Ciencia de Datos

Victor Muñiz

victor_m@cimat.mx

Asistente:

Víctor Gómez

victor.gomez@cimat.mx

Maestría en Cómputo Estadístico.

Centro de Investigación en Matemáticas.

Unidad Monterrey.

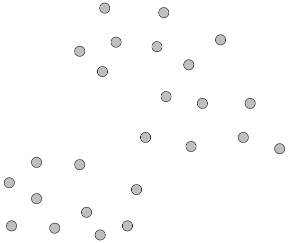
Enero-Junio 2021

Aprendizaje de variedades (Manifold Learning)

Spectral embeddings y clustering espectral

Spectral embeddings

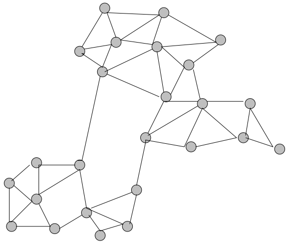
Considera los siguientes datos en 2D



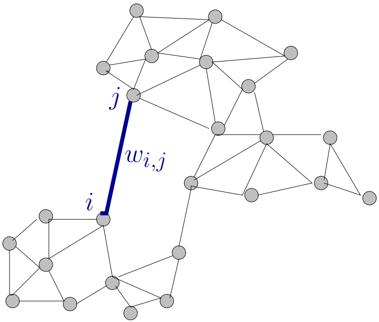
Spectral embedding es una técnica para encontrar proyecciones de baja dimensión y no lineales de los datos, basado en una representación particular de las similaridades entre ellos.

Spectral embeddings

Esta representación de las similaridades, se obtiene considerando los datos como vértices de un grafo:

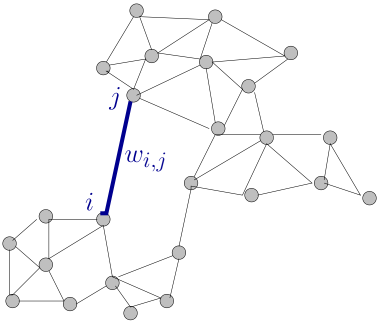


Spectral embeddings



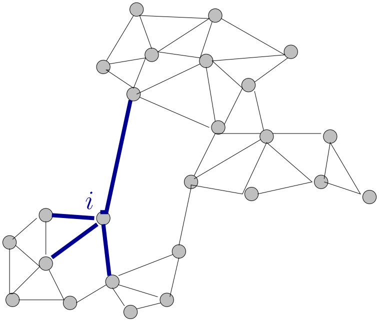
- Sea V un conjunto de n vértices. $G = (V, E)$ es una gráfica no dirigida, con pesos $w_{i,j} \geq 0$ en los arcos (edges) que unen dos vértices.
- Definimos la matriz de **adyacencias o similaridades o pesos** $\mathbf{W}$ con entradas $w_{i,j}$. Es simétrica. Si $w_{i,j} = 0$ entonces los vértices v_i y v_j no están conectados.

Spectral embeddings



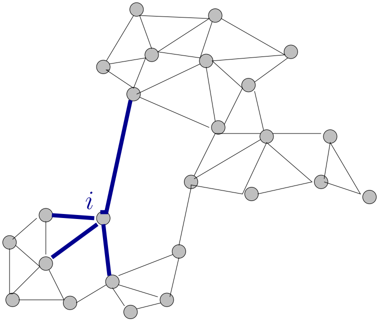
- Sea V un conjunto de n vértices. $G = (V, E)$ es una gráfica no dirigida, con pesos $w_{i,j} \geq 0$ en los arcos (edges) que unen dos vértices.
- Definimos la matriz de **adyacencias o similaridades o pesos** $\mathbf{W}$ con entradas $w_{i,j}$. Es simétrica. Si $w_{i,j} = 0$ entonces los vértices v_i y v_j no están conectados.

Spectral embeddings



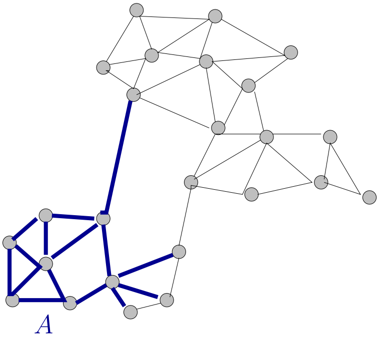
- Grado de un vértice: $d_i = \sum_{j=1}^{n} w_{i,j}$.
- Matriz de grados: $\mathbf{D}$ =diag$(d_i, \ldots, d_n)$

Spectral embeddings



- Grado de un vértice: $d_i = \sum_{j=1}^{n} w_{i,j}$.
- Matriz de grados: $\mathbf{D} = \text{diag}(d_i, \ldots, d_n)$

Spectral embeddings



- Tamaño de un conjunto $A \subset V$: $|A|$ = número de vértices en A

$$vol(A) = \sum_{i \in A}^{n} d_i$$

Spectral embeddings

- Función de similaridad. En general,

$$w_{ij} = f(d(\mathbf{x}_i, \mathbf{x}_j); \theta),$$

algunas opciones para f son similaridad de coseno y sobre todo (la que aquí veremos), la distancia Gaussiana:

$$w_{i,j} = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

- Gráficas de similaridad. Una vez calculadas las similaridades entre los puntos, se construye la gráfica de similaridad usando algún criterio para la conexión de vértices:
  - $\epsilon-$neighborhood graph
  - $k-$nearest neighbor graph
  - fully connected graph

Spectral embeddings

- Función de similaridad. En general,

$$w_{ij} = f(d(\mathbf{x}_i, \mathbf{x}_j); \theta),$$

  algunas opciones para f son similaridad de coseno y sobre todo (la que aquí veremos), la distancia Gaussiana:

$$w_{i,j} = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

- Gráficas de similaridad. Una vez calculadas las similaridades entre los puntos, se construye la gráfica de similaridad usando algún criterio para la conexión de vértices:
  - $\epsilon-$neighborhood graph
  - $k-$nearest neighbor graph
  - fully connected graph

Spectral embeddings

Una vez obtenidos los elementos del grafo, podemos obtener una representación del mismo en una matriz. Las propiedades del grafo podemos obtenerlas de ésta matriz llamada Laplaciano (spectral graph theory):

- Laplaciano no normalizado:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

Spectral embeddings

Propiedades de $\mathbf{L}$

1. Para todo $\mathbf{f} \in \mathbb{R}^n$ se tiene que

$$\mathbf{f}'\mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2$$

(usa la definición de $\mathbf{L}$, completa el cuadrado...)

2. $\mathbf{L}$ es simétrica y semidefinida positiva.
(por construcción de $\mathbf{L}$ y propiedad 1)
3. El valor propio más pequeño de $\mathbf{L}$ es 0 y su vector propio asociado es $\mathbf{1}$.
(Obvia... recuerda tu clase de álgebra matricial)
4. $\mathbf{L}$ tiene eigenvalores reales, no negativos $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
(Por 1 a 3)

Spectral embeddings

Propiedades importante de $\mathbf{L}$.

- Número de componentes conectados.
  Sea G un grafo no dirigido con pesos no negativos. Entonces la multiplicidad k del valor propio 0 de $\mathbf{L}$ es igual al número de componentes conectados $A_1, \ldots, A_k$ en el grafo. El eigenespacio del valor propio 0 está formado por los vectores indicadores $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$ de esos componentes.

Spectral embeddings

- Laplaciano normalizado de un grafo. En la literatura (Pothen et. al. 1989, Meila & Shi, 2001, por ejemplo) se han usado dos matrices para el laplaciano normalizado.

  Simétrico:

$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

  Random walk:

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

Spectral embeddings

Propiedades del Laplaciano normalizado (demostraciones similares a las anteriores, usando conceptos básicos de álgebra de matrices)

1. Para todo $\mathbf{f} \in \mathbb{R}^n$ se tiene que
$$\mathbf{f}'\mathbf{L}_{sym}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$
2. λ es un valor propio de $\mathbf{L}_{rw}$ con vector propio $\mathbf{v}$ si y solo si λ es un valor propio de $\mathbf{L}_{sym}$ con vector propio $\mathbf{u} = \mathbf{D}^{1/2}\mathbf{v}$
3. λ es un valor propio de $\mathbf{L}_{rw}$ con vector propio $\mathbf{v}$ si y solo si λ y $\mathbf{v}$ resuelven el problema generalizado de eigenvectores $\mathbf{L}\mathbf{v} = \lambda\mathbf{D}\mathbf{v}$.
4. 0 es un valor propio de $\mathbf{L}_{rw}$ con vector propio $\mathbf{1}$. 0 es un valor propio de $\mathbf{L}_{sym}$ con vector propio $\mathbf{D}^{1/2}\mathbf{1}$ y $\mathbf{L}_{rw}$ son matrices semidefinidas positivas con n valores propios reales no negativos $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Spectral embeddings

Similar al caso anterior (Laplaciano no normalizado), tenemos este resultado importante.

- Número de componentes conectados.
  Sea G un grafo no dirigido con pesos no negativos. Entonces, la multiplicidad k del valor propio 0 de L_{rw} y $\mathbf{L}_{sym}$ es igual al número de componentes conectados $A_1, \ldots, A_k$ en el grafo. Para $\mathbf{L}_{rw}$, el eigenespacio del valor propio 0 está formado por los vectores indicadores $\mathbf{1}_{A_i}$ de esos componentes. Para $\mathbf{L}_{sym}$ el eigenespacio del valor propio 0 está formado por los vectores $\mathbf{D}^{1/2}\mathbf{1}_{A_i}$.

Clustering espectral

Algorithm 1 Clustering espectral no normalizado

1: Input: datos $\{\mathbf{x}_i\}_{i=1}^n$, número de clusters k
2: Calcular las disimilaridades $\mathbf{W}_{n\times n}$ usando el kernel Gaussiano
3: Construir gráfica de similaridad
4: Calcular el Laplaciano no normalizado $\mathbf{L}$
5: Calcular los primeros k eigenvectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ de $\mathbf{L}$
6: Sea $\mathbf{V}_{n\times k}$ la matriz que contiene los k vectores propios como columnas. Realizar clustering en $\mathbf{V}$ usando $k-$medias.
7: Salida: clusters $A_1 \ldots, A_k$ con $A_i = \{j|\mathbf{x}_j \in k_i\}$

Clustering espectral

Algorithm 2 Clustering espectral normalizado (Shi & Malik, 2000)

1: Input: datos $\{\mathbf{x}_i\}_{i=1}^{n}$, número de clusters k.
2: Calcular las disimilaridades $\mathbf{W}_{n\times n}$ usando el kernel Gaussiano
3: Construir gráfica de similaridad
4: Calcular el Laplaciano normalizado $\mathbf{L}_{rw}$
5: Calcular los primeros k eigenvectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ de $\mathbf{L}_{rw}$ (que resuelven $\mathbf{Lv} = \lambda\mathbf{Dv}$)
6: Sea $\mathbf{V}_{n\times k}$ la matriz que contiene los k vectores propios como columnas. Realizar clustering en $\mathbf{V}$ usando $k-$medias.
7: Salida: clusters $A_1 \ldots, A_k$ con $A_i = \{j|\mathbf{x}_j \in k_i\}$

Clustering espectral

Algorithm 3 Clustering espectral normalizado (Ng, Jordan & Weiss, 2002)

1: Input: datos $\{\mathbf{x}_i\}_{i=1}^n$, número de clusters k.
2: Calcular las disimilaridades $\mathbf{W}_{n\times n}$ usando el kernel Gaussiano
3: Construir gráfica de similaridad
4: Calcular el Laplaciano normalizado $\mathbf{L}_{sym}$
5: Calcular los primeros k eigenvectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ de $\mathbf{L}_{sym}$
6: Sea $\mathbf{V}_{n\times k}$ la matriz que contiene los k vectores propios como columnas. Realizar la normalización por renglones $\mathbf{U}_{n\times k}$, donde $u_{ij} = \frac{v_{ij}}{\left(\sum_k v_{ik}^2\right)^{1/2}}$.
7: Realizar clustering en $\mathbf{U}$ usando $k-$medias.
8: Salida: clusters $A_1 \ldots, A_k$ con $A_i = \{j|\mathbf{x}_j \in k_i\}$

Clustering espectral

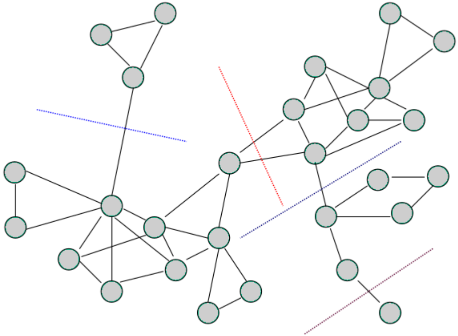
Código

`notebooks/clustering3-1.ipynb`

`notebooks/clustering3-2.ipynb`

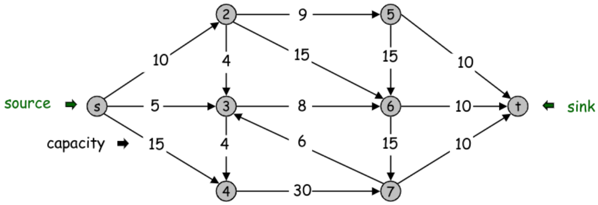
Clustering espectral

Clustering espectral y Graph Cut.



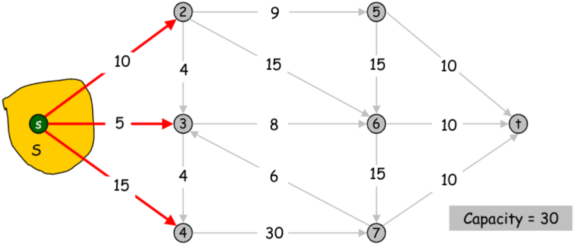
Clustering espectral

Clustering espectral y Graph Cut.



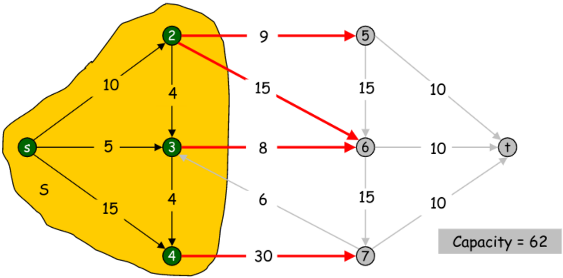
Clustering espectral

Clustering espectral y Graph Cut.



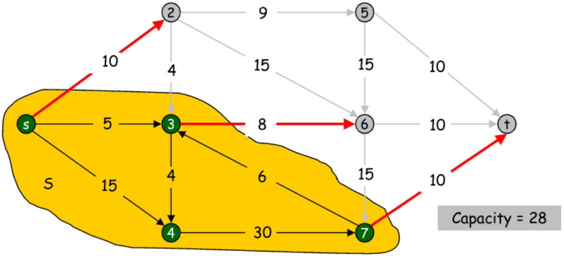
Clustering espectral

Clustering espectral y Graph Cut.



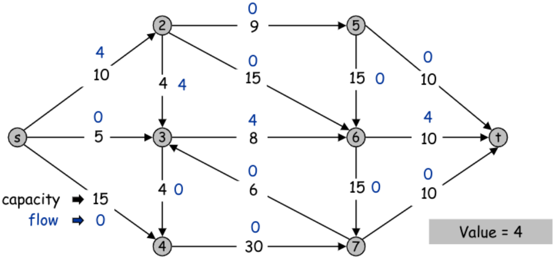
Clustering espectral

Clustering espectral y Graph Cut.



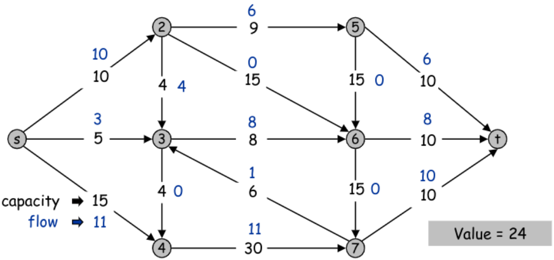
Clustering espectral

Clustering espectral y Graph Cut.



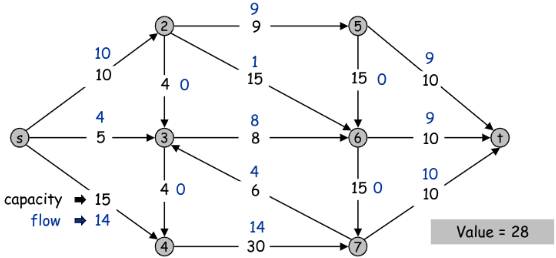
Clustering espectral

Clustering espectral y Graph Cut.



Clustering espectral

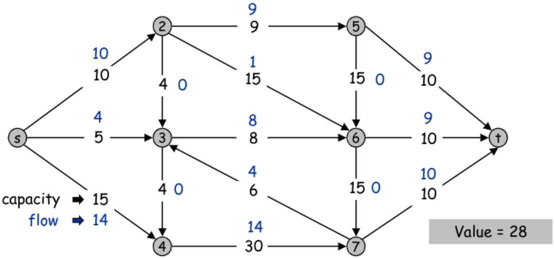
Clustering espectral y Graph Cut.



Max-flow min-cut theorem.
The maximum value of an $s-t$ flow is equal to the minimum capacity over all $s-t$ cuts.

Clustering espectral

Clustering espectral y Graph Cut.



Max-flow min-cut theorem.
The maximum value of an $s-t$ flow is equal to the minimum capacity over all $s-t$ cuts.

Clustering espectral

Para un grafo dirigido, se relaciona con

THE \$25,000,000,000* EIGENVECTOR
THE LINEAR ALGEBRA BEHIND GOOGLE

KURT BRYAN† AND TANYA LEISE‡

Abstract. Google's success derives in large part from its PageRank algorithm, which ranks the importance of webpages according to an eigenvector of a weighted link matrix. Analysis of the PageRank formula provides a wonderful applied topic for a linear algebra course. Instructors may assign this article as a project to more advanced students, or spend one or two lectures presenting the material with assigned homework from the exercises. This material also complements the discussion of Markov chains in matrix algebra. Maple and Mathematica files supporting this material can be found at www.rose-hulman.edu/∼bryan.

Clustering espectral

Queremos encontrar una partición de G tal que las conexiones entre diferentes grupos tengan bajo peso y las conexiones dentro de un grupo tengan peso alto, con la restricción de que las particiones sean lo suficientemente grandes.
Considera dos subconjuntos disjuntos $A, B \subset V$. Definimos

$$\mathsf{cut}(A, B) = \sum_{i \in A, j \in B} w_{i,j}$$

Para k particiones:

$$\mathsf{cut}(A_i \ldots, A_k) = \sum_{i=1}^{k} \mathsf{cut}(A_i, \bar{A}_i),$$

donde $\bar{A}$ es el complemento de $A \subset V$.

Clustering espectral

Para las restricciones del tamaño de las A_i usamos:

$$\mathsf{RatioCut}(A_i \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathsf{cut}(A_i, \bar{A}_i)}{|A_i|}$$

$$\mathsf{Ncut}(A_i \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathsf{cut}(A_i, \bar{A}_i)}{\mathsf{vol}(A_i)}$$

Clustering espectral

Problema a resolver para RatioCut

- $k = 2$:

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) \Leftrightarrow \min_{\mathbf{f} \in \mathbb{R}^n} \mathbf{f}'\mathbf{L}\mathbf{f} \quad \text{s. a. } \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{n}$$

La solución está dada por el eigenvector correspondiente al segundo eigenvalor más pequeño de $\mathbf{L}$.

Clustering espectral

Problema a resolver para RatioCut

- k arbitraria.
  Definimos vectores indicadores $\mathbf{h}_i$, para $i = 1, \ldots, k$, donde $h_{ij} = 1/\sqrt{|A_i|}$ si $i \in A_i$, y 0 en caso contrario.
  El problema a resolver es:

$$\min_{A_i,\ldots,A_k \subset V} \text{RatioCut}(A_i, \ldots, A_k) \quad \Leftrightarrow \quad \min_{\mathbf{H}\in\mathbb{R}^{n\times k}} Tr(\mathbf{H}'\mathbf{L}\mathbf{H})$$
$$\text{s. a. } \mathbf{H}'\mathbf{H} = \mathbf{I}$$

  La solución está dada por $\mathbf{H}$ conteniendo los primeros k eigenvectores de $\mathbf{L}$.

Con esto, se obtiene el algoritmo de clustering espectral no normalizado.
Detalles en Shi & Malik (2000).

Clustering espectral

Problema a resolver para Ncut

- $k = 2$:

$$\min_{A \subset V} \text{Ncut}(A, \bar{A}) \quad \Leftrightarrow \quad \min_{\mathbf{g} \in \mathbb{R}^n} \mathbf{g}' \mathbf{L}_{sym} \mathbf{g}$$
$$\text{s. a. } \mathbf{g} \perp \mathbf{D}^{1/2} \mathbf{1}, \|\mathbf{g}\|^2 = \text{vol}(V)$$

La solución está dada por el eigenvector correspondiente al segundo eigenvalor más pequeño de $\mathbf{L}_{sym}$.

Clustering espectral

Problema a resolver para Ncut

- k arbitraria:
  Como antes, definimos vectores indicadores $\mathbf{u}_i$, para $i = 1, \ldots, k$, donde $u_{ij} = 1/\sqrt{vol(A_i)}$ si $i \in A_i$, y 0 en caso contrario.
  El problema a resolver es:

$$\min_{A_i, \ldots, A_k \subset V} \text{Ncut}(A_i, \ldots, A_k) \quad \Leftrightarrow \quad \min_{\mathbf{U} \in \mathbb{R}^{n \times k}} Tr(\mathbf{U}'\mathbf{L}_{sym}\mathbf{U})$$
$$\text{s. a. } \mathbf{U}'\mathbf{U} = \mathbf{I}$$

  La solución está dada por $\mathbf{U}$ conteniendo los primeros k eigenvectores de $\mathbf{L}_{sym}$.
  Con esto, se obtiene el algoritmo de clustering espectral normalizado de Ng, Jordan y Weiss y el de Shi y Malik (por las propiedades de los laplacianos normalizados).

Detalles en Shi & Malik (2000).

Clustering espectral

Enfoque Random Walk (Meila y Shi, 2001).

- Una caminata aleatoria en un grafo es un proceso estocástico X_t, $t = 0, 1, 2, \ldots$ que salta aleatoriamente de un vértice a otro.
- La probabilidad de transición de un vértice i a otro j es proporcional al peso w_{ij}, y está dado por $p_{ij} = w_{ij}/d_i$.
- El clustering espectral, desde este punto de vista puede interpretarse como encontrar una partición tal que la caminata aleatoria se quede mucho tiempo dentro del mismo clúster y rara vez salte entre clusters.

Clustering espectral

Definimos la matriz de transiciones P como

$$P = \mathbf{D}^{-1}\mathbf{W}$$

si el grafo está conectado, puede mostrarse que la caminata aleatoria tiene una distribución única estacionaria π, con $\pi_i = d_i/\text{vol}(G)$.

Dos resultados importantes:

- Relación entre P y $\mathbf{L}_{rw}$:
  λ es un valor propio de $\mathbf{L}_{rw}$ con vector propio v si y solo si $1 - \lambda$ es un vector propio de P con vector propio v.

Clustering espectral

Dos resultados importantes:

- Ncut mediante probabilidades de transición:
  Considera que realizamos la caminata aleatoria $(X_t)_{t\geq 0}$ iniciando con X_0 en la distribución estacionaria π. Para conjuntos disjuntos $A, B \subset V$, denotamos $P(B|A) = P(X_1 \in B | X_0 \in A)$, entonces

  $$\text{Ncut}(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A})$$

  Entonces, al minimizar Ncut, se minimiza la probabilidad de que la caminata aleatoria salte de un cluster a otro.

Clustering espectral

Aplicación: Segmentación de imágenes (Meila y Shi, 2001. Shi y Malik, 2000).

Cada pixel de una imágen es un vértice.
Caso sencillo: segementación basado en intensidades.
Una forma sencilla de formar las similaridades es incluir un término espacial y el término de intensidad:

$$s_{i,j} = w_{i,j} = e^{\frac{-\|F(i)-F(j)\|^2}{\sigma_I}} * \gamma$$

donde

$$\gamma = e^{-\frac{\|X(i)-X(j)\|^2}{\sigma_X}}$$

si $\|X(i) - X(j)\| < r$ y 0 en caso contrario.

Referencias

- Jianbo Shi and Jitendra Malik, *Normalized Cuts and Image Segmentation*, IEEE Transactions on PAMI, Vol. 22, No. 8, Aug 2000.
- Ng, A., Jordan, M., and Weiss, Y. (2002). *On spectral clustering: analysis and an algorithm*. In T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Advances in Neural Information ProcessingSystems 14(pp. 849 – 856). MIT Press
- Ulrike Luxburg. 2007. *A tutorial on spectral clustering*. Statistics and Computing 17, 4 (December 2007), 395-416. DOI=http://dx.doi.org/10.1007/s11222-007-9033-z
- Dhillon, I.S. and Guan, Y. and Kulis, B. (2004). *Kernel k-means: spectral clustering and normalized cuts*. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 551–556.
- Marina Meilă and Jianbo Shi, *Learning Segmentation by Random Walks*, Neural Information Processing Systems 13 (NIPS 2000), 2001, pp. 873–879

t-Stochastic Neighbor Embeddings

t-SNE

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

- Una extensión de SNE (Hinton and Roweis, 2002)
- Idea: Convertir distancias (similaridades) euclideanas entre observaciones en alta dimensión en probabilidades condicionales.

t-SNE

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

- Una extensión de SNE (Hinton and Roweis, 2002)
- Idea: Convertir distancias (similaridades) euclideanas entre observaciones en alta dimensión en probabilidades condicionales.

t-SNE

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

- Una extensión de SNE (Hinton and Roweis, 2002)
- Idea: Convertir distancias (similaridades) euclideanas entre observaciones en alta dimensión en probabilidades condicionales.

t-SNE

Sea $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ datos en un espacio de entrada $\mathcal{X}$ generalmente de alta dimensión (por ejemplo, $\mathbb{R}^d$), y $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} \in \mathcal{Y}$ sus correspondientes datos transformados (manifold).
Nuestro problema es entonces

$$\min_{\mathbf{y}_i, \mathbf{y}_j} \|P_{\mathbf{x}_j|\mathbf{x}_i} - Q_{\mathbf{y}_j|\mathbf{y}_i}\|,$$

donde P, Q, son probabilidades tales que $p_{j|i}$ es la probabilidad de que $\mathbf{x}_j$ sea vecino de $\mathbf{x}_i$, y equivalentemente, $q_{j|i}$ es la probabilidad de que $\mathbf{y}_j$ sea vecino de $\mathbf{y}_i$.
En este caso, $\|\cdot\|$ es una medida de distancia definida entre distribuciones de probabilidad.

t-SNE

Para SNE, las distribuciones de probabilidad son Gaussianas: $P_{\mathbf{x}_j|\mathbf{x}_i} \sim \mathcal{N}(\mathbf{x}_i, \sigma_i^2)$ y $Q_{\mathbf{y}_j|\mathbf{y}_i} \sim \mathcal{N}(\mathbf{y}_i, 1/\sqrt{2})$, es decir,

$$p_{j|i} = \frac{e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}}},$$

$$q_{j|i} = \frac{e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq i} e^{-\|\mathbf{y}_i - \mathbf{y}_k\|^2}}$$

y $p_{i|i} = q_{i|i} = 0$

t-SNE

En SNE, la medida de distancia entre distribuciones de probabilidad es la divergencia de Kullback-Leibler, entonces nuestro problema de optimización es:

$$\begin{aligned} \min_{\mathbf{y}_i} C &= \sum_i \mathsf{KL}(P_i \| Q_i) \\ &= \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \end{aligned}$$

NOTA: varios detalles sobre la divergencia de Kullback-Leibler los vimos en clase. Espero hayas tomado notas...

t-SNE

Sobre la función de costo y su optimización (detalles en clase) para SNE:

- La minimización de C se realiza usando descenso por gradiente con momentum
- No se usa un solo valor para σ porque se espera que la densidad de los datos tenga variación en distintas regiones del espacio de entrada
- Una buena elección de σ_i es muy importante para una buena modelación de la estructura local de los datos en el espacio reducido.
- Puede verse fácilmente que la entropía de la distribución P_i aumenta cuando σ_i se incrementa, esto permite que en SNE se pueda realizar una búsqueda binaria para σ_i que produzca una distribución P_i con una “perplejidad” determinada:

$$\mathrm{Perp}(P_i) = 2^{H(P_i)} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$$

t-SNE

2 problemas con SNE:

- Complejo de optimizar, ya que es sensible a mínimos locales y se usan diversos mecanismos de regularización
- Apilamiento (crowding, el cual ya explicamos en clase)

La solución propuesta por Van der Maaten y Hinton es $t-$SNE.

- Usa una versión simétrica de la función de costo (KL-divergence)
- Usa una distribución de colas pesadas, en particular la $t-$Student para la similaridad de los puntos en el manifold. Los autores muestran que esto simplifica el problema de optimización y ayuda a solucionar el problema de crowding.

t-SNE

2 problemas con SNE:

- Complejo de optimizar, ya que es sensible a mínimos locales y se usan diversos mecanismos de regularización
- Apilamiento (crowding, el cual ya explicamos en clase)

La solución propuesta por Van der Maaten y Hinton es $t-$SNE.

- Usa una versión simétrica de la función de costo (KL-divergence)
- Usa una distribución de colas pesadas, en particular la $t-$Student para la similaridad de los puntos en el manifold. Los autores muestran que esto simplifica el problema de optimización y ayuda a solucionar el problema de crowding.

t-SNE

- Sobre la función de costo: usa KL-Div entre las distribuciones conjuntas de P y Q en lugar de las condicionales. La versión simétrica de la función de costo de SNE es:

$$
\begin{aligned}
C &= \sum_i \mathsf{KL}(P\|Q) \\
&= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}},
\end{aligned}
$$

donde $p_{ij} = p_{ji}$, $q_{ij} = q_{ji}$ $\forall i, j$ y $p_{ii} = q_{ii} = 0$.

t-SNE

- Sobre la función de costo. Y en este caso, las similaridades son

$$
\begin{aligned}
p_{ij} &= \frac{e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}}{\sum_{k \neq l} e^{\frac{-\|\mathbf{x}_l - \mathbf{x}_k\|^2}{2\sigma^2}}} \\
q_{ij} &= \frac{e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq l} e^{-\|\mathbf{y}_l - \mathbf{y}_k\|^2}},
\end{aligned}
$$

y también, se usa una versión simétrica de p_{ij} para aliviar el efecto de datos atípicos de datos en el espacio de entrada:

$$
p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}.
$$

El gradiente de la función de costo para ésta versión simétrica de SNE se simplifica notablemente (ver detalles en el paper, y las notas de clase).

t-SNE

- Sobre el problema de crowding: usa una distribución de colas pesadas en el mapeo de baja dimensión:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}},$$

que es una distribución $t-$Student con 1 grado de libertad.
El gradiente de la función de costo para $t-$SNE es (ver detalles en el paper):

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4\sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1},$$

que simplifica la minimización de la función de costo además de disminuir el efecto de crowding, entre otras ventajas.

t-SNE

Desventajas:

- El tiempo de cómputo puede ser muy grande para grandes volúmenes de datos y/o en muy alta dimensión.
- No hay una estrategia simple para elegir los parámetros del modelo, en particular, de perplejidad. Pero esto sucede con casi todos los métodos de ML... Para la perplejidad, el autor menciona[1]:
  "The performance of t-SNE is fairly robust under different settings of the perplexity. The most appropriate value depends on the density of your data. Loosely speaking, one could say that a larger / denser dataset requires a larger perplexity. Typical values for the perplexity range between 5 and 50".

[1] https://lvdmaaten.github.io/tsne/

t-SNE

Desventajas:

- El tiempo de cómputo puede ser muy grande para grandes volúmenes de datos y/o en muy alta dimensión.
- No hay una estrategia simple para elegir los parámetros del modelo, en particular, de perplejidad. Pero esto sucede con casi todos los métodos de ML... Para la perplejidad, el autor menciona[1]:
  "The performance of t-SNE is fairly robust under different settings of the perplexity. The most appropriate value depends on the density of your data. Loosely speaking, one could say that a larger / denser dataset requires a larger perplexity. Typical values for the perplexity range between 5 and 50".

[1] https://lvdmaaten.github.io/tsne/

t-SNE

Desventajas:

- No es posible hacer el embedding de datos nuevos de forma directa.
  Respecto a esto, el autor menciona[2]:
  "t-SNE learns a non-parametric mapping, which means that it does not learn an explicit function that maps data from the input space to the map. Therefore, it is not possible to embed test points in an existing map (although you could re-run t-SNE on the full dataset). A potential approach to deal with this would be to train a multivariate regressor to predict the map location from the input data. Alternatively, you could also make such a regressor minimize the t-SNE loss directly, which is what I did in this paper."
  El paper es "Learning a Parametric Embedding by Preserving Local Structure", que pondré en la página del curso.

[2] https://lvdmaaten.github.io/tsne/

t-SNE

Implementaciones: ver `TSNE` en el módulo `sklearn.manifold` de scikit Learn.

Ver los ejemplos con datos sintéticos del sitio `https://distill.pub/2016/misread-tsne/`