Reducción de la dimensionalidad

Alfredo Cuesta Infante

E. T. S. Ingeniería Informática
Universidad Rey Juan Carlos

Master Univ. en Visión Artificial
Reconocimiento de Patrones

Contenidos

Contenidos

Motivación

Hemos visto técnicas de aumentado de la dimensionalidad que nos permitían hacer clasificación no lineal, es decir más expresiva y general que la lineal:

- Ingeniería de características
- Transformaciones no lineales de las características
- Utilización de Kernels

entonces...

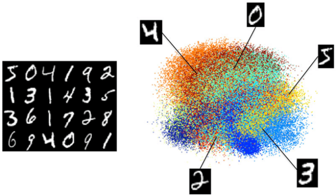
¿Para qué?

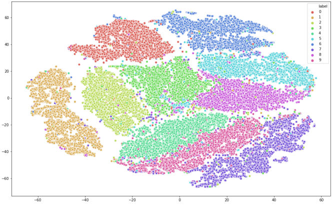
- Reduce el coste computacional, en ejecución y en memoria.
- Nos permite visualizar los datos cuando estos tienen alta dimensionalidad.
  - Las técnicas de visualización **no** son capaces de mostrar más de 6 o 7 características.
    Por ej. en este sitio web se describen diferentes técnicas para mostrar hasta 6 dimensiones
- Puede facilitar la separación lineal.

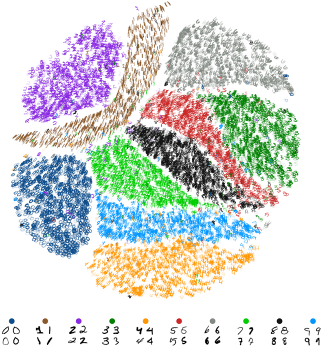
¿Cómo?

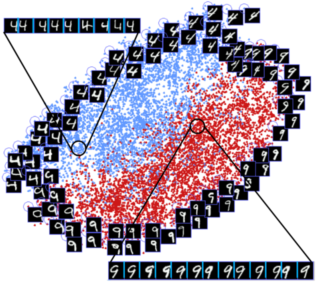
- Eliminando características redundantes
- Eliminando características poco correladas con la etiqueta
- Proyectando a subespacios

Motivación









Ejemplo de selección por correlación

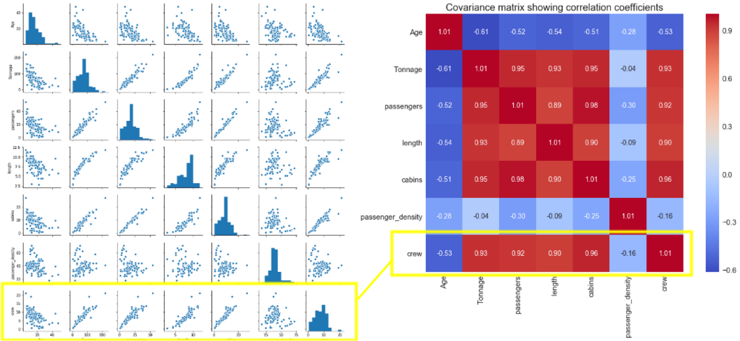


Figura: Si la etiqueta fuera la última columna de la tabla, entonces las características más relacionadas con ella son la 2ª, 3ª, 4ª y 5ª

Ejemplo de reducción por proyección

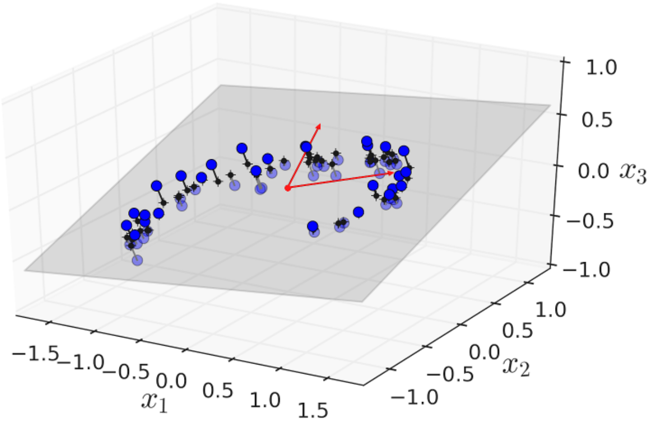
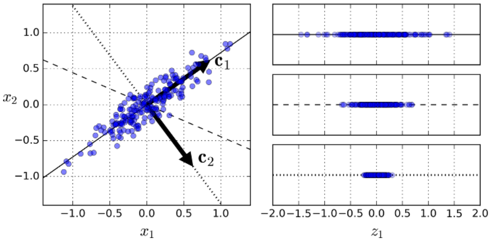


Figura: Reducción de 3 a 2 dimensiones proyectando sobre el subespacio que más información recoge

Contenidos

Principal Component Analysis (PCA)

- Buscar la proyección que conserva la mayor varianza de los datos.

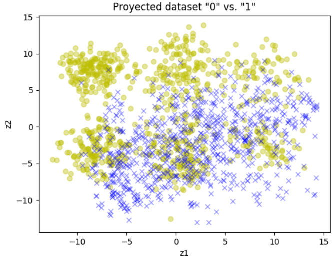


- Datos $= \mathbf{X}$, descomponiendo en (SVD) $\mathbf{X} = \mathbf{U\Sigma V}^T$.

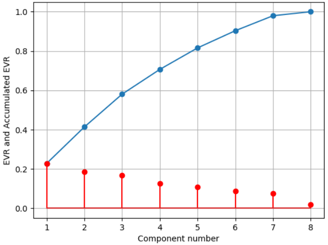
  Σ son los componentes principales, y las matrices se reordenan de tal manera que la diagonal de Σ esté ordenada de mayor a menor.
- Las columnas de $\mathbf{V}^T$ son una base del espacio donde viven $\mathbf{X}$
- Con los n primeros tengo la base de un subespacio, y $\mathbf{X}_{\text{proy}} = \mathbf{XW}$ es la proyección de los datos en esa base.

  $\rightarrow$ es decir, los datos con dimensión reducida.
- Con $\mathbf{U}$, $\mathbf{V}$ y Σ podemos reconstruir $\mathbf{X}$ a partir de cualquier $\mathbf{X}_{\text{proy}}$.

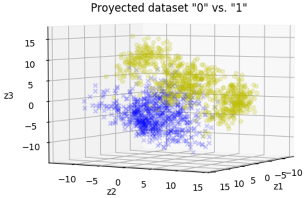
Principal Component Analysis (PCA)



(a)



(c)



(b)

```
First 8 principal components
[[-0.24  0.49  0.79 -0.02  0.20  0.08  0.00 -0.14]
 [ 0.01  0.01  0.07  0.53 -0.48  0.54  0.40 -0.00]
 [-0.12 -0.02 -0.15 -0.48  0.32  0.22  0.75  0.00]
 [-0.14 -0.03 -0.09 -0.38 -0.02  0.75 -0.50  0.02]
 [ 0.04 -0.06 -0.11 -0.00 -0.02  0.00 -0.01 -0.98]
 [-0.05 -0.84  0.43  0.13  0.27  0.07 -0.00 -0.00]
 [-0.67  0.07 -0.35  0.47  0.42  0.03 -0.07 -0.00]
 [-0.66 -0.18  0.07 -0.29 -0.59 -0.26  0.04 -0.01]]

First 8 singular values
[ 259.1  233.7  221.8  192.6  179.4  160.5  149.8  77.6]

Explained Variance Ratio
[ 0.22   0.18   0.16   0.12   0.10   0.08   0.07   0.02]

Accumulated Explained Variance Ratio
[ 0.22   0.41   0.58   0.70   0.81   0.90   0.97   1.]
```

(d)

Figura: (a-b) ‘0’ vs. ‘1’ con los 2 y 3 primeros componentes principales. (c) Porcentaje de varianza explicado según se añaden componentes principales, en rojo, y su acumulado, en azul. (d) Resultados numéricos.

Kernel PCA

- Sea $\mathbf{K} = \mathbf{X}^T\mathbf{X}/(m-1)$ la matriz de covarianza de los datos,
  $\mathbf{K} = \mathbf{W\Lambda W}^T$ su descomposición en autovalores, y
  $\mathbf{X} = \mathbf{U\Sigma V}^T$ sea la SVD del conjunto de datos.
- Reescribimos $\mathbf{XX}^T = \left(\mathbf{U\Sigma V}^T\right)\left(\mathbf{U\Sigma V}^T\right)^T = \mathbf{U\Sigma}(\mathbf{V}^T\mathbf{V})\mathbf{\Sigma U}^T$.
- Como $\mathbf{VV}^T = \mathbf{V}^T\mathbf{V} = \mathbf{1}$ nos queda $\mathbf{XX}^T = \mathbf{U\Sigma\Sigma U}^T = \mathbf{U\Sigma}^2\mathbf{U}^T$.
- Por tanto $\mathbf{X}^T\mathbf{X}/(m-1) = \mathbf{W\Lambda W}^T = \mathbf{U\Sigma}^2\mathbf{U}^T\,/(m-1)$.
- La matriz de autovalores de la descomposición de la covarianza es el cuadrado de la matriz de valores singulares del conjunto de datos, salvo por la constante.
- ¿Y si en vez de calcular $\mathbf{X}^T\mathbf{X}$ utilizamos un kernel $\mathcal{K}(\mathbf{X},\mathbf{X}^T)$ que nos devuelva este resultado en otro espacio diferente?
- Obtenemos *Kernel-PCA*
  - Infinitos vectores principales $\rightarrow$ ya no "existe" la reconstrucción.
    Python tiene un método de reconstrucción basado en aprender la función de mapeo del conjunto de datos original al conjunto de datos proyectado.

Kernel PCA

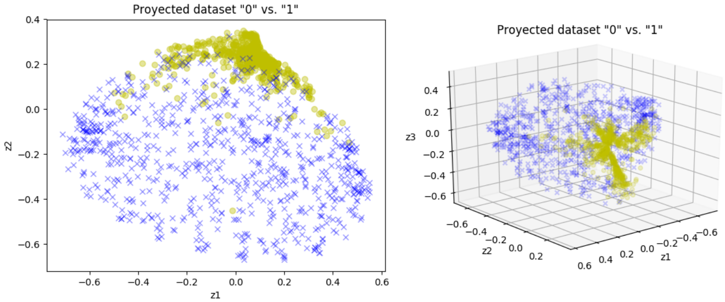


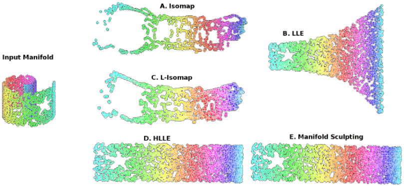
Figura: Proyección del conjunto de datos '0' vs. '1' en los 2 y 3 primeros componentes principales con un PCA de kernel RBF, con parámetro 0.01. [Fuente: Original de A. Cuesta]

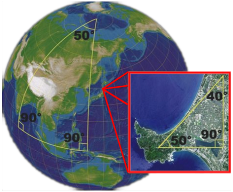
Contenidos

Locally Linear Embedding (LLE)

Objetivo. Proyectar nuestros puntos sobre una **variedad** *manifold*

- Una variedad es un espacio donde, a escala local, se conservan las propiedades euclídeas. *Por ej. el mapa de una ciudad*





1. Identificar los k vecinos más próximos de cada ejemplo $\mathbf{x}^{(i)}$,
2. Se reconstruye dicho ejemplo mediante una función lineal de los vecinos

$$\mathbf{x}^{(i)}_{\mathtt{rec}} = \sum_{j=1}^{k} w^*_{ij}\mathbf{x}^{(j)} \text{ , tal que } \mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{m} \left\| \mathbf{x}^{(i)} - \sum_{j=1}^{k} w_{ij}\mathbf{x}^{(j)} \right\|^2$$

   Los pesos se normalizan de manera que $\sum_{j=1}^{m} w_{ij} = 1,\ i = 1, \ldots, m$.

3. Sea $\mathbf{Z} = \{\mathbf{z}^{(i)}\}$ el mapeo de $\mathbf{X} = \{\mathbf{x}^{(i)}\}$:

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} \sum_{i=1}^{m} \left\| \mathbf{z}^{(i)} - \sum_{j=1}^{k} w^*_{ij}\mathbf{z}^{(j)} \right\|^2$$

Locally Linear Embedding (LLE)

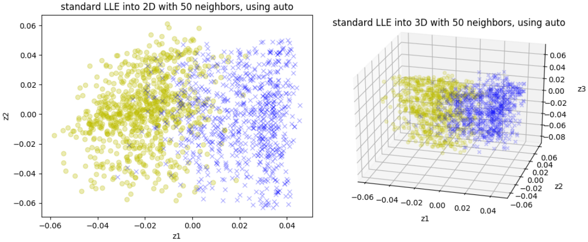


Figura: Proyección del conjunto de datos '0' vs. '1' en una variedad 2D (izq.) y en una 3D (der.) mediante LLE estandar, con 50 vecinos y método 'auto'. [Fuente: Original de A. Cuesta]

Contenidos

Linear Discriminant Analysis (LDA)

- **Asumiendo** que
  - tenemos dos clases,
  - los ejemplos de cada clase tienen distribución MVN,
  - la covarianza es la misma para todas las MVN

➔ hiperplano discriminante = LDA

La proyección sobre el hiperplano ortogonal al discriminante tendrá el menor solapamiento posible.

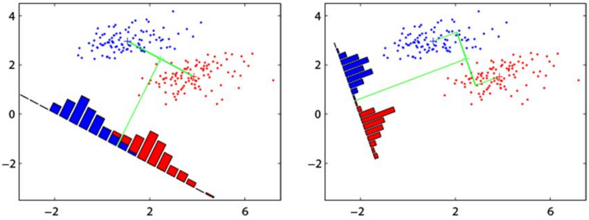


Figura: Las proyecciones de las clases se solapan más a la izquierda que a la derecha. LDA nos asegura la máxima separación. [Fuente: "Pattern recognition and machine learning" C.M. Bishop]

Las condiciones son muy fuertes por lo que no es muy efectivo.