FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
UNIVERSIDAD DE CHILE

Big Data Analytics en Confiabilidad y Mantenimiento

Adquisición y procesamiento de señales para el diagnostico y pronóstico de fallas

Viviana Meruane N.

Introducción

Introducción

Hoy en día se hace cada vez más importante poder anticipar las fallas de un sistema de manera de planificar acciones de mantenimiento y así reducir costos y riesgos.

El desarrollo de técnicas de "Prognostics and Health Managment" (PHM) juega un rol cada vez más importante. Siendo el proceso de pronóstico uno de los temas principales de investigación a nivel mundial.

Introducción

Desarrollar un sistema de PHM adecuado permite anticiparse a las fallas en sistemas o componentes críticos, lo que previene riesgos y aumenta la seguridad de las personas y bienes.

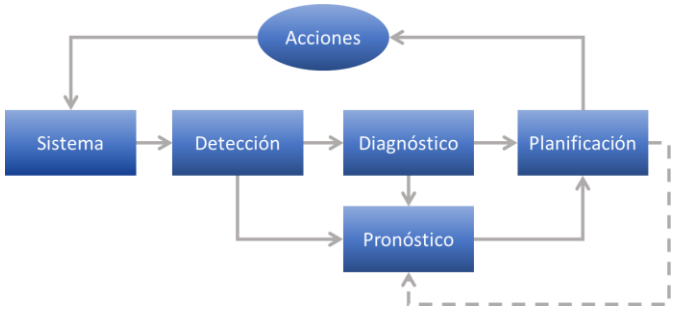
Adicionalmente, el pronóstico de daños está en línea con los principios de sustentabilidad, esto es, un aumento de la disponibilidad y vida útil de los sistemas.

Introducción

Una estrategia de PHM está compuesta de tres etapas principales:

1. Detección: Trata de identificar el modo de operación del sistema y su estado.
2. Diagnóstico: Cuando se detecta una anomalía, el diagnóstico identifica el componente que tiene la anomalía y su severidad (de los efectos a las causas).
3. Pronóstico: El pronóstico trata de predecir el estado futuro del sistema y su vida útil remanente (de las causas a los efectos).

Introducción



Detección, diagnóstico, pronóstico y planificación en un sistema de mantenimiento inteligente.

Para lograr la detección, diagnóstico y pronóstico de las fallas de un sistema se requiere de un sistema adecuado de adquisición y procesamiento de los datos.

Adquisición de datos

Estrategia general para la obtención de datos para PHM:

Sistema

1. Selección de los componentes críticos
2. Variables físicas a monitorear y sensores a instalar
3. Adquisición y pre-procesamiento de los datos

Datos para PHM

Componentes críticos

Un componente crítico se define como un componente cuya falla lleva a una detención del sistema completo y cuya frecuencia de fallas es alta.

Sistema

1. División en subsistemas
2. Descripción técnica (componentes, relaciones funcionales, etc.)
3. Análisis de fallas:
   - Pareto
   - Árboles de falla
   - FMECA
   - HAZOP
   - …

Componentes críticos

Variables físicas a monitorear

El monitoreo del estado de salud de un componente se realiza mediante el análisis de algunos de sus parámetros físicos u operacionales. La selección de estos parámetros es muy importante ya que una mala selección puede llevar a la no detección o a falsas alarmas.

Sin embargo, esta selección requiere de un conocimiento profundo entre la evolución de la degradación y las variables que se ven afectadas.

Por ejemplo, las vibraciones axiales en un rodamiento entregan información sobre la presencia de defectos en las bolas, jaula o pistas. Por otro lado, la medición de la humedad alrededor del rodamiento no es pertinente y no va a ayudar en la detección de estas fallas.

Variables físicas a monitorear

Ejemplo de variables físicas a medir:

Dominio	Variable física
Térmico	Temperatura, flujo de calor, disipación térmica.
Eléctrico	Voltaje, corriente, resistencia eléctrica, inductancia, impedancia, capacitancia, constante dieléctrica, carga, polarización, campo eléctrico, frecuencia, potencia, nivel de ruido.
Mecánico	Largo, área, volumen, desplazamiento, velocidad, aceleración, flujo, fuerza, densidad, densidad relativa, rigidez, fricción, presión, emisión acústica.
Químico	Concentración química, reactividad.
Humedad	Humedad relativa, humedad absoluta.
Biológico	PH, concentración de moléculas biológicas, micro-organismos.
Óptico	Intensidad lumínica, fase, largo de onda, polarización, reflectancia, transmitancia, refracción, amplitud, frecuencia.
Magnético	Campo magnético, momento magnético, permeabilidad, dirección, posición, distancia.

Adquisición y almacenamiento

La adquisición, almacenamiento y pre-procesamiento de las señales representa el tercer paso en PHM.

Tipos de sensores

Para una cierta variable física a monitorear existen distintos tipos de sensores. Por ejemplo, sensores generadores de corriente, sensores de generación de carga, sensores resistivos, sensores inductivos y sensores capacitivos.

Ejemplo de sensores para medición de fuerza y aceleración

Tipos de sensores

Las consideraciones más relevantes al momento de seleccionar un sensor son:

- Desempeño: un set de características metrológicas de los sensores (precisión, linealidad, sensibilidad, etc.).
- Confiabilidad: los sensores deben ser seleccionados de manera que no alteren la confiabilidad del sistema monitoreado.
- Costo: se deben considerar los costos para que la solución sea competitiva.
- Número y posición: el número de sensores depende de cada aplicación (redundancia, cobertura). La posición también es relevante y debe seleccionarse de manera que se puedan medir las variables deseadas.

Tipos de sensores

Las consideraciones más relevantes al momento de seleccionar un sensor son:

- Tipo de fijación: existen diferentes soluciones para la fijación (pegamento, atornillado, magnético o con cera) y la selección depende de la calidad requerida para las mediciones y del ambiente de operación del componente.
- Dimensión y peso: se debe considerar la forma, tamaño, peso y cubierta de cada sensor de forma de no influenciar las mediciones y de respectar las restricciones de espacio.
- Ambiente: los sensores deben soportar las variaciones ambientales a las que serán sometidos (temperaturas, humedad, radiación nuclear o electromagnética, etc.)

Sistema de adquisición

El sistema de adquisición está compuesto por la tarjeta de adquisición, un computador, un software de adquisición y procesamiento y eventualmente un disco externo para el almacenamiento de grandes volúmenes de datos.

La tarjeta de adquisición puede instalarse en el pc o venir como un sistema externo. La mayoría de las tarjetas vienen un sistema de conversión análogo-digital. Cada tarjeta viene con un software de adquisición y procesamiento. Este software permite configurar, la frecuencia de muestreo, tiempo de adquisición y otros.

Sistema de adquisición

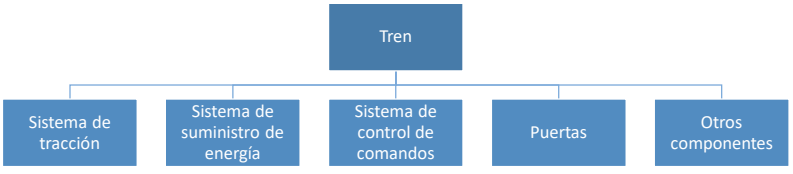
Los parámetros más importantes de una tarjeta de adquisición son: la resolución (bits), velocidad de transferencia de datos, búfer de datos, numero de canales, frecuencia máxima de adquisición.

Los datos adquiridos son guardados en archivos en distintos formatos (usualmente .txt, .csv o .mat). Estos datos pueden ser leídos, analizados y procesados posteriormente por programas o algoritmos desarrollados especialmente para aplicaciones de PHM.

Ejemplo - PHM en rodamientos

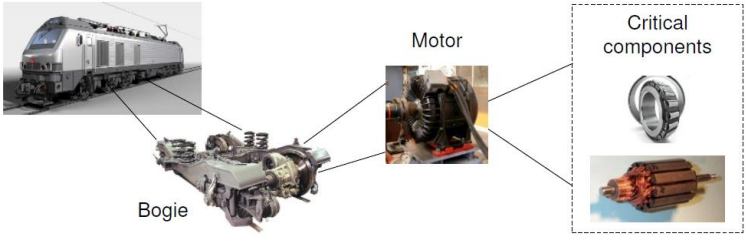
Para ilustrar el procedimiento consideremos un tren de pasajeros como caso de estudio.

El tren es un sistema complejo, compuesto por varios sub-sistemas.



Ejemplo - PHM en rodamientos

El sistema de tracción en particular, esta compuesto por el bogie que lleva el motor eléctrico. Este último consta de un rotor (basado en un imán permanente) y un estator (basado en bobinas).



Ejemplo - PHM en rodamientos

Del análisis de fallas se concluye que los rodamientos y el estator son los componentes asociados al mayor número de fallas.

Failure percentage %					
Component	Bloch & Geitner [BLO 99]	O'Donnell [O'DO 85]	IEEE-ERPI [LAN 02]	Albrecht *et al.* [ALB 86]	Alstom transport
Bearings	41	45–50	45–55	41	40
Stator	37	30–40	26–36	36	38
Axle	10	8–12		9	10
Other	12			14	12

Distribución de fallas en motores

Ejemplo - PHM en rodamientos

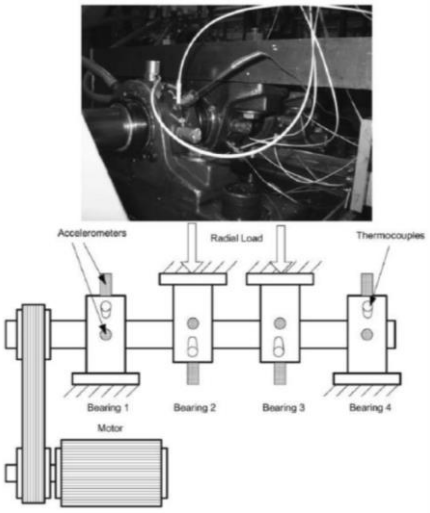
De un análisis de Análisis de Modos y Efectos de Falla y Criticidad (FMECA) se concluyó que los rodamientos tienen la mayor criticidad. Esto se debe al hecho que su falla es suficiente para detener el eje del motor, lo que puede llevar a una detención del tren.

Los rodamientos pueden fallar por distintos motivos relacionados a desgaste, mala lubricación o presencia de objetos extraños, entre otros. Por lo tanto, no es fácil definir a priori como son las mediciones de un rodamiento defectuoso y es necesario realizar un estudio más detallado.

Ejemplo - PHM en rodamientos

Para generación la información se trabajó en un montaje experimental que permite operar los rodamientos en condiciones desfavorables (velocidad y carga). Obteniendo una gran cantidad de información sobre la degradación del rodamiento.

La velocidad de giro es constante a 2000 RPM, se aplica una carga 6000 lbs (2721 kgf).



Ejemplo - PHM en rodamientos

Ejemplo de degradación en rodamiento.

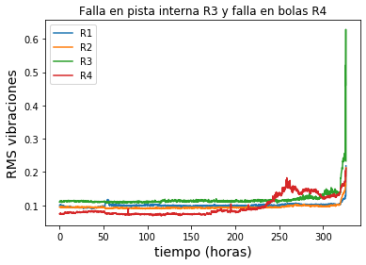
Ejemplo 1

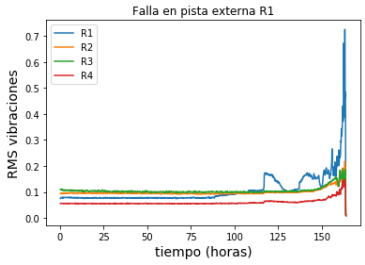
- Abrir Anaconda Navigator y luego desde el home abrir el programa Spider.
- En Spider, seleccionar Archivo -> Nuevo
- Seleccionar como directorio de trabajo la carpeta donde están los archivos

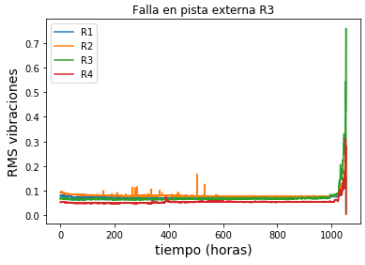
Ejemplo 1

```
import pandas as pd
import matplotlib.pyplot as plt

#registro de datos de vibraciones en RMS
Datos1 = pd.read_csv('RMSvibraciones1.txt')
Datos2 = pd.read_csv('RMSvibraciones2.txt')
Datos3 = pd.read_csv('RMSvibraciones3.txt')

#registro de tiempo en minutos
t1 = pd.read_csv('time1.txt')
t2 = pd.read_csv('time2.txt')
t3 = pd.read_csv('time3.txt')

plt.figure()
plt.plot(t1/60,Datos1)
plt.xlabel('tiempo (horas)', fontsize=14)
plt.ylabel('RMS vibraciones', fontsize=14)
plt.legend(['R1','R2','R3','R4'])
plt.title('Falla en pista interna R3 y falla en bolas R4')

plt.figure()
plt.plot(t2/60,Datos2)
plt.xlabel('tiempo (horas)', fontsize=14)
plt.ylabel('RMS vibraciones', fontsize=14)
plt.legend(['R1','R2','R3','R4'])
plt.title('Falla en pista externa R1')

plt.figure()
plt.plot(t3/60,Datos3)
plt.xlabel('tiempo (horas)', fontsize=14)
plt.ylabel('RMS vibraciones', fontsize=14)
plt.legend(['R1','R2','R3','R4'])
plt.title('Falla en pista externa R3')
```

Ejemplo 1



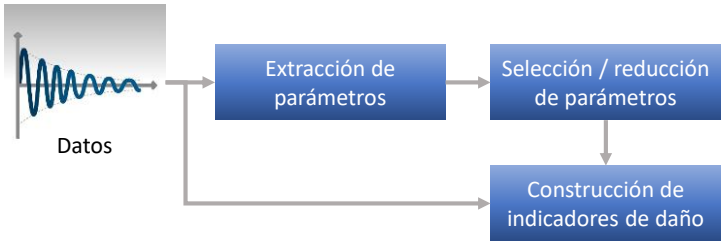




Procesamiento de los datos

Procesamiento de los datos

Los datos adquiridos tienen información relevante sobre la aparición y evolución de una falla. Sin embargo, su uso directo no es sencillo y es necesario procesarlos.



Procesamiento de los datos

- **Extracción de parámetros**: tiene por objetivo transformar la señal bruta en otra señal en el dominio del tiempo, frecuencia o tiempo-frecuencia, la que es luego procesada para construir indicadores que contengan información relacionada a la degradación en el sistema en estudio.

  La extracción de parámetros requiere de conocimiento del fenómeno físico involucrado y además depende del uso. Los parámetros usados para detección pueden ser distintos a los usados para diagnóstico o pronóstico.

Procesamiento de los datos

- **Selección/reducción de parámetros:** Es bastante común la extracción de más parámetros de los realmente necesarios, lo que dificulta el procesamiento. Hay dos estrategias para resolver esto:
  - Selección de parámetros: Es posible seleccionar un set de parámetros de acuerdo a algún criterio. Existen diferentes métricas que miden que tan adecuados son los parámetros.
  - Reducción de parámetros: Los parámetros se proyectan en un espacio de dimensión menor mientras que se mantiene la varianza de los datos.

Procesamiento de los datos

- **Construcción de indicadores de daño:** Los indicadores de daño son señales construidas a partir de la señal original o de los parámetros seleccionados. En ambos casos, se requiere de procesamiento (fusión de datos, filtros, etc.) con el propósito de obtener descriptores que contengan suficiente información para revelar el estado del componente.

Extracción de parámetros

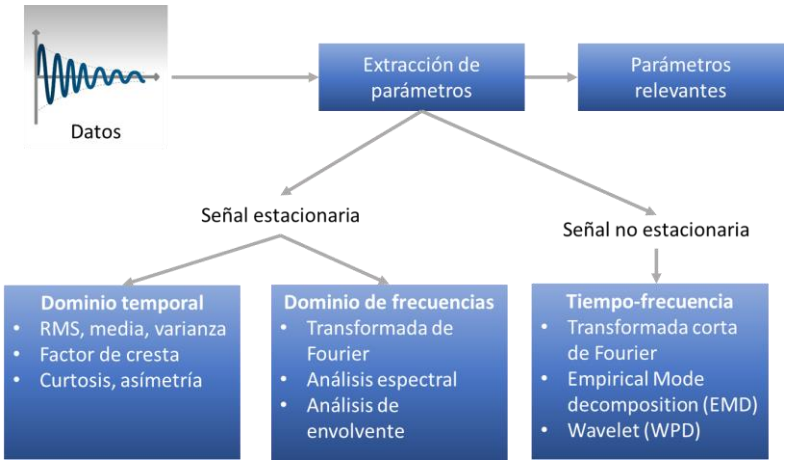
Extracción de parámetros

La extracción de parámetros consiste en procesar los datos brutos para construir indicadores (parámetros) que se puedan interpretar o que al menos contengan suficiente información para los algoritmos de detección, diagnóstico y pronóstico.

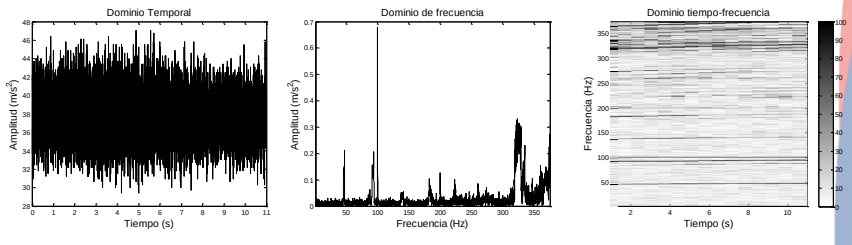
Los métodos de extracción dependen del dominio de la señal y se pueden separar en dos categorías, dependiendo si la señal es o no estacionaria.

- Una señal estacionaria, en la que sus propiedades promedio no varían en el tiempo. Se pueden utilizar métodos en el dominio temporal o de frecuencia.
- Una señal no estacionaria varía en el tiempo. Por lo tanto, se deben utilizar métodos en el dominio tiempo-frecuencia.

Extracción de parámetros



Extracción de parámetros



Parámetros en el dominio del tiempo

Dominio del tiempo

Los parámetros en el dominio temporal se basan en el calculo de parámetros estadísticos de la señal.

Consideremos una señal $x(t) = \{x_1 \quad x_2 \quad \cdots \quad x_n\}$, se pueden calcular los siguientes parámetros:

- **RMS,** El valor RMS describe el contenido de energía de la señal. RMS se utiliza para evaluar el estado general de los componentes. Debido a esto no es muy sensible a fallo incipiente, sin embargo, se utiliza para realizar un seguimiento de la progresión de fallo general:

$$x_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}$$

Dominio del tiempo

- **Valor peak**, el valor peak es la máxima amplitud de la señal en cierto intervalo de tiempo.:

$$x_p = \max_i x_i$$

- **Amplitud Peak to Peak**, corresponde a la distancia entre la amplitud máxima y mínima de una señal :

$$x_{p-p} = \max_i x_i - \min_i x_i$$

- **Factor de cresta**, da cuenta de la diferencia entre el peak de una señal respecto a su valor RMS. El valor del Factor cresta es normalmente entre 2 y 6. Un factor cresta mayor a 6 indica un posible fallo de la máquina.

$$CF = \frac{x_p}{x_{RMS}}$$

Dominio del tiempo

- **Media aritmética**, se refiere al valor promedio de la señal en un intervalo de tiempo

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- **Varianza** (σ^2)(segundo momento)

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

Dominio del tiempo

- **Asímetría** (tercer momento), indica la simetría en la amplitud de la función de densidad de probabilidad de una serie de tiempo. Una serie de tiempo con un número igual de amplitudes grandes y pequeñas tiene una asimetría de cero.

$$\gamma_3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^3}{\sigma^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2}\right)^3}$$

Dominio del tiempo

- **Curtosis** (cuarto momento), es un parámetro estadístico que da cuenta de cuan agudo son en promedio los peaks de una señal. Si el valor kurtosis es cercano a 3, se tendrá una distribución Gaussiana en relación a los peaks. Kurtosis mayores a 3 implican peaks más puntiagudos y Kurtosis menores a 3 implican peaks más planos.

$$\gamma_4 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^4}{\sigma^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2\right)^2}$$

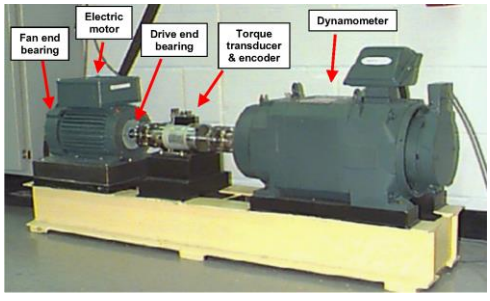
Dominio del tiempo

- Momentos centrales de orden mayor

$$\gamma_m = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^m}{\sigma^m} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^m}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{i-}\mu)^2}\right)^m}$$

Ejemplo 2

- Se introdujeron fallas en los dos rodamientos que soportan el eje del motor
- Se midieron las vibraciones en ambos rodamientos.
- Velocidad de giro aproximado 1700 RPM



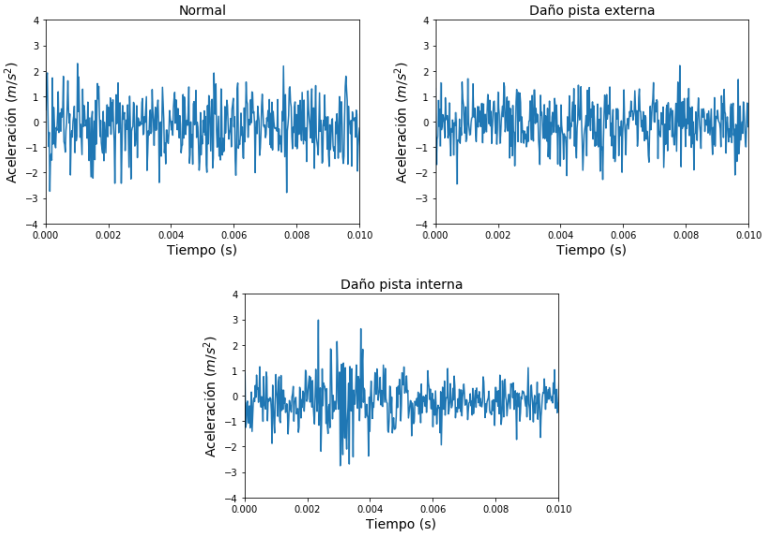
Ejemplo 2

```
#importar librerías
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.stats import kurtosis, skew
from numpy import mean, sqrt, square

#Leer datos
Datos0=sio.loadmat('normal.mat')
Datos1=sio.loadmat('outer.mat')
Datos2=sio.loadmat('inner.mat')
Normal=Datos0['normal']
Outer=Datos1['outer']
Inner=Datos2['inner']

#vector de tiempo
Fs=48828 #sampling rate
dt=1/Fs #paso de tiempo
N=len(Normal)
t=np.linspace(0,dt*(N-1),N)
```


Ejemplo 2

```
plt.plot(t,Normal)
plt.xlabel('Tiempo (s)', fontsize=14)
plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
plt.title('Normal', fontsize=14)
plt.xlim(0,0.01)
plt.ylim(-4,4)
plt.show()

plt.plot(t,Outer)
plt.xlabel('Tiempo (s)', fontsize=14)
plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
plt.title('Daño pista externa', fontsize=14)
plt.xlim(0,0.01)
plt.ylim(-4,4)
plt.show()

plt.plot(t,Inner)
plt.xlabel('Tiempo (s)', fontsize=14)
plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
plt.title('Daño pista interna', fontsize=14)
plt.xlim(0,0.01)
plt.ylim(-4,4)
plt.show()
```

Ejemplo 2



Ejemplo 2

```
#calcular parametros por tramos
L=5000 #largo de los segmentos
l=1000 #overlap
Nt=math.floor((N-l)/(L-l)) #total de tramos


#inicializar matrices con parametros
Pn=np.zeros((Nt,8))
Po=np.zeros((Nt,8))
Pi=np.zeros((Nt,8))
```

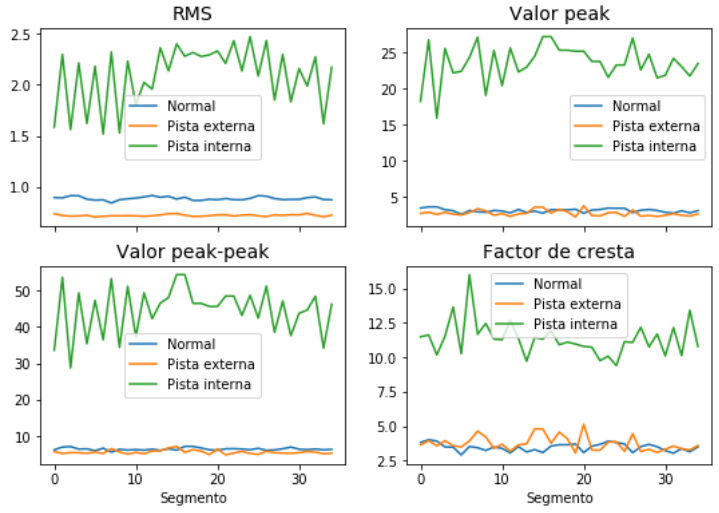
Ejemplo 2

```
for i in range(1,Nt+1):
    inicio=(i-1)*L-(i-1)*l+1
    fin=i*L-(i-1)*l

    Pn[i-1,0]=sqrt(mean(square(Normal[inicio:fin]))) #RMS
    Pn[i-1,1]=np.amax(Normal[inicio:fin]) #Peak
    Pn[i-1,2]=np.amax(Normal[inicio:fin])-np.amin(Normal[inicio:fin]) #peak-peak
    Pn[i-1,3]=Pn[i-1,1]/Pn[i-1,0] #crest
    Pn[i-1,4]=np.mean(Normal[inicio:fin]) #Media
    Pn[i-1,5]=np.var(Normal[inicio:fin]) #var
    Pn[i-1,6]=skew(Normal[inicio:fin])[0] #asimetria
    Pn[i-1,7]=kurtosis(Normal[inicio:fin])[0] #curtosis

    Po[i-1,0]=sqrt(mean(square(Outer[inicio:fin]))) #RMS
    Po[i-1,1]=np.amax(Outer[inicio:fin]) #Peak
    Po[i-1,2]=np.amax(Outer[inicio:fin])-np.amin(Outer[inicio:fin]) #peak-peak
    Po[i-1,3]=Po[i-1,1]/Po[i-1,0] #crest
    Po[i-1,4]=np.mean(Outer[inicio:fin]) #Media
    Po[i-1,5]=np.var(Outer[inicio:fin]) #var
    Po[i-1,6]=skew(Outer[inicio:fin])[0] #asimetria
    Po[i-1,7]=kurtosis(Outer[inicio:fin])[0] #curtosis

    Pi[i-1,0]=sqrt(mean(square(Inner[inicio:fin]))) #RMS
    Pi[i-1,1]=np.amax(Inner[inicio:fin]) #Peak
    Pi[i-1,2]=np.amax(Inner[inicio:fin])-np.amin(Inner[inicio:fin]) #peak-peak
    Pi[i-1,3]=Pi[i-1,1]/Pi[i-1,0] #crest
    Pi[i-1,4]=np.mean(Inner[inicio:fin]) #Media
    Pi[i-1,5]=np.var(Inner[inicio:fin]) #var
    Pi[i-1,6]=skew(Inner[inicio:fin])[0] #asimetria
    Pi[i-1,7]=kurtosis(Inner[inicio:fin])[0] #curtosis

#guardar parámetros
np.save('Pn.npy', Pn)
np.save('Po.npy', Po)
np.save('Pi.npy', Pi)
```


Ejemplo 2

```
#graficar datos
f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex='col',figsize=(9,6))
ax1.plot(Pn[:,0])
ax1.plot(Po[:,0])
ax1.plot(Pi[:,0])
ax1.legend(['Normal','Pista externa','Pista interna'])
ax1.set_title('RMS', fontsize=14)

ax2.plot(Pn[:,1])
ax2.plot(Po[:,1])
ax2.plot(Pi[:,1])
ax2.legend(['Normal','Pista externa','Pista interna'])
ax2.set_title('Valor peak', fontsize=14)

ax3.plot(Pn[:,2])
ax3.plot(Po[:,2])
ax3.plot(Pi[:,2])
ax3.legend(['Normal','Pista externa','Pista interna'])
ax3.set(xlabel='Segmento')
ax3.set_title('Valor peak-peak', fontsize=14)

ax4.plot(Pn[:,3])
ax4.plot(Po[:,3])
ax4.plot(Pi[:,3])
ax4.legend(['Normal','Pista externa','Pista interna'])
ax4.set(xlabel='Segmento')
ax4.set_title('Factor de cresta', fontsize=14)
```

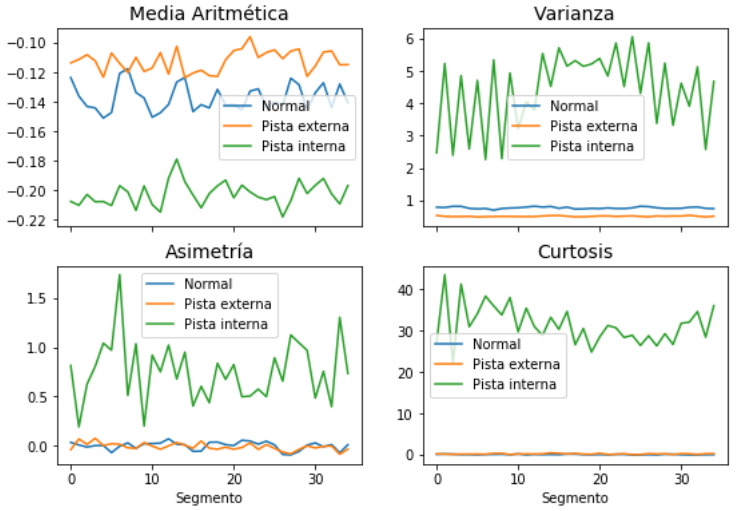
Ejemplo 2



Ejemplo 2

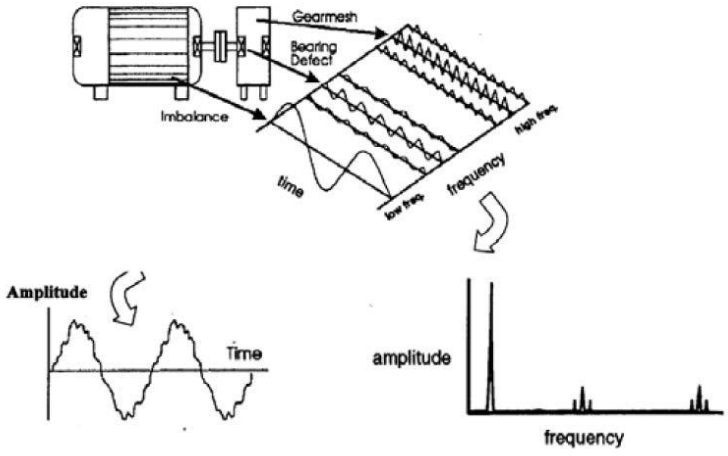
```
f, ((ax5, ax6), (ax7, ax8)) = plt.subplots(2, 2, sharex='col',figsize=(9,6))
ax5.plot(Pn[:,4])
ax5.plot(Po[:,4])
ax5.plot(Pi[:,4])
ax5.legend(['Normal','Pista externa','Pista interna'])
ax5.set_title('Media Aritmética', fontsize=14)

ax6.plot(Pn[:,5])
ax6.plot(Po[:,5])
ax6.plot(Pi[:,5])
ax6.legend(['Normal','Pista externa','Pista interna'])
ax6.set_title('Varianza', fontsize=14)

ax7.plot(Pn[:,6])
ax7.plot(Po[:,6])
ax7.plot(Pi[:,6])
ax7.legend(['Normal','Pista externa','Pista interna'])
ax7.set(xlabel='Segmento')
ax7.set_title('Asimetría', fontsize=14)

ax8.plot(Pn[:,7])
ax8.plot(Po[:,7])
ax8.plot(Pi[:,7])
ax8.legend(['Normal','Pista externa','Pista interna'])
ax8.set(xlabel='Segmento')
ax8.set_title('Curtosis', fontsize=14)
```

Ejemplo 2

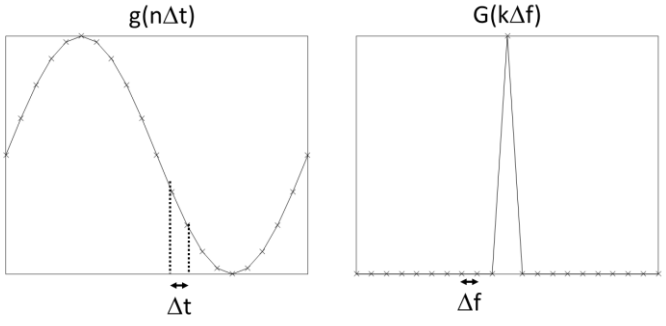


Parámetros en el dominio de frecuencias

Dominio en frecuencias



Transformada discreta de Fourier



$$g(n\Delta t) = \frac{1}{f_s}\sum_{k=0}^{N_s-1} G(k\Delta f)e^{2\pi jnk/N_s}$$

$$G(n\Delta f) = \frac{1}{N_s}\sum_{k=0}^{N_s-1} g(k\Delta t)e^{-2\pi jnk/N_s}$$

Con N_s: número de datos, T= $N_s\Delta t$ y $f_s = N_s\Delta f$

Transformada discreta de Fourier

La evaluación directa de la transformada discreta de Fourier requiere N^2 operaciones. Con la transformada rápida de Fourier (FFT) se reduce el número de operaciones a $N_s \log_2 N_s$. La transformada rápida de Fourier es el núcleo de todos los procesadores de señal modernos.

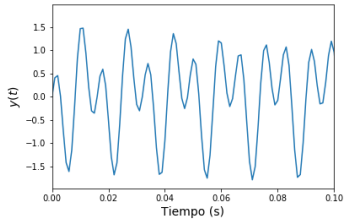
Transformada discreta de Fourier

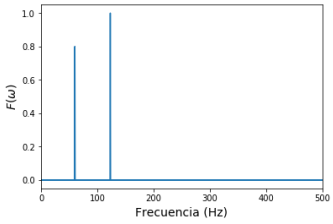
- **N_s**: Numero de muestras en el periodo T.
- **f_s**: Frecuencia de muestreo, frecuencia a la cual se adquieren y digitalizan los datos.
- Δt : Intervalo de muestreo: intervalo de tiempo al cual la señal es muestreada. $\Delta t = 1/f_s$.
- T: Periodo de tiempo en donde se adquieren los datos. $T = N_s/f_s$.
- Δf : Espaciado de frecuencias en el espectro (resolución en frecuencias). $\Delta f = f_s/N_s = 1/T$.
- f_{max}: frecuencia mayor contenida o permitida en la señal temporal. Según el teorema de Shannon: $f_{max} \leq f_s/2$.

Ejemplo 3

```
from scipy.fftpack import fft, fftfreq
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi

N = 2000  # Número de datos
fs= 1000  # Frecuencia de muestreo/adquisición
dt = 1/fs # Espaciado, 16 puntos por período
t = np.linspace(0, (N-1)*dt, N)  # Intervalo de tiempo en segundos
y = np.sin(2*pi *123*t) - 0.8*np.sin(2*pi*60*t)  # Señal

#Transformada de Fourier
Y = fft(y)[0:int(N/2)]/(N/2)
frq = fftfreq(N,dt)[0:int(N/2)] #vector de frecuencias

fig,ax = plt.subplots(1)
ax.plot(t, y)
plt.xlabel('Tiempo (s)', fontsize=14)
plt.ylabel('$y(t)$', fontsize=14)
plt.xlim(0,0.1)
plt.show()

fig,ax = plt.subplots(1)
plt.plot(frq,abs(Y))
plt.xlabel('Frecuencia (Hz)', fontsize=14)
plt.ylabel('$F(\omega)$', fontsize=14)
plt.show()
```


Ejemplo 2



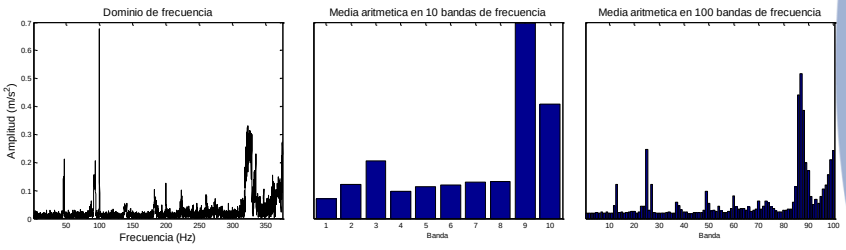


Extracción de parámetros

$F(\omega) = \{F_1 \quad F_1 \quad \cdots \quad F_N\}$ es la transformada de Fourier de la señal *y(t)*. La **media aritmética** del espectro de frecuencias divida en bandas de frecuencias (usualmente octavos) se define como:

$$B_i = \frac{1}{N_{i+1} - N_i} \sum_{j=N_i}^{N_{i+1}-1} \text{abs}(F_i)$$

donde N_i y N_{i+1} son el primer y último componente de la banda i.



Ejemplo 4

```
#importar librerias
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.fftpack import fft, fftfreq
from numpy import mean, sqrt, square

#Leer datos
Datos0=sio.loadmat('normal.mat')
Datos1=sio.loadmat('outer.mat')
Datos2=sio.loadmat('inner.mat')
Normal=Datos0['normal']
Outer=Datos1['outer']
Inner=Datos2['inner']

#Datos
Fs=48828 #sampling rate
dt=1/Fs #paso de tiempo
N=len(Normal) #numero de datos

#Dividir por segmentos
L=5000 #largo de los segmentos
l=1000 #overlap
Nt=math.floor((N-l)/(L-l)) #total de segmentos
nb=8 #numero de bandas
```

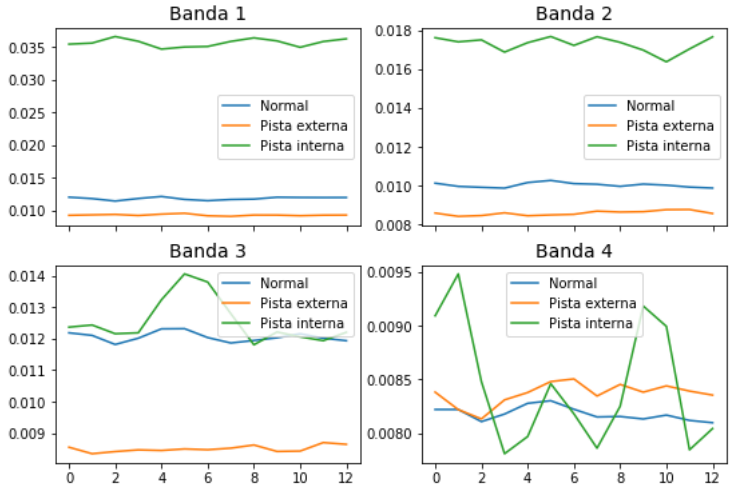
Ejemplo 4

```
En=np.zeros((Nt,nb))
Eo=np.zeros((Nt,nb))
Ei=np.zeros((Nt,nb))

for i in range(1,Nt+1):
   inicio=(i-1)*L-(i-1)*l+1
   fin=i*L-(i-1)*l

   Fn = fft(Normal[inicio:fin,0])[0:int(L/2)]/(L/2)
   Fo = fft(Outer[inicio:fin,0])[0:int(L/2)]/(L/2)
   Fi = fft(Inner[inicio:fin,0])[0:int(L/2)]/(L/2)
   frq = fftfreq(L,dt)[0:int(L/2)]

   Lb=int(L/2/nb)
   for k in range(1,nb+1):
      inicio=Lb*(k-1)+1
      fin=k*Lb
      En[i-1][k-1]=mean(abs(Fn[inicio:fin]))
      Eo[i-1][k-1]=mean(abs(Fo[inicio:fin]))
      Ei[i-1][k-1]=mean(abs(Fi[inicio:fin]))

#guardar parámetros
np.save('En.npy', En)
np.save('Eo.npy', Eo)
np.save('Ei.npy', Ei)
```


Ejemplo 4

```
f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex='col',figsize=(9,6))

ax1.plot(En[:,0])
ax1.plot(Eo[:,0])
ax1.plot(Ei[:,0])
ax1.legend(['Normal','Pista externa','Pista interna'])
ax1.set_title('Banda 1', fontsize=14)

ax2.plot(En[:,1])
ax2.plot(Eo[:,1])
ax2.plot(Ei[:,1])
ax2.legend(['Normal','Pista externa','Pista interna'])
ax2.set_title('Banda 2', fontsize=14)

ax3.plot(En[:,2])
ax3.plot(Eo[:,2])
ax3.plot(Ei[:,2])
ax3.legend(['Normal','Pista externa','Pista interna'])
ax3.set_title('Banda 3', fontsize=14)

ax4.plot(En[:,3])
ax4.plot(Eo[:,3])
ax4.plot(Ei[:,3])
ax4.legend(['Normal','Pista externa','Pista interna'])
ax4.set_title('Banda 4', fontsize=14)
```

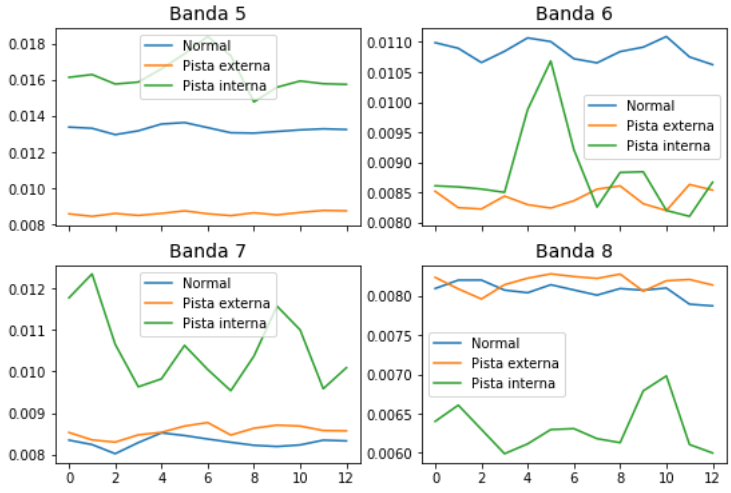
Ejemplo 4



Ejemplo 4

```
f, ((ax5, ax6), (ax7, ax8)) = plt.subplots(2, 2, sharex='col',figsize=(9,6))
ax5.plot(En[:,4])
ax5.plot(Eo[:,4])
ax5.plot(Ei[:,4])
ax5.legend(['Normal','Pista externa','Pista interna'])
ax5.set_title('Banda 5', fontsize=14)

ax6.plot(En[:,5])
ax6.plot(Eo[:,5])
ax6.plot(Ei[:,5])
ax6.legend(['Normal','Pista externa','Pista interna'])
ax6.set_title('Banda 6', fontsize=14)

ax7.plot(En[:,6])
ax7.plot(Eo[:,6])
ax7.plot(Ei[:,6])
ax7.legend(['Normal','Pista externa','Pista interna'])
ax7.set_title('Banda 7', fontsize=14)

ax8.plot(En[:,7])
ax8.plot(Eo[:,7])
ax8.plot(Ei[:,7])
ax8.legend(['Normal','Pista externa','Pista interna'])
ax8.set_title('Banda 8', fontsize=14)
```

Ejemplo 4



Parámetros en el dominio tiempo-frecuencia

Métodos tiempo-frecuencia

- Transformada corta de Fourier (espectrograma)
- Transforma de Wavelet (escalograma)

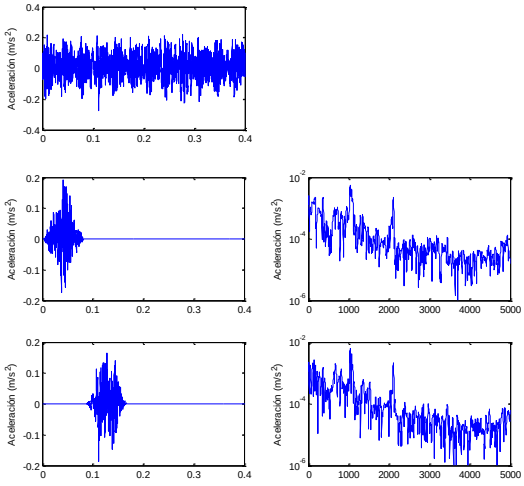
Extracción de parámetros
Transformada corta de Fourier

La transformada corta de Fourier se utiliza para señales no estacionarias y consiste en aplicar la transformada de Fourier a una ventana de datos “deslizante”.

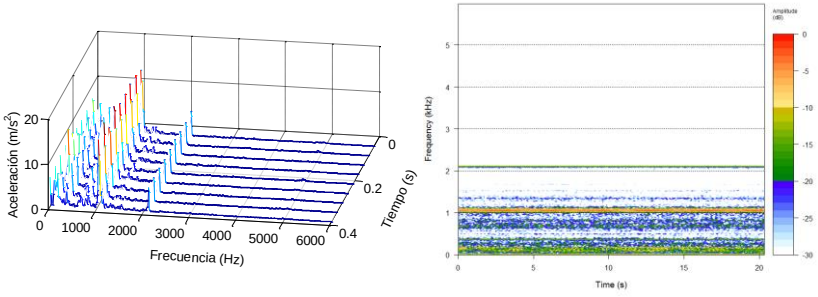
El resultado es una representación en el tiempo del espectro en frecuencia de la señal.

Es importante la selección del ancho de la ventana, ya que define la resolución en el tiempo y en frecuencia.

Extracción de parámetros
Transformada corta de Fourier



Extracción de parámetros
Transformada corta de Fourier

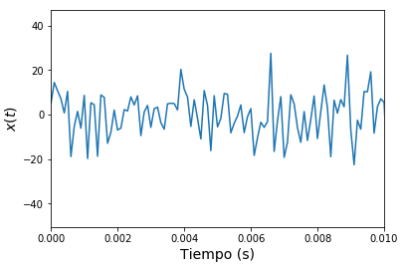


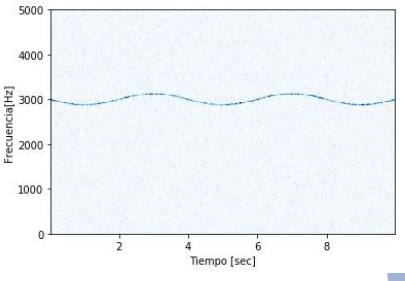
La segunda imagen se conoce como espectrograma

Ejemplo 5

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi

fs = 10e3 #frecuencia de muestreo Hz
N = 100000 #numero de datos
dt = 1/fs # Espaciado temporal
t = np.linspace(0, (N-1)*dt, N)  # Intervalo de tiempo en segundos
mod = 500*np.cos(2*np.pi*0.25*t)
carrier = 5 * np.sin(2*np.pi*3e3*t + mod)
mu, sigma = 0, 10
noise = np.random.normal(mu,sigma,N)
x = carrier + noise

plt.plot(t, x)
plt.xlabel('Tiempo (s)', fontsize=14)
plt.ylabel('$x(t)$', fontsize=14)
plt.xlim(0,0.01)
plt.show()

f, t, Sxx = signal.spectrogram(x, fs,window='hann',nperseg=500,noverlap=250)
plt.figure()
plt.pcolormesh(t, f, Sxx,cmap='Blues')
plt.ylabel('Frecuencia[Hz]')
plt.xlabel('Tiempo [sec]')
plt.show()
```


Ejemplo 5





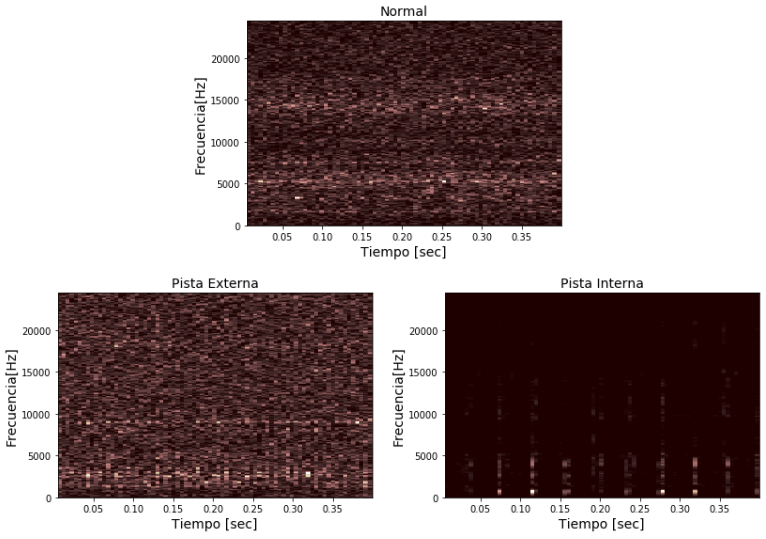
Ejemplo 6

```
#importar librerías
import scipy.io as sio
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi
import math
from scipy.fftpack import fft, fftfreq

#Leer datos
Datos0=sio.loadmat('normal.mat')
Datos1=sio.loadmat('outer.mat')
Datos2=sio.loadmat('inner.mat')
Normal=Datos0['normal']
Outer=Datos1['outer']
Inner=Datos2['inner']

#Datos
Fs=48828 #sampling rate
Nd=20000
```


Ejemplo 6

```
x1=Normal[1:Nd,0]
f, t, S1 = signal.spectrogram(x1,Fs,window='hann',nperseg=500,noverlap=250)

x2=Outer[1:Nd,0]
f, t, S2 = signal.spectrogram(x2,Fs,window='hann',nperseg=500,noverlap=250)

x3=Inner[1:Nd,0]
f, t, S3 = signal.spectrogram(x3,Fs,window='hann',nperseg=500,noverlap=250)

plt.pcolormesh(t, f, S1,cmap='pink')
plt.ylabel('Frecuencia[Hz]', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Normal', fontsize=14)
plt.show()

plt.pcolormesh(t, f, S2,cmap='pink')
plt.ylabel('Frecuencia[Hz]', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Pista Externa', fontsize=14)
plt.show()

plt.pcolormesh(t, f, S3,cmap='pink')
plt.ylabel('Frecuencia[Hz]', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Pista Interna', fontsize=14)
plt.show()
```

Ejemplo 6



Extracción de parámetros Wavelets

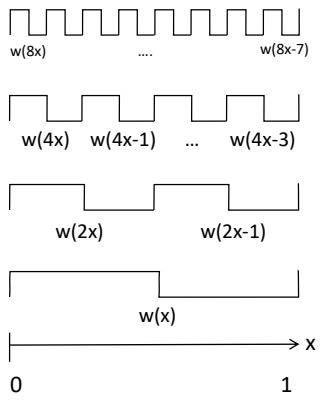
A diferencia de la transformada de Fourier donde se ocupan funciones base estacionarias (exponencial compleja), la transformada de Wavelet utiliza funciones base no estacionarias. Estas funciones se denominan “wavelets” y pueden dilatarse y desfasarse en función del tiempo.

La principal ventaja es que la resolución en tiempo y frecuencia son independientes. Por lo tanto, se puede estudiar el contenido en frecuencia de la señal sin perder información en el dominio temporal.

Extracción de parámetros Wavelets

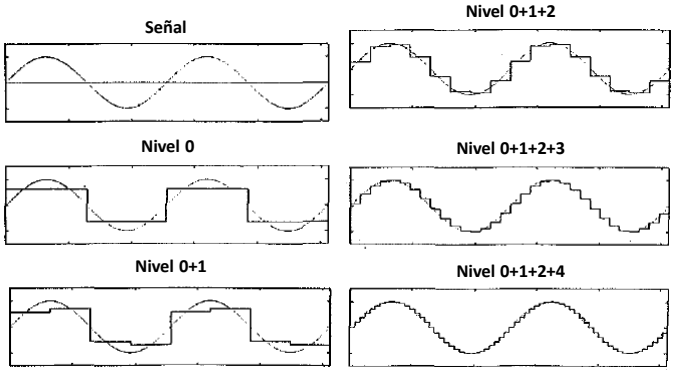
- Familia de wavelets haar

$$f(x) = a_0 + a_1 w(x) + a_2 w(2x) + a_3 w(2x-1) + a_4 w(4x) + \ldots a_9 w(8x-7)$$



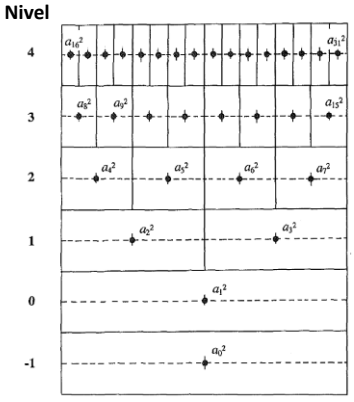
Extracción de parámetros Wavelets

- Ejemplo con wavelets haar



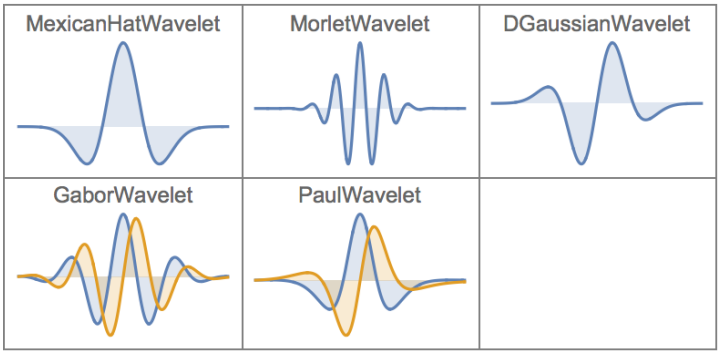
Extracción de parámetros Wavelets

- Mapa de wavelets

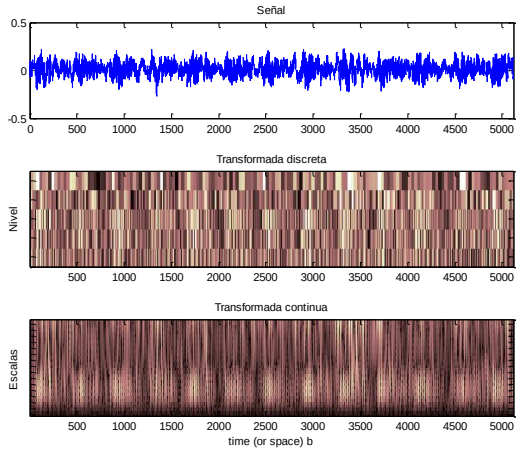


Extracción de parámetros Wavelets

- Tipos de wavelets



Extracción de parámetros Wavelets



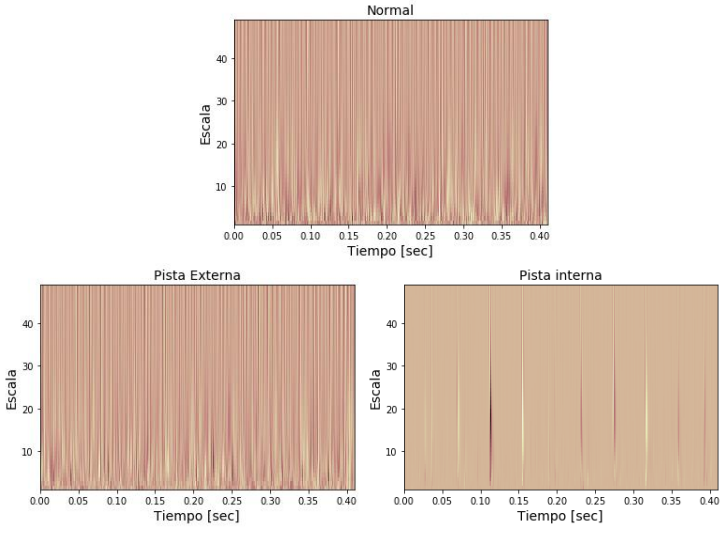
Ejemplo 7

```
#importar librerías
import scipy.io as sio
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi
import math
from scipy.fftpack import fft, fftfreq

#Leer datos
Datos0=sio.loadmat('normal.mat')
Datos1=sio.loadmat('outer.mat')
Datos2=sio.loadmat('inner.mat')
Normal=Datos0['normal']
Outer=Datos1['outer']
Inner=Datos2['inner']

#Datos
Fs=48828 #sampling rate
dt=1/Fs
Nd=20000

scales=np.arange(1, 50)
```

Ejemplo 7

```
x1=Normal[1:Nd,0]
x2=Outer[1:Nd,0]
x3=Inner[1:Nd,0]
t = np.linspace(0, (Nd-1)*dt, Nd)  # Intervalo de tiempo en segundos

cwt1 = signal.cwt(x1, signal.ricker,scales)
cwt2 = signal.cwt(x2, signal.ricker,scales)
cwt3 = signal.cwt(x3, signal.ricker,scales)

plt.pcolormesh(t,scales,cwt1,cmap='pink')
plt.ylabel('Escala', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Normal', fontsize=14)
plt.show()

plt.pcolormesh(t,scales,cwt2,cmap='pink')
plt.ylabel('Escala', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Pista Externa', fontsize=14)
plt.show()

plt.pcolormesh(t,scales,cwt3,cmap='pink')
plt.ylabel('Escala', fontsize=14)
plt.xlabel('Tiempo [sec]', fontsize=14)
plt.title('Pista interna', fontsize=14)
plt.show()
```

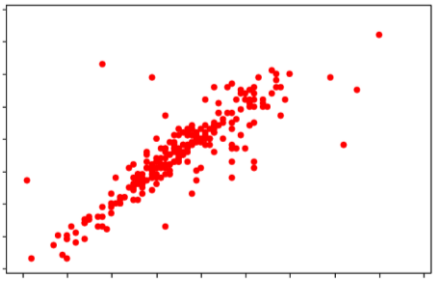

Ejemplo 6



Métodos de reducción de dimensionalidad

Introducción

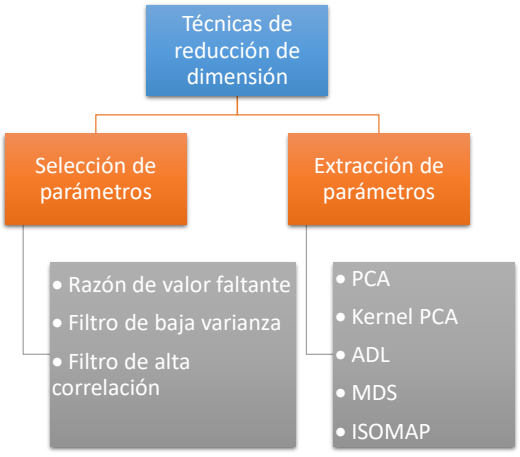
Una de las formas más comunes de hacer visualización es a través de gráficos. Sin embargo, a medida que aumenta el numero de variables en un set de datos, visualizarlas y hacer inferencias se hace cada vez más complicado.



¿Porque reducir la dimensión de los datos?

- El espacio requerido para almacenar los datos se reduce a medida que disminuye el número de dimensiones.
- Una menor dimensión resulta en un tiempo menor de entrenamiento (menor complejidad del algoritmo).
- Algunos algoritmos no funcionan bien con dimensiones grandes.
- Se elimina redundancia en los datos.
- Ayuda en la visualización de datos. Reducir nuestro espacio a 2D o 3D puede permitirnos trazar y observar patrones más claramente.

Técnicas de reducción de dimensionalidad



Selección de parámetros

Razón de valor faltante

ID	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	count
AB101	1.0	0.0	0.0	1.0	9.84	14.395	81.0	NaN	16
AB102	1.0	NaN	0.0	NaN	9.02	13.635	80.0	NaN	40
AB103	1.0	0.0	NaN	1.0	9.02	13.635	80.0	NaN	32
AB104	NaN	0.0	NaN	1.0	9.84	14.395	75.0	NaN	13
AB105	1.0	NaN	0.0	NaN	9.84	14.395	NaN	16.9979	1
AB106	1.0	0.0	NaN	2.0	9.84	12.880	75.0	NaN	1
AB107	1.0	0.0	0.0	1.0	9.02	13.635	80.0	NaN	2
AB108	1.0	NaN	0.0	1.0	8.20	12.880	86.0	NaN	3
AB109	NaN	0.0	0.0	NaN	9.84	14.395	NaN	NaN	8
AB110	1.0	0.0	0.0	1.0	13.12	17.425	76.0	NaN	14

Razón de valor faltante

$$\text{Razón de valor faltante} = \frac{\text{Número de valores faltantes}}{\text{Número total de valores}}$$

Variable	Razón de valor faltante (%)
ID	0
season	20
holyday	30
workingday	30
weather	30
temp	0
atemp	0
humidity	20
windspeed	90

Filtro de baja varianza

ID	season	holiday	workingday	weather	f5	temp	atemp	humidity	windspeed	count
AB101	1	0	0	1	7	9.84	14.395	81	0.0000	16
AB102	1	0	0	1	7	9.02	13.635	80	0.0000	40
AB103	1	0	0	1	7	9.02	13.635	80	0.0000	32
AB104	1	0	0	1	7	9.84	14.395	75	0.0000	13
AB105	1	0	0	1	7	9.84	14.395	75	0.0000	1
AB106	1	0	0	2	7	9.84	12.880	75	6.0032	1
AB107	1	0	0	1	7	9.02	13.635	80	0.0000	2
AB108	1	0	0	1	7	8.20	12.880	86	0.0000	3
AB109	1	0	0	1	7	9.84	14.395	75	0.0000	8
AB110	1	0	0	1	7	13.12	17.425	76	0.0000	14

Filtro de baja varianza

Los datos se deben normalizar primero:

- Estandarización de los datos:

$$x' = \frac{x - \bar{x}}{\sigma}$$

La estandarización no sirve para este caso

- Normalización max-min

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Normalización media

$$x' = \frac{x - \bar{x}}{\max(x) - \min(x)}$$

Filtro de baja varianza

Varianza → dispersión en los datos

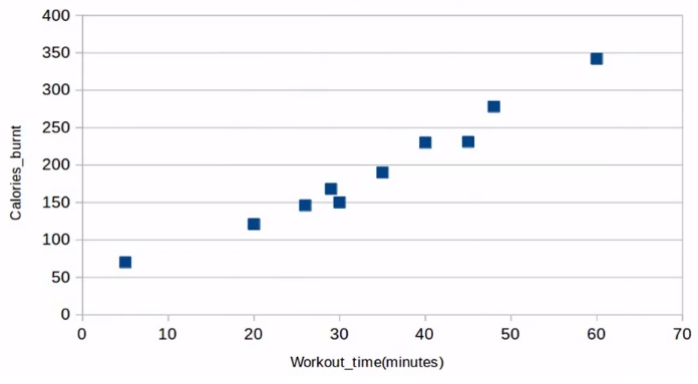
$$\sigma^2 = \frac{\sum(x - \bar{x})^2}{n}$$

Se eliminan las variables cuya varianza sea menor a un valor previamente establecido.

Filtro de alta correlación

ID	Workout_time(minutes)	Calories_burnt	Gender	Plays_Sport?	Fitness_Level
1	20	121	M	Yes	Fit
2	40	230	M	No	Fit
3	60	342	F	No	Unfit
4	5	70	M	Yes	Fit
5	48	278	F	Yes	Unfit
6	26	146	M	Yes	Fit
7	29	168	F	No	Unfit
8	45	231	F	Yes	Fit
9	30	150	M	No	Fit
10	35	190	F	No	Fit

Filtro de alta correlación



Coeficiente de correlación de Pearson= 0.9819

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una medida de dependencia lineal entre dos variables aleatorias.

Donde:

$$\rho_{X,Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y}$$

$\sigma_{X,Y}$ es la covarianza de (X,Y)

σ_X es la desviación estándar de la variable X

σ_y es la desviación estándar de la variable Y

Coeficiente de correlación de Pearson

De manera análoga podemos calcular este coeficiente sobre un estadístico muestral, denotado como $r_{X,Y}$:

$$r_{X,Y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}}$$

- Toma valores entre -1 y 1
- Magnitudes más altas implican una mayor relación

Filtro de alta correlación

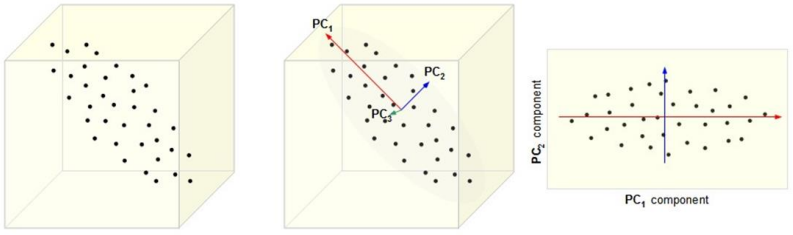
- Determinar la correlación entre todas las variables independientes
- Eliminar una de las variables si la magnitud de la correlación supera un cierto límite (usualmente ~ 0.6). Se puede eliminar la variable que tenga una menor correlación con el valor objetivo.

Reducción de parámetros

Análisis de Componentes Principales (PCA)

PCA es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos.

Busca la proyección según la cual un conjunto de variables posiblemente correlacionadas se conviertan en un conjunto de variables sin correlación llamadas **componentes principales**.



Análisis de Componentes Principales (PCA)

Consideremos un set de d parámetros representados en el vector $\boldsymbol{x}$. La proyección de este vector en la dirección de $\boldsymbol{w}$ es:

$$z = \boldsymbol{w}^T \boldsymbol{x}$$

PCA busca maximizar la varianza en las variables proyectadas. Si consideramos el primer componente $\boldsymbol{w}_1$, entonces:

$$\mathrm{Var}(z_1) = \boldsymbol{w}_1^T \sum \boldsymbol{w}_1$$

Donde $\sum = \mathrm{Cov}(\boldsymbol{x}) = \boldsymbol{X}^T \boldsymbol{X}$. Se busca $\boldsymbol{w}_1$ que maximice $\mathrm{Var}(z_1)$ sujeto a $\boldsymbol{w}_1^T \boldsymbol{w}_1 = 1$.

Análisis de Componentes Principales (PCA)

Ocupando el método de multiplicadores de Lagrange:

$$\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^T \sum \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^T \boldsymbol{w}_1 - 1)$$

Derivando e igualando a cero se obtiene:

$$\sum \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$$

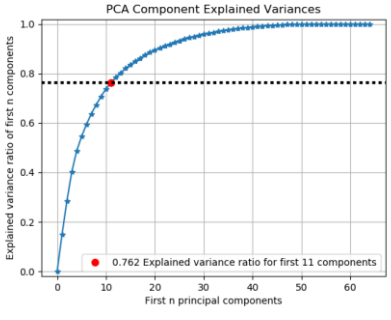
Esto implica que $\boldsymbol{w}_1$ es un vector propio de $\sum$ y α es un valor propio. Dado que se busca maximizar:

$$\boldsymbol{w}_1^T \sum \boldsymbol{w}_1 = \boldsymbol{\alpha} \boldsymbol{w}_1^T \boldsymbol{w}_1 = \boldsymbol{\alpha}$$

Se selecciona el vector propio con el mayor valor propio asociado.

Análisis de Componentes Principales (PCA)

De forma similar se puede demostrar que los siguientes vectores $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, … corresponden a los vectores propios de la matriz $\sum$, ordenando los valores propios de mayor a menor. Los *k* valores propios mayores a cero corresponden a la dimensión del nuevo espacio.



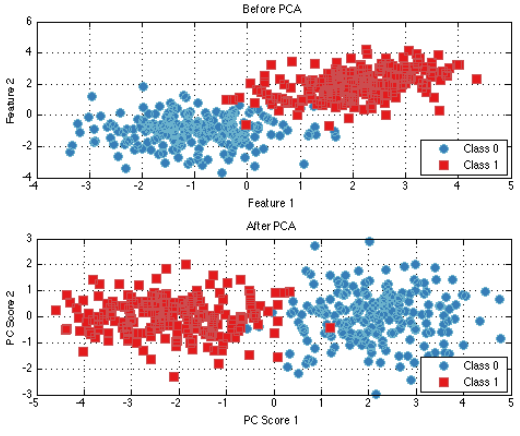
Análisis de Componentes Principales (PCA)

- Calcular la matriz de covarianza, $\sum$.
- Calcular los vectores y valores de $\sum$ y ordenarlos de mayor a menor.
- Seleccionar los p mayores y construir la matriz de proyección $\boldsymbol{w}$ con los vectores seleccionados.
- Determinar las nuevas variables como:

$$z = \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu})$$

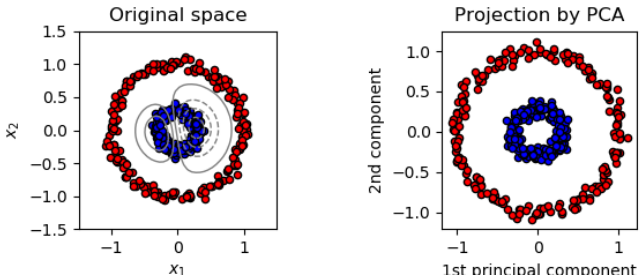
Notar que se debe sustraer la media antes de proyectar, se recomienda normalizar $\boldsymbol{x}$ antes de aplicar PCA.

Análisis de Componentes Principales (PCA)



Kernel PCA

PCA es una técnica de reducción de dimensionalidad lineal que facilita la compresión de datos de alta dimensión. Sin embargo, esta técnica no puede manejar datos con relaciones no lineales.

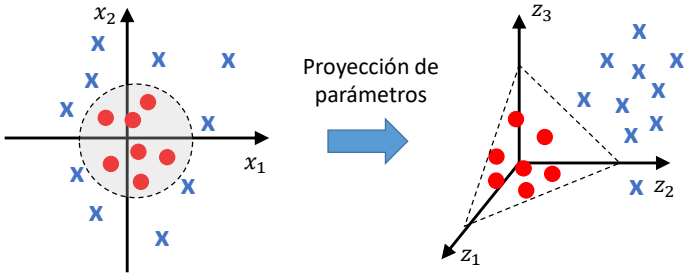


Kernel PCA

Se busca proyectar los datos en un nuevo espacio de mayor dimensión (usualmente) donde se encuentren patrones lineales.

$$\phi(x)\colon \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \to (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



Kernel PCA

El truco del kernel: se busca una transformación no lineal

$$\phi(x)$$

a un espacio de mayor dimensionalidad provisto de un producto escalar que puede ser expresado como (kernel):

$$\phi(x_i)^T\phi(x_j) = \kappa(x_i, x_j)$$

Kernel PCA

$$\begin{aligned}\phi(\boldsymbol{x})^T\phi(\boldsymbol{z}) &= \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)^T\left(z_1^2, \sqrt{2}z_1z_2, z_2^2\right)\\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2\\ &= (x_1z_1 + x_2z_2)^2\\ &= (\boldsymbol{x}^T\boldsymbol{z})^2\\ &= \kappa(\boldsymbol{x}, \boldsymbol{z})\end{aligned}$$

Se podría realizar PCA en el espacio proyectado: calcular la matriz de covarianza de los datos proyectados y calcular sus vectores propios.

Pero, se puede ocupar la matriz de kernel:

$$\mathrm{K_{ij}} = \kappa(\mathbf{z}_\mathrm{i}, \mathbf{z}_\mathrm{j}) = \phi(\mathbf{z}_\mathrm{i})^\mathrm{T}\phi(\mathbf{z}_\mathrm{j}).$$

Esto es más eficiente!

Kernel PCA

Algunas funciones típicas son:

- Polinomial:

$$K(x, y) = (1 + x^T y)^d$$

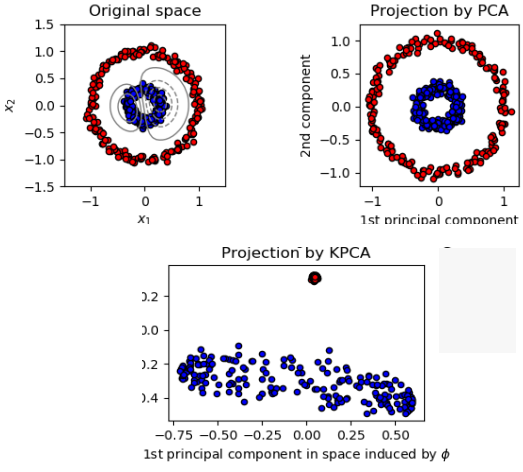
- Radial Basis Function (RBF):

$$K(x, y) = \exp(^{-(x-y)^T(x-y)}/_{2\sigma^2})$$

- Sigmoid:

$$K(x, y) = \tanh(\rho_1 x^T y + \rho_2)$$

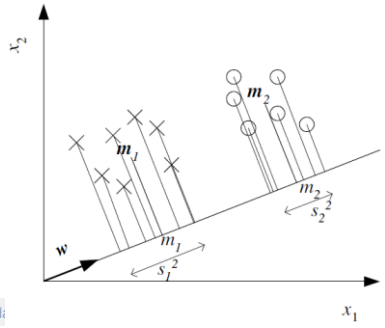
Kernel PCA



Análisis Discriminante Lineal (ADL)

ADL es un método **supervisado** para reducción de dimensión en **problemas de clasificación**.

Consideremos el caso dos clases C_1 y C_2, queremos encontrar una dirección definida por el vector $\boldsymbol{w}$ de manera que al proyectar los datos sobre $\boldsymbol{w}$ las clases estén lo más separas posibles.



Análisis Discriminante Lineal (ADL)

La proyección de los datos en la dirección de $\boldsymbol{w}$ viene dado por:

$$z = \boldsymbol{w}^T \boldsymbol{x}$$

Definimos $\boldsymbol{m}_i$ y m_i como el valor medio de los datos de la clase C_i antes y después de la proyección, respectivamente.

Consideremos la variable r^t cuyo valor es igual a 1 si $\boldsymbol{x}^t \in C_1$ y es igual a 0 si $\boldsymbol{x}^t \in C_2$. Por lo tanto,

$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} = \boldsymbol{w}^T \boldsymbol{m}_1$$

$$m_2 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t (1 - r^t)}{\sum_t (1 - r^t)} = \boldsymbol{w}^T \boldsymbol{m}_2$$

Análisis Discriminante Lineal (ADL)

La varianza de los datos en las clases C_1 y C_2 luego de la proyección vienen dados por:

$$s_1^2 = \sum_t (\boldsymbol{w}^T \boldsymbol{x}^t - m_1)^2 r^t$$
$$s_2^2 = \sum_t (\boldsymbol{w}^T \boldsymbol{x}^t - m_2)^2 (1 - r^t)$$

Luego de la proyección se espera que las clases estén lo más separadas posibles y que la varianza en los grupos sea pequeña. Por lo tanto se espera que $|m_1 - m_2|$ sea grande y que $s_1^2 + s_2^2$ sea pequeño.

Análisis Discriminante Lineal (ADL)

Por lo tanto, el problema consiste en encontrar $\boldsymbol{w}$ que maximize:

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$
$$= \frac{|\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)|^2}{\boldsymbol{w}^T S_w \boldsymbol{w}}$$

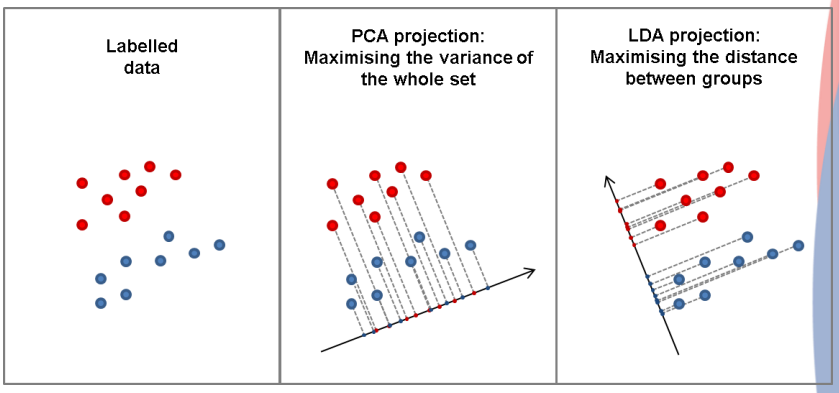
Con $S_w = \sum_t r^t (\boldsymbol{x}^t - \boldsymbol{m}_1)(\boldsymbol{x}^t - \boldsymbol{m}_1)^T + \sum_t (1 - r^t)(\boldsymbol{x}^t - \boldsymbol{m}_2)(\boldsymbol{x}^t - \boldsymbol{m}_2)^T$

Derivando con respecto a $\boldsymbol{w}$ e igualando a cero se obtiene que:

$$\boldsymbol{w} = c S_w^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

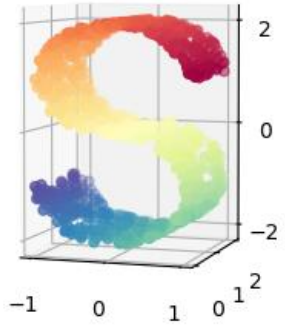
Con c igual a una constante, dado que lo relevante es la dirección de $\boldsymbol{w}$, se puede imponer c=1.

Análisis Discriminante Lineal (ADL)



Manifold learning

Manifold: agrupación de puntos que forman un cierto tipo de conjunto, como los de una superficie topológicamente cerrada o un análogo en tres o más dimensiones.



- Numero de conexiones
- Distancia entre puntos
- Propiedades locales vs globales?

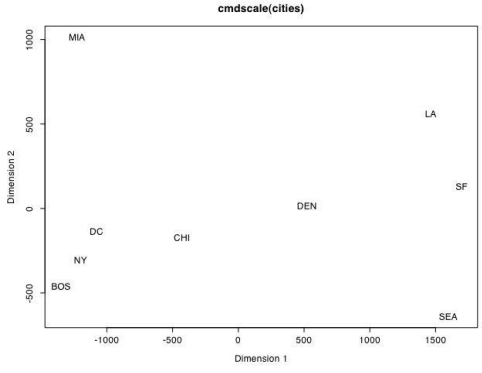
Escalamiento Multidimensional (MDS)

- Supongamos que tenemos una lista de ciudades Chicago, DC, Denver, etc., en conjunto con las distancia entre las ciudades.

	BOS	CHI	DC	DEN	LA	MIA	NY	SEA	SF
BOS	0	963	429	1949	2979	1504	206	2976	3095
CHI	963	0	671	996	2054	1329	802	2013	2142
DC	429	671	0	1616	2631	1075	233	2684	2799
DEN	1949	996	1616	0	1059	2037	1771	1307	1235
LA	2979	2054	2631	1059	0	2687	2786	1131	379
MIA	1504	1329	1075	2037	2687	0	1308	3273	3053
NY	206	802	233	1771	2786	1308	0	2815	2934
SEA	2976	2013	2684	1307	1131	3273	2815	0	808
SF	3095	2142	2799	1235	379	3053	2934	808	0

Escalamiento Multidimensional (MDS)

¿Podemos ubicar estas ciudades en un espacio 2D de manera que se mantengan las distancias entre ciudades?

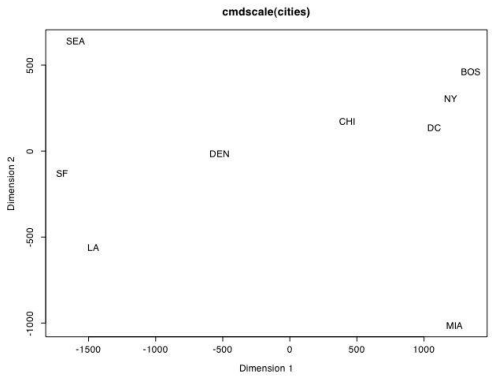


- La matriz de distancias se reproduce perfectamente
- La matriz de distancias no incluye información sobre origen y orientación.

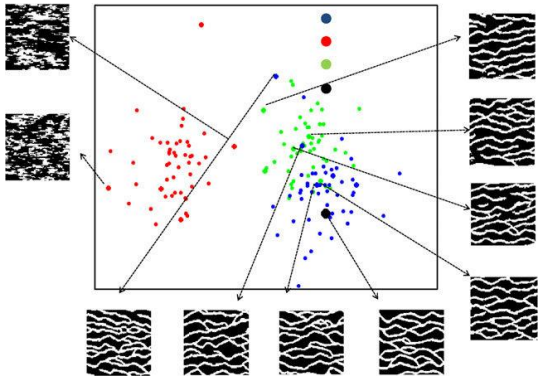
Escalamiento Multidimensional (MDS)

Podemos rotar la solución, manteniendo las distancias entre pares.

.



Escalamiento Multidimensional (MDS)



Es un método popular para visualización de datos

Escalamiento Multidimensional (MDS)

MDS busca minimizar la distancia euclidiana entre los pares de datos en el espacio original y los datos proyectados:

$$\min\left(\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\left(\|z_i - z_j\| - \delta_{i,j}\right)^2\right)^{1/2}$$

Donde $\|z_i - z_j\|$ es la distancia entre los puntos *i*, *j* en el espacio proyectado y $\delta_{i,j}$ la misma distancia en el espacio original.

Escalamiento Multidimensional (MDS)

- Calcular la matriz de distancias al cuadrado $D^2 = \left[\delta_{i,j}^2\right]$
- Aplicar doble centrado: $B = JD^2J$, donde J es una matriz de centrado.
- Calcular los valores propios, $\lambda_1, \lambda_2, \ldots, \lambda_p$ y vectores propios, $v_1, v_2, \ldots, v_p$
- Calcular los parámetros proyectados:

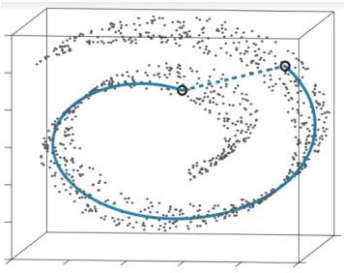
$$\boldsymbol{Z} = \boldsymbol{V}_p\sqrt{\boldsymbol{\Lambda}_p}$$

Donde $\boldsymbol{V}_p$es una matriz de vectores propios y $\boldsymbol{\Lambda}_p$una matriz diagonal con valores propios.

ISOMAP

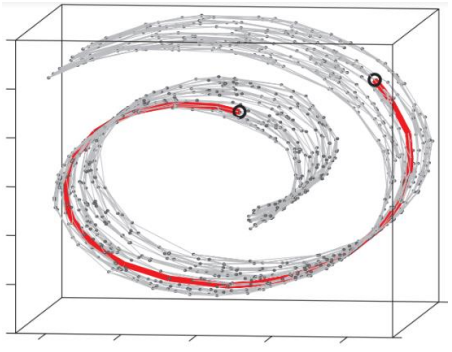
Si los datos se encuentran en una curva, la distancia euclediana puede interpretar que dos puntos están cercanos, mientras que su distancia en el manifold es mayor.

Isomap considera la distancia geodésica, que es la distancia entre un par de puntos medidos sobre el manifold.



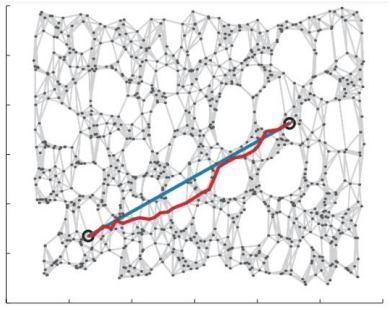
ISOMAP

Primero, se construye un gráfico de vecindad G, donde cada punto está conectado con sus vecinos más cercanos.

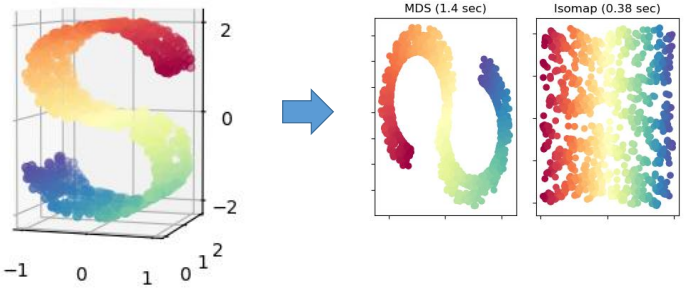


ISOMAP

Luego, todos los caminos más cortos en G se calculan usando una estimación de la distancia geodésica, produciendo una matriz de distancia $\widetilde{D}$. Los datos se proyectan siguiendo el mismo procedimiento que en MDS.



Manifold Learning



Ejemplo 8

En los ejemplos 2 y 4, se guardaron los parámetros extraídos en el dominio del tiempo y frecuencia en los archivos:

- Pn.npy: Parámetros temporales caso normal
- Po.npy: Parámetros temporales caso falla en pista externa
- Pi.npy: Parámetros temporales caso falla en pista interna
- En.npy: Parámetros en frecuencia caso normal
- Eo.npy: Parámetros en frecuencia caso falla en pista externa
- Ei.npy: Parámetros en frecuencia caso falla en pista interna

Ejemplo 8

```
#importar librerias
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.decomposition import PCA, KernelPCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.manifold import MDS
from sklearn.manifold import Isomap
from mpl_toolkits.mplot3d import Axes3D

Pn=np.load('Pn.npy')
Po=np.load('Po.npy')
Pi=np.load('Pi.npy')
En=np.load('En.npy')
Eo=np.load('Eo.npy')
Ei=np.load('Ei.npy')

P0=np.concatenate((Pn,En),axis=1)
P1=np.concatenate((Po,Eo),axis=1)
P2=np.concatenate((Pi,Ei),axis=1)

Nt=len(Pi) #numero de datos

X=np.concatenate((P0,P1,P2),axis=0)

# estandarizar los datos
scaler = preprocessing.StandardScaler().fit(X)
Xs=scaler.transform(X)
```

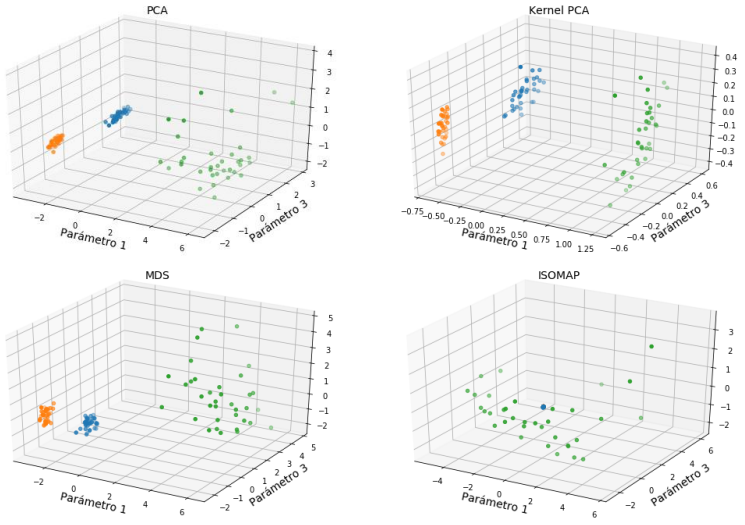
Ejemplo 8

```
#PCA
pca = PCA(n_components=3)
Xt = pca.fit_transform(Xs)

# #Kernel PCA
# kpca = KernelPCA(n_components=3, kernel="cosine") #linear, rbf, poly, sigmmoid
# Xt = kpca.fit_transform(Xs)

# #MDS
# embedding = MDS(n_components=3)
# Xt =embedding.fit_transform(Xs)

# #ISOMAP
# embedding = Isomap(n_components=3)
# Xt =embedding.fit_transform(Xs)
```


Ejemplo 8

```
fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Xt[1:Nt, 0], Xt[1:Nt, 1],Xt[1:Nt, 2])
ax.scatter(Xt[Nt+1:2*Nt, 0], Xt[Nt+1:2*Nt, 1],Xt[Nt+1:2*Nt, 2])
ax.scatter(Xt[2*Nt+1:3*Nt, 0], Xt[2*Nt+1:3*Nt, 1],Xt[2*Nt+1:3*Nt, 2])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.ylabel('Parámetro 3', fontsize=14)
plt.title('PCA', fontsize=14)
```

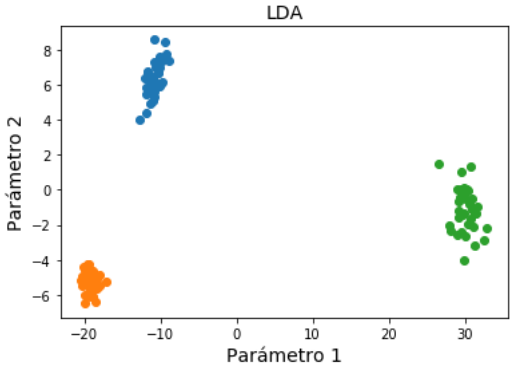
Ejemplo 8



Ejemplo 8

```
#LDA
#etiquetas
y1=np.ones(len(P0));
y2=2*np.ones(len(P1));
y3=3*np.ones(len(P2));
Y=np.concatenate((y1,y2,y3),axis=0)
lda = LinearDiscriminantAnalysis(n_components=2) #n_components < n_classes - 1
Xt = lda.fit_transform(Xs,Y)

plt.figure()
plt.scatter(Xt[1:Nt, 0], Xt[1:Nt, 1])
plt.scatter(Xt[Nt+1:2*Nt, 0], Xt[Nt+1:2*Nt, 1])
plt.scatter(Xt[2*Nt+1:3*Nt, 0], Xt[2*Nt+1:3*Nt, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.title('LDA', fontsize=14)
```

Ejemplo 8



Tarea

En la siguiente página se encuentra disponible un set de datos para un rodamiento sin falla, con falla en la pista externa, con falla en la pista interna, con falla en las bolas y con una combinación de fallas. Adicionalmente los datos consideran condiciones de operación variables:

https://data.mendeley.com/datasets/v43hmbwxpm/2

Tarea

Considere los cinco estados del rodamiento y la condición de operación: “increasing then decreasing rotational speed condition 01”.

A. Evalué las distintas opciones de extracción de parámetros vistas en clases (en el tiempo, frecuencia y tiempo-frecuencia), y concluya cuales son las más adecuadas para este caso.

Puede leer los datos con el siguiente código:

```
import scipy.io as sio

#Leer datos
Datos0=sio.loadmat('H-C-1.mat') #Healthy
Datos1=sio.loadmat('I-C-1.mat') #Inner
Datos2=sio.loadmat('O-C-1.mat') #Outer
Datos3=sio.loadmat('B-C-1.mat') #Ball
Datos4=sio.loadmat('C-C-1.mat') #Combination

Normal=Datos0['Channel_1']
Inner=Datos1['Channel_1']
Outer=Datos2['Channel_1']
Ball=Datos3['Channel_1']
Combination=Datos4['Channel_1']
```


Tarea

Considere nuevamente los datos sin procesar. Dividida cada en señal en segmentos de 5000 datos, sin overlap. Para cada segmento calcule la transformada de Fourier y guarde la amplitud en una matriz de Nsx2500 (2500 frecuencias y Ns segmentos). De acuerdo al siguiente código:

```
#importar librerías
import scipy.io as sio
# from scipy import signal
# import matplotlib.pyplot as plt
import numpy as np
# from sklearn import preprocessing
# from sklearn.decomposition import PCA, KernelPCA
# from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
# from sklearn.manifold import MDS
# from sklearn.manifold import Isomap
from scipy.fftpack import fft

#Leer datos
Datos0=sio.loadmat('H-C-1.mat') #Healthy
Datos1=sio.loadmat('I-C-1.mat') #Inner
Datos2=sio.loadmat('O-C-1.mat') #Outer
Datos3=sio.loadmat('B-C-1.mat') #Ball
Datos4=sio.loadmat('C-C-1.mat') #Combination

Normal=Datos0['Channel_1']
Inner=Datos1['Channel_1']
Outer=Datos2['Channel_1']
Ball=Datos3['Channel_1']
Combination=Datos4['Channel_1']
```

Tarea

```
#vector de tiempo
Fs=200000 #sampling rate
dt=1/Fs #paso de tiempo
Nt=len(Normal)

#calcular espectros por segmentos
L=5000 #largo de los segmentos
l=0 #overlap
Ns=int((Nt-l)/(L-l)) #total de tramos

fr=int(L/2)

P1=np.zeros((fr,Ns))
P2=np.zeros((fr,Ns))
P3=np.zeros((fr,Ns))
P4=np.zeros((fr,Ns))
P5=np.zeros((fr,Ns))
```


Tarea

```
for i in range(1,Ns+1):
    inicio=(i-1)*L-(i-1)*l+1
    fin=i*L-(i-1)*l

    x1=Normal[inicio:fin,0]
    F1 = fft(x1)[0:int(L/2)]/(L/2)
    P1[:,i-1]=abs(F1)

    x2=Inner[inicio:fin,0]
    F2 = fft(x2)[0:int(L/2)]/(L/2)
    P2[:,i-1]=abs(F2)

    x3=Outer[inicio:fin,0]
    F3 = fft(x3)[0:int(L/2)]/(L/2)
    P3[:,i-1]=abs(F3)

    x4=Ball[inicio:fin,0]
    F4 = fft(x4)[0:int(L/2)]/(L/2)
    P4[:,i-1]=abs(F4)

    x5=Combination[inicio:fin,0]
    F5 = fft(x5)[0:int(L/2)]/(L/2)
    P5[:,i-1]=abs(F5)

X=np.concatenate((P1,P2,P3,P4,P5),axis=1)
X=np.transpose(X)
```

Tarea

B. Para cada segmento tenemos 2500 parámetros. Implemente los distintos métodos de reducción de parámetros vistos en clase para obtener 3 parámetros por segmento. Grafique y discuta los resultados.