



Machine Learning y Data Analytics

HW 9 - Grupo 1

Ian Amighini
Julieta Brey
Lorenzo Nasti
Camila Sobrino
Matias Rodriguez Brun

Profesor Titular: Sergio Pernice

Consigna

Entender el código y responder a las preguntas: ¿Por qué PCA funciona mejor en un estudio y regresiones en el otro?

(2.a) ¿Por qué PCA funciona mal con aleatoriedad solo en Y?

```
"""2.a

Consideramos el caso donde x es la variable independiente (también conocida como explicativa),
a la que conocemos perfectamente, pero solo podemos obtener mediciones "ruidosas" de y, o mas
específicamente, con errores de medicion en y que en general se distribuyen de manera normal.

Alternativamente, podemos pensar que y depende de x, y quizas tambien de otras variables que
no estan tenidas en cuenta en el modelo, pero cuyos efectos se pueden simular con una
aleatoriedad Gaussiana.

Fix X = [x1, x2,..., x1000] = [0.001, 0.002, 0.003,..., 1]. xi = i/1000, i = 1,...,1000
yi = 2*xi + Norm(0,s^2) = 2*i/1000, Y = [y1, y2,..., y1000].

1.c.1 Por que pasa lo que pasa en esta celda? """

"""Datos"""

n = 10000
s = 0.5
x = np.arange(0.0, 1, 1/n)
y = 2*x + s*np.random.randn(n)
```

(2.b) ¿Por qué PCA funciona peor cuanto mayor es la aleatoriedad en Y (y solo en Y)?

```
"""2.b

Para cada s en [0, 0.05, 0.1,..., 0.45, 0.5] calculamos
30 veces la pendiente predicha por pca y ls y hacemos un scatter plot.

De nuevo la misma pregunta, por que pasa lo que pasa? """
```

```
"""Estudio"""
n = 1000
x = np.arange(1/n, 1+1/n, 1/n)
s = np.linspace(0, 0.5, 11)
prc = np.zeros(11)
regr = np.zeros(11)
```

(3.a) ¿Por qué PCA funciona bien con aleatoriedad en X e Y?

```
"""3.a

Ahora consideramos el caso donde para ambas variables, x e y, solo podemos obtener mediciones
"ruidosas", o con errores de medicion en que en general se distribuyen de manera normal para ambas.

Alternativamente, podemos pensar que y depende de x, y quizas tambien de otras variables que
no estan tenidas en cuenta en el modelo, pero cuyos efectos se pueden simular con una
aleatoriedad Gaussiana, y ademas a la variable x la medimos con cierto error que simulamos
como Gaussiano.

Ahora consideremos el caso donde ambos, X e Y tienen ruido
xi = i/1000 + Norm(0,s^2), ; yi = 2*xi + Norm(0,s^2), i = 1,...,1000
X = [x1, x2,..., x1000] , Y = [y1, y2,..., y1000].

Hacer los mismos estudios que en 2.a"""

"""Datos"""

# jugar con diferentes valores de n y s
n = 10000
s = 0.2
z = np.arange(0.0, 1, 1/n)
x = z + s*np.random.randn(n)
y = 2*z + s*np.random.randn(n)
```

(3.b) ¿Por qué la regresión funciona mal con aleatoriedad en X e Y? ¿Por qué la regresión funciona peor cuanto más grande es el error en ambas variables?

```
"""Estudio"""
n = 1000
z = np.arange(1/n, 1+1/n, 1/n)
s = np.linspace(0, 0.5, 11)
prc = np.zeros(11)
regr = np.zeros(11)
```

(4) Por qué en la "nube cuadrada" PCA da cualquier cosa mientras que regresiones siempre da lo mismo?

```
"""4

Los elementos de X e Y son elegidos al azar de forma idéntica e independiente
(por ejemplo, cada elemento está distribuido uniformemente en el cuadrado [0; 1] x [0; 1]).
Repetir varias veces y mirar los gráficos ¿Qué genera PCA y qué LS?, Por qué?

Jugar con el parámetro n y s. También pasar de

np.random.rand a np.random.randn (ajustando l para escalar apropiadamente la región
del gráfico)

Por qué pasa lo que pasa?"""

"""Datos"""

n = 10000
s = 1.0
x = s*np.random.rand(n,1)
y = np.random.rand(n,1)
```

Introducción

En análisis de datos lineales existen dos enfoques populares para ajustar una “línea central” a una nube de puntos (X_i, Y_i) :

- **Regresión lineal ordinaria (OLS)**

- Minimiza la suma de los **errores verticales** $\sum_i (Y_i - (\beta X_i + \alpha))^2$.
- Asume que **X es exacta** (sin ruido) y todo el error está en Y.
- Recupera la pendiente verdadera siempre que **solo Y** esté contaminada por aleatoriedad.

- **Análisis de Componentes Principales (PCA)**

- Encuentra la dirección (autovector) que maximiza la **varianza total**, midiendo distancias **ortogonales** a la recta.
- Trata simétricamente ruido en X e Y, sin privilegiar uno sobre otro.
- Recupera la dirección real cuando ambas variables tienen error, pero falla si el ruido está desbalanceado.

ENTONCES...

Si el único ruido está en Y, OLS es óptimo (minimiza precisamente ese error).

Si el ruido está en ambas variables, PCA es más adecuado (captura la dispersión conjunta).

Analogía visual: imagina una nube de puntos y quieres trazar una línea “al centro”:

- OLS “tira” de la línea ajustándose para que los puntos queden lo más cerca posible en vertical.
- PCA la orienta para que explique la mayor dispersión total, sin privilegiar vertical u horizontal.

Análisis

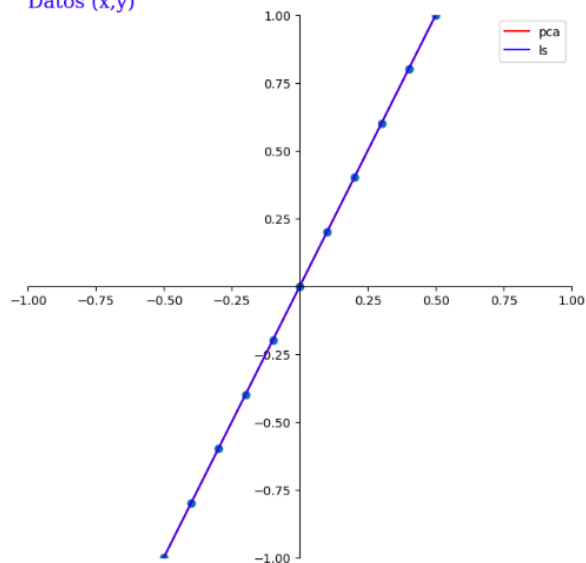
Para comprender bien cuándo funciona mejor que enfoque, en el código se plantean diferentes escenarios:

1. **Datos ideales (sin ruido)**

- Generas datos perfectamente lineales:
 $Y=2X$

sin añadir aleatoriedad.

Datos (x,y)



- ¿Qué sucede?
 - La nube de puntos está exactamente sobre una línea de pendiente 2.
 - OLS elige pendiente = 2 (cero errores verticales).
 - PCA el primer componente apunta sobre esa misma línea de máxima varianza, también pendiente = 2.
- Conclusión: Sin ruido, ambas recuperan perfectamente la relación verdadera.

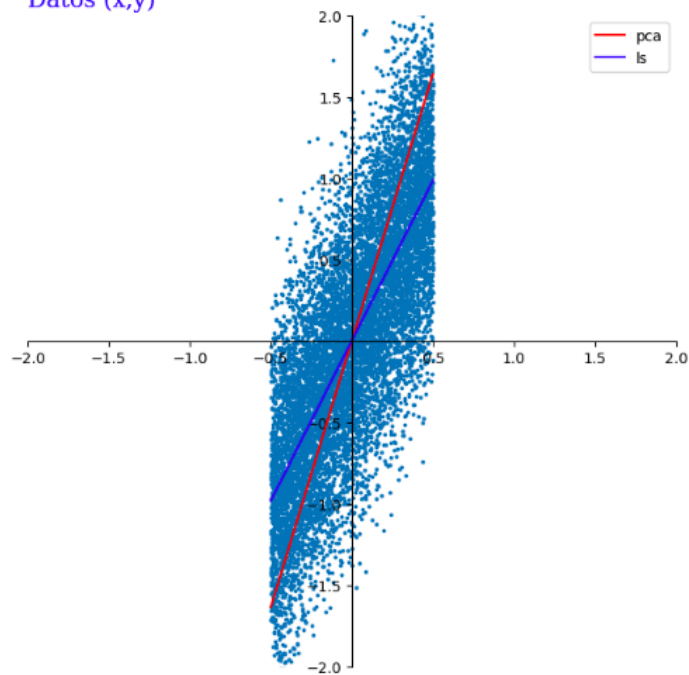
2. Ruido *solo* en Y

Se añade aleatoriedad a Y (ruido vertical), dejando X exactas.

- **2.a ¿Por qué PCA funciona mal con ruido solo en Y?**
 - Al crecer el ruido, la nube se “estira” verticalmente.
 - PCA busca la dirección de máxima varianza total: con mucho más en Y, la componente principal se inclina casi vertical (pendiente muy grande),

ignorando la verdadera relación $Y \approx 2X$.

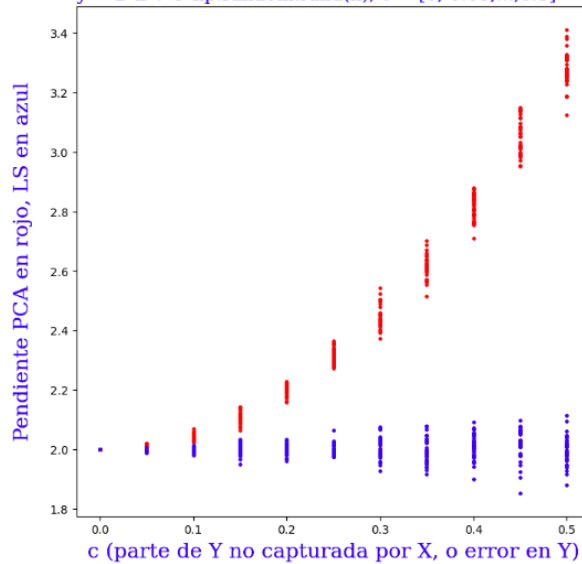
Datos (x,y)



- 2.b ¿Por qué empeora cuanto mayor es el ruido en Y?

27

$y = 2*x + c*np.random.rand(n)$, $c = [0, 0.05, ..., 0.5]$



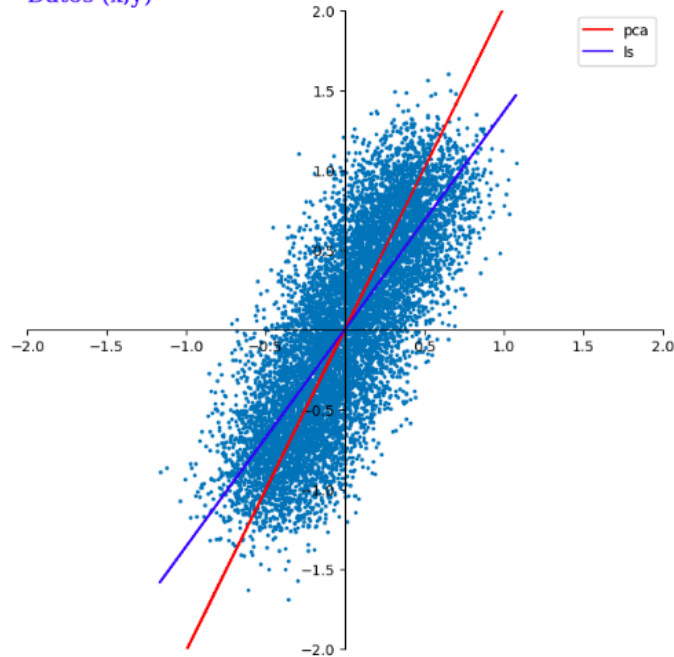
- A más varianza en Y vs X, la línea de PCA rota cada vez más hacia el eje vertical.
- En el límite de ruido muy grande, la PCA daría prácticamente “x = constante” (línea vertical), capturando sólo el ruido, no la pendiente real.

3. Ruido en X y Y

Se añaden errores tanto a X como a Y.

- 3.a ¿Por qué PCA funciona bien con ruido en ambas variables?

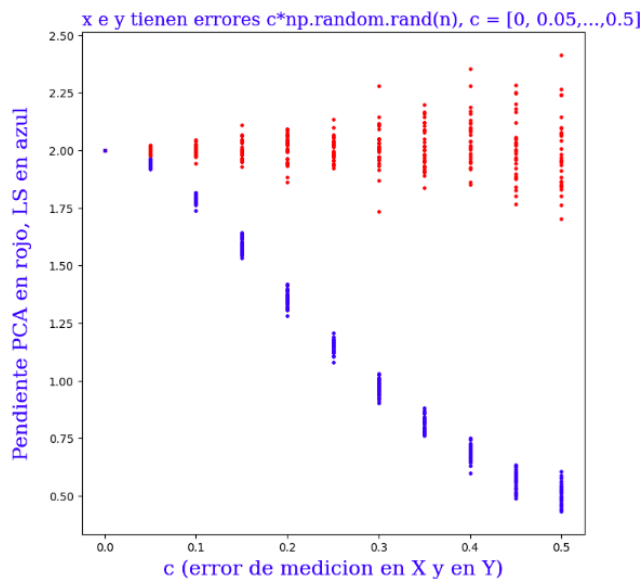
Datos (x,y)



- La nube queda “difusa” alrededor de la recta original, formando una elipse cuyo eje mayor sigue la pendiente verdadera.
- PCA identifica ese eje como componente principal, recuperando la pendiente subyacente.

- 3.b ¿Por qué la regresión OLS funciona mal y empeora con más ruido?

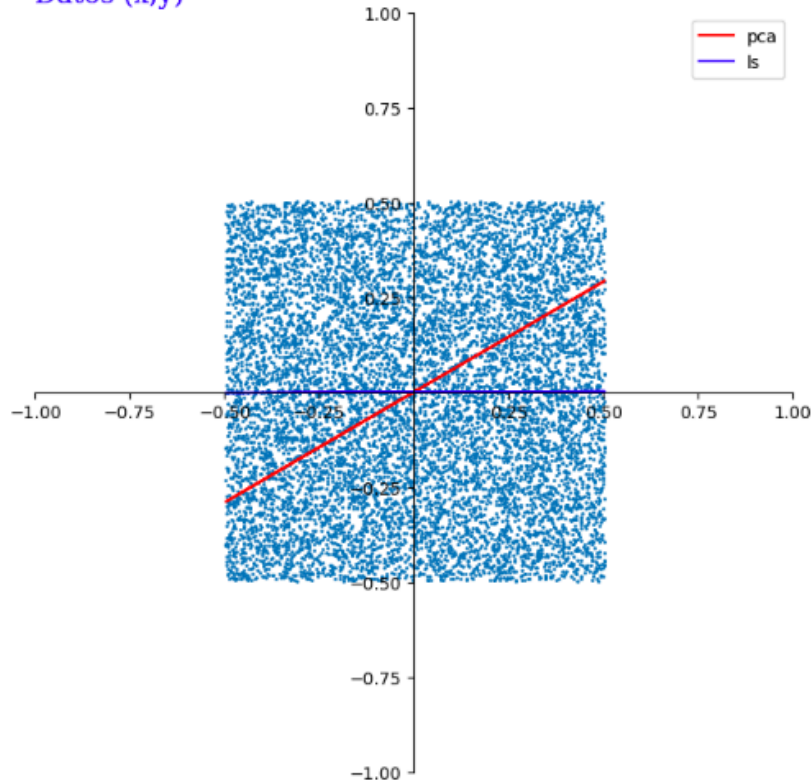
- OLS asume X sin error. Si X tiene ruido, la fórmula $\hat{\beta} = \frac{\text{Cov}(X,Y)}{\text{Var}(X)}$ sufre sesgo de atenuación: la pendiente se “achata” hacia 0.
- Cuanto más ruido en X, más decrece $\text{Cov}(X,Y)$ relativo a $\text{Var}(X)$, y más pequeña (distorsionada) sale la pendiente.



4. “Nube cuadrada” (X e Y independientes)

Ambas variables se generan sin relación (por ejemplo, uniformes e independientes).

Datos (x,y)




- PCA
 - La varianza es igual en todas las direcciones (distribución isótropa).
 - El autovector principal resulta arbitrario: cada ejecución puede devolver una pendiente distinta.
- OLS
 - Como no hay correlación, $\text{Cov}(X,Y) \approx 0$.
 - La pendiente siempre sale casi 0 (línea horizontal), de forma consistente.

Conclusiones

Habiendo analizado lo que sucede en cada escenario planteado por el código, podemos llegar a la conclusión de que la elección entre PCA y regresión lineal depende de la distribución del ruido en las variables:

Escenario	Mejor método	Por qué
1. Sin ruido	Ambos	Ningún método “falla”: la relación verdadera es clara y se recupera perfectamente (pendiente = 2).
2. Ruido solo en Y	OLS	Minimiza exactamente el error vertical en Y; PCA se desvía hacia el eje vertical y funciona mal cuanto mayor es la aleatoriedad en Y.
3. Ruido en X y Y	PCA	Encuentra la dirección de máxima dispersión conjunta; OLS sufre sesgo de atenuación y funciona peor cuanto más grande es el error en X e Y.
4. Sin relación real	Ninguno útil	PCA da resultados arbitrarios (varianza isotropa); OLS mantiene pendiente ≈ 0 , reflejando falta de correlación.

Código

 TP_PCA2.ipynb.txt