Reducción de dimensiones

Organización de Datos, 2020-1C

Definición

$$A \in \mathbb{R}^{n \times m} \longrightarrow \text{  } \longrightarrow A' \in \mathbb{R}^{n \times k}$$

$$(k < m)$$

Flashback: el lema de Johnson Lindenstrauss nos reducía las dimensiones con f tal que:

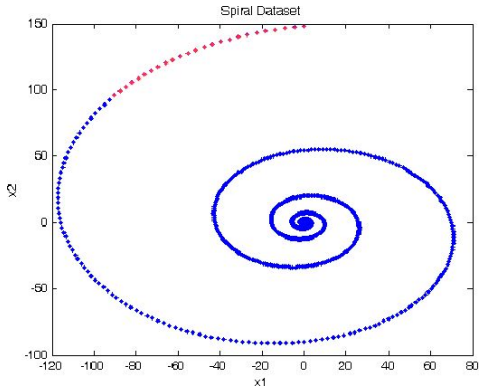
$$(1-\epsilon)\left\|u-v\right\|^2 \leq \left\|f(u)-f(v)\right\|^2 \leq (1+\epsilon)\left\|u-v\right\|^2$$

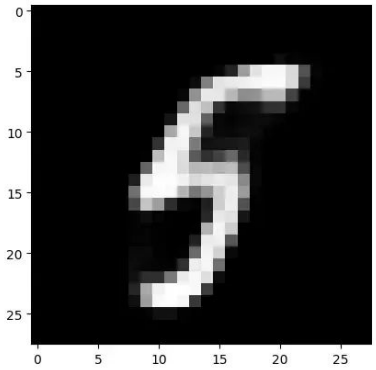
Motivaciones para reducir dimensiones

- Visualización
  - Buscar agrupamientos y relaciones con variables categóricas
- Reducir tiempo de cómputo
  - Ej. algoritmos que no escalan bien en dimensiones
- Obtener los features *importantes* del dataset
  - *Manifold theory*

Manifold theory + manifold learning

- Las dimensiones *reales* de los datos generalmente no son iguales a las dimensiones *aparentes*
  - i.e. no es posible cualquier combinación de las columnas (no existe una casa con 15 baños y una habitación)
- Concepto de grados de libertad y restricciones





¿Qué hay en la caja?

- ¿Es importante?
  - Sí
- Fundamental en visualización
  - Según el método hay relaciones y patrones que agregan significado o nos pueden llevar a la ruina
- Es importante tener alguna intuición de los parámetros

Análisis de componentes principales: PCA

- Método lineal
- Queremos encontrar las **componentes** que maximicen la varianza de los datos
  - Para datos en *m* dimensiones originales, obtenemos *m* componentes ortogonales
  - La clave es que están ordenadas según maximizan la varianza (ej. podemos quedarnos con las primeras 2)
- Una componente es un vector (=> CL de las dimensiones originales)

$$\text{Varianza} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

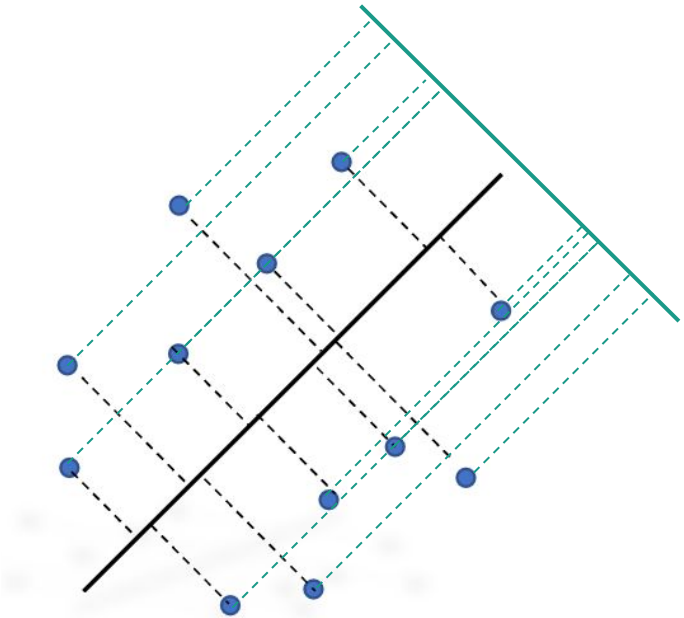
$$(\mu = \quad)$$

$$\mathbf{a} \text{ proyectado en } \mathbf{b} = \mathbf{a} \cdot \frac{\mathbf{b}}{\|\mathbf{b}\|}$$

dispersión de los datos según **w**

$$\Rightarrow \sum_i \left(\mathbf{x}_{(i)} \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \boldsymbol{\mu} \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \right)^2$$

Intuición sobre la dispersión de los datos



¿Cómo son las varianzas de las proyecciones?

Análisis de componentes principales: PCA

- Podemos centrar los datos de antemano (🦶 = 0)
- Las componentes que PCA nos da son vectores **unitarios** (||w|| = 1)

$$\Rightarrow \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2$$

- Efectivamente la primer componente principal maximiza esto:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2$$

- Un poco de magia: $\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \|\mathbf{X}\mathbf{w}\|^2 = \arg\max_{\|\mathbf{w}\|=1} \mathbf{w}^T\mathbf{X}^\mathbf{T}\mathbf{X}\mathbf{w}$

Análisis de componentes principales: PCA

- El $w_{(1)}$ que maximiza la ecuación anterior es el autovector con autovalor más grande de $\mathbf{X}^T\mathbf{X}$
- Para obtener $w_{(k)}$, la ecuación es la misma, solo que restamos los primeros $w_{(1)},...,w_{(k-1)}$ a los datos.
- Resulta que también $w_{(k)}$ es el autovector de $\mathbf{X}^T\mathbf{X}$ cuyo autovalor es el k-más-alto
    - E.g. $w_{(2)}$ es el segundo autovector de $\mathbf{X}^T\mathbf{X}$.

Método de PCA a través de la covarianza

Para una matriz de datos X:

1. Centramos las columnas, i.e., a cada columna le restamos el promedio de sus valores
2. (Opcional) Dividir cada elemento por la desviación estándar (√Varianza) de su columna
3. Calcular la matriz de covarianza

La matriz de covarianza nos da información interesante sobre los features

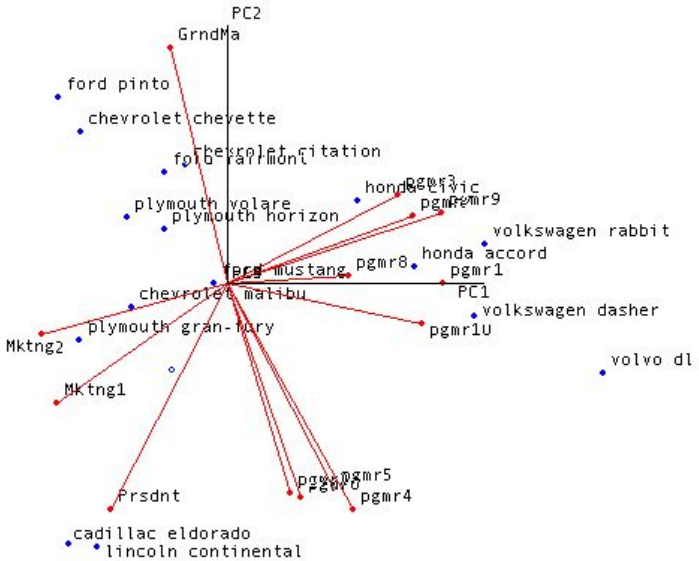
$$\frac{1}{n-1}\mathbf{X}^{\mathbf{T}}\mathbf{X}$$

El n-1 es un truquito estadístico (corrección de Bessel)

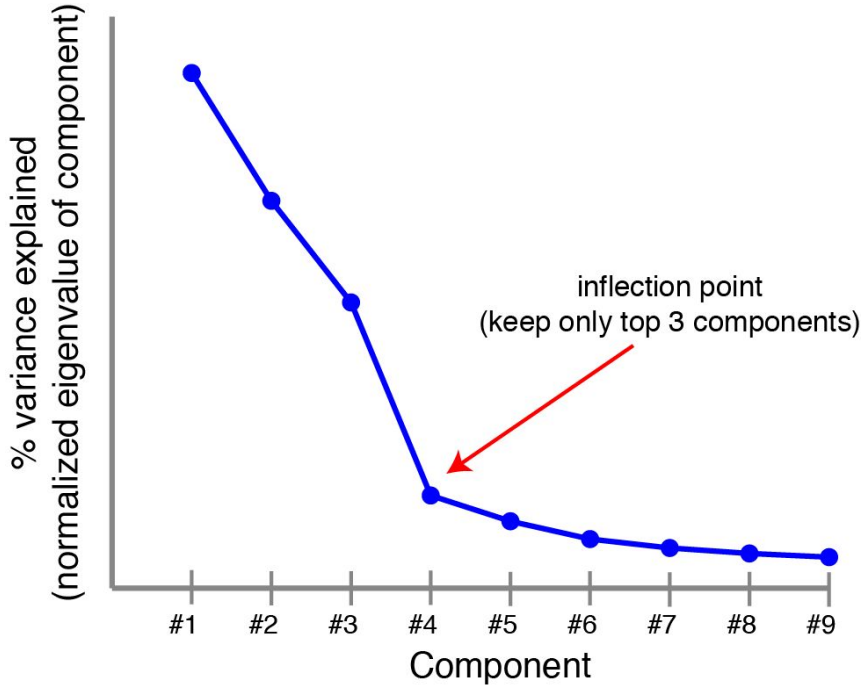
4. Ordenamos los autovectores de la matriz de covarianza según sus autovalores, de mayor a menor, esos son los componentes principales.

Interpretación de un biplot

- Podemos proyectar los puntos
- Pero también las dimensiones!
  - Son los vectores unitarios



¿Cuántas dimensiones necesitamos?



Descomposición en valores singulares

Recordemos que cualquier matriz se puede descomponer en un producto de tres matrices:

$$X = U\Sigma V^T \qquad \left(X \in \mathbb{R}^{n\times m}, U \in \mathbb{R}^{n\times n}, \Sigma \in \mathbb{R}^{n\times m}, V \in \mathbb{R}^{m\times m}\right)$$

U es una matriz unitaria; sus columnas son los autovectores de $\mathbf{XX^T}$
Σ es una matriz diagonal cuyos elementos son las raíces de los autovalores de $\mathbf{XX^T}$ o $\mathbf{X^TX}$ ordenados de mayor a menor ("**valores singulares**").
V es una matriz unitaria; sus columnas son los autovectores de $\mathbf{X^TX}$

Descomposición en valores singulares reducida

La mejor aproximación de rango **r** es:

$$\hat{X} = U_r \Sigma_r V_r^T \quad \left(\hat{X} \in \mathbb{R}^{n \times m}, U_r \in \mathbb{R}^{n \times r}, \Sigma_r \in \mathbb{R}^{r \times r}, V_r \in \mathbb{R}^{m \times r}\right)$$

Nos quedamos con los primeros **r** valores singulares.

Reducción de dimensiones con SVD

Recordemos la relación:

$$XV = U\Sigma$$

Esto nos dice que proyectar los datos con V es lo mismo que escalar U con los valores singulares.

Si usamos la versión reducida de la SVD, $\mathbf{U}_r$ $\mathbf{\Sigma}_r$ es una reducción a **r** dimensiones de los datos originales.

Teorema: SVD = PCA

Observemos que $\mathbf{V}$ tiene los autovectores de $\mathbf{X}^T\mathbf{X}$.

En la aproximación de rango $\mathbf{r}$ nos quedamos con las primeras $\mathbf{r}$ columnas de $\mathbf{U}$, $\mathbf{\Sigma}$ y $\mathbf{V}$. Como están ordenados según los valores singulares, nos quedamos con los AVEs más importantes

=> Hacer $\mathbf{X}\mathbf{V}_r$ es PCA

=> Hacer $\mathbf{U}_r\mathbf{\Sigma}_r$ es lo mismo que PCA!

Esto funciona si teníamos centrada la matriz antes.

Cuando con un paquete hacemos PCA, internamente se realiza la SVD porque es más estable numéricamente.

Multidimensional scaling

1. Elevar la matriz de distancias al cuadrado
2. Centrar la matriz para que las filas y columnas tengan promedio cero
3. Calculamos la SVD de la matriz
4. Las primeras **k** columnas de U dan las coordenadas para los puntos reducidos

Motivación para lo que sigue

1. En teoría de grafos hay un problema conocido que trata de hallar el *corte* más “esparso”
2. Una aproximación es analizando la matriz de adyacencia
3. Definimos el Laplaciano del grafo: L = D - A
    a. D: matriz diagonal con los grados
    b. A: matriz de adyacencia
4. El laplaciano tiene muchísimas propiedades
    a. El autovalor más pequeño es 0
    b. Si tiene multiplicidad > 1, el grafo no es conexo
5. **Teorema:** usando el AVE asociado al AVA no-nulo más pequeño podemos asignar a cada nodo del grafo un valor; según eso, ordenarlos y establecer el corte más esparso entre dos grupos.

Laplacian eigenmaps

- Vamos a aprovechar esto para poder reducir las dimensiones
  a. Esos AVEs nos permiten capturar diferenciar bien los nodos (símil “capturar varianza” en PCA)
- En esencia:
  a. Construimos un grafo a partir de nuestros puntos
  b. Calculamos el laplaciano
  c. Si queremos *k* dimensiones, usamos *k* AVEs
- Bonus: este método es no lineal

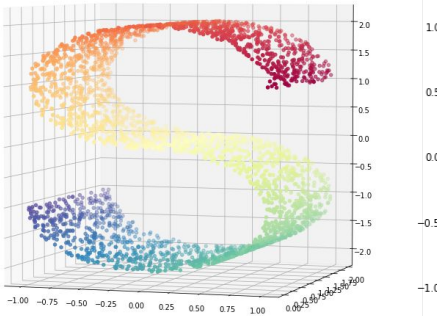
Laplacian Eigenmaps

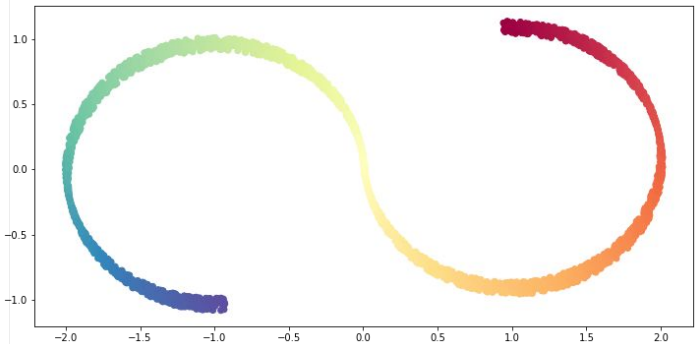
Ahora sí, el método:

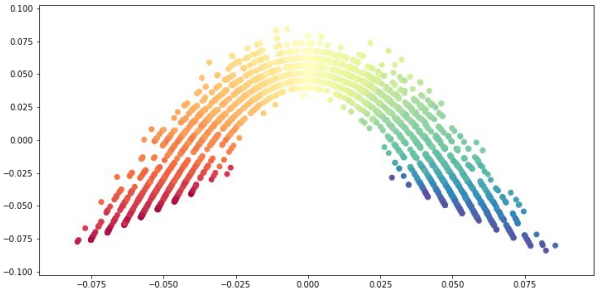
1. Armamos una matriz de adyacencias con los *v* vecinos más cercanos
2. Asignamos pesos a esas aristas con un kernel RBF: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$
3. Calculamos el laplaciano: L = D - A
4. Hacemos descomposición espectral: tomamos los *k* autovectores más chicos (pero ignorando el autovalor 0) y asignamos los elementos a los nodos correspondientes

Laplacian Eigenmaps

El hecho de que no sea lineal permite, a veces, mejores resultados:

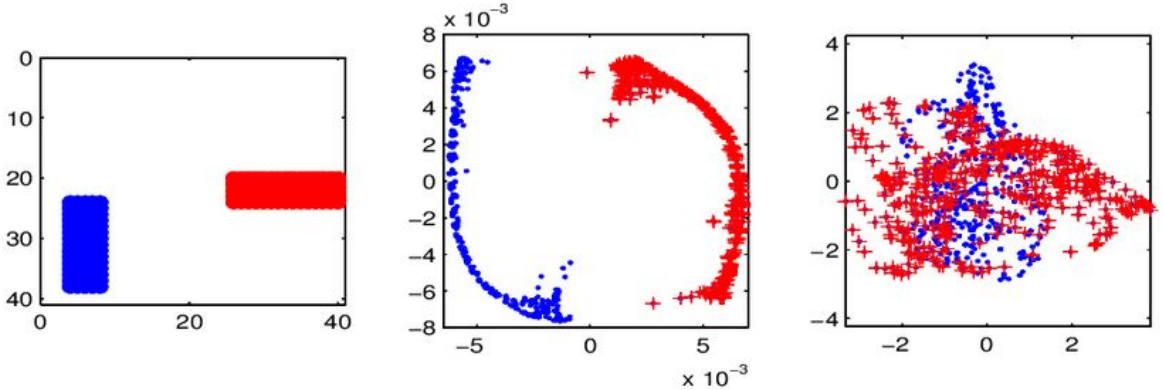




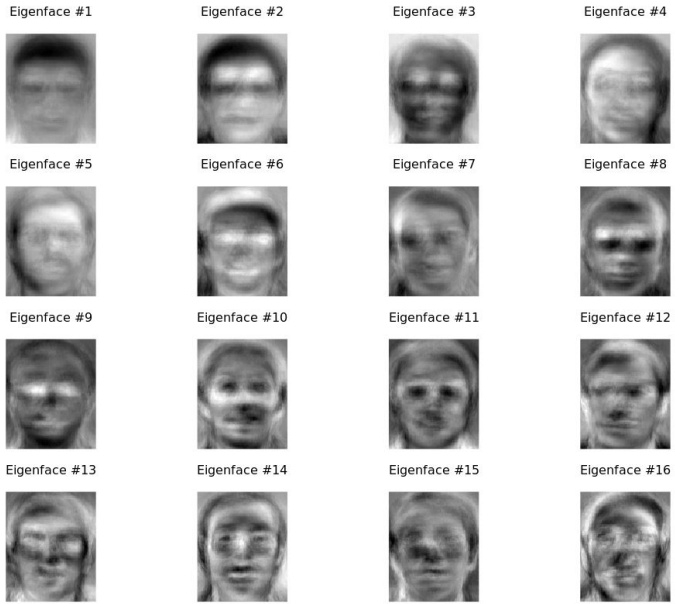


Laplacian Eigenmaps

El hecho de que no sea lineal permite, a veces, mejores resultados:



Aplicación: eigenfaces



Teorema fundamental de la reducción de dimensiones

No siempre es mejor reducir las dimensiones de nuestro conjunto de datos.