Clustering de múltiplas tabelas de dissimilaridade com pesos adaptativos

Anderson B. S. Dantas[1], Francisco de A. T. de Carvalho[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (CIn/UFPE)
Av. Prof. Luiz Freire, s/n - Cidade Universitária, CEP 50740-540, Recife – PE – Brazil

{absd,fatc}@cin.ufpe.br

Abstract.

Resumo.

1. Introdução

2. Self-organizing map (SOM)

A classe de métodos chamados *self-organizing maps* (mapas auto-organizáveis) segue uma série de procedimentos para associar um número finito de vetores de objetos (entradas) com um número finito de pontos representativos, tal que as relações de semelhança entre as entradas são respeitadas por esses pontos. O algoritmo SOM representa uma abordagem de aprendizado não-supervisionado, todas as propriedades de cada cluster são estimados ou aprendidos sem o uso de informação a priori [Murtagh and Hernández-Pajares 1995].

O SOM consiste em um conjunto de neurônios organizados topologicamente em uma grade, geralmente com uma, duas, ou três dimensões, chamada mapa. De modo mais formal, o mapa é descrito por um grafo não-orientado (C, Γ). C é um conjunto de m neurônios interconectados com topologia definida por Γ. A estrutura de grafo permite que se defina uma função de distância entre dois neurônios no mapa. Para cada par de neurônios (*c,r*) no mapa, $\delta(c, r)$ é o comprimento do caminho mais curto entre *c* e *r* no grafo *C*. Essa função de distância permite que haja uma relação de vizinhança entre neurônios.

O conceito de vizinhança é incorporado ao algoritmo através do uso de uma função *kernel* K positiva e tal que $\lim_{|x| \to \infty} K(x) = 0$ [Badran et al. 2005]. As distâncias $\delta(c, r)$ entre os neurônios *c* e *r* do mapa permitem a definição da influência relativa dos neurônios sobre os objetos. $K(\delta(c, r))$ quantifica essa influência. A região ao redor do neurônio que melhor representa um dado objeto de entrada é modificada para se aproximar mais do objeto apresentado, ou seja, não só o neurônio vencedor é atualizado, como também seus vizinhos no mapa. O resultado é que os neurônios no mapa ficam ordenados, neurônios vizinhos possuem vetores referência (ou protótipos) similares.

O SOM recebe como entrada um conjunto de vetores rotulados e tem como resultado um vetor de neurônios com os vetores de entrada atrelados. Seja n o número de objetos $x_i \in R^p$ e $x_i \in E$(conjunto de dados de entrada), $i = 1, 2, \ldots, n$, onde cada objeto é formados por um vetor e cada vetor x_i é identificado por um rótulo. O algoritmo básico SOM segue as seguintes etapas:

1. Fixe os valores de m, $\eta(0)$ (taxa de aprendizado inicial), N_{iter} (número máximo de iterações) e $h_{\delta(j,l)}(0)$ (função de vizinhança inicial, onde $\delta(j,l)$ é uma função de distância fixa entre os neurônios j e l);
2. Inicialização: selecione aleatoriamente m vetores distintos para protótipos iniciais $w_c(0), c = 1, 2, \ldots, m$ e faça $t = 1$;
3. Selecione aleatoriamente um vetor de entrada $x_i(t)$;
4. Encontre o melhor neurônio (vencedor) $w_c(t)$ em relação à menor distância Euclidiana:
$$f(x_i(t))^T = \min_{1 \leq j \leq m} \sum_{k=1}^{p} (x_{ik}(t) - w_{jk}(t))^2;$$
5. Para todos os neurônios j dentro do raio de vizinhança do neurônio vencedor w_c, ajuste os pesos segundo a equação:
$$w_j(t+1) = w_j(t) + \eta(t) h_{\delta(w_c, w_j)}(t)(x_i(t) - w_j(t));$$
6. Atualize a taxa de aprendizado $\eta(t)$ e a função de vizinhança $h_{l,j}(t)$;
7. Critério de parada: se $t = N_{iter}$, pare; senão vá para 3.

3. Batch self-organizing map aplicado em dados de dissimilaridade

O SOM para dados de dissimilaridade conforme proposto por [Golli et al. 2004] também é descrito por um grafo (C, Γ). A principal diferença está nos dados que serão classificados que consistem em um conjunto arbitrário onde uma dissimilaridade é definida.

Cada neurônio c é representado por um "*individual referent*"$a_c = x_{j_i},\ x_{j_i} \in \Omega$. Denotamos a *individuals codebook*, i.e. a lista $a = \{a_c; c = 1, \ldots, m\}$ de *individual referents* do mapa.

Uma nova dissimilaridade d^T de $\Omega \times P(\Omega)$ para $\mathbb{R}$ é definida por:

$$d^T(x_i, a_c) = \sum_{r \in C} K^T(\delta(c, r))\, d^2(x_i, a_r) \tag{1}$$

Tal dissimilaridade é baseada na função kernel positiva K, como descrita na seção anterior. K é usada para definir uma família de funções K^T parametrizadas por T, onde $K^T(\delta) = K(\frac{\delta}{T})$. O parâmetro T controla o tamanho da vizinhança dos neurônios: se o valor de T for pequeno, poucos neurônios serão atingidos pela vizinhança. Um simples exemplo para K^T é definido por $K^T(\delta) = e^{-\frac{\delta^2}{T^2}}$.

Durante a etapa de aprendizado, a seguinte função custo E é minimizada alternando-se o passo de afetação e o passo de representação:

$$E(f, a) = \sum_{x_i \in \Omega} d^T(x_i, a_{f(x_i)}) = \sum_{x_i \in \Omega} \sum_{r \in C} K^T(\delta(f(x_i), r)) d^2(x_i, a_r) \tag{2}$$

Na etapa de afetação, a função f afeta cada indivíduo x_i ao neurônio mais próximo. Em termos da dissimilaridade d^T:

$$f(x_i) = arg \min_{c \in C} d^T(x_i, a_c) \tag{3}$$

Na etapa de representação, são selecionados novos protótipos que representam o conjunto de objetos. Esta etapa de otimização pode ser realizada independentemente para cada neurônio. De fato, as seguintes m funções são minimizadas:

$$E_r = \sum_{x_i \in \Omega} K^T(\delta(f(x_i), r))d^2(x_i, a_r) \tag{4}$$

O Algoritmo

Inicialização: iteração $k = 0$; selecione os protótipos iniciais a^0; fixe $T = T_{max}$ e o número total de iterações N_{iter}
Iteração: Na iteração k, o conjunto de protótipos da iteração anterior a^{k-1} é conhecido. Calcule o novo valor de T:

$$T = T_{max} * (\frac{T_{min}}{T_{max}})^{\frac{k}{N_{iter}-1}}$$

- **etapa de afetação**: Afetar cada indivíduo X_i ao protótipo como definido na equação 3.
- **etapa de representação**: determinar os novos protótipos a^k* que minimizam a função $E(f_{a^k}, a)$, a_c^k* é definido pela equação 4.

Repetir **Iteração** até $T = T_{min}$.

4. Self-organizing map adaptativo para múltiplas tabelas de dissimilaridade

4.1. Introdução

Nesta seção apresentamos um algoritmo baseado no algoritmo SOM para dados de dissimilaridade [Golli et al. 2004] e no algoritmo de *clustering* de múltiplas tabelas de dissimilaridade de [Lechevallier et al. 2010]. O objetivo do modelo apresentado é mapear objetos levando em conta suas descrições relacionais dadas por múltiplas matrizes de dissimilaridade. O algoritmo é capaz de associar dados de entrada com pontos representativos de um mapa formando grupos (*clusters*) de objetos.

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos e p o número de matrizes de dissimilaridade $\mathbf{D}_j = [d_j(e_i, e_l)](j = 1, \ldots, p)$, onde $d_j(e_i, e_l)$ é a dissimilaridade entre os objetos e_i e $e_l(i, l = 1, \ldots, n)$ na matriz de dissimilaridades $\mathbf{D}_j$. Assuma que o protótipo g_l do *cluster* C_l é um elemento do conjunto de objetos E, i.e., $g_l \in E \; \forall l = 1, \ldots, L$.

O algoritmo SOM ponderado para múltiplas tabelas de dissimilaridade calcula uma partição $P = (C_1, \ldots, C_l)$ de E em L clusters e o respectivo protótipo g_l representando o cluster C_l em P tal que um determinado critério de adequação (função objetivo) que mede o ajuste entre os clusters e seus repectivos protótipos é localmente otimizado. O critério de adequação é definido como:

$$
\begin{aligned}
E &= \sum_{i=1}^{n}\sum_{l=1}^{c} K(\delta(f(e_i), l), T) D_{\lambda_l}(e_i, g_l) \\
&= \sum_{i=1}^{n}\sum_{l=1}^{c} \exp\{-\frac{(\delta((f(e_i)), l))^2}{2T^2} \sum_{j=1}^{p} \lambda_{lj} d_j(e_i, g_l)
\end{aligned} \quad (5)
$$

onde K é uma função kernel positiva, tal que $\lim_{|\delta|\to\infty} K(\delta) = 0$. E $\delta(c, r)$ calcula a distância entre dois neurônios no mapa. A matriz de pesos λ é composta por L vetores de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^j, \ldots, \lambda_k^p)$, muda a cada iteração do algoritmo, ou seja, os vetores não possuem valores determinados absolutamente e é diferente para cada matriz de dissimilaridades. O modelo segue três etapas definidas (representação, atualização de pesos e afetação):

Etapa 1: Definição dos melhores protótipos

Nesta etapa, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos. O protótipo $g_l^{(t)} = g^* \in E$ do cluster C_l, que minimiza o critério de adequação E é calculado segundo a equação:

$$
g^* = argmin_{e\in E} \sum_{i=1}^{n} \exp\{-\frac{(\delta((f(e_i))^{(t-1)}, l))^2}{2T^2} \sum_{j=1}^{p} \lambda_{lj}^{(t-1)} d_j(e_i, e) \quad (6)
$$

Etapa 2: Definição dos melhores pesos

Nesta etapa, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ são fixos. O elemento j do vetor de pesos $\lambda_l = (\lambda_k^1, \ldots, \lambda_k^p)$, que minimiza o critério de adequação E é calculado segundo a expressão:

$$
\lambda_{lj}^{(t)} = \frac{\left\{\prod_{h=1}^{p}(\sum_{i=1}^{n} \exp\{-\frac{\delta^2(f^{(t-1)}(x_i), l)}{2T^2}\} d_h(e_i, g_l^{(t)}))\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} \exp\{-\frac{\delta^2(f^{(t-1)}(x_i), l)}{2T^2}\} d_j(e_i, g_l^{(t)})} \quad (7)
$$

Etapa 3: Definição da melhor partição

Nesta etapa, os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos. O cluster C_r que minimiza o critério E é determinado de acordo com a equação:

$$
r = (f(e_i))^{(t)} = argmin_{1\leq h\leq c} \sum_{l=1}^{c} \exp\{-\frac{(\delta(h, l))^2}{2T^2}\} \sum_{j=1}^{p} \lambda_{lj}^{(t)} d_j(e_i, g_l^{(t)}) \quad (8)
$$

4.2. O algoritmo

1. *Inicialização*
   Fixe o número c de clusters;
   Fixe δ;
   Fixe a função kernel K;
   Fixe o número de iterações N_{iter};
   Fixe T_{min}, T_{max}; Atribua $T \leftarrow T_{max}$; Atribua $t \leftarrow 0$;
   Selecione c protótipos aleatoriamente $g_l^{(0)} \in E(l = 1, \ldots, c$;
   Configure $\lambda_l^{(0)} = (\lambda_{l1}^{(0)}, \ldots, \lambda_{lp}^{(0)}) = (1, \ldots, 1)(l = 1, \ldots, c)$;
   Configure o mapa $L(c, \mathbf{G}^0)$, onde $\mathbf{G}^0 = (g_1^{(0)}, \ldots, g_c^{(0)})$.
   Cada objeto e_i é afetado ao protótipo mais próximo com o objetivo de obter a partição $P^{(0)} = (P_1^{(0)}, \ldots, P_c^{(0)})$ de acordo com o seguinte critério:

$$(f(e_i))^{(0)} = argmin_{1 \leq r \leq c} \sum_{l=1}^{c} \exp\{-\frac{(\delta(r,l))^2}{2T^2}\} \sum_{j=1}^{p} \lambda_{lj}^{(0)} d_j(e_i, g_l) \qquad (9)$$

2. *Passo 1: Representação*
   Atribua $t = t + 1$;
   Atribua $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos.
   Calcule o protótipo $g_l^{(t)} = g^* \in E$ do cluster $P_l^{(t-1)}(l = 1, \ldots, c)$ de acordo com a equação 6.
3. Passo 2: Atualização dos pesos
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ são fixos.
   Atualize o vetor de pesos de acordo com a equação 7.
4. *Passo 3: Afetação*
   Os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos
   $P^{(t)} \leftarrow P^{(t-1)}$
   para $i = 1$ até n faça:
   encontre o cluster $C_m^{(t)}$ ao qual o objeto e_i pertence
   encontre o cluster vencedor $C_r^{(t)}$, onde r é obtido pela equação 8

   se $r \neq m$:
   $C_r^{(t)} \leftarrow C_r^{(t)} \cup \{e_i\}$
   $C_m^{(t)} \leftarrow C_m^{(t)} \setminus \{e_i\}$

5. *Critério de parada*
   Se $T == T_{min}$ então PARE; senão volte ao passo 1

5. Experimentos e resultados

Para provar a utilidade do algoritmo proposto, foram realizados experimentos utilizando bases de dados obtidas do repositório UCI Machine Learning Repository [Frank and Asuncion 2010]. Com a finalidade de avaliar os resultados da clusterização realizada pelo método proposto neste trabalho foi considerado um índice externo:

o índice de Rand corrigido (CR) [Hubert and Arabie 1985], a medida F-*measure* [van Rijsbergen 1979] e a taxa de erro global de classificação (OERC - overall error rate of classification) [Breiman et al. 1984].

O índice CR avalia o grau de similaridade entre a partição a priori e a partição fornecida pelo algoritmo de clustering. Além disso, o índice CR não é sensível ao número de classes nas partições ou à distribuição dos itens dentro dos clusters. O índice CR pode assumir valores no intervalo [-1,1], onde o valor 1 indica perfeita combinação entre as partições, valores perto de 0 ou negativos indicam pouca semelhança. O índice F-*measure* entre a partição a priori e a partição obtida pelo algoritmo de clustering assume valores no intervalo [0,1], em que 1 indica perfeita combinação entre as partições. O índice OERC mede a habilidade de um algoritmo de clustering encontrar classes a priori presentes na base de dados.

5.1. Base de dados Iris

Esta base de dados consiste em três tipos (classes) de planta íris: iris setosa, iris versicolor e iris virginica. Cada uma das classes possui 50 exemplos. Cada instância é descrita por quatro atributos, que são o comprimento (length) e a largura (width) da sépala (sepal) e da pétala (petal), em centímetros.

Cluster/Classe	1	2	3

6. Conclusão

Referências

Badran, F., Yacoub, M., and Thiria, S. (2005). Self-organizing maps and unsupervised classification. In *Neural networks: methodology and applications*, pages 379 – 442. Springer-Verlag.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Frank, A. and Asuncion, A. (2010). Uci machine learning repository.

Golli, A. E., Conan-Guez, B., and Rossi, F. (2004). A self-organizing map for dissimilarity data. *Classification, Clustering, and Data*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

Lechevallier, Y., de Carvalho, F. A. T., Despeyroux, T., and de Melo, F. M. (2010). Clustering of Multiple Dissimilarity Data Tables for Documents Categorization. *Proceedings of COMPSTAT'2010*.

Murtagh, F. and Hernández-Pajares, M. (1995). The Kohonen Self-Organizing Map Method: An Assessment. *Journal of Classification*.

van Rijsbergen, C. J. (1979). *Information Retrieval*. London: Butterworths.