Universidade Federal de São Carlos – Departamento de Computação
Reconhecimento de padrões – Turma A – 2021
Prof. Alexandre Luis Magalhães Levada

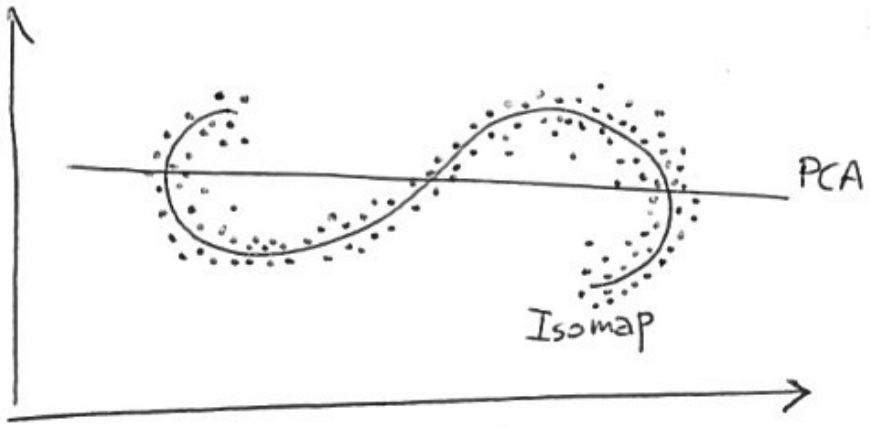
Aula 7 – O algoritmo ISOMAP para aprendizado de variedades

Isometric Feature Mapping (ISOMAP)

Ideia geral: Construir um grafo unindo os vizinhos mais próximos, computar os menores caminhos entre cada par de vértices e, conhecendo as distâncias entre os pontos, encontrar um mapeamento para o plano que preserve essas distâncias.

Hipótese: caminhos mínimos em grafos podem aproximar bem as verdadeiras distâncias geodésicas nos espaços não Euclidianos (variedades)

Aprendizado de métricas: Métodos lineares falham em aprender uma medida de distância adequada na presença de não linearidades nos dados. Distância Euclidiana não é uma medida adequada nesse caso.



O algoritmo ISOMAP pode ser resumido em 3 grandes passos:

Passo1: Induzir um grafo a partir do conjunto de dados $\{\vec{x}_i, y_i\}$ para i = 1,2,…,n onde $\vec{x}_i$ denota o vetor de características que representa a i-ésima amostra e y_i denota a classe ou categoria a que o vetor pertence, sendo geralmente um inteiro maior que zero.

Passo 2: Montar a matriz de distâncias ponto a ponto D
Para cada amostra $\vec{x}_i$ do conjunto
    Aplicar o algoritmo de Dijkstra para obter os caminhos mínimos de $\vec{x}_i$ aos demais
    Fazer D_{ij} igual ao tamanho do menor caminho entre $\vec{x}_i$ e $\vec{x}_j$

Passo 3: De posse da matriz D, encontrar um conjunto de pontos no subespaço Euclidiano R^k tal que as distâncias sejam preservadas. Esse problema é solucionado pelo algoritmo MDS (Multidimensional Scaling)

Obs: Trata-se de uma abordagem global (utiliza todos os pontos para estimar as distâncias)

Teorema: (*Asymptotic Convergence Theorem*)
Dados $\lambda_1, \lambda_2, \mu > 0$ tão pequenos quanto desejados, então para uma densidade suficientemente grande de amostras a desigualdade a seguir:

$$(1-\lambda_1) d_M(\vec{x}_i, \vec{x}_j) \leq d_G(\vec{x}_i, \vec{x}_j) \leq (1+\lambda_2) d_M(\vec{x}_i, \vec{x}_j)$$

é satisfeita com probabilidade $(1-\mu)$. Em outras palavras, esse resultado nos diz que a distância estimada no grafo (d_G) fica limitada num intervalo muito estreito em relação a verdadeira distância na variedade/manifold (d_M), ou seja d_G tende a d_M .

Como estimar as distâncias geodésicas?

Def: Caminho ótimo
Seja G = (V, E) e $w: E \rightarrow R^+$ uma função de custo para as arestas. Um caminho P* de v_0 a v_n é ótimo se seu peso

$$w(P^*) = \sum_{i=0}^{n-1} w(v_i, v_{i+1}) = w(v_0, v_1) + w(v_1, v_2) + ... + w(v_{n-1}, v_n)$$ (soma dos pesos das arestas)

é o menor possível, ou seja, o w(P*) é a distância geodésica de v_0 a v_n.

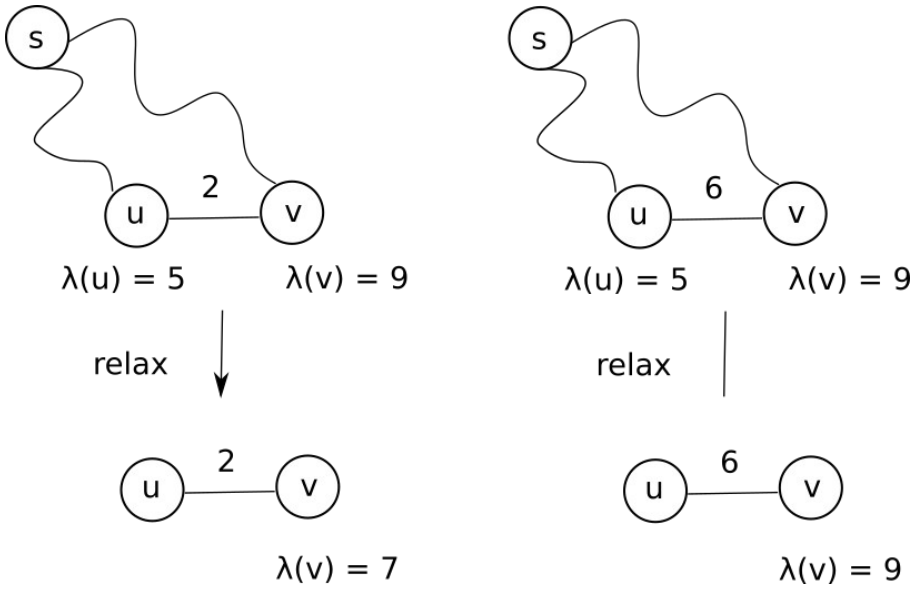
Antes de introduzirmos o algoritmo de Dijkstra, iremos apresentar a primitiva relax. Ela aplica a operação conhecida como relaxamento de uma aresta em um grafo ponderado.

A primitiva relax

relax(u, v, w): relaxar a aresta (u,v) de peso w sabendo os valores de $\lambda(u)$ e $\lambda(v)$

Quem é $\lambda(u)$? É o custo atual de sair da origem s e chegar até u
Quem é $\lambda(v)$? É o custo atual de sair da origem s e chegar até v

Ideia geral: é uma boa passar por u para chegar em v sabendo que o custo de ir de u até v é w?



Obs: A operação relax(u, v, w) nunca aumenta o valor de $\lambda(v)$, apenas diminui

PSEUDOCODIGO

```
relax(u, v, w)
{
        if   λ(v)>λ(u)+w(u,v)
        {
             λ(v)=λ(u)+w(u,v)
             π(v)=u
        }
}
```

```
relax(u, v, w)
{
        λ(v)=min{λ(v),λ(u)+w(u,v)}
        if   λ(v)  was updated
             π(v)=u
}
```

O que varia nos diversos algoritmos para encontrar caminhos mínimos são os seguintes aspectos:
i) Quantas e quais arestas devemos relaxar? ii) Quantas vezes devemos relaxar as arestas?
iii) Em que ordem devemos relaxar as arestas?

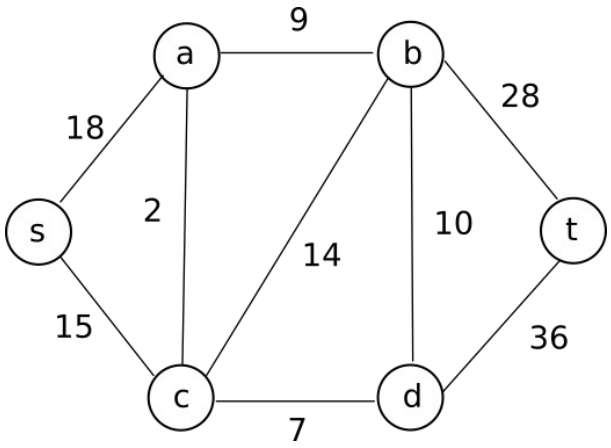
A seguir veremos um algoritmo eficiente para resolver o problema do caminho mínimo: o algoritmo de Dijkstra. Basicamente, esse algoritmo faz uso de uma política de gerenciamento de vértices baseada em aspectos de programação dinâmica. O que o método faz é basicamente criar uma fila de prioridades para organizar os vértices de modo que quanto menor o custo $\lambda(v)$ maior a prioridade do vértice em questão. Assim, a ideia é expandir primeiramente os menores ramos da árvore de caminhos mínimos, na expectativa de que os caminhos mínimos mais longos usarão como base os subcaminhos obtidos anteriormente. Trata-se de um mecanismo de reaproveitar soluções de subproblemas para a solução do problema como um todo.

Definição das variáveis
$\lambda(v)$: menor custo até o momento para o caminho s-v
$\pi(v)$: predecessor de v na árvore de caminhos mínimos
Q: fila de prioridades dos vértices (maior prioridade = menor $\lambda(v)$)

PSEUDOCODIGO

```
Dijkstra(G, w, s)
{
    for each   v∈V
    {
          λ(v)=∞
          π(v)=nil
    }
      λ(s)=0
      π(s)=nil
    Q = V     // fila de prioridades
    while   Q≠Ø
     {
        u = ExtractMin(Q)
        S=S∪{u}
        for each   v∈N(u)
            relax(u, v, w)
    }
}
```

Ex:



Fila de prioridades

	s	a	b	c	d	t
$\lambda^{(0)}(v)$	0	∞	∞	∞	∞	∞
$\lambda^{(1)}(v)$		18	∞	15	∞	∞
$\lambda^{(2)}(v)$		17	29		22	∞
$\lambda^{(3)}(v)$			27		22	∞
$\lambda^{(4)}(v)$			27			58
$\lambda^{(5)}(v)$						55

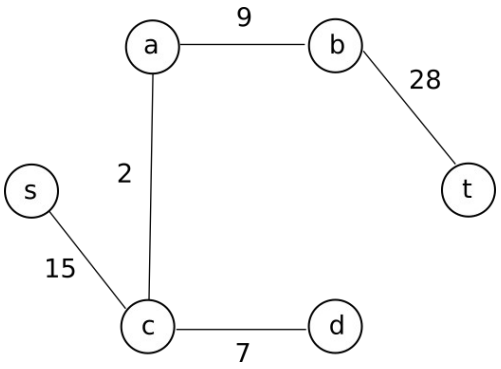
Ordem de acesso aos vértices

u	$V'=\{v \in N(u) \wedge v \in Q\}$	$\lambda(v), \forall v \in V'$	$\pi(v)$
s	{a, c}	$\lambda(a)=min\{\lambda(a),\lambda(s)+w(s,a)\}=min\{\infty,18\}=18$	$\pi(a)=s$
		$\lambda(c)=min\{\lambda(c),\lambda(s)+w(s,c)\}=min\{\infty,15\}=15$	$\pi(c)=s$
c	{a, b, d}	$\lambda(a)=min\{\lambda(a),\lambda(c)+w(c,a)\}=min\{18,17\}=17$	$\pi(a)=c$
		$\lambda(b)=min\{\lambda(b),\lambda(c)+w(c,b)\}=min\{\infty,29\}=29$	$\pi(b)=c$
		$\lambda(d)=min\{\lambda(d),\lambda(c)+w(c,d)\}=min\{\infty,22\}=22$	$\pi(d)=c$
a	{b}	$\lambda(b)=min\{\lambda(b),\lambda(a)+w(a,b)\}=min\{29,27\}=27$	$\pi(b)=a$
d	{b, t}	$\lambda(b)=min\{\lambda(b),\lambda(d)+w(d,b)\}=min\{27,32\}=27$	---
		$\lambda(t)=min\{\lambda(t),\lambda(d)+w(d,t)\}=min\{\infty,58\}=58$	$\pi(t)=d$
b	{t}	$\lambda(t)=min\{\lambda(t),\lambda(b)+w(b,t)\}=min\{58,55\}=55$	$\pi(t)=b$
t	$\emptyset$	---	---

Mapa de predecessores (árvore final)

v	s	a	b	c	d	t
$\pi(v)$	---	c	a	s	c	b

Árvore de caminhos mínimos (armazena os menores caminhos de s a todos os demais vértices)



Teorema: O algoritmo de Dijkstra termina com $\lambda(v)=d(s,v), \forall v \in V$

Obs: Note que sempre $\lambda(v) \geq d(s,v)$ (*)

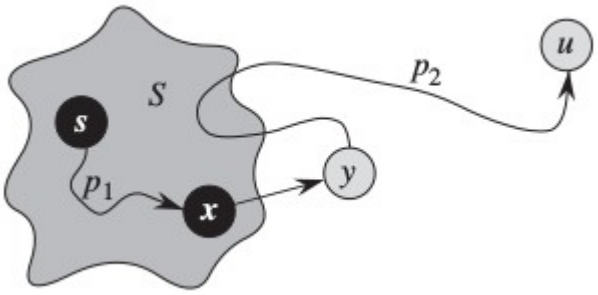
1. Suponha que u seja o 1º vértice para o qual $\lambda(u) \neq d(s,u)$ quando u entra em S.

2. Então, $u \neq s$ pois senão $\lambda(s)=d(s,s)=0$

3. Assim, existe um caminho P_{su} pois senão $\lambda(u)=d(s,u)=\infty$. Portanto, existe um caminho mínimo P^*_{su}

4. Antes de adicionar u a S, P^*_{su} possui $s \in S$ e $u \in V-S$

5. Seja y o 1º vértice em P^*_{su} tal que $y \in V-S$ e seja x seu predecessor ($x \in S$)



P^*_{su} = s → xy → u
p1 p2

Obs: Note que tanto p1 quanto p2 não precisam ter arestas

6. Como $x \in S$, $\lambda(x)=d(s,x)$ e no momento em que ele foi inserido a S, a aresta (x,y) foi relaxada, ou seja:

$$\lambda(y)=\lambda(x)+w(x,y)=d(s,x)+w(x,y)=d(s,y)$$

7. Mas y antecede a u no caminho e como $w: E \rightarrow R^+$ (pesos positivos), temos:

$$d(s,y) \leq d(s,u)$$

e portanto

$$\lambda(y) = d(s,y) \leq d(s,u) \leq \lambda(u)$$
(6) (7) (*)

8. Mas como ambos y e u pertencem a V – S, quando u é escolhido para entrar em S temos $\lambda(u) \leq \lambda(y)$

9. Como $\lambda(y) \leq \lambda(u)$ e $\lambda(u) \leq \lambda(y)$ então temos que $\lambda(u) = \lambda(y)$, o que implica em:

$$\lambda(y) = d(s,y) = d(s,u) = \lambda(u)$$

o que gera uma contradição. Portanto $\nexists u \in V$ tal que $\lambda(u) \neq d(s,u)$ quando u entra em S.

Encontrando as coordenadas da imersão

O próximo passo consiste em, dada a matriz de distâncias geodésicas, estimar as coordenadas dos pontos no R^d que satisfazem essas distâncias ponto a ponto. A solução para esse problema é dada pelo método MDS (Multidimensional Scaling).

Multidimensional Scaling (MDS)

Objetivo: Dada uma matriz de distâncias par a par, recuperar quem são as coordenadas dos pontos $\vec{x}_r \in R^k, r = 1,2,...,n$, onde K é definido pelo usuário (plano, espaço 3D, etc…)

A distância entre os vetores $\vec{x}_r$ e $\vec{x}_s$ é $d_{rs}^2 = \|\vec{x}_r - \vec{x}_s\|^2 = (\vec{x}_r - \vec{x}_s)^T(\vec{x}_r - \vec{x}_s)$

A matriz de distâncias é dada por $D = \{d_{rs}^2\}, r, s = 1,2,...,n$ (r é linha, s é coluna)

Seja B a matriz dos produtos internos

$$B = \{b_{rs}\}, \quad onde \quad b_{rs} = \vec{x}_r^T \vec{x}_s$$

O método MDS baseia-se na resolução de 2 subproblemas:
i) Encontrar a matriz B a partir de D
ii) recuperar as coordenadas dos pontos a partir de B

<u>Subproblema 1:</u> Encontrar B a partir de D

Hipótese: a média dos dados é nula (pontos estão ao redor do vetor nulo)

$$\sum_{r=1}^{n} \vec{x}_r = 0$$ (caso contrário há infinitas possibilidades, basta transladar os pontos)

De d_{rs}^2 a partir da distributiva, temos:

$$d_{rs}^2 = \vec{x}_r^T \vec{x}_r + \vec{x}_s^T \vec{x}_s - 2\vec{x}_r^T \vec{x}_s \quad (*)$$

Assim, a partir da matriz D, podemos obter a média de uma coluna s como:

$$\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_s^T \vec{x}_s - \frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_s = \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \vec{x}_s^T \vec{x}_s \quad (**)$$

Analogamente, podemos computar a média de uma linha r como:

$$\frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{s=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T \vec{x}_s - \frac{2}{n}\sum_{s=1}^{n} \vec{x}_r^T \vec{x}_s = \vec{x}_r^T \vec{x}_r + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T \vec{x}_s \quad (***)$$

E finalmente, podemos computar a média dos elementos de D como:

$$\frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2 = \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_s^T \vec{x}_s - \frac{2}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \vec{x}_r^T \vec{x}_s = \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T \vec{x}_s =$$

$$= \frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r \quad (****)$$

Note que de (*) é possível definir b_{rs} como:

$$b_{rs} = \vec{x}_r^T \vec{x}_s = -\frac{1}{2}\left(d_{rs}^2 - \vec{x}_r^T \vec{x}_r - \vec{x}_s^T \vec{x}_s\right)$$

Mas de (***) podemos isolar o termo $-\vec{x}_r^T \vec{x}_r$:

$$-\vec{x}_r^T \vec{x}_r = -\frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{1}{n}\sum_{s=1}^{n} \vec{x}_s^T \vec{x}_s \quad (1)$$

E de (**) podemos isolar o termo $-\vec{x}_s^T \vec{x}_s$:

$$-\vec{x}_s^T \vec{x}_s = -\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 + \frac{1}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r \quad (2)$$

Então, fazendo (1) – (2) temos:

$$-\vec{x}_r^T \vec{x}_r - \vec{x}_s^T \vec{x}_s = -\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r$$

De (****) podemos escrever:

$$\frac{2}{n}\sum_{r=1}^{n} \vec{x}_r^T \vec{x}_r = \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2$$

De modo que temos uma expressão completa para b_{rs} em função dos elementos de D:

$$b_{rs} = -\frac{1}{2}\left(d_{rs}^2 - \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2\right)$$

Chamando de $a_{rs} = -\frac{1}{2} d_{rs}$ podemos escrever:

$a_{r.} = \frac{1}{n}\sum_{s=1}^{n} a_{rs}$ (média na linha r)

$a_{.s} = \frac{1}{n}\sum_{r=1}^{n} a_{rs}$ (média na coluna s)

$a_{..} = \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} a_{rs}$ (média em D)

expressando b_{rs} como:

$b_{rs} = a_{rs} - a_{r.} - a_{.s} + a_{..}$ (@)

Definindo a matriz $A = \{a_{rs}\}, r, s = 1,2,\ldots,n$, como $A = -\frac{1}{2}D$ pode-se mostrar que a relação entre B e A é ainda mais simplificada, sendo dada por:

$B = H\,A\,H$ onde a matriz H é definida por

$H = I - \frac{1}{n}\vec{1}\vec{1}^T$ sendo $\vec{1}^T = [1,1,\ldots,1]$ (vetor de 1's com n dimensões)

Dessa forma tem-se que $\vec{1}\vec{1}^T = U = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & \ldots & 1 \end{bmatrix}$

Note que $B = H\,A\,H$ nada mais é que a forma matricial da equação (@) uma vez que

$$B = H\,A\,H = \left(I - \frac{1}{n}U\right)A\left(I - \frac{1}{n}U\right) = \left(A - \frac{1}{n}U\,A\right)\left(I - \frac{1}{n}U\right) = A - A\,\frac{1}{n}U - \frac{1}{n}U\,A + \frac{1}{n^2}U\,A\,U$$

Portanto, temos a matriz B.

<u>Subproblema 2:</u> Recuperar as coordenadas $\vec{x}_r \in R^p$ a partir de B

Note que a matriz B dos produtos internos pode ser expressa por:

$B_{n\times n} = X_{n\times p}\,X^T_{p\times n}$

onde n é denota número de amostras e p denota o número de dimensões.

A matriz B possui 3 propriedades importantes:
a) simétrica
b) O rank de B é p (número de linhas/colunas linearmentre independente: gera uma base em R^p)
c) positiva semi-definida: $\forall \vec{x} \in R^n, \vec{x}^T B\,\vec{x} \geq 0$

Isso implica em dizer que a matrix B possui p autovalores não negativos e n - p autovalores nulos. Assim, pela decomposição espectral de B (eigendecomposition) pode-se escrever:

$B = V\,\Lambda\,V^T$ onde

$\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$ é a matriz diagonal dos autovalores de B e

$$V = \begin{bmatrix} | & | & \dots & | \\ | & | & \dots & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \\ | & | & \dots & | \\ | & | & \dots & | \end{bmatrix}_{n \times n}$$ é a matriz dos autovetores de B

Sem perda de generalidade iremos considerar $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n$
Devido aos n – p autovalores nulos, B pode ser escrita como:

$B = V' \Lambda' V'^T$ onde

$\Lambda' = diag(\lambda_1, \lambda_2, \dots, \lambda_p)$ é a matriz diagonal dos autovalores de B e

$$V' = \begin{bmatrix} | & | & \dots & | \\ | & | & \dots & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_p \\ | & | & \dots & | \\ | & | & \dots & | \end{bmatrix}_{n \times p}$$

Mas como

$$B_{n \times n} = X_{n \times p} X^T_{p \times n} = V' \Lambda' V'^T = V' \Lambda'^{1/2} \Lambda'^{1/2} V'^T$$

temos finalmente que

$X_{n \times p} = V'_{n \times p} \Lambda'^{1/2}_{p \times p}$ onde $\Lambda'^{1/2} = diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_p})$

Cada linha de $X_{n \times p}$ terá a coordenada de um vetor $\vec{x}_i \in R^p$, onde p é uma parâmetro que controla o número de dimensões do espaço de saída: se desejamos um plot em 2D, p = 2, em caso de um plot 3D, k = 3, etc.

Algoritmo MDS

Entrada: $D = \{d^2_{rs}\}$ (obtida criando grafo e executando Dijkstra n vezes)

1. Faça $A = -\frac{1}{2} D$

2. Faça $H = I - \frac{1}{n} \vec{1} \vec{1}^T$

3. Compute $B = H A H$

4. Encontrar os autovalores e autovetores de B

5. Tomar os K autovetores associados aos K maiores autovalores de B e montar $V'_{n \times k}$ e $\Lambda' = diag(\lambda_1, \lambda_2, \dots, \lambda_k)$

6. Calcular $X_{n\times k} = V'_{n\times k}\,\Lambda'^{1/2}_{k\times k}$

Convém destacar que o algoritmo ISOMAP é totalmente não supervisionado, no sentido que ele não usa qualquer informação sobre a distribuição das classes. Ele tenta encontrar uma representação mais compacta dos dados originais apenas preservando as distâncias.

Limitação do ISOMAP: não convexidade dos dados (presença de buracos na variedade) pode ser um problema.

"Antes de pensar em desistir, lembre-se: a última parte de uma árvore a crescer são os frutos."
(Anônimo)