \in h(j_m, j, t)$ (*inclusive* j_m) **faça**

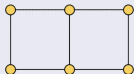
$$\vec{\omega}_j(t+1) \leftarrow \vec{\omega}_j(t) + \eta(t)h(j_m, j, t)(\vec{x}(t) - \vec{\omega}_j(t))$$

$t \leftarrow t + 1$

até ($\|\vec{W}_t - \vec{W}(t-1)\| < \varepsilon$) *ou* ($t > t_{max}$)

Redes de Kohonen

Funcionamento

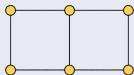


Fonte: [16]

Posicione os neurônios no espaço de dados (gerando \vec{w}_j aleatório)

Redes de Kohonen

Funcionamento

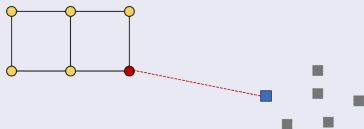


Fonte: [16]

Selecione um exemplo de entrada

Redes de Kohonen

Funcionamento

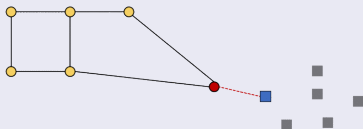


Fonte: [16]

Identifique o neurônio
mais próximo (BMU)

Redes de Kohonen

Funcionamento

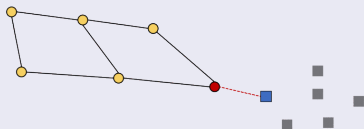


Fonte: [16]

Mova-o para mais
próximo do exemplo

Redes de Kohonen

Funcionamento

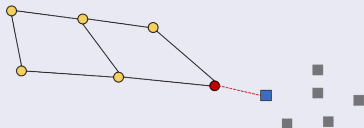


Fonte: [16]

Mova também seus vizinhos,
com vizinhos mais afas-
tados se movendo menos

Redes de Kohonen

Funcionamento



Fonte: [16]

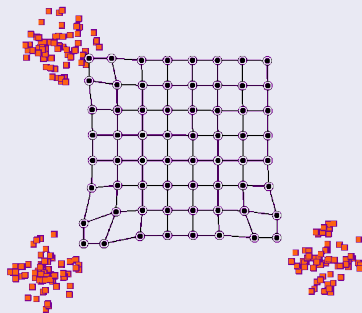
Mova também seus vizinhos,
com vizinhos mais afas-
tados se movendo menos

Fonte: [17]
(Requer Adobe Acrobat Reader)

Redes de Kohonen

Funcionamento

- Os nós em torno do vencedor j_m são levados com ele
- Ao longo do ciclo de treino, convergem para uma representação “média” da classe

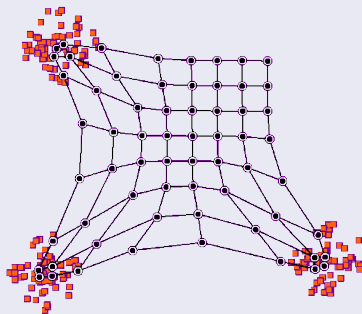


Fonte: [15]

Redes de Kohonen

Funcionamento

- Os nós em torno do vencedor j_m são levados com ele
- Ao longo do ciclo de treino, convergem para uma representação “média” da classe

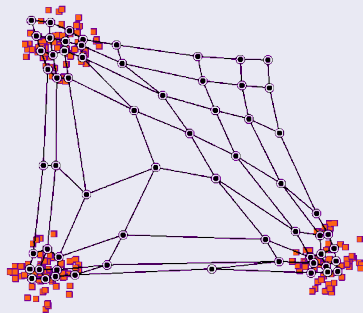


Fonte: [15]

Redes de Kohonen

Funcionamento

- Os nós em torno do vencedor j_m são levados com ele
- Ao longo do ciclo de treino, convergem para uma representação “média” da classe

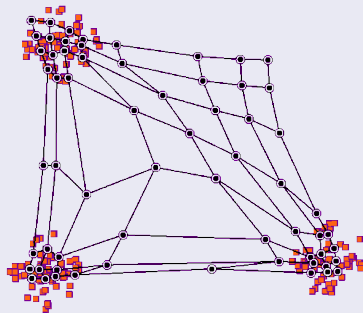


Fonte: [15]

Redes de Kohonen

Funcionamento

- Os nós em torno do vencedor j_m são levados com ele
- Ao longo do ciclo de treino, convergem para uma representação “média” da classe
- Gradualmente se juntando em áreas de alta densidade de pontos
 - E assim refletindo agrupamentos nos dados

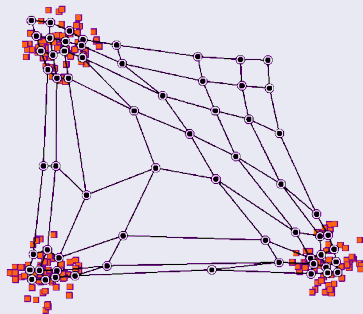


Fonte: [15]

Redes de Kohonen

Funcionamento

- Assim, vetores próximos dos exemplos de treino serão agrupados corretamente
- Mesmo que a rede nunca os tenha visto



Fonte: [15]

Neurônio

- Diferentemente do perceptron, o neurônio do SOM não possui função de ativação
- A decisão sobre que neurônio será ativado vem da distância ao dado de entrada

Neurônio

- Diferentemente do perceptron, o neurônio do SOM não possui função de ativação
- A decisão sobre que neurônio será ativado vem da distância ao dado de entrada
- Da mesma forma, os pesos não são separados dos nós de saída
 - Eles pertencem ao próprio neurônio
 - Funcionam como coordenadas do neurônio no espaço da entrada (um mapa)

Neurônio: Resposta da rede

- Depende da aplicação, pode ser
 - O índice do neurônio vencedor \rightarrow um indicativo da classe
 - O vetor de pesos $\vec{\omega}_{jm}$ mais próximo da entrada
 - A classe pode então ser representada pelos valores nesse vetor
 - Ex: se o vetor tiver 3 posições, podemos interpretá-lo como uma cor (RGB) – ver [11] para um bom exemplo

Neurônio: Resposta da rede

- Depende da aplicação, pode ser
 - O índice do neurônio vencedor \rightarrow um indicativo da classe
 - O vetor de pesos $\vec{\omega}_{j_m}$ mais próximo da entrada
 - A classe pode então ser representada pelos valores nesse vetor
 - Ex: se o vetor tiver 3 posições, podemos interpretá-lo como uma cor (RGB) – ver [11] para um bom exemplo
- Validação
 - Uma possível validação é verificar se o agrupamento formado pelos múltiplos pontos que mapeiam a um mesmo neurônio faz sentido

Redes de Kohonen

Definindo $h(j_m, j, t)$

- O neurônio vencedor j_m define o centro de uma vizinhança topológica de neurônios cooperativos j

Redes de Kohonen

Definindo $h(j_m, j, t)$

- O neurônio vencedor j_m define o centro de uma vizinhança topológica de neurônios cooperativos j
- Como definir uma vizinhança neurobiologicamente correta?
 - Existe evidência para interação lateral entre neurônios
 - Um neurônio ativado tende a excitar seus vizinhos imediatos mais que os que estão mais longe

Definindo $h(j_m, j, t)$

- O neurônio vencedor j_m define o centro de uma vizinhança topológica de neurônios cooperativos j
- Como definir uma vizinhança neurobiologicamente correta?
 - Existe evidência para interação lateral entre neurônios
 - Um neurônio ativado tende a excitar seus vizinhos imediatos mais que os que estão mais longe
- Precisamos então de uma vizinhança $h(j_m, j)$ que decaia suavemente com a distância lateral

Definindo $h(j_m, j, t)$

- $h(j_m, j)$ deve então
 - Ser simétrica em torno de j_m , onde $d_{j_m, j_m} = 0$
 - Decrescer sua amplitude monotonicamente com a distância lateral na malha $d_{j_m, j}$, indo a zero quando $d_{j_m, j} \rightarrow \infty$
 - $d_{j_m, j} = \|\vec{c}_{j_m} - \vec{c}_j\|$, onde \vec{c}_j é a posição de j na malha

Definindo $h(j_m, j, t)$

- $h(j_m, j)$ deve então
 - Ser simétrica em torno de j_m , onde $d_{j_m, j_m} = 0$
 - Decrescer sua amplitude monotonicamente com a distância lateral na malha $d_{j_m, j}$, indo a zero quando $d_{j_m, j} \rightarrow \infty$
 - $d_{j_m, j} = \|\vec{c}_{j_m} - \vec{c}_j\|$, onde \vec{c}_j é a posição de j na malha
- Escolhemos a Gaussiana

$$h(j_m, j) = \exp\left(-\frac{d_{j_m, j}^2}{2\sigma^2}\right)$$

onde σ é a largura da vizinhança topológica, medindo o grau com que os neurônios vizinhos participam no aprendizado

Definindo $h(j_m, j, t)$

- Além disso, a vizinhança deve encolher com o tempo

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

onde σ_0 é o valor de σ no início do algoritmo (em $t = 0$), e τ_1 é uma constante a ser definida em projeto

Definindo $h(j_m, j, t)$

- Além disso, a vizinhança deve encolher com o tempo

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

onde σ_0 é o valor de σ no início do algoritmo (em $t = 0$), e τ_1 é uma constante a ser definida em projeto

- E $h(j_m, j, t) = \exp\left(-\frac{d_{j_m, j}^2}{2\sigma^2(t)}\right)$
 - Assim, a cada iteração a vizinhança de cada nó encolhe exponencialmente até possuir um único neurônio

Definindo $h(j_m, j, t)$

- Em $t = 0$, $h(j_m, j, t)$ deveria incluir quase todos os neurônios na malha
 - E então reduzir com o tempo

Definindo $h(j_m, j, t)$

- Em $t = 0$, $h(j_m, j, t)$ deveria incluir quase todos os neurônios na malha
 - E então reduzir com o tempo
- Podemos então fazer σ_0 ser igual ao “raio” da malha (em uma malha bidimensional)
 - E $\tau_1 = \frac{1000}{\log(\sigma_0)}$

Definindo $\eta(t)$

- A taxa de aprendizado também varia no tempo:

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

onde η_0 é seu valor inicial e τ_2 outra constante para o algoritmo

- Em geral, $\eta_0 = 0.1$ e $\tau_2 = 1000$

Definindo $\eta(t)$

- A taxa de aprendizado também varia no tempo:

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

onde η_0 é seu valor inicial e τ_2 outra constante para o algoritmo

- Em geral, $\eta_0 = 0.1$ e $\tau_2 = 1000$
- Trata-se de uma heurística, como $h(j_m, j, t)$
 - Tanto σ_0 quanto η_0 e τ_2 , bem como o melhor número de neurônios na malha devem ser definidos via validação

Vantagens

- Não-supervisionado
 - Nenhuma resposta precisa ser especificada para uma dada entrada

Vantagens

- Não-supervisionado
 - Nenhuma resposta precisa ser especificada para uma dada entrada
- Sem função de ativação ou conexão com outros neurônios
 - Nenhuma derivada precisa ser calculada, como no *gradient descent*
 - A única conexão é a que surge é o “empurra e puxa” dos nós e BMUs baseada no raio da vizinhança de cada BMU

Desvantagens

- Assim como no k-vizinhos e k-médias, SOM usa distância
- Também aqui atributos em diferentes escalas podem influir na precisão
- Também aqui a solução passa por uma normalização de todas as variáveis, criando assim uma escala uniforme

Desvantagens

- Assim como no k-vizinhos e k-médias, SOM usa distância
 - Também aqui atributos em diferentes escalas podem influir na precisão
 - Também aqui a solução passa por uma normalização de todas as variáveis, criando assim uma escala uniforme
- Não lida bem com variáveis categóricas
 - Pois assume o espaço dos dados como contínuo
 - Fica difícil o mapeamento dos $\vec{\omega}_j$ de volta ao espaço dos dados

Referências

- ① Russell, S.; Norvig P. (2010): Artificial Intelligence: A Modern Approach. Prentice Hall. 2a e 3a ed.
- ② Haykin, S. (2009): Neural Networks and Learning Machines. Pearson. 3 ed.
- ③ Hastie, T; Tibshirani, R.; Friedman, J. (2017): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer. 2 ed.
- ④ Alpaydm, E. (2010): Introduction to Machine Learning. MIT Press. 2 ed.
- ⑤ Theodoridis, S.; Koutroumbas, K. (2009): Pattern Recognition. Academic Press. 4 ed.
- ⑥ Michie, D.; Spiegelhalter, D.J.; Taylor, C.C. (1994): Machine Learning, Neural and Statistical Classification.

Referências

- 7 Beale, R.; Jackson, T. (1990): Neural Computing: An Introduction. Adam Hilger.
- 8 Kohonen, T. (1990): The Self-Organizing Map. Proceedings of the IEEE, 78(9). <https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1990-Kohonen-PIEEE.pdf>
- 9 <http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/>
- 10 https://en.wikipedia.org/wiki/K-means_clustering
- 11 <https://www.superdatascience.com/blogs/the-ultimate-guide-to-self-organizing-maps-soms>
- 12 <https://stanford.edu/~cpiech/cs221/handouts/kmeans.html>

Referências

- 13 Bullinaria, J.A. (2004): Self Organizing Maps: Fundamentals.
<http://www.cs.bham.ac.uk/~jxb/NN/l16.pdf>
- 14 <https://towardsdatascience.com/self-organizing-maps-ff5853a118d4>
- 15 https://en.wikipedia.org/wiki/Self-organizing_map
- 16 <https://algobeans.com/2017/11/02/self-organizing-map/>
 - Código em R: <https://github.com/algobeans/Self-Organizing-Map/blob/master/analysis.R>
- 17 https://pt.wikipedia.org/wiki/Mapas_de_Kohonen