MS777 - Projeto supervisionado I

Geometria de Distâncias e Aplicações

Autor: Edgar Lavor
Orientador: Prof. Dr. Carlile Lavor

2015

1 Introdução

Este texto é baseado no artigo *"Euclidean Distance Matrices: A Short Walk Through Theory, Algorithms and Applications"* [1] e tem como objetivo explorar, com exemplos, alguns conceitos apresentados no artigo original e detalhar algumas passagens como forma de estudo e aprofundamento do conteúdo.

A geometria de distância é uma teoria que tem por objetivo determinar a localização de uma coleção de pontos com informações apenas sobre as distâncias entre estes pontos.

Este problema surge em uma série de aplicações, como, por exemplo, determinar a estrutura de uma molécula com informações sobre a distância entre os átomos que a compõe [2], ou então determinar a topologia de uma rede sem fio medindo-se a intensidade do sinal entre os pares de sensores, o que permite estimar as distâncias [3].

O uso de Matrizes de Distância Euclidiana (EDM, do inglês Euclidean Distance Matrices) pode ser empregado na resolução de problemas de geometria de distância, e é essencialmente sobre EDMs que trata este trabalho.

Uma EDM é uma matriz cujas entradas são as distâncias ao quadrado de uma coleção de pontos, ou, em outras palavras, o valor na linha i e coluna j de uma EDM representa a distância ao quadrado entre os pontos i e j.

Como veremos, se soubermos a distância exata entre todos os pares de pontos de uma coleção de pontos, determinar as posições é uma tarefa simples utilizando-se uma EDM, porém, na maioria dos problemas temos um ou mais dos seguintes complicadores:

- As distâncias não são exatas;
- Não sabemos algumas das distâncias;
- Não se sabe a que par de pontos refere-se cada distância.

Abaixo temos um problema fictício, que será utilizado para ilustrar os conceitos explorados neste trabalho:

"Um viajante muito experiente, depois de diversas viagens entre um grupo de cidades, foi capaz de determinar as distâncias entre cada par de cidades. Com essas informações o viajante gostaria de montar um mapa, porém, não sabe como fazer isso.

Vamos denominar as cidades por A, B, C, D e E. O problema consiste em ajudar o viajante a montar um mapa para este grupo de cidades utilizando as distâncias medidas."

Segue abaixo a matriz com as distâncias ao quadrado (uma EDM) entre os pares de cidades:

$$S = \begin{bmatrix} 0 & 17 & 5 & 2 & 29 \\ 17 & 0 & 26 & 13 & 90 \\ 5 & 26 & 0 & 13 & 32 \\ 2 & 13 & 13 & 0 & 37 \\ 29 & 90 & 32 & 37 & 0 \end{bmatrix}$$

2 EDMs e o teorema do Posto

O problema proposto na introdução consiste em, dada uma EDM, determinar o conjunto de pontos que gerou essa EDM. Este é o problema inverso de um problema muito mais simples, que é encontrar a EDM dado um grupo de pontos.

Abaixo vamos desenvolver uma expressão para uma EDM a partir de uma coleção de pontos, o que nos permitirá identificar uma propriedade muito importante relativa ao posto de uma EDM.

Considere uma matriz X do $\mathbb{R}^{d\times n}$, onde cada coluna de X representa um ponto no $\mathbb{R}^d$, $X = [x_1, x_2, \ldots, x_n]$. Esta matriz representa uma coleção de pontos do $\mathbb{R}^d$.

A distância ao quadrado entre os pontos x_i e x_j é dada por

$$d_{ij} = ||x_i - x_j||^2,$$

onde $||.||$ denota a norma Euclidiana. Expandindo a norma temos:

$$d_{ij} = (x_i - x_j)^T(x_i - x_j) = x_i^T x_i - 2x_i^T x_j + x_j^T x_j \tag{1}$$

A partir desta expressão para d_{ij}, podemos determinar uma equação matricial para a matriz $D = [d_{ij}]$:

$$D = edm(X) \stackrel{def}{=} diag(X^T X)\, \mathbf{1}^T - 2X^T X + \mathbf{1}\, diag(X^T X)^T, \tag{2}$$

onde $\mathbf{1}$ denota o vetor coluna com todas as entradas iguais a 1 e $diag(A)$ é o vetor coluna com as entradas iguais aos valores na diagonal de A.

A compreensão desta fórmula pode não ser imediata, e para ficar mais claro o que ela de fato representa, vamos mostrar o desenvolvimento para $d = 3$ e $n = 5$.

Primeiramente tomamos um X genérico de dimensões $d \times n$:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \end{bmatrix}$$

Agora vamos calcular os termos envolvidos no cálculo de $edm(X)$:

$$X^T X = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 & x_1^T x_4 & x_1^T x_5 \\ x_2^T x_1 & x_2^T x_2 & x_2^T x_3 & x_2^T x_4 & x_2^T x_5 \\ x_3^T x_1 & x_3^T x_2 & x_3^T x_3 & x_3^T x_4 & x_3^T x_5 \\ x_4^T x_1 & x_4^T x_2 & x_4^T x_3 & x_4^T x_4 & x_4^T x_5 \\ x_5^T x_1 & x_5^T x_2 & x_5^T x_3 & x_5^T x_4 & x_5^T x_5 \end{bmatrix},$$

$$diag(X^T X) = \begin{bmatrix} x_1^T x_1 \\ x_2^T x_2 \\ x_3^T x_3 \\ x_4^T x_4 \\ x_5^T x_5 \end{bmatrix}, \quad \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$diag(X^T X)\, \mathbf{1}^T = \begin{bmatrix} x_1^T x_1 & x_1^T x_1 & x_1^T x_1 & x_1^T x_1 & x_1^T x_1 \\ x_2^T x_2 & x_2^T x_2 & x_2^T x_2 & x_2^T x_2 & x_2^T x_2 \\ x_3^T x_3 & x_3^T x_3 & x_3^T x_3 & x_3^T x_3 & x_3^T x_3 \\ x_4^T x_4 & x_4^T x_4 & x_4^T x_4 & x_4^T x_4 & x_4^T x_4 \\ x_5^T x_5 & x_5^T x_5 & x_5^T x_5 & x_5^T x_5 & x_5^T x_5 \end{bmatrix},$$

$$\mathbf{1}\, diag(X^T X)^T = \begin{bmatrix} x_1^T x_1 & x_2^T x_2 & x_3^T x_3 & x_4^T x_4 & x_5^T x_5 \\ x_1^T x_1 & x_2^T x_2 & x_3^T x_3 & x_4^T x_4 & x_5^T x_5 \\ x_1^T x_1 & x_2^T x_2 & x_3^T x_3 & x_4^T x_4 & x_5^T x_5 \\ x_1^T x_1 & x_2^T x_2 & x_3^T x_3 & x_4^T x_4 & x_5^T x_5 \\ x_1^T x_1 & x_2^T x_2 & x_3^T x_3 & x_4^T x_4 & x_5^T x_5 \end{bmatrix}$$

Dessa forma, fica simples perceber que o elemento na linha i e coluna j de $edm(X)$, dado pela fórmula (2), é exatamente a distância ao quadrado entre os pontos x_i e x_j. Abaixo mostramos de qual matriz veio cada termo:

$$d_{ij} = \underbrace{x_i^T x_i}_{diag(X^T X)\, \mathbf{1}^T} \underbrace{- 2 x_i^T x_j}_{2X^T X} + \underbrace{x_j^T x_j}_{\mathbf{1}\, diag(X^T X)^T}$$

Agora com a fórmula (2) podemos chegar a uma propriedade para EDMs que é exposta no teorema a seguir:

Teorema 2.1 (Posto de EDMs)**.** *O posto de uma EDM correspondente a pontos do $\mathbb{R}^d$ é no máximo $d + 2$.*

Demonstração. Como a matriz X tem d linhas, então terá um posto de no máximo d.

Vamos supor, sem perda de generalidade, que as g primeiras colunas de X sejam LI, e como o posto de X é no máximo d, então $g \leq d$.
A i-ésima linha da matriz $X^T X$ pode ser escrita como:

$$x_i^T [x_1 \; x_2 \; \ldots \; x_n]$$

Se $i > g$, então x_i é uma combinação linear de x_1, x_2,..., x_g e, portanto, a i-ésima linha de $X^T X$ será uma combinação linear das g primeiras linhas. Isso nos permite concluir que $X^T X$ terá no máximo g linhas LI e, assim, terá posto no máximo g.
Como $g \leq d$, então $X^T X$ também tem posto no máximo d.
Além disso, a matriz $diag(X^T X)\ \mathbf{1}^T$ possui todas as colunas iguais e a matriz $\mathbf{1}\ diag(X^T X)^T$ possui todas as linhas iguais, assim, cada uma delas possui posto 1.
Finalmente, como o posto da soma de matrizes não pode exceder a soma do posto das parcelas, o posto de uma EDM, baseado na fórmula (2), é no máximo $d + 2$. □

Este teorema nos diz que o posto de uma EDM independe do número de pontos que a gera. Em muitas aplicações d é menor ou igual a três, enquanto n pode estar na casa dos milhares.

3 A quantidade de soluções

Conforme já citado, o problema de se determinar a posição de uma coleção de pontos a partir das distâncias entre estes pontos é um problema inverso, e ao se trabalhar com um problema inverso devemos ter em mente aquilo que podemos e aquilo que não podemos recuperar. Para representar uma coleção de pontos através das distâncias geralmente precisamos de mais informação.

A quantidade de distâncias (números) para se representar n pontos do $\mathbb{R}^d$ é $\binom{n}{2}$, enquanto que para representar estes mesmos pontos com coordenadas usamos dn números. Para as aplicações mais interessantes teremos $\binom{n}{2} > dn$, ou seja, uma EDM possui mais escalares que os pontos listados por coordenadas.

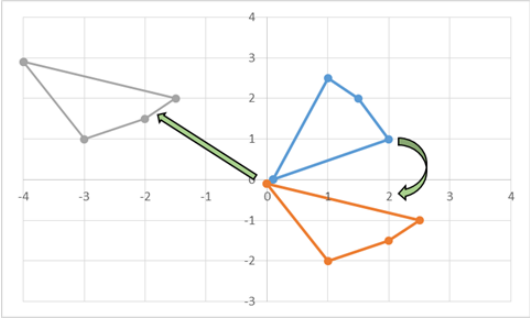
Apesar disso, nesta conversão perdemos a informação da posição absoluta e da orientação do grupo de pontos. Intuitivamente é possível perceber que se eu translado ou rotaciono uma coleção de pontos, as distâncias entre os pontos permanecem as mesmas. Vamos mostrar que isso de fato ocorre através da fórmula (2).

Notemos que $edm(X)$ é uma função de $X^T X$. Além disso, toda rotação no $\mathbb{R}^d$ pode ser realizada por uma matriz $Q \in \mathbb{R}^{d \times d}$, onde Q é ortogonal (uma matriz A é ortogonal se $A^T A = I$, onde I é a matriz identidade). Sendo assim, se X é o meu conjunto de pontos, $X_r = QX$ é o meu conjunto de pontos rotacionados.

Vamos mostrar que $X_r^T X_r = X^T X$ e que, portanto, $edm(X) = edm(X_r)$:

$$X_r^T X_r = (QX)^T (QX) = X^T Q^T Q X = X^T I X = X^T X$$

Figura 1: Exemplo de rotação e translação. A translação leva cada ponto x ao ponto $x + b$, onde $b = [-4\ 3]^T$, enquanto que a rotação leva o ponto $x = [a\ b]^T$ ao ponto $x = [b\ -a]^T$, o que corresponde a uma rotação de 90°.



O caso da translação é um pouco mais complicado. A translação de um conjunto X de pontos por um vetor $b \in \mathbb{R}^d$ é dada por:

$$X_t = X + b\,\mathbf{1}^T$$

Neste caso não temos $X_t^T X_t = X^T X$, porém, através da fórmula (2) podemos concluir que $edm(X_t) = edm(X)$:

$$X_t^T X_t = (X^T + b^T\,\mathbf{1})(X + b\,\mathbf{1}^T) = X^T X + \mathbf{1}b^T X + X^T b\,\mathbf{1}^T + \mathbf{1}\,b^T b\mathbf{1}^T$$

Assim, o elemento x_{tij} na linha i e coluna j de $X_t^T X_t$ é dado por:

$$x_{tij} = x_i^T x_j + b^T x_i + b^T x_j + b^T b$$

Sendo assim, os elementos y_{ij} e w_{ij} das matrizes $diag(X_t^T X_t)\,\mathbf{1}^T$ e $\mathbf{1}^T\,diag(X_t^T X_t)$, respectivamente, são dados por:

$$y_{ij} = x_i^T x_i + 2b^T x_i + b^T b$$

$$w_{ij} = x_j^T x_j + 2b^T x_j + b^T b$$

Como $edm(X_t) = diag(X_t^T X_t)\,\mathbf{1}^T - 2X_t^T X_t + \mathbf{1}^T\,diag(X_t^T X_t)$, então o elemento z_{ij} da linha i e coluna j de $edm(X_t)$ é dado por:

$$z_{ij} = y_{ij} - 2x_{tij} + w_{ij}$$

$$z_{ij} = x_i^T x_i - 2x_i^T x_j + x_j^T x_j$$

Ou seja, z_{ij} tem o mesmo valor que d_{ij} na fórmula (1), que é o elemento da linha i e coluna j de $edm(X)$. Finalmente concluímos que $edm(X_t) = edm(X)$. Em resumo,

$$edm(QX) = edm(X + b\,\mathbf{1}^T) = edm(X) \tag{3}$$

A consequência dessa invariância é que nunca conseguiremos reconstruir a posição absoluta de um conjunto de pontos usando apenas as distâncias. Diferentes processos de reconstrução podem levar a localizações diferentes, e todas elas sendo transformações rígidas umas das outras.

4 Reconstruindo o conjunto de pontos a partir das distâncias

A equação (2) é capaz de nos levar a um processo para computar a posição de um conjunto de pontos a partir da matriz de distâncias. Considere a seguinte escolha: o primeiro ponto x_1 do nosso conjunto está na origem. Então a primeira coluna de $D = edm(X)$ contém a norma ao quadrado dos vetores representados pelo conjunto de pontos,

$$d_{i1} = ||x_i - x_1||^2 = ||x_i - 0||^2 = ||x_i||^2.$$

Com isso podemos calcular o termo $\mathbf{1}^T\ diag(X^TX)$ e sua transposta em (2), uma vez que a diagonal de X^TX contém exatamente as normas ao quadrado $||x_i||^2$. Assim,

$$\mathbf{1}\ diag(X^TX)^1 = \mathbf{1}\ d_1^T,$$

onde $d_1 = De_1$ é a primeira coluna de D. Isolando o termo X^TX na equação (2) obtemos a seguinte expressão,

$$G = X^TX = -\frac{1}{2}(D - \mathbf{1}\ d_1^T - d_1\ \mathbf{1}^T).$$

Pelo desenvolvimento na seção (2) é possível verificar que X^TX é uma matriz simétrica, além disso, dado $x \in \mathbb{R}^n$, com $x \neq 0$, $x^T(X^TX)x = (Xx)^T(Xx) = ||Xx|| \geq 0$, portanto, $G = X^TX$ é semidefinida positiva. Sendo assim, podemos escrever $G = U\Lambda U^T$, onde $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, sendo os $\lambda_i's$ auto-valores de G e U a matriz dos auto-vetores de G.

O processo de se escrever G desta forma é chamado de diagonalização e por G ser simétrica e semidefinida positiva, é possível garantir que os $\lambda_i's$ são todos não negativos e que U é ortonormal.

A partir daqui vamos assumir que os auto-valores estão ordenados em ordem decrescente, ou seja, $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. Agora podemos fazer a seguinte definição

$$\hat{X} \stackrel{def}{=} [diag(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}, 0_{d\times(n-d)}]U^T. \tag{4}$$

Poderíamos ter simplesmente tomado $\Lambda^{1/2}U^T$ como a reconstrução do nosso conjunto de pontos, porém, se X realmente descreve um conjunto de pontos d-dimensional, os auto-valores desconsiderados devem ser zeros.

Pela construção, o conjunto de pontos $\hat{X}$ deve gerar a EDM original, $D = edm(X)$.

Para ilustrar este processo, vamos aplicá-lo à matriz S do problema proposto na introdução,

$$S = \begin{bmatrix} 0 & 17 & 5 & 2 & 29 \\ 17 & 0 & 26 & 13 & 90 \\ 5 & 26 & 0 & 13 & 32 \\ 2 & 13 & 13 & 0 & 37 \\ 29 & 90 & 32 & 37 & 0 \end{bmatrix}.$$

Primeiramente vamos determinar a matriz G,

$$G = -\frac{1}{2}(S - \mathbf{1}\, s_1^T - s_1\, \mathbf{1}^T),$$

onde s_1 é a primeira coluna de S.

Fazendo os cálculos chegamos a

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 17 & -2 & 3 & -22 \\ 0 & -2 & 5 & -3 & 1 \\ 0 & 3 & -3 & 2 & -3 \\ 0 & -22 & 1 & -3 & 29 \end{bmatrix}.$$

O próximo passo é encontrar as matrizes Λ e U. No MatLab, o comando $eig(A)$ retorna as matrizes Λ e U da diagonalização de A, e utilizando este comando obtemos o seguinte,

$$\Lambda = \begin{bmatrix} 46.3305 & 0 & 0 & 0 & 0 \\ 0 & 6.6695 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -0.6050 & 0.0782 & 0.7663 & 0 & -0.2015 \\ 0.0555 & -0.8534 & 0.2503 & 0 & 0.4538 \\ -0.0980 & 0.4828 & 0.1006 & 0 & 0.8644 \\ 0.7882 & 0.1802 & 0.5831 & 0 & -0.0791 \end{bmatrix}.$$

No MatLab o comando eig não retorna os auto-valores em ordem decresente, portanto, é necessário ajustar as colunas de Λ, e ao trocar duas colunas em Λ deve-se trocar também as colunas correspondentes em U.

Veja que nesse caso $\Lambda^{1/2}$ já está na forma usada na equação (4), e então temos:

$$\hat{X} = \Lambda^{1/2} U^T = \begin{bmatrix} 0 & -4.1182 & 0.3775 & -0.6673 & 5.3650 \\ 0 & 0.2020 & -2.2040 & 1.2469 & 0.4653 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Como $d = 2$, ficamos somente com as duas primeiras linhas,

$$\tilde{X} = \begin{bmatrix} 0 & -4.1182 & 0.3775 & -0.6673 & 5.3650 \\ 0 & 0.2020 & -2.2040 & 1.2469 & 0.4653 \end{bmatrix},$$

e esse seria o nosso conjunto de pontos.

Podemos verificar que,

$$edm(\tilde{X}) = \begin{bmatrix} 0 & 17.0004 & 5.0001 & 2.0000 & 28.9997 \\ 17.0004 & 0 & 26.0002 & 13.0005 & 90.0004 \\ 5.0001 & 26.0002 & 0 & 13.0003 & 32.0003 \\ 2.0000 & 13.0005 & 13.0003 & 0 & 36.9995 \\ 28.9997 & 90.0004 & 32.0003 & 36.9995 & 0 \end{bmatrix}.$$

Há uma pequena diferença entre $edm(\tilde{X})$ e S, e embora isto não seja desejável, era esperado, pois ao se fazer as contas no computador trabalhamos com erros de arredondamento de ponto flutuante. Neste caso o efeito é desprezível e não vai afetar os nossos resultados.

Vamos agora a uma outra questão. O conjunto de pontos utilizado originalmente para gerar S foi o seguinte,

$$X = \begin{bmatrix} 1 & 2 & 3 & 0 & -1 \\ 3 & 7 & 2 & 4 & -2 \end{bmatrix}.$$

Vemos então que $\tilde{X} \neq X$, mas, conforme vimos na seção anterior, eles devem estar relacionados por uma transformação rígida. A próxima seção trata de como encontrar tal transformação .

5 Ancorando o grupo de pontos

Como a posição absoluta de um conjunto de pontos é perdida quando tomamos esses pontos por suas distâncias, é necessário um método para alinhar o conjunto de pontos reconstruído com um conjunto de *âncoras* - pontos cujas coordenadas são conhecidas.

Isto pode ser realizado da seguinte forma. Seja Y a matriz cujas colunas são os pontos *âncoras*, e suponha que queremos alinhar os pontos de X com os pontos em Y. Tomemos X_a como uma submatriz (uma seleção de colunas) de X que deve ser alinhada com Y.

O primeiro passo é remover as médias y_c e $x_{a,c}$, de Y e X_a, respectivamente, obtendo as matrizes $\overline{Y}$ e $\overline{X}_a$. Este passo elimina a diferença de translação entre os dois conjuntos de pontos. Ao subtrair a média do conjunto de pontos levamos o ponto médio do conjunto para a origem, e isso estamos fazendo para os dois conjuntos. Assim, poderemos encontrar a rotação e reflexão que levará $\overline{X}_a$ em $\overline{Y}$ a partir da origem.

No próximo passo, chamado de análise de Procrutes, nós procuramos a rotação e reflexão que melhor mapeia $\overline{X}_a$ em $\overline{Y}$,

$$R = \underset{Q:QQ^T=I}{arg\ min}\ ||Q\overline{X}_a - \overline{Y}||_F^2. \tag{5}$$

A norma de Frobenius é definida da seguinte forma, $||A||_F^2 \stackrel{def}{=} \sum a_{ij}^2 = trace(A^T A)$.

A solução de (5) - dada por Schönemann em sua tese de PhD [4] - é dada pela decomposição em valores singulares (SVD, do inglês) [5]. Seja $U\Sigma V^T$ a decomposição SVD de $\overline{X}_a\overline{Y}$, ou seja, $\overline{X}_a\overline{Y} = U\Sigma V^T$, onde as matrizes U e V são ortogonais.

Desenvolvendo (5) obtemos o seguinte,

$$
\begin{aligned}
R &= \underset{Q:QQ^T=I}{arg\,min}\ ||Q\overline{X}_a - \overline{Y}||_F^2 && (6)\\
&= \underset{Q:QQ^T=I}{arg\,min}\ trace((Q\overline{X}_a - \overline{Y})^T(Q\overline{X}_a - \overline{Y})) && (7)\\
&= \underset{Q:QQ^T=I}{arg\,min}\ trace(\overline{X}_a^T Q^T Q\overline{X}_a) + trace(\overline{Y}^T\overline{Y}) && (8)\\
&\qquad\qquad -trace(\overline{X}_a^T Q^T\overline{Y}) - trace(\overline{Y}^T Q\overline{X}_a)\\
&= \underset{Q:QQ^T=I}{arg\,min}\ ||\overline{X}_a||_F^2 + ||\overline{Y}||_F^2 - 2\times trace(Q\overline{X}_a\overline{Y}^T) && (9)\\
&= \underset{Q:QQ^T=I}{arg\,max}\ trace(QU\Sigma V^T) && (10)\\
&= \underset{Q:QQ^T=I}{arg\,max}\ trace(V^TQU\Sigma) && (11)
\end{aligned}
$$

Nas passagens acima usamos o fato de que $trace(A) = trace(A^T)$ e que $trace(ABC) = trace(BCA)$, propriedade conhecida como invariância cíclica do traço. Além disso, na passagem da equação (9) para a (10) removemos os termos que não dependiam do argumento Q e, como estávamos minimizando um termo negativo, passamos a maximizar este mesmo termo com o sinal positivo.

Neste ponto é importante notar que $\Sigma = diag(\sigma_1, \ldots \sigma_k)$, com $\sigma_i's \geq 0$. Sendo assim, tomando $\overline{Q} = V^TQU$ temos que,

$$trace(\overline{Q}\Sigma) = \sum \bar{q}_{ii}\sigma_i.$$

Como $\overline{Q}$ é produto de matrizes ortogonais, então é uma matriz ortogonal também, e por isso $\bar{q}_{ii} \leq 1$. Isso nos permite concluir que o máximo de $trace(\overline{Q}\Sigma)$ ocorrerá quando $\bar{q}_{ii} = 1$, ou seja, $\overline{Q} = I$.

Então, $V^TQ^*U = I$, onde Q^* é o argumento que minimiza a direita da equação (6), e temos, finalmente, $R = Q^* = VU^T$.

Uma vez obtida a transformação rígida ótima, o alinhamento pode ser aplicado a todo o conjunto de pontos pela expressão,

$$R(X - x_{a,c}\,\mathbf{1}^T) + y_c\,\mathbf{1}^T.$$

O que esta última expressão faz é levar o conjunto de pontos para a posição que foi tomada como base para calcular a transformação R, aplicar a transformação R e depois fazer a translação para se adequar aos pontos âncoras.

Vamos agora aplicar o método exposto acima no nosso problema. Na seção anterior havíamos encontrado o seguinte conjunto de pontos,

$$\tilde{X} = \begin{bmatrix} 0 & -4.1182 & 0.3775 & -0.6673 & 5.3650 \\ 0 & 0.2020 & -2.2040 & 1.2469 & 0.4653 \end{bmatrix}.$$

E o conjunto original é dado por,

$$X_o = \begin{bmatrix} 1 & 2 & 3 & 0 & -1 \\ 3 & 7 & 2 & 4 & -2 \end{bmatrix}.$$

Consideremos $Y = \begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}$ o nosso conjunto de âncoras e $X_a = \begin{bmatrix} 0 & -4.1182 \\ 0 & 0.2020 \end{bmatrix}$. Os pontos médios serão $y_c = \begin{bmatrix} 1.5 \\ 5.0 \end{bmatrix}$ e $x_{a,c} = \begin{bmatrix} -2.0591 \\ 0.1010 \end{bmatrix}$ e assim,

$$\overline{Y} = \begin{bmatrix} -0.5 & 0.5 \\ -2.0 & 2.0 \end{bmatrix} \quad e \quad \overline{X}_a = \begin{bmatrix} 2.0591 & -2.0591 \\ -0.1010 & 0.1010 \end{bmatrix}.$$

O Matlab possui o comando *svd*, que retorna a decomposição em valores singulares de uma matriz. Utilizando este comando obtemos $\overline{X}_a \overline{Y}^T = U\Sigma V^T$, com

$$U = \begin{bmatrix} -0.9988 & 0.0490 \\ 0.0490 & 0.9988 \end{bmatrix},$$

$$S = \begin{bmatrix} 8.5001 & 0 \\ 0 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 0.2425 & -0.9701 \\ 0.9701 & 0.2425 \end{bmatrix}.$$

Portanto, $R = VU^T = \begin{bmatrix} -0.2898 & -9571 \\ -0.9571 & 0.2898 \end{bmatrix}$.

Podemos agora, a partir de $\tilde{X}$, voltar ao nosso conjunto original X_o,

$$R(\tilde{X} - x_{a,c}\,\mathbf{1}^T) + y_c\,\mathbf{1}^T = \begin{bmatrix} 1 & 2 & 3 & 0 & -1 \\ 3 & 7 & 2 & 4 & -2 \end{bmatrix}.$$

E com isto finalizamos esta seção.

6 Contando os graus de liberdade

Nesta seção vamos explorar os diferentes objetos relacionados a uma EDM e contar os graus de liberdade de cada um deles. Por grau de liberdade entendemos os valores independentes que representam o objeto, por exemplo, se temos n pontos do $\mathbb{R}^d$, então teremos,

$$\#x = n \times d$$

graus de liberdade, uma vez que não há nenhuma relação de dependência entre cada escalar que representa os n pontos, este número corresponde à quantidade de escalares envolvidos na descrição dos pontos.

Vamos agora imaginar que não conhecemos nenhuma propriedade específica de uma EDM para uma determinada matriz, exceto que esta matriz é simétrica, positiva, zero-diagonal e que possui posto no máximo $d + 2$. Vamos tentar encontrar os graus de liberdade associados a tal matriz. Podemos fazer a contagem olhando para a decomposição em auto-valores de uma matriz simétrica,

$D = U\Lambda U^T$. A matriz diagonal Λ é especificada por $d+2$ graus de liberdade, uma vez que D possui posto $d+2$. O primeiro auto-vetor de tamanho n nos dá $n-1$ graus de liberdade, devido à normalização; o segundo nos dá $n-2$ graus de liberdade, pois deve ser ortogonal ao primeiro; para o último auto-vetor teremos $n-(d+2)$ graus de liberdade. Os outros auto-vetores não precisam ser contados pois correspondem a auto-valores nulos. Asssim, o total de graus de liberdade é dado por,

$$\begin{aligned} \#DOF &= \underbrace{(d+2)}_{Auto-valores} + \underbrace{(n-1)+\ldots+[n-(d+2)]}_{Auto-vetores} - \underbrace{n}_{Zero-diagonal} \\ &= n\times(d+1) - \frac{(d+1)\times(d+2)}{2}. \end{aligned}$$

Para n grande e d fixo temos,

$$\frac{\#DOF}{\#x} \sim \frac{d+1}{d}. \tag{12}$$

Apesar de a propriedade do posto ser útil e levar a algoritmos eficientes, há uma questão a se considerar. Para $d=3$ a razão (12) é $\frac{4}{3}$, e assim a propriedade do posto nos dá 30% a mais de escalares a se determinar, para os quais devemos estabelecer valores consistentes. Colocando de outra maneira, precisaríamos de 30% a mais de dados para explorar a propriedade do posto do que precisaríamos para explorar a estrutura completa da EDM.

A redundância na representação de uma EDM é o que torna possível os algoritmos para completar EDMs e para trabalhar com dados imprecisos.

7 EDM incompleta e com ruído, explorando a propriedade do posto

Vamos ver agora um algoritmo que tem por objetivo completar uma EDM e remover ruidos. Considere então uma EDM D que possui algumas entradas faltando e outras com ruidos.

Seja W a matriz cuja entrada w_{ij} é 1 se a distância entre os pontos i e j são conhecidas e 0 caso a distância não seja conhecida ou possui algum ruído.

O seguinte algoritmo escrito no Matlab explora a propriedade do posto e vai, alternadamente, forçando o posto em $d+2$ e as entradas conhecidas:

```
function EDM_completion( W, Dc, n, d )
    %W is the mask of know distances without errors
    %Dc is the matrix with the distances
    %   (including the ones with error)
    %n is the matrix dimension
    %d is the dimension from EDM points

    max_it = 10000;
```

```
eps = 0.000001;
D = zeros(n,n);

%Initialize the D entries
for i = 1:n
    for j = 1:n
        D(i,j) = Dc(i,j);
    end
end

D_aux = zeros(n,n);
count = 0;
while (count<max_it) &&  (norm(D-D_aux,'fro')>eps)
    D_aux = D;

    %Make the EVD decomposition
    [U,V] =eig(D);
    v_auto_val = zeros(n,2);
    for i=1:n
        v_auto_val(i,1) = i;
    end
    v_auto_val(:,2) = diag(V);

    %Sort eigenvalues in absolute value ascending
    %order
    for i=1:n
        for j=i:n
            if (abs(v_auto_val(i,2))>
                    abs(v_auto_val(j,2)))
               aux = v_auto_val(i,:);
               v_auto_val(i,:) = v_auto_val(j,:);
               v_auto_val(j,:) = aux;
            end
        end
    end

    %Zero the smallers eigenvalues
    %(leting only d+2 non-zero)
    for i=1:(n-d-2)
       V(v_auto_val(i,1),v_auto_val(i,1)) = 0;
    end

    D = U*V*transpose(U);

    %Enforce know entries
    for i = 1:n
```

```
            for j = 1:n
                if W(i,j) == 1
                    D(i,j) = Dc(i,j);
                end
            end
        end

        %Zero negative entries
        for i = 1:n
            for j = 1:n
                if D(i,j)<0
                    D(i,j)=0;
                end
            end
        end

        count = count + 1;
    end

    disp(count);
    disp(D);
end
```

Este algoritmo deve então receber uma EDM com problemas na entrada e retornar uma EDM com as informações corretas.

Vamos ver alguns resultados deste algoritmo. Considere a seguinte EDM,

$$G = \begin{bmatrix} 0 & 17 & 5 & 2 & 29 & 5.54 & 65 & 53 \\ 17 & 0 & 26 & 13 & 90 & 41.94 & 146 & 72 \\ 5 & 26 & 0 & 13 & 32 & 7.94 & 72 & 26 \\ 2 & 13 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29 & 90 & 32 & 37 & 0 & 9.54 & 8 & 90 \\ 5.54 & 41.94 & 7.94 & 11.14 & 9.54 & 0 & 34.34 & 56.34 \\ 65 & 146 & 72 & 73 & 8 & 34.34 & 0 & 146 \\ 53 & 72 & 26 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix},$$

referente ao seguinte conjunto de pontos,

$$X = \begin{bmatrix} 1 & 7 & 3 & 0 & -1 & 0.5 & -3 & 8 \\ 3 & 2 & 2 & 4 & -2 & 0.7 & -4 & 1 \end{bmatrix}.$$

Veja que este grupo de pontos corresponde ao nosso conjunto de pontos do problema proposto na introdução com a adição de mais 3 pontos.

Vamos inserir algumas perturbações, primeiramente vamos zerar as posições (1, 2) e (2, 1), correspondentes à distância 17. Nesse caso o algoritmo converge para G em 82 iterações.

Vamos agora tomar as seguintes mudanças em G, zerando 3 distâncias:

$$H = \begin{bmatrix} 0 & 17 & 5 & 0 & 29 & 5.54 & 65 & 53 \\ 17 & 0 & 26 & 13 & 90 & 41.94 & 0 & 72 \\ 5 & 26 & 0 & 13 & 32 & 7.94 & 72 & 26 \\ 0 & 13 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29 & 90 & 32 & 37 & 0 & 0 & 8 & 90 \\ 5.54 & 41.94 & 7.94 & 11.14 & 0 & 0 & 34.34 & 56.34 \\ 65 & 0 & 72 & 73 & 8 & 34.34 & 0 & 146 \\ 53 & 72 & 26 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix}.$$

Neste caso a convergência ocorre para G em 399 iterações.

Agora ao invés de zerar as entradas, vamos inserir perturbações e ver o que acontece,

$$I = \begin{bmatrix} 0 & 17 & 5 & 2.4 & 29 & 5.54 & 65 & 53 \\ 17 & 0 & 26 & 13 & 90 & 41.94 & 143 & 72 \\ 5 & 26 & 0 & 13 & 32 & 7.94 & 72 & 26 \\ 2.4 & 13 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29 & 90 & 32 & 37 & 0 & 9 & 8 & 90 \\ 5.54 & 41.94 & 7.94 & 11.14 & 9 & 0 & 34.34 & 56.34 \\ 65 & 143 & 72 & 73 & 8 & 34.34 & 0 & 146 \\ 53 & 72 & 26 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix},$$

e a convergência para G se deu em 321 iterações.

Vamos ser mais drásticos e remover uma quantidade considerável de distâncias,

$$J = \begin{bmatrix} 0 & 17 & 5 & 0 & 29 & 5.54 & 65 & 53 \\ 17 & 0 & 26 & 13 & 90 & 41.94 & 0 & 72 \\ 5 & 26 & 0 & 0 & 32 & 0 & 72 & 0 \\ 0 & 0 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29 & 90 & 32 & 37 & 0 & 0 & 8 & 90 \\ 5.54 & 41.94 & 0 & 11.14 & 0 & 0 & 0 & 56.34 \\ 65 & 0 & 72 & 73 & 8 & 0 & 0 & 146 \\ 53 & 72 & 0 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix},$$

aqui obtemos a convergência para G em 2526 iterações.

Vamos olhar agora para um caso onde a convergência não ocorre para G, inserindo uma pequena perturbação nas distâncias que envolvem a primeira cidade,

$$K = \begin{bmatrix} 0 & 17.1 & 5.1 & 2.1 & 29.1 & 5.64 & 65.1 & 53.1 \\ 17.1 & 0 & 26 & 13 & 90 & 41.94 & 146 & 72 \\ 5.1 & 26 & 0 & 13 & 32 & 7.94 & 72 & 26 \\ 2.1 & 13 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29.1 & 90 & 32 & 37 & 0 & 9.54 & 8 & 90 \\ 5.64 & 41.94 & 7.94 & 11.14 & 9.54 & 0 & 34.34 & 56.34 \\ 65.1 & 146 & 72 & 73 & 8 & 34.34 & 0 & 146 \\ 53.1 & 72 & 26 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix},$$

aqui a convergência ocorre em 27 iterações, porém, para a seguinte matriz,

$$K' = \begin{bmatrix} 0 & 17.06 & 5.01 & 2.01 & 29.05 & 5.55 & 65.13 & 53.12 \\ 17.06 & 0 & 26 & 13 & 90 & 41.94 & 146 & 72 \\ 5.01 & 26 & 0 & 13 & 32 & 7.94 & 72 & 26 \\ 2.01 & 13 & 13 & 0 & 37 & 11.14 & 73 & 73 \\ 29.05 & 90 & 32 & 37 & 0 & 9.54 & 8 & 90 \\ 5.55 & 41.94 & 7.94 & 11.14 & 9.54 & 0 & 34.34 & 56.34 \\ 65.13 & 146 & 72 & 73 & 8 & 34.34 & 0 & 146 \\ 53.12 & 72 & 26 & 73 & 90 & 56.34 & 146 & 0 \end{bmatrix},$$

que não é a EDM correta. Como estamos utilizando somente a propriedade do posto neste algoritmo, estamos sujeitos a esse tipo de problema. Veja no entanto que esta matriz, K, possui uma diferença em relação às outras que testamos, nas outras os problemas estão dispersos pela matriz, enquanto que nesta os problemas estão todos concentrados em uma única cidade.

Para finalizar é importante ressaltar a importância deste algoritmo, uma vez que na maioria das aplicações encontraremos distâncias não exatas, ou então não teremos como determinar certas distâncias devido a uma série de motivos.

8 Considerações finais - Projeto supervisionado

Este trabalho é o resultado de um primeiro contato com a geometria de distâncias e EDMs. Primeiramente realizamos a caracterização de uma EDM e a sua construção algébrica, depois discutimos a unicidade do conjunto de pontos que gera a EDM, que é uma questão muito pertinente pelo fato de estarmos trabalhando com um problema inverso.

A partir daí, começamos de fato a resolver um problema, primeiramente com distâncias exatas e com uma EDM completa, e depois com pequenas perturbações e ausência de distâncias.

Evidentemente esta é uma incursão inicial, e há muito mais a se explorar. Como já dito, a maioria dos problemas reais não possuem dados exatos e o desenvolvimento da teoria segue exatamente no sentido de se trabalhar este tipo de problema.

No problema da topologia de uma rede, por exemplo, pode-se ter interferências nos sinais, o que acarreta ruídos na medida da distância, ou então podemos ter dois dispositivos que estão muito longe entre si, de forma que um não capta o sinal do outro [3]. Aqui temos o problema de EDM incompleta e medidas com ruído.

Um outro problema muito interessante é o da reconstrução da geometria de um quarto através da medida de ecos. Neste problema temos um quarto com a forma de um poliedro convexo no qual posicionamos microfones em diferentes pontos. Ao se emitir um som no quarto e medir o tempo de resposta dos ecos até cada microfone é possível determinar a posição das paredes. Porém, aqui ocorre um problema; ao se emitir um som pode ser que os ecos não cheguem na mesma ordem nos diferentes microfones. Para exemplificar, suponha uma situação onde

há dois microfones e duas paredes e emite-se um som no quarto. O primeiro microfone pode receber o sinal da parede 1 e depois da parede 2, enquanto que o outro microfone pode receber o eco da parade 2 e depois da parede 1, sem saber de qual parede veio cada eco. Neste caso temos as distâncias, mas não sabemos exatamante a que objetos elas se referem [6].

E, como último exemplo, temos a reconstrução da estrutura 3D de uma molécula, que é, atualmente, a aplicação de maior destaque da Geometria de Distâncias [7]. Neste caso também temos incertezas nas medidas das distâncias, de forma que em algumas situações as distâncias podem acabar sendo substituídas por intervalos que devem conter as distâncias corretas.

A Geometria de Distâncias é um tema que explora uma gama bastante variada de conceitos matemáticos e computacionais, como a Geometria, Álgebra Linear, Otimização, Análise Numérica, Algoritmos Computacionais, Grafos, Complexidade de Algoritmos [7], e por isso, na minha visão, representa com grande sucesso o conceito de Matemática Aplicada.

Referências

[1] D. Ivan, P. Reza, R. Juri and V. Martin, *"Euclidean Distance Matrices: A Short Walk Through Theory, Algorithms and Applications"*, IEEE Signal Process. Mag., 2014.

[2] T. F. Havel and K. Wuthrich, *"An Evaluation of the Combined Use of Nuclear Magnetic Resonance and Distance Geometry for the Determination of Protein Conformations in Solution"*, J. Mol. Biol., vol. 182, no. 2, pp. 281–294, 1985.

[3] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, *"Locating the Nodes: Cooperative Localization in Wireless Sensor Networks"*, IEEE Signal Process. Mag., vol. 22, no. 4, pp. 54-69, Jul. 2005.

[4] P. H. Schonemann, *"A Solution of the Orthogonal Procrustes Problem With Applications to Orthogonal and Oblique Rotation"*, Ph.D. dissertations, University of Illinois at Urbana-Champaign, 1964

[5] D. S. Watkins, *"Fundamentals of matrix computations"*, John Wiley & Sons, 2nd ed., cap. 4, 2002.

[6] P. Reza, *"Euclidean Distance Matrices: Properties, Algorithms and Applications"*, Ph.D dissertations, École Polythecnique Fédérale de Lausanne, 2014

[7] Lavor C., Liberti L., *"Um convite à Geometria de Distâncias"*, SBMAC, Notas em Matemática Aplicada, Vol. 71, 2014